Doble Grado en Ingeniería Informática y Matemáticas

Aprendizaje Automático

(E. Computación y Sistemas Inteligentes)

PROYECTO FINAL

Ajuste y selección del mejor predictor a la BBDD de la UCI Default of Credit Card Clients con el apoyo de la libreria Scikit-Learn.

UNIVERSIDAD DE GRANADA

Alberto Estepa Fernández
Email: albertoestep@correo.ugr.es

Carlos Santiago Sánchez Muñoz
Email: carlossamu7@correo.ugr.es

18 de junio de 2020

Índice

1. Default of Credit Card Clients

1.1. Definición del problema a resolver y enfoque elegido.

Tenemos entre manos un conjunto de datos de crédito de un banco de Taiwán a fecha de octubre de 2005. El conjunto de datos recoge información de los clientes e indica si se les concede o no un crédito de forma predeterminada. La base de datos está disponible en [1].

El dataset consiste en 24 variables de las cuales una de ellas será la variable a predecir (`default payment next month`), una variable binaria que indica si se le concede el crédito (`1`) o si no se le concede el crédito (`0`). Así podemos afirmar que estamos ante un problema de clasificación binaria. Las otras 23 características serán las variables explicativas del problema que nos ayudaran a predecir la variable objetivo. Explicamos las 23 variables explicativas del conjunto de datos a continuación:

- `LIMIT_BAL`: Cantidad del crédito otorgado (en NT dollar, es decir, nuevo dólar taiwanés). Incluye tanto el crédito al consumidor individual como el crédito (complementario) de su familia.
- `SEX`: Género (`1` = masculino; `2` = femenino).
- `EDUCATION`: Educación (`1` = escuela de posgrado; `2` = universidad; `3` = escuela secundaria; `4` = otros).
- `MARRIAGE`: Estado civil (`1` = casado; `2` = soltero; `3` = otras).
- `AGE`: Edad (en años).
- `PAY_0`: el estado del reembolso en septiembre de 2005.
- `PAY_2`: el estado del reembolso en agosto de 2005.
- `PAY_3`: el estado del reembolso en julio de 2005.
- `PAY_4`: el estado del reembolso en junio de 2005.
- `PAY_5`: el estado del reembolso en mayo de 2005.
- `PAY_6`: el estado del reembolso en abril de 2005.
- `BILL_AMT1`: Cantidad de dinero (en NT dollar) en la cuenta en septiembre de 2005.
- `BILL_AMT2`: Cantidad de dinero (en NT dollar) en la cuenta en agosto de 2005.
- `BILL_AMT3`: Cantidad de dinero (en NT dollar) en la cuenta en julio de 2005.
- `BILL_AMT4`: Cantidad de dinero (en NT dollar) en la cuenta en junio de 2005.
- `BILL_AMT5`: Cantidad de dinero (en NT dollar) en la cuenta en mayo de 2005.

- `BILL_AMT6`: Cantidad de dinero (en NT dollar) en la cuenta en abril de 2005.
- `PAY_AMT1`: Cantidad de dinero (en NT dollar) pagado en septiembre de 2005.
- `PAY_AMT2`: Cantidad de dinero (en NT dollar) pagado en agosto de 2005.
- `PAY_AMT3`: Cantidad de dinero (en NT dollar) pagado en julio de 2005.
- `PAY_AMT4`: Cantidad de dinero (en NT dollar) pagado en junio de 2005.
- `PAY_AMT5`: Cantidad de dinero (en NT dollar) pagado en mayo de 2005.
- `PAY_AMT6`: Cantidad de dinero (en NT dollar) pagado en abril de 2005.

Como vemos tenemos información historial de pagos pendientes, es decir, tenemos los últimos registros de pagos mensuales (de abril a septiembre de 2005) (atributos `PAY_x`, donde `x` $\in \{0, 2, 3, 4, 5, 6\}$. La escala de medición para el estado de reembolso es:

- `-1`: pagado debidamente.
- `1`: retraso en el pago de un mes.
- `2`: retraso en el pago de dos meses.
- ...
- `8`: retraso en el pago de ocho meses.
- `9`: retraso en el pago de nueve meses o más.

También disponemos de información del dinero del que dispuso el cliente en cuenta en los últimos 6 meses (`BILL_AMTx`) y el dinero ya pagado (`PAY_AMTx`), donde `x` $\in \{1, 2, 3, 4, 5, 6\}$.

