Data Intensive Computing – CSE 587A

Project Phase – 2

Problem Statement: To develop a machine learning model that can classify credit scores using information gathered from banks and credit-related datasets.

Explanation and Analysis

Model Applications

Random Forest Classifier:

The Random Forest Classifier is favoured in areas like credit risk analysis due to its ability to handle complex, high-dimensional datasets common in finance, capturing non-linear relationships between variables without succumbing to overfitting. Its robustness against missing values and outliers ensures reliability, while the model's capacity to rank feature importance offers valuable insights for decision-making. Additionally, its versatility for both classification and regression tasks, combined with ease of implementation, makes it a practical and insightful choice for financial modelling, where accuracy and interpretability are paramount.

Model Tuning/Training:

Model tuning and training in machine learning involve iteratively adjusting and optimizing the model's parameters to improve its performance on a given task. During training, the model learns to predict outcomes from the input data by minimizing a loss function, which quantifies the difference between the model's predictions and the actual data. Tuning, on the other hand, focuses on refining the model's hyperparameters, such as the learning rate, the number of trees in a Random Forest, or the depth of these trees, to find the optimal configuration that yields the best performance, typically measured by accuracy, precision, recall, or F1 score. This process often involves techniques like cross-validation to ensure the model's generalizability and prevent overfitting. The goal is to achieve a well-trained model that not only performs well on the training data but also generalizes effectively to new, unseen data, ensuring its practical applicability in real-world scenarios.

Effectiveness of Algorithm:

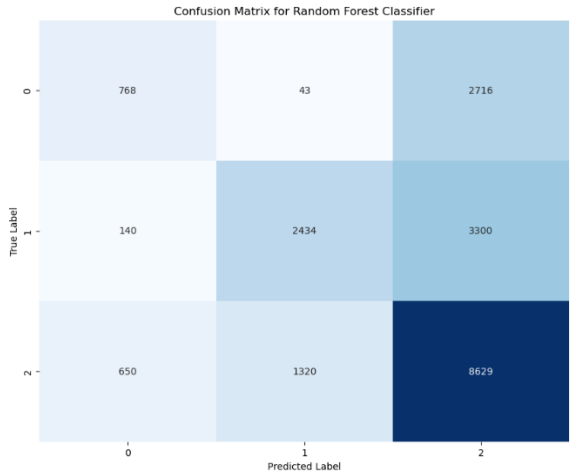
Based on the provided metrics, the effectiveness of the algorithm can be summarized as follows:

- **Accuracy (0.77545):** The algorithm achieves an accuracy of approximately 77.5%, indicating that it correctly predicts the class for about 77.5% of the instances in the test set. This level of accuracy is decent and suggests that the model is relatively effective for the given task, especially if the dataset is balanced across classes.
- **Precision**: The precision scores for the classes are 0.73 (Class 0), 0.78 (Class 1), and 0.78 (Class 2). These values indicate that the model is fairly reliable in its positive

predictions across all classes, with Class 1 and Class 2 having higher precision than Class 0. High precision for Class 1 and Class 2 suggests that when the model predicts an instance to be in these classes, it is correct around 78% of the time.

- **Recall:** The recall scores are 0.66 (Class 0), 0.79 (Class 1), and 0.81 (Class 2). These scores show the model's capability to identify all relevant instances for each class. The model is particularly effective for Class 2 with a recall of 0.81, indicating it captures most of the true Class 2 instances. However, for Class 0, the recall is relatively lower at 0.66, suggesting the model misses a significant proportion of true Class 0 instances.
- **F1-Score:** The F1-scores, which balance precision and recall, are 0.69 (Class 0), 0.78 (Class 1), and 0.80 (Class 2). These scores reflect a good balance for Classes 1 and 2 but indicate room for improvement in Class 0, where the lower recall negatively impacts the F1-score.

Confusion Matrix:



Here's an overview of the intelligence we can gain from it:

- Class 0 Predictions: There are 768 true positives where the model correctly predicted Class 0, but there are also 2716 false negatives where Class 0 was not identified. This suggests a tendency of the model to under-predict Class 0.
- Class 1 Predictions: The model has 2434 true positives for Class 1, but there's a significant number of false negatives (3300) as well, indicating that the model often confuses Class 1 with Class 2.
- Class 2 Predictions: Class 2 has the highest number of true positives (8629), with relatively fewer false negatives (1320), showing that the model is most effective at identifying Class 2.
- False Positives: For Classes 0 and 1, there are relatively few false positives (43 for Class 0 and 650 for Class 1), indicating the model is cautious about predicting these classes. However, there are a substantial number of instances where Classes 0 and 1 are incorrectly predicted as Class 2.
- Overall Insights: The model shows a strong ability to identify Class 2 but struggles with Class 0 and Class 1, confusing these with Class 2. The low recall for Class 0 and Class 1

(as seen in the provided metrics) corresponds with the high false negatives for these classes in the confusion matrix.

Decision Tree Classifier:

Justification for Decision Tree Classifier:

The Decision Tree Classifier in the context of credit risk analysis provides a transparent, interpretable model that's easy to visualize and understand, making it well-suited for regulatory environments where decisions need to be explainable. Its inherent capability to handle non-linear relationships and mixed data types (numerical and categorical) without the prerequisite of feature scaling is a significant advantage. Furthermore, Decision Trees naturally perform feature selection, highlighting the most predictive factors for credit risk, which is invaluable for identifying key risk indicators. However, while Decision Trees can be a good starting point for modeling due to their simplicity, they may be prone to overfitting if not carefully pruned, and this characteristic would be a key consideration in their use and would typically inform the approach to their tuning and evaluation.

Model Tuning/Training:

Model tuning and training encompass the process of refining a machine learning algorithm to optimize its performance. During training, the model learns to map the relationship between input features and the target variable, iteratively adjusting its internal parameters to minimize prediction errors, as quantified by a chosen loss function. Tuning, which often follows training, involves systematic experimentation with the model's hyperparameters—those settings not learned from the data—to discover the combination that yields the best results, typically evaluated using metrics like accuracy, precision, recall, and the F1 score. Methods such as cross-validation are employed to ensure that the model not only fits the training data well but also generalizes effectively to unseen data, thereby enhancing its real-world applicability. The ultimate aim of training and tuning is to derive a model that strikes an optimal balance between bias and variance, learns the data's underlying structure, and performs consistently and accurately on new data.

Effectiveness of Algorithm:

The performance metrics for the Decision Tree Model are as follows:

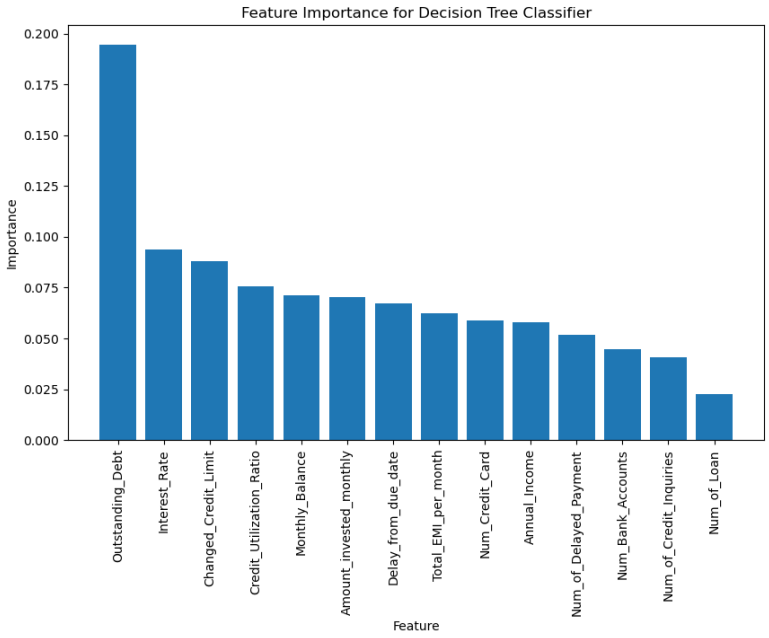
- Accuracy (0.69705): The model has an accuracy of about 69.7%, which is a moderate level of correctness across all classes. It indicates that roughly 7 out of 10 predictions made by the model are correct.
- Precision: The model's precision is 0.60 for Class 0, 0.70 for Class 1, and 0.73 for Class 2. These figures suggest that the model is relatively more precise in predicting Class 2 and least precise for Class 0. In other words, when the model predicts Class 2, it is correct about 73% of the time, but this drops to 60% for Class 0.
- Recall: The recall is 0.62 for Class 0, 0.69 for Class 1, and 0.72 for Class 2. The model is best at correctly identifying instances of Class 2 and worse at identifying Class 0. This

means it captures most of the true Class 2 instances but misses more true instances of Class 0.

