Trabajo Fin de Máster

Análisis de datos: Estudio de la probabilidad de marcha de un empleado

Rubén Moirón Sánchez

Curso 2020/2021

UNIVERSIDAD DE GRANADA

MÁSTER DE ESTADÍSTICA APLICADA

Trabajo Fin de Máster

Análisis de datos: Estudio de la probabilidad de marcha de un empleado

Autor: Rubén Moirón Sánchez

Tutora: Yolanda Román Montoya

15/07/2021

UNIVERSIDAD DE GRANADA

Trabajo propuesto

Área del Conocimiento: Estadística e Investigación Operativa
Título: Análisis de datos: Estudio de la probabilidad de marcha de un empleado
Breve descrición del contenido
Análisis de los motivos y causas de la huida de un empleado de su empresa desde diversos enfoques, determinar patrones y métodos para la predicción de la marcha del empleado (*employee turnover*).

Índice general

Abstracto

Hoy en día, uno de los mayores problemas de las empresas es conseguir una correcta gestión de sus recursos. Dentro de estos recursos, cobra especial importancia el control de su principal activo, los empleados. Ellos son la cara de la empresa y afectan directamente en la experiencia y la satisfacción del cliente.

No cabe duda de que en las últimas décadas las empresas han empezado a preocuparse más por sus empleados. Aun así, la aplicación de programas de servicios para los trabajadores y de beneficios por trabajar en una compañía es todavía novedosa, al menos en lo que a cuestiones de alcance y de número de empresas aplicadoras se refiere.

Todo este creciente interés por el que los trabajadores se sientan cómodos y contentos trabajando en la compañía está muy ligado con los cambios generacionales (las expectativas de los trabajadores jóvenes respecto a la empresa), con los cambios sociales (las compañías se han adentrado en la responsabilidad social corporativa en las últimas décadas) y con los cambios de expectativas de los propios consumidores.

Por todo ello, una correcta gestión y control de la tasa de abandono o *turnover* es vital para el devenir de la empresa. Este trabajo se fundamenta en un exhaustivo análisis de información obtenida a partir de repositorios de Internet en el que, mediante el uso de la herramienta de programación R, se han observado los posibles condicionantes que un empleado pueda tener para decidirse a abandonar una empresa.

Dicho análisis se ha acompañado con representaciones gráficas que muestran las influencias que estas variables independientes (características del empleado) pueden tener sobre la variable objetivo o respuesta (marcha de empleado).

Se realizará un completo estudio sobre el tema en cuestión a través de distintas perspectivas.

Por un lado, se buscará desde una visión más sociológica o psicológica los motivos causas por los que se produce dicho suceso.

Por otro lado, se buscara un punto de vista más estadístico donde intervendrán desde

métodos estadísticos clásicos, hasta técnicas de *Machine Learning* y *Deep Learning*. El análisis tendrá como objetivo encontrar relaciones entre datos, pero, principalmente, la búsqueda del algoritmo o técnica más precisa para la predicción de marcha de un empleado, considerando la fuga de un empleado como la huida voluntaria de la empresa, es decir, que la compañía no es detonante de dicha huida.

Una vez realizado el estudio estadístico se mostraran unos resultados en forma de métricas que se comentarán y se buscará unas conclusiones técnicas del porqué suceden estos resultados, además de compararlos y comentar su eficiencia.

Todos estos métodos, técnicas y tests que se utilizan para realizar el análisis estadístico, serán debidamente explicados de una forma técnica e intuitiva, añadiendo, siempre que sea posible, gráficos y fórmulas que ayuden a su correcta compresión y entendimiento.

En un penúltimo apartado mostraremos un cuadro de mandos con un caso de uso dónde pueda verse la importancia de dicho análisis. Para la creación de este *dashboard* me he ayudado con la herramienta de visualización *SAP Analytics Cloud* para graficar los resultados. Se observará la plantilla de una empresa y aparecerán diferentes gráficos e indicadores que ayudarán a entender qué sucede con dicho personal para así tomar decisiones convenientes.

Por último, se mostrarán las conclusiones finales de dicho análisis centrándonos en los resultados obtenidos con los modelos predictivos creados.

Capítulo 1

Introducción

En esta primera sección se expondrán los motivos que me han llevado a elegir este tema y los objetivos que quiero abarcar en su desarrollo.

1.1. Motivación

Hoy en día, es frecuente escuchar el término "Big Data", las empresas buscan a expertos en esta área con la idea de que solucionen sus problemas de negocio, simplemente con el pensamiento de que estos expertos sacarán de la nada resultados que les harán ganar millones.

No todo pinta de color de rosa. Las expectativas generadas son muy grandes y los problemas muy complejos. Uno de los frentes que más preocupa a los empresarios es el de cómo evitar que sus trabajadores abandonen su empresa. Este histórico contratiempo se ha afrontado desde muchos puntos de vista tales como el filosófico, el psicológico, el económico, marketing... sin llegar a tener un "receta" perfecta que lo solucione.

Últimamente, con el crecimiento de las nuevas tecnologías y los enfoques científicos se está intentando atacar este problema a través de conceptos de inteligencia artificial, concretamente haciendo uso de la ciencia de datos. En este trabajo se á realizar un análisis que aporte más evidencia respecto a los procesos existentes tras estos problemas y claves para su solución.

1.2. Objetivos

El objetivo fundamental de este trabajo es obtener una respuesta "científica" al problema de la marcha de los empleados. Para ello, se realizará un análisis pormenorizado de cada una de las causas y motivos por los que pueda suceder esto, con el fin de extraer los posibles patrones y tendencias que le provoque a una persona abandonar, de manera voluntria, su puesto en una compañía.

Además, una vez obtenidas todas estas conductas, buscaremos para un caso actual de empleados en una plantilla, la probabilidad de marcha de estos empleados. Y una vez conseguida esta probabilidad, se pretenderá crear un marco donde se observe el qué ha pasado, porqué ha pasado y qué puede suceder con el plantel de esta empresa.

Los datos que usaremos para el desarrollo del ejercicio son obtenidos de Internet y están seudonimizados. Son usados como ejemplo del potencial que la ciencia de datos puede sacar de la información que una empresa tiene de ellos.

Capítulo 2

Turnover

A lo largo de este apartado trataremos el tema principal del trabajo, el *Employee Turnover* o marcha de empleado. Comenzaremos con un análisis semántico de dicha palabra y sus relaciones con el ámbito que nos interesa abarcar y continuaremos hablando sobre los motivos que lo provocan. También haremos una clasificación de los diferentes tipos de *turnover* que hay y sus impactos dentro de la estructura empresarial. Finalmente, comentaremos qué es el ratio de abandono y estableceremos una forma de cómo definirlo.

2.1. ¿Qué es el *turnover*?

La palabra *turnover* es de denominación inglesa y tiene como significado "acto o resultado de dar vueltas" (Dictionary.com, 2009). Una breve apreciación sobre la evolución etimológica de este vocablo permitió verificar que, además de éste, su significado original, pasó a utilizarse posteriormente para referirse al hecho de "cantidad de negocio que una compañía realiza en un periodo de tiempo" (Cambridge Online Dictionary, 2020).

Actualmente *turnover* se define también en el diccionario como "ratio según el cual los empleados dejan una compañía y son reemplazados por nuevas personas" (Cambridge Online Dictionary 2013).

Turnover se aplica asimismo en el área de gestión de personal de una organización para designar el *churn* o rotación del personal. Aunque esta última, suele ser más usada para referirse a la pérdida de clientes.

Sin embargo, en la revisión bibliográfica se ha verificado que *turnover* asume significados genéricamente similares en artículos publicados en lengua inglesa pero que presentan algunos matices tales como:

- Ratio de ventas para un periodo fijado para equilibrar el inventario (Merriam-Webster Online Dictionary, 2020).
- Ciclo de compra, venta y reemplazamiento de un stock de bienes (Merriam-Webster Online Dictionary, 2020).
- Número de persona contratadas en un periodo de tiempo para reemplazar a estas que han dejado por fuerza laboral (Merriam-Webster Online Dictionary, 2020).
- Movimiento (tanto de bienes como de personas) de entrada, salida o a través de un lugar (Merriam-Webster Online Dictionary, 2020).
- Cantidad de negocio hecho (Merriam-Webster Online Dictionary, 2020).

También tiene significados en otros campos como el deporte (en especial en el fútbol y el baloncesto, que se refieren al cambio de posesión) o el alimentario (un postre típico a base de hojaldre relleno de fruta).

En este contexto, proponemos traducir el vocabulario *turnover* por pérdida, y consecuentemente *turnover rate* por tasa de pérdida y *turnover management* por gestión de la pérdida. Por otro lado, en base a la revisión de la literatura especializada, y teniendo además en cuenta a la Real Academia Española y, sobretodo, para los efectos prácticos de este trabajo, el término abandono se entenderá como dejar o renunciar por voluntad propia un trabajo.

2.2. La importancia del *turnover*

La gestión del *turnover* es esencial para las empresas, principalmente respecto a la producción a la que se enfrentan diariamente. Un cierto índice de rotación es normal y es parte de la vida corporativa, sin embargo, cuando estos ratios son muy altos, la organización precisa de hacer un cuidadoso análisis interno e identificar lo que provocan esas dimisiones.

La pérdida de trabajadores no solo significa la salida de un profesional, sino que supone la pérdida de experiencia, conocimiento y compresión de los procesos internos. Además de eso, la productividad y los beneficios se ven seriamente afectados cuando se produce una desvinculación de los trabajadores en gran número.

Conocer los tipos *turnover*, porqué ocurren y las consecuencias que trazan para la empresa es esencial para entender mejor cómo las bajas funcionan. Esto provocará una

reflexión más profunda desde el proceso de selección, aumentando el cuidado a la hora de contratar, hasta el tratamiento con los funcionarios y las condiciones de trabajo ofrecidas.

De tal forma, es posible trazar estrategias de prevención y, políticas de gestión de personal con la vista puesta en mejorar la productividad de la empresa. Al fin y al cabo, con un cuadro de funcionarios satisfecho, la empresa conseguirá mejorar sus números.

2.3. ¿Por qué el *turnover*?

Los motivos de un alto índice de *turnover* son varios. A continuación, presentamos algunas razones que puede hacer que los trabajadores dejen una empresa.

2.3.1. Interferencia de terceras Empresas

El mercado puede ser cruel, principalmente cuando está caliente, y esto genera que pierdas algunos de tus mejores colaboradores. El motivo fundamental, en la mayoría de los casos, por lo que una persona deja un puesto de trabajo es porque tiene oferta de trabajo mejor. Los motivos que llevan a esta persona a aceptar otra oferta pueden ser de diversa índole. El tema económico suele pesar bastante en esta decisión, ya que es el principal motivo por la cual una persona comienza a trabajar.

También tiene especial importancia otros motivos, como el cambio de trabajo por mejores instalaciones, por ejemplo, cada vez más gente les interesa trabajar en oficinas más modernas, con espacios donde puedan desconectar, puntos con más luz que les amenizarán el trabajo, lugares donde puedan practicar ejercicio físico... Destacar, dadas las largas jornadas de trabajo, muchos trabajadores se ven obligados a comer/desayunar en la oficina, un servicio de restaurante es un punto a tener en cuenta en estas situaciones.

Otros motivos por los que se decantan los trabajadores a cambiar de trabajo por encontrar mejores horarios, más días de trabajo en remoto o desde casa o más días de vacaciones.

2.3.2. Personas no idóneas para los puestos de trabajo

Empresas con procesos de selección sin patrones establecidos o conducidos de forma ineficaz acaban contratando a personas con perfiles inadecuados para sus vacantes. Cuando sucede esto, los nuevos colaboradores no consiguen integrarse correctamente a los equipos y ejecutar bien sus tareas, provocando desmotivación y perjuicios para el negocio. Otros puntos que generan el *turnover* que están relacionados con los colaboradores son la falta de experiencia y la inadecuación al perfil de la vacante, este último es muy importante. Saber

identificar el perfil correcto para la vacante de tu empresa es fundamental. Por ejemplo, contratar a gente joven que aún está en proceso de formación puede ser un problema, porque mientras para el empresario puede ser que él/ella posee un camino trazado para su carrera, para el trabajador puede ver ese trabajo como algo temporal.

2.3.3. Problemas de comunicación

Procesos de comunicación ineficientes hacen a que los colaboradores no reciben informaciones e instrucciones de forma clara. Por consecuencia, no consiguen tener éxito en la realización de sus actividades lo que genera desvalorización del trabajo e insatisfacción.

2.3.4. Falta de nuevos líderes

Uno de los motivos más recurrentes por el cual los trabajadores deciden dejar una empresa es el descontento con sus superiores. Esto ocurre porque, en muchas organizaciones, los funcionarios que ocupan cargos de responsabilidad, no están capacitados para actuar como líderes y no consiguen motivar, inspirar o influenciar a sus colaboradores.

2.3.5. Gestión ineficiente

En empresas donde los procesos y las personas no están bien gestionados y donde, los funcionarios no identifican sus obligaciones y sus papeles dentro del negocio... Esto provoca que un equipo no se sienta respetado ni apreciado para reconocerlo. Uno de los mayores problemas está en la gestión de personas, cuando un gerente o supervisor no posee el tacto necesario para sacar el máximo potencial a sus colaboradores. Cuando eso ocurre, normalmente el colaborador se desanima, pues no se siente desafiado o motivado, y acaba optando por salir de la empresa.

2.3.6. Insatisfacción con las actividades desarrolladas

Los colaboradores que tienen que ejecutar actividades para las cuales no han sido preparados – o que no les permiten aplicar y desarrollar sus mejores conocimientos y habilidades – tienden a estar descontentos y a procurar nuevas oportunidades de trabajo.

2.3.7. Ambiente hostil

Un buen ambiente es esencial para que los funcionarios deseen permanecer en la organización. Cuando el clima es hostil, inestable y repleto de conflictos, los colaboradores no

se sienten cómodos, lo que genera insatisfacción y, en consecuencia, la voluntad de dejar la compañía.

2.3.8. Ausencia de plan de carrera

Los profesionales que desean evolucionar precisan de trabajar en un lugar que proporcione la aplicación y desarrollo de sus habilidades, así como la valoración de su desempeño.

2.4. Tipos de *turnover*

La manera en la que un empleado marcha de la empresa puede darse de diversas formas. Tanto es así, que podría decirse que las causas pueden ser únicas para cada empresa. La empresa "Kenoby"[6], especializada en la rotación de las empresas, ha creado una clasificación general de estos motivos dividiéndolos en cuatro:

2.4.1. Funcional

Este tipo de marchas ocurren cuando un trabajador pide desligarse de la empresa. En tal situación la empresa tiene la oportunidad de sustituirlo por otro trabajador sin necesidad de cubrir los costes por rescisión o despido.

El principal impacto provocado por este tipo de marcha es la recolocación de funciones, que las tareas de este trabajador sean absorbidas por otro. Otro gran impacto es el coste de selección de un nuevo trabajador para su posición.

Cuando un empleado pide su desvinculación, el probablemente está muy descontento con la empresa, así como que la empresa está muy descontenta con él. Hay una gran probabilidad de que ese trabajado haga valoraciones negativas sobre su experiencia a su grupo más cercano, en el cual puede estar su familia, amigos, compañeros de profesión y lugares donde evalúen empresas.

2.4.2. Disfuncional

Esta es una situación opuesta a la anterior. Este tipo de situaciones suponen cuando se da la desvinculación de un empleado de alto rendimiento. Es una situación que genera preocupación para la empresa, ya que provoca la pérdida de un trabajador importante para la composición de capital humano.

Además, demuestra que la organización no es capaz de retener ese talento, lo que indica problemas en las condiciones de trabajo ofrecidas.

La empresa pierde tanto financieramente (índices de productividad e innovación) como en conocimientos específicos que ese trabajador ofrecía.

2.4.3. Voluntario

Como su propio nombre indica, la pérdida voluntaria ocurre cuando el trabajador pide desligarse de la empresa. Es una de las marchas más comunes en los sectores privados, indicando problemas en la gestión y retención de talentos, y no solamente en los salarios o beneficios ofrecidos por la organización.

Este tipo de pérdida ocurre generalmente en determinadas circunstancias, como:

- El empleado recibe ofertas más interesantes de terceras empresas.
- Conflictos entre los trabajadores que convierten la permanencia en la empresa como algo inviable.
- Falta de plan de carrera claro o de oportunidades de crecimiento.
- Descontento con la remuneración o con las políticas internas.
- Motivos geográficos (cambio de ubicación laboral, tiempo invertido en el desplazamiento al trabajo, etc.).
- Motivos sociales.

Por la definición que hemos contado en el anterior apartado, este tipo de *turnover* voluntario también se le puede llamar abandono. Además, este será el tipo de *turnover* que utilizaremos para nuestro análisis, por lo que, siempre que nos referamos a *turnover* será siempre del tipo voluntario.

2.4.4. Involuntario

En estas situaciones, la marcha ocurre por iniciativa de la empresa. Es el tipo de marcha más habitual. Esta rotación puede provocar que la organización se enfrenta a serios problemas y generalmente sucede por los siguientes motivos:

- Bajo desempeño profesional
- Causa justificada.
- Conflictos con jefes o compañeros.

- Problemas económicos de la empresa.
- Fallos en el proceso de captación de la empresa

Además de los altos costes con los que se paga el despido, la empresa también precisa promover un nuevo proceso de selección y formar a los nuevos profesionales que va a contratar. El periodo de integración provoca grandes costes, puesto que la productividad durante ese periodo se verá lastrada con un ritmo más lento.

El *turnover* involuntario también afecta al clima organizativo, dejando al equipo restante temeroso en relación a su futuro en la empresa.

2.5. Impactos

Por ende, como se ha observado en la sección anterior, el *turnover* puede darse de dos maneras: cuando la empresa despide al operario o cuando este renuncia. Este primero puede generar un problema grande en tu empresa. Cuando el despido no es por causa justificada, los gravámenes laborales son costosos para la empresa. Eso puede causar un problema económico cuando es hecho varias veces consecutivas. El clima organizativo también puede quedar sacudido. Cuando una empresa comienza a despedir colaboradores sin una causa justificada tiende a generar comentarios de operarios. La inseguridad puede permanecer en la atmósfera del departamento o incluso en la empresa como un todo.

El despido por causa justificada, normalmente tiende a ser más fácil para la empresa. Este despido difícilmente va a perjudicar el clima organizativo, a no ser que el motivo de la dimisión sea algo que los colaboradores hagan a menudo, como utilizar el email corporativo o la impresora para fines personales.

Si alguien se marcha de la empresa debe entrar alguien para sustituir sus funciones o traspasárselas a otra persona. En ambos casos se produce un proceso de aprendizaje para ese empleado, un entrenamiento. El entrenamiento de un colaborador puede costar caro para la empresa. Encontrar alguien que entre en el ritmo de la empresa y traiga resultados no siempre sucede de manera fluida. Para eso es necesario entrenar esos colaboradores y encontrar un perfil que se encuadre en tu empresa.

El coste medio de remplazar a un candidato en la empresa se estima entre tres y seis meses de salario. Pero en ocasiones este gasto puede ser proporcional a nueve meses sueldo al incluir el coste del período de formación del nuevo colaborador.

Esta cifra incluye los costes directos derivados de la gestión administrativa de la dimisión y del proceso de contratación del nuevo candidato; pero también hay que tener en

cuenta los costes indirectos, como la pérdida de productividad vinculada a la falta de un miembro del equipo y el tiempo de inactividad hasta que el nuevo colaborador termina la etapa de formación.

Las pérdidas también incluyen el tiempo de gestión del equipo de RRHH, tanto para la dimisión como para la nueva contratación.

2.6. ¿Cómo se calcula la tasa de rotación?

El índice de rotación de personal mide la relación entre las personas que se incorporan al equipo y las que se marchan; es decir, el porcentaje de altas y bajas en relación al número de empleados en un determinado periodo temporal.

Existen diversas fórmulas y esta es una de las más exactas:

$$\frac{\text{Número de bajas} + \text{ Número de contrataciones}}{2 \times \text{ Empleados el 1 de enero}} \times 100$$

Pese a que e ste índice es realmente útil de conocer para cualquier empresa. Nosotros lo utilizaremos testimonialmente en la representación gráfica de última sección.

Capítulo 3

La Estadística en el análisis del *Turnover*

Hacemos un repaso de conceptos teóricos básicos de la metodología implementada en nuestro estudio. Esto es desarrollado a fin de tener los conocimientos necesarios a la hora de entender los trabajos, los resultados obtenidos y su interpretación. Para ello, veremos algunas definiciones, conceptos y propiedades que serán de utilidad para el análisis posterior.

3.1. Descripción del problema y conceptos básicos

El problema con el que nos enfrentamos es la búsqueda de empleados que se quieran fugar voluntariamente de la empresa, es decir, si un empleado se va o no se va. Solo se admiten dos posibilidades. Los problemas que admiten un número limitado de opciones se les considera de clasificación:

Decimos que un **problema** es **de clasificación** si tiene como objetivo predecir una respuesta cualitativa de una observación, es decir, predecir la categoría o clase de dicha observación. Los problemas de clasificación se caracterizan por tener una variable cualitativa como respuesta, estas variables cualitativas también reciben el nombre de variables categóricas.

Una **matriz de correlación** es una tabla que indica los coeficientes de conexión entre los factores. Cada celda de la tabla muestra la "conexión" entre los dos factores, esta "conexión" es el coeficiente de correlación de Pearson, que miden el grado de relación lineal

entre cada par de elementos o variables

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\,\mathrm{Var}(Y)}}$$

donde:

- (X, Y) son un par de variables aleatorias
- σ_{XY} es la covarianza de (X, Y)
- σ_X es la desviación estándar de la variable X
- σ_Y es la desviación estándar de la variable Y

Nuestro objetivo sera construir modelos que sean capaces de predecir la baja de clientes y atendiendo a la información de la que se dispone, podemos distinguir dos tipos de técnicas:

- **Aprendizaje supervisado**: está encaminado a la obtención de un modelo capaz de predecir el valor correspondiente de cualquier objeto (en nuestro caso, bajas de empleados) después de haber analizado la muestra de entrenamiento, dónde la variable respuesta es conocida. Basan su aprendizaje en un juego de datos de entrenamiento previamente etiquetados. Algunas técnicas son el algoritmo *Random Forest*, árboles de decisión, algoritmos de boosting....

- **Aprendizaje no supervisado**: son aquellos que realizan el ajuste de un modelo sin conocer datos a priori. Es decir, basan su proceso de entrenamiento en un juego de datos sin etiquetas o clases previamente definidas. Algunas de las más conocidas son: análisis de *clúster*, análisis de componentes principales (ACP), estudio de correlaciones, detección de anómalias, reglas de asociación, minería de datos...

La principal diferencia entre estos modelos es la variable que se busca predecir, ya que en un caso es conocida (aprendizaje supervisado) mientras que en el otro no.

En este trabajo abordaremos las técnicas predictivas basadas en aprendizaje supervisado ya que disponemos de la información proporcionada por el conjunto de datos, además de conocer los posibles valores de nuestra variable respuesta (fuga/ no fuga). Aunque también usaremos métodos no supervisados para otras tareas como en el análisis de conglomerados del que hablaremos en la próxima sección.

La **prueba chi cuadrado de Pearson** es un estudio no paramétrica que mide la discrepancia entre una distribución observada y otra teórica (bondad de ajuste), indicando en qué medida las diferencias existentes entre ambas, de haberlas, se deben al azar en el contraste de hipótesis. También se utiliza para probar la independencia de dos variables entre sí, mediante la presentación de los datos en tablas de contingencia.

