APRENDIZAJE AUTOMÁTICO:

PRACTICA 4:
PROYECTO FINAL

Jesús Moyano Doña - 76626194-S
Antonio Manuel Fresneda Rodríguez - 77447672-W
jesusmoyano97@correo.ugr.es
antoniomfr@correo.ugr.es
Grupo A3

Índice

1. Introducción:

La practica consiste en el análisis de un conjunto de datos basado en la información proveída por un emisor de tarjetas de crédito de Taiwan. Esta base de datos cuenta con información de 30000 clientes. Esta información de cada cliente viene dada en 24 variables, donde están incluidos datos demográficos, bancarios, historial de pagos y dinero en la cuenta desde Abril del 2005 a septiembre de 2005.

A continuación una breve descripción de las variables:

- **ID:** Identificador de cada cliente.
- **LIMIT_BAL:** Cantidad dinero dado en el crédito, su unidad es el NT dolar (New Taiwan Dolar). Se incluye el crédito individual como el familiar (suplementario).
- **SEX:** Genero del cliente. Masculino=1, Femenino=2.
- **EDUCATION:** Formacion Profesional=1, Universidad=2, Secundaria=3, Otros=4, 5/6=Desconocido .
- **MARRIAGE:** Estado Civil. Casado=1, Soltero=2, Otros=3.
- **AGE:** Edad en años.
- **Pay_X:** Estado del pago. Sin consumo=-2, Pago Debidamente=-1, Uso del credito revolvemente=0, Retraso en el pago por un mes=1, Retraso en el pago por dos meses=2, ... Retraso en el pago por ocho meses=8, Retraso en el pago por nueve meses o más=9.
  PAY_0 corresponde a Septiembre 2005, PAY_2 Agosto 2005, ... PAY_6 Abril 2005.
- **BILL_AMTX:** Estado de la cuenta en NT dolars. BILL_AMT1 corresponde a Septiembre 2005, BILL_AMT2 Agosto 2005, ... BILL_AMT6 Abril 2005.
- **PAY_AMTX:** Cantidad del pago anterior. PAY_AMT1 corresponde a Septiembre 2005, PAY_AMT2 Agosto 2005, ... PAY_AMT6 Abril 2005.

Sobre el conjunto de datos se efectuará la clasificación. La variable que define las etiquetas es default.payment.next.month. Define si se da un credito o no y toma el valor 1 para Sí y 0 para No.
El orden de las variables ha sido alterada en la fase de pre-procesado para una mayor simplicidad.

2. Pre-procesado de datos

Antes de comenzar la clasificación es necesario realizar un pre-procesado de los datos, para ello se usa estandarizado. La forma en que obtiene los datos es la siguiente: $x^{'} = \frac{x-\overline{x}}{\sigma}$.

Se probó la ampliación de la clase de funciones, introduciendo polinomios de mayor grado,pero los resultados eran peores.

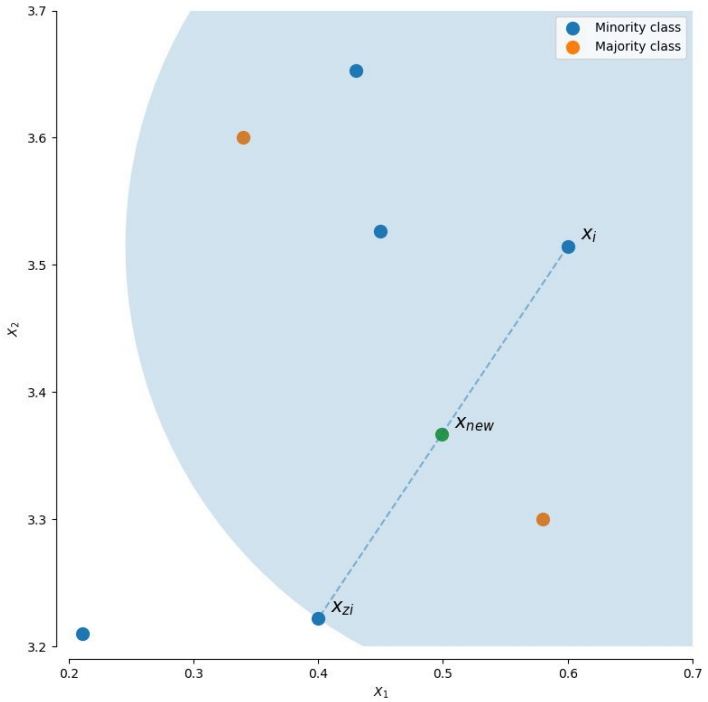
Los conjuntos de datos entrenamiento y test usados para esta practica han sido asignados de la siguiente forma: conjunto de entrenamiento es una partición del 80 % de la base de datos y el conjunto de test ha sido obtenido con el 20 % restante.

Como el conjunto de datos contiene variables categóricas se han tenido que transformar mediante una codificación, realizada con el One Hot Encoder, implementado en la librería Sk-Learn. Este método convierte cada variable categórica en una codificación binaria.

La base de datos está desbalanceada:

- **Clase 0(No):** Cuenta con 23364 instancias, un 77.88 % del total.
- **Clase 1(Si):** Cuenta con 6636 instancias, un 22.12 % del total.

