LOAN APPROVAL PREDICTION

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

in

Computer Science & Engineering

by

Mohit Sharma, Parv Gatecha, Bhagya Shah

(Roll Nos. MT2024091, MT2024108, MT2024135)

to the

DEPARTMENT OF CSE
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
BANGALORE - 560100, INDIA

December 2024

ABSTRACT

The process of accepting conventional loans mostly involves handling huge datasets involving multiple variables, which can cause challenges in terms of computational efficiency and extracting usefull informations. **Our project aims to address these problems by applying various ML models to predict if a loan will be approved or rejected.** A main part of our project is using **Principle Component Analysis (PCA)** to reduce the dimensions of the dataset, to make it simpler with simulaneously retaining the most important information as well.

Many classifications algorithms we have explored, including **Logistic Regression, Support Vector Machine (SVM), Naive Bayes and an Ensembled model** which is based on the **Maximum Voting Method.** All the mentioned models were trained and tested using a real-world loan dataset, and their performances were evaluated using metrics like accuracy, precision, recall and F1-score.

This report covers **Exploratory Data Analysis, data preprocessing,** the concepts of all the models used, and at the end a **comparision of all the model results**. The insights shown contributes to the understanding of how Machine Learning can be applied in financial decision making systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Loan approval step is a major decision making process for a financial institutions, as it includes the studying the eligibility of applicants for issuing loans. The Loan Approval Prediction problem mainly focuses on using data driven ways to predict whether a loan application has to be approved or not based on the **applicant's financial background and demographic attributes**. These attributes include factors such as **CIBIL score, income, employement status, loan term, loan amount and assets valuation**. The final result we want is the loan status, if it is **approved or rejected**.

The intricacy and subjectivity of the loan evaluation process provide one of the main obstacles. Manual evaluations can be ineffective, biassed, and slow, particularly when processing a large number of applications. In addition to putting institutions at risk of financial losses, inaccurate evaluations result in the unjust rejection of qualified applicants. This makes the necessity for objective, automated solutions to expedite the approval process evident.

The objective is to create a system that guarantees **accuracy and fairness** while also expediting loan reviews. We can identify the critical elements influencing loan approvals and produce more accurate forecasts for upcoming applications by examining historical data and applying machine learning algorithms. The efficiency could be significantly increased with this type of approach.

Chapter 2

Description of Dataset

Loan ID	Customer ID	Loan Status	Current Loan Amount	Term	Credit Score	Annual Income	Years in current job	Home Ownership	Purpose
14dd8831-6af5-400b-83ec-68e61888a048	981165ec-3274-42f5-a3b4-d104041a9ca9	Fully Paid	445412	Short Term	709	1167493	8 years	Home Mortgage	Home Improvements
4771cc26-131a-45db-b5aa-537ea4ba5342	2de017a3-2e01-49cb-a581-08169e83be29	Fully Paid	262328	Short Term			10+ years	Home Mortgage	Debt Consolidation
4eed4e6a-aa2f-4c91-8651-ce984ee8fb26	5efb2b2b-bf11-4dfd-a572-3761a2694725	Fully Paid	99999999	Short Term	741	2231892	8 years	Own Home	Debt Consolidation
77598f7b-32e7-4e3b-a6e5-06ba0d98fe8a	e777faab-98ae-45af-9a86-7ce5b33b1011	Fully Paid	347666	Long Term	721	806949	3 years	Own Home	Debt Consolidation
d4062e70-befa-4995-8643-a0de73938182	81536ad9-5ccf-4eb8-befb-47a4d608658e	Fully Paid	176220	Short Term			5 years	Rent	Debt Consolidation
89d8cb0c-e5c2-4f54-b056-48a645c543dd	4ffe99d3-7f2a-44db-afc1-40943f1f9750	Charged Off	206602	Short Term	7290	896857	10+ years	Home Mortgage	Debt Consolidation
273581de-85d8-4332-81a5-19b04ce68666	90a75dde-34d5-419c-90dc-1e58b04b3e35	Fully Paid	217646	Short Term	730	1184194	< 1 year	Home Mortgage	Debt Consolidation
db0dc6e1-77ee-4826-acca-772f9039e1c7	018973c9-e316-4956-b363-67e134fb0931	Charged Off	648714	Long Term			< 1 year	Home Mortgage	Buy House
8af915d9-9e91-44a0-b5a2-564a45c12089	af534dea-d27e-4fd6-9de8-efaa52a78ec0	Fully Paid	548746	Short Term	678	2559110	2 years	Rent	Debt Consolidation
0b1c4e3d-bd97-45ce-9622-22732fcdc9a0	235c4a43-dadf-483d-aa44-9d6d77ae4583	Fully Paid	215952	Short Term	739	1454735	< 1 year	Rent	Debt Consolidation
32c2e48f-1ba8-45e0-a530-9a6622c18d9c	0de7bcdb-ebf4-4608-ba39-05f083f855b6	Fully Paid	99999999	Short Term	728	714628	3 years	Rent	Debt Consolidation
fa096848-6143-4907-b2cf-852a0b06171c	aa0a6a22-a95e-48e0-ba4f-b83456d424e4	Fully Paid	541970	Short Term			10+ years	Home Mortgage	Home Improvements
403d7235-0284-4bb6-919a-09402fecbf7b	11581f68-de3c-49d8-80d9-22268ebb323b	Fully Paid	99999999	Short Term	740	776188	< 1 year	Own Home	Debt Consolidation

Figure 2.1: Dataset Features

Monthly Debt	Years of Credit History	Months since last delinquent	Number of Open Accounts	Number of Credit Problems	Current Credit Balance	Maximum Open Credit	Bankruptcies	Tax Liens
5214.74	17.2	NA	6	1	228190	416746	1	0
33295.98	21.1	8	35	0	229976	850784	0	0
29200.53	14.9	29	18	1	297996	750090	0	0
8741.9	12	NA	9	0	256329	386958	0	0
20639.7	6.1	NA	15	0	253460	427174	0	0
16367.74	17.3	NA	6	0	215308	272448	0	0
10855.08	19.6	10	13	1	122170	272052	1	0
14806.13	8.2	8	15	0	193306	864204	0	0
18660.28	22.6	33	4	0	437171	555038	0	0
39277.75	13.9	NA	20	0	669560	1021460	0	0
11851.06	16	76	16	0	203965	289784	0	0
23568.55	23.2	NA	23	0	60705	1634468	0	0
11578.22	8.5	25	6	0	134083	220220	0	0

Figure 2.2: Dataset Features

- **Dataset Overview:** This dataset consists of records which are related to the loan applications, showing details of applicants financial and demographic attributes and it also shows the outcome of the loan approval process.
- **Features and Target:** The dataset has various input features also known as **independent variables** and **one target feature** it has called as dependent variable:
  - **CIBIL Score:** A numerical representation of the applicant's creditworthiness.
  - **Income:** The applicant's monthly or annual income, indicating their financial capacity.

  - **Employment Status:** A categorical variable representing the type of employment (e.g., salaried, self-employed, unemployed).
  - **Loan Term:** The duration (in months or years) for which the loan is requested.
  - **Loan Amount:** The amount of money the applicant seeks to borrow.
  - **Assets Value:** The total value of the applicant's assets, representing their financial stability.
  - **Loan Status:** The target variable indicating whether the loan was approved or rejected.

- **Data Type:**
  - Numerical: Features like CIBIL Score, Income, Loan Amount, Assets Value.
  - Categorical: Features like Employment Status and Loan Status.

- **Size of the Dataset:** The dataset contains **100514 records (rows)** and **19 features (columns).**

- **Missing Data:** The dataset contain missing values for few features, which need to be addressed during the preprocessing task.

- **Imbalanced Target Variable:** The distribution of the `Loan Status` (approved/rejected) may be skewed, requiring appropriate handling during model development.

- **Source of Data:** The dataset originates from historical loan applications, capturing real-world scenarios.

- **Use Case:** This dataset is widely used to build machine learning models for predicting loan approvals, aiding financial institutions in automating and optimizing the decision-making process.

Chapter 3

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crutial step in understanding the structure, quality, and statistical properties of the dataset. This chapter shows the steps which are being taken to explore the dataset and visualize its features.

3.1 Dataset Overview

The dataset used consists the following columns in it:

`Customer ID, Loan ID, Loan Status, Current Loan Amount, Credit Score, ...`

After reading the dataset, the following steps were performed for analysis:

- Displaying the first few rows using `df.head()`.
- Checking the shape of the dataset with `df.shape`.
- Listing the column data types using `df.dtypes`.
- Calculating summary statistics using `df.describe()`.

3.2 Dataset Overview

- Total columns initially: **19.**
- Dropped `cust_id` and `loan_id`, leaving **17 columns**.

3.3 Target Variable Analysis

- Unique values in `Loan Status`: `['Fully Paid', 'Charged Off', NaN]`.
  - **Fully Paid → 1.**
  - **Charged Off → 0.**
- Dropped rows with NaN values in the `Loan Status` column.

3.4 Correlation Insights

- Observed correlations:

  Credit Score - Loan Status: -0.467328 Loan Status - Credit Score: -0.467328

3.5 Data Visualization

- **Categorical Columns:**
  - Generated count plots for the following categorical columns:
    - `Term`
    - `Years in Current Job`
    - `Home Ownership`
    - `Purpose`

  These plots provide insights into feature distributions (e.g., Figure 3.3, Figure 3.4).

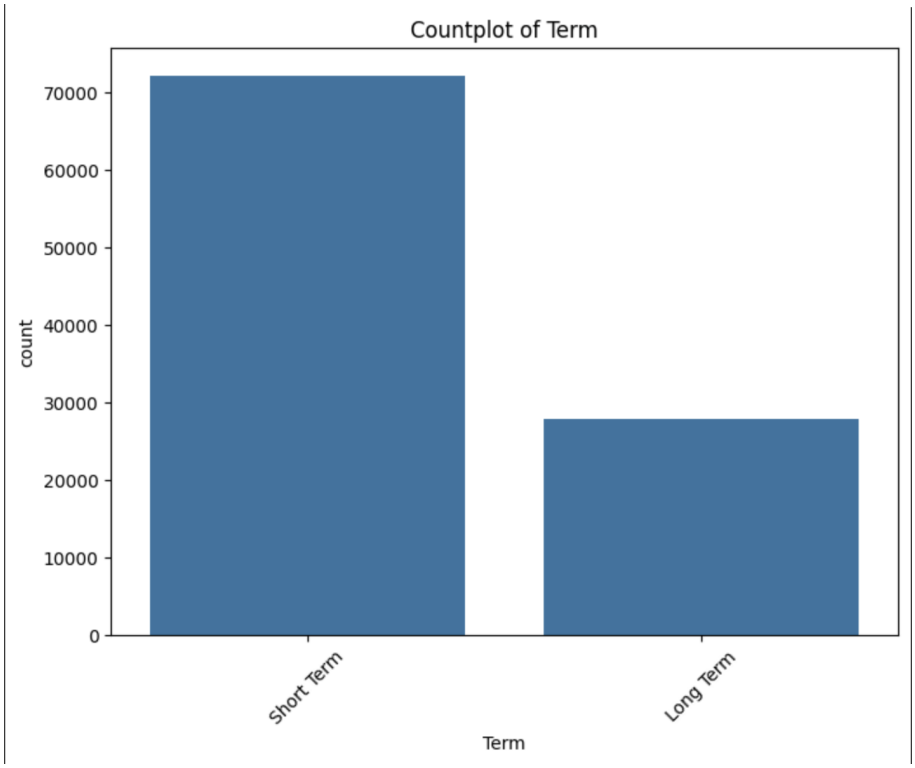

Figure 3.1: Countplot of Term

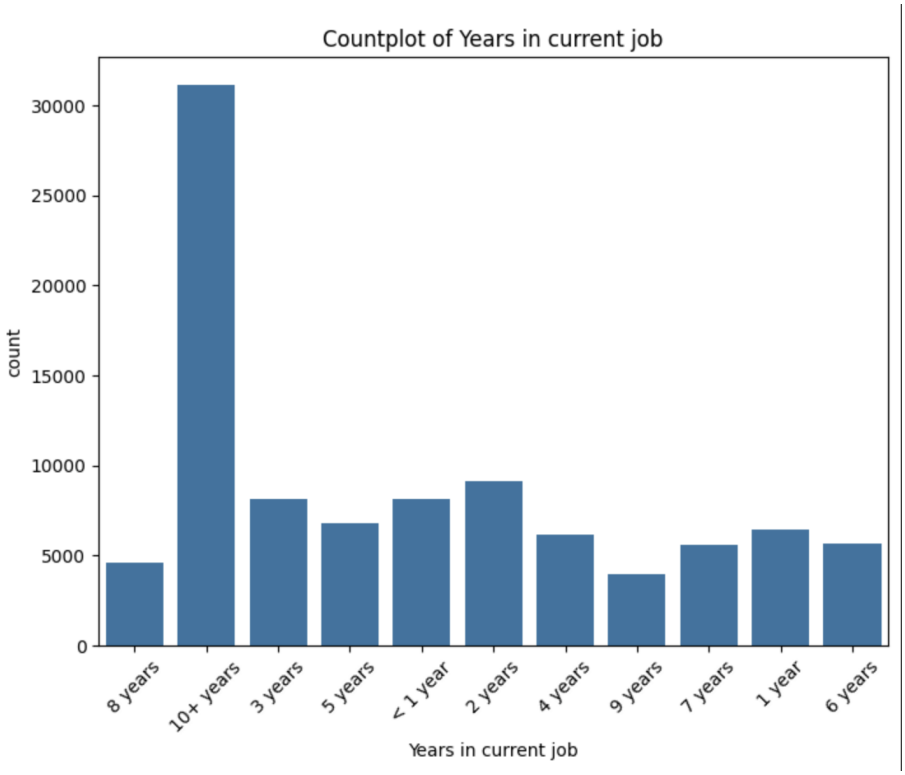

Figure 3.2: Countplot of Years in current job

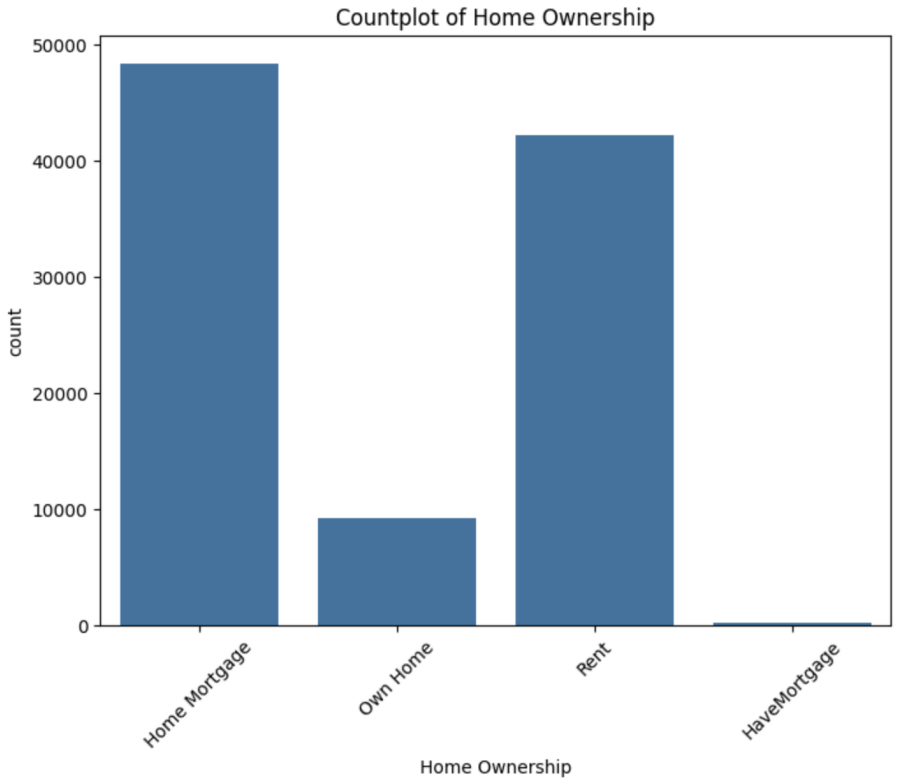

Figure 3.3: Countplot of Home ownership

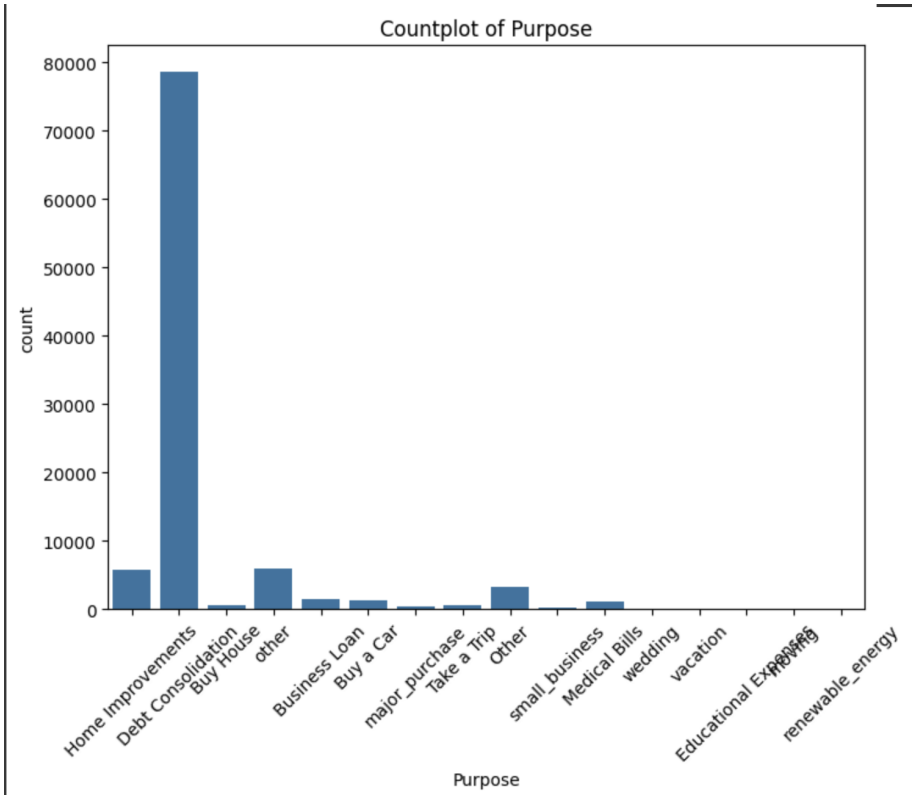

Figure 3.4: Countplot of Purpose

- **Numerical Columns:**
  - Created histograms for the following numerical columns:
    * Loan Status
    * Current Loan Amount
    * Credit Score
    * Annual Income
    * Monthly Debt
    * Years of Credit History
    * Months since Last Delinquent
    * Number of Open Accounts
    * Number of Credit Problems
    * Current Credit Balance
    * Maximum Open Credit
    * Bankruptcies
    * Tax Liens
- **Insights:**
  - Observed that most numerical columns follow a normal distribution.
  - This insight was critical for guiding the outlier removal process using a standard deviation-based approach.

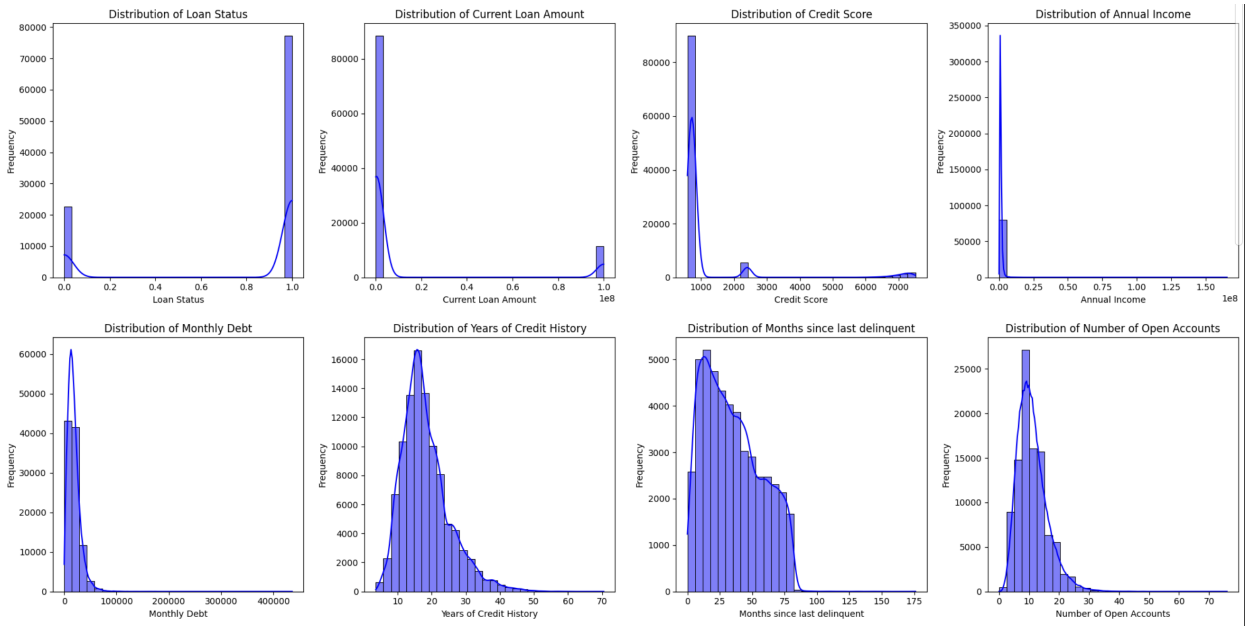

Figure 3.5: Histogram Plots

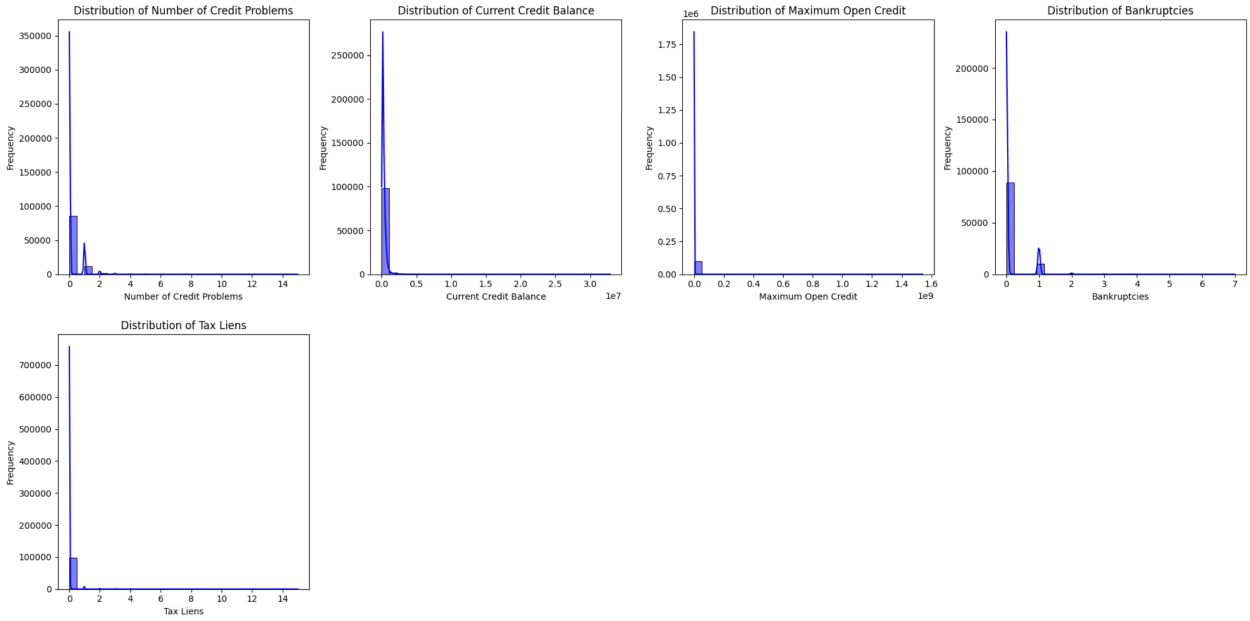

Figure 3.6: Histogram Plots

Boxplots were created for numerical columns to detect outliers. These can be removed by the method suggested above. An example figure is provided below:

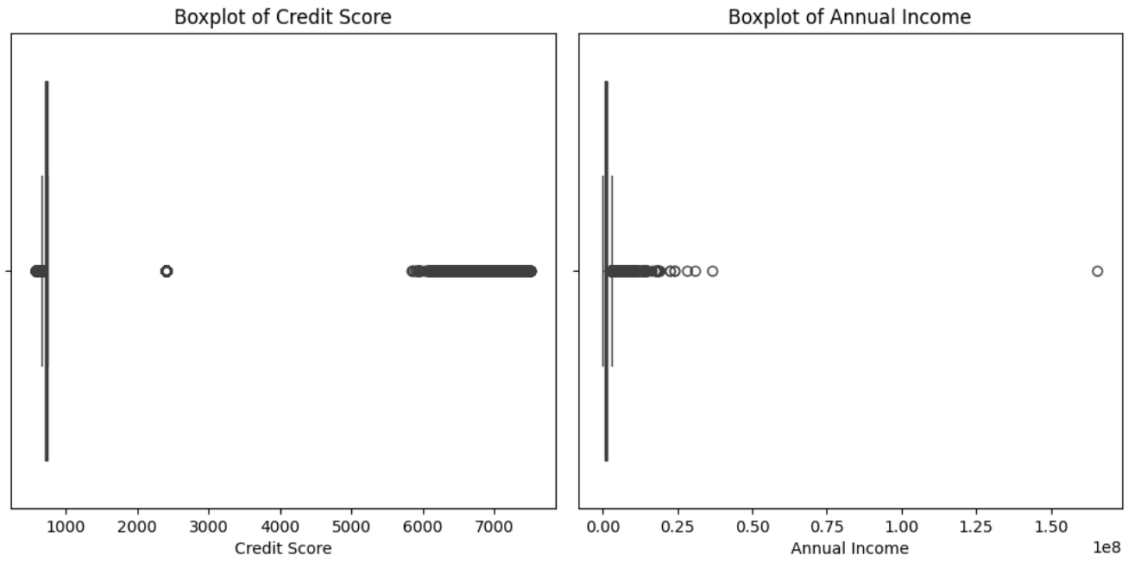

Figure 3.7: Boxplots for Numerical Columns

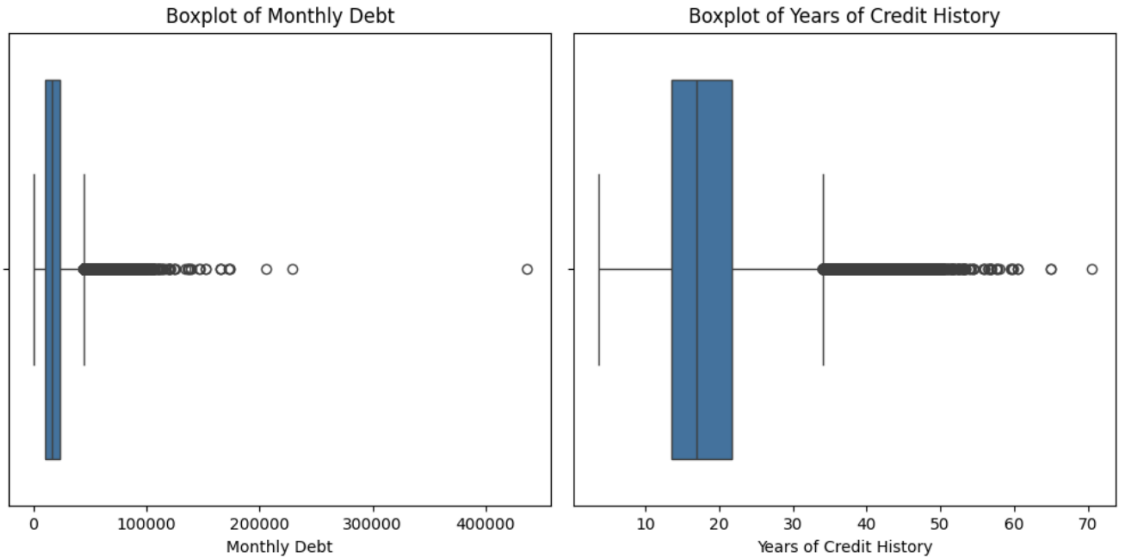

Figure 3.8: Boxplots for Numerical Columns

3.5.1 Analysis of Loan Purpose

- The counts of each loan purpose category are listed below:

Purpose	**Count**
Debt Consolidation	78,552
Other	6,037
Home Improvements	5,839
Business Loan	1,569
Buy a Car	1,265
Medical Bills	1,127
Buy House	678
Take a Trip	573
Major Purchase	352
Small Business	283
Moving	150
Wedding	115
Vacation	101
Educational Expenses	99
Renewable Energy	10

Table 3.1: Counts of Loan Purpose Categories

- Rows with loan purposes having a count of ¡ **100** were removed.
- **Original Shape:** (100,000, 17) → **Filtered Shape:** (99,891, 17).

3.6 Analysis of Home Ownership

- Unique value counts in `Home Ownership`.
- Rows with `Home Ownership` categories having a count < **200** were removed:
  - **Shape remained unchanged:** (99,891, 17).

3.7 Correlation Analysis

- Plotted a heatmap for numerical features, observing:

  Months Since Last Delinquent (MSLD): Correlation with Loan Status = **0.0136**

- Dropped the `MSLD` column due to weak correlation with `Loan Status`.

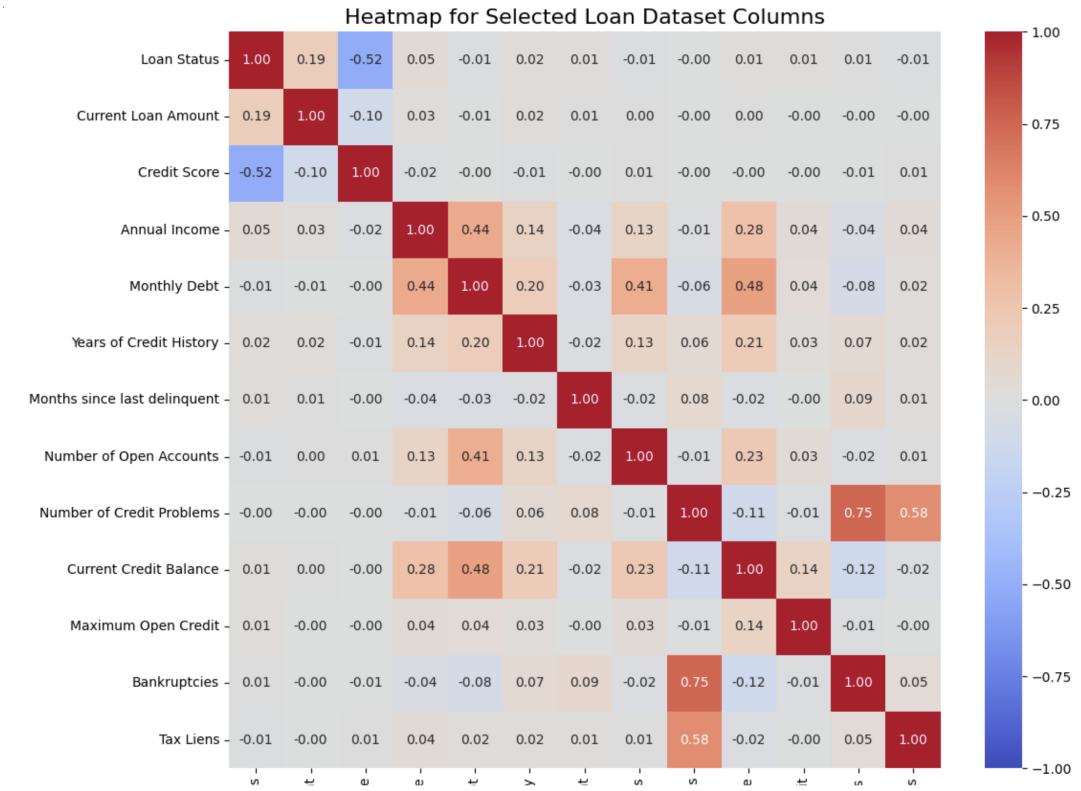

Figure 3.9: Heatmap

Chapter 4

Data Preprocessing

Preprocessing is essential to prepare the dataset for Machine Learning models. This particular chapter explains the transformations we have done on the data set.

4.1 Handling Missing Values

- Replaced NaN values in the `Credit Score` column with the mean, computed separately for approved and rejected loans.
- Replaced missing values in `Annual Income` with the mean of corresponding approved or not-approved loan statuses.
- Removed rows with NaN values in `Maximum Open Credit`, `Bankruptcies`, and `Tax Liens` due to weak correlations with `Loan Status`.

4.2 One-Hot Encoding (OHE)

- Applied OHE on categorical columns:
  - `Years in Current Job`: Increased column count to **26**.
  - `Term`: Short Term → `TRUE`, Long Term → `FALSE`, retaining **26** columns.
  - `Purpose` and `Home Ownership`: Increased column count to **40**.

4.3 Outlier Removal

- Removed outliers using the standard deviation approach:
  - **Shape before removing outliers:** (99,691, 39).
  - **Shape after removing outliers:** (96,068, 39).
- Replotted KDE histograms, confirming columns now followed a more normal distribution.

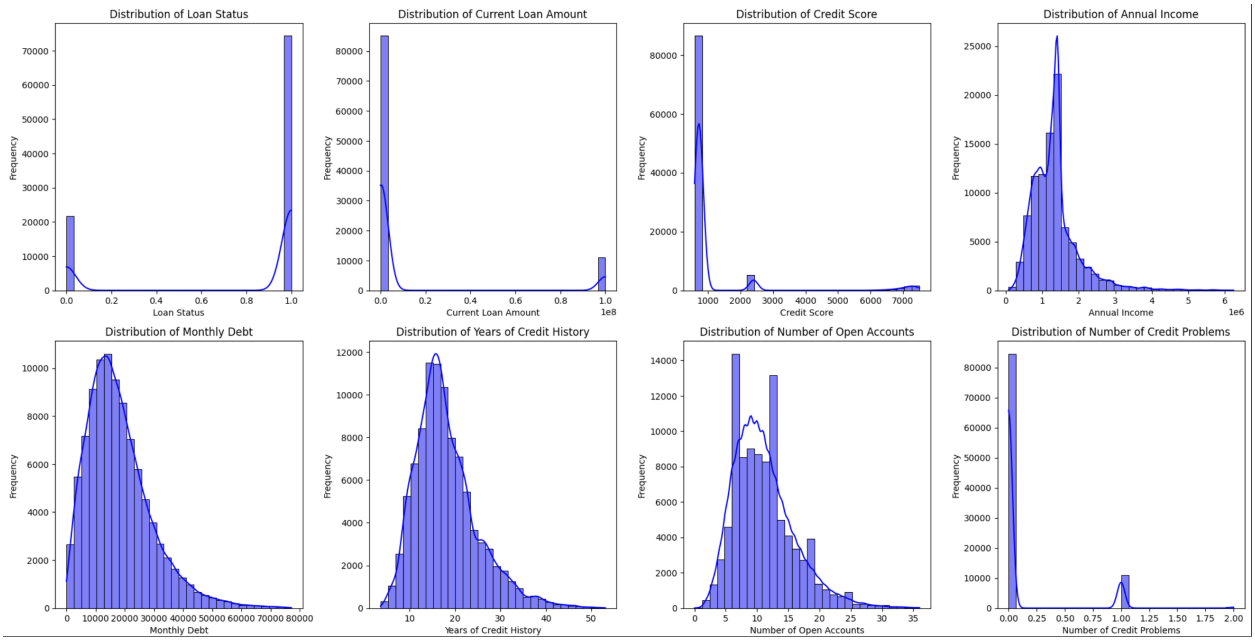

Figure 4.1: Histogram Plots

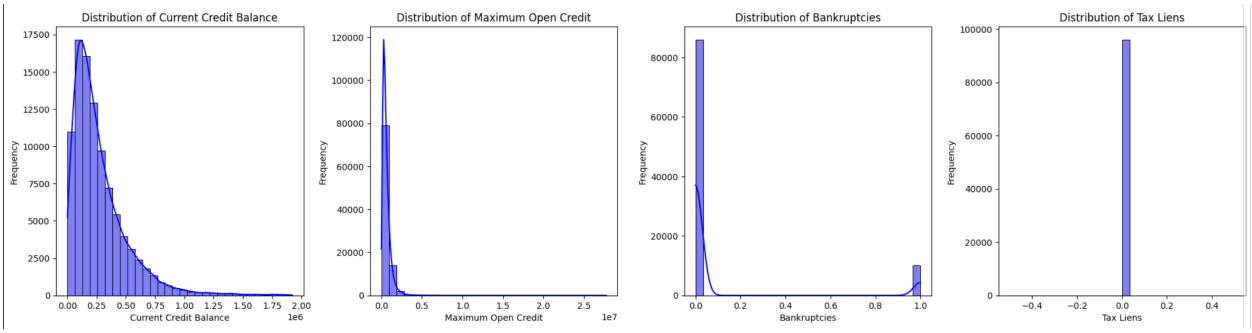

Figure 4.2: Histogram Plots

4.4 Scaling and Normalization

- Standardized the dataset (features in `X`) using `StandardScaler`.

4.5 Train-Test Split

The dataset was split into training and testing subsets with a ratio of 70:30 using `train_test_split()`.

- **X_train:** (67,247, 39).
- **X_test:** (28,821, 39).

4.6 Dimensionality Reduction

- Implemented PCA, selecting the top **11 components** based on **maximum variance**:

– **Transformed X_train:** (67,247, 11).
– **Transformed X_test:** (28,821, 11).

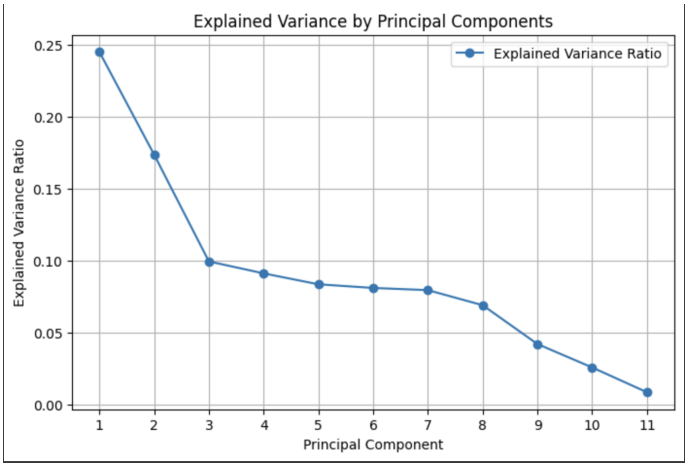

Figure 4.3: Variance by Principle Components

4.7 Prepared Data Summary

- Categorical features encoded as numerical values.
- Numerical features scaled to **zero mean** and **unit variance**, since they follow normal distribution as shown above.
- Outliers removed for improved model performance.

Chapter 5

Metrics Used

We used the following metrics to evaluate the performance of models applied to the loan approval dataset.

5.1 Accuracy

Accuracy is commonly used evaluation metrics for classification tasks. It measures the proportion of correctly classified instances out of the total instances. It is calculated using the formula:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Accuracy was used to evaluate the Logistic Regression, Support Vector Machine (SVM), and Naive Bayes models. While high accuracy indicates good performance, it may not always be reliable for imbalanced datasets.

5.2 Confusion Matrix

The confusion matrix gives a detailed breakdown of the model's predictions compared to the actual outcomes. It includes four components:

- True Positives (TP): Correctly predicted positive cases.
- True Negatives (TN): Correctly predicted negative cases.
- False Positives (FP): Incorrectly predicted positive cases.
- False Negatives (FN): Incorrectly predicted negative cases.

5.3 Precision, Recall, and F1-Score

These metrics are often derived from the confusion matrix:

- **Precision:** Measures the accuracy of positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall:** Measures the model's ability to identify all positive instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score:** The harmonic mean of precision and recall.

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

Chapter 6

Description of Models used

6.1 Model 1: Logistic Regression

6.1.1 Introduction of the Model

- Logistic regression is a commonly used supervised learning algorithm for binary classification, which predicts probabilities for two possible outcomes, here in this case: loan approved or not approved.
- Implementation of this model employs gradient descent for optimization and improves model parameters by minimizing the error through multiple iterations.

6.1.2 Core Ideas of the Model

- **Learning Rate (lr):**
  - Learning Rate determines the step size at each iteration of gradient descent.
  - To ensure convergence while avoiding overshooting the minimum, the model is trained at a learning rate of 0.01.
- **Number of Iterations (n_iters):**
  - Number of Iterations specifies the total number of times the optimization loop runs to update weights and bias.
  - To ensure sufficient training for the model, number of iteration is configured as a large value of 1000.
- **Gradient Descent Optimization:**
  - Gradients (dw and db) are calculated to minimize the loss function:
    - dw: It represents the gradient of the weights, indicating how much each feature contributes to the overall error. It is computed as the average contribution of all features.
    - db: It represents gradient with respect to the bias term. It is computed as the average error across all the data points.

- Parameters (weights and bias) are updated after each iteration using the following mathematical equations:

$$\text{weights} -= \text{lr} \times \text{dw}$$

$$\text{bias} -= \text{lr} \times \text{db}$$

- **Sigmoid Function:**
  - It is used to map the linear model output to probabilities in the range [0, 1].
  - The formula for sigmoid is:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  - The values are limited to a range between -500 and 500 to avoid numerical overflow condition.
- **Binary Classification Threshold:**
  - Predictions are classified as follows:
    * Probability > 0.5: Class 1.
    * Probability ≤ 0.5: Class 0.

6.1.3 Performance

- **Evaluation Metric Calculations:**
  - The predicted values (`y_pred`) are compared to the actual labels (`y_test`) to compute the performance metrics.
  - The formulas for each metric are as follows:
    * **Accuracy:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

    * **Precision:**

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

    * **Recall:**

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

    * **F1-Score:**

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

- **Model Performance:**
  - After training, the model achieved the following metrics on the test dataset:

    - **Accuracy:** 82.88%
    - **Precision:** 85.99%
    - **Recall:** 82.88%
    - **F1-Score:** 78.66%
- These metrics were computed to assess the model's ability to classify data points accurately while maintaining a balance between false positives and false negatives.

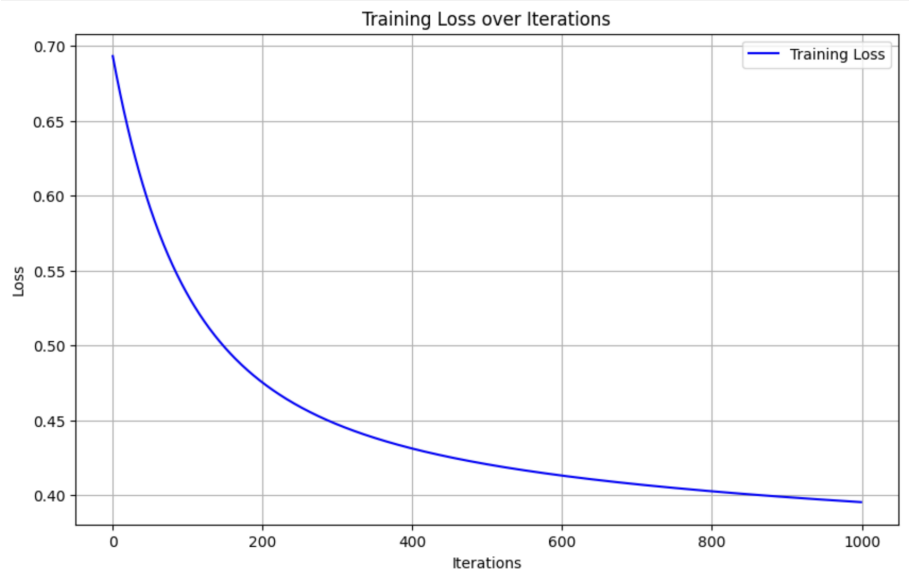

Figure 6.1: Logistic Reg loss over Iterations

6.2 Model 2: Support Vector Machine (SVM)

6.3 Introduction of the Model

- Support Vector Machines are powerful supervised learning models that are designed to achieve the optimal hyperplane that separates two classes with maximum separation.
- Implementation of this model focuses on a linear SVM with hinge loss and uses L2 regularization to overcome overfitting.

6.4 Core Ideas of the Model

- **Learning Rate (lr):**
  - To ensure stable convergence while iteratively minimizing loss, while updating weights and bias during gradient descent, Learning rate is set as 0.001.
- **Regularization Parameter (lambda_param):**
  - It adds a penalty term to prevent overfitting and control the magnitude of the weight vector.
  - It also helps in maintaining the balance between maximizing the margin between two classes and minimizing misclassifications.
- **Optimization Process:**
  - Gradients are calculated, and the weights (w) and bias (b) are updated step by step to reduce the hinge loss.
  - Two cases for updating weights and bias:
    - Condition Met: If $y_i \times (\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1$, only the regularization term is updated.
    - Condition Not Met: If $y_i \times (\mathbf{w} \cdot \mathbf{x}_i - b) < 1$, both the regularization and misclassification terms are updated.
- **Hinge Loss:**
  - Hinge loss measures the model's ability to correctly classify points while maximizing the margin.
  - Defined as:
$$\text{Loss} = \frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i \cdot (\mathbf{w} \cdot \mathbf{x}_i - b)\right)$$
- **Prediction Rule:**
  - The sign of the decision boundary $(\mathbf{w} \cdot \mathbf{x} - b)$ determines class labels:
    - Positive sign: Class 1.
    - Negative sign: Class -1.

6.5 Performance

- **Evaluation Metric Calculations:**
  - The predicted values (y_{pred}) are compared to the actual labels (y_{test}) to compute the performance metrics.
  - The formulas for each metric are as follows:
    - **Accuracy:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

    - **Precision:**

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

    - **Recall:**

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

    - **F1-Score:**

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

- **Model Performance:**
  - After training the model with 1000 iterations, a learning rate of 0.001, and a regularization parameter of 0.01, the SVM achieved the following metrics on the test dataset:
    - **Accuracy:** 77.84%
    - **Precision:** 79.01%
    - **Recall:** 77.84%
    - **F1-Score:** 78.92%

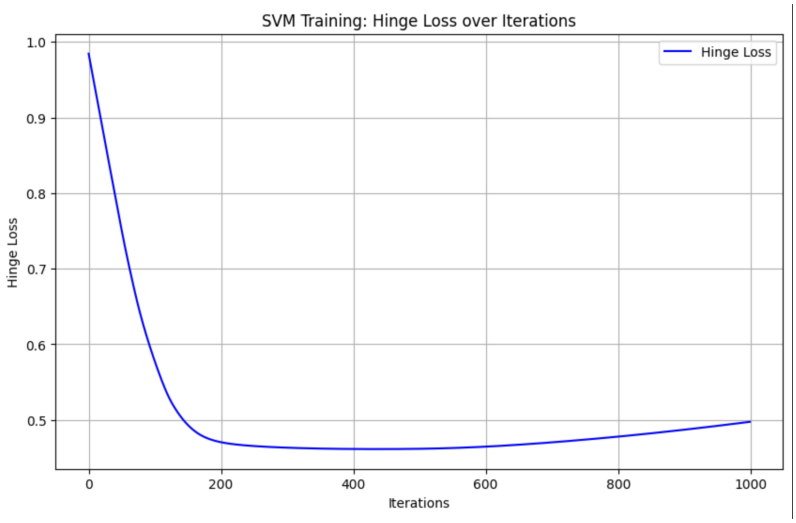

Figure 6.2: SVM hinge loss over iterations

6.6 Model 3: Naive Bayes

6.7 Naive Bayes Model

6.7.1 Introduction of the Model

- Naive Bayes is another effective probabilistic model used for classification tasks.
- Naive Bayes assumes that the features are independent of each other when the class label is given, which makes it computationally efficient, especially for large datasets.
- Implementation of the model uses Laplace smoothing to handle zero probabilities and trains on small batches of data at a time to optimize memory usage.

6.7.2 Core Ideas of the Model

- **Prior Probabilities:**
  - The prior probability for each class is calculated as:

$$P(\text{class}_c) = \frac{\text{Number of Samples in Class } c}{\text{Total Number of Samples}}$$

- **Likelihood Probabilities:**
  - The likelihood of each feature value given a class is calculated as:

$$P(x_j|\text{class}_c) = \frac{\text{Count of } x_j \text{ in class } c + 1}{\text{Total Samples in class } c + \text{Number of Unique Values in } x_j}$$

  - Laplace smoothing (+1 to each count) ensures that no probability is zero.
- **Posterior Probabilities:**
  - The posterior probability is proportional to the product of the prior and the likelihoods:

$$P(\text{class}_c|x) \propto P(\text{class}_c) \prod_{j=1}^{n} P(x_j|\text{class}_c)$$

  - The model predicts the class with the highest posterior probability.
- **Incremental Training:**
  - The dataset is divided into small batches of data, and the model is updated incrementally to handle large datasets.
- **Loss Calculation (Optional):**
  - The loss function computes the negative log likelihood of the true class posterior, averaged over all samples.

6.7.3 Performance

- **Evaluation Metric Calculations:**
  - The predicted values (y_{pred}) are compared to the actual labels (y_{test}) to compute the performance metrics.
  - The formulas for each metric are as follows:
    - **Accuracy:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

    - **Precision:**

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

    - **Recall:**

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

    - **F1-Score:**

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

- **Model Performance:**
  - The Naive Bayes model, trained using chunk-based updates, achieved the following metrics on the test dataset:
    - **Accuracy:** 79.17%
    - **Precision:** 76.62%
    - **Recall:** 79.17%
    - **F1-Score:** 75.01%

6.8 Model 5: Ensemble Model Using Maximum Voting

6.8.1 Introduction of the Ensemble Model

- The Ensemble Model is an implementation of an ensemble learning technique combining the following three models:
  - Support Vector Machine (SVM)
  - Logistic Regression
  - Naive Bayes
- Ensemble learning leverages the strengths of multiple models to produce a more robust and accurate prediction.
- This approach uses the Maximum Voting Algorithm for aggregating predictions from the individual models, ensuring balanced decision-making.

6.8.2 Core Ideas of the Ensemble Model

- **Maximum Voting Algorithm:**
  - In maximum voting, each model in the ensemble casts a "vote" for its predicted class.
  - The final prediction is determined by the class with the highest number of votes across all models.
  - This approach reduces the shortcomings of individual models by relying on their collective decision.
  - For example, if the predictions from the models for a data point are $[0, 1, 1]$, the final ensemble prediction will be 1, as it has the majority votes.
- **Implementation Steps:**
  1. **Predictions Gathering:**
     - Each model generates predictions for the test dataset.
  2. **Aggregating Predictions:**
     - Predictions are organized into a matrix where each row corresponds to predictions for a single sample across all models.
  3. **Majority Voting:**
     - For each sample, the class with the maximum votes is selected as the final prediction.
     - If there is a tie, the class with the smallest numerical value is chosen.
- **Advantages of Maximum Voting:**
  - Reduces the impact of individual model biases.
  - Improves overall generalization and performance.
  - Works well for binary and multi class classification tasks.

6.8.3 Performance

- **Evaluation Metric Calculations:**
  - The ensemble model predictions (y_{pred}) are compared to the actual labels (y_{test}) to compute the performance metrics.
  - The formulas for each metric are as follows:
    - **Accuracy:**

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

    - **Precision:**

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

    - **Recall:**

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

    - **F1-Score:**

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

- **Model Performance:**
  - The Ensemble Model, constructed using the Maximum Voting algorithm, achieved the following metrics on the test dataset:
    - **Accuracy:** 85.68%
    - **Precision:** 86.83%
    - **Recall:** 85.68%
    - **F1-Score:** 83.40%

Chapter 7

Results

We present the outcomes of our implementation of various machine learning models, including **Logistic Regression**, **Support Vector Machines (SVM)**, **Naive Bayes**, and an **ensemble model using the Maximum Voting algorithm**. The performance of these models was evaluated based on **Accuracy**, **Precision**, **Recall**, and **F1-Score**. These metrics were computed to assess the models' ability to correctly classify the data points and handle imbalances in the dataset.

The following table shows the testing results for each model:

Model	Accuracy	Precision	Recall	F1-Score
SVM	0.788361	0.790107	0.788361	0.789209
Logistic Regression	0.828824	0.859948	0.828824	0.786568
Naive Bayes	0.791730	0.766248	0.791730	0.750102
Ensemble Model	0.856798	0.868303	0.856798	0.833976

Table 7.1: Evaluation of SVM, Logistic Regression, Naive Bayes, and Ensemble Models on Test Data

- The Ensemble model, implemented using the Maximum Voting algorithm, achieved the highest accuracy among all models tested. By combining the predictions of all individual models, it improved performance, particularly in terms of Precision, Recall, and F1-Score.
- Logistic Regression also performed near to Ensemble model, performing well in both Precision and Recall, with a slightly lower F1-Score compared to the Ensemble model.
- The SVM and Naive Bayes models showed relatively lower performance. Naive Bayes had slightly lower Precision and F1-Score, while SVM has a better balance between Precision and Recall.

Chapter 8

Conclusion

In our project we have used many machine learning models for evaluation for a classification problem, such as Logistic Regression, Support vector machine (SVM), Naive Bayes, and an ensembled model which uses Maximum Voting Method.

The following inferences were being made by the results we got:

- **Ensemble Model**: The ensemble model outperformed all the other models we have used, by achieving the highest Accuracy, Precision, Recall and F1-Score. This shows how ensembling effectiveness in improving the overall performance.
- **Logistic Regression**: Logistic regression performed well just showed a bit less F1-Score in comparision to the ensembled mode, suggesting a trade off between Recall and Precision.
- **SVM**: SVM showed a balanced performance but it fell behind the ensembled and LR models in overall F1-score.
- **Naive Bayes**: Naive Bayes achieved the lowest performance across most of the metrics, and in particular Precision.

Overall conclusion is, the ensemble model comes out to be the most reliable, which shows the benefits of combining diverse models.

Future work could explore hyperparameter tuning and other ensembing models to further improve results.

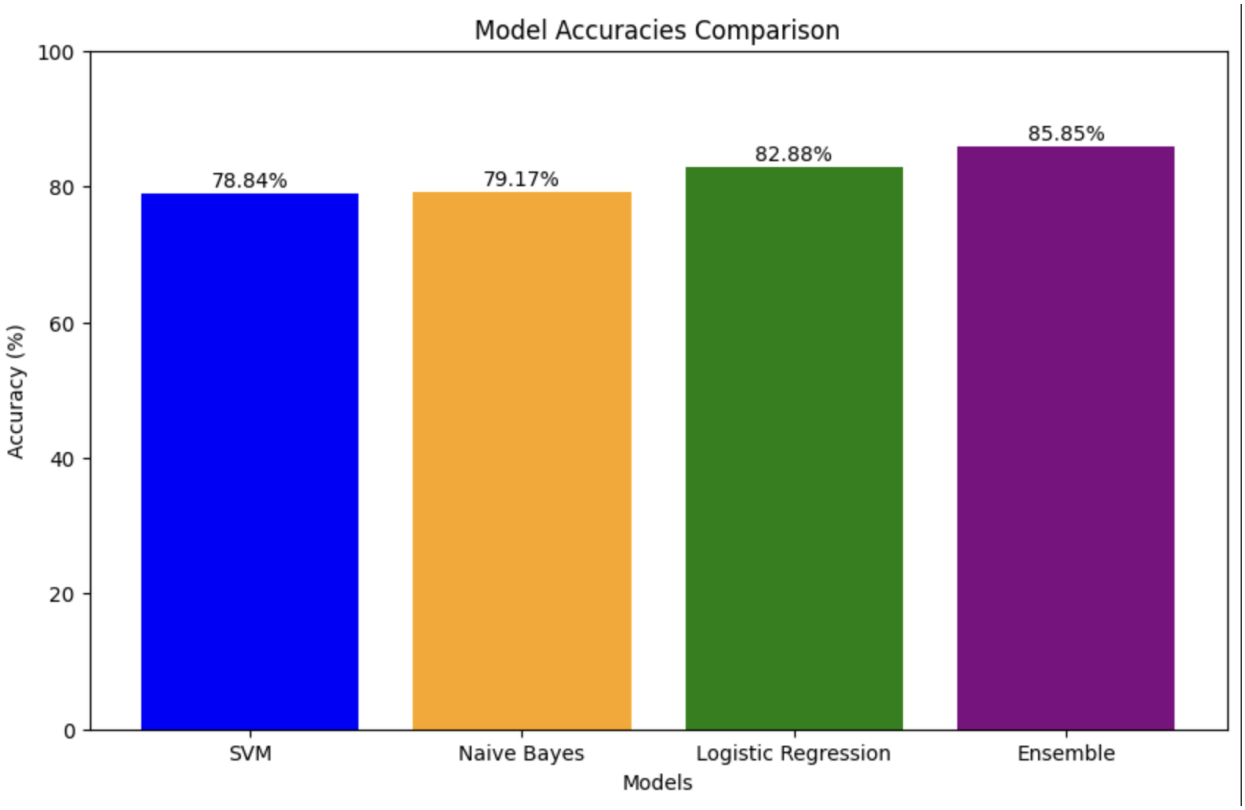

Figure 8.1: Models Comparision

Chapter 9

Bibliography

1. Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Comprehensive coverage of Logistic Regression, Support Vector Machines, and other supervised learning techniques.

2. Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press. Detailed explanation of probabilistic models like Naive Bayes and their applications.

3. Dietterich, T. G. (2000). "Ensemble Methods in Machine Learning". *International Workshop on Multiple Classifier Systems*. Springer. Foundational work on ensemble techniques, including voting and bagging.

4. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media. Practical guide for implementing machine learning models using Scikit-learn and ensemble methods.

5. Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research*, 12, 2825-2830. Scikit-learn library documentation used for model implementation.

6. Scikit-learn Documentation. Resource for model development, scaling, and PCA. Available at: https://scikit-learn.org/

7. Seaborn Library Documentation. Used for exploratory data analysis and visualizations. Available at: https://seaborn.pydata.org/