Proyecto Final. Default of credit card clients

Aprendizaje Automático
Víctor Manuel Arroyo Martín
Sergio Cabezas González de Lara

Primera parte (hasta 15 puntos): Comparación entre Regresión Logística y Random Forest.
Segunda parte (hasta 25 puntos): Comparación también con Perceptrón Multicapa (MLP).
BONUS: Añadido Máquina de Soporte de Vectores (SVM).

1 Definición del problema a resolver y enfoque elegido.

Se han extraido los datos de los usuarios de una tarjeta de crédito y su uso durante medio año tanto de reembolsos de crédito prestado como de pagos con la tarjeta. Queremos predecir si un usuario va a fallar en reembolsar el crédito prestado de la tarjeta de crédito o no y con ello si se le permite un pago con la tarjeta o no. Tenemos una base de datos con 23 características y 30000 muestras realizadas donde cada muestra aparece etiquetada así que es un problema aprendizaje supervisado. La base no presenta valores perdidos.
Es un problema de clasificación binaria pues tenemos dos etiquetas: 1 bloqueo de pago, 0 aceptación del pago. Queremos aproximar una función de clasificación que es desconocida y que a partir de datos con las caracterícticas que presentan los de la base de datos pueda predecir si aceptar o rechazar un pago. Para ello vamos a entrenar varios modelos de aprendizaje y seleccionar el mejor de ellos.

2 Argumentos a favor de la elección de los modelos.

Hemos elegido como modelo lineal Regresión Logística ya que previamente hemos eliminado las variables muy correladas entre sí que suelen causar problemas a este modelo y es eficiente (fácil de entrenar), simple y con resultados que se pueden interpretar como una probabilidad de que tome cierta etiqueta, lo cual tiene mucho sentido en nuestro problema: la probabilidad de que se rechace el pago del usuario.
También hemos usado SVM, entre sus ventajas están que se pueden modelar relaciones complejas, no lineales y es robusto al ruido, esto se debe a que maximizan los márgenes.
Random Forest: Puede manejar fácilmente un gran número de muestras como es nuestro caso y selecciona las características más significativas para el ajuste, y no tiene por qué usarlas todas.
Perceptrón multicapa (MLP): Usa aprendizaje adaptativo y usa información probabilística de los datos que nos es útil en nuestro problema.

3 Codificación de los datos de entrada para hacerlos útiles a los algoritmos.

Los datos vienen un archivo excel (xls) y para leer de él hemos usado la librería pandas y una función para leer esa extensión. Hemos sacado las etiquetas de la última columna y hemos separado tanto etiquetas como datos en un 30% test y un 70% training al que luego aplicamos validación cruzada.

4 Valoración del interés de las variables para el problema y selección de un subconjunto (en su caso).

Hemos eliminado la primera columna que no aportaba nada para el aprendizaje (es el identificador de cada muestra) y hemos quitado la primera fila que contenía los nombres de las características.
Hemos usado todas las variables ya que no hay un gran número y hemos eliminado las que están muy correladas entre sí. Para ello hemos hecho la matriz de correlación y las parejas de variables que superaran el 0'9 entre ellas, hemos dejado sólo una de ellas para el aprendizaje ya que ambas aportan prácticamente la misma información para el problema.

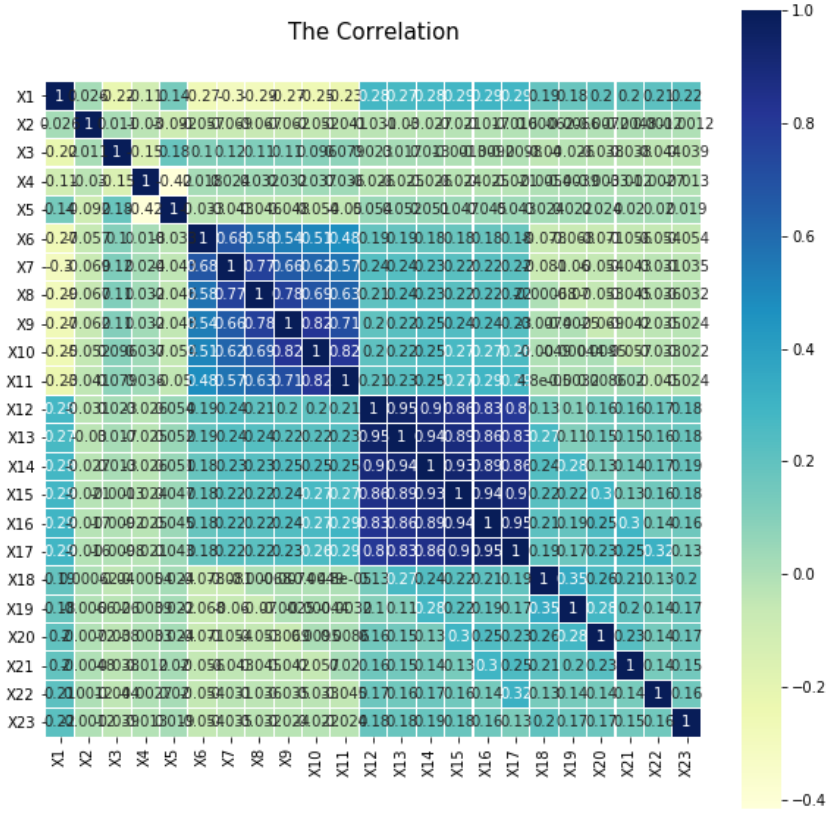

Figure 1: Matriz de correlación de las variables.

Para eliminar variables sólo tenemos que mirar la matriz triangular superior de la matriz de correlación pues es simétrica.

5 Normalización de las variables (en su caso)

Hemos usado la función StandardScaler que estandariza (normaliza) las características eliminando la media y escalando a la varianza unidad. Si u es la media y s la desviación típica de nuestros datos de entrenamiento, standardScaler transforma un dato x en

$$\text{z}=\frac{(x-u)}{s}$$

En nuestro modelo es útil ya que tenemos características con valores muy grandes y otros muy pequeños, como por ejemplo, el sexo (X2) es 1 o 2 y en cambio cantidad de crétido prestado (X1) son del orden de 1000 así que la normalización aquí es casi necesaria.

6 Justificación de la función de pérdida usada.

Para SVM hemos usado la función de perdida "hinge" de sklearn ya que es la función de pérdida más usada en este modelo y se usa para conseguir el máximo margen en la clasificación, que es lo que queremos en nuestro problema. La función está definida por:
$Loss(y) = max0, 1 - ty$ con t=$\pm$1 que indica si la muestra está bien clasificada o no en base a la función de clasificación.

El MLP y la regresión logística de sklearn usa la función de pérdida Cross-Entropy que en un problema de clasificación binaria como la nuestra está dada por:
$Loss(\hat{y}, y, W) = -ylog(\hat{y}) - (1-y)log(1-\hat{y}) + \alpha \parallel W \parallel_2^2$
Donde $\alpha \parallel W \parallel_2^2$ es el término de regularización L2 y α>0 es un hiperparámetro de ésta. Hemos usado esta porque es la única elegida por sklearn para clasificación.

Para random forest usamos también la función de pérdida Cross-Entropy ya que el modelo de sklearn puede usar dos funciones de pérdida (gini y entropy) y el modelo hemos probado experimentalmente que se ajusta mejor con entropy.

7 Selección del modelo lineal paramétrico y valoración de su idoneidad frente a otras alternativas

Hemos elegido como modelo lineal a comparar regresión logística ya que la función de pérdida en sklearn de este modelo es la misma que usamos para varios de los modelos no lineales empleados. Gracias a ello podemos comparar fielmente nuestro modelo lineal con el resto de no lineales que hemos elegido.

8 Aplicación de las técnicas especificando claramente que algoritmos se usan en la estimación de los parámetros, los hiperparámetros y el error de generalización.

No hemos necesitado aplicar ninguna técnica de estimación de los valores ya que nuestra base de datos no presenta ningún dato perdido.
Hemos estimado que la profundidad del árbol en random forest debe ser mayor que 4 pues tenemos 16 características y según lo que vimos en teoría, la profundidad debe ser mayor que la raíz de 16, que es 4. Tras experimentar con el número de estimadores hemos concluido que 100 y 200 dan buenos resultados. El parámetro n-jobs está a -1 para que haga el aprendizaje en paralelo.
Para la técnica de ajuste en Regresión Logística hemos usado el lbfgs que ha dado mejores resultados que sgd y que newton-cg. El parámetro de regularización C indica el grado de impacto que tiene la regularización sobre el aprendizaje: cuanto más cercano a 0 sea su valor, mayor imopacto. Por ello, hemos escogido 5 valores entre -2 y 2 en escala logarítmica. Para la regularización hemos usado L1 (explicado en el siguiente apartado).
Para la técnica de ajuste de MLP hemos usado sgd que es el que mejores resultados lanzaba. Para activar las "hidden layers" probamos dos funciones: "relu", que es un rectificador lineal y "logistic" que es la función sigmoide de regresión logística. Hemos puesto 1000 iteraciones como máximo para dar cierto margen al algoritmo. El número de capas está por defecto a 3, como pide el enunciado de la práctica.
Para el SVM hemos puesto el kernel rbf como recomendaba el enunciado de las prácticas y para el parámetro gamma hemos usado scaler en vez de auto ya que lanzaba mejores resultados. El parámetro C tiene la misma explicación que en Regresión Logística y el grado está puesto a 2 porque lanza unos resultados mucho mejores que 1 o 3.

