Predicting stock price directon using 200+ financial indicators

Danayal Khan
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
b00069350@aus.edu

Shavaiz Khan
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
b00067567@aus.edu

Ahmed Riaz
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, UAE
b00070620@aus.edu

***Abstract*— The stock market is a market place where individuals can buy or sell shares of a company. The perdition of stock prices has received billions of dollars of investment as huge profits could be made for accurate predictions. In in this paper, we introduce the topic of stock price prediction, discuss the financial data that we use for our machine learning models, the different methods (such as data preprocessing, model selection) used, the classification methods used, and evaluation metrics of the various classifiers. The results we achieved will be tabulated and described).**

Keywords—component, formatting, style, styling, insert (*key words****)***

I. Introduction

The stock market is a marketplace that allows anyone to buy and share shares of companies listed on the market. There are different motivations for different people to buy and sell on the stock market; some would like to make a profit; others would simply like to have shares of a company that they believe in.

Predicting stock prices has seen an influx in investment with hedge funds and banks investing billions of dollars in state-of-the-art algorithms to maximize profits that could be made at the stock market. They use advanced machine learning and deep learning models that takes in a number of data points to predict the future price of a particular stock index. The data points are determined after analyzing different parameters such as a company's financial data which includes their revenue, gross profit, costs, debts, liabilities, etc., or by making predictions by looking at the past history of their stock price or a combination of both. The aim of any stock price predictive model is to achieve a high alpha; which is a parameter that determines how well a particular model does in comparison to a baseline model such as the S&P 500 Index. Alpha determines a financial trader's competitive edge and directly results in how much money can be made.

Can we even predict stock prices? There's been an age-long debate about whether predicting stock prices is even possible. Stock prices are influenced by real world and macroeconomic events. For example, due to the COVID-19 pandemic, the stock prices of pharmaceutical companies rose. But this cannot be determined just by looking at historical numerical data of stock. Researchers have suggested conducting sentiment analysis of news websites to determine the effects of real world events on stock prices.

Some researchers have trained their machine learning model on daily European opening and closing stock prices data and use this model to make predictions and trades on the American market since, due to the time difference, the American stock market closes at a later time.

The efficient market hypothesis claims that it is not possible to look at historic data of stock prices to predict future stock prices. The dataset that we are using describes a company's yearly performance and we aim to make predicts based on the company specific characteristics rather than their historical stock prices.

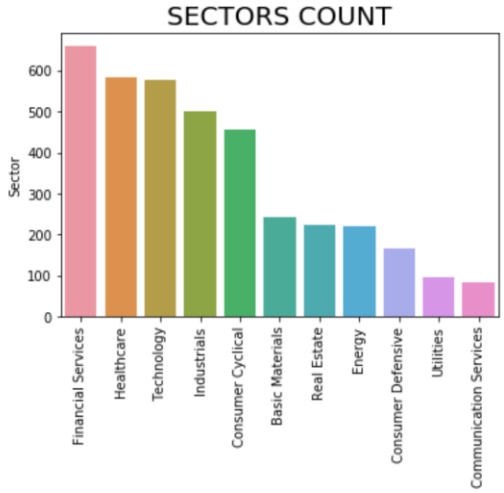

Figure 1: Sectors of stock indices

II. Literature Review

A review of previously conducted experiments was imperative for our project, as without the review we would not have been able to understand the exact goals we wanted to achieve. Predicting the actual value of a stock is a very difficult task, as it requires insight into the consumer market and also competing stock prices and histories.

Patel et al. instead focused on predicting the direction of stocks in the Indian stock market. They used Artificial Neural Networks (ANN), Support Vector Machines (SVM), Random Forests and the Gaussian Naïve Bayes classifiers. The authors used trend prediction data and assigned a 0 or a 1 to downward or upward movement respectively. By the end of the experiment, they achieved accuracies of 90% with SVM, Random Forests and Naïve Bayes classifiers [1].

The purpose [2] was to evaluate the use of stock price prediction using machine learning alongside natural language processing models (for news regarding the companies). The researchers compared the use of neural networks with machine learning based support vector machines (SVMs) in predicting the stock price movement direction. Furthermore, the authors detailed how to use libraries such as sklearn in order to preprocess data and also alleviate issues caused by missing data records. This is the main contribution of the article for our project, moreover, the results of the experiment proved SVMs to be the most accurate model for prediction methods and data sets.

Whilst our project uses numerical features in order to train our models, it is also a viable option to use online text sentiment analysis in order to predict stock price movement. Since online news articles can sway public opinion about a company and therefore also affect the stock price direction. This research article helps us in understanding alternate methods used to achieve the same goal that our project is striving for [3].

The purpose of this article is to determine which key factors affect the direction of the stock price. As our data set has more than 200 indicators, we will have to perform statistical analysis techniques in order to find the key indicators that will give our model predictions a high accuracy. According to the researchers, there are 5 key factors that can impact stock market prices the most. The key factors include Industry Performance, Market Influences, Company Performance as well as the Earnings per Share (EPS) [4].

The paper by Qian and Rasheed examined the performance of ANN, kNN and Decision Trees in predicting the direction of stocks from the Dow Jones Industrial Average. Individually, the error rates were 41.01%, 43.07% and 38.02% respectively, but they found that combining models reduced these error rates. The authors achieved error rates of 38.39% for ANN-kNN, 36.88% for ANN-DT, and 34.64 for ANN-kNN-DT [5]. Similarly, in [6], Patel, Shah, Thakkar, and Koteha combined Support Vector Regression (SVR), ANN and Random Forests (RF) to achieve a reduced error rate.

In [7], the authors extracted six features from a dataset of AAPL stock: Relative Strength Index (RSI), Stochastic Oscillator (SO), Williams percentage Range (W%R), Moving Average Convergence Divergence (MACD), Price rate of change (PROC), and On Balance Volume (OBV). They noted that computing GINI impurity with increasing windows of observation increased both accuracy and F-score.

Xianya, Mo, and Haifeng perform binary classification of stocks (rising or falling) according to stock price and other information. They compare four different algorithms: Naïve Bayes, Random Forests, Decision Trees, and Logistic Regression. They find that random forests and decision trees perform best, with the accuracy being 0.83 each, which the accuracy of logistic regression reached 0.80 and the accuracy of Naïve Bayes reached 0.75. On the other hand, the AUC (area under curve) of both Naïve Bayes and logistic regression was 0.50, which the authors note was no better than random classification. Random forests and decision trees had AUC values of 0.7 and 0.72 respectively [8].

Albanis and Batchelor examine the problem of classifying high-performance shares in the London Stock Exchange. They use the data from 700 companies, with predictions made over 1-year holding periods from 1993 to 1997. They use Linear Discriminant Analysis (LDA), Probabilistic Neural Networks (PNN), Learning Vector Quantization (LVQ), oblique recursive partitioning (OC), and a rule induction algorithm. The authors tabulate their findings for the individual algorithms, and it is seen that the poorest performer is LDA, which is followed closely by PNN and LVQ. The highest performing algorithms are the UVH (rules are unanimous, best), OC1, MVH (Majority rules agree). Even though LDA is the poorest performing, at 31.3% correct predictions of high performing shares it is not far off from PNN and LVQ (31.6%, 31.8%). But, when these are combined using rules, especially unanimous voting, the performance rises significantly to 37.9%, which in turn also increases profitability [9].

In [10], the authors compare Artificial Neural Networks with Support Vector Machines in predicting the trends of the Turkish stock market. The authors learned that using ANNs was not suitable for this application as ANNs do not perform well when the data is noisy or very inconsistent, and SVM outperformed BPN, random forest, neural network, and CBR algorithms respectively.

In [11] the authors compared the SVM algorithm with Linear Discriminant Analysis Quadratic Discriminant Analysis and Elman Backpropagation Neural Networks. The researchers found that SVM outperforms the rest of the algorithms when predicting stock price movements, however, much like [5] and [6], the authors found that a hybrid model of a combination of these different algorithm is the best-performing.

Jiao and Jakubowicz examine four different types of classification models in [12]: logistic regression, artificial neural network (ANN), random forest and gradient boosted trees. They used these models to predict the direction of 463 stocks, which are the constituents of the S&P 500 index. The authors find that logistic regression with lasso penalization performs better compared to random forests, boosted trees and neural networks. They also conclude that US markets can be better predicted with data with European and Asian markets, and that historic data cannot be used to accurately predict stocks, but very recent data can.

In [13], authors discuss and evaluate different techniques to predict stock prices. They state that there are unknown parameters such as election results and rumors that affect the price of a stock which makes predicting stock prices a very difficult task to conduct. The authors used historical stock price data from Yahoo Finance on Google Inc. (GOOG) and Yahoo Inc. (YHOO). The Machine Learning technique that worked the best was the Support Vector Machine with a Root Mean Square Error of 0.485. The authors state that using Textual Analysis of different news websites, such as the Wall Street Journal, and using a bag of words for text analysis, articles can be classified as positive, negative or neutral. Positive articles directly increase the price of the stock of the company in mention, negative articles decrease the price, while neutral articles do not have an effect.

[14] delves into combining multiple heterogeneous classifier model predictions. According to the author the best way of combining the classifiers is to use "majority voting" on the model predictions. This is a technique we can use for our project in order to ensure higher accuracies of the predictions as the results of the researchers experiments shows that those stocks on which all models predict to be profitable, are the ones that offer the most profitability. The classifiers combined are Linear Discriminant Analysis, Learning Vector Quantization, Probabilistic Neural Network, Recursive Partitioning and RIPPER Rule Induction. By the end of the article the researchers do conclude the using majority voting is not as viable as unanimous voting in terms of profitability.

In [15], the authors use a dataset from 9 years worth of Standard and Poors 500 index, which is a stock market index that measures the stock prices of the 500 largest US companies which are believed to affect the US economy the most. They use Boosting, Bagging, SVMs, kNN, Probabilistic Neural Networks, and CART. The authors also share the accuracy of the classifiers used and found the best to be boosting classifier.

III. Methods, Datasets and Results

A. Description of the data set

The dataset we have used in our research includes 224 financial indicators of stock (such as revenue, profit, loss, etc.), 3788 stock indexes, 11 different sectors of stocks, and an output price variation label that is 0 for negative growth and 1 for positive growth which makes it a classification problem: will the stock price increase or decrease? Figure 1 shows the number of different stock indexes classified into their sectors. Since stocks belonging to the same sector generally perform in a similar manner, KNNs could be a useful classifier in predicting stock prices. Furthermore, since our data is high in dimensionality, SVMs could perform well as well.

B. Preprocessing and model selection

To improve the performance of our models, we applied feature selection and dimensionality reduction to our data set. For feature selection, we used the methods of SelectKBest and SelectPercentile from the sklearn library. Principal Component Analysis (PCA) was used for dimensionality reduction.

To compare results, baseline models were run. All five of our models were trained with the help of Gridsearch, a hyperparameter tuning algorithm. The models were cross validated across 30 folds (cv = 30) except for SVM Non-Linear, which was run with cv = 10 due to long training times. These results were tabulated in a table, and the table was sorted according to precision, recall, testing and training score, and f-score.

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 1 With Scoring method recall	0.582854	0.539236	0.415677	0.888712	0.918644
Gaussian Naive Bayes RUN 3 With Scoring method recall	0.557445	0.529272	0.434273	0.884176	0.886850
Gaussian Naive Bayes RUN 2 With Scoring method recall	0.542192	0.521457	0.400995	0.885103	0.886598
KNN RUN 1 With Scoring method accuracy	0.611479	0.583688	0.579709	0.680198	0.643799
SVM NON Linear RUN 3 With Scoring method precision	0.617537	0.534076	0.465132	0.765432	0.637931

Figure 2: Baseline run of Gridsearch sorted by test score

1) SelectPercentile

SelectPercentile was used on the dataset before being input into the classification algorithms. Both f_classif and chi2 scoring methodologies were used, but the chi2 results did not perform well and as a result were disregarded. The top 1, 5, and 10 percentile of scoring features were selected and used for training.

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 1 percentile=5 With Scoring method accuracy	0.714330	0.568825	0.513412	0.626733	0.630607
Decision Tree Classifier RUN 2 percentile=10 With Scoring method accuracy	0.701110	0.564859	0.510951	0.626073	0.631926
Decision Tree Classifier RUN 1 percentile=10 With Scoring method precision	0.697838	0.554101	0.493001	0.776190	0.777778
Decision Tree Classifier RUN 2 percentile=5 With Scoring method precision	0.687941	0.566834	0.513712	0.694805	0.767123
Decision Tree Classifier RUN 3 percentile=5 With Scoring method precision	0.674234	0.576225	0.541532	0.664773	0.714286

Figure 3: Top SelectPercentile results sorted by precision

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111

Figure 4: Top SelectPercentile results sorted by recall

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751

Figure 5: Top SelectPercentile results sorted by F-score

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053

Figure 6: Top SelectPercentile results sorted by test score

As is evident from the results, the best performing classification algorithm was the Gaussian Naïve Bayes

classifier. The top 10 percentile of data was the ideal value, with which we achieved a 90% test accuracy. Even though the F-score is poor, using SelectPercentile definitely improved on it compared to the baseline run.

2) SelectKBest

SelectKBest is similar to SelectPercentile, but instead of the top scoring k percentile of features, we choose the k highest scoring features. Again, the f_classif function was used for the scoring, and k values of 5, 10, and 20 were used. More granular values were not used due to very long training times, as each algorithm was run with Gridsearch was run thrice.

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 2 k=5 With Scoring method precision	0.708477	0.561904	0.497743	0.630033	0.617414
Decision Tree Classifier RUN 3 k=10 With Scoring method f1	0.701767	0.559174	0.481807	0.635314	0.591029
Decision Tree Classifier RUN 3 k=5 With Scoring method precision	0.698355	0.552820	0.494244	0.624752	0.637203
Decision Tree Classifier RUN 1 k=10 With Scoring method f1	0.692689	0.550886	0.487470	0.629043	0.626649
Decision Tree Classifier RUN 1 k=5 With Scoring method accuracy	0.690814	0.559348	0.505724	0.626073	0.634565

Figure 7: Top SelectKBest results sorted by precision

Classifier	Precision	Recall	Fscore	Train score	Test score
KNN RUN 1 k=5 With Scoring method accuracy	0.637204	0.590014	0.577628	0.632013	0.645119
KNN RUN 3 k=10 With Scoring method precision	0.627876	0.587357	0.574398	0.633562	0.616352
Decision Tree Classifier RUN 1 k=20 With Scoring method precision	0.593963	0.587299	0.588245	0.700660	0.621372
Decision Tree Classifier RUN 1 k=10 With Scoring method recall	0.594889	0.585946	0.586047	0.679208	0.618734
KNN RUN 3 k=20 With Scoring method accuracy	0.614756	0.584473	0.576137	0.630693	0.631926

Figure 8: Top SelectKBest results sorted by recall

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 1 k=20 With Scoring method precision	0.593963	0.587299	0.588245	0.700660	0.621372
Decision Tree Classifier RUN 1 k=10 With Scoring method recall	0.594889	0.585946	0.586047	0.679208	0.618734
KNN RUN 1 k=5 With Scoring method accuracy	0.637204	0.590014	0.577628	0.632013	0.645119
KNN RUN 3 k=20 With Scoring method accuracy	0.614756	0.584473	0.576137	0.630693	0.631926
KNN RUN 3 k=10 With Scoring method precision	0.627876	0.587357	0.574398	0.633562	0.616352

Figure 9: Top SelectKBest results sorted by F-score

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 3 k=10	0.564526	0.535425	0.429828	0.891374	0.887043
Gaussian Naive Bayes RUN 2 k=20	0.563041	0.536070	0.433858	0.872306	0.880000
Gaussian Naive Bayes RUN 3 k=20	0.529243	0.516149	0.425288	0.870732	0.857585
Gaussian Naive Bayes RUN 1 k=20	0.559163	0.538115	0.447890	0.846093	0.856187
Gaussian Naive Bayes RUN 2 k=5	0.541733	0.529028	0.449282	0.844249	0.823920

Figure 10: Top SelectKBest results sorted by test score

Again, the best performing classification was the Gaussian Naïve Bayes algorithm, with the top 10 highest scoring features. This model achieved an 88.7% test score, however there was not much improvement in the other scoring criteria.

3) Principal Component Analysis (PCA)

Principal component analysis works differently from the feature selection methodologies. The main working principle is that the features are combined in several ways so that a high dimensional data can be reduced to only a few dimensions. For our dataset, we tested the classification algorithms after reducing the dimensionality of the training and testing data with n_components of values 2, 5, 10, 20, and 30.

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN n_components = 10 scoring = precision	0.754132	0.570226	0.511070	0.821429	0.890909
Decision Tree Classifier RUN n_components = 10 scoring = accuracy	0.748367	0.570263	0.505565	0.641914	0.624011
SVM NON Linear RUN 1 n_components = 2 scoring = accuracy	0.744845	0.511921	0.400316	0.593069	0.604222
Decision Tree Classifier RUN n_components = 20 scoring = precision	0.739759	0.559015	0.497922	0.800885	0.854167
Decision Tree Classifier RUN n_components = 20 scoring = precision	0.708787	0.559572	0.497779	0.827586	0.807018

Figure 11: Top PCA results sorted by precision

Classifier	Precision	Recall	Fscore	Train score	Test score
KNN RUN 1 n_components = 10 scoring = precision	0.676274	0.605351	0.584923	0.635317	0.718519
KNN RUN 2 n_components = 20 scoring = precision	0.647342	0.591271	0.573452	0.625229	0.657143
Decision Tree Classifier RUN n_components = 5 scoring = recall	0.625814	0.590413	0.577298	0.400163	0.331288
KNN RUN 2 n_components = 5 scoring = accuracy	0.619477	0.590368	0.583864	0.681518	0.635884
KNN RUN 3 n_components = 5 scoring = precision	0.644105	0.588955	0.569318	0.635514	0.657143

Figure 12: Top PCA results sorted by recall

Classifier	Precision	Recall	Fscore	Train score	Test score
KNN RUN 2 n_components = 20 scoring = recall	0.589604	0.585795	0.586037	0.693745	0.472050
Decision Tree Classifier RUN n_components = 30 scoring = f1	0.601692	0.587500	0.585496	0.655904	0.463551
KNN RUN 1 n_components = 10 scoring = precision	0.676274	0.605351	0.584923	0.635317	0.718519
KNN RUN 2 n_components = 5 scoring = accuracy	0.619477	0.590368	0.583864	0.681518	0.635884
KNN RUN 1 n_components = 20 scoring = f1	0.586030	0.583167	0.583492	0.733929	0.503268

Figure 13: Top PCA results sorted by F-score

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN n_components = 20 scoring = recall	0.581116	0.542255	0.433297	0.878400	0.904290
Decision Tree Classifier RUN n_components = 10 scoring = precision	0.754132	0.570226	0.511070	0.821429	0.890909
Decision Tree Classifier RUN n_components = 20 scoring = precision	0.739759	0.559015	0.497922	0.800885	0.854167
Gaussian Naive Bayes RUN n_components = 20 scoring = recall	0.525051	0.514950	0.421681	0.861958	0.843648
Gaussian Naive Bayes RUN n_components = 30 scoring = recall	0.533299	0.520247	0.436190	0.848583	0.842767

Figure 14: Top PCA results sorted by test score

With PCA, the best algorithm was again the Gaussian Naïve Bayes algorithm (n_components = 20) if we only considered the test score. The Decision Tree classifier, however, should be considered owing to its better F-score with n_components = 10. Again, there was not a significant improvement in the other scoring criteria, although this can be attributed to the dataset itself. The explained variance was plotted against the number of principal components and compared between n_components = 2 and n_components = 30.

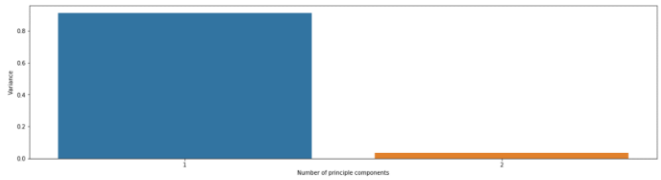

Figure 15: Variance vs n_components for 2 principle values

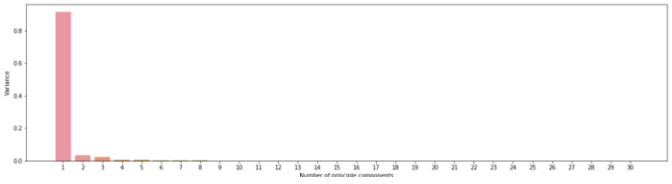

Figure 16: Variance vs n_components for 30 principle components

It is interesting to see that the first principle component consistently had greater than 90% of the variance across all values of n_components we tested.

C. Classification methods and Results

The requirements of the assignment mandated the use of supervised learning-based classification methods. The methods were:

- SVM Linear
- SVM Non-Linear
- K-Nearest Neighbor
- Naïve Bayes
- Decision Tree Classification (DTC)

Grid Search hyperparameter optimization with cross validation was also used.

1) SVM Linear

The working concept of Support Vector Machine learning is that it is a method that searches for the decision boundary (also known as the hyperplane) between records. For SVM Linear, it creates a linear equation in order to do this.

The parameters we tweaked are C and Gamma, these two parameters can help us control the tradeoff between train and test error. The benefit of SVM Linear is that it performs better than other classifiers when there are a lot of features, therefore as seen during the SVM Linear performance degrades in our case after performing feature selection. In terms of implementation we used a pipeline method where our sets are scaled and then the model is called for fitting.

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM Linear C = 0.1	0.614714	0.577637	0.559909	0.645545	0.617414
SVM Linear C = 0.5	0.613765	0.578405	0.561818	0.647855	0.617414
SVM Linear C = 0.7	0.613765	0.578405	0.561818	0.646865	0.617414
SVM Linear C = 0.9	0.613765	0.578405	0.561818	0.647195	0.617414
SVM Linear C = 1	0.613765	0.578405	0.561818	0.647195	0.617414
SVM Linear C = 2	0.615958	0.579559	0.562858	0.646865	0.618734
SVM Linear C = 10	0.610287	0.575328	0.557841	0.644884	0.614776
SVM Linear C = 15	0.610287	0.575328	0.557841	0.645215	0.614776
SVM Linear C = 20	0.610287	0.575328	0.557841	0.645215	0.614776

Figure 17: A run of SVM linear

The model resulted in a best train and test score of 64 and 61, with very average and middling scores for precision, recall, f-score and train score. After close observation of the results, it's clear that the any choice of C and Gamma delivers almost the same score for the scoring criteria. Therefore we believe the default parameters of C and Gamma, which in sklearn is 1.0, is the optimal parameter for this method and dataset.

2) SVM Non-Linear

Non-Linear Support Vector Machines instead apply a kernel trick to have a nonlinear transformation when creating the hyperplane. The two main parameters tweaked for this method are C and the polynomial degree. C helps us control the tradeoff between test and test error, whereas, degree can be used to change the polynomial kernel function.

Classifier	Precision	Recall	Fscore	Train score	Test score
polynomial_svm_clf C = 10, degree = 2	0.787443	0.757268	0.762596	0.777228	0.777045
polynomial_svm_clf C = 15, degree = 2	0.792362	0.761116	0.766648	0.779208	0.781003
polynomial_svm_clf C = 20, degree = 2	0.792663	0.763038	0.768534	0.780198	0.782322

Figure 18: A run of SVM Non-Linear

The results are evident on how Non-Linear Support Vector Machines produce much better results than Linear Machines with our specific dataset. All scoring criteria are higher on average by 15% to 20%.

3) K-Nearest Neighbor

The next model was K-Nearest Neighbors, this classifies based on the majority vote of the nearest neighbor of each point. The main parameter tweaked was K itself, affecting how many neighbors to take votes from. A lower K resulted in a more overfitted dataset resulting in a train and test score of 78 and 58 respectively. However, K at 10 and 50 fared much better results with train and tests of both at 64 and 60.

Classifier	Precision	Recall	Fscore	Train score	Test score
knn 50	0.593584	0.554951	0.525431	0.649175	0.600264
knn 10	0.593584	0.554951	0.525431	0.649175	0.600264
knn 3	0.593584	0.554951	0.525431	0.783168	0.575198

Figure 19: A run of KNN

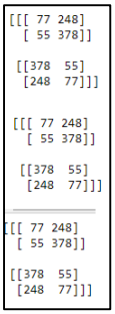

Figure 20: Confusion matrix for KNN

Results produces by KNN were not favorable as even with non-overfitted models scores for precision, f-score and recall are low at around 50%. The confusion matrix seen is a good indicator that this model has poor f-score results.

4) Gaussian Naïve Bayes

Gaussian Naïve Bayes applies Bayes theorem with the naïve assumption of conditional independence between every pair of features. As a result of this, after performing PCA which performs dimensionality reduction. Which removes the correlated features, naïve Bayes performed well since it assumes a conditional independence between every pair of features. Resulting in high test and train scores of 88 and 91. However low f score, precision and recall were also seen.

Classifier	Fscore	Precision	Recall	Test score	Train score
Gaussian Naive Bayes RUN 1 With Scoring method recall	0.582854	0.539236	0.415677	0.888712	0.918644
Gaussian Naive Bayes RUN 1 With Scoring method precision	0.513262	0.506738	0.395869	0.423313	0.406154
Gaussian Naive Bayes RUN 1 With Scoring method accuracy	0.291557	0.500000	0.368333	0.591749	0.583113
Gaussian Naive Bayes RUN 1 With Scoring method f1	0.503617	0.501757	0.384598	0.571726	0.546973
Gaussian Naive Bayes RUN 2 With Scoring method recall	0.542192	0.521457	0.400995	0.885103	0.886598
Gaussian Naive Bayes RUN 2 With Scoring method precision	0.598333	0.549084	0.439478	0.425000	0.432515
Gaussian Naive Bayes RUN 2 With Scoring method accuracy	0.299472	0.500000	0.374587	0.587789	0.598945
Gaussian Naive Bayes RUN 2 With Scoring method f1	0.519466	0.510047	0.405421	0.571577	0.563087
Gaussian Naive Bayes RUN 3 With Scoring method recall	0.557445	0.529272	0.434273	0.884176	0.886850
Gaussian Naive Bayes RUN 3 With Scoring method precision	0.540159	0.521303	0.411407	0.422120	0.407752
Gaussian Naive Bayes RUN 3 With Scoring method accuracy	0.462691	0.463830	0.462739	0.489109	0.489446
Gaussian Naive Bayes RUN 3 With Scoring method f1	0.549094	0.524694	0.409895	0.572909	0.560924

Figure 21: A run of Gaussian NB

The best model from the GNB method had test and train scores of around 89%, however the rest of the scoring criteria showed poor results. With precision and recall being less than 54 percent.

5) Decision Tree classifier

Decision Tree Classifier creates a model that predicts values based on inferred rules seen from the features. The parameters tweaked were the max tree depth, minimum samples split and leaf. For our application, DTC performed the best in terms of high test and train scores, while maintaining higher than average scores for precision, recall, f measure.

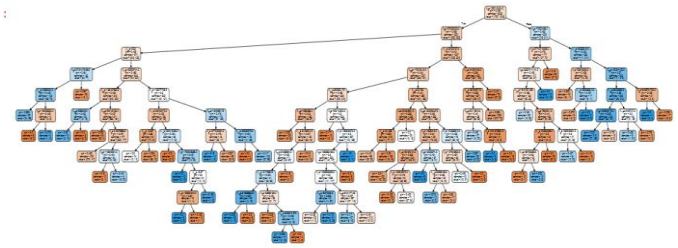

Figure 22: Tree created by the decision tree classifier

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN n_components = 10 scoring = precision	0.754132	0.570226	0.511070	0.821429	0.890909
Decision Tree Classifier RUN n_components = 20 scoring = precision	0.739759	0.559015	0.497922	0.800885	0.854167

Figure 23: Results after using PCA

After comparing results from all 5 models with various preprocessing and optimization techniques, we found that our best model is a Decision Tree Classifier with 20 components with PCA preprocessing. Furthermore, the best model had a max depth of 10 and maximum leaf nodes of 70.

IV. Conclusion

In conclusion, our dataset allowed us to create models for predicting stock price direction using over 200 financial indicators. In order to find the best model, we created models with various different preprocessing techniques and classification methods. The end result showed that the best model is a Decision Tree Classifier with test and train scores of 89%, and precision score of 75. However, reviewing all the results collectively it is clear that our f score, and recall scores are low throughout all the experiments. This may be due to imbalanced dataset as well as how companies in different sectors have stocks that perform differently through different years. For example, stocks are in the financial sectors behave differently than in the utilities. Therefore, creating models that are using various sectors whose stock records are not indicative of one another will create machine learning models with undesirable results.

References

[1] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques," Expert Systems with Applications, vol. 42, no. 1, pp. 259–268, 2015.

[2] M. Moukalled, W. El-Hajj , and M. Jaber, "Automated Stock Price Prediction Using Machine Learning".

[3] F. Dařena, J. Petrovský, J. Žižka, and J. Přichystal, "Machine Learning-Based Analysis of the Association Between Online Texts and Stock Price Movements," Inteligencia Artificial, vol. 21, no. 61, p. 95, Sep. 2018.

[4] M. S. Islam, M. A. Salam, and M. M. Hasan, "Factors Affecting the Stock Price Movement: A Case Study on Dhaka Stock Exchange," International Journal of Business and Management, vol. 10, no. 10, 2015.

[5] B. Qian and K. Rasheed, "Stock market prediction with multiple classifiers," Applied Intelligence, vol. 26, no. 1, pp. 25–33, 2006.

[6] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," Expert Systems with Applications, vol. 42, no. 4, pp. 2162– 2172, 2015.

[7] S. Basak, S. Kar, S. Saha, L. Khaidem, and S. R. Dey, "Predicting the direction of stock market prices using tree-based classifiers," The North American Journal of Economics and Finance, vol. 47, pp. 552–567, 2019

[8] J. Xianya, H. Mo, and L. Haifeng, "Stock Classification Prediction Based on Spark," Procedia Computer Science, vol. 162, pp. 243–250, 2019.M. Moukalled, W. El-Hajj , and M. Jaber, "Automated Stock Price Prediction Using Machine Learning".

[9] G. Albanis and R. Batchelor, "Combining heterogeneous classifiers for stock selection," Intelligent Systems in Accounting, Finance and Management, vol. 15, no. 1-2, pp. 1–21, 2007.

[10] Y. Kara, M. A. Boyacioglu, and Ö. K. Baykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange," Expert Systems with Applications, vol. 38, no. 5, pp. 5311–5319, 2011.

[11] W. Huang, Y. Nakamori, and S.-Y. Wang, "Forecasting stock market movement direction with support vector machine," Computers & Operations Research, vol. 32, no. 10, pp. 2513– 2522, 2005.

[12] Y. Jiao and J. Jakubowicz, "Predicting stock movement direction with machine learning: An extensive study on S&P 500 stocks," 2017 IEEE International Conference on Big Data (Big Data), 2017.

[13] V. H. Shah, "Machine Learning Techniques for Stock Prediction ," Foundations of Machine Learning, vol. 1, no. 1, pp. 6–12, 2007.

[14] G. Albanis and R. Batchelor, "Combining heterogeneous classifiers for stock selection," Intelligent Systems in Accounting, Finance and Management, vol. 15, no. 1-2, pp. 1–21, 2007.

[15] M. Park, M. L. Lee, and J. Lee, "Predicting Stock Market Indices Using Classification Tools," Asian Economic and Financial Review, vol. 9, no. 2, pp. 243–256, 20