INTELIGENCI

PROYECTO

Kenneth Romero López

New Technology School

Tokio.

New
Technology
School

NOMBRE Y APELLIDOS

KENNETH ROMERO LÓPEZ

Fecha: 7/4/2025

INTELIGENCIA ARTIFICIAL
PROYECTO FINAL
Nombre y Apellidos

Tokio.

DESARROLLO

1. **Definición del problema y sus objetivos.**

   Para el proyecto final se parte de unos datos que recogen transacciones con tarjeta de crédito y se especifica si son fraudulentas o no. El objetivo del proyecto es, mediante el uso de diversos modelos de inteligencia artificial, poder predecir si una futura transferencia es segura. Se trata de aprendizaje supervisado, puesto que en los datos de entrenamiento contamos con la columna *label* que indica qué tipo de transacción es. Se emplearán principalmente modelos de aprendizaje automático con Scikit-learn, así como una prueba de *deep learning* con TensorFlow y Keras.

2. **Definición de los datos**

   Los datos que he empleado para este proyecto final los he obtenido de Kaggle.

   Se trata de un archivo CSV con 8 columnas con información sobre las transacciones. Veámoslas en mayor detalle:

   - Distance from home: La distancia de dónde tuvo lugar la transacción respecto a la residencia del titular. No he encontrado cuál es la unidad empleada, he sobreentendido que son kilómetros o millas.
   - Distance from last transaction: La distancia respecto a la última transacción.

- Ratio to median purchase price: La relación entre la cantidad de la transacción actual y la mediana de la cantidad de transacciones previas del titular (mayor que 1 indica que la transacción es mayor y menos que 1 que es menor).
- Repeat retailer: Especifica, mediante valores binarios (1-0), si la transacción ha tenido lugar en el mismo vendedor.
- Used chip: Si se ha empleado chip o no.
- Used pin number: Si en la transacción se ha introducido el código PIN.
- Online order: Si se trata o no de una compra por internet.
- Fraud: Por último, la *label* de este conjunto de datos, es decir, la confirmación de si la transacción ha sido fraudulenta o no.

999999	2.914856986796768,1.4726866870566833,0.21807548601452645,1.0,1.0,0.0,1.0,0.0
1000000	4.258729391720518,0.24202336594551443,0.4758220644483312,1.0,0.0,0.0,1.0,0.0
1000001	58.10812496080576,0.3181101207150046,0.3869198471921602,1.0,1.0,0.0,1.0,0.0

Este *dataset* tiene, en general, una calidad muy alta. Contiene 1.000.001 registros, por lo que los datos son muy abundantes. He decidido, para reducir el tiempo de computación, limitar las entradas a 50.000, viendo en clase que se han obtenido buenos resultados partiendo de datos con menos registros. A juzgar por las puntuaciones que han logrado algunos de los modelos empleados, creo que los 50.000 registros han sido suficientes.

```
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   distance_from_home              50000 non-null  float64
 1   distance_from_last_transaction  50000 non-null  float64
 2   ratio_to_median_purchase_price  50000 non-null  float64
 3   repeat_retailer                 50000 non-null  float64
 4   used_chip                       50000 non-null  float64
 5   used_pin_number                 50000 non-null  float64
 6   online_order                    50000 non-null  float64
 7   fraud                           50000 non-null  float64
dtypes: float64(8)
memory usage: 3.1 MB
None
```

Además de la cantidad, vemos que no hay valores nulos, y todas las filas son valores numéricos, ninguno categórico.

```
RECUENTO TRANSACCIONES FRAUDULENTAS/NO FRAUDULENTAS
Fraudulentas:  4317
No fraudulentas:  45683
```

El único problema que creo que tiene este *dataset* es que está muy desequilibrado. Es un caso similar a datos médicos, en los que la mayoría de pacientes no tienen una enfermedad y muy pocos sí. Lo mismo sucede con estos datos, ya que la mayor parte de transacciones son normales, mientras que solo una minoría es de carácter fraudulento, como, al fin y al cabo, cabría esperar. A pesar de esta limitación, como se ha mencionado anteriormente, pienso que ha habido modelos que han arrojado buenas métricas.

3. Preprocesado

La verdad es que, como he mencionado en el apartado anterior, los datos de este *dataset* son excelentes, a excepción del desequilibrio entre transacciones normales y fraudulentas.

Es por ello por lo que no he tenido que llevar a cabo ninguna limpieza exhaustiva de los datos.

```
# Cargamos los datos y reducimos su tamaño a 50 000 registros
df_completo = pd.read_csv('card_transdata.csv')
df = df_completo.iloc[:50000]
```

Me ha parecido, sin embargo, que un millón de registros era demasiado, y como ya he comentado, he decidido limitarlo.

Pienso además que todas las columnas son relevantes para la consecución de una buena predicción (no había, por ejemplo nombres de clientes, códigos postales… u otras categorías que no aportaran, a priori, nada al modelo) así que no he tenido que eliminar ninguna.

Tampoco había valores nulos que obligaran a eliminar registros o bien imputarlos.

4. Exploración de los datos y observaciones

Empecemos por la visualización del primer registro para ver un ejemplo de con qué datos tratamos y cómo son sus valores.

```
PRIMER REGISTRO

   distance_from_home  distance_from_last_transaction  \
0           57.877857                         0.31114

   ratio_to_median_purchase_price  repeat_retailer  used_chip  \
0                         1.94594              1.0        1.0

   used_pin_number  online_order  fraud
0              0.0           0.0    0.0
```

Antes hemos comprobado que no había valores nulos, podemos hacer lo mismo con los duplicados.

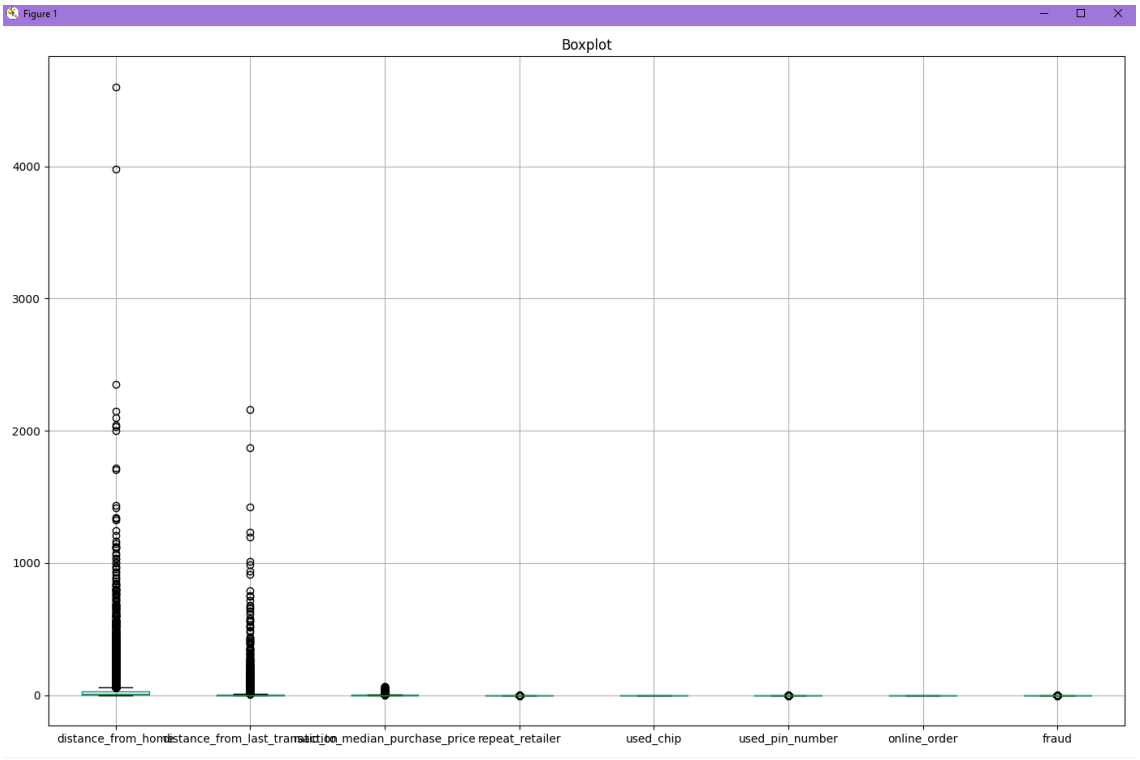
```
print("COMPROBACIÓN DE DUPLICADOS\n")
duplicados = df.duplicated().sum()
print("Número de duplicados: ", duplicados)
```

```
COMPROBACIÓN DE DUPLICADOS

Número de duplicados:  0
```

Sigamos con una familiarización general con los datos mediante el uso de `describe()`, tras lo cual resulta llamativo lo siguiente.

```
DESCRIPCIÓN DE LOS DATOS

       distance_from_home  distance_from_last_transaction  \
count        50000.000000                    50000.000000
mean            26.979933                        5.164165
std             68.502716                       26.713719
min              0.021322                        0.000705
25%              3.905343                        0.296871
50%             10.047077                        0.997287
75%             25.970365                        3.367123
max           4601.011222                     2160.499922
```

Vemos cómo en las dos primeras columnas hay una gran disparidad entre la media (26.97 y 5.16 respectivamente) y los valores máximos (4601.01 y 2160.49).



He empleado una gráfica de tipo boxplot para mostrar estos valores atípicos. Vemos cómo éstos se dan en las tres primeras columnas, en especial en las dos primeras, esto es, la distancia desde la residencia y la distancia de la transacción actual respecto a la anterior. Debemos tener en cuenta también que, pese a que hay muchos valores atípicos, en realidad también estamos trabajando con un volumen bastante grande (50.000 registros).

Puede haber casos en que estos valores tan dispares puedan ser fruto de errores humanos o de los ordenadores, y que ello pueda llevar a la distorsión de las métricas finales. Sin embargo, en este caso he creído que son importantes dada problemática con la que tratamos. Entiendo que tiene bastante sentido el hecho de que las transacciones fraudulentas por internet se hagan a mucha distancia del lugar de residencia del titular (o de la última compra) y por lo tanto, no solo no habría que eliminar estos *outliers*, sino que pueden aportar una información valiosa al modelo. Dicho con otras palabras, puede haber una relación entre la lejanía y el hecho de que sea una transacción fraudulenta (siempre con cautela, porque podría ser que la persona realmente estuviera de vacaciones, por ejemplo).

Sea como sea, si se trata de valores que tienen sentido a escala terrestre, no creo que sea necesario eliminarlos.

También he querido hacer más comparaciones mediante funciones pivotantes (mostraré solo un ejemplo con código y el resto solo los resultados).

```
pin_number = df[['used_pin_number', 'fraud']].groupby(['used_pin_number'],
                                                      as_index=False).mean().sort_values(by='fraud',ascending=False)
```

used_pin_number	fraud
0.0	0.095883
1.0	0.003486

used_chip	fraud
0.0	0.099273
1.0	0.062393

Aquí podemos apreciar como el hecho de que no se haya usado (en especial) PIN ni chip da valores más altos en el caso de las transacciones fraudulentas.

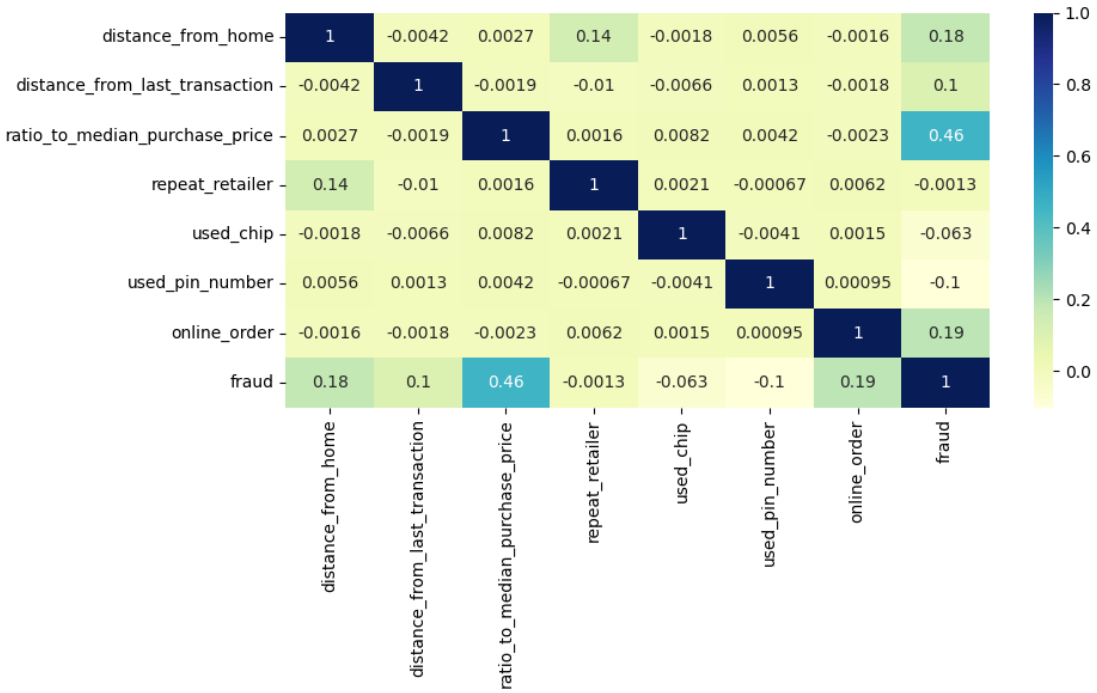
online_order	fraud
1.0	0.125891
0.0	0.012603

Por lo que respecta a las acciones realizadas por internet, vemos que éstas parecen ser el método preferido por los delincuentes, pues presentan números más elevados.

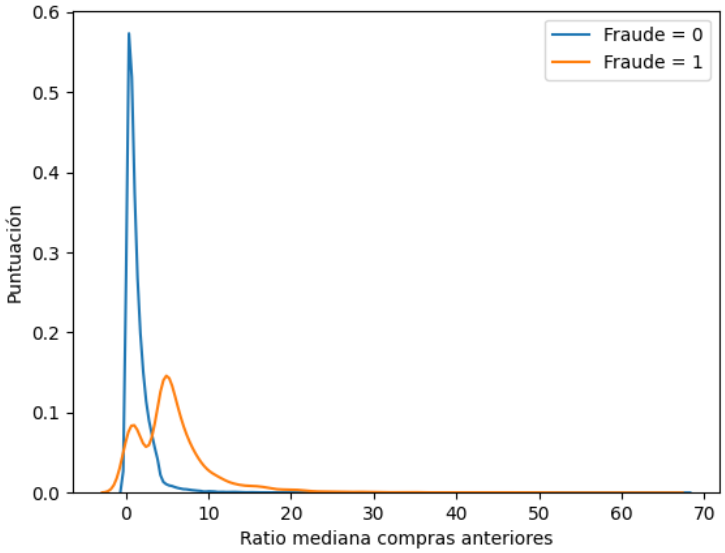
	distance_from_home	fraud
5935	1.953158	1.0
3049	1.178081	1.0
3057	1.180755	1.0
3055	1.180428	1.0
3053	1.179492	1.0
35548	22.024632	1.0
9514	2.952972	1.0
44231	54.026817	1.0
38015	27.163930	1.0
11270	3.485355	1.0
32715	17.587913	1.0
3045	1.176701	1.0
34144	19.578763	1.0
7606	2.406423	1.0
29262	13.543144	1.0
38009	27.149527	1.0
27448	11.909723	1.0
20333	7.259969	1.0
34149	19.586361	1.0
6307	2.049856	1.0

	distance_from_home	fraud
17337	5.779860	0.0
17338	5.780042	0.0
17339	5.780124	0.0
17340	5.780161	0.0
17341	5.781405	0.0
17342	5.781901	0.0
17343	5.782365	0.0
17344	5.782621	0.0
17345	5.783504	0.0
17346	5.783883	0.0
17347	5.784708	0.0
17348	5.784871	0.0
17349	5.785441	0.0
17350	5.785564	0.0
17351	5.785651	0.0
17352	5.785753	0.0
17353	5.786116	0.0
17354	5.786312	0.0
17355	5.786744	0.0
49999	4601.011222	0.0

Si hacemos lo mismo con la distancia de casa (haciendo un `head()` y un `tail()`), parece que pudiera haber una relación entre la distancia y el tipo de transacción, ya que hay valores más lejanos en las acciones fraudulentas. Sin embargo, pienso que no es una muestra representativa, y las gráficas posteriores parecen indicar que no hay una relación muy clara entre mayor distancia > mayor probabilidad de transacción fraudulenta, contradiciendo mi primera impresión.



A continuación vemos un mapa de calor con las diversas correlaciones entre los campos del *dataset*. Por desgracia no arroja resultados muy evidentes, pero podemos discernir que hay una correlación negativa (mínima) entre fraude y que no se usara PIN o chip. También, pero en positivo, que la transacción fuera por internet. El valor más claro es la ratio a la mediana del precio de compra, que es el que obtiene una puntuación más alta y por lo tanto tiene una mayor correlación.



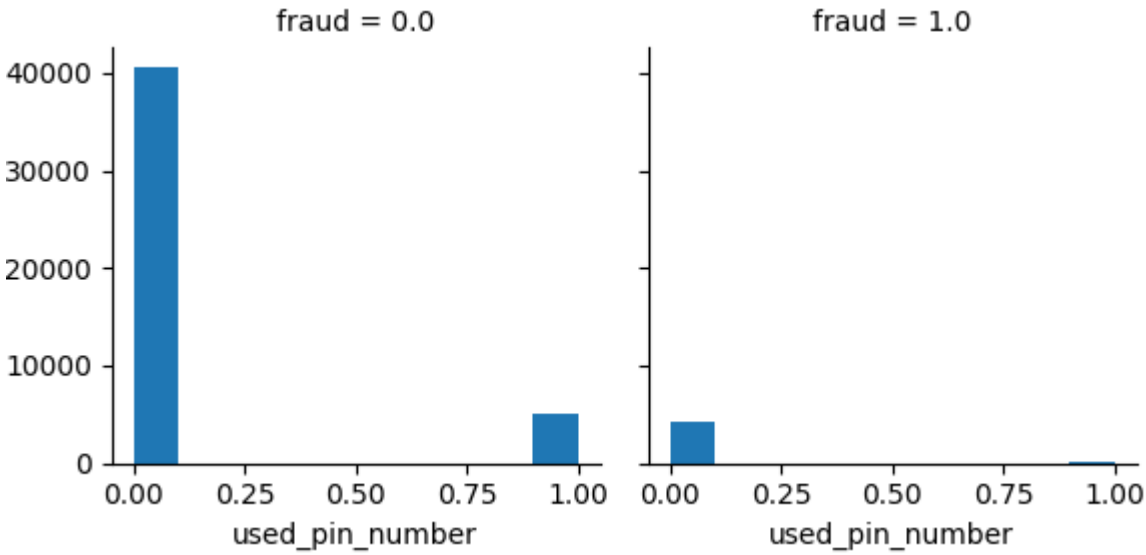
A priori no parecía una de las métricas más relevantes, pero a raíz del mapa de calor he decidido profundizar en los datos relacionados con la ratio, puesto que como se verá más adelante al final sí cobran una gran importancia. En esta gráfica KDE vemos en azul las transacciones normales y en naranja las fraudulentas. En el eje x se muestra la ratio en sí, es decir, cuanto más a la derecha esté la curva, mayor es la diferencia con las compras anteriores (más elevada será la cantidad económica de la transacción) y en el eje y, cuanto más alta sea la curva, más común será ese ratio en su grupo correspondiente, es decir, fraude/no fraude. Por lo tanto, vemos que la mayoría de movimientos que no son fraudulentos se mueven en valores similares, mientras que los que sí lo son se desplazan a la derecha, lo que significa un gasto mayor comparado con las transacciones normales.

```
DESCRIBE DE ratio_to_median_purchase_price AGRUPANDO POR FRAUDE
         count      mean       std       min       25%       50%       75%  \
fraud
0.0    45683.0  1.420570  1.934398  0.011373  0.449750  0.912069  1.784358
1.0     4317.0  5.809493  5.158130  0.032955  2.727026  4.930769  7.138892

             max
fraud
0.0    67.601896
1.0    64.459199
```

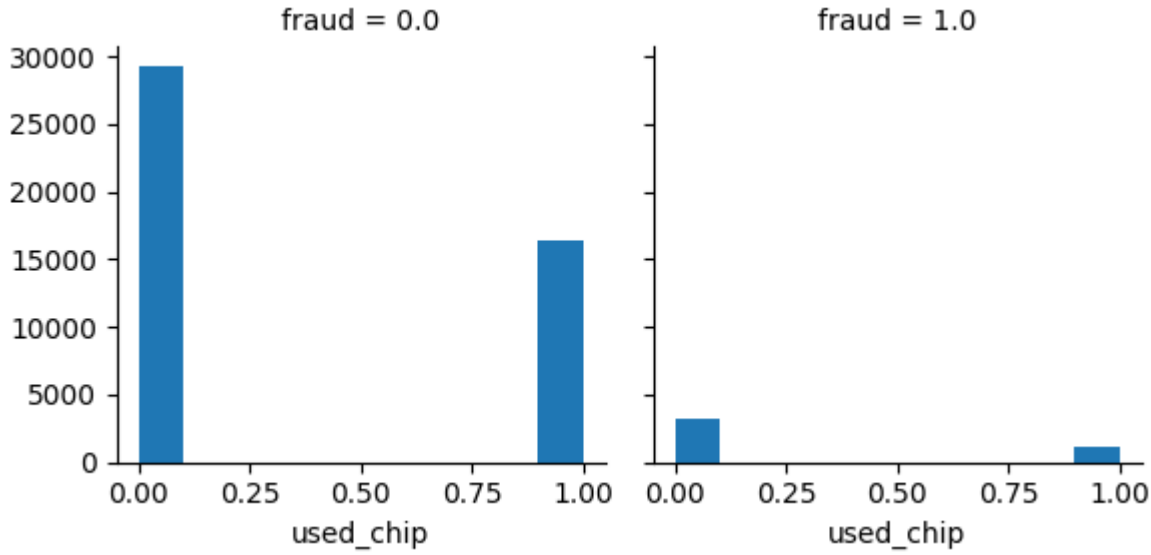
Si hacemos un `describe()` de la ratio agrupando por fraude o no, vemos que en los casos fraudulentos los valores siempre son más altos, en todas las métricas salvo el valor máximo. Esto corrobora lo que se venía apuntando, el gasto es mayor en el caso de las transacciones fraudulentas.

He intentado también comparar los valores mediante gráficos de barras, viendo la diferencia entre un valor fijo y su relación con las transacciones fraudulentas. En este primer caso, vemos que la inmensa mayoría de movimientos por internet no son

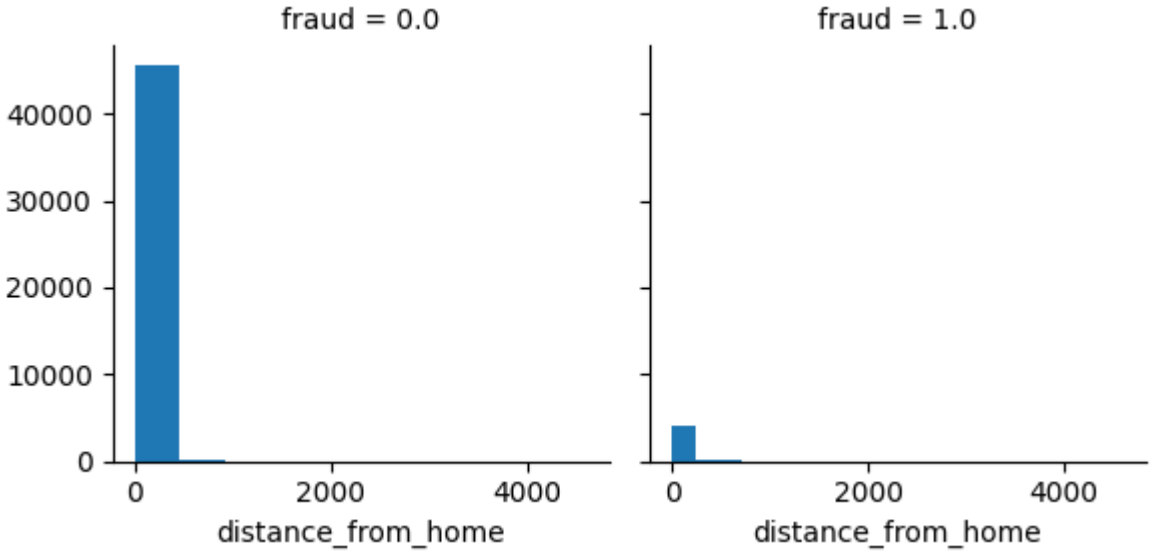
fraudulentos, pero también llama la atención que la mayoría de los que sí que son ilegales también se realizan *online*.



La estadística es similar si nos centramos en el uso o no de número PIN. Como en la casuística anterior, más de 40.000 registros en los que no sé empleó PIN eran legales, pero también hay muchísimos más casos (en proporción) de transacciones fraudulentas en las que tampoco se empleó PIN respecto a las que sí.



El caso del uso del chip está en la misma línea que los casos anteriores, aunque los resultados no son tan evidentes porque hay bastantes registros en los que se empleó chip pero aun así la transacción no era legal. Aunque sea un número bajo, es mayor que en el caso del PIN, lo que indica que el chip no es tan seguro como el método anterior.



Esta gráfica parece desmentir mi primera impresión por lo que respecta a la distancia del lugar de residencia. Vemos cómo la inmensa mayoría de transacciones, tanto legales como no, se realizan a poca distancia.

Así pues, en líneas generales, y mediante el análisis que acabamos de realizar, diría que los factores más relevantes a la hora de catalogar si la transacción es fraudulenta son que se haya hecho por internet, sin PIN y/o sin chip y que la ratio respecto a compras previas se alta.

5. **Escoge un modelo y justifícalo.**

   Antes de poder escoger un modelo en concreto, he querido probar varios modelos de aprendizaje automático para ver cuál era el que obtenía mejor puntuación y poder decidir, entonces, en función de sus métricas. Los modelos que he elegido han sido: Decision Tree, KNN, Random Forest, Logistic Regression, SGD Classifier y Linear SVG. Además, he querido probar también el uso de una red neuronal, aunque en una situación como esta es más adecuado el uso de aprendizaje automático y no deep learning. Las métricas finales para evaluar los modelos varían levemente en función de la ejecución, pero se mantienen las tendencias generales. Todas las capturas de las métricas las hice en la misma ejecución para que los datos no llevaran a equívoco.

   Como he hecho con anterioridad mostraré el código en el primer caso (puesto que he seguido el mismo procedimiento con todos los modelos) y luego solo destacaré las métricas.

```python
# LINEAR SVC
linear_svc = LinearSVC()
linear_svc.fit(X_train, y_train)
y_pred_svc = linear_svc.predict(X_test)
linear_svc_results = pd.DataFrame({
    'Reales': y_test_array,
    'Predichos': y_pred_svc
})
print("\nLINEAR SVC")
print(linear_svc_results.head(20))
print(linear_svc_results.tail(20))

svc_confusion_mat = confusion_matrix(y_test, y_pred_svc)
print("\nMATRIZ DE CONFUSIÓN LINEAR SVC ")
print(svc_confusion_mat)

svc_report = classification_report(y_test, y_pred_svc)
print("\nCLASSIFICATION REPORT LINEAR SVC")
print(svc_report)
print("")

f1_svc = f1_score(y_test, y_pred_svc)
svc_metric = round(f1_svc * 100, 2)
print(f"Métrica final f1: {svc_metric}")
```

En primer lugar he definido el modelo, lo he entrenado con `fit()` y ha realizado las predicciones. He guardado los valores reales y los predichos para mostrar una pequeña porción a modo de comprobación. Posteriormente, para poder evaluar los distintos modelos he empleado una matriz de confusión, un classification report y por último el f1 individualizado, que es el valor que servirá para comparación definitiva entre los modelos elegidos.

LINEAR SVC

	Reales	Predichos
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
5	0.0	0.0
6	0.0	0.0
7	1.0	0.0
8	0.0	0.0
9	0.0	0.0
10	0.0	0.0
11	0.0	0.0
12	0.0	0.0
13	0.0	0.0
14	0.0	0.0
15	1.0	1.0
16	0.0	0.0
17	0.0	0.0
18	0.0	0.0
19	0.0	0.0

En esta captura vemos la comparativa entre valores reales y predichos que acabamos de mencionar. A continuación se mostrarán las métricas más relevantes de los distintos modelos.

MATRIZ DE CONFUSIÓN LINEAR SVC

```
[[9140   36]
 [ 510  314]]
```

CLASSIFICATION REPORT LINEAR SVC

	precision	recall	f1-score	support
0.0	0.95	1.00	0.97	9176
1.0	0.90	0.38	0.53	824
accuracy			0.95	10000
macro avg	0.92	0.69	0.75	10000
weighted avg	0.94	0.95	0.94	10000

Métrica final f1: 53.49

Podemos comprobar que aunque la accuracy es muy elevada, la métrica final no es para nada comprable, puesto que no ha identificado muchos casos que sí que eran fraudulentos.

```
MATRIZ DE CONFUSIÓN SGD CLASSIFIER
[[8973  203]
 [ 365  459]]

CLASSIFICATION REPORT SGD CLASSIFIER
              precision    recall  f1-score   support

         0.0       0.96      0.98      0.97      9176
         1.0       0.69      0.56      0.62       824

    accuracy                           0.94     10000
   macro avg       0.83      0.77      0.79     10000
weighted avg       0.94      0.94      0.94     10000


Métrica final f1: 61.78
```

SGD Classifier, que según algunas fuentes debería adaptarse bien al tipo de proyecto (clasificación, aprendizaje supervisado y menos de 100.000 registros), apenas mejora los resultados del modelo anterior, por lo que tampoco se puede considerar como un modelo fiable. En este caso, los Falsos Positivos son mayores.

```
MATRIZ DE CONFUSIÓN LOGISTIC REGRESSION
[[9126   50]
 [ 360  464]]

CLASSIFICATION REPORT LOGISTIC REGRESSION
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.98      9176
         1.0       0.90      0.56      0.69       824

    accuracy                           0.96     10000
   macro avg       0.93      0.78      0.84     10000
weighted avg       0.96      0.96      0.95     10000


Métrica final f1: 69.36
```

Logistic regression sigue obteniendo una puntuación algo mejor, pero insuficiente, ya que sigue en la línea de los modelos previos.

```
MATRIZ DE CONFUSIÓN KNN
[[8943  233]
 [ 214  610]]


CLASSIFICATION REPORT KNN
              precision    recall  f1-score   support

         0.0       0.98      0.97      0.98      9176
         1.0       0.72      0.74      0.73       824

    accuracy                           0.96     10000
   macro avg       0.85      0.86      0.85     10000
weighted avg       0.96      0.96      0.96     10000


Métrica final f1: 73.19
```

La misma evaluación se podría aplicar también a K-Nearest Neighbors, que arroja resultados similares a Logistic Regression, aunque algo superiores.

```
MATRIZ DE CONFUSIÓN RANDOM FOREST
[[9176    0]
 [   1  823]]

CLASSIFICATION REPORT RANDOM FOREST
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      9176
         1.0       1.00      1.00      1.00       824

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000


Métrica final f1: 99.94
```

Random forest, en cambio, logra unas métricas extremadamente positivas, tanto en el f1 score como en la matriz de confusión.

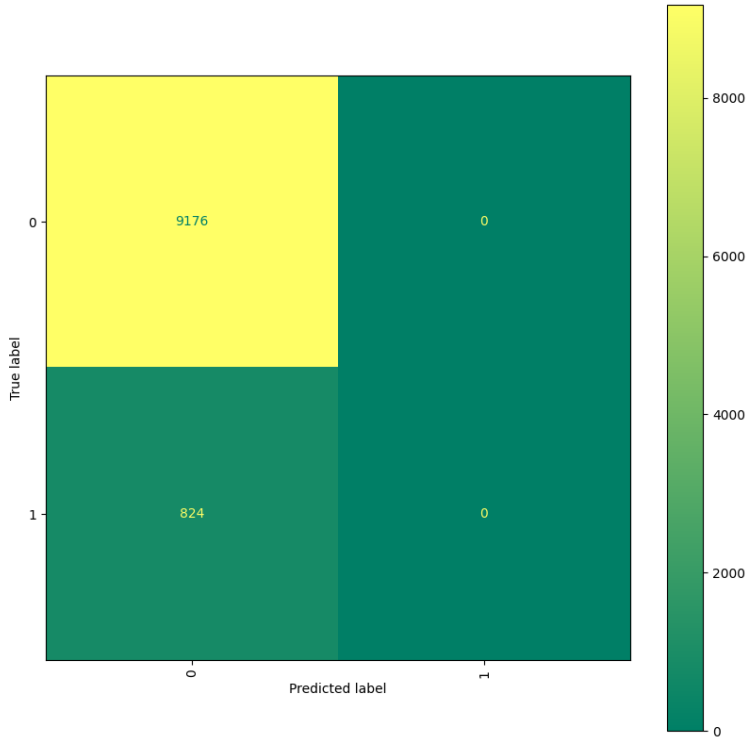
```
MATRIZ DE CONFUSIÓN DECISION TREE
[[9174    2]
 [   3  821]]


CLASSIFICATION REPORT DECISION TREE
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      9176
         1.0       1.00      1.00      1.00       824

    accuracy                           1.00     10000
   macro avg       1.00      1.00      1.00     10000
weighted avg       1.00      1.00      1.00     10000


Métrica final f1: 99.7
```

Para acabar con los modelos de aprendizaje automático, Decision Tree también logra una puntuación altísima, muy parecida a Random Forest. Fijémonos en ambos casos en la matriz de confusión y en la métrica final de la parte inferior. Sin duda, estos dos modelos no tienen nada que ver con los que se han empleado anteriormente; así pues, a juzgar por los resultados, sin duda el mejor modelo para esta problemática sería Random Forest, seguido por muy poco por Decision Tree.

A continuación se muestran los resultados de la red neuronal o neural newtork.

```
val_loss -> 30 min: 0.013029576279222965 max: 0.1740419864654541
acc_hist -> 30 min: 0.9567499756813049 max: 0.9958750009536743
```

A priori los resultados de la NN parecían óptimos, puesto que las pérdidas eran muy bajas y el accuracy muy alto, como podemos ver en la captura superior. Sin embargo, al realizar una matriz de confusión con un mapa de calor se aprecia que en realidad el modelo no es tan preciso.



Sin duda, es excelente por lo que se refiere a los Verdaderos Negativos (9176 casos en que no había fraude en la transacción), pero vemos que el problema está en los Verdaderos Positivos (sí que era fraudulenta), es decir, no ha logrado predecir correctamente ni un solo registro. Los Falsos Negativos dan cuenta de ello (824). Añadir capas Dropout y Batch Normalization no mejoró los resultados.

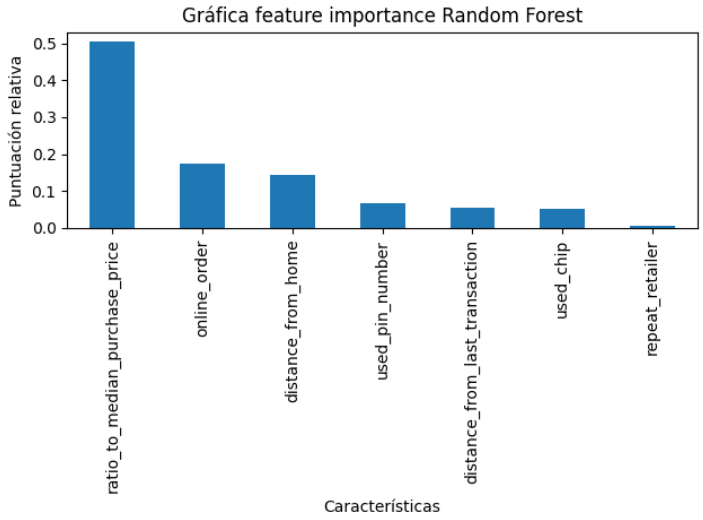
	precision	recall	f1-score	support
0.0	0.92	1.00	0.96	9176
1.0	0.00	0.00	0.00	824
accuracy			0.92	10000
macro avg	0.46	0.50	0.48	10000
weighted avg	0.84	0.92	0.88	10000

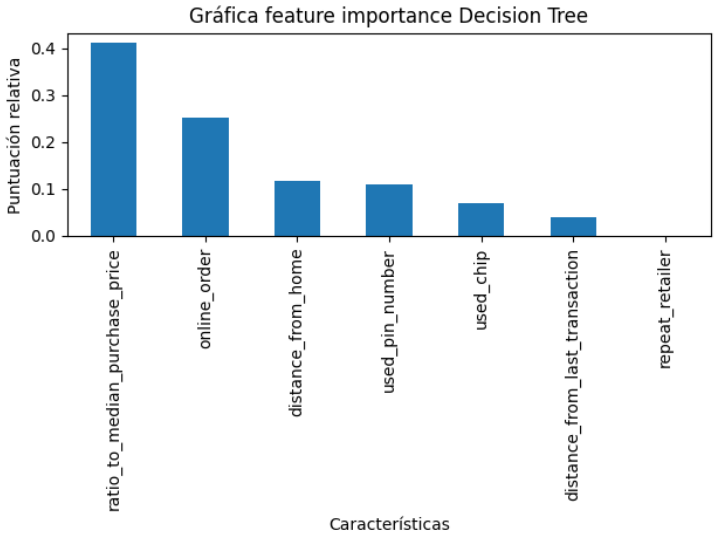
En el classification report confirmamos que, pese a que el accuracy es muy alto, el f1-score para la categoría “fraude” es 0.00.

```
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

He intentado aplicar SMOTE para que tuviera más peso la clase fraude, pero aun así los resultados no han mejorado. Dada la complejidad y el tiempo de computación en relación a los resultados (y de intentar optimizar los resultados mediante ajustes como estos), creo que en esta situación una aproximación mediante modelos de *deep learning* no es la más adecuada.

Para acabar con el análisis de los datos, he querido incluir dos últimas gráficas para los dos modelos que han obtenido mejores resultados, el Random Forest y el Decision Tree.



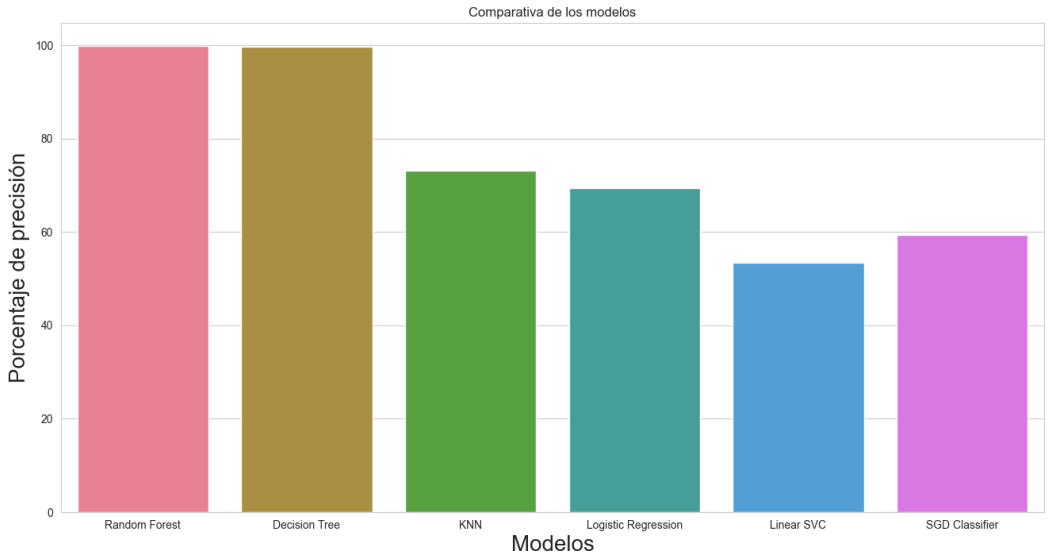


Cuando me documentaba encontré Feature importance dentro de Sci-kit Learn, que nos dice qué peso ha tenido cada característica en la decisión final del modelo. Vemos que en los dos modelos las tres características que han sido más relevantes son la ratio relativa al precio de compra, si la transacción es por internet y la distancia respecto al lugar de residencia. Esto está en sintonía con el mapa de calor de la correlación que hemos tratado anteriormente y con los análisis posteriores, y demuestra que realmente sí se trataba de un valor relevante.

Puede que sea a causa de mi inexperiencia en el mundo del análisis de datos, pero como ya he mencionado, al principio esta no me parecía la categoría más importante para determinar si sería fraude o no; pensaba más bien en si la transacción sería por internet, si se haría lejos del lugar de residencia o si se usaban métodos de seguridad como chip o PIN. Sin embargo he

tenido que readaptar mis sesgos previos, porque las gráficas y los modelos han demostrado que es, seguramente, el factor más determinante. De ahí la importancia del aprendizaje automático, que ha sabido ver patrones y la importancia de categorías a las que yo no habría dado el peso que se merecían.

6. Representación final de los resultados



En esta última gráfica podemos ver claramente como hay dos modelos (Random Forest y Decision Tree) que obtienen resultados excelentes (basados en el f1- score que hemos analizado anteriormente) y que están a mucha distancia del resto. Estos serían, sin duda, los modelos que elegiría para poder llevar a cabo predicciones sobre si una transacción es fraudulenta o no.

CONCLUSIONES

Para mi proyecto final he decidido centrarme en un caso de clasificación con **machine learning**. Como hemos visto en la presentación del proyecto, mi objetivo era generar un **modelo de aprendizaje automático** que pudiera clasificar si las **transacciones** con tarjeta de crédito de los datos eran **fraudulentas o no**. Previamente he intentado realizar un análisis exhaustivo de los datos para comprenderlos, poder interpretarlos mejor y discernir si los modelos rendían como cabía esperar.

Los modelos **Random Forest** y **Decision Tree** han obtenido **métricas f1** que **superan** el **95%**, por lo que creo que la tarea principal del proyecto final ha quedado satisfecha.