In [1]:
```
import pandas as pd
```

In [2]:
```
uber = pd.read_csv('uber.csv')
uber
```

Out[2]:

	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
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

In [3]:
```
uber.columns
```

Out[3]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [4]:
```
uber.drop(['Unnamed: 0','key'], axis=1, inplace=True)
```

In [5]:
```
uber
```

Out[5]:

	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
...	...	...	...	...	...	...	...
199995	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.986525	40.740297	1
199996	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-74.006672	40.739620	1
199997	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.858957	40.692588	2
199998	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	-73.983215	40.695415	1
199999	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	-73.985508	40.768793	1

200000 rows × 7 columns

In [6]:
```
uber.describe()
```

Out[6]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	200000.000000	200000.000000	200000.000000	199999.000000	199999.000000	200000.000000
mean	11.359955	-72.527638	39.935885	-72.525292	39.923890	1.684535
std	9.901776	11.437787	7.720539	13.117408	6.794829	1.385997
min	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	6.000000	-73.992065	40.734796	-73.991407	40.733823	1.000000
50%	8.500000	-73.981823	40.752592	-73.980093	40.753042	1.000000
75%	12.500000	-73.967154	40.767158	-73.963658	40.768001	2.000000
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

In [7]:
```
uber.isna().sum()
```

Out[7]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [8]:
```
uber.dropna(inplace=True)
```

In [9]:
```
uber.isna().sum()
```

Out[9]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

In [10]:
```
uber = uber[uber['fare_amount']>0]
uber.describe()
```

Out[10]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	199977.000000	199977.000000	199977.000000	199977.000000	199977.000000	199977.000000
mean	11.362586	-72.527844	39.935995	-72.526243	39.924410	1.684489
std	9.897063	11.437285	7.720462	13.115114	6.793438	1.385972
min	0.010000	-1340.648410	-74.015515	-3356.666300	-881.985513	0.000000
25%	6.000000	-73.992065	40.734795	-73.991407	40.733825	1.000000
50%	8.500000	-73.981823	40.752592	-73.980093	40.753042	1.000000
75%	12.500000	-73.967155	40.767158	-73.963659	40.768001	2.000000
max	499.000000	57.418457	1644.421482	1153.572603	872.697628	208.000000

In [11]:
```
import numpy as np
import math
```

In [12]:
```
def haversine(long1, long2, lat1, lat2):
    long1 = long1 * (math.pi/180)
    long2 = long2 * (math.pi/180)
    lat1 = lat1 * (math.pi/180)
    lat2 = lat2 * (math.pi/180)

    long = long2 - long1
    lat = lat2 - lat1
    r = 6371

    dist = 2 * r * np.arcsin(np.sqrt(((1 - np.cos(lat)) + np.cos(lat1) * np.cos(lat2) * (1 - np.cos(long)))/2))
    return dist
```

In [13]:
```
uber['distance'] = haversine(uber['pickup_longitude'], uber['dropoff_longitude'], uber['pickup_latitude'], uber['dropoff_latit
```

```
C:\Users\Himanshu B. Kale\AppData\Local\Temp\ipykernel_21520\2457494978.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  uber['distance'] = haversine(uber['pickup_longitude'], uber['dropoff_longitude'], uber['pickup_latitude'], uber['dropoff_la
titude'])
```

In [14]:
```
uber
```

Out[14]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	distance
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1	1.683323
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1	2.457590
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1	5.036377
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3	1.661683
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5	4.475450
...	...	...	...	...	...	...	...	...
199995	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.986525	40.740297	1	0.112210
199996	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-74.006672	40.739620	1	1.875050
199997	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.858957	40.692588	2	12.850319
199998	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	-73.983215	40.695415	1	3.539715
199999	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	-73.985508	40.768793	1	5.417783

199977 rows × 8 columns

In [15]:
```
uber.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 199977 entries, 0 to 199999
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   fare_amount        199977 non-null  float64
 1   pickup_datetime    199977 non-null  object
 2   pickup_longitude   199977 non-null  float64
 3   pickup_latitude    199977 non-null  float64
 4   dropoff_longitude  199977 non-null  float64
 5   dropoff_latitude   199977 non-null  float64
 6   passenger_count    199977 non-null  int64
 7   distance           199977 non-null  float64
dtypes: float64(6), int64(1), object(1)
memory usage: 13.7+ MB
```

In [16]:
```
uber['pickup_datetime'] = pd.to_datetime(uber['pickup_datetime'])
uber
```

```
C:\Users\Himanshu B. Kale\AppData\Local\Temp\ipykernel_21520\1906792355.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  uber['pickup_datetime'] = pd.to_datetime(uber['pickup_datetime'])
```

Out[16]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	distance
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1	1.683323
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1	2.457590
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1	5.036377
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3	1.661683
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5	4.475450
...	...	...	...	...	...	...	...	...
199995	3.0	2012-10-28 10:49:00+00:00	-73.987042	40.739367	-73.986525	40.740297	1	0.112210
199996	7.5	2014-03-14 01:09:00+00:00	-73.984722	40.736837	-74.006672	40.739620	1	1.875050
199997	30.9	2009-06-29 00:42:00+00:00	-73.986017	40.756487	-73.858957	40.692588	2	12.850319
199998	14.5	2015-05-20 14:56:25+00:00	-73.997124	40.725452	-73.983215	40.695415	1	3.539715
199999	14.1	2010-05-15 04:08:00+00:00	-73.984395	40.720077	-73.985508	40.768793	1	5.417783

199977 rows × 8 columns

In [17]:
```
uber['hour'] = uber['pickup_datetime'].apply(lambda x:x.hour)
uber
```

```
C:\Users\Himanshu B. Kale\AppData\Local\Temp\ipykernel_21520\4189286700.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  uber['hour'] = uber['pickup_datetime'].apply(lambda x:x.hour)
```

Out[17]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	distance	hour
0	7.5	2015-05-07 19:52:06+00:00	-73.999817	40.738354	-73.999512	40.723217	1	1.683323	19
1	7.7	2009-07-17 20:04:56+00:00	-73.994355	40.728225	-73.994710	40.750325	1	2.457590	20
2	12.9	2009-08-24 21:45:00+00:00	-74.005043	40.740770	-73.962565	40.772647	1	5.036377	21
3	5.3	2009-06-26 08:22:21+00:00	-73.976124	40.790844	-73.965316	40.803349	3	1.661683	8
4	16.0	2014-08-28 17:47:00+00:00	-73.925023	40.744085	-73.973082	40.761247	5	4.475450	17
...	...	...	...	...	...	...	...	...	...
199995	3.0	2012-10-28 10:49:00+00:00	-73.987042	40.739367	-73.986525	40.740297	1	0.112210	10
199996	7.5	2014-03-14 01:09:00+00:00	-73.984722	40.736837	-74.006672	40.739620	1	1.875050	1
199997	30.9	2009-06-29 00:42:00+00:00	-73.986017	40.756487	-73.858957	40.692588	2	12.850319	0
199998	14.5	2015-05-20 14:56:25+00:00	-73.997124	40.725452	-73.983215	40.695415	1	3.539715	14
199999	14.1	2010-05-15 04:08:00+00:00	-73.984395	40.720077	-73.985508	40.768793	1	5.417783	4

199977 rows × 9 columns

In [18]:
```
uber.drop(['pickup_datetime','pickup_longitude','pickup_latitude','dropoff_longitude', 'dropoff_latitude'], axis=1, inplace=Tr
```

```
C:\Users\Himanshu B. Kale\AppData\Local\Temp\ipykernel_21520\2575693911.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  uber.drop(['pickup_datetime','pickup_longitude','pickup_latitude','dropoff_longitude', 'dropoff_latitude'], axis=1, inplace
=True)
```

In [19]:
```
uber
```

Out[19]:

	fare_amount	passenger_count	distance	hour
0	7.5	1	1.683323	19
1	7.7	1	2.457590	20
2	12.9	1	5.036377	21
3	5.3	3	1.661683	8
4	16.0	5	4.475450	17
...	...	...	...	...
199995	3.0	1	0.112210	10
199996	7.5	1	1.875050	1
199997	30.9	2	12.850319	0
199998	14.5	1	3.539715	14
199999	14.1	1	5.417783	4

199977 rows × 4 columns

In [20]:
```
uber.describe()
```

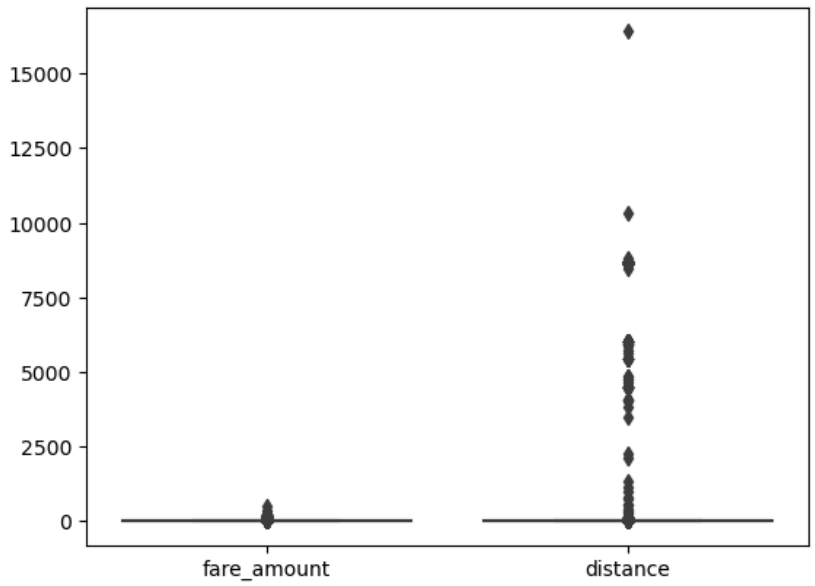
Out[20]:

	fare_amount	passenger_count	distance	hour
count	199977.000000	199977.000000	199977.000000	199977.000000
mean	11.362586	1.684489	20.770717	13.491452
std	9.897063	1.385972	382.008526	6.515383
min	0.010000	0.000000	0.000000	0.000000
25%	6.000000	1.000000	1.215372	9.000000
50%	8.500000	1.000000	2.121109	14.000000
75%	12.500000	2.000000	3.875211	19.000000
max	499.000000	208.000000	16409.239135	23.000000

In [21]:
```
import seaborn as sns
```

In [22]:
```
sns.boxplot(uber[['fare_amount', 'distance']])
```

Out[22]: <Axes: >



In [24]:
```
def outliers(data_item):
    outliers=[]
    data_item=sorted(data_item)
    q1 = np.percentile(data_item,25)
    q3 = np.percentile(data_item,75)
    iqr = q3-q1
    lower_bound = q1-(1.5*iqr)
    upper_bound = q3+(1.5*iqr)
    print(lower_bound,upper_bound)
    return lower_bound,upper_bound
```

In [25]:
```
lower,upper = outliers(uber['distance'])
```

```
-2.7743876581438416 7.864970610538522
```

In [26]:
```
uber = uber[uber['distance']>lower]
uber = uber[uber['distance']<upper]
```

In [27]:
```
uber
```

Out[27]:

	fare_amount	passenger_count	distance	hour
0	7.5	1	1.683323	19
1	7.7	1	2.457590	20
2	12.9	1	5.036377	21
3	5.3	3	1.661683	8
4	16.0	5	4.475450	17
...	...	...	...	...
199994	12.0	1	1.122878	14
199995	3.0	1	0.112210	10
199996	7.5	1	1.875050	1
199998	14.5	1	3.539715	14
199999	14.1	1	5.417783	4

183225 rows × 4 columns

In [28]:
```
lowerf,upperf = outliers(uber['fare_amount'])
```

```
-2.7 19.700000000000003
```

In [29]:
```
uber = uber[uber['fare_amount']>lowerf]
uber = uber[uber['fare_amount']<upperf]
```

In [30]:
```
uber
```

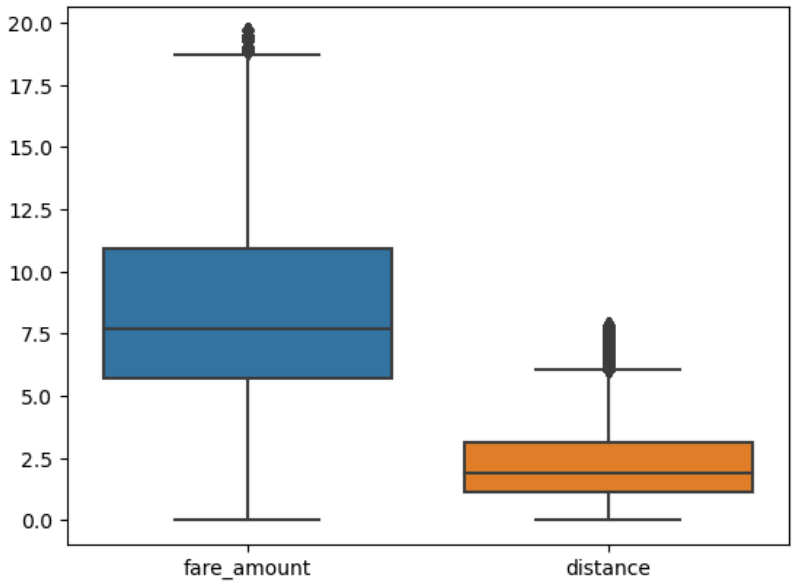
Out[30]:

	fare_amount	passenger_count	distance	hour
0	7.5	1	1.683323	19
1	7.7	1	2.457590	20
2	12.9	1	5.036377	21
3	5.3	3	1.661683	8
4	16.0	5	4.475450	17
...	...	...	...	...
199994	12.0	1	1.122878	14
199995	3.0	1	0.112210	10
199996	7.5	1	1.875050	1
199998	14.5	1	3.539715	14
199999	14.1	1	5.417783	4

176770 rows × 4 columns

In [31]:
```
sns.boxplot(uber[['fare_amount', 'distance']])
```

Out[31]:
```
<Axes: >
```



In [32]:
```
X = uber.drop(['fare_amount'],axis=1)
y = uber['fare_amount']
```

In [33]:
```
from sklearn.model_selection import train_test_split
```

In [34]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

In [35]:
```
from sklearn.linear_model import LinearRegression
```

In [36]:
```
model_LR = LinearRegression()
model_LR.fit(X_train,y_train)
```

Out[36]:
```
LinearRegression
LinearRegression()
```

In [39]:
```
y_pred_LR = model_LR.predict(X_test)
```

In [40]:
```
from sklearn.metrics import r2_score,mean_squared_error
```

In [43]:
```
LR_r2 = r2_score(y_test,y_pred_LR)
LR_mse = mean_squared_error(y_test,y_pred_LR)
print(LR_r2)
print(LR_mse)
```

```
0.6087650825727711
5.4138004057037445
```

In [44]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [45]:
```
model_RF = RandomForestRegressor(n_estimators=100, random_state=101)
model_RF.fit(X_train,y_train)
```

Out[45]:
```
RandomForestRegressor
RandomForestRegressor(random_state=101)
```

In [46]:
```
y_pred_RF = model_RF.predict(X_test)
```

In [47]:
```
RF_r2 = r2_score(y_test,y_pred_RF)
RF_mse = mean_squared_error(y_test,y_pred_RF)
print(RF_r2)
print(RF_mse)
```

```
0.5904760622472572
5.666878802461641
```

In [ ]:
```

```