CLASIFICACIÓN DE LA CALIFICACIÓN DE DEUDA DE MOODY'S DE LOS PAÍSES MEDIANTE MODELOS DE APRENDIZAJE ESTADÍSTICO

ISADORE NABI

isadorenabi@marxiststatistics.com

Resumen:

En esta tercera entrega de *Introducción a los Modelos de Clasificación, Métodos de Validación, Métricas de Desempeño y Ensamble de Modelos*, se identificó el mejor modelo estadístico (o mejores) entre los vistos analizados en las entregas anteriores. Aquí tales modelos se utilizan para clasificar el nivel de riesgo de largo plazo en moneda local de los títulos de deuda de los países en "A lo sumo de bajo riesgo" y en "Al menos de riesgo moderado" con base en la calificación de Moody's y sus determinantes teóricos fundamentales. Se tomó una muestra de 157 observaciones para el año 2022, compuesta por cuatro variables, siendo la calificación de deuda de Moody's la variable a explicar, mientras que el diferencial de incumplimiento ajustado, la prima riesgo país y la prima riesgo accionaria las variables explicativas. Los resultados sugieren que, dada la presencia de multicolinealidad entre las variables explicativas y en ausencia de aplicación de métodos de validación, el mejor modelo individualmente considerado es el de árboles de decisión. Tales resultados se discuten en términos de la teoría estadística y de la filosofía de la ciencia.

I. INTRODUCCIÓN

¿Cuál de los modelos usualmente implementados en la clasificación/predicción, sean estos de aprendizaje supervisado o no-supervisado, paramétricos o no-paramétricos, es el más apropiado para estudiar variables económicas de tipo financiero, específicamente las relacionadas con aquellas que vuelven a los títulos de deuda soberana de a lo sumo bajo riesgo o de al menos moderado riesgo? ¿Es

posible confiar en la validez inferencial de tales resultados de clasificación/predicción y usarlos para validar teóricamente políticas económicas gubernamentales que mejoren la calificación de deuda soberana de un determinado país y con ello acceder a mejores prestaciones crediticias por parte de los inversionistas internacionales?

Partiendo de tales preguntas, esta investigación identifica, con base en la capacidad clasificatoria/predictiva y la significancia estadística, cuál entre los modelos de Redes Neuronales Artificiales, Máquinas de Vectores de Soporte, K-Vecinos Más Cercanos, Regresión Logística Multinomial (que, en este caso, se reduce a binomial dada a la reestructuración realizada sobre la respuesta), Análisis Discriminante Lineal y Árboles de Decisión es el mejor para estudiar el nivel de riesgo de un título de deuda soberana percibido por determinado grupo de expertos (cuyo criterio sirve de referencia a los inversionistas internacionales en la toma de decisiones de inversión) según los determinantes teóricos fundamentales establecidos por la teoría financiera construida desde la escuela de pensamiento económico neoclásica.

II. MATERIALES Y MÉTODOS

II.I. Objetivo General

Identificar el mejor modelo estadístico (o mejores) entre los estudiados en la primera entrega de esta investigación para clasificar el nivel de riesgo de largo plazo en moneda local de los títulos de deuda de los países en "A lo sumo de bajo riesgo" y en "Al menos de riesgo moderado" con base en la calificación de Moody's y sus determinantes teóricos fundamentales.

II.II. Tipo de Investigación

Investigación experimental.

II.III. Diseño de la Investigación

Cuantitativa

II.IV. Alcance de la Investigación

Inferencial.

II.V. Muestra

Se tomaron los diferenciales de incumplimiento ("default spreads" en la literatura en inglés), los diferenciales de incumplimiento ajustados, las primas de riesgo de las acciones (de las empresas que pertenecen a los países cuyos bonos se analizan), las primas de riesgo (de bonos) y la calificación de la deuda de largo plazo en moneda local (de bonos) hecha por Moody's de los países para el 5 de enero de 2022. Los datos se tomaron de (Damodaran, 2022). La base de datos se modificó de tal forma que los países con calificación categoría A se enlistaron con "yes", que representa que son instrumentos que tienen, a lo sumo, un bajo riesgo, mientras que a los demás se les asignó un valor de "no", lo que representa que son instrumentos que, como mínimo, poseen un riesgo medio y que en el mejor de los casos podrían poseer elementos especulativos (en el peor, son altamente especulativos -Ca- o están en situación de impago con pocas perspectivas de recuperación -C-).

II.VI. Instrumentos

Los modelos de clasificación utilizados fueron redes neuronales artificiales, máquinas de vectores de soporte, K-Vecinos Más Cercanos, Árboles de Clasificación, Regresión Logística Binomial y Análisis de Discriminante Lineal.

Los modelos antes enlistados se validaron mediante el método de retención, validación cruzada dejando uno afuera y validación cruzada de K-Pliegues.

Finalmente, se implementaron los ensambles de modelos conocidos como método de agregación bootstrap ("bagging"), bosques aleatorios ("random forests"), método de potenciación ("boosting") y método de apilamiento ("stacking"), contrastándose sus resultados con los resultados obtenidos implementando únicamente un solo modelo a la vez.

III. CONCEPTUALIZACIÓN

III.I. Diferencial de Incumplimiento y Diferencial de Incumplimiento Ajustado

Como señalan (Delianedis & Geske, 2001, pág. 4), el diferencial de crédito ("default spread" en la literatura en inglés) real u observado generalmente se mide como el rendimiento de mercado del bono menos el rendimiento del Tesoro libre de riesgo comparable[1]. El diferencial de incumplimiento, tal como se define en el presente, es una medida teórica del componente de incumplimiento del diferencial de incumplimiento.

Por su parte, como señala (Damodaran, 2022), se puede estimar una prima de riesgo país ajustada multiplicando el diferencial de crédito por la volatilidad relativa del mercado de acciones para ese mercado (es decir, por la desviación estándar en el mercado de acciones del país o por la desviación estándar en el bono del país en el mercado internacional de bonos).

III.II. Prima Riesgo

Como señala (Hayes, 2021), una prima de riesgo es el rendimiento de la inversión que se espera que rinda un activo por encima de la tasa de rendimiento libre de riesgo. La prima de riesgo de un activo es, según las finanzas basadas en los postulados teóricos neoclásicos, una forma de compensación para los inversores. Representa, según la visión antes descrita, el pago a los inversionistas por tolerar el riesgo adicional en una inversión dada sobre el de un activo libre de riesgo.

[1] Es decir, como el rendimiento que ofrece el Tesoro de los Estados Unidos, libre de riesgo, por un instrumento financiero comparable al bono comprado en el mercado.

III.III. Calificación de Deuda de Largo Plazo de Moody's

Como señala (Moody's, 2022, pág. 1), las calificaciones de deuda de Moody's se dividen en calificaciones sobre el corto plazo y calificaciones sobre el largo plazo. En esta investigación se usaron las calificaciones de deuda de largo plazo.

Así, las calificaciones de obligaciones a largo plazo[2] de Moody's son opiniones[3] sobre el riesgo crediticio relativo de las obligaciones de renta fija con un vencimiento original de un año o más. Abordan la posibilidad de que una obligación financiera no se cumpla como se prometió. Tales calificaciones reflejan la probabilidad de incumplimiento y cualquier pérdida financiera sufrida en caso de incumplimiento.

Tabla 1. Clasificación de Deuda de Largo Plazo de Moody's

Aaa	Obligations rated Aaa are judged to be of the highest quality, with minimal risk.
Aa	Obligations rated Aa are judged to be of high quality and are subject to very low credit risk.
A	Obligations rated A are considered upper-medium-grade and are subject to low credit risk.
Baa	Obligations rated Baa are subject to moderate credit risk. They are considered medium-grade and as such may possess speculative characteristics.
Ba	Obligations rated Ba are judged to have speculative elements and are subject to substantial credit risk.
B	Obligations rated B are considered speculative and are subject to high credit risk.
Caa	Obligations rated Caa are judged to be of poor standing and are subject to very high credit risk.
Ca	Obligations rated Ca are highly speculative and are likely in, or very near, default, with some prospect of recovery in principal and interest.
C	Obligations rated C are the lowest-rated class of bonds and are typically in default, with little prospect for recovery of principal and interest.

Fuente: (Moody's, 2022, pág. 1).

[2] Las calificaciones de corto plazo de Moody's, a diferencia de las calificaciones de largo plazo, se aplican a la capacidad de un emisor individual para pagar todas las obligaciones a corto plazo y no a programas específicos de endeudamiento a corto plazo. Una vez asignada a un emisor, una calificación de corto plazo tiene un alcance global: se aplica a todas las obligaciones senior no garantizadas del emisor con un vencimiento original inferior a un año, independientemente de la moneda o el mercado en el que se emitan las obligaciones. Una excepción a la naturaleza global de estas calificaciones ocurre si la calificación de un emisor está respaldada por otra entidad a través de vehículos tales como una carta de crédito o garantía.

[3] La fuente no especifica de quiénes.

III.IV. Modelos de Clasificación

III.IV. I. Modelo de Regresión Logística Binomial

Como señalan (Montgomery, Peck, & Vining, 2012, pág. 424), generalmente, cuando la variable de respuesta es binaria, existe una considerable evidencia empírica que indica que la forma de la función de respuesta debe ser no lineal. Por lo general, se emplea una función en forma de S (o en forma de S inversa) monótonamente creciente (o decreciente)

Esta función se llama función de respuesta logística y tiene la siguiente forma:

$$E(y) = \frac{e^{(x'\beta)}}{1 + e^{(x'\beta)}} = \frac{1}{1 + e^{(-x'\beta)}}$$

En la ecuación anterior, $x' = (1, x_{i1}, x_{i2}, \dots, x_{ik})$, $\beta = (\beta_0, \beta_1, \beta_2, \dots, \beta_k)$ y y es una variable aleatoria que, como se adelantó, sigue una distribución de Bernoulli cuyas probabilidades están modeladas según lo establecido en la figura 5.

Como señala (Montgomery, Peck, & Vining, 2012, pág. 424), la función de respuesta logística se puede linealizar fácilmente. Desde la perspectiva de los modelos lineales generalizados descubiertos por Nelder y Wedderburn (1972), se define la parte estructural del modelo en términos de una función de la media de la función de respuesta, conocida como *función enlace* (McCullagh & Nelder, 1989, pág. 31). Así, sea el predictor lineal $\eta = x'\beta$, puede realizarse sobre el mismo una transformación de manera tal que

$$\eta = \log\left\{\frac{\pi}{1-\pi}\right\} = \log\left\{\frac{P(x)}{1-P(x)}\right\} \qquad (7)$$

La transformación anterior es a menudo llamada *transformación logit* de la probabilidad π y la razón $\frac{\pi}{1-\pi}$ es a menudo llamada *odds*, puesto que $\pi = P(x)$, mientras que $1 - \pi = 1 - P(x)$. A veces, por la misma razón, la transformación logit es llamada *log-odds*. La razón $\frac{\pi}{1-\pi}$ generalmente se interpreta como la razón de

ventaja que tiene una categoría[4]. A continuación, se presenta la gráfica de la función teórica logit.

III.IV. II. Análisis de Discriminante Lineal

Como señala (Zelterman, 2015, pág. 257), si el investigador dispone de observaciones multivariadas de dos o más poblaciones identificadas, ¿cómo puede caracterizarlas? ¿Existe una combinación de medidas[5] que se pueda usar para distinguir claramente entre estos grupos? No basta con decir simplemente que la media de una variable es estadísticamente más alta en un grupo para resolver este problema porque los histogramas de los grupos pueden tener una superposición considerable, lo que hace que el proceso discriminatorio sea solo un poco mejor que las conjeturas. Para pensar en términos multivariados, no se usa solo una variable a la vez para distinguir entre grupos de individuos, sino que se usa una combinación de variables explicativas. La diferencia entre el análisis discriminante y el análisis de componentes principales es que, en el análisis discriminante, el punto de partida es conocer la pertenencia al grupo. En el análisis discriminante se busca identificar combinaciones lineales de varias variables que se pueden usar para distinguir entre los grupos. En PCA, por el contrario, no está claro si hay grupos separados. Si los componentes principales identifican grupos, entonces no hay un paso de confirmación porque es posible que no se tenga conocimiento apriorístico de la separación de los grupos.

Así, con base en (Zelterman, 2015, págs. 265-267), el análisis discriminante lineal (LDA, por su nombre en inglés) se trata de separar las observaciones de cada población mediante combinaciones lineales de un grupo de variables importantes que permita diferenciarlos. En general, a diferencia de la regresión logística, el análisis de discriminante linear prioriza la clasificación, y en modelos que tengan muchas variables podría llegar a ser más estable. Para el caso de los modelos lineales generalizados (incluidos los modelos de regresión) tiene varios objetivos

[4] Lo que deba entenderse por la categoría dependerá del diseño experimental concreto.
[5] Variables relevantes.

además de la clasificación. En grupos pequeños podría llegar a tener resultados similares a otras técnicas como la regresión logística

Al realizar LDA se recomienda estandarizar los datos, ya que las unidades de medida heterogéneas pueden afectar el análisis. Además, a diferencia de la regresión logística que no asume nada con relación a las variables explicativas, en LDA se asume normalidad para todas las variables y sus combinaciones lineales, aunque con diferentes medias, así como también que todas las clases/categorías de la variable de respuesta comparten la misma matriz de varianzas-covarianzas.

Como señala (Amat Rodrigo, 2016) desde la perspectiva de la Teoría del Aprendizaje Estadístico[6] el LDA es un método de clasificación supervisado de variables cualitativas en el que dos o más grupos son conocidos a priori y nuevas observaciones se clasifican en uno de ellos en función de sus características.

III.IV. III. Árboles de Clasificación

Como señalan (Hastie, Tibshirani, & Friedman, 2016, pág. 305), los métodos basados en árboles dividen el espacio de características en un conjunto de rectángulos y luego ajustan un modelo simple (como una constante) en cada uno. Son conceptualmente simples pero poderosos.

Como señala (Ganegedara, 2018), un árbol de decisión es simplemente un conjunto de preguntas en cascada. Cuando obtiene un punto de datos (es decir, un conjunto de características y valores), utiliza cada atributo (es decir, un valor de una característica dada del punto de datos) para responder una pregunta. La respuesta a cada pregunta decide la siguiente pregunta. Al final de esta secuencia de preguntas, obtendrá una probabilidad de que el punto de datos pertenezca a cada clase.

Como señala (Dale, 2020), los árboles de decisión son uno de los conceptos más importantes en el aprendizaje automático moderno. No solo son un enfoque eficaz

[6] Esta es una visión que consiste en una actualización de la Estadística Clásica desde los nuevos descubrimientos provistos por la Ciencia de Datos, el Aprendizaje Automático y el Aprendizaje Profundo.

para los problemas de clasificación y regresión, sino que también son la base para algoritmos más sofisticados, como bosques aleatorios y aumento de gradiente.

III.IV. IV. K-Vecinos Más Cercanos

Como se señala en (IBM, 2022), el algoritmo de k-vecinos más cercanos, también conocido como KNN o k-NN, es un clasificador de aprendizaje supervisado no paramétrico, que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de una observación individualmente considerada. Si bien se puede usar para problemas de regresión o clasificación, generalmente se usa como un algoritmo de clasificación, partiendo de la suposición de que se pueden encontrar puntos similares cerca uno del otro. Así, esta técnica parte de los principios de aglomeración del método de agrupación K-medias por cuanto, como señala (Christopher, 2021), intenta predecir la clase correcta para cada observación perteneciente a los datos de prueba ("test data" en la literatura en inglés) calculando la distancia (euclidiana o de otro tipo) entre los datos de prueba (los que se desea clasificar en una u otra clase) y los datos de entrenamiento (los datos que conforman las distintas clases[7], "training data" en la literatura en inglés, que son los datos prior o apriorísticos, en términos de estadística bayesiana), para posteriormente seleccionar el número de K puntos[8] (número k de observaciones, cada una perteneciente a alguna de las clases de los datos de entrenamiento) que está más cerca de cada una de las observaciones de los datos de prueba.

III.IV. V. Máquinas de Vectores de Soporte

Una subregión (o subconjunto[9]) de un espacio (conjunto) es convexa si, dados dos elementos dentro de sí (dentro de la subregión), es posible generar una trayectoria lineal que los una y que esté contenida dentro de dicha subregión.

---

[7] Por ejemplo, para la base de datos IRIS las K distintas clases son setosa, versicolor y virginica.

[8] Donde K es una constante no nula que pertenece a los naturales y que es especificada por el investigador con base en aspectos objetivos como el criterio experto, el estado del arte (teórico y/o aplicado) de la técnica, las necesidades específicas de la investigación y/u otro aspecto vinculado.

[9] Formalmente lo que se denomina en Teoría del Aprendizaje Estadístico (el fundamento para el Aprendizaje Automático y el Aprendizaje Profundo basado en el marco teórico y metodológico de la Estadística

Para clasificar adecuadamente diferentes observaciones dentro de un determinado espacio muestral, se requiere construir con ellas subregiones (subconjuntos) convexas disjuntas, es decir, subregiones (subconjuntos) convexas cuya intersección es vacía, lo implica que no tienen elementos en común. En el contexto de espacios muestrales $n-dimensionales$ esto se conoce como *problema del hiperplano de separación*.

Vladimir N. Vapnik planteó en 1996 un algoritmo que encuentra un hiperplano separador óptimo (Hastie, Tibshirani, & Friedman, 2016, pág. 102). Un *hiperplano separador óptimo* es, como señalan (Hastie, Tibshirani, & Friedman, 2016, pág. 132), aquel que separa dos clases y maximiza la distancia al punto más cercano para cada clase. Esto no solo brinda una solución única al problema del hiperplano de separación, sino que, al maximizar el margen entre las dos clases en los datos de entrenamiento, esto conduce a un mejor rendimiento de clasificación en los datos de prueba.

Como señala (Vapnik, 2000, pág. 138), una máquina de vectores de soporte (SV) implementa la siguiente idea: mapea[10] los vectores de entrada x en un espacio de características de alta dimensión Z a través de un mapeo no lineal, elegido a priori[11].

III.IV. VI. Redes Neuronales Artificiales

La idea central es del modelo de redes neuronales artificiales es extraer combinaciones lineales de las entradas (variables explicativas) como características derivadas y luego modelar el objetivo (la variable a explicar) como una función no-lineal de estas características. El resultado es un poderoso método de aprendizaje, con amplias aplicaciones en muchos campos (Hastie, Tibshirani, & Friedman, 2016, pág. 389). Como señalan los autores referidos en el mismo lugar, se trata de un

---

Matemática) como "clase", "grupo" o "conjunto" es en realidad un conjunto, puesto que desde la Teoría de Conjuntos el término "clase" tiene un significado más general.

[10] Aplica sobre los vectores de entrada alguna operación definida mediante algún operador. Como se ve más adelante, estos operadores y las operaciones implicadas en ellos son no-lineales.

[11] Antes de realizar el mapeo.

modelo de aprendizaje supervisado que emplea un modelo aditivo (*i.e.*, un modelo de regresión no-paramétrico), pero es aditivo únicamente en las combinaciones lineales generadas con las variables explicativas originales, no propiamente respecto a tales variables originales. El modelo de regresión de búsqueda de proyección[12] (PPR, por su nombre en inglés) toma la siguiente forma:

$$f(X) = \sum_{m=1}^{M} g_m(\omega_m^T X) \qquad (11)$$

En la expresión anterior, X es el vector de entradas (variables explicativas) con $p - ésimos$ componentes con un objetivo Y. Adicionalmente, ω_m, con $m = 1, 2, \dots, M$ (donde M es un hiperparámetro), es la estructura que contiene a los $M - ésimos$ vectores unitarios de longitud p (*i.e.*, de p elementos) de parámetros desconocidos (tales parámetros son las ponderaciones de las conexiones entre neuronas en la red neuronal artificial -*i.e.*, el peso específico de cada una de ellas en la red-, que se encuentran a partir del conjunto de datos usando algún método numérico de suavizado flexible[13] buscando minimizar la función de error), siendo ω_m^T su versión

[12] Es una generalización de los modelos aditivos. El mecanismo empleado por este modelo consiste en proyectar primero la matriz de datos de las variables explicativas en la dirección óptima antes de aplicar funciones de suavizado a estas variables explicativas. El modelo, como se verá a continuación, es una combinación lineal de funciones de cresta ("ridge functions" en la literatura en inglés).

[13] Como se señala en la discusión (stack overflow, 2014), la flexibilidad de un modelo describe su capacidad de aumentar los grados de libertad disponibles para ajustarse a los datos de entrenamiento. En la misma línea, se señala en que (Johnson, 2022), que un modelo tiene que caminar por una delgada línea entre el aprendizaje específico de un concepto y su aplicación general. La "flexibilidad" de un modelo determina este equilibrio. La flexibilidad de un modelo se puede describir como cuánto está influenciado el comportamiento del modelo por las características de los datos; por lo tanto, la flexibilidad o inflexibilidad de un modelo es una característica que no debe pasarse por alto. Puede significar la diferencia entre una herramienta útil o inútil. Adicionalmente, como se señala en la discusión (Cross Validated, 2013), cuando la cantidad de predictores p es extremadamente grande y el número de observaciones n es pequeña relación entre X y Y es lineal o muy semejante a una relación lineal, un modelo flexible (como el de splines) generará sobreajuste, por lo que es mejor utilizar un modelo inflexible (como la regresión lineal, debido a que el error de prueba será mejor). Lo mismo ocurre, dada una relación lineal o muy parecida a ella entre X y Y, si la varianza del término de error es extremadamente elevada. En el caso de que la relación sea altamente no lineal, determinar apriorísticamente si es mejor usar un modelo flexible o inflexible no es posible y dependerá del fenómeno social o natural analizado (que implica el marco teórico de la ciencia aplicada que describa sus relaciones fundamentales y no fundamentales, así como el conjunto de datos en el que se cristalizan las mediciones realizadas sobre el fenómeno en estudiado).

transpuesta; se busca el ω_m tal que el modelo ajuste bien[14], de ahí el nombre "búsqueda de proyección", dado que las ponderaciones en ω_m son conocidas también como las direcciones de la red neuronal artificial. Por su parte, $V_m = \omega_m^T X$ son las variables escalares generadas con las combinaciones lineales de las variables explicativas o, dicho de otra forma, son las proyecciones de X sobre el vector unitario w_m. Finalmente, $g_m(\omega_m^T X)$ son funciones univariantes[15] desconocidas (denominadas *funciones de activación tipo cresta o tipo lineal*[16]) y son estimadas también, junto con las ponderaciones/direcciones ω_m, usando algún método numérico de suavizado flexible y pertenecen a un espacio $\mathbb{R}^p$. Las funciones de cresta no son susceptibles a la maldición de dimensionalidad[17], por lo que la convierte en una herramienta instrumental en varios problemas de estimación[18]. Son fundamentales para analizar fenómenos sobre los cuales sólo se dispone de información parcial (Bojanov, 1992, pág. 371).

III.V. Métodos de Validación

III.V. I. Generalidades

Para evaluar los modelos se suele utilizar la totalidad del conjunto de datos usado para generar el modelo, así como también para calcular las medidas de precisión posteriormente. El problema inherente a ello es que de tal manera la estimación de la precisión y del error posee un sesgo relevante, específicamente uno en el que la

---

[14] Específicamente, como señala (Vapnik, 2000, pág. 125), por "ajustar bien" debe entenderse que permita minimizar el funcional (función de funciones) de riesgo empírico $R_{emp}(\omega) = \frac{1}{l} \sum_{j=1}^{l} \left(y_j - f(x_j, \omega)\right)^2$, donde, en términos de la identidad (11), l equivale a M, j a m, $f(x_j, \omega)$ equivale a $g_m(\omega_m^T X)$ y y_j a $f(X)$ o, más específicamente, y_j a $f(x_m)$, donde y_j a $x_m \in X$.

[15] Véase (Bojanov, 1992, pág. 371).

[16] La función de activación de un nodo (aquí "nodo" tiene el mismo sentido que en la sección III.III) define la salida de ese nodo dada una entrada o un conjunto de entradas (las variables explicativas). Las funciones de cresta son un tipo de funciones de activación, un tipo de activación de carácter lineal.

[17] Véase (Nabi, 2021, pág. 2).

[18] Esto es lo que se señala en la discusión de Wikipedia relativa a la entrada "Ridge function", sin embargo, en (Zaitseva, Malykhin, & Ryutin, 2021, págs. 1-2), (Doerra & Mayerb, 2020, pág. 1), (Bastounis & Hansen, 2017, pág. 356) y (Zhang, Gan, Ling, & Sun, 2017, pág. 8) se señalan casos en que las redes y otros modelos que empleen funciones de cresta no están exentos del problema de recuperación (la forma fundamental de la maldición de dimensionalidad).

estimación del error indica una magnitud inferior de este con relación a su magnitud real. A esto se le conoce como *subestimación del error*.

III.V. III. Validación Cruzada por el Método de K-Pliegues

(Lantz, 2013, pág. 319) señala que la retención repetida es la base de una técnica conocida como validación cruzada k-fold (o k-fold CV), que se ha convertido en el estándar de la industria para estimar el rendimiento del modelo. Pero en lugar de tomar muestras aleatorias repetidas que potencialmente podrían usar el mismo registro más de una vez, k-fold CV divide aleatoriamente los datos en k particiones aleatorias completamente separadas llamadas pliegues.

Aunque k se puede establecer en cualquier número, con mucho, la convención más común es utilizar una validación cruzada de 10 veces (CV de 10 veces). ¿Por qué 10 pliegues? La evidencia empírica sugiere que hay poco beneficio adicional al usar un número mayor. Para cada uno de los 10 pliegues (cada uno comprende el 10 por ciento de los datos totales), se construye un modelo de aprendizaje automático sobre el 90 por ciento restante de los datos. La muestra coincidente del 10 por ciento del pliegue se usa luego para la evaluación del modelo. Después de que el proceso de entrenamiento y evaluación del modelo haya ocurrido 10 veces (con 10 combinaciones diferentes de entrenamiento/prueba), se informa el rendimiento promedio en todos los pliegues.

III.VI. Ensambles de Modelos

Como señala (Lantz, 2013, pág. 337), como alternativa a aumentar el rendimiento de un solo modelo, es posible combinar varios modelos para formar un equipo poderoso. Así como los mejores equipos deportivos tienen jugadores con conjuntos de habilidades complementarios en lugar de superpuestos, algunos de los mejores algoritmos de aprendizaje automático utilizan equipos de modelos complementarios. Debido a que un modelo aporta un sesgo único a una tarea de aprendizaje, puede aprender fácilmente un subconjunto de ejemplos, pero tener problemas con otro. Por lo tanto, mediante el uso inteligente de los talentos de

varios miembros diversos del equipo, es posible crear un equipo fuerte de múltiples aprendices[19] débiles. Esta técnica de combinar y administrar las predicciones de múltiples modelos se encuentra dentro de un conjunto más amplio de *métodos de meta-aprendizaje* que abarcan ampliamente cualquier técnica que implique aprender a aprender. Esto puede incluir cualquier cosa, desde algoritmos simples que mejoran gradualmente el rendimiento mediante la iteración automática de las decisiones de diseño (por ejemplo, el ajuste de parámetros automatizado que se usó anteriormente en este capítulo) hasta algoritmos altamente complejos que usan conceptos prestados de la biología evolutiva y la genética para automodificarse y adaptarse a las tareas de aprendizaje. Muchas de las técnicas basadas en el trabajo en equipo son bastante poderosas y se usan con bastante frecuencia para construir clasificadores más efectivos.

