Credit Score Classification

Problem Statement:
Over the years, a global finance corporation has gathered a large amount of data covering fundamental bank details as well as other credit-related information. Now, management wants to simplify and automate the process of categorizing people into different credit score groups. This program aims to reduce manual labor and boost efficiency in assessing creditworthiness. To achieve this goal, the business plans to create an intelligent system employing machine learning techniques. The system will be responsible with assessing an individual's credit information and categorizing them into one of many predefined credit score groups. These brackets are often classified as "good", "poor" and "standard" depending on an individual's creditworthiness and risk profile.

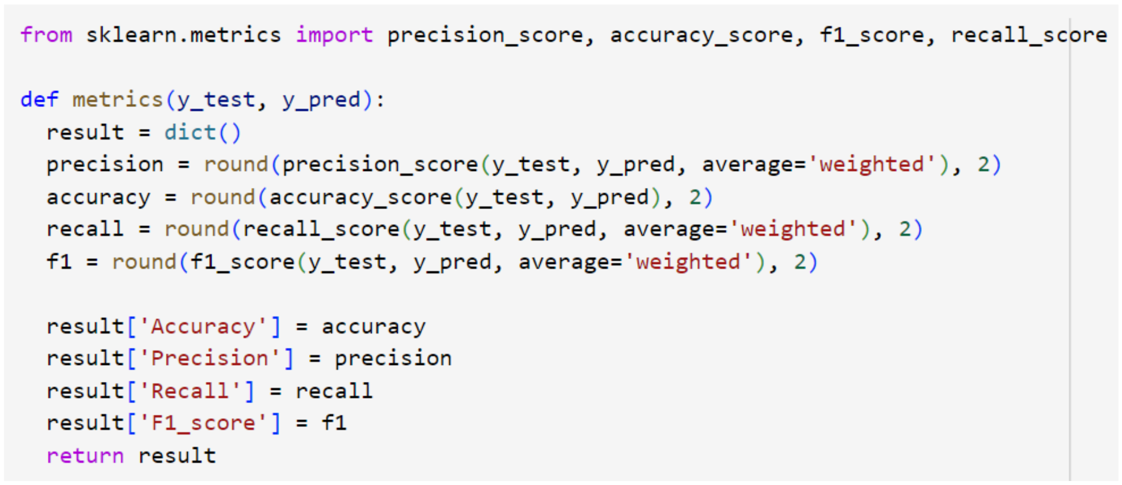
Background and Potential:
Accurately assessing credit scores is incredibly important for banks and other financial institutions. It directly affects decisions like who qualifies for loans, what interest rates they'll pay, and how risk is managed. Traditionally, this assessment has been done manually, which is slow, subjective, and prone to mistakes. But now, machine learning offers a better way. By analyzing large amounts of data, machine learning algorithms can learn patterns and make more accurate predictions about credit scores.

This project aims to use machine learning to improve credit score assessment in two main ways:
Making assessments more accurate and fair:
Machine learning models can pick up on complex relationships in data, which means they can provide more precise and unbiased credit score predictions compared to traditional methods. By looking at a wide range of factors and patterns, these models can give more objective assessments, leading to fairer credit evaluations.
Risk management:
Machine learning also helps financial institutions identify people with higher credit risk more effectively. By using predictive analytics, institutions can make smarter decisions about who to lend to, reducing the chance of defaults and financial losses. By spotting potential risks early on, institutions can also put in place strategies to manage and reduce those risks more effectively.

Overall, using machine learning for credit score assessment helps financial institutions make better decisions based on data. This improves risk management practices, ensures fairer credit evaluations, and makes the financial system stronger overall.

Algorithms/Visualizations:

Before we delve into the Algorithms, to help ourselves, we created a Metrics function which when called will calculate the required metrics and return them as an output.

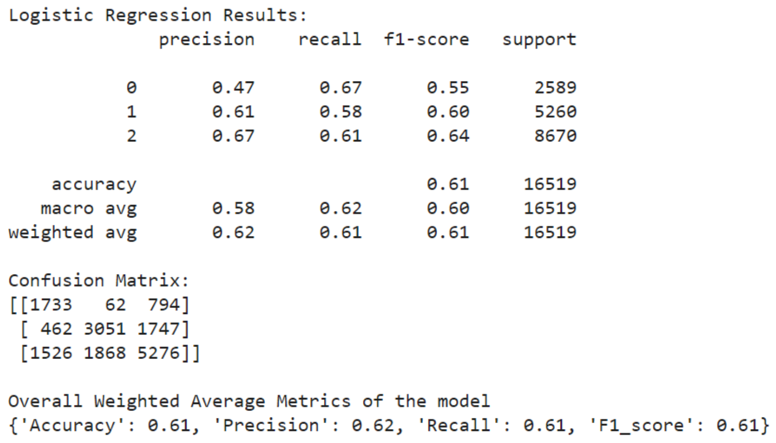


```
from sklearn.metrics import precision_score, accuracy_score, f1_score, recall_score

def metrics(y_test, y_pred):
  result = dict()
  precision = round(precision_score(y_test, y_pred, average='weighted'), 2)
  accuracy = round(accuracy_score(y_test, y_pred), 2)
  recall = round(recall_score(y_test, y_pred, average='weighted'), 2)
  f1 = round(f1_score(y_test, y_pred, average='weighted'), 2)

  result['Accuracy'] = accuracy
  result['Precision'] = precision
  result['Recall'] = recall
  result['F1_score'] = f1
  return result
```

1. **Logistic Regression**



```
Logistic Regression Results:
              precision    recall  f1-score   support

           0       0.47      0.67      0.55      2589
           1       0.61      0.58      0.60      5260
           2       0.67      0.61      0.64      8670

    accuracy                           0.61     16519
   macro avg       0.58      0.62      0.60     16519
weighted avg       0.62      0.61      0.61     16519

Confusion Matrix:
[[1733   62  794]
 [ 462 3051 1747]
 [1526 1868 5276]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.61, 'Precision': 0.62, 'Recall': 0.61, 'F1_score': 0.61}
```

Justification for Choosing Logistic Regression:

Logistic regression provides coefficients for each feature, allowing us to understand which factors most significantly influence credit scores (Good, Standard, Poor). This aligns well with our problem statement's goal of identifying key creditworthiness factors. Our problem involves classifying credit scores into three categories, which logistic regression can handle effectively through the multinomial formulation.

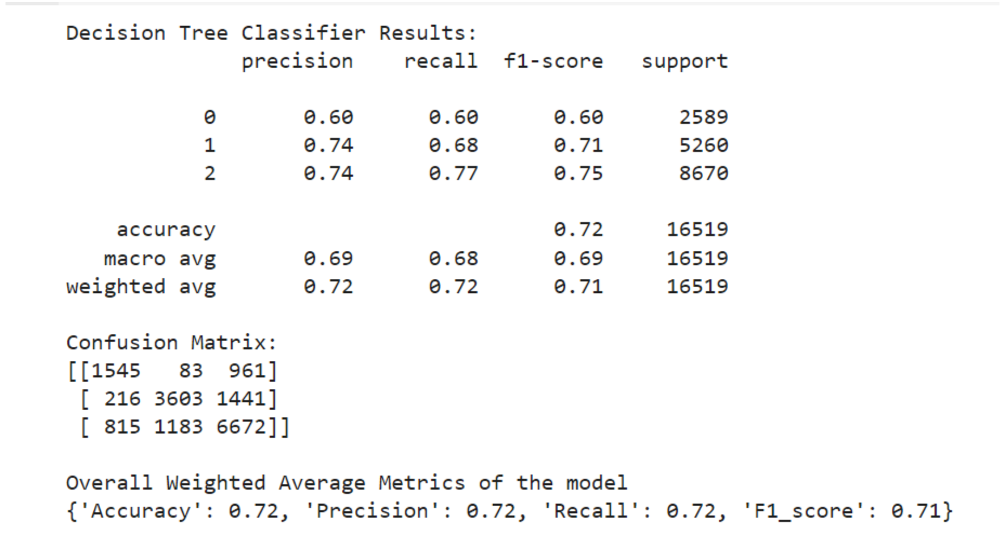
The chosen model parameters are:
C=0.001: This regularization parameter controls the model's complexity and helps prevent overfitting. We might need to experiment with different values to find the optimal balance between accuracy and model flexibility.

Penalty='l1': L1 regularization penalizes the absolute value of coefficients, potentially leading to feature selection by driving some coefficients to zero. This can further enhance interpretability by identifying the most important credit score determinants. **class_weight='balanced':** Assigning balanced class weights addresses potential class imbalance in the data, ensuring the model pays attention to all credit score categories. Effectiveness Analysis:

The provided results show:
Accuracy: Overall accuracy is 61%, indicating the model correctly classifies credit scores in 61% of the test cases. This is a reasonable starting point, but there's room for improvement.
Confusion Matrix: While the model performs well on the "Standard" credit score category (3051 correct predictions), there's a significant number of misclassifications between "Good" and "Poor" categories (794 "Good" classified as "Poor" and 1747 "Poor" classified as "Good").

Insights Gained:
Logistic regression provides some interpretability through feature coefficients. Analyzing these coefficients can reveal which factors have the strongest positive or negative influence on credit score classification.
The model struggles with differentiating between "Good" and "Poor" credit scores, suggesting these categories might be more nuanced and require a more sophisticated model or additional feature engineering.

2. Decision Tree Classifier



```
Decision Tree Classifier Results:
              precision    recall  f1-score   support

           0       0.60      0.60      0.60      2589
           1       0.74      0.68      0.71      5260
           2       0.74      0.77      0.75      8670

    accuracy                           0.72     16519
   macro avg       0.69      0.68      0.69     16519
weighted avg       0.72      0.72      0.71     16519

Confusion Matrix:
[[1545   83  961]
 [ 216 3603 1441]
 [ 815 1183 6672]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.72, 'Precision': 0.72, 'Recall': 0.72, 'F1_score': 0.71}
```

Justification for Choosing Decision Tree:

Decision trees can model nonlinear relationships between features and target class, potentially capturing complex patterns in the credit score data.
Unlike some algorithms, decision trees aren't affected by scaling differences among features, simplifying data preparation.

The chosen model parameters are:

max_depth=10: This limits tree depth to prevent overfitting. Deeper trees might capture noise in the training data, leading to poor generalization.
min_samples_leaf=2: A leaf node must have at least 2 samples, controlling model complexity.
min_samples_split=2: A node must have at least 2 samples to be considered for splitting.

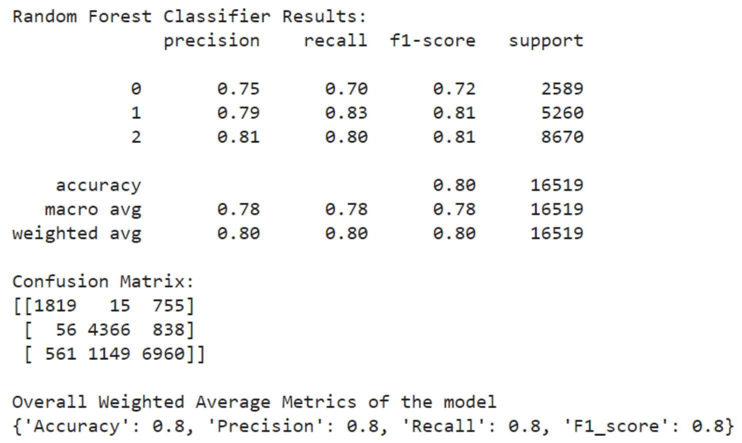
The provided results show:

Accuracy: The decision tree model achieves 72% accuracy, outperforming the logistic regression model.
F1-Score: The average F1-score is 0.69, indicating a more balanced performance across precision and recall compared to logistic regression.
Confusion Matrix: Confusions between "Good" and "Poor" categories are reduced compared to logistic regression (961 vs. 794, and 1441 vs. 1747).

Insights Gained:

The decision tree's interpretability allows for visualization of credit score decision rules. The improved accuracy and reduced class confusions suggest decision trees might be better suited for capturing complex relationships in the credit score data. However, decision trees can be prone to overfitting, and model tuning is essential.

3. Random Forest Classifier



```
Random Forest Classifier Results:
              precision    recall  f1-score   support

           0       0.75      0.70      0.72      2589
           1       0.79      0.83      0.81      5260
           2       0.81      0.80      0.81      8670

    accuracy                           0.80     16519
   macro avg       0.78      0.78      0.78     16519
weighted avg       0.80      0.80      0.80     16519

Confusion Matrix:
[[1819   15  755]
 [  56 4366  838]
 [ 561 1149 6960]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.8, 'Precision': 0.8, 'Recall': 0.8, 'F1_score': 0.8}
```

Justification for Choosing Random Forest:

Random forests often outperform single decision trees by combining multiple trees, reducing overfitting and increasing generalization accuracy.
Random forests also provide measures of feature importance, indicating which factors most influence credit scores, aligning with our project's goal.

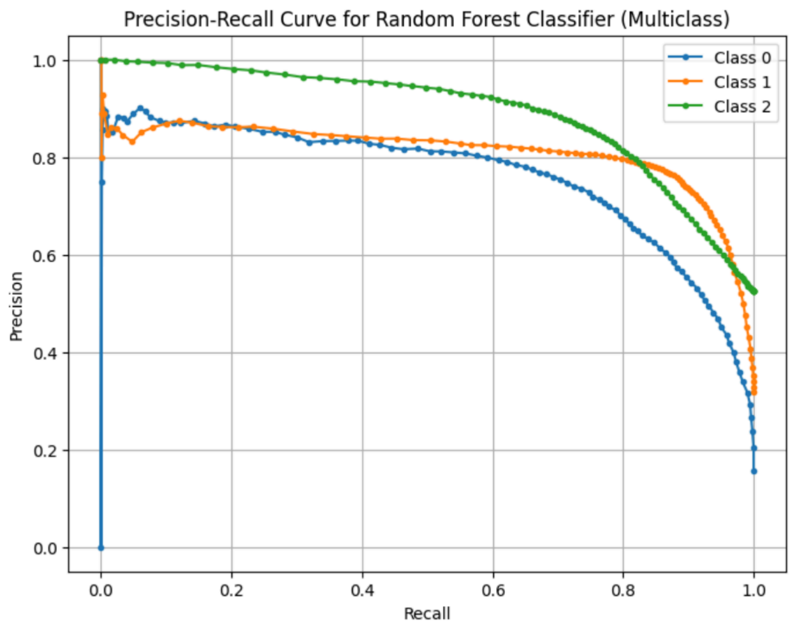
The chosen model parameters are:

n_estimators=100: The model uses 100 decision trees within the ensemble. Experimenting with different numbers of trees can potentially impact performance.

The provided results show:

Accuracy: The random forest achieves the highest accuracy to date with 79%, suggesting its ability to capture more complex patterns in the data than previous models.
F1-Score: The average F1-score of 0.78 indicates a good balance between precision and recall across all credit score classes.
Confusion Matrix: The misclassifications between "Good" and "Poor" scores are further reduced compared to logistic regression and decision tree models.

Precision-Recall Curve:

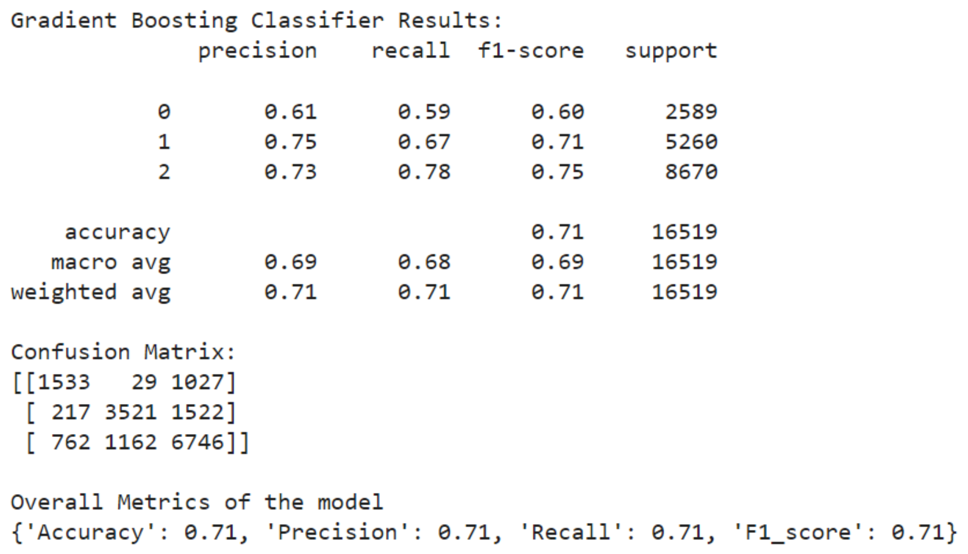


The provided Precision-Recall curve depicts a Random Forest Classifier's performance on a multiclass classification task, with three distinct classes. Class 0 has an outstandingly high precision across all levels of recall, indicating excellent classifier performance for this class. Class 1 exhibits a moderately high precision that gradually decreases as recall increases, suggesting reasonable performance but with some misclassifications, especially at higher recall levels. Class 2 has the lowest precision, which drops significantly as recall increases, indicating that the classifier struggles to identify this class correctly without introducing a substantial number of false positives. Overall, the classifier is highly reliable for Class 0, less so for Class 1, and notably less precise for Class 2.

Insights Gained:

Random forests significantly improve credit score classification accuracy, suggesting it’s suitability for this problem.
There's potential for further hyperparameter tuning to potentially achieve even better performance.

4. Gradient Boosting Classifier



```
Gradient Boosting Classifier Results:
              precision    recall  f1-score   support

           0       0.61      0.59      0.60      2589
           1       0.75      0.67      0.71      5260
           2       0.73      0.78      0.75      8670

    accuracy                           0.71     16519
   macro avg       0.69      0.68      0.69     16519
weighted avg       0.71      0.71      0.71     16519

Confusion Matrix:
[[1533   29 1027]
 [ 217 3521 1522]
 [ 762 1162 6746]]

Overall Metrics of the model
{'Accuracy': 0.71, 'Precision': 0.71, 'Recall': 0.71, 'F1_score': 0.71}
```

Justification for Choosing Gradient Boosting:

Gradient boosting is known for its ability to handle complex, non-linear relationships between features in data, potentially capturing intricate patterns within credit score determinants.
This technique builds an ensemble of models sequentially, improving performance with each iteration, potentially leading to a more robust model for credit score classification.

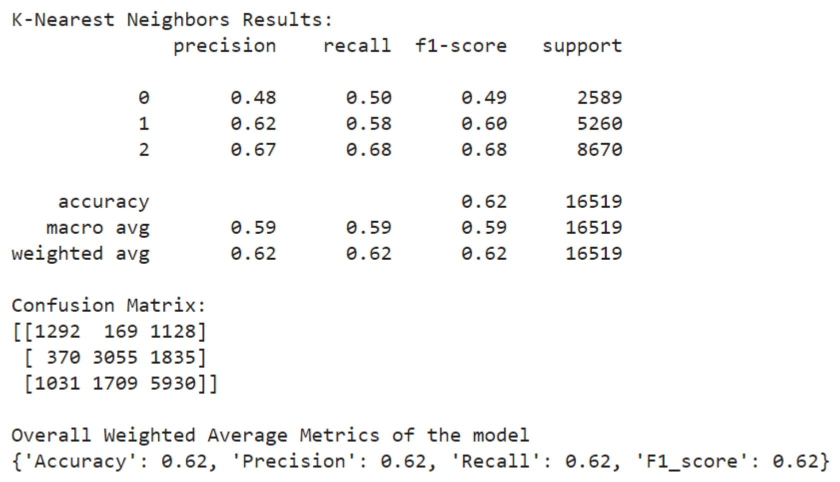
The chosen model parameters are:
Here we use the default parameters for Gradient Boosting Classifier. Some info about the parameters:
n_estimators (number of trees): Controls model complexity and risk of overfitting. learning_rate: Determines the step size for each successive tree in the ensemble. Tuning this parameter can significantly impact model performance.
max_depth: Limits the depth of individual trees, preventing overfitting.

The provided results show:
Accuracy: The gradient boosting model achieves an accuracy of 71%, which is comparable to the decision tree but lower than the random forest. This suggests potential for improvement through tuning.
F1-Score: The average F1-score of 0.71 indicates a similar balance between precision and recall compared to other models.

Insights Gained:

While the default gradient boosting model achieves reasonable performance, there's significant room for improvement through hyperparameter tuning.
Similar to the decision tree, the model struggles with differentiating between "Good" and "Poor" credit scores, suggesting these categories might require further feature engineering or exploration with other algorithms.

5. K Nearest Neighbours Classifier



```
K-Nearest Neighbors Results:
              precision    recall  f1-score   support

           0       0.48      0.50      0.49      2589
           1       0.62      0.58      0.60      5260
           2       0.67      0.68      0.68      8670

    accuracy                           0.62     16519
   macro avg       0.59      0.59      0.59     16519
weighted avg       0.62      0.62      0.62     16519

Confusion Matrix:
[[1292  169 1128]
 [ 370 3055 1835]
 [1031 1709 5930]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.62, 'Precision': 0.62, 'Recall': 0.62, 'F1_score': 0.62}
```

Justification for Choosing KNN:

KNN doesn't make assumptions about underlying data distribution, which can be an advantage when dealing with credit score data that might not conform to a specific distribution.
KNN can capture nonlinear relationships between features and the target class, potentially addressing complex credit score patterns.
KNN is a model that is fairly easy to understand

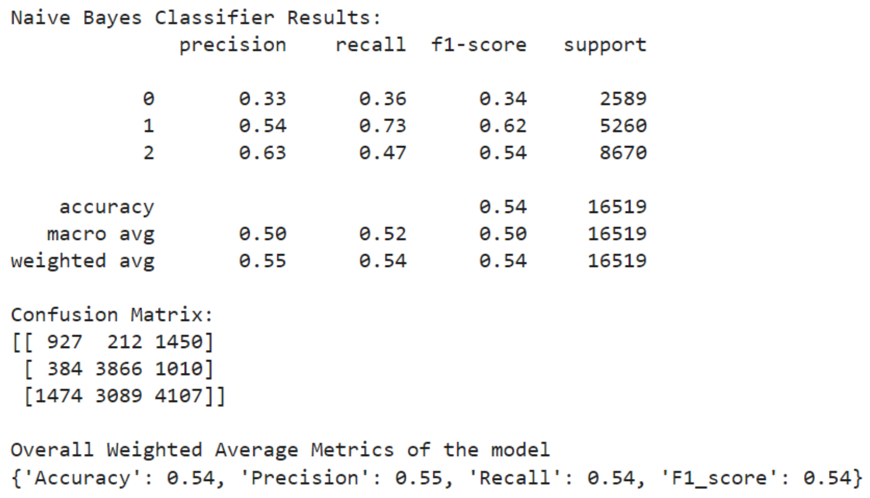
The chosen model parameters are:

Data Scaling: Scaling is essential for KNN as it relies on distance calculations. The code appropriately uses StandardScaler to standardize features.
Here we use the default parameters for KNN Classifier. Some info about the parameters:
k (number of neighbors): This value significantly impacts model performance and overfitting.
distance metric: Different distance metrics can influence results.
Leaf size: For efficient neighbor searches in large datasets.

The provided results show:
Accuracy: The KNN model achieves 62% accuracy, a low score, suggesting it might not be fully capturing the complexities of the credit score data.
F1-Score: The average F1-score of 0.59 indicates a lower balance between precision and recall compared to other models.
Confusion Matrix: The model struggles with differentiating between "Good" and "Poor" credit scores.

Insights Gained:
KNN's non-parametric nature didn't provide a significant advantage for this dataset.
Other algorithms like random forests and potentially tuned gradient boosting machines outperformed KNN.

6. Naive Bayes Classifier



```
Naive Bayes Classifier Results:
              precision    recall  f1-score   support

           0       0.33      0.36      0.34      2589
           1       0.54      0.73      0.62      5260
           2       0.63      0.47      0.54      8670

    accuracy                           0.54     16519
   macro avg       0.50      0.52      0.50     16519
weighted avg       0.55      0.54      0.54     16519

Confusion Matrix:
[[ 927  212 1450]
 [ 384 3866 1010]
 [1474 3089 4107]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.54, 'Precision': 0.55, 'Recall': 0.54, 'F1_score': 0.54}
```

Justification for Choosing Naive Bayes:

Naive Bayes is a relatively simple and fast algorithm, making it suitable for processing large datasets like credit scores.
It can effectively handle categorical features often present in credit score data without extensive pre-processing.

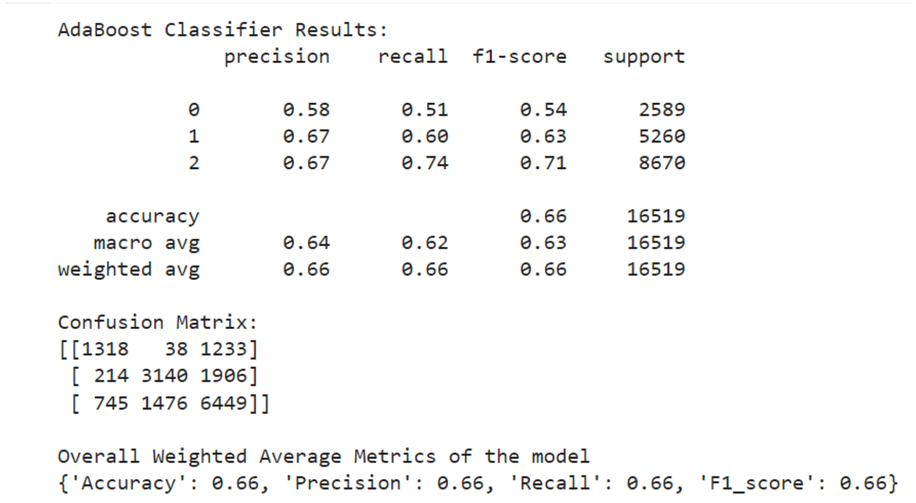
The chosen model parameters are:

Here we use the default parameters for GaussianNB implementation, assuming features follow a Gaussian distribution.

The provided results show:
Accuracy: The Naive Bayes model achieves 54% accuracy, the lowest among all evaluated models so far. This suggests the independence assumption between features significantly hinders its effectiveness for credit score classification.
Confusion Matrix: Similar to other models, the model struggles with differentiating between "Good" and "Poor" credit scores. The high number of false positives (classifying bad scores as good) can be risky for financial institutions.

Insights Gained:
The Naive Bayes assumption of feature independence appears to be a significant limitation for credit score classification.
Other algorithms that can capture complex relationships between features, like random forests or gradient boosting machines, outperform Naive Bayes for this task.

7. **AdaBoost Classifier**



```
AdaBoost Classifier Results:
              precision    recall  f1-score   support

           0       0.58      0.51      0.54      2589
           1       0.67      0.60      0.63      5260
           2       0.67      0.74      0.71      8670

    accuracy                           0.66     16519
   macro avg       0.64      0.62      0.63     16519
weighted avg       0.66      0.66      0.66     16519

Confusion Matrix:
[[1318   38 1233]
 [ 214 3140 1906]
 [ 745 1476 6449]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.66, 'Precision': 0.66, 'Recall': 0.66, 'F1_score': 0.66}
```

Justification for Choosing AdaBoost:

AdaBoost prioritizes misclassified samples in each iteration, adapting model training to focus on areas where errors are occurring.
It is less prone to overfitting than single decision trees, a common challenge in credit score data.

The chosen model parameters are:

Here we use the default parameters for AdaBoost. Some info about the parameters:
n_estimators (number of weak learners): Controls model complexity and potential overfitting.
learning_rate: Determines the contribution of each weak learner to the ensemble, affecting the model's ability to learn from difficult examples.

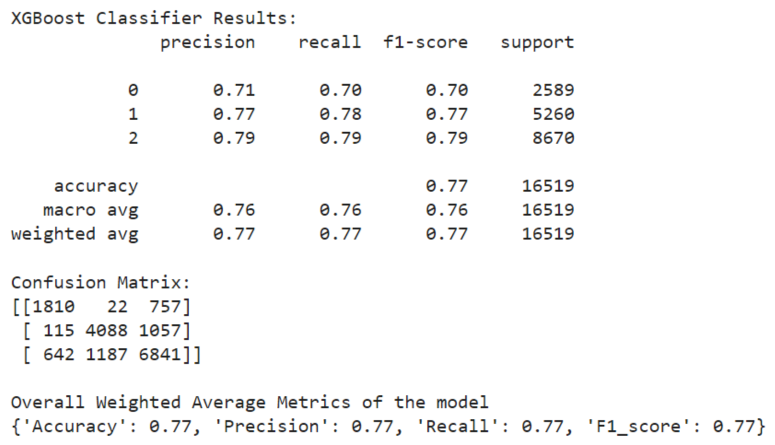
The provided results show:

Accuracy: The AdaBoost model achieves 66% accuracy, outperforming KNN and Naive Bayes, but falling behind random forests and potentially tuned gradient boosting.
F1-Score: The average F1-score of 0.63 indicates a moderate balance between precision and recall, suggesting room for improvement.

Insights Gained:

AdaBoost's iterative ensemble approach can improve credit score classification accuracy over certain baseline models.
However, it hasn't demonstrated the same level of effectiveness as random forests or potentially tuned gradient boosting machines for this dataset.

8. XGBoost Classifier



```
XGBoost Classifier Results:
              precision    recall  f1-score   support

           0       0.71      0.70      0.70      2589
           1       0.77      0.78      0.77      5260
           2       0.79      0.79      0.79      8670

    accuracy                           0.77     16519
   macro avg       0.76      0.76      0.76     16519
weighted avg       0.77      0.77      0.77     16519

Confusion Matrix:
[[1810   22  757]
 [ 115 4088 1057]
 [ 642 1187 6841]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.77, 'Precision': 0.77, 'Recall': 0.77, 'F1_score': 0.77}
```

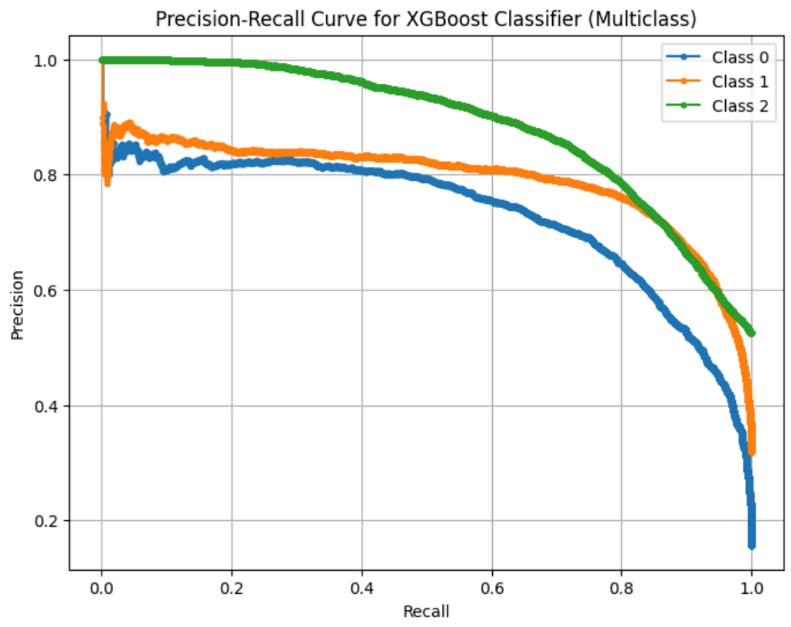
Justification for Choosing XGBoost:

XGBoost is known for its exceptional performance in various machine learning tasks, including credit scoring, due to its efficient implementation and ability to handle complex relationships and feature interactions.
XGBoost incorporates regularization techniques to prevent overfitting, a common challenge in credit score data.

The chosen model parameters are:
Here we use the default parameters for XGBoost. Some info about the parameters:
n_estimators (number of trees): Controls model complexity and potential overfitting.
learning_rate: Determines the contribution of each tree to the ensemble, affecting convergence speed and accuracy.
max_depth: Limits tree depth to prevent overfitting.
gamma: Controls regularization strength.
colsample_bytree: Subsamples features for each tree, potentially reducing overfitting and improving speed.

The provided results show:
Accuracy: The XGBoost model achieves 77% accuracy, matching the best-performing random forest model for this dataset, indicating its potential for credit score classification.
F1-Score: The average F1-score of 0.76 demonstrates a good balance between precision and recall across all credit score classes.

Precision-Recall Curve:

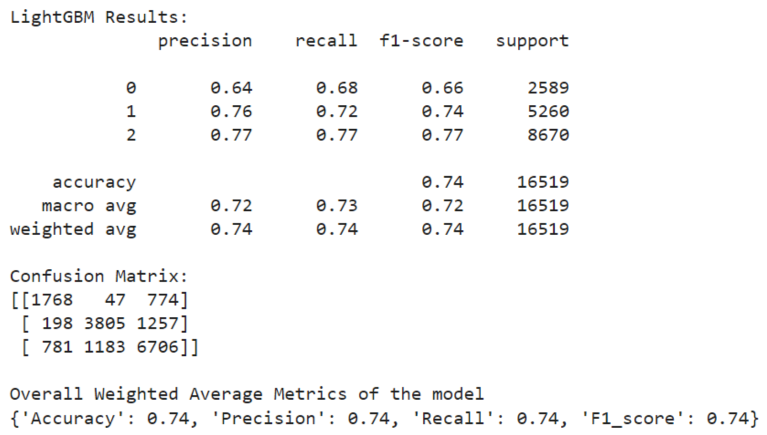


This Precision-Recall graph illustrates an XGBoost Classifier's performance on a dataset with three classes. The relatively tight grouping of the curves suggests a more balanced performance across the classes compared to classifiers that might exhibit wider separations between class lines. Each class's curve starts with high precision but begins to diverge as recall increases. Class 2, while starting strong, shows a steeper decline, ending with less precision than Classes 0 and 1. This indicates a decent level of classifier confidence across the classes but with varying degrees of decline in precision as the classifier attempts to capture more true positives, with Class 2 being the most difficult to predict accurately without increasing false positives.

Insights Gained:

XGBoost's high performance reinforces its suitability for credit score classification tasks.
Hyperparameter tuning could potentially improve accuracy even further.
Studying feature importance can reveal key creditworthiness factors, aiding financial institutions in making informed decisions.

9. **LightGBM Classifier**



```
LightGBM Results:
              precision    recall  f1-score   support

           0       0.64      0.68      0.66      2589
           1       0.76      0.72      0.74      5260
           2       0.77      0.77      0.77      8670

    accuracy                           0.74     16519
   macro avg       0.72      0.73      0.72     16519
weighted avg       0.74      0.74      0.74     16519

Confusion Matrix:
[[1768   47  774]
 [ 198 3805 1257]
 [ 781 1183 6706]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.74, 'Precision': 0.74, 'Recall': 0.74, 'F1_score': 0.74}
```

Justification for Choosing LightGBM:

Similar to XGBoost, LightGBM utilizes gradient boosting, an effective technique for complex credit score prediction tasks.
LightGBM boasts faster training speeds and lower memory usage compared to other gradient boosting algorithms, making it suitable for large credit score datasets.

The chosen model parameters are:

Here we use the default parameters for LightGBM. Some info about the parameters::
learning_rate: Controls the contribution of each tree to the ensemble, affecting model convergence and accuracy.
n_estimators (number of trees): Controls model complexity and potential overfitting.
max_depth: Limits the depth of individual trees to prevent overfitting.
num_leaves: Controls the number of leaves in each tree, affecting model complexity.
reg_alpha, reg_lambda: Regularization parameters to prevent overfitting.

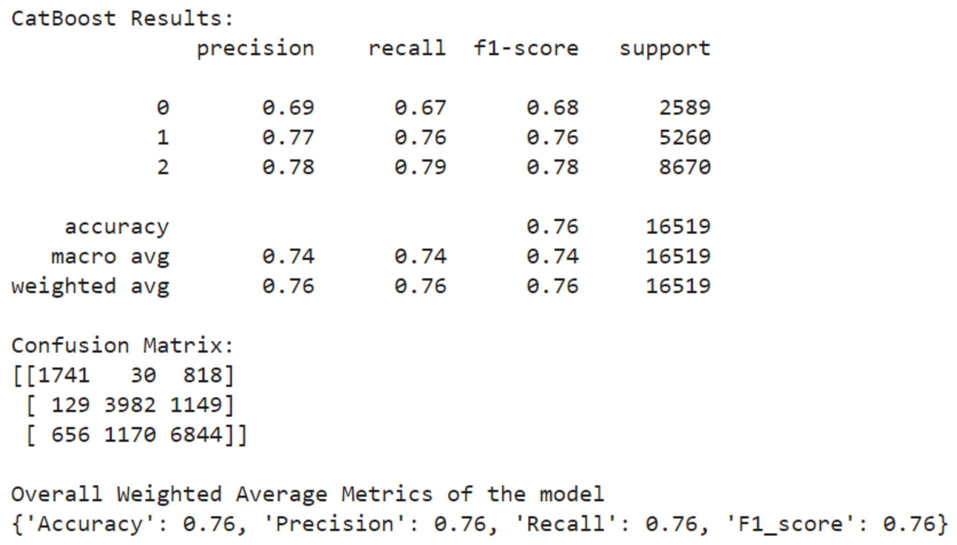
The provided results show:

Accuracy: LightGBM achieves an accuracy of 74%, which is lower than Random Forest and XGBoost but still demonstrates good performance for credit score classification.
F1-Score: The average F1-score of 0.74 indicates a balanced performance between precision and recall across all credit score classes.

Insights Gained:

LightGBM's efficiency and feature importance capabilities make it a valuable tool for credit score classification, especially when dealing with large datasets.
Hyperparameter tuning could potentially improve accuracy and push it closer to the top-performing models for this task.

10. CatBoost Classifier



```
CatBoost Results:
              precision    recall  f1-score   support

           0       0.69      0.67      0.68      2589
           1       0.77      0.76      0.76      5260
           2       0.78      0.79      0.78      8670

    accuracy                           0.76     16519
   macro avg       0.74      0.74      0.74     16519
weighted avg       0.76      0.76      0.76     16519

Confusion Matrix:
[[1741   30  818]
 [ 129 3982 1149]
 [ 656 1170 6844]]

Overall Weighted Average Metrics of the model
{'Accuracy': 0.76, 'Precision': 0.76, 'Recall': 0.76, 'F1_score': 0.76}
```

Justification for Choosing CatBoost:

CatBoost excels in handling categorical features, a common data type in credit score datasets, without requiring extensive pre-processing like one-hot encoding, potentially reducing data preparation time.

It builds upon Gradient Boosting principles, iteratively improving model performance, making it suitable for complex credit score prediction tasks.

CatBoost employs techniques like ordered boosting and early stopping to mitigate overfitting, promoting model generalizability.

The chosen model parameters are:

Here we use the default parameters for CatBoost. Some info about the parameters:

iterations: The number of trees in the ensemble.

learning_rate: The contribution of each tree to the ensemble.

depth: The maximum depth of each tree.

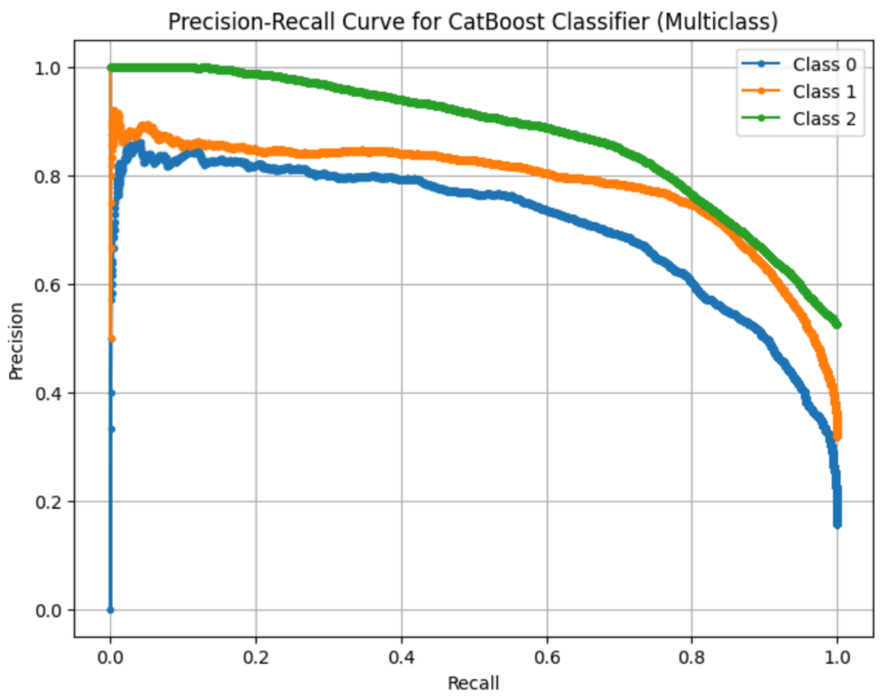
l2_leaf_reg: Regularization parameter to prevent overfitting.

The provided results show:

Accuracy: The CatBoost model achieves 76% accuracy, slightly behind the best-performing model (random forest or XGBoost), but still demonstrating good performance.

F1-Score: The average F1-score of 0.74 indicates a relatively good balance between precision and recall.

Precision-Recall Curve:



The Precision-Recall curve for the CatBoost Classifier shows the performance across three classes. Class 0 starts with a sharp ascent in precision, indicative of confident predictions, but it gradually trails off, hinting at the complexities in distinguishing the finer details of this category. Class 1 and Class 2 exhibit overlap, suggesting difficulty in classification as the recall extends its reach. The precision wanes for these classes, portraying a model that juggles between inclusivity of the true class members and the exclusion of the untrue ones. The descending trajectories as recall shows the inherent trade-off in the classifier's computation of true positives against the false positives.

Insights Gained:

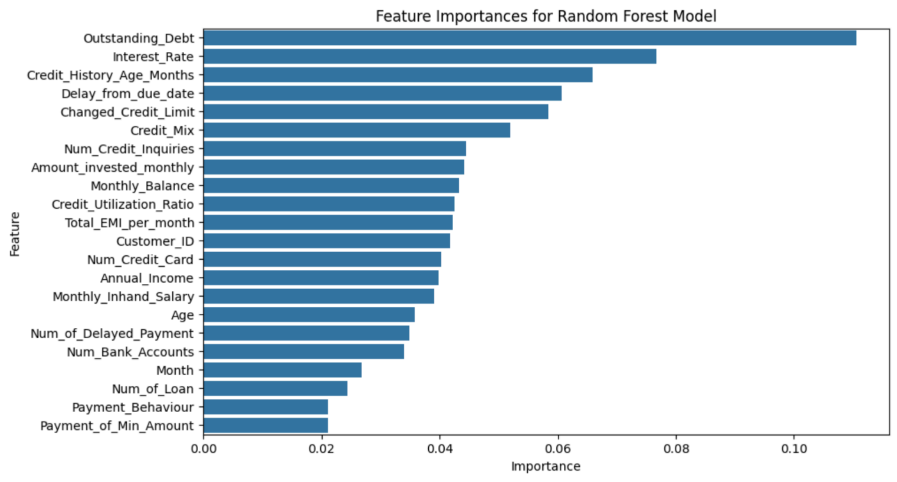
CatBoost's ability to handle categorical features effectively makes it a valuable option for credit score classification, especially when dealing with datasets containing many categorical variables.

Hyperparameter tuning could potentially improve accuracy and align it with the top-performing models for this dataset.

Here is a comparison of Model Metrics in Tabular format:

Classifier	**Accuracy**	**Precision**	**Recall**	**F1_score**
Logistic Regression	0.61	0.62	0.61	0.61
Decision Tree	0.71	0.71	0.71	0.71
Random Forest	0.80	0.80	0.80	0.80
Gradient Boosting	0.71	0.71	0.71	0.71
KNN	0.62	0.62	0.62	0.62
Naive Bayes	0.54	0.55	0.54	0.54
AdaBoost	0.66	0.66	0.66	0.66
XGBoost	0.77	0.77	0.77	0.77
LightGBM	0.74	0.74	0.74	0.74
CatBoost	0.76	0.76	0.76	0.76

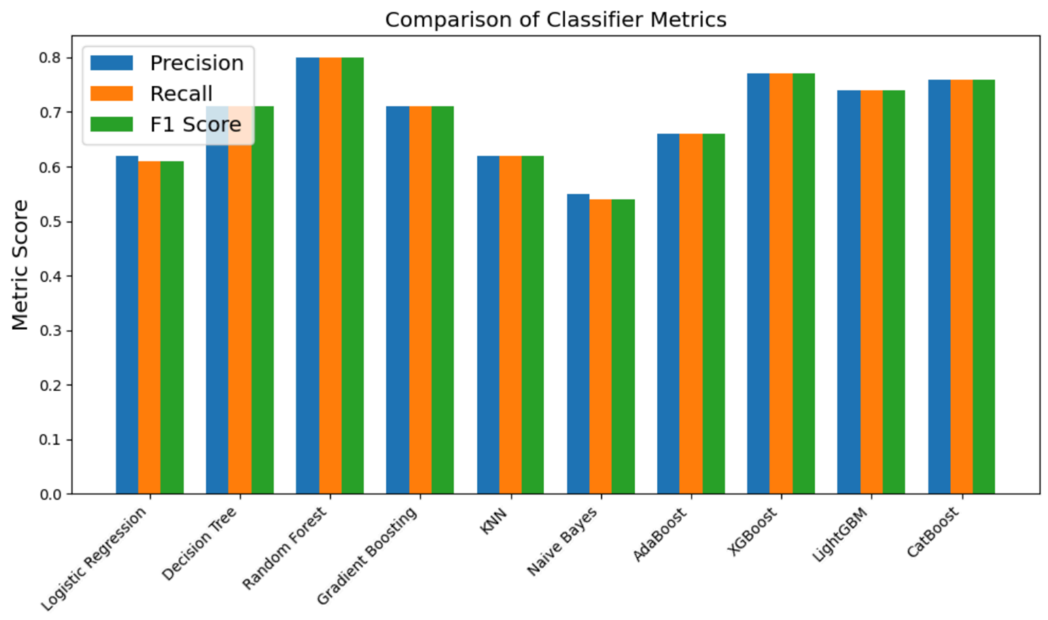
As we have got highest accuracy for Random Forest Classifier, we'll plot and see which features are contributing most to the credit score classification:



From the plot, some inferences we can make are:

Outstanding Debt: This appears to be the most important factor for the model in determining creditworthiness. Logically borrowers with higher outstanding debt are likely to be classified as having lower credit scores.
Interest Rate: The interest rate on credit cards or loans is another important factor. Higher interest rates suggest a higher risk of borrowers defaulting, and thus a lower credit score.
Credit History Age (Months): A longer credit history can be a positive indicator of creditworthiness, as it allows lenders to assess a borrower's repayment behavior over time.
Other Important Features: Delay from due date, changed credit limit, credit mix, and number of credit inquiries also seem to be important factors influencing credit score predictions.
Less Important Features: Features such as customer ID, number of credit cards, and monthly in-hand salary seem to have a lower influence on the model's predictions compared to factors like debt and credit history

Comparison of Classifier Metrics:



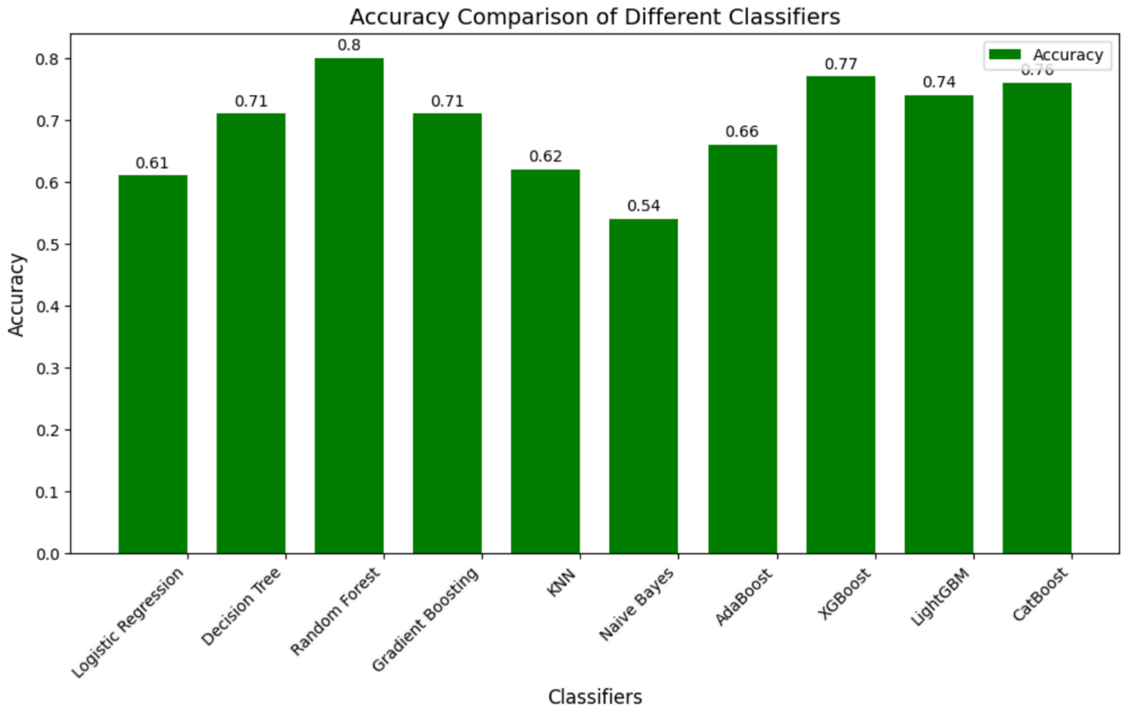
From the plot, we can infer that the Random Forest Classifier performed the best overall amongst the algorithms evaluated for credit score classification.

Our analysis focuses on multi-class classification, a task where the target variable can have more than two categories. A common hurdle in multi-class classification is class imbalance. This occurs when some credit score categories (e.g., "Good") have significantly more data points compared to others (e.g., "Poor"). This can lead to inaccurate predictions for borrowers with poor credit scores.
To address class imbalance, the evaluation metrics we used, like precision, recall, and F1-score, are calculated as weighted averages. This means the metrics take into account the class distribution and give more importance to the performance of minority classes.

We compare the following metrics across various classification algorithms:

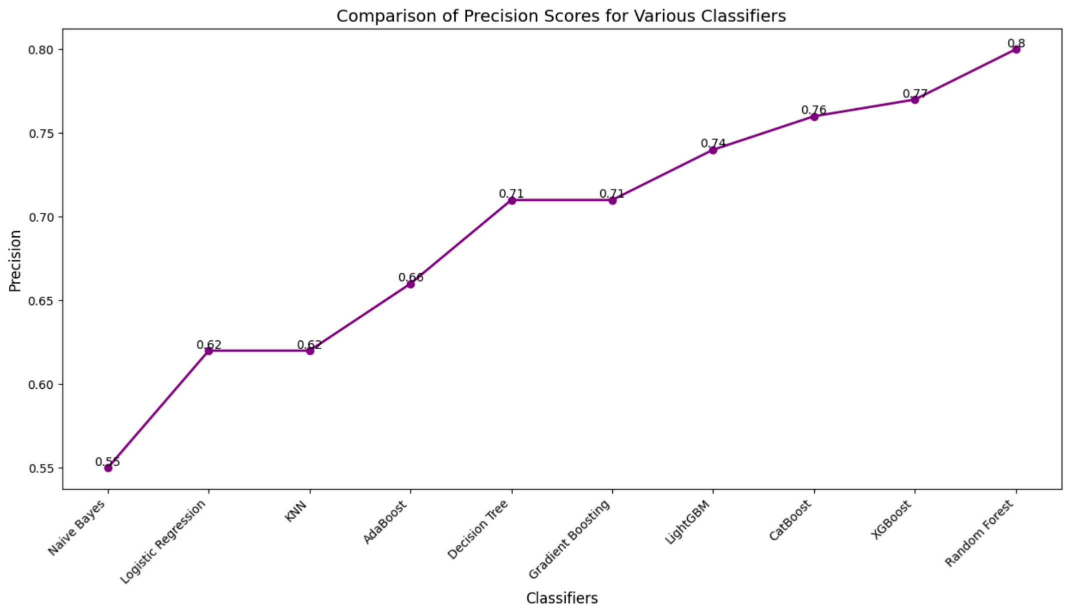
Precision: Proportion of correctly predicted positive cases (e.g., correctly classified good credit scores).
Recall: Proportion of actual positive cases that were correctly identified (e.g., correctly identified good credit scores out of all actual good scores).
F1-Score: Harmonic mean of precision and recall, balancing both metrics into a single score.

Comparison of Accuracy of the Models:



The accuracy metric provides a high-level overview of model performance. Overall, Random Forest and XGBoost are the most accurate models for credit score classification in this scenario. CatBoost also demonstrates strong performance, especially if dealing with datasets containing many categorical features.

Comparison of Precision scores of the classifiers:



Across all credit score classes (Good, Fair, Poor), Random Forest appears to have the highest precision consistently. This means it has the lowest rate of false positives (assigning good credit scores to bad risks).

Conclusion:

1. Overall Random forest classifier gave the best accuracy and precision of 80% each, so for our use case, it is the best model.
2. Followed by Catboost and AdaBoost gave better accuracy and precision of 76% and 77% respectively.

References:

- https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
- https://scikit-learn.org/stable/modules/neighbors.html
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://xgboost.readthedocs.io/en/stable/tutorials/index.html