Due: May 28th, 2023
CS3946: Advanced Machine Learning

Home Assignment 2: Ensemble Learning

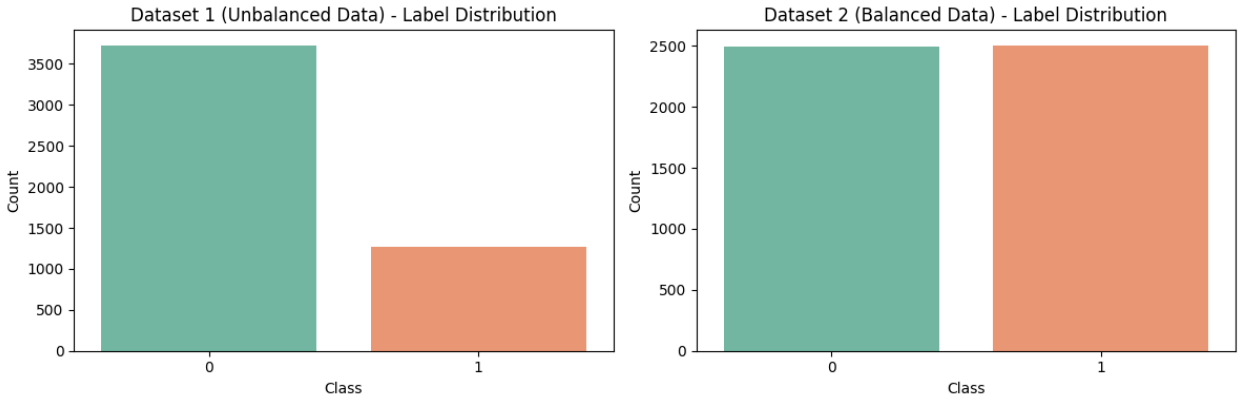
In this assignment, we explored two methods for Ensemble Learning: Gradient Boosted Regression Trees and AdaBoost. Ensemble learning combines the predictions of multiple individual models (called base learners) to improve the overall performance. By leveraging the collective knowledge of the base learners, the final model is able to make more accurate predictions than any individual model could. Boosting is an ensemble learning method that iteratively trains a sequence of base learners, where each subsequent learner focuses on correcting the mistakes of the previous learners by minimizing a loss function. The final learner is obtained by combining the predictions of all base learners. Both GBRT and AdaBoost are boosting algorithms, where the differences are within the algorithmic approach, weighting of samples and the training process. While GBRT focuses on minimizing the residuals using decision trees, AdaBoost adjusts sample weights to emphasize misclassified samples.

Data Generation

In order to test and demonstrate both boosting techniques, we generate two datasets with binary labels. The generated datasets used in this study were created synthetically using the make_classification function from the sklearn.datasets module. Two distinct datasets, labeled Dataset 1 and Dataset 2, were generated, each possessing unique characteristics that could potentially impact classification tasks. Dataset 1 consisted of 5000 samples, each characterized by 5 features. Among these features, 4 were informative, while no redundant features were specifically generated. The class distribution in Dataset 1 was intentionally skewed to have 75% of the samples belonging to one class and 25% belonging to the other class. The data generation process for Dataset 1 employed a specific random seed value (42) to ensure replicability. The target variable in Dataset 1 was transformed into a binary format, with class labels converted to 0s and 1s. Dataset 2 also comprised 5000 samples, but was generated with 7 features. Among these features, 5 were informative, containing significant information relevant to the classification problem, while 2 features were redundant and potentially introduced noise. The data was generated with an equal distribution of labels. The data generation process for Dataset 2 employed a different random seed value (11) for replicability. The target variable in Dataset 2 was also transformed into a binary format, with class labels converted to 0s and 1s.

These dataset characteristics were selected as we wanted to demonstrate their influence on the performance of Gradient Boosted Regression Trees (GBRT) and AdaBoost. GBRT seeks to minimize residuals through the utilization of decision trees, and its effectiveness can be influenced by the number of features and how many are informative. Dataset 2's greater number of informative features and balanced classes may enhance the algorithm's ability to capture underlying patterns and make accurate predictions. However, the presence of redundant features in Dataset 2 could challenge the algorithm by introducing noise. Like GBRT, AdaBoost's performance may also be influenced by the characteristics of the datasets. AdaBoost adjusts sample weights during training. We hypothesized that Dataset 2's higher number of informative features

could potentially improve AdaBoost's ability to learn and assign appropriate weights to misclassified samples, leading to enhanced performance

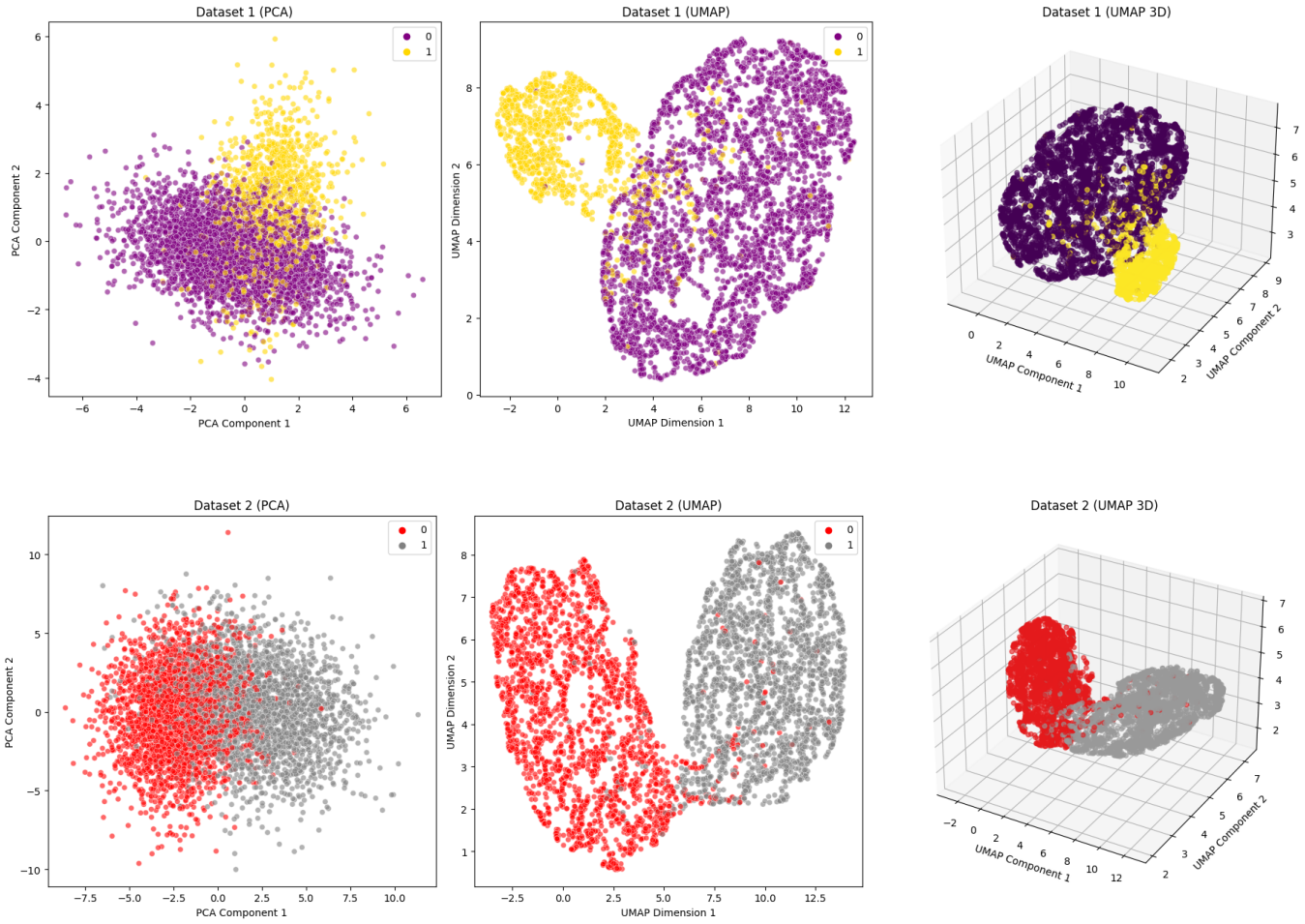


In this study, we deliberately introduced variations in class balance between the two datasets to examine the impact of imbalance on the learning process and evaluate the performance of classification algorithms. Algorithms tend to be more biased towards predicting the majority class, leading to a misclassification of the minority class, resulting in lower accuracy, precision and recall. The clustering of the samples per class may also impact the distribution of weights assigned to the samples, with well-separated clusters receiving higher weight assignments. The clustering of samples per class in the datasets may affect whether our algorithms are able to accurately capture class boundaries. Well-separated clusters facilitate the creation of effective decision boundaries, whereas overlapping clusters may pose challenges to accurate classification.

