Consumer Mit Lab Evidencia 3 Equipo2

En la elaboración de este código para pasar los datos de un archivo .csv sin datos nulos ni outliers por regresiones lineales y logísticas como primer paso tenemos que importar las librerías que necesitaremos para darle la estructura necesaria al código para que tenga el comportamiento que deseamos que tenga:

```
#Importamos las librerias pandas, numpy y matplotlib respectivamente
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as special
from scipy.optimize import curve_fit
import seaborn as sns
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Cargamos nuestro archivo al código para empezar a trabajar sobre el

```
#Cargar archivo csv desde equipo
from google.colab import files
files.upload()
```

En este caso estaremos trabajando con el archivo .csv que se realizó gracias a la ayuda del código pasado, ya que este no contiene datos nulos ni outliers lo que nos permite someterlo a un procesos de regresiones para así poder determinar cuál es su matriz de función, la predicción que se realiza sobre este, la matriz de confusión, la precisión que tiene el modelo, la exactitud del modelo y la sensibilidad del modelo, así que para poder determinar todos estos parámetros es necesario realizar una regresión logística para la cual:

Cargamos nuestro archivo libre de nulos y outliers Datos_limpios_Consumer_Mit_Lab_Evidencia

```
#Cargar archivo csv desde equipo
from google.colab import files
files.upload()
```

```
#Carga desde un archivo .csv sin indice
df= pd.read_csv('Datos_limpios_Consumer_Mit_Lab_Evidencia.csv')
df.head(5)
```

Y rellenamos los nulos que pueda llegar a haber dentro del Data Frame

```
#Rellenamos nulos dentro del código
df=df.fillna(method='ffill')
```

Declaramos nuestras variables dependientes e independientes para empezar con la regresión

```
#Declaramos las variables dependientes e independientes para la regresión logística.
Vars_Indep=                          df[['238_frequency_buying_store', '81_number_online_purchases_month', '288_number_app_purchases_month']]
Var_Dep= df['308_topups_in_store']
```

```
#Redefinimos las variables
X= Vars_Indep
y= Var_Dep
```

```
#Dividimos el conjunto de datos en la parte de entrenamiento y pruba:
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=None)
```

```
#Se escalan todos los datos
escalar= StandardScaler()
```

Después de realizar el escalamiento dentro de las variables de x definimos nuestro algoritmo para después entrenarlo, y de cierta forma enseñarle que es lo que debe hacer y cómo debe comportarse

```
#Para realizar el escalamiento de las variables "X" tanto de entrenamiento como de prueba
X_train=escalar.fit_transform(X_train)
X_test= escalar.transform(X_test)
```

```
#Definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo=LogisticRegression()
```

```
#Entrenamos el modelo
algoritmo.fit(X_train, y_train)
```

Después de haber entrenado a nuestro algoritmo podremos empezar a buscar lo que queremos obtener de este algoritmo que en esta caso realizaremos una predicción para ver el comportamiento del algoritmo:

```
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred
```

]
#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

```
array(['no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no',
       'no', 'no', 'no'], dtype=object)
```

Verificamos la matriz de confusión que se genera por parte del algoritmo ya entrenado

```
#Verifico matriz de confusión
from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de Confusión')
print(matriz)
```

Calculamos la precisión del modelo en vase al algoritmo que ya entrenamos

```
#Calculo la precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred, average='binary',
pos_label="yes")
print('Precision del modelo:')
print(precision)
```

Calculamos la exactitud del modelo en base al algoritmo que ya entrenamos

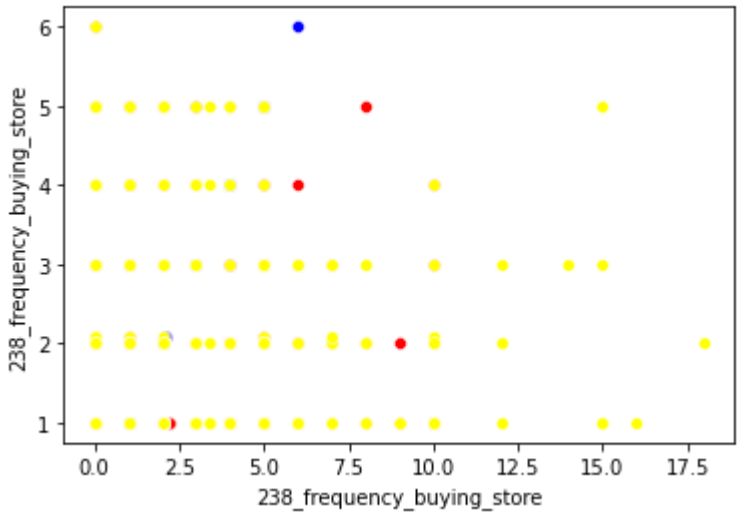
```
#Calculo la exactitud del modelo
from sklearn.metrics import accuracy_score
exactitud = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:')
print(exactitud)
```

Calculamos la sensibilidad que tiene el modelo

```
#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred, average="binary",
pos_label="yes")
print('Sensibilidad del modelo:')
print(sensibilidad)
```

Y con eso tenemos determinada una regresión logística y de la misma forma como se realizó una regresión logística podemos realizar una regresión lineal para este se toma 3 variables las cuales se graficaron para analizar su comportamiento

```
#Se grafican mapas de dispersion de las 3 variables
from turtle import color

sns.scatterplot(x='238_frequency_buying_store',
y='238_frequency_buying_store', color='blue', data=df)
sns.scatterplot(x='81_number_online_purchases_month',
y='238_frequency_buying_store', color='red', data=df)
sns.scatterplot(x='288_number_app_purchases_month',
y='238_frequency_buying_store', color='yellow', data=df)
```




```
#Declaramos las variables dependientes e independientes para la
regresión lineal.
Vars_Indep= df[['288_number_app_purchases_month',
'81_number_online_purchases_month', '238_frequency_buying_store']]
Var_Dep= df['238_frequency_buying_store']
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model= LinearRegression()
```

```
#Verificamos la funcion relacionada al modelo
type(model)
```

Este paso, nos comprobará si estamos trabajando bajo el modelo de regresión lineal que es el que estamos buscando, para poder determinar lo que queremos de este tipo de regresión en base al archivo sin nulos

```
#Ajustamos el modelo con las variables antes declaradas
model.fit(X=Vars_Indep, y=Var_Dep)
```

```
#Verificamos los coeficientes obtenidos para el modelo ajustado
model.__dict__
```

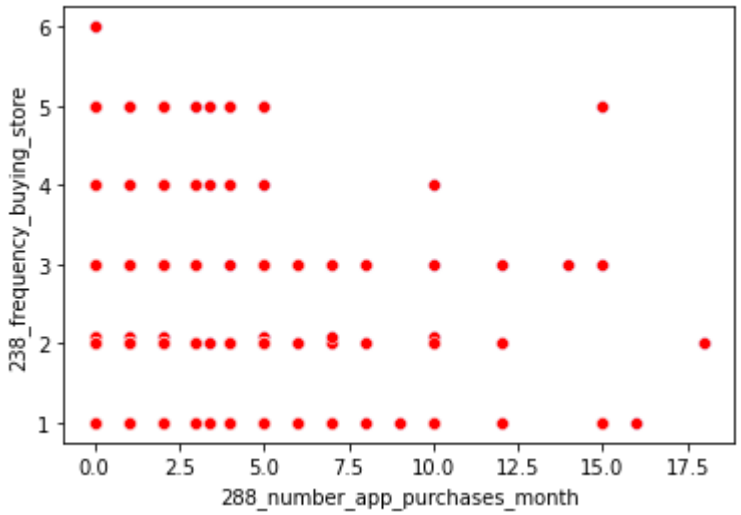
```
#Evaluamos la eficiencia del modelo obtenido por medio del coeficiente R^2 Determinación
model.score(Vars_Indep,Var_Dep)
```

```
#Predecimos los valores de total en base a las variables que tenemos para el DF
y_pred=           model.predict(X=df[['288_number_app_purchases_month', '81_number_online_purchases_month', '238_frequency_buying_store']])
y_pred
```

```
#Insertamos la columna de predicciones en el Dataframe
df.insert(0, 'Predicciones', y_pred)
df
```

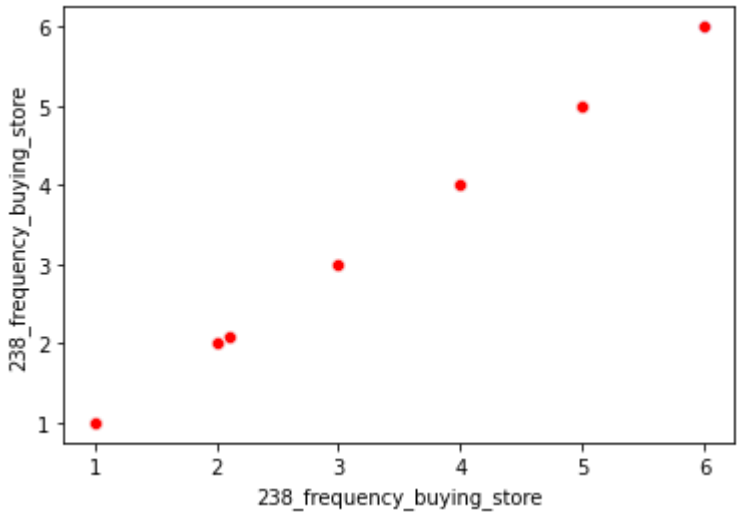
Aquí juntamos nuestros valores que predecimos que serían parte del comportamiento de DF en el mismo lo que nos da una visualización de una gráfica comparativa entre el total real y el predecido para cada una de las variables del DF .

```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x=          '288_number_app_purchases_month',          y= '238_frequency_buying_store', color= 'blue', data=df)
sns.scatterplot(x= '288_number_app_purchases_month', y= 'Predicciones', color= 'red', data=df)
```

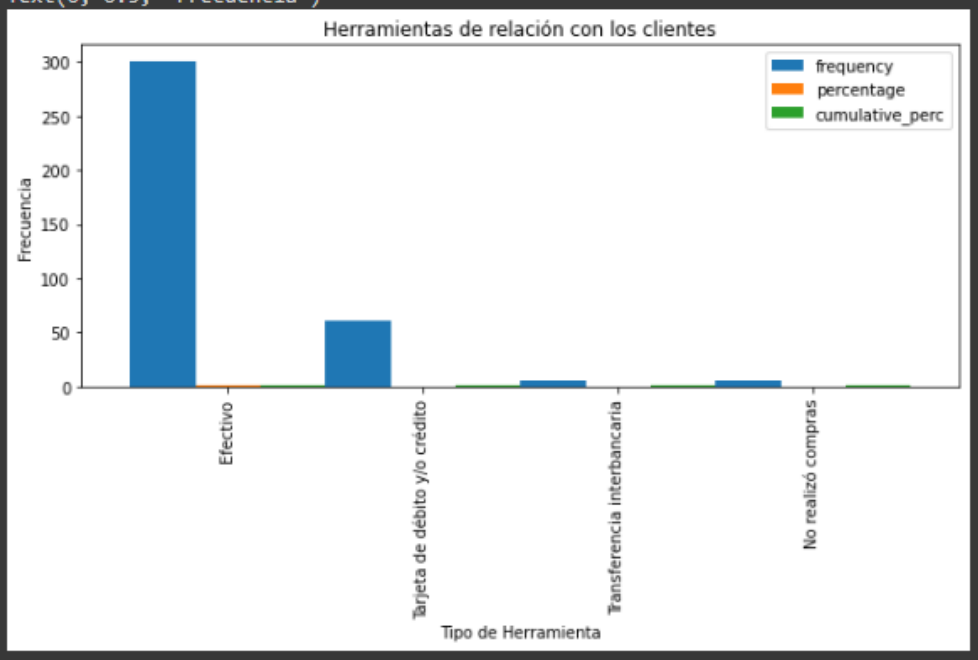



```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x= '81_number_online_purchases_month', y= '238_frequency_buying_store', color= 'blue', data=df)
sns.scatterplot(x= '81_number_online_purchases_month', y= 'Predicciones', color= 'red', data=df)
```


```
#Visualizamos la gráfica comparativa entre el total real y el total predecido

sns.scatterplot(x= '238_frequency_buying_store', y= '238_frequency_buying_store', color= 'blue', data=df)
sns.scatterplot(x= '238_frequency_buying_store', y= 'Predicciones', color= 'red', data=df)
```


238_frequency_buying_store
238_frequency_buying_store
1
2
3
4
5
6

```
Text(0, 0.5, 'Frecuencia')
```




```
[ ]  #Gráfico de pastel
     Filtro_index["frequency"].plot(kind='pie', figsize=(10,5), shadow=True, autopct="%0.1f %%")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe96a4a6a00>
```