Micro Retailer Mit Lab Evidencia 3

EQUIPO 2

Al igual que con el código de la evidencia 2, como primer paso importamos todas las librerías que se utilizaran para poder realizar el código.

```
#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
```

Luego cargaremos el archivo de Datos_limpios_Micro_retailer_evidencia2 que anteriormente obtuvimos. Este archivo es el csv sin nulos ni outliers.

```
#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_Micro_retailer_evidencia2.csv')
```

Regresión Logística 1

Para la primera regresión escogimos una regresión logística en donde nuestras variables independientes fueron: *'97_number_of_customers_in_store', '2_current_permanent_employees', '315_frequency_topups'* y la variable dependiente fue: '*56_procurement_changes_pandemic*'

```
[5]  #Declaramos las variables dependientes e independientes para la regresión logística.
     Vars_Indep= df[['97_number_of_customers_in_store', '2_current_permanent_employees', '315_frequency_topups']]
     Var_Dep= df['56_procurement_changes_pandemic']
```

```
[188] #Redefinimos las variables
      X= Vars_Indep
      y= Var_Dep
```

Entrenamos y probamos todos los datos y se realiza una predicción del modelo

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

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

```
[202] #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)
```

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

Predicción:

```
array([1., 2., 1., 1., 1., 1., 2., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2.,
       1., 1., 1., 1., 1., 2., 1., 1., 1., 1., 1., 1., 1., 1., 2., 1., 2.,
       1., 1., 1., 1., 1., 1., 1., 1., 2., 1., 1., 1., 1., 1., 1., 1., 2.,
       1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 1., 1., 1., 1., 2., 1.,
       2., 1., 2., 1., 1., 2., 1., 1., 1., 1., 2., 2., 1., 2., 1., 2., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 1., 1., 1.,
       1., 1., 1., 1., 1., 2., 1., 1., 1., 1., 1., 1., 2., 1., 1., 1., 1.,
       1., 3., 1., 1., 1., 1., 1., 1., 2., 1., 1., 1., 1., 2., 1.])
```

Generamos una matriz de confusión por parte del algoritmo ya entrenado

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

```
Matriz de Confusión
[[40  9  1  0  0  0  0]
 [28  4  0  0  0  0  0]
 [ 9  2  0  0  0  0  0]
 [ 3  2  0  0  0  0  0]
 [ 4  1  0  0  0  0  0]
 [ 0  2  0  0  0  0  0]
 [25  4  0  0  0  0  0]]
```

Por último, calculamos la precisión, exactitud y sensibilidad del modelo

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

      precision = precision_score(y_test, y_pred, average='micro', pos_label="no")
      print('Precision del modelo:')
      print(precision)
```

```
Precision del modelo:
0.34328358208955223
```

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

```
Exactitud del modelo:
0.34328358208955223
```

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

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

```
Sensibilidad del modelo:
0.34328358208955223
```

Esto lo aplicamos otras 3 veces, pero con diferentes variables dependientes e independientes.

Regresión Logística 2

```
208] #Declaramos las variables dependientes e independientes para la regresión logística.
     Vars_Indep= df[['145_number_direct_competitors', '317_home_deliveries', '24_burnout']]
     Var_Dep= df['310_burnout']
```

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

```
210] #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)
```

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

```
     #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)
```

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

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

Predicción:

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

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

Matriz de confusión:

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

```
Matriz de Confusión
[[65  9]
 [50 10]]
```

Precisión, exactitud y sensibilidad:

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

      precision = precision_score(y_test, y_pred, average='binary', pos_label="no")
      print('Precision del modelo:')
      print(precision)
```

```
Precision del modelo:
0.5652173913043478
```

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

```
Exactitud del modelo:
0.5597014925373134
```

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

sensibilidad = recall_score(y_test, y_pred, average="binary", pos_label="no")
print('Sensibilidad del modelo:')
print(sensibilidad)
```

```
Sensibilidad del modelo:
0.8783783783783784
```

Regresión Logística 3

```
[102] #Declaramos las variables dependientes e independientes para la regresión logística.
      Vars_Indep= df[['2_current_permanent_employees', '172_supplier_frequency', '272_card_days_receive_money']]
      Var_Dep= df['20_reviews_finances_monthly']
```

```
[103] #Redefinimos las variables
      X= Vars_Indep
      y= Var_Dep
```

```
[122] #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)
```

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

```
[106] #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)
```

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

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

Predicción:

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

```
array(['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes'],
      dtype=object)
```

Matriz de confusión:

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

```
Matriz de Confusión
[[ 0 49]
 [ 0 85]]
```

Precisión, exactitud y sensibilidad:

```
[111] #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)
```

```
Precision del modelo:
0.6343283582089553
```

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

```
Exactitud del modelo:
0.6343283582089553
```

```
#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)
```

```
Sensibilidad del modelo:
1.0
```

Regresión Logística 4

[116]
```
#Declaramos las variables dependientes e independientes para la regresión logística.
Vars_Indep= df[['97_number_of_customers_in_store', '145_number_direct_competitors', '163_number_high_perishable_products']]
Var_Dep= df['99_does_the_micro_retailer_exhibits_products_outside_']
```

[117]
```
#Redefinimos las variables
X= Vars_Indep
y= Var_Dep
```

[118]
```
#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()
```

[120]
```
#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)
```

[121]
```
#Definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo=LogisticRegression()
```

[114]
```
#Entrenamos el modelo
algoritmo.fit(X_train, y_train)
```

Predicción:

```
#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', 'yes', '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)
```

Matriz de confusión:

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

```
Matriz de Confusión
[[98  1]
 [35  0]]
```

Precisión, exactitud y sensibilidad:

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

     precision = precision_score(y_test, y_pred, average='binary', pos_label="no")
     print('Precision del modelo:')
     print(precision)
```

```
Precision del modelo:
0.7368421052631579
```

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

```
Exactitud del modelo:
0.7313432835820896
```

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

      sensibilidad = recall_score(y_test, y_pred, average="binary", pos_label="no")
      print('Sensibilidad del modelo:')
      print(sensibilidad)
```

```
Sensibilidad del modelo:
0.98989898989899
```

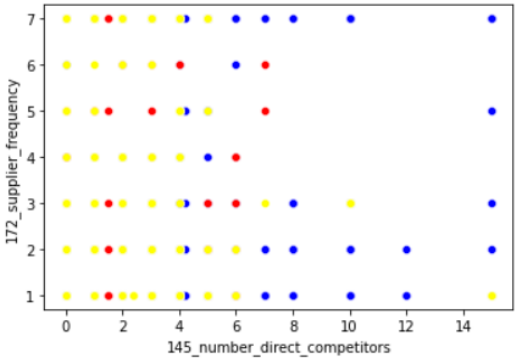
Regresión lineal

En el caso de la regresión lineal, se pueden tomar las mismas variables dependientes e independientes, pero esta vez se grafica un mapa de dispersión de las tres variables independientes en función de la variable dependiente.

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

sns.scatterplot(x='145_number_direct_competitors', y='172_supplier_frequency', color='blue', data=df)
sns.scatterplot(x='97_number_of_customers_in_store', y='172_supplier_frequency', color='red', data=df)
sns.scatterplot(x='2_current_permanent_employees', y='172_supplier_frequency', color='yellow', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f704ef77730>
```



Declaramos todas las variables independientes: *'145_number_direct_competitors', '97_number_of_customers_in_store', '2_current_permanent_employees'* y la variable dependiente: *'172_supplier_frequency'*

```
[163] #Declaramos las variables dependientes e independientes para la regresión lineal.
      Vars_Indep= df[['145_number_direct_competitors', '97_number_of_customers_in_store', '2_current_permanent_employees']]
      Var_Dep= df['172_supplier_frequency']
```

Definimos un modelo de regresión lineal, verificamos la función relacionada a este, ajustamos el modelo con las variables independientes y la variable dependiente y evaluamos la eficiencia de este modelo.

```
#Se define model como la función de regresión lineal
from sklearn.linear_model import LinearRegression
model= LinearRegression()
```

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

```
sklearn.linear_model._base.LinearRegression
```

```
[177] #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__
```

+ Código

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

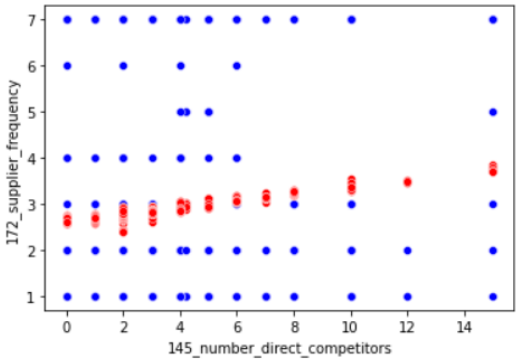
```
0.012577406352438425
```

Realizamos las predicciones de cada variable independiente para después generar graficar separadas de cada una en donde se muestre la variable independiente respectiva contra la variable dependiente, y la misma variable independiente contra las predicciones.

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

sns.scatterplot(x= '145_number_direct_competitors', y= '172_supplier_frequency', color= 'blue', data=df)
sns.scatterplot(x= '145_number_direct_competitors', y= 'Predicciones', color= 'red', data=df)
```

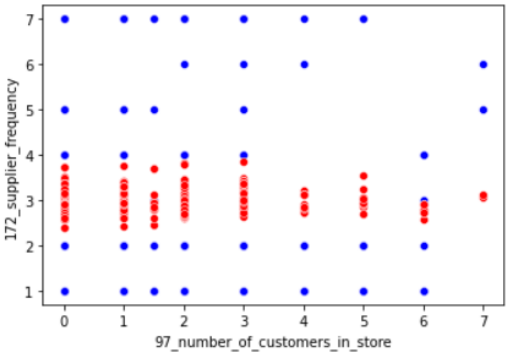
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f704f0e4460>
```




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

sns.scatterplot(x= '97_number_of_customers_in_store', y= '172_supplier_frequency', color= 'blue', data=df)
sns.scatterplot(x= '97_number_of_customers_in_store', y= 'Predicciones', color= 'red', data=df)
```

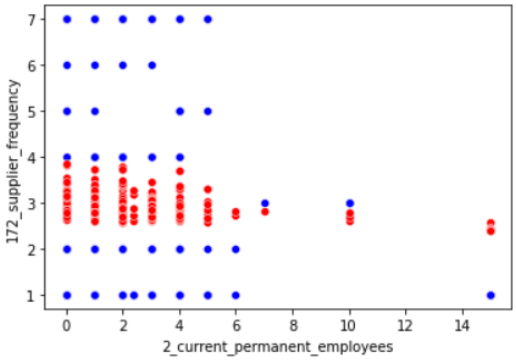
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f704f097ac0>
```



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

sns.scatterplot(x= '2_current_permanent_employees', y= '172_supplier_frequency', color= 'blue', data=df)
sns.scatterplot(x= '2_current_permanent_employees', y= 'Predicciones', color= 'red', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f704f00ddf0>
```



Para terminar, determinamos el coeficiente de correlación del modelo el cual nos dice la intensidad de relación entre dos variables.

```
#Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl=np.sqrt(coef_Deter)
coef_Correl
```

```
0.11214903634199638
```