Profitability of Different Credit Risk Classification Algorithms

Elliott Oates

Word Count: 3142

Contents

1 Introduction

Retail lending has and always will be a pillar in the constantly evolving landscape of the financial industry products and thus Lenders credit risk modelling and loan application assessment must develop as well . This report evaluates different machine learning classification algorithms to best predict credit risk in loan applications. Credit risk assessment is a pivotal task for loan managers, and involving the evaluation of the likelihood that a borrower may default or fail to meet payment obligations on a loan. The accuracy of this prediction has far-reaching consequences, not only for the banks in terms of potential losses but also for the broader economic stability.

While using machine learning algorithms in credit risk assessment is not new, the relevance of this report stems from the proliferation of digital banking. Digital platforms and NeoBanks are making it increasingly convenient to access concurrent banking products, such as personal and business loans from many different providers and thus the need for robust predictive models is even more pronounced. Digital banks, characterized by their agility and customer-centric approaches, are reshaping the financial landscape. They offer easier and quicker access to their products processes, which, while beneficial in terms of customer experience, also pose new challenges in credit risk assessment. As these digital entities attract a diverse range of customers, loan managers are compelled to employ more sophisticated, data-driven approaches to accurately assess credit risk.

2 Data and exploratory analysis

2.1 Dataset Background and Due Diligence

The german credit dataset used in this report is described as a "Stratified sample from actual credits with bad credits heavily over sampled" and contains 1,000 observations of German bank loan data with a response indicating the credit risk("South German Credit", 2019). 700 of the observations have a credit risk of good indicating that the loan was repaid and 300 are bad indicating a default on the loan. Ratings agency Fitch forecasts default rates to be between 3.5% and 4% for leveraged loans in 2024 which is very low compared to our sample which represents a default rate of 30% (Fitch Ratings, 2023). It is important to over-represent a class as it can lead to improved model performance and reduced bias. If we used a 1,000 observation dataset with default rates closer to population default rates of 3.5% then we would only have 35 observations of bad credit, 30% of which would be excluded from the training data. This would lead to our models being biased towards the majority classification "good credit" due to insufficient exposure to the minority class.

While the original data is encoded in German, the dataset is supplied alongside an R.script that encodes all the variable names in English and converts all the binary and categorical variables into factor and multilevel factor variables respectively. Having run the script provided, we obtain a preliminary view of the data types can be seen as in table 1.

Table 1: First 5 Rows of Unprocessed South German Credit Dataset

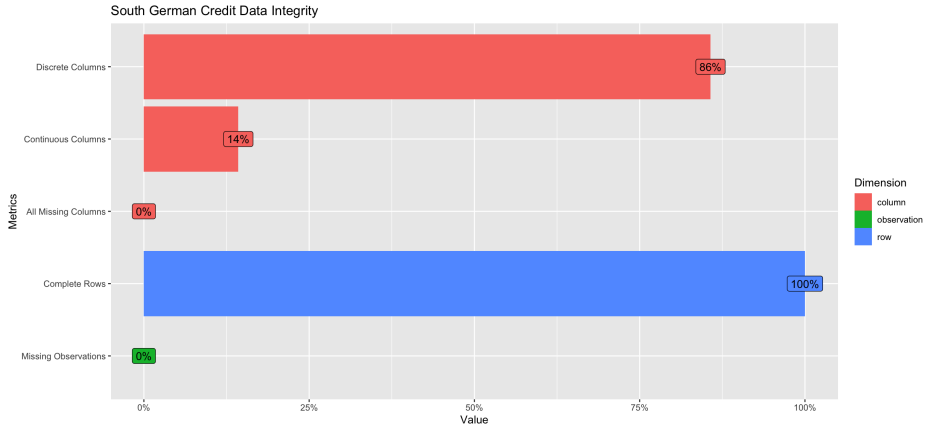
	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_debtors
1	no checking account	18	all credits at this bank paid back duly	car (used)	1049	unknown/no savings account	< 1 yr	< 20	female : non-single or male : single	none
2	no checking account	9	all credits at this bank paid back duly	others	2799	unknown/no savings account	1 <= ... < 4 yrs	25 <= ... < 35	male : married/widowed	none
3	... < 0 DM	12	no credits taken/all credits paid back duly	retraining	841	... < 100 DM	4 <= ... < 7 yrs	25 <= ... < 35	female : non-single or male : single	none
4	no checking account	12	all credits at this bank paid back duly	others	2122	unknown/no savings account	1 <= ... < 4 yrs	20 <= ... < 25	male : married/widowed	none
5	no checking account	12	all credits at this bank paid back duly	others	2171	unknown/no savings account	1 <= ... < 4 yrs	< 20	male : married/widowed	none

	present_residence	property	age	other_installment_plans	housing	number_credits	job	people_liable	telephone	foreign_worker	credit_risk
1	>= 7 yrs	car or other	21	none	for free	1	skilled employee/official	0 to 2	no	no	good
2	1 <= ... < 4 yrs	unknown / no property	36	none	for free	2-3	skilled employee/official	3 or more	no	no	good
3	>= 7 yrs	unknown / no property	23	none	for free	1	unskilled - resident	0 to 2	no	no	good
4	1 <= ... < 4 yrs	unknown / no property	39	none	for free	2-3	unskilled - resident	3 or more	no	yes	good
5	>= 7 yrs	car or other	38	bank	rent	2-3	unskilled - resident	0 to 2	no	yes	good

Several variables that traditionally one would assume to be recorded as continuous numerical data such as savings, employment duration, instalment rate, and number_credits are recorded as categorical variables. The Deutschemark (DM),the currency that the loans are recorded in was the currency of west Germany

between 1948 and 1990 we can assume the loans where recorded at the latest in 1990. Over this period computing, data entry and storage process and technology where elementary and costly and thus we can attribute the choice to record several numerical variables as categorical ones as a result of the limitations of the times.

Figure 1: Plot and Table Describing the Data Types and Completeness of the Unprocessed Dataset



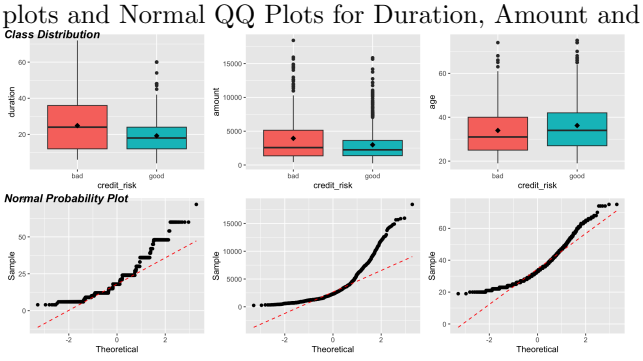
	Info
rows	1000.00
columns	21.00
discrete_columns	18.00
continuous_columns	3.00
all_missing_columns	0.00
total_missing_values	0.00
complete_rows	1000.00
total_observations	21000.00
memory_usage	101936.00

It is also important to check the integrity of the data to ensure that there are no observations with missing values as they cause the machine learning algorithms to fail and the ones such as K-Nearest Neighbours that support missing values would be result in biased biased predictions. Figure (1) which includes a graphical and tabular description of the amount and proportions for different data types in the dataset confirms that there are no missing values or empty columns.

2.2 Numerical Predictors

This section evaluates the numerical predictors: Duration, Amount, and Age (see Figure 2). The "Normal QQ" plots reveal deviations from the line of symmetry, indicating asymmetric distributions. For all three variables, outliers are noted, particularly in Amount and Age. These outliers are crucial for modeling, especially for loans with longer durations and larger amounts, typically associated with older customers. Box plots indicate a wider interquartile range for bad loans, suggesting a greater variability in these cases. Notably, the right skewness in these variables might affect the normality assumptions in our machine learning algorithms.

Figure 2: Box plots and Normal QQ Plots for Duration, Amount and Age Variables



The correlation matrix (Table 2) shows no concerning correlations among these variables, with the highest being a logical 0.62 between Amount and Duration.

Table 2: Correlation Matrix

	duration	age	amount
duration	1.00	-0.04	0.62
age	-0.04	1.00	0.03
amount	0.62	0.03	1.00

2.3 Categorical and Binary Variables

This section contains several plots of the original class distributions for each categorical variable. Following the visualisation of the original distributions and subsequent analysis the data set was refined by restructuring and removal of certain variables as elaborated.

Regarding categorical variables, including ordinal and binary types, an imbalance in class distribution is noted, which might lead to overfitting. For instance, the status variable (Figure 3.1) and credit history classes (Figure 3.2) display imbalances, but due to their distinct implications, all classes are retained. In contrast, significant imbalances in purpose classes (Figure 3.3) necessitate combining similar categories for better model performance.

Figure 3: Distribution across categories for Status, Credit History and Purpose

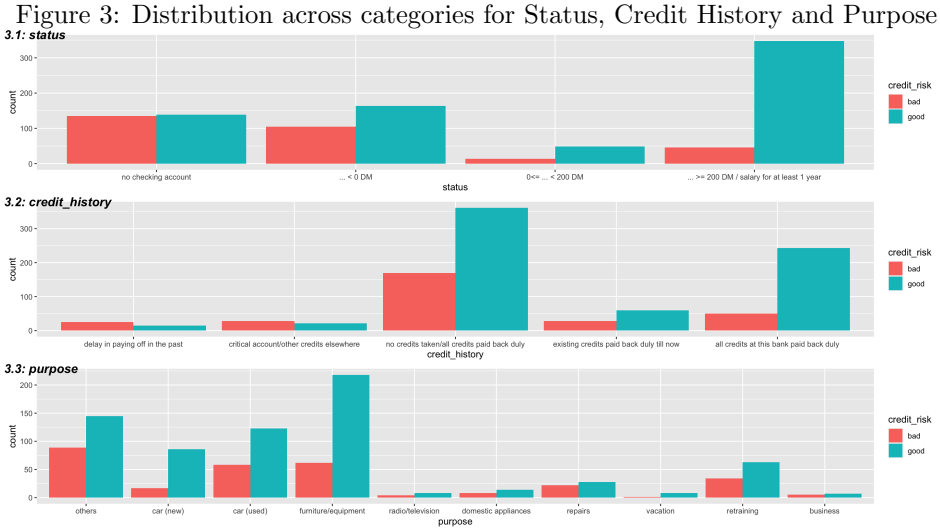
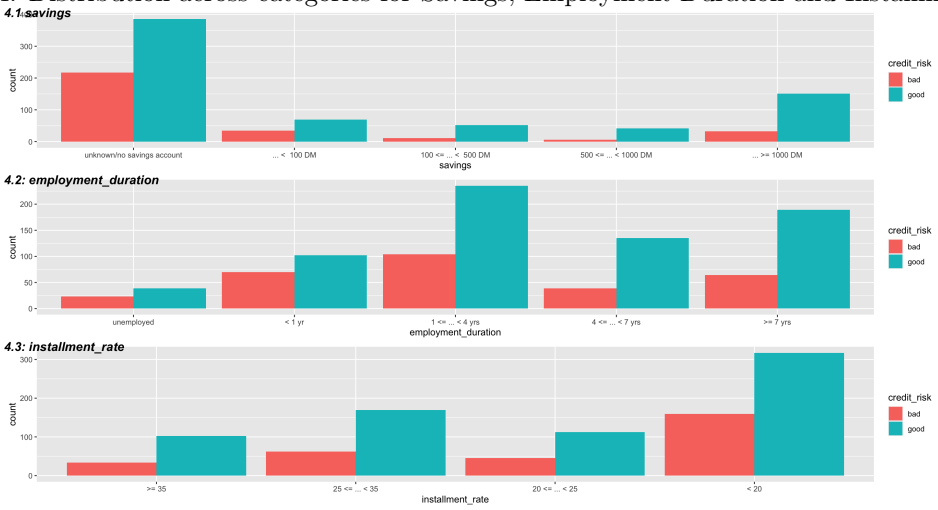


Figure 4: Distribution across categories for Savings, Employment Duration and Installment Rate



Savings categories (Figure 4.1) are restructured to address class imbalances, while maintaining the unique category for unemployed individuals, given their distinct credit implications.

The personal status sex variable is quite troublesome. Due to peculiar data recording, combining categories to distinguish either sex or relationship status is not possible and thus it is best to drop the variable to address the class imbalance. Other debtors (Figure 5.2) is dropped due to significant imbalances.

Figure 5: Distribution across categories for Personal_status_sex, Other_debtors and Present_residence

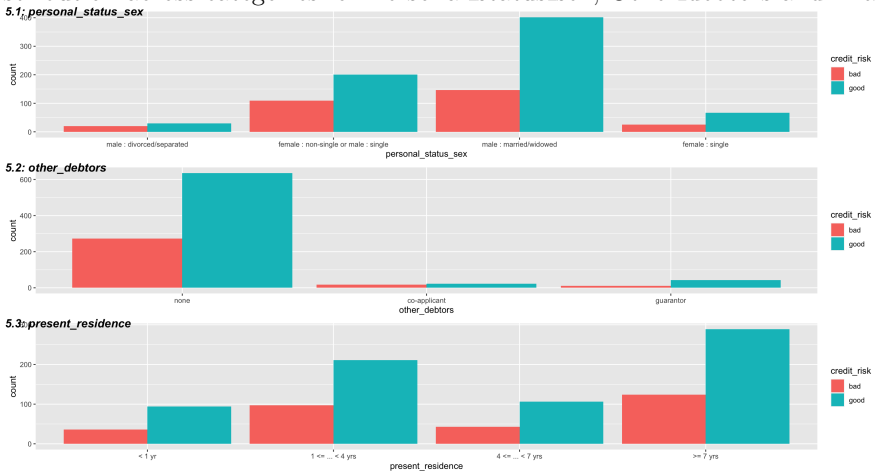
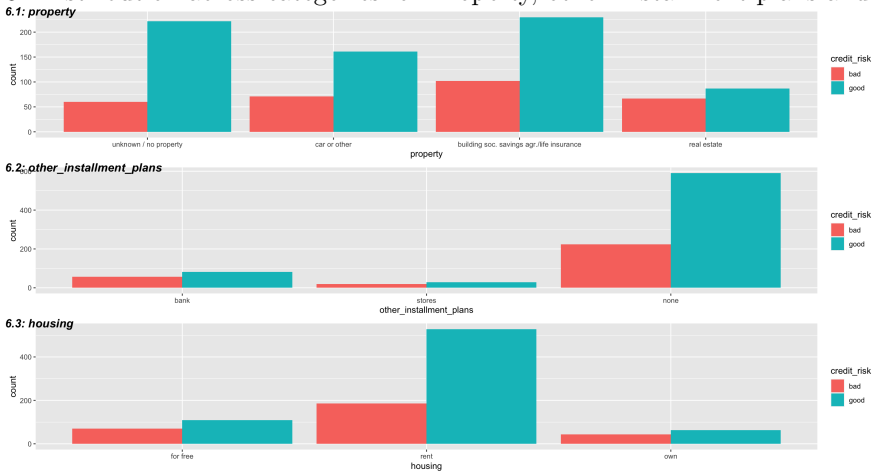
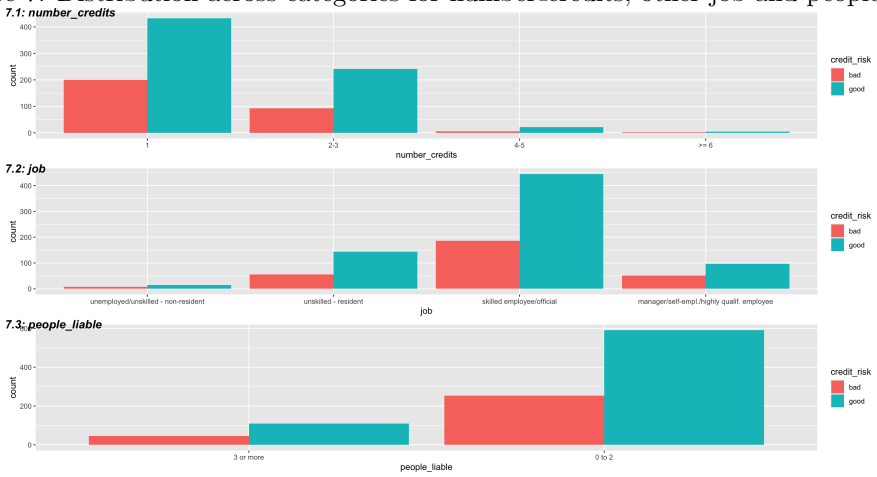


Figure 6: Distribution across categories for Property, other Installment plans and Housing



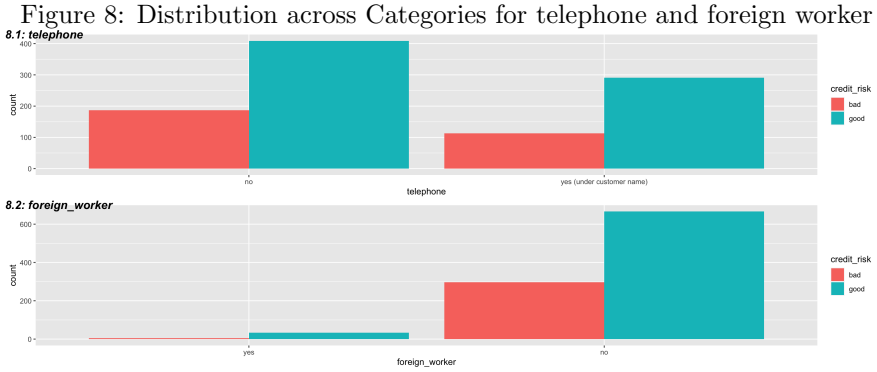
Other installment plans (Figure 6.2) and housing categories (Figure 6.3) are condensed into binary variables for a more balanced representation. Similarly, the number of credits (Figure 7.1) is simplified to a binary indicator to address class imbalances. The variable for people liable (Figure 7.3) is dropped due to its significant imbalance.

Figure 7: Distribution across categories for number_credits, other job and people_liable



Lastly, the telephone (Figure 8.1) and foreign worker variables (Figure 8.2) are removed due to their diminished predictive relevance and extreme imbalances, respectively.

Figure 8: Distribution across Categories for telephone and foreign worker



Based on the above analysis, there have been several variables dropped from the data-set and several reclassified. Table (3) depicts the new class proportions for the categorical variables in the final dataset.

Table 3: Class Proportions following the changes to data structure and variable omissions

credit_risk	good 70%	bad 30%					
status	no checking account 27.4%	... <0 DM 26.9%	0<= ... <200 DM 6.3%	... >= 200 DM / salary for at least 1 year 39.4%			
credit_history	delay in paying off in the past 4%	critical account/other credits elsewhere 4.9%	no credits taken/all credits paid back duly 53.0%	existing credits paid back duly till now 8.8%	all credits at this bank paid back duly 29.3%		
purpose	others 24.3%	car (new) 10.3%	car (used) 18.1%	furniture/equipment 28.0%	domestic appliances/radio/television 3.4%	repairs 5.0%	business/retraining 10.9%
savings	unknown/no savings account 60.3%	... <100 DM 10.3%	100 <= ... <1000 DM 11.1%	... >= 1000 DM 18.3%			
employment_duration	unemployed 6.2%	<1 yr 17.2%	1 <= ... <4 yrs 33.9%	4 <= ... <7 yrs 17.4%	>= 7 yrs 25.3%		
installment_rate	>= 35 13.6%	25 <= ... <35 23.1%	20 <= ... <25 15.7%	<20 47.6%			
present_residence	<1 yr 13%	1 <= ... <4 yrs 30.8%	4 <= ... <7 yrs 14.9%	>= 7 yrs 41.3%			
property	unknown / no property 28.2%	car or other 23.2%	building soc. savings agr./life insurance 33.2%	real estate 15.4%			
number_credits	1 63.3%	>=2 36.7%					
job	unskilled/Unemployed 22.2%	skilled employee/official 62.0%	manager/self-empl./highly qualif. employee 14.8%				
renting	yes 71.4%	no 28.6%					

3 Machine Learning Models

3.1 Logistic regression

Arguably the most fundamental binary classification method in machine learning is the logistic regression. The logistic function as in equation (1) models the probability that Y (credit_risk) = 1 given X (Predictor Variables).

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_k X_k)}} \tag{1}$$

The coefficients for the logistic regression of credit_risk on the full set of variables can be seen in table table (18) in Appendix A. The models performance can be assesed using the AIC which is reported as 718.46 as well as the test misclassification rate retrieved from table (4. As this model is untuned and built on the full set of models it provides a useful benchmark to compare other more complex algorithms.

Table 4: Logistic Regression Full Model Confusion Matrix

		Response Test	
		bad	good
LogPred	bad	20	8
	good	73	199

Test MSE	$(73 + 8)/300 = 27\%$

3.2 LDA, QDA and KNN

3.2.1 Linear Discriminant Analysis

The out of sample performance of the Linear Discriminant Analysis model can be retrieved from table(5). The out of sample misclassification error rate is 28% when using a 0.3 cut-off probability and 27% using the prior defualt probabilities.

LDA assumes that the predictors are gaussian distributed and given our dataset's features and particularly the mix of un-ordered categorical variables this assumption may not hold leading to reduced accuracy. While LDA differs from logistic regression in that it models the distribution of predictors for each of the response classes separately since both produce linear decision boundaries they often have similar performance as is the case here.

Table 5: Linear Discriminant Analysis Confusion Matrix

		Response Test	
		bad	good
LDA Pred	bad	21	12
	good	72	195

Test MSE	$(72 + 12)/(300) = 28\%$

3.2.2 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is an extension of LDA that allows for different covariance for each class. This flexibility means that QDA may preform better when the assumption of equal class covariance

is not held.

The QDA models out of sample performance as in table (6) has a test error rate of 29% this rises to 30% when using prior probabilities of groups as a cutoff instead of 0.3. While the assumptions are relaxed increasing flexibility this comes with the potential for over fitting particularly with smaller datasets or ones with a relatively large amount of features such as ours.

Table 6: Quadratic Discriminant Analysis Confusion Matrix

		Response Test	
		bad	good
QDA Pred	bad	40	34
	good	53	173

Test MSE	$(53+34)/(300) = 29\%$

3.2.3 KNN

K-Nearest Neighbours (KNN) a non-parematric method that assigns an observations class based on the majority class for the k-nearest neighbours in the training set. The application of KNN involves using 10 fold cross validation to find the optimal value of k in the range of 10:200. As shown in figure9 the optimal value of $K = 40$ was then used.

Shown in table (7) the test error rate with $k = 40$ was 29%. KNN despite increased flexibility and less assumptions on the distribution offers no improvement to the more interpretable algorithms. However it does have a lower false positive rate which in our application will be more profitable for a loan manager.

Figure 9: 10 fold Cross Validation Error rates for KNN algorithm

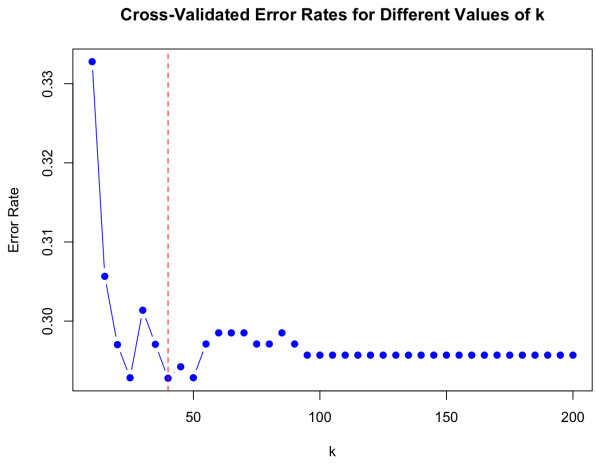


Table 7: KNN (k=40) Confusion Matrix

		Response Test	
		bad	good
KNN Pred	bad	7	3
$k = 40$	good	86	204

Test MSE	$(86+3)/(300) = 29.666\%$

3.3 Subset Selection

Subset selection involves selection only the most relevant features in a dataset for use in the final model in order to reduce over fitting and enhance interpretability and performance. The method of subset selection such as best subset, forwards or backwards as well as the selection criteria determine which variables are included in the final model.

3.3.1 BIC criteria

Bayesian Information Criterion (BIC) chooses the best variable by balancing model complexity with goodness of fit and penalizing models with more parameters. Figure (10) depicts how initially model performance increases but then as more predictors are included the BIC increases due to the penalty. The formula for BIC is given by equation 2 where n is the number of observations, $\hat{L}$ is the maximum value of the likelihood function and k is the number of parameters in the model.

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L}) \tag{2}$$

The model chosen with the optimum amount of variables (4) has chosen a subset of variables to include 'credit_history','duration' and 'status' and the values of each coefficients can be found in Appendix B. The out of sample performance described in table 8 shows that the best model in terms of balancing simplicity and goodness of fit has an out of sample error rate of 29.66% most of which is made up from false positives.

Figure 10: BIC for different values of K

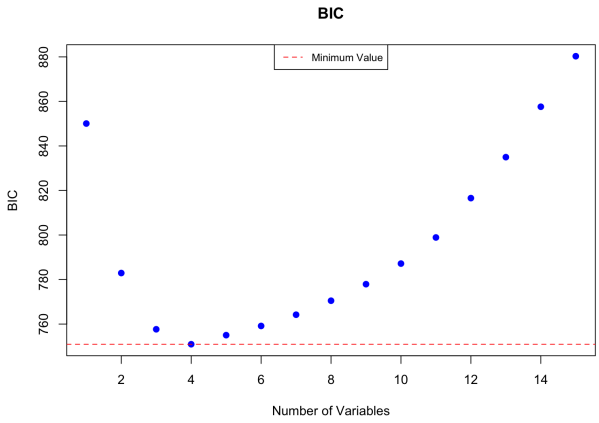


Table 8: BIC Subset Selection Confusion Matrix

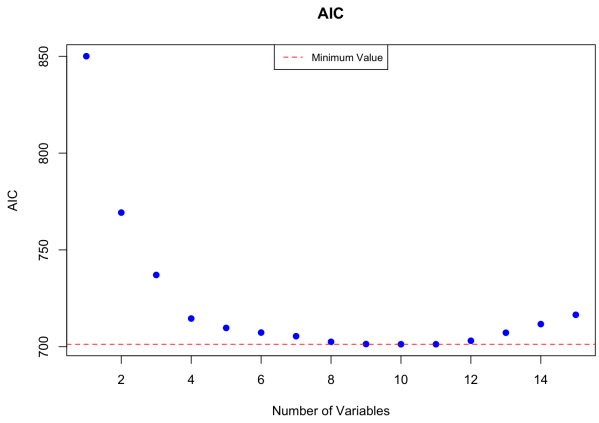
		Response Test	
		bad	good
BIC Pred $k = 40$	bad	12	8
	good	81	199

Test MSE	$(81 + 8)/(300) = 29.666\%$

3.3.2 Akaike Information Criterion

Akaike Information Criterion (AIC) is another method for model selection that balances out complexity and the goodness of fit by penalizing over fitting. However unlike BIC, AIC favours more complex models. The formula for AIC is given by equation 3 where $\hat{L}$ and k are as before.

Figure 11: AIC for different model complexities



$$\text{AIC} = 2k - 2\ln(\hat{L}) \tag{3}$$

As shown in figure 11 the optimal number of predictors in terms of AIC was 9. The coefficients of this model can be found in Appendix C. The best model in terms of AIC has a test classification rate of 28.66%. This is an improvement compared to the BIC chosen model suggesting that it may be better to use a model with more variables.

Table 9: AIC Subset Selection Confusion Matrix

		Response Test	
		bad	good
AIC Pred	bad	17	10
	good	76	197

Test MSE	$(76 + 10)/(300) = 28.66\%$

3.4 Shrinkage Methods

Shrinkage methods are another set of techniques that tune the model fitting process to reduce over-fitting and improve prediction accuracy. By 'shrinking' the coefficients for parameters either towards 0 or each other they result in models that generalise to new data by penalizing the size of coefficients. In applications with large numbers or highly correlated predictors shrinkage methods are especially effective in by addressing multicollinearity and the variance-bias trade off.

3.4.1 Ridge Regression

Ridge regression (L2 Regularisation) is very similar to least squares but the coefficients are found by minimizing the objective function as in 4 where λ is the tuning parameter that control the impact of the penalty. The optimal value for is determined separately λ . Figure 12 shows that the optimal value for λ was 0.01747528.

$$\text{Minimize} \left(\sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right) \tag{4}$$

Figure 12: Misclassification Error Rates for different values of λ

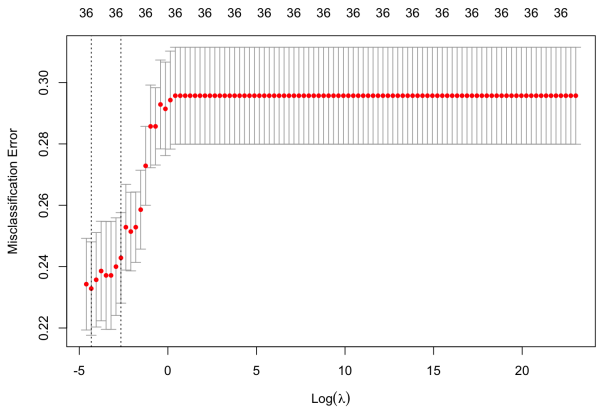


Table 10: Ridge Regression, Optimal λ Confusion Matrix

		Response Test	
		bad	good
Ridge Pred opt λ	bad	34	24
	good	59	183

Test MSE	$(59+24)/(300) = 27.66\%$

The test error rate for the ridge regression model with the optimal λ was 27.66% and this slight improvement in performance suggests models that deal with multicollinearity may be best. Additionally it has a comparatively low false positive rate.

3.4.2 Lasso

Lasso Regression (L1 Regularisation) solves the objective function as in equation 5 where as before λ determines the strength of the penalty. The difference between lasso and ridge is that lasso by using the absolute value of the coefficient leads so some coefficients being shrunk to 0.

$$\text{Minimize} \left(\sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right) \tag{5}$$

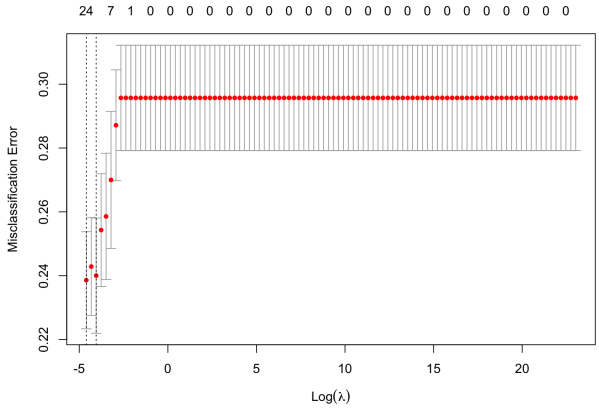
Table 11: Lasso Selection, Optimal λ Confusion Matrix

		Response Test	
		bad	good
Lasso Pred opt λ	bad	30	23
	good	63	184

Test MSE	$(63+23)/(300) = 28.66\%$

As shown in figure 13 the optimal value for λ was chosen to be 0.01321941 and the model that was fit with that value had the test performance shown in table 11. The error rate for the lasso regression was 28.67 which is similar to the ridge regression indicating similar predictive accuracy. However ridge regression remains more suitable by better controlling multicollinearity.

Figure 13: Lasso Regression Missclassification Error for Different λ



3.5 Logistic regression GAM

Generalised Additive models (GAM) are a flexible approach to model non-linear relationships between the predictors and response. Gam allows for non-linear relationships by implementing smoothing functions on some or all predictors. In classification problems a logistic GAM model is given by 6 where $s_m(Z_m)$ are smoothing functions applied to non linear functions. Natural splines are splines that prevent extreme fluctuations at the boundaries which given the inclusion of outliers in numerical predictors such as duration or amount are the most suitable smoothing function.

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + s_1(Z_1) + s_2(Z_2) + \ldots + s_m(Z_m) \qquad (6)$$

The model smoothed with natural splines on duration and amount as well as the other variables selected in the best AIC model has the test performance as in table 12. The test error rate of 28.33% shows that while the model fails to improve upon accuracy over the ridge regression, the GAM has made the model less easily interpret-able and therefore less preferable.

Table 12: GAM with Variables from AIC subset selection Confusion Matrix

		Response Test	
		bad	good
GAM Pred	bad	17	9
	good	76	198

Test MSE	$(76+9)/(300) = 28.33\%$

3.6 Regression Tree

Regression trees are a type of decision tree used to capture non linear relationships and interactions. Regression trees split the data into subsets based on the predictor values. The splits minimize the variance of the response variable within each subset. Each node represents a decision splitting the dataset in two or more sets creating further branches. The terminal nodes of regression trees represent the mean of the response variable. To prevent over fitting pruning is used to reduce the size of the tree by eliminating branches using features that have little predictive power. Using cross validation to find the optimal pruning of the tree we find the best prune parameter to be 6. The structure of the pruned tree is shown by figure 14.

Figure 14: Pruned (6) Regression Tree Structure

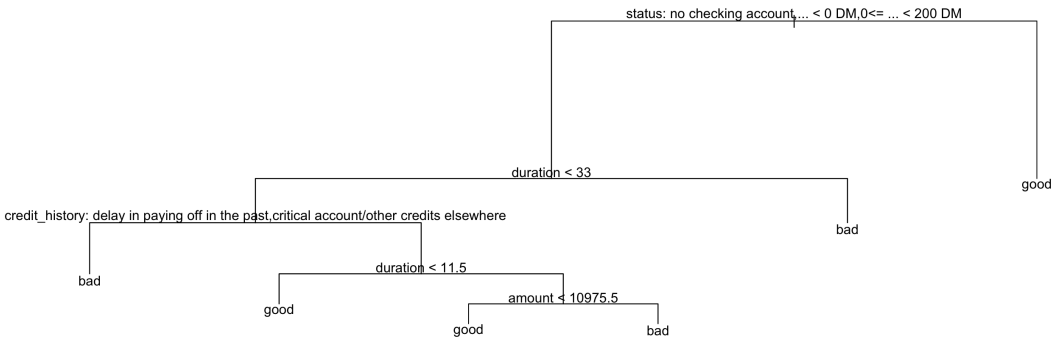


Table 13: Pruned Regression Tree Confusion Matrix

		Response Test	
		bad	good
Pruned Tree Pred	bad	29	26
	good	64	181

Test MSE	$(64 + 26)/(300) = 30\%$

The pruned tree was constructed with variables such as 'status','duration','credit_history' and 'amount'. The test performance when evaluated on the test date is as shown in table 13. While the pruned tree already offers benefits in being able to visualise and better interpret the decision making process, the error rate of 30% indicates that there is room for improvement with more advanced tree-based methods.

3.6.1 Boosting

Boosting is an ensemble method that combines multiple weaker decision trees to create a stronger model. It sequentially adds trees where each new tree corrects errors made by previous ones and it is effective in reducing bias and variance leading to higher accuracy in predictions. The boosted model is tuned to 3 parameters: the number of trees, interaction depth and shrinkage which control the model complexity. The best model is then chosen on the lowest error rate for the given parameters.

The optimal parameters were found to be 200 trees, interaction depth of 3 and shrinkage parameter of 0.15. Based on this the model preformed as shown in table 14 with a test error rate of 23.67%. Boosting seems to be the most effective method for the classification task so far due to its ability to combile multiple simple models which makes it strong with complex datasets.

Table 14: Boosted Tree Confusion Matrix

		Response Test	
		bad	good
Boost Pred	bad	34	12
	good	59	195

Test MSE	$(59 + 12)/(300) = 23.66\%$

Figure 15: Relative Influence in Boosting Algorithm

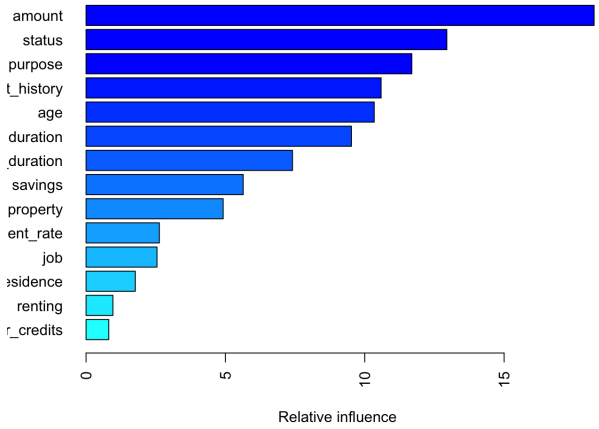
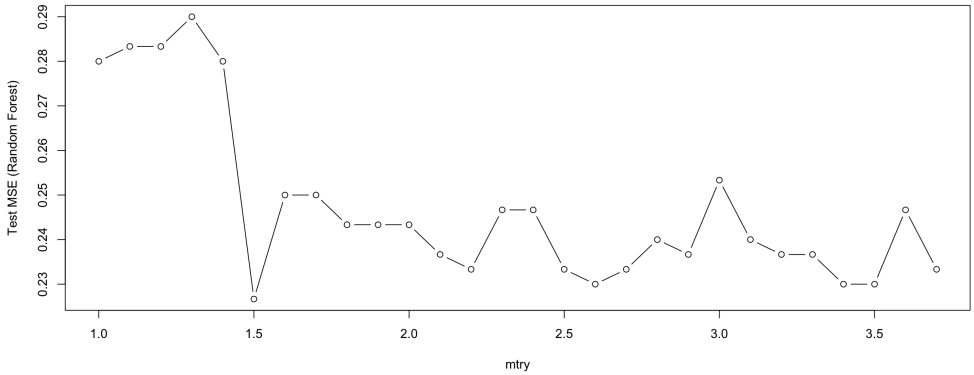


Figure 16: Random Forest: Optimal mtry Value



Another benefit to boosting is the ease at which one can interpret the influence of variables. Relative influence as shown in figure 15. Relative influence in boosting algorithms is assesed on how often variables are used for splitting across boosting stages and the impact those splits have on reducing the error.

3.6.2 Random Forest

Random Forests are an ensemble method that constructs a multitude of decision trees then in a classification problem will predict the modal class for a given observation. The random forest algorithm can be tuned to number of variables at each split in a classification problem is given as the square root of the total amount -1. Using 500 trees, the optimal value of variables as shown in figure 16 was found to be 1.5.

Table 15: Random Forest best mtry confusion matrix

		Response Test	
		bad	good
RF Pred	bad	29	13
	good	64	194

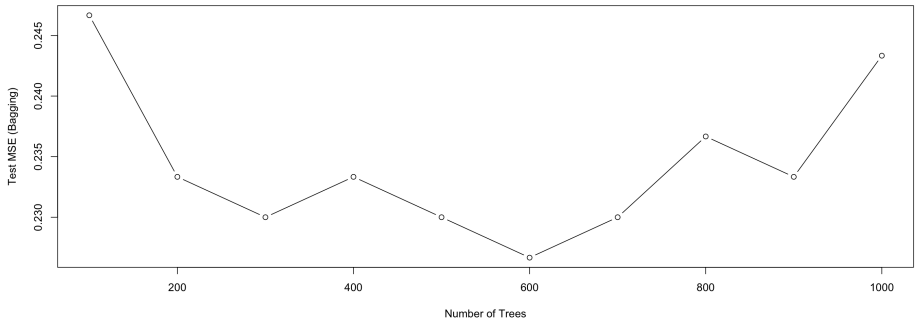
Test MSE	$(64 + 13)/(300) = 25.66\%$

Setting the mtry tuning parameter to 1.5, the model had an out-of-bag error rate of 24% which is a measure of training performance. The test performance is slightly higher at 25.66% which is worse than the boosting algorithm but better than the decision tree or any other method so far. Random Forest's strength comes from the fact it averages multiple trees and therefore is less prone to over fitting compared to individual trees.

3.6.3 Bagging

Boostrap Aggregating (Bagging) improves the stability and accuracy of classification algorithms while reducing variance to prevent over fitting. Bagging creates several versions of the model trained on random subsets of the training dataset then aggregates the predictions to produce a single result. The Bagging algorithm was tuned to find the optimal number of trees in the range of 100 to 1000 with the mtry value set to the number of predictors. The optimal amount of trees as shown in figure 17 is found to be 600.

Figure 17: Bagging : Optimal number of trees



The error rate of bagging is shown to be 24% This is a great improvement over non-ensemble methods and is indication that the ability bagging to integrate predictions from numerous models similar to how other ensemble methods work are a great strength in dealing with the credit risk clasification problem and dataset.

Table 16: Tuned Bagging (Tree =600) Confusion Matrix

		Response Test	
		bad	good
Bagging	bad	42	21
	good	51	186

Test MSE	$(21 + 51)/(300) = 24.0\%$

4 Conclusion

While the methods so far have been evaluated on test MSE, the application of the models have to factor in to the composition of test error rates. For a loan manager, a false positive in which they would approve an application that ends up being classified as bad credit and is not repayed has a much higher cost. For a unit loan amount a default means that the profit is -1. Therefore it is important to explore which models have lower false positive rates. If we set an arbitrary expected return on good loans to be 35% then the per unit profit for a loan manager that approves all applications will be given as in equation 7

$$70\%(0.35) + 30\%(-1) = -0.055. \tag{7}$$

To fully compare performance the following metrics have been calculated in addition to test MSE. Table 17 reveals all these metrics for each discussed machine learning classification algorithms. One should note that all the methods are shown to increase the per unit profit (technically a loss) of the loan manager with no model.

- Per Unit Profit = True Positive Rate(0.35) + False Positive Rate(-1)
- Mean Approved Amount = Average Loan Amount for the observations predicted as good in test set
- Per Applicant Profit = Mean Approved Amount $*$ Per Unit Profit
- Overall Test profit = Per Applicant Profit $*$ Amount of approved loans
- Opportunity Cost = Per Applicant Profit $*$ Amount of False Negatives

Table 17: Results table

	Prediction_Method	Test_MSE	Per_Unit_Profit	Per_Applicant_Profit	Mean_Approved_Amount	Profit_Overall	Opportunity_Cost
1	Full Logistic Model	0.27	-0.01	-34.35	3075.82	-9342.28	-274.77
2	LDA Classifier, Threshold = 0.3	0.28	-0.01	-38.52	3081.25	-10283.68	-462.19
3	LDA Classifier, threshold = Prior Prob	0.27	0.03	86.61	2887.06	20267.16	2338.52
4	QDA, threshold = Prior Prob	0.30	0.03	73.31	2819.58	16054.66	2859.05
5	QDA, threshold = 0.3	0.29	0.03	70.76	2811.62	15991.55	2405.81
6	KNN, K = 40	0.31	-0.06	-199.15	3273.68	-58748.94	-398.30
7	Subset Selection: BIC	0.30	-0.04	-121.66	3215.60	-34063.92	-973.25
8	Subset Selection: AIC	0.29	-0.02	-72.31	3076.82	-19739.32	-723.05
9	Ridge: Best lambda	0.28	0.02	48.91	2905.69	11836.81	1173.90
10	Ridge: Best lambda, 1se	0.29	-0.03	-82.17	3062.36	-22515.47	-821.73
11	Lasso: Best lambda	0.29	0.00	1.01	3029.63	253.48	21.21
12	Sparse Lasso: best lambda with 1se	0.30	-0.03	-75.24	3009.67	-19863.83	-1279.11
13	GAM(AIC) Logistic Model	0.28	-0.02	-68.06	3047.68	-18649.74	-612.58
14	Classification Tree	0.30	-0.00	-6.11	2818.60	-1496.21	-158.78
15	Pruned Classification Tree	0.30	-0.00	-6.11	2818.60	-1496.21	-158.78
16	Boosting, Tuned	0.24	0.03	93.03	3017.28	23630.30	1116.39
17	Bagging, Best number of trees	0.24	0.05	137.36	2922.54	32554.22	2884.55
18	Random Forest: Best mtry	0.26	0.01	38.81	2985.01	10011.72	504.47

It is clear that while improvements in the overall test error may be marginal when comparing two methods, the implication of how this error is distributed between false positives and false negatives is huge. Ensemble methods in particular boosting and bagging proved to be the most effective in both reducing the MSE but also getting the lowest false positive rate. This is because the use of multiple models in producing a final classifier algorithm results balanced due to their ability to reduce variance and bias in predictions. These methods are also prone to over fitting and when dealing asymmetric dataset where the undesirable response is the minority is very important.

References

Fitch Ratings. (2023, December). Default rates to rise in u.s. and europe as weaker growth offsets rate cuts. https://www.fitchratings.com/research/corporate-finance/default-rates-to-rise-in-us-europe-as-weaker-growth-offsets-rate-cuts-27-12-2023

South German Credit [DOI: https://doi.org/10.24432/C5X89F]. (2019). https://archive.ics.uci.edu/dataset/522/south+german+credit

5 Appendix

A Logistic Regression Coefficients

Table 18: Coefficients of Logistic Regression on All Variables

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-1.1257	0.8718	-1.29	0.1966
status... < 0 DM	0.4296	0.2521	1.70	0.0884
status0<= ... < 200 DM	0.4437	0.4380	1.01	0.3111
status... >= 200 DM / salary for at least 1 year	1.6981	0.2787	6.09	1.1e-09
duration	-0.0324	0.0114	-2.83	0.0047
credit_historycritical account/other credits elsewhere	-0.0645	0.6421	-0.10	0.9199
credit_historyno credits taken/all credits paid back duly	1.0583	0.5219	2.03	0.0426
credit_historyexisting credits paid back duly till now	1.0273	0.5740	1.79	0.0735
credit_historyall credits at this bank paid back duly	2.0465	0.5406	3.79	0.0002
purposecar (new)	1.4569	0.4271	3.41	0.0006
purposecar (used)	0.4695	0.3159	1.49	0.1373
purposefurniture/equipment	0.6189	0.2888	2.14	0.0321
purposedomestic appliances/radio/television	-0.1264	0.5199	-0.24	0.8080
purposerepairs	-0.0080	0.5016	-0.02	0.9873
purposebusiness/retraining	0.6029	0.3841	1.57	0.1165
amount	-0.0001	0.0001	-1.57	0.1155
savings... < 100 DM	0.1389	0.3297	0.42	0.6734
savings100 <= ... < 1000 DM	0.4276	0.3877	1.10	0.2701
savings... >= 1000 DM	0.9030	0.3098	2.91	0.0036
employment_duration< 1 yr	-0.1850	0.4917	-0.38	0.7068
employment_duration1 <= ... < 4 yrs	0.0962	0.4647	0.21	0.8360
employment_duration4 <= ... < 7 yrs	0.5022	0.5129	0.98	0.3276
employment_duration>= 7 yrs	0.0859	0.4561	0.19	0.8506
installment_rate.L	-0.6446	0.2517	-2.56	0.0104
installment_rate.Q	-0.0320	0.2401	-0.13	0.8940
installment_rate.C	0.0394	0.2443	0.16	0.8719
present_residence.L	-0.0640	0.2460	-0.26	0.7948
present_residence.Q	0.2523	0.2327	1.08	0.2783
present_residence.C	0.0073	0.2295	0.03	0.9748
propertycar or other	0.1572	0.3019	0.52	0.6025
propertybuilding soc. savings agr./life insurance	-0.0154	0.2733	-0.06	0.9552
propertyreal estate	-0.3385	0.3889	-0.87	0.3841
age	0.0185	0.0106	1.75	0.0807
number_credits.L	-0.3492	0.1996	-1.75	0.0802
jobskilled employee/official	-0.0765	0.2657	-0.29	0.7735
jobmanager/self-empl./highly qualif. employee	-0.3719	0.3720	-1.00	0.3175
rentingno	-0.1818	0.2610	-0.70	0.4860

B Best BIC subset selection Coefficients

Table 19: Subset selection with BIC logistic model coefficients

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.5292	0.5047	-1.05	0.2943
credit_historycritical account/other credits elsewhere	0.3873	0.5875	0.66	0.5097
credit_historyno credits taken/all credits paid back duly	1.4673	0.4679	3.14	0.0017
credit_historyexisting credits paid back duly till now	1.1512	0.5477	2.10	0.0356
credit_historyall credits at this bank paid back duly	2.1350	0.4974	4.29	0.0000
duration	-0.0384	0.0079	-4.87	0.0000
status... < 0 DM	0.4428	0.2265	1.95	0.0506
status0<= ... < 200 DM	0.5222	0.4017	1.30	0.1936
status... >= 200 DM / salary for at least 1 year	1.8409	0.2498	7.37	0.0000

C Best AIC coefficients

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-1.1102	0.6297	-1.76	0.0779
number_credits.L	-0.2791	0.1936	-1.44	0.1494
age	0.0163	0.0091	1.79	0.0735
installment_rate.L	-0.7315	0.2406	-3.04	0.0024
installment_rate.Q	-0.0640	0.2327	-0.27	0.7834
installment_rate.C	0.0439	0.2401	0.18	0.8549
savings... < 100 DM	0.1691	0.3205	0.53	0.5979
savings100 <= ... < 1000 DM	0.4065	0.3805	1.07	0.2854
savings... >= 1000 DM	0.9354	0.3046	3.07	0.0021
amount	-0.0001	0.0001	-2.57	0.0102
purposecar (new)	1.3223	0.4167	3.17	0.0015
purposecar (used)	0.4266	0.3054	1.40	0.1625
purposefurniture/equipment	0.5926	0.2802	2.11	0.0344
purposedomestic appliances/radio/television	-0.1134	0.5075	-0.22	0.8232
purposerepairs	-0.1087	0.4821	-0.23	0.8217
purposebusiness/retraining	0.6526	0.3754	1.74	0.0822
credit_historycritical account/other credits elsewhere	-0.0287	0.6256	-0.05	0.9634
credit_historyno credits taken/all credits paid back duly	1.1606	0.5080	2.28	0.0223
credit_historyexisting credits paid back duly till now	1.0085	0.5586	1.81	0.0710
credit_historyall credits at this bank paid back duly	2.1190	0.5216	4.06	0.0000
duration	-0.0268	0.0109	-2.46	0.0137
status... < 0 DM	0.3800	0.2451	1.55	0.1210
status0<= ... < 200 DM	0.3500	0.4272	0.82	0.4127
status... >= 200 DM / salary for at least 1 year	1.6745	0.2696	6.21	0.0000

D R Code

```
library(DataExplorer)#introduce
#library(ggplot)
library(egg) #ggarange function
library(MASS)#lda
library(class)
library(bestglm)#bestglm
library(glmnet)#ridge and lasso
library(gam) #gam
library(tree) #tree
library(gbm) #gbm
library(randomForest) #forest
library(boot)#cv.glm
library(ggplot2)


#LOAD DATA
dat <- read.table("SouthGermanCredit.asc", header=TRUE)

## dat contains numbers for all variables.
## variables duration, amount and age are truly quantitative
## variables installment_rate, present_residence and number_credits are
### quantitative in the data, but are in fact discretized scores for
### an underlying quantitative variable
### and are thus stored as ordered factors below
## variable people_liable is quantitative in the data but is in fact
### a binarized score (less 0 to 2 versus 3 or more)
### and is thus stored as a factor below
## all the numeric values (=level codes)
### for the categorical variables
### (including the discretized quantitative variables),
### are the P2 scores from Häußler (1979)
### which can be directly used in credit scoring (larger=better).
### (Exceptions have been corrected in the raw data,
###     which implies that columns pers and gastarb have
###     entries opposite to those in Open Data LMU (2010)
###     and the GermanCredit data from the UCI ML Repo.)
```

```
## variable names from Fahrmeir/Hamerle book
nam_fahrmeirbook <- colnames(dat)

### assign levels
### level assignment can be sanity-checked
### with Table 2.1 from the Fahrmeir/Hamerle book,
###     which gives proportions separated for good and bad credit risks.
### That table is provided with by Open Data LMU
###     (https://doi.org/10.5282/ubm/data.23)
###     together with a German language version of the data set.
### A corresponding table for the English language data is produced
###     below for the final data (levels ordered by increasing code).
### Level labels have been taken from package evtree, except for
###     the variable telephone (where the yes level has been made more detailed)
###     and those variables that were quantitative and do not have level labels
###     in evtree.

nam_evtree <- c("status", "duration", "credit_history", "purpose", "amount",
                "savings", "employment_duration", "installment_rate",
                "personal_status_sex", "other_debtors",
                "present_residence", "property",
                "age", "other_installment_plans",
                "housing", "number_credits",
                "job", "people_liable", "telephone", "foreign_worker",
                "credit_risk")
names(dat) <- nam_evtree

## make factors for all except the numeric variables
## make sure that even empty level of factor purpose = verw (dat[[4]]) is included
for (i in setdiff(1:21, c(2,4,5,13)))
  dat[[i]] <- factor(dat[[i]])
## factor purpose
dat[[4]] <- factor(dat[[4]], levels=as.character(0:10))

## assign level codes
## make intrinsically ordered factors into class ordered
levels(dat$credit_risk) <- c("bad", "good")
levels(dat$status) = c("no checking account",
```

```
                       "... < 0 DM",
                       "0<= ... < 200 DM",
                       "... >= 200 DM / salary for at least 1 year")
## "critical account/other credits elsewhere" was
## "critical account/other credits existing (not at this bank)",
levels(dat$credit_history) <- c(
  "delay in paying off in the past",
  "critical account/other credits elsewhere",
  "no credits taken/all credits paid back duly",
  "existing credits paid back duly till now",
  "all credits at this bank paid back duly")
levels(dat$purpose) <- c(
  "others",
  "car (new)",
  "car (used)",
  "furniture/equipment",
  "radio/television",
  "domestic appliances",
  "repairs",
  "education",
  "vacation",
  "retraining",
  "business")
levels(dat$savings) <- c("unknown/no savings account",
                         "... <  100 DM",
                         "100 <= ... <  500 DM",
                         "500 <= ... < 1000 DM",
                         "... >= 1000 DM")
levels(dat$employment_duration) <-
  c(  "unemployed",
      "< 1 yr",
      "1 <= ... < 4 yrs",
      "4 <= ... < 7 yrs",
      ">= 7 yrs")
dat$installment_rate <- ordered(dat$installment_rate)
levels(dat$installment_rate) <- c(">= 35",
                                  "25 <= ... < 35",
                                  "20 <= ... < 25",
                                  "< 20")
```

```
levels(dat$other_debtors) <- c(
  "none",
  "co-applicant",
  "guarantor"
)
## female : nonsingle was female : divorced/separated/married
##    widowed females are not mentioned in the code table
levels(dat$personal_status_sex) <- c(
  "male : divorced/separated",
  "female : non-single or male : single",
  "male : married/widowed",
  "female : single")
dat$present_residence <- ordered(dat$present_residence)
levels(dat$present_residence) <- c("< 1 yr",
                                   "1 <= ... < 4 yrs",
                                   "4 <= ... < 7 yrs",
                                   ">= 7 yrs")
## "building soc. savings agr./life insurance",
##    was "building society savings agreement/life insurance"
levels(dat$property) <- c(
  "unknown / no property",
  "car or other",
  "building soc. savings agr./life insurance",
  "real estate"
)
levels(dat$other_installment_plans) <- c(
  "bank",
  "stores",
  "none"
)
levels(dat$housing) <- c("for free", "rent", "own")
dat$number_credits <- ordered(dat$number_credits)
levels(dat$number_credits) <- c("1", "2-3", "4-5", ">= 6")
## manager/self-empl./highly qualif. employee  was
##   management/self-employed/highly qualified employee/officer
levels(dat$job) <- c(
  "unemployed/unskilled - non-resident",
  "unskilled - resident",
  "skilled employee/official",
```

```
  "manager/self-empl./highly qualif. employee"
)
levels(dat$people_liable) <- c("3 or more", "0 to 2")
levels(dat$telephone) <- c("no", "yes (under customer name)")
levels(dat$foreign_worker) <- c("yes", "no")

## checks against fahrmeir table
tabs <-
  list(status = round(100*prop.table(xtabs(~status+credit_risk, dat),2),2),
       credit_history = round(100*prop.table(xtabs(~credit_history+credit_risk, dat),2),2),
       purpose = round(100*prop.table(xtabs(~purpose+credit_risk, dat),2),2),
       savings = round(100*prop.table(xtabs(~savings+credit_risk, dat),2),2),
       employment_duration = round(100*prop.table(xtabs(~employment_duration+credit_risk, dat),2),2),
       installment_rate = round(100*prop.table(xtabs(~installment_rate+credit_risk, dat),2),2),
       personal_status_sex = round(100*prop.table(xtabs(~personal_status_sex+credit_risk, dat),2),2),
       other_debtors = round(100*prop.table(xtabs(~other_debtors+credit_risk, dat),2),2),
       present_residence = round(100*prop.table(xtabs(~present_residence+credit_risk, dat),2),2),
       property = round(100*prop.table(xtabs(~property+credit_risk, dat),2),2),
       other_installment_plans = round(100*prop.table(xtabs(~other_installment_plans+credit_risk, dat),2),2),
       housing = round(100*prop.table(xtabs(~housing+credit_risk, dat),2),2),
       number_credits = round(100*prop.table(xtabs(~number_credits+credit_risk, dat),2),2),
       job = round(100*prop.table(xtabs(~job+credit_risk, dat),2),2),
       people_liable = round(100*prop.table(xtabs(~people_liable+credit_risk, dat),2),2),
       telephone = round(100*prop.table(xtabs(~telephone+credit_risk, dat),2),2),
       foreign_worker = round(100*prop.table(xtabs(~foreign_worker+credit_risk, dat),2),2))

## variables for which a tab entry is available
## (all except 2, 5 and 13)
tabwhich <- setdiff(1:20, c(2,5,13))


set.seed(10)

german_credit <- dat
attach(german_credit)

#Data Exploration
head(german_credit,5)
str(german_credit)
```

```
metadata<-t(introduce(german_credit))
colnames(metadata)<-"Info"

#xtable(metadata) for latex
plot_intro(german_credit,title = "South German Credit Data Integrity")

#Plots for Continuous Variables
#duration
g1 <- ggplot(german_credit, aes(x = credit_risk, y = duration, fill = credit_risk)) +
  geom_boxplot() +
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black", position=position_dodge(0.75)) +
  theme(legend.position = "none")
g1.2 <- ggplot(german_credit, aes(sample = duration)) +
  stat_qq() + geom_qq_line(color = "red", linetype = "dashed") + labs(x = "Theoretical",y = "Sample")
#amount
g1.3<-ggplot(german_credit, aes(x = credit_risk, y = amount, fill = credit_risk)) + geom_boxplot() +
  stat_summary(fun=mean, geom="point", shape=18, size=3, color="black", position=position_dodge(0.75)) +
  theme(legend.position = "none")
g1.4 <- ggplot(german_credit, aes(sample = amount)) +
  stat_qq() + geom_qq_line(color = "red", linetype = "dashed") + labs(x = "Theoretical",y = "Sample")
#AGE
g1.5<-ggplot(german_credit, aes(x =credit_risk, y = age, fill = credit_risk)) +
  geom_boxplot() + stat_summary(fun=mean, geom="point", shape=18, size=3, color="black", position=position_dodge(0.75)) +
  theme(legend.position = "none")
g1.6 <- ggplot(german_credit, aes(sample = age)) +
  stat_qq() + geom_qq_line(color = "red", linetype = "dashed") + labs(x = "Theoretical",y = "Sample")

ggarrange(g1,g1.3,g1.5, g1.2,g1.4,g1.6,
          labels = c("Class Distribution", "","", "Normal Probability Plot","", ""),
          ncol = 3, nrow = 2)

#Plots for Categorical and Binary Variables
g3<-ggplot(german_credit, aes(status, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g4<-ggplot(german_credit, aes(credit_history, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g5<-ggplot(german_credit, aes(purpose, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")
```

```
g6<-ggplot(german_credit, aes(savings, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g7<-ggplot(german_credit, aes(employment_duration, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g8<-ggplot(german_credit, aes(installment_rate, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g9<-ggplot(german_credit, aes(personal_status_sex, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g10<-ggplot(german_credit, aes(other_debtors, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g11<-ggplot(german_credit, aes(present_residence, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g12<-ggplot(german_credit, aes(property, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g13<-ggplot(german_credit, aes(other_installment_plans, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g14<-ggplot(german_credit, aes(housing, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g15<-ggplot(german_credit, aes(number_credits, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g16<-ggplot(german_credit, aes(job, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g17<-ggplot(german_credit, aes(people_liable, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g18<-ggplot(german_credit, aes(telephone, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

g19<-ggplot(german_credit, aes(foreign_worker, ..count..)) + geom_bar(aes(fill = credit_risk), position = "dodge")

#Plots
ggarrange(g3, g4, g5, labels = c("3.1: status ", "3.2: credit_history","3.3: purpose"),ncol = 1, nrow = 3)
ggarrange(g6, g7, g8, labels = c("4.1 savings ", "4.2: employment_duration","4.3: installment_rate"),ncol = 1, nrow = 3)
ggarrange(g9, g10, g11, labels = c("5.1: personal_status_sex ", "5.2: other_debtors","5.3: present_residence"),ncol = 1, nrow = 3)
ggarrange(g12, g13, g14, labels = c("6.1: property ", "6.2: other_installment_plans","6.3: housing"),ncol = 1, nrow = 3)
ggarrange(g15, g16, g17, labels = c("7.1: number_credits ", "7.2: job","7.3: people_liable"),ncol = 1, nrow = 3)
ggarrange(g18, g19, labels = c("8.1: telephone ", "8.2: foreign_worker"),ncol = 1, nrow = 2)

#Excluding
#Table to inspect which to exclude
factor_variable_names <- names(Filter(is.factor, german_credit))
```

```
# Create a list to store proportions
proportions_list <- list()
# Loop through the factor variables and populate the list
for (variable in factor_variable_names) {
  proportions <- prop.table(table(german_credit[, variable])) * 100
  proportions_list[[variable]] <- proportions
}
proportions_list
# Print the resulting list

#Exclude :
#personal_status_sex #Drop for data issues
#foreign_worker #proportions
#telephone #almost everyone now will have
#other_debtors #
#People liable
#other_installment_plans#

german_credit <- german_credit[, !(names(german_credit) %in% c("personal_status_sex", "foreign_worker", "telephone", "other_debtors","people_li

#Amend
#Housing
german_credit$renting <- ifelse(german_credit$housing %in% c("for free", "own"), "no", "yes")
german_credit <- german_credit[, !(names(german_credit) %in% "housing")] # Remove the original "housing" variable
german_credit$renting <- factor(german_credit$renting, levels = c("yes", "no")) # Convert the new "renting" variable to a factor
summary(german_credit$renting)

#Number of credits join 4-5 and >6 and just make it >= 6
levels(german_credit$number_credits) <- ifelse(levels(german_credit$number_credits) %in% c("2-3","4-5", ">= 6"),
                                               ">=2",
                                               levels(german_credit$number_credits))

# Assuming 'job' is a factor variable
levels(german_credit$job) <- ifelse(levels(german_credit$job) %in% c("unemployed/unskilled - non-resident", "unskilled - resident"),
                                    "unskilled/Unemployed",
                                    levels(german_credit$job))

#Education
sum(german_credit$purpose == "education") #remove educaiton as purpose
```

```
levels(german_credit$purpose) <- ifelse(levels(german_credit$purpose) %in% c("education", "others"),
                                        "others",
                                        levels(german_credit$purpose))

str(german_credit)
#Join domestic appliances and tv/radio and furniture / equipment
levels(german_credit$purpose) <- ifelse(levels(german_credit$purpose) %in% c("domestic appliances", "radio/television"),
                                        "domestic appliances/radio/television",
                                        levels(german_credit$purpose))
#Join business and training
levels(german_credit$purpose) <- ifelse(levels(german_credit$purpose) %in% c("business", "retraining"),
                                        "business/retraining",
                                        levels(german_credit$purpose))

levels(german_credit$purpose) <- ifelse(levels(german_credit$purpose) %in% c("vacation"),
                                        "others",
                                        levels(german_credit$purpose))

#Savings Join 100< 500 and 500 < 1000
levels(german_credit$savings) <- ifelse(levels(german_credit$savings) %in% c("100 <= ... <  500 DM", "500 <= ... < 1000 DM"),
                                        "100 <= ... < 1000 DM",
                                        levels(german_credit$savings))

german_credit <- german_credit[, c("credit_risk", setdiff(names(german_credit), "credit_risk"))]

str(german_credit)

### Part 1 ###
train <- sample(1:1000,700)

gc_train <- german_credit[train,]
gc_test <- german_credit[-train,]

response_train <- gc_train$credit_risk
response_test <- gc_test$credit_risk

#since data has assymettric distribution, set cutt-off probability to 30%

#Logistic Regression
```

```
log.fit <- glm(credit_risk ~., data = gc_train, family = binomial)
summary(log.fit)

log.prob <- predict(log.fit, gc_test, type = "response") #Calculate probabilities

log.pred <- rep("bad", length(response_test))
log.pred[log.prob > 0.3] <- "good"
table(log.pred, response_test)
mean(log.pred != response_test) # overall error rate = 0.27

#LOOCV
cost.fn <- function(r, pi = 0) mean((r==1 & pi<0.3) | (r==0 & pi>0.3))
cv.glm(gc_train,log.fit,cost=cost.fn)$delta[1] # Log LOOCV = 0.255 #Might be on full dataset

#Part 2
#LDA
lda.fit <- lda(credit_risk ~ ., data = gc_train)

lda.prob <- predict(lda.fit , gc_test)

# For a 0.3 cut-off
lda.pred <- rep("bad",length(response_test))
lda.pred[lda.prob$posterior[,2] > 0.3] <- "good"
mean(lda.pred != response_test) # error rate = 0.28

#LDA default prior probability
mean(lda.prob$class != response_test) #0.27

#QDA
qda.fit <- qda(credit_risk ~., data = gc_train)
qda.fit
qda.prob <- predict(qda.fit , gc_test)

# For a 0.3 cut-off
qda.pred <- rep("bad",length(response_test))
qda.pred[qda.prob$posterior[,2] > 0.3] <- "good"
table(qda.pred,response_test)
mean(qda.pred != response_test) # error rate = 0.29
```

```
mean(qda.prob$class != response_test)#0.3

#KNN
attach(german_credit)
train.X=cbind(status,duration,credit_history,purpose,amount,savings,employment_duration,installment_rate,present_residence,property,age,number_
test.X=cbind(status,duration,credit_history,purpose,amount,savings,employment_duration,installment_rate,present_residence,property,age,number_c

set.seed(10)
# Set the range of k values to test
k_values <- seq(10, 200, by = 5)

# Create a data frame to store the results
results <- data.frame(k = k_values, error_rate = rep(NA, length(k_values)))

# Define the control parameters for cross-validation
ctrl <- trainControl(method = "cv",   # Use cross-validation
                     number = 10)      # Number of folds

# Train the kNN model using cross-validation
KNN <- train(x = train.X,
             y = response_train,
             method = "knn",
             trControl = ctrl,
             tuneGrid = data.frame(k = k_values))

# Store the cross-validated error rates in the results data frame
results$error_rate <- 1 - KNN$results$Accuracy

# Plot the results with k on the x-axis and error_rate on the y-axis
plot(results$k, results$error_rate, type = "b", pch = 19, col = "blue",
     xlab = "k", ylab = "Error Rate", main = "Cross-Validated Error Rates for Different Values of k")

# Add a vertical line at the optimal k
abline(v = results$k[which.min(results$error_rate)], col = "red", lty = 2)

set.seed(10)
knn.pred1 <- knn(train=train.X, test=test.X, cl=response_train, k = 40)
table(knn.pred1, response_test)
mean(knn.pred1 != response_test) # Error rate =  0.2966
```

```
#Part 5
#BIC
bestlogits <- bestglm(rev(gc_train), IC = "BIC",family = binomial)
summary(bestlogits$BestModel) #this is logit regression of best model
bestlog.prob <- predict(bestlogits$BestModel, rev(gc_test), type = "response")

bestlogits$Subsets

bestlog.pred <- rep("bad", length(response_test))
bestlog.pred[bestlog.prob > 0.3] <- "good"
table(bestlog.pred, response_test)
mean(bestlog.pred !=  response_test) # overall error rate = 0.29666

plot(x = 1:15, y = bestlogits$Subsets$BIC, xlab = "Number of Variables", ylab = "BIC", main = "BIC", pch = 19, col = "blue")
abline(h = min(bestlogits$Subsets$BIC), col = "red", lty = 2)# Adding a line to highlight the minimum value
legend("top", legend = "Minimum Value", col = "red", lty = 2, cex = 0.8,) # Adding a line to highlight the minimum value

#AIC
bestaic <- bestglm(rev(gc_train),IC="AIC",family = binomial)
summary(bestaic$BestModel)
bestaic.prob <-  predict(bestaic$BestModel, rev(gc_test), type = "response")

bestaic.pred <- rep("bad", length(response_test))
bestaic.pred[bestaic.prob > 0.3] <- "good"
table(bestaic.pred, response_test)
mean(bestaic.pred !=  response_test) # overall error rate = 0.33

plot(x = 1:15, y = bestaic$Subsets$AIC, xlab = "Number of Variables", ylab = "AIC", main = "AIC", pch = 19, col = "blue")
abline(h = min(bestaic$Subsets$AIC), col = "red", lty = 2)# Adding a line to highlight the minimum value
legend("top", legend = "Minimum Value", col = "red", lty = 2, cex = 0.8,) # Adding a line to highlight the minimum value

#Part 5
#ridge
set.seed(1)
x <- model.matrix(credit_risk~.,data = gc_train)[,-1]
y <- response_train
newx <- model.matrix(credit_risk~.,data = gc_test)[,-1]
```

```
grid <- 10^seq(10, -2, length = 100)

ridge.mod <- glmnet(x,y,alpha = 0,family = "binomial",lambda = grid) #ridge
ridge.cv <- cv.glmnet(x,y,alpha = 0,family = "binomial",lambda = grid,type.measure = "class")
plot(ridge.cv)

ridge.bestlam <- ridge.cv$lambda.min
ridge.bestlam
ridge.pred <- predict(ridge.mod, s = ridge.bestlam, newx = newx,type = 'class')

table(ridge.pred, response_test)
mean(ridge.pred != response_test)#0.285

#1SE
ridge.1se <- ridge.cv$lambda.1se
ridge.1se
ridge.pred2 <- predict(ridge.mod, s = ridge.1se, newx = newx,type = 'class')
table(ridge.pred2, response_test)
mean(ridge.pred2 != response_test)#0.33

#Lasso
set.seed(10)
lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid,family = 'binomial')
lasso.cv <- cv.glmnet(x, y, alpha = 1, lambda = grid,family='binomial',type.measure = "class")

plot(lasso.cv)

lasso.bestlam <- lasso.cv$lambda.min
lasso.bestlam #0.01
lasso.pred <- predict(lasso.mod, s = lasso.bestlam, newx = newx,type = 'class')
table(lasso.pred, response_test)
mean(lasso.pred != response_test)#0.2866

# Most sparse lasso model with 1 se
lasso.bestlam2 <- lasso.cv$lambda.1se
lasso.bestlam2 #0.017
lasso.pred2 <- predict(lasso.mod, s = lasso.bestlam2, newx = newx,type = 'class')
mean(lasso.pred2 != response_test)#0.34
```

```
#GAM
selected_columns <- names(bestaic$BestModels)[as.logical(bestaic$BestModels[1,])]
selected_columns <- selected_columns[selected_columns != "Criterion"]
n <- selected_columns

gamtrain <- gc_train
gamtest <- gc_test
gamtrain$credit_risk <- ifelse(gamtrain$credit_risk == "bad", 0, 1)
gamtest$credit_risk <- ifelse(gamtest$credit_risk == "bad", 0, 1)

cont<-c("duration","credit_risk","age")
f <- as.formula(paste("I(credit_risk) ~ ns(duration) +ns(amount)+", paste(n[!n %in% cont], collapse = " + ")))

logit.gam <- gam(formula= f, data = gamtrain, family =binomial)

summary(logit.gam)

gam.prob <- predict(logit.gam,newdata = gamtest, type = "response")

gam.pred <- rep("bad", length(response_test))
gam.pred[gam.prob > 0.3] <- "good"
table(gam.pred, response_test)
mean(gam.pred != response_test) # overall error rate = 0.286

#Part 7
#Tree
set.seed(10)
tree.credit <- tree(credit_risk ~ ., data = gc_train)
summary(tree.credit)
plot(tree.credit)
text(tree.credit, pretty = 0)

tree.pred <- predict(tree.credit, newdata = gc_test,type = 'class')
table(tree.pred,response_test)
mean(tree.pred != response_test) #0.3


cv.tree <- cv.tree(tree.credit, FUN = prune.misclass)
```

```
par(mfrow = c(1, 2)) #graphs
plot(cv.tree$size, cv.tree$dev, type = "b")
plot(cv.tree$k, cv.tree$dev, type = "b")


par(mfrow = c(1, 1)) #graphs
prune.tree.credit <- prune.misclass(tree.credit, best = 6)#pruned tree
plot(prune.tree.credit)
text(prune.tree.credit, pretty = 0)

tree.pred2 <- predict(prune.tree.credit, gc_test, type = "class")
table(tree.pred2, response_test)
mean(tree.pred2 != response_test) #0.3


#boost
boost.credit <- gbm(credit_risk ~., gamtrain, distribution = "bernoulli", n.trees = 500, interaction.depth = 4)
summary(boost.credit,las=2)

#Tune it
# Define ranges for tuning parameters
n_trees_grid <- seq(100, 1000, by = 100)
interaction_depth_grid <- seq(2, 6, by = 1)
shrinkage_grid <- seq(0.01, 0.2, by = 0.01)

# Initialize a 3D array to store test errors for each combination
test.error <- array(NA, dim = c(length(n_trees_grid), length(interaction_depth_grid), length(shrinkage_grid)))

set.seed(10)
# Loop over each combination of parameters
for (i in 1:length(n_trees_grid)) {
  for (j in 1:length(interaction_depth_grid)) {
    for (k in 1:length(shrinkage_grid)) {
      # Train a boosted model with the current combination
      boost.credit <- gbm(
        credit_risk ~ .,
        data = gamtrain,
        distribution = "bernoulli",
```

```
        n.trees = n_trees_grid[i],
        interaction.depth = interaction_depth_grid[j],
        shrinkage = shrinkage_grid[k]
      )

      # Make predictions on the test set
      pred.credit.test <- predict(boost.credit, gamtest, n.trees = n_trees_grid[i], type = "response")

      # Calculate test error (assuming binary classification)
      test.error[i, j, k] <- mean((pred.credit.test - gamtest$credit_risk)^2)
    }
  }
}

# Find the indices of the minimum test error
min_error_indices <- which(test.error == min(test.error), arr.ind = TRUE)
best_n_trees <- n_trees_grid[min_error_indices[1, 1]]
best_interaction_depth <- interaction_depth_grid[min_error_indices[1, 2]]
best_shrinkage <- shrinkage_grid[min_error_indices[1, 3]]

set.seed(10)
best.boost <- gbm(credit_risk ~ ., data = gamtrain, distribution = "bernoulli", n.trees = best_n_trees, interaction.depth = best_interaction_de
summary(best.boost,las=2)

boost_prob <- predict(best.boost,gamtest, n.trees = best_n_trees, type = "response")#predict with best model one
best.boost.pred <- rep("bad", length(response_test))
best.boost.pred[boost_prob > 0.3] <- "good"
table(best.boost.pred, response_test)
mean(best.boost.pred !=  response_test) # overall error rate = 0.28

#Random Forest
set.seed(10)
credit.rf <- randomForest(as.factor(credit_risk)~., data = gc_train,mtry=sqrt(ncol(gc_train)-1),ntree=500)
credit.rf
#tuned
mtry_values <- seq(1, sqrt(ncol(gc_train) - 1), by = 0.1)  #range of mtry
test.error_rf <- rep(-1, length(mtry_values))  #Initialize a vector to store test errors for each mtry
set.seed(10)
# Loop over each mtry value
```

```
for (i in 1:length(mtry_values)) {
  # Train a random forest model with the current mtry value
  credit.rf <- randomForest(as.factor(credit_risk) ~ ., data = gc_train, mtry = mtry_values[i], ntree = 500)

  # Make predictions on the test set
  pred.credit.test_rf <- predict(credit.rf, gc_test, type = "response")

  test.error_rf[i] <- mean(pred.credit.test_rf != gc_test$credit_risk)

}

# Plot the test errors for each mtry value
plot(mtry_values, test.error_rf, type = "b", xlab = "mtry", ylab = "Test MSE (Random Forest)")

# Find the optimal mtry value and corresponding test error
best_mtry_rf <- mtry_values[which.min(test.error_rf)]
mse_rf <- min(test.error_rf)

best_rf <- randomForest(as.factor(credit_risk)~., data = gc_train,mtry=best_mtry_rf,ntree=500)
best_rf.pred <- predict(best_rf,newdata = gc_test,type='response')
table(best_rf.pred,response_test)
mean(best_rf.pred!= response_test) #0.256

#Bagging

credit.bag <- randomForest(as.factor(credit_risk)~., data = gc_train,mtry=ncol(gamtrain)-1,ntree=500,importance = TRUE)
set.seed(10)

#tuning
num_trees <- seq(100, 1000, by = 100) # Adjust the range and step size as needed
# Initialize a vector to store test errors for each number of trees
test.error_bagging <- rep(-1, length(num_trees))
# Loop over each number of trees
for (i in 1:length(num_trees)) {
  # Train a bagged model with the current number of trees
  bag.credit <- randomForest(as.factor(credit_risk) ~ ., data = gc_train,mtry=ncol(gc_train)-1, ntree = num_trees[i],importance = TRUE)

  # Make predictions on the test set
  pred.credit.test_bagging <- predict(bag.credit, gc_test,type = "response")
```

```
  # Calculate test error (assuming binary classification)
  test.error_bagging[i] <- mean(pred.credit.test_bagging != gc_test$credit_risk)
}
# Plot the test errors for each number of trees
plot(num_trees, test.error_bagging, type = "b", xlab = "Number of Trees", ylab = "Test MSE (Bagging)")
# Find the optimal number of trees and corresponding test error
best_num_trees_bagging <- num_trees[which.min(test.error_bagging)]
mse_bagging <- min(test.error_bagging)

best.bag <- randomForest(as.factor(credit_risk)~., data = gc_train,mtry=ncol(gc_train)-1,ntree=best_num_trees_bagging,importance = TRUE)

best.bag.pred <- predict(best.bag,newdata = gc_test,type='response')
table(best.bag.pred,response_test)
mean(best.bag.pred!= response_test) #0.24

credit.bag <- randomForest(as.factor(credit_risk)~., data = gc_train,mtry=ncol(gamtrain)-1,ntree=500,importance = TRUE)
set.seed(10)
#tuning
num_trees <- seq(100, 1000, by = 100) # Adjust the range and step size as needed
# Initialize a vector to store test errors for each number of trees
test.error_bagging <- rep(-1, length(num_trees))
# Loop over each number of trees
for (i in 1:length(num_trees)) {
  # Train a bagged model with the current number of trees
  bag.credit <- randomForest(as.factor(credit_risk) ~ ., data = gc_train,mtry=ncol(gc_train)-1, ntree = num_trees[i],importance = TRUE)

  # Make predictions on the test set
  pred.credit.test_bagging <- predict(bag.credit, gc_test,type = "response")

  # Calculate test error (assuming binary classification)
  test.error_bagging[i] <- mean(pred.credit.test_bagging != gc_test$credit_risk)
}
# Plot the test errors for each number of trees
plot(num_trees, test.error_bagging, type = "b", xlab = "Number of Trees", ylab = "Test MSE (Bagging)")
# Find the optimal number of trees and corresponding test error
best_num_trees_bagging <- num_trees[which.min(test.error_bagging)]
mse_bagging <- min(test.error_bagging)
```

```
best.bag <- randomForest(as.factor(credit_risk)~., data = gc_train,mtry=ncol(gc_train)-1,ntree=best_num_trees_bagging,importance = TRUE)

best.bag.pred <- predict(best.bag,newdata = gc_test,type='response')
table(best.bag.pred,response_test)
mean(best.bag.pred!= response_test) #0.24




#Conclusion #Statistics
calculate_profit <- function(predictions,method) {
  # Create confusion matrix
  confusion_matrix <- table(predictions, response_test)

  # Calculate proportions
  proportion_good_good <- confusion_matrix[4] / sum(confusion_matrix)
  proportion_bad_good <- confusion_matrix[2] / sum(confusion_matrix)

  # Calculate per-unit profit
  per_unit_profit <- proportion_good_good * 0.35 + proportion_bad_good * (-1)

  #Calculate average amount for applications predicted good
  mean_approved_amount<- mean(subset(data.frame(predictions,gc_test$amount), predictions == 'good')$gc_test.amount)

  #calculate per applicant profit =
  per_applicant_profit <- mean_approved_amount * per_unit_profit

  #calculate overall profit for test data
  profit = per_applicant_profit * (confusion_matrix[2]+confusion_matrix[4])

  #calculate test mse
  test_mse = (confusion_matrix[2]+confusion_matrix[3])/sum(confusion_matrix)

  #calculate opportunity cost / profit missed
  opportunity.cost = per_applicant_profit * confusion_matrix[3]

  result_table <- data.frame(
    Prediction_Method = method,
```

```
    Test_MSE = test_mse,
    Per_Unit_Profit = per_unit_profit,
    Per_Applicant_Profit = per_applicant_profit,
    Mean_Approved_Amount = mean_approved_amount,
    Profit_Overall = profit,
    Opportunity_Cost = opportunity.cost
  )
  return(result_table)
}

all_results <- rbind(
  calculate_profit(log.pred, "Full Logistic Model"),
  calculate_profit(lda.pred, "LDA Classifier, Threshold = 0.3"),
  calculate_profit(lda.prob$class, "LDA Classifier, threshold = Prior Prob"),
  calculate_profit(qda.prob$class, "QDA, threshold = prior"),
  calculate_profit(qda.pred, "QDA, threshold = 0.3"),
  calculate_profit(knn.pred1, "KNN, K = 20"),
  calculate_profit(bestlog.pred, "Subset Selection: BIC"),
  calculate_profit(bestaic.pred, "Subset Selection: AIC"),
  calculate_profit(ridge.pred, "Ridge: best lambda"),
  calculate_profit(ridge.pred2, "Ridge: best lambda, 1se"),
  calculate_profit(lasso.pred, "Lasso: best lambda"),
  calculate_profit(lasso.pred2, "Sparse Lasso: best lambda with 1se"),
  calculate_profit(gam.pred, "GAM: Logistic Model"),
  calculate_profit(tree.pred, "Classification Tree"),
  calculate_profit(tree.pred2, "Pruned Classification Tree"),
  calculate_profit(best.boost.pred, "Boosting, tuned"),
  calculate_profit(best.bag.pred, "Bagging, Best number of trees"),
  calculate_profit(best_rf.pred, "Random Forest: best mtry")
)
all_results
```