Aprendizaje Automático (2019)
Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada

Proyecto Final

Ignacio Aguilera Martos, Luis Balderas Ruiz
nacheteam@correo.ugr.es, luisbalderas@correo.ugr.es

7 de junio de 2019

Índice

Índice de figuras

Índice de tablas

1. APS Failure Scania Trucks Classification

1.1. Introducción

Nos encontramos ante un problema de clasificación binaria (clase positiva y negativa) en la que la clase positiva consiste en el fallo o mal funcionamiento de un componente específico del sistema APS para la compañía de camiones Scania. Por su parte, la clase negativa está formada por camiones con fallos en componentes que no están relacionado con el sistema APS. En lo que se refiere al dataset en sí, el conjunto de entrenamiento está formado por 60000 instancias y 171 características. El test tiene 16000 muestras. Sabemos que los nombres de los atributos han sido anonimizados por motivos de privacidad. En el presente documento explicamos con detalle cada uno de los pasos que hemos seguido en el estudio y diseño de un sistema inteligente para la clasificación con varios modelos, a saber, un modelo lineal (SGD), una Máquina de Soporte de Vectores, un modelo basado en Boosting (AdaBoost, concretamente) y Random Forest. Previo uso de los modelos, hemos hecho un profundo estudio de los datos que nos han llevado a modificarlos y tratarlos para mejorar los resultados, entre lo que destaca el tratamiento de valores perdidos con la imputación de la mediana, eliminación de estancias poco relevantes, oversampling con SMOTE para compensar el gran desbalanceo y análisis factorial para reducir la dimensionalidad en las características.

2. Preprocesamiento

2.1. Análisis explotario y valores perdidos

Nos encontramos ante un problema que tiene una gran cantidad de datos. En principio, este hecho es una ventaja, dado que la capacidad de generalización aumenta enormemente. Sin embargo, uno de los grandes retos ha sido, por un lado, el gran desbalanceo entre las clases y, por otro, la gran cantidad de valores perdidos e inconsistencias en los datos. Nos ocupamos de esto segundo en primer lugar. El análisis exploratorio de los datos nos hizo llegar a la conclusión de que, de entre las pocas instancias que forman parte de la clase positiva (esto es, con un fallo en el sistema APS), los valores de las variables estaban extraviados o, si no era así, más parecían outliers que otra cosa. En seguida nos dimos cuenta de que, desde un punto de vista físico tiene sentido, dado que la avería en un componente puede hacer que el comportamiento en los demás sea errático y, como consecuencia, puede generar observaciones de lo más variopintas. No obstante, nos dedicamos a eliminar los valores perdidos y le dimos validez total a los datos de los que disponemos.

En primer, vimos que había características que tenían más de un 70 % de sus valores como NA. Decidimos eliminarlas, ya que iban a entorpecer enormemente el trabajo de

los modelos (en alguno de ellos, incluso son incompatibles). Eliminamos por tanto las características 2,75,76,77,78,79 y 113, por lo que pasamos de 171 a 164. En lo que se refiere a instancias, observamos que gran cantidad de ellas también estaban pobladas de valores perdidos. Teniendo en cuenta el gran desbalanceo de las clases, eliminamos aquellas instancias con etiqueta negativa que tuvieran más de un 15 % de sus entradas como NA, de forma que pasamos de 60000 instancias a 55964.

A partir de ahí, nuestro conjunto de datos es más tratable. Sin embargo, aunque más dispersos, aún hay muchos valores perdidos con los que los modelos no pueden lidiar. Para solucionarlo, consultando la bibliografía, utilizamos dos tipos de imputación de valores perdidos: imputación de la mediana y de la media. La mediana nos ha dado mejores resultados en general, por lo que continuamos nuestro desarrollo con la mediana.

2.1.1. Desbalanceo de clases

Sin duda alguna, uno de los grandes problemas a los que nos enfrentamos cuando tratamos este conjunto de datos es el gran desbalanceo entre las clases. Hicimos una primera visualización de los datos observando el desbalanceo pero, a su vez, leve solapamiento (lo que a priori es una ventaja).

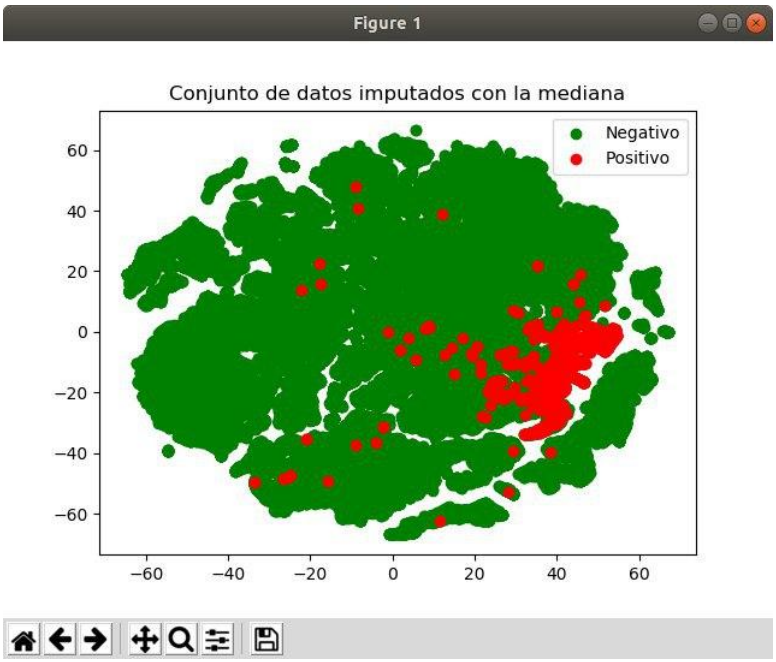

Figura 2.1: Representación de las instancias tras la imputación de VP con la mediana

Sin embargo, a pesar de la separación de las clases éramos incapaces de generar buenos

resultados con los modelos y los clasificadores siempre etiquetaban las instancias de test como pertenecientes a la clase negativa. Esto daba lugar a un buen accuracy, pero las demás medidas que tenemos en cuenta, como son recall o f1-score eran nefastas.

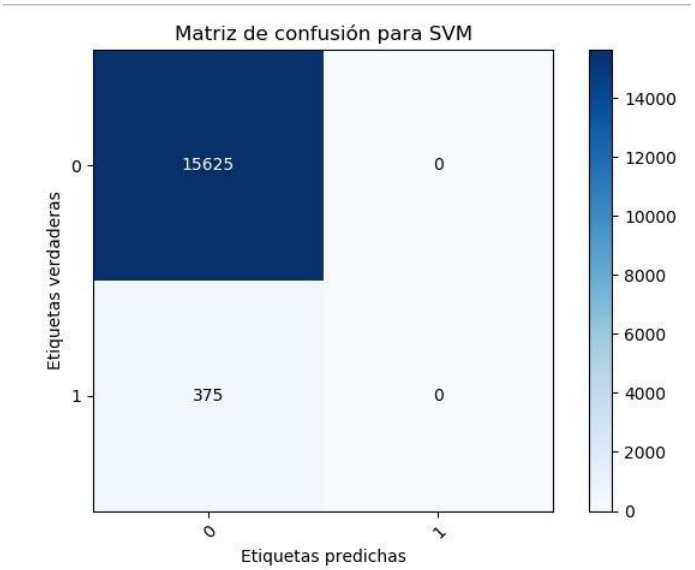

Figura 2.2: Matriz de confusión generada por SVM /RF/SGD/Red Neuronal hasta el momento

Por tanto, a pesar de la buena disposición de los datos, no se obtenían buenos resultados. A la vista de los mismos, nos decantamos por utilizar técnicas de oversampling para equilibrar el número de instancias de cada clase. En particular, elegimos SMOTE ([3]) como herramienta. Tras su aplicación, contamos finalmente con 109928 instancias de cada clase, apareciendo la siguiente disposición de datos:

