Loan Data Analysis and Prediction

Pramod Giri

Student ID: 23189635

Tutor: Rupak Koirala

Total Page Count: 23

Date: February 22, 2025

Contents

1 Introduction **3**

2 Motivation behind the chosen domain **3**

3 Data Exploration **4**
3.1 About the dataset 4
3.2 Duplicate Values 5
3.3 Data Visualization 5

4 Feature Engineering and Data Preprocessing **10**

5 Split for Test data **11**

6 Standardizing Numerical Features For Machine Learning **12**

7 Applying Various Model for Prediction **12**
7.1 Random Forest 13
7.2 Logistic Regression 14
7.3 Decision Tree 15
7.4 XGBoost 16
7.5 LghtGBM 17
7.6 CatBoost 18

8 Best Model Selection For Loan Prediction **19**

9 Using Shap for Model Interpretation **20**

10 Summary **22**

11 Future Work **22**

12 Conclusion **22**

List of Figures

1 Checking for Duplicate Values 5
2 Loan Status vs Person Income 5
3 Loan Status vs Loan Amount 6
4 Loan Status vs Education Level 6
5 Loan Status vs Person Home Ownership 7
6 Pairplot for Numerical Features 7
7 Boxplot for Outlier Detection 8
8 Histograms for Data Distribution 8
9 Numbers of Outliers Detected 8
10 K-Means Clustering of Load Data 9
11 Feature Engineering: Encoding Categorical Variables 10
12 Replacing Outliers with Median 10
13 Correlation Heatmap 11
14 Split for Test Data 11
15 Standardizing Numerical Features For Machine Learning 12
16 Apply Various Model for prediction 13
17 Classification Report for Random Forest 14
18 Classification Report for Logistic Regression 15
19 Classification Report for Decision Tree 16
20 Classification Report for XGBoost 17
21 Classification Report for LightGBM 18
22 Classification Report for CatBoost 19
23 Best Model Selection 19
24 Waterfall plot for model interpretation 20
25 Bar plot for model interpretation 21
26 Beeswarm plot for model interpretation 21

Comprehensive Analysis of Loan Data: Risk Assessment, Approval Prediction, and Financial Insights from Borrower Profiles

1 Introduction

Machine learning (ML) is like giving computers the ability to learn from experience. Instead of being explicitly programmed for every task, ML systems analyze historical data, such as past financial transactions or customer behavior, to uncover patterns and make informed predictions. In the financial world, this technology is transforming how decisions are made. Automate processes, improve risk assessments, and even personalize customer experiences. For example, supervised learning techniques, where models are trained on labeled data (such as 'approved' or 'denied' loan applications), are widely used in credit scoring, fraud detection and loan approvals. Using vast amounts of data, ML helps financial institutions make smarter, faster and more accurate decisions.

One of the most impactful uses of ML in finance is to predict loan approvals and defaults. Imagine a lender trying to decide whether to approve a loan application. ML models analyze a wealth of historical data, such as an applicant's income, credit score, employment history, and past repayment behavior, to predict the likelihood of repayment. Algorithms such as logistic regression, decision trees, random forests, and even deep learning models sift through this data to identify patterns and assess risk. This not only helps lenders minimize defaults but also speeds up the approval process, making it more efficient and fair. By relying on data-driven insights, financial institutions can make better lending decisions, reduce risks, and ensure that loans are granted responsibly.

2 Motivation behind the chosen domain

The financial sector is vital for economic growth, but lending institutions often struggle with assessing credit risk and reducing loan defaults. Traditional loan approval methods, which rely on manual evaluations and rigid rules, can be slow, inconsistent, and biased. Machine Learning (ML) steps in as a game-changer, offering data-driven solutions to make loan approvals faster, fairer, and more accurate. By analyzing vast amounts of applicant data—like income, credit history, and employment—ML models can predict creditworthiness with remarkable precision, transforming how lending decisions are made.

Beyond improving efficiency, ML also addresses the challenge of financial inclusion. By considering factors beyond traditional credit scores, ML models can identify creditworthy individuals and businesses that conventional systems might overlook. This opens doors for underserved populations to access loans, empowering them while reducing risks for lenders. As FinTech and AI-driven finance continue to grow, specializing in ML-based loan prediction isn't just innovative—it's a step toward a more inclusive and forward-thinking financial future.

3 Data Exploration

3.1 About the dataset

The dataset contains 45,000 loan application records. It includes details about the borrowers, like their age, gender, education, income, job experience, and whether they own a home. It also has loan-related information, such as the loan amount, interest rate, the purpose of the loan (like for personal use, medical needs, or education), and how much of their income goes toward the loan.

The dataset also includes credit-related details, such as *credit score*, *how long they've had credit*, and *if they've failed to repay loans in the past*. The target column, `loan_status`, shows whether the loan was approved (1) or rejected (0).

This data is useful for predicting whether someone is likely to repay a loan. It helps banks and lenders make better decisions about who to lend money to. It's a great resource for understanding credit risk and improving lending processes.

Column Descriptions:

- `person_age` – Age of the borrower (numeric).
- `person_gender` – Gender of the borrower (categorical: "male", "female").
- `person_education` – Educational qualification (categorical: "High School", "Bachelor", "Master", etc.).
- `person_income` – Annual income of the borrower (numeric).
- `person_emp_exp` – Years of employment experience (numeric).
- `person_home_ownership` – Type of home ownership (categorical: "RENT", "OWN", "MORTGAGE").
- `loan_amnt` – Loan amount requested (numeric).
- `loan_intent` – Purpose of the loan (categorical: "PERSONAL", "EDUCATION", "MEDICAL", etc.).
- `loan_int_rate` – Interest rate on the loan (numeric).
- `loan_percent_income` – Loan amount as a percentage of annual income (numeric).
- `cb_person_cred_hist_length` – Credit history length in years (numeric).
- `credit_score` – Borrower's credit score (numeric).
- `previous_loan_defaults_on_file` – Indicates if the borrower has defaulted on previous loans (categorical: "Yes", "No").
- `loan_status` – Loan approval status (binary: 1 for approved, 0 for rejected).