- F1-Score: The F1-scores are balanced across all classes: 0.61 for Class 0, 0.70 for Class 1, and 0.73 for Class 2. These scores suggest that the balance between precision and recall is relatively consistent across classes, though there is room for improvement in Class 0.

Overall, the Decision Tree Model shows a solid baseline performance, with the effectiveness being the highest for Class 2, both in terms of precision and recall. However, compared to the Random Forest Classifier from the previous data you provided, it appears to have lower accuracy and F1-scores for all classes, suggesting that the ensemble approach of the Random Forest could be providing a more robust prediction by aggregating multiple decision trees to improve the overall prediction quality.

Feature Importance:



This visualization offers insights into which features the model found most influential in predicting the target variable. The most important feature is 'Outstanding_Debt', followed by 'Interest_Rate', 'Credit_Limit', 'Credit_Utilization_Ratio', 'Monthly_Balance', 'Amount_Invested_Monthly', 'Delay_from_due_date', 'Total_EMI_per_month', 'Num_Credit_Card', 'Annual_Income', 'Num_of_Delayed_Payment', 'Num_Bank_Accounts', 'Num_of_Credit_Inquiries', and 'Num_of_Loan'.

This intelligence is critical for understanding the driving factors behind the model's predictions. For example, the high importance of 'Outstanding_Debt' suggests that this feature significantly influences the likelihood of a credit event, such as a default, which is key information for credit risk analysis. Features with lower importance might have a lesser impact on the model's decision-making process or may be providing redundant information that is already captured by more important features. This insight into feature importance can

guide further data collection, feature engineering, and the refinement of the model to improve performance and interpretability.

KNN Classifier

Justification for KNN:

K-Nearest Neighbors (KNN) has been selected for its simplicity, effectiveness in capturing complex patterns in data without making assumptions about the underlying distribution, and its ability to provide probability estimates for predictions. KNN is well-suited for applications where the decision boundary is not linear and can be particularly effective when the data exhibits natural clustering. It also adapts easily to new data, making it versatile for dynamic environments. However, its reliance on feature space and sensitivity to irrelevant features likely necessitated careful preprocessing and feature selection as indicated in the notebook's earlier steps. The choice of KNN suggests a need for a model that is both interpretable and flexible, traits that are particularly valued in domains like credit risk analysis.

Model Tuning/Training:

Model tuning and training are critical stages in the development of a machine learning model, focusing on improving and validating the model's ability to make accurate predictions. During the training phase, the model learns from a dataset by adjusting its internal parameters to reduce the difference between its predictions and the actual outcomes, often using algorithms like gradient descent. Tuning, or hyperparameter optimization, follows, where the model's hyperparameters—settings not learned from the data—are systematically varied and the model's performance is evaluated to find the best set. Techniques like grid search, random search, or more sophisticated methods such as Bayesian optimization are employed. Validation strategies such as k-fold cross-validation ensure the model's generalizability to unseen data. Together, these processes aim to build a robust model that performs well not just on the training data but also holds its predictive power in real-world situations, balancing bias and variance to avoid overfitting and underfitting.