Definiendo matemática y computacionalmente nuestro problema, es obvio que estamos ante un problema de aprendizaje supervisado ya que, para un conjunto finito de instancias del problema, conocemos la etiqueta que corresponde a cada una de las instancias y a partir de dicho conjunto de instancias, correctamente clasificadas, podemos aprender una función de clasificación que nos indique la clase de cada instancia del dominio. Vemos así que $\mathcal{X}$ es el conjunto de 23 características que incluye el dataset que incluye la información sobre el cliente, es decir:

$$\begin{aligned}\mathcal{X} =& \texttt{LIMIT\_BAL} \times \{1, 2\} \times \{1, 2, 3, 4\} \times \{1, 2, 3\} \times \{18, \ldots, 100\} \times \\ & \times \{-1, \ldots, 9\}^6 \times \texttt{BILL\_AMTi} \times \texttt{PAY\_AMTi}\end{aligned}$$

donde `LIMIT_BAL` $\in [0, +\infty)$, `BILL_AMTi` $\in (-\infty, +\infty)^6$ y `PAY_AMTi` $\in [0, +\infty)^6$.

El conjunto de etiquetas $\mathcal{Y}$ incluye números enteros (`0` o `1`) que indican si se le concede (`1`) o si no se le concede (`0`) el crédito a cada cliente, es decir: $\mathcal{Y} = \{\texttt{0}, \texttt{1}\}$.

La función de objetivo es $f : \mathcal{X} \to \mathcal{Y}$ que para cada instancia $(x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ verifica $f(x_n) = y_n$.

TODO: En la búsqueda de la función objetivo f es necesario fijar la clase de funciones o conjunto de hipótesis $\mathcal{H}$. Más adelante elegiremos $g \in \mathcal{H}$ de modo que $g \approx f$.

En el caso lineal:

$$\mathcal{H}_1 = \left\{ w_0 + \sum_{i=1}^{N} w_i x_i, \quad w \in \mathbb{R}^{N+1} \right\}.$$

y en el otro: (HACER)

$$\mathcal{H}_2 = \left\{ w_0 + \sum_{i=1}^{N} w_i x_i, \quad w \in \mathbb{R}^{N+1} \right\}.$$

TODO: Meter gráficas de cada variable que tenga sentido para ver como están distribuidas.

Son 13 gráficas de barras metemos aquí una y luego seis y seis.

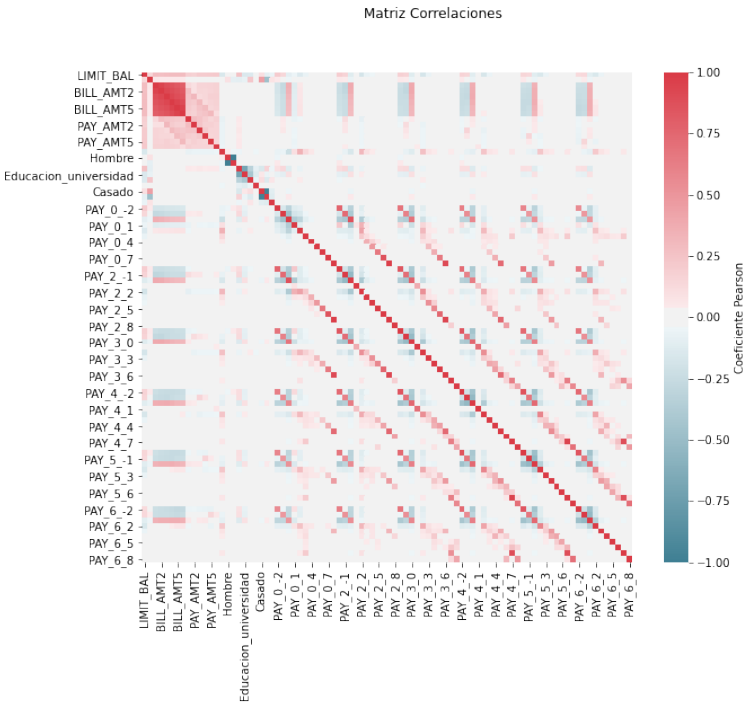
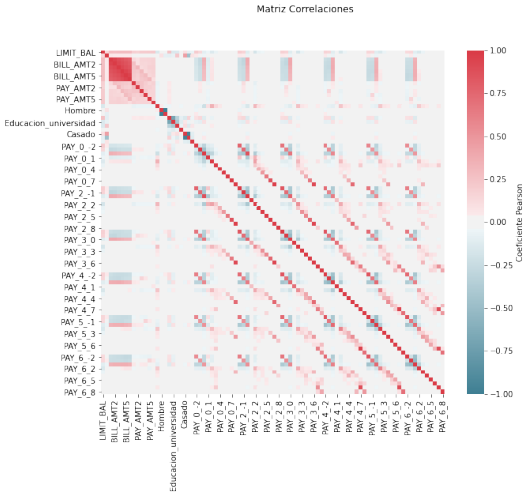


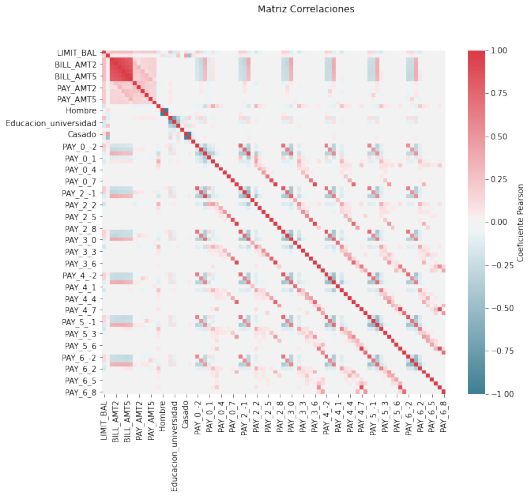
Imagen 1: Atributo `LIMIT_BALL`

Texto.

