Práctica 2: Limpieza y validación de los datos

Andrea González Vicario y Mª Dolores Higuera González

5 de enero de 2021

Contenidos

1. Descripción del dataset....2
2. Integración y selección de los datos....4
3. Limpieza de los datos....5
   3.1. Detección de datos nulos o vacíos....5
   3.2. Identificación y tratamiento de valores extremos....6
4. Análisis de los datos....7
   4.1. Selección de los grupos de datos.....7
   4.2. Comprobación de la normalidad y homogeneidad de la varianza.....10
   4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. 14
5. Resolución del problema de estudio 18
6. Código del análisis en Python....18

1. Descripción del dataset

Esta práctica tiene como objetivo el tratamiento de datos de un dataset con objetivo de aprender a identificar los datos relevantes para un proyecto analítico y usar las herramientas de integración, limpieza, validación y análisis.

Para ello, hemos elegido un dataset procedente de la web Kaggle (https://www.kaggle.com/) llamado "Credit Card customers". Este dataset recoge los datos personales y de actividades bancarias de unos 10.000 clientes de un banco.

La intención de este estudio es conseguir crear un modelo con el que el gerente de banco pueda prever que clientes tienen más posibilidades de abandonar el barco. El conjunto de datos tiene un hándicap que dificulta mucho la obtención de un modelo eficaz, es por eso que el objetivo principal será conseguir los factores que tienen más peso en la pérdida de clientes.

Como datos personales recogidos de los clientes tenemos:

- *CLIENTNUM*: Numero de cliente, identificador único del titular de la cuenta.
- *Customer_Age*: Edad del cliente.
- *Gender*: Sexo del cliente (M-masculino, F-femenino)
- *Dependent_count*: Número de personas dependientes del cliente.
- *Education_Level*: Nivel de educación del titular.
- *Marital_Status*: Estado civil del cliente.
- *Income_Category*: Ingresos anuales del titular de la cuenta.

Como información bancaria encontramos:

- *Attrition_Flag*: Variable que distingue entre los clientes que han dejado el banco y los que siguen siendo clientes.
- *Card_Category*: Tipo de tarjeta del titular.
- *Months_on_book*: Antigüedad del cliente en meses.
- *Total_Relationship_Count*: Número total de productos (cuentas, tarjetas) que ha tenido el cliente.
- *Months_Inactive_12_mon*: Número de meses que el cliente ha pasado inactivo en el último año.
- *Contacts_Count_12_mon*: Número de contactos que ha tenido el cliente durante el último año.
- *Credit_Limit*: Límite de crédito en la tarjeta.
- *Total_Revolving_Bal*: Balance total de la tarjeta de crédito.
- *Avg_Open_To_Buy*: Media total de aperturas de credito en el ultimo año.
- *Total_Amt_Chng_Q4_Q1*: Diferencia total de cantidades en los créditos entre el primer y cuarto cuartil.
- *Total_Trans_Amt*: Total anual de las cantidades en transacciones del cliente.
- *Total_Trans_Ct*: Total de transacciones anuales del cliente.

- *Total_Ct_Chng_Q4_Q1*: Diferencia total del número de créditos entre el primer y cuarto cuartil.
- *Avg_Utilization_Ratio*: Cuantificador de la utilización media de la tarjeta de crédito.

2. Integración y selección de los datos

El conjunto de datos para el análisis está formado por 23 columnas (variables, características de los clientes) y 10.127 filas (entradas, clientes). En la propia descripción del dataset nos recomiendan eliminar las dos últimas columnas puesto que no aportan ningún valor al análisis.

Tras valorar el objetivo de nuestro análisis y los pasos y metodología que vamos a seguir decidimos prescindir también de las siguientes variables:

- *CLIENTUM*: El identificador del cliente es un número con actuación categórica que como mucho nos podría servir para ordenar los clientes por antigüedad. Sin embargo, ya tenemos esa información en la variable *Months_on_book*.
- *Total_Amt_Chng_Q4_Q1*: La diferencia en la cantidad de las transacciones entre el cuatro y primer cuartil no es una información muy relevante para nuestro análisis. Podemos trabajar con la información del total de cantidad anual en transacciones en la variable *Total_Trans_Amt*.
- *Total_Ct_Chng_Q4_Q1*: La diferencia en el número de transacciones entre el cuatro y primer cuartil tampoco sería una información muy relevante para nuestro análisis. Trabajaremos con el total de transacciones anuales en la variable *Total_Trans_Ct*.

Descartamos las variables que no utilizaremos de la siguiente manera:

```
# Primero debemos deshacernos de las columnas que no son necesarias para el estudio del dataset

datos = data.drop(['CLIENTNUM','Total_Amt_Chng_Q4_Q1', 'Total_Ct_Chng_Q4_Q1',
                   'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Educatic
                   'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Educatic
                  ], axis=1)
```

3. Limpieza de los datos

3.1. Detección de datos nulos o vacíos

Se revisará el conjunto de datos para encontrar todos los valores que pudieran perturbar el estudio de los mismos por ser nulos, estar vacíos o no encajar con alguna de las posibilidades de las variables categóricas.

En primer lugar, revisaremos la existencia de valores nulos con un conjunto de funciones que nos devuelve el número de nulos por variable.

```
# Revisamos si hay datos nulos

datos.isnull().sum()
```

```
Attrition_Flag              0
Customer_Age                0
Gender                      0
Dependent_count             0
Education_Level             0
Marital_Status              0
Income_Category             0
Card_Category               0
Months_on_book              0
Total_Relationship_Count    0
Months_Inactive_12_mon      0
Contacts_Count_12_mon       0
Credit_Limit                0
Total_Revolving_Bal         0
Avg_Open_To_Buy             0
Total_Trans_Amt             0
Total_Trans_Ct              0
Avg_Utilization_Ratio       0
dtype: int64
```

También, revisaremos si existen duplicados.

```
# Revisamos si hay duplicados

duplicados = datos.duplicated()
duplicados.sum()
```

```
0
```

Después, basándonos en la información que nos ofrece la página del dataset comprobaremos que todas las variables categóricas tienen su información correctamente categorizada. Es decir, que en la variable género no hubiera F, M, m, f, man, woman… sino tan solo M y F.

```
# Revisamos que no haya datos mal escritos o incorrectos (basándonos en la explicacion del dataset https://www.kaggle.c

for i in datos.columns:
    print(datos[i].name)
    print(len(datos[i].unique()), datos[i].unique())
```

```
Attrition_Flag
2 ['Existing Customer' 'Attrited Customer']
Customer_Age
45 [45 49 51 40 44 32 37 48 42 65 56 35 57 41 61 47 62 54 59 63 53 58 55 66
 50 38 46 52 39 43 64 68 67 60 73 70 36 34 33 26 31 29 30 28 27]
Gender
2 ['M' 'F']
Dependent_count
6 [3 5 4 2 0 1]
Education_Level
7 ['High School' 'Graduate' 'Uneducated' 'Unknown' 'College' 'Post-Graduate'
 'Doctorate']
Marital_Status
4 ['Married' 'Single' 'Unknown' 'Divorced']
Income_Category
6 ['$60K - $80K' 'Less than $40K' '$80K - $120K' '$40K - $60K' '$120K +'
 'Unknown']
Card_Category
4 ['Blue' 'Gold' 'Silver' 'Platinum']
```

Como vemos, no hay valores nulos, duplicados que eliminar ni información mal categorizada. Tenemos un conjunto de datos con muy buena calidad para seguir trabajando con él.

3.2. Identificación y tratamiento de valores extremos

Por otro lado, debemos hacer una revisión de los valores extremos u outliers, que son aquellos valores que no parecen seguir la tónica normal del resto de datos del conjunto.

Para identificarlos hemos decido utilizar el método de z-score. Este método estadístico selecciona las filas de datos numéricos cuyos valores se consideran normales. Se considera que todos los datos que se alejen de la media en más de tres unidades pueden clasificarse como outliers.

```
# Descartamos los outilers con el método del z-score
# Dividimos las columnas del dataset en numéricas y categóricas

num_datos = datos.select_dtypes(include=["number"])
cat_datos = datos.select_dtypes(exclude=["number"])

outliers_row = np.all(stats.zscore(num_datos) < 3, axis=1)

datos_cleaned = pd.concat([num_datos.loc[outliers_row], cat_datos.loc[outliers_row]], axis=1)
```

Tras esta revisión y limpieza nos quedamos con un conjunto de datos de 9.560 entradas.

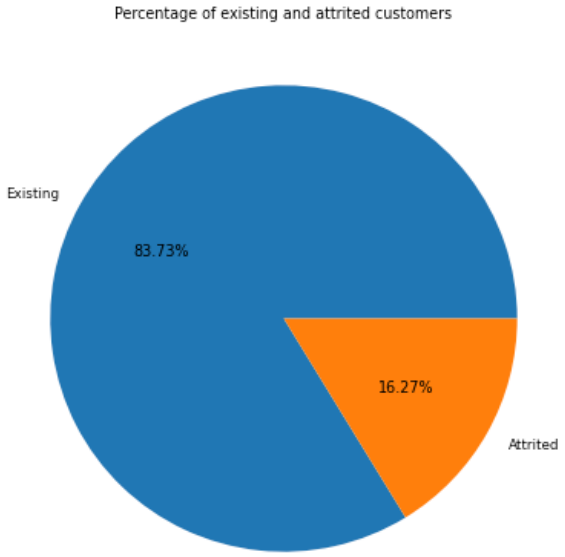
4. Análisis de los datos

4.1. Selección de los grupos de datos.

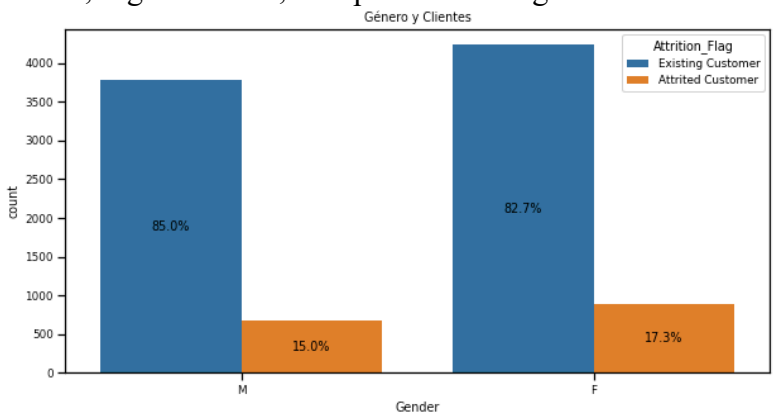
Hemos agrupado por sexo, por cliente y por tipo de tarjeta, para poder analizarlos posteriormente.

```
_____________Agrupación por sexo_____________
Hombres:  4443
Mujeres:  5117

_____________Agrupación por cliente_____________
Clientes existentes:  8005
Clientes que se han ido:  1555

_____________Agrupación por tipo de tarjeta _____________
Tarjeta azul:  8966
Tarjeta oro:  96
Tarjeta plata:  481
Tarjeta platino:  17
```

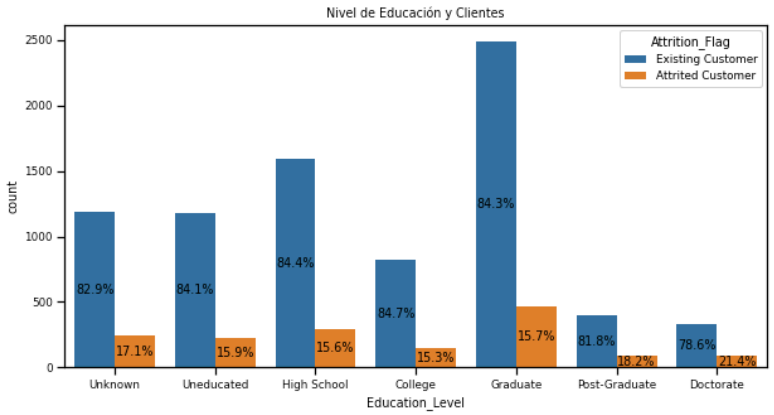
Además, hemos ido graficando las variables, para ver que cantidad de datos tenemos. En primer lugar, sabemos que la mayoría de los usuarios de nuestra base de datos, se quedan en la misma entidad (83.73%), mientras que el 16.27% se ha ido.



Estos clientes, según su sexo, se reparten de la siguiente manera:

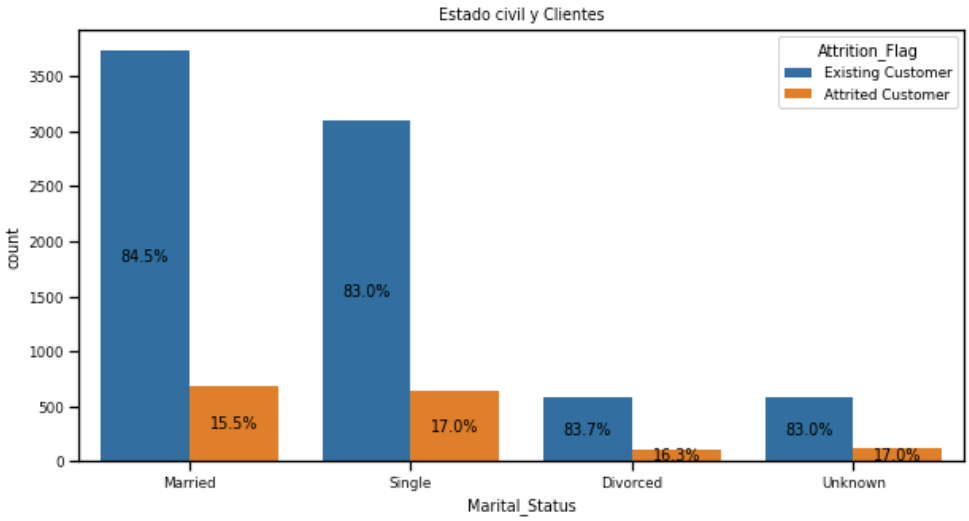


Comprobamos que existe un mayor número de mujeres en nuestra base de datos, que de hombres (5117 mujeres con respecto a 4443 hombres). En porcentajes, existe una mayor cantidad de hombres que se quedan en la entidad, con respecto a los que se van, según podemos apreciar en el gráfico anterior, siendo un 85% de clientes masculinos los que deciden continuar en la entidad frente al 82.7% en cuanto a mujeres se refiere.
En el siguiente gráfico encontramos el nivel de educación de los clientes. La mayoría de ellos (un 70.4% del total), tiene algún tipo de estudios (secundaria, universidad, graduados, postgraduados, doctorados...), pero entorno al 14.6%, no los tienen. Existe una pequeña parte de la población (el 15%) del cual no conocemos si tiene estudios o si no los tiene.

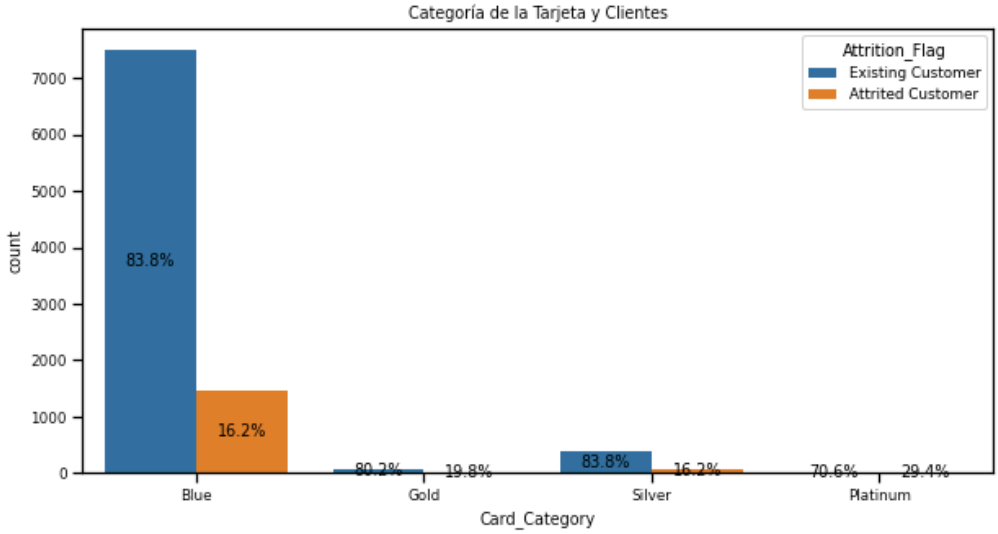



```
El número total de personas sin estudios es: 1400, lo que representa un 14.6 % de los clientes
El número total de personas con algún estudio a partir de secundaria es: 6730, lo que representa un 70.4 % de los clientes
Existen 1430 personas que no sabemos su nivel de estudios, lo que representa un 15.0 % de los clientes
```

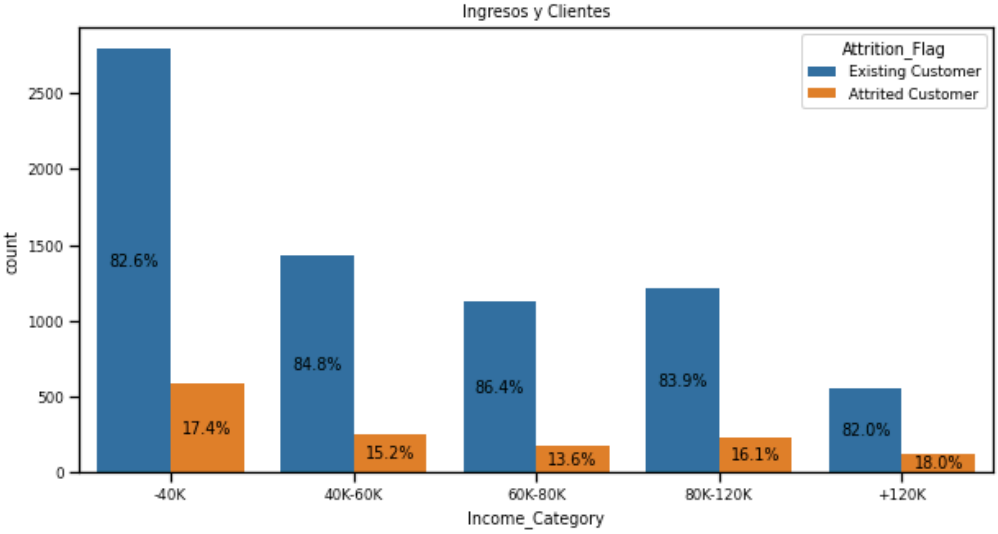
En la siguiente figura, podemos apreciar que la mayoría de los usuarios están casados, de los cuales el 84.5% se queda en la entidad.



Entre los gráficos mostrados, también podemos comprobar que la gran mayoría de los usuarios tiene la tarjeta Blue.



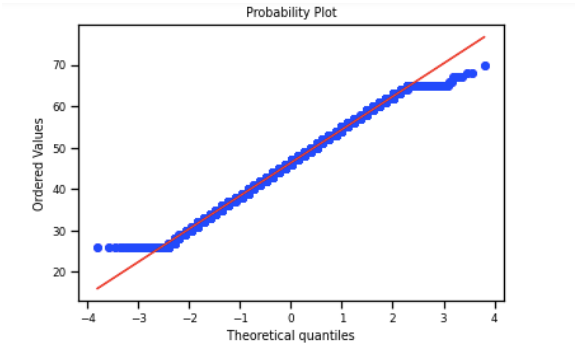
Por último, en este análisis hemos comprobado que la mayoría de nuestros usuarios, tienen ingresos menos a 40K$.



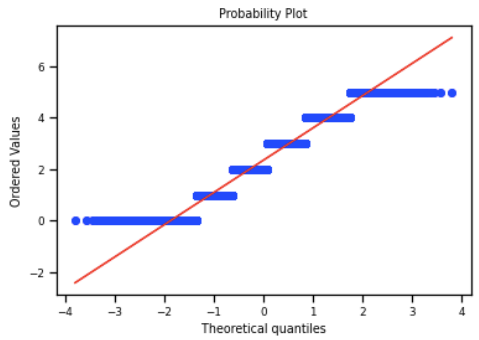
4.2. Comprobación de la normalidad y homogeneidad de la varianza.

En primer lugar, comprobaremos la normalidad de nuestras variables numéricas. Las someteremos a un test gráfico y a otro estadístico.

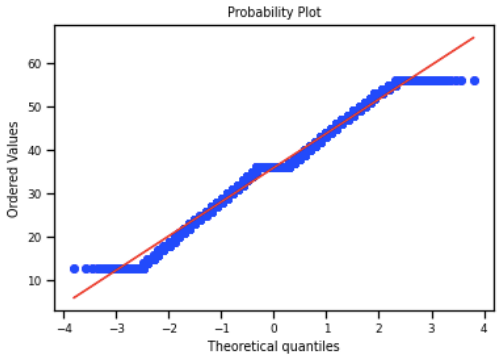
Tras seleccionar las variables numéricas hemos mostrado por pantalla los resultados de un quantile-quantile plot y un test Shapiro de cada una de ellas.



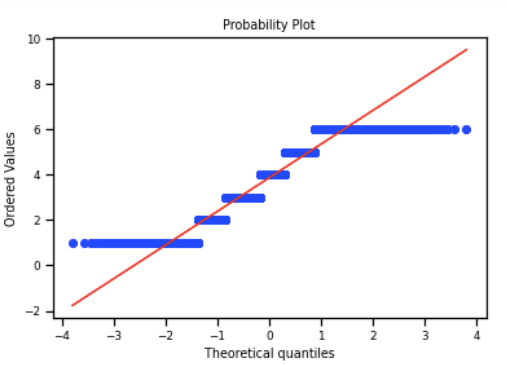

```
Customer_Age
El p-valor es: 7.418087970739772e-16
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```



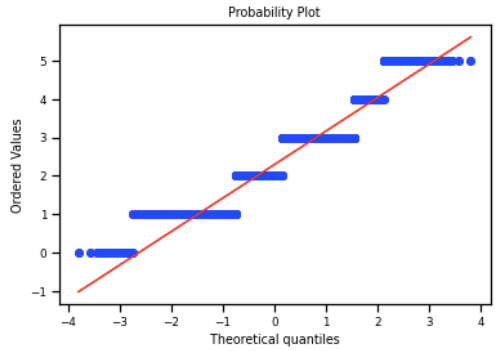

```
Dependent_count
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```



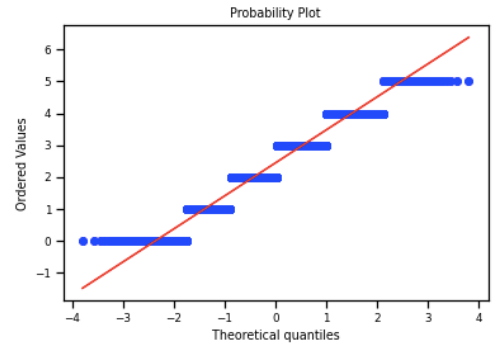

```
Months_on_book
El p-valor es: 1.738110168737946e-36
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```



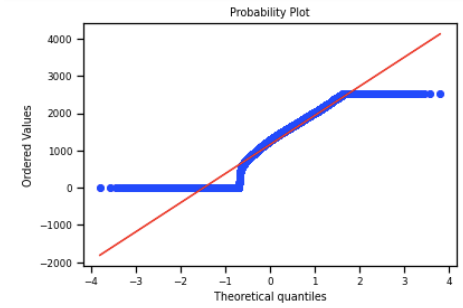
Total_Relationship_Count
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.



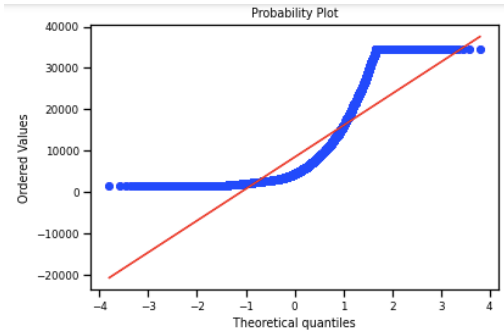
Months_Inactive_12_mon
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.



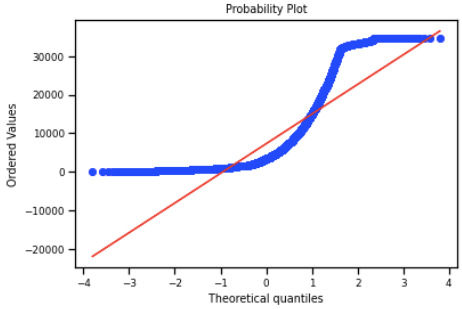
Contacts_Count_12_mon
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.



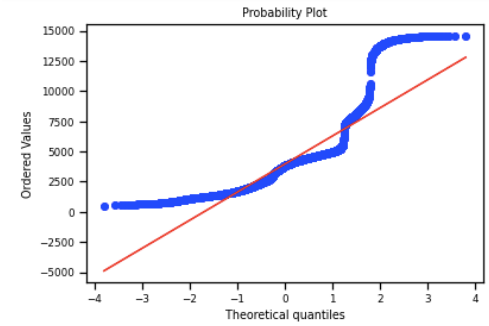
Total_Revolving_Bal
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.



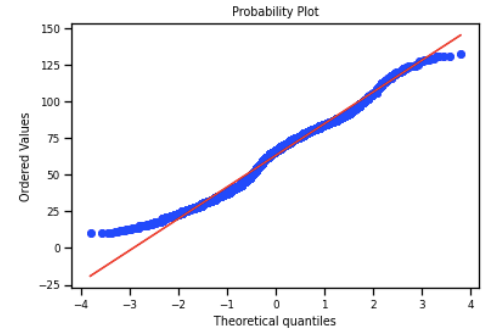
Credit_Limit
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.



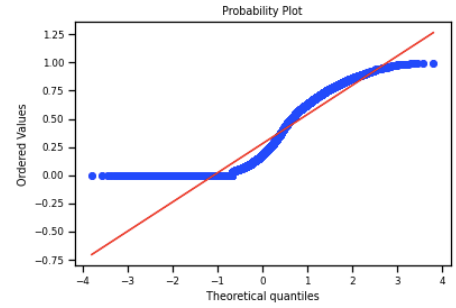

```
Avg_Open_To_Buy
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```




```
Total_Trans_Amt
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```




```
El p-valor es: 1.9122488431825953e-33
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```




```
Avg_Utilization_Ratio
El p-valor es: 0.0
Rechazamos la hipótesis nula.
La distribución de nuestra variable no es normal o Gaussiana.
```

Además, hemos realizado una prueba de la homogeneidad de las varianzas aplicando el test Levene, el cual está recomendado para conjuntos de datos no normales (como hemos visto que es nuestro caso).
Los resultados son los siguientes:

```
Customer_Age  W: 7.090523985983596  P-val: 0.007762239436544139
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Dependent_count  W: 0.10722623655708224  P-val: 0.7433315688154655
No podemos rechazar la hipótesis nula.
La variables son homogéneas.

Months_on_book  W: 5.003988291856731  P-val: 0.025311944865797714
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Total_Relationship_Count  W: 2.956105235329942  P-val: 0.08558616949705176
No podemos rechazar la hipótesis nula.
La variables son homogéneas.

Months_Inactive_12_mon  W: 78.02195950920128  P-val: 1.1987012349556468e-18
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Contacts_Count_12_mon  W: 88.28400506036  P-val: 6.97532880711331e-21
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Credit_Limit  W: 0.34844532879392026  P-val: 0.5550084691665839
No podemos rechazar la hipótesis nula.
La variables son homogéneas.

Total_Revolving_Bal  W: 16.061079491067808  P-val: 6.179725041090802e-05
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Avg_Open_To_Buy  W: 0.35444106346266613  P-val: 0.5516239492135611
No podemos rechazar la hipótesis nula.
La variables son homogéneas.

Total_Trans_Amt  W: 26.448763079306605  P-val: 2.75985161241335e-07
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Total_Trans_Ct  W: 288.71949976902397  P-val: 8.133616400367962e-64
Rechazamos la hipótesis nula.
La variables no son homogéneas.

Avg_Utilization_Ratio  W: 186.74456991009288  P-val: 4.0569737597620686e-42
Rechazamos la hipótesis nula.
La variables no son homogéneas.
```

Tras este estudio podemos ver que nuestras muestras no son normales, pero sin embargo, alguna de sus varianzas si son homogéneas entre las dos poblaciones (clientes existentes y perdidos).

Sin embargo, que las variables no sean normales no quiere decir que no sean normalizables. Según el teorema del limite central al tener mas de 30 elementos en las observaciones podemos aproximarla como una distribución normal de media 0 y desviación standard 1.

4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos.

Para comparar los grupos de datos, hemos aplicado un estudio de **correlación.**

Con este estudio hemos podido ver las variables numéricas que tienen alta relación entre sí. Como era lógico y de esperar, la edad del cliente y el tiempo que ha sido cliente están altamente correlacionadas.

Además, la cantidad de veces que hace transferencia y la cantidad de las mismas también lo está.

Estás variables podrían ser eliminadas ya que ofrecen información redundante.

También hemos realizado un estudio de **regresión logística**.

Para poder realizar este proceso, necesitamos que todas nuestras variables sean numéricas. Por ello, antes debemos codificar las variables categóricas. Este proceso lo hemos realizado con “Pandas” (get_dummies):

```
data_num = pd.get_dummies(datos_cleaned, columns=['Attrition_Flag', 'Gender', 'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category'],drop_first=True)
```

Borramos la variable que queremos clasificar (Attrition_Flag_Existing Customer), y después aplicamos la regresión logística:

```
from sklearn.linear_model import LogisticRegression
# Vamos a crear el modelo, haciendo fit al conjunto de datos de entrada X y de salida Y
model = LogisticRegression(C=1000,max_iter=50000)
model.fit(X,y)
```

```
LogisticRegression(C=1000, max_iter=50000)
```

```
#Clasificamos el conjuunto de entradas X, con el metodo predict.
predictions = model.predict(X)
print(predictions)
```

```
[1 1 1 ... 1 1 1]
```

```
print("Precision media de las predicciones: ")
model.score(X,y)
```

```
Precision media de las predicciones:


0.8746861924686192
```

Confirmamos que nuestro modelo es bastante bueno, teniendo una precisión del 87.46%. Para poder validar el modelo, vamos a subdividir el conjunto de datos de entrada en un set de entrenamiento, y otro de validación (test). La subdivisión elegida es 80% para entrenamiento y 20% para test.

```
from sklearn import model_selection
validation_size=0.20
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, y, test_size=validation_size)
```

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
kf = KFold(n_splits=5)
model_LR = LogisticRegression(C=1000,max_iter=50000)
model_LR.fit(X_train, Y_train)
print("Metrica del modelo", clf.score(X_train,Y_train))


model_LR_cv_tr=cross_val_score(model_LR,X_train,Y_train, cv=kf, scoring="accuracy")
print("Cross_validation train:")
print(model_LR_cv_tr)
print("\n")
print("Media Cross-Validation", model_LR_cv_tr.mean())

```

```
Metrica del modelo 0.8824529288702929
Cross_validation train:
[0.86470588 0.86601307 0.89215686 0.87900589 0.8705036 ]


Media Cross-Validation 0.8744770600631794
```

Con los datos de entrenamiento, nos da una precisión del 87.4% (80% de los datos). Vamos a calcularlo con el 20% de los datos de test

```
from sklearn.metrics import accuracy_score
predictions = model_LR.predict(X_test)
print("Precisión del conjunto de test")
print(accuracy_score(Y_test, predictions))
```

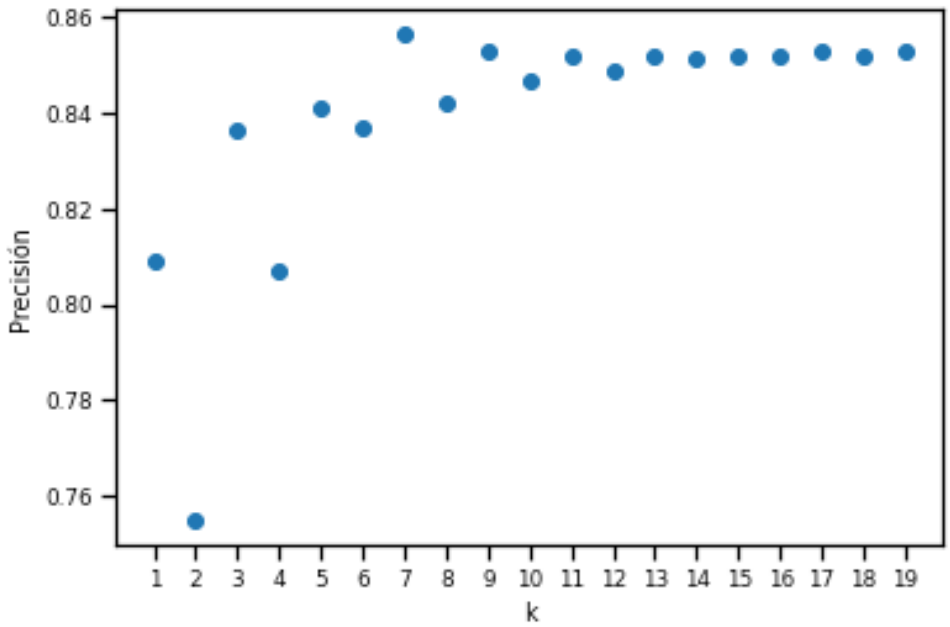
```
Precisión del conjunto de test
0.8645397489539749
```

En este caso, la muestra es menor, y la precisión es algo menor también, aunque sigue siendo una precisión del modelo bastante alta (86.4%).
Para ver cuantos resultados equivocados ha predicho de cada clase, realizamos la matriz de confusión. Todos los valores que no se encuentran en la diagonal, son valores confundidos, es decir, hay 207 valores que los predijo como Existing (valor 1), y eran Attrited (valor 0), y por el contrario 53 que eran Existing, los predijo como Attrited.

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_test, predictions))
```

```
[[ 107  207]
 [  53 1545]]
```

Vamos a utilizar un segundo método de predicción para poder comparar los resultados: KNeighborsClassifier()
En este caso, necesitamos un valor para "k", por lo tanto, vamos a graficar todos los valores posibles del 1 al 20, para ver cual es el valor que mejor se adapta a nuestro objetivo:



Comprobamos que el que mayor predicción tiene es k=7:

```
n_neighbors=7
knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, Y_train)
print('Precisión del modelo para el conjunto de entrenamiento: {:.2f}'.format(knn.score(X_train, Y_train)))
print('Precisión del modelo para el conjunto de test: {:.2f}'.format(knn.score(X_test, Y_test)))
```

```
Precisión del modelo para el conjunto de entrenamiento: 0.87
Precisión del modelo para el conjunto de test: 0.86
```

La precisión obtenida para este método es del 87% en el conjunto de datos de entrenamiento y del 86% para el caso de los datos de test.
Calculamos la precisión del modelo con la matriz de confusión sobre el conjunto de test.

```
pred = knn.predict(X_test)
print(confusion_matrix(Y_test, pred))
```

```
[[  91  228]
 [  46 1547]]
```

```
print(accuracy_score(Y_test, pred))
```

```
0.856694560669456
```

En este caso, la muestra es menor, y la precisión es algo menor también, aunque sigue siendo una precisión del modelo bastante alta (85.6%). Para ver cuantos resultados equivocados ha predicho de cada clase, realizamos la matriz de confusión. Todos los valores que no se encuentran en la diagonal, son valores confundidos, es decir, hay 228 valores que los predijo como Existing, y eran Attrited, y por el contrario 46 que eran Existing, los predijo como Attrited.

5. Resolución del problema de estudio

Tras el estudio de este conjunto de datos hemos llegado a la conclusión de que gracias a la calidad y cantidad de la información proporcionada podemos crear un modelo que determine el futuro del cliente.
Esta herramienta sería de gran ayuda en un caso real, en el que fuéramos acumulando datos con el paso del tiempo. De esta manera podríamos crear un perfil más definido del cliente con altas posibilidades de abandonar el banco con objetivo de impactarle con ofertas que fidelizaran su relación.

Los estudios que hemos hecho tienen una alta fiabilidad y las matrices de confusión mostraban buenos resultados.

6. Código del análisis en Python

El código completo correspondiente a las partes que hemos ido mostrando en este informe está disponible en el siguiente repositorio de GitHub:
https://github.com/andreagonvic/Data-cleansing.git

En él podéis encontrar un archivo .ipynb Jupyter Notebook, este mismo informe en .pdf y el conjunto de datos brutos en un archivo .csv .

Contribuciones	Firmas
Investigación previa	AGV & MHG
Redacción de las respuestas	AGV & MHG
Desarrollo código	AGV & MHG