

```
# Importing necessary libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib as plt
```

```
import seaborn as sns
```

```
# Loading the dataset
```

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

```
df.head()
```

	Unnamed: 0	key	fare_amount	\
0	24238194	2015-05-07 19:52:06.00000003	7.5	
1	27835199	2009-07-17 20:04:56.00000002	7.7	
2	44984355	2009-08-24 21:45:00.000000061	12.9	
3	25894730	2009-06-26 08:22:21.00000001	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

```
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
```

```
df.describe()
```

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude
count	2.000000e+05	200000.000000	200000.000000	200000.000000
mean	2.771250e+07	11.359955	-72.527638	39.935885
std	1.601382e+07	9.901776	11.437787	7.720539
min	1.000000e+00	-52.000000	-1340.648410	-74.015515
25%	1.382535e+07	6.000000	-73.992065	40.734796
50%	2.774550e+07	8.500000	-73.981823	40.752592
75%	4.155530e+07	12.500000	-73.967154	40.767158
max	5.542357e+07	499.000000	57.418457	1644.421482

	dropoff_longitude	dropoff_latitude	passenger_count
count	199999.000000	199999.000000	200000.000000
mean	-72.525292	39.923890	1.684535
std	13.117408	6.794829	1.385997
min	-3356.666300	-881.985513	0.000000
25%	-73.991407	40.733823	1.000000
50%	-73.980093	40.753042	1.000000
75%	-73.963658	40.768001	2.000000
max	1153.572603	872.697628	208.000000

```
df.shape
```

```
(200000, 9)
```

1.Pre-process the dataset

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

Unnamed: 0	0
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	

```
#dropping rows with missing values
```

```
df.dropna(inplace = True)
```

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

```
Unnamed: 0      0
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
```

```
#dropping unwanted columns
```

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

```
df.drop(labels='key',axis=1,inplace=True)
```

```
df.head()
```

	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				

	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

```
#converting datatype of column "pickup_datetime" from object to DateTime
```

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

df.dtypes

```
df.describe()
```

	fare_amount	pickup_longitude	pickup_latitude	
dropoff_longitude \				
count	199999.000000	199999.000000	199999.000000	
199999.000000				
mean	11.359892	-72.527631	39.935881	-
72.525292				
std	9.901760	11.437815	7.720558	
13.117408				
min	-52.000000	-1340.648410	-74.015515	-
3356.666300				
25%	6.000000	-73.992065	40.734796	-
73.991407				
50%	8.500000	-73.981823	40.752592	-
73.980093				
75%	12.500000	-73.967154	40.767158	-
73.963658				
max	499.000000	57.418457	1644.421482	
1153.572603				

	dropoff_latitude	passenger_count
count	199999.000000	199999.000000
mean	39.923890	1.684543
std	6.794829	1.385995
min	-881.985513	0.000000
25%	40.733823	1.000000
50%	40.753042	1.000000
75%	40.768001	2.000000
max	872.697628	208.000000

2. Identify outliers

```
# OUTLIER: An object that deviates significantly from rest of the objects
```

```
# data visualization
```

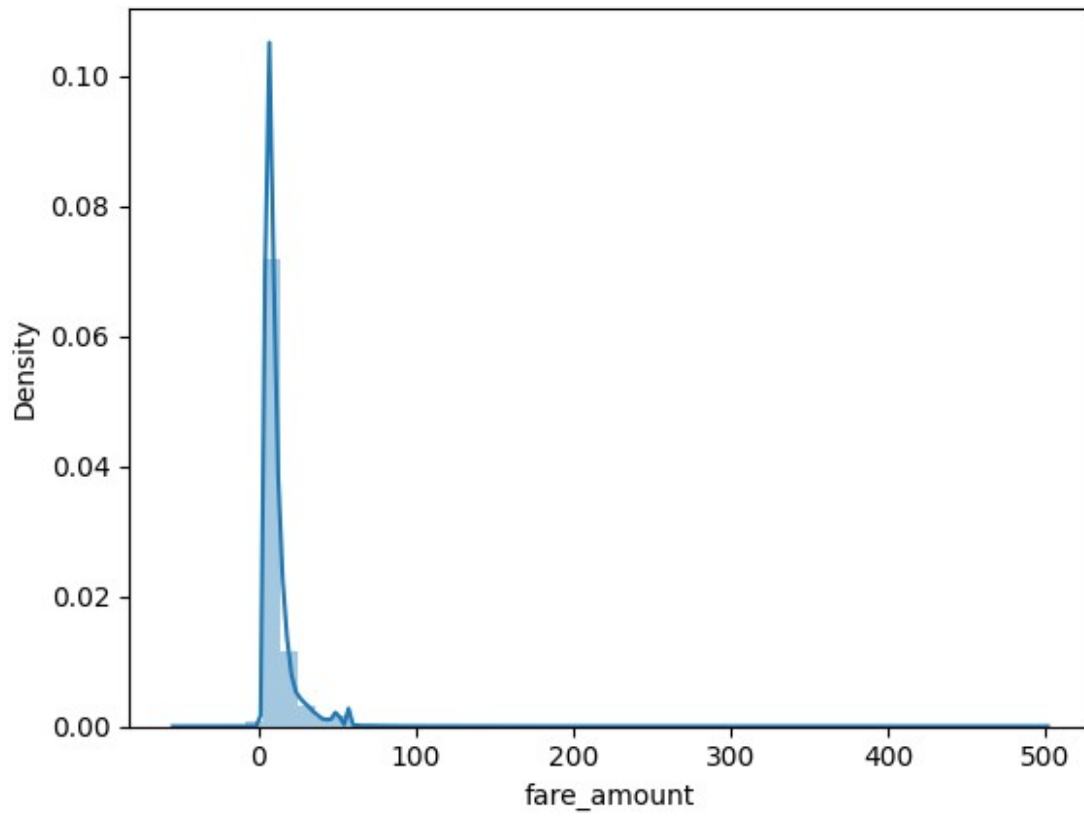
```
# plotting distribution plot
```

```
import warnings
```

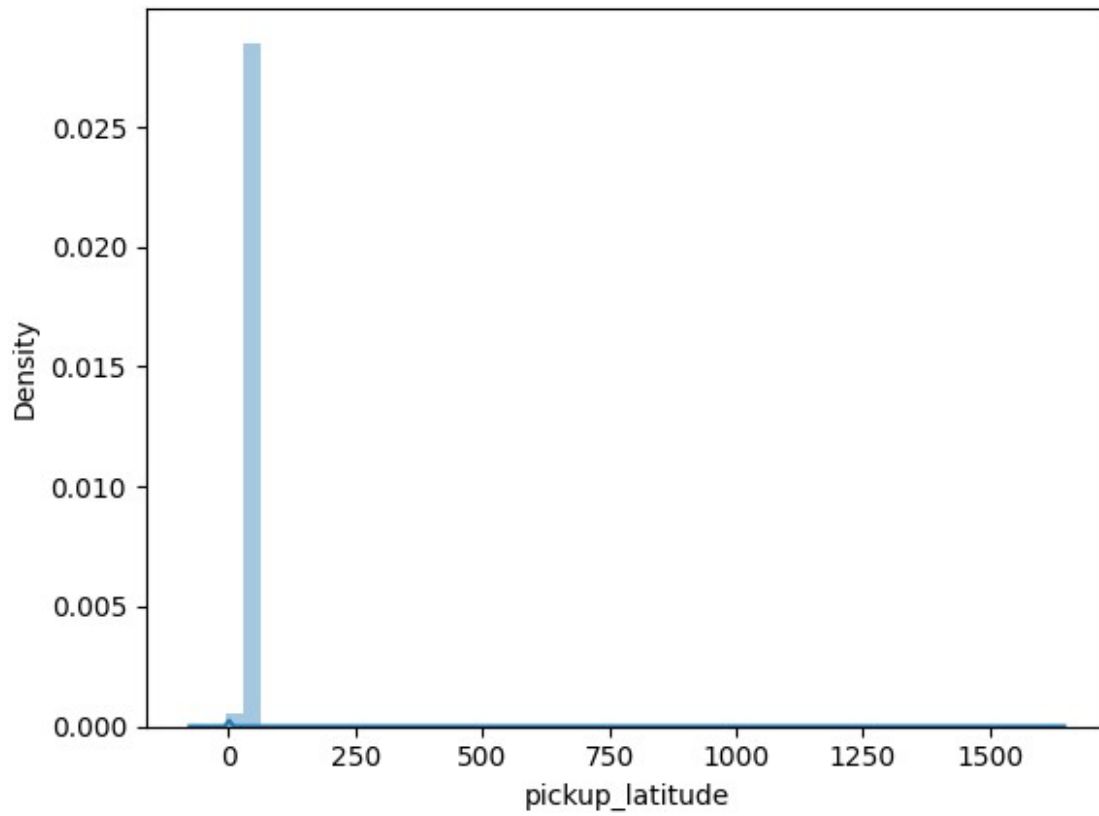
```
warnings.filterwarnings("ignore")
```

```
sns.distplot(df['fare_amount'])
```

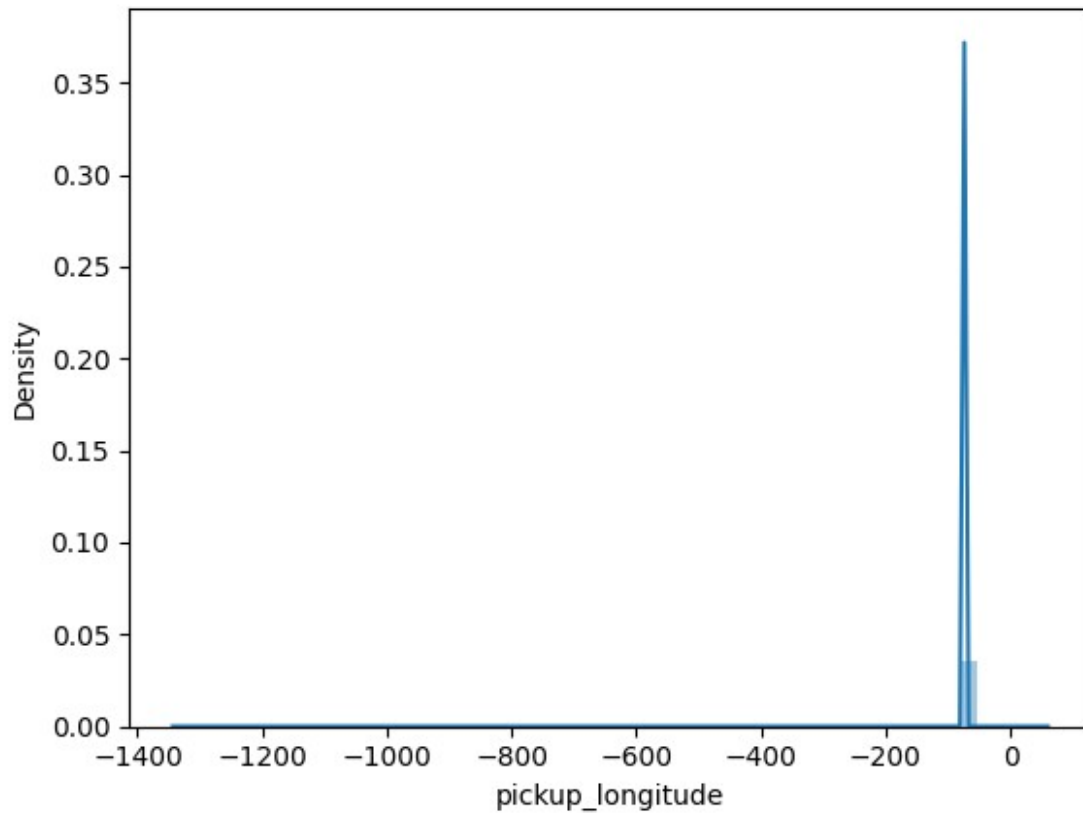
```
<Axes: xlabel='fare_amount', ylabel='Density'>
```



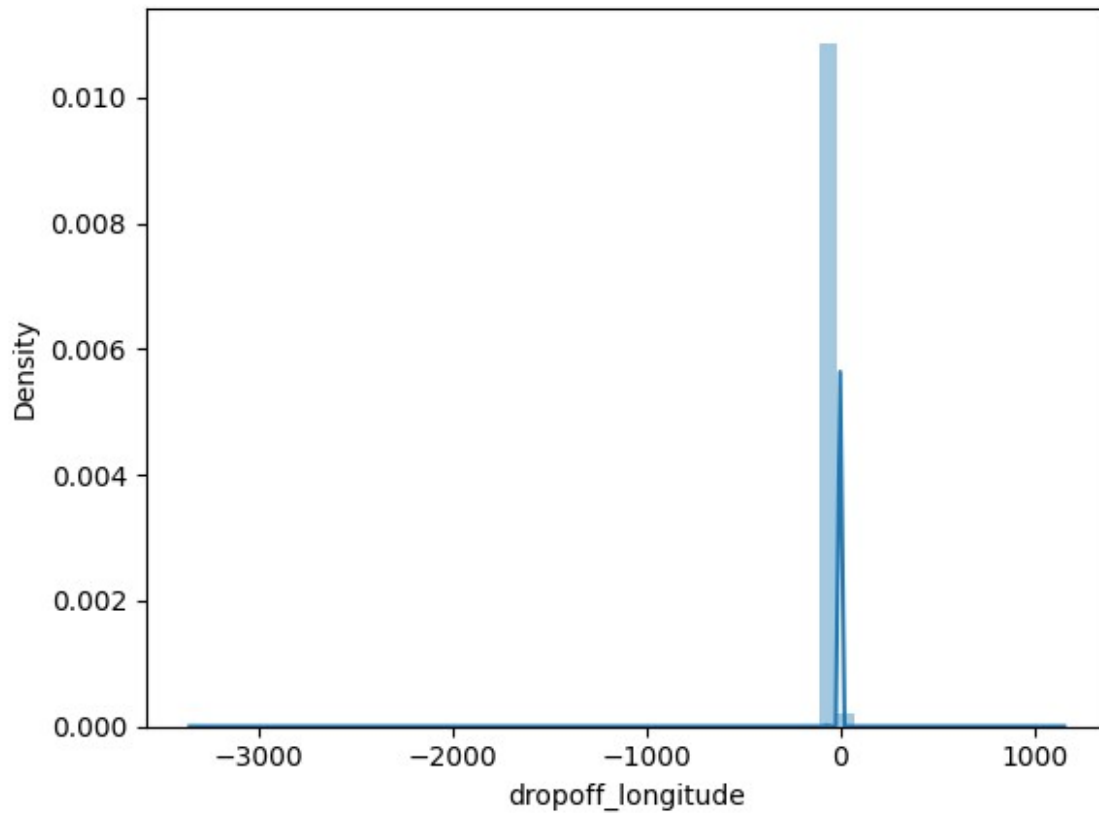
```
sns.distplot(df['pickup_latitude'])  
<Axes: xlabel='pickup_latitude', ylabel='Density'>
```



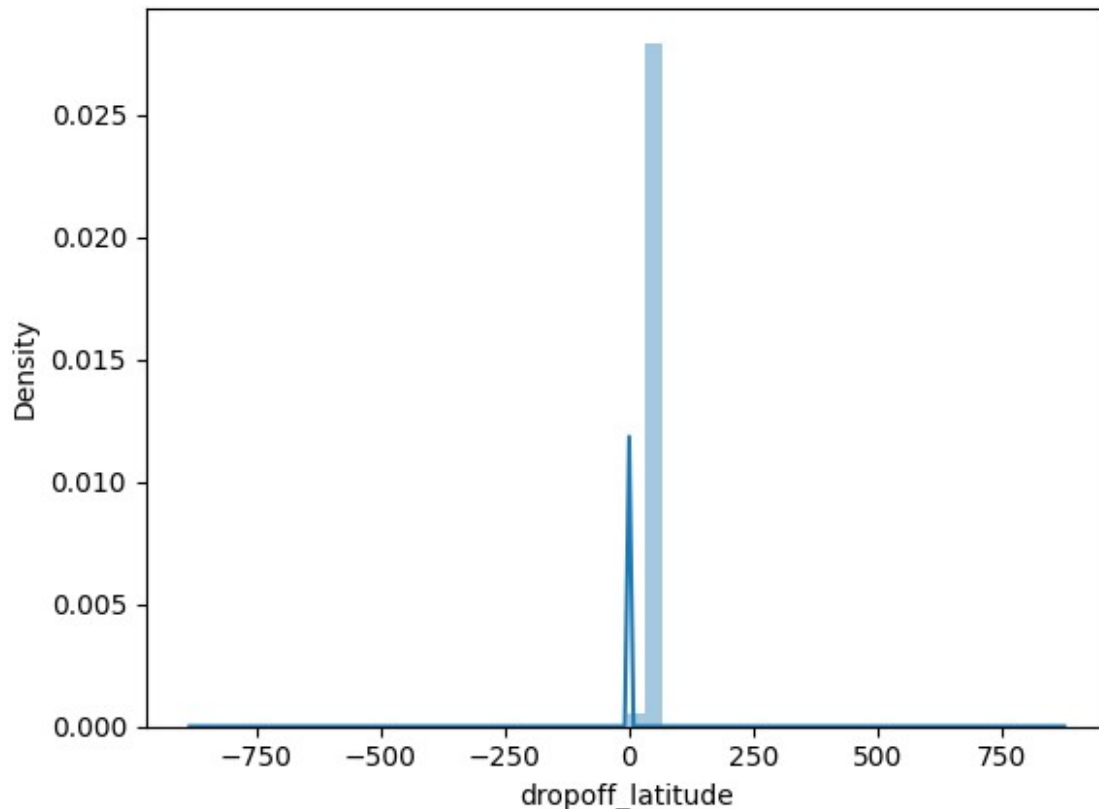
```
sns.distplot(df['pickup_longitude'])  
<Axes: xlabel='pickup_longitude', ylabel='Density'>
```



```
sns.distplot(df['dropoff_longitude'])  
<Axes: xlabel='dropoff_longitude', ylabel='Density'>
```



```
sns.distplot(df['dropoff_latitude'])  
<Axes: xlabel='dropoff_latitude', ylabel='Density'>
```

#creating a function to identify outliers

```
def find_outliers_IQR(df):  
    q1 = df.quantile(0.25)  
    q3 = df.quantile(0.75)  
    IQR = q3-q1  
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]  
    return outliers
```

#getting outlier details for column "fare_amount" using the above function

```
outliers = find_outliers_IQR(df["fare_amount"])  
print("number of outliers: "+ str(len(outliers)))  
print("max outlier value: "+ str(outliers.max()))  
print("min outlier value: "+ str(outliers.min()))  
outliers
```

```
number of outliers: 17166  
max outlier value: 499.0  
min outlier value: -52.0
```

```
6      24.50  
30     25.70  
34     39.50
```

```

39      29.00
48      56.80
...
199976   49.70
199977   43.50
199982   57.33
199985   24.00
199997   30.90
Name: fare_amount, Length: 17166, dtype: float64

#you can also pass two columns as argument to the function (here
"passenger_count" and "fare_amount")

outliers = find_outliers_IQR(df[["passenger_count", "fare_amount"]])
outliers

   passenger_count  fare_amount
0              NaN           NaN
1              NaN           NaN
2              NaN           NaN
3              NaN           NaN
4              5.0           NaN
...
199995          NaN           NaN
199996          NaN           NaN
199997          NaN          30.9
199998          NaN           NaN
199999          NaN           NaN

[199999 rows x 2 columns]

#upper and lower limit which can be used for capping of outliers

upper_limit = df['fare_amount'].mean() + 3*df['fare_amount'].std()
print(upper_limit)
lower_limit = df['fare_amount'].mean() - 3*df['fare_amount'].std()
print(lower_limit)

41.06517154774142
-18.345388448825922

```

3. Check the correlation

```

#creating a correlation matrix

corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()

```

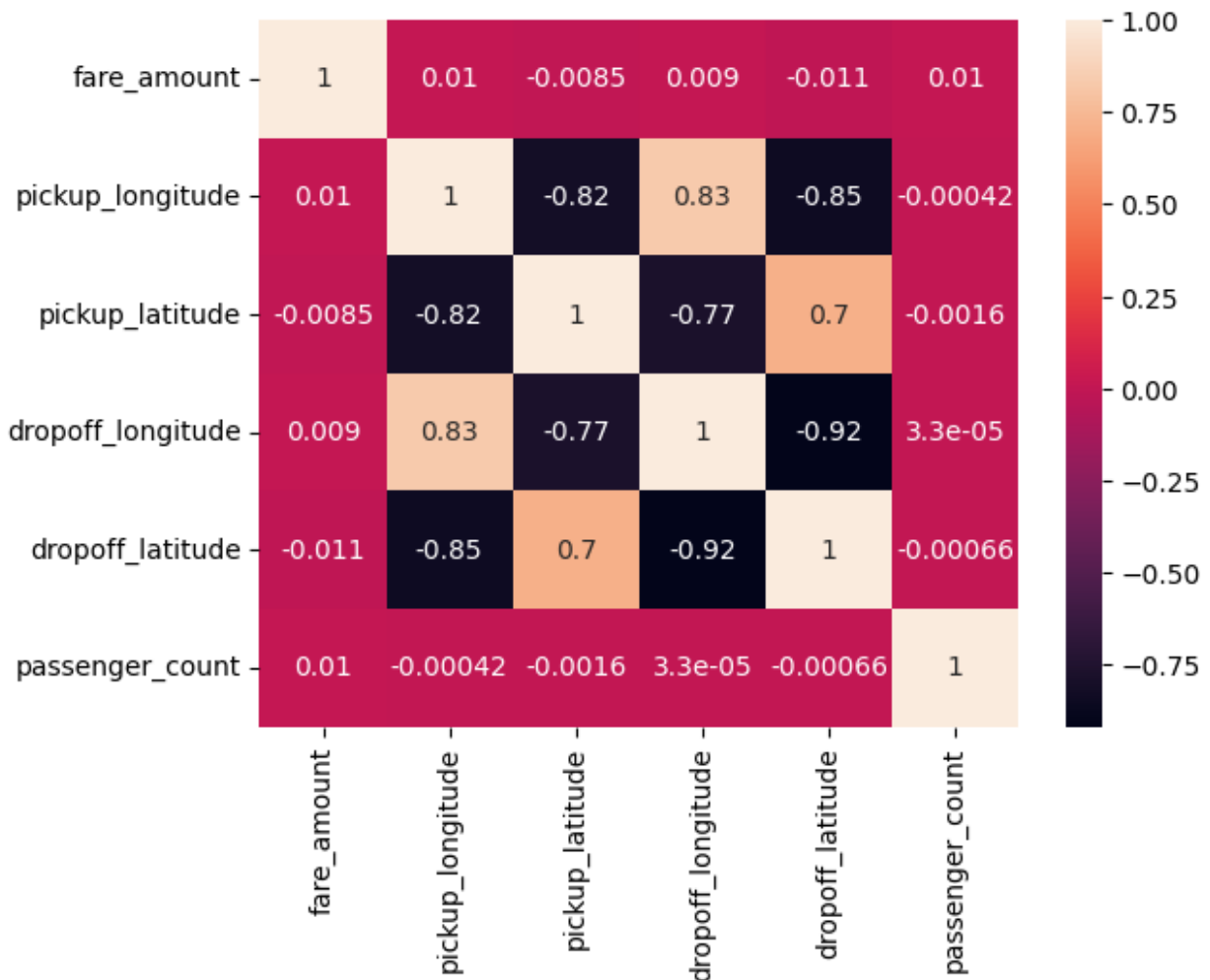
```

-----
-----
AttributeError                                Traceback (most recent call
last)
Cell In[26], line 5
      3 corrMatrix = df.corr()
      4 sns.heatmap(corrMatrix, annot=True)
----> 5 plt.show()

File ~\anaconda3\envs\pandas\lib\site-packages\matplotlib\_api\
__init__.py:226, in caching_module_getattr.<locals>.__getattr__(name)
    224 if name in props:
    225     return props[name].__get__(instance)
--> 226 raise AttributeError(
    227     f"module {cls.__module__!r} has no attribute {name!r}")

AttributeError: module 'matplotlib' has no attribute 'show'

```



```
#splitting column "pickup_datetime" into 5 columns: "day", "hour",
"month", "year", "weekday"
#for a simplified view
```

```
import calendar
df['day']=df['pickup_datetime'].apply(lambda x:x.day)
df['hour']=df['pickup_datetime'].apply(lambda x:x.hour)
df['month']=df['pickup_datetime'].apply(lambda x:x.month)
df['year']=df['pickup_datetime'].apply(lambda x:x.year)
df['weekday']=df['pickup_datetime'].apply(lambda x:
calendar.day_name[x.weekday()])
df.drop(['pickup_datetime'],axis=1,inplace=True)
```

```
#label encoding (categorical to numerical)
```

```
df.weekday =
df.weekday.map({'Sunday':0, 'Monday':1, 'Tuesday':2, 'Wednesday':3, 'Thurs
day':4, 'Friday':5, 'Saturday':6})
```

```
df.head()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude \
0	7.5	-73.999817	40.738354	-73.999512
1	7.7	-73.994355	40.728225	-73.994710
2	12.9	-74.005043	40.740770	-73.962565
3	5.3	-73.976124	40.790844	-73.965316
4	16.0	-73.925023	40.744085	-73.973082

	dropoff_latitude	passenger_count	day	hour	month	year	weekday
0	40.723217	1	7	19	5	2015	4
1	40.750325	1	17	20	7	2009	5
2	40.772647	1	24	21	8	2009	1
3	40.803349	3	26	8	6	2009	5
4	40.761247	5	28	17	8	2014	4

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 199999 entries, 0 to 199999
Data columns (total 11 columns):
```

#	Column	Non-Null Count	Dtype
0	fare_amount	199999 non-null	float64
1	pickup_longitude	199999 non-null	float64
2	pickup_latitude	199999 non-null	float64
3	dropoff_longitude	199999 non-null	float64
4	dropoff_latitude	199999 non-null	float64
5	passenger_count	199999 non-null	int64
6	day	199999 non-null	int64
7	hour	199999 non-null	int64
8	month	199999 non-null	int64
9	year	199999 non-null	int64
10	weekday	199999 non-null	int64

dtypes: float64(5), int64(6)

memory usage: 18.3 MB

#splitting the data into train and test

from sklearn.model_selection import train_test_split

#independent variables (x)

x=df.drop("fare_amount", axis=1)

x

	pickup_longitude	pickup_latitude	dropoff_longitude	\
0	-73.999817	40.738354	-73.999512	
1	-73.994355	40.728225	-73.994710	
2	-74.005043	40.740770	-73.962565	
3	-73.976124	40.790844	-73.965316	
4	-73.925023	40.744085	-73.973082	
...	
199995	-73.987042	40.739367	-73.986525	
199996	-73.984722	40.736837	-74.006672	
199997	-73.986017	40.756487	-73.858957	
199998	-73.997124	40.725452	-73.983215	
199999	-73.984395	40.720077	-73.985508	

	dropoff_latitude	passenger_count	day	hour	month	year
weekday						
0	40.723217	1	7	19	5	2015
4						
1	40.750325	1	17	20	7	2009
5						
2	40.772647	1	24	21	8	2009
1						
3	40.803349	3	26	8	6	2009
5						
4	40.761247	5	28	17	8	2014
4						

```

...
...
199995      40.740297      1    28    10    10    2012
0
199996      40.739620      1    14     1     3    2014
5
199997      40.692588      2    29     0     6    2009
1
199998      40.695415      1    20    14     5    2015
3
199999      40.768793      1    15     4     5    2010
6

```

[199999 rows x 10 columns]

#dependent variable (y)

y=df["fare_amount"]

x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.2,random_state=101)

x_train.head()

	pickup_longitude	pickup_latitude	dropoff_longitude	\
80768	-73.983703	40.725752	-73.972000	
111783	-73.961175	40.760667	-73.976507	
24615	-73.947784	40.783111	-73.955408	
46932	-73.980596	40.733797	-73.972092	
86655	-73.963035	40.758380	-73.987877	

	dropoff_latitude	passenger_count	day	hour	month	year
weekday						
80768	40.793888	1	22	1	2	2009
0						
111783	40.747570	1	7	14	3	2009
6						
24615	40.779405	1	17	11	3	2011
4						
46932	40.747297	1	15	7	1	2010
5						
86655	40.745477	2	28	19	6	2014
6						

x_test.head()

	pickup_longitude	pickup_latitude	dropoff_longitude	\
13588	-73.982810	40.771687	-73.977065	
29803	-73.991985	40.725763	-73.995762	
138265	-73.985730	40.767882	-73.998525	
82856	-73.973200	40.748100	-73.973500	

162747	-74.007432	40.716580	-73.986858				
	dropoff_latitude	passenger_count	day	hour	month	year	
13588	40.763200	1	25	22	6	2013	
2							
29803	40.759797	1	20	20	2	2011	
0							
138265	40.760667	1	20	14	10	2013	
0							
82856	40.748200	1	17	18	11	2011	
4							
162747	40.761328	1	10	22	2	2009	
2							

```
y_train.head()
```

```
80768    19.7
111783     7.7
24615     4.5
46932     4.5
86655    10.0
Name: fare_amount, dtype: float64
```

```
y_test.head()
```

```
13588     5.5
29803    11.3
138265     6.5
82856    18.1
162747    11.3
Name: fare_amount, dtype: float64
```

```
print(x_train.shape)
print(x_test.shape)
print(y_test.shape)
print(y_train.shape)
```

```
(159999, 10)
(40000, 10)
(40000,)
(159999,)
```

4. Implementing linear regression and random forest regression models

```
#Linear Regression
```

```

from sklearn.linear_model import LinearRegression
lrmodel=LinearRegression()
lrmodel.fit(x_train, y_train)

LinearRegression()

predictedvalues = lrmodel.predict(x_test)

#Random Forest Regression

from sklearn.ensemble import RandomForestRegressor
rfrmodel = RandomForestRegressor(n_estimators=100, random_state=101)

rfrmodel.fit(x_train,y_train)
rfrmodel_pred= rfrmodel.predict(x_test)

```

5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

```

#Calculating the value of RMSE for Linear Regression

from sklearn.metrics import mean_squared_error
lrmodelrmse = np.sqrt(mean_squared_error(predictedvalues, y_test))
print("RMSE value for Linear regression is", lrmodelrmse)

RMSE value for Linear regression is 9.806687708433813

#Calculating the value of RMSE for Random Forest Regression

rfrmodel_rmse=np.sqrt(mean_squared_error(rfrmodel_pred, y_test))
print("RMSE value for Random forest regression is ",rfrmodel_rmse)

RMSE value for Random forest regression is 4.756011315216782

```