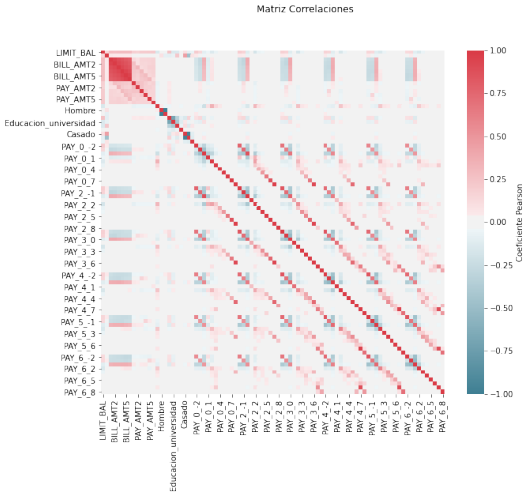
Otro aspecto importante a comentar es que hemos comprobado que disponemos de 30.000 instancias en nuestro conjunto de datos. Cada una de éstas instancias se



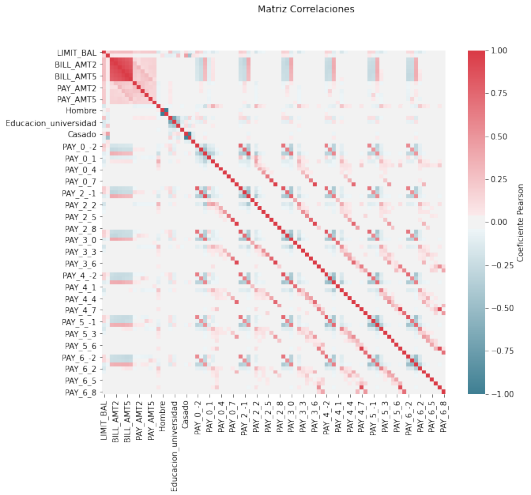
(a) Atributo `BILL_AMT1`



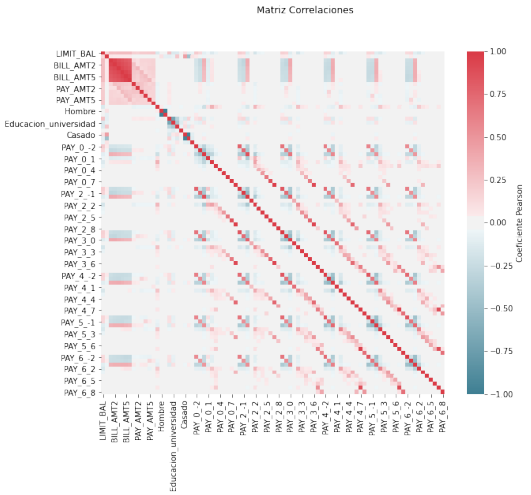
(b) Atributo `BILL_AMT2`



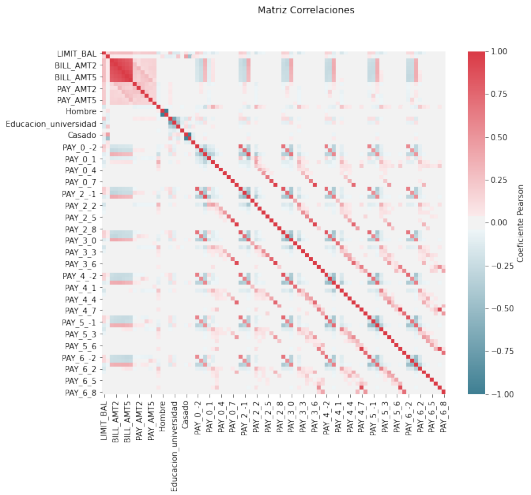
(c) Atributo `BILL_AMT3`



(d) Atributo `BILL_AMT4`

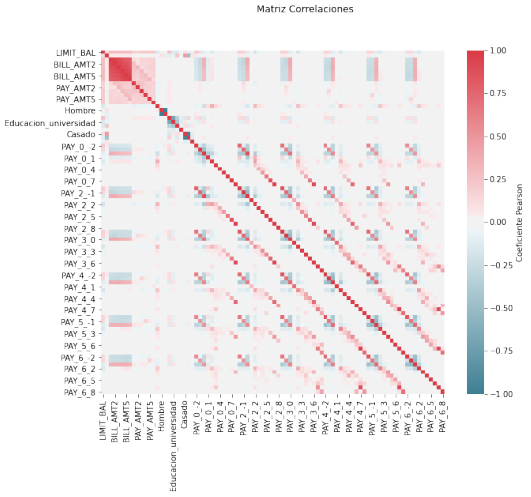


(e) Atributo `BILL_AMT5`

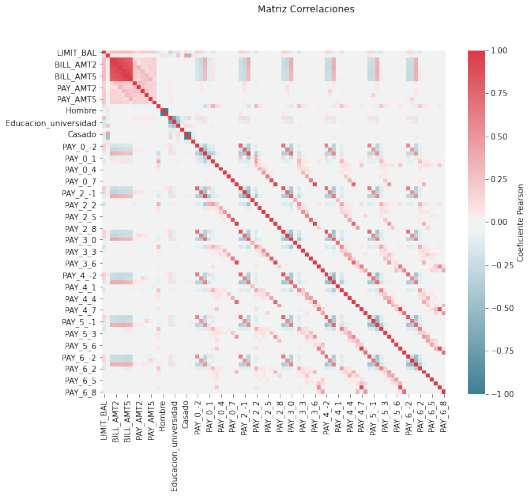


(f) Atributo `BILL_AMT6`

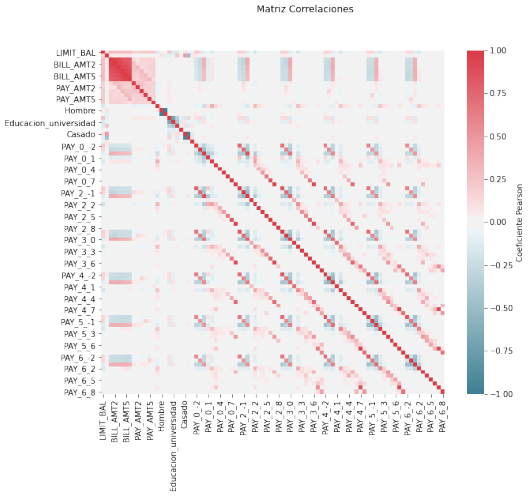
Imagen 2: Gráficas de barras de `BILL_AMTx`