La fórmula que da el estadístico es la siguiente:

$$\chi^2 = \sum_i \frac{(\text{ observada }_i - \text{ teorica }_i)^2}{\text{teorica }_i}$$

La prueba chi-cuadrado, se encuentra dentro de las pruebas pertenecientes a la estadística descriptiva, concretamente la estadística descriptiva aplicada al estudio de dos variables. Por su parte, la estadística descriptiva se centra en extraer información sobre la muestra. En cambio, la estadística inferencial extrae información sobre la población.

3.2. Análisis de conglomerados

El **análisis por conglomerados**, como técnica de análisis exploratoria, permite clasificar o separar volúmenes de valores o variables en diferentes grupos a subconjuntos de manera que los pertenecientes a una misma clase son homogéneos entre sí, difiriéndose de aquellos pertenecientes a otras clases.

Los algoritmos de agrupamiento jerárquico están dentro de la categoría de algoritmos de aprendizaje no supervisado

El análisis de conglomerados abarca tres tipos de problemas: la partición de los datos, la construcción de jerarquías y la clasificación de variables [21].

- **Partición de los datos**: Consiste en desarrollar la división en un número de grupos prefijados con el fin de salvar la dudosa heterogeneidad de los datos, de manera que todo elemento quede clasificado en un único grupo y sea internamente homogéneo.

- ***Construcción de jerarqías***: La similitud en los datos es tomada en consideración a fin de estructurar los mismos implicando una ordenación por niveles. Las medidas de similitud para las variables a menudo toman la forma de coeficientes de correlación de muestra. Además, en algunas aplicaciones de agrupamiento, las correlaciones negativas se reemplazan por sus valores absolutos.

- ***Clasificación de variables***: A fin de plantear los modelos formales para reducir la dimensión de datos y resultados, la gran mayoría de los modelos vienen precedidos por análisis exploratorios orientados a dividir las variables en grupos o estructuras jerárquicas. Como veremos más adelante, estas agrupaciones pueden definirse mediante matrices de distancias o similitudes entre elementos.

El análisis de conglomerados puede ser dividido en dos metodologías dependiendo del tipo de datos disponibles y del propósito particular del análisis. Estos dos métodos son las **técnicas de conglomerados jerárquicos**, que utilizan la matriz de distancias o

similitudes entre elementos, y las **técnicas de conglomerados no jerárquicos o de partición**, las cuales utilizan la matrices de datos.

Dentro de las técnicas de conglomerados jerárquicos nos interesa comentar un método que utilizaremos posteriormente, el **método de Ward**. Este método fue planteado como un procedimiento de agrupamiento enfocado en la búsqueda de la formación de particiones de una manera que minimice la pérdida asociada con cada agrupación y cuantifique esa pérdida en forma fácilmente interpretable. El objetivo del método de Ward es unificar grupos de modo que la variación dentro de estos grupos no aumente demasiado drásticamente. Esto da como resultado grupos en conglomerados que son lo más homogéneos posible, no agrupa grupos con la menor distancia, sino que une grupos que no aumentan demasiado en una determinada medida de heterogeneidad.

3.3. Discretización

Consideramos como **discretización** a los procesos que transforman datos cuantitativos en datos cualitativos.

Un método importante de discretización es el algoritmo AMEVA. Este algoritmo, propuesto por Gonzalez-Abril, Cuberos, Velasco and Ortega (2009) [17], maximiza el coeficiente de contingencia basado en el estadístico chi-cuadrado y genera un número mínimo potencial de intervalos discretos. Su más importante ventaja, en contraste con otros algoritmos de discretización, es que no necesita que el usuario indique el número de intervalos [22].

3.4. Métodos de balanceo de clases

Técnicas de **balanceo de clases o de remuestreo** se basan en modificar la distribución inicial de los datos para balancear las clases. Algunas de las más importantes y que usaremos en nuestro ensayo:

- ***Undersampling* o Submuestro**: Consiste en balancear la distribución de los datos eliminando instancias de la clase mayoritaria, es decir, el valor de la variable independiente en el estudio que tiene más registros que la mitad de los registros totales (en nuestro caso, se tratará de los empleados que permanecen).

  A pesar de su sencillez y de la reducción del tiempo de procesado de los datos, existe el riesgo de eliminar elementos de la muestra potencialmente importantes en el proceso

de clasificación, por eso se han desarrollado métodos capaces de realizar una selección inteligente sobre los elementos del conjunto de datos de la clase mayoritaria. La idea en la que se basan estos algoritmos es suponer que las instancias próximas entre sí tienen mayor probabilidad de pertenecer a la misma clase.

Algunas técnicas de submuestreo son *Random undersampling* (utilizaremos este método en el estudio), Tomek Links y Wilson Editing.

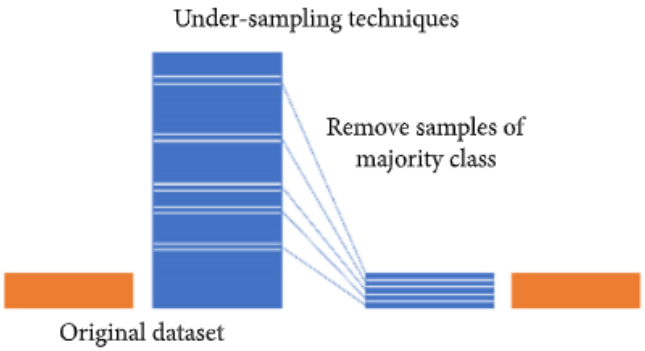


Figura 3.1: Estructura técnica de submuestro [23]

- ***Oversampling* o Sobremuestro:**

  Consiste en balancear la distribución de los datos añadiendo ejemplos de la clase minoritaria, es decir, el valor de la variable independiente en el estudio que tiene menos registros que la mitad de los registros totales (en nuestro caso, se tratará de los empleados que se fugan).

  Además de aumentar el tiempo de procesado de los datos, el principal problema de este método tiene lugar cuando generamos ejemplos ruidosos que se producen al realizar un elevado número de réplicas de ciertas instancias, modificando en exceso la distribución de las fugas, y que puede derivar en un incremento desmesurado de las probabilidades de fuga finales.

  Algunas técnicas de sobremuestreo son *Random oversampling* (utilizaremos este método en el estudio) y SMOTE (*Synthetic Minority Oversampling Method)*).

- ***Resampling* o Remuestro:**

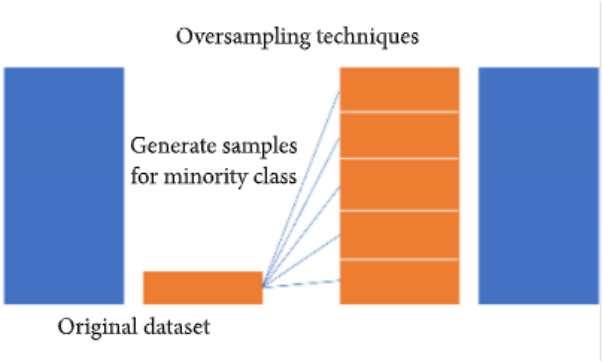


Figura 3.2: Estructura técnica de sobremuestro [23]

Consisten en combinar las técnicas de submuestreo y remuestreo que acabamos de describir.

Estos algoritmos constituyen una de las técnicas más empleadas hoy en día por los investigadores en cuanto a la corrección del problema del desbalanceo ya que se compone de dos técnicas que corrigen este problema modificando la distribución de los datos, además de tener una simple implementación a nivel computacional. A pesar de ello, sigue conservando los inconvenientes que lleva implícitos en cada una de las dos técnicas que lo conforman (*undersampling* y *oversampling*).

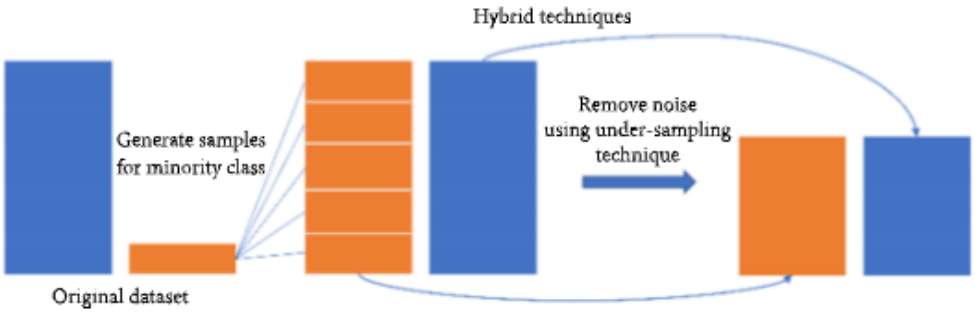


Figura 3.3: Estructura técnica de remuestro [23]

Existen varios métodos de hibridación como consecuencia de la gran diversidad de algoritmos de *undersampling* y *oversampling* que se han ido desarrollando, sin embar-

go en este trabajo combinaremos *random undersampling* con *random oversampling*, dada su practicidad.

3.5. Algoritmos y conjuntos de Algoritmos

3.5.1. Árboles de decisión

Un **árbol de decisión** es una forma gráfica y analítica de representar todos los posibles sucesos que pueden ocurrir a partir de una decisión asumida en un momento concreto. Además, nos ayudan a tomar la decisión más "acertada" desde un punto de vista probabilístico, ante un abanico de posibles decisiones. Por otro lado permite desplegar visualmente un problema y organizar el trabajo de cálculo que se deba realizar.

Los árboles de decisión constituyen una técnica estadística para la segmentación, la estratificación, la predicción, la reducción de datos, el

filtrado de variables, la identificación de interacciones, la fusión de categorías y la discretización de variables continuas. Estos múltiples usos han beneficiado a los analistas en la necesidad de describir condiciones y acciones así como las consecuencias que se deben considerar en la toma de decisiones. A pesar de esto, los árboles de decisión no siempre son la mejor herramienta para este tipo de análisis ya que el árbol de decisión de un sistema complejo con muchas secuencias de pasos y combinaciones de condiciones puede tener un tamaño considerable. El gran número de posibilidades con sus distintas alternativas puede suponer un problema más que una ayuda para el análisis.

Entre sus ventajas, están: la sencillez del modelo, la amplitud de implementaciones que existen, la rapidez de clasificación de nuevos patrones, la posibilidad de representarlos gráficamente aportando así una explicación de la división efectuada y la fácil interpretación.

Algunos ejemlos de árboles de decisión son: ID3, C4.5 y CHAID.

En la Figura 3.4 se puede observar un ejemplo de árbol de decisión formado por siete nodos, cuatro de ellos finales y, debajo de cada nodo, aparece la condición por la que se "parte" el árbol, es decir, los puntos de inflexión que usan para tomar las decisiones de como clasificar los casos. Se ve dentro de cada nodo, en primer lugar, el número del nodo, en la segunda fila, el porcentaje de casos que pasan por dicho nodo para cada clase, y en la tercera, el porcentaje total de casos que pasan por dicho nodo.

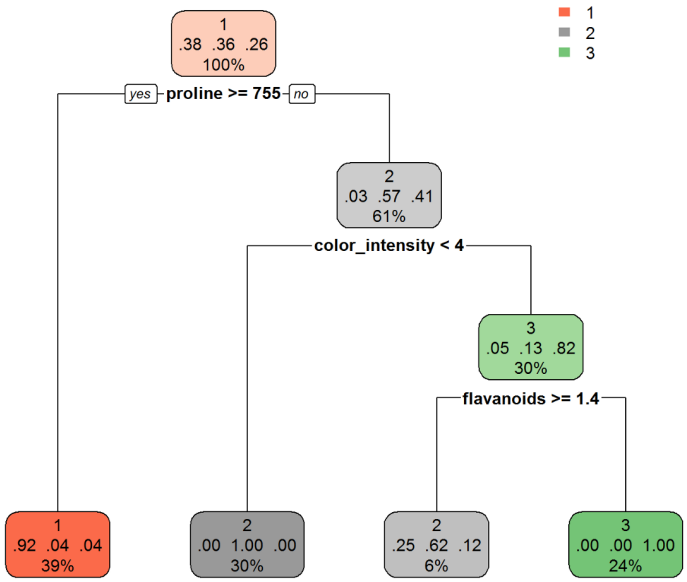


Figura 3.4: Ejemplo árbol de decisión [24]

3.5.2. Boosting

Este multiclasificador nace en el año 1989 de la mano del profesor Robert Schapire y mejorado por John E. Freund. Se basa en la construcción de sucesivos clasificadores, sobre modificaciones de la muestra de entrenamiento realizadas en función de los errores cometidos por el clasificador en cada una de las iteraciones, que posteriormente combinaría para obtener el clasificador final. Esto evidencia una clara dependencia entre los subclasificadores, lo cual constituye uno de los inconvenientes más grandes de este método.

Destacaremos el algoritmo ***Adaboost*** [26] como uno de los de algoritmos boosting más utilizados, ya que dirige su atención en aquellos casos que son más difíciles de clasificar. El procedimiento en el que se basa este algoritmo multiclasificador para construir el modelo de clasificación es el siguiente:

1. Inicialmente a todos los datos del conjunto de entrenamiento (muestra training) se les ssigna el mismo peso, $w_i = \frac{1}{n}$, donde n es el tamaño de la muestra.

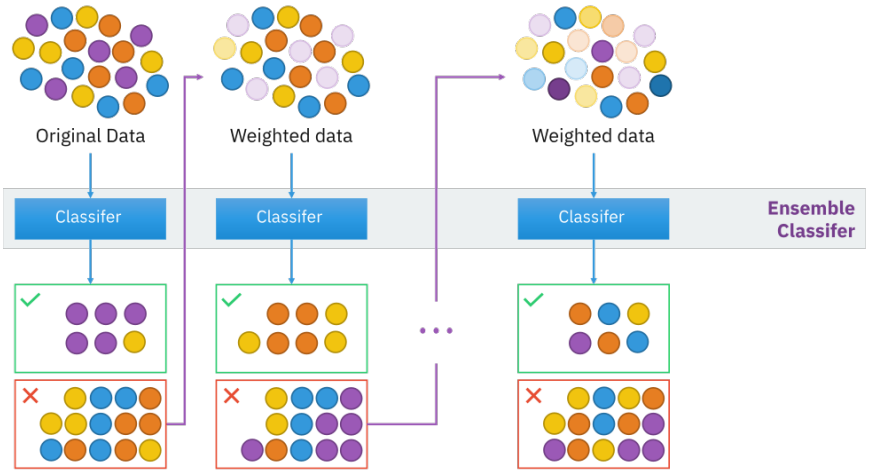


Figura 3.5: Estructura clasificador Boosting [25]

2. Se procede con una primera clasificación y se calcula el error cometido por el modelo usando la muestra training, contando e identificando los objetos mal clasificados.

3. Se incrementan los pesos de los casos de la muestra training que el modelo ha clasificado de forma errónea.

4. Entrenamos de nuevo el modelo usando el conjunto de pesos modificados.

5. Volvemos al punto 2y repetimos este procesos tantas veces como el número de iteraciones que se haya fijado.

6. Nuestro modelo final será una votación ponderads por los pesos de todos los modelos que hemos construidos durante la ejecución del algoritmo.

3.5.3. Bagging

El método ***Bagging*** (***Bootstrap Aggregating***) es un clasificador estadístico propuesto por Leo Breiman en el año 1994.

Particularidades a parte, su funcionamiento consta de varias fases:

1. En primer lugar, genera a partir de la muestra inicial varios subconjuntos del mismo tamaño con reemplazamiento (asegurando así la diversidad) y de forma independiente.

2. Para cada uno subconjunto, construye un subclasificador.

3. Utilizando el voto mayoritario obtiene la clasificación final para cada individuo.

A diferencia del *boosting*, en este método no existe ningún tipo de dependencia entre

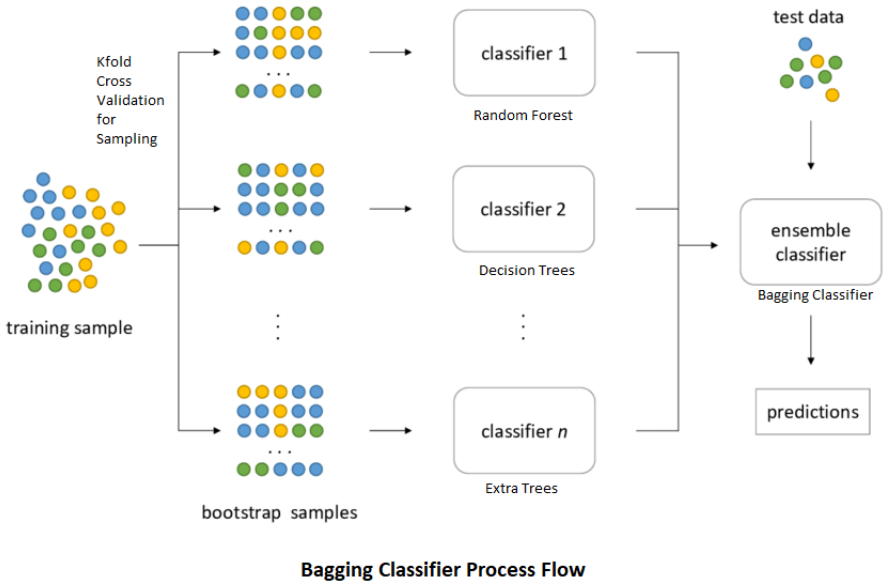


Figura 3.6: Estructura clasificador Bagging [27]

los subclasificadores por lo que el tiempo necesario para construir el clasificador final no aumenta de forma proporcional al número de subclasificadores utilizados. Otra diferencia es que en el *boosting* utilizamos todas las observaciones en cada iteración (en el *bagging* no necesariamente). Para datos con bajo nivel de ruido, *boosting* tiene mejor rendimiento que *bagging*, sin embargo, si los datos de entrenamiento contienen ejemplos ruidosos, entonces *boosting* degrada su rendimiento ya que focaliza el algoritmo en este tipo de datos.

3.5.4. *Random Forest*

Un refinamiento de los algoritmos de *bagging* es el algortimo ***Random Forest***: propuesto por Leo Breiman en 2001, presenta la misma estructura que él a excepción de una previa selección de variables antes de generar las muestras *bootstrap*. Esto le permite ser un clasificador muy certero aunque de compleja interpretación. Trabaja de forma eficiente con bases de datos grandes y puede manejar cientos de variables sin excluir a ninguna. Sin embargo, en los casos dónde que la base de datos contiene variables categóricas con diferente número de niveles, el *Random Forest* se parcializa a favor de aquellos atributos con más categorías. Por tanto, la posición que marca la variable no es fiable para este tipo de datos aunque existen soluciones que remedian esta cuestión, por ejemplo, las permutaciones

parciales.

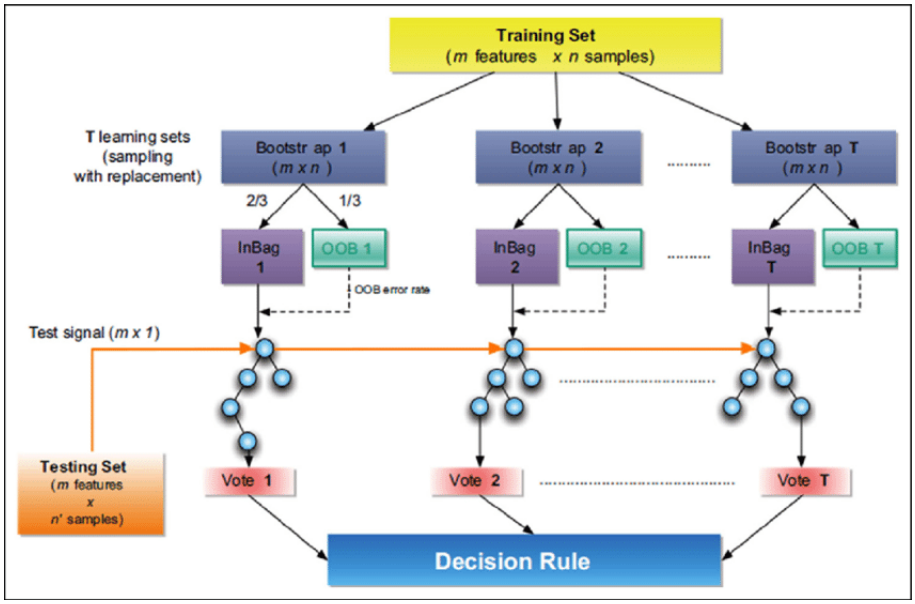


Figura 3.7: Estructura algoritmo Random Forest [28]

La forma de construir el clasificador final mediante este método puede resumirse en los pasos que a continuación resumimos:

1. Creamos aleatoriamente varios conjuntos de entrenamiento (con reemplazamiento) con el mismo tamaño que el conjunto original (muestra *training*). Al seleccionarse de esta forma, no todos los datos de la muestra *training* estarán necesariamente en los subconjuntos de entrenamiento. En cada punto de división del árbol, la búsqueda de la mejor variable para dividir los datos no se realiza sobre todas las variables de las que dispone el modelo, sino sobre un subconjunto de las mismas. Se busca la mejor división de los datos de entrenamiento teniendo en cuenta solo las variables seleccionadas.

2. Para cada uno de los diferentes subconjuntos construidos se obtiene el correspondiente subclasificador.

3. Una vez construidos todos los subclasificadores, la evaluación de cada nueva entrada se realiza con el conjunto de árboles generados por el *Random Forest*. La categoría final de cada individuo (clasificación) se lleva a cabo mediante el voto mayoritario del conjunto de árboles, y, en caso de regresión, por el valor promedio de los resultados.

3.5.5. Regresión Logística Simple

La **regresión logística simple** [30] es un clasificador que consiste en obtener una función logística de las variables independientes que permita clasificar a los individuos en una de las dos subpoblaciones o grupos establecidos por los dos valores de la variable independiente. Para este caso se supone que la variable binaria que se quiere predecir sigue una distribución de *Bernouilli*.

La función logística es aquella que halla, para cada registro según los valores de una serie de atributos, la probabilidad (p) de que presente el efecto estudiado.

La expresión matemática de la regresión logística (respecto a la egresión lineal)[31]:

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 - \beta_2 x_2 + \ldots + \beta_k x_k$$

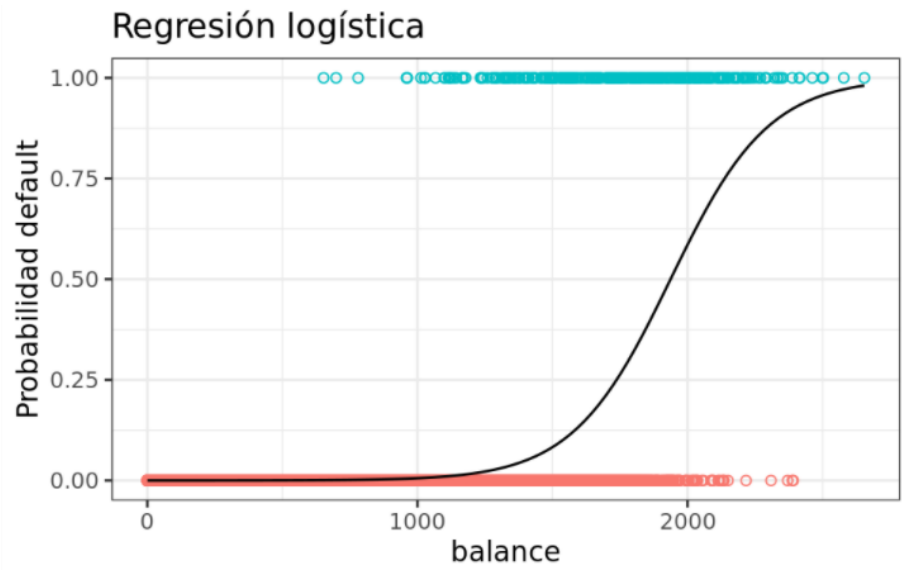


Figura 3.8: Ejemplo Regresión Logística [29]

La denominación del tipo de regresión y la función que se toma en el lado izquierdo provienen de la función sigmoide y los logits. Se entiende como un logit la expresión que está en el lado izquierdo de la fórmula, la cual permite que el valor predicho por la regresión logística sea un número entre 0 y 1 . Por tanto, un logit es:

$$\text{logit}(p) = \log\left(\frac{p}{1-\bar{p}}\right) = \log(p) \mid + \log(1-p)$$

A su vez la función sigmoide o curva logistica ocupa un lugar importante en este tipo de regresión ya que transforma cualquier número en un número en el intervalo $[0, 1]$, esto es, en una probabilidad.

