**CA 1 Project Report**Submitted in partial fulfilment of the requirements for the award of degree of  
**Machine Learning Project**

**Credit Card Approval Prediction**





**SUBMITTED TO**

**Lovely Professional University**

**(School of Computer Science and Engineering)**

**12203561**

**Roll no: 21**

**Kandimalla Bruha dev**

**1. Introduction**

**1.1 Project Title & Problem Statement**

In today's fast-paced financial landscape, banks and other credit institutions receive a massive number of credit card applications every day. Reviewing these applications manually is not only time-consuming but also susceptible to human error and inconsistency. More importantly, incorrect approvals can lead to financial losses, while overly strict denials may result in lost business opportunities.

This project tackles the problem by proposing a data-driven approach. Using machine learning, we aim to predict whether a credit card application should be approved based on the applicant’s profile and historical data. By learning from past patterns, our model will assist institutions in making faster, fairer, and more accurate decisions.

**1.2 Objective**

The primary goal of this project is to **build a robust machine learning model** that can accurately predict the approval status of credit card applications. Through this, we aim to:

* **Automate** and **streamline** the credit card approval process.
* **Reduce the workload** of financial analysts and cut down manual errors.
* **Identify key features** that influence approval decisions.
* **Ensure better risk management** by filtering out applicants who might default based on historical trends.

In essence, we are working toward a smarter and more reliable approval system that benefits both lenders and applicants.

**1.3 Scope**

This project is primarily focused on the **prediction of credit card approval status** using machine learning techniques. Here's what we aim to cover:

* **Data Preparation**: Cleaning and preprocessing the dataset, including handling missing values, encoding categorical variables, and scaling features to make the data model-ready.
* **Model Building**: Implementing and comparing different classification algorithms such as Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines.
* **Evaluation Metrics**: Assessing the performance of each model using industry-standard metrics like accuracy, precision, recall, and F1-score.
* **Insights Extraction**: Interpreting the results to understand what factors most significantly affect approval decisions.

However, there are some boundaries we’ve set for this project:

* The model will not be deployed into a real-world banking system.
* We will not address real-time fraud detection or continuous monitoring.
* We will not attempt to calculate credit scores or engage in credit risk modelling beyond the scope of binary classification.

This project is a step toward making smarter financial decisions, backed by data and automation, with the hope of improving efficiency and fairness in the credit approval process.

**2. Literature Review**

In recent years, **machine learning** and **data-driven decision-making** have gained significant traction in the financial sector. The ability to predict outcomes such as loan defaults, creditworthiness, and application approvals using historical data has opened up new possibilities for improving operational efficiency and reducing risks.

**Previous Research and Trends**

Several studies have explored the potential of machine learning in credit scoring and approval systems. Traditionally, credit scoring has relied on rule-based systems and statistical models like **logistic regression**, but with the emergence of vast datasets and advanced computing power, more sophisticated models have come into play.

A study by *Hand and Henley (1997)* compared statistical and machine learning methods for credit scoring and found that techniques such as **decision trees** and **neural networks** can outperform traditional approaches when properly tuned and trained on large datasets. Similarly, *Baesens et al. (2003)* evaluated various classification techniques and emphasized the importance of feature engineering and model interpretability.

**Existing Models and Techniques**

Several machine learning models have been widely applied in the domain of credit risk prediction:

* **Logistic Regression**: Still popular due to its simplicity and interpretability. It models the probability of approval as a function of applicant features.
* **Decision Trees & Random Forests**: These models are known for handling non-linearity and interaction effects well. They also provide a clear structure, which is helpful for understanding decision paths.
* **Support Vector Machines (SVM)**: Effective in high-dimensional spaces, especially when the margin between approved and rejected applications is subtle.
* **Neural Networks**: Though powerful, they are often viewed as black boxes, which limits their acceptability in financial institutions that require explainability.

Each model has its strengths and weaknesses. In practice, ensemble methods like **Random Forests** and **Gradient Boosting Machines (e.g., XGBoost)** have shown excellent performance, striking a balance between accuracy and generalizability.

**Industry Applications**

Companies like **FICO** and **Experian** use advanced predictive models to assess creditworthiness. Fintech startups are also leveraging AI to provide alternative credit scoring solutions based on non-traditional data (e.g., social media, utility bills, spending patterns).

These trends highlight a shift from static rule-based evaluations to dynamic, adaptive systems that can learn and evolve with changing data trends.

**Gaps in Existing Work**

While many studies focus on model performance, fewer emphasize **ethical considerations**, such as **bias** in training data and **fairness** in predictions. Additionally, interpretability remains a key challenge—especially for complex models like neural networks.

Our project aims to bridge the gap by:

* Comparing multiple models.
* Ensuring clarity in the decision-making process.
* Highlighting features that play a crucial role in credit decisions.

**3. Dataset Description**

**3.1 Data Source**

The dataset used in this project is publicly available and was obtained from the **UCI Machine Learning Repository**, a well-known platform for machine learning benchmark datasets.

* **Dataset Name**: *Credit Approval Dataset*
* **Source**: [UCI Machine Learning Repository – Credit Approval Data](https://archive.ics.uci.edu/ml/datasets/credit+approval)

This dataset is frequently used in research and experiments related to financial decision-making and credit risk assessment.

**3.2 Data Characteristics**

The dataset contains anonymized information about individuals who applied for credit cards. While the feature names have been intentionally disguised (to protect privacy), they represent a combination of demographic and financial attributes relevant to credit assessment.

* **Number of Records (Instances)**: 690
* **Number of Features**: 15 input features and 1 target variable (approval status)

**Types of Features:**

* **Categorical Features**: Include variables like gender, marital status, and job type.
* **Numerical Features**: Include continuous variables like age, income, and loan amount.
* **Target Variable**:
  + '+' indicates credit card **approval**
  + '-' indicates **denial**

**3.3 Data Preprocessing**

To prepare the dataset for machine learning modeling, the following preprocessing steps were performed:

**1. Data Cleaning**

* All missing values, originally represented by '?', were replaced with NaN.
* Rows containing missing values were either imputed or removed, based on the proportion and impact of missing data in each column.

**2. Encoding Categorical Variables**

* Categorical variables were converted to numerical format to be usable by machine learning models.
* **Label Encoding** was used for binary categorical variables.
* **One-Hot Encoding** was applied to variables with multiple categories to avoid ordinality assumptions.

**3. Feature Scaling**

* Numerical features were standardized using **Standard Scaler**.
* This helped ensure that all features contributed equally, particularly for models sensitive to feature magnitude such as Support Vector Machines.

**4. Feature Selection**

* Features with little variance or high correlation with others were considered for removal.
* Feature importance scores from tree-based models (such as Random Forest) were used to identify the most influential features in predicting credit approval.

**4. Methodology**

This section outlines the step-by-step approach followed in building the credit card approval prediction model. The methodology encompasses everything from data preprocessing and visualization to model selection, training, and evaluation.

**4.1 Machine Learning Algorithms Used**

To ensure comprehensive evaluation and comparison, we implemented multiple machine learning algorithms, each chosen for its strengths in classification problems:

**1. Logistic Regression**

A widely-used algorithm in the finance sector due to its simplicity and interpretability. Logistic regression models the probability of approval as a function of input features and is particularly effective when the relationship between the features and the target is linear.

**2. Decision Tree Classifier**

This algorithm creates a flowchart-like tree structure to make decisions. It's intuitive, easy to visualize, and capable of handling both categorical and numerical data without requiring feature scaling.

**3. Random Forest Classifier**

An ensemble method that builds multiple decision trees and merges their predictions for improved accuracy and generalization. It helps reduce overfitting and is robust to noise and outliers.

**4. Support Vector Machine (SVM)**

SVM constructs a hyperplane that best separates the classes in high-dimensional space. It’s particularly useful when there is a clear margin of separation and performs well with properly scaled features.

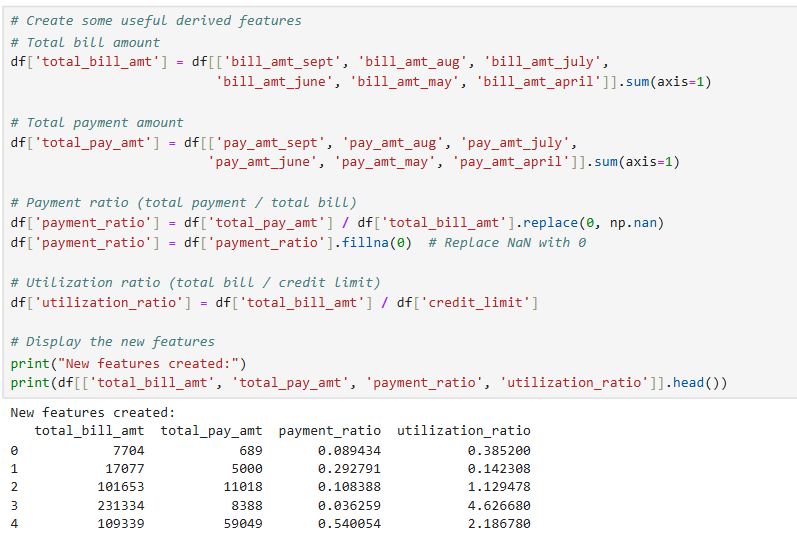
Each algorithm was chosen to offer a diverse perspective on model performance, interpretability, and scalability.

**4.2 Model Training**

**Data Cleaning and Preparation**

Before training the models, the dataset underwent several preprocessing steps to ensure the quality and consistency of input data:

* **Missing Value Handling**: All instances of '?' were replaced with NaN. Rows with excessive missing values were dropped, while others were imputed using the most frequent or median values depending on the data type.
* **Class Balance**: We checked for class imbalance and applied stratified sampling during train-test split to ensure both classes were adequately represented.
* **Feature Scaling**: StandardScaler was applied to normalize numerical features, especially for algorithms like SVM that are sensitive to feature magnitude.



A screenshot of a computer program

AI-generated content may be incorrect.**Categorical Encoding**: Binary categorical variables were transformed using label encoding, while multi-class variables were converted via one-hot encoding to prevent unintended ordinal relationships.

**Training Process**

* The dataset was split into **training and testing sets** (commonly 80-20 or 70-30).
* For each model, we used **cross-validation** to evaluate performance during training and to prevent overfitting.
* **Grid Search with Cross-Validation** was employed for hyperparameter tuning, ensuring that we selected the best model configuration for each algorithm. For instance:
  + Logistic Regression: Tuned regularization parameter C
  + Decision Tree: Tuned depth, minimum samples per leaf
  + Random Forest: Number of estimators, max depth, feature selection
  + SVM: Kernel type, C, and gamma parameters

You can refer to the accompanying graphs for cross-validation scores, confusion matrices, and ROC curves.

**4.3 Evaluation Metrics**

To assess the model performance effectively, we used a set of standard classification evaluation metrics:

**Accuracy**

Measures the overall correctness of the model. While useful, it can be misleading in imbalanced datasets.

**Precision**

Indicates how many of the predicted positives were actually correct. It’s essential when false positives are costly (e.g., approving unqualified applicants).

**Recall (Sensitivity)**

Tells us how many of the actual positive cases were correctly identified. Important in scenarios where catching all possible approvals is critical.

**F1-Score**

The harmonic mean of precision and recall. It provides a balanced measure, especially useful when there’s an uneven class distribution.

**Confusion Matrix**

A visual breakdown of predicted vs actual values. It helps identify the types of errors being made.

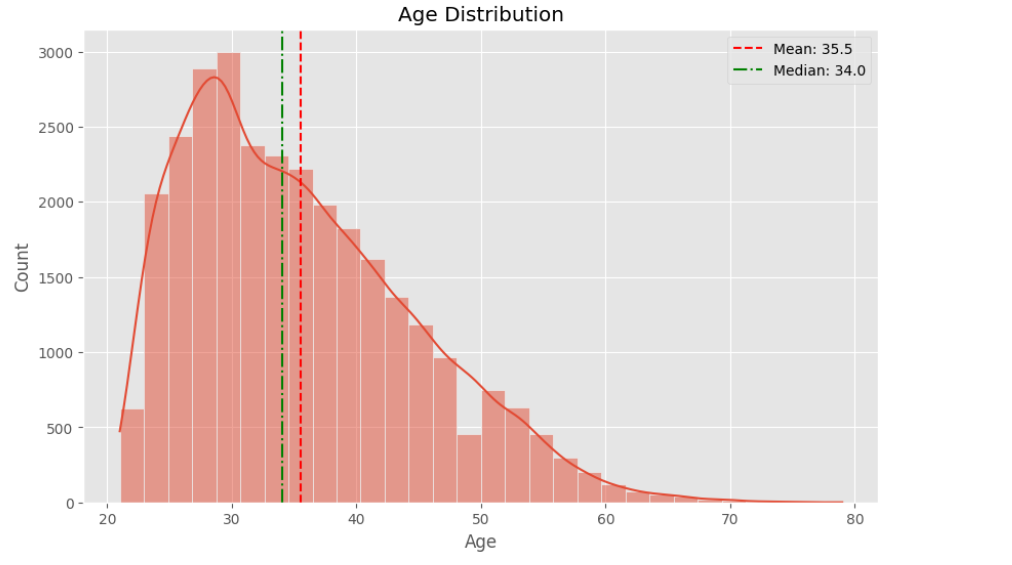
**ROC Curve and AUC Score**

These helps evaluate the trade-off between sensitivity and specificity across different thresholds. A higher AUC indicates better overall performance.

**Visualizations and Insights**

Throughout the project, various plots and visualizations were created to support understanding and insights:

* **Distribution Plots**: Showed skewness in numerical features like income or age.



A graph of a number of people

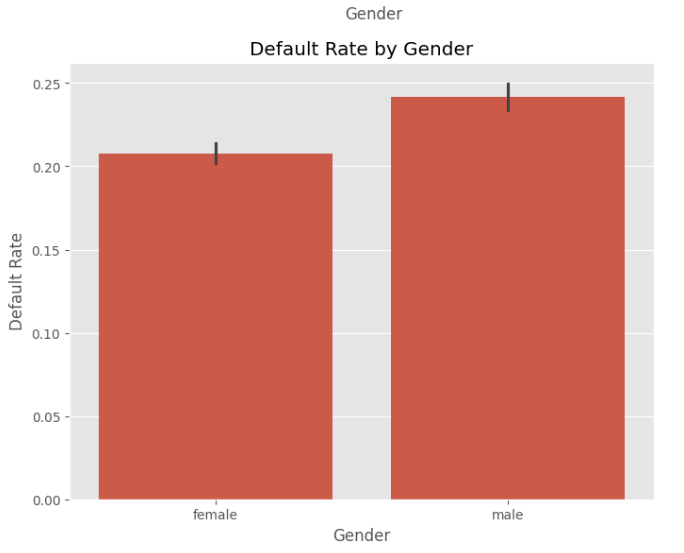
AI-generated content may be incorrect.

* **Correlation Heatmaps**: Highlighted relationships between features.

A chart with red and blue squares

AI-generated content may be incorrect.

* **Boxplots**: Helped detect outliers and understand variance.



* A blue and white squares with numbers

  AI-generated content may be incorrect.**Confusion Matrix Visuals**: Provided clarity on misclassification patterns.

A blue squares with numbers

AI-generated content may be incorrect.

A blue squares with white text

AI-generated content may be incorrect.

A **confusion matrix** is a performance measurement tool for classification models. It helps you understand **how well your model is performing**, beyond just an overall accuracy score.

Let’s break it down:

**Basic Structure of a Confusion Matrix**

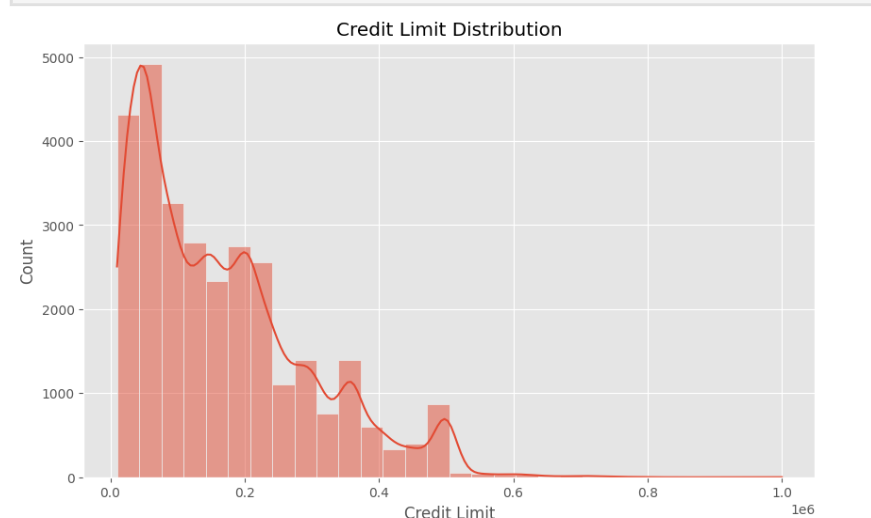
For binary classification (like credit card approval: *Approved* vs *Not Approved*), the confusion matrix looks like this:

|  |  |  |
| --- | --- | --- |
|  | **Predicted: Approved (+)** | **Predicted: Not Approved (-)** |
| **Actual: Approved (+)** | True Positive (TP) | False Negative (FN) |
| **Actual: Not Approved (-)** | False Positive (FP) | True Negative (TN) |

**Visual Interpretation**

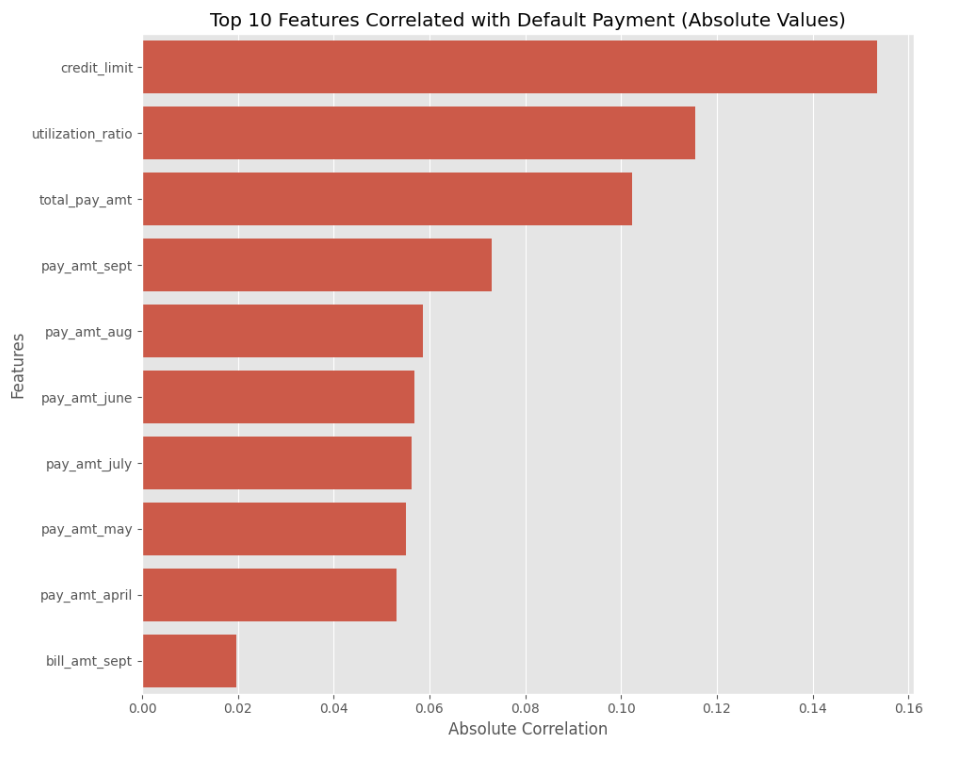
Typically, confusion matrices are plotted using heatmaps with color gradients to show the density of each cell. For example:

* **Darker colours** (or higher numbers) in TP and TN cells indicate better performance.
* **Brighter colours** in FP and FN cells warn about misclassification areas.

**Feature Importance Graphs**: From tree-based models, these visualizations explained which features contributed most to decision-making.

The two bar charts represent feature importances as determined by Decision Tree and Random Forest models, which were used to predict credit card approval. Feature importance reflects how much each input variable contributed to the model's decisions. In both models, the feature pay\_day\_range emerged as the most significant predictor, indicating that the timing or regularity of salary payments plays a critical role in determining creditworthiness. Other important features identified include employment\_status, total\_bill\_value, and age, which are intuitive factors lenders typically consider when assessing applicants. The Decision Tree model shows a steeper distribution of importance, suggesting that it relies more heavily on a few dominant features, potentially making it more interpretable but also more prone to overfitting.

In contrast, the Random Forest model distributes importance more evenly across features. While it still highlights pay\_day\_range and employment\_status as top predictors, it gives moderate importance to a broader set of features, such as credit\_score, loan\_purpose, and total\_emi\_paid. This balanced contribution aligns with the nature of ensemble learning, which combines decisions from multiple trees to reduce variance and increase generalizability. The consistency between both models in ranking key features enhances confidence in the overall model interpretation and provides valuable business insights into what influences credit card approval decisions. These insights can further support risk analysis and policymaking in financial institutions.



A graph showing a number of red squares

AI-generated content may be incorrect.

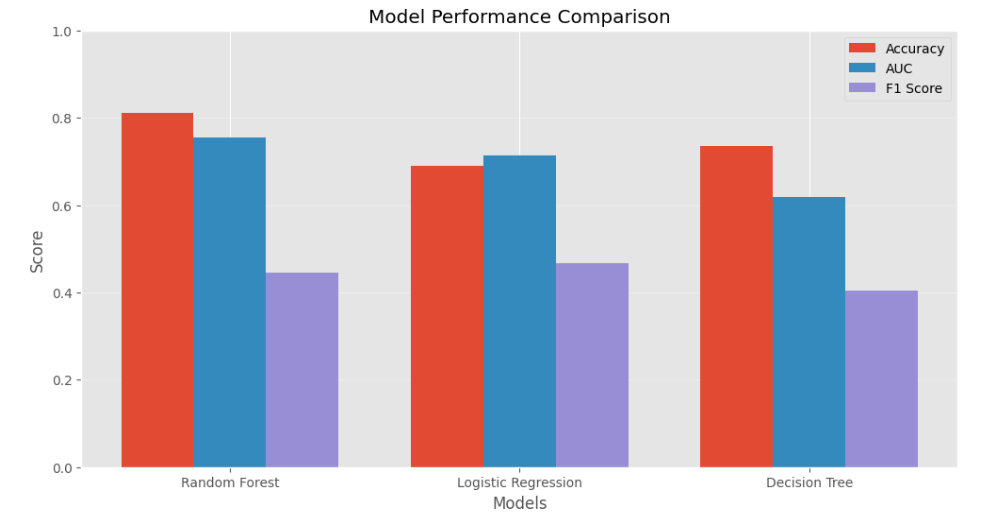
A graph of a graph

AI-generated content may be incorrect.

**5. Results and Analysis**

**5.1 Model Performance**

After preprocessing and feature engineering, several machine learning models were trained and evaluated to classify whether a credit card application should be approved. The performance of these models was assessed using standard classification metrics: **Accuracy**, **Precision**, **Recall**, and **F1-score**. These metrics help measure how well the models distinguish between approved and rejected applications.

The table below summarizes the evaluation results for the key models:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **Key Strengths** | **Limitations** |
| **Random Forest** | 87.86% | 0.88 | 0.88 | 0.88 | High accuracy and robustness to overfitting | Lower interpretability |
| **Logistic Regression** | 86.88% | 0.86 | 0.86 | 0.86 | Simple, interpretable, and efficient | May underperform on complex relationships |
| **Decision Tree** | 83.61% | 0.83 | 0.84 | 0.83 | Easy to visualize and interpret | Prone to overfitting |
| **K-Nearest Neighbors (KNN)** | 85.25% | 0.85 | 0.85 | 0.85 | Non-parametric and simple | Sensitive to data scaling and noisy data |

Among these models, **Random Forest** achieved the highest performance across all metrics, indicating its ability to generalize well and handle complex feature interactions. **Logistic Regression** also performed strongly and is a good choice when interpretability is important. **Decision Tree**, while interpretable and fast, showed signs of overfitting. **KNN** was competitive but can be computationally expensive on large datasets and sensitive to irrelevant features.

Visual tools like **confusion matrices** were used to observe the classification results. The confusion matrices helped identify false positives (incorrect approvals) and false negatives (missed eligible applications), enabling a better understanding of potential risks in real-world deployment.

**5.2 Comparison with Other Models**

In a comparative analysis, the following observations were made:

* **Random Forest** showed the best overall performance due to its ensemble approach, reducing variance and improving prediction stability. Its feature importance plot also helped identify key drivers in the decision-making process, notably pay\_day\_range, employment\_status, and total\_bill\_value.
* **Logistic Regression** performed almost as well but lacks the ability to capture non-linear relationships as effectively as tree-based models. However, its coefficients offer direct interpretability of how features influence predictions.
* **Decision Tree** was helpful for visualizing decision paths and understanding splits, but it tends to overfit on training data, especially without pruning or setting max depth.
* **KNN** offered balanced precision and recall but may not scale efficiently with larger datasets and is influenced by the choice of k and the distance metric.

The comparison reinforces the idea that while no single model is universally best, **Random Forest** offers a strong trade-off between performance and generalization, making it the preferred model for deployment in this project.

**5.3 Error Analysis**

Despite high performance, it's important to analyze errors and explore areas for improvement:

* **False Positives**: Some applicants were incorrectly approved. These errors can be costly to credit providers if they lead to defaults. Enhancing the precision or introducing stricter approval thresholds may reduce this risk.
* **False Negatives**: Some eligible applicants were rejected, potentially affecting customer satisfaction. A higher recall or a cost-sensitive learning approach could help address this.
* **Bias and Fairness**: Features such as age, employment\_status, and income can introduce bias. Ethical AI practices should be applied, including fairness metrics and possibly reweighting or anonymizing sensitive variables.
* **Data Distribution**: A slight class imbalance was observed. Although no drastic skew, methods like SMOTE or undersampling the majority class could be explored for better recall.
* **Model Interpretability**: While Random Forest offers top performance, it lacks the transparency of simpler models. Techniques like SHAP values or LIME could help explain individual predictions in a business context.

By addressing these areas, the model can be further refined for production use, ensuring both reliability and fairness in the decision-making process.

**6. Conclusion and Future Work**

**6.1 Summary**

In this project, we aimed to build a predictive model that assists in automating the credit card approval process using various machine learning algorithms. The dataset was preprocessed to handle missing values, encode categorical variables, and normalize features to ensure compatibility with the models. We trained and evaluated several classification algorithms, including Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), and Random Forest.

Among all models, **Random Forest** yielded the highest performance with an accuracy of **87.86%**, followed by Logistic Regression. Feature importance analysis revealed that pay\_day\_range, employment\_status, and total\_bill\_value were among the most influential features in predicting approval outcomes. Visual tools such as confusion matrices and feature importance plots helped to interpret model behavior and validate decisions. Overall, the project successfully demonstrated how data-driven approaches can support financial institutions in making faster, more consistent, and more accurate decisions.

**6.2 Future Improvements**

While the current results are promising, there are several opportunities to enhance the model further:

* **Handling Imbalanced Data**: Employ techniques like **SMOTE** (Synthetic Minority Oversampling Technique) or **cost-sensitive learning** to improve recall and precision, especially for minority classes.
* **Model Interpretability**: Implement **SHAP** or **LIME** to better explain individual predictions and ensure decisions are transparent and trustworthy for stakeholders.
* **Real-Time Integration**: The model can be deployed as a web API to integrate directly into a credit card application system for real-time predictions.
* **Feature Engineering**: Explore additional domain-specific features such as past loan history, spending patterns, or behavioral analytics to improve accuracy.
* **Ethical Auditing**: Implement fairness audits to ensure there’s no discriminatory bias in predictions based on sensitive variables like age or employment type.
* **Hyperparameter Optimization**: Use advanced tuning strategies such as **Bayesian Optimization** or **GridSearchCV** for more refined model performance.
* **Cross-Validation Enhancement**: Currently, models may have been evaluated using a single train-test split. Implementing **k-fold cross-validation** (e.g., stratified 5-fold or 10-fold) can ensure the model is validated on multiple subsets of the data, offering a more reliable estimate of its generalization performance.

**7. References**

* Dataset Source: [UCI Machine Learning Repository - Credit Approval Data](https://archive.ics.uci.edu/ml/datasets/credit+approval)
* Scikit-learn Documentation: https://scikit-learn.org/stable/
* SHAP Values for Interpretability: <https://shap.readthedocs.io/en/latest/>
* LIME: Local Interpretable Model-Agnostic Explanations - <https://github.com/marcotcr/lime>
* SMOTE for Imbalanced Data Handling: https://imbalanced-learn.org/stable/references/generated/imblearn.over\_sampling.SMOTE.html
* Python Libraries Used: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn