Uber_Random_forest

November 7, 2023

[2]:
```
import pandas as pd
import numpy as np
```

[3]:
```
df = pd.read_csv(r"C:\Users\lalit\OneDrive\Desktop\practicals\machine␣
↪learning\practical1\uber.csv")
```

[4]:
```
df.head()
```

[4]:
```
   Unnamed: 0                            key  fare_amount
0    24238194    2015-05-07 19:52:06.0000003          7.5  \
1    27835199    2009-07-17 20:04:56.0000002          7.7
2    44984355   2009-08-24 21:45:00.00000061         12.9
3    25894730    2009-06-26 08:22:21.0000001          5.3
4    17610152  2014-08-28 17:47:00.000000188         16.0

           pickup_datetime  pickup_longitude  pickup_latitude
0  2015-05-07 19:52:06 UTC        -73.999817        40.738354  \
1  2009-07-17 20:04:56 UTC        -73.994355        40.728225
2  2009-08-24 21:45:00 UTC        -74.005043        40.740770
3  2009-06-26 08:22:21 UTC        -73.976124        40.790844
4  2014-08-28 17:47:00 UTC        -73.925023        40.744085

   dropoff_longitude  dropoff_latitude  passenger_count
0         -73.999512         40.723217                1
1         -73.994710         40.750325                1
2         -73.962565         40.772647                1
3         -73.965316         40.803349                3
4         -73.973082         40.761247                5
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Unnamed: 0         200000 non-null  int64
 1   key                200000 non-null  object
```

```
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

[6]:
```
df= df.drop(['Unnamed: 0', 'key'], axis= 1) #To drop unnamed column as it isn't␣
 ↪required
```

[8]:
```
df.head(50)
```

[8]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	\
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
5	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	
6	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	
7	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	
8	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.745767	
9	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.741787	
10	6.5	2015-05-22 17:32:27 UTC	-73.974388	40.746952	
11	8.5	2011-05-23 22:15:00 UTC	0.000000	0.000000	
12	3.3	2011-05-17 14:03:00 UTC	-73.966378	40.804440	
13	10.9	2011-06-25 11:19:00 UTC	-73.953352	40.767382	
14	6.9	2010-04-06 22:20:27 UTC	-73.973370	40.755193	
15	9.7	2012-02-21 09:33:00 UTC	-73.990718	40.751920	
16	4.9	2011-09-01 09:21:40 UTC	-73.988908	40.756982	
