Automating Loan Approval Decisions

Enhancing Credit Assessment with Predictive Analytics

Loan Approval Classification Report

Project Title and Objective

Title: Loan Approval Classification
Objective: The goal of this project is to develop a classification model to predict loan approval outcomes (Approved or Not Approved) based on historical data. The model aims to automate and streamline the credit assessment process for financial institutions, reducing loan default risks, improving operational efficiency, and ensuring fair and consistent applicant evaluations. The primary focus is to enhance decision-making by leveraging predictive analytics to identify creditworthy applicants while minimizing financial losses due to high-risk approvals.

Dataset Overview

The dataset used is a simulated Loan Prediction dataset, reflective of real-world financial data for credit evaluation.

- **Rows:** 45,000
- **Columns:** 14
- **Target Variable:** loan_status (1 = Approved, 0 = Not Approved)

Feature Categories:

- **Personal Info:** person_gender, person_age, person_education, person_home_ownership, person_emp_exp
- **Financial Info:** person_income, loan_amnt, loan_percent_income, loan_int_rate
- **Credit Profile:** credit_score, cb_person_cred_hist_length, previous_loan_defaults_on_file
- **Loan Details:** loan_intent

Feature Types:

- **Numerical (9):** person_age, person_income, person_emp_exp, loan_amnt, loan_int_rate, loan_percent_income, cb_person_cred_hist_length, credit_score, loan_status
- **Categorical (5):** person_gender, person_education, person_home_ownership, loan_intent, previous_loan_defaults_on_file

Target Class Distribution:

The dataset contains 45,000 entries with a mean loan_status of 0.222, indicating approximately 22.2% of applications are approved (1) and 77.8% are not approved (0). This suggests an imbalanced dataset, with a higher proportion of non-approved loans.

Data Preprocessing Steps

Missing Values:

- The dataset has no missing values, as confirmed by df.info(), which shows 45,000 non-null entries for all 14 columns.

Outlier Handling:

- The summary statistics (df.describe()) revealed potential outliers in numerical features:
    - Filtering out records where person_age exceeded 80 years, as ages above this threshold are unrealistic for loan applicants.
    - Removing records where person_income exceeded the 99th percentile to eliminate extreme income values while retaining 99% of the data.
    - Filtering out records where person_emp_exp exceeded 60 years, assuming a maximum plausible career length.
    -

Categorical Encoding:

- Categorical variables (person_gender, person_education, person_home_ownership, loan_intent, previous_loan_defaults_on_file) encoded using techniques such as one-hot encoding or label encoding within the preprocessing pipeline.

Data Splitting:

- The dataset was split into training and testing sets using train_test_split, with a test size of 20% (9,000 test samples).

Class Imbalance Handling:

- The Synthetic Minority Oversampling Technique (SMOTE) was applied in the SVM pipeline to address the class imbalance in loan_status, ensuring the model is not biased toward the majority class (Not Approved).

Feature Engineering

Feature Transformations:

- Numerical features were likely standardized (e.g., using StandardScaler) within the preprocessing pipeline to ensure consistent scales across features.
- Categorical features were encoded using **OneHotEncoder** within the preprocessing pipeline. This transformation converts categorical variables into a sparse binary matrix format

Feature Selection:

- Feature selection was explicitly performed using SelectKBest with the f_classif scoring function, retaining all features (k='all'). This step ensures that only statistically significant features are considered, even if all are ultimately kept.

Model Training and Tuning

The notebook evaluates four classification algorithms using GridSearchCV to optimize hyperparameters, with a focus on recall as the primary evaluation metric due to the importance of identifying approved loans (positive class) in an imbalanced dataset.

Algorithms Used:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)

Parameter Grids Explored:

A comprehensive hyperparameter search was conducted for each classification model using GridSearchCV, exploring the following ranges:

- **Logistic Regression**
    - classifier__C: [0.1, 1, 10, 100]
    - classifier__solver: ['saga', 'liblinear']
    -
- **Decision Tree**
    - classifier__max_depth: [None, 10, 20, 30, 40]
    - classifier__min_samples_split: [2, 5, 10, 20]
    -
- **Random Forest**
    - classifier__n_estimators: [100, 200, 300]
    - classifier__max_depth: [None, 10, 20]
    -
- **Support Vector Machine (SVM)**
    - classifier__C: [0.1, 1, 10, 100]
    - classifier__kernel: ['linear', 'rbf']
    - classifier__gamma: ['scale', 'auto', 0.01, 0.1]

-

Evaluation Metric:

- Recall was used as the primary metric to evaluate model performance, emphasizing the ability to correctly identify approved loans.

Training Pipeline:

- A Pipeline was used, incorporating preprocessing, SMOTE for class imbalance, and the classifier. GridSearchCV was applied to tune hyperparameters.

Model Performance Comparison

Model	Best Parameters	Test Recall
Logistic Regression	{'classifier__C': 0.1, 'classifier__solver': 'liblinear'}	0.9134
Decision Tree	{'classifier__max_depth': 10, 'classifier__min_samples_split': 2}	0.8821
Random Forest	{'classifier__max_depth': 10, 'classifier__n_estimators': 100}	0.9035
SVM	{'classifier__C': 0.1, 'classifier__gamma': 'scale', 'classifier__kernel': 'linear'}	0.9652

Key Observations:

- SVM achieved the highest test recall (0.9652), indicating it is the most effective at identifying approved loans.
- Decision Tree and Random Forest have balanced performance with higher accuracy (0.89) and better precision for the positive class compared to SVM.
- Logistic Regression offers a good balance between recall (0.9134) and accuracy (0.86).
- SVM has the lowest precision (0.53) for the positive class, suggesting a higher false positive rate.

Best Model Deep Dive

Best Model: Support Vector Machine (SVM)

Hyperparameters:

- **C:** 0.1
- **gamma:** 'scale'
- **kernel:** 'linear'
- **probability:** True

Performance Metrics (Test Set):

- **Accuracy:** 0.80
- **Recall Score:** 0.9652

Why SVM Was Chosen:

- The SVM model was selected due to its superior recall score (0.9652), which aligns with the business objective of maximizing the identification of creditworthy applicants.
- The linear kernel suggests that the data is largely linearly separable after preprocessing, making SVM an effective choice.
- The use of SMOTE in the pipeline helps mitigate class imbalance.
- The model was saved as best_svm_model_pipeline.joblib for deployment.

Insights from Data:

- The imbalanced target distribution underscores the importance of SMOTE.
- Further cleaning of features like person_age and person_emp_exp may improve model robustness.

Deployment Considerations

Saving the SVM Model

To ensure the re-usability and efficiency of the trained Support Vector Machine (SVM) model, it was saved using the joblib library. The model, encapsulated in a pipeline including preprocessing, SMOTE, and the SVM classifier (SVC with C=0.1, gamma='scale', kernel='linear', probability=True), was serialized and saved as best_svm_model_pipeline.joblib. The joblib library is widely used for serializing and deserializing Python objects, particularly machine learning models, enabling seamless deployment and reuse without retraining. This step confirms the model’s readiness for integration into a production environment for real-time loan approval decisions.

Conclusion

Conclusion:
The Loan Approval Classification project successfully developed a predictive model to automate credit assessment. The SVM model, with a recall of 0.9652, emerged as the best performer for identifying approved loans. While the model sacrifices precision , its high recall ensures that most eligible applicants are approved.