Virginia Polytechnic Institute and State University

---

# S&P 500 Forecasts and Predictions Using Machine Learning Algorithms
**Fall 2022 CS 5644**
**December 10th, 2022**

---

Matt Hargenrader (moh5644)
Caitlin Hermann (caitlinh99)

## Table of Contents

## Introduction

Evaluating and analyzing stock prices is a massive industry that has a major impact on everything from a family's personal finances to multinational companies' business strategies. The Standard and Poor's (S&P) 500 index is one of the most widely known and cited stock market indices, as it describes the performance of the 500 largest publicly traded companies. Due to the scale and impact of this single indicator, this project will analyze historical S&P 500 data to predict and forecast relevant metrics based on S&P 500 member stock data.

Due to the stock market's highly volatile nature, and the fact that the stock metrics consist of real number values, we will not be using any classification algorithms based on supervised learning covered during the course. Instead, we will be implementing a variety of regression algorithms, including Linear Regression, K-Nearest Neighbor (kNN) Regression, Autoregression, and Neural Networks.

The following report details our approach to preprocessing the S&P 500 dataset and implementing suitable algorithms to predict relevant metrics, as well as a discussion regarding our results and lessons learned.

## Business Objective

Since the stock market is known to be volatile and essentially unpredictable (or we would all be millionaires), our goal with this project is to simply grant greater cognizance of the future market situation using sound machine learning models in order to benefit anyone with interests in the stock market. We saw this dataset as an opportunity to experiment with a variety of machine learning (ML) tools and algorithms to produce various different data science insights within the stocks domain. As previously mentioned, the S&P 500 index is a wildly important economic indicator and an essential benchmark index for the U.S. stock market that is followed closely by many. In light of this, we found it one of the most obvious applications to *forecast* the S&P 500 index. The S&P 500 index not only gives a big picture snapshot of the stock market, but of the entire U.S. economy. This draws great interest in forecasts of this metric from economists, politicians, mutual fund managers, or simply individuals managing their retirement.

Although the index provides an important picture of the overall stock market, it does not give much insight into people who care about individual stock prices, for a specific company or otherwise. In addition to forecasting the S&P 500, we will also use the data to attempt to predict stock prices. This could help investors determine whether they would be better off buying or selling a stock at any given time. We will focus on predicting the *Adjusted Close* (i.e., closing price of the stock at the end of the trading day) metric as our target.

Essentially, our goal is that anyone with a vested interest in the stock market could benefit from the insights provided by these models, and potentially remove a small amount of the uncertainty associated with participating in the stock market.

## Dataset

The dataset we used for this project came from kaggle.com (link to dataset) which includes three .csv files containing information regarding each company's stock and the overall S&P 500 index value, as well as general information about each company. We have provided an overview of the three files and their features below:

| Feature Name | Description | Example | Type |
|---|---|---|---|
| Date | The date of the recorded stock instance | 1/4/2010 | Categorical |
| Symbol | Stock ticker symbol used to identify each company; relates this table with *sp500_companies.csv* | MMM | Categorical |
| Adj Close | Adjusted Closing Price; amends the closing price of a stock based on corporate actions | 59.31889 | Numerical |
| Close | The closing price of the stock at the end of the trading day | 83.02 | Numerical |
| High | The highest price at which the stock was traded during the trading day | 83.45 | Numerical |
| Low | The lowest price at which the stock was traded during the trading day | 82.67 | Numerical |
| Open | The opening price of the stock at the start of the trading day | 83.09 | Numerical |
| Volume | The number of shares traded in a stock | 3043700 | Numerical |

sp500_stocks.csv (8 features, 1,614,630 instances)
This file contains unique instances of stock price information for each company in the S&P 500, for each day dating back to 2009.

| Feature Name | Description | Example | Type |
|---|---|---|---|
| Date | The date of the recorded S&P 500 index | 9/24/2012 | Categorical |
| S&P500 | The S&P 500 index value for that day | 1456.89 | Numerical |

sp500_index.csv (2 features, 2,517 instances)
This file contains the S&P 500 index value for each day dating back to 2012.

| Feature Name | Description | Example | Type |
|---|---|---|---|
| Exchange | Symbol for the exchange or marketplace each company participates in | NMS | Categorical |
| Symbol | Stock ticker symbol used to identify each company | AAPL | Categorical |
| Shortname | A shortened version of the company name | Raytheon Technologies Corporati | Categorical |
| Longname | The full company name | Raytheon Technologies Corporation | Categorical |
| Sector | The group of stocks the company fits in | Technology | Categorical |
| Industry | The industry the company belongs to | Consumer Electronics | Categorical |
| Currentprice | The most recent selling price of a stock | 150.43 | Numerical |
| Marketcap | Market Capitalization; the total value of all a company's share of stocks | 2.46801E+12 | Numerical |
| Ebitda | Earnings Before Interest, Taxes, Depreciation, and Amortization; value for the company's overall financial performance | 1.3E+11 | Numerical |
| Revenuegrowth | The percentage of growth in a company's revenue | 0.019 | Numerical |
| City | The city the company's headquarters is located in | Cupertino | Categorical |
| State | The state the company's headquarters is located in | California | Categorical |
| Country | The country the company's headquarters is located in | United States | Categorical |
| Fulltimeemployees | The number of full-time employees | 154000 | Numerical |
| Longbusinesssummary | Summary of the company | Apple Inc. designs, manufactures, and markets smart phones… | Categorical |