Esto va a provocar que al modelo le cueste más clasificar la clase que esta desbalanceada. Para intentar balancearla se ha utilizado el algoritmo SMOTE *(Sintetic minority oversmapling technique)*. Este algoritmo considera una muestra X_i y genera una nueva X_{new} utilizando su K vecino más cercano (mediante el clasificador KNN que calcula la distancia entre cada dato y predice en función de esta misma). Una vez generado K vecinos se escoge uno X_{zi}, se obtiene una nueva muestra de la siguiente forma: $X_{new} = X_i + \lambda \times (X_{zi} - X_i)$, donde λ es un numero aleatorio entre 0 y 1.
Esto es una representación gráfica de como se calcularía:



Este algoritmo esta implementado en la librería Imbalanced-Learn. Existen distintas variantes de este algoritmo que difieren en como selecciona las muestras X_i.

- **Regular:** No usa una regla, escoge de forma aleatoria todos los posibles X_i disponibles.
- **BorderLine:** Clasifica cada X_i en:
  - **Ruido:** Todos los vecinos calculados son de una clase distinta.
  - **En peligro:**Al menos uno de los vecinos es de la misma clase que X_i.
  - **Seguro:** Todos los vecinos son de la misma clase que X_i.

  Existen dos variantes y ambas usan las muestras *in danger* para generar nuevas muestras:
  - **BorderLine 1:** X_{zi} pertenecerá a una clase distinta de X_i.
  - **BorderLine 2:** X_{zi} puede pertenecer a cualquier clase.
- **SVM:** Utiliza un clasificador SVM para encontrar los vectores soporte y genera las muestras considerandolos.

Hay que destacar una cosa muy importante. Este algoritmo se aplica al conjunto de entrenamiento **UNICAMENTE**, ya que si aplicamos el algoritmo en test los resultados del mismo se van a ver muy modificados ya que estamos generando muestras sintéticas las que muy probablemente el clasificador clasifique perfectamente.

3. Hiper-parámetros

En la elección de los parametros a tener encuenta se usa una funcion de Sk-Learn llamada gridSearch. Esta función tiene como parámetros:

- **Param grid**: Un diccionario (o una serie de diccionarios en la lista) en la que la clave es un string el parámetro que queremos probar y como valor para la clave una lista con los distintos valores para el parámetro.
- **Estimator**: Es un modelo (regresión ó clasificación) implementado en sk-learn.
- **CV**: En este parámetro se especifica el número de particiones que se van a usar para realizar una cross-validation con los datos que se han pasado (lo hace internamente).
- **Scoring**: Con este parámetro indicamos que tipo de métrica queremos que use para ver que parámetro es mejor o peor. Hay que destacar que internamente intenta maximizar el resultado de la métrica. Ya que la clase está des-balanceada se usa la métrica F1 ya que tiene en cuenta tanto Precision(Habilidad del clasificador de no etiquetar como positiva una muestra negativa) como Recall(Habilidad del clasificador para encontrar todas las muestras positivas):

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

.

También se lleva a cabo el proceso de validación. Se realiza mediante una validación cruzada sobre conjunto de entrenamiento, dividiendo el mismo en 5 partes.

4. Modelos

4.1. Modelos Lineales:

4.1.1. Regresión Logística:

Como modelo lineal se ha escogido regresión logística porque clasifica los datos y da las probabilidades, que indican como de seguro está el clasificador respecto a las perdiciones obtenidas.

4.2. Modelos No Lineales:

4.2.1. Decision Tree:

Se usa este modelo como punto de inicio ya que explica el valor predicho. Se obtiene un sesgo bajo pero una gran varianza. En el modelo Random forest se combinan varios de estos.
Este modelo no es adecuado para los datos, puesto que es obvio que hay una gran varianza en los estos debido al des-balanceo de los mismos.
Tras los experimentos se verifica lo dicho anteriormente. En el próximo apartado se mostrarán gráficas sobre el estudio.

4.2.2. Random Forest:

Esta construido sobre la idea de bagging pero aporta una mejora de construir correlados. Este modelo pierde expresividad respecto a los arboles de decisión ya que su objetivo es disminuir la varianza. Tiene pocos parámetros: Número de predictores y el numero de variables a tener en cuenta en la ramificación de los arboles. Pueden actuar de buena forma con conjuntos de datos des-balanceados.

4.2.3. Boosting:

Boosting combina varios predictores simples con el objetivo de obtener un mejor predictor. Tiene las siguientes ventajas: Pocos parámetros (Unicamente el número de estimadores a usar), eficiente computacionalmente, versátil ya que puede usar distintos tipos de datos (texto, numéricas y discretos). Ya que el tamaño del conjunto de datos es grande funcionará bien. Usamos AdaBoost.

5. Análisis de los modelos:

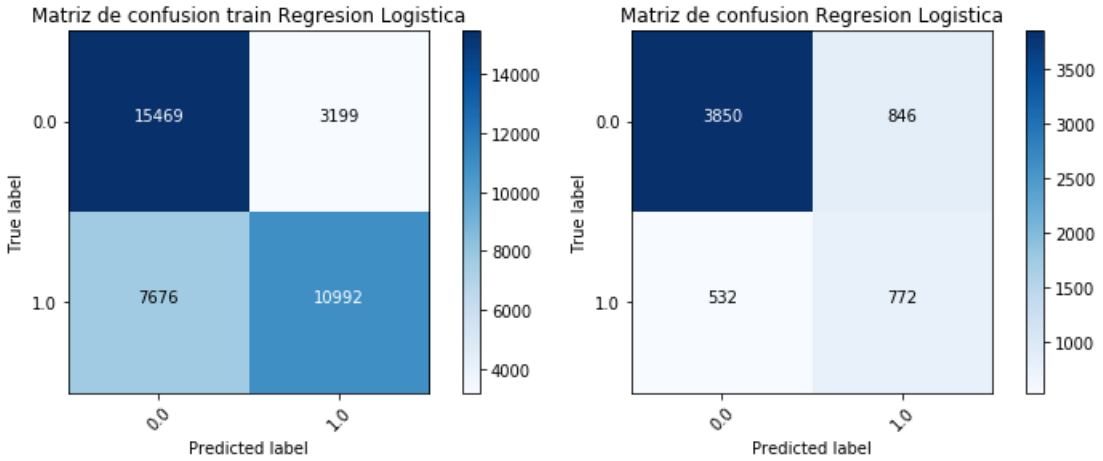
Para la comparación de los anteriores modelos se ha usado la curva roc, el area bajo la curva y la matriz de confusión. Se ha usado la semilla 123 para los cálculos sobre aleatorios.

