# M.S. Data Analytics - Executive Summary (Recap)

## Research Question & Statement of the Problem

**Research Question:** Using historical market stock prices and technical indicators (RSI, MACD, MFI), how accurately can a neural network model, specifically an LSTM-based Recurrent Neural Network, predict stock price movements after the occurrence of a bullish candlestick pattern (e.g., 1 day, 3 days, 5 days, 10 days, and 15 days afterwards)? Additionally, how does the performance differ when using binary classification versus regression for price prediction?

**Statement of the Problem**

The idea that I had in mind is that by answering this research question, I will be able to help out short-term stock traders. Specifically, there is a term called, swing trading, which involves holding stocks for several days to weeks, capitalizing on short-term price movements. By using historical market data, technical indicators like RSI, MACD, and MFI, and an LSTM, traders could identify whether a bullish candlestick pattern will likely lead to price increases in the next 1, 3, 5, 10, or 15 days. This ability to predict post-pattern price movements could help traders make more informed entry and exit decisions for optimal returns.

If the model can accurately predict price movements after bullish patterns, it would allow swing traders to enter positions with a higher probability of profit. Knowing the expected future price movement gives traders an edge in deciding whether to buy a stock which results in a higher likelihood of success.

I chose to implement a Long short-term memory (LSTM), which is a type of recurrent neural network (RNN) for this research question. The reason for this is because the key feature of an RNN is its ability to maintain a hidden state that acts as memory, enabling the network to remember past information and use it to influence predictions for future inputs. This makes RNNs ideal for tasks such as time series problems, where the sequence and context of data are crucial. LSTMs, which are an advanced form of RNNs, overcome common issues like vanishing gradients and are expected to improve the accuracy of our predictions. *To understand the issue of a vanishing gradient and how an LSTM works,* *please refer to my extended executive summary.*

Lastly, LSTMs can be used for classification or regression tasks. Classification is used to predict a class label. Regression is used to predict a continuous value. In the case of my research question, although classification and regression are generally used for different purposes, my goal is to compare them apples to apples using accuracy as the evaluation metric. In the case of regression, which outputs a continuous value, if this continuous value is greater than a predefined threshold, it could be classified as a positive outcome, while values below or equal to the threshold could be considered negative outcomes. This approach allows for a direct comparison of classification and regression models by transforming regression outputs into a binary decision, thereby evaluating both models based on accuracy in predicting price movements.

## Hypothesis

After performing our data analysis, we should be able to conclude either the null or alternative hypothesis. The null hypothesis is the default assumption, stating that our independent variables (past stock price [low, high, open, close], RSI, MACD, MFI) have no effect, no relationship, or cause no change, to our dependent variable (future stock price).

**Null Hypothesis:** The LSTM-based Recurrent Neural Network model does not significantly predict stock price movements (up or down) better than random chance when using historical market stock prices and technical indicators (RSI, MACD, MFI) for prediction.

**Alternative Hypothesis:** The LSTM-based Recurrent Neural Network model significantly predicts stock price movements (up or down) better than random chance when using historical market stock prices and technical indicators (RSI, MACD, MFI) for prediction.

## Introduction (Summary of Data-Analysis Process)

To answer my research question, I initially set to out to identify which candlestick patterns that I can use for my analysis. There were five candlestick patterns that I identified; all of these candlestick patterns signal bullish movements. This means that these patterns are used by technical traders to signal potential upward price movements in the near future.

These five candlestick patterns were chosen for my analysis because they were found to be among the most commonly used by technical traders:

1. Hammer
2. Inverted Hammer
3. Bullish Engulfing
4. Bullish Harami
5. Three White Soldiers

When defining my candlestick patterns, I had very specific criteria on how each of those patterns were defined. This approach ensures consistency and a systematic method for recognizing patterns, which leads to more reliable and repeatable results when analyzing trends. While the specifications may differ and agree in certain details (with some other examples found online), they still encompass the core definition of each candlestick pattern. *Please refer to my extended executive summary on how exactly I defined each candlestick pattern.*

Additionally, to provide a point of comparison for these candlestick patterns, I introduced a **'Random'** pattern, which represents randomly selected days in the stock market.

I then chose the stock ticker “SPY” for my analysis because it encompasses many stocks and represents a broad market index, making it a reliable proxy for overall market performance. Specifically, this ticker, represents the S&P 500 index which includes 500 of the largest publicly traded companies in the U.S., making the “SPY” ETF a way for investors to gain exposure to the overall U.S. stock market. Once I decided on the stock ticker, using my definitions for each candlestick pattern, I identified those five patterns.

Using stock market data starting from, 1/1/2000, and ending, 2/14/2025, there were 6,318 daily candles that were available. From those candles, the hammer pattern occurred-70 times (1.108%), the inverted hammer-52 times (0.823%), the bullish engulfing-24 times (0.380%), the bullish harami-29 times (0.459%), and the three white soldiers-34 times (0.538%).

Although the sample sizes were small for all candlestick patterns, I decided to not include the bullish engulfing, bullish harami, and three white soldiers patterns in my analysis because I felt that that their sample sizes were too small.

I then calculated the independent variables for our stock, including RSI, MACD (and its Signal line, which is a component of MACD), and MFI. Additionally, I normalized the stock prices two different ways. First, using a log transformation. Second, with a built-in package from ‘Sklearn,’ it enabled me to normalize the data by scaling it to a standard range between 0 and 1.

At this stage, to input our data into an LSTM for training the machine learning model, I needed to format the metrics into sequences, as LSTMs require data in that structure. I decided to use 30-day sequences, as I believe this timeframe offers sufficient historical data to capture short-term price movements following the candlestick pattern.

I aimed to experiment with a variety of different combinations of independent variables. For example, here is how the shape of a sequence would be when inputted into my LSTM model and the different combinations of independent variables that I tried. Altogether, there were 15 different combinations of independent variables:

**(69, 30, 4)** – 69 observations, 30 time-steps, 4 features (independent variables) per time-step

1. Stock price open, Stock price close, Stock price high, Stock price low
2. log(Stock price open), log(Stock price close), log(Stock price high), log(Stock price low)]
3. min-max(Stock price open), min-max(Stock price close), min-max(Stock price high), min-max(Stock price low)

**(69, 30, 5)** – 69 observations, 30 time-steps, 5 features per time-step

1. Stock price open, Stock price close, Stock price high, Stock price low, RSI
2. log(Stock price open), log(Stock price close), log(Stock price high), log(Stock price low), RSI
3. min-max(Stock price open), min-max(Stock price close), min-max(Stock price high), min-max(Stock price low), RSI
4. Stock price open, Stock price close, Stock price high, Stock price low, MFI
5. log(Stock price open), log(Stock price close), log(Stock price high), log(Stock price low), MFI
6. min-max(Stock price open), min-max(Stock price close), min-max(Stock price high), min-max(Stock price low), MFI

**(69, 30, 6)** – 69 observations, 30 time-steps, 6 features per time-step

1. Stock price open, Stock price close, Stock price high, Stock price low, MACD, Signal Line
2. log(Stock price open), log(Stock price close), log(Stock price high), log(Stock price low), MACD, Signal Line
3. min-max(Stock price open), min-max(Stock price close), min-max(Stock price high), min-max(Stock price low), MACD, Signal Line

**(69, 30, 8)** – 69 observations, 30 time-steps, 8 features per time-step

1. Stock price open, Stock price close, Stock price high, Stock price low], RSI, MFI, MACD, Signal Line
2. log(Stock price open), log(Stock price close), log(Stock price high), log(Stock price low), RSI, MFI, MACD, Signal Line
3. min-max(Stock price open), min-max(Stock price close), min-max(Stock price high), min-max(Stock price low), RSI, MFI, MACD, Signal Line

***Note****: That later in this document I will refer to the first combination of independent variables as independent\_array #1, the second combination as independent\_array #2,… the 15th combination as independent\_array #15*

While creating the 30-day sequence data for each combination of independent variables, I also generated the dependent variable set. The dependent variable was reliant on two different parameters for my **binary classification task**:

1. Determining the number of days after the candlestick pattern to use for the dependent variable:
   1. 1 day
   2. 3 days
   3. 5 days
   4. 10 days
   5. 15 days
2. Defining the percent increase from the current price to classify as a positive outcome:
   1. 1.00 = For example, if we are using one day after the conclusion of the candlestick pattern, the closing price will be labeled as a positive class if it is greater than the closing price at the end of the pattern. If the price is the same or lower, it will be classified as a negative class. If $100 was the closing price at the end of the pattern, the closing price one day out has to be greater than $100 to be labeled as a positive class.
   2. 1.01 = For example, if we are using one day after the conclusion of the candlestick pattern, the closing price will be labeled as a positive class if it is 1% greater than the closing price at the end of the pattern. If the price is the same or lower, it will be classified as a negative class. If $100 was the closing price at the end of the pattern, the closing price one day out has to be greater than $101 to be labeled as a positive class.
   3. 1.02 = For example, if we are using one day after the conclusion of the candlestick pattern, the closing price will be labeled as a positive class if it is 1% greater than the closing price at the end of the pattern. If the price is the same or lower, it will be classified as a negative class. If $100 was the closing price at the end of the pattern, the closing price one day out has to be greater than $102 to be labeled as a positive class.

This means that I also have 15 combinations of dependent variables:

1. 1 day out, 1.00
2. 1 day out, 1.01
3. 1 day out, 1.02
4. 3 days out, 1.00
5. 3 days out, 1.01
6. 3 days out, 1.02
7. 5 days out, 1.00
8. 5 days out, 1.01
9. 5 days out, 1.02
10. 10 days out, 1.00
11. 10 days out, 1.01
12. 10 days out, 1.02
13. 15 days out, 1.00
14. 15 days out, 1.01
15. 15 days out, 1.02

**For my regression task,** the 15 different combinations of independent variables will remain consistent. However, when training, the dependent variable will be determined solely by the number of days after the candlestick pattern. This is because, rather than predicting a class label, we are forecasting a continuous value. For instance, if the "days out" parameter is set to 1 day, the dependent variable will represent the stock price close one day after the pattern concludes. The "percent increase" parameter will only be used after training the LSTM model and obtaining its predictions to evaluate the results.

As mentioned earlier, I want to reiterate that, although classification and regression are typically used for different purposes in my research, my goal is to compare them on an equal basis using accuracy as the evaluation metric. In the case of regression, which produces a continuous value, if the predicted value exceeds a predefined threshold (which is where the “percent increase” parameter comes in), it can be classified as a positive outcome, while values below or equal to the threshold would be considered negative outcomes. This method enables a direct comparison between classification and regression models by converting regression results into binary decisions, allowing both models to be evaluated based on their accuracy in predicting price movements.

**I then constructed my LSTM model for classification.**

A screenshot of a computer code

AI-generated content may be incorrect.

This model is designed for binary classification tasks, where the goal is to predict one of two possible outcomes ("Yes" or "No"). The model begins with an LSTM layer of 128 units, using the "tanh" activation function to capture patterns in the sequential input data. The return\_sequences=True means this layer outputs sequences, which are passed on to the next LSTM layer. A dropout layer is added to reduce overfitting by randomly "dropping" some of the units during training.

The second LSTM layer, with 64 units, processes the sequence further, but without returning sequences (return\_sequences=False). This makes the output a single vector, which is then passed through another dropout layer. The model ends with a dense output layer, which has one unit with a sigmoid activation function, giving a probability between 0 and 1 for binary classification. The model is compiled using the Adam optimizer, a commonly used optimization algorithm, with binary cross-entropy as the loss function, as this is suitable for binary classification problems. The performance of the model is evaluated using accuracy as the metric.

With this network architecture, I implemented stratified K-fold cross validation. I set the value of ‘K’ equal to ‘5’ when training my model. Stratified K-Fold Cross Validation is often preferred over regular K-Fold in classification problems because it ensures that each fold of the data has a similar distribution of the target classes. This is particularly important when dealing with imbalanced datasets where some classes may be underrepresented. In regular K-Fold cross-validation, the data is randomly split, which could result in some folds having disproportionately many samples from one class and too few from another. This can lead to biased model performance estimates, as the model might not be exposed to enough of the minority class to learn effectively.

In contrast, Stratified K-Fold ensures that each fold contains roughly the same proportion of each class as in the original dataset. This helps the model train and validate on a more balanced representation of the target variable, leading to more reliable and generalizable performance metrics. Stratified K-Fold is particularly beneficial for classification tasks where the goal is to maintain fairness in model evaluation, and it can help prevent skewed results caused by class imbalances.

In my case, when splitting my dataset, I want to make sure I have the same ratio of positive and negative classes in each fold as in the original dataset. This will ensure that my model is consistently evaluated on balanced data and help improve its ability to predict both classes effectively.

For each 15 different combinations of independent variables, I performed Stratified 5-fold cross validation for each of the 15 different combinations of dependent variables. For example, using training data associated with the inverted hammer pattern, here is the independent variable combination ‘independent\_array #15’ and its reported test accuracy scores after performing stratified 5-fold cross validation:

A screenshot of a computer

AI-generated content may be incorrect.

This means that for each candlestick pattern – random, hammer, inverted hammer – there are going to be 225 (15 \* 15 = 225), separate models trained using stratified 5-fold cross validation. Each model will report it’s best accuracy on the test set, and it’s average accuracy on the test set.

**I then constructed my LSTM model for regression.**

A screenshot of a computer code

AI-generated content may be incorrect.

This model consists of two LSTM layers, each followed by a Dropout layer to reduce the risk of overfitting. The first LSTM layer has 128 units and returns sequences of data, allowing the next LSTM layer to process the sequence. The second LSTM layer has 64 units and does not return sequences. The final layer is a Dense layer with a single neuron and a linear activation function, which outputs a continuous value suitable for regression tasks. The model is compiled using the Adam optimizer and mean squared error (MSE) loss function, which is commonly used for regression.

As mentioned previously, a regression model is designed to predict continuous values, not for classification tasks. However, I needed a way to compare the performance of both my regression and classification models. To do this, I evaluated the predicted values from my regression model using the following approach:

1. If the predicted closing price is greater than the closing price of the last candle in the identified candlestick pattern, and the actual closing price (from the test set) is also greater than the last candle’s closing price, this is considered a correct prediction.
2. If the predicted closing price is less than or equal to the closing price of the last candle in the identified candlestick pattern, and the actual closing price (from the test set) is also less than or equal to the last candle’s closing price, this is also considered a correct prediction.
3. I then count the number of correct predictions from both cases (steps 1 and 2), and divide this by the total number of observations in the test set to calculate the accuracy.

Similar to training my classification model, I also implement k-fold cross validation. However, instead of stratified k-fold cross validation, I implemented the standard k-fold cross validation as it is recommended for regression tasks. Again, I set the ‘K’ value to equal ‘5’.

Again, similar to the classification problem, this means that for each candlestick pattern – random, hammer, inverted hammer – there are going to be 225 (15 \* 15 = 225), separate models trained using regular 5-fold cross validation. Each model will report it’s best accuracy on the test set, and it’s average accuracy on the test set.

**After reviewing all of the training result data and respective accuracy scores – comparing both classification and regression – I have determined that the classification model performed better and more consistent!!!**

**I also found that independent array #15 was the best performing set of independent variables for my classification model.**

**At this point,** I had enough data to answer my research question, however, now I went to increase the number of randomly generated sequences in order to make my model more adaptable for use on any given day as the hammer and inverted candlestick patterns have around a 1% occurrence rate. My hope was that even when a true candlestick pattern is not present, the model could still make reliable predictions for future closing prices.

With that being said, to increase the accuracy of my predictions for when a ‘Random’ pattern is used as the training data, I increased the number of observations and implemented ensemble learning, while still applying 5-fold stratified cross-validation.

I created and trained 9 models, all of which have the same architecture and trained them all on my best performing set of independent variables, independent array #15. Each of these 9 models utilized Stratified 5-fold cross validation, and I saved the best weights for each model during training using a ModelCheckpoint callback - the epoch with the highest validation accuracy across all five folds will have its model weights saved.

After the models have been trained and their best weights saved, I loaded these models and combined their predictions using an ensemble approach. Specifically, I will take each model and output its predictions; I used a voting-based system where if any 5 of the 9 models (the majority) predict the class as a positive class, then its final prediction will be classified as positive, and if less than 5 models predict it as positive, it will be classified as negative. This ‘ensemble’ approach ensured that the final decision is based on the collective judgment of the majority of the models, reducing the likelihood of errors from individual models.

Lastly, the accuracy of the ensemble model was evaluated on a separate test dataset to assess the effectiveness of this combined approach. This ensemble method aimed to leverage the strengths of each individual model and provide a more robust prediction.

## Outline of the Findings

To restate my research question:

*Using historical market stock prices and technical indicators (RSI, MACD, MFI), how accurately can a neural network model, specifically an LSTM-based Recurrent Neural Network, predict stock price movements after the occurrence of a bullish candlestick pattern (e.g., 1 day, 3 days, 5 days, 10 days, and 15 days afterwards)? Additionally, how does the performance differ when using binary classification versus regression for price prediction?*

To directly answer the first question, **my LSTM-based Recurrent Neural Network can accurately predict stock price movements following the occurrence of a bullish candlestick pattern**. Specifically, it performs about 9-10% better in predicting the outcome compared to when a random occurrence is used. By "random occurrence," I mean a randomly generated set of 30-day sequences that do not depend on the presence of a candlestick pattern on the 30th day of the sequence.

To directly answer the second question, **the performance is more consistent and predictable while using a classification model in predicting price fluctuations rather than a regression model**.

Below, I will answer in high detail the first and second part of the research question and how I came to each conclusion.

**Answering the second part of my research question:**

To answer the second part of my research question, I conclude that the LSTM classification model was the better model overall for my purposes of this research project. The reason for this is as follows:

When comparing the results from the classification and regression models, it’s easier to identify the most effective independent variables for the classification model. It seems that the top-performing independent variables for the regression models were somewhat random. This is because, for the classification model, independent array #15 yielded the best results, showing the highest average and maximum accuracy scores across different combinations of model parameters.

These model parameters were used to define the dependent variable. Specifically, I had two key parameters when training the models: the number of days ahead from the last closing price in the 30-day sequence that I am predicting, and the percentage change from the last closing price of the 30-day sequence (e.g., whether the closing price on the specified future day is greater than, 1% higher than, or 2% higher than the last closing price on the 30th day of the sequence).

The classification results indicate that when the candlestick pattern is set to "Random" (**I am using the "Random" pattern because the actual candlestick patterns have too small a data size**) — meaning that random 30-day sequences are generated without relying on a specific candlestick pattern — independent array #15 produces the best outcomes, achieving the top spot 7 times. Independent array #14 follows closely in second place, with 6 occurrences at the top for yielding the highest average of maximum and average accuracy scores. I selected the candlestick pattern to "Random" because I wanted to analyze a larger sample size. Both arrays use all available independent variables (open, close, low, high, RSI, MFI, MACD, Signal Line), but there is a difference in how they process the data. Independent array #14 applies a log transformation to the open, close, low, and high prices, while independent array #15 uses Sklearn’s scaler function to normalize these prices. The classification results are as shown below:

* The classification results table is self-explanatory, but to clarify:
  + "Most Frequent Class": This represents the class (negative or positive) that occurs more frequently within the entire dependent variable dataset.
  + "Highest Frequency": This is calculated as the ratio of the "Most Frequent Class" frequency to the "Total Observations" (i.e., Most Frequent Class / Total Observations). This metric is used to evaluate whether the model's accuracy scores are better than simply predicting the most frequent class.
  + "Weighted Score": The weighted score is calculated as the average of the "Best Accuracy" and "Avg Accuracy" (i.e., (Best Accuracy + Avg Accuracy) / 2). The purpose of this metric is to account for the possibility that the "Best Accuracy" might be an outlier or lucky result, and by combining it with the "Avg Accuracy," we get a more balanced measure.

A screenshot of a computer screen

AI-generated content may be incorrect.

The regression results below indicate that when the candlestick pattern is set to "Random" — meaning that random 30-day sequences are generated without relying on a specific candlestick pattern — independent array #1 produces the best outcomes, achieving the top spot 5 times. This is interesting because independent array #1 solely includes price action (open, close, high, low) independent variables. These variables are also not normalized. The classification results are as shown below:

* The regression results table is as follows:

A screenshot of a computer screen

AI-generated content may be incorrect.

Looking at the results, it seems that both the classification and regression models outperform simply predicting the most frequently occurring class. To determine which model performs better—classification or regression—I will calculate the difference between the 'Weighted Score' and 'Highest Frequency' for each entry in the table, and then sum these differences across all observations. The model with the highest total of these differences will be considered the best performing, as it indicates that the model is yielding a greater improvement over the baseline of predicting the most frequent class, effectively demonstrating stronger overall performance.

|  |  |  |
| --- | --- | --- |
| Classification\_Highest\_frequency | Classification\_Weighted\_score | Classification\_Differences |
| 0.552764 | 0.718173 | 0.165409271 |
| 0.829146 | 0.866455 | 0.037309385 |
| 0.954774 | 0.964615 | 0.009841511 |
| 0.552764 | 0.719744 | 0.166979766 |
| 0.693467 | 0.759769 | 0.066301903 |
| 0.884422 | 0.907756 | 0.023334303 |
| 0.61809 | 0.736128 | 0.118037753 |
| 0.613065 | 0.723256 | 0.110191089 |
| 0.849246 | 0.90084 | 0.051593522 |
| 0.643216 | 0.756333 | 0.113117264 |
| 0.527638 | 0.716378 | 0.188740022 |
| 0.693467 | 0.780244 | 0.086776257 |
| 0.633166 | 0.734385 | 0.101218792 |
| 0.502513 | 0.706859 | 0.204346419 |
| 0.638191 | 0.763301 | 0.125110334 |

* The resulting sum of the differences for the classification table is: 1.568307591

|  |  |  |
| --- | --- | --- |
| Regression\_Highest\_frequency | Regression\_Weighted\_score | Regression\_Differences |
| 0.552764 | 0.732692 | 0.179928489 |
| 0.829146 | 0.876731 | 0.047585041 |
| 0.954774 | 0.977244 | 0.02246972 |
| 0.552764 | 0.608782 | 0.056018232 |
| 0.693467 | 0.816346 | 0.122878817 |
| 0.884422 | 0.917051 | 0.032629171 |
| 0.61809 | 0.678846 | 0.060755702 |
| 0.613065 | 0.656538 | 0.043473135 |
| 0.849246 | 0.887051 | 0.037805051 |
| 0.643216 | 0.653974 | 0.010758279 |
| 0.527638 | 0.666731 | 0.139092578 |
| 0.693467 | 0.846731 | 0.153263433 |
| 0.633166 | 0.624103 | -0.009063265 |
| 0.502513 | 0.796923 | 0.294410514 |
| 0.638191 | 0.872244 | 0.234052635 |

* The resulting sum of the differences for the regression table is: 1.426057531

To conclude, the classification model emerged as the best performing model overall. However, it is important to acknowledge that the regression model outperformed the classification model in several specific parameter combinations. Despite this, I believe the classification model is the superior choice for this task due to its greater consistency. The best performing independent variables for the classification model consistently yield high accuracy scores (independent array #15), regardless of the parameter combinations (e.g., days out and percent increase).

In contrast, it is more challenging to identify which independent variables in the regression model contribute to its accuracy, as the performance tends to fluctuate more significantly based on different parameter settings. This variability in the regression model’s performance makes it less predictable and harder to interpret, whereas the classification model’s stability provides a more reliable foundation for decision-making.

**Lastly, and perhaps most importantly, while we are not analyzing a specific candlestick pattern here, but instead working with a randomly generated occurrence that simulates the presence of random 30-day sequences, we can clearly observe that our trained models—whether for classification or regression—outperform the "Highest Frequency" column. This metric is used to evaluate the accuracy of predicting the most frequent class. When compared to the "Weighted Score" column—representing the accuracy score produced by our model—we consistently find that the "Weighted Score" is higher in nearly all cases (and sometimes a lot more), regardless of the parameter combinations used. This shows us the efficacy of our model.**

**Answering the first part of my research question:**

Since I have decided that classification was the better performing model with independent array #15 being the best combination of indepdendent variables, I am going to use that model with that set of independent variables to answer this question. I have decided to not use the bullish engulfing, bullish harami, and three white soldiers patterns for this analysis because of their small sample size. Although the hammer and inverted patterns had small sample sizes as well, they were a bit larger of a sample size than the other three.

Below I am going to compare all three results: for a randomly generated occurrence, the hammer pattern occurrence, and the inverted hammer occurrence.

* Random Occurrence

A screenshot of a computer

AI-generated content may be incorrect.

* Hammer Pattern Occurrence

A screenshot of a computer

AI-generated content may be incorrect.

* Inverted Hammer Occurrence

A screenshot of a computer

AI-generated content may be incorrect.

Based on the tables above, it appears that the model returns higher accuracy scores when trained on candlestick patterns. For example, for each table if you take the sum of the "Classification Highest Frequency" and the "Classification Weighted Score" columns and subtract them this will give us the distance of our weighted accuracy to predicting the majority class. For each of these tables, the values are as follows:

* Random Occurrence: SUM(Classification\_Weighted\_score) - SUM(Classification\_Highest\_frequency) = 1.411378076
* Hammer Occurrence: SUM(Classification\_Weighted\_score) - SUM(Classification\_Highest\_frequency) = 2.815309199
* Inverted Hammer Occurrence: SUM(Classification\_Weighted\_score) - SUM(Classification\_Highest\_frequency) = 2.7458251

Now, if I take each of these values and divide them by 15 (which is the total number of parameter combinations), I get the average weighted average accuracy scores across all 15 parameter combinations.

* Random Occurrence: 1.411378076 / 15 = 0.09409187173
* Hammer Occurrence: 2.815309199 / 15 = 0.18768727993
* Inverted Hammer Occurrence: 2.7458251 / 15 = 0.18305500666

**This shows that, for each parameter combination where a true candlestick pattern is present, the model performs approximately 9-10% better in predicting the outcome compared to when a random occurrence is used.**

It is possible that because the total number of observations for random occurrences was set higher than for the hammer and inverted hammer patterns (which are fixed values), the model may have struggled to learn the patterns effectively due to the larger data set for random occurrences. However, upon reviewing the results, it seems that when a true candlestick pattern is present, the model is able to predict the future closing price with significantly higher accuracy compared to when a candlestick pattern is not present or not likely to be present (which represents the randomly generated occurrence).