El enfoque de meta-aprendizaje que utiliza, en términos conceptuales, un principio similar al de crear un equipo variado de expertos, conocido en la teoría del aprendizaje estadístico como *ensamble*. Todos los métodos de ensamblaje se basan en la idea de que, al combinar varios métodos de aprendizaje más débiles, se crea un método de aprendizaje más fuerte.

IV. RESULTADOS

IV.I. Resultados realizando una partición del archivo de datos usando una validación K-Pliegues con K=10 para los modelos de redes neuronales artificiales, regresión logística multinomial, árboles de clasificación, K-Vecinos Más Cercanos, Análisis Discriminante y Máquinas de Vectores de Soporte

1. Selección de datos

```
library(readxl)
COUNTRIES <- read_excel("C:/Users/Usuario/Desktop/COUNTRIES.xlsx")
COUNTRIES$MoodysRat <- as.factor(COUNTRIES$MoodysRat)
View(COUNTRIES)

set.seed(10)  #se fija una semilla, para que sea la misma muestra aleatoria. Se fija un valor
```

[19] En el documento original “learners”.

```
de 10 arbitrariamente
entre=sample(1:157, 117)  #se trabaja, según (Lantz, 2013, pág. 220), con aproximdamente el 75% de las observaciones, que equivale aproximadamente a 117 observaciones. Así, se le indica al algoritmo que extraiga de esa muestra aleatoriamente 117 elementos a usar para entrenamiento.
datos.entre=COUNTRIES[entre,] #definiendo la estructura de datos que contendrá los datos de entrenamiento
datos.vali=COUNTRIES[-entre,] #definiendo la estructura de datos que contendrá los datos de prueba
```

2. **Realizando una partición del archivo de datos usando una validación K-Pliegues con K=10**

2.1. Redes Neuronales Artificiales

2.1.1. Construcción del Algoritmo de Validación Requerido

```
n=dim(COUNTRIES)[1]

error_ind <- 0
error_kfold <- 0
prop_correctos_vector <- 0

for(i in 1:10){

 error_ind <- 0
 folds <- createFolds(1:n, k = 10)

 for(j in 1:10){
  datos.entre=COUNTRIES[-folds[[j]],]
  datos.vali=COUNTRIES[folds[[j]],]

  #Modelo de ANN
  modelo_redes <- nnet(MoodysRat~ ., data = datos.entre, size = 4, rang = 0.1, decay = 5e-04,
                       maxit = 200, trace = FALSE)
  prediccion_redes <- predict(modelo_redes, type = "raw", newdata = datos.vali[,-1])
  (matriz_confusion_redes=table(prediccion_redes, datos.vali$MoodysRat,dnn=c("Predicho","Real")))
  prop_correctos <- (matriz_confusion_redes[1,1]+matriz_confusion_redes[2,2])/sum(matriz_confusion_redes) # proporcion de correctos
  prop_incorrectos <- 1 - prop_correctos # proporcion de incorrectos
```

```
   (correcto_redes <-
(sum(diag(matriz_confusion_redes)))/sum(matriz_confusion_redes))
   (incorrecto_redes<- 1 - correcto_redes )

   temp = incorrecto_redes
   error_ind <- error_ind + temp # suma de los errores del modelo construido con los 10
folds
  }

  prop_correctos_vector <- c(prop_correctos_vector, prop_correctos)
  temp_vector <- error_ind/10
  error_kfold <- c(error_kfold, temp_vector)

}

prop_correctos_vector <- prop_correctos_vector[-1]
promedio_correctos_redes <- mean(prop_correctos_vector) # proporcion promedio de
correctos de validacion
error_kfold <- error_kfold[-1]

print(c("promedio_correctos_redes", promedio_correctos_redes))

## [1] "promedio_correctos_redes" "0.524583333333333"
```

2.1.2. Error Obtenido en Cada Iteración por el Método de Validación Cruzada de K-Pliegues

```
error_kfold

##  [1] 0.4950000 0.4979167 0.4958333 0.4962500 0.4958333 0.5012500 0.4945833
##  [8] 0.4841667 0.4667857 0.4954167
```

2.1.3. Graficando el Error Estimado por el Método de Validación Cruzada de K-Pliegues

```
#Modelo de ANN Entrenamiento
modelo_redes_entre <- nnet(MoodysRat ~ ., data = datos.entre, size = 4, rang = 0.1,
decay = 5e-04,
                maxit = 200, trace = FALSE)
prediccion_redes_entre <- predict(modelo_redes_entre, type = "raw", newdata =
datos.entre[,-1])
(matriz_confusion_redes_entre=table(prediccion_redes_entre,
datos.entre$MoodysRat,dnn=c("Predicho","Real")))

##                 Real
## Predicho          no yes
```

```
##   0                71   0
##   7.14073589740826e-06  5   0
##   0.000141348165026535  4   0
##   0.00344285175297536  11   0
##   0.089376976885403     6   0
##   0.890771340447213     0   5
##   0.997489054995817     0   6
##   0.999445288267337     0   7
##   0.999810410972553     0   9
##   0.999938693166107     0   4
##   0.999977423209386     0   2
##   0.999999472282071     0  12
```

```
prop_correctos_redes_entre <-
(matriz_confusion_redes_entre[1,1]+matriz_confusion_redes_entre[2,2])/sum(mat
riz_confusion_redes_entre) # proporcion de correctos
```

```
#Modelo de ANN total
modelo_redes_total <- nnet(MoodysRat ~ ., data = datos.entre, size = 4, rang = 0.1,
decay = 5e-04,
                maxit = 200, trace = FALSE)
prediccion_redes_total <- predict(modelo_redes_total, type = "raw", newdata =
COUNTRIES[,-1])
(matriz_confusion_redes_total=table(prediccion_redes_total,
COUNTRIES$MoodysRat,dnn=c("Predicho","Real")))
```

```
##                   Real
## Predicho             no yes
##   0                 79   0
##   6.22727741426906e-06  5   0
##   0.000132896450382249  6   0
##   0.00336790745407524  12   0
##   0.0885101150424791    6   0
##   0.889363658921267     0   5
##   0.997472890156113     0   7
##   0.999448444921449     0   7
##   0.999813945804787     0  10
##   0.999940941241383     0   4
##   0.999978702927519     0   2
##   0.999999560948914     0  14
```

```
prop_correctos_redes_total <-
(matriz_confusion_redes_total[1,1]+matriz_confusion_redes_total[2,2])/sum(matri
z_confusion_redes_total) # proporcion de correctos
```

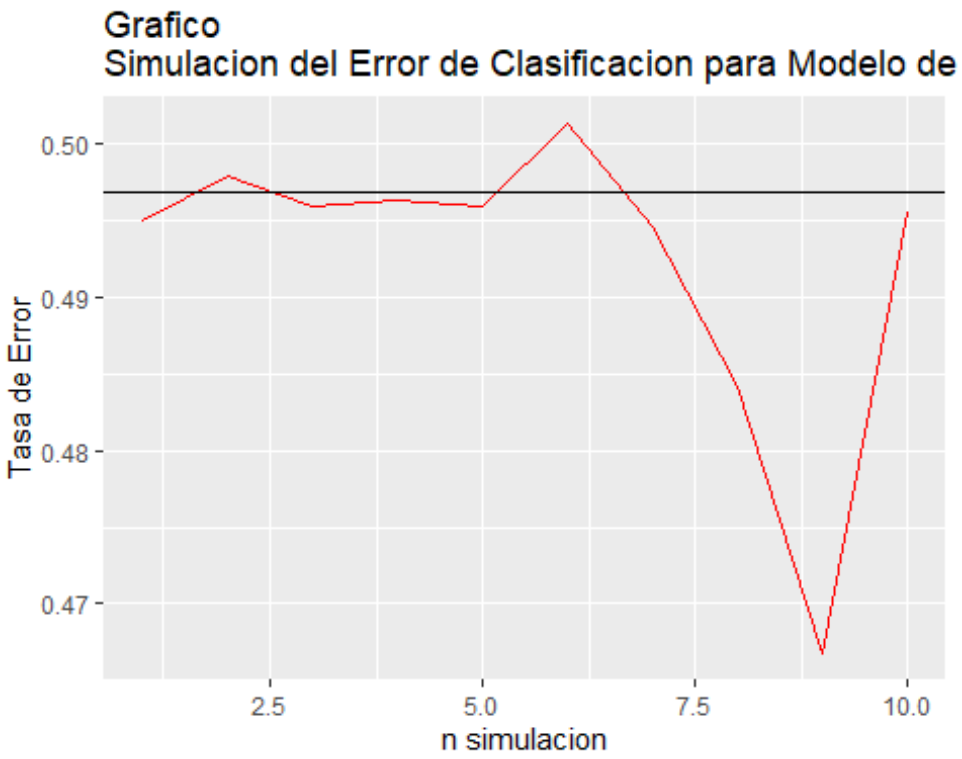
```
(correcto_redes_total <-
(sum(diag(matriz_confusion_redes_total)))/sum(matriz_confusion_redes_total))
```

```
## [1] 0.5031847
```

```
(incorrecto_redes_total<- 1 - correcto_redes_total )
```

```
## [1] 0.4968153
```

```
dat_error=data.frame(error_kfold, n=1:10)
ggplot(data = dat_error,mapping = aes(x = n, y =error_kfold )) +
  geom_line(color = "red")+
  geom_hline(yintercept=incorrecto_redes_total)+
  labs(title = "Grafico
Simulacion del Error de Clasificacion para Modelo de ANN ",
       x = "n simulacion",
       y = "Tasa de Error")
```



2.2. Regresión Logística Multinomial

2.2.1. Construcción del Algoritmo de Validación Requerido

```
n=dim(COUNTRIES)[1]

error_ind <- 0
```

```
error_kfold <- 0
prop_correctos_vector <- 0

for(i in 1:10){

 error_ind <- 0
 folds <- createFolds(1:n, k = 10)

 for(j in 1:10){
  datos.entre=COUNTRIES[-folds[[j]],]
  datos.vali=COUNTRIES[folds[[j]],]

  #Modelo de Regresión Logística Multinomial
  mod_multi <- nnet::multinom(MoodysRat~., data = datos.entre, maxit = 100)
  prediccion_multi=predict(mod_multi,newdata = datos.vali[,-1])
  (matriz_confusion_multi=table(prediccion_multi,
                    datos.vali$MoodysRat,dnn=c("Predicho","Real")))
  prop_correctos <-
(matriz_confusion_multi[1,1]+matriz_confusion_multi[2,2])/sum(matriz_confusio
n_multi) # proporcion de correctos
  prop_incorrectos <- 1 - prop_correctos # proporcion de incorrectos

  (correcto_multi_vali <-
(sum(diag(matriz_confusion_multi)))/sum(matriz_confusion_multi))
  (incorrecto_multi_vali<- 1 - correcto_multi_vali )

  temp = incorrecto_multi_vali
  error_ind <- error_ind + temp # suma de los errores del modelo construido con los 10
folds
 }

 prop_correctos_vector <- c(prop_correctos_vector, prop_correctos)
 temp_vector <- error_ind/10
 error_kfold <- c(error_kfold, temp_vector)

}
```

```
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.762954
## iter  20 value 3.930374
## iter  30 value 3.444563
## iter  40 value 0.280249
## iter  50 value 0.279104
```

```
## iter  60 value 0.196510
## iter  70 value 0.195228
## iter  80 value 0.161941
## iter  90 value 0.152313
## iter 100 value 0.122171
## final  value 0.122171
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.905315
## iter  20 value 3.891299
## iter  30 value 3.316646
## iter  40 value 1.080331
## iter  50 value 1.018248
## iter  60 value 0.764486
## iter  70 value 0.745411
## iter  80 value 0.490897
## iter  90 value 0.440208
## iter 100 value 0.311578
## final  value 0.311578
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.341321
## iter  20 value 3.843037
## iter  30 value 3.322576
## iter  40 value 1.175470
## iter  50 value 1.147548
## iter  60 value 0.263566
## iter  70 value 0.260555
## iter  80 value 0.227225
## iter  90 value 0.221305
## iter 100 value 0.204378
## final  value 0.204378
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.759749
## iter  20 value 4.134998
## iter  30 value 3.609540
## iter  40 value 1.495702
## iter  50 value 1.458717
## iter  60 value 1.064217
```

```
## iter  70 value 1.013042
## iter  80 value 0.867765
## iter  90 value 0.806691
## iter 100 value 0.649792
## final  value 0.649792
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.288488
## iter  20 value 4.020855
## iter  30 value 3.460346
## iter  40 value 0.440511
## iter  50 value 0.439008
## iter  60 value 0.350622
## iter  70 value 0.345607
## iter  80 value 0.330641
## iter  90 value 0.318238
## iter 100 value 0.279018
## final  value 0.279018
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.440360
## iter  20 value 4.056750
## iter  30 value 3.508853
## iter  40 value 0.603328
## iter  50 value 0.596840
## iter  60 value 0.418338
## iter  70 value 0.411542
## iter  80 value 0.333067
## iter  90 value 0.322020
## iter 100 value 0.292330
## final  value 0.292330
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.606712
## iter  20 value 3.462067
## iter  30 value 2.840206
## iter  40 value 0.447522
## iter  50 value 0.445129
## iter  60 value 0.364390
## iter  70 value 0.359685
```

```
## iter  80 value 0.342159
## iter  90 value 0.332738
## iter 100 value 0.305332
## final  value 0.305332
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.428014
## iter  20 value 4.348403
## iter  30 value 3.829721
## iter  40 value 0.672314
## iter  50 value 0.658355
## iter  60 value 0.463080
## iter  70 value 0.453828
## iter  80 value 0.353005
## iter  90 value 0.333955
## iter 100 value 0.278575
## final  value 0.278575
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.487477
## iter  20 value 3.770882
## iter  30 value 3.177340
## iter  40 value 0.084398
## iter  50 value 0.073000
## iter  60 value 0.072795
## iter  70 value 0.061182
## iter  80 value 0.060577
## iter  90 value 0.054011
## iter 100 value 0.049692
## final  value 0.049692
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.774848
## iter  20 value 3.930163
## iter  30 value 3.432730
## iter  40 value 0.553947
## iter  50 value 0.551253
## iter  60 value 0.408369
## iter  70 value 0.401166
## iter  80 value 0.317176
```

```
## iter  90 value 0.300069
## iter 100 value 0.249399
## final  value 0.249399
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.792111
## iter  20 value 4.178572
## iter  30 value 3.676680
## iter  40 value 0.597129
## iter  50 value 0.591933
## iter  60 value 0.393832
## iter  70 value 0.387045
## iter  80 value 0.261199
## iter  90 value 0.248572
## iter 100 value 0.206783
## final  value 0.206783
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.315953
## iter  20 value 3.432518
## iter  30 value 2.847223
## iter  40 value 0.188874
## iter  50 value 0.188209
## iter  60 value 0.135707
## iter  70 value 0.135043
## iter  80 value 0.116909
## iter  90 value 0.111267
## iter 100 value 0.093013
## final  value 0.093013
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.781422
## iter  20 value 4.028662
## iter  30 value 3.483723
## iter  40 value 0.375850
## iter  50 value 0.374929
## iter  60 value 0.299950
## iter  70 value 0.296242
## iter  80 value 0.257313
## iter  90 value 0.249837
```

```
## iter 100 value 0.228585
## final  value 0.228585
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.455902
## iter  20 value 3.755529
## iter  30 value 3.195678
## iter  40 value 0.949987
## iter  50 value 0.938061
## iter  60 value 0.834490
## iter  70 value 0.753750
## iter  80 value 0.711772
## iter  90 value 0.437112
## iter 100 value 0.342485
## final  value 0.342485
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.228385
## iter  20 value 3.667657
## iter  30 value 3.099531
## iter  40 value 0.514858
## iter  50 value 0.511754
## iter  60 value 0.405712
## iter  70 value 0.399271
## iter  80 value 0.379634
## iter  90 value 0.365535
## iter 100 value 0.322348
## final  value 0.322348
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.107179
## iter  20 value 4.618775
## iter  30 value 3.971342
## iter  40 value 1.708401
## iter  50 value 1.642030
## iter  60 value 1.174841
## iter  70 value 1.129266
## iter  80 value 1.048660
## iter  90 value 0.931459
## iter 100 value 0.764979
```

```
## final  value 0.764979
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.580845
## iter  20 value 4.011254
## iter  30 value 3.463044
## iter  40 value 1.796919
## iter  50 value 1.738407
## iter  60 value 0.698023
## iter  70 value 0.669955
## iter  80 value 0.632589
## iter  90 value 0.606048
## iter 100 value 0.567358
## final  value 0.567358
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.189259
## iter  20 value 3.901468
## iter  30 value 3.362048
## iter  40 value 0.684654
## iter  50 value 0.669734
## iter  60 value 0.297628
## iter  70 value 0.293665
## iter  80 value 0.206389
## iter  90 value 0.197704
## iter 100 value 0.168193
## final  value 0.168193
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.265485
## iter  20 value 3.960250
## iter  30 value 3.415008
## iter  40 value 0.362134
## iter  50 value 0.361260
## iter  60 value 0.283162
## iter  70 value 0.277874
## iter  80 value 0.180624
## iter  90 value 0.173093
## iter 100 value 0.147491
## final  value 0.147491
```

```
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.000103
## iter  20 value 3.950523
## iter  30 value 3.385934
## iter  40 value 0.158357
## iter  50 value 0.158089
## iter  60 value 0.129719
## iter  70 value 0.128670
## iter  80 value 0.103722
## iter  90 value 0.100237
## iter 100 value 0.088034
## final  value 0.088034
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.382511
## iter  20 value 4.129875
## iter  30 value 3.643903
## iter  40 value 0.758128
## iter  50 value 0.730125
## iter  60 value 0.595017
## iter  70 value 0.580915
## iter  80 value 0.383101
## iter  90 value 0.350951
## iter 100 value 0.259036
## final  value 0.259036
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.335757
## iter  20 value 3.820398
## iter  30 value 3.294121
## iter  40 value 0.842802
## iter  50 value 0.813900
## iter  60 value 0.643281
## iter  70 value 0.629570
## iter  80 value 0.448294
## iter  90 value 0.428067
## iter 100 value 0.373812
## final  value 0.373812
## stopped after 100 iterations
```

```
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.385823
## iter  20 value 4.133901
## iter  30 value 3.592124
## iter  40 value 0.439467
## iter  50 value 0.438321
## iter  60 value 0.353502
## iter  70 value 0.348324
## iter  80 value 0.311893
## iter  90 value 0.295128
## iter 100 value 0.244380
## final  value 0.244380
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.090806
## iter  20 value 3.843665
## iter  30 value 3.257764
## iter  40 value 1.207675
## iter  50 value 1.188545
## iter  60 value 0.966133
## iter  70 value 0.940799
## iter  80 value 0.740851
## iter  90 value 0.667096
## iter 100 value 0.477007
## final  value 0.477007
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.233511
## iter  20 value 3.871035
## iter  30 value 3.314504
## iter  40 value 0.454219
## iter  50 value 0.452674
## iter  60 value 0.362878
## iter  70 value 0.357570
## iter  80 value 0.341516
## iter  90 value 0.328577
## iter 100 value 0.287823
## final  value 0.287823
## stopped after 100 iterations
## # weights:  5 (4 variable)
```

```
## initial  value 97.733752
## iter  10 value 11.289509
## iter  20 value 4.370617
## iter  30 value 3.857177
## iter  40 value 0.607649
## iter  50 value 0.601171
## iter  60 value 0.390960
## iter  70 value 0.384657
## iter  80 value 0.262284
## iter  90 value 0.249072
## iter 100 value 0.205724
## final  value 0.205724
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.718380
## iter  20 value 3.929813
## iter  30 value 3.353126
## iter  40 value 0.104534
## iter  50 value 0.104117
## iter  60 value 0.097609
## iter  70 value 0.097037
## iter  80 value 0.086383
## iter  90 value 0.084444
## iter 100 value 0.077851
## final  value 0.077851
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 99.120047
## iter  10 value 11.933355
## iter  20 value 3.493900
## iter  30 value 2.945996
## iter  40 value 0.732680
## iter  50 value 0.690248
## iter  60 value 0.670628
## iter  70 value 0.289165
## iter  80 value 0.278986
## iter  90 value 0.192272
## iter 100 value 0.173610
## final  value 0.173610
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
```

```
## iter  10 value 10.589060
## iter  20 value 4.150448
## iter  30 value 3.584804
## iter  40 value 1.020577
## iter  50 value 1.007108
## iter  60 value 0.919038
## iter  70 value 0.826713
## iter  80 value 0.784948
## iter  90 value 0.577085
## iter 100 value 0.497702
## final  value 0.497702
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.958606
## iter  20 value 3.655620
## iter  30 value 3.096499
## iter  40 value 1.329243
## iter  50 value 1.303415
## iter  60 value 1.203819
## iter  70 value 1.150995
## iter  80 value 1.095638
## iter  90 value 0.995947
## iter 100 value 0.829743
## final  value 0.829743
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.859541
## iter  20 value 3.855211
## iter  30 value 3.345851
## iter  40 value 0.569341
## iter  50 value 0.564928
## iter  60 value 0.430958
## iter  70 value 0.423363
## iter  80 value 0.396087
## iter  90 value 0.371687
## iter 100 value 0.297051
## final  value 0.297051
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.676979
```

```
## iter  20 value 4.232484
## iter  30 value 3.733717
## iter  40 value 0.639774
## iter  50 value 0.630778
## iter  60 value 0.412406
## iter  70 value 0.405037
## iter  80 value 0.272027
## iter  90 value 0.258316
## iter 100 value 0.213293
## final  value 0.213293
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.531135
## iter  20 value 3.742361
## iter  30 value 3.132005
## iter  40 value 0.891004
## iter  50 value 0.868782
## iter  60 value 0.609795
## iter  70 value 0.593331
## iter  80 value 0.471800
## iter  90 value 0.447446
## iter 100 value 0.382633
## final  value 0.382633
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.597548
## iter  20 value 3.754976
## iter  30 value 3.281164
## iter  40 value 0.938544
## iter  50 value 0.925032
## iter  60 value 0.779413
## iter  70 value 0.766230
## iter  80 value 0.583586
## iter  90 value 0.538399
## iter 100 value 0.421028
## final  value 0.421028
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.372768
## iter  20 value 3.569091
```

```
## iter  30 value 2.985775
## iter  40 value 0.172616
## iter  50 value 0.172204
## iter  60 value 0.131190
## iter  70 value 0.130484
## iter  80 value 0.111385
## iter  90 value 0.105490
## iter 100 value 0.086475
## final  value 0.086475
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 10.609013
## iter  20 value 3.729506
## iter  30 value 3.106183
## iter  40 value 1.502462
## iter  50 value 1.464160
## iter  60 value 1.110145
## iter  70 value 1.039249
## iter  80 value 0.855979
## iter  90 value 0.796130
## iter 100 value 0.635049
## final  value 0.635049
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.398740
## iter  20 value 4.392583
## iter  30 value 3.874341
## iter  40 value 0.115496
## iter  50 value 0.115104
## iter  60 value 0.096640
## iter  70 value 0.095931
## iter  80 value 0.077875
## iter  90 value 0.075993
## iter 100 value 0.069123
## final  value 0.069123
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.708881
## iter  20 value 3.503694
## iter  30 value 2.862616
```

```
## iter  40 value 0.991501
## iter  50 value 0.977651
## iter  60 value 0.802901
## iter  70 value 0.781890
## iter  80 value 0.481573
## iter  90 value 0.382871
## iter 100 value 0.198968
## final  value 0.198968
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.753209
## iter  20 value 4.355888
## iter  30 value 3.831359
## iter  40 value 0.742737
## iter  50 value 0.716744
## iter  60 value 0.578642
## iter  70 value 0.564624
## iter  80 value 0.378817
## iter  90 value 0.346232
## iter 100 value 0.253998
## final  value 0.253998
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.265825
## iter  20 value 4.480682
## iter  30 value 3.949638
## iter  40 value 0.597113
## iter  50 value 0.591094
## iter  60 value 0.384596
## iter  70 value 0.378566
## iter  80 value 0.259674
## iter  90 value 0.246483
## iter 100 value 0.203290
## final  value 0.203290
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 12.019881
## iter  20 value 3.883363
## iter  30 value 3.347301
## iter  40 value 0.154245
```

```
## iter  50 value 0.153830
## iter  60 value 0.116684
## iter  70 value 0.116154
## iter  80 value 0.101492
## iter  90 value 0.097168
## iter 100 value 0.082900
## final  value 0.082900
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.177080
## iter  20 value 4.293719
## iter  30 value 3.783603
## iter  40 value 0.371006
## iter  50 value 0.370114
## iter  60 value 0.294718
## iter  70 value 0.290709
## iter  80 value 0.232780
## iter  90 value 0.217332
## iter 100 value 0.170723
## final  value 0.170723
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.873909
## iter  20 value 3.680049
## iter  30 value 3.131049
## iter  40 value 0.885129
## iter  50 value 0.862203
## iter  60 value 0.543780
## iter  70 value 0.528879
## iter  80 value 0.355191
## iter  90 value 0.324420
## iter 100 value 0.237029
## final  value 0.237029
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.302012
## iter  20 value 4.498443
## iter  30 value 3.958618
## iter  40 value 0.864116
## iter  50 value 0.840382
```

```
## iter  60 value 0.587585
## iter  70 value 0.571772
## iter  80 value 0.458773
## iter  90 value 0.433624
## iter 100 value 0.365614
## final  value 0.365614
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.877162
## iter  20 value 4.157834
## iter  30 value 3.646495
## iter  40 value 0.577031
## iter  50 value 0.573128
## iter  60 value 0.397306
## iter  70 value 0.390390
## iter  80 value 0.275176
## iter  90 value 0.257888
## iter 100 value 0.203927
## final  value 0.203927
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.352648
## iter  20 value 3.987916
## iter  30 value 3.460460
## iter  40 value 0.486590
## iter  50 value 0.485015
## iter  60 value 0.382613
## iter  70 value 0.376577
## iter  80 value 0.336020
## iter  90 value 0.316548
## iter 100 value 0.258248
## final  value 0.258248
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.775968
## iter  20 value 4.031729
## iter  30 value 3.449395
## iter  40 value 1.082469
## iter  50 value 0.965405
## iter  60 value 0.918023
```

```
## iter  70 value 0.592520
## iter  80 value 0.564268
## iter  90 value 0.408826
## iter 100 value 0.320808
## final  value 0.320808
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 99.120047
## iter  10 value 11.708800
## iter  20 value 3.599842
## iter  30 value 3.061494
## iter  40 value 0.153581
## iter  50 value 0.153187
## iter  60 value 0.123753
## iter  70 value 0.122889
## iter  80 value 0.101290
## iter  90 value 0.097507
## iter 100 value 0.084495
## final  value 0.084495
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.520356
## iter  20 value 3.875246
## iter  30 value 3.346202
## iter  40 value 0.497329
## iter  50 value 0.495661
## iter  60 value 0.390240
## iter  70 value 0.383953
## iter  80 value 0.342726
## iter  90 value 0.322178
## iter 100 value 0.260901
## final  value 0.260901
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.268516
## iter  20 value 3.538697
## iter  30 value 2.862321
## iter  40 value 1.007794
## iter  50 value 0.993059
## iter  60 value 0.829522
## iter  70 value 0.808690
```

```
## iter  80 value 0.531901
## iter  90 value 0.415810
## iter 100 value 0.216152
## final  value 0.216152
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 99.120047
## iter  10 value 11.501065
## iter  20 value 4.015310
## iter  30 value 3.487871
## iter  40 value 1.178559
## iter  50 value 0.772811
## iter  60 value 0.016038
## iter  70 value 0.016007
## iter  80 value 0.015087
## iter  90 value 0.014858
## iter 100 value 0.014024
## final  value 0.014024
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.759575
## iter  20 value 3.763601
## iter  30 value 3.269784
## iter  40 value 0.302781
## iter  50 value 0.301441
## iter  60 value 0.209839
## iter  70 value 0.208404
## iter  80 value 0.171522
## iter  90 value 0.160989
## iter 100 value 0.128236
## final  value 0.128236
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.286160
## iter  20 value 3.770281
## iter  30 value 3.217600
## iter  40 value 0.441715
## iter  50 value 0.440439
## iter  60 value 0.358342
## iter  70 value 0.353313
## iter  80 value 0.335628
```

```
## iter  90 value 0.320430
## iter 100 value 0.271697
## final  value 0.271697
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 10.902576
## iter  20 value 4.580698
## iter  30 value 3.987040
## iter  40 value 1.088724
## iter  50 value 1.058123
## iter  60 value 0.982638
## iter  70 value 0.825161
## iter  80 value 0.799528
## iter  90 value 0.629392
## iter 100 value 0.517897
## final  value 0.517897
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.120250
## iter  20 value 3.856528
## iter  30 value 3.336115
## iter  40 value 1.250537
## iter  50 value 1.231304
## iter  60 value 1.181459
## iter  70 value 1.135573
## iter  80 value 1.085317
## iter  90 value 0.998286
## iter 100 value 0.881629
## final  value 0.881629
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.916082
## iter  20 value 4.463075
## iter  30 value 3.847528
## iter  40 value 1.742673
## iter  50 value 1.692330
## iter  60 value 0.949048
## iter  70 value 0.935187
## iter  80 value 0.822345
## iter  90 value 0.759779
```

```
## iter 100 value 0.596720
## final  value 0.596720
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.404173
## iter  20 value 4.055373
## iter  30 value 3.500501
## iter  40 value 1.265428
## iter  50 value 1.170320
## iter  60 value 1.066304
## iter  70 value 0.978058
## iter  80 value 0.570949
## iter  90 value 0.465988
## iter 100 value 0.258234
## final  value 0.258234
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.233908
## iter  20 value 3.434448
## iter  30 value 2.840382
## iter  40 value 0.597400
## iter  50 value 0.565103
## iter  60 value 0.543153
## iter  70 value 0.106526
## iter  80 value 0.105957
## iter  90 value 0.099472
## iter 100 value 0.087707
## final  value 0.087707
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.701591
## iter  20 value 3.586110
## iter  30 value 2.952579
## iter  40 value 1.087194
## iter  50 value 1.051734
## iter  60 value 0.912249
## iter  70 value 0.876787
## iter  80 value 0.623552
## iter  90 value 0.578016
## iter 100 value 0.464498
```

```
## final  value 0.464498
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.733746
## iter  20 value 3.909193
## iter  30 value 3.410375
## iter  40 value 0.591303
## iter  50 value 0.587025
## iter  60 value 0.404674
## iter  70 value 0.397273
## iter  80 value 0.274866
## iter  90 value 0.258991
## iter 100 value 0.208428
## final  value 0.208428
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.861559
## iter  20 value 3.798138
## iter  30 value 3.212471
## iter  40 value 0.421898
## iter  50 value 0.420803
## iter  60 value 0.333159
## iter  70 value 0.327754
## iter  80 value 0.256652
## iter  90 value 0.236062
## iter 100 value 0.176682
## final  value 0.176682
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.107945
## iter  20 value 3.954605
## iter  30 value 3.360577
## iter  40 value 0.539044
## iter  50 value 0.535912
## iter  60 value 0.364900
## iter  70 value 0.358299
## iter  80 value 0.232689
## iter  90 value 0.222995
## iter 100 value 0.189997
## final  value 0.189997
```

```
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.182234
## iter  20 value 3.927418
## iter  30 value 3.391186
## iter  40 value 0.476439
## iter  50 value 0.474895
## iter  60 value 0.362134
## iter  70 value 0.353974
## iter  80 value 0.223595
## iter  90 value 0.213885
## iter 100 value 0.181070
## final  value 0.181070
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.388113
## iter  20 value 3.506436
## iter  30 value 2.913357
## iter  40 value 0.186106
## iter  50 value 0.185531
## iter  60 value 0.135472
## iter  70 value 0.134806
## iter  80 value 0.116591
## iter  90 value 0.110901
## iter 100 value 0.092504
## final  value 0.092504
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.471221
## iter  20 value 3.975213
## iter  30 value 3.441268
## iter  40 value 0.521578
## iter  50 value 0.519276
## iter  60 value 0.391992
## iter  70 value 0.385842
## iter  80 value 0.335393
## iter  90 value 0.321150
## iter 100 value 0.278998
## final  value 0.278998
## stopped after 100 iterations
```

```
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.978479
## iter  20 value 3.665251
## iter  30 value 3.093425
## iter  40 value 0.208954
## iter  50 value 0.208405
## iter  60 value 0.156066
## iter  70 value 0.155089
## iter  80 value 0.129324
## iter  90 value 0.121966
## iter 100 value 0.098487
## final  value 0.098487
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.426539
## iter  20 value 4.209082
## iter  30 value 3.697092
## iter  40 value 0.510982
## iter  50 value 0.509073
## iter  60 value 0.390386
## iter  70 value 0.383794
## iter  80 value 0.315004
## iter  90 value 0.302033
## iter 100 value 0.264464
## final  value 0.264464
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 10.973187
## iter  20 value 4.215777
## iter  30 value 3.686016
## iter  40 value 1.169579
## iter  50 value 1.151842
## iter  60 value 0.877580
## iter  70 value 0.864075
## iter  80 value 0.725827
## iter  90 value 0.680584
## iter 100 value 0.546784
## final  value 0.546784
## stopped after 100 iterations
## # weights:  5 (4 variable)
```

```
## initial  value 97.733752
## iter  10 value 11.870462
## iter  20 value 4.086066
## iter  30 value 3.584371
## iter  40 value 0.534569
## iter  50 value 0.532304
## iter  60 value 0.401801
## iter  70 value 0.394771
## iter  80 value 0.318132
## iter  90 value 0.303519
## iter 100 value 0.260550
## final  value 0.260550
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.692238
## iter  20 value 4.111818
## iter  30 value 3.604119
## iter  40 value 0.356209
## iter  50 value 0.355237
## iter  60 value 0.274666
## iter  70 value 0.270762
## iter  80 value 0.197428
## iter  90 value 0.188717
## iter 100 value 0.159396
## final  value 0.159396
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.649372
## iter  20 value 3.753235
## iter  30 value 3.153333
## iter  40 value 1.564307
## iter  50 value 1.523201
## iter  60 value 1.072609
## iter  70 value 1.024825
## iter  80 value 0.850314
## iter  90 value 0.789962
## iter 100 value 0.627514
## final  value 0.627514
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
```

```
## iter  10 value 11.011678
## iter  20 value 4.152796
## iter  30 value 3.603727
## iter  40 value 0.212198
## iter  50 value 0.211882
## iter  60 value 0.178707
## iter  70 value 0.177376
## iter  80 value 0.162811
## iter  90 value 0.158753
## iter 100 value 0.145617
## final  value 0.145617
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.637496
## iter  20 value 4.262887
## iter  30 value 3.676996
## iter  40 value 0.805011
## iter  50 value 0.776797
## iter  60 value 0.475070
## iter  70 value 0.463919
## iter  80 value 0.282673
## iter  90 value 0.269873
## iter 100 value 0.226902
## final  value 0.226902
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.473648
## iter  20 value 4.110160
## iter  30 value 3.612559
## iter  40 value 0.372405
## iter  50 value 0.371447
## iter  60 value 0.292131
## iter  70 value 0.288065
## iter  80 value 0.222773
## iter  90 value 0.207403
## iter 100 value 0.160618
## final  value 0.160618
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.365742
```

```
## iter  20 value 4.105004
## iter  30 value 3.586331
## iter  40 value 0.335223
## iter  50 value 0.334261
## iter  60 value 0.254090
## iter  70 value 0.249831
## iter  80 value 0.162820
## iter  90 value 0.156392
## iter 100 value 0.134371
## final  value 0.134371
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.793024
## iter  20 value 4.078526
## iter  30 value 3.549301
## iter  40 value 1.251274
## iter  50 value 1.227207
## iter  60 value 1.179484
## iter  70 value 1.123814
## iter  80 value 1.080230
## iter  90 value 0.986435
## iter 100 value 0.679513
## final  value 0.679513
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.204177
## iter  20 value 3.971603
## iter  30 value 3.393965
## iter  40 value 1.254951
## iter  50 value 1.234299
## iter  60 value 1.182070
## iter  70 value 1.084457
## iter  80 value 0.974430
## iter  90 value 0.878766
## iter 100 value 0.803256
## final  value 0.803256
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.960008
## iter  20 value 3.256952
```

```
## iter  30 value 2.740926
## iter  40 value 0.778466
## iter  50 value 0.741008
## iter  60 value 0.710729
## iter  70 value 0.379417
## iter  80 value 0.371546
## iter  90 value 0.295996
## iter 100 value 0.229164
## final  value 0.229164
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.793615
## iter  20 value 3.850673
## iter  30 value 3.329435
## iter  40 value 0.074958
## iter  50 value 0.063194
## iter  60 value 0.063080
## iter  70 value 0.055205
## iter  80 value 0.054493
## iter  90 value 0.047083
## iter 100 value 0.044472
## final  value 0.044472
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.922473
## iter  20 value 3.738010
## iter  30 value 3.201808
## iter  40 value 0.451142
## iter  50 value 0.449981
## iter  60 value 0.365467
## iter  70 value 0.359939
## iter  80 value 0.322060
## iter  90 value 0.304010
## iter 100 value 0.249671
## final  value 0.249671
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.974482
## iter  20 value 4.252198
## iter  30 value 3.703452
```

```
## iter  40 value 1.609695
## iter  50 value 1.567274
## iter  60 value 0.996391
## iter  70 value 0.980176
## iter  80 value 0.879852
## iter  90 value 0.811310
## iter 100 value 0.679063
## final  value 0.679063
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.725295
## iter  20 value 3.857583
## iter  30 value 3.280969
## iter  40 value 0.098559
## iter  50 value 0.098218
## iter  60 value 0.090648
## iter  70 value 0.090175
## iter  80 value 0.083900
## iter  90 value 0.082324
## iter 100 value 0.077070
## final  value 0.077070
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.240064
## iter  20 value 4.160361
## iter  30 value 3.668864
## iter  40 value 0.593873
## iter  50 value 0.588698
## iter  60 value 0.390897
## iter  70 value 0.384280
## iter  80 value 0.259861
## iter  90 value 0.247327
## iter 100 value 0.205824
## final  value 0.205824
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.085741
## iter  20 value 3.545687
## iter  30 value 2.984475
## iter  40 value 0.715780
```

```
## iter  50 value 0.681399
## iter  60 value 0.659112
## iter  70 value 0.478624
## iter  80 value 0.460158
## iter  90 value 0.345475
## iter 100 value 0.271628
## final  value 0.271628
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.123956
## iter  20 value 3.926035
## iter  30 value 3.396897
## iter  40 value 0.754054
## iter  50 value 0.723962
## iter  60 value 0.642599
## iter  70 value 0.631331
## iter  80 value 0.455665
## iter  90 value 0.436434
## iter 100 value 0.387573
## final  value 0.387573
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.346712
## iter  20 value 4.359654
## iter  30 value 3.804485
## iter  40 value 0.991136
## iter  50 value 0.979474
## iter  60 value 0.887610
## iter  70 value 0.842851
## iter  80 value 0.705823
## iter  90 value 0.655995
## iter 100 value 0.511321
## final  value 0.511321
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.102219
## iter  20 value 3.469714
## iter  30 value 2.887398
## iter  40 value 0.870077
## iter  50 value 0.856886
```

```
## iter  60 value 0.747217
## iter  70 value 0.536859
## iter  80 value 0.133799
## iter  90 value 0.017590
## iter 100 value 0.016839
## final  value 0.016839
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.867549
## iter  20 value 3.943340
## iter  30 value 3.446392
## iter  40 value 0.572869
## iter  50 value 0.569473
## iter  60 value 0.406399
## iter  70 value 0.399085
## iter  80 value 0.292976
## iter  90 value 0.269998
## iter 100 value 0.202767
## final  value 0.202767
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.393838
## iter  20 value 4.077909
## iter  30 value 3.505866
## iter  40 value 0.155068
## iter  50 value 0.154807
## iter  60 value 0.124480
## iter  70 value 0.123574
## iter  80 value 0.100112
## iter  90 value 0.096770
## iter 100 value 0.085041
## final  value 0.085041
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.871997
## iter  20 value 3.979587
## iter  30 value 3.412253
## iter  40 value 1.637948
## iter  50 value 1.596085
## iter  60 value 0.902524
```

```
## iter  70 value 0.886596
## iter  80 value 0.742792
## iter  90 value 0.697495
## iter 100 value 0.566920
## final  value 0.566920
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.729480
## iter  20 value 3.660313
## iter  30 value 3.074839
## iter  40 value 0.113533
## iter  50 value 0.113429
## iter  60 value 0.100221
## iter  70 value 0.099658
## iter  80 value 0.091043
## iter  90 value 0.089559
## iter 100 value 0.084857
## final  value 0.084857
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 10.261780
## iter  20 value 3.866165
## iter  30 value 3.277098
## iter  40 value 1.510759
## iter  50 value 1.472700
## iter  60 value 1.115297
## iter  70 value 1.044721
## iter  80 value 0.864526
## iter  90 value 0.803441
## iter 100 value 0.640286
## final  value 0.640286
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 10.112241
## iter  20 value 3.728152
## iter  30 value 3.072208
## iter  40 value 1.784013
## iter  50 value 1.718485
## iter  60 value 0.872857
## iter  70 value 0.850845
```

```
## iter  80 value 0.661813
## iter  90 value 0.615858
## iter 100 value 0.487770
## final  value 0.487770
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.674772
## iter  20 value 3.917352
## iter  30 value 3.428412
## iter  40 value 0.122975
## iter  50 value 0.122852
## iter  60 value 0.107380
## iter  70 value 0.106678
## iter  80 value 0.093677
## iter  90 value 0.090678
## iter 100 value 0.080527
## final  value 0.080527
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.795264
## iter  20 value 4.199966
## iter  30 value 3.678710
## iter  40 value 0.337157
## iter  50 value 0.336241
## iter  60 value 0.258646
## iter  70 value 0.255051
## iter  80 value 0.182698
## iter  90 value 0.175799
## iter 100 value 0.151940
## final  value 0.151940
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 98.426900
## iter  10 value 11.377811
## iter  20 value 4.525338
## iter  30 value 3.935225
## iter  40 value 1.427070
## iter  50 value 1.391209
## iter  60 value 1.107276
## iter  70 value 1.061320
## iter  80 value 0.965177
```

```
## iter  90 value 0.870734
## iter 100 value 0.796186
## final  value 0.796186
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 12.114517
## iter  20 value 3.978200
## iter  30 value 3.470061
## iter  40 value 0.083360
## iter  50 value 0.082515
## iter  60 value 0.082105
## iter  70 value 0.079848
## iter  80 value 0.078658
## iter  90 value 0.066477
## iter 100 value 0.062072
## final  value 0.062072
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.225075
## iter  20 value 3.811388
## iter  30 value 3.268944
## iter  40 value 0.532574
## iter  50 value 0.528259
## iter  60 value 0.430257
## iter  70 value 0.424162
## iter  80 value 0.389898
## iter  90 value 0.369820
## iter 100 value 0.305743
## final  value 0.305743
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.194022
## iter  20 value 3.769653
## iter  30 value 3.208199
## iter  40 value 0.178664
## iter  50 value 0.178285
## iter  60 value 0.143954
## iter  70 value 0.142827
## iter  80 value 0.116432
## iter  90 value 0.111765
```

```
## iter 100 value 0.095897
## final  value 0.095897
## stopped after 100 iterations
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.308230
## iter  20 value 4.091156
## iter  30 value 3.597855
## iter  40 value 0.832228
## iter  50 value 0.803547
## iter  60 value 0.588320
## iter  70 value 0.572919
## iter  80 value 0.442610
## iter  90 value 0.420733
## iter 100 value 0.362445
## final  value 0.362445
## stopped after 100 iterations
```

```
prop_correctos_vector <- prop_correctos_vector[-1]
promedio_correctos_multi <- mean(prop_correctos_vector) # proporcion promedio de correctos de validacion
error_kfold <- error_kfold[-1]

print(c("promedio_correctos_multi", promedio_correctos_multi))
```

```
## [1] "promedio_correctos_multi" "1"
```

