Loan Approval Prediction
DATA1030 - Fall 2024

Qingyang Cheng (GitHub repository here)
Brown University - Data Science Institute

December 2024

1 Introduction

In this project, a machine learning pipeline is developed for **loan approval prediction**, a binary classification task. The goal is to predict whether a new loan application, given its information, should be approved or not. The dataset used throughout this project can be found in the `data` folder. It was retrieved from Kaggle in October 2024, but since the dataset got updated by its author after the retrieval, it is no longer available there now. Nonetheless, sufficiently detailed information about it can still be accessed via the updated dataset.

2 Exploratory Data Analysis

The raw dataset contains 32,581 rows and 12 columns. After a preliminary inspection, 7 unreasonable data points are identified. Each of them has either `person_age` greater than 100 or `person_emp_length` (number of years for which the person is employed) greater than 100. These data points are discarded, leaving the dataset with 32,574 rows.

The target variable `loan_status` is represented by a binary indicator — `1` for an approved loan (*class 1*) and `0` for a rejected loan (*class 0*). *Class 1* samples make up 7,101 entries, which is approximately 21.8% of the entire dataset.

For the remaining 11 features, the correlation between each feature and the target variable can be assessed by examining the in-class distribution of that feature. If the distributions of the same feature differ across classes, the feature is likely to be an influential predictor. Figure 1 highlights four features that exhibit this pattern.

Out of the dataset, 3,942 data points (12.10%) contain missing values, which occur in two features:

- `loan_int_rate`: 3,115 data points (9.56%)
- `person_emp_length`: 895 data points (2.75%)

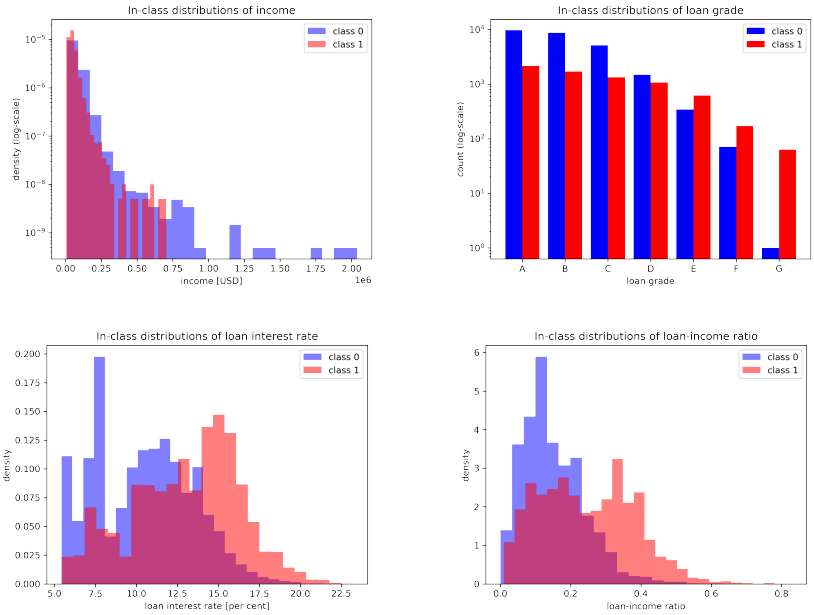

Figure 1: Each plot shows two overlapping distributions of the same feature for the two classes. For each feature, the distributions differ noticeably. Therefore, these four features are likely to be influential in predicting the target variable.

It is straightforward to observe that $3115 + 895 - 3942 = 68$ rows have missing values in both features. Since both features are numerical, additional steps for feature reduction are required before applying any model that cannot handle missing data. This leads to partitioning the dataset into four groups, one of which contains only 68 data points with both features missing. Given the small size of this group, it is unsuitable for training, so an alternative solution should be considered.

Recall that feature `person_emp_length` has only 895 missing values, making up only 2.75% of the entire dataset. Additionally, it is not one of the four potentially predictive features mentioned earlier. Therefore, it is reasonable to drop the 895 rows where `person_emp_length` is missing. As a result, the dataset now contains 31,679 rows, with 3,047 (9.6%) having `loan_int_rate` missing.

3 Methods

The machine learning algorithms explored in this project are *XGBoost*, *logistic regression*, *support vector machine*, and *k-nearest neighbors*. Before diving into details, some essential data preparation steps should be considered.

3.1 Feature Engineering

To preprocess the dataset, note that 7 features are numerical, 3 are categorical, and 1 is ordinal. One of the categorical features is special because it is binary, with values `Y` and `N` representing "yes" and "no" respectively. One-hot encoding a binary feature produces two correlated features with a correlation coefficient of -1, leading to redundancy. A better approach is to directly map this binary feature to the binary space $\{0, 1\}$, where `Y` is mapped to 1 and `N` to 0. It will pass through the preprocessor without any modification.

3.2 Splitting and Preprocessing

The splitting strategy used is *stratified shuffle split*. Although the target variable is not highly imbalanced, stratifying by the target is always beneficial.

For preprocessing, a standard scaler is applied to each numerical feature, a one-hot encoder to each categorical feature (except the binary one), and an ordinal encoder to the ordinal feature.

3.3 Hyperparameter Tuning

The tuning process consists of 5 iterations, each using a different random state, denoted as $42k$ for $k = 0, 1, 2, 3, 4$.

Among the four algorithms mentioned earlier, XGBoost can automatically handle missing values, so no feature reduction is necessary. With a dataset of 31,679 rows, the split ratio is set to $0.8 : 0.1 : 0.1$, and the preprocessor transforms the dataset into 19 features.

For the other three algorithms, recall that missing values now occur in only one feature. As a result, feature reduction produces two groups. The larger group, which contains 28,632 rows and has no missing values, is split with a ratio of $0.8 : 0.1 : 0.1$ and preprocessed into 19 features. The smaller group, which contains 3,047 rows and has its entire feature `loan_int_rate` missing, is split with a ratio of $0.6 : 0.2 : 0.2$ and preprocessed into 18 features.

The fixed parameters for each algorithm are listed below, while the tuned parameters and their optimal values are presented in Section 4.

3.3.1 XGBoost

- `early_stopoping_rounds = 50`
- `missing = np.nan`
- `colsample_bytree = 0.9`
- `subsample = 0.8`
- `random_state = 42`
- `n_jobs = -1`

3.3.2 Logistic Regressor (Log-Reg)

- `penalty = 'elasticnet'`
- `solver = 'saga'`
- `max_iter = 5000`
- `random_state = 42`
- `n_jobs = -1`

3.3.3 Support Vector Classifier (SVC)

- `kernel = 'rbf'`

3.3.4 K-Nearest Neighbors (KNN)

- `n_jobs = -1`

4 Results

This section presents the optimal parameter set for each algorithm and provides an in-depth analysis of the best-performing one.

4.1 Optimal Parameters

The tuned parameters of each algorithm are listed below, with the optimal values highlighted in red. Recall that feature reduction results in two groups, so for the three algorithms that require feature reduction, separate models are tuned on each group.

4.1.1 XGBoost

- `max_depth` $\in \{2, 5, 10\}$
- `reg_alpha` $\in \{0.1, 0.5, 1\}$
- `reg_lambda` $\in \{0.1, 0.5, 1\}$

4.1.2 Log-Reg

On the larger group (28,632 points; 19 features):

- `C` $\in \{0.1, 1, 10\}$
- `l1_ratio` $\in \{0.1, 0.5, 0.9\}$

On the smaller group (3,047 points; 18 features):

- `C` $\in \{0.1, 1, 10\}$
- `l1_ratio` $\in \{0.1, 0.5, 0.9\}$

4.1.3 SVC

On both groups:

- `C` $\in \{0.1, 1, 10\}$
- `gamma` $\in \{0.01, 0.1, 1\}$

4.1.4 KNN

On the larger group (28,632 points; 19 features):

- `n_neighbors` $\in \{5, 10, 20\}$

On the smaller group (3,047 points; 18 features):

- `n_neighbors` $\in \{5, 10, 20\}$

4.2 Overall Performance

	XGBoost	Log-Reg	SVC	KNN
Mean of accuracy	0.936	0.855	0.916	0.894
Standard deviation	< 0.001	0.004	0.003	0.004
Runtime	38.0 sec	$\sim$ 4 min	$\sim$ 20 min	5.5 sec

Table 1: Overall performance of each algorithm, measured by the mean and standard deviation of accuracy scores across the 5 iterations. The runtime row serves as a reference, as it varies with each run.

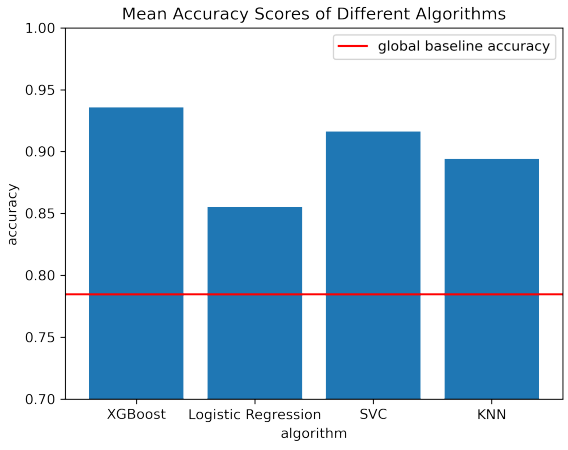

Figure 2: Mean accuracies of the four algorithms and the global baseline 0.785

The overall performance of each algorithm is summarized in Table 1. All four models exhibit high stability, with very low standard deviations of accuracy scores. Figure 2 displays the mean accuracy score for each algorithm, compared to a global baseline accuracy 0.785 calculated over the entire dataset.

Clearly, *XGBoost* is the best algorithm for this dataset, with the highest mean accuracy and the lowest standard deviation. Its runtime is also reasonable.

4.3 In-Depth Analysis of XGBoost

To conduct a deeper analysis of XGBoost, it is retrained on the entire dataset, with various types of evaluation metrics and feature importances computed.

4.3.1 XGBoost Retrained

Since the optimal parameter set of XGBoost has been determined, the dataset is split with a ratio of $0.9 : 0.05 : 0.05$ for retraining. The test set now contains 1,584 data points, with a baseline accuracy of 0.785. The resulting test accuracy is 0.938.

4.3.2 More Evaluation Metrics

Besides accuracy, various F-beta scores are computed to evaluate the model's precision and recall. Figure 3 displays the confusion matrix measured on the test set.

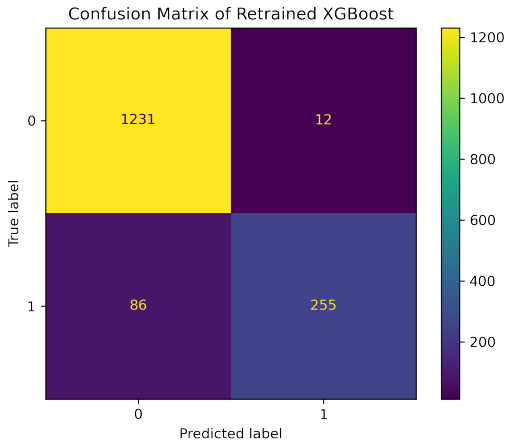

Figure 3: Confusion matrix of the retrained XGBoost

When $\beta = 1$, the model's F-1 score is 0.839, marking a significant improvement over the baseline score of 0.354.

In the context of this dataset, the cost of approving a loan for an unqualified applicant is higher than the cost of rejecting a loan for a qualified applicant.

Therefore, the model's *precision* should be prioritized over its *recall*. From the formula

$$F_\beta = \left(1 + \beta^2\right) \frac{PR}{\beta^2 P + R},$$

it is clear that choosing a value of $\beta < 1$ meets this objective. With $\beta = 0.4$, the resulting F-0.4 score is 0.920, demonstrating an even greater improvement over the baseline score of 0.241. Therefore, the XGBoost model aligns well with the contextual requirements.

4.4 Feature Importances

Rank	Permutation	Total gain	SHAP value
1	loan_grade	loan_percent_income	income
2	loan_percent_income	loan_grade	loan_grade
3	income	income	loan_percent_income

Table 2: Features `loan_grade`, `loan_percent_income`, and `income` are the top 3 important features for all three global metrics, regardless of their exact rank

Figure 4: Force plots of data points `42`, `250`, and `1437` of the test set

To interpret the model, a few types of feature importances are computed. Global feature importance metrics include permutation, total gain, and SHAP value. A few local SHAP values are also computed.

Impressively, as shown by Table 2, all of the three global metrics yield the same top 3 most important features, and they are among the four potentially influential predictors discussed in Section 2. However, the other one feature `loan_int_rate` is not even in top 5 for any of the metrics, which is a bit surprising.

Figure 4 presents the force plots of three specifically indexed data points in the test set. Some of the top 3 features of global importance can be seen in the force plots among those strongest pushing forces.

5 Outlook

As mentioned in Section 1, the loan approval dataset has been updated by its author. It contains more features and is available for a new regression task: credit score prediction. With the updated dataset, better models can be trained, which can provide more insights into understanding financial risk factors, A neural network model can be considered for a potentially stronger predictive power if the issue of missing values is eliminated.

6 References

Source of dataset:
https://www.kaggle.com/datasets/taweilo/loan-approval-classification-data