Dimensionality Reduction

To further investigate the impact of these dataset characteristics, we employed dimensionality reduction techniques. By visualizing the reduced-dimensional representations of the datasets, we gained insights into the inherent structure and separability of the data based on the features used for classification. The results of the dimensionality reduction analysis revealed interesting patterns. In Dataset 1, which exhibited class imbalance, the features showed significant overlap even after applying both PCA and UMAP. This indicates that the original features were not able to sufficiently differentiate the data points, suggesting a lack of inherent structure or separability in the data based on those features. We found that Dataset 2, which had a more balanced class distribution, exhibited much more clear separation of clusters, suggesting that the features in Dataset 2 contain more differentiable information.

These findings have implications for the estimation of accuracy of GBRT and AdaBoost on these datasets. As observed in the dimensionality reduction visualizations, the lack of differentiation in Dataset 1 suggests that the data may be inherently challenging to classify accurately based on the available features. This indicates that GBRT and AdaBoost may face difficulties in capturing the underlying patterns of the data without parameter tuning.



Gradient Boosting Regression Trees (GBRT)

Gradient Boosting Regression Trees (GBRT) is an advanced machine learning technique that combines gradient descent and boosting to optimize the weights of base learners in regression tasks. The aim of GBRT is to minimize the loss function iteratively, seeking the most suitable weights for the base learners. In our implementation, we employed the mean squared error (MSE) as the loss function, which calculates the average of the squared differences between the predicted and actual target values. At each iteration of the GBRT boosting process, the following steps are performed:

1. Initialization: The predicted values for both the training and test datasets are initialized, either as 0 or as the mean of the target variable.
2. Iteration: For each iteration, determined by the specified number of estimators, the following actions are taken:
    2.1. Residual Calculation: The residuals are computed as the differences between the actual target values and the predictions made by the current ensemble of base learners.

2.2. Base Learner Fitting: The base learner is fitted to the training data using the current residuals. This step aims to improve the predictions by capturing the remaining patterns or trends in the data.
2.3. Prediction Generation: Using the trained base learner, new predictions are generated for the data.
2.4. Learning Rate Adjustment: The predictions are multiplied by a learning rate and combined with the previous predictions. This aggregation process allows the model to gradually adapt and refine its predictions based on the base learners' performance.

3. Final Prediction: The ultimate prediction is obtained by converting the predicted values into binary labels. A threshold of 0.5 is set, where values greater than 0.5 are assigned the label 1, and values less than or equal to 0.5 are assigned the label 0.

This procedure of GBRT allows the model to iteratively refine its predictions, gradually improving its performance. By converting the predictions into binary labels, we can establish a clear threshold for classification, enhancing the interpretability and practicality of the model.

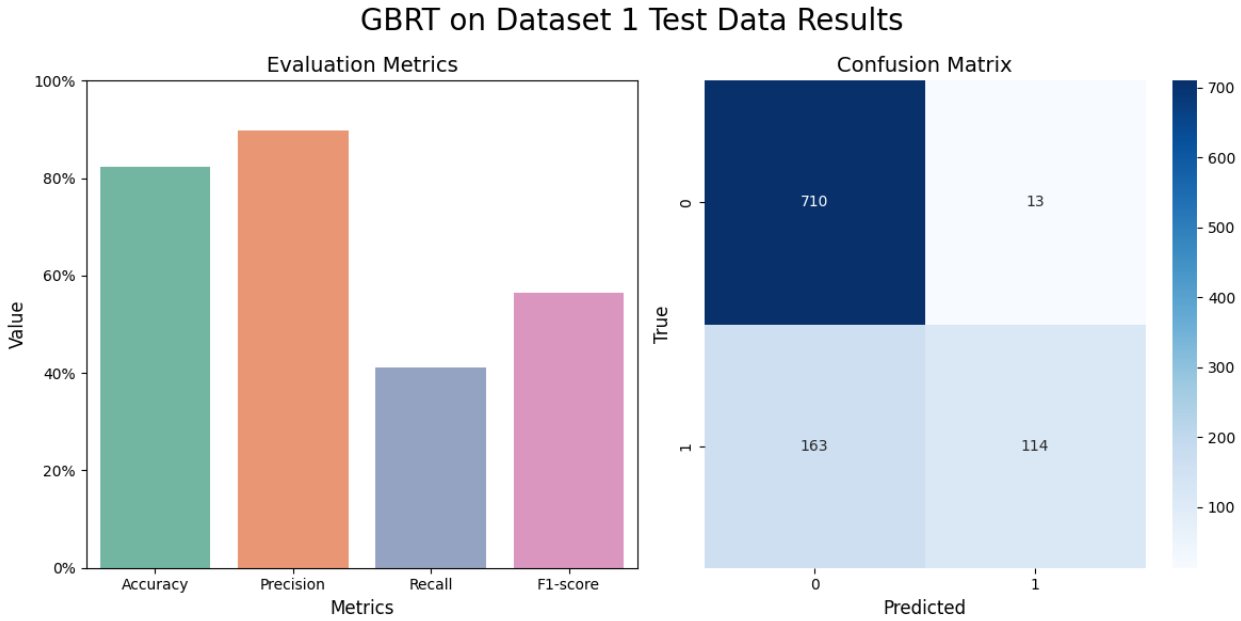
Base Classifier Analysis

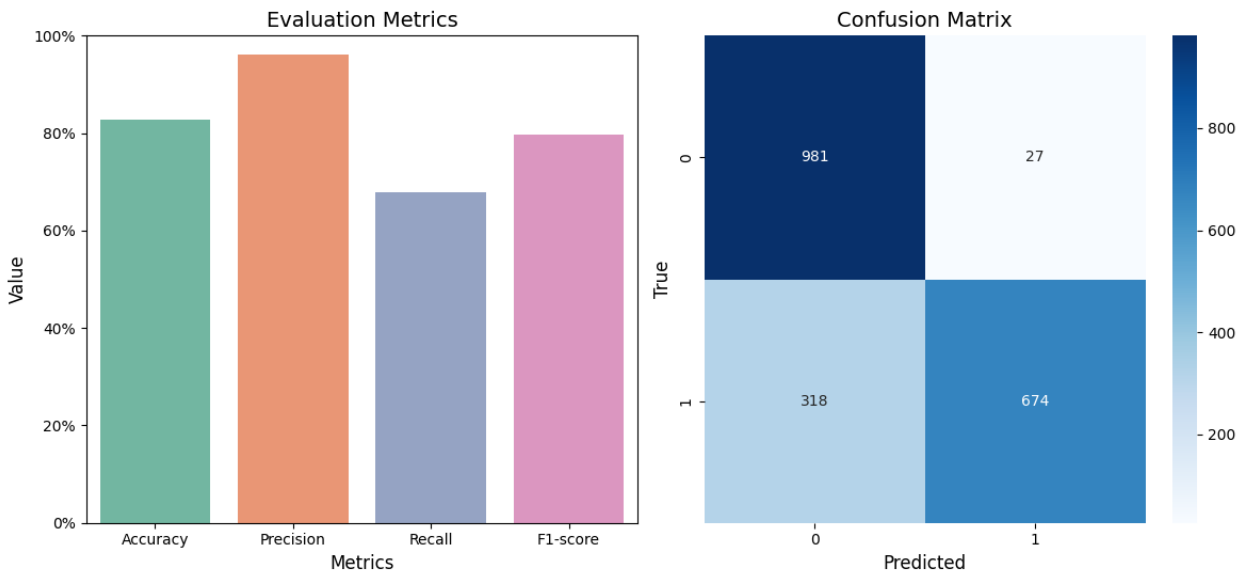
We initially ran a base version of our GBRT model on our datasets to set a baseline and measure performance. The hyperparameters of the model are as follows:

- Learning Rate: The learning rate determines the contribution of each individual tree in the ensemble. For our base classifier, the learning rate is set to 0.1.
- Number of Estimators: This refers to the number of decision trees to be built during the boosting process. In our model, we have 10 estimators.
- Maximum Depth: The maximum depth limits the number of levels in each decision tree. In our case, the maximum depth is set to 1, indicating that each tree will have a maximum of one split.
- Minimum Samples Leaf: This parameter specifies the minimum number of samples required to form a leaf node in the decision tree. We set a value of 1, so each leaf node contains one sample.

We found that the imbalanced nature of dataset 1, with a majority class of 75% of the data, may have influenced the performance metrics. The lower precision for class 0 suggests that the model had a higher tendency to misclassify instances of class 0 as class 1, indicating that the model may have been biased towards predicting the majority class. We also found that the higher recall for class 0 indicates a lower rate of false negatives, as the model performed well in identifying instances of class 0. However, the lower recall for class 1 suggests a higher rate of false negatives, indicating that the model struggled to correctly classify instances of class 1. In Dataset 2, which had a balanced distribution, the model performed better overall, as evidenced by higher precision, recall, and F1-scores for both classes. The balanced dataset allowed the model to learn from an equal representation of both classes, resulting in more balanced predictions. The precision for both class 0 and class 1 was relatively high, indicating a lower rate of false positives, and the recall for both classes was also high, indicating a lower rate of false negatives. The F1-scores for both classes

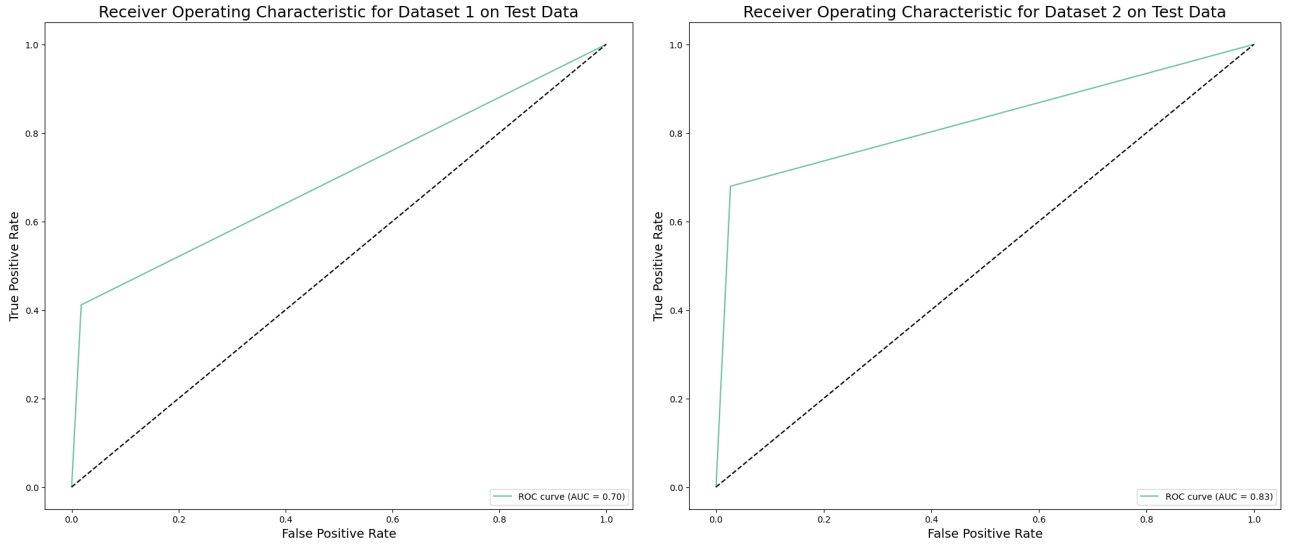
were relatively balanced and comparable. Surprisingly, the accuracy for both datasets 1 and 2 were comparable, at 82% and 83% respectively.





Dataset	Precision (Class 0)	Precision (Class 1)	F1-Score (Class 0)	F1-Score (Class 0)	Recall (Class 0)	Recall (Class 1)	Accuracy
Dataset 1 (Unbalanced)	.81	.90	.89	.56	.98	.41	.82
Dataset 2 (Balanced)	.76	.96	.85	.80	.97	.68	.83

In our analysis, we observed that the AUC for dataset 1 was 0.7, while for dataset 2, was 0.83. The AUC of 0.7 for dataset 1 suggests a moderate level of success in discriminating between positive and negative instances. For dataset 2, the AUC of 0.83 suggests a higher level of discrimination ability. The model, on average, performs better at ranking positive instances higher than negative instances compared to dataset 1. This indicates that the model's predictions are more reliable and better aligned with the true class labels in dataset 2, most likely due to the balanced distribution. A side by side comparison of dataset 1 and dataset 2 is found below:



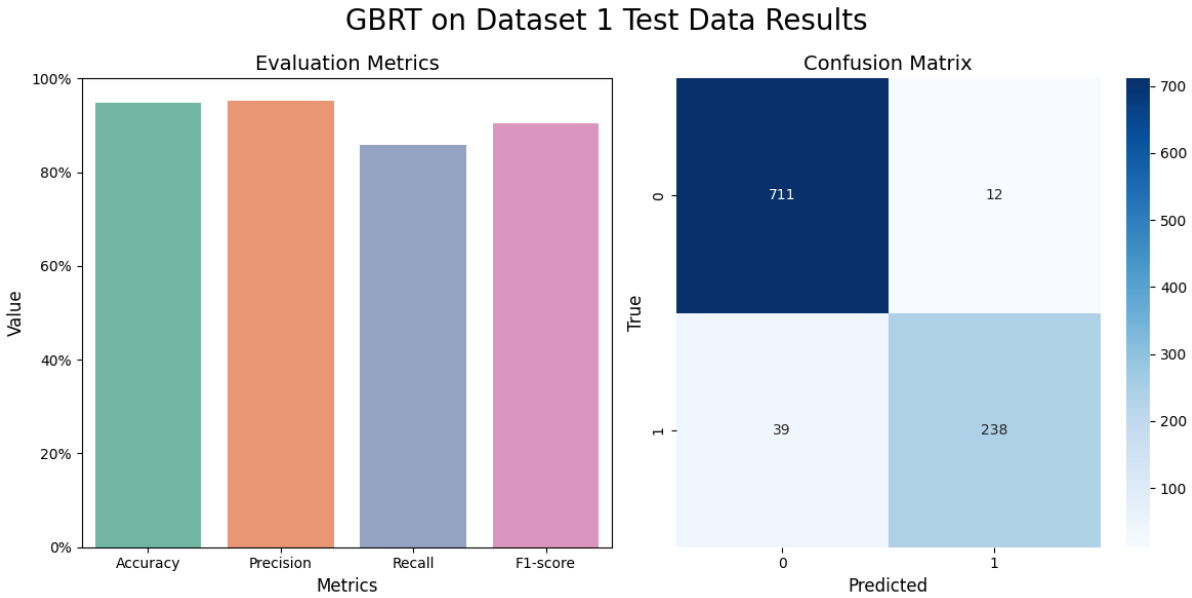
We further evaluated the MSE at every iteration and found that as the number of trees grew we saw an overall decrease in the MSE, in dataset 1 from .23 at the first iteration to .13 at the last, and in dataset 2 from .42 to .18. In dataset 1, the initial MSE was lower, indicating a simpler relationship between the features and the target variable. The boosting process allowed the model to improve its predictions and reduce the squared differences between the predicted and actual values. In dataset 2, with more redundant features and a larger feature space, the initial MSE was higher, suggesting a more complex relationship. However, the addition of 10 trees still led to a significant improvement in the model's performance. The difference in the magnitude of MSE reduction can be attributed to the varying complexity of the datasets. Overall, the boosting process demonstrated its effectiveness in enhancing the model's accuracy and reducing prediction errors.

Hyperparameter Tuning

We ran several experiments to test the performance of our model (this included varying one hyperparameter while keeping the other ones stable). We evaluated the following hyperparameters:

- The learning rate: 0.01, 0.1, 1.0
- The number of estimators: 10, 500, 1000
- The maximum depth of the tree: 1, 2, 5
- The minimum number of samples required to be at a leaf node: 1, 5, 10

Dataset 1 Findings



- Learning rate: In the case of imbalanced data, a lower learning rate (e.g., 0.01) can help prevent overfitting and improve the model's ability to capture patterns in the minority class by taking smaller steps towards the optimal solution. We observed that as our learning rate went up we had decreased accuracy even with larger estimators. We also found that increasing the maximum depth tends to improve accuracy, especially for lower learning rates (0.01). However, the accuracy did not improve consistently with increasing depth for all combinations of learning rate and estimators.
- Number of estimators: Increasing the number of estimators generally led to better accuracy, particularly for lower learning rates. Increasing the number of estimators can be beneficial for imbalanced data, as it allows the model to learn more complex decision boundaries and capture minority class instances better. At a learning rate of .001 and 10 estimators, we had a stable accuracy of 72.3% even when changing other parameters. With learning_rate=0.1 and max_depth=5, the accuracy improved between 10 and 500 estimators, indicating that a higher number of estimators allows the model to better capture the underlying patterns in the unbalanced data. At higher numbers, the interplay of other parameters impacted the accuracy, occasionally reducing the accuracy even as the estimators went up.

- Maximum depth: We found that the accuracy remained constant at 72.3% for a learning rate of 0.01, 10 estimators, and different max depths (1, 2, and 5), irrespective of the minimum samples per leaf. With a learning rate of 0.01 and 500 estimators, increasing the max depth from 1 to 2 improved accuracy significantly, from 83.3% to 91.7%. Further increasing the max depth to 5 results in improved accuracy of 95.8%. Similarly, with 1,000 estimators, increasing the max depth from 1 to 2 improves accuracy from 84.3% to 93.1%, and further increasing the max depth to 5 can yield an accuracy of 96.0% depending on other parameters.
- Minimum samples per leaf: We found that the impact of the minimum samples per leaf on accuracy was not consistent across different hyperparameter configurations. In some cases, increasing it led to a slight improvement in accuracy, but it did not hold true for all combinations.

After tuning the hyperparameters of the model on dataset 1, the best accuracy achieved was 96.2% with a learning rate of 0.1, 500 estimators, a maximum depth of 5, and 5 minimum samples per leaf. These parameter settings resulted in an area under the curve (AUC) value of .92.

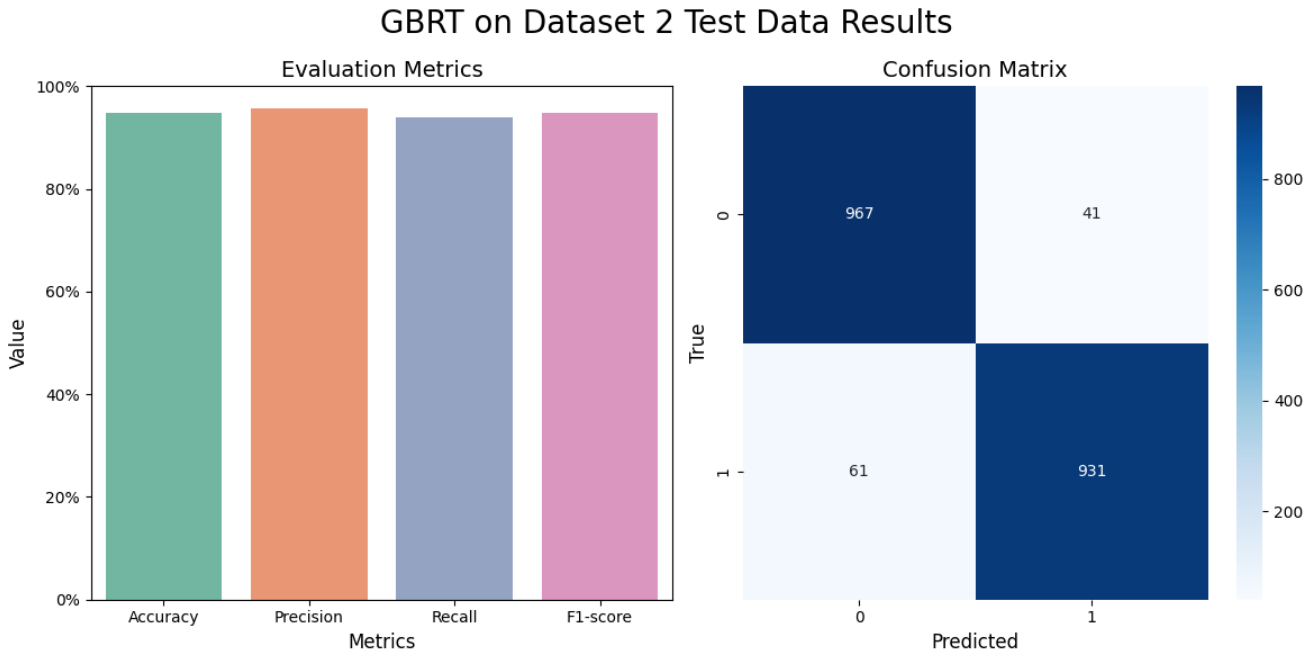
Dataset 2 Findings

- Learning rate: For dataset 2, we observed that a learning rate of 0.1 achieved the highest accuracy of 96.95%, while the learning rates of 0.01 and 1.0 achieved accuracies of 96.1% and 96.7%, respectively. In the context of a balanced dataset, where the number of instances in each class is roughly equal, the learning rate may not have a significant impact on the model's performance. This is because the learning rate determines the step size at which the model's parameters are updated during the training process. In a balanced dataset, the model is exposed to an equal number of instances from each class, allowing it to learn from both classes effectively. Since the classes are equally represented, the model's parameter updates are influenced by both classes, and a wide range of learning rates can yield reasonable performance.
- Number of estimators: Our experiments showed that increasing the number of estimators from 10 to 500 improves the accuracy significantly, indicating that a higher number of estimators allows the model to better capture the underlying patterns in the data. However, increasing the number of estimators from 500 to 1000 does not lead to a significant improvement in accuracy. Notably, some parameter combinations show continued accuracy improvement with more estimators, suggesting benefits beyond diminishing returns. This suggests that for balanced datasets, increasing estimators enhances accuracy until a saturation point. The optimal number will depend on the data used.
- Maximum depth: On our balanced dataset, with a learning rate of 0.01 and a low number of estimators (10), increasing the max depth from 1 to 5 did not result in any improvement in accuracy and it was constant at 50.4%. This suggests that increasing the max depth does not necessarily lead to better performance when the model is not sufficiently complex. When the learning rate is 0.01 and the number of estimators is increased to 500, we see a significant improvement in accuracy as the max depth increases. The accuracy increases from 92.25% to 95.9% as the max depth grows from 1 to 5. This indicates that increasing the max depth allows the model to capture more complex patterns in the data, resulting in improved accuracy. Similarly, when the learning rate is 0.01 and the number of estimators is further increased to 1000, we

observe a similar trend. The accuracy gradually increases from 93.85% to 96.1% as the max depth increases from 1 to 5. This suggests that increasing the max depth continues to improve accuracy, although the gains become smaller as the max depth reaches higher values. This is similar to the case of a learning rate of .1. However, when the learning rate is set to 1.0, we saw a there is a greater risk of overfitting. Overall, the results suggest that increasing the max depth can improve accuracy on a balanced dataset up to a certain point. However, there is a trade-off between complexity and generalization. Selecting an optimal max depth involves finding the right balance to avoid overfitting and achieve the best accuracy on unseen data. Cross-validation or using a separate validation set can help determine the optimal max depth value.

- Minimum samples per leaf: For a learning rate of 0.01, the accuracy remained constant at 50.4% across different values for all possible combinations. This suggests that the parameter has no significant effect on the accuracy in this scenario. For a learning rate of 0.01, as the number of estimators increased and the maximum depth increased, the accuracy improved. However, the accuracy remained constant within each combination despite the varying value of minimum samples. This indicates that the impact of the parameter is negligible when the learning rate is low. For a learning rate of 0.1 and 1.0, as the value increased from 1 to 10, we observed a slight decrease in accuracy.

By analyzing the results, we determined that the best combination of hyperparameters for dataset 2 was a learning rate of 0.1, 1000 estimators (trees), a maximum depth of 1, and a minimum number of samples at a leaf of 1. This configuration achieved the highest accuracy of our parameter tuning, attaining 96.95%. With these parameters we attained an AUC of .95.



Final Comparison

For Dataset 1, which is unbalanced, the best GBRT model achieved higher precision for both classes compared to the base GBT. The precision for class 0 increased from 0.81 to 0.95, indicating a significant improvement in correctly classifying instances of class 0. Similarly, the precision for class 1 increased from 0.90 to 0.95, indicating a better ability to correctly classify instances of class 1. The F1-scores for both classes also improved in the GBRT model. The F1-score for class 0 increased from 0.89 to 0.97, while the F1-score for class 1 increased from 0.56 to 0.90. The recall for class 0 remained high at 0.98 in both models, indicating a consistent ability to correctly identify instances of class 0. However, the recall for class 1 increased from 0.41 to 0.86 in the GBRT model, indicating a significant reduction in the rate of false negatives. Overall, the accuracy improved from 0.82 to 0.96 in the best GBRT model, indicating a substantial improvement in correctly classifying instances.

For dataset 2, which is balanced, the best GBRT model also achieved higher precision for both classes compared to the base model. The precision for class 0 increased from 0.76 to 0.94, indicating an improvement in correctly classifying instances of class 0. Similarly, the precision for class 1 increased from 0.96 to 0.96, indicating a consistent ability to correctly classify instances of class 1. The F1-scores for both classes remained relatively similar in both models, with the GBRT model achieving an F1-score of 0.95 for both class 0 and class 1. The recall for both classes also remained high in the GBRT model, with the recall for class 0 at 0.96 and the recall for class 1 at 0.94. This indicates a low rate of false negatives, with the GBRT model effectively identifying instances of both classes. The accuracy remained the same at 0.96 in both models, indicating a consistent overall performance in correctly classifying instances.

Dataset	Precision (Class 0)	Precision (Class 1)	F1-Score (Class 0)	F1-Score (Class 1)	Recall (Class 0)	Recall (Class 1)	Accuracy
Dataset 1 (Unbalanced)	.95	.95	.97	.90	.98	.86	.96
Dataset 2 (Balanced)	.94	.96	.95	.95	.96	.94	.96

AdaBoost

Adaptive Boosting (AdaBoost) is an ensemble learning algorithm that aims to improve classification accuracy by iteratively training a series of base learners. This algorithm adapts its training process by assigning higher weights to misclassified examples, allowing subsequent weak learners to focus on the challenging samples. The key steps involved in AdaBoost are as follows:

1. Initialize the weights for the training dataset
2. Iteratively train a series of base learners
    a. Each base learner is trained on a weighted version of the training data

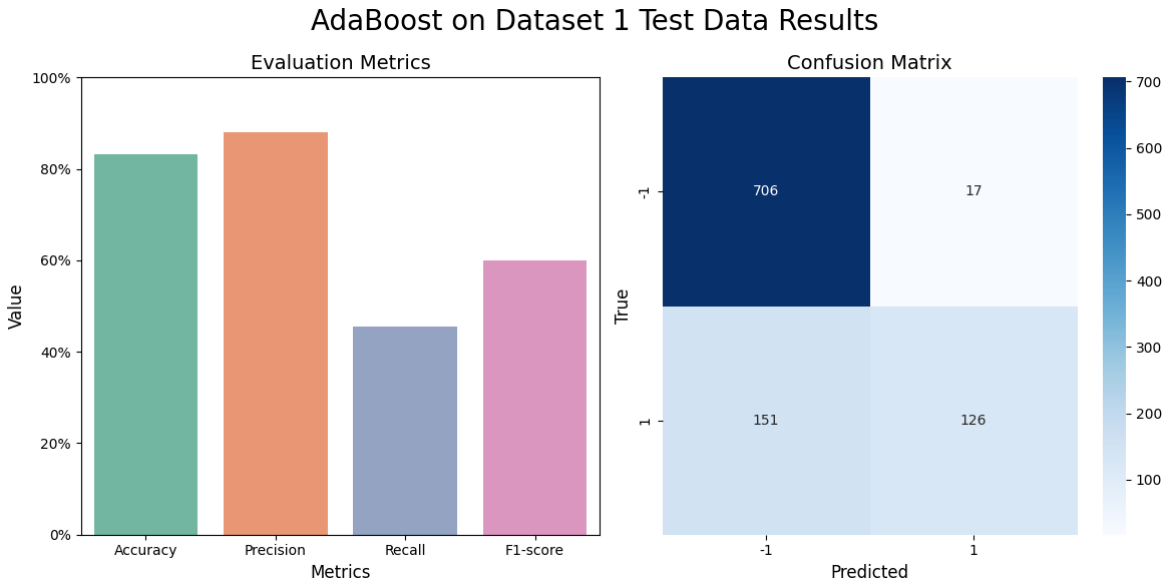
    i. In the first iteration, all training examples are assigned equal weights but in subsequent iterations, the weights are adjusted based on the performance of the previous base learners
3. Combine the individual predictions of the base learner to form a strong learner
4. Final prediction is made by aggregating the predictions, weighted based on the performance of each base learner during training

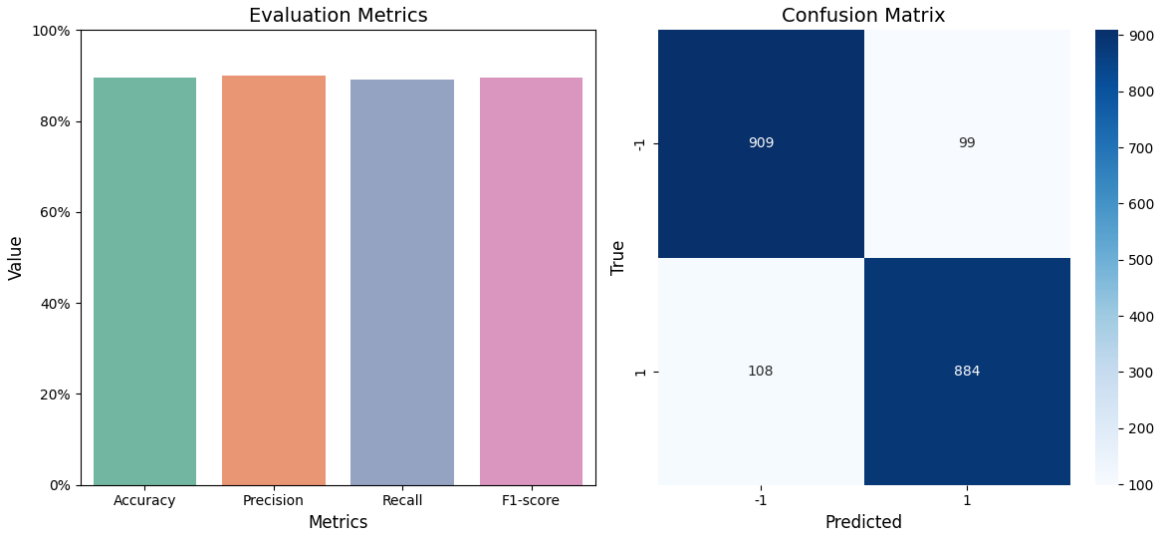
Base Classifier Analysis

We initially ran a base version of our AdaBoost model on our datasets to set a baseline and measure performance. The hyperparameters of the model are as follows:

- Learning Rate: The learning rate determines the contribution of each individual tree in the ensemble. For our base classifier, the learning rate is set to 1.0.
- Number of Estimators: This refers to the number of decision trees to be built during the boosting process. In our model, we have 5 estimators.

AdaBoost tends to be biased towards the majority class since misclassifying those instances would result in higher classification errors and, therefore, higher weights for the subsequent weak learners. As a result, the minority class may receive less attention during training, leading to lower accuracy and performance metrics for that class. In dataset 1, with an imbalanced class distribution, the model performed relatively well in terms of precision for both classes, with a precision of 0.82 for class -1 and 0.88 for class 1. The F1-scores also reflect a similar trend, with a value of 0.89 for class -1 and 0.60 for class 1. The recall for class -1 is quite high at 0.98, indicating that the model effectively identified most instances of class -1. However, the recall for class 1 is lower at 0.45, suggesting that the model struggled to correctly classify instances of class 1 (which is 25% of the labels). The overall accuracy for dataset 1 is 0.83, indicating a solid performance compared to the base GBRT model. When the data is balanced, AdaBoost can effectively learn from both classes and give appropriate consideration to each class during the training process. This balanced representation allows the algorithm to capture the patterns and characteristics of both classes, leading to more accurate predictions. In dataset 2, which has a balanced distribution of classes, the model achieved higher precision, F1-scores, and recall for both classes. The precision for both class -1 and class 1 is relatively high, with values of 0.89 and 0.90, respectively. The F1-scores for both classes are also balanced and comparable, with a value of 0.90. The recall for both classes is high, with a value of 0.90 for class -1 and 0.89 for class 1. The accuracy for Dataset 2 is 0.90, indicating that the model correctly predicted 90% of the instances.





Dataset	Precision (Class -1)	Precision (Class 1)	F1-Score (Class -1)	F1-Score (Class 1)	Recall (Class -1)	Recall (Class 1)	Accuracy
Dataset 1 (Unbalanced)	.82	.88	.89	.60	.98	.45	.83
Dataset 2 (Balanced)	.89	.90	.90	.90	.90	.89	.90

We found that the AUC value for dataset 1 with AdaBoost was 0.72, indicating moderate discrimination ability due to the imbalanced class distribution. In contrast, dataset 2, which is balanced, achieved an AUC value of 0.9, representing high discrimination ability. This demonstrates that the imbalanced nature of

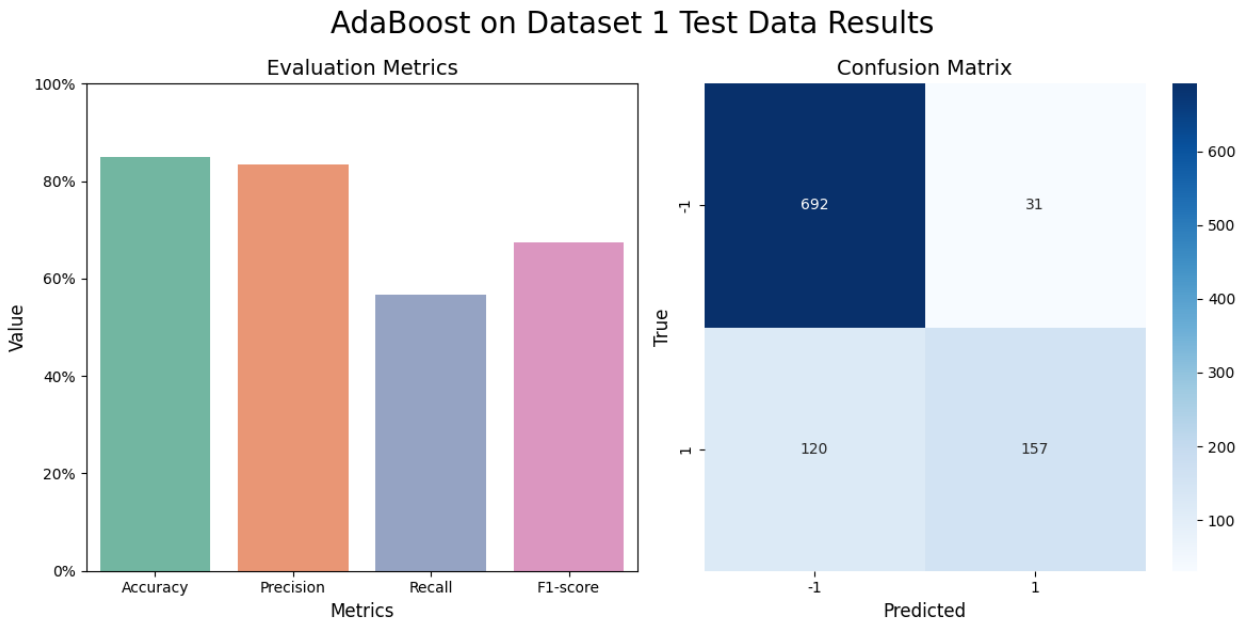
dataset 1 limits the model's performance with the base classifier, while the balanced dataset allows for improved classification accuracy in dataset 2 even without tuning.

Hyperparameter Tuning

We ran several experiments to test the performance of our model (this included varying one hyperparameter while keeping the other ones stable). We evaluated the following hyperparameters:

- The learning rate: 0.01, 0.1, 1.0
- The number of estimators: 5, 50, 100

Dataset 1 Findings



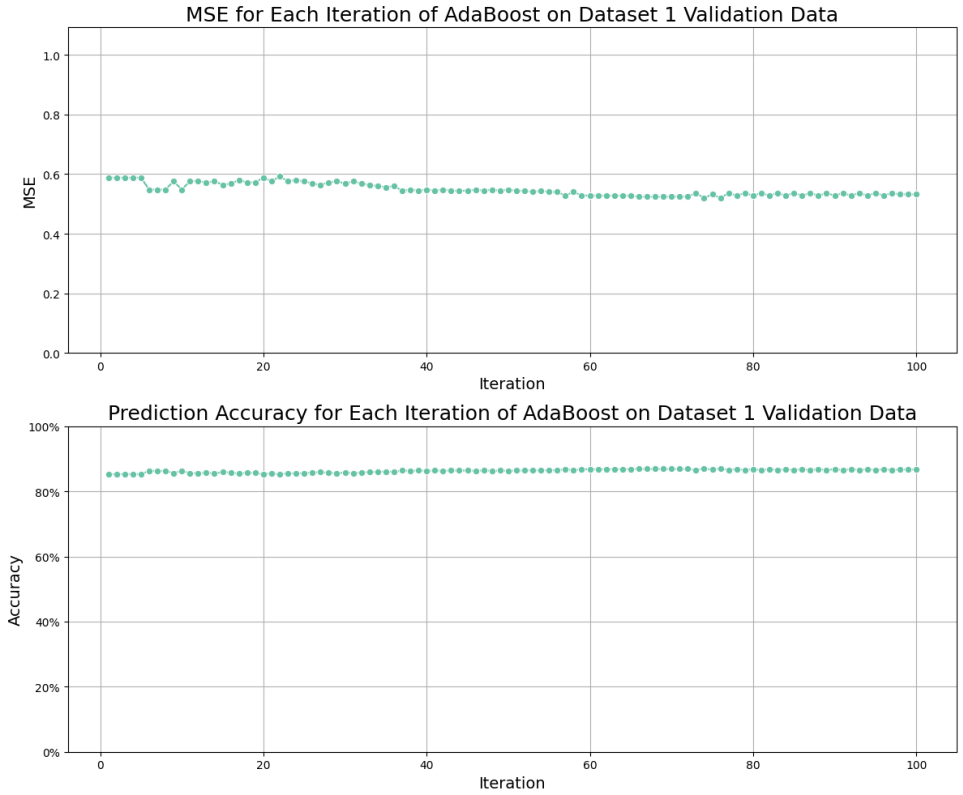
AdaBoost is designed to address the challenges posed by imbalanced datasets by assigning higher weights to misclassified instances, thereby allowing subsequent weak learners to focus more on the challenging samples. This adaptive training process enables the model to pay more attention to the minority class, improving its ability to correctly classify instances from both classes. Therefore, we expected that we could reach a higher accuracy with fewer estimators with AdaBoost than GBRT, but we found that with fewer estimators in GBRT but other parameters we could attain higher accuracy (accuracy is not necessarily the best measure though). The tradeoff between AdaBoost and GBRT in this case however is that AdaBoost takes longer to run in our implementation.

- Learning rate: In the case of imbalanced data, the learning rate may have a limited impact on the model's performance due to the inherent bias towards the majority class. The learning rate determines the contribution of each individual weak learner (decision tree) in the ensemble, and it affects how much weight is assigned to each weak learner's prediction during the training process. Despite changing the learning rate across different values (0.01, 0.1, and 1.0), the accuracy remained the same (0.832) for different combinations of learning rate and estimators (for lower estimators in particular), or at just a low learning rate of .01.

- Number of estimators: We observed that increasing the number of estimators from 5 to 100 leads to an improvement in accuracy. Specifically, for estimators=100, the accuracy reaches its highest value of 0.849. This indicates that the model benefits from having a larger number of weak learners to boost its performance.

For the unbalanced dataset, the AdaBoost model with 100 estimators and a learning rate of 1.0 achieved an accuracy of 0.85. It showed better performance in predicting the majority class (-1) with precision of 0.85 and recall of 0.96. However, it struggled to accurately classify instances of the minority class (1), as indicated by lower precision (0.84) and recall (0.57). The overall F1-scores were 0.90 for class -1 and 0.68 for class 1. These results highlight the model's bias towards the majority class and its difficulty in capturing instances of the minority class, however this is still an improvement over the base model.

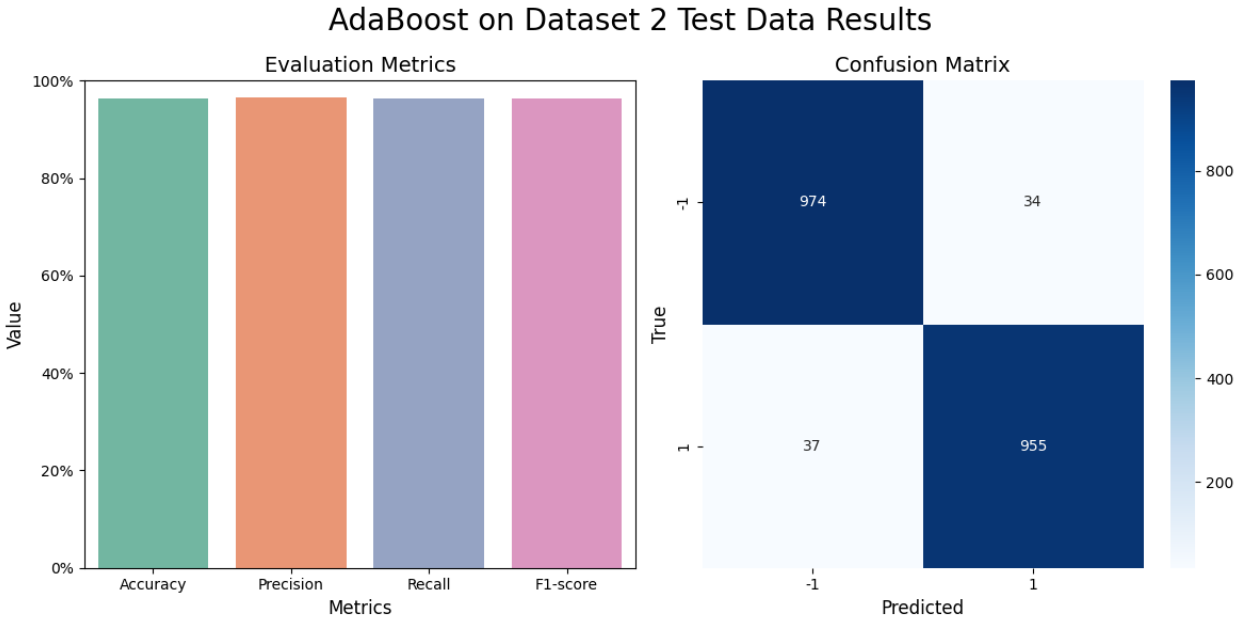
We found that we attained a highest accuracy of 84.9% with 100 estimators and a learning rate of 1.0. With this model we had an AUC of .76 which is slightly improved from our base classifier, but shows that the model struggles to differentiate. We also observed on the unbalanced dataset that as we increased the number of estimators in the Adaboost model, the Mean Squared Error (MSE) remained relatively constant at around 0.6, dropping only to a lowest point of around .55, and the accuracy plateaued around 0.85 without significant improvement. We attribute this observation to the imbalanced class distribution within the dataset. With a majority class heavily outweighing the minority class, Adaboost tends to prioritize the majority class during training, leading to biased predictions and difficulty capturing patterns in the minority class. Consequently, the model struggled to improve its performance in terms of accuracy and MSE.



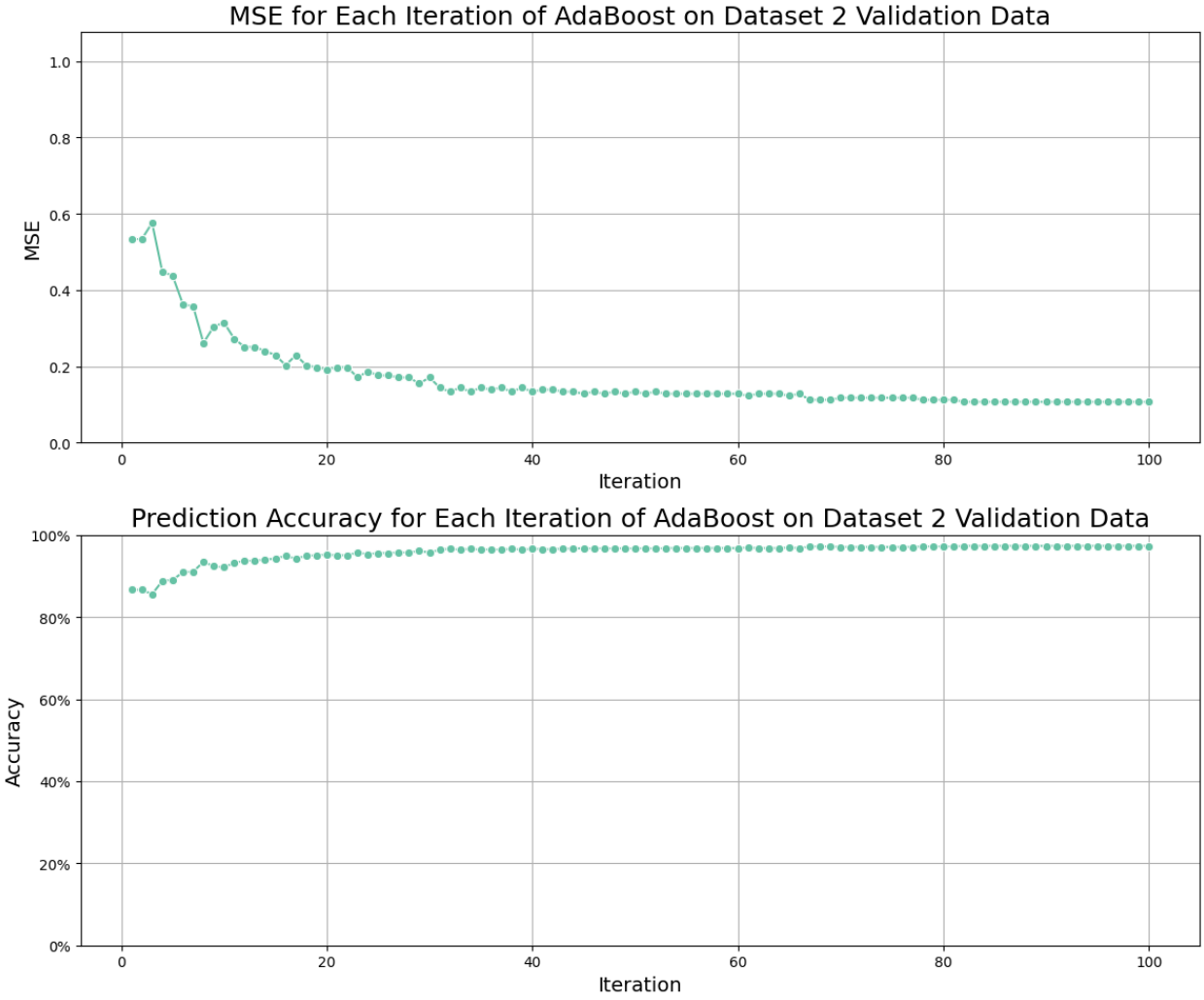
Dataset 2 Findings

- Learning rate: We observed that as the learning rate increased, the accuracy generally improved. For example, with 5 estimators, the accuracy increased from 0.8495 (learning_rate=0.01) to 0.85 (learning_rate=0.1) and further to 0.8965 (learning_rate=1.0). Similarly, with 50 and 100 estimators, increasing the learning rate from 0.01 to 1.0 resulted in higher accuracy. A higher learning rate allows the model to learn faster from the training data but can also increase the risk of overfitting.
- Number of estimators: Increasing the number of estimators generally led to better accuracy. With a learning rate of 1.0, the accuracy increased from 0.887 (estimators=50) to 0.9645 (estimators=100). This suggests that increasing the number of estimators allows the model to capture more complex patterns and improve its predictive performance. However, it's worth noting that the improvement in accuracy may diminish or plateau after a certain number of estimators, but due to computational limitations we did not find where this threshold is for our data.

Overall, the best model achieved an accuracy of 96.45% with 100 estimators and a learning rate of 1.0. This model demonstrated high precision, recall, and F1-scores for both classes, indicating balanced performance. The accuracy of 0.9645 indicates that the model correctly predicted the majority and minority classes with a high level of accuracy. Additionally, the best model achieved an area under the curve (AUC) value of 96%, suggesting that the model performed well in classifying instances from both the majority and minority classes, further confirming the effectiveness of the chosen hyperparameters. Furthermore, the precision, recall, and F1-score values were consistently high for both classes, indicating a lower rate of false positives and false negatives. This suggests that the model achieved a good balance between correctly classifying instances from both the majority and minority classes and significantly improved from our base model.



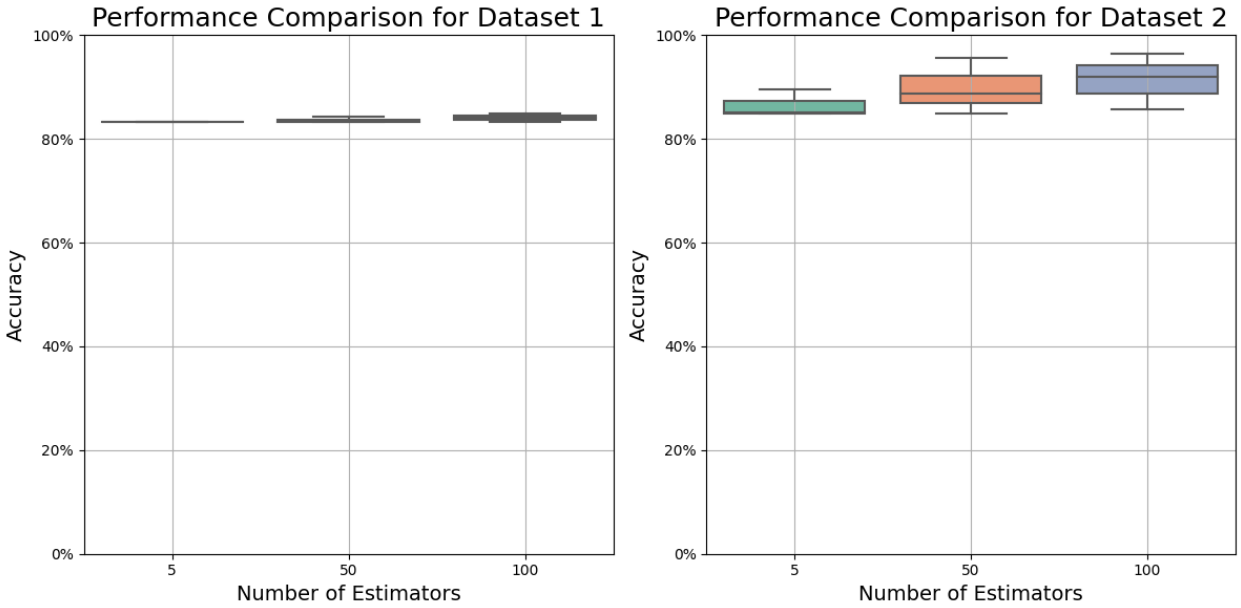
For the balanced data in dataset 2, we observed that as the number of estimators increased in our best model, the mean squared error (MSE) decreased from approximately 0.6 to 0.1. This indicates that the model improved in its ability to fit the data and make more accurate predictions. The accuracy of the model also increased from around 0.84 to around 0.95 as the number of estimators increased, reaching a plateau around 25 estimators. With each additional estimator, the model focused on correcting the mistakes made by previous weak learners and placed more emphasis on instances that were misclassified. As a result, the subsequent weak learners better captured the patterns, leading to improved accuracy.



The plateau observed around 25 estimators suggests that the model may have reached a point of diminishing returns, where further increasing the number of estimators did not significantly improve the model's performance, and this is indeed a very similar final accuracy to our model with a learning rate of 1.0 and 50 estimators (it attained over 95% accuracy). This could be due to the dataset being relatively simple. Adding more estimators beyond 100 may introduce more complexity to the model without providing substantial gains in accuracy.

Final Comparison

We saw that increasing the number of estimators on our unbalanced dataset did not lead to improved accuracy, and that the range of accuracies across the different learning rates was very small. This suggests that, in the context of imbalanced datasets, increasing the number of estimators may not necessarily lead to substantial improvements in the model's performance. This could be due to the inherent bias of AdaBoost towards the majority class in such datasets, as misclassifying those instances results in higher weights for subsequent weak learners. As a result, the model may struggle to effectively learn and capture the patterns in the minority class, leading to limited improvements in accuracy despite increasing the number of estimators. On the other hand, for the balanced data, it is notable that the range of accuracies at each estimator grew significantly with larger estimators. This suggests that increasing the number of estimators allows the model to explore more complex decision boundaries and capture a greater variety of patterns in the data. As a result, the model's accuracy improves as it becomes more capable of accurately classifying instances from both classes. This observation aligns with the principle of AdaBoost, where the subsequent weak learners focus on correcting the mistakes made by previous learners and can lead to improved overall accuracy with increasing estimators.



Overall we found that for dataset 1, which is unbalanced, the precision for both classes improved in the best model compared to the base model. The precision for class -1 increased from 0.82 to 0.85, indicating a higher percentage of correctly classified instances of class -1. Similarly, the precision for class 1 increased from 0.88 to 0.84, indicating a better ability to correctly classify instances of class 1. The F1-scores for both classes also improved in the best model, with the F1-score for class -1 increasing from 0.89 to 0.96 and the F1-score for class 1 increasing from 0.60 to 0.57. The recall for class -1 remained the same at 0.96, indicating a consistent ability to correctly identify instances of class -1. However, the recall for class 1 decreased from 0.45 to 0.57, suggesting a higher rate of false negatives in the best model. Overall, the accuracy improved from 0.83 to 0.85 in the best model, indicating a better overall performance in correctly classifying instances. For Dataset 2, which is balanced, the precision for both classes also improved in the

best model. The precision for class -1 increased from 0.89 to 0.96, indicating a higher accuracy in classifying instances of class -1. Similarly, the precision for class 1 increased from 0.90 to 0.97, indicating an improved ability to correctly classify instances of class 1. The F1-scores for both classes remained the same at 0.96, indicating consistent performance in capturing the trade-off between precision and recall. The recall for both classes remained high in the best model, with the recall for class -1 and class 1 both at 0.97. This indicates a low rate of false negatives, with the model being effective in identifying instances of both classes. The accuracy improved to 96, indicating a much better model overall.

Dataset	Precision (Class -1)	Precision (Class 1)	F1-Score (Class -1)	F1-Score (Class 1)	Recall (Class -1)	Recall (Class 1)	Accuracy
Dataset 1 (Unbalanced)	.85	.84	.96	.57	.96	.57	.85
Dataset 2 (Balanced)	.96	.97	.96	.96	.97	.96	.96

Discussion

In this study, we aimed to investigate the performance of Gradient Boosted Regression Trees (GBRT) and AdaBoost algorithms on two synthetic datasets with distinct characteristics. Dataset 1 was characterized by class imbalance, with 75% of samples belonging to one class and 25% belonging to the other class. Dataset 2 had a balanced distribution of classes and included redundant features, potentially introducing noise to the classification task.

Our analysis revealed that dataset characteristics significantly influenced the performance of the boosting techniques. Our study highlights the importance of understanding dataset characteristics and selecting appropriate boosting techniques and hyperparameters in order to achieve optimal performance in classification tasks. Overall, our results highlight the influence of databases, such as class imbalance, informative features, and feature redundancy, on the performance of boosting algorithms. Class imbalance and the presence of noisy features can significantly impact model performance. Dataset imbalance posed challenges for both GBRT and AdaBoost, leading to biased predictions and lower accuracy, precision, and recall for the minority class. On the other hand, by having a balanced class distribution, GBRT and AdaBoost were able to learn from an equitable representation of both classes. This led to predictions that were more evenly balanced.

To achieve optimal results, practitioners should assess all previously mentioned dataset considerations. By addressing these considerations, it is possible to enhance the robustness of boosting algorithms in various real world applications.