University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project

7PAM2002-0901-2024

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Predicting Home Loan Sanction Using Machine Learning

Student Name and SRN:

Bharanidharan Thirumaran - 22076992

Supervisor: Vidas Regelskis

Date Submitted: 06/01/2025

Word Count: 7469

GitHub address: https://github.com/Bharanimaran/Msc_Dissertation_Project

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at Assessment Offences and Academic Misconduct and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6).

I have not used chatGPT, or any other generative AI tool, to write the report or code (other than where declared or referenced).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student SRN number: 22076992

Student Name printed: Bharanidharan Thirumaran

Student signature: 

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

Abstract:

This study focuses on building a reliable machine learning system to predict whether a loan application will be approved, using information about applicant's demographics, financial details, and other relevant factors. The main goal is to simplify and improve the loan approval process in financial institutions by automating decision-making through machine learning models. The research compares the performance of three widely used models: Logistic Regression, Random Forest and Support Vector Machines (SVM). It also identifies key factors like Credit History, Applicant Income, and Loan Amount that play a significant role in loan approval decisions.

The dataset used in this project contains anonymized records of loan applicants, including essential details that influence approval outcomes. To prepare the data for analysis, steps like handling missing values, scaling numerical data, and converting categorical information into numbers were taken. The models were further optimized using advanced methods such as hyperparameter tuning and feature importance analysis to improve their accuracy and reliability.

The results showed that advanced model like Random Forest performed better at identifying complex patterns in the data compared to simpler models. Credit History was found to be the most critical factor in determining loan approval, followed by Applicant Income and Loan Amount. These findings offer meaningful insights into how financial institutions can make faster, more accurate, and fairer loan decisions by using machine learning.

This study also highlights the importance of ethical considerations, such as protecting applicant's data and ensuring fairness in decision-making. The recommendations provided can help organizations deploy machine learning models effectively in real-world loan approval systems. Future research can expand on this work by using larger datasets, exploring additional features, and testing newer machine learning models to improve accuracy and scalability further.

Table of Contents

1.Introduction:

Loan approval decisions are crucial in the financial industry, determining access to resources for individuals and businesses. Traditionally, these decisions have relied on manual assessments, which, while functional, are often slow, inconsistent, and prone to human error or bias. With growing demands for efficiency and fairness, financial institutions are exploring innovative solutions to modernize their processes. This study focuses on using machine learning as a transformative tool to streamline loan approvals. Machine learning offers a data-driven approach to automate decisions, identifying patterns in applicant data such as credit history, income, and loan amount that might otherwise be missed. Models like Logistic Regression, Random Forest, and Support Vector Machines (SVM) each bring distinct strengths, from simplicity and interpretability to handling complex non-linear relationships.

Beyond improving efficiency and accuracy, this research highlights the ethical considerations of automating loan approvals. Transparency, fairness, and accountability are essential to prevent biases present in historical data from being reinforced. By balancing technical innovation with ethical best practices, this study provides practical and scalable solutions for financial institutions. Ultimately, it demonstrates how machine learning can revolutionize loan approvals, enabling faster, fairer, and more reliable decisions that benefit both lenders and applicants.

2. Background:

Loan approval decisions are critical in shaping the relationship between financial institutions and their customers. Traditionally, these decisions relied on manual assessments based on factors like credit history, income, and demographics. While effective in smaller settings, manual processes are often slow, inconsistent, and susceptible to human error or bias. Machine learning offers a more efficient, accurate, and fair alternative by uncovering complex patterns in data that traditional methods might miss.

Machine learning models, such as Logistic Regression, Random Forest, and Support Vector Machines (SVM), can process large volumes of applications quickly and consistently. They excel at identifying relationships between factors like credit history and income, making data-driven decisions faster and more scalable. However, ethical considerations, such as avoiding biases in data and ensuring transparency, remain vital to building trust. This study bridges traditional loan approval methods with modern machine learning approaches, focusing on accuracy, fairness, and transparency to create smarter and more inclusive financial systems.

3. Aim:

The aim of this project is to develop a machine learning framework for predicting loan approval status using demographic, financial, and relevant applicant features. By implementing and comparing multiple machine learning models including Logistic Regression, Random Forest and Support Vector Machines (SVM), the project seeks to enhance the efficiency and accuracy of loan

approval processes. The ultimate goal is to identify the most effective model while offering insights into the key factors influencing loan sanction decisions.

4. Objectives:

1. To analyze and preprocess the loan dataset, addressing missing values, encoding categorical variables, and scaling numerical data.

2. To explore and identify relationships between applicant features and loan approval status through exploratory data analysis (EDA).

3. To train and implement machine learning models (Logistic Regression, Random Forest and SVM) for loan approval prediction.

4. To evaluate and compare the performance of these models using metrics such as accuracy, precision, recall, and F1-score.

5. To determine the most significant predictors of loan approval, such as Credit History and Applicant Income, through feature importance analysis.

6. To provide recommendations for deploying machine learning techniques in real-world financial systems to enhance decision-making processes.

5. Literature Review:

- **Loan Prediction Using Machine Learning, Sarisa, H.K., Khurana, V., Koti, V.C. & Garg, N. (2023). 2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT). IEEE.**
  Sarisa et al. explore the use of machine learning, specifically logistic regression, in predicting loan approvals by integrating applicants' CIBIL scores. They show that incorporating financial indicators like credit scores greatly enhances the model's predictive accuracy, enabling banks to make faster and more accurate loan decisions, thus reducing risks and improving operational efficiency.
- **Machine Learning Models for Predicting Bank Loan Eligibility, Orji, U.E., Ugwuishiwu, C.H., Nguemaleu, J.C.N. & Ugwuanyi, P.N. (2022). IEEE NIGERCON. IEEE.**
  Orji et al. analyze six machine learning models for loan eligibility prediction, identifying Random Forest as the most effective with a 95.5% accuracy rate. Their work underscores the strength of ensemble models, particularly in handling complex patterns within loan data, making Random Forest a preferred choice in banking applications.
- **Machine Learning Based Loan Eligibility Prediction Using Random Forest Model, Reddy, C.S., Siddiq, A.S. & Jayapandian, N. (2022). 2022 Seventh International Conference on Communication and Electronics Systems (ICCES). IEEE.**
  Reddy et al. adopt a Random Forest model to predict loan eligibility, highlighting its accuracy, especially in high-dimensional datasets. Their findings illustrate the effectiveness

of ensemble methods like Random Forest, making it a reliable model for banks seeking to optimize loan approval workflows and reduce potential errors.

- **Loan Prediction Using Machine Learning and Its Deployment on Web Application, Gudipalli, A., Sujatha, C.N., Pushyami, B.H., Narra, N.K. & Sanjana, B.N. (2021). Innovations in Power and Advanced Computing Technologies (i-PACT). IEEE.** Gudipalli et al. developed a loan prediction system combining logistic regression and K-Nearest Neighbors (KNN), deployed as a web application. This web-based system supports real-time loan assessments, allowing banks to make quicker, data-driven decisions, thereby improving usability and operational efficiency.
- **Loan Approval Prediction System Using Logistic Regression and CIBIL Score, Kadam, E., Gupta, A., Jagtap, S., Dubey, I. & Tawde, G. (2023). 2023 Fourth International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE.** Kadam et al. leverage logistic regression for loan prediction, emphasizing the role of CIBIL scores and other financial metrics in enhancing prediction accuracy. Their study demonstrates the practical application of logistic regression in banks, simplifying loan approvals by integrating critical financial data, thereby aiding in risk and operational assessments.
- **Prediction of the Approval of Bank Loans Using Various Machine Learning Algorithms, Rahman, A.T., Purno, M.R.H. & Mim, S.A. (2023). 2023 IEEE World Conference on Applied Intelligence and Computing (AIC). IEEE.** Rahman et al. provide a comparative analysis of multiple machine learning algorithms, including Decision Trees and Gradient Boosting, with Random Forest emerging as the most reliable for bank loan prediction. Their findings emphasize the importance of model selection in high-stakes financial decisions, showcasing Random Forest as a resilient choice for accurate loan approvals.

  Machine learning has transformed loan approval processes, surpassing traditional methods. Models like Logistic Regression, Random Forest, and SVM effectively analyze financial and demographic data, enhancing accuracy and fairness. Random Forest excels at capturing non-linear patterns, while Logistic Regression is valued for simplicity. Credit History, Applicant Income, and Loan Amount are key predictors, with Credit History being the strongest. Challenges like data imbalances and model interpretability remain. This study emphasizes transparency and employs robust methods to improve accuracy, efficiency, and fairness in loan predictions.

6. Ethical Considerations:

This study prioritized ethical considerations to ensure integrity, fairness, and reliability throughout the research process.

6.1 Data Privacy

- Sensitive applicant information was anonymized and handled securely in compliance with GDPR.
- Data was stored and processed securely, employing encryption and restricted access to authorized personnel only.

6.2 Bias Mitigation

- Techniques like SMOTE were applied to address class imbalances, ensuring fair evaluation of loan applications.
- Feature importance analysis helped identify potential biases in predictors, promoting equitable treatment for all groups.
- The study avoided reliance on variables that could perpetuate discrimination.

6.3 Transparency and Interpretability

- Transparency was maintained in data preprocessing, model selection, and evaluation for reproducibility.
- Efforts were made to ensure model predictions were interpretable, fostering trust and accountability.
- Findings were documented and presented clearly, highlighting both model strengths and limitations.

6.4 Responsible Reporting

- Results were reported objectively, avoiding misinterpretation or undue influence on market behavior.
- Both strengths and limitations of the study were emphasized to ensure balanced analysis.

6.5 University of Hertfordshire Ethical Standards

- The study adhered to the University of Hertfordshire's guidelines for responsible and ethical research conduct.

7. Data Collection and Preprocessing

7.1 Data Collection

Collecting the right dataset is a critical starting point for any machine learning project, and this study was no exception. The dataset used for this project was sourced from a publicly available

repository, containing 614 loan applications with 13 distinct features. These features provided a well-rounded snapshot of the factors that influence loan approval decisions, such as the applicant's financial background, credit history, and demographic details.

The dataset included the following key attributes:

- Loan_ID: A unique identifier for each application.
- Gender: The gender of the primary applicant (e.g., Male, Female).
- Married: The marital status of the applicant (e.g., Yes, No).
- Dependents: The number of dependents supported by the applicant.
- Education: The applicant's education level (Graduate or Not Graduate).
- Self_Employed: Whether the applicant is self-employed (Yes or No).
- ApplicantIncome: The monthly income of the primary applicant.
- CoapplicantIncome: The monthly income of a co-applicant, if applicable.
- LoanAmount: The loan amount requested, expressed in thousands.
- Loan_Amount_Term: The repayment period for the loan, given in months.
- Credit_History: Whether the applicant has a favorable credit history (1 for Yes, 0 for No).
- Property_Area: The location of the property (e.g., Urban, Semiurban, Rural).
- Loan_Status: The target variable, indicating whether the loan was approved (1) or rejected (0).

This dataset was chosen for its relevance and ability to represent a real-world scenario, where applicants' demographic and financial profiles play a crucial role in determining loan eligibility. Ethical considerations were maintained during the collection process, ensuring compliance with privacy standards. While the data did not include personal identifiers, it was treated with care to uphold anonymity and confidentiality.

7.2 Data Preprocessing

Once the dataset was collected, it underwent thorough preprocessing to prepare it for analysis and machine learning model training. This step was essential to clean the data, handle inconsistencies, and transform it into a format that could be easily used by algorithms.

7.2.1 Handling Missing Values

Missing values can significantly affect the performance of machine learning models, so addressing them was a priority. During the initial analysis, features like Gender, Married, Dependents, and Loan Amount were found to have gaps. These gaps were handled as follows:

- Mode Imputation: For categorical variables like Gender, Married, and Dependents, the missing values were filled with the most frequently occurring value (mode). This ensured consistency while preserving the overall distribution of the data.

- Mean Imputation: For numerical variables like Loan Amount, the missing values were replaced with the mean. This approach retained the central tendency of the data without introducing significant biases.

By filling in the missing data, the dataset was made complete, allowing all rows to be utilized in the analysis and modeling stages.

7.2.2 Encoding Categorical Features

Since machine learning models work best with numerical data, all categorical features were converted into numerical representations:

- Label Encoding: Simple binary categories, such as Gender and Self_Employed, were converted into numbers (e.g., Male = 0, Female = 1).

- One-Hot Encoding: For features with more than two categories, such as Property_Area, separate binary columns were created for each category. This avoided implying any order or ranking between categories.

This transformation ensured that all data could be processed by machine learning algorithms while preserving the meaning of the categorical features.

7.2.3 Scaling Numerical Data

The numerical features in the dataset, such as Applicant Income, Coapplicant Income, and Loan Amount, had widely varying ranges. For instance, income values were much larger than loan amounts, which could lead to certain features dominating the model's training process. To address this:

- Standard Scaler was used to standardize these features, transforming them to have a mean of zero and a standard deviation of one. This scaling made all features equally important and ensured that the models performed optimally.

Additional Steps

- Removing Redundant Columns: Features like Loan_ID, which were unique to each application and irrelevant for prediction, were removed to streamline the dataset.

- Splitting the Dataset: The dataset was split into training and testing sets (80-20 split) to evaluate the models' performance on unseen data.

8. Methodology

8.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was conducted to gain deeper insights into the relationships, trends, and patterns within the dataset. The process involved detailed visualizations to explore numerical and categorical features, ensuring a clear understanding of the variables influencing loan approval. Below are the detailed insights derived from the visualizations.

8.1.1 Correlation Matrix

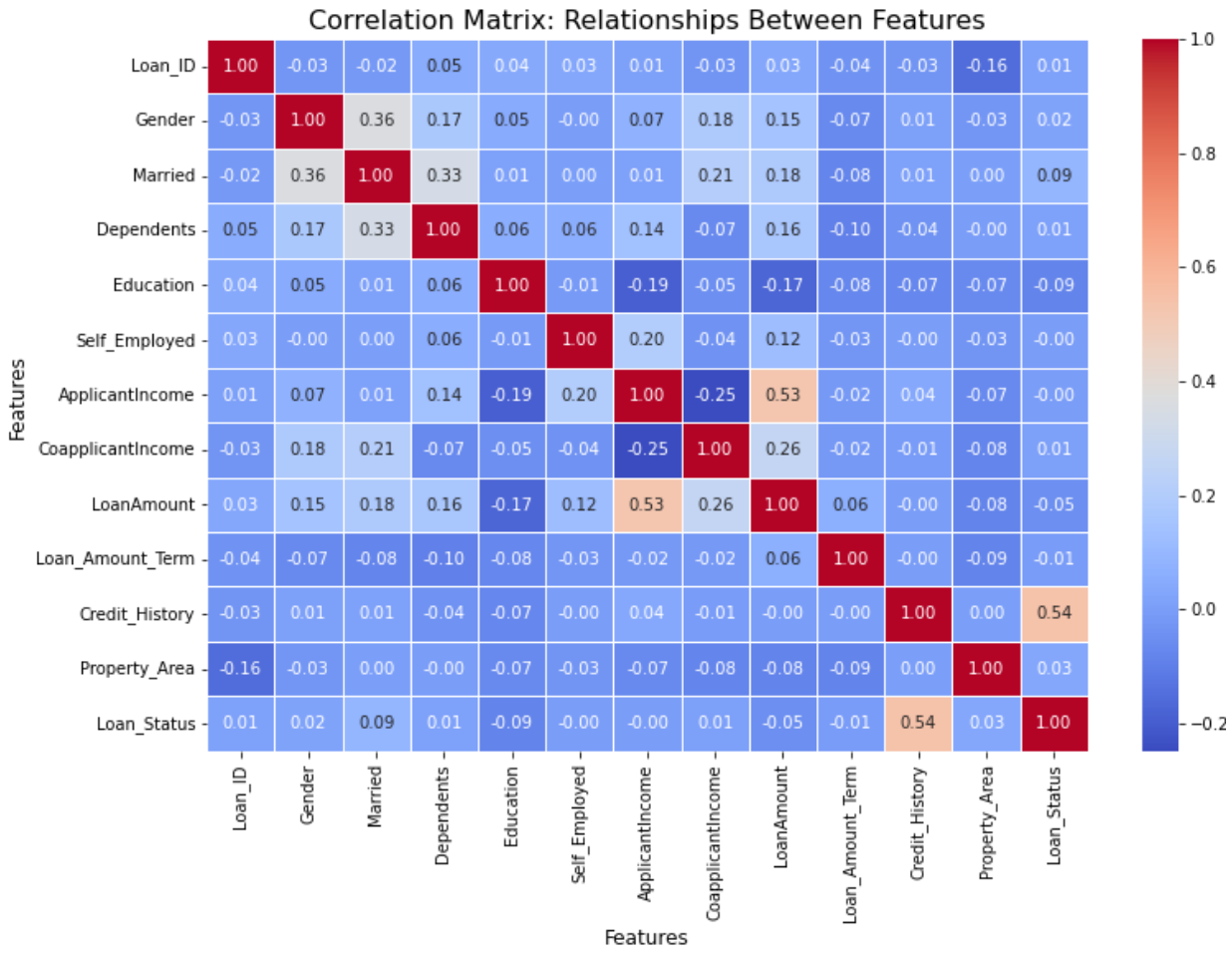

Figure 1

The correlation matrix heatmap (Figure 1) highlights key relationships among numerical features in the dataset:

1. **Credit History and Loan Status:**
    - **Correlation Value: 0.54**
    - Credit History is the strongest predictor of loan approval, with a positive correlation of 0.54. Good credit history significantly increases approval likelihood, reflecting repayment reliability.
2. **Applicant Income and Loan Amount:**
    - **Correlation Value: 0.53**
    - A moderate correlation of 0.53 shows that higher-income applicants tend to apply for larger loans, reflecting financial flexibility. This relationship indirectly highlights borrowing behavior.
3. **Minimal Relationships with Loan Status:**
    - **Gender:** Correlation of 0.03 shows no gender bias in loan decisions.
    - **Education:** Correlation of 0.01 indicates education level has negligible impact on loan approval.
    - **Property Area:** Correlation of 0.03 suggests the applicant's location does not significantly influence approval.
4. **Negative Relationships and Other Insights:**
    - **Loan Amount Term and Loan Amount:** Correlation of -0.25 indicates weak inverse relationships, suggesting shorter terms for larger loans.
    - **Gender and Credit History:** Slight negative correlation of -0.16 hints at subtle trends but has minimal overall impact.

8.1.2 Education vs. Average Applicant Income

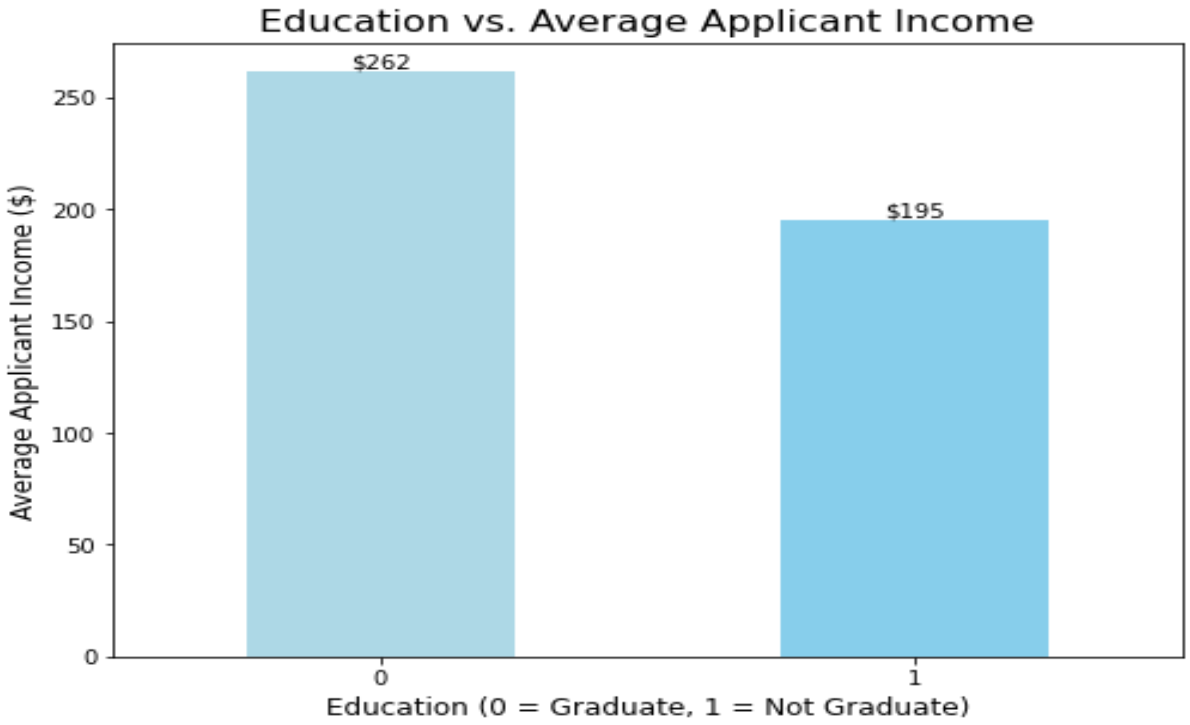

Figure 2

The bar chart in (Figure 2) compares average applicant income by education levels: Graduate (0) and Not Graduate (1), highlighting how education impacts earning potential.

1. **Graduate Applicants (Category 0):**
    - Average income is £262, notably higher than non-graduates, indicating better job opportunities and earning capacity for graduates.
    - Higher income reflects stronger financial stability, making graduates more attractive to lenders as lower-risk candidates for loan approvals.
2. **Non-Graduate Applicants (Category 1):**
    - Average income is £195, significantly lower than graduates, showcasing financial challenges non-graduates might face, potentially affecting their loan repayment capabilities.
3. **Income Gap:**
    - The £67 gap between graduates (£262) and non-graduates (£195) highlights the link between education and earning potential.

- While education has a weak correlation with loan status, its effect on income indirectly influences loan approval, as higher incomes improve financial stability and approval likelihood.

8.1.3 Credit History vs. Loan Amount by Loan Status

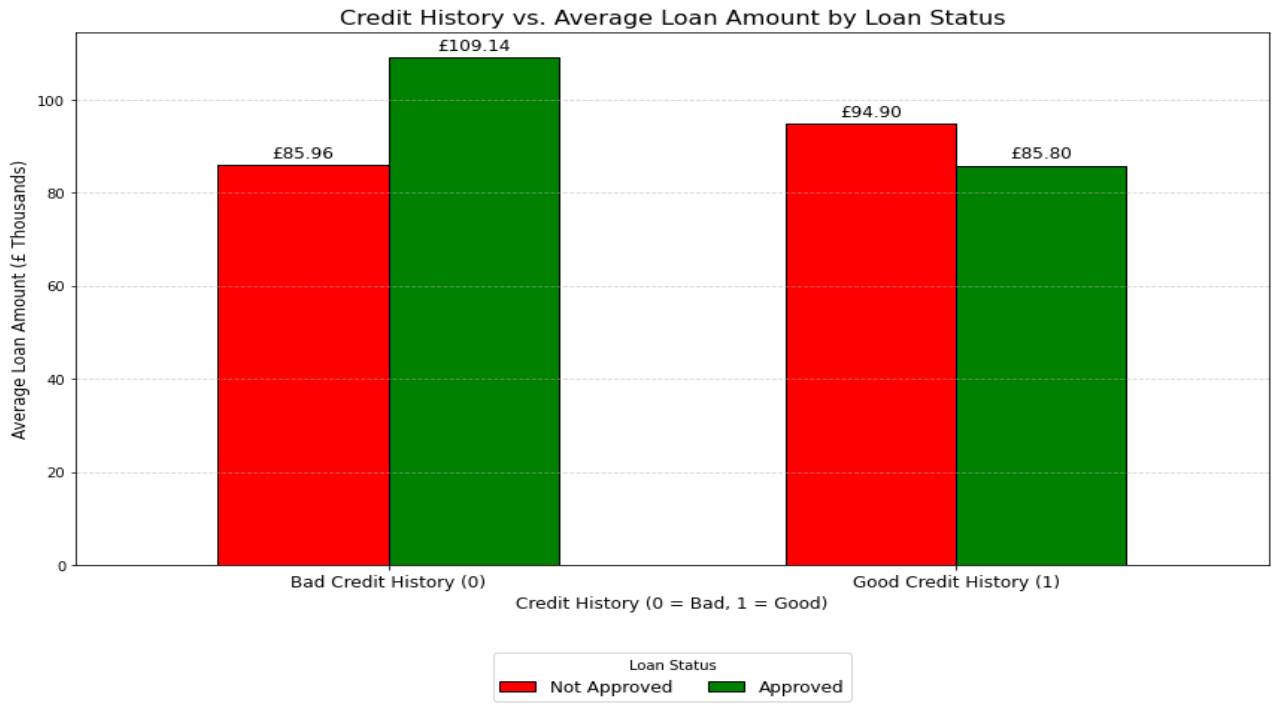

Figure 3

Figure 3 explores the relationship between Credit History, Loan Amount, and loan approval likelihood, distinguishing between Approved (green) and Not Approved (red) applicants across credit history groups.

1. **Good Credit History (Credit History = 1):**
    - **Approved Loans:** Average loan amount is £109.14, reflecting lenders' confidence in granting larger loans to applicants with positive financial records.
    - **Not Approved Loans:** Average loan amount is £85.96, significantly lower than approved loans, showing that factors like income or loan terms also influence rejections.
    - **Insight:** The difference between approved (£109.14) and not approved (£85.96) loan highlights how a strong credit history enhances approval chances, especially for higher-value loans.

2. **Bad Credit History (Credit History = 0):**
    - **Approved Loans:** Average loan amount is £85.80, much lower than for good credit history, indicating cautious lending to applicants with poor credit records.
    - **Not Approved Loans:** Average loan amount is £94.90, slightly higher than approved loans, suggesting stricter scrutiny for larger loan requests.
    - **Insight:** The higher rejected loan average (£94.90) compared to approved (£85.80) reflects stringent lending criteria for applicants with bad credit.

This analysis emphasizes the importance of credit history in loan approval, with good credit increasing approval chances and bad credit leading to stricter evaluations, particularly for higher loan amounts.

8.1.4 Property Area vs. Average Applicant Income by Loan Status

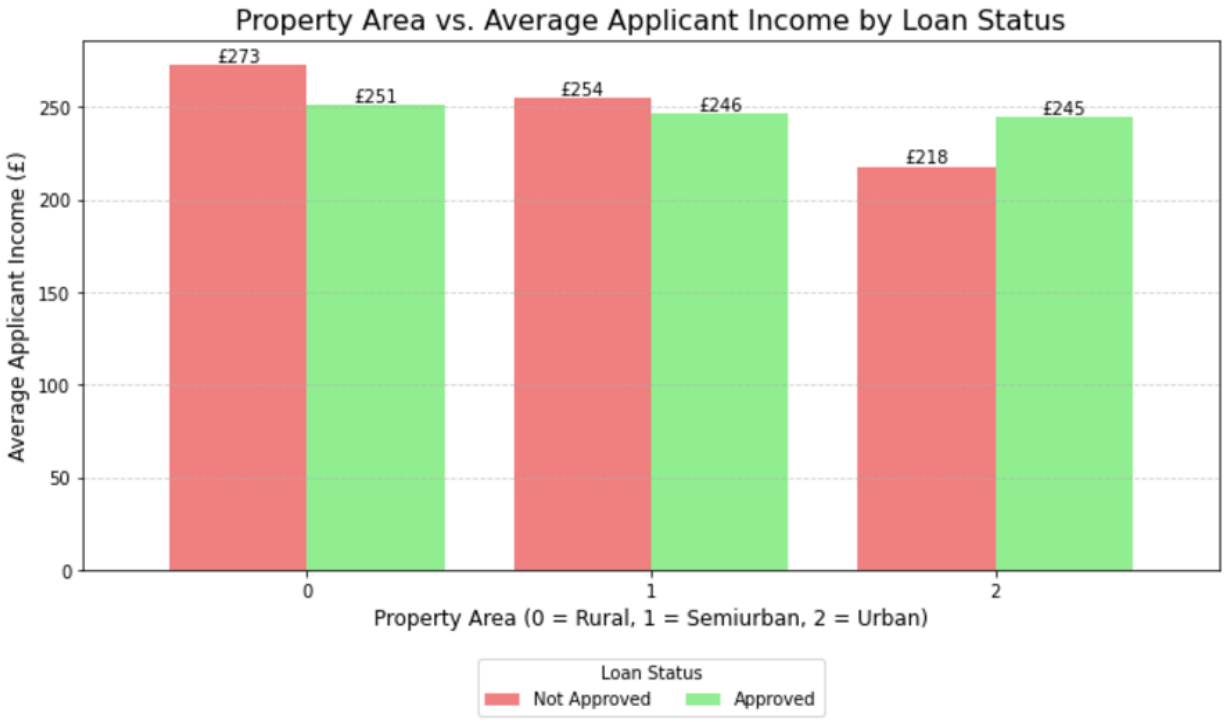

Figure 4

Figure 4 illustrates the relationship between Property Area, Applicant Income, and loan approval. It compares applicants from Urban (2), Semiurban (1), and Rural (0) areas, distinguishing between Approved (green) and Not Approved (red) loans.

1. **Urban Areas (Property Area = 2):**
    - **Approved Loans:** Average income £251, indicating financial stability contributing to approvals.
    - **Not Approved Loans:** Average income £273, showing income alone isn't sufficient, as factors like Credit History or Loan Amount also play significant roles.
2. **Semiurban Areas (Property Area = 1):**
    - **Approved Loans:** Average income £246, slightly lower than urban areas but strong enough for loan approval.
    - **Not Approved Loans:** Average income £254, indicating factors other than income (e.g., credit history) contributed to rejections.
3. **Rural Areas (Property Area = 0):**
    - **Approved Loans:** Average income £245, comparable to semiurban but slightly lower than urban areas, showing modest incomes still met loan criteria.
    - **Not Approved Loans:** Average income £218, significantly lower, highlighting income as a critical factor in rural rejections.
4. **Comparative Insights:**
    - **Urban Areas:** Higher incomes (£273) in rejections suggest lenders prioritize risk factors like creditworthiness over income.
    - **Semiurban Areas:** Slightly higher incomes for rejections (£254) compared to approvals (£246) reinforce that income isn’t the sole determinant.
    - **Rural Areas:** Lower incomes (£218) strongly correlate with rejections, while higher incomes (£245) improve approval chances, showing income plays a more decisive role in rural areas.

8.1.5 Loan Status vs. Applicant Income Distribution (Box Plot)

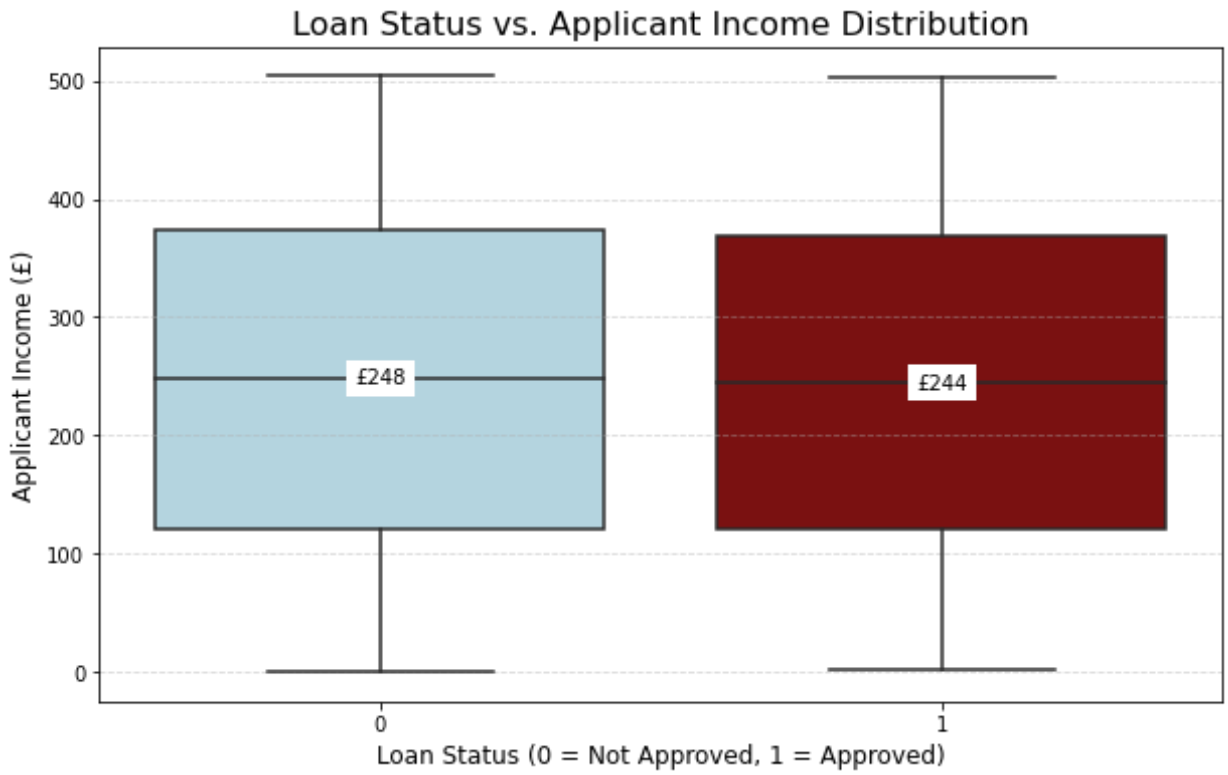

Figure 5

The box plot (Figure 5) compares Applicant Income for approved (1) and not approved (0) loans, offering insights into its role in loan approval decisions:

Median Income:

- Approved Loans: Median income is £248.
- Not Approved Loans: Median income is £244.

While approved applicants have a slightly higher median income, the small difference suggests that income alone does not significantly influence loan approval.

Income Range:

Both approved and not approved groups show a similar range of incomes, with considerable overlap. This indicates that loans are approved or rejected across various income levels, depending on other factors.

Outliers:

Both groups include high-income outliers, some of whom were rejected. This highlights that high income alone doesn't guarantee approval if other eligibility criteria, such as Credit History or Loan Amount, are not favorable.

Key Observations from EDA

1. Credit History:
    - Credit History is the most influential factor, with a correlation of 0.54 to Loan Status.
    - Applicants with good credit history are approved for larger loans (£109.14) compared to those with bad credit history (£85.80).
    - Rejected loans for bad credit history average £94.90, showing stricter scrutiny for higher loan requests.
2. Applicant Income:
    - Graduates earn an average of £262, while non-graduates earn £195 – a £67 gap.
    - Median income for approved applicants is £248 vs. £244 for rejected ones, showing minimal difference.
    - Income alone does not guarantee approval, as overlapping ranges exist for approved and rejected applicants.
3. Loan Amount:
    - Approved loans are concentrated in the lower-to-mid range.
    - Rejected loans show a wider distribution, with larger loan amounts more prone to rejection.
4. Property Area:
    - Urban Areas: Approved income is £251, rejected income is £273.
    - Semiurban Areas: Approved income is £246, rejected income is £254.
    - Rural Areas: Approved income is £245, rejected income is £218.

- Higher income in urban areas does not guarantee approval, reinforcing the importance of other factors.

5. Income Distribution:

    - Approved loans have a slightly higher median income (£248) than rejected loans (£244).

    - Overlapping ranges show income alone is not a decisive factor.

8.2 Feature Engineering

Feature engineering is a critical process that transforms raw data into meaningful inputs for machine learning algorithms. By improving the quality of features, models can better identify patterns, leading to higher accuracy and performance. For this project, a systematic approach was used to clean, transform, and enhance the dataset. Below is a detailed explanation of the key steps performed during feature engineering:

1. Removing Irrelevant Features

In any dataset, not all features contribute to the predictive power of machine learning models. Some features are purely informational or redundant and may even introduce noise into the training process.

- Loan_ID:

    - This column served as a unique identifier for each loan application and carried no predictive value. It was removed as it would not contribute to the learning process of the model.

    - Retaining it could lead to unnecessary computational overhead without improving the model's accuracy.

By dropping such irrelevant features, the dataset was streamlined to focus only on meaningful inputs that could impact loan approval outcomes.

2. Encoding Categorical Variables

Machine learning algorithms require numerical input; hence, categorical variables in the dataset needed to be converted into numerical representations. Two encoding methods were applied based on the nature of the categorical feature:

- **Label Encoding (For Binary Categorical Features):**

    - Label encoding was used for binary categories, where each category was mapped to a numerical value (0 or 1).

    - Examples:

        - Gender: Male → 0, Female → 1
        - Education: Graduate → 0, Not Graduate → 1
    - This approach ensured that binary features retained their simple structure without introducing additional dimensions.

- **One-Hot Encoding (For Multi-Class Features):**
    - Multi-class variables like Property Area, which had three categories (Urban, Semiurban, and Rural), were converted into separate binary columns using one-hot encoding.
    - Example transformation:

Original Property Area	**One-Hot Encoded Columns**
Urban	[1, 0, 0]
Semiurban	[0, 1, 0]
Rural	[0, 0, 1]

This transformation created distinct binary columns for each class while preventing the introduction of misleading ordinal relationships between categories.

3. Handling Class Imbalance

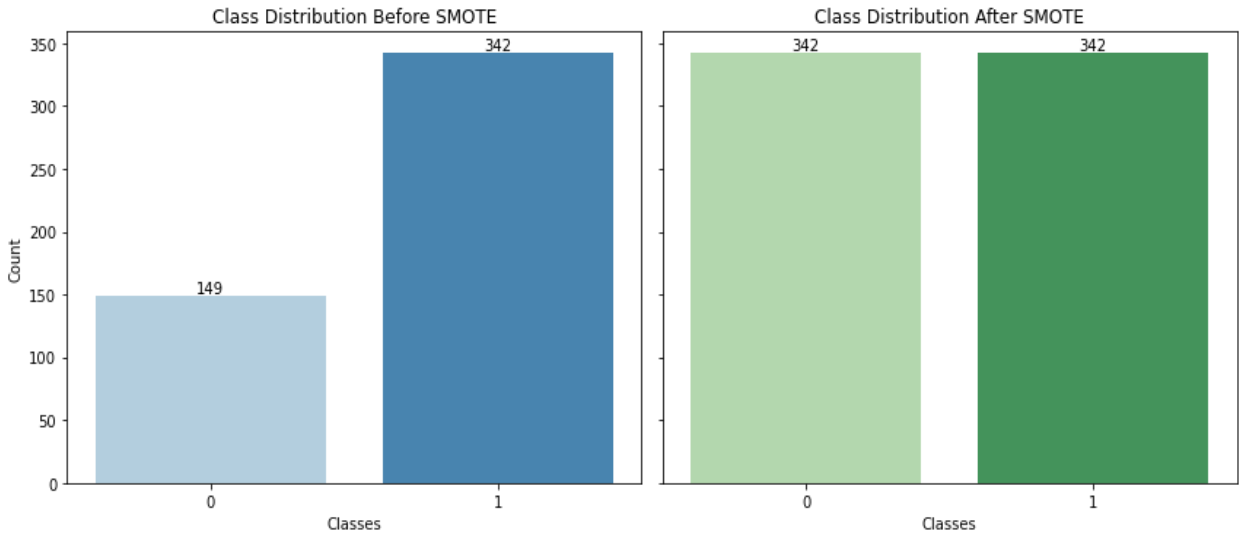

Figure 7

Figure 7 shows the target variable Loan Status exhibited class imbalance, where approved loans significantly outnumbered rejected loans. Training a model on imbalanced data can lead to biased predictions, where the majority class dominates. To address this:

- SMOTE (Synthetic Minority Oversampling Technique) was applied:
    - SMOTE works by generating synthetic examples for the minority class (rejected loans) rather than simply duplicating them.
    - It creates synthetic data points by interpolating between existing samples, ensuring a balanced dataset without overfitting to repeated data.

4. Creating New Features

To enhance the predictive power of the dataset, new features were engineered by combining or transforming existing variables. These derived features aimed to capture additional relationships within the data that might not have been obvious initially:

- Income-to-Loan Ratio:
    - This feature was calculated to measure an applicant's ability to afford the requested loan:
        - Income-to-Loan Ratio = Applicant Income / Loan Amount
    - Purpose: A higher ratio indicates that an applicant's income is sufficient relative to the loan amount, signaling lower risk to lenders.

- Co-applicant Contribution:
    - This feature assessed the proportion of income contributed by the co-applicant relative to the total household income:
      Co-applicant Contribution = Co-applicant Income / (Applicant Income + Co-applicant Income)
    - Purpose: This feature highlights the financial role of the co-applicant in supporting the loan repayment. Even if the primary applicant has a low income, a significant contribution from the co-applicant can positively impact approval chances.

5. Scaling Numerical Features

Numerical features often have different ranges of values, which can cause machine learning models to assign disproportionate importance to features with larger scales. To address this issue:

- Standard Scaler was applied to normalize the numerical features:
    - Standardization transforms each numerical value to have a mean of 0 and a standard deviation of 1:

$$z = \frac{x - \mu}{\sigma}$$

      where x is the original value, μ is the mean, and σ is the standard deviation.

- Features scaled included:
    - Applicant Income
    - Loan Amount
    - Loan Amount Term

8.3 Model Selection

Selecting the right machine learning models is a critical step in achieving robust and reliable predictions. This project leveraged four widely used machine learning algorithms, chosen for their distinct strengths:

1. Logistic Regression:
    - As a simple and interpretable linear model, Logistic Regression served as the baseline for comparison. It provided a clear understanding of feature importance and helped establish a performance benchmark.
2. Random Forest:

- This ensemble learning technique combines multiple decision trees to improve predictive accuracy and reduce overfitting. Random Forest was particularly suited to handling non-linear relationships and variable importance analysis.

3. Support Vector Machines (SVM):

    - SVM was employed to capture complex, non-linear relationships in the data. By using different kernel functions (linear, polynomial, and radial basis), SVM aimed to separate data points effectively in high-dimensional spaces.

Each model was trained using an 80-20 train-test split, ensuring fair evaluation. Hyperparameter tuning was applied to optimize performance, using techniques such as Grid Search to identify the best combinations of model parameters.

8.4 Evaluation Metrics

To comprehensively evaluate and compare the performance of the selected models, multiple metrics were employed:

1. Accuracy:

    - Represents the proportion of correctly classified instances out of the total. While a useful metric, accuracy can be misleading for imbalanced datasets, necessitating additional metrics.

2. Precision:

    - Focuses on the proportion of correctly predicted positive instances (approved loans) out of all predicted positives. This is crucial when the cost of false positives is high.

3. Recall:

    - Measures the proportion of actual positives (approved loans) that were correctly predicted. High recall ensures that most approved loans are identified by the model.

4. F1-Score:

    - Combines precision and recall into a single metric, providing a balanced evaluation even when there's a trade-off between the two.

5. Learning Curves:

    - Used to analyze training and validation performance over different dataset sizes, revealing potential issues like overfitting or underfitting.

9. Model Training and Results

The model training process involved evaluating various machine learning algorithms to determine the most effective for predicting loan approval status. Each model was trained on the processed dataset, and the results were analyzed using multiple performance metrics, such as accuracy, precision, recall and F1-score. Below is a detailed overview of the training process and outcomes for each model:

9.1 Logistic Regression

Logistic Regression was selected as the baseline model because of its simplicity, interpretability and efficiency in binary classification tasks like predicting loan approval. It works by estimating the probability that an applicant's loan will be approved (class 1) or rejected (class 0) using a linear relationship between the input features. As a foundational model, it serves as a reliable starting point for understanding the dataset and evaluating initial performance.

9.1.1 Training Process:

The following steps were followed to train and evaluate the Logistic Regression model:

1. Data Preprocessing:
    - All numerical features were normalized to ensure they had a uniform scale. This was essential for maintaining model performance since Logistic Regression is sensitive to varying ranges in feature values.
2. Cross-Validation:
    - To ensure the model generalizes well to unseen data, cross-validation was applied. This technique splits the dataset into multiple subsets, where the model is repeatedly trained and validated.
    - This process reduced the risk of overfitting and provided a robust performance estimate.
3. Hyperparameter Tuning:

```
# Logistic Regression Hyperparameter Tuning
log_reg = LogisticRegression(max_iter=1000, random_state=42)
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],  # Regularization strength
    'penalty': ['l1', 'l2'],  # Type of regularization
    'solver': ['liblinear', 'saga']  # Solvers compatible with l1/l2
}
grid_search_lr = GridSearchCV(log_reg, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search_lr.fit(X_train, y_train)
```

- Grid Search was used to optimize key hyperparameters, such as:
    - Regularization Strength (C): Controlled the trade-off between bias and variance. A lower C value encouraged stronger regularization, improving generalization.
    - Penalty Type (L2): L2 regularization was selected as it penalizes large feature weights, promoting a more balanced model.
- The best combination of hyperparameters identified was C = 0.01 with L2 regularization, using the lib linear solver for efficiency.

9.1.2 Results:

The model demonstrated strong overall performance, achieving an accuracy of 78.86% on the test set. Below are the detailed performance metrics:

- **Precision:**
    - **Class 0 (Rejected Loans): 0.95**

      The model performed well in identifying rejected loans, achieving very high precision, which means it correctly labeled most rejected cases with minimal false positives.
    - **Class 1 (Approved Loans): 0.76**

      For approved loans, the precision value reflects a reasonable balance between correct and incorrect classifications.
- **Recall:**
    - **Class 0: 0.42**

      The model struggled to identify all rejected loans, as indicated by the relatively low recall for class 0. Only 42% of actual rejected loans were captured.
    - **Class 1: 0.99**

      In contrast, the model excelled in capturing approved loans, achieving an exceptionally high recall of 99%. This means nearly all approved loans were correctly identified.
- **F1-Score:**
    - **Class 0: 0.58**

The F1-Score for rejected loans reflects a trade-off between precision and recall, showing moderate performance for this class.

- **Class 1: 0.86**

  For approved loans, the model achieved a high F1-Score, confirming its reliability in making accurate predictions.

These results show that while the model strongly identifies approved loans (class 1), its performance on rejected loans (class 0) was less consistent.

Metric	**Value (Class 0)**	**Value (Class 1)**
Precision	0.95	0.76
Recall	0.42	0.99
F1- Score	0.58	0.86

9.1.3 Feature Importance:

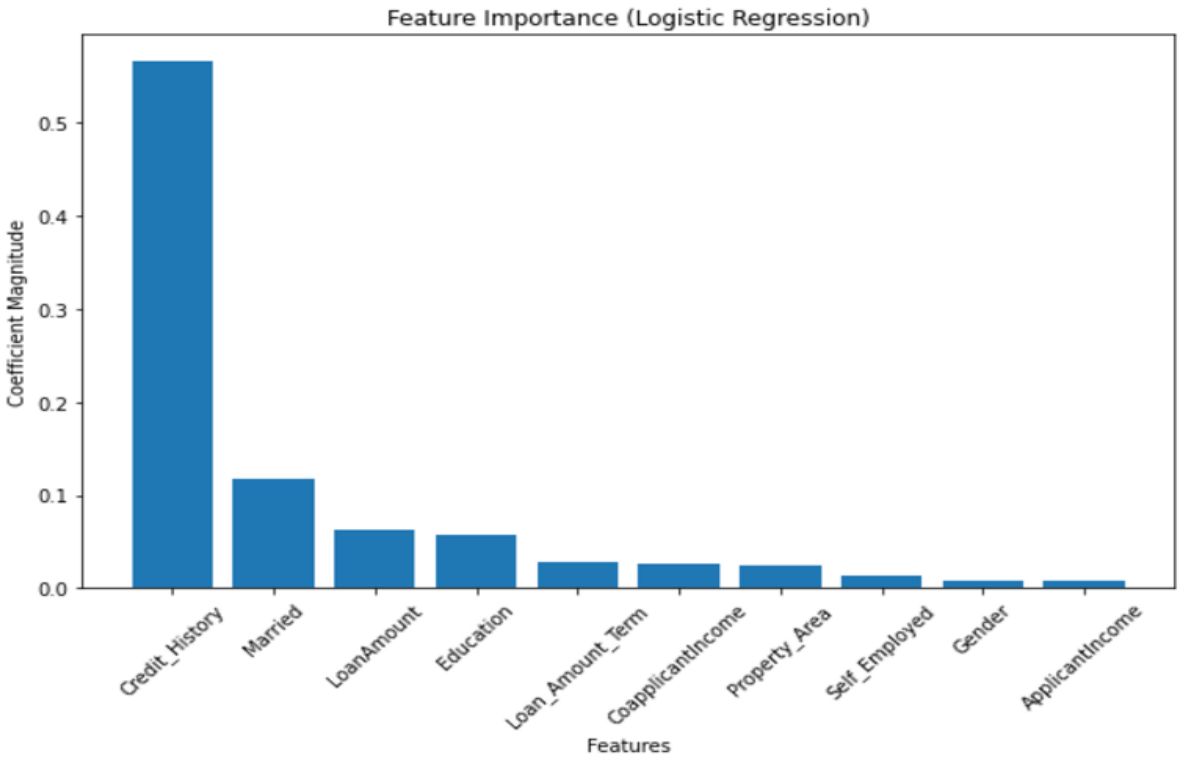

Figure 8

One of the major strengths of Logistic Regression is its ability to provide interpretable feature importance through its coefficients. The feature importance analysis revealed that:

1. Credit History was the most influential feature, with a positive coefficient, meaning applicants with good credit history were significantly more likely to have their loans approved.

2. Applicant Income also emerged as a strong predictor, indicating that higher income levels positively influenced loan approval probabilities.

The remaining features contributed less to the model's predictions, emphasizing the critical role of Credit History and Income in loan approval decisions.

9.2 Random Forest

Random Forest, an ensemble machine learning algorithm, was selected for its ability to effectively manage non-linear relationships and reduce overfitting by combining the predictions of multiple decision trees. Unlike single decision tree models, Random Forest operates by aggregating results from numerous decision trees, which increases robustness and generalization. Another significant

advantage of Random Forest is its ability to rank feature importance, allowing us to identify which variables contribute the most to predicting loan approval status.

9.2.1 Training Process

1. Initial Training with Default Hyper-parameters:

   The model was first trained using default parameters to establish a baseline performance. However, the results showed room for improvement, particularly in terms of generalization.

2. Feature Importance Analysis:

   Random Forest's ability to calculate feature importance was leveraged to identify which predictors had the largest influence on loan approval. The most important features were Credit History, Loan Amount, and Applicant Income.

9.2.1 Results

The Random Forest model demonstrated satisfactory performance without hyperparameter tuning, achieving an accuracy of 76.42%. Below are the detailed performance metrics for Class 0 (Rejected Loans) and Class 1 (Approved Loans).

1. **Precision**:
   - **Class 0**: **0.82**

     The model performed moderately well in identifying rejected loans, with 82% of its predictions being correct for this class.

   - **Class 1**: **0.75**

     For approved loans, the precision reflects a reasonable balance between correctly predicted approvals and false positives.

2. **Recall**:
   - **Class 0**: **0.42**

     The model struggled to capture all rejected loans, as indicated by the relatively low recall of 42%.

   - **Class 1: 0.95**

     The model excelled in identifying most approved loans, ensuring minimal false negatives for this class.

3. **F1-Score**:

- **Class 0**: **0.55**

  The F1-Score reflects a trade-off between precision and recall, showing moderate performance for rejected loans.

- **Class 1**: **0.84**

  For approved loans, the F1-Score demonstrates strong reliability, balancing precision and recall effectively.

Metric	**Value (Class 0)**	**Value (Class 1)**
Precision	0.82	0.75
Recall	0.42	0.95
F1-Score	0.55	0.84

9.2.3 Feature Importance Analysis

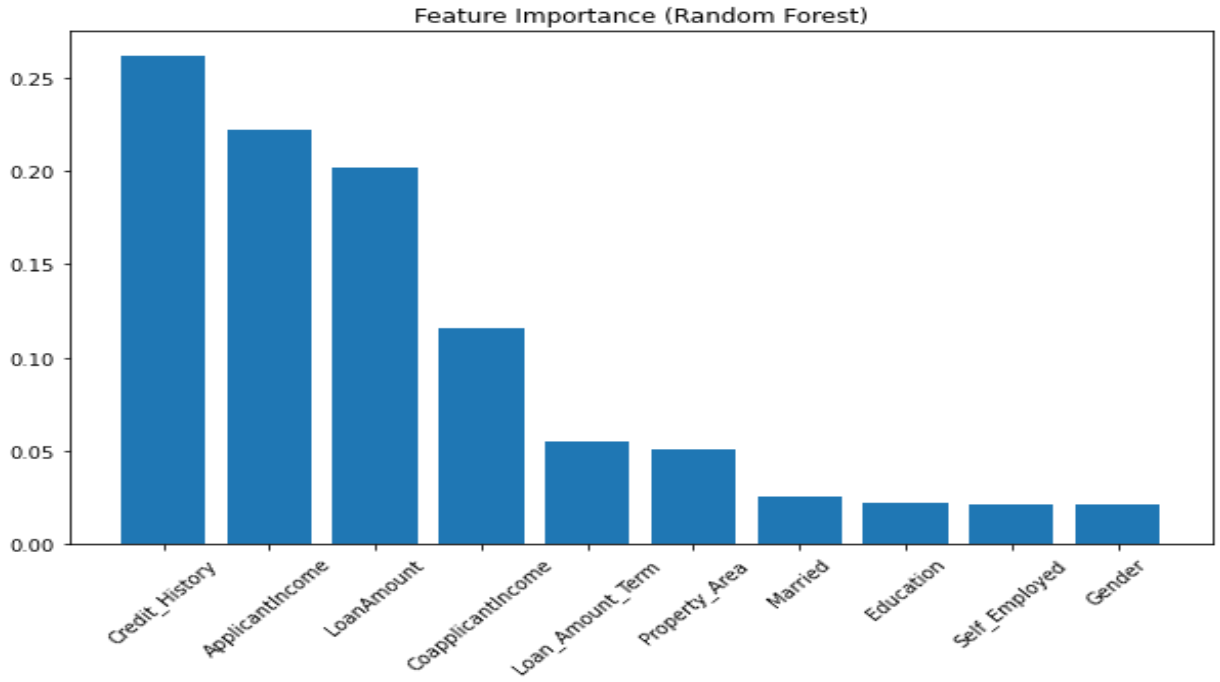

Figure 9

Feature importance analysis revealed the relative contribution of each variable to the model's predictions. The top three most influential features were:

1. Credit History:
    - Credit History emerged as the most critical predictor in the Random Forest model. This aligns with prior analysis, where Credit History exhibited a strong positive correlation (0.54) with Loan Status. Applicants with a positive credit history were far more likely to have their loans approved.
2. Loan Amount:
    - Loan Amount was the second most influential feature. It showed that higher loan amounts, while critical, often posed a higher risk, leading to stricter approval criteria. This insight helps explain why larger loan amounts frequently appear in rejected loan applications.
3. Applicant Income:
    - Applicant Income was also a significant predictor. Higher incomes generally improved the likelihood of loan approval, as they suggested better financial stability. However, the model recognized that income alone does not guarantee approval, as factors like credit history weighed more heavily.

The feature importance plot visually reinforced that Credit History dominates the decision-making process, followed by Loan Amount and Applicant Income, while other features contributed marginally.

9.3 Support Vector Machine (SVM)

Support Vector Machines (SVM) were included in the model evaluation process to test their ability to separate the two classes approved loans and not approved loans by finding the optimal hyperplane. SVMs are known for their flexibility to handle both linear and non-linear relationships using different kernel functions. The linear kernel was used for its simplicity and interpretability, while the Radial Basis Function (RBF) and polynomial kernels were applied to capture more complex patterns.

9.3.1 Training Process:

The SVM model underwent a systematic training and optimization process to maximize its predictive power:

1. Hyperparameter Tuning with Grid Search:

```
# SVM Hyperparameter Tuning
svm_model = SVC(probability=True, random_state=42)
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']  # Only for 'rbf' and 'poly' kernels
}
grid_search_svm = GridSearchCV(svm_model, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search_svm.fit(X_train, y_train)
```

   - Grid Search was used to identify the optimal combination of the penalty parameter (C) and the kernel coefficient (gamma) for non-linear kernels like RBF and polynomial.
     - The C parameter controls the trade-off between achieving a smooth decision boundary and correctly classifying the training data.
     - The gamma parameter determines how far the influence of a single data point reaches, making it critical for non-linear kernels.

2. Linear Kernel and Feature Importance:
   - For the linear kernel, the model provided feature importance through coefficients. The coefficients indicate how strongly each feature influences the decision boundary.
   - Key features like Credit History and Loan Amount emerged as the most impactful predictors, confirming their significance in determining loan approval.

3. Non-Linear Kernels (RBF and Polynomial):
   - For RBF and polynomial kernels, feature importance was analyzed using permutation-based importance, which evaluates the contribution of each feature by observing changes in the model's accuracy when a feature's values are randomly shuffled.

9.3.2 Results:

SVM achieved a balanced accuracy of 78.86% with strong performance across classes. Below are the metrics for Class 0 (Rejected Loans) and Class 1 (Approved Loans):

1. **Precision:**
   - **Class 0: 0.91**

The model effectively minimized false positives for rejected loans.

- **Class 1: 0.74**

  Precision for approved loans reflects reasonable performance in distinguishing approvals.

2. **Recall:**

   - **Class 0: 0.62**

     The model had moderate success in identifying rejected loans, with some missed cases.

   - **Class 1: 0.82**

     Recall for approved loans indicates that the model correctly identified most approvals.

3. **F1-Score:**

   - **Class 0: 0.74**

     The F1-Score highlights a good balance between precision and recall for rejected loans.

   - **Class 1: 0.78**

     For approved loans, the F1-Score confirms the model's balanced performance.

Metric	**Value (Class 0)**	**Value (Class 1)**
Precision	0.91	0.74
Recall	0.62	0.82
F1- Score	0.74	0.78

9.3.3 Feature Importance:

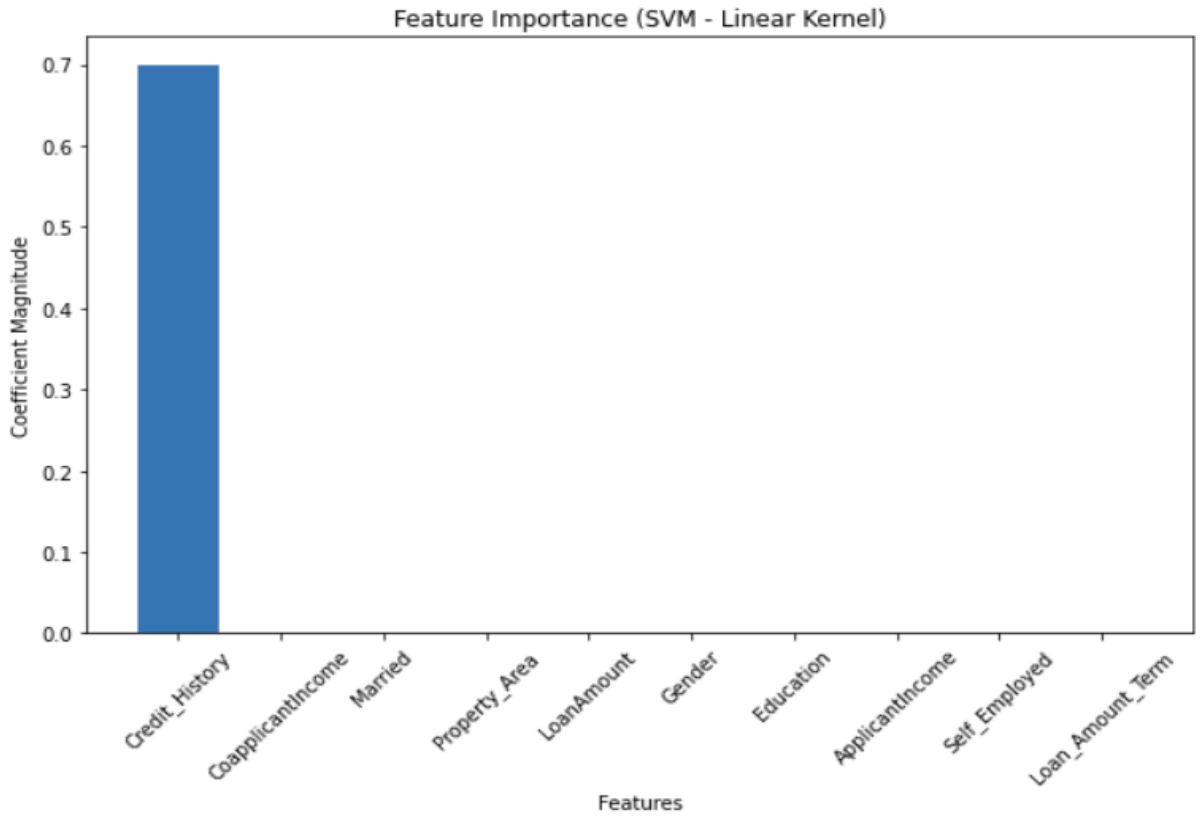

Figure 10

- For the linear kernel, Credit History and Loan Amount were identified as the most significant features influencing loan approval decisions.
- In contrast, the RBF kernel was able to better capture non-linear relationships in the data, leading to improved recall performance. However, the trade-off was a higher computational cost.

The feature importance results reaffirm that Credit History continues to be the strongest predictor, followed by Loan Amount, aligning with findings from other models like Random Forest.

10. Summary of Model Performance

The performance of all machine learning models was evaluated based on four key metrics: accuracy, precision, recall, and F1-score for both the positive class (Class 1: Approved Loans) and the negative class (Class 0: Rejected Loans). Each model demonstrated its strengths and limitations, which are discussed in detail below:

10.1 Logistic Regression:

- Accuracy: Achieved the highest accuracy of 78.86%, making it the most reliable and interpretable model in the study.
- **Class 1 (Approved Loans):**
    - Precision: 0.76, meaning 76% of the loans predicted as approved were correct.
    - Recall: 0.99, showcasing its exceptional ability to identify nearly all approved loans, with minimal false negatives.
    - F1-Score: 0.86, reflecting a strong balance between precision and recall, making it highly effective for predicting approved loans.
- **Class 0 (Rejected Loans):**
    - Precision: 0.95, ensuring minimal false positives for rejected loans.
    - Recall: 0.42, indicating that the model struggled to capture rejected loans accurately.
    - F1-Score: 0.58, demonstrating moderate performance for rejected loans due to low recall.

10.2 Random Forest:

- Accuracy: The model achieved an accuracy of approximately 76.42%, indicating its effectiveness in correctly predicting loan approvals and rejections.
- **Class 1 (Approved Loans):**
    - Precision: 0.75, signifying that 75% of the loans predicted as approved were indeed approved.
    - Recall: 0.95, demonstrating the model's proficiency in identifying 95% of all actual approved loans.
    - F1-Score: 0.84, reflecting a strong balance between precision and recall for approved loan predictions.
- **Class 0 (Rejected Loans):**
    - Precision: 0.82, indicating that 82% of the loans predicted as rejected were correctly identified.
    - Recall: 0.42, suggesting that the model correctly identified 42% of all actual rejected loans.

- F1-Score: 0.55, showing moderate performance in predicting rejected loans, primarily due to the lower recall.

10.3 Support Vector Machine (SVM):

- Accuracy: Matched Logistic Regression with an accuracy of 78.86%, showing its effectiveness in classification tasks.
- **Class 1 (Approved Loans):**
  - Precision: 0.74, indicating good precision in predicting approved loans.
  - Recall: 0.82, successfully identifying a large proportion of approved loans but lower than Logistic Regression.
  - F1-Score: 0.78, reflecting a balanced performance for approved loans.
- **Class 0 (Rejected Loans):**
  - Precision: 0.89, ensuring minimal false positives for rejected loans.
  - Recall: 0.38, indicating that the model struggled to identify rejected loans accurately.
  - F1-Score: 0.53, showing moderate performance due to low recall for rejected loans.

10.4 Final Comparison and Observations:

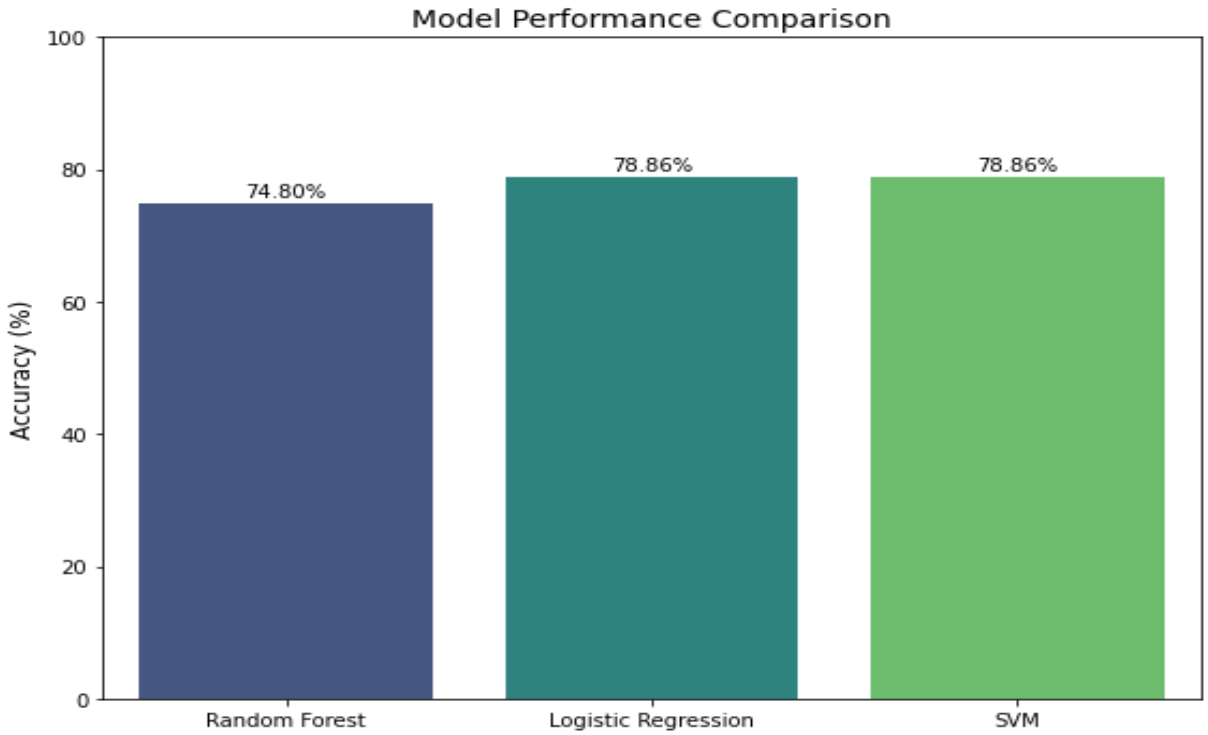

Figure 11

Key Observations:

1. **Logistic Regression:**

   Accuracy: Logistic Regression achieved an accuracy of 78.86%, which was the highest among the evaluated models. This highlights its effectiveness in handling binary classification tasks like loan approval prediction.

2. **Random Forest:**

   Accuracy: Random Forest achieved an accuracy of 76.42%, slightly lower than Logistic Regression and SVM.

3. **Support Vector Machine (SVM):**

   Accuracy: SVM matched Logistic Regression with an accuracy of 78.86%, demonstrating its capability in distinguishing between approved and rejected loans.

11. Learning Curves

Learning curves are essential tools for evaluating machine learning model performance, providing insights into how models improve with increasing training data. By plotting the training and

validation scores, we can diagnose overfitting (memorizing training data but failing to generalize) or underfitting (failing to capture meaningful patterns). In this section, I will discuss the learning curves of three models: Logistic Regression, Random Forest and Support Vector Machines (SVM).

11.1 Training and Validation Curves

11.1.1 Logistic Regression

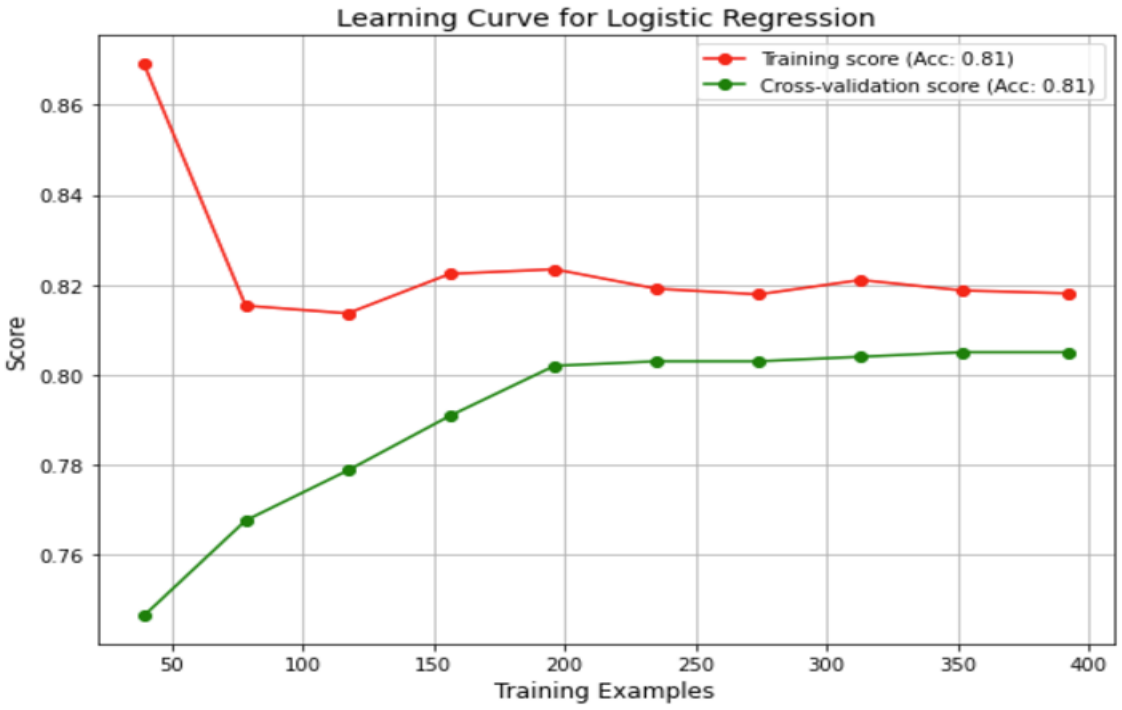

Figure 12

- Training Accuracy: 81%
- Validation Accuracy: 81%

Observations:

- Initially, the training accuracy started high but dropped slightly as more training examples were added. However, it stabilized around 81%, demonstrating that the model could learn effectively as the dataset grew.
- The validation accuracy followed closely, also stabilizing at 81%. This indicates that the model generalized well, as there was no significant gap between the training and validation curves.

11.1.2 Random Forest

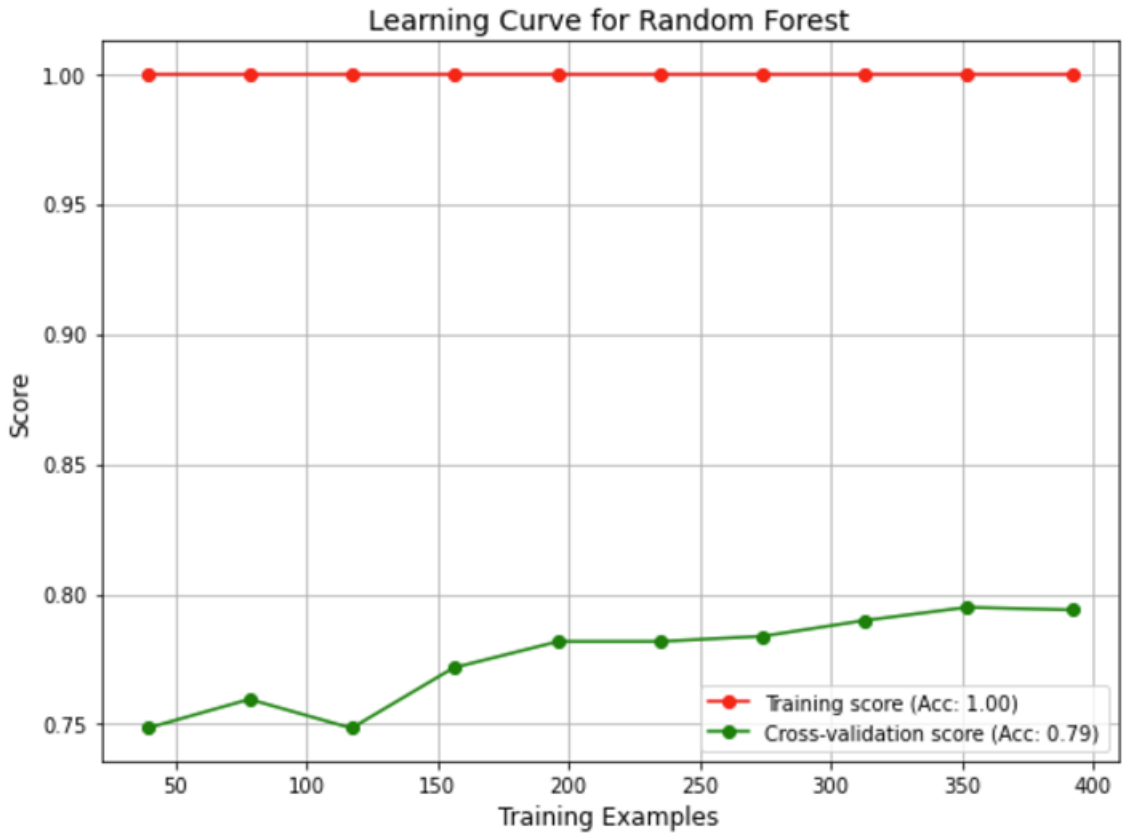

Figure 13

- Training Accuracy: 100%
- Validation Accuracy: 79-80%

Observations:

- The training accuracy remained 100% throughout, which clearly highlights Random Forest's tendency to memorize training data. Even with small datasets, the model achieved perfect training accuracy.
- The validation accuracy, however, stabilized at around 79-80%. A significant gap exists between the training and validation curves, indicating overfitting. The model is learning the training data very closely but struggles to perform equally well on unseen data.
- As more data was added, the validation score improved slightly, but the gap persisted, reinforcing the need for better regularization.

11.1.3 Support Vector Machines (SVM)

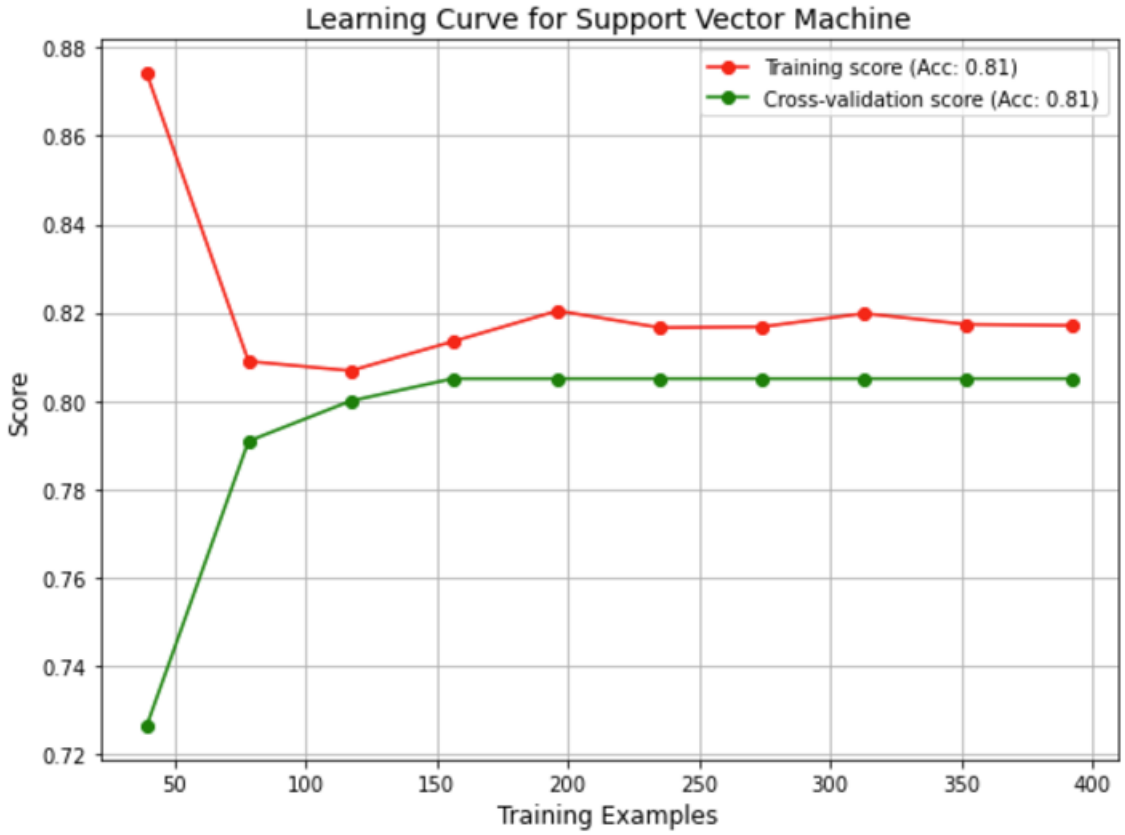

Figure 14

- Training Accuracy: 81%
- Validation Accuracy: 81%

Observations:

- At the start, SVM's training curve was significantly higher than the validation curve, indicating underfitting on smaller datasets. The model struggled to capture relationships due to the limited training examples.
- As more data was added, the validation curve improved steadily and ultimately stabilized at 81%, matching the training accuracy. This demonstrates that SVM requires larger datasets to generalize effectively.
- While the linear kernel was used initially, switching to non-linear kernels like RBF would help SVM handle more complex patterns, potentially improving performance further.

11.2 Overfitting and Underfitting Analysis

The learning curves allow us to analyze each model for overfitting and underfitting, as summarized below:

Model	**Training Accuracy (%)**	**Validation Accuracy (%)**	**Behavior**
Logistic Regression	81	81	Balanced (Good Generalization)
Random Forest	100	79-80	Overfitting
SVM	81	81	Initial Underfitting → Improved

11.3 Key Insights from Learning Curves:

1. Logistic Regression:
    - Achieved consistent performance with 81% accuracy for both training and validation data.
    - No significant signs of overfitting or underfitting, making it a robust baseline model.
2. Random Forest:
    - Overfitted the training data, reaching 100% training accuracy but stabilizing at 79-80% validation accuracy.
    - Regularization techniques are required to improve generalization.
3. Support Vector Machines:
    - Started with underfitting but improved as more data was added, stabilizing at 81% accuracy.
    - Performs well but requires significant computational resources.

12. Limitations and Recommendations

While this study successfully demonstrated the effectiveness of machine learning in predicting loan approvals, it also faced certain challenges that highlight opportunities for improvement. Below, I discuss the key limitations encountered and provide recommendations for future work to address these issues.

12.1 Dataset Limitations

1. **Small Dataset Size**
    - The dataset used in this study contained only 614 records, which limited the diversity and variety of applicant profiles available for analysis. A small dataset restricts the model's ability to learn robust patterns and generalize well to unseen data.
    - Impact: Models might perform well on the training data but fail when applied to larger, real-world datasets.
2. **Class Imbalance**
    - The dataset was heavily imbalanced, with significantly more approved loans than rejected ones. While techniques like SMOTE (Synthetic Minority Oversampling Technique) were used to mitigate this issue, the imbalance still made it difficult for the models to predict rejected loan cases accurately.
    - Impact: Precision and recall for the minority class (rejected loans) were lower, reducing overall performance.
3. **Missing Features**
    - Critical features that could provide deeper insights such as credit scores, employment history, debt-to-income ratio, and loan repayment behavior were absent from the dataset.
    - Impact: The lack of these features limited the models' predictive capabilities, as they could not fully capture an applicant's financial situation or creditworthiness.
4. **Lack of Diversity**
    - The dataset was obtained from a single source, which might not reflect the diversity of loan applicants across various regions, income levels, and financial institutions.
    - Impact: This reduces the general applicability of the findings, as the models may not perform as effectively when applied to different populations or environments.

12.2 Recommendations for Future Work

1. **Gather Larger and More Diverse Data**
    - Future research should aim to collect larger datasets that include a wider range of applicant profiles from diverse regions and financial institutions.
    - Recommendation: Collaborate with banks or lending institutions to access real-world datasets that represent different economic backgrounds and loan categories. A larger dataset will help models learn better and improve generalization.
2. **Include Additional Features**
    - Incorporating additional features like credit scores, employment history, debt-to-income ratio, loan repayment timelines, and applicant savings would significantly enhance predictive power.
    - Recommendation: Feature-rich datasets can help uncover more detailed patterns and enable models to make more reliable decisions.
3. **Improve Handling of Class Imbalances**
    - While SMOTE was effective, future studies can explore advanced techniques like cost-sensitive learning or ensemble methods that give more weight to minority class predictions.
    - Recommendation: These methods will help improve precision and recall for rejected loans, balancing performance across classes.
4. **Explore Advanced Machine Learning Approaches**
    - Investigating deep learning models or hybrid approaches that combine multiple algorithms may help capture more complex relationships in the data.
    - Recommendation: Techniques like stacked models or neural networks could be explored to improve predictive performance further, especially for larger datasets.
5. **Focus on Model Interpretability**
    - Improving transparency in model decisions is essential for gaining trust in loan approval systems. Tools like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) can provide insights into how features influence predictions.
    - Recommendation: Striking a balance between accuracy and explainability will ensure the models are both powerful and trustworthy for real-world use.

6. **Test Models in Real-World Scenarios**

    - Collaborating with financial institutions to test the models on real-world loan applications will validate their effectiveness and identify areas for refinement.

    - Recommendation: Pilot testing the models in production environments will provide valuable feedback, ensuring they work as expected and can adapt to evolving financial landscapes.

13. Conclusion

This study highlights how machine learning can revolutionize loan approvals, making them faster, more accurate, and scalable. Logistic Regression proved to be the most practical model, with 78.86% accuracy and a recall of 0.99 for approved loans, offering simplicity and transparency. Random Forest excelled in capturing complex patterns but faced overfitting, while SVM showed strong performance but required significant computational resources. Credit History emerged as the most influential factor in loan approvals, with Applicant Income and Loan Amount playing supporting roles but lacking standalone predictive power.

Challenges included a small, imbalanced dataset of 614 records, limiting model generalization and accurate rejection predictions. Missing key features, such as repayment history and debt-to-income ratios, further constrained the analysis. Ethical considerations, including fairness, transparency, and data privacy, were prioritized, with techniques like SMOTE addressing class imbalances and GDPR compliance maintained.

Looking ahead, financial institutions can benefit from diverse datasets, advanced modeling techniques, and real-world testing to improve model accuracy and scalability. Machine learning offers a promising path to modernize loan processes while ensuring fairness and transparency, paving the way for smarter and more inclusive financial decision-making.

14. References:

- Cheng, D., Niu, Z., Liu, X., and Zhang, L. (2021). 'Risk Assessment for Contagion Path in Complex Loan Network'. *Science China Information Sciences*. Available at: https://ijesr.org/admin/uploads/Prediction%20of%20Modernized%20Loan%20Approval%20System%20Based%20on%20Machine%20Learning%20Approach.pdf

- Goliya, B. (2024) 'Loan Approval Prediction Using Machine Learning', *International Journal of Research Publication and Reviews*, 5(1), pp. 95–98. Available at: https://ijrpr.com/uploads/V5ISSUE1/IJRPR21516.pdf

- Gudipalli, A., Sujatha, C.N., Pushyami, B.H., Narra, N.K. and Sanjana, B.N. (2021) 'Loan Prediction Using Machine Learning and Its Deployment on Web Application', *Innovations in Power and Advanced Computing Technologies (i-PACT)*. IEEE. Available at: https://ieeexplore.ieee.org/document/9696448

- Kadam, E., Gupta, A., Jagtap, S., Dubey, I. and Tawde, G. (2023) 'Loan Approval Prediction System Using Logistic Regression and CIBIL Score', *2023 Fourth International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE. Available at: https://ieeexplore.ieee.org/document/10182799

- Mamun, M.A., Farjana, A. and Mamun, M. (2022) 'Predicting Bank Loan Eligibility Using Machine Learning Models and Comparison Analysis', *Proceedings of the 7th North American International Conference on Industrial Engineering and Operations Management*. Available at: https://ieomsociety.org/proceedings/2022orlando/328.pdf

- Orji, U.E., Ugwuishiwu, C.H., Nguemaleu, J.C.N. and Ugwuanyi, P.N. (2022) 'Machine Learning Models for Predicting Bank Loan Eligibility', *IEEE NIGERCON*. IEEE. Available at: https://ieeexplore.ieee.org/document/9803172

- Rahman, A.T., Purno, M.R.H. and Mim, S.A. (2023) 'Prediction of the Approval of Bank Loans Using Various Machine Learning Algorithms', *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*. IEEE. Available at: https://ieeexplore.ieee.org/document/10730968

- Sarisa, H.K., Khurana, V., Koti, V.C. and Garg, N. (2023) 'Loan Prediction Using Machine Learning', *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*. IEEE. Available at: https://ieeexplore.ieee.org/document/10466049

- Shaik, B. and Siva Prasad, Y.A. (2024) 'Loan Eligibility Prediction Using Machine Learning', *International Journal of Novel Research and Development*, 9(7). Available at: https://www.ijnrd.org/papers/IJNRD2407179.pdf

- Sharmila, S., Sandhya, P.V.S., Kousar, P.S., Anuradha, P. and Deekshitha, S. (2024) 'Bank Loan Approval Using Machine Learning', *2024 International Conference on Integrated Circuits and Communication Systems (ICICACS)*. IEEE. Available at: https://ieeexplore.ieee.org/document/10466103

- Turiel, J.D. and Aste, T. (2019) 'P2P Loan Acceptance and Default Prediction with Artificial Intelligence', *arXiv preprint arXiv:1907.01800*. Available at: https://arxiv.org/abs/1907.01800

15. Appendices:

15.1 Data Preparation

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report
from imblearn.over_sampling import SMOTE
from sklearn.inspection import permutation_importance
from collections import Counter
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit, learning_curve

# Load dataset
file_path =
r"C:\Users\Bharani\Downloads\Dissertation_Project\Dataset\loan_sanction_train.csv"
data = pd.read_csv(file_path)

# Display basic information
data.info()

# Display the first few rows
data.head()

# Check for missing values
missing_values = data.isnull().sum()
print("Missing values:\n", missing_values)

# Fill missing values
def fill_missing_with_mode(df, column_name):
    mode_value = df[column_name].mode()[0]
    df[column_name].fillna(mode_value, inplace=True)

def fill_missing_with_mean(df, column_name):
    mean_value = df[column_name].mean()
    df[column_name].fillna(mean_value, inplace=True)
```

```
# Cleaning
def clean_data(df):
    mode_columns = ['Gender', 'Married', 'Dependents', 'Self_Employed',
'Loan_Amount_Term', 'Credit_History']
    for column in mode_columns:
        fill_missing_with_mode(df, column)
    mean_columns = ['LoanAmount']
    for column in mean_columns:
        fill_missing_with_mean(df, column)

clean_data(data)

# Convert categorical features to numerical
le = LabelEncoder()
for column in data.columns:
    data[column] = le.fit_transform(data[column])

# Verify cleaned data
data.head()
```

Visualise and Summarise Data

```
# Correlation Matrix

plt.figure(figsize=(12, 8))

correlation_matrix = data.corr()

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)

plt.title("Correlation Matrix: Relationships Between Features", fontsize=16)

plt.xlabel("Features", fontsize=12)

plt.ylabel("Features", fontsize=12)

plt.show()



# Bar Plot: Education vs. Applicant Income

education_income = data.groupby('Education')['ApplicantIncome'].mean()

plt.figure(figsize=(8, 6))

education_income.plot(kind='bar', color=['lightblue', 'skyblue'])
```

```
plt.title("Education vs. Average Applicant Income", fontsize=16)

plt.xlabel("Education (0 = Graduate, 1 = Not Graduate)", fontsize=12)

plt.ylabel("Average Applicant Income ($)", fontsize=12)

plt.xticks(rotation=0)

for index, value in enumerate(education_income):

    plt.text(index, value, f"${value:,.0f}", ha='center', va='bottom', fontsize=10)

plt.show()


# Bar Plot: Credit History vs. Loan Amount by Loan Status

credit_history_loan_amount = data.groupby(['Credit_History',
'Loan_Status'])['LoanAmount'].mean().unstack()

plt.figure(figsize=(12, 8))

bars = credit_history_loan_amount.plot(kind='bar', color=['red', 'green'], width=0.6,
edgecolor='black', figsize=(12, 8))

plt.title("Credit History vs. Average Loan Amount by Loan Status", fontsize=16)

plt.xlabel("Credit History (0 = Bad, 1 = Good)", fontsize=12)

plt.ylabel("Average Loan Amount (£ Thousands)", fontsize=12)

plt.xticks(ticks=[0, 1], labels=["Bad Credit History (0)", "Good Credit History (1)"],
fontsize=12, rotation=0)

for container in bars.containers:

    bars.bar_label(container, fmt="£%.2f", fontsize=12, padding=3)

plt.legend(

    ["Not Approved", "Approved"],

    title="Loan Status",

    loc='upper center',

    bbox_to_anchor=(0.5, -0.15),
```

```
    ncol=2,
    fontsize=12
)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()


# Bar Plot: Property Area vs. Applicant Income by Loan Status
plt.figure(figsize=(10, 6))
property_income = data.groupby(['Property_Area',
'Loan_Status'])['ApplicantIncome'].mean().unstack()
property_income.plot(kind='bar', color=['lightcoral', 'lightgreen'], figsize=(10, 6),
width=0.8)
plt.title("Property Area vs. Average Applicant Income by Loan Status", fontsize=16)
plt.xlabel("Property Area (0 = Rural, 1 = Semiurban, 2 = Urban)", fontsize=12)
plt.ylabel("Average Applicant Income (£)", fontsize=12)
plt.xticks(rotation=0)
for container in plt.gca().containers:
    plt.bar_label(container, fmt="£%.0f", fontsize=10)
plt.legend(
    ["Not Approved", "Approved"],
    title="Loan Status",
    loc='upper center',
    bbox_to_anchor=(0.5, -0.15),
    ncol=2,
    fontsize=10
```

```
)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()


# Box Plot: Applicant Income by Loan Status
colors = ['lightblue', 'darkred']
plt.figure(figsize=(10, 6))
sns.boxplot(x='Loan_Status', y='ApplicantIncome', data=data, palette=colors)
plt.title("Loan Status vs. Applicant Income Distribution", fontsize=16)
plt.xlabel("Loan Status (0 = Not Approved, 1 = Approved)", fontsize=12)
plt.ylabel("Applicant Income (£)", fontsize=12)
medians = data.groupby('Loan_Status')['ApplicantIncome'].median()
for i, median in enumerate(medians):
    plt.text(i, median, f"£{median:,.0f}", ha='center', va='center', fontsize=10,
color='black', bbox=dict(facecolor='white', edgecolor='none'))
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```

15.2 Model Implementation

Feature Selection

```
# Feature selection
features = ['Gender', 'Married', 'Education', 'Self_Employed', 'ApplicantIncome',
        'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History',
'Property_Area']
X = data[features]
y = data['Loan_Status']
```

 # Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Standardize features

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

SMOTE

 # Class distribution before SMOTE

before_smote = Counter(y_train)

 # SMOTE for Class Balancing

smote = SMOTE(random_state=42)

 # Class distribution after SMOTE

X_res, y_res = smote.fit_resample(X_train, y_train)

after_smote = Counter(y_res)

 # Plot class distributions

fig, axes = plt.subplots(1, 2, figsize=(12, 5), sharey=True)

 # Before SMOTE

sns.barplot(x=list(before_smote.keys()), y=list(before_smote.values()), ax=axes[0], palette="Blues")

axes[0].set_title("Class Distribution Before SMOTE")

```
axes[0].set_xlabel("Classes")

axes[0].set_ylabel("Count")

axes[0].bar_label(axes[0].containers[0])


# After SMOTE

sns.barplot(x=list(after_smote.keys()), y=list(after_smote.values()), ax=axes[1],
palette="Greens")

axes[1].set_title("Class Distribution After SMOTE")

axes[1].set_xlabel("Classes")

axes[1].bar_label(axes[1].containers[0])

plt.tight_layout()

plt.show()
```

Logistic Regression

```
# Logistic Regression model

lr_model = LogisticRegression(random_state=42)

lr_model.fit(X_train, y_train)

y_pred_lr = lr_model.predict(X_test)


# Evaluation

print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))

print("Classification Report:\n", classification_report(y_test, y_pred_lr))

 # Logistic Regression Hyperparameter Tuning

log_reg = LogisticRegression(max_iter=1000, random_state=42)

param_grid = {

    'C': [0.01, 0.1, 1, 10, 100],  # Regularization strength

    'penalty': ['l1', 'l2'],  # Type of regularization
```

```
    'solver': ['liblinear', 'saga']  # Solvers compatible with l1/l2
}
grid_search_lr = GridSearchCV(log_reg, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search_lr.fit(X_train, y_train)
# Best Logistic Regression Model
best_log_reg = grid_search_lr.best_estimator_
y_pred_lr = best_log_reg.predict(X_test)
print("Logistic Regression Best Parameters:", grid_search_lr.best_params_)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Classification Report:\n", classification_report(y_test, y_pred_lr))

# Feature Importance (Logistic Regression)
coefficients = best_log_reg.coef_[0]
sorted_idx = np.argsort(np.abs(coefficients))[::-1]
plt.figure(figsize=(10, 6))
plt.bar(range(len(coefficients)), np.abs(coefficients[sorted_idx]), align='center')
plt.xticks(range(len(coefficients)), [features[i] for i in sorted_idx], rotation=45)
plt.title("Feature Importance (Logistic Regression)")
plt.xlabel("Features")
plt.ylabel("Coefficient Magnitude")
plt.show()
```

Random Forest Classifier

```
# Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

```
y_pred_rf = rf_model.predict(X_test)
# Evaluation
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
# Feature importance
importances = rf_model.feature_importances_
sorted_indices = np.argsort(importances)[::-1]
plt.figure(figsize=(10, 6))
plt.bar(range(len(importances)), importances[sorted_indices], align='center')
plt.xticks(range(len(importances)), [features[i] for i in sorted_indices], rotation=45)
plt.title("Feature Importance (Random Forest)")
plt.show()
```

Support Vector Machine (SVM)

```
# SVM model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
# Evaluation
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
from sklearn.inspection import permutation_importance
# SVM Hyperparameter Tuning
svm_model = SVC(probability=True, random_state=42)
param_grid = {
    'C': [0.1, 1, 10, 100],
```

```
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']  # Only for 'rbf' and 'poly' kernels
}
grid_search_svm = GridSearchCV(svm_model, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search_svm.fit(X_train, y_train)
# Best SVM Model
best_svm = grid_search_svm.best_estimator_
y_pred_svm = best_svm.predict(X_test)
print("SVM Best Parameters:", grid_search_svm.best_params_)
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
# Feature Importance (SVM - Linear Kernel)
if grid_search_svm.best_params_['kernel'] == 'linear':
    coefficients = best_svm.coef_[0]
    sorted_idx = np.argsort(np.abs(coefficients))[::-1]
    plt.figure(figsize=(10, 6))
    plt.bar(range(len(coefficients)), np.abs(coefficients[sorted_idx]), align='center')
    plt.xticks(range(len(coefficients)), [features[i] for i in sorted_idx], rotation=45)
    plt.title("Feature Importance (SVM - Linear Kernel)")
    plt.xlabel("Features")
    plt.ylabel("Coefficient Magnitude")
    plt.show()
else:
    # Permutation Importance for non-linear kernels
```

```
    perm_importance = permutation_importance(best_svm, X_test, y_test, n_repeats=10,
random_state=42)

    sorted_idx = perm_importance.importances_mean.argsort()

    plt.figure(figsize=(10, 6))

    plt.barh(range(len(sorted_idx)), perm_importance.importances_mean[sorted_idx],
align="center")

    plt.yticks(range(len(sorted_idx)), [features[i] for i in sorted_idx])

    plt.xlabel("Permutation Importance")

    plt.title("Feature Importance (SVM - Non-linear Kernel)")

    plt.show()
```

15.3 Summary of Results

Comparing Model Performance

```
# Model accuracy summary

models = ['Random Forest', 'Logistic Regression', 'SVM']

accuracies = [

    accuracy_score(y_test, y_pred_rf) * 100,

    accuracy_score(y_test, y_pred_lr) * 100,

    accuracy_score(y_test, y_pred_svm) * 100

]

# Bar plot for comparison

plt.figure(figsize=(8, 6))

ax = sns.barplot(x=models, y=accuracies, palette='viridis')

for index, value in enumerate(accuracies):

    plt.text(index, value + 1, f"{value:.2f}%", ha='center', fontsize=10)

plt.title("Model Performance Comparison", fontsize=14)

plt.ylabel("Accuracy (%)", fontsize=12)
```

```
plt.ylim(0, 100)
plt.xticks(fontsize=10)
plt.tight_layout()
plt.show()
```

Training Score and Cross Validation Score

```
# Initialize models
models = {
    'Support Vector Machine': SVC(kernel='linear', random_state=42),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'Logistic Regression': LogisticRegression(random_state=42)
}

# Evaluate each model
for name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Calculate Training Score
    train_predictions = model.predict(X_train)
    training_score = accuracy_score(y_train, train_predictions)

    # Calculate Cross-Validation Score
    cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
    cross_validation_score = cv_scores.mean()
    print(f'{name}:')
```

```
    print(f'  Training Score: {training_score:.2f}')
    print(f'  Cross-Validation Score: {cross_validation_score:.2f}\n')
```

Learning Curve

```
# Define models
models = {
    'Support Vector Machine': SVC(kernel='linear', random_state=42),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'Logistic Regression': LogisticRegression(random_state=42)
}
# Split shuffle strategy
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=42)
# Plot individual learning curves for each model
for name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Calculate Training Score
    train_predictions = model.predict(X_train)
    training_score = accuracy_score(y_train, train_predictions)
    # Calculate Cross-Validation Score
    cv_scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
    cross_validation_score = cv_scores.mean()
    # Compute learning curve
    train_sizes, train_scores, test_scores = learning_curve(
        model, X_train, y_train, cv=cv, n_jobs=-1,
```

```
        train_sizes=np.linspace(0.1, 1.0, 10), scoring='accuracy'
    )
    train_scores_mean = np.mean(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)

    # Plot learning curve
    plt.figure(figsize=(8, 6))
    plt.plot(train_sizes, train_scores_mean, 'o-', color='r', label=f'Training score (Acc: {training_score:.2f})')
    plt.plot(train_sizes, test_scores_mean, 'o-', color='g', label=f'Cross-validation score (Acc: {cross_validation_score:.2f})')
    plt.title(f'Learning Curve for {name}', fontsize=14)
    plt.xlabel('Training Examples', fontsize=12)
    plt.ylabel('Score', fontsize=12)
    plt.legend(loc='best', fontsize=10)
    plt.grid()
    plt.tight_layout()
    plt.show()
```