Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Puebla

TC3006C. Inteligencia artificial avanzada para la ciencia de datos
(Grupo 103)

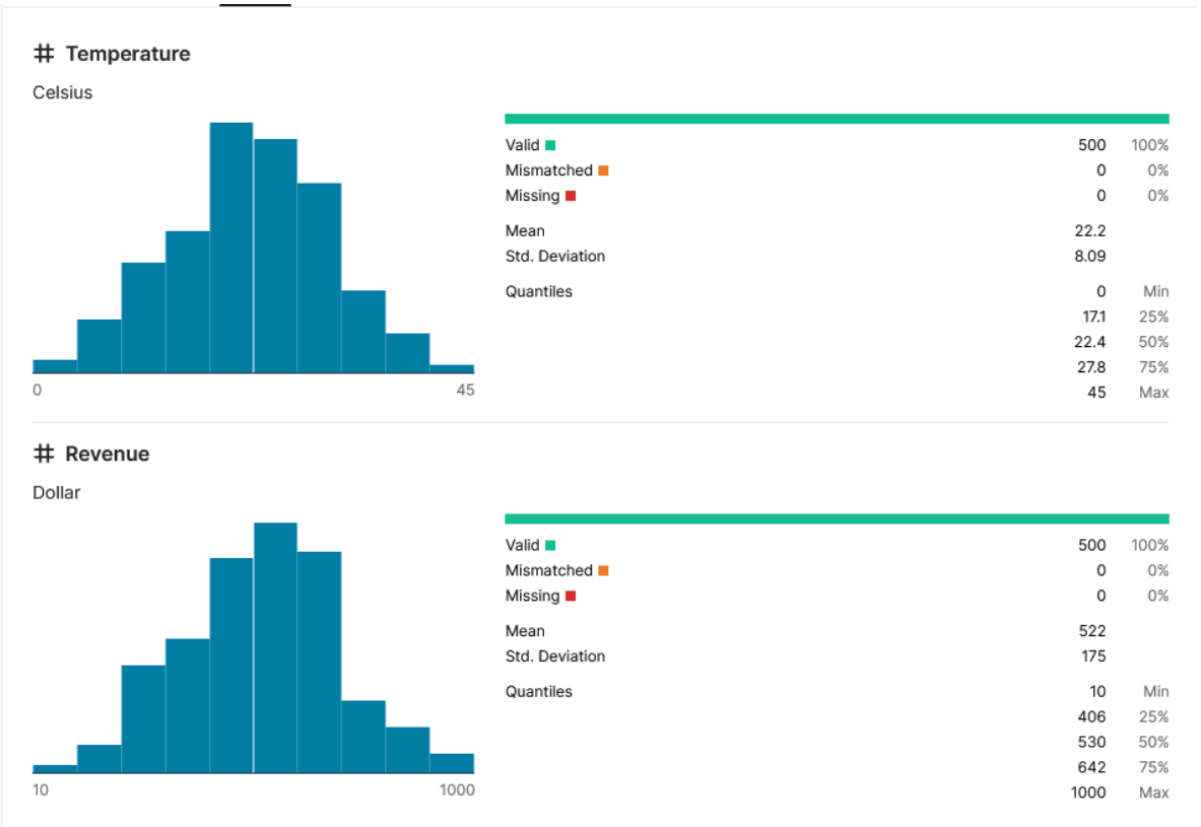
Módulo 2 Análisis y Reporte sobre el desempeño del modelo.

Daniel Munive Meneses A01734205

14 de Septiembre del 2022

DataSet

Para la elección de la Database con la que se va a trabajar se hizo uso de la pagina de kaggle. La DataBase con la que se decidió trabajar fue la de Ice Cream Revenue, la cual es una database de un negocio de helados, una base de datos con la que se pretende poder observar en un modelo la predicción de los ingresos diarios en dólares que tendría la heladería en función a la de la temperatura exterior (siendo que esta estaría en grados Celsius para este caso).



Aquí podemos observar que la limpieza de los datos está correcta, y que el sistema no necesita que se le haga una limpieza de algún tipo, está sin embargo fue agregada dentro de la implementación de Python a razón de asegurarnos de esto y evitar alguna incompatibilidad del Código en caso de que se utilice algún otro Dataframe.

Código

Primero se importaron las librerías con las que íbamos a trabajar.

- pandas para poder trabajar con los datos de la database
- matplotlib.pyplot para el ploteo de las gráficas
- Sci-kit Learn: Librería de Python dedicada al análisis de datos y aprendizaje máquina que va desde modelos simples como regresiones lineales o logísticas hasta redes neuronales complejas.

```
#Se importan las librerias necesarias del proyecto

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.model_selection import cross_val_score
```

Primero se buscan las columnas de Database con las que queremos trabajar, siendo en este caso "Temperature" y "Revenue", posteriormente convertimos los datos a tipo float en caso de que algunos de estos dentro del CSV, hayan cambiado a un tipo diferente de dato, posteriormente se hace una limpieza para asegurar que los datos con los que vamos a trabajar puedan ser utilizados de manera correcta sin afectar al resultado del modelo.

```
#Obtención de los datos a partir de un csv con pandas
columns = ["Temperature", "Revenue"]
df = pd.read_csv('IceCreamData.csv', names = columns)

#Me aseguro que los datos que voy a ocupar esten el el tipo de dato correcto, para poder trabajar con ellos
df['Temperature']= df['Temperature'][1:].astype(float)
df['Revenue'] = df['Revenue'][1:].astype(float)

#Limpieza de datos, considerando que no puede haber datos vacíos en dichas columnas
df = df.drop(df[df.Temperature.isnull()].index)
df = df.drop(df[df.Revenue.isnull()].index)
df = df[df['Revenue'] > 0]

print(df)


#Determinación de la variable correspondiente al 'eje X' y 'eje Y'
X = np.array(df['Temperature']).reshape(-1,1)
Y = np.array(df['Revenue']).reshape(-1,1)
```

Ploteo de los Datos para ver que están correctamente cargados

Posterior a esto procedo a realizar el modelo de regresión lineal con ayuda de las librerías de sklearn,

$$\gamma = \alpha \pm \beta\chi + \varepsilon_i$$

$\gamma = variable\ dependiente$
$\alpha = intersección\ o\ constante$
$\beta = coeficiente\ angular\ de\ la\ regresión$
$\chi = variable\ independiente$
$\varepsilon_i = error$

```
#Division del data set en train y test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.30)

#Se hace el calculo del la regresion por medio de los metodos de sklearn
regr = LinearRegression(fit_intercept = False).fit(X_train,Y_train)

#Coeficiente e Intercepto
print("Coefficient: ",regr.coef_)
print("Intercept: ",regr.intercept_)

#Predicciones
print("Predicitons:")
custom_pred = [[50.0], [30.0], [25.0], [0.0], [-20.0]]

for i in custom_pred:
    print(f"Temperatura en Celsius: {i} Ganancia(Dolares): {regr.predict([i])}")

#Obtencion del error de prediccion en test y train
Y_pred = regr.predict(X_test)
Pred_error_test = Y_pred - Y_test
Y_pred_train = regr.predict(X_train)
Pred_error_train = Y_train - Y_pred_train
```

Para esta implementación primero se hace una división de los datos en train(75%) y test(25%) para los futuros histogramas y gráficas que vamos a requerir, además de esto a redimensionas los arreglos de las columnas con las que se desea trabajar a matrices para que así con esto sea posible implementar la regresión linear de este modelo, una vez a con esto hecho aplicamos la función de LinearRegression, de Sklearn.

```
print("MSE test: ",mean_squared_error(Y_test, Y_pred))
print("Model score test: ", regr.score(X_test, Y_test))

print("MSE train: ",mean_squared_error(Y_train, Y_pred_train))
print("Model score train: ", regr.score(X_train, Y_train))

cv = abs(cross_val_score(regr, X_train, Y_train, cv=10, scoring='r2').mean())
print ("Cross validation: ", cv)
```

Procedo a calcular el ‘mean squared error’ de test y train, asi como el correspondiente score de cada uno de estos modelos, además también procedo a calcular el validation con ayuda del comando de Sklearn, Cross validation, para con esto asegurarnos que el modelo no esta siendo afectado por la división entre test y train, donde con esto obtengo lo siguiente.

```
MSE test:  958.544815363788
Model score test:  0.9692346465882511
MSE train:  815.5072837488008
Model score train:  0.9732620311276792
Cross validation:  0.9702718456851498
```

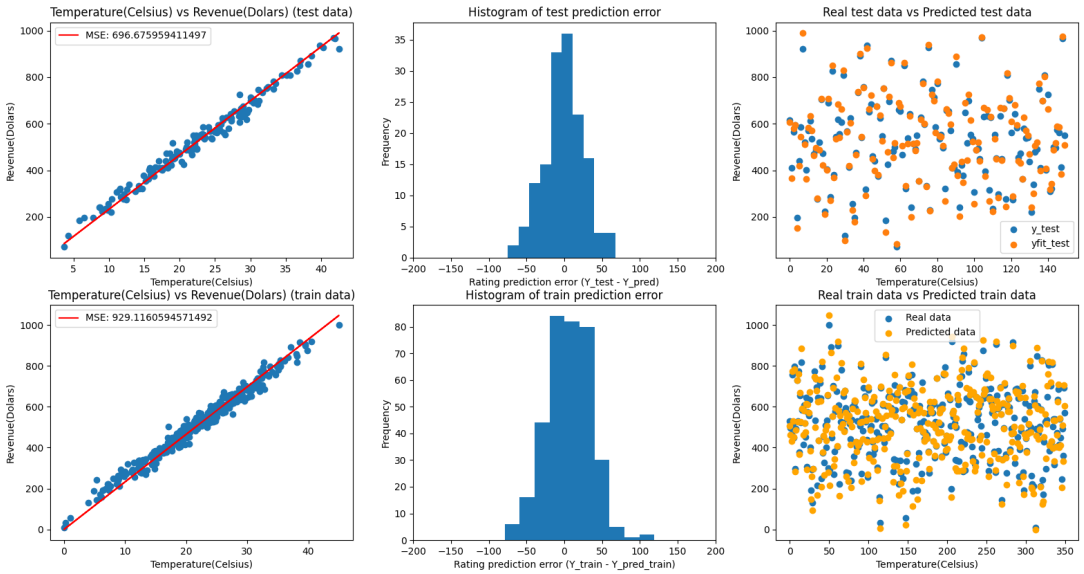
```
#Predicciones
print("Predicitons:")
custom_pred = [[50.0], [30.0], [25.0], [0.0], [-20.0]]

for i in custom_pred:
    print(f"Temperatura en Celsius: {i} Ganancia(Dolares): {regr.predict([i])}")
```

Se colocan diferentes ejemplos de temperaturas dentro de un arreglo, para con estos proceder a hacer la predicción de la regresión lineal, siendo estos los resultados obtenidos.

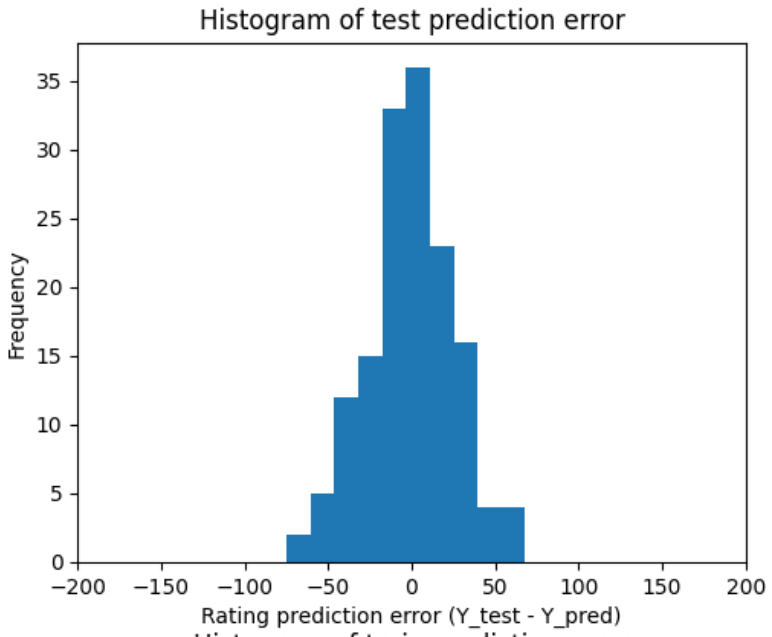
```
Predicitons:
Temperatura en Celsius: [50.0] Ganancia(Dolares): [[1159.88618205]]
Temperatura en Celsius: [30.0] Ganancia(Dolares): [[695.93170923]]
Temperatura en Celsius: [25.0] Ganancia(Dolares): [[579.94309103]]
Temperatura en Celsius: [0.0] Ganancia(Dolares): [[0.]]
Temperatura en Celsius: [-20.0] Ganancia(Dolares): [[-463.95447282]]
```

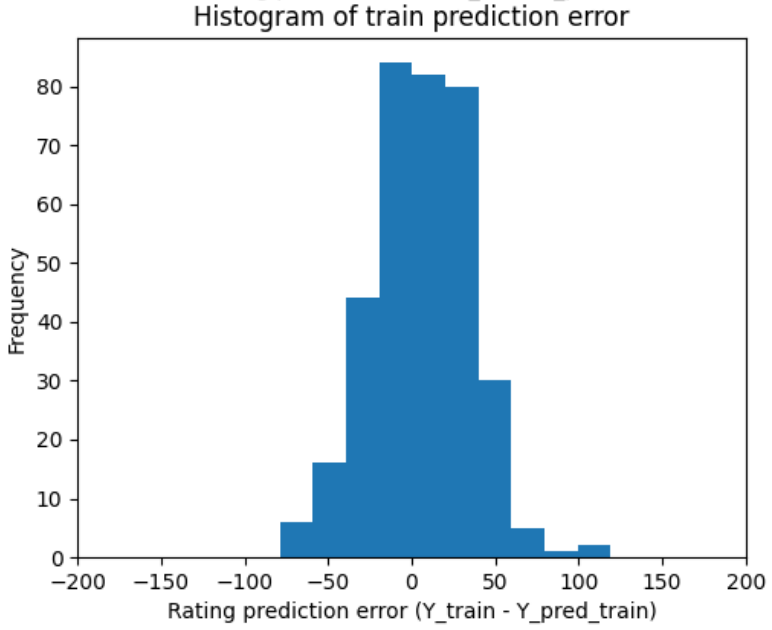
Para los resultados además calculamos el error de predicción en test y train, para posteriormente proceder a plotearlos, siendo que así obtenemos los siguientes resultados.



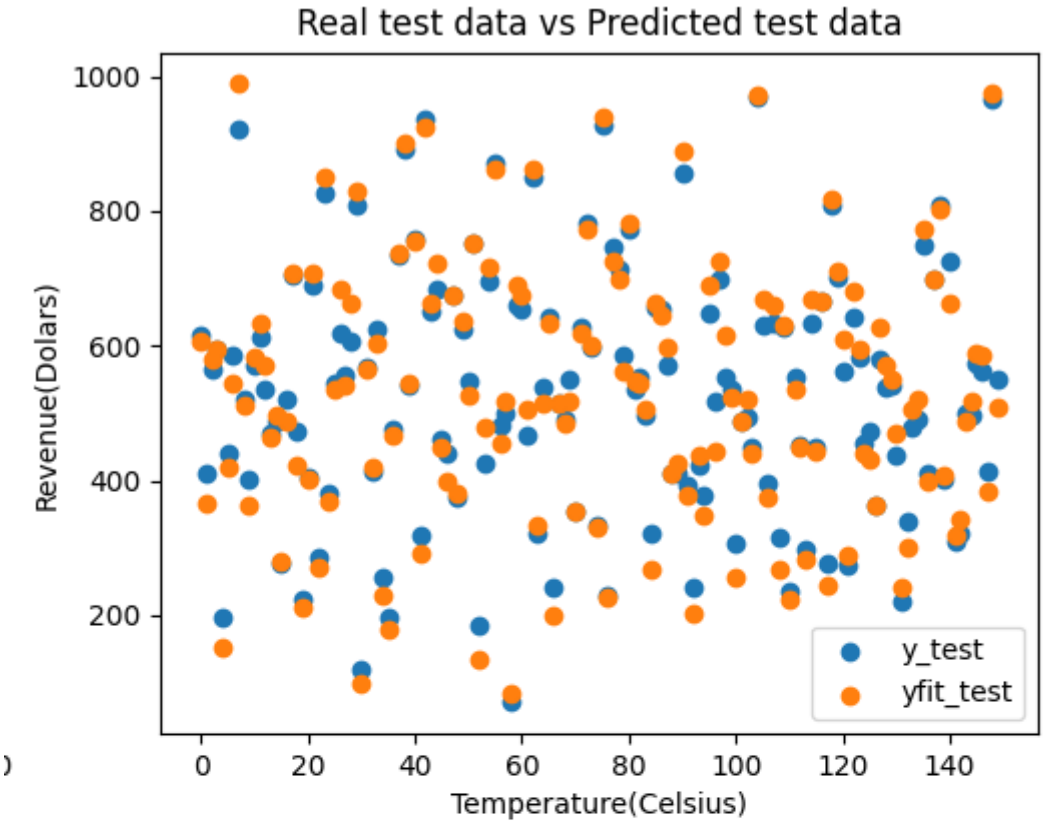
Así mismo como podemos observar en esta foto, el modelaje se encuentra con un score de confianza calculado en base a la R^2 bastante cercano a 1, por lo que podríamos decir que este modelo es bastante certero y confiable.

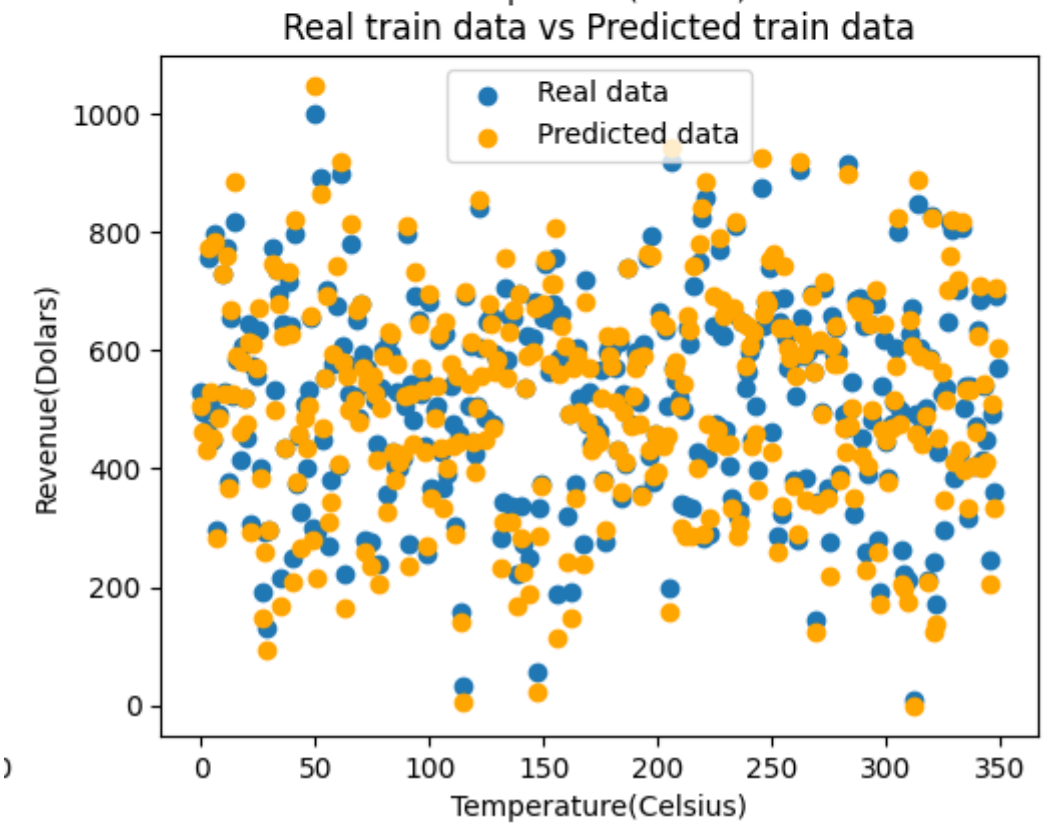
```
[500 rows x 2 columns]
Coefficient:  [[23.19772364]]
Intercept:  0.0
Predicitons:
Temperatura en Celsius: [50.0] Ganancia(Dolares): [[1159.88618205]]
Temperatura en Celsius: [30.0] Ganancia(Dolares): [[695.93170923]]
Temperatura en Celsius: [25.0] Ganancia(Dolares): [[579.94309103]]
Temperatura en Celsius: [0.0] Ganancia(Dolares): [[0.]]
Temperatura en Celsius: [-20.0] Ganancia(Dolares): [[-463.95447282]]
MSE test:  958.544815363788
Model score test:  0.9692346465882511
MSE train:  815.5072837488008
Model score train:  0.9732620311276792
Cross validation:  0.9702718456851498
```





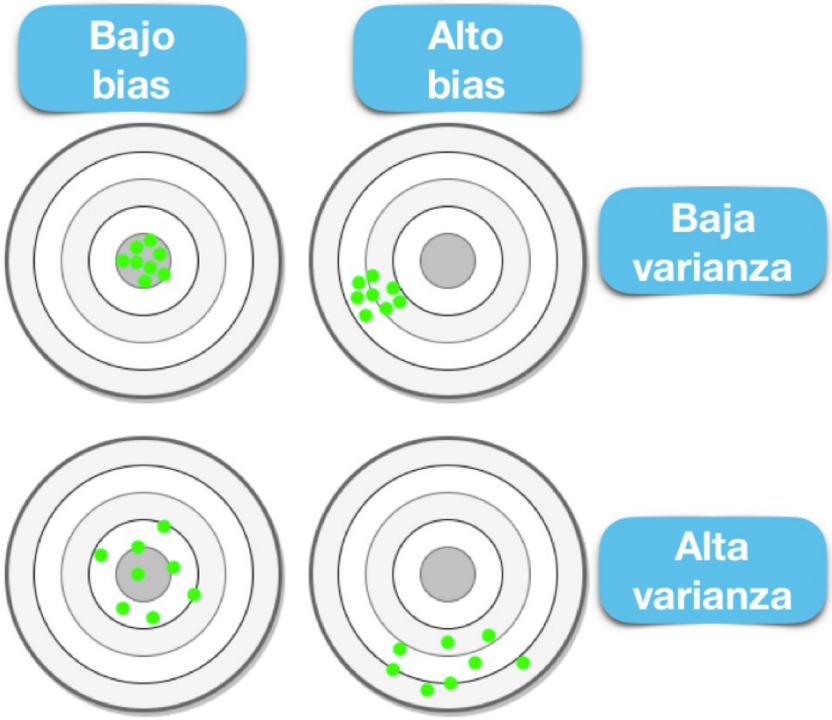
En cuanto al sesgo del dataset se tomó en cuenta el error de la predicción del modelo tanto en Train Dataset (EP = Y_train - Y_pred_train) y en Test Dataset(EP = Y_tests - Y_pred), y viendo los histogramas resultantes podemos concluir que en el caso de test parece haber un ajuste adecuado, con el pico más alto en lo que parece ser 0-25, mientras que en el caso del train data podemos notar que hay una dispersión más equivalente entre los distintos picos, luciendo un poco asimétrico.





Posteriormente se puede ver en el siguiente plot scatter que la diferencia entre las secuencias de los resultados de mi data en comparación con la secuencia de las

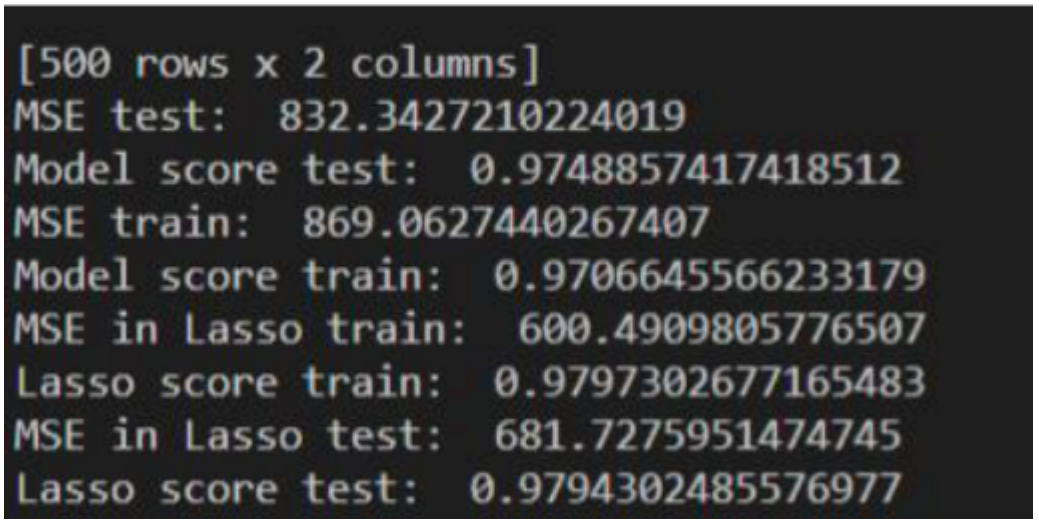
predicciones podemos observar que estas se encuentran bastante cercanas unas a las otras por lo que podemos concluir que este modelo tiene un **Bias Bajo.**



En cuanto a la varianza podemos notar que gracias el tipo de database con la que estamos trabajando esta puede ser predecida con ayuda de una regresión lineal, así mismo con razón a esto, y a lo observado en la gráfica anterior podemos deducir que nuestro modelo cuenta con una **Varianza baja.**

Con esto finalmente podemos concluir como que el modelo es **Fit**, ya que el modelo tiene bias bajo y varianza baja.

Finalmente, para buscar una mejora en el modelo se utilizó el modelo Lasso



Donde se muestra una reducción de los datos tanto en Test y en Train, y aunque con esto se muestra un score ligeramente superior al que teníamos anteriormente creo que con esto podemos concluir que, si bien el modelo de regresión linear nos está dando una calificación buena, no es del todo confiable si es que queremos tener predicciones 100% certeras, por lo que un área de mejora que podría ser implementada es probar con un modelo que nos permita tener una pendiente más clara.

Conclusión

Fue interesante aprender sobre los diferentes modelos que se pueden aplicar para comprender una Database, siendo que si bien en un ejemplo como el que decidí elegir para esta entrega, ya estaba un poco pensado para ser usado en un diagrama de regresión linear, es curioso ver como este término siendo Fit, además el uso de librerías de Python me ayudó a graficar o calcular los diferentes factores y variables necesarios para poder interpretar el modelo de una manera más sencilla, cosa que me sorprendió por el gran apoyo que ofreció esto para el proceso de modelaje.

Herramientas

- Scikit-Learn: Librería de Python dedicada al análisis de datos y aprendizaje máquina que va desde modelos simples como regresiones lineales o logísticas hasta redes neuronales complejas.
- Matplotlib: Librería de Python dedicada a la graficación de datos, fundamental para poder comprender el comportamiento de los datos, por ejemplo, a partir de su ordenamiento o introducción a un modelo determinado.
- Pandas: Librería de Python open source dedicada al análisis de datos de manera intuitiva y flexible. Pandas puede abarcar la gran mayoría de ETL, sobre todo siendo fundamental para la extracción, limpieza y transformación de datos a emplear.
- Numpy: Librería de Python dedicada a ciencia de datos desde la perspectivade vectorización, indexado, funciones matemáticas, funciones algebraicas, permutación de matrices, cálculo vectorial, etcétera.