IE 7300 Project
Credit Card Clients Delinquency Prediction

Group 4

AMITOJ SINGH KOHLI
ANKITA NITIN BANDAL
ADITYA GORAKH VELAPURKAR
KIMAYA NIKAM
VINAY SRIDHAR

kohli.am@northeastern.edu
bandal.a@northeastern.edu
velapurkar.a@northeastern.edu
nikam.k@northeastern.edu
sridhar.vi@northeastern.edu

Submission Date: 12/08/2023

ABSTRACT

This project's main aim is to predict credit card delinquency for the upcoming month, and we are using a dataset from the UCI Machine Learning Repository. The dataset has 30,000 instances, 24 features, and one target variable, including credit limit, demographic details, payment history, and bill statements.

We prepared the dataset for model training by meticulously processing the data and engineering features. The preprocessing included handling irregularities in categorical variables, and we introduced a derived feature called 'Total_due.'

We developed four machine learning algorithms, which are Logistic Regression using Gradient Descent, Hard-Margin SVM, Gaussian Naive Bayes, and a Neural Network implemented via TensorFlow and Keras. We evaluated each model based on precision, recall, accuracy, and other relevant metrics.

Our focus was to assess predictive power and computational efficiency. The models' performances provided insights that could aid financial institutions in mitigating risks and making decisions related to credit lending.

INTRODUCTION

Business Problem Definition:

Financial institutions continuously seek reliable methods to predict client delinquency on credit card payments. Accurate predictions can significantly reduce financial risk, refine lending strategies, and enhance customer service.

The ability to predict client credit card payment delinquencies is a key challenge in the constantly changing world of financial institutions. Precise forecasts not only reduce financial hazards but also enable organizations to improve lending practices and customer support. This research aims to estimate customer defaults in the following month by using previous data to build predictive models from scratch in response to this problem. In order to minimize potential financial losses, the initiative seeks to facilitate preventive steps by identifying the underlying traits and patterns of clients who are at risk of default.

The project takes a careful approach to model building and explores machine learning algorithms that are created independently of pre-built libraries. The aim of binary classification is to determine if a client will default or not. This project focuses on developing predictive models from scratch that predict client defaults in the subsequent month based on historical data.

Problem Setting:

The problem at hand is a binary classification task where the outcome variable (Y) indicates whether a client will default (1) or not (0) in the next month. The challenge lies in utilizing historical payment data, billing statements, and client information to build a robust predictive model without reliance on pre-built libraries for model development.

DATA DESCRIPTION

Data Source:

Dataset Link: https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

The dataset is sourced from the UCI Machine Learning Repository and contains 30,000 records and 25 instances. It comprises various features such as credit limit, gender, education, marital status, age, past payment history, bill statements, and previous payment amounts.

This diverse dataset will be leveraged to understand the relationship between client attributes and default behavior. Given the absence of missing values, the dataset offers a suitable foundation for analysis.

Data Dictionary:

Given below is the data dictionary of the dataset: -

- Default Payment Next Month (Yes = 1, No = 0)
- Sex (1 = male; 2 = female)
- Education (1 = graduate school; 2 = university; 3 = high school; 4 = others)
- Marriage (1 = married; 2 = single; 3 = others)
- Age (years)
- Pay_0; . . . ; Pay_6 (-2 = No credit consumed; -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months)
- Bill_Amt1; . . . ;Bill_Amt6 (in NT dollar) (Amount of bill in September 2005; . . . ; Amount of bill in April 2005)
- Pay_Amt1; . . .; Pay_Amt6 (in NT dollar) (Amount paid in September 2005; . . . ; Amount paid in April 2005)

METHODS: MODEL DEVELOPMENT

Dataset Splitting into Training and Test Sets:
Before training our machine learning models, we partitioned our balanced dataset into two sets - a training set and a test set. This crucial step was taken to enable us to evaluate the performance of our models against unseen data confidently. A 75-25 data split can help achieve this, where adequate data is used for learning patterns (X_train and y_train), while a significant portion is kept aside for testing predictions (X_test and y_test). The random_state parameter ensures that the split is deterministic, essential for the reproducibility of model performance evaluation.

Model Development:

In our methodological approach, we have meticulously crafted and rigorously evaluated four distinct machine learning algorithms to construct a robust predictive framework. Our methodology was grounded in the principles of algorithmic development from the ground up, with the notable exception of the neural network model, where we utilized the advanced capabilities of TensorFlow and Keras. Below is a detailed account of the models developed:

1. **Logistic Regression Model Using Gradient Descent (Base Model):** Our initial model was a logistic regression classifier implemented to benchmark subsequent models. The simplicity of logistic regression, paired with the robustness of gradient descent for optimization, provided a solid foundation for our binary classification task. This model's development involved iteratively adjusting the weights to minimize the cost function evaluating the gradients of the cost with respect to the model parameters.

2. **Hard-Margin SVM:** Recognizing the prowess of SVMs in handling high-dimensional data, we developed a Hard-Margin SVM model designed to find the optimal separating hyperplane for our data. This model is especially adept at classifying data with a clear margin of separation, and our implementation focused on maximizing this margin to achieve the best possible classification boundary.

3. **Gaussian Naive Bayes:** We integrated the Gaussian Naive Bayes algorithm, which is predicated on the assumption of independence between predictors and the normal distribution of features. This model is known for its computational efficiency and suitability for datasets with many categorical features.

4. **Neural Networks (Multi-Layer Perceptron Classifier):** To explore the complex nonlinear relationships within our data, we employed a Multi-Layer Perceptron Classifier. This neural network was constructed using TensorFlow and Keras, enabling us to define a sophisticated architecture with multiple layers and activation functions. The model was designed to discern intricate patterns that simpler models might overlook.

Each model was diligently tuned and calibrated to align with the specific characteristics of our dataset. The hyperparameters for each model were fine-tuned to optimize performance, and feature scaling was employed to ensure that each input contributed equitably to the learning process.

We also engaged in performance tuning, a critical step to ensure that our models generalize well to new data while avoiding overfitting.

Subsequent sections will delve into the detailed implementation, optimization strategies employed, and the comparative analysis of the model performances. This comprehensive evaluation will underscore the strengths and limitations of each algorithm within the context of our dataset.

DATA PRE-PROCESSING AND FEATURE ENGINEERING

During the data pre-processing and feature engineering phase of our project, we implemented crucial measures to ready the dataset for modeling. These measures comprised data cleaning, feature engineering, feature scaling, normalization, and feature selection. As a result, our model's accuracy and performance were significantly enhanced.

ID Column Removal:
The 'ID' column was removed from the dataset early in the pre-processing stage. This column consisted of identifiers unique to each client and did not hold predictive power for determining the 'Default Payment Next Month' outcome. Its exclusion helped streamline the dataset and focus on features with potential predictive significance.

Data Type Conversion:
Data type consistency is paramount for accurate computational analysis. As such, we converted the data types of several columns to formats conducive to manipulation and computation. Specifically:

- Columns representing monetary values such as 'LIMIT_BAL', 'BILL_AMT1' through 'BILL_AMT6', and 'PAY_AMT1' through 'PAY_AMT6' were cast to float types to maintain precision in numerical operations.
- Demographic and categorical columns, including 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0' through 'PAY_6', and the target variable 'default payment next month', were converted to integer types to facilitate categorical analysis and model input standardization.

Categorical Variable Transformation:
The 'EDUCATION' and 'MARRIAGE' columns contained categories not initially planned for analysis (namely, the values 5, 6, and 0). To address this, we applied transformations that consolidated these unexpected values into a singular 'other' category (represented by the value 4 for 'EDUCATION' and 3 for 'MARRIAGE'). This consolidation ensured that our categorical variables remained comprehensive and representative of the dataset's scope.

Descriptive Statistics Computation:
Utilizing the describe() function, we computed summary statistics for each feature, providing valuable insights into our dataset's central tendencies, dispersions, and overall distributions. This step was instrumental in identifying data scales, ranges, and potential outliers.

Introduction of 'Total_due' Column:
To deepen our analysis, we engineered a new feature, 'Total_due', calculated as the net amount due by each individual after summing up the bill amounts and subtracting the total payments over the previous six months. This feature aimed to capture the financial behavior of clients more holistically.

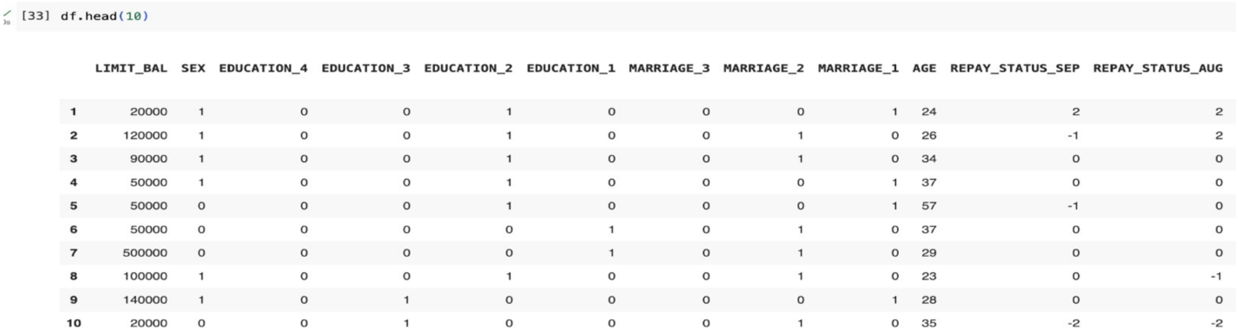
Column Renaming for Clarity:
Several columns were renamed to enhance the clarity and interpretability of the dataset. This renaming aligned the column names more closely with the data they represented, thereby improving the readability of our dataset, and aiding in the subsequent analysis.

Column Rearrangement for Modeling:
We restructured the DataFrame such that the 'Total_due' column was positioned next to the target variable 'default payment next month'. This repositioning was strategically done to align with modeling conventions where the target variable is typically placed as the last column in the dataset, facilitating easier access and manipulation during the model training process.

Encoding Categorical Variables:

- One-hot encoding was applied using the get_dummies function to 'EDUCATION' and 'MARRIAGE' to convert these categorical variables into a format suitable for machine learning algorithms that require numerical input. This created binary columns for each category, preserving the original data structure.

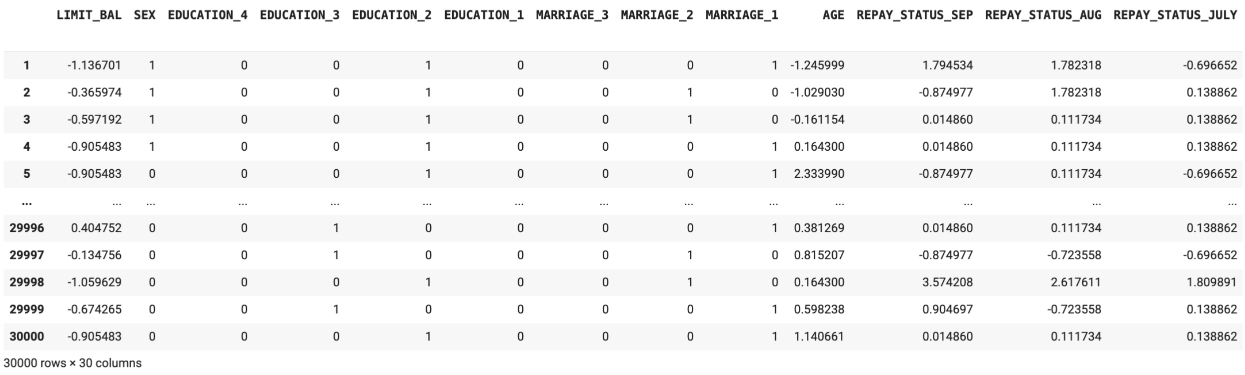
- Label encoding was implemented using the map function on the 'Sex' column, assigning '0' to 'Male' and '1' to 'Female'. This transformation is crucial for simplifying the dataset and aligning it with the requirements of machine learning algorithms.



```
[33] df.head(10)
```

	LIMIT_BAL	SEX	EDUCATION_4	EDUCATION_3	EDUCATION_2	EDUCATION_1	MARRIAGE_3	MARRIAGE_2	MARRIAGE_1	AGE	REPAY_STATUS_SEP	REPAY_STATUS_AUG
1	20000	1	0	0	1	0	0	0	1	24	2	2
2	120000	1	0	0	1	0	0	1	0	26	-1	2
3	90000	1	0	0	1	0	0	1	0	34	0	0
4	50000	1	0	0	1	0	0	0	1	37	0	0
5	50000	0	0	0	1	0	0	0	1	57	-1	0
6	50000	0	0	0	0	1	0	1	0	37	0	0
7	500000	0	0	0	0	1	0	1	0	29	0	0
8	100000	1	0	0	1	0	0	1	0	23	0	-1
9	140000	1	0	1	0	0	0	0	1	28	0	0
10	20000	0	0	1	0	0	0	1	0	35	-2	-2

Feature Standardization:
Numeric features are standardized for each data point. The mean is subtracted, and the standard deviation is divided to normalize numeric attributes. This transformation ensures that each feature contributes proportionately to the final prediction, preventing larger-scale features from disproportionately influencing the model.

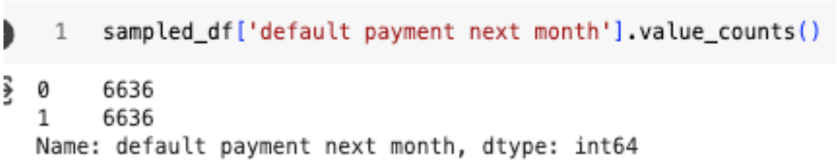


	LIMIT_BAL	SEX	EDUCATION_4	EDUCATION_3	EDUCATION_2	EDUCATION_1	MARRIAGE_3	MARRIAGE_2	MARRIAGE_1	AGE	REPAY_STATUS_SEP	REPAY_STATUS_AUG	REPAY_STATUS_JULY
1	-1.136701	1	0	0	1	0	0	0	1	-1.245999	1.794534	1.782318	-0.696652
2	-0.365974	1	0	0	1	0	0	1	0	-1.029030	-0.874977	1.782318	0.138862
3	-0.597192	1	0	0	1	0	0	1	0	-0.161154	0.014860	0.111734	0.138862
4	-0.905483	1	0	0	1	0	0	0	1	0.164300	0.014860	0.111734	0.138862
5	-0.905483	0	0	0	1	0	0	0	1	2.333990	-0.874977	0.111734	-0.696652
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29996	0.404752	0	0	1	0	0	0	0	1	0.381269	0.014860	0.111734	0.138862
29997	-0.134756	0	0	1	0	0	0	1	0	0.815207	-0.874977	-0.723558	-0.696652
29998	-1.059629	0	0	0	1	0	0	1	0	0.164300	3.574208	2.617611	1.809891
29999	-0.674265	0	0	1	0	0	0	0	1	0.598238	0.904697	-0.723558	0.138862
30000	-0.905483	0	0	0	1	0	0	0	1	1.140661	0.014860	0.111734	0.138862

30000 rows × 30 columns

Balancing the Dataset:

An analysis of the target variable distribution highlighted a significant class imbalance, which could bias our models toward predicting the majority class. To mitigate this issue, an under-sampling (resampling technique) was employed to balance the class distribution on the target column ‘default payment next'. The resultant balanced dataset ‘sampled_df’ is created where both class instances are equal.



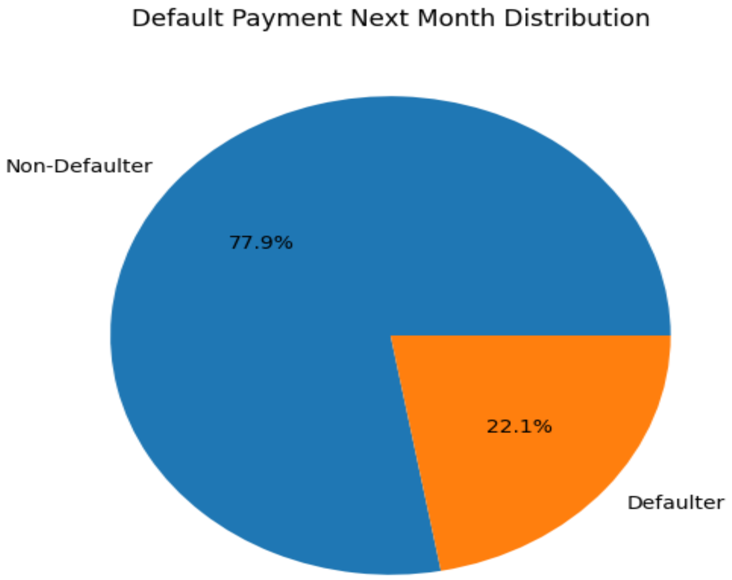
```
sampled_df['default payment next month'].value_counts()
```

```
0    6636
1    6636
Name: default payment next month, dtype: int64
```

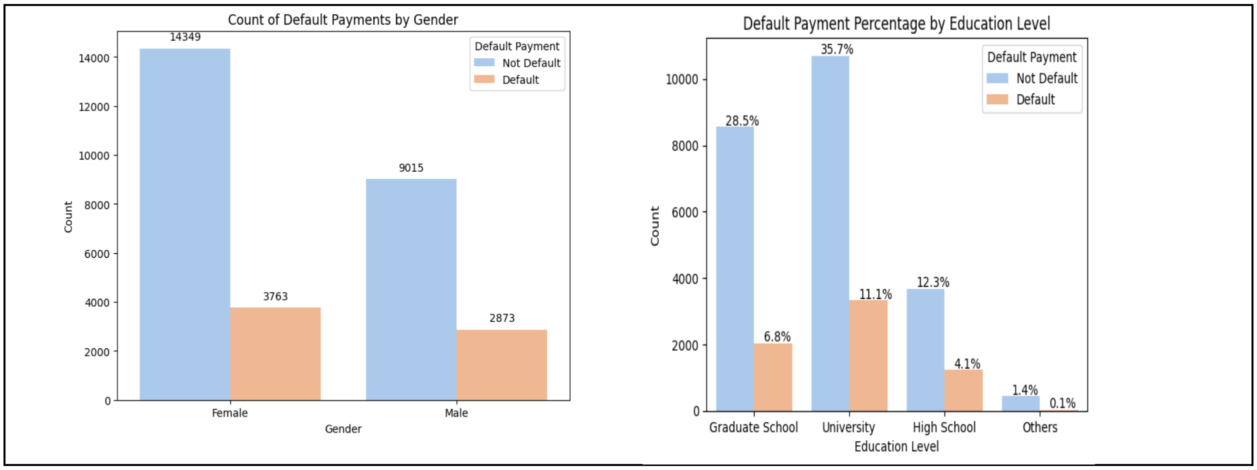
These pre-processing and feature engineering steps were fundamental in shaping the dataset into a refined form optimized for the following machine learning tasks. They were executed carefully to preserve data integrity and relevance, ensuring a solid foundation for building reliable predictive models.

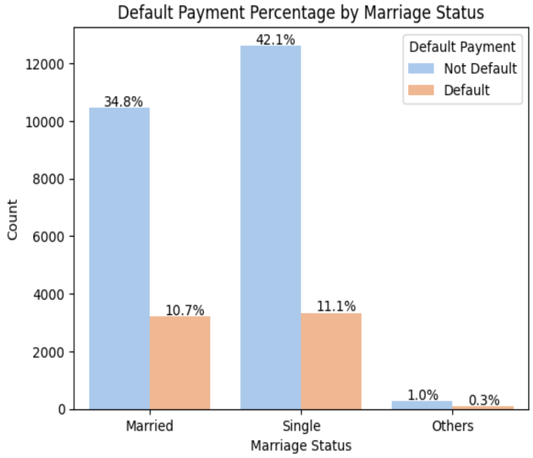
EDA:

1) Pie Chart and Bar Charts

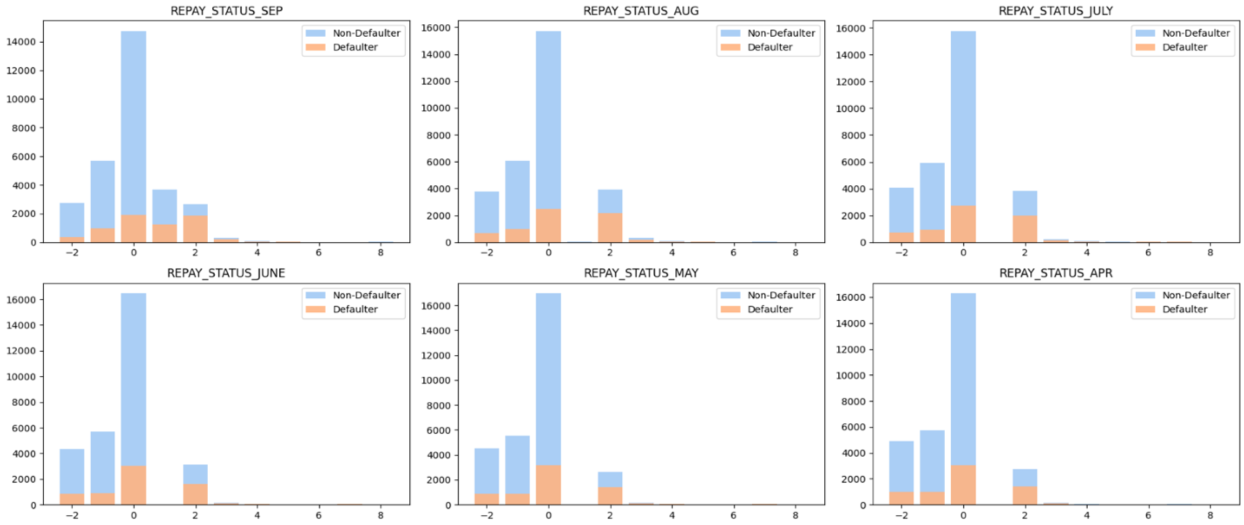


- The Pie chart represents the percentage of customers who will be either non-defaulter or defaulter next month.
- It is visible that majority of the customers will be non-defaulter the next month with 77.9%, while 22.1% of the customers might be defaulter next month.



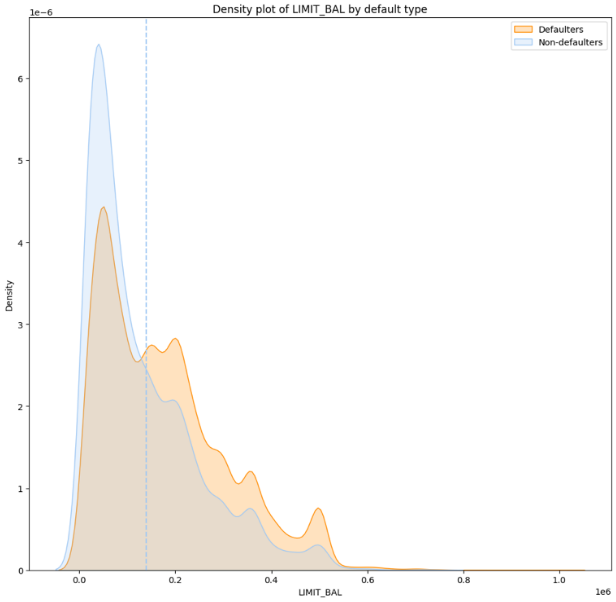


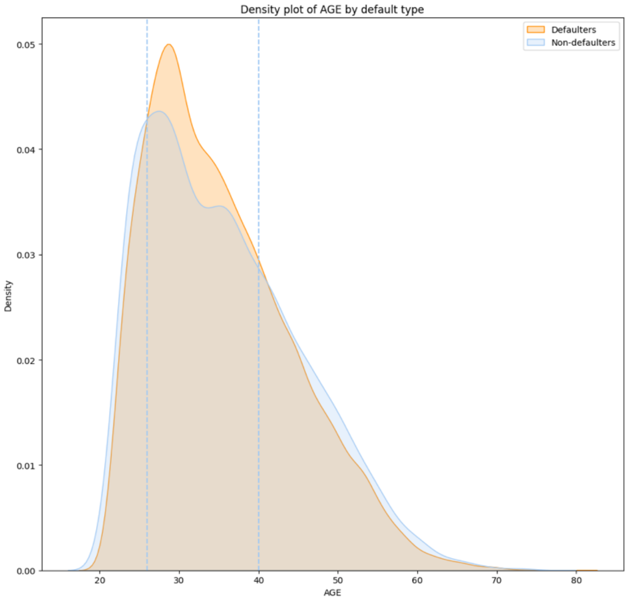
- The above bar plot shows count and percentage distribution of credit card users for different parameters like Gender, Education level, Marriage Status.



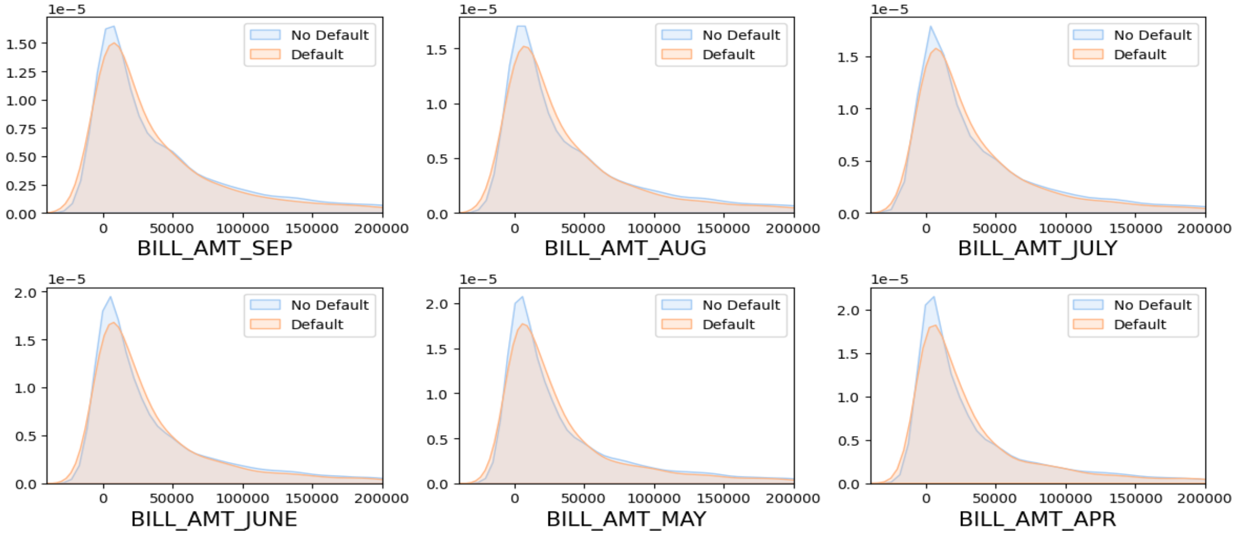
- The chart shows us the repay Status for 6 months and their respective counts for Users.

2) Density Distribution Charts

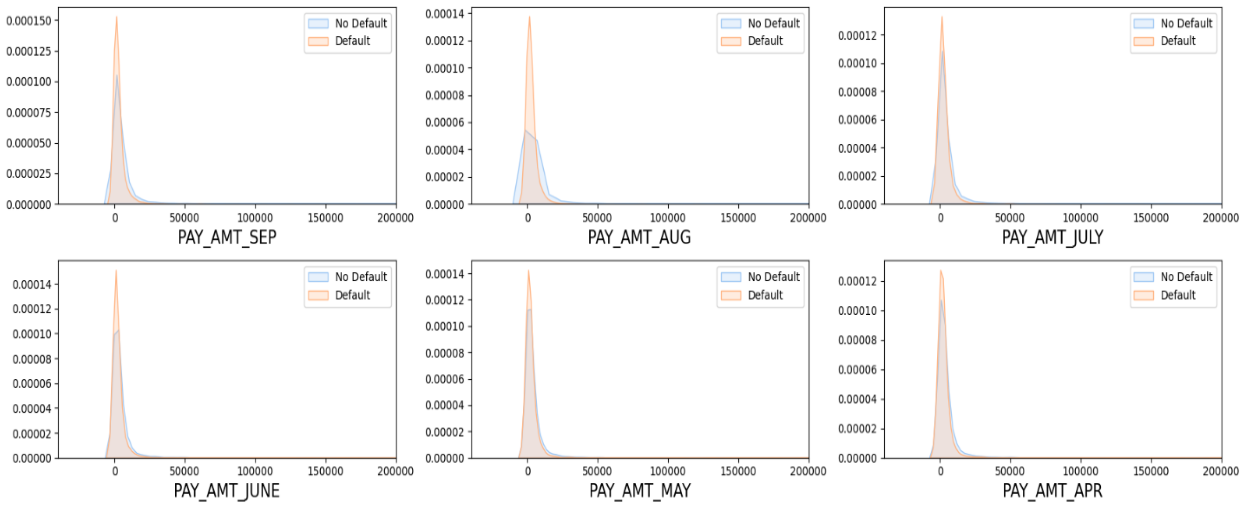




- The above graph shows density plot of Defaulter Type vs LIMIT_BAL and Age vs Defaulter Type.
- Users having low credit card limits tends to be non-defaulters.
- As the limit increases the probability of being defaulter increases.
- Approximately From ages 25 - 40 had highest number of defaulters which started to decrease with increase in age.

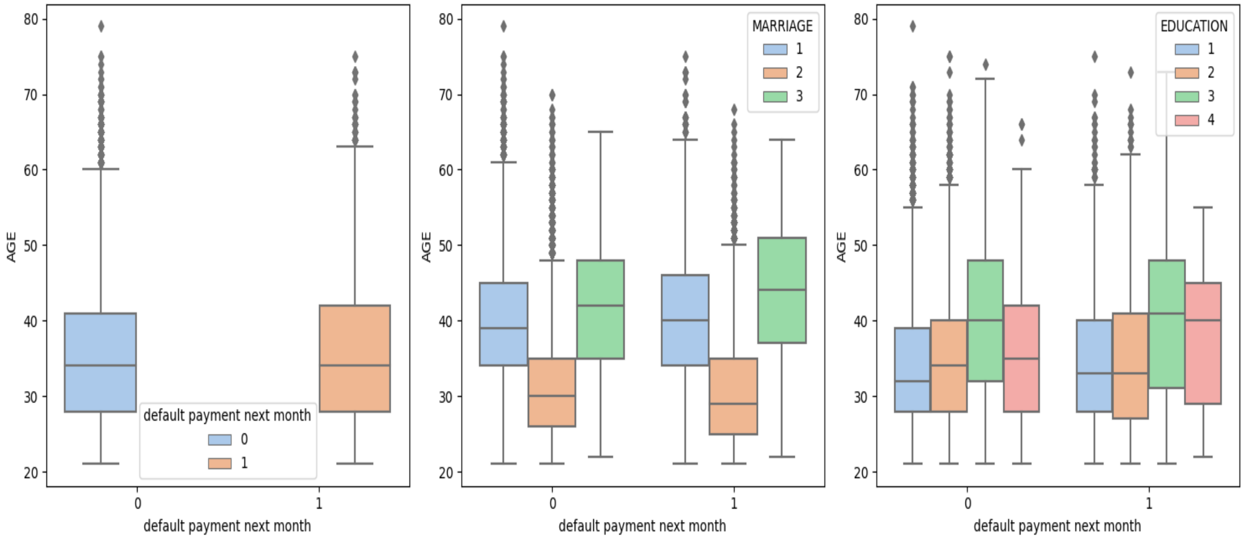


- The above graph shows density distributions of Bill amounts for 6 months (April, May, June, July, August, and September) for defaulters and non-defaulters over time.
- It is visible that for lower billing amount users tend to be non-defaulters.

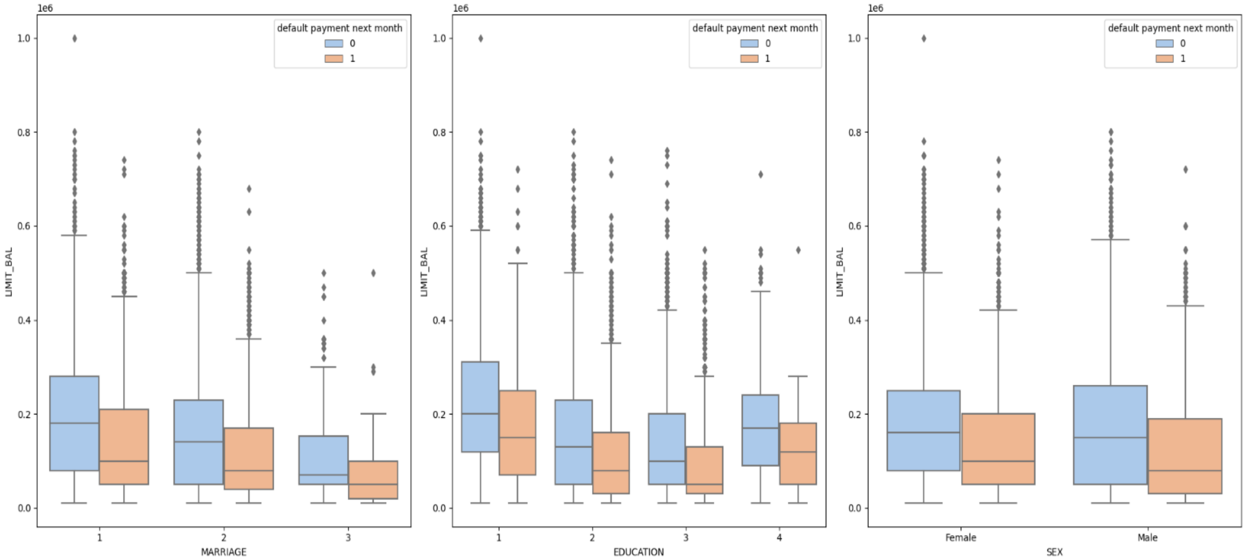


- The above graph shows density distributions of Pay Amount for 6 months (April, May, June, July, August, September) for defaulters and non-defaulters over the time.
- It is visible that for low paying amount users tend to be defaulters.

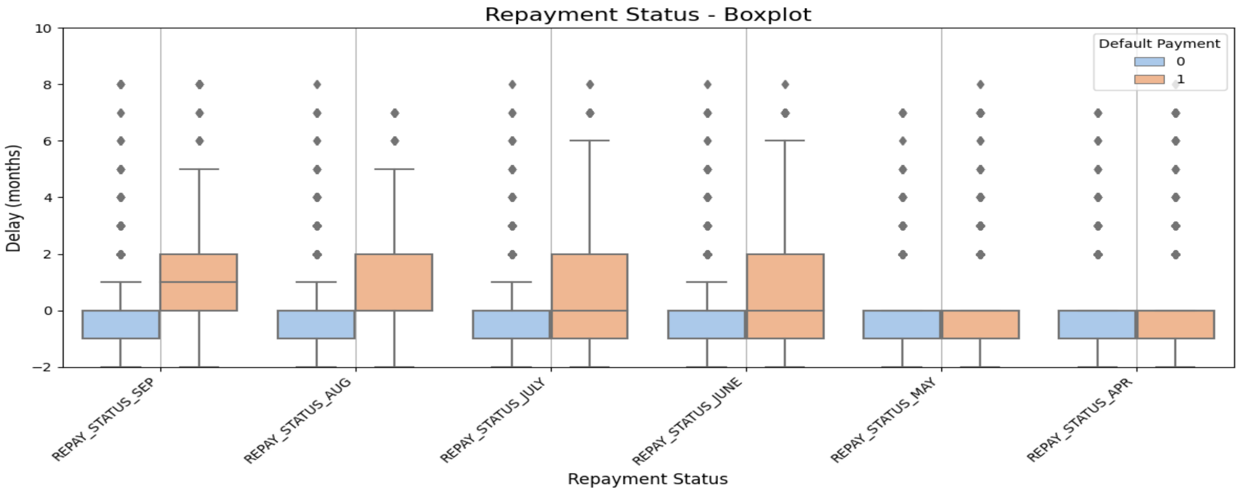
3) Box Plots



- The first graph shows spread of data for default payment next month Vs. Age, here we can see that there are lot of outliers in non-default users.
- The next graphs show spread of data for default payment next month Vs. Age with respect to Marriage & Education.
- It is visible that Marriage group 2 has the highest number of outliers for both users, while group 3 has no outliers.
- For Education data the median lies between 30 – 40 range with maximum number of outliers in group 1.

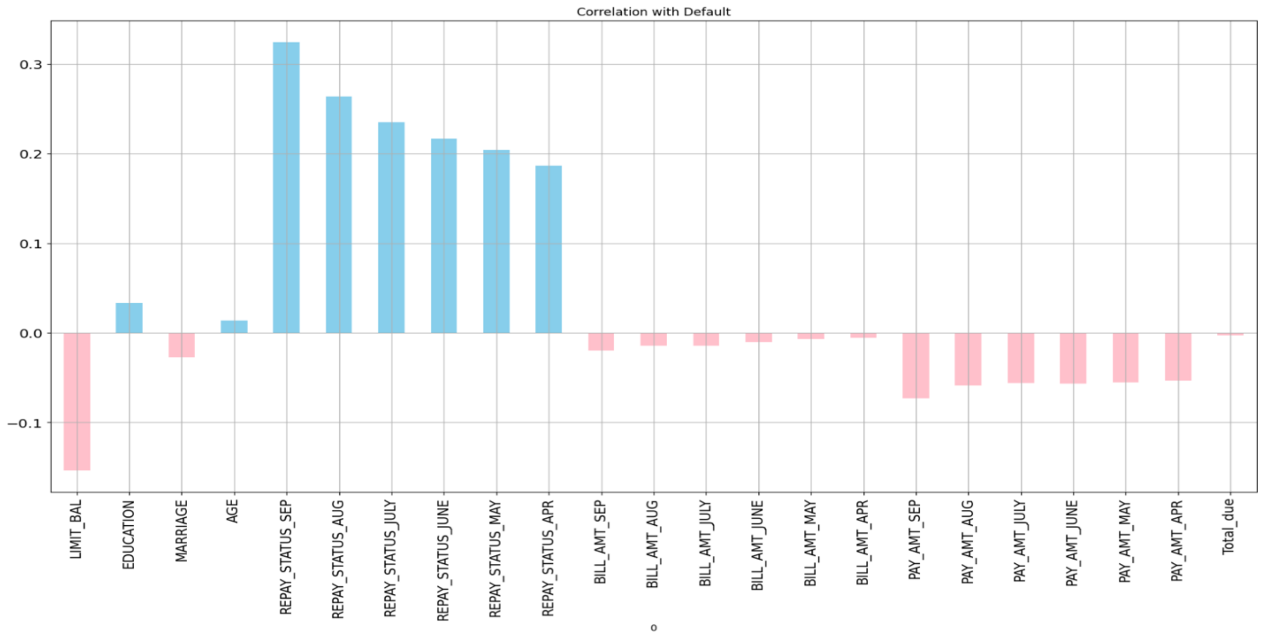


- These graphs show spread of data for Marriage, Education, Sex Vs. Limit Balance with respect to two user groups in our data.
- The position of median for most of the groups suggests that the lower half of the data is more densely packed.
- For all three visualizations non-defaulters have the maximum number of outliers.



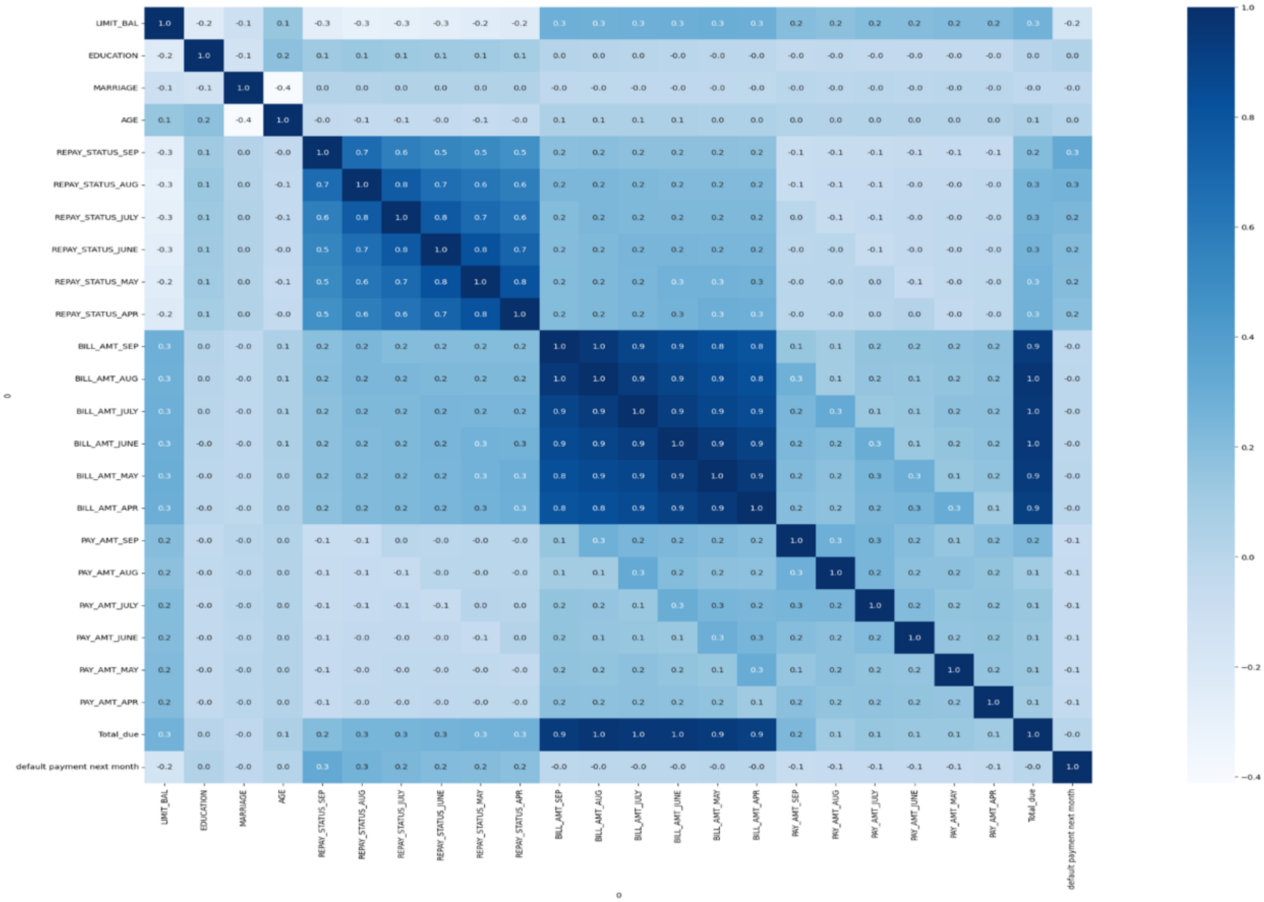
- The above graph shows plots of Repay status for 6 months (April, May, June, July, August, September) with respect to Delay.
- There are more outliers for group 0, with more dense distribution towards lower delay values.

4) Correlation Plots



- This graph shoes correlation of Default variable with all the variables in data.
- Here, we can observe that Repay Status for Sept has the highest positive correlation and Limit_Bal has the highest negative correlation.

5) Correlation Matrix Heatmap



- The heatmap shows visual correlation matrices between all the variables in the data.
- Hence, we can easily identify relationships between variables. High or low correlation coefficients are visible by lighter or darker color intensities.

RESULTS

1) Logistic Regression Model Using Gradient Descent (Base Model):

Implementation Overview:

The logistic regression model was developed from scratch to serve as a foundational classifier for our dataset. The core algorithm was structured around the gradient descent optimization technique. This model is an ideal benchmark for comparing more complex models due to its interpretability and solid theoretical foundations. It outperforms other models when accurately classifying linearly separable data, which perfectly aligns with our initial dataset exploration.

Key Implementation Details:

- Data Preparation: The dataset was preprocessed, including feature scaling, and encoding, to ensure optimal conditions for model training.
- Parameter Initialization: Model parameters (weights) were initialized to zeros, offering a neutral starting point for gradient descent.
- Gradient Descent Algorithm: A loop was established to iteratively update the model parameters, aiming to minimize the cost function, calculated as the negative log-likelihood.

Optimization Strategies:

- Learning Rate Selection: The learning rate, an essential hyperparameter for gradient descent, was carefully chosen (0.001) to balance convergence speed and the risk of overshooting the minimum cost.
- Convergence Criteria: The algorithm was set to terminate upon reaching 5,000 iterations or if the change in cost between iterations fell below a predefined tolerance threshold, thus ensuring efficiency and preventing unnecessary computations.

Metrics of Model Performance:

The model's performance was evaluated using standard classification metrics on the test dataset:

- Precision (61.990%): Indicates a moderate level of reliability in the model's positive (default) predictions.
- Recall (59.881%): Reflects the model's ability to identify actual default cases.
- Accuracy (61.212%): Demonstrates the model's general predictability.
- Specificity (62.568%): Indicates the model’s ability to recognize non-default cases accurately.
- F1-Score (60.917%): Balances precision and recall, which is important for datasets with an uneven class distribution.

Model Complexity and Error Analysis:

- Bias (0.386): Suggests a moderate average error, indicating that the model could be underfitting.
- Variance (0.249): Indicates a reasonable level of prediction fluctuation, implying a balanced model in terms of model flexibility.
- Noise (0.388): Reflects the intrinsic complexity and randomness present in the dataset.
- Expected Loss (1.023): Combines bias, variance, and noise to measure the model's prediction error comprehensively.

Utilization of Computational Resources:

The logistic regression model demonstrated efficiency in resource utilization:

- Memory Usage (369.71 MiB): The peak memory usage during model training was within manageable limits, suggesting the model's feasibility for systems with moderate memory capacities.
- Elapsed Time (63.852 seconds): The model completed its training in a reasonably short time, making it suitable for iterative experiments and real-world applications where time efficiency is crucial.

Conclusion:

The findings from the study suggest that the logistic regression model is particularly effective in situations where computational resources are limited. Although it is a relatively simple algorithm, it provides a strong starting point for comparing and evaluating the performance of more complex models. Overall, the results highlight the usefulness and potential of logistic regression in practical applications.

2) Hard-Margin SVM:

Implementation Overview:

The Hard-Margin Support Vector Machine (SVM) model was architected to establish a hyperplane that maximizes the margin between the two classes in our dataset. By enforcing a stringent margin, this model is designed to be robust against overfitting, making it suitable for datasets where the classes are distinctly separable.

Key Implementation Details:

- Initialization: We initialized the model with a learning rate (alpha) of 0.001 and an iteration cap (max_iter) set at 700. The weight vector (w) and bias term (b) were initialized to zero.
- Model Fitting: The fit method entails iterating over the dataset, adjusting the weight vector and bias term whenever a data point falls within the margin or on the wrong side of the hyperplane.
- Prediction: The predict method determines class labels by computing the sign of the linear combination of input features and the weight vector, offset by the bias.

Optimization Strategies:

- Iterative Training: We employed an iterative approach to refine the model parameters (w and b), enforcing the hard margin constraint.
- Convergence Check: The training loop was iterated 700 times until the model parameters stabilized, indicating an optimal hyperplane.

Metrics of Model Performance:

The model's performance was evaluated using standard classification metrics on the test dataset:

- Precision (78.193%): Indicate a high likelihood that a predicted default is indeed a default case.
- Recall (51.164%): Suggests the model's moderate effectiveness in identifying all actual defaults.
- Accuracy (68.143%): Demonstrates the model's general predictability.
- Specificity (85.453%): Indicates the model’s ability to recognize non-default cases accurately.
- F1-Score (61.855%): Balances precision and recall, which is important for datasets with an uneven class distribution.

Model Complexity and Error Analysis

- Bias (1.454): The model showed a notable deviation from the true values, which could suggest underfitting.
- Variance (0.889): The model's predictions were moderately sensitive to the specific data it was trained on.
- Noise (1.481): The noise level, recorded at 1.481, quantified the unpredictability inherent in the dataset.
- Expected Loss (3.825): Combines bias, variance, and noise to measure the model's prediction error comprehensively.

Utilization of Computational Resources:

The Hard-Margin SVM model showcased commendable efficiency in the utilization of computational resources during its training and evaluation phases:

- Memory Usage (369.77 MiB): The model's peak memory consumption during the training process was recorded at 369.77 MiB. This level of memory usage is within the manageable spectrum for most modern computing systems, indicating the model's suitability even in environments with moderate computational resources.
- Elapsed Time (79.924 seconds): The training of the Hard-Margin SVM model was completed in under 80 seconds. It demonstrates the model's expedience, rendering it a practical choice for iterative development cycles and potential real-world application scenarios where prompt model training is beneficial.

Conclusion:

The performance of the Hard-Margin SVM model, in terms of computational resource utilization, aligns with our project's goal of developing efficient predictive models. Its swift training time and modest memory requirements suggest that the model can operate effectively in various computational settings, which is advantageous for deployment in systems with limited resources. Moreover, the model provides a robust comparison point for evaluating the computational demands of more complex algorithms that will be explored later in our analysis.

3) Gaussian Naive Bayes:

Implementation Overview:

We integrated a Gaussian Naive Bayes classifier, predicated on the probabilistic foundation of Bayes' theorem coupled with the assumption of normally distributed features. This statistical algorithm is recognized for its simplicity and effectiveness, especially in classification tasks with underlying feature independence.

Key Implementation Details:

- Initialization: The classifier was initialized with a smoothing factor, enhancing the robustness of probability estimates.
- Probability Estimation: We fitted a normal distribution to each feature within the training data, incorporating the smoothing factor to account for the possibility of outliers.
- Predictive Analysis: The predict function was devised to evaluate the class probabilities for each instance, selecting the class with the highest posterior probability as the predicted label.

Optimization Strategies:

- Smoothing: A smoothing factor was utilized to temper the model's sensitivity to the data, mitigating the issues arising from zero-frequency counts.
- Probability Distributions: The model was equipped to handle the possibility of non-uniform feature distributions, preparing it for a wide range of data scenarios.

Metrics of Model Performance:

Upon execution, the Gaussian Naive Bayes model demonstrated:

- Precision (58.410%): Reflects the model's accuracy in predicting default instances indicative of a moderate level of precision.
- Recall (79.403%): It signifies the model's strength in identifying actual defaults demonstrating a high recall rate.
- Accuracy (61.061%): The overall accuracy confirms the model's reliability in the general classification task.
- Specificity (42.362%): Indicates the model's performance in correctly predicting non-default instances, suggesting potential for improvement.
- F1-Score (67.308%): Balances precision and recall, which is important for datasets with an uneven class distribution.

Model Complexity and Error Analysis:

- Bias (0.404): The bias measurement suggests the model's predictions are reasonably close to the actual values.
- Variance (0.216): Indicates that the model has a moderate level of flexibility in response to the training data.
- Noise (0.389): Reflects the error level inherent to the dataset.
- Expected Loss (1.009): The total expected loss, aggregating bias, variance, and noise offer an overview of the model's predictive error.

Utilization of Computational Resources:

- Memory Usage (369.95 MiB): The model's training process exhibited a peak memory consumption of 369.95 MiB, indicating efficient memory usage.
- Elapsed Time (345.39 seconds): The classifier completed its training and prediction processes in under six minutes, showcasing its practicality for applications where timely predictions are valued.

Conclusion:

With its respectable precision and high recall, the Gaussian Naive Bayes classifier serves as a viable model for our classification objectives. Although its specificity could be enhanced, the model's high recall is particularly beneficial for identifying most actual default cases, a critical requirement in credit risk assessment. The model's low memory footprint and reasonable computation time make it an attractive option for real-time prediction tasks and iterative modeling processes. Overall, the Gaussian Naive Bayes is an essential component in our suite of predictive models, offering substantive insights into the nature of default prediction.

4) Neural Networks (Multi-Layer Perceptron Classifier using TensorFlow, Keras)

Implementation Overview:

We implemented a Multi-Layer Perceptron (MLP) Classifier using TensorFlow and Keras, harnessing the power of neural networks for our binary classification task. This advanced model architecture is known for its ability to capture complex, nonlinear patterns that simpler models may not capture effectively.

Model Architecture and Training:

- Network Design**:** The MLP consisted of an input layer matching the feature count, a hidden layer with 32 neurons and an output layer with sigmoid activation for binary classification.
- Compilation and Training**:** The Adam optimizer and binary cross-entropy loss function were used to compile the model. It underwent 40 training epochs with a batch size of 200, alongside a validation split of 30%, to monitor performance and prevent overfitting.

Model Performance Metrics:

Post-training, the MLP Classifier's performance was assessed:

- Accuracy (69.65%): Indicates the model's general effectiveness in correctly classifying the data.
- Precision (74.0%): Highlights the model's accuracy in predicting default instances.
- Recall (61.194%): Reflects the model's ability to identify a high proportion of actual defaults.
- Specificity (78.271%): Demonstrates the model's strength in correctly identifying non-default cases.
- F1 Score (67.059%): Balances precision and recall, which is important for datasets with an uneven class distribution.

Error Analysis and Model Complexity:

- Bias (0.188): The low bias value suggests that the model predictions align closely with the actual data.
- Variance (0.057): Indicates a low level of fluctuation in the model's predictions, showing stability across different data sets.
- Noise (0.197): This represents the error introduced by the inherent variability in the dataset.
- Expected Loss (0.442): A composite measure of bias, variance, and noise, providing an overview of the model's overall prediction error.

Computational Resource Utilization:

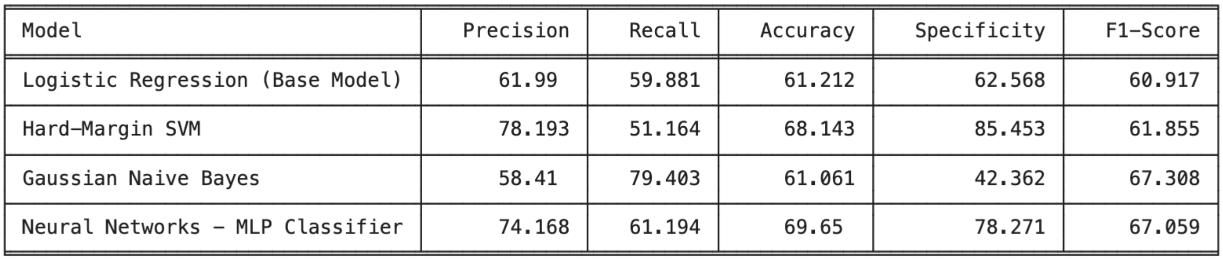
- Memory Usage (805.96 MiB): Peak memory usage during the model training and evaluation process was noted, showing the resource demand of the MLP Classifier.
- Elapsed Time (25.315 seconds): The model completed its training and evaluation process efficiently, suggesting its suitability for applications requiring rapid model retraining or deployment.

Conclusion:

The MLP Classifier demonstrated a commendable balance of accuracy, precision, recall, and specificity, indicating its efficacy in our classification task. The model's architecture successfully captured complex patterns in the dataset, while its computational efficiency underscores its potential for real-time prediction applications. The results affirm the MLP Classifier as a potent tool in our predictive modeling arsenal, capable of providing deep insights into credit default prediction.

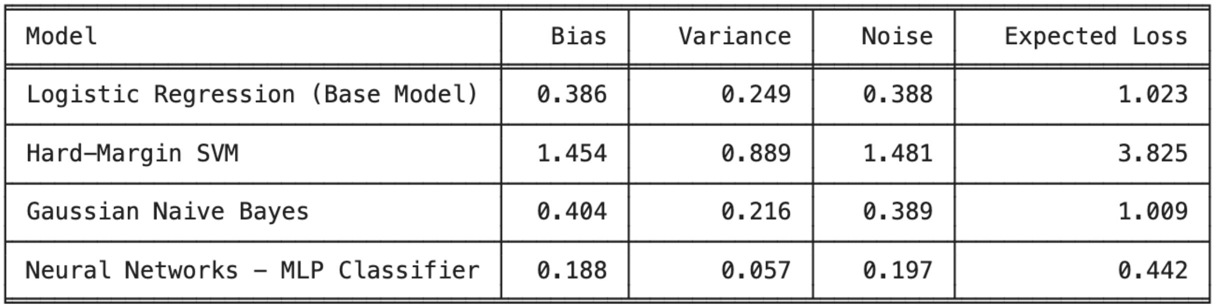
DISCUSSION
(PERFORMANCE EVALUATION AND MODEL SELECTION)

A comprehensive evaluation of four distinct machine learning models was conducted to ascertain their effectiveness in predicting credit card defaults. The assessment criteria included precision, recall, accuracy, specificity, F1-score, bias-variance trade-off, time complexity, and space complexity. The following table summarizes the performance metrics for each model:



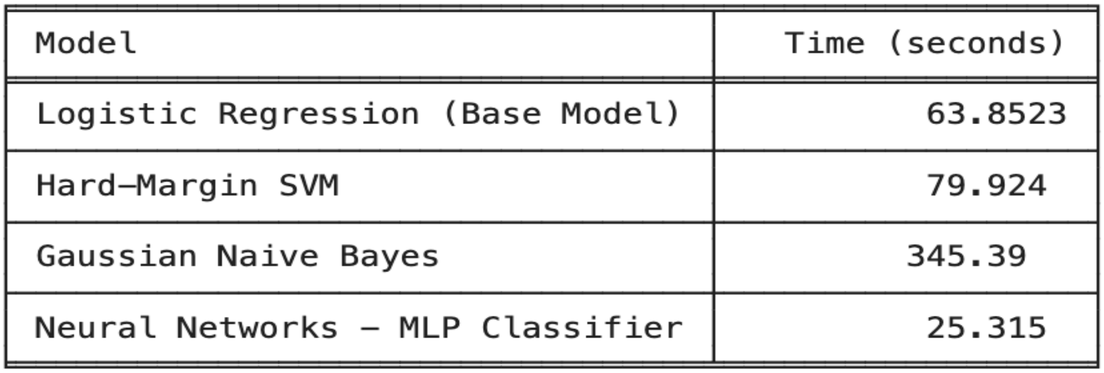
Model	Precision	Recall	Accuracy	Specificity	F1-Score
Logistic Regression (Base Model)	61.99	59.881	61.212	62.568	60.917
Hard-Margin SVM	78.193	51.164	68.143	85.453	61.855
Gaussian Naive Bayes	58.41	79.403	61.061	42.362	67.308
Neural Networks - MLP Classifier	74.168	61.194	69.65	78.271	67.059

Considering the bias-variance trade-off, the following observations were made:



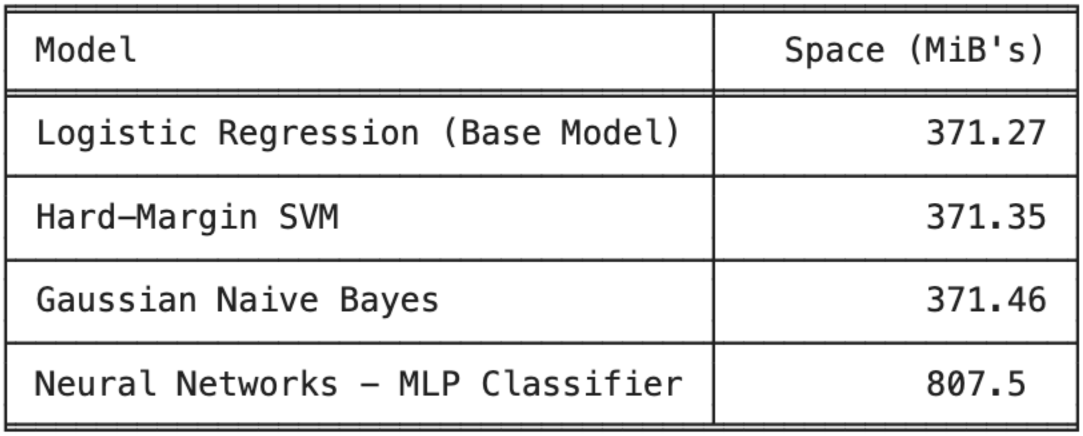
Model	Bias	Variance	Noise	Expected Loss
Logistic Regression (Base Model)	0.386	0.249	0.388	1.023
Hard-Margin SVM	1.454	0.889	1.481	3.825
Gaussian Naive Bayes	0.404	0.216	0.389	1.009
Neural Networks - MLP Classifier	0.188	0.057	0.197	0.442

Considering the time complexity, the following observations were made:



Model	Time (seconds)
Logistic Regression (Base Model)	63.8523
Hard-Margin SVM	79.924
Gaussian Naive Bayes	345.39
Neural Networks - MLP Classifier	25.315

Considering the space complexity, the following observations were made:



Model	Space (MiB's)
Logistic Regression (Base Model)	371.27
Hard-Margin SVM	371.35
Gaussian Naive Bayes	371.46
Neural Networks - MLP Classifier	807.5

Model Selection:

Upon analysis, the **Neural Networks – MLP Classifier emerged as the superior model** for our dataset. The rationale for its selection is multifold:

- **Balanced Accuracy and Precision**: The MLP model achieved the highest accuracy and precision among the evaluated models, suggesting it is the most capable of correctly predicting defaults and non-defaults.

- **Bias-Variance Trade-Off:** Exhibiting the lowest bias and variance, the MLP model demonstrates an optimal balance, indicating that it generalizes well without being overly complex or overly simplistic.

- **Time Efficiency:** The MLP had the shortest training time, showing that it can be retrained and deployed rapidly, which is crucial for practical applications.

- **High Accuracy**: The MLP Classifier leads with an accuracy of 69.65%, highlighting its superior predictive reliability and generalization to new data.

- **F1-Score:** With an F1-score competitive with the highest (Gaussian Naive Bayes), the MLP provides a balanced performance between precision and recall.