**Building off my research question**

Because there was a small sample size for my true candlestick patterns, I was a little disappointed. The reason being is that I wanted to deploy my model for real-world scenarios, especially for swing trading, where accurate predictions based on candlestick patterns could lead to more informed and timely trading decisions. However, as I have shown although I have received high accuracy scores, the frequency for these patterns are quite rare, occurring once every hundred days, or about a 1% occurrence.

Later in this document, I increased the number of randomly generated sequences in order to make my model more adaptable for use on any given day. The goal was to train the model on over 2,000 randomly generated sequences, hoping that even when a true candlestick pattern is not present, the model could still make reliable predictions for future closing prices.

Through this approach, I found that by increasing the sample size for training, using stratified 5-fold cross-validation, and incorporating ensemble learning methods, my model was able to deliver reliable results. These results were significantly more accurate than simply predicting the majority class. For these predictions, I used a single parameter combination: the closing price 10 days out, with a price increase greater than 1% on that day to be considered a positive class. I used independent array #15 as the basis for my independent variables which has 8 different features (Normalized\_close, Normalized\_open, Normalized\_low, Normalized\_high, RSI, MFI, MACD, Signal Line).

The results were as follows:

* Number of total observations in entire dependent array dataset: 2188
* Number of actual false labels in entire dependent array dataset: 1230
* Number of actual true labels in entire dependent array dataset: 958
* Number of total observations in test dataset: 438
* Number of incorrect predictions on test dataset: 137
* Number of correct predictions on test dataset: 301
* Prediction accuracy on test dataset: 68.72%
* Number of actual false labels in test dataset: 256
* Number of actual true labels test dataset: 182
* Majority class label percent in test dataset: 58.45%

The results of using ensemble methods are significant as we can see the accuracy scores. Based on this data sample, if we always predicted that the future closing price would be false, we would be correct 1230 out of 2188 times or 56.22% of the time. However, our model performs better, as our model has 301 correct predictions out of 438 observations or a correct prediction rate of 68.72%. Also, when looking at the actual false labels in the test dataset, which is the majority class, we see 256 observations; 256 out of 438 is 58.45%. Our model clearly performs better when compared against the actual labels in the entire dependent array (dataset used for training/validation) dataset and when compared against the actual labels in the test dataset.

The results are significant, as we can outperform the expected market outcome (which assumes predicting the future price as false) by around 10% with this parameter combination when comparing against the actual values of the test set labels. This demonstrates the effectiveness of the model in identifying patterns and making predictions that exceed a baseline strategy of predicting no price increase of greater than 1% over the next ten days (which was the parameter combination I used for this ensemble model (pct\_increase=1.01)).

I have demonstrated the implementation of the ensemble learning method (with stratified 5-fold cross-validation used to train these models) using the following parameters: The dependent variable represents the closing price 10 days in the future. The label for the dependent variable is assigned as 0 (false) if the future closing price is less than or equal to a 1% increase from the closing price of the last identified candle. It is labeled as 1 (true) if the future closing price is greater than a 1% increase from the closing price of the last identified candle.

I will run this implementation multiple times through multiple parameter combinations using independent array #15 as mentioned before, I found it to be the best performing combination of independent variables. I will run this separately outside of this document, for the submission for the next part of this project. Specifically, I will build a web application which will allow a user to easily implement this ensemble model, using the same independent variables as found on independent array #15.

So, to answer the last part on the WGU grading rubric in which I am to propose a directions or approach for future study of the data set. The main thing I want to do is train an ensemble model for other stock tickers rather than just 'SPY' as shown in this project, and run those models on different parameter combinations as well to test the results. Again, this will be accomplished by creating a web application which allows me and any user to easily implement a trained model to predict stock price, for any stock ticker, for any parameter combination that is requested.