Loan Default Prediction

for financial loan services

Jing Xu, Brown DSI, 12/11/2024

GitHub:https://github.com/Jing-Xu1223/DATA1030-Project

Recap:

Introduction:

- Importance: Decrease payment defaults and ensure that all individuals are paying back their loans as expected.
- Employ ML models to predict which individuals are at the highest risk of defaulting on their loans, based on their personal demographics. Thus, proper interventions can be effectively deployed to the right audience.

Dataset Collection:

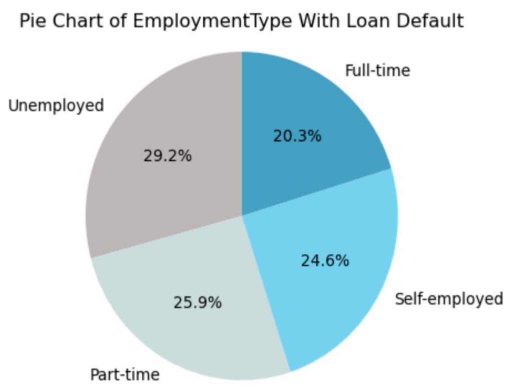
- Source: Kaggle:Loan Default Prediction
- This dataset is collected by Coursera Project Network:Loan Default Prediction Coding Challenge, which includes a sample of individuals who took financial loans in 2021.

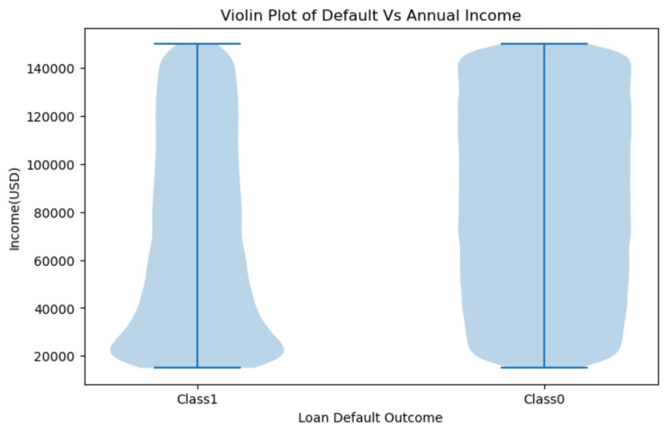
Dataset Overview:

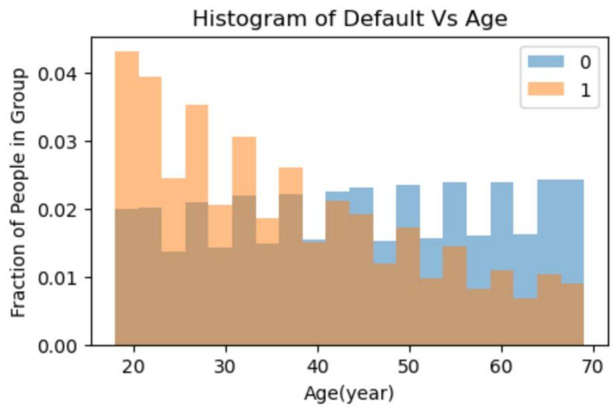
Binary Classification

- Highly Imbalanced: nearly 90% in class 0(no loan default), 10% in class 1(cause loan default)
- 16 features& 1 target "default"
- Large dataset(255347 rows), No Missing Value, IID Dataset

Recap: EDA Highlights







Relationship Between Target: Loan Default

- **Employment Type:** It doesn't seem quite a difference for customers having different employment types.
- **Annual Income:** Indicates individual with lower incomes are more likely to encounter loan default.
- **Age:** The likelihood of a loan default is four times higher for young individuals in their 20s than for older individuals in their 60s. Therefore, age might play a significant role in feature importance.

Preprocessor:

OneHot Encoder: 6 Categorical Features

- EmploymentType, MaritalStatus, LoanPurpose, HasMortgage, HasDependents, HasCoSigner

Ordinal Encoder: 1 Ordinal Feature

- Education(Order: High School, Bachelor's, Master's, PhD)

Standard Scalar: 9 Continuous Features

- Age, Income, LoanAmount, CreditScore, MonthsEmployed, NumCreditLines, InterestRate, LoanTerm, DTIRatio

12 columns are added after preprocessing: 16 columns—>28 columns

Splitting Strategy:

Stratified Kfold Split:

- Highly imbalanced: Stratify to ensure each set contains same ratio of class 0 vs class 1
- Large dataset, 10% test set, 90% train&validation set with 3 folds

Cross Validation Pipeline:

- **Preprocessor Setup:** ColumnTransformer for different encoders.
- **Model Setup:** Define 4 supervised ML models and specify corresponding hyperparameter grids.
- **Evaluation Metric:** F1 score as weighted harmonic mean of P and R for imbalanced data.
- **Cross-Validation Function (MLpipeline_StratifiedKFold):**

  Loop through 5 random states, for each:
  - Stratify-split the dataset into test set and train&validation set .
  - Perform 3-fold CV on the train&validation set using grid search.
  - Train the pipeline on training data and validate on validation data.
  - Save F1 test scores, as well as best hyperparameters.

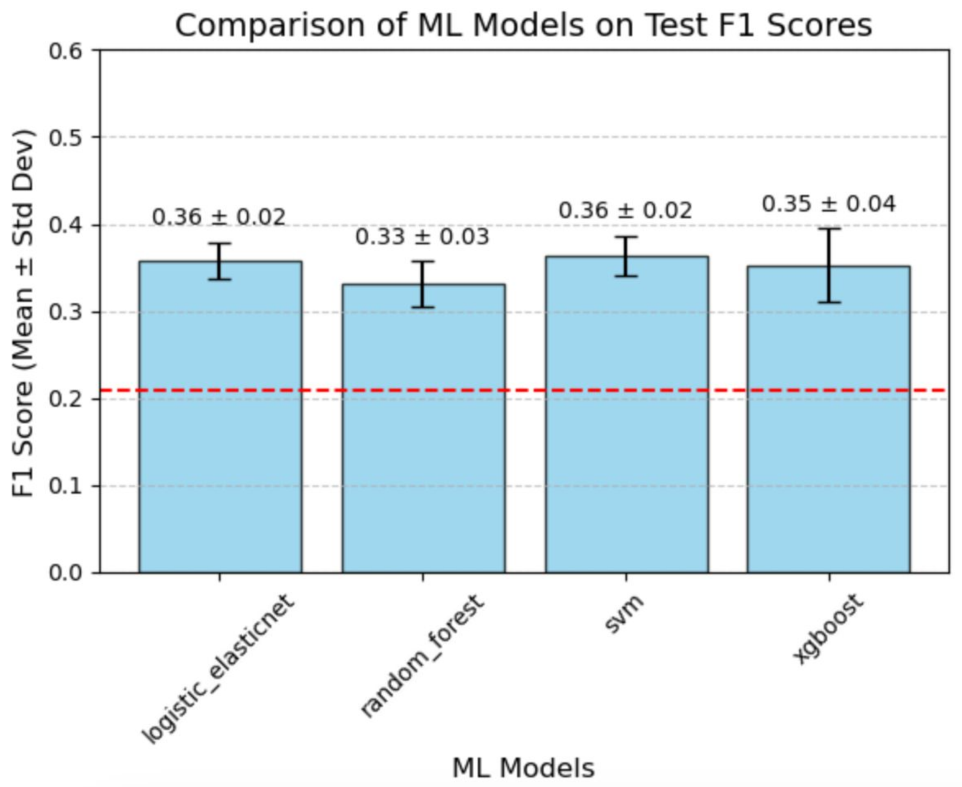
  After all iterations, compute the mean and standard deviation of test F1 scores for each model.
- **Model Evaluation:** Compare mean and standard deviation of test F1 scores for all models, and Identify the best model based on the highest mean test F1 score.

ML Algorithms Summary:

Model	Hyperparameter Tunning	Class weight	Evaluation Metric
Logistic regression (elastic net regularization)	C: [0.01, 0.1, 1, 10, 100] l1_ratio: [0.1, 0.3, 0.5, 0.7, 0.9]	balanced	F1
Random Forest Classification	max_depth: [1, 3, 5, 10, 20, 30] max_features: [0.1, 0.3, 0.5, 0.7, 0.9]	balanced	F1
SVC (support vector classification)	C: [0.01, 0.1, 1, 10, 100] kernel': ['linear', 'rbf'] gamma': ['scale', 'auto', 0.01, 0.1, 1]	balanced	F1
XGBoost	n_estimators: [50, 100, 200], learning_rate: [0.01, 0.1, 0.2], max_depth: [1, 3, 5, 7, 10], subsample: [0.8, 1.0], colsample_bytree: [0.8, 1.0]	scale_pos_weight= imbalance ratio (class0 count / class1 count)	F1

Results: Test Score Summary

Model	Best Hyperparameters	Mean f1 score	Std
Logistic regression (elastic net regularization)	C: 0.1, l1_ratio: 0.3	0.3581	0.0209
Random Forest Classification	max_depth: 5, max_features: 0.1	0.3312	0.0260
SVC (support vector classification)	C': 0.01, Gamma: scale, Kernel: rbf	0.3633	0.0226
XGBoost	N_estimators: 200, Learning_rate: 0.01, Max_depth: 5, Subsample: 0.8, Colsample_bytree: 0.8	0.3529	0.0421



- All of my ML models shows close test results around 0.35~ with very low standard deviations around 0.03~;
- SVC performs a bit better and therefore would be chosen as best model for subsequent interpretations.

Baseline f1 score:0.2081

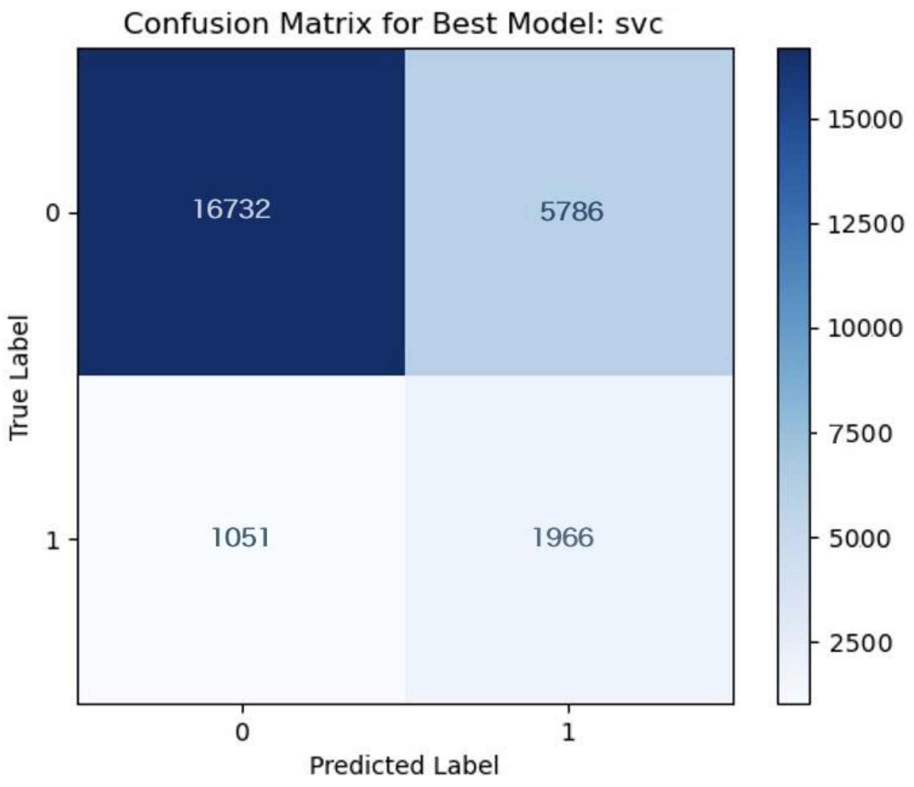
All models demonstrate improvements over baselines scores:

Based on results table and plot,

Best Model: SVC

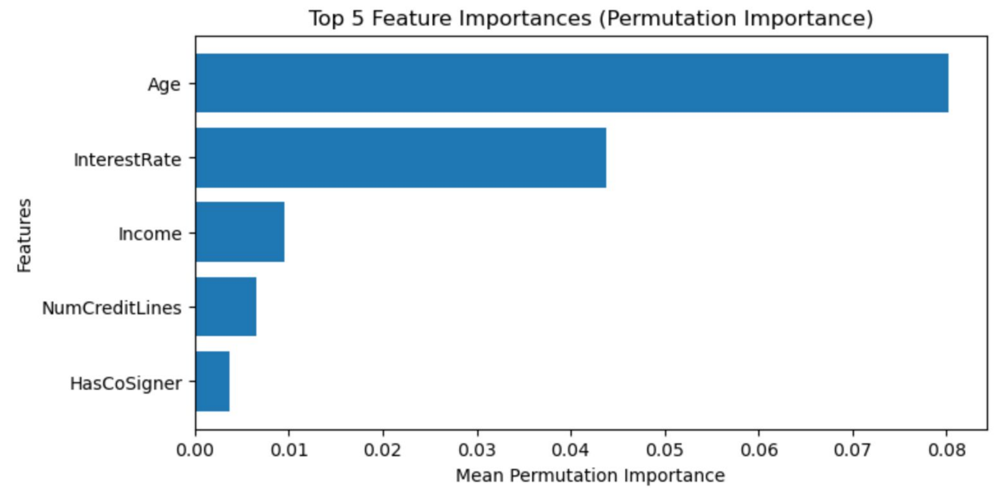
Best Hyperparameter:
'C': 0.01, 'gamma': 'scale', 'kernel': 'rbf'

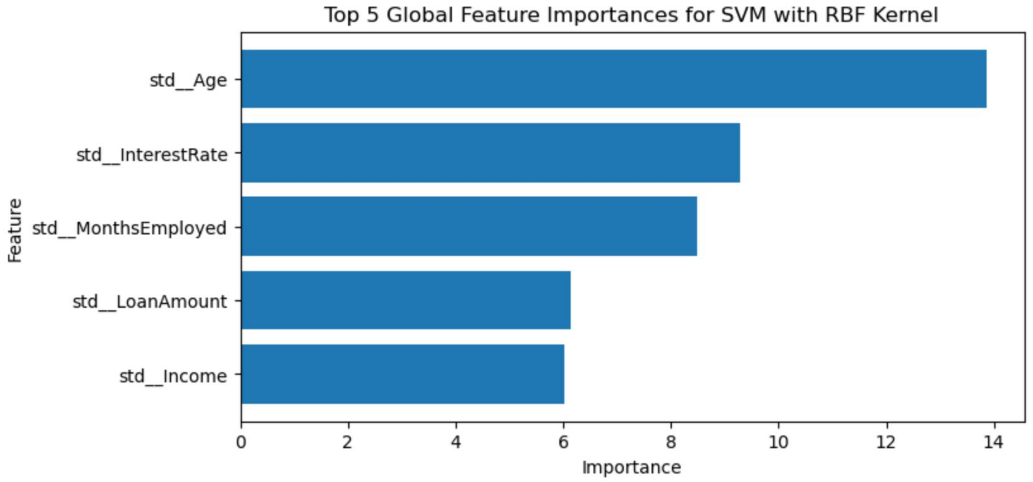
Confusion Matrix:



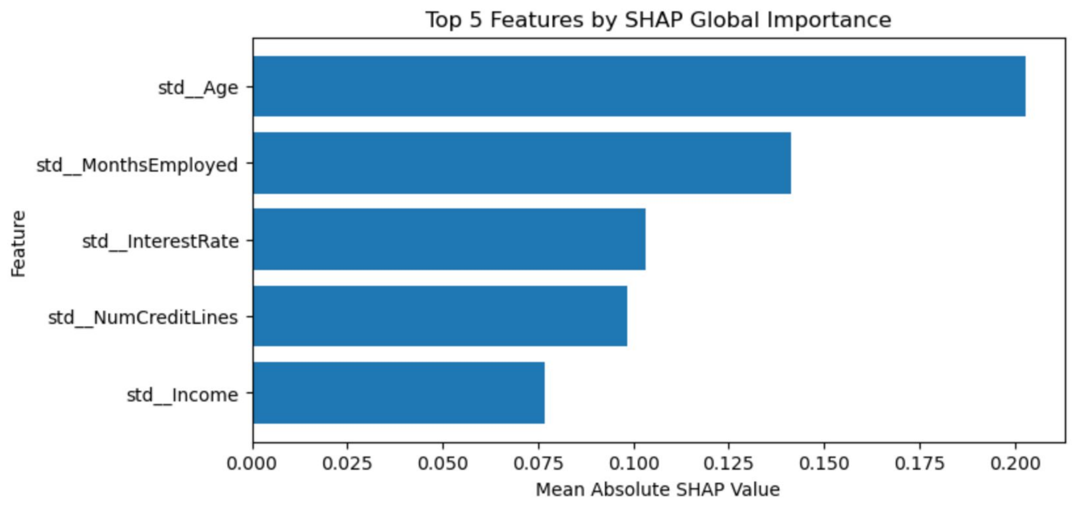
- **Imbalance Handling:** SVC model shows an improvement in detecting the minority class.
- **P-R Trade-off:** The higher false positive rate suggests the model is leaning slightly toward improving recall.
- **Imbalance Impact:** Imbalance in dataset still significantly impacts the model's performance, as evident from the false positive count.

Model Interpretation: Global Feature Importance





Model Interpretation: Global Feature Importance

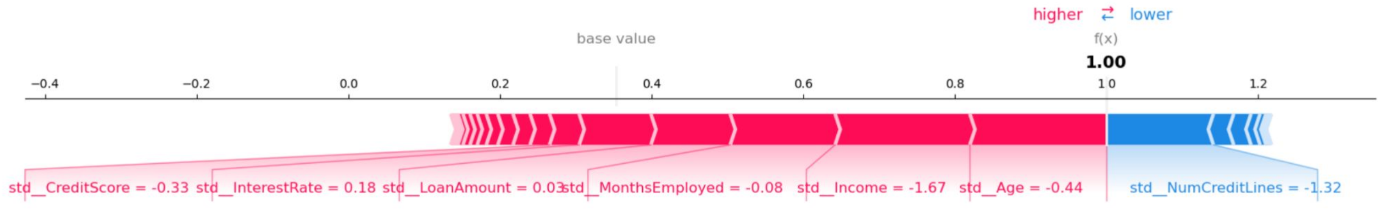


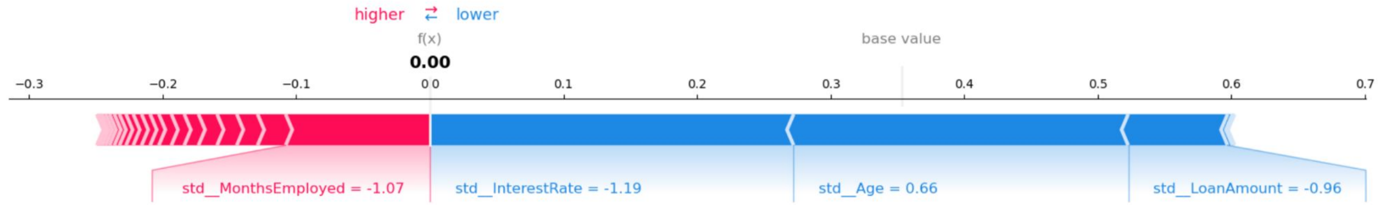
Approaches:

- **Permutation feature importance**
- **Weights of the SVC with RBF kernel**
- **SHAP Global Importance**

Conclusion: Age, Interest Rate, Income, CreditLine plays significant roles in feature performance in predicting if an individual's would cause loan default.

Model Interpretation: Local SHAP Force Plot





Plot 1: InterestRate, Age increase the prediction toward 1, while NumCreditLines decrease it.

Plot 2: Age increase the prediction, while InterestRate (red) decreases it.

Outlook:

To improve the predictive power and interpretability:

1. **Hyperparameter Tuning:**

   Perform more explicit hyperparameter tuning, such as Bayesian optimization to refine the model's performance.

2. **Feature Engineering:**

   Potentially remove the least important features based on feature importance scores to reduce noise and improve generalization.

   Investigate feature correlations and interactions to identify significant relationships.

3. **Addressing Class Imbalance:**

   Explore other techniques to handle class imbalance,:

   ○ Oversampling the minority class as well as undersampling the majority class.

   ○ Combining sampling techniques with ensemble methods like cost-sensitive algorithms.

THANK YOU