```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
#We do not want to see warnings
warnings.filterwarnings("ignore")

#import data
data = pd.read_csv("uber.csv")

#Create a data copy
df = data.copy()

#Print data
df.head
```

```
<bound method NDFrame.head of         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
...            ...                            ...          ...
199995    42598914   2012-10-28 10:49:00.00000053          3.0
199996    16382965    2014-03-14 01:09:00.0000008          7.5
199997    27804658   2009-06-29 00:42:00.00000078         30.9
199998    20259894    2015-05-20 14:56:25.0000004         14.5
199999    11951496   2010-05-15 04:08:00.00000076         14.1

                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
...                         ...               ...              ...
199995  2012-10-28 10:49:00 UTC        -73.987042        40.739367
199996  2014-03-14 01:09:00 UTC        -73.984722        40.736837
199997  2009-06-29 00:42:00 UTC        -73.986017        40.756487
199998  2015-05-20 14:56:25 UTC        -73.997124        40.725452
199999  2010-05-15 04:08:00 UTC        -73.984395        40.720077

        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
```

```
...                   ...              ...              ...
199995         -73.986525        40.740297                1
199996         -74.006672        40.739620                1
199997         -73.858957        40.692588                2
199998         -73.983215        40.695415                1
199999         -73.985508        40.768793                1

[200000 rows x 9 columns]>
```

```
#Get Info
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
```

```
#pickup_datetime is not in required data format
df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"])

df.info()
```

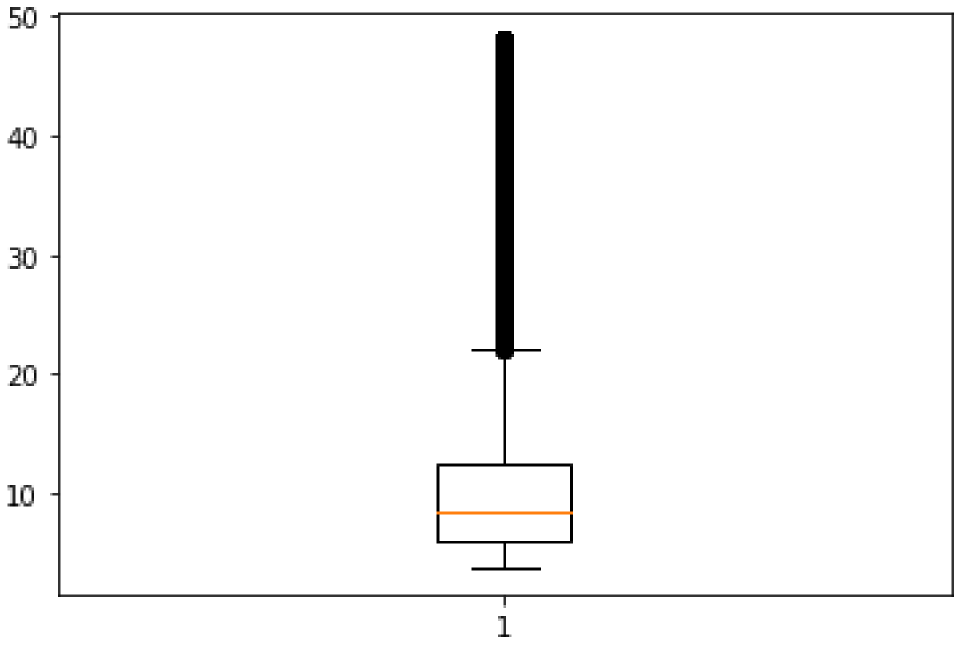
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype              
---  ------             --------------   -----              
 0   Unnamed: 0         200000 non-null  int64              
 1   key                200000 non-null  object             
 2   fare_amount        200000 non-null  float64            
 3   pickup_datetime    200000 non-null  datetime64[ns, UTC]
 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: datetime64[ns, UTC](1), float64(5), int64(2), object(1)
memory usage: 13.7+ MB
```

```
#Statistics of data
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

#Number of missing values
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

#Correlation
df.corr()

                   Unnamed: 0  fare_amount  pickup_longitude
pickup_latitude  \
Unnamed: 0           1.000000    -0.000223         -0.000266
0.000061
fare_amount         -0.000223     1.000000          0.004654        -
0.003154
pickup_longitude    -0.000266     0.004654          1.000000        -
0.806902
pickup_latitude      0.000061    -0.003154         -0.806902
1.000000
dropoff_longitude   -0.000310     0.003021          0.830658        -
0.770049
dropoff_latitude     0.000938    -0.004621         -0.844705
0.691893
passenger_count      0.002311     0.010705         -0.000644        -
0.001441

                   dropoff_longitude  dropoff_latitude
passenger_count
Unnamed: 0                 -0.000310          0.000938
0.002311
fare_amount                 0.003021         -0.004621
0.010705
pickup_longitude            0.830658         -0.844705        -
0.000644
pickup_latitude            -0.770049          0.691893        -
0.001441
dropoff_longitude           1.000000         -0.912750
0.000105
dropoff_latitude           -0.912750          1.000000        -
0.000726
passenger_count             0.000105         -0.000726
1.000000
```

```
#Drop the rows with missing values
df.dropna(inplace=True)
```

```
plt.boxplot(df['fare_amount'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x241e10fadc0>,
  <matplotlib.lines.Line2D at 0x241e11130d0>],
 'caps': [<matplotlib.lines.Line2D at 0x241e1113460>,
  <matplotlib.lines.Line2D at 0x241e11137f0>],
 'boxes': [<matplotlib.lines.Line2D at 0x241e10fa970>],
 'medians': [<matplotlib.lines.Line2D at 0x241e1113b80>],
 'fliers': [<matplotlib.lines.Line2D at 0x241e1113f10>],
 'means': []}
```



```
#Remove Outliers
q_low = df["fare_amount"].quantile(0.01)
q_hi  = df["fare_amount"].quantile(0.99)

df = df[(df["fare_amount"] < q_hi) & (df["fare_amount"] > q_low)]

#Check the missing values now
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
```

```
#Time to apply learning models
from sklearn.model_selection import train_test_split

#Take x as predictor variable
x = df.drop("fare_amount", axis = 1)
#And y as target variable
y = df['fare_amount']

#Necessary to apply model
x['pickup_datetime'] =
pd.to_numeric(pd.to_datetime(x['pickup_datetime']))
x = x.loc[:, x.columns.str.contains('^Unnamed')]
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 1)
```

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

```
lrmodel = LinearRegression()
lrmodel.fit(x_train, y_train)
```

```
LinearRegression()
```

```
#Prediction
predict = lrmodel.predict(x_test)
```

```
#Check Error
from sklearn.metrics import mean_squared_error
lrmodelrmse = np.sqrt(mean_squared_error(predict, y_test))
print("RMSE error for the model is ", lrmodelrmse)
```

```
RMSE error for the model is  7.083585521002763
```

```
#Let's Apply Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
rfrmodel = RandomForestRegressor(n_estimators = 100, random_state =
101)
```

```
#Fit the Forest
rfrmodel.fit(x_train, y_train)
rfrmodel_pred = rfrmodel.predict(x_test)
```

```
#Errors for the forest
rfrmodel_rmse = np.sqrt(mean_squared_error(rfrmodel_pred, y_test))
print("RMSE value for Random Forest is:",rfrmodel_rmse)
```

```
RMSE value for Random Forest is: 8.565996490346976
```