12
22

PREVENCION DE FRAUDE

por DSConsultores

RESUMEN
de los objetivos

MODELOS
de Machine Learning

CONCLUSION
del trabajo realizado

RESUMEN

El presente informe tiene por objetivo mostrar de manera resumida todo el trabajo realizado para arribar a un modelo de machine learning optimo que nos permita realizar la clasificacion de transacciones realizadas con tarjeta, para predecir si las mismas seran fraudulentas o no.

Todo el trabajo se realizo en Python en un entorno de Jupyter Lab, aplicando los conocimientos adquiridos dentro del curso de Data Science de Digital House.

A continuacion se muestra todo el trabajo realizado y se adjunta el notebook con el codigo correspondiente.

INDICE

1 INTRODUCCION

El fraude cibernético e informático se refiere al **FRAUDE** realizado a través del uso de una computadora o del Internet. La piratería informática (hacking) es una forma común de fraude: el delincuente usa herramientas tecnológicas sofisticadas para acceder a distancia a una computadora con información confidencial. Otra forma de fraude involucra la intercepción de una transmisión electrónica. Esto puede ocasionar el robo de la contraseña, el número de cuenta de una tarjeta de crédito u otra información confidencial sobre la identidad de una persona.

La **PREVENCION DEL FRAUDE** es la implementación de una estrategia para detectar transacciones fraudulentas o acciones bancarias y evitar que estas acciones causen daños financieros y de reputación al cliente y a la institución financiera (FI). A medida que los canales de banca en línea y móvil se vuelven más populares y las instituciones financieras continúan digitalizándose, una estrategia sólida de prevención de fraude solo será más que importante.

La prevención del fraude y el cibercrimen están conectados y siempre cambian. A medida que los profesionales de prevención de fraude desarrollan nuevas soluciones de autenticación y detección de fraude, los estafadores se conectan entre sí, monetizan e intercambian información en la Dark Web. Los estafadores de hoy utilizan estrategias sofisticadas y malware para tener éxito en sus actividades fraudulentas. Aunque tecnología de prevención de fraude ha hecho grandes avances y continúa haciéndolo, es importante tener en cuenta las tácticas fraudulentas y comprender cómo prevenir el fraude.

A partir del trabajo realizado vamos a arribar a un modelo de Machine Learning que le permita a sus usuarios poder detectar cuando una transaccion puede llegar a ser fraudulenta o no.

ENTREGABLE I

2.1) Tema de investigación

Prevención y Detección de Fraude de operaciones realizadas con tarjeta de de credito y/o debito

2.2) Antecedentes sobre el tema

El fraude con tarjeta de crédito/debito implica el uso no autorizado de la información de la tarjeta de una persona con el propósito de cargar compras en la cuenta de la víctima o extraer fondos de su cuenta. El fraude con tarjeta de está considerado como una forma de robo de identidad. Debido a la popularidad de las compras en línea, los delincuentes ya no necesitan una tarjeta física. Con el nombre del titular, el número de la tarjeta y la fecha de vencimiento es suficiente, lo que ha provocado que el caso de fraudes se eleve en los ultimos años y sea un tema muy importante a resolver.
Decidimos trabajar sobre esta tematica ya que nos interesa mucho dicha problematica y a su vez es uno de los mas populares dentro del Machine Learning siendo uno de los algoritmos más utilizados en Argentina y el mundo.

2.3) Aporte esperado

Conocer qué variables son las más importantes para la detección de Fraude y qué metodologías implementan los Bancos o diferentes empresas para poder hacer frente a una problemática que hoy en día es permanente en las diferentes industrias, y aumenta a día junto con el crecimiento de la era digital y el crecimiento del comercio electrónico.

2.4) Disponibilidad de datos e infraestructura

https://www.kaggle.com/datasets/ban7002/fraud-challenge-data?select=fraud_challenge_150k.csv. El Dataset cuenta con 149,871 Registros y 26 Columnas (una de ellas es la Variable Target, que nos indica si el registro se trataba de un hecho lícito o de Fraude. Event_Label= 1 : Fraud ; Event_Label=0 : Not Fraud.

2.5)Plan de trabajo y cronograma tentativo

1. Limpieza de datos, tratar de imputar de alguna manera los valores faltantes. Hacer un análisis de correlación para ver a priori cuáles están mas correlacionadas con la variable target.
2. Creación de variables dummies para las variables categóricas (card bin, postal codes, user agent, etc).
3. Análisis de Outliers usando DBScan. Si identificamos outliers, los eliminamos del Dataset.
4. Ver si el dataset está desbalanceado (mostrar en un countplot que está desbalanceado). Implementar Random Under Sampling o Near Miss para balancearlo.
5. Calcular la importancia de las features, que nos permite reducir el conjunto de features en uno menor, que genera un modelo con bajo o nulo costo computacional en performance, y que por ende va a correr más rápido. Calculamos la importancia de features mediante un modelo de Random Forest o un modelo de CART.
6. Utilizar distintos modelos de Algoritmos Supervisados (tenemos la variable Target) como KNN, Regresión Logística, Naive Bayes, Árboles de Decisión (DecisionTreeClassifier), Bagging, Random Forest, XGBoost, utilizando en todos GridSearch para encontrar los mejores Hiperparámetros para encontrar los modelos que optimicen el Recall (Nos interesa maximizar la predicción de las personas que cometen fraude, es decir nos interesa focalizarnos en el universo de los casos reales positivos – los que cometen fraude- , y en particular maximizar los TP, que son los casos que predecimos correctamente como positivos, cuando realmente lo eran). Ir guardando en un diccionario las métricas (accuracy, recall, AUC, etc) de cada modelo y mostrar las curvas ROC de todos los modelos para identificar qué modelo es más robusto a cambios de thresholds. Mostrar las Matrices de Confusión de cada Modelo.
7. Implementar un modelo de ensamble de distintos modelos para tratar de mejorar la predicción de nuestro modelo resultante (mejorar el Recall).
8. Implementar un modelo de Pipeline para tratar de automatizar todo lo hecho anteriormente.

3 DATASET

https://www.kaggle.com/datasets/ban7002/fraud-challenge-data

Se realizo un analisis de todos los datasets de fuente abierta que pudimos encontrar sobre el tema que decidimos trabajar y elegimos el denominado **"Fraud Challenge Data"**, que se puede observar en el enlace superior. Cabe destacar que no tuvimos la posibilidad de acceder a datos de una empresa real por temas de confidencialidad de la informacion.
El mismo contiene un conjunto de datos sobre operaciones realizadas con tarjetas y cuenta con las siguientes columnas:

- **account_age_days:** días que lleva abierta la cuenta asociada a la transacción
- **transaction_amt / transaction_adj_amt:** monto de las transacciones
- **historic_velocity:** velocidad de compra teniendo en cuenta la ultima transaccion
- **applicant_name / ip_address / user_agent / email_domain / phone_number/ billing_city / billing_postal / billing_state / billing_address:** informacion del usuario
- **card_bin** (Primeros 5 números de tarjeta) / **cvv** (codigo de seguridad) / **signature_image** (firma digital) **:** informacion de tarjeta- metodo de pago
- **currency:** moneda **/ transaction_type:** cuotas, debito, credito
- **transaction_env:** cajero, homebanking, app
- **EVENT_TIMESTAMP:** dia y hora de la transaccion
- **merchant_id:** id del comercio / **locale:** idioma del pais
- **days_since_last_logon:** dias desde el ultimo inicio de sesion
- **inital_amount** monto inicial que tiene en la cuenta el usuario al momento de iniciar la transacción
- **EVENT_LABEL -Esta es nuestra variable TARGET donde 1 = es fraude 0 = no es fraude**

El OBJETIVO es prevenir una operacion fraudulenta realizada con tarjeta

El dataset cuenta con 150.000 registros y 26 columnas, debido al gran volumen de los datos que contiene el mismo, planteamos distintas opciones para poder trabajarlo de manera optima.

1) En una primera instancia trabajamos con el dataset original, el problema surgio al momento de crear las variables dummies de las dimensiones categoricas las cuales se transformaban en una cantidad de columnas imposibles de procesar, por lo cual intentamos otras maneras de trabajar los datos hasta que arribamos a la optima

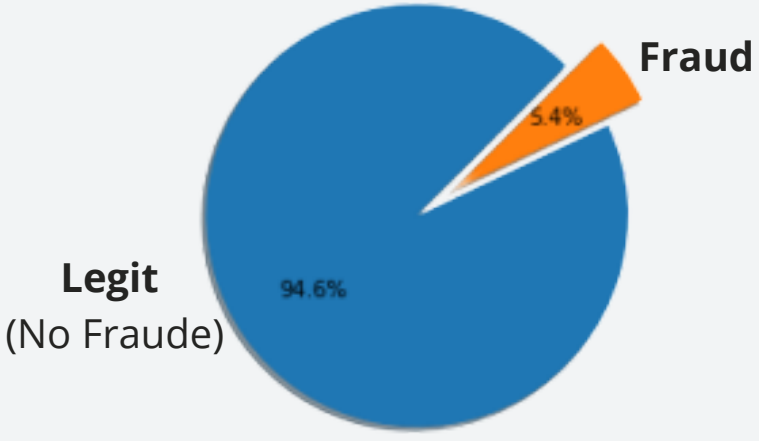
2) Trabajamos con un porcentaje del dataset original

3) Trabajamos con una porción del dataset, manteniendo la variable target minoritaria original pero disminuyendo la mayoritaria con sample(muestra) de un n% y fuimos probando diferentes porcentajes

Al seguir profundizando en el contenido del dataset planteamos los siguientes criterios para reducir las variables:

- **ratio_para_eliminar:** es el % de variables unicas por columna sobre la cantidad de filas, si las variables unicas de la columna / data.shape[0] > ratio, entonces la columna se elimina automáticamente.
- **X_rep:** es el numero de repeticiones minima de cada variable unica por columna que consideramos limite para poder eliminar.
- **corte:** dentro de las features importance, hacemos una suma acumulada de la importancia de cada feature, y nos quedamos con las features que acumulan el valor de corte

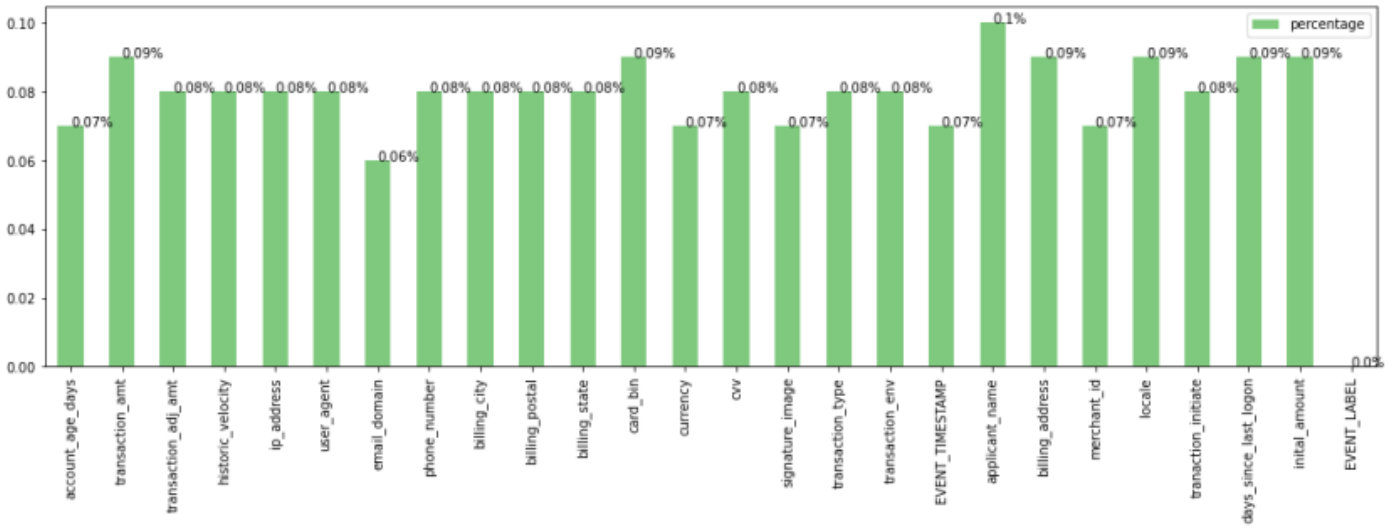
Analizamos como esta distribuida nuestra variable target y observamos un claro desbalanceo de los datos:



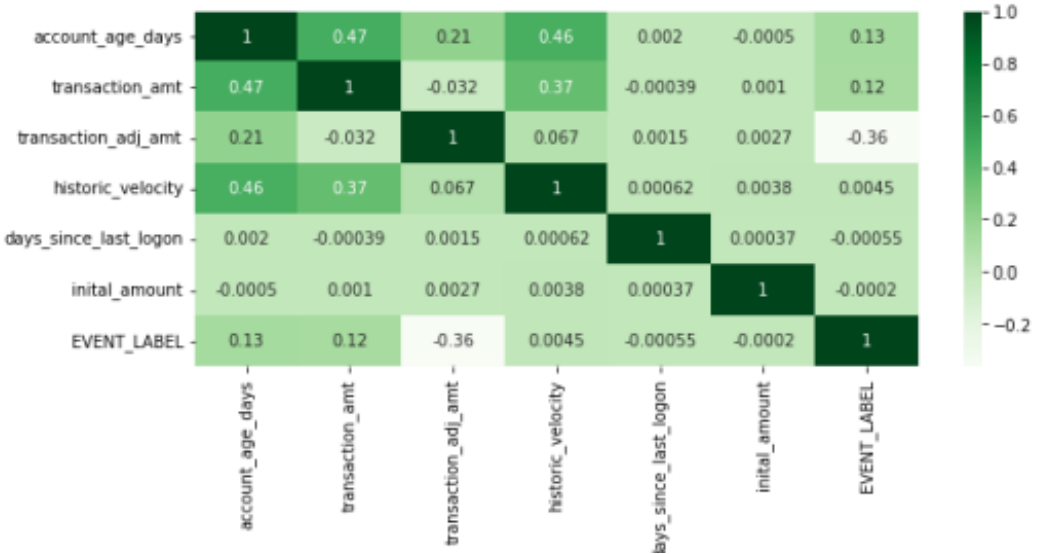
4 FEATURE ENGINEERING - LIMPIEZA DE DATOS

En esta parte del analisis lo que realizamos es todo el proceso para obtener un dataset prolijo, cambiando los tipos de datos de las columnas, analizando duplicados, nulos, la cantidad de repeticiones unicas que hay en cada columna, datos que consideramos no aportan al modelo y los descartamos.

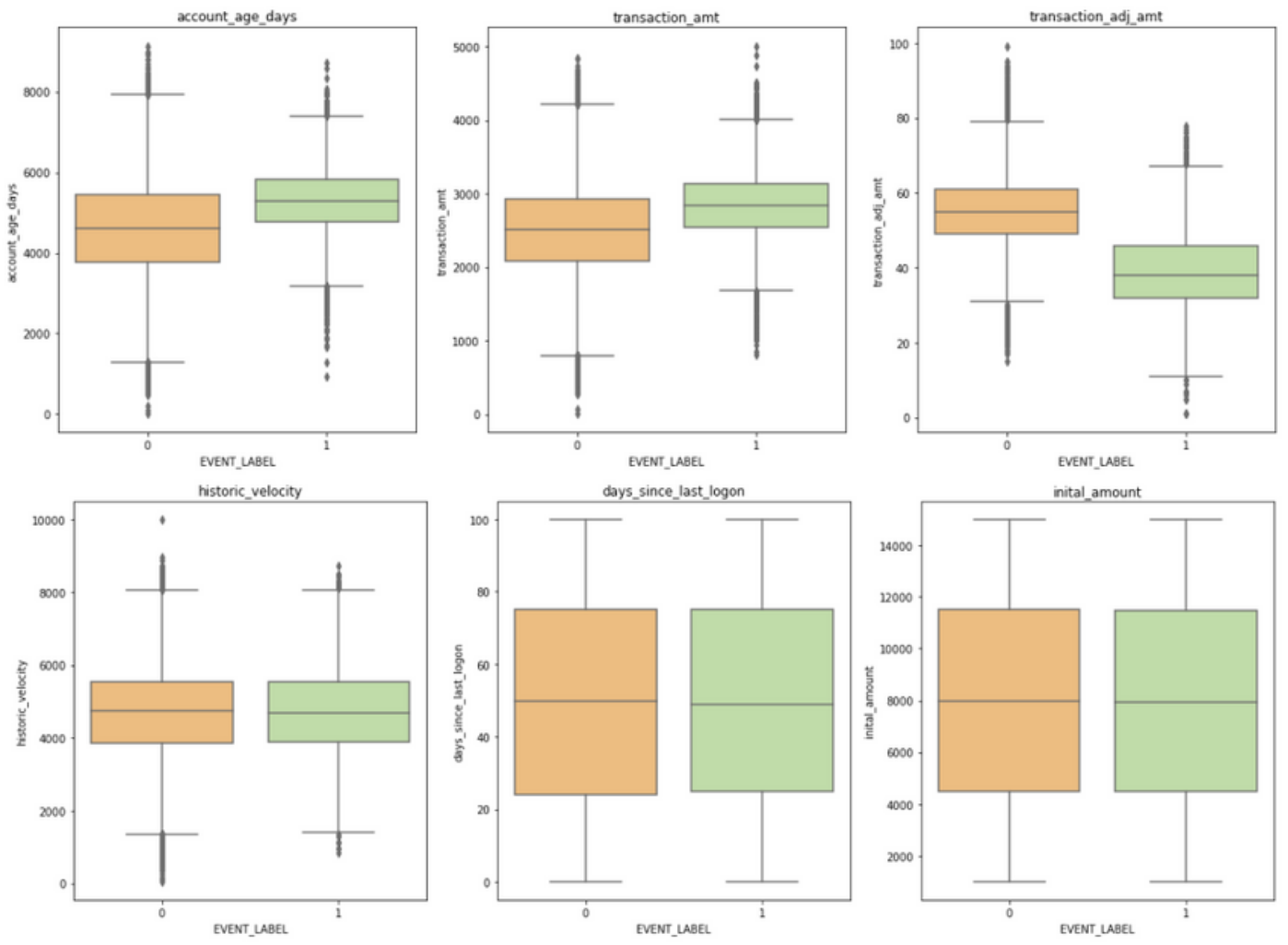
- Transformamos la variable target en binaria
- Chequeamos valores nulos del dataset y eliminamos las filas donde tengamos al menos un valor nulo. En este paso se eliminaron 3.000 registros de 150.000 (2% de la muestra original)



- Chequeamos si habia algun registro duplicado.
- Eliminamos aquellas columnas que tuvieran poca variabilidad en los registros (explicado en la filmina anterior)
- Pasamos las columnas 'billing_postal' y 'card_bin' a tipo 'object' ya que no pueden ser consideradas como un número
- Realizamos un Análisis gráfico de las variables para ver su correlacion:



- Creamos un boxplot de las variables numéricas:



Con estos graficos podemos observar rapidamente una presentacion estadistica para conocer como estan distribuidos los datos de las variables numericas en relacion a la variable target y detectamos que :

- En todos los casos los datos presentan una distribucion simetrica, ya que la mediana se encuentra en el medio de las cajas
- No hay grandes cantidades de outliers
- Para las variables 'account_age_days' y 'transaction_amount' la mediana se encuentra por encima en el caso de fraude con respecto a los casos de no fraude
- Por otro lado en el caso de la variable 'transaction_adj_amt' la mediana de los casos detectados como fraude se encuentran por debajo de los no fraude
- Y para las variables 'historic_velocity', 'days_since_last_logon' y 'initial_amount' las medianas coinciden en ambas situaciones, es decir, tanto para los casos de fraude como no fraude.

- Analizamos la distribucion de las variables categoricas, considerando la variable target: (se muestran las primeras 7, ver resto en la notebook)

Pudimos a traves de los graficos detectar rapidamente lo desbalanceado que se encontraba el dataset y determinar algunos insights que luego corroboraremos con los modelos tales como por ejemplo:

- Habian 3 IP_address que tenian la misma cantidad de fraudes cometidos
- La 'billing_city' con mayor cantidad de fraudes cometidos era East Laurieland
- El 'billing_state' con mayor cantidad de fraudes cometidos era Michigan
- La principal moneda a la hora de cometer los fraudes era 'cad' seguida por usd.

- Buscamos las columnas categóricas para saber cuantas variables únicas tiene cada una, si son muchas variables unicas que se repiten pocas veces (casi la misma cantidad de variables del dataset) vamos a eliminar esa columna ya que no aporta valores significantes al modelo

	col	%		col	%
15	billing_address	99.996	7	card_bin	4.407
16	merchant_id	99.994	17	locale	0.199
13	EVENT_TIMESTAMP	99.781	6	billing_state	0.034
14	applicant_name	65.300	9	cvv	0.018
0	ip_address	9.274	12	transaction_env	0.018
3	phone_number	8.292	11	transaction_type	0.018
5	billing_postal	7.686	10	signature_image	0.018
4	billing_city	6.237	18	tranaction_initiate	0.018
1	user_agent	5.984	8	currency	0.002
2	email_domain	4.846			

- Luego realizamos un **ANALISIS DE CADA DIMENSION** en esta etapa, antes de eliminar las columnas que tienen todas las variables como únicas, trabajamos sobre columnas categóricas aplicando distintas regex para lograr separar en partes los datos contenidos en las columnas y así volver a chequear el aporte de las nuevas columnas creadas.
    - Separamos la columna **'EVENT_TIMESTAMP'** en año, mes, día, hora y minuto
    - Trabajamos sobre la columna **'card_bin'** para asociar el bin de la tarjeta a la marca de la tarjeta

The first digit is different for each card network:

- **Visa cards begin with a 4 and have 13 or 16 digits**
- **Mastercard cards begin with a 5 and has 16 digits**
- **American Express cards begin with a 3, followed by a 4 or a 7 has 15 digits**
- **Discover cards begin with a 6 and have 16 digits**
- **Diners Club and Carte Blanche cards begin with a 3, followed by a 0, 6, or 8 and have 14 digits**

link api bin card: https://binlist.net/

** **

Digits 1 – 6: Issuer identifier numbers First digit: Represents the network that produced the credit card. It is called the Major Industry Identifier (MII). Each digit represents a different industry.

0: ISO/TC 68 and other industry assignments
1: Airlines
2: Airlines, financial and other future industry assignments
3: Travel and entertainment
4: Banking and financial
5: Banking and financial
6: Merchandising and banking/financial
7: Petroleum and other future industry assignments
8: Healthcare, telecommunications and other future industry assignments
9: For assignment by national standards bodies

	EVENT_LABEL		total	%_1
EVENT_LABEL	0	1		
card_bin_brand				
airlines	196	13	209	6.22
airlines & financial	12686	696	13382	5.20
american express	15941	881	16822	5.24
diners	12248	670	12918	5.19
discovers	8289	438	8727	5.02
mastercard	9897	595	10492	5.67
ravel & entertainment	14388	789	15177	5.20
visa	65351	3922	69273	5.66

- Trabajamos sobre la columna **'merchant_id'** separando el numero por '-' obteniendo 'merchant_id_first', 'merchant_id_middle', 'merchant_id_end'
- Trabajamos sobre la columna **'locale'** separando el numero por '-' obteniendo 'locale_language' y 'locale_country'
- Trabajamos sobre la columna **'billing_address'** separando el numero por '-' obteniendo 'billing_address_number_street', 'billing_address_name_street'
- Trabajamos sobre la columna **'email_domain'** separando el numero por '.' obteniendo 'email_server', 'email_dot_domain'
- Trabajamos sobre la columna **'user_agent'** separando el numero por '.' obteniendo 'user_agent_browser' y 'user_agent_SO_match'
- Trabajamos sobre la columna **'ip_address'** separando el numero por '.' obteniendo 'ip_address_A', 'ip_address_B', 'ip_address_C', 'ip_address_D', 'ip_address_E' dependiendo la clase de dirección IP,

Address Class	RANGE	Default Subnet Mask
A	1.0.0.0 to 126.255.255.255	255.0.0.0
B	128.0.0.0 to 191.255.255.255	255.255.0.0
C	192.0.0.0 to 223.255.255.255	255.255.255.0
D	224.0.0.0 to 239.255.255.255	Reserved for Multicasting
E	240.0.0.0 to 254.255.255.255	Experimental

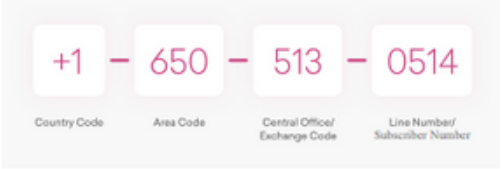
Note: Class A addresses 127.0.0.0 to 127.255.255.255 cannot be used and is reserved for loopback testing.

	EVENT_LABEL		total	%_1
EVENT_LABEL	0	1		
ip_address_class				
A	79191	4533	83724	5.41
B	40806	2369	43175	5.49
C	18999	1102	20101	5.48

A: Grandes Empresas **B:** Empresas **C:** Hogares

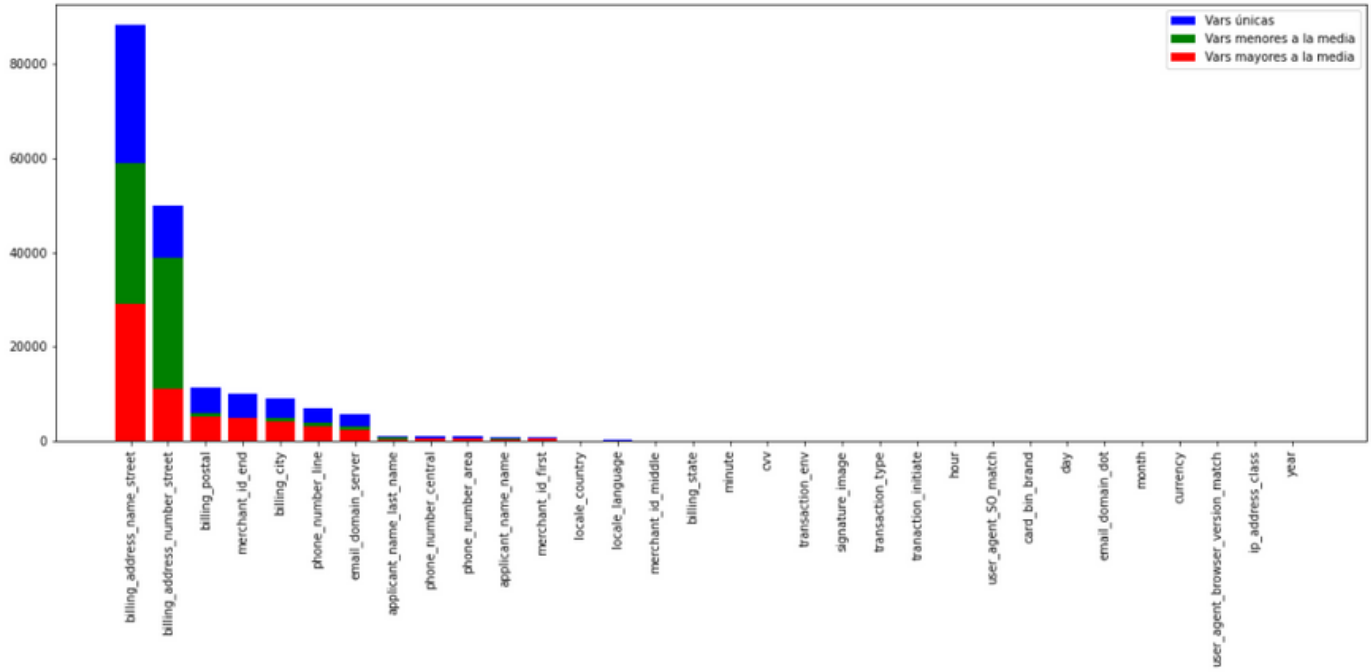
- Trabajamos sobre la columna **'phone_number'** separando el numero para obtener 'phone_number_country_code', 'phone_number_area_code', 'phone_number_central_office', 'phone_number_line', 'phone_number_ext', de las cuales solo nos quedaremos con 'phone_number_area_code', 'phone_number_central_office' que consideramos las variables más significativas para nuestro análisis.

=ParsePhoneNumbers(A2,3,"X")		SplitWhat argument set to 3		
Phone Number	Area Code	Prefix	Number	Extension
9135551212	913	555	1212	
913-5551212	913	555	1212	
913-555-1212	913	555	1212	
(913)5551212	913	555	1212	
(913)555-1212	913	555	1212	
9135551212X1234	913	555	1212	X1234
913-555-1212 X1234	913	555	1212	X1234
(913)555-1212X1234	913	555	1212	X1234
bad text here	#VALUE!	#VALUE!	#VALUE!	#VALUE!



- Trabajamos sobre la columna **'applicant_name'** separando el nombre del apellido

Una vez realizadas todas las transformaciones pasamos de tener 26 columnas a 39 (previo a la creacion de dummies), analizamos visualmente los cambios y volvimos a controlar el efecto de las variables unicas y sus repeticiones para ver que columnas vamos a eliminar
Realizamos pruebas para arribar a la solucion mas optima, que nos permita quedarnos con datos que consideramos van a aportar al modelo y eliminando aquellos que pueden convertirse en ruido y perjudicarlo al aumentar el tamaño del dataset, analizamos las features que tienen muchas variables únicas unitarias, es decir sin repeticiones, las cuales no colaboran en el modelo de predicción. Buscamos eliminar un poco de esas variables ya que luego van a ser transformadas a dummies, y éstas van a aumentar considerablemente su dimensión. En el dataset tenemos -x- variables unicas de las cuales hay -n- que se repiten solo una vez y (si comparo con las medias de cada columna) hay (x-n) que son menores a la media.



Con este gráfico podemos observar que hay muchas variables únicas en el dataset que no colaboran en la clasificación por lo cual eliminamos todas las variables únicas que se repiten menos de una milecima parte de la cantidad de filas dentro del dataset reduciendolo de 186.648 variables únicas a 26.835

5 REDUCION DE DIMENSIONALIDAD

Lo que vamos a hacer en esta estapa es buscar disminuir aun mas el tamaño del dataset, por lo que recurrimos al analisis de importancia de features de XGBoost y aplicamos tambien PCA.

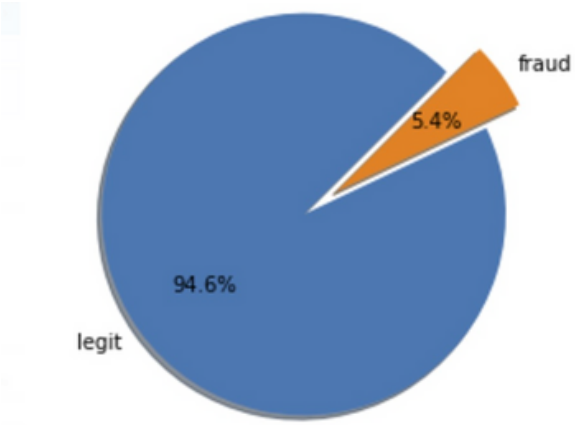
Aplicando XGBoost para arribar a las dimensiones de mayor importancia primero establecimos un número de acumulacion de importancias bajo el cual tomar las columnas que lo contienen y trabajar en adelante con ellas corte = 0.8, obtuvimos de esta manera 14 columnas, sobre las cuales luego generamos las variables dummies correspondientes.

Luego si en el dataset de dummies teniamos menos de 1000 columnas trabajamos con ese, de lo contrario utilizabamos el dataset creado con PCA

	nombre_col	Q_vars	%_x_row	max_rep	media_rep	mediana_rep	min_rep
8	billing_postal	4647	3.16	40135	31.63	22.0	16
9	phone_number_line	4083	2.78	21660	36.00	27.0	16
3	email_domain_server	3306	2.25	13177	44.46	37.0	16
5	phone_number_area	1000	0.68	323	147.00	144.0	7
10	phone_number_central	1000	0.68	375	147.00	143.5	20
4	merchant_id_first	898	0.61	301	163.70	163.0	127
11	locale_language	182	0.12	893	807.69	805.5	739
7	billing_state	44	0.03	8656	3340.91	2257.0	17
0	transaction_type	24	0.02	15865	6125.00	3822.0	16
2	transaction_env	25	0.02	22173	5880.00	1571.0	11
1	cvv	22	0.01	22787	6681.82	2222.5	22
6	currency	3	0.00	111543	49000.00	31151.0	4306

6 MODELOS SIN BALANCEO

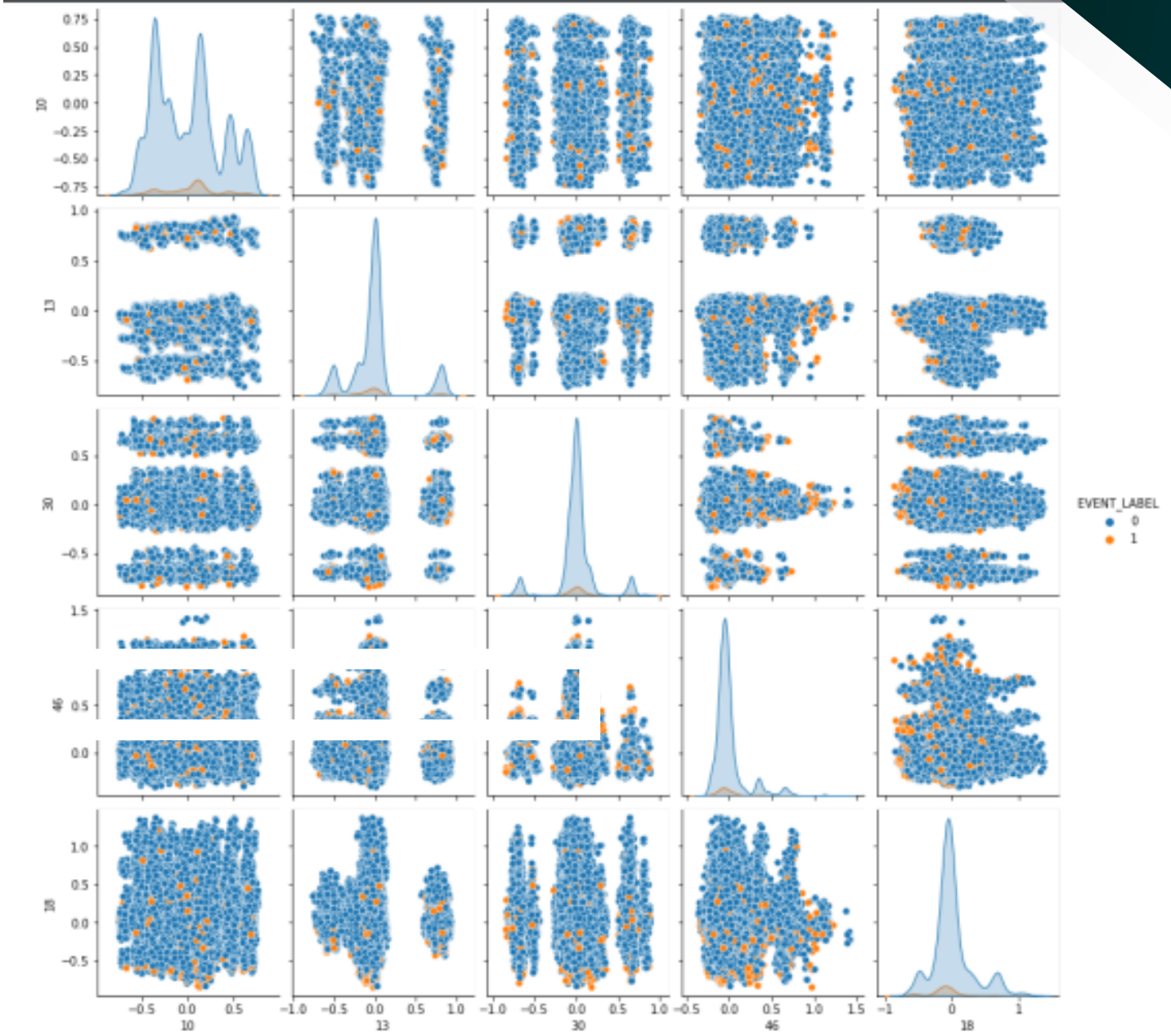
Tal como observamos anteriormente, nuestro dataset estaba desbalanceado, con menos del 6% de los registros catalogados como fraude.



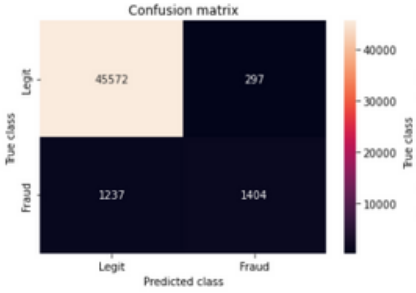
Previo a testear distintos metodos de balanceo (que describiremos en la etapa siguiente) corrimos distintos modelos de clasificacion con el dataset sin balancear, pero ya habiendo reducido la cantidad de variables (columas) con las herramientas que mencionamos anteriormente.
Aplicamos los siguientes modelos de clasificacion tanto para el dataset sin balancear como con cada uno de los metodos de balanceo:

- Logistic Regression
- Random Forest Classifier
- Gaussian Naive Bayes
- KNN
- CART - Decision Tree Classifier
- XGBoost
- Ensamble, compuesto por
    - Un modelo de clasificación Naive Bayes (Gaussian)
    - Un modelo de clasificación KNN
    - Un modelo de regresión logística con regularización
    - Un árbol de clasificación

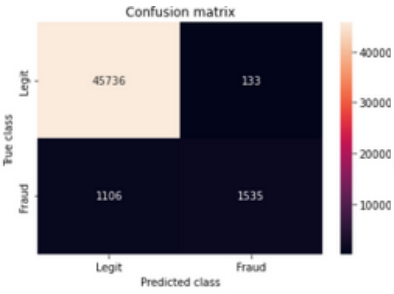
En la imagen siguiente podemos observar la distribucion de las variables sin balancear y las matrices de confusion a las que arribamos en cada modelo:



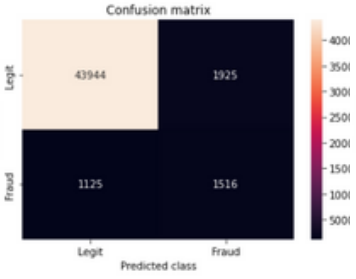
LOGISTIC REGRESSION



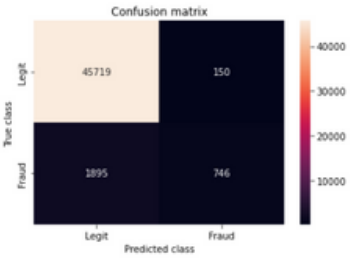
RANDOM FOREST CLASSIFIER



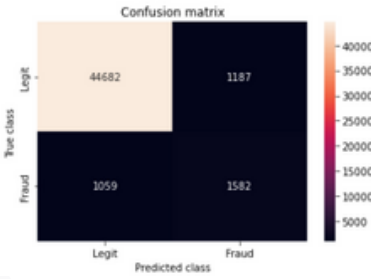
GAUSSIAN NAIVE BAYES



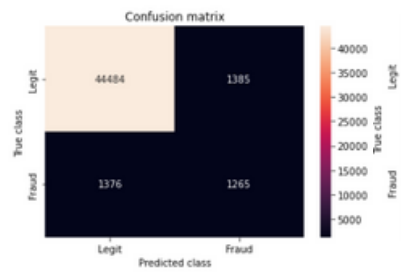
KNN



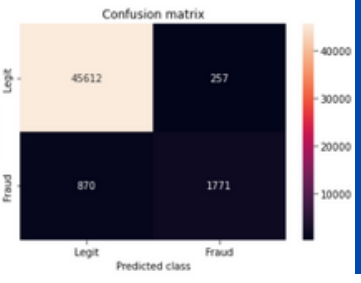
CART -DECISION TREE CLASSIFIER



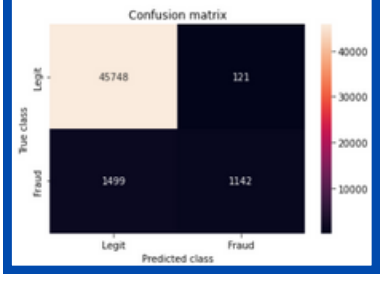
Extra Tree Classifier



XGBOOSTING



ENSAMBLE



por DSConsultores

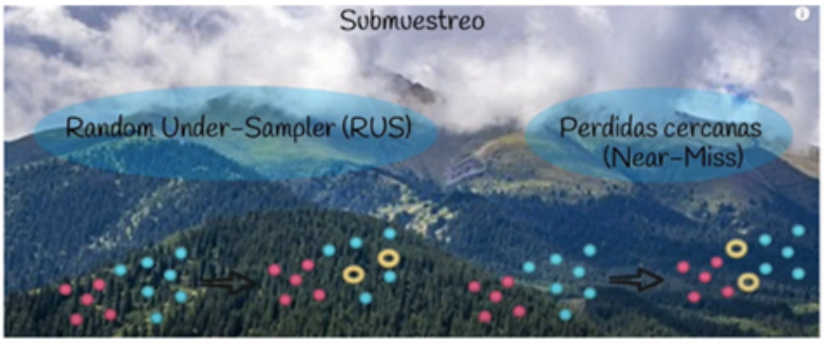
MÉTODOS DE BALANCEO

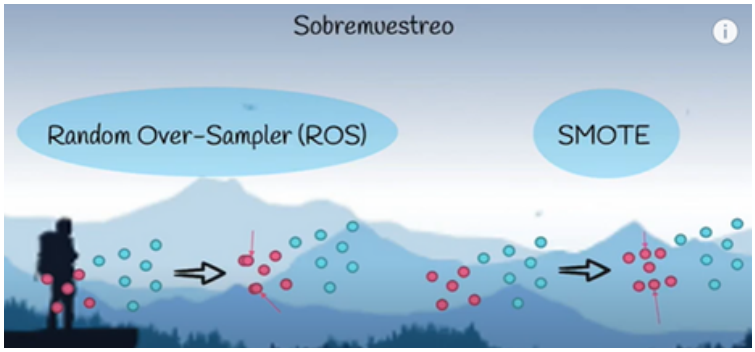
1. **Near Miss:** Elimina valores de la frontera de la clase mayoritaria.
2. **Random Under Sampler:** Elimina valores random de la clase mayoritaria.
3. **SMOTE Tomek:**

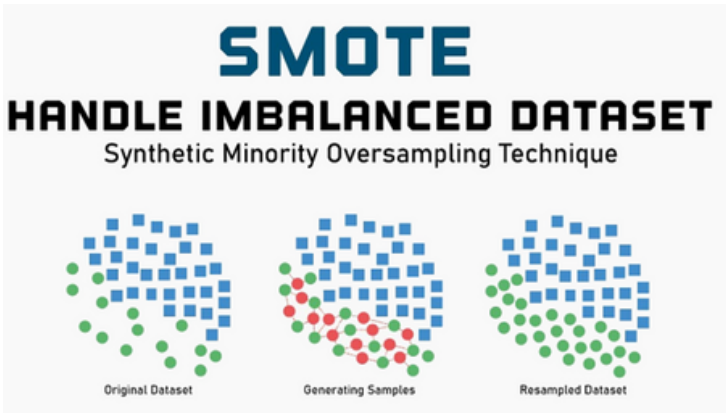
- **SMOTE:** Crea variables artificiales para la clase minoritaria. Toma dos puntitos arbitrarios y calcula la media ponderada de esas dos características y se sitúa entre esos dos puntos.
- **Tomek Link:** Busca los pares que hay en la frontera, los encapsula, y en nuestro caso nos quedamos con los Event_label=1 (clase minoritaria) y eliminamos los Event_label= 0 (clase mayoritaria).

4. **ROS/RUS:**

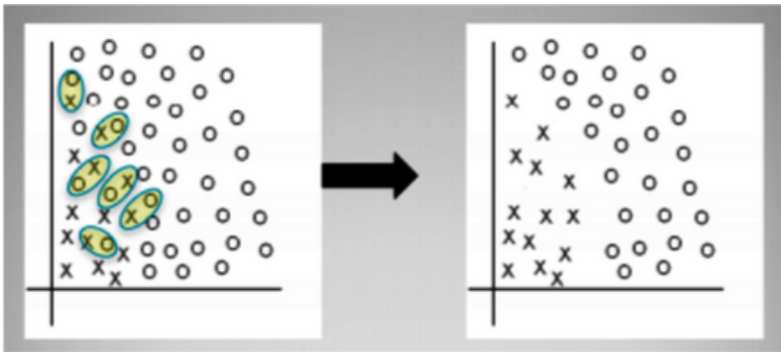
- **ROS:** Primero hace un ROS, es decir, añade valores random de la clase minoritaria.
- **RUS:** Luego hace un RUS sobre el resultado del ROS, es decir elimina puntos random de la clase mayoritaria.







Tomek Link:



8 RESULTADOS DE MODELOS DE CLASIFICACIÓN

TOP 10 MEJORES BALANCEO-MODELOS ORDENADOS POR RECALL DESCENDENTE:

	index	Accuracy	F1	Recall	Precision	AUC	AP
14	NearMiss_XGBC	0.412533	0.139085	0.871640	0.075572	0.690614	0.099736
9	NearMiss_RFC	0.508720	0.161199	0.867096	0.088860	0.826613	0.520112
22	RUS_XGBC	0.951577	0.652155	0.833775	0.535506	0.940311	0.788476
32	ROS_RUS_LR	0.922820	0.538689	0.827717	0.399269	0.934165	0.713247
17	RUS_RFC	0.956215	0.671716	0.822794	0.567511	0.940721	0.772304
8	NearMiss_LR	0.743414	0.255963	0.810678	0.151973	0.842717	0.230607
16	RUS_LR	0.932261	0.565344	0.809163	0.434438	0.934077	0.715784
38	ROS_RUS_XGBC	0.963265	0.704674	0.804998	0.626584	0.941365	0.803520
11	NearMiss_KNN	0.616553	0.184417	0.796289	0.104284	0.767381	0.150374
34	ROS_RUS_GNB	0.876355	0.411730	0.794775	0.277829	0.903434	0.504633

PROMEDIO DE LOS VALORES OBTENIDOS CON TODOS LOS MODELOS SEGÚN EL MÉTODO DE BALANCEO, ORDENADOS POR RECALL DESCENDENTE:

mod_balanceo	AP	AUC	Accuracy	F1	Precision	Recall
NearMiss	0.181538	0.728079	0.565821	0.174048	0.098429	0.796999
RUS	0.528689	0.888830	0.908168	0.506289	0.389886	0.771630
ROS_RUS	0.542148	0.876004	0.926793	0.546383	0.468175	0.713035
STK	0.566240	0.877191	0.951799	0.599647	0.585886	0.635081
ModeloNoBalanceado	0.577163	0.870774	0.959745	0.585774	0.730622	0.518790

PROMEDIO DE LOS VALORES OBTENIDOS CON TODOS LOS MODELOS , ORDENADOS POR RECALL DESCENDENTE:

mod_classif	AP	AUC	Accuracy	F1	Precision	Recall
XGBC	0.661235	0.891205	0.855762	0.601579	0.582478	0.778342
LR	0.620902	0.915544	0.903810	0.528293	0.470865	0.748504
RFC	0.734390	0.919134	0.877378	0.606742	0.632950	0.734722
GNB	0.423456	0.850683	0.826044	0.400051	0.307874	0.706096
DTC	0.260695	0.767735	0.838858	0.442747	0.373175	0.687921
ENS	0.621829	0.902314	0.908143	0.554233	0.595051	0.653313
ETC	0.194562	0.730906	0.841620	0.372800	0.307236	0.606664
KNN	0.316176	0.807884	0.848110	0.352980	0.367169	0.581295

9 PIPELINE

Por ultimo en esta etapa aplicamos un proceso a traves de la aplicacion de PIPELINE, comprendido en varias fases secuenciales, siendo cada salida la entrada del anterior, sin perder datos y conocimiento.

Mediante funciones IF fuimos identificando y al mismo tiempo entrenando los 2 mejores métodos de balanceo y los 2 mejores clasificadores, para que luego el pipeline decida la mejor combinación entre ambas y filtrando las features por su varianza a partir de un threshold :

```
balance_n_1 = df_results_balance_pivot.index[0]
balance_n_2 = df_results_balance_pivot.index[1]

clasificador_n_1 = df_results_clasif_pivot.index[0]
clasificador_n_2 = df_results_clasif_pivot.index[1]
```

```
parameters = {'balance': [model_balance_1, model_balance_2],
              'selector__threshold': [1, 0.5, 0, 0.00001, 0.0001, 0.001],
              'classifier':[modelo_clasif_1, modelo_clasif_2]}
```

Llegamos al mejor modelo aplicando Gridsearch:

```
grid.best_estimator_

Pipeline(steps=[('balance', NearMiss(n_neighbors=5)),
                ('selector', VarianceThreshold(threshold=0.5)),
                ('classifier',
                 GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=1234, shuffle=True),
                              estimator=XGBClassifier(base_score=None,
                                                      booster=None,
                                                      callbacks=None,
                                                      colsample_bylevel=None,
                                                      colsample_bynode=None,
                                                      colsample_bytree=None,
                                                      early_stopping_rounds=None,
                                                      e...
                                                      max_delta_step=None,
                                                      max_depth=None,
                                                      max_leaves=None,
                                                      min_child_weight=None,
                                                      missing=nan,
                                                      monotone_constraints=None,
                                                      n_estimators=100,
                                                      n_jobs=None,
                                                      num_parallel_tree=None,
                                                      predictor=None,
                                                      random_state=123,
                                                      reg_alpha=None,
                                                      reg_lambda=None, ...),
                              n_jobs=-1,
                              param_grid={'learning_rate': [0.01, 0.1],
                                          'max_depth': [1, 3, 5],
                                          'n_estimators': [10, 50, 150]},
                              scoring='recall', verbose=1))])
```

Metodo de Balanceo

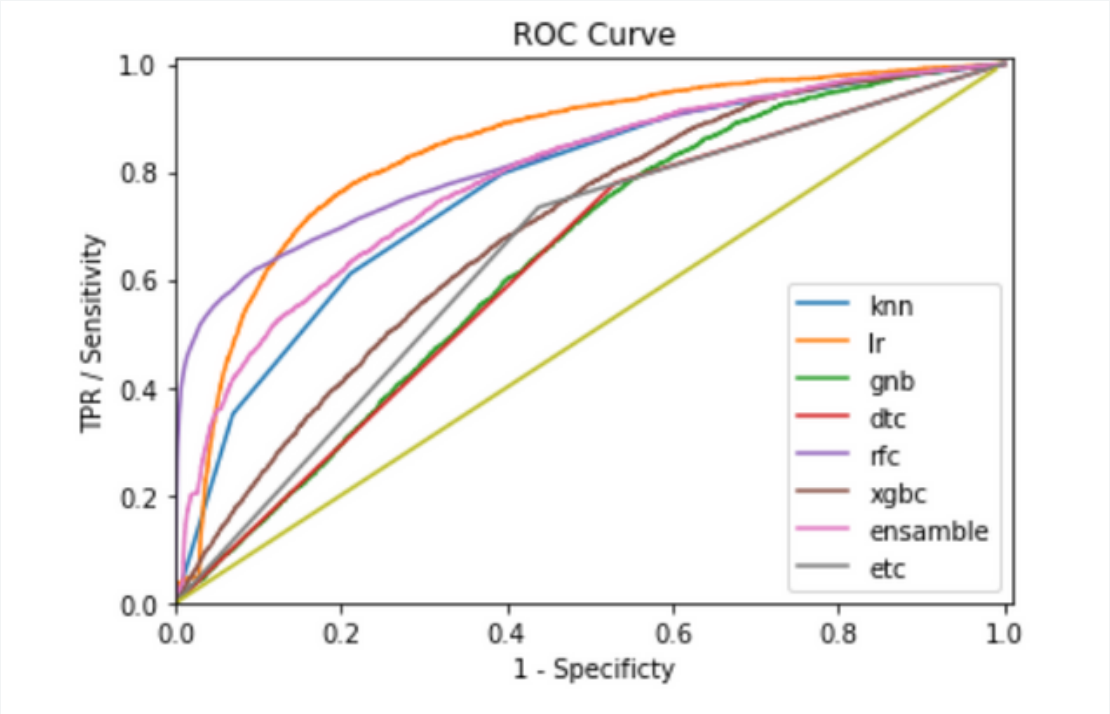
Modelo

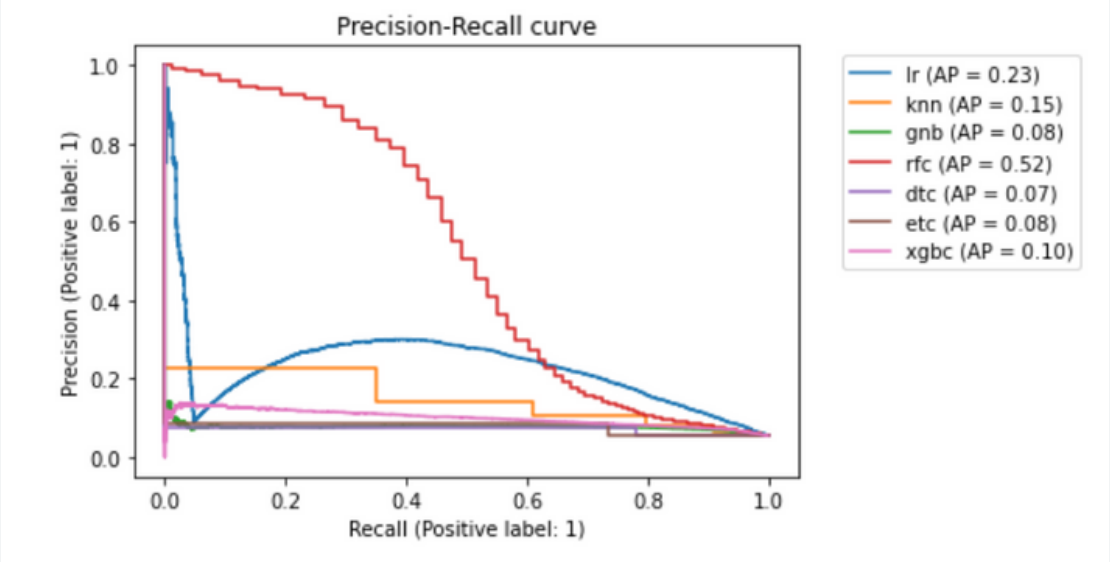
Hiperparámetros

10

CONCLUSIONES

En una primera instancia comparamos las principales metricas y las curvas ROC y Precision- Recall obtenidas a partir de cada uno de los distintos modelos aplicados para el Método de Balanceo Near Miss, que era el que en general mejor balanceaba:





Vemos que los Modelos de Random Forest y Regresión Logística parecen ser los modelos más robustos y que mejor performan a nivel general ante cambios de thresholds. Además, son los modelos que maximizan la combinación Precision- Recall.

RECORDAR: Este análisis es previo a la utilización de GridSearch en los Modelos.

GRÁFICO DE POLÍGONO DE TOP 3 MEJORES MÉTODOS DE BALANCEO, ORDENADOS POR MAYOR RECALL:

