Group B : MACHINE LEARNING

****

Assignment No: 1

Title Name: Predict the price of the Uber ride from a given pickup point to the agreed drop-off location.

Name: Aniket Rajani

Class : BE **Div**: 1 **Batch**: A

Roll No: 405A008

****

In [1]:

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

In [2]:

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

In [3]:

```
df.head()
df.info() #To get the required information of the dataset
df.columns #TO get number of columns in the dataset
df = df.drop(['Unnamed: 0', 'key'], axis= 1) #To drop unnamed column as it isn't req
df.head()
df.shape #To get the total (Rows,Columns)
df.dtypes #To get the type of each column
df.info()
df.describe() #To get statistics of each columns
```

```
<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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   fare_amount        200000 non-null  float64
 1   pickup_datetime    200000 non-null  object
 2   pickup_longitude   200000 non-null  float64
 3   pickup_latitude    200000 non-null  float64
 4   dropoff_longitude  199999 non-null  float64
 5   dropoff_latitude   199999 non-null  float64
 6   passenger_count    200000 non-null  int64
dtypes: float64(5), int64(1), object(1)
memory usage: 10.7+ MB
```

Out[3]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passen

count	200000.000000	200000.000000	200000.000000	199999.000000	199999.000000	2000
mean	11.359955	-72.527638	39.935885	-72.525292	39.923890	
std	9.901776	11.437787	7.720539	13.117408	6.794829	
min	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	
25%	6.000000	-73.992065	40.734796	-73.991407	40.733823	
50%	8.500000	-73.981823	40.752592	-73.980093	40.753042	
75%	12.500000	-73.967154	40.767158	-73.963658	40.768001	

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passen
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	2

In [4]:
```
df.isnull().sum()
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()
df.dtypes
```

Out[4]:
```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
dtype: object
```

In [5]:
```
df.pickup_datetime  =  pd.to_datetime(df.pickup_datetime,  errors='coerce')
df.dtypes
```

Out[5]:
```
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 [6]:
```
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()
```

Out[6]:

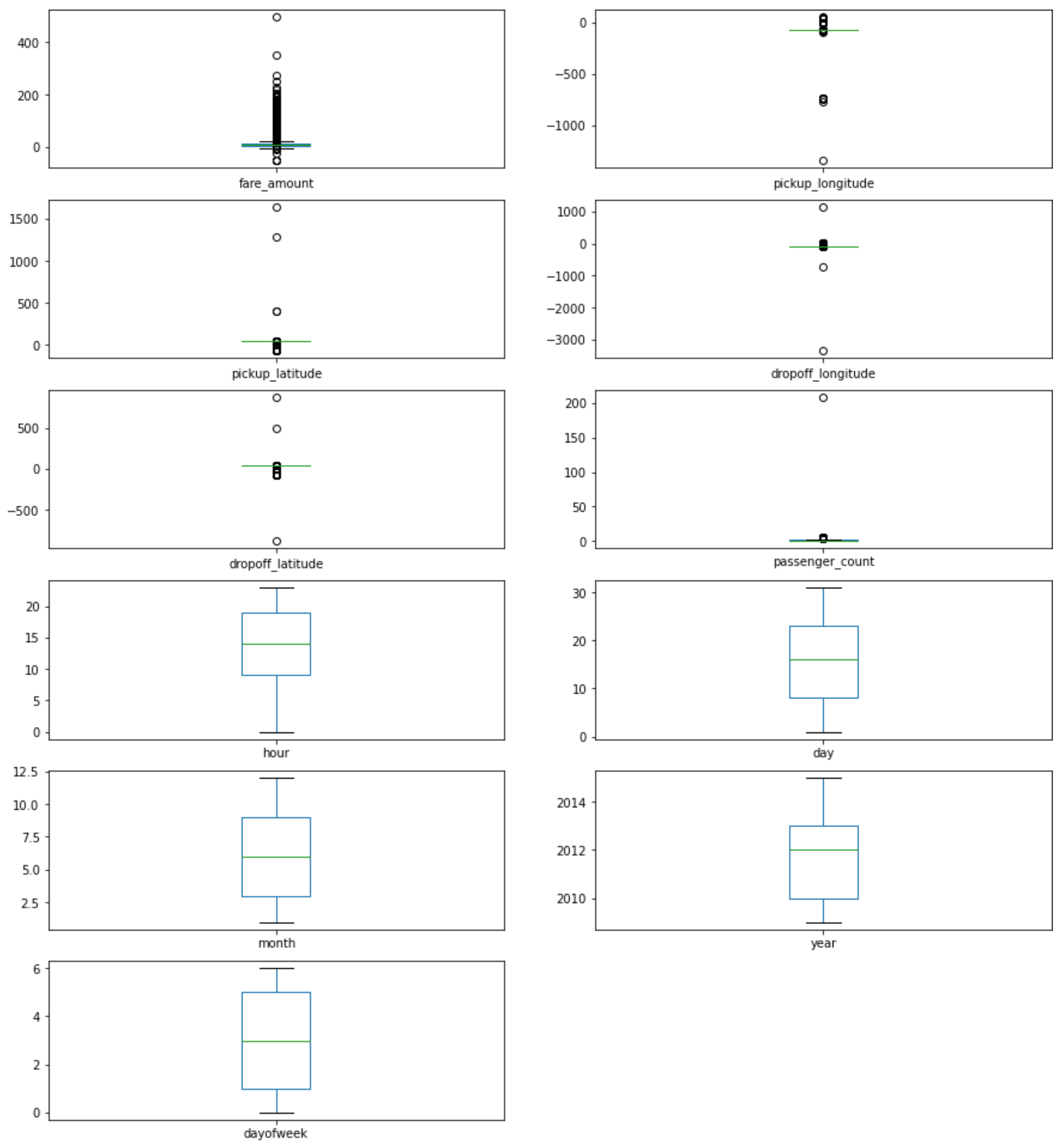
	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latit
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723
1	7.7	2009-07-17 20:	-73.994355	40.728225	-73.994710	40.750
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761

In [7]:
```
df  =  df.drop('pickup_datetime',axis=1)
df.head()
df.dtypes
```

```
Out[7]: fare_amount          float64
        pickup_longitude     float64
        pickup_latitude      float64
        dropoff_longitude    float64
        dropoff_latitude     float64
        passenger_count        int64
        hour                   int64
        day                    int64
        month                  int64
        year                   int64
        dayofweek              int64
        dtype: object
```

```
In [8]: df.plot(kind = "box",subplots = True,layout = (7,2),figsize=(15,20))
```

```
Out[8]: 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
```

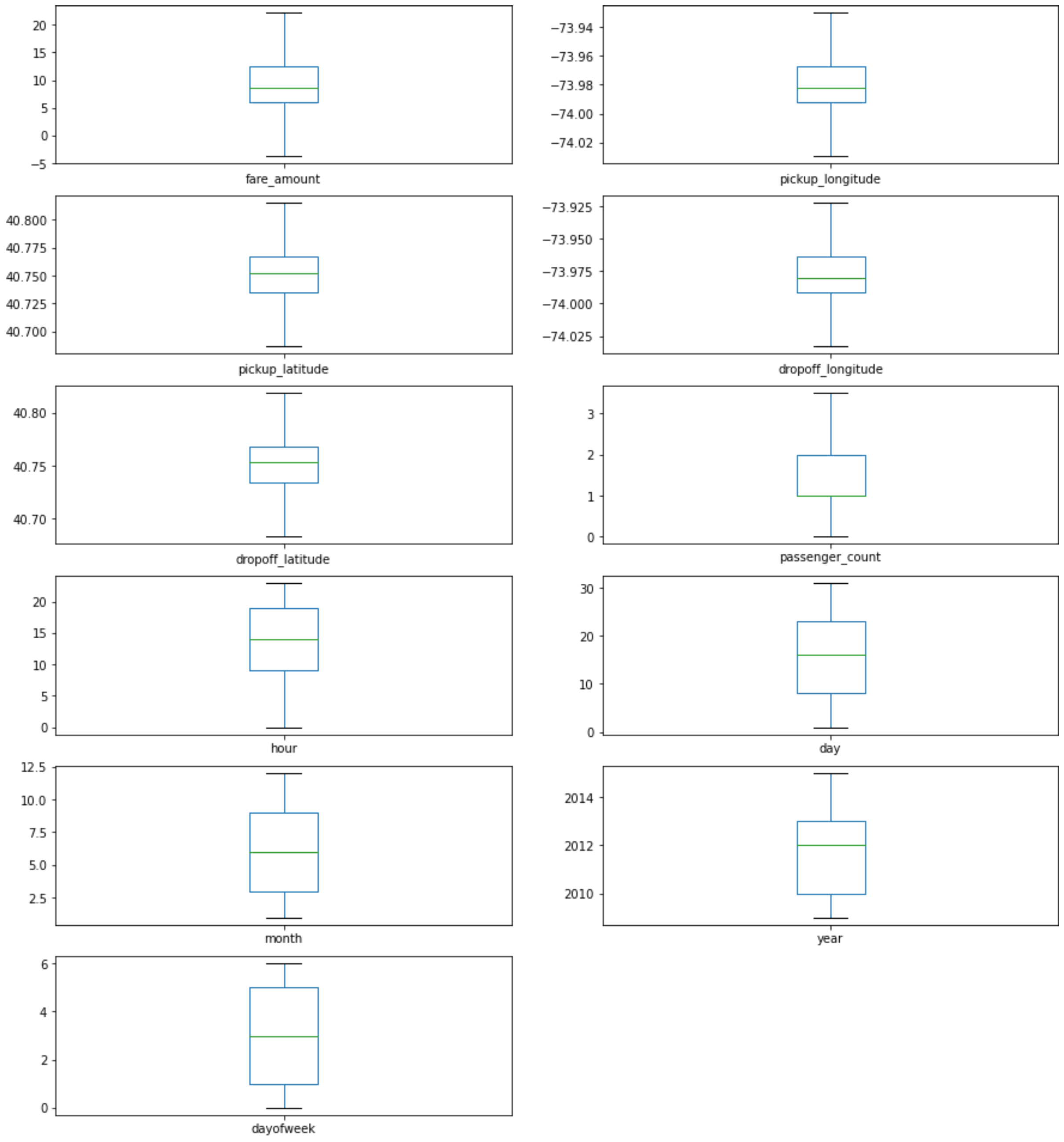


In [9]:
```
def remove_outlier(df1 , col):
 Q1 = df1[col].quantile(0.25)
 Q3 = df1[col].quantile(0.75)
 IQR = Q3 - Q1
 lower_whisker = Q1-1.5*IQR
 upper_whisker = Q3+1.5*IQR
 df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
 return df1
```

In [12]:
```
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
```

Out[12]:
```
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
```



In [14]:
```
pip install haversine
```

```
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
Note: you may need to restart the kernel to use updated packages.
```

In [22]:
```
import haversine as hs #Calculate the distance using Haversine to calculate the dist
travel_dist = []
for pos in range(len(df['pickup_longitude'])):
    long1,lati1,long2,lati2 = [df['pickup_longitude'][pos],df['pickup_latitude'][pos
    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()
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Out[22]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_co
0	7.5	-73.999817	40.738354	-73.999512	40.723217	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	
4	16.0	-73.929786	40.744085	-73.973082	40.761247	

In [23]:

```
df= df.loc[(df.dist_travel_km >= 1) | (df.dist_travel_km <= 130)]
print("Remaining observastions in the dataset:", df.shape)
```

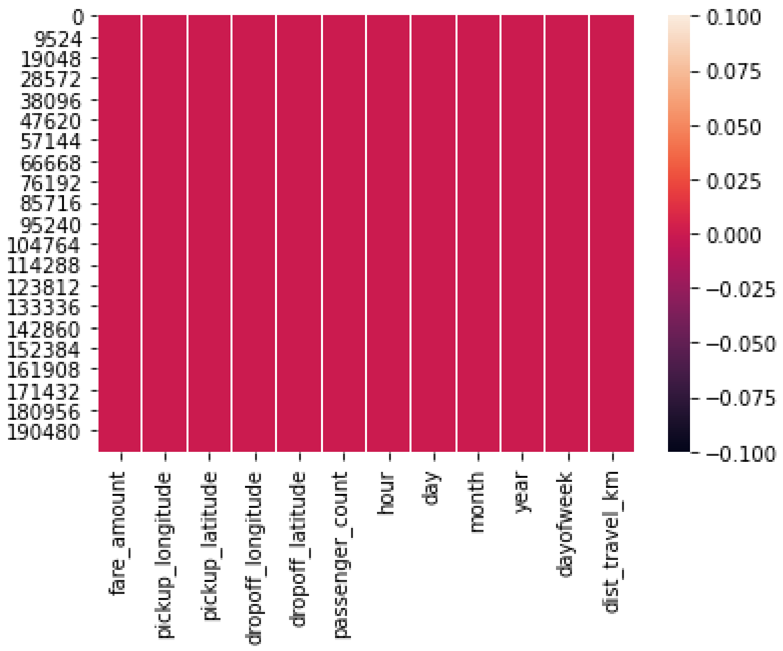
```
Remaining observastions in the dataset: (200000, 12)
```

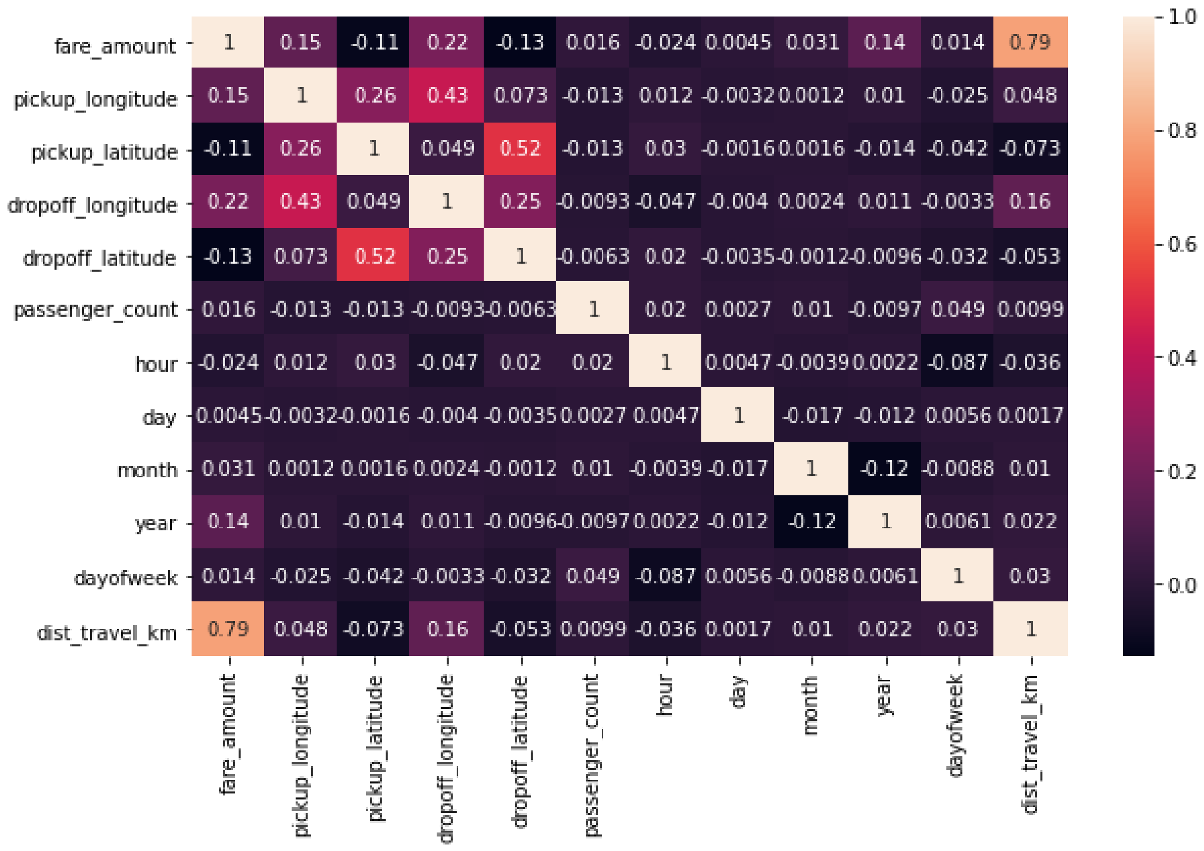
In [26]:

```
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()
df.isnull().sum()
sns.heatmap(df.isnull()) #Free for null values
corr = df.corr() #Function to find the correlation
corr
fig,axis = plt.subplots(figsize = (10,6))
sns.heatmap(df.corr(),annot = True)
```

Out[26]:

```
<AxesSubplot:>
```





In [28]:
```
x  =  df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude',
y = df['fare_amount']
```

In [29]:
```
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)
```

In [31]:
```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(X_train,y_train)
regression.coef_ #To find the linear coeeficient
regression.intercept_ #To find the linear intercept
prediction = regression.predict(X_test) #To predict  the  target  values
```

```
print(prediction)
y_test
```

```
[17.28050585 11.44946862 13.22284482 ... 15.04497674 18.34524502
  9.91445235]
```

Out[31]:
```
30406     18.50
122525    13.00
145989    22.25
50071     17.50
2065       4.50
          ...
95147      4.50
107084    14.10
36958     11.50
65775     14.10
39173      8.50
Name: fare_amount, Length: 66000, dtype: float64
```

In [33]:
```
from sklearn.metrics  import  r2_score
r2_score(y_test,prediction)
from sklearn.metrics  import  mean_squared_error
MSE =  mean_squared_error(y_test,prediction)
MSE
RMSE = np.sqrt(MSE)
RMSE
3.156187085348032
```

Out[33]:
```
3.156187085348032
```

In [ ]: