Time series analysis of stock market data

Aim: to visualize stock market data using time series analysis.

Description:

Time series analysis is a statistical technique that focuses on analyzing data points collected sequentially over time to identify patterns, trends, and seasonality. In the context of stock prediction, time series analysis seeks to capture the inherent dynamics of stock market data and uncover meaningful patterns that can inform future price movements.

Key concepts in time series analysis include:

1. Trend: The long-term movement or direction of a stock's price over time. Trends can be upward (bullish), downward (bearish), or sideways (range-bound).
2. Seasonality: Regular patterns or cycles that repeat at fixed intervals within the data, such as daily, weekly, monthly, or yearly.
3. Autocorrelation: The correlation between a data point and its lagged values. Autocorrelation allows us to understand how past values influence future values.
4. Stationarity: A desirable property of time series data where statistical properties, such as mean and variance, remain constant over time. Stationary data simplifies the analysis and enhances prediction accuracy.

Time series analysis is a powerful tool for stock prediction, offering investors and traders valuable insights into the complex dynamics of the stock market. By leveraging historical data, capturing patterns, and applying statistical techniques, time series analysis allows for accurate forecasting of stock prices, identification of trends and seasonality, and evaluation of risk.

With its diverse applications, including trend analysis, volatility forecasting, trading strategy development, portfolio optimization, and event impact assessment, time series analysis equips market participants with the tools needed to make informed investment decisions in the dynamic world of stocks. Incorporating time series analysis into investment strategies can enhance the probability of success and provide a competitive edge in the quest for financial prosperity.

Stock Market Analysis and Time Series Prediction

Steps:

(Code to be executed given in BOLD, outputs are also given.) You can have your own analysis and visualization by changing the code.)

1. Importing Libraries
2. Preprocessing
3. Tesla Stock Market Analyis

4. Tesla ARIMA (AutoRegressive Integrated Moving Average) Time Series Prediction

Importing Libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from subprocess import check_output
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from pandas.plotting import lag_plot
from pandas import datetime
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

warnings.filterwarnings('ignore')
```

Preprocessing

```
# This Python3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

# Input data files are available in the "../input/" directory.
# For example, running this will list the files in the input directory

print(os.listdir("../input"))

print(check_output(["ls", "../input/Data"]).decode("utf8"))

# Any results you write to the current directory are saved as output.

['Data']
ETFs
Stocks


#print(check_output(["ls", "../input/Data/Stocks"]).decode("utf8"))
```

Tesla Stock Market Analyis

```
df = pd.read_csv("../input/Data/Stocks/tsla.us.txt")
df.head()
```

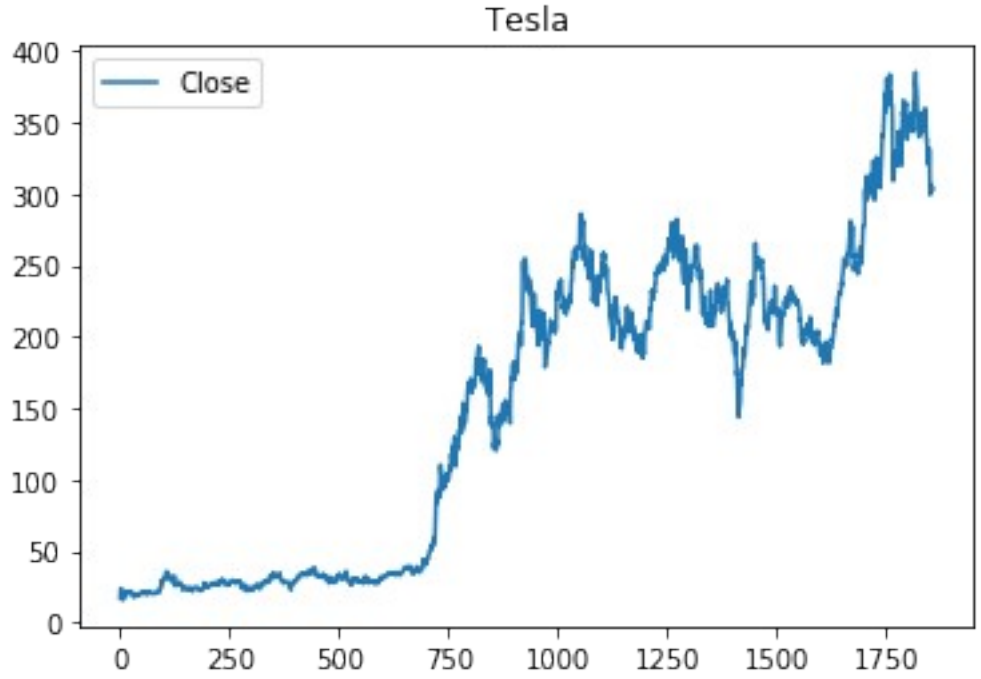
	Date	Open	High	Low	Close	Volume	OpenInt
0	2010-06-28	17.00	17.00	17.00	17.00	0	0
1	2010-06-29	19.00	25.00	17.54	23.89	18783276	0
2	2010-06-30	25.79	30.42	23.30	23.83	17194394	0
3	2010-07-01	25.00	25.92	20.27	21.96	8229863	0

	Date	Open	High	Low	Close	Volume	OpenInt
4	2010-07-02	23.00	23.10	18.71	19.20	5141807	0

```
print(df.head())
print(df.shape)
print(df.columns)
```

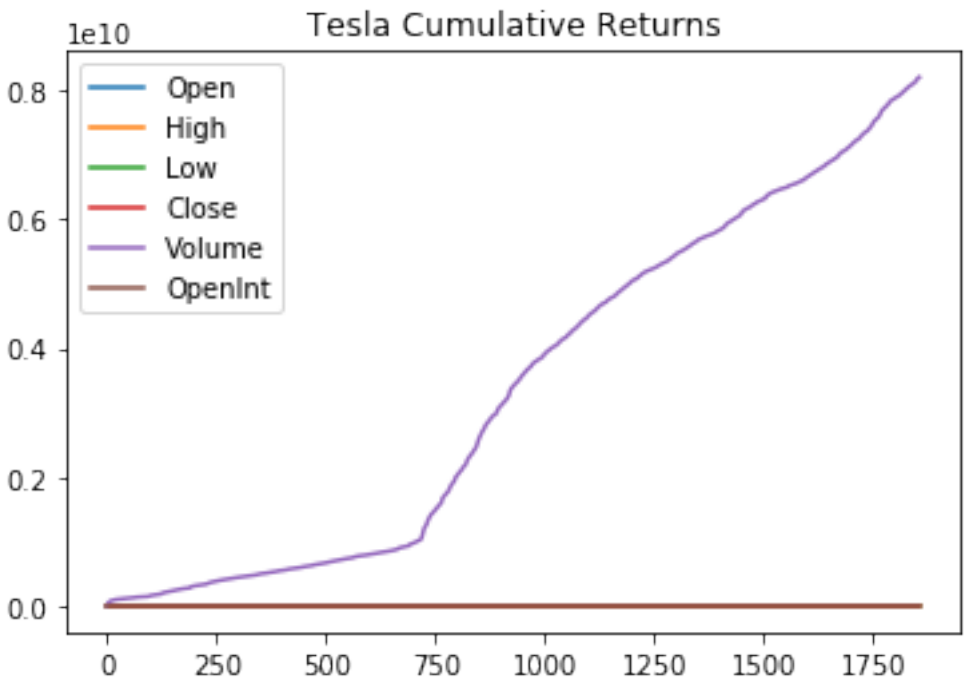
```
         Date   Open   High    Low  Close    Volume  OpenInt
0  2010-06-28  17.00  17.00  17.00  17.00         0        0
1  2010-06-29  19.00  25.00  17.54  23.89  18783276        0
2  2010-06-30  25.79  30.42  23.30  23.83  17194394        0
3  2010-07-01  25.00  25.92  20.27  21.96   8229863        0
4  2010-07-02  23.00  23.10  18.71  19.20   5141807        0
(1858, 7)
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'OpenInt'],
dtype='object')
```

```
df[['Close']].plot()
plt.title("Tesla")
plt.show()
```



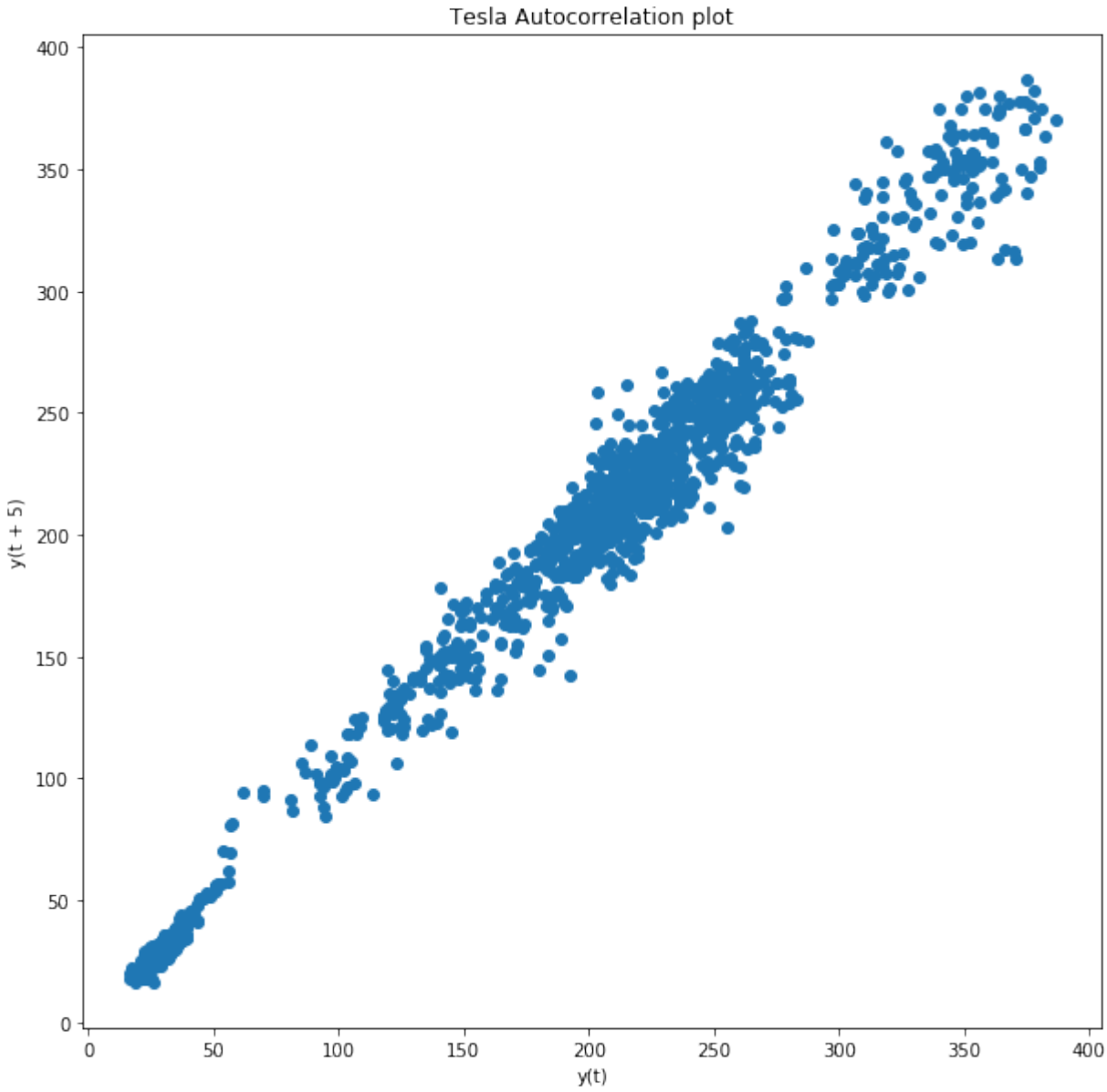

```
# Comulative Return
dr = df.cumsum()
dr.plot()
plt.title('Tesla Cumulative Returns')
```

```
Text(0.5, 1.0, 'Tesla Cumulative Returns')
```




```
plt.figure(figsize=(10,10))
lag_plot(df['Open'], lag=5)
plt.title('Tesla Autocorrelation plot')
```

```
Text(0.5, 1.0, 'Tesla Autocorrelation plot')
```



ARIMA (AutoRegressive Integrated Moving Average) for Time Series Prediction

```
df['Date'][1857]
```

Output:

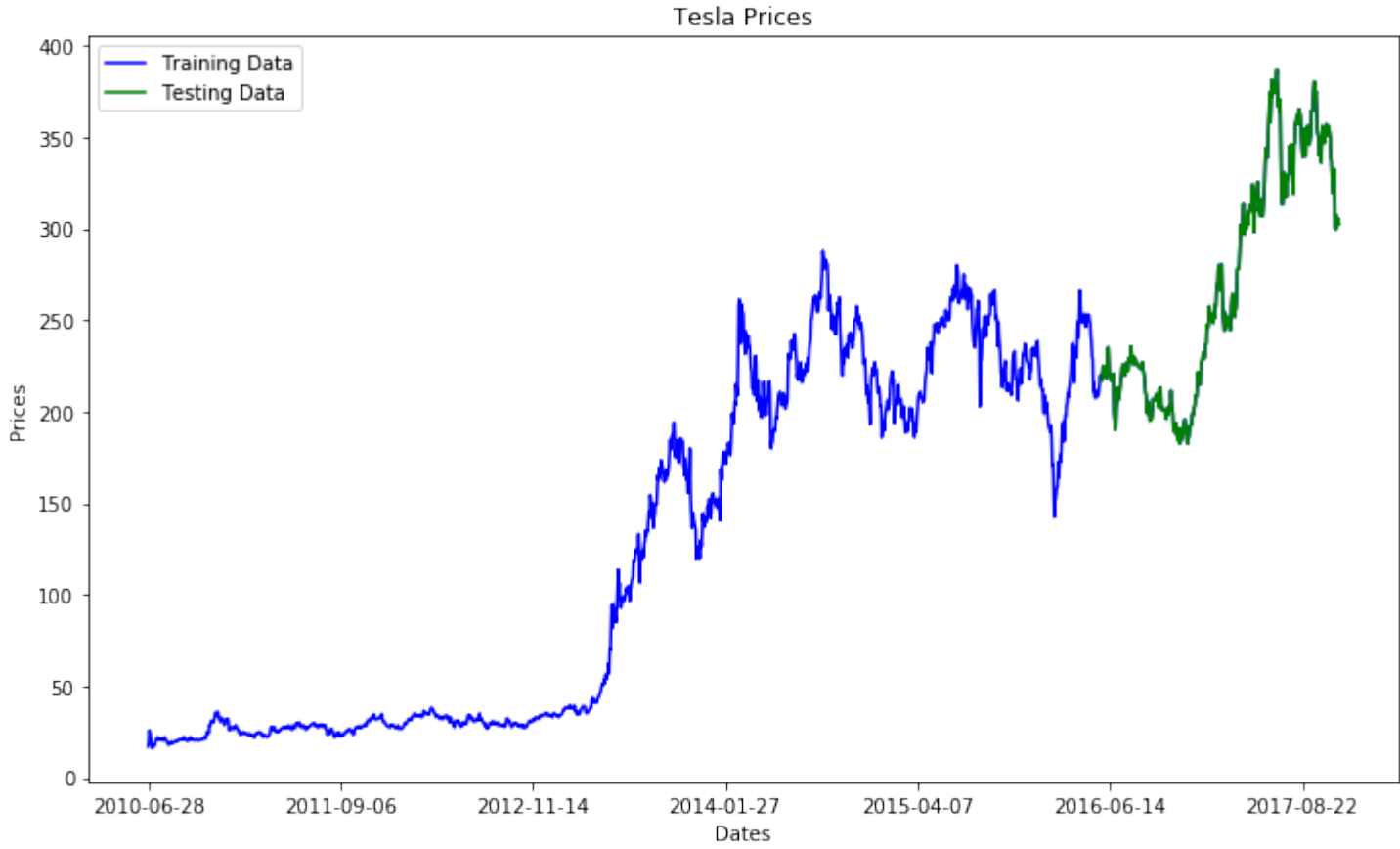
```
'2017-11-10'
```

```
train_data, test_data = df[0:int(len(df)*0.8)], df[int(len(df)*0.8):]
plt.figure(figsize=(12,7))
plt.title('Tesla Prices')
plt.xlabel('Dates')
plt.ylabel('Prices')
```

```
plt.plot(df['Open'], 'blue', label='Training Data')
plt.plot(test_data['Open'], 'green', label='Testing Data')
plt.xticks(np.arange(0,1857, 300), df['Date'][0:1857:300])
plt.legend()
```

output:

```
<matplotlib.legend.Legend at 0x7f4d054e5d68>
```




```
def smape_kun(y_true, y_pred):
    return np.mean((np.abs(y_pred - y_true) * 200/ (np.abs(y_pred) +
np.abs(y_true))))
```

```
train_ar = train_data['Open'].values
test_ar = test_data['Open'].values
```

```
history = [x for x in train_ar]
print(type(history))
predictions = list()
for t in range(len(test_ar)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test_ar[t]
    history.append(obs)
    #print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_squared_error(test_ar, predictions)
```

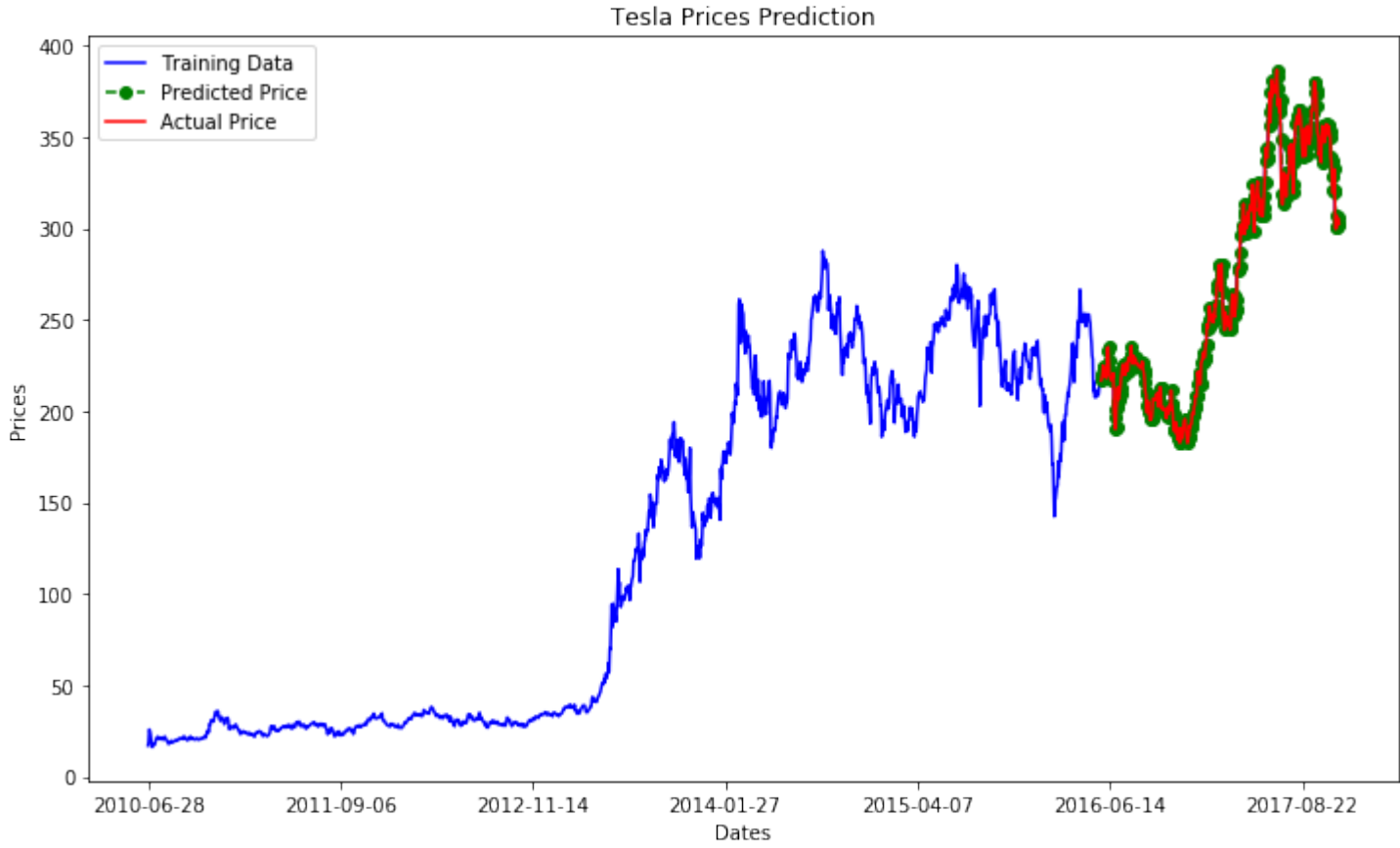
```
print('Testing Mean Squared Error: %.3f' % error)
error2 = smape_kun(test_ar, predictions)
print('Symmetric mean absolute percentage error: %.3f' % error2)
```

```
<class 'list'>
Testing Mean Squared Error: 40.550
Symmetric mean absolute percentage error: 26.015
```

```
plt.figure(figsize=(12,7))
plt.plot(df['Open'], 'green', color='blue', label='Training Data')
plt.plot(test_data.index, predictions, color='green', marker='o',
linestyle='dashed',
         label='Predicted Price')
plt.plot(test_data.index, test_data['Open'], color='red', label='Actual Price')
plt.title('Tesla Prices Prediction')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.xticks(np.arange(0,1857, 300), df['Date'][0:1857:300])
plt.legend()
```

ouput:

