Uso de modelos de clasificación de Machine Learning para la previsión de la morosidad en carteras hipotecarias y tarjetas de crédito.

NEOLAND

Memoria del proyecto final de Bootcamp

Ismael García Iñigo

Indice Página

1.0.- Breve introducción

El riesgo de crédito es una de los principales problemáticas a las que se enfrenta el negocio bancario (PILAR I), es por ello que la utilización de herramientas de inteligencia artificial para identificar patrones en los clientes, futuros o no, sobre su posible solvencia o morosidad futura, es una herramienta que puede tener amplio recorrido en este campo del sector financiero. Debido a la gran cantidad de datos que se proporciona a las ECAs a la hora de acceder a cualquiera de los dos instrumentos que en este proyecto se van a tratar, puede hacer aún más si cabe primordial el uso de herramientas de Machine Learning para la calificación de la calidad crediticia de los clientes de una entidad y que dicho modelo pueda ser una ayuda para la calificación crediticia de los clientes en futuras operaciones.

Aunque como se afirma en el párrafo anterior la cantidad de datos obtenidos por las ECAs es inmensa, se debe entender que se hace extremadamente difícil encontrar un dataset con datos completos debido a la normativa de protección de datos existente, así como el celo de las propias entidades para publicar información sobre si mismas. Esto hace que muchos datasets estén o incompletos o con una tipología de información u otra.

2.0.- Introducción y descripción del primer proyecto

El objetivo de este proyecto final de Bootcamp trata de aplicar los conocimientos alcanzados durante las horas lectivas para así medir nuestra capacidad y absorción de conocimientos a lo largo de estos 3 meses.

Tal y como el título propone, el proyecto que a continuación se va a explicar trata sobre la morosidad en el ámbito de la banca más concretamente en las hipotecas y en las tarjetas de crédito. El proyecto consta de dos datasets, uno de ellos es un dataset de una competición de machine learning denominada **PAKDD** (The Pacific-Asia Conference on Knowledge Discovery and Data Mining) de 2010 sobre defaults en tarjetas de crédito en Brasil. El dataset consta de 50000 individuos con los que se tiene que aproximar en la mejor medida posible un clasificador de que anticipe en base a unos atributos si el cliente va a entrar en mora o no. Los atributos de dicho dataset y su significado son los siguientes:

ID_CLIENT	Identificador cliente
CLERK_TYPE	Tipo de administrativo
PAYMENT_DAY	Día del mes en el que se liquida la cuenta, elegido por el cliente
APPLICATION_SUBMISSION_TYPE	Contratación vía web o en persona
QUANT_ADDITIONAL_CARDS	Cantidad de tarjetas adicionales contratadas en un mismo contrato
POSTAL_ADDRESS_TYPE	Si el código postal indicado en el contrato es el de la residencia habitual o no.
SEX	Sexo
MARITAL_STATUS	Estado civil
QUANT_DEPENDANTS	Cantidad de personas a su cargo
EDUCATION_LEVEL	Nivel educativo.
STATE_OF_BIRTH	Estado de Brasil de nacimiento

CITY_OF_BIRTH	Ciudad de nacimiento
NACIONALITY	Nacionalidad
RESIDENCIAL_STATE	Estado de Brasil en el que reside
RESIDENCIAL_CITY	Ciudad en la que reside
RESIDENCIAL_BOROUGH	Distrito/Vecindario en el que reside
FLAG_RESIDENCIAL_PHONE	Marcador que indica si tiene teléfono fijo o no
RESIDENCIAL_PHONE_AREA_CODE	Prefijo telefónico
RESIDENCE_TYPE	Indica el tipo relación de propiedad con su residencia habitual.(codigo numérico)
MONTHS_IN_RESIDENCE	Tiempo en meses que lleva en la residencia habitual.
FLAG_MOBILE_PHONE	Marcador que indica si el cliente tiene teléfono movil.
FLAG_EMAIL	Marcador que indica si hay una dirección de correo electrónico.
PERSONAL_MONTHLY_INCOME	Ingresos mensuales habituales en Reales Brasileños
OTHER_INCOMES	Otro tipología de ingresos indicados en Reales Brasileños
FLAG_VISA	Marcador de cliente con tarjeta visa
FLAG_MASTERCARD	Marcador de cliente con tarjeta mastercard
FLAG_DINERS	Marcador de cliente con tarjeta dinners
FLAG_AMERICAN_EXPRESS	Marcador de cliente con tarjeta American Express
FLAG_OTHER_CARDS	Marcador de cliente de otro tipo de tarjetas
QUANT_BANKING_ACCOUNTS	Cantidad de cuentas corrientes
QUANT_SPECIAL_BANKING_ACCOUNTS	Cantidad de otro tipo de cuentas bancaria
PERSONAL_ASSETS_VALUE	Valor total de sus activos tanto mobiliarios como inmobiliarios en Reales Brasileños
QUANT_CARS	Cantidad de coches en propiedad
COMPANY	El cliente ha facilitado el nombre de la compañía en la cual trabaja
PROFESSIONAL_STATE	Estado de Brasil donde trabaja
PROFESSIONAL_CITY	Ciudad de Brasil donde trabaja
PROFESSIONAL_BOROUGH	Distrito/Vecindario de Brasil donde trabaja
FLAG_PROFESSIONAL_PHONE	Marcador de si tiene un telefono corporativo

PROFESSIONAL_PHONE_AREA_CODE	Prefijo telefónico del número de telefono corporativo
MONTHS_IN_THE_JOB	Tiempo medido en meses en el trabajo actual
PROFESSION_CODE	Código de profesión
OCCUPATION_TYPE	Tipo de ocupación (código numérico)
MATE_PROFESSION_CODE	Código de profesión del compañero sentimental del cliente
EDUCATION_LEVEL	Nivel educativo del compañero sentimental del cliente
FLAG_HOME_ADDRESS_DOCUMENT	Marcador que indica si el cliente ha introducido la dirección del domicilio
FLAG_RG	Marcador que indica si el cliente ha introducido el número de documento de identidad
FLAG_CPF	Marcador que indica si el cliente ha introducido el información fiscal necesaria
FLAG_INCOME_PROOF	Marcador que indica si el cliente ha introducido un comprobante de la nómina
PRODUCT	Tipo de producto contratado
FLAG_ACSP_RECORD	Marcador que indica si el cliente ha entrado en mora en algún momento anterior
AGE	Edad
RESIDENCIAL_ZIP_3	Tres últimos números del código postal de la residencia habitual.
PROFESSIONAL_ZIP_3	Tres últimos números del código postal de la ciudad en la que trabaja
TARGET_LABEL_BAD=1	Variable de Clase que indica la morosidad es igual a 1.

Se trata de un dataset en el cual la practica totalidad de variables son categóricas, excepción de los atributos que describen la edad, las personas a su cargo y los la valoración de salarios/activos.

2.1.- EDA

En primer lugar, a la hora de hacer de nuestro EDA(Exploratory data analysis) se ve que al importar el archivo vía ***csv*** con la librería pandas no se visualizan los datos correctamente dado que el archivo que queremos importar es un ***txt*** que además viene con la característica de que los datos están separados por una tabulación (\t) , para visualizar correctamente el dataset dentro de la función de importación del ***csv*** , agregamos una orden ***delimiter*** ,para así eliminar la problemática de la tabulación y tener el dataset en condiciones para empezar a trabajar en él.

Con el dataset ya en condiciones ya para trabajar en el EDA añadimos al mismo los nombres de las columnas extrayéndolos de la ficha de excel que incluye la información aclaratoria sobre los atributos del proyecto. Una vez terminado este paso intermedio, se utiliza una función programada por el alumno denominada (***def eda***) en la cual se creara un dataframe con la siguiente información: número de valores nulos por columna, porcentaje de nulos por columna, tipo de dato que contiene cada columna, número de datos por columna (en este caso debería ser de 50000).

```
def eda(df):
    eda = {}
    eda['null_sum'] = df.isnull().sum()
    eda['null_pct'] = df.isnull().mean()
    eda['dtypes'] = df.dtypes
    eda['count'] = df.count()

    return pd.DataFrame(eda)
```

Una vez llamada la función con el dataframe de estudio nos sale que la totalidad de las columnas del dataframe están exentas de nulos con excepción de las siguientes:

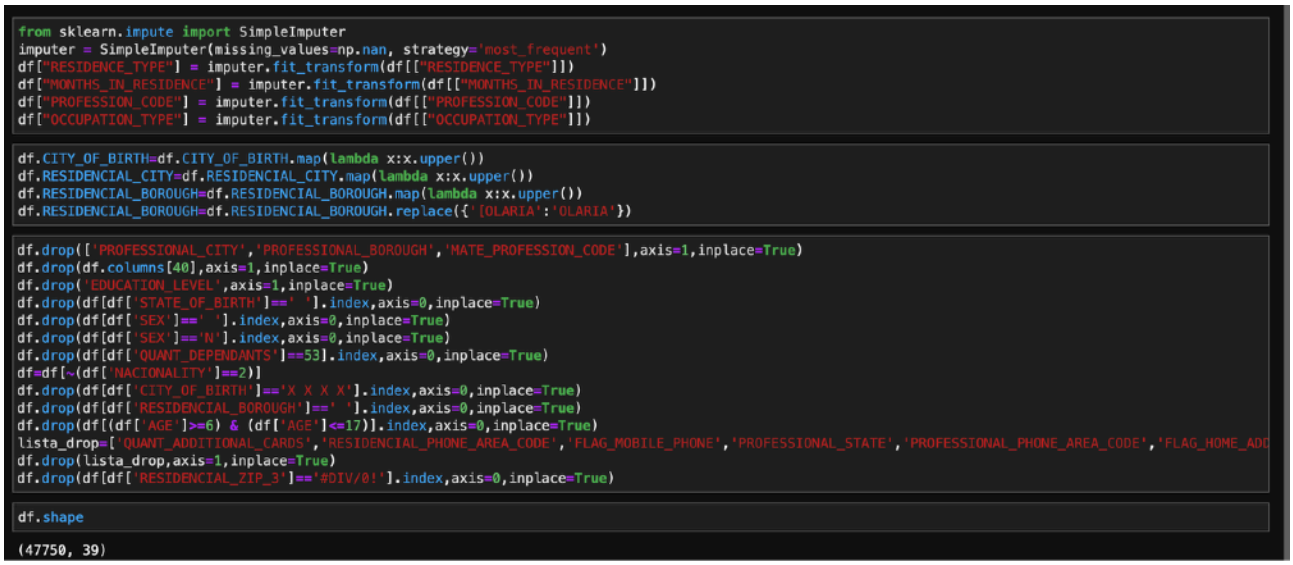
	null_sum	null_pct	count	dtypes
RESIDENCE_TYPE	1349	0.02698	48651	float64
MONTHS_IN_RESIDENCE	3777	0.07554	46223	float64
PROFESSIONAL_CITY	33783	0.67566	16217	object
PROFESSIONAL_BOROUGH	33783	0.67566	16217	object
PROFESSION_CODE	7756	0.15512	42244	float64
OCCUPATION_TYPE	7313	0.14626	42687	float64
MATE_PROFESSION_CODE	28884	0.57768	21116	float64
EDUCATION_LEVEL_del	32338	0.64676	17662	float64

Ante la información obtenida, el alumno toma la decisión de eliminar las columnas que cuyo porcentaje de valores nulos supera el 16% y a los restantes se le sustituye el valor nulo por el valor modal de la columna, con la función ***Simple Imputer*** de ***sklearn***.

A continuación de resolver la problemática de los valores nulos, se observa en el dataframe que los datos categóricos introducidos tienen distintos valores según columna, es decir, valores dummie categóricos 1,0 en una columna y valores categóricos 1,2 en otra. Además se percibe que en los valores de las columnas que hablan de la población, existen nombres con mayúscula y minúscula lo que va ha hacer que el algoritmo de clasificación interprete que son valores distintos, para solucionar esta problemática recurrimos a las funciones ***lambda*** de python para poner todos los nombres de poblaciones en mayusculas y así homogeneizar los nombres , para poder ver si existen algún tipo de fallo en la toma de datos.

El siguiente paso es el borrado de todos los datos que tengan valores que no concuerden con el objeto de estudio (edades muy jóvenes), además de distintos fallos en la toma de datos ya sea por símbolos añadidos o espacios en blanco que actúan como valores nulos.

El proceso anteriormente descrito se puede observar en la imagen, que en su parte inferior se puede ver mediante la función ***shape*** de la librería pandas, que el dataframe se queda en 44750 filas y 39 columnas, de las anteriores 50000 filas y 54 columnas.



```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
df["RESIDENCE_TYPE"] = imputer.fit_transform(df[["RESIDENCE_TYPE"]])
df["MONTHS_IN_RESIDENCE"] = imputer.fit_transform(df[["MONTHS_IN_RESIDENCE"]])
df["PROFESSION_CODE"] = imputer.fit_transform(df[["PROFESSION_CODE"]])
df["OCCUPATION_TYPE"] = imputer.fit_transform(df[["OCCUPATION_TYPE"]])
```

```
df.CITY_OF_BIRTH=df.CITY_OF_BIRTH.map(lambda x:x.upper())
df.RESIDENCIAL_CITY=df.RESIDENCIAL_CITY.map(lambda x:x.upper())
df.RESIDENCIAL_BOROUGH=df.RESIDENCIAL_BOROUGH.map(lambda x:x.upper())
df.RESIDENCIAL_BOROUGH=df.RESIDENCIAL_BOROUGH.replace({'[OLARIA':'OLARIA'})
```

```
df.drop(['PROFESSIONAL_CITY','PROFESSIONAL_BOROUGH','MATE_PROFESSION_CODE'],axis=1,inplace=True)
df.drop(df.columns[40],axis=1,inplace=True)
df.drop('EDUCATION_LEVEL',axis=1,inplace=True)
df.drop(df[df['STATE_OF_BIRTH']==' '].index,axis=0,inplace=True)
df.drop(df[df['SEX']==' '].index,axis=0,inplace=True)
df.drop(df[df['SEX']=='N'].index,axis=0,inplace=True)
df.drop(df[df['QUANT_DEPENDANTS']==53].index,axis=0,inplace=True)
df=df[~(df['NACIONALITY']==2)]
df.drop(df[df['CITY_OF_BIRTH']=='X X X X'].index,axis=0,inplace=True)
df.drop(df[df['RESIDENCIAL_BOROUGH']==' '].index,axis=0,inplace=True)
df.drop(df[(df['AGE']>=6) & (df['AGE']<=17)].index,axis=0,inplace=True)
lista_drop=['QUANT_ADDITIONAL_CARDS','RESIDENCIAL_PHONE_AREA_CODE','FLAG_MOBILE_PHONE','PROFESSIONAL_STATE','PROFESSIONAL_PHONE_AREA_CODE','FLAG_HOME_ADD
df.drop(lista_drop,axis=1,inplace=True)
df.drop(df[df['RESIDENCIAL_ZIP_3']=='#DIV/0!'].index,axis=0,inplace=True)
```

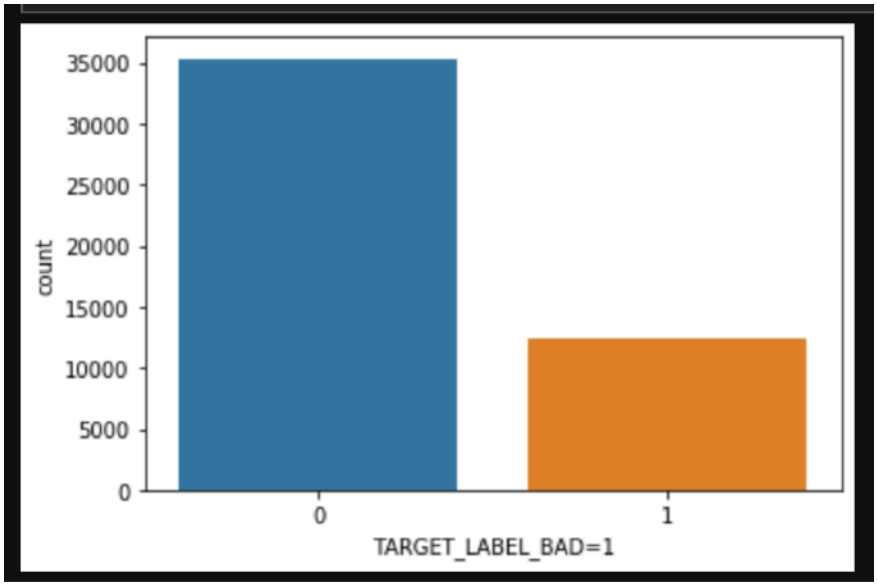
```
df.shape
```

```
(47750, 39)
```

2.2.- VISUALIZACIÓN DE LOS DATOS

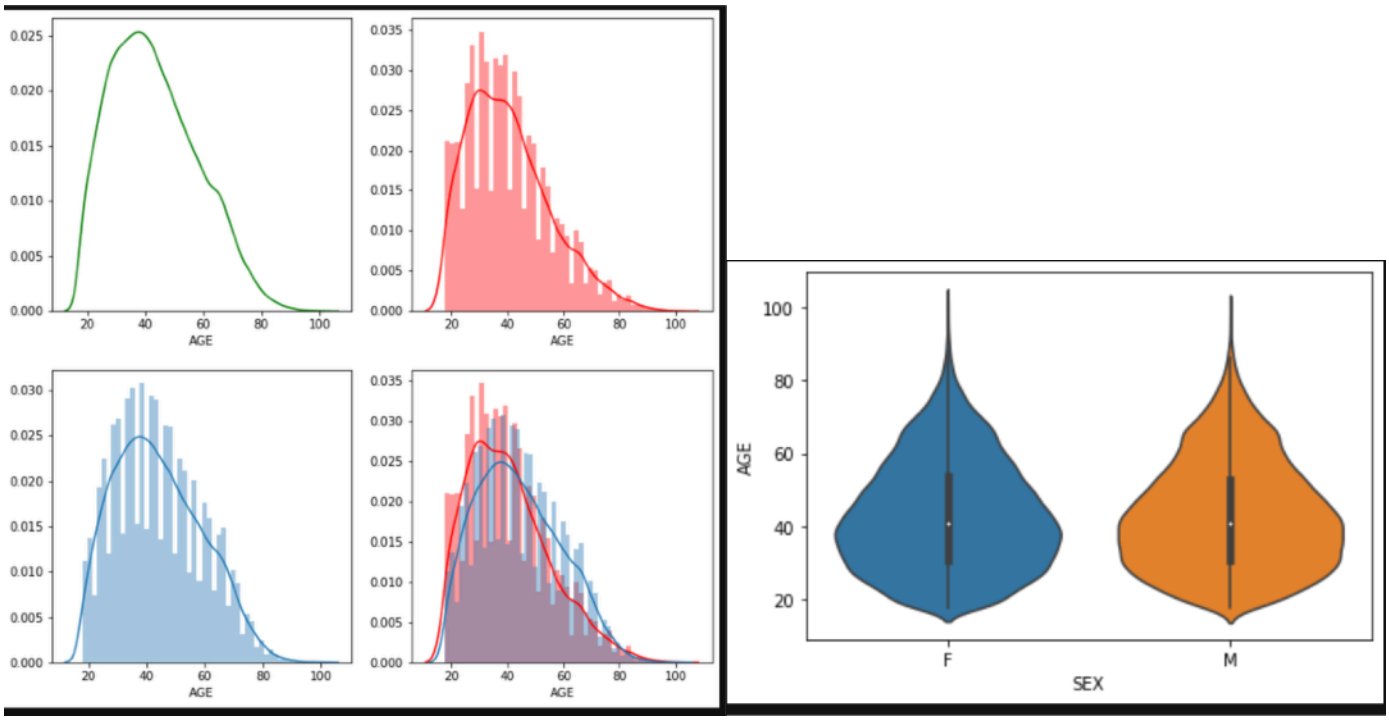
La visualización de la información del dataframe es una parte primordial de cualquier estudio de data science, en este caso se ha optado por empezar desde una visualización de datos más genérica, hacia una más específica de los datos del dataframe. La abundancia de variables categóricas del dataframe condiciona la tipología de los ***plots*** utilizados.

En primer lugar quise ver en un diagrama de barras la cantidad de patrones en mora y los que no lo están en la variables de clase.



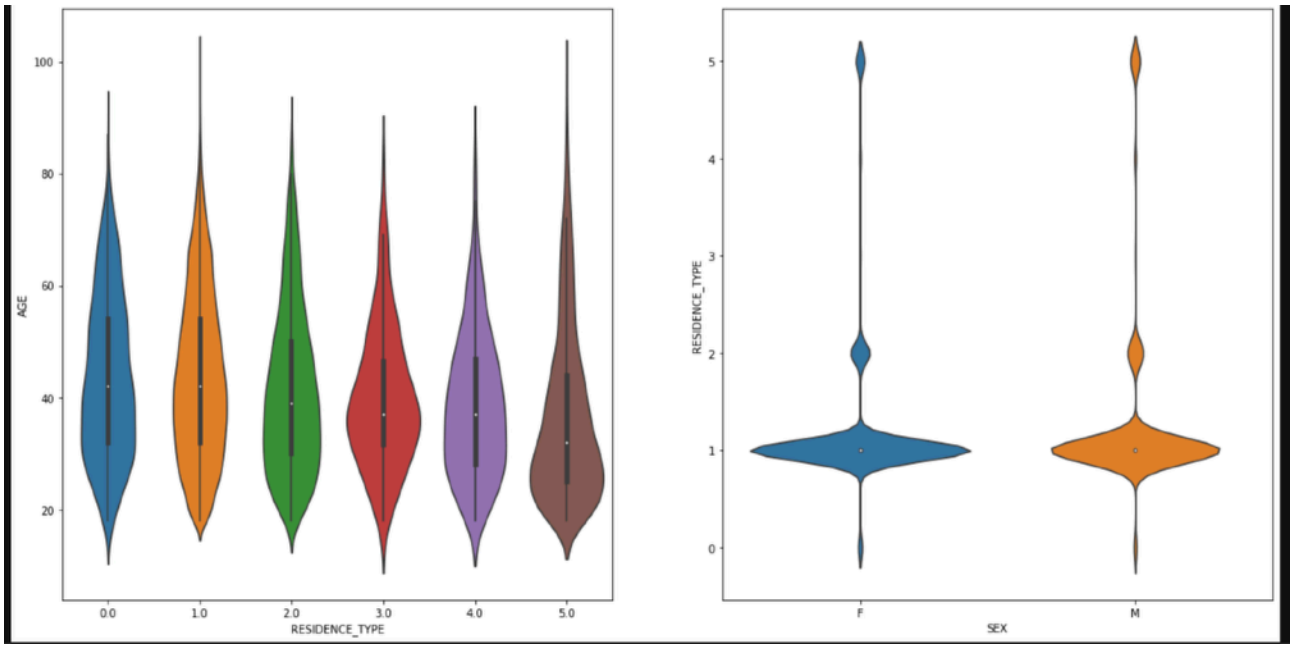
Como se puede observar existe un desbalance de clase hacia los parámetros que no están en mora, esto es algo que tiene toda lógica ya que si la mayoría de los clientes con tarjeta de crédito están en mora el negocio bancario sería ruinoso. En cuanto a la perspectiva del data scientist el desbalance de clase condiciona la posterior elección de algoritmo de clasificación y las métricas de resultados del mismo. Esta problemática tiene solución que en la parte de modernización se tratará in extenso.

En segundo lugar se quiso visualizar la información de los clientes respecto a la distribución de la edad con una consecución de histogramas en los cuales, en primer lugar se ve una distribución de la edad genérica del dataframe, luego una distribución de la edad de los clientes en mora (rojo), en azul una distribución de la edad de los clientes que no están en estado de morosidad y por último una superposición de las gráficas de mora y sin mora.

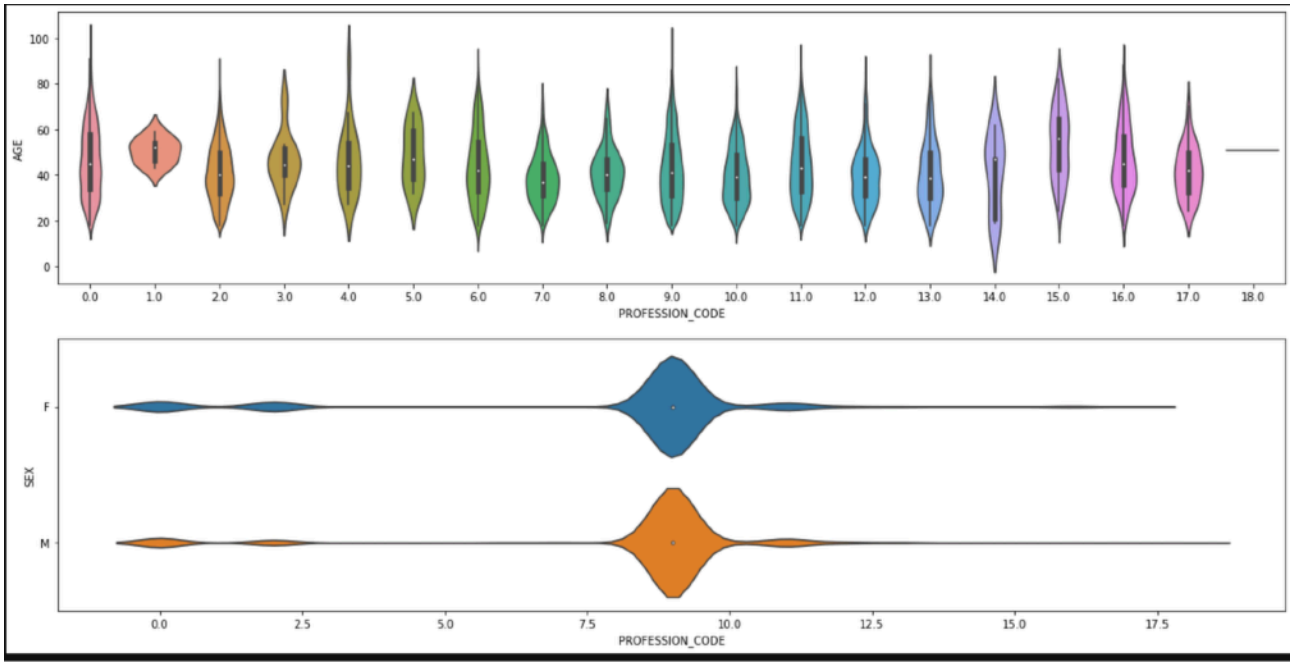


Se puede observar a simple vista en las gráficas superpuestas que la edad de los individuos en mora es ligeramente inferior y que además dicha curva tiene una doble joroba para luego descender de manera más acentuada lo que da a entender que los individuos a medida que supera la edad de unos 50 años tiende a tener una menor morosidad en las tarjetas de crédito. Al lado del conjunto de curvas un ***plot*** en forma de violín que nos indica la distribución de la edad con respecto al sexo en el dataframe, lo que nos indica que en cuanto al sexo el dataset está balanceado.

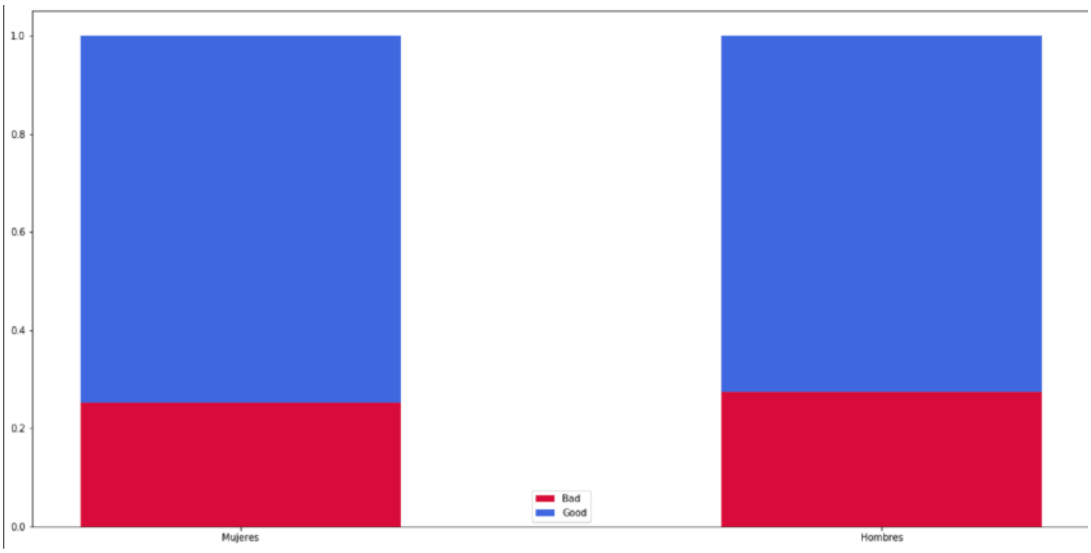
A continuación viene una consecución de ***plots*** en forma de violín para ver la distribución de dichos atributos, en primer lugar se hace una visualización de el tipo de propiedad (en propiedad, hipoteca, alquiler, propiedad de un familiar…) de la columna (residence_type) de la primera vivienda con respecto a la edad. La gráfica de la derecha es esencialmente la misma pero con respecto al sexo. Existe una mayor densidad de tipo 1 en los individuos femeninos.

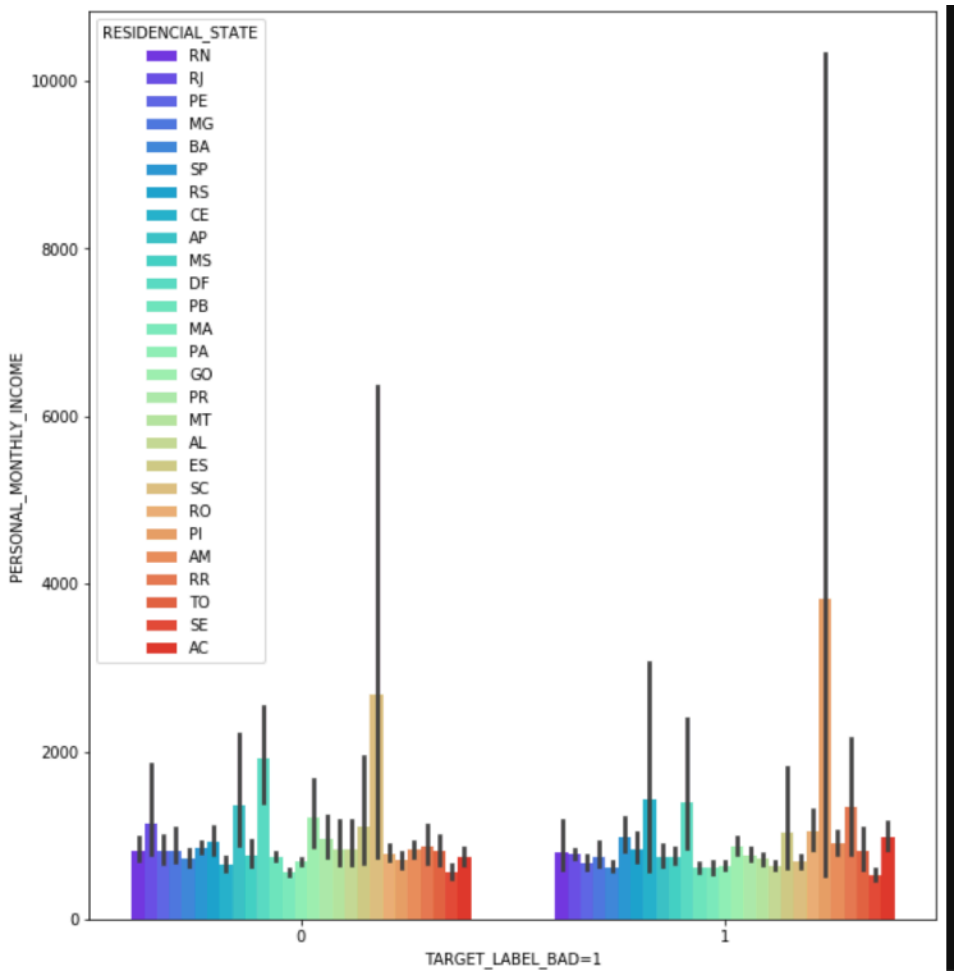


El segundo de los ploteos de tipo violín son la visualización de los tipos de profesiones (en código), con su contrapartida con diferenciación por sexos.

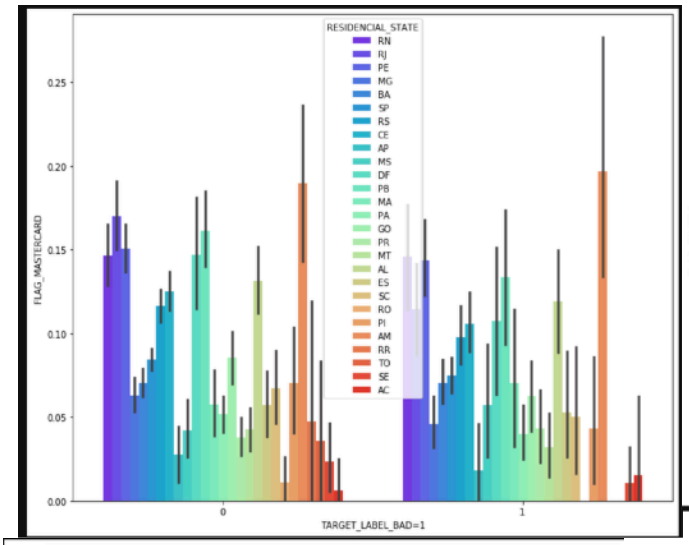


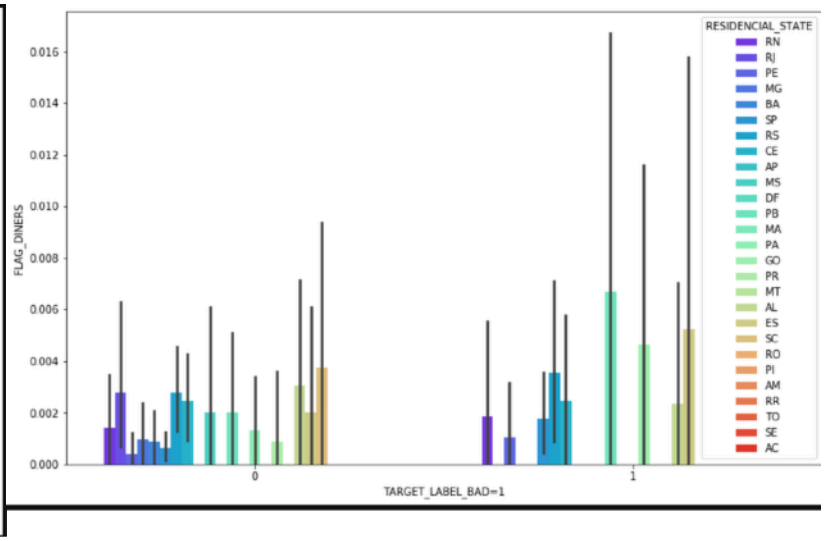
Por último en cuanto a visualización de datos genéricos dos diagramas de barras. En el primero una visualización de la proporciones de moroso y solventes por sexos, dándonos a indicar de que la morosidad no tiene nada que ver con el sexo del cliente. En el segundo una proporción entre morosos y solventes con respecto a los ingresos mensuales y distribuidos por los estados de Brasil.

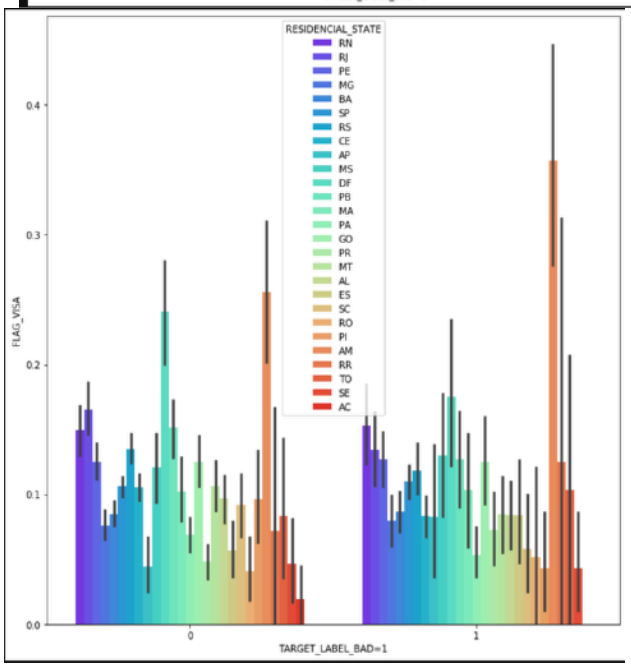


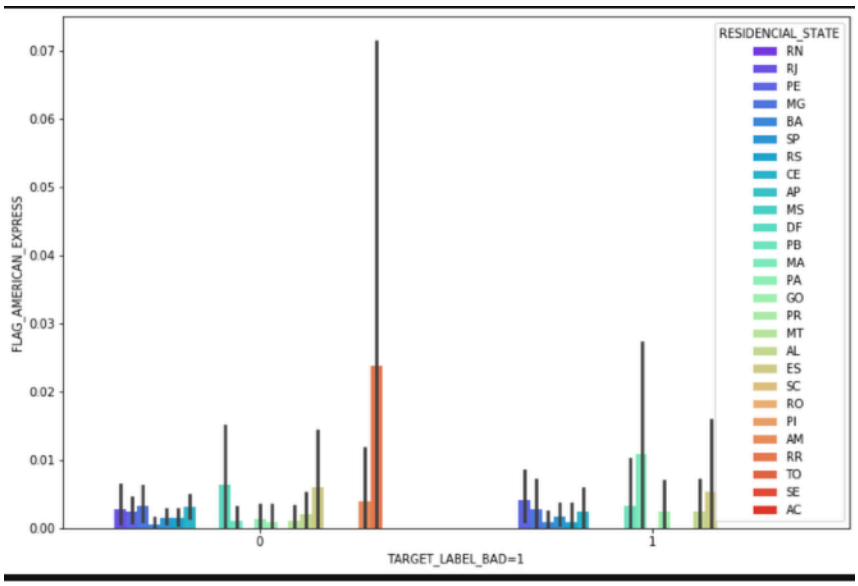


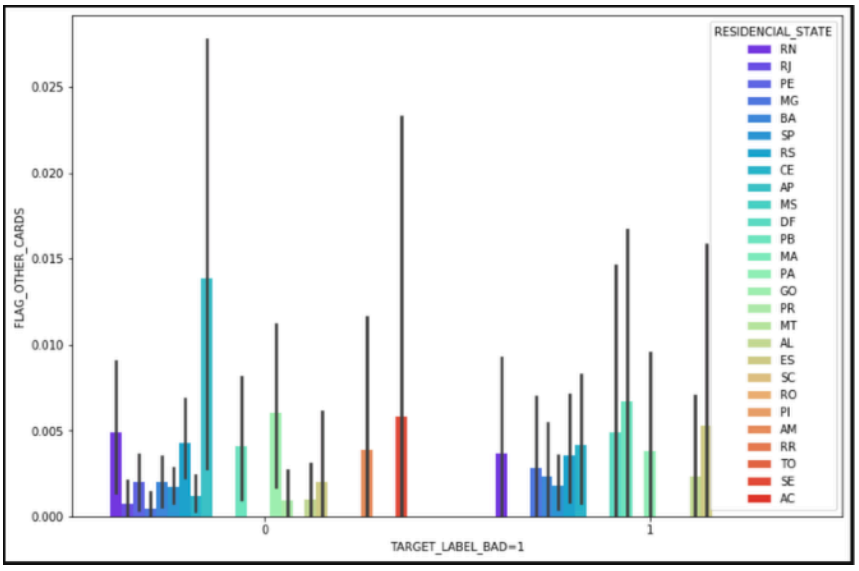
En cuanto a la visualización de datos con un carácter más específico he decidido fijarme en las variables categóricas del dataset que reflejaban las marcas de tarjetas crédito (Visa, Mastercard, etc) y para ello replicaba la metodología genérica anterior, es decir, diagrama de barras de fallido o no fallido con respecto a la marca de tarjeta y a su vez con distribución geográfica y por otra parte gráficas de tipo violín para cada marca de tarjeta de crédito para cada una de las profesiones y a su vez con distribución con respecto al sexo.

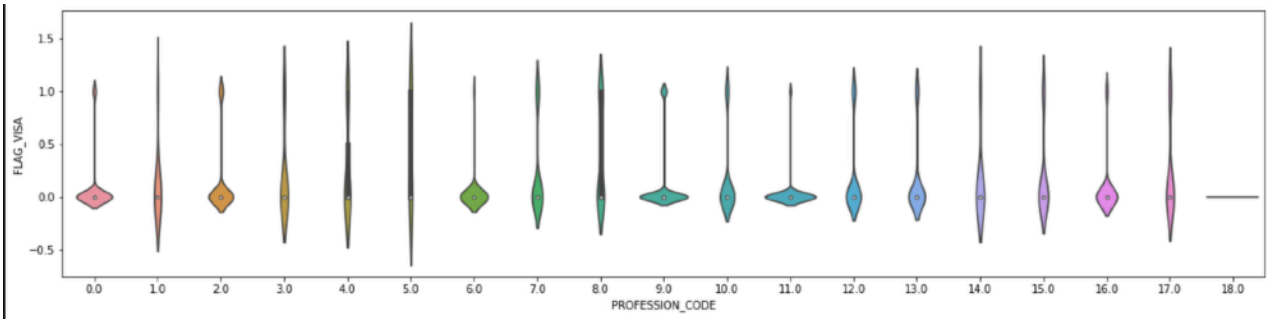


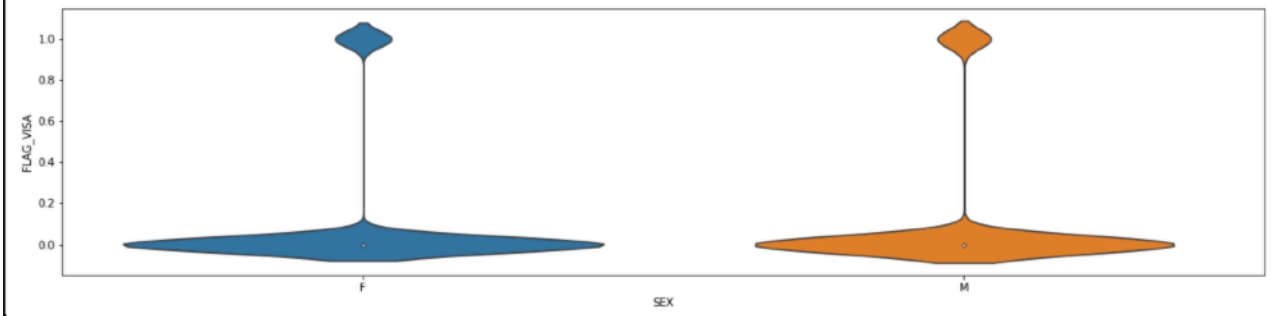


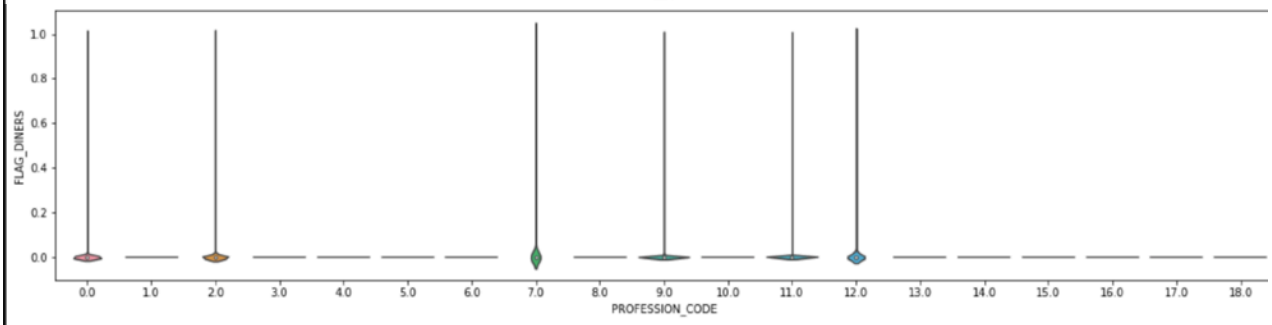


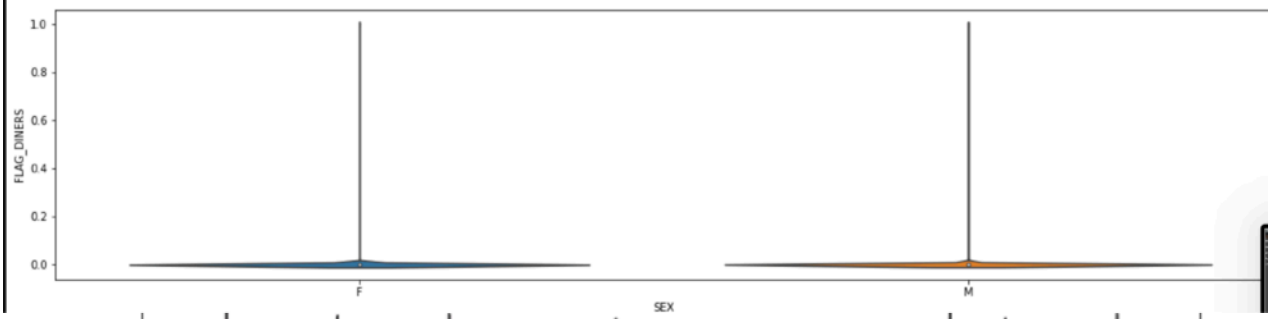


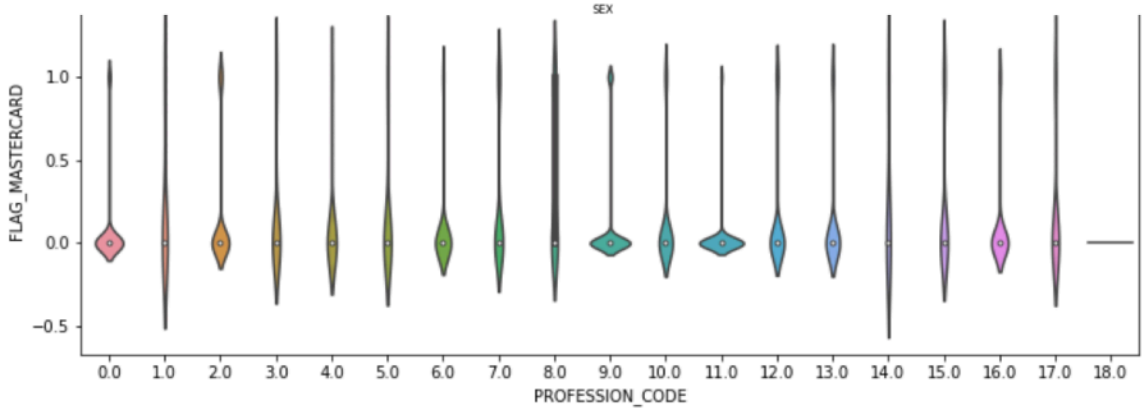


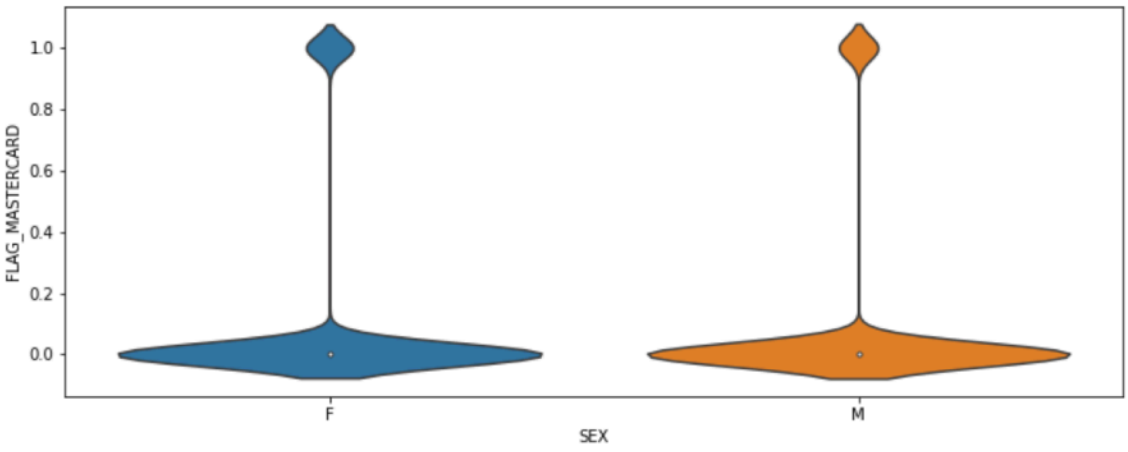

FLAG_VISA
PROFESSION_CODE

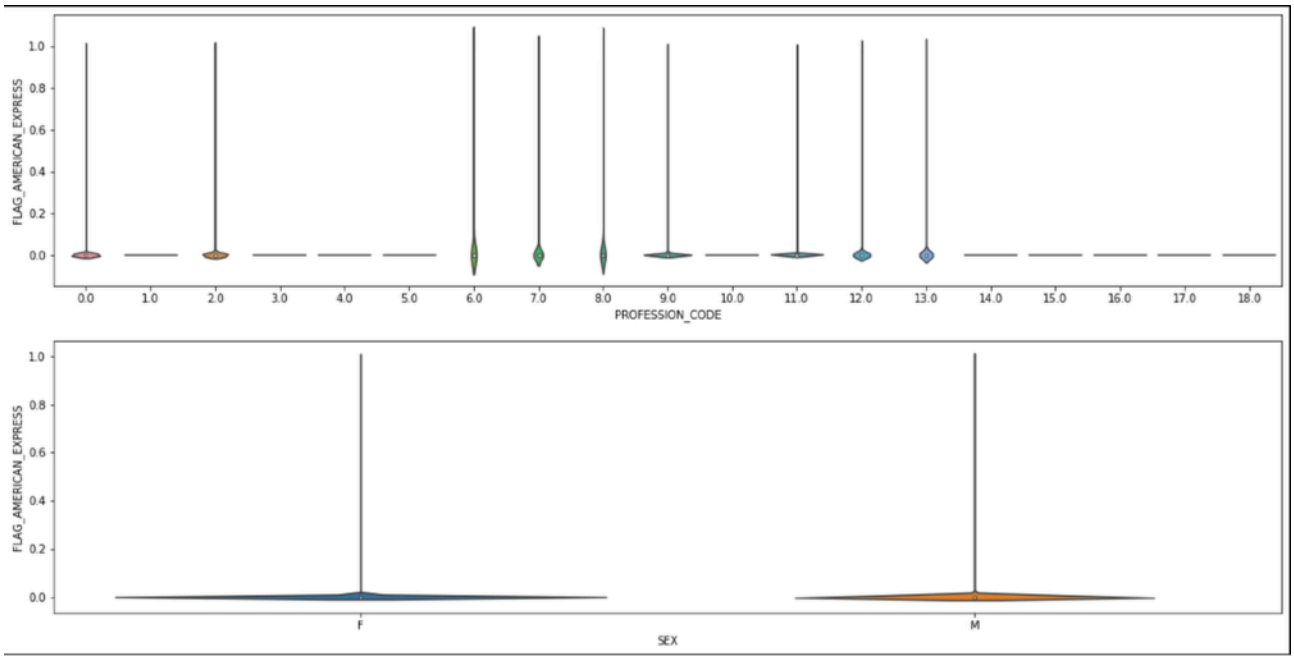

FLAG_VISA
F
M
SEX

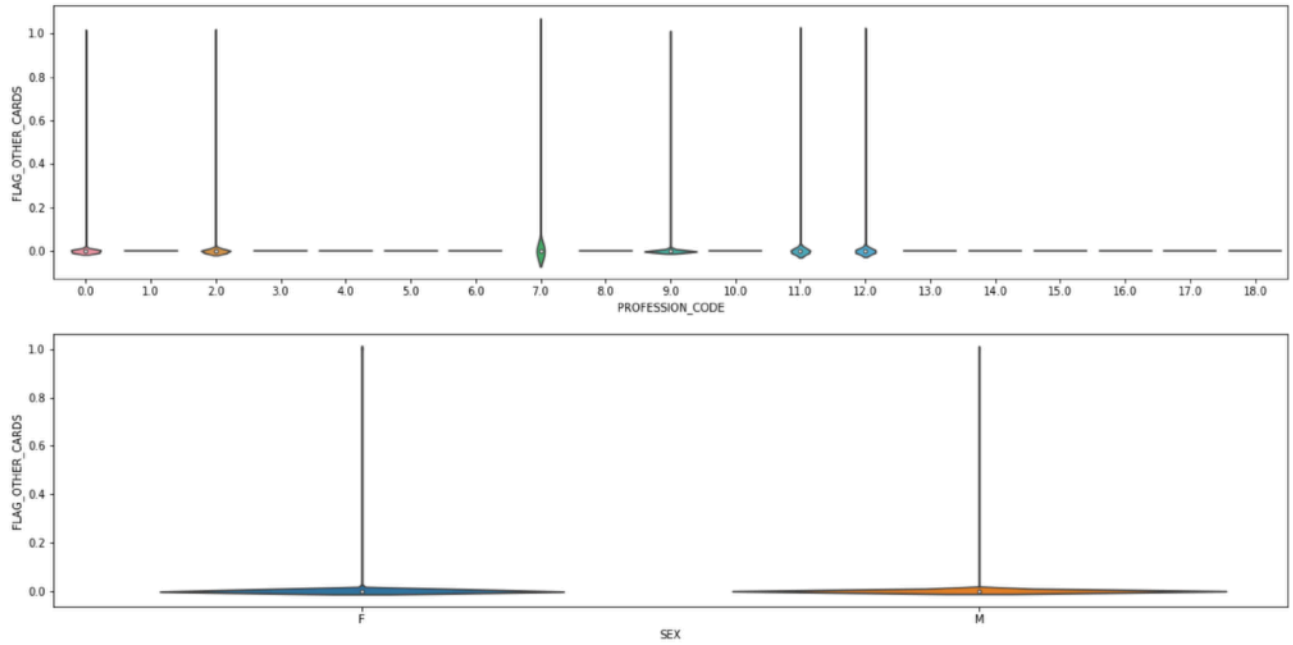

FLAG_DINERS
PROFESSION_CODE


FLAG_DINERS
F
M
SEX


FLAG_MASTERCARD
PROFESSION_CODE


FLAG_MASTERCARD
F
M
SEX





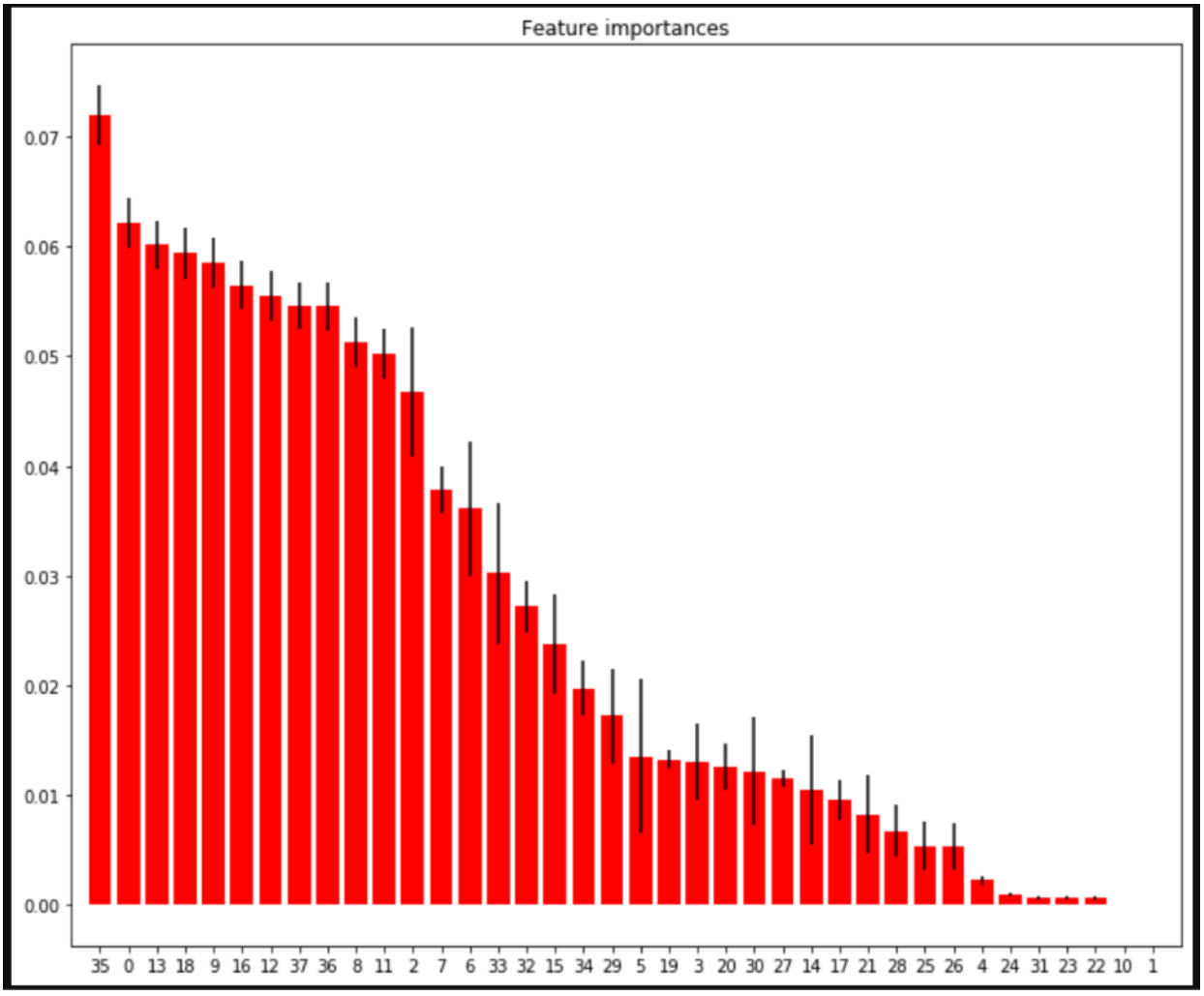
Tal y como se puede observar las marcas mas utilizados por los individuos son visa y mastercard, cosa que no es extraña dado que son las dos marcas que copan la mayor cuota de mercado. Otra cosa que se puede intuir de esta visualización ,dado que la tipología de profesión en la documentación se da en formación numérica, es que las profesiones que tienen violines en todas las marcas de tarjetas pueden ser profesiones de un cierto nivel adquisitivo, debido a las marcas que aparecen en el dataframe (American Express, Dinners), dado que alguna de ellas no sólo está relacionada con los servicios básicos de tarjetas de crédito sino también con las oferta que pueden darte como programas de puntos en base a consumo.

2.3.- Modelizacion

Para el inicio de la modelización de los algoritmos de clasificación, en primer lugar se tienen que aplicar una serie de funciones para cada uno de los tipos de atributos, en el caso de las variables numéricas se normaliza los resultados con la función ***MinMaxScaler*** , para el caso de los atributos categóricos se utiliza un ***label encoding*** para otorgarle un valor numérico a cada uno de los parámetros.

A continuación se separa el dataframe en dos variables X e Y, en las cuales estarán, la variable de clase en la Y, y el resto de datos en la variable X. Ambos transformados en ***array*** con la función ***value*** de pandas.

Una vez ya se tienen las variables X e Y, se obtiene el ***feature importances*** para obtener las columnas más importantes del dataframe con respecto a la variable de clase.



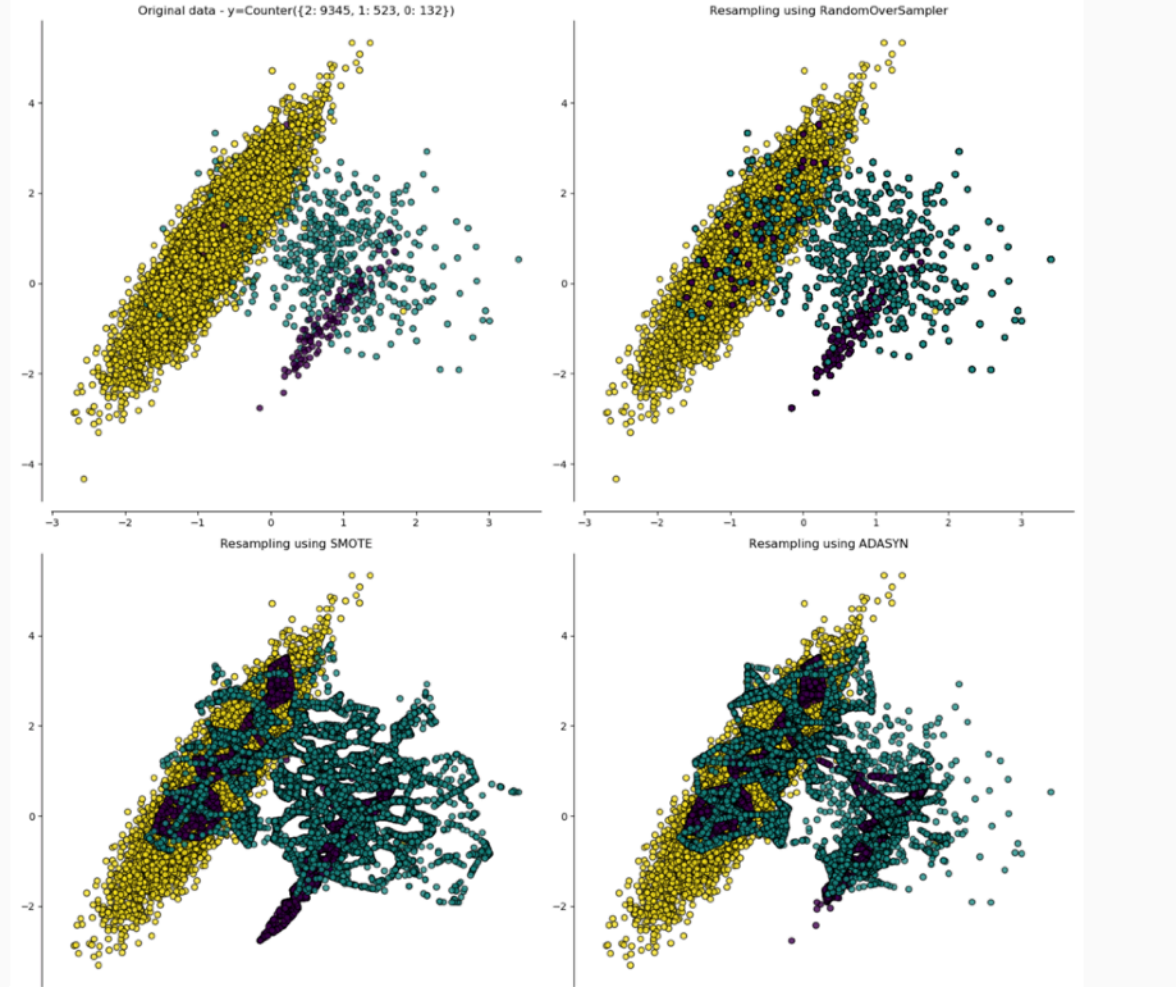
Como se puede observar en la gráfica anterior existen dos escalones muy marcados en cuanto a importancia de los atributos en el dataframe, por ello se ha planteado la siguiente estrategia en la modelización; en primer lugar se va a experimentar con el dataframe sin ningún tipo de corte, es decir, con todas las columnas, en segundo lugar se va ha hacer el estudio de clasificación practicando un borrado de todas las columnas a partir de la columna número 2 tal y como se ve en la gráfica, y por último se va a proceder al estudio de la clasificación practicando un borrado de las columnas a partir de la columna 34 según la gráfica.
