MEMORIA DE LA COMPETICIÓN EN KAGGLE

MÁSTER EN CIENCIA DE DATOS

Minería de Datos: Pre-procesamiento y clasificación.

Autores

José Ángel Díaz García, Wences, Álvaro Mateos Toset, Gerardo

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Enero de 2018

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo de la memoria veremos en detalle la introducción al problema que se pedía resolver como parte de la evaluación práctica de la asignatura **Minería de Datos: Pre-procesamiento y clasificación**, enmarcada dentro del Máster en Ciencia de Datos de la Universidad de Granada.

1.1. Introducción

La reciente incursión de las técnicas de minería de datos en actividades cotidianas o diarias constatando sus innumerables aplicaciones en diversos dominios o problemas, han propiciado que de en año en año sean más los profesionales de sectores muy dispares que deciden formarse en estas tareas que les lleven a ostentar en un futuro próximo el título de científicos de datos. Este creciente interés ha propiciado también un caldo de cultivo perfecto para diversas plataformas online que ayudan a estos profesionales en su proceso de formación siendo una de las plataformas más conocidas Kaggle [1].

Esta plataforma, se basa en proponer problemas reales de minería de datos que cualquier persona interesada en la temática puede intentar resolver, en muchos casos sin grandes necesidades de computo o máquinas potentes. Esta plataforma evalúa los resultados aportados por cada participante y evalúa un ranking en función de diversas medidas de bondad, lo que la ha convertido también en una herramienta esencial entre los docentes del ámbito de la

ciencia de datos que la usan para enfrentar a sus alumnos con problemas reales de ciencia de datos. Esta memoria, se centra por tanto en detallar el proceso llevado a cabo por los autores para resolver un problema propuesto por los profesores de la asignatura en la plataforma Kaggle. El problema en cuestión, será definido en la siguiente sección.

1.2. Definición del problema

El problema propuesto para su resolución es un problema de clasificación que debería abordarse usando técnicas de pre-procesado y cuatro algoritmos de clasificación de diversas vertientes, a saber, reglas, árboles, vecinos más cercanos y regresión logística. El grueso por tanto del problema residirá en el proceso de pre-procesado de datos ya que los algoritmos que veremos en la sección 2 no admiten muchas modificaciones en sus parámetros.

El conjunto de datos de entrenamiento está formado por 2728 observaciones con 74 variables además de la clase catalogada como **Y** y que puede tomar cuatro valores distintos (0,1,2,3). Las variables de entrenamiento, tienen todas valores numéricos continuos exceptuando las que se pueden ver en la siguiente tabla y que son strings con distintos dominios.

Variable	***Tipo***	***Dominio***
x0	string	VS, S, M, L , VL
x14	string	true / false
x17	string	way1, way2, way3...
x51	string	no / yes
x63	string	active / inactiva

Tabla 1.1: Variables no numéricas

Utilizando por tanto las variables descritas anteriormente, se crearán modelos que se usen para evaluar un conjunto de test de unas 600 observaciones sin etiquetar que serán las que Kaggle use para la obtención del valor de accuracy [1] con el que se creará el ranking final.

[1] Porcentaje de acierto.

1.3. Organización del trabajo

Tras esta introducción al problema y a la asignatura, los siguientes capítulos abordan el problema comenzando por la metodología y planificación del equipo seguida (capítulo 2), siguiendo por una detallada explicación del proceso de pre-procesado (capítulo 3) y finalizando con las conclusiones finales y un resumen del trabajo realizado (capítulo 4).

Capítulo 2

Metodología y planificación

En este capítulo se detallan los algoritmos usados durante los procesos siguientes así como la metodología de trabajo y desarrollo llevada a cabo por el equipo durante el transcurso de la práctica para favorecer mejores resultados y sobre todo optimizar tiempos.

2.1. Algoritmos utilizados

Los algoritmos propuestos para la realización de la práctica pertenecen a ramas dispares dentro del aprendizaje automático, estos son: Ripper, basado en reglas, 1-NN, basado en proximidad y similitud, Rpart basado en árboles y Glm, basado en regresión.

2.1.1. Ripper

El algoritmo RIPPER [2] es una extensión del algoritmo IREP (Incremental Reduced Error Pruning), que construye un set de reglas donde todos los ejemplos positivos son cubiertos, por lo que es bastante útil en problemas con muchos ejemplos y ruidosos. Se basa en la construcción de dos sets de reglas durante el proceso de aprendizaje, el set de crecimiento y el de podado, generando reglas con el primero.

2.1.2. Rpart

El algoritmo Rpart es un método basado en árboles [3] y que puede ser usado tanto para problemas de regresión como de clasificación. Como en todos los métodos basados en árboles, se generan particiones en los datos en función de las variables que se usan para clasificar. La diferencia de este método radica en que es recursivo y cada una de las particiones puede volver a tenerse en cuenta como parte de otra partición hasta que se alcancen los criterios de parada.

2.1.3. 1NN

El algoritmo KNN [4] está englobado dentro de los algoritmos de *Lazy Learning* y se basa en la similitud entre instancias. Hace uso de diversas medidas de distancia, como por ejemplo la euclídea, para determinar el número de ejemplos de cada clase que se encuentran a una distancia determinada del ejemplo a clasificar. Cada uno de los ejemplos de una determinada clase, corresponden con un "voto"para esa clase, finalmente la clase asignada será la que más votos tenga.

2.1.4. Glm

El algoritmo GLM [5] es una generalización del algoritmo de regresión lineal que permite ser aplicado en problemas con variables que no siguen distribuciones normales. Puede usarse tanto para clasificación como para regresión y solo acepta dos clases de salida por lo que los problemas deben descomponerse en binarios para poder aplicar GLM sobre ellos en el caso de ser utilizado para clasificación. El algoritmo, propone una combinación de técnicas típicas de regresión lineal, regresión logística y regresión de poisson.

2.2. Metodología de trabajo

Cualquier trabajo en equipo debe conllevar una buena planificación para conseguir buenos resultados en las etapas finales de cualquier proceso. Si además el ámbito de aplicación es la informática una correcta planificación

es de vital importancia para llevar el proyecto a buen fin. En esta sección veremos por tanto algunos puntos claves en la coordinación y planificación del equipo de trabajo.

2.2.1. Planificación temporal

El proyecto ha sido planificado temporalmente acorde a los siguientes puntos, tareas u objetivos:

- **Pruebas con modelos individuales**: En los primeros días, se dividieron los algoritmos de manera que cada miembro pudiera explotar los puntos fuertes y débiles de su algoritmo para el problema de manera que en puntos siguientes se potenciara solo el mejor algoritmo de los 4 para el problema en cuestión.
- **Pruebas con todos los modelos**: Una vez estudiado el algoritmo asignado para cada miembro del equipo, se llevó a cabo una bateria de pruebas en las que con base en los resultados individuales obtenidos se probaron los de los demás.
- **Optimización**: Una vez encontramos un modelo de pre-procesado y algoritmo que se comportaba bien, se explotó el mismo con diversas combinaciones y pruebas.
- **Reuniones de grupo**: Reuniones de puesta en común de resultados y debate de técnicas y vías de estudio.
- **Redacción**: Tras la finalización de la competición se abre un período de redacción del presente documento.

En la figura 2.1, puede verse un croquis sobre el calendario de los meses de enero-marzo, en el que se detallan las distintas tareas que se han detallado en puntos anteriores.

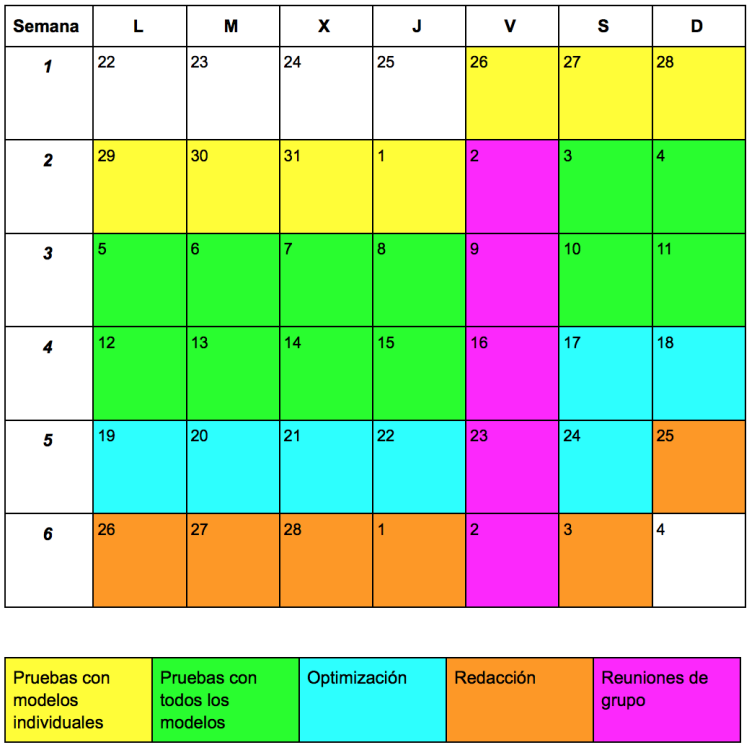

Figura 2.1: Croquis de la planificación seguida.

2.2.2. Desarrollo

Para el desarrollo del código se ha usado un repositorio privado para el control de versiones en GitHub, con ramas para cada uno de los miembros del equipo que eran fusionadas al realizar algún avance por parte de cada uno de los implicados.

Dado que el proceso de pre-procesado de datos es incremental y secuencial, se ha llevado a cabo un desarrollo basado en **soluciones intermedias**. Con este método, se guardan soluciones intermedias que son etiquetadas con el pre-procesado aplicado de manera que estas pueden ser usadas por otro miembro del equipo sin necesidad de tener que aplicar técnicas que otro miembro ya uso. Este método ha permitido ahorrar en tiempos de ejecución así como tener constancia de que técnicas estaban dando mejor resultados que otras para su posterior refinamiento.

Capítulo 3

Proceso exploratorio y pre-procesado

En este capítulo veremos el proceso seguido para afrontar y resolver el problema definido en puntos anteriores. El capítulo comienza detallando el proceso exploratorio inicial para continuar con el grueso de la memoria, la especificación de los procesos de pre-procesado llevados a cabo y la solución final aportada.

3.1. Proceso exploratorio

En esta sección detallamos el proceso exploratorio seguido para obtener más información del problema y de los datos que tenemos entre manos. Los pasos seguidos en este proceso serían:

1. **Tipos de datos y dimensiones**: El primer paso para enfrentarnos a los datos era conocer la dimensionalidad y el tipo de datos. Por ello, hicimos uso de los comandos **describe** y **str**, para comprobar como eran estos datos y sus distribuciones. Acotamos así las variables numéricas y los strings o factores que vimos en la tabla 1.1.

2. **Valores perdidos**: Al usar estudiar las distribuciones de los datos en el punto anterior descubrimos la existencia de valores perdidos en algunas variables. Para ver si este problema era muy acentuado se creó

una función que nos ofrece el número de valores perdidos de un dataset por variables con diversos estadísticos. Tras obtener estos valores se representaron gráficamente para ver cuantos eran estos valores perdidos en función de la variable y el conjunto de train (figura 3.1) o test (figura 3.2).

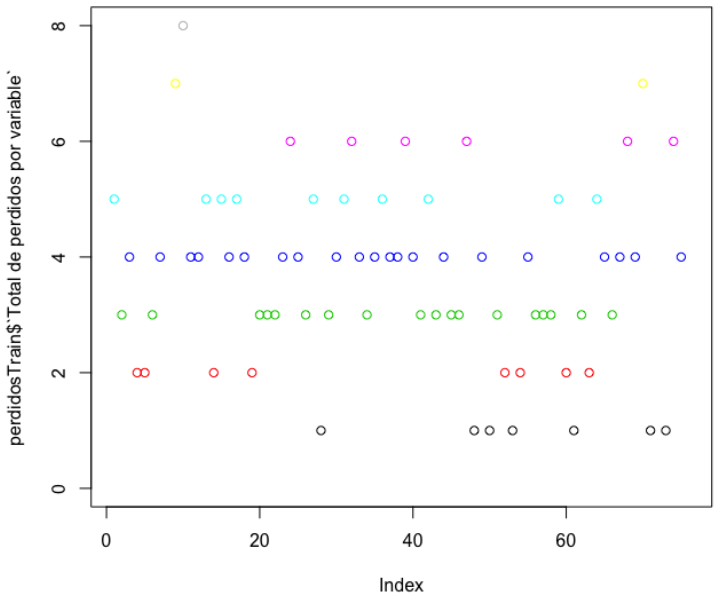

Figura 3.1: Distribución de valores perdidos en train.

Para comprobar la distribución de valores perdidos y si se asemejan en número en training y test, también se llevo a cabo un histograma (3.3) conjunto que representa los valores perdidos en cada una de las particiones de datos. Este gráfico nos llevo a comprobar que los valores perdidos **no siguen patrones** sino que son valores perdidos que parecen haber sido añadidos aleatoriamente o pertenecer a fallos en la toma de datos.

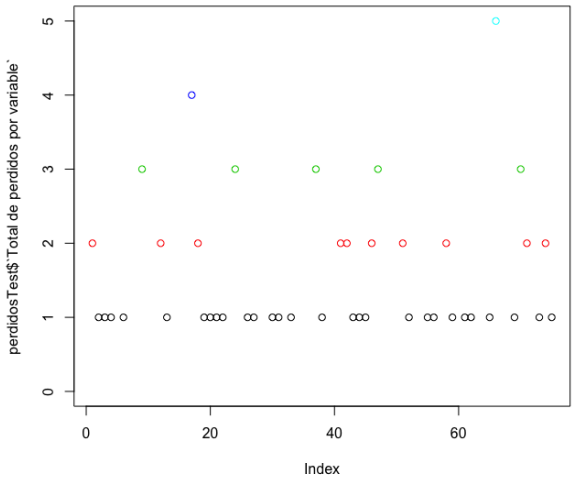

Figura 3.2: Distribución de valores perdidos en test.

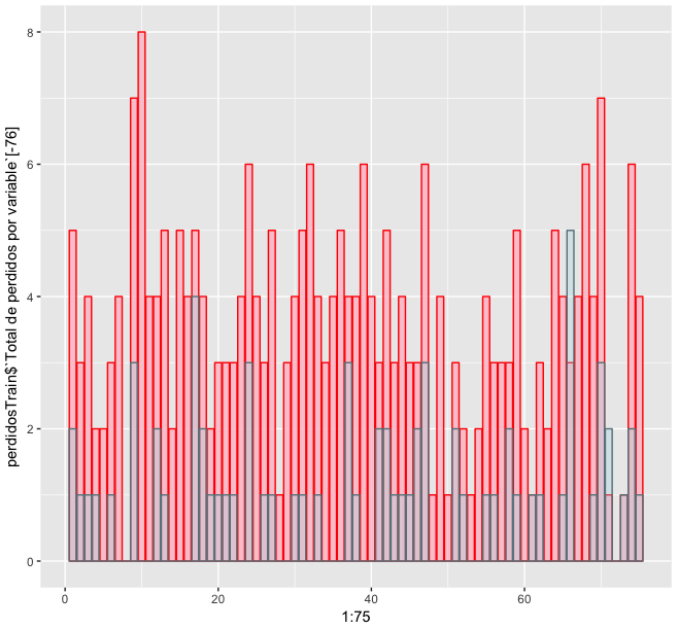

Figura 3.3: Distribución de valores perdidos en train y test.

3. **Correlaciones**: El tener tantas variables (75) y tanta presencia de valores perdidos hizo interesante la obtención de correlaciones para comprobar si podemos eliminar variables en pos de otras o imputar los valores perdidos con los de otra variable muy correlada. Para ello, usamos la función **corrplot**. El resultado podemos verlo en la figura 3.4 y descubrimos que la variable **x41** tiene correlacion de 1 con la **x48** siendo una el resultado del producto de la otra.

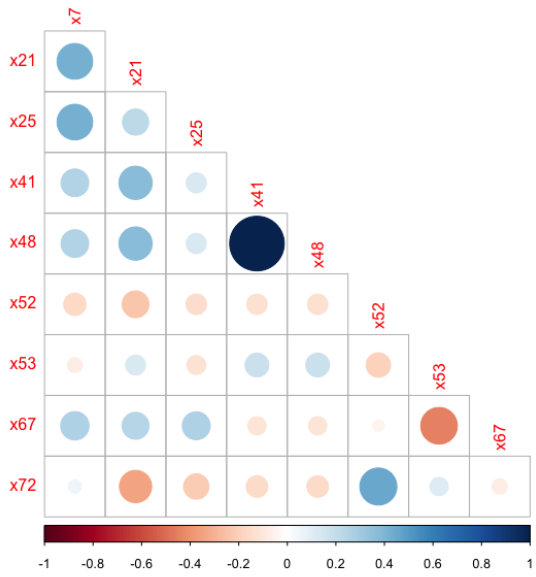

Figura 3.4: Correlación de variables.

4. **Outliers**: Dado el volumen del problema, se llevó a cabo un estudio de outliers univariate básico basado en distancia intercuartil (IQR). Para que este proceso obtenga buenos resultados, se escalaron las variables y se analizaron solo aquellas cuyo dominio es continuo. Los resultados para las variables 1:30 pueden verse en el gráfico 3.5 mientras que las variables 31:70 pueden verse en el gráfico 3.6.

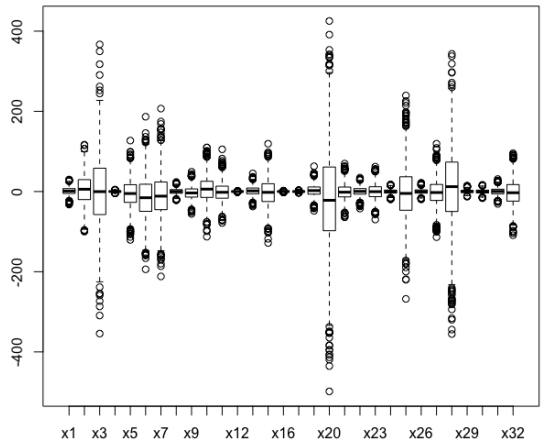

Figura 3.5: Boxplot de la primera mitad.

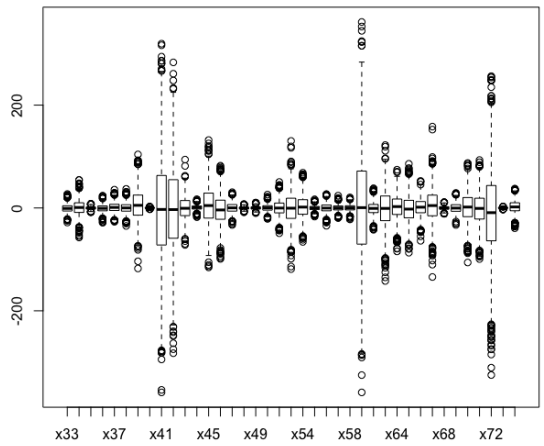

Figura 3.6: Boxplot de la segunda mitad.

Tras el análisis de los boxplot podemos concluir que hay un gran número de outliers, lo que nos lleva a pensar que probablemente estaremos ante

un **dataset ruidoso** por lo que se deberá de probar técnicas de limpieza de ruido.

5. **Distribución clases**: Por último, en nuestro proceso de análisis exploratorio, se realizó un gráfico de distribución de variables para comprobar si estamos ante un problema de clases balanceadas o en su defecto no balanceadas. El resultado puede verse en el gráfico 3.7, donde queda constatado que estamos ante un problema donde la clase 0 y la 1 están en clara desventaja por lo que habrá que usar técnicas de **oversampling** o **undersampling**.

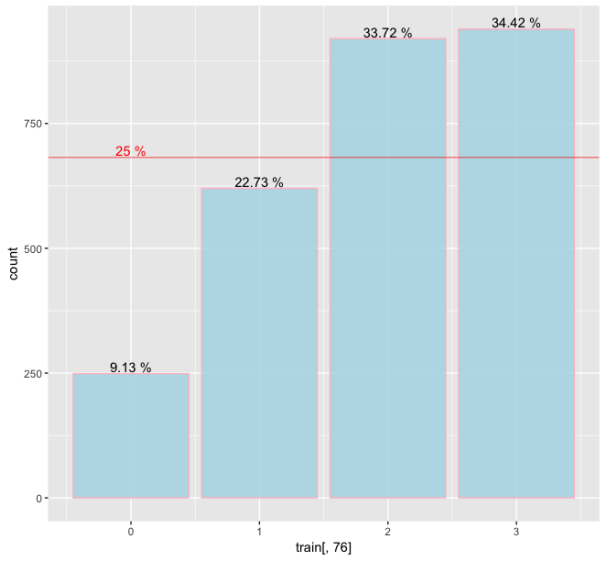

Figura 3.7: Distribución de clases.

3.2. Pre-procesado

3.2.1. Valores numéricos

:

3.2.2. Imputación de valores perdidos

: Uno de los primeros problemas a los que nos hemos enfrentado para comenzar el preprocesamiento de los datos ha sido la existencia de valores perdidos (NAs) en el dataset. Estos valores perdidos se encuentran en variables tanto de tipo numérico como categórico, por lo que necesitamos trabajos con dos formas distintas de imputación de datos, una para las numéricas y otra para las categóricas.

Mice

Para seleccionar el método más adecuado de imputación, hemos tenido en cuenta los paquetes existentes que permiten realizar dicha tarea como ?MICE?, el cual, nos permite imputar distintos tipos de variable, mediante distintos métodos como PMM (Predictive Mean Matching), ejemplos aleatorios, medias, regresiones lineales bayesianas, etc. Sin embargo, a la hora de aplicar este paquete hemos observado unos tiempos de ejecución muy altos, junto a unos resultados muy pobres, lo cual quedó patentado a la hora de utilizar otros métodos que describiremos a continuación.

Media y Moda

Ante esta perspectiva, se ha llevado a cabo una declaración propia de funciones que realicen una imputación de los datos numéricos mediante la media, y de los datos categóricos mediante la moda, cuyo funcionamiento queda reflejado en el script en formato .R entregado.

Una vez confeccionadas las funciones para realizar la imputación de valores perdidos, ha sido necesario comprender cómo funcionan los distintos algoritmos que debemos aplicar (1NN, GLM, RIPPER y RPART) a los datos preprocesados, para confeccionar los conjuntos adecuados para cada uno.

El algoritmo KNN, en nuestro caso 1NN, requiere que los datos no sean categóricos por lo que, no se ha realizado imputación de valores categóricos mediante la moda, sino que, en un primer momento, se ha realizado una imputación de los valores numéricos por media y no se han tenido en cuenta las variables categóricas.

Los resultados, sin embargo, a pesar de representar una mejora frente a MICE no eran buenos en términos generales.

KNN

Las imputaciones pobres resultantes de los métodos anteriores nos han llevado a realizar un método alternativo que ha resultado más eficaz para imputar valores, mediante el uso de KNN. Mediante el cálculo de los vecinos más cercanos, en este caso instancias, hemos realizado una función que imputa tanto el valor de las variables categóricas como numéricas.

El resultado ha sido un aumento en la calidad de la predicción en todos los algoritmos, lo que nos ha llevado a decantarnos por este método, el cual, en líneas generales, ha permitido una mejor actuación del resto de técnicas implementadas.

3.2.3. Ruido

Hemos tenido en cuenta la posible existencia de valores que podemos considerar ?ruidosos?, a simple vista outliers por casos atípicos, o incluso inconsistentes y que pueden estar relacionados con errores en los instrumentos de medición, en las entradas de datos, etc... Como medida de choque contra el ruido existente en los datos, hemos recurrido a la función IPF (Iterative Partitioning Filter) que se encuentra disponible para su utilización en R mediante el paquete NoiseFiltersR.

El funcionamiento de IPF consiste en ir eliminando las instancias que considera ?ruidosas? en múltiples iteraciones, hasta que el porcentaje de las instancias que considere ruidosa en cada una de las interacciones sea menor que un porcentaje p del tamaño original de los datos. Adicionalmente, hemos añadido el argumento s que establece de forma fija el número de interacciones que se van a realizar sobre los datos.

El resultado de la utilización de IPF no ha sido homogéneo, ya que ha habido una gran variabilidad en los distintos casos en los que se ha empleado. Uno de los factores que influye en los resultados de IPF es el método de selección de características empleado anteriormente, ya que la aplicación de Boruta provoca un sobre aprendizaje muy grande sobre los datos tanto en KNN como GLM. El caso contrario sucede al realizar una selección de variables con

Random forest previamente, siempre y cuando se aplique SMOTE, debido a que IPF desbalancea mucho las clases en los datos, dejando muy mermada la clase 0. Con esta combinación de IPF + SMOTE el modelo mejora tanto en KNN como en GLM.

Como estrategia final, hemos replicado los diferentes casos, aplicando o no IPF y observando cuando producía sobre-aprendizaje y cuando eliminaba instancias que realmente aportaban ruido al conjunto de datos, para lo cual ha sido necesario aplicar distintos métodos que aseguren un balanceo de clases y un mínimo de ejemplos de cada clase.

3.2.4. Selección de características

Este apartado va a tratar sobre los procesos y técnicas que hemos seguido para realizar selección de características. La selección de características o atributos consiste en seleccionar un subconjunto relevante de características para poder construir un modelo.

Los motivos por los que se realizan este tipo de técnicas son:

- Simplificación del modelo.
- Tratar de eliminar la maldición de la dimensionalidad.
- Intento de minimizar el overfitting.
- Reducción de tiempo de aprendizaje.

Cuando se aplican técnicas de este tipo hay que tener en cuenta la interpretabilidad del modelo. Se debe conseguir un equilibrio a la hora de que el modelo sea interpretable y se reduzcan las características. En el caso particular de nuestro problema no tenemos ese requerimiento puesto que no tenemos información sobre el significado de las variables. La idea de aplicar estas estrategias es que el conjunto de datos puede contener variables que sean redundantes o poco relevantes para explicar el problema. Los conceptos de redundancia y relevancia son distintos, puesto que, una característica relevante puede ser redundante en la presencia de otra relevante. Para el desarrollo de la práctica planteamos 2 estrategias para la selección de características:

1. Métodos Wrapper: La idea de los algoritmos wrapper para la selección de características es generar todos los subconjuntos posibles y determinar la relevancia de ellos a través del uso de filtros. En nuestro caso hemos seleccionado el algoritmo Boruta del paquete Boruta para realizar el estudio pertinente.

2. Métodos de filtro: La idea de este tipo de métodos es fijar la importancia de cada una de las variables y posteriormente ir eliminando aquellas menos relevantes.Para el desarrollo de este tipo de filtro hemos optado por realizar una clasificación por randomforest y posteriormente hemos visto la importancia de cada uno de atributos a la hora de realizar la clasificación. Estos se ordenaron y se fueron viendo cuál era el resultado óptimo. Finalmente se decidió eliminar las 25 menos relevantes. Con la eliminación de estas se obtuvieron los resultados más relevantes.

3.2.5. Oversampling y Undersampling

Al tratarse de una conjunto de datos desbalanceado, como se observó en el análisis exploratorio que se hizo sobre los datos, se vió fundamental la introducción de técnicas de oversampling y undersampling para ajustar la distribución de los datos y poder mejorar en los resultados del estudio. En este sentido vamos a aplicar técnicas de **oversampling y undersampling** para buscar el equilibrio entre clases que nos permitan desarrollar un modelo que se ajuste a la realidad.

Oversampling

Como indica su nombre las técnicas de oversampling consiste en generar nuevas muestras de las clases minoritarias. Para llevar a cabo este método valoramos distintos algoritmos que pasaremos a explicar.

- **Synthetic Minority Over-sampling Technique (SMOTE)**: Esta técnica consiste en potenciar la clase minoritaria a través de la generación de instancias de esa clase. Para la generación de estas obtenemos una instancia de la clase minoritaria y buscamos los k-vecinos más próximos y se generan instancias viendo la distancia del elemento seleccionado y sus vecinos.Para la aplicación de esta técnica usamos la

librería unbalanced y dentro de las funciones el que hemos utilizado ha sido ubSMOTE.

- **Random Oversampling (ROS)**: Esta técnica consiste en generar muestras de la clase minoritaria de forma aleatoria. Para su uso utilizamos la librería ROSE y la función ovun.sample.

- **ROS One Versus All (OVA**: Una variante del anterior, para hacer que las nuevas instancias estén generadas atendiendo a los datos se va a aplicar ROS sobre la clase minoritaria pero atendiendo a su diferencia con respecto al resto de clases de forma individual.Con ello pretendemos que la clase 0 no crezca tanto y haya que aplicar luego otras técnicas.

- **ROS clase0**: De los casos anteriores hemos visto que el mejor comportamiento lo tiene cuando usamos técnicas de ROS. Por ello decidimos hacer un estudio de la medida concreta en que podíamos aumentar la clase minoritaria de forma aleatoria para obtener mejores resultados. Se planteó la posibilidad de duplicar las instancias minoritarias obteniéndose, de esta forma una mejora significativa en los resultados obtenidos

- **ROS clase0 y clase1**: Tras el estudio anterior se detectó que una vez aplicado el oversampling sobre la clase minoritaria esta pasaba a ser la clase 1. Por este motivo parecía interesante pensar si podíamos aplicar el mismo comportamiento sobre la clase 1 para terminar de equilibrar el problema. Por este motivo hicimos ROSen un segundo nivel, en este caso sobre la clase 1. La forma en la que lo llevamos a cabo fue elegir de forma aleatoria un 25 % de los elementos de la clase 1 y añadirlos al conjunto de los datos

Undersampling

Los métodos de undersampling a diferencia del anterior se centran en las clases mayoritarias para eliminar. Para nuestro conjunto de datos planteamos que las técnicas más interesantes podrían ser undersampling focalizado (FUS) o Tomeklinks.

- **FUS Extended Nearest Neighbor ENN**: Este tipo de técnicas se centran en eliminar instancias de la clase mayoritaria situadas entre

fronteras de dos clases. Para su aplicación seleccionamos el algoritmo ubENN de la clase unbalanced. El problema de este algoritmo en nuestro problema será, que al no existir fronteras bien definidas, no podrá hacer una selección apropiada

- **Tomeklinks**: En alguna bibliografía se inlucye Tomeklinks como un algoritmo de tipo FUS. Podemos usar este tipo de algoritmos para eliminar ejemplo de la clase mayoritarias de las fronteras así como para eliminación de ruido.

3.3. Técnicas de clasificación

3.4. Solución aportada

Tras todas las pruebas realizadas la que mejor resultado nos ha aportado ha pasado por eliminar una de las dos variables altamente correlacionadas, habiendo imputado anteriormente los valores perdidos entre ellas, imputar todos los valores perdidos de todas las demás variables mediante K-NN (tanto para train como para test), transformar las variables categóricas a numéricas, duplicar todas las instancias con clase 0 (debido al desbalanceo) y, aleatoriamente, una cuarta parte de las de clase 1 y aplicar 1-NN escogiendo desechando las 25 peores variables marcadas mediante Random Forest.

3.4.1. Propuesta descartada

Propuesta con mejor resultado descartada La solución que mejor resultado nos aportó en Kaggle, pero que finalmente decidimos descartar debido a que nos pareció que incumplía en cierto modo las normas de la competición, así como que produciría finalmente un sobreajuste con el 40 % del test final, pasaba por retroalimentar el modelo en base a las predicciones obtenidas, inspirándonos en la filosofía de la de redes neuronales. De este modo, predecimos test con nuestro mejor modelo, asignamos estos valores a test, a este conjunto le aplicamos IPF y lo unimos al conjunto de train empleado para el anterior modelo y predecimos. Repitiendo este modelo además es posible ir mejorando el resultado, con el consecuente sobreaprendizaje que ello conlleva.

Capítulo 4

Conclusión

En este capítulo se concluye el trabajo aportando un pequeño resumen al proceso realizado así como a los objetivos alcanzados con el mismo y la solución final aportada. Se finaliza el capítulo, con una seríe de valoraciones personales sobre el proceso llevado a cabo.

4.1. Resumen

4.2. Conclusiones finales

Bibliografía

[1] Web del portal Kaggle https://www.kaggle.com. Accedido el 1 de marzo de 2018.

[2] Johannes Furnkranz, Gerhard Widmer: Incremental Reduced Error Pruning. *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pp. 70-77, New Brunswick, NJ, 1994. Morgan Kaufmann.

[3] L. Breiman, J.H. Friedman, R.A. Olshen, , and C.J Stone. *Classification and Regression Trees*. Wadsworth, Belmont, Ca, 1983

[4] T.M. Cover, P.E.Hart. Nearest Neighbor Pattern Classification. *IEEE* (1967)

[5] Nelder, John; Wedderburn, Robert (1972). Generalized Linear Models. *Journal of the Royal Statistical Society*. 135 (3): 370?384.