CAPABL

SUMMER INTERNSHIP PROGRAM -DATA-SCIENCE

MACHINE LEARNING PROJECT REPORT
PREDICTING TAXI TRIP DURATION

Submitted by -

Tejas Prasad

STUDYING THE DATA SET

For our project we are provided with 2 datasets , 1 for training the model and another one for testing it. Hence we do not need to divide our datasets any further. Let us briefly analyse our datasets.

Training data

The training dataset contains 1,458,644 rows and 11 columns. Each row represents a taxi trip with information such as a unique identifier for the trip, the vendor or taxi company associated with the trip, the pickup and dropoff date and time, the number of passengers, longitude and latitude coordinates for the pickup and dropoff locations, a flag indicating if the trip data was stored before forwarding, and the duration of the trip in seconds.

Testing data

The testing dataset consists of 625,134 rows and 9 columns. Each row represents a taxi trip with information such as a unique identifier for the trip, the vendor or taxi company associated with the trip, the pickup date and time, the number of passengers, longitude and latitude coordinates for the pickup and dropoff locations, and a flag indicating if the trip data was stored before forwarding. The dataset includes numerical and categorical columns. The pickup_datetime column requires datetime processing for further analysis.

Both the datasets include both numerical and categorical columns, and some columns may require preprocessing, such as converting datetime objects, handling missing values, and potentially correcting data types. Before performing any analysis or building predictive models, it is essential to ensure data quality and conduct necessary data preprocessing steps.

IMPORTING THE NECESSARY LIBRARIES

Before I begin the project, i need to import the necessary libraries in python -

```
[59] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
     from sklearn.linear_model import LinearRegression
     from geopy.distance import geodesic
```

The libraries and modules ive imported here are -
1. `pandas` (imported as `pd`): A powerful data manipulation library that provides data structures and functions for working with structured data, like DataFrames.
2. `numpy` (imported as `np`): A fundamental library for numerical computations in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions.
3. `matplotlib.pyplot` (imported as `plt`): A popular plotting library used to create various visualizations, such as line plots, scatter plots, bar plots, and more. It works seamlessly with NumPy arrays and Pandas DataFrames.
4. `sklearn.metrics.mean_squared_error`: A function from scikit-learn (imported as `mean_squared_error`) used to calculate the mean squared error, a common metric for regression tasks, to evaluate the performance of a regression model by comparing the true target values with the predicted values.
5. `sklearn.metrics.mean_absolute_error`: A function from scikit-learn (imported as `mean_absolute_error`) used to calculate the mean absolute error, another regression metric, which measures the average absolute difference between the true and predicted values.
6. `sklearn.metrics.r2_score`: A function from scikit-learn (imported as `r2_score`) used to calculate the R-squared score (coefficient of determination), which indicates how well the regression model fits the observed data.
7. `sklearn.linear_model.LinearRegression`: A class from scikit-learn (imported as `LinearRegression`) used to implement linear regression models, one of the simplest and most widely used regression techniques for predicting continuous target variables based on the input features.
8. `geopy.distance.geodesic`: A function from the geopy library used for calculating geodesic distances between two points given their latitude and longitude coordinates. This can be helpful for calculating distances between pickup and dropoff locations in geographical applications.