2.2.2. Error Obtenido en Cada Iteración por el Método de Validación Cruzada de K-Pliegues

```
error_kfold
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

2.2.3. Graficando el Error Estimado por el Método de Validación Cruzada de K-Pliegues

```
#Modelo de Regresión Logística Multinomial Entrenamiento
mod_multi_entre <- nnet::multinom(MoodysRat~., data = datos.entre, maxit = 100)
```

```
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.308230
## iter  20 value 4.091156
## iter  30 value 3.597855
## iter  40 value 0.832228
```

```
## iter  50 value 0.803547
## iter  60 value 0.588320
## iter  70 value 0.572919
## iter  80 value 0.442610
## iter  90 value 0.420733
## iter 100 value 0.362445
## final  value 0.362445
## stopped after 100 iterations
```

```
prediccion_multi_entre=predict(mod_multi_entre,newdata = datos.entre[,-1])
(matriz_confusion_multi_entre=table(prediccion_multi_entre,
                      datos.entre$MoodysRat,dnn=c("Predicho","Real")))
```

```
##         Real
## Predicho no yes
##      no  95   0
##      yes  0  46
```

```
prop_correctos_multi_entre <-
(matriz_confusion_multi_entre[1,1]+matriz_confusion_multi_entre[2,2])/sum(mat
riz_confusion_multi_entre) # proporcion de correctos
```

```
#Modelo de Regresión Logística Multinomial
mod_multi_total <- nnet::multinom(MoodysRat~., data = datos.entre, maxit = 100)
```

```
## # weights:  5 (4 variable)
## initial  value 97.733752
## iter  10 value 11.308230
## iter  20 value 4.091156
## iter  30 value 3.597855
## iter  40 value 0.832228
## iter  50 value 0.803547
## iter  60 value 0.588320
## iter  70 value 0.572919
## iter  80 value 0.442610
## iter  90 value 0.420733
## iter 100 value 0.362445
## final  value 0.362445
## stopped after 100 iterations
```

```
prediccion_multi_total=predict(mod_multi_total,newdata = COUNTRIES[,-1])
(matriz_confusion_multi_total=table(prediccion_multi_total,
                      COUNTRIES$MoodysRat,dnn=c("Predicho","Real")))
```

```
##         Real
## Predicho  no yes
##      no  108   0
##      yes   0  49
```

```
prop_correctos_multi_total <-
(matriz_confusion_multi_total[1,1]+matriz_confusion_multi_total[2,2])/sum(matri
z_confusion_multi_total) # proporcion de correctos
```

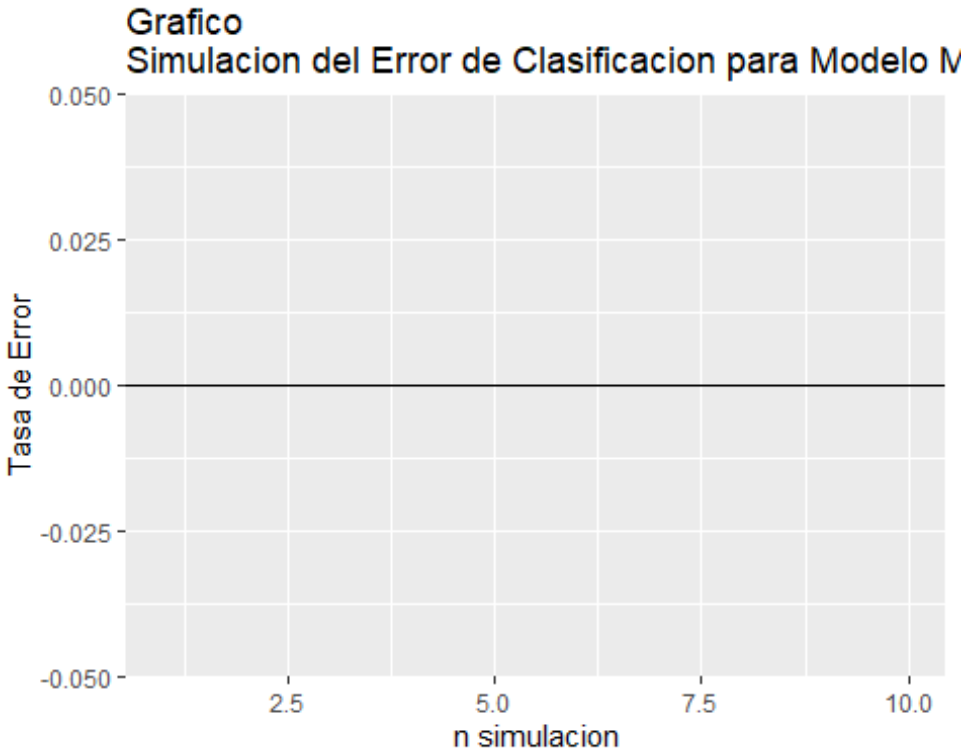
```
(correcto_multi_vali_total <-
(sum(diag(matriz_confusion_multi_total)))/sum(matriz_confusion_multi_total))
```

```
## [1] 1
```

```
(incorrecto_multi_vali_total<- 1 - correcto_multi_vali_total )
```

```
## [1] 0
```

```
dat_error=data.frame(error_kfold, n=1:10)
ggplot(data = dat_error,mapping = aes(x = n, y =error_kfold )) +
  geom_line(color = "red")+
 geom_hline(yintercept=incorrecto_multi_vali_total)+
  labs(title = "Grafico
Simulacion del Error de Clasificacion para Modelo Multinomial ",
       x = "n simulacion",
       y = "Tasa de Error")
```



2.3. Análisis Discriminante Lineal

2.3.1. Construcción del Algoritmo de Validación Requerido

```
n=dim(COUNTRIES)[1]

error_ind <- 0
error_kfold <- 0
prop_correctos_vector <- 0

for(i in 1:10){

 error_ind <- 0
 folds <- createFolds(1:n, k = 10)

 for(j in 1:10){
  datos.entre=COUNTRIES[-folds[[j]],]
  datos.vali=COUNTRIES[folds[[j]],]

  #Modelo de Análisis Discriminante Lineal
  modelo_lda <- lda(MoodysRat ~ ., data = datos.entre)
  predicciones <- predict(object = modelo_lda, newdata = datos.vali[,-1])
  (matriz_confusion_discrim_tot=table(predicciones$class,
datos.vali$MoodysRat,dnn=c("Predicho","Real")))
```

```
   prop_correctos <-
(matriz_confusion_discrim_tot[1,1]+matriz_confusion_discrim_tot[2,2])/sum(matr
iz_confusion_discrim_tot) # proporcion de correctos
   prop_incorrectos <- 1 - prop_correctos # proporcion de incorrectos

   (correcto_discrim_tot <-
(sum(diag(matriz_confusion_discrim_tot)))/sum(matriz_confusion_discrim_tot))
   (incorrecto_discrim_tot<- 1 - correcto_discrim_tot)

   temp = incorrecto_discrim_tot
   error_ind <- error_ind + temp # suma de los errores del modelo construido con los 10
folds
  }

  prop_correctos_vector <- c(prop_correctos_vector, prop_correctos)
  temp_vector <- error_ind/10
  error_kfold <- c(error_kfold, temp_vector)

}
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
prop_correctos_vector <- prop_correctos_vector[-1]
promedio_correctos_discrim <- mean(prop_correctos_vector) # proporcion promedio de correctos de validacion
error_kfold <- error_kfold[-1]

print(c("promedio_correctos_discrim", promedio_correctos_discrim))

## [1] "promedio_correctos_discrim" "0.955"
```

2.3.2. Error Obtenido en Cada Iteración por el Método de Validación Cruzada de K-Pliegues

```
error_kfold

##  [1] 0.05041667 0.05125000 0.05125000 0.05125000 0.05041667 0.05178571
##  [7] 0.05178571 0.05125000 0.05089286 0.05083333
```

2.3.3. Graficando el Error Estimado por el Método de Validación Cruzada de K-Pliegues

```
#Modelo de Análisis Discriminante Lineal Entrenamiento
modelo_lda_entre <- lda(MoodysRat ~ ., data = datos.entre)

## Warning in lda.default(x, grouping, ...): variables are collinear

predicciones_entre <- predict(object = modelo_lda_entre, newdata = datos.entre[,-1])
(matriz_confusion_discrim_tot_entre=table(predicciones_entre$class, datos.entre$MoodysRat,dnn=c("Predicho","Real")))

##         Real
## Predicho no yes
##      no  94   2
##      yes  4  41

prop_correctos_discrim_tot_entre <- (matriz_confusion_discrim_tot_entre[1,1]+matriz_confusion_discrim_tot_entre[2,2])/sum(matriz_confusion_discrim_tot_entre) # proporcion de correctos

#Modelo de Análisis Discriminante Lineal
modelo_lda_total <- lda(MoodysRat ~ ., data = datos.entre)

## Warning in lda.default(x, grouping, ...): variables are collinear

predicciones_total <- predict(object = modelo_lda_total, newdata = COUNTRIES[,-1])
```

```
(matriz_confusion_discrim_tot_total=table(predicciones_total$class,
COUNTRIES$MoodysRat,dnn=c("Predicho","Real")))
```

```
##        Real
## Predicho  no yes
##      no  102   2
##      yes   6  47
```

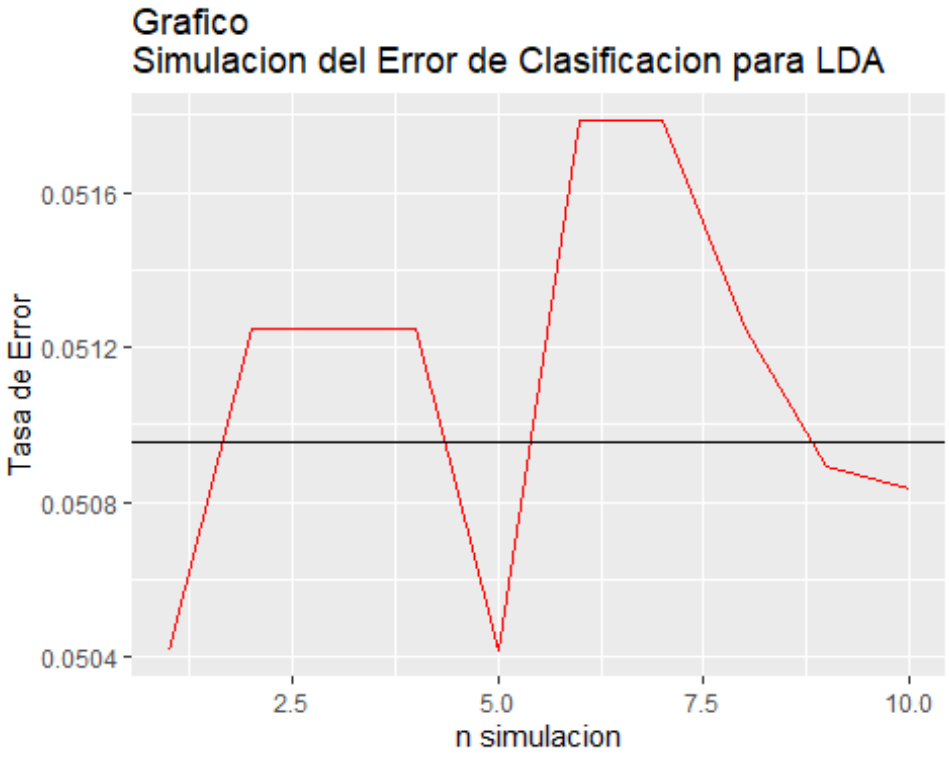
```
prop_correctos_discrim_tot_total <-
(matriz_confusion_discrim_tot_total[1,1]+matriz_confusion_discrim_tot_total[2,2])
/sum(matriz_confusion_discrim_tot_total) # proporcion de correctos

(correcto_discrim_tot_total <-
(sum(diag(matriz_confusion_discrim_tot_total)))/sum(matriz_confusion_discrim_
tot_total))
```

```
## [1] 0.9490446
```

```
(incorrecto_discrim_tot_total<- 1 - correcto_discrim_tot_total)
```

```
## [1] 0.05095541
```

```
dat_error=data.frame(error_kfold, n=1:10)
ggplot(data = dat_error,mapping = aes(x = n, y =error_kfold )) +
  geom_line(color = "red")+
 geom_hline(yintercept=incorrecto_discrim_tot_total)+
  labs(title = "Grafico
Simulacion del Error de Clasificacion para LDA ",
       x = "n simulacion",
       y = "Tasa de Error")
```



2.4. Árboles de Clasificación y Decisión

2.4.1. Construcción del Algoritmo de Validación Requerido

```
n=dim(COUNTRIES)[1]

error_ind <- 0
error_kfold <- 0
prop_correctos_vector <- 0

for(i in 1:10){

 error_ind <- 0
 folds <- createFolds(1:n, k = 10)

 for(j in 1:10){
  datos.entre=COUNTRIES[-folds[[j]],]
  datos.vali=COUNTRIES[folds[[j]],]

  #Modelo de Árboles de Clasificación
  modelo_arboles <- rpart(MoodysRat ~ ., data = datos.entre)
  (predi_arboles=predict(modelo_arboles, type = "class", newdata = datos.vali[,-
1]))
  (matriz_confusion_arboles=table(predi_arboles,
```

```
datos.vali$MoodysRat,dnn=c("Predicho","Real")))
   prop_correctos <-
(matriz_confusion_arboles[1,1]+matriz_confusion_arboles[2,2])/sum(matriz_confu
sion_arboles) # proporcion de correctos
   prop_incorrectos <- 1 - prop_correctos # proporcion de incorrectos

   (correcto_arboles_entre <-
(sum(diag(matriz_confusion_arboles)))/sum(matriz_confusion_arboles))
   (incorrecto_arboles_entre<- 1 - correcto_arboles_entre)

   temp = incorrecto_arboles_entre
   error_ind <- error_ind + temp # suma de los errores del modelo construido con los 10
folds
  }

  prop_correctos_vector <- c(prop_correctos_vector, prop_correctos)
  temp_vector <- error_ind/10
  error_kfold <- c(error_kfold, temp_vector)

}

prop_correctos_vector <- prop_correctos_vector[-1]
promedio_correctos_arboles <- mean(prop_correctos_vector) # proporcion promedio
de correctos de validacion
error_kfold <- error_kfold[-1]

print(c("promedio_correctos_arboles", promedio_correctos_arboles))

## [1] "promedio_correctos_arboles" "1"
```

2.4.2. Error Obtenido en Cada Iteración por el Método de Validación Cruzada de K-Pliegues

```
error_kfold

##  [1] 0 0 0 0 0 0 0 0 0 0
```

2.4.3. Graficando el Error Estimado por el Método de Validación Cruzada de K-Pliegues

```
#Modelo de Árboles de Clasificación Entrenamiento
modelo_arboles_entre <- rpart(MoodysRat ~ ., data = datos.entre)
(predi_arboles_entre =predict(modelo_arboles_entre , type = "class", newdata =
datos.entre[,-1]))
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## yes  no  no  no  no  no  no  no  no  no  no  no yes  no yes  no  no yes  no  no
## 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  no  no  no yes  no yes yes yes  no  no  no  no  no  no  no  no  no yes  no  no
## 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  no  no yes  no yes yes  no  no yes  no  no yes  no yes  no yes  no  no  no yes
## 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
## yes yes  no  no  no yes yes  no  no  no yes  no  no yes  no yes yes yes  no yes
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##  no  no yes  no  no  no  no  no  no  no  no yes yes  no  no  no yes  no  no  no
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  no  no yes  no yes yes  no  no  no yes  no  no  no yes yes yes  no yes  no  no
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##  no  no  no yes yes yes  no  no  no  no  no  no  no  no yes yes yes  no  no
## 141 142
##  no  no
## Levels: no yes
```

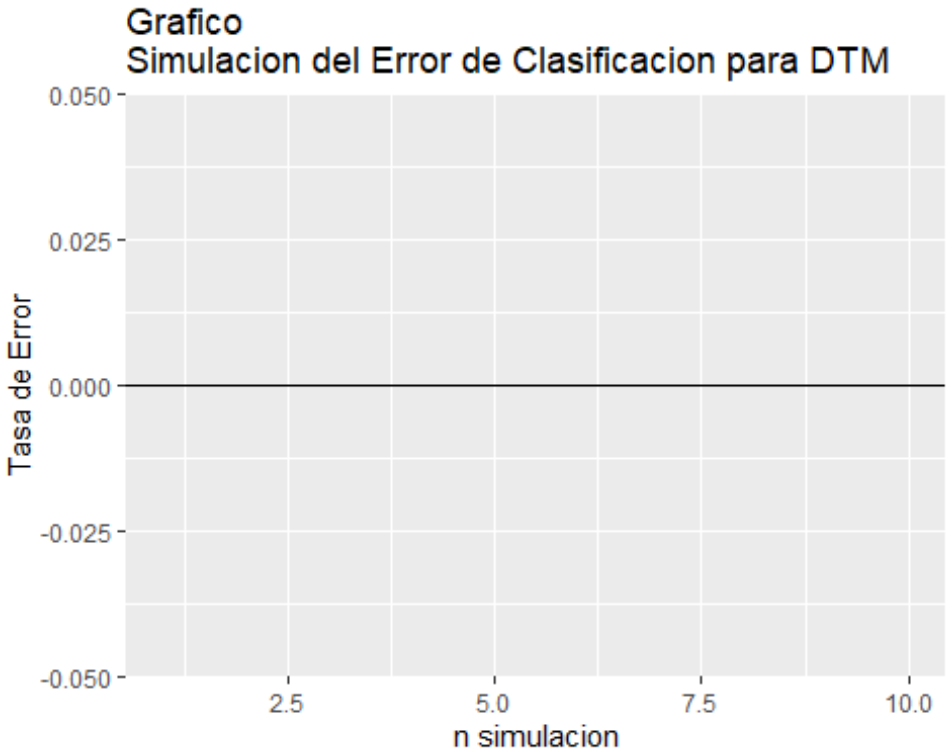
```
(matriz_confusion_arboles_entre =table(predi_arboles_entre ,
datos.entre$MoodysRat,dnn=c("Predicho","Real")))
```

```
##        Real
## Predicho no yes
##      no  97   0
##      yes  0  45
```

```
prop_correctos_arboles_entre <-
(matriz_confusion_arboles_entre[1,1]+matriz_confusion_arboles_entre[2,2])/sum(
matriz_confusion_arboles_entre) # proporcion de correctos

#Modelo de Árboles de Clasificación
modelo_arboles_total <- rpart(MoodysRat ~ ., data = datos.entre)
(predi_arboles_total =predict(modelo_arboles_total , type = "class", newdata =
COUNTRIES[,-1]))
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## yes  no  no  no  no  no  no yes yes  no  no  no  no  no  no yes  no  no yes  no
## 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  no yes  no  no  no  no  no yes  no yes yes yes  no  no  no  no  no  no  no  no
## 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
##  no yes yes  no  no  no  no yes  no  no yes yes  no  no yes  no  no  no yes  no
## 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
## yes  no yes  no  no  no yes yes yes  no  no  no yes yes  no  no  no yes  no  no
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

```
## yes  no yes yes yes yes  no yes  no  no yes  no  no  no  no  no  no  no  no  no
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
## yes yes  no  no  no yes  no  no  no  no  no  no  no yes  no yes yes  no  no  no
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
## yes  no  no  no yes yes yes  no  no yes  no  no  no  no  no  no yes yes yes  no
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
##  no  no  no  no  no  no  no  no  no yes yes yes  no  no  no  no  no
## Levels: no yes

(matriz_confusion_arboles_total =table(predi_arboles_total ,
COUNTRIES$MoodysRat,dnn=c("Predicho","Real")))

##         Real
## Predicho  no yes
##      no  108   0
##      yes   0  49

prop_correctos_arboles_total <-
(matriz_confusion_arboles_total[1,1]+matriz_confusion_arboles_total[2,2])/sum(m
atriz_confusion_arboles_total) # proporcion de correctos

(correcto_arboles_entre_total  <- (sum(diag(matriz_confusion_arboles_total
))))/sum(matriz_confusion_arboles_total ))

## [1] 1

(incorrecto_arboles_entre_total <- 1 - correcto_arboles_entre_total )

## [1] 0

dat_error=data.frame(error_kfold, n=1:10)
ggplot(data = dat_error,mapping = aes(x = n, y =error_kfold )) +
  geom_line(color = "red")+
 geom_hline(yintercept=incorrecto_arboles_entre_total )+
  labs(title = "Grafico
Simulacion del Error de Clasificacion para DTM ",
       x = "n simulacion",
       y = "Tasa de Error")
```



2.5. K-Vecinos Más Cercanos

2.5.1. Construcción del Algoritmo de Validación Requerido

```
n=dim(COUNTRIES)[1]

error_ind <- 0
error_kfold <- 0
prop_correctos_vector <- 0

for(i in 1:10){

 error_ind <- 0
 folds <- createFolds(1:n, k = 10)

 for(j in 1:10){
  datos.entre=COUNTRIES[-folds[[j]],]
  datos.vali=COUNTRIES[folds[[j]],]

  #Modelo de K-Vecinos Más Cercanos
  modelo_kvecinos <- train.kknn(MoodysRat ~ ., data = datos.entre, kmax = 10)
  (predi_kvecinos <- predict(modelo_kvecinos, newdata = datos.vali[,-1]))
  (matriz_confusion_kvecinos_tot=table(predi_kvecinos,
datos.vali$MoodysRat,dnn=c("Predicho","Real")))
```

```
   prop_correctos <-
(matriz_confusion_kvecinos_tot[1,1]+matriz_confusion_kvecinos_tot[2,2])/sum(m
atriz_confusion_kvecinos_tot) # proporcion de correctos
   prop_incorrectos <- 1 - prop_correctos # proporcion de incorrectos

   (correcto_kvecinos_tot <-
(sum(diag(matriz_confusion_kvecinos_tot)))/sum(matriz_confusion_kvecinos_tot)
)
   (incorrecto_kvecinos_tot<- 1 - correcto_kvecinos_tot)

   temp = incorrecto_kvecinos_tot
   error_ind <- error_ind + temp # suma de los errores del modelo construido con los 10
folds
  }

  prop_correctos_vector <- c(prop_correctos_vector, prop_correctos)
  temp_vector <- error_ind/10
  error_kfold <- c(error_kfold, temp_vector)

}

prop_correctos_vector <- prop_correctos_vector[-1]
promedio_correctos_kvecinos <- mean(prop_correctos_vector) # proporcion
promedio de correctos de validacion
error_kfold <- error_kfold[-1]

print(c("promedio_correctos_kvecinos", promedio_correctos_kvecinos))

## [1] "promedio_correctos_kvecinos" "1"
```

2.5.2. Error Obtenido en Cada Iteración por el Método de Validación Cruzada de K-Pliegues

```
error_kfold

##  [1] 0 0 0 0 0 0 0 0 0 0
```

2.5.3. Graficando el Error Estimado por el Método de Validación Cruzada de K-Pliegues

```
#Modelo de K-Vecinos Más Cercanos Entrenamiento
modelo_kvecinos_entre <- train.kknn(MoodysRat ~ ., data = datos.entre, kmax =
10)
(predi_kvecinos_entre <- predict(modelo_kvecinos_entre, newdata = datos.entre[,-
1]))
```

```
##   [1] yes no  no  no  no  no  yes yes no  no  no  no  no  no  yes no  no  yes
##  [19] no  no  yes no  no  no  no  no  no  yes yes no  no  no  no  no  no
##  [37] no  yes yes no  no  no  no  yes no  no  yes no  yes no  no  yes no  yes
##  [55] no  yes no  no  no  yes yes no  no  no  yes yes no  no  no  yes no  no
##  [73] yes yes yes yes no  yes no  no  yes no  no  no  no  no  no  no  no  yes
##  [91] yes no  no  no  yes no  no  no  no  no  no  no  yes no  yes yes no  no
## [109] no  yes no  no  yes no  no  yes no  no  no  no  no  yes yes yes no  no
## [127] no  no  no  no  no  no  no  no  yes yes yes no  no  no  no  no
## Levels: no yes
```

```
(matriz_confusion_kvecinos_entre=table(predi_kvecinos_entre,
datos.entre$MoodysRat,dnn=c("Predicho","Real")))
```

```
##         Real
## Predicho  no yes
##      no  100   0
##      yes   0  42
```

```
prop_correctos_kvecinos_entre <-
(matriz_confusion_kvecinos_entre[1,1]+matriz_confusion_kvecinos_entre[2,2])/su
m(matriz_confusion_kvecinos_entre) # proporcion de correctos

#Modelo de K-Vecinos Más Cercanos
modelo_kvecinos_total <- train.kknn(MoodysRat ~ ., data = datos.entre, kmax =
10)
(predi_kvecinos_total <- predict(modelo_kvecinos_total, newdata =
COUNTRIES[,-1]))
```

```
##   [1] yes no  no  no  no  no  no  yes yes no  no  no  no  no  no  yes no  no
##  [19] yes no  no  yes no  no  no  no  no  yes no  yes yes yes no  no  no  no
##  [37] no  no  no  no  no  yes yes no  no  no  no  yes no  no  yes yes no  no
##  [55] yes no  no  no  yes no  yes no  yes no  no  no  yes yes yes no  no  no
##  [73] yes yes no  no  no  yes no  no  yes no  yes yes yes yes no  yes no  no
##  [91] yes no  no  no  no  no  no  no  no  no  yes yes no  no  no  yes no  no
## [109] no  no  no  no  no  yes no  yes yes no  no  no  yes no  no  no  yes yes
## [127] yes no  no  yes no  no  no  no  no  no  yes yes yes no  no  no  no  no
## [145] no  no  no  no  no  yes yes yes no  no  no  no  no
## Levels: no yes
```

```
(matriz_confusion_kvecinos_total=table(predi_kvecinos_total,
COUNTRIES$MoodysRat,dnn=c("Predicho","Real")))
```

```
##         Real
## Predicho  no yes
```

```
##     no  108   0
##     yes   0  49
```

```
prop_correctos_kvecinos_total <-
(matriz_confusion_kvecinos_total[1,1]+matriz_confusion_kvecinos_total[2,2])/su
m(matriz_confusion_kvecinos_total) # proporcion de correctos
```

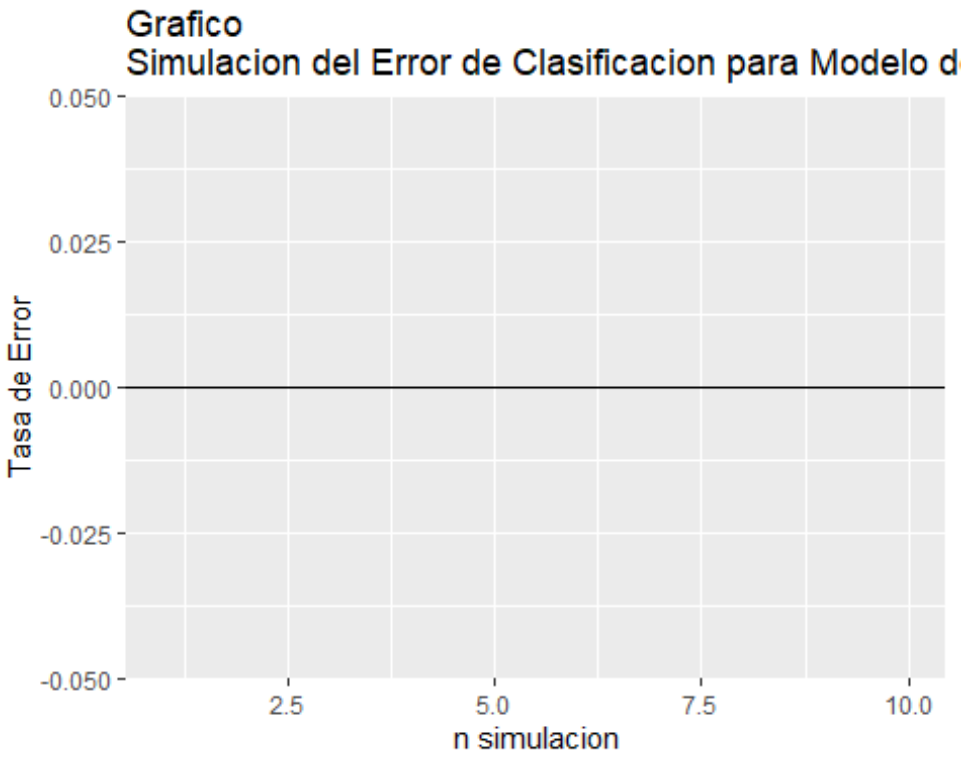
```
(correcto_kvecinos_total <-
(sum(diag(matriz_confusion_kvecinos_total)))/sum(matriz_confusion_kvecinos_t
otal))
```

```
## [1] 1
```

```
(incorrecto_kvecinos_total<- 1 - correcto_kvecinos_total)
```

```
## [1] 0
```

```
dat_error=data.frame(error_kfold, n=1:10)
ggplot(data = dat_error,mapping = aes(x = n, y =error_kfold )) +
  geom_line(color = "red")+
 geom_hline(yintercept=incorrecto_kvecinos_total)+
  labs(title = "Grafico
Simulacion del Error de Clasificacion para Modelo de KNN ",
       x = "n simulacion",
       y = "Tasa de Error")
```



2.6. Máquinas de Vectores de Soporte

2.6.1. Construcción del Algoritmo de Validación Requerido

```
n=dim(COUNTRIES)[1]

error_ind <- 0
error_kfold <- 0
prop_correctos_vector <- 0

for(i in 1:10){

 error_ind <- 0
 folds <- createFolds(1:n, k = 10)

 for(j in 1:10){
  datos.entre=COUNTRIES[-folds[[j]],]
  datos.vali=COUNTRIES[folds[[j]],]

  #Modelo de Máquinas de Vectores de Soporte
  modelo_svm <- svm(MoodysRat ~ ., data = datos.entre, kernel = "linear")
  (predi_svm <- predict(modelo_svm, newdata = datos.vali[,-1]))
  (matriz_confusion_svm_tot=table(predi_svm,
datos.vali$MoodysRat,dnn=c("Predicho","Real")))
  prop_correctos <-
(matriz_confusion_svm_tot[1,1]+matriz_confusion_svm_tot[2,2])/sum(matriz_con
fusion_svm_tot) # proporcion de correctos
  prop_incorrectos <- 1 - prop_correctos # proporcion de incorrectos

  (correcto_svm_tot <-
(sum(diag(matriz_confusion_svm_tot)))/sum(matriz_confusion_svm_tot))
  (incorrecto_svm_tot<- 1 - correcto_svm_tot)

  temp = incorrecto_svm_tot
  error_ind <- error_ind + temp # suma de los errores del modelo construido con los 10
folds
 }

 prop_correctos_vector <- c(prop_correctos_vector, prop_correctos)
 temp_vector <- error_ind/10
 error_kfold <- c(error_kfold, temp_vector)

}
```

```
prop_correctos_vector <- prop_correctos_vector[-1]
promedio_correctos_svm <- mean(prop_correctos_vector) # proporcion promedio de correctos de validacion
error_kfold <- error_kfold[-1]

print(c("promedio_correctos_svm", promedio_correctos_svm))
```

```
## [1] "promedio_correctos_svm" "1"
```

2.6.2. Error Obtenido en Cada Iteración por el Método de Validación Cruzada de K-Pliegues

```
error_kfold
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

2.6.3. Graficando el Error Estimado por el Método de Validación Cruzada de K-Pliegues

```
#Modelo de Máquinas de Vectores de Soporte Entrenamiento
modelo_svm_entre <- svm(MoodysRat ~ ., data = datos.entre, kernel = "linear")
(predi_svm_entre <- predict(modelo_svm_entre, newdata = datos.entre[,-1]))
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## yes  no  no  no  no  no  no yes yes  no  no  no  no yes  no  no yes  no  no yes
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  no  no  no  no yes  no yes yes  no  no  no  no  no  no  no  no  no yes yes  no
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##  no  no yes  no  no yes  no  no yes  no  no  no yes  no  no yes  no  no  no yes
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
## yes  no  no  no yes yes  no  no  no yes  no  no yes  no yes yes yes  no yes  no
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
## yes  no  no  no  no  no  no  no  no yes yes  no  no  no yes  no  no  no  no  no
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##  no  no yes  no yes yes  no  no  no yes  no  no  no yes yes yes  no  no yes  no
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##  no  no  no yes yes yes  no  no  no  no  no  no  no yes yes yes  no  no  no
## 141 142
##  no  no
## Levels: no yes
```

```
(matriz_confusion_svm_tot_entre=table(predi_svm_entre,
datos.entre$MoodysRat,dnn=c("Predicho","Real")))
```

```
##         Real
## Predicho no yes
```

```
##     no  98   0
##     yes  0  44
```

```
prop_correctos_svm_tot_entre <-
(matriz_confusion_svm_tot_entre[1,1]+matriz_confusion_svm_tot_entre[2,2])/sum
(matriz_confusion_svm_tot_entre) # proporcion de correctos
```

```
#Modelo de Máquinas de Vectores de Soporte
modelo_svm_total <- svm(MoodysRat ~ ., data = datos.entre, kernel = "linear")
(predi_svm_total <- predict(modelo_svm_total, newdata = COUNTRIES[,-1]))
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## yes  no  no  no  no  no  no yes yes  no  no  no  no  no  no yes  no  no yes  no
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##  no yes  no  no  no  no  no yes  no yes yes yes  no  no  no  no  no  no  no  no
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##  no yes yes  no  no  no  no yes  no  no yes yes  no  no yes  no  no  no yes  no
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
## yes  no yes  no  no  no yes yes yes  no  no  no yes yes  no  no  no yes  no  no
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
## yes  no yes yes yes yes  no yes  no  no yes  no  no  no  no  no  no  no  no  no
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
## yes yes  no  no  no yes  no  no  no  no  no  no  no yes  no yes yes  no  no  no
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
## yes  no  no  no yes yes yes  no  no yes  no  no  no  no  no  no yes yes yes  no
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
##  no  no  no  no  no  no  no  no  no yes yes yes  no  no  no  no  no
## Levels: no yes
```

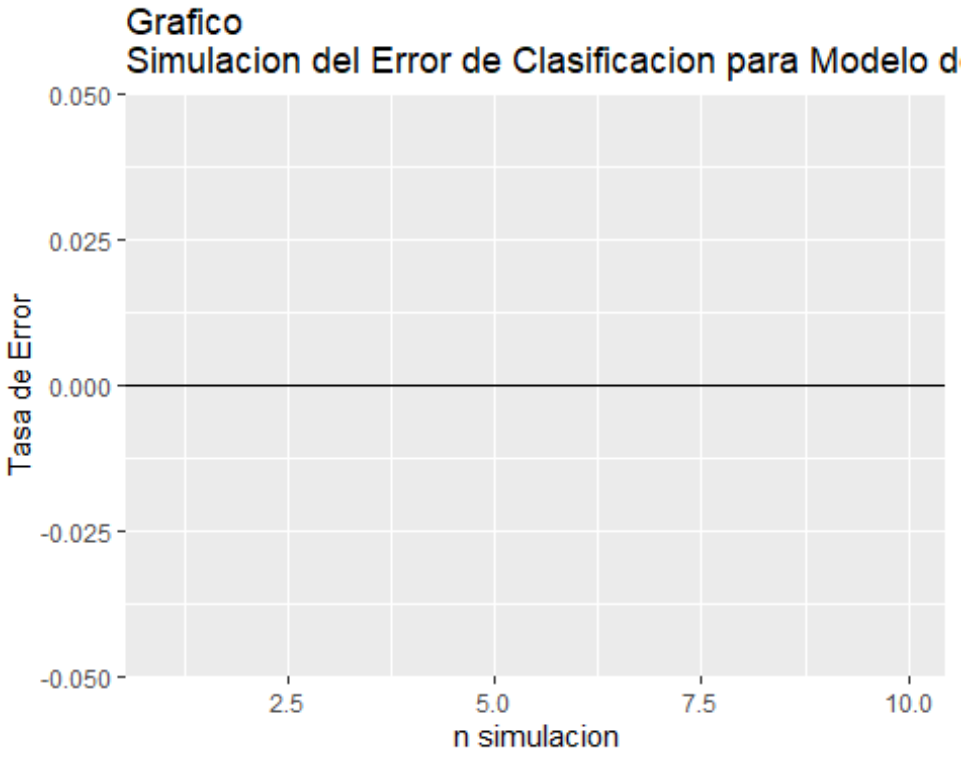
```
(matriz_confusion_svm_tot_total=table(predi_svm_total,
COUNTRIES$MoodysRat,dnn=c("Predicho","Real")))
```

```
##         Real
## Predicho  no yes
##     no  108   0
##     yes   0  49
```

```
prop_correctos_svm_tot_total <-
(matriz_confusion_svm_tot_total[1,1]+matriz_confusion_svm_tot_total[2,2])/sum(
matriz_confusion_svm_tot_total) # proporcion de correctos
```

```
(correcto_svm_tot_total <-
(sum(diag(matriz_confusion_svm_tot_total)))/sum(matriz_confusion_svm_tot_tot
al))
```

```
## [1] 1

(incorrecto_svm_tot_total<- 1 - correcto_svm_tot_total)

## [1] 0

dat_error=data.frame(error_kfold, n=1:10)
ggplot(data = dat_error,mapping = aes(x = n, y =error_kfold )) +
  geom_line(color = "red")+
 geom_hline(yintercept=incorrecto_svm_tot_total)+
  labs(title = "Grafico
Simulacion del Error de Clasificacion para Modelo de SVM ",
       x = "n simulacion",
       y = "Tasa de Error")
```




```
# 2.7. Cuadro de indicadores de clasificación correcta e incorrecta

# Correctos total

promedios_correctos_total <- c(prop_correctos_redes_total, prop_correctos_multi_total,
prop_correctos_discrim_tot_total, prop_correctos_arboles_total,
prop_correctos_kvecinos_total, prop_correctos_svm_tot_total)


# Correctos entrenamiento
```

```
promedios_correctos_entre <- c(prop_correctos_redes_entre, prop_correctos_multi_entre,
prop_correctos_discrim_tot_entre, prop_correctos_arboles_entre,
prop_correctos_kvecinos_entre, prop_correctos_svm_tot_entre)

# Correctos validacion

promedios_correctos_validacion <- c(promedio_correctos_redes, promedio_correctos_multi,
promedio_correctos_discrim, promedio_correctos_arboles, promedio_correctos_kvecinos,
promedio_correctos_svm)

nombres_modelos <- c("Redes", "Multinomial", "Discriminante", "Árboles", "K
vecinos", "SVM")

cuadro_correctos <- data.frame(nombres_modelos, promedios_correctos_total,
promedios_correctos_validacion, promedios_correctos_entre)

colnames(cuadro_correctos) <- c("Modelo", "Modelo completo", "Entrenamiento",
"Validación")

datatable(cuadro_correctos, caption = 'Cuadro Resumen de la Precisión de los Modelos de
Clasificación Implementados')

# Incorrectos total

promedios_incorrectos_total <- 1 - promedios_correctos_total

# Incorrectos entrenamiento

promedios_incorrectos_entre <- 1 - promedios_correctos_entre

# Incorrectos validacion

promedios_incorrectos_validacion <- 1 - promedios_correctos_validacion

cuadro_incorrectos <- data.frame(nombres_modelos, promedios_incorrectos_total,
promedios_incorrectos_validacion, promedios_incorrectos_entre)

colnames(cuadro_incorrectos) <- c("Modelo", "Modelo completo", "Entrenamiento",
"Validación")

datatable(cuadro_incorrectos, caption = 'Cuadro Resumen del Error de los Modelos de
Clasificación Implementados')
```

Cuadro Resumen del Error de los Modelos de Clasificación Implementados

	Modelo	Modelo completo	Entrenamiento	Validación
1	Redes	0.496815286624204	0.475416666666667	0.5
2	Multinomial	0	0	0
3	Discriminante	0.0509554140127388	0.045	0.0425531914893617
4	Árboles	0	0	0
5	K vecinos	0	0	0
6	SVM	0	0	0

Showing 1 to 6 of 6 entries Previous 1 Next

IV.II. Resultados del ensamble de modelos con 10 pliegues de validación cruzada y 5 repeticiones

#3. Ensamble de Modelos con 10 Pliegues de Validación Cruzada y 5 Repeticiones

#3.1. VALIDACIÓN CRUZADA CON REPETICIONES TIPO ÁRBOLES DE DECISIÓN AGREGADOS (BAGGED DECISION TREES)

```
#3.1.1. Modelo
library(caret)
control <- trainControl(method="repeatedcv", number=10, repeats=5)

set.seed(1)
modelo_bag_CART <- train(MoodysRat ~., data=COUNTRIES, method="treebag",
metric="Accuracy", trControl=control)
modelo_bag_CART
```

```
## Bagged CART
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 142, 141, 141, 142, 141, 141, ...
## Resampling results:
##
##   Accuracy  Kappa
##  1         1
```

```
#3.1.2. Predicción
prediccion_bag_CART= predict(modelo_bag_CART,data=COUNTRIES[,-1])

#3.1.3. Proporción de Clasificación Correcta
(correcto_bag_Cart<-modelo_bag_CART$results[2])

##   Accuracy
## 1        1

#3.1.4. Proporción de Clasificación Incorrecta
(incorrecto_bag_Cart<-1-correcto_bag_Cart)

##   Accuracy
## 1        0
```

#3.2. BOSQUES ALEATORIOS

```
#3.2.1. Modelo
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

control <- trainControl(method="repeatedcv", number=10, repeats=5)
set.seed(1)
modelo_bosques_2 <- train(MoodysRat ~., data=COUNTRIES, method="rf",
metric="Accuracy", trControl=control)

## note: only 2 unique complexity parameters in default grid. Truncating the grid
to 2 .

modelo_bosques_2

## Random Forest
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 142, 141, 141, 142, 141, 141, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##   2     1         1
##   3     1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
#3.2.2. Predicción
prediccion_bosques2 = predict(modelo_bosques_2,data=COUNTRIES[,-1], type =
"raw")
```

```
#3.2.3. Proporción de Clasificación Correcta
(correcto_bosques<-modelo_bosques_2$results[2,2])
```

```
## [1] 1
```

```
#3.2.4. Proporción de Clasificación Incorrecta
(incorrecto_bosques<-1-correcto_bosques)
```

```
## [1] 0
```

#3.3. BOOSTING

```
#3.3.1. Modelo (Estimado por el Método de Impulso de Gradiente Estocástico)
library(carData)
summary(COUNTRIES)
```

```
##  MoodysRat AdjDefaultSpread  CountryRiskPremium EquityRiskPremium
##  no :108   Min.   :0.00000   Min.   :0.00000    Min.   :0.04240
##  yes: 49   1st Qu.:0.00720   1st Qu.:0.00840    1st Qu.:0.05080
##            Median :0.02560   Median :0.02970    Median :0.07210
##            Mean   :0.03173   Mean   :0.03689    Mean   :0.07929
##            3rd Qu.:0.04680   3rd Qu.:0.05440    3rd Qu.:0.09680
##            Max.   :0.17500   Max.   :0.20340    Max.   :0.24580
```

```
library(ada)
control <- trainControl(method="repeatedcv", number=10, repeats=5)
set.seed(1)
modelo_boost_gbm <- train(MoodysRat~., data = COUNTRIES, method="gbm",
metric="Accuracy", trControl=control)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0996
##     2        0.9023             nan     0.1000    0.0697
##     3        0.7848             nan     0.1000    0.0503
##     4        0.6876             nan     0.1000    0.0477
##     5        0.6056             nan     0.1000    0.0388
##     6        0.5355             nan     0.1000    0.0344
##     7        0.4751             nan     0.1000    0.0297
##     8        0.4227             nan     0.1000    0.0287
##     9        0.3768             nan     0.1000    0.0229
##    10        0.3366             nan     0.1000    0.0204
##    20        0.1154             nan     0.1000    0.0062
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0854
##     2        0.9023             nan     0.1000    0.0746
##     3        0.7848             nan     0.1000    0.0575
##     4        0.6876             nan     0.1000    0.0486
##     5        0.6056             nan     0.1000    0.0380
##     6        0.5355             nan     0.1000    0.0374
##     7        0.4751             nan     0.1000    0.0287
##     8        0.4227             nan     0.1000    0.0275
##     9        0.3768             nan     0.1000    0.0222
##    10        0.3366             nan     0.1000    0.0213
##    20        0.1154             nan     0.1000    0.0063
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0996
##     2        0.9023             nan     0.1000    0.0746
```

```
##     3        0.7848             nan     0.1000    0.0599
##     4        0.6876             nan     0.1000    0.0468
##     5        0.6056             nan     0.1000    0.0425
##     6        0.5355             nan     0.1000    0.0332
##     7        0.4751             nan     0.1000    0.0307
##     8        0.4227             nan     0.1000    0.0246
##     9        0.3768             nan     0.1000    0.0194
##    10        0.3366             nan     0.1000    0.0204
##    20        0.1154             nan     0.1000    0.0063
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1039
##     2        0.9056             nan     0.1000    0.0809
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0523
##     5        0.6079             nan     0.1000    0.0359
##     6        0.5376             nan     0.1000    0.0345
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0275
##     9        0.3783             nan     0.1000    0.0226
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0611
##     4        0.6902             nan     0.1000    0.0459
##     5        0.6079             nan     0.1000    0.0440
```

```
##     6        0.5376             nan     0.1000    0.0368
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0247
##    10        0.3379             nan     0.1000    0.0193
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0760
##     3        0.7878             nan     0.1000    0.0647
##     4        0.6902             nan     0.1000    0.0468
##     5        0.6079             nan     0.1000    0.0381
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0263
##     9        0.3783             nan     0.1000    0.0237
##    10        0.3379             nan     0.1000    0.0199
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1015
##     2        0.9056             nan     0.1000    0.0696
##     3        0.7878             nan     0.1000    0.0599
##     4        0.6902             nan     0.1000    0.0496
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0267
```

```
##     9       0.3783             nan     0.1000    0.0237
##    10       0.3379             nan     0.1000    0.0196
##    20       0.1158             nan     0.1000    0.0064
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.0992
##     2       0.9056             nan     0.1000    0.0793
##     3       0.7878             nan     0.1000    0.0587
##     4       0.6902             nan     0.1000    0.0486
##     5       0.6079             nan     0.1000    0.0381
##     6       0.5376             nan     0.1000    0.0362
##     7       0.4770             nan     0.1000    0.0302
##     8       0.4244             nan     0.1000    0.0283
##     9       0.3783             nan     0.1000    0.0226
##    10       0.3379             nan     0.1000    0.0190
##    20       0.1158             nan     0.1000    0.0067
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.0945
##     2       0.9056             nan     0.1000    0.0793
##     3       0.7878             nan     0.1000    0.0575
##     4       0.6902             nan     0.1000    0.0505
##     5       0.6079             nan     0.1000    0.0403
##     6       0.5376             nan     0.1000    0.0333
##     7       0.4770             nan     0.1000    0.0322
##     8       0.4244             nan     0.1000    0.0275
##     9       0.3783             nan     0.1000    0.0202
##    10       0.3379             nan     0.1000    0.0202
##    20       0.1158             nan     0.1000    0.0063
```

```
##    40        0.0154             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0592             nan     0.1000    0.0893
##    2        0.9126             nan     0.1000    0.0725
##    3        0.7941             nan     0.1000    0.0586
##    4        0.6959             nan     0.1000    0.0478
##    5        0.6130             nan     0.1000    0.0411
##    6        0.5422             nan     0.1000    0.0374
##    7        0.4811             nan     0.1000    0.0308
##    8        0.4280             nan     0.1000    0.0235
##    9        0.3816             nan     0.1000    0.0230
##   10        0.3408             nan     0.1000    0.0202
##   20        0.1168             nan     0.1000    0.0064
##   40        0.0153             nan     0.1000    0.0008
##   60        0.0021             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0592             nan     0.1000    0.0870
##    2        0.9126             nan     0.1000    0.0757
##    3        0.7941             nan     0.1000    0.0586
##    4        0.6959             nan     0.1000    0.0496
##    5        0.6130             nan     0.1000    0.0397
##    6        0.5422             nan     0.1000    0.0340
##    7        0.4811             nan     0.1000    0.0303
##    8        0.4280             nan     0.1000    0.0255
##    9        0.3816             nan     0.1000    0.0220
##   10        0.3408             nan     0.1000    0.0202
##   20        0.1168             nan     0.1000    0.0062
##   40        0.0153             nan     0.1000    0.0008
##   60        0.0021             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
```

```
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0592             nan     0.1000    0.0915
##     2        0.9126             nan     0.1000    0.0663
##     3        0.7941             nan     0.1000    0.0575
##     4        0.6959             nan     0.1000    0.0558
##     5        0.6130             nan     0.1000    0.0418
##     6        0.5422             nan     0.1000    0.0345
##     7        0.4811             nan     0.1000    0.0289
##     8        0.4280             nan     0.1000    0.0267
##     9        0.3816             nan     0.1000    0.0230
##    10        0.3408             nan     0.1000    0.0199
##    20        0.1168             nan     0.1000    0.0061
##    40        0.0153             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1039
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0514
##     5        0.6079             nan     0.1000    0.0440
##     6        0.5376             nan     0.1000    0.0362
##     7        0.4770             nan     0.1000    0.0283
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0226
##    10        0.3379             nan     0.1000    0.0208
##    20        0.1158             nan     0.1000    0.0071
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
```

```
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517           nan     0.1000    0.0875
##     2       0.9056           nan     0.1000    0.0647
##     3       0.7878           nan     0.1000    0.0587
##     4       0.6902           nan     0.1000    0.0468
##     5       0.6079           nan     0.1000    0.0432
##     6       0.5376           nan     0.1000    0.0333
##     7       0.4770           nan     0.1000    0.0322
##     8       0.4244           nan     0.1000    0.0259
##     9       0.3783           nan     0.1000    0.0243
##    10       0.3379           nan     0.1000    0.0175
##    20       0.1158           nan     0.1000    0.0063
##    40       0.0152           nan     0.1000    0.0008
##    60       0.0020           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517           nan     0.1000    0.1039
##     2       0.9056           nan     0.1000    0.0712
##     3       0.7878           nan     0.1000    0.0587
##     4       0.6902           nan     0.1000    0.0496
##     5       0.6079           nan     0.1000    0.0425
##     6       0.5376           nan     0.1000    0.0368
##     7       0.4770           nan     0.1000    0.0292
##     8       0.4244           nan     0.1000    0.0259
##     9       0.3783           nan     0.1000    0.0212
##    10       0.3379           nan     0.1000    0.0205
##    20       0.1158           nan     0.1000    0.0066
##    40       0.0152           nan     0.1000    0.0008
##    60       0.0020           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.0517             nan     0.1000    0.0875
##      2        0.9056             nan     0.1000    0.0696
##      3        0.7878             nan     0.1000    0.0575
##      4        0.6902             nan     0.1000    0.0468
##      5        0.6079             nan     0.1000    0.0381
##      6        0.5376             nan     0.1000    0.0345
##      7        0.4770             nan     0.1000    0.0322
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0226
##     10        0.3379             nan     0.1000    0.0190
##     20        0.1158             nan     0.1000    0.0064
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0922
##      2        0.9056             nan     0.1000    0.0776
##      3        0.7878             nan     0.1000    0.0587
##      4        0.6902             nan     0.1000    0.0486
##      5        0.6079             nan     0.1000    0.0432
##      6        0.5376             nan     0.1000    0.0351
##      7        0.4770             nan     0.1000    0.0307
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0233
##     10        0.3379             nan     0.1000    0.0210
##     20        0.1158             nan     0.1000    0.0063
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0899
##      2        0.9056             nan     0.1000    0.0744
##      3        0.7878             nan     0.1000    0.0587
```

```
##      4        0.6902             nan     0.1000    0.0523
##      5        0.6079             nan     0.1000    0.0440
##      6        0.5376             nan     0.1000    0.0356
##      7        0.4770             nan     0.1000    0.0283
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0240
##     10        0.3379             nan     0.1000    0.0216
##     20        0.1158             nan     0.1000    0.0063
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0945
##      2        0.9056             nan     0.1000    0.0680
##      3        0.7878             nan     0.1000    0.0623
##      4        0.6902             nan     0.1000    0.0431
##      5        0.6079             nan     0.1000    0.0410
##      6        0.5376             nan     0.1000    0.0356
##      7        0.4770             nan     0.1000    0.0327
##      8        0.4244             nan     0.1000    0.0246
##      9        0.3783             nan     0.1000    0.0209
##     10        0.3379             nan     0.1000    0.0202
##     20        0.1158             nan     0.1000    0.0064
##     40        0.0153             nan     0.1000    0.0008
##     60        0.0021             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0852
##      2        0.9056             nan     0.1000    0.0712
##      3        0.7878             nan     0.1000    0.0671
##      4        0.6902             nan     0.1000    0.0468
##      5        0.6079             nan     0.1000    0.0432
##      6        0.5376             nan     0.1000    0.0380
```

```
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0279
##     9        0.3783             nan     0.1000    0.0223
##    10        0.3379             nan     0.1000    0.0196
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1086
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0599
##     4        0.6902             nan     0.1000    0.0514
##     5        0.6079             nan     0.1000    0.0388
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0317
##     8        0.4244             nan     0.1000    0.0279
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0193
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1109
##     2        0.9056             nan     0.1000    0.0712
##     3        0.7878             nan     0.1000    0.0599
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0418
##     6        0.5376             nan     0.1000    0.0374
##     7        0.4770             nan     0.1000    0.0288
##     8        0.4244             nan     0.1000    0.0267
##     9        0.3783             nan     0.1000    0.0240
```

```
##    10        0.3379             nan     0.1000    0.0210
##    20        0.1158             nan     0.1000    0.0059
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0992
##    2        0.9056             nan     0.1000    0.0631
##    3        0.7878             nan     0.1000    0.0527
##    4        0.6902             nan     0.1000    0.0459
##    5        0.6079             nan     0.1000    0.0366
##    6        0.5376             nan     0.1000    0.0380
##    7        0.4770             nan     0.1000    0.0307
##    8        0.4244             nan     0.1000    0.0254
##    9        0.3783             nan     0.1000    0.0233
##   10        0.3379             nan     0.1000    0.0216
##   20        0.1158             nan     0.1000    0.0065
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0992
##    2        0.9056             nan     0.1000    0.0712
##    3        0.7878             nan     0.1000    0.0599
##    4        0.6902             nan     0.1000    0.0477
##    5        0.6079             nan     0.1000    0.0374
##    6        0.5376             nan     0.1000    0.0374
##    7        0.4770             nan     0.1000    0.0307
##    8        0.4244             nan     0.1000    0.0250
##    9        0.3783             nan     0.1000    0.0233
##   10        0.3379             nan     0.1000    0.0213
##   20        0.1158             nan     0.1000    0.0070
##   40        0.0152             nan     0.1000    0.0008
```

```
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0922
##     2        0.9056             nan     0.1000    0.0760
##     3        0.7878             nan     0.1000    0.0539
##     4        0.6902             nan     0.1000    0.0505
##     5        0.6079             nan     0.1000    0.0440
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0230
##     9        0.3783             nan     0.1000    0.0226
##    10        0.3379             nan     0.1000    0.0190
##    20        0.1158             nan     0.1000    0.0056
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0969
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0611
##     4        0.6902             nan     0.1000    0.0459
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0362
##     7        0.4770             nan     0.1000    0.0312
##     8        0.4244             nan     0.1000    0.0275
##     9        0.3783             nan     0.1000    0.0240
##    10        0.3379             nan     0.1000    0.0210
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
```

```
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517           nan     0.1000    0.1015
##     2        0.9056           nan     0.1000    0.0776
##     3        0.7878           nan     0.1000    0.0611
##     4        0.6902           nan     0.1000    0.0477
##     5        0.6079           nan     0.1000    0.0418
##     6        0.5376           nan     0.1000    0.0368
##     7        0.4770           nan     0.1000    0.0297
##     8        0.4244           nan     0.1000    0.0259
##     9        0.3783           nan     0.1000    0.0216
##    10        0.3379           nan     0.1000    0.0199
##    20        0.1158           nan     0.1000    0.0067
##    40        0.0152           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482           nan     0.1000    0.0949
##     2        0.9023           nan     0.1000    0.0811
##     3        0.7848           nan     0.1000    0.0624
##     4        0.6876           nan     0.1000    0.0496
##     5        0.6056           nan     0.1000    0.0388
##     6        0.5355           nan     0.1000    0.0362
##     7        0.4751           nan     0.1000    0.0317
##     8        0.4227           nan     0.1000    0.0250
##     9        0.3768           nan     0.1000    0.0236
##    10        0.3366           nan     0.1000    0.0207
##    20        0.1154           nan     0.1000    0.0059
##    40        0.0151           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
```

```
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0878
##     2        0.9023             nan     0.1000    0.0680
##     3        0.7848             nan     0.1000    0.0587
##     4        0.6876             nan     0.1000    0.0496
##     5        0.6056             nan     0.1000    0.0425
##     6        0.5355             nan     0.1000    0.0338
##     7        0.4751             nan     0.1000    0.0297
##     8        0.4227             nan     0.1000    0.0271
##     9        0.3768             nan     0.1000    0.0233
##    10        0.3366             nan     0.1000    0.0201
##    20        0.1154             nan     0.1000    0.0063
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0878
##     2        0.9023             nan     0.1000    0.0680
##     3        0.7848             nan     0.1000    0.0575
##     4        0.6876             nan     0.1000    0.0505
##     5        0.6056             nan     0.1000    0.0447
##     6        0.5355             nan     0.1000    0.0320
##     7        0.4751             nan     0.1000    0.0322
##     8        0.4227             nan     0.1000    0.0283
##     9        0.3822             nan     0.1000    0.0218
##    10        0.3412             nan     0.1000    0.0204
##    20        0.1167             nan     0.1000    0.0064
##    40        0.0153             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0969
```

```
##      2        0.9056             nan     0.1000    0.0680
##      3        0.7878             nan     0.1000    0.0563
##      4        0.6902             nan     0.1000    0.0459
##      5        0.6079             nan     0.1000    0.0440
##      6        0.5376             nan     0.1000    0.0368
##      7        0.4770             nan     0.1000    0.0302
##      8        0.4244             nan     0.1000    0.0275
##      9        0.3783             nan     0.1000    0.0237
##     10        0.3379             nan     0.1000    0.0202
##     20        0.1158             nan     0.1000    0.0062
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0992
##      2        0.9056             nan     0.1000    0.0696
##      3        0.7878             nan     0.1000    0.0587
##      4        0.6902             nan     0.1000    0.0523
##      5        0.6079             nan     0.1000    0.0403
##      6        0.5376             nan     0.1000    0.0356
##      7        0.4770             nan     0.1000    0.0332
##      8        0.4244             nan     0.1000    0.0275
##      9        0.3783             nan     0.1000    0.0219
##     10        0.3379             nan     0.1000    0.0193
##     20        0.1158             nan     0.1000    0.0064
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.1015
##      2        0.9056             nan     0.1000    0.0760
##      3        0.7878             nan     0.1000    0.0599
##      4        0.6902             nan     0.1000    0.0514
```

```
##     5        0.6079             nan     0.1000    0.0418
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0246
##     9        0.3783             nan     0.1000    0.0226
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1015
##     2        0.9056             nan     0.1000    0.0696
##     3        0.7878             nan     0.1000    0.0635
##     4        0.6902             nan     0.1000    0.0496
##     5        0.6079             nan     0.1000    0.0396
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0267
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0210
##    20        0.1158             nan     0.1000    0.0061
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0809
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0496
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0362
##     7        0.4770             nan     0.1000    0.0312
```

```
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0219
##    10        0.3379             nan     0.1000    0.0213
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0776
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0447
##     6        0.5376             nan     0.1000    0.0279
##     7        0.4770             nan     0.1000    0.0317
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0233
##    10        0.3379             nan     0.1000    0.0199
##    20        0.1158             nan     0.1000    0.0067
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0807
##     2        0.9023             nan     0.1000    0.0713
##     3        0.7848             nan     0.1000    0.0612
##     4        0.6876             nan     0.1000    0.0496
##     5        0.6056             nan     0.1000    0.0432
##     6        0.5355             nan     0.1000    0.0368
##     7        0.4751             nan     0.1000    0.0312
##     8        0.4227             nan     0.1000    0.0279
##     9        0.3768             nan     0.1000    0.0233
##    10        0.3366             nan     0.1000    0.0195
```

```
##    20        0.1154           nan     0.1000    0.0066
##    40        0.0151           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0482           nan     0.1000    0.0902
##    2        0.9023           nan     0.1000    0.0746
##    3        0.7848           nan     0.1000    0.0599
##    4        0.6876           nan     0.1000    0.0514
##    5        0.6056           nan     0.1000    0.0432
##    6        0.5355           nan     0.1000    0.0350
##    7        0.4751           nan     0.1000    0.0277
##    8        0.4227           nan     0.1000    0.0262
##    9        0.3768           nan     0.1000    0.0257
##   10        0.3366           nan     0.1000    0.0207
##   20        0.1154           nan     0.1000    0.0066
##   40        0.0151           nan     0.1000    0.0008
##   60        0.0020           nan     0.1000    0.0001
##   80        0.0003           nan     0.1000    0.0000
##  100        0.0000           nan     0.1000    0.0000
##  120        0.0000           nan     0.1000    0.0000
##  140        0.0000           nan     0.1000    0.0000
##  150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0482           nan     0.1000    0.0973
##    2        0.9023           nan     0.1000    0.0778
##    3        0.7848           nan     0.1000    0.0551
##    4        0.6876           nan     0.1000    0.0486
##    5        0.6056           nan     0.1000    0.0417
##    6        0.5355           nan     0.1000    0.0332
##    7        0.4751           nan     0.1000    0.0317
##    8        0.4227           nan     0.1000    0.0250
##    9        0.3768           nan     0.1000    0.0233
##   10        0.3366           nan     0.1000    0.0192
##   20        0.1154           nan     0.1000    0.0060
##   40        0.0151           nan     0.1000    0.0008
##   60        0.0020           nan     0.1000    0.0001
```

```
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0760
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0523
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0339
##     7        0.4770             nan     0.1000    0.0288
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0240
##    10        0.3379             nan     0.1000    0.0196
##    20        0.1158             nan     0.1000    0.0061
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0899
##     2        0.9056             nan     0.1000    0.0680
##     3        0.7878             nan     0.1000    0.0611
##     4        0.6902             nan     0.1000    0.0459
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0292
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0243
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0067
##    40        0.0152             nan     0.1000    0.0007
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
```

```
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517            nan     0.1000    0.0922
##     2       0.9056            nan     0.1000    0.0696
##     3       0.7878            nan     0.1000    0.0587
##     4       0.6902            nan     0.1000    0.0514
##     5       0.6079            nan     0.1000    0.0425
##     6       0.5376            nan     0.1000    0.0351
##     7       0.4770            nan     0.1000    0.0292
##     8       0.4244            nan     0.1000    0.0263
##     9       0.3783            nan     0.1000    0.0226
##    10       0.3379            nan     0.1000    0.0196
##    20       0.1158            nan     0.1000    0.0059
##    40       0.0152            nan     0.1000    0.0008
##    60       0.0020            nan     0.1000    0.0001
##    80       0.0003            nan     0.1000    0.0000
##   100       0.0000            nan     0.1000    0.0000
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517            nan     0.1000    0.1015
##     2       0.9056            nan     0.1000    0.0583
##     3       0.7878            nan     0.1000    0.0551
##     4       0.6902            nan     0.1000    0.0496
##     5       0.6079            nan     0.1000    0.0440
##     6       0.5376            nan     0.1000    0.0368
##     7       0.4770            nan     0.1000    0.0302
##     8       0.4244            nan     0.1000    0.0279
##     9       0.3783            nan     0.1000    0.0237
##    10       0.3379            nan     0.1000    0.0208
##    20       0.1158            nan     0.1000    0.0063
##    40       0.0152            nan     0.1000    0.0008
##    60       0.0020            nan     0.1000    0.0001
##    80       0.0003            nan     0.1000    0.0000
##   100       0.0000            nan     0.1000    0.0000
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0899
##     2        0.9056             nan     0.1000    0.0809
##     3        0.7878             nan     0.1000    0.0647
##     4        0.6902             nan     0.1000    0.0486
##     5        0.6079             nan     0.1000    0.0454
##     6        0.5376             nan     0.1000    0.0345
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0254
##     9        0.3783             nan     0.1000    0.0240
##    10        0.3379             nan     0.1000    0.0196
##    20        0.1158             nan     0.1000    0.0067
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0899
##     2        0.9056             nan     0.1000    0.0728
##     3        0.7878             nan     0.1000    0.0527
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0432
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0234
##     9        0.3783             nan     0.1000    0.0243
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0067
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0899
##     2        0.9056             nan     0.1000    0.0776
```

```
##      3        0.7878             nan     0.1000    0.0611
##      4        0.6902             nan     0.1000    0.0496
##      5        0.6079             nan     0.1000    0.0381
##      6        0.5376             nan     0.1000    0.0321
##      7        0.4770             nan     0.1000    0.0302
##      8        0.4244             nan     0.1000    0.0275
##      9        0.3783             nan     0.1000    0.0247
##     10        0.3379             nan     0.1000    0.0210
##     20        0.1158             nan     0.1000    0.0064
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0969
##      2        0.9056             nan     0.1000    0.0825
##      3        0.7878             nan     0.1000    0.0599
##      4        0.6902             nan     0.1000    0.0486
##      5        0.6079             nan     0.1000    0.0432
##      6        0.5376             nan     0.1000    0.0368
##      7        0.4770             nan     0.1000    0.0292
##      8        0.4244             nan     0.1000    0.0279
##      9        0.3783             nan     0.1000    0.0233
##     10        0.3379             nan     0.1000    0.0213
##     20        0.1158             nan     0.1000    0.0065
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0922
##      2        0.9056             nan     0.1000    0.0631
##      3        0.7878             nan     0.1000    0.0587
##      4        0.6902             nan     0.1000    0.0505
##      5        0.6079             nan     0.1000    0.0418
```

```
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0233
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0065
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0647
##     3        0.7878             nan     0.1000    0.0563
##     4        0.6902             nan     0.1000    0.0514
##     5        0.6079             nan     0.1000    0.0396
##     6        0.5376             nan     0.1000    0.0327
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0279
##     9        0.3783             nan     0.1000    0.0243
##    10        0.3379             nan     0.1000    0.0199
##    20        0.1158             nan     0.1000    0.0067
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0992
##     2        0.9056             nan     0.1000    0.0728
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0468
##     5        0.6079             nan     0.1000    0.0440
##     6        0.5376             nan     0.1000    0.0339
##     7        0.4770             nan     0.1000    0.0292
##     8        0.4244             nan     0.1000    0.0275
```

```
##     9       0.3783             nan     0.1000    0.0226
##    10       0.3379             nan     0.1000    0.0199
##    20       0.1158             nan     0.1000    0.0063
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.0852
##     2       0.9056             nan     0.1000    0.0776
##     3       0.7878             nan     0.1000    0.0599
##     4       0.6902             nan     0.1000    0.0496
##     5       0.6079             nan     0.1000    0.0410
##     6       0.5376             nan     0.1000    0.0362
##     7       0.4770             nan     0.1000    0.0307
##     8       0.4244             nan     0.1000    0.0275
##     9       0.3783             nan     0.1000    0.0237
##    10       0.3379             nan     0.1000    0.0208
##    20       0.1158             nan     0.1000    0.0059
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0592             nan     0.1000    0.0915
##     2       0.9126             nan     0.1000    0.0694
##     3       0.7941             nan     0.1000    0.0610
##     4       0.6959             nan     0.1000    0.0505
##     5       0.6130             nan     0.1000    0.0432
##     6       0.5422             nan     0.1000    0.0357
##     7       0.4811             nan     0.1000    0.0289
##     8       0.4280             nan     0.1000    0.0267
##     9       0.3816             nan     0.1000    0.0244
##    10       0.3408             nan     0.1000    0.0208
##    20       0.1168             nan     0.1000    0.0069
```

```
##    40        0.0153             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0592             nan     0.1000    0.0938
##    2        0.9126             nan     0.1000    0.0819
##    3        0.7941             nan     0.1000    0.0610
##    4        0.6959             nan     0.1000    0.0496
##    5        0.6130             nan     0.1000    0.0425
##    6        0.5422             nan     0.1000    0.0345
##    7        0.4811             nan     0.1000    0.0317
##    8        0.4280             nan     0.1000    0.0271
##    9        0.3816             nan     0.1000    0.0227
##   10        0.3408             nan     0.1000    0.0199
##   20        0.1168             nan     0.1000    0.0057
##   40        0.0153             nan     0.1000    0.0009
##   60        0.0021             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0592             nan     0.1000    0.0960
##    2        0.9126             nan     0.1000    0.0757
##    3        0.7941             nan     0.1000    0.0552
##    4        0.6959             nan     0.1000    0.0487
##    5        0.6130             nan     0.1000    0.0404
##    6        0.5422             nan     0.1000    0.0345
##    7        0.4811             nan     0.1000    0.0298
##    8        0.4280             nan     0.1000    0.0263
##    9        0.3816             nan     0.1000    0.0224
##   10        0.3408             nan     0.1000    0.0202
##   20        0.1168             nan     0.1000    0.0061
##   40        0.0153             nan     0.1000    0.0009
##   60        0.0021             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
```

```
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0830
##     2        0.9023             nan     0.1000    0.0778
##     3        0.7848             nan     0.1000    0.0587
##     4        0.6876             nan     0.1000    0.0514
##     5        0.6056             nan     0.1000    0.0395
##     6        0.5355             nan     0.1000    0.0320
##     7        0.4751             nan     0.1000    0.0282
##     8        0.4227             nan     0.1000    0.0275
##     9        0.3768             nan     0.1000    0.0233
##    10        0.3366             nan     0.1000    0.0192
##    20        0.1154             nan     0.1000    0.0062
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0973
##     2        0.9023             nan     0.1000    0.0713
##     3        0.7848             nan     0.1000    0.0599
##     4        0.6876             nan     0.1000    0.0477
##     5        0.6056             nan     0.1000    0.0425
##     6        0.5355             nan     0.1000    0.0350
##     7        0.4751             nan     0.1000    0.0317
##     8        0.4227             nan     0.1000    0.0258
##     9        0.3768             nan     0.1000    0.0233
##    10        0.3366             nan     0.1000    0.0192
##    20        0.1154             nan     0.1000    0.0065
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
```

```
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0482           nan     0.1000    0.0902
##     2       0.9023           nan     0.1000    0.0697
##     3       0.7848           nan     0.1000    0.0551
##     4       0.6876           nan     0.1000    0.0496
##     5       0.6056           nan     0.1000    0.0403
##     6       0.5355           nan     0.1000    0.0368
##     7       0.4751           nan     0.1000    0.0322
##     8       0.4227           nan     0.1000    0.0271
##     9       0.3768           nan     0.1000    0.0229
##    10       0.3366           nan     0.1000    0.0195
##    20       0.1154           nan     0.1000    0.0061
##    40       0.0151           nan     0.1000    0.0008
##    60       0.0020           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517           nan     0.1000    0.1015
##     2       0.9056           nan     0.1000    0.0712
##     3       0.7878           nan     0.1000    0.0611
##     4       0.6902           nan     0.1000    0.0486
##     5       0.6079           nan     0.1000    0.0403
##     6       0.5376           nan     0.1000    0.0368
##     7       0.4770           nan     0.1000    0.0283
##     8       0.4244           nan     0.1000    0.0259
##     9       0.3783           nan     0.1000    0.0216
##    10       0.3379           nan     0.1000    0.0205
##    20       0.1158           nan     0.1000    0.0062
##    40       0.0152           nan     0.1000    0.0008
##    60       0.0020           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.0517             nan     0.1000    0.0922
##      2        0.9056             nan     0.1000    0.0760
##      3        0.7878             nan     0.1000    0.0611
##      4        0.6902             nan     0.1000    0.0496
##      5        0.6079             nan     0.1000    0.0418
##      6        0.5376             nan     0.1000    0.0351
##      7        0.4770             nan     0.1000    0.0283
##      8        0.4244             nan     0.1000    0.0250
##      9        0.3783             nan     0.1000    0.0230
##     10        0.3379             nan     0.1000    0.0205
##     20        0.1158             nan     0.1000    0.0065
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0992
##      2        0.9056             nan     0.1000    0.0696
##      3        0.7878             nan     0.1000    0.0539
##      4        0.6902             nan     0.1000    0.0496
##      5        0.6079             nan     0.1000    0.0418
##      6        0.5376             nan     0.1000    0.0339
##      7        0.4770             nan     0.1000    0.0312
##      8        0.4244             nan     0.1000    0.0275
##      9        0.3783             nan     0.1000    0.0216
##     10        0.3379             nan     0.1000    0.0205
##     20        0.1158             nan     0.1000    0.0063
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0969
##      2        0.9056             nan     0.1000    0.0776
##      3        0.7878             nan     0.1000    0.0563
```

```
##      4        0.6902             nan     0.1000    0.0505
##      5        0.6079             nan     0.1000    0.0440
##      6        0.5376             nan     0.1000    0.0339
##      7        0.4770             nan     0.1000    0.0278
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0243
##     10        0.3379             nan     0.1000    0.0216
##     20        0.1158             nan     0.1000    0.0064
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0805
##      2        0.9056             nan     0.1000    0.0663
##      3        0.7878             nan     0.1000    0.0587
##      4        0.6902             nan     0.1000    0.0477
##      5        0.6079             nan     0.1000    0.0440
##      6        0.5376             nan     0.1000    0.0345
##      7        0.4770             nan     0.1000    0.0302
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0230
##     10        0.3379             nan     0.1000    0.0196
##     20        0.1158             nan     0.1000    0.0060
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.1062
##      2        0.9056             nan     0.1000    0.0793
##      3        0.7878             nan     0.1000    0.0575
##      4        0.6902             nan     0.1000    0.0477
##      5        0.6079             nan     0.1000    0.0396
##      6        0.5376             nan     0.1000    0.0351
```

```
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0279
##     9        0.3783             nan     0.1000    0.0243
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0007
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0829
##     2        0.9056             nan     0.1000    0.0760
##     3        0.7878             nan     0.1000    0.0551
##     4        0.6902             nan     0.1000    0.0514
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0254
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0829
##     2        0.9056             nan     0.1000    0.0680
##     3        0.7878             nan     0.1000    0.0659
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0345
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0263
##     9        0.3783             nan     0.1000    0.0240
```

```
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0922
##    2        0.9056             nan     0.1000    0.0728
##    3        0.7878             nan     0.1000    0.0587
##    4        0.6902             nan     0.1000    0.0468
##    5        0.6079             nan     0.1000    0.0403
##    6        0.5376             nan     0.1000    0.0368
##    7        0.4770             nan     0.1000    0.0327
##    8        0.4244             nan     0.1000    0.0275
##    9        0.3783             nan     0.1000    0.0230
##   10        0.3379             nan     0.1000    0.0187
##   20        0.1158             nan     0.1000    0.0065
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0992
##    2        0.9056             nan     0.1000    0.0712
##    3        0.7878             nan     0.1000    0.0599
##    4        0.6902             nan     0.1000    0.0496
##    5        0.6079             nan     0.1000    0.0403
##    6        0.5376             nan     0.1000    0.0362
##    7        0.4770             nan     0.1000    0.0273
##    8        0.4244             nan     0.1000    0.0275
##    9        0.3783             nan     0.1000    0.0243
##   10        0.3379             nan     0.1000    0.0202
##   20        0.1158             nan     0.1000    0.0062
##   40        0.0152             nan     0.1000    0.0007
```

```
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.1086
##    2        0.9056             nan     0.1000    0.0776
##    3        0.7878             nan     0.1000    0.0539
##    4        0.6902             nan     0.1000    0.0486
##    5        0.6079             nan     0.1000    0.0432
##    6        0.5376             nan     0.1000    0.0380
##    7        0.4770             nan     0.1000    0.0288
##    8        0.4244             nan     0.1000    0.0275
##    9        0.3783             nan     0.1000    0.0226
##   10        0.3379             nan     0.1000    0.0208
##   20        0.1158             nan     0.1000    0.0063
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0945
##    2        0.9056             nan     0.1000    0.0696
##    3        0.7878             nan     0.1000    0.0551
##    4        0.6902             nan     0.1000    0.0459
##    5        0.6079             nan     0.1000    0.0432
##    6        0.5376             nan     0.1000    0.0351
##    7        0.4770             nan     0.1000    0.0332
##    8        0.4244             nan     0.1000    0.0275
##    9        0.3783             nan     0.1000    0.0230
##   10        0.3379             nan     0.1000    0.0199
##   20        0.1158             nan     0.1000    0.0063
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
```

```
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517            nan     0.1000    0.0969
##     2        0.9056            nan     0.1000    0.0663
##     3        0.7878            nan     0.1000    0.0611
##     4        0.6902            nan     0.1000    0.0496
##     5        0.6079            nan     0.1000    0.0418
##     6        0.5376            nan     0.1000    0.0327
##     7        0.4770            nan     0.1000    0.0337
##     8        0.4244            nan     0.1000    0.0267
##     9        0.3783            nan     0.1000    0.0226
##    10        0.3379            nan     0.1000    0.0199
##    20        0.1158            nan     0.1000    0.0059
##    40        0.0152            nan     0.1000    0.0008
##    60        0.0020            nan     0.1000    0.0001
##    80        0.0003            nan     0.1000    0.0000
##   100        0.0000            nan     0.1000    0.0000
##   120        0.0000            nan     0.1000    0.0000
##   140        0.0000            nan     0.1000    0.0000
##   150        0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517            nan     0.1000    0.0969
##     2        0.9056            nan     0.1000    0.0680
##     3        0.7878            nan     0.1000    0.0599
##     4        0.6902            nan     0.1000    0.0505
##     5        0.6079            nan     0.1000    0.0388
##     6        0.5376            nan     0.1000    0.0345
##     7        0.4770            nan     0.1000    0.0342
##     8        0.4244            nan     0.1000    0.0279
##     9        0.3783            nan     0.1000    0.0230
##    10        0.3379            nan     0.1000    0.0199
##    20        0.1158            nan     0.1000    0.0061
##    40        0.0152            nan     0.1000    0.0008
##    60        0.0020            nan     0.1000    0.0001
##    80        0.0003            nan     0.1000    0.0000
##   100        0.0000            nan     0.1000    0.0000
##   120        0.0000            nan     0.1000    0.0000
##   140        0.0000            nan     0.1000    0.0000
##   150        0.0000            nan     0.1000    0.0000
```

```
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0969
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0563
##     4        0.6902             nan     0.1000    0.0532
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0374
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0237
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0592             nan     0.1000    0.0938
##     2        0.9126             nan     0.1000    0.0710
##     3        0.7941             nan     0.1000    0.0540
##     4        0.6959             nan     0.1000    0.0522
##     5        0.6130             nan     0.1000    0.0404
##     6        0.5422             nan     0.1000    0.0345
##     7        0.4811             nan     0.1000    0.0289
##     8        0.4280             nan     0.1000    0.0251
##     9        0.3816             nan     0.1000    0.0240
##    10        0.3408             nan     0.1000    0.0191
##    20        0.1168             nan     0.1000    0.0064
##    40        0.0153             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0592             nan     0.1000    0.0915
```

```
##      2        0.9126             nan     0.1000    0.0819
##      3        0.7941             nan     0.1000    0.0610
##      4        0.6959             nan     0.1000    0.0487
##      5        0.6130             nan     0.1000    0.0439
##      6        0.5422             nan     0.1000    0.0351
##      7        0.4811             nan     0.1000    0.0289
##      8        0.4280             nan     0.1000    0.0251
##      9        0.3816             nan     0.1000    0.0240
##     10        0.3408             nan     0.1000    0.0208
##     20        0.1168             nan     0.1000    0.0064
##     40        0.0153             nan     0.1000    0.0008
##     60        0.0021             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0592             nan     0.1000    0.0983
##      2        0.9126             nan     0.1000    0.0757
##      3        0.7941             nan     0.1000    0.0633
##      4        0.6959             nan     0.1000    0.0469
##      5        0.6130             nan     0.1000    0.0411
##      6        0.5422             nan     0.1000    0.0345
##      7        0.4811             nan     0.1000    0.0294
##      8        0.4280             nan     0.1000    0.0271
##      9        0.3816             nan     0.1000    0.0220
##     10        0.3408             nan     0.1000    0.0197
##     20        0.1168             nan     0.1000    0.0066
##     40        0.0153             nan     0.1000    0.0008
##     60        0.0021             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0482             nan     0.1000    0.0783
##      2        0.9023             nan     0.1000    0.0615
##      3        0.7848             nan     0.1000    0.0636
##      4        0.6876             nan     0.1000    0.0449
```

```
##      5        0.6056             nan     0.1000    0.0440
##      6        0.5355             nan     0.1000    0.0362
##      7        0.4751             nan     0.1000    0.0317
##      8        0.4227             nan     0.1000    0.0254
##      9        0.3768             nan     0.1000    0.0257
##     10        0.3366             nan     0.1000    0.0204
##     20        0.1154             nan     0.1000    0.0062
##     40        0.0151             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0482             nan     0.1000    0.1044
##      2        0.9023             nan     0.1000    0.0778
##      3        0.7848             nan     0.1000    0.0660
##      4        0.6876             nan     0.1000    0.0496
##      5        0.6056             nan     0.1000    0.0417
##      6        0.5355             nan     0.1000    0.0338
##      7        0.4751             nan     0.1000    0.0317
##      8        0.4227             nan     0.1000    0.0296
##      9        0.3768             nan     0.1000    0.0219
##     10        0.3366             nan     0.1000    0.0207
##     20        0.1154             nan     0.1000    0.0063
##     40        0.0151             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0482             nan     0.1000    0.0996
##      2        0.9023             nan     0.1000    0.0762
##      3        0.7848             nan     0.1000    0.0563
##      4        0.6876             nan     0.1000    0.0496
##      5        0.6056             nan     0.1000    0.0417
##      6        0.5355             nan     0.1000    0.0368
##      7        0.4751             nan     0.1000    0.0292
```

```
##     8        0.4227             nan     0.1000    0.0266
##     9        0.3768             nan     0.1000    0.0222
##    10        0.3366             nan     0.1000    0.0195
##    20        0.1154             nan     0.1000    0.0062
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0925
##     2        0.9023             nan     0.1000    0.0713
##     3        0.7848             nan     0.1000    0.0575
##     4        0.6876             nan     0.1000    0.0486
##     5        0.6056             nan     0.1000    0.0417
##     6        0.5355             nan     0.1000    0.0344
##     7        0.4751             nan     0.1000    0.0312
##     8        0.4227             nan     0.1000    0.0262
##     9        0.3768             nan     0.1000    0.0222
##    10        0.3366             nan     0.1000    0.0207
##    20        0.1154             nan     0.1000    0.0062
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0830
##     2        0.9023             nan     0.1000    0.0778
##     3        0.7848             nan     0.1000    0.0599
##     4        0.6876             nan     0.1000    0.0468
##     5        0.6056             nan     0.1000    0.0440
##     6        0.5355             nan     0.1000    0.0362
##     7        0.4751             nan     0.1000    0.0312
##     8        0.4227             nan     0.1000    0.0262
##     9        0.3768             nan     0.1000    0.0247
##    10        0.3366             nan     0.1000    0.0210
```

```
##    20        0.1154           nan     0.1000    0.0062
##    40        0.0151           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0482           nan     0.1000    0.0925
##    2        0.9023           nan     0.1000    0.0778
##    3        0.7848           nan     0.1000    0.0563
##    4        0.6876           nan     0.1000    0.0468
##    5        0.6056           nan     0.1000    0.0395
##    6        0.5355           nan     0.1000    0.0326
##    7        0.4751           nan     0.1000    0.0297
##    8        0.4227           nan     0.1000    0.0271
##    9        0.3768           nan     0.1000    0.0243
##   10        0.3366           nan     0.1000    0.0195
##   20        0.1154           nan     0.1000    0.0062
##   40        0.0151           nan     0.1000    0.0008
##   60        0.0020           nan     0.1000    0.0001
##   80        0.0003           nan     0.1000    0.0000
##  100        0.0000           nan     0.1000    0.0000
##  120        0.0000           nan     0.1000    0.0000
##  140        0.0000           nan     0.1000    0.0000
##  150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517           nan     0.1000    0.0852
##    2        0.9056           nan     0.1000    0.0744
##    3        0.7878           nan     0.1000    0.0551
##    4        0.6902           nan     0.1000    0.0486
##    5        0.6079           nan     0.1000    0.0418
##    6        0.5376           nan     0.1000    0.0356
##    7        0.4770           nan     0.1000    0.0292
##    8        0.4244           nan     0.1000    0.0259
##    9        0.3783           nan     0.1000    0.0233
##   10        0.3379           nan     0.1000    0.0199
##   20        0.1158           nan     0.1000    0.0064
##   40        0.0152           nan     0.1000    0.0008
##   60        0.0020           nan     0.1000    0.0001
```

```
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0969
##     2        0.9056             nan     0.1000    0.0809
##     3        0.7878             nan     0.1000    0.0647
##     4        0.6902             nan     0.1000    0.0450
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0240
##    10        0.3379             nan     0.1000    0.0199
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0922
##     2        0.9056             nan     0.1000    0.0857
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0312
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0226
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
```

```
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517            nan     0.1000    0.0992
##     2       0.9056            nan     0.1000    0.0809
##     3       0.7878            nan     0.1000    0.0599
##     4       0.6902            nan     0.1000    0.0477
##     5       0.6079            nan     0.1000    0.0425
##     6       0.5376            nan     0.1000    0.0351
##     7       0.4770            nan     0.1000    0.0307
##     8       0.4244            nan     0.1000    0.0283
##     9       0.3783            nan     0.1000    0.0230
##    10       0.3379            nan     0.1000    0.0208
##    20       0.1158            nan     0.1000    0.0067
##    40       0.0152            nan     0.1000    0.0008
##    60       0.0020            nan     0.1000    0.0001
##    80       0.0003            nan     0.1000    0.0000
##   100       0.0000            nan     0.1000    0.0000
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517            nan     0.1000    0.0922
##     2       0.9056            nan     0.1000    0.0728
##     3       0.7878            nan     0.1000    0.0611
##     4       0.6902            nan     0.1000    0.0496
##     5       0.6079            nan     0.1000    0.0403
##     6       0.5376            nan     0.1000    0.0362
##     7       0.4770            nan     0.1000    0.0297
##     8       0.4244            nan     0.1000    0.0263
##     9       0.3783            nan     0.1000    0.0212
##    10       0.3379            nan     0.1000    0.0213
##    20       0.1158            nan     0.1000    0.0064
##    40       0.0152            nan     0.1000    0.0008
##    60       0.0020            nan     0.1000    0.0001
##    80       0.0003            nan     0.1000    0.0000
##   100       0.0000            nan     0.1000    0.0000
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1015
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0635
##     4        0.6902             nan     0.1000    0.0532
##     5        0.6079             nan     0.1000    0.0418
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0275
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0199
##    20        0.1158             nan     0.1000    0.0067
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0922
##     2        0.9056             nan     0.1000    0.0809
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0486
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0327
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0267
##     9        0.3783             nan     0.1000    0.0223
##    10        0.3379             nan     0.1000    0.0184
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0663
```

```
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0468
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0288
##     8        0.4244             nan     0.1000    0.0267
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0193
##    20        0.1158             nan     0.1000    0.0060
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0922
##     2        0.9056             nan     0.1000    0.0712
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0339
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0247
##    10        0.3379             nan     0.1000    0.0219
##    20        0.1158             nan     0.1000    0.0068
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0659
##     4        0.6902             nan     0.1000    0.0450
##     5        0.6079             nan     0.1000    0.0410
```

```
##     6        0.5376             nan     0.1000    0.0333
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0240
##    10        0.3379             nan     0.1000    0.0208
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0712
##     3        0.7878             nan     0.1000    0.0623
##     4        0.6902             nan     0.1000    0.0468
##     5        0.6079             nan     0.1000    0.0396
##     6        0.5376             nan     0.1000    0.0368
##     7        0.4770             nan     0.1000    0.0312
##     8        0.4244             nan     0.1000    0.0263
##     9        0.3783             nan     0.1000    0.0243
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0066
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0712
##     3        0.7878             nan     0.1000    0.0527
##     4        0.6902             nan     0.1000    0.0523
##     5        0.6079             nan     0.1000    0.0418
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0267
```

```
##     9       0.3783             nan     0.1000    0.0226
##    10       0.3379             nan     0.1000    0.0208
##    20       0.1158             nan     0.1000    0.0061
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.1015
##     2       0.9056             nan     0.1000    0.0760
##     3       0.7878             nan     0.1000    0.0587
##     4       0.6902             nan     0.1000    0.0468
##     5       0.6079             nan     0.1000    0.0447
##     6       0.5376             nan     0.1000    0.0339
##     7       0.4770             nan     0.1000    0.0292
##     8       0.4244             nan     0.1000    0.0287
##     9       0.3783             nan     0.1000    0.0226
##    10       0.3379             nan     0.1000    0.0205
##    20       0.1158             nan     0.1000    0.0061
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.0992
##     2       0.9056             nan     0.1000    0.0696
##     3       0.7878             nan     0.1000    0.0539
##     4       0.6902             nan     0.1000    0.0486
##     5       0.6079             nan     0.1000    0.0418
##     6       0.5376             nan     0.1000    0.0333
##     7       0.4770             nan     0.1000    0.0292
##     8       0.4244             nan     0.1000    0.0263
##     9       0.3783             nan     0.1000    0.0216
##    10       0.3379             nan     0.1000    0.0199
##    20       0.1158             nan     0.1000    0.0063
```

```
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0992
##    2        0.9056             nan     0.1000    0.0696
##    3        0.7878             nan     0.1000    0.0563
##    4        0.6902             nan     0.1000    0.0486
##    5        0.6079             nan     0.1000    0.0396
##    6        0.5376             nan     0.1000    0.0356
##    7        0.4770             nan     0.1000    0.0288
##    8        0.4244             nan     0.1000    0.0287
##    9        0.3783             nan     0.1000    0.0233
##   10        0.3379             nan     0.1000    0.0199
##   20        0.1158             nan     0.1000    0.0062
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0992
##    2        0.9056             nan     0.1000    0.0712
##    3        0.7878             nan     0.1000    0.0599
##    4        0.6902             nan     0.1000    0.0496
##    5        0.6079             nan     0.1000    0.0432
##    6        0.5376             nan     0.1000    0.0345
##    7        0.4770             nan     0.1000    0.0283
##    8        0.4244             nan     0.1000    0.0259
##    9        0.3783             nan     0.1000    0.0230
##   10        0.3379             nan     0.1000    0.0213
##   20        0.1158             nan     0.1000    0.0061
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
```

```
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0712
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0425
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0312
##     8        0.4244             nan     0.1000    0.0254
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0190
##    20        0.1158             nan     0.1000    0.0068
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0728
##     3        0.7878             nan     0.1000    0.0563
##     4        0.6902             nan     0.1000    0.0486
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0356
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0275
##     9        0.3783             nan     0.1000    0.0219
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0065
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
```

```
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0558           nan     0.1000    0.0912
##     2       0.9094           nan     0.1000    0.0706
##     3       0.7912           nan     0.1000    0.0618
##     4       0.6933           nan     0.1000    0.0511
##     5       0.6107           nan     0.1000    0.0394
##     6       0.5401           nan     0.1000    0.0355
##     7       0.4792           nan     0.1000    0.0316
##     8       0.4263           nan     0.1000    0.0262
##     9       0.3801           nan     0.1000    0.0236
##    10       0.3395           nan     0.1000    0.0204
##    20       0.1164           nan     0.1000    0.0070
##    40       0.0153           nan     0.1000    0.0008
##    60       0.0021           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0558           nan     0.1000    0.1047
##     2       0.9094           nan     0.1000    0.0737
##     3       0.7912           nan     0.1000    0.0606
##     4       0.6933           nan     0.1000    0.0484
##     5       0.6107           nan     0.1000    0.0423
##     6       0.5401           nan     0.1000    0.0349
##     7       0.4792           nan     0.1000    0.0306
##     8       0.4263           nan     0.1000    0.0270
##     9       0.3801           nan     0.1000    0.0229
##    10       0.3395           nan     0.1000    0.0201
##    20       0.1164           nan     0.1000    0.0061
##    40       0.0153           nan     0.1000    0.0008
##    60       0.0021           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.0558             nan     0.1000    0.1070
##      2        0.9094             nan     0.1000    0.0659
##      3        0.7912             nan     0.1000    0.0583
##      4        0.6933             nan     0.1000    0.0520
##      5        0.6107             nan     0.1000    0.0444
##      6        0.5401             nan     0.1000    0.0343
##      7        0.4792             nan     0.1000    0.0292
##      8        0.4263             nan     0.1000    0.0278
##      9        0.3801             nan     0.1000    0.0232
##     10        0.3395             nan     0.1000    0.0221
##     20        0.1164             nan     0.1000    0.0064
##     40        0.0153             nan     0.1000    0.0008
##     60        0.0021             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0922
##      2        0.9056             nan     0.1000    0.0728
##      3        0.7878             nan     0.1000    0.0539
##      4        0.6902             nan     0.1000    0.0505
##      5        0.6079             nan     0.1000    0.0396
##      6        0.5376             nan     0.1000    0.0309
##      7        0.4770             nan     0.1000    0.0297
##      8        0.4244             nan     0.1000    0.0259
##      9        0.3783             nan     0.1000    0.0223
##     10        0.3379             nan     0.1000    0.0202
##     20        0.1158             nan     0.1000    0.0065
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0829
##      2        0.9056             nan     0.1000    0.0809
##      3        0.7878             nan     0.1000    0.0599
```

```
##     4        0.6902             nan     0.1000    0.0496
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0315
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0279
##     9        0.3783             nan     0.1000    0.0233
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0992
##     2        0.9056             nan     0.1000    0.0631
##     3        0.7878             nan     0.1000    0.0599
##     4        0.6902             nan     0.1000    0.0477
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0307
##     8        0.4244             nan     0.1000    0.0254
##     9        0.3783             nan     0.1000    0.0226
##    10        0.3379             nan     0.1000    0.0219
##    20        0.1158             nan     0.1000    0.0064
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0996
##     2        0.9023             nan     0.1000    0.0795
##     3        0.7848             nan     0.1000    0.0527
##     4        0.6876             nan     0.1000    0.0468
##     5        0.6056             nan     0.1000    0.0388
##     6        0.5355             nan     0.1000    0.0356
```

```
##     7        0.4751             nan     0.1000    0.0307
##     8        0.4227             nan     0.1000    0.0250
##     9        0.3768             nan     0.1000    0.0240
##    10        0.3366             nan     0.1000    0.0201
##    20        0.1154             nan     0.1000    0.0059
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0902
##     2        0.9023             nan     0.1000    0.0778
##     3        0.7848             nan     0.1000    0.0648
##     4        0.6876             nan     0.1000    0.0524
##     5        0.6056             nan     0.1000    0.0388
##     6        0.5355             nan     0.1000    0.0362
##     7        0.4751             nan     0.1000    0.0317
##     8        0.4227             nan     0.1000    0.0254
##     9        0.3768             nan     0.1000    0.0222
##    10        0.3366             nan     0.1000    0.0192
##    20        0.1154             nan     0.1000    0.0065
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0949
##     2        0.9023             nan     0.1000    0.0647
##     3        0.7848             nan     0.1000    0.0624
##     4        0.6876             nan     0.1000    0.0440
##     5        0.6056             nan     0.1000    0.0432
##     6        0.5355             nan     0.1000    0.0326
##     7        0.4751             nan     0.1000    0.0297
##     8        0.4227             nan     0.1000    0.0262
##     9        0.3768             nan     0.1000    0.0240
```

```
##    10        0.3366             nan     0.1000    0.0192
##    20        0.1154             nan     0.1000    0.0059
##    40        0.0151             nan     0.1000    0.0007
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.1039
##    2        0.9056             nan     0.1000    0.0728
##    3        0.7878             nan     0.1000    0.0659
##    4        0.6902             nan     0.1000    0.0496
##    5        0.6079             nan     0.1000    0.0425
##    6        0.5376             nan     0.1000    0.0362
##    7        0.4770             nan     0.1000    0.0302
##    8        0.4244             nan     0.1000    0.0267
##    9        0.3783             nan     0.1000    0.0230
##   10        0.3379             nan     0.1000    0.0199
##   20        0.1158             nan     0.1000    0.0065
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0922
##    2        0.9056             nan     0.1000    0.0712
##    3        0.7878             nan     0.1000    0.0587
##    4        0.6902             nan     0.1000    0.0459
##    5        0.6079             nan     0.1000    0.0388
##    6        0.5376             nan     0.1000    0.0345
##    7        0.4770             nan     0.1000    0.0312
##    8        0.4244             nan     0.1000    0.0267
##    9        0.3783             nan     0.1000    0.0219
##   10        0.3379             nan     0.1000    0.0205
##   20        0.1158             nan     0.1000    0.0063
##   40        0.0152             nan     0.1000    0.0008
```

```
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1       1.0517             nan     0.1000    0.1086
##    2       0.9056             nan     0.1000    0.0744
##    3       0.7878             nan     0.1000    0.0575
##    4       0.6902             nan     0.1000    0.0505
##    5       0.6079             nan     0.1000    0.0403
##    6       0.5376             nan     0.1000    0.0333
##    7       0.4770             nan     0.1000    0.0292
##    8       0.4244             nan     0.1000    0.0263
##    9       0.3783             nan     0.1000    0.0230
##   10       0.3379             nan     0.1000    0.0205
##   20       0.1158             nan     0.1000    0.0067
##   40       0.0152             nan     0.1000    0.0008
##   60       0.0020             nan     0.1000    0.0001
##   80       0.0003             nan     0.1000    0.0000
##  100       0.0000             nan     0.1000    0.0000
##  120       0.0000             nan     0.1000    0.0000
##  140       0.0000             nan     0.1000    0.0000
##  150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1       1.0517             nan     0.1000    0.1015
##    2       0.9056             nan     0.1000    0.0760
##    3       0.7878             nan     0.1000    0.0647
##    4       0.6902             nan     0.1000    0.0468
##    5       0.6079             nan     0.1000    0.0418
##    6       0.5376             nan     0.1000    0.0362
##    7       0.4770             nan     0.1000    0.0312
##    8       0.4244             nan     0.1000    0.0267
##    9       0.3783             nan     0.1000    0.0223
##   10       0.3379             nan     0.1000    0.0202
##   20       0.1158             nan     0.1000    0.0064
##   40       0.0152             nan     0.1000    0.0008
##   60       0.0020             nan     0.1000    0.0001
##   80       0.0003             nan     0.1000    0.0000
##  100       0.0000             nan     0.1000    0.0000
```

```
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517           nan     0.1000    0.0969
##     2        0.9056           nan     0.1000    0.0663
##     3        0.7878           nan     0.1000    0.0575
##     4        0.6902           nan     0.1000    0.0450
##     5        0.6079           nan     0.1000    0.0410
##     6        0.5376           nan     0.1000    0.0351
##     7        0.4770           nan     0.1000    0.0292
##     8        0.4244           nan     0.1000    0.0263
##     9        0.3783           nan     0.1000    0.0226
##    10        0.3379           nan     0.1000    0.0202
##    20        0.1158           nan     0.1000    0.0062
##    40        0.0152           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517           nan     0.1000    0.0899
##     2        0.9056           nan     0.1000    0.0680
##     3        0.7878           nan     0.1000    0.0587
##     4        0.6902           nan     0.1000    0.0477
##     5        0.6079           nan     0.1000    0.0396
##     6        0.5376           nan     0.1000    0.0345
##     7        0.4770           nan     0.1000    0.0292
##     8        0.4244           nan     0.1000    0.0238
##     9        0.3783           nan     0.1000    0.0223
##    10        0.3379           nan     0.1000    0.0208
##    20        0.1158           nan     0.1000    0.0061
##    40        0.0152           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
```

```
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1015
##     2        0.9056             nan     0.1000    0.0760
##     3        0.7878             nan     0.1000    0.0551
##     4        0.6902             nan     0.1000    0.0505
##     5        0.6079             nan     0.1000    0.0432
##     6        0.5376             nan     0.1000    0.0374
##     7        0.4770             nan     0.1000    0.0283
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0233
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0065
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0945
##     2        0.9056             nan     0.1000    0.0776
##     3        0.7878             nan     0.1000    0.0575
##     4        0.6902             nan     0.1000    0.0486
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0362
##     7        0.4770             nan     0.1000    0.0288
##     8        0.4244             nan     0.1000    0.0283
##     9        0.3783             nan     0.1000    0.0219
##    10        0.3379             nan     0.1000    0.0222
##    20        0.1158             nan     0.1000    0.0066
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0922
```

```
##      2        0.9056             nan     0.1000    0.0663
##      3        0.7878             nan     0.1000    0.0659
##      4        0.6902             nan     0.1000    0.0486
##      5        0.6079             nan     0.1000    0.0440
##      6        0.5376             nan     0.1000    0.0345
##      7        0.4770             nan     0.1000    0.0307
##      8        0.4244             nan     0.1000    0.0275
##      9        0.3783             nan     0.1000    0.0230
##     10        0.3379             nan     0.1000    0.0196
##     20        0.1158             nan     0.1000    0.0064
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0899
##      2        0.9056             nan     0.1000    0.0760
##      3        0.7878             nan     0.1000    0.0575
##      4        0.6902             nan     0.1000    0.0514
##      5        0.6079             nan     0.1000    0.0418
##      6        0.5376             nan     0.1000    0.0351
##      7        0.4770             nan     0.1000    0.0297
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0223
##     10        0.3379             nan     0.1000    0.0193
##     20        0.1158             nan     0.1000    0.0063
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0021             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0969
##      2        0.9056             nan     0.1000    0.0696
##      3        0.7878             nan     0.1000    0.0575
##      4        0.6902             nan     0.1000    0.0468
```

```
##     5        0.6079             nan     0.1000    0.0403
##     6        0.5376             nan     0.1000    0.0362
##     7        0.4770             nan     0.1000    0.0312
##     8        0.4244             nan     0.1000    0.0263
##     9        0.3783             nan     0.1000    0.0233
##    10        0.3379             nan     0.1000    0.0205
##    20        0.1158             nan     0.1000    0.0059
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1062
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0527
##     4        0.6902             nan     0.1000    0.0459
##     5        0.6079             nan     0.1000    0.0454
##     6        0.5376             nan     0.1000    0.0351
##     7        0.4770             nan     0.1000    0.0292
##     8        0.4244             nan     0.1000    0.0267
##     9        0.3783             nan     0.1000    0.0223
##    10        0.3379             nan     0.1000    0.0210
##    20        0.1158             nan     0.1000    0.0060
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0854
##     2        0.9023             nan     0.1000    0.0795
##     3        0.7848             nan     0.1000    0.0575
##     4        0.6876             nan     0.1000    0.0458
##     5        0.6056             nan     0.1000    0.0403
##     6        0.5355             nan     0.1000    0.0320
##     7        0.4751             nan     0.1000    0.0272
```

```
##     8        0.4227             nan     0.1000    0.0258
##     9        0.3768             nan     0.1000    0.0236
##    10        0.3366             nan     0.1000    0.0201
##    20        0.1154             nan     0.1000    0.0062
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0949
##     2        0.9023             nan     0.1000    0.0729
##     3        0.7848             nan     0.1000    0.0599
##     4        0.6876             nan     0.1000    0.0486
##     5        0.6056             nan     0.1000    0.0403
##     6        0.5355             nan     0.1000    0.0344
##     7        0.4751             nan     0.1000    0.0302
##     8        0.4227             nan     0.1000    0.0242
##     9        0.3768             nan     0.1000    0.0226
##    10        0.3366             nan     0.1000    0.0207
##    20        0.1154             nan     0.1000    0.0066
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0949
##     2        0.9023             nan     0.1000    0.0778
##     3        0.7848             nan     0.1000    0.0612
##     4        0.6876             nan     0.1000    0.0477
##     5        0.6056             nan     0.1000    0.0462
##     6        0.5355             nan     0.1000    0.0338
##     7        0.4751             nan     0.1000    0.0297
##     8        0.4227             nan     0.1000    0.0262
##     9        0.3768             nan     0.1000    0.0212
##    10        0.3366             nan     0.1000    0.0201
```

```
##    20        0.1154           nan     0.1000    0.0059
##    40        0.0151           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517           nan     0.1000    0.1015
##    2        0.9056           nan     0.1000    0.0728
##    3        0.7878           nan     0.1000    0.0671
##    4        0.6902           nan     0.1000    0.0468
##    5        0.6079           nan     0.1000    0.0425
##    6        0.5376           nan     0.1000    0.0356
##    7        0.4770           nan     0.1000    0.0297
##    8        0.4244           nan     0.1000    0.0259
##    9        0.3783           nan     0.1000    0.0240
##   10        0.3379           nan     0.1000    0.0210
##   20        0.1158           nan     0.1000    0.0065
##   40        0.0152           nan     0.1000    0.0008
##   60        0.0020           nan     0.1000    0.0001
##   80        0.0003           nan     0.1000    0.0000
##  100        0.0000           nan     0.1000    0.0000
##  120        0.0000           nan     0.1000    0.0000
##  140        0.0000           nan     0.1000    0.0000
##  150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517           nan     0.1000    0.0875
##    2        0.9056           nan     0.1000    0.0728
##    3        0.7878           nan     0.1000    0.0575
##    4        0.6902           nan     0.1000    0.0505
##    5        0.6079           nan     0.1000    0.0418
##    6        0.5376           nan     0.1000    0.0351
##    7        0.4770           nan     0.1000    0.0307
##    8        0.4244           nan     0.1000    0.0259
##    9        0.3783           nan     0.1000    0.0216
##   10        0.3379           nan     0.1000    0.0196
##   20        0.1158           nan     0.1000    0.0061
##   40        0.0152           nan     0.1000    0.0008
##   60        0.0020           nan     0.1000    0.0001
```

```
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.1039
##    2        0.9056             nan     0.1000    0.0776
##    3        0.7878             nan     0.1000    0.0599
##    4        0.6902             nan     0.1000    0.0477
##    5        0.6079             nan     0.1000    0.0396
##    6        0.5376             nan     0.1000    0.0356
##    7        0.4770             nan     0.1000    0.0292
##    8        0.4244             nan     0.1000    0.0242
##    9        0.3783             nan     0.1000    0.0230
##   10        0.3379             nan     0.1000    0.0213
##   20        0.1158             nan     0.1000    0.0064
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0899
##    2        0.9056             nan     0.1000    0.0776
##    3        0.7878             nan     0.1000    0.0587
##    4        0.6902             nan     0.1000    0.0532
##    5        0.6079             nan     0.1000    0.0403
##    6        0.5376             nan     0.1000    0.0368
##    7        0.4770             nan     0.1000    0.0307
##    8        0.4244             nan     0.1000    0.0279
##    9        0.3783             nan     0.1000    0.0233
##   10        0.3379             nan     0.1000    0.0208
##   20        0.1158             nan     0.1000    0.0064
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
```

```
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517            nan     0.1000    0.0899
##     2       0.9056            nan     0.1000    0.0696
##     3       0.7878            nan     0.1000    0.0623
##     4       0.6902            nan     0.1000    0.0413
##     5       0.6079            nan     0.1000    0.0403
##     6       0.5376            nan     0.1000    0.0362
##     7       0.4770            nan     0.1000    0.0302
##     8       0.4244            nan     0.1000    0.0275
##     9       0.3783            nan     0.1000    0.0233
##    10       0.3379            nan     0.1000    0.0208
##    20       0.1158            nan     0.1000    0.0063
##    40       0.0152            nan     0.1000    0.0008
##    60       0.0020            nan     0.1000    0.0001
##    80       0.0003            nan     0.1000    0.0000
##   100       0.0000            nan     0.1000    0.0000
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517            nan     0.1000    0.0992
##     2       0.9056            nan     0.1000    0.0744
##     3       0.7878            nan     0.1000    0.0647
##     4       0.6902            nan     0.1000    0.0523
##     5       0.6079            nan     0.1000    0.0410
##     6       0.5376            nan     0.1000    0.0333
##     7       0.4770            nan     0.1000    0.0297
##     8       0.4244            nan     0.1000    0.0259
##     9       0.3783            nan     0.1000    0.0237
##    10       0.3379            nan     0.1000    0.0205
##    20       0.1158            nan     0.1000    0.0061
##    40       0.0152            nan     0.1000    0.0008
##    60       0.0020            nan     0.1000    0.0001
##    80       0.0003            nan     0.1000    0.0000
##   100       0.0000            nan     0.1000    0.0000
##   120       0.0000            nan     0.1000    0.0000
##   140       0.0000            nan     0.1000    0.0000
##   150       0.0000            nan     0.1000    0.0000
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0925
##     2        0.9023             nan     0.1000    0.0729
##     3        0.7848             nan     0.1000    0.0599
##     4        0.6876             nan     0.1000    0.0496
##     5        0.6056             nan     0.1000    0.0432
##     6        0.5355             nan     0.1000    0.0338
##     7        0.4751             nan     0.1000    0.0322
##     8        0.4227             nan     0.1000    0.0254
##     9        0.3768             nan     0.1000    0.0222
##    10        0.3366             nan     0.1000    0.0204
##    20        0.1154             nan     0.1000    0.0064
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0949
##     2        0.9023             nan     0.1000    0.0762
##     3        0.7848             nan     0.1000    0.0491
##     4        0.6876             nan     0.1000    0.0514
##     5        0.6056             nan     0.1000    0.0395
##     6        0.5355             nan     0.1000    0.0350
##     7        0.4808             nan     0.1000    0.0236
##     8        0.4276             nan     0.1000    0.0272
##     9        0.3810             nan     0.1000    0.0236
##    10        0.3404             nan     0.1000    0.0205
##    20        0.1164             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0482             nan     0.1000    0.0973
##     2        0.9023             nan     0.1000    0.0795
```

```
##     3        0.7848             nan     0.1000    0.0575
##     4        0.6876             nan     0.1000    0.0486
##     5        0.6056             nan     0.1000    0.0469
##     6        0.5355             nan     0.1000    0.0338
##     7        0.4751             nan     0.1000    0.0312
##     8        0.4227             nan     0.1000    0.0258
##     9        0.3768             nan     0.1000    0.0222
##    10        0.3366             nan     0.1000    0.0186
##    20        0.1154             nan     0.1000    0.0062
##    40        0.0151             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0899
##     2        0.9056             nan     0.1000    0.0647
##     3        0.7878             nan     0.1000    0.0611
##     4        0.6902             nan     0.1000    0.0486
##     5        0.6079             nan     0.1000    0.0381
##     6        0.5376             nan     0.1000    0.0368
##     7        0.4770             nan     0.1000    0.0302
##     8        0.4244             nan     0.1000    0.0271
##     9        0.3783             nan     0.1000    0.0230
##    10        0.3379             nan     0.1000    0.0210
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1015
##     2        0.9056             nan     0.1000    0.0744
##     3        0.7878             nan     0.1000    0.0623
##     4        0.6902             nan     0.1000    0.0523
##     5        0.6079             nan     0.1000    0.0432
```

```
##     6        0.5376           nan     0.1000    0.0356
##     7        0.4770           nan     0.1000    0.0292
##     8        0.4244           nan     0.1000    0.0263
##     9        0.3783           nan     0.1000    0.0240
##    10        0.3379           nan     0.1000    0.0205
##    20        0.1158           nan     0.1000    0.0064
##    40        0.0152           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517           nan     0.1000    0.0992
##     2        0.9056           nan     0.1000    0.0663
##     3        0.7878           nan     0.1000    0.0563
##     4        0.6902           nan     0.1000    0.0486
##     5        0.6079           nan     0.1000    0.0432
##     6        0.5376           nan     0.1000    0.0351
##     7        0.4770           nan     0.1000    0.0273
##     8        0.4244           nan     0.1000    0.0271
##     9        0.3783           nan     0.1000    0.0237
##    10        0.3379           nan     0.1000    0.0202
##    20        0.1158           nan     0.1000    0.0062
##    40        0.0152           nan     0.1000    0.0008
##    60        0.0020           nan     0.1000    0.0001
##    80        0.0003           nan     0.1000    0.0000
##   100        0.0000           nan     0.1000    0.0000
##   120        0.0000           nan     0.1000    0.0000
##   140        0.0000           nan     0.1000    0.0000
##   150        0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517           nan     0.1000    0.0945
##     2        0.9056           nan     0.1000    0.0712
##     3        0.7878           nan     0.1000    0.0587
##     4        0.6902           nan     0.1000    0.0505
##     5        0.6079           nan     0.1000    0.0388
##     6        0.5376           nan     0.1000    0.0374
##     7        0.4770           nan     0.1000    0.0297
##     8        0.4244           nan     0.1000    0.0250
```

```
##     9       0.3783             nan     0.1000    0.0230
##    10       0.3379             nan     0.1000    0.0190
##    20       0.1158             nan     0.1000    0.0061
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.0969
##     2       0.9056             nan     0.1000    0.0728
##     3       0.7878             nan     0.1000    0.0527
##     4       0.6902             nan     0.1000    0.0496
##     5       0.6079             nan     0.1000    0.0410
##     6       0.5376             nan     0.1000    0.0339
##     7       0.4770             nan     0.1000    0.0292
##     8       0.4244             nan     0.1000    0.0254
##     9       0.3783             nan     0.1000    0.0226
##    10       0.3379             nan     0.1000    0.0205
##    20       0.1158             nan     0.1000    0.0063
##    40       0.0152             nan     0.1000    0.0008
##    60       0.0020             nan     0.1000    0.0001
##    80       0.0003             nan     0.1000    0.0000
##   100       0.0000             nan     0.1000    0.0000
##   120       0.0000             nan     0.1000    0.0000
##   140       0.0000             nan     0.1000    0.0000
##   150       0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517             nan     0.1000    0.0945
##     2       0.9056             nan     0.1000    0.0728
##     3       0.7878             nan     0.1000    0.0623
##     4       0.6902             nan     0.1000    0.0450
##     5       0.6079             nan     0.1000    0.0447
##     6       0.5376             nan     0.1000    0.0345
##     7       0.4770             nan     0.1000    0.0292
##     8       0.4244             nan     0.1000    0.0275
##     9       0.3783             nan     0.1000    0.0230
##    10       0.3379             nan     0.1000    0.0208
##    20       0.1158             nan     0.1000    0.0063
```

```
##    40        0.0152             nan     0.1000    0.0009
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.0829
##    2        0.9056             nan     0.1000    0.0696
##    3        0.7878             nan     0.1000    0.0527
##    4        0.6902             nan     0.1000    0.0496
##    5        0.6079             nan     0.1000    0.0396
##    6        0.5376             nan     0.1000    0.0315
##    7        0.4770             nan     0.1000    0.0322
##    8        0.4244             nan     0.1000    0.0263
##    9        0.3783             nan     0.1000    0.0223
##   10        0.3379             nan     0.1000    0.0193
##   20        0.1158             nan     0.1000    0.0062
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
##  100        0.0000             nan     0.1000    0.0000
##  120        0.0000             nan     0.1000    0.0000
##  140        0.0000             nan     0.1000    0.0000
##  150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##    1        1.0517             nan     0.1000    0.1015
##    2        0.9056             nan     0.1000    0.0631
##    3        0.7878             nan     0.1000    0.0611
##    4        0.6902             nan     0.1000    0.0440
##    5        0.6079             nan     0.1000    0.0410
##    6        0.5376             nan     0.1000    0.0356
##    7        0.4770             nan     0.1000    0.0307
##    8        0.4244             nan     0.1000    0.0283
##    9        0.3783             nan     0.1000    0.0237
##   10        0.3379             nan     0.1000    0.0205
##   20        0.1158             nan     0.1000    0.0061
##   40        0.0152             nan     0.1000    0.0008
##   60        0.0020             nan     0.1000    0.0001
##   80        0.0003             nan     0.1000    0.0000
```

```
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.1039
##     2        0.9056             nan     0.1000    0.0760
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0459
##     5        0.6079             nan     0.1000    0.0388
##     6        0.5376             nan     0.1000    0.0333
##     7        0.4770             nan     0.1000    0.0283
##     8        0.4244             nan     0.1000    0.0279
##     9        0.3783             nan     0.1000    0.0216
##    10        0.3379             nan     0.1000    0.0213
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0875
##     2        0.9056             nan     0.1000    0.0728
##     3        0.7878             nan     0.1000    0.0587
##     4        0.6902             nan     0.1000    0.0496
##     5        0.6079             nan     0.1000    0.0418
##     6        0.5376             nan     0.1000    0.0345
##     7        0.4770             nan     0.1000    0.0297
##     8        0.4244             nan     0.1000    0.0263
##     9        0.3783             nan     0.1000    0.0237
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
```

```
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517           nan     0.1000    0.1086
##     2       0.9056           nan     0.1000    0.0712
##     3       0.7878           nan     0.1000    0.0563
##     4       0.6902           nan     0.1000    0.0523
##     5       0.6079           nan     0.1000    0.0403
##     6       0.5376           nan     0.1000    0.0345
##     7       0.4770           nan     0.1000    0.0307
##     8       0.4244           nan     0.1000    0.0267
##     9       0.3783           nan     0.1000    0.0223
##    10       0.3379           nan     0.1000    0.0196
##    20       0.1158           nan     0.1000    0.0064
##    40       0.0152           nan     0.1000    0.0008
##    60       0.0020           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1       1.0517           nan     0.1000    0.0899
##     2       0.9056           nan     0.1000    0.0696
##     3       0.7878           nan     0.1000    0.0551
##     4       0.6902           nan     0.1000    0.0514
##     5       0.6079           nan     0.1000    0.0403
##     6       0.5376           nan     0.1000    0.0356
##     7       0.4770           nan     0.1000    0.0268
##     8       0.4244           nan     0.1000    0.0271
##     9       0.3783           nan     0.1000    0.0237
##    10       0.3379           nan     0.1000    0.0208
##    20       0.1158           nan     0.1000    0.0063
##    40       0.0153           nan     0.1000    0.0008
##    60       0.0021           nan     0.1000    0.0001
##    80       0.0003           nan     0.1000    0.0000
##   100       0.0000           nan     0.1000    0.0000
##   120       0.0000           nan     0.1000    0.0000
##   140       0.0000           nan     0.1000    0.0000
##   150       0.0000           nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##     1        1.0592             nan     0.1000    0.0983
##     2        0.9126             nan     0.1000    0.0772
##     3        0.7941             nan     0.1000    0.0633
##     4        0.6959             nan     0.1000    0.0487
##     5        0.6130             nan     0.1000    0.0432
##     6        0.5422             nan     0.1000    0.0340
##     7        0.4811             nan     0.1000    0.0317
##     8        0.4280             nan     0.1000    0.0263
##     9        0.3816             nan     0.1000    0.0207
##    10        0.3408             nan     0.1000    0.0208
##    20        0.1168             nan     0.1000    0.0064
##    40        0.0153             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0592             nan     0.1000    0.0915
##     2        0.9126             nan     0.1000    0.0678
##     3        0.7941             nan     0.1000    0.0598
##     4        0.6959             nan     0.1000    0.0513
##     5        0.6130             nan     0.1000    0.0404
##     6        0.5422             nan     0.1000    0.0374
##     7        0.4811             nan     0.1000    0.0327
##     8        0.4280             nan     0.1000    0.0275
##     9        0.3816             nan     0.1000    0.0237
##    10        0.3408             nan     0.1000    0.0202
##    20        0.1168             nan     0.1000    0.0062
##    40        0.0153             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0592             nan     0.1000    0.0893
##     2        0.9126             nan     0.1000    0.0788
##     3        0.7941             nan     0.1000    0.0633
```

```
##      4        0.6959             nan     0.1000    0.0558
##      5        0.6130             nan     0.1000    0.0382
##      6        0.5422             nan     0.1000    0.0363
##      7        0.4811             nan     0.1000    0.0322
##      8        0.4280             nan     0.1000    0.0263
##      9        0.3816             nan     0.1000    0.0224
##     10        0.3408             nan     0.1000    0.0208
##     20        0.1168             nan     0.1000    0.0064
##     40        0.0153             nan     0.1000    0.0008
##     60        0.0021             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.1015
##      2        0.9056             nan     0.1000    0.0696
##      3        0.7878             nan     0.1000    0.0611
##      4        0.6902             nan     0.1000    0.0450
##      5        0.6079             nan     0.1000    0.0403
##      6        0.5376             nan     0.1000    0.0339
##      7        0.4770             nan     0.1000    0.0302
##      8        0.4244             nan     0.1000    0.0254
##      9        0.3783             nan     0.1000    0.0216
##     10        0.3379             nan     0.1000    0.0196
##     20        0.1158             nan     0.1000    0.0067
##     40        0.0152             nan     0.1000    0.0008
##     60        0.0020             nan     0.1000    0.0001
##     80        0.0003             nan     0.1000    0.0000
##    100        0.0000             nan     0.1000    0.0000
##    120        0.0000             nan     0.1000    0.0000
##    140        0.0000             nan     0.1000    0.0000
##    150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.0517             nan     0.1000    0.0945
##      2        0.9056             nan     0.1000    0.0712
##      3        0.7878             nan     0.1000    0.0623
##      4        0.6902             nan     0.1000    0.0505
##      5        0.6079             nan     0.1000    0.0425
##      6        0.5376             nan     0.1000    0.0351
```

```
##     7        0.4770             nan     0.1000    0.0288
##     8        0.4244             nan     0.1000    0.0259
##     9        0.3783             nan     0.1000    0.0219
##    10        0.3379             nan     0.1000    0.0202
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0021             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0969
##     2        0.9056             nan     0.1000    0.0728
##     3        0.7878             nan     0.1000    0.0635
##     4        0.6902             nan     0.1000    0.0496
##     5        0.6079             nan     0.1000    0.0410
##     6        0.5376             nan     0.1000    0.0315
##     7        0.4770             nan     0.1000    0.0273
##     8        0.4244             nan     0.1000    0.0246
##     9        0.3783             nan     0.1000    0.0223
##    10        0.3379             nan     0.1000    0.0196
##    20        0.1158             nan     0.1000    0.0063
##    40        0.0152             nan     0.1000    0.0008
##    60        0.0020             nan     0.1000    0.0001
##    80        0.0003             nan     0.1000    0.0000
##   100        0.0000             nan     0.1000    0.0000
##   120        0.0000             nan     0.1000    0.0000
##   140        0.0000             nan     0.1000    0.0000
##   150        0.0000             nan     0.1000    0.0000
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##     1        1.0517             nan     0.1000    0.0872
##     2        0.9056             nan     0.1000    0.0721
##     3        0.7878             nan     0.1000    0.0593
##     4        0.6903             nan     0.1000    0.0491
##     5        0.6080             nan     0.1000    0.0394
##     6        0.5377             nan     0.1000    0.0359
##     7        0.4771             nan     0.1000    0.0322
##     8        0.4244             nan     0.1000    0.0257
##     9        0.3784             nan     0.1000    0.0225
```

```
##    10      0.3380           nan     0.1000   0.0208
##    20      0.1158           nan     0.1000   0.0062
##    40      0.0152           nan     0.1000   0.0008
##    50      0.0056           nan     0.1000   0.0003
```

#3.3.2. Resultados del Modelo

```
modelo_boost_gbm
```

```
## Stochastic Gradient Boosting
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 142, 141, 141, 142, 141, 141, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa
##   1                   50      1         1
##   1                  100      1         1
##   1                  150      1         1
##   2                   50      1         1
##   2                  100      1         1
##   2                  150      1         1
##   3                   50      1         1
##   3                  100      1         1
##   3                  150      1         1
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  1, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
as.data.frame(modelo_boost_gbm$results[,])
```

```
##   shrinkage interaction.depth n.minobsinnode n.trees Accuracy Kappa
AccuracySD
## 1       0.1                 1             10      50        1     1          0
## 4       0.1                 2             10      50        1     1          0
## 7       0.1                 3             10      50        1     1          0
```

```
## 2     0.1             1          10    100      1    1       0
## 5     0.1             2          10    100      1    1       0
## 8     0.1             3          10    100      1    1       0
## 3     0.1             1          10    150      1    1       0
## 6     0.1             2          10    150      1    1       0
## 9     0.1             3          10    150      1    1       0
##   KappaSD
## 1       0
## 4       0
## 7       0
## 2       0
## 5       0
## 8       0
## 3       0
## 6       0
## 9       0
```

```
#3.3.3. Predicción
prediccion_boots_gbm = predict(modelo_boost_gbm,COUNTRIES)
```

```
#3.3.4. Proporción de Clasificación Correcta
(correcto_boost_gbm<-modelo_boost_gbm$results[8,5])
```

```
## [1] 1
```

```
#3.3.5. Proporción de Clasificación Incorrecta
(incorrecto_boost_gbm<-1-correcto_boost_gbm)
```

```
## [1] 0
```

#3.4. STACKING

```
#3.4.1. Modelo
library(caretEnsemble)
```

```
##
## Attaching package: 'caretEnsemble'
```

```
## The following object is masked from 'package:ggplot2':
##
##     autoplot
```

```
control_stacking <- trainControl(method="repeatedcv", number=5, repeats=3,
savePredictions=TRUE, classProbs=TRUE)
algoritmos <- c('rpart','knn', 'svmRadial','rf','treebag',"glm","adaboost")
```

```
set.seed(1)
modelo_stacking <- caretList(MoodysRat~., data = COUNTRIES,
trControl=control_stacking, methodList=algoritmos)
```

```
## Warning in trControlCheck(x = trControl, y = target): x$savePredictions ==
TRUE
## is depreciated. Setting to 'final' instead.

## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.

## note: only 2 unique complexity parameters in default grid. Truncating the grid
to 2 .

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

modelo_stacking

## $rpart
## CART
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
## Resampling results across tuning parameters:
##
##   cp   Accuracy  Kappa
##   0.0  1         1
##   0.5  1         1
##   1.0  1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 1.
##
```

```
## $knn
## k-Nearest Neighbors
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy  Kappa
##   5  1         1
##   7  1         1
##   9  1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
##
## $svmRadial
## Support Vector Machines with Radial Basis Function Kernel
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy  Kappa
##   0.25  1         1
##   0.50  1         1
##   1.00  1         1
##
## Tuning parameter 'sigma' was held constant at a value of 2.32781
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 2.32781 and C = 0.25.
##
## $rf
## Random Forest
```

```
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##   2     1         1
##   3     1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
##
## $treebag
## Bagged CART
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
##
##
## $glm
## Generalized Linear Model
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
```

```
## Resampling results:
##
##   Accuracy  Kappa
##   1         1
##
##
## $adaboost
## AdaBoost Classification Trees
##
## 157 samples
##   3 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 125, 126, 125, 126, 126, 126, ...
## Resampling results across tuning parameters:
##
##   nIter  method        Accuracy  Kappa
##    50    Adaboost.M1   1         1
##    50    Real adaboost 1         1
##   100    Adaboost.M1   1         1
##   100    Real adaboost 1         1
##   150    Adaboost.M1   1         1
##   150    Real adaboost 1         1
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nIter = 50 and method =
Adaboost.M1.
##
## attr(,"class")
## [1] "caretList"
```

```
#3.4.2. Resultados del Modelo
resultados_staking <- resamples(modelo_stacking)
summary(resultados_staking )
```

```
##
## Call:
## summary.resamples(object = resultados_staking)
##
## Models: rpart, knn, svmRadial, rf, treebag, glm, adaboost
## Number of resamples: 15
##
```

```
## Accuracy
##          Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## rpart       1      1     1    1      1    1    0
## knn         1      1     1    1      1    1    0
## svmRadial   1      1     1    1      1    1    0
## rf          1      1     1    1      1    1    0
## treebag     1      1     1    1      1    1    0
## glm         1      1     1    1      1    1    0
## adaboost    1      1     1    1      1    1    0
##
## Kappa
##          Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## rpart       1      1     1    1      1    1    0
## knn         1      1     1    1      1    1    0
## svmRadial   1      1     1    1      1    1    0
## rf          1      1     1    1      1    1    0
## treebag     1      1     1    1      1    1    0
## glm         1      1     1    1      1    1    0
## adaboost    1      1     1    1      1    1    0
```

#3.4.3. Stacking usando Modelos Lineales Generalizados

```
glm_stack <- caretStack(modelo_stacking, method="glm", metric="Accuracy",
trControl=control_stacking)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

print(glm_stack)

## A glm ensemble of 7 base models: rpart, knn, svmRadial, rf, treebag, glm,
adaboost
##
## Ensemble results:
## Generalized Linear Model
##
## 471 samples
##   7 predictor
##   2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 376, 376, 378, 377, 377, 377, ...
## Resampling results:
##
##   Accuracy  Kappa
##   1         1

#3.4.4. Predicción Stacking usando Modelos Lineales Generalizados
stack_predicteds <- predict(glm_stack)

## Warning in predict.caretList(object$models, newdata = newdata, na.action =
## na.action): Predicting without new data is not well supported. Attempting to
## predict on the training data.

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

#3.4.5. Proporción de Clasificación Correcta
(correcto_stacking<-glm_stack$error[2])

##   Accuracy
## 1        1
```

#3.4.6. Proporción de Clasificación Incorrecta
(incorrecto_stacking<-1-correcto_stacking)

Accuracy
1 0

#3.5. CUADRO RESUMEN

```{r}

#3.5.1. Proporción de Clasificación Correcta Según Modelo

library(DT)

(correcto_bag_Cart<-modelo_bag_CART$results[2])

(correcto_bosques<-modelo_bosques_2$results[2,2])

(correcto_boost_gbm<-modelo_boost_gbm$results[8,5])

(correcto_stacking<-glm_stack$error[2])

correctos_general <- c(correcto_bag_Cart, correcto_bosques, correcto_boost_gbm, correcto_stacking)


MODELOS <- c("Bagged Trees", "Bosques", "Boostring GBM", "Stacking GLM")


cuadro_correctos <- data.frame(MODELOS, correctos_general)


cuadro_correctos


#3.5.2. Proporción de Clasificación Incorrecta Según Modelo

(incorrecto_bag_Cart<-1-correcto_bag_Cart)

(incorrecto_bosques<-1-correcto_bosques)

(incorrecto_boost_gbm<-1-correcto_boost_gbm)

(incorrecto_stacking<-1-correcto_stacking)
```

```
incorrectos_general <- c(incorrecto_bag_Cart, incorrecto_bosques, incorrecto_boost_gbm, incorrecto_stacking)

nombres_modelos <- c("Bagged Trees", "Bosques", "Boostring GBM", "Stacking GLM")

MODELOS <- c("Bagged Trees", "Bosques", "Boostring GBM", "Stacking GLM")

cuadro_incorrectos <- data.frame(MODELOS, incorrectos_general)

cuadro_incorrectos
```

```
##          MODELOS Accuracy
## 1  Bagged Trees        1
## 2        Bosques        1
## 3 Boostring GBM        1
## 4  Stacking GLM        1
```

V. CONCLUSIONES Y RECOMENDACIONES

Sobre los modelos individuales evaluados, con excepción del modelo de redes neuronales artificiales (ANN) y del modelo de análisis discriminante lineal (LDA), las diferencias en las tasas de error son nulas; en los dos primeros mencionados, ANN tiene un alto error (llegando al 0.5 en validación), mientras que LDA tiene un error bajo (en validación es aproximadamente del 0.0425). Al estudiar el desempeño del modelo LDA se obtuvieron notificaciones por parte de RStudio de

colinealidad entre las variables explicativas, lo cual explica la menor calidad en desempeño del modelo LDA en relación con sus pares dentro de los modelos que obtuvieron errores de clasificación del 5% o inferiores (todos, salvo ANN), puesto que LDA involucra directamente técnicas de regresión lineal, las cuales se ven afectadas porque las variables explicativas sean combinación lineal unas de otras. Esta dependencia lineal entre variables es muy probable que exista (o al menos con más intensidad) entre la prima riesgo de las acciones y la prima riesgo de los bonos como resultado no sólo de su metodología de cálculo (ambas son primas de riesgo) sino también porque sus determinantes teóricos fundamentales son similares, al igual que el de cualesquiera pares de variables económicas (incluidas por consiguiente también las demás variables explicativas aquí empleadas), salvo que se analicen desde una escuela de pensamiento económico diferente o distinta; sin embargo, este fenómeno de estrecha interrelación entre variables explicativas no es exclusivo de los sistemas económicos. Precisamente por ello el problema de multicolinealidad es sumamente frecuente en el análisis de fenómenos económicos y la afectación consiste en que, a pesar de que los estimadores obtenidos siguen siendo los mejores estimadores lineales insesgados (MELI), la multicolinealidad implica alta varianza de tales estimadores y dificulta la estimación precisa (Gujarati & Porter, 2010, pág. 327). Esto ha dado pie en la econometría (Gujarati & Porter, 2010, pág. 320) al surgimiento de una amplia gama de procedimientos para lidiar con la multicolinealidad, y lo mismo ocurre en ecología y otras ramas (Dormann, y otros, 2013, pág. 27). Sin embargo, a veces el remedio puede ser peor que la enfermedad (Gujarati & Porter, 2010, pág. 320) y el investigador debe realizar consideraciones adicionales, tales como lo establecido por la teoría científica bajo la cual se describe y explica el fenómeno estudiado, el desempeño empírico de los modelos ante violaciones de los supuestos a través de simulaciones y los resultados de investigaciones previas similares a la que se está llevando a cabo, antes de descartar la validez de los resultados en presencia de multicolinealidad o, por el contrario, de no realizar consideraciones adicionales

ante la presencia de tal problema. En (Gujarati & Porter, 2010, págs. 320-342) se hace un análisis relativamente exhaustivo sobre la multicolinealidad, las técnicas estadísticas para solucionarlo y la opinión de expertos en el campo de la econometría sobre las consideraciones que sobre este problema deben tenerse.

Para el caso de las ANN, señalan (De Veaux & Ungar, 1994, pág. 2) que este tipo de modelos generalmente no sufren de multicolinealidad porque tienden a estar sobreparametrizadas. Los pesos adicionales aprendidos crean redundancias que hacen que las cosas que afectan a cualquier pequeño subconjunto de características (como la multicolinealidad) no sean importantes. Es importante resaltar que las ANN son un algoritmo no lineal.

Para el caso de la regresión logística multinomial, cuya estimación empírica implica generalmente el uso de modelos lineales generalizados (por la reducción de costo computacional que implica la linealización de la respuesta), la multicolinealidad afecta la precisión de la estimación de la misma manera en que lo hace en el modelo de regresión lineal clásico, puesto que un requisito de los GLM es que las observaciones sean independientes entre sí (McCullagh & Nelder, 1989, pág. 5).

De los dos párrafos anteriores se deduce que, para el caso de las máquinas de vectores de soporte, la ausencia de multicolinealidad sólo es estadísticamente importante si el kernel[20] empleado es lineal, precisamente como el implementado en esta investigación.

Los árboles de decisión y los algoritmos de árboles potenciados son inmunes a la multicolinealidad por naturaleza. Esto ocurre puesto que, cuando decide la división, el árbol elegirá solo una de las características perfectamente correlacionadas (Badr, 2018). Por su parte, la generalización de los árboles de decisión en el contexto de los métodos de ensamble, es decir, los bosques

[20] Los kernel o las funciones del kernel son métodos que utilizan los clasificadores lineales, como SVM, para clasificar puntos de datos separables de forma no lineal. Existen kernel no lineales.

aleatorios, utilizan intrínsecamente remuestreo bootstrapping (genera muestras isométricas -que las observaciones que las componen guardan la misma distancia entre sí con relación a la original- a la original permutando -muestreo aleatorio con reemplazo- las filas) y el remuestreo de características (genera muestras isométricas a la original permutando las columnas). Por ello, los bosques aleatorios no se ven importantemente afectados por la multicolinealidad, ya que con los métodos de remuestreo antes mencionados selecciona diferentes conjuntos de características para diferentes modelos y, por supuesto, cada modelo ve un conjunto diferente de muestras, por lo que la relevancia estadística de la apriorística multicolinealidad entre variables explicativas se desvanece. Es importante recordar que los bosques aleatorios son un algoritmo no-lineal.

Por su parte, se señala en (Cross Validated, 2018), el modelo de K-Vecinos Más Cercanos, así como todo modelo de análisis por agrupamiento de datos, no se ve afectado negativamente por la heterocedasticidad (que en los modelos de regresión es quizás el problema más grave que puede presentarse), los resultados sí se ven afectados negativamente por la multicolinealidad de las características/variables utilizadas en el agrupamiento, ya que la característica/variable correlacionada tendrá un peso adicional en el cálculo de la distancia con relación al peso deseado (al que debería tener considerando únicamente la métrica implementada como medida de cercanía entre las observaciones). Así, K-Vecinos Más Cercanos, a pesar de no ser un algoritmo lineal (porque es un modelo no-paramétrico y, por consiguiente, no hace ningún supuesto sobre el modelo subyacente al proceso de generación de los datos), sí es afectado (aunque por otros motivos) por la multicolinealidad.

Así, si se desea eliminar la multicolinealidad, es posible utilizar el Análisis de Componentes Principales (PCA) para proyectar los datos en un nuevo espacio donde las 'nuevas características' serán ortogonales entre sí (por el teorema de Johnson-Linderstrauss, que exige la linealidad de la vinculación entre variables a lo

largo del espacio original). Hecho lo anterior, se puede entrenar el modelo con las nuevas características, pero en modelos no-lineales el rendimiento será el mismo, puesto que en ellos el empleo de PCA es simplemente rotar los límites de decisión del espacio con relación a los límites del espacio original. La relevancia de la utilización de las variables obtenidas por PCA se debe a que, como señala (Lawrence, 2022), los modelos de variables latentes son una perspectiva probabilística del aprendizaje no supervisado que conduce a algoritmos de reducción de dimensionalidad. Complementariamente a lo anterior, señalan (Dormann, y otros, 2013, pág. 36), los modelos de variables latentes de tipo reducción de dimensionalidad se encuentran entre los más poderosos para tratar el problema de multicolinealidad.

Establecido lo anterior, la validez inferencial de los modelos K-Vecinos Más Cercanos, SVM con kernel lineal, LDA y regresión logística multinomial debe tomarse de manera conservadora (con altas reservas) a falta de realizar más pruebas al respecto. Puesto que son inmunes a tal problema los modelos de ANN y los bosques aleatorios, siendo los resultados predictivos de las ANN muy pobres, debe considerarse como el mejor modelo individual al modelo de bosques aleatorios.

Sin embargo, puesto que los resultados de los modelos individuales aquí evaluados son a su vez resultados obtenidos a través de la implementación de métodos de validación cruzada (que usa remuestreo), el problema de multicolinealidad se diluye al igual que como ocurre intrínsecamente (sin la intervención de validación cruzada) con los bosques aleatorios. Así, habiendo empleado los métodos de validación cruzada, puede confiarse estadísticamente en los aspectos inferenciales que de estos modelos se pueden derivar.

Los resultados de los ensambles descartan que los resultados obtenidos individualmente se deban a errores de programación en las funciones manuales, puesto que los ensambles son funciones automatizadas de librerías de R y se

obtienen resultados similares (con la diferencia que en todos los ensambles sin excepción la precisión de la predicción es perfecta). En los resultados de ensambles aquí obtenidos, dado que también fueron aplicados en ellos los mismos métodos de validación cruzada antes descritos, también puede confiarse estadísticamente en términos inferenciales.

Sin embargo, la última palabra con relación a si la multicolinealidad es verdaderamente un problema o no, siempre que los factores de inflación de varianza sean menores a 10 (puesto que esto implica intervalos de confianza de los parámetros demasiado amplios y, con ello, la imposibilidad de predicción precisa), no la tienen los métodos de validación cruzada, sino si desde el marco teórico de la ciencia aplicada que se estudia el fenómeno su relación lineal es lógicamente esperable o no. Esto es así porque la ciencia se hace, en última instancia, desde la cualidad, no desde la cantidad, es decir, no desde la instrumentalización de las teorías o de los resultados arrojados por tales instrumentos, sino desde las relaciones cualitativas para cuyo modelaje tales instrumentos han sido creados, aunque evidentemente existe una retroalimentación de carácter dinámico y complejo entre la cualidad y la cantidad. Como consecuencia de lo anterior, más allá de la robustez predictiva presentada por los árboles de decisión, los modelos de ensambles y los modelos individuales a los cuales se les aplicó métodos de validación cruzada, uso de una teoría pseudocientífica que ocurre en las finanzas (*i.e.*, la teoría económica neoclásica) invalida gnoseológicamente tal robustez, que es puramente estadística, no filosófica ni científica[21].

VI. REFERENCIAS

Amat Rodrigo, J. (Septiembre de 2016). *Análisis discriminante lineal (LDA) y análisis discriminante cuadrático (QDA)*. Obtenido de cienciadedatos.net:

[21] A pesar de que la teoría estadística y su instrumental es rigurosamente científica, la validez científica última del fenómeno estudiado (sea este de índole social o natural) radica precisamente en la teoría de las ciencias aplicadas bajo la cual se estudia. Es precisamente a causa de lo anterior que el concepto de correlación espuria es, en última instancia, de carácter científico-aplicado y no de carácter formal (Kliman, 2014, pág. 346). La razón de ello es porque "Los fundamentos de la ciencia se encuentran en la historia y la sociología, no en el análisis formal." (Levins, 1993, pág. 547).

https://www.cienciadedatos.net/documentos/28_linear_discriminant_analysis_lda_y_quadratic_discriminant_analysis_qda

Badr, W. (18 de Enero de 2018). *Why Feature Correlation Matters…. A Lot!* Obtenido de Towards Data Science: https://towardsdatascience.com/why-feature-correlation-matters-a-lot-847e8ba439c4Why Feature Correlation Matters …. A Lot!

Bastounis, A., & Hansen, A. C. (2017). On the Absence of Uniform Recovery in Many Real-World Applications of Compressed Sensing and the Restricted Isometry Property and Nullspace Property in Levels. *Society for Industrial and Applied Mathematics Journal of Imaging Sciences, 10*(1), 335-371. Obtenido de https://epubs.siam.org/doi/abs/10.1137/15M1043972

Bojanov, B. (1992). Optimal Recovery of Functions and Integrals. En A. Joseph, F. Mignot, F. Murat, B. Prum, & R. Rentschler, *First European Congress of Mathematics. Paris, July 6-10, 1992. Volume I. Invited Lectures (Part 1)* (págs. 371-390). Basel: Birkhauser Verlag.

Christopher, A. (2 de Febrero de 2021). *K-Nearest Neighbor*. Obtenido de Medium: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

Cross Validated. (04 de Agosto de 2013). *Flexible and inflexible models in machine learning*. Obtenido de Questions: https://stats.stackexchange.com/questions/69237/flexible-and-inflexible-models-in-machine-learning

Cross Validated. (31 de Julio de 2018). *Will collinearity be a problem for K-nearest neigbor?* Obtenido de StackExchange: https://stats.stackexchange.com/questions/359905/will-collinearity-be-a-problem-for-k-nearest-neigbor

Dale, S. (8 de Noviembre de 2020). *An intuitive explanation of the decision tree algorithm*. Obtenido de Towards Data Science: https://towardsdatascience.com/building-an-intuition-for-the-decision-tree-algorithm-75e0786e86d

Damodaran, A. (5 de Enero de 2022). *Country Default Spreads and Risk Premiums*. Obtenido de New York University: https://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/ctryprem.html

De Veaux, R. D., & Ungar, L. H. (1994). Multicollinearity: A tale of two nonparametric regressions. *Selecting Models from Data*, 1-10. Obtenido de

https://www.cis.upenn.edu/%7Eungar/Datamining/Publications/tale2.pdf

Delianedis, G., & Geske, R. (2001). The Components of Corporate Credit Spreads: Default, Recovery, Tax, Jumps, Liquidity, and Market Factors. *Finance*, 1-39. Obtenido de https://www.anderson.ucla.edu/documents/areas/adm/media/geske.pdf

Doerra, B., & Mayerb, S. (2020). The recovery of ridge functions on the hypercube suffers from the curse of dimensionality. *arvix*, 1-37.

Dormann, C. F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., . . . Reineking, B. (2013). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*, 27-46. Obtenido de https://onlinelibrary.wiley.com/doi/10.1111/j.1600-0587.2012.07348.x/epdfv

Ganegedara, T. (29 de Noviembre de 2018). *Intuitive Guide to Understanding Decision Trees*. Obtenido de Towards Data Science: https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-decision-trees-adb2165ccab7

Gujarati, D., & Porter, D. (2010). *Econometría.* México, D.F.: Fondo de Cultura Económica.

Hastie, T., Tibshirani, R., & Friedman, J. (2016). *The Elements of Statistical Learning. Data Mining, Inference, and Prediction* (Segunda ed.). New York: Springer.

Hayes, A. (13 de Diciembre de 2021). *Risk Premium*. Obtenido de Investopedia: https://www.investopedia.com/terms/r/riskpremium.asp

IBM. (1 de Julio de 2022). *K-Nearest Neighbors Algorithm*. Obtenido de Topics: https://www.ibm.com/topics/knn

Johnson, R. (2 de Julio de 2022). *Data Science: Back to Basics – It's about being flexible*. Obtenido de spr: https://spr.com/data-science-back-basics-flexible/

Kliman, A. (2014). What is spurious correlation? A reply to Díaz and Osuna. *Journal of Post Keynesian Economics, 21*(2), 345-356.

Lantz, B. (2013). *Machine Learning with R.* Birmingham: Packt Publishing Ltd. Obtenido de https://edu.kpfu.ru/pluginfile.php/278552/mod_resource/content/1/MachineLearningR__Brett_Lantz.pdf

Lawrence, N. D. (14 de Julio de 2022). *Dimensionality Reduction: Latent Variable Modelling*. Obtenido de Cambridge University: https://mlatcl.github.io/mlai/lectures/08-dimensionality-reduction.html

Levins, R. (Diciembre de 1993). A Response to Orzack and Sober: Formal Analysis and the Fluidity of Science. *The Quarterly Review of Biology, 68*(4), 547-55. Obtenido de https://www.jstor.org/stable/3037251

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models* (Segunda ed.). London: Chapman & Hall.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to Linear Regression Analysis* (Quinta ed.). New Jersey: John Wiley & Sons, Inc.

Moody's. (3 de Junio de 2022). *Rating Scale and Definitions.* Obtenido de Products: https://www.moodys.com/sites/products/productattachments/ap075378_1_1408_ki.pdf

Nabi, I. (3 de Abril de 2021). *Sobre el Análisis de Componentes Principales (PCA).* Obtenido de Marxist Statistics: https://marxianstatistics.files.wordpress.com/2021/04/sobre-el-analisis-de-componentes-principales-isadore-nabi-4.pdf

stack overflow. (18 de Octubre de 2014). *what is the definition of "flexibility" of a method in Machine Learning?* Obtenido de Questions: https://stackoverflow.com/questions/26437372/what-is-the-definition-of-flexibility-of-a-method-in-machine-learning#:~:text=Flexibility%20describes%20the%20ability%20to,fit%22%20to%20the%20training%20data.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory* (Segunda ed.). New York: Springer.

Zaitseva, T., Malykhin, Y., & Ryutin, K. (2021). Recovery of regular ridge functions on the ball. *arvix*, 1-27. Obtenido de https://arxiv-export-lb.library.cornell.edu/pdf/2102.13203

Zelterman, D. (2015). *Applied Multivariate Statistics with R.* Gewerbestrass, Switzerland: Springer International Publishing Switzerland.

Zhang, P., Gan, L., Ling, C., & Sun, S. (2017). Uniform Recovery Bounds for Structured Random Matrices in Corrupted Compressed Sensing. *arvix*, 1-12. Obtenido de https://arxiv.org/abs/1706.09087v3