In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries insta
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
# You can also write temporary files to /kaggle/temp/, but they won't be save
```

In [11]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

In [16]:

```
df=pd.read_csv('../input/uber-fares-dataset/uber.csv')
df.head()
```

Out[16]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latit
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744

In [17]:
```
df.dtypes
```

Out[17]:
```
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 [19]:
```
df.isnull().sum()
```

Out[19]:
```
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 [21]:
```
df.drop(['Unnamed: 0','key'],axis=1,inplace=True)
df.dropna(axis=0,inplace=True)
df.head()
```

Out[21]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40

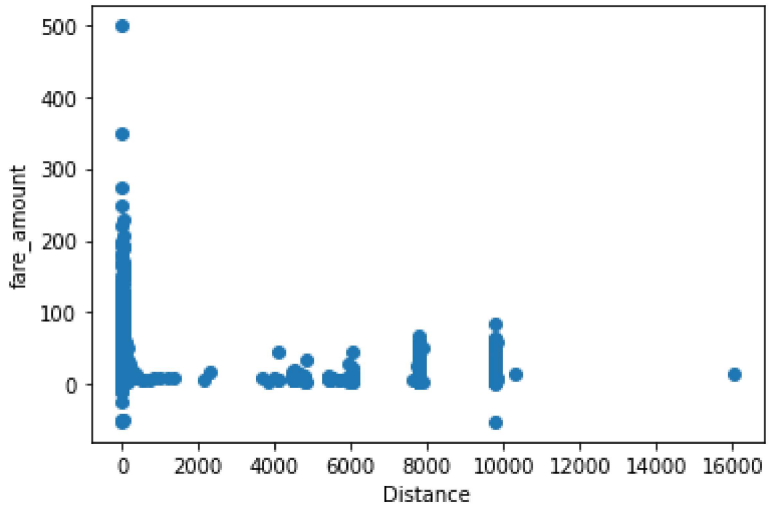
In [24]:
```
def haversine(lon_1,lon_2,lat_1,lat_2):
    lon_1,lon_2,lat_1,lat_2=map(np.radians,[lon_1,lon_2,lat_1,lat_2])
    diff_lon=lon_2-lon_1
    diff_lat=lat_2-lat_1
    distance=2* 6371* np.arcsin(np.sqrt(np.sin(diff_lat/2.0)**2+np.cos(lat_2)
    return distance
df['Distance']=haversine(df['pickup_longitude'],df['dropoff_longitude'],df['p
df['Distance']=df['Distance'].astype(float).round(2)
df.head()
```

Out[24]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40

In [26]:
```
plt.scatter(df['Distance'],df['fare_amount'])
plt.xlabel('Distance')
plt.ylabel("fare_amount")
```

Out[26]: Text(0, 0.5, 'fare_amount')



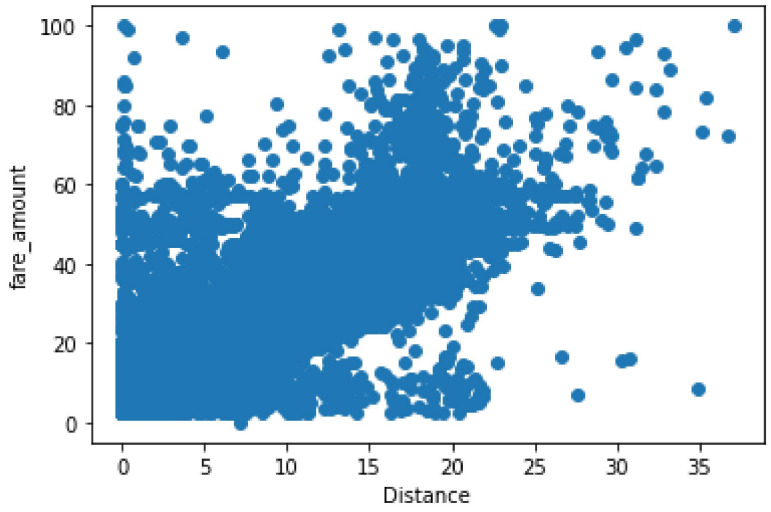
In [29]:

```
df.drop(df[df['Distance']>60].index, inplace=True)
df.drop(df[df['Distance']==0].index, inplace=True)
df.drop(df[df['Distance']<0].index, inplace=True)
df.drop(df[df['fare_amount']==0].index, inplace=True)
df.drop(df[df['fare_amount']<0].index, inplace=True)
df.drop(df[df['Distance']>100].index, inplace=True)
df.drop(df[df['fare_amount']>100].index, inplace=True)
df.drop(df[(df['fare_amount']>100)& (df['Distance']<1)].index, inplace=True)
df.drop(df[(df['fare_amount']<100)& (df['Distance']>100)].index, inplace=True
#Dealing with Outliers via removing rows with non-plausible fare amounts and


#Plotting a Scatter Plot to check for any more outliers and also to show corr

plt.scatter(df['Distance'], df['fare_amount'])
plt.xlabel("Distance")
plt.ylabel("fare_amount")
```

Out[29]: Text(0, 0.5, 'fare_amount')



In [35]:
```
x = df['Distance'].values.reshape(-1, 1)

#Independent Variable

y = df['fare_amount'].values.reshape(-1, 1) #Dependent Variante
std= StandardScaler()

Y = std.fit_transform(y)
X= std.fit_transform(x)

#splitting the date into training and testing set

X_train, X_test, Y_train, Y_test=train_test_split(x, y, test_size=0.2, random
```

In [43]:
```
def apply_model(model):
    model.fit(X_train,Y_train)
    print('Training score=', model.score(X_train, Y_train))
    print ('Testing score=', model.score(X_test, Y_test))
    print('Accuracy =', model.score(X_test, Y_test))
    Y_pred = model.predict(X_test)
    print("Predicted values:\n", Y_pred)
    print ("Mean Absolute Error=", metrics.mean_absolute_error(Y_test, Y_pred
    print("Mean Squared Errar=", metrics.mean_squared_error(Y_test, Y_pred))
    print("Root Mean Squared Error=", np.sqrt(metrics.mean_squared_error (Y_t
```

In [44]:
```
lr=LinearRegression()
apply_model(lr)
```

```
Training score= 0.8024105826058359
Testing score= 0.8001502430280464
Accuracy = 0.8001502430280464
Predicted values:
 [[10.48573998]
 [24.33918948]
 [12.28155751]
 ...
 [ 9.62281467]
 [ 7.31390642]
 [ 7.82699715]]
Mean Absolute Error= 2.2670737352240997
Mean Squared Errar= 17.09789090798564
Root Mean Squared Error= 4.134959601735625
```

In [46]:

```
rf= RandomForestRegressor(n_estimators=100,random_state=10)
apply_model(rf)
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:2: DataConvers
ionWarning: A column-vector y was passed when a 1d array was expected. Plea
se change the shape of y to (n_samples,), for example using ravel().


Training score= 0.8245770565753652
Testing score= 0.7949764832424238
Accuracy = 0.7949764832424238
Predicted values:
 [10.30179632 28.66229949 12.07421932 ...  9.2839272   7.37217115
  7.77629638]
Mean Absolute Error= 2.295347655571429
Mean Squared Errar= 17.54052532365377
Root Mean Squared Error= 4.188141034355668
```