3.5.6. Máquinas de Vector de Soporte (SVM)

Las **Máquinas de Vector de Soporte** (SVM, *Support Vector Machines*) [33] son un conjunto de algoritmos en los que se utiliza un hiperplano para separar los puntos etiquetados con diferentes categorías, dejando los puntos de una categoría a un lado del plano y los de otra al otro.

Un **hiperplano** es la generalización del concepto de plano que divide el espacio tridimensional en dos con cualquier número de dimensiones. Por ejemplo, en un espacio unidimensional (una recta) el hiperplano es un punto, mientras que en un espacio bidimensional (un plano) el hiperplano es una línea.

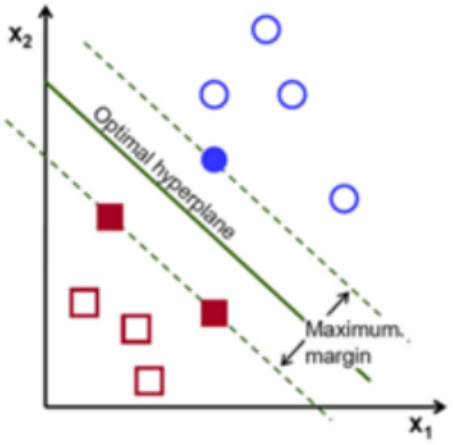


Figura 3.9: SVM

3.5.7. *Naïves-Bayes*

Este clasificador se basa en el **Teorema de *Bayes***. Se busca la hipótesis más probable dedos los datos. Se pretende clasificar un nuevo ejemplo de acuerdo con el valor más probable dados los valores de sus atributos.

El teorema de *Bayes* [19] se define como:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Dónde:

- A, B son eventos.
- $P(A \mid B)$ es la probabilidad de A dado B.
- $P(B \mid A)$ es la probabilidad de B dado A.
- $P(A), P(B)$ son las probabilidades independientes de A y B.

Y un clasificador bayesiano ingenuo se puede escribir como:

$$P\left(A \mid x_1, x_2, \cdots, x_n\right) = \frac{1}{P(\bar{x})} P(A) \prod_{1=1}^{n} P\left(x_i \mid A\right)$$

$P(\bar{x})$ es un factor de escala que depende de los valores de $x_1, x_2, \cdots, x_n$, es decir, una constante si son conocidos los valores de x_i.

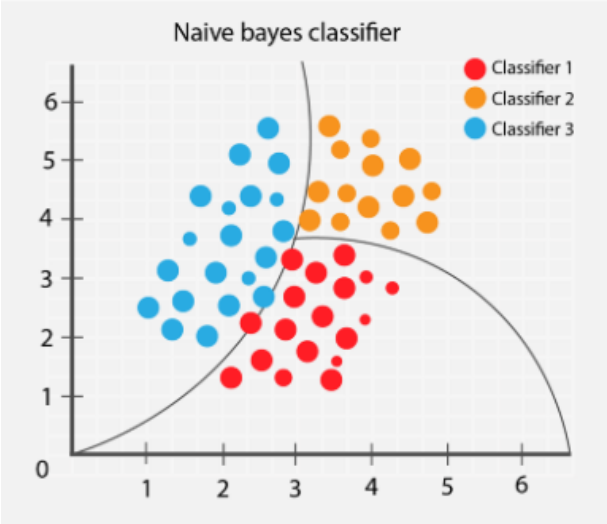


Figura 3.10: Ejemplo clasificación Naïve Bayes[20]

El problema de este clasificador es la asunción de que los valores de los atributos son condicionalmente independientes dado el valor de la clase y que los dicho valores son independientes de los valores de otro atributo. A esta suposición se le llama "independencia condicional de clase". Provoca una simplificación de los cálculos involucrados. Esta suposición es una simplificación de la realidad. En general, este clasificador funciona muy bien,

sobre todo cuando se filtra el conjunto de atributos seleccionado para eliminar redundancia, con lo que se elimina también dependencia entre datos.

3.6. Modelos de Redes Neuronales

Las **redes neuronales** [32] son modelos inspirados en el comportamiento de las neuronas de los cerebros, que basan su inteligencia en las conexiones entre sistemas más simples llamados neuronas.

Son modelos creados con operaciones matemáticas y que tienen una estructura formada por un nnúmero de capas, formadas por neuronas dónde cada neurona está conectadas con neuronas de las capas anterior y posterior mediante pesos, con el objetivo de regular la informacón que se propaga entre neuronas y, además, realiza una operación más sencilla.

Las capas de se dividen como capa de entrada o *input*, capa de salida o *output* y capa intermedia o *hidden layer*. La capa intermedia recibe los valores (ponderados por los pesos) de la capa de entrada y la capa de salida, combina los valores que salend e la intermedia y crea la predicción final.

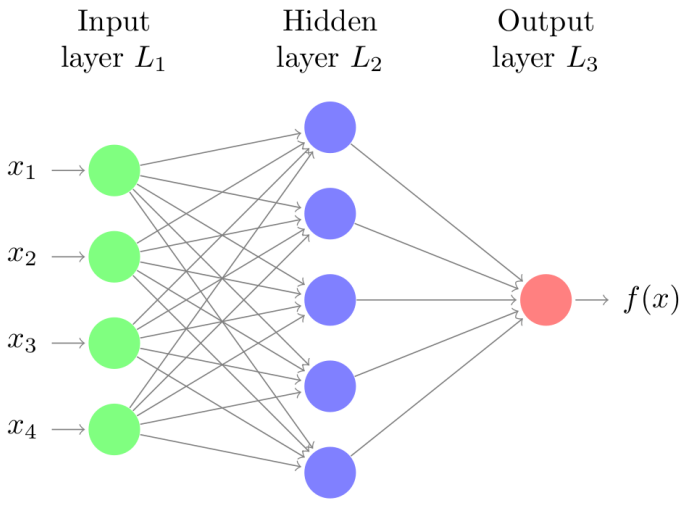


Figura 3.11: Estructura de red neuronal sencilla [32]

3.7. Estadísticos de evaluación de los modelos

Para medir el rendimiento de un modelo clasificador debemos

fijarnos en la tasa de error cometido por el mismo, es decir, tendremos que calcular el porcentaje de casos clasificados de forma incorrecta sobre el conjunto de datos con los que estamos trabajando.

Realmente, medir la tasa de error sobre los mismo datos sobre los que se ha generado el modelo, no es una buena práctica y puede concurrir en sobre juste o *overfitting* que sucede cuando el modelo tiene una tasa de error muy pequeña, que puede deberse a que el modelo no predice correctamente, sino que ha "memorizado" los resultados.

Para solucionar este problema, un posible solución es dividir los datos en dos subconjuntos distintos: la muestra de entrenamiento que será de un tamaño similar al 70 % del conjunto total de datos y que usaremos para la creación de los modelos, y la muestra de test, que tendrá de tamaño el 30 % restante usaremos para comprobar la calidad de predicciones. Una vez obtenidos estos dos subconjuntos, el procedimiento a seguir es el siguiente: generamos un modelo predictivo con los datos de entrenamiento usando el algoritmo de clasificación que corresponda y aplicamos dicho modelo a los datos de test.

Para medir la habilidad haciendo predicciones de los clasificadores se hace uso de una tabla conocida como la **matriz de confusión**, que constituye la herramienta de visualización más empleada en aprendizaje supervisado. Su papel es comparar las predicciones del clasificador con los valores reales. Para nuestro caso, compararemos las clases que el modelo predice para cada empleado (fuga o no fuga) con el estado real en el conjunto de datos.

	Real	
Predicción	NO	SI
NO	a	b
SI	c	d

Las métricas son indicadores que comprueban cómo de bueno es el modelo elegido. Existen cientos de métricas, pero nosotros nos centraremos en las siguientes:

- ***Accuracy***: mide la frecuencia con la que el clasificador hace una predicción correcta. Es la relación entre el número de predicciones correctas y el número total de predicciones.

$$Accuracy = \frac{\text{Predicciones correctas}}{\text{Individuos totales}} = \frac{\text{a} + \text{d}}{\text{a} + \text{b} + \text{c} + \text{d}}$$

- ***Precission***: representa el porcentaje de casos positivos predichos correctamente, en nuestro caso, el porcentaje de marchas clasificadas de forma correcta. Es la relación entre el número de predicciones positivas correctas y el número total de predicciones positivas.

$$Precission = \frac{\text{Verdaderos positivos}}{\text{Positivos predichos}} = \frac{\text{d}}{\text{c + d}}$$

- ***Sensibility***: representa la capacidad para detectar una condición correctamente, en nuestro caso, corresponde a la capacidad para detectar una baja. Es la relación entre el número de predicciones positivas correctas y el número total de positivos reales.

$$Senbility = \frac{\text{Verdaderos positivos}}{\text{Positivos reales}} = \frac{\text{d}}{\text{c¡a + d}}$$

- ***F1-Score***: es la medida de precisión que tiene un test y suele emplearse en la fase de pruebas de algoritmos de búsqueda y recuperación de información y clasificación de documentos. Se considera como una media armónica que combina los valores de la precisión y de la sensibilidad

$$F_1 = 2 \cdot \frac{\text{Precision } \cdot \text{ Sensibilidad}}{\text{Precision } + \text{ Sensibilidad}}$$

- **Curva *ROC* (*Receiver Operating Characteristic*)**: Muestra la sensibilidad del clasificador por el trazado de la tasa de verdaderos positivos respecto a la tasa de falsos positivos. En otras palabras, muestra cómo se pueden obtener muchas clasificaciones positivas correctas cuando se permiten más y más falsos positivos.

- **AUC (*Area Under Curve*):** es el valor bajo la curva ROC (el área entre el eje de abscisas y la curva ROC).

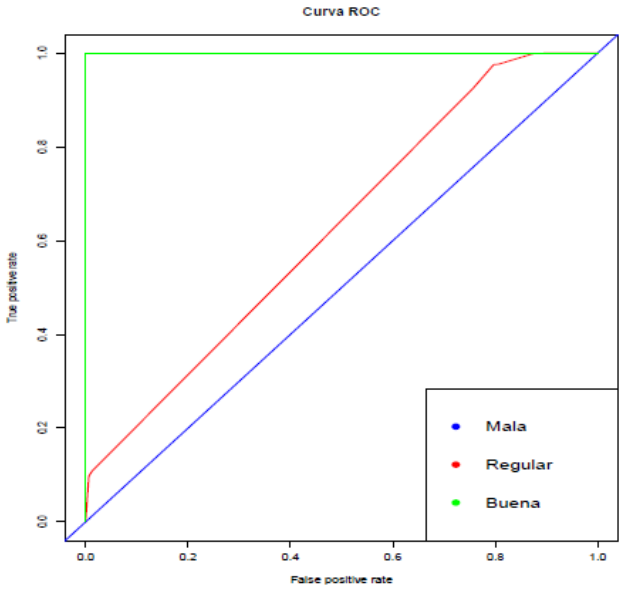


Figura 3.12: Curva ROC [15]

Capítulo 4

Análisis de Datos con R

En esta parte nos centraremos en el análisis estadístico de nuestros datos.

En primer lugar, hablaremos de los datos que hemos usado para este análisis, daremos una breve explicación de cada una de las variables que interviene en el análisis.

Habrá una segunda parte que se centra en el análisis de estos datos de formas diversas.

Y, por último, mostraremos una tabla de resultados de las diferentes medidas de los algoritmos utilizados y discutiremos cuál de ellos es el óptimo para nuestro caso.

4.1. Conjunto de datos

Para el desarrollo del análisis estadístico hemos usado un conjunto de datos extraído de internet [1] que consta de los siguientes campos:

- *Satisfaction_level* (Nivel de satisfacción): variable que mide el estado de satisfacción del empleado.
- *Last_evaluation* (Última evaluación): resultado de la última calificación efectuada sobre el trabajo por parte de su responsable.
- *Number_project* (número de proyectos): cantidad de proyectos en los que está asignados dicho empleado.
- *Average_monthly_hours* (media de horas mensuales): número de horas, de media, que realiza al mes.
- *Time_spend_company* (tiempo en la compañía): antigüedad en la empresa.

- *Work_accident* (accidente laboral): indica si el empleado ha tenido un accidente laboral en la compañía.
- *Promotion_last_5_years* (promoción últimos 5 años): si ha sido promocionado en los últimos 5 años.
- *Departament* (departamento): Departamento donde trabaja dicha persona.
- *Salary* (salario): escala salarial a tres niveles: salario alto, salario medio y salario bajo.
- *Age* (edad).
- *Gender* (Género).
- *Left* (marcha): indica si el empleado se ha ido o permanece en la compañía.

La variable *left*, que durante el transcurso del análisis cambiaremos a *voluntary_ turnover* (marcha voluntaria), es nuestra variable objetivo. Al ser una variable que toma solamente dos valores, marcha (1) o permanencia (0), nos encontramos ante un problema de clasificación.

	A	B	C	D	E	F	G	H	I	J	K	L
1	satisfaction_	last_evaluati	number_pro	average_mo	time_spend_	Work_accide	promotion_l	department	salary	Age	Gender	left
2	0.58	0.74	4	215	3	0	0	sales	low	57	Female	0
3	0.82	0.67	2	202	3	0	0	sales	low	39	Male	0
4	0.45	0.69	5	193	3	0	0	sales	low	62	Male	0
5	0.78	0.82	5	247	3	0	0	sales	low	37	Male	0
6	0.49	0.6	3	214	2	0	0	sales	low	35	Male	0
7	0.36	0.95	3	206	4	0	0	sales	low	34	Female	0
8	0.54	0.37	2	176	2	0	0	sales	low	43	Male	0
9	0.99	0.91	5	136	4	0	0	sales	low	33	Male	0
10	0.5	0.75	6	127	3	0	0	sales	low	46	Male	0
11	0.74	0.64	4	268	3	0	0	sales	low	55	Male	0
12	0.56	0.58	4	258	3	0	0	sales	medium	44	Male	0
13	0.34	0.39	2	136	3	0	0	sales	medium	36	Male	0
14	0.48	0.94	5	255	6	0	0	accounting	medium	41	Male	0
15	0.73	0.62	3	218	3	0	0	accounting	medium	20	Male	0
16	0.59	0.87	3	268	4	0	0	accounting	medium	24	Male	0
17	0.81	0.57	3	224	2	0	0	hr	medium	43	Male	0
18	0.9	0.66	3	231	3	0	0	hr	medium	22	Male	0
19	0.41	0.84	6	191	6	0	0	hr	medium	30	Female	0
20	0.89	0.92	4	165	5	0	0	hr	medium	36	Male	0
21	0.48	0.84	4	252	3	0	0	technical	medium	46	Female	0
22	0.79	0.97	5	266	2	0	0	technical	medium	52	Male	0
23	0.98	0.66	5	248	3	0	0	technical	medium	32	Female	0
24	0.75	0.7	4	144	4	0	0	technical	high	48	Female	0
25	1	0.41	4	174	3	0	0	technical	low	24	Male	0
26	0.24	0.82	5	179	6	0	0	technical	medium	60	Male	0

Figura 4.1: Tabla de datos

El resto de variables son las llamadas variables independientes que usaremos para entender el porqué de la variable *left*, es decir, buscaremos mediante estas variables llegar a saber el porqué de la variable *left* y conseguir preverla en futuros casos.

4.2. Análisis de datos

Una vez explicado con qué vamos a predecir el *turnover* podemos empezar con el análisis.

4.2.1. Libros estadísticos

En primer lugar, comentaremos los paquetes estadísticos de *R* que se usaran para el análisis y se comentará su función en dicho análisis:

- *randomForest*: se utilizarà para la calsificación y regresión basada en *forest* de árboles usando entradas aleatorias (*Random Forest*).
- *dplyr*: herramienta rápida y consitente para trabajar con *data frames* como objetos, en memoria o fuera de memoria.
- *adabag*: se utilizará dicho libro para la aplicación de algoritmos de *bagging*.
- *rpart*: se utilizará dicho libro para la construcción de algoritmos de *boosting*.
- *rpart.plot*: visualización de resultados obtenidos con funciones de la librería *rpart*.
- *ROCR*: obtención gráficos *ROC*, matrices de confusión y sus métricas.
- *FactoMineR*: análisis y minería de exploración multivariable de datos.
- *nomclust*: análisis de clústeres jerárquicos.
- *fastDummies*: creación de columnas y filas *dummies* para variables categóricas.
- *corrplot*: matrices de correlación.
- *lattice*: visualización de gráficos "enrejados".
- *arulesCBA*: clasificación basada en reglas de asociación.
- *caret*: entrenamiento para procesos de clasificación y regresión.

- *DMwR2*: funciones de *Data Mining*.
- *tidyr*: realizar operaciones sobre el dataset de datos.
- *keras*: operaciones sobre redes neuronales.
- *ggplot2*: visualizaciones de datos.
- *recipes*: preprocesamiento de datos.
- *e1071*: aplicación de algoritmo *"Support vector machines (SVM)"*.
- *yardstick*: operaciones y métricas de Matrices de confusión.
- *formattable*: visualización de tablas.
- *r-tensorflow:* Interfaz a *Tensorflow* utilizada para computación numérica utilizando grafos de flujo de datos.

4.2.2. Datos de estudio

El conjunto de datos tratado a lo largo de nuestro estudio ha sido obtenida a partir de la web *Kaggle* [1]. Dicho conjunto de datos trata de 1499 registros formados por 12 variables de las cuales una es la variable objetivo o dependiente (*left*). Se expone, a continuación, al detalle cada una de las variables seleccionadas:

- *satisfaction_level*: representa el nivel de satisfación del empleado con la empresa.
- *last_evaluation*: resultado de la última evaluación del empleado por parte de la empresa.
- *number_project*: número de proyectos asignados a dicho empleado.
- *average_montly_hours*: media de horas mensuales realizadas.
- *time_spend_company*: tiempo que el empleado ha estado trabajando en la compañía (en años).
- *Work_accident*: si ha sufrido o no algún accidente laboral.
- *promotion_last_5years*: si ha sido promocionado en los últimos cinco años.
- *department*: departamento en el que está asignado el empleado.

- *salary*: rango salarial del empleado (alto, bajo o medio).
- *Age*: edad de dicho empleado.
- *Gender*: género del empleado.
- *left*: si el empleado ha dejado o no la compañía, será la variable de estudio.

4.2.3. Limpieza de dataset

Visualizamos como está distribuida la variable objetivo.

```
df %>%
  count('left')
```

```
##   left  freq
## 1    0 11428
## 2    1  3571
```

Como podemos observar, en la mayoría de casos, la variable objetivo es igual a 0, es decir, que el empleado no se marcha.

Eliminaremos los registros para las variables independientes y contaremos de nuevo los registros que quedan. Para la variable objetivo no hace falta ya que en el comando anterior se ha visto que todos los registros son "0" o "1".

```
df <- df %>%
  drop_na()
```

Volvemos a contar con cuántas variables nos quedamos. Observamos que tenemos las mismas filas que el fichero sin limpiar, es decir, no había registros vacíos en las variables independientes.

```
nrow(df)
```

```
## [1] 14999
```

Eliminamos espacios en blanco de nuestros datos para no tener problemas en futuros cálculos.

```
for(j in 1:ncol(df))
{
  df[,j][df[,j]==""] <- NA
  df[,j] <- sapply(df[,j], function(x){gsub("[[:space:]]", "", x)})
}
```

Con esto, finalizaremos nuestra limpieza de datos. Ahora, crearemos un dataframe renombrando alguna variable, para tener una compresión más fácil de ellas. Destacar, que renombraremos la variable objetivo a "*voluntary_ turnover*".

```
df <- data.frame(
satisfaction_level=as.numeric(as.character(df$X.U.FEFF.satisfaction_level)),
last_evaluation=as.numeric(as.character(df$last_evaluation)),
number_project=as.numeric(as.character(df$number_project)),
average_montly_hours=as.numeric(as.character(df$average_montly_hours)),
time_spend_company=as.numeric(as.character(df$time_spend_company)),
Work_accident=as.factor(as.character(df$Work_accident)),
promotion_last_5years=as.factor(as.character(df$promotion_last_5years)),
department=as.factor(as.character(df$department)),
salary=as.factor(as.character(df$salary)),
Gender=as.factor(as.character(df$Gender)),
Age=as.numeric(as.character(df$Age)),
voluntary_turnover=as.factor(as.character(df$left))
)
```

4.2.4. Análisis Exploratorio

Para el análisis exploratorio de datos (*EDA*), en primer lugar, visualizaremos la distribución de las variables independientes en base al *turnover* y luego buscaremos las dependencias entre estas. Primeros veremos cómo están distribuidas las variables categóricas.

```
df %>%
  select_if(is.factor) %>%
  select(voluntary_turnover, everything()) %>%
  gather(x, y, Work_accident:Gender) %>%
  dplyr::count(voluntary_turnover, x, y) %>%
```

```
  ggplot(aes(x = y, y = n, fill = voluntary_turnover, color =
 voluntary_turnover)) +
  facet_wrap(~ x, ncol = 3, scales = "free") +
  geom_bar(stat = "identity", alpha = 0.5) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
legend.position = "top")
```

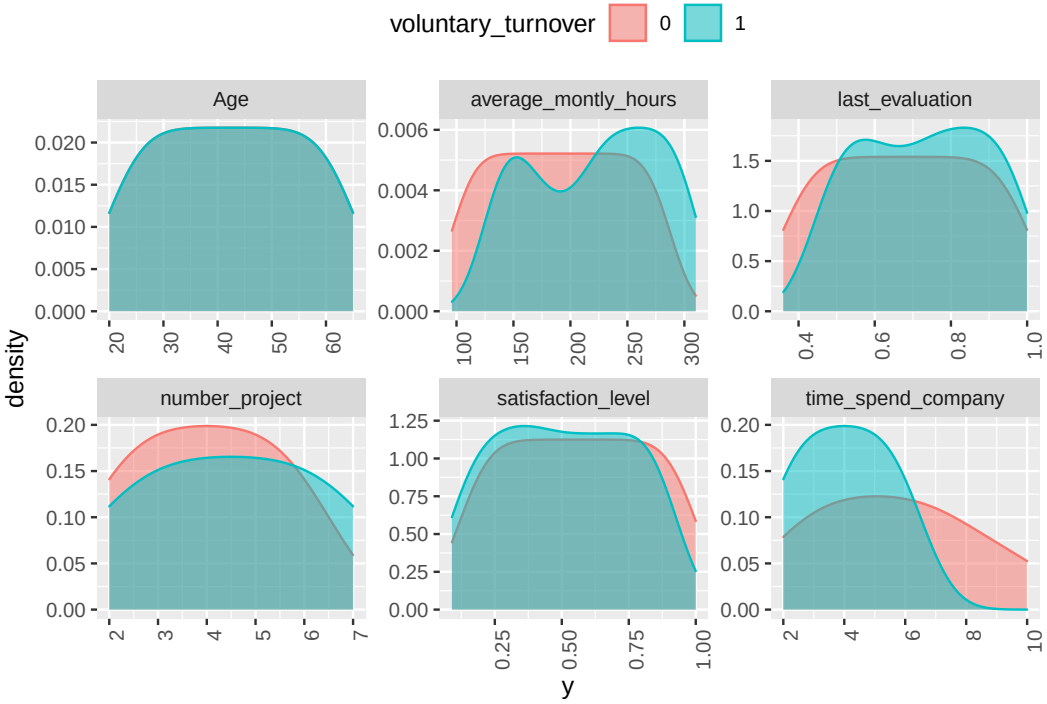
Vemos que hay ciertos departamentos ("*sales*", "*support*" y "*Technical*") que acumulan la gran parte del *turnover*, también tiene que ver que son los tres más grandes. También parece que influye que no hayan sido promocionados en los últimos 5 años, que tengan un salario medio o bajo y que no hayan tenido un accidente laboral.

Mientras que el género parece no importar (aunque se aprecia que para en el género femenino hay, a nivel proporcional, más fugas).

De la misma forma, para las variables continuas.

```
df %>%
  # select_if(is.numeric) %>%
  select(voluntary_turnover, satisfaction_level, last_evaluation,
         number_project, average_montly_hours, time_spend_company, Age) %>%
```

```
  gather(x, y, satisfaction_level:Age) %>%
  dplyr::count(voluntary_turnover, x, y) %>%
  ggplot(aes(x = y, fill = voluntary_turnover, color = voluntary_turnover))
+   facet_wrap(~ x, ncol = 3, scales = "free") +
  geom_density(alpha = 0.5) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
legend.position = "top")
```



Observamos que la distribución no presenta diferencias evidentes a simple vista excepto, para el *time_spend_company*, donde se aprecia que los que los que se han marchado llevaban menos años en la compañía.

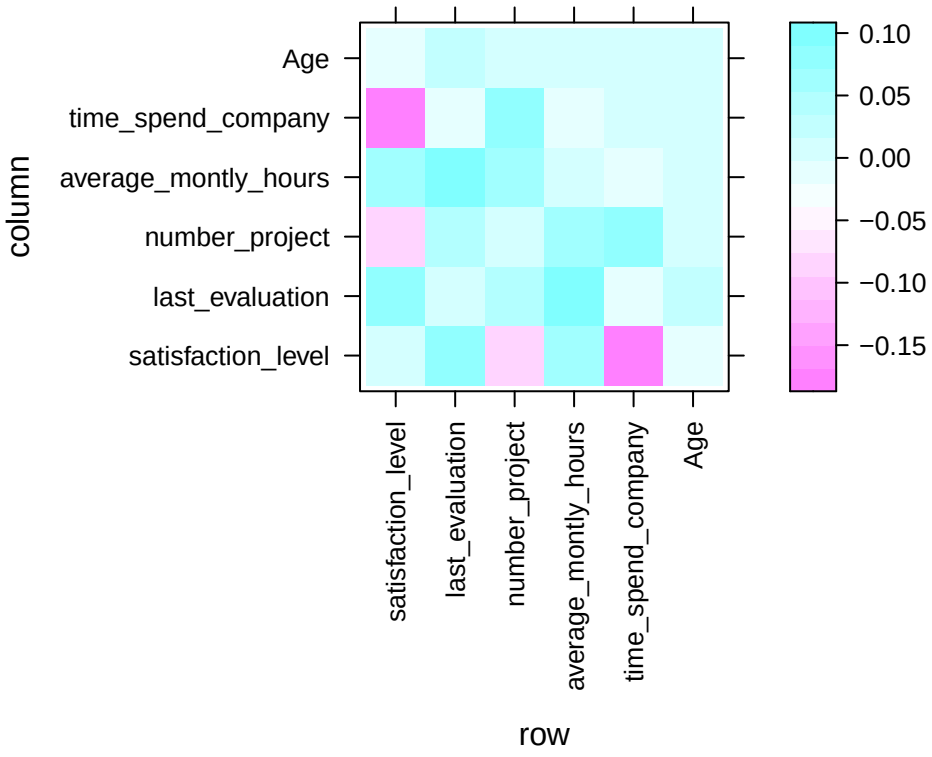
Una vez vista la distribución de las variables independentes con respecto a la dependiente, nos centraremos en ver las relaciones entre las variables independientes. Empezaremos con las continuas.

Creamos un nuevo dataframe solo con estas variables.

```
df_num.0 <- df[df$voluntary_turnover==0,unlist(lapply(df, function(x){
class(x)=="numeric"})) & !colnames(df)%in%c("voluntary_turnover")]
```

Creamos y visualizamos la matriz de correlación.

```
df_cor <- cor(df_num.0, method = c("pearson"))-diag(1,ncol(df_num.0))
levelplot(df_cor, scales=list(x=list(rot=90)))
```

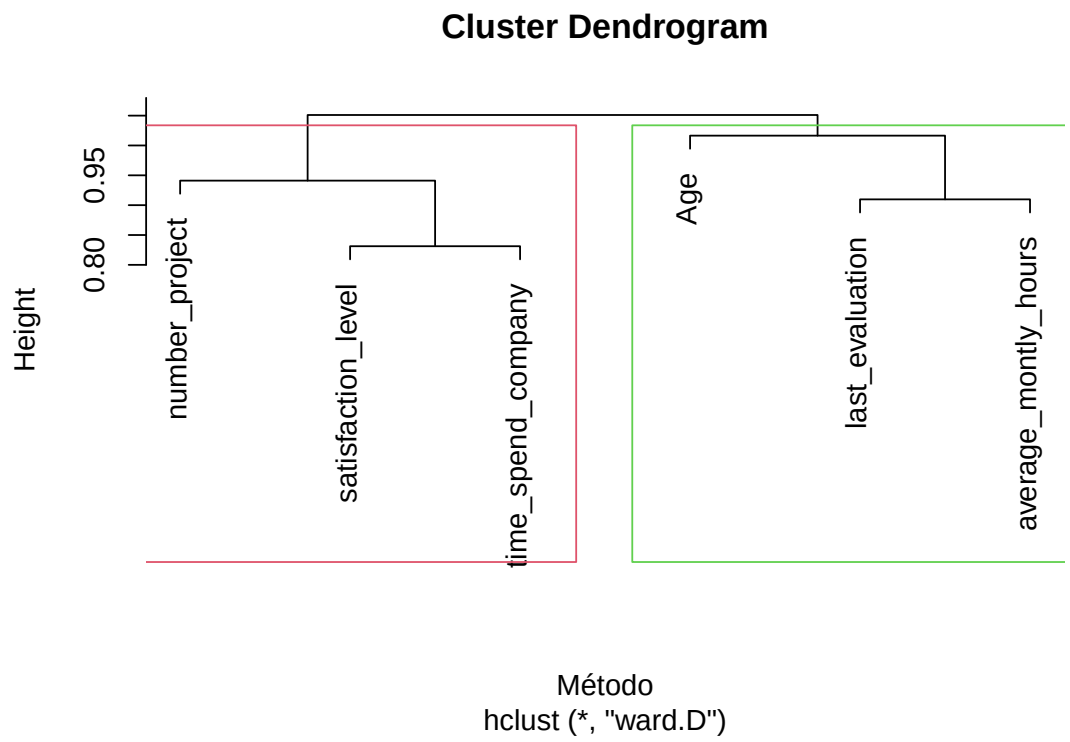


Veamos de este modo que las correlaciones que tenemos están en un rango de entre "-0.15" y "0.10", es decir, las correlaciones son débiles en general.

Realizaremos un clúster jerárquico con estas variables continuas para ver cómo de fuertes son sus relaciones, usaremos el método de Ward del que ya hemos hablado en el anterior apartado. Utilizaremos como distancia la resta de 1 menos el valor absoluta de la correlación entre estas variables, de forma que a mayor correlación entre variables, su distancia será menor.

Además, lo veremos gráficamente con un dendograma. Un dendograma es un árbol en el que el largo de las ramas está asociado inversamente a la fortaleza de la relación. Una vez visto el árbol, observaremos que existen dos grupos diferenciados y pintaremos una línea para separarlos.

```
cl_num <- hclust(as.dist(1-abs(df_cor)), method = "ward.D")
plot(cl_num, xlab = "Método")
nk <- 2
rect.hclust(cl_num, k = nk, border = 2:(2+nk-1))
# Clasificamos las variables en dos clústeres
clusterCut <- cutree(cl_num, nk)
```



Aquí podemos ver un primer grupo con los valores de *number_project, satisfaction_level y time_spend_company* y otro con *Age, last_evaluation y average_monthly_hours*. Nos interesa, para las variables *evaluation* y *average_monthly_hours*, ver sus relaciones con las variables categóricas. Para ello, utilizaremos un método supervisado de discretización para las variables cuantitativas (*last_evaluation* y *average_monthly_hours*) en base a la variable voluntary_turnover.

```
df.disc <- discretizeDF.supervised(voluntary_turnover ~ last_evaluation +
average_montly_hours+Age+satisfaction_level, data=df, method = "ameva",
dig.lab = 2)
```

```
df <- discretizeDF(df, methods = df.disc)
```

Ahora haremos algo similar con las variables categóricas, donde estarán estas dos últimas variables discretizadas. Primero, creamos un dataframe con estas. Al ser variables categóricas, no podemos usar la matriz de correlación, sino que comprobaremos si estas variables siguen las mismas distribuciones mediante una prueba "chi cuadrado". A continuación, se muestra la matriz con los p-valores de tal test.

```
df_cat.0 <- df[df$voluntary_turnover==0,unlist(lapply(df, function(x){
class(x)=="factor"})) & !colnames(df)%in%c("voluntary_turnover")]
round(as.dist(unlist(apply(df_cat.0, 2, function(x){
unlist(apply(df_cat.0, 2, function(y){
chisq.test(table(x,y))$p.value
  }))
}))), digits=4)
```

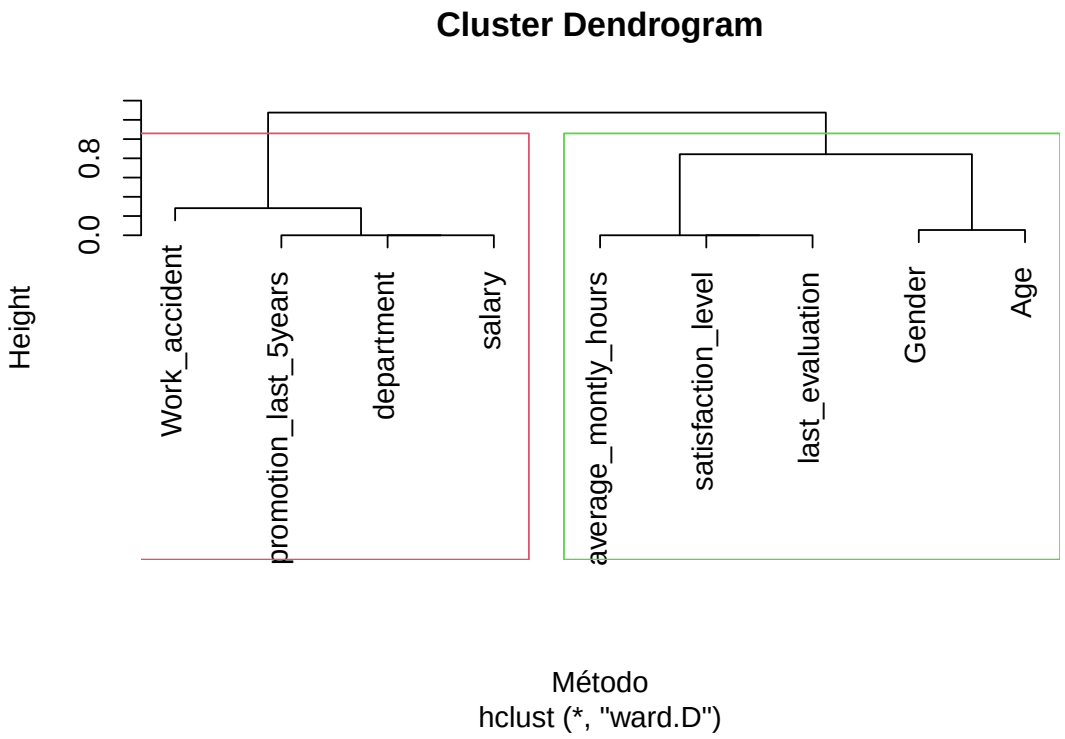
```
##                     satisfaction_level last_evaluation average_montly_hours
## last_evaluation                 0.0000
## average_montly_hours            0.0000          0.0000
## Work_accident                   0.9315          0.4588                0.6998
## promotion_last_5years           0.3951          0.9894                0.0750
## department                      0.3150          0.0173                0.4595
## salary                          0.0350          0.6088                0.1390
## Gender                          0.7179          0.2809                0.3749
## Age                             0.3493          0.3721                0.0937
##                       Work_accident promotion_last_5years department salary
## last_evaluation
## average_montly_hours
## Work_accident
## promotion_last_5years        0.0034
## department                   0.4754                0.0000
## salary                       0.0849                0.0000     0.0000
## Gender                       0.0636                0.5207     0.1539 0.5262
## Age                          0.7051                0.1509     0.6243 0.3706
##                       Gender
```

```
## last_evaluation
## average_montly_hours
## Work_accident
## promotion_last_5years
## department
## salary
## Gender
## Age                  0.0546
```

Vemos distribuciones similares para las variables *satisfaction_level* con *last_evaluation*, *average_montly_hours* y *salary*, de *last_evaluation* con *departament*, de *Work_accident* con *promotion_last_5years*, de *promotion_last_5years* con *department* y *salary*, de *department* con *salary* y de *Gender* con *Age*. En resumen, existen muchas similitudes entre estas variables, de modo que para verlas de una forma más gráfica y entendible, visualizaremos esto en un dendograma y los dividiremos en dos clústeres. Usaremos el método de *Ward* también.

```
cl_cat <- hclust(as.dist(unlist(apply(df_cat.0, 2, function(x){
unlist(apply(df_cat.0, 2, function(y){
chisq.test(table(x,y))$p.value
  }))
}))), method = "ward.D")
```

```
# barplot(cl_cat$height-min(cl_cat$height))
plot(cl_cat,  xlab = "Método")
nk <- 2
rect.hclust(cl_cat, k = nk, border = 2:(2+nk-1))
```




```
clusterCut <- cutree(cl_cat, nk)
```

Observamos un primer grupo con los valores de *work_ accident, promotion_ last_ 5years, department* y *salary* y otro con *Age, Gender y average_ monthly_ hours, satisfaction_ level y last_ evaluation.*

4.2.5. Partición del dataset en conjuntos de entrenamiento y validación

Una vez realizado el *EDA*, podemos empezar a lanzar las diferentes predicciones y ver sus resultados. Pero primero deberemos particionar los datos. Dividiremos el dataset en dos partes, el conjunto de entrenamiento, en el que "estudiaremos" los datos y obtendremos un modelo, y el conjunto de validación, donde se comprobará cómo de bueno es ese modelo.

Usaremos el 90 % de los datos como conjunto de entrenamiento y el 10 % restante como conjunto de validación.

Una vez seleccionado el modelo que más nos interesa, lo ejecutaremos con otro dataset independiente, llamado test, el cual no tenemos valores la variable objetivo y queremos obtener su valor. Con ese dataset realizaremos la visión de los datos en el último apartado.

```
set.seed(123456)
idx <- sample.int(n = nrow(df), size = floor(0.1*nrow(df)), replace = FALSE)
df.va <- df[idx,]
df.tr <- df[-idx,]
```

4.2.6. Predicciones con diferentes modelos

Llegamos a la parte más interesante del estudio. Después de observar las relaciones de las variables independientes toca centrarse en la dependiente. Este apartado se centrará en probar diferentes modelos y comprobar sus métricas. Las métricas ya han sido definidas en el anterior apartado, trabajaremos con: *accuracy*, *precission*, *sensibility*, *F1-Score*, Curva *ROC* y *AUC*.

Estas son las medidas que usaremos para constatar la validez del modelo usado, por cada predicción mostraremos estos indicadores y los guardaremos en una tabla que mostraremos en el último apartado de esta sección y discutiremos cuál es el mejor modelo. Destacar que cuando nos referíamos a casos/valores positivos, nos referimos a las fugas de empleados, que es el motivo que nos preocupa, detectar cuales son los que tiene más posibilidades de irse. Por ello la mayoría de los indicadores se centran en tales valores. Las métricas serán lanzadas y guardadas para los conjuntos de entrenamiento y validación, el motivo de guardar para las del conjunto de validación es, entre otras cosas, para comprobar que el modelo no hace sobreajuste o *overfitting*.

Muchas de estas métricas se crean de la misma forma en *R*, por lo tanto, para no repetir código, crearemos una función para modularizar estos procesos. En este caso, esta función pintará la curva *ROC* en el conjunto de validación y en el de entrenamiento y sacará el valor de *AUC* correspondiente.

Pese a que se mostrarán más métricas de las definidas (que forman parte del libro de R que utilizaremos), nos centraremos únicamente en las anteriores comentadas. En especial, con las relacionadas a la correcta clasificación de casos positivos (la clase minoritaria, es decir, fugas de empleado) ya que lo que buscamos en nuestro modelo no es la precisión general del modelo, sino la buena precisión de la fuga de empleados. Por lo que, pese a haberla definido, la métrica *Accuracy*, pese a que es muy importante y útil ya que es el indicador general que explica la calidad de un clasificar, pasará a un segundo plano.

```
roc<-function(pred.tr.roc,pred.va.roc,metodo){
pred.tr.roc <- prediction(pred.tr.roc, df.tr$voluntary_turnover)
perf.tr <- performance(pred.tr.roc,"tpr","fpr")
plot(perf.tr, colorize=TRUE, main=paste("ROC turnover with ", metodo))
auc_ROCR_train <- performance(pred.tr.roc, measure = "auc")
auc_ROCR_train <- auc_ROCR_train@y.values[[1]]
pred.va.roc <- prediction(pred.va.roc, df.va$voluntary_turnover)
perf.va <- performance(pred.va.roc,"tpr","fpr")
lines(perf.va@x.values[[1]], perf.va@y.values[[1]], lty=2, col=1)
auc_ROCR_val <- performance(pred.va.roc, measure = "auc")
auc_ROCR_val <- auc_ROCR_val@y.values[[1]]
return(list(auc_ROCR_train,auc_ROCR_val))
}
```

Ahora sólo nos queda lanzar los diferentes modelos y comprobar las métricas comentadas. Los modelos usados pueden ser de diversas formas, desde un simple algoritmo hasta una técnica de muestreo, al inicio de cada modelo se hará una breve explicación.

ARBOL DE DECISION SIMPLE

Empezaremos con un árbol de decisión, pondremos como restricciones que la profundidad máxima del arbol sea de máximo 3 y que para hacer una corte tenga al menos 2 valores diferentes, además de pedirle que para cualquier corte que haga siempre se mejore al algoritmo. Buscamos árboles simples para ver que variables intervienen (cuántas variables diferentes intervienen) y así, prevenir el sobreajuste.

```
tree <- rpart(voluntary_turnover ~ ., method="class", data = df.tr, model =
TRUE, parms = list(split = "information"),
control = rpart.control(minsplit = 2, xval = 10, cp = 0, usesurrogate = 2,
maxdepth = 3))
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = voluntary_turnover ~ ., data = df.tr, method = "class",
##     model = TRUE, parms = list(split = "information"), control =
```

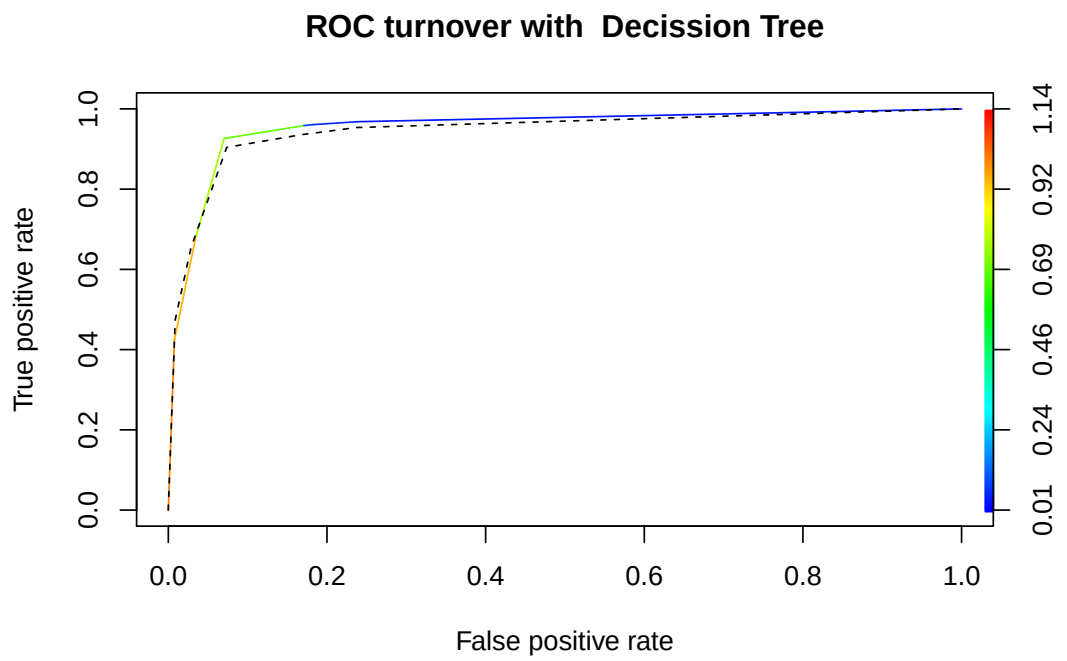
```
 rpart.control(minsplit = 2,
##         xval = 10, cp = 0, usesurrogate = 2, maxdepth = 3))
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      number_project
## [4] satisfaction_level   time_spend_company
##
## Root node error: 3227/13500 = 0.23904
##
## n= 13500
##
##          CP nsplit rel error  xerror      xstd
## 1 0.2482182      0   1.00000 1.00000 0.0153561
## 2 0.1439417      1   0.75178 0.75178 0.0138239
## 3 0.0663155      3   0.46390 0.48621 0.0115394
## 4 0.0312984      5   0.33127 0.34056 0.0098460
## 5 0.0021692      6   0.29997 0.31639 0.0095200
## 6 0.0000000      7   0.29780 0.31608 0.0095157
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.cm<-predict(tree,newdata = df.tr, type="class")
cm_t<-confusionMatrix(pred.tr.cm,df.tr$voluntary_turnover,positive='1')
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9550  238
##          1  723 2989
##
##                Accuracy : 0.9288
##                  95% CI : (0.9243, 0.9331)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.8139
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9262
##             Specificity : 0.9296
##          Pos Pred Value : 0.8052
##          Neg Pred Value : 0.9757
##              Prevalence : 0.2390
##          Detection Rate : 0.2214
##    Detection Prevalence : 0.2750
##       Balanced Accuracy : 0.9279
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.cm<-predict(tree,newdata = df.va, type="class")
cm_v<-confusionMatrix(pred.va.cm,df.va$voluntary_turnover,positive='1')

#Curva ROC
# Dataset de Entrenamiento
pred.tr.roc<-predict(tree,newdata = df.tr, type="prob")[,2]
# Dataset de Validación
pred.va.roc<-predict(tree,newdata = df.va, type="prob")[,2]
AUC_dt<-roc(pred.tr.roc,pred.va.roc,"Decission Tree")
```




```
# Creamos un dataframe para guardar los resultados
metodo<-list(); sensitivity_train<-list();Precision_train<-list();
F_Score_train<-list();AUC_train<-list();sensitivity_test<-list();
Precision_test<-list();F_Score_test<-list();AUC_test<-list()
df_result <- data.frame(metodo,sensitivity_train, Precision_train,
F_Score_train, AUC_train, sensitivity_test, Precision_test, F_Score_test,
AUC_test)

df_result<-rbind(data.frame(Metodo="Decission Tree",Sensitivity_train=
cm_t$byClass[1],Precission_train=cm_t$byClass[5],F_Score_train=
cm_t$byClass[7],AUC_train=as.numeric(AUC_dt[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_dt[2])), df_result,
make.row.names=F)
```

Como hemos dicho, utilizamos un árbol de decisión bastante sencillo, con pocos cortes y poca profundidad, lo que provoca pocas divisiones en el los datos para clasificar los datos. Pese ha ello, se observan buenas métricas, ya que obtenemos más de un 90 % de

accuracy y de precisión, mientras que la sensibilidad obtiene un buen resultado pero algo más bajo. Por otro lado se puede decir que no tenemos sobreajuste, ya que los métricas para el conjunto de validación y de entrenamiento son bastante similares aunque, como es lógico, ligeramente mejores en el conjunto de entrenamiento.

