```
#Importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

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

pre process the dataset

```
df.head()
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropo
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	

```
df.info() #To get the required information of the dataset
```

```
     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 187717 entries, 0 to 187716
     Data columns (total 9 columns):
      #   Column             Non-Null Count   Dtype  
     ---  ------             --------------   -----  
      0   Unnamed: 0         187717 non-null  int64  
      1   key                187717 non-null  object 
      2   fare_amount        187717 non-null  float64
      3   pickup_datetime    187717 non-null  object 
      4   pickup_longitude   187717 non-null  float64
      5   pickup_latitude    187717 non-null  float64
      6   dropoff_longitude  187716 non-null  float64
      7   dropoff_latitude   187715 non-null  float64
      8   passenger_count    187716 non-null  float64
     dtypes: float64(6), int64(1), object(2)
     memory usage: 12.9+ MB
```

```
df.columns #TO get number of columns in the dataset
```

```
     Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
            'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
            'dropoff_latitude', 'passenger_count'],
           dtype='object')
```

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

```
df.head()
```

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_cou
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
		2009-08-24					

```
df.shape #To get the total (Rows,Columns)
```

```
(187717, 7)
```

```
df.dtypes #To get the type of each column
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 187717 entries, 0 to 187716
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   fare_amount        187717 non-null  float64
 1   pickup_datetime    187717 non-null  object 
 2   pickup_longitude   187717 non-null  float64
 3   pickup_latitude    187717 non-null  float64
 4   dropoff_longitude  187716 non-null  float64
 5   dropoff_latitude   187715 non-null  float64
 6   passenger_count    187716 non-null  float64
```

```
    dtypes: float64(6), object(1)
    memory usage: 10.0+ MB
```

```
df.describe() #To get statistics of each columns
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	187717.000000	187717.000000	187717.000000	187716.000000	187715.000000	187716.000000
mean	11.357142	-72.527021	39.936938	-72.526449	39.923029	1.684209
std	9.879560	11.396782	7.823140	13.277703	6.850351	1.390584
min	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	6.000000	-73.992070	40.734810	-73.991405	40.733835	1.000000
50%	8.500000	-73.981827	40.752611	-73.980087	40.753040	1.000000
75%	12.500000	-73.967137	40.767160	-73.963685	40.767993	2.000000
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

filling missing values

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

```
    fare_amount          0
    pickup_datetime      0
    pickup_longitude     0
    pickup_latitude      0
    dropoff_longitude    1
    dropoff_latitude     2
    passenger_count      1
    dtype: int64
```

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

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

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

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

```
df.dtypes
```

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

Column pickup_datetime is in wrong format (Object). Convert it to DateTime Format

```
df.pickup_datetime = pd.to_datetime(df.pickup_datetime, errors='coerce')
```

```
df.dtypes
```

```
fare_amount                      float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                 float64
pickup_latitude                  float64
dropoff_longitude                float64
dropoff_latitude                 float64
```

```
passenger_count              float64
dtype: object
```

To segregate each time of date and time

```
df= df.assign(hour = df.pickup_datetime.dt.hour,
             day= df.pickup_datetime.dt.day,
             month = df.pickup_datetime.dt.month,
             year = df.pickup_datetime.dt.year,
             dayofweek = df.pickup_datetime.dt.dayofweek)
```

```
df.head()
```

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_cou
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5

```
# drop the column 'pickup_daetime' using drop()
# 'axis = 1' drops the specified column

df = df.drop('pickup_datetime',axis=1)
```

```
df.head()
df.head()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	mor
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1.0	19	7	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1.0	20	17	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1.0	21	24	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3.0	8	26	
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5.0	17	28	

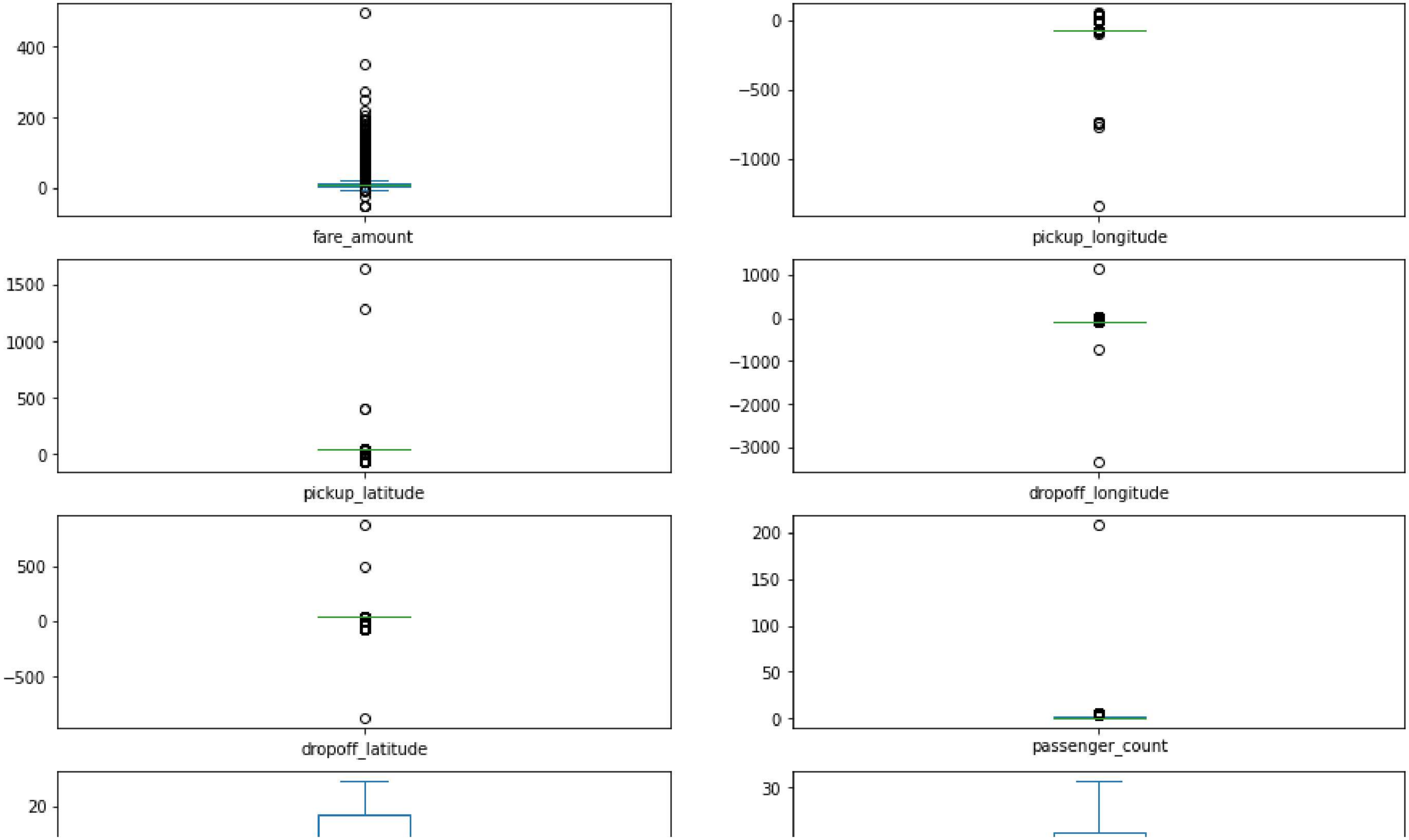
```
df.dtypes
```

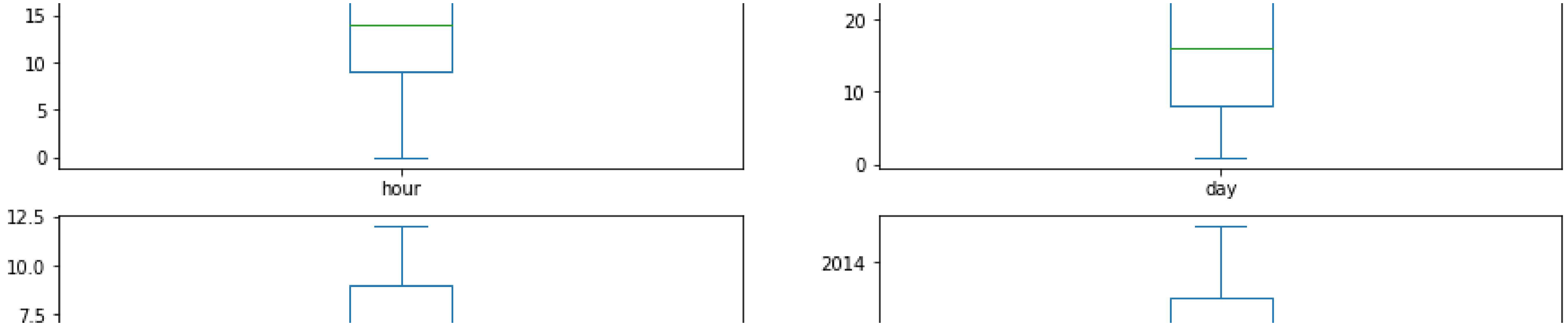
```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count      float64
hour                   int64
day                    int64
month                  int64
year                   int64
dayofweek              int64
dtype: object
```

Checking outliers and filling them

```
df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20)) #Boxplot to check t
```

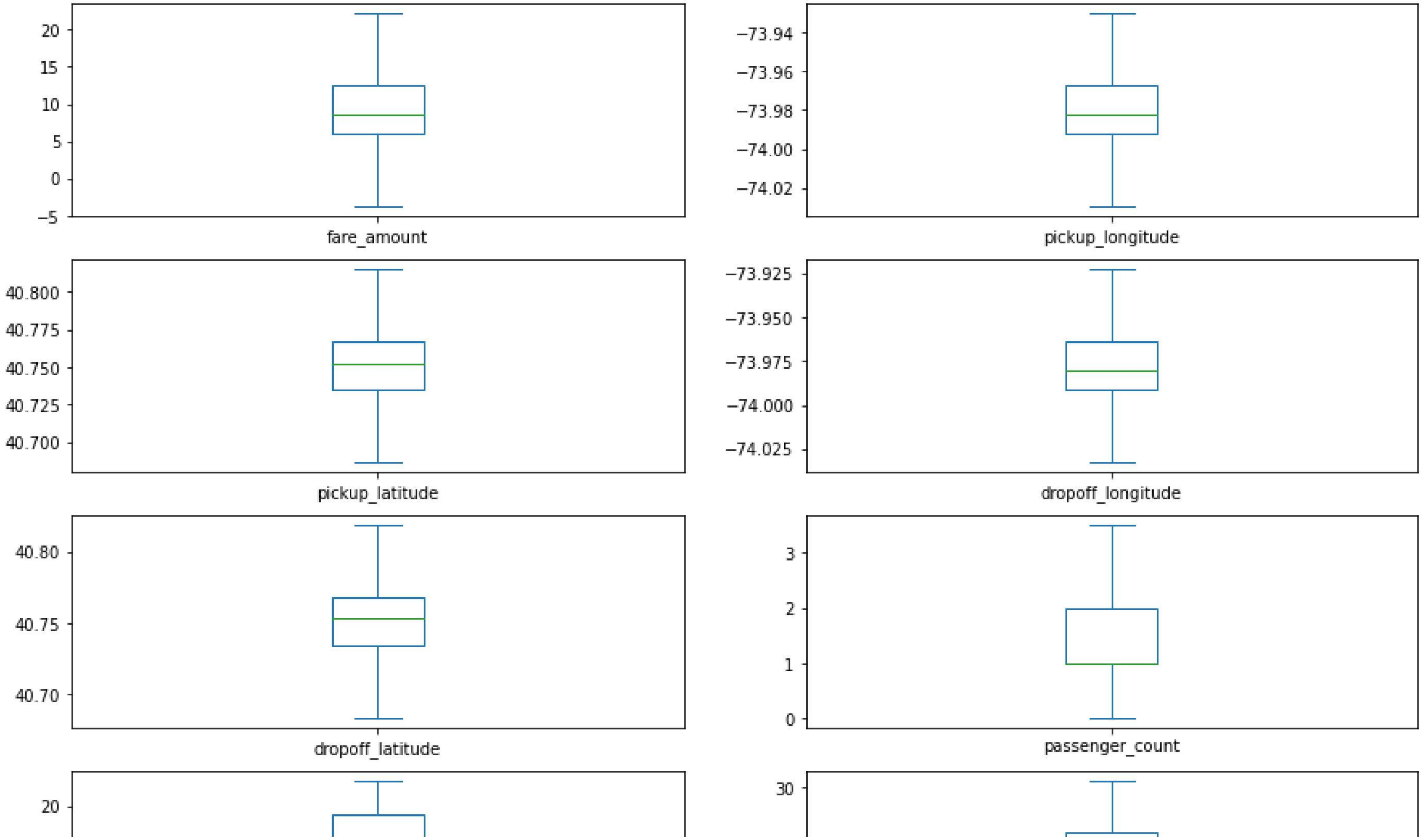
```
fare_amount            AxesSubplot(0.125,0.787927;0.352273x0.0920732)
pickup_longitude    AxesSubplot(0.547727,0.787927;0.352273x0.0920732)
pickup_latitude        AxesSubplot(0.125,0.677439;0.352273x0.0920732)
dropoff_longitude   AxesSubplot(0.547727,0.677439;0.352273x0.0920732)
dropoff_latitude       AxesSubplot(0.125,0.566951;0.352273x0.0920732)
passenger_count     AxesSubplot(0.547727,0.566951;0.352273x0.0920732)
hour                   AxesSubplot(0.125,0.456463;0.352273x0.0920732)
day                 AxesSubplot(0.547727,0.456463;0.352273x0.0920732)
month                  AxesSubplot(0.125,0.345976;0.352273x0.0920732)
year                AxesSubplot(0.547727,0.345976;0.352273x0.0920732)
dayofweek              AxesSubplot(0.125,0.235488;0.352273x0.0920732)
dtype: object
```

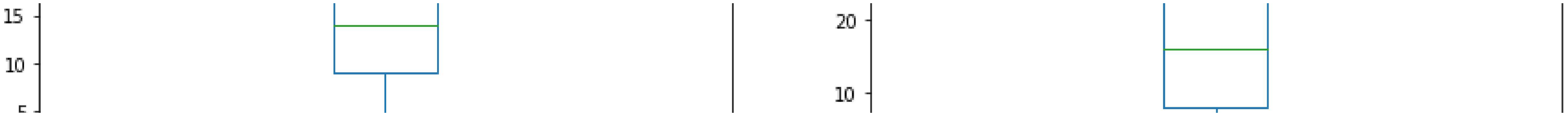




```
#Using the InterQuartile Range to fill the values
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    return df1

def treat_outliers_all(df1 , col_list):
    for c in col_list:
        df1 = remove_outlier(df , c)
    return df1


df = treat_outliers_all(df , df.iloc[: , 0::])


df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20)) #Boxplot shows that
```

```
fare_amount            AxesSubplot(0.125,0.787927;0.352273x0.0920732)
pickup_longitude    AxesSubplot(0.547727,0.787927;0.352273x0.0920732)
pickup_latitude        AxesSubplot(0.125,0.677439;0.352273x0.0920732)
dropoff_longitude   AxesSubplot(0.547727,0.677439;0.352273x0.0920732)
dropoff_latitude       AxesSubplot(0.125,0.566951;0.352273x0.0920732)
passenger_count     AxesSubplot(0.547727,0.566951;0.352273x0.0920732)
hour                   AxesSubplot(0.125,0.456463;0.352273x0.0920732)
day                 AxesSubplot(0.547727,0.456463;0.352273x0.0920732)
month                  AxesSubplot(0.125,0.345976;0.352273x0.0920732)
year                AxesSubplot(0.547727,0.345976;0.352273x0.0920732)
dayofweek              AxesSubplot(0.125,0.235488;0.352273x0.0920732)
dtype: object
```





```
!pip install haversine
import haversine as hs  #Calculate the distance using Haversine to calculate the distance between to points. Can't use Eucla
travel_dist = []
for pos in range(len(df['pickup_longitude'])):
        long1,lati1,long2,lati2 = [df['pickup_longitude'][pos],df['pickup_latitude'][pos],df['dropoff_longitude'][pos],df['d
        loc1=(lati1,long1)
        loc2=(lati2,long2)
        c = hs.haversine(loc1,loc2)
        travel_dist.append(c)

print(travel_dist)
df['dist_travel_km'] = travel_dist
df.head()
```

```
     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
     Collecting haversine
       Downloading haversine-2.7.0-py2.py3-none-any.whl (6.9 kB)
     Installing collected packages: haversine
     Successfully installed haversine-2.7.0
     IOPub data rate exceeded.
     The notebook server will temporarily stop sending output
```

```
#Uber doesn't travel over 130 kms so minimize the distance
df= df.loc[(df.dist_travel_km >= 1) | (df.dist_travel_km <= 130)]
print("Remaining observastions in the dataset:", df.shape)
```

```
     Remaining observastions in the dataset: (187717, 12)
     NotebookApp.rate_limit_window=3.0 (secs)
```

```
#Finding inccorect latitude (Less than or greater than 90) and longitude (greater than or less than 180)
incorrect_coordinates = df.loc[(df.pickup_latitude > 90) |(df.pickup_latitude < -90) |
                                   (df.dropoff_latitude > 90) |(df.dropoff_latitude < -90) |
                                   (df.pickup_longitude > 180) |(df.pickup_longitude < -180) |
                                   (df.dropoff_longitude > 90) |(df.dropoff_longitude < -90)
                                    ]


df.drop(incorrect_coordinates, inplace = True, errors = 'ignore')
```

```
df.head()
```

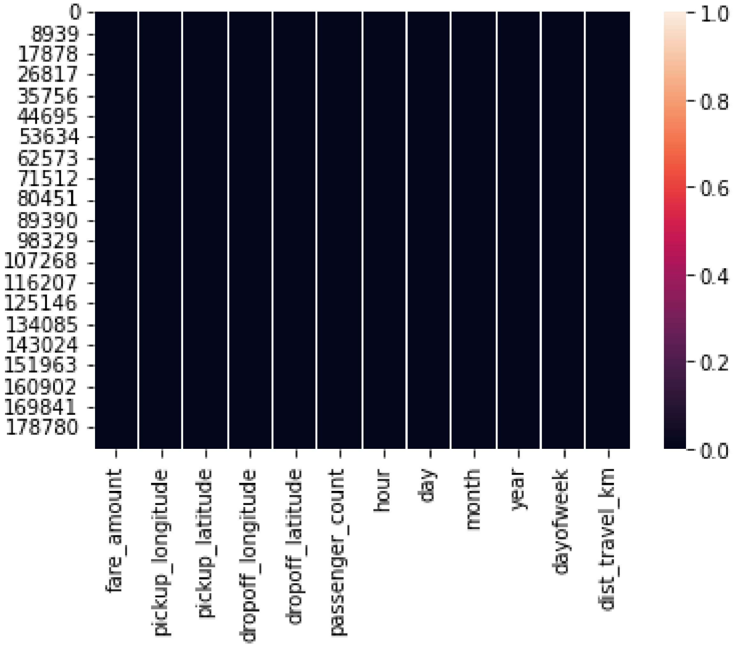
```
        fare_amount  pickup_longitude  pickup_latitude  dropoff_longitude  dropoff_latitude  passenger_count  hour  day  mor
```

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

```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      1
hour                 0
day                  0
month                0
year                 0
dayofweek            0
dist_travel_km       0
dtype: int64
```

```
sns.heatmap(df.isnull()) #Free for null values
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7febcb4f04d0>
```



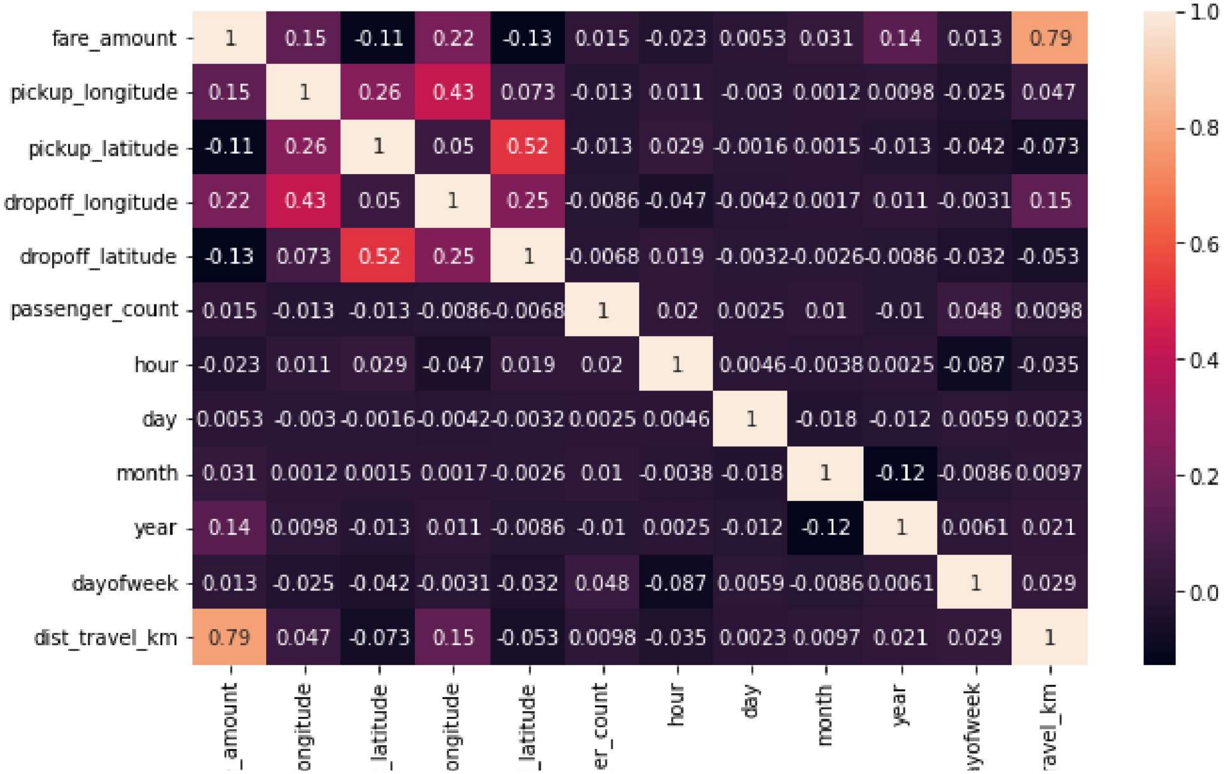
```
corr = df.corr() #Function to find the correlation
```

```
corr
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
fare_amount	1.000000	0.153739	-0.110943	0.217658	-0.126442	0.015481
pickup_longitude	0.153739	1.000000	0.260639	0.425748	0.073404	-0.012587
pickup_latitude	-0.110943	0.260639	1.000000	0.050062	0.516141	-0.012590
dropoff_longitude	0.217658	0.425748	0.050062	1.000000	0.246565	-0.008615
dropoff_latitude	-0.126442	0.073404	0.516141	0.246565	1.000000	-0.006843
passenger_count	0.015481	-0.012587	-0.012590	-0.008615	-0.006843	1.000000
hour	-0.023112	0.010873	0.028834	-0.046539	0.019404	0.020272
day	0.005341	-0.003039	-0.001598	-0.004191	-0.003238	0.002537
month	0.030533	0.001178	0.001495	0.001733	-0.002644	0.010320
year	0.140274	0.009791	-0.012692	0.010938	-0.008649	-0.010034
dayofweek	0.012615	-0.024642	-0.042372	-0.003101	-0.031581	0.048305
dist_travel_km	0.785796	0.047197	-0.072991	0.154418	-0.053017	0.009810

```
fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True) #Correlation Heatmap (Light values means highly correlated)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7febc4d67450>
```



Dividing the dataset into feature and target values

```
x = df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude','passenger_count','hour','day','month','
```

```
y = df['fare_amount']
```

Dividing the dataset into training and testing dataset

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.33)
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
```

```
y_train = np.nan_to_num(y_train)
```

```
X_train= np.nan_to_num(X_train)
```

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

```
LinearRegression()
```

```
regression.intercept_ #To find the linear intercept
```

```
3712.436030622846
```

```
regression.coef_ #To find the linear coeeficient
```

```
array([ 2.59282080e+01, -7.29704623e+00,  2.02139949e+01, -1.80678549e+01,
        6.63557805e-02,  7.43757744e-03,  3.34792249e-03,  5.70327286e-02,
        3.67382492e-01, -3.65053829e-02,  1.85403830e+00])
```

```
prediction = regression.predict(X_test) #To predict the target values
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but LinearRegression was
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print(prediction)
```

```
[10.24860732  6.74222458 10.52011102 ...  5.29117953 22.67254805
  8.45302783]
```

```
y_test
```

```
array([22.25,  5.7 ,  9.3 , ...,  4.5 , 22.25,  7.5 ])
```

Metrics Evaluation using R2, Mean Squared Error, Root Mean Sqared Error

```
from sklearn.metrics import r2_score
r2_score(y_test,prediction)
```

```
0.6629487983734947
```

```
from sklearn.metrics import mean_squared_error
MSE = mean_squared_error(y_test,prediction)
MSE
```

```
9.935937305343638
```

```
RMSE = np.sqrt(MSE)
RMSE
```

```
3.152132183989694
```

Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=100) #Here n_estimators means number of trees you want to build before making the pr
rf.fit(X_train,y_train)
```

```
RandomForestRegressor()
```

```
y_pred = rf.predict(X_test)
y_pred
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but RandomForestRegressor
```

```
  f"X has feature names, but {self.__class__.__name__} was fitted without"
array([ 7.501 ,  6.275 , 10.443 , ...,  5.098 , 22.0455,  8.16  ])
```

Metrics evaluatin for Random Forest

```
R2_Random = r2_score(y_test,y_pred)
R2_Random
```

```
0.7937849498842736
```

```
MSE_Random = mean_squared_error(y_test,y_pred)
MSE_Random
```

```
6.079016480227936
```

```
RMSE_Random = np.sqrt(MSE_Random)
RMSE_Random
```

```
2.4655661581527144
```

✓ 0s completed at 3:44 PM ● ×