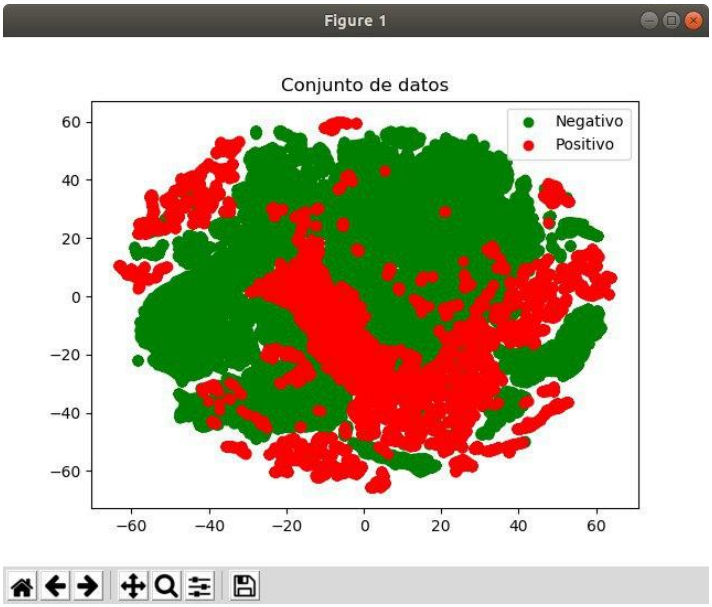

Figura 2.3: Disposición del dataset tras SMOTE

Como se puede apreciar, hemos perdido la localidad de las clases, aunque, como se verá, va a ser muy positivo.

2.1.2. Selección de características

Tras la eliminación de características por poseer demasiados valores perdidos, poseemos 164. No nos parece un número adecuado para utilizar PCA (más recomendado para cuando hay muchas más variables, ya que el sacrificio de la interpretabilidad es menos costoso) así que nos decantamos por otras técnicas de estadística multivariante, como es análisis factorial ([5],[6],[2], [4]). Análisis factorial sí nos ayuda a darle interpretabilidad al modelo y a encontrar qué características son las más determinantes. Reducimos así de 164 a 151. Sin embargo, los resultados en los modelos nos harán comprobar que, a pesar de mejorar el accuracy, se empeora la clasificación de las instancias de la clase positiva, como se puede ver en la siguiente comparación entre matrices de confunsión para Random Forest:

Verd \Pred	0	1
0	56160	2840
1	77	923

Tabla 2.1: Matriz de confusión sin FA

Verd \Pred	0	1
0	57077	1923
1	113	887

Tabla 2.2: Matriz de confusión con FA

Por tanto, como nuestro objetivo es maximizar todas las medidas, especialmente recall (la clase minoritaria sigue siendo, de forma natural, la clase positiva), optamos por no aplicar de forma definitiva análisis factorial. Después de explorar estas dos técnicas, llegamos a la conclusión de que el tamaño del dataset hace que el número de características no sea muy elevado, por lo que acabamos por no seleccionar ningún subconjunto de ellas.

2.1.3. Normalización

En el análisis exploratorio observamos que el rango de las variables es absolutamente dispar. Todos los clasificadores basados en instancias (que incorporan la distancia) necesitan de un rango de variables equivalente para poder dar el peso adecuada a cada una. Por eso, realizamos una normalización estándar a los datos.

3. Modelos y clasificación

3.1. Error de generalización y dimensión de VC

Pueden calcularse dos cotas ([1]). Trato primera la de E_{in}. Sea $h \in H$, fija, $f(x)$ la función objetivo (desconocida), D el conjunto de entrenamiento de tamaño N. Podemos escribir la desigualdad de Hoeffding como sigue:

$$P(D : |E_{out}(h) - E_{in}(h)| > \epsilon) \leq 2e^{-2\epsilon^2 N} \quad \forall \epsilon > 0$$

Llamando $\delta = 2e^{-2\epsilon^2 N}$, tenemos

$$P(D : |E_{out}(h) - E_{in}(h)| > \epsilon) \leq \delta \Rightarrow P(D : |E_{out}(h) - E_{in}(h)| < \epsilon) \geq 1 - \delta$$

Podemos escribir, por tanto, la siguiente desigualdad:

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{1}{2N} \log \frac{2}{\delta}}$$

con probabilidad al menos $1 - \delta$ en D.

Considerando una clase de funciones finita, la expresión se convierte en

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{1}{2N} \log \frac{2|H|}{\delta}}$$

Sin embargo, el tamaño de la clase de funciones ahora entra en juego y puede hacer menos útil la cota. Para poder garantizar una buena cota, surge la teoría de la generalización y la Dimensión de Vapnik-Chervonenkis (VC). La cota a utilizar es la siguiente:

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{8}{N} \log \frac{4((2N)^{dVC} + 1)}{\delta}}$$

Para el caso de E_{test}, cuando afirmamos que E_{test} es un estimador de E_{out}, estamos afirmando que E_{test} generaliza muy bien E_{out}. En el caso anterior, intentábamos buscar la función que minimizaba E_{in}. Ahora, sin embargo, ya tenemos una función fija y queremos simplemente estimar el error. Por tanto, la desigualdad de Hoeffding con una hipótesis es suficiente para el caso de conjunto de test.

$$E_{out}(h) \leq E_{test}(h) + \sqrt{\frac{1}{2N} \log \frac{2}{\delta}}$$

En cada caso, utilizaremos la que más convenga. En los modelos en los que la dimensión de VC sea excesivamente elevada (debido a la complejidad, como las redes y los basados en árboles) utilizaremos exclusivamente la que involucra a E_{test}.

3.2. Gradiente descendente estocástico

El primer modelo que utilizamos es Gradiente Descendente Estocástico. Establecemos un número máximo de iteraciones de 10000 y una tolerancia de 1^{-6}. Se produce la convergencia tras 46 épocas con los siguientes resultados:

Score	0.97105
Precision	0.9883861
Recall	0.97105
F1-Score	0.97734082

Tabla 3.1: Resumen de las medidas obtenidas con SGD

Además, obtenemos la siguiente matriz de confusión:

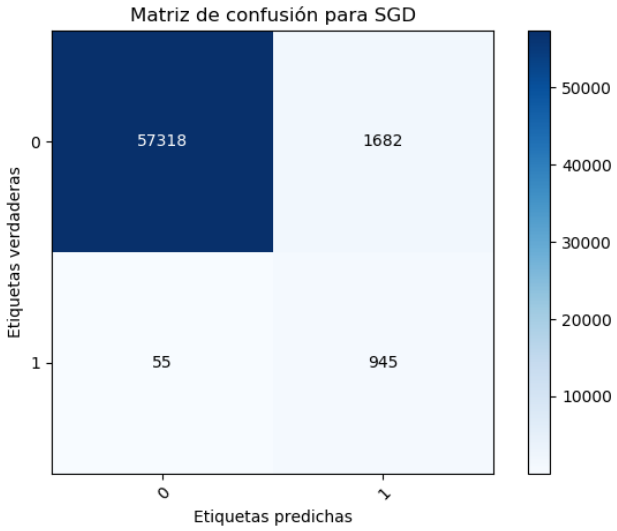

Figura 3.1: Matriz de confusión para SGD

y la siguiente curva ROC:

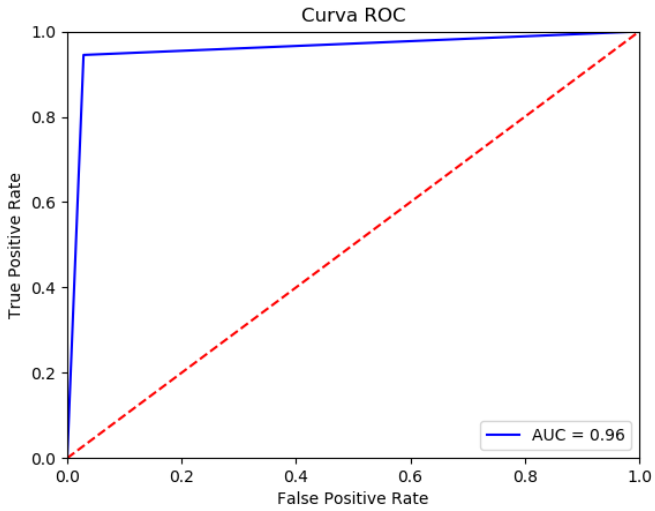

Figura 3.2: Curva ROC para SGD

Como se puede ver, los resultados son altamente satisfactorios. Además, con un problema que posee un volumen de datos tan grande, el tiempo de computación es un factor a tener en cuenta, y, en efecto, SGD apenas tarda un minuto en concluir.

Como se trata de un modelo lineal, dimensión de VC = 3, por lo que la cota de genera-

lización a través de E_{in} vale:

$$E_{out} \leq$$

Con respecto a la segunda forma de calcularlo (con E_{test}),

$$E_{out} \leq$$

3.2.1. AdaBoost

El segundo modelo empleado es la implementación de Boosting conocida como AdaBoost con un parámetro de estimadores de 100.

Score	0.9694
Precision	0.98717
Recall	0.9694
F1-Score	0.976038

Tabla 3.2: Resumen de las medidas obtenidas con AdaBoost

Como podemos ver los resultados obtenidos son satisfactorios, vamos a ver ahora la matriz de confusión para tener una idea mejor y más completa del resultado.