INCREMENTO DE LA CLASE MINORITARIA

Ahora realizaremos una técnica de *oversampling*, remplazaremos mediante selección aleatoria con reemplazo de registros de la clase minoritaria. Un vez balanceadas las clases, haremos la predicción usando un árbol de decisión con los mismos criterios en el apartado anterior. Es decir, probamos si el árbol de decisión con un método de remuestreo previos, mejora los resultados.

```
# Divido 1/0:
df.tr.1 <- df.tr[df.tr$voluntary_turnover==1,]
df.tr.0 <- df.tr[df.tr$voluntary_turnover==0,]
# Miramos el número de registros en cada conjunto
num.1<-as.numeric(nrow(df.tr.1))
num.0<-nrow(df.tr.0)
# Duplico los 1s, seleccionaremos aleatoriamente num.1-num.0 del conjunto
df.tr.1, esta técnica es conocida como RandomOversampling:
id.1.over <- sample.int(n = num.1, size = num.0-num.1, replace = TRUE)
df.tr.bal.ov <- rbind(df.tr.1,df.tr.1[id.1.over,], df.tr.0)

# Comprobamos si tienen el mismo número de casos de churn y no churn
# Train
nrow(df.tr.bal.ov[df.tr.bal.ov$voluntary_turnover==1,])
```

```
## [1] 10273
```

```
# Test
nrow(df.tr.bal.ov[df.tr.bal.ov$voluntary_turnover==0,])
```

```
## [1] 10273
```

```
# Modelo
tree_ov <- rpart(voluntary_turnover ~ ., method="class", data = df.tr.bal.ov
,model = TRUE, parms = list(split = "information"),
control = rpart.control(minsplit = 2, xval = 10, cp = 0, usesurrogate = 2,
maxdepth = 3))
printcp(tree_ov)
```

```
##
## Classification tree:
## rpart(formula = voluntary_turnover ~ ., data = df.tr.bal.ov,
##     method = "class", model = TRUE, parms = list(split = "information"),
##     control = rpart.control(minsplit = 2, xval = 10, cp = 0,
##         usesurrogate = 2, maxdepth = 3))
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      number_project
## [4] satisfaction_level   time_spend_company
##
## Root node error: 10273/20546 = 0.5
##
## n= 20546
##
##          CP nsplit rel error  xerror      xstd
## 1 0.5699406      0   1.00000 1.01888 0.0069752
## 2 0.1563321      1   0.43006 0.43006 0.0057325
## 3 0.0503261      2   0.27373 0.27373 0.0047957
## 4 0.0391317      3   0.22340 0.22340 0.0043952
## 5 0.0241410      4   0.18427 0.18748 0.0040668
## 6 0.0018495      5   0.16013 0.16032 0.0037888
## 7 0.0000000      6   0.15828 0.15916 0.0037762
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.bal.ov.cm<-predict(tree_ov,newdata = df.tr, type="class")
cm_t<-confusionMatrix(pred.tr.bal.ov.cm,df.tr$voluntary_turnover,
```

```
positive='1')
# Train
```

```
cm_t
```

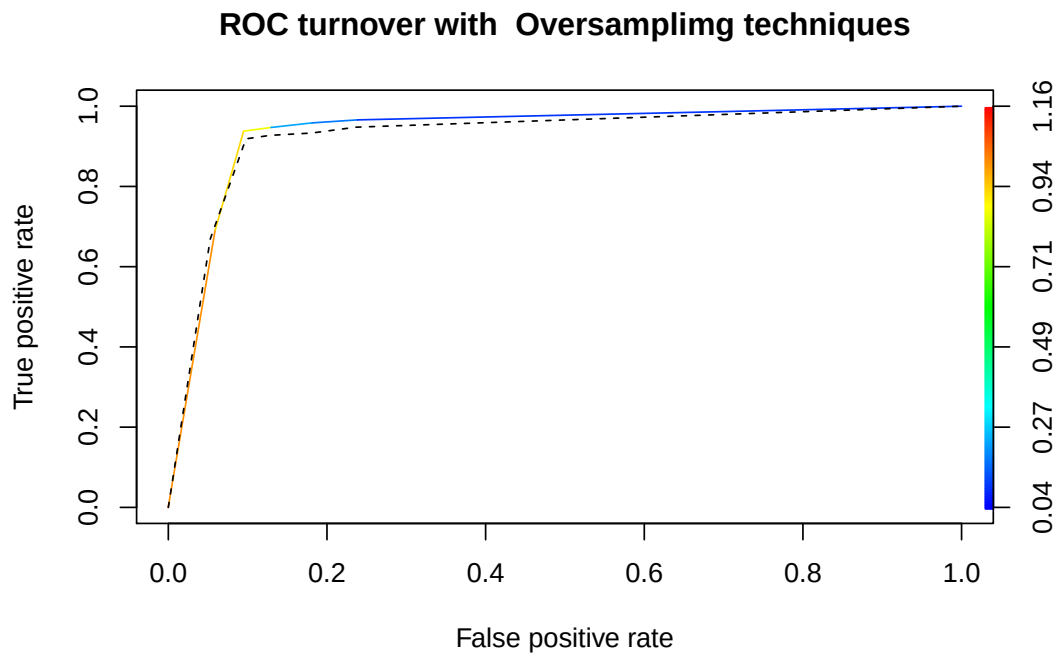
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9298  201
##          1  975 3026
##
##                Accuracy : 0.9129
##                  95% CI : (0.908, 0.9176)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7787
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9377
##             Specificity : 0.9051
##          Pos Pred Value : 0.7563
##          Neg Pred Value : 0.9788
##              Prevalence : 0.2390
##          Detection Rate : 0.2241
##    Detection Prevalence : 0.2964
##       Balanced Accuracy : 0.9214
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.bal.ov.cm<-predict(tree_ov,newdata = df.va, type="class")
cm_v<-confusionMatrix(pred.va.bal.ov.cm,df.va$voluntary_turnover,
positive='1')
# Valid
```

```
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1042   28
##          1  113  316
##
##                Accuracy : 0.9059
##                  95% CI : (0.89, 0.9202)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7553
##
##  Mcnemar's Test P-Value : 1.504e-12
##
##             Sensitivity : 0.9186
##             Specificity : 0.9022
##          Pos Pred Value : 0.7366
##          Neg Pred Value : 0.9738
##              Prevalence : 0.2295
##          Detection Rate : 0.2108
##    Detection Prevalence : 0.2862
##       Balanced Accuracy : 0.9104
##
##        'Positive' Class : 1
##
```

```
#Curva ROC
pred.tr.bal.ov.roc<-predict(tree_ov,newdata = df.tr, type="prob")[,2]
pred.va.bal.ov.roc<-predict(tree_ov,newdata = df.va, type="prob")[,2]
AUC_bal.ov<-roc(pred.tr.bal.ov.roc,pred.va.bal.ov.roc,
"Oversamplimg techniques")
```




```
df_result<-rbind(df_result, data.frame(Metodo="Oversamplimg techniques",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_bal.ov[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_bal.ov[2])),
make.row.names=F)
```

Seguimos observando buenos resultados, aunque el hecho de "adulterar" la base de datos con la que se entrena el modelo, provoca que se iguale el número de predicciones para cada clase, ya que al entrenar el modelo, aparecen los mismos datos casos para clase. Esto provoca que las métricas que miden la calidad de las precisiones de la clase minoritaria mejoren, aunque no sustancialmente, mientras que la de la clase mayoritaria disminuyan.

Esto es uno de los problemas que tiene el utilizar este método, aumentar el riesgo de eliminar elementos de la muestra potencialmente importantes en el proceso de clasificación.

REDUCCIÓN DE LA CLASE MAYORITARIA

Este caso es análogo al anterior pero usando la técnica de balanceo *undersampling* o submuestreo. Para ello, haremos una selección aleatoria de entre los registros de la clase mayoritaria. Después volveremos a lanzar un árbol de decisión con las mismas condiciones que en los anteriores apartados.

```
id.1.under <- sample.int(n = num.0, size = num.1, replace = FALSE)
df.tr.bal.un <- rbind(df.tr.1,df.tr.0[id.1.under,])

# Comprobamos que el dataset se ha creado proporcionadamente
# Train

nrow(df.tr.bal.un[df.tr.bal.un$voluntary_turnover==1,])
```

```
## [1] 3227
```

```
# Valid
nrow(df.tr.bal.un[df.tr.bal.un$voluntary_turnover==0,])
```

```
## [1] 3227
```

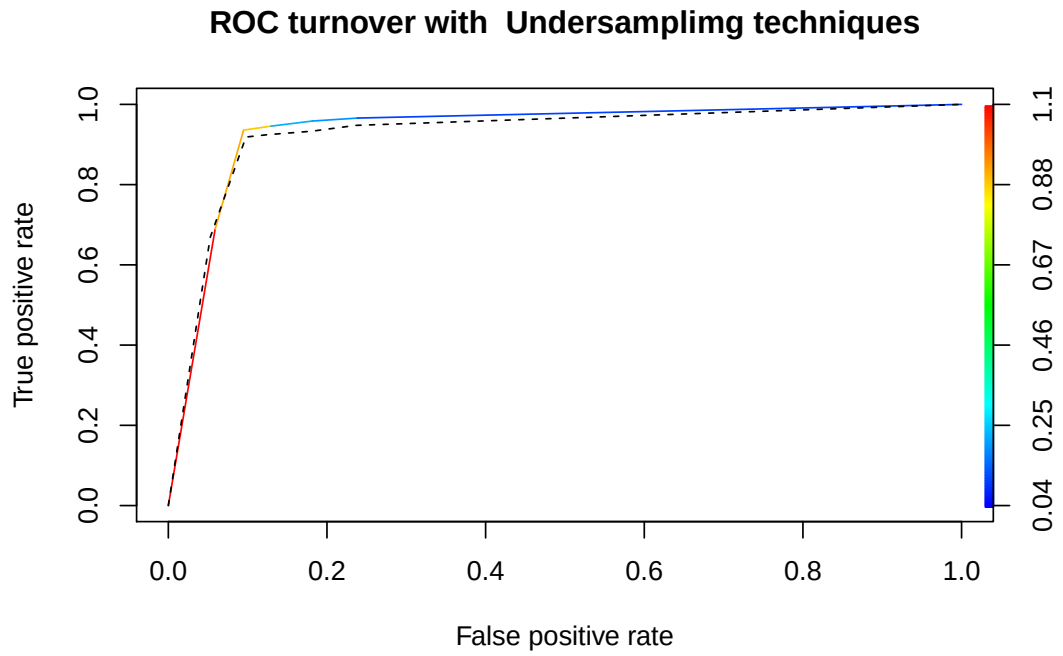
```
# Modelo
tree_un <- rpart(voluntary_turnover ~ ., method="class", data = df.tr.bal.un
,model = TRUE, parms = list(split = "information"),
control = rpart.control(minsplit = 2, xval = 10, cp = 0, usesurrogate = 2,
maxdepth = 3))
printcp(tree_un)
```

```
##
## Classification tree:
## rpart(formula = voluntary_turnover ~ ., data = df.tr.bal.un,
##     method = "class", model = TRUE, parms = list(split = "information"),
```

```
##     control = rpart.control(minsplit = 2, xval = 10, cp = 0,
##         usesurrogate = 2, maxdepth = 3))
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      satisfaction_level
## [4] time_spend_company
##
## Root node error: 3227/6454 = 0.5
##
## n= 6454
##
##         CP nsplit rel error  xerror      xstd
## 1 0.565851      0   1.00000 1.02138 0.0124448
## 2 0.163619      1   0.43415 0.43415 0.0102631
## 3 0.052061      2   0.27053 0.27053 0.0085143
## 4 0.038736      3   0.21847 0.21847 0.0077656
## 5 0.024171      4   0.17973 0.18593 0.0072292
## 6 0.000000      5   0.15556 0.15463 0.0066493
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.bal.un.cm<-predict(tree_un,newdata = df.tr, type="class")

cm_t<-confusionMatrix(pred.tr.bal.un.cm,df.tr$voluntary_turnover,
positive='1')

#Pred sobre validation para mostrar matriz de conf
pred.va.bal.un.cm<-predict(tree_un,newdata = df.va, type="class")

cm_v<-confusionMatrix(pred.va.bal.un.cm,df.va$voluntary_turnover,
positive='1')

#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.bal.un.roc<-predict(tree_un,newdata = df.tr, type="prob")[,2]
pred.va.bal.un.roc<-predict(tree_un,newdata = df.va, type="prob")[,2]
```

```
AUC_bal.un<-roc(pred.tr.bal.un.roc,pred.va.bal.un.roc,
"Undersamplimg techniques")
```




```
df_result<-rbind(df_result, data.frame(Metodo="Undersamplimg techniques",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_bal.un[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_bal.un[2])),
make.row.names=F)
```

Se pueden observar que las resultados obtenidos son similares a si usaramos el método de *oversampling* por lo que podríamos sacar conclusiones análogas, es decir, buenas métricas en general pero se penaliza en exceso la clase mayoritaria. Destacar que tanto este caso como en el anterior las métricas para el conjunto de test son exactamente las mismas. Pos lo que el modelo es muy similar, por no decir que es el mismo.

COMBINACIÓN REDUCCIÓN CLASE MAYORITARIA Y AUMENTO CLASE MINORITARIA. TÉCNICA 2-50

En este caso, usaremos la técnica de balanceo llamado *resampling*, que es una mezcla de las dos anteriores (*undersampling* y *oversampling*). También usaremos métodos de selección aleatoria de datos para tales procesos.

```
# Duplicamos los churn:
df.tr.1.dup <- rbind(df.tr.1,df.tr.1)
# Reducimos a la mitad los no churn:
id.0.half <- sample.int(n = num.0, size = num.0/2, replace = FALSE)

# Unimos conjuntos
df.tr.bal.re<-rbind(df.tr.1.dup,df.tr.0[id.0.half,])

# Comprobamos
# Test
nrow(df.tr.bal.re[df.tr.bal.re$voluntary_turnover==1,])
```

```
## [1] 6454
```

```
# Valid
nrow(df.tr.bal.re[df.tr.bal.re$voluntary_turnover==0,])
```

```
## [1] 5136
```

```
# Modelo
tree_re <- rpart(voluntary_turnover ~ ., method="class", data = df.tr.bal.re
,model = TRUE, parms = list(split = "information"),
control = rpart.control(minsplit = 2, xval = 10, cp = 0, usesurrogate = 2,
maxdepth = 3))
printcp(tree_re)
```

```
##
## Classification tree:
## rpart(formula = voluntary_turnover ~ ., data = df.tr.bal.re,
```

```
##     method = "class", model = TRUE, parms = list(split = "information"),
##     control = rpart.control(minsplit = 2, xval = 10, cp = 0,
##         usesurrogate = 2, maxdepth = 3))
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      satisfaction_level
## [4] time_spend_company
##
## Root node error: 5136/11590 = 0.44314
##
## n= 11590
##
##         CP nsplit rel error  xerror      xstd
## 1 0.495327      0   1.00000 1.00000 0.0104126
## 2 0.224688      1   0.50467 0.50467 0.0087342
## 3 0.046924      2   0.27998 0.27998 0.0069102
## 4 0.033879      3   0.23306 0.23306 0.0063790
## 5 0.022975      4   0.19918 0.20230 0.0059881
## 6 0.000000      5   0.17621 0.17601 0.0056211
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.bal.re.cm<-predict(tree_re,newdata = df.tr, type="class")
cm_t<-confusionMatrix(pred.tr.bal.re.cm,df.tr$voluntary_turnover,
positive='1')
# Train:
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9298  207
##          1  975 3020
##
##                Accuracy : 0.9124
```

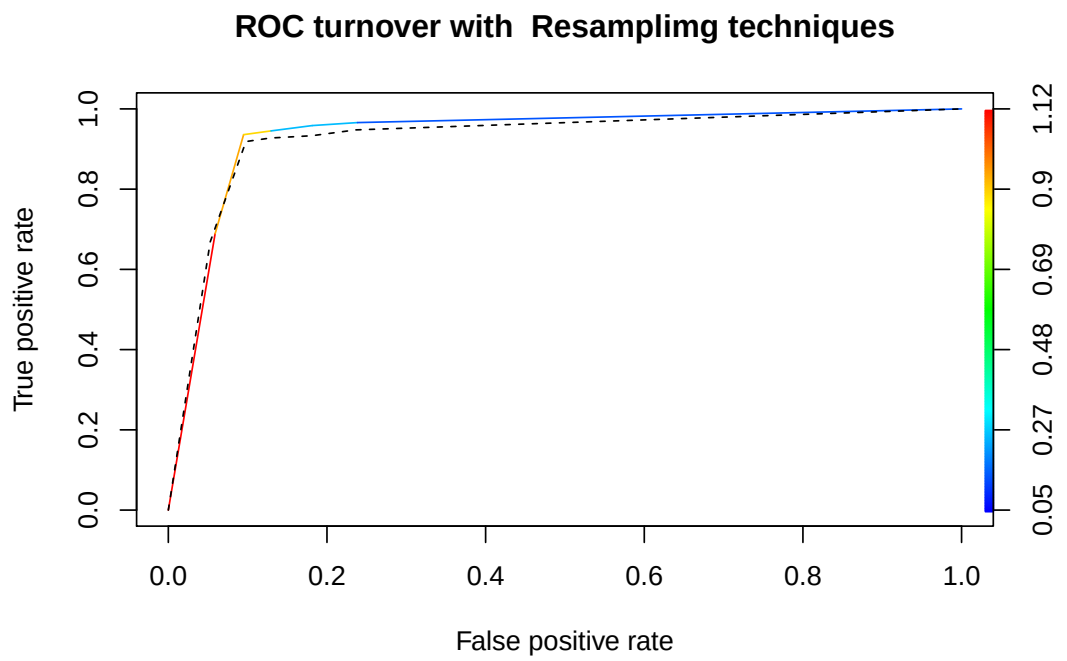
```
##                  95% CI : (0.9076, 0.9172)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7775
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9359
##             Specificity : 0.9051
##          Pos Pred Value : 0.7559
##          Neg Pred Value : 0.9782
##              Prevalence : 0.2390
##          Detection Rate : 0.2237
##    Detection Prevalence : 0.2959
##       Balanced Accuracy : 0.9205
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.bal.re.cm<-predict(tree_re,newdata = df.va, type="class")
cm_v<-confusionMatrix(pred.va.bal.re.cm,df.va$voluntary_turnover,
positive='1')
# Valid
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1042   28
##          1  113  316
##
##                Accuracy : 0.9059
```

```
##                  95% CI : (0.89, 0.9202)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7553
##
##  Mcnemar's Test P-Value : 1.504e-12
##
##             Sensitivity : 0.9186
##             Specificity : 0.9022
##          Pos Pred Value : 0.7366
##          Neg Pred Value : 0.9738
##              Prevalence : 0.2295
##          Detection Rate : 0.2108
##    Detection Prevalence : 0.2862
##       Balanced Accuracy : 0.9104
##
##        'Positive' Class : 1
##
```

```
#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.bal.re.roc<-predict(tree_re,newdata = df.tr, type="prob")[,2]

pred.va.bal.re.roc<-predict(tree_re,newdata = df.va, type="prob")[,2]
AUC_bal.re<-roc(pred.tr.bal.re.roc,pred.va.bal.re.roc,
"Resamplimg techniques")
```




```
df_result<-rbind(df_result, data.frame(Metodo=
"Resamplimg techniques",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_bal.re[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_bal.re[2])),
make.row.names=F)
```

Al ser este método un composición de los métodos de submuestreo y sobremuestro, es bastante probable, como ha sucedido, que sus virtudes y defectos se hereden. Las métricas siguen siendo buenas, muy similares a los dos otros métodos de muestreo que se comentan, pero se sigue penalizando la clase mayoritaria, lo que deriva en un gran incremento de las probabilidades de fuga finales. Por lo que no obtenemos una substancial mejora de los resultados mientras que aumentamos el tiempo de procesamiento de los datos.

BOOSTING

Recordemos que los algoritmos de *boosting* son algoritmos multiclasificadores que se basan en la construcción de sucesivos clasificadores sobre modificaciones de la muestra de

entrenamiento, estas modificaciones son realizadas en función de los errores cometidos por el clasificador en cada una de las iteraciones y, posteriormente se combinan los clasificadores para obtener el clasificador final.

Nosotros usaremos uno de los algoritmos más utilizados, el *Adaboost*, que, como ya hemos dicho, dirige su atención en aquellos casos que son más difíciles de clasificar. En primer lugar, *Adaboost* les da un peso a los datos de entrenamiento, luego realiza la clasificación y en base a los fallos obtenidos, cambia el peso de estos datos. Vuelve a repetir este proceso tantas veces como se le indique. En nuestro caso, solo realizará dos iteraciones.

```
# Modelo:
adaboost <- boosting(voluntary_turnover ~ ., data=df.tr, boos=TRUE,
mfinal=2) # Dos iteraciones
summary(adaboost)
```

```
##            Length Class   Mode
## formula         3  formula call
## trees           2  -none-  list
## weights         2  -none-  numeric
## votes       27000  -none-  numeric
## prob        27000  -none-  numeric
## class       13500  -none-  character
## importance     11  -none-  numeric
## terms           3  terms   call
## call            5  -none-  call
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.boost.cm<-predict(adaboost,newdata = df.tr, type="class")
cm_t<-confusionMatrix(table(pred.tr.boost.cm$class,df.tr$voluntary_turnover)
,positive='1')

# Train
cm_t
```

```
## Confusion Matrix and Statistics
##
```

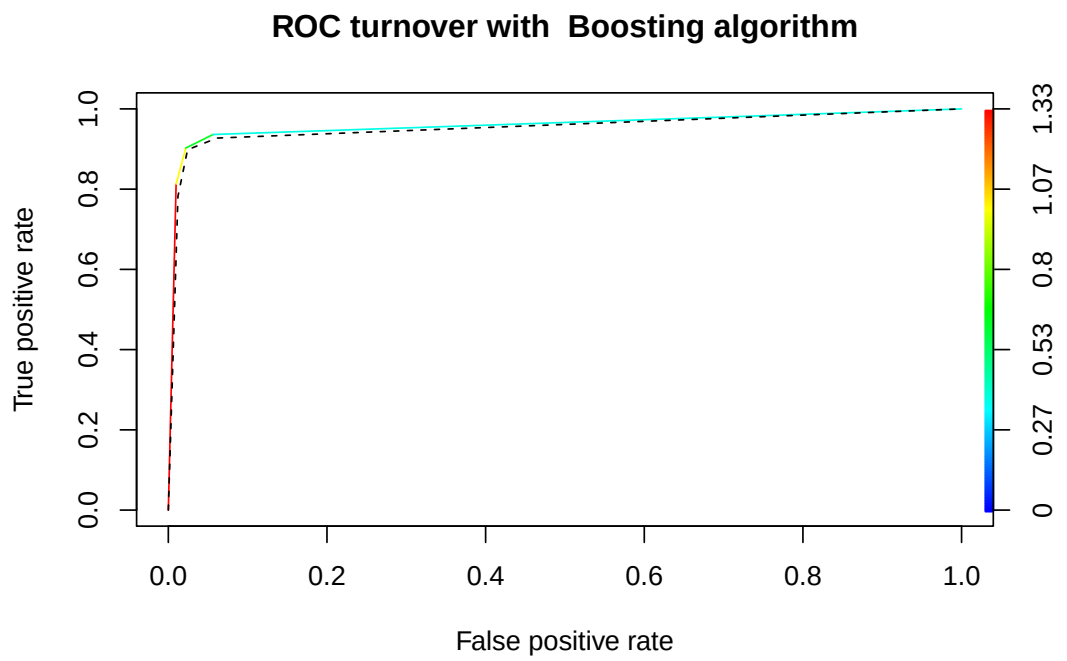
```
##
##          0     1
##    0 10047   315
##    1   226  2912
##
##                Accuracy : 0.9599
##                  95% CI : (0.9565, 0.9632)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8888
##
##  Mcnemar's Test P-Value : 0.0001547
##
##             Sensitivity : 0.9024
##             Specificity : 0.9780
##          Pos Pred Value : 0.9280
##          Neg Pred Value : 0.9696
##              Prevalence : 0.2390
##          Detection Rate : 0.2157
##    Detection Prevalence : 0.2324
##       Balanced Accuracy : 0.9402
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.boost.cm<-predict(adaboost,newdata = df.va, type="class")
cm_v<-confusionMatrix(table(pred.va.boost.cm$class,df.va$voluntary_turnover)
,positive='1')

# Valid
cm_v
```

```
## Confusion Matrix and Statistics
```

```
##
##
##        0    1
##   0 1127   35
##   1   28  309
##
##                Accuracy : 0.958
##                  95% CI : (0.9465, 0.9676)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8803
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.8983
##             Specificity : 0.9758
##          Pos Pred Value : 0.9169
##          Neg Pred Value : 0.9699
##              Prevalence : 0.2295
##          Detection Rate : 0.2061
##    Detection Prevalence : 0.2248
##       Balanced Accuracy : 0.9370
##
##        'Positive' Class : 1
##
```

```r
#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.boost.roc<-predict(adaboost,newdata = df.tr, type="prob")$prob[,2]

# Ahora la calculamos para el conjunto de validación
pred.va.boost.roc<-predict(adaboost,newdata = df.va, type="prob")$prob[,2]

AUC_boost<-roc(pred.tr.boost.roc,pred.va.boost.roc,"Boosting algorithm")
```




```
df_result<-rbind(df_result, data.frame(Metodo="Boosting",Sensitivity_train=
cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_boost[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_boost[2])),
make.row.names=F)
```

Este algoritmo es, hoy en día, la base de uno de los algoritmos más sólidos y usados en las predicciones en el mundo de los negocios (el algoritmo *XGboost*) ya que asegura buenos resultados. Podemos ver una sustancial mejora de la precisión a costa de un ligero empeoramiento de la sensibilidad, lo que demuestra con un *F1-Score* más alto que en los casos anteriores. Esto se podría traducir a que suele acertar en los verdaderos casos de fuga sin seleccionar erróneamente casos de no fuga como fugados, esto es realmente útil en las empresas a la hora de reducir recursos y costes para evitar este problema.

BAGGING

Otro tipo de algoritmo multiclasificador son los de *bagging*. Estos se generan a partir de la muestra inicial varios subconjuntos del mismo tamaño con reemplazamiento (ase-

gurando así la diversidad) y de forma independiente. A continuación, para cada uno de ellos, construye un subclasificador y, finalmente, utilizando el voto mayoritario, obtiene la clasificación final para cada individuo. Le hemos puesto la restricción de que solo use dos árboles, es decir que solo tendrá dos clasificadores.

```
bag <- bagging(voluntary_turnover ~ .,data=df.tr,mfinal=2) #2 arboles
summary(bag)
```

```
##            Length Class   Mode
## formula         3  formula call
## trees           2  -none-  list
## votes       27000  -none-  numeric
## prob        27000  -none-  numeric
## class       13500  -none-  character
## samples     27000  -none-  numeric
## importance     11  -none-  numeric
## terms           3  terms   call
## call            4  -none-  call
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.bag.cm<-predict(bag,newdata = df.tr, type="class")
cm_t<-confusionMatrix(table(pred.tr.bag.cm$class,df.tr$voluntary_turnover),
positive='1')

#Train
cm_t
```

```
## Confusion Matrix and Statistics
##
##
##          0     1
##   0 10066   317
##   1   207  2910
##
##                Accuracy : 0.9612
```

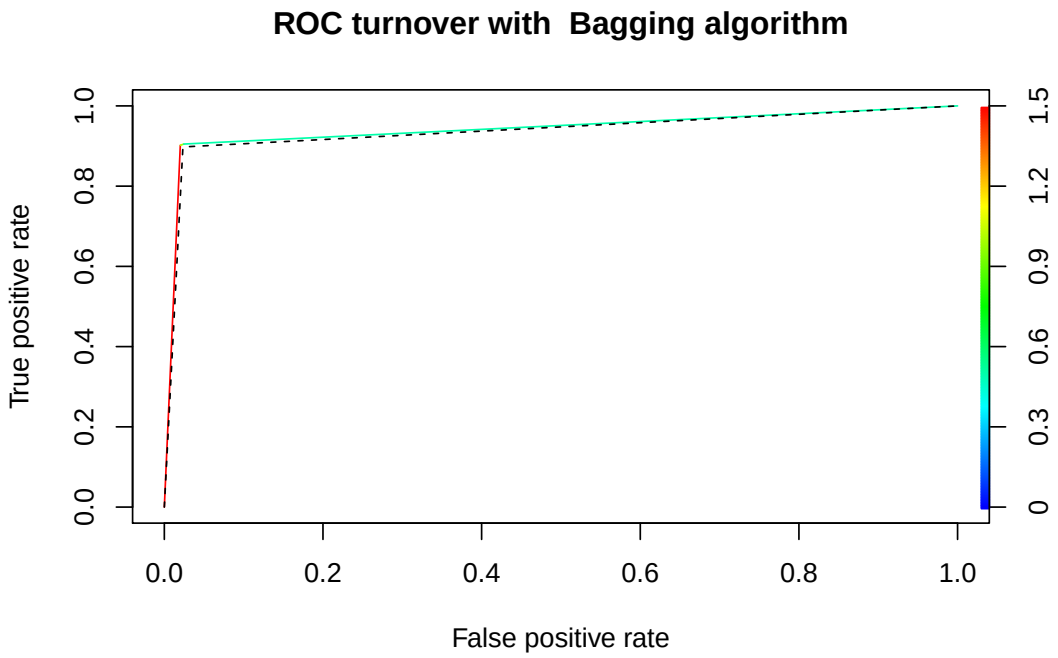
```
##                  95% CI : (0.9578, 0.9644)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.892
##
##  Mcnemar's Test P-Value : 1.92e-06
##
##             Sensitivity : 0.9018
##             Specificity : 0.9799
##          Pos Pred Value : 0.9336
##          Neg Pred Value : 0.9695
##              Prevalence : 0.2390
##          Detection Rate : 0.2156
##    Detection Prevalence : 0.2309
##       Balanced Accuracy : 0.9408
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.bag.cm<-predict(bag,newdata = df.va, type="class")
cm_v<-confusionMatrix(table(pred.va.bag.cm$class,df.va$voluntary_turnover),
positive='1')

#Valid
cm_v
```

```
## Confusion Matrix and Statistics
##
##
##        0    1
##   0 1128   36
##   1   27  308
##
```

```
##                Accuracy : 0.958
##                  95% CI : (0.9465, 0.9676)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8801
##
##  Mcnemar's Test P-Value : 0.3135
##
##             Sensitivity : 0.8953
##             Specificity : 0.9766
##          Pos Pred Value : 0.9194
##          Neg Pred Value : 0.9691
##              Prevalence : 0.2295
##          Detection Rate : 0.2055
##    Detection Prevalence : 0.2235
##       Balanced Accuracy : 0.9360
##
##        'Positive' Class : 1
##
```

```
#Curva ROC
# Primero la calculamos para el conjunto de entrenamiento
pred.tr.bag.roc<-predict(bag,newdata = df.tr, type="prob")$prob[,2]

# Ahora la calculamos para el conjunto de validación
pred.va.bag.roc<-predict(bag,newdata = df.va, type="prob")$prob[,2]

AUC_bag<-roc(pred.tr.bag.roc,pred.va.bag.roc,"Bagging algorithm")
```




```
df_result<-rbind(df_result,data.frame(Metodo="Bagging algorithm",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_bag[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_bag[2])),
make.row.names=F)
```

Este método obtiene resultados muy buenos, muy similares al algoritmo de *Boosting* con sensibilidad y precisión sobre 0.9, lo que provoca un mayor *F1-Score* que en los casos de muestreo o en el árbol de decisión. Las conclusiones que se pueden obtener son las mismas que con el anterior algoritmo por lo que la elección de cualquiera de estos dos modelos es igual de acertada.

Random Forest

Utilizaremos el *Random Forest* con las condiciones de usaremos un total 8 variables en cada muestra y realizaremos la votación con solo 5 árboles.

```
# Modelo predictivo: 8 variables y 5 _arboles a combinar (con
 reemplazamiento)
rf <- randomForest(voluntary_turnover ~ .,data=df.tr,ntree=5,mtry=8,
replace=T)
rf
```

```
##
## Call:
##  randomForest(formula = voluntary_turnover ~ ., data = df.tr,
##  ntree = 5, mtry = 8, replace = T)
##                Type of random forest: classification
##                      Number of trees: 5
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 3.54%
## Confusion matrix:
##      0    1 class.error
## 0 8987  223  0.02421281
## 1  206 2703  0.07081471
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.rf.cm<-predict(rf,newdata = df.tr, type="class")
cm_t<-confusionMatrix(pred.tr.rf.cm,df.tr$voluntary_turnover,positive='1')

#Train
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10205    57
##          1    68  3170
##
##                Accuracy : 0.9907
```

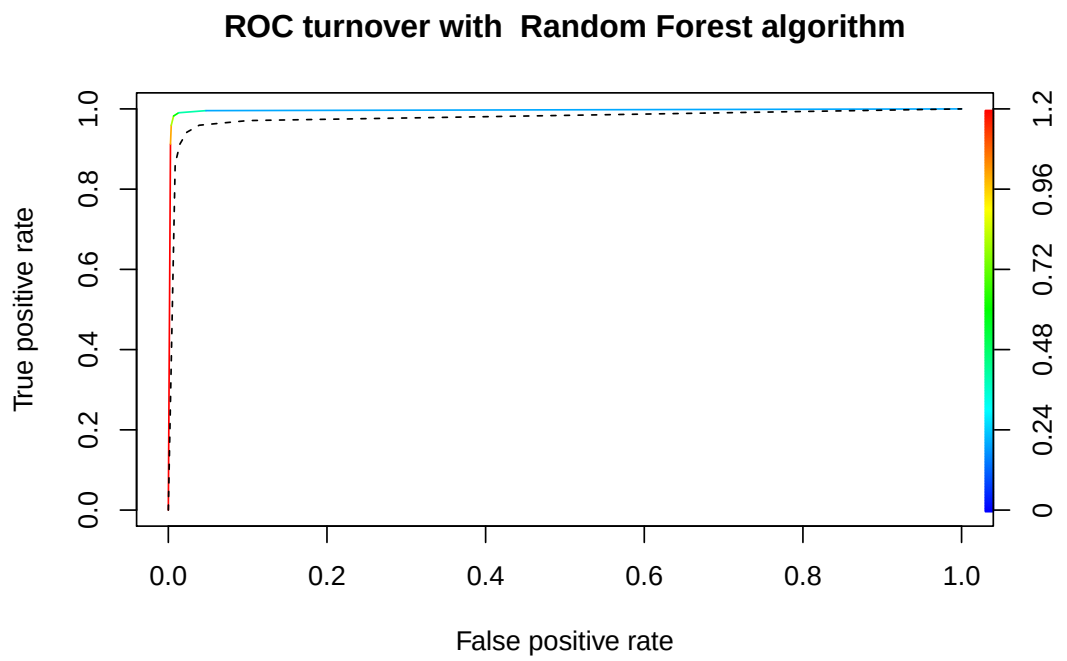
```
##                  95% CI : (0.989, 0.9923)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9746
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9823
##             Specificity : 0.9934
##          Pos Pred Value : 0.9790
##          Neg Pred Value : 0.9944
##              Prevalence : 0.2390
##          Detection Rate : 0.2348
##    Detection Prevalence : 0.2399
##       Balanced Accuracy : 0.9879
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.rf.cm<-predict(rf,newdata = df.va, type="class")
cm_v<-confusionMatrix(pred.va.rf.cm,df.va$voluntary_turnover,positive='1')

#Valid
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1128   20
##          1   27  324
##
##                Accuracy : 0.9686
```

```
##                  95% CI : (0.9585, 0.9769)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.912
##
##  Mcnemar's Test P-Value : 0.3815
##
##             Sensitivity : 0.9419
##             Specificity : 0.9766
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.9826
##              Prevalence : 0.2295
##          Detection Rate : 0.2161
##    Detection Prevalence : 0.2342
##       Balanced Accuracy : 0.9592
##
##        'Positive' Class : 1
##
```

```
#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.rf.roc<-predict(rf,newdata = df.tr, type="prob")[,2]

# Ahora la calculamos para el conjunto de validación
# Ahora mostramos la curva ROC para el validation dataset
pred.va.rf.roc<-predict(rf,newdata = df.va, type="prob")[,2]

AUC_rf<-roc(pred.tr.rf.roc,pred.va.rf.roc,"Random Forest algorithm")
```




```
df_result<-rbind(df_result, data.frame(Metodo="Random Forest",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_rf[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_rf[2])),
make.row.names=F)
```

Se obtienen resultados realmente buenos y difíciles de mejorar. Pese a que la diferencia de puntaje entre los conjuntos de entrenamiento y validación es mayor. Los grandes resultados que se observan en ambos grupos hace pensar que no se está incurriendo en *overfitting*. Destaca en la curva ROC como de rápido crece, sobre todo para el conjunto de *train*, que parece una línea recta vertical, paralela al eje de abscisas. Esto se observa al ver el área de dicha curva, la métrica AUC, que es muy próximo a 1. Cuando dicho indicador es 1 estaríamos ante el hipotético caso de que el modelo es perfecto.

Regresión logísitica

Utilizaremos la regresión logística con la función de distribución binomial para el modelo.

```
lr<-glm(voluntary_turnover ~ .,data=df.tr, family= binomial)
lr
```

```
##
## Call:  glm(formula = voluntary_turnover ~ ., family = binomial,
data = df.tr)
##
## Coefficients:
##                     (Intercept)     satisfaction_level[0.465,1)
##                        -0.10093                         -2.79019
##     satisfaction_level[1, Inf]    last_evaluation[0.575,0.765)
##                       -17.67596                         -2.77358
##       last_evaluation[0.765,1)         last_evaluation[1, Inf]
##                         0.25942                          2.15495
##                 number_project   average_montly_hours[160,218)
##                        -0.16722                         -2.41057
##  average_montly_hours[218,288)   average_montly_hours[288,310)
##                         0.37847                         18.47505
## average_montly_hours[310, Inf]              time_spend_company
##                        16.97158                          0.31805
##                 Work_accident1          promotion_last_5years1
##                        -1.57290                         -1.42104
##                   departmenthr                   departmentIT
##                         0.34732                          0.01242
##           departmentmanagement             departmentmarketing
##                        -0.26352                          0.11899
##          departmentproduct_mng                departmentRandD
##                         0.04814                         -0.41794
##                departmentsales               departmentsupport
##                         0.09009                          0.20921
##            departmenttechnical                      salarylow
##                         0.26670                          1.91998
##                   salarymedium                     GenderMale
##                         1.43516                         -0.60593
##                 Age[20.5,59.5)                   Age[59.5,65)
```

```
##                       -0.85496                       -0.67786
##                   Age[65, Inf]
##                       -0.45224
##
## Degrees of Freedom: 13499 Total (i.e. Null);  13471 Residual
## Null Deviance:       14850
## Residual Deviance: 7446  AIC: 7504
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.lr.cm<-predict(lr,newdata = df.tr, type="response")
pred.tr.lr.cm[pred.tr.lr.cm>=0.5]<-1
pred.tr.lr.cm[pred.tr.lr.cm<0.5]<-0
cm_t<-confusionMatrix(as.factor(pred.tr.lr.cm),reference =
df.tr$voluntary_turnover,positive='1')

# Train
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9694  926
##          1  579 2301
##
##                Accuracy : 0.8885
##                  95% CI : (0.8831, 0.8938)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6818
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7130
```

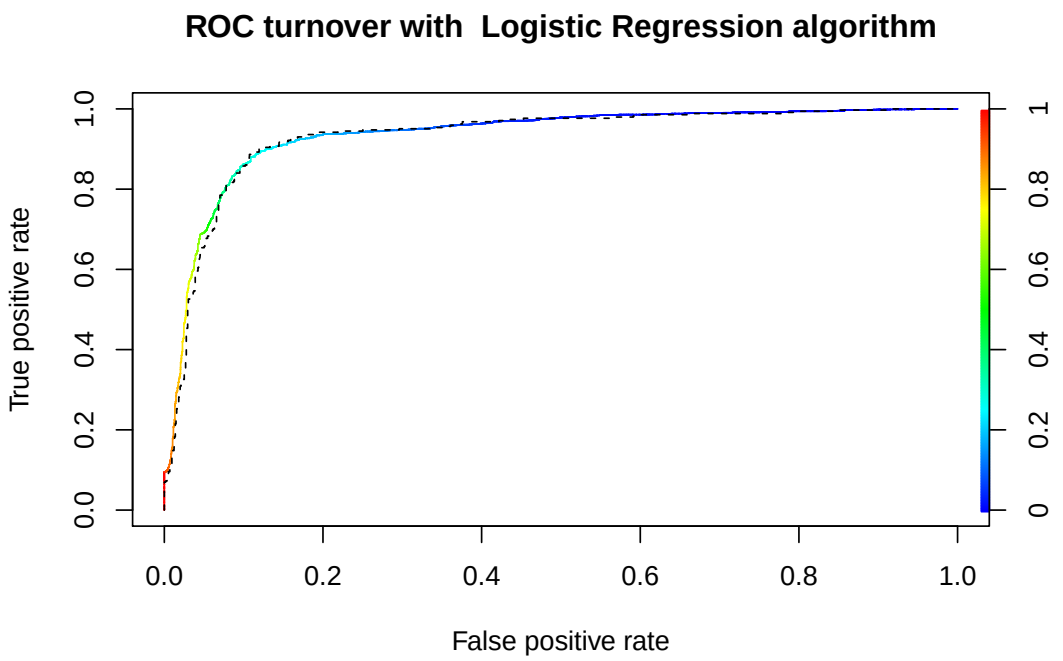
```
##             Specificity : 0.9436
##          Pos Pred Value : 0.7990
##          Neg Pred Value : 0.9128
##              Prevalence : 0.2390
##          Detection Rate : 0.1704
##    Detection Prevalence : 0.2133
##       Balanced Accuracy : 0.8283
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.lr.cm<-predict(lr,newdata = df.va, type="response")
pred.va.lr.cm[pred.va.lr.cm>=0.5]<-1
pred.va.lr.cm[pred.va.lr.cm<0.5]<-0
cm_v<-confusionMatrix(as.factor(pred.va.lr.cm),df.va$voluntary_turnover,
positive='1')

# Valid
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1088  108
##          1   67  236
##
##                Accuracy : 0.8833
##                  95% CI : (0.8659, 0.8991)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6555
##
```

```
##  Mcnemar's Test P-Value : 0.002497
##
##             Sensitivity : 0.6860
##             Specificity : 0.9420
##          Pos Pred Value : 0.7789
##          Neg Pred Value : 0.9097
##              Prevalence : 0.2295
##          Detection Rate : 0.1574
##    Detection Prevalence : 0.2021
##       Balanced Accuracy : 0.8140
##
##        'Positive' Class : 1
##
```

```
#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.lr.roc<-predict(lr,newdata = df.tr, type="response")

# Ahora la calculamos para el conjunto de validación
# Ahora mostramos la curva ROC para el validation dataset
pred.va.lr.roc<-predict(lr,newdata = df.va, type="response")


AUC_lr<-roc(pred.tr.lr.roc,pred.va.lr.roc,"Logistic Regression algorithm")
```




```
df_result<-rbind(df_result, data.frame(Metodo="Logistic Regression algorithm",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_lr[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_lr[2])),
make.row.names=F)
```

Para este indicador obtenemos, con bastante diferencia, las métricas más bajas. Además, se puede observar, viendo la matriz de confusión para el conjunto de validación o observando la sensibilidad y precisión de dicho conjunto, que es un clasificador que no suele acertar tanto con las fugas, ya que suele clasificar muchos casos como no fugas lo que penalizaría mucho la clase minoritaria, que es la que nos interesa en este análisis. Todo esto se debe al asunción de linealidad entre las variables independientes, se debe realizar un estudio de la multicolinealidad de dichas variables y realizar una selección de variables antes de hacer la regresión.

SVM (Support Vector Machine)

Como ya hemos comentado, las Máquinas de Vector de Soporte (SVM, Support Vector Machines) son un conjunto de algoritmos en los que se utiliza un hiperplano para separar los puntos etiquetados con diferentes categorías, dejando los puntos de una categoría a un lado del plano y los de otra al otro. Un hiperplano es la generalización del concepto de plano que divide el espacio tridimensional en dos con cualquier número de dimensiones.

