S&P 500 Classification Using PySpark

Group 2, Candidate numbers: 47565, 56117, 56290

ABSTRACT

Forecasting the movement of stock prices has always been challenging and has received widespread attention over the years. Being able to make an accurate prediction carries strong implications for trading strategies. In this paper, we use big data analytics to analyze S&P 500 data collected through Yahoo Finance for the past 20 years and build machine learning pipelines to apply machine learning classifiers available in MLlib library over PySpark to predict the directions of various stocks forming the S&P 500 index. We use the Apache PySpark big-data framework provided by the Google Cloud Platform(GCP) to analyze the large dataset which is stored leveraging the Hadoop Distributed File System. In terms of results, we observe that Random Forest is the model which shows the least runtime across the different cluster configurations employed while performing scalable machine learning. Moreover, we observe that the training and testing error remains the same for the random forest model while changing the number of trees across different clusters operating under different worker nodes. Also, with the increased number of worker nodes in different clusters, the computation time decreases across all machine learning models as the processing gets distributed across the worker nodes.

KEYWORDS

Big Data, PySpark, HDFS, Distributed Computing, Clusters, Classification, Scalable Machine Learning, GCP

1 INTRODUCTION

The stock market is one of the most well-known financial markets worldwide where millions of transactions happen every day, in which shares of publicly held companies are bought and sold. The stock market is a very critical factor in the global economy and is very significant in stimulating the flow of capital between investors and businesses. Various factors, such as economic statistics, company performance, interest rates, political and geopolitical events, and market mood, can have an impact on the stock market. These elements may affect how investors behave and cause changes in stock values. Therefore, it is essential to understand how the data behave within the market to invest.

In order to efficiently analyze vast amounts of data, investors must overcome significant difficulties brought on by the exponential rise of technology. Distributed computing is a technique for handling massive volumes of data. In data science, distributed computing refers to a method that combines several computers, or nodes, to tackle complicated data issues. Programming frameworks like MapReduce and storage frameworks such as Hadoop Distributed File System makes distributed computing incredibly simple and easy for users to store large datasets and expand algorithms to analyze them. This method makes it possible to handle huge amounts of data fast and effectively.

When it comes to analyzing massive volumes of stock market data, distributed computing can be a game-changer for investors as it enables the opportunity to extract information from big volumes of data. Nonetheless, by the Law of Large numbers, the more data or information we have the more unbiased the analysis is. Hence, investors may get more valuable insights into market patterns, identify opportunities, and make better investment decisions by utilizing the power of distributed computing. For many algorithms, increasing the input data amount can considerably minimize the learning error and is frequently more efficient than employing more complicated techniques. In this sense, distributed computing has evolved into a vital tool for investors trying to stay competitive and negotiate the complexity of the stock market.

In this paper, the use of the Hadoop Distributed File System (HDFS) offered by GCP is utilized to store the big data and the analysis of it is going to be made by distributing the data into several numbers of virtual machines available over the cluster provided by GCP which is made up of different combinations of master node and worker nodes. After the pre-processing stage, which includes loading the data from Hadoop storage into PySpark dataframes, applying map functions, and feature engineering, we are going to make use of the MLlib library available in PySpark to build machine learning pipelines for applying classification models to forecast the direction of various components of S&P500. Multiple classification methods like Supporting Vector Machines, Logistic Regression, Random Forest, and Gradient Boosted Trees Classifier are going to be tested and compared for their prediction accuracy using the AUC-ROC score and the AUC-PR score while their computation time is measured to account for the effect of distributed computing on the large scale dataset. We also observe the effect of change in computation time by employing different cluster settings which gives access to a different number of worker nodes for processing the data in an effective parallelized manner.

2 RELATED WORK

[11] predicted the movement of stock price by leveraging the Cloudera-Hadoop-based data pipeline to perform analysis of the US stocks based on real-time data from Yahoo Finance. The advantage of using this method is the ability to process on multiple machines (distributed architecture) parallelly (parallel computing) in the Hadoop ecosystem. In this paper, the direction was predicted by working with the MLlib library available in Spark (a leading tool in the Hadoop Ecosystem) as it can work stand-alone or over the Hadoop framework to leverage the big-data and perform real-time data analytics in a distributed computing environment which is 100 times faster than Hadoop MapReduce Framework for large scale data processing as it performs in-memory computations thus providing increased speed over MapReduce.

The study by [5] concerns stock classification using Spark. An information-rich dataset from the past 20 years that included data on 7001 US stocks was used for supervised learning and it was used to test the efficiency and precision of four classification methods in various-sized clusters. They discovered that the runtimes of the four algorithms were essentially the same. They discovered a negative correlation between the runtime and cluster size. Therefore, as the number of clusters increases, the runtime of a model decreases. The models were tested using four unique clusters, each having 11 nodes, 8 nodes, 6 nodes, and 3 nodes respectively. A significantly reduced runtime was obtained by adding more nodes. For example, the cluster with 11 nodes outperformed the cluster with 3 nodes in terms of runtime, with an execution time that was about five times less. The default task of the scheduling mode of Spark is that the task scheduler randomly selects some node from the registered Worker Nodes, selects the highest localized task execution node, and assigns the task to it, which does not consider the current resource surplus of the cluster nodes. Therefore, it may cause every machine's state to become different. In conclusion, they found that the Random Forest and Decision tree algorithms are more suitable for predicting Stock Classification with corresponding accuracies found to be 0.829 and 0.832 respectively.

The idea of distributed machine learning is covered in the research of [1], which entails leveraging multinode machine learning algorithms and systems to boost efficiency, boost precision, and scale to bigger input data sets. According to the article, distributed machine learning systems can be divided into three main groups which are 'database', 'general', and 'purpose-built systems'. Users had to write handwritten answers to run machine learning algorithms before distributed frameworks were available. However, brand-new programming frameworks like MapReduce have made distributed computing much simpler and enabled users to scale algorithms to bigger datasets. The article describes how to conceptually transform single-threaded algorithms into parallel algorithms and construct parallel algorithms to run machine learning algorithms in a distributed environment. The article offers a summary and information about gradient parallelization, regression, K-Means Clustering, and Ensemble Learning.

[2] leveraged the fundamental and technical analysis aspect for predicting the direction of stock price movement. Fundamental analysis is performed by doing sentiment analysis over the Hadoop framework using Hive script which produces sentiment of news and tweets. The sentiment score is incorporated with other quantitative features such as closing price, opening price, and highest price, among others to build a logistic regression model which achieves an accuracy of 70% for predicting the direction of stock price movement.

[7] looks at a comparison between the accuracy metric of different classification models such as Support Vector Machines, Naive Bayes, Random Forest, and Logistic Regression by integrating different information while predicting the stock trend of the Dabur Company and Maruti Suzuki Company. For predicting the trend of stocks the following information is considered: historical data, news sentiments, tweets sentiments, news and tweets sentiments, and product review statements which are transformed into quantitative values by performing sentiment analysis. Moreover, this paper also accounts for different computation times observed between different cluster settings which shows that spark performs the computation processes eight times faster than a Map-Reduce process for doing sentiment analysis of textual data and ultimately concludes by presenting that prediction accuracy is improved by incorporating textual data in the dataset with random forest giving the best result out of all the classifiers with an accuracy score of 87.02%.

[6] found that there has been a lot of work in recent years for predicting the direction of stock movement for devising financial trading strategies. This was further supported by [3] who found that the emergence of AI has led to better modeling and has outperformed traditional statistical methods when problems such as credit evaluation, portfolio management, and financial prediction are considered

[4] research was about the Day-of-the-Week impact as one of the calendar's influences on stock market returns using the index S&P500. An important note of the research was the average return of each day during a 25 years period which showed that all weekdays had positive returns except Monday (Day-of-the-Week effect). They achieved that by examining clustering, machine learning, and AI algorithms that are able to find characteristic features that can be used to effectively predict the Day-of-the-Week effect. The main algorithms that were applied in the research were LDA, SVM, and ANN algorithms with 64.43%, 51.64%, and 54.07% accuracy measures respectively.

There are several techniques for predicting the direction of stock price movement. [9] tried predicting the direction of price movement by using 53 technical indicators as features for Support Vector Machine (SVM) based approach and got an accuracy of 55% and 65%.

The distributed machine learning framework PySpark's MLlib package, which enables scalable training and prediction on huge data, is described in [14]. The package covers a number of machine learning models, related to classification, regression, clustering, and collaborative filtering. It demonstrates how MLlib outperforms battling distributed machine learning frameworks like Mahout in terms of speed and how it can be used for building scalable machine learning pipelines.

[12] predicted the direction by using an ensemble technique of majority voting over three models: Decision Trees (DT), K-Nearest Neighbours (KNN), and Neural Networks (NN) to obtain a result that achieved better accuracy than any single classifier. Attempts have been made to predict the direction of price movement by considering just the price data.

[13] predicted the Korean and Hong Kong markets by performing PCA on the features based on the price to reduce the number of input features before applying SVM. On similar lines, [8] predicted the stock direction of three stocks: Apple, Samsung, and GE.

3 METHODOLOGY

3.1 Cluster Configuration:

For our analysis, we set up a cluster with **'Europe-West 2' as the region** and **'Europe-West 2-c' as the zone** on Google Compute Engine (GCE). The cluster gives us access to a group of virtual machines that can be managed as a single unit having an architecture as described in Figure 1.

(1) **Master Nodes**: These refer to the component of the cluster which manages and controls the worker nodes in the cluster as discussed by [7]. The main tasks that are overlooked by the cluster include scheduling various tasks for worker nodes, scaling the cluster by monitoring the load on the worker nodes, and managing the allocation & distribution of resources such as CPU and memory among worker nodes.

    We use an 'n1-standard-4' type virtual machine with 500GB as the primary disk size, 4 virtual CPUs, and 15 GB memory as our master node.

(2) **Worker Nodes**: These are the components of the cluster which run application workloads and execute tasks assigned to it by the master node. They are considered to be essential as they provide the computational power necessary to perform tasks assigned by master nodes.

    We use an 'n1-standard-2' type virtual machine with 500 GB as the primary disk size, 2 virtual CPUs, and 8 GB memory as our worker nodes.

n1 machine refers to the first generation platform of these machines.

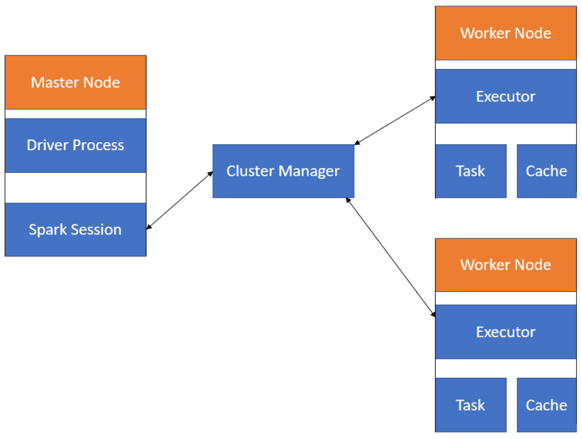

Figure 1: Cluster Management

3.2 Data Acquisition and Characterisation:

We collect the dataset for S&P 500 since its inception through the Yahoo Finance API. Our dataset has the financial history of all the stock components of the S&P 500 for the last 20 years. We have about 2.5 million rows in our dataset and it is characterized by seven different features: 'Open', 'High', 'Low', 'Close', 'Adj Close' represent the prices associated with the stock, and 'Volume' represent the number of shares that were traded corresponding to the stock on a particular day. All seven of these features are float types

3.3 Data Storage:

We obtain our dataset for S&P 500 and create a directory named 'data' in our master node which leverages the **Hadoop Distributed File System (HDFS)** architecture functionality for storing the data such that it can be divided among multiple worker nodes for parallel computation in the worker nodes of our cluster when the process of analyzing the dataset starts.

We choose HDFS as discussed by [10] for storage purposes as it is highly scalable and fault-tolerant, enabling it to run on a cluster including several hundred or thousand nodes in which hardware failure is normal. It supports master-worker architecture, with a master known as Name-Node for file-system namespace management and regulation of file accessibility by users along with some workers called DataNodes each located at a machine for data storage. The data stored over HDFS are partitioned into several blocks with fixed sizes (e.g., 64 MB per data block) such that the NameNode dispatches these data blocks to different DataNodes which save and manage the data assigned to them. It achieves the improvement in data reliability by replicating each data block three times (the replicator is 3 by default) and saving the replicas in different racks as shown in Figure 2.

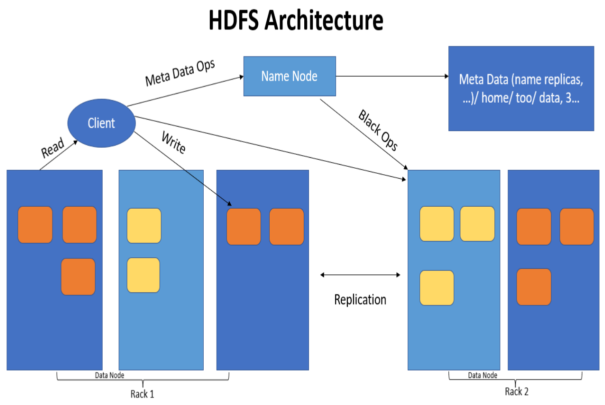

Figure 2: HDFS Architecture

3.4 Data Preprocessing:

For improving the scope of our analysis, we perform feature engineering over pre-existing features to obtain artificial features.

We perform this task over **PySpark Dataframes** as it performs execution on all the cores of the available worker nodes and hence achieves faster processing operations than Pandas Dataframes.

We introduce the following features in our dataset:

(1) **Returns**: It is the difference of log adjusted closing price ('Adj Close') over consecutive days.

(2) **Surge**: It is calculated by taking the log ratio of volume ('Volume') and the average volume calculated over the past 10 days.

(3) **Momentum**: It is obtained by taking the log ratio of the adjusted closing price and the average adjusted closing price calculated over 10 days.

(4) **Spread**: It is obtained by taking the log ratio of the highest price('High') and the lowest price('Low') for each day.

(5) **Volatility**: It is obtained by taking the standard deviation of the Returns over the past 10 days.

3.5 Scalable Machine Learning

We apply the classification models to our dataset by applying the MLlib library and Pipeline API offered by PySpark for building scalable machine learning pipelines proposed in Figure 3.

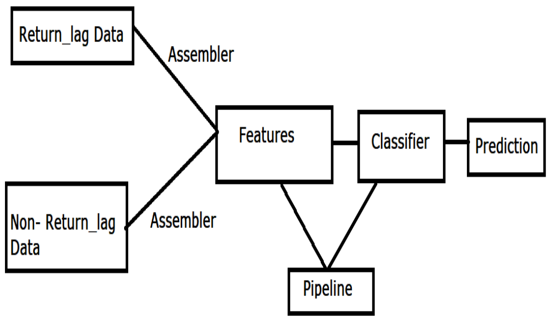

Figure 3: Scalable Machine Learning Pipeline

The key advantage of these libraries as reported by [14] is that it allows for distributed machine learning as it divides the processing task among the multiple worker nodes employed under the master node of the cluster to achieve parallel computing and hence achieve faster processing times. Due to its support for distributed computing, it is considered an ideal choice for deploying the following machine learning models in our analysis: Logistic Regression, Random Forest, Gradient Boosted Trees, and Support Vector Machine.

4 NUMERICAL RESULTS

4.1 Goals

This project's main goal is to use distributed computation techniques to effectively forecast the movements of numerous equities utilizing a wealth of data obtained by Yahoo Finance. We may expand the amount of data used and, as a result, improve the accuracy of our machine-learning models by using distributed computation. Analysis bias will significantly decrease as a result of this. Furthermore, distributed computation will make it possible to process the analysis and modeling runtimes more quickly.

We will use a distributed computing architecture capable of handling and processing huge datasets to accomplish this goal. As a result, we will be able to use parallel processing to accelerate and optimize our computations. Furthermore, to analyze the data and produce reliable results, we will make use of sophisticated machine-learning techniques and algorithms.

4.2 Dataset

The dataset comes from Yahoo Finance which includes the price information about 503 stocks that are included in the index S&P500 for the past 20 years. The data set used for the distributed machine learning part consists of the 5 columns mentioned in the feature engineering process which are derived from the existing 7 columns present at the original data set, and approximately 2.5 million rows, no null values, no duplicated rows.

4.3 Evaluation Metrics

4.3.1 Evaluation across machine learning models

As we are primarily dealing with a classification problem. So, our main evaluation metric used for comparing the performance of various classifiers will be the accuracy metric. We define the accuracy of any classifier as follows,

$$\text{Accuracy} = 0.5\frac{TP}{TP + FN} + 0.5\frac{TN}{FP + TN}$$

Here TP, TN, FP, and FN refer to the respective numbers of true positives, true negatives, false positives, and false negatives. We chose this definition of accuracy because it takes the average of accuracy on data with ground truth labels of positive and negative, so as to account for a potentially imbalanced sample.

Receiver Operating Characteristic (ROC): We'll use ROC, which is a graph that shows how well a binary classification model performs. Plotting the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds is a frequent method in machine learning and statistics to assess the performance of a classifier model.

Each point on the ROC curve corresponds to a distinct threshold value for the classifier, and it is a plot of TPR on the y-axis versus FPR on the x-axis. Then the **AUC (Area Under ROC)** will be used as another accuracy measure. Regardless of the precise classification threshold, the AUC measures the model's general ability to discern between positive and negative cases.

Another measure used for accuracy in **Precision - Recall (PR)**. **Precision** measures how accurate a model is in identifying positive instances. In other words:

$$\text{Precision} = \frac{TP}{TP+FP}$$

while **Recall** measures how complete a model is in identifying all positive instances. Its Mathematical expression is

$$\text{Recall} = \frac{TP}{TP+FN}$$

They have an inverse relationship and finding the right balance is important for an effective model, meaning that as precision increases, recall tends to decrease, and vice versa. Thus, finding the right balance between precision and recall is important in developing an effective model.

The precision-recall curve, which displays recall on the x-axis and precision on the y-axis, is represented by the AUC-PR statistic. The curve illustrates how the categorization threshold affects the model's recall and precision. An increase in the AUC-PR score, which runs from 0 to 1, indicates improved performance. The closer to 1 the better the model

4.3.2 Evaluation across distributed computing methods

In order to evaluate the computation performance across different cluster settings, we use **'time'** as a metric to see the comparison in performance across them. We measure the computation time across the implementation of the entire machine learning pipeline over our dataset by making use of the **time library** available in Python to take into account the effect of the different numbers of worker nodes and resources available to the master node at our disposal.

4.4 Results

Our study focused on evaluating the performance of several machine-learning models using different cluster configurations. Specifically, we tested our models using three different cluster settings. The primary cluster configuration involved utilizing an 'n1-standard-4' virtual machine with 4 virtual CPUs, 15 GB memory, and a 500GB primary disk size as our master node. Then, for our final computations, we altered the number of worker nodes of two, four, and six.

The Random Forest model had the quickest runtime, taking only 77 seconds, compared to the Logistic Regression, GBT, and SVM models' runtimes of 90.1, 767.6, and 1295.7 seconds, respectively while using just two worker nodes. Below is a table with our findings that contains the accuracy, runtime, AUR, and AUC-PR .

Model	Accuracy	Time	AUC	AUC-PR
RF	0.505	77	0.514	0.508
LR	0.504	90.1	0.518	0.52
GBT	0.507	767.6	0.519	0.511
SVM	0.5	1295.7	0.501	0.499

Following the initial cluster configuration, we further evaluated the performance of our machine-learning models using four worker nodes and one master node. Our results indicate that utilizing four worker nodes resulted in reduced runtime across all four models, with Random Forest demonstrating the shortest runtime of 58.8 seconds. In comparison, Logistic Regression, GBT, and SVM models took 67.9, 504.2, and 949.5 seconds, respectively. Despite the reduction in runtime, the accuracy of all four models remained relatively consistent.

Model	Accuracy	Time	AUC	AUC-PR
RF	0.505	58.8	0.514	0.508
LR	0.504	67.9	0.518	0.52
GBT	0.506	504.2	0.518	0.511
SVM	0.5	949.5	0.516	0.514

By increasing the number of worker nodes to six in the distributed computing environment, the runtime of the machine learning models was reduced even further. Specifically, the runtime of the four models became, 49.1 50.8, 382.7, and 679.7 for Random Forest, Logistic Regression, GBT, and SVM respectively.

Model	Accuracy	Time	AUC	AUC-PR
RF	0.501	49.1	0.514	0.508
LR	0.504	50.8	0.518	0.52
GBT	0.507	382.7	0.52	0.512
SVM	0.5	679.7	0.503	0.5

Our findings suggest that using several worker nodes can significantly increase a cluster's capacity for parallel processing. It is clear that there is a negative correlation between the number of worker nodes and the model runtime. Hence, we conclude that the model's runtime decreases as the number of worker nodes increases. Machine learning models run more quickly because the computing resources of the cluster get used more effectively by spreading the task over several nodes.

In our case, Random Forests outperformed all the other models and consistently had the lowest runtime across all the cluster configurations tested, while maintaining similar levels of accuracy compared to other machine learning models mentioned (Logistic Regression, GBT, and SVM). The table below shows the runtime of each model under different cluster configurations.

Worker Nodes	RF	LR	GBT	SVM
1 MN, 2 WN	76.0	90.1	767.6	1295.7
1 MN, 4 WN	58.8	67.9	504.2	949.5
1 MN, 6 WN	49.1	50.8	382.7	679.7

The optimal model might or might not be selected, depending on the specific use case and the 'trade-offs' between accuracy, runtime, and other performance indicators. It is important to note that 'trade - off' in our case does not mean that there is a negative correlation between runtime and accuracy. In some circumstances, a model with a faster runtime but lower accuracy could be preferred to a model with higher accuracy. It will depend on the unique requirements of the user and the specific application of the model. Since runtime is a crucial factor in our study, the Random Forest model is the best choice as it has a very similar accuracy compared to other models, and the runtime is significantly less than the other models.

It is also crucial to keep in mind that depending on the particular specifications of the model and dataset being used, the effect of various cluster configurations on the performance of machine learning models might change. However, this may not always be true for all datasets and machine learning algorithms. In our example, we discovered that having six worker nodes resulted in considerable runtime savings across all four models. The first graph below shows the runtime of all models under the different cluster configurations, while the second graph shows the runtime explicitly for the Random Forest and Logistic Regression

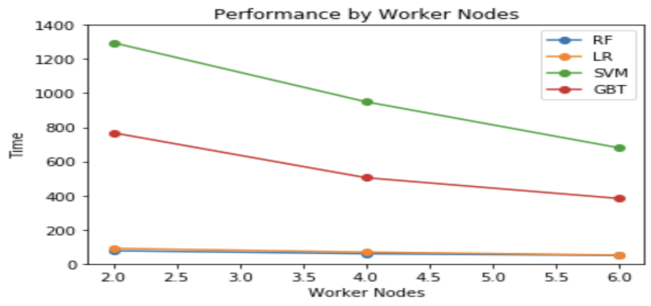

Figure 4: Runtime - Worker Nodes (All four models)

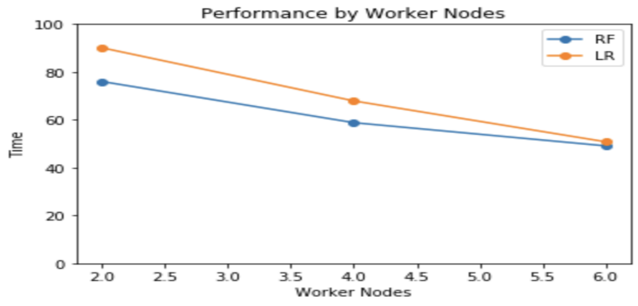

Figure 5: Runtime - Worker Nodes (LR & RF)

4.5 Comparison Between Models

Since our objective is to use the Distributed Computation technique, we strongly emphasize the interpretation of each model's runtime and its associated correctness. As mentioned above, the Random Forests model has a lower runtime than the other models, followed by Logistic Regression which showed very similar results, then GBT, and SVM, respectively. The Random Forest model outperformed all four models under all different configurations.

Despite the differences in runtime, the four models exhibited similar accuracy. GBT showed slightly higher accuracy and AUC-ROC values across all different cluster configurations, while Logistic Regression had a slightly higher AUC-PR value under all different cluster configurations. However, the accuracy measures were almost the same for the two models, followed by Random Forest which also showed extremely similar accuracy scores.

4.6 Experimentation on Optimisation of Algorithm over different worker nodes

We conducted testing errors and runtime measurements for various numbers of trees in our Random Forest Model after receiving the study's results. We gave the time results extra attention because distributed computation is the primary subject of our study. According to our research, there is a direct correlation between the number of trees and runtime, meaning that as the number of trees grows, so does the runtime. Moreover, as mentioned above, we can see once again that the runtime of the model decreases while distributing the dataset into more worker nodes. Furthermore, we found that five trees are the ideal quantity because it produced a testing error that was noticeably low. The graphs below provide a summary of these findings under the 3 different cluster configurations.

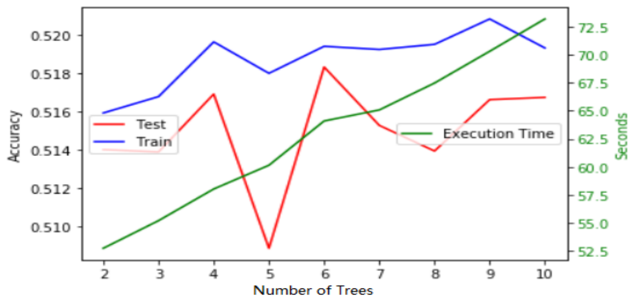

Figure 6: Random Forest- 2 Worker Nodes

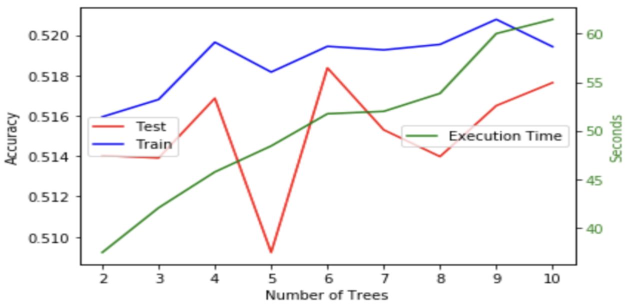

Figure 7: Random Forest- 4 Worker Nodes

It is clear from the examination of the three graphs that regardless of the cluster arrangement, the test and training error of the

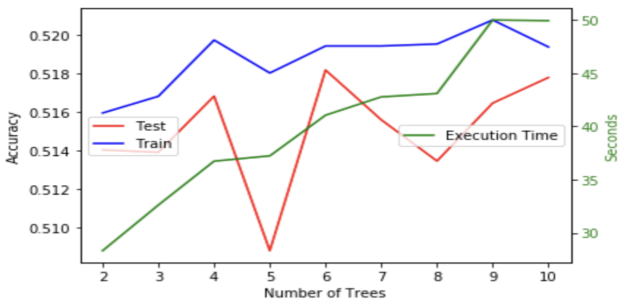

Figure 8: Random Forest- 6 Worker Nodes

model is constant while exploring the number of trees in the model. However, when the number of worker nodes rises, the model's runtime lowers. Given that it enables investors to use a big quantity of data to generate more precise forecasts while drastically lowering the time needed to run the model, this study shows that distributed computation has the potential to revolutionise the area of finance.

Investors may save money as a result of this considerable runtime decrease since they can now complete more calculations in less time. Furthermore, using huge datasets for machine learning models has the potential to improve prediction accuracy, which can

5 CONCLUSION

We observe that the cluster settings play a vital role in deciding the runtimes of the various models under the regime of scalable machine learning and distributed computing. In our paper, the runtimes improved whenever we increased the number of worker nodes in our cluster setting. In terms of accuracy across machine learning models deployed for predicting the direction of various components of S&P 500, we observe that the gradient-boosted model performs well while the support vector machine model performs relatively poorly. We observe that the runtimes of our models are reduced significantly (approximately 2 times faster) when we shift from two worker nodes to six worker nodes. The runtime of the gradient-boosted model shows the maximum decrease in runtime by 51% while for the random forest, it decreases by 36%. Thus we can conclude that the distribution of tasks among worker nodes achieves the process of parallel computing in an efficient and resourceful manner such that big data can be processed to be analyzed strategically.

5.1 Summary

The study looked at how well various machine learning models performed when used in distributed computation to forecast the direction of different S&P 500 components. The findings show that adding more worker nodes to a cluster considerably boosts its capability for parallel processing, resulting in a shorter runtime for models.

By distributing the computational workload among several worker nodes in a cluster, the computing resources are used more efficiently, leading to a reduction in the runtime of machine learning models.

In our study, we found that increasing the number of worker nodes from two to six resulted in significant reductions in the runtime of the models tested. This finding suggests that using distributed computation techniques can be an effective way to achieve faster processing times for large-scale machine-learning tasks.

The Random Forests model consistently outperformed the other machine learning models when evaluated, having the shortest runtime across all cluster configurations while retaining a similar degree of accuracy. The accuracy scores for Logistic Regression and GBT were similarly quite strong, however, SVM performed only somewhat well.

Our study made clear how crucial it is to take into account the trade-offs between runtime, accuracy, and other performance measures when choosing the best model. The optimal model to choose will depend on the particular use case and dataset.

5.2 Limitations

The main limitation is that the configuration of the cluster structure employed in the study was rather straightforward. We did not take into account more complicated cluster setups as those with many layers of worker nodes or diverse computing resources.

Hyperparameter tweaking was not applied to the machine learning models in the study, which might have enhanced their performance and perhaps altered the outcomes.

The study's limited scope prevented it from being generalized to other datasets or machine learning tasks because it concentrated on a particular **financial dataset** and task.

Since financial data is often highly dimensional, has intricate connections, and exhibits nonlinear patterns, it can be difficult to create precise prediction models. Financial data can also be delicate and be the subject of data quality problems, such as missing or insufficient data, which can impair the models' accuracy. The performance of the models can vary based on the unique properties of the data and the problem being addressed, hence the results of this study may not be generalized to areas outside of finance.

5.3 Further Scope

Our paper primarily focuses on understanding the effect of different cluster configurations over the runtimes of various machine-learning models. There is scope for further exploration for understanding the effect of the following in the domain of scalable machine learning:

(1) Parallelism of Data- It involves breaking down larger datasets into smaller subsets such that they can be processed parallelly among different worker nodes. It can lead to more efficient training of machine learning models with a key focus on scalability.

(2) Resource Allocation- It refers to the process of allocating computing resources in a distributed environment such

that the training of machine learning models can be optimized. This can help in understanding further the effect of resource allocation over the performance of machine learning models in terms of runtimes.

(3) Optimization Techniques- This involves dealing with improving the accuracy, efficiency, and performance of machine learning models based on the exploration of parameters that can be deployed when training and testing the model.

REFERENCES

[1] Andrew Crotty Alex Galakatos and Tim Kraska. 2017. Distributed Machine Learning. *Database Group, Brown University, Providence, RI, USA* (2017), 1–6.

[2] Girija V Attigeri, Manohara Pai MM, Radhika M Pai, and Aparna Nayak. 2015. Stock market prediction: A big data approach. In *TENCON 2015-2015 IEEE Region 10 Conference*. IEEE, 1–5.

[3] Arash Bahrammirzaee. 2010. A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications* 19, 8 (2010), 1165–1195.

[4] Szu-Hao Huang Chiao-Chun Cheng Yu-Fu Chen An-Pin Chen Hui-Hsuan Tung, Yu-Ying Chen. 2016. Binary Classification and Data Analysis for Modelling Calendar Anomalies in Financial Markets. *2016 7th International Conference on Cloud Computing and Big Data* (2016).

[5] Hai Mo Jiang X and Li H. 2019. Stock Classification Prediction Based on Spark. In *Information Technology and Quantitative Management*. IEEE, 1–8.

[6] Yang Jiao and Jeremie Jakubowicz. 2017. Predicting stock movement direction with machine learning: An extensive study on S&P 500 stocks. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 4705–4713.

[7] Sneh Kalra, Sachin Gupta, and Jay Shankar Prasad. 2019. Performance evaluation of machine learning classifiers for stock market prediction in big data environment. *Journal of Mechanics of Continua and Mathematical Sciences* 14, 5 (2019).

[8] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. 2016. Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003* (2016).

[9] Yuling Lin, Haixiang Guo, and Jinglu Hu. 2013. An SVM-based approach for stock market trend prediction. In *The 2013 international joint conference on neural networks (IJCNN)*. IEEE, 1–7.

[10] Karwan Jameel Merceedi and Nareen Abdulla Sabry. 2021. A Comprehensive Survey for Hadoop Distributed File System. *Asian Journal of Research in Computer Science* (2021).

[11] Zhihao Peng. 2019. Stocks analysis and prediction using big data analytics. In *2019 international conference on intelligent transportation, big data & smart City (ICITBS)*. IEEE, 309–312.

[12] Bo Qian and Khaled Rasheed. 2007. Stock market prediction with multiple classifiers. *Applied Intelligence* 26 (2007), 25–33.

[13] Yanshan Wang. 2014. Stock price direction prediction by directly using prices data: an empirical study on the KOSPI and HSI. *International Journal of Business Intelligence and Data Mining* 9, 2 (2014), 145–160.

[14] Burak Yavuz Evan Sparks Shivaram Venkataraman-Davies Liu Jeremy Freeman DB Tsai-Manish Amde Sean Owen Doris Xin Reynold Xin Michael J. Franklin Reza Zadeh Matei Zaharia Ameet Talwalkar Xiangrui Meng, Joseph Bradley. 2016. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research* (2016).

Individual Contribution

Members	Report	Research	Data Manipulation	Data Analysis
47567	33%	33%	33%	33%
56117	33%	33%	33%	33%
56290	33%	33%	33%	33%