```
import os
from google.colab import drive
drive.mount('/content/gdrive/')
%cd /content/gdrive/MyDrive/MLproject
```

```
Mounted at /content/gdrive/
/content/gdrive/MyDrive/MLproject
```

```
pip install fastai
```

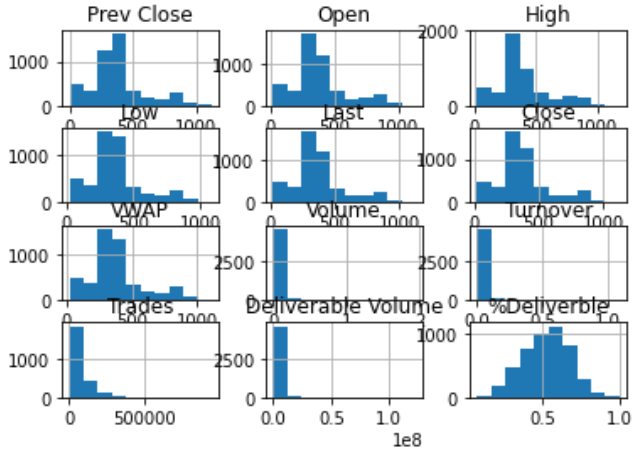
```
import csv
import fastai
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from datetime import datetime
from fastai.tabular import add_datepart
```

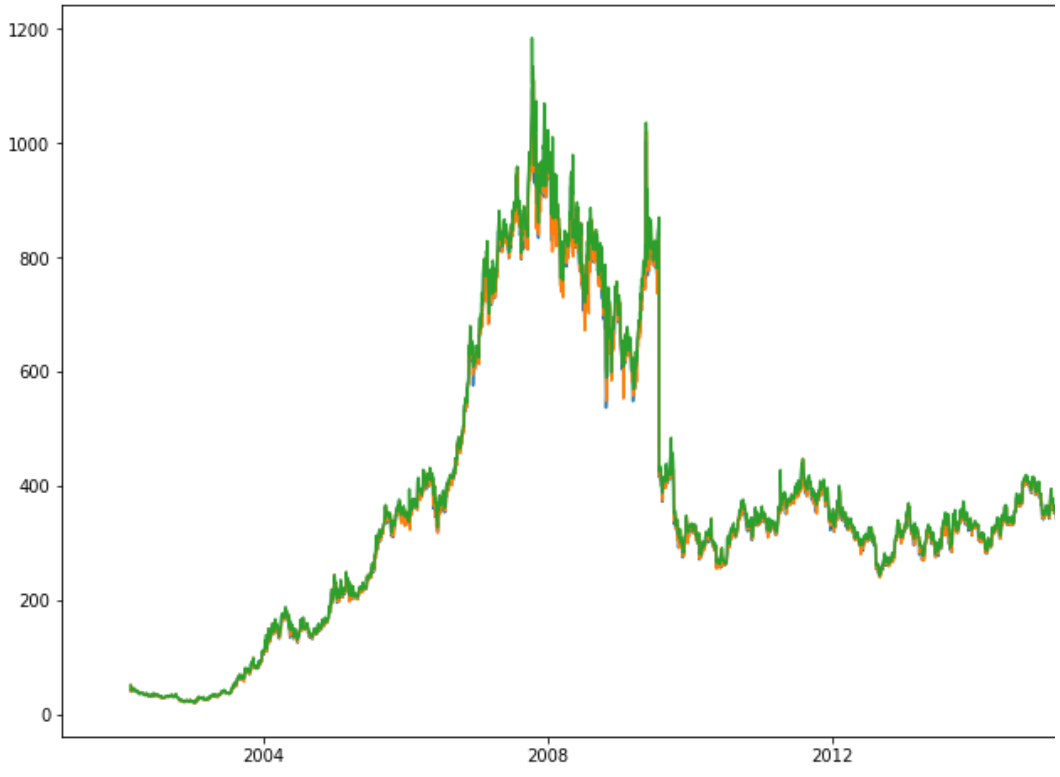
```
def alggo(data):
  print("\n-----------------------------------------\ndata of : ",data)
  #check close values is False
  df=pd.read_csv(data)
  df = pd.DataFrame(df)
  print(df.describe())
  if (df['Close'].isna().sum())!=1 and (df['High'].isna().sum())!=1:
    print("No Missing values in the Close price coloumn")

  #changing Date formate
  df['Date'] = pd.to_datetime(df.Date,format='%Y-%m-%d')
  df.index = df['Date']

  #sorting
  data = df.sort_index(ascending=True, axis=0)

  #creating a separate dataset
  new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close','Open','High','Volume'])

  new_data['Date'] = df['Date'].values
  new_data['Close'] = df['Close'].values
  new_data['Open'] = df['Open'].values
  new_data['High'] = df['High'].values
  new_data['Volume'] = df['Volume'].values

  #breaking date
  add_datepart(new_data, 'Date')
  new_data.drop('Elapsed', axis=1, inplace=True)

  #standatization
  scaler = MinMaxScaler(feature_range=(0, 1))
  scal = scaler.fit_transform(new_data)

  #normalization
  x_array = np.array(df['Close'])
  normalized_arr = preprocessing.normalize([x_array])

  #Summarization
  print("normalized Close & high column",'\n')
  print("mean: ",normalized_arr.mean())
```

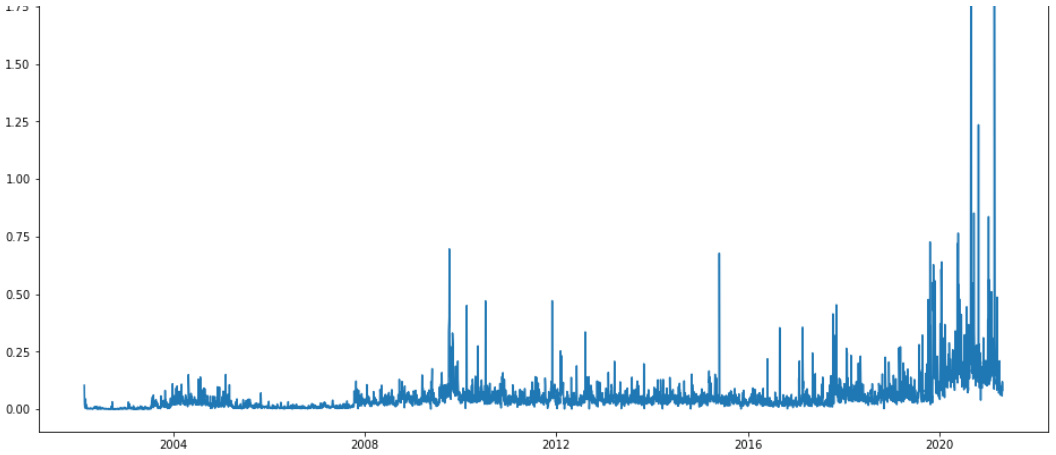
```
#Visualization
#histogram
fig = plt.figure(figsize =(20,8))
df.hist()
fig.show()

#data Close, High, Open
plt.figure(figsize=(16,8))
plt.plot(df['Date'],df['Close'], label='Close Price history')
plt.plot(df['Date'],df['Open'], label='open Price history')
plt.plot(df['Date'],df['High'], label='High Price history')
plt.legend()
plt.show()
plt.figure(figsize=(16,8))
plt.plot(df['Date'],df['Volume'], label='Volume history')
plt.legend()
plt.show()

#subplots
plt.figure(figsize=(16,9))
fig, ax = plt.subplots(figsize=(10,10))
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)

# Horizontal Bar Plot
df1 = df
df1.index=df1['Date']
df1['year'] = df1.index.year
fig = plt.figure(figsize =(20,8))
plt.bar(df1['year'],df1['Close'])
plt.show()


#data split 70:30
n = len(new_data)
sev = int(0.7 * n)
train = new_data[:sev]
test = new_data[sev:]

pred = ['Close','High']
x_train = train.drop(pred, axis=1)
y_train = train[pred]
x_test = test.drop(pred, axis=1)
y_test = test[pred]


#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)

#prediction
preds = model.predict(x_test)
print('Variance score: %.2f' % model.score(x_test, y_test))
rms=np.sqrt(np.mean(np.power((np.array(y_test)-np.array(preds)),2)))
print("rms : %.2f" %rms)

#vizulization predictions Vs actual

test['Predictions close'] = [x[0] for x in preds]
test['Predictions high'] = [x[1] for x in preds]

test.index = new_data[sev:].index
train.index = new_data[:sev].index

x = np.array(y_test['High'])
```

```
  y = np.array(test['Predictions high'])
  plt.scatter(x, y)
  plt.xlabel("Actual Price High")
  plt.ylabel("Predicted Price High")
  plt.show()

  x = np.array(y_test['Close'])
  y = np.array(test['Predictions close'])
  plt.scatter(x, y)
  plt.xlabel("Actual Price close")
  plt.ylabel("Predicted Price close")
  plt.show()

  fig = plt.figure(figsize =(20,8))

  plt.plot(test['Close'],label = 'actual')
  plt.plot(test['Predictions close'],label = 'prediction')
  plt.xlabel('Days')
  plt.ylabel('Price')
  plt.title('prediction for close price')
  plt.legend()
  plt.show()

  fig = plt.figure(figsize =(20,8))
  plt.plot(test['High'], label = 'actual')
  plt.plot(test['Predictions high'], label = 'prediction')
  plt.xlabel('Days')
  plt.ylabel('Price')
  plt.title('prediction for High price')
  plt.legend()
  plt.show()

  #accuracy of close and high
  predsss = ['Predictions close', 'Predictions high' ]
  y2 = test[predsss].mean()
  x2 = test[pred].mean()
  print('Close accarucay: %.2f' %  (x2[0]/y2[0] * 100))
  print('High accarucay: %.2f' % (x2[1]/y2[1] * 100))


if __name__ == '__main__':
  filename=['BHARTIARTL.csv','CIPLA.csv','DRREDDY.csv']
  n = int(input("--------\n1. BHARTIARTL.csv\n2. CIPLA.csv\n3. DRREDDY's.csv\nInput no of CSV file to get pr
  alggo(filename[n-1])
```

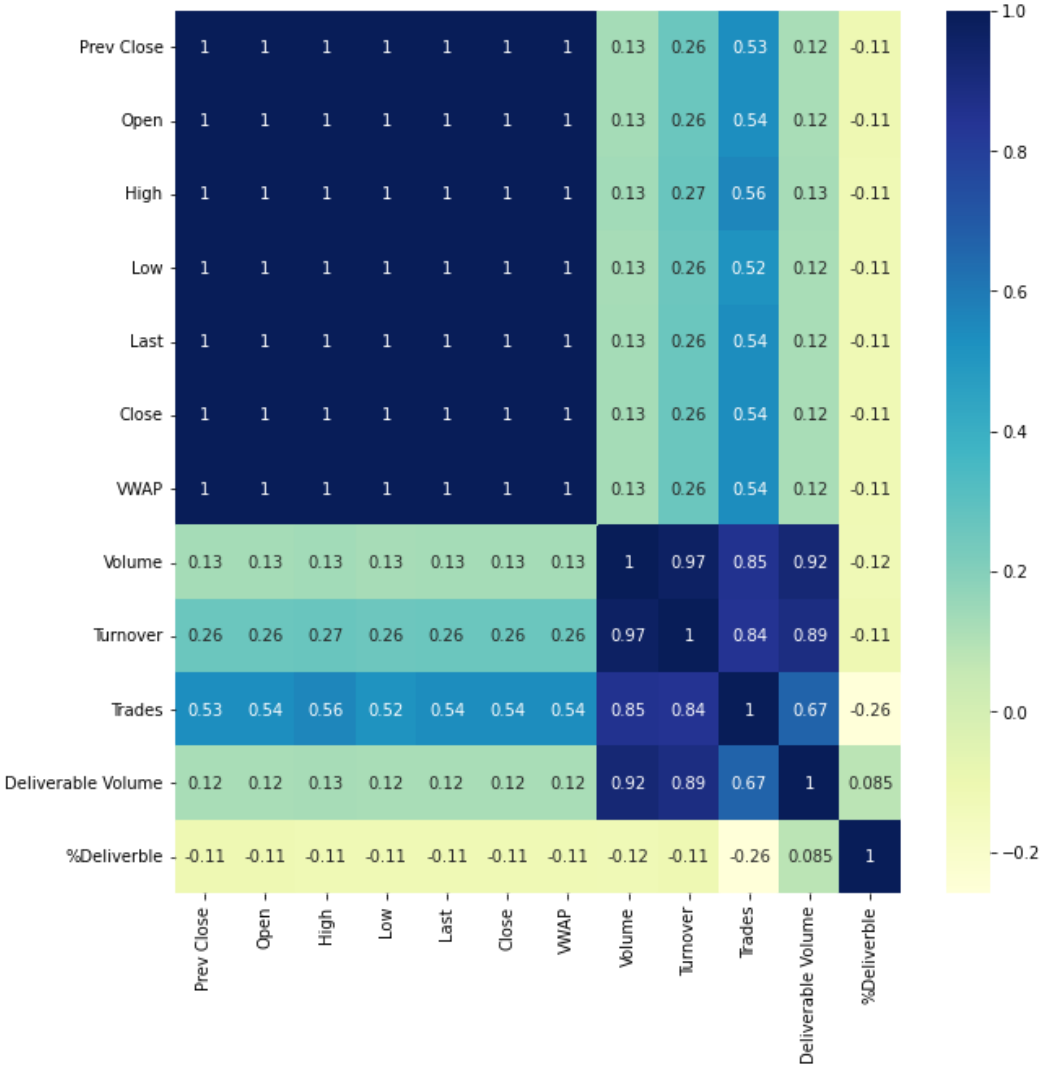
```
--------
1. BHARTIARTL.csv
2. CIPLA.csv
3. DRREDDY's.csv
Input no of CSV file to get prediction: 1

-----------------------------------------------
data of :  BHARTIARTL.csv
        Prev Close         Open  ...  Deliverable Volume  %Deliverble
count  4774.000000  4774.000000  ...        4.758000e+03  4758.000000
mean    379.688333   380.478456  ...        2.653730e+06     0.521711
std     207.237329   207.774041  ...        4.016530e+06     0.148377
min       0.000000    21.100000  ...        1.830100e+04     0.071900
25%     301.325000   302.000000  ...        8.125398e+05     0.417850
50%     348.800000   349.000000  ...        1.793994e+06     0.530000
75%     423.850000   425.000000  ...        3.254393e+06     0.628475
max    1125.650000  1133.900000  ...        1.229199e+08     0.999800

[8 rows x 12 columns]
No Missing values in the Close price coloumn
normalized Close & high column

mean:  0.012705914068712462
/usr/local/lib/python3.7/dist-packages/fastai/tabular/transform.py:63: FutureWarni
  for n in attr: df[prefix + n] = getattr(field.dt, n.lower())
<Figure size 1440x576 with 0 Axes>
```

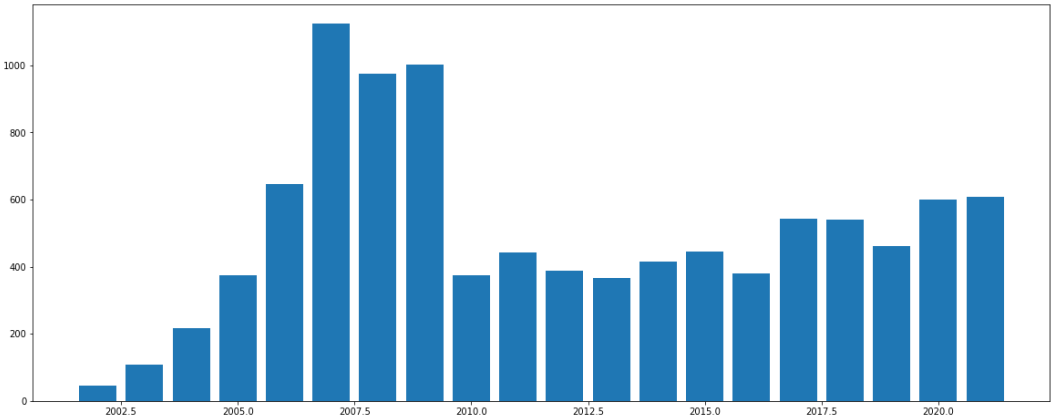






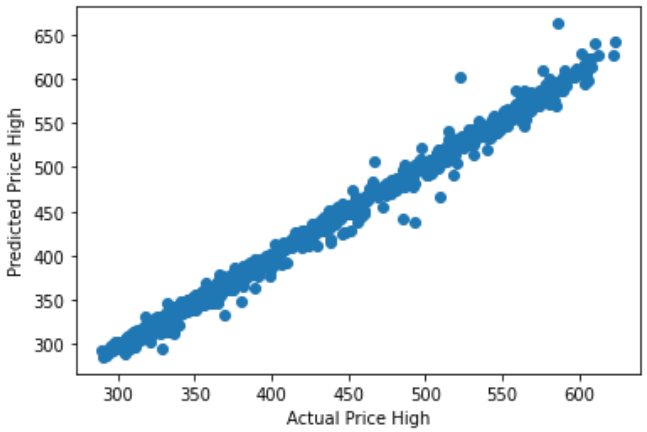
<Figure size 1152x648 with 0 Axes>

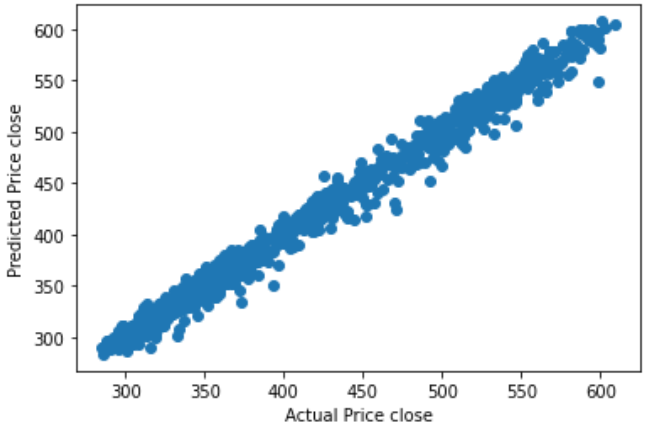


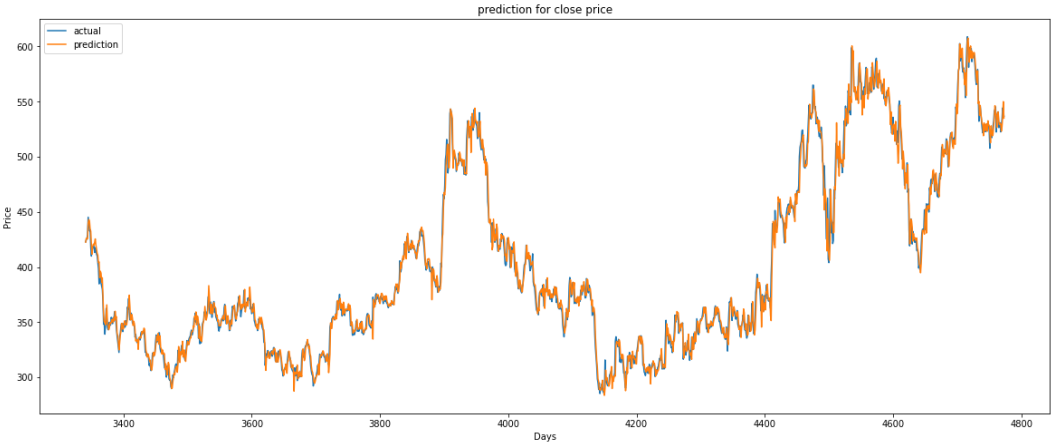


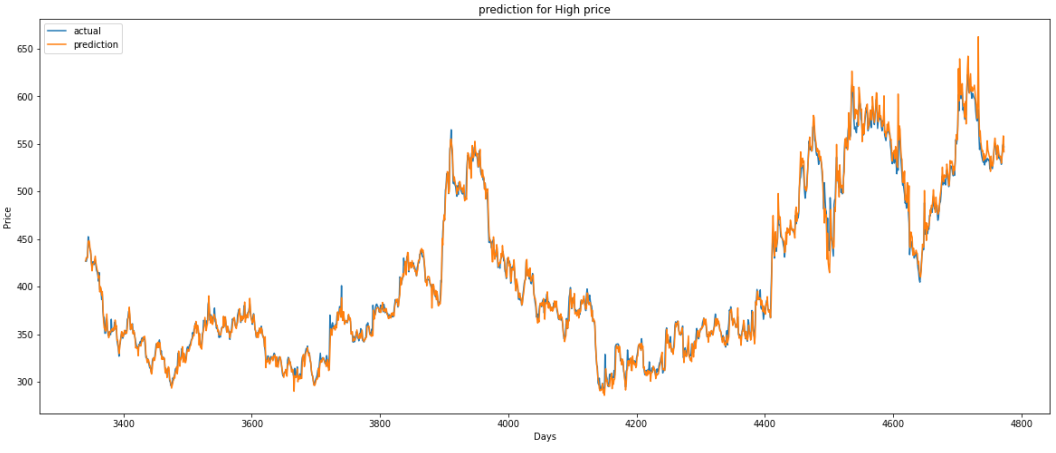
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:434: FutureWarning: The def
"multioutput='uniform_average').", FutureWarning)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:100: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:101: SettingWithCopyW
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
Variance score: 0.99
rms : 8.02
```










```
Close accaracy: 99.97
```

```
Close accuracay: 99.97
High accarucay: 99.94
```