Summer Industrial Internship Program

Taxi Trip Duration prediction analysis

Capabl.

Table of Content

Capabl.

Objective :

CHALLENGE

BUSINESS GOAL: TO IMPROVE THE EFFICIENCY OF ELECTRONIC TAXI DISPATCHING SYSTEMS IT IS IMPORTANT TO BE ABLE TO PREDICT HOW LONG A DRIVER WILL HAVE HIS TAXI OCCUPIED. IF A DISPATCHER KNEW APPROXIMATELY WHEN A TAXI DRIVER WOULD BE ENDINGTHEIR CURRENT RIDE, THEY WOULD BE BETTER ABLE TO IDENTIFY WHICH DRIVER TO ASSIGN TO EACH PICKUP REQUEST.

ML GOAL: TO BUILD A MODEL THAT PREDICTS THE TOTAL RIDE DURATION OF TAXI TRIPS IN NEW YORK CITY

Capabl.

Data Gathering

The training dataset provided is a popular dataset used for various machine learning and data analysis tasks. It contains information about taxi trips in New York City, including details such as pickup and drop-off locations, timestamps, passenger counts, trip distances, and fare amounts. The dataset is typically provided in CSV (Comma-Separated Values) format

Data Gathering

- the dataset consists of the following features

id	a unique identifier for each trip
vendor_id	a code indicating the provider associated with the trip record
pickup_datetime	date and time when the meter was engaged
dropoff_datetime	date and time when the meter was disengaged
passenger_count	the number of passengers in the vehicle (driver entered value)
pickup_longitude	the longitude where the meter was engaged
pickup_latitude	the latitude where the meter was engaged
dropoff_longitude	the longitude where the meter was disengaged
dropoff_latitude	the latitude where the meter was disengaged
store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server Y=store and forward; N=not a store and forward trip
trip_duration	duration of the trip in seconds

Data Pre-Processing :

- Data preprocessing is an essential step in preparing data for analysis and machine learning models. It involves transforming raw data into a format suitable for further analysis and ensuring its quality, consistency, and compatibility with the chosen algorithms

Capabl.

Here are the processing steps involved :

Data Cleaning

keeping essential columns in the dataset for training our model and prediction of values.

Data Cleaning

- here our target variable is trip_duration
- to predict trip_duration we dont need id, vendor_id so we drop the columns
- the time and date can be know using the latitudes and longitudes so we dont need the (pickup and dropoff) datetime columns so we drop them also.

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
ds1 = ds1.drop(['id','vendor_id','pickup_datetime','dropoff_datetime'],axis=1)
ds1.head()
```

[16]:

	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration
0	1	-73.982155	40.767937	-73.964630	40.765602	N	455
1	1	-73.980415	40.738564	-73.999481	40.731152	N	663
2	1	-73.979027	40.763939	-74.005333	40.710087	N	2124
3	1	-74.010040	40.719971	-74.012268	40.706718	N	429
4	1	-73.973053	40.793209	-73.972923	40.782520	N	435

+ Code + Markdown

Missing Values : (none found)

```
[6]: ds1.isnull().sum()
```

```
[6]: id                  0
     vendor_id           0
     pickup_datetime     0
     dropoff_datetime    0
     passenger_count     0
     pickup_longitude    0
     pickup_latitude     0
     dropoff_longitude   0
     dropoff_latitude    0
     store_and_fwd_flag  0
     trip_duration       0
     dtype: int64
```

```
ds2.isnull().sum()
```

```
[7]: id                  0
     vendor_id           0
     pickup_datetime     0
     passenger_count     0
     pickup_longitude    0
     pickup_latitude     0
     dropoff_longitude   0
     dropoff_latitude    0
     store_and_fwd_flag  0
     dtype: int64
```

+ Code + Markdown

Exploratory Data Analysis (EDA) :

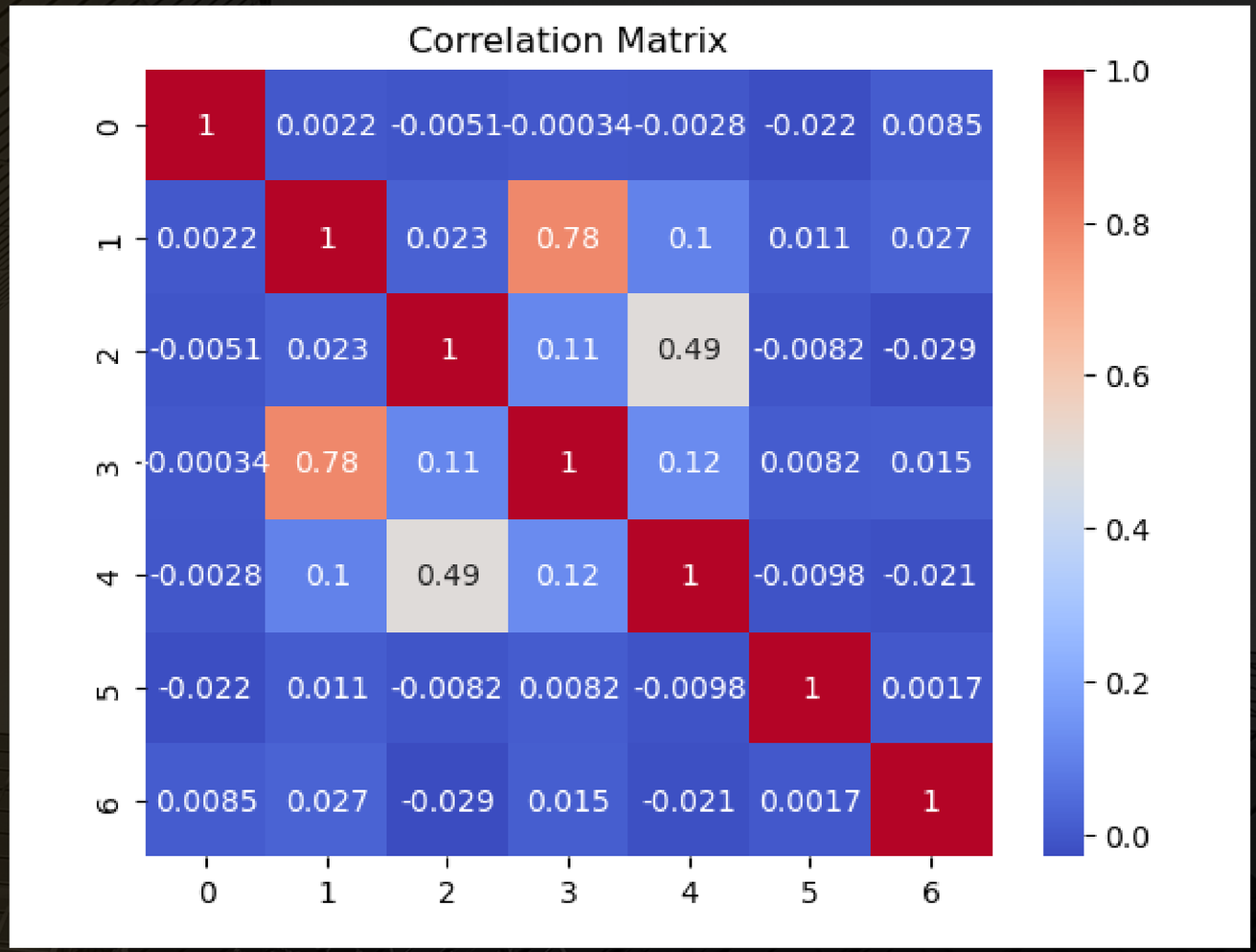
correlation matrix

[37]:

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
import seaborn as sns
# Correlation matrix
corr_matrix =ds1.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Exploratory Data Analysis (EDA) :

correlation matrix



Capabl.

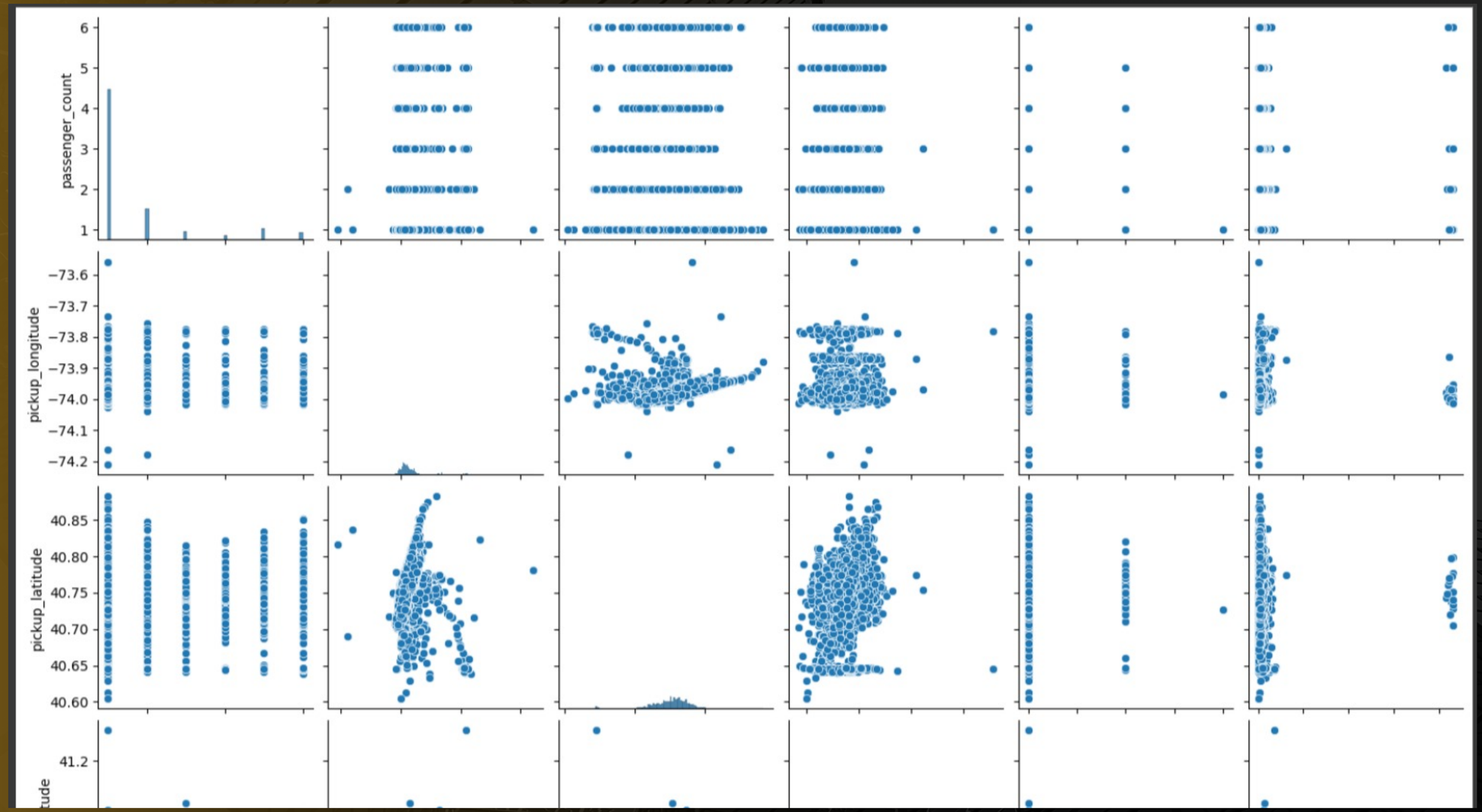
Exploratory Data Analysis (EDA) :

Pair Plot :

```
#SIDDHARTH UPADHYAY ; RGNO:21BCE6103-
sns.pairplot(df)
plt.title('Pair Plot')
plt.show()
```

Exploratory Data Analysis (EDA) :

Pair Plot :



Capabl.

Data Encoding : (Label Encoder)

DATA ENCODING REFERS TO THE PROCESS OF TRANSFORMING CATEGORICAL OR TEXTUAL DATA INTO A NUMERICAL REPRESENTATION THAT CAN BE UNDERSTOOD BY MACHINE LEARNING ALGORITHMS.

- now we can see that the column store_and_fwd_flag has string data so we need to encode it and we will do so using label encoder

```
# SIDDHARTH UPADHYAY; RGNO : 21BCE6103
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
ds1['store_and_fwd_flag'] = le.fit_transform(ds1['store_and_fwd_flag'])
```

+ Code + Markdown

Data Encoding : (Label Encoder)

- now we have converted the string column to numerical values

+ Code + Markdown

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
print(ds1['store_and_fwd_flag'].value_counts())
ds1.head()
```

```
0    1450599
1       8045
Name: store_and_fwd_flag, dtype: int64
```

[24]:

	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration
0	1	-73.982155	40.767937	-73.964630	40.765602	0	455
1	1	-73.980415	40.738564	-73.999481	40.731152	0	663
2	1	-73.979027	40.763939	-74.005333	40.710087	0	2124
3	1	-74.010040	40.719971	-74.012268	40.706718	0	429
4	1	-73.973053	40.793209	-73.972923	40.782520	0	435

Data Scaling :

Data scaling, also known as feature scaling or normalization, is a preprocessing step in machine learning that aims to bring all input features to a similar scale or range.

Min-Max Scaling (Normalization): This method scales the data to a fixed range, often between 0 and 1. It subtracts the minimum value from each feature and then divides by the difference between the maximum and minimum values.

Data Scaling :

- here we can see that there is a lot of variation among the observations so we need to scale the data for that we'll use min max scaler

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
```

+ Code + Markdown

[31]:

```
ds1 = mms.fit_transform(ds1)
```

Data Scaling :

- the resultant data after scaling is a numpy array so we convert it to a pandas dataframe.

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
ds1 = pd.DataFrame(ds1)
ds1.head()
```

[33]:

	0	1	2	3	4	5	6
0	0.111111	0.791302	0.365738	0.791591	0.731222	0.0	0.000129
1	0.111111	0.791331	0.364062	0.791016	0.728287	0.0	0.000188
2	0.111111	0.791354	0.365510	0.790920	0.726493	0.0	0.000602
3	0.111111	0.790842	0.363001	0.790805	0.726206	0.0	0.000121
4	0.111111	0.791452	0.367181	0.791454	0.732663	0.0	0.000123

Feature Creation :

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
X = ds1.drop([6],axis=1)
y = ds1[6]
print(X.head())
y.head()
```

```
          0         1         2         3         4    5
0  0.111111  0.791302  0.365738  0.791591  0.731222  0.0
1  0.111111  0.791331  0.364062  0.791016  0.728287  0.0
2  0.111111  0.791354  0.365510  0.790920  0.726493  0.0
3  0.111111  0.790842  0.363001  0.790805  0.726206  0.0
4  0.111111  0.791452  0.367181  0.791454  0.732663  0.0
```

[19]:
```
0    0.000129
1    0.000188
2    0.000602
3    0.000121
4    0.000123
Name: 6, dtype: float64
```

+ Code + Markdown

Model Training :

- USING TRAIN_TEST_SPLIT AND LINEAR REGRESSION .

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.3 , random_state = 1)
```

+ Code + Markdown

[25]:

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
print(X_train.size,y_train.size,X_test.size,y_test.size)
```

```
6126300 1021050 2625564 437594
```

Model Training :

[26]:
```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
lr.fit(X_train,y_train)
```

[27]:
```
▾ LinearRegression
LinearRegression()
```

Model Training :

- PREDICTING NEW VALUES USING OUR TRAINED MODELL

[28]:
```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
y_pred = lr.predict(X_test)
```

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
y_pred.reshape(-1,1)
```

[29]:
```
array([[0.0002378 ],
       [0.00019682],
       [0.00026766],
       ...,
       [0.0003252 ],
       [0.00020642],
       [0.00020395]])
```

+ Code + Markdown

Model Testing :

- MODEL TESTING IS A CRUCIAL STEP IN THE MACHINE LEARNING WORKFLOW, WHERE THE TRAINED MODEL'S PERFORMANCE IS EVALUATED ON UNSEEN DATA TO ASSESS ITS GENERALIZATION CAPABILITIES. IT INVOLVES APPLYING THE TRAINED MODEL TO TEST DATA AND ANALYZING ITS PREDICTIONS TO MEASURE ITS ACCURACY, ROBUSTNESS, AND RELIABILITY.
- IMPORTING NECESSARY LIBRARIES

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error
print("imported !!")
```

```
imported !!
```

Model Testing :

R2_SCORE

The R2 score measures the proportion of the variance in the dependent variable (target) that is predictable from the independent variables (features).

MEAN_SQUARED_ERROR

The MSE represents the average of the squared differences between the true values and the predicted values. A lower MSE indicates better performance.

MEAN_ABSOLUTE_ERROR

The MAE represents the average absolute difference between the true values and the predicted values. A lower MAE indicates better performance.

Model Testing :

- MEAN_SQUARED_ERROR: IN THIS CASE, WITH A VERY LOW MSE OF 2.4724E-06, IT FURTHER CONFIRMS THAT THE MODEL IS MAKING VERY ACCURATE PREDICTIONS, WITH VERY SMALL SQUARED DIFFERENCES BETWEEN THE PREDICTED AND TRUE VALUES.IMPORTING NECESSARY LIBRARIES

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
mse = mean_squared_error(y_test,y_pred)
print("mean squarred error : ",mse)
```

```
mean squarred error :  2.4724180383370207e-06
```

Model Testing :

- MEAN_ABSOLUTE_ERROR : IN THIS CASE, WITH A VERY LOW MAE VALUE OF 0.000170, IT SUGGESTS THAT THE MODEL IS MAKING VERY ACCURATE PREDICTIONS, WITH AN AVERAGE ABSOLUTE DIFFERENCE OF APPROXIMATELY 0.000170 BETWEEN THE PREDICTED AND TRUE VALUES.

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
mae = mean_absolute_error(y_test,y_pred)
print("mean absolute error : ",mae)
```

```
mean absolute error :  0.00017047543640886 07
```

Model Testing :

- R2_SCORE : IN THIS CASE, WITH AN R2 SCORE OF 0.000460, IT SUGGESTS THAT THE MODEL FOUND ONLY SMALL AMOUNT OF THE VARIANCE IN THE TARGET VARIABLE.

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
r2 = r2_score(y_test,y_pred)
print("r2 score : ",r2)
```

```
r2 score :  0.00046074011679686 5
```

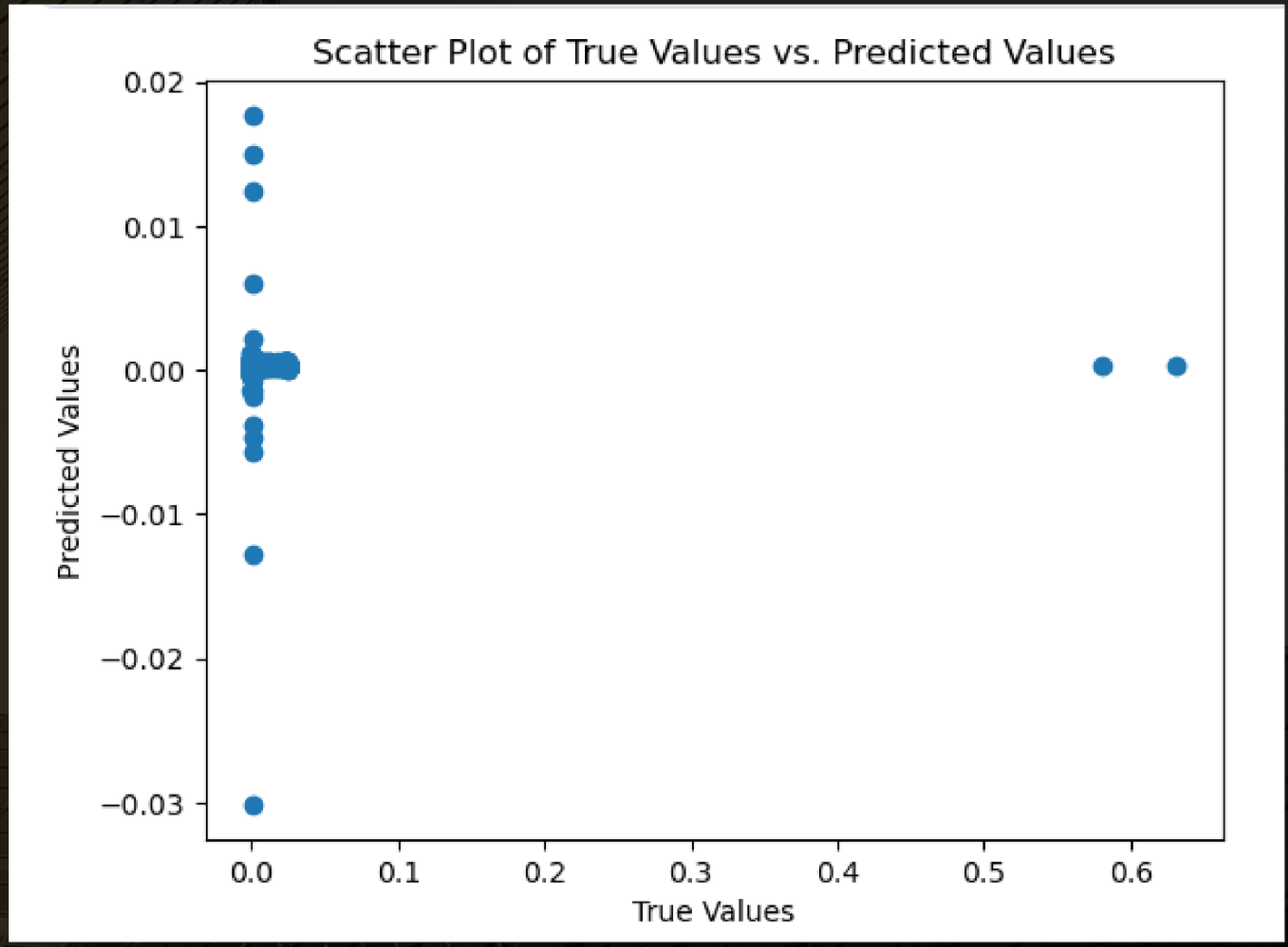
Data Analysis (EDA) :

scatter plot :

```
# SIDDHARTH UPADHYAY ; RGNO : 21BCE6103
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('Scatter Plot of True Values vs. Predicted Values')
plt.show()
```

Data Analysis (EDA) :

scatter plot :



Conclusion:

- from the above results we can conclude that our model has performed amazingly good in case of testing metrics
- The predicted values were close to the true values therefore its a good fit model.

Thank You

Let's connect !

NAME : SIDDHARTH UPADHYAY

REGISTRATION NO: 21BCE6103

siddharth.upadhyay2021@vitstudent.ac.in

7989660587

Vellore Institute of Technology(chennai) , 600127