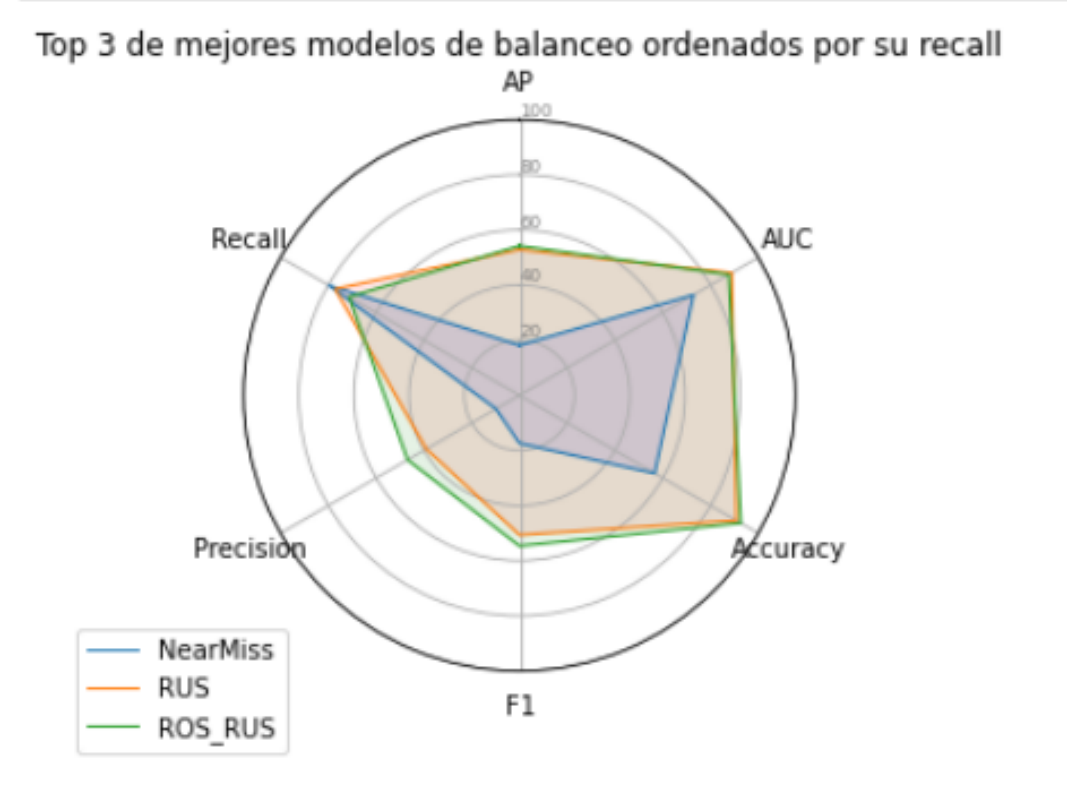
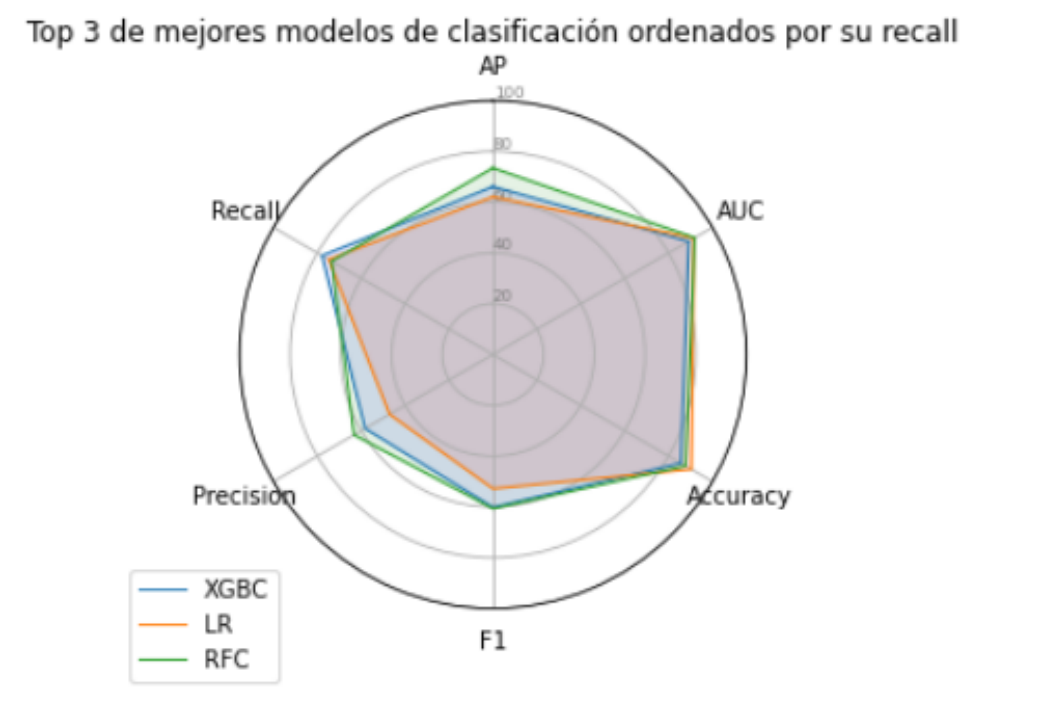


GRÁFICO DE POLÍGONO DE TOP 3 MEJORES MODELOS, ORDENADOS POR MAYOR RECALL:

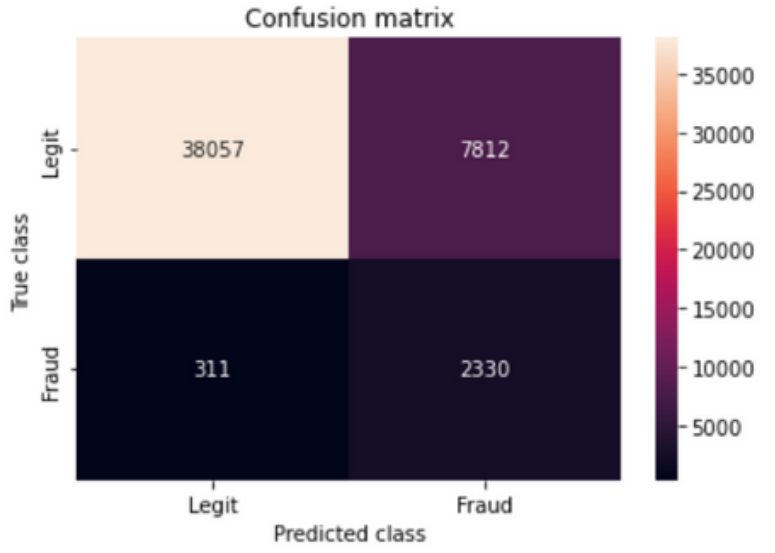


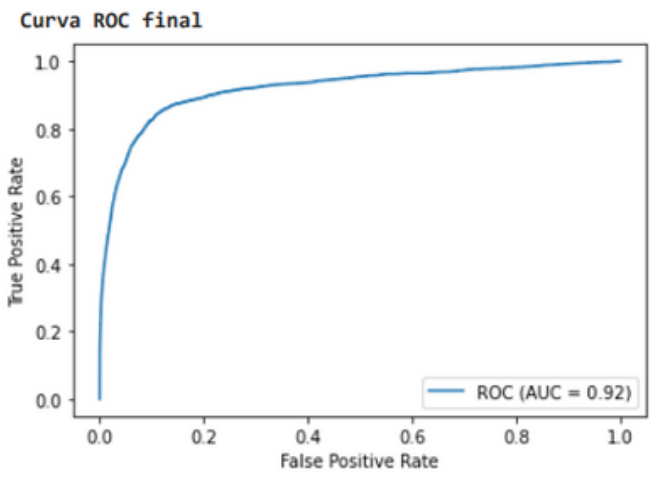
IMPORTANTE: Graficamos solo los 3 mejores para que se pueda entender bien el gráfico

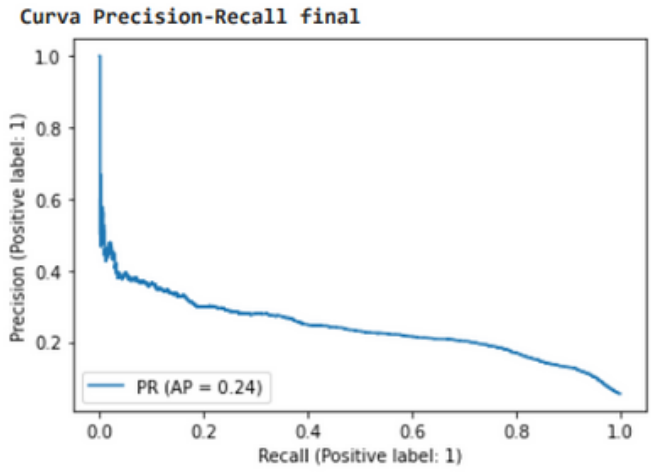
Sin embargo, frente a nuestra problemática de buscar prevenir el Fraude, decidimos adoptar como mejor modelo al que obtuvimos del análisis de Pipeline, luego de que el Pipeline nos identificara el mejor modelo con su combinación de Hiperparámetros para maximizar el Recall, y **aplicando al mismo el uso de Gridsearch** para identificar la mejor combinación de Hiperparámetros:

MEJOR BALANCEO- MODELO PARA PREDECIR: NEAR MISS - XGBOOSTING

```
Accuracy_grid: 0.8325499896928469

F1_grid: 0.36454666353751075

Recall_grid: 0.8822415751609239

Precision_grid: 0.22973772431473083

AUC_grid: 0.9204471999920026

AP_grid: 0.6142893207863207
```







En Resumen, concluimos que el modelo de Near Miss- XGBoosting es el que mejor nos sirve para nuestro objetivo de predecir el Fraude, ya que es el modelo que **maximiza el Recall (Sensitivity)**, es decir **maximiza los TP** (predice más eficientemente a las transacciones Fraudulentas) y **minimiza a los FN** (minimiza el error de predecir mal cuando una transacción en realidad es fraudulenta, pero nosotros predecimos que no lo era).

Además, vemos que el Modelo de XGBoosting tiene un **AUC de 0.92**, lo cual nos da una idea de que el modelo performa bien en general, y nos indica que el modelo va a ser robusto frente a cambios de thresholds.

Los modelos de XGBoosting y Random Forest con método de Balanceo RUS son buenas opciones también, ya que si bien no tienen un Recall tan alto como los de Near Miss, tienen valores más alto en el resto de las métricas como Precision, Accuracy, etc.