**Trading Strategy Using Python**

**Final Project Report – IST 652**

**Paul Sujith Rayi**

**The Proposal**

**Decision to work individually or in a group:**

I have chosen to work individually for this project. I think I will get to learn more if I work individually in this class. I am not against working in teams and I have gelled well with teams and group projects in the past. I am also aware that once I enter the workforce, I will be working with various teams across many dimensions in an organization and I am looking forward to it.

**Topic of Investigation:**

My topic of investigation for this project is Stock Analysis using Python programming language.

**Planned datasets for usage:**

The datasets I will be working with are stock prices of specific public companies that are traded in the stock market. The typical columns in the dataset would include date, open, high, low, close and volume. There might be other datasets that I will be working on such as stock indices and company financial statements. I might also work with news articles for the stock or company that I am analyzing. This depends on the scope constraint and time constraint of the project. The major source or website from which I will be downloading my dataset is Quandl. Quandl is a premier source for financial, economic, and alternative datasets, serving investment professionals. Quandl’s platform is used by over 400,000 people, including analysts from the world’s top hedge funds, asset managers and investment banks. The link to the website is here

<https://www.quandl.com/>

**Methods of Data Acquisition and Analysis:**

I will be using pandas to put the acquired datasets in data frame objects to perform analysis on. Then I will explore the data by cleaning it and saving just the required columns. I will then do summarization on a unit of analysis and look at how to manipulate the data so as to gain useful insights out of it. I will be using pivot table and other techniques to do the summarization. I will also perform some data visualizations so as to have a better view of the data. If I get a chance to analyze text data as part of this project, then I will be using packages such as scikit-learn and Natural Language Tool Kit (NLTK) to do text data analyzing.

**Potential Development Tasks and Additional Guidance:**

I will try to make a strategy for investing or come up with an idea of utilizing a strategy for investing. I may come up with a plan of combining this data set analysis with the information given in the annual statements and also news articles of the company and try to see the correlation between the two so as to utilize the insights for analysis. I will be needing help from Quandl, so as to how to efficiently work with their datasets and analyze them. I will be looking and seeking help from online financial analysis users and their projects so as to emulate them and achieve the desired results. Last, but not the least, I will be needing help from my professor to guide me through the journey and help me achieve great insights by doing this project.

**The Report**

**The Difference**

The proposal was a preliminary view as to what I am about to do for my scripting project. My data source for this project was Yahoo Finance. I chose this instead of Quandl (which I mentioned in the proposal) for extracting and collecting data because I was able to use the pandas datareader package to extract the data from Yahoo Finance website without the need of any API. This technique I thought was easy and hassle free.

I did not do any text mining using packages such as Natural Language Processing Toolkit (NLTK) which I mentioned in my proposal. The reason I did not do this is because my focus was on demand and supply of shares in the stock market and price fluctuation based on that movement. My perspective to develop this trading strategy is that of a day trader as opposed to a long-term investor. Hence, I did not look at the annual statements data, analyst reviews and news articles of the company as I mentioned in the proposal.

**Importing Data Using Pandas Datareader Package**

I imported the pandas datareader package into my notebook (jupyter notebook). Then I also imported the datetime package which we will be using to extract our desired data. Then I used the get\_data\_yahoo function to extract data of company Apple’s stock and I stored it in a variable. I used the start and end methods which are part of the get\_data\_yahoo function to specify the dates for which I wanted to extract my data. I collected Apple stock price data for the past five years starting from Apr 1st, 2014 and until Apr 1st, 2019.

The data gets returned to us in the form of a dataframe. I used the head and the tail functions to check out the first and last five records/rows of our dataframe. The Dates are our index of the dataframe. We have ‘High’, ‘Low’, ‘Open’, ‘Close’, ‘Volume’ and ‘Adj Close’ as our columns. The ‘High’ here stands for the highest price for which the stocks have been traded during that particular date. The ‘Low’ here stands for the lowest price for which the stocks have been traded during that particular date. The ‘Open’ here stands for the opening price of the stock for that particular day. The ‘Close’ here stands for the closing price for the stock for that particular day. The ‘Volume’ stands for the total number of company shares traded during that day. The ‘Adj Close’ here stands for the closing price plus or minus any adjustments to its price based on events that happened before the start of trading for that particular day. I then ran the describe function on my dataframe to run some summary statistics on my dataframe. The mean, standard deviation, minimum and maximum values are pretty close for all columns excluding the volume column based on my observation of the summary statistics. Well in order to properly conclude this we have to also consider the amount (volume) of Apple stock that has traded in the market.

**Using Pandas**

Now I import the pandas package into my notebook and using the to\_csv function store my dataframe in a csv file in the same directory my notebook is saved. This is especially helpful if you have lost connection to the Yahoo Finance website through the pandas data reader package and you need your extracted data for analysis purpose. Then I read in the data using the read\_csv function which parses my dataframe from csv into my worksheet. I give specifications in my read\_csv function as to make my first row as the header row, Date column to be the index and parse the dates as true.

Then I print my dataframe in the worksheet that I stored in the variable. And it looks wonderful.

**Working with time series data**

Now I print the index of my dataframe. I print the columns of my dataframe. I then print the last ten rows of my dataframe for only the ‘Close’ column stored in a variable and finally print the type of the variable. As it turns out it is a pandas data series.

Now I use the loc function and the timestamp method to print all the rows of May and June months of 2014 year specifically. Then I use the loc and head functions to print the first few rows of only year 2015. Then I use the iloc function to print all rows of May month of 2014 year. I then use the iloc function to print values of Open and Close columns for two specific dates in my dataframe.

Now I use the sample and print methods to randomly select and print twenty rows from my dataframe. Then I use the resample method to store my values on a monthly basis and store the result in a variable. When I print the variable, it gives me all the values of the methods in the resampled data.

Now what I do is, I add a new column to my dataframe called ‘diff’, whose values are the difference between the values open and close columns of all the records/rows. I then use the head function to print the first few rows of our dataframe with our newly added column. I then delete the newly created column as we do not need it for our analysis.

The steps that I have taken in the above four paragraphs is to show how we can explore and extract our desired values/records from our dataframe.

**Visualizing Time Series Data using Bokeh**

Bokeh is the advanced library package which I wanted to use for this project. It is a sophisticated and easy to use data visualization library that contains useful features and functions to represent graphs elegantly.

I use ‘conda install bokeh’, to install bokeh package in my notebook. Then I import figure, output\_notebook and show methods from boke.plotting library. Then I store the values of ‘Close’ column of my dataframe in a variable. I then store the index of my dataframe in a variable too. Then I use the figure function to give the dimensions of my plot and the axis type of my x-axis and store it in a variable. I then use the line function from bokeh as I want to print a line graph and give the necessary method specifications to it. I then make use of different methods available as part of the bokeh library to specify different parameters to my plot such as title, label etc. Then I use the show function to print the plot on the console. The plot gets printed beautifully on the console with the line depicting the closing stock price of Apple. Looking at the line, we can say that the stock price of Apple stock has slowly risen which is a good thing for the company. The output\_notebook method which we imported earlier basically makes sure that the plot gets printed in the console of our notebook.

**Common Financial Analysis**

**a) Returns**

I now store all values belonging to the ‘Adj Close’ column in a variable. I use the percentage change function to calculate the percentage change on the values stored in the variable and store the results in another variable. Then I use the fillna function to replace all NA values with 0 in our variable that contains percentage changes. I now print the daily percentage changes on the console. I import numpy package into my notebook. I use the log function to calculate the log returns of our percentage changes of our ‘Close’ column values which we stored in a variable and store the results in another variable. I then print the results of our log returns. I calculate the log returns here because I believe that the log returns show us the true/actual rise or fall in the price of the stock over time.

Now I use the resample method again to change/aggregate our dataframe values into business months and I use the apply function and list comprehension to aggregate the values to the end of the business months. Then I use the percentage change method on the business monthly data to calculate the percentage change in the values of our business monthly dataframe. I now use the resample method again to aggregate the values of our original dataframe into quarters and take the mean of the values per quarter for every column. I store the resultant dataframe into a new variable called quarter. I then calculate the percentage change in the values of our new dataframe using the percentage change column. As you can see on the console, our quarterly dataframe gets displayed along with the values of percentage changes for each column.

Now I use the shift function on the daily close variable which contains values of the ‘Adj Close’ of each day and calculate the returns and store it in a new variable. I print the results of this variable on the console.

Now I import the matplotlib.pyplot package into my worksheet to do some visualization. I create a histogram using the hist function provided by the matplotlib library for the values stored in the variable which I described in my previous paragraph and set the number of bins to be 50. I use the show function to print the histogram on the console and it gets printed beautifully. I now use the describe function to look at the summary statistics of our variable which we visualized and print it on the console.

I use the cumprod function to calculate the cumulative product of our percentage returns stored in the variable and store the result in a new variable. I now print this new variable on the console. We do this to understand how returns have moved over the years, that is, whether they increased or decreased and by how much.

Now we plot the cumulative daily adjusted closing returns using bokeh. We import the relevant methods from our bokeh package we installed earlier. We then store our ‘Adj Close’ column which is part of our cumulative returns in a variable and we store the index of our cumulative returns in another variable. We then use the figure function to give the necessary specifications to our plot. We then use the line function to pass in our variable containing the values and index and give values to other necessary methods based on our specifications. We then use various methods to add more specifications to our plot such as title, x-axis label, y-axis label etc. We then use the show function to print our plot beautifully on the console. The line slowly rises in a consistent pattern which is a good sign that the cumulative returns are slowly rising with time. It shows good performance by Apple stock over time. We call the output\_notebook function to make sure that the plot gets printed inline on the console of our jupyter notebook.

We now use the resample method to aggregate our cumulative returns monthly and store the results in a variable. We print the results of the variable on the console.

We now use nested functions to extract the stock data for our selected tickers. The companies for which we extracted the data are Apple, Microsoft, IBM and Google. We concatenate all the data we extracted into one dataframe and specify ticker name and dates as our index.

Now we just select values of ‘Adj Close’ column of all our tickers and store it as a dataframe in a variable. We use the reset.index and pivot functions to reset our index and pivot our values so that Date is our index and each ticker column represents ‘Adj Close’ values of the respective tickers. If you look at the console of our notebook where I printed this dataframe, you will be able to understand this better. Now I use the percentage change method to calculate the percentage change values for all tickers and store it in a variable. Now I use the hist function with the necessary methods to produce a beautiful histogram on the console. We will see four separate histograms as we have four stock tickers in our variable, and they all share the same x-axis.

Now I use a scatter matrix to print of all my tickers with their percentage change values on the console.

**b) Moving Windows**

I again take the values of ‘Adj Close’ column of my Apple stock and store it in a variable. I then calculated the rolling mean of it with my window (number of observations) being 40 observations. I stored the rolling mean value in a variable. I then print the last ten values of my variable which has the rolling means on the console.

Now I created two columns in my Apple dataframe where one column consists of the rolling mean with a window of 42 observations and the other window consisting of the rolling mean for 252 observations. Then I imported the relevant methods from my advanced data visualization library bokeh. I have then assigned the values of the two columns that I created for the Apple dataframe, along with the ‘Adj Close’ column values and the index of my Apple dataframe to different variables. Now I call the output\_notebook function so that the plot we are printing gets printed inline on the screen of our notebook console. I then use the figure function provided by bokeh to specify the necessary methods and I use the line function thrice for each of our three columns and specify the necessary methods for it too. Then I use some of the methods available to me from the bokeh library to specify certain characteristics to my plot such as title, legend location, x-axis label etc. Then I use the show function to print the graph on the console. Our plot gets displayed beautifully on the console. By looking at the plot, I can say that the line resembling the short window of 42 observations aligns closely with the line representing ‘Adj Close’ values over the years as compared to the long moving window of 252 observations.

**c) Volatility Calculation**

I import the matplotlib.pyplot library to do some visualization now. I assign the number 75 to a variable. Then I calculate the rolling standard deviation of the values of the dataframe based on my window of 75 observations. This contains values of percentage changes for all four of our tickers namely Apple, Microsoft, Google and IBM. I then divide the rolling standard deviation by the square root of 75 which is our rolling window and store the results in a variable. I do this because this is one of the ways in which volatility is calculated in the stock market. I then visualize my results using the plot function by specifying the necessary methods. The plot gets printed beautifully on the console with different lines representing volatility of different stocks/tickers.

**d) Ordinary Least Squares**

Now I import the package ‘statsmodel’ into my notebook, which we will be using to create a regression model using the ordinary least squares method. I also import other necessary methods/objects from the pandas and pandas.core library. I store the values of ‘Adj Close’ column from my all\_data dataframe which consists of the values of all four tickers which we extracted in a variable. I then calculate the log returns on the variable using the log and shift functions and store the results in another variable. Now I use the iloc function to extract all log returns pertaining to Apple stock and store it in a variable. I store the index of those log returns in another variable where I drop the ticker level and just store the dates because I know that all my values are of the ticker Apple. Just like how I did for Apple ticker, I do the same for Microsoft ticker. I store the returns in a variable and the index in another. Now I concatenate both the returns stored in variables into one dataframe and give it the column names of ‘AAPL’ and ‘MSFT’. I use the add\_constant to set my Apple returns to be my constant in the regression model. Now I use the OLS function to create a model in which my Microsoft returns is my dependent variable and I use the fit function to fit the model for our returns. I store everything in a variable and print the summary statistics of the variable using the summary function. The summary statistics get printed beautifully on the console. In the summary statistics one of the most important is the R-squared and the Adjusted R-squared values of the model. The R-squared tells how much variance (error difference) is explained by the model in predicting the returns. Here our Adjusted R-squared value is the same as the R-squared value which tells us the degrees of freedom of residuals did not have much of an impact on the explanation of the errors on the predicted values. I am not going to explain what the degrees of freedom of the residuals mean here as this is beyond the scope of my project. Same goes for other statistical results produced from the model. If you need any further explanation feel free to email me at [parayi@syr.edu](mailto:parayi@syr.edu).

The above statistical models like ordinary least squares can be helpful while analyzing stocks belonging to the same industry with similar goals and products. Such model results however must be perceived with caution as they are built purely on the past demand and supply of those particular stocks in the market.

Now I use the plot function to print the regression line on the console. The regression line tells us how the model fits. And I also produce a scatter plot of the returns of Apple and Microsoft tickers in the same graph. I specify the values for necessary methods such as grid, axis, label etc. for the plot before printing it on the console.

Finally, I use the plot and correlation functions to graph the rolling correlation between the returns of the apple and Microsoft tickers with a rolling window specification of 252 observations.

**Building a Trading Strategy**

The trading strategy that I am creating is called the moving average cross over. Here I create two simple moving averages of a time series with differing lookback periods of 40 and 100 dates/observations. If the short moving average exceeds the long moving average for the short window we go long and in the reverse circumstance, we go short. Going long is buying the stock and going short is to sell the stock.

Now I create two windows, one is 40 and the other is 100. These two represent dates/days/observations. I create a dataframe named signals to which I assign the index of my Apple dataframe. I then create a column called signal in my signals dataframe and assign all values as 0.0 for all my dates. I create a new column called short\_mavg to which I assign rolling mean values for all observations where my window is short and that is 40 days. I create another column called long\_mavg to which I assign rolling mean values for all observations where my window is long and that is 100 days. Now to the signal column in my dataframe I assign values of 1.0 if my short moving average is greater than the long moving average for the short window and I assign 0.0 if my short moving average is less than the long moving average for the short window. After this I create a new column called positions which is the difference between signal values from one observation to the other. Now I print my signals dataframe on the screen.

Now I use the figure and plot functions with the necessary specifications to produce a wonderful graph depicting the moving averages and the triangle depicting the change in positions. The straight triangle depicts the long position and the inverted triangle depicts the short position.

**Backtesting the Trading Strategy**

For backtesting our trading strategy, we need four things. The first one is a strategy which we already have with us. The second one is a data handler/interface which we already have and used which is our pandas datareader package. The third thing is a portfolio which we are going to build and implement now. The last and final piece is an execution handler/broker. We will not be making use of an execution handler in this project as we are not actually going to trade in the stock market.

**Implementing a simple backtester using pandas**

First, I set our initial capital to be $100,000 which I store as float type in a variable. I then create a dataframe called the positions to which I assign the index of my signals dataframe that I created earlier. I then create a column called ‘AAPL’ in my dataframe whose values are calculated to be 100 times the values in the signal column of the signals dataframe and assigned. Now I use the multiply function in pandas to multiply all values of my positions dataframe with the ‘Adj Close’ values of my Apple dataframe and store the results in a variable. Then I use the diff function on values in my positions dataframe to calculate the difference in values columnwise simultaneously and store the results in a variable. I add a new column to my portfolio dataframe called holdings by applying the multiply function on the ‘Adj Close’ column values of my Apple dataframe and aggregating using the sum function. Then I add another column called cash to my portfolio dataframe which is the difference between our initial capital and the cumulative sum of values stored in our difference variable and multiplying the same with the ‘Adj Close’ values of our Apple dataframe. Here the multiplying takes place first, before the sum and then cumulative sum takes place. All of which is subtracted from the initial capital and the values are stored in the cash column which we created. Then I add another column to my dataframe called the ‘total’ which is basically the addition of values in the cash and holdings columns. The ‘total’ column values is the value of our portfolio on that particular date. Then I create another column called ‘returns’ which is basically the percentage change of values in my ‘total’ column. I use the percentage change function to calculate these values.

Now we use the figure and plot functions to plot the graph of the total values of our portfolio from our portfolio dataframe. We mark points where the signal is positive indicating a long position and represented on the graph by a straight triangle. We also mark points where the signal is negative indicating a short position and represented by an inverted triangle on the total line on the plot.

**Data Questions that I tried to answer from doing this project**

1) How to elegantly extract stock price data from a reliable source and explore it

2) How to produce good data visualization graphs to better understand and analyze the stock price data we have with us

3) How to create a simple regression model using ordinary least squares method and explore the results of our statistical analysis

4) How to develop a trading strategy and backtest it and look at the returns generated

**Conclusion and Final Remarks**

This project should not be perceived as financial advice and the trading strategy should not be treated as a viable strategy to trade in the stock market. Stock price movement depends on various factors other than just demand and supply of the stock in the market. This is an exploratory project of the various concepts that I learned in my scripting class. I thank the professor for this opportunity.

**References**

1) <https://www.datacamp.com/community/tutorials/finance-python-trading>