Level A

Pick a dataset and objective

Dataset:

- Dataset of Nifty Stock prices of Indian companies. (https://www.kaggle.com/rohanrao/nifty50-stock-market-data)

Problem Statement:

- Creating a Predictive Model using any Algorithm (Deep Learning/Machine Learning) that can predict the stock price(Close column) of ASIANPAINTs.
- A prediction for the year of 2016 using data from 2009-2015.
- Show accuracy of the algorithms and explaining a choice of accuracy metric (RMSE/MAE/MAPE,R^2, Adjusted R^2).

This algorithm predits the Close column of the data set for 2016 year using 2009 - 20015 years data.

In [1]:
```
### Loading the libraries required

import pandas as pd       #Loading pandas for creating and adjusting dataframes
import numpy as np        # Loading Numpy for creating and adjusting arrays

import seaborn as sns               # Loading seaborn for visualizations
import matplotlib.pyplot as plt     # Loading matplotlib for visualizations
%matplotlib inline

from sklearn.preprocessing import StandardScaler    # Loading standard scaler to normalize the data

from sklearn.model_selection import GridSearchCV   # Loading gridsearch CV for hyperparameter tuning using Crss Validation
from sklearn.neighbors import KNeighborsRegressor  #Loading KNN regressor to cretae a model
from sklearn.ensemble import RandomForestRegressor #Loading Random Forest regressor to cretae a model
from sklearn.ensemble import AdaBoostRegressor     #Loading Ada Boost regressor to cretae a model
```

```
from tensorflow.keras.models import Sequential     #Loading Sequential model from tensor flow to craete a ANN model
from tensorflow.keras.layers import Dense          #Loading Dense layer from tensor flow to craete a Neural network layers
from tensorflow.keras.layers import Dropout        #Loading Drop out Layer

from tensorflow.keras.optimizers import Adam       #Loading Adam optimizer for NN

from sklearn.metrics import mean_squared_error     #Loading Mean Squared Error to to check the error metrics of each model
from sklearn.metrics import mean_absolute_percentage_error  #Loading Mean absolute percentage error to to check the error metrics
from sklearn.metrics import r2_score       #Loading R^2 (R squared) to check the efficiency metrics of each model
```

In [2]:

```
df = pd.read_csv("ASIANPAINT.csv")   #Loading the data set using read_csv to dataframe
```

In [3]:

```
# Adjusting the "Date" column to date time format and creating a Year column to split the data
df["Year"] = pd.to_datetime(df.Date, format="%d-%m-%Y").dt.year
```

In [16]:

```
df_2016 = df[(df["Year"] <= 2016) & (df["Year"]>=2009)]  #Spliiting the dat to use only fronm 2009 to 2016
```

In [18]:

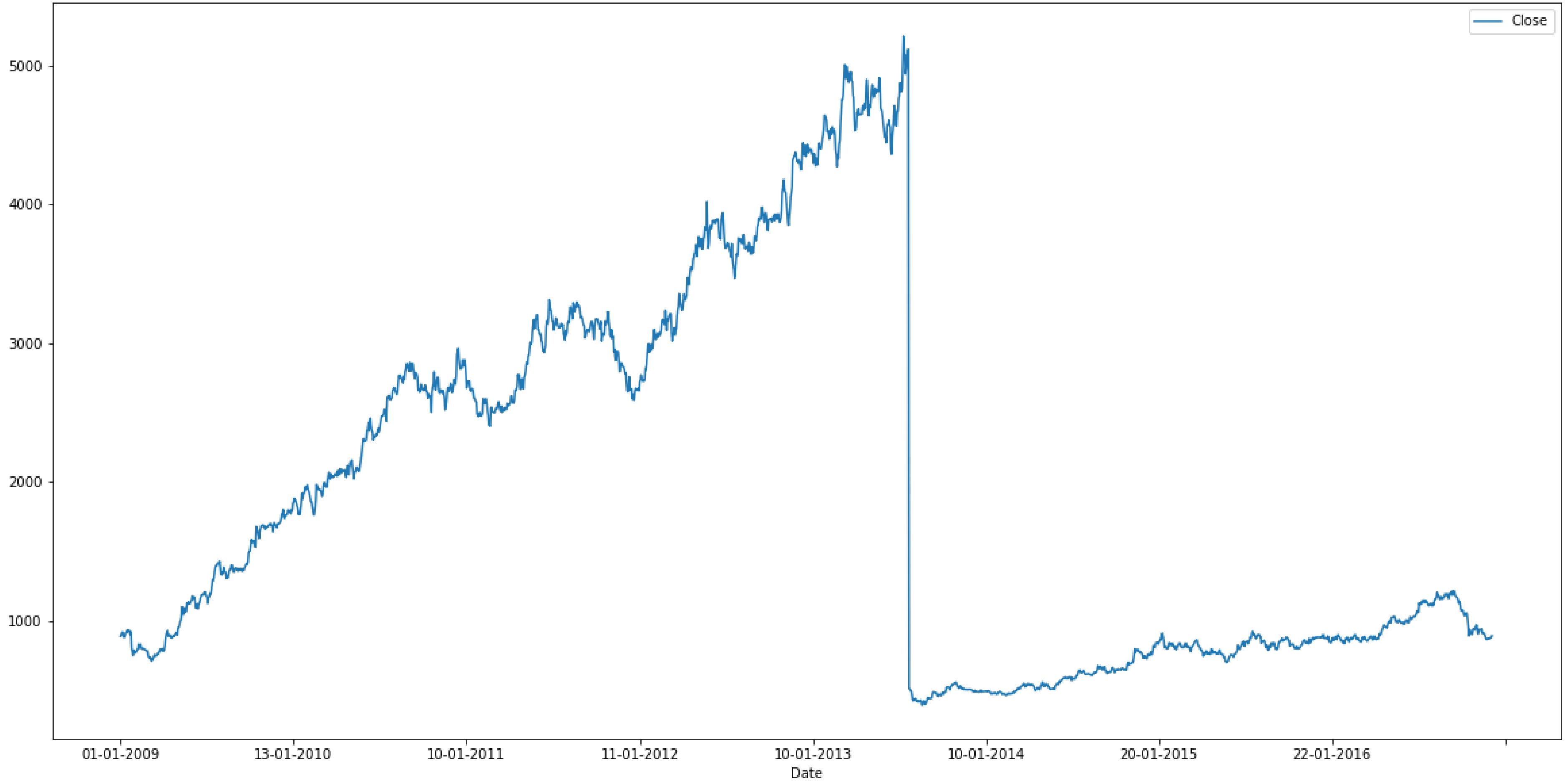
```
#Checking the NON null Count of and Data type of the columns
df_2016.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1982 entries, 2253 to 4234
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Date                1982 non-null   object 
 1   Symbol              1982 non-null   object 
 2   Series              1982 non-null   object 
 3   Prev Close          1982 non-null   float64
 4   Open                1982 non-null   float64
 5   High                1982 non-null   float64
 6   Low                 1982 non-null   float64
 7   Last                1982 non-null   float64
 8   Close               1982 non-null   float64
 9   VWAP                1982 non-null   float64
 10  Volume              1982 non-null   int64  
 11  Turnover            1982 non-null   float64
 12  Trades              1385 non-null   float64
```

```
 13  Deliverable Volume  1982 non-null   float64
 14  %Deliverble         1982 non-null   float64
 15  Year                1982 non-null   int64
dtypes: float64(11), int64(2), object(3)
memory usage: 263.2+ KB
```

In [23]:

```
# Plotting the "Close" column of the data set
fig = df_2016.plot(x = "Date",y = "Close",figsize=(20,10))
fig.figure.savefig("Close_plot.jpg",bbox_inches='tight')
plt.show()
```



In [24]:

```
#Dropping the unnecessary columns of the data
```

```
df_2016 = df_2016.drop(["Date","Symbol","Series"],axis=1)
```

In [26]:

```
#Splitting the dataset into train and test
# Train Data for 2009 -2015 years
# Test Data for 2016
train = df_2016.loc[df_2016["Year"] < 2016]
test = df_2016.loc[df_2016["Year"] == 2016]
```

In [27]:

```
# Checking null values count in all the coulmns of the train dataset
train.isnull().sum()
```

Out[27]:

```
Prev Close              0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
VWAP                    0
Volume                  0
Turnover                0
Trades                597
Deliverable Volume      0
%Deliverble             0
Year                    0
dtype: int64
```

In [28]:

```
# Checking the skewness of the "Trades"
train["Trades"].skew(axis = 0, skipna = True)
```

Out[28]:

```
1.8170049925548912
```

In [29]:

```
# Imputing the Columns with mean if not skewed
# Imputing the column with median if skewed
train["Trades"].fillna(train["Trades"].median(),inplace = True)
train.isnull().sum()
```

```
C:\Users\cricl\anaconda3\lib\site-packages\pandas\core\generic.py:6392: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

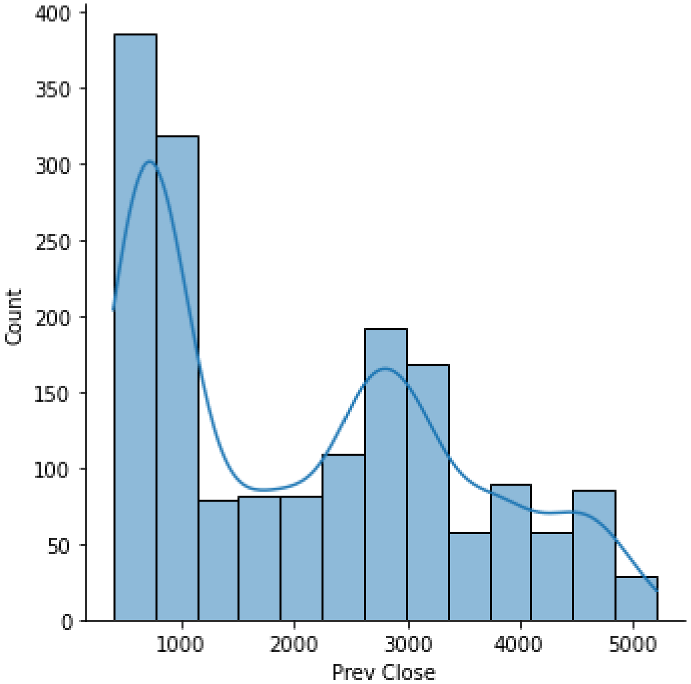
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy
  return self._update_inplace(result)
```

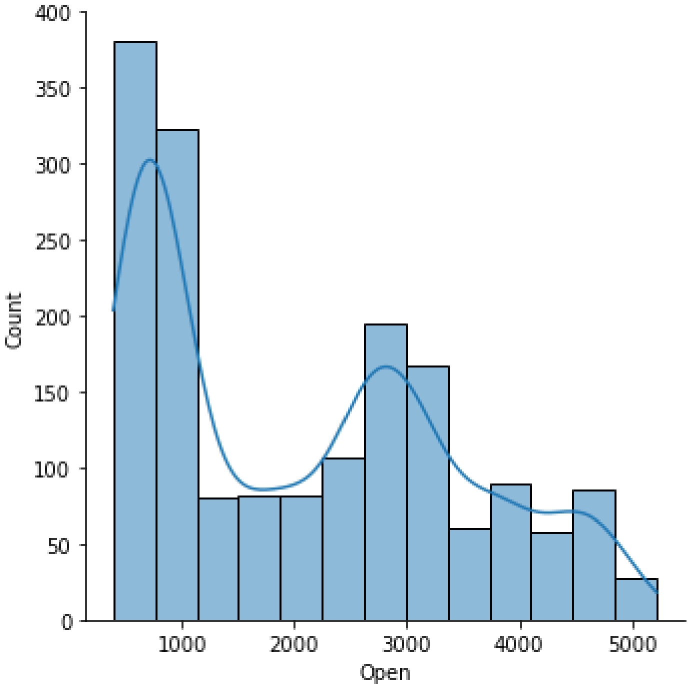
Out[29]:

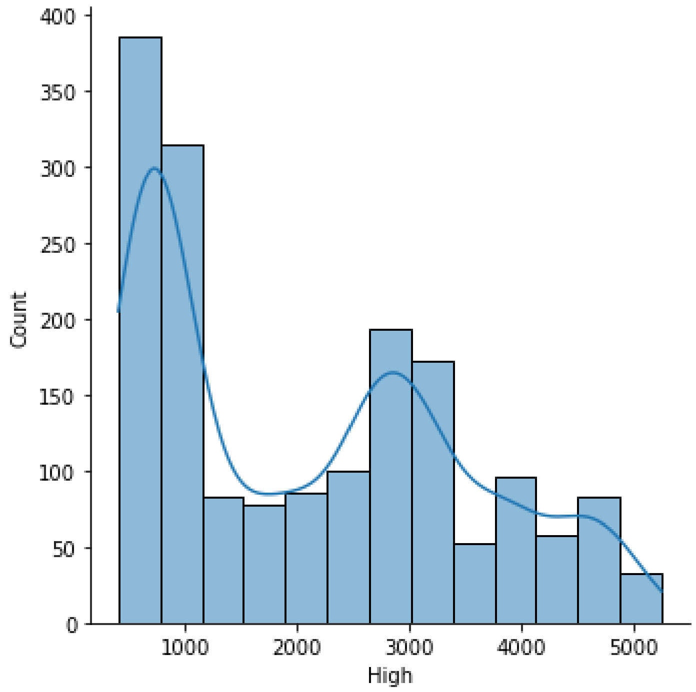
```
Prev Close           0
Open                 0
High                 0
Low                  0
Last                 0
Close                0
VWAP                 0
Volume               0
Turnover             0
Trades               0
Deliverable Volume   0
%Deliverble          0
Year                 0
dtype: int64
```

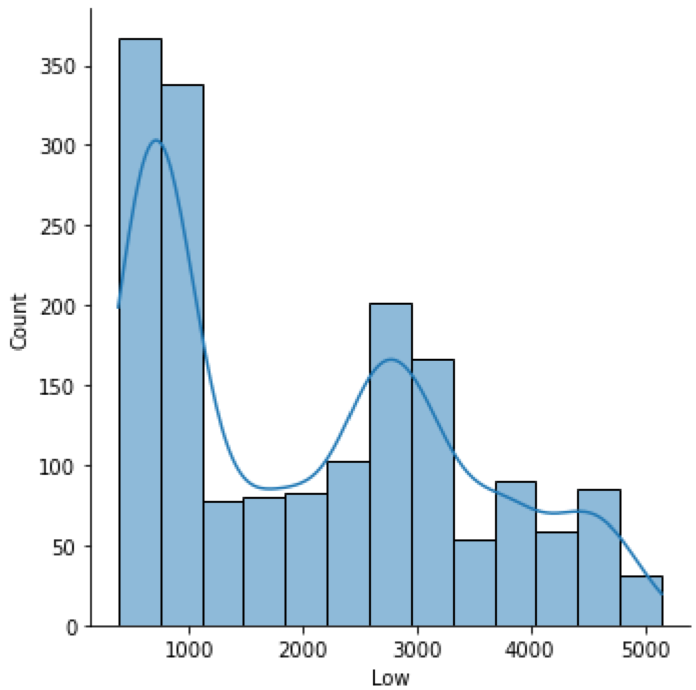
In [30]:

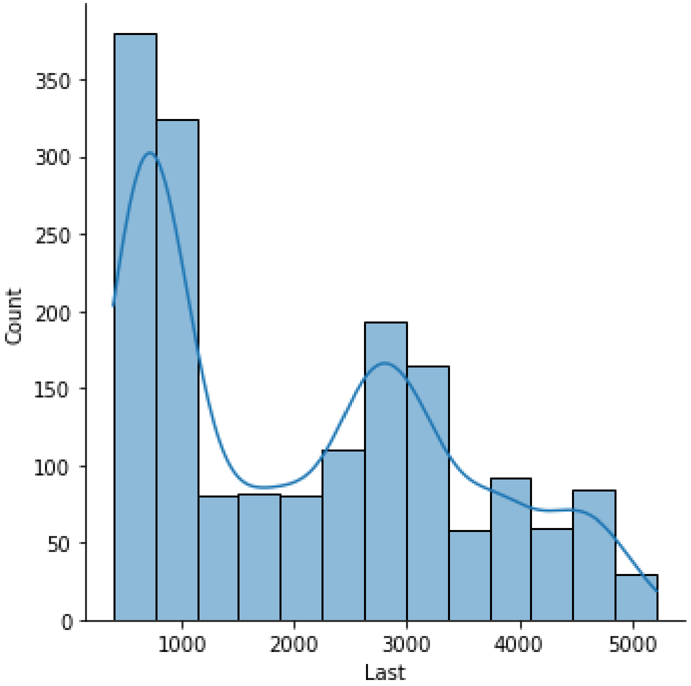
```
#Visualizxing the distributaion of the data for all the independent Columns
for i in range(len(train.columns)):
    x = train.columns[i]
    fig = sns.displot(data=train, x=x, kde=True)
    filename = "{} Histogram.jpg".format(train.columns[i])
    fig.figure.savefig(filename,bbox_inches='tight')
```


