Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Puebla

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

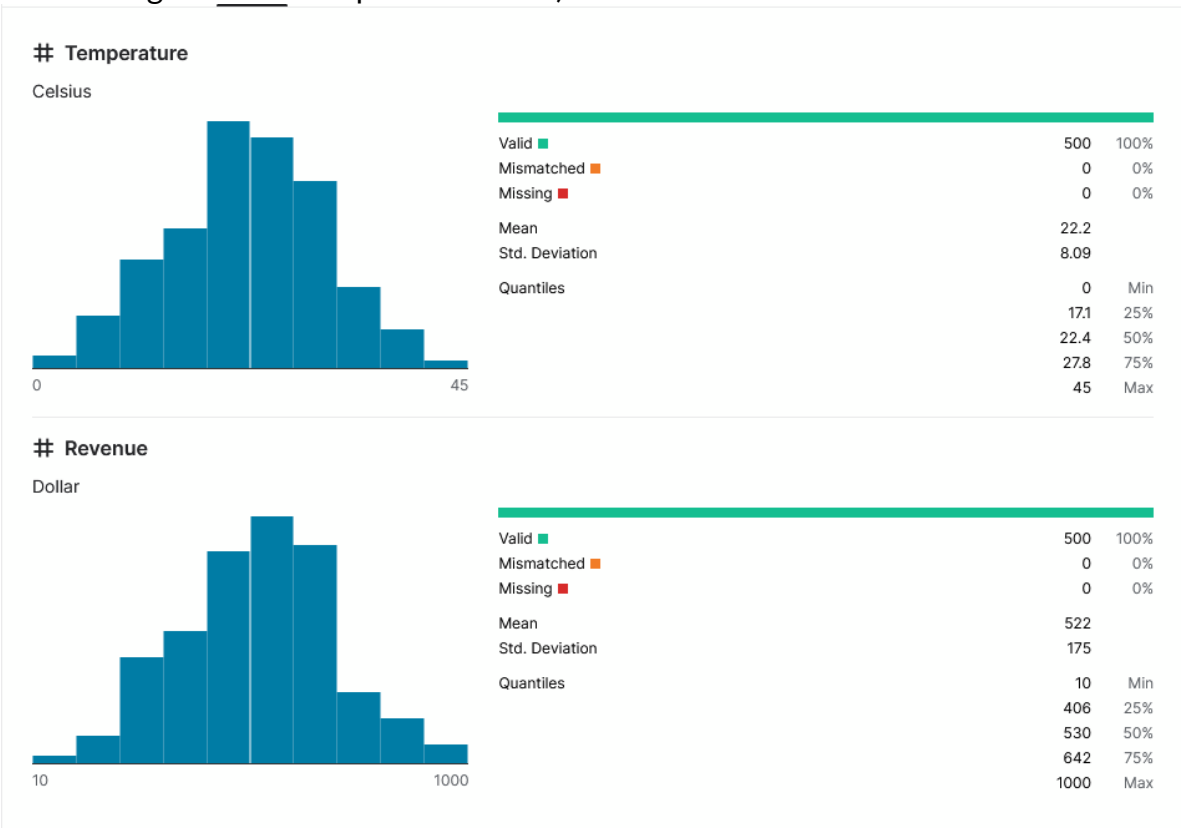
Módulo 2 Análisis y Reporte sobre el desempeño del modelo.

Daniel Munive Meneses A01734205

10 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 esta correcta, y que el sistema no necesita que se le haga una limpieza de algún tipo, esta 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.

Codigo

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 graficas
- 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.

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.

```python
#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)
```

```
PS C:\Users\danie\Documents\AAA
     Temperature     Revenue
1      24.566884  534.799028
2      26.005191  625.190122
3      27.790554  660.632289
4      20.595335  487.706960
5      11.503498  316.240194
..           ...         ...
496    22.274899  524.746364
497    32.893092  755.818399
498    12.588157  306.090719
499    22.362402  566.217304
500    28.957736  655.660388
```

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 metoddos 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

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

#Plot de la regresión
figure, axis = plt.subplots(2,3)
```

Para esta implementación primero se hace una división de los datos en train(75%) y test(25%) para los futuros histogramas y graficas 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.

```python
#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 esos los resultados obtenidos.

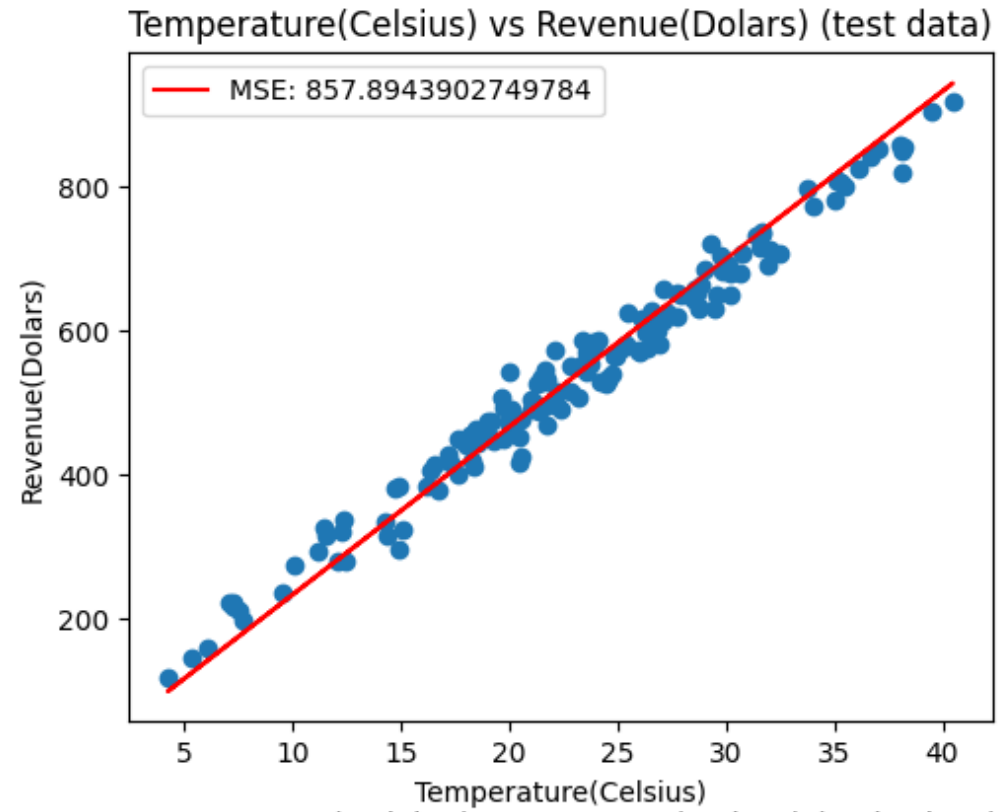
```
Coefficient:  [[23.30991617]]
Intercept:  0.0
Predicitons:
Temperatura en Celsius: [50.0] Ganancia(Dolares): [[1165.49580869]]
Temperatura en Celsius: [30.0] Ganancia(Dolares): [[699.29748522]]
Temperatura en Celsius: [25.0] Ganancia(Dolares): [[582.74790435]]
Temperatura en Celsius: [0.0] Ganancia(Dolares): [[0.]]
Temperatura en Celsius: [-20.0] Ganancia(Dolares): [[-466.19832348]]
MSE test:  857.8943902749784
Model score test:  0.9691047146374929
MSE train:  863.9171593485383
Model score train:  0.9727733368014434
```