sp500_companies.csv (15 features, 493 instances)
This file contains a unique instance for each company in the S&P 500.

**Preprocessing Steps**

Because we are investigating two different business objectives (i.e. predicting closing prices and forecasting the S&P 500 index), we have split our project into two experiments. The preprocessing steps for each experiment are outlined below:

*Experiment 1: Forecasting the S&P 500 Index*

Given the time-series data of the S&P 500 since 2012, all we had to do was some slight preprocessing to prepare the data for a linear regression algorithm. First, we simply created a copy of the *index* data frame, which we had already read into Python, so that we could perform the necessary alterations for this specific experiment. These alterations included: dropping the

date column, reversing the data frame so that the most recent data is at the head, and creating (t-n) columns. This brought us to a data frame with the previous 7 days of indices as the features/inputs. Below are images of the original data frame, along with the preprocessed data frame after making the alterations (where S&P500 is the S&P value at time *t*, S&P1 is the S&P value at time *t-1*, S&P2 is the S&P value at time *t-2*..., and so on).

| | Date | S&P500 |
|---|---|---|
| **0** | 2012-10-11 | 1432.84 |
| **1** | 2012-10-12 | 1428.59 |
| **2** | 2012-10-15 | 1440.13 |
| **3** | 2012-10-16 | 1454.92 |
| **4** | 2012-10-17 | 1460.91 |
| **...** | ... | ... |
| **2510** | 2022-10-04 | 3790.93 |
| **2511** | 2022-10-05 | 3783.28 |
| **2512** | 2022-10-06 | 3744.52 |
| **2513** | 2022-10-07 | 3639.66 |
| **2514** | 2022-10-10 | 3612.39 |

| | S&P500 | S&P1 | S&P2 | S&P3 | S&P4 | S&P5 | S&P6 | S&P7 |
|---|---|---|---|---|---|---|---|---|
| **0** | 3612.39 | 3639.66 | 3744.52 | 3783.28 | 3790.93 | 3678.43 | 3585.62 | 3640.47 |
| **1** | 3639.66 | 3744.52 | 3783.28 | 3790.93 | 3678.43 | 3585.62 | 3640.47 | 3719.04 |
| **2** | 3744.52 | 3783.28 | 3790.93 | 3678.43 | 3585.62 | 3640.47 | 3719.04 | 3647.29 |
| **3** | 3783.28 | 3790.93 | 3678.43 | 3585.62 | 3640.47 | 3719.04 | 3647.29 | 3655.04 |
| **4** | 3790.93 | 3678.43 | 3585.62 | 3640.47 | 3719.04 | 3647.29 | 3655.04 | 3693.23 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **2510** | 1460.91 | 1454.92 | 1440.13 | 1428.59 | 1432.84 | NaN | NaN | NaN |
| **2511** | 1454.92 | 1440.13 | 1428.59 | 1432.84 | NaN | NaN | NaN | NaN |
| **2512** | 1440.13 | 1428.59 | 1432.84 | NaN | NaN | NaN | NaN | NaN |
| **2513** | 1428.59 | 1432.84 | NaN | NaN | NaN | NaN | NaN | NaN |
| **2514** | 1432.84 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

Finally, we used the dropna() function to drop the last 7 rows where there were missing values due to not having the previous data.

*Experiment 2: Predicting Stock Prices*

For this experiment, our first preprocessing step was joining the three csv files into one data frame in order to combine all of the relevant data points. Once we merged all of the data points, we began the feature selection process where we weeded out the noisy (i.e., irrelevant) columns from the data frame. The features we decided to drop include *Shortname*, *Longname*, *City*, *Longbusinesssummary*, *Symbol*, and *Date*. We concluded that these features do not have any bearing on predicting the closing stock prices, nor S&P index, so it was best to drop them from the data frame not only to optimize performance but also to avoid negatively impacting the results.

The next step was performing one-hot encoding which is the process of converting categorical data to numeric data. The categorical data we one-hot encoded, included the following features: *Exchange*, *Sector*, *Industry*, *State*, and *Country*. This brought the total number of features in the data frame to 187 and although this is shy of the 200 feature requirement, we do not believe it makes sense to include any of the other features outlined in the Dataset section.

Once we had the ideal dataset, it was time to handle missing values. Since the percentage of missing data is only around 2% for most columns (and only as high as 6% for one column), we decided it was practical to simply drop all these instances from the dataset to arrive at a fully preprocessed dataset.

## Methods, Models, and Results

*Experiment 1: Forecasting the S&P 500 Index*

After the data was preprocessed as outlined in the previous section, we could utilize a simple linear regression model with the S&P 500 as the target, and the previous week of S&P indices as the input values to perform an autoregression by hand. We performed a 5-fold cross validation on the model and reported the appropriate evaluation measures for a time-series model, *root mean squared error* (RMSE) and *mean absolute error (MAE)*. Below are the average results over the 5 folds.

| Autoregression | |
|---|---|
| RMSE | 32.313 ± 2.486 |
| MAE | 19.887 ± 1.244 |

*Experiment 2: Predicting Stock Prices*

a. *Predicting the Adjusted Closing Stock Prices*

For this experiment, we worked with the merged data frame of all three .csv files presented in the dataset, which gave us information on individual company characteristics, their stock prices, and the S&P index dating back to 2012. Using a subset of our dataset (200,000 instances), we used Linear and kNN regression methods to predict the closing stock prices.

For each prediction, we were sure to drop *all* potential target columns, as we did not want to overfit the model. For example, when predicting the adjusted closing price, we dropped the following features: *Close*, *High*, *Low*, Open, and *Currentprice* columns.

This experiment included two trials, as we performed both regression methods on the adjusted closing price prediction. For each trial, we completed a 5-fold cross validation on the model, as well as calculated several performance metrics based on the model's ability to predict the prices. Below are the average results over the 5 folds.

*Evaluation metrics for predicting Adjusted Close over entire dataset:*

| Regressions | Linear | kNN (k=1) | kNN (k=3) | kNN (k=5) |
|---|---|---|---|---|
| $R^2$ | 0.812 ± 0.022 | 0.877 ± 0.0598 | 0.940 ± 0.005 | 0.946 ± 0.002 |
| RMSE | 33.222 ± 5.802 | 25.545 ± 3.085 | 19.085 ± 0.700 | 18.006 ± 0.447 |

| Neural Networks | Activation = 'relu' | | Activation = 'tanh' | |
|---|---|---|---|---|
| | Layers=2 Size=100 | Layers=5 Size=100 | Layers=2 Size=100 | Layers=5 Size=100 |
| $R^2$ | 0.996 ± 0.000 | 0.996 ± 0.000 | 0.996 ± 0.000 | 0.991 ± 0.007 |
| RMSE | 4.879 ± 0.240 | 5.264 ± 0.486 | 4.901 ± 0.266 | 6.860 ± 2.687 |

*b. Predicting Individual Companies' Stock Prices*

For the following experiments, we started with the same dataset as in the previous section, however we wanted to look at specific companies, so we created two new data frames containing only data from Google and Twitter stock, respectively. We chose these two companies specifically because we wanted to see how much stock volatility would affect the model's performance. Twitter's stock is much more volatile than Google's. The results from running a 5-fold cross validation on all of the following models are presented below.

*i. Evaluation metrics for predicting GOOG stock*

| Regressions | Linear | kNN (k=1) | kNN (k=5) | kNN (k=11) |
|---|---|---|---|---|
| $R^2$ | 0.972 ± 0.002 | Neg. $R^2$ | 0.073 ± 0.041 | 0.156 ± 0.036 |
| RMSE | 5.876 ± 0.201 | | 33.936 ± 0.990 | 32.294 ± 1.192 |

| Neural Networks | Activation = 'relu' | | Activation = 'tanh' | |
|---|---|---|---|---|
| | Layers=2 Size=100 | Layers=5 Size=100 | Layers=2 Size=100 | Layers=5 Size=100 |
| $R^2$ | 0.988 ± 0.001 | 0.988 ± 0.001 | 0.989 ± 0.001 | -0.003 ± 0.004 |
| RMSE | 3.789 ± 0.121 | 3.850 ± 0.112 | 3.672 ± 0.135 | 35.266 ± 1.292 |

*ii. Evaluation metrics for predicting TWTR stock*

| Regressions | Linear | kNN (k=1) | kNN (k=5) | kNN (k=11) |
|---|---|---|---|---|
| $R^2$ | 0.243 ± 0.015 | Neg. $R^2$ | Neg. $R^2$ | Neg. $R^2$ |
| RMSE | 12.235 ± 0.247 | | | |

| Neural Networks | Activation = 'relu' | | Activation = 'tanh' | |
|---|---|---|---|---|
| | Layers=2 Size=100 | Layers=5 Size=100 | Layers=2 Size=100 | Layers=5 Size=100 |
| $R^2$ | 0.545 ± 0.012 | 0.551 ± 0.043 | 0.612 ± 0.027 | -0.001 ± 0.001 |
| RMSE | 9.493 ± 0.263 | 9.420 ± 0.499 | 8.751 ± 0.222 | 14.077 ± 0.309 |

**Conclusions**

*Experiment 1: Forecasting*

Since we are dealing with historical data, we wanted to try out some level of forecasting based on what we learned about autoregressions during lecture. We decided to do it by hand so as to not venture outside of the packages we learned in class. The results from our experiments were sensible. The RMSE was 32.313 and the MAE was 19.887. With S&P values of 1400-3000+, this error seems reasonable, and we were able to conclude that our model was able to forecast the S&P 500 with decent accuracy. Now, someone can use our model to make forecasts based on the previous week of S&P activity!

*Experiment 2: Predicting Stock Prices*

a. *Predicting the adjusted closing stock prices*

According to our results from this portion of the experiment, the kNN model performed better than the linear regression model. In fact, all three kNN trials, each with a different K value, performed better. The average difference between the two models' $R^2$ and RMSE values were 0.109 and 12.343, respectively. The kNN implementation that performed best was the one where K = 5. We believe this is because as K increases, the model takes more data into account, which helps improve its predictions. However, we learned in lecture that after a certain number of increases to K, the model's score will eventually begin to dip.

Our results from this experiment also show that the neural network models performed significantly better than both the linear regression and kNN models. The average $R^2$ and RMSE values were 0.995 (0.101 higher than the average $R^2$ values of both the linear regression and kNN models) and 5.476 (18.489 lower than the average RMSE values of both the linear regression and kNN models), respectively. Additionally, we observed that the 2 layered "relu" implementation performed the best, but we were very surprised to see that the 5 layered "tanh" implementation performed worse than the other implementations (the $R^2$ was 0.005 lower than the other implementations, while the RMSE was 1.845 higher than the average of the other RMSE values).

*b. Individual company prices*

The first thing we realized in this experiment was that the models were far better at predicting Google's stock than they were Twitter's. Due to this we concluded that volatility does in fact have an impact on the ML model's ability to predict stock prices – across all models implemented. This conclusion is intuitive because a model can only be as good as the data that it is trained on, therefore a stock with more stable prices, such as Google, would be easier to predict.

Again, the neural networks outperformed all of the regression models. Based on the results above, the 'tanh' activation function with 2 layers of size=100 performed the best for both companies. In the case of GOOG stock, the $R^2$ score was very high at .989. The more volatile TWTR stock had a more modest $R^2$ of 0.612 (which is still a great improvement from the linear regression with an $R^2$ of 0.243). Something weird happened with the 'tanh' activation function neural network at 5 layers, in both companies. We suspect the number of layers was too large and we overfit the model.

The kNN models performed really poorly with predicting these individual companies. We were not expecting such poor results from these experiments. However, we know kNN does not perform well with high dimensional data. With the number of features presented to the algorithm, it may have become very computationally expensive and impacted the performance of the algorithm. Furthermore, we also know kNN is sensitive to noisy data and missing values. Perhaps imputing missing values and addressing any outliers could have improved the performance of this algorithm.

Although we were not able to make millions from analyzing this dataset, we did learn a few things and reinforce some important concepts learned throughout the course of the semester. Our first major conclusion is that neural networks are the state of the art algorithm for many, if not most, machine learning tasks. It was proven through the results of all of our experiments that it was best at predicting stock prices.

Another thing we learned was that creating accurate models to abstract real world phenomena, such as the stock market, is very complex! As beginners to the subject, there were many times we wanted to try something on the dataset, but realized how out of scope it was for our introductory knowledge of machine learning. As mentioned earlier, if this was simple, we would all be rich. The simple ML models learned over the semester certainly cannot predict the next market crash or the next hot stock, but we hope our solutions will provide anyone with vested interest in the stock market with a couple of key insights. First, if you want to use ML models to predict stock prices, you are better off experimenting with stable stocks. Second, you will certainly need more complex models to do more sophisticated predicting/forecasting. However, it is our hope that we have laid the groundwork for others to build upon our research and perhaps even make a little money in the future.

**Individual Contributions**

| | |
|---|---|
| Matt Hargenrader | ● Authored the introduction section<br>● Authored the dataset section<br>● Collaborated on feature selection and preprocessing<br>● Performed Linear, kNN Regression, and Neural Network algorithms on data<br>● Collaborated on methods, models, and results section<br>● Collaborated on conclusion section |
| Caitlin Hermann | ● Authored the business objective section<br>● Authored the preprocessing steps section<br>● Collaborated on feature selection and preprocessing<br>● Performed Linear Regression, Autoregression, and Neural Network algorithms on data<br>● Collaborated on methods, models, and results section<br>● Collaborated on conclusion section |