In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [6]:
```
data = pd.read_csv('C:/Users/Adamin/OneDrive/Desktop/uber.csv')
```

In [7]:
```
df = data.copy()
```

In [8]:
```
df.head()
```

Out[8]:

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

In [9]:
```
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
```

In [10]:
```
df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"])
```

In [11]:
```
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
```

In [12]:
```
df.describe()
```

Out[12]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longituc
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.00000
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.52529
std	1.601382e+07	9.901776	11.437787	7.720539	13.11740
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.66630
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.99140
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.98009
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.96365
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.57260

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
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
```

In [14]:
```
df.select_dtypes(include=[np.number]).corr()
```

Out[14]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_l
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	

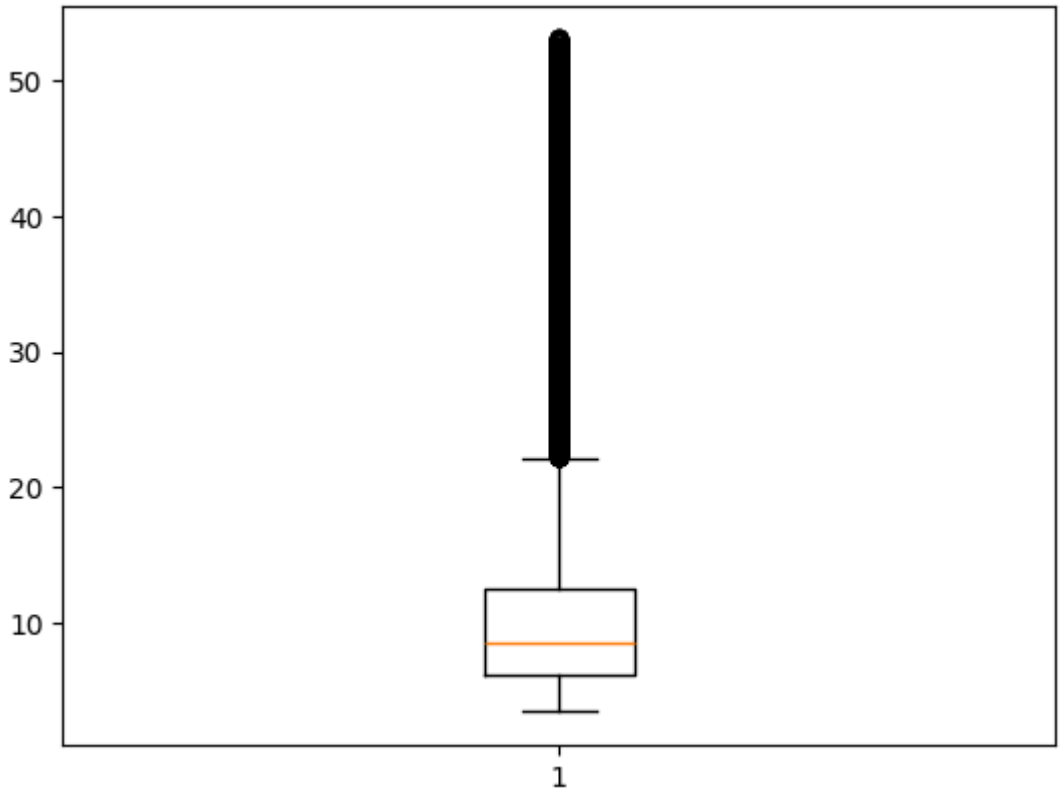
In [15]:
```
print(df.columns)
```

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

In [16]:
```
df.dropna(inplace=True)
```

In [22]:
```
plt.boxplot(df['fare_amount'])
```

Out[22]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x22172151810>,
  <matplotlib.lines.Line2D at 0x22172151950>],
 'caps': [<matplotlib.lines.Line2D at 0x22172151a90>,
  <matplotlib.lines.Line2D at 0x22172151bd0>],
 'boxes': [<matplotlib.lines.Line2D at 0x221721516d0>],
 'medians': [<matplotlib.lines.Line2D at 0x22172151d10>],
 'fliers': [<matplotlib.lines.Line2D at 0x22172151e50>],
 'means': []}
```



In [24]:
```
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)]
```

In [25]:
```
df.isnull().sum()
```

Out[25]:
```
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
```

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
x = df.drop("fare_amount", axis = 1)
#And y as target variable
y = df['fare_amount']
```

In [28]:
```
x['pickup_datetime'] = pd.to_numeric(pd.to_datetime(x['pickup_datetime']))
x = x.loc[:, x.columns.str.contains('^Unnamed')]
```

In [29]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, randc
```

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [32]:
```
lrmodel = LinearRegression()
```

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

Out[32]:

```
▾ LinearRegression
LinearRegression()
```

In [33]:
```
predict = lrmodel.predict(x_test)
```

In [34]:
```
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: 7.083585521002763 R²: -0.00015874177771135756
```

In [35]:
```
from sklearn.ensemble import RandomForestRegressor
rfrmodel = RandomForestRegressor(n_estimators = 100, random_state = 101)
```

In [ ]:
```
rfrmodel.fit(x_train, y_train)
rfrmodel_pred = rfrmodel.predict(x_test)
```

In [ ]:
```
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)
```

In [ ]:
```

```