Alan Chen

CICS-397 Final Project

12/11/2019

Stock Analysis

Project Description: Exploration and applying one or more of the techniques we studied over the course of the semester to the any dataset of your choice with any method. For instance, you may choose to cluster your examples, and offer some interpretation of the output. Alternatively, you could build a predictive model to try and classify your data or compare the results of multiple classifiers. The goal is to find a question (or set of questions) about the data that interest you and attempt to answer them through your analysis.

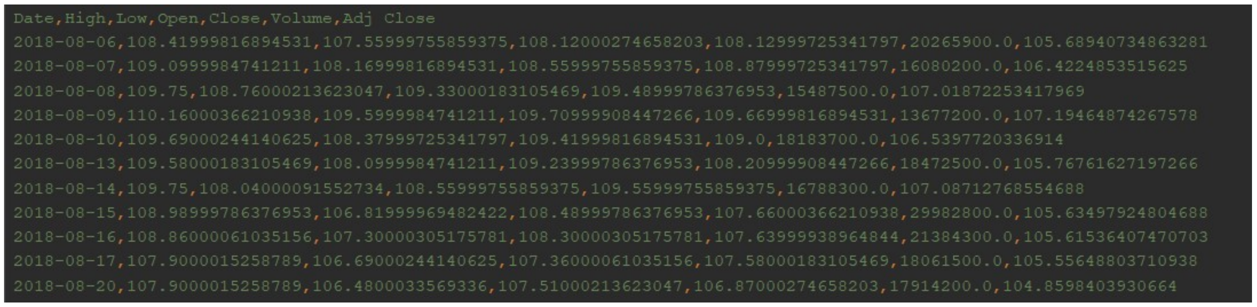
My Objective: Finding correlation between the top 15 largest Holdings in the S&P 500 and building predictive models based on past adjusted close prices to decide whether it is a good stock to invest in. Then using Quantopian to test stock based on moving averages.

Note: Run Main.py

Steps Involved:

1. Used Web Scraper for top 15 largest Holdings in S&P 500 (ticker symbol) from https://www.investopedia.com/articles/investing/053116/10-largest-holdings-sp-500-aaplamznfb.asp and store it in a .pickle file
   Example: ['MSFT', 'AAPL', 'AMZN', 'FB', 'BRK-B', 'GOOG', 'GOOGL', 'JPM', 'JNJ', 'XOM', 'V', 'BAC', 'INTC', 'PG', 'CSCO']

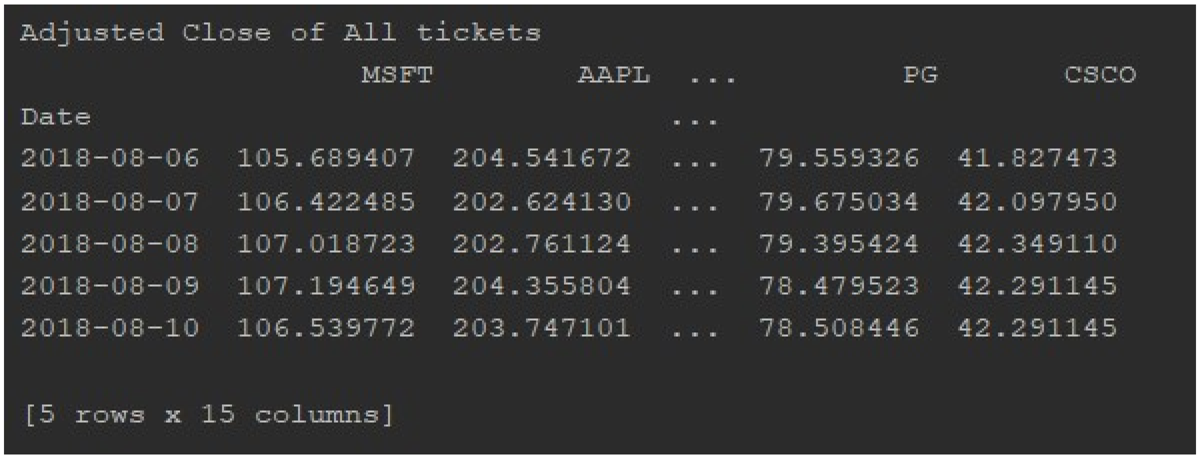
2. Used Panda Data Reader to get basic stats for each ticker symbol (from the .pickle file) from yahoo and stores it in its own directory and .csv file because we do not want to scrape it every time especially if it is a huge dataset.
   Example for MSFT:



```
Date,High,Low,Open,Close,Volume,Adj Close
2018-08-06,108.41999816894531,107.55999755859375,108.12000274658203,108.12999725341797,20265900.0,105.68940734863281
2018-08-07,109.0999984741211,108.16999816894531,108.55999755859375,108.87999725341797,16080200.0,106.4224853515625
2018-08-08,109.75,108.76000213623047,109.33000183105469,109.48999786376953,15487500.0,107.01872253417969
2018-08-09,110.16000366210938,109.5999984741211,109.70999908447266,109.66999816894531,13677200.0,107.19464874267578
2018-08-10,109.69000244140625,108.37999725341797,109.41999816894531,109.0,18183700.0,106.5397720336914
2018-08-13,109.58000183105469,108.0999984741211,109.23999786376953,108.20999908447266,18472500.0,105.76761627197266
2018-08-14,109.75,108.04000091552734,108.55999755859375,109.55999755859375,16788300.0,107.08712768554688
2018-08-15,108.98999786376953,106.81999969482422,108.48999786376953,107.66000366210938,29982800.0,105.63497924804688
2018-08-16,108.86000061035156,107.30000305175781,108.30000305175781,107.63999938964844,21384300.0,105.61536407470703
2018-08-17,107.9000015258789,106.69000244140625,107.36000061035156,107.58000183105469,18061500.0,105.55648803710938
2018-08-20,107.9000015258789,106.4800033569336,107.51000213623047,106.87000274658203,17914200.0,104.8598403930664
```

3. Create a combined CSV for all adjusted-close for all ticker symbols using panda dataframes.join function

   Example:

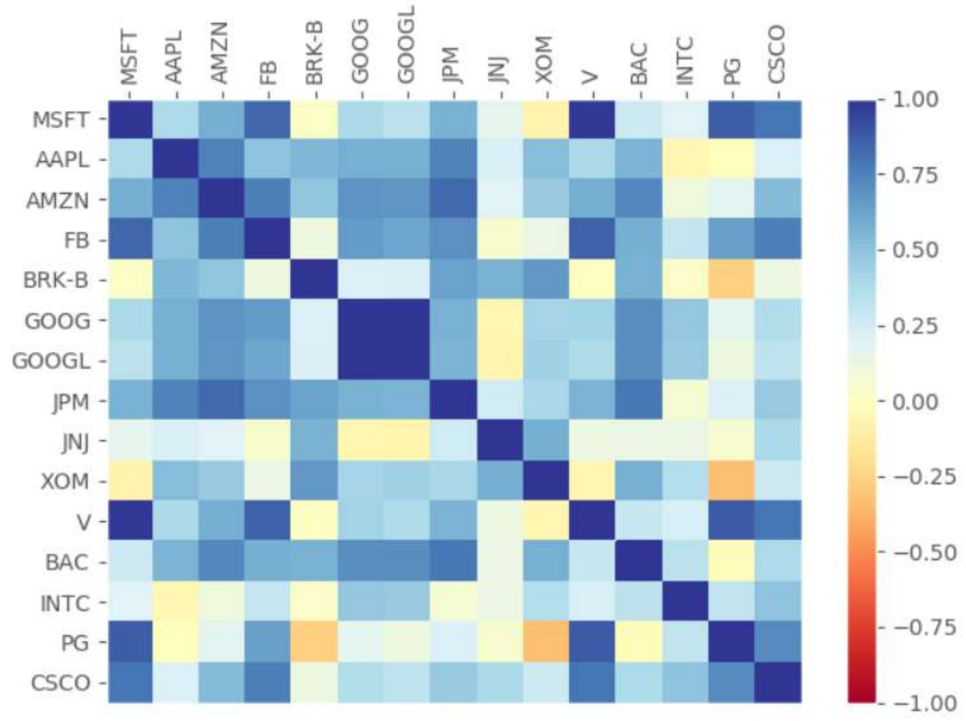


```
Adjusted Close of All tickets
                   MSFT        AAPL  ...         PG       CSCO
Date                                 ...
2018-08-06   105.689407  204.541672  ...  79.559326  41.827473
2018-08-07   106.422485  202.624130  ...  79.675034  42.097950
2018-08-08   107.018723  202.761124  ...  79.395424  42.349110
2018-08-09   107.194649  204.355804  ...  78.479523  42.291145
2018-08-10   106.539772  203.747101  ...  78.508446  42.291145

[5 rows x 15 columns]
```

4. Utilizing Panda Dataframe to create correlation heatmap for the combined CSV and formatting matplotlib in an understandable way.

   Example:

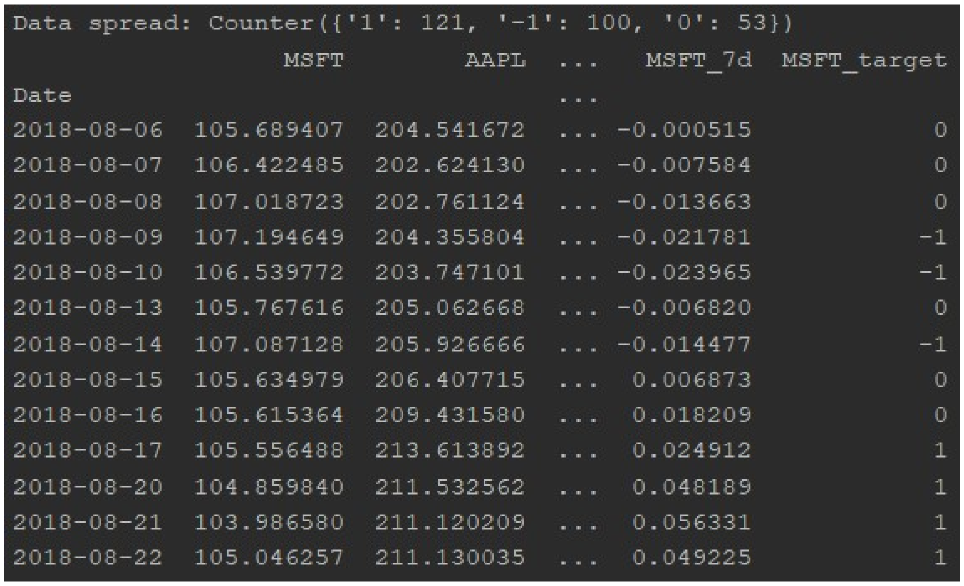


5. Using Sklearn to build machine learning models

   High Level View

   - Format panda data such that there are 7 columns for the 7 days with each column representing differences in the adjusted closing price for the respective timespan
   - For the next 7 days interval, if the adjusted close price is 2%(adjustable) higher, return 1, if it is 2% lower, return -1, else return 0

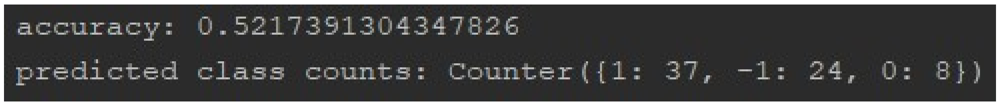
     Example for MSFT:



```
Data spread: Counter({'1': 121, '-1': 100, '0': 53})
                  MSFT        AAPL  ...   MSFT_7d  MSFT_target
Date                                ...
2018-08-06  105.689407  204.541672  ... -0.000515            0
2018-08-07  106.422485  202.624130  ... -0.007584            0
2018-08-08  107.018723  202.761124  ... -0.013663            0
2018-08-09  107.194649  204.355804  ... -0.021781           -1
2018-08-10  106.539772  203.747101  ... -0.023965           -1
2018-08-13  105.767616  205.062668  ... -0.006820            0
2018-08-14  107.087128  205.926666  ... -0.014477           -1
2018-08-15  105.634979  206.407715  ...  0.006873            0
2018-08-16  105.615364  209.431580  ...  0.018209            0
2018-08-17  105.556488  213.613892  ...  0.024912            1
2018-08-20  104.859840  211.532562  ...  0.048189            1
2018-08-21  103.986580  211.120209  ...  0.056331            1
2018-08-22  105.046257  211.130035  ...  0.049225            1
```

   - Finally, split the data into 75% training and 25% test set. I chose to use Voting Classifier which is a collection of several models working together on a single set. In short, it combines the prediction of multiple machine learning algorithms. As a result, it has lower error rate and less chance for over-fitting. Train the Classifier and then perform prediction on the test set.

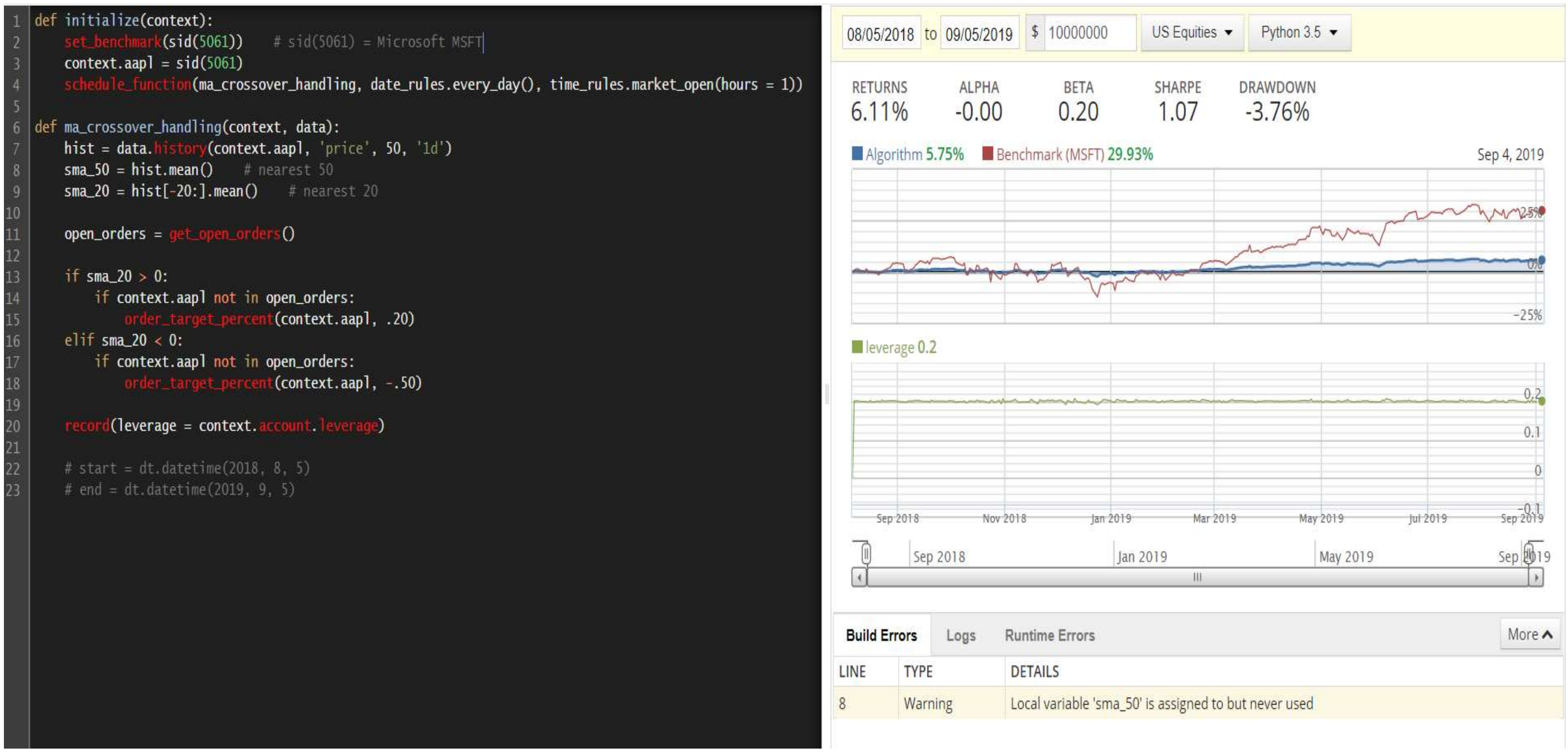
     Example Output for MSFT:



```
accuracy: 0.5217391304347826
predicted class counts: Counter({1: 37, -1: 24, 0: 8})
```

     Summary: The model predicted that of the 69days (37+24+8), 37 of them will have a return that is greater than 2% and 24 days that is less 2%

6. Quantopian is a quantitative research platform used to analyze, develop, test, and use trading algorithms. We will be testing moving averages for Microsoft on this platform. The code is on Quantopian and therefore will not be submitted with this project, but there is a screenshot below.



```
def initialize(context):
    set_benchmark(sid(5061))    # sid(5061) = Microsoft MSFT
    context.aapl = sid(5061)
    schedule_function(ma_crossover_handling, date_rules.every_day(), time_rules.market_open(hours = 1))

def ma_crossover_handling(context, data):
    hist = data.history(context.aapl, 'price', 50, '1d')
    sma_50 = hist.mean()    # nearest 50
    sma_20 = hist[-20:].mean()    # nearest 20

    open_orders = get_open_orders()

    if sma_20 > 0:
        if context.aapl not in open_orders:
            order_target_percent(context.aapl, .20)
    elif sma_20 < 0:
        if context.aapl not in open_orders:
            order_target_percent(context.aapl, -.50)

    record(leverage = context.account.leverage)

    # start = dt.datetime(2018, 8, 5)
    # end = dt.datetime(2019, 9, 5)
```

08/05/2018 to 09/05/2019 $ 10000000 US Equities Python 3.5

RETURNS	ALPHA	BETA	SHARPE	DRAWDOWN
6.11%	-0.00	0.20	1.07	-3.76%

Build Errors Logs Runtime Errors More

LINE	TYPE	DETAILS
8	Warning	Local variable 'sma_50' is assigned to but never used

Result for MSFT: Moving average is the mean of the latest n days. We looked at 20 days specifically. If the moving average is less than 0 than we short MSFT with 50% of our equities, else if it is greater than 0, we buy MSFT calls with 20% of our equities. This algorithm is designed to have low risk as you can see our volatility (BETA value on the top right) is low. Although my algorithm has a 5.75% return, the benchmark, MSFT itself, have 29.93% return (profits are still profits 😊).

For this project, we looked at MSFT a lot for our examples, and based on the analysis, it seems like a look stock to invest in.

References:

https://pythonprogramming.net/getting-stock-prices-python-programming-for-finance/