```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt



# ### Loop the data lines
# with open("taxitrip.csv", 'r') as temp_f:
#     # get No of columns in each line
#     col_count = [ len(l.split(",")) for l in temp_f.readlines() ]

# ### Generate column names  (names will be 0, 1, 2, ..., maximum columns - 1)
# column_names = [i for i in range(0, max(col_count))]

# ### Read csv
# df = pd.read_csv("taxitrip.csv", header=None, delimiter=",", names=column_names)


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# % matplotlib inline
plt.style.use('seaborn-whitegrid')
```

```
<ipython-input-3-df9f1b8b3082>:6: MatplotlibDeprecationWarning: The seaborn styles shi
  plt.style.use('seaborn-whitegrid')
```

```
df_train =  pd.read_csv('train.csv', nrows = 2_000_000)
# list first few rows (datapoints)
df_train.head()
```

```
<ipython-input-4-19bb455f4bd8>:1: DtypeWarning: Columns
  df_train =  pd.read_csv('train.csv', nrows = 2_000_00
```

	VendorID	lpep_pickup_datetime	lpep_dropoff_dateti
0	1.0	2021-07-01 00:30:52	2021-07-01 00:35:
1	2.0	2021-07-01 00:25:36	2021-07-01 01:01:
2	2.0	2021-07-01 00:05:58	2021-07-01 00:12:
3	2.0	2021-07-01 00:41:40	2021-07-01 00:47:
4	2.0	2021-07-01 00:51:32	2021-07-01 00:58:

```
df_train.dtypes
```

```
VendorID                 float64
lpep_pickup_datetime      object
lpep_dropoff_datetime     object
store_and_fwd_flag        object
RatecodeID               float64
PULocationID               int64
```

```
DOLocationID               int64
passenger_count          float64
trip_distance            float64
fare_amount              float64
extra                    float64
mta_tax                  float64
tip_amount               float64
tolls_amount             float64
ehail_fee                float64
improvement_surcharge    float64
total_amount             float64
payment_type             float64
trip_type                float64
congestion_surcharge     float64
dtype: object
```

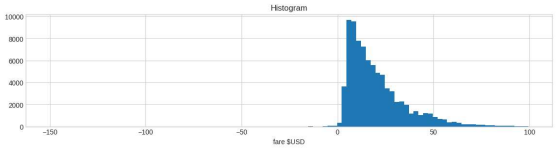
```
df_train.describe()
```

	VendorID	RatecodeID	PULocationID	DOLoca
count	51173.000000	51173.000000	83691.000000	83691.
mean	1.851113	1.159244	108.362572	133.
std	0.355981	0.773260	70.370170	77.
min	1.000000	1.000000	3.000000	1.
25%	2.000000	1.000000	56.000000	69.
50%	2.000000	1.000000	75.000000	132.
75%	2.000000	1.000000	166.000000	205.
max	2.000000	5.000000	265.000000	265.

```
df_train.describe()
```

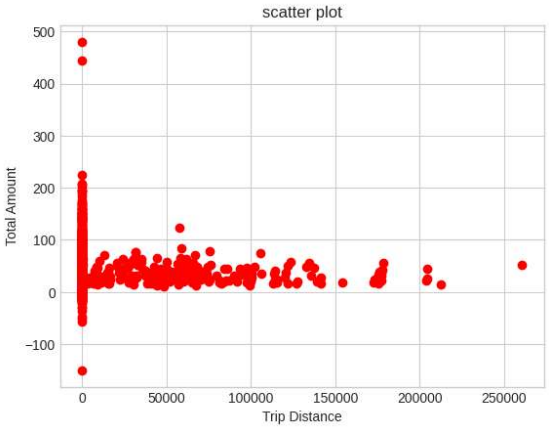
	VendorID	RatecodeID	PULocationID	DOLoca
count	51173.000000	51173.000000	83691.000000	83691.
mean	1.851113	1.159244	108.362572	133.
std	0.355981	0.773260	70.370170	77.
min	1.000000	1.000000	3.000000	1.
25%	2.000000	1.000000	56.000000	69.
50%	2.000000	1.000000	75.000000	132.
75%	2.000000	1.000000	166.000000	205.
max	2.000000	5.000000	265.000000	265.

```
# plot histogram of fare
df_train[df_train.fare_amount<100].fare_amount.hist(bins=100, figsize=(14,3))
plt.xlabel('fare $USD')
plt.title('Histogram');
```

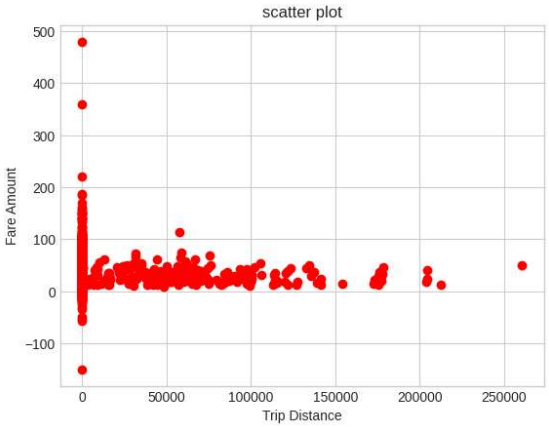


Visualization

```
plt.scatter(df_train['trip_distance'],df_train['total_amount'], color='red')
plt.title("scatter plot")
plt.xlabel("Trip Distance")
plt.ylabel("Total Amount")
plt.show()
```

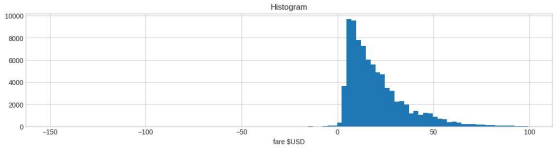


```
plt.scatter(df_train['trip_distance'],df_train['fare_amount'], color='red')
plt.title("scatter plot")
plt.xlabel("Trip Distance")
plt.ylabel("Fare Amount")
plt.show()
```

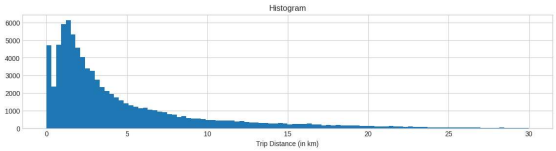



```
plt.scatter(df_train['trip_distance'],df_train['tip_amount'], color='red')
plt.title("scatter plot")
plt.xlabel("Trip Distance")
plt.ylabel("Tip Amount")
plt.show()
```

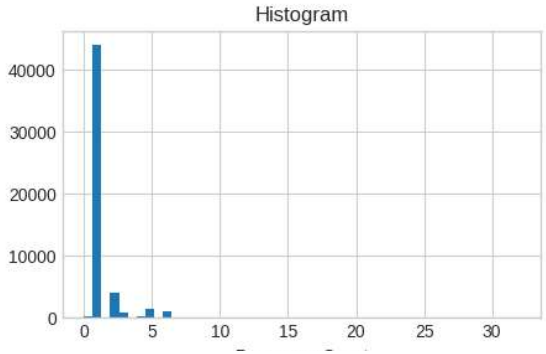

```
df_train[df_train.fare_amount<100].fare_amount.hist(bins=100, figsize=(14,3))
plt.xlabel('fare $USD')
plt.title('Histogram');
```




```
df_train[df_train.trip_distance<30].trip_distance.hist(bins=100, figsize=(14,3))
plt.xlabel('Trip Distance (in km)')
plt.title('Histogram');
```

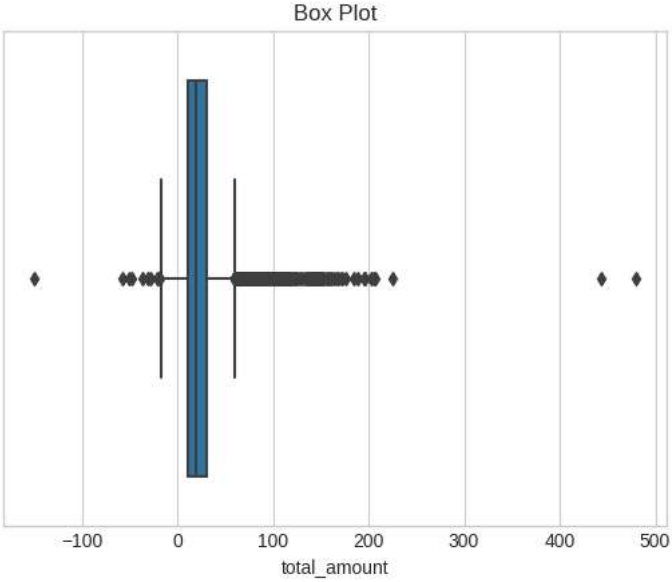



```
df_train[df_train.passenger_count<100].passenger_count.hist(bins=50, figsize=(5,3))
plt.xlabel('Passenger Count')
plt.title('Histogram');
```



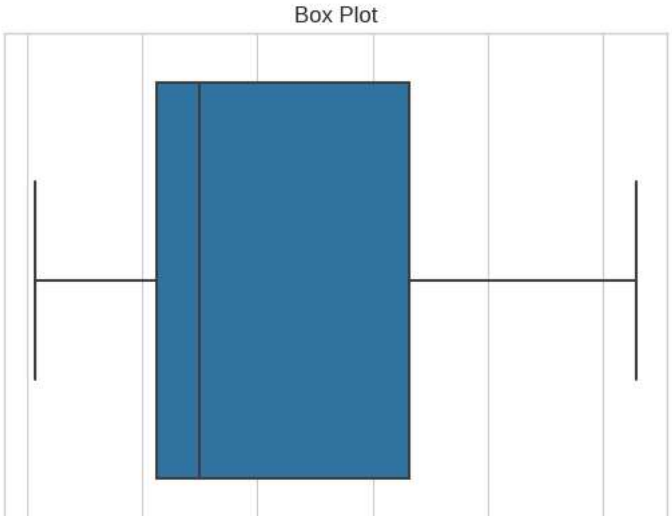

```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(x="total_amount", data=df_train)
plt.title("Box Plot")
```

```
Text(0.5, 1.0, 'Box Plot')
```



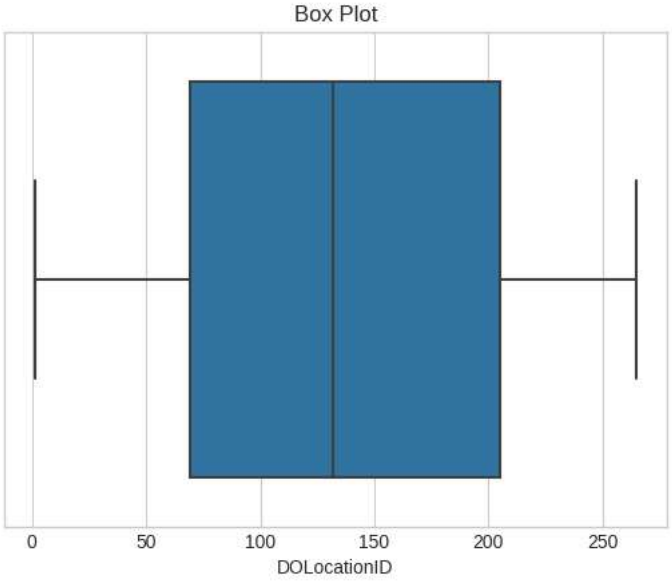

```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(x="PULocationID", data=df_train)
plt.title("Box Plot")
```

```
Text(0.5, 1.0, 'Box Plot')
```




```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
sb.boxplot(x="DOLocationID", data=df_train)
plt.title("Box Plot")
```

```
Text(0.5, 1.0, 'Box Plot')
```




```
print(df_train.isnull().sum())
```

```
VendorID                 32518
lpep_pickup_datetime         0
lpep_dropoff_datetime        0
store_and_fwd_flag       32518
RatecodeID               32518
PULocationID                 0
DOLocationID                 0
passenger_count          32518
trip_distance                0
fare_amount                  0
extra                        0
mta_tax                      0
tip_amount                   0
tolls_amount                 0
ehail_fee                83691
improvement_surcharge        0
total_amount                 0
payment_type             32518
trip_type                32518
congestion_surcharge     32518
dtype: int64
```

```
df_train.groupby('trip_distance').mean()
```

```
<ipython-input-19-17bcad3aff26>:1: FutureWarning: The default value of numeric_only in
  df_train.groupby('trip_distance').mean()
```

trip_distance	VendorID	RatecodeID	PULocationID	DOLocationID	passenger_count	fare
0.00	1.4597	1.755350	134.659913	145.273517	1.091298	18
0.01	2.0000	2.165138	118.525000	126.206250	1.229358	12
0.02	2.0000	2.153846	109.277228	110.554455	1.323077	15
0.03	2.0000	2.017857	101.753623	106.260870	1.410714	13
0.04	2.0000	1.867925	106.261538	110.323077	1.320755	9
...	...	...	...	...	...	
204048.30	NaN	NaN	218.000000	218.000000	NaN	18
204333.20	NaN	NaN	210.000000	76.000000	NaN	24
204624.55	NaN	NaN	262.000000	81.000000	NaN	41
212426.90	NaN	NaN	75.000000	263.000000	NaN	11
260517.93	NaN	NaN	141.000000	220.000000	NaN	50

3153 rows × 16 columns

```
df_train=df_train.fillna(df_train.groupby('trip_distance').transform('mean'))
```

```
<ipython-input-20-d09119dc1f88>:1: FutureWarning: The default value of numeric_only ir
  df_train=df_train.fillna(df_train.groupby('trip_distance').transform('mean'))
```

```
df_train
```

	VendorID	lpep_pickup_datetime	lpep_dropoff_datetime	store_and_fwd_flag	Rate
0	1.0	2021-07-01 00:30:52	2021-07-01 00:35:36	N	
1	2.0	2021-07-01 00:25:36	2021-07-01 01:01:31	N	
2	2.0	2021-07-01 00:05:58	2021-07-01 00:12:00	N	
3	2.0	2021-07-01 00:41:40	2021-07-01 00:47:23	N	
4	2.0	2021-07-01 00:51:32	2021-07-01 00:58:46	N	
...	...	...	...	...	
83686	2.0	2021-07-02 07:59:00	2021-07-02 08:33:00	NaN	
83687	2.0	2021-07-02 07:02:00	2021-07-02 07:18:00	NaN	
83688	2.0	2021-07-02 07:53:00	2021-07-02 08:15:00	NaN	
83689	2.0	2021-07-02 07:58:00	2021-07-02 08:30:00	NaN	
83690	2.0	2021-07-02 07:00:00	2021-07-02 07:26:00	NaN	

83691 rows × 20 columns

```
df_train=df_train.drop('store_and_fwd_flag',axis=1)
```

```
print('Old size: %d' % len(df_train))
df_train = df_train[df_train.fare_amount>=0]
print('New size: %d' % len(df_train))
```

```
Old size: 83691
New size: 83546
```

```
df_train
```

	VendorID	lpep_pickup_datetime	lpep_dropoff_datetime	RatecodeID	PULocationID
0	1.0	2021-07-01 00:30:52	2021-07-01 00:35:36	1.0	74
1	2.0	2021-07-01 00:25:36	2021-07-01 01:01:31	1.0	116
2	2.0	2021-07-01 00:05:58	2021-07-01 00:12:00	1.0	97
3	2.0	2021-07-01 00:41:40	2021-07-01 00:47:23	1.0	74
4	2.0	2021-07-01 00:51:22	2021-07-01 00:58:46	1.0	42

```
df_train=df_train.drop('ehail_fee',axis=1)
df_train=df_train.drop('lpep_pickup_datetime',axis=1)
df_train=df_train.drop('lpep_dropoff_datetime',axis=1)
```

```
df_train
```

	VendorID	RatecodeID	PULocationID	DOLocationID	passenger_count	trip_distance
0	1.0	1.0	74	168	1.000000	1.20
1	2.0	1.0	116	265	2.000000	13.69
2	2.0	1.0	97	33	1.000000	0.95
3	2.0	1.0	74	42	1.000000	1.24
4	2.0	1.0	42	244	1.000000	1.10
...	...	...	...	...	...	...
83686	2.0	1.7	218	169	1.300000	18.04
83687	2.0	1.0	74	137	1.125000	5.50
83688	2.0	1.0	69	75	1.166667	5.13
83689	2.0	1.0	117	82	1.500000	12.58
83690	2.0	1.0	218	196	1.200000	11.32

83546 rows × 16 columns

```
print(df_train.isnull().sum())
```

```
VendorID                1259
RatecodeID              1259
PULocationID               0
DOLocationID               0
passenger_count         1259
trip_distance              0
fare_amount                0
extra                      0
mta_tax                    0
tip_amount                 0
tolls_amount               0
improvement_surcharge      0
total_amount               0
payment_type            1259
trip_type               1259
```

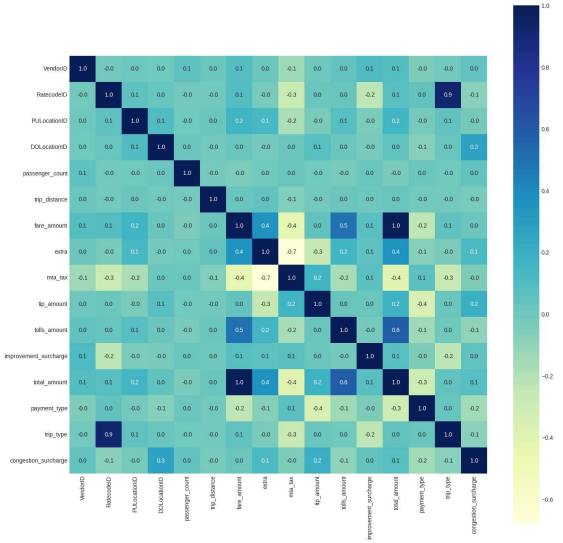
```
congestion_surcharge     1259
dtype: int64
```

```
df_train = df_train.fillna(df_train.median())
```

```
print(df_train.isnull().sum())
```

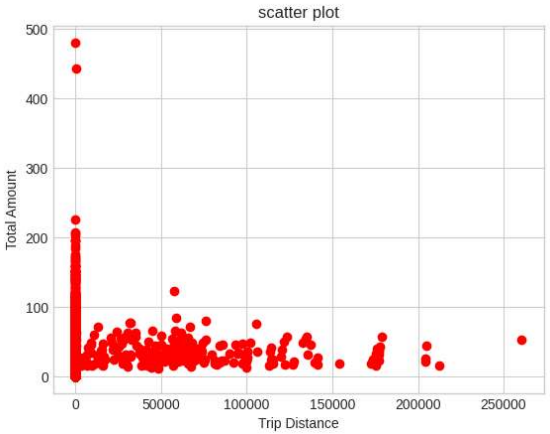
```
VendorID                 0
RatecodeID               0
PULocationID             0
DOLocationID             0
passenger_count          0
trip_distance            0
fare_amount              0
extra                    0
mta_tax                  0
tip_amount               0
tolls_amount             0
improvement_surcharge    0
total_amount             0
payment_type             0
trip_type                0
congestion_surcharge     0
dtype: int64
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the correlation matrix
corr_matrix = df_train.corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(16, 16))
sns.heatmap(corr_matrix, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size': 1

# Show the plot
plt.show()
```



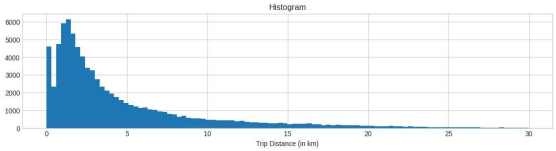

```
# plot histogram of fare
df_train[df_train.fare_amount<100].fare_amount.hist(bins=100, figsize=(14,3))
plt.xlabel('fare $USD')
plt.title('Histogram');
```


```
plt.scatter(df_train['trip_distance'],df_train['total_amount'], color='red')
plt.title("scatter plot")
plt.xlabel("Trip Distance")
plt.ylabel("Total Amount")
plt.show()
```




```
plt.scatter(df_train['trip_distance'],df_train['fare_amount'], color='red')
plt.title("scatter plot")
plt.xlabel("Trip Distance")
plt.ylabel("Fare Amount")
plt.show()
```


```
df_train[df_train.trip_distance<30].trip_distance.hist(bins=100, figsize=(14,3))
plt.xlabel('Trip Distance (in km)')
plt.title('Histogram');
```




```
x=df_train.drop(['total_amount'],axis = 1)
y=df_train['total_amount']
y
```

```
0         7.30
1        43.30
2        10.14
3         7.80
4         8.30
         ...
83686    59.84
83687    25.87
83688    22.75
83689    54.12
83690    48.89
Name: total_amount, Length: 83546, dtype: float64
```

```
# example of a normalization
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
scaler= MinMaxScaler()

x= scaler.fit_transform(x)
x
```

```
array([[0.        , 0.        , 0.27099237, ..., 0.25      , 0.        ,
        0.        ],
```

```
       [1.        , 0.        , 0.43129771, ..., 0.25      , 0.        ,
        0.        ],
       [1.        , 0.        , 0.35877863, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [1.        , 0.        , 0.2519084 , ..., 0.0625    , 0.        ,
        0.5       ],
       [1.        , 0.        , 0.4351145 , ..., 0.        , 0.        ,
        0.25      ],
       [1.        , 0.        , 0.82061069, ..., 0.1       , 0.        ,
        0.4       ]])
```

```
# # example of a normalization
# from numpy import asarray
# from sklearn.preprocessing import MinMaxScaler
# scaler= MinMaxScaler()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.metrics import mean_squared_error
from scipy.stats import pearsonr
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn import metrics
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
#No need to drop any columns since the Pearson Correlations are upwards 0.2 (medium relatio
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2) #80% for Training and 20%
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
```

```
(66836, 15) (16710, 15) (66836,) (16710,)
```

```
from sklearn import ensemble
df_predict = ensemble.GradientBoostingRegressor(n_estimators = 100, max_depth = 5, min_samp
          learning_rate = 0.1, loss = 'squared_error')
```

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

```
▾ GradientBoostingRegressor
GradientBoostingRegressor(max_depth=5)
```

```
df_predict_test=df_predict.predict(x_test)
df_predict_train=df_predict.predict(x_train)
```

```
pd.DataFrame({'actual unseen data':y_train,'predicted unseen data':df_predict_train})
```

	actual unseen data	predicted unseen data
6162	46.55	46.070641
22408	71.22	71.092521
41104	12.36	12.253029
72277	25.50	25.193434
44807	7.56	7.516188
...	...	...
28902	19.81	19.978548
69062	20.74	20.900150
20303	24.06	24.075911
29627	17.46	17.696754
79409	23.70	23.789672

66836 rows × 2 columns

```
scores = cross_val_score(df_predict, x_test, y_test, cv=5)
scores
```

```
array([0.99891223, 0.99826301, 0.99714878, 0.99286136, 0.99755431])
```

```
predictions = cross_val_predict(df_predict, x_test, y_test, cv=5)
accuracy = metrics.r2_score(y_test, predictions)
accuracy
```

```
0.9967806990060651
```

```
x_ax = range(len(y_test))
plt.figure(figsize=(20,6))
plt.scatter(x_ax, y_test, s=5, color="blue", label="original")
plt.plot(x_ax, df_predict_test, lw=0.8, color="red", label="predicted")
plt.legend()
plt.show()
```


```
print('MAE= ',metrics.mean_absolute_error(y_test,df_predict_test))
print('MSE= ',metrics.mean_squared_error(y_test,df_predict_test))
print('R2 value= ',df_predict.score(x_test,y_test))
print('Adjusted R2 value= ',1 - (1 - (df_predict.score(x_test,y_test))) * ((756 - 1)/(756-1
print('RMSE (train)= ',np.sqrt(mean_squared_error(y_train,df_predict_train)))
print('RMSE (test)= ',np.sqrt(mean_squared_error(y_test,df_predict_test)))
```

```
MAE=  0.26363339918508977
MSE=  0.24053658005918263
R2 value=  0.9991662548830335
Adjusted R2 value=  0.9991550636734098
RMSE (train)=  0.3906672429498933
RMSE (test)=  0.49044528752877486
```