9 Argumentar sobre la idoneidad de la función regularización usada (en su caso)

Cuando hubiera elección (como es el caso de Regresión Logística) hemos usado L1 (Regularización Lasso) pues es el que mejor funciona en general cuando no hay variables muy correladas entre sí (nosotros se las hemos quitado al principio con la matriz de correlación) así que es idóneo para este problema.

10 Valoración de los resultados (gráficas, métricas de error, análisis de residuos, etc)

Hemos usado GridSearchCV para seleccionar el mejor modelo mediante validación cruzada partiendo el conjunto de train en 5 partes para ello, obteniendo los siguientes resultados:

Accuracy en CV: 82.233%
Accuracy en training: 82.790%

Accuracy en test: 81.967%

Son resultados bastante aceptables pues el margen de error es pequeño. Para obtener gráficas, hemos hecho una reducción de características mediante PCA a 2 y 3 características y por la distribución de los datos podemos intuir que son difícilmente separables y por ello necesitamos las 16 características.

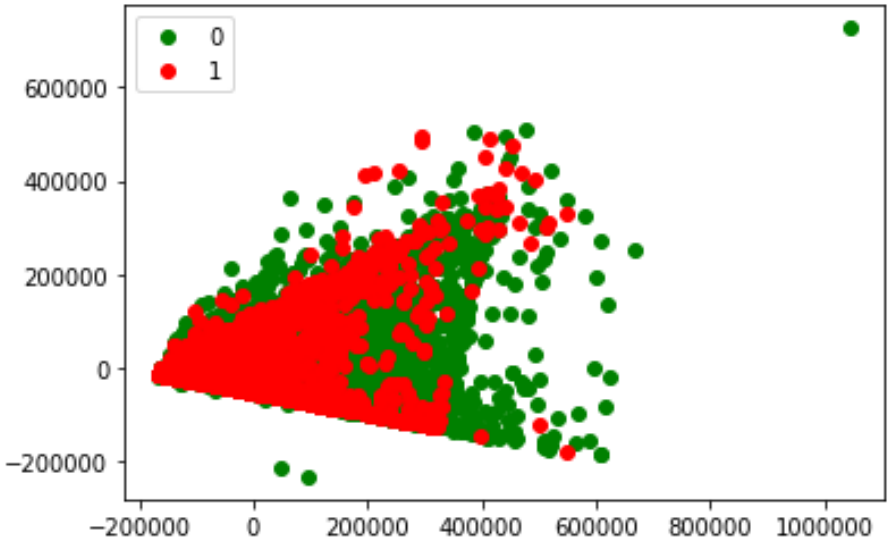

Figure 2: Gráfica con dimensión 2

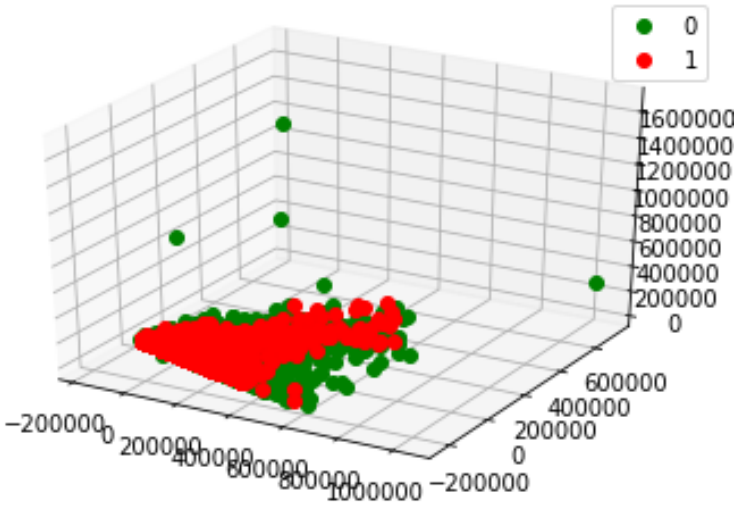

Figure 3: Gráfica con dimensión 3

La métrica de error que hemos usado para este problema es precisión (accuracy) pues queremos darle importancia a que las predicciones que hagamos, tanto positivas como negativas, sean correctas porque no nos interesa bloquearle la tarjeta a un cliente erróneamente ni generar una pérdida para el banco.

11 Justificar que se ha obtenido la mejor de las posibles soluciones con la técnica elegida y la muestra dada. Argumentar en términos de los errores de ajuste y generalización.

En ningún caso del aprendizaje automático podemos asegurar que el modelo elegido sea el mejor de todos los posibles. Sin embargo, sí podemos asegurar que de entre los modelos que hemos elegido para comparar el mejor es Random Forest, ya que es el que nos muestra como mejor modelo GridSearchCV y por tanto el que mejor error de validación tiene y el que mejor generaliza el problema a partir de los datos.