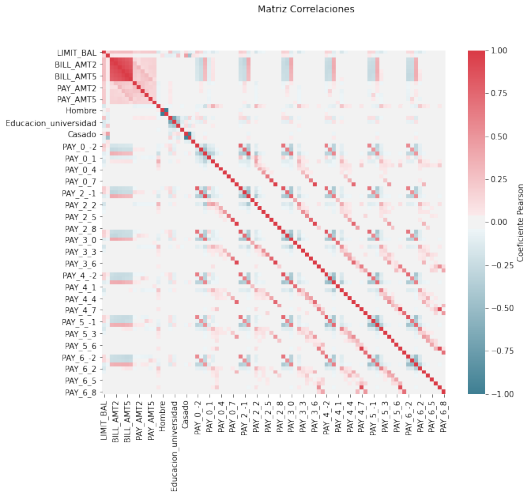
(a) Atributo `PAY_AMT1`



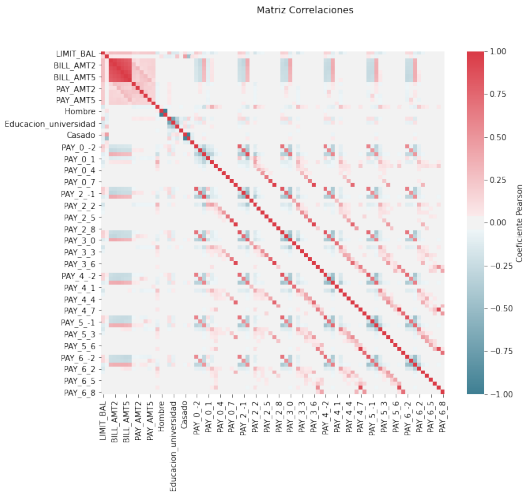
(b) Atributo `PAY_AMT2`



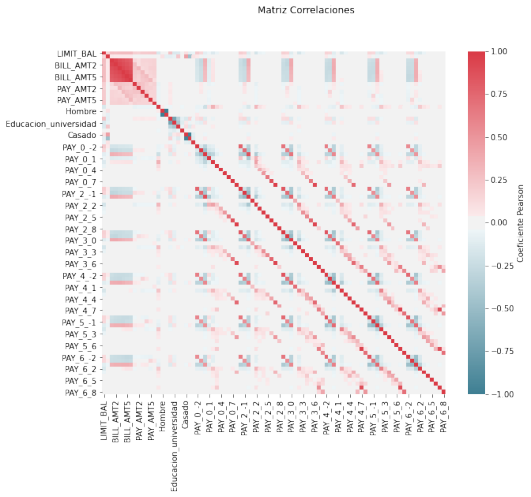
(c) Atributo `PAY_AMT3`



(d) Atributo `PAY_AMT4`



(e) Atributo `PAY_AMT5`



(f) Atributo `PAY_AMT6`

Imagen 3: Gráficas de barras de `PAY_AMTx`

corresponde con un cliente del banco distinto. Analizando el número de instancias que identifican cada una de las dos clases podemos ver que disponemos de 23364 instancias (un 77.88 % del total) de la clase 0 (no se le concede el crédito) y 6636 instancias un 22.12 % del total) de la clase 1 (se le concede el crédito). Aunque esto pueda parecer que las clases están bastante desbalanceadas, hay que recordar la situación en la que estamos: el banco necesita ser restrictivo sobre a que cliente concede un crédito o no. El banco concederá un crédito a aquellos clientes que puedan devolver con garantías el dinero y es importante, por el beneficio del banco, no conceder créditos a aquellos clientes que no tengan garantías de devolver el crédito. Por esto es normal que las clases mantengan dicha proporción en nuestro conjunto de datos. De esta forma, dado el gran número de instancias totales del conjunto de datos y que el número total de instancias de la clase más pequeña permite aprender de forma suficiente, consideramos que no debemos tratar de forma especial éste tema.

TODO: Hablar del enfoque elegido (en nuestro caso clasificación en vez de regresión). Podemos mencionar el paper donde se hace y se explica una regresión.

1.2. Argumentos a favor de la elección de los modelos.

Nuestro problema es de clasificación binaria y hay varios modelos que se pueden usar. Se han elegido tres modelos, uno de ellos lineal. Veamos:

(A) Regresión Logística.

El primer modelo elegido es el de Regresión Logística, que sabemos que es un modelo lineal.

De sobra conocemos que el modelo de regresión logística aplica una una función logística (denominada sigmoide) a los datos, es decir aplica $\sigma(\omega^T x)$, donde $\sigma(z) \in [0, 1]$ y

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Así la estimación de los parámetros ω del modelo se efectúa por medio de del método de estimación por máxima verosimilitud.

Como tenemos un problema de clasificación binaria, podemos usar dicho método para comprobar si las clases son linealmente separables (en dicho caso obtendremos resultados bastante óptimos con dicho modelo). Sabemos que es más robusto al ruido que otros modelos de clasificación lineales. Además la regresión logística resulta bastante eficaz al enfrentarnos a un gran volumen de datos, como es nuestro caso. Por todo esto y ya que en la práctica anterior obtuvimos bastante buenos resultados con él, nos hemos decantado por dicho modelo frente a otros modelos lineales.