Count
Prev Close
0
50
100
150
200
250
300
350
400
1000
2000
3000
4000
5000

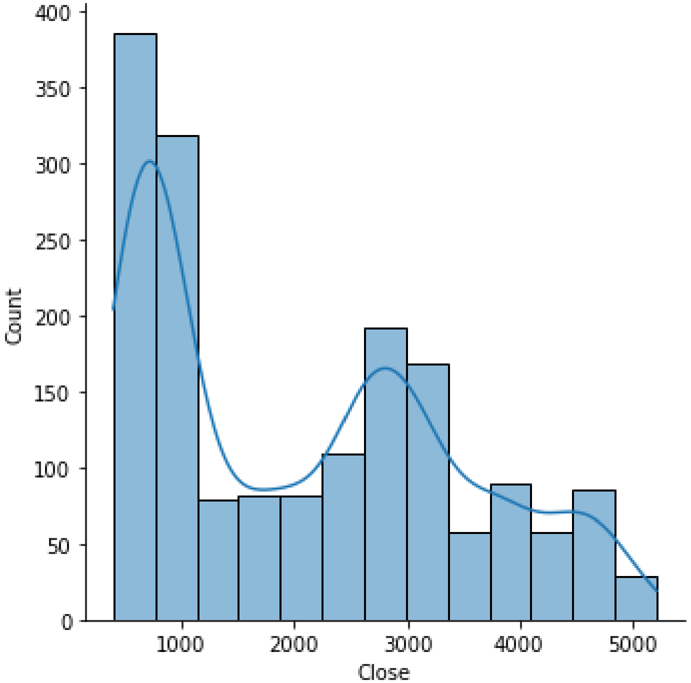

Count
Open
0
50
100
150
200
250
300
350
400
1000
2000
3000
4000
5000


Count
High
0
50
100
150
200
250
300
350
400
1000
2000
3000
4000
5000

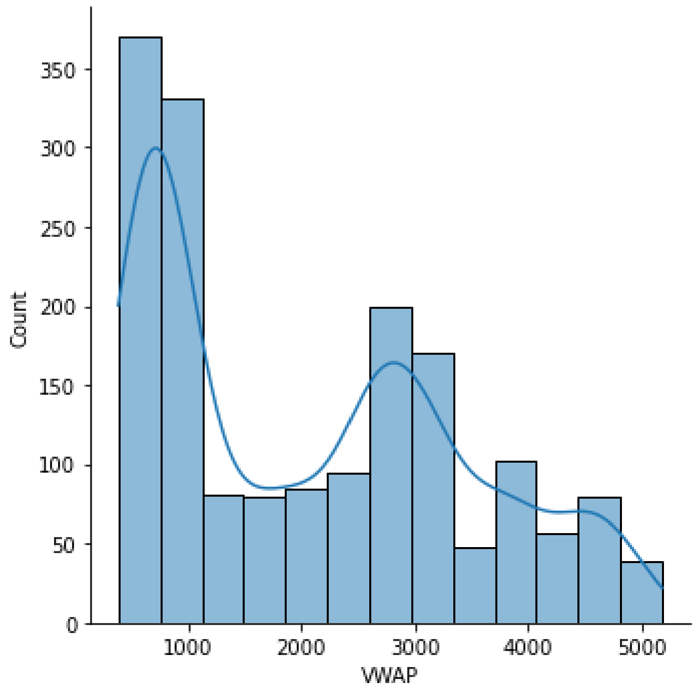

Count
Low
0
50
100
150
200
250
300
350
1000
2000
3000
4000
5000

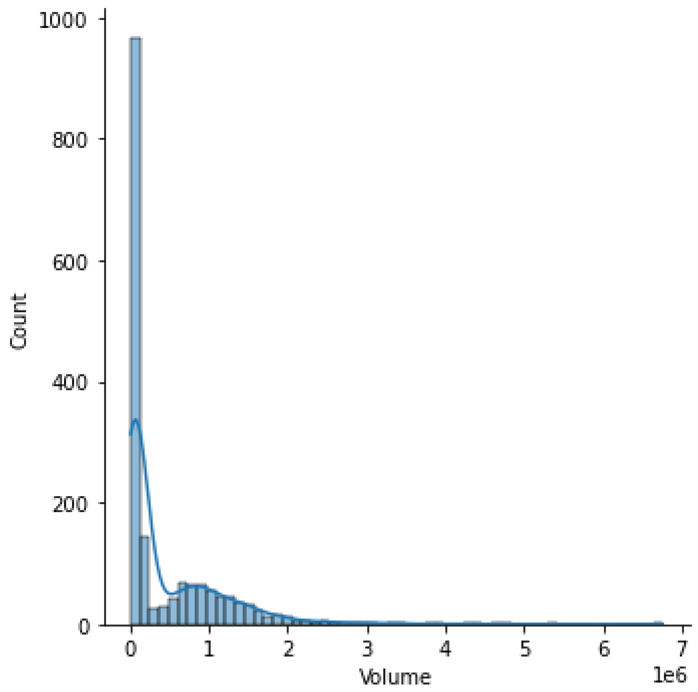


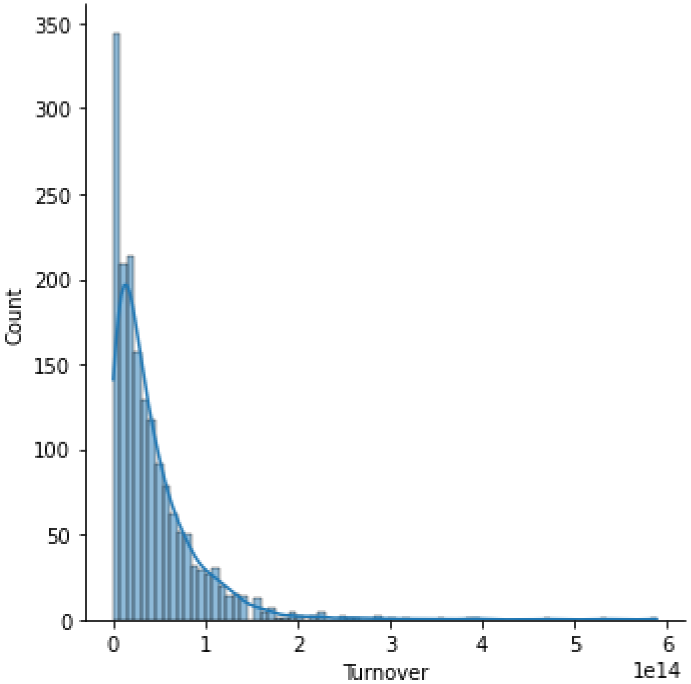
Count
Last

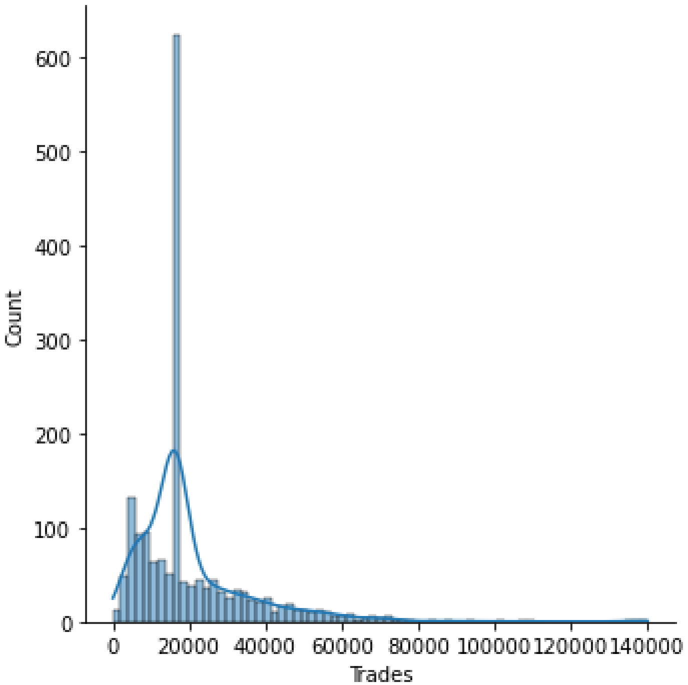


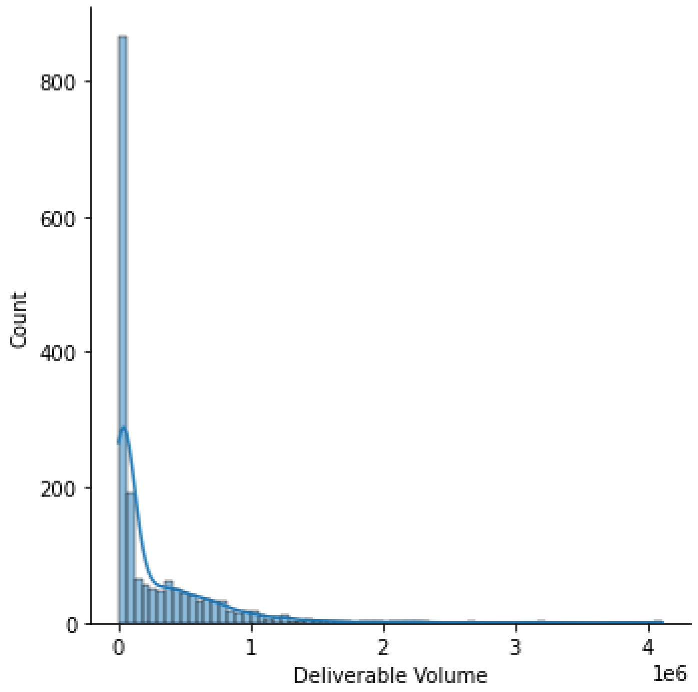
Count
Close

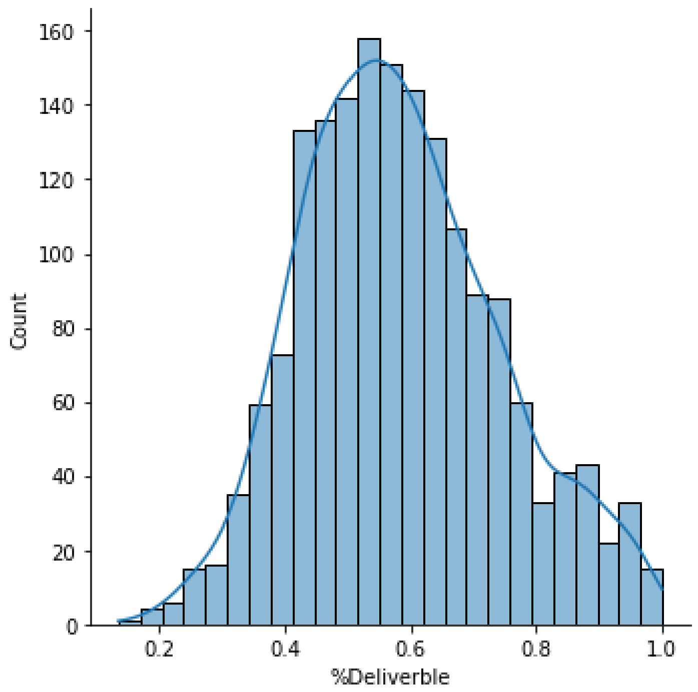

350
300
250
200
150
100
50
0
Count
1000
2000
3000
4000
5000
VWAP

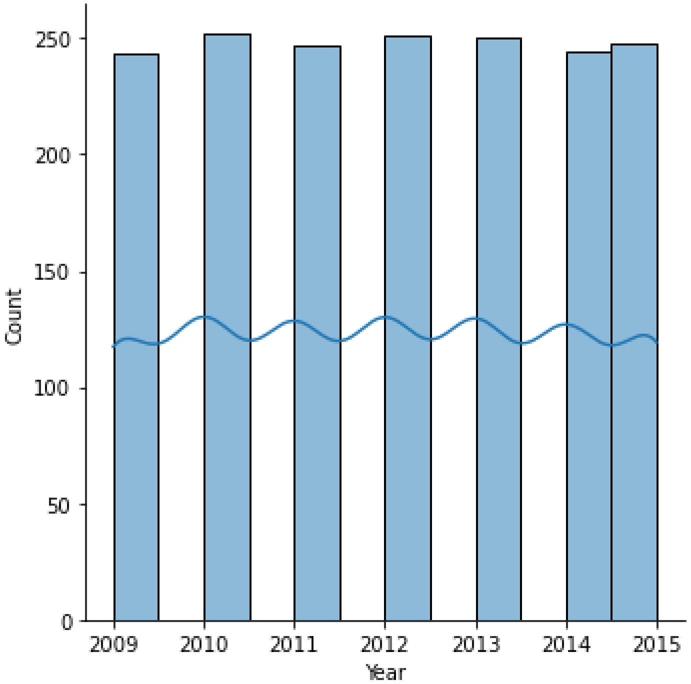

1000
800
600
400
200
0
Count
0
1
2
3
4
5
6
7
1e6
Volume


Count
Turnover
1e14


Count
Trades

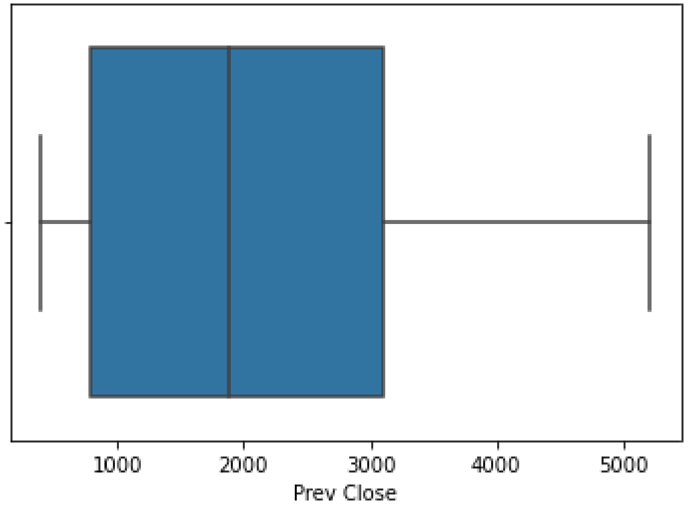

Count
Deliverable Volume
1e6
0
200
400
600
800
0
1
2
3
4

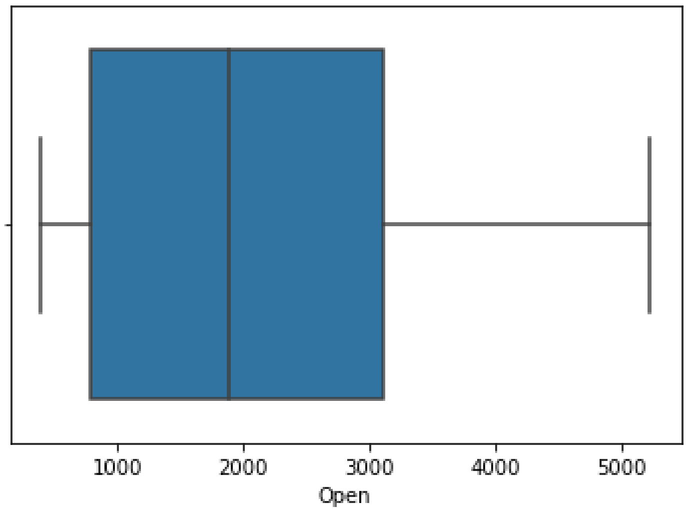

Count
%Deliverble
0
20
40
60
80
100
120
140
160
0.2
0.4
0.6
0.8
1.0

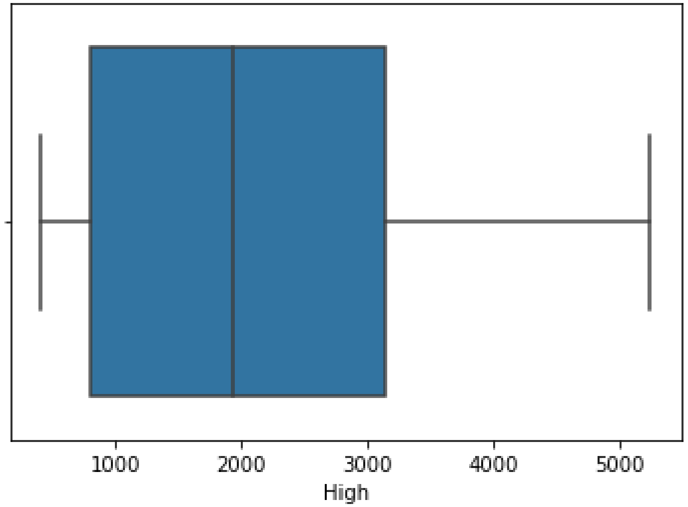


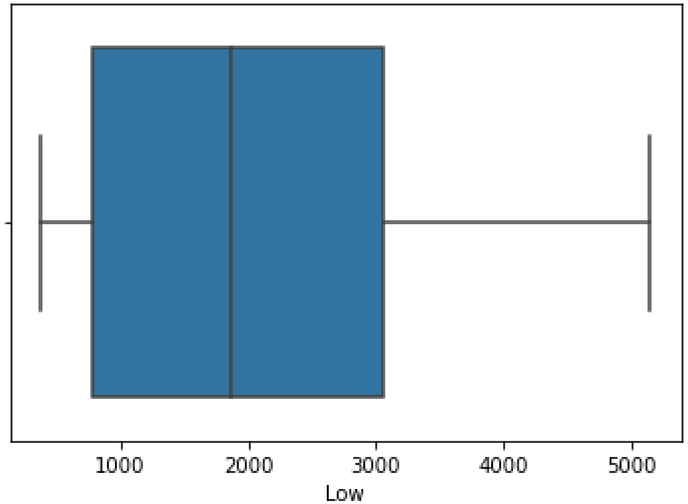
In [31]:

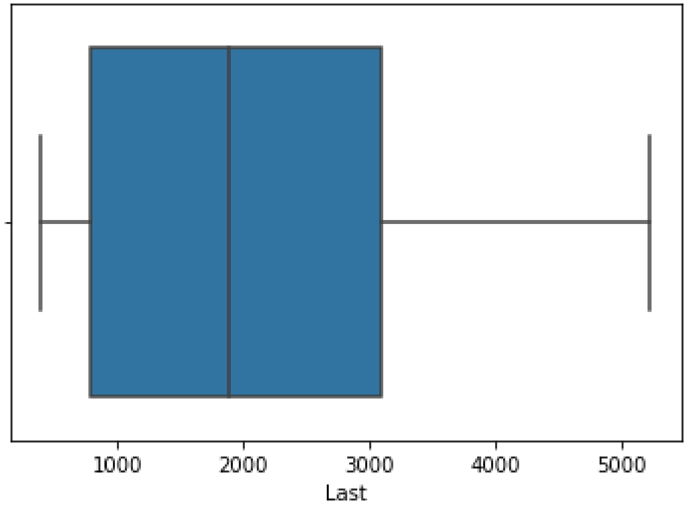
```
# Visualizing th data set for OUTLiers using BOXPLOT
for i in range(train.shape[1]):
    x = train.columns[i]
    fig = sns.boxplot(data=train, x=x)
    filename = "{} Boxplot.jpg".format(train.columns[i])
    fig.figure.savefig(filename,bbox_inches='tight')
    plt.show()
```

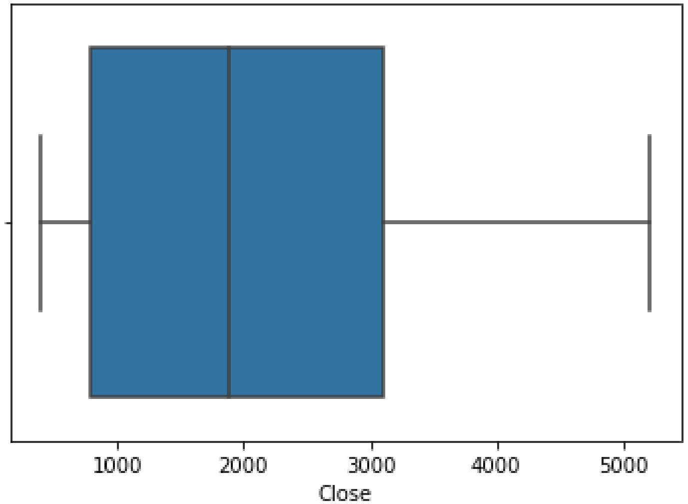

1000
2000
3000
4000
5000
Prev Close

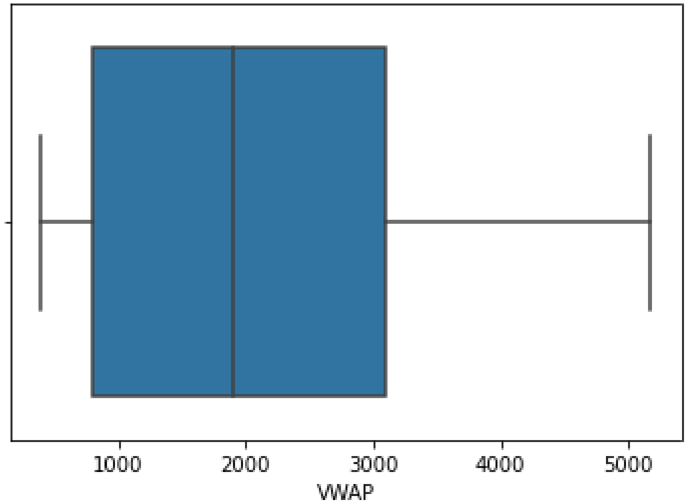

1000
2000
3000
4000
5000
Open

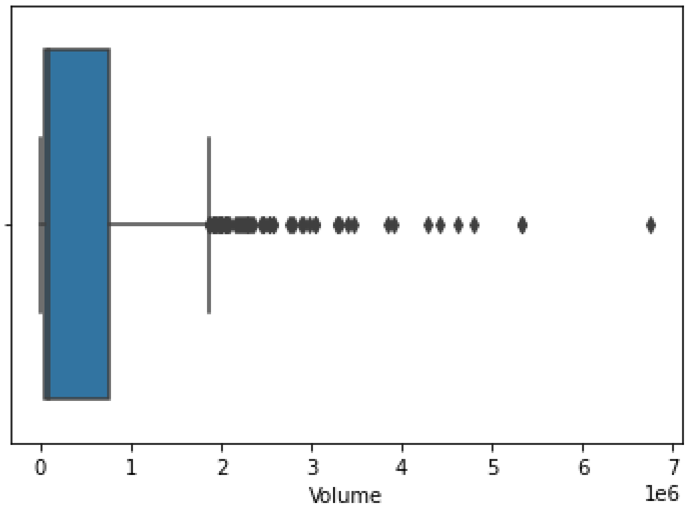

1000
2000
3000
4000
5000
High

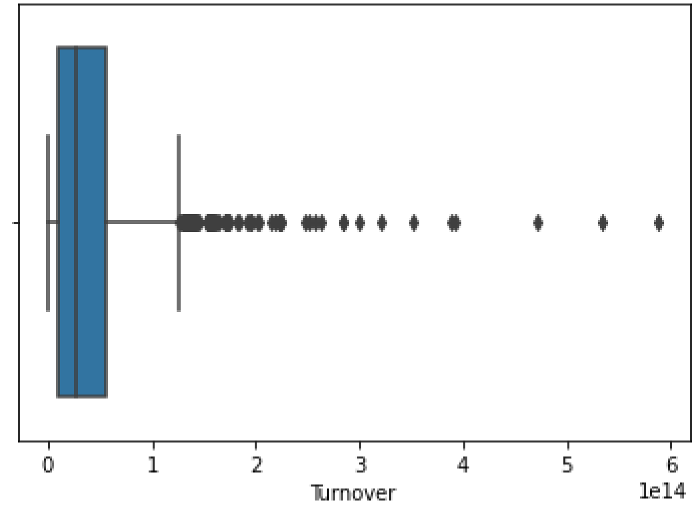

1000
2000
3000
4000
5000
Low

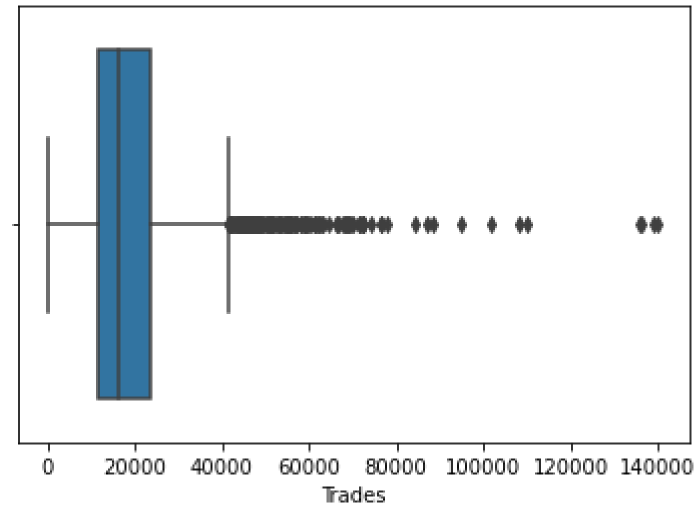

1000
2000
3000
4000
5000
Last

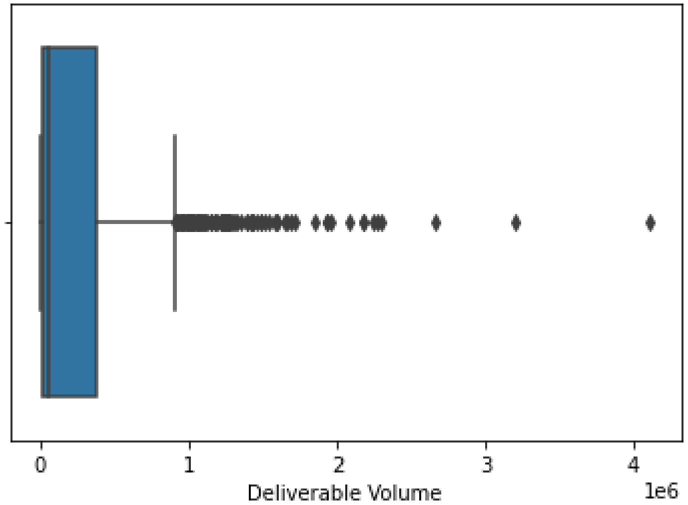

1000
2000
3000
4000
5000
Close

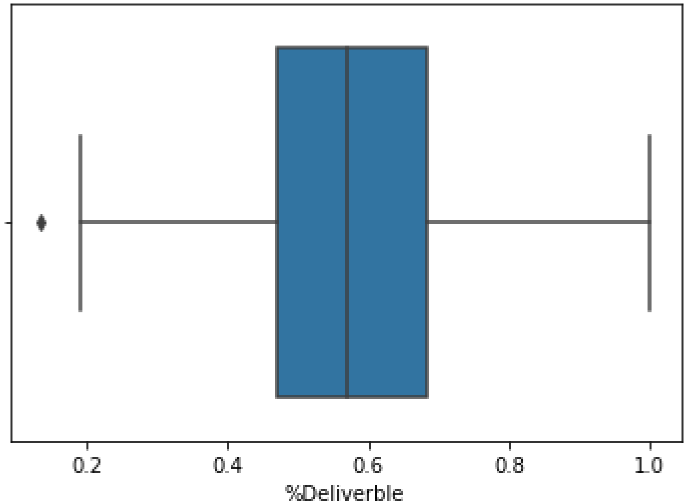

1000
2000
3000
4000
5000
VWAP

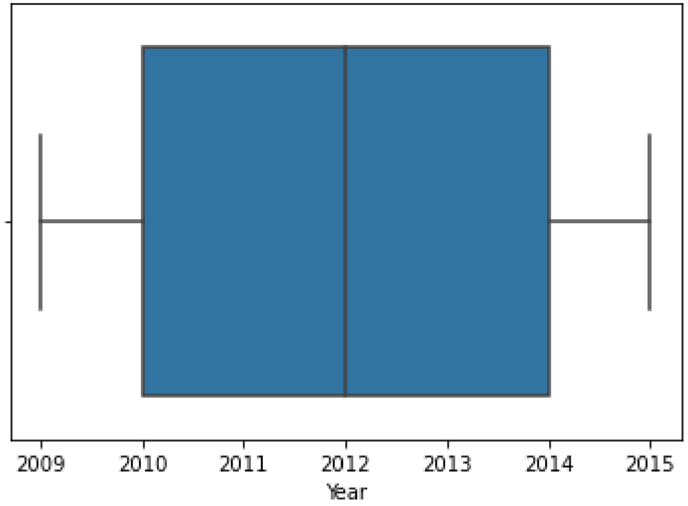

0
1
2
3
4
5
6
7
1e6
Volume


0
1
2
3
4
5
6
Turnover
1e14


0
20000
40000
60000
80000
100000
120000
140000
Trades


0
1
2
3
4
Deliverable Volume
1e6


0.2
0.4
0.6
0.8
1.0
%Deliverble



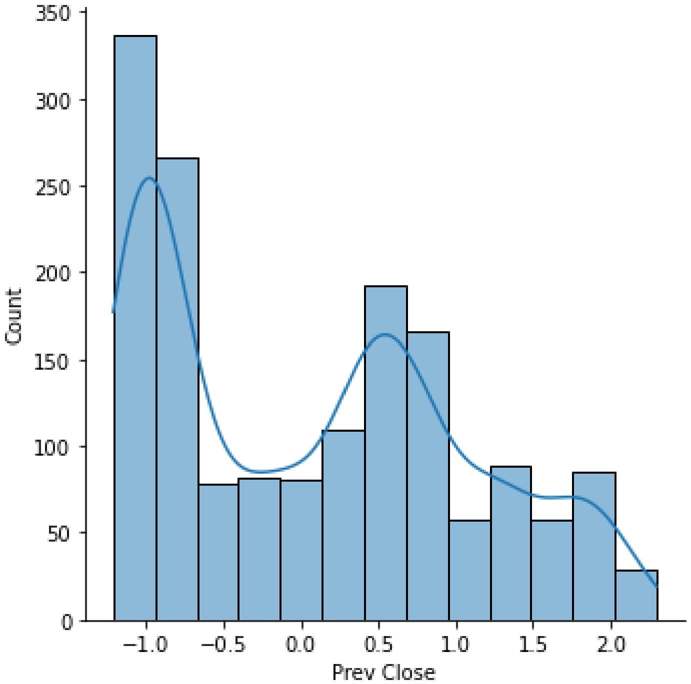
In [32]:
```
#SCALing the Train and Test Datasets Using STANDARD SCALER
scaler = StandardScaler()
scaled_array_train = scaler.fit_transform(train)
train = pd.DataFrame(scaled_array_train, columns = train.columns)
scaled_array_test = scaler.fit_transform(test)
test = pd.DataFrame(scaled_array_test, columns = test.columns)
```

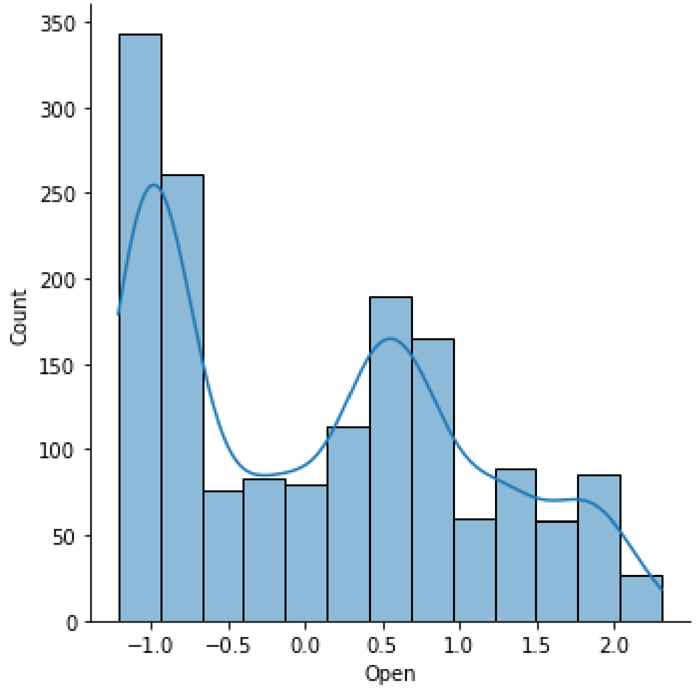
In [33]:
```
# Creating a FUNCTION to remove outliers
def remove_outliers(df):
    for i in range(df.shape[1]):
        col_name = df.columns[i]
        upper_limit = df[col_name].mean() +3*df[col_name].std()
        lower_limit = df[col_name].mean() -3*df[col_name].std()
        df = df[(df[col_name]<upper_limit) & (df[col_name]>lower_limit)]
    return(df)
```

In [34]:
```
#Remove outliers
train = remove_outliers(train)
```

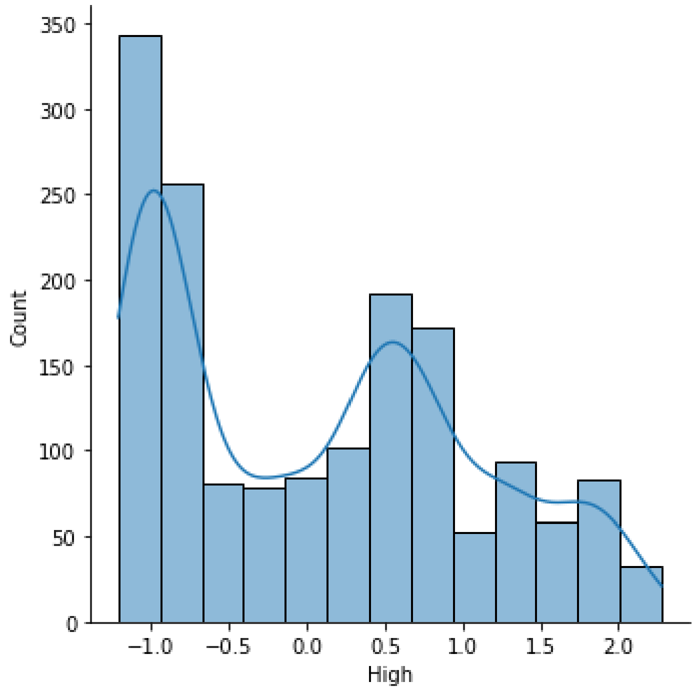
In [35]:
```
# Visualizing the cleaned Data
for i in range(train.shape[1]):
    x = train.columns[i]
```

```
    fig = sns.displot(data=train, x=x, kde=True)
    filename = "{} Cleaned_Histogram.jpg".format(train.columns[i])
    fig.figure.savefig(filename,bbox_inches='tight')
```

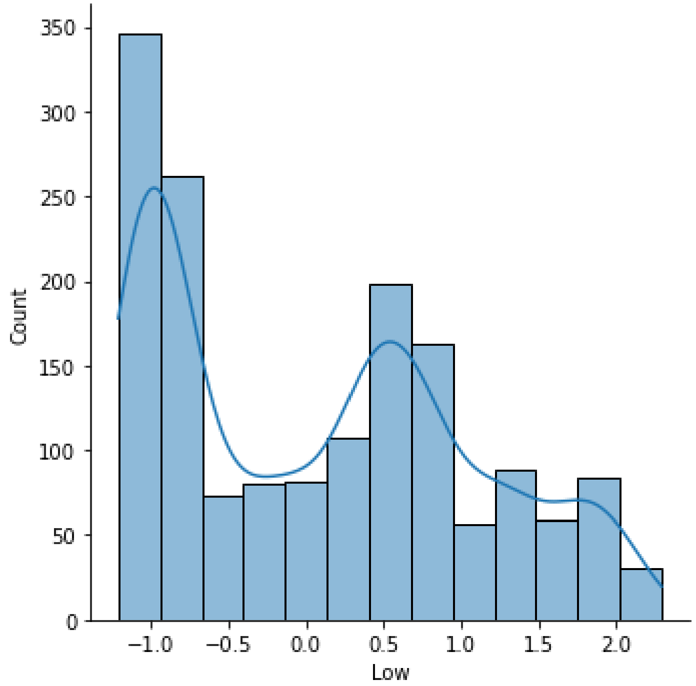


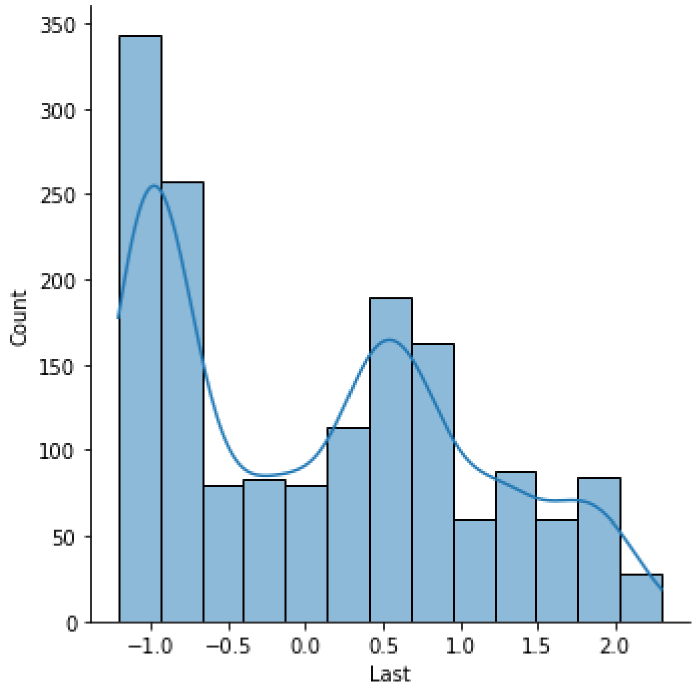


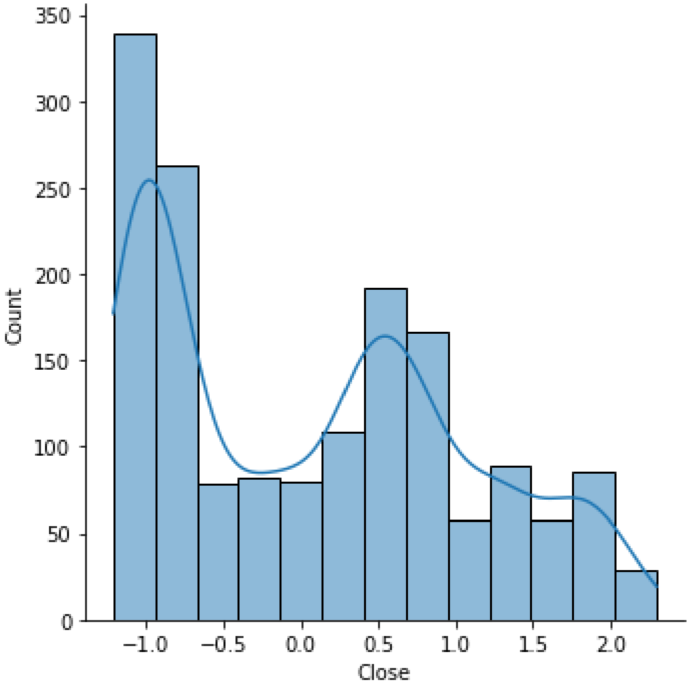
Count
350
300
250
200
150
100
50
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Open

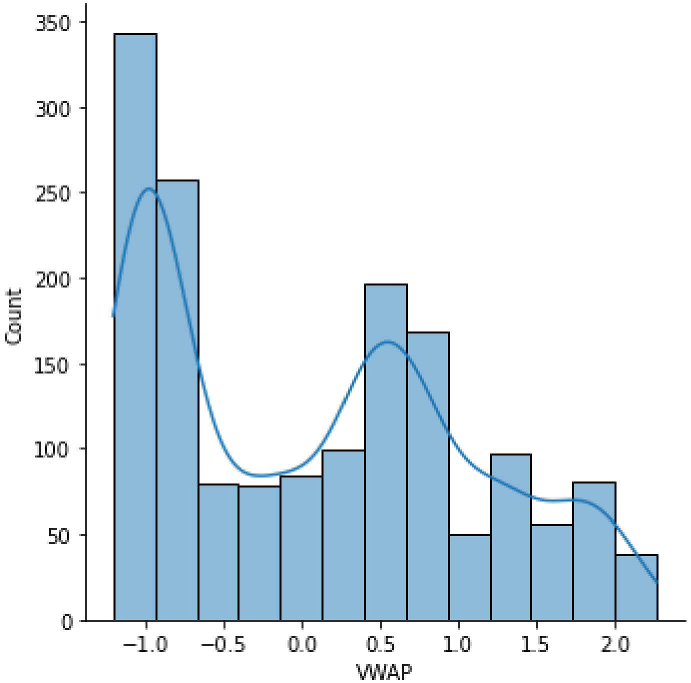


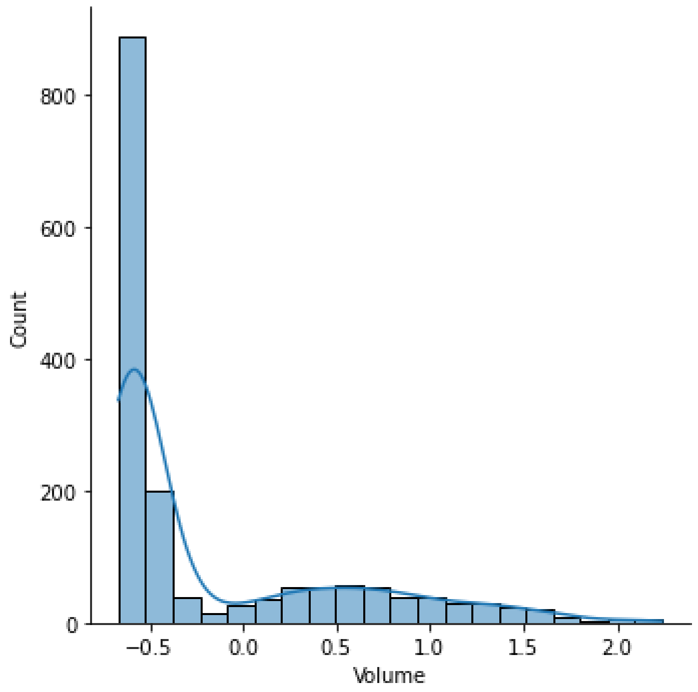
Count
350
300
250
200
150
100
50
0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
High

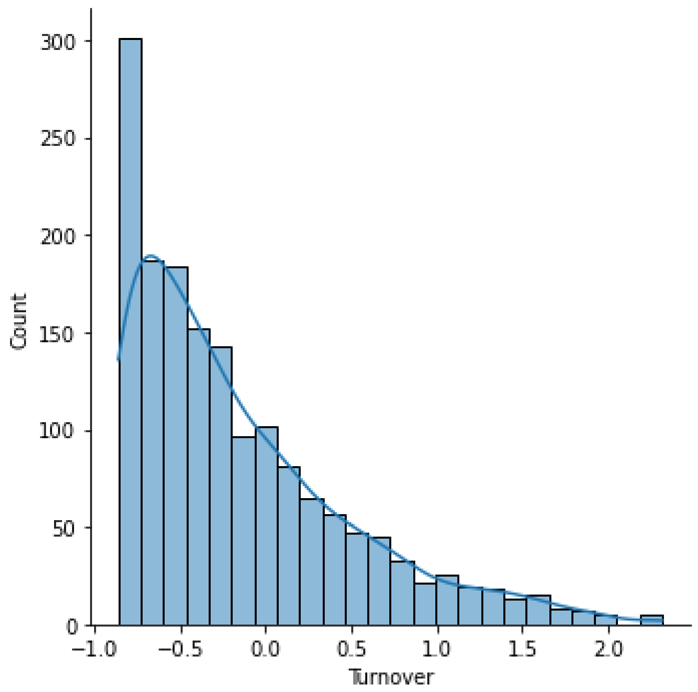

Count
0
50
100
150
200
250
300
350
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Low

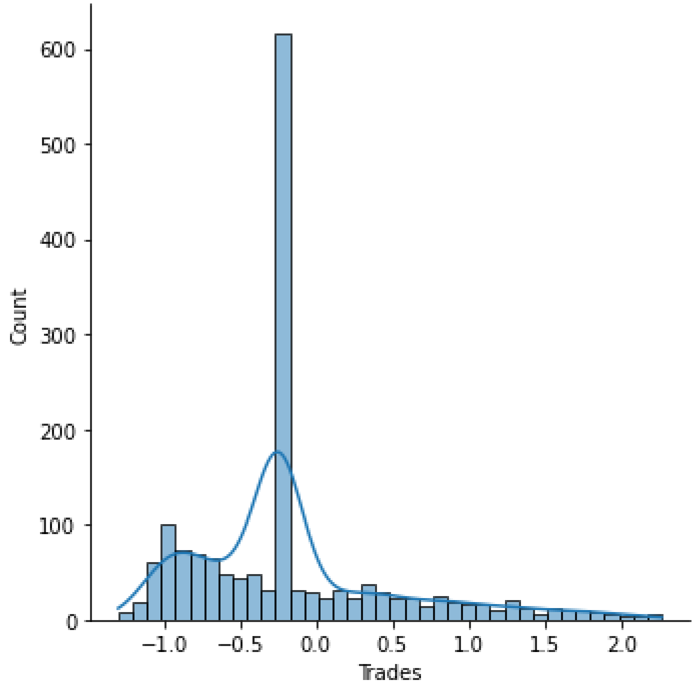

Count
0
50
100
150
200
250
300
350
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Last

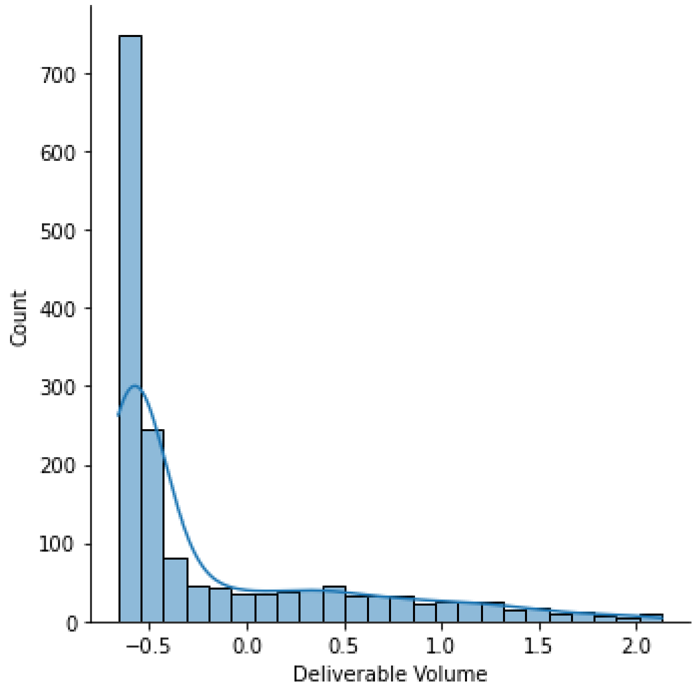

Count
0
50
100
150
200
250
300
350
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Close

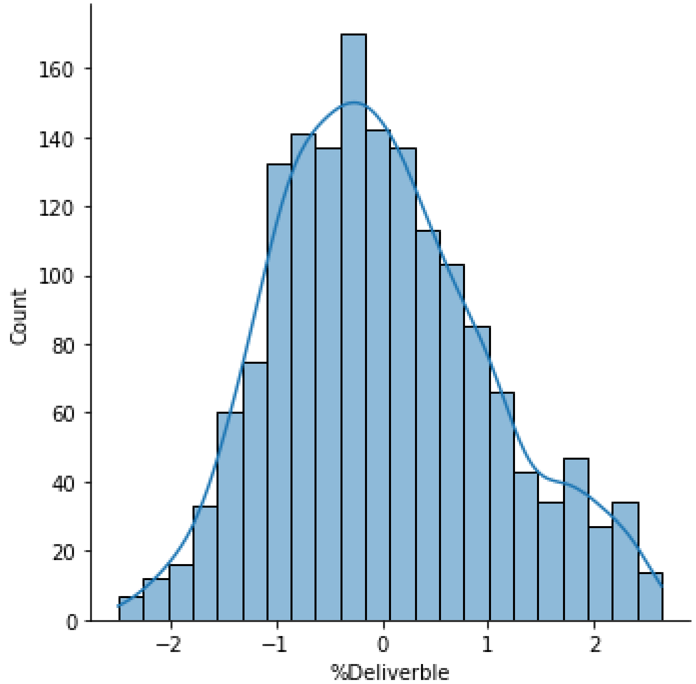

Count
0
50
100
150
200
250
300
350
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
VWAP

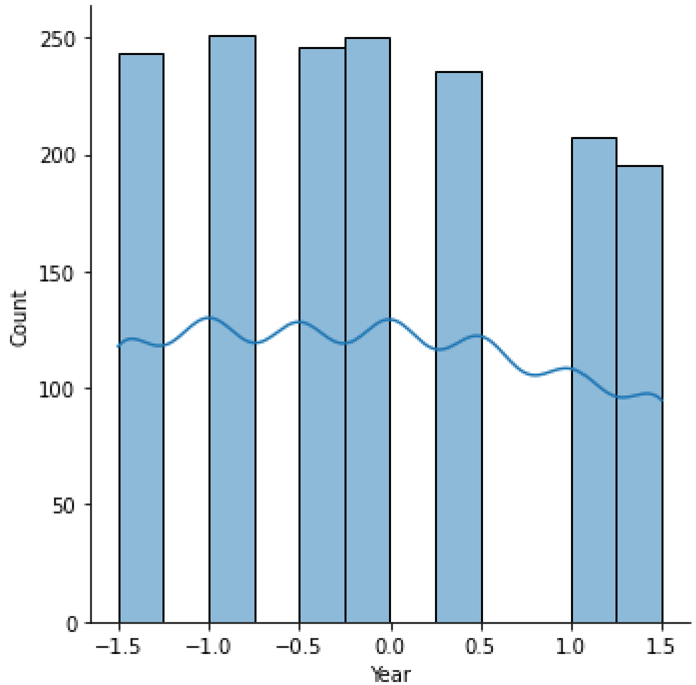

800
600
400
200
0
Count
−0.5
0.0
0.5
1.0
1.5
2.0
Volume


300
250
200
150
100
50
0
Count
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Turnover


600
500
400
300
200
100
0
Count
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Trades


700
600
500
400
300
200
100
0
Count
−0.5
0.0
0.5
1.0
1.5
2.0
Deliverable Volume


Count
%Deliverble

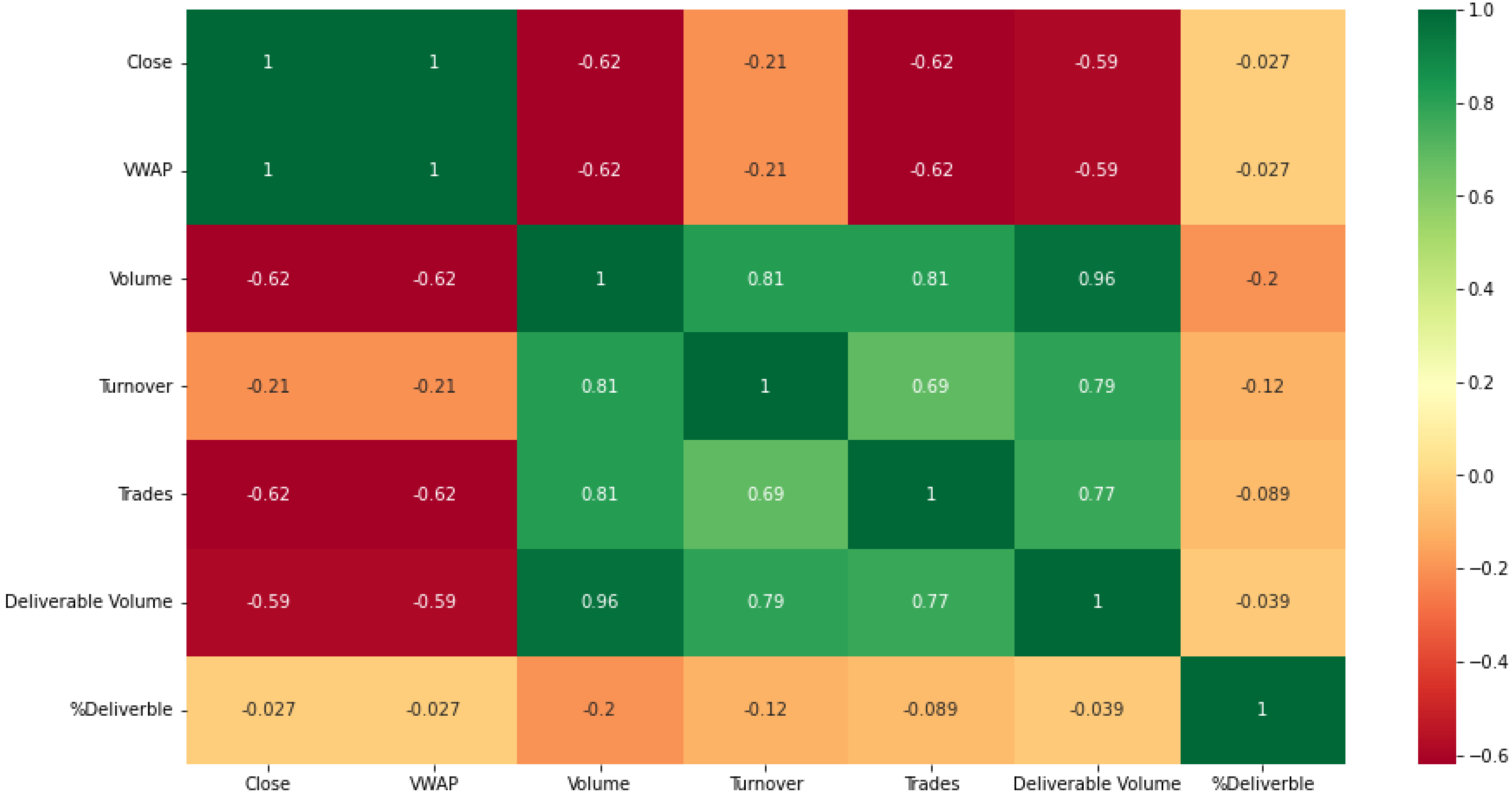

Count
Year

In [36]:

```python
# Visualizing the Data for Corelations Using HEAT MAP
corr= train.corr()
plt.figure(figsize=(15,8))
fig = sns.heatmap(corr,annot=True,cmap='RdYlGn')
fig.figure.savefig("Heat map.jpg",bbox_inches='tight')
plt.show()
```

In [37]:
```
#Dropping the Columns with Corelation of 1 as they donot comtribute to the model creation
train = train.drop(["Prev Close", "Open", "High","Low","Last","Year"],axis = 1)
test = test.drop(["Prev Close", "Open", "High","Low","Last","Year"],axis = 1)
```

In [38]:
```
corr= train.corr()
plt.figure(figsize=(15,8))
fig = sns.heatmap(corr,annot=True,cmap='RdYlGn')
fig.figure.savefig("Feature_selected_Heat map.jpg",bbox_inches='tight')
plt.show()
```



In [39]:
```
#Splitting X_train, Y_train, X_test, Y_test
```

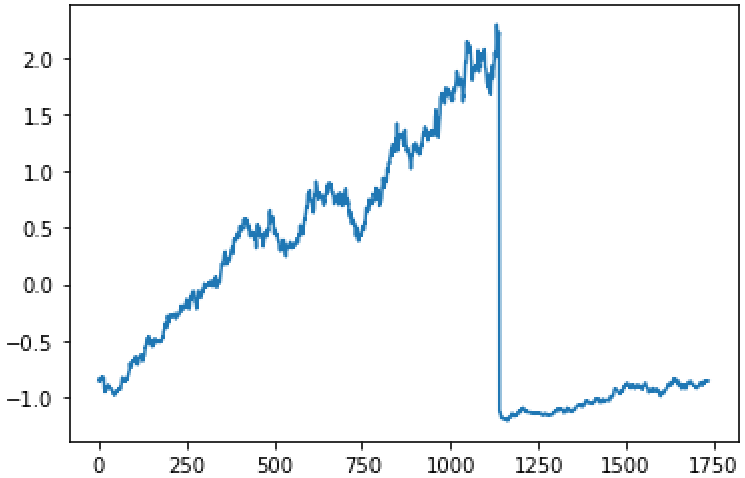
```
X_train = train.drop(["Close"],axis=1)
Y_train = train["Close"]

X_test = test.drop(["Close"],axis=1)
Y_test = test["Close"]
```

In [40]:

```
Y_train.plot()
```

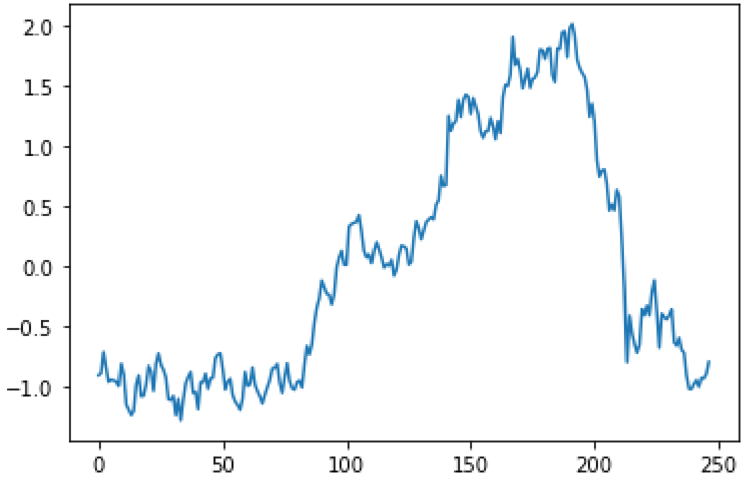
Out[40]: <AxesSubplot:>



In [41]:

```
Y_test.plot()
```

Out[41]: <AxesSubplot:>



In [42]:

```
#List Hyperparameters that we want to tune.
n_neighbors = list(range(1,15))
p=[1,2]
#Convert to dictionary
params = dict(n_neighbors=n_neighbors, p=p)
#Create new KNN object
KNN = KNeighborsRegressor()
#Use GridSearch
GSCV = GridSearchCV(KNN, params, cv=10)
#Fit the model
best_model = GSCV.fit(X_train,Y_train)
#Print The value of best Hyperparameters
print('Best p:', best_model.best_estimator_.get_params()['p'])
print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
```

```
Best p: 2
Best n_neighbors: 2
```

In [43]:

```
# Assigning the parameters tuned
p = best_model.best_estimator_.get_params()['p']
n_neighbors = best_model.best_estimator_.get_params()['n_neighbors']
```

In [44]:

```
#KNN Model creation and fitting the data
model_KNN = KNeighborsRegressor(n_neighbors=n_neighbors,p = p)
model_KNN.fit(X_train,Y_train)
```

Out[44]:
```
KNeighborsRegressor(n_neighbors=2)
```

In [45]:
```
#Random Forest Model creation and fitting the data
model_RF = RandomForestRegressor()
model_RF.fit(X_train, Y_train)
```

Out[45]:
```
RandomForestRegressor()
```

In [46]:
```
#Ada Boost Model creation and fitting the data
model_Ada = AdaBoostRegressor()
model_Ada.fit(X_train, Y_train)
```

Out[46]:
```
AdaBoostRegressor()
```

In [47]:
```
#Creating the structure of the Neural Network model
# Assigning the num of neurons per layer
hidden_layer1 = 150
hidden_layer2 = 200
hidden_layer3 = 250
# Learnig rate and Input dimensions
learning_rate = 0.01
input_dim = X_train.shape[1]
# Creating a Sequential Nueral network model with 3 Dense and 3 Dropuout layers altenatively
model_NN = Sequential()

model_NN.add(Dense(hidden_layer1, input_dim=input_dim, kernel_initializer='normal', activation='relu'))

model_NN.add(Dropout(0.2))

model_NN.add(Dense(hidden_layer2, kernel_initializer='normal', activation='relu'))

model_NN.add(Dropout(0.2))

model_NN.add(Dense(hidden_layer3, kernel_initializer='normal', activation='relu'))

model_NN.add(Dropout(0.2))

model_NN.add(Dense(1, kernel_initializer='normal', activation='linear'))
```

```
#Compilimng the model using Optimizer and learning rate
model_NN.compile(loss='mse',optimizer=Adam(learning_rate=learning_rate),metrics=['mse'])
# train the model
history = model_NN.fit(X_train,Y_train,epochs=300,batch_size=10)
```

```
Epoch 1/300
163/163 [==============================] - 1s 4ms/step - loss: 0.0875 - mse: 0.0876
Epoch 2/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0565 - mse: 0.0566
Epoch 3/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0555 - mse: 0.0554
Epoch 4/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0502 - mse: 0.0503
Epoch 5/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0371 - mse: 0.0371
Epoch 6/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0330 - mse: 0.0330
Epoch 7/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0405 - mse: 0.0405
Epoch 8/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0403 - mse: 0.0404
Epoch 9/300
163/163 [==============================] - 1s 4ms/step - loss: 0.0384 - mse: 0.0383
Epoch 10/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0332 - mse: 0.0332
Epoch 11/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0428 - mse: 0.0428
Epoch 12/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0379 - mse: 0.0379
Epoch 13/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0393 - mse: 0.0393
Epoch 14/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0402 - mse: 0.0402
Epoch 15/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0505 - mse: 0.0505
Epoch 16/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0478 - mse: 0.0478
Epoch 17/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0447 - mse: 0.0446
Epoch 18/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0330 - mse: 0.0330
Epoch 19/300
```

```
163/163 [==============================] - 1s 3ms/step - loss: 0.0418 - mse: 0.0418
Epoch 20/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0330 - mse: 0.0330
Epoch 21/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0323 - mse: 0.0323
Epoch 22/300
163/163 [==============================] - 1s 4ms/step - loss: 0.0318 - mse: 0.0318
Epoch 23/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0318 - mse: 0.0318
Epoch 24/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0327 - mse: 0.0327
Epoch 25/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0369 - mse: 0.0369
Epoch 26/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0393 - mse: 0.0393
Epoch 27/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0368 - mse: 0.0368
Epoch 28/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0392 - mse: 0.0392
Epoch 29/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0303 - mse: 0.0303
Epoch 30/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0427 - mse: 0.0428
Epoch 31/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0435 - mse: 0.0435
Epoch 32/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0351 - mse: 0.0351
Epoch 33/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0430 - mse: 0.0429
Epoch 34/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0401 - mse: 0.0400
Epoch 35/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0406 - mse: 0.0406
Epoch 36/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0345 - mse: 0.0346
Epoch 37/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0387 - mse: 0.0388
Epoch 38/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0434 - mse: 0.0434
Epoch 39/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0405 - mse: 0.0405
Epoch 40/300
163/163 [==============================] - 1s 4ms/step - loss: 0.0504 - mse: 0.0505
Epoch 41/300
```

```
163/163 [==============================] - 1s 3ms/step - loss: 0.0380 - mse: 0.0380
Epoch 42/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0337 - mse: 0.0337
Epoch 43/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0562 - mse: 0.0562
Epoch 44/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0495 - mse: 0.0495
Epoch 45/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0528 - mse: 0.0529
Epoch 46/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0421 - mse: 0.0421
Epoch 47/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0467 - mse: 0.0467
Epoch 48/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0445 - mse: 0.0445
Epoch 49/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0493 - mse: 0.0493
Epoch 50/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0472 - mse: 0.0472
Epoch 51/300
163/163 [==============================] - 0s 3ms/step - loss: 0.0365 - mse: 0.0366
Epoch 52/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0401 - mse: 0.0401
Epoch 53/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0420 - mse: 0.0419
Epoch 54/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0349 - mse: 0.0349
Epoch 55/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0388 - mse: 0.0388
Epoch 56/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0353 - mse: 0.0353
Epoch 57/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0383 - mse: 0.0383
Epoch 58/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0426 - mse: 0.0426
Epoch 59/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0439 - mse: 0.0438
Epoch 60/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0468 - mse: 0.0468
Epoch 61/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0448 - mse: 0.0448
Epoch 62/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0367 - mse: 0.0368
Epoch 63/300
```

```
163/163 [==============================] - 1s 3ms/step - loss: 0.0383 - mse: 0.0383
Epoch 64/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0335 - mse: 0.0335
Epoch 65/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0317 - mse: 0.0318
Epoch 66/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0593 - mse: 0.0593
Epoch 67/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0547 - mse: 0.0548
Epoch 68/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0471 - mse: 0.0470
Epoch 69/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0447 - mse: 0.0446
Epoch 70/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0375 - mse: 0.0376
Epoch 71/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0411 - mse: 0.0411
Epoch 72/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0436 - mse: 0.0436
Epoch 73/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0393 - mse: 0.0393
Epoch 74/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0490 - mse: 0.0490
Epoch 75/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0392 - mse: 0.0392
Epoch 76/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0465 - mse: 0.0465
Epoch 77/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0476 - mse: 0.0476
Epoch 78/300
163/163 [==============================] - 1s 3ms/step - loss: 0.0469 - mse: 0.0470
Epoch 79/300
163/163 [==============================] - 2s 15ms/step - loss: 0.0404 - mse: 0.0404: 0s - loss: 0.0432 - ETA: 0s - loss: 0.0415 -
- ETA: 0s - loss: 0.0407 - mse
Epoch 80/300
163/163 [==============================] - 0s 3ms/step - loss: 0.0438 - mse: 0.0437
Epoch 81/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0479 - mse: 0.0479
Epoch 82/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0451 - mse: 0.0451
Epoch 83/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0370 - mse: 0.0370
Epoch 84/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0455 - mse: 0.0455
```

```
Epoch 85/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0400 - mse: 0.0401
Epoch 86/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0402 - mse: 0.0402
Epoch 87/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0400 - mse: 0.0399
Epoch 88/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0402 - mse: 0.0402
Epoch 89/300
163/163 [==============================] - 0s 3ms/step - loss: 0.0360 - mse: 0.0360
Epoch 90/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0305 - mse: 0.0305
Epoch 91/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0327 - mse: 0.0327
Epoch 92/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0364 - mse: 0.0365
Epoch 93/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0341 - mse: 0.0341
Epoch 94/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0559 - mse: 0.0559
Epoch 95/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0499 - mse: 0.0499
Epoch 96/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0599 - mse: 0.0600
Epoch 97/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0514 - mse: 0.0514
Epoch 98/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0473 - mse: 0.0473
Epoch 99/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0482 - mse: 0.0482
Epoch 100/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0420 - mse: 0.0419
Epoch 101/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0381 - mse: 0.0381
Epoch 102/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0427 - mse: 0.0427
Epoch 103/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0390 - mse: 0.0390
Epoch 104/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0371 - mse: 0.0371
Epoch 105/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0376 - mse: 0.0377
Epoch 106/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0409 - mse: 0.0410
```

```
Epoch 107/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0521 - mse: 0.0522
Epoch 108/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0504 - mse: 0.0504
Epoch 109/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0439 - mse: 0.0440
Epoch 110/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0730 - mse: 0.0731
Epoch 111/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0479 - mse: 0.0479
Epoch 112/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0414 - mse: 0.0414
Epoch 113/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0496 - mse: 0.0496
Epoch 114/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0456 - mse: 0.0454
Epoch 115/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0463 - mse: 0.0464
Epoch 116/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0374 - mse: 0.0374
Epoch 117/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0415 - mse: 0.0415
Epoch 118/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0406 - mse: 0.0406
Epoch 119/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0396 - mse: 0.0396
Epoch 120/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0365 - mse: 0.0365
Epoch 121/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0400 - mse: 0.0401
Epoch 122/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0568 - mse: 0.0568
Epoch 123/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0586 - mse: 0.0584
Epoch 124/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0484 - mse: 0.0483
Epoch 125/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0491 - mse: 0.0492
Epoch 126/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0439 - mse: 0.0439
Epoch 127/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0444 - mse: 0.0444
Epoch 128/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0400 - mse: 0.0400
```

```
Epoch 129/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0356 - mse: 0.0356
Epoch 130/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0392 - mse: 0.0391
Epoch 131/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0379 - mse: 0.0379
Epoch 132/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0387 - mse: 0.0387
Epoch 133/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0490 - mse: 0.0489
Epoch 134/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0526 - mse: 0.0526
Epoch 135/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0459 - mse: 0.0459
Epoch 136/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0434 - mse: 0.0435
Epoch 137/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0385 - mse: 0.0385
Epoch 138/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0453 - mse: 0.0454
Epoch 139/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0379 - mse: 0.0380
Epoch 140/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0476 - mse: 0.0477
Epoch 141/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0425 - mse: 0.0426
Epoch 142/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0406 - mse: 0.0406
Epoch 143/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0380 - mse: 0.0380
Epoch 144/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0373 - mse: 0.0373
Epoch 145/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0380 - mse: 0.0380
Epoch 146/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0377 - mse: 0.0377
Epoch 147/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0456 - mse: 0.0457
Epoch 148/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0407 - mse: 0.0406
Epoch 149/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0400 - mse: 0.0400
Epoch 150/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0369 - mse: 0.0369
```

```
Epoch 151/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0401 - mse: 0.0401
Epoch 152/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0349 - mse: 0.0348
Epoch 153/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0381 - mse: 0.0381
Epoch 154/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0416 - mse: 0.0416
Epoch 155/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0432 - mse: 0.0432
Epoch 156/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0427 - mse: 0.0427
Epoch 157/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0354 - mse: 0.0354
Epoch 158/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0443 - mse: 0.0443
Epoch 159/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0442 - mse: 0.0443
Epoch 160/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0336 - mse: 0.0335
Epoch 161/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0376 - mse: 0.0376
Epoch 162/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0420 - mse: 0.0420
Epoch 163/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0400 - mse: 0.0400
Epoch 164/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0431 - mse: 0.0431
Epoch 165/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0398 - mse: 0.0398
Epoch 166/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0377 - mse: 0.0377
Epoch 167/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0485 - mse: 0.0485
Epoch 168/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0441 - mse: 0.0442
Epoch 169/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0437 - mse: 0.0437
Epoch 170/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0421 - mse: 0.0421
Epoch 171/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0378 - mse: 0.0378
Epoch 172/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0435 - mse: 0.0434
```

```
Epoch 173/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0417 - mse: 0.0416
Epoch 174/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0462 - mse: 0.0461
Epoch 175/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0473 - mse: 0.0473
Epoch 176/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0445 - mse: 0.0445
Epoch 177/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0417 - mse: 0.0417
Epoch 178/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0467 - mse: 0.0467
Epoch 179/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0497 - mse: 0.0497
Epoch 180/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0529 - mse: 0.0529
Epoch 181/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0499 - mse: 0.0499
Epoch 182/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0426 - mse: 0.0426
Epoch 183/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0431 - mse: 0.0431
Epoch 184/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0463 - mse: 0.0463
Epoch 185/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0456 - mse: 0.0457
Epoch 186/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0372 - mse: 0.0371
Epoch 187/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0416 - mse: 0.0416
Epoch 188/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0465 - mse: 0.0465
Epoch 189/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0459 - mse: 0.0458
Epoch 190/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0680 - mse: 0.0677
Epoch 191/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0467 - mse: 0.0467
Epoch 192/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0365 - mse: 0.0364
Epoch 193/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0386 - mse: 0.0386
Epoch 194/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0525 - mse: 0.0525
```

```
Epoch 195/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0472 - mse: 0.0473
Epoch 196/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0478 - mse: 0.0478
Epoch 197/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0504 - mse: 0.0504
Epoch 198/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0395 - mse: 0.0394
Epoch 199/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0464 - mse: 0.0464
Epoch 200/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0587 - mse: 0.0588
Epoch 201/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0411 - mse: 0.0411
Epoch 202/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0543 - mse: 0.0543
Epoch 203/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0604 - mse: 0.0604
Epoch 204/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0529 - mse: 0.0529
Epoch 205/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0529 - mse: 0.0529
Epoch 206/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0506 - mse: 0.0507
Epoch 207/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0465 - mse: 0.0466
Epoch 208/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0529 - mse: 0.0529
Epoch 209/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0430 - mse: 0.0430
Epoch 210/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0399 - mse: 0.0398
Epoch 211/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0411 - mse: 0.0411
Epoch 212/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0539 - mse: 0.0540
Epoch 213/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0596 - mse: 0.0597
Epoch 214/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0470 - mse: 0.0470
Epoch 215/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0455 - mse: 0.0455
Epoch 216/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0599 - mse: 0.0600
```

```
Epoch 217/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0499 - mse: 0.0499
Epoch 218/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0526 - mse: 0.0526
Epoch 219/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0625 - mse: 0.0626
Epoch 220/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0450 - mse: 0.0450
Epoch 221/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0623 - mse: 0.0623
Epoch 222/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0551 - mse: 0.0551
Epoch 223/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0496 - mse: 0.0496
Epoch 224/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0400 - mse: 0.0399
Epoch 225/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0448 - mse: 0.0448
Epoch 226/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0416 - mse: 0.0416
Epoch 227/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0414 - mse: 0.0414
Epoch 228/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0532 - mse: 0.0532
Epoch 229/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0402 - mse: 0.0402
Epoch 230/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0404 - mse: 0.0404
Epoch 231/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0425 - mse: 0.0424
Epoch 232/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0453 - mse: 0.0453
Epoch 233/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0574 - mse: 0.0575
Epoch 234/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0529 - mse: 0.0530
Epoch 235/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0506 - mse: 0.0506
Epoch 236/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0570 - mse: 0.0570
Epoch 237/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0512 - mse: 0.0512
Epoch 238/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0521 - mse: 0.0522
```

```
Epoch 239/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0597 - mse: 0.0597
Epoch 240/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0629 - mse: 0.0629
Epoch 241/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0813 - mse: 0.0814
Epoch 242/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0662 - mse: 0.0662
Epoch 243/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0627 - mse: 0.0627
Epoch 244/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0497 - mse: 0.0498
Epoch 245/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0471 - mse: 0.0472
Epoch 246/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0495 - mse: 0.0493
Epoch 247/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0494 - mse: 0.0495
Epoch 248/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0505 - mse: 0.0506
Epoch 249/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0503 - mse: 0.0502
Epoch 250/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0431 - mse: 0.0430
Epoch 251/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0407 - mse: 0.0407
Epoch 252/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0474 - mse: 0.0474
Epoch 253/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0633 - mse: 0.0633
Epoch 254/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0444 - mse: 0.0443
Epoch 255/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0456 - mse: 0.0456
Epoch 256/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0453 - mse: 0.0453
Epoch 257/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0640 - mse: 0.0641
Epoch 258/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0398 - mse: 0.0398
Epoch 259/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0473 - mse: 0.0473
Epoch 260/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0481 - mse: 0.0481
```

```
Epoch 261/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0507 - mse: 0.0507
Epoch 262/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0513 - mse: 0.0514
Epoch 263/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0606 - mse: 0.0607
Epoch 264/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0610 - mse: 0.0610
Epoch 265/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0551 - mse: 0.0550
Epoch 266/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0570 - mse: 0.0570
Epoch 267/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0578 - mse: 0.0577
Epoch 268/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0484 - mse: 0.0484
Epoch 269/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0507 - mse: 0.0507
Epoch 270/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0427 - mse: 0.0426
Epoch 271/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0463 - mse: 0.0463
Epoch 272/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0431 - mse: 0.0431
Epoch 273/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0551 - mse: 0.0551
Epoch 274/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0532 - mse: 0.0532
Epoch 275/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0534 - mse: 0.0534
Epoch 276/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0565 - mse: 0.0565
Epoch 277/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0474 - mse: 0.0474
Epoch 278/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0524 - mse: 0.0523
Epoch 279/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0544 - mse: 0.0545
Epoch 280/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0640 - mse: 0.0640
Epoch 281/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0415 - mse: 0.0416
Epoch 282/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0487 - mse: 0.0487
```

```
Epoch 283/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0421 - mse: 0.0420
Epoch 284/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0486 - mse: 0.0486
Epoch 285/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0409 - mse: 0.0409
Epoch 286/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0467 - mse: 0.0467
Epoch 287/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0488 - mse: 0.0489A: 0s - loss: 0.0472 - mse: 0.047
Epoch 288/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0499 - mse: 0.0500
Epoch 289/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0513 - mse: 0.0514
Epoch 290/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0425 - mse: 0.0425
Epoch 291/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0441 - mse: 0.0441
Epoch 292/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0417 - mse: 0.0416
Epoch 293/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0343 - mse: 0.0342
Epoch 294/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0466 - mse: 0.0465
Epoch 295/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0578 - mse: 0.0578
Epoch 296/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0640 - mse: 0.0640
Epoch 297/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0466 - mse: 0.0463
Epoch 298/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0359 - mse: 0.0360
Epoch 299/300
163/163 [==============================] - 0s 2ms/step - loss: 0.0399 - mse: 0.0399
Epoch 300/300
163/163 [==============================] - 0s 1ms/step - loss: 0.0484 - mse: 0.0484
```

Metrics

- MSE -- Mean Squared Error
- RMSE -- Root mean squred error

- R2 -- R Squared
- ADJ_R2 -- Adjusted R Squared
- MAPE -- Mean Absolute Percentage Error

In [48]:

```
# Created a function to evaluate the metrics of all the models

def Evaluate_models(model):
    y_pred = model.predict(X_test)

    MSE = mean_squared_error(Y_test, y_pred)

    RMSE = mean_squared_error(Y_test, y_pred, squared=False)

    R2 = r2_score(Y_test, y_pred)

    ADJ_R2 = 1 - (1-R2)*(len(Y_train)-1)/(X_train.shape[0]-X_train.shape[1]-1)

    MAPE = mean_absolute_percentage_error(Y_test, y_pred)

    return (MSE,RMSE,R2,ADJ_R2,MAPE)
```

In [49]:

```
# Evaluating all the metrics for K Nearest Neighbour model
Metrics_KNN  = Evaluate_models(model_KNN)
MSE_KNN = Metrics_KNN[0]
RMSE_KNN = Metrics_KNN[1]
R2_KNN = Metrics_KNN[2]
ADJ_R2_KNN = Metrics_KNN[3]
MAPE_KNN = Metrics_KNN[4]
```

In [50]:

```
# Evaluating all the metrics for Random Forest model
Metrics_RF  = Evaluate_models(model_RF)
MSE_RF= Metrics_RF[0]
RMSE_RF = Metrics_RF[1]
R2_RF = Metrics_RF[2]
ADJ_R2_RF = Metrics_RF[3]
MAPE_RF = Metrics_RF[4]
```

In [51]:

```
# Evaluating all the metrics for Ada Boost model
```

```
Metrics_Ada  = Evaluate_models(model_Ada)
MSE_Ada= Metrics_Ada[0]
RMSE_Ada = Metrics_Ada[1]
R2_Ada = Metrics_Ada[2]
ADJ_R2_Ada = Metrics_Ada[3]
MAPE_Ada = Metrics_Ada[4]
```

In [52]:

```
# Evaluating all the metrics for Neural Network model
Metrics_NN  = Evaluate_models(model_NN)
MSE_NN= Metrics_NN[0]
RMSE_NN = Metrics_NN[1]
R2_NN = Metrics_NN[2]
ADJ_R2_NN = Metrics_NN[3]
MAPE_NN = Metrics_NN[4]
```

Creating the Dataframe with Metrics corresponding to all the models created

In [53]:

```
Comp={'Models':['KNN','Random Forest','AdaBoost',"Neural Network"],
      'MSE' :[MSE_KNN,MSE_RF,MSE_Ada,MSE_NN],
      'RMSE' :[RMSE_KNN,RMSE_RF,RMSE_Ada,RMSE_NN],
      'R2' :[R2_KNN,R2_RF,R2_Ada,R2_NN],
      'ADJ R2' :[ADJ_R2_KNN,ADJ_R2_RF,ADJ_R2_Ada,ADJ_R2_NN],
      'MAPE' :[MAPE_KNN,MAPE_RF,MAPE_Ada,MAPE_NN]

   }

Compare=pd.DataFrame(Comp)
Compare
```

Out[53]:

	Models	MSE	RMSE	R2	ADJ R2	MAPE
0	KNN	0.138955	0.372766	0.861045	0.860531	1.319430
1	Random Forest	0.002505	0.050049	0.997495	0.997486	0.143844
2	AdaBoost	0.007262	0.085217	0.992738	0.992711	0.361345
3	Neural Network	0.191309	0.437389	0.808691	0.807983	1.496710

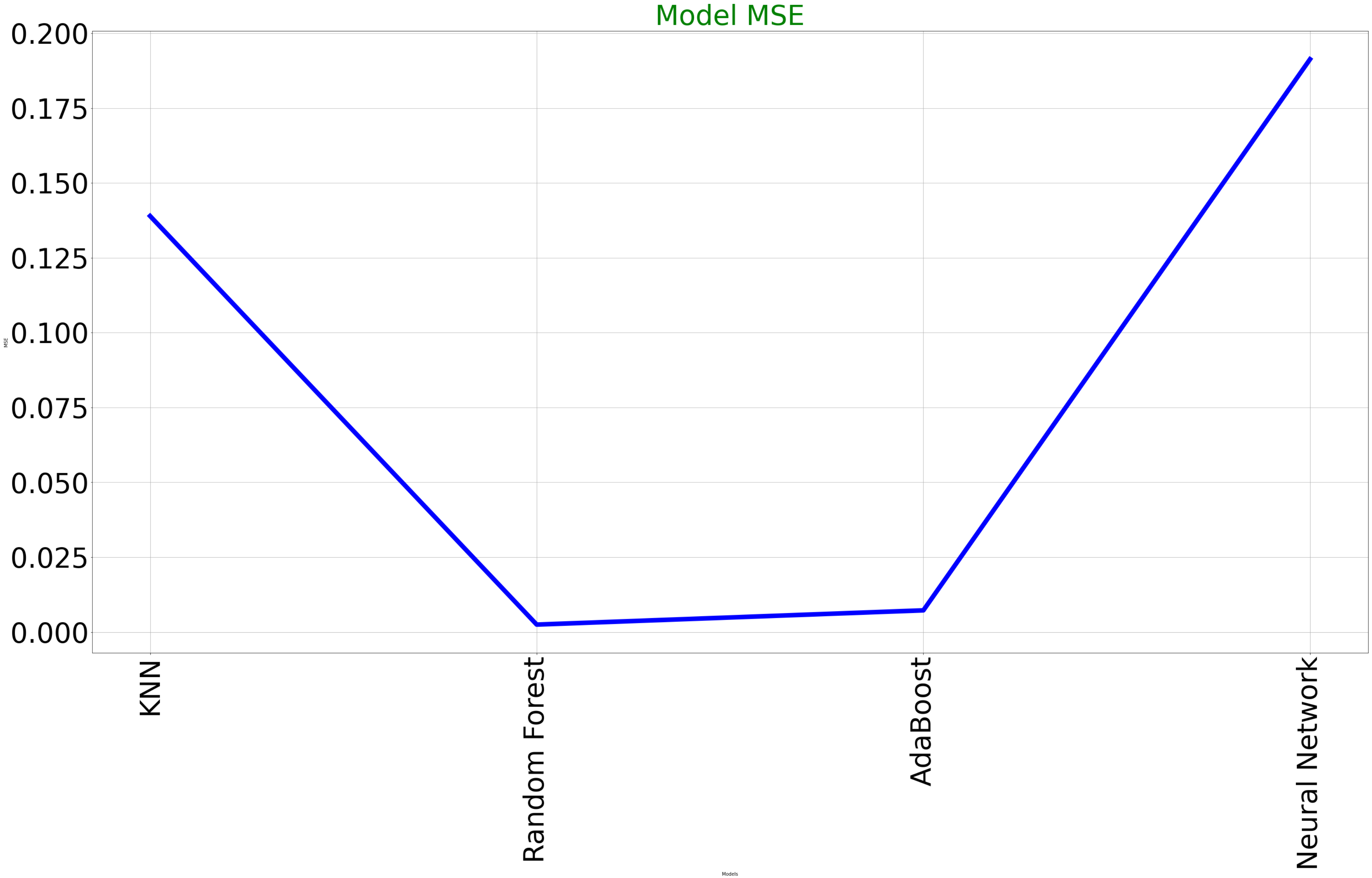
Plotting Model MSE

In [54]:

```
plt.figure(figsize =(50, 25))
plt.plot(Compare['Models'],Compare['MSE'],c='blue', lw=10)

plt.title('Model MSE',fontdict={'fontsize': 60,'fontweight' : 60,'color' : 'g'})
plt.xlabel('Models')
plt.ylabel('MSE')

plt.yticks(fontsize=60)
plt.xticks(rotation=90, fontsize=60)
plt.grid()
plt.savefig("MSE.jpg",bbox_inches='tight')
plt.show()
```



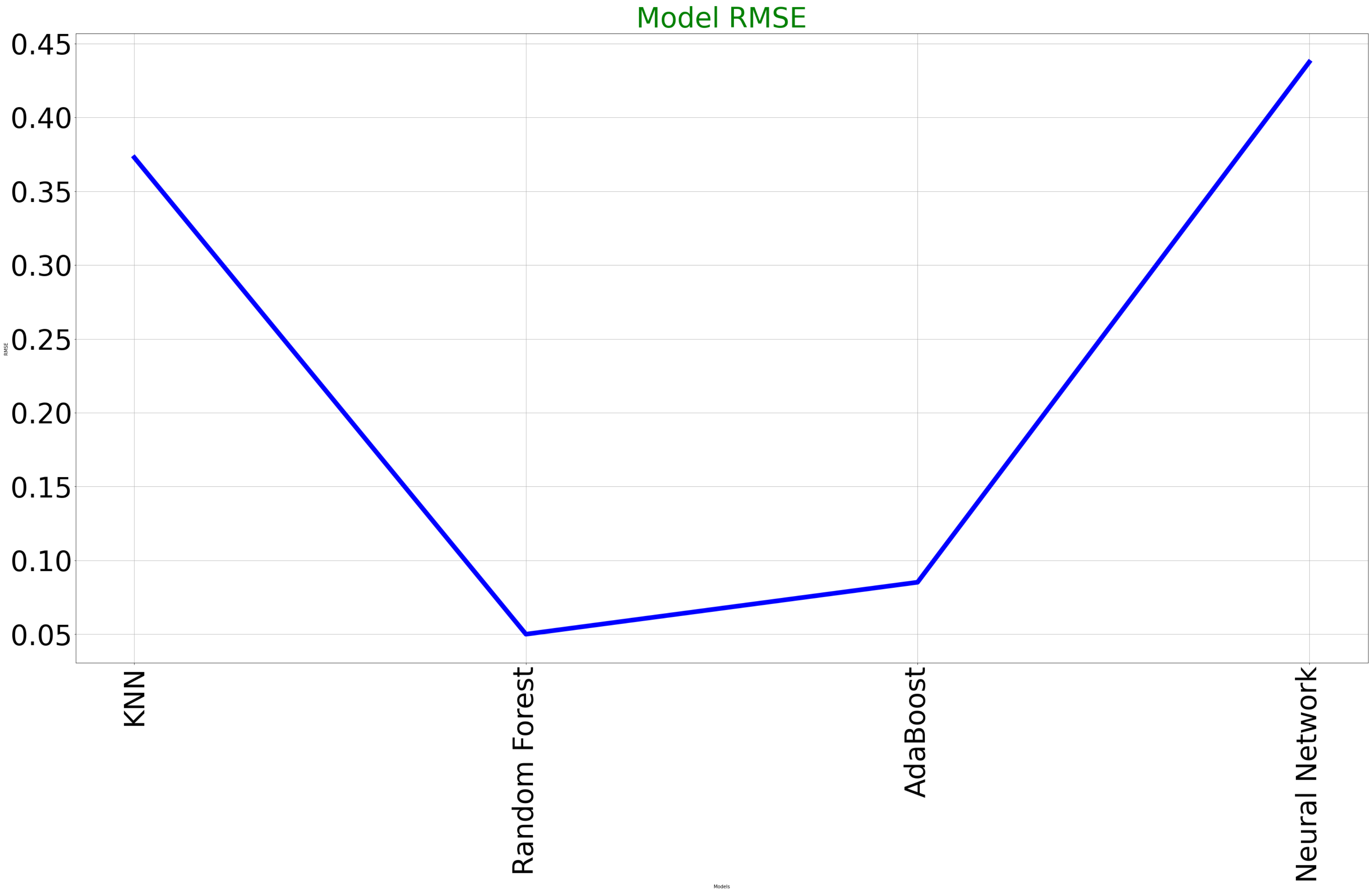
Plotting Model RMSE

In [55]:
```
plt.figure(figsize =(50, 25))
```

```
plt.plot(Compare['Models'],Compare['RMSE'],c='blue', lw=10)

plt.title('Model RMSE',fontdict={'fontsize': 60,'fontweight' : 60,'color' : 'g'})
plt.xlabel('Models')
plt.ylabel('RMSE')

plt.yticks(fontsize=60)
plt.xticks(rotation=90, fontsize=60)
plt.grid()
plt.savefig("RMSE.jpg",bbox_inches='tight')
plt.show()
```



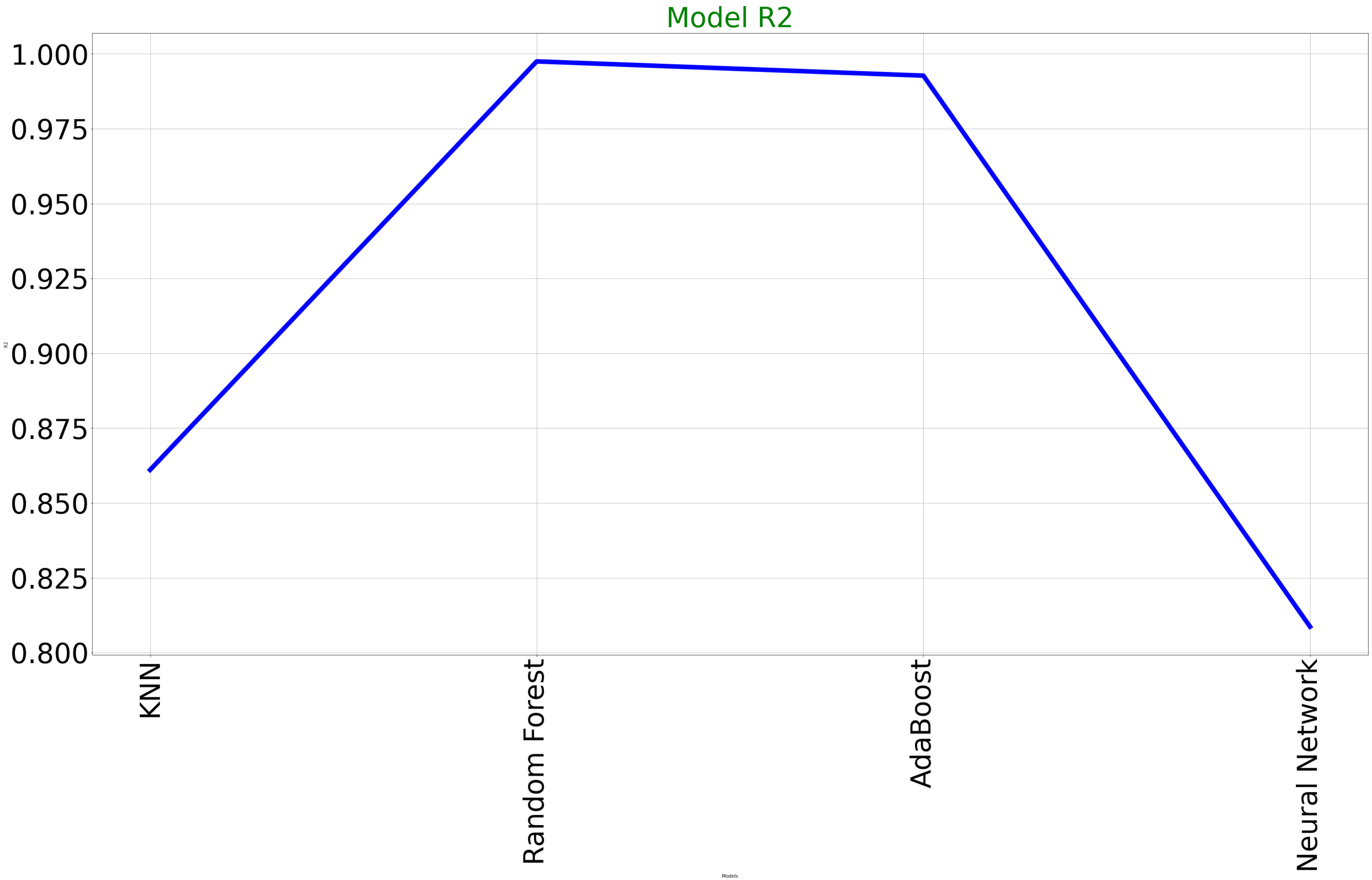
Plotting Model R Squared

In [56]:
```
plt.figure(figsize =(50, 25))
```

```
plt.plot(Compare['Models'],Compare['R2'],c='blue', lw=10)

plt.title('Model R2',fontdict={'fontsize': 60,'fontweight' : 60,'color' : 'g'})
plt.xlabel('Models')
plt.ylabel('R2')

plt.yticks(fontsize=60)
plt.xticks(rotation=90, fontsize=60)
plt.grid()
plt.savefig("R2.jpg",bbox_inches='tight')
plt.show()
```



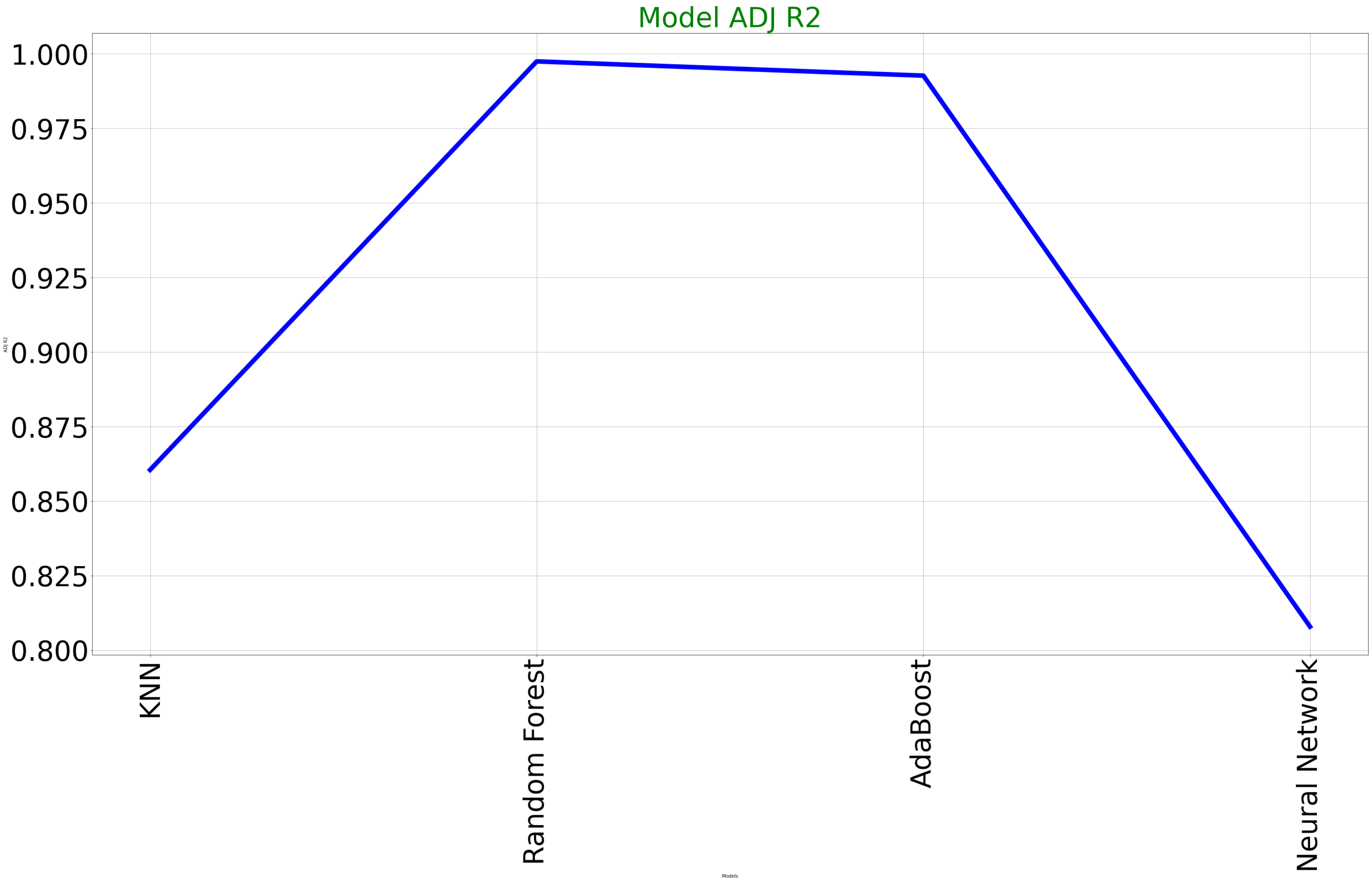
Plotting Model Adjusted R Squared

In [57]:
```
plt.figure(figsize =(50, 25))
```

```
plt.plot(Compare['Models'],Compare['ADJ R2'],c='blue', lw=10)

plt.title('Model ADJ R2',fontdict={'fontsize': 60,'fontweight' : 60,'color' : 'g'})
plt.xlabel('Models')
plt.ylabel('ADJ R2')

plt.yticks(fontsize=60)
plt.xticks(rotation=90, fontsize=60)
plt.grid()
plt.savefig("ADJ R2.jpg",bbox_inches='tight')
plt.show()
```



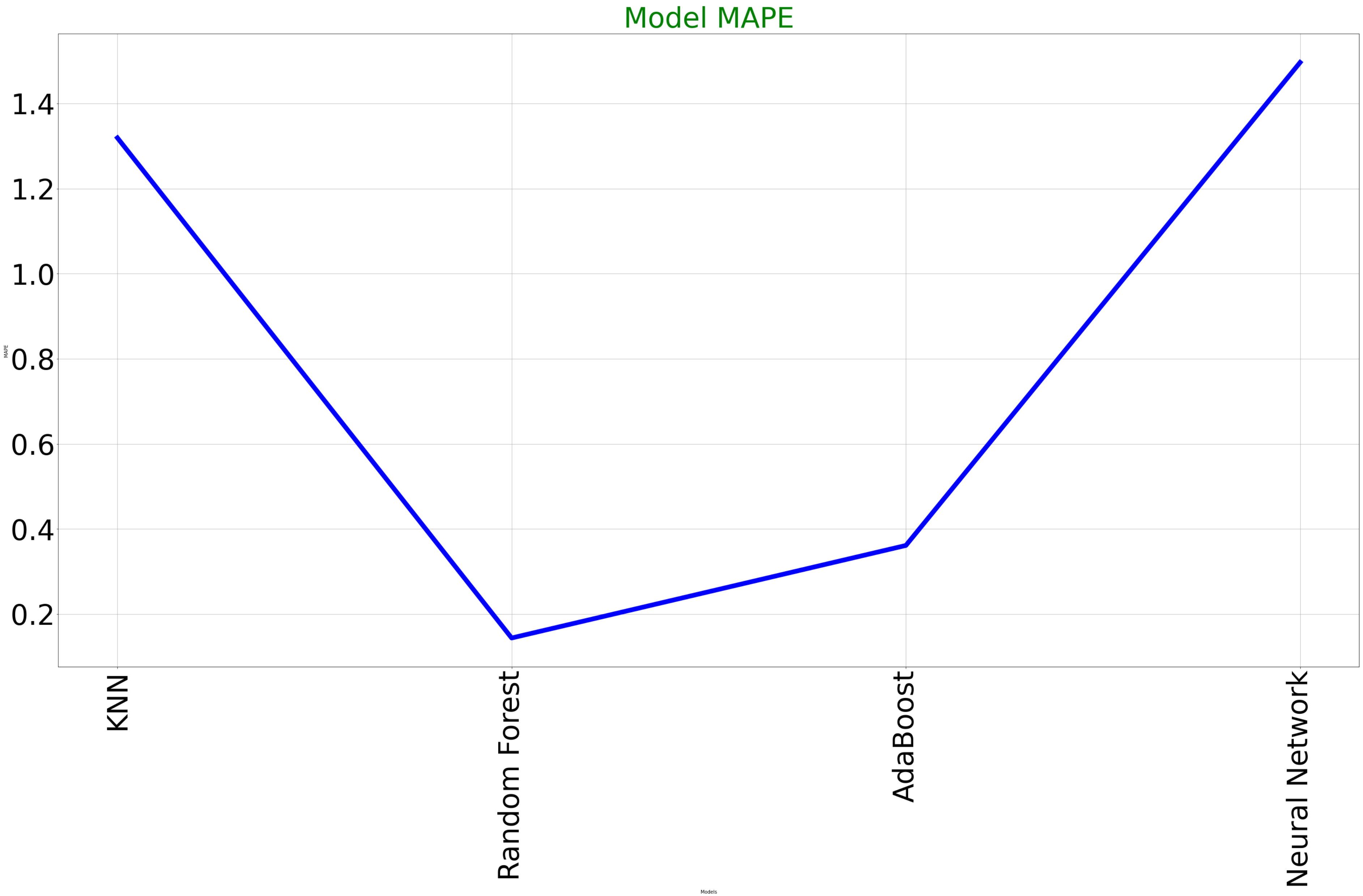
Plotting Model MAPE

In [58]:
```
plt.figure(figsize =(50, 25))
```

```
plt.plot(Compare['Models'],Compare['MAPE'],c='blue', lw=10)

plt.title('Model MAPE',fontdict={'fontsize': 60,'fontweight' : 60,'color' : 'g'})
plt.xlabel('Models')
plt.ylabel('MAPE')

plt.yticks(fontsize=60)
plt.xticks(rotation=90, fontsize=60)
plt.grid()
plt.savefig("MAPE.jpg",bbox_inches='tight')
plt.show()
```

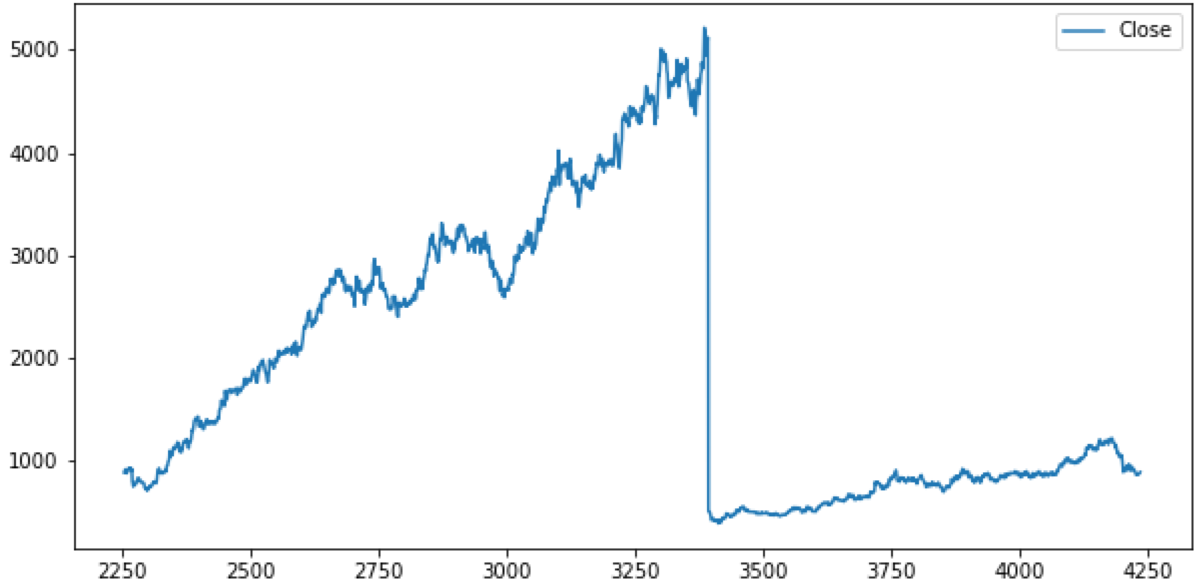


In [61]:

```
# Visualizing the dataset created for "Close" column
df_2016.plot(y = "Close",figsize=(10,5))
```

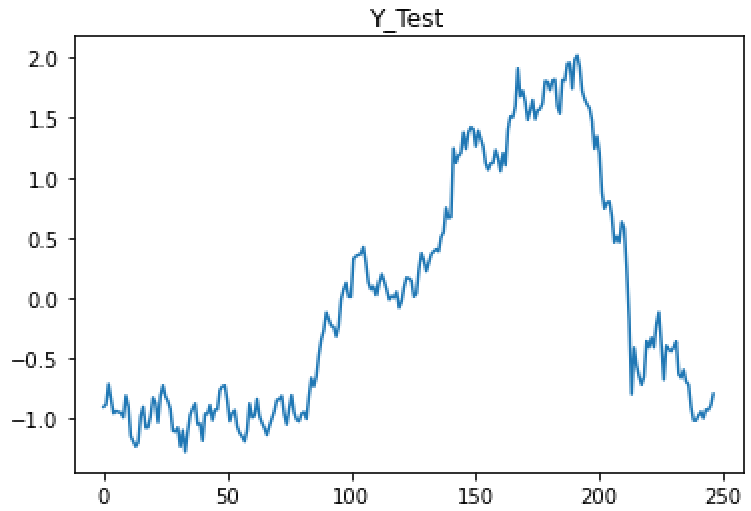
```
<AxesSubplot:>
```

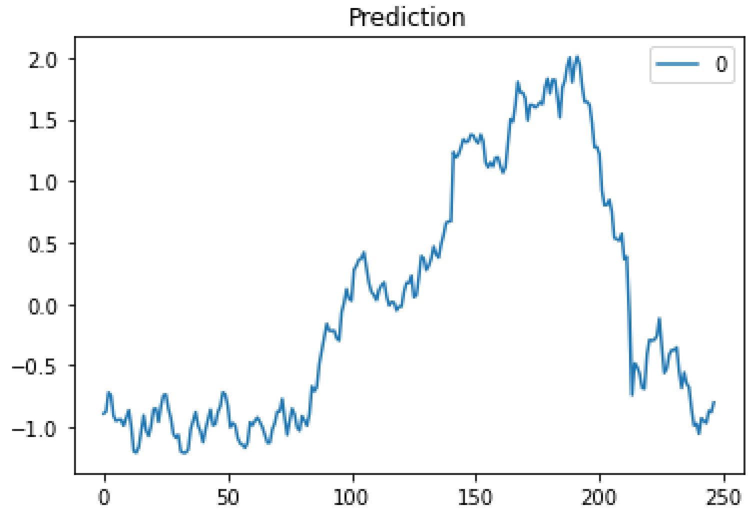
Out[61]:



In [64]:

```
# Visualizing the plot comparing between Y_test and the predicted values for 2016 year
y_pred = model_RF.predict(X_test)
y_pred = pd.DataFrame(y_pred)
Y_test.plot()
plt.title("Y_Test")
plt.savefig("Y_test.jpg",bbox_inches='tight')
y_pred.plot()
plt.title("Prediction")
plt.savefig("Prediction.jpg",bbox_inches='tight')
```


Y_Test
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250


Prediction
0
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250

```
In [ ]:
```