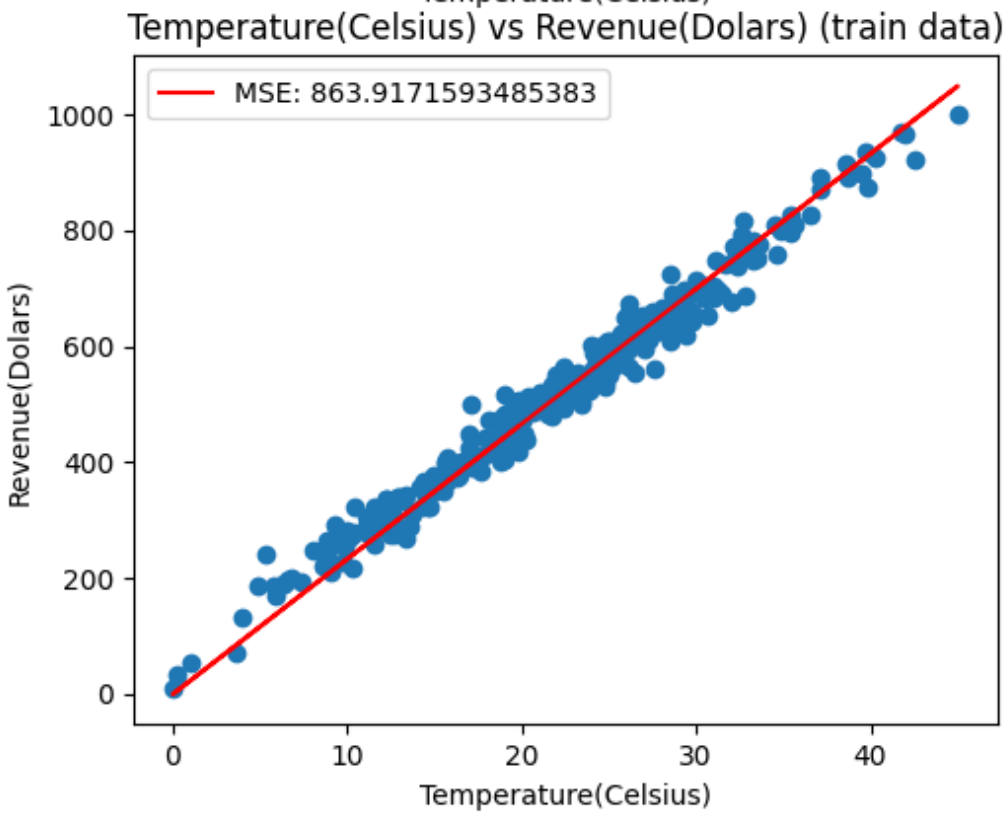
Para los resultados además calculamos el error de predicción en test y train, para posteriormente proceder a plotearlos, siendo que así obtenemos lo siguientes resultados.

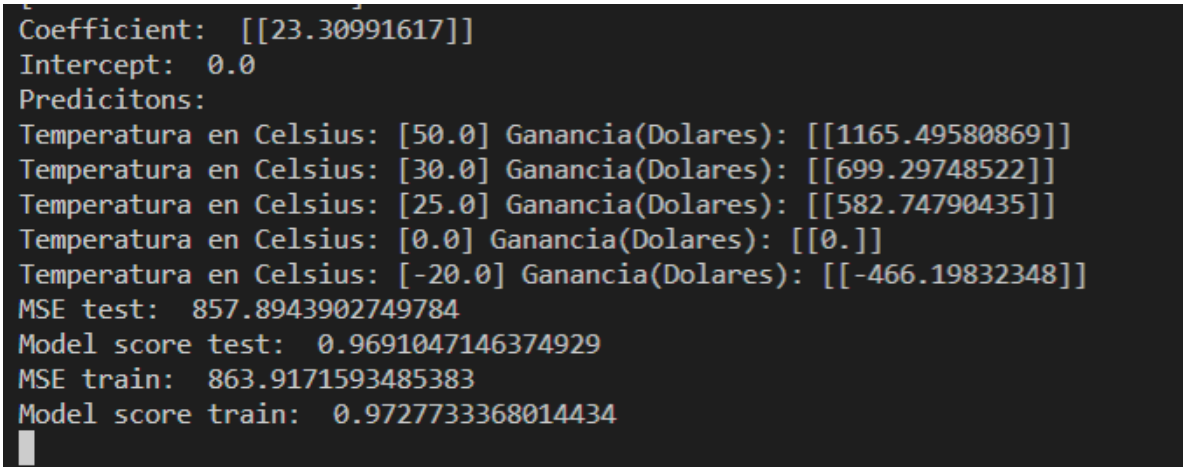
```python
#Obtencion del error de prediccion en test y train
Y_pred = regr.predict(X_test)
Pred_error_test = Y_pred - Y_test
Y_pred_train = regr.predict(X_train)
Pred_error_train = Y_train - Y_pred_train

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))

#Plot de la regresión
figure, axis = plt.subplots(2,3)
```

```python
#TEST
axis[0,0].scatter(X_test, Y_test)
axis[0,0].plot(X_test, Y_pred, color='red', label = "MSE: " + str(mean_squared_error(Y_test, Y_pred)))
axis[0,0].set_title("Temperature(Celsius) vs Revenue(Dolars) (test data)")
axis[0,0].set(xlabel = 'Temperature(Celsius)', ylabel = 'Revenue(Dolars)')
axis[0,0].legend()

#histograma(bias)
axis[0,1].hist(Pred_error_test)
axis[0,1].set_title('Histogram of test prediction error')
axis[0,1].set_xlim(-200, 200)
axis[0,1].set(xlabel = 'Temperature(Celsius)', ylabel = 'Revenue(Dolars)')

#varianza(?)
axis[0,2].scatter(X_test, Y_test, alpha = 0.5, label = 'Real data')
axis[0,2].scatter(X_test, Pred_error_test, color='orange',alpha = 0.1, label = 'Predicted data')
axis[0,2].set_title("Real test data vs Predicted test data")
axis[0,2].set(xlabel = 'Temperature(Celsius)', ylabel = 'Revenue(Dolars)')
axis[0,2].legend()
```

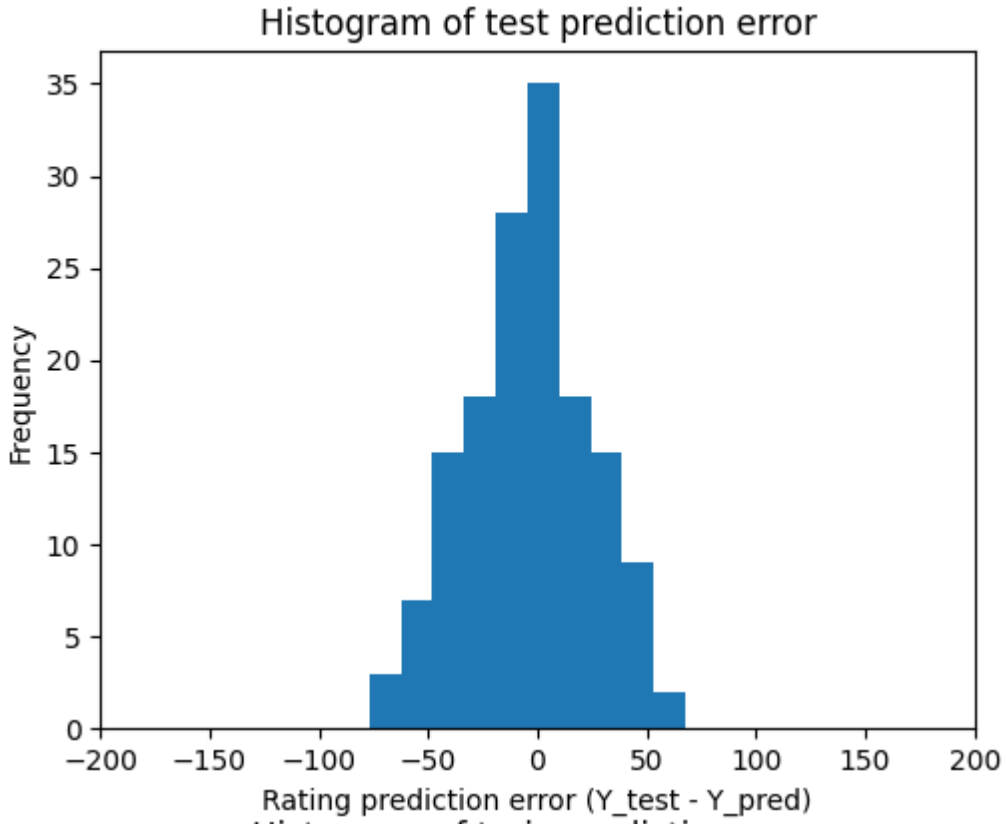
```python
#TRAIN
axis[1,0].scatter(X_train, Y_train)
axis[1,0].plot(X_train, Y_pred_train, color ='red',label = "MSE: " + str(mean_squared_error(Y_train, Y_pred_train)))
axis[1,0].set_title("Temperature(Celsius) vs Revenue(Dolars) (train data)")
axis[1,0].set(xlabel = 'Temperature(Celsius)', ylabel = 'Revenue(Dolars)')
axis[1,0].legend()

#histograma(bias)
axis[1,1].hist(Pred_error_train)
axis[1,1].set_title('Histogram of train prediction error')
axis[1,1].set_xlim(-200, 200)
axis[1,1].set(xlabel = 'Temperature(Celsius)', ylabel = 'Revenue(Dolars)')

#varianza(?)
axis[1,2].scatter(X_train, Y_train, alpha = .5, label = 'Real data')
axis[1,2].scatter(X_train, Pred_error_train, color='orange',alpha = 0.1, label = 'Predicted data')
axis[1,2].set_title("Real train data vs Predicted train data")
axis[1,2].set(xlabel = 'Temperature(Celsius)', ylabel = 'Revenue(Dolars)')
axis[1,2].legend()
plt.show()
```

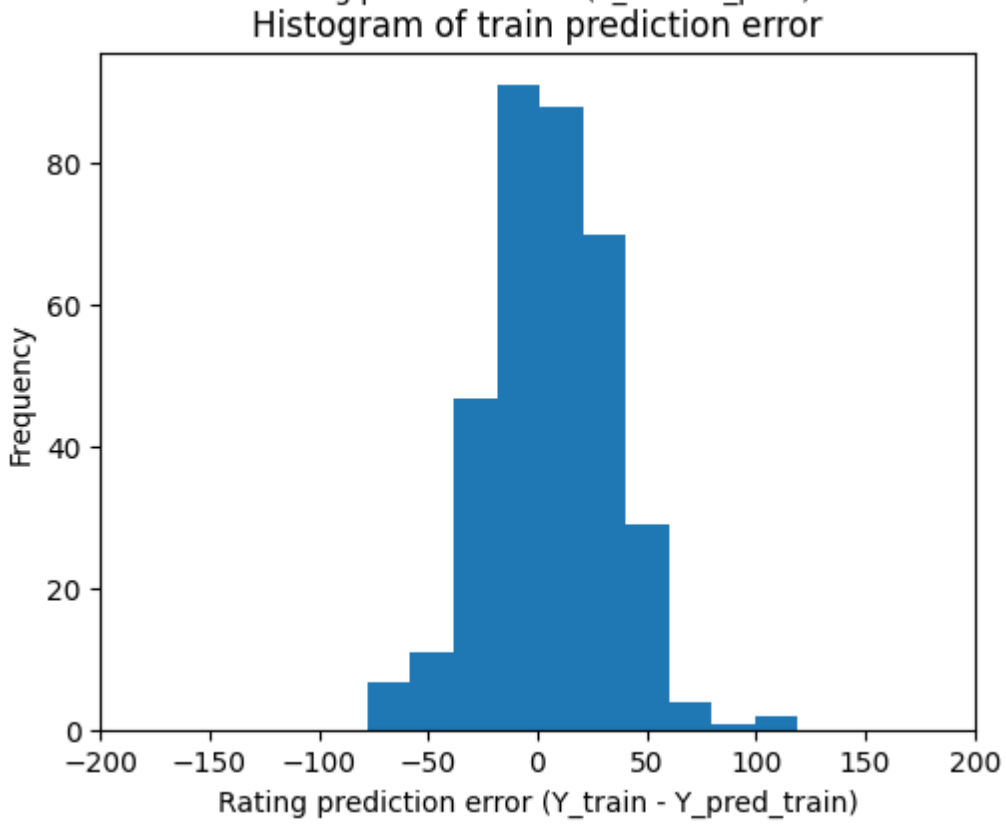

Temperature(Celsius) vs Revenue(Dolars) (test data)
MSE: 857.8943902749784
Revenue(Dolars)
Temperature(Celsius)
200
400
600
800
5
10
15
20
25
30
35
40


Temperature(Celsius) vs Revenue(Dolars) (train data)
MSE: 863.9171593485383
Revenue(Dolars)
Temperature(Celsius)
0
200
400
600
800
1000
0
10
20
30
40

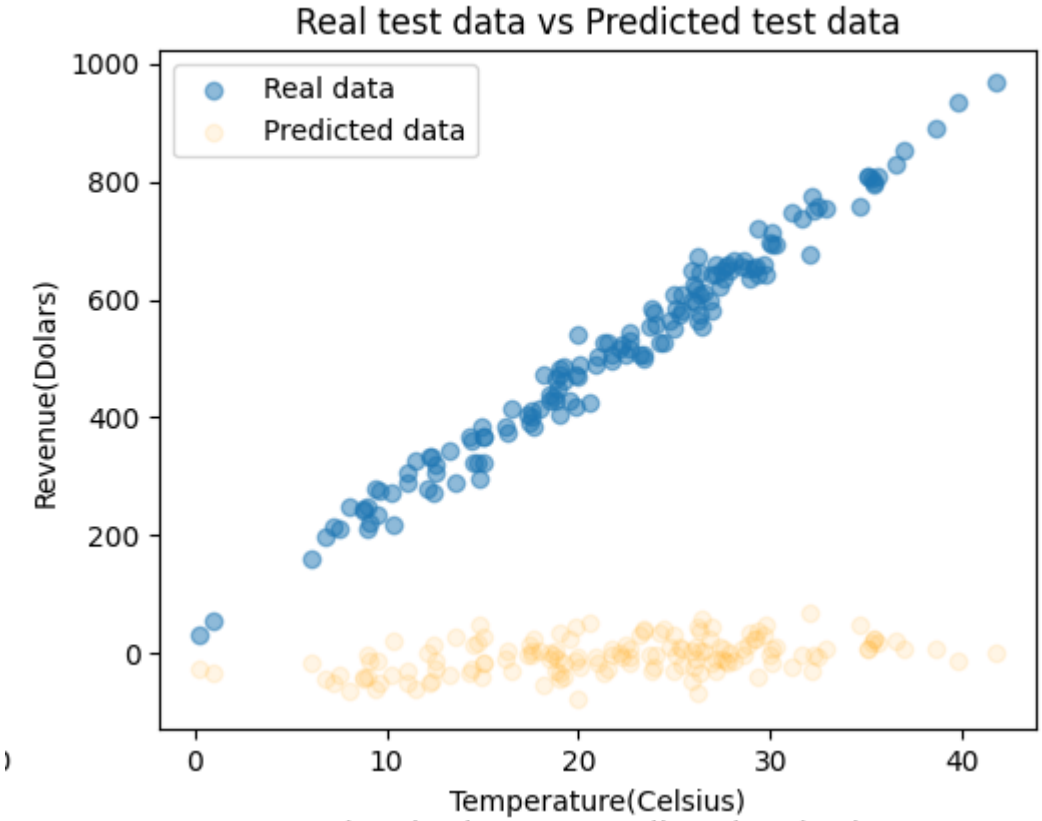


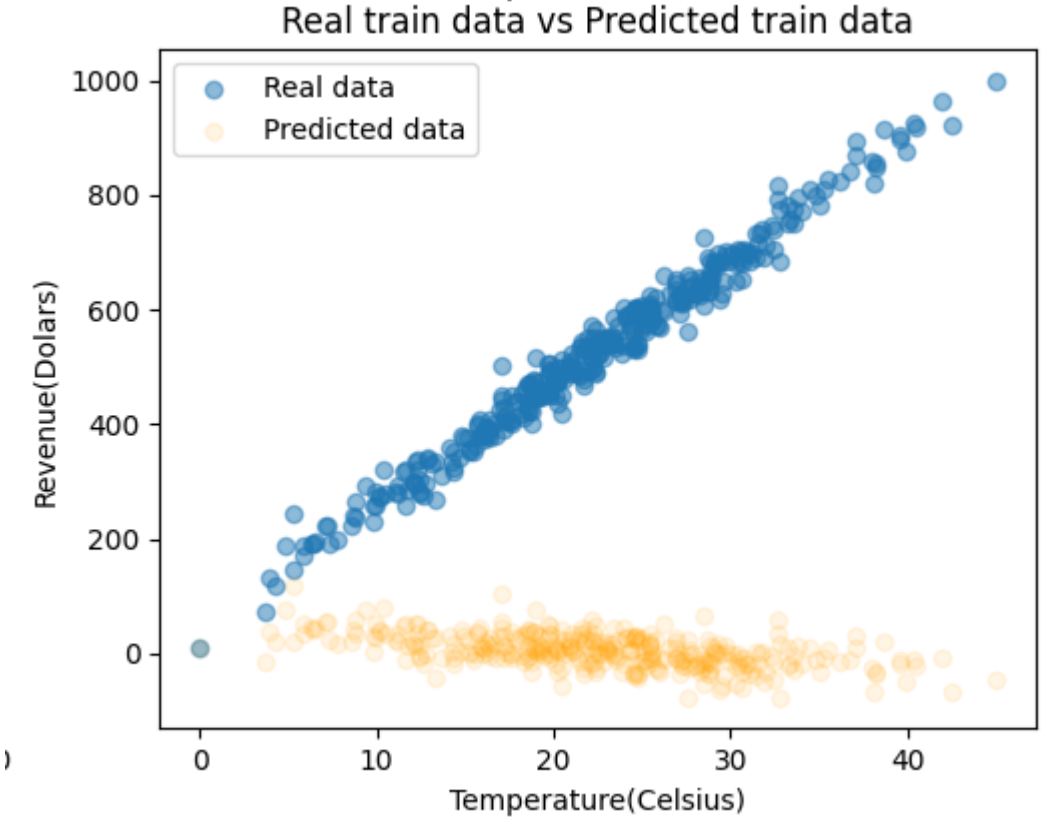
Así mismo como podemos observar en esta foto, el modelaje se encuentra con un score de confianza calculado en base a la R^2 bastante cercano a 1, por lo que podríamos decir que este modelo es bastante certero y confiable.


Histogram of test prediction error
Frequency
0
5
10
15
20
25
30
35
−200
−150
−100
−50
0
50
100
150
200
Rating prediction error (Y_test - Y_pred)

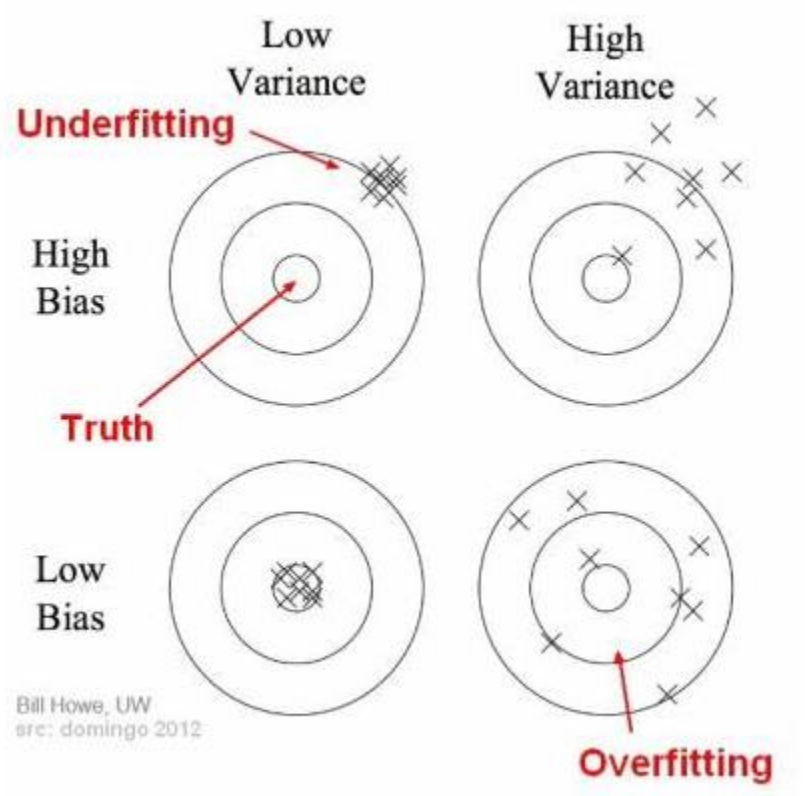

Histogram of train prediction error
Frequency
0
20
40
60
80
−200
−150
−100
−50
0
50
100
150
200
Rating prediction error (Y_train - Y_pred_train)

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, pero sin embargo podemos concluir en que el modelo tiene un **Bias alto.**


Real test data vs Predicted test data
Real data
Predicted data
Revenue(Dolars)
Temperature(Celsius)
0
200
400
600
800
1000
0
10
20
30
40

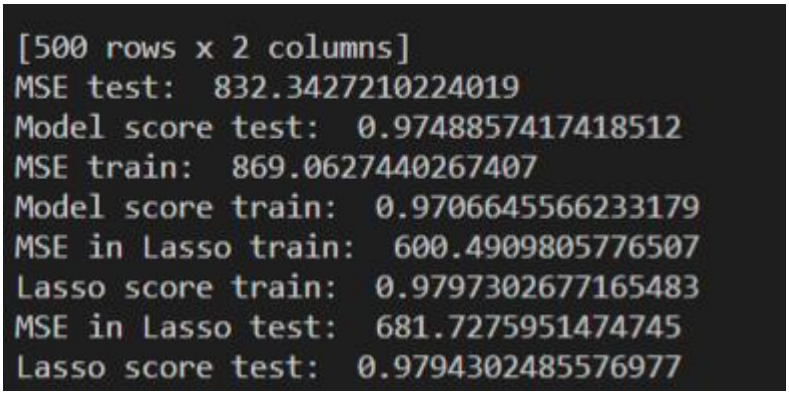

Real train data vs Predicted train data
Real data
Predicted data
Revenue(Dolars)
Temperature(Celsius)
0
200
400
600
800
1000
0
10
20
30
40

En cuanto a la varianza se puede ver una separación clara entre los datos reales y los que se predijeron, se pude observar que si bien aparentemente ambos modelos parece que tienen una pendiente parecida, estos tienen una separación considerable, lo que no llega a concluir que el modelo tiene una **Varianza baja o Low Variance.**



Esto no llevaría a concluir que el modelo es **Underfitting**, esto a decir verdad era algo esperable ya que los diagramas de regresión lineal tienden a entrar dentro de este nivel de ajuste, y únicamente nos ayudan como aproximación de una dependencia entre dos variables, una siendo dependiente de la otra.

Finalmente, para buscar mejor 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 underfittig, además el uso de librerías de Python me ayudo 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.