3.2 Duplicate Values

There are no duplicate values in the dataset, making it suitable for analysis. Without imputing or deleting any missing numbers, we can continue with the data exploration and visualization process.

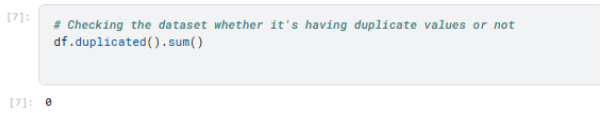


Figure 1: Checking for Duplicate Values

3.3 Data Visualization

The graph visually compares the `person_income` between the two `loan_status` groups. It suggests that there is a difference in income between the two groups. The specific nature of the difference (e.g., whether one group has a significantly higher average income than the other) can be determined by examining the actual bar heights.

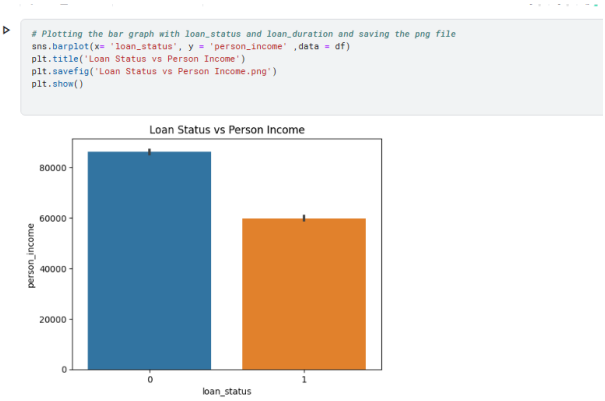


Figure 2: Loan Status vs Person Income

The graph visually compares the `loan_amnt` between the two `loan_status` groups. It suggests that there is a difference in loan amount between the two groups. The specific nature of the difference (e.g., whether one group has a significantly higher average loan amount than the other) can be determined by examining the actual bar heights. In this specific graph, it appears that `loan_status` 1 has a higher average loan amount than `loan_status` 0.

```
[9]: # Plotting the bar graph with loan_status and loan_amount and saving the png file
sns.barplot(x= 'loan_status', y = 'loan_amnt' ,data = df)
plt.title('Loan Status vs Loan Amount')
plt.savefig('Loan Status vs Loan Amount.png')
plt.show()
```

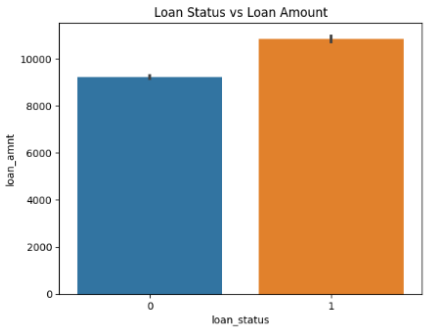


Figure 3: Loan Status vs Loan Amount

This graph shows the relationship between education level and loan status. It suggests that the proportion of individuals with a particular loan status (likely default, based on the range of values) varies across different education levels. For example, it seems that the "Doctorate" level has a lower proportion of individuals with the loan status being analyzed, while "Master" and "High School" have higher proportions.

```
# Plotting the bar graph with loan_status and person_education and saving the png file
sns.barplot(x= 'loan_status', y = 'person_education' ,data = df)
plt.title('Loan Status vs Person Education')
plt.savefig('Loan Status vs Person Education.png')
plt.show()
```

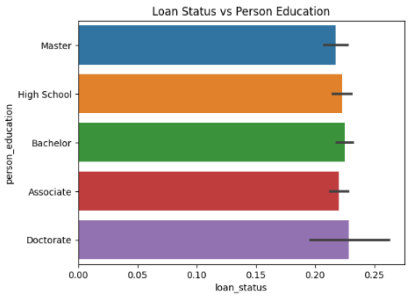


Figure 4: Loan Status vs Education Level

The below graph shows how loan status is distributed across different home ownership categories. It suggests that there's a difference in the proportion of individuals with a certain loan status (again, likely default) depending on whether they rent, own, have a mortgage, or fall into the "other" category. The "OTHER" category seems to have the highest proportion, followed by "RENT", while "OWN" and "MORTGAGE" have lower proportions.

```
# Plotting the bar graph with loan_status and person_home_ownership and saving the png file
sns.barplot(x= 'loan_status', y = 'person_home_ownership' ,data = df)
plt.title('Loan Status vs Person Home Ownership')
plt.savefig('Loan Status vs Person Home Ownership.png')
plt.show()
```

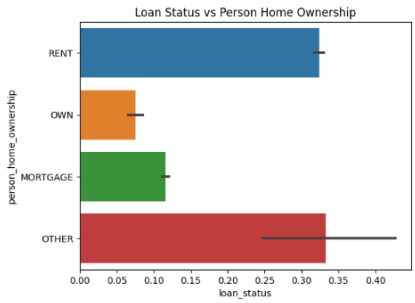


Figure 5: Loan Status vs Person Home Ownership

The pair plot displays scatter plots for each pair of variables (below the diagonal) and kernel density estimations for individual variables (on the diagonal), allowing for analysis of correlations, patterns, and distributions; the corner=True argument displays only the lower triangle for brevity.

```
# Pairplot for numerical features
# Define numerical columns for visualization
num_cols = ["person_age", "person_income", "person_emp_exp", "loan_amnt", "loan_int_rate", "credit_score"]
sns.pairplot(df[num_cols], diag_kind='kde', corner=True)
plt.suptitle("Pairplot for Numerical Features", fontsize=16)
plt.show()
```

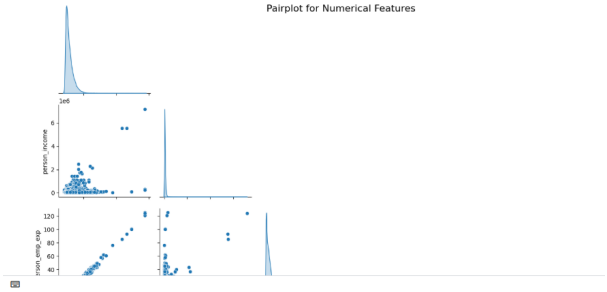


Figure 6: Pairplot for Numerical Features

The code iterates through specified columns (`num_cols`) and creates a 2x3 grid of box plots, each showing the distribution of a single column. The resulting figure helps in identifying data points that fall outside the typical range of each variable, suggesting possible outliers.

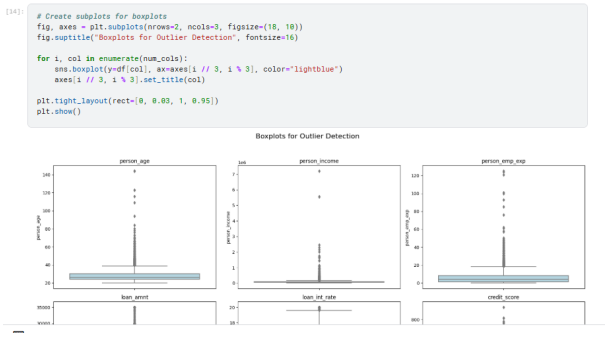


Figure 7: Boxplot for Outlier Detection

The histograms allow for visual inspection of the distribution shape, central tendency, and spread of each numerical variable.

Figure 8: Histograms for Data Distribution

The image displays the number of outliers found in several different data categories (like age, income, employment experience, loan amount, interest rate, and credit score).

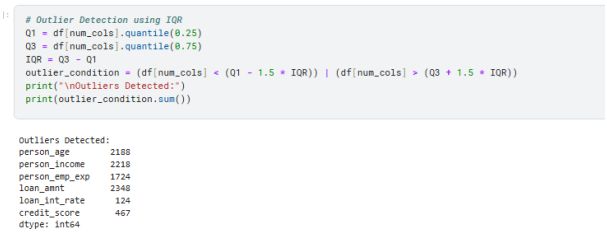


Figure 9: Numbers of Outliers Detected

The scatter plot represents K-Means clustering of loan data based on Person Income and Loan Amount, dividing the data into three clusters. Cluster 0 (purple) includes low-income individuals taking smaller loans, Cluster 1 (teal) spans a broad income range with varying loan amounts, and Cluster 2 (yellow) likely represents slightly higher-income individuals. Most data points are concentrated at lower income levels, with a few outliers having extremely high incomes. This clustering helps identify patterns in loan distribution, potentially aiding financial institutions in risk assessment and decision-making.

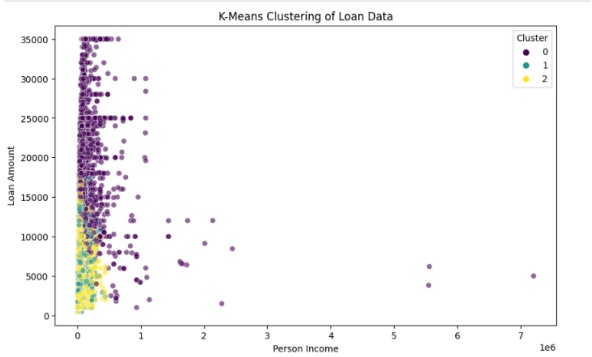
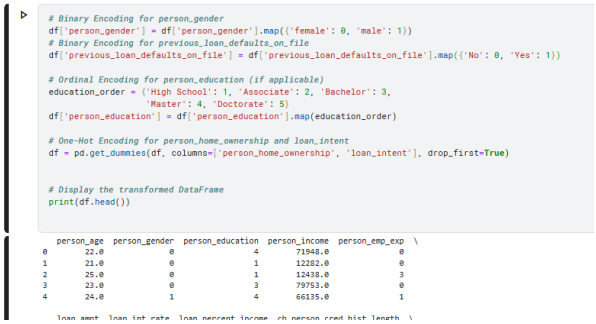


Figure 10: K-Means Clustering of Load Data

4 Feature Engineering and Data Preprocessing

The code demonstrates common feature engineering practices aimed at converting categorical variables into numerical representations, a crucial step for enabling compatibility with most machine learning algorithms. It employs binary encoding for features with two categories, ordinal encoding for ordered categorical variables, and one-hot encoding for unordered categories. These techniques ensure that the categorical data is appropriately transformed into a format suitable for algorithmic processing. The `print(df.head())` statement provides a preview of the dataset, allowing users to verify the results of these transformations and ensuring that the encoding process has been applied correctly. This approach enhances the dataset's readiness for subsequent machine learning tasks.



```
# Binary Encoding for person_gender
df['person_gender'] = df['person_gender'].map({'female': 0, 'male': 1})
# Binary Encoding for previous_loan_defaults_on_file
df['previous_loan_defaults_on_file'] = df['previous_loan_defaults_on_file'].map({'No': 0, 'Yes': 1})

# Ordinal Encoding for person_education (if applicable)
education_order = {'High School': 1, 'Associate': 2, 'Bachelor': 3,
                   'Master': 4, 'Doctorate': 5}
df['person_education'] = df['person_education'].map(education_order)

# One-Hot Encoding for person_home_ownership and loan_intent
df = pd.get_dummies(df, columns=['person_home_ownership', 'loan_intent'], drop_first=True)


# Display the transformed DataFrame
print(df.head())
```

```
   person_age  person_gender  person_education  person_income  person_emp_exp  \
0        22.0              0                 4        71948.0               0
1        21.0              0                 1        12282.0               0
2        25.0              0                 1        12438.0               3
3        23.0              0                 3        79753.0               0
4        24.0              1                 4        66135.0               1
```

Figure 11: Feature Engineering: Encoding Categorical Variables

The provided code addresses outliers in the `person_age` column by replacing values exceeding 100 with the median age, a method commonly employed to mitigate the influence of extreme data points on statistical analyses and machine learning models. This approach ensures that the central tendency of the dataset remains representative without being disproportionately affected by anomalous entries. The median is favored over the mean for such replacements due to its inherent resistance to outliers; unlike the mean, which can be significantly skewed by extreme values, the median provides a more robust measure of the dataset's central value. By employing this technique, the integrity of the data is preserved, enhancing the reliability of subsequent analytical processes and model training phases.

```
# Replacing Outliers with Median
median_age = df['person_age'].median()
df['person_age'] = df['person_age'].apply(lambda x: median_age if x > 100 else x)
df['person_age']
```

```
0        22.0
1        21.0
2        25.0
3        23.0
4        24.0
        ...
44995    27.0
44996    37.0
44997    33.0
44998    29.0
44999    24.0
Name: person_age, Length: 45000, dtype: float64
```

Figure 12: Replacing Outliers with Median

The resulting heatmap, titled "Correlation Heatmap," provides a clear visual representation of the pairwise correlations between all numerical columns in the data frame, allowing for quick identification of strong positive or negative relationships.

Figure 13: Correlation Heatmap

5 Split for Test data

The below Python code snippet utilizing the Pandas and Scikit-learn libraries for preparing data before training a machine learning model. The code begins by separating the dataset into features (X) and the target variable (y). Specifically, $X =$ `df.drop(columns=['loan_statu` creates the feature matrix X by removing the `loan_status` column from the original DataFrame `df`. Simultaneously, $y =$ `df.loan_status` assigns the `loan_status` column to the target variable y. To provide a quick overview of the data structure, the code displays the first few rows of both the feature matrix X and the target variable y using `display(X.head())` and `display(y.head())`.

Next, the code splits the data into training and validation sets using the `train_test_split` function from Scikit-learn. The line $X\_train, X\_val, y\_train, y\_val =$ `train_test_split(X, y, test_s` divides the features (X) and the target variable (y) into training sets (X_train, y_train) and validation sets (X_val, y_val). The argument `test_size=0.3` specifies that 30% of the data is reserved for validation, while the remaining 70% is used for training. The `random_state=42` parameter ensures reproducibility by fixing the random number generator's seed, guaranteeing the same split each time the code is executed. This step is essential for evaluating the model's performance on unseen data during the training process. The displayed tables showcase the first five rows of the features (X) and the target variable (y) after the split.

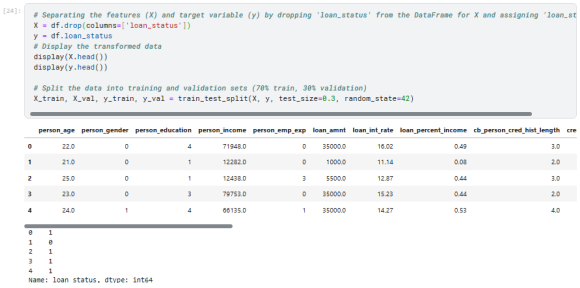


Figure 14: Split for Test Data

6 Standardizing Numerical Features For Machine Learning

This code snippet demonstrates the standardization of numerical features in both the training (X_train) and validation (X_val) datasets using Scikit-learn's `StandardScaler`. The process begins by initializing the scaler. The scaler is then fitted exclusively on the training data to compute the mean and standard deviation of each feature. Once the scaler has learned these parameters, it is applied to transform both the training and validation datasets. This transformation ensures that each feature in the datasets has a mean of 0 and a standard deviation of 1, which is a common requirement for many machine learning algorithms. Finally, the transformed data is converted back into Pandas DataFrames, preserving the original column names and indices for ease of use in subsequent machine learning tasks.

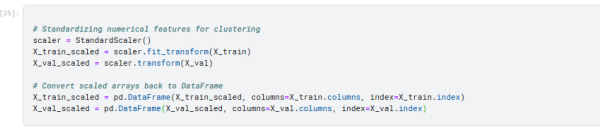


Figure 15: Standardizing Numerical Features For Machine Learning

7 Applying Various Model for Prediction

The code defines and evaluates multiple machine learning models to classify data, comparing their performance on a validation set. It starts by creating a dictionary called `models` containing six classifiers: Random Forest, Logistic Regression, Decision Tree, XGBoost, LightGBM, and CatBoost, each initialized with specific parameters for reproducibility. The code then iterates through each model, training it on the training data (X_train, y_train) and making predictions on the validation set (X_val). For each model, it calculates the training and validation accuracy scores, as well as the overall accuracy using `accuracy_score`. The results, including the model name, training score, validation score, and accuracy, are stored in a list called `results`. Additionally, the code generates a classification report and a confusion matrix for each model, visualized using a heatmap, to provide detailed insights into the model's performance. Finally, the results are compiled into a DataFrame (`results_df`) and displayed, summarizing the performance of all models for easy comparison. This process ensures a comprehensive evaluation of each model's effectiveness in handling the classification task.

```python
# Define models
models = {
    "Random Forest": RandomForestClassifier(random_state=42),
    "Logistic Regression": LogisticRegression(max_iter=1000, random_state=42),
    "Decision Tree": DecisionTreeClassifier(),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42),
    "LightGBM": LGBMClassifier(verbosity=-1, random_state=42),
    "CatBoost": CatBoostClassifier(verbose=0, random_state=42)

}

results = []

for name, model in models.items():
    model.fit(X_train, y_train)

    # Predictions on validation set
    y_val_pred = model.predict(X_val)

    # Train and Test Scores
    train_score = model.score(X_train, y_train)
    test_score = model.score(X_val, y_val)

    # Accuracy Score
    accuracy = accuracy_score(y_val, y_val_pred)

    results.append({
        'Model': name,
        'Train Score': train_score,
        'Test Score': test_score,
        'Accuracy Score': accuracy
    })
```

Figure 16: Apply Various Model for prediction

7.1 Random Forest

The Random Forest model demonstrates strong overall performance, achieving an accuracy of 0.93. However, it performs better at predicting rejected loans (class 0) than approved loans (class 1), as evidenced by higher precision, recall, and F1-scores for class 0. The lower recall for class 1 indicates that the model fails to identify a significant portion of loans that should have been approved. This issue may stem from factors such as class imbalance (fewer examples of approved loans in the dataset) or limitations in the features used for training. The confusion matrix offers a detailed breakdown of the model's performance, revealing specific strengths and weaknesses. In a real-world loan approval system, it is essential to strike a careful balance between precision and recall, as failing to identify potentially good loan applications (false negatives) can have significant business consequences.

Classification Report for Random Forest:

	precision	recall	f1-score	support
0	0.94	0.97	0.95	10493
1	0.89	0.77	0.82	3007
accuracy			0.93	13500
macro avg	0.91	0.87	0.89	13500
weighted avg	0.93	0.93	0.92	13500

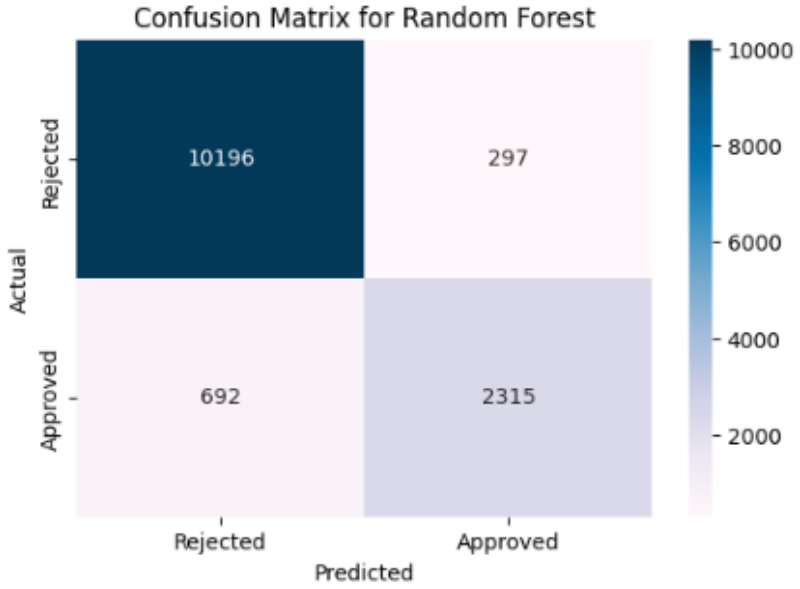


Figure 17: Classification Report for Random Forest

7.2 Logistic Regression

The Logistic Regression model shows moderate performance, achieving an overall accuracy of 0.82. However, there is a noticeable difference in its ability to predict rejected loans compared to approved loans. The model performs significantly better at identifying loan rejections (class 0), as reflected by its higher precision, recall, and F1-score for this class. On the other hand, it struggles to correctly predict approved loans (class 1), with a particularly low recall score. This indicates that the model fails to capture a large portion of loan applications that should have been approved, which could be problematic in real-world scenarios where identifying good loan candidates is critical.

Classification Report for Logistic Regression:

	precision	recall	f1-score	support
0	0.84	0.95	0.89	10493
1	0.68	0.38	0.49	3007
accuracy			0.82	13500
macro avg	0.76	0.66	0.69	13500
weighted avg	0.81	0.82	0.80	13500

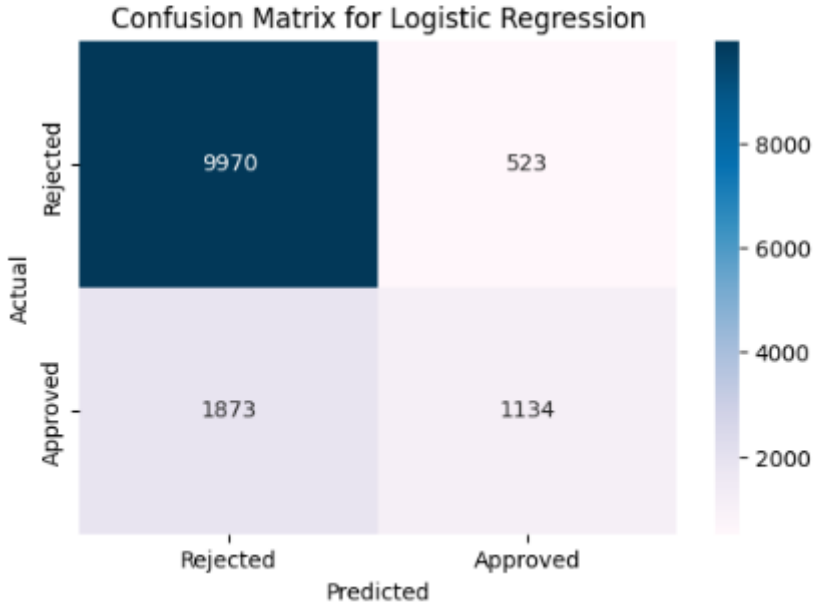


Figure 18: Classification Report for Logistic Regression

7.3 Decision Tree

The Decision Tree model demonstrates strong overall performance, achieving an accuracy of 0.90. It shows a balanced ability to predict both rejected and approved loans, as evidenced by the similar precision and recall values for both classes. This suggests that the model is equally effective at identifying loan rejections (class 0) and approvals (class 1). However, the confusion matrix reveals that there are still some misclassifications, as indicated by the off-diagonal elements. While the model performs well, these errors highlight areas where further refinement could improve its predictive capabilities.

Classification Report for Decision Tree:

	precision	recall	f1-score	support
0	0.94	0.94	0.94	10493
1	0.78	0.78	0.78	3007
accuracy			0.90	13500
macro avg	0.86	0.86	0.86	13500
weighted avg	0.90	0.90	0.90	13500

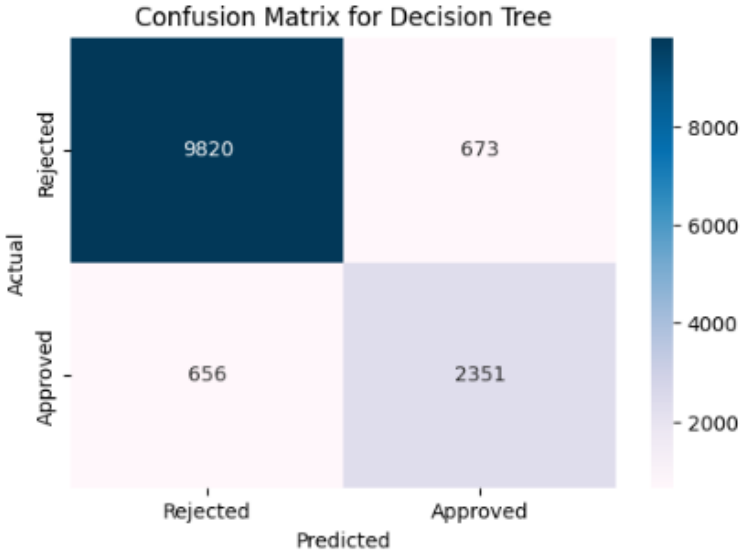


Figure 19: Classification Report for Decision Tree

7.4 XGBoost

The XGBoost model delivers strong overall performance, achieving an accuracy of 0.93. It excels at predicting both rejected and approved loans, as reflected by its relatively high precision and recall scores for both classes. When compared to the Decision Tree model, XGBoost achieves a similar level of accuracy but shows a better balance between precision and recall, particularly for the "Approved" class. This suggests that XGBoost is more effective at correctly identifying loan approvals while maintaining robust performance across both classes, making it a reliable choice for this classification task.

Classification Report for XGBoost:

	precision	recall	f1-score	support
0	0.94	0.97	0.96	10493
1	0.88	0.80	0.84	3007
accuracy			0.93	13500
macro avg	0.91	0.88	0.90	13500
weighted avg	0.93	0.93	0.93	13500

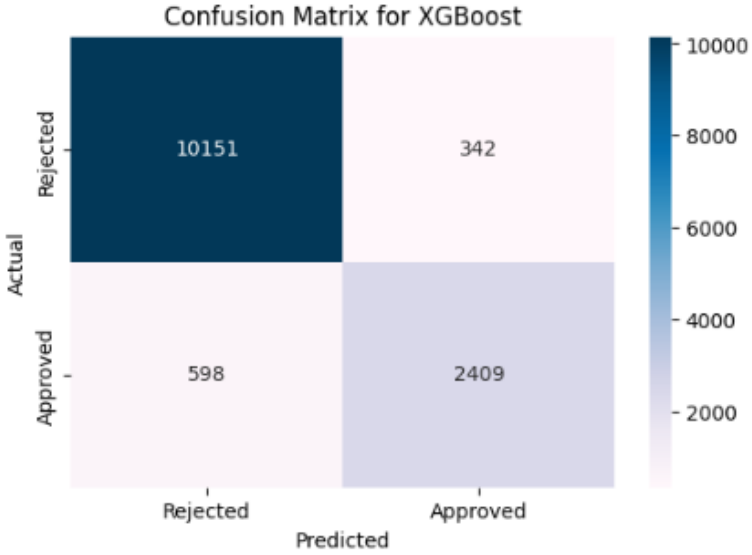


Figure 20: Classification Report for XGBoost

7.5 LghtGBM

The LightGBM model performs exceptionally well, achieving an overall accuracy of 0.93. It shows a strong capability to predict both rejected and approved loans, with high precision and recall scores for both classes. Its performance is nearly identical to that of the XGBoost model, indicating that both algorithms are highly effective for this particular task. This consistency in results highlights the robustness of these models in handling the classification challenge at hand.

Classification Report for LightGBM:

	precision	recall	f1-score	support
0	0.94	0.97	0.95	10493
1	0.87	0.80	0.83	3007
accuracy			0.93	13500
macro avg	0.91	0.88	0.89	13500
weighted avg	0.93	0.93	0.93	13500

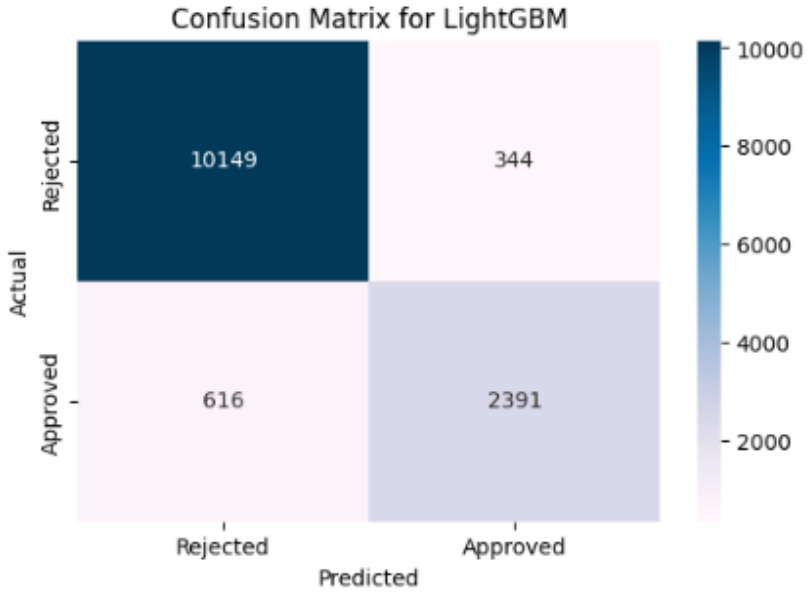


Figure 21: Classification Report for LightGBM

7.6 CatBoost

The CatBoost model delivers excellent performance, achieving an overall accuracy of 0.93. It excels at predicting both rejected and approved loans, with high precision and recall scores for both classes. Its performance is nearly identical to that of the XGBoost and LightGBM models, indicating that all three gradient boosting algorithms are highly effective for this task. This consistency across models underscores the strength of gradient boosting techniques in handling the classification challenge, making them reliable choices for this type of problem.

Classification Report for CatBoost:

	precision	recall	f1-score	support
0	0.94	0.97	0.96	10493
1	0.89	0.80	0.84	3007
accuracy			0.93	13500
macro avg	0.91	0.88	0.90	13500
weighted avg	0.93	0.93	0.93	13500

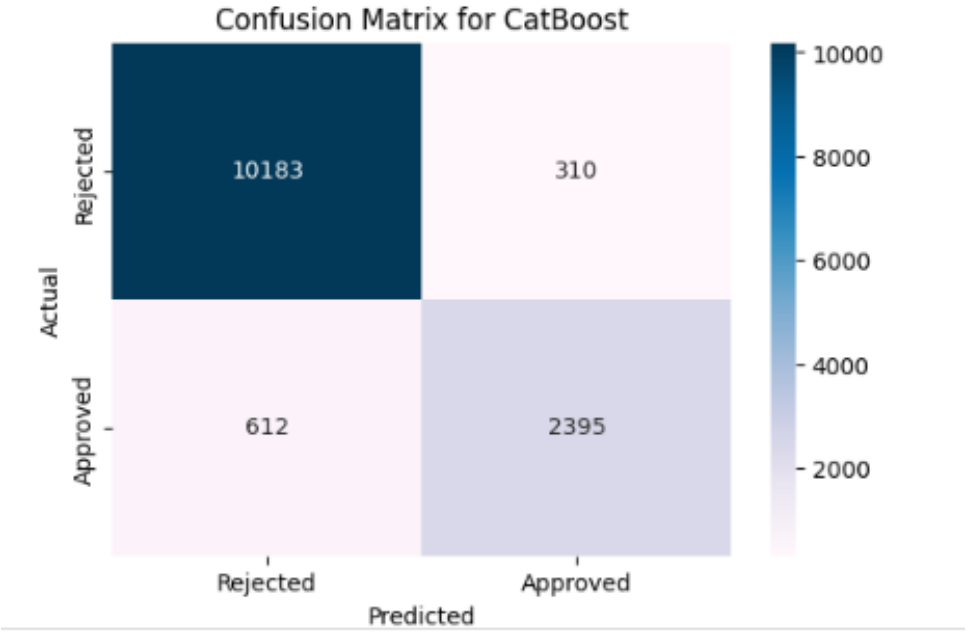
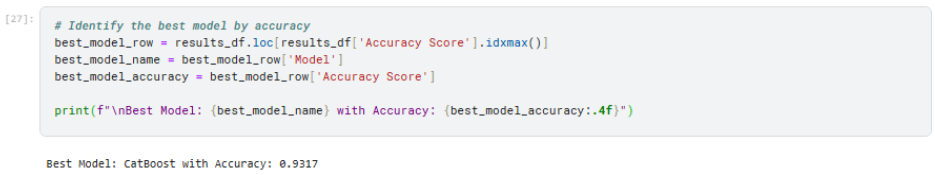


Figure 22: Classification Report for CatBoost

8 Best Model Selection For Loan Prediction

The output displayed in the image, labeled `Best Model: CatBoost with Accuracy: 0.9317`, highlights that the CatBoost model achieved the highest accuracy score of 0.9317 among all the models evaluated and stored in the `results_df` DataFrame. This result underscores CatBoost's superior performance in this classification task, making it the top-performing model in the comparison.



```
[27]: # Identify the best model by accuracy
best_model_row = results_df.loc[results_df['Accuracy Score'].idxmax()]
best_model_name = best_model_row['Model']
best_model_accuracy = best_model_row['Accuracy Score']

print(f"\nBest Model: {best_model_name} with Accuracy: {best_model_accuracy:.4f}")
```

```
Best Model: CatBoost with Accuracy: 0.9317
```

Figure 23: Best Model Selection

9 Using Shap for Model Interpretation

The waterfall plot illustrates the contribution of each feature in shifting the model's prediction from the base value—the average expected prediction across all data points—to the final predicted value for this specific loan application, which is 6.995. It provides a clear breakdown of how individual features influence the model's decision, helping to understand the factors driving the prediction for this particular case.

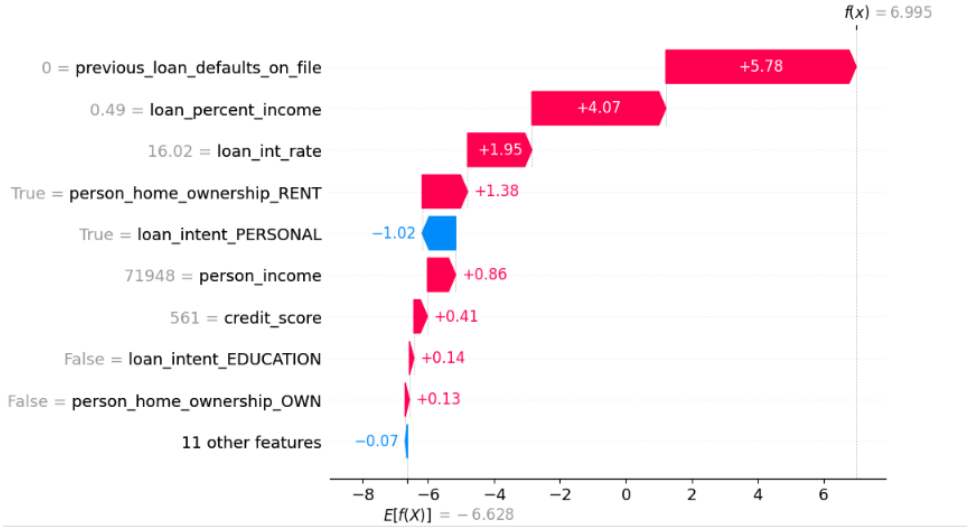


Figure 24: Waterfall plot for model interpretation

The bar plot ranks the features based on their importance, as measured by the mean absolute SHAP values across all data instances. It also visually indicates whether each feature has a positive or negative impact on the model's predictions. By highlighting both the significance and the direction of each feature's influence, the plot provides valuable insights into how the model makes decisions and which factors play the most critical roles in shaping its outputs.

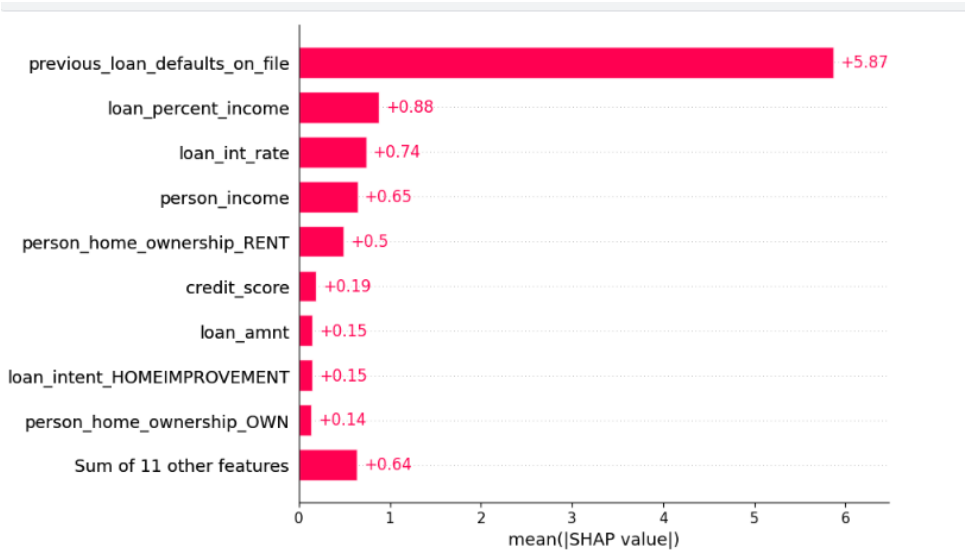


Figure 25: Bar plot for model interpretation

The beeswarm plot displays the distribution of SHAP values for each feature across all data instances. It not only highlights the importance of each feature but also illustrates how the impact of a feature changes based on its value. By showing the range and variability of each feature's influence, the plot provides deeper insights into how different values of a feature contribute to the model's predictions, helping to better understand the model's behavior and decision-making process.

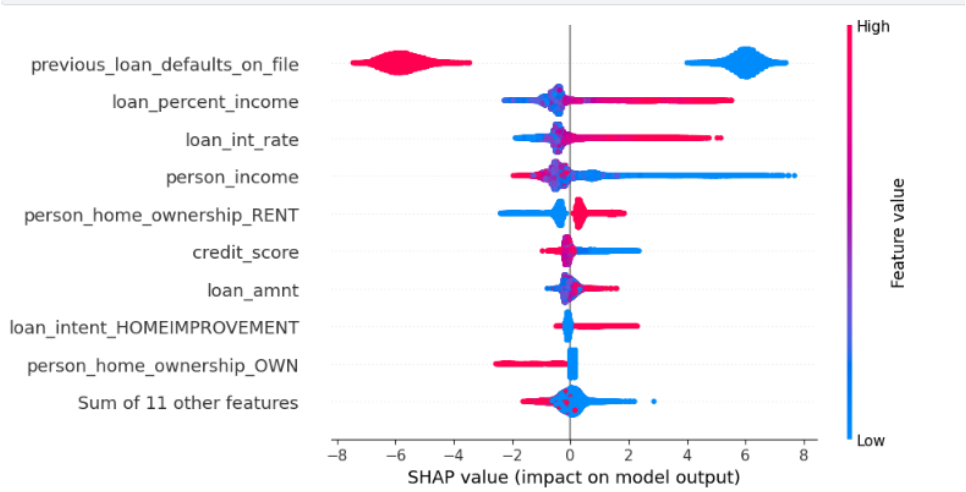


Figure 26: Beeswarm plot for model interpretation

10 Summary

The report provides a comprehensive analysis of the loan approval prediction process, starting with data exploration, which includes visualizations and outlier detection. It then delves into feature engineering, where categorical variables are encoded, and outliers are addressed to prepare the data for modeling. Several machine learning models—Random Forest, Logistic Regression, Decision Tree, XGBoost, LightGBM, and CatBoost—are trained and evaluated using key metrics such as accuracy, precision, recall, and F1-score. Their performance is visualized through classification reports and confusion matrices, offering a clear comparison of their strengths and weaknesses. Among these, the CatBoost model emerges as the top performer based on its accuracy. To better understand the model's predictions, SHAP values are used to interpret feature importance and their impact on individual loan applications. The report concludes that machine learning can effectively predict loan approvals, emphasizing the importance of selecting the right model and ensuring interpretability to make informed decisions.

11 Future Work

Further feature engineering, including creating new features from existing ones or refining current features, might uncover additional predictive power and improve the model's ability to generalize to unseen data. Exploring model stacking or ensembling, where predictions from multiple models are combined, could leverage the strengths of each individual model and lead to more accurate and reliable results. Beyond SHAP values, investigating other explainability methods could provide a more comprehensive understanding of the model's decision-making process. Addressing the potential class imbalance by collecting more data, particularly for the minority class (approved loans), could improve the model's ability to accurately predict loan approvals. Deployment of the model into a practical application would allow for real-world testing and usage in a loan approval workflow. A cost-benefit analysis evaluating the financial impact of the model's predictions would be crucial for assessing its value and guiding business decisions. Finally, analyzing the model for potential bias and implementing strategies to ensure fairness and prevent discriminatory lending practices are essential for responsible AI development.

12 Conclusion

To sum up, This project explored how machine learning can be applied to predict loan approvals using a dataset of 45,000 loan applications. By thoroughly analyzing the data, engineering meaningful features, and training and evaluating multiple machine learning models, the CatBoost algorithm stood out as the best performer, achieving an impressive accuracy of 93.17%. Using SHAP value analysis, the project also uncovered the key factors influencing the model's decisions, such as previous loan defaults, the percentage of income allocated to the loan, and the loan interest rate. These insights highlight the potential of machine learning to improve loan approval processes, enabling faster, more accurate, and potentially fairer lending decisions through data-driven approaches.