```
<matplotlib.legend.Legend at 0x7f4d05471f28>
```

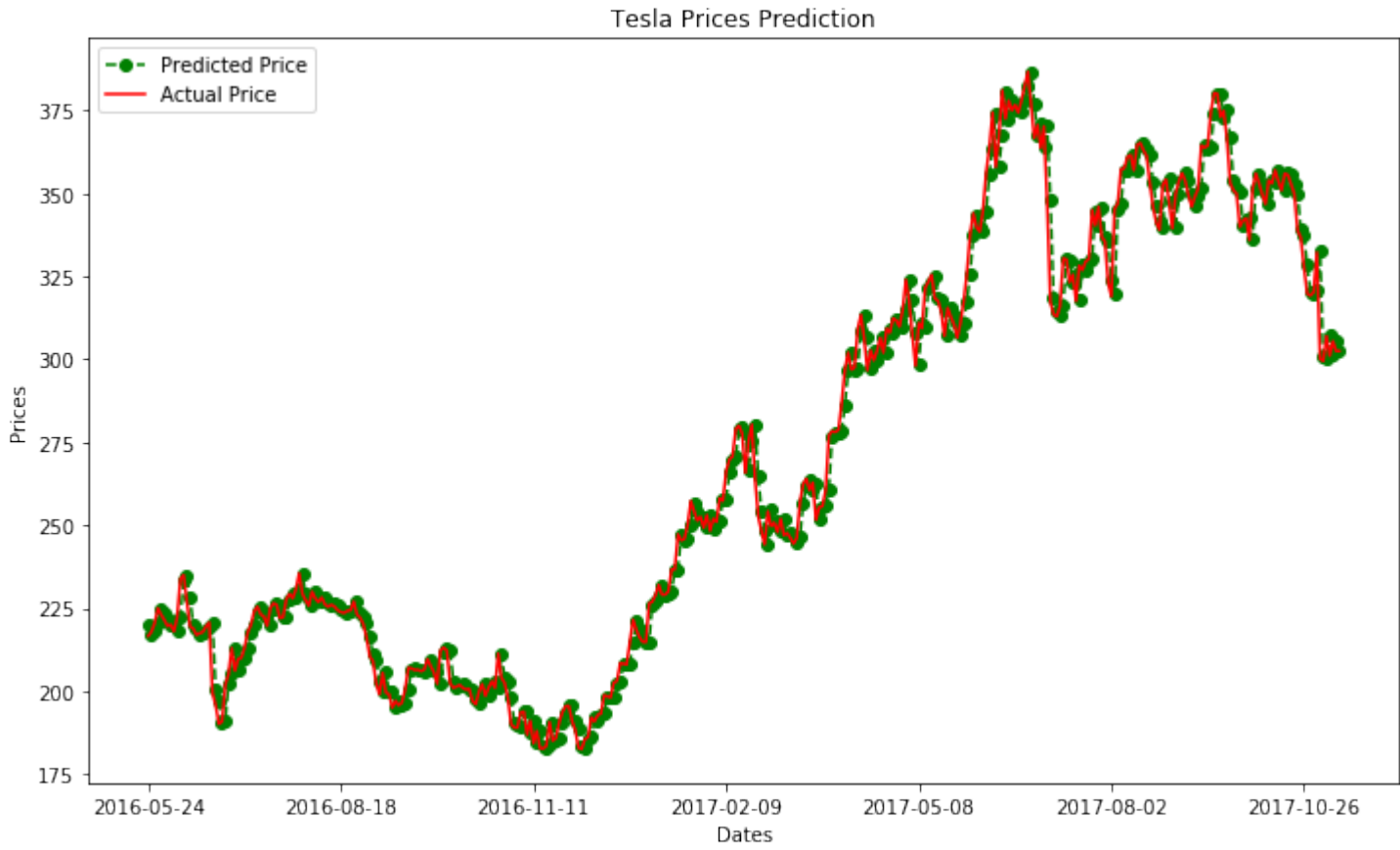


```
#test_data['Date'][0:1856:10]
```

```
plt.figure(figsize=(12,7))
plt.plot(test_data.index, predictions, color='green', marker='o',
linestyle='dashed',
         label='Predicted Price')
plt.plot(test_data.index, test_data['Open'], color='red', label='Actual Price')
plt.xticks(np.arange(1486,1856, 60), df['Date'][1486:1856:60])
plt.title('Tesla Prices Prediction')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
```

```
output:

<matplotlib.legend.Legend at 0x7f4d08ecc6a0>
```



For More learning see the link given below:

Microsoft Stock Market Analyis

Microsoft ARIMA (AutoRegressive Integrated Moving Average) Time Series Prediction

https://www.kaggle.com/code/pierpaolo28/stock-market-analysis-and-time-series-prediction

Note:

Every student need to learn above code and visalizations in today afternoon lab after BDA lab.

Submit the record in print form on A4 sheets. With color prints if you can afford.

Get your record corrected in this week and bring for internal lab exam.

Use stick files for making your record.