17	10.5	2011-03-19 23:58:27 UTC	-74.005665	40.741138	
18	12.0	2015-03-25 08:58:35 UTC	-73.962532	40.767189	
19	4.9	2009-08-08 00:20:00 UTC	-73.992075	40.719633	
20	10.5	2014-02-18 14:26:00 UTC	-73.980022	40.745990	
21	5.0	2015-03-03 23:15:03 UTC	-73.989189	40.729141	
22	4.1	2009-11-26 02:58:00 UTC	-74.010798	40.726085	
23	7.7	2010-09-04 16:12:00 UTC	-73.994300	40.739512	
24	12.9	2010-05-12 22:32:00 UTC	-73.972987	40.764040	
25	9.5	2009-02-12 17:52:18 UTC	-73.986059	40.757159	
26	5.0	2014-01-21 06:55:00 UTC	-73.957802	40.776372	
27	12.0	2012-11-21 17:37:19 UTC	-73.993909	40.741551	
28	4.9	2009-05-06 20:06:23 UTC	-73.977780	40.763656	
29	7.3	2011-12-24 02:52:00 UTC	-73.971075	40.787833	
30	25.7	2011-05-21 09:00:00 UTC	-73.944815	40.834367	

31	7.7	2009-02-28 15:54:57 UTC	-74.004184	40.712975
32	10.5	2013-02-11 19:09:00 UTC	-73.982085	40.771387
33	11.0	2013-09-10 20:50:25 UTC	-73.991186	40.741393
34	39.5	2014-06-04 06:49:00 UTC	-73.788080	40.642187
35	8.1	2009-06-05 05:35:00 UTC	-73.988690	40.751273
36	5.7	2011-02-19 16:31:00 UTC	-74.010863	40.710813
37	6.9	2011-08-31 19:47:00 UTC	-73.968697	40.754998
38	7.7	2010-05-18 21:28:00 UTC	-73.968370	40.759005
39	29.0	2014-02-13 17:57:00 UTC	-73.992600	40.753172
40	15.7	2010-04-01 14:42:00 UTC	-73.973360	40.786952
41	9.0	2014-04-02 14:58:32 UTC	-73.970164	40.765016
42	4.9	2011-02-01 15:25:03 UTC	-73.987139	40.753038
43	5.4	2009-01-10 22:43:36 UTC	-73.994222	40.751229
44	3.3	2012-07-12 00:59:02 UTC	-73.982371	40.765501
45	8.9	2009-02-19 08:28:42 UTC	-73.977137	40.779272
46	17.0	2014-01-16 14:58:09 UTC	-73.993900	40.751714
47	12.0	2015-01-04 09:17:47 UTC	-73.979523	40.727310
48	56.8	2013-01-03 22:24:41 UTC	-73.993498	40.764686
49	13.5	2013-05-23 10:57:00 UTC	-73.962043	40.805372

	dropoff_longitude	dropoff_latitude	passenger_count
0	-73.999512	40.723217	1
1	-73.994710	40.750325	1
2	-73.962565	40.772647	1
3	-73.965316	40.803349	3
4	-73.973082	40.761247	5
5	-73.969019	40.755910	1
6	-73.871195	40.774297	5
7	0.000000	0.000000	1
8	-74.002720	40.743537	1
9	-73.963040	40.775012	1
10	-73.988586	40.729805	1
11	0.000000	0.000000	1
12	-73.965890	40.807133	5
13	-73.972510	40.796137	1
14	-73.978265	40.766375	1
15	-73.973053	40.744230	2
16	-73.981246	40.760050	1
17	-73.977830	40.749338	2
18	-73.974457	40.753860	1
19	-73.985323	40.727405	1
20	-74.003432	40.759667	1
21	-73.987282	40.720634	2
22	-74.009767	40.737402	2
23	-73.988070	40.724482	2
24	-74.007820	40.714993	5
25	-73.976130	40.780589	1

```
26          -73.957422          40.782870                1
27          -73.987874          40.770232                1
28          -73.989486          40.772966                1
29          -73.944957          40.784302                5
30          -73.989332          40.721920                4
31          -73.988385          40.725102                1
32          -73.959610          40.769445                1
33          -74.009710          40.716455                2
34          -73.865042          40.725997                4
35          -73.958810          40.764643                3
36          -74.001205          40.718745                1
37          -73.987302          40.735940                1
38          -73.979105          40.777570                3
39          -73.908508          40.816192                1
40          -73.972810          40.743548                1
41          -73.962801          40.774780                1
42          -73.977794          40.751834                1
43          -73.998331          40.744342                1
44          -73.985248          40.762808                1
45          -73.970457          40.757972                1
46          -73.958575          40.760390                1
47          -73.984879          40.760651                1
48          -73.993498          40.764686                1
49          -73.988447          40.770140                1
```

[9]:
```
df.shape
```

[9]:
```
(200000, 7)
```

[10]:
```
df.dtypes
```

[10]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

[11]:
```
#Removing the null values
df.isnull().sum()
```

[11]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
```

```
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

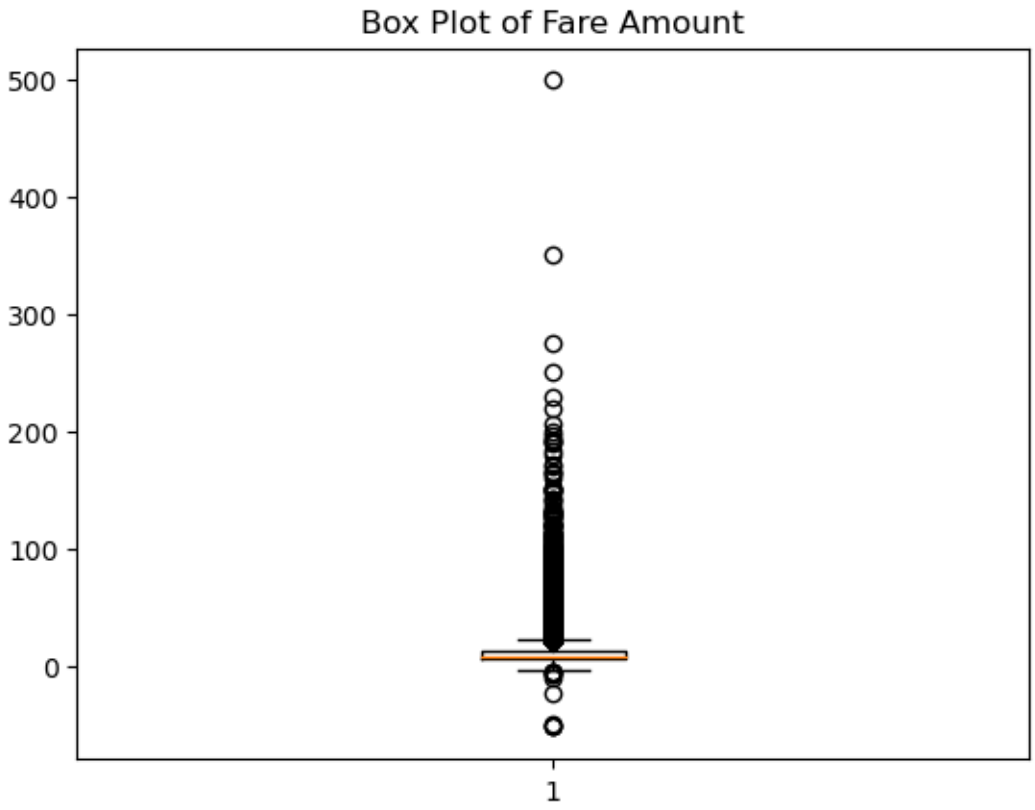
[12]:
```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(),inplace= True)
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].median(),inplace=
 ↪True)
```

[13]:
```
df.isnull().sum()
```

[13]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

[14]:
```
import matplotlib.pyplot as plt
```

[16]:
```
plt.boxplot(df['fare_amount'])
plt.title("Box Plot of Fare Amount")
plt.show()
```



[19]:
```
# Use IQR method to detect and remove outliers
Q1 = df['fare_amount'].quantile(0.25)
Q3 = df['fare_amount'].quantile(0.75)
IQR = Q3 - Q1
data = df[~((df['fare_amount'] < (Q1 - 1.5 * IQR)) | (df['fare_amount'] > (Q3 +␣
↪1.5 * IQR)))]  # Remove outliers
```

[32]:
```
X=␣
↪df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_
↪]]
y= df['fare_amount']
```

[33]:
```
# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[34]:
```
# Train a Linear Regression model
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
lr_model = LinearRegression()
```

```
lr_model.fit(X_train, y_train)
```

[34]:
```
LinearRegression()
```

[35]:
```
# Train a Random Forest Regression model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

[35]:
```
RandomForestRegressor(random_state=42)
```

[38]:
```
# 5. Evaluate the models and compare their respective scores
# Make predictions
lr_predictions = lr_model.predict(X_test)
rf_predictions = rf_model.predict(X_test)
```

[39]:
```
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
```

[40]:
```
# Calculate R-squared (R2)
lr_r2 = r2_score(y_test, lr_predictions)
rf_r2 = r2_score(y_test, rf_predictions)
```

[41]:
```
# Calculate Root Mean Squared Error (RMSE)
lr_rmse = sqrt(mean_squared_error(y_test, lr_predictions))
rf_rmse = sqrt(mean_squared_error(y_test, rf_predictions))
```

[42]:
```
print("Linear Regression R2:", lr_r2)
print("Random Forest Regression R2:", rf_r2)
print("Linear Regression RMSE:", lr_rmse)
print("Random Forest Regression RMSE:", rf_rmse)
```

```
Linear Regression R2: 0.00032785007561042523
Random Forest Regression R2: 0.6991200520641053
Linear Regression RMSE: 10.30951786467544
Random Forest Regression RMSE: 5.655958074905309
```

[ ]:
```

```