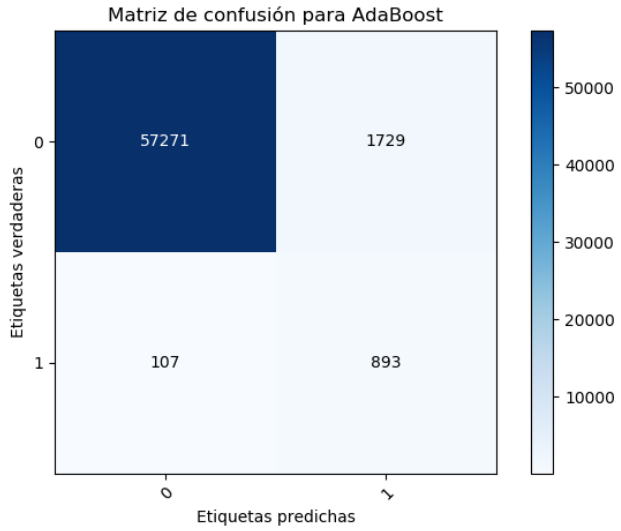

Figura 3.3: Matriz de confusión para AdaBoost

Analizando ahora el resultado podemos ver que la variable más determinante, que en

nuestro caso es acertar las etiquetas 1 (los fallos en frenos) es menor que en el caso de SGD, es decir en SGD obtenemos una mayor tasa de acierto en los verdaderos positivos de la clase 1 por lo que de momento nos interesa más dicho modelo que Boosting.

Podemos visualizar asimismo la curva ROC.

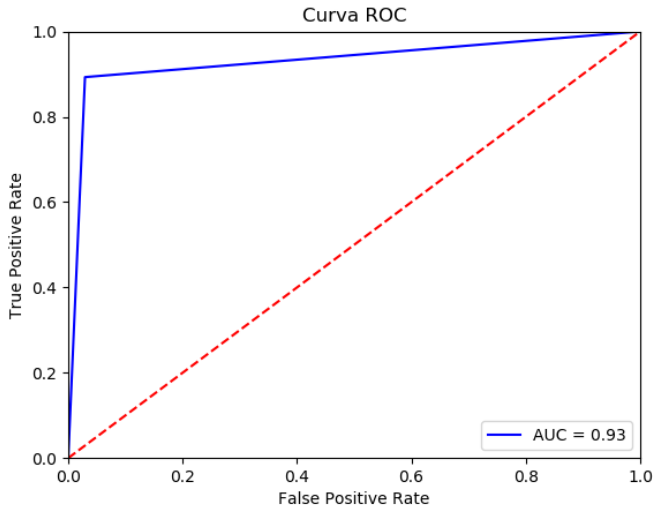

Figura 3.4: Curva ROC para AdaBoost

3.2.2. Red Neuronal (Perceptrón multicapa)

El tercer modelo empleado ha sido una Red Neuronal, en concreto un Perceptrón Multicapa. Los parámetros empleados en el modelo han sido 3 capas con 100 estimadores simples por capa, con un número máximo de iteraciones de 10000. La convergencia del modelo se produce mucho antes de estas iteraciones máximas, en concreto en la iteración 63. Los resultados han sido:

Score	0.98575
Precision	0.992269
Recall	0.98575
F1-Score	0.98782

Tabla 3.3: Resumen de las medidas obtenidas con Perceptrón Multicapa

Los resultados generales como podemos ver parecen tremendamente prometedores, vamos a ver la matriz de confusión:

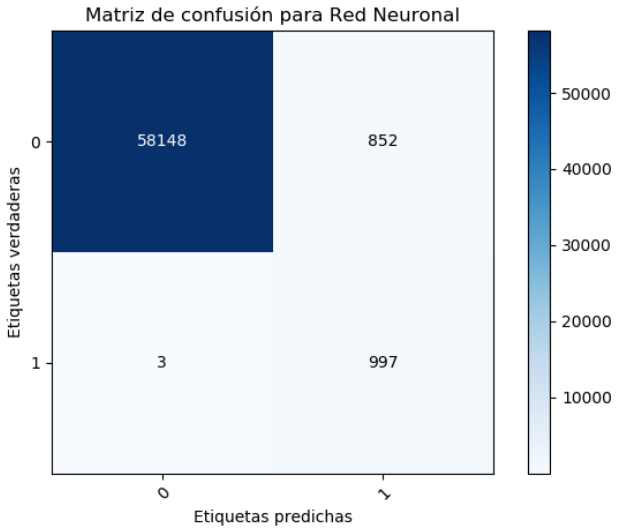

Figura 3.5: Matriz de confusión para Perceptrón Multicapa

Como podemos ver en este caso se producen 997 aciertos y sólo 3 fallos con lo que hemos conseguido superar incluso el resultado obtenido por Gradiente Descendente Estocástico con el modelo de perceptrón multicapa.

Veamos la curva ROC:

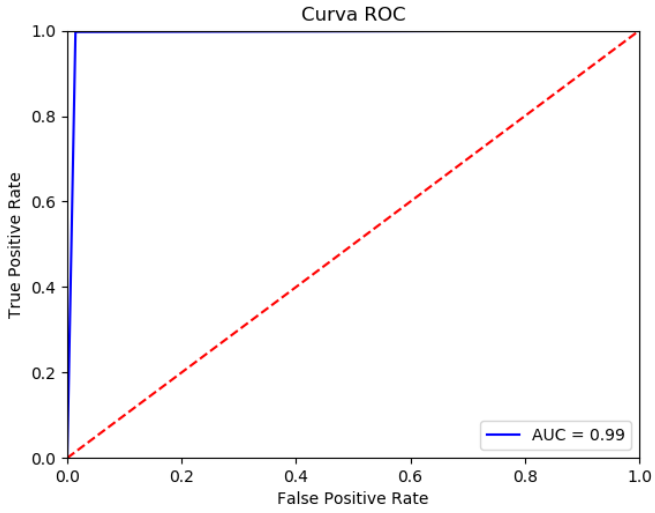

Figura 3.6: Curva ROC para Perceptrón Multicapa

3.2.3. Random Forest

El cuarto modelo empleado ha sido un Random Forest. Los parámetros del modelo han sido obtenidos a través de un GridSearch para obtener el mejor número de estimadores o clasificadores simples. En nuestro caso el mejor valor obtenido es con 189 estimadores. Los resultados han sido:

Score	0.9864
Precision	0.9925
Recall	0.9864
F1-Score	0.9883

Tabla 3.4: Resumen de las medidas obtenidas con Random Forest

Los resultados generales como podemos ver parecen tremendamente prometedores al igual que nos ha ocurrido con la red neuronal, vamos a ver la matriz de confusión:

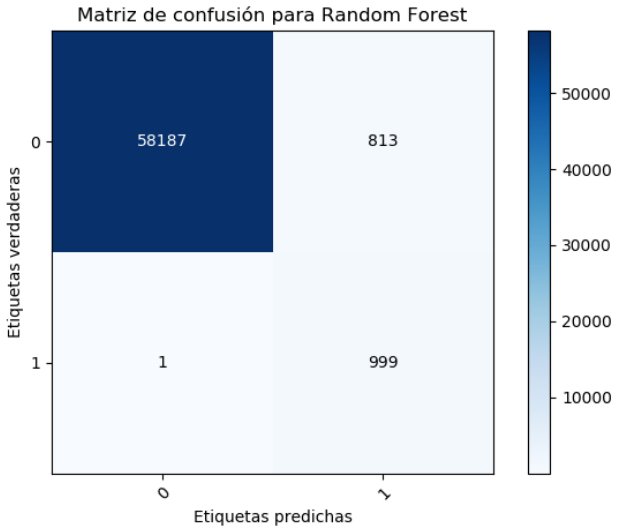

Figura 3.7: Matriz de confusión para Random Forest

Como podemos ver en este caso se producen 999 aciertos y sólo 1 fallos con lo que hemos conseguido superar tanto el resultado obtenido por Gradiente Descendente Estocástico como el modelo de perceptrón multicapa con lo que Random Forest es actualmente el modelo que mejores resultados ha obtenido.

Veamos la curva ROC:

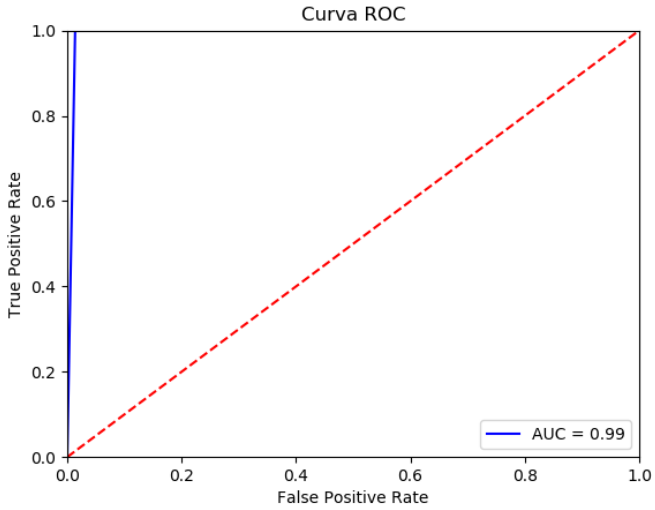

Figura 3.8: Curva ROC para Random Forest

3.2.4. SVM

El último modelo empleado ha sido SVM. Los parámetros del modelo han sido obtenidos a través de un GridSearch para obtener los mejores valores de gamma y C tal y como se indica en el PDF. En nuestro caso el mejor valor obtenido de gamma ha sido 0.01 y C 0.95 con el kernel RBF. Los resultados han sido:

Score	0.97665
Precision	0.98972
Recall	0.97665
F1-Score	0.98120

Tabla 3.5: Resumen de las medidas obtenidas con SVM

Los resultados generales como podemos ver no son tan buenos a priori como en la red neuronal y random forest, aún así vamos a ver la matriz de confusión:

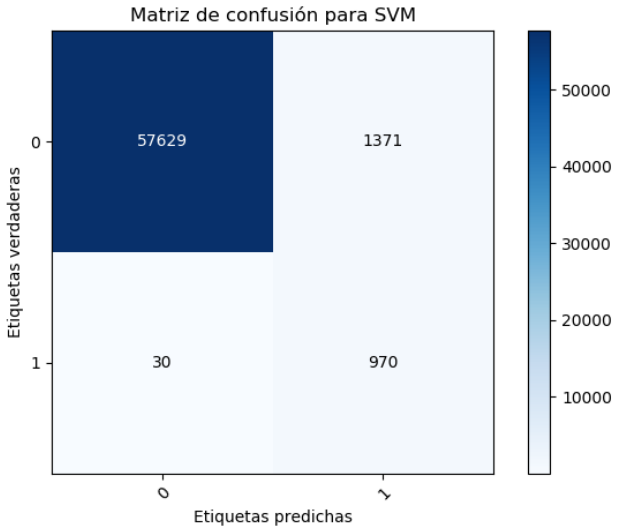

Figura 3.9: Matriz de confusión para SVM

Como podemos ver en este caso se producen 970 aciertos y 30 fallos con lo que este modelo ha conseguido funcionar mejor que SGD y AdaBoost pero no que Random Forest ni Perceptrón multicapa.

Veamos la curva ROC:

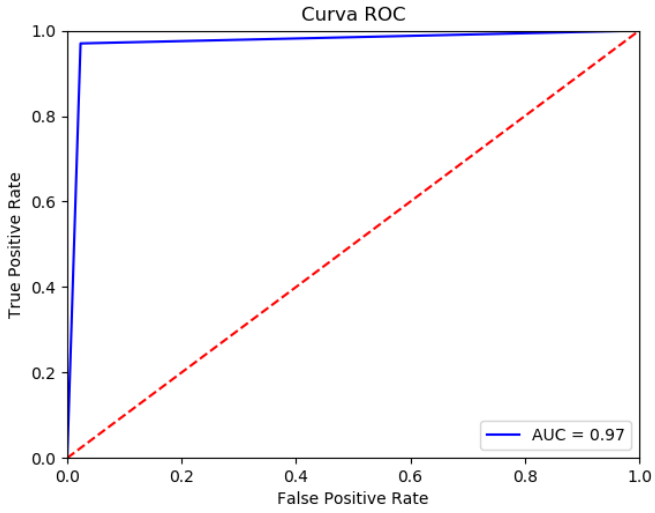

Figura 3.10: Curva ROC para SVM

4. Bibliografía

Referencias

[1] Yaser S. Abu-Mostafa. *Learning from Data.* AMLBook, 2012.

[2] R.B. Cattell. Factor analysis. *New York: Harper*, 1952.

[3] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 2002.

[4] B. Fruchter. Introduction to Factor Analysis. *Van Nostrand*, 1954.

[5] W.K. Härdle and Z. Hlávka. Multivariate Statistics. *Springer*, 2015.

[6] L. Simar and W.K. Härdle. Applied Multivariate Statistical Analysis. *Springer*, 2015.