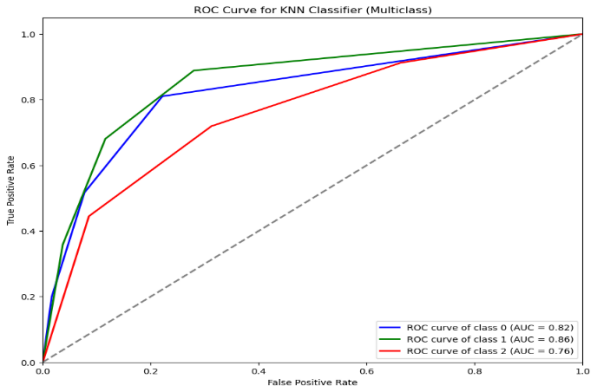
Effectiveness of Algorithm:

The performance of the K-Nearest Neighbors (KNN) model can be assessed as follows:

- Accuracy (0.6811): The model has an accuracy of about 68.1%, indicating that it correctly predicts the class label for roughly 68 out of every 100 instances. While this suggests moderate predictive power, it is somewhat lower than the accuracy of the Decision Tree and Random Forest models previously discussed.
- Precision: The precision for the three classes are 0.53 for Class 0, 0.71 for Class 1, and 0.72 for Class 2. This means the model is less precise when predicting Class 0, as compared to the other two classes.
- Recall: The recall scores are 0.57 for Class 0, 0.68 for Class 1, and 0.72 for Class 2. The model is more reliable at identifying true instances of Class 2, while it struggles slightly more with Class 0.

- F1-Score: The F1-scores, which are the harmonic mean of precision and recall, are 0.55 for Class 0, 0.69 for Class 1, and 0.72 for Class 2, showing a balanced performance between precision and recall for each class.

ROC Curve for KNN:



The image shows a Receiver Operating Characteristic (ROC) curve for a K-Nearest Neighbors (KNN) classifier, which is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The plot depicts three ROC curves, each representing one of the classes in a multiclass classification task, with their respective areas under the curve (AUC) scores:

- Class 0: The ROC curve for Class 0 is the blue line with an AUC of 0.82. This is relatively close to 1, indicating a good measure of separability for Class 0, meaning the model does well at distinguishing between Class 0 and not Class 0.
- Class 1: The ROC curve for Class 1 is shown in green with an AUC of 0.86. This is the highest AUC value among the three classes, suggesting that the model is most effective at distinguishing Class 1 from the other classes.
- Class 2: The ROC curve for Class 2 is the red line with an AUC of 0.76. While this is the lowest AUC score of the three, it still indicates a fair level of separability.

Overall, the curves show that the KNN classifier has good classification ability across all classes. The AUC values suggest that the classifier is quite effective, with Class 1 being the most distinguishable. However, the classifier might benefit from further optimization, as there is room for improvement, especially for Class 2, which has a lower AUC score compared to the others. The fact that none of the curves approaches the top-left corner of the graph also indicates that there may be potential to enhance the true positive rate and reduce the false positive rate through model tuning or additional feature engineering.

Gradient Boosting Classifier

Justification for Gradient Boosting Classifier

Due to its strong performance in a wide range of applications and its ability to handle various types of predictive modelling tasks effectively. Gradient Boosting is a powerful ensemble technique that builds a model in stages, allowing it to optimize for accuracy and handle complex datasets with potentially non-linear relationships. It often performs well with

unbalanced data, typical in risk assessment scenarios, by focusing on difficult-to-classify instances and improving on them iteratively. Moreover, Gradient Boosting provides a way to assess feature importance, which can offer valuable insights, especially in credit risk contexts where understanding the influence of different factors is crucial. It's also versatile and flexible, allowing for extensive hyperparameter tuning to adapt the model to the specific characteristics of the dataset, such as adjusting the number of boosting stages, learning rate, and depth of individual trees. These factors, combined with its predictive power, likely contributed to the decision to employ Gradient Boosting in the analysis workflow outlined in the notebook.

Model Tuning/Training:

Training a Gradient Boosting Classifier involves incrementally building an ensemble of weak learners, typically decision trees, to form a strong predictive model. The tuning process requires careful adjustment of hyperparameters such as the number of trees, tree depth, learning rate, and subsample size to optimize performance. Techniques like grid search and cross-validation are crucial in finding the right combination that minimizes overfitting and maximizes the model's ability to generalize from the training data to unseen data. The aim is to achieve a finely-tuned model that delivers high predictive accuracy and robustness in practical scenarios, striking a balance between capturing complex data patterns and maintaining simplicity to avoid overfitting.

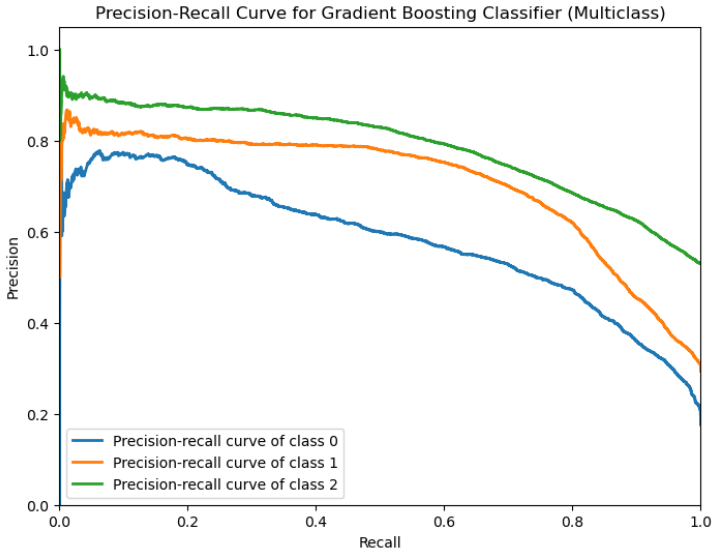
Effectiveness of Algorithm:

The image depicts a Precision-Recall Curve for a Gradient Boosting Classifier applied to a multiclass classification task. Each curve corresponds to one of the three classes and illustrates the trade-off between precision (the y-axis) and recall (the x-axis) for different threshold settings:

- Class 0 (Blue Curve): This curve starts with high precision but rapidly decreases as recall increases. The steep decline suggests that achieving high recall significantly reduces precision for Class 0, indicating that more false positives are accepted as the model tries to capture more true positives.
- Class 1 (Green Curve): The precision for Class 1 is lower than Class 0 at low recall levels but decreases at a slower rate as recall increases. This implies a more balanced trade-off between precision and recall for Class 1, where increasing the number of true positives doesn't lead to as rapid an increase in false positives.
- Class 2 (Orange Curve): Class 2 shows a higher precision than Class 1 at almost all levels of recall, which declines more gradually, indicating a more stable performance even as the recall increases. This suggests that the model is better at identifying true positives for Class 2 without incurring many false positives.

Overall, the Precision-Recall Curves illustrate the performance of the classifier for each class at different thresholds. The area under each curve can give an indication of the overall effectiveness of the model for that class—the larger the area, the better the combined precision and recall performance. The curves show that the model performs best on Class 2, with Class 0 being the most challenging to classify correctly. These insights can guide efforts

to improve the classifier, perhaps by focusing on feature engineering, class re-balancing, or adjusting the decision threshold specifically for Class 0 where the precision-recall trade-off is most severe



Logistic Regression

Justification for Logistic Regression:

Logistic Regression, this model is generally favored for its simplicity, efficiency, and interpretability, especially in binary classification problems. Its extension to multiclass problems, often through techniques like the one-vs-rest (OvR) approach, makes it versatile for broader applications. Logistic Regression works well when the relationship between the input features and the probability of the output class is approximately linear. It is also beneficial when the importance and impact of individual features on the prediction need to be clearly understood and explained, which is often required in credit risk assessment and other financial applications. The model's coefficients can be directly associated with the odds ratios for the respective features, providing insights into feature influence. Despite its limitations in handling non-linear relationships and interactions between features without additional feature engineering, Logistic Regression's performance is generally quite good when the underlying assumptions of the model are met. In practice, it often serves as a baseline model against which more complex models can be compared.