Usaremos por tanto la clase *LogisticRegression* de la librería Sklearn de Python.

(B) Máquina de Soporte de Vectores (SVM).

TODO: Ver si usamos este u otro.

El siguiente método escogido es una Máquina de Soporte de Vectores (SVM) con un núcleo no lineal. Esto permitirá precisar una buena clasificación si nuestros datos no son linealmente separables.

Sabemos un SVM busca un hiperplano óptimo de separación entre las clases.

Usaremos por tanto la clase SVC de la librería Sklearn de Python.

(C) Random Forest.

El último método escogido para la comparación es Random Forest. Dichos modelos no esperan que las dependencias sean lineales o incluso que los atributos interactúen de forma lineal.

Además sabemos que se comportan de forma eficaz ante los espacios de alta dimensión (anticipánndonos a los acontecimientos, vamos a aumentar la dimensionalidad del dataset en las secciones posteriores), así como con un gran número de ejemplos de entrenamiento, como es nuestro caso.

Usaremos por tanto la clase $RandomForestClassifier$ de la librería Sklearn de Python.

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

Los datos se han proporcionado en un fichero *.xls* que es el formato de archivo implementado por las versiones antiguas de Microsoft Excel. Usaremos la estructura $dataframe$ proporcionada por Python para el tratamiento de los datos. Para su lectura hemos hecho uso de la función $read\_excel$ de la librería Pandas de Python. Con dicha función le indicamos también que no considere la primera fila del dichero, pues no proporciona información al conjunto de datos, así como que tome como columna índice la denominada `ID`. De esta forma no hemos necesitado modificar el fichero original proporcionado por la UCI.

```
import pandas as pd
df = pd.read_excel('datos/default of credit card clients.xls',
                   skiprows = 1, index_col = 'ID')
```

En la sección inicial (1.1) estudiamos los dominios de las variables de nuestro problema de acuerdo a la documentación proporcionada por la UCI, sin embargo nos hemos encontrado que no se corresponden con los datos encontrados en el dataset. Veamos los problemas en forma de 'outliers' encontrados, uno a uno:

- `EDUCATION`: Recordamos que la variable que indicaba los estudios del cliente se encontraba en un rango 1 y 4, siendo un valor natural, es decir, `EDUCATION` $= \{1, 2, 3, 4\}$. Sin embargo al estudiar el conjunto de datos nos hemos encontrado con que existen 280 instancias del valor 5, 51 instancias del valor 6 y 14 instancias del valor 0. Al ser tan pocas instancias y no resultan significativas en

el conjunto de datos (aproximadamente el 1 % de las instancias) hemos considerado incluirlos en el valor 4, que recordemos que significaba 'Otros niveles de estudios' (de los contemplados por los valores 1, 2 y 3) ya que es lógico que estén ahí incluidas.

- `MARRIAGE`: Recordamos que dicha variable indicaba el estado civil del cliente y sus posibles valores eran 1, 2 o 3. Sin embargo, hemos encontrado 54 instancias del valor 0 (menos del 0.2 % de las instancias). De nuevo, hemos considerado incorporarlos al valor 3, que indica 'Otro estado civil', basándonos en el mismo razonamiento que en la variable `EDUCATION`.
- `PAY_x` (donde `x` $= \{0, 2, 3, 4, 5, 6\}$). Este conjunto de atributos ha sido el más problemático, pues el dominio no se corresponde con el proporcionado en la documentación: se indicaba que los posibles valores eran $\{-1, 1, 2, \ldots, 9\}$ y nos hemos encontrado que los valores encontrados son $\{-2, -1, 0, 2, 3, 4, 5, 6, 7, 8\}$. Como no existe una relación lógica entre lo que indica la documentación y los valores encontrados hemos decidido no tratarlos de forma especial, pues eliminar dichas instancias problemáticas podría incluir el eliminar información clave del conjunto de datos a la hora de entrenar los diferente algoritmos.

Por otro lado, como comentamos en la sección inicial (1.1), todas las variables son numéricas, por lo que no tenemos variables categóricas a tratar.

Por último, respecto a ésta sección, aunque podíamos tratarla en la sección 1.5, hemos decidido binarizar cada una de las variables discretas multivaluadas. La motivación de realizar dicha transformación es que sabemos que los algoritmos a tratar trabajan mucho mejor con variables binarias (0 y 1) que con variables cuyo dominio abarca un gran número de valores, además de que el costo computacional de dicha transformación para nuestro conjunto de datos no será excesivamente grande. Para dicha transformación hemos recurrido a la función *get_dummies* de la librería Pandas, indicándole las columnas que convertiremos en binarias (que como hemos comentado serán aquellas que contemplan valores discretos en un rango pequeño, para no sobrecargar después computacionalmente los algoritmos).

```
df = pd.get_dummies(df, columns=['SEX', 'EDUCATION', 'MARRIAGE',
'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'])
```

Por último renombramos las variables resultantes (por ejemplo `SEX` se ha transformado en las variables `Hombre` y `Mujer`; y así con las demás transformaciones). Así damos por concluída la codificación de los datos de entrada para hacerlos útiles a los algoritmos, teniendo 87 variables explicativas en nuestro conjunto de datos.

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

En la sección 1.1 se estudió el significado de cada una de las variables y durante la sección 1.3 se mostró como hemos transformado cada una de dichas variables para

el correcto funcionamiento de los siguiente algoritmos que usaremos. Para ésta sección usaremos las variables ya transformadas para estudiar la selección de variables de interés para nuestro problema.

El primer paso que daremos es eliminar aquellas variables multicolineadas. Recordemos que la multicolinealidad entre variables se produce cuando existen relaciones lineales entre dichas variables, lo que implica que se aporte información repetida al problema y puede provocar un sobrecoste computacional del modelo.

Nos basaremos en el coeficiente de Pearson para calcular las correlaciones entre variables. Recordemos que si este coeficiente es igual a 1 o -1 (o cercano a estos valores) significa que las variables sufren multicolinealidad (aproximada, si no es 1 o -1; o exacta, si alcanza dichos valores). Para ello usaremos la matriz de correlaciones (no se incluye en los ejes el nombre de todas las variables por la dificultad visual de dicha acción, pero la matriz, obviamente, esta calculada sobre todos los atributos):

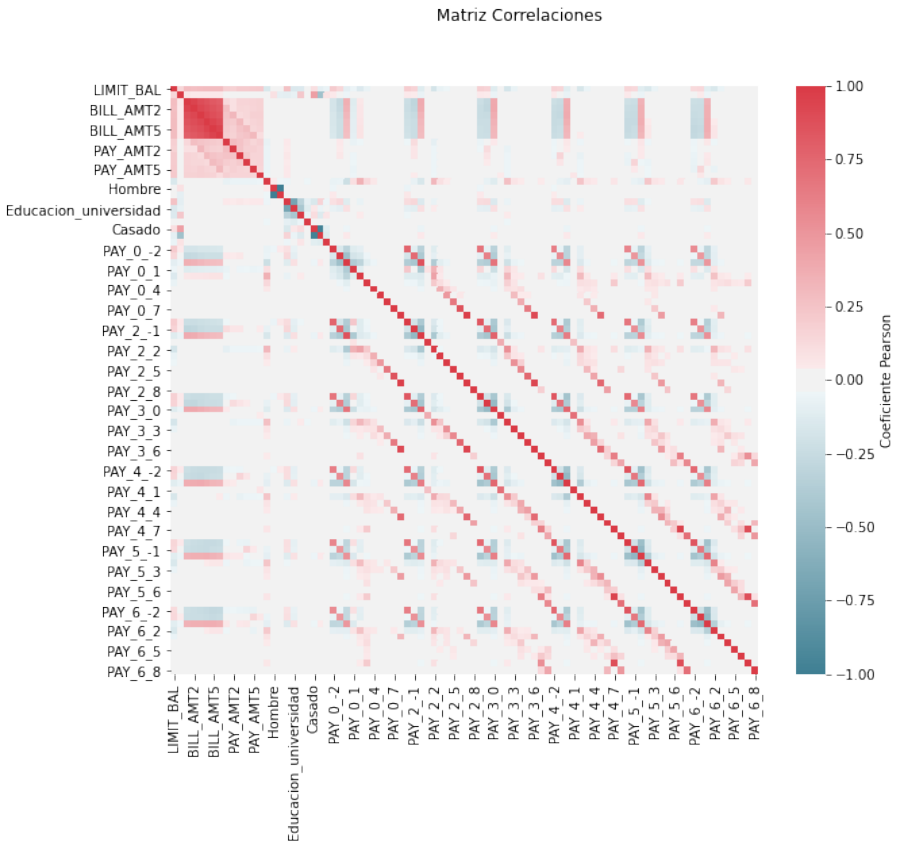


Imagen 4: Matriz de correlaciones

Vemos que existe una gran correlación entre las variables `BILL_AMTx` con `x` $= \{1, 2, 3, 4, 5, 6\}$. Además se observa una gran correlación inversa entre las variables `Mujer` y `Hombre` o entre las variables `Casado` y `Soltero`, entre otras.

Vamos a estudiar aquellas variables con multicolinealidad exacta (que tienen un coeficiente de Pearson en valor absoluto de 1, pues mediante una combinación lineal podemos obtener la otra variable, por lo que no es necesario su uso). Para ésta acción hemos implementado dos funciones en Python: la primera que calcula las variables que superen un valor de correlación dado; y la segunda que imprima la matriz de correlación ampliada ordenada de mayor a menor correlaci ón de Pearson, de cada una de las variables anteriores.

```
# Seleccionamos las columnas que son multicolineales
def columnas_correladas(dataframe, coeficiente = 1):
    # Matriz de correlación en valor absoluto
    corr_matrix = dataframe.corr(method = 'pearson').abs()

    # Seleccionamos la matriz triangular superior de la matriz de correlación anterior
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
                                                    k=1).astype(np.bool))

    # Buscamos la columnas con una correlacion dada con alguna otra
    to_drop = [column for column in upper.columns
                      if any(upper[column] >= coeficiente)]
    return to_drop
```

```
# Matriz de correlación en valor absoluto ampliada por columnas
def matriz_correlacion_ampliada(dataframe, columnas, k = 3):
# k: Número de variables a mostrar.
    # Matriz de correlación en valor absoluto
    corr_matrix = dataframe.corr(method = 'pearson').abs()
    for i in columnas:
        cols = corr_matrix.nlargest(k, i)[i].index
        cm = np.abs(np.corrcoef(dataframe[cols].values.T))
        ax = plt.axes()
        sns.set(font_scale = 1.25)
        hm = sns.heatmap(cm, cbar = True, annot = True, square = True,
                         fmt = '.2f', annot_kws = {'size': 10},
                         yticklabels = cols.values,
                         xticklabels = cols.values,
                         cbar_kws={'label': 'Coeficiente Pearson \
                                             en valor absoluto'})
        ax.set_title('M. de corr. ampliada de {}'.format(i))
        plt.show()
```

De esta forma, si realizamos las siguientes ejecuciones:

```
to_drop = columnas_correladas(dataframe = df, coeficiente = 1)
matriz_correlacion_ampliada(dataframe = df, k = 3, columnas = to_drop)
```

Obtenemos la siguiente salida:

Vemos que las variables `Mujer` y `Hombre` están multicolineadas de forma exacta entre sí (coeficiente de Pearson 1 en valor absoluto). Por tanto, apoyándonos en lo anteriormente descrito, podemos eliminar una de las dos variables (eliminaremos la variable `Hombre`, por ejemplo).

Si relajamos el coeficiente de correlación de la función anterior (1.4) vemos que la siguiente pareja de variables multicolineadas es `Casado` y `Soltero` con un coefi-

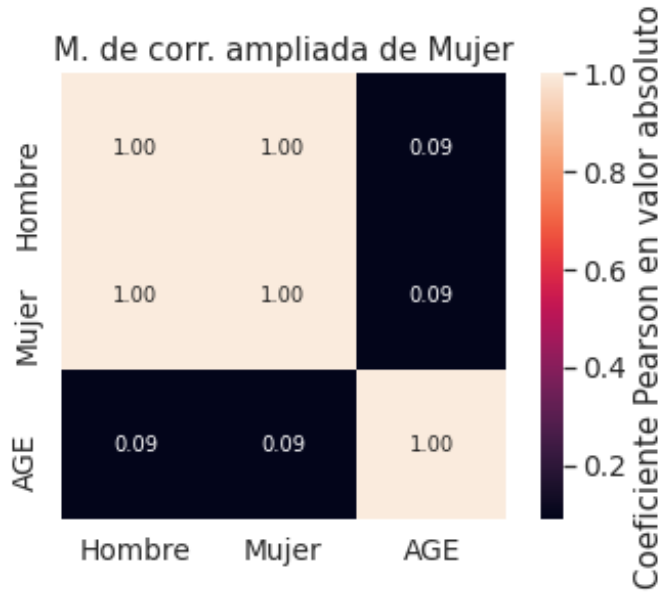


Imagen 5: Matriz de correlaciones en valor absoluto ampliada entre `Mujer` y `Hombre`

ciente de Pearson en valor absoluto de 0.98. Consideramos que, como el conjunto de variables no es demasiado grande (86 variables explicativas al eliminar la variable `Hombre`) y como queremos explicar toda la varianza posible de nuestro conjunto de datos, preferimos mantener dichas variables en nuestro conjunto de datos para no perder nada de información que puede ser útil para los algoritmos que vamos a utilizar.

De esta forma, consideramos que las demás variables son de interés para nuestro problema y que no necesitamos seleccionar un subconjunto más pequeño de entre las 87 variables de estudio).

1.5. Normalización de las variables (en su caso).

Tal y como comentamos en la sección 1.3, hemos binarizado cada una de las variables discretas multivaluadas con menos de 10 valores en su dominio. El rango de las nuevas variables obtenidas solo consiste en $\{0, 1\}$, por lo que no es necesario normalizarlas para nuestro algoritmo.

Sin embargo existen variables que no hemos tratado todavía: `LIMIT_BAL`, `AGE`, `BILL_AMTx` y `PAY_AMTx` donde `x` $= \{1, 2, 3, 4, 5, 6\}$. Dichas variables poseen valores enteros en un rango bastante amplio (para más información consulte la sección 1.1 donde se concretan los diferentes dominios de las variables). Para evitar problemas entre las escalas de las diferentes variables vamos a normalizar los datos de dichas variables entre 0 y 1.

Es de sobra conocido que el objetivo de la normalización es cambiar los valores

de las columnas numéricas del conjunto de datos para usar una escala común, sin distorsionar las diferencias en los intervalos de valores ni perder información. Además, la normalización también es necesaria para que algunos algoritmos modelen los datos correctamente. La normalización evita los problemas anteriormente expuestos mediante la creación de nuevos valores que mantienen la distribución general y las relaciones en los datos de origen. [4]

Para ello hemos usado la clase de Sklearn $MinMaxScaler$ que transforma las características escalando cada una a un rango dado (usaremos el rango por defecto $[0, 1]$. La transformación está dada por:

$$X\_std = (X - X.min(axis = 0))/(X.max(axis = 0) - X.min(axis = 0))$$

$$X\_scaled = X\_std * (max - min) + min$$

donde $[min, max] = [0, 1]$ en nuestro caso.

De esta forma tenemos normalizadas todas las variables de nuestro conjunto de datos.

1.6. Justificación de la función de pérdida usada.

TODO: Para la LogisticRegression https://stats.stackexchange.com/questions/188699/loss-function-of-scikit-learn-logisticregression [5]

1.7. Selección de las técnica (parámetrica) y valoración de la idoneidad de la misma frente a otras alternativas.

Si la técnica depende del modelo puede ser útil usar Modelo (A), (B) y (C).

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

Puede ser útil usar Modelo (A), Modelo (B) y Modelo (C).

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

Usar [3].

Puede ser útil usar Modelo (A), Modelo (B) y Modelo (C).

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

Puede ser útil usar Modelo (A), Modelo (B) y Modelo (C).

1.11. 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.

(a) Sin normalizar (b) Normalizada

Imagen 6: Matriz de confusión

Referencias

[1] http://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[2] I-Cheng Yeh & Che-hui Lien, *The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients*. ScienceDirect, Expert Systems with Applications **36** (2009). https://doi.org/10.1016/j.eswa.2007.12.020

[3] https://iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/

[4] https://docs.microsoft.com/es-es/azure/machine-learning/algorithm-module-reference/normalize-data

[5] https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression