FE-520: Python for Financial Applications

Final Report

I pledge my honor that I have abided by the Stevens Honor System

Project Motivation:

With the modernization in technology, monetary administrations have outfit the potential through different methods to improve gauges. The essential for any venture judicious begins with the goal to make significant expectations about the eventual fate of a monetary instrument and put resources into stocks whose characteristic worth isn't precisely reflected on the lookout. With the capacity to assemble and handle colossal measure of recorded information investigator in the course of recent years have had more significant yields than normal.. The techniques which have yielded the highest average returns use quantitative and technical analysis for investment in the equity investment horizon

The explanation we needed to profound jump into the value market as opposed to the fixed pay market is a result of the incredible risk-adjusted return. The difficult part is to precisely break down the risk related to a venture. The broadening of resources/Diversification has in this manner become a vital piece of all the significant speculation firms on the planet. Every investor is unique in a way that their risk-taking appetite and the expected returns are different. Although at its core all investors are risk-averse. If they do invest in risky assets the risk-adjusted returns need to be accordingly reflected in the assets price. An investor holds multiple stocks in his/her portfolio and uses diversification to hedge the risk of the overall portfolio. Diversification is done in many ways like investing in different sectors, different asset classes. Equity universe assets are pegged against a stable index like the S&P500, S&P500 to measure the overall performance of the portfolio. Having established the fact that risk management is critical to the investment process we wanted to explore this avenue as the next subset of our research.

Beta is one of the most critical factors taken into consideration to measure the risk of a portfolio which cannot be reduced even after diversification. It measures the risk an investment adds to an already diversified portfolio. Beforehand the solitary regular approach to diminish the risk related to a portfolio was to expand and spread out the resource across different sectors. The basic idea behind analyzing the beta is to measure the performance of a stock to a stable index. We took the S&P500 index which consists of 30 large-cap equity stocks and gives a reasonable estimate of the overall market risk. The exactness of the relapse to compute beta relies upon the example space of the data set. Ideally, at least one year of historical data is considered for the regression.

The closing prices of the stocks that need analysis along with the closing price of the index help us get an idea of the monthly or weekly returns. This is when we establish the relationship between some of the key components required for calculating beta. The mathematical formula for the calculation of beta is the covariance of the stock and the index divided by the variance of the benchmark which in our case is the S&P500. Covariance is the relationship between the movement of two asset prices. A positive correlation means the assets move together and are positively correlated. If they move in opposite direction it is negative correlated. Variance is a statistical measure of how much a set of observations differ from each other which is why we calculated the monthly or weekly returns. Current closing price subtracted by the previous closing price divided by the current price gives us how much the observational values are apart. Beta are defined into different ranges to give us a holistic picture as to what the beta values indicate.

A beta of one reflects that the risk associated with the stock is the average market risk and mimics the reference index. A beta higher than 1 indicates that the stock is riskier than average and beta of less than one indicates that the stock is less risky than average market risk. We achieved these estimates of beta of various stocks using a linear regression in python. Downloading the statistics from the data source like yahoo finance for each company individually is a very taxing and time consuming activity. To optimize this process and save time we utilized API to import data from yahoo finance. We then proceeded to calculate the beta based on a custom function in python that was built from scratch without making use of any libraries. Post the beta calculation we proceeded to calculate beta for a variable and moving window. In a time series, each value is affected by the values just preceding this value which is also referred to as autocorrelation. This gives us an idea about how beta values change over time and enable us to make effective investment decisions

Approach:

After much arranging and meeting, the objective of making a regression package for Python was set by our group. This regression package would be totally worked from the beginning, exploiting libraries just for handling and manipulating the information. The essential constants for performing linear regression would be implemented as individual functions in the regression class. Our primary focus was on developing a linear regression method since this component is most relevant to the problem we were determined to solve. The formula for the beta coefficient of a single stock is exactly the same as that used to determine the slope of the best-fit line in linear regression.

Our package starts by taking in the accompanying data: stock tickers the client wants to check, the beginning and end dates of the time of interest, and the sample size of the stock data. Stock ticker data is imported utilizing the yfinance (Yahoo Finance) library for Python. Data for every ticker for the whole time frame is downloaded as a consolidated pandas information outline, from which we extract only the closing prices. We repeat similar steps to the S&P500 since it will fill in as the market reference. In specific cases a stock ticker may not contain data that covers the whole indicated time-frame, so all sections containing NA esteems will be dropped naturally.

Subsequent to cleaning the data, we test from it utilizing the sample size that the client determined. The package will create subsets (windows) of the original dataset equal with sizes equal to that of the sample size. It does so by taking the first 1 to N entries of data in the original dataset and storing the resulting data frame as an element of a list. The next window will contain entries between 2 and N+1 and will be appended to the same list containing the first window. Each window will be advanced by 1 entry, and this process repeats until the window reaches the end of the original data frame. In testing our code, we utilized a fixed window size of 30 days since it approximates beta for about a month of time; yet we wound up utilizing a variable window size since this boundary influences the measure of data in a given period, and therefore influences the estimation of beta.

From here, we would now have the option to begin figuring the pertinent qualities expected to decide beta and perform linear regression. In these assessments, the results are put away like the examples of the prices: in lists of pandas data frames or series objects where each data frame/series is an individual element of the list and stock tickers are used to index the value(s). Using a for loop and NumPy, the means of each ticker are determined for every single window. The resulting information can then be used to determine the variance and covariance using the following formulas

$$\text{covariance} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{N - 1} \qquad \text{variance} = \frac{\Sigma(x_i - \bar{x})^2}{N - 1}$$

X and y in the above formulas refer to individual stock ticker data and S&P500 data, respectively.

The slope of the regression line (beta) can then be found by taking the covariance of the stock price and S&P500 price, and dividing it by the variance of the stock price for the corresponding window. The intercepts of the regression lines are then calculated using the following formula.

$$\text{intercept} = \bar{y} - (\text{slope}) * \bar{x}$$

At this point, all of the basic values/coefficients for determining a best-fit line have been calculated and stored in lists of pandas series objects. Using the same logic and approach used to determine the above values, we determined the R-Squared value, the standard error coefficient, and the t-score. All of the values so far were determined from formulas within the code, but the remaining one (p-value) had to be determined using a scipy statistics package. This is because p-values require referencing a table (or using a very convoluted set of formulas). The p-value function takes the t-score from a window for a specific stock and the window size as inputs.

$$R^2 = 1 \frac{\Sigma(xi - fi)2}{\Sigma(xi - \bar{x})}$$

$$\text{Standard Error} = \frac{\sqrt{\frac{\Sigma(yi - fi)2}{N - 2}}}{\sqrt{\Sigma(x_i - \bar{x})2}}$$

t = (slope)/(Standard Error)

All of the aforementioned logic and code is packaged into a linreg.py python class, which contains the various functions necessary to run linear regression and plotting on a portfolio of stocks. These functions all share the same instance variables declared in the init function, therefore the functions must be called in the order provided.3

The functions included in the package are as follows:

- __init__ (tickers)
    - Takes as input a list of stock tickers as the input and creates all the class instance variables that will be filled by later functions

- PullPrices(start_date, end_date)
    - Takes as input the date range of stocks that should be pulled from Yahoo Finance and returns a pandas dataframe containing closing prices for the list of stocks.

- DefineWindow(window_size)
    - Takes as input a window size (int) that will be used for moving window calculations at later steps. Returns a list of subsets for each moving window period.

- CalculateVariance()
    - Takes instance variables populated by previous function calls and returns a subset list of variance for each stock in the portfolio.

- CalculateCovariance()
    - Takes instance variables populated by previous function calls and returns a subset list of covariance for each stock in the portfolio.

- CalculateSlopes()
    - Calculates the slope of the regression line, also known as the beta value. Returns a list of subset (for each window) slopes for each stock.

- CalculateIntercepts()
    - Calculates and returns the intercept values for each stock in each window.

- PlottingWindow(window_num, ticker)
    - Plots a matplotlib plot for the ticker and window number that is input into the function

- CalculateRsquared(window_num, ticker_list)
    - Takes in the window number and a list of tickers and returns the R squared value for those specific stocks in that specific window.

- CalculateStdcoeff(window_num, ticker_list)
    - Takes in the window number and a list of tickers and returns the standard error coefficients for those specific stocks in that specific window.

- CalculateTscores(window_num, ticker_list)
    - Takes in the window number and a list of tickers and returns the t-scores for those specific stocks in that specific window.

- CalculatePvalues(window_num, ticker_list)
    - Takes in the window number and a list of tickers and returns the p-values for those specific stocks in that specific window.

Results:

Using the linear regression package on the following list of stocks from Jan. 01, 2020 to now.

tickers = [‘FB’, ’AMZN’, ‘AAPL’, ‘NFLX’, ‘GOOGL’]

start = ‘2020-01-01’
end = datetime.datetime.today()

Get all the relevant coefficients/values to perform linear regression (variance, covariance, slope of the line, etc.). Use a sample window of 30 days.

linearpackage = StockLinearRegression(tickers)
linreg_prices =

linearpackage.PullPrices(start, end)

linreg_window =

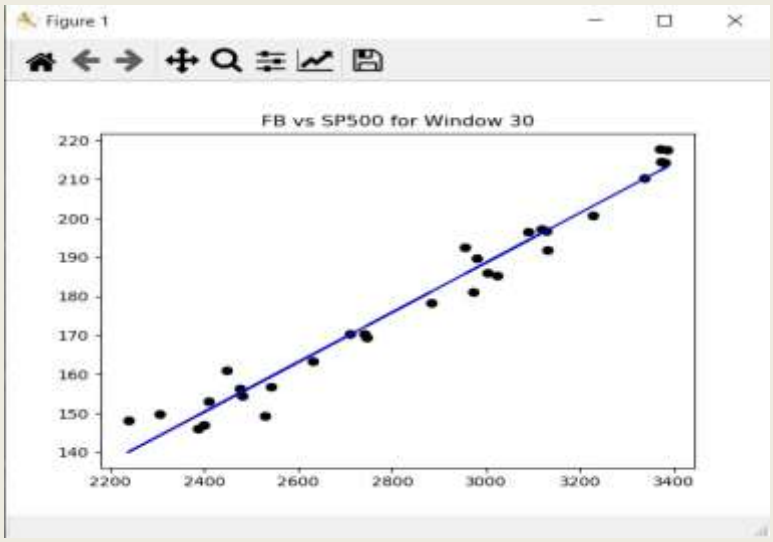
linearpackage.DefineWindow(30)

linreg_variance =
linearpackage.CalculateVaria
nce()
linreg_covariance =
linearpackage.CalculateCovar
iance()

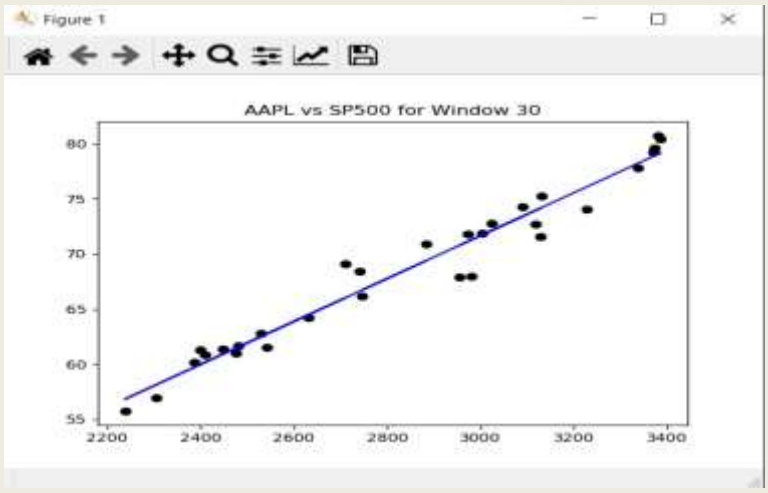
linreg_slopes =
linearpackage.CalculateSlopes()
linreg_intercepts =
linearpackage.CalculateIntercepts()

Plotting stock price vs S&P500 data (some examples):

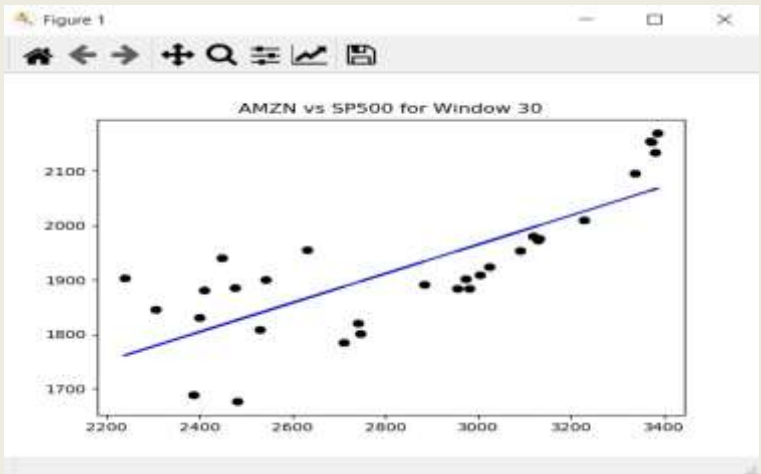
linearpackage.PlottingWindow(30, ‘FB’)



