Indian Institute of Technology, Dharwad

CS209 : Artificial Intelligence
And
CS214 : Artificial Intelligence Laboratory
Project Report : **Credit Score Classification**

Course Instructor:
Dr. Dileep A.D.

Mentor Name:
Ms. Neha RP

Submitted by:
1. Prajwal N Prasad
2. Bandikatla Vivek Kumar
3. Manne Pravallika
4. Maryada Jaswika

Contents

List of Figures

List of Tables

1 Abstract

This is the project report written by Team 22, guided by Ms Neha RP. The project focuses on training and building a machine-learning algorithm for credit score classification, the aim of which is to automate and enhance credit risk management and assessment. The objective of this project is to predict credit score categories (Good, Standard and Poor) based on extensive real-world data on financial behaviour and credit history. The project involves the implementation of various machine learning algorithms including, but not limited to, Random Forest, Decision Trees, Support Vector Machine(SVM), Neural Networks, gradient boosting and Logistic Regression on top of an extensively preprocessed data frame. Model performance is evaluated using standard metrics of accuracy, recall, f1 score, precision and ROC-AUC. Several visualisation techniques, such as confusion matrices and heat maps, have also been used. The results and findings, as described below, offer key insights into the applications of machine learning in the financial domain.

2 Introduction

A *credit score* is a numerical representation of an individual's creditworthiness, commonly ranging from 300 to 850. (In our dataset, instead of a numeric estimation, there is a categorical representation, namely Good, Standard and Poor). Financial institutions such as banks and credit card companies use this score to evaluate the risk involved in lending money to a person. The score is calculated using multiple factors including payment history, amount owed, length of credit history, types of credit, and recent credit activity [4, 2].

Higher scores indicate lower credit risk, potentially leading to better financial opportunities for borrowers. Two of the most widely used credit scoring models are the FICO Score and the VantageScore, both of which use similar criteria but differ slightly in their methodologies.

3 Methodology

3.1 Model Architectures

We evaluated 10 distinct Machine Learning Models:

- Logistic Regression
- Random Forest
- KNN
- Perceptron Model
- Neural Networks
- Naive Bayes
- Decision Trees
- Gradien Boost

- Support Vector Machine
- Cat Boosting

3.2 Preprocessing

3.2.1 Handling Irrevelant Data types In the attributes

Our DataFile **train.csv** contains many missing values and mixed data types.Some columns that were expected to hold numerical values included invalid string entries, so to remove this irrelevancy, we converted all the data types that were expected to be numerical to numerical. During this conversion, any invalid entries were automatically replaced with **NaN** (Not a Number), making them easier to handle during further pre-processing.

3.2.2 Handling Missing Values

For the Features Name, Age, SSN, Occupation, AnnualIncome, MonthlyInhandSalary, NumberofLoan, TypeofLoan for the same CustomerID all of the above Attributes Should be same so we used Forward and Backward fill method to fill the missing values in it.

To handle missing values in specific features, we applied a mode-based imputation strategy by grouping the data on CustomerId. This ensures that the imputation is performed within the context of each individual customer. The features considered for this step include NumOfDelayedPayment, ChangedCreditLimit, NumCreditInquiries, OutstandingDebt, AmountInvestedMonthly, MonthlyBalance, PaymentOfMinAmount, and PaymentBehaviour. For each of these features, missing values were filled using the mode, which is the most frequently occurring value within the corresponding customer group.**This mode approach used for all features that are Categorical.**

For the Feature 'CreditHistoryAge' we group by CustomerId for each customer, we process their credit history records in order. When a missing value is encountered, we check its position within the group. If the missing value is not the first in the group, we take the previous non-missing value and fill the current missing value by adding one month to it. However, if the missing value is the first entry in the group, we look ahead to the next available non-missing value and fill the current one by subtracting one month from it. If the immediate next value is also missing, we continue this forward search until we find a valid value, then perform the imputation recursively—first filling the furthest known value by subtracting one month at each step back until all preceding missing values are filled.

3.2.3 Handling Outliers

After filling the all missing values our next step is to find the outliers in our data set.To detect outliers, we can use various statistical techniques such as the Interquartile Range (IQR) method, where values falling below the first quartile minus 1.5 times the IQR or above the third quartile plus 1.5 times the IQR are considered outliers.These outliers are filled with NaN(as missing values) in our data set.We can also visualize outliers by boxplots.

```python
def remove_outliers_iqr(df, columns):

    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        df.loc[(df[column] < lower_bound) | (df[column] > upper_bound), column] = np.nan

    return df
```

Figure 3.1: CodeSnippet For Identifying Outliers

After Filling the all the Outliers as missing value We fill all the missing values again as mentioned in **section 2.1.2**

3.2.4 Normalizing the Data

After filling all the missing values we identify all the numerical columns and we normalize it in the range **0-1**.Features with larger magnitudes can dominate the distance calculations and bias the model..

```python
def normalize_columns(df, columns):
    for column in columns:
        min_val = df[column].min()
        max_val = df[column].max()
        df[column] = (df[column] - min_val) / (max_val - min_val)

    return df
```

Figure 3.2: Code Snippet For Normalization

Normalization ensures that each feature contributes equally to the model's learning process.

3.2.5 Encoding Categorical columns

After filling all the values now its time to encode Categorical columns.We encode all categorical columns using **Label Encoding**.

```python
label_encoder = LabelEncoder()
df['Credit_Score'] = label_encoder.fit_transform(df['Credit_Score'])

categorical_cols = ['Occupation', 'Credit_Mix', 'Payment_of_Min_Amount', 'Payment_Behaviour']
le = LabelEncoder()

for col in categorical_cols:
    df[col] = le.fit_transform(df[col])
```

Figure 3.3: CodeSnippet For LabelEncoding

3.2.6 Feature Engineering

We remove irrelevant features like **ID,CustomerID,Name,Month,SSN,TypeofLoan** from our dataset.

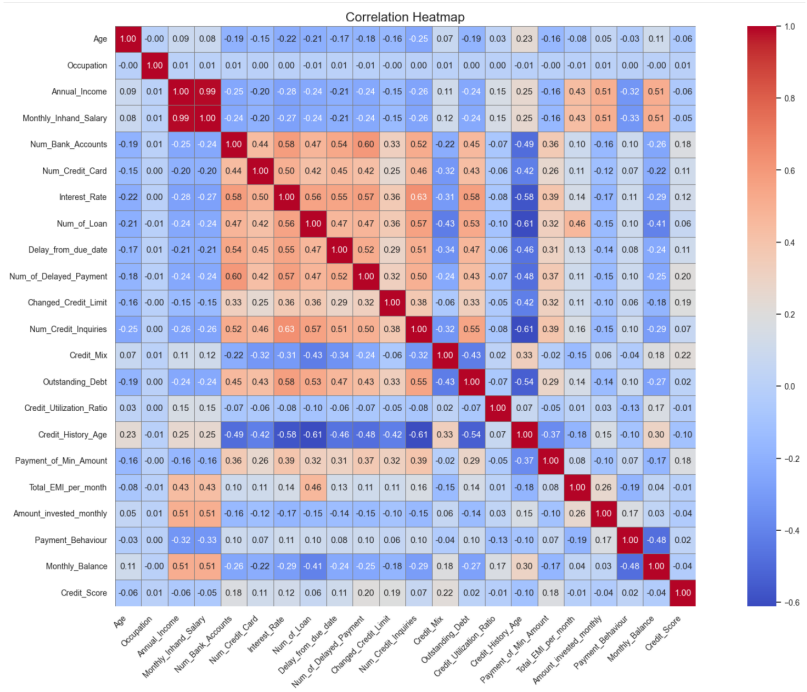

Figure 3.4: HeatMap of Corelation between Attributes

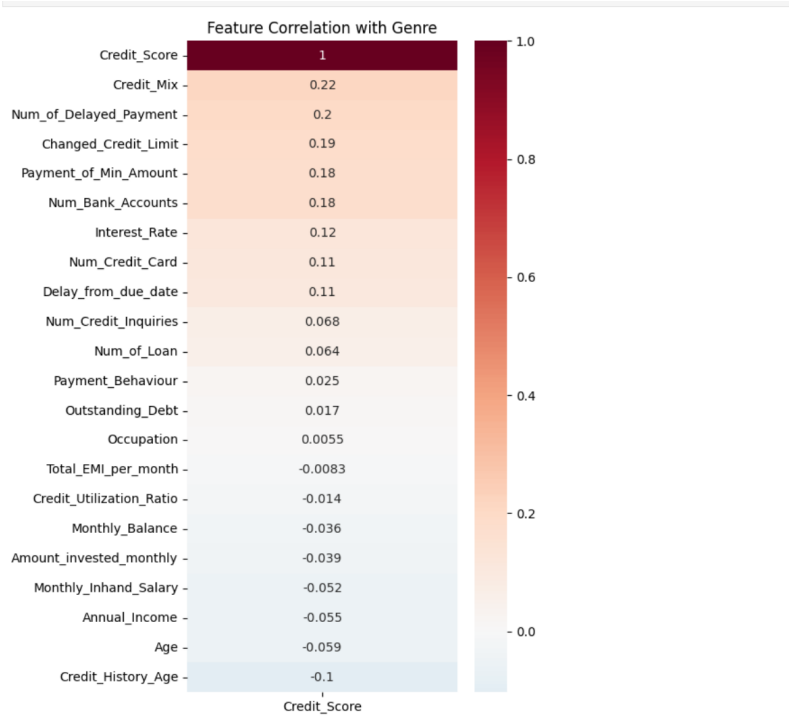

Figure 3.5: HeatMap of Corelation between Attribute and TargetClass

We remove highly corelated Features like AnnualIncome and MonhtlyInhandSalary attributes as they have a very high corelation of 0.99.

3.2.7 PCA Reduction

If we want to reduce the our Dataset we Use PCA Analysis to reduce Our Dataset to our required Dimensions.In this Project We have Reduced OUr Dataset to 10Columns.

3.2.8 Evaluation Metrics

The performance of each model was evaluated using the following metrics:

- **Accuracy:** Accuracy measures the proportion of correctly classified samples out of the total samples:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

  It gives a general measure of how well the model performs overall.

- **Precision:** Precision evaluates the proportion of correctly predicted positive cases out of all predicted positive cases:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

  This metric is critical when the cost of false positives is high.

- **Recall (Sensitivity):** Recall measures the ability of the model to identify actual positive cases:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

  It is essential in scenarios where missing positive cases is costly.

- **F1-Score:** The F1-score provides a harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

  It balances the trade-off between precision and recall, especially when classes are imbalanced.

- **Confusion Matrix:** A confusion matrix visually represents the counts of true positives, true negatives, false positives, and false negatives. It helps identify the strengths and weaknesses of the model by showing the complexity of misclassification.

4 Visualization of Data

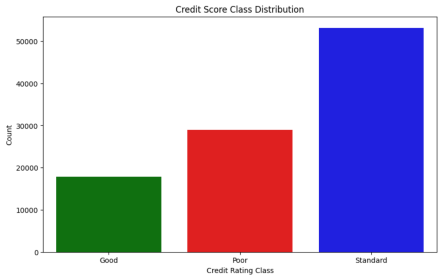

Figure 4.1: Bar plot for Distribution of CreditScore

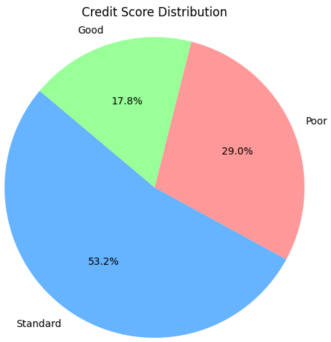

Figure 4.2: Pie Chart for Distribution of CreditScore

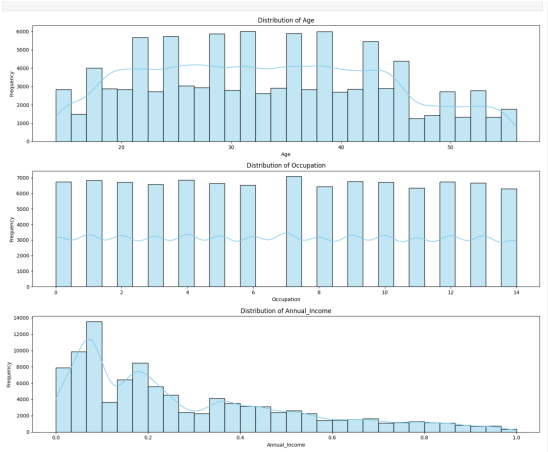

Figure 4.3: Distribution of Feature 1

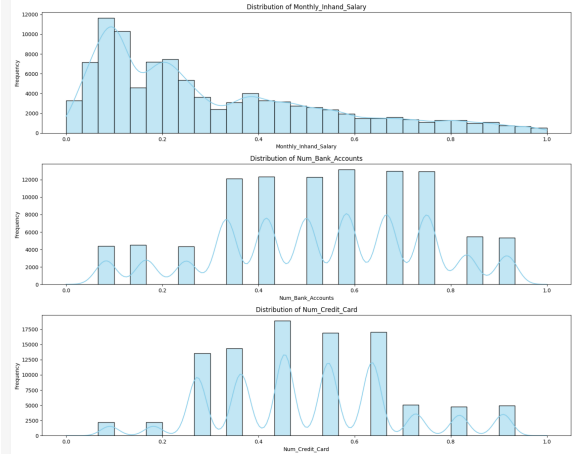

Figure 4.4: Distribution of Feature 2

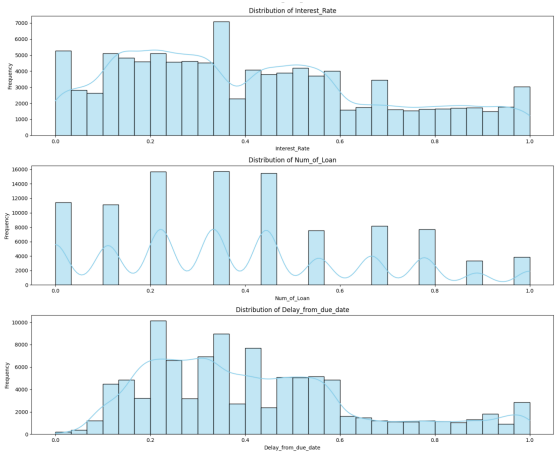

Figure 4.5: Distribution of Feature 3

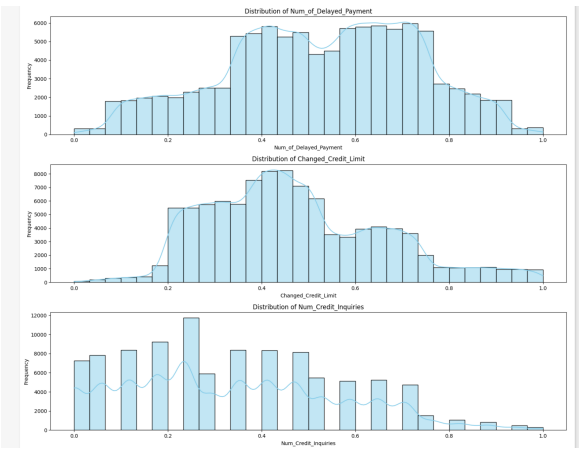

Figure 4.6: Distribution of Feature 4

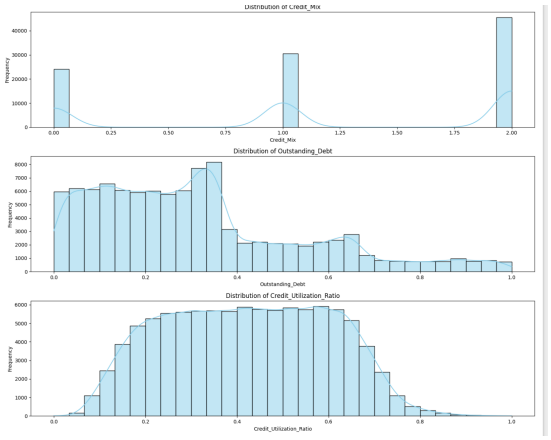

Figure 4.7: Distribution of Feature 5

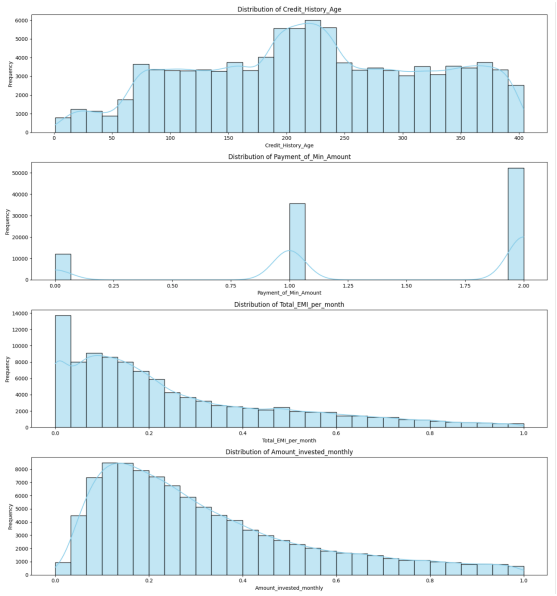

Figure 4.8: Distribution of Feature 6

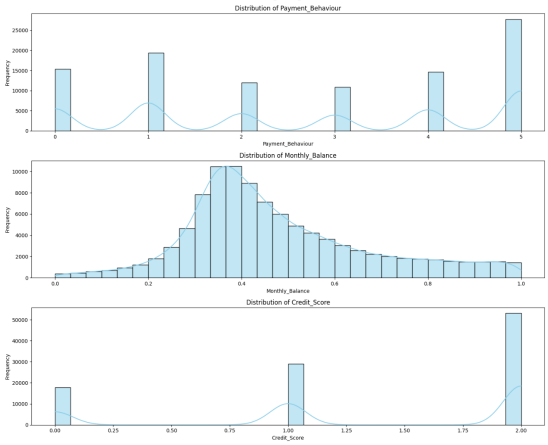

Figure 4.9: Distribution of Feature 7

5 Model Implementation

[9]

5.1 Logistic Regression

5.1.1 Model Description

In this model, we used a logistic regression to classify credit scores into three categories: **Good, Poor, and Standard**. The features were standardized using StandardScaler, and the data was split into training and testing sets. Logistic regression, models the probability of each class using a linear combination of the input features. Since this is a multi-class classification problem, the logistic regression model internally uses the **softmax function** to compute the probability distribution across the three classes. This

allows the model to assign a probability score to each class label for a given input. After training, the model was evaluated using metrics such as **accuracy, precision, recall, F1-score, ROC AUC, and a confusion matrix**, providing a comprehensive assessment of performance. The use of classweight='balanced' helped to handle the class imbalance by adjusting weights inversely proportional to class frequencies. The softmax function is defined as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Where:

- z_i is the score (logit) for class i
- K is the total number of classes

This function outputs a **probability distribution** over all classes — exactly what `predict_proba()` gives you.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	64.5950	63.1999	68.5855	63.9020	79.8852
Logistic Regression(PCA)	62.06	60.79	65.7633	61.3065	78.0142

Table 5.1: Performance Metrics

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	80.49	48	80	60	86.42
Poor	67.65	63	68	65	80.24
Standard	57.61	79	58	67	73.00

Table 5.2: Class-Wise Metrics For Logistic Regression (Original Dataset)

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	78.79	45	79	57	84.69
Poor	61.92	62	62	62	78.93
Standard	56.58	75	57	65	70.42

Table 5.3: Class-Wise Metrics For Logistic Regression (PCA reduced Dataset)

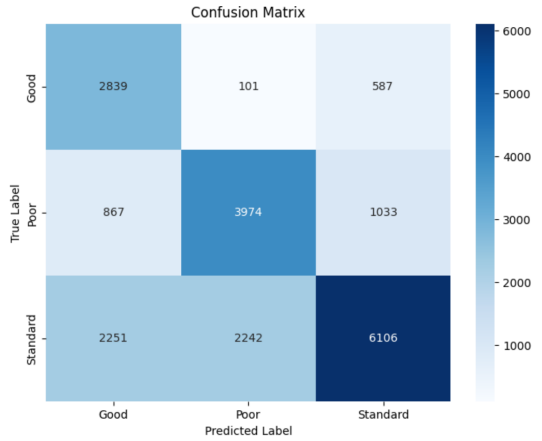

Figure 5.1: Confusion Matrix for Original Dataset for Logistic Regression

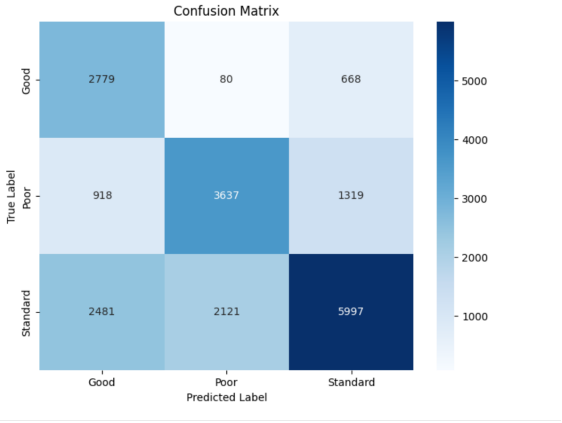

Figure 5.2: Confusion Matrix for Reduced Dataset for Logistic Regression

5.1.2 Observation

In this Logistic Regression Model

- Applying PCA to logistic regression resulted in a consistent degradation across all metrics. Accuracy dropped by 2.5% (64.60% → 62.06%), while precision, recall, and F1-score each declined by 2–3%. The ROC-AUC also decreased by 1.87%, indicating reduced discriminative power. This suggests that PCA, while reducing dimensionality, may have eliminated features critical for classification, particularly for the multi-class problem addressed by softmax regression.
- This Suggests that Applying Logistic Regression on original Dataset yields Better results Than performing On PCA Reduced Data Set

5.2 Random Forest

[8]

5.2.1 Model Description

Random Forest classifier was applied to the dataset where the input features (X) and the target variable (y) were separated from the dataset using column indexing.To handle class imbalance in the training data, the Synthetic Minority Over-sampling Technique (SMOTE) was Used.This method synthetically generates new instances of the minority class to balance the class distribution.The balanced training data was then used to train a Random Forest classifier with 100 decision trees (nestimators=100). After training, Evaluation is done on test dataset.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	81.48	79.9601	81.8182	80.7937	92.518
Random Forest (PCA)	69.9250	67.619	71.92	68.67	85.14

Table 5.4: Performance Metrics for Random Forest

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	80.29	75	80	78	95.72
Poor	85.53	79	86	82	94.23
Standard	79.63	85	80	82	87.60

Table 5.5: Class-Wise Metrics For Random Forest (Original Dataset)

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	77.69	52	78	62	89.05
Poor	71.76	71	72	71	86.89
Standard	66.33	80	66	73	79.48

Table 5.6: Class-Wise Metrics For Random Forest (PCA reduced Dataset)

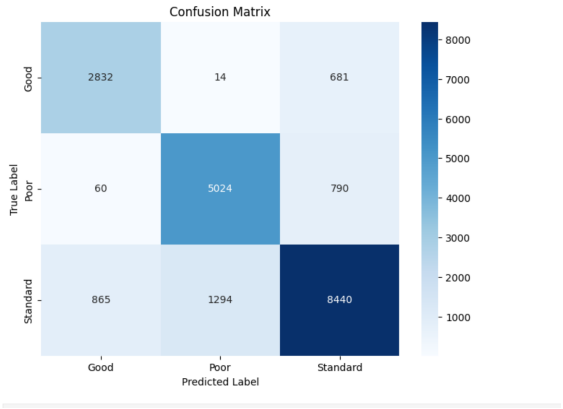

Figure 5.3: Confusion Matrix for Original Dataset for Random Forest Model

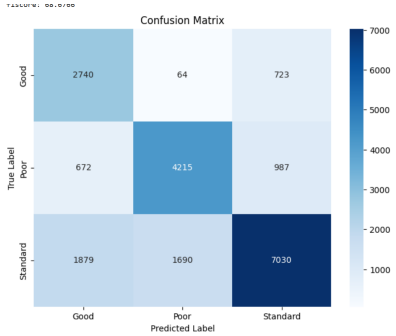

Figure 5.4: Confusion Matrix for Reduced Dataset for Random Forest Model

5.2.2 Observations

We observed several key points after evaluating the Random Forest classifier models:

- **Parameter Tuning:** The Random Forest classifier's performance was highly influenced by the selection of hyper-parameters, particularly the number of trees and feature subsets. We found that a random subset of features in each tree reduced overfitting and improved the generalization of the model. Additionally, PCA reduction helped in reducing the dimensionality while maintaining performance.
- **Class-wise Performance:** The model showed a high recall for Class Poor,with precision of 85% for Class Standard.
- **Dimensionality Reduction:** Using PCA our random forest didn't perform well in all aspects.This suggests that Random Forest can handle high-dimensional features effectively, but dimensionality reduction decrease accuracy.

5.3 K-Nearest Neighbours(KNN)

5.3.1 Model Description

KNN is used to classify a new data point based on the majority label of its closest neighbors in the training dataset. When a prediction is required, the algorithm calculates the distance between the new point and all other points in the training data (typically using Euclidean distance). It then identifies the 'K=5' closest data points—where 'K' is a predefined constant—and assigns the most common class among those neighbors to the new point. KNN is considered a lazy learning algorithm because it does not require any training or model building; instead, it stores the entire training dataset and makes predictions during the inference stage. While this makes the algorithm intuitive and easy to implement, it can be computationally expensive for large datasets.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
KNN	73.6650	71.4201	72.27	71.8174	59.7731
KNN (PCA)	66.15	63.1162	63.5266	63.1619	77.965

Table 5.7: Performance Metrics for KNN

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	67.76	63	68	65	68.46
Poor	73.17	74	73	73	58.72
Standard	75.90	78	76	77	52.14

Table 5.8: Class-Wise Metrics For KNN (Original Dataset)

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	56.76	52	57	54	82.77
Poor	62.55	66	63	64	79.31
Standard	71.27	71	71	71	71.82

Table 5.9: Class-Wise Metrics For KNN (PCA reduced Dataset)

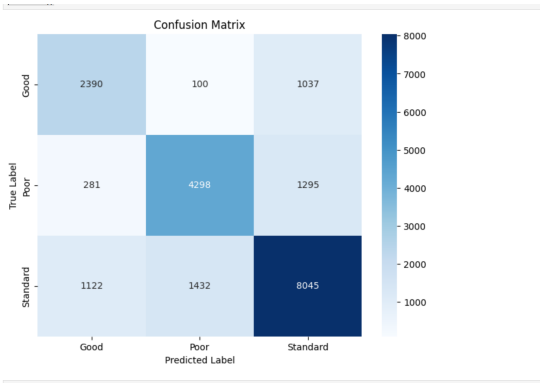

Figure 5.5: Confusion Matrix for Original Dataset for KNN Model

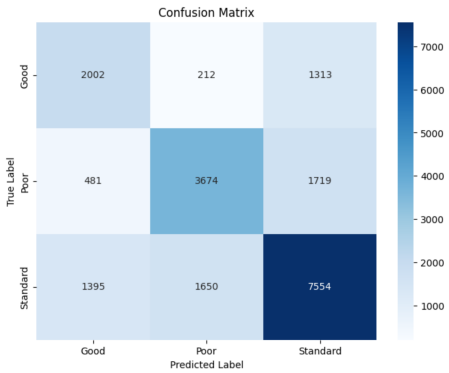

Figure 5.6: Confusion Matrix for Reduced Dataset for KNN Model

5.3.2 Observations

During hyperparameter tuning, the performance of KNN was evaluated for various values of k: 2,3, 5, 7, 11, 13, 17, 19, 23, 29, and 31.

- The optimal K value where we get Maximum accuracy is at k='5'.After this value Accuracy will decrease.The Precision and Recall is also maximum at this k value. When this model is applied to PCA reduced data set The optimal K value is still 5 Where the accuracy is maximum,The Precision and Recall is also maximum at this K value.

5.4 Perceptron Model

[11]

5.4.1 Model Description

Perceptron is one of the simplest and oldest machine learning algorithms used for binary and multiclass classification tasks. It is a type of linear classifier that makes predictions based on a weighted sum of the input features, followed by an activation function. In this project, we used the Scikit-learn Perceptron model, which is an implementation of the stochastic gradient descent (SGD) version of the algorithm. The Perceptron iteratively adjusts weights using gradient-based learning by minimizing the number of misclassified examples. Although it does not provide probability estimates like logistic regression, it is computationally efficient and effective for linearly separable data.Since our data has multiclass and Perceptron algorithm was designed for binary classification, the Perceptron implementation in scikit-learn supports multiclass classification by using a strategy called "one-vs-rest" (OvR) behind the scenes.

Model	Accuracy	Precision	Recall	F1-Score
Perceptron	58.72	58.1595	43.9543	40.8457
Perceptron (PCA)	58.35	51.06	50.49	49.32

Table 5.10: Performance Metrics for Perceptron

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
Good	67.76	63	68	65
Poor	73.17	74	73	73
Standard	75.90	78	76	77

Table 5.11: Class-Wise Metrics For Perceptron (Original Dataset)

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
Good	14.2	33	14	20
Poor	71.57	59	72	65
Standard	65.71	61	66	63

Table 5.12: Class-Wise Metrics For Perceptron (PCA reduced Dataset)

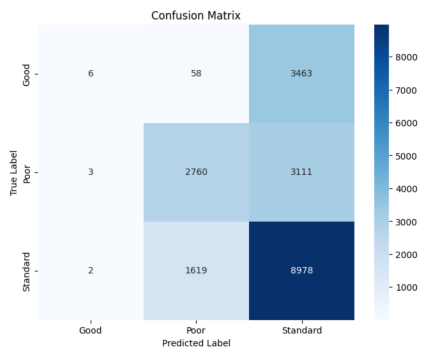

Figure 5.7: Confusion Matrix for Original Dataset for Perceptron Model

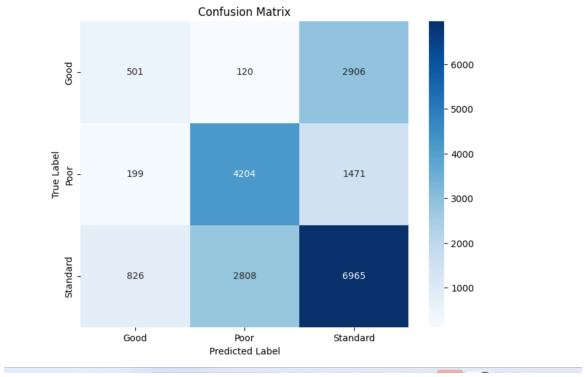

Figure 5.8: Confusion Matrix for Reduced Dataset for Perceptron Model

5.4.2 Observations

Using This model

- We got the Maximum Accuracy for the Class Poor and precision is highest for Standard and F1-score is highest For the Class Standard.
- Even for the PCA reduced Data all the performance metrics is very less as compared to original Data set.This Tells us the that the Model is performing Better on The original Dataset than the PCA reduced data set.

5.5 Neural Networks

[5]

5.5.1 Model Description

Neural Networks are powerful models which classify a new data point based on numerous layers (One input layer, one output layer, and several hidden layers). Individual elements of each layer, called nodes are interconnected and process raw data. They then apply an activation function(softmax function in our case) and pass the result to the next layer. Neural networks excel in handling complex, high-dimensional data, making them suitable for tasks such as image classification, speech recognition, and natural language processing. However, they require substantial computational resources and a large amount of labeled data for effective training.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Neural Network	70.06	68.29	65.89	66.89	82.26
Neural Network(PCA)	68.73	66.765	66.50	66.14	81.42

Table 5.13: Performance Metrics

Class	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC AUC (%)
Good	70.65	58.49	64.24	61.23	87.42
Standard	63.38	74.03	65.38	68.29	81.24
Bad	76.51	73.08	75.52	74.76	78.78

Table 5.14: Per-Class Performance Metrics for Neural Network(Original Dataset)

Class	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	ROC AUC (%)
Good	62.96	58.49	54.12	58.21	84.66
Standard	58.97	72.98	58.97	65.23	81.86
Bad	75.18	71.57	75.18	73.33	77.68

Table 5.15: Per-Class Performance Metrics for Neural Network(PCA Reduced Dataset)

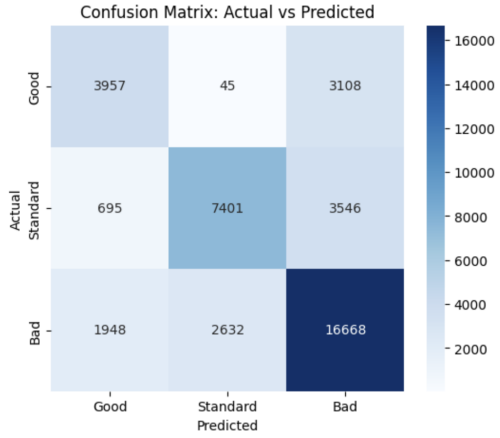

Figure 5.9: Confusion Matrix for Original Dataset for Neural Networks

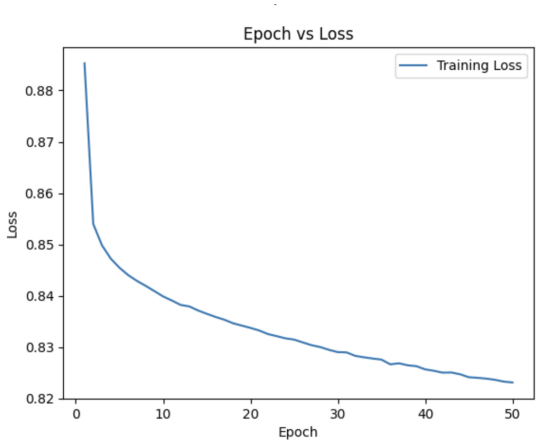

Figure 5.10: Epochs vs Loss for Original Dataset

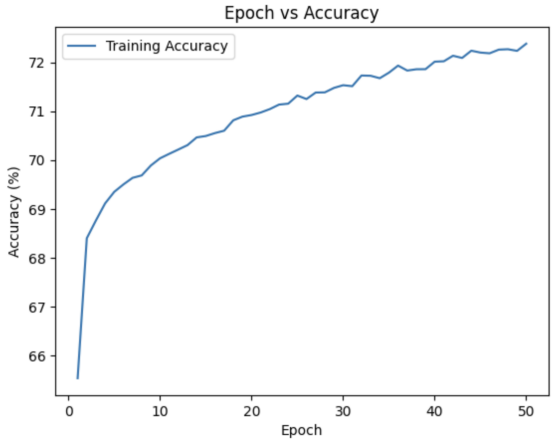

Figure 5.11: Epochs vs Accuracy for Original Dataset

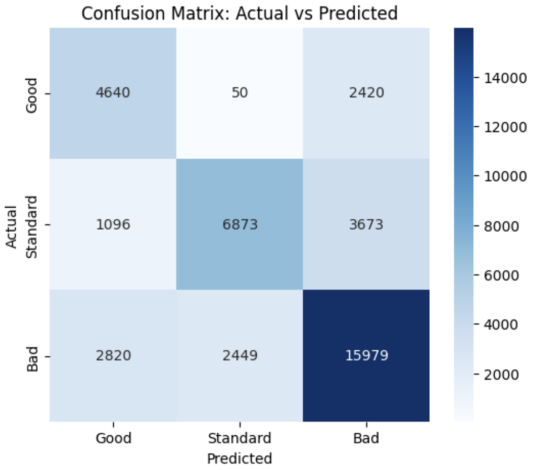

Figure 5.12: Confusion Matrix for Reduced Dataset for Neural Networks

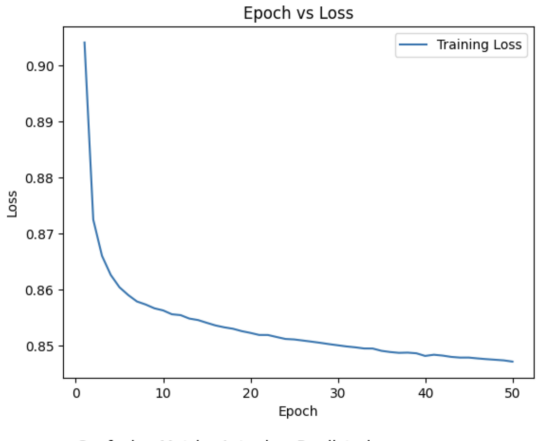

Figure 5.13: Epochs vs Loss for Reduced Dataset

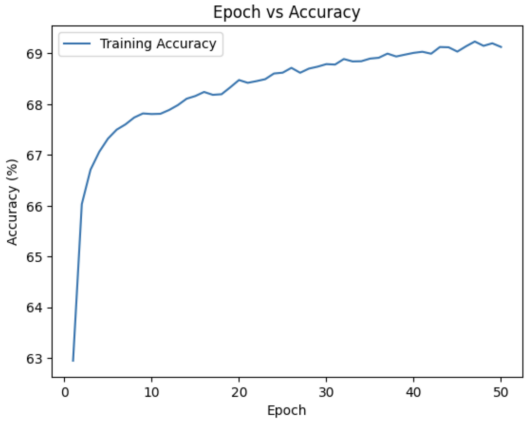

Figure 5.14: Epochs vs Accuracy for Reduced Dataset

5.5.2 Observations

- The accuracy is maximum for the Good class and the Precision is highest for the Standard Class, while the F1-Score is maximum for the Poor class.
- While the original dataset shows higher metric values, the PCA values are still early equal, showing that the neural network model has almost equal metrics for both PCA Reduced and the original dataset.

5.6 Naive Bayes

[10]

5.6.1 Model Description

Naive Bayes is a probabilistic machine learning algorithm based on applying Bayes' Theorem with a strong assumption of independence between features. It calculates the probability of each class,given the input features and predicts the class,with the highest posterior probability. In this project, we used the Gaussian Naive Bayes variant,

which assumes that the features follow a normal distribution. This model is particularly efficient for high-dimensional datasets. Despite assumption of feature independence, it often performs surprisingly well in classification tasks. Although Naive Bayes doesn't allow for extensive tuning or regularization, it provides a strong baseline for comparison against more complex models.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Naive Bayes	62.66	69.6921	62.66	62.9239	77.32
Naive Bayes (PCA)	61.58	66.04	61.58	62.0716	77.85

Table 5.16: Performance Metrics for Naive Bayes

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	83.19	44	83	58	84.96
Poor	72.66	62	73	67	76.41
Standard	50.29	83	50	62	70.26

Table 5.17: Class-Wise Metrics For Naive Bayes (Original Dataset)

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC-AUC(%)
Good	80.46	44	80	57	84.37
Poor	61.41	63	61	62	78.39
Standard	55.39	75	55	64	70.79

Table 5.18: Class-Wise Metrics For Naive Bayes (PCA reduced Dataset)

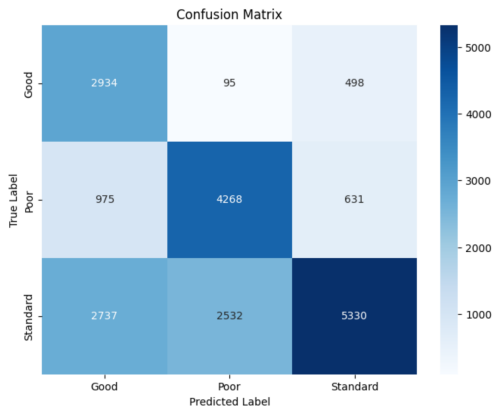

Figure 5.15: Confusion Matrix for Original Dataset for Naive Bayes Model

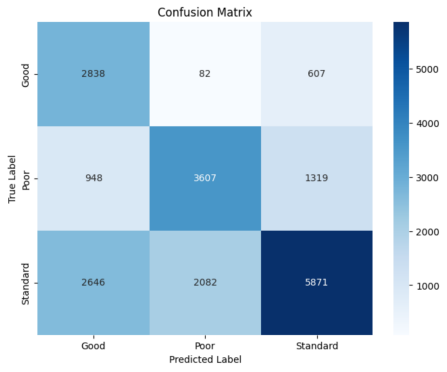

Figure 5.16: Confusion Matrix for Reduced Dataset for Naive Bayes Model

5.6.2 Observation

Using Naive Bayes Model

- The Naive Bayes classifier shows minimal degradation when PCA is applied. Overall accuracy drops slightly from 62.66% to 61.58%, while precision sees a more

noticeable decline from 69.69% to 66.04%. Interestingly, the ROC-AUC improves marginally from 77.32% to 77.85%, suggesting that PCA may help the model better discriminate between classes despite the minor drop in other metrics. This indicates that Naive Bayes is relatively robust to dimensionality reduction compared to logistic regression.

- Naive Bayes model performs reasonably well, particularly for the "Good" and "Poor" classes, but struggles with the "Standard" class, likely due to feature overlap. The minimal impact of PCA suggests it could be used to improve computational efficiency without significant performance loss.

5.7 Decision Trees

[7]

5.7.1 Model Description

Decision Trees model is a powerful Machine Learning classification algorithm, especially for linearly non separable classes. It works on the principle of trees, where each internal node is a condition and each leaf is a class label. The branches are the outcome of the test. A passed test takes one branch, and a failed test takes another. Decision trees are useful because there is no need for scaling the data and that it can handle both numeric and categorical data. However, decision trees can be prone to overfitting, especially when they grow deep and complex. Techniques such as pruning, setting maximum depth, or using ensemble methods like Random Forests and Gradient Boosted Trees can mitigate this issue.

Decision Trees are widely used in applications such as medical diagnosis, credit scoring, and decision support systems due to their transparency and ease of use.

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
Decision Tree	71.81	70.10	70.29	70.19	77.12
Decision Tree (PCA)	59.35	55.77	55.94	55.85	66.59

Table 5.19: Performance Metrics for Decision Tree

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC AUC(%)
Good	65.56	65.65	66.70	65.65	79.19
Standard	70.91	71.34	71.34	71.34	71.34
Poor	74.40	75.92	74.77	75.12	73.44

Table 5.20: Class-Wise Metrics For Decision Trees (Original Dataset)

Class	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)	ROC AUC(%)
Good	46.15	45.40	46.15	46.15	67.98
Standard	56.19	56.70	56.70	56.70	69.55
Poor	65.49	66.57	65.49	66.82	64.13

Table 5.21: Class-Wise Metrics For Decision Trees (PCA reduced Dataset)

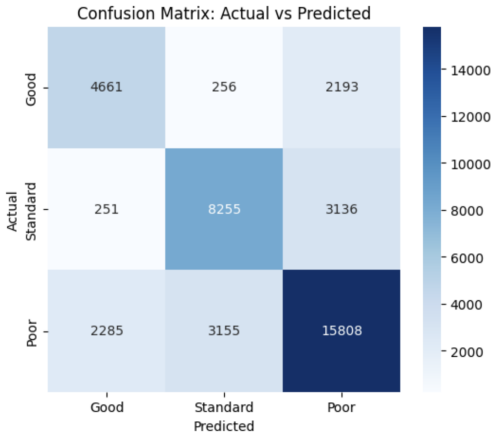

Figure 5.17: Confusion Matrix for Original Dataset for Decision Trees

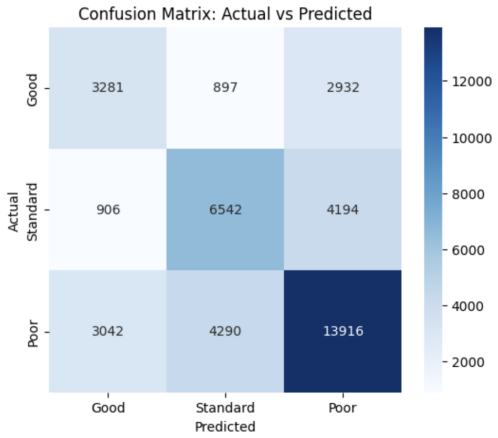

Figure 5.18: Confusion Matrix for Reduced Dataset for Decision Trees

5.7.2 Observations

- The decision trees model shows a significant difference in metric scores between the original dataset and the PCA reduced dataset. This indicates that this model is not sensitive to dimensionality reduction, unlike neural networks and Naive Bayes.
- The model struggles a bit with the Good class, and performs reasonably well with the Standard and Poor class. This is likely because Decision Trees are sensitive to class imbalance, and the Good class has the least number of training examples in the dataset.

5.8 Gradient Boost

[6]

5.8.1 Model Description

Gradient Boost is a powerful machine learning algorithm that can be used for both classification and regression tasks. It is an ensemble algorithm. This means that it combines many weaker algorithms to create a more accurate results. Specifically, Gradient Boosting creates a series of decision trees, each learning to correct the errors made by the previous trees in the sequence.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Gradient Boost	70.70	68.45	69.35	68.73	85.12
Gradient Boost(PCA)	66.48	63.86	63.44	63.49	82.18

Table 5.22: Performance Metrics

Class	Accuracy	Precision	Recall	F1-Score	ROC AUC
Good	67.55	0.58	0.68	0.62	0.89
Standard	66.35	0.73	0.66	0.69	0.87
Poor	74.14	0.75	0.74	0.74	0.80

Table 5.23: Class-Wise Metrics For Gradient Boost (Original Dataset)

Class	Accuracy	Precision	Recall	F1-Score	ROC AUC
Good	56.82	0.52	0.57	0.54	0.86
Standard	60.51	0.70	0.61	0.65	0.84
Poor	72.99	0.70	0.73	0.72	0.77

Table 5.24: Class-Wise Metrics For Gradient Boost (PCA Reduced Dataset)

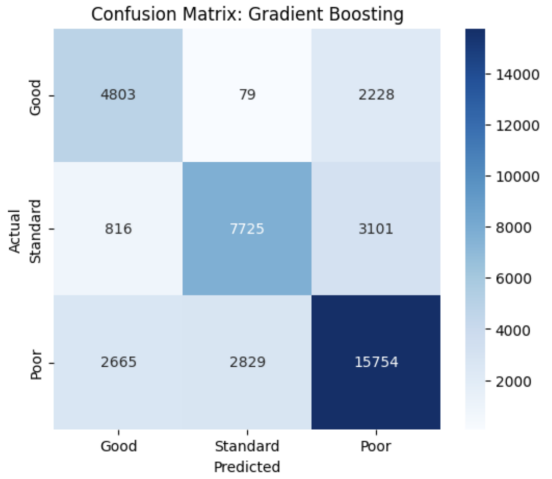

Figure 5.19: Confusion Matrix for Original Dataset for Gradient Boost

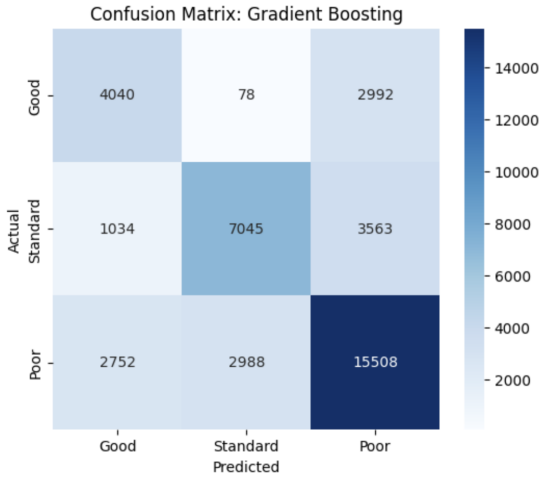

Figure 5.20: Confusion Matrix for Reduced Dataset for Gradient Boost

5.8.2 Observations

- Gradient Boosting is based on decision trees and combines multiple of them to create a stronger model. Hence, we expect the trends to follow the decision tree model, while giving better metrics.
- Following the trend, the good class has slightly lower precision and F1-score values as compared to the other two, because of fewer training examples, though the difference in percentage gap is significantly better.
- The huge difference in performance between PCA reduced and the original dataset is much better in gradient boosting as compared to decision trees. However, following the trend of tree-based algorithms, PCA still underperforms when compared to the original dataset.

5.9 Support Vector Machine(SVM)

[1]

5.9.1 Model Description

Support Vector Machine is a machine Learning algorithm used for classification of both linear and non-linear classification problems. It works on the principle of finding the optimal hyperplane separating two classes. When data is not linearly separable, it employs a kernel trick to transform the data to a higher degree where it is separable and then forms a hyperplane in that degree.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Support Vector Machine	66	65	70	65	79

Table 5.25: Performance Metrics

Class	Accuracy	Precision	Recall	F1-Score	ROC AUC
Good	82.56	0.48	0.83	0.60	0.86
Standard	69.31	0.64	0.69	0.66	0.79
Bad	58.15	0.82	0.58	0.68	0.73

Table 5.26: Class-Wise Metrics For SVM (Original Dataset)

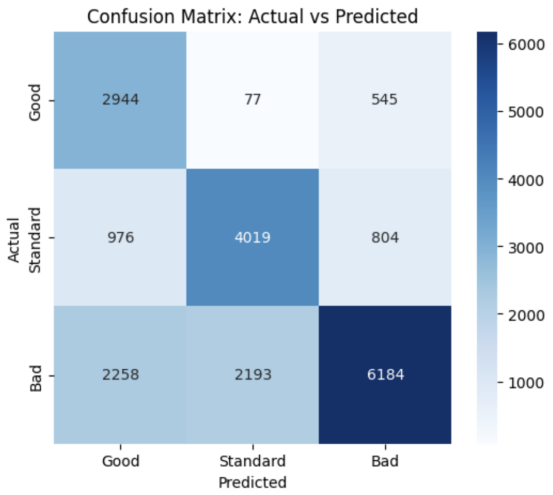

Figure 5.21: Confusion Matrix for Original Dataset for SVM

5.9.2 Observations

- The SVM Model takes a huge amount of time to run, especially for larger datasets because they run in $O(n^2)$ time.
- The Good class has the best accuracy with high recall and low precision. This indicates that the model is overly predicting the good class. This could be because of significant overlap between the good and standard classes which affect the formation of the hyperplane and thus the metrics.

5.10 Catboosting

[3]

5.10.1 Model Description

Category Boosting(CatBoosting) is an open-source gradient boosting library for machine learning. It is based on gradient boosting with the added advantage that it uses ordered boosting. Ordered boosting is a technique that prevents overfitting. It also handles categorical data natively, meaning that there is no need to manually preprocess it.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
CatBoost	70.97	68.65	69.82	69.05	85.36
CatBoost(PCA)	66.75	64.17	64.19	63.97	82.57

Table 5.27: Performance Metrics

Class	Accuracy	Precision	Recall	F1-Score	ROC AUC
Good	68.65	0.5804	0.6865	0.6290	0.8926
Standard	66.76	0.7255	0.6676	0.6954	0.8676
Poor	74.05	0.7537	0.7405	0.7470	0.8007

Table 5.28: Class-Wise Metrics For CatBoost (Original Dataset)

Class	Accuracy	Precision	Recall	F1-Score	ROC AUC
Good	59.25	0.5137	0.5925	0.5503	0.8663
Standard	60.81	0.7019	0.6081	0.6516	0.8410
Poor	72.52	0.7097	0.7252	0.7173	0.7698

Table 5.29: Class-Wise Metrics For CatBoost (PCA Reduced Dataset)

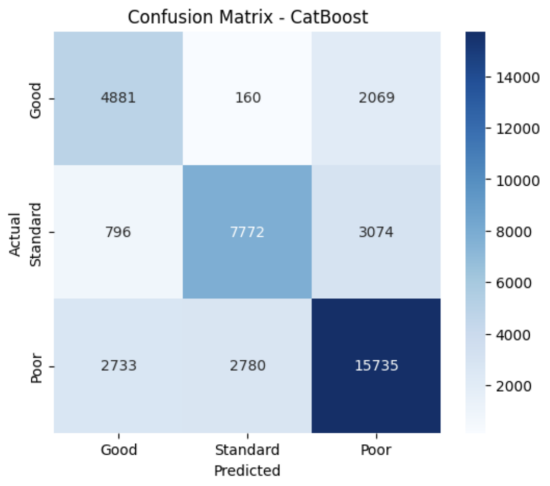

Figure 5.22: Confusion Matrix for Original Dataset for CatBoosting

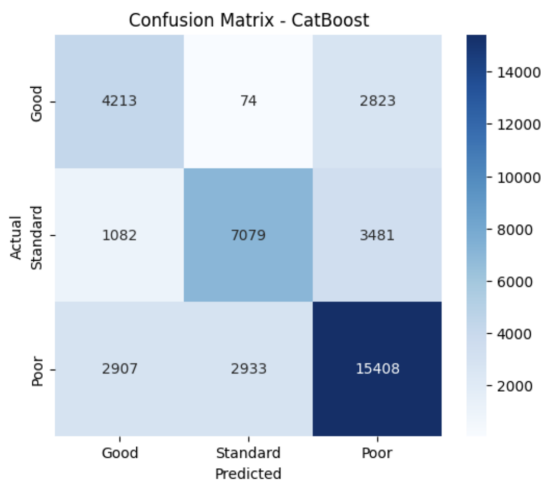

Figure 5.23: Confusion Matrix for Reduced Dataset for CatBoosting

5.10.2 Observations

- CatBoosting is a more efficient and accurate version of gradient boosting. Hence, we expect the same trends to continue in CatBoosting as was in gradient boosting, but with better metrics.
- We observe that the good class performs slightly better in the precision and F1-scores in CatBoosting when compared to gradient boosting, but they are still significantly low compared to the other classes.
- The difference in the PCA reduced vs Original dataset metrics has further reduced to $< 3\%$.

6 Comparison of Models

This Section provides a Comprehensive comparison of all the evaluated models,considering their performance across key metrics such as Accuracy,Precision,Recall,f1 Score.

6.0.1 Bar Plots For Metric Comparison

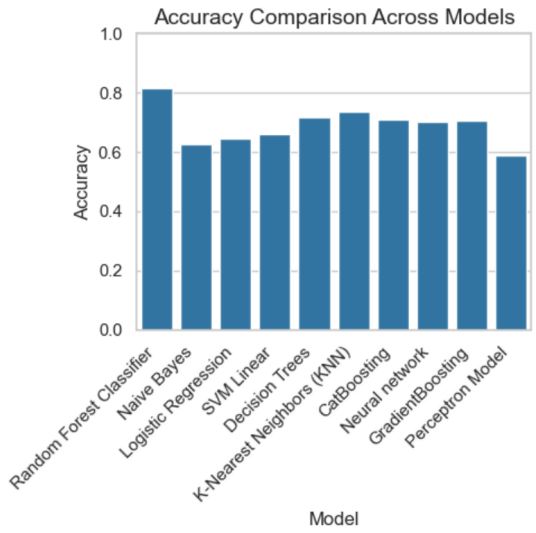

Figure 6.1: Accuracy of All Models

Figure 6.1 illustrates the accuracy comparison across various classification models. Among all the models, the Random Forest,Neural Network, Gradient Boosting, and CatBoosting models demonstrate the highest accuracy.This indicates their strong ability to correctly classify both positive and negative samples. Models such as Logistic Regression, SVM Linear, and Decision Trees show moderate performance, with accuracy values ranging between 0.65 and 0.75. On the other hand, the Perceptron Model and Naive Bayes exhibit comparatively lower accuracy, indicating its inability in capturing complex patterns in the data.

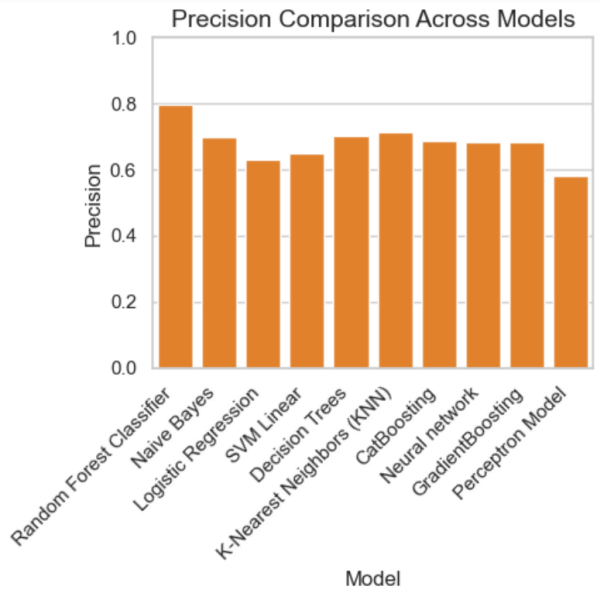

Figure 6.2: Precision of All Models

Figure 6.2 represents the precision scores for all the models. Precision indicates the proportion of true positives among all predicted positives, reflecting the model's ability to avoid false positives. The Random Forest Classifier and Neural Network achieve the highest precision, implying that their positive predictions are highly reliable. Gradient Boosting, CatBoosting, and K-Nearest Neighbors (KNN) also maintain competitive precision values, slightly below 0.8. In contrast, the Perceptron Model and Naive Bayes again perform poorly in terms of precision, indicating a relatively higher false positive rate.

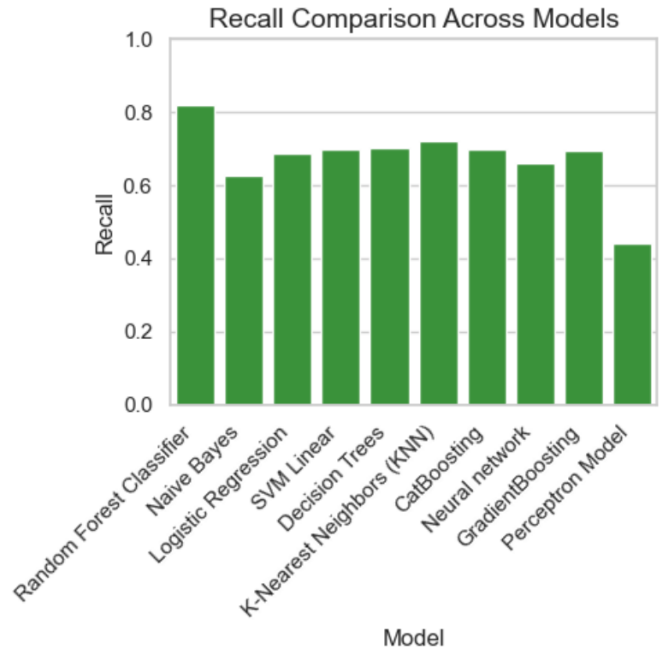

Figure 6.3: Recall of All models

Figure 6.3 represents the Recall scores for all the models.Recall measures the ability

of a model to identify all relevant instances (true positives) in a dataset.Among the all Models Random Forest has the highest Recall compared to all other models indicating its ability at minimizing false negatives.Perceptron has the lowest Recall among all the models showing its inability at minimizing errors.

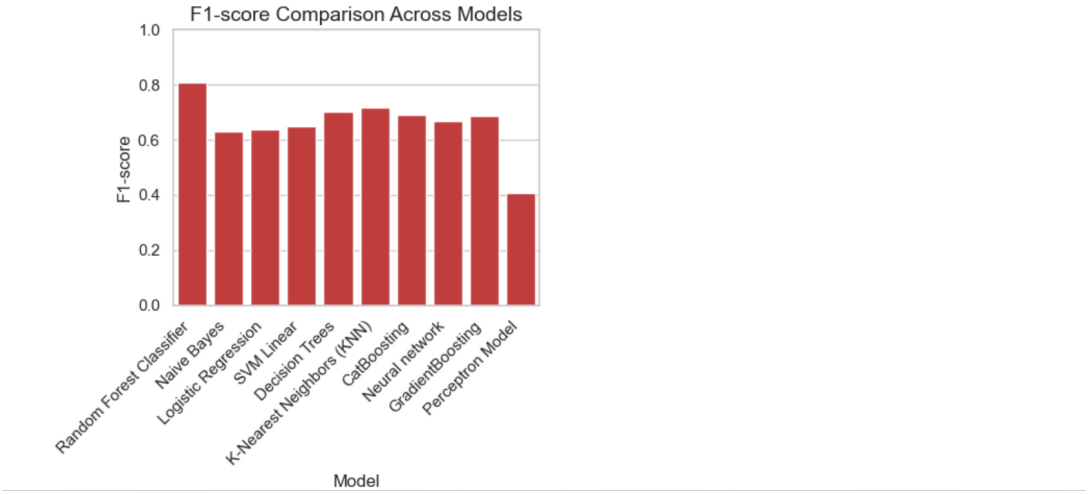

Figure 6.4: F1Score of All models

Figure 6.4 represents F1score for all the models.Purpose of f1score is it balances precision and recall, providing a single metric for model performance, especially useful when class distribution is uneven.Among all the models Random Forest Classifier Has the highest f1score indicating that it balances precision and recall.Naive Bayes has relatively better F1score as compared to its accuracy.Perceptron Model has lowest F1 score indicating its inability to handle Complex Data.

6.0.2 Bar Plots For Metric Comparision (PCA)

Below Are The Bar Plots for PCA data set.

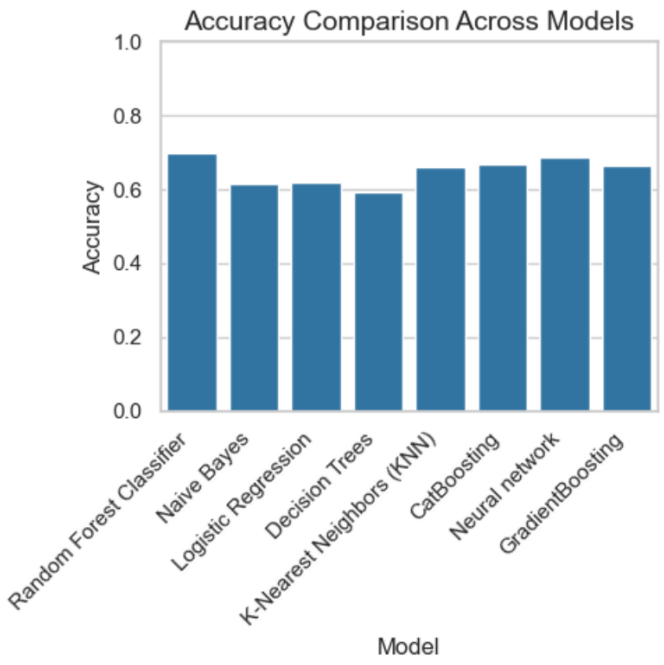

Figure 6.5: Accuracy of All Models

Figure 6.5 illustrates the accuracy comparison across various classification models On the PCA reduced Dataset. For the PCA reduced dataset Acuuracy is decreased Among all the models when compared to original Dataset, the Random Forest,Neural Network, Gradient Boosting, and CatBoosting models demonstrate the highest accuracy.This indicates their strong ability to correctly classify both positive and negative samples.Relatively All remaining models Have same Accuracy.

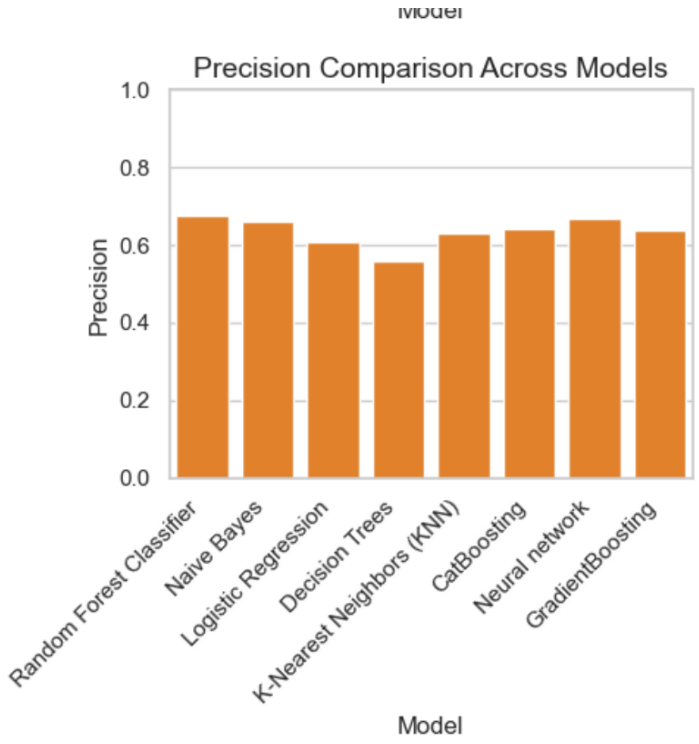

Figure 6.6: Precision of All Models

Figure 6.6 represents the precision scores for all the models. Precision indicates the proportion of true positives among all predicted positives, reflecting the model's ability to avoid false positives. For the PCA reduced dataset Precision is decreased Among all the models when compared to original Dataset,The Random Forest Classifier the highest precision, implying that their positive predictions are highly reliable. Gradient Boosting, CatBoosting, and K-Nearest Neighbors (KNN) also maintain competitive precision values. In contrast, Decision Trees perform poorly in terms of precision, indicating a relatively higher false positive rate.

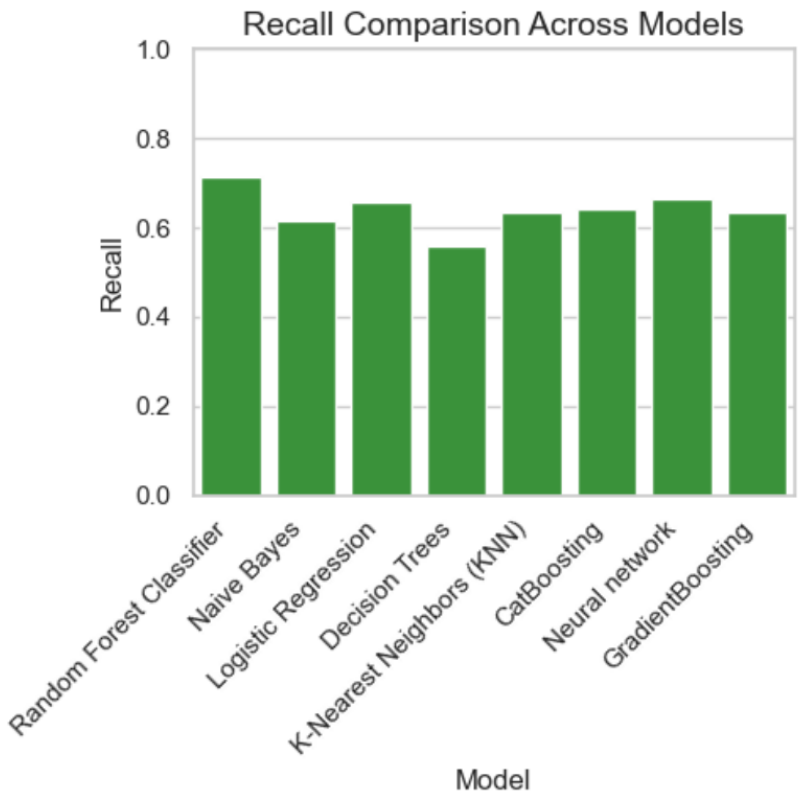

Figure 6.7: Recall of All models

Figure 6.7 represents the Recall scores for all the models.Recall measures the ability of a model to identify all relevant instances (true positives) in a dataset. For the PCA reduced dataset Recall is decreased Among all the models when compared to original Dataset,among the all Models Random Forest has the highest Recall compared to all other models indicating its ability at minimizing false negatives.Decision Trees has the lowest Recall among all the models showing its inability at minimizing errors.

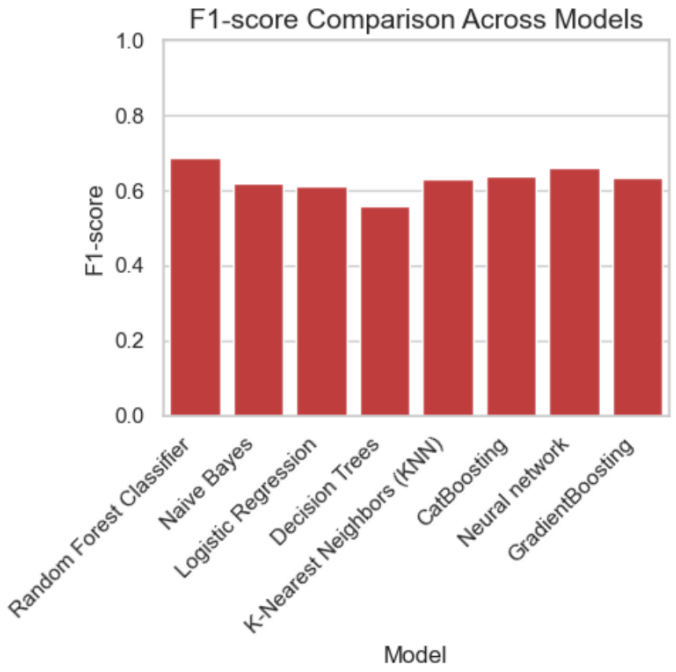

Figure 6.8: F1Score of All models

Figure 6.8 represents F1score for all the models.Purpose of f1score is it balances precision and recall, providing a single metric for model performance, especially useful when class distribution is uneven.For the PCA reduced data set F1score is relatively less for all the models performed as compared to Original Dataset,among all the models Random Forest Classifier Has the highest f1score indicating that it balances precision and recall.Naive Bayes has relatively better F1score as compared to its accuracy.Decision Trees has lowest F1 score indicating its inability to handle Complex Data.

7 Conclusion

In this project, we explored the various factors influencing credit scores and analyzed their impact using data-driven methods. By applying machine learning techniques and statistical analysis, we gained valuable insights into how variables such as payment history, credit utilization, and length of credit history contribute to an individual's credit score. It was seen that the **random forest algorithm** proved to be the best classification technique to predict the credit scores. Overall, this project emphasizes the critical role of credit scores in the financial ecosystem and demonstrates how data science can be leveraged to improve transparency and fairness in credit evaluation.

References

[1] 1.4. Support Vector Machines. https://scikit-learn.org/stable/modules/svm.html.

[2] Consumer Financial Protection Bureau. https://www.consumerfinance.gov/ask-cfpb/what-is-a-credit-score-en-315/.

[3] CatBoost Team. https://catboost.ai/.

[4] Fair Isaac Corporation. https://www.myfico.com/credit-education/credit-scores.

[5] IBM Corporation. https://www.ibm.com/think/topics/neural-networks.

[6] ScienceDirect. https://www.sciencedirect.com/topics/computer-science/gradient-boosting.

[7] Scikit-learn. https://scikit-learn.org/stable/modules/tree.html.

[8] Scikit-learn developers. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[9] Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/.

[10] Tutorials Point Tech. https://www.tpointtech.com/machine-learning-naive-bayes-classifier.

[11] W3Schools. https://www.w3schools.com/ai/ai_perceptrons.asp.