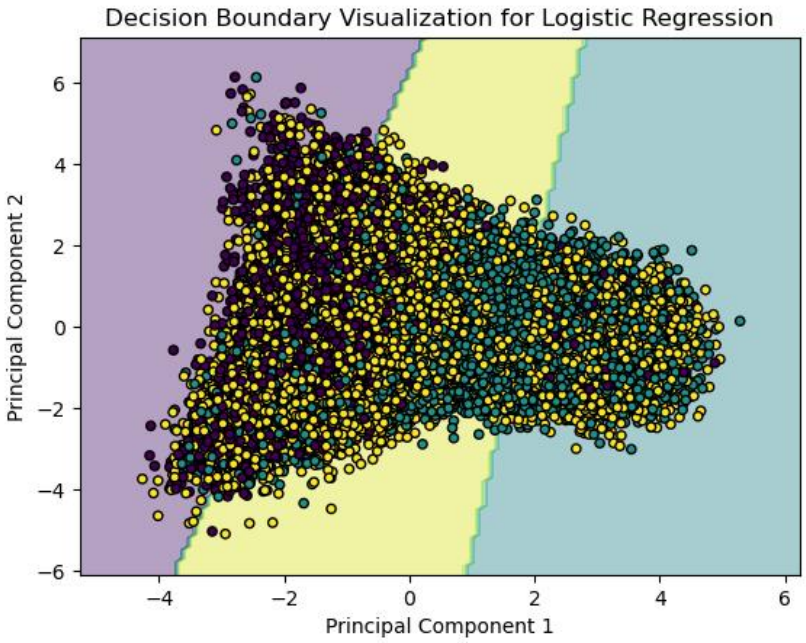
Model Tuning/Training:

The overview of model tuning and training would involve systematically preparing the Logistic Regression model to perform effectively on the given task. The training phase entails fitting the model to the dataset, allowing it to learn the relationship between the features and the target variable. For Logistic Regression, this involves finding the set of coefficients that best predict the target class probabilities through a process like maximum likelihood estimation.

Effectiveness of Algorithm:

The performance of the Logistic Regression Model as indicated by the provided metrics is moderate to low:

- Accuracy (0.59155): With an accuracy of about 59.2%, the model correctly identifies the outcome for a little over half of the cases in the dataset, which suggests room for improvement in the model's ability to generalize.
- Precision: The precision scores indicate the model's ability to predict positive instances correctly: 0.49 for Class 0, 0.64 for Class 1, and 0.59 for Class 2. This suggests the model is most precise when identifying Class 1 but less so for Class 0 and Class 2.
- Recall: The recall scores reflect the model's sensitivity to detecting positive instances: 0.22 for Class 0, 0.41 for Class 1, and 0.81 for Class 2. The model is most effective at identifying true positives for Class 2 but struggles significantly with Class 0 and to a lesser extent with Class 1.
- F1-Score: The F1-scores are a harmonic mean of precision and recall, with the model scoring 0.30 for Class 0, 0.50 for Class 1, and 0.68 for Class 2. These scores suggest that the balance between precision and recall is best for Class 2, but there is a notable deficiency in Class 0 and a moderate one in Class 1.



XG Boost

Justification for XG Boost

XGBoost is celebrated for its execution speed, model accuracy, and the ability to fine-tune many hyperparameters, which can greatly enhance model performance. It also features

regularization to avoid overfitting, making it a powerful choice for predictive modeling. The algorithm's effectiveness in both classification and regression tasks, as well as its capability to manage missing data, likely contributed to its selection in the modeling process detailed within the notebook.

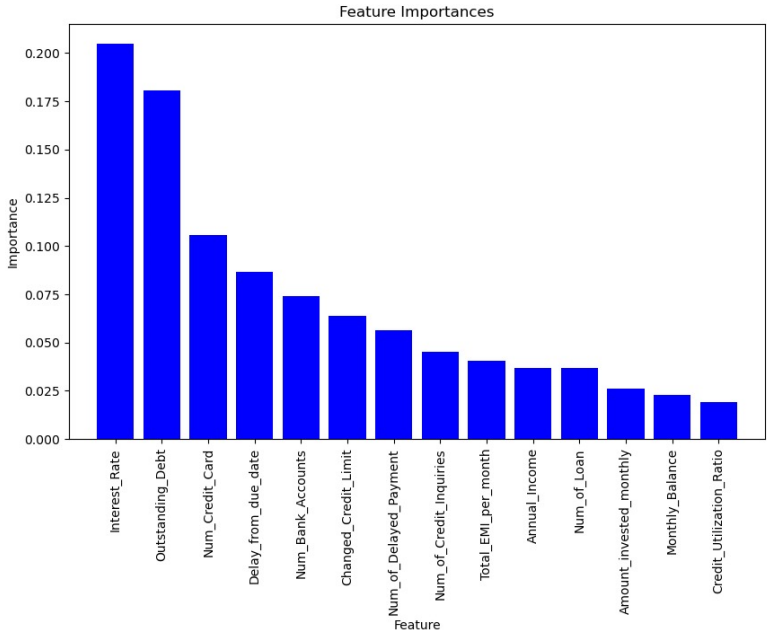
Model Tuning/Training:

Training and tuning the XGBoost model entails iteratively building an ensemble of decision trees, with each one correcting the errors of its predecessors, to improve the predictive accuracy. This process is fine-tuned through hyperparameter adjustments like learning rate, tree depth, and regularization, using cross-validation techniques to prevent overfitting and ensure that the model generalizes well. The overarching aim is to harness XGBoost's powerful capabilities to strike a balance between model complexity and performance, optimizing for the best results on both the training set and unseen data.

Effectiveness of Algorithm:

The provided metrics indicate that the model exhibits a relatively good level of performance:

- Accuracy (0.7328): The model has an overall accuracy of about 73.3%, which suggests that it is able to correctly predict the class label for nearly three-quarters of the instances in the test set.
- Precision: The precision scores for the classes are 0.66 for Class 0, 0.76 for Class 1, and 0.74 for Class 2. These scores demonstrate the model's ability to correctly identify positive instances, performing best in Class 1.
- Recall: The recall scores are 0.60 for Class 0, 0.70 for Class 1, and 0.80 for Class 2. This indicates how well the model can identify all relevant instances within each class, with the highest recall for Class 2.
- F1-Score: The F1-scores, which are the harmonic mean of precision and recall, are 0.63 for Class 0, 0.73 for Class 1, and 0.77 for Class 2, indicating a balanced performance between precision and recall, especially for Class 2.



Model Comparision

Based on the performance metrics provided for each algorithm, we make a comparative overview:

Logistic Regression:

- Accuracy: 59.2%
- Precision: Ranges from 0.49 (Class 0) to 0.64 (Class 1)
- Recall: Ranges from 0.22 (Class 0) to 0.81 (Class 2)
- This model has the lowest accuracy among the algorithms compared. Its recall for Class 0 is notably low, indicating a significant number of false negatives. It performs best in identifying Class 2 instances.

K-Nearest Neighbors (KNN):

- Accuracy: 68.1%
- Precision: Ranges from 0.53 (Class 0) to 0.72 (Class 2)
- Recall: Ranges from 0.57 (Class 0) to 0.72 (Class 2)
- KNN shows a moderate improvement over Logistic Regression, with better overall accuracy and balanced precision and recall scores.

Gradient Boosting:

- Accuracy: 68.9%
- Precision: Ranges from 0.61 (Class 0) to 0.74 (Class 2)
- Recall: Ranges from 0.48 (Class 0) to 0.80 (Class 2)
- Gradient Boosting's performance is close to that of KNN but shows a slightly higher precision for Class 2 and better recall for Class 2, making it more balanced in terms of precision and recall.

Random Forest:

- Accuracy: 77.5%
- Precision: Ranges from 0.73 (Class 0) to 0.78 (Class 2)
- Recall: Ranges from 0.66 (Class 0) to 0.81 (Class 2)
- Random Forest outperforms the other models in accuracy and maintains relatively high precision and recall across all classes.

Decision Tree:

- Accuracy: 69.7%
- Precision: Ranges from 0.60 (Class 0) to 0.73 (Class 2)
- Recall: Ranges from 0.62 (Class 0) to 0.72 (Class 2)
- The Decision Tree has a comparable performance to KNN but is outperformed by the Random Forest, which is expected given that the Random Forest is an ensemble of Decision Trees.

XGBoost:

- Accuracy: 73.3%
- XGBoost shows higher accuracy than KNN, Decision Tree, and Gradient Boosting, but is slightly behind Random Forest.
- XGBoost has balanced precision and recall, indicating strong performance, particularly for Class 2, where it has a recall of 0.80 and a precision of 0.74.