Loan Prediction Model

Prepared By:

Name	Student ID
Wajiha Adnan	DT-23001

Instructor: Miss Mehar Fatima
Department: Computer Science and Information Technology
Institution: NED University of Engineering and Technology

Table of Contents

Abstract

This report analyzes loan default patterns using a real-world financial dataset and machine learning techniques. The objective is to identify key socioeconomic and financial indicators that influence a borrower's likelihood of loan default. The dataset underwent extensive preprocessing, including handling missing values, correcting inconsistencies, detecting errors, and treating outliers using both Yeo–Johnson transformation and Winsorization. Exploratory Data Analysis (EDA), correlation mapping, and predictive modeling were applied to uncover trends and evaluate the contribution of variables such as income, credit score, employment experience, and loan amount. The findings highlight dominant risk factors, illustrate borrower behavior patterns, and demonstrate the importance of data-driven decision-making for improving loan approval strategies and minimizing financial losses.

1. Introduction

Loan default prediction is essential for financial institutions to manage credit risk and make informed lending decisions. As digital lending grows, understanding borrower behavior through data analysis has become increasingly important. This project, titled **"Loan Default Prediction Using Machine Learning,"** examines key demographic and financial factors—such as income, credit score, employment experience, and loan amount—to identify patterns linked to default risk. Through data preprocessing, exploratory analysis, and predictive modeling, the study aims to classify borrowers accurately and support data-driven loan approval strategies.

2. Dataset Description

The dataset used in this project was obtained from Kaggle and contains detailed borrower and loan-related information commonly used in credit risk analysis.

Column	Description	Type
`person_age`	Age of the person	Float
`person_gender`	Gender of the person	Categorical
`person_education`	Highest education level	Categorical
`person_income`	Annual income	Float
`person_emp_exp`	Years of employment experience	Integer
`person_home_ownership`	Home ownership status (e.g., rent, own, mortgage)	Categorical
`loan_amnt`	Loan amount requested	Float
`loan_intent`	Purpose of the loan	Categorical
`loan_int_rate`	Loan interest rate	Float
`loan_percent_income`	Loan amount as a percentage of annual income	Float
`cb_person_cred_hist_length`	Length of credit history in years	Float
`credit_score`	Credit score of the person	Integer
`previous_loan_defaults_on_file`	Indicator of previous loan defaults	Categorical
`loan_status` (target variable)	Loan approval status: 1 = approved; 0 = rejected	Integer

3.DataSet Summary

Total records: 45003

```
df.shape
```

```
(45003, 14)
```

➢ Extracting:

```
df = pd.read_csv("loan_data.csv")
```

```
df.head(2)
```

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_sco
0	22	female	Master	71948.0	0	RENT	35000	PERSONAL	16.02	0.49	3	5
1	24	male	Master	NaN	1	NaN	45000	EDUCATION	12.09	0.08	2	3

revious_loan_defaults_on_file	loan_status
No	1
No	0

➢ Detail Description:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45003 entries, 0 to 45002
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   person_age                      45003 non-null  int64
 1   person_gender                   45003 non-null  object
 2   person_education                45003 non-null  object
 3   person_income                   45002 non-null  float64
 4   person_emp_exp                  45003 non-null  int64
 5   person_home_ownership           45002 non-null  object
 6   loan_amnt                       45003 non-null  int64
 7   loan_intent                     45003 non-null  object
 8   loan_int_rate                   45003 non-null  float64
 9   loan_percent_income             45003 non-null  float64
 10  cb_person_cred_hist_length      45003 non-null  int64
 11  credit_score                    45003 non-null  int64
 12  previous_loan_defaults_on_file  45003 non-null  object
 13  loan_status                     45003 non-null  int64
dtypes: float64(3), int64(6), object(5)
memory usage: 4.8+ MB
```

➢ Missing Values

```
df.isnull().sum()
```

	0
person_age	0
person_gender	0
person_education	0
person_income	1
person_emp_exp	0
person_home_ownership	1
loan_amnt	0
loan_intent	0
loan_int_rate	0
loan_percent_income	0
cb_person_cred_hist_length	0
credit_score	0
previous_loan_defaults_on_file	0
loan_status	0

Person_income and person_home_ownership as a null value

➢ Duplicated Rows:

```
df.duplicated().sum()
```

```
np.int64(2)
```

2 duplicates row found

➢ Class Imbalancing:

```
df["loan_status"].value_counts()
```

loan_status	count
0	35001
1	10002

dtype: int64

➢ Detecting Error / Outlier:

Detect & Correct Errors

```
num_cols = ['person_age','person_income','person_emp_exp','loan_amnt','loan_int_rate',
            'loan_percent_income','cb_person_cred_hist_length','credit_score']
```

```
skewness = df[num_cols].skew()
print("Skewness of numerical features:\n", skewness)
```

```
Skewness of numerical features:
 person_age                    2.548186
person_income                34.137942
person_emp_exp                2.594940
loan_amnt                     1.182051
loan_int_rate                 0.213762
loan_percent_income           1.034546
cb_person_cred_hist_length    1.631728
credit_score                 -0.614315
dtype: float64
```

Some of the coluns have high skenwess maybe rising outlier

4. Methodology

The methodology of this project outlines the systematic workflow followed for data acquisition, preprocessing, visualization, model development, and evaluation. Each step ensures data integrity, consistency, and reliable application of machine learning techniques for predicting loan defaults.

1. **Data Acquisition:**
   The loan default dataset was downloaded from Kaggle through manual download. The CSV file was loaded into Python using Pandas for further analysis and processing.
2. **Data Exploration:**
   Initial inspection was performed using functions such as df.head(), df.shape(), df.info(), and df.describe() to understand the dataset structure, detect missing values, examine distributions, and identify numerical and categorical variables.
3. **Data Preprocessing:**
   Comprehensive preprocessing steps were applied, including handling missing values using appropriate imputation strategies, removing duplicate records, fixing inconsistent data formats, and correcting errors detected during validation. Employment experience, income entries, and categorical inconsistencies were checked and cleaned.
4. **Outlier Detection & Treatment:**
   Outliers in numerical variables such as *person_age, person_income, loan_amnt,* and *credit_score* were detected using the Interquartile Range (IQR) method. To reduce skewness and stabilize variance, Yeo–Johnson transformation was applied to multiple numeric columns. Additionally, Winsorization (IQR-based capping) was used on heavy-tailed features to prevent extreme values from negatively impacting model training.
5. **Feature Transformation:**
   Categorical variables including *person_gender, person_home_ownership,* and *loan_intent* were encoded using Label Encoding and One-Hot Encoding where necessary. Numerical features were standardized using StandardScaler to improve model convergence and ensure uniform feature scaling.
6. **Data Visualization:**
   Exploratory Data Analysis (EDA) was conducted using Matplotlib, Seaborn, and Plotly to visualize distributions, detect class imbalance, analyze correlations, and observe the behavior of variables before and after transformation. Boxplots and histograms were used to represent outliers and the effect of winsorization.

7. **Model Selection and Implementation:**
   Multiple machine learning models were implemented for the classification task, including Logistic Regression and Random Forest Classifier. The dataset was split into training and testing subsets using train_test_split() to ensure unbiased model evaluation. Model pipelines were constructed to integrate preprocessing and prediction steps efficiently.
8. **Model Evaluation:**
   Model performance was assessed using accuracy, precision, recall, F1-score, and confusion matrices. The results of different algorithms were compared to determine which model provided the most reliable prediction of loan default risk.

5. Data Preprocessing

Handling Numeric Missing Values

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
df['person_income'] = imputer.fit_transform(df[['person_income']])
```

Handling Categorial Missing Values

```
mode_value = df['person_home_ownership'].mode()[0]

df['person_home_ownership'] = df['person_home_ownership'].fillna(mode_value)
```

Fix Inconsistent Formats

```
df['person_gender'] = df['person_gender'].str.title()
df['person_education'] = df['person_education'].str.title()
df['person_home_ownership'] = df['person_home_ownership'].str.title()
df['loan_intent'] = df['loan_intent'].str.title()
```

Format Datatypes

```
df["person_age"] = df["person_age"].astype(np.int16)
```

Handling Duplicates

```
df[df.duplicated(keep = False)]
```

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_
6	21	Female	High School	12951.0	0	Own	
7	21	Female	High School	12951.0	0	Own	
26	22	Male	Bachelor	97420.0	1	Rent	3
27	22	Male	Bachelor	97420.0	1	Rent	3

```
df = df.drop_duplicates()
```

Outlier Removal:

```
from feature_engine.outliers import Winsorizer

winsor = Winsorizer(
    capping_method='iqr',
    tail='both',
    fold=1.5,
    variables=numeric_cols
)

df[numeric_cols] = winsor.fit_transform(df[numeric_cols])

print(" Winsorization completed for numeric columns.")
print(numeric_cols)
```

```
 Winsorization completed for numeric columns.
['person_age', 'person_income', 'person_emp_exp', 'loan_amnt', 'loan_int_rate', 'loan_percent_income', 'cb_person_cred_hist_length', 'credit_score']
```

5.5 Data Verification:

- Before:

```
df.shape
```

```
(45003, 14)
```

```
Skewness of numerical features:
 person_age                    2.548186
person_income                 34.137942
person_emp_exp                 2.594940
loan_amnt                      1.182051
loan_int_rate                  0.213762
loan_percent_income            1.034546
cb_person_cred_hist_length     1.631728
credit_score                  -0.614315
dtype: float64
```

- After:

```
df.shape
```

```
(45001, 14)
Skewness of numerical features:
 person_age                    0.914670
person_income                 0.874619
person_emp_exp                1.073219
loan_amnt                     0.813509
loan_int_rate                 0.207146
loan_percent_income           0.828131
cb_person_cred_hist_length    1.105052
credit_score                 -0.532783
dtype: float64
```

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score
0	22	Female	Master	71948.0	0	Rent	35000	Personal	16.02	0.49	3	561
1	24	Male	Master	67048.0	1	Rent	45000	Education	12.09	0.08	2	321
2	21	Female	High School	12282.0	0	Own	1000	Education	11.14	0.08	2	504
3	25	Female	High School	12438.0	3	Mortgage	5500	Medical	12.87	0.44	3	635
4	23	Female	Bachelor	79753.0	0	Rent	35000	Medical	15.23	0.44	2	675

Loading

```
df.to_csv("ETL_Perfromed_dataset.csv")
```

(csv)Extract -> Transform -> load(csv)

5.6 Feature Scaling:

➢ Yeo-Johnson:

```
from sklearn.preprocessing import PowerTransformer
# High-skew numeric columns (skew > 0.75)
high_skew_cols = [
    'person_age',
    'person_income',
    'person_emp_exp',
    'loan_amnt',
    'loan_percent_income',
    'cb_person_cred_hist_length'
]
pt = PowerTransformer(method='yeo-johnson', standardize=False)
df[high_skew_cols] = pt.fit_transform(df[high_skew_cols])
print("Skewness after Yeo-Johnson transformation:")
print(df[high_skew_cols].skew())
```

```
Skewness after Yeo-Johnson transformation:
person_age                   0.123188
person_income               -0.009936
person_emp_exp              -0.046653
loan_amnt                   -0.041961
loan_percent_income          0.090951
cb_person_cred_hist_length   0.068107
dtype: float64
```

➢ Standard Scaler:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
print("Skewness after Yeo-Johnson (high-skew features):")
print(df[high_skew_cols].skew())
```

```
Skewness after Yeo-Johnson (high-skew features):
person_age                   0.123188
person_income               -0.009936
person_emp_exp              -0.046653
loan_amnt                   -0.041961
loan_percent_income          0.090951
cb_person_cred_hist_length   0.068107
dtype: float64
```

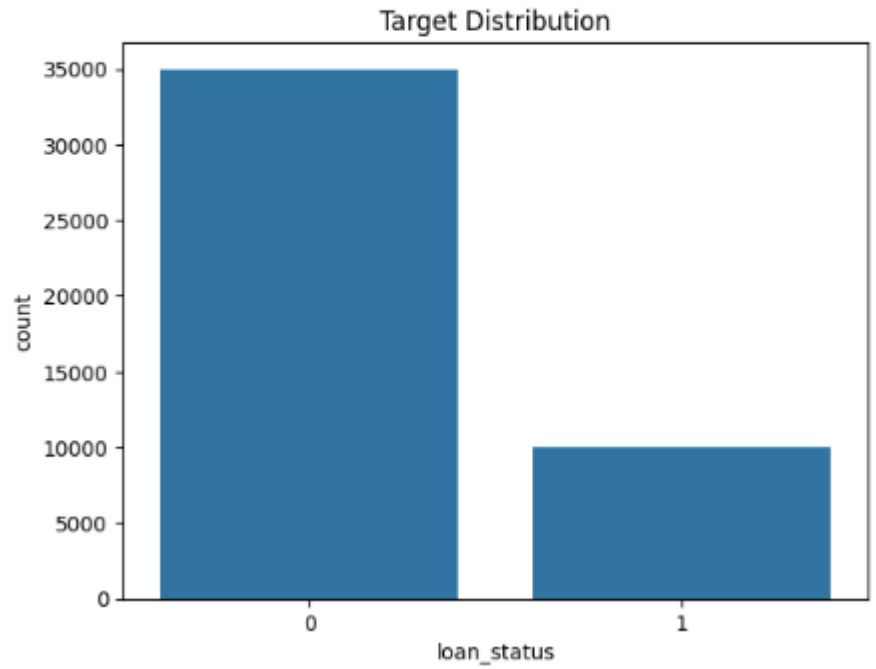
```
skewness = df[numeric_cols].skew()
print("Skewness of numerical features:\n", skewness)
```

```
Skewness of numerical features:
 person_age                   0.123188
person_income               -0.009936
person_emp_exp              -0.046653
loan_amnt                   -0.041961
loan_int_rate                0.207146
loan_percent_income          0.090951
cb_person_cred_hist_length   0.068107
credit_score                -0.532783
dtype: float64
```

6. Data Visualization:

Checking Imbalancement

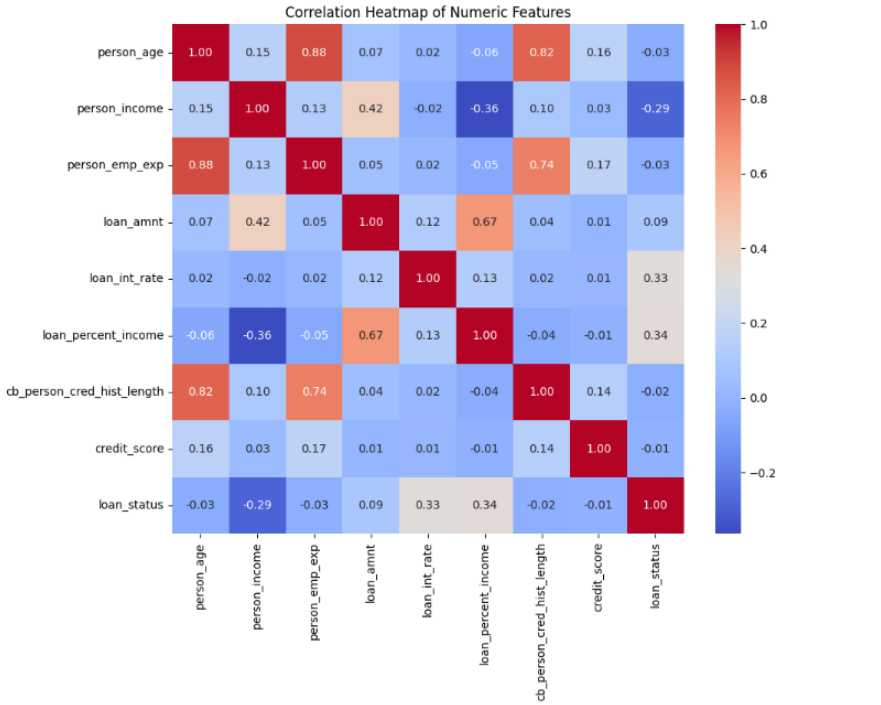
```
sns.countplot(x='loan_status', data=df)
plt.title('Target Distribution')
plt.show()
```



There is imbalancement in class

➢ **Heatmap**:

```
df2 = df.select_dtypes(exclude= ["object"])
plt.figure(figsize=(10, 8))
sns.heatmap(df2.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap of Numeric Features")
plt.show()
```



Conclusion

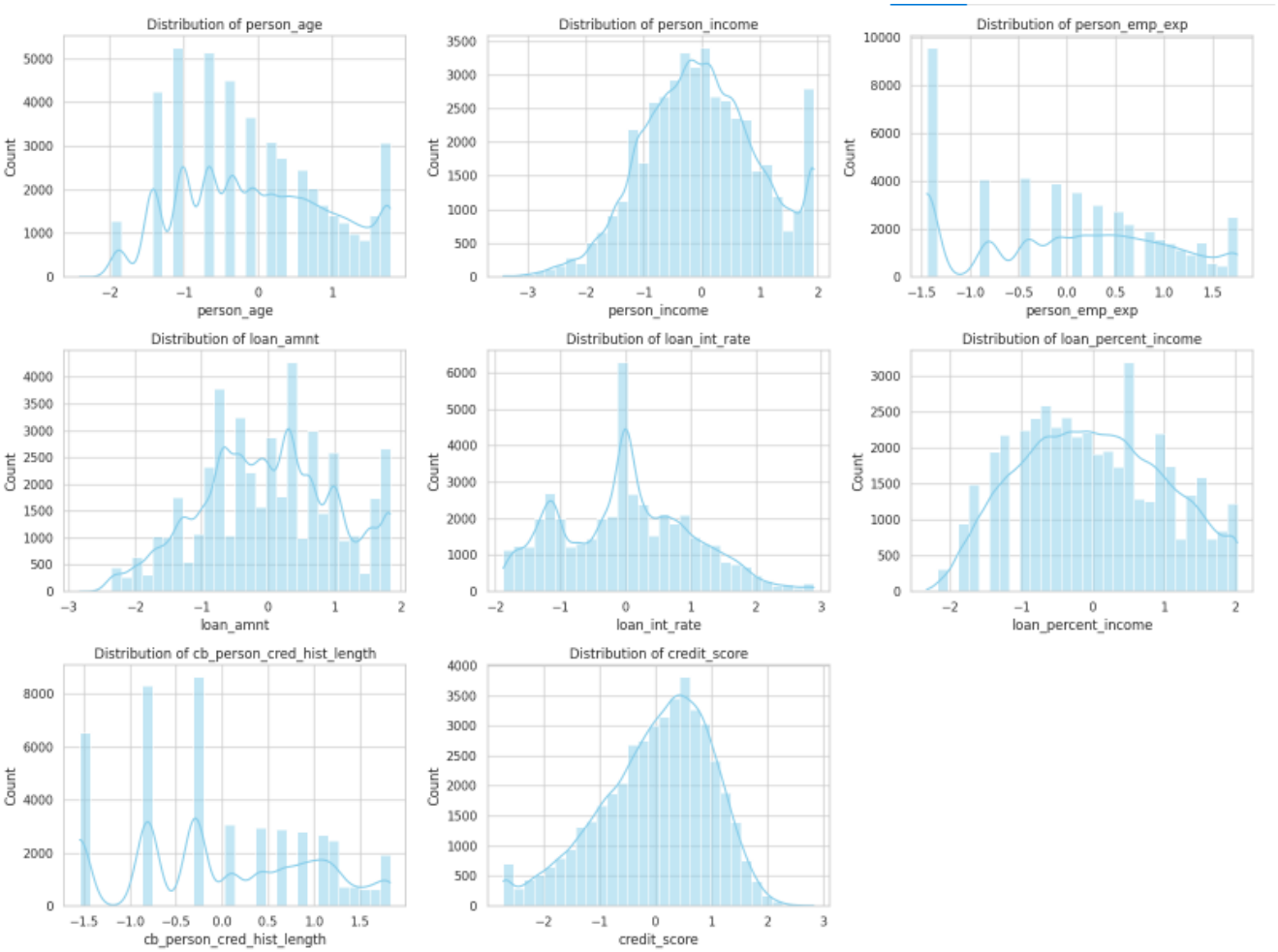
The analysis identifies the most influential factors affecting loan default and reveals key relationships within the dataset. Loan interest rate and loan-to-income ratio show the strongest positive correlations with loan_status, making them the most important predictors of default. Loan amount also contributes, though weakly.

Several features exhibit high inter-correlation—especially person_age, employment experience, and credit history length—indicating multicollinearity. These relationships suggest potential redundancy and the need for careful feature selection in models like Logistic Regression.

Finally, variables such as age and credit score display almost no correlation with loan_status, implying limited predictive ability on their own. Overall, the findings highlight which borrower and loan characteristics are most relevant for accurate loan default prediction**.**

Univariate Analysis (one variable at a time)

> Numeric Features Distribution, skew, outliers

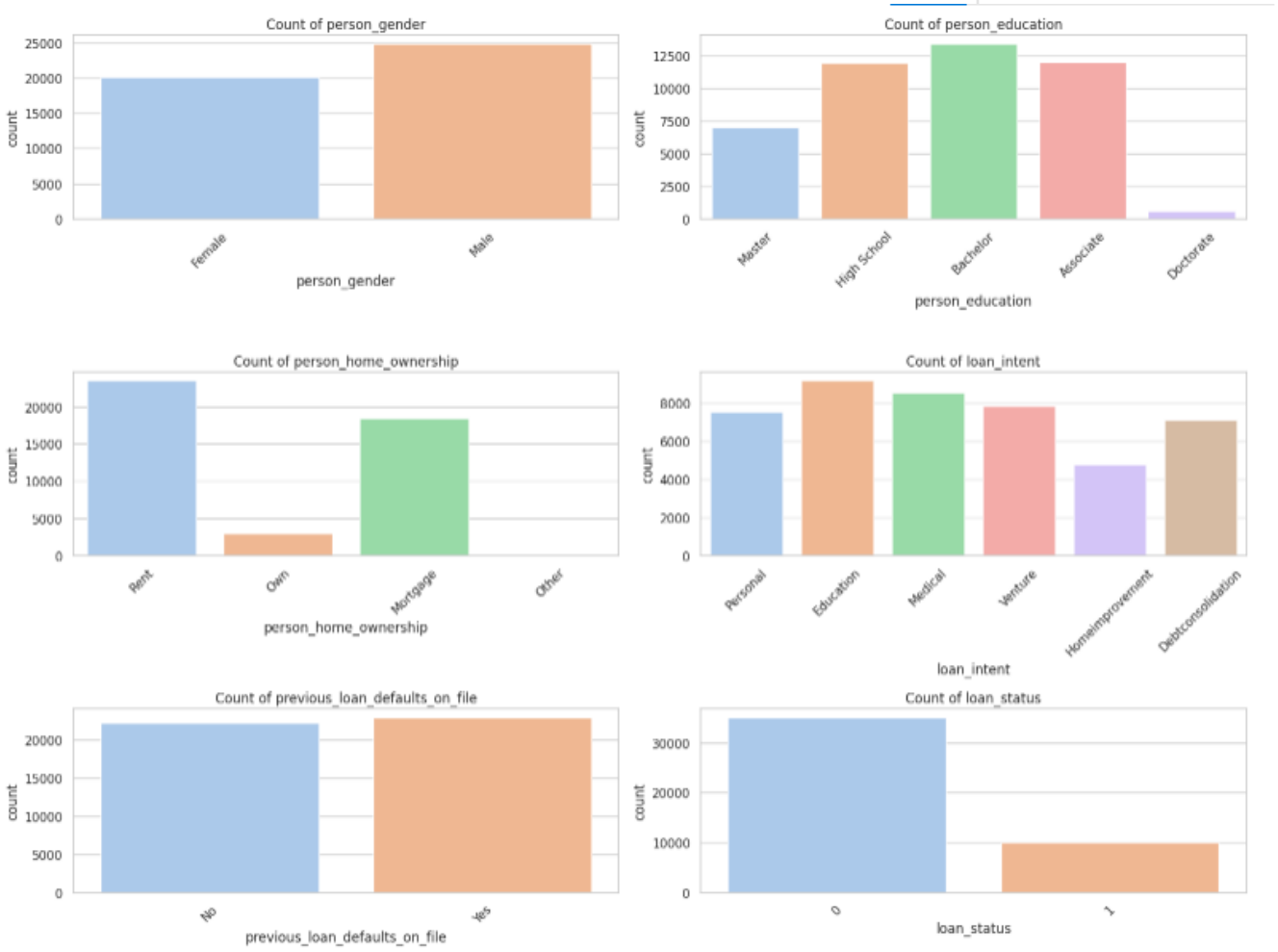


Several features show bimodal or skewed patterns. **loan_int_rate** and **person_income** display clear multimodal behavior, indicating distinct borrower groups. Features like **person_age** and **loan_amnt** are moderately skewed, while **credit_score** and **loan_percent_income** follow distributions closer to normal.
Overall, the dataset contains mixed distribution types, and preprocessing methods such as Yeo–Johnson transformation, scaling, or segmentation are necessary to stabilize variance and improve model performance.

Categorical Features

> Count distribution, imbalance

```
categorical_cols = ['person_gender', 'person_education', 'person_home_ownership',
                    'loan_intent', 'previous_loan_defaults_on_file', 'loan_status']

plt.figure(figsize=(16,12))
for i, col in enumerate(categorical_cols):
    plt.subplot(3,2,i+1)
    sns.countplot(x=df[col], palette='pastel')
    plt.title(f'Count of {col}')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

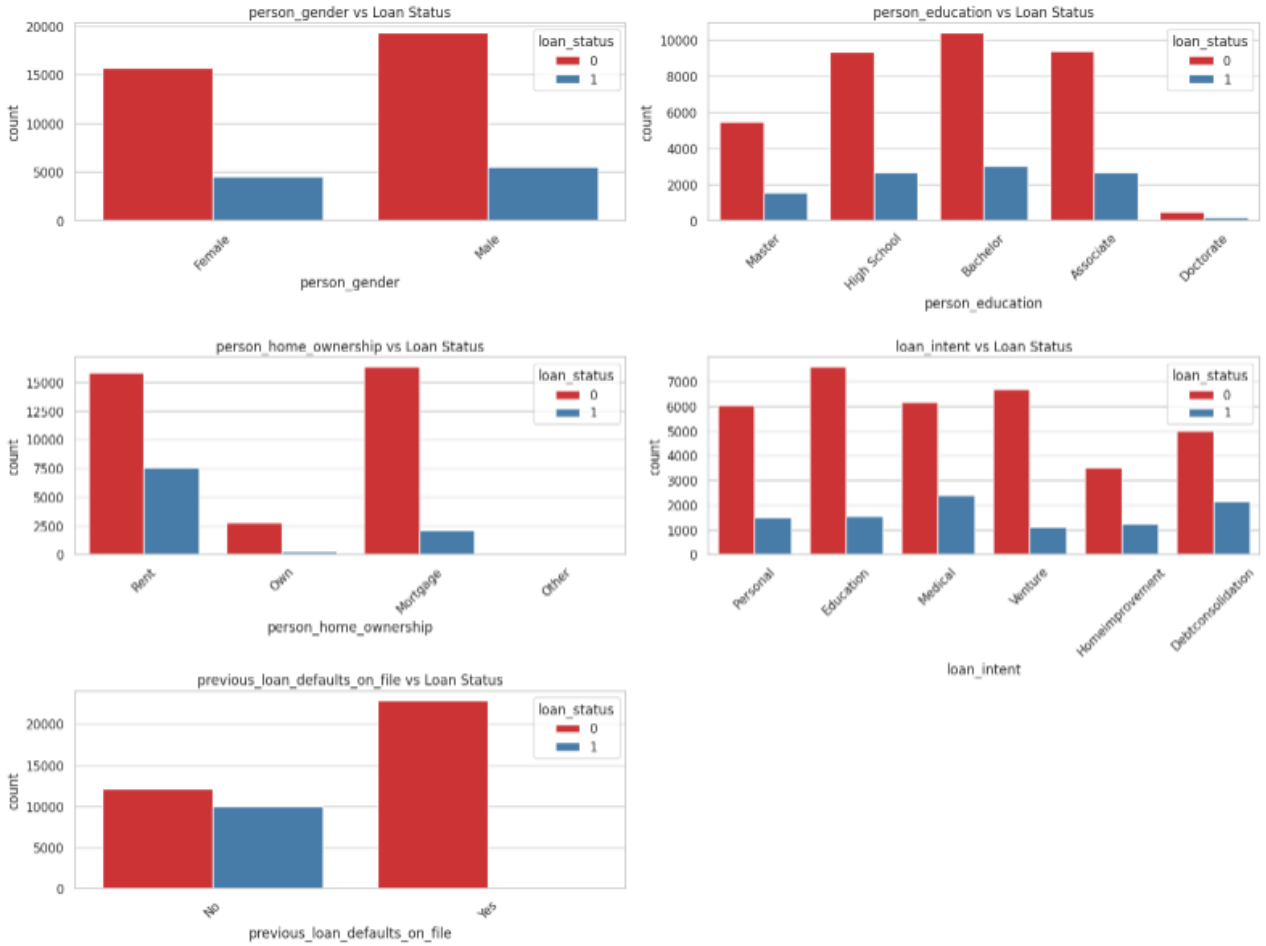


The dataset is characterized by an imbalanced target variable and a majority of applicants who rent their homes and lack previous defaults. These features provide a clear baseline for understanding the risk profiles present in the data before modeling begins.

Categorical vs Target

Countplot with hue

```
plt.figure(figsize=(16,12))
for i, col in enumerate(categorical_cols):
    if col != 'loan_status':
        plt.subplot(3,2,i+1)
        sns.countplot(x=col, hue='loan_status', data=df, palette='Set1')
        plt.title(f'{col} vs Loan Status')
        plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



The visualization indicates that previous default history and home ownership status are the most discriminative categorical features for predicting loan status. Features like gender and education level appear to have less influence on the final outcome.

7. Machine Learning Model Preparation

1.1 Encoding:

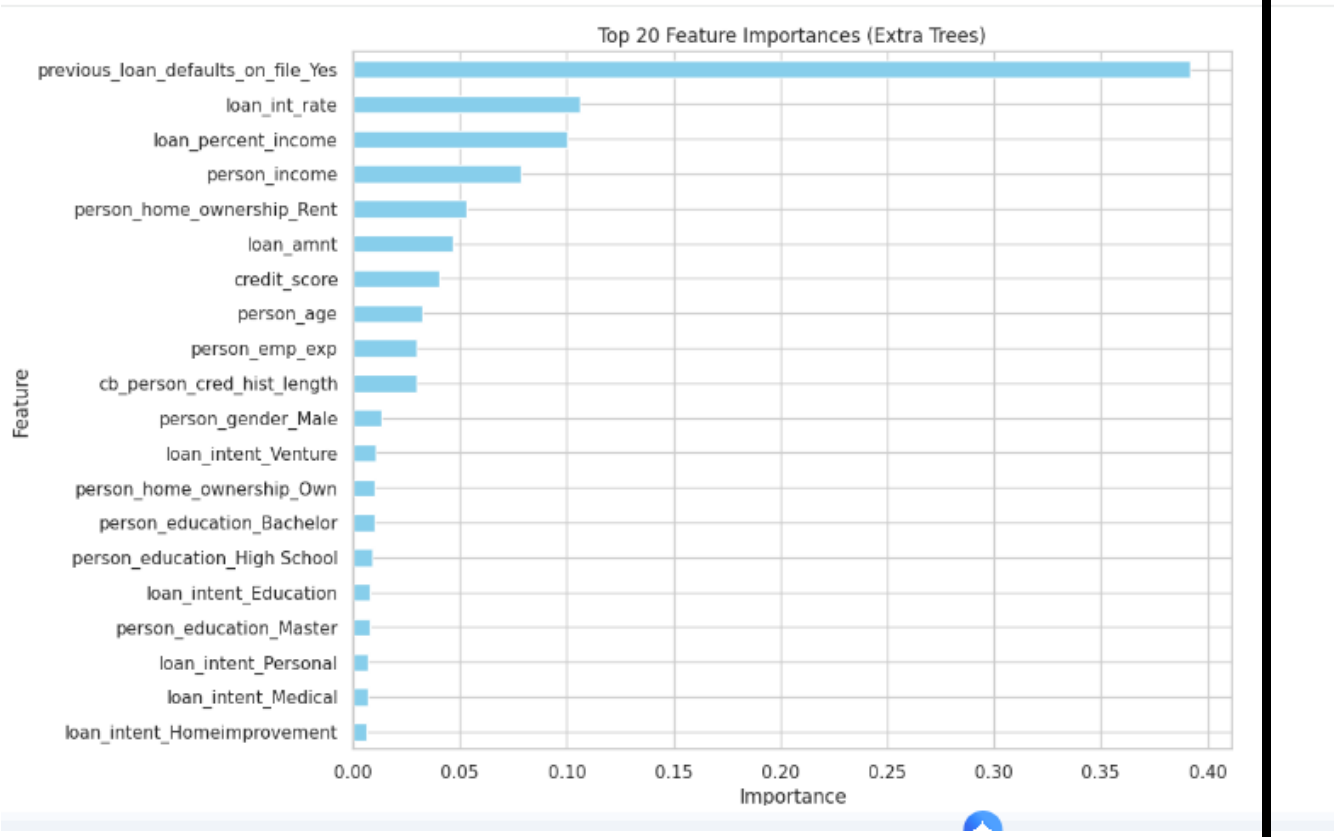
```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(drop='first', sparse_output=False)
cat_encoded = ohe.fit_transform(df[cat_cols])
cat_encoded_df = pd.DataFrame(cat_encoded, columns=ohe.get_feature_names_out(cat_cols))
df = df.drop(cat_cols, axis=1)
df = pd.concat([df.reset_index(drop=True), cat_encoded_df.reset_index(drop=True)], axis=1)
```

1.2 Feature Selection:

```
from sklearn.ensemble import ExtraTreesClassifier
et = ExtraTreesClassifier(n_estimators=100, random_state=42, class_weight='balanced')
et.fit(X, y)
importances = pd.Series(et.feature_importances_, index=X.columns)
top_features = importances.sort_values(ascending=False).head(20)
print(top_features)
```

```
previous_loan_defaults_on_file_Yes    0.391772
loan_int_rate                         0.106235
loan_percent_income                   0.100422
person_income                         0.078377
person_home_ownership_Rent            0.052863
loan_amnt                             0.046967
credit_score                          0.040144
person_age                            0.032186
person_emp_exp                        0.029787
cb_person_cred_hist_length            0.029614
person_gender_Male                    0.013518
loan_intent_Venture                   0.010862
person_home_ownership_Own             0.010223
person_education_Bachelor             0.009862
person_education_High School          0.008927
loan_intent_Education                 0.007877
person_education_Master               0.007877
loan_intent_Personal                  0.006886
loan_intent_Medical                   0.006764
loan_intent_Homeimprovement           0.006196
dtype: float64
```

1.3 Selected Features:



1.4 Train-Test -Split:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)
```

1.5 RandomForest – LogisticRegression - SVM:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, roc_auc_score, classification_report
```

```
lr = LogisticRegression(class_weight='balanced', max_iter=1000, random_state=42)
rf = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=42)
svm = SVC(class_weight='balanced', probability=True, random_state=42)
```

```
models = {'Logistic Regression': lr, 'Random Forest': rf, 'SVM': svm}

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

    # Predict
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:,1] if hasattr(model, "predict_proba") else None

    # Metrics
    print(f"--- {name} ---")
    print("Accuracy:", round(accuracy_score(y_test, y_pred), 4))
    print("F1 Score:", round(f1_score(y_test, y_pred), 4))
    print("ROC-AUC:", round(roc_auc_score(y_test, y_prob), 4) if y_prob is not None else "N/A")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("\n")
```

```
--- Logistic Regression ---
Accuracy: 0.8568
F1 Score: 0.7429
ROC-AUC: 0.9509
Confusion Matrix:
 [[5850 1151]
 [ 138 1862]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.84      0.90      7001
           1       0.62      0.93      0.74      2000

    accuracy                           0.86      9001
   macro avg       0.80      0.88      0.82      9001
weighted avg       0.90      0.86      0.87      9001
```

```
--- Random Forest ---
Accuracy: 0.9216
F1 Score: 0.8107
ROC-AUC: 0.9724
Confusion Matrix:
 [[6783  218]
 [ 488 1512]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      7001
           1       0.87      0.76      0.81      2000

    accuracy                           0.92      9001
   macro avg       0.90      0.86      0.88      9001
weighted avg       0.92      0.92      0.92      9001
```

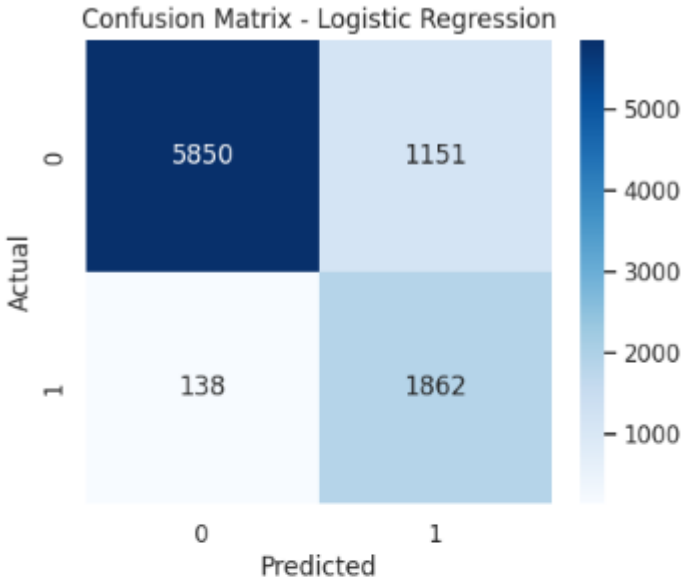
```
--- SVM ---
Accuracy: 0.8657
F1 Score: 0.7564
ROC-AUC: 0.9589
Confusion Matrix:
 [[5915 1086]
 [ 123 1877]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.84      0.91      7001
           1       0.63      0.94      0.76      2000

    accuracy                           0.87      9001
   macro avg       0.81      0.89      0.83      9001
weighted avg       0.90      0.87      0.87      9001
```

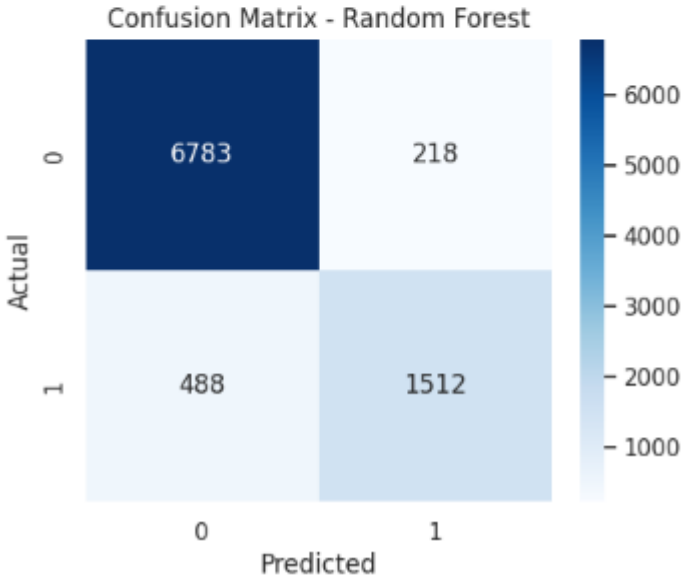
Random Forest is the recommended model due to its high precision and F1 score for predicting loan defaults, along with excellent overall accuracy and ROC-AUC.Logistic Regression and SVM favor recall over precision, which could lead to more false positives

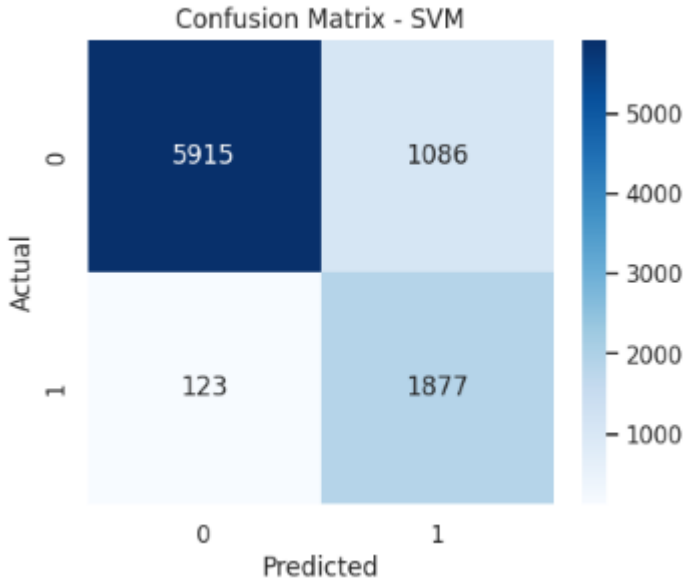
1.6 Model Comparison:

```
comparison_df
```

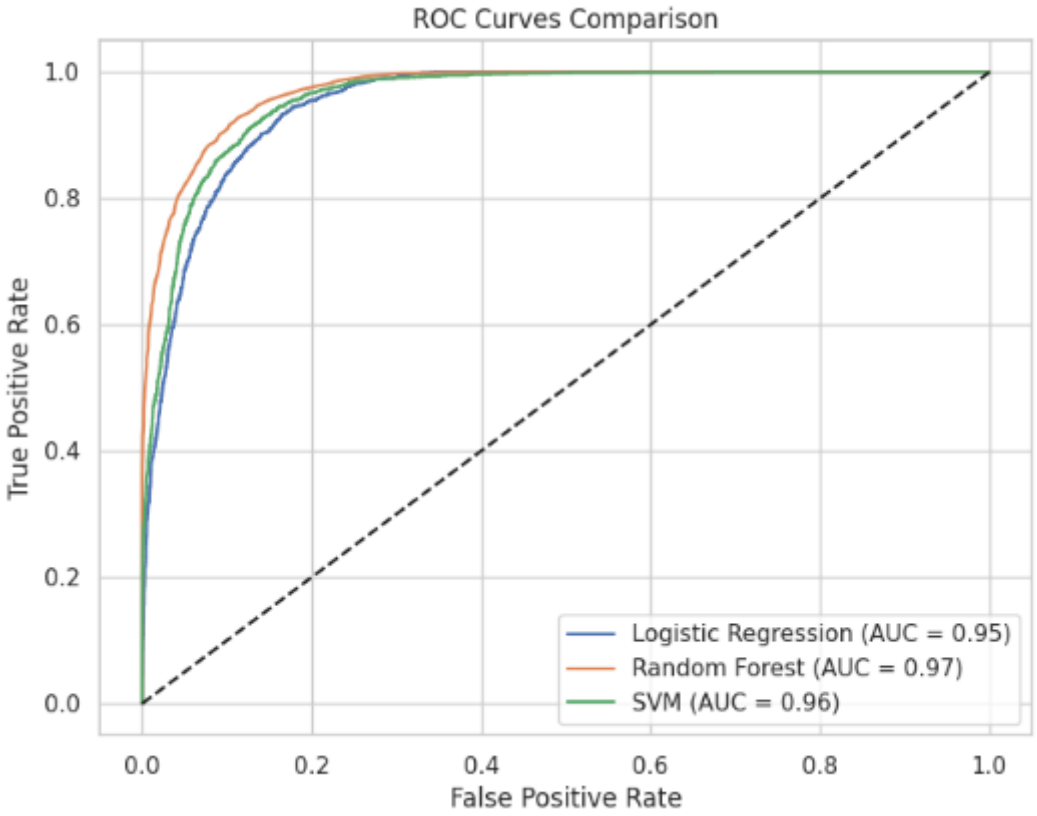
	Model	Accuracy	F1 (minority class)	F1 (overall)	ROC-AUC	Precision (minority class)	Precision (overall)
0	Logistic Regression	0.8568	0.7429	0.8657	0.9509	0.6180	0.8972
1	Random Forest	0.9216	0.8107	0.9195	0.9724	0.8740	0.9198
2	SVM	0.8657	0.7564	0.8738	0.9589	0.6335	0.9027



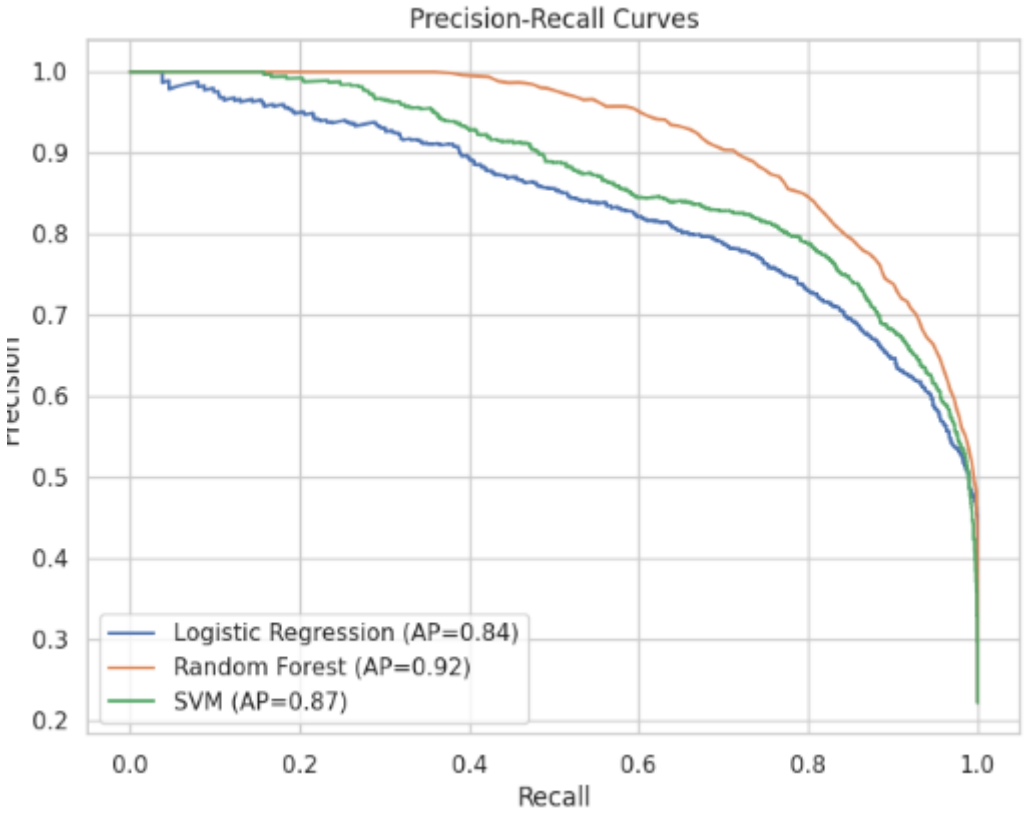




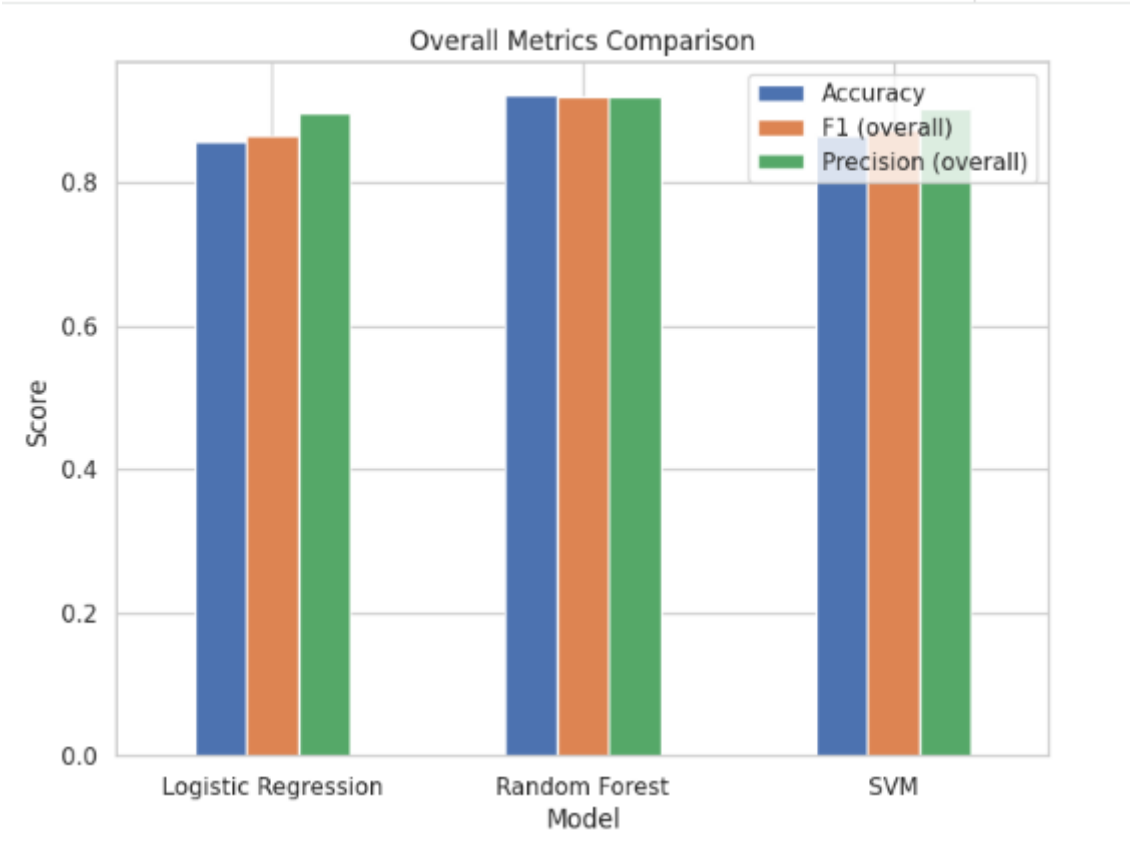
The Random Forest model provides the best balance of these metrics and is the recommended model for deployment, as it offers a strong F1 score 81.1% and the highest precision {87.4%} of the three tested algorithms.



All models are highly effective, but Random Forest is the marginally best-performing model based on the ROC curve



All three models perform well (AP > 0.80), but Random Forest is superior. It maintains higher precision across a wider range of recall values, making it the most robust choice for balancing correctly identifying loan defaults (recall) and minimizing false alarms (precision) in this task.



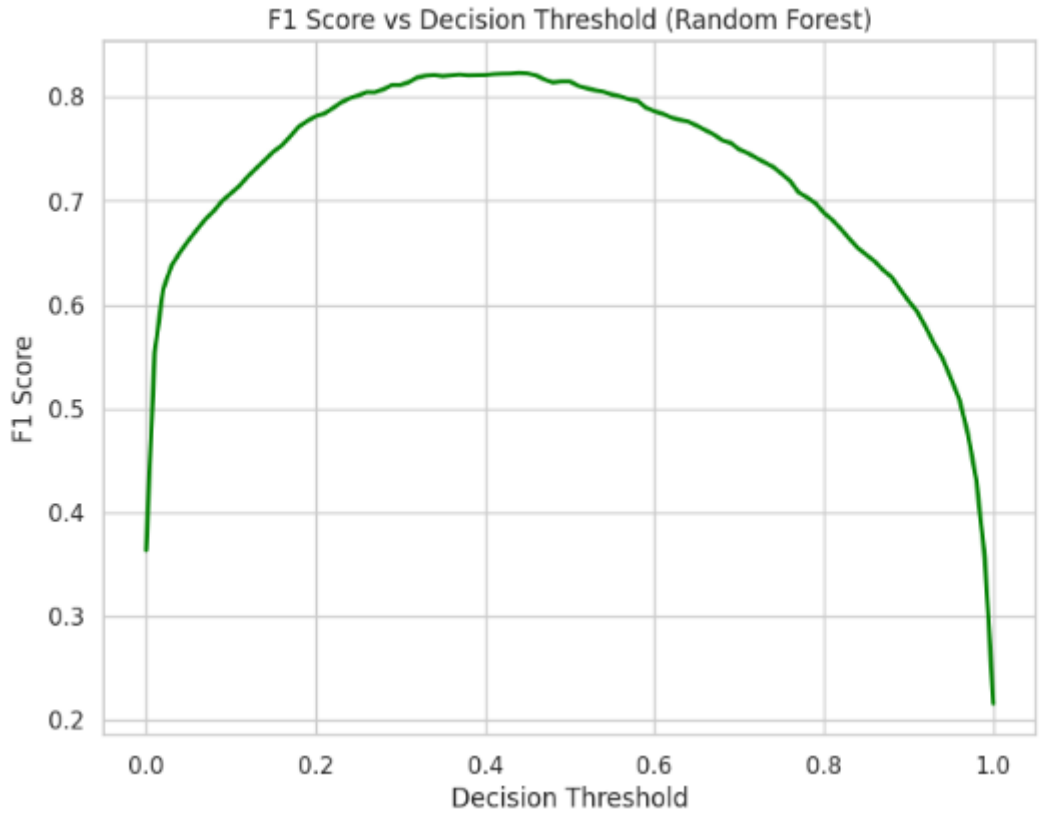
Based on the graph and chart random Forest Performed BEST throughout

1.7 Model Selection:

➢ Random Forest Classification:

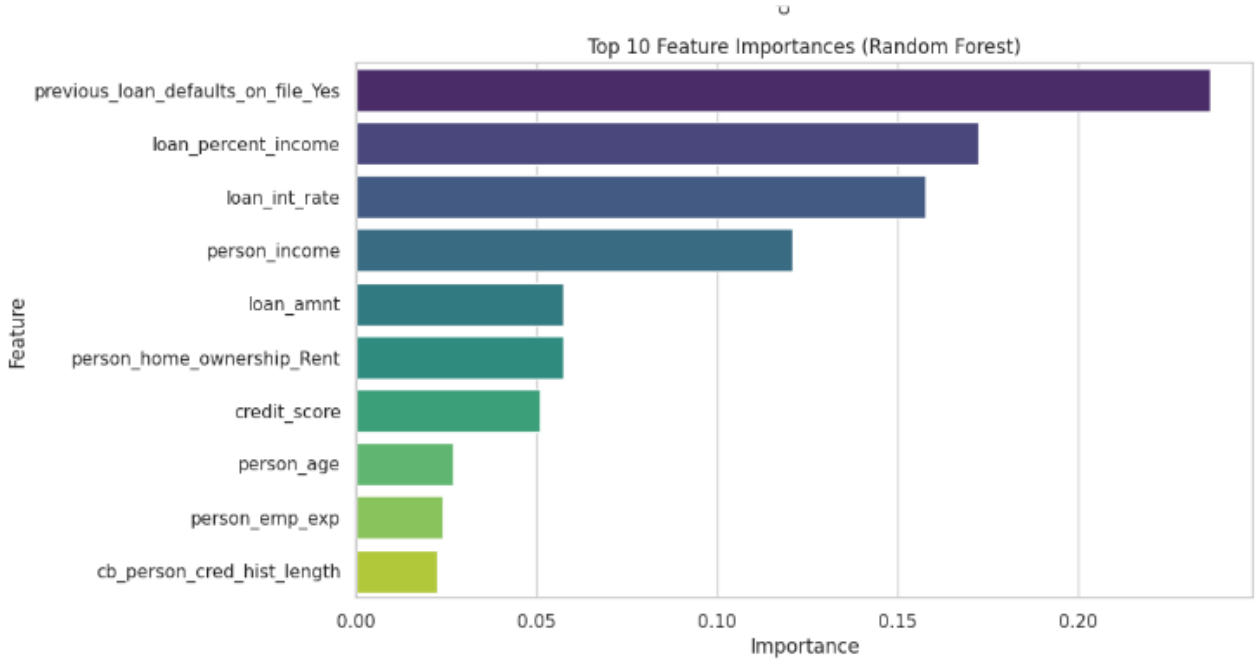
```
from sklearn.model_selection import GridSearchCV
rf_model = rf

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_rf = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    scoring='f1',
    cv=3,
    n_jobs=-1
)

grid_rf.fit(X_train, y_train)

print("Best parameters for Random Forest:", grid_rf.best_params_)

best_rf = grid_rf.best_estimator_
y_pred_best_rf = best_rf.predict(X_test)
y_prob_best_rf = best_rf.predict_proba(X_test)[:,1]

print("\nRandom Forest after GridSearchCV tuning:")
print("Accuracy:", round(accuracy_score(y_test, y_pred_best_rf), 4))
print("F1 Score:", round(f1_score(y_test, y_pred_best_rf), 4))
print("ROC-AUC:", round(roc_auc_score(y_test, y_prob_best_rf), 4))
```

```
Best parameters for Random Forest: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200}

Random Forest after GridSearchCV tuning:
Accuracy: 0.9215
F1 Score: 0.8235
ROC-AUC: 0.9722
```



Summary of Findings

- The Random Forest model achieves its best performance (F1 Score > 0.8) when the decision threshold is set to approximately 0.4.
- At this threshold, the model balances precision and recall most effectively for predicting loan defaults.
- Performance drops significantly if the threshold is set too low (~0.0) or too high (~1.0), causing a large imbalance between precision and recall.
- Recommendation: For the most accurate and balanced predictions in this loan detection project, set the classification threshold to 0.4.



10. Future Enhancement

- **Feature Engineering:** Introduce debt-to-income ratios, employment stability, or interaction terms.
- **Advanced Models:** Use Gradient Boosting, XGBoost, or Neural Networks for higher accuracy.
- **Hyperparameter Tuning:** Optimize parameters with Grid Search or Random Search.
- **Explainability:** Use SHAP or LIME to understand feature impact.
- **Dynamic Updates:** Regularly retrain models with new data to maintain predictive performance.

11. Conclusion

The analysis shows that **loan interest rate** and **loan-to-income percentage** are the strongest predictors of loan approval. Features like age and credit score have minimal impact. Machine learning models, especially Random Forest, effectively classify loan status. Future improvements can include feature engineering, advanced models, and explainability techniques to enhance predictive performance.

12. References

- Kaggle. *Loan Prediction Dataset*. https://www.kaggle.com/datasets/taweilo/loan-approval-classification-data
- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly Media.
- Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing.
- Brownlee, J. (2020). *Master Machine Learning Algorithms*. Machine Learning Mastery.