# Predicting the Future Returns of the S&P 500

Introduction to Machine Learning

Josh Rohauer
UNC Charlotte
Charlotte, NC
jrohauer@charlotte.edu

Cristian Salitre
UNC Charlotte
Charlotte, NC
csalitre@charlotte.edu

Aiden Lamar
UNC Charlotte
Charlotte, NC
alamar2@charlotte.edu

Charlie Monroe
UNC Charlotte
Charlotte, NC
cmonro12@charlotte.edu

***Abstract*—We propose the development of a binary classifier using data from the US Federal Reserve website to predict the monthly overall market state (up or down). We will use the S&P 500 monthly returns as our market proxy, and we will include a variety of market features, such as economic data, sentiment/survey data, and commodity data. We aim to use logistic regression to predict the directional movement of the market in consecutive future months. We will use three common machine learning techniques to tackle this problem: SVM (Support Vector Machine), FCN (Fully Connected Neural Network), and CNN (Convolutional Neural Network). We then compare the results of these three approaches after optimizing the models for precision and accuracy.**

## I. Introduction

Predicting the market has always been the primary goal of financial analysts. The effectiveness of machine learning algorithms has improved exponentially over the past decade, and now have the opportunity to aid analysts by adding another tool to help in their forecasts. Currently, analysts look at a broad range of market data to make predictions on the direction of markets. In this project, we will use the most popular market data as ranked by the St. Louis Federal Reserve Bank, as well as some well-known market indexes, to try and predict the direction of the S&P 500 stock index. We first train an SVM classifier model (also known as an SVC), and then compare the results to those of a FCN and CNN. We will optimize these models for real-world use, and compare the models via precision and accuracy.

## II. Motivation

The S&P 500 is a stock market index tracking the performance of the largest 500 companies in the United States. It is one of the most commonly followed indexes and is usually what people mean when they reference "the market". A vast majority of investors' retirement wealth is tied up in the stock market, and as they get close to retiring, their goals shift to wealth preservation. Our model's intent is to give investors another tool to help achieve their investment goals. Our models aim to classify the market as either "up" or "down" the following month after market data has been collected using the changes in market data month over month, as well as the overall fiscal cycle. Having a model that could confidently predict the direction of the market would lower the risk and/or increase the expected returns of an investor essentially shifting the efficient frontier of a portfolio to the left.

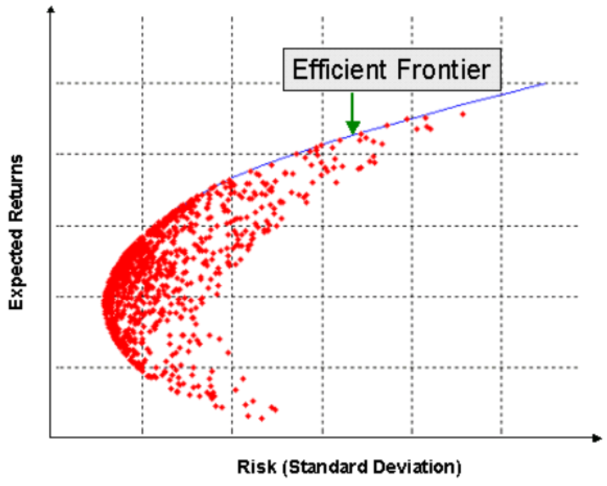

Figure 1. Efficient Frontier

## III. Approach

Our first step is to collect data. The St. Louis Federal Reserve has over 800,000 data points, and selecting the correct data is a task unto itself. We selected 58 features from this dataset, chosen by the data with the highest search frequencies. We assume that if market participants are looking at this data the most, then it is likely useful for investment decisions. Our data is a time series, which is made up of month-over-month changes to eliminate any non-stationary of the data. We then used the performance of the S&P 500 for the 'next month' as our target variable, and defined it as a binary 'up or down' using the softmax classification function. Any missing data was zeroed out. To format the date appropriately, it was first split into year and month columns. We then experimented with two different methods to retain the cyclic nature of the month feature. The first method tested was taking the cosine of the month number, multiplying it by pi, and dividing by 12. This value would replace the month feature in our explanatory variables. The second method was 'one-hot encoding' which aims to capture the natural ordering of categorical data. This method adds a new feature for each category element. If the month of the input data was January, we used 1 as the first new feature and 0 for the remaining 11. We then normalized our data using the standard normalization on a normal curve. The market usually has outlier data and we wanted to keep the high standard deviation events away from the center of our data distribution.

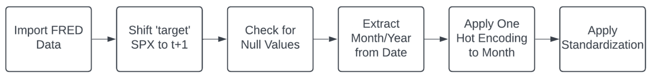
Import FRED Data
Shift 'target' SPX to t+1
Check for Null Values
Extract Month/Year from Date
Apply One Hot Encoding to Month
Apply Standardization

Figure 2. Preprocessing Flow Chart

## IV. Dataset and Training Setup

The dataset consisted of 58 features from both the St. Louis Federal Reserve as well as Yahoo Finance. The training for this data was done with three different models, where we iterated through their parameters to optimize for precision and accuracy. Precision was used as the first optimization parameter as we wanted to minimize false positives. A false positive in our result means that our model predicted the market to be "up" when it was actually "down". Investing off of this result would lose the investor money. We then optimized for accuracy as we wanted to test our model for the number of correct classifications.

### A. Dataset

The dataset consisted of four major market categories. Economic data measures the performance of an entire area of a given market, and it is a real-time look at asset prices established by supply and demand. Sentiment data is survey data across different segments of the market as to how investors see the economy regarding its future trends. Commodity data includes prices of hard assets.

Table 1: Dataset Composition

Dataset Constituents – 58 Features
Years 2000 - 2023
Over 15,000 data points

| **Data Type** | **Fields** | **Examples** |
|---|---|---|
| Economic | 16 | Inflation / GDP / House Prices |
| Market | 25 | Stock Indexes / Yields / Volatility |
| Sentiment | 7 | Consumer / Bank / Recession |
| Commodity | 10 | Gold / Oil / Lumber /Currency |

Source: St. Louis Federal Reserve , Yahoo Finance

### B. Training Setup

For the SVC model, we applied GridSearchCV to test different model hyperparameters and find an optimal model. One parameter was PCA components, which can reduce the feature space of the data by finding dependencies. Another parameter was the SVM kernel (polynomial, radial basis function (rbf), linear, and sigmoid), used to test for non-linearities by mapping the features to a higher dimension. We also tested different margin parameters (C parameter) as well as the parameter for the kernel coefficient, gamma. We also tested various degrees for the polynomial kernel. The neural network model was also tested over a grid of different hyperparameters. We test a varying number of hidden layers, learning rates, optimizers, and dropout probabilities. We also tested these parameters for the convolutional neural network, with the addition of convolution and max pooling layers to capture any locality within the input features.

## V. Results and Analysis

The expectations for our models were that the neural network models were going to perform better than the SVC model, since the stock market is chaotic and capturing non-linearities would be key to describing it in any meaningful way.

### A. Support Vector Machine Classifier

The SVC GridSearch was used to test a total of 768 different models. The model that achieved the best precision and accuracy had a C (margin) value of 1000, a gamma value of 0.01, and used the rbf kernel. The results for precision vs accuracy had a different number of principal components so a graph was created to model all scoring metrics over 1 to 9 principal components. As shown in Fig. 3 the highest precision and accuracy was with 8 principal components. Using these hyperparameters, we achieved a precision of 71.4%, an accuracy of 60.1%, a recall of 66%, and an F1 score of 0.69.

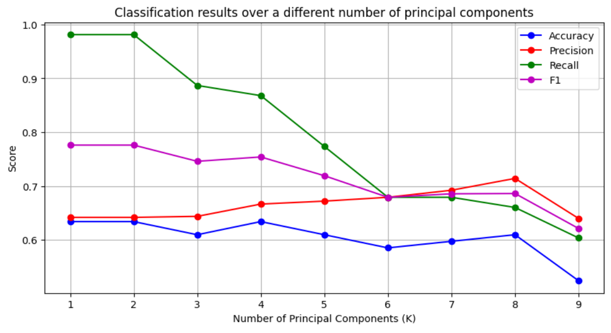

Figure 3. Performance of varying PCA components

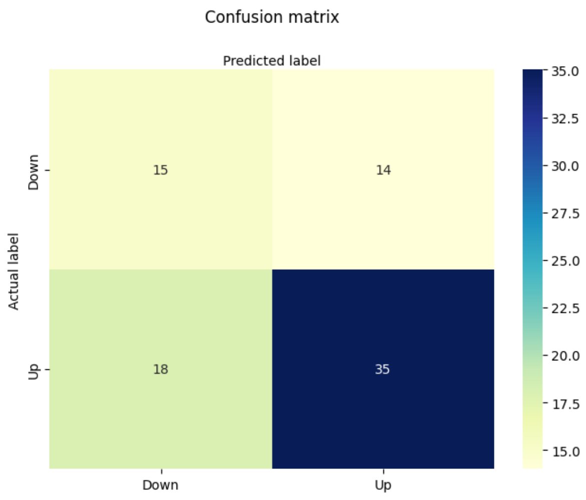

Figure 4. SVC Confusion Matrix

## B. *Fully Connected Neural Network*

The FCN introduces non-linearities through an activation function to help capture the complex relationships in the feature map. The approach we took for the FCN involved performing extensive hyperparameter tuning, which involved experimenting with various learning rates, activation functions, optimizers, and layer sizes. Dropout regularization was implemented as needed to help improve the result. The search for hyperparameters resulted in 1,152 models. The optimal model was chosen manually after testing all parameter combinations sorted by both precision and accuracy. Manual selection was necessary as some models produced a very high precision but with low accuracy.

The first model chosen generated fewer false positives compared to the classical (SVC) model and consisted of 2 hidden layers with sizes of 16 and 32. The dropout probability was set to 0.6, and we used the Adam optimizer and Tanh activation function. This model had a precision of 77.1% and an accuracy of 69.1%. The confusion matrix in Fig. 5 shows improved performance over the SVC model.

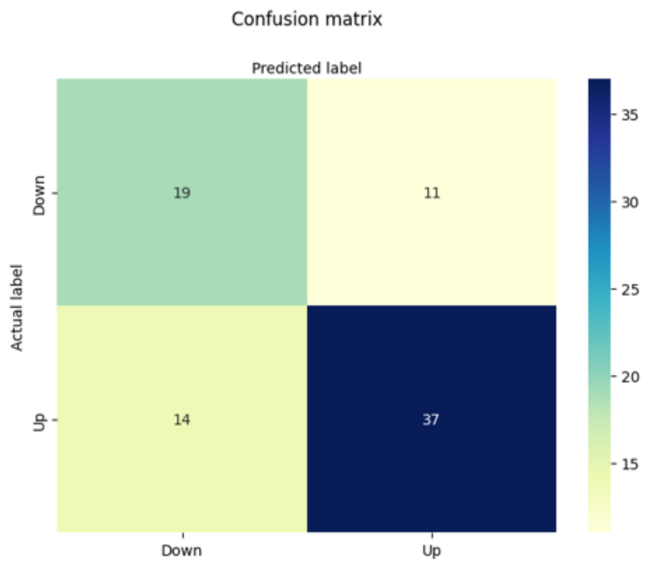

Figure 5. FCN Confusion Matrix (timestep: 1M)

Additional testing was also carried out after receiving feedback during the presentation. This testing involved implementing the timestep feature that had previously only been used for the CNN model. A sliding window was used to group features into 6 consecutive months, where the new target variable was the SPX performance for the month following the group. By flattening this data, it could also be used for the FCN. The idea behind this change is that the model may have a better chance to learn any time-based patterns leading up to the target month.

The results from the new model, along with an 80/20 training and test split, had improved precision to 82.8% and accuracy to 71.7%. This model had three hidden layers of size 1024, 2048, and 512. The total number of parameters was 3,526,146. This is rather large for an FCN and demonstrates the scale at which the parameters grow as the network gets more complex. The confusion matrix in Fig. 6 shows the outstanding precision of this model.

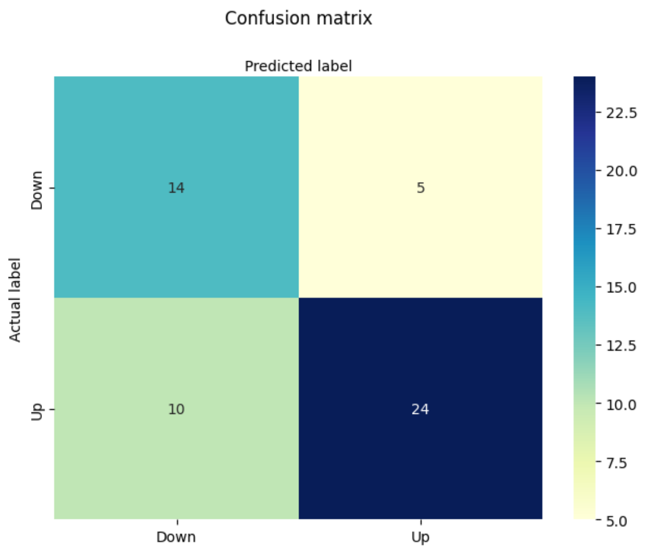

Figure 6. FCN Confusion Matrix (timestep: 6M)

## C. *Convolutional Neural Network*

While the FCN achieved favorable results, testing was also done with a CNN to see if any additional benefit was possible. With the use of convolution and max pooling layers, the feature space could be expanded, while the number of trainable parameters remained small. For this network, it was decided that the input features would be the number of filters in the input of the first convolution layer. This leaves the time dimension to be condensed by the max pooling layer.

Using timesteps of 6 months, the best model had 2 convolution layers, the first with 32 filters, and the second with 64 filters. The fully connected layer had 32 nodes and the best optimizer and activation function was Adam and Tanh, respectively. This model had a precision of 75% and an accuracy of 69.8%. These results, shown in Fig. 7, are not an improvement over the FCN.

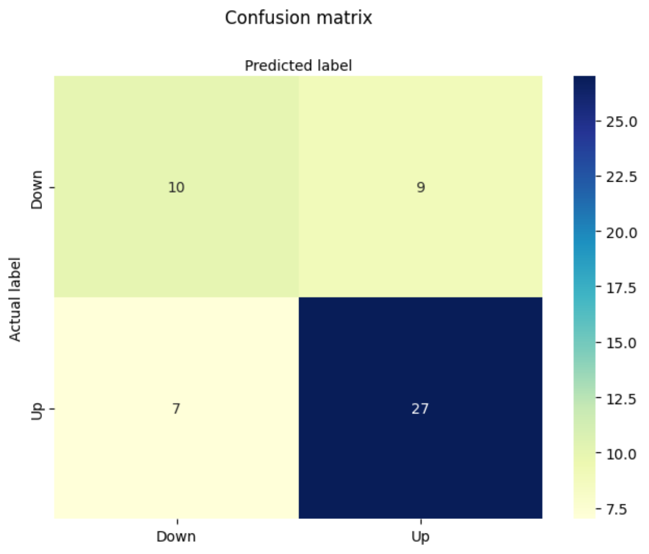

Figure 7. CNN Confusion Matrix (timestep: 6M)

One potential issue with this design is the two max-pooling layers with kernel size 2 over the time series of 6 months. To better adhere to the halving from the max-pool, the timesteps were increased to 8 months. As illustrated in Fig. 8, this new model improved the precision to 84.9% and the accuracy to 71.7%. Compared to the FCN, this model had higher precision and the same accuracy. While they both had the same number of false positives, the CNN had more true positives, and the FCN had more true negatives.

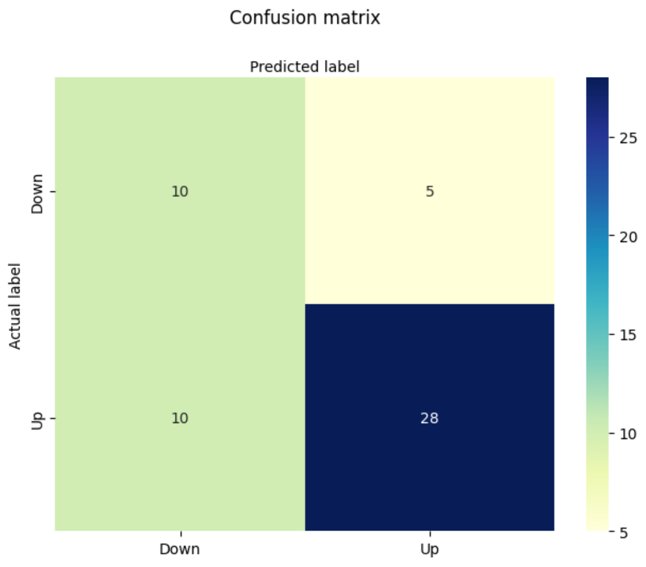

Figure 8. CNN Confusion Matrix (timestep: 8M)

## D. *Results Summary*

As expected, the SVC model, which had the least amount of parameters, performed the worst as measured by both precision and accuracy. This model has challenges when trying to capture non-linearities in the explanatory variables, and is limited in size to keep the total number of parameters reasonable. The fully connected neural network performed much better by adding multiple hidden layers, non-linearity through activation functions, and optimizers to the model. Both the precision and accuracy scores saw significant improvements over the SVC model. The convolutional neural network showed promise as well, with results very similar to that of the FCN, and with 82 times fewer parameters. The final model chosen was the CNN as it had the best precision. While the FCN was able to correctly predict more of the down market cases, the overall goal was to minimize the number of false positives as this has the potential to lose investor's money.

Table 2: Results Summary

| Model | False Positives | Precision | Accuracy | Recall | F1 | Parameters |
|---|---|---|---|---|---|---|
| SVC | 14 | 71.4% | 60.1% | 66.0% | 0.69 | 8 |
| Fully Connected Neural Network | 5 | 82.8% | 71.7% | 70.6% | 0.76 | 3,526,146 |
| Convolutional Neural Network | 5 | 84.9% | 71.7% | 73.7% | 0.79 | 42,930 |

## VI. Lessons Learned

Predicting the direction of the market is a challenging endeavor. Going into this project, we knew we would probably not achieve a model that could accurately predict the market with a high probability, so we focused on getting the best results possible. The first challenge was finding the correct data. There are hundreds of thousands of market data points available but they are not centralized. Some of these data points also do not have a long history so finding the right data with a long enough history is the first place these models can be improved upon. Our next challenge was model complexity. As machine learning can be more art than science, we iterated through many models testing all combinations of hyperparameters. This training took a large amount of time and any changes to the preprocessing or model code structure required rerunning all tests. Another issue faced during this project was overfitting. As the models got more complex, they were better able to memorize the training set. Dropout and batch normalization were added to help mitigate this issue. Our final challenge was unbalanced data. 63% of our observations were 'up' while only 37% were 'down'. This led to some biasing in the ranking of our results. The best models were handpicked to eliminate the outlier metrics where the models predicted either mostly up or mostly down. In conclusion, we believe our results are substantial given the chaotic nature of the stock market. With relatively high precision and decent accuracy, this model may prove to be a useful tool for investors.

## Github

[1] https://github.com/jrohauer/ECGR4105_Paper