```
Name:: Gayatri Manikchand Deshmukh
Class :: BE
Division :: A
Roll No. :: 38
```

Problem Statement

Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks:

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression and random forest regression models.
5. Evaluate the models and compare their respective scores like R2, RMSE, etc. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares- (https://www.kaggle.com/datasets/yasserh/uber-fares-) dataset

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("uber.csv")
```

In [3]:

```
df.shape
```

Out[3]:

```
(200000, 9)
```

In [4]:

```
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
```

In [5]:

```
df.head()
```

Out[5]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitu
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.7383
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.7282
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.7407
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.7908
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.7440

In [6]:

```
df.tail()
```

Out[6]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_l
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.

In [7]:

```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [9]:

```
df.dtypes
```

Out[9]:

```
key                 object
fare_amount        float64
pickup_datetime     object
pickup_longitude   float64
pickup_latitude    float64
dropoff_longitude  float64
dropoff_latitude   float64
passenger_count      int64
dtype: object
```

In [10]:

```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])
```

In [11]:

```
df.dtypes
```

Out[11]:

```
key                               object
fare_amount                      float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                 float64
pickup_latitude                  float64
dropoff_longitude                float64
dropoff_latitude                 float64
passenger_count                    int64
dtype: object
```

In [12]:

```
df.isnull().sum()
```

Out[12]:

```
key                 0
fare_amount         0
pickup_datetime     0
pickup_longitude    0
pickup_latitude     0
dropoff_longitude   1
dropoff_latitude    1
passenger_count     0
dtype: int64
```

In [13]:

```
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].mean(),inplace=True)
```

In [14]:

```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].median(),inplace=True)
```

In [15]:

```
df.isnull().sum()
```

Out[15]:

```
key                  0
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [16]:

```
df.head()
```

Out[16]:

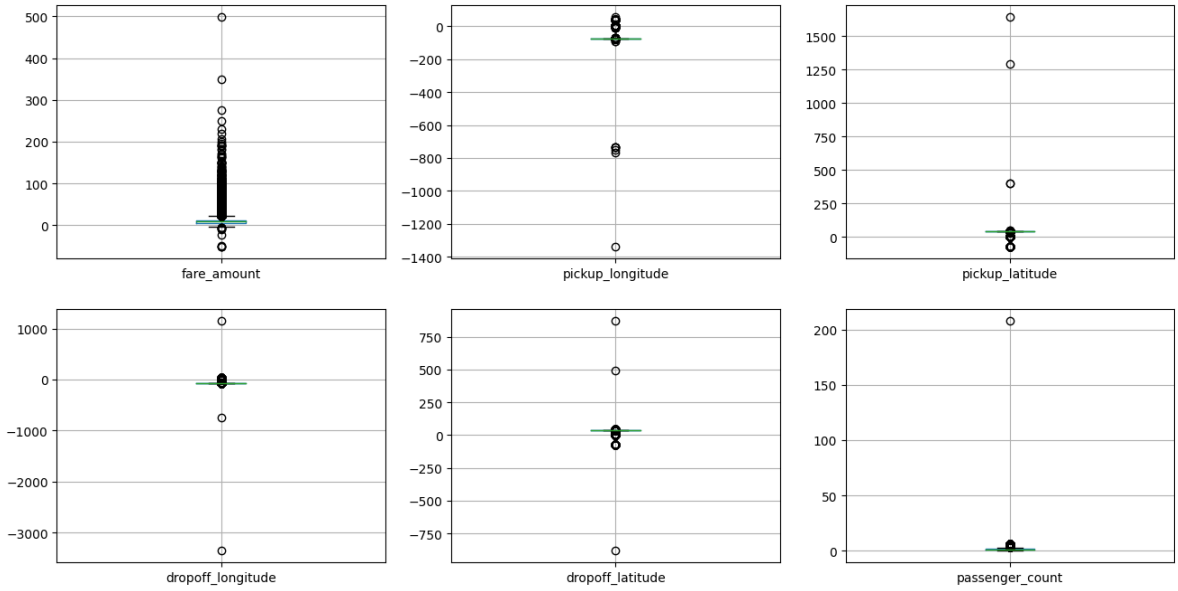
	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_
0	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-
1	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-
2	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-
3	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-
4	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-

In [17]:

```
fig, axes = plt.subplots(2, 3, figsize=(16,8))
df.boxplot(column="fare_amount", ax=axes[0,0])
df.boxplot(column="pickup_longitude", ax=axes[0,1])
df.boxplot(column="pickup_latitude", ax=axes[0,2])
df.boxplot(column="dropoff_longitude", ax=axes[1,0])
df.boxplot(column="dropoff_latitude", ax=axes[1,1])
df.boxplot(column="passenger_count", ax=axes[1,2])
```

Out[17]:

```
<AxesSubplot:>
```



In [18]:

```
df['fare_amount'] = df['fare_amount'].fillna(0)
Q1 = np.quantile(df['fare_amount'],0.25)
Q3 = np.quantile(df['fare_amount'],0.75)
IQR = Q3-Q1
print("Q1",Q1)
print("Q3",Q3)
print("IQR = ",IQR)
upper = Q3 + IQR*1.5
lower = Q1 - IQR*1.5
print("Upper Quartile :: ",upper)
print("Lower Quartile :: ",lower)
```

```
Q1 6.0
Q3 12.5
IQR =  6.5
Upper Quartile ::  22.25
Lower Quartile ::  -3.75
```

In [19]:

```
outlier = []
for i in df['fare_amount']:
    if((i>upper) or (i<lower)):
        outlier.append(i)
print('Outliers are ',outlier)
```