```
#Fit a model. The function syntax is very similar to lm function
svm <- svm(voluntary_turnover ~ .,data=df.tr, probability=TRUE)
print(svm)
```

```
##
## Call:
## svm(formula = voluntary_turnover ~ ., data = df.tr, probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  2309
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.svm.cm<-predict(svm,newdata = df.tr, type="class")
cm_t<-confusionMatrix(pred.tr.svm.cm,df.tr$voluntary_turnover,positive='1')

# Train
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10045   277
```

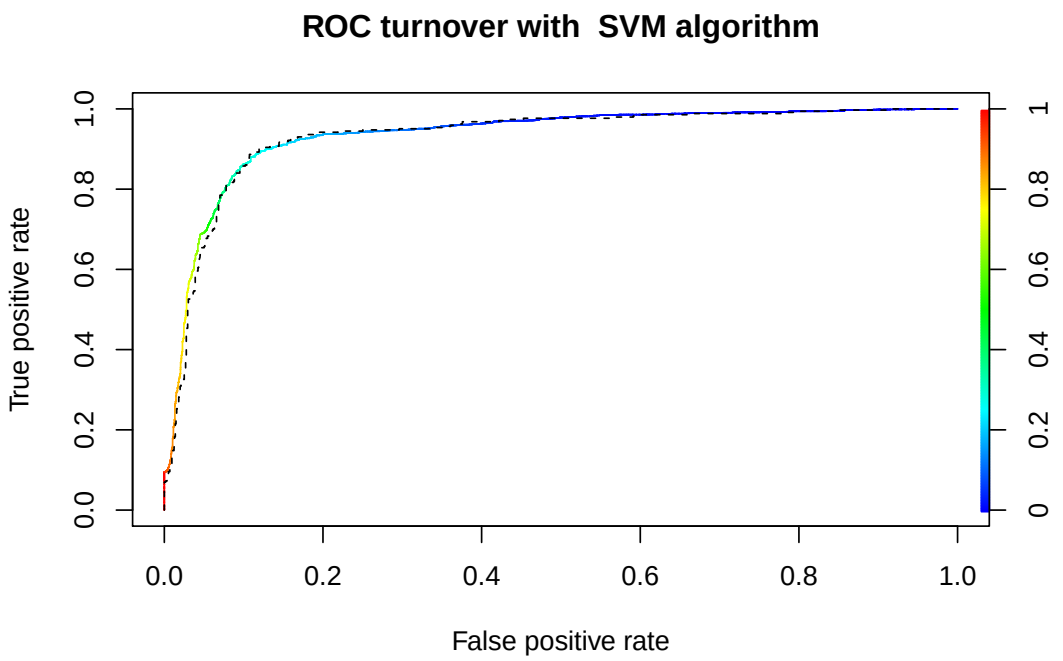
```
##          1   228  2950
##
##                Accuracy : 0.9626
##                  95% CI : (0.9593, 0.9657)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8966
##
##  Mcnemar's Test P-Value : 0.03268
##
##             Sensitivity : 0.9142
##             Specificity : 0.9778
##          Pos Pred Value : 0.9283
##          Neg Pred Value : 0.9732
##              Prevalence : 0.2390
##          Detection Rate : 0.2185
##    Detection Prevalence : 0.2354
##       Balanced Accuracy : 0.9460
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.svm.cm<-predict(svm,newdata = df.va, type="class")
cm_v<-confusionMatrix(pred.va.svm.cm,df.va$voluntary_turnover,positive='1')

# Valid
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1126   35
```

```
##          1   29  309
##
##                Accuracy : 0.9573
##                  95% CI : (0.9458, 0.967)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8785
##
##  Mcnemar's Test P-Value : 0.532
##
##             Sensitivity : 0.8983
##             Specificity : 0.9749
##          Pos Pred Value : 0.9142
##          Neg Pred Value : 0.9699
##              Prevalence : 0.2295
##          Detection Rate : 0.2061
##    Detection Prevalence : 0.2255
##       Balanced Accuracy : 0.9366
##
##        'Positive' Class : 1
##
```

```
#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.svm.roc<-attr(predict(svm,newdata = df.tr, probability=TRUE),
"probabilities")[,2]
# Ahora la calculamos para el conjunto de validación
# Ahora mostramos la curva ROC para el validation dataset
pred.va.svm.roc<-attr(predict(svm,newdata = df.va, probability=TRUE),
"probabilities")[,2]

AUC_svm<-roc(pred.tr.lr.roc,pred.va.lr.roc,"SVM algorithm")
```




```
df_result<-rbind(df_result,data.frame(Metodo="SVM",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_svm[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_svm[2])),
make.row.names=F)
```

En principio, el *SVM* no es un método que se ajusta perfectamente a lo que tenemos. Por un lado, porque no suele ser eficiente en conjutos de datos extensos. Por otro lado, porque es más eficiente en casos donde el número de variables excede al de muestras, cuándo, claramente este no es el caso.

Pese a eso, las métricas son bastante buenas, bastante similares a los algoritmos de *Bagging* y al *Adaboost* y esto se debe a que, pese a todo lo anterior, las clases están bastante diferenciadas, por lo que la selección del hiperplano que las separe es posible.

Naïves Bayes

Un clasificador Bayesiano Ingenuo (*Naïve Bayes*) es un clasificador probabilístico que se basa en el teorema de Bayes y en algunas hipótesis simplificadoras adicionales.

```
nb<-naiveBayes(voluntary_turnover ~ .,data=df.tr, probability=TRUE)
nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace, probability = TRUE)
##
## A-priori probabilities:
## Y
##        0        1
## 0.760963 0.239037
##
## Conditional probabilities:
##    satisfaction_level
## Y   [-Inf,0.09) [0.09,0.465)  [0.465,1)   [1, Inf]
##   0  0.00000000   0.14552711 0.84493332 0.00953957
##   1  0.00000000   0.71149675 0.28850325 0.00000000
##
##    last_evaluation
## Y   [-Inf,0.36) [0.36,0.575) [0.575,0.765)  [0.765,1)   [1, Inf]
##   0  0.00000000   0.23702911    0.36328239 0.38956488 0.01012363
##   1  0.00000000   0.44220638    0.03005888 0.48032228 0.04741246
##
##    number_project
## Y       [,1]      [,2]
##   0 3.786041 0.9751447
##   1 3.879764 1.8255734
##
##    average_montly_hours
## Y     [-Inf,96)    [96,160)   [160,218)   [218,288)   [288,310)  [310, Inf]
##   0 0.000000000 0.241798890 0.380317337 0.377883773 0.000000000 0.000000000
##   1 0.000000000 0.436008677 0.035946700 0.433219709 0.089556864 0.005268051
##
```

```
##    time_spend_company
## Y       [,1]      [,2]
##   0 3.380609 1.5647318
##   1 3.880694 0.9715525
##
##    Work_accident
## Y            0          1
##   0 0.82273922 0.17726078
##   1 0.95320731 0.04679269
##
##    promotion_last_5years
## Y             0           1
##   0 0.974593595 0.025406405
##   1 0.995661605 0.004338395
##
##    department
## Y   accounting         hr         IT management  marketing product_mng
##   0 0.04964470 0.04526429 0.08439599 0.04691911 0.05665336  0.06210455
##   1 0.05453982 0.06073753 0.07809111 0.02479083 0.05794856  0.05825844
##    department
## Y        RandD      sales    support  technical
##   0 0.05947630 0.27216977 0.14572179 0.17765015
##   1 0.03346762 0.28075612 0.15308336 0.19832662
##
##    salary
## Y         high        low     medium
##   0 0.10240436 0.44913852 0.44845712
##   1 0.02293152 0.60613573 0.37093275
##
##    Gender
## Y      Female      Male
##   0 0.2994257 0.7005743
##   1 0.4480942 0.5519058
##
```

```
##    Age
## Y     [-Inf,20)   [20,20.5) [20.5,59.5)   [59.5,65)   [65, Inf]
##   0 0.000000000 0.010512995 0.875985593 0.103961842 0.009539570
##   1 0.000000000 0.016423923 0.854663774 0.119305857 0.009606446
```

```
#Pred sobre train para mostrar matriz de conf
pred.tr.svm.cm<-predict(nb,newdata = df.tr, type="class")
cm_t<-confusionMatrix(pred.tr.svm.cm,df.tr$voluntary_turnover,positive='1')

# Train
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9719  931
##          1  554 2296
##
##                Accuracy : 0.89
##                  95% CI : (0.8846, 0.8952)
##     No Information Rate : 0.761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.685
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7115
##             Specificity : 0.9461
##          Pos Pred Value : 0.8056
##          Neg Pred Value : 0.9126
##              Prevalence : 0.2390
##          Detection Rate : 0.1701
##    Detection Prevalence : 0.2111
```

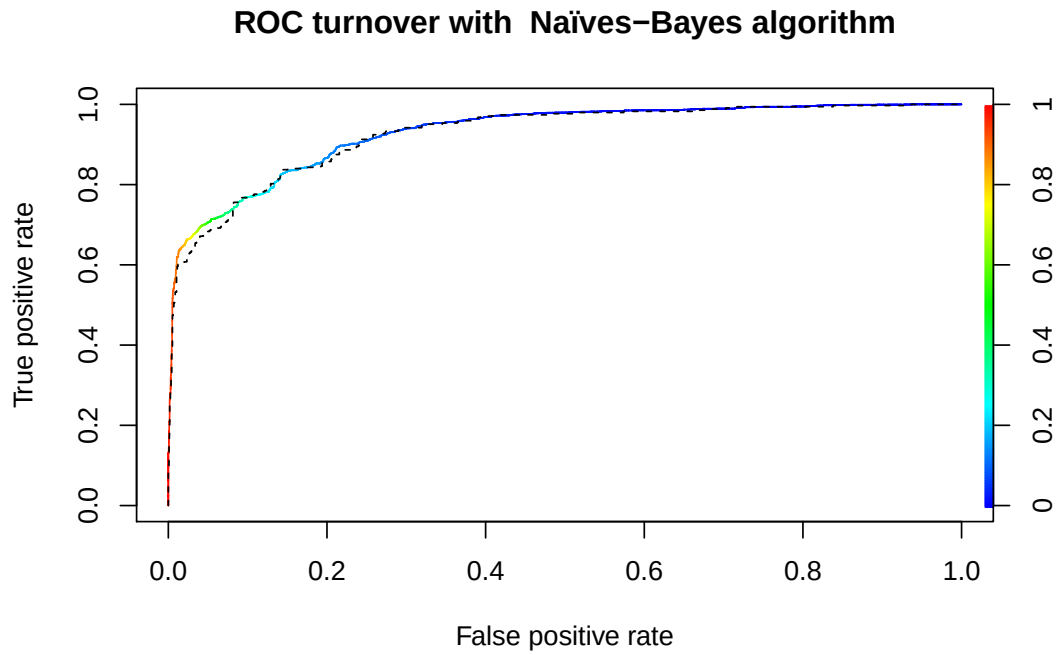
```
##       Balanced Accuracy : 0.8288
##
##        'Positive' Class : 1
##
```

```
#Pred sobre validation para mostrar matriz de conf
pred.va.svm.cm<-predict(nb,newdata = df.va, type="class")
cm_v<-confusionMatrix(pred.va.svm.cm,df.va$voluntary_turnover,positive='1')

# Valid
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1093  108
##          1   62  236
##
##                Accuracy : 0.8866
##                  95% CI : (0.8694, 0.9022)
##     No Information Rate : 0.7705
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6635
##
##  Mcnemar's Test P-Value : 0.0005578
##
##             Sensitivity : 0.6860
##             Specificity : 0.9463
##          Pos Pred Value : 0.7919
##          Neg Pred Value : 0.9101
##              Prevalence : 0.2295
##          Detection Rate : 0.1574
##    Detection Prevalence : 0.1988
```

```
##        Balanced Accuracy : 0.8162
##
##         'Positive' Class : 1
##
```

```
#Curva ROC
# Priero la calculamos para el conjunto de entrenamiento
pred.tr.nb.roc<-predict(nb,newdata = df.tr, type="raw")[,2]

# Ahora la calculamos para el conjunto de validación
# Ahora mostramos la curva ROC para el validation dataset
pred.va.nb.roc<-predict(nb,newdata = df.va, type="raw")[,2]


AUC_nb<-roc(pred.tr.nb.roc,pred.va.nb.roc,"Naïves-Bayes algorithm")
```



```
df_result<-rbind(df_result, data.frame(Metodo="Naïves-Bayes",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_nb[1]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_nb[2])),
make.row.names=F)
```

Obtenemos resultados bastante peores que los anteriores métodos, a excepción de la regresión logística y esto se debe a la asunción del algoritmo de que las variables que se usan para la predicción son condicional independientes. Durante el EDA se han creados grupos con los atributos dadas su similitud, se debería haber realizado una selección de variables, eliminando las dependientes y por lo tanto, disminuyendo el total de las variables que se utilizarían en el modelo.

Red Neuronal

En primer lugar, prepararemos nuestros datos, ya que en una red neuronal solo acepta valores numéricos usaremos diferentes técnicas de *preprocesing* tales como el *"one-hot encoding"* o la creación de una *"recipe"* dónde convertiremos las variables nominales a binarias que luego normalizaremos buscando que la media y la desviación típica sea cero.

Luego crearemos nuestra red neuronal con tres capas con diferente número de neuronas en cada capa (18, 8 y 10). Entrenaremos la red neuronal y comprobaremos nuevas medidas para *train* y para *test* (*loss*, *binnary_ accuracy* y *mse*). Sobre estos indicadores hablaremos más adelante.

Finalmente, mostraremos las mismas métricas que hemos mostrado en los otros modelos y los guardaremos en la tabla.

```
set.seed(123456)
idx <- sample.int(n = nrow(df), size = floor(0.1*nrow(df)), replace = FALSE)
df.va <- df[idx,]
df.tr <- df[-idx,]
df$voluntary_turnover<-as.numeric(df$voluntary_turnover)

train_data <- df.tr
test_data  <- df.va
```

```
recipe <- recipe(voluntary_turnover ~ ., train_data) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_center(all_predictors(), -all_outcomes()) %>%
  step_scale(all_predictors(), -all_outcomes()) %>%
  prep(data = train_data)

# Apply recipe to three datasets
train_data <- bake(recipe, new_data = train_data) %>%
  select(voluntary_turnover, everything())

test_data <- bake(recipe, new_data = test_data) %>%
  select(voluntary_turnover, everything())

train_y_drop <- as.double(as.character(train_data$voluntary_turnover))
test_y_drop <- as.double(as.character(test_data$voluntary_turnover))

train_data_bk <- select(train_data, -voluntary_turnover)
test_data_bk <- select(test_data, -voluntary_turnover)
# Define a simple MLP
model_keras <- keras_model_sequential()

model_keras %>%
  layer_dense(units = 18, kernel_initializer = "uniform",
activation = "relu",
input_shape = ncol(train_data_bk)) %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 8, kernel_initializer = "uniform",
activation = "relu") %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 10, kernel_initializer = "uniform",
activation = "relu") %>%
  layer_dropout(rate = 0.1) %>%
  layer_dense(units = 1,
kernel_initializer = "uniform", activation = "sigmoid") %>%
```

```
  compile(
optimizer = 'adamax',
loss      = 'binary_crossentropy',
metrics   = c("binary_accuracy", "mse")
  )

summary(model_keras)
```

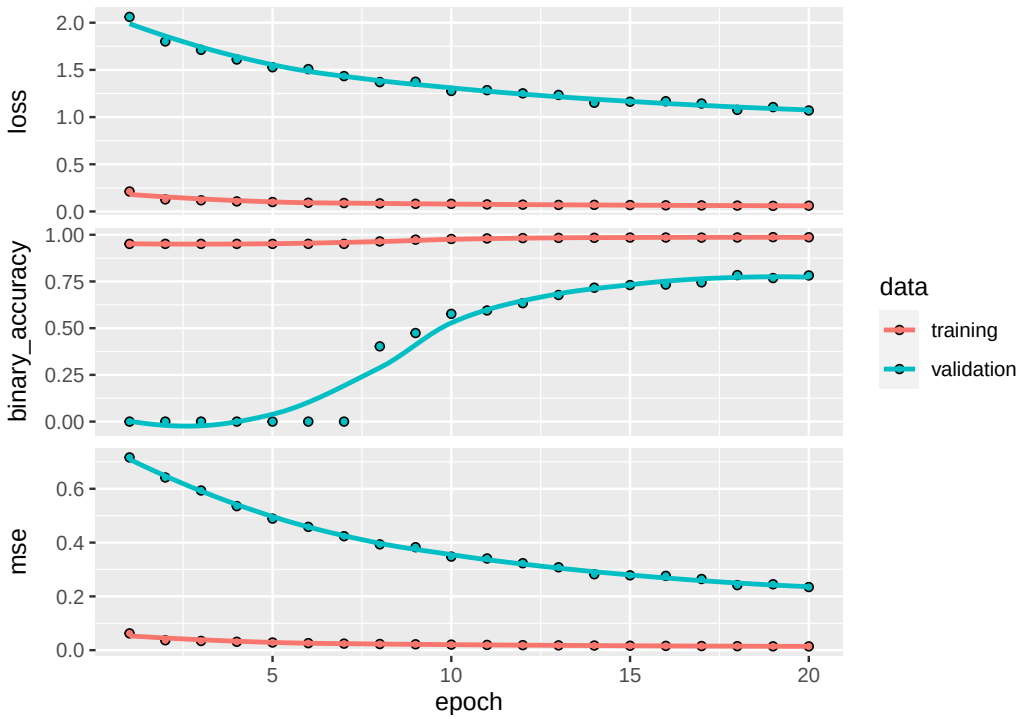
```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                    Output Shape                  Param #
## =========================================================================
## dense (Dense)                   (None, 18)                    594
## _________________________________________________________________________
## dropout (Dropout)               (None, 18)                    0
## _________________________________________________________________________
## dense_1 (Dense)                 (None, 8)                     152
## _________________________________________________________________________
## dropout_1 (Dropout)             (None, 8)                     0
## _________________________________________________________________________
## dense_2 (Dense)                 (None, 10)                    90
## _________________________________________________________________________
## dropout_2 (Dropout)             (None, 10)                    0
## _________________________________________________________________________
## dense_3 (Dense)                 (None, 1)                     11
## =========================================================================
## Total params: 847
## Trainable params: 847
## Non-trainable params: 0
## _________________________________________________________________________
```

```
summary(factor(train_y_drop))
```

```
##     0     1
## 10273  3227
```

```
fit_keras <- fit(model_keras,
x = as.matrix(train_data_bk),
y = train_y_drop,
batch_size = 5,
epochs = 20,
validation_split = 0.20,
# validation_data = list(as.matrix(valid_data_bk), valid_y_drop),
verbose = 2)

# fit_keras

# Plot Keras training results
plot(fit_keras)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
pred_classes_train <- predict_classes(object = model_keras,
x = as.matrix(train_data_bk))
pred_proba_train  <- predict_proba(object = model_keras,
x = as.matrix(train_data_bk))
# summary(factor(pred_classes_train))

train_results <- tibble(
actual_yes = factor(as.vector(train_y_drop)),
pred_classes_train = factor(pred_classes_train),
Yes = as.vector(pred_proba_train),
No = 1 - as.vector(pred_proba_train))
# summary(train_results)

cm_t<-confusionMatrix(train_results$actual_yes,
train_results$pred_classes_train,positive = "1")
cm_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10224    49
##          1   675  2552
##
##                Accuracy : 0.9464
##                  95% CI : (0.9424, 0.9501)
##     No Information Rate : 0.8073
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8421
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9812
##             Specificity : 0.9381
```

```
##          Pos Pred Value : 0.7908
##          Neg Pred Value : 0.9952
##              Prevalence : 0.1927
##          Detection Rate : 0.1890
##    Detection Prevalence : 0.2390
##       Balanced Accuracy : 0.9596
##
##        'Positive' Class : 1
##
```

```
AUC_NR_TR <-train_results %>%
  roc_auc(actual_yes, Yes)
AUC_NR_TR
```

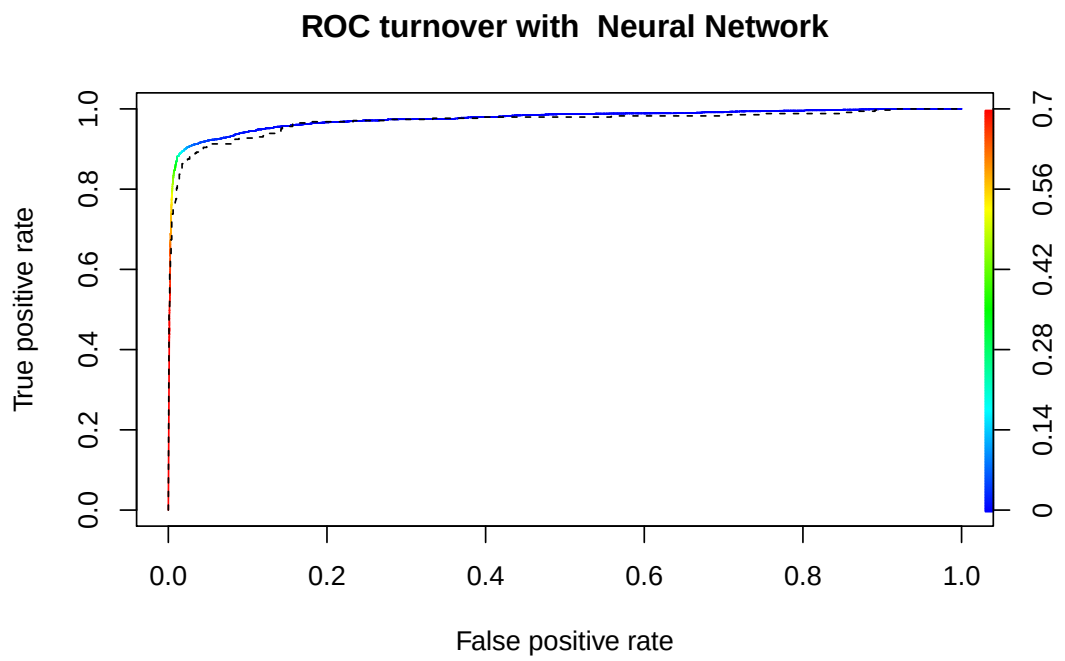
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary        0.0244
```

```
############
# Predict classes and probabilities
pred_classes_test <- predict_classes(object = model_keras,
x = as.matrix(test_data_bk))
pred_proba_test  <- predict_proba(object = model_keras,
x = as.matrix(test_data_bk))

test_results <- tibble(
actual_yes = factor(as.vector(test_y_drop)),
pred_classes_test = factor(as.vector(pred_classes_test),levels = c(0,1)),
Yes = as.vector(pred_proba_test),
No = 1 - as.vector(pred_proba_test))
# summary(test_results)

cm_v<-confusionMatrix(test_results$actual_yes,test_results$pred_classes_test
,positive = "1")
```

```
cm_v
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1146    9
##          1   84  260
##
##                Accuracy : 0.938
##                  95% CI : (0.9245, 0.9496)
##     No Information Rate : 0.8205
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.81
##
##  Mcnemar's Test P-Value : 1.674e-14
##
##             Sensitivity : 0.9665
##             Specificity : 0.9317
##          Pos Pred Value : 0.7558
##          Neg Pred Value : 0.9922
##              Prevalence : 0.1795
##          Detection Rate : 0.1734
##    Detection Prevalence : 0.2295
##       Balanced Accuracy : 0.9491
##
##        'Positive' Class : 1
##
```

```
AUC_nn<-roc(pred_proba_train,pred_proba_test,"Neural Network")
```




```
AUC_NR_TE<-test_results %>%
  roc_auc(actual_yes, Yes)
AUC_NR_TE
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary        0.0307
```

```
df_result<-rbind(df_result, data.frame(Metodo="Neural Network",
Sensitivity_train=cm_t$byClass[1],Precission_train=cm_t$byClass[5],
F_Score_train=cm_t$byClass[7],AUC_train=as.numeric(AUC_NR_TR[3]),
Sensitivity_test=cm_v$byClass[1],Precission_test=cm_v$byClass[5],
F_Score_test=cm_v$byClass[7],AUC_test=as.numeric(AUC_NR_TE[3])),
make.row.names=F)
```

Pese a que una red neuronal, es considerada una "black box" por lo que no se pude saber concretamente, el funcionamiento "interior" del modelo, observando las métricas obtenemos

los mejores resultados de sensibilidad hasta el momento, a costa de una precisión peor que en los demás casos. Esto se podría traducir a que consigue seleccionar a casi toda la población que se fuga a costa un exceso de clasificación errónea de empleados que no se fugan como fugas. Por otro lado, también se debería comentar que las métricas (*binary_ accuracy*, *loss* y *mse*) difieren bastante en *train* y en *validation*, lo que podría ser un sobreajuste.

4.3. Resultados del análisis

Una vez realizado el análisis sólo queda ver los resultados y tomar la decisión de qué modelo elegir.

Conviene recordar que lo que se pretende es encontrar aquel método con una serie de métricas favorables, es decir, una buena precisión en la clasificación, un elevado porcentaje de clasificación correcta (*AUC* alto) y un número de predicciones que se aproxime al valor real presente en la muestra. Por ello nos centraremos en métricas como el *F1-Score, Precission o Sensitivity*, que se centran en las clases positivas detectadas (en las fugas de empleados, nuestro problema a solucionar), mientras que otros indicadores como la *accuracy* los dejaremos de lado.

En la imagen 4.3 se observa los resultados obtenidos, están coloreadas en rojo los máximos por columna. Destacan dos métodos por encima de todos, la red neuronal y el algoritmo *Random Forest*. Pero podemos destacar varias conclusiones:

- Los métodos de balanceo de datos provocan índices de *Precission* muy bajos, tanto en *train* como en *test*. En general la sensibilidad es alta al haber pocos casos en la clase minoritaria.
- Utilizar técnicas de balanceo con respecto a ejecutar simplemente el árbol de decisión no mejora los resultados.
- Las métricas de las técnicas de balanceo son casi iguales para las tres técnicas de balanceo usadas.
- Los algoritmos de *bagging*, *SVM* y de *boosting* obtienen resultados similares (ligeramente mejores los de *SVM*).
- El algoritmo *Naïves-Bayes* y la regresión logística obtienen los peores resultados y

Método	Sensitivity_train	Precission_train	F_Score_train	AUC_train	Sensitivity_test	Precission_test	F_Score_test	AUC_test
Decission Tree	0.9262473	0.8052263	0.8615074	0.9544605	0.9040698	0.7853535	0.8405405	0.9456785
Oversamplimg techniques	0.9377130	0.7563109	0.8372994	0.9351846	0.9186047	0.7365967	0.8175938	0.9262496
Undersamplim g techniques	0.9358537	0.7559449	0.8363334	0.9349224	0.9186047	0.7365967	0.8175938	0.9261540
Resamplimg techniques	0.9358537	0.7559449	0.8363334	0.9348956	0.9186047	0.7365967	0.8175938	0.9262496
Boosting	0.9023861	0.9279796	0.9150039	0.9595350	0.8982558	0.9169139	0.9074890	0.9533298
Bagging algorithm	0.9017663	0.9335900	0.9174023	0.9421161	0.8953488	0.9194030	0.9072165	0.9371338
Random Forest	0.9823365	0.9789994	0.9806651	0.9959760	0.9418605	0.9230769	0.9323741	0.9777824
Logistic Regression algorithm	0.7130462	0.7989583	0.7535615	0.9337021	0.6860465	0.7788779	0.7295209	0.9308492
SVM	0.9141618	0.9282568	0.9211553	0.9337021	0.8982558	0.9142012	0.9061584	0.9308492
Naïves-Bayes	0.7114967	0.8056140	0.7556360	0.9296614	0.6860465	0.7919463	0.7352025	0.9260797
Neural Network	0.9830303	0.7539510	0.8533848	0.9770108	0.9694656	0.7383721	0.8382838	0.9697184

se debe a la no realización de test que busque la independencia y multicolinealidad de las variables, respectivamente.

- Pese a que la red neuronal y el *Random Forest* son los algoritmos que acaparan los máximos por columna, nos decantaremos por el segundo, ya que mantiene todas sus métricas altas y, además, observando los gráficos del aprendizaje de la red neuronal, vemos que las métricas (*binary_ accuracy*, *loss* y *mse*) difieren bastante en *train* y en *validation*, lo que podría ser un sobreajuste.

Por último, comentar, como futuros modelos que no desarrollaremos en este trabajo, la creación de modelos híbridos combinando las técnicas y modelos anteriores suelen ser

soluciones bastante óptimas a este problema.

Capítulo 5

Visualización de datos con *SAP Analytics Cloud*

Una vez realizado el análisis del conjunto de datos y elegido el modelo que usaremos, ahora sólo queda lanzarlo sobre el conjunto de datos que queremos predecir (los que no sabemos si se marchan o no).

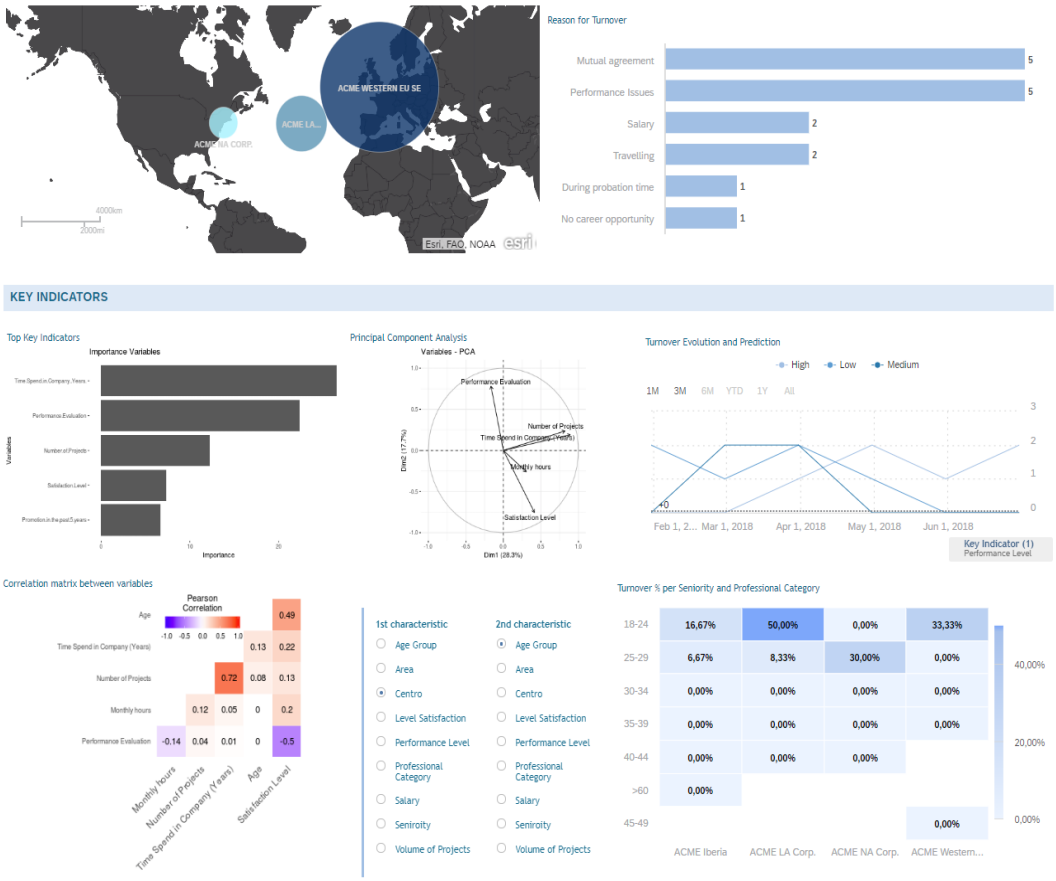
A modo de dar una explicación de la utilidad de esta aplicación, realizaremos un caso de uso de este conjunto con las predicciones realizadas. Para ello usaremos un dataset con las variables del análisis (a excepción de la variable objetivo, que será sustituida por la predicción) añadiéndole alguna más como, por ejemplo, la localización. Destacar, que los datos utilizados son ficticios inventados o obtenidos a través de Internet.

El caso de uso consistirá en un cuadro de mandos de la plantilla de una empresa donde se verá qué ha pasado con la plantilla y qué va a pasar. Se comentará por encima cada uno de los gráficos sin entrar en detalle. *SAP Analytics Cloud* es una herramienta de visualización de datos desarrollada por la compañía *SAP* con el fin de englobar todas sus herramientas de visualización y *reporting* en una única más potente.

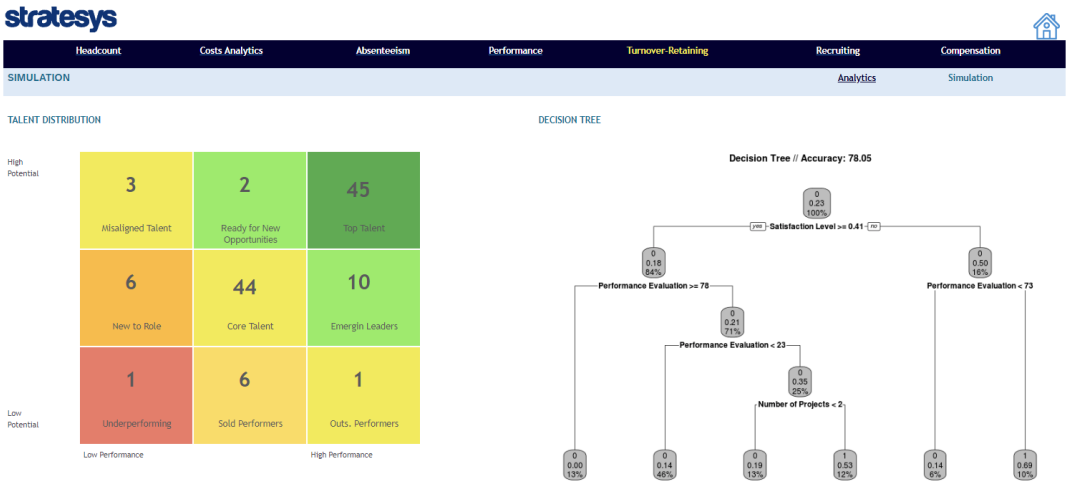
Volviendo a nuestro caso de uso, estará formado por dos pestañas:

La primera pestaña se realizará un análisis que se centrará en el pasado, en qué ha pasado hasta el momento, veremos indicadores de los totales de empleados de la compañía, la tasa de rotación, comparaciones con respecto a la competencia, evolución temporal (con predicción) del porcentaje de fuga, análisis del salario medio por categoría profesional, *turnover* por área, por categoría ...

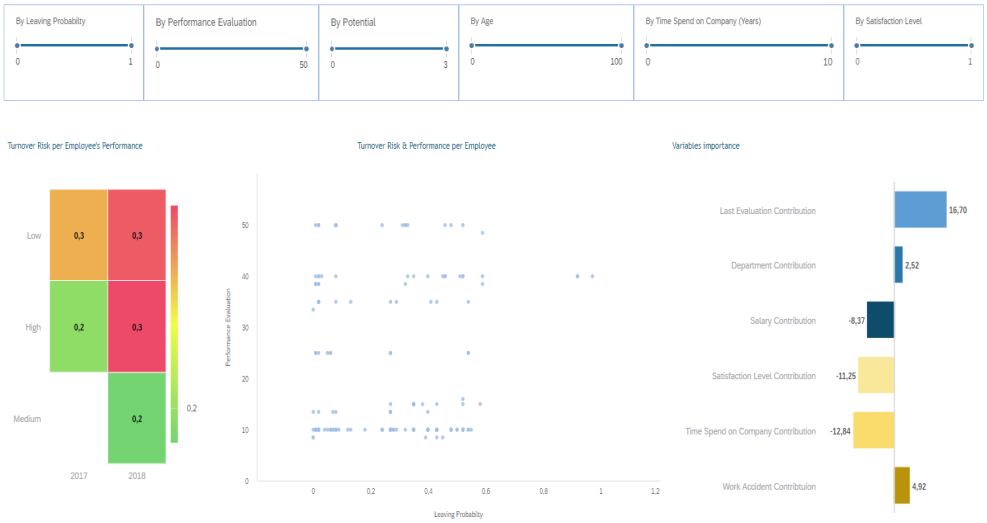
Esta pestaña tiene una segunda parte en la que, después de mostrar dónde están distribuidos los empleados que se han fugado y su razón de fuga, se hablará de las razones de marcha. Sobre los datos históricos se efectúan algoritmos (Random Forest, PCA...) para observar cuales son los patrones que más se repiten en la marcha de empleados.



La segunda pestaña del dashboard, se centra en una parte más predictiva, se visualizan indicadores sobre los motivos de marchas y un árbol de decisión con los principales atributos de los empleados y su distribución en base a una posible fuga.



Por último, se observa, con la intención de encontrar patrones entre los empleados y usando los datos obtenidos por la predicción, una comparación de la probabilidad de marcha del empleado con respecto a su rendimiento. Además se puede visualizar, para cada empleado, los motivos que influyen positivamente o negativamente en esta decisión.



Este cuadro de mando es una herramienta muy útil a nivel empresarial, ya que se pueden sacar conocimientos sobre tu plantilla y realizar operaciones personalizas a cada empleado o grupo de empleados, como, por ejemplo, la subida salarial, en caso de ser el motivo de marcha de un empleado, con el fin de buscar retenerlo reduciendo lo costes al

hacerla subida al grupo de personas con mayor probabilidad de marcha.

Capítulo 6

Conclusiones

Una vez comentado el concepto de *turnover*, los tipos que hay y los diferentes factores que influyen, además de, conocer las bases y conceptos estadísticos que utilizamos en nuestro estudio y las pruebas pertinentes sobre nuestros datos y mostrar un caso da uso con dichos datos, me gustaría recordar, que dicho análisis ha sido aplicado sobre la variable *turnover* en el caso de que sea voluntario. Finalmente, podemos sacar varias conclusiones:

- Pese a realizarse un árbol de decisión bastante limitado y simple, los resultados son bastante buenos y mejores que al añadirle un método de muestro a la hora de preprocesar los datos.
- Los métodos de balanceo obtienen buenos resultados a excepción de la métrica *Precission*, que empeora debido al cambio de proporción de las clases del conjunto.
- El mayor coste computacional y los pobres resultados hacen pensar que los métodos de muestreo no sean los más indicados para este caso.
- Los algoritmos de *bagging*, *boosting* y *SVM* consiguen solucionar el problema de la diferencia de valor con las métricas *Sensibility* y *Precission*, obteniendo resultados más próximos y altos. INformación que se puede corroborar al observar el alto valor del *f1-Score*. Por lo que, pese a no ser el calificador con mejores métricas, se considera un modelo robusto y eficiente que consigue aplacar el problema de buscar a los empreados que se fugan.
- Sorprenden los buenos resultados para el modelo que usa el algoritmo *SVM* pese a no tener los datos con las características que favorecen el uso de este método: ya que

los *datasets* muy largos y con más registros que variables no favorecen a su correcta ejecución.

- El algoritmo *Naïve Bayes* obtiene uno de los peores resultados, y eso es debido a la asunción de que las variables que influyen en el estudio son independientes. Además se puede ver en la tabla de correlaciones y al hacer un clúster jerárquico, ciertas relaciones entre las variables del estudio por lo que se podría concluir que, para un con correcto uso de este algoritmo conviene una selección previa de variables.

- Algo similar a lo que le sucede al algoritmo *Naïve-Bayes* le sucede a la regresión logística, no se obtienen los resultados esperados y esto es debido a la asunción de no multicolinealidad entre las variables independientes. Por lo que deberías ser necesaria la busqueda de linealidad entre las variables independientes y realizar un selección de variables, solucionando dicho problema, antes de la ejecución del algoritmo.

- Las redes neuronales obtienen buenos resultados, pero analizando su ajuste vemos que recurre a *overfitting*.

- El algoritmo *Random Forest*, conocido por su robustez y eficacia, es considerado como el método más eficiente para solucionar nuestro problema con la marchade empleado, ya que se obtiene muy buenas métricas dónde, en muchos casos son las más altas de todas las estudiadas en los otros algoritmos.

Por último, comentar que existen más métodos interesantes para aplicar a nuestros datos pero por motivos de tiempo, no han sido desarrollados en este trabajo.

Bibliografía

[1] DATASET "HR_COMMA_SEP.CSV", HTTPS://WWW.KAGGLE.COM/LIUJIAQI/HR-COMMA-SEPCSV

[2] "APROXIMACIÓN A LOS FACTORES DETERMINANTES DEL CHURN DESDE UN ENFOQUE DE MARKETING RELACIONAL INNOVADOR: EL PUNTO DE VISTA DE LOS PROVEEDORES Y CLIENTES DE SERVICIOS" DOCTORANDA: FERNANDA ANDRADE, MADRID 2014

[3] ¿POR QUÉ LOS EMPLEADOS FELICES SON UN ACTIVO MÁS PARA LAS EMPRESAS? , HTTPS://WWW.PUROMARKETING.COM/14/29501/EMPLEADOS-FELICES-SON-ACTIVO-PARA-EMPRESAS.HTML

[4] ¿QUÉ ES EL ÍNDICE DE ROTACIÓN DE PERSONAL Y CÓMO SE CALCULA?, HTTPS://NEWS.EASYRECRUE.COM/ES/ROTACION-DE-PERSONAL

[5] CODE FOR CASE STUDY - CUSTOMER CHURN WITH KERAS/TENSORFLOW AND H2O, DR. SHIRIN ELSINGHORST, DECEMBER 12, 2018, HTTPS://SHIRINSPLAYGROUND.NETLIFY.COM/2018/12/CUSTOMER_CHURN _CODE/

[6] CONHEÇA OS 4 TIPOS DE TURNOVER, HTTPS://KENOBY.COM/BLOG/TIPOS-DE-TURNOVER/

[7] CONHEÇA OS PRINCIPAIS MOTIVOS DO ALTO ÍNDICE DE TURNOVER, HTTPS://PASSADORI.COM.BR/CONHECA-OS-PRINCIPAIS-MOTIVOS-DO-ALTO-INDICE-DE-TURNOVER/

[8] DATASET "HR_COMMA_SEP.CSV", HTTPS://WWW.KAGGLE.COM/LIUJIAQI/HR-COMMA-SEPCSV

[9] DICTIONARY CAMBRIDGE ONLINE, HTTPS://DICTIONARY. CAMBRIDGE.ORG/ES/

[10] DICTIONARY.COM ONLINE, HTTPS://WWW.DICTIONARY.COM/

[11] DICTIONARY MERRIAM WEBSTER ONLINE, HTTPS://WWW.MERRIAM-WEBSTER.COM/

[12] GUIDE TO KERAS BASICS, HTTPS://TENSORFLOW.RSTUDIO.COM/GUIDE/KERAS/HTTPS://TENSORFLOW.RSTUDIO.COM/GUIDE/KERAS/ GUIDE_KERAS/

[13] GUIDE TO THE SEQUENTIAL MODEL, HTTPS://KERAS.RSTUDIO.COM/ARTICLES/SEQUENTIAL_MODEL.HTML

[14] GETTING STARTED WITH KERAS, HTTPS://CRAN.R-PROJECT.ORG/WEB/PACKAGES/KERAS/VIGNETTES/GETTING_ STARTED.HTML

[15] MODELOS SUPERVISADOS, EDICIONES ROBLE, S.L.

[16] MODELOS CONEXIONISTAS, EDICIONES ROBLE, S.L.

[17] REGRESIÓN LOGÍSTICA, MODELOS RESTRINGIDOS DE RIDGE Y LASSO Y GRADIENTE DESCENDIENTE, EDICIONES ROBLE, S.L.

[18] AMEVA: AN AUTONOMOUS DISCRETIZATION ALGORITHM, EXPERT SYSTEMS WITH APPLICATIONS, 36, 3, PART 1, 5327-5332, 2009, 0957-4174, HTTPS://DOI.ORG/10.1016/J.ESWA.2008.06.063, HTTPS://WWW.SCIENCEDIRECT.COM/SCIENCE/ARTICLE/PII/S0957417408003801, L. GONZALEZ-ABRIL AND F.J. CUBEROS AND F. VELASCO AND J.A. ORTEGA

[19] TEOREMA DE BAYES, HTTPS://ES.WIKIPEDIA.ORG/WIKI/TEOREMA_DE_BAYES

[20] AN INTRODUCTION TO NAÏVE BAYES CLASSIFIER, HTTPS://TOWARDSDATASCIENCE.COM/INTRODUCTION-TO-NA %C3 %AFVE-BAYES-CLASSIFIER-FA59E3E24AAF

[21] ANÁLISIS CONGLOMERADOS, HTTPS://WWW.UM.ES/ESTADEMPRESA/MULTIVARIANTER/CLUS/RESUMEN_CLUS.HTML

[22] DESARROLLO Y VERSATILIDAD DEL ALGORITMO DE DISCRETIZACIÓN AMEVA, HTTPS://DIALNET.UNIRIOJA.ES/SERVLET/TESIS?CODIGO=151101

[23] A HYBRID APPROACH USING OVERSAMPLING TECHNIQUE AND COST-SENSITIVE LEARNING FOR BANKRUPTCY PREDICTION, HTTPS://WWW.HINDAWI.COM/JOURNALS/COMPLEXITY/2019/8460934/

[24] ARBOLES DE DECISIÓN CON R — CLASIFICACIÓN, HTTPS://MEDIUM.COM/@JBOSCOMENDOZA/ARBOLES-DE-DECISI%C3%B3N-CON-R-CLASIFICACI%C3%B3N-C6C583B16125

[25] WHAT IS BOOSTING?, HTTPS://CORPORATEFINANCEINSTITUTE.COM/RESOURCES/KNOWLEDGE/OTHER/BOOSTING/

[26] ADABOOST: ADABOOST CLASSIFIER, HTTPS://WWW.RDOCUMENTATION.ORG/PACKAGES/JOUSBOOST/VERSIONS/2.1.0/TOPICS/ADABOOST

[27] BAGGING — ENSEMBLE META ALGORITHM FOR REDUCING VARIANCE, HTTPS://MEDIUM.COM/ML-RESEARCH-LAB/BAGGING-ENSEMBLE-META-ALGORITHM-FOR-REDUCING-VARIANCE-C98FFFA5489F

[28] RANDOM FOREST, HTTPS://WWW.RESEARCHGATE.NET/FIGURE/FLOW-CHART-OF-RANDOM-FOREST-ALGORITHM-23_FIG3_329263557

[29] REGRESIÓN LOGÍSTICA SIMPLE Y MÚLTIPLE, HTTPS://WWW.CIENCIADEDATOS.NET/DOCUMENTOS/27_REGRESION_LOGISTICA_SIMPLE_Y_MULTIPLE#INTRODUCCI%C3%B3N

[30] QUÉ ES Y CÓMO INTERPRETAR UNA REGRESIÓN LOGÍSTICA, HTTPS://CONCEPTOSCLAROS.COM/QUE-ES-REGRESION-LOGISTICA/

[31] LA REGRESIÓN LOGÍSTICA: UNA HERRAMIENTA VERSÁTIL, HTTPS://WWW.REVISTANEFROLOGIA.COM/ES-LA-REGRESION-LOGISTICA-UNA-HERRAMIENTA-VERSATIL-ARTICULO-X0211699500035664

[32] REDES NEURONALES CON PYTHON, HTTPS://WWW.CIENCIADEDATOS.NET/DOCUMENTOS/PY35-REDES-NEURONALES-PYTHON.HTML

[33] SUPPORT VECTOR MACHINE - INSIGHTS, HTTPS://TOWARDSAI.NET/P/MACHINE-LEARNING/SUPPORT-VECTOR-MACHINE-AN-INSIGHT-CDAE000758DC