Customer Churn Prediction in Telecommunications Industry

Big Data Parallel Programming (DT8034): Project Report

Analysis of Programs (1 and 2)

By Mohammad Hammoud

Contents

Contents 1
1.Introduction 2
2.Data Preprocessing 2
2.2Data Cleaning and Transformation 2
2.3Exploratory Data Analysis 3
2.4Feature Selection 5
2.5Data Balancing 6
3.Spark Implementation 6
3.2DecisionTreeClassifier 6
3.3RandomForestClassifier 6
3.4GBTClassifier 6
3.5LogisticRegression 7
4.Results 7
4.2Performance Metrics Across Varying Data Partitions with Workers 7
1.DecisionTreeClassifier 7
2.RandomForestClassifier 9
3.GBTClassifier 11
4.LogisticRegression 13
4.3Performance Metrics Across Varying Workers 14
5.Discussion 17
5.2General Discussion 17
5.3Discussion on "Performance Metrics Across Varying Data Partitions with Workers" 20
5.4Discussion on "Performance Metrics Across Varying Workers" 22
6.Conclusion 22

Table of Tables

Table 1: shows the distribution of churned (1.0) and non-churned (0.0) customers in balanced_df. 7
Table 2: shows the results of DecisionTreeClassifier for each partition using two workers. 9
Table 3: shows the results of DecisionTreeClassifier for each partition using four workers. 9
Table 4: shows the results of DecisionTreeClassifier for each partition using seven workers. 10
Table 5: shows the results of RandomForestClassifier for each partition using two workers. 10
Table 6: shows the results of RandomForestClassifier for each partition using four workers. 11
Table 7: shows the results of RandomForestClassifier for each partition using seven workers. 11
Table 8: shows the results of GBTClassifier for each partition using two workers. 12
Table 9: shows the results of GBTClassifier for each partition using four workers. 13
Table 10: shows the results of GBTClassifier for each partition using seven workers. 13
Table 11: shows the results of LogisticRegression for each partition using two workers. 14
Table 12: shows the results of LogisticRegression for each partition using four workers. 14
Table 13: shows the results of LogisticRegression for each partition using seven workers. 15
Table 14: shows the results of all models with different numbers of workers. 16

Table of Figures

Figure 1: Compare mean values of features for Churned and Not Churned groups. 4
Figure 2: Percentage of Churn against Not Churn. 4
Figure 3: Countplots to visualize relationship between churn and 'Contract_indexed', 'PaperlessBilling_indexed', and 'PaymentMethod_indexed'. 5
Figure 4: Percentages for each category. 5
Figure 5: A set of pie charts to visualize the distribution of the churned customers for each service. 5
Figure 6: Heatmap to visualize correlations between features. 6
Figure 7: A bar plot of the correlations between the features and 'Churn_indexed'. 6
Figure 8: Decision Tree results showed as plots. 9
Figure 9: Decision Tree results showed as plots. 10
Figure 10: Decision Tree results showed as plots. 10
Figure 11: Random Forest results showed as plots. 11
Figure 12: Random Forest results showed as plots. 11
Figure 13: Random Forest results showed as plots. 12
Figure 14: GBTClassifier results showed as plots. 12
Figure 15: GBTClassifier results showed as plots. 13
Figure 16: GBTClassifier results showed as plots. 13
Figure 17: LogisticRegression results showed as plots. 14
Figure 18: LogisticRegression results showed as plots. 15
Figure 19: LogisticRegression results showed as plots. 15
Figure 20: Comparative Visualization of Training Durations Across All Algorithms 16
Figure 21: Comparative Visualization of Evaluation Durations Across All Algorithms. 17
Figure 22: Comparative Visualization of AUC Values Across All Algorithms. 17
Figure 23: Time taken to train two different machine learning algorithms using Spark cluster with increasing number of workers on the x-axis. 18
Figure 24: Time taken to train a machine learning algorithm while increasing the number of worker nodes in the cluster as shown in the x-axis, while the number of RDD partitions is fixed and equals 2. 19
Figure 25: Time taken to train a machine learning algorithm using a cluster of 4 worker nodes and changing the number of partitions in RDD containing training data. 20
Figure 26: Comparative Visualization of All Models with partitions using two workers. 22
Figure 27: Comparative Visualization of All Models with partitions using four workers. 22
Figure 28: Comparative Visualization of All Models with partitions using seven workers. 22

1. Introduction

In today's highly competitive telecommunications market, retaining customers and minimizing churn is critical for businesses to maintain profitability and grow their customer base. Customer churn refers to the loss of customers to competing service providers. This project aims to develop a machine learning model that accurately predicts customer churn in the telecommunications industry using Apache Spark's MLlib library. By identifying customers at risk of churning, companies can implement targeted retention strategies to enhance customer loyalty, mitigate revenue loss, and improve overall business performance.

We will utilize the Telco Customer Churn dataset, available on Kaggle (https://www.kaggle.com/blastchar/telco-customer-churn), containing approximately 7,000 data points with comprehensive information on customer attributes, churn status, and service usage. The dataset will be preprocessed to extract relevant features for training various machine learning models, including DecisionTreeClassifier, RandomForestClassifier, GBTClassifier, and LogisticRegression.

The primary goal is to predict the likelihood of a customer churning, which will be approached as a binary classification problem. The target variable is the customer churn status, represented as Yes or No. The performance of the selected machine learning algorithms will be compared based on their accuracy, training time, and other relevant metrics to identify the best-performing model for predicting customer churn.

By creating and deploying a robust machine learning model with high predictive accuracy, we aim to support telecommunications companies in their efforts to retain customers, enhance customer loyalty, and ultimately improve overall business performance.

2. Data Preprocessing

The initial phase of any machine learning project involves preprocessing the data to prepare it for training with the chosen machine learning algorithms. This section outlines the steps taken to preprocess the Telecom customer churn dataset.

2.2 Data Cleaning and Transformation

We began by loading the Telecom customer churn dataset into a PySpark DataFrame, displaying the first row, number of rows, and column names to understand the structure of the data. We then examined the schema of the DataFrame to identify the data types for each column. Numeric columns were selected and cast to appropriate data types, and we inspected the summary statistics DataFrame in a transposed format to identify any unusual values.

For instances where TotalCharges could not be cast to double, we created a new column with null values, filled null values with the previous value using a window function, and discarded the original and intermediate columns. Next, we applied the StringIndexer to string columns and dropped the original string columns to prepare the data for machine learning algorithms. We then displayed the resulting data frame to verify any data changes.

2.3 Exploratory Data Analysis

We further examined the summary statistics using the describe() function to understand the dataset and converted the PySpark DataFrame to a Pandas DataFrame for easier manipulation. We created separate DataFrames for churned and non-churned customers, plotting heatmaps to compare the mean values of features for both groups as shown in Figure 1.

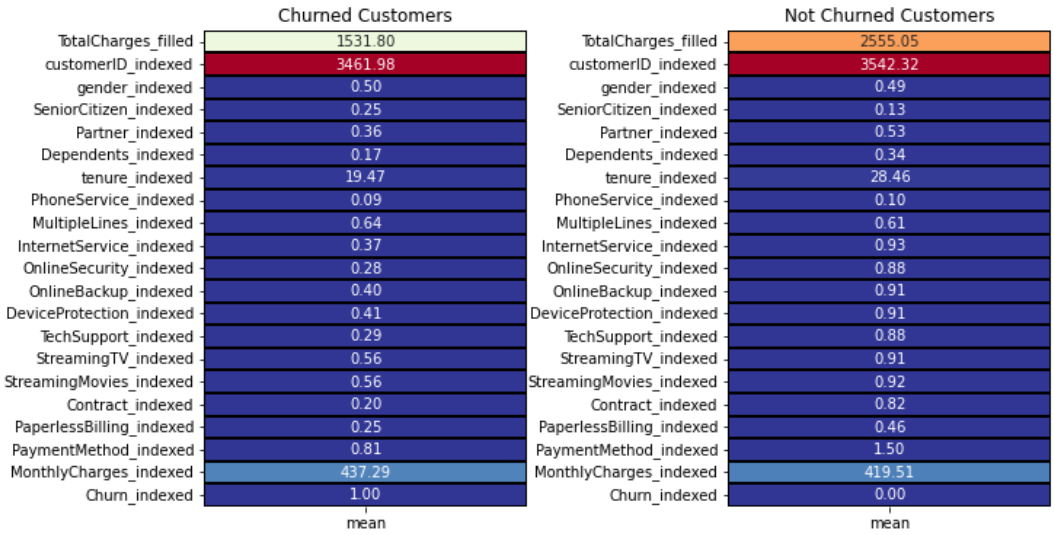

Figure 1: Compare mean values of features for Churned and Not Churned groups.

We reviewed the schema to determine the data types for each column, classifying them into numerical and categorical features. We calculated the percentages of churned and non-churned customers, creating pie charts and bar charts to visualize these figures. The categorical features were divided into customer information, Services Signed Up for, and Payment Information. We used bar charts to visualize the relationship between each feature within these groups and churn.

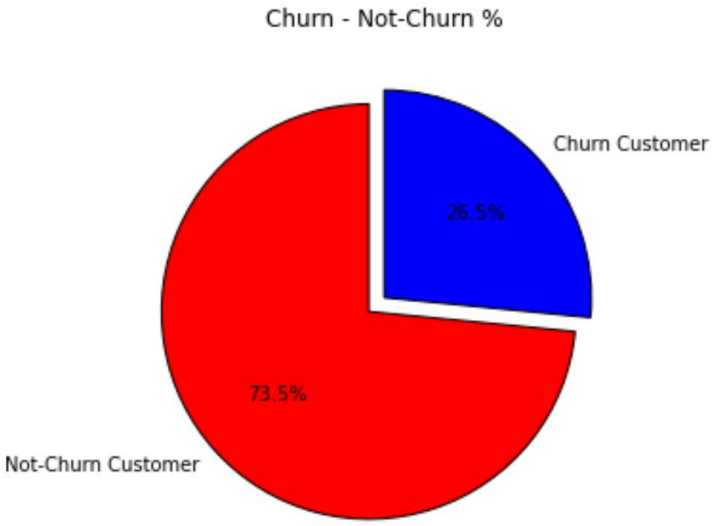

Figure 2: Percentage of Churn against Not Churn.

Further dataset exploration involved creating a list of columns to plot, designing subplots for each column, and plotting countplots using Seaborn. We converted the PySpark DataFrame to

a Pandas DataFrame for additional visualization and plotted countplots for 'Contract_indexed', 'PaperlessBilling_indexed', and 'PaymentMethod_indexed' as shown in Figure 3. We also calculated the percentages for each category in 'gender', 'SeniorCitizen', 'Partner', and 'Dependents' as shown in Figure 4, plotting pie charts for each in Figure 5.

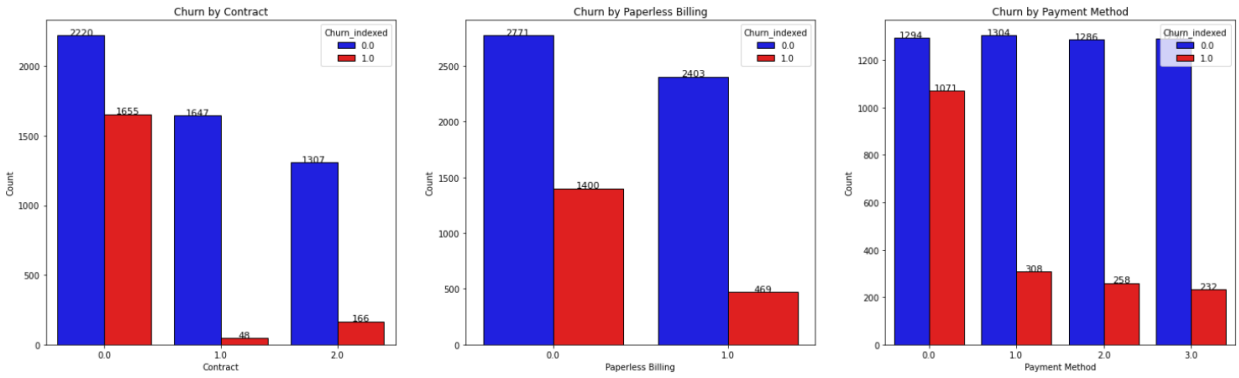

Figure 3: Countplots to visualize relationship between churn and 'Contract_indexed', 'PaperlessBilling_indexed', and 'PaymentMethod_indexed'.

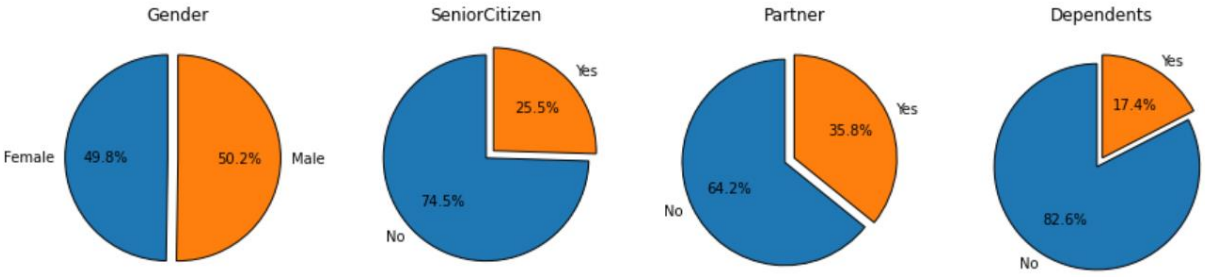

Figure 4: Percentages for each category.

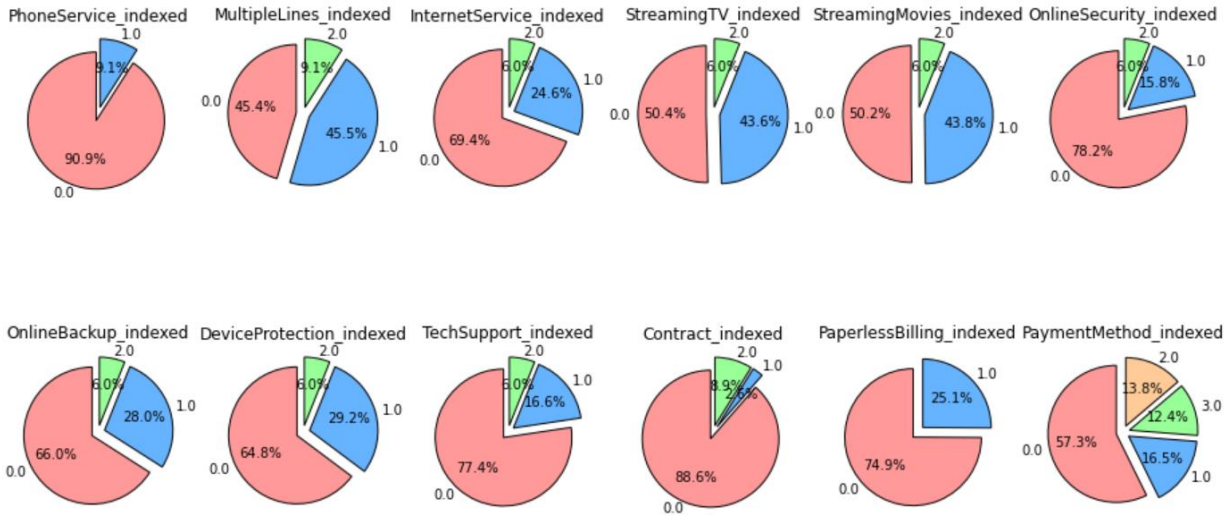

Figure 5: A set of pie charts to visualize the distribution of the churned customers for each service.

By filtering churned customers, we analyzed the distribution of churned customers for each service, plotting pie charts to represent the distribution of churned customers per service.

2.4 Feature Selection

To identify the most relevant features for machine learning algorithms, we created a heatmap to visualize correlations between features as shown in Figure 6 and 7. We dropped columns with low correlation to 'Churn_indexed' to form a new DataFrame (df1).

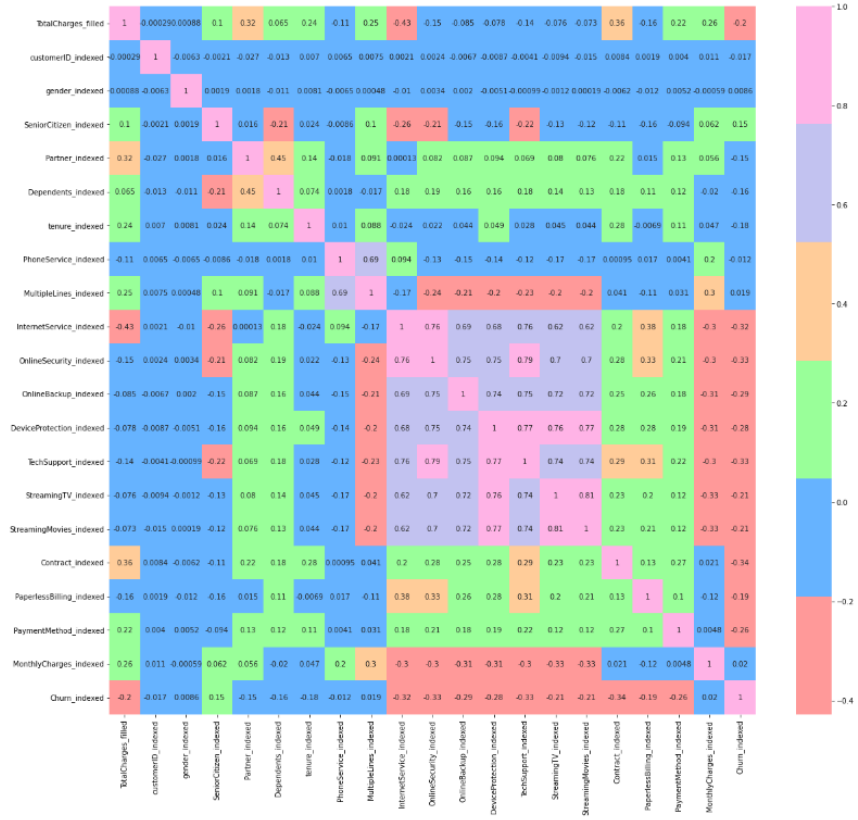

Figure 6: Heatmap to visualize correlations between features.

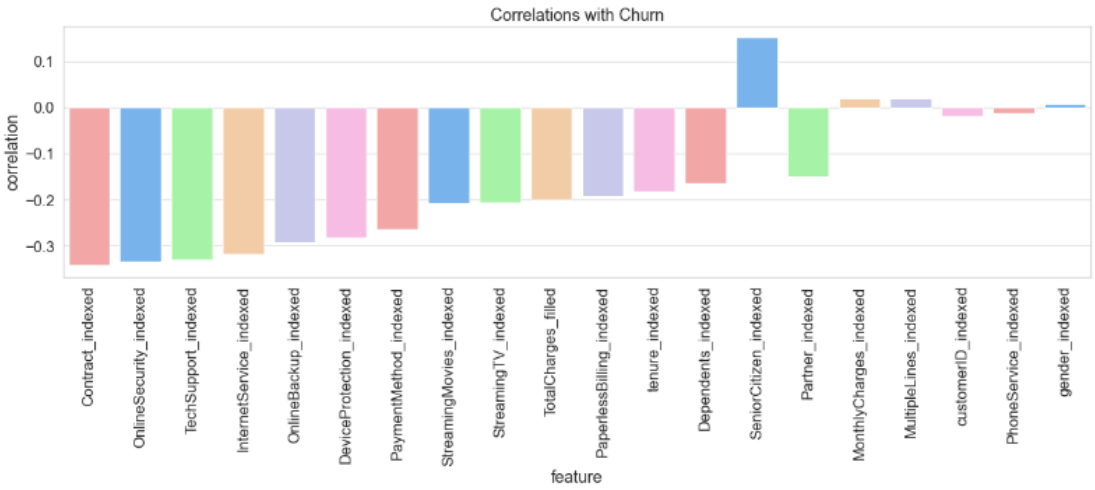

Figure 7: A bar plot of the correlations between the features and 'Churn_indexed'.

2.5 Data Balancing

To balance the dataset, we oversampled negative cases and combined them with positive cases to create a new DataFrame (balanced_df). We then reviewed the schema of the balanced DataFrame to ensure data types for each column remained consistent as shown in Table 1. This balanced dataset was then used for training and evaluating the chosen machine learning algorithms, ensuring that the models are not biased towards any particular class.

Table 1: shows the distribution of churned (1.0) and non-churned (0.0) customers in balanced_df.

Churn_indexed	Count
1.0	1869
0.0	1891

3. Spark Implementation

In this section, various classification models utilizing machine learning algorithms can be employed to predict customer churn in the telecommunications industry. We focus on tree-based and logistic regression algorithms, including DecisionTreeClassifier, RandomForestClassifier, GBTClassifier, and LogisticRegression.

3.2 DecisionTreeClassifier

We started by implementing the DecisionTreeClassifier, a tree-based method that recursively splits the input space for predictions. We trained a decision tree model for customer churn prediction by specifying the features and target variables. We selected the model's parameters, such as the maximum depth, to optimize the model's performance. After training the model, we evaluated its performance using the test dataset and calculated relevant metrics, such as accuracy and the area under the ROC curve.

3.3 RandomForestClassifier

Next, we implemented the RandomForestClassifier, an ensemble technique aggregating multiple decision trees to enhance prediction accuracy and prevent overfitting. We trained a random forest model for customer churn prediction, specifying the features and target variable and the number of trees to be used in the ensemble. We adjusted the model's parameters to optimize its performance. After training the RandomForestClassifier, we evaluated its performance using the test dataset and calculated the accuracy and the area under the ROC curve.

3.4 GBTClassifier

We then implemented the GBTClassifier, another ensemble method that combines multiple weak learners (decision trees) in a stage-wise fashion to improve prediction accuracy. We trained a gradient-boosted trees model for the customer churn prediction problem, specifying

the features and target variable, and selected the model's parameters, such as the maximum number of iterations, to optimize its performance. After training the GBTClassifier, we evaluated its performance using the test dataset and calculated relevant metrics, such as accuracy and the area under the ROC curve.

3.5 LogisticRegression

Finally, we implemented the LogisticRegression algorithm, a linear model that predicts the probability of a binary response based on one or more input features. We trained a logistic regression model for the customer churn prediction problem, specifying the features and target variable, and adjusted the model's parameters, such as regularization strength, to optimize its performance. After training the LogisticRegression model, we evaluated its performance using the test dataset and calculated relevant metrics, such as accuracy and the area under the ROC curve.

After implementing the selected machine learning algorithms, a comprehensive analysis was conducted to evaluate their performance under various data partitions. These partitions were strategically set at 2, 4, 8, 16, and 32 to observe the effect of data distribution on model efficiency and accuracy. In order to further investigate the scalability and parallelism of our algorithms, these specified partition configurations were subjected to multiple runs utilizing different numbers of workers. This approach, employing configurations of 2, 4, and 7 workers, allowed us to discern the impact of varying worker counts on the performance and execution time of the algorithm, as elucidated in Section 4.2.

Subsequently, in Section 4.3, we shifted our focus to the models' performance when operated with specific worker counts alone. These counts included 2, 4, 6, and 7 workers. This analysis stage aimed to uncover the influence of these exclusive workers on the models' performance, thereby enriching our understanding of the interplay between worker counts and machine learning algorithm performance.

4. Results

This section presents the results from applying the machine learning algorithms discussed in the previous section to the preprocessed Telco Customer Churn dataset. We will compare the performance of each model based on their accuracy, training time, evaluation time, number of partitions, and number of workers.

4.2 Performance Metrics Across Varying Data Partitions with Workers

1. DecisionTreeClassifier

Table 2 presents a comprehensive breakdown of the results of applying the DecisionTreeClassifier model across varying data partition sizes using two workers. Figure 8 provides a graphical representation of these results, facilitating a clear visualization of the model's performance across different partition configurations.

Table 2: shows the results of DecisionTreeClassifier for each partition using two workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.6944	17.80	0.69
4	0.6944	18.36	0.61
8	0.6944	22.68	0.96
16	0.6944	32.34	0.92
32	0.6944	43.65	1.11

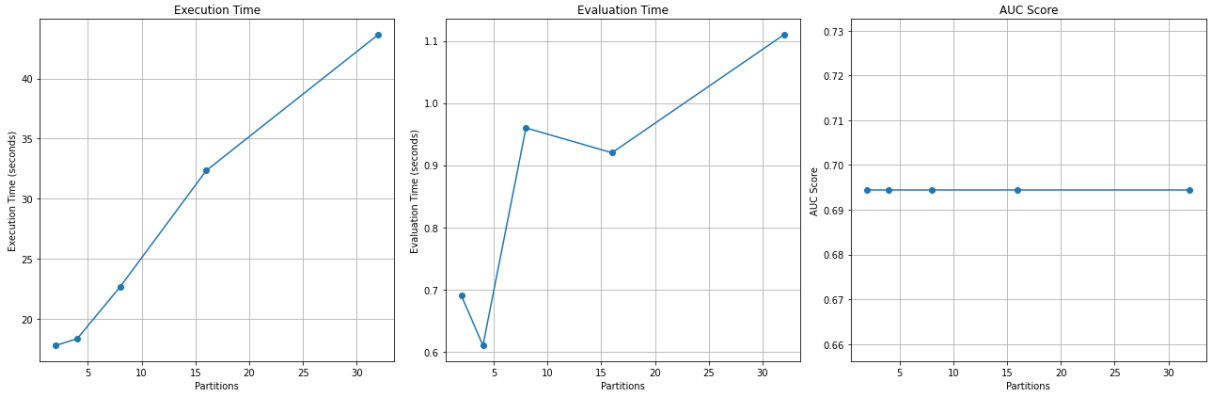

Figure 8: Decision Tree results showed as plots.

Table 3 presents the results of using four workers instead and Figure 9 provides the graphical representation of these results.

Table 3: shows the results of DecisionTreeClassifier for each partition using four workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.7512	9.78	3.80
4	0.7512	5.27	1.80
8	0.7512	5.05	1.68
16	0.7512	5.75	1.73
32	0.7512	7.10	1.98

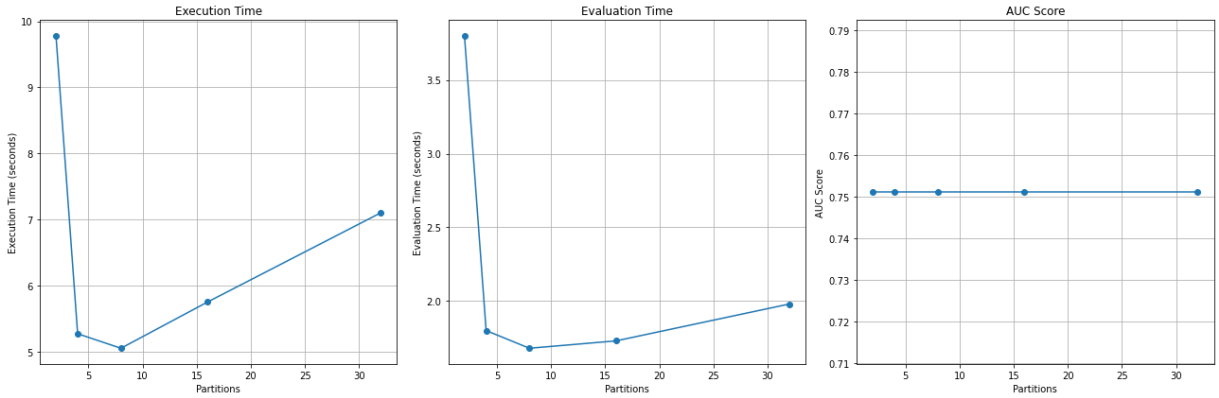

Figure 9: Decision Tree results showed as plots.

Table 4 presents the results of varying data partition sizes using seven workers and Figure 10 provides a graphical representation of these results.

Table 4: shows the results of DecisionTreeClassifier for each partition using seven workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.6965	10.04	3.40
4	0.6965	7.09	1.78
8	0.6965	6.61	1.94
16	0.6965	5.59	1.82
32	0.6965	7.90	2.02

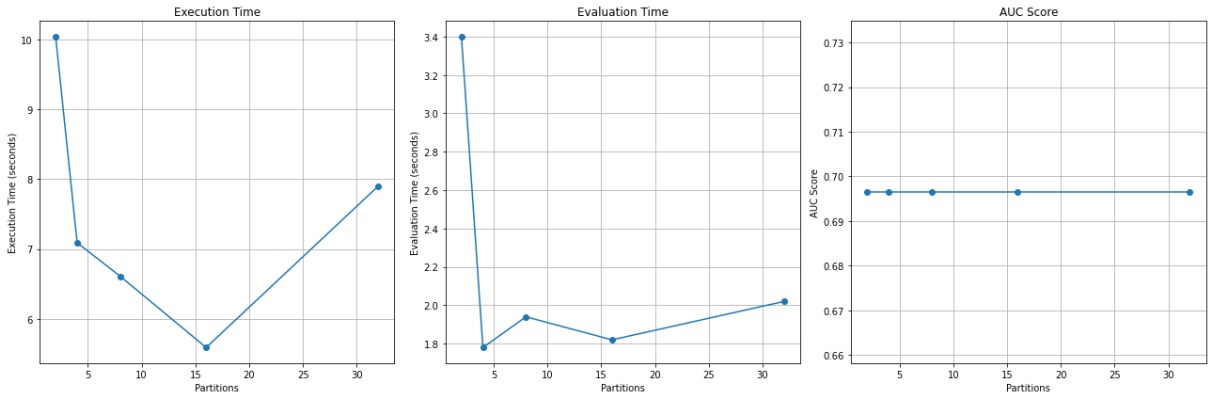

Figure 10: Decision Tree results showed as plots.

2. RandomForestClassifier

Table 5 presents a comprehensive breakdown of the results of applying the RandomForestClassifier model across varying data partition sizes. Figure 11 provides a graphical representation of these results, facilitating a clear visualization of the model's performance across different partition configurations.

Table 5: shows the results of RandomForestClassifier for each partition using two workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.7725	39.37	0.77
4	0.7530	43.06	1.00
8	0.7839	51.28	1.13
16	0.7750	70.43	1.29
32	0.7648	107.47	1.98

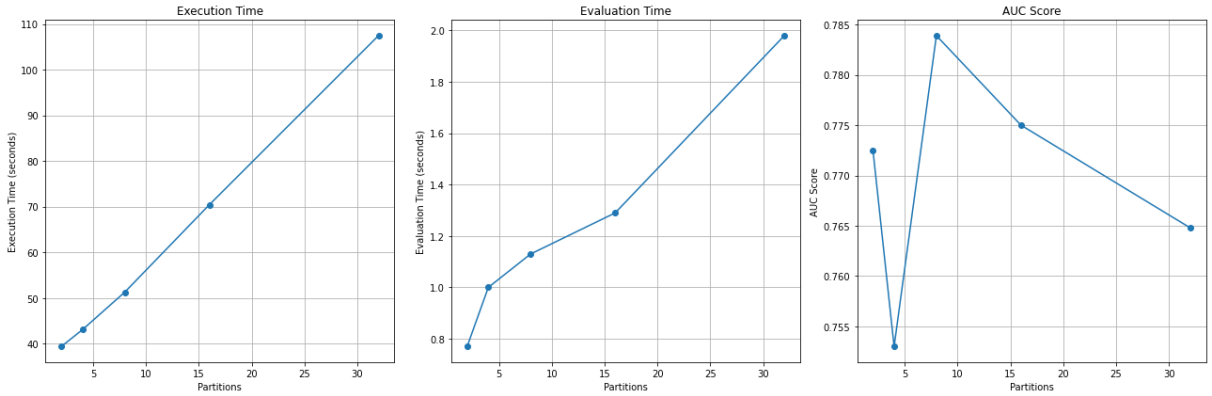

Figure 11: Random Forest results showed as plots.

Table 6 presents the results of applying the model across varying data partition sizes using four workers while Figure 12 provides a graphical representation of these results.

Table 6: shows the results of RandomForestClassifier for each partition using four workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.7830	7.82	1.27
4	0.7747	7.24	1.44
8	0.7782	7.74	1.52
16	0.7792	9.20	1.94
32	0.7720	13.45	2.56

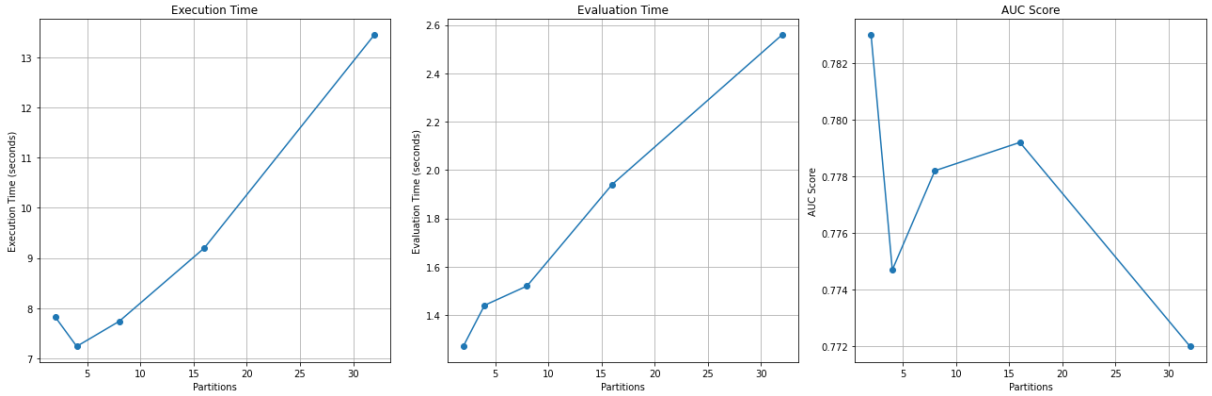

Figure 12: Random Forest results showed as plots.

Table 7 presents the results of seven workers and Figure 13 provides a graphical representation of these results.

Table 7: shows the results of RandomForestClassifier for each partition using seven workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.7661	7.01	1.26
4	0.7557	6.82	1.14
8	0.7549	7.59	1.49
16	0.7509	9.24	1.88
32	0.7434	13.64	3.22

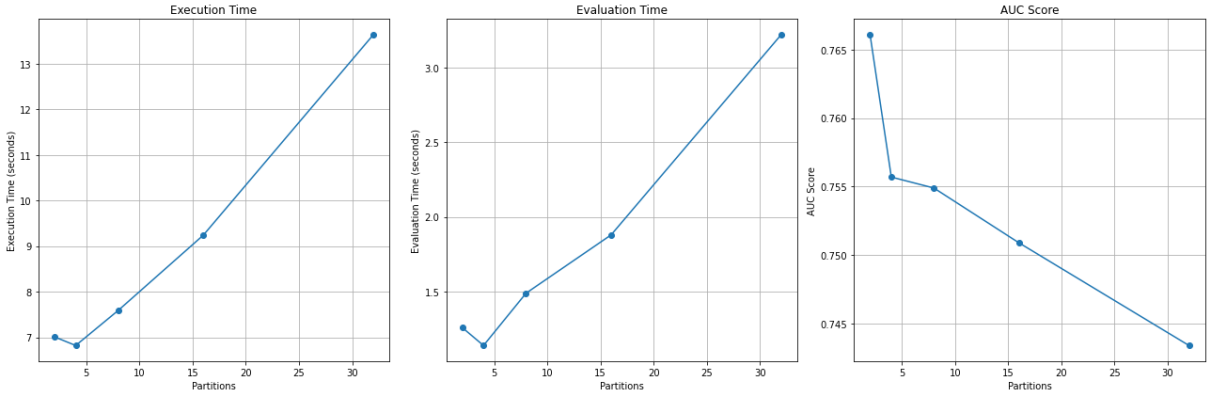

Figure 13: Random Forest results showed as plots.

3. GBTClassifier

Table 8 presents a comprehensive breakdown of the results of applying the GBTClassifier model across varying data partition sizes. Figure 14 provides a graphical representation of these results, facilitating a clear visualization of the model's performance across different partition configurations.

Table 8: shows the results of GBTClassifier for each partition using two workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.6829	188.99	0.78
4	0.6829	245.37	0.91
8	0.6829	359.66	1.12
16	0.6829	578.96	1.66
32	0.6829	1001.33	2.50

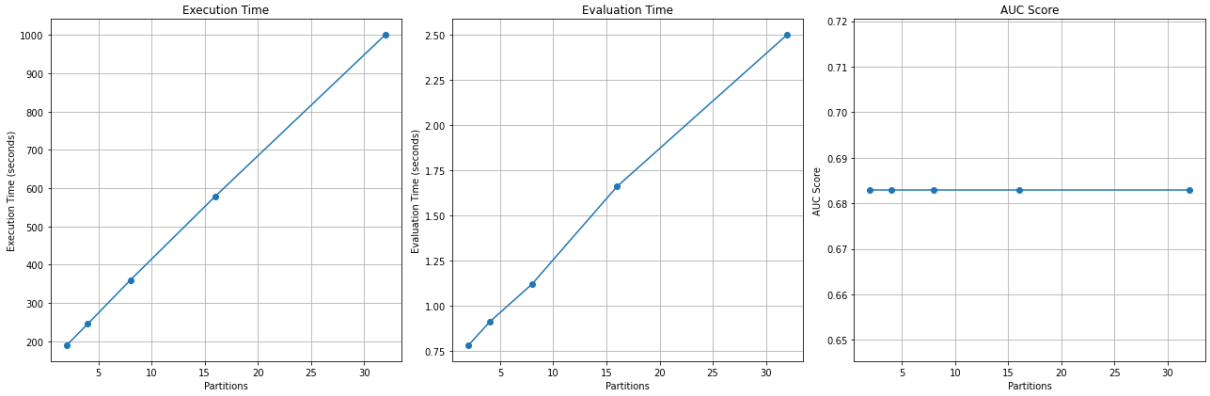

Figure 14: GBTClassifier results showed as plots.

Table 9 the results of the model across varying data partition sizes using four workers. Figure 15 provides a graphical representation of these results.

Table 9: shows the results of GBTClassifier for each partition using four workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.6255	28.40	1.05
4	0.6255	31.83	1.03
8	0.6257	42.46	1.16
16	0.6257	53.09	1.29
32	0.6257	63.72	1.42

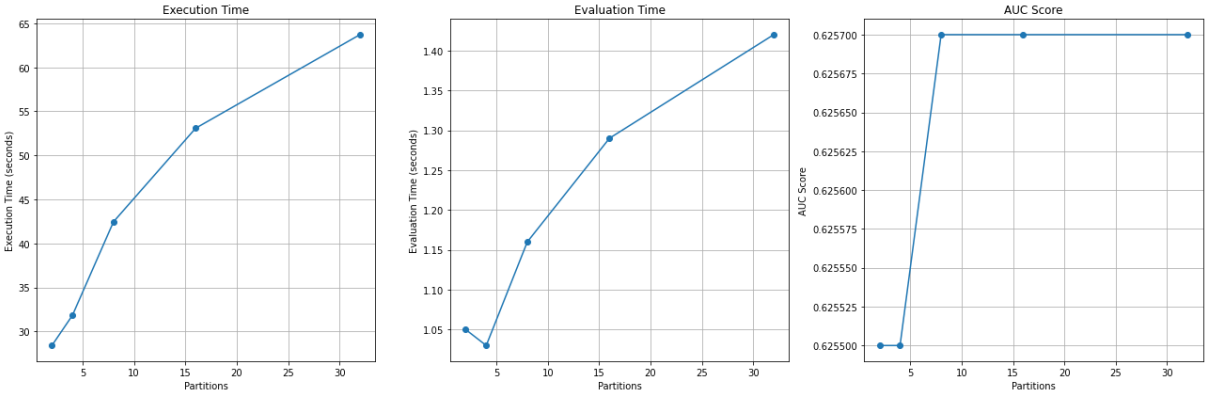

Figure 15: GBTClassifier results showed as plots.

Table 10 the results using seven workers. Figure 16 provides a graphical representation of these results.

Table 10: shows the results of GBTClassifier for each partition using seven workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.5966	27.78	1.12
4	0.5965	32.74	1.03
8	0.5965	44.25	1.29
16	0.5965	55.76	1.55
32	0.5965	67.27	1.81

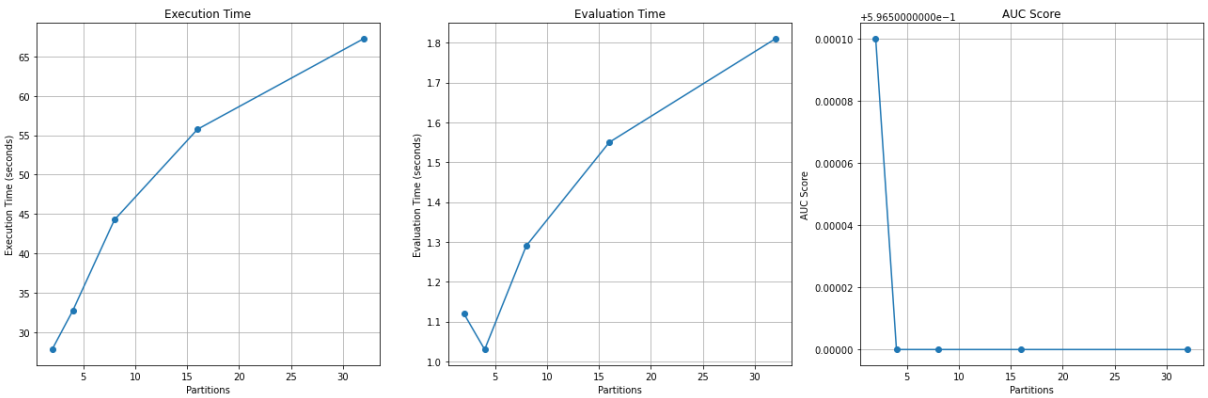

Figure 16: GBTClassifier results showed as plots.

4. LogisticRegression

Table 11 presents a comprehensive breakdown of the results of applying the LogisticRegression model across varying data partition sizes. Figure 17 provides a graphical representation of these results, facilitating a clear visualization of the model's performance across different partition configurations.

Table 11: shows the results of LogisticRegression for each partition using two workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.8346	18.07	0.58
4	0.8346	20.13	0.66
8	0.8346	38.91	0.80
16	0.8346	60.18	0.97
32	0.8346	88.14	1.43

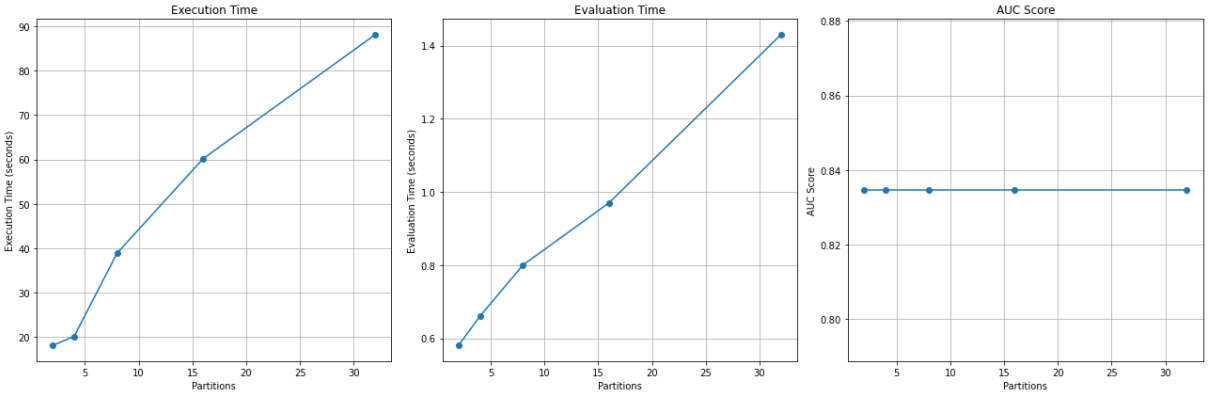

Figure 17: LogisticRegression results showed as plots.

Table 12 presents the results of using four workers instead and Figure 18 provides the graphical representation of these results.

Table 12: shows the results of LogisticRegression for each partition using four workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.8231	7.04	0.80
4	0.8231	5.78	0.98
8	0.8231	9.19	1.02
16	0.8231	12.60	1.06
32	0.8231	16.01	1.10

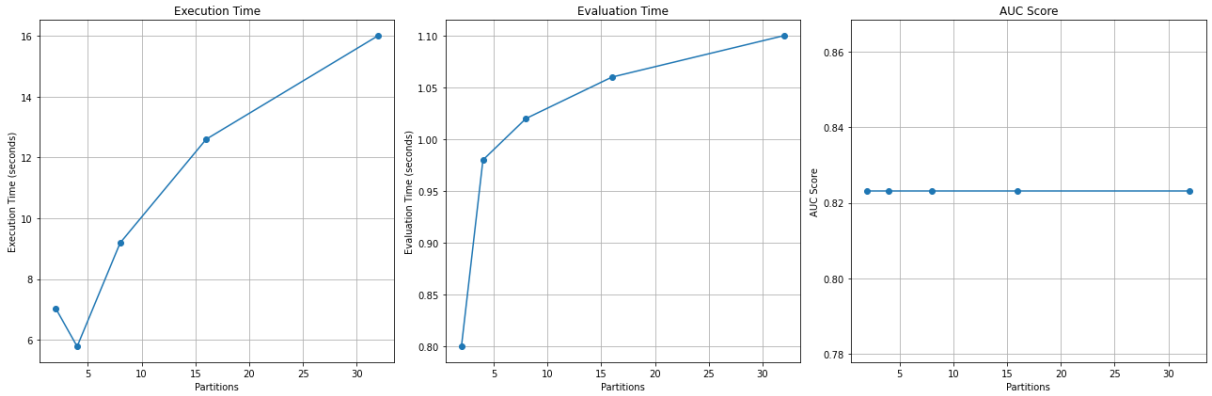

Figure 18: LogisticRegression results showed as plots.

Table 13 presents the LogisticRegression model's results using seven workers; Figure 19 provides the graphical representation.

*T*able 13: shows the results of LogisticRegression for each partition using seven workers.

Partitions	AUC Score	Time Taken (seconds)	Evaluation Time (seconds)
2	0.8140	14.78	2.59
4	0.8140	9.48	1.53
8	0.8140	13.47	1.58
16	0.8140	16.35	1.81
32	0.8140	22.73	2.32

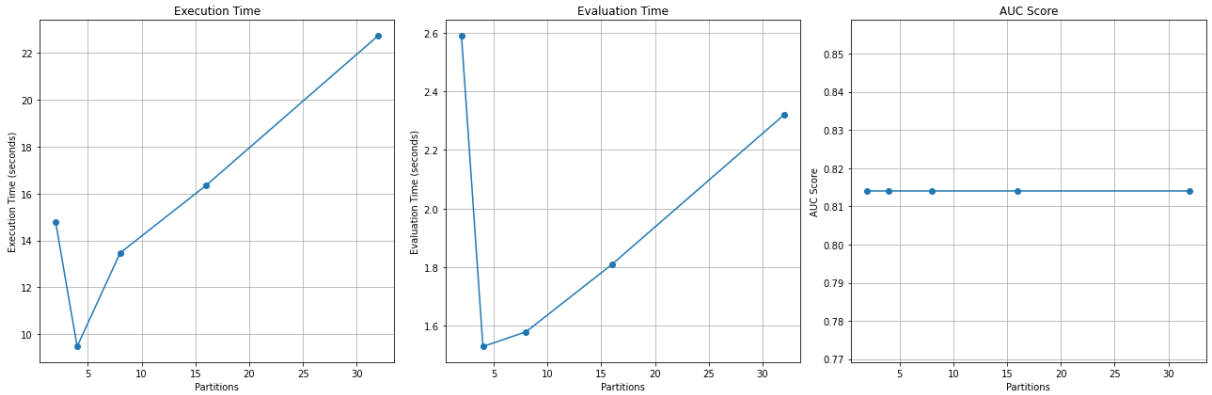

Figure 19: LogisticRegression results showed as plots.

4.3 Performance Metrics Across Varying Workers

The table presented below provides an overview of the results obtained from the four machine learning models. These models were tested using various workers, specifically 2, 4, 6, and 7. Figures 20, 21, and 22 translate this data into bar plots for a more visual and intuitive understanding of these outcomes. These graphical representations enable easier comparison and

evaluation of the machine learning models' performances across the different worker configurations.

Table 14: shows the results of all models with different numbers of workers.

Workers	Model	AUC Score	Training Time (Seconds)	Evaluation Time (Seconds)
2	DecisionTree	0.7271	9.9492	1.9701
2	RandomForest	0.7583	9.7997	1.5379
2	GBTClassifier	0.5940	34.7232	1.7151
2	LogisticRegression	0.8216	9.0247	1.3609
4	DecisionTree	0.6856	10.4694	4.0456
4	RandomForest	0.7562	9.3998	1.6949
4	GBTClassifier	0.6157	37.2921	1.4036
4	LogisticRegression	0.8241	8.5571	1.1517
6	DecisionTree	0.7129	10.5315	2.6206
6	RandomForest	0.7814	9.7788	1.6876
6	GBTClassifier	0.5804	38.8643	1.7392
6	LogisticRegression	0.8386	8.6998	1.1169
7	DecisionTree	0.6845	10.8111	3.0871
7	RandomForest	0.7368	9.8017	1.6610
7	GBTClassifier	0.6165	40.4467	1.3737
7	LogisticRegression	0.8146	10.1691	1.1760

Figure 20: Comparative Visualization of Training Durations Across All Algorithms

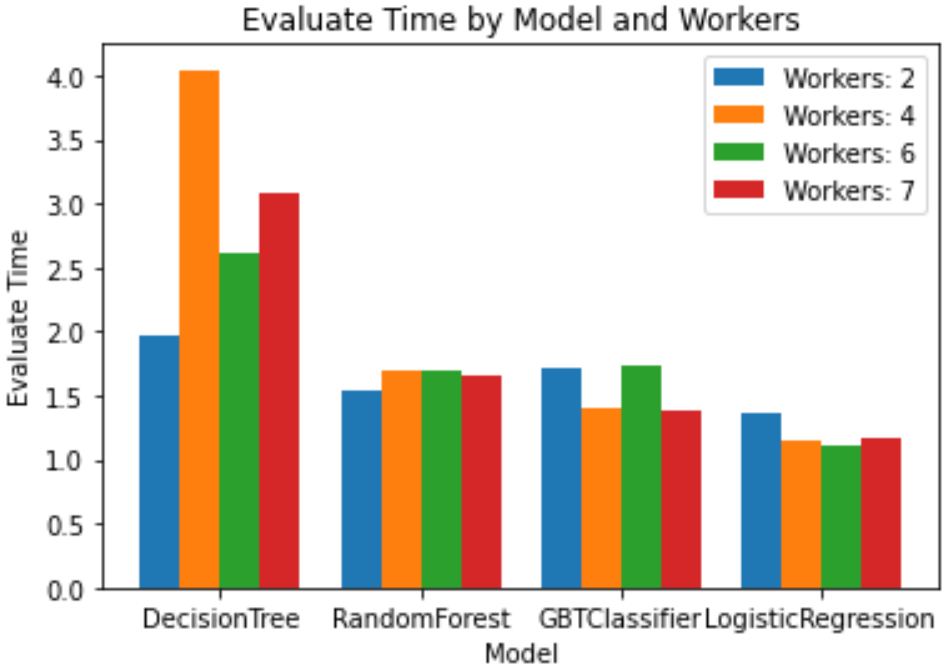

Figure 21: Comparative Visualization of Evaluation Durations Across All Algorithms.

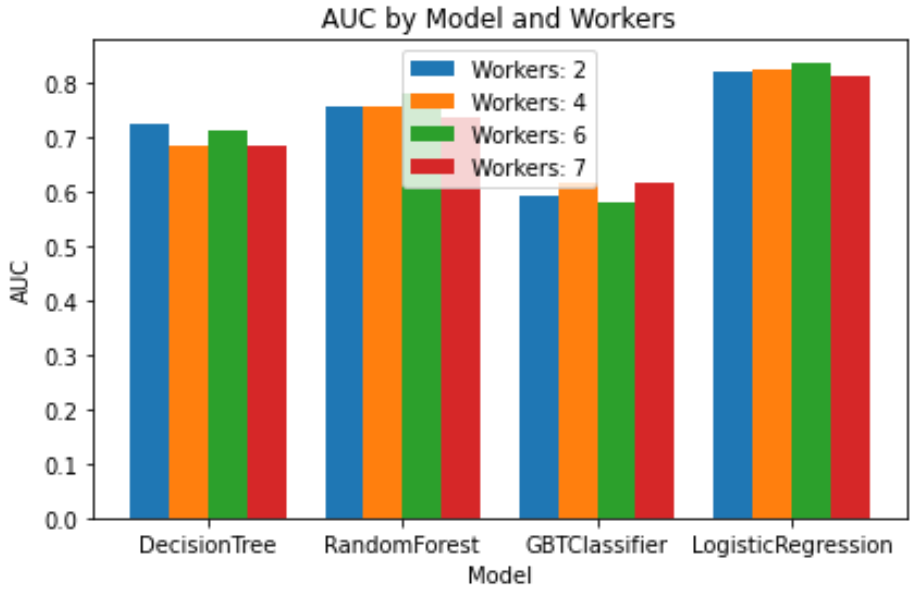

Figure 22: Comparative Visualization of AUC Values Across All Algorithms.

5. Discussion

5.2 General Discussion

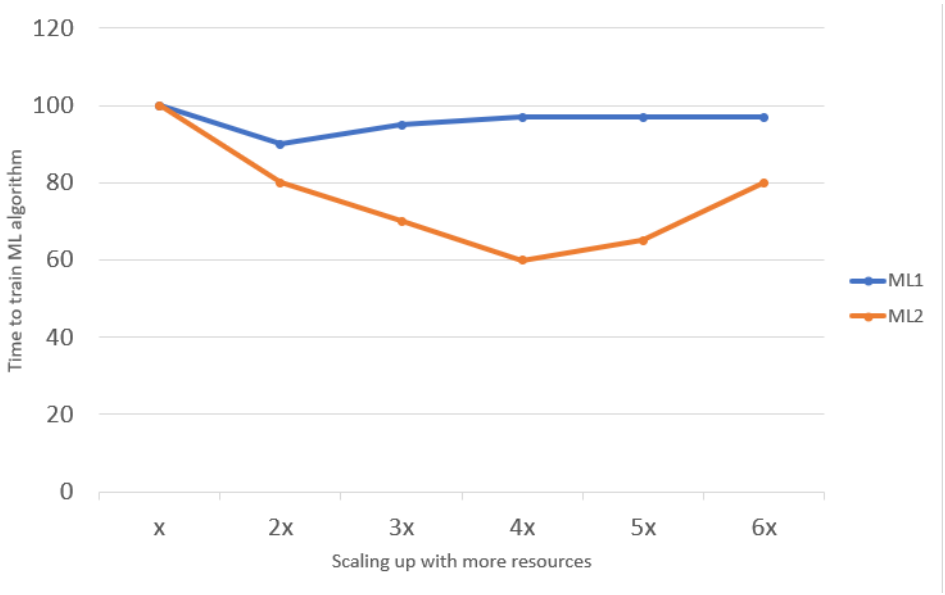

Figure 23: Time taken to train two different machine learning algorithms using Spark cluster with increasing number of workers on the x-axis.

In order to compare both machine learning models, ML1 and ML2, according to Figure 23, shows a decrease in training time for both models with an initial increase in the number of workers. This is expected because parallelization of tasks in machine learning across multiple nodes generally tends to decrease the overall training time, assuming the problem is amenable to parallelization and the system has been appropriately designed to avoid data distribution or communication bottlenecks.

ML1 shows decreased training time from 100 seconds to 90 seconds when moving from one worker to two workers. However, the time taken to train the model increases beyond two workers and remains consistent at 98 seconds. This behavior could indicate that ML1 experiences diminishing returns with added workers, potentially due to issues like communication overhead between the workers, inefficient parallelization, or bottlenecks in the data pipeline.

On the other hand, ML2 demonstrates consistent time decreases as more workers are added, going from 100 seconds with one worker down to 60 seconds with four workers. This means ML2 benefits more effectively from parallel processing. However, the training time increases again when there are more than four workers. This could indicate that the system managing ML2 reaches its optimal capacity at four workers, and beyond that, adding more workers could result in increased communication or synchronization overhead, diminishing the overall benefit.

Another aspect to consider could be Amdahl's Law, which essentially states that the maximum improvement in speed from parallelization is limited by the portion of the task that cannot be parallelized. ML2 appears to have a more significant portion of its workload that can be

parallelized compared to ML1, hence the more significant decrease in training time with an increase in the number of worker nodes.

Therefore, while both ML1 and ML2 exhibit decreases in training time with added worker nodes, ML2 seems to handle parallelization more efficiently, up to a certain point (four workers in this case). Beyond that point, both models appear to experience a plateau or even an increase in training time, possibly due to communication overhead or other limitations in the parallelization infrastructure.

The best cluster setting for ML2 is four workers. This was determined based on the minimum training time achieved, which was 60 seconds, compared to the longer training times required with other cluster sizes.

The benefits of parallelization did not extend indefinitely with the addition of more workers. It was noted that the training time increased beyond the four-worker mark. This phenomenon can be attributed to several factors. A key factor is the overhead associated with worker coordination and communication. As more workers are added to the cluster, the time required for these coordination tasks outweighs the time saved through parallel processing.

In the context of ML2, the cluster size of four workers represented an equilibrium point, where the benefits of parallelization were maximized while the costs, in terms of time, were minimized. However, it is crucial to acknowledge that this optimal point can vary depending on the specifics of the ML algorithm, the dataset characteristics, and the resources and infrastructure available.

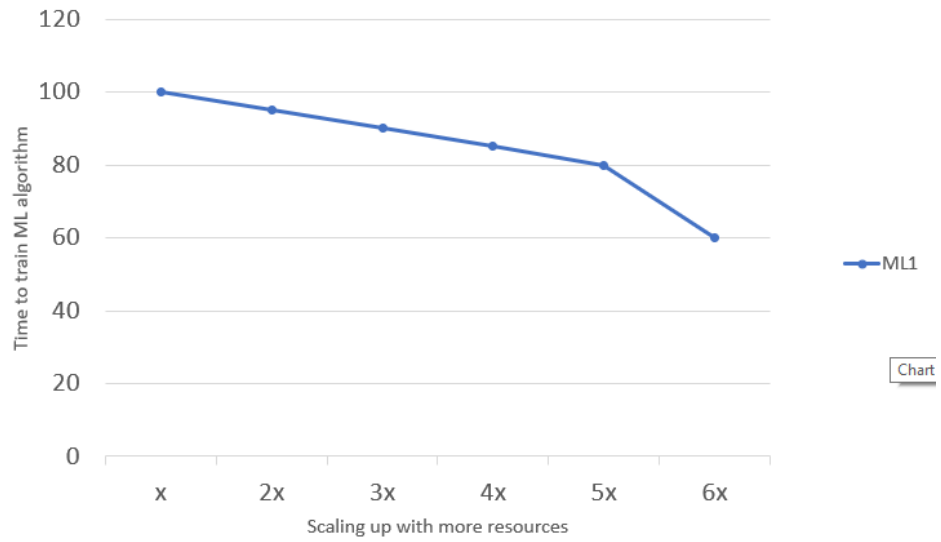

Figure 24: Time taken to train a machine learning algorithm while increasing the number of worker nodes in the cluster as shown in the x-axis, while the number of RDD partitions is fixed and equals 2.

The plot presented in Figure 24 raises several intriguing questions and potential issues concerning the performance optimization of the machine learning algorithm (ML3). There is a noticeable irregularity in the training times recorded when varying the number of worker nodes while maintaining a static number of data partitions (2 partitions).

Specifically, the transition from 4 to 5 worker nodes resulted in an unexpected increase in training time from 60 seconds to 80 seconds, counter to the intuitive expectation of stable or

reduced training time with additional computational resources. Additionally, further increasing the number of worker nodes to 6 caused a decrease in the training time back to 60 seconds. This non-linear, non-monotonic behavior strongly suggests the presence of underlying factors that are influencing the training time.

Crucially, we must address the experimental decision to keep the number of data partitions constant at 2. In scenarios where the number of partitions is less than the number of CPUs (or workers, as in this case), we risk underutilization of available resources, with some workers potentially idle while others process data. On the other hand, an excessively high number of partitions can lead to increased overhead associated with managing multiple partitions and the communication between them, adversely affecting performance.

In the context of this experiment, maintaining a constant two partitions irrespective of the number of worker nodes may yield less meaningful and efficient results, particularly when the number of workers exceeds 2. With 3, 4, 5, or 6 workers, some worker nodes likely need more available partitions to process. This inefficiency could account for the irregular behavior observed during the training times.

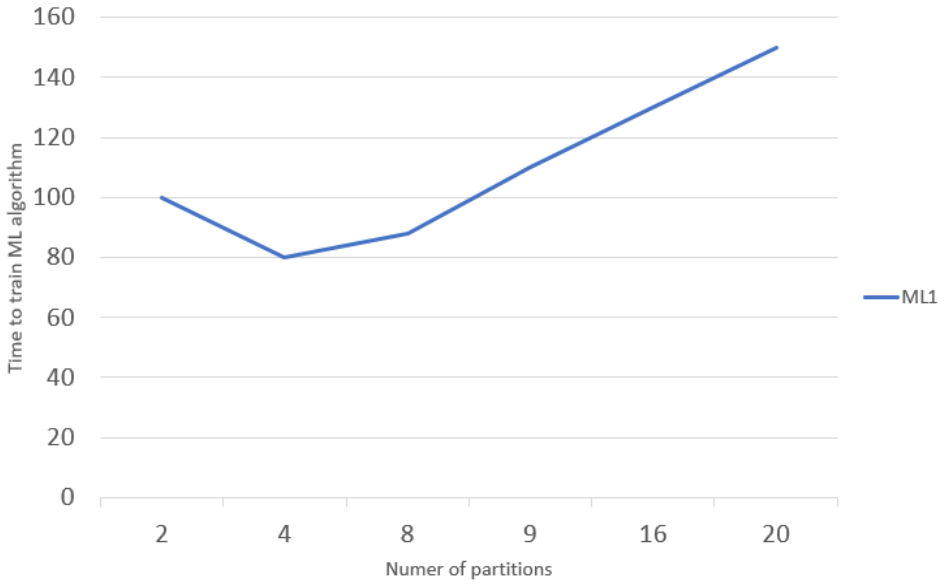

Figure 25: Time taken to train a machine learning algorithm using a cluster of 4 worker nodes and changing the number of partitions in RDD containing training data.

The theoretical shape of the curve in Figure 25 can be described as a U-shaped curve, showing the initial reduction in training time with an increase in the number of partitions and a subsequent rise in training time beyond a specific number of partitions.

In computational terms, 'overhead' denotes the extra tasks that need to be performed, indirectly related to actual data processing. When dealing with a larger number of partitions, these overhead tasks can demand significant time and resources, negating the benefits of parallel processing. The overhead associated with an increase in partitions can be broken down into Task Scheduling and Initiation, Data Shuffling and Communication, and Data Serialization and Deserialization.

5.3 Discussion on "Performance Metrics Across Varying Data Partitions with Workers"

The performance results of the machine learning algorithms on the Telco Customer Churn dataset demonstrate exciting trends in terms of model performance, measured through the Area Under the Curve (AUC) score, and computational efficiency, evaluated through the training and evaluation times across various numbers of data partitions and workers.

The DecisionTreeClassifier model's AUC score remained consistent across varying partition sizes for a specific number of workers, indicating that the model's predictive power was not affected by changes in data partitioning. However, an interesting trend emerges when we compare the AUC scores across different worker configurations. Specifically, the AUC score improved when we moved from two to four workers but then slightly dropped when we further increased the number of workers to seven. This observation might suggest an optimal worker configuration for this model and data combination. Regarding computational efficiency, the time taken for training and evaluation generally increased with the number of partitions, which could be attributed to the overhead associated with managing more partitions.

Similar to the DecisionTreeClassifier, the RandomForestClassifier's AUC score was reasonably stable across varying partition sizes for a specific number of workers. However, the AUC score fluctuated slightly when varying the number of workers, indicating that the model performance may be sensitive to the worker configuration. Regarding computational efficiency, the time taken for training and evaluation tended to increase with the number of partitions, again likely due to the overhead of managing more partitions.

The GBTClassifier model demonstrated a similar trend regarding the AUC score remaining consistent across varying partition sizes for a given worker configuration. However, the AUC score dropped significantly when moving from two to four workers and decreased slightly with seven workers. This suggests that the GBTClassifier model may be sensitive to the worker configuration and perform better with fewer workers. Regarding computational efficiency, there was a clear trend of increasing training and evaluation time with the number of partitions.

The LogisticRegression model's AUC score remained consistent across varying partition sizes for a specific number of workers, similar to the trends observed in the other models. However, the AUC score decreased from two to four workers and further to seven workers, indicating sensitivity to the worker configuration. Regarding computational efficiency, the time taken for training and evaluation generally increased with the number of partitions, possibly due to the overhead of managing more partitions.

These results underscore the importance of appropriately configuring the data partitioning and worker nodes for computational efficiency and model performance. Moreover, they highlight that different models might respond differently to these configurations, underscoring the need for thorough testing and validation to identify optimal settings for a given model and dataset combination. Despite these variations, it is notable that the models' predictive performance, as measured by the AUC score, remained largely stable across different partition sizes, suggesting that these models' predictive power was not overly sensitive to data partitioning.

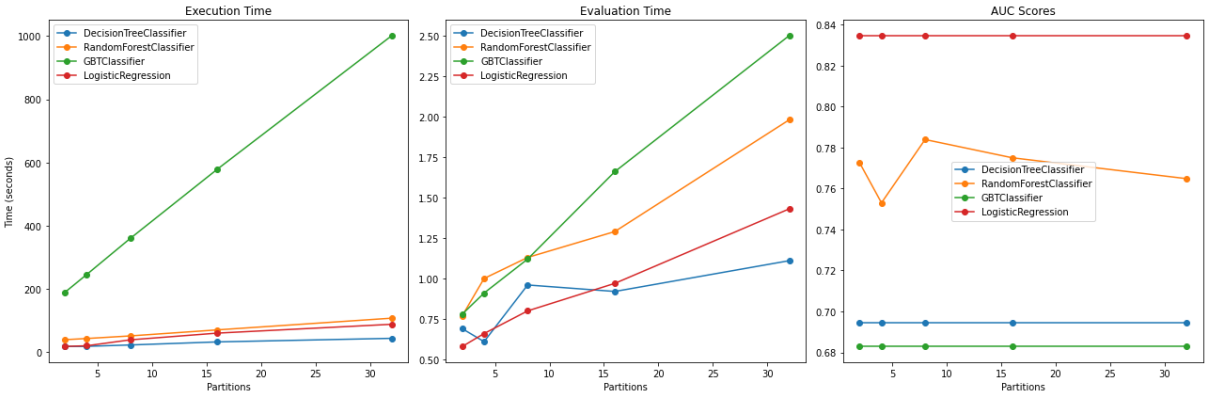

Figure 26: Comparative Visualization of All Models with partitions using two workers.

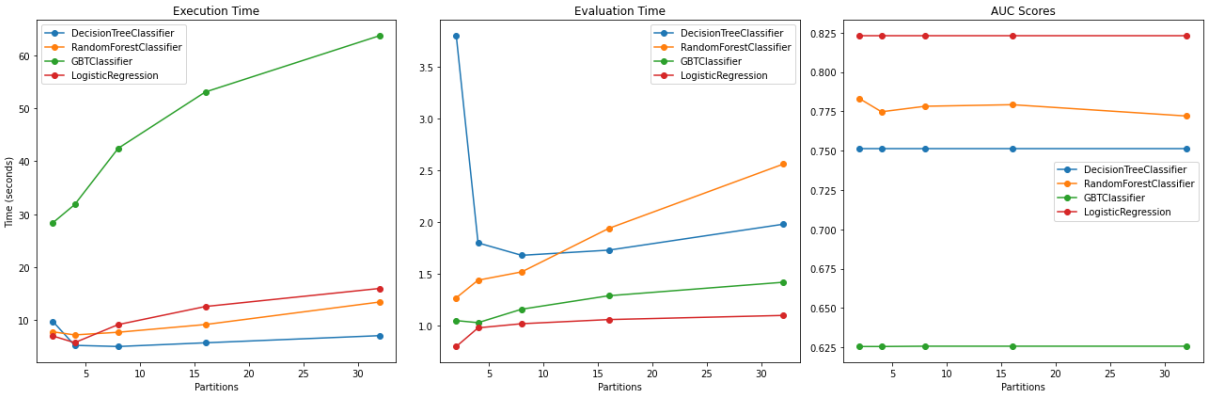

Figure 27: Comparative Visualization of All Models with partitions using four workers.

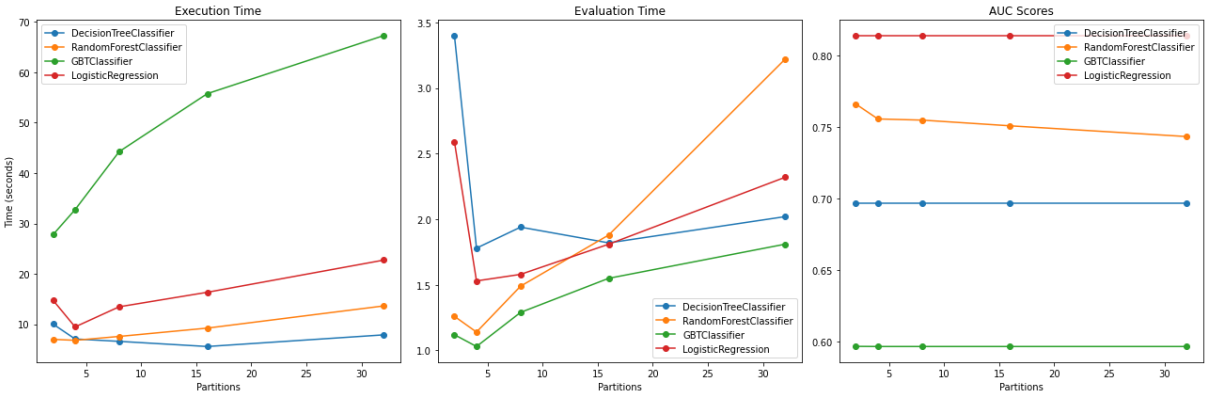

Figure 28: Comparative Visualization of All Models with partitions using seven workers.

5.4 Discussion on "Performance Metrics Across Varying Workers"

As seen from Table 14, the performance metrics used to compare these models are the Area Under the Curve (AUC) Score, Training Time, and Evaluation Time. The AUC score reflects the model's ability to classify the positive class from the negative class correctly. The higher the AUC score, the better the model's performance. On the other hand, the Training Time and Evaluation Time provide insights into the models' efficiency and speed.

Interestingly, Logistic Regression consistently exhibits the highest AUC scores across all workers, indicating superior classification performance. Logistic Regression reaches an AUC peak of 0.8386 when using six workers. Concurrently, it showcases efficient training and evaluation times, primarily reaching its best efficiency (lowest Training and Evaluation Time) at four workers.

Random Forest maintains a relatively stable AUC score across all worker configurations. It demonstrates a slight dip in performance when moving from six to seven workers (AUC of 0.7814 to 0.7368). Nonetheless, it is worth noting that the Random Forest model maintains relatively steady and low Training and Evaluation Times across all configurations, indicating its robustness to the number of workers.

In contrast, the Decision Tree model's AUC score seems less stable across different worker configurations. This model yields the highest AUC score of 0.7271 for two workers, after which the AUC score decreases and shows little consistency. The training time increases as more workers are added, from approximately 10 seconds for two workers to 10.81 seconds for seven workers. The evaluation time also needs to be consistent across the different worker configurations.

The GBTClassifier model, however, demonstrates the most significant difference in Training Time, considerably higher than the other models, which increases with the addition of more workers. Its AUC score remains comparatively low and less consistent, indicating that GBTClassifier may not efficiently leverage the benefits of additional workers in this context.

In conclusion, based on the results, Logistic Regression and Random Forest models demonstrate strong performance and efficiency across different worker configurations. They offer relatively high AUC scores and manage to maintain short training and evaluation times. However, the optimal number of workers may vary depending on the specific model and task. For instance, Logistic Regression achieves peak performance at six workers, while Random Forest's performance appears robust across all worker configurations. Conversely, the Decision Tree and GBTClassifier models exhibit less consistency in their performance metrics across varying numbers of workers. This suggests that these models may require additional tuning or optimization in their parallelization process to benefit more effectively from the increased computational resources.

6. Conclusion

This report comprehensively evaluated four machine learning models - DecisionTreeClassifier, RandomForestClassifier, GBTClassifier, and LogisticRegression - for predicting customer churn using the Telco Customer Churn dataset. The models were examined based on three crucial metrics: Area Under Curve (AUC) scores, training time, and evaluation time across varying partition sizes and worker numbers.

In comparing the four models' performance on varying data partitions and worker configurations, the AUC scores remained largely stable, suggesting that the predictive power of these models was not significantly affected by changes in data partitioning or worker configuration. However, the training and evaluation times generally increased with the number of partitions, likely due to the overhead of managing more partitions.

Specifically, Logistic Regression and Random Forest models showcased robust performance across different worker configurations, achieving relatively high AUC scores while maintaining short training and evaluation times. In contrast, Decision Tree and GBTClassifier models showed less consistency across different worker configurations, indicating the need for further tuning or optimization in their parallelization processes.

Future work could involve exploring more advanced techniques for handling imbalanced data, integrating ensemble learning methods, performing extensive feature engineering, and fine-tuning hyperparameters to optimize model performance. Additionally, considering other performance metrics would provide a more robust assessment of model performance.

In conclusion, the results underscore the importance of tailoring the parallelization strategy to the specifics of the machine learning algorithm, the data, and the available resources. The number of worker nodes and data partitions should be appropriately configured to balance computational efficiency and predictive performance. Furthermore, these configurations may need to be adjusted dynamically based on the machine learning task's evolving requirements and constraints. Through ongoing experimentation and validation, we can continue to refine our understanding of these dynamics and optimize the performance of machine learning applications.