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-dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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

```
#importing the dataset
df  = pd.read_csv("/content/drive/MyDrive/ML/uber.csv")
```

1. Pre-process the dataset.

```
df.head()
```

```
   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  
```

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

```
<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.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 it isn't required
```

```
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  
```

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

```
(200000, 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        int64
dtype: object
```

```
df.info()
```

```
<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
```

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

```
         fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
count  200000.000000     200000.000000    200000.000000
199999.000000
mean       11.359955        -72.527638        39.935885         -
72.525292
std         9.901776         11.437787         7.720539
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    200000.000000
```

```
mean          39.923890        1.684535
std            6.794829        1.385997
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
```

Filling Missing values

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

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

```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(),inpl
ace = 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      0
dtype: int64
```

```
df.dtypes
```

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

To segregate each time of date and time

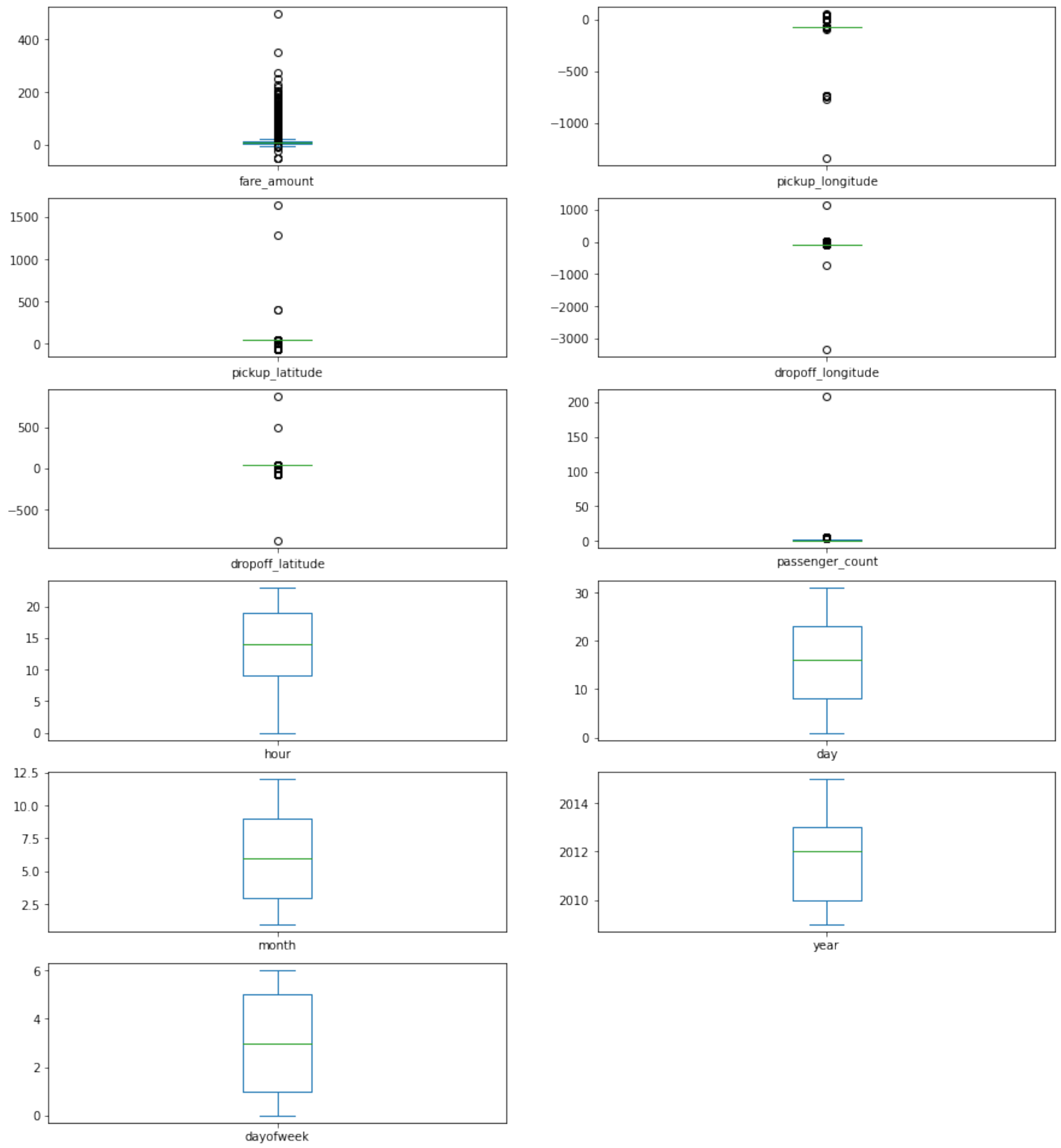
```
df= df.assign(hour = df.pickup_datetime.dt.hour,
             day= df.pickup_datetime.dt.day,
             month = df.pickup_datetime.dt.month,
             year = df.pickup_datetime.dt.year,
             dayofweek = df.pickup_datetime.dt.dayofweek)

df.head()

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

   dropoff_longitude  dropoff_latitude  passenger_count  hour  day
month  \
0         -73.999512         40.723217                1    19    7
5
1         -73.994710         40.750325                1    20   17
7
2         -73.962565         40.772647                1    21   24
8
3         -73.965316         40.803349                3     8   26
6
4         -73.973082         40.761247                5    17   28
8

   year  dayofweek
0  2015          3
1  2009          4
2  2009          0
3  2009          4
4  2014          3
```

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

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

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

df.dtypes

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
```

Checking outliers and filling them

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

```
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
```

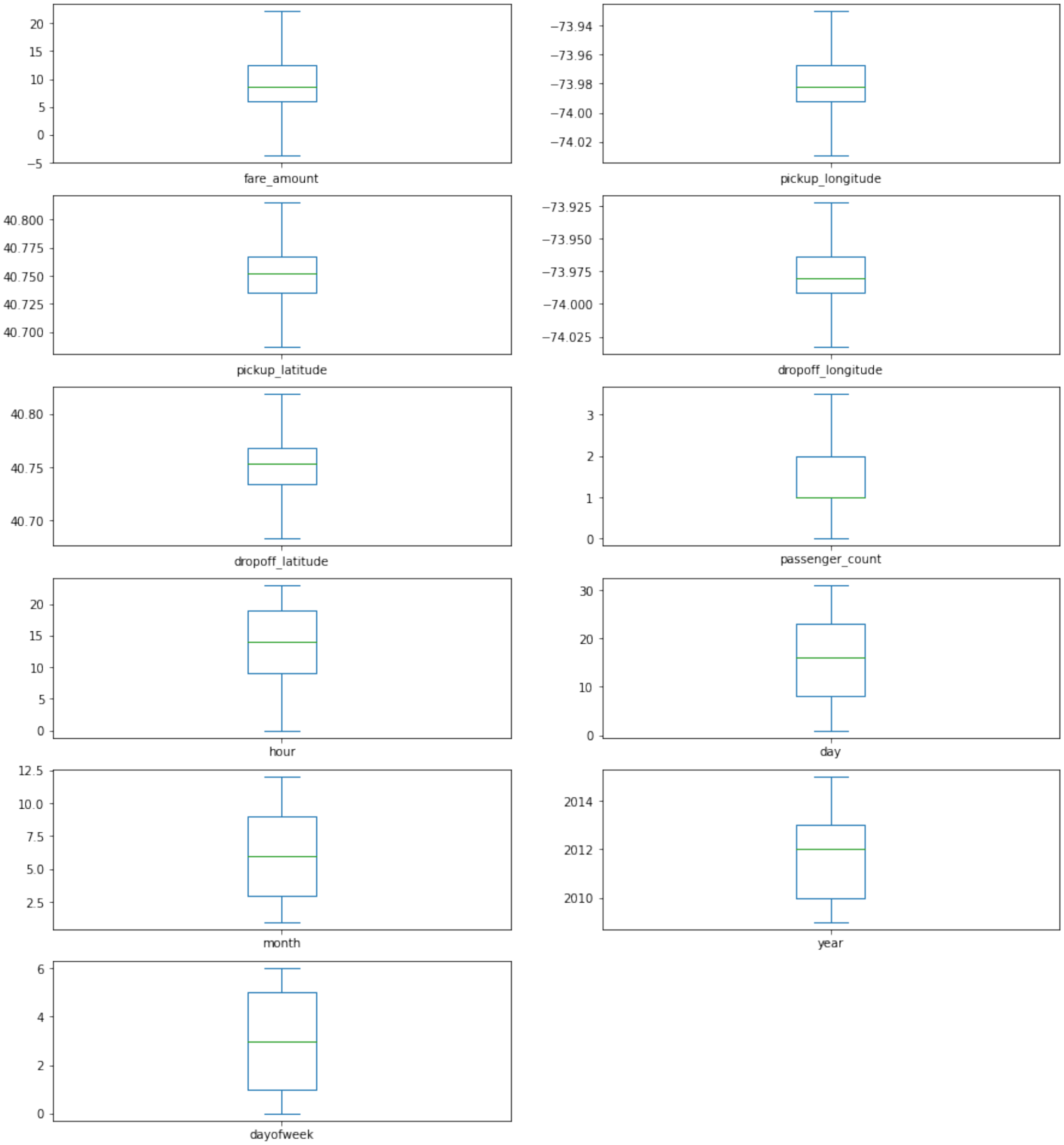


```python
#Using the InterQuartile Range to fill the values
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1-1.5*IQR
    upper_whisker = Q3+1.5*IQR
    df[col] = np.clip(df1[col] , lower_whisker , upper_whisker)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

```
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
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: haversine in
/usr/local/lib/python3.7/dist-packages (2.7.0)
```

```
#pip install haversine
import haversine as hs  #Calculate the distance using Haversine to
calculate the distance between to points. Can't use Eucladian as it is
for flat surface.
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['dropoff_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)


   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        -73.929786        40.744085         -73.973082


   dropoff_latitude  passenger_count  hour  day  month  year
dayofweek  \
0         40.723217              1.0    19    7      5  2015
3
1         40.750325              1.0    20   17      7  2009
4
2         40.772647              1.0    21   24      8  2009
0
3         40.803349              3.0     8   26      6  2009
4
4         40.761247              3.5    17   28      8  2014
3

   dist_travel_km
0        1.683325
1        2.457593
```

```
2        5.036384
3        1.661686
4        4.116088
```

```
#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: (200000, 12)
```

```
#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  \
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        -73.929786        40.744085         -73.973082


   dropoff_latitude  passenger_count  hour  day  month  year
dayofweek  \
0         40.723217              1.0    19    7      5  2015
3
1         40.750325              1.0    20   17      7  2009
4
2         40.772647              1.0    21   24      8  2009
0
3         40.803349              3.0     8   26      6  2009
4
4         40.761247              3.5    17   28      8  2014
3
```

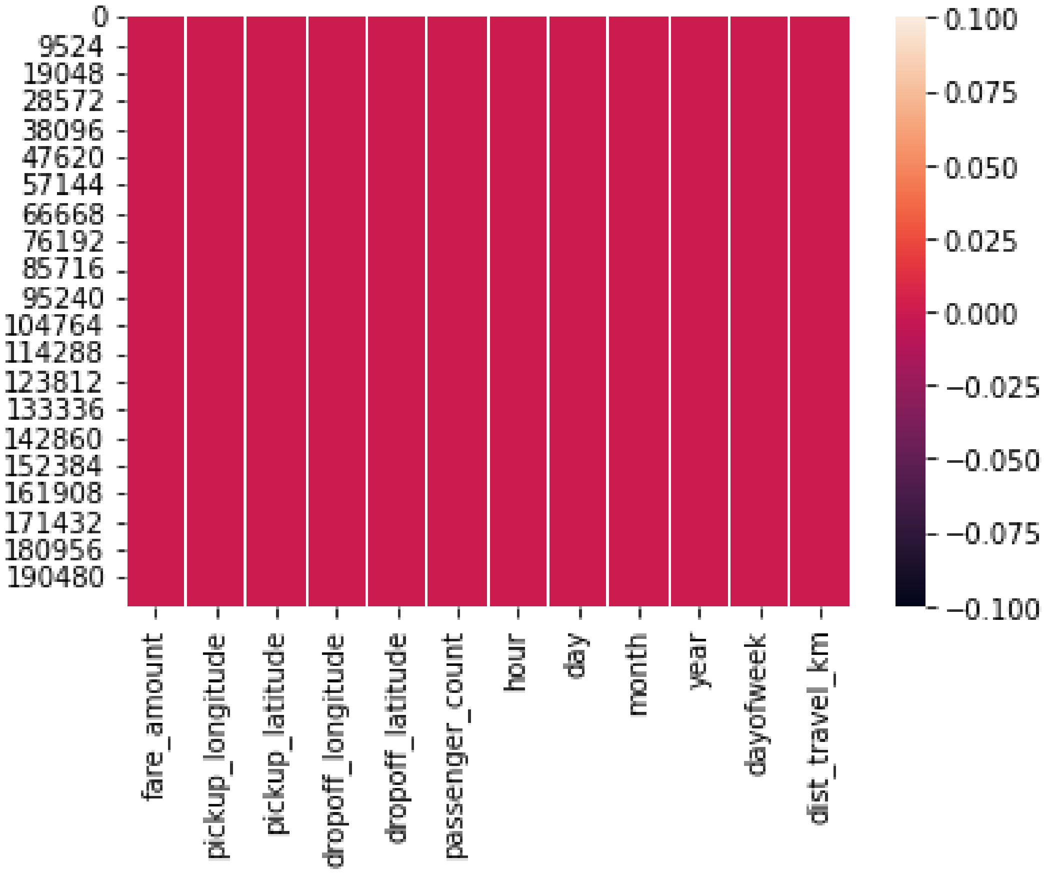
```
   dist_travel_km
0        1.683325
1        2.457593
2        5.036384
3        1.661686
4        4.116088
```

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

```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
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 0x7fbe4defdad0>
```



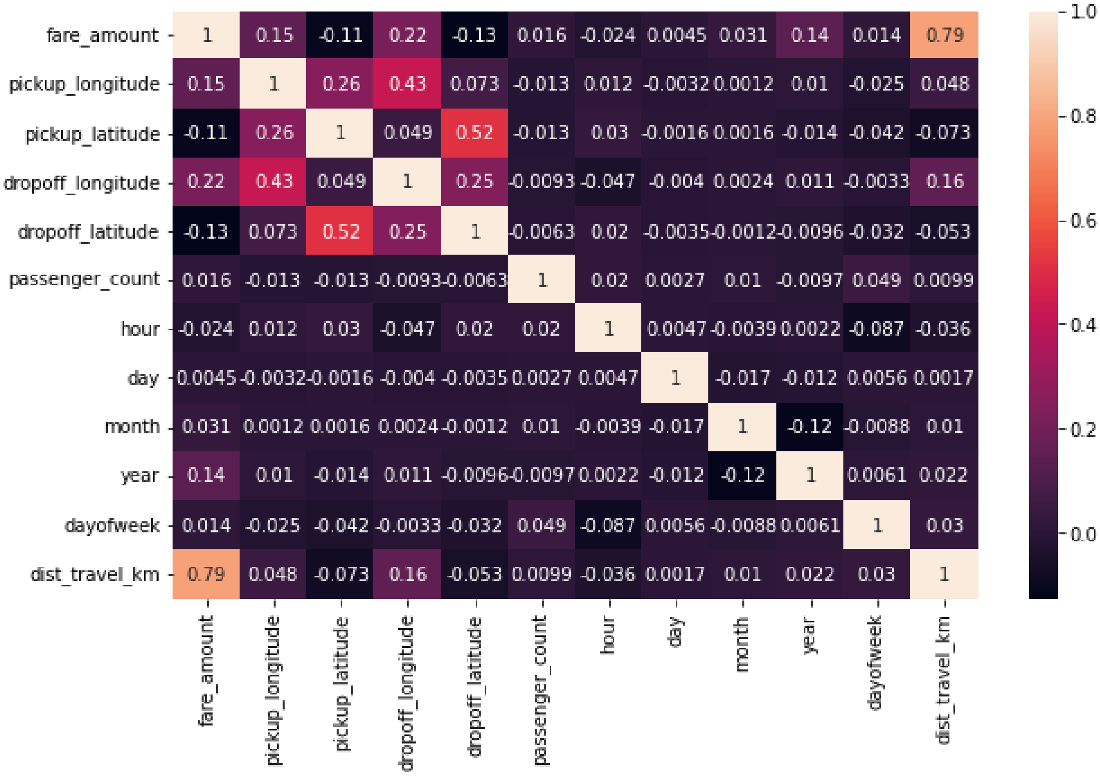
```
corr = df.corr() #Function to find the correlation

corr

                   fare_amount  pickup_longitude  pickup_latitude  \
fare_amount           1.000000          0.154069        -0.110842
pickup_longitude      0.154069          1.000000         0.259497
pickup_latitude      -0.110842          0.259497         1.000000
dropoff_longitude     0.218675          0.425619         0.048889
dropoff_latitude     -0.125898          0.073290         0.515714
passenger_count       0.015778         -0.013213        -0.012889
hour                 -0.023623          0.011579         0.029681
day                   0.004534         -0.003204        -0.001553
month                 0.030817          0.001169         0.001562
year                  0.141277          0.010198        -0.014243
dayofweek             0.013652         -0.024652        -0.042310
dist_travel_km        0.786385          0.048446        -0.073362

                   dropoff_longitude  dropoff_latitude
passenger_count  \
fare_amount                 0.218675         -0.125898
0.015778
pickup_longitude            0.425619          0.073290        -
0.013213
```

```
pickup_latitude             0.048889          0.515714        -
0.012889
dropoff_longitude           1.000000          0.245667        -
0.009303
dropoff_latitude            0.245667          1.000000        -
0.006308
passenger_count            -0.009303         -0.006308
1.000000
hour                       -0.046558          0.019783
0.020274
day                        -0.004007         -0.003479
0.002712
month                       0.002391         -0.001193
0.010351
year                        0.011346         -0.009603        -
0.009749
dayofweek                  -0.003336         -0.031919
0.048550
dist_travel_km              0.155191         -0.052701
0.009884

                       hour       day     month      year
dayofweek  \
fare_amount       -0.023623  0.004534  0.030817  0.141277   0.013652

pickup_longitude   0.011579 -0.003204  0.001169  0.010198  -0.024652

pickup_latitude    0.029681 -0.001553  0.001562 -0.014243  -0.042310

dropoff_longitude -0.046558 -0.004007  0.002391  0.011346  -0.003336

dropoff_latitude   0.019783 -0.003479 -0.001193 -0.009603  -0.031919

passenger_count    0.020274  0.002712  0.010351 -0.009749   0.048550

hour               1.000000  0.004677 -0.003926  0.002156  -0.086947

day                0.004677  1.000000 -0.017360 -0.012170   0.005617

month             -0.003926 -0.017360  1.000000 -0.115859  -0.008786

year               0.002156 -0.012170 -0.115859  1.000000   0.006113

dayofweek         -0.086947  0.005617 -0.008786  0.006113   1.000000

dist_travel_km    -0.035708  0.001709  0.010050  0.022294   0.030382


                   dist_travel_km
```

```
fare_amount            0.786385
pickup_longitude       0.048446
pickup_latitude       -0.073362
dropoff_longitude      0.155191
dropoff_latitude      -0.052701
passenger_count        0.009884
hour                  -0.035708
day                    0.001709
month                  0.010050
year                   0.022294
dayofweek              0.030382
dist_travel_km         1.000000
```

```
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 0x7fbe4b72acd0>
```



	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	hour	day	month	year	dayofweek	dist_travel_km
fare_amount	1	0.15	-0.11	0.22	-0.13	0.016	-0.024	0.0045	0.031	0.14	0.014	0.79
pickup_longitude	0.15	1	0.26	0.43	0.073	-0.013	0.012	-0.0032	0.0012	0.01	-0.025	0.048
pickup_latitude	-0.11	0.26	1	0.049	0.52	-0.013	0.03	-0.0016	0.0016	-0.014	-0.042	-0.073
dropoff_longitude	0.22	0.43	0.049	1	0.25	-0.0093	-0.047	-0.004	0.0024	0.011	-0.0033	0.16
dropoff_latitude	-0.13	0.073	0.52	0.25	1	-0.0063	0.02	-0.0035	-0.0012	-0.0096	-0.032	-0.053
passenger_count	0.016	-0.013	-0.013	-0.0093	-0.0063	1	0.02	0.0027	0.01	-0.0097	0.049	0.0099
hour	-0.024	0.012	0.03	-0.047	0.02	0.02	1	0.0047	-0.0039	0.0022	-0.087	-0.036
day	0.0045	-0.0032	-0.0016	-0.004	-0.0035	0.0027	0.0047	1	-0.017	-0.012	0.0056	0.0017
month	0.031	0.0012	0.0016	0.0024	-0.0012	0.01	-0.0039	-0.017	1	-0.12	-0.0088	0.01
year	0.14	0.01	-0.014	0.011	-0.0096	-0.0097	0.0022	-0.012	-0.12	1	0.0061	0.022
dayofweek	0.014	-0.025	-0.042	-0.0033	-0.032	0.049	-0.087	0.0056	-0.0088	0.0061	1	0.03
dist_travel_km	0.79	0.048	-0.073	0.16	-0.053	0.0099	-0.036	0.0017	0.01	0.022	0.03	1

Dividing the dataset into feature and target values

```
x =
df[['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_
latitude','passenger_count','hour','day','month','year','dayofweek','d
ist_travel_km']]
```

```
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()

regression.fit(X_train,y_train)

LinearRegression()

regression.intercept_ #To find the linear intercept

3707.6952527541293

regression.coef_ #To find the linear coeeficient

array([ 2.54362240e+01, -7.13054295e+00,  2.05289125e+01, -
1.83667516e+01,
        5.98983223e-02,  7.34373566e-03,  3.36220771e-03,
5.64469103e-02,
        3.65922129e-01, -3.27782176e-02,  1.83970573e+00])

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

print(prediction)

[ 6.4902753  14.68401463  5.79106026 ... 10.65066025 15.71862999
  8.32175215]

y_test

149665     5.70
12889     19.87
113519     4.90
57831      8.50
145652    22.25
          ...
173368    10.50
198227     8.90
90825     12.50
66427     18.50
40078      6.90
Name: fare_amount, Length: 66000, dtype: float64
```

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

```
from sklearn.metrics import r2_score

r2_score(y_test,prediction)

0.6617231754196249
```

```
from sklearn.metrics import mean_squared_error

MSE = mean_squared_error(y_test,prediction)

MSE

9.926619175564893

RMSE = np.sqrt(MSE)

RMSE

3.150653769547662
```

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 prediction

rf.fit(X_train,y_train)

RandomForestRegressor()

y_pred = rf.predict(X_test)

y_pred

array([ 6.69  , 16.7574,  4.779 , ..., 14.2275, 18.2475,  8.312 ])
```

Metrics evaluatin for Random Forest

```
R2_Random = r2_score(y_test,y_pred)

R2_Random

0.7939576374520466

MSE_Random = mean_squared_error(y_test,y_pred)

MSE_Random

6.046243545014832

RMSE_Random = np.sqrt(MSE_Random)

RMSE_Random

2.458911048617829
```