Fall 2021

Project 1

Python Programming

Name: Ravleen Kaur

Class ID: 43

School ID: 1113138

Course: Statistics for Data Science

Course ID: DTSC-620-W01

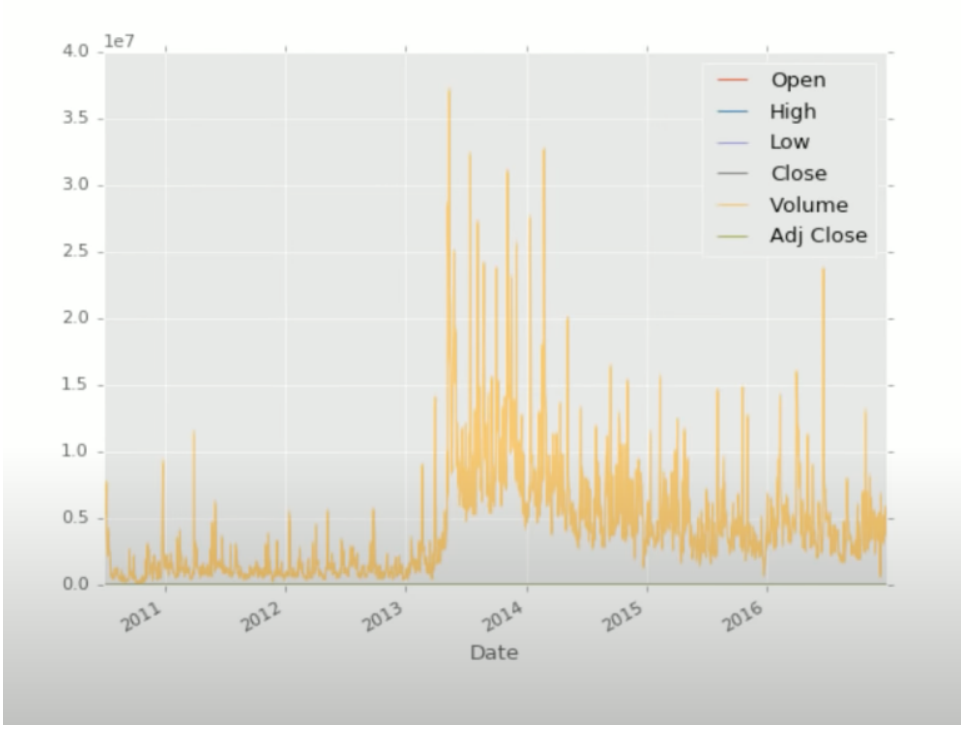
Part 1: basic data handling with the Pandas module and data visualization with Matplotlib

```
      Open        High         Low       Close   Volume  \
208.220001  209.990005  206.500000  208.449997  3106900
208.000000  213.449997  207.710007  213.339996  4662900
214.880005  222.250000  214.419998  219.529999  5901400
221.529999  223.800003  217.199997  219.740005  3766900
218.559998  219.199997  214.119995  214.679993  4035900
216.300003  217.500000  211.679993  213.690002  4632700
```

```
1  import datetime as ft
2  import matplotlib.pyplot as plt
3  from matplotlib import style
4  import pandas as pd
5  import pandas_datareader.data as web
6
7  style.use('ggplot')
8
9  start = dt.datetime(2000,1,1)
10 end = dt.datetime(2016,12,31)
11
12 df = web.DataReader('TSLA','yahoo',start,end)
13
14 print(df.head())
```

Part 2: More data handling with the Pandas module and data visualization with Matplotlib

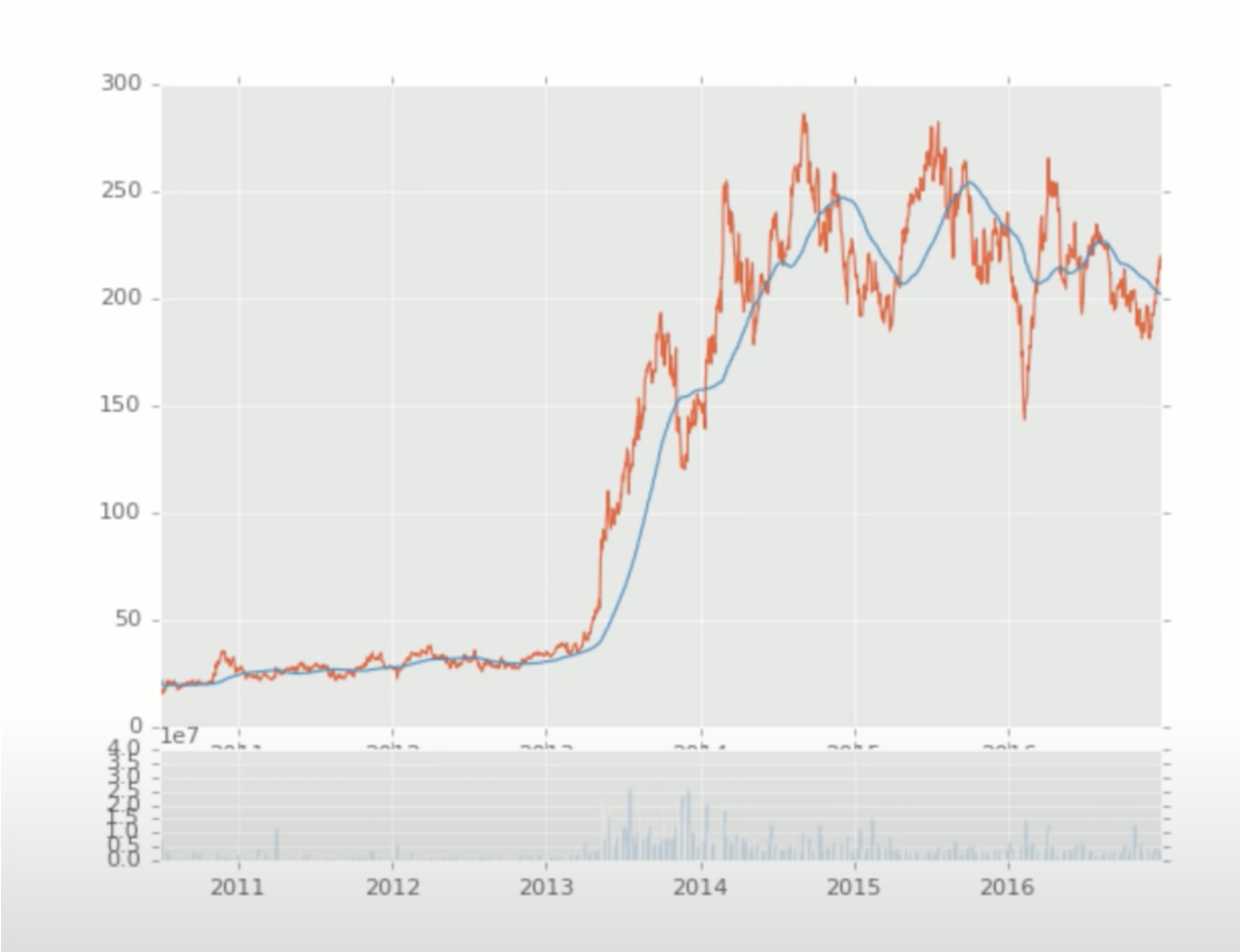
```
1  import datetime as dt
2  import matplotlib.pyplot as plt
3  from matplotlib import style
4  import pandas as pd
5  import pandas_datareader.data as web
6
7  style.use('ggplot')
8
9  #start = dt.datetime(2000, 1, 1)
10 #end = dt.datetime(2016, 12, 31)
11
12 #df = web.DataReader('TSLA', 'yahoo', start, end)
13 #print df.head()
14 #df.to_csv('tsla.csv')
15 df = pd.read_csv('tsla.csv', parse_dates=True, index_col=0)
16
17
18 print(df[['Open','High']].head())
19 df['Adj Close'].plot()
20 plt.show()
```



Part 3: basic data manipulation and visualizations with our stock data

```
1  import datetime as dt
2  import matplotlib.pyplot as plt
3  from matplotlib import style
4  import pandas as pd
5  import pandas_datareader.data as web
6
7  style.use('ggplot')
8
9  #start = dt.datetime(2000, 1, 1)
10 #end = dt.datetime(2016, 12, 31)
11
12 #df = web.DataReader('TSLA', 'yahoo', start, end)
13 #print df.head()
14 #df.to_csv('tsla.csv')
15 df = pd.read_csv('tsla.csv', parse_dates=True, index_col=0)
16
17 print(df[['Open','High']].head())
18 df['100ma'] = df['Adj Close'].rolling(window=100).mean()
19
20 df.dropna(inplace=True)
21
22 print df.head()
23
24 ax1 = plt.subplot2grid((6,1),(0,0),rowspan=5, colspan=1)
25 ax2 = plt.subplot2grid((6,1),(5,0),rowspan=1, colspan=1, sharex
       =ax1)
```

```
24 ax1 = plt.subplot2grid((6,1),(0,0),rowspan=5, colspan=1)
25 ax2 = plt.subplot2grid((6,1),(5,0),rowspan=1, colspan=1, sharex
       =ax1)
26
27 ax1.plot(df.index, df['Adj Close'])
28 ax1.plot(df.index, df['100ma'])
29 ax2.bar (df.index, df['Volume'])
30
31 plt.show()
```

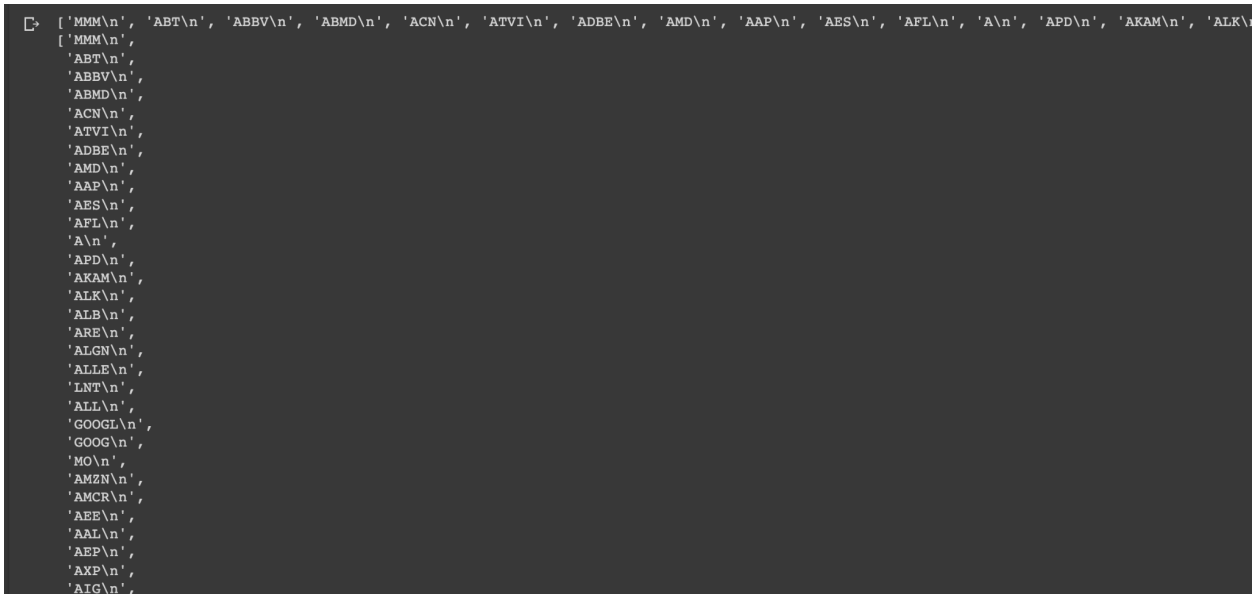

300
250
200
150
100
50
0
1e7
2011
2012
2013
2014
2015
2016

Part 4:

```
import datetime as dt
import matplotlib.pyplot as plt
from matplotlib import style
from matplotlib.finance import candlestick_ohlc
import matplotlib.dates as mdates
import pandas as pd
import pandas_datareader.data as web

style.use('ggplot')

df = pd.read_csv('tsla.csv', parse_dates=True, index_col=0)
#df['100ma'] = df['Adj Close'].rolling(window=100).mean()
#df.dropna(inplace=True)


df_ohlc = df['Adj Close'].resample('10D').ohlc()
df_volume = df['Volume'].resample('10D').sum()

df_ohlc.reset_index(inplace=True)

df_ohlc['Date'] = df_ohlc['Date'].map(mdates.date2num)

ax1 = plt.subplot2grid((6,1),(0,0),rowspan=5, colspan=1)
ax2 = plt.subplot2grid((6,1),(5,0),rowspan=1, colspan=1, sharex
    =ax1)
ax1.xaxis_date()
```


```
23  ax1 = plt.subplot2grid((6,1),(0,0),rowspan=5, colspan=1)
24  ax2 = plt.subplot2grid((6,1),(5,0),rowspan=1, colspan=1, sharex
        =ax1)
25  ax1.xaxis_date()
26
27  candlestick_ohlc(ax1, df_ohlc.values, width=2, colorup='g')
28  ax2.fill_between(df_volume.index.map(mdates.date2num),df_volume
        .values, 0)
29  plt.show()
```

Part 5: grabbing pricing information en masse for a larger list of companies



```
['MMM\n', 'ABT\n', 'ABBV\n', 'ABMD\n', 'ACN\n', 'ATVI\n', 'ADBE\n', 'AMD\n', 'AAP\n', 'AES\n', 'AFL\n', 'A\n', 'APD\n', 'AKAM\n', 'ALK\
['MMM\n',
 'ABT\n',
 'ABBV\n',
 'ABMD\n',
 'ACN\n',
 'ATVI\n',
 'ADBE\n',
 'AMD\n',
 'AAP\n',
 'AES\n',
 'AFL\n',
 'A\n',
 'APD\n',
 'AKAM\n',
 'ALK\n',
 'ALB\n',
 'ARE\n',
 'ALGN\n',
 'ALLE\n',
 'LNT\n',
 'ALL\n',
 'GOOGL\n',
 'GOOG\n',
 'MO\n',
 'AMZN\n',
 'AMCR\n',
 'AEE\n',
 'AAL\n',
 'AEP\n',
 'AXP\n',
 'AIG\n',
```

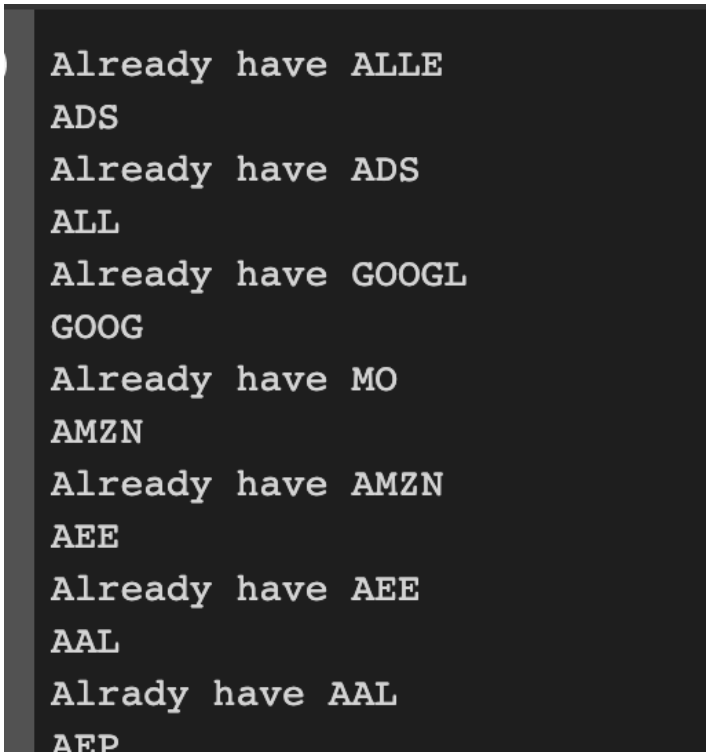
```python
import bs4 as bs
import pickle
import requests

def save_sp500_tickers():
    resp = requests.get('https://en.wikipedia.org/wiki
        /List_of_S%26P_500_companies')
    soup = bs.BeautifulSoup(resp.text)
    table = soup.find('table', {'class':'wikitable sortable'})
    tickers = []
    for row in table.findAll('tr')[1:]:
        ticker = row.findAll('td')[0].text
        tickers.append(str(ticker))

    with open("sp500tickers.pickle", "wb") as f:
        pickle.dump(tickers, f)

    print tickers
    return tickers

save_sp500_tickers()
```

```
    'EMN\n',
    'ETN\n',
    'EBAY\n',
    'ECL\n',
    'EIX\n',
    'EW\n',
    'EA\n',
    'EMR\n',
    'ENPH\n',
    'ETR\n',
    'EOG\n',
    'EPAM\n',
    'EFX\n',
    'EQIX\n',
    'EQR\n',
    'ESS\n',
    'EL\n',
    'ETSY\n',
    'EVRG\n',
    'ES\n',
    'RE\n',
    'EXC\n',
    'EXPE\n',
    'EXPD\n',
    'EXR\n',
    'XOM\n',
    'FFIV\n',
    'FAST\n',
    'FRT\n',
    'FDX\n',
    'FIS\n',
    'FITB\n',
```

Project Part 6:

```
    print tickers
    return tickers

#save_sp500_tickers()

def get_data_from_yahoo(reload_sp500=False):
    if reload_sp500 == True:
        tickers = save_sp500_tickers()
    else:
        with open("sp500tickers.pickle", "rb") as f:
            tickers = pickle.load(f)

    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = dt.datetime(2000, 1, 1)
    end = dt.datetime(2016, 12, 31)

    for ticker in tickers:
        print ticker
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
            df = web.DataReader(ticker, 'yahoo', start, end)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
        else:
            print('Already have {}'.format(ticker))

get_data_from_yahoo()
```

```
import datetime as dt
import os
import pandas as pd
import pandas_datareader.data as web
import bs4 as bs
import pickle
import requests

def save_sp500_tickers():
    resp = requests.get('https://en.wikipedia.org/wiki
        /List_of_S%26P_500_companies')
    soup = bs.BeautifulSoup(resp.text)
    table = soup.find('table', {'class':'wikitable sortable'}
    tickers = []
    for row in table.findAll('tr')[1:]:
        ticker = row.findAll('td')[0].text
        tickers.append(str(ticker))

    with open("sp500tickers.pickle", "wb") as f:
        pickle.dump(tickers, f)

    print tickers
    return tickers

#save_sp500_tickers()

def get_data_from_yahoo(reload_sp500=False):
```

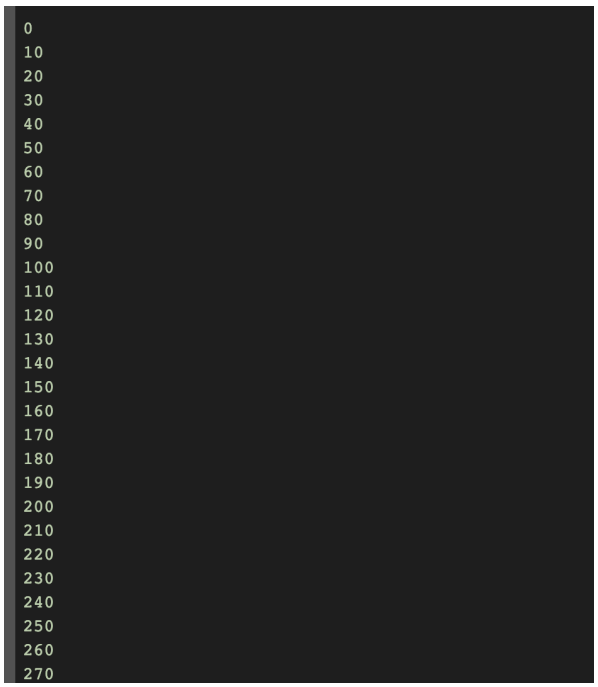


Project Part 7

```
    else:
        with open("sp500tickers.pickle", "rb") as f:
            tickers = pickle.load(f)

    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = dt.datetime(2000, 1, 1)
    end = dt.datetime(2016, 12, 31)

    for ticker in tickers:
        print ticker
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
            df = web.DataReader(ticker, 'yahoo', start, end)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
        else:
            print('Already have {}'.format(ticker))

def compile_data():
    with open("sp500tickers.pickle", "rb") as f:
        tickers = pickle.load(f)

    main_df = pd.DataFrame()

    for count, ticker in enumerate(tickers):
        df = pd.read_csv('/Users/kachunfung/python/finance
            /stock_dfs/{}.csv'.format(ticker))
```

```
        df.rename(columns = {'Adj Close': ticker}, inplace = True)
        df.drop(['Open', 'High', 'Low', 'Close', 'Volume'], 1,
            inplace=True)

        if main_df.empty:
            main_df = df
        else:
            main_df = main_df.join(df, how='outer')

        if count % 10 == 0:
            print (count)

    print (main_df.head())
    main_df.to_csv('sp500_joined_closes.csv')

compile_data()
```

```
import datetime as dt
import os
import pandas as pd
import pandas_datareader.data as web
import bs4 as bs
import pickle
import requests

def save_sp500_tickers():
    resp = requests.get('https://en.wikipedia.org/wiki
        /List_of_S%26P_500_companies')
    soup = bs.BeautifulSoup(resp.text)
    table = soup.find('table', {'class':'wikitable sortable'})
    tickers = []
    for row in table.findAll('tr')[1:]:
        ticker = row.findAll('td')[0].text
        tickers.append(str(ticker))

    with open("sp500tickers.pickle", "wb") as f:
        pickle.dump(tickers, f)

    print tickers
    return tickers

def get_data_from_yahoo(reload_sp500=False):
    if reload_sp500 == True:
        tickers = save_sp500_tickers()
```



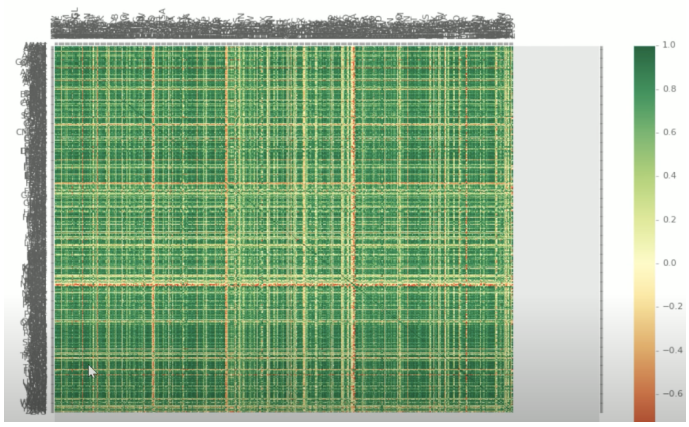
Project Part 8

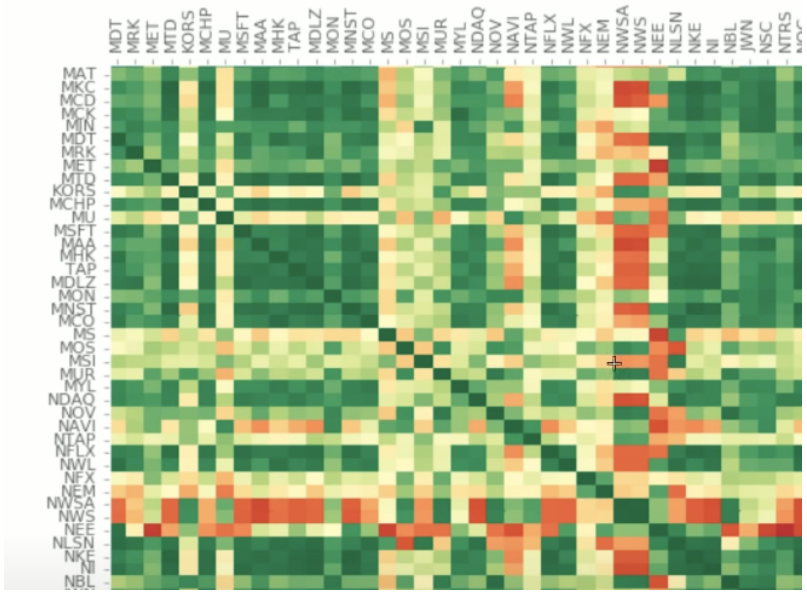
```
def get_data_from_yahoo(reload_sp500=False):
    if reload_sp500 == True:
        tickers = save_sp500_tickers()
    else:
        with open("sp500tickers.pickle", "rb") as f:
            tickers = pickle.load(f)

    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = dt.datetime(2000, 1, 1)
    end = dt.datetime(2016, 12, 31)

    for ticker in tickers:
        print ticker
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
            df = web.DataReader(ticker, 'yahoo', start, end)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
        else:
            print('Already have {}'.format(ticker))

def compile_data():
    with open("sp500tickers.pickle", "rb") as f:
        tickers = pickle.load(f)
```

