UNIVERSIDAD DE ANTIOQUIA

MODELOS DE MACHINE LEARNING PARA LA DETECCIÓN DE FRAUDE FINANCIERO

Autor(es)

Maricela Carmona Mora

Leidy Marcela Londoño Morales

Universidad de Antioquia

Facultad de Ingeniería

Medellín, Colombia

2021

Modelos de machine learning para la detección de fraude financiero

Maricela Carmona Mora
Leidy Marcela Londoño Morales

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:
Especialista en Analítica y Ciencia de Datos

Asesores (a):
Efraín Alberto Oviedo
Magíster en TICs

Universidad de Antioquia
Facultad de Ingeniería
Medellín, Colombia
2021

TABLA DE CONTENIDOS

1. RESUMEN EJECUTIVO

En la actualidad y con la constante evolución de la tecnología, cada vez nos volcamos a un mundo mucho más digital, es por esto, que nos centramos en revisar cómo desde una vista financiera y con el uso de una billetera móvil podemos identificar de todas las transacciones que se realizan cuales corresponden a fraudes.

Para el desarrollo de esta monografía se partió de la simulación de transacciones de dinero móvil basadas en una muestra real extraída de un mes de registros financieros implementado en un país africano. Los registros originales fueron proporcionados por una empresa multinacional, proveedor del servicio financiero móvil. Dichos datos se toman de Kaggle, los cuales corresponden a una cuarta parte del conjunto de datos original.

El mayor reto presentado fue contar con un problema asociado al desbalanceo de los datos en términos de los eventos de fraude, ya que dada la ocurrencia del evento es más confuso poder identificar con mayor precisión el fraude y contar adicionalmente con las mejores métricas para tomar decisiones acertadas y poder escoger el modelo a implementar.

Dentro de la solución se hace uso de la generación de datos sintéticos para balancear los eventos de fraude y poder comparar los resultados de los modelos teniendo o no este tipo de balanceo.

No se evidencian diferencias significativas comparando los resultados de los modelos con y sin sobremuestreo, por lo tanto, para este caso de detección de fraudes resulta más eficiente implementar los modelos sin datos sintéticos teniendo en cuenta el costo computacional que implica generarlos.

Los mejores resultados se obtienen mediante el uso de la *Regresión Logística*, sin tener en cuenta el sobremuestreo.

Enlace GitHub: https://github.com/leidylondonom/Monografia_Deteccion_de_fraude

2. DESCRIPCIÓN DEL PROBLEMA

El fraude contra el sistema financiero constituye un gran problema para los bancos ya que ocasionan pérdida económica, pérdida de imagen y desconfianza de los clientes. En la actualidad los delitos financieros representan uno de los problemas más graves para las autoridades y las instituciones bancarias. Cada año un gran porcentaje de dinero es robado por este tipo de actividades delictivas, así como también, son comprometidos los datos y la privacidad de muchas personas.

La tarea de detección de fraude no es un tema fácil de resolver, teniendo en cuenta las múltiples modalidades y evolución rápida que este tema ha tenido en la actualidad. El aumento significativo del fraude, su complejidad cada vez mayor y su especialización hacen que los recursos utilizados por parte de las organizaciones para combatirlos sean cada vez más significativos. La búsqueda de fraude es un trabajo altamente especializado que consume una gran cantidad de tiempo; gracias a la Ciencia de Datos, y en especial al Machine Learning, este trabajo se automatiza logrando una solución óptima.

2.1 PROBLEMA DE NEGOCIO

Las entidades financieras están constantemente expuestas a las pérdidas económicas por transacciones fraudulentas, lo cual las lleva a mejorar continuamente los procesos para gestionar y prevenir dicho fraude. Se les hace necesario contar con estrategias para detectar rápidamente las amenazas y estar un paso al frente de este delito, una de éstas, es contar con modelos analíticos predictivos apropiados, que revelen oportunamente las actividades sospechosas, adicional, contar con mecanismos que les ayuden a reportar y escalar los eventos anormales.

Si las empresas son capaz de actuar para reducir al mínimo el efecto del fraude, las pérdidas ocasionadas se reducirían al mínimo, y por lo tanto, este tipo de estudios podrían marcar la diferencia dentro de las compañías, por lo que su importancia podría llegar a ser vital para su futuro, y así poder conocer los patrones que definen a los clientes fraudulentos, para poder actuar en consecuencia y generar regla de negocio que ayuden a mitigar estos posibles eventos, evitando así futuros fraudes.

2.2 APROXIMACIÓN DESDE LA ANALÍTICA DE DATOS

El objetivo principal de este estudio en términos generales consiste en comprender de la forma más exacta posible, el comportamiento de la variable “Fraude” en base a otras variables; en términos estadísticos este conocimiento reflejará cuánta capacidad de predecir tienen las variables para detectar el fraude.

El uso de modelos predictivos ayudará a implementar estrategias encaminadas a limitar ya sea el número de transacciones o los montos transferidos, en aquellos clientes que según su comportamiento transaccional puedan ser susceptibles a cometer fraudes, también permitirán disminuir las tasas de dichos fraudes y el costo.

2.3 ORIGEN DE LOS DATOS

En el dataset utilizado la mayor parte de las variables son continuas y serán utilizadas como variables explicativas para modelizar la variable objetivo (isfraud), en total tenemos 11 variables:

Variable	Tipo	Frecuencia	Datos faltantes
step	Numérica	6362619	0
isfraud	Respuesta/Binaria	6362619	0
type	String	6362619	0
amount	Numérica	6362619	0
nameOrig	String	6362619	0
oldbalanceOrg	Numérica	6362619	0
newbalanceOrig	Numérica	6362619	0
nameDest	String	6362619	0
oldbalanceDest	Numérica	6362619	0
newbalanceDest	Numérica	6362619	0
isFlaggedFraud	Numérica	6362619	0

2.4 MÉTRICAS DE DESEMPEÑO

En cada modelo implementado se utilizan las métricas de exactitud, precisión y sensibilidad con el fin de validar su comportamiento; con los resultados obtenidos se ratifica que para conjuntos de datos desbalanceados resulta poco válida la métrica de exactitud ya que puede generar datos engañosos por dicho desbalanceo; adicional se hace uso de la curva ROC, la cual representa la tasa de verdaderos positivos frente a falsos positivos en diferentes umbrales de clasificación.

Métricas de negocio:

- Disminuir la tasa de transacciones fraudulentas
- Aumentar el nivel de satisfacción de los clientes con respecto a la seguridad que le brinda realizar transacciones bancarias
- Disminuir el riesgo del lavado de activos en una compañía financiera
- Aplicabilidad en la toma de decisiones asertivas

Al final esperamos que el modelo definido para detectar las transacciones fraudulentas pueda identificar un alto porcentaje de ellas e impactar de manera positiva los indicadores de la compañía y la confianza de los clientes, teniendo en cuenta que para el negocio es mucho más riesgoso dejar pasar un fraude que detener una transacción sospechosa que no era fraudulenta.

3. DATOS

3.1 DATOS ORIGINALES

La fuente de datos tomada de Kaggle (https://www.kaggle.com/ntnu-testimon/paysim1), es el producto de una simulación de transacciones de dinero móvil, basadas en una muestra de transacciones reales extraídas de un mes de registros financieros, implementado en un país africano. Los registros originales fueron proporcionados por una empresa multinacional, proveedora de servicio móvil financiero. Los datos utilizados se reducen a 1/4 del conjunto de datos original, lo cual corresponde a 6 millones de transacciones aproximadamente.

Para acceder a dichos datos es necesario crear una cuenta en Kaggle (https://www.kaggle.com). En el perfil ingresar a la opción: account, crear un nuevo api token, el cual descargara un archivo .json, necesario para acceder a los datos.

La mayor parte de las variables son variables continuas que serán utilizadas como variables explicativas para modelizar la variable objetivo, en total.

La información que se tiene es la siguiente:
step: Asigna una unidad de tiempo en el mundo real. Representa la hora y fecha en la que se realizó la transacción
isFraud: Variable respuesta, donde 1 representa una transacción fraudulenta
type: Cash-in, Cash-out, débito, pago, transferencia
amount: Monto de la transacción
nameOrig: Cliente que inició la transacción
oldbalanceOrg: Saldo inicial del remitente antes de la transacción
newbalanceOrig: Nuevo saldo del remitente después de la transacción
nameDest: Cliente destinatario de la transacción
oldbalanceDest: Saldo inicial del destinatario antes de la transacción
newbalanceDest: Nuevo saldo del destinatario después de la transacción
isFlaggedFraud: Intentos ilegales. Un intento ilegal en este conjunto de datos es un intento de transferir más de 200.000 en una sola transacción

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
3	1	CASH_OUT	181.00	C840083671	181.00	0.0	C38997010	21182.0	0.00	1	0
15	1	CASH_OUT	229133.94	C905080434	15325.00	0.0	C476402209	5083.0	51513.44	0	0
42	1	CASH_OUT	110414.71	C768216420	26845.41	0.0	C1509514333	288800.0	2415.16	0	0
47	1	CASH_OUT	56953.90	C1570470538	1942.02	0.0	C824009085	70253.0	64106.18	0	0
48	1	CASH_OUT	5346.89	C512549200	0.00	0.0	C248609774	652637.0	6453430.91	0	0

Tabla 1. Dataset original

3.2 DATASETS

Una consideración importante en la creación de modelos es evitar el sobre entrenamiento, lo cual se logra separando el conjunto de datos en dos subconjuntos (entrenamiento y

prueba). Uno mayor que corresponde aproximadamente al 70% de los datos originales que será el conjunto que usaremos para entrenar el modelo; el 30% restante lo usaremos para medir la precisión del modelo creado.

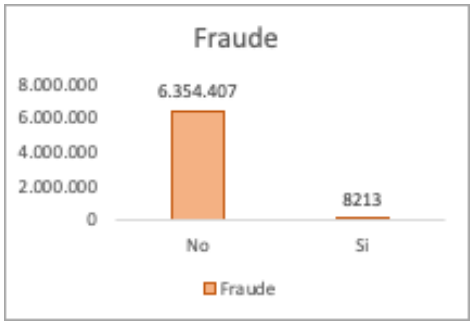
Además, se debe separar la variable objetivo del resto de variables, para la fase de entrenamiento.

Ya que nuestro variable objetivo “IsFraud”, está fuertemente desbalanceada debemos utilizar la opción *stratify*, de manera que dividamos nuestro conjunto de entrenamiento y test de manera que se mantenga los casos positivos y negativos en las mismas proporciones.

```
X_train, X_test, y_train, y_test = train_test_split(X.drop(['isFraud'], axis = 1), X['isFraud'],
                                                    random_state=0, test_size=0.3, stratify=X['isFraud'])
```

3.3 DESCRIPTIVA

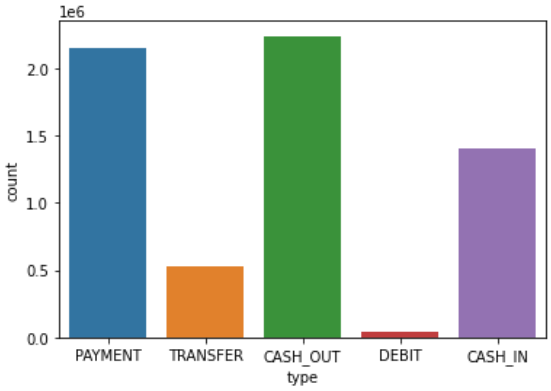
Realizamos análisis descriptivo de las variables que tenemos en el dataset para tener un mayor contexto del problema y poder entender el comportamiento transaccional cuando se realiza un fraude, nuestra base inicial cuenta con 6 millones de transacciones que no presentan fraude y sólo 8 mil transacciones marcadas como fraudes.



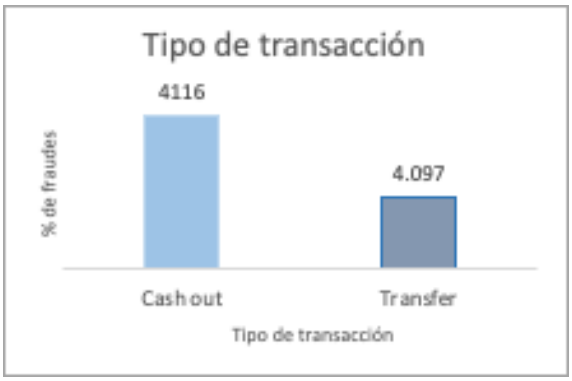
Gráfica 1. Distribución original de la variable respuesta

Además, cómo se distribuye el número de transacciones de acuerdo con el tipo, donde la mayor concentración está en transacciones de payment, cash out y cash in, pero cuando

revisamos en qué transacciones se están ocasionando los fraudes vemos que es en Cash-Out y Transfer donde se presentan.



Gráfica 2. Distribución tipo de transacción



Gráfica 3. Distribución de transacciones de fraude

Como se indicó anteriormente, encontramos que las transacciones de fraude se presentan en los tipos de transacción Cash-out y Transfer, por lo que tomamos la decisión de realizar el modelamiento teniendo en cuenta sólo estos tipos de transacción, de esta manera pasamos de tener 6 millones de transacciones a 2,7 millones.

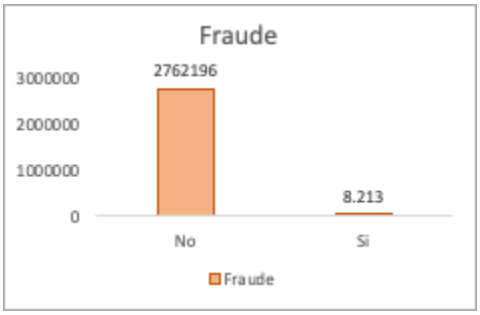


Gráfico 4. Distribución final de la variable respuesta

Eliminamos las variables NameOrig y NameDest ya que corresponde a identificadores de los usuarios que enviaron y recibieron el dinero de la transacción. Se categorizó la variable type (0 cuando la transacción sea transfer y 1 cuando la transacción sea cash out).

isFlaggedFraud no parece correlacionarse con ninguna variable explicativa. Se identifica isFlaggedFraud cuando se intenta transferir una cantidad superior a 200.000. De hecho,

como se muestra a continuación, isFlaggedFraud puede permanecer sin configurar a pesar de que se cumpla esta condición.

Revisamos de esta manera también el comportamiento de la variable Step (Asigna una unidad de tiempo en el mundo real. Representa la hora y fecha en la que se realizó la transacción), esta variable fue transformada para entender el comportamiento de las transacciones de fraude y revisar en qué franjas se están presentando más eventos.

Cuando comparamos la distribución de transacciones totales, vemos que en las horas de la madrugada claramente se nota una disminución de las transacciones, pero cuando vemos la distribución sólo con las transacciones de fraude estas presentan un comportamiento muy homogéneo o con una distribución uniforme en todas las horas del día.

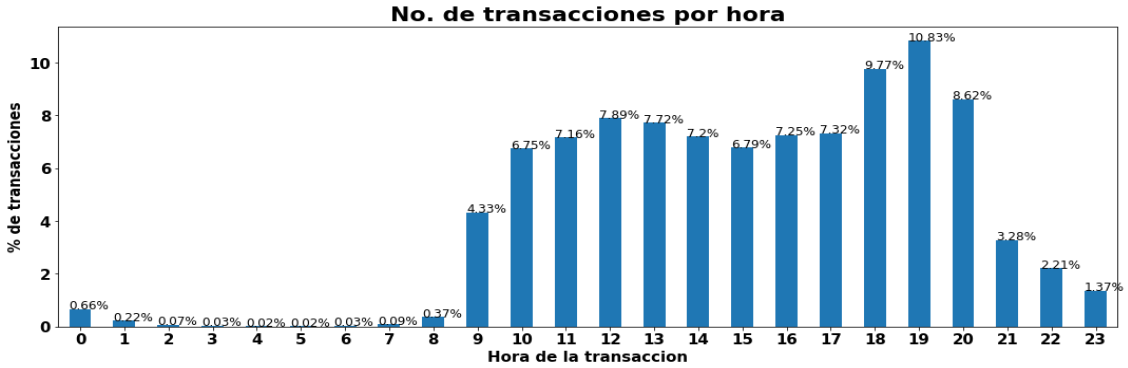


Gráfico 5. Distribución de las transacciones por hora

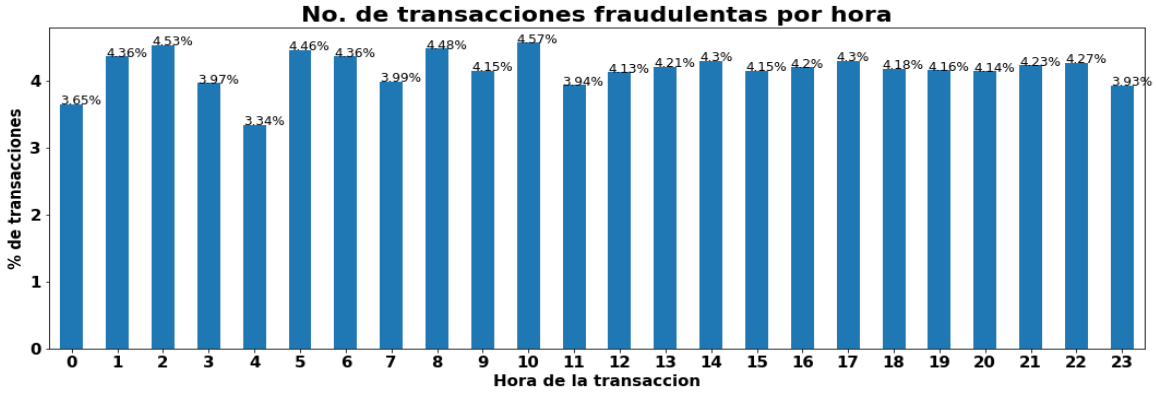


Gráfico 6. Distribución de las transacciones de fraude por hora

Esta es la base final con la que vamos a trabajar:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
2	1	0	181.00	181.00	0.0	0.00	0.00	1
3	1	1	181.00	181.00	0.0	21182.00	0.00	1
15	1	1	229133.94	15325.00	0.0	5083.00	51513.44	0
19	1	0	215310.30	705.00	0.0	22425.00	0.00	0
24	1	0	311685.89	10835.00	0.0	6267.00	2719172.89	0

Tabla 2. Dataset final luego de la limpieza y eliminación de variables

Generamos la matriz de correlación:

	step	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
step	1.000000	-0.016022	0.037007	0.005744	-0.011280	0.037778	0.037475	0.048671
type	-0.016022	1.000000	-0.326040	-0.013202	0.018726	-0.099794	-0.157024	-0.042400
amount	0.037007	-0.326040	1.000000	0.120389	0.018296	0.307133	0.497027	0.070660
oldbalanceOrg	0.005744	-0.013202	0.120389	1.000000	0.778826	-0.020403	-0.010029	0.347582
newbalanceOrig	-0.011280	0.018726	0.018296	0.778826	1.000000	-0.012277	-0.015439	0.063557
oldbalanceDest	0.037778	-0.099794	0.307133	-0.020403	-0.012277	1.000000	0.970060	-0.014960
newbalanceDest	0.037475	-0.157024	0.497027	-0.010029	-0.015439	0.970060	1.000000	-0.008978
isFraud	0.048671	-0.042400	0.070660	0.347582	0.063557	-0.014960	-0.008978	1.000000

Tabla 3. Correlación de las variables

Observamos que la variable más correlacionada con isFraud es oldbalanceOrg, con una correlación de 0,347. Aunque las demás variables tienen correlaciones muy bajas no las podemos despreciar ya que no contamos con más información para determinar las transacciones de fraude.

Cuando revisamos las correlaciones entre las variables predictoras, lo ideal es que no se presenten correlaciones altas; no obstante, las variables newbalanceDest y oldbalanceDest

tienen una correlación muy alta de 0.97, es decir, son prácticamente la misma variable, en este caso podríamos excluir una de las dos, pero teniendo en cuenta el reducido número de variables del Dataset empleado decidimos dejarlas.

4. PROCESO DE ANALÍTICA

4.1 PREPROCESAMIENTO

Ya que nuestra base está altamente desbalanceada vamos a realizar la modelación de dos maneras diferentes y compararemos los resultados:

1. Usaremos la función stratify de la librería scikit-learn para balancear, con el objetivo de dividiendo nuestros datos de entrenamiento y test mantenemos la proporción de valores negativos y positivos.
2. Utilizaremos la técnica de sobremuestreo oversampling, que se emplea para incrementar el número de observaciones positivas que corresponde a la clase minoritaria. Para esto se utilizará el algoritmo SMOTE con el cual se realizará un muestreo con reemplazo

```
from imblearn.over_sampling import SMOTE
smt = SMOTE()
X_train_SMOTE, y_train_SMOTE = smt.fit_sample(X_train_n, y_train)
```

Antes de proceder a aplicar los algoritmos normalizamos todos los datos de las variables categóricas, que se habían transformado, a matriz de enteros

```
X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
```

Presentamos un conjunto de datos sintéticos utilizando el método SMOTE los cuales asemejan el funcionamiento normal de las transacciones e inyecta comportamientos maliciosos para luego evaluar el desempeño de los métodos de detección de fraude.

4.2 MODELOS

Cada modelo considerado tendrá unos hiperparámetros que se ajustarán de una forma adecuada para obtener el mejor resultado posible. Para ello se utilizará la técnica de ajuste de búsqueda de rejilla (Grid Search), de forma que se entrenará cada modelo con la combinación de modelos posible y posteriormente se evaluará con validación cruzada para obtener la mejor combinación, y así, ahorrar tiempo, esfuerzo y recursos.

Dentro de las técnicas de predicción estadística aplicadas para predecir la variable "Fraude", utilizaremos los modelos de Regresión Logística, Random Forest, Naive Bayes y Redes Neuronales.

4.2.1. REGRESIÓN LOGÍSTICA

Este modelo clasifica las instancias según la probabilidad de pertenecer a alguno de los valores de la variable respuesta, siempre encontrándose en la franja de 0 ó 1.

En nuestro caso se incluye el parámetro weight = "balanced" y con esto el algoritmo se encargará de equilibrar a la clase minoritaria durante el entrenamiento.

```
clf = LogisticRegression(class_weight='balanced')
grid_values = {'penalty': ['l2'],'C':[0.001,0.01,0.8]}
grid_clf = GridSearchCV(clf, param_grid = grid_values, cv=skf, scoring = 'recall')
```

```
grid_clf.fit(X_train_n, y_train)
```

Después de aplicar el GridSearch obtenemos que los mejores hiperparámetros para el modelo, han sido con una regularización L2, más penalización de 0.8

```
LogisticRegression(C=0.8, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

4.2.2. RANDOM FOREST

El random forest funciona como una combinación de árboles predictores en donde cada árbol actuará como un árbol independiente

```
B=[5, 10, 15]
grid_values1 = {'n_estimators': B, 'max_depth':[3, 4]}

clf1 = RandomForestClassifier(random_state=0, class_weight='balanced_subsample', n_jobs=-1)
grid_clf1 = GridSearchCV(estimator=clf1, param_grid = grid_values1, cv=st, scoring='balanced_accuracy',return_train_score=True)
grid_clf1.fit(X_train_n, y_train)
```

Los mejores estimadores del Grid son

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                       class_weight='balanced_subsample', criterion='gini',
                       max_depth=4, 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=15, n_jobs=-1, oob_score=False,
                       random_state=0, verbose=0, warm_start=False)
```

4.2.3. NAIVE BAYES

Este modelo se basa en una técnica de clasificación estadística llamada “Teorema de Bayes”. En dicho modelo se asume que las variables predictoras son independientes entre sí, es decir, que la presencia de cierta característica en un conjunto de datos no está en absoluto relacionada con la presencia de cualquier otra característica, lo cual proporciona una manera fácil de construir modelos con un comportamiento muy bueno debido a su simplicidad. Esto se consigue proporcionando una forma de calcular la probabilidad ‘posterior’ de que ocurra un cierto evento A, dadas algunas probabilidades de eventos ‘anteriores’. Es útil para conjuntos de datos muy grandes.

```
grid_values2 = {'var_smoothing': [1e-2, 1e-4, 1e-6, 1e-10, 1e-15]}
```

```
grid_clf2 = GridSearchCV(GaussianNB(), cv=skf, param_grid=grid_values2)
grid_clf2.fit(X_train_n, y_train)
```

Los mejores estimadores del Grid son:

```
GaussianNB(priors=None, var_smoothing=0.01)
```

4.2.3. RED NEURONAL

Las redes neuronales son una de las familias de algoritmos de Machine Learning. Se trata de una técnica que se inspira en el funcionamiento de las neuronas de nuestro cerebro. Se basan en que dados unos parámetros hay una forma de combinarlos para predecir un cierto resultado. Los datos de entrada van pasando secuencialmente por distintas "capas" en las que se aplican una serie de reglas de aprendizaje moduladas por una función peso. Tras pasar por la última capa, los resultados se comparan con el resultado "correcto", y se van ajustando los parámetros (dados por las funciones "peso"). Aunque los algoritmos y en general el proceso de aprendizaje son complejos, una vez la red ha aprendido, puede congelar sus pesos y funcionar en modo recuerdo o ejecución.

Adicional se puede utilizar la métrica Loss para penalizar las clases mayoritarias.

```
modelA= keras.models.Sequential()
modelA.add(keras.layers.Dense(25, input_dim=X_train_n.shape[1], activation='relu', kernel_initializer='he_uniform'))
modelA.add(keras.layers.Dropout(0.4))
modelA.add(keras.layers.Dense(10, input_dim=X_train_n.shape[1], activation='relu', kernel_initializer='he_uniform'))
modelA.add(keras.layers.Dense(1, activation='sigmoid'))

opt = keras.optimizers.SGD(lr=0.01, momentum=0.9)
modelA.compile(loss = 'binary_crossentropy', optimizer = opt, metrics=[tf.keras.metrics.AUC(curve='ROC')])
```

4.3 MÉTRICAS

Como se comentó anteriormente es común para la evaluación de modelos de clasificación utilizar la métrica de exactitud, pero en los casos en los que la clase objetivo es muy desbalanceada esta métrica no es válida y puede llevar a resultados erróneos, para calcular la exactitud se utiliza la siguiente ecuación

$$Exactitud = \frac{VP + VN}{VP + FP + FN + VN}$$

Donde;

VP, corresponde a los verdaderos positivos

VN, corresponde a los verdaderos negativos

FP, corresponde a los falsos positivos

FN, corresponde a los falsos negativos

Para nuestro caso de estudio teniendo en cuenta que los datos están altamente desbalanceados el clasificador que siempre tiene cero tendrá una exactitud casi siempre del 100% ya que casi el total de las observaciones pertenece a dicha clase. Por esto es mejor usar métricas como la precisión y la sensibilidad, o una métrica combinada de ambas como es el valor F.

$$Precisión = \frac{VP}{VP+FP} \qquad\qquad Sensibilidad = \frac{VP}{VP+FN}$$

$$F1 = \frac{2 * Precisión * Sensibilidad}{Precisión + Sensibilidad}$$

5. RESULTADOS

5.1 MÉTRICAS

Inicialmente se ejecutan los modelos con los datos desbalanceados sin aplicar ninguna técnica de sobremuestreo, analizando las métricas de precisión y recall (sensibilidad) ya que como se evidenciará en los resultados, la métrica de accuracy se puede ver sesgada por la clase mayoritaria reflejando buenos resultados que no siempre son reales.

- Modelo **Regresión Logística**

```
print(confusion_matrix(y_test, pred))
```

```
[[782064  46595]
 [   263   2201]]
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	828659
1	0.05	0.89	0.09	2464
accuracy			0.94	831123
macro avg	0.52	0.92	0.53	831123
weighted avg	1.00	0.94	0.97	831123

Se evidencia que el modelo acierta 2201 muestras y falla en 263, dando un recall (sensibilidad) de 0.89, adicional se nota que la métrica f1-score es muy baja, en este punto es cuando nos debemos preguntar qué es mejor para el negocio y para la empresa financiera, tener que revisar los falsos positivos manualmente o fallar en la detección de los verdaderos casos de fraude. Teniendo en cuenta la baja precisión que existe en relación con una alta sensibilidad, se puede evidenciar una buena detección de la clase objetivo, pero se está incluyendo muestras de la otra clase.

El riesgo de este caso se puede ver reflejado en la satisfacción del cliente, ya que se está tomando como falso positivo 46595 transacciones; y en el tiempo dedicado por los analistas para validar dichos casos, sin embargo, es el modelo que mejor detecta los fraudes.

- Modelo **Random Forest**

```
print(confusion_matrix(y_test, pred1))
```

```
[[822888   5771]
 [  1315   1149]]
```

	precision	recall	f1-score	support
0	1.00	0.99	0.99	828659
1	0.16	0.60	0.25	2464
accuracy			0.99	831123
macro avg	0.58	0.80	0.62	831123
weighted avg	1.00	0.99	0.99	831123

Se evidencia un resultado más bajo con el acierto de 1149 muestras, se nota un aumento en la precisión y disminución en la sensibilidad en relación con el modelo anterior, con lo que se puede concluir que el modelo no detecta la clase muy bien, pero cuando lo hace es altamente confiable.

- Modelo **Naive Bayes**

```
print(confusion_matrix(y_test, pred2))
```

```
[[818818   9841]
 [  1475    989]]
```

	precision	recall	f1-score	support
0	1.00	0.99	0.99	828659
1	0.09	0.40	0.15	2464
accuracy			0.99	831123
macro avg	0.54	0.69	0.57	831123
weighted avg	1.00	0.99	0.99	831123

Con este modelo se evidencia una disminución tanto en la precisión como en la sensibilidad, lo cual indica que dicho modelo no logra clasificar la clase correctamente.

- Modelo **Red Neuronal**

```
([[828621,     38],
  [   1042,   1422]])
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	828659
1	0.97	0.58	0.72	2464
accuracy			1.00	831123
macro avg	0.99	0.79	0.86	831123
weighted avg	1.00	1.00	1.00	831123

En este caso, el modelo realiza una buena detección de muestras, con una disminución de falsos positivos, pero tiene desacierto en 1042 transacciones fraudulentas, lo cual no es muy bueno para la compañía financiera.

En resumen y teniendo en cuenta los resultados de lo modelos sin datos sintéticos, el mejor para el caso de detección de fraude, es la *Regresión Logística* ya que para el negocio es mucho más riesgoso dejar pasar un fraude que detener una transacción sospechosa que no era fraudulenta

Modelo	Exactitud	Precisión	Sensibilidad	Score F1
Regresión Logística	0.94	0.05	0.89	0.09
Random Forest	0.99	0.16	0.60	0.25
Naive Bayes	0.99	0.09	0.40	0.15
Red Neuronal	1	0.97	0.58	0.72

Tabla 4. Resultados de los modelos sin datos sintéticos

RESULTADOS DE LAS MÉTRICAS USANDO SOBREMUESTREO

- Modelo **Regresión Logística**

```
[[784638  44021]
 [   246   2218]]
```

	precision	recall	f1-score	support
0	1.00	0.95	0.97	828659
1	0.05	0.90	0.09	2464
accuracy			0.95	831123
macro avg	0.52	0.92	0.53	831123
weighted avg	1.00	0.95	0.97	831123

En comparación con los resultados de este modelo aplicado en datos desbalanceados, no se observa una diferencia significativa.

- Modelo **Random Forest**

```
[[818189  10470]
 [  1145   1319]]
```

	precision	recall	f1-score	support
0	1.00	0.99	0.99	828659
1	0.11	0.54	0.19	2464
accuracy			0.99	831123
macro avg	0.56	0.76	0.59	831123
weighted avg	1.00	0.99	0.99	831123

Teniendo en cuenta los bajos valores en las métricas de este modelo, podemos concluir que no logran clasificar los fraudes correctamente aún con datos sintéticos.

- Modelo **Naive Bayes**

```
[[809442  19217]
 [  1256   1208]]
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	828659
1	0.06	0.49	0.11	2464
accuracy			0.98	831123
macro avg	0.53	0.73	0.55	831123
weighted avg	1.00	0.98	0.98	831123

Este modelo presenta un comportamiento similar al Random Forest, lo cual nos lleva a deducir que no siempre la generación de datos sintéticos es el mejor camino para resolver casos con datos desbalanceados, esto depende mucho del problema propiamente que se esté tratando y su contexto.

- Modelo **Red Neuronal**

```
([[824612,   4047],
  [   323,   2141]])
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	828659
1	0.35	0.87	0.49	2464
accuracy			0.99	831123
macro avg	0.67	0.93	0.75	831123
weighted avg	1.00	0.99	1.00	831123

Este modelo detecta bien los fraudes, pero tiene un alto número de falsos positivos.

Teniendo en cuenta los siguientes resultados de los modelos aplicando el sobremuestreo, se observa que la ***Regresión Logística*** sigue siendo el mejor modelo para la detección de fraude, aunque presenta alto número de falsos positivos, para las empresas financieras es más importante lograr detectar los fraudes para que no se vea afectado los ingresos de la compañía.

Modelo	Exactitud	Precisión	Sensibilidad	Score F1
Regresión Logística	0.95	0.05	0.90	0.09
Random Forest	0.99	0.11	0.54	0.19
Naive Bayes	0.98	0.06	0.49	0.11
Red Neuronal	0.99	0.35	0.87	0.49

Tabla 5. Resultados de los modelos con datos sintéticos

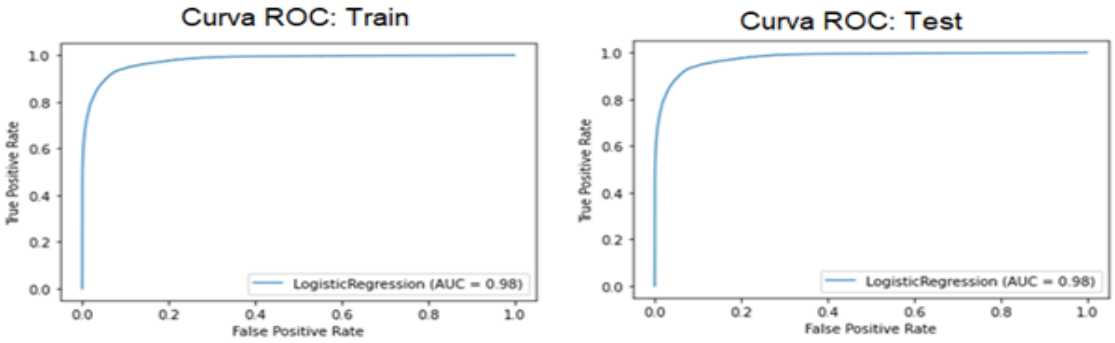
Finalmente, comparando la eficiencia de los modelos con y sin datos sintéticos no tenemos diferencias significativas en los resultados, pero si en el costo computacional y tiempo de espera en las estimaciones de los parámetros ya que fueron bastante altos en comparación al tiempo y validaciones sin sobremuestreo.

5.2 EVALUACIÓN CUALITATIVA

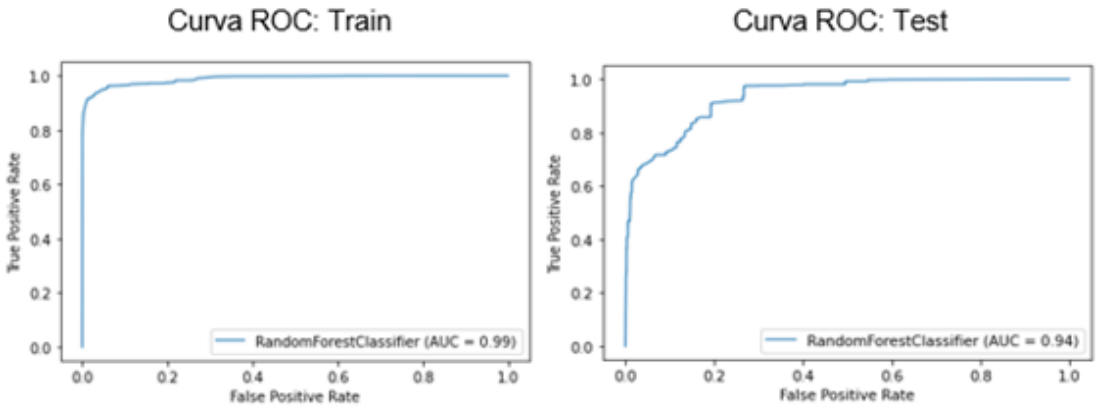
Realizamos el análisis de la curva ROC en todos los modelos validando su comportamiento con los datos de entrenamiento y prueba y así lograr evidenciar si se presenta overfitting o underfitting, dicha curva muestra la compensación entre la tasa de verdaderos positivos y la tasa de falsos positivos, clasifica estadísticamente bajo hipótesis el mejor modelo que obtenga dado a su aprendizaje y entrenamiento.

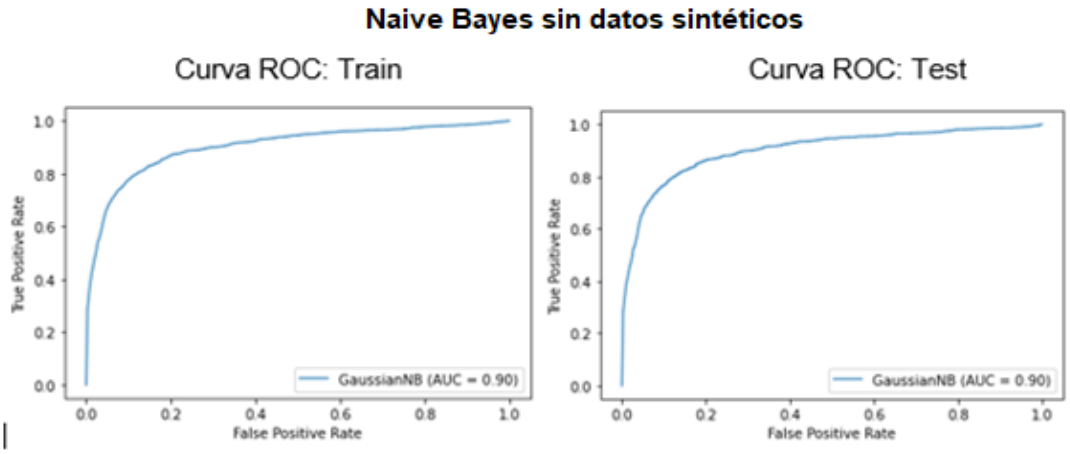
Curvas Roc sin utilizar técnicas de ***sobremuestreo***

Regresión Logística sin datos sintéticos



Random Forest sin datos sintéticos





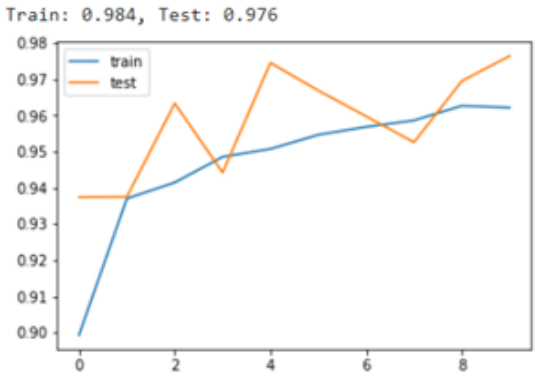
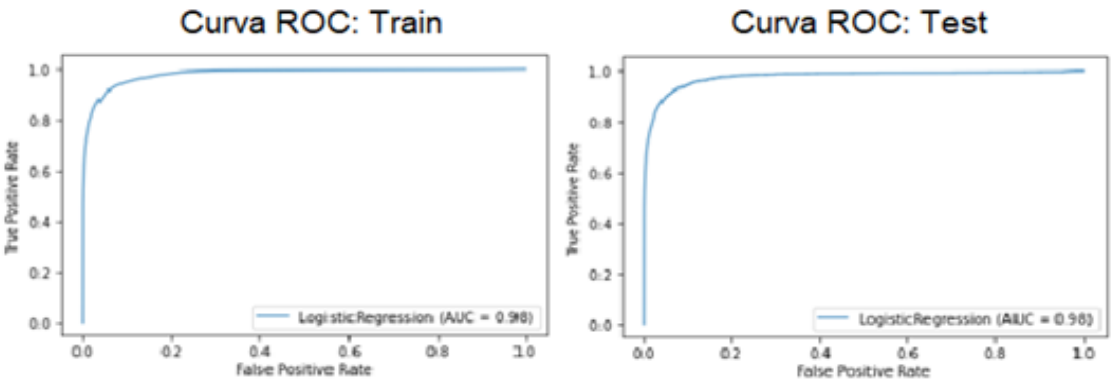


Gráfico 7. Curvas ROC de los modelos (Regresión Logística, Random Forest, Naive Bayes y la Red Neuronal) en entrenamiento y prueba sin datos sintéticos

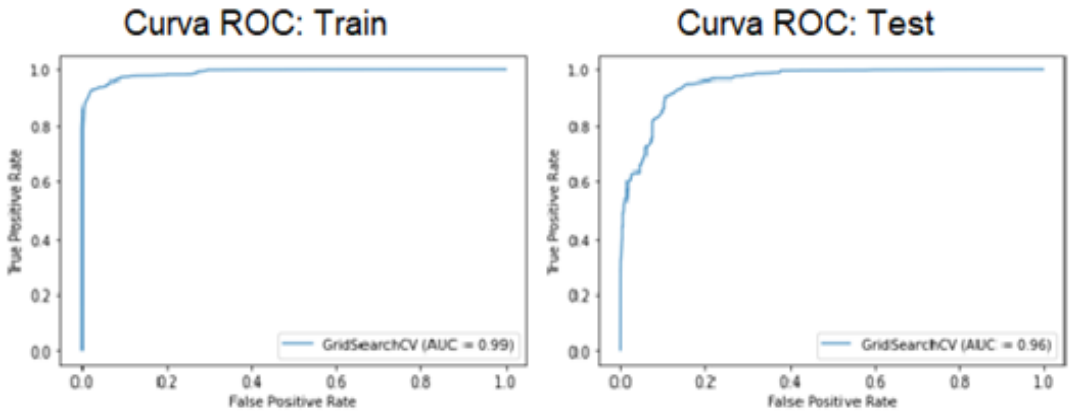
Teniendo en cuenta el resultado del análisis no se presenta overfitting ni underfitting significativo, solo en el modelo de Random Forest se presenta un bajo underfitting, entre los datos de entrenamiento y prueba.

Curvas Roc utilizando técnicas de ***sobremuestreo*** (SMOTE)

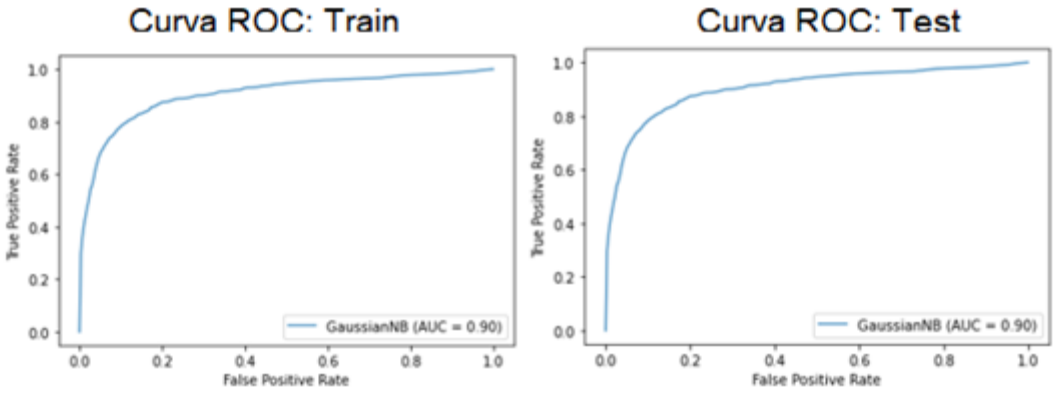
Regresión Logística con datos sintéticos



Random Forest con datos sintéticos



Naive Bayes con datos sintéticos



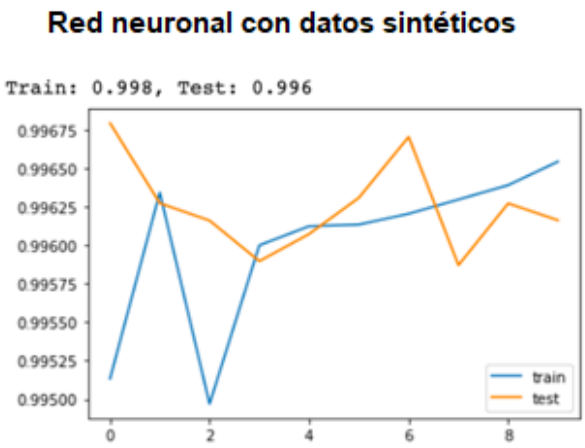


Gráfico 8. Curvas ROC de los modelos (Regresión logística, Random Forest, Naive Bayes y Red Neuronal) en entrenamiento y prueba CON datos sintéticos

En la validación con datos sintéticos, igualmente se observa en el modelo de Random Forest, un bajo underfitting, entre los datos de entrenamiento y prueba.

5.3 CONSIDERACIONES DE PRODUCCIÓN

Consideramos que para realizar una salida a producción es muy importante que se pueda contar con la información transaccional en tiempo real, ya que el tema que estamos tratando de detección de fraude requiere que sea identificado en el menor tiempo posible; y se tomen las medidas adecuadas, ya sea en bloqueos de cuentas, cancelación de transacción o posibles llamadas, para evitar los riesgos implícitos como son: lavado de dinero reputacional y pérdidas de dinero.

Adicional tener en cuenta cada cuanto se le debe hacer backtesting al modelo y evaluar que no se ha descalibrado y sigue dando los resultados esperados. Continuar evaluando nuevas métricas de los modelos y si es posible, mejorar las condiciones que no impliquen un alto costo computacional pero que tampoco se pierda valor en los resultados.

6. CONCLUSIONES

Se buscó implementar un modelo analítico que tuviera un buen ajuste a la hora de detectar transacciones fraudulentas realizadas a través del sistema de billetera móvil, cuyo costo operacional y tiempo de identificación fuera mínimo.

Partiendo del hecho que se cuenta con un dataset altamente desbalanceado y que existe gran variedad de técnicas para afrontar dicho problema, con sus pro y sus contras, se decide inicialmente aplicar varios modelos sin generar datos sintéticos y luego se aplicó la técnica de sobremuestreo de minorías (SMOTE). Adicional se aplican las métricas de exactitud, precisión y sensibilidad, dando mayor credibilidad a las dos últimas, ya que para este tipo de datos resulta poco válida la métrica de exactitud, porque puede generar datos engañosos por dicho desbalanceo, lo cual se evidencia en el resultado de los modelos implementados.

La comparación de eficiencia de los modelos con y sin datos sintéticos no reportó diferencias significativas en sus resultados, pero si en el costo computacional y tiempo de espera en las estimaciones de los parámetros ya que fueron bastante altos en comparación al tiempo y validaciones sin sobremuestreo.

La ejecución de los modelos utilizando datos sintéticos se puede ver afectada por el alto consumo de memoria, lo cual no es muy bueno para los casos de detección de fraude, ya que se necesita tener los resultados de forma rápida y oportuna.

En cuanto a los resultados, el mejor se obtuvo mediante el uso de la Regresión Logística, sin tener en cuenta el sobremuestreo. Aunque la totalidad de los modelos obtuvieron resultados similares con diferentes métricas, se evidencia un bajo desempeño en los modelos Naive Bayes y Random Forest.

Es importante tener en cuenta que la base con la que se trabaja tiene una gran cantidad de transacciones, que al implementar técnicas o modelos analíticos como una máquina de soporte vectorial (SVM) o realizar técnicas de sobremuestreo implican un costo computacional muy grande para lograr detectar de manera rápida aquellas transacciones que puedan ser fraude.

Como aprendizaje dentro de este trabajo nos queda la posibilidad no de trabajar con todo el dataset sino realizar una muestra aleatoria que nos permita operar muchos más modelos que puedan correr de manera fácil y rápida en una máquina convencional.

7. BIBLIOGRAFIA

- Á silicia, D.Rodriguez, J.Riquelme. (2009). SMOTE-I: Mejora del algoritmo SMOTE para balanceo de clases minoritarias. https://www.researchgate.net/publication/229045207_SMOTE-I_mejora_del_algoritmo_SMOTE_para_balanceo_de_clases_minoritarias

- Amat Rodrigo, Joaquín. (2020). Machine learning con Python y Scikit-learn. https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikit learn.html

- Brownlee, Jason. (2020). SMOTE for Imbalanced Classification with Python. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

- Dye, Steven. (2020). How to Handle SMOTE Data in Imbalanced Classification Problems. https://towardsdatascience.com/how-to-handle-smote-data-in-imbalanced-classification-problems-cf4b86e8c6a1