INITIAL DATASET DESCRIPTION

Heres what the dataset initially looks like. I have used the pandas csv reader function to read the training dataset into ‘data’ and testing dataset into ‘data1’. I have also used the head function to print the first few rows of each feature -

```
[4]  data=pd.read_csv("train.csv")
     print(data.head())

            id  vendor_id      pickup_datetime     dropoff_datetime  \
   0  id2875421          2  2016-03-14 17:24:55  2016-03-14 17:32:30
   1  id2377394          1  2016-06-12 00:43:35  2016-06-12 00:54:38
   2  id3858529          2  2016-01-19 11:35:24  2016-01-19 12:10:48
   3  id3504673          2  2016-04-06 19:32:31  2016-04-06 19:39:40
   4  id2181028          2  2016-03-26 13:30:55  2016-03-26 13:38:10

      passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
   0              1.0        -73.982155        40.767937         -73.964630
   1              1.0        -73.980415        40.738564         -73.999481
   2              1.0        -73.979027        40.763939         -74.005333
   3              1.0        -74.010040        40.719971         -74.012268
   4              1.0        -73.973053        40.793209         -73.972923

      dropoff_latitude store_and_fwd_flag  trip_duration
   0         40.765602                  N          455.0
   1         40.731152                  N          663.0
   2         40.710087                  N         2124.0
   3         40.706718                  N          429.0
   4         40.782520                  N          435.0
```

```
[24] data1=pd.read_csv("test.csv")
     print(data.head())

            id  vendor_id      pickup_datetime     dropoff_datetime  \
   0  id2875421          2  2016-03-14 17:24:55  2016-03-14 17:32:30
   1  id2377394          1  2016-06-12 00:43:35  2016-06-12 00:54:38
   2  id3858529          2  2016-01-19 11:35:24  2016-01-19 12:10:48
   3  id3504673          2  2016-04-06 19:32:31  2016-04-06 19:39:40
   4  id2181028          2  2016-03-26 13:30:55  2016-03-26 13:38:10

      passenger_count  pickup_longitude  pickup_latitude  dropoff_longitude  \
   0              1.0        -73.982155        40.767937         -73.964630
   1              1.0        -73.980415        40.738564         -73.999481
   2              1.0        -73.979027        40.763939         -74.005333
   3              1.0        -74.010040        40.719971         -74.012268
   4              1.0        -73.973053        40.793209         -73.972923

      dropoff_latitude  trip_duration  distance_km
   0         40.765602          455.0     1.502172
   1         40.731152          663.0     1.808660
   2         40.710087         2124.0     6.379687
   3         40.706718          429.0     1.483632
   4         40.782520          435.0     1.187038
```

I have further used the info() function to get a more technical description of the datasets with information about each feature -

```
[9]  data.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 1311508 entries, 0 to 1311507
     Data columns (total 11 columns):
      #   Column              Non-Null Count    Dtype
     ---  ------              --------------    -----
      0   id                  1311508 non-null  object
      1   vendor_id           1311508 non-null  int64
      2   pickup_datetime     1311508 non-null  object
      3   dropoff_datetime    1311508 non-null  object
      4   passenger_count     1311508 non-null  float64
      5   pickup_longitude    1311508 non-null  float64
      6   pickup_latitude     1311508 non-null  float64
      7   dropoff_longitude   1311508 non-null  float64
      8   dropoff_latitude    1311508 non-null  float64
      9   store_and_fwd_flag  1311508 non-null  object
      10  trip_duration       1311508 non-null  float64
     dtypes: float64(6), int64(1), object(4)
     memory usage: 120.1+ MB
```

```
[31] data1.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 625134 entries, 0 to 625133
     Data columns (total 9 columns):
      #   Column              Non-Null Count   Dtype
     ---  ------              --------------   -----
      0   id                  625134 non-null  object
      1   vendor_id           625134 non-null  int64
      2   pickup_datetime     625134 non-null  object
      3   passenger_count     625134 non-null  int64
      4   pickup_longitude    625134 non-null  float64
      5   pickup_latitude     625134 non-null  float64
      6   dropoff_longitude   625134 non-null  float64
      7   dropoff_latitude    625134 non-null  float64
      8   store_and_fwd_flag  625134 non-null  object
     dtypes: float64(4), int64(2), object(3)
     memory usage: 42.9+ MB
```

DATA PRE-PROCESSING

In order to process the datasets given and make it more usable, i have taken multiple measures like checking for missing and NULL values, extracting new features from existing ones and so on. Let me walk you through the steps ive taken.

Checking for NULL and missing values

By using the isnull() function on both the testing and training datasets we can see if there are and null or missing values

```
[6]  data.isnull().sum()

     id                   0
     vendor_id            0
     pickup_datetime      0
     dropoff_datetime     0
     passenger_count      1
     pickup_longitude     1
     pickup_latitude      1
     dropoff_longitude    1
     dropoff_latitude     1
     store_and_fwd_flag   1
     trip_duration        1
     dtype: int64
```

```
[9]  data1.isnull().sum()

     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
```

Here we can see that our testing data has no null values but our training data does, so ill go ahead and remove them using the drop function

```
[10] data=data.dropna()
     data.isnull().sum()

     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
```

Now we can see that all the null and missing values have been removed.

Feature extraction

In our data set we have been given 4 features, The pickup and drop off latitudes and longitudes. Using these co-ordinates we can extract a more meaningful feature which will give is the distance of each of the rides. We can do this for both the datasets

```
[15] pickup_coords = list(zip(data['pickup_latitude'], data['pickup_longitude']))
     dropoff_coords = list(zip(data['dropoff_latitude'], data['dropoff_longitude']))

     data['distance_km'] = [geodesic(pickup, dropoff).kilometers for pickup, dropoff in zip(pickup_coords, dropoff_coords)]
```

```
[65] pickup_coords = list(zip(data1['pickup_latitude'], data1['pickup_longitude']))
     dropoff_coords = list(zip(data1['dropoff_latitude'], data1['dropoff_longitude']))

     data1['distance_km'] = [geodesic(pickup, dropoff).kilometers for pickup, dropoff in zip(pickup_coords, dropoff_coords)]
```

Dropping Obsolete Features

In our datasets we have a feature called “store_and_fwd_flag” which basically indicates if the details of a ride have been forwarded to the operator. For this feature, if most of the rides have a flag ‘Yes’ , then this could potentially have an impact on our model. However if most of the flags indicate ‘no’ then this feature could be useless to our model. I will go ahead and check the ratio of yes and no flags

```
[11] print(data['store_and_fwd_flag'].value_counts())

     N    1304277
     Y       7231
     Name: store_and_fwd_flag, dtype: int64
```

Here I can see that there are insignificantly less number of yes flags. Hence i will go ahead and drop this feature.

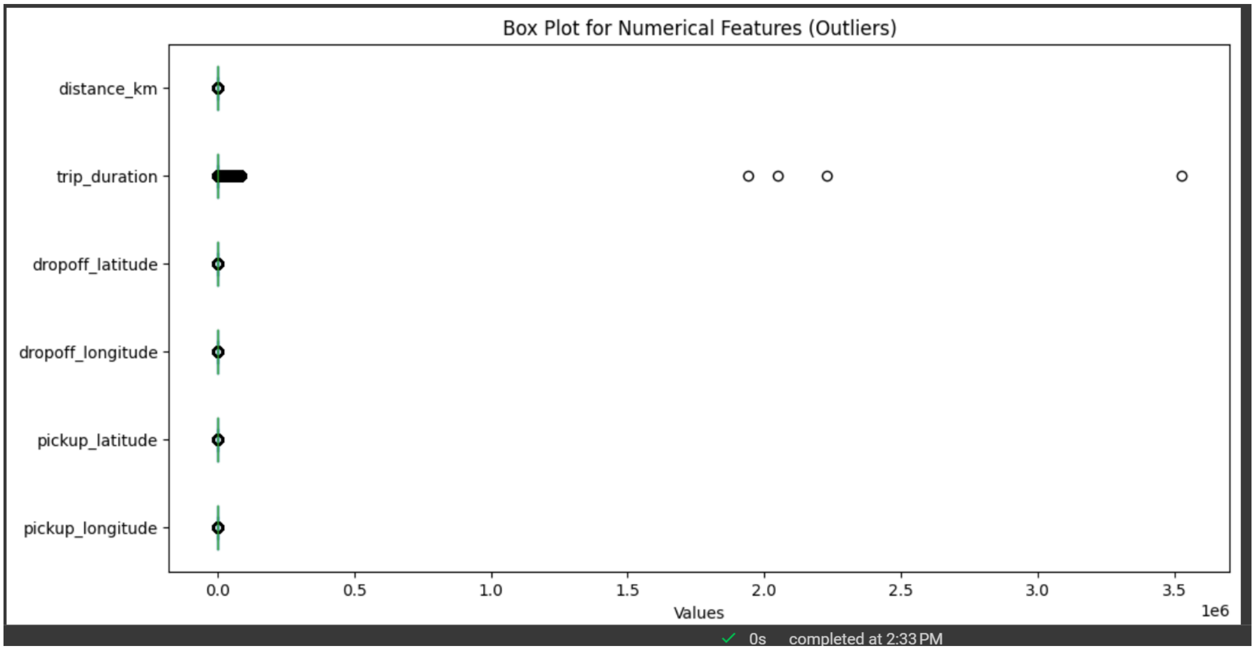
```
data.drop('store_and_fwd_flag', axis=1, inplace=True)
data1.drop('store_and_fwd_flag', axis=1, inplace=True)
```

Checking for outliers

I will now go ahead and check for any outliers in our dataset. For this ill use the matplot library to plot a boxplot for the given dataset

```
numerical_columns = ['pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'trip_duration', 'distance_km']
data3[numerical_columns].plot(kind='box', vert=False, figsize=(12, 6))
plt.title("Box Plot for Numerical Features (Outliers)")
plt.xlabel("Values")
plt.show()
```

I will get a graph that looks something like this



From this i can see that there arent any significant outliers so i can move on from here

Description of Pre-Processed data

After i have pre-processed my datasets it looks like this

```
data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1311508 entries, 0 to 1311507
Data columns (total 10 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   id                 1311508 non-null  object
 1   vendor_id          1311508 non-null  int64
 2   pickup_datetime    1311508 non-null  object
 3   dropoff_datetime   1311508 non-null  object
 4   passenger_count    1311508 non-null  float64
 5   pickup_longitude   1311508 non-null  float64
 6   pickup_latitude    1311508 non-null  float64
 7   dropoff_longitude  1311508 non-null  float64
 8   dropoff_latitude   1311508 non-null  float64
 9   distance_km        1311508 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 142.3+ MB
```

```
[67] data1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 625134 entries, 0 to 625133
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 625134 non-null  object
 1   vendor_id          625134 non-null  int64
 2   pickup_datetime    625134 non-null  object
 3   passenger_count    625134 non-null  int64
 4   pickup_longitude   625134 non-null  float64
 5   pickup_latitude    625134 non-null  float64
 6   dropoff_longitude  625134 non-null  float64
 7   dropoff_latitude   625134 non-null  float64
 8   distance_km        625134 non-null  float64
dtypes: float64(5), int64(2), object(2)
memory usage: 42.9+ MB
```

Here you can see that the data now has a feature ‘distance_km’ but does not have the ‘store_and_fwd_flag’. I would call this a success but I have made a mistake while dropping the features and accidently removed the time_duration feature from the training dataset which is very crucial to us. Hence i will create another variable data3 which has concatenation of data and the time_duration feature from the original dataset.

```
data3 = pd.concat([data3, data['time_duration']], axis=1)
```

The data3 looks something like this

```
[64]
    data3.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 1220005 entries, 0 to 1220004
    Data columns (total 11 columns):
     #   Column             Non-Null Count    Dtype
    ---  ------             --------------    -----
     0   id                 1220005 non-null  object
     1   vendor_id          1220005 non-null  int64
     2   pickup_datetime    1220005 non-null  object
     3   dropoff_datetime   1220005 non-null  object
     4   passenger_count    1220005 non-null  float64
     5   pickup_longitude   1220005 non-null  float64
     6   pickup_latitude    1220005 non-null  float64
     7   dropoff_longitude  1220005 non-null  float64
     8   dropoff_latitude   1220005 non-null  float64
     9   distance_km        1220005 non-null  float64
     10  trip_duration      1220005 non-null  float64
    dtypes: float64(7), int64(1), object(3)
    memory usage: 111.7+ MB
```

I will be using data3 as the training set for any further applications in my model

TRAINING AND TESTING THE MODEL

To implement a linear regression model i have used the LinearRegression module from sklearn.linear_regression. I have trained this model on my processed training data (‘data3’) and tested it on my processed testing data (‘data1’). The implementation looks like this

```
feature_cols = ['vendor_id', 'passenger_count', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'distance_km']
target_col = 'trip_duration'

# Drop rows with missing 'trip_duration' values in the training data
data3.dropna(subset=[target_col], inplace=True)

# Separate the features and target variable in training sets
X_train = data3[feature_cols]
y_train = data3[target_col]

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the testing data
X_test = data1[feature_cols]
y_pred = model.predict(X_test)



print(y_pred)
```

```
[ 795.95450985  934.01110493  643.95412827 ... 1393.78337065 2701.77169967
 1372.33865764]
```

Here, i have assigned all the feature columns to a variable featrue_cols and the target column which is trip_duration to the variable target_col
I have then trained the model by assigning the target column to the y axis and the feature column to the x axis. The model will then try to fit the best line for the data points by using multiple regression
After the im done training the model have tested it on the testing data and asked it to predict the trip_duration for the trips in the testing dataset. I the stored the predicted values in an array called y_pred.

RESULT AND DISCUSSION

After training my model on the training dataset, I am now able to get it to get it to predict the trip durations for the trips in the testing dataset. I have stored it in a array y_pred and when i print it i get something like this

```
[69] print(y_pred)

[ 795.95450985  934.01110493  643.95412827 ... 1393.78337065 2701.77169967
 1372.33865764]
```

Since there are too many values in the testing dataset, it will not display each and every prediction.
I am not able to check the accuracy of the models predictions here since there arent any values of the trip duration given in the dataset to compare to.
However, I can still check the accuracy by extracting a feature with all the trip durations for the testing data using the pick up time and drop off time in the dataset. However the accuracy measured here will still depend on the accuracy of the time values given so i will not be doing this
In conclusion, Our model in no working and can predict the trip_duration of any new trips given the input features