```
Outliers are  [24.5, 25.7, 39.5, 29.0, 56.8, 26.1, 49.57, 30.9, 26.9, 43.
0, 35.3, 38.54, 29.0, 24.0, 23.0, 45.0, 29.5, 23.7, 24.0, 49.8, 24.0, 23.
0, 34.25, 39.33, 45.0, 35.7, 29.3, 49.8, 43.0, 57.33, 37.5, 37.47, 49.57,
29.7, 33.7, 23.7, 25.7, 36.0, 25.7, 57.33, 49.8, 22.5, 38.83, 57.33, 45.0,
22.5, 26.33, 39.5, 26.1, 25.3, 49.8, 31.8, 26.1, 49.57, 33.3, 49.8, 25.07,
40.5, 43.0, 52.0, 27.0, 52.0, 30.83, 35.33, 93.16, 27.0, 40.33, 24.5, 32.
9, 34.0, 26.0, 23.0, 23.0, 31.83, 22.5, 29.8, 69.25, 26.1, 57.33, 25.0, 5
1.5, 35.3, 39.33, 23.0, 57.33, 41.5, 52.0, 45.0, 42.8, 40.3, 23.5, 23.0, 2
8.5, 25.7, 25.3, 36.8, 36.8, 33.5, 41.83, 43.7, 22.5, 27.6, 29.7, 46.1, 2
3.5, 33.33, 31.07, 28.9, 45.33, 24.5, 37.5, 33.3, 49.83, 29.47, 52.0, 22.
5, 40.54, 52.0, 45.33, 25.0, 32.83, 45.0, 57.33, 50.0, 29.0, 26.1, 45.0, 5
7.54, 24.0, 41.33, 22.5, 27.5, 31.33, 25.3, 23.5, 45.0, 27.5, 56.0, 27.0,
45.0, 45.0, 33.47, 52.0, 23.5, 27.5, 49.8, 49.8, 34.5, 45.0, 34.5, 34.83,
49.57, 24.0, 26.33, 27.0, 73.0, 32.33, 45.0, 57.33, 52.08, 28.5, 57.33, 2
9.3, 57.33, 23.0, 25.87, 38.94, 37.83, 48.04, 38.83, 49.8, 113.66, 49.57,
35.0, 24.1, 31.33, 38.9, 44.0, 48.83, 55.0, 52.33, 49.8, 57.33, 39.0, 28.6
7, 33.33, 45.0, 32.1, 32.0, 25.3, 40.9, 26.5, 26.1, 32.5, 38.33, 57.33, 4
8.5, 41.83, 43.3, 45.0, 57.33, 25.5, 42.83, 49.8, 22.67, 23.0, 32.5, 37.7,
31.0, 35.33, 52.0, 24.1, 34.83, 28.8, 57.33, 24.67, 27.3, 24.5, 24.1, 57.3
```

In [20]:

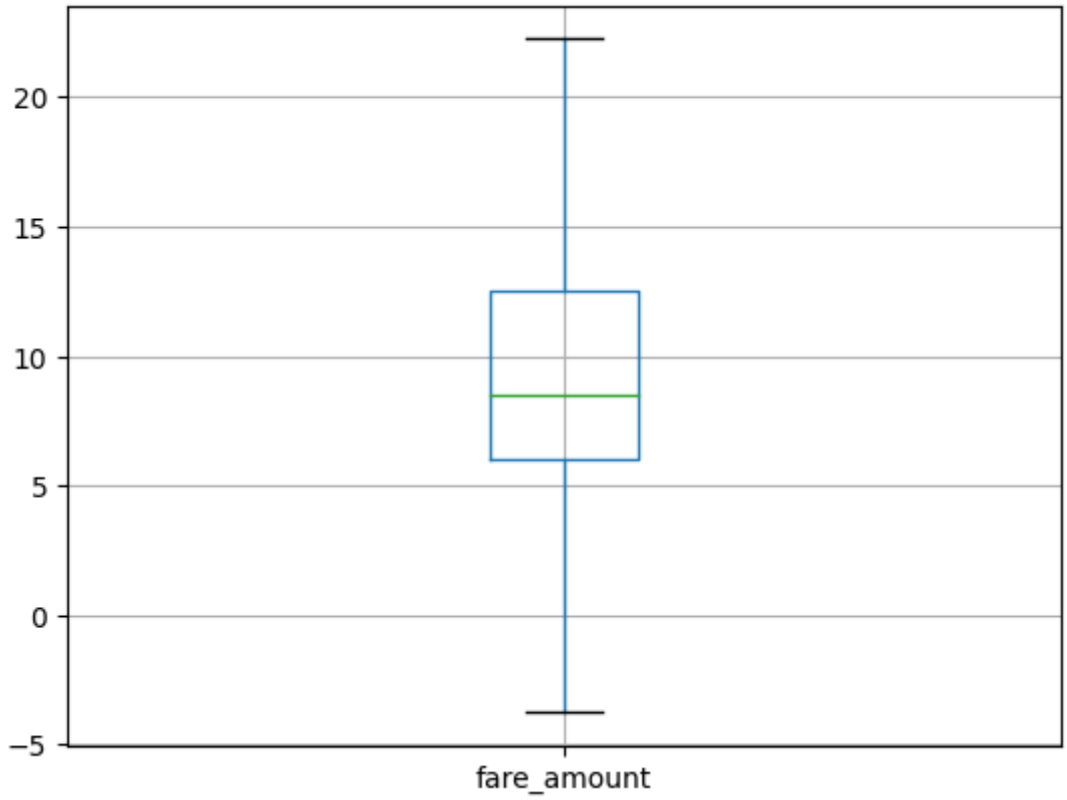
```
df['fare_amount']=np.where(df['fare_amount'] < lower, lower, df['fare_amount'])
df['fare_amount']=np.where(df['fare_amount'] > upper, upper, df['fare_amount'])
```

In [21]:

```
df.boxplot(column = 'fare_amount')
```

Out[21]:

<AxesSubplot:>



In [22]:

```
df['pickup_longitude'] = df['pickup_longitude'].fillna(0)
Q1 = np.quantile(df['pickup_longitude'],0.25)
Q3 = np.quantile(df['pickup_longitude'],0.75)
IQR = Q3-Q1
print("Q1",Q1)
print("Q3",Q3)
print("IQR = ",IQR)
upper = Q3 + IQR*1.5
lower = Q1 - IQR*1.5
print("Upper Quartile :: ",upper)
print("Lower Quartile :: ",lower)
```

```
Q1 -73.992065
Q3 -73.96715350000001
IQR =  0.02491149999998754
Upper Quartile ::  -73.92978625000003
Lower Quartile ::  -74.02943224999999
```

In [23]:

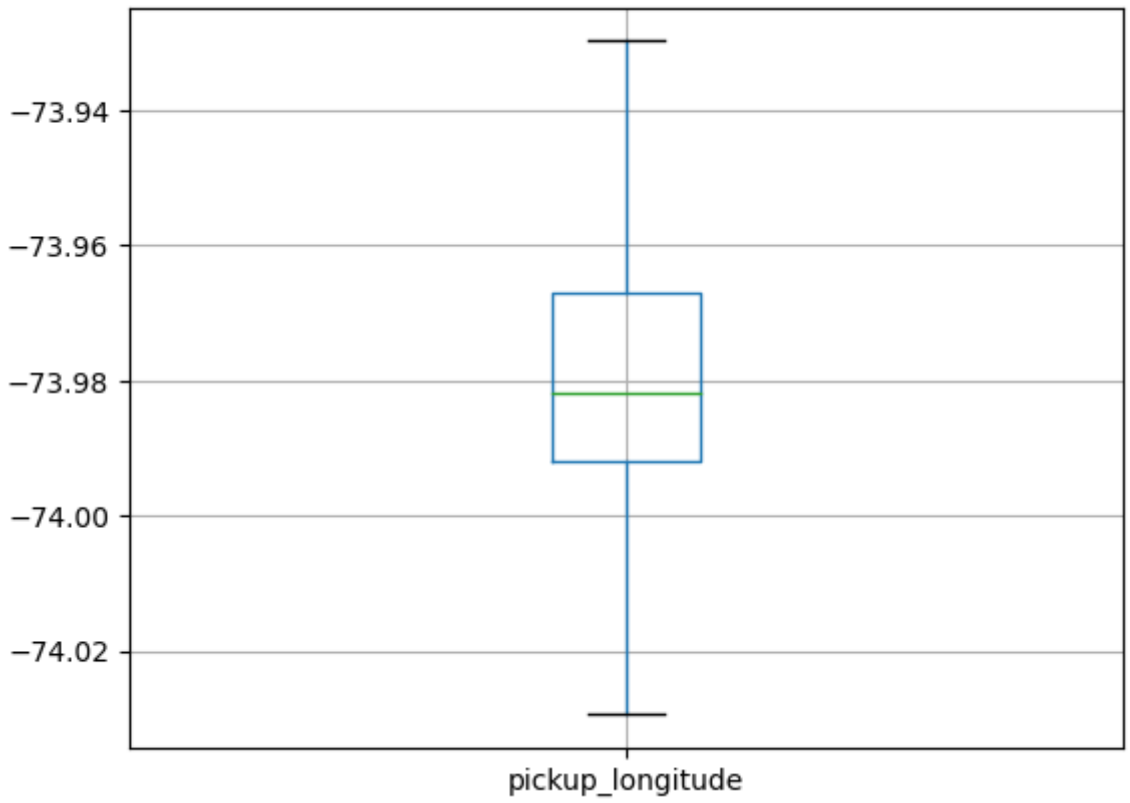
```
df['pickup_longitude']=np.where(df['pickup_longitude'] < lower, lower, df['pickup_longitude
df['pickup_longitude']=np.where(df['pickup_longitude'] > upper, upper, df['pickup_longitude
```

In [24]:

```
df.boxplot(column="pickup_longitude")
```

Out[24]:

```
<AxesSubplot:>
```



In [25]:

```
df['pickup_latitude'] = df['pickup_latitude'].fillna(0)
Q1 = np.quantile(df['pickup_latitude'],0.25)
Q3 = np.quantile(df['pickup_latitude'],0.75)
IQR = Q3-Q1
print("Q1",Q1)
print("Q3",Q3)
print("IQR = ",IQR)
upper = Q3 + IQR*1.5
lower = Q1 - IQR*1.5
print("Upper Quartile :: ",upper)
print("Lower Quartile :: ",lower)
```

```
Q1 40.73479575
Q3 40.767158
IQR =  0.03236224999999848
Upper Quartile ::  40.815701375
Lower Quartile ::  40.68625237500001
```

In [26]:

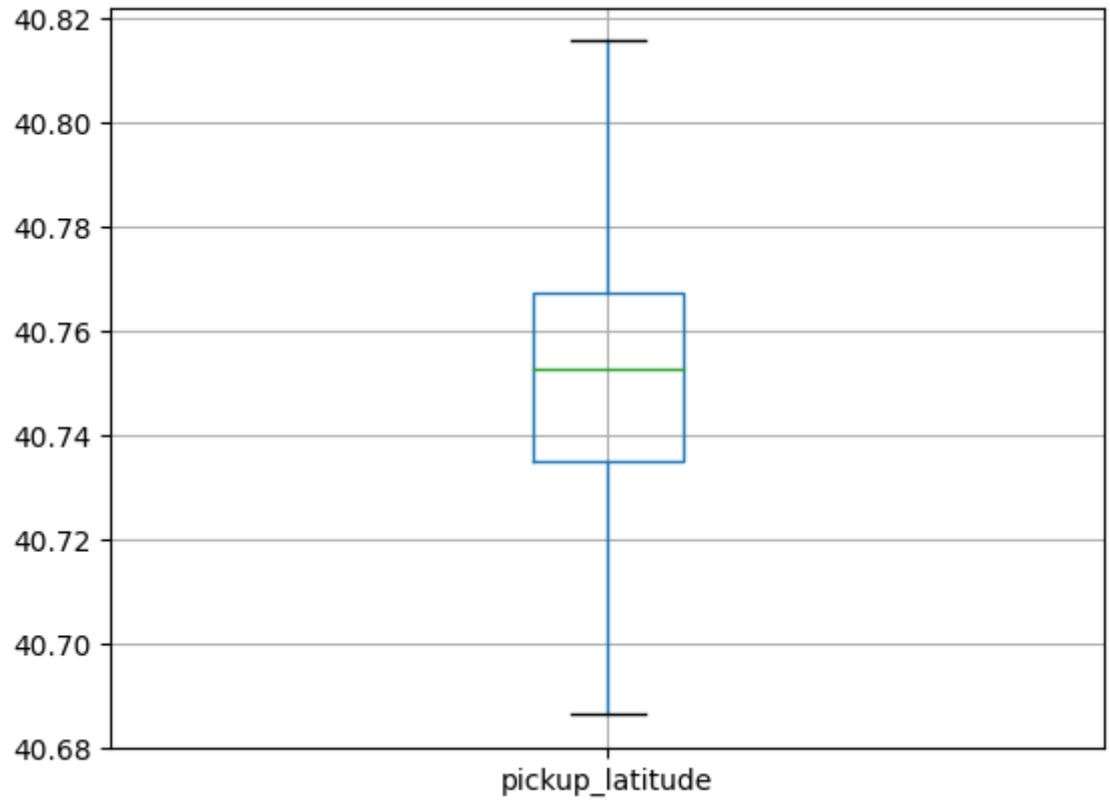
```
df['pickup_latitude']=np.where(df['pickup_latitude'] < lower, lower, df['pickup_latitude'])
df['pickup_latitude']=np.where(df['pickup_latitude'] > upper, upper, df['pickup_latitude'])
```

In [27]:

```
df.boxplot(column="pickup_latitude")
```

Out[27]:

```
<AxesSubplot:>
```



In [28]:

```
df['dropoff_longitude'] = df['dropoff_longitude'].fillna(0)
Q1 = np.quantile(df['dropoff_longitude'],0.25)
Q3 = np.quantile(df['dropoff_longitude'],0.75)
IQR = Q3-Q1
print("Q1",Q1)
print("Q3",Q3)
print("IQR = ",IQR)
upper = Q3 + IQR*1.5
lower = Q1 - IQR*1.5
print("Upper Quartile :: ",upper)
print("Lower Quartile :: ",lower)
```

```
Q1 -73.991407
Q3 -73.963658
IQR =  0.027749000000000024
Upper Quartile ::  -73.9220345
Lower Quartile ::  -74.0330305
```

In [29]:

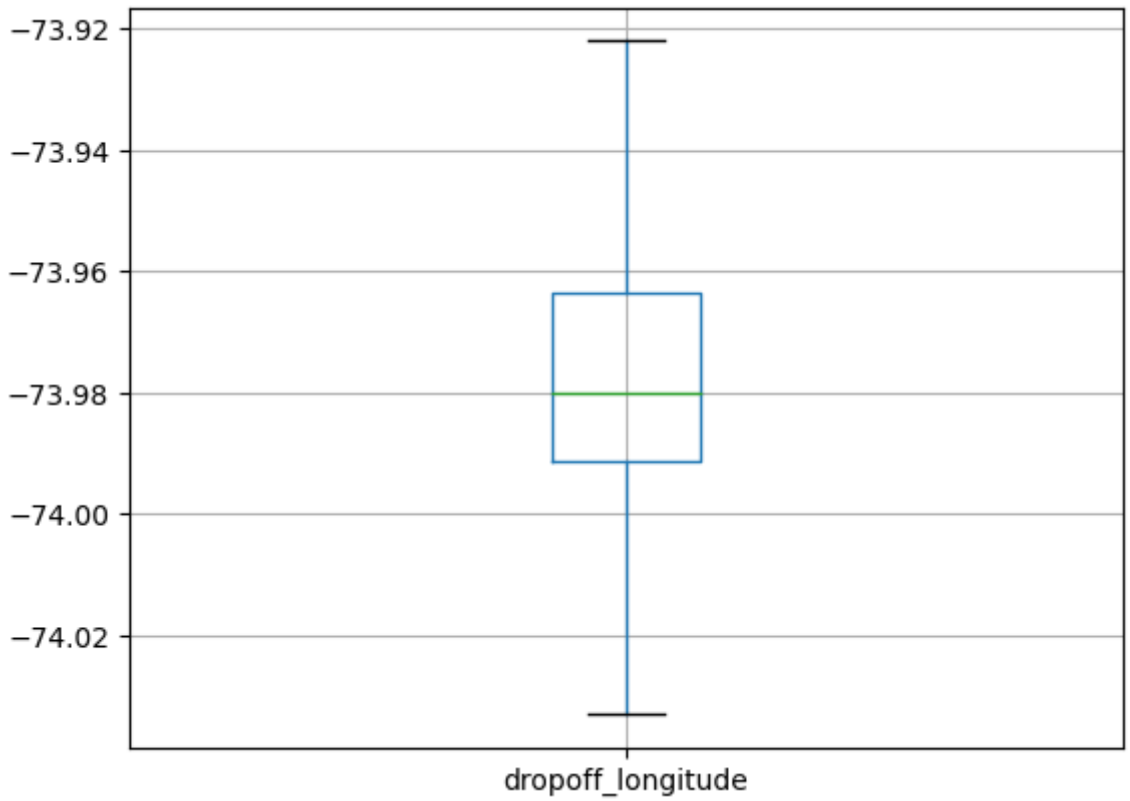
```
df['dropoff_longitude']=np.where(df['dropoff_longitude'] < lower, lower, df['dropoff_longit
df['dropoff_longitude']=np.where(df['dropoff_longitude'] > upper, upper, df['dropoff_longit
```

In [30]:

```
df.boxplot(column="dropoff_longitude")
```

Out[30]:

```
<AxesSubplot:>
```



In [31]:

```
df['dropoff_latitude'] = df['dropoff_latitude'].fillna(0)
Q1 = np.quantile(df['dropoff_latitude'],0.25)
Q3 = np.quantile(df['dropoff_latitude'],0.75)
IQR = Q3-Q1
print("Q1",Q1)
print("Q3",Q3)
print("IQR = ",IQR)
upper = Q3 + IQR*1.5
lower = Q1 - IQR*1.5
print("Upper Quartile :: ",upper)
print("Lower Quartile :: ",lower)
```

```
Q1 40.73382375
Q3 40.76800113909912
IQR =  0.0341773890991206
Upper Quartile ::  40.8192672227478
Lower Quartile ::  40.682557666351315
```

In [32]:

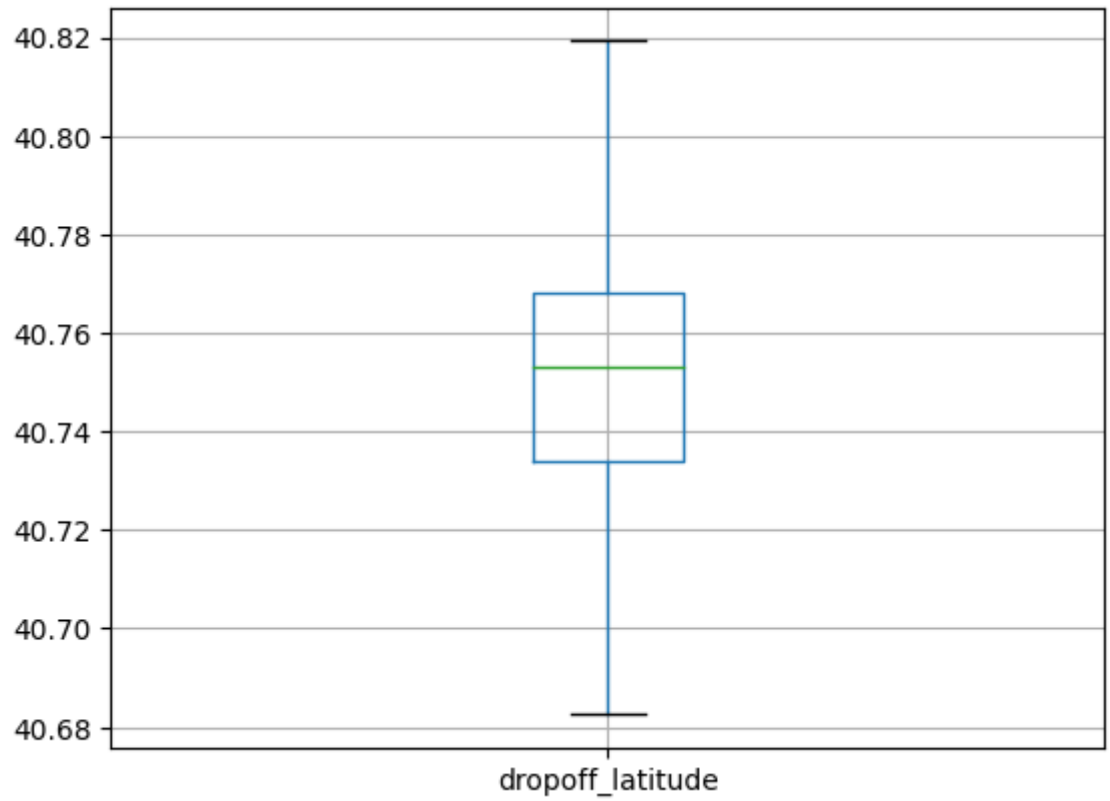
```
df['dropoff_latitude']=np.where(df['dropoff_latitude'] < lower, lower, df['dropoff_latitude
df['dropoff_latitude']=np.where(df['dropoff_latitude'] > upper, upper, df['dropoff_latitude
```

In [33]:

```
df.boxplot(column="dropoff_latitude")
```

Out[33]:

```
<AxesSubplot:>
```



In [34]:

```
df['passenger_count'] = df['passenger_count'].fillna(0)
Q1 = np.quantile(df['passenger_count'],0.25)
Q3 = np.quantile(df['passenger_count'],0.75)
IQR = Q3-Q1
print("Q1",Q1)
print("Q3",Q3)
print("IQR = ",IQR)
upper = Q3 + IQR*1.5
lower = Q1 - IQR*1.5
print("Upper Quartile :: ",upper)
print("Lower Quartile :: ",lower)
```

```
Q1 1.0
Q3 2.0
IQR =  1.0
Upper Quartile ::  3.5
Lower Quartile ::  -0.5
```

In [35]:

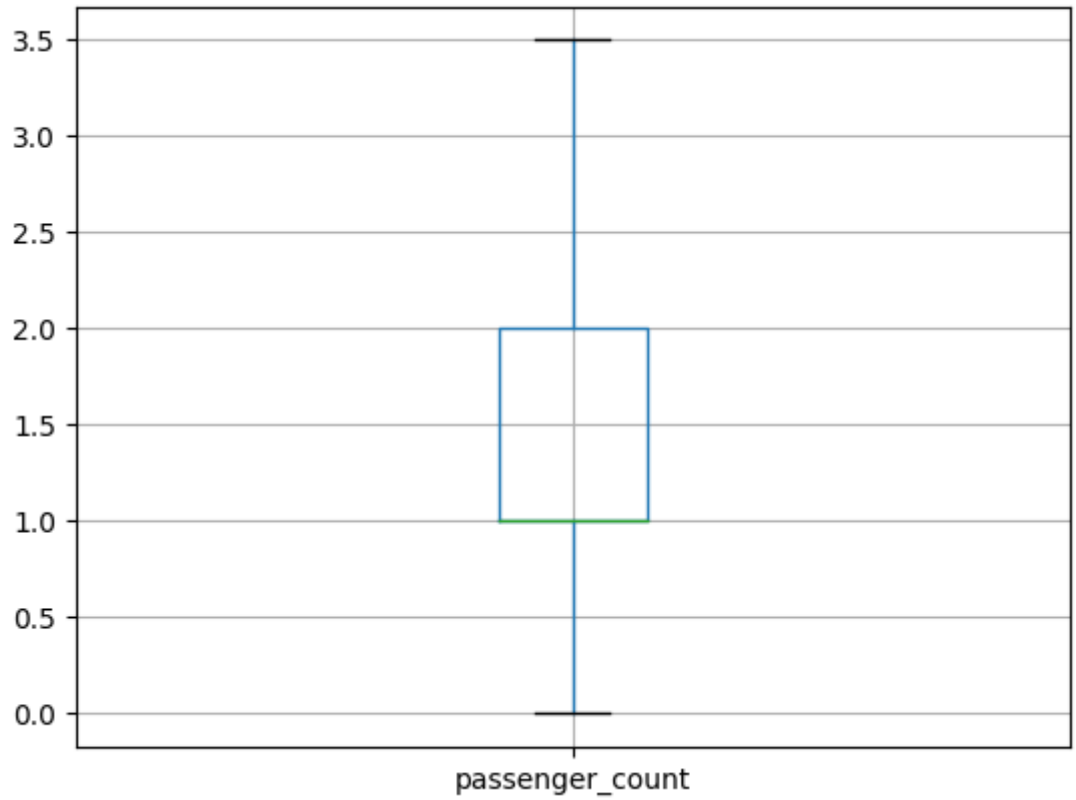
```
df['passenger_count']=np.where(df['passenger_count'] < lower, lower, df['passenger_count'])
df['passenger_count']=np.where(df['passenger_count'] > upper, upper, df['passenger_count'])
```

In [36]:

```
df.boxplot(column="passenger_count")
```

Out[36]:

```
<AxesSubplot:>
```

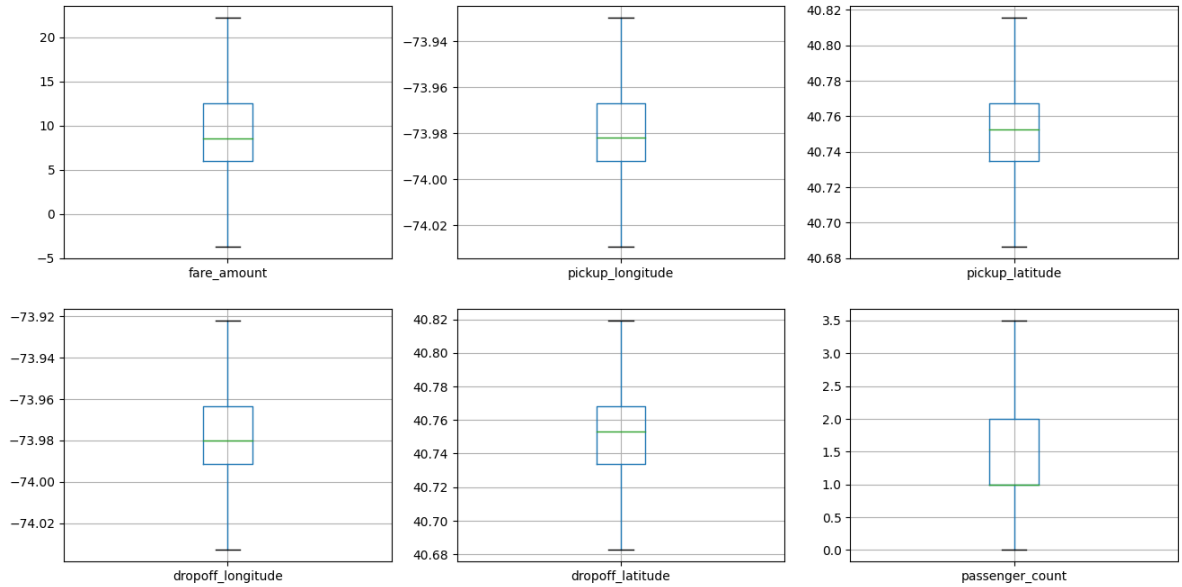


In [37]:

```
fig, axes = plt.subplots(2, 3, figsize=(16,8))
df.boxplot(column="fare_amount", ax=axes[0,0])
df.boxplot(column="pickup_longitude", ax=axes[0,1])
df.boxplot(column="pickup_latitude", ax=axes[0,2])
df.boxplot(column="dropoff_longitude", ax=axes[1,0])
df.boxplot(column="dropoff_latitude", ax=axes[1,1])
df.boxplot(column="passenger_count", ax=axes[1,2])
```

Out[37]:

<AxesSubplot:>



In [38]:

```
import haversine as hs
```

In [39]:

```
def get_total_fare(row):
  loc1 = (row['pickup_latitude'], row['pickup_longitude'])
  loc2 = (row['dropoff_latitude'], row['dropoff_longitude'])

  dis = hs.haversine(loc1,loc2);
  return dis * row['fare_amount']

def get_dis(row):
  loc1 = (row['pickup_latitude'], row['pickup_longitude'])
  loc2 = (row['dropoff_latitude'], row['dropoff_longitude'])

  return hs.haversine(loc1,loc2);

df['total_fare'] = df.apply (lambda row: get_total_fare(row), axis=1)
df['total_distance'] = df.apply (lambda row: get_dis(row), axis=1)
```

In [40]:

```
df['total_fare'].head()
```

Out[40]:

```
0    12.624938
1    18.923468
2    64.969355
3     8.806934
4    65.857411
Name: total_fare, dtype: float64
```

In [41]:

```
df.head()
```

Out[41]:

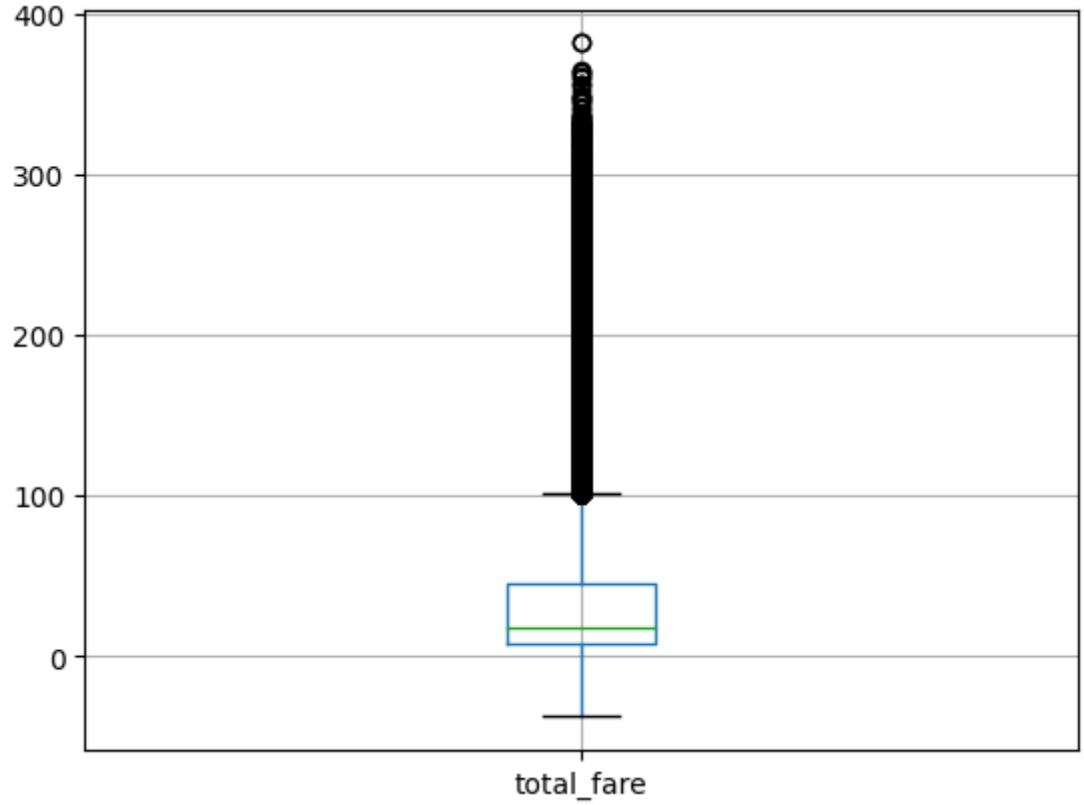
	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_
0	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-
1	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-
2	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-
3	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-
4	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00+00:00	-73.929786	40.744085	-

In [42]:

```
df.boxplot(column="total_fare")
```

Out[42]:

<AxesSubplot:>



In [43]:

```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(df.drop(labels=['total_fare','pickup_dateti

print("xtrain shape : ", xtrain.shape)
print("xtest shape : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape : ", ytest.shape)
```

```
xtrain shape :  (150000, 7)
xtest shape :  (50000, 7)
ytrain shape :  (150000,)
ytest shape :  (50000,)
```

In [44]:

```
from sklearn.linear_model import LinearRegression
```

In [45]:

```
reg = LinearRegression()
reg.fit(xtrain, ytrain)
```

Out[45]:

```
LinearRegression()
```

In [46]:

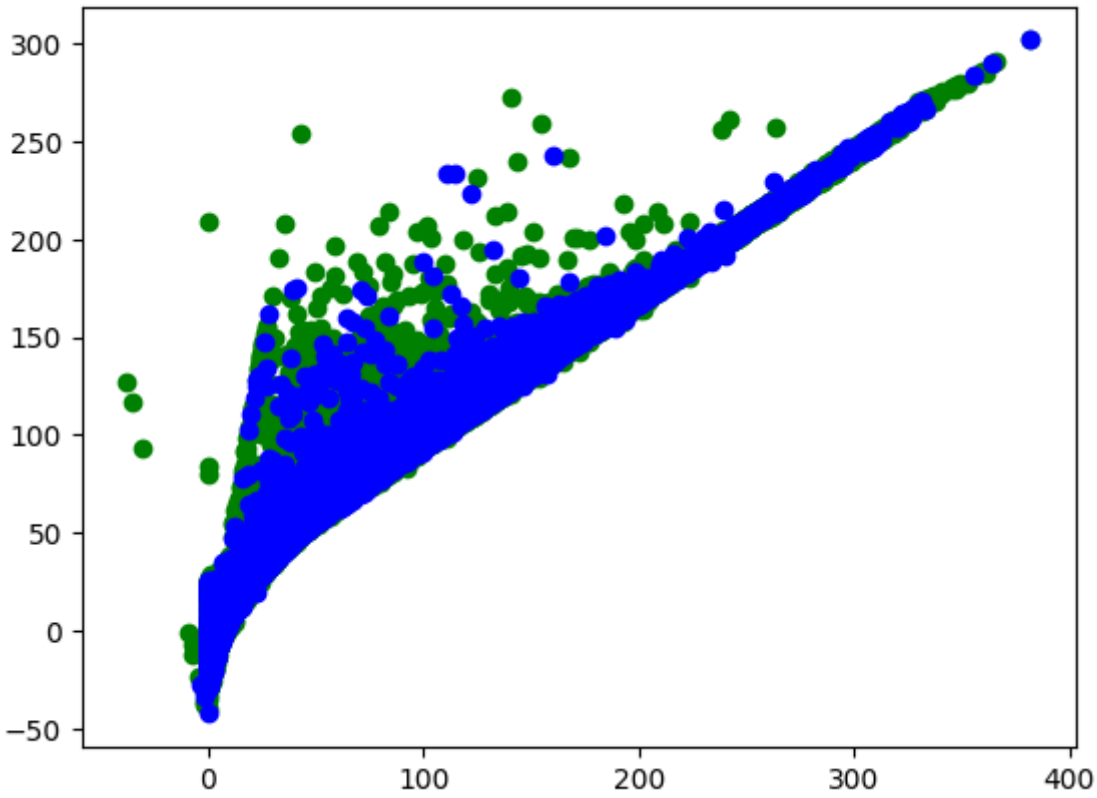
```
ytest_pred=reg.predict(xtest)
ytrain_pred=reg.predict(xtrain)
```

In [47]:

```
plt.scatter(ytrain,ytrain_pred,c='green',label="Training data")
plt.scatter(ytest,ytest_pred,c="blue",label="Testing data")
```

Out[47]:

```
<matplotlib.collections.PathCollection at 0x1c174811a30>
```



In [48]:

```
correlation_matrix = df.corr()
```

In [49]:

```
print(correlation_matrix["total_fare"].sort_values(ascending=False))
```

```
total_fare           1.000000
total_distance       0.951147
fare_amount          0.854607
dropoff_longitude    0.200939
pickup_longitude     0.105679
passenger_count      0.010995
dropoff_latitude    -0.085830
pickup_latitude     -0.093742
Name: total_fare, dtype: float64
```

In [50]:

```
df.corr()
```

Out[50]:

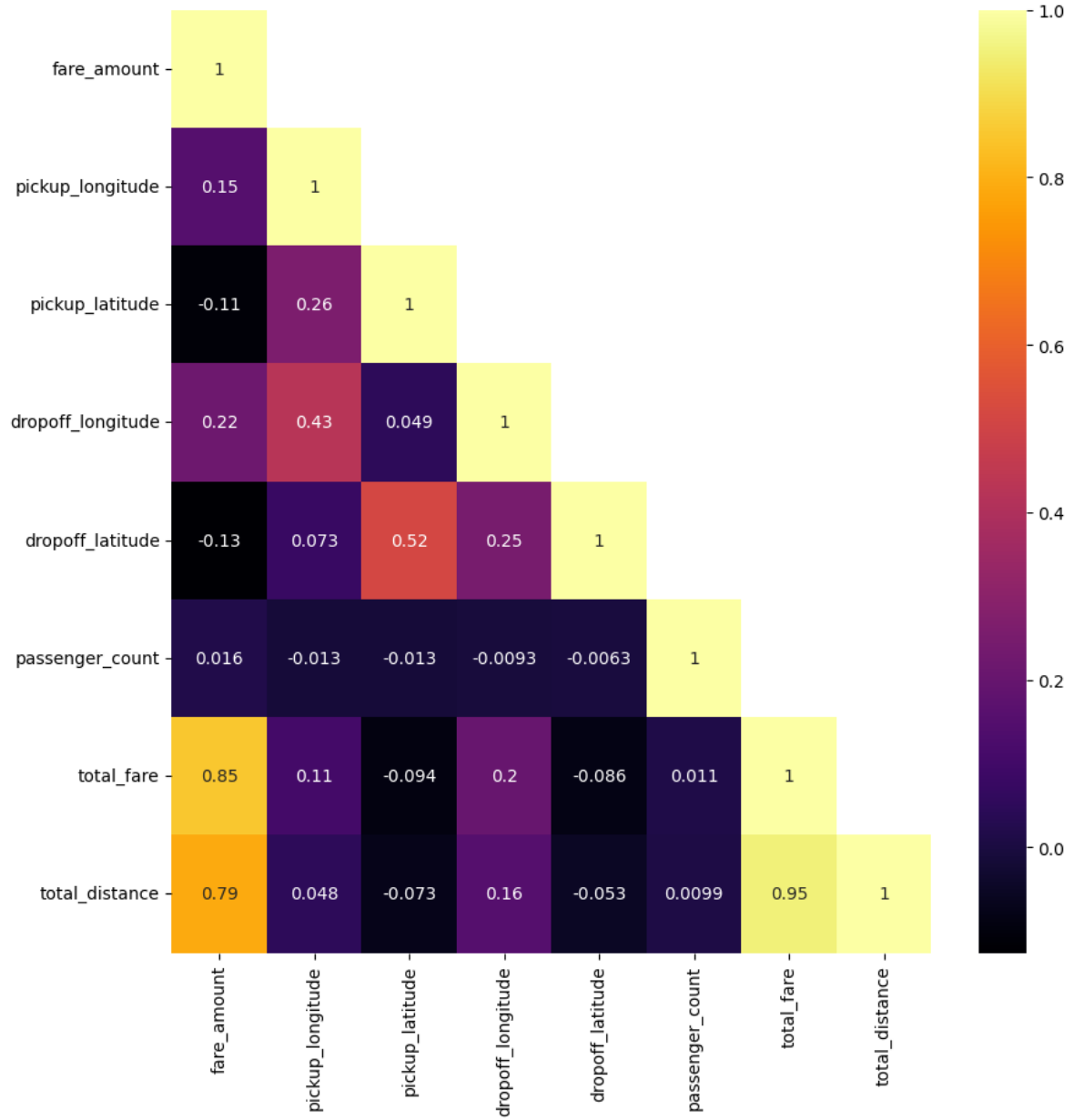
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_lat
fare_amount	1.000000	0.154069	-0.110842	0.218704	-0.12
pickup_longitude	0.154069	1.000000	0.259497	0.425631	0.07
pickup_latitude	-0.110842	0.259497	1.000000	0.048898	0.51
dropoff_longitude	0.218704	0.425631	0.048898	1.000000	0.24
dropoff_latitude	-0.125871	0.073311	0.515735	0.245665	1.00
passenger_count	0.015778	-0.013213	-0.012889	-0.009325	-0.00
total_fare	0.854607	0.105679	-0.093742	0.200939	-0.08
total_distance	0.786377	0.048427	-0.073383	0.155208	-0.05

In [51]:

```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), annot=True, cmap='inferno', mask=np.triu(df.corr(), k=1))
```

Out[51]:

<AxesSubplot:>



In [52]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [57]:

```
rf = RandomForestRegressor()
rf.fit(xtrain,ytrain)
```

Out[57]:

RandomForestRegressor

RandomForestRegressor()

In [ ]:

```
ytest_pred2=rf.predict(xtest)
ytrain_pred2=rf.predict(xtrain)
```

In [ ]:

```
plt.scatter(ytrain,ytrain_pred2, c='blue', label="Training data")
plt.scatter(ytest,ytest_pred2, c='green', label="Testing data")
```

In [122]:

```
from sklearn.metrics import mean_squared_error,r2_score
mse_test=mean_squared_error(ytest,ytest_pred)
print("mse_test = ",mse_test)
print("rmseTest = ", np.sqrt(mse_test))
mse_train=mean_squared_error(ytrain,ytrain_pred)
print("mse_train = ",mse_train)
print("rmseTrain = ", np.sqrt(mse_train))
```

```
mse_test =  156.0251276778257
rmseTest =  12.491001868458179
mse_train =  158.98402516371482
rmseTrain =  12.608886753544693
```

In [ ]: