DETECCIÓN DE FRAUDE FINANCIERO MEDIANTE MODELOS DE APRENDIZAJE AUTOMÁTICO

Por:

Juan Pablo Gómez López

Introducción a la Inteligencia Artificial

Raúl Ramos Pollan

Universidad de Antioquia

Facultad de Ingeniería

Medellín

2023

Introducción

Las organizaciones que procesan transacciones a través de medios electrónicos tienen el riesgo de que las operaciones que transen resulten fraudulentas. Esto genera millonarias pérdidas para estas empresas, por lo que es importante un sistema de seguridad que sea capaz de detectar las operaciones de esta naturaleza. Con los datos proporcionados por Vesta Corporation en la competición lanzada por la IEEE Computational intelligence Society, se plantea el reto de predecir la probabilidad de que una transacción online sea fraudulenta.

El dataset a utilizar proviene de de transacciones bancarias reales del e-commerce de Vesta Corporation. La información principal está separada en cuatro archivos .csv: *train_transaction, train_identity, test_transaction y test_identity.* El campo (columna) que une a *transaction* con *identity* es *TransactionID.*

Contamos con el archivo sample_submission.csv el cuál nos muestra un pequeño ejemplo de cómo deberíamos retornar o mostrar el resultado final de nuestro reporte acerca de la veracidad de las transacciones trabajadas. Este archivo contiene la siguiente información:

- TransactionID - Identificador de las transacciones.
- isFraud - Muestra la probabilidad de que una transacción sea fraudulenta.

Los archivos de tipo “Transaction” se dividen en dos: test_transaction,csv y train_transaction,csv, uno para realizar pruebas y el otro para el entrenamiento del algoritmos, respectivamente. La información que contienen ambos es la siguiente:

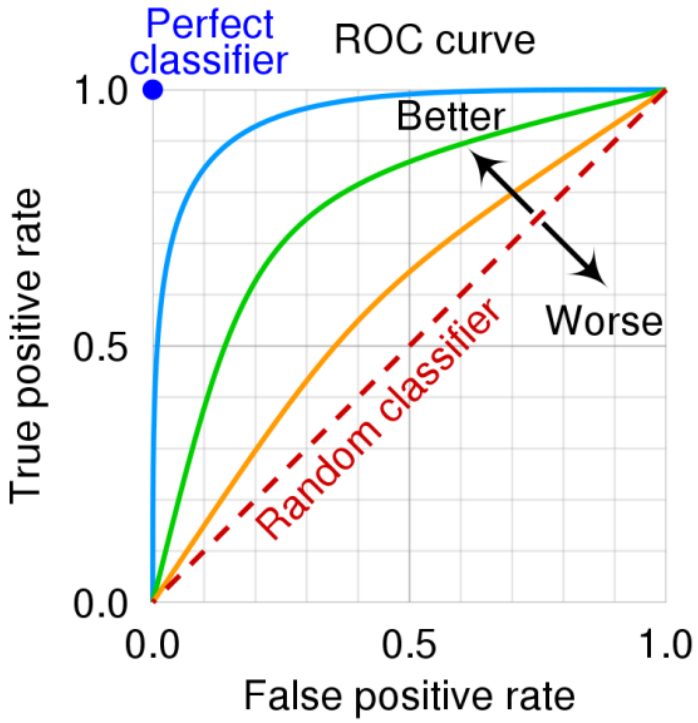
- ProductCD: Código de Producto,el producto o servicio de cada transacción
- card1 - card6: Información de la tarjeta de pago, tal como tipo de tarjeta, categoría, banco, país, entre otros.
- addr1, addr2: Dirección de facturación por región y país.
- P_emaildomain: Correo del comprador
- R_emaildomain: Correo del destinatario
- M1 - M9: Coincidencias en la información del nombre de la tarjeta, direcciones, entre otros.
- TransactionDT: timedelta de una fecha y hora de referencia dada.
- TransactionAMT: Monto del pago de la transacción en USD
- dist - Distancia entre las distintas direcciones
- C1 - C14: Contadores, tal como cuántas direcciones se encuentran asociadas a la tarjeta de pago, etc.
- D1-D15: timedelta, como días entre transacciones anteriores, etc.
- Vxxx: Características diseñadas por Vesta, que incluyen clasificación, conteo y otras relaciones de entidad.

Los archivos de tipo “Identity” se dividen en dos: test_identity,csv y train_identity,csv, uno para realizar pruebas y el otro para el entrenamiento del algoritmos, respectivamente. La información que contienen ambos es la siguiente:

- DeviceType: Tipo de dispositivo en el cual se realiza la transacción
- DeviceInfo: Información del dispositivo en el cual se realiza la transacción
- id_01 - id_38: Características asociadas a la identidad del usuario, anonimizadas.

Métricas

La métrica de evaluación principal es la curva ROC, donde se pide medir la probabilidad de que sea fraude para cada transacción. La curva ROC es la siguiente:



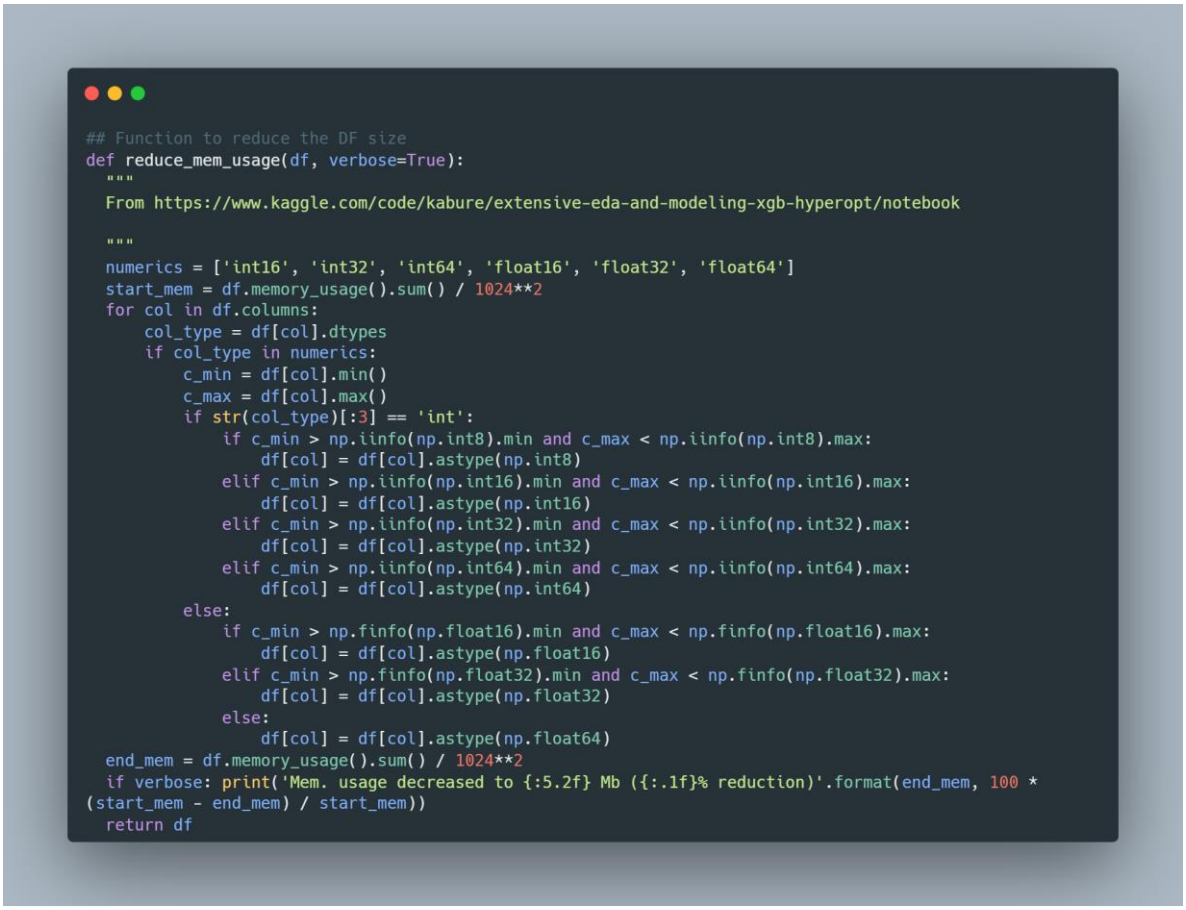
Obtenido de: https://upload.wikimedia.org/wikipedia/commons/3/36/ROC_space-2.png

Como métrica de negocio, la idea es disminuir lo máximo posible la cantidad de falsos positivos que el modelo arroja al analizar una transacción.

Para este proyecto, se trabajará sólo con el dataset de entrenamiento de la competencia, para no tener que estar enviando las predicciones para obtener un score de validación con el conjunto de prueba.

Exploración de datos

En primer lugar, vale la pena reportar la dificultad de la exploración por las limitaciones en memoria RAM de la máquina. Para solventar este problema, en las discusiones de la competencia en Kaggle un participante desarrolló una función para reducir el tamaño de la memoria usada por los datasets. La función es la siguiente:



```
## Function to reduce the DF size
def reduce_mem_usage(df, verbose=True):
  """
  From https://www.kaggle.com/code/kabure/extensive-eda-and-modeling-xgb-hyperopt/notebook

  """
  numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
  start_mem = df.memory_usage().sum() / 1024**2
  for col in df.columns:
      col_type = df[col].dtypes
      if col_type in numerics:
          c_min = df[col].min()
          c_max = df[col].max()
          if str(col_type)[:3] == 'int':
              if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                  df[col] = df[col].astype(np.int8)
              elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                  df[col] = df[col].astype(np.int16)
              elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                  df[col] = df[col].astype(np.int32)
              elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                  df[col] = df[col].astype(np.int64)
          else:
              if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                  df[col] = df[col].astype(np.float16)
              elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                  df[col] = df[col].astype(np.float32)
              else:
                  df[col] = df[col].astype(np.float64)
  end_mem = df.memory_usage().sum() / 1024**2
  if verbose: print('Mem. usage decreased to {:5.2f} Mb ({:.1f}% reduction)'.format(end_mem, 100 *
(start_mem - end_mem) / start_mem))
  return df
```

La reducción de memoria fue considerable , por lo que se pudo continuar con el proceso de exploración (y de entrenamiento) sin ningún percance.

El dataset de las transacciones tiene 590540 muestras con 394 columnas. El dataset con identificación de los clientes tiene 144233 muestras con 41 columnas. El *merged* dataset tiene en total 590540 filas con 178 columnas.

Vamos a usar una función que construye tablas para resumir los datos de cada columna, por grupos de interés como se establecieron en la introducción.

	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
0	TransactionID	int32	0.0	590540	2987000	2987001	2987002
1	isFraud	int8	0.0	2	0	0	0
2	TransactionDT	int32	0.0	573349	86400	86401	86469
3	TransactionAmt	float16	0.0	8195	68.5	29.0	59.0
4	ProductCD	object	0.0	5	W	W	W

	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
5	card1	int16	0.000000	13553	13926	2755	4663
6	card2	float16	1.512683	500	NaN	404.0	490.0
7	card3	float16	0.265012	114	150.0	150.0	150.0
8	card4	object	0.267044	4	discover	mastercard	visa
9	card5	float16	0.721204	119	142.0	102.0	166.0
10	card6	object	0.266028	4	credit	credit	debit

	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
13	dist1	float16	59.652352	2412	19.0	NaN	287.0
14	dist2	float16	93.628374	1699	NaN	NaN	NaN

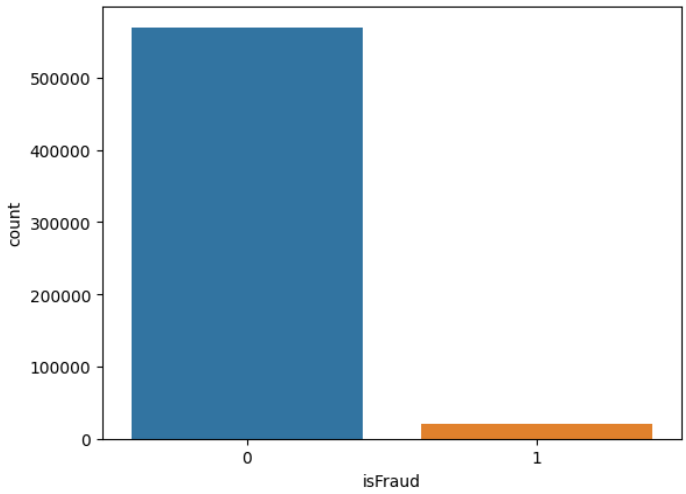
	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
4	ProductCD	object	0.0	5	W	W	W
17	C1	float16	0.0	1495	1.0	1.0	1.0
18	C2	float16	0.0	1167	1.0	1.0	1.0
19	C3	float16	0.0	27	0.0	0.0	0.0
20	C4	float16	0.0	1223	0.0	0.0	0.0
21	C5	float16	0.0	319	0.0	0.0	0.0
22	C6	float16	0.0	1291	1.0	1.0	1.0
23	C7	float16	0.0	1069	0.0	0.0	0.0
24	C8	float16	0.0	1130	0.0	0.0	0.0
25	C9	float16	0.0	205	1.0	0.0	1.0
26	C10	float16	0.0	1122	0.0	0.0	0.0
27	C11	float16	0.0	1343	2.0	1.0	1.0
28	C12	float16	0.0	1066	0.0	0.0	0.0
29	C13	float16	0.0	1464	1.0	1.0	1.0
30	C14	float16	0.0	1108	1.0	1.0	1.0

	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
31	D1	float16	0.214888	641	14.0	0.0	0.0
32	D2	float16	47.549192	641	NaN	NaN	NaN
33	D3	float16	44.514851	649	13.0	NaN	NaN
34	D4	float16	28.604667	808	NaN	0.0	0.0
35	D5	float16	52.467403	688	NaN	NaN	NaN
36	D6	float16	87.606767	829	NaN	NaN	NaN
37	D7	float16	93.409930	597	NaN	NaN	NaN
38	D8	float16	87.312290	5367	NaN	NaN	NaN
39	D9	float16	87.312290	24	NaN	NaN	NaN
40	D10	float16	12.873302	818	13.0	0.0	0.0
41	D11	float16	47.293494	676	13.0	NaN	315.0
42	D12	float16	89.041047	635	NaN	NaN	NaN
43	D13	float16	89.509263	577	NaN	NaN	NaN
44	D14	float16	89.469469	802	NaN	NaN	NaN
45	D15	float16	15.090087	859	0.0	0.0	315.0

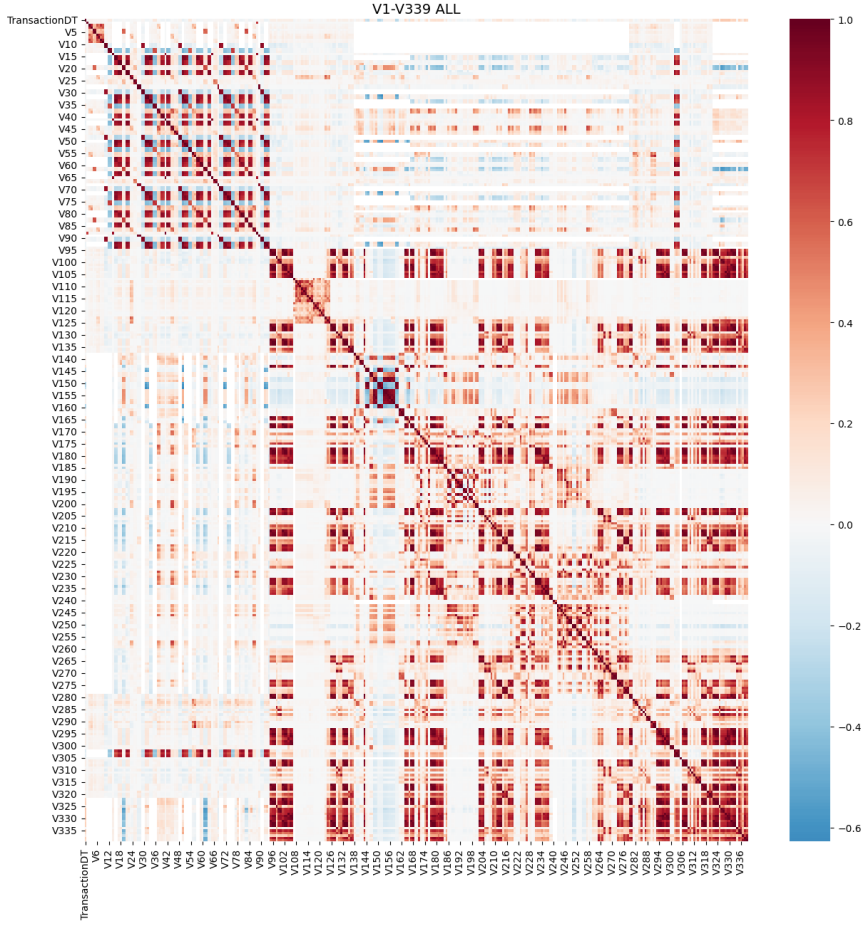
	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
46	M1	object	45.907136	2	T	NaN	T
47	M2	object	45.907136	2	T	NaN	T
48	M3	object	45.907136	2	T	NaN	T
49	M4	object	47.658753	3	M2	M0	M0
50	M5	object	59.349409	2	F	T	F
51	M6	object	28.678836	2	T	T	F
52	M7	object	58.635317	2	NaN	NaN	F
53	M8	object	58.633115	2	NaN	NaN	F
54	M9	object	58.633115	2	NaN	NaN	F

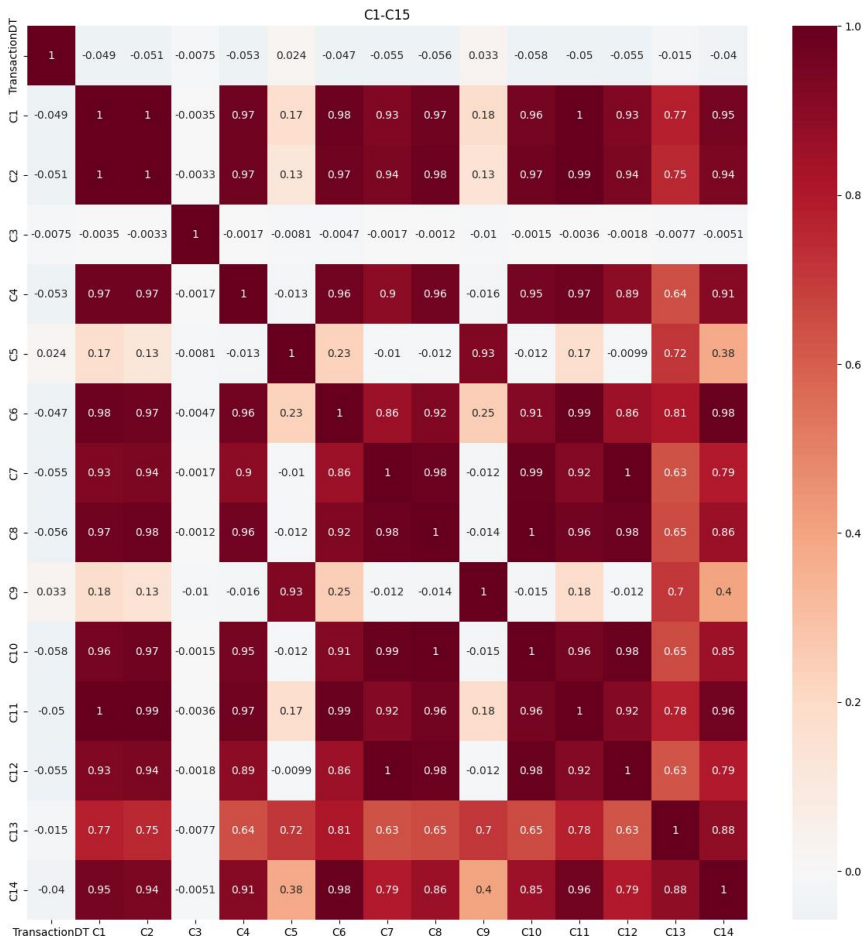
	Nombre	dtypes	% NaN	Unique	Primer Valor	Segundo Valor	Tercer Valor
55	V1	float16	47.293494	2	1.0	NaN	1.0
56	V2	float16	47.293494	9	1.0	NaN	1.0
57	V3	float16	47.293494	10	1.0	NaN	1.0
58	V4	float16	47.293494	7	1.0	NaN	1.0
59	V5	float16	47.293494	7	1.0	NaN	1.0
...	...	...	...	...	...	...	...
389	V335	float16	86.054967	669	NaN	NaN	NaN
390	V336	float16	86.054967	355	NaN	NaN	NaN
391	V337	float32	86.054967	254	NaN	NaN	NaN
392	V338	float32	86.054967	380	NaN	NaN	NaN
393	V339	float32	86.054967	334	NaN	NaN	NaN

Una observación importante para recalcar es que el dataset está fuertemente desbalanceado, que de hecho es muy común en este tipo de problemas.



Algunas columnas están altamente correlacionadas, como se muestran en los siguientes gráficos.





Preprocesado de datos

Para el preprocesado, se va a hacer la siguiente estrategia: Se eliminaron las columnas que tengan más del 80% de los datos como null; se hizo imputación de datos faltantes en las columnas numéricas con la mediana, y se aplica one-hot encoding para las categóricas, añadiendo una columna adicional para los datos faltantes.

A continuación se muestra un resumen de la cantidad de datos nulos por columna

```
1 k = train.isna().sum()
2 k[k!=0]
```

```
card2           8933
card3           1565
card4           1577
card5           4259
card6           1571
                ...
id_36         449555
id_37         449555
id_38         449555
DeviceType    449730
DeviceInfo    471874
Length: 158, dtype: int64
```

Dividimos los datos en numéricos y categóricos para si respectiva limpieza.

```
1 X_n = X.loc[:,X.dtypes != object]
2 X_c = X.loc[:,X.dtypes == object]
3 X_n.shape, X_c.shape
```

```
((590540, 150), (590540, 26))
```

Los datos numéricos luego de ser imputados con la mediante usando SimpleImputer de la librería de sklearn, queda así:

```
1 imp_median_test = SimpleImputer(missing_values=np.nan, strategy='median') # Imputación con la mediana
2 X_n_imp = pd.DataFrame(imp_median_test.fit_transform(X_n),
3                        columns = X_n.columns)
4 X_n_imp = reduce_mem_usage(X_n_imp)
5 X_n_imp.head()
```

```
Mem. usage decreased to 179.09 Mb (73.5% reduction)
```

	TransactionID	TransactionAmt	card1	card2	card3	card5	addr1	addr2	dist1	C1	...	V320	id_01	id_02	id_05	id_06	id_11	id_13	id_17	id_19	id_20
0	2987000.0	68.5	13928.0	361.0	150.0	142.0	315.0	87.0	19.0	1.0	...	0.0	-5.0	125800.5	0.0	0.0	100.0	52.0	166.0	341.0	472.0
1	2987001.0	29.0	2756.0	404.0	150.0	102.0	325.0	87.0	8.0	1.0	...	0.0	-5.0	125800.5	0.0	0.0	100.0	52.0	166.0	341.0	472.0
2	2987002.0	59.0	4664.0	490.0	150.0	166.0	330.0	87.0	287.0	1.0	...	0.0	-5.0	125800.5	0.0	0.0	100.0	52.0	166.0	341.0	472.0
3	2987003.0	50.0	18128.0	567.0	150.0	117.0	476.0	87.0	8.0	2.0	...	0.0	-5.0	125800.5	0.0	0.0	100.0	52.0	166.0	341.0	472.0
4	2987004.0	50.0	4496.0	514.0	150.0	102.0	420.0	87.0	8.0	1.0	...	0.0	0.0	70787.0	0.0	0.0	100.0	52.0	166.0	542.0	144.0

5 rows × 150 columns

Quedando finalmente sin datos nulos:

```
1 X_n_imp.isna().sum().sum()
```

```
0
```

La codificación de los datos categóricos se hace usando one-hot añadiendo una columna adicional para los datos faltantes, por cada columna categórica. Como observación, se decidió eliminar la columna DeviceInfo ya que estaba mejor representada en la columna DeviceType de manera más sencilla, y no agregaría muchas más columnas a la codificación.

Además, se codificaron los correos electrónicos para ser identificados directamente por la empresa, y reducir la dimensionalidad de la codificación.

```
# Vamos a mapear los emails para mayor simplicidad.
emails = {'gmail': 'google', 'att.net': 'att', 'twc.com': 'spectrum',
          'scranton.edu': 'other', 'optonline.net': 'other', 'hotmail.co.uk': 'microsoft',
          'comcast.net': 'other', 'yahoo.com.mx': 'yahoo', 'yahoo.fr': 'yahoo',
          'yahoo.es': 'yahoo', 'charter.net': 'spectrum', 'live.com': 'microsoft',
          'aim.com': 'aol', 'hotmail.de': 'microsoft', 'centurylink.net': 'centurylink',
          'gmail.com': 'google', 'me.com': 'apple', 'earthlink.net': 'other', 'gmx.de': 'other',
          'web.de': 'other', 'cfl.rr.com': 'other', 'hotmail.com': 'microsoft',
          'protonmail.com': 'other', 'hotmail.fr': 'microsoft', 'windstream.net': 'other',
          'outlook.es': 'microsoft', 'yahoo.co.jp': 'yahoo', 'yahoo.de': 'yahoo',
          'servicios-ta.com': 'other', 'netzero.net': 'other', 'suddenlink.net': 'other',
          'roadrunner.com': 'other', 'sc.rr.com': 'other', 'live.fr': 'microsoft',
          'verizon.net': 'yahoo', 'msn.com': 'microsoft', 'q.com': 'centurylink',
          'prodigy.net.mx': 'att', 'frontier.com': 'yahoo', 'anonymous.com': 'other',
          'rocketmail.com': 'yahoo', 'sbcglobal.net': 'att', 'frontiernet.net': 'yahoo',
          'ymail.com': 'yahoo', 'outlook.com': 'microsoft', 'mail.com': 'other',
          'bellsouth.net': 'other', 'embarqmail.com': 'centurylink', 'cableone.net': 'other',
          'hotmail.es': 'microsoft', 'mac.com': 'apple', 'yahoo.co.uk': 'yahoo', 'netzero.com': 'other',
          'yahoo.com': 'yahoo', 'live.com.mx': 'microsoft', 'ptd.net': 'other', 'cox.net': 'other',
          'aol.com': 'aol', 'juno.com': 'other', 'icloud.com': 'apple'}

us_emails = ['gmail', 'net', 'edu']
```

Una parte del dataset queda así

Mem. usage decreased to 137.48 Mb (85.9% reduction)

	ProductCD_C	ProductCD_H	ProductCD_R	ProductCD_S	ProductCD_W	card4_american express	card4_discover	card4_mastercard
0	0	0	0	0	1	0	1	0
1	0	0	0	0	1	0	0	1
2	0	0	0	0	1	0	0	0
3	0	0	0	0	1	0	0	1
4	0	1	0	0	0	0	0	1

5 rows × 243 columns

Finalmente se concatena la parte numérica con la parte categórica, y además **se particiona de una vez el data set en parte de entrenamiento y parte de prueba,** para trabajar siempre con los mismos datos en el resto del trabajo. El dataset final quedó con 394 columnas.

Modelos supervisados

Inicialmente se comienza con una selección de modelos para trabajar con el que posiblemente sea el mejor modelo. Se proponen las siguientes tres plantillas de modelos:

1. Regresión logística
2. Árboles de decisión
3. GaussianNB

Se encontró lo siguiente: La regresión logística no logró converger en el número máximo de pasos, y los otros dos mostraron un desempeño más o menos similar, siendo los Árboles de decisión el mejor modelo por ahora.

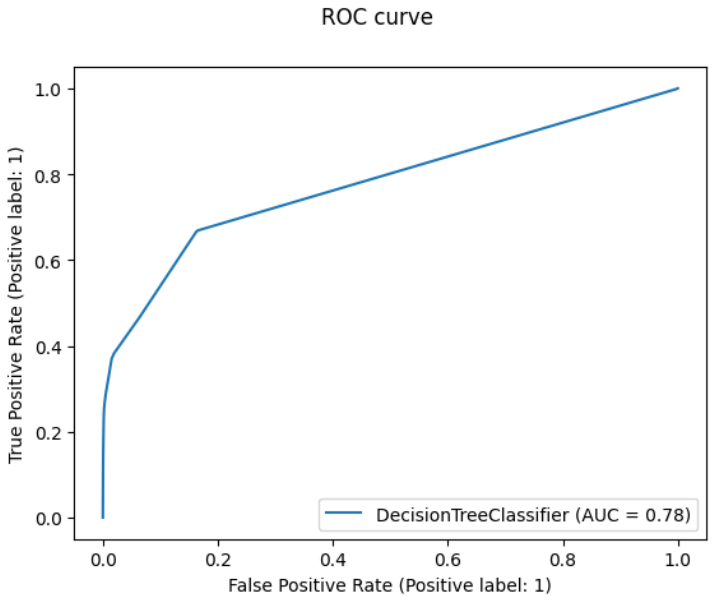
```
test score   0.635 (±0.0164) with 5 splits
train score  0.633 (±0.0151) with 5 splits
--
test score   0.775 (±0.0044) with 5 splits
train score  0.781 (±0.0065) with 5 splits
--
test score   0.715 (±0.0083) with 5 splits
train score  0.712 (±0.0050) with 5 splits
selecting  1

selected model
DecisionTreeClassifier(max_depth=5)
```

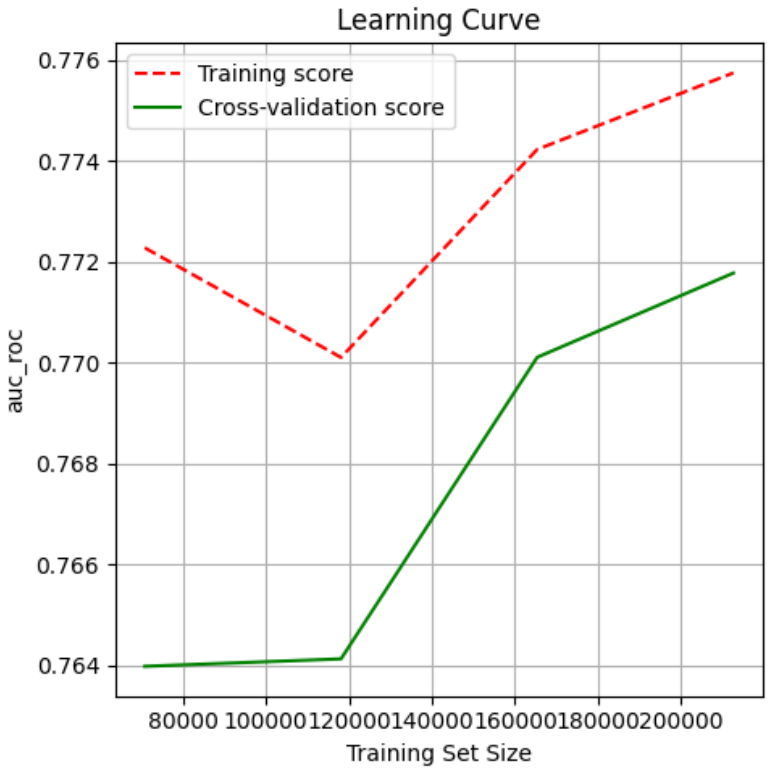
Reentrenando este modelo se reporta el siguiente desempeño en el conjunto de datos test:

```
2 best_estimator.fit(X_train, y_train)
3
4 y_score = best_estimator.predict_proba(X_test)[:, 1]
5 reported_performance = roc_auc_score(y_true = y_test, y_score = y_score)
6 print ("reported performance of selectd model %.3f"%reported_performance)
```

```
reported performance of selectd model 0.779
```



La curva de aprendizaje para el mejor modelo es la siguiente:



Finalmente se hace un gridsearch para el mejor modelo, obteniendo los siguientes resultados:

```
Mejor estimador Decision Tree:  DecisionTreeClassifier(max_depth=10)
Mejores parámetros para el estimador Decision Tree:  {'max_depth': 10}
```

```
ROC_AUC del Decision Tree en entrenamiento:  0.85603
ROC_AUC del Decision Tree seleccionado:  0.84424
```

Observando una mejoría significativa respecto a los modelos anteriores.

Modelos no supervisados + supervisados

En este caso se propone aplicar PCA y NMF antes de aplicar los mismos modelos supervisados de la sección anterior. La metodología es la misma, seleccionamos el mejor modelo y comenzamos a hacer pruebas con el.

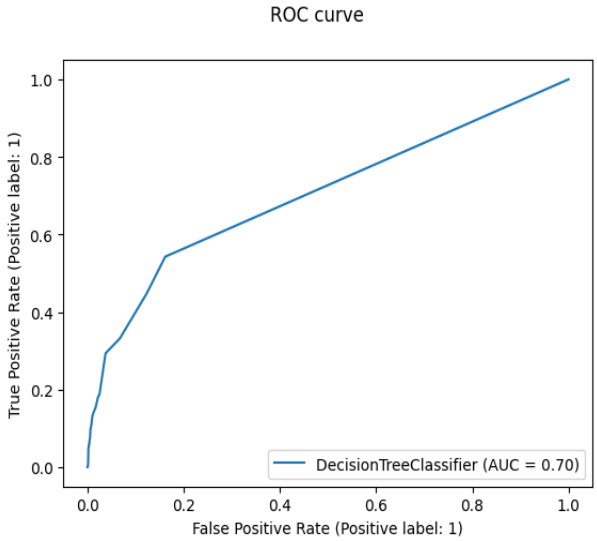
PCA + Modelo supervisado

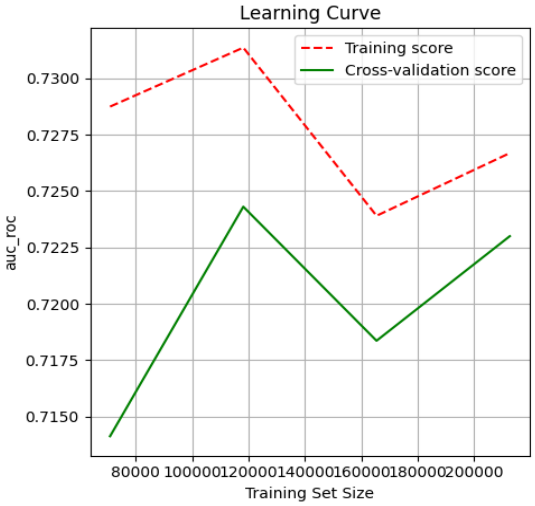
Se propone un PCA con 300 componentes, y los resultados son los siguientes:

```
test score   0.615 (±0.0227) with 5 splits
train score  0.613 (±0.0199) with 5 splits
--
test score   0.774 (±0.0048) with 5 splits
train score  0.779 (±0.0062) with 5 splits
--
test score   0.705 (±0.0049) with 5 splits
train score  0.707 (±0.0056) with 5 splits
selecting  1

selected model
DecisionTreeClassifier(max_depth=5)
```

Donde se reporta un rendimiento con los datos de prueba de 0.703. El rendimiento es inferior a sólo usar el Árbol de decisión solo. A continuación, se muestra la curva de aprendizaje y la curva ROC.





NMF + Modelo supervisado

Se aplicó la misma estrategia que con el PCA, con la diferencia que en este caso se debe eliminar las columnas con valores negativos. Las siguientes columnas tienen valores negativos.

```
D4        True
D11       True
D15       True
id_01     True
id_05     True
id_06     True
dtype: bool
```

Aplicando este método para 300, 250 y 200 componentes, se obtuvo lo siguiente:

```
roc_auc del modelo con  300 elementos:  0.50000
------------
roc_auc del modelo con  250 elementos:  0.50000
------------
roc_auc del modelo con  200 elementos:  0.50000
------------
Mejor roc_auc:  0.50000   ; obtenido con  300  componentes para NMF
```

Donde el rendimiento es muy por debajo, incluso que el PCA.

Retos y consideraciones de despliegue

La idea de colocar un modelo de esta naturaleza en producción es tal que analice de forma automática las transacciones de los clientes de X organización, y que catalogue con buena precisión las operaciones que considere como fraudulentas, y que además, el número las transacciones etiquetadas como fraude que en realidad no lo sean (falsos positivos) sea el más bajo posible.

En este momento, la precisión está muy por debajo de lo que sería aceptable para un modelo de esta naturaleza, lo que muestra el reto de tratar de predecir clientes fraudulentos mediante tarjetas de crédito.

Conclusiones

- El reto principalmente será en la ingeniería de características, ya que al leer los resultados de los ganadores de la competición , se observa mucho trabajo en esta parte, además de introducir técnicas y conceptos no vistos en el curso hasta el momento.
- Se debe tener muy presente la cantidad de recursos disponible en la máquina para hacer todo el flujo de trabajo de ML, demostrando que estas tecnologías son pesadas en recursos computacionales.
- El mejor algoritmo hasta el momento es el Árbbol de decisión con max_depth de 10, obteniendo 0.84424 de roc_auc_score para los datos de prueba (los nunca antes vistos).
- Ninguna técnica de reducción de dimensionalidad ayudó para mejorar el desempeño del modelo. Sin embargo, no se hizo una búsqueda muy exhaustiva en el número de componentes, por lo que esta parte se debe mejorar.

Discusiones de apoyo

Kaggle, «IEEE Fraud Detection - First Look and EDA,» 23 04 2023. [En línea]. Available: https://www.kaggle.com/code/robikscube/ieee-fraud-detection-first-look-and-eda/notebook#Train-vs-Test-are-Time-Series-Split.

Kaggle, «Extensive EDA and Modeling XGB Hyperopt,» 23 04 2023. [En línea]. Available: https://www.kaggle.com/code/kabure/extensive-eda-and-modeling-xgb-hyperopt/notebook#Target-Distribution.

Kaggle, «EDA and models,» 23 04 2023. [En línea]. Available: https://www.kaggle.com/code/artgor/eda-and-models/notebook#Data-loading-and-overview.

Kaggle, «IEEE Transaction columns Reference,» 23 04 2023. [En línea]. Available: https://www.kaggle.com/code/alijs1/ieee-transaction-columns-reference/notebook.