Oliver Hugh
Bhavya Vira

EECE 5644 Final Project Report

EECE 5644
Final Project Presentation:
S&P 500 Closing Price Predictor

Oliver Hugh
Bhavya Vira

Our primary goal is to develop a predictive model for forecasting future S&P 500 closing stock prices based on historical data.

Our project code is available via GitHub:
https://github.com/Oliver-Hugh/SP500_Predictor/tree/main

Motivation and Scope

- The S&P500 index is a widely recognized benchmark that reflects the performance of 500 leading companies in the U.S. stock market. Its broad coverage makes it a key indicator of overall market health and economic trends.
- Our project focuses on historical data spanning several years, allowing us to analyze trends and patterns that may contribute to more accurate predictions.
- Project split into 2 parts
  - Binary Classification: Class 0 = Decrease, Class 1 = Increase
  - Time Series: Predict actual closing stock prices
- Classification:
  - Different Linear Discriminant Analyses
  - Maximum Likelihood Estimation
  - Naïve Bayes
  - Naïve Bayes 7 days out
- Time Series
  - Linear Regression
  - Long Short-Term Memory (LSTM)

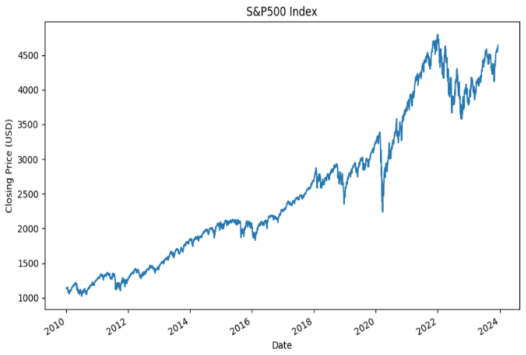
Financial markets exhibit inherent volatility. The S&P 500 is a very popular index fund in the US that includes stock of 500 leading companies in the economy, which makes it a very good case to examine, as it generalizes nicely to the economy at large. Navigating market fluctuations and being able to predict movements and patterns is key for high returns when it comes to investing. Such foresight enables investors to make judicious decisions regarding stock transactions—be it buying, selling, or holding. Predictive capabilities empower investors to optimize financial portfolios, unlocking strategic advantages in an ever-changing landscape.

In the pursuit of improving stock forecasting abilities and making good investments, we turn to the integration of machine learning and data science techniques in finance. Our report delves into the innovative realm of predictive modeling, exploring how these advanced techniques, both taught in class and beyond the scope of the course, redefine the landscape of financial decision-making. Through a comprehensive analysis, we unravel the potential of machine learning to provide nuanced insights and a strategic edge to investors in an environment defined by uncertainty.

The project employs two overarching categories of techniques: Binary Classification and Time Series Prediction. In the binary classification part of the project, class 0 is defined as a decrease in value from opening to closing price, and class 1 as an increase. The algorithms explored in this section include Linear Discriminant Analysis, Maximum Likelihood Estimation, and Naive Bayes Classification. On the other hand, in the Time Series domain, we utilize Linear Regression and the Long Short-Term Memory (LSTM) algorithm to predict numerical values for the closing stock price.

Classification: Getting and Manipulating Data

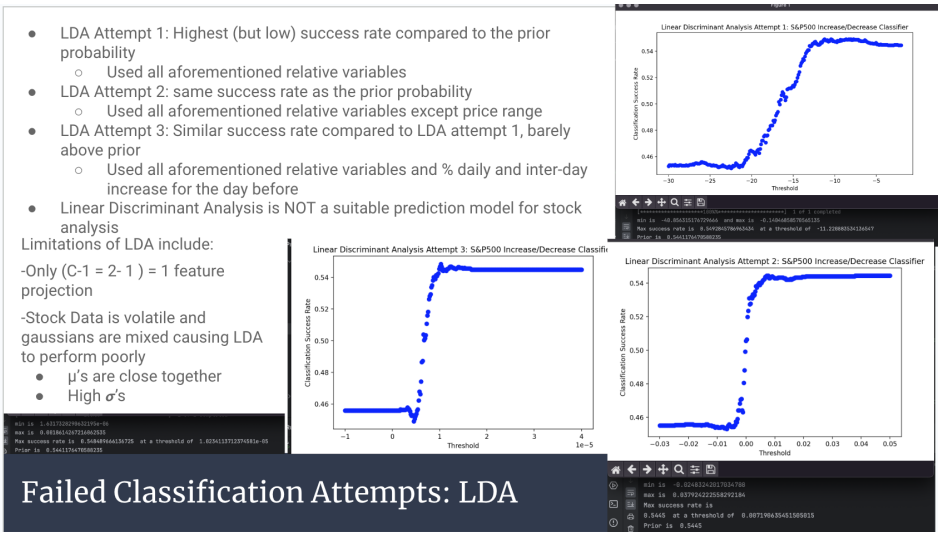
- The is fetched historical S&P500 data from Yahoo Finance using the yfinance library
- Data is filtered based on a specified time frame
- The DataFrame is then converted into a relative array, including
  - Daily percent increase
  - Inter-day percent increase
  - Normalized volume
  - Price range
- The data used ranges from 1st January 2010 to 12th December 2023



The historical data for the S&P 500 data was obtained from Yahoo Finance utilizing the yfinance library. Subsequently, the dataset was filtered based on a user-defined time frame, ensuring relevance to the intended analysis. Our workflow involves converting the DataFrame into a numpy array. This transformation is a pivotal step, enabling the derivation of key relative data crucial for subsequent analyses.

From the processed numpy array, we computed essential relative variables, including daily percent increase, day-to-day percent increase, normalized volume of traded stock, and price range. These variables are fundamental to understanding and predicting market dynamics, forming the basis of our classification models. Furthermore, this was essential to find relative variables for classification, to account for the typical growth of the market over time. The dataset spans from January 1, 2010, to December 12, 2023, encompassing a significant time frame that allows for a comprehensive analysis of market trends and patterns.

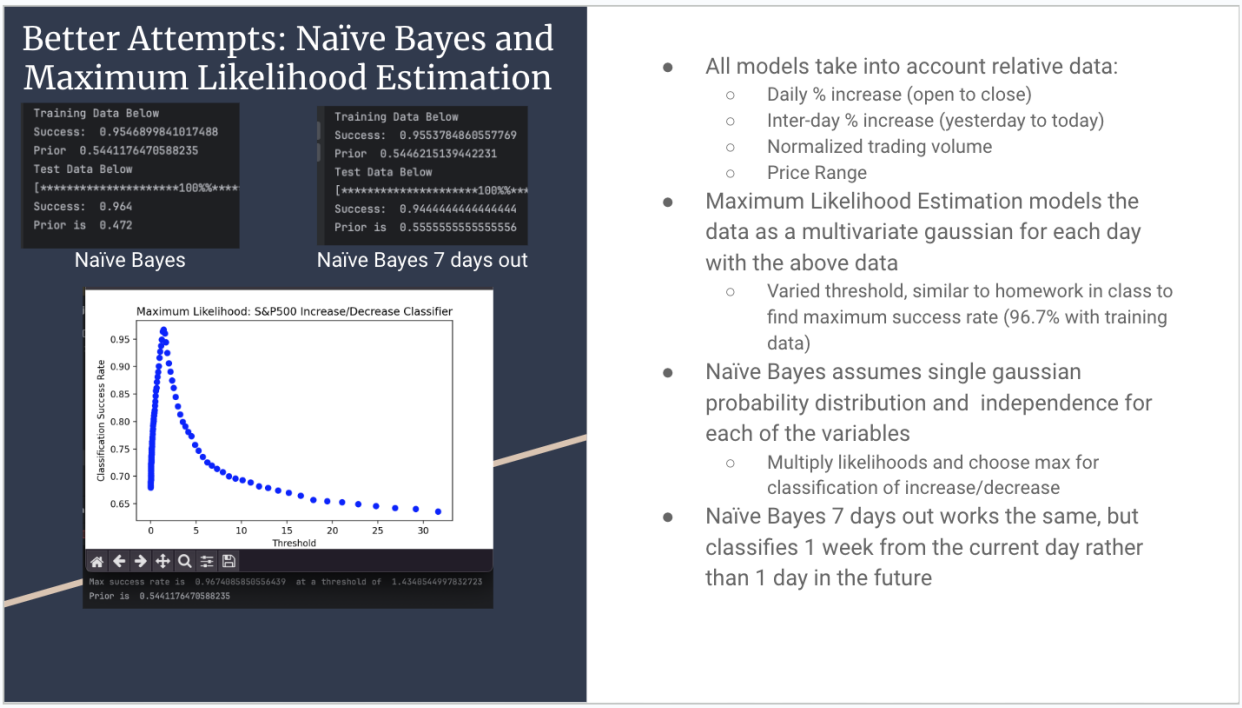
The groundwork laid in this phase establishes a robust framework for the subsequent stages of our project. The modular design of our functions and the comprehensive dataset provide the flexibility required for various analyses and model implementations. This approach sets the stage for our exploration into classification algorithms, specifically tailored to S&P 500 stock data. In conclusion, the meticulous extraction and transformation of historical S&P 500 data form the bedrock of our project. This groundwork positions us for a detailed exploration of classification models, utilizing the derived variables to gain insights into market movements and trends.



The first model employed for the classification part of this project were three different linear discriminant analysis (LDA) models. LDA, as discussed in class, is a method/model of classification in which the data points for classes are projected onto an optimal vector that maximizes the distance between class means, normalized with respect to their variances.

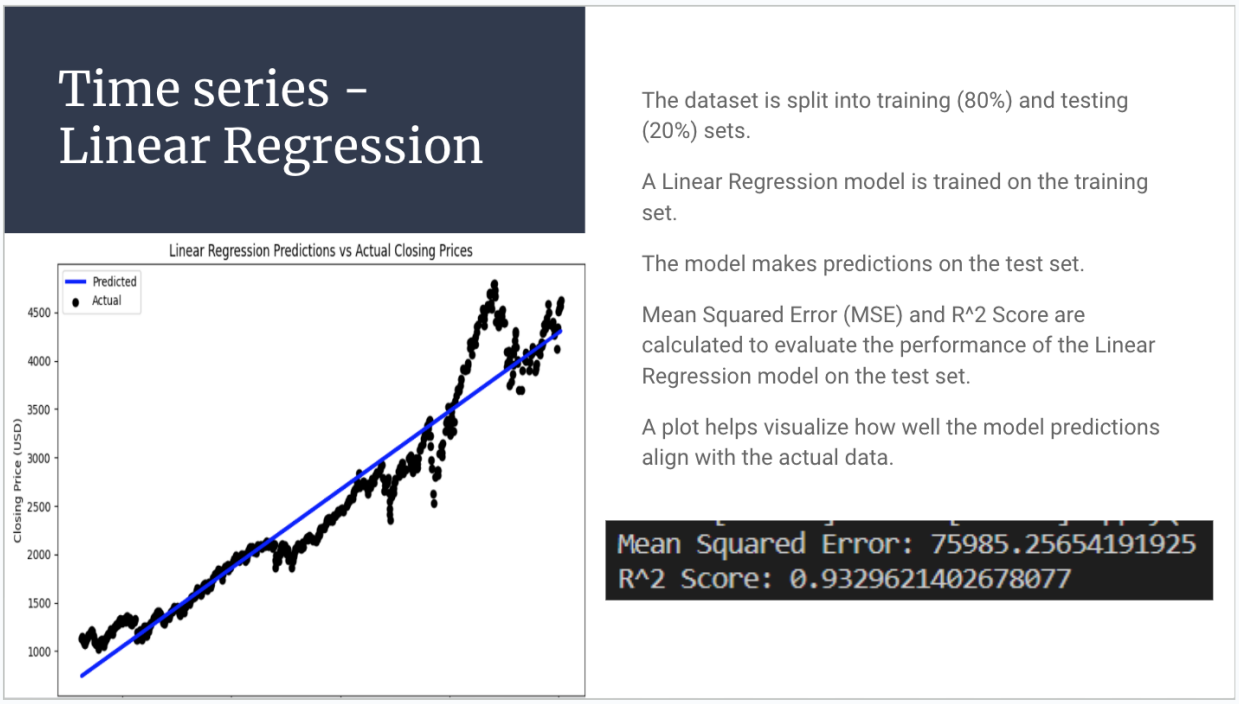
As can be seen on the slides, three different LDA models/classifiers were constructed. The first uses all of the relative variables previously derived and discussed in this report: daily percent increase, inter-day percent increase, normalized volume, and price range. The threshold on the calculated optimal vector was varied for all 3, and a plot was produced of the threshold vs. the classification success rate when the models were fed the training data. The prior probability for the number of days where the closing price increased was about 54.4% for the three models. The first model had a classification success rate of 54.9%. The second model used all of the aforementioned relative variables except for the price range, and its success rate was not better than the prior probability. The last model took into account all of the aforementioned variables, as well as the percent daily increase and the percent inter-day increase for the day prior to the target date. This model had a success rate of about 54.8%.

Clearly, these classifiers do not say much about the data, as they are only marginally, if any better, than simply classifying everyday as an increase. This suggests LDA is a terrible classifier for predicting if a stock's price will increase or decrease. This makes sense due to the nature of the data as well as the drawbacks of LDA. The stock data is very volatile, with high variances and somewhat close means. LDA is inefficient for highly mixed datasets, as there will not be distinct separation between class data points on a 1D vector. Additionally, since there are only 2 classes, there is only 1 feature projection. Other models will better take into account the effect of the different features of the data, and their effect on the closing price.



The next two models fared far far better and were much more accurate. First, a maximum likelihood estimator classifier was implemented, which modeled all of the previously discussed relative variables as components of a multivariate gaussian distribution. Just as before, the threshold for the decision boundary for the proportion of posterior probabilities was varied, and the corresponding percent accuracy was plotted on the y-axis. Note that this is not an ROC curve, but instead the threshold of the decision boundary vs. the classification accuracy, similar to the plots provided for previously discussed models. The prior probability was similar to before at about 54.4% for the training data. For the MLE model, the classifier was able to correctly classify the training data 96.7% of the time.

Two Naïve Bayes classifiers were also constructed. The first, similar to the rest of the classifiers, classified the following day as an increase or decrease. The second, unlike the others, looked farther ahead and classified what class (increase or decrease) 7 days in the future would be. Both of these classifiers assumed that each variable, of the ones discussed, was a single variable gaussian distribution and that all of these variables were independent. It determined the likelihood, based on the gaussian distribution for all of these variables, multiplied them all, along with the estimated priors, and decided class 0 or class 1 based on which of these products were higher. For future work, where there may be far more data used/processed, it might be advantageous to instead compute the log-likelihood and compare these values. The normal Naïve Bayes classifier had an accuracy of 95.4% with its training data, and 96.4% with its test data. The 7-day out one had an accuracy of 95.5% with its training data and 94.4% with its test data. The greater prior probability between class 0 and class 1, for the 1-day and 7-day classifiers, were 52.8% and 55.5%, respectively. This suggests that both the Naïve Bayes and MLE classifiers are fantastic classifiers for this data and powerful models based on their respective classification goals.

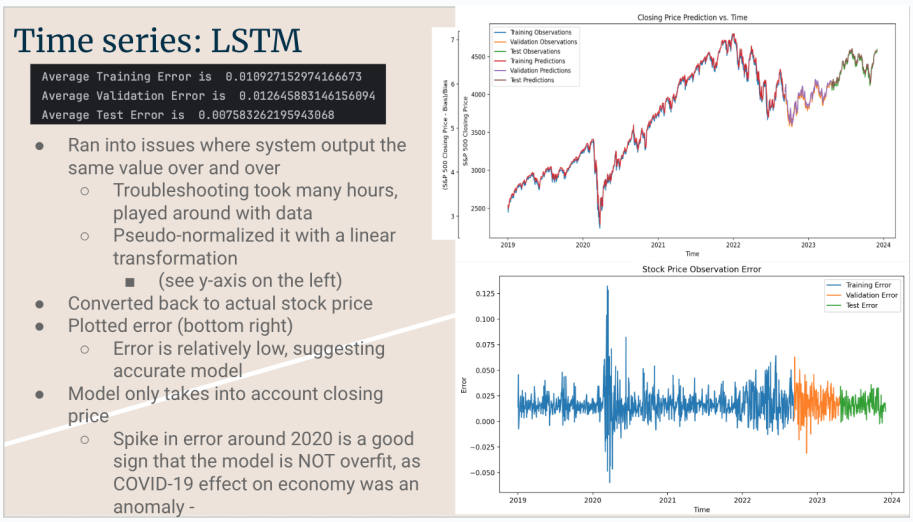


For linear regression training, the data was preprocessed by resetting the index and converting the 'Date' feature to Unix timestamp format. The model was then trained using scikit-learn's Linear Regression algorithm, with 'Date' as the independent variable and closing prices as the dependent variable. The dataset was split into training (80%) and testing (20%) sets to evaluate the model's performance.

The evaluation metrics provide insights into the model's accuracy. The Mean Squared Error (MSE), measuring the average squared difference between predicted and actual closing prices, was found to be 75985.25. Additionally, the R^2 score, a measure of the proportion of variance in the dependent variable explained by the model, yielded a high value of 0.9329. These metrics collectively indicate a strong predictive capability of the linear regression model.

To visually assess the model's predictions, a scatter plot was generated, illustrating the alignment of actual closing prices with the predictions. The blue line on the plot represents the linear regression line fitted to the test data.

In conclusion, the linear regression model demonstrates robust predictive performance, suggesting that the closing prices of the S&P 500 index can be effectively predicted based on the provided time frame. Further refinements or analyses could be explored to enhance the model's accuracy or adaptability to evolving market conditions.



The second model used for this project's time series prediction of the closing price of the S&P 500 closing price stock was Long Short-Term Memory, or LSTM. This was accomplished using functions and methods available through tensor flow libraries. LSTM was not touched upon in the course, but the hope was to tackle a challenging problem for the final product of this project. Very simply explained, LSTM is a type of recurrent neural network (RNN) which is able to both take into account long term data that is relevant and also 'forget' such long term data as it becomes obsolete with new incoming short-term data, with differing weights.

Even with the help of readily available/open source machine-learning libraries, this code took a very long time to get working. The program creates a modified dataframe of the closing price of the S&P 500 stock and date. It then goes through and selects a variable number of days (set to 5 for the final product displayed) before the target date. So, for each day, a numpy array including the closing price of the current day, as well as for the past window_size days is fed into the LSTM model to predict the closing price for the next day. For many hours of debugging, the model kept producing the same number over and over again. Eventually, through troubleshooting and examining different datasets, a linear transformation was performed on the closing price data. The first datapoint in the set was set as the bias, and each point was transformed to be (itself -bias)/bias.The LSTM model was able to handle this pseudo-normalized data far better than the raw closing price of the stock. For conciseness, just the graph of the re-transformed stock price predictions is included in the slides, but the scale of the pseudo-normalized y-axis is included to the left of this graph.

The error was then computed for every prediction made. This graph is included in the bottom left of the slide. As can be seen, the error is relatively low. The average error was also calculated, displayed in a screenshot of the console print-out at the top of the slide, and was found to be about 1.01% for the training data, 1.26% for the validation data, and .76% for the test data. This is very low and suggests that LSTM is a great model for predicting stock price!