Credit-Risk-Assessment Project Report

Saranya Pal
Department of Chemistry
210930
saranyap21@iitk.ac.in

Kamal Raj Rawat
Department of Chemistry
210484
kamakrr21@iitk.ac.in

Ayush kumar kharwar
Department of Civil Engineering
210248
ayushkk21@iitk.ac.in

Abstract

This project assesses the credit quality of bank customers using machine learning. The dataset includes various customer attributes, with a binary target variable classifying customers as "Good" or "Bad." We begin by fitting a logistic regression model, adjusting probability thresholds (20%, 35%, 50%) to evaluate performance through confusion matrices. The dataset is split into training and test sets for generalization assessment. We also explore tree-based models (classification trees, bagging, and random forests) and compare their performance with logistic regression using metrics like AUC and ROC curves. Additionally, we investigate k-nearest neighbors (KNN) and support vector machines (SVMs) while addressing dataset imbalance for improved model fairness. Overall, the project emphasizes a balanced approach to predicting customer credit quality, focusing on accuracy and interpretability.

Data Description and Overview

The dataset consists of 1000 data points with a total of 62 features. Out of these, 7 features (Duration, Amount, InstallmentRatePercentage, ResidenceDuration, Age, NumberExistingCredits, NumberPeopleMaintenance) are numerical and the remaining 55 are categorical. However, we found some problems present in our datset, especially the categorical features:-

- **Categorical Feature Analysis**
  - During analysis, it was observed that columns such as:
    - `Personal.Female.Single`
    - `Purpose.Vacation`

    contain only one unique value across all data points. Since these columns **do not exhibit any variance**, they do not contribute to distinguishing between different observations in the data. As a result, these features do not carry meaningful information for the model. Therefore, it would be ideal to drop these columns during preprocessing, as their inclusion does not add value to the model's predictive ability. Despite that, in this instance, the columns were retained for the credit modelling analysis.

Target Variable: The target or the dependent variable is the `Class` column. As we cannot analyse these classes keeping them as string datatypes, so for better interpretation during analysis, the target was transformed into a binary variable where:

$$\text{bad} = 0$$
$$\text{good} = 1$$

This allows for a more straightforward analysis and model training on the binary classification task.

Preprint. Under review.

Question 1

In this section, we fitted a Logistic Regression model to our dataset. It predicts the probability that a given input belongs to a particular class (good or bad). The model outputs the probability value (probability of class being `good`) and based on a chosen threshold, the model assigns the input to one of the two classes. The formula for logistic regression is as follows:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n)}}$$

Where:

- $P(Y = 1|X)$ is the probability of the target variable Y being 1 given the input X.
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients (weights) for each feature $X_1, X_2, \ldots, X_n$.
- e represents the exponential factor

This formula forms the basis for calculating the probability that a data point belongs to the "positive" class (in this case, the "good" or "bad" credit class).

This probability can be converted into class labels (e.g., 0 or 1) by setting a threshold, typically 0.5. However, thresholds like 20%, 35%, and 50% can be chosen based on the problem context, as mentioned in our project.

To classify customers as either "Good" or "Bad", we must choose a threshold. If the predicted probability of being "Good" exceeds the threshold, we classify the observation as "Good". Otherwise, it's classified as "Bad". The task requires evaluating three different thresholds: 0.20, 0.35, and 0.50.

Part-(a):-

In this part, we used our entire dataset to train the Logistic Regression model. After our model was trained, we used 30% observations of the same dataset to evaluate our model. For each threshold, we create a confusion matrix, which provides the counts of:

- **True Positives (TP)**: Correctly classified "Good" customers.
- **False Positives (FP)**: Incorrectly classified "Bad" customers as "Good".
- **True Negatives (TN)**: Correctly classified "Bad" customers.
- **False Negatives (FN)**: Incorrectly classified "Good" customers as "Bad".

		Actual	
		Positive	**Negative**
Predicted	**Positive**	# True Positives	# False Positives
	Negative	# False Negatives	# True Negatives

Based on the confusion matrix, we calculate:

- **True Positive Rate (TPR)**: The proportion of actual "Good" customers correctly identified.
- **False Positive Rate (FPR)**: The proportion of actual "Bad" customers incorrectly classified as "Good".

Comparison of Thresholds

- **Threshold 0.20**: A lower threshold classifies more customers as "Good", increasing the True Positive Rate (TPR) but also raising the False Positive Rate (FPR). This is useful when it's more important to correctly identify "Good" customers, even at the cost of misclassifying some "Bad" customers. This will reduce the False Negative Rate(FNR), which sometimes may be important, especially in credit modelling and analysis tasks.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	40	260
	Negative	4	696

Threshold 0.35:

		Actual	
		Positive	**Negative**
Predicted	**Positive**	95	205
	Negative	23	677

Threshold 0.50:

		Actual	
		Positive	**Negative**
Predicted	**Positive**	157	143
	Negative	71	629

Choosing the Best Model

Threshold 0.50 gives the best accuracy. However as we discussed before, accuracy is not a perfect measure always in these scenarios. So, taking FNR into consideration, the best model is obtained with the threshold of 0.2 (having the least FNR).

Threshold	**0.20**	**0.35**	**0.50**
Accuracy	0.73	0.77	0.78
FNR	0.09	0.19	0.31

Table 1: Comparison of Metrics for Different Thresholds

Part-(b):-

This question is similar to the previous one, with the key difference being the division of the dataset into training (70%) and test (30%) sets before training the model. The training set is used to fit the logistic regression model, and the test set is used for evaluating its performance. Below are the steps that are different from the previous question. The `train_test_split()` function is used to split the data into training (70%) and test (30%) sets. This step ensures that the model is trained on one portion of the data (training set) and evaluated on a completely separate portion (test set). This prevents the model from overfitting by learning patterns from the entire dataset.

- Model Training on Training Set: The logistic regression model is explicitly trained on X_{train} and y_{train}. This differs from the previous question, where the model was trained on the entire dataset.
- Predictions on the Test Set: Instead of predicting on the same data used for training, the model now generates predictions on X_{test} (the test set). This evaluates how well the model generalizes to unseen data, which is a critical aspect of model performance.

Comparison of Thresholds

		Actual	
		Positive	**Negative**
Predicted	**Positive**	9	82
	Negative	1	208

		Actual	
		Positive	**Negative**
Predicted	**Positive**	25	66
	Negative	10	199

		Actual	
		Positive	**Negative**
Predicted	**Positive**	44	47
	Negative	21	188

Table 2: Comparison of Confusion Matrices at Thresholds 0.2, 0.35 and 0.5 respectively

Again, threshold 0.50 is the best model in terms of accuracy and threshold of 0.2 gives the best model in terms of FNR.

Threshold	**0.20**	**0.35**	**0.50**
Accuracy	0.72	0.75	0.77
FNR	0.1	0.28	0.32

Table 3: Comparison of Metrics for Different Thresholds

Part-(c):-

The ROC (Receiver Operating Characteristic) curve is used to visualize the tradeoff between TPR and FPR across all possible thresholds. The AUC (Area Under the Curve) is a summary metric that shows the overall performance of the model. A higher AUC value indicates better model performance. An AUC value of 0.5 indicates that the model has no predictive capabilities. Both of our models have a decent predictive power as their AUC score stands at 0.83 and 0.82. The drop from 0.83 to 0.82 is because model was trained and tested on same dataset in part-(a), which led to overestimation of the AUC score.

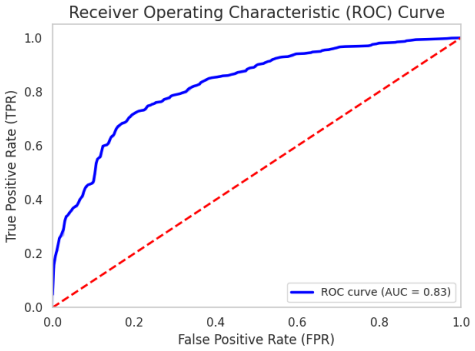

Figure 1: ROC Curve (Part (a))

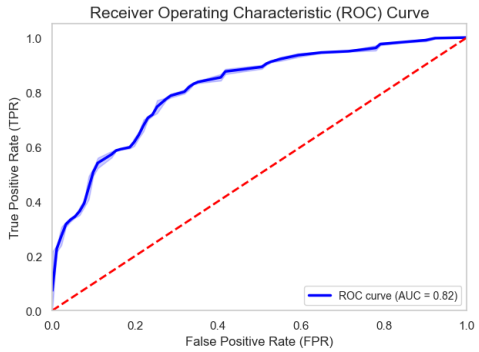

Figure 2: ROC Curve (Part(b))

We now try to also plot the Error Rate and False Negative Rates at varying thresholds. This will help us to gain a deeper understanding of the effect of Changing threshold on Accuracy or Error Rate and False Negative Rate.

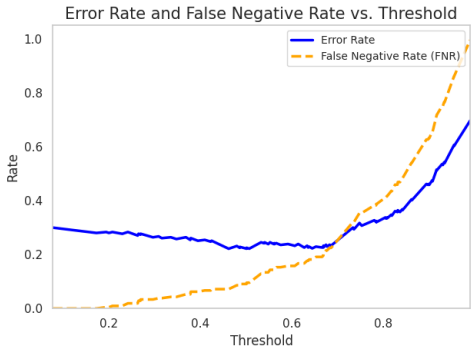

Figure 3: Error and FNR vs Threshold (Part (a))

Figure 4: Error and FNR vs Threshold (Part(b))

Question 2

Part-(a):-

- **Decision Tree Classifier:** A decision tree is fitted to the entire dataset with default hyperparameters, meaning the tree grows fully (no depth restriction), resulting in overfitting. This leads to 100% accuracy on the training data, indicating that the model has memorized the data rather than learning general patterns, which causes high variance. The diagam of the fitted tree and it's ROC curve is shown below:-

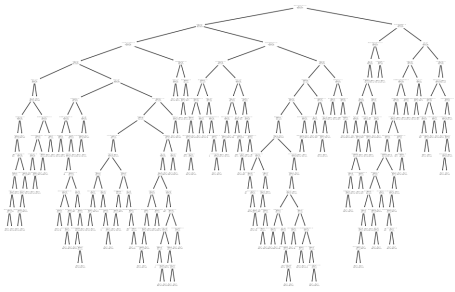

Figure 5: Tree Diagram

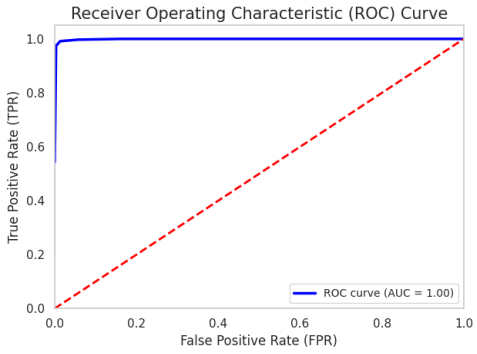

Figure 6: ROC-AUC curve

		Actual	
		Positive	**Negative**
Predicted	**Positive**	300	0
	Negative	0	700

Confusion Matrix For Decision Tree Classifier

Bagging Classifier: Bagging reduces variance by training multiple decision trees on random subsets of the data and averaging their predictions. It shows improved generalization compared to a single decision tree. However, since this model is still evaluated on the entire dataset, its performance may still be overestimated. The ROC curve shows a more reasonable but still overly optimistic performance due to a lack of test data evaluation.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	296	4
	Negative	6	694

Table 4: Confusion Matrix For Bagging

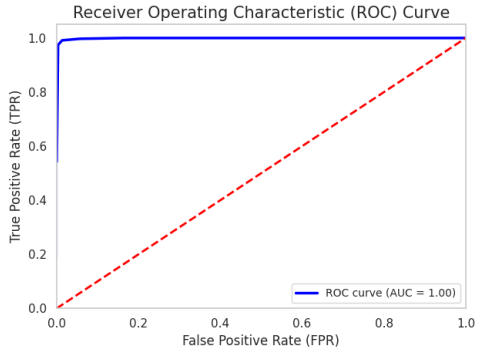

Table 5: ROC-AUC curve

Random Forest Classifier: Random Forests enhance bagging by introducing random feature selection, further reducing overfitting. While this approach improves robustness, the model's performance is still assessed on the entire dataset, leading to potentially overinflated results. The ROC curve for the Random Forest is likely more realistic, but true performance can only be judged on unseen data.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	300	0
	Negative	0	700

Table 6: Confusion Matrix For Random Forest

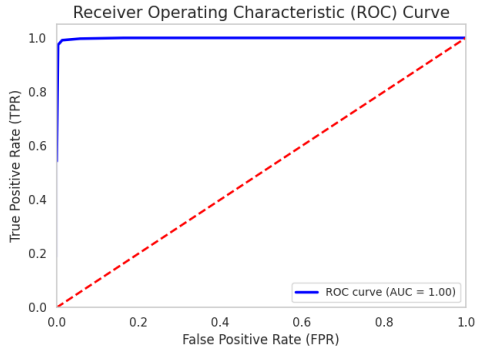

Table 7: ROC-AUC curve

Conclusion: We are likely overestimating the performance of these models because they are trained and evaluated on the same dataset. Without splitting the data into training and test sets or using cross-validation, models will appear to perform better than they would in real-world scenarios. This is visible in the ROC-AUC curve as well.

Part-(b):-

In this part, we split the dataset into 70% training and 30% test sets. The models are trained on the training data and evaluated on the test set to avoid overfitting and ensure generalizability.

- **Decision Tree Classifier:** A decision tree is fitted to the the train set and evaluated on the test set.

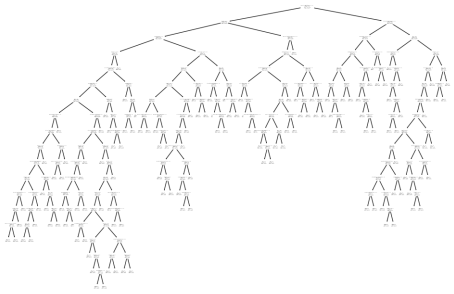

Figure 7: Tree Diagram

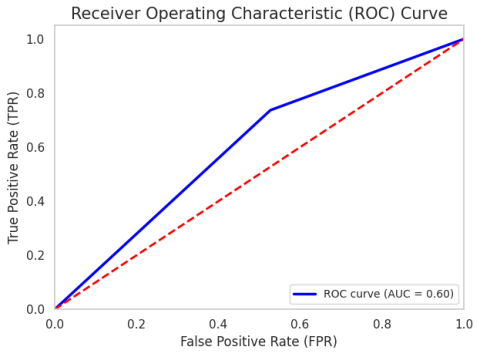

Figure 8: ROC-AUC curve

		Actual	
		Positive	**Negative**
Predicted	**Positive**	44	47
	Negative	54	155

Table 8: Confusion Matrix for Decision Tree

We then performed hyperparameter tuning with GridSearch CV and this resulted in an optimised decision tree; which also improve the performance (accuracy) by a small margin.

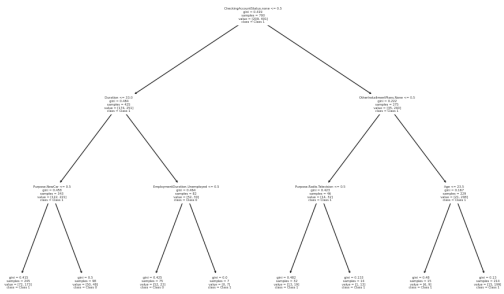

Figure 9: Optimised Decision Tree (Performing Grid Search CV and Hyperparameter tuning)

		Actual	
		Positive	**Negative**
Predicted	**Positive**	44	47
	Negative	54	155

Table 9: Confusion Matrix for Optimised Decision Tree

Bagging Classifier: Bagging trains multiple decision trees on random subsets of the training data and averages the results to reduce variance. It also predicts outcomes for the test set. The confusion matrix and ROC curve are shown side by side. Notice that we have already tuned the hyperparameters of Bagging. For more details, refer to the code.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	49	42
	Negative	43	166

Table 10: Confusion Matrix

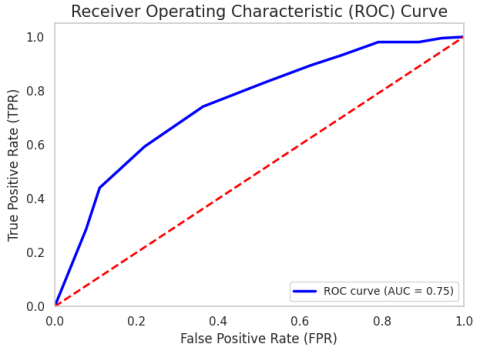

Table 11: ROC-AUC curve

Random Forest: The Random Forest classifier reduces overfitting more effectively than Bagging and single decision trees by using multiple trees and feature sampling, achieving higher predictive power while still having some variance. The confusion matrix and ROC curve are shown side by side. Notice that we have already tuned the hyperparameters of Bagging. For more details, refer to the code.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	35	56
	Negative	13	196

Table 12: Confusion Matrix

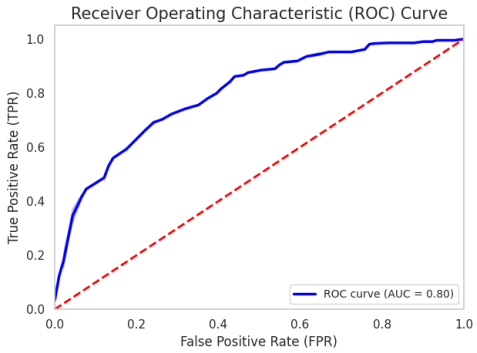

Table 13: ROC-AUC curve

Results:

In this part also, the Random Forest Classifier was the best model with the highes accuracy as well as the least False Negative Rate.

Conclusion:

We also try to plot the Error Rate and FNR vs Different thresholds. Notice that Random Forests, Decision Trees and Bagging in general, dont output probabilities. However, we can access their probabilities and then set a threshold to assign the classes, and in this way we can plot threshold vs error and FNR.

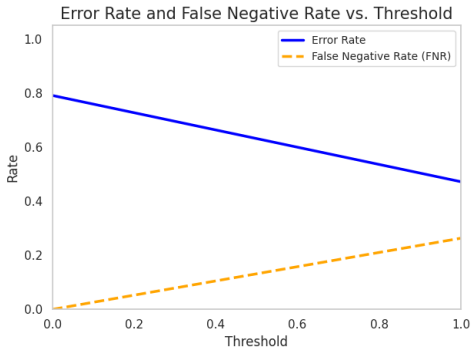

Figure 10: Error Rate vs Threshold (Decision Tree)

Figure 11: Error Rate vs Threshold (Bagging)

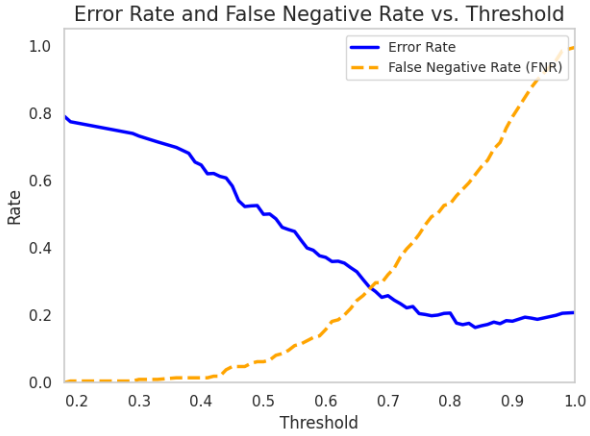

Figure 12: Error Rate vs Threshold

We would choose the Random Forest model as it has highest accuracy and lowest False Negative Rate (FNR), making it the most suitable for credit risk classification. Its ability to generalize well to unseen data and low misclassification rate of good customers as bad customers (FNR) makes it highly reliable for this task. To evaluate the three models from Question 2, two key factors are considered:

1. A lower error rate indicates a better-performing model. 2. A lower False Negative Rate reflects improved model sensitivity.

Upon comparing the three models based on these criteria, the Random Forest Classifier demonstrates the highest accuracy (i.e., the lowest error rate) and the lowest False Negative Rate. Therefore, the Random Forest Classifier is the most effective model among the three and can be considered the superior choice.

Part-(c):-

Feature Importance for Random Forest Model: In this part, the goal is to rank and plot the most important predictors for the Random Forest model, which was identified as the best-performing model in the previous section. Feature importance indicates how much each feature contributes to the model's predictive outcomes.

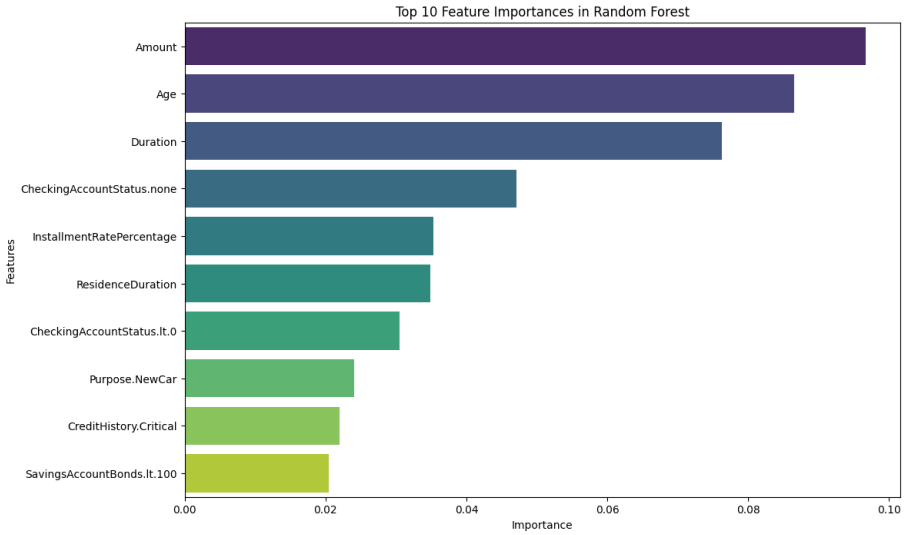

Figure 13: Feature Importance arranged in Descending Order

Conclusion: The feature importance plot highlights the most influential variables in predicting credit risk. This helps in understanding which features the Random Forest model relies on the most when making decisions. By identifying the top predictors, we can gain insights into the key drivers of customer credit risk and improve model interpretation.

Part-(d):-

Models in Question 2 is better as compared to the models in Question 1. Upon comparing the best-performing Logistic Regression model (accuracy: 0.72, FNR: 0.003) with the Random Forest model (accuracy: 0.75, FNR: 0.06), it is evident that the Random Forest model achieves higher accuracy. However, when considering the False Negative Rate (FNR), which is a critical metric in credit risk modeling, the Logistic Regression model outperforms Random Forest with a significantly lower FNR. Therefore, while Random Forest is superior in terms of accuracy, Logistic Regression emerges as the better model when False Negative Rate is prioritized.

Question 3

Part-(a):-

Standardization Technique: Before fitting the K-Nearest Neighbors (KNN) model, it is crucial to standardize the numerical variables. Two procedures are possible:

- **Procedure-I:** First, split the data into training and testing sets. Then, standardize the test data using the parameters (mean and standard deviation) learned from the training data. This approach ensures that the model is evaluated correctly without overestimating its performance.
- **Procedure-II:** Standardizing the entire dataset before splitting it into training and testing sets can lead to data leakage and overestimation of model performance.

Procedure-I is the correct procedure and we have implemented the same in our code as well. Note that we have standardized only the numerical predictors and not the categorical predictors, as per the method. Then KNN classifiers were fitted using different values for K (the number of nearest neighbors). The values tested were $K = 1$, $K = 3$, $K = 5$, and $K = 10$. We trained on the training data and evaluated our model only on the test data:-

Results Summary:

- **K = 1:** This model showed lower accuracy and a higher false negative rate, indicating sensitivity to noise in the data.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	41	50
	Negative	38	171

Table 14: Confusion Matrix For K = 1

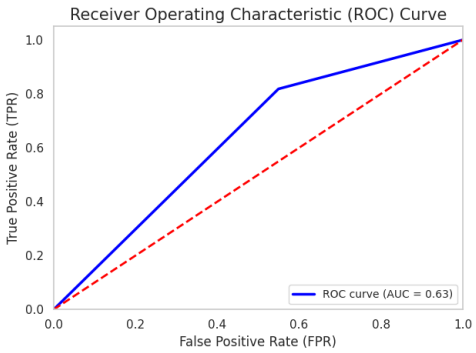

Table 15: ROC-AUC curve

KNN with K=1 has an accuracy score of: 0.71 and a False Negative Rate of: 0.123

- **K = 3:** There was a slight improvement in both accuracy and the false negative rate.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	34	57
	Negative	21	188

Table 16: Confusion Matrix For K = 3

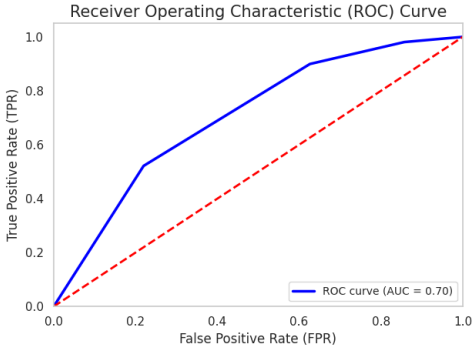

Table 17: ROC-AUC curve

KNN with K=3 has an accuracy score of: 0.727 and a False Negative Rate of: 0.073

- **K = 5:** This value provided the best performance, with an accuracy score of 0.733 and the lowest false negative rate (FNR) of 0.067, suggesting that it effectively captured the patterns in the data.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	32	59
	Negative	20	189

Table 18: Confusion Matrix For K = 5

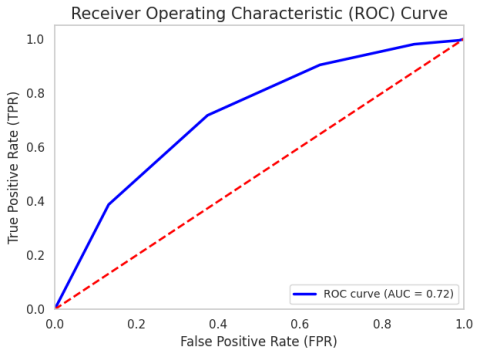

Table 19: ROC-AUC curve

KNN with K=5 has an accuracy score of: 0.733 and a False Negative Rate of: 0.067

- **K = 10:** The performance declined compared to $K = 5$, as too much averaging led to a loss of important data distinctions.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	40	260
	Negative	4	696

Table 20: Confusion Matrix For K = 10

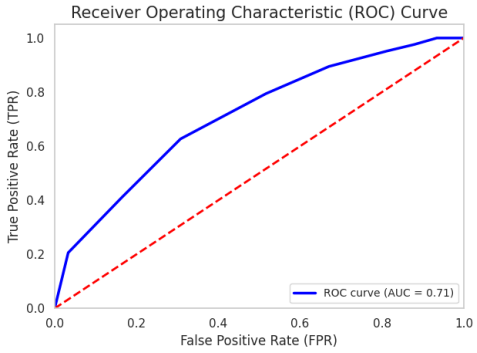

Table 21: ROC-AUC curve

KNN with K=10 has an accuracy score of: 0.72 and a False Negative Rate of: 0.077

Conclusion: So, out of the 4 different variants of KNN with K =1, 3, 5 and 10, it is clearly visible that the KNN with K=5 has the highest accuracy of 0.733 among the KNN models as well as the lowest False Negative Score of 0.067 at the same time. So, in this setting, undoubtedly, KNN with K=5 is the best model among the 4 KNN models.

Part-(b):-

Based on these metrics, the Random Forest Classifier exhibits the highest accuracy among the models evaluated. However, from the perspective of False Negative Rate, which is critical for alert systems, Logistic Regression with a threshold of 0.2 outperforms the other models, Therefore, itis the best model. The KNN models' performance was in between the Logistic Regression and Random Forest/Ensemble models.

Question 4

Part (a): SVM Classifier Performance Evaluation

In this analysis, we fitted a Support Vector Machine (SVM) classifier with a Radial Basis Function (RBF) kernel. The **numerical features** were standardized using `StandardScaler` using the same technique that we used in our KNN model. This is essential for SVM to enhance performance and ensure proper distance calculations.

Results

- **SVM Classifier:** The confusion matrix along with the ROC curve are shown below. Note that we utilised the decision boundary function of SVM and set thresholds on it for the ROC curve. This is done because SVM does not output probability.

		Actual	
		Positive	**Negative**
Predicted	**Positive**	31	60
	Negative	9	200

Table 22: Confusion Matrix

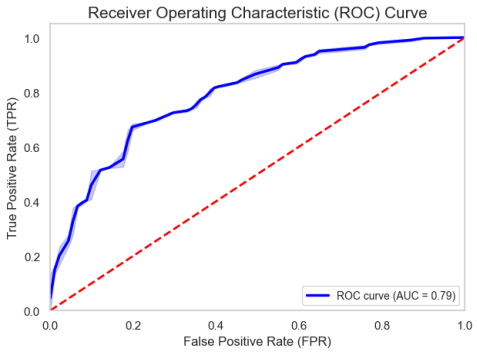

Table 23: ROC-AUC curve

Comparison with Other Models

The performance of the SVM classifier can be compared with the previously evaluated models:

- **Random Forest:** Exhibited the highest accuracy (0.776) and a low FNR (0.043).
- **Logistic Regression:** Showed good performance at various thresholds, especially at 0.2 with an FNR of 0.0033.
- **KNN:** The best model among the KNN variations was with $K = 5$, which had an accuracy of 0.733 and an FNR of 0.067.

The SVM classifier performed comparatively similar to the Random Forest model; however, random forest was a slight better than SVM. We can easily observe this from the confusion matrix.

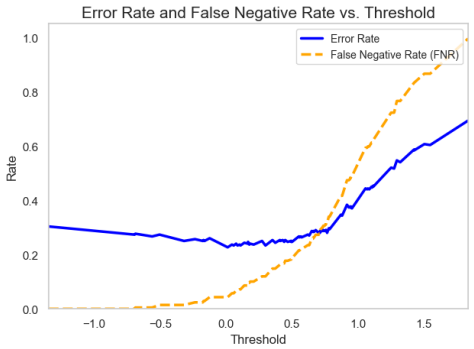

Figure 14: Error Rate vs Threshold

Conclusion

The SVM model provides an additional comparison point against the other classifiers. Depending on the accuracy and FNR results, it may offer competitive performance relative to tree-based models and logistic regression, thereby contributing valuable insights for model selection in the context of credit risk prediction.

Part-(b):-

To assess the balance of our credit dataset, we compare "Good" and "Bad" samples, revealing a significant imbalance with many more "Good" transactions. This aligns with real-world patterns, making the dataset imbalanced.

Figure 15: ROC Curve

Fitting statistical models on unbalanced data poses challenges, as they tend to favor the majority class, leading to misleading accuracy rates and poor performance on the minority class. A confusion matrix provides insights into true positives, false positives, and other metrics, but additional measures like Precision, Recall, and F1-score are essential for evaluating the minority class, particularly the False Negative Rate (FNR) in credit modeling.

Several techniques can address class imbalance:

1. **Resampling Techniques**:
    - *Oversampling the Minority Class*: This increases the representation of the minority class by duplicating or synthesizing new instances, as seen in the Synthetic Minority Over-sampling Technique (SMOTE).
    - *Undersampling the Majority Class*: This method reduces the number of instances in the majority class to achieve balance.
2. **Cost-Sensitive Learning**: This involves assigning higher misclassification costs to the minority class, penalizing the model more for incorrect predictions.
3. **Ensemble Methods**: Techniques such as Random Forest and XGBoost can handle imbalanced datasets effectively by focusing on different subsets of data, allowing the model to better learn from the minority class.

Among these methods, SMOTE is particularly effective because it creates new samples for the minority class through interpolation, providing a more balanced dataset without simple duplication. This enhances the model's ability to generalize and accurately predict the minority class while reducing the risk of overfitting. We have employed SMOTE to counter the imbalance in our dataset. After applying it, tis

By employing these techniques, we can mitigate the challenges of class imbalance, leading to improved predictive performance and more reliable model evaluations.

Work Distribution

- **Saranya Pal** - Question 1 and Question 3
- **Kamal Raj Rawat** - Question 2 and assisted in Question 4
- **Ayush Kumar Kharwar** - Hyperparamter tuning in Question 2 and Question 4