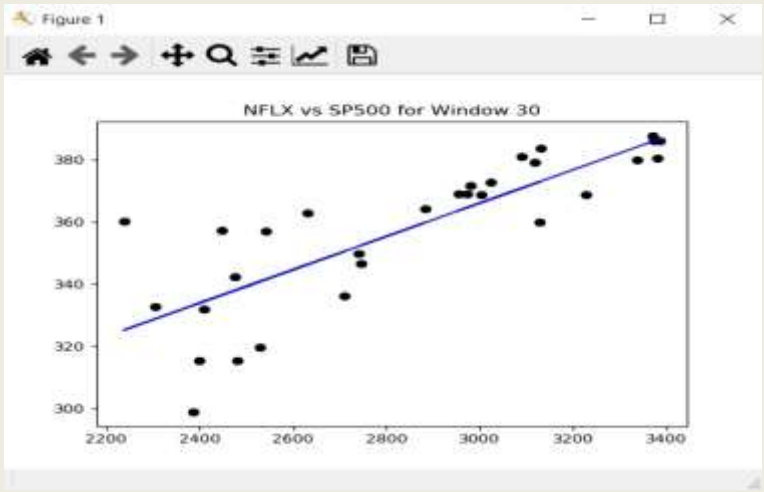
linearpackage.PlottingWindow(30, ‘AAPL’)



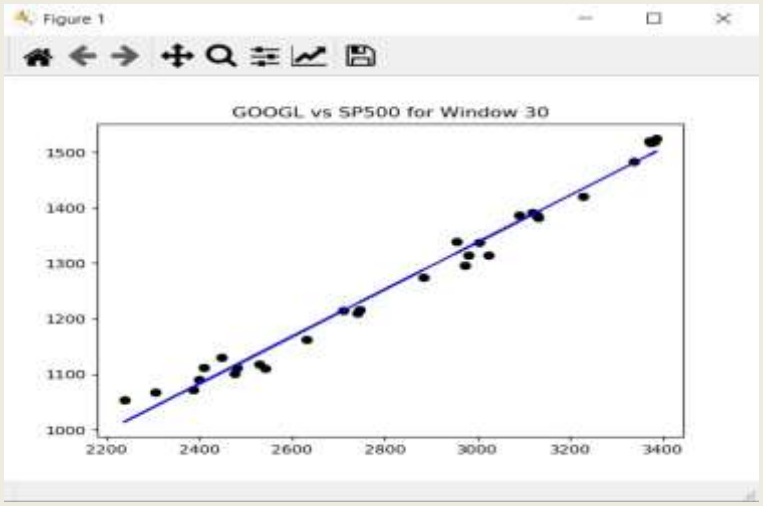
linearpackage.PlottingWindow(30, ‘AMZN’)



linearpackage.PlottingWindow(30, ‘NFLX’)



linearpackage.PlottingWindow(30, ‘GOOGL)



Get the relevant coefficients for each of the above examples

R-Squared Values:

```
linreg_rsquared = linearpackage.CalculateRsquared(
      170, ['FB', 'AMZN', 'AAPL', 'NFLX', 'GOOGL'])
```

FB	0.511075
AMZN	0.780132
AAPL	0.804812
NFLX	0.734545
GOOGL	0.588193

Linear Standard Error Co-eff Values:

```
linreg_stdcoefferr = linearpackage.CalculateStdcoeff(
     170, ['FB', 'AMZN', 'AAPL', 'NFLX', 'GOOGL'])
```

FB	8.026784
AMZN	0.562020
AAPL	8.411893
NFLX	7.407315
GOOGL	4.848872

T-Score:

```
linreg_tscores = linearpackage.CalculateTscores(
      170, ['FB', 'AMZN', 'AAPL', 'NFLX', 'GOOGL'])
```

FB	0.011936
AMZN	2.754950
AAPL	0.006224
NFLX	0.039566
GOOGL	0.112712

P-Values:

```
linreg_pvalues = linearpackage.CalculatePvalues(
    ['FB', 'AMZN', 'AAPL', 'NFLX', 'GOOGL'])
```

FB	0.495244
AMZN	0.003189
AAPL	0.497520
NFLX	0.484238
GOOGL	0.455183

Challenges and Roadblocks :

One of the main challenges that our group experienced when arranging this undertaking was deciding how to sample from the dataset and which index to use. We experimented with various size windows and indexes like DOWJONES, NASDAQ and S&P500. Then after running regression model on every index we decided to keep S&P500 as our index as it has all our subjected stocks.

To smooth out our work process, the group utilized Google Drive for bunch conceptualizing, documentation, and introduction work. As we started the first composition of the code, the group chose to utilize Jupyter Notebook as the IDE of the decision to work out the rationale and record the code scraps but then decided to stick with visual studio code. The rationale for the linear regression was first hard-coded to work with a particular stock to confirm that the code was utilitarian.

In spite of the fact that this permitted us to rapidly create and test our code, this implied that the majority of the code must be adjusted to acknowledge different contributions to each capacity including start date, end date, window, and so forth.

Given that pieces of the code have been dealt with by various colleagues, we worked with code cooperatively on zoom meetings.

Going Forward/Future Works:

One approach to make our package a stride further is actualize extra strategies that play out some sort of nonlinear regression. In a portion of our above models, similar to window 170 for AMZN, it gives the idea that some sort of nonlinear regression would be more qualified to demonstrate the conduct. At first glance, it seems like the relationship between the variables would be best described by a polynomial function.

The first step up from linear regression would be quadratic regression, and we attempted to develop a package to perform this. However due to time restrictions and considerably more convoluted formulas (relative to linear regression), we were not able to successfully develop this part of the package. The following formulas are used in the calculation of the best-fit line and correlation coefficient only.

Quadratic regression

(1) *mean* : $\bar{x} = \frac{\sum x_i}{n}$, $\bar{y} = \frac{\sum y_i}{n}$ $\bar{x^2} = \frac{\sum x_i^2}{n}$

(2) *trend line* : $y = A + Bx + Cx^2$

$$B = \frac{SxySx^2x^2 - Sx^2ySxx^2}{SxxSx^2x^2 - (Sxx^2)^2}$$

$$C = \frac{Sx^2ySxx - SxySxx^2}{SxxSx^2x^2 - (Sxx^2)^2}$$

$$A = \bar{y} - B\bar{x} - C\bar{x^2}$$

(3) *correlation coefficient* :

$$r = \sqrt{1 - \frac{\sum(y_i - (A + Bx_i + Cx_i^2))^2}{\sum(y_i - \bar{y})^2}}$$

$$S_{xx} = \sum(x_i - \bar{x})^2 = \sum x_i^2 - n \cdot \bar{x}^2$$

$$S_{xy} = \sum(x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - n \cdot \bar{x}\bar{y}$$

$$S_{xx^2} = \sum(x_i - \bar{x})(x_i^2 - \bar{x^2}) = \sum x_i^3 - n \cdot \bar{x}\bar{x^2}$$

$$S_{x^2x^2} = \sum(x_i^2 - \bar{x^2})^2 = \sum x_i^4 - n \cdot \bar{x^2}\bar{x^2}$$

$$S_{x^2y} = \sum(x_i^2 - \bar{x^2})(y_i - \bar{y}) = \sum x_i^2 y_i - n \cdot \bar{x^2}\bar{y}$$

Source: Keisan Online Calculator

As one can see, the process is more included and would be much more confounded when extended to cover test windows of different stock tickers.

This package may likewise be extended to incorporate relapse AI techniques. Notwithstanding making decisions utilizing just past qualities, we can likewise then grow to make forecasts about how the stock will act compared with the market later on.

One other thought we made was to incorporate a plotting technique that permits us to plot both the conduct of the S&P500 and a given stock for a given window in a solitary chart. This would take into account one more approach to evaluate stock conduct comparative with the market.

Sources/References

Bodie, Zvi, and Alan Marcus. “Index Models.” Investments, by Alex Kane, 11th ed., McGraw Hill Education, 2018, pp. 245–276.

"Quadratic Regression Calculator." High accuracy calculation for life or science.. N. p., 2019. Web. 11 Dec. 2020.
https://keisan.casio.com/exec/system/14059932254941

"Beta Coefficient - Learn How To Calculate Beta Coefficient." Corporate Finance Institute. N. p., 2019. Web. 11 Dec. 2020.
https://corporatefinanceinstitute.com/resources/knowledge/finance/beta-coefficient/

"T-Test Using Python And Numpy." *Medium*. N. p., 2017. Web. 11 Dec. 2020.
https://towardsdatascience.com/inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f

" Regression Slope Test ." Stattrek.com. N. p., 2019. Web. 11 Dec. 2020. https://stattrek.com/regression/slope-test.aspx

"PEP 8 -- Style Guide For Python Code." *Python.org*. N. p., 2019. Web. 11 Dec. 2020. https://www.python.org/dev/peps/pep-0008/

Linear Regression
https://en.wikipedia.org/wiki/Linear_regression