Project Name: Web Scraping Stocks

Project Owner: Omer Orhan

Instructor: Dr. Ravishankar Chityala

Course Name: CMPR.X416.(19) Python for Programmers

Index

Introduction

My starting point is Project Suggestions on Canvas. I chose Web Scraping with Yahoo.

My project gets APPLE stock prices and dates from finance.yahoo.com.

The project includes five main functions;

- getdatafromSource
- drawplot
- createCsvFile
- makeprediction
- valid_date

After getting prices and dates, The prices/dates plot can be seen within matplotlib.pyplot.

A CSV file is created to write price and date information. Due to learning future stock price, linear regression is implemented into the project within numpy and sklearn. A user is able to enter a specific date to predict price, and the project validates the date as well within Regex and try-except expressions.

Requirements

The project uses many modules listed below.

```
from bs4 import BeautifulSoup
from lxml import html
import numpy as np
from sklearn.linear_model import LinearRegression
import urllib.request
import requests
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
from datetime import datetime
import re
import matplotlib.pyplot as plt
```

In order to use these modules, the user has to download packages with pip install commands.

These are;

```
# pip install requests
# pip install BeautifulSoup4
# pip install lxml
# pip install numpy
# pip install sklearn
# pip install matplotlib
```

Description of the Python program

Modules are used;

```
from bs4 import BeautifulSoup
from lxml import html
import numpy as np
```

```
from sklearn.linear_model import LinearRegression
import urllib.request
import requests
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
from datetime import datetime
import re
import matplotlib.pyplot as plt
```

All the functions are created in Webscraping class.

Functions;

1. **getdatafromSource:**
   The function gets Apple stock prices from Yahoo history page.
   The web pages structure is;
   <tr class="BdT Bdc($c-fuji-grey-c) Ta(end) Fz(s) Whs(nw)" data-reactid="51">
   <td class="Py(10px) Ta(start) Pend(10px)" data-reactid="52">Dec 06, 2018</td>
   <td class="Py(10px) Pstart(10px)" data-reactid="54">171.76</td>
   <td class="Py(10px) Pstart(10px)" data-reactid="56">174.78</td>

   I used span and data-reactid to get price and date value. For getting a price, there is occurrence by 15 in the HTML resource code, so I increase the counter variable by 15 to get next price values. Dates are the same as prices. I limit the values with twenty prices and dates.

```
def getdatafromSource(self):
    page = urllib.request.urlopen(self.link)
    soup = BeautifulSoup(page, 'html.parser')
    price_box = soup.find('span', attrs={'data-reactid': 53})
    pricelist = []
    datelist = []
    datelistfloat = []
    pricecounter = 53
    datecounter = 51

    while (True):
        price_box = soup.find('span', attrs={'data-reactid': pricecounter})
        date_box = soup.find('span', attrs={'data-reactid': datecounter})
        if price_box == None:
            pricecounter = pricecounter + 15
            datecounter = datecounter + 15
            continue
        pricelist.append(float(price_box.string))
        date = datetime.strptime(date_box.string, '%b %d, %Y')
        datelistfloat.append(datetime.fromisoformat(str(date)).timestamp())
        datelist.append(str(date.month) + "-" + str(date.day))
        pricecounter = pricecounter + 15
        datecounter = datecounter + 15
        #get first 20 prices
        if pricecounter > 300:
            break
    return pricelist, datelist, datelistfloat
```

2. **drawplot**
   I draw plot to see prices movements. In that case, matplotlib.pyplot is used to draw plot. Prices and dates is ordered newest to oldest in the HTML. Due to list prices from oldest to newest, I use the reverse function.

```
def drawplot(self, pricelist, datelist):
    pricelist.reverse()
    datelist.reverse()
    plt.plot(datelist, pricelist)
    plt.show()
```

3. **createCsvFile**
   All prices and dates are written in csv file.

```
def createCsvFile(self, csvfilename, pricelist, datelist):
    dic = dict(zip(datelist, pricelist))
    with open(csvfilename, 'w') as csv_file:
        columntitlerow = "date,price\n"
        csv_file.write(columntitlerow)
        [csv_file.write(key + "," + str(value) + "\n") for key, value in
dic.items()]
```

4. **makeprediction**

   I use sklearn linear regression model with prices and dates. Dates are converted timestamp format to use for linear regression. After creating linear model, predict function is used to make a prediction with a specific date. I also validate date format and use raise exception. NotValidDateFormat is created to raise specific exception.

```
def makeprediction(self, pricelist, datelist, newdate):
    if not self.valid_date(newdate):
        raise NotValidDateFormat
    datelist = np.array(datelist).reshape((len(datelist), 1))
    reg = LinearRegression().fit(datelist, pricelist)
    newdate = datetime.strptime(newdate, '%m/%d/%Y')
    newdatetimestamp = datetime.fromisoformat(str(newdate)).timestamp()
    newdata = [(newdatetimestamp)]
    newdata = np.array(newdata).reshape((len(newdata), 1))
    print(str(newdate) + "predicted price =" + str(reg.predict(newdata)))
```

5. **valid_date**

   Regex contols whether date's format is accaptable or not.

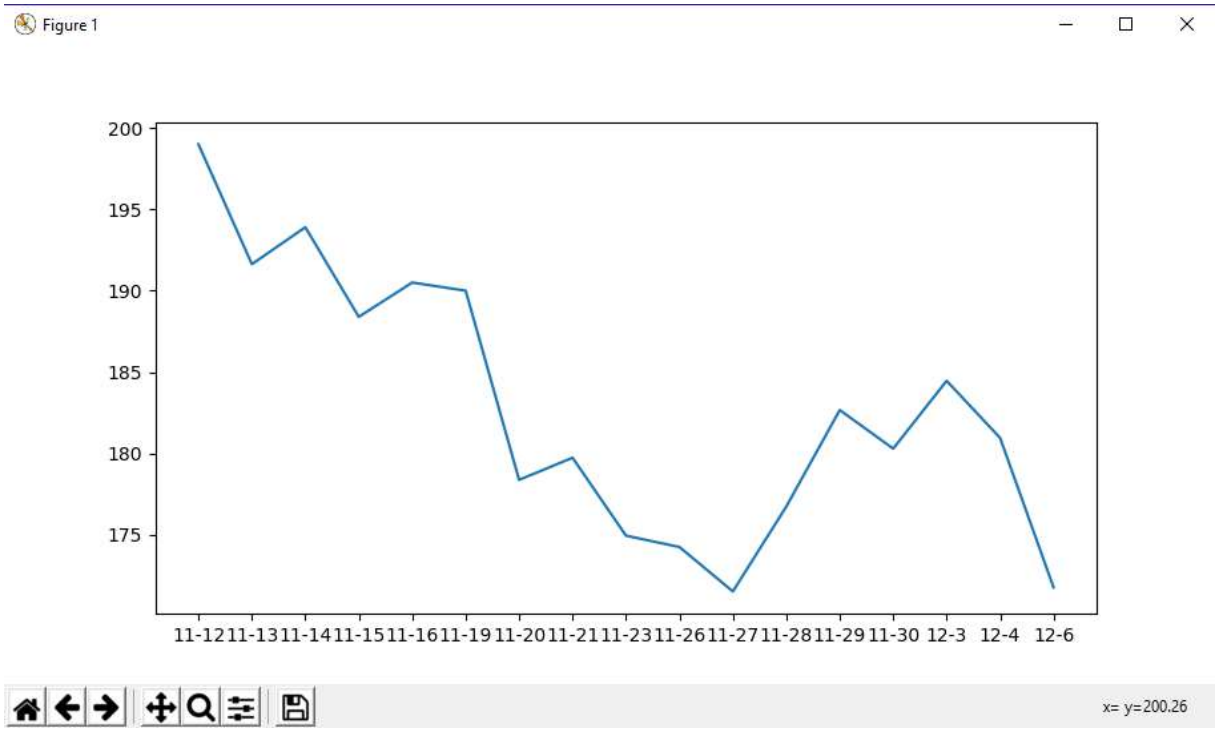
```
def valid_date(selstrptimef, datestring):
    mat = re.match('(\d{2})[/.-](\d{2})[/.-](\d{4})$', datestring)
    if mat is not None:
        return True
    else:
        return False
```

6. **Inputs**

```
7.  web = Webscraping('https://finance.yahoo.com/quote/AAPL/history?p=AAPL')
    historyList = web.getdatafromSource();
    web.drawplot(historyList[0], historyList[1])
    web.createCsvFile("C:/Users/omer/Downloads/stocks.csv", historyList[0],
    historyList[1])
    web.makeprediction(historyList[0], historyList[2], '12/30/2018')
```

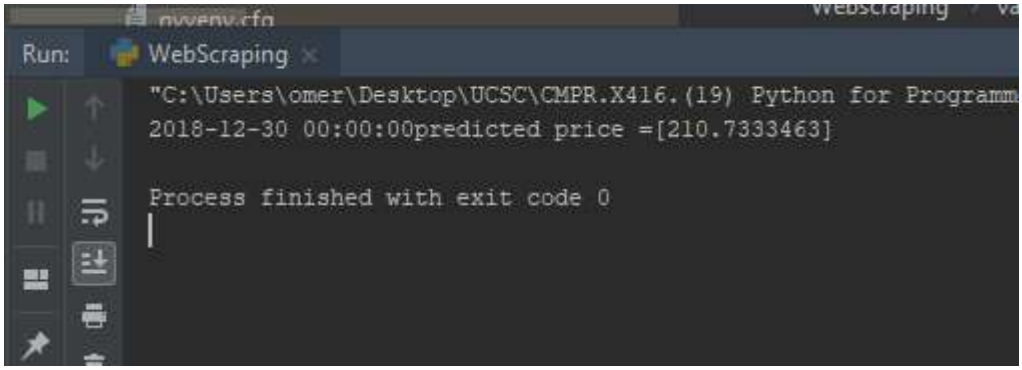
Screenshots of the program output

The output of Plot;



Prediction of Result;

When the date shows 2018-12-30, the apple stock price prediction is 210.73



Conclusion

I have learned how to combine python elements. I used these elements in my project;

1. Use any data structure like list, dictionary, set or tuple
2. Dictionary comprehension
3. Functions
4. Classes
5. Importing external modules
6. Error checks using try-except
7. File input and output
8. Regular expression

My solution shows how to get data from web resources and analyze the data for predictions.

Python program

```
from bs4 import BeautifulSoup
from lxml import html
import numpy as np
from sklearn.linear_model import LinearRegression
import urllib.request
import requests
import plotly.plotly as py
import plotly.graph_objs as go
import plotly.tools as tls
from datetime import datetime
import re
import matplotlib.pyplot as plt

# pip install requests
# pip install BeautifulSoup4
# pip install lxml
# pip install numpy
# pip install sklearn
# pip install matplotlib

class NotValidDateFormat(Exception):
    pass

class Webscraping():

    def __init__(self, link):
        self.link = link

    def getdatafromSource(self):
        page = urllib.request.urlopen(self.link)
```

```
        soup = BeautifulSoup(page, 'html.parser')
        price_box = soup.find('span', attrs={'data-reactid': 53})
        pricelist = []
        datelist = []
        datelistfloat = []
        pricecounter = 53
        datecounter = 51

        while (True):
            price_box = soup.find('span', attrs={'data-reactid': pricecounter})
            date_box = soup.find('span', attrs={'data-reactid': datecounter})
            if price_box == None:
                pricecounter = pricecounter + 15
                datecounter = datecounter + 15
                continue
            pricelist.append(float(price_box.string))
            date = datetime.strptime(date_box.string, '%b %d, %Y')
            datelistfloat.append(datetime.fromisoformat(str(date)).timestamp())
            datelist.append(str(date.month) + "-" + str(date.day))
            pricecounter = pricecounter + 15
            datecounter = datecounter + 15
            #get first 20 prices
            if pricecounter > 300:
                break
        return pricelist, datelist, datelistfloat

    def drawplot(self, pricelist, datelist):
        pricelist.reverse()
        datelist.reverse()
        plt.plot(datelist, pricelist)
        plt.show()

    def createCsvFile(self, csvfilename, pricelist, datelist):
        dic = dict(zip(datelist, pricelist))
        with open(csvfilename, 'w') as csv_file:
            columntitlerow = "date,price\n"
            csv_file.write(columntitlerow)
            [csv_file.write(key + "," + str(value) + "\n") for key, value in
dic.items()]

    def makeprediction(self, pricelist, datelist, newdate):
        if not self.valid_date(newdate):
            raise NotValidDateFormat
        datelist = np.array(datelist).reshape((len(datelist), 1))
        reg = LinearRegression().fit(datelist, pricelist)
        newdate = datetime.strptime(newdate, '%m/%d/%Y')
        newdatetimestamp = datetime.fromisoformat(str(newdate)).timestamp()
        newdata = [(newdatetimestamp)]
        newdata = np.array(newdata).reshape((len(newdata), 1))
        print(str(newdate) + "predicted price =" + str(reg.predict(newdata)))

    def valid_date(selstrptimef, datestring):
        mat = re.match('(\d{2})[/.-](\d{2})[/.-](\d{4})$', datestring)
        if mat is not None:
            return True
        else:
            return False

web = Webscraping('https://finance.yahoo.com/quote/AAPL/history?p=AAPL')
historyList = web.getdatafromSource();
web.drawplot(historyList[0], historyList[1])
web.createCsvFile("C:/Users/omer/Downloads/stocks.csv", historyList[0],
historyList[1])
web.makeprediction(historyList[0], historyList[2], '12/30/2018')
```