```
import pandas as pd
```

```
import numpy as np
```

```
df=pd.read_csv("uber.csv")
```

```
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
```

```
df.describe()
```

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	199999.000000	200000.000000
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	39.923890	1.684535
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	6.794829	1.385997
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	40.733823	1.000000
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	40.753042	1.000000
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.963658	40.768001	2.000000

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

```
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
```

```
df['dropoff_latitude'].fillna(value=df['dropoff_latitude'].mean(),inplace = True)
```

```
df['dropoff_longitude'].fillna(value=df['dropoff_longitude'].median(),inplace = True)
```

```
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
```

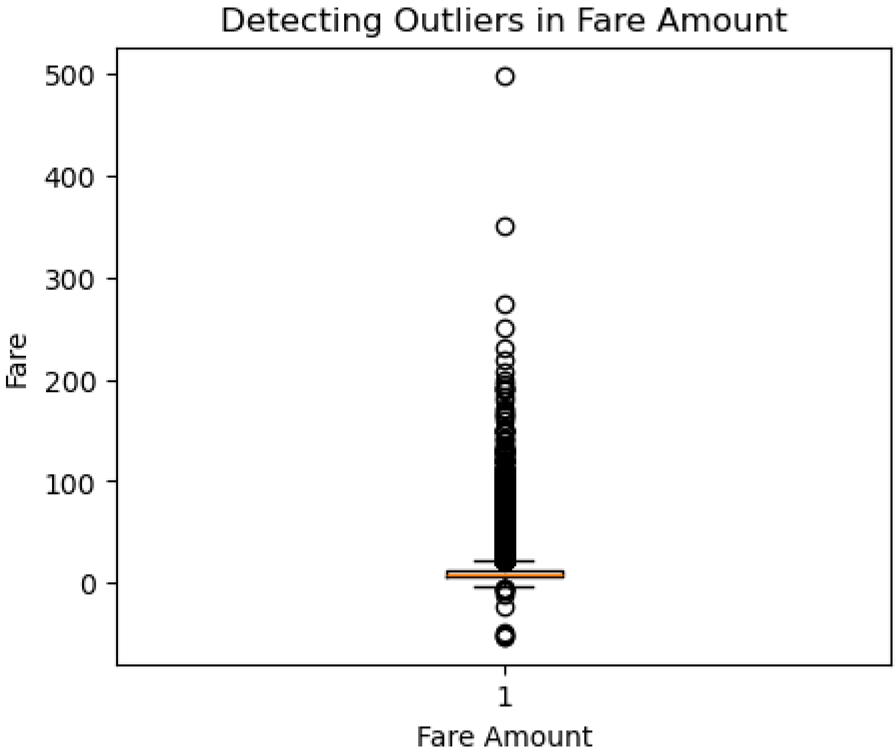
```
df
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1

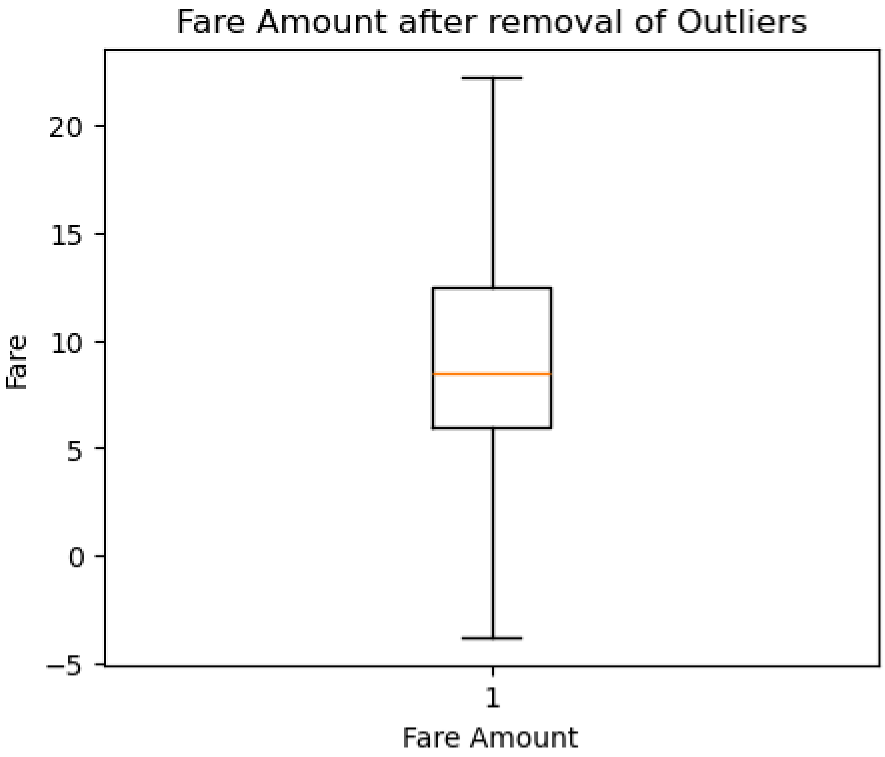
	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5
...	...	...	...	...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.986525	40.740297	1
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-74.006672	40.739620	1
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.858957	40.692588	2
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	-73.983215	40.695415	1
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	-73.985508	40.768793	1

```
200000 rows × 9 columns
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(5,4))

plt.boxplot(df['fare_amount'])

plt.xlabel('Fare Amount')

plt.ylabel('Fare')

plt.title('Detecting Outliers in Fare Amount')

plt.show()
```



```
def remove_outliers(df, columns):


    # looping through each column

    for col in columns:

        feature = df[col]

        q1 = feature.quantile(0.25)
        q3 = feature.quantile(0.75)
        IQR = q3 - q1
        lower_bound = float(q1 - 1.5 * IQR)  # Cast to float
        upper_bound = float(q3 + 1.5 * IQR)  # Cast to float

        # replacing outliers with bounds
        df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
        df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])

    return df
df = remove_outliers(df, ['fare_amount'])

# plotting the boxplot for cleaned data
plt.figure(figsize=(5, 4))
plt.boxplot(df['fare_amount'])
plt.xlabel('Fare Amount')
plt.ylabel('Fare')
plt.title('Fare Amount after removal of Outliers')
plt.show()
```



```
df=df.drop(['Unnamed: 0','key'],axis=1)
```

```
df.head()
```

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5

```
df['pickup_datetime']=pd.to_datetime(df['pickup_datetime'],utc=True)
```

```
df
```

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	7.50	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1
1	7.70	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1
2	12.90	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1
3	5.30	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3
4	16.00	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5
...	...	...	...	...	...	...	...
199995	3.00	2012-10-28 10:49:00+00:00	-73.987042	40.739367	-73.986525	40.740297	1
199996	7.50	2014-03-14 01:09:00+00:00	-73.984722	40.736837	-74.006672	40.739620	1
199997	22.25	2009-06-29 00:42:00+00:00	-73.986017	40.756487	-73.858957	40.692588	2
199998	14.50	2015-05-20 14:56:25+00:00	-73.997124	40.725452	-73.983215	40.695415	1
199999	14.10	2010-05-15 04:08:00+00:00	-73.984395	40.720077	-73.985508	40.768793	1

200000 rows × 7 columns

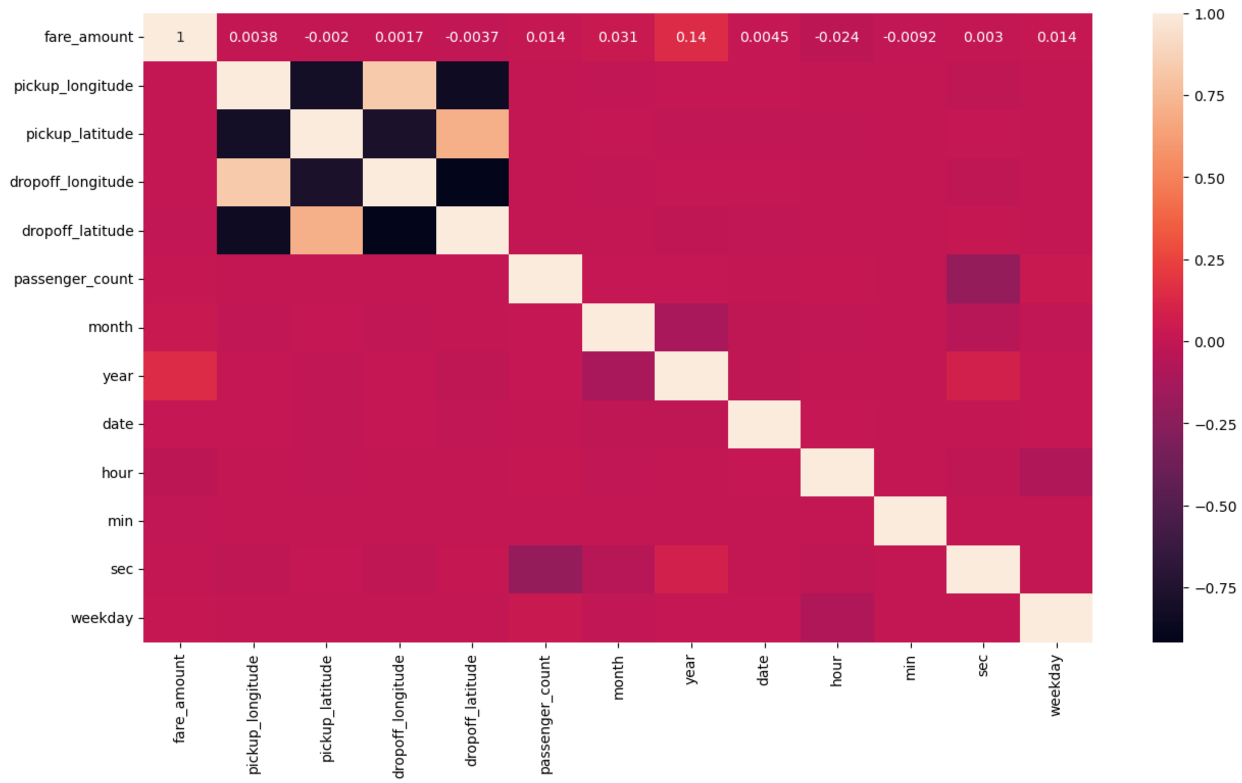
```
df['month']=df['pickup_datetime'].dt.month
```

```
df['year']=df['pickup_datetime'].dt.year
```

```
df['date']=df['pickup_datetime'].dt.day

df['hour']=df['pickup_datetime'].dt.hour

df['min']=df['pickup_datetime'].dt.minute

df['sec']=df['pickup_datetime'].dt.second

df['weekday']=df['pickup_datetime'].dt.weekday

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

df.head()
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	month	year	date	hour	min	sec	weekday
0	7.5	-73.999817	40.738354	-73.999512	40.723217	1	5	2015	7	19	52	6	3
1	7.7	-73.994355	40.728225	-73.994710	40.750325	1	7	2009	17	20	4	56	4
2	12.9	-74.005043	40.740770	-73.962565	40.772647	1	8	2009	24	21	45	0	0
3	5.3	-73.976124	40.790844	-73.965316	40.803349	3	6	2009	26	8	22	21	4
4	16.0	-73.925023	40.744085	-73.973082	40.761247	5	8	2014	28	17	47	0	3

```
import seaborn as sns

corr=df.corr()

plt.figure(figsize=(15,8))

co=sns.heatmap(corr,annot=True)
```



```
import math


def haversine_distance(lat1, lon1, lat2, lon2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # Convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(math.radians, [lon1, lat1, lon2, lat2])

    # Haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon/2)**2
    c = 2 * math.asin(math.sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371 * c
    return km
import pandas as pd
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

# Assuming df is your dataframe containing the Uber ride data

# Calculate distance using Haversine formula
df['distance_km'] = df.apply(lambda row: haversine_distance(row['pickup_latitude'], 
row['pickup_longitude'],
                                                            row['dropoff_latitude'], row['dropoff_longitude']), axis=1)

# Split data into training and testing sets
X = df[['distance_km']]
y = df['fare_amount']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize models
linear_reg = LinearRegression()
rf_reg = RandomForestRegressor()

# Fit models
linear_reg.fit(X_train, y_train)
rf_reg.fit(X_train, y_train)
```

```
# Predict on test set
y_pred_lr = linear_reg.predict(X_test)
y_pred_rf = rf_reg.predict(X_test)

# Evaluate models
r2_lr = r2_score(y_test, y_pred_lr)
rmse_lr = mean_squared_error(y_test, y_pred_lr, squared=False)

r2_rf = r2_score(y_test, y_pred_rf)
rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)

print(f"Linear Regression R2: {r2_lr}, RMSE: {rmse_lr}")
print(f"Random Forest Regression R2: {r2_rf}, RMSE: {rmse_rf}")
```

```
Linear Regression R2: 0.0001380597101960923, RMSE: 5.454139357692611
Random Forest Regression R2: 0.6633839737169384, RMSE: 3.1646348482147735
```