```
#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()
```

```
   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
```

```
#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.select_dtypes(include=[np.number]).corr()
```

```
                   Unnamed: 0  fare_amount  pickup_longitude
pickup_latitude  \
Unnamed: 0           1.000000     0.000589          0.000230        -
0.000341
fare_amount          0.000589     1.000000          0.010457        -
0.008481
pickup_longitude     0.000230     0.010457          1.000000        -
0.816461
pickup_latitude     -0.000341    -0.008481         -0.816461
1.000000
dropoff_longitude    0.000270     0.008986          0.833026        -
0.774787
dropoff_latitude     0.000271    -0.011014         -0.846324
0.702367
passenger_count      0.002257     0.010150         -0.000414        -
0.001560

                   dropoff_longitude  dropoff_latitude
passenger_count
Unnamed: 0                  0.000270          0.000271
0.002257
fare_amount                 0.008986         -0.011014
0.010150
```

```
pickup_longitude             0.833026         -0.846324         -
0.000414
pickup_latitude             -0.774787          0.702367         -
0.001560
dropoff_longitude            1.000000         -0.917010
0.000033
dropoff_latitude            -0.917010          1.000000         -
0.000659
passenger_count              0.000033         -0.000659
1.000000
```

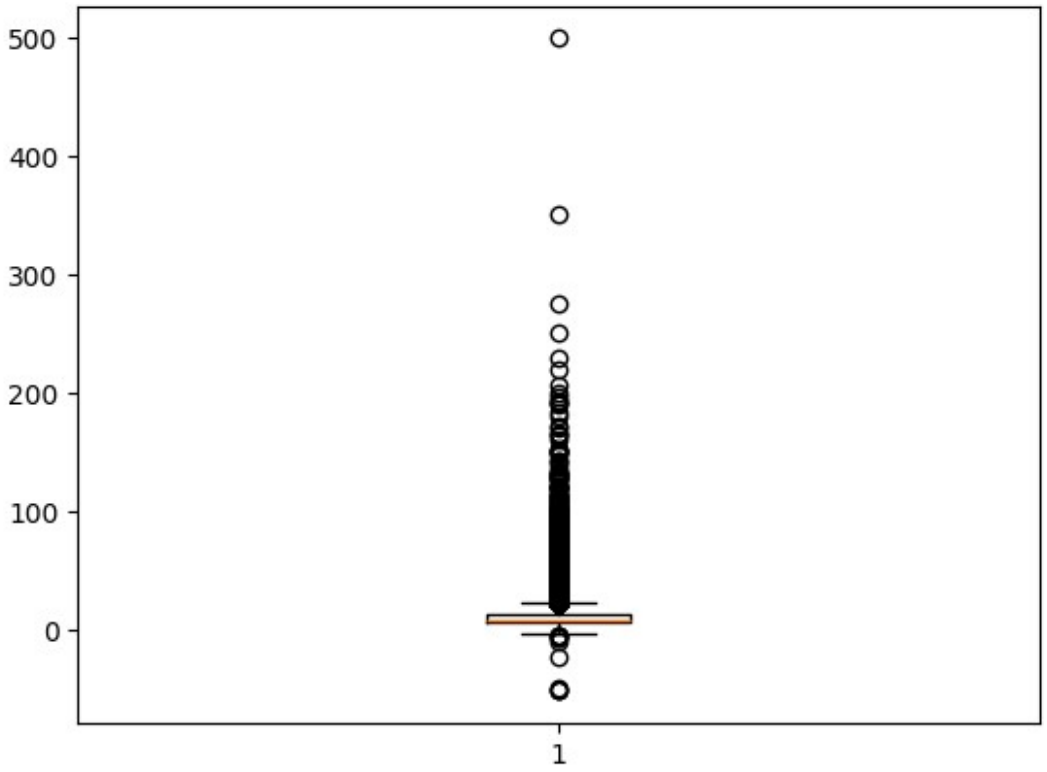
```
print(df.columns)
```

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

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

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

```
{'whiskers': [<matplotlib.lines.Line2D at 0x2174a300cd0>,
  <matplotlib.lines.Line2D at 0x2174a300e10>],
 'caps': [<matplotlib.lines.Line2D at 0x2174a300f50>,
  <matplotlib.lines.Line2D at 0x2174a301090>],
 'boxes': [<matplotlib.lines.Line2D at 0x2174a2f0b90>],
 'medians': [<matplotlib.lines.Line2D at 0x2174a3011d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2174a301310>],
 '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)

# evaluation

from sklearn.metrics import mean_squared_error, r2_score

lr_rmse = np.sqrt(mean_squared_error(y_test, predict))
lr_r2 = r2_score(y_test, predict)

print("Linear Regression → RMSE:", lr_rmse, "R²:", lr_r2)
```

```
Linear Regression → RMSE: 8.063863046328835 R²: -2.6395537326528995e-
05
```

```
#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)

rfr_rmse = np.sqrt(mean_squared_error(y_test, rfrmodel_pred))
rfr_r2 = r2_score(y_test, rfrmodel_pred)

print("Random Forest → RMSE:", rfr_rmse, "R²:", rfr_r2)
```

```
Random Forest → RMSE: 9.757713738069647 R²: -0.4642705335969681
```