Loan Approval Project:

- This repository contains an attempt to apply classification algorithms to the Loan Approval Prediction dataset from the Kaggle Tabular Playground Challenge (Loan Approval Prediction | Kaggle).

Overview:

- The Kaggle challenge aims to predict loan approval using features like age, income, and credit history. Using logistic regression, the data was preprocessed with encoding, normalization, and data splitting. The model achieved 90% accuracy, with a precision of 67.04%.

Summary of work done:

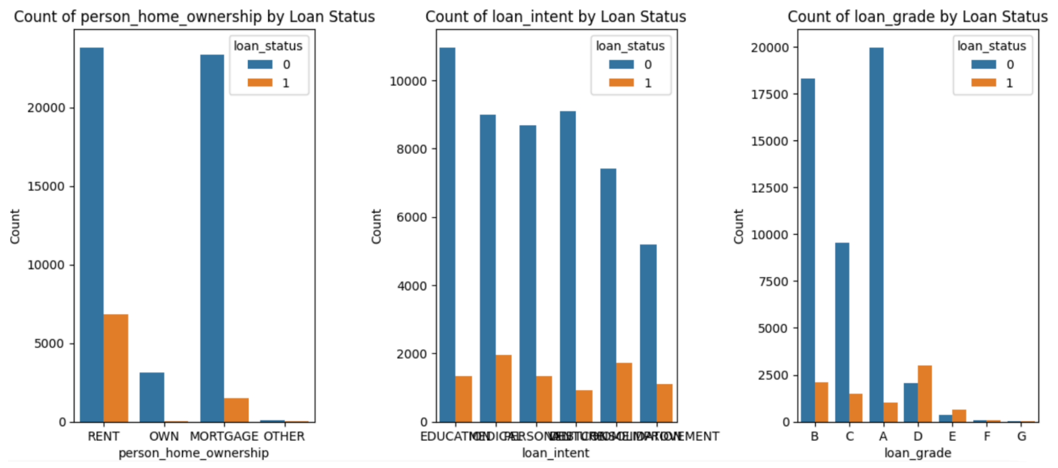
Data
Type: Tabular data
The dataset consists of 50,827 loan applications with 27 features, including numerical data (e.g., age, income, loan amount, credit history) and one-hot encoded categorical data (e.g., homeownership status, loan intent, loan grade, and default history). The target variable, "loan_status," is binary (1 for approved, 0 for denied). The data is split into 70% training, 15% validation, and 15% test sets. The goal is to predict loan approval status based on applicant information.

Preprocessing / Clean up:

The data was preprocessed by handling missing values, encoding categorical variables, and normalizing numerical features. Categorical features like person_home_ownership, loan_intent, loan_grade, and cb_person_default_on_file were one-hot encoded. The target variable loan_status was converted to binary values (0 for "No" and 1 for "Yes"). Redundant columns, such as the unique identifier id, were dropped to avoid introducing unnecessary noise into the model.

Data Visualization:

Visualizations showed that most applicants were aged 20-40, with debt consolidation being the most common loan intent. Income had significant variability, with some high-income outliers. Correlations between features, like loan_amnt and loan_int_rate, indicated that larger loans generally have higher interest rates. These insights informed feature selection and model development.



Problem Formulation:

The task aims to predict loan approval status (loan_status) using features like person_age, person_income, and loan_amnt. Models tested include Logistic Regression, Random Forest, SVM, and XGBoost. Logistic Regression was chosen for its simplicity, while Random Forest and XGBoost were tested for better handling of complex relationships. Hyperparameters like regularization for Logistic Regression and n_estimators for Random Forest were tuned. Logistic Regression provided the best balance between accuracy and simplicity.

Training:

I trained the models using Python and libraries like scikit-learn and XGBoost. The training process was done using Google Colab and a MacBook with an M1 processor. Logistic Regression was trained on the processed data. The training process involved fitting the model on the training set and then validating its performance on a separate validation set.

Performance Comparison:

The model's key performance metrics include accuracy (90.4%), precision (67.04%), recall (32.02%), F1-score (43.34%), and ROC AUC (84.54%). The ROC curve shows the model's ability to distinguish between classes, with a high AUC indicating good performance, though recall could be improved.

Conclusions:

The logistic regression model performed reasonably well, achieving good accuracy and AUC, but recall was relatively low, indicating it struggled with predicting the minority class (approved loans). Future improvements could focus on enhancing recall through techniques like resampling or using more complex models like random forests or XGBoost.

Future Work:

Next, I would experiment with more complex models like Random Forests or XGBoost to improve performance, especially in terms of recall for the minority class. Additionally, I would try hyperparameter tuning, resampling techniques (such as SMOTE), or feature engineering to enhance the model's ability to predict loan approvals. Future studies could explore deep learning models like neural networks, compare different ensemble methods, or investigate the impact of additional features such as applicant's credit score or external economic factors.

How to reproduce results:

To reproduce the results, install necessary libraries (pandas, numpy, scikit-learn, xgboost) and load the dataset. Preprocess the data by handling missing values, encoding categorical features, and scaling numerical ones. Train a model (e.g., Logistic Regression, Random Forest) on the training set, evaluate it on the validation set, and apply it to test data. Google Colab is recommended for cloud-based computation.

Overview of files in repository:

The repository contains the following files: Loan Approval Project.ipynb, the main notebook where all steps of data preprocessing, model training, evaluation, and submission generation are performed; train.csv, the training dataset used for model training; and test.csv, the test dataset used for model evaluation and generating Kaggle submissions. All tasks are handled within the single notebook.

Software Setup:

The required packages for this project include pandas, numpy, scikit-learn, matplotlib, seaborn, xgboost, and imbalanced-learn. These can be installed using the following command in a terminal or Colab notebook: pip install pandas numpy scikit-learn matplotlib seaborn xgboost imbalanced-learn. Google Colab typically has most of these packages pre-installed, but they can be added if needed

Data:

The data for this project can be downloaded from the Kaggle Loan Approval Prediction Challenge page, where you'll find train.csv and test.csv files. After downloading, preprocessing steps such as handling missing values (e.g., filling or dropping them) and encoding categorical variables (using one-hot encoding) are necessary to prepare the data. You can also separate the features and target variable, ensuring that the test data undergoes similar preprocessing. These steps will make the data ready for training machine learning models and generating predictions.

Training:

To train the model, load and preprocess the training data from train.csv. Split it into training and validation sets using train_test_split. Initialize and fit your chosen model (e.g., Logistic Regression) on the training data. Evaluate the model using the validation set with metrics like accuracy, precision, and recall. Optionally, tune hyperparameters for better performance.

Performance Evaluation:

To evaluate the model's performance, use the trained model to make predictions on the validation set. Calculate key metrics such as accuracy, precision, recall, F1-score, and ROC AUC. You can also generate a confusion matrix to visualize true positive, true negative, false positive, and false negative counts. Additionally, use classification reports to assess model performance for each

Citations:

https://www.kaggle.com/competitions/playground-series-s4e10/data