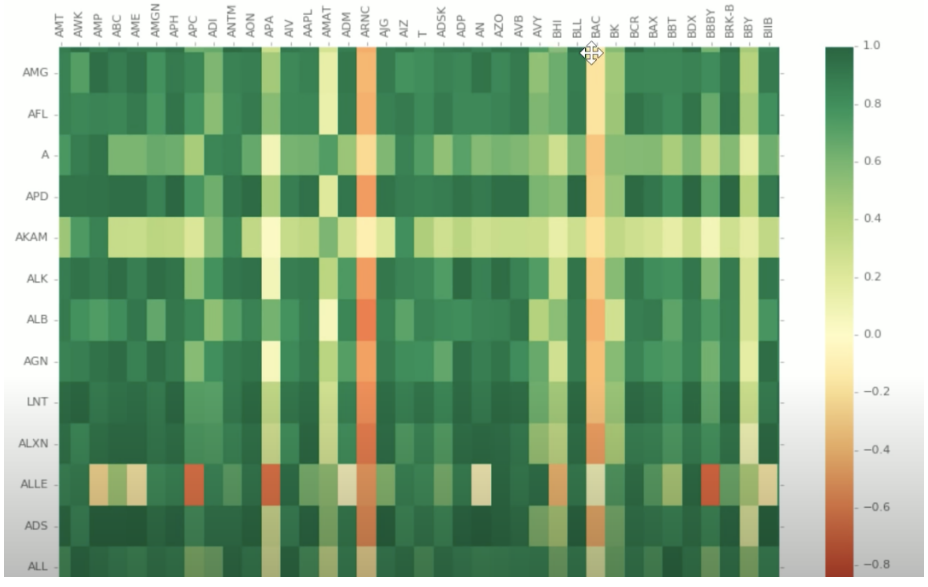
```
import datetime as dt
import os
import pandas as pd
import pandas_datareader.data as web
import bs4 as bs
import pickle
import requests
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np

style.use('ggplot')

def save_sp500_tickers():
    resp = requests.get('https://en.wikipedia.org/wiki
        /List_of_S%26P_500_companies')
    soup = bs.BeautifulSoup(resp.text)
    table = soup.find('table', {'class':'wikitable sortable'})
    tickers = []
    for row in table.findAll('tr')[1:]:
        ticker = row.findAll('td')[0].text
        tickers.append(str(ticker))

    with open("sp500tickers.pickle", "wb") as f:
        pickle.dump(tickers, f)

    print tickers
```

```
    # want a more natural, table-like display
    ax.invert_yaxis() # top of matplotlib graph
    ax.xaxis.tick_top() # move xaxis tick to the top

    # set the labels
    column_labels = df_corr.columns
    row_labels = df_corr.index

    ax.set_xticklabels(column_labels)
    ax.set_yticklabels(row_labels)
    plt.xticks(rotation=90) # rotating the label
    heatmap.set_clim(-1, 1) # limit of color
    plt.tight_layout() # clean things out, show better
    plt.show()
```

```
def visualize_data():
    df = pd.read_csv('sp500_joined_closes.csv')
    #df['AAPL'].plot()
    #plt.show()
    df_corr = df.corr()

    print (df_corr)
    data = df_corr.values # inner value of the dataframe, no
        index, no header
    fig = plt.figure() # create a figure object
    ax = fig.add_subplot(1,1,1) # create an axes object in the
        figure,
    # number of rows, columns, and the ID of the subplot, betw
        1 and the number of columns times the number of rows.

    heatmap = ax.pcolor(data, cmap=plt.cm.RdYlGn) #plot colour
        cmap = from Red to Yello to Green
    fig.colorbar(heatmap) # legend


    # put the major ticks at the middle of each cell
    ax.set_xticks(np.arange(data.shape[0]) + 0.5, minor=False)
    # set_xticks = Set the x ticks with list of ticks
    # arange = form an array
    # shape[0] = how many rows, shape[1] = how many columns
    # data.shape[0] = 505 rows
```


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


MDT
MRK
MET
MTD
KORS
MCHP
MU
MSFT
MAA
MHK
TAP
MDLZ
MON
MNST
MCO
MS
MOS
MSI
MUR
MYL
NDAQ
NOV
NAVI
NTAP
NFLX
NWL
NFX
NEM
NWSA
NWS
NEE
NLSN
NKE
NI
NBL
JWN
NSC
NTRS


AMT
AWK
AMP
ABC
AME
AMGN
APH
APC
ADI
ANTM
AON
APA
AIV
AAPL
AMAT
ADM
ARNC
AJG
AIZ
T
ADSK
ADP
AN
AZO
AVB
AVY
BHI
BLL
BAC
BK
BCR
BAX
BBT
BDX
BBBY
BRK-B
BBY
BIIB
AMG
AFL
A
APD
AKAM
ALK
ALB
AGN
LNT
ALXN
ALLE
ADS
ALL
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

```
import numpy as np
import pandas as pd
import pickle

# 7days, 2% up buy, 2% down sell

def process_data_for_labels(ticker):
    hm_days = 7 # how many days
    df = pd.read_csv('sp500_joined_closes.csv', index_col=0)
    tickers = df.columns.values.tolist()
    df.fillna(0, inplace=True)

    for i in range(1, hm_days+1):
        # up = -i = i days in future
        # (new - old) / old
        df['{}_{}d'.format(ticker, i)] = (df[ticker].shift(-i
            df[ticker]) / df[ticker]

    df.fillna(0, inplace=True)
    return tickers, df

print process_data_for_labels('XOM')
```

```
        if col < -requirement:
            return -1 # sell
    return 0 # hold
```

```
import numpy as np
import pandas as pd
import pickle

# 7days, 2% up buy, 2% down sell

def process_data_for_labels(ticker):
    hm_days = 7 # how many days
    df = pd.read_csv('sp500_joined_closes.csv', index_col=0)
    tickers = df.columns.values.tolist()
    df.fillna(0, inplace=True)

    for i in range(1, hm_days+1):
        # '{}'.format = String formatting
        # up = -i = i days in future
        # (new - old) / old
        df['{}_{}d'.format(ticker, i)] = (df[ticker].shift(-i) -
            df[ticker]) / df[ticker]

    df.fillna(0, inplace=True)
    return tickers, df

def buy_sell_hold(*args): # *args = any number of argument
    cols = [c for c in args] # list comprehension
    requirement = 0.02 # if stock price changes for 2%
    for col in cols:
        if col > requirement:
```

```
    plt.xticks(rotation=90)
    heatmap1.set_clim(-1,1)
    plt.tight_layout()
    #plt.savefig("correlations.png", dpi = (300))
    plt.show()


def process_data_for_labels(ticker):
    hm_days = 7
    df = pd.read_csv('sp500_joined_closes.csv', index_col=0)
    tickers = df.columns.values.tolist()
    df.fillna(0, inplace=True)

    for i in range(1,hm_days+1):
        df['{}_{}d'.format(ticker,i)] = (df[ticker].shift(-i
            df[ticker]) / df[ticker]

    df.fillna(0, inplace=True)
    return tickers, df

def buy_sell_hold(*args):
    cols = [c for c in args]
    requirement = 0.02
    for col in cols:
        if col > requirement:
            return 1
        if col < -requirement:
```

```
def compile_data():
    with open("sp500tickers.pickle","rb") as f:
        tickers = pickle.load(f)

    main_df = pd.DataFrame()

    for count,ticker in enumerate(tickers):
        df = pd.read_csv('stock_dfs/{}.csv'.format(ticker))
        df.set_index('Date', inplace=True)

        df.rename(columns={'Adj Close':ticker}, inplace=True)
        df.drop(['Open','High','Low','Close','Volume'],1,inpl
            =True)

        if main_df.empty:
            main_df = df
        else:
            main_df = main_df.join(df, how='outer')

        if count % 10 == 0:
            print(count)
    print(main_df.head())
    main_df.to_csv('sp500_joined_closes.csv')
```

```
                                (ticker)],
                                        df['{}_5d'.format
                                (ticker)],
                                        df['{}_6d'.format
                                (ticker)],
                                        df['{}_7d'.format
                                (ticker)] ))


    vals = df['{}_target'.format(ticker)].values.tolist()
    str_vals = [str(i) for i in vals]
    print('Data spread:', str(Counter(str_vals)))

    df.fillna(0, inplace=True)
    df = df.replace([np.inf, -np.inf], np.nan)
    df.dropna(inplace=True)

    df_vals = df[[ticker_name for ticker_name in tickers]]
        .pct_change()
    df_vals = df_vals.replace([np.inf, -np.inf], 0)
    df_vals.fillna(0, inplace=True)

    X = df_vals.values
    y = df['{}_target'.format(ticker)].values
    return X,y,df

print extract_featuresets('XOM')
```

```
Data spread: Counter({'1': 1713, '-1': 1456, '0': 1108})
>>>
```

```python
import bs4 as bs
from collections import Counter
import datetime as dt
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np
import os
import pandas as pd
import pandas_datareader.data as web
import pickle
import requests
from sklearn import svm, cross_validation, neighbors
from sklearn.ensemble import VotingClassifier, RandomForestClassifier
import statistics

style.use('ggplot')

def save_sp500_tickers():
    resp = requests.get('http://en.wikipedia.org/wiki/List_of_S%26P_500_companie
    soup = bs.BeautifulSoup(resp.text, 'lxml')
    table = soup.find('table', {'class': 'wikitable sortable'})
    tickers = []
    for row in table.findAll('tr')[1:]:
        ticker = row.findAll('td')[0].text
        tickers.append(ticker)

    with open("sp500tickers.pickle","wb") as f:
        pickle.dump(tickers,f)

    return tickers
```

```
def get_data_from_yahoo(reload_sp500=False):

    if reload_sp500:
        tickers = save_sp500_tickers()
    else:
        with open("sp500tickers.pickle","rb") as f:
            tickers = pickle.load(f)

    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = dt.datetime(2000, 1, 1)
    end = dt.datetime(2016, 12, 31)

    for ticker in tickers:
        # just in case your connection breaks, we'd like to save our progress!
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
            df = web.DataReader(ticker, "yahoo", start, end)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
        else:
            print('Already have {}'.format(ticker))


def compile_data():
    with open("sp500tickers.pickle","rb") as f:
        tickers = pickle.load(f)

    main_df = pd.DataFrame()

    for count,ticker in enumerate(tickers):
        df = pd.read_csv('stock_dfs/{}.csv'.format(ticker))
        df.set_index('Date', inplace=True)
```

```
            main_df = df
        else:
            main_df = main_df.join(df, how='outer')

        if count % 10 == 0:
            print(count)
    # print(main_df.head())
    main_df.to_csv('sp500_joined_closes.csv')


def visualize_data():
    df = pd.read_csv('sp500_joined_closes.csv')
    #df['AAPL'].plot()
    #plt.show()
    df_corr = df.corr()
    # print(df_corr.head())
    df_corr.to_csv('sp500corr.csv')

    data1 = df_corr.values
    fig1 = plt.figure()
    ax1 = fig1.add_subplot(111)

    heatmap1 = ax1.pcolor(data1, cmap=plt.cm.RdYlGn)
    fig1.colorbar(heatmap1)

    ax1.set_xticks(np.arange(data1.shape[1]) + 0.5, minor=False)
    ax1.set_yticks(np.arange(data1.shape[0]) + 0.5, minor=False)
    ax1.invert_yaxis()
```

```python
def extract_featuresets(ticker):
    tickers, df = process_data_for_labels(ticker)

    df['{}_target'.format(ticker)] = list(map( buy_sell_hold,
                                               df['{}_1d'.format(ticker)],
                                               df['{}_2d'.format(ticker)],
                                               df['{}_3d'.format(ticker)],
                                               df['{}_4d'.format(ticker)],
                                               df['{}_5d'.format(ticker)],
                                               df['{}_6d'.format(ticker)],
                                               df['{}_7d'.format(ticker)] ))


    vals = df['{}_target'.format(ticker)].values.tolist()
    str_vals = [str(i) for i in vals]
    print('Data spread:',Counter(str_vals))

    df.fillna(0, inplace=True)
    df = df.replace([np.inf, -np.inf], np.nan)
    df.dropna(inplace=True)

    df_vals = df[[ticker_name for ticker_name in tickers]].pct_change()
    df_vals = df_vals.replace([np.inf, -np.inf], 0)
    df_vals.fillna(0, inplace=True)

    X = df_vals.values
    y = df['{}_target'.format(ticker)].values

    return X,y,df
```

```python
    #clf = neighbors.KNeighborsClassifier()

    clf = VotingClassifier([('lsvc',svm.LinearSVC()),
                            ('knn',neighbors.KNeighborsClassifier()),
                            ('rfor',RandomForestClassifier())])


    clf.fit(X_train, y_train)
    confidence = clf.score(X_test, y_test)
    print('accuracy:',confidence)
    predictions = clf.predict(X_test)
    print('predicted class counts:',Counter(predictions))
    print()
    print()
    return confidence

from statistics import mean

with open("sp500tickers.pickle","rb") as f:
    tickers = pickle.load(f)

accuracies = []
for count,ticker in enumerate(tickers):

    if count%10==0:
        print(count)

    accuracy = do_ml(ticker)
    accuracies.append(accuracy)
    print("{} accuracy: {}. Average accuracy:{}".format(ticker,accuracy,mean(accuracies)))
print(do_ml('AAPL'))
```