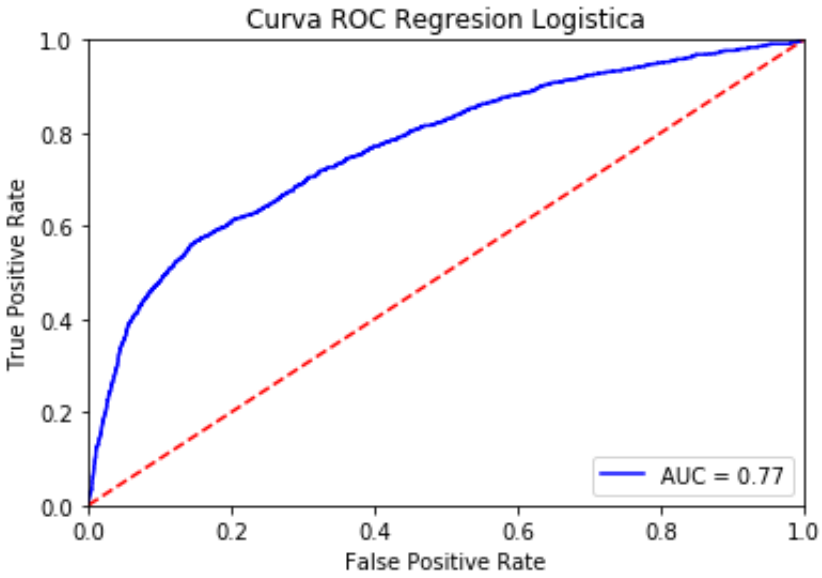
5.1. Regresión Logística

Se han realizado experimentos con los siguientes parametros:

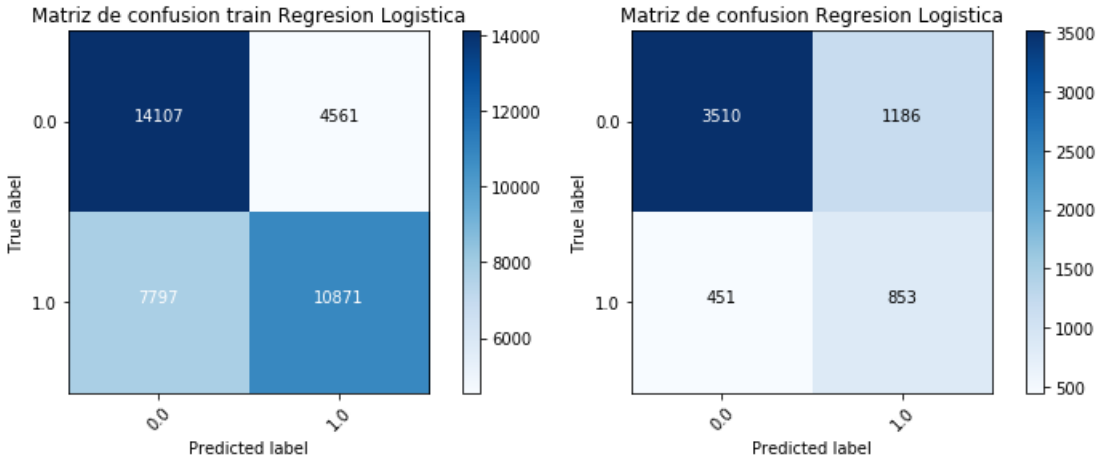
- **Penalty:** L1(Lasso) y L2(Ridge).
- **C**: Inversa de la fuerza que aplicamos a la regularización. Se han escogido los siguientes valores: [3,2,0.9, 0.5, 0.2,] . Se ha elegido con el objetivo de no aumentar mucho la fuerza de la regularización, ya que puede estropear el ajuste.

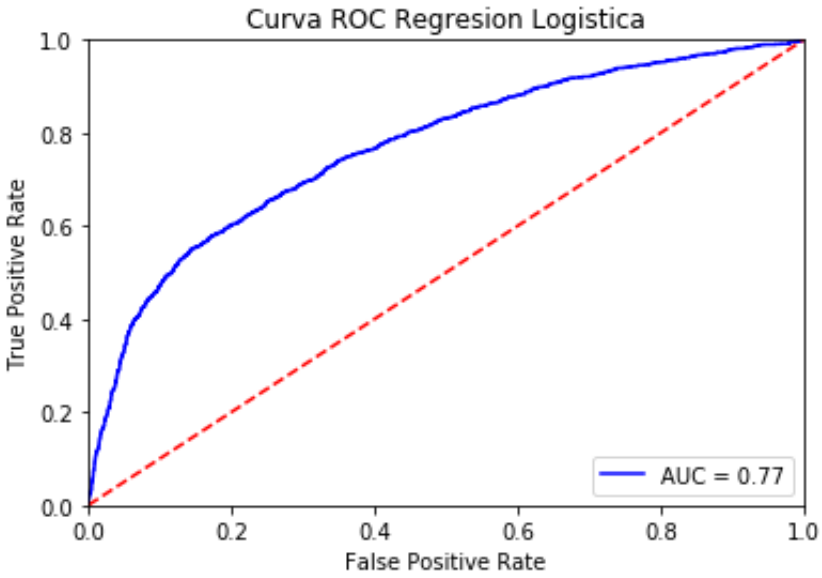
Resultados Para SMOTE 'REGULAR':
Los mejores parámetros obtenidos han sido 'C': 3, 'penalty': 'l2'.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.



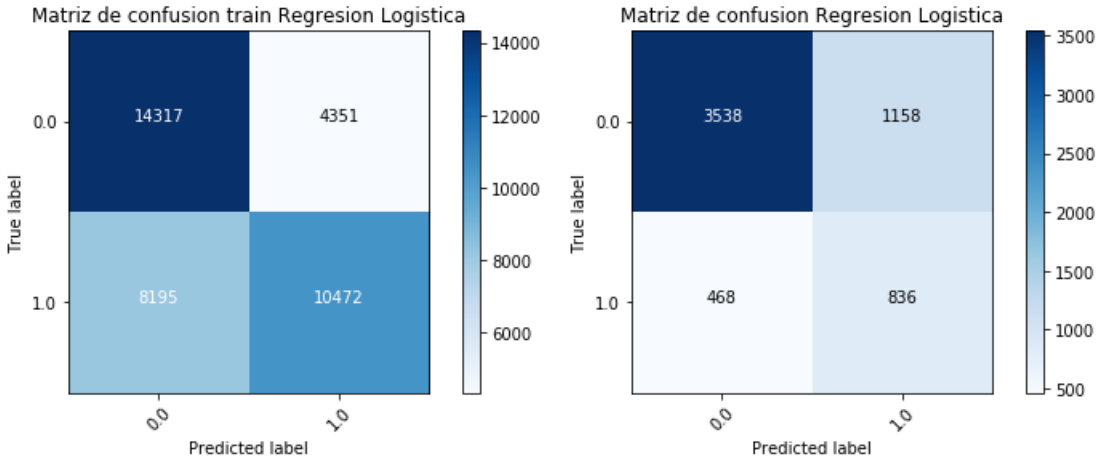


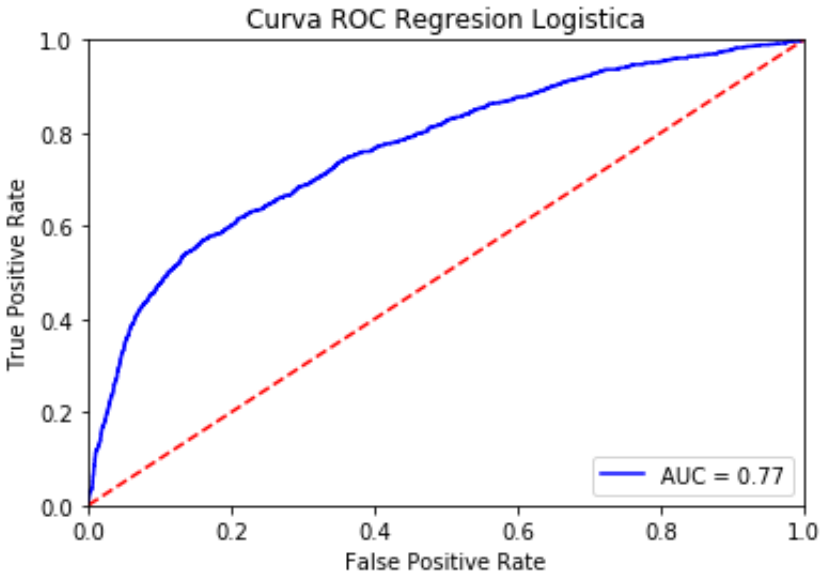
Resultados Para SMOTE 'BorderLine1':
Los mejores parámetros obtenidos han sido 'C': 0.2, 'penalty': 'l2'.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.





Resultados Para SMOTE 'BorderLine2':
Los mejores parámetros obtenidos han sido 'C': 2, 'penalty': 'l2'.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.





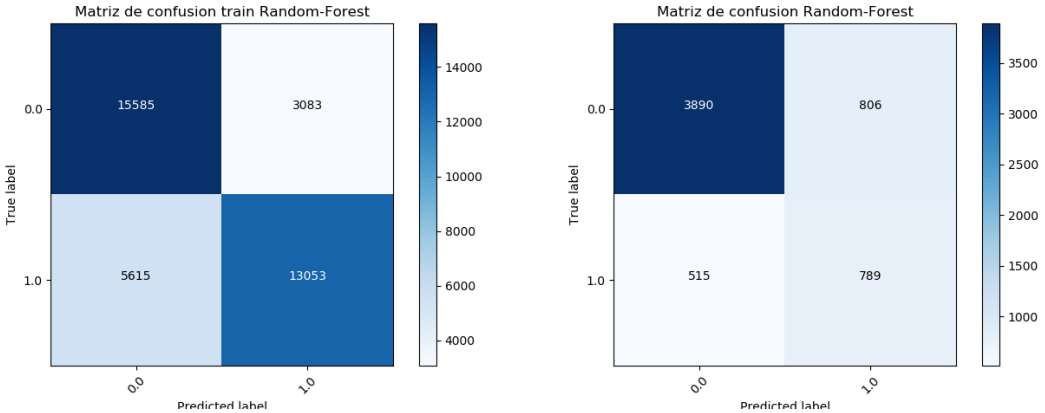
5.2. Random Forest

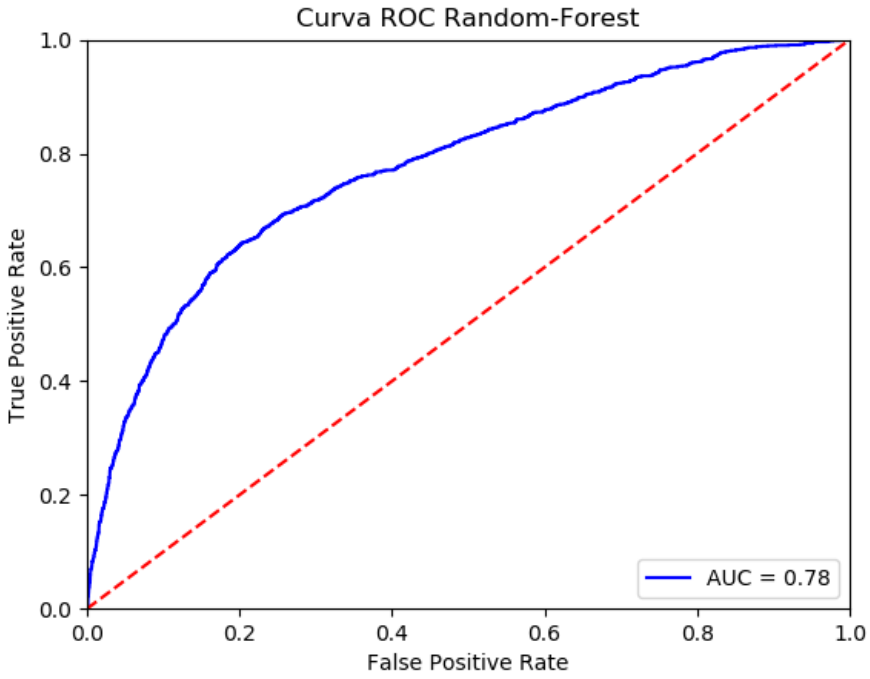
Se han realizado experimentos con los siguientes parametros:

- **N_estimators:**: Número de arboles que usa. Se han utilizado estos valores: [10,15,19,30,35]
- **max_features:**: Numero de características a tener en cuenta. Por defecto ['sqrt'].
- **max_deph**: Profundidad de los árboles. Se han utilizado estos valores: [10,20,30,35,40,50,60,70]
- **min_samples_leaf**: Numero mínimo de muestras necesarias para que sea un nodo hoja. Se han utilizado estos valores:[80,90,100][1]

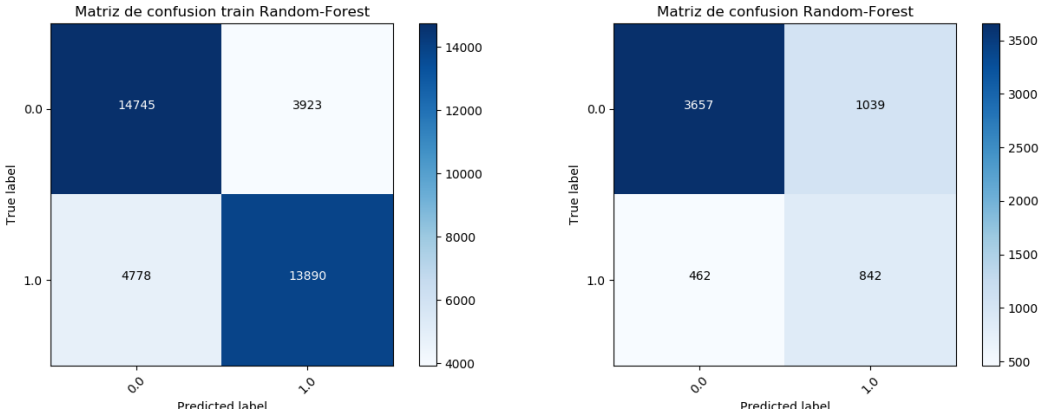
[1]Tanto este parámetro como max_deph los hemos modificado puesto que el se producía sobre-aprendizaje. La clase 0 la clasificaba correctamente (sobre un 80 %) pero en la clase 1 clasificaba muy mal (50 %) en el conjunto de test.

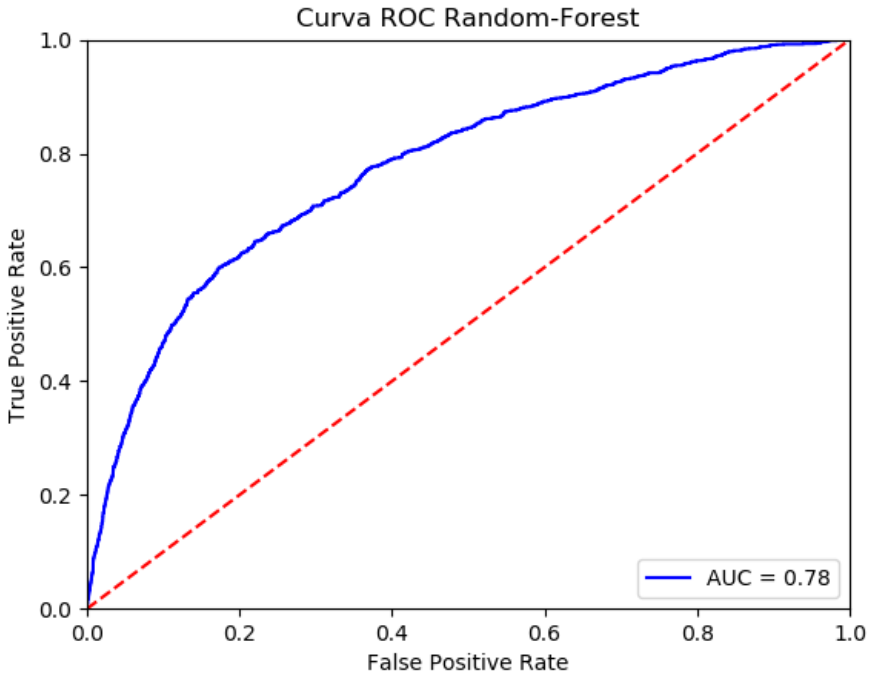
Resultados Para SMOTE 'REGULAR':
Los mejores parámetros obtenidos han sido'max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 80, 'n_estimators': 19.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.



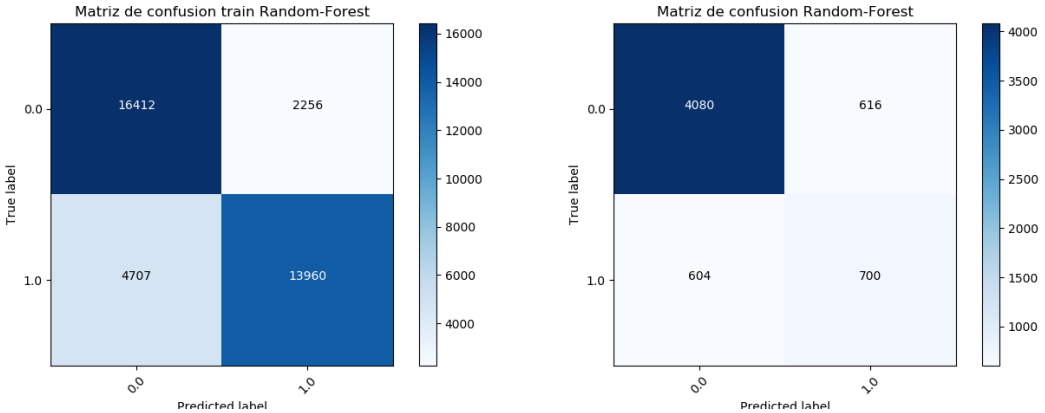


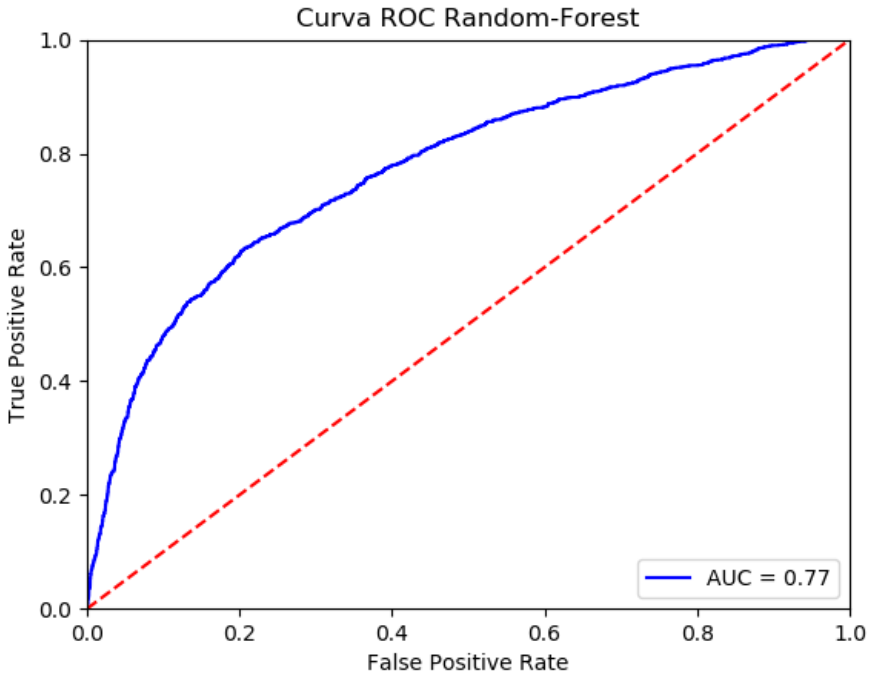
Resultados Para SMOTE 'BorderLine1':
Los mejores parámetros obtenidos han sido 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 80, 'n_estimators': 30.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.





Resultados Para SMOTE 'BorderLine2':
Los mejores parámetros obtenidos han sido'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 80, 'n_estimators': 30 .
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.



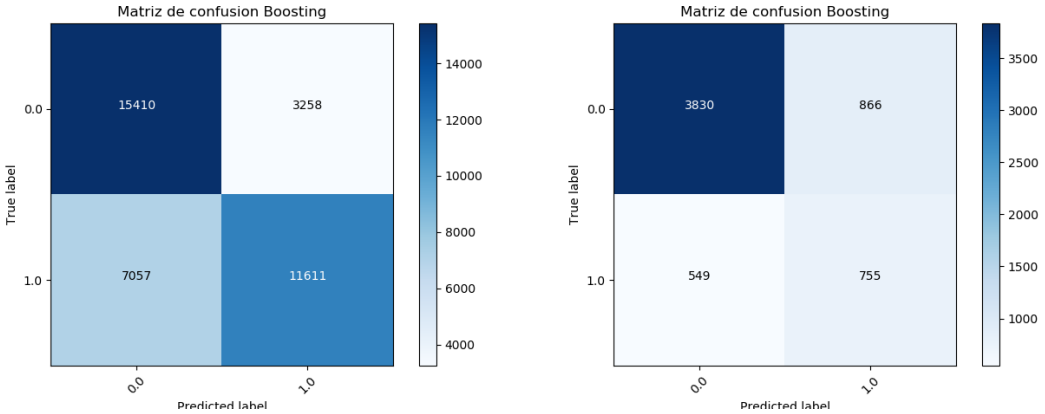


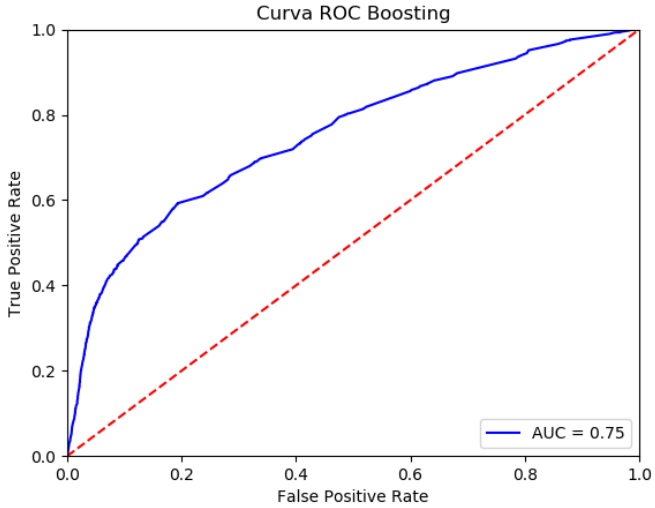
5.3. Boosting

Se han realizado experimentos con los siguientes parametros:

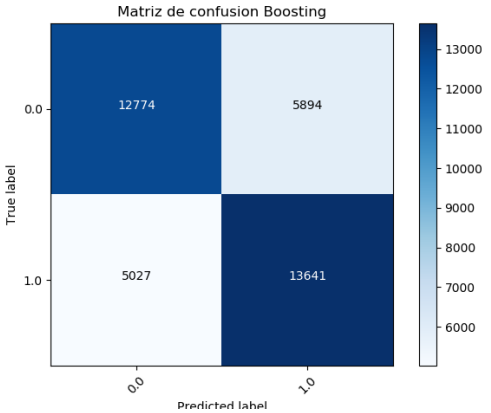
- **learning_rate:** Tasa de aprendizaje que reduce la contribución. Se ha probado [0.9,0.65,0.5,0.2].
- **n_estimators:** Número de estimadores. En este caso se han usado como estimadores débiles arboles de decisión. Se ha probado con [11,12,13,14,15]

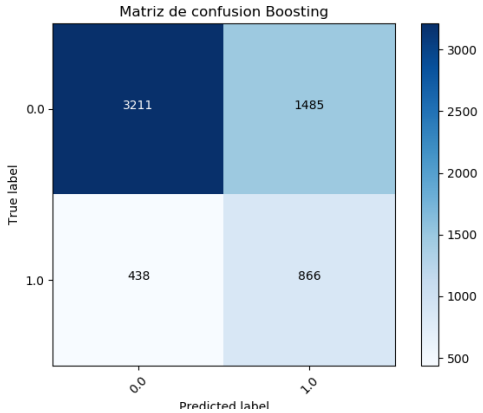
Resultados Para SMOTE 'REGULAR':
Los mejores parámetros obtenidos han sido: 'learning_rate': 0.9, 'n_estimators': 70 .
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.

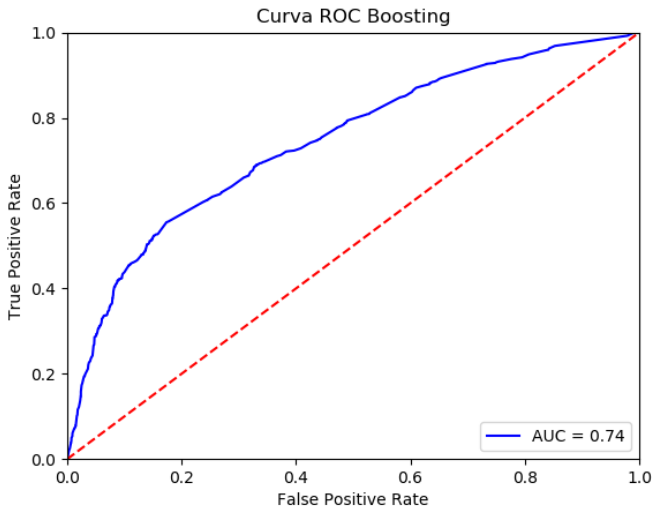




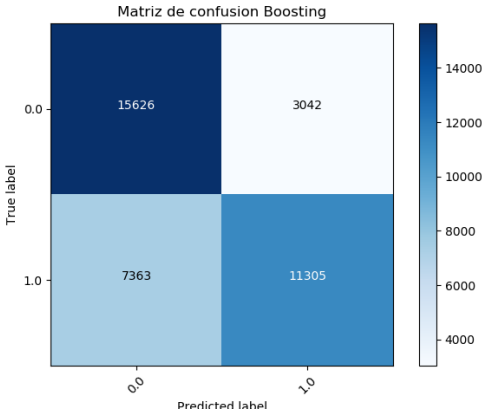
Resultados Para SMOTE 'BorderLine1':
Los mejores parámetros obtenidos han sido 'learning_rate': 0.9, 'n_estimators': 13.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.

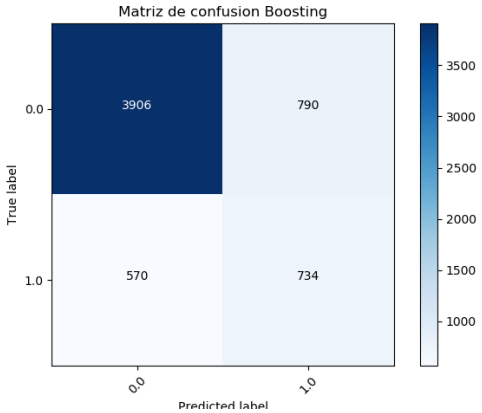


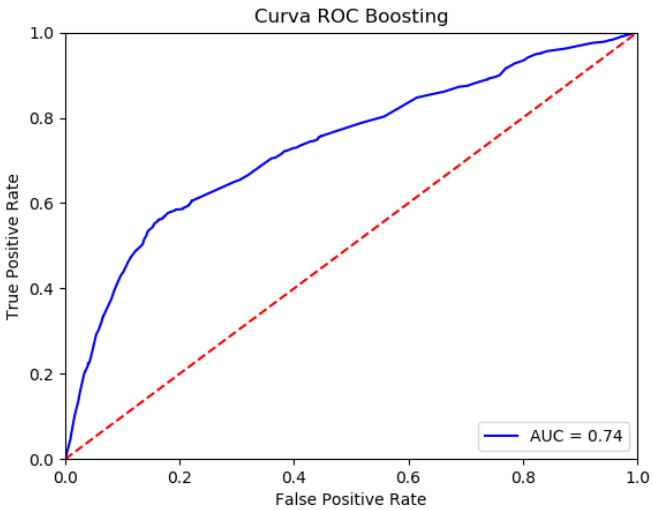




Resultados Para SMOTE 'BorderLine2':
Los mejores parámetros obtenidos han sido 'learning_rate': 0.65 y 'n_estimators': 14.
A la izquierda está la matriz de confusión para el error del conjunto de entrenamiento y a la derecha la de test.

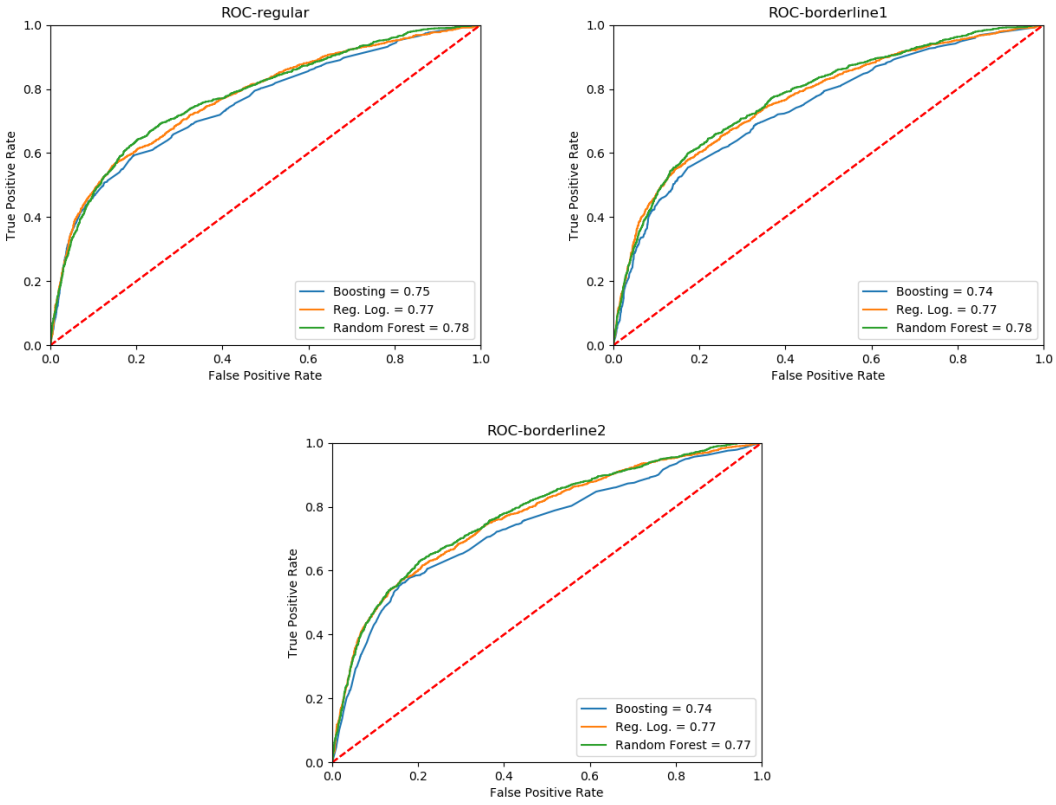






6. Conclusiones

6.1. Curvas ROC para la comparación



Analizando las gráficas, se puede ver que no hay un claro ganador. Todos los modelos que se han usado han sido capaces de lograr una capacidad de predicción de entre un 78 % y un 74 %.
Habría que tener en cuenta la opinión del banquero a la hora de elegir el modelo ya que, aunque todos han predicho en los valores mencionados, hay diferencias en el porcentaje de clasificación de la clase 1. Si el banquero considera que para el es más importante clasificar a los que sí se les dá crédito (clase 1) el mejor modelo que se consideraría sería el que mejor la clasifica (Boosting usando SMOTE borderline1).

7. Bibliografia

- https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-mach
- http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html
- http://contrib.scikit-learn.org/imbalanced-learn/stable/over_sampling.html#smote-adasyn
- https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/
- Documentación de Scikit-learn.