https://github.com/rfinegan1/FE-520_FinalProject
GitHub Repository: Checkpoints, Final Code, and Files

FINAL PROJECT

Ryan Finegan

FE-520 Python for Financial Applications

1. Introduction:

For this final project, I was inspired to look into major market indices and ETFs that track these indices. These ETFs produce superior returns to other asset classes, and they offer reduced volatility due to the holdings comprising of a large number of quality firms. This project was meant to demonstrate my knowledge of the python programming language with respect to financial applications. In the process of this concluding project, I used pandas, NumPy, matplotlib, datetime, Yahoo Finance and FRED APIs, Scikit-Learn, and TensorFlow with Keras to create supervised deep learning neural networks to solve regression and binary classification problems observing indices that are popular and obtain the purpose to summarize market segments in finance.

2. Motivation and Tasks:

I was motivated to forecast tick price and return data of market indices for three main reasons.

The first included the fact that knowing the direction of the general market is advantageous to anyone in finance because many assets are associated with the performance of these major indices, such as the S&P 500 or Dow Jones Industrial. Knowing the direction of these markets, could provide valuable information to outperform the market on entering and exiting positions.

Moreover, the idea of compounding was extremely interesting to me. Compounding daily or even weekly can increase returns relative to if you were to just hold the asset. With that said, trading a security that tracks the overall market, instead of holding, is generally criticized as most of its gains are realized in after-market hours. However, I believe that a great model can achieve extremely accurate predictions yielding the ability to enter and exit positions regarding a security aligned with an index to make more profits over time due to continuous compounding.

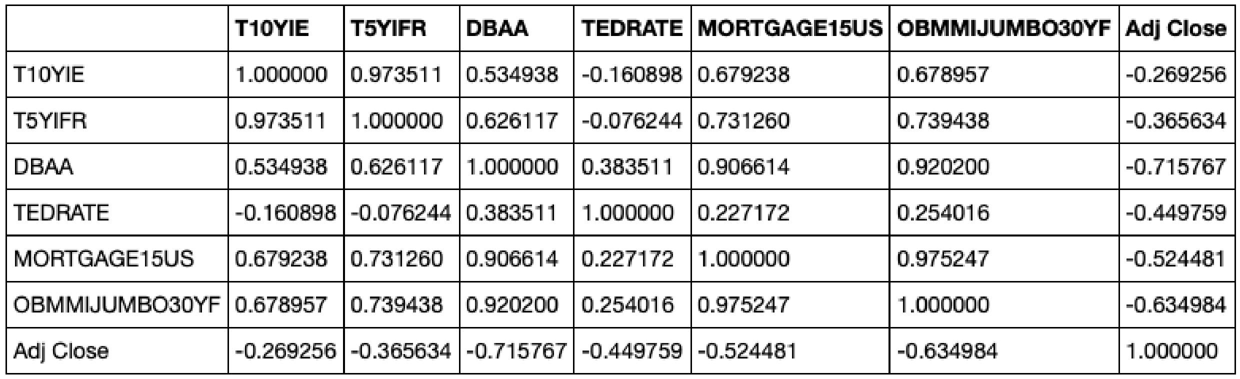
Lastly, indices were attractive because if the model failed at some point, the risk in your position would be negated because of the small volatility in these ETFs containing many firms that possess extreme upside potential.

3. Project:

The overall stock market is said to be driven by fundamental factors, technical factors, and market sentiment. Sentiment was my first choice; however, retrieving tweets from my Tweepy API was limited to the last 1000 tweets for a specific hashtag or cashtag. Due to the limited size of data and since I was observing diverse indices, I decided to obtain technical factors from the FRED database. These factors were macroeconomic statistics that portrayed inflation, interest rates, and economic growth. Another element influencing my feature selection was the daily nature of my model. I aspired to produce models that predicted tomorrow's tick price and return of the security. Therefore, I needed daily updated macroeconomic features that were popular among finance professionals.

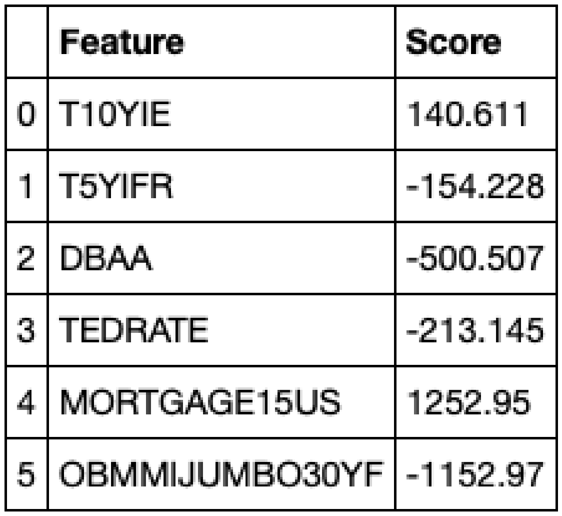
3a. Linear Regression Coefficients, Correlation Matrix, and Trial and Error:

Finding the correct features was the first significant task to this project. Since I narrowed it down to six popular macroeconomic tickers, I knew I would be able to perform trial and error until the model performed at its most accurate. The six initial features were recording US inflation rates, interest rates in the form of BBB corporate yields and TED Spread which is the spread between the three-month LIBOR and Treasury Bill, and mortgages such as 15-year and 30-year fixed mortgage rates. These features were updated daily, popular among FRED users, and represented the performance of the American economy. All of these features along with the S&P 500 was queried for using either Yahoo Finance or FRED API given the pandas-datareader library. A 10-year, daily history was collected and compared given a correlation matrix. The results are posted below:



	T10YIE	T5YIFR	DBAA	TEDRATE	MORTGAGE15US	OBMMIJUMBO30YF	Adj Close
T10YIE	1.000000	0.973511	0.534938	-0.160898	0.679238	0.678957	-0.269256
T5YIFR	0.973511	1.000000	0.626117	-0.076244	0.731260	0.739438	-0.365634
DBAA	0.534938	0.626117	1.000000	0.383511	0.906614	0.920200	-0.715767
TEDRATE	-0.160898	-0.076244	0.383511	1.000000	0.227172	0.254016	-0.449759
MORTGAGE15US	0.679238	0.731260	0.906614	0.227172	1.000000	0.975247	-0.524481
OBMMIJUMBO30YF	0.678957	0.739438	0.920200	0.254016	0.975247	1.000000	-0.634984
Adj Close	-0.269256	-0.365634	-0.715767	-0.449759	-0.524481	-0.634984	1.000000

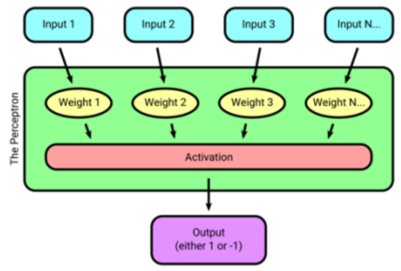
Following the correlation matrix, I used sklearn to run a linear regression with the six aforementioned macroeconomic indicators as features and the label being the tick price of the S&P 500 index. Using this package, once the model was fitted to the features and target, coefficients were extracted and observed for building the model in TensorFlow 2.x. The results of the linear regression are below.



	Feature	Score
0	T10YIE	140.611
1	T5YIFR	-154.228
2	DBAA	-500.507
3	TEDRATE	-213.145
4	MORTGAGE15US	1252.95
5	OBMMIJUMBO30YF	-1152.97

3b. Deep Neural Networks Overview - The Perceptron:

The perceptron is the most basic form of a neural network, and it will be used to explain the process of a simple neural network and beyond into my two deep neural networks which composed of a single hidden layer. At first, input data is applied into the perceptron. After, fluctuating weights are given to each input. A summation transpires from the product of the weights and inputs with an additional bias variable. An activation function is applied for non-linearity after the summation is complete. For my model predicting tomorrow's price, a leaky rectified linear unit was used for my model's non-linearity. For my binary model predicting the classification of index returns, a leaky rectified linear unit was used for the input and hidden layer, with a sigmoid activation function applied to the output layer. Sigmoid is often used in binary classification machine learning problems. Finally, an output is generated whether it be a regression or classification problem, and in both of these cases, one output unit was needed because both models predicted either the price or return of tomorrow, given the previous macroeconomic data that served as features. A model showcasing the perceptron is pasted below.



3c. Deep Neural Networks - My Models:

Due to slight commonalities of the indices collected, I knew a communal deep learning model might be advantageous. With that being said, I stuck to the same features and hyperparameters in both models only varying in activation functions, the loss function, scaler to normalize features, epochs, and the metrics to see how the model did. For the regression model predicting the price of the security, I selected a leaky rectified linear unit for each of the three layers. The loss function was mean squared error and the preprocessing was a min-max scaler. Epochs were nine to save time and the metric was mean absolute percentage error because I liked to see percentages when I was observing how well the model did given the relationship between the features and target price. For the binary classification model, I was predicting whether tomorrow's returns will be either positive (1) or negative (0). I only used five epochs, with a standard scaler, and the output layer activation function was sigmoid due to the binary classification problem. The loss function was binary cross entropy and my added metric was binary accuracy to observe how the model performed on the validation set. With that being said, the data split for the regression model was 80/10/10 for the training, testing and validation data respectively. It was 67/16.5/16.5 for the binary classification model. Before constructing the model, I was aware of the reliance to the overall health of the economy. Since it was forecasting many securities bundled into one, economic performance played a greater role than when just looking at one stock. Therefore, I believed possible communal drivers of these securities had to be daily updated economic indicators whether that be inflation, interest rates, or housing market statistics. The results of the linear regression and correlation matrix was meant to aid me in the process of feature selection. Thankfully, due to my small amount of pre-selected features, I saw that I was completely wrong in my intuition that the features identified by the linear regression coefficients and correlation matrix would be optimal when applied to a deep learning model. After trial and error of throwing features into the model, I saw the best performing features were that of the three-day moving averages of the Ten-Year Breakeven Inflation Rate, Five Year Forward Inflation Expectation Rate, and Baa Corporate Bond Yield, ultimately serving as an interest rate indicator. Model results were much better when all three of these features were there, with the inclusion of the two moving averages of the security, and this was shown in the really low mean absolute percentage errors on the validation dataset of each selected index.

The models had 3 layers: an input layer with fifty units, an output layer with one unit, and a hidden layer in the middle with twenty-five units. For the price prediction model, each layer had a leaky rectified linear unit activation function for non-linearity and He initializer for the exploding or vanishing gradient problem as the activation outputs pass forward through the deep learning network. The Leaky ReLU function formula is below.

$$\max(0.1x, x)$$

The binary classification model prediction returns of tomorrow used a sigmoid activation function in the output layer. This formula is shown below.

$$S(x) = \frac{1}{1 + e^{-x}}$$

A mean squared error loss function was applied in the price prediction model because that is most common for regression problems. The mean squared error loss is the sum of squared distances between the historical target and the model forecasted variables. For the binary model, the loss function was binary cross entropy. This formula is shown below.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

A loss function is a procedure evaluating how well the model is executing. Also, observing the validation data's performance can help with identifying if the model is overfitting on the training data. Optimizers are used in deep neural networks to transform attributes like weights and learning rates to minimize losses. Both models used an Adam optimizer. This formula is shown below.

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Finally, for the regression price prediction model, A min-max scaler was applied to normalize the feature data between datapoints one and zero.

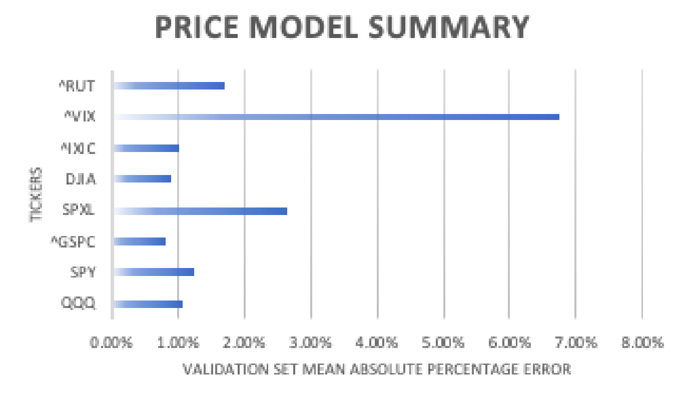
$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

For the binary model predicting returns, a standard scaler was applied. This formula can be referenced below.

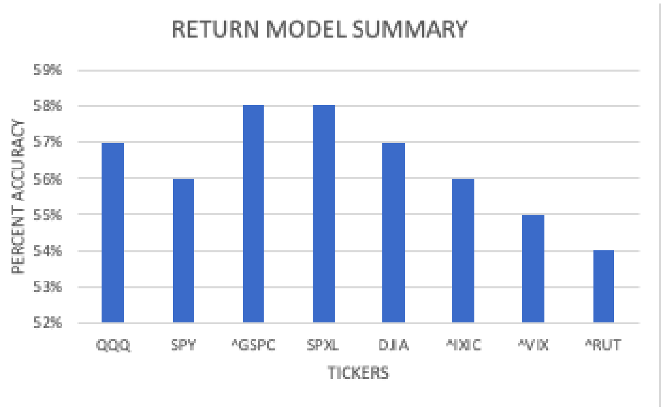
$$z = \frac{x - \mu}{\sigma}$$

3d. Regression Price and Binary Classification Returns Model Performance:

The regression model only had nine epochs with the design to see if a few indices indicated signs of strong performance of price prediction. All of the eight indices experienced a mean absolute percentage error below 7%. The S&P 500 had the lowest MAPE with the NASDAQ Composite and Dow Jones Industrial following behind closely.



The binary model did not perform as well as I would have liked. Each index experienced a binary accuracy around 55% which is approximately the same as guessing. This model was the most time consuming as hyperparameter tuning and optimization was the most tedious. The most optimal model ended up being very similar to that of the price regression model. I was unsure of whether to change the features to binary variables as well. For this I would have displayed a one if the macroeconomic indicator was greater than the previous quantity applied to that indicator. I was unaware of how to best go about a binary classification problem and given the amount of time I was given, I did not see reasoning in wasting my time on a task that deemed already time consuming and unwavering to produce quality results. The binary accuracy performance results of the binary return model is displayed below.



4. Conclusion:

In essence, both of these models are not accurate enough to be used to perform well on their own. However, the price regression deep learning model shows great signs if certain features were tweaked and added. I believe if there's a dataset of tweets with a certain hashtag or cashtag like "$SPY" that has a history of at least ten years, the model would perform significantly better. On the other hand, both of these models are for my knowledge and practice with applying python to the world of finance. I am new to machine learning and TensorFlow, but this project was great for me to learn the tools as well as the direction of different models as they are applied to financial problems.

5. Future Work:

As mentioned many times previously in this paper and in my PowerPoint presentation, I look forward to attaining a dataset with enough market sentiment data to be an additional feature in the model. Market sentiment is a relatively new data type used in the capital markets, and I believe this data, especially if it's taken from Twitter, will serve well when matched with moving averages of both the target security and daily macroeconomic indicators.