New York City Taxi fare Prediction

Spoorthy Reddy Jarugu

Vellore Institute of Technology - India

Deakin University

2022-07-03

Overview

Problem Definition

- Definition

Data Processing

- Loading Data
- Read Dataset and Print Dataset
- Cleaning and Checking Dataset
- Selecting Data

Extracing the data

- Data Visualization of year, month, hour, weekday
- Plotting the trip distance Vs fare amount
- Heat map for training set

Model built and prediction

- Evaluating the model

Conclusion

Problem Definition

Definition

Defn

Task is to predict the fare amount for a taxi ride in New York City. In table we have pickup and dropoff locations. We have to calculate the disatnce based on the date provided.

- The intersing characteristic is how to calculate the distance from latitude and longitude given.
- Depedign upon the calcuated distance (trip duration), taxi fare is predicted

Data Processing

Loading Data

- Load data - import data from csv files
  - Two types of dataset train and test dataset

Train dataset

- Named train dataset as df
- There are 55423856 rows and 8 coulmns in the train dataset

Test dataset

- Named test dataset as df_test
- There are 9915 rows and 7 coulmns in the test dataset.

Read Dataset and Print Dataset

- First step is to read the dataset from the CSV file
- Second print the both train and test Dataset
- Third check for NA values in the dataset

```
In [6]: #check for NA values in train set
        df.isnull().any()
        print(df.isnull().any())
```

```
key                False
fare_amount        False
pickup_datetime    False
pickup_longitude   False
pickup_latitude    False
dropoff_longitude   True
dropoff_latitude    True
passenger_count    False
dtype: bool
```

Figure 1: Listing missing vlaues in train dataset

```
In [7]: #check for NA values in test set
        df_test.isnull().any()
        print(df_test.isnull().any())
```

```
key                False
pickup_datetime    False
pickup_longitude   False
pickup_latitude    False
dropoff_longitude  False
dropoff_latitude   False
passenger_count    False
dtype: bool
```

Figure 2: Listing missing vlaues in test dataset

Cleaning and Checking Dataset

Identified that NAN values are present in dropoff_longitude and dropoff_latitude

- Removing the NAN values present in the train dataset by dropna command shown below and checking again for NAN values

In [8]:
```
#removing NA values
df=df.dropna(axis=0)
df.shape
```

Out[8]: (55423480, 8)

In [9]:
```
#after removing NA values check
df.isnull().any()
print(df.isnull().any())
```

```
key                  False
fare_amount          False
pickup_datetime      False
pickup_longitude     False
pickup_latitude      False
dropoff_longitude    False
dropoff_latitude     False
passenger_count      False
dtype: bool
```

Figure 3: Removing NAN vlaues

Cleaning Dataset further

- Removing the data where pickup and dropoff locations are same (i.e pickup_longitude and dropoff_longitude; pickup_latitude and dropoff_latitude).
- Checking for outliers by fixing the boundery of New York City
  - minimum_latitude is 40.573143,
  - minimum_langitude is -74.252193,
  - maximum_latitude is 41.709555,
  - maximum_langitude is -72.986532
- Removing outliers as they are identified

```
In [18]: df=df[~((df['pickup_latitude']<=boundary['min_lat']) | (df['pickup_latitude']>=boundary['max_lat']))]
         df=df[~((df['pickup_longitude']<=boundary['min_lang']) | (df['pickup_longitude']>=boundary['max_lang']))]

         df=df[~((df['dropoff_latitude']<=boundary['min_lat']) | (df['dropoff_latitude']>=boundary['max_lat']))]
         df=df[~((df['dropoff_longitude']<=boundary['min_lang']) | (df['dropoff_longitude']>=boundary['max_lang']))]

         df.shape

Out[18]: (53662868, 8)
```

Figure 4: Removing NAN vlaues

Selecting Data

- As dataset is huge randomly selecting 10% of dataset using the below process

```
In [19]:  #Randomly select 10% data

          df = df.sample(frac=0.1)
          df.shape

Out[19]:  (5366287, 8)
```

Figure 5: Selecting 10 percent of data

Data Visualization

- There are two paraters pickup - lattitude, longitude, dropoff - lattitude, longitude
- Let us scatter plot the above parameters as pickup data and dropoff data

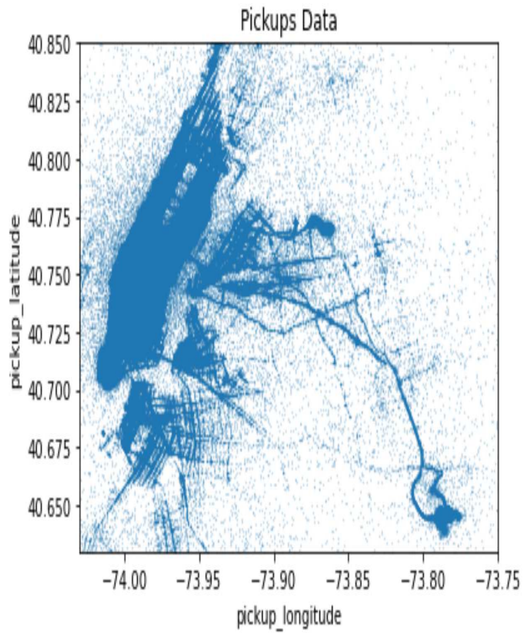

Figure 6: Pickup data

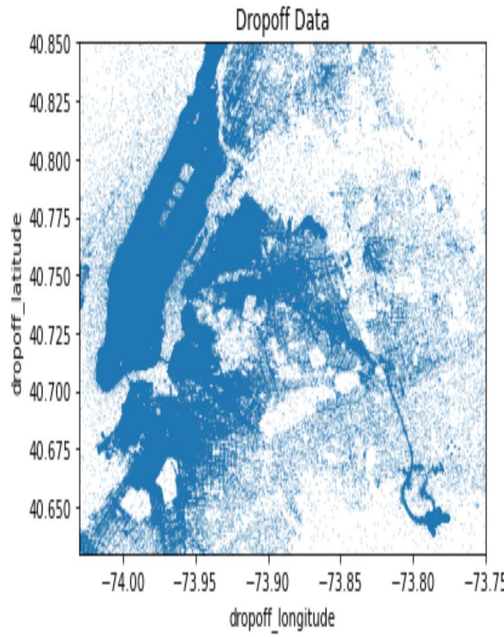

Figure 7: Dropoff data

Cleaning Passanger data

- Print the count for passengers

```
In [22]: df['passenger_count'].value_counts()
Out[22]: 1      3711947
         2       792877
         5       379206
         3       235947
         6       113793
         4       113539
         0        18972
         208          2
         9            2
         7            1
         8            1
         Name: passenger_count, dtype: int64
```

Figure 8: Passenger count

- Print the maximum and minimum value in passenger and cleanign the data for maximum count of 6 passengers per ride

```
In [23]: df['passenger_count']=df['passenger_count'].astype(int)
         print(df['passenger_count'].max())
         print(df['passenger_count'].min())

         208
         0

In [24]: df=df[~((df['passenger_count']>6) | (df['passenger_count'] == 0))]
         df.shape

Out[24]: (5347309, 8)
```