Por lo tanto, se configuran 3 dataframe de estudio. El primero sin modificaciones de 47750 filas y 39 columnas, el segundo (corte en 2) con 47750 filas y 13 columnas y el último con las mismas filas pero con 19 columnas.

Otra problemática surgida por la composición de los datos es, como ya se vio en la fase de visualización de los datos, el desbalance de clase. Esto es así dado que cuando se practica el entrenamiento del algoritmo de clasificación seleccionado este tiende a clasificar hacia la clase prevalente (la más numerosa), lo que en este caso llevaría a decir que ningún cliente vaya a entrar en mora en un futuro, cosa que se contrapone con la realidad, además de con el propósito de este proyecto y haría carente de valor todo el trabajo realizado.

Para la resolución de este problema se realiza un sobredimensionado de la clase minoritaria , en este caso el parámetro default, mediante la creación de parámetros denominados "sintéticos" para que exista el mismo número de casos con default (mora), que sin ellos. Esto hace, en teoría, que el algoritmo no tienda hacia una determinada clasificación de una manera tan marcada. Para ello se utiliza la librería , ***imbalanced-learn*** , que a su vez tiene tres variantes para la elaboración de patrones sintéticos, todas ellas utilizadas en este estudio.

La primera metodología se denomina ***Random Over-Samplig***, que consiste en la generación de duplicados a partir de la clase minoritaria ya existente de una manera aleatoria.

La segunda metodología se denomina ***the Synthetic Minority Oversampling Technique*** , a partir de ahora por su acrónimo ***SMOTE***, que genera sintéticos de la clase minoritaria no a partir de la duplicación sino de la interpolación lineal de lo parámetros más cercanos de la clase minoritaria bajo la metodología ***KNN.*** Esto lo hace en teoría más eficaz. La tercera y última metodología es ***ADASYN Adaptive Synthetic,*** que utiliza una metodología a similar al SMOTE pero con la particularidad de que al utilizar el KNN hace distinción entre los puntos de las muestras de la variable de clase que son fáciles de clasificar y las que no.



```
[42]: X=df.iloc[:,:-1].values
      y=df['TARGET_LABEL_BAD=1'].values
      Counter(y)

[42]: Counter({1: 12446, 0: 35304})

[43]: X_resampled1, y_resampled1 = SMOTE().fit_resample(X, y)

[44]: Counter(y_resampled1)

[44]: Counter({1: 35304, 0: 35304})
```

Por último hay que hablar de los algoritmos de clasificación utilizados, las métricas de validación observadas y el método de validación elegido. Empezando por el último en la enumeración, el método de validación elegido es el ***Kfold cross validation*** con 10 separaciones para todos los experimentos.

Las métricas utilizadas para la evaluación de los experimentos son básicamente el ***Recall*** para las variables (0,1) ,es decir no moroso y moroso, el ***F-1 Score*** para las variables (0,1) y el ***F-1 Score*** del accuracy, todas ellas extraídas del ***Classification Report*** de la librería ***sklearn.*** Así se puede tener una visión amplia de el sesgo de cada uno de los algoritmos en la clasificación hacia una clase u otra.

Como es lógico hay que hacer mención a los algoritmos de clasificación utilizados para el estudio del dataframe, para ello se ha seleccionado los algoritmos que menos se ven influenciados por el desbalance de clase, y a continuación ya con el sobredimensionados realizado se ha añadido otro algoritmo de clasificación para ver si con las clases balanceadas existía una mejoría en las métricas. Los algoritmos de clasificación utilizados son los siguientes:

- ***Naive Bayes***
- ***Arbol de decisión***
- ***Random Forest***
- ***KNN (en clases balanceadas)***
- ***Xgboost***
- ***Adaboost***

Por último se utilizara un ***Vooting*** con los algoritmos con las mejores métricas para ver si en conjunto pueden elevar la clasificación.

Recapitulando la exposición anterior sobre como se va a acometer el modelado del dataframe es la siguiente; se van a utilizar 3 dataframe con los cortes de ***feature importance,*** se va a utilizar el método ***kfold*** para la validación, y se van a aplicar los algoritmos de clasificación anteriores a las siguiente situaciones; sin elaboración de sintéticos (desbalance de clase), con ***Random Over-Sample***, con ***SMOTE*** y con ***ADASYN***, lo que da un total de 75 experimentos con los 3 Dataframes anteriormente mencionados.

2.4.- Resultados y Conclusiones del primer proyecto

De los 75 experimentos realizados se extrajeron los mejores resultados que se indican en la siguiente tabla:

RESULTADOS DE LOS CLASIFICADORES							
			RECALL		F-1 SCORE		F-1 (ACCURACY)
			0	1	0	1	
DATA SET SIN MODIFICACIÓN DE FEATURE IMPORTANCE	SMOTE	NAIVE BAYES	0,199	0,811	0,287	0,621	0,505
		ARBOL DECISIÓN	0,521	0,816	0,611	0,711	0,668
		RANDOM FOREST	0,717	0,759	0,732	0,743	0,738
		KNN	0,028	0,942	0,052	0,646	0,485
		XGBOOST	0,722	0,742	0,729	0,734	0,732
		ADABOOST	0,690	0,743	0,708	0,724	0,716
	ADASYN	NAIVE BAYES	0,220	0,528	0,266	0,444	0,367
		ARBOL DECISIÓN	0,507	0,644	0,552	0,590	0,572
		RANDOM FOREST	0,710	0,720	0,722	0,708	0,715
		KNN	0,022	0,694	0,034	0,503	0,344
		XGBOOST	0,712	0,714	0,721	0,704	0,713
		ADABOOST	0,685	0,703	0,700	0,687	0,693

Como se puede observar los mejores resultados obtenidos han sido en el dataframe en el que no se le ha aplicado el ***feature importance***, en cuanto a las técnicas de sobredimensionado, ***SMOTE*** es la que mejor se comporta, aunque ***ADASYN*** no está muy lejos de sus resultados, en el caso de la metodología ***Random Over-Sample*** , tiende ha cambiar el sesgo de las métricas hacia la minoritaria lo que lo hace una metodología poco sólida. En cuanto a los clasificadores, las mejores y más equilibradas métricas obtenidas en el estudio, son; ***Random Forest, Xgboost*** y ***Adaboost***, todas ellas resaltados en la tabla resumen.

Como conclusión de este primer estudio, se da a entender que al ser un dataset para una competición de machine learning, las puntuaciones obtenidas no van a ser lo más altas posibles, además de que existe un exceso de variables categóricas sobre numéricas , que en muchos casos no aportaban absolutamente nada, ya que la totalidad de la columna tenía el mismo dato.

3.0.- Descripción del segundo proyecto

El segundo dataset consiste en la toma de 50000 créditos hipotecarios en los Estados Unidos durante un periodo determinado (60 periodos/12 meses=5años). Esto quiere decir que tal y como pasa en la vida real algunos clientes(id) han recibido el crédito antes de la primera fecha de toma de datos y otros después de la misma, lo que conlleva es que el número de filas por individuo no es homogéneo en todo el dataframe. Las columnas de datos extraídos de la pagina web son las siguientes:

id	Prestatatio ID
time	Time stamp de la observación
orig_time	Time stamp del origen de la posición
first_time	Time stamp de la primera observación
mat_time	Time stamp que falta para la madurez del activo
balance_time	Deuda viva en el momento de la observación
LTV_time	Loan-to-value ratio en el momento de la observacion, en %
interest_rate_time	Tipo de interés en el momento de la observación, en %
hpi_time	Indice de precios de los inmuebles en el momento de la observación, base year = 100
gdp_time	Crecimiento del PIB en el momento de la observación , en %
uer_time	Tasa de desempleo en el momento de la observación, in %
REtype_CO_orig_time	Tipo de inmueble, condominium = 1, otherwise = 0
REtype_PU_orig_time	Tipo de planificación urbana, development = 1, otherwise = 0
REtype_SF_orig_time	Vivienda Unifamiliar ,home = 1, otherwise = 0
investor_orig_time	Inversor borrower = 1, otherwise = 0
balance_orig_time	Deuda viva en la fecha de origen
FICO_orig_time	FICO score en la fecha de origen, en %
LTV_orig_time	Loan-to-value ratio en la fecha de origen, en %
Interest_Rate_orig_time	Tipo de interés en la fecha de origen in %
hpi_orig_time	Indice de precios de los inmuebles en la fecha de origen, base year = 100
default_time	Default en la fecha de observación
payoff_time	Payoff en la fecha de observación
status_time	Default (1), payoff (2), y nondefault/nonpayoff (0) observation at observation time

Antes de empezar con la descripción del dataset primero se tienen que aclarar una serie de conceptos relacionados con la calificación crediticia de un individuo antes de pedir un préstamo hipotecario, que se encuentra en el dataframe. Uno es el denominado como Loan to Value, es una métrica que mide la calidad crediticia de un prestatario bajo una simple fracción, es sino como de la propia traducción del termino se puede extraer la proporción del préstamo hipotecario sobre el valor de la tasación del inmueble que se aporta como colateral de la deuda.

$$Loan - to - value = \frac{Loan}{Value}$$

En España el valor de tasación de la hipotecas las hace las sociedades de tasación que están bajo la supervisión del Banco de España. La práctica regulatoria dice que los préstamos con un Loan to value superior al 0,8 (80%) , son préstamos de alto riesgo y por ello conlleva una penalización de provisiones (cuenta de resultados) y capital regulatorio.

En segundo lugar esta la denominada como métrica FICO, esta es ampliamente utilizada por las diversas entidades de préstamo hipotecario de los Estados Unidos. Abajo se adjunta una imagen de un cuadro resumen para su mejor entendimiento.

FICO Score	Rating	What the Score Means
< 580	Poor	• Well below average • Demonstrates to lenders that you're a risky borrower
580 – 669	Fair	• Below average • Many lenders will approve loans
670 – 739	Good	• Near or slightly above average • Most lenders consider this a good score
740 – 799	Very Good	• Above average • Demonstrates to lenders you're a very dependable borrower
800+	Exceptional	• Well above average • Demonstrates to lenders you're an exceptional borrower

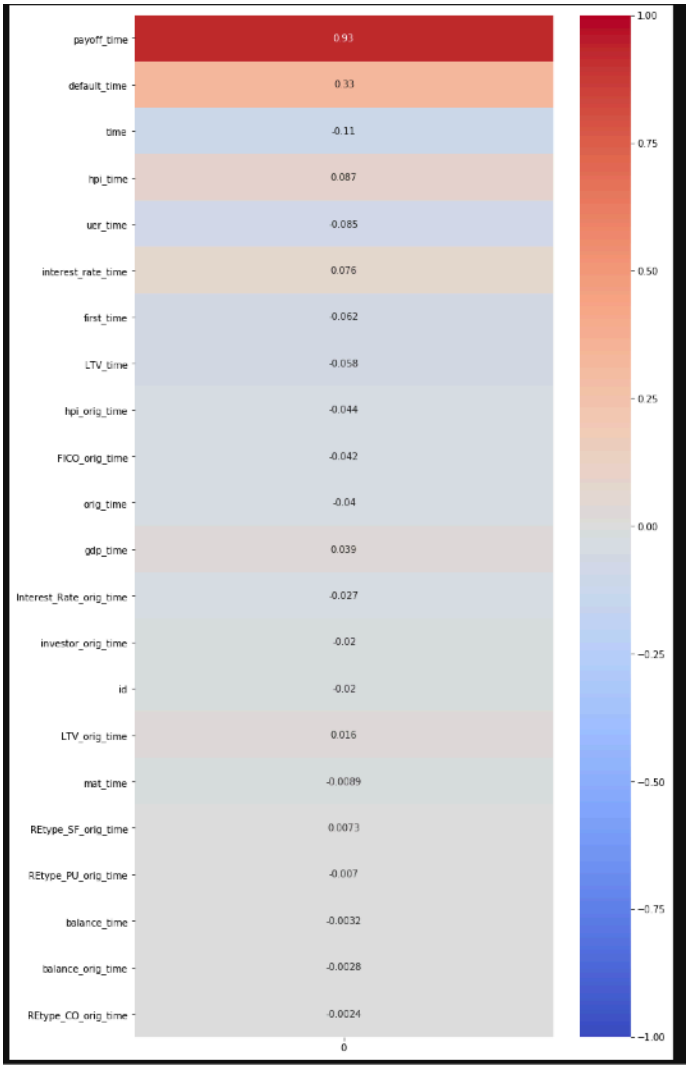
La finalidad de este segundo proyecto es mediante el machine Learning poder elaborar un modelo predictivo de riesgo de morosidad para la entidad bancaria, en base a datos tales como el LTV, el tipo de interés, crecimiento del PIB, etc.

3.1.- EDA

A la hora de empezar nuestro EDA (exploratory data analysis), utilizamos la función ***def eda*** descrita anteriormente para localizar los posibles valores nulos, y además de ver la tipología de datos que tiene el dataframe que se nos presenta.

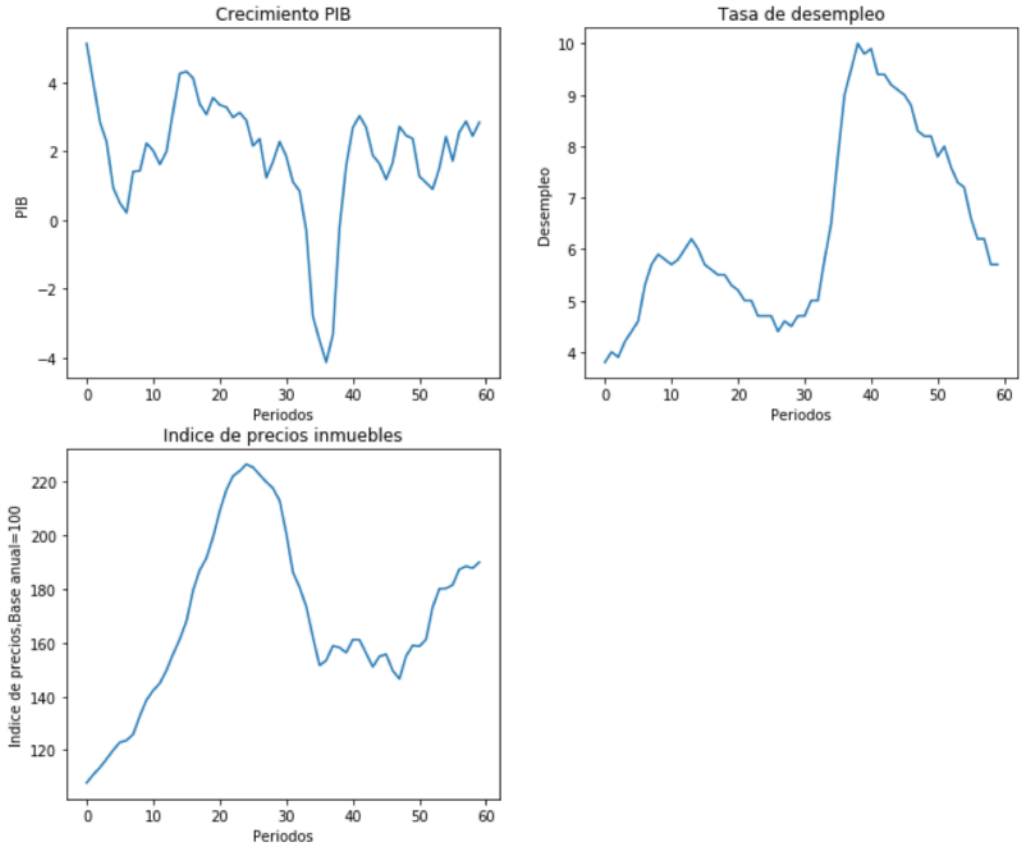
```
[5]: def eda(df):
         eda = {}
         eda['null_sum'] = df.isnull().sum()
         eda['null_pct'] = df.isnull().mean()
         eda['dtypes'] = df.dtypes
         eda['count'] = df.count()
         eda['mean'] = df.mean()
         eda['median'] = df.median()
         eda['min'] = df.min()
         eda['max'] = df.max()

         return pd.DataFrame(eda)
```

La visualización del dataframe que origina la función programada, nos indica que apenas tiene valores nulos para el tamaño que tiene el dataframe , solamente 270 en la columna LTV_time, para un dataframe de 622489 filas y 23 columnas. También se observa un gran número de atributos numéricos, frente a los categóricos del anterior dataframe. También se usó un código que permite la visualización de la gráfica de las columnas con mayor correlación con la variable de clase.

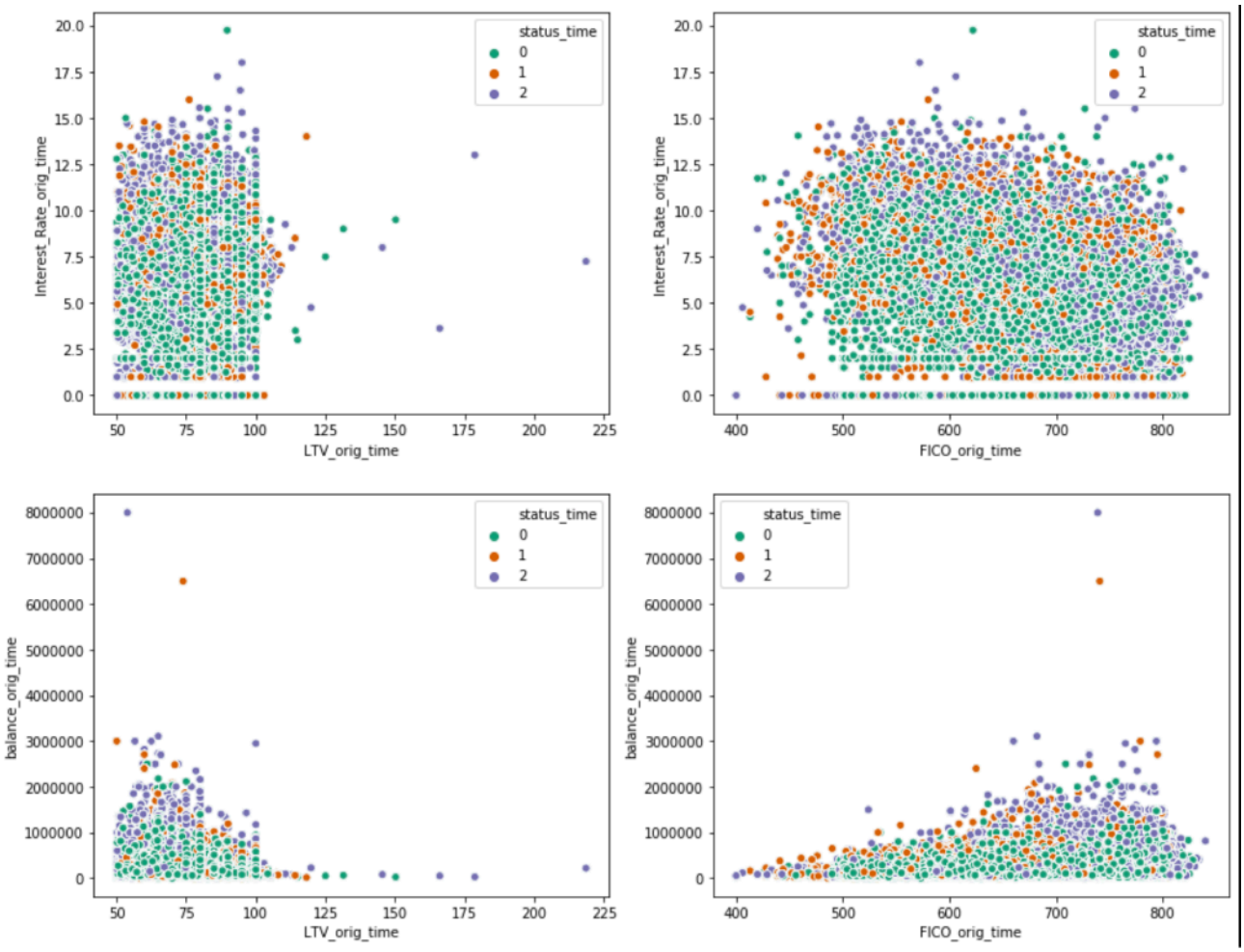


3.2.- Visualización de los datos

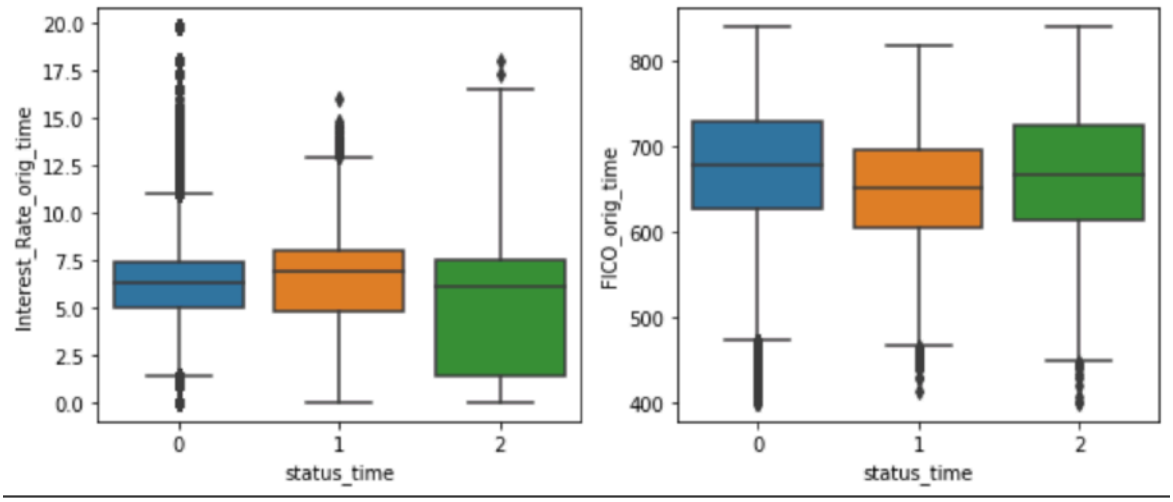
Tal y como se hizo en el dataset anterior se va a proceder a una visualización de parámetros más genéricos hacia parámetros más específicos, al contrario que en el dataframe anterior, este tiene muchas columnas con valores numéricos, lo que permite el ploteo de tipo ***scatter*** (puntos) y diagramas de cajas.

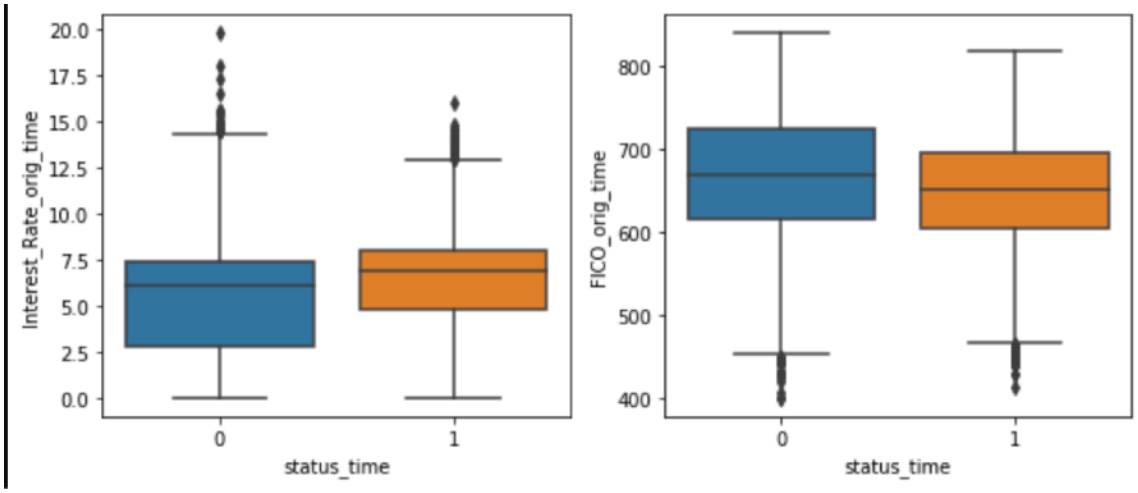


En primer lugar se hace una visualización de los datos macroeconómicos del dataframe a lo largo de los 60 periodos en los que está reverenciado. Se puede apreciar de una notablemente una caída del PIB (recesión) de hasta 6 puntos porcentuales entre el periodo 30 a 40. Esto viene acompañado de una subida del desempleo y una bajada de los indices de precios de los inmuebles durante el mismo periodo y consiguientes. Como es lógico una recesión económica provoca un aumento de la tasa de desempleo y por lo tanto disminuye la los demandantes o posibles compradores de inmuebles (es simple oferta y demanda), desde la perspectiva del banco hay una bajada de cantidad y la calidad de los prestatarios.



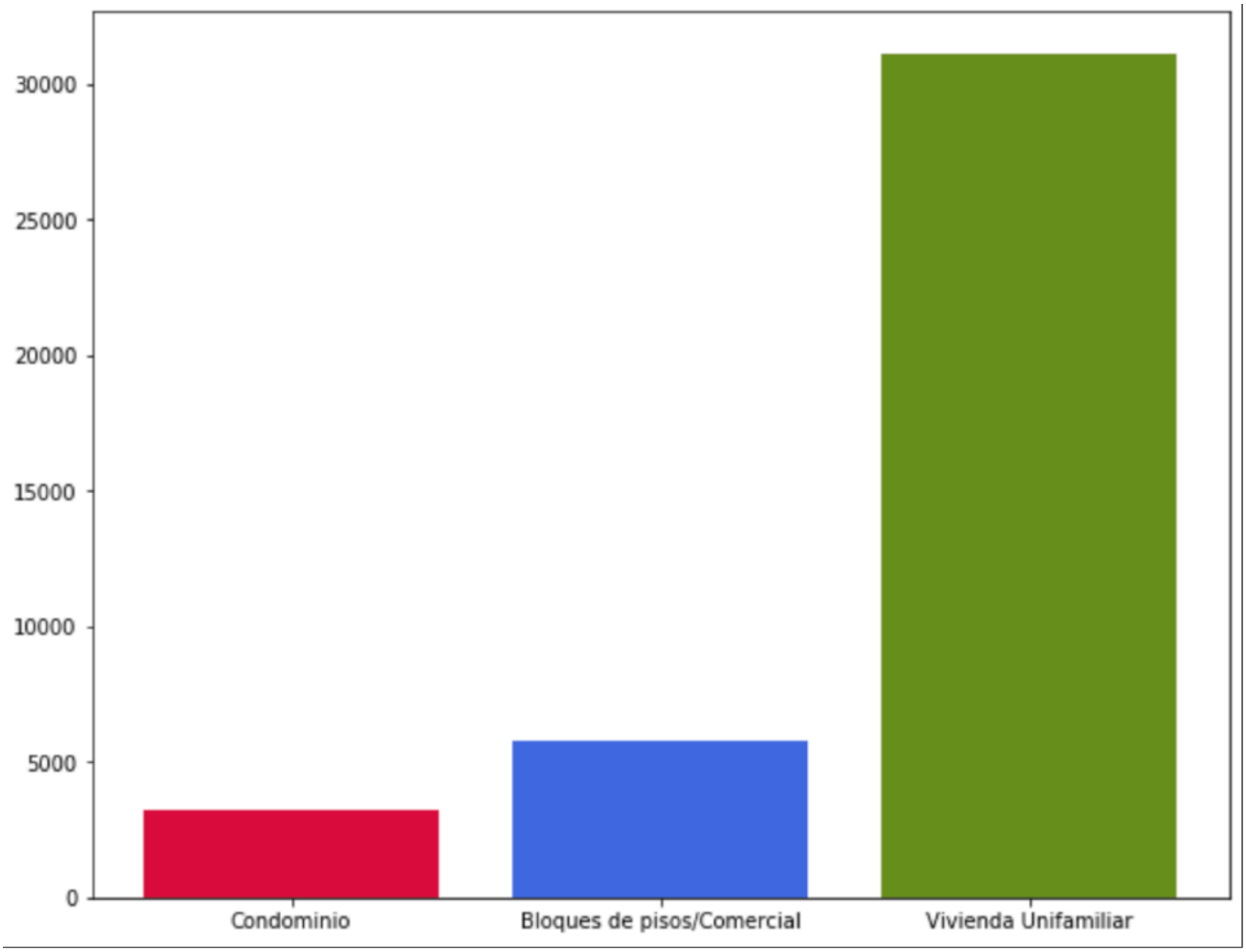
En el conjunto de plots de arriba se puede observar que las nubes de puntos están diferenciadas por cada uno de los atributos que puede tener la variable de clase. El cero es para los clientes que continuan con la deuda viva pero no está en mora ni está liquidada, el 1 es para las posiciones en mora, el 2 para las posiciones en la que se salda completamente la deuda. En los ejes de ordenadas de los cuatro gráficos se utiliza los parámetros de calidad crediticia anteriormente explicados LTV y FICO y en el de abscisas se eligen los tipos de interés para cada una de las posiciones en el momento de firma de la hipoteca y la cantidad otorgada por la Entidad de crédito. Como se puede observar en los gráficos con eje LTV, este raramente excede de un 100, lo que hace que la entidad sea consciente de los riesgo que se enfrenta cada vez que toma una posición. En las visualizaciones con el parámetro FICO se puede ver a simple vista que la nube de puntos se orienta hacia las posiciones con mayor puntuación (mejor calidad) y que las posiciones en mora, en puntos marrones, empiezan a abundar en zonas donde la puntuación es menor.





En las dos imágenes anteriores, se pueden apreciar sendas gráficas de cajas que indican los principales indicadores estadísticos (media, mediana, cuartiles, outliers) de las variables , FICO y tipo de interés en con respecto a la variable de clase, que para simplificar la metodología y la futura modelización a las posiciones que estaban en la variable 2 (payoff) se pasan a la variable 0 que es deuda viva. Dado que el propósito de este proyecto es predecir un default o no.

Por último un diagrama de barras indicando la tipología de inmuebles que existen como colaterales de las hipotecas en la cartera.



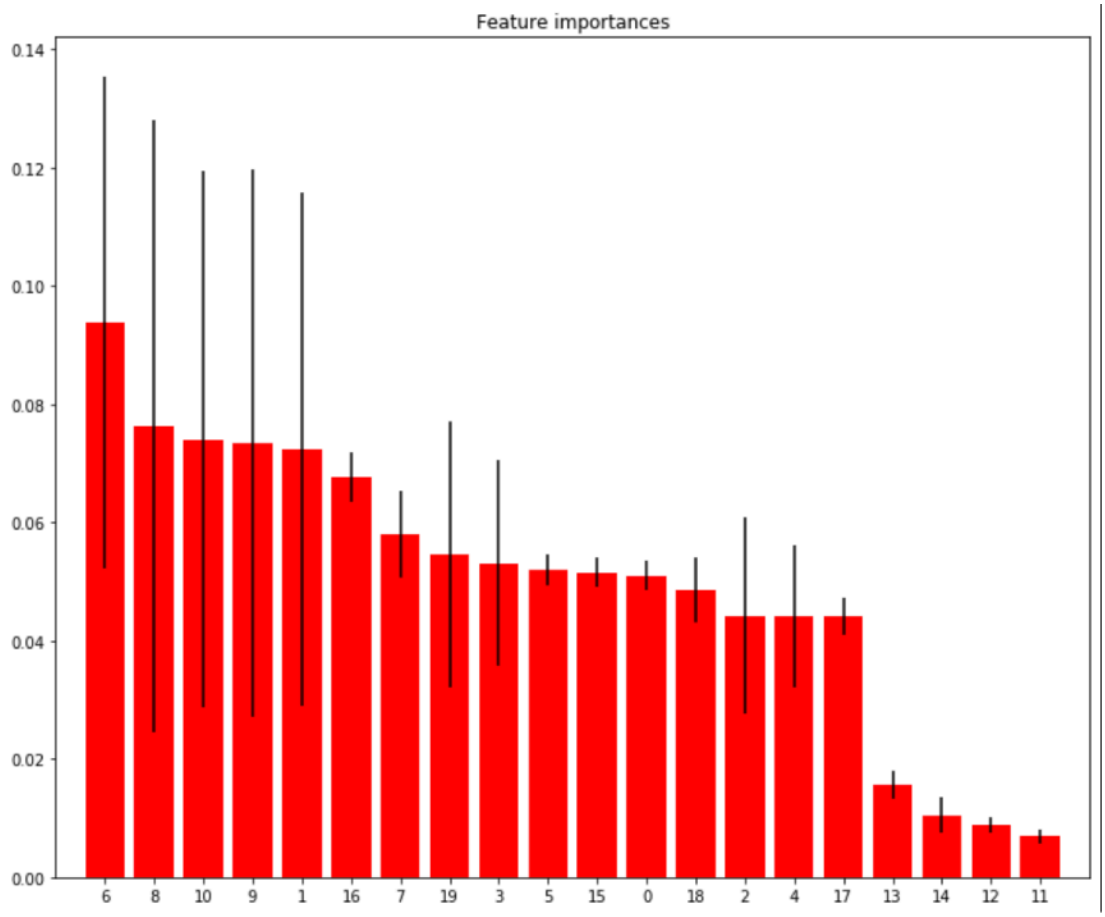
3.3.- Modelización

La particularidad de este dataframe es que al ser una "foto" de una cartera crediticia a lo largo del tiempo (60 periodos), los individuos no son homogéneos, ¿esto qué quiere decir?, que no todos los individuos de las 50000 hipotecas tiene información a lo largo de los 60 periodos que estudia el dataframe, no obstante para eliminar dicha problemática, y por lo tanto quede una información homogénea de los 50000 individuos, se ha seleccionado la última fila del dataframe de cada uno de los individuos y de ha formado un dataframe con los mismo. Así se pasa de un dataframe de 622489 filas a uno de 50000.

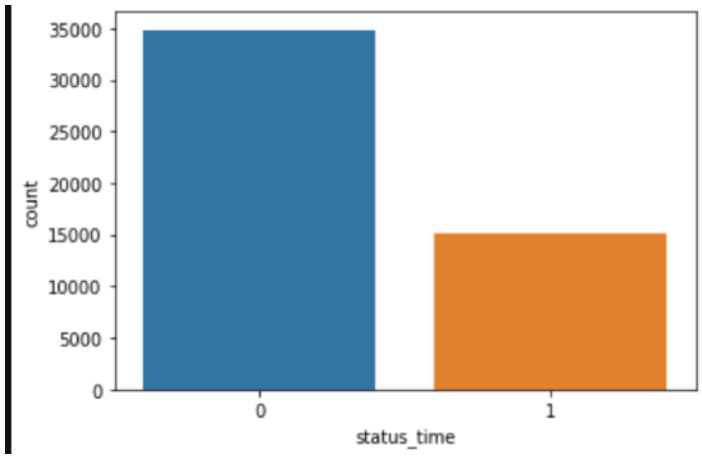
Para el inicio de la modelización de los algoritmos de clasificación, en primer lugar se tienen que aplicar una serie de funciones para cada uno de los tipos de atributos, en el caso de las variables numéricas se normaliza los resultados con la función ***MinMaxScaler*** , para el caso de los atributos categóricos se utiliza un ***label encoding*** para otorgarle un valor numérico a cada uno de los parámetros.

A continuación se separa el dataframe en dos variables X e Y, en las cuales estarán, la variable de clase en la Y, y el resto de datos en la variable X. Ambos transformados en ***array*** con la función ***value*** de pandas.

Una vez ya se tienen las variables X e Y, se obtiene el ***feature importances*** para obtener las columnas más importantes del dataframe con respecto a la variable de clase.



Tal y como se puede apreciar en la gráfica todos los atributos son importantes ya que no hay escalones muy marcados con excepción de las cuatro últimas columnas, que coincide con las columnas de tipo categórico. Ya se adelanta en este espacio del proyecto que el alumno hizo un pequeño experimento sin las columnas categóricas y las métricas que se obtuvieron no fueron satisfactorias por lo tanto al contrario que en el anterior proyecto solo se va a utilizar un dataframe sin corte en ninguna columna.



Como sucedía en el proyecto anterior existe un desbalance de clase, el cual provoca que haya un sesgo en la clasificación y por lo tanto un fallo en el mismo. Como ya se explicó anteriormente se van a utilizar la mismas metodologías (***Random Over, SMOTE)*** para el sobredimensionamiento de la clase minoritaria para evitar dicho sesgo. En cuanto a las métricas serán las mismas que las utilizadas en el proyecto anterior (***Recall, F-1 Score***) , y la validación será mediante ***k-fold cross validation*** .

```
[32]: from collections import Counter
      Counter(y)

[32]: Counter({1: 15154, 0: 34846})

[33]: from imblearn.over_sampling import RandomOverSampler,SMOTE, ADASYN
      X_resampled, y_resampled = SMOTE().fit_resample(X, y)

[34]: Counter(y_resampled)

[34]: Counter({1: 34846, 0: 34846})
```

Los algoritmos de clasificación son los siguientes:

- ***Naive Bayes***
- ***Arbol de decisión***
- ***Random Forest***
- ***KNN (en clases balanceadas)***
- ***Xgboost***
- ***Adaboost***

Por último se utilizara un ***Vooting*** con los algoritmos con las mejores métricas para ver si en conjunto pueden elevar la clasificación.

Una vez ya descrito toda la metodología a utilizar, se tiene que avisar al lector de que debido a que en la fase de experimentación existía un ***overfitting*** ,o lo que es lo mismo, métricas que daban un acierto del 100% en todos los parámetros, se procedió a eliminar las columnas payoff_time, default_time de la variable X debido a su alta correlación con la variable de clase Y

3.4.- Resultados y Conclusiones del segundo proyecto

Los resultados de la experimentación fueron los siguientes

RESULTADOS DE LOS CLASIFICADORES							
			RECALL		F-1 SCORE		F-1 (ACCURACY)
			0	1	0	1	
DATA SET SIN MODIFICACIÓN DE FEATURE IMPORTANCE	SIN SINTÉTICOS	NAIVE BAYES	0,971	0,178	0,834	0,286	0,731
		ARBOL DECISIÓN	0,795	0,599	0,807	0,578	0,735
		RANDOM FOREST	0,896	0,668	0,879	0,701	0,827
		KNN	###	###	###	###	###
		XGBOOST	0,888	0,675	0,875	0,699	0,824
		ADABOOST	0,889	0,650	0,871	0,682	0,816
	SMOTE	NAIVE BAYES	0,199	0,811	0,287	0,621	0,505
		ARBOL DECISIÓN	0,596	0,854	0,684	0,756	0,725
		RANDOM FOREST	0,670	0,932	0,771	0,824	0,801
		KNN	0,001	0,808	0,002	0,576	0,405
		XGBOOST	0,615	0,919	0,726	0,798	0,767
		ADABOOST	0,627	0,891	0,722	0,787	0,759
	RANDOM OVER SAMPLER	NAIVE BAYES	0,615	0,647	0,625	0,637	0,631
		ARBOL DECISIÓN	0,474	0,980	0,635	0,782	0,727
		RANDOM FOREST	0,584	0,989	0,732	0,822	0,786
		KNN	0,001	0,810	0,002	0,577	0,406
		XGBOOST	0,365	0,877	0,490	0,698	0,621
		ADABOOST	0,340	0,874	0,463	0,689	0,607

Tal y como se puede comprobar los algoritmos con mejores y más equilibradas métricas con los mismos que en el proyecto anterior, no obstante cabe a destacar las métricas obtenidas por el ***Xgboost*** y el ***Random Forest*** en la opción sin sintéticos, pero menos equilibrada.

Como conclusión sobre este proyecto a diferencia del anterior cabe a destacar que la abundancia de variables numéricas sobre las categóricas ha aumentado ostensiblemente la el valor de las métricas, también como se vio reflejado en la ***feature importances*** prácticamente todas las columnas tenían una valoración importante por lo que no se tuvo que hacer ningún corte sobre el dataframe. Además en el otro dataframe existía un exceso de parámetros categóricos en el que la totalidad de la columna tenía el mismo valor, por lo tanto eso ha debido ha elevar como contraposición la métrica de este que se estudia.

Los resultados en el Vooting para los tres algoritmos remarcados en la tabla son los siguientes:

```
clf0=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                            class_weight=None,
                                            criterion='gini', max_depth=None,
                                            max_features='auto',
                                            max_leaf_nodes=None,
                                            max_samples=None,
                                            min_impurity_decrease=0.0,
                                            min_impurity_split=None,
                                            min_samples_leaf=1,
                                            min_samples_split=2,
                                            min_weight_fraction_leaf=0.0,
                                            n_estimators=100, n_jobs=-1,
                                            random_state=None, verbose=0,
                                            warm_start=False)
clf1= XGBClassifier()
clf2= AdaBoostClassifier()
```

```
from sklearn.ensemble import VotingClassifier
clf=VotingClassifier(estimators=[('rf',clf0),('XGB',clf1),('ADaB',clf2)],voting='hard', weights=[1,1,1], n_jobs=-1)
```

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer, f1_score



print(cross_val_score(clf, X_resampled, y_resampled, cv=10, scoring = make_scorer(f1_score)))
```

```
[0.57047342 0.54484137 0.60040459 0.69474666 0.75336206 0.78087837
 0.82018143 0.86910178 0.73551804 0.64328853]
```

A pesar de que en las vueltas número 7 y 8 otorga las mejores métricas del proyecto, al hacer la media deja un valor muy inferior al los algoritmos por separado.

4.0.-CONCLUSIONES GENERALES

Como se ha explicado en la memoria y se ha podido ver en los resultados de ambos experimentos, la utilización de las técnicas de sobredimensionamiento mediante la elaboración de patrones sintéticos mejora de manera muy pronunciada las métricas de estudio de los algoritmos de clasificación lo que conlleva a una mejora en el entrenamiento del algoritmo y de su capacidad de predicción. Esta técnica es un elemento esencial en los casos en que los dataset proporcionados al data scientist tienen un desbalance de clase muy marcado, y una ejemplos muy indicativo son los utilizados en ambos proyectos en el que en el ámbito del mundo financiero y bancario, los ratio de morosidad (variable de clase) son muy bajos, debido a la naturaleza intrínseca del negocio bancario y su supervivencia sobre la morosidad. No obstante la técnica de sobredimensionamiento de la librería ***imbalanced-learn*** es solo una de las muchas que se puede elaborar.