In [1]:
```
# PS1: Predict the price of the Uber ride from a given pickup point to the agreed drop-o
# 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
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

In [3]:
```
data= pd.read_csv("uber.csv")
```

In [4]:
```
data.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 [5]:
```
data
```

Out[5]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	drop
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
...	...	...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	

200000 rows × 9 columns

In [6]:
```
data.head()
```

Out[6]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_lo
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-7
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-7
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-7
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-7
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-7

In [7]:
```
data.dtypes
```

Out[7]:
```
Unnamed: 0            int64
key                  object
fare_amount         float64
pickup_datetime      object
pickup_longitude    float64
pickup_latitude     float64
dropoff_longitude   float64
dropoff_latitude    float64
passenger_count       int64
dtype: object
```

In [8]:
```
data.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 [9]:
```
data.isnull().sum()
```

Out[9]:
```
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
```

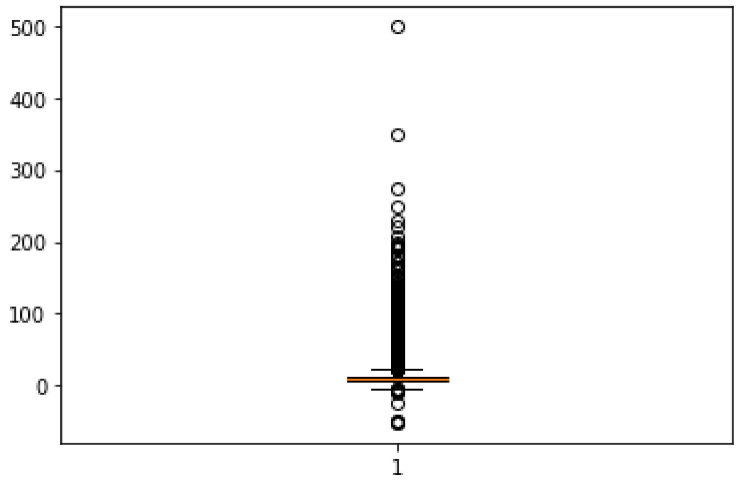
In [10]:
```
data.dropna(inplace = True)
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
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 [12]:
```
# Check for outliers using boxplot
plt.boxplot(data['fare_amount'])
```

Out[12]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x277a616e5b0>,
  <matplotlib.lines.Line2D at 0x277a616e880>],
 'caps': [<matplotlib.lines.Line2D at 0x277a616ec10>,
  <matplotlib.lines.Line2D at 0x277a616ee20>],
 'boxes': [<matplotlib.lines.Line2D at 0x277a616e2e0>],
 'medians': [<matplotlib.lines.Line2D at 0x277a63ec130>],
 'fliers': [<matplotlib.lines.Line2D at 0x277a63ec400>],
 'means': []}
```

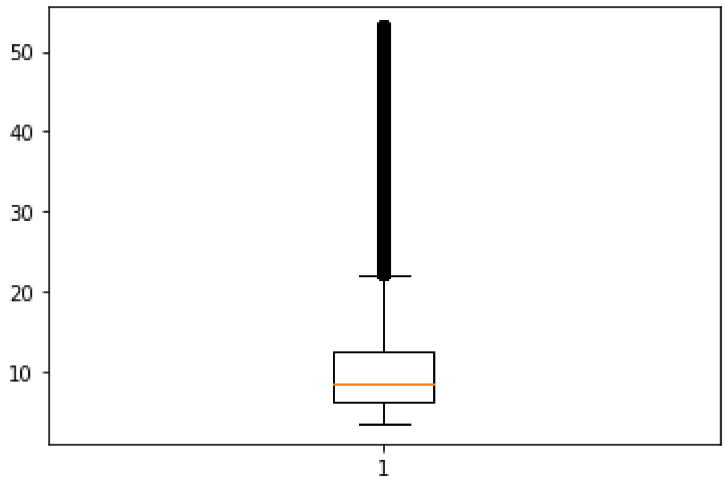


In [13]:
```
# Remove outliers by keeping values within the 1st and 99th percentiles
q_low = data["fare_amount"].quantile(0.01)
q_hi  = data["fare_amount"].quantile(0.99)

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

In [14]:
```
# Check for outliers using boxplot
plt.boxplot(data['fare_amount'])
```

Out[14]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x277a68824c0>,
  <matplotlib.lines.Line2D at 0x277a6882790>],
 'caps': [<matplotlib.lines.Line2D at 0x277a6882a90>,
  <matplotlib.lines.Line2D at 0x277a6882d60>],
 'boxes': [<matplotlib.lines.Line2D at 0x277a68821f0>],
 'medians': [<matplotlib.lines.Line2D at 0x277a688c070>],
 'fliers': [<matplotlib.lines.Line2D at 0x277a688c340>],
 'means': []}
```



In [15]:
```
#Correlation
data.corr()
```

Out[15]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitu
Unnamed: 0	1.000000	-0.000339	-0.000209	-0.000028	-0.000107	0.000
fare_amount	-0.000339	1.000000	0.006534	-0.004843	0.006012	-0.007
pickup_longitude	-0.000209	0.006534	1.000000	-0.810943	0.832846	-0.846
pickup_latitude	-0.000028	-0.004843	-0.810943	1.000000	-0.773097	0.697
dropoff_longitude	-0.000107	0.006012	0.832846	-0.773097	1.000000	-0.914
dropoff_latitude	0.000647	-0.007651	-0.846712	0.697275	-0.914509	1.000
passenger_count	0.002241	0.012145	-0.000737	-0.001288	-0.000020	-0.000

In [16]:
```
X = data.drop(columns=["fare_amount","pickup_datetime","key"])
y = data["fare_amount"]
```

In [17]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [18]:
```
linear_reg_model = LinearRegression()
linear_reg_model.fit(X_train, y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
random_forest_model = RandomForestRegressor()
random_forest_model.fit(X_train, y_train)
```

Out[19]:
```
RandomForestRegressor()
```

In [20]:
```
# Step 5: Evaluate the models and compare their respective scores
# Evaluate linear regression model
y_pred_linear = linear_reg_model.predict(X_test)
r2_linear = r2_score(y_test, y_pred_linear)
rmse_linear = np.sqrt(mean_squared_error(y_test, y_pred_linear))
```

In [21]:
```

print("Linear Regression - R2 Score:", r2_linear)
print("Linear Regression - RMSE:", rmse_linear)
```

```
Linear Regression - R2 Score: 0.00013541786208948192
Linear Regression - RMSE: 8.11069071043841
```

In [22]:
```
# Evaluate random forest model
y_pred_rf = random_forest_model.predict(X_test)
r2_rf = r2_score(y_test, y_pred_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
```

In [23]:
```

print("Random Forest Regression - R2 Score:", r2_rf)
print("Random Forest Regression - RMSE:", rmse_rf)
```

```
Random Forest Regression - R2 Score: 0.7912228329195581
Random Forest Regression - RMSE: 3.7061991172170936
```