Compulsory Exercise 2: Credit risk

Håvard Moseby Sindre Skau Gulliksrud

03 April, 2025

Abstract

In this project, we aim to classify defaults versus non-defaults using a publicly available credit-risk dataset consisting of multiple predictors and a binary response variable. After performing the necessary data preprocessing and visualizing the distributions graphically, we split the data into training set containing 80% of the data and a test set of 20%. On the training set, we implemented two methods: Logistic Regression with Lasso regularization to reduce high bias, and XGBoost. Both models were tuned using 5-fold cross-validation to optimize their respective hyperparameters. By evaluating performance using AUC and F1-score, we found that XGBoost achieved higher predictive accuracy and a better balance between precision and recall compared to the Lasso-logistic model. Additionally, our analysis provided insights into which predictors were most correlated with the default outcome, offering valuable guidance for credit-risk assessment.

Introduction

In the modern economy, characterized by high inflation and rising interest rates, banks face a challenging task in determining who are suitable candidates for home loans and how much credit they can safely extend. The core responsibility of the banks, in relation to mortgage loans, is to identify individuals who are unlikely to default, thus minimizing losses and ensuring profits.

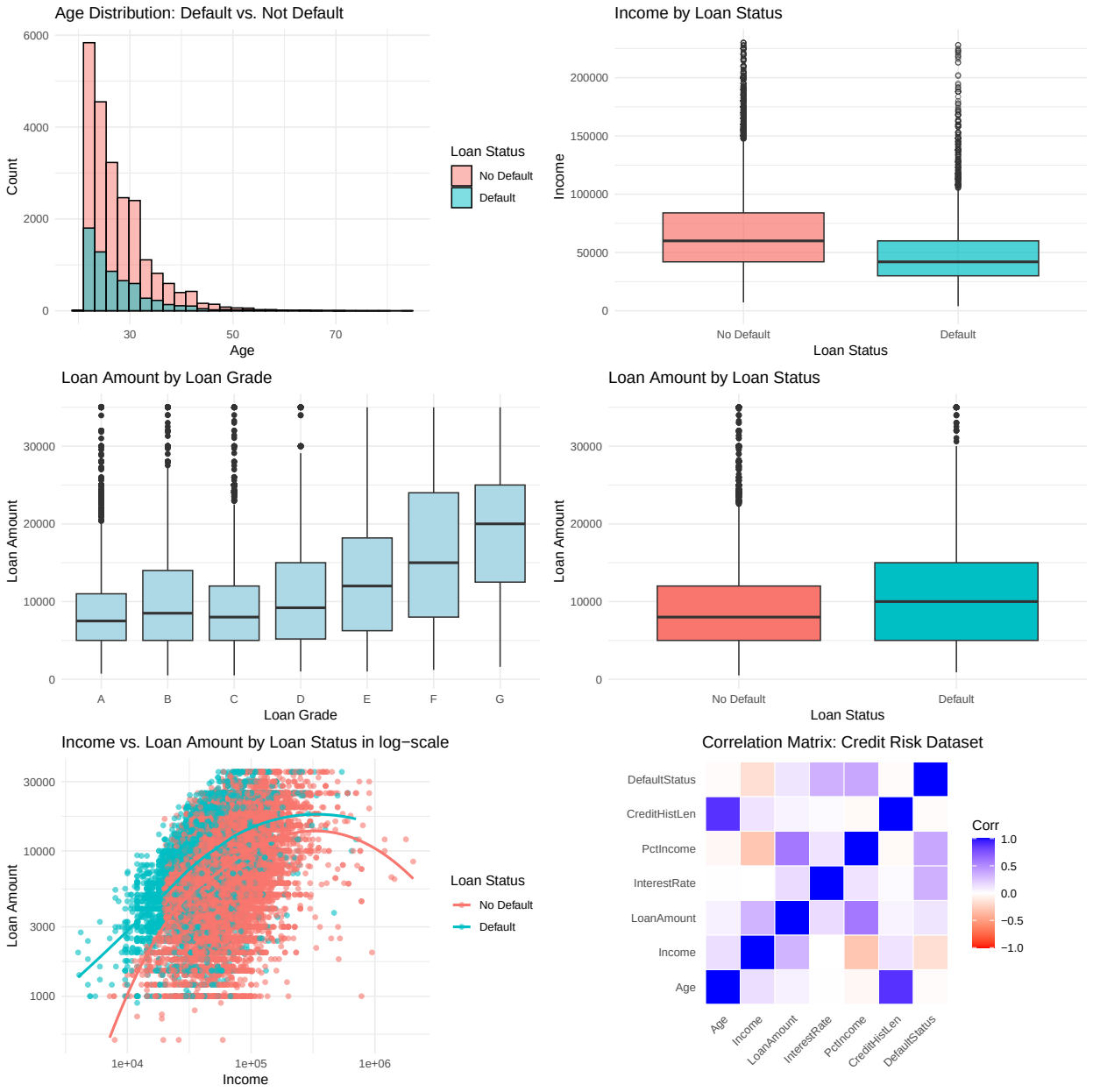
By examining a publicly available credit-risk data set, we aim to identify significant predictors such as income, employment status and credit history. We will also compare various classification methods to determine which model offers the most accurate predictions of default risk.

This topic is especially relevant for banks, but also for students and other young borrowers who often have little or no credit history. These individuals may face uncertainty when it comes to stable income and their ability to repay a loan. By understanding what makes someone more likely to default, banks can make better lending decisions, while young borrowers can better understand what factors affect their chances of getting a loan.

Descriptive data analysis/statistics

```
##    person_age    person_income     person_home_ownership person_emp_length
##  Min.   :20.00   Min.   :   4000   Length:28634          Min.   :  0.000
##  1st Qu.:23.00   1st Qu.:  39462   Class :character      1st Qu.:  2.000
##  Median :26.00   Median :  55900   Mode  :character      Median :  4.000
##  Mean   :27.71   Mean   :  66431                         Mean   :  4.789
##  3rd Qu.:30.00   3rd Qu.:  80000                         3rd Qu.:  7.000
##  Max.   :84.00   Max.   :2039784                         Max.   :123.000
##  loan_intent         loan_grade          loan_amnt     loan_int_rate
##  Length:28634       Length:28634       Min.   :  500   Min.   : 5.42
##  Class :character   Class :character   1st Qu.: 5000   1st Qu.: 7.90
##  Mode  :character   Mode  :character   Median : 8000   Median :10.99
```

```
##                                      Mean   : 9657   Mean   :11.04
##                                      3rd Qu.:12500   3rd Qu.:13.48
##                                      Max.   :35000   Max.   :23.22
##      loan_status    loan_percent_income cb_person_default_on_file
##  No Default:22431   Min.   :0.0000      Length:28634
##  Default   : 6203   1st Qu.:0.0900      Class :character
##                     Median :0.1500      Mode  :character
##                     Mean   :0.1695
##                     3rd Qu.:0.2300
##                     Max.   :0.8300
##  cb_person_cred_hist_length
##  Min.   : 2.000
##  1st Qu.: 3.000
##  Median : 4.000
##  Mean   : 5.793
##  3rd Qu.: 8.000
##  Max.   :30.000
```



Before our analysis, we removed rows with deficient data points, leaving us with 28,634 observations. The dataset includes the variables age, income, home ownership, loan intent, loan grade, employment length, loan amount, interest rate, loan status, percent income, historical default, and credit history length.

Since we aim to predict loan status, it is worth noting that this variable is somewhat imbalanced (only 6,203 defaulted loans out of 28,634). This class imbalance may influence our choice of modeling methods and performance metrics, as accuracy alone might not fully capture the model's performance on the minority class.

The histogram of age shows that the dataset largely consists of younger adults. From the boxplots comparing default vs. no default, we see indications that lower income and higher loan amounts are associated with a higher proportion of defaults.

When examining loan amount by loan grade, we observe that the grades D, F, and G are linked to higher loan amounts than A, B, and C. This seems counterintuitive, as a higher letter grade is meant to signify

"better" creditworthiness. It may be that these specific grades reflect different loan products, or that riskier borrowers are taking out larger loans. This will be interesting to investigate further.

Finally, in our correlation matrix, we see some moderate correlation between variables. This suggests potential predictive value in several covariates, though we also need to be mindful of possible multicollinearity if certain pairs of predictors are strongly correlated. We can see in the scatter plot for Income vs. Loan amount that higher income correlates strong with high loan amount.

Overall, the exploratory plots and summary statistics provide a good starting point for model building and indicate that variables such as income, loan amount, and credit history are likely to be important for predicting default

Methods

Logistic regression

In this project, we will look at two different methods to predict default status. Our first model is logistic regression with a binomial response $Y \in \{0, 1\}$. Specifically, we model the probability $p = P(Y = 1 \mid \mathbf{X})$ using the logit link function:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

Here, $\beta_0, \ldots, \beta_p$ are coefficients determined by maximizing the penalized log-likelihood. The link function ensures $p \in (0, 1)$.

To perform variable selection and prevent overfitting, we include an ℓ_1-penalty, ie. Lasso regularization.

The optimization problem then becomes:

$$\max_{\beta} \ \ell(\beta) - \lambda \sum_{j=1}^{p} |\beta_j|,$$

where $\ell(\beta) = \sum_{j=1}^{n} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$ is the usual log-likelihood for logistic regression with

$$p_i = \frac{1}{1 + \exp\left(-\left(\beta_0 + \beta_1 X_{1,i} + \cdots + \beta_p X_{p,i}\right)\right)}$$

, and $\lambda \geq 0$ controls the strength of regularization.

Large λ values shrink more coefficients toward zero, effectively removing less important predictors.

A key advantage of using Lasso is that it can handle potential multicollinearity and automatically select a subset of relevant features. However, it also introduces a small bias in the estimates. In addition when using logistic regression we assume independent observations.
We find it particularly appealing for this problem given the relatively large set of potential predictors in our dataset (e.g., age, income, loan amount). Since some of our data has outliers, such as income, we will use the log function for those parameters.

Gradient boost

When comparing the logistic regression model, we will use one of the strongest learning ideas that is currently around, XGB boosting. Boosting builds trees (weak classifiers) sequentially, where each new tree is designed to correct the errors made by the previous trees, this is called boosting. The final prediction will be a combination of the contribution of all the trees. Mathematically this can be expressed as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

where K is the total number of trees and f_k represent a decision tree.

XGBoost uses a gradient descent optimization method to minimize an objective function that consists of two parts: the loss function and a regularization term. The objective function is given by:

$$L(\phi) = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where - $\ell(y_i, \hat{y}_i) = -[y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y}_i))]$ is the convex function that measures the difference between the actual outcome and the predicted outcome. - $\Omega(f_k) = \gamma T_k + \frac{1}{2}\lambda \|w_k\|^2$ is the regulation term where λ, γ is hyperparameters that penalize overly complex trees. T_k is the number of leaves in the $K-th$ tree. w_k is the vector of scores assigned to each leaf in that tree. In other words, each leaf node receives a weight (or score), and these weights determine the contribution of the leaf to the final prediction. When implementing XGBoost model, we use 5-fold CV the find the optimal hyperparameters, as in logistic regression.

In practice, when deciding how to split a node, we use a Gain-function. Let G_L, G_R be the sums of gradients in the left and right child nodes, and H_L, H_R are the sums of corresponding Hessian. The Gain from a potential split is computed as:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

This Gain-function indicates whether it is beneficial to split a node. If the gain is less than γ, it is better not to perform the split. Since the algorithm is greedy, it uses weighted quantiles to quickly find the optimal split.

One of the strengths of XGBoost is that it can handle large datasets efficiently. Although boosting can be prone to overfitting on very small datasets, with our 28,634 data points it is expected to perform very well, even though XGBoost is capable of scaling to millions or billions of observations.

In comparison to logistic regression, XGBoost requires more careful tuning of hyperparameters. XGBoost can also be more computationally intensive.

Implementation of our methods

To train and evaluate our models on this credit-risk dataset, we first randomly split the data into a training set (80%) and a test set (20%). We used the training set for both fitting the models and performing hyperparameter tuning via 5-fold cross-validation (CV). For Logistic regression with Lasso we selected the λ that yielded the best average CV performance, and used it in our final model.

For XGBoost we tuned number of boosting rounds and kept the other hyperparameters fixed. We tried up to 200 rounds and applied early stopping at 10 rounds of no improvement. We then chose the iteration that yields the highest mean AUC. Finally, we take the corresponding best_iteration and train our final XGBoost model on the entire training set. Both models are then evaluated on the test set to assess their performance, keeping the test data fully separate until this final step.

When evaluating the two models, we considered metrics such as AUC and F1-score,both of which are less sensitive to skewed class distributions than just accuracy alone. AUC-score(Area Under Curve) measures how well the model ranks defaults above non-defaults across all possible classification thresholds.

F1-score, however, combines precision (the fraction of predicted defaults that are truly defaults) and recall (the fraction of all actual defaults the model captures) into a single metric. This makes it especially

relevant for our imbalanced credit-risk data, where a high accuracy alone can be misleading if the model overwhelmingly predicts "No Default".

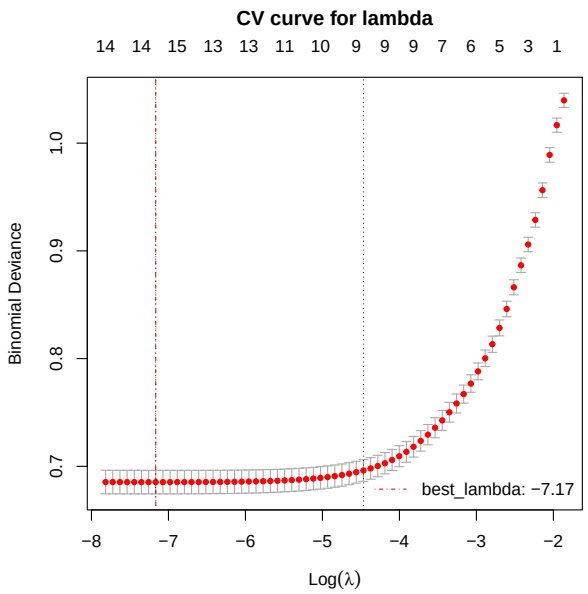
Even though our dataset is skewed, with far fewer defaults than non-defaults, we kept the classification threshold at 0.5.

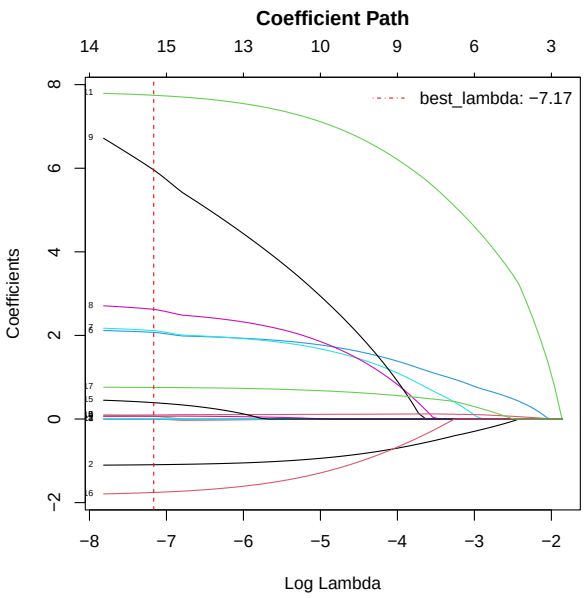
Results and interpretation

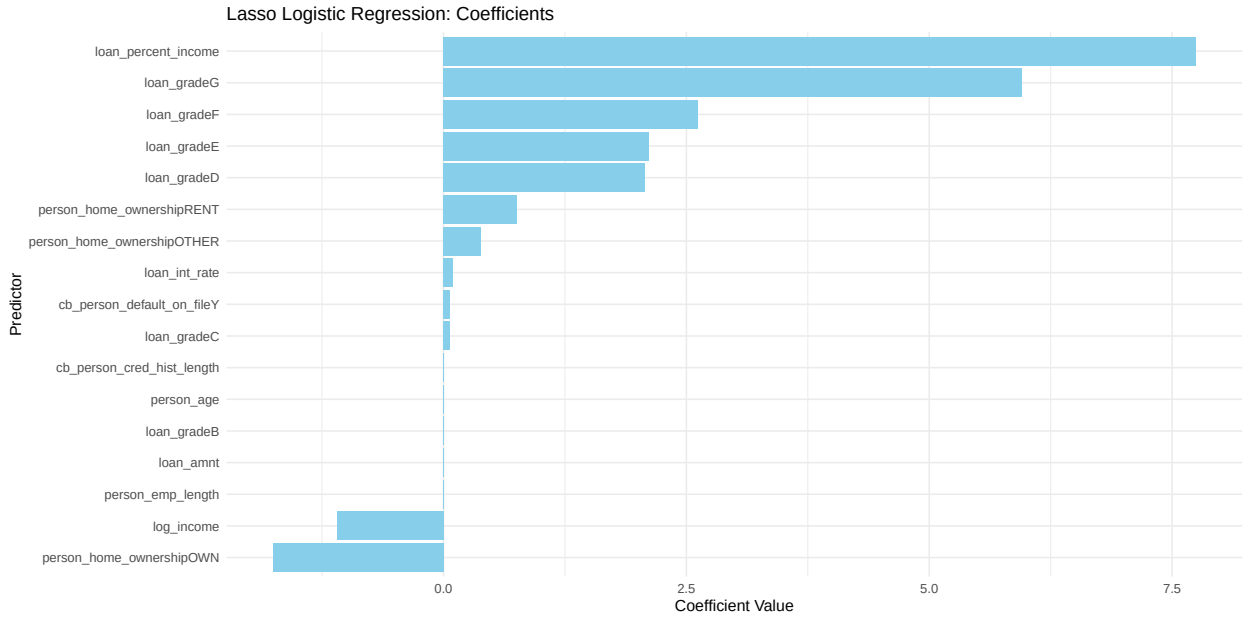
Logistic regression

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4240  584
##          1  204  699
##
##                Accuracy : 0.8624
##                  95% CI : (0.8532, 0.8712)
##     No Information Rate : 0.776
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5577
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9541
##             Specificity : 0.5448
##          Pos Pred Value : 0.8789
##          Neg Pred Value : 0.7741
##              Prevalence : 0.7760
##          Detection Rate : 0.7404
##    Detection Prevalence : 0.8423
##       Balanced Accuracy : 0.7495
##
##        'Positive' Class : 0
##
```

```
## F1-score on Test Set for Logistic regression: 0.9149763
```





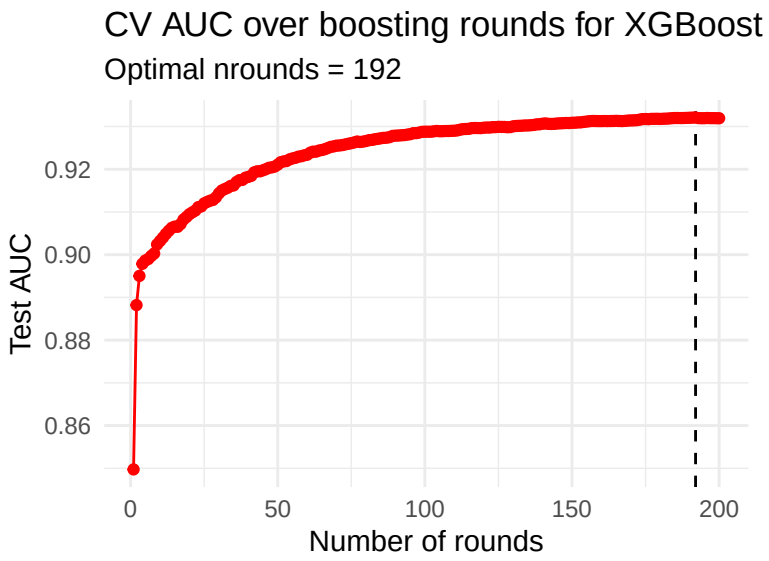


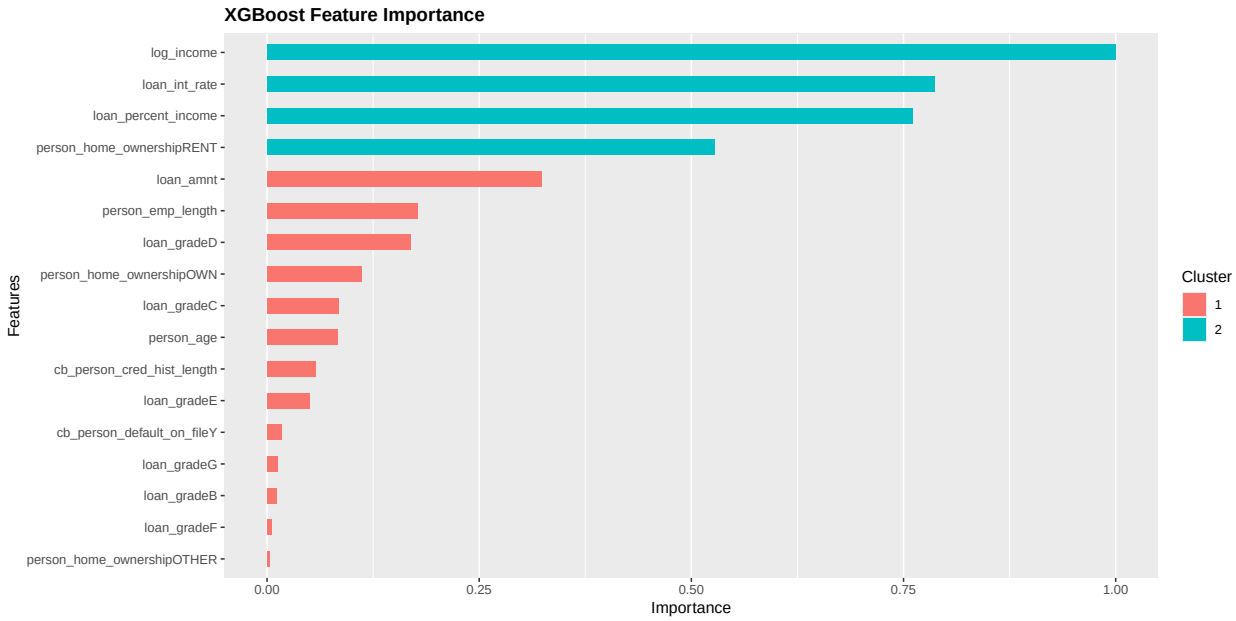
XGBoost

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 4352  417
##          1   92  866
##
##                Accuracy : 0.9111
##                  95% CI : (0.9035, 0.9184)
```

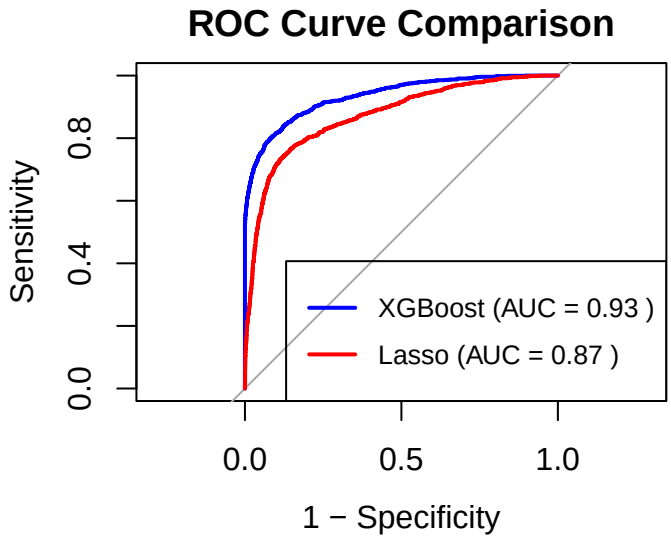
```
##     No Information Rate : 0.776
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7191
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9793
##             Specificity : 0.6750
##          Pos Pred Value : 0.9126
##          Neg Pred Value : 0.9040
##              Prevalence : 0.7760
##          Detection Rate : 0.7599
##    Detection Prevalence : 0.8327
##       Balanced Accuracy : 0.8271
##
##        'Positive' Class : 0
##
```

```
## F1-score on Test Set for XGboost: 0.944752
```





Comparison



Interpretation

Both models were evaluated on the credit risk classification task, and the results indicate that the XGBoost model achieved overall better predictive performance than the logistic regression with Lasso regularization. XGBoost obtained a higher AUC and a slightly higher F1-score, and thus showcased its ability to distinguish defaulters from non-defaulters and maintain a good balance between precision and recall. The confusion matrices highlight that XGBoost had a higher specificity, meaning fewer false positives, contributing to a more balanced accuracy between the classes. In contrast, the Lasso-regularized logistic model, while performing well with a high F1-score, showed somewhat lower specificity. It detected most of the defaults, but at the cost of more false alarms. Thus, in terms of pure predictive metrics, XGBoost provided more accurate and balanced classifications of credit risk in this case.

However, this improved performance comes with trade-offs. One of them being the computational cost and model complexity. Logistic regression with Lasso is relatively simple and computationally lightweight. Training involves convex optimization and the Lasso penalty automatically performs feature selection by shrinking less important coefficients to zero. This simplicity means the logistic model is quick to train and easy to tune. Using CV we found the optimized value of λ. As the Coefficient-plot indicates, most of the parameters are still close to their original value.

XGBoost, on the other hand, is an ensemble of many decision trees and is significantly more complex. It required careful hyperparameter tuning (number of boosting rounds) and uses techniques like early stopping to prevent overfitting. We therefore found the optimal value of boosted rounds. Training XGBoost was more computationally intensive and time-consuming compared to the logistic model. In this project's dataset, containing tens of thousands of records, the computational cost was still manageable, but in general XGBoost scales less easily to extremely large datasets or very high dimensional data without substantial computing resources compared to logistic regression. Thus, logistic regression wins on efficiency and simplicity, whereas XGBoost expends more computation to achieve higher accuracy.

XGBoost's complexity also reflects its greater flexibility. The logistic model is a generalized linear model assuming a linear relationship between features and the log-odds of default, which can impose a strong bias. If the true relationship between predictors and default risk is nonlinear or involves complex interactions, a simple logistic model may have high bias because it cannot capture those patterns unless they are manually encoded. In contrast, XGBoost can naturally model the nonlinear effects and interactions among features through its tree-based structure. This flexibility allows XGBoost to fit the training data more closely and therefore reduce the bias. However,the downside is that such a flexible model can have higher variance as it might overfit noise in the training data. With our method, techniques like cross-validation and regularization helped control XGBoost's variance, resulting in strong test-set performance that indicates it generalized well. The logistic regression model, due to its simplicity and the regularizing effect of the Lasso penalty, inherently had lower variance and a lower risk of overfitting. Its bias was higher, but this bias-variance trade-off can actually be advantageous in scenarios with limited data or very noisy features. In summary, logistic regression represents a high-bias/low-variance approach, providing more stable, if less flexible predictions, whereas XGBoost is a lower-bias/higher-variance approach that can capture complex relationships at the cost of needing more careful tuning to avoid overfitting. These differences are also evident in the ROC curves, where XGBoost's higher AUC compared to logistic regression's reflects its lower bias and ability to capture more complex patterns,

Another important consideration is interpretability. Logistic regression with Lasso yields a sparse linear model that is relatively easy to interpret. Each coefficient in the model indicates the direction and strength of the association between a predictor and the likelihood of non-default vs. default as seen in the Lasso Logistic Regression: Coefficients.

XGBoost, by contrast, is essentially a black-box model consisting of an ensemble of decision trees. Its individual predictions result from complex interactions across many trees, which makes it difficult to explain why a particular borrower was classified as high or low risk without specialized tools. While techniques such as feature importance plot can provide some insight into XGBoost's behavior as seen in XGBoost Feature Importance, these interpretations are more approximate and complex than the clear coefficient-based explanation from logistic regression.

Therefore, there is a trade-off: XGBoost offers better raw predictive power and flexibility, whereas logistic regression offers simplicity and interpretability, which can be particularly important in domains like finance where decisions need to be transparent.

Summary

Our findings correspond well with our expectations. Both models demonstrated good predictive performance on the credit-risk classification task. However, XGBoost achieved higher AUC and F1-score compared to Lasso-logistic regression, indicating a better ability to distinguish between defaulters and non-defaulters. This enhanced performance comes at the cost of increased model complexity and computational demand. In

practice, this means that while XGBoost may offer superior accuracy, its complex, "black-box" nature can make it challenging to implement and interpret in environments where transparency is crucial. In contrast, Lasso-logistic regression yields a sparser, more interpretable model with coefficients that clearly indicate the direction and magnitude of each predictor's impact. Therefore, in settings where explainability and ease of implementation are important, Lasso-logistic regression may be the more suitable choice despite its slightly lower predictive performance.

References

1. XGBoost from xgboost.readthedocs.io.
2. Credit Risk Dataset from Kaggle.com.