Figure 9: Cleaned passenger data

Passanger data visualization

- Print the count for passengers

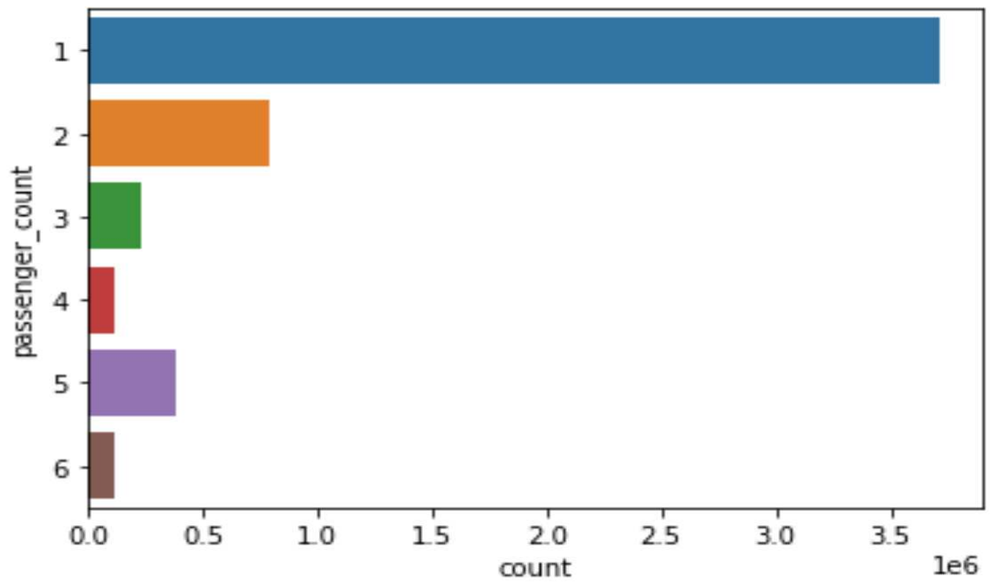

Figure 10: Visualizing passenger data

Extracing the data

Extracing - Day, Month, Year

- To predict the taxi fare accurately we are extracting the
  - Hour is calcuted to find weather its mid_night_trip or rush_hour_trip is noted
  - Day on which the passanger is picked upon
  - Month of trip
  - Year of travel

  from the pickup_datetime coulmns
- From the pickup_month weather its snow_season or not is noted
- Finally trip_diatance is calculated from pickup_latitude, pickup_longitude, dropoff_latitude, dropoff_longitude and stored it in trip_distance

```
In [35]:
from geopy.distance import geodesic

def distance_calculate(lat,long,drop_lat,drop_long):
    newport_ri = (lat,long)
    cleveland_oh = (drop_lat,drop_long)
    dist=geodesic(newport_ri, cleveland_oh).km
    return dist
```

```
In [36]:
df['trip_distance']=list(map(distance_calculate,df['pickup_latitude'],df['pickup_longitude'],
                             df['dropoff_latitude'],df['dropoff_longitude']))
df.head()
```

Figure 11: Distance Calculation

Data Visualization of year, month, hour, weekday

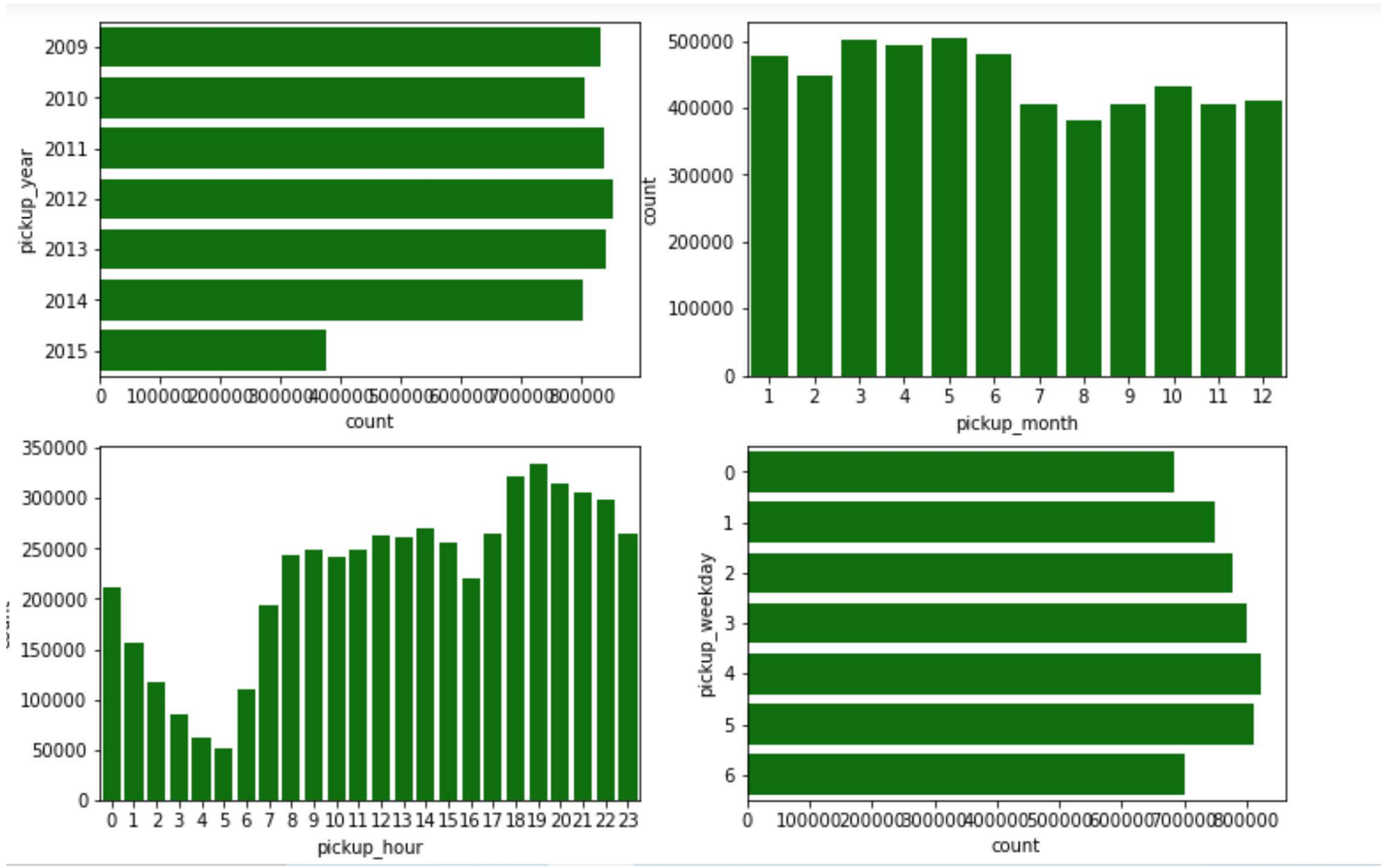

Figure 12: Visualizing year, month, hour, weekday count

Plotting the trip distance Vs fare amount

In [38]:
```
plt.figure(figsize=(8,8))

df.plot(x='fare_amount',y='trip_distance',kind='scatter')
```

Out[38]:
```
<AxesSubplot:xlabel='fare_amount', ylabel='trip_distance'>

<Figure size 576x576 with 0 Axes>
```

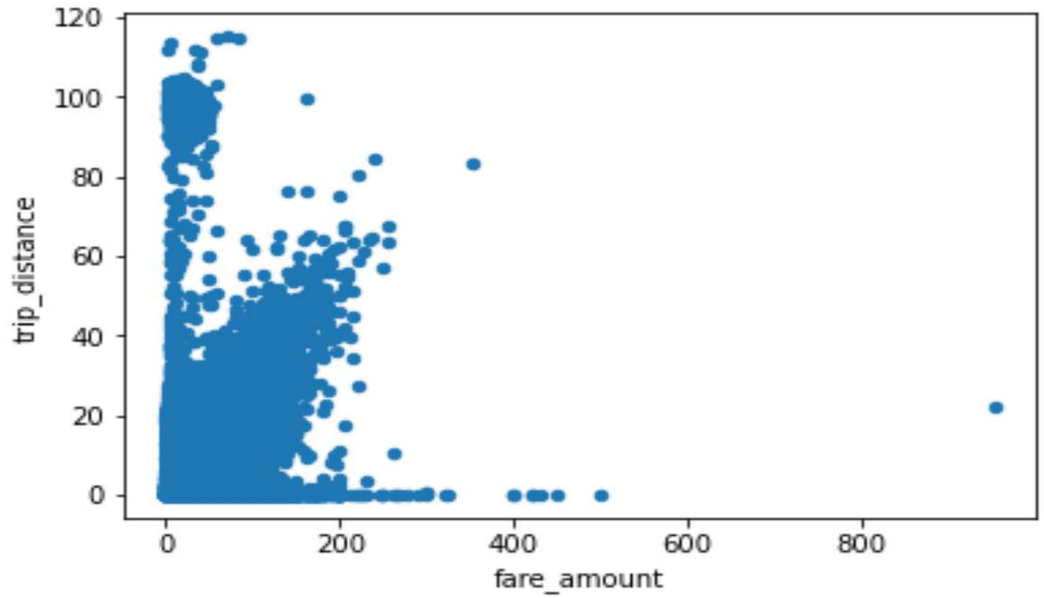

Figure 13: Visualizing trip_distance and fare_amount

Heat map for training set

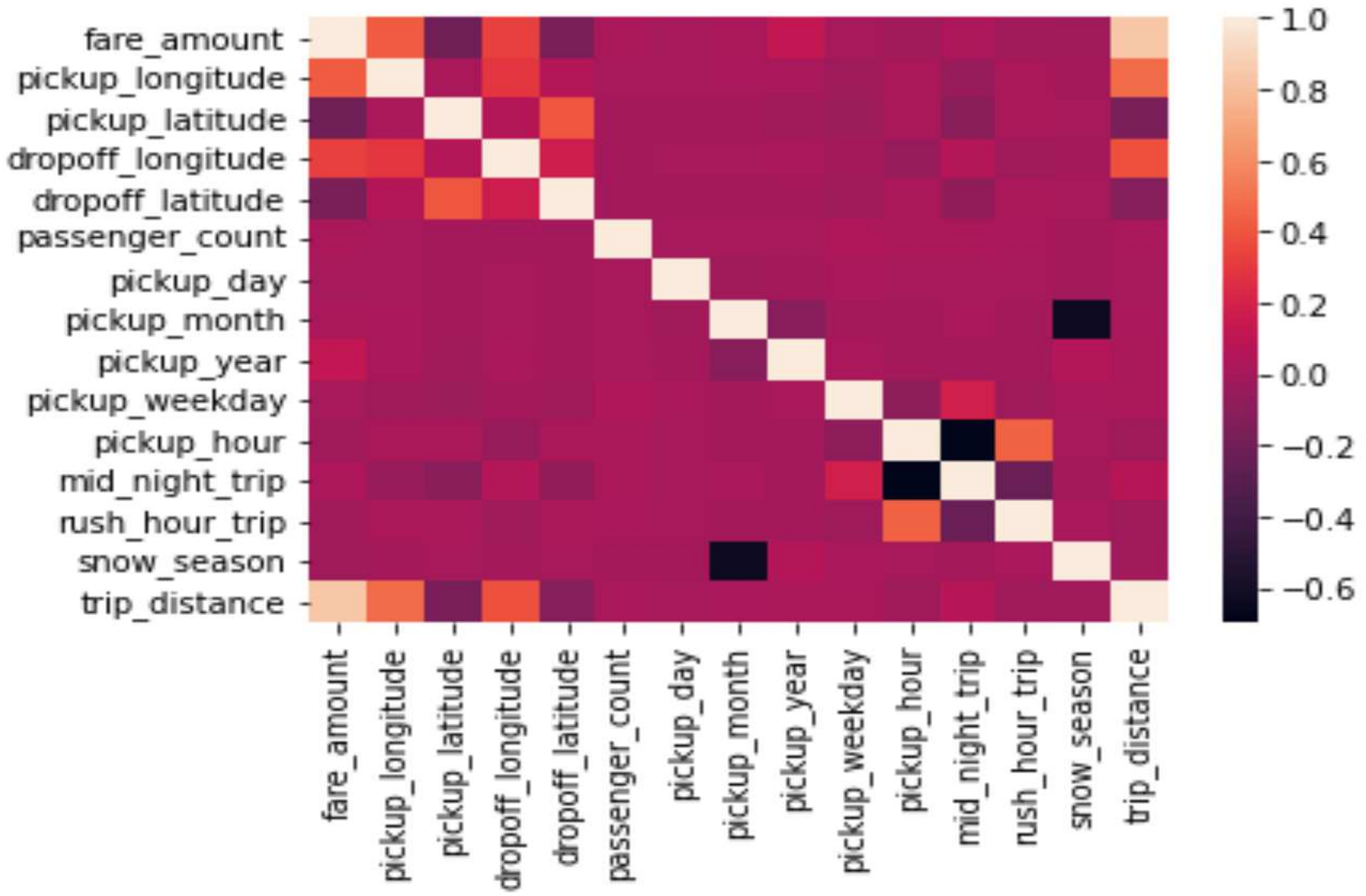

Figure 14: Visualizing heatmap

Model built and prediction

Linear Regression model


```
In [42]: X=df.drop(columns=['key','fare_amount'])
         y=df['fare_amount']
```

Linear Regression

```
In [43]: from sklearn.model_selection import train_test_split
```

```
In [44]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=101)
```

```
In [45]: from sklearn.linear_model import LinearRegression
```

```
In [46]: lm = LinearRegression()
```

```
In [47]: lm.fit(X_train,y_train)
Out[47]: LinearRegression()
```

Figure 15: Linear Regression

- Bulit a Linear Regression model predict the fare_amount of the trip in New York city

Evaluating the model

In [58]:
```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test,prediction))
print('MSE:', metrics.mean_squared_error(y_test,prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE: 2.4184007631012028
MSE: 26.256078544064415
RMSE: 5.124068553802185
```

In [59]:
```
lm.score(X_test,y_test)
```

Out[59]: 0.7109492969947017

Figure 16: Evaluation Score

Visualization of predicted data


```
In [51]: plt.scatter(y_test,prediction)

Out[51]: <matplotlib.collections.PathCollection at 0x1473954db80>
```

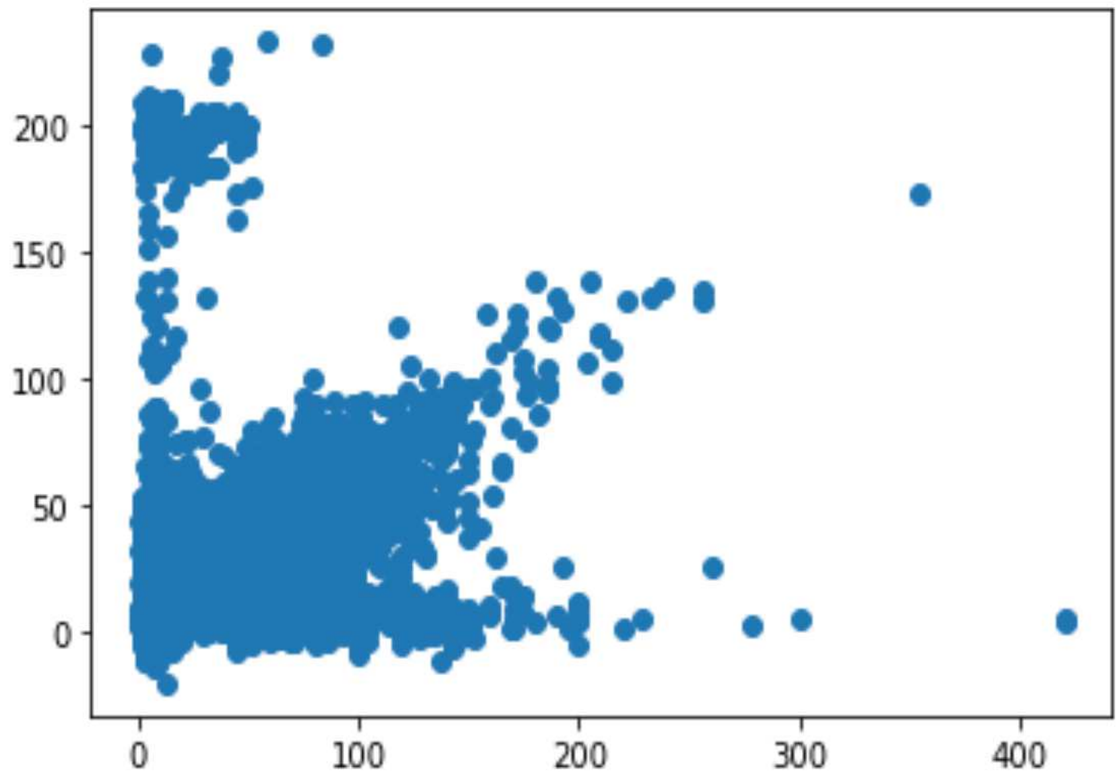

Figure 17: Visualizing predicted data

Final test data prediction


```
In [63]: df_test
```

Out[63]:

	passenger_count	mid_night_trip	rush_hour_trip	snow_season	trip_distance	fare_price
0	1	0	0	1	2.320991	8.973470
1	1	0	0	1	2.423802	9.193202
2	1	0	0	0	0.618182	5.599607
3	1	0	0	0	1.959671	8.466715
4	1	0	0	0	5.382833	15.782896
...	...	...	...	...	...	...
9909	6	0	0	0	2.124110	9.056012
9910	6	0	1	1	3.268511	11.188403
9911	6	0	1	0	19.217032	45.539987
9912	6	1	0	1	8.339644	21.200150
9913	6	0	0	1	1.182767	6.778640

9914 rows × 6 columns

Figure 18: Test data prediction

Conclusion

Conclusion

- Fare prediction using latitude and longitude infomration is showcased.
- Additionally mid_night_trip, Rush_hour_trip, show_season parameters are also considerd in fare calculation.
- The prediction model helps both passangers and drivers for effctive fare prediction compared to conventional prediction

Questions?

Contact Information

Spoorthy Reddy Jarugu
Vellore Institute of Technology, India
Deakin University, Australia

JARUGUSPOORTHYREDDY@GMAIL.COM

TEAM FOR UNIVERSAL LEARNING AND INTELLIGENT PROCESSING