UNIVERSITÉ DE NEUCHÂTEL Institut de statistique

Time Series Analysis Project

Master of Science in Statistics

Faculty of Science
Institute of Statistics
University of Neuchâtel

Credit Applicant Classification:

German Data

By : **Kevin Nyamai and Jan Kasperek**

May 2023

Introduction

The goal of this project is to set up a model to classify credit applicants according to their credit risk (good vs bad), based on a dataset of applicants in the German bank. The model could be used by banks to support their credit decisions. The goal is to maximize profit from the bank's perspective hence the bank needs a decision rule who to give approval of the loan and which applications to reject. The whole process is structured along the lines of the Cross Industry Standard Process for Data Mining (CRISP-DM)withsix phases:

- **Business Understanding**: Understanding the bank's needs and objectives.
- **Data Understanding**: Discovering and evaluating the composition of the dataset (section 2 of this report).
- **Data Preparation**: Cleaning and preparing the data for modelling (section 2).
- **Modelling** : Selection and application of different suitable statistical and Machine Learning models.Since the goal of of this project is to produce accurate predictions, models are not chosen based on their interpretability or ability to conduct statistical inference with them. We will consider **parametric** and **non-parametric** regression techniques (*Elastic Net Logistic Regression* and *Multivariate Adaptive Regression splines*, respectively) as well as **tree-based ensemble methods** (*Random Forests* and *Bayesian Additive Regression Trees*). Modelling is described in section 3.
- **Evaluation**: Evaluation of predictive ability of the applied models with an aim of picking the best that meets the bank's objectives (section 4).
- **Deployment** : Suggestions regarding the application of the model detailing additions and constraints pertinent to effective and efficient implementation of the model chosen (section 5).

Business understanding must start with a proper understanding of what we are trying to predict,i.e.our binary target variable (i.e. good vs bad credit risk). Credit risk is the probability of a financial loss resulting from a borrower's failure to repay a loan. An applicant with good credit risk is likely to repay the loan i.e., owed principal and interest on time, hence not approving the loan to the individual results in loss of business to the bank[1]. Applicants with bad credit risk are unlikely to repay the loan within time, hence approving the loan leads to financial loss to the bank resulting from an interruption of cash flows and increased costs for debt collection.

An applicant's demographic and socio-economic profile are considered by loan managers before decision making on a loan application. A model developed from the data is expected to provide the loan managers guidance in loan approvals based on applicants profiles. Deployed in a

[1]https://www.investopedia.com/terms/c/creditrisk.asp

business setting, such a model could be applied in the banking business to partially automate the credit decision process or simply aid loan clerks in the manual decision process. This could be useful for banks to save the workload involved in checking all applications manually, thus saving costs. If the expected profit when using the model exceeds the expected profit when not using the model, the model could be considered useful.

Data Description and Preparation

The raw dataset contains 1000 rows across 32 variables (columns) with no missing values:

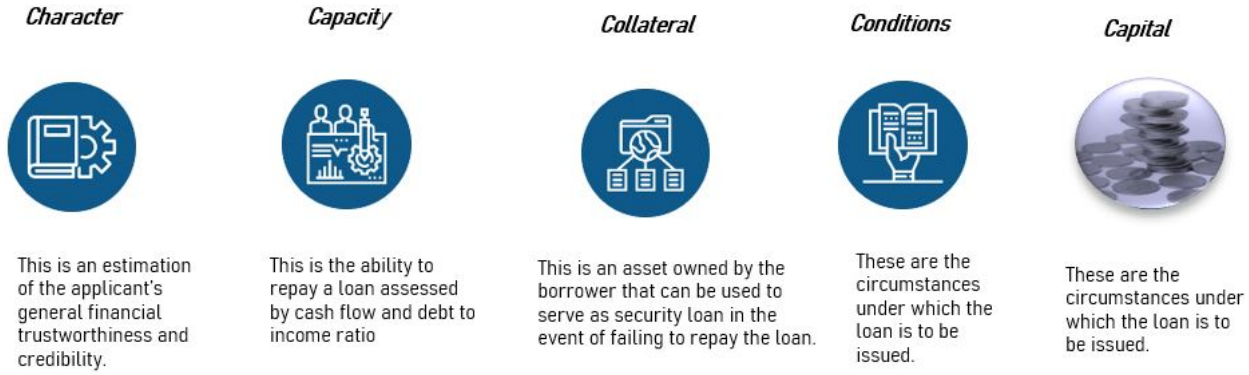
- **1 ID variable**, not used for modeling and thus removed.
- **Binary response variable**: Good (700 observations) vs Bad credit (300 observations).
- **5 numerical predictors**, from which 3 can be considered continuous (*Age*, *number of dependents*,*credit amount*, *duration of credit in months*, *installment rate as % of disposable income*). The empirical distribution of continuous predictors conditional on the response is given in Fig. 1.
- **25 categorical predictors**

The predictors can broadly be classified under the "**5 Cs**" described in credit management (see Figure 0.1) [2]in addition to demographic factors as listed below:

- **Character**- This is an estimation of the applicant's general financial trustworthiness and credibility. The predictors determining this facet include *credit history* and *number of existing credits at with this bank*.
- **Capacity**- This is the ability to repay a loan assessed by cash flow and debt to income ratio. The predictors under this facet include average balance in savings account, present employment since and installment rate as % of disposable income.
- **Capital**- This is the amount an applicant can invest. It differs from capacity as gives a short glimpse of the applicant's ability to invest rather than estimation of future financial status.
- **Collateral**- This is an asset owned by the borrower that can be used to serve as security loan in the event of failing to repay the loan. The predictors include whether applicant has a co-applicant, whether an applicant has a guarantor, whether an applicant owns property and whether an applicant owns real estate.
- **Conditions**- These are the circumstances under which the loan is to be issued. The predictors that determine this facet in our data include the purpose of credit(i.e.*newcar*, *usedcar*,*furniture*,*education*,or *retraining*)

[2]https://www.lexingtonlaw.com/education/credit-risk

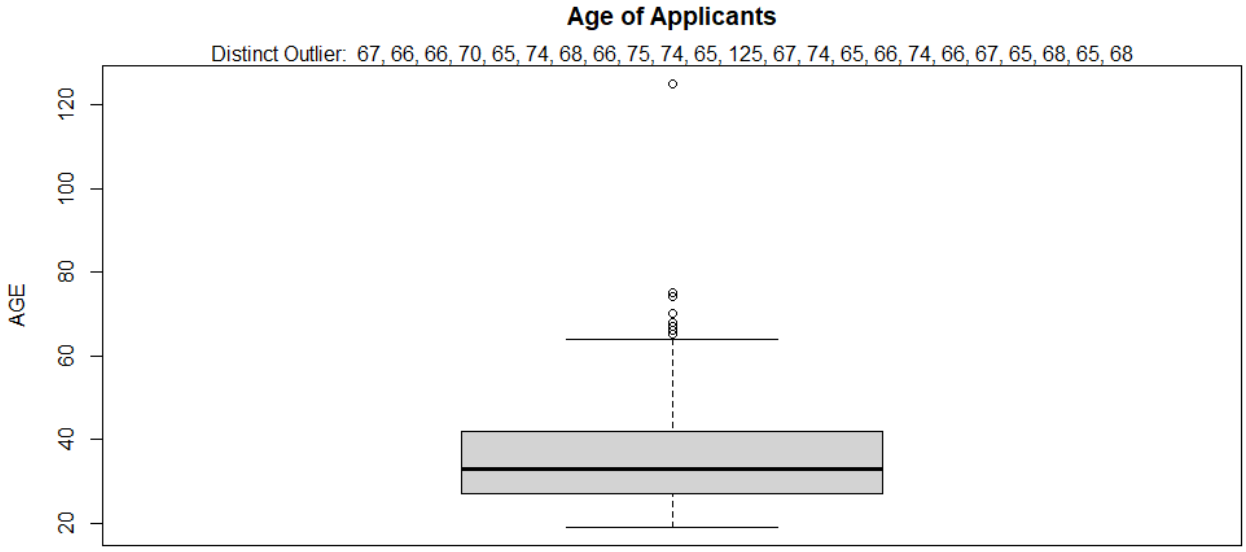
Figure 0.1. 5 C's of Credit Management



The rest are demographic factors and include, among others information regarding *age*, *marriage/relations* *status* and *rents* (whether applicant rents).

Three implausible observations are removed, one where age (in years) is 125 (see Figure 0.2), and two where binary predictors (*education* and *guarantor*,respectively)take values not defined in the data description. As the raw dataset contains 1000 observations, this is not expected to influence the modeling noticeably. All potential predictors contained in the dataset could plausibly be related to credit risk, so we have no reason to remove some variables a priori, with one exception.

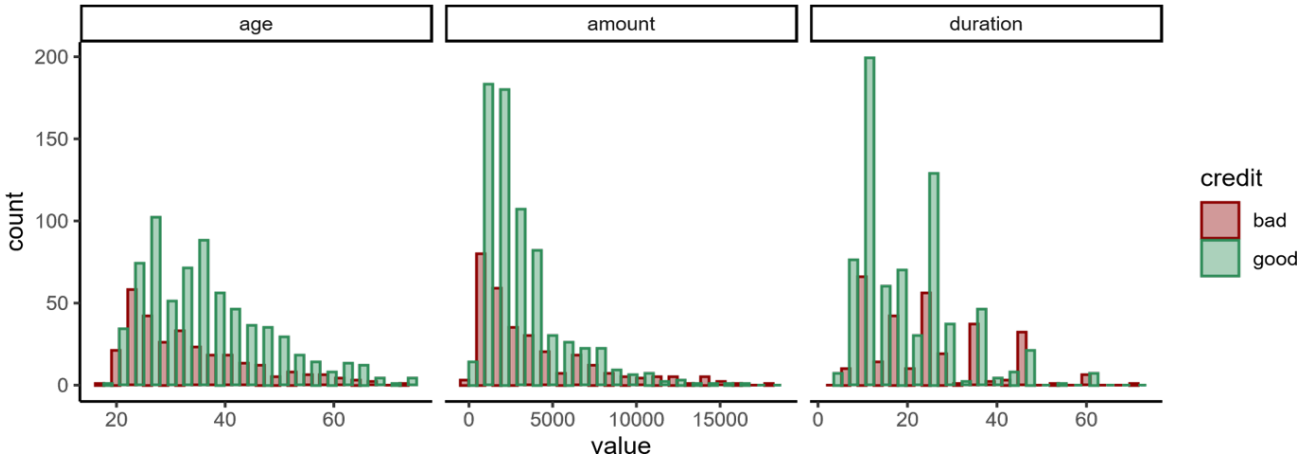
Figure 0.2. Boxplot of Age of Applicants



The column *foreign* (whether the applicant is a foreign worker) is deleted since there are only very few cases (3.6%) and it seems likely that, at least in the future, determining an applicants credit risk using his nationality is illegal due to "discrimination"[3].

For the remaining, cleaned dataset, the following observations can be made. The predictor *amount* has a highly skewed distribution with a long right tail (see Figure 0.3 below).

Figure 0.3. Histogram of Amount by Continuous Variables

