Predicting Stock Price Movements with Machine Learning

Name: Erik Buinevicius Email: embuinevicius@crimson.ua.edu

*Abstract*— In this study, we will attempt to predict and forecast the price of stocks, and more specifically an exchange-traded fund that tracks the S&P 500, using three distinct machine learning models. Forecasting stocks is not an easy task, and with solutions having been developed in both the academic and professional worlds, the question of what parameters and models are the most effective for forecasting the direction of the stock market has many answers. This study uses multiple classification models including logistic regression and an extra trees classifier to predict whether an S&P 500 tracking ETF (SPY) will close higher or lower one day in the future based on historical data. It also uses a long short-term memory neural network to predict the next-day’s closing price of the SPY. The LSTM model will then be used to perform predictions of the SPY’s price multiple days into the future. Additional features such as the CBOE Volatility Index are also explored within our models, to see what factors may be relevant to price movements in addition to historical price. Root Mean Square Error (RMSE) is used to evaluate our LSTM model’s performance. The LSTM model was able to provide a prediction on historical price data within a very minimal RMSE, while the logistic regression model performed in-line with market returns. The results of this study are a promising basis for developing a more complex price forecasting model that can predict the price of a security far into the future. A trading algorithm could be built on top of such a model to execute legitimate trades in the stock market.

# Introduction

Trillions of dollars’ of assets are exchanged on financial markets around the world every day. The stock market has become an extremely a hot topic in the last couple of years, following a major interest boost in retail trading powered by so-called “meme stocks,” and more recently a bear market in 2022 that has wiped out a majority of the gains assumed by investors who have entered the market in the last 5 years. In the last couple of decades, algorithmic trading has become its own force in the market, as firms such as Renaissance Technologies and Two Sigma, each with tens of billions of dollars under management, have funds dedicated purely to algorithmic trading. These funds utilize a long-short methodology, meaning they can bet against the market in addition to making traditional bets on securities. If we can develop an algorithm that can accurately predict the direction the market is moving, it can become a tool used to expand as well as preserve existing wealth.

# Research Questions

Three related but distinct research questions will be explored in this study. The first of these questions is can we accurately predict whether a stock will move up or down tomorrow? Secondly, can we accurately forecast what the price of a stock should be in the future based on historical moves using machine learning? The final question is what machine learning models perform the best when working with historical financial data? The first two questions relate to the actual predictions we will be performing – that is, can we use a classification model to identify the direction that a stock will move tomorrow, up, or down, based on the data that we have for today. Furthermore, an actual price prediction will be performed based on historical stock price data and a machine learning model will be used to forecast this price into the future. The final question relates to analysis of the work done in this study, and finding the best methods possible which would in theory offer the best chance of outsized returns relative to the market’s historical average performance, while also influencing additional tweaking of the models and building of new models in the future.

# Related Work

Although the quantitative trading firms such as those mentioned in the introduction are careful about retaining the proprietary state of their work, the problem of forecasting financial market movements has been well-explored by researchers around the world, meaning plenty of amazing existing works on the topic are available to the public.

Shen and Shafiq in [1]explore very similar problems to those that are be addressed in this paper. However, the predictions they performed were surrounding the Chinese stock market, and they had the capability to work with a much larger dataset and more complex models than those be explored here. Specifically, Shen and Shafiq use a dataset consisting of 3558 stocks from the Chinese stock market – a number which is simply not feasible for the computational power available for our research. Instead, we will focus one stock ticker specifically (SPY), since it tracks the 500 largest companies in the United States (S&P 500) and is generally a good indicator of the overall market. Shen and Shafiq also spent a significant portion of their research on complex recursive feature engineering, which was important as they had to identify the relevant features from within their 3558-stock dataset. While in this paper we will indeed discuss feature engineering and relevancy, a complex recursive approach will not be used to identify relevant features. The primary focus of this study will be building an LSTM model that can perform well given SPY historical data as well as two other stock-price indicators, and tuning hyperparameters to optimize performance. Finally, while Shen and Shafiq’s LSTM model simply returns a binary output of 0 or 1 representing whether a stock is going down or going up, the LSTM model developed in this study outputs a real price prediction, which can then be used recursively to forecast even further into the future.

Bhandari, Rimal, Pokhrel, Rimal, Dahal, and Khatri in [8] also explore a very similar topic to that which is addressed in this paper. These researchers used an LSTM model to attempt to predict the next-day’s closing price of the S&P 500 index. These authors trained their model on 15 years of data, ranging from 2006 to 2020 [8]. These years were selected because they include times of bull markets as well as two major bear markets, being the 2008 financial crisis and the COVID-19 pandemic [8]. In this study, we will also capture the most recent bear market in 2022 that has occurred primarily due to inflation and interest-rate hikes by the Federal Reserve. Bhandari, Rimal, Pokhrel, Rimal, Dahal, and Khatri also used indicators such as the CBOE Volatility Index (VIX), interest rate data, unemployment rates, consumer sentiment index, U.S. Dollar index, and various moving averages [8]. They trained various LSTM models, algorithmically tweaking hyperparameters based on past performance and training. When looking at this paper, it had already been decided that the VIX and 10-Year Treasury would be used as features for this study’s research. Therefore, it is great validation that this research paper also used these features for their model.

# Methods

## Data Acuqisition

All the data used in this study can be easily accessed and gathered from Yahoo Finance’s historical databases [2]. Yahoo finance data can be downloaded directly by simply [visiting the website](https://finance.yahoo.com/), looking up the stock or other security ticker you wish to view data for, and visiting the historical tab. The time period of data that is desired can then be selected, and the dataset can be downloaded in CSV format [2].

Alternatively, Yahoo Finance’s data can be accessed using a web-scraping technique. For our purposes, we will use the pandas-datareader module in Python, which allows us to automatically collect the data for a given ticker, which is essentially a unique identifier for a security in the U.S., and automatically dump the dataset into a dataframe [3].

The pandas-datareader method will be more useful for this study’s purposes than manually downloading the data, as it will allow for easy experimentation with various companies’ data without having to continually visit Yahoo’s website. It will also make it easy to implement company selection on the web-based application that is developed, so that users can test the model’s performance on a variety of company data.

As mentioned previously, this study will primarily be using the historical price data of the SPDR S&P 500 ETF Trust (SPY) for forecasting, because it tracks the largest 500 companies in the United States. This means that the model will provide a great way to track general stock market movements, and will additionally be extendable to individual companies’ securities.

In addition to historical SPY price data, this study will be using two other significant indicators, being the CBOE Volatility Index (VIX) and U.S. 10 Year Treasury yield proxy (TNX). Like SPY’s historical data, historical values for these items can be gathered from Yahoo Finance in similar fashion. The VIX is a measure of volatility in the overall stock market and is typically accepted to be inversely correlated with stock market movements [8]. This means that if the VIX moves up over a period of time, it would be expected that the broader market would move down, and vice versa. The 10 Year Treasury will be a good feature to include in our model because it can essentially act as a proxy for interest rates set by the Federal Reserve. With the market’s current inflationary environment, interest rates have been a hot topic, and tracking this will be important for our models to perform.

## Data Cleaning and Labeling

We will primarily be focused on historical price data and alternative index data in our research, and it will be important to ensure that these datasets have no missing values when working with them. An example of what the dataframes scraped from Yahoo Finance look like is shown below.

Text

Description automatically generated

Fig. 1. Example of data gathered from Yahoo Finance.

The fields shown here are the same ones present in all three datasets, namely the SPY, VIX and TNX, meaning renaming of some columns must be performed. We will perform renaming such that company price data retains its original column names, while VIX and TNX data columns are renamed to VIX\_Name and TenYr\_Name respectively, where “Name” represents the original column name present in the dataset. Fortunately, the historical data did not have any missing values. However, since data is read in dynamically with pandas-datareader, it is possible that there will be missing values in the future. Therefore, we should come up with a method for filling missing values in case it is necessary in the future. Since the value of SPY stock price generally increases throughout time (primarily due to inflation as well as other factors) we will use the mean of two days in the future and two days in the past to fill any missing values for price data. This will ensure that any recent data that is missing is not influenced by price data from 10 years ago, for example.

Since VIX data and treasury yield data fluctuate in more of a static range we will simply use the average of the entire column of data to fill empty values for these datasets. This can be performed with pandas’ dataframe.isna() function combined with dataframe[column\_name].mean().

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 2. TNX all-time price chart.

Graphical user interface

Description automatically generated

Fig. 3. VIX all-time price chart.

The three datasets that have been discussed will be joined on the “Date” column, as it will be the one identifying feature that can be used to link the datasets. Pandas’ merge( ) function can be used in a straightforward manner to perform an inner join on the datasets

## Feature Engineering

This study will explore three different models, all of which will be trained on unique datasets. Hence, the feature engineering discussed below may be unique to one or two of our datasets, without being used in the other(s). For example, the LSTM model that will be used to forecast the future price of stocks will only use historical data for the SPY, VIX, and TNX and won’t require the “NextDayChange” or “Direction” features that will be essential to our other two models that explore classification. This is important to keep in mind as our three models and their respective datasets are different from one another, primarily because the goal of this research was to identify the best way to forecast stocks, so that further in-depth analysis and research can be executed from a solid basis of experimental results.

Most of the data that we will be using is already in a state where it is ready to be normalized and trained on. However, for our classification models we will have to engineer a feature for the output, or y-values, of our training and test data. What this means is based on the data for a given day, we must identify if the stock moves up or down on the next day. To do so, we can engineer a feature titled “NextDayChange”. We can do this by taking the value of the “Close” column and dividing the value of “Close” from one row in the future by the current value, doing so for each row in the dataframe. Pandas’ shift() method will come in handy here, as it can be used to look one row into the future. Once we have the value of NextDayChange, we must engineer another feature to simply identify whether this change value indicates an up move or a down move for tomorrow’s stock price. We can do this by identifying bins that range from [-100, 0] and (0, 100] respectively. We can then cut the NextDayChange column on these bins, creating a new binary feature labeled “Direction.” After creating this feature, any row where the next day’s percent change is greater than zero will have a direction of “Up”, while a percent change less than or equal to zero will be identified as “Down”. In our model which uses Logistic Regression, we will further engineer this feature such that “Up” is represented by 1, and “Down” is represented by 0, which allows for scikit-learn’s LogisticRegression function to handle our data.

The number of up days in the dataset, which begins on January 1, 2006, outnumbers the number of down days, and represents approximately 54% of our dataset for the SPY.

Chart, bar chart

Description automatically generated

Fig. 4. Direction distribution.

The NextDayChange and Direction values are the most fundamental features that were engineered, and are the only modifications necessary for AutoML’s TPOTClassifier to work with the data, which is one of the models that is explored. For our Logistic Regression model on the other hand, we will not just be using the VIX and 10-Year Treasury inputs along with our stock price data and will additionally look at 30 trailing days of price data to see if we can predict the next day’s move as up or down. In order to get this trailing price data, we will again use the shift() method from pandas, but instead use it within a loop, storing 30 trailing days of price data along with a given day’s price in each row.

We will also generate a feature titled “Prev Change” in similar fashion to the way we generated “NextDayChange,” except we will shift the data in the other direction (backwards one day). This means that for our Logistic Regression dataset, each row will contain the day’s adjusted closing price, the VIX adjusted closing price, the TNX adjusted closing price, 30 trailing days of price data, and the previous day’s percent change, all of which will be used as inputs for the model.

## Model Development

In this study we developed three distinct models to predict stock prices, each of which have already been briefly mentioned and will have distinct datasets that are used as input.

The first two models that were developed are classification models that attempt to predict whether a stock will move up or down tomorrow based on the known data for a given day. For the first of these models, we used AutoML’s TPOTClassifier tool that exists as a Python package. The TPOTClassifier package’s fit() method takes in two datasets, one with input data than can exist as a dataframe with a variety of features, and one with output data that exists as a one-row dataframe containing classification outputs corresponding to the input rows (X and y data, respectively) [5]. The TPOTClassifier tool handles selection and optimization of the entire machine learning pipeline, and outputs a model with the optimized parameters based on its trial, modification, and error methodology. As the X-input for this model, we will use our three datasets joined together which contains historical data for AAPL, the VIX, and the 10-Year Treasury, with our y-output consisting of an up or down classifier.

The second model that was developed was a logistic regression model using scikit-learn’s LogisticRegression module. Logistic regression is a “statistical analysis method to predict a binary outcome, such as a yes or no, based on prior observations of a data set” [6]. Logistic regression allows the user to input unrelated features and attempts to use this data along with historical outcomes that it is trained on to predict whether the output falls into one of two categories. In our case, the model will once again be predicting either up or down regarding the next day’s price movement, however this time up will be classified as 1 and down will be classified as 0 in the data due to the limitations of scikit-learn’s LinearRegression implementation. We can input the dataset that we engineered that consists of a day’s closing price, 30 days of trailing price data, VIX adjusted close, and previous day’s percent change, with the output being either a 0 or 1 value that will be mapped to down and up respectively. This model was the most straightforward to develop out of the three.

The final model that will be develop will be a long short-term memory, or LSTM, model. This is the model that the most time was spent on, and it is well-known as a classical method of stock price prediction and forecasting. The reason that LSTM is often used for stock price prediction is that it is a variation of a Recurrent Neural Network which is better at retaining long-term information while still processing short-term memory effectively [7]. Classic RNNs suffer from the problems of vanishing gradients which results in them having difficulty retaining long-term patterns, and LSTM networks are designed to address this problem [8]. In our LSTM model we will use a subset of features used by Bhandari et al. in their LSTM model – namely SPY historical price data and VIX historical price data, as well as 10-Year Treasury historical yield data, an alternative to the EFFR interest rate tracker used in [8]. A single-layer LSTM will be used as the primary model, since in Bandari et al.’s research, they found that single-layer LSTMs outperform multi-layer LSTMs on average, when used for stock-price predictions [8]. Various 3-layer LSTMs will also be developed with different number of neurons, dropout rates, batch-sizes, and look-backs.

## Model Training

The classification model that will be trained in this study uses the tpot module and will be constructed with primarily default hyperparameters. The primary parameters that were tweaked from defaults include generations, which is the number of iterations that the entire pipeline optimization process is run [9]. The default for this value is 100, but the training time was already extremely long compared to the other two models, so it was reduced to 10 for the sake of time. The warm\_start parameter was also set to true, as this allows for pausing the TPOT optimization and resuming from where it left off. Just in case of an unexpected machine shutdown, this parameter prevents optimization state from being lost [5]. Verbosity was updated to 2, since the training of this model took especially long, and being able to view a progress bar was helpful [5]. Overall, the total number of pipelines evaluated by tpot is population\_size + generations \* offspring\_size, which with the hyperparameters that were used, shown below, means for our model, TPOT evaluated 100 + 100 \* 10 pipelines in total, which equates to 1100 total pipeline evaluations (offspring\_size gets set to population\_size if left as None) [9].

Text

Description automatically generated

Fig. 5. TPOTClassifier hyperparameter inputs.

The logistic regression model that was trained on 30 days of historical price data as well as VIX\_AdjClose and TenYr\_AdjClose was trained using mostly default parameters from scikit-learn’s linear\_model.LogisticRegression function. The random\_state parameter was set to 2 in the specific verseion of the model discussed in this paper. However, even with other random\_state values such as 10 and 40, the accuracy of the model remained relatively constant. For this model, 80% of data was used for training and 20% for testing, as the model can evaluate results on the testing data, as well as validating on the training data. We can therefore view two separate accuracies for this model, one on the testing data and one on the training data.

For the LSTM, or long short-term memory, model Keras was used, which is a neural network library built on top of Tensorflow with a built-in LSTM layer [10]. Various training method inputs were tested for our LSTM model and resulted in the widest variation in results, although every model that was tested on the dataset performed relatively well (under 12% Root Mean Square Error). Both single and multi-layer LSTM models were tested, with the multi-layer models that were trained consisting of three layers specifically. A dropout layer was always introduced after an LSTM layer, whether it was an intermediate layer or not, to reduce overfitting. Various dropout percentages were tested, including 20%, 40%, and 30%. Different units, or number of neurons, were also tested in each of our LSTM models, including 25-unit LSTM, 50-unit LSTM, and 100-unit LSTM. The LSTM models with more neurons did not necessarily always perform better and took significantly longer to train. 50 was settled on as the default unit parameter used for the “finalized” models. Possibly the most important hyperparameter for the LSTM model was the look-back, which will be referred to as the window, that was used for training and testing. The look-back simply means the number of consecutive historical inputs that are used to predict a single output value using LSTM. Windows of 15, 30, 50, 60, and 180 were tested, each with various number of layers and hyperparameter inputs. Shorter windows simply mean that that data used for training is more recent. So, for example, if a stock went up 50% in the last 2 months but went down 10% in the last 15 days and a 15-day window was used, the model will only use the more recent data to perform a prediction for the next day’s stock price. Keras’ mean\_squared\_error function as used as used as the loss function for the LSTM model, as it is generally accepted as a good evaluator of time-series data, and will also be used for validation of the LSTM model [10]. The optimizer used for the LSTM model was “Adam”, as it combines the advantages of AdaGrad and RMSProp optimizers and is generally thought of as a solid optimizer for LSTM [11]. The default learning rate used with the Adam optimizer is 0.001 [11]. The number of epochs was kept constant at 100, after variation showed that more epochs did not necessarily improve validation accuracy while contributing to a significant increase in training time. Various batch sizes were experimented with for LSTM, including 64, 32, 16 and 8. An example of both a single-layer LSTM and multi-layer LSTM creation using Keras in Python are shown below, as well as the model summary output and training method for each.

Text

Description automatically generated

Fig. 6. Single-layer LSTM.

Text

Description automatically generated

Fig. 7. Three-layer LSTM.

Text

Description automatically generated with medium confidence

Fig. 8. LSTM model training source code.

## Model Evaluation and Validation

For both of the studied classification models, built with TPOTClassifier and Logistic Regression, respectively, a confusion matrix was used to validate the models. Scikit-learn makes it extremely easy to evaluate classification models using a confusion matrix, and since we only have two possible outputs, being up or down, the confusion matrices displayed in the section titled *Model Performance in Evaluation* are easy to understand. Matplotlib is used alongside scikit-learn to display the confusion matrix.

For our LSTM model, Root Mean Square Error (RMSE) was used to evaluate the performance of the model.

Text

Description automatically generated

Fig. 9. Root Mean Square Error (RMSE) formula [12].

RMSE is a good evaluator of accuracy in LSTM models since its output will always be a positive value and it is easy to understand. An RMSE of 0 would represent a perfect fit of the model to the validation data. The division by the number of observations means that RMSE will be a good estimator of the “the standard deviation of the error for a typical single observation rather than some kind of ‘total error’” [12]. This means that RMSE should remain relatively consistent regardless of the number of observations and is essentially how far off we should expect the next prediction to be from its validation input [12].

## Model Deployment

The model was built and deployed using Streamlit, which is an all-in-one data science dashboarding tool that serves the model to the web while also allowing for data dashboarding with pre-configured interfaces for common components out of the box [13]. Streamlit makes it easy to visualize a machine learning model on a webpage, providing features such as sidebars and plotting. The source code for the web-based app is less than 300 lines of code thanks to Streamlit’s built-in functionality in the Python library.

The model was deployed using Streamlit Cloud, which makes it extremely easy to deploy an existing Streamlit application to their hosting platform for free, making it accessible to anyone [13]. To deploy the app, the repository containing the app.py file must first be uploaded to GitHub, which can then be linked to a Streamlit account. A requirements.txt file must exist within the same directory level as the application, so that Streamlit can install necessary dependencies for hosting the web application. Once the GitHub repository containing the application and requirements file is linked to your Streamlit account, the app can be deployed in just a few clicks from Streamlit’s Cloud Dashboard [13]. It should be noted that for this project, Python version 3.10 was used for the web application deployment, as earlier versions of Python led to dependency conflicts during deployment. The web application can be viewed [here](https://ebuinevicius-ml-stock-prediction-app-u5rfoz.streamlit.app/). All source code, notebooks, and datasets used can be found in the [GitHub repository](https://github.com/ebuinevicius/ML-stock-prediction).

There are multiple alternatives to Streamlit Cloud that could be used for a more professional deployment of the web-based model in the future. One of these alternatives is Heroku, which is a Platofrm-as-a-servies (PaaS) implementation used for deploying and managing scalable web applications. Applications built on Streamlit are compatible with Heroku, which provides deployment and monitoring for all times of web applications. Another PaaS option is Amazon Elastic Beanstalk, which allows users to upload their code to AWS, and then handles configuration of resources for them. Like Heroku, Elastic Beanstalk provides users with a dashboarding tool to manage their application’s metrics. Both Heroku and Amazon Elastic Beanstalk are more scalable alternatives to Streamlit’s native cloud, and our web-based model will likely be deployed to one of these platforms in the future.

# Results

## Model Performance in Evaluation

The model that we will evaluate first is our worst performing model, being the classification model built using TPOTClassifier, which came up with an ExtraTreesClassifier as the best performing model for our dataset. The model seemed to perform relatively well during the evaluation on training data, at performed with ~55% accuracy throughout training.

Text

Description automatically generated

Fig. 10. Evaluation of AutoML TPOTClassifier during training.

However, this training was performed on 80% of the dataset, and when the model was tested on the remaining 20% of the dataset, the model performed with less than 50% accuracy.

Chart, treemap chart

Description automatically generated

Fig. 11. Root Mean Square Error – ExtraTreesClassifier (49.6% accurate).

With this validation result, it is clear that the TPOTClassifier AutoML model has not been a success for our purposes, as we would have simply been better off predicting that the stock market would go up on any given day, since the percentage of up days is greater than the percentage of down days, as up days represent ~54% of the data (as displayed in Fig. 4). This model should not be used for trading under any circumstances in its current state, and further optimization of the hyperparameters passed into TPOTClassifier would be required to improve this model.

The logistic regression model built using scikit-learn’s LogisticRegression function performed slightly better than the AutoML model used for classification. This model performs with 54.1% accuracy, which is essentially in line with the performance that would be obtained by simply predicting that every day the market would go up (Fig. 4). Therefore, for this model to be used in real-world trading, it would ideally be further optimized by performing further feature evaluation and optimization to improve the way that logistic regression performs on the model. Additionally, more in-depth hyperparameter optimization could be performed on scikit-learn’s LogisticRegression function. Finally, a confidence score could instead be used, with a prediction only being made if the confidence is above a certain threshold, for example 60%. This method would need to be back tested on the data to determine how accurate prediction using confidence score would be for various percentage thresholds.

Chart, treemap chart

Description automatically generated

Fig. 12. Root Mean Square Error – Logistic Regression (54.1% accurate).

To validate the accuracy our LSTM model, Root Mean Square Error was used. For our three-layer LSTM models, RMSE ranged from ~4.6% to ~11.5%, depending on other hyperparameters and the window used. The best performing three-layer model was a 100-neuron LSTM with 3 30% dropout layers and a 60-day look-back. The worst-performing 3-layer model consisted of 25-neuron LSTM layers with 3 40% dropout layers. This seemingly revealed that using 40% dropout was too much for this dataset as it is not extremely large, and that 25-units for LSTM was not enough.

On average, the one-unit LSTM models performed better than the three-layer models based on the RMSE metric. The best-performing one-layer model was a 50-day lookback with 50 neurons, and one 20% dropout layer. This model had a Root Mean Square Error of only 1.3% on the test data, which represented 25% of the dataset. The worst-performing one-layer LSTM model that was tested had a Root Mean Square Error of ~9.5%. However, this model used a bidirectional LSTM layer, with the hyperparameters remaining the same as our best performing model. This revealed that a bidirectional model would not be the best resolution to our stock market prediction question. When considering only unidirectional LSTM models, the worst performing model still had a RMSE value of ~2.6%, which is far better than the best-performing three-layer model. This model had a look-back of 180 days, 50 neurons in the LSTM layer, and used a single 20% dropout layer. All the validations discussed thus far were performed on the test dataset for SPY.

Chart, line chart

Description automatically generated

Fig. 13. 50-day window, 50-unit, single-layer LSTM model prediction vs validation data.

Chart

Description automatically generated

Fig 14. 180-day window, 50-unit, single-layer LSTM model prediction vs validation data.

To continue experimenting and testing the limits of the model, an actual stock price forecast was performed using our best performing model, being the 50-day single-layer LSTM model. We used this model to predict the price of the SPY five days into the future. In order to perform this prediction, the most recent 50-day window from our dataset must be input to the model to predict the next expected output, which represents tomorrow’s expected stock price. This value can then be input back into the model recursively, which will allow for forecasting multiple days into the future. Since the only output from the model is the predicted day’s price value, we must come up with inputs for the VIX adjusted close and the 10-Year Treasury adjusted close. To do so, we took the average of these respective values for the previous 50 historical days of data.

Chart, line chart

Description automatically generated

Fig. 15. 5-day forecast of SPY, visualized following 50-days of historical price predictions.

Because values predicted by the model are fed back into the model to project future values, the forecast potentially contains a great deal of uncertainty, which could lead to inaccurate results. Therefore, this model would need to be back tested extensively before it could be used to execute trades with reliably.

## Feature Importance

In our logistic regression model, we found that interestingly, the most important feature is not exactly clear, although it seems like the VIX close data and 10-Year Treasury data are being drowned out by the closing price features. In the chart displayed, features 0-2 represent SPY adjusted close, VIX adjusted close, and TNX adjusted close respectively, while the remaining features represent trailing price data for days looking backward in increasing order.

Chart, bar chart

Description automatically generated

Fig. 16. Visualization of logistic regression feature importance.

While in our logistic regression model the VIX adjusted close and 10-year adjusted close did not seem relevant, they were indeed important for the LSTM model. This is proven by the fact that LSTM models trained on a dataset with the only feature being trailing price performed worse than the “finalized” model that used trailing price in addition to VIX and Treasury data. “Finalized” is put in quotations because this model is not perfect and should be continually optimized in the future. This optimization could be continued by implementing and considering even more additional features that could potentially be proxy indicators for stock market movements and in turn be inputs to the model.

## Runtime Monitoring

The first classmate who reviewed the web-based model responded, “Playing around with this model was very fun and also extremely intuitive. By allowing the user to input their own stocks and historical data points, it creates a fun and interactive way to explore using the model. The model seems very accurate, but may have a slight bias towards underpredicting the value for the stock prices. It is impressive however, that any stock can be used with the model. When using large amounts of years, the graph of the prediction is hard to see and compare the values of predictions. There is also no error handling when a user types an incorrect stock so any typos force the website into a complete error. Overall, this website is very fun to use and explore and adding spelling checks and a zoom/magnify-like feature for the graphs would make this a great deployment.”

The second classmate who reviewed the model stated “Overall, a very pleasing Streamlit model to look at and very informative. It was very clear what the model was doing and what I had to input as the user. I think for the future it would be beneficial to have the user input the name of the company and the model could find the stock ticker. This would increase the ease of use for users who do not have a background in stocks. One of the companies I used for my testing data sample was LoveSac and I encountered a remote data error. I restarted the web application and tried again with the same company and output was given with no issues. I also used a new company that just started publicly training on NASDAQ for my testing data sample. When trying to get historical data from a date before the company went public, the application gave me an error.”

These reviews of the model indicate that the web-based model is useful and enjoyable to use, but could and should still be improved from a technical standpoint before being deployed at scale. A check should be performed on the stock ticker that the user enters, to ensure that querying Yahoo Finance’s database will not error-out because of the input. Additional features on the displayed charts would also be useful to users, and make it easier to visualize and understand what is going on with the displayed predictions. Furthermore, the deployment could be made more friendly for users who are unfamiliar with stocks, by allowing them to input a company name or select from a list rather than having to know the stock ticker, for example. Finally, since there is currently no error checking for whether a company’s data goes back to the date that the user selected for historical data, the website sometimes errors out for companies that recently started trading. All of these items will be considered and addressed in the future.

# Discussion

## Answers to Research Questions

Overall, our research questions were answered with relative success. Our first question was related to accurately predicting whether a stock will move up or down tomorrow. While the extra trees classifier model chosen by AutoML was unsuccessful at performing this task, the linear regression model used was on-par with the accuracy of predicting that a stock will move up tomorrow, which comprises most days on the stock market. While this model did not significantly outperform the market, it technically was over 50% in its predictions, which is not a complete failure.

The second question related to accurately forecasting the future price of a stock based on historical data. We were successful in answering this question. The best LSTM model created, being the single-layer LSTM with a 50-day window, accurately predicted the future stock price with a Root Mean Square Error of less than 2% on the test data. This is a great success and a great starting point for creating a complex forecasting algorithm could go even further in answering how accurate it is possible to be when forecasting the price of a stock far into the future.

The third and final question introduced at the beginning of this paper surrounded exploring the most relevant features and the best type of machine learning model for predicting stock prices. The LSTM model was the most valuable of the three models, and therefore the best type of model for forecasting stock prices. The classification models that were explored would simply be unable to predict the price of a stock more than one day into the future and are also unable to predict the actual price of a stock, and rather focus on the direction. Therefore, the LSTM model is by far the most valuable as it can be used not only to predict direction, but to predict the price of a stock many days into the future. Our research suggests that the most important features for predicting the price of a stock are simply historical values of items such as trailing price data as well as external proxy indicators.

## Limitations of your research.

As mentioned previously, the primary limitation of both classification models developed are that they are unable to forecast the actual price of a stock and are instead focused on a simply up or down move. This means that the models could potentially be predicting up or down days correctly that are small in magnitude, such as 0.1% moves, while they are incorrectly predicting moves that are large in magnitude, such as those 5% and above. This would mean that even if the models are predicting the binary movement accurately, using the model to execute trades in the market could result in less-than-ideal returns. Further validation would need to be performed on these models to correlate the up or down predictions to the actual size of moves.

Our LSTM model also has limitations, especially regarding forecasting. As mentioned, price values predicted by the model are fed back into the model recursively to perform future predictions, meaning as the forecast goes further into the future, the prediction becomes more uncertain. Additionally, since the model does not perform predictions for the VIX or 10-year Treasury data, recent averages must be used as inputs for the forecast. It would be interesting to explore a better methodology for performing these future predictions such as creating a model to predict VIX and Treasury values themselves, which could then be input back into the model. It may also be useful to add additional features to the LSTM model such as some of those discussed in [1].

## Lessons Learned

Overall, this project was an extremely valuable learning experience, and I feel that I gained more from knowledge from it than any other project throughout my college career. Coming into this class, I was extremely intimidated by the concepts of artificial intelligence and machine learning. Even throughout the semester, discussing some of the concepts felt daunting and I was unsure if I fully understood what was going on at times. While completing this project, all the concepts of the data science workflow began to make more sense and come together to me, from data gathering and cleaning, to feature extraction and engineering, to model development and training, and finally evaluation, I now understand what it takes to be a true data scientist, and I will be much more confident when tackling future data science projects. In addition to my knowledge of data science and machine learning concepts growing tremendously, my ability to put these concepts into practice using data science tools for Python was greatly enhanced throughout this project.

One thing that I would take away for the future when working on data science projects, is not to overlook the data exploration and feature engineering phases of the pipeline. When jumping into this project, I was excited to begin the machine learning aspect as that is what I was most unfamiliar with. This caused me to begin training models on incomplete datasets, which ultimately led to me having to revisit the feature engineering and evaluation phases towards the end of my work, and thereby forcing me to retrain models. In future works, I will be sure not to overlook the data-focused portion of the process before jumping into the machine learning side.

##### Acknowledgment

I would like to acknowledge Sam Perry for being an overall influence on my learning of the concepts of data science and machine learning throughout the semester. Sam was always willing to explain and help me understand a topic that I may have been confused on or needed clarification on following a lecture, thanks to his background in artificial intelligence being more extensive than my own. He also was the first one to test my web-based model deployment and give feedback, so I would like to thank him again for that.

##### References

1. Shen, J., Shafiq, M.O. “Short-term stock market price trend prediction using a comprehensive deep learning system.” *J Big Data* **7**, 66 (2020). <https://doi.org/10.1186/s40537-020-00333-6>
2. “Yahoo Finance - Stock Market Live, quotes, Business & Finance News,” *Yahoo! Finance*. [Online]. Available: https://finance.yahoo.com/. [Accessed: 05-Dec-2022].
3. “Remote Data access¶,” *pandas*. [Online]. Available: https://pandas-datareader.readthedocs.io/en/latest/remote\_data.html#remote-data-yahoo . [Accessed: 05-Dec-2022].
4. “Get started,” *Streamlit Docs*. [Online]. Available: https://docs.streamlit.io/streamlit-cloud/get-started. [Accessed: 05-Dec-2022].
5. R. S. Olson, “Using TPOT,” *Using TPOT - TPOT*. [Online]. Available: http://epistasislab.github.io/tpot/. [Accessed: 05-Dec-2022].
6. G. Lawton, E. Burns, and L. Rosencrance, “What is logistic regression? - definition from Searchbusinessanalytics,” *Business Analytics*, 20-Jan-2022. [Online]. Available: https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression#:~:text=Logistic%20regression%20is%20a%20statistical,or%20more%20existing%20independent%20variables. [Accessed: 05-Dec-2022].
7. M. Phi, “Illustrated guide to LSTM's and GRU's: A step by step explanation,” *Medium*, 28-Jun-2020. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed: 05-Dec-2022].
8. Bhandari, H.N., Rimal, B., Pokhrel, N.R., Rimal, R., Dahal, K.R., Khatri, R.K.C. “Predicting stock market index using LSTM.” *Machine Learning With Applications*, 15 September 2022. <https://www.sciencedirect.com/science/article/pii/S2666827022000378>
9. J. Hale, “TPOT automated machine learning in Python,” *Medium*, 10-Jul-2020. [Online]. Available: https://towardsdatascience.com/tpot-automated-machine-learning-in-python-4c063b3e5de9. [Accessed: 06-Dec-2022].
10. Keras Team, “Keras documentation: About keras,” *Keras*. [Online]. Available: https://keras.io/about/#:~:text=Keras%20is%20a%20deep%20learning,focus%20on%20enabling%20fast%20experimentation. [Accessed: 06-Dec-2022].
11. J. Brownlee, “Gentle introduction to the adam optimization algorithm for deep learning,” *MachineLearningMastery.com*, 12-Jan-2021. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed: 06-Dec-2022].
12. J. Moody, “What does RMSE really mean?,” *Medium*, 06-Sep-2019. [Online]. Available: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e. [Accessed: 06-Dec-2022].
13. “Deploy an app,” *Streamlit Docs*, 2022. [Online]. Available: https://docs.streamlit.io/streamlit-cloud/get-started/deploy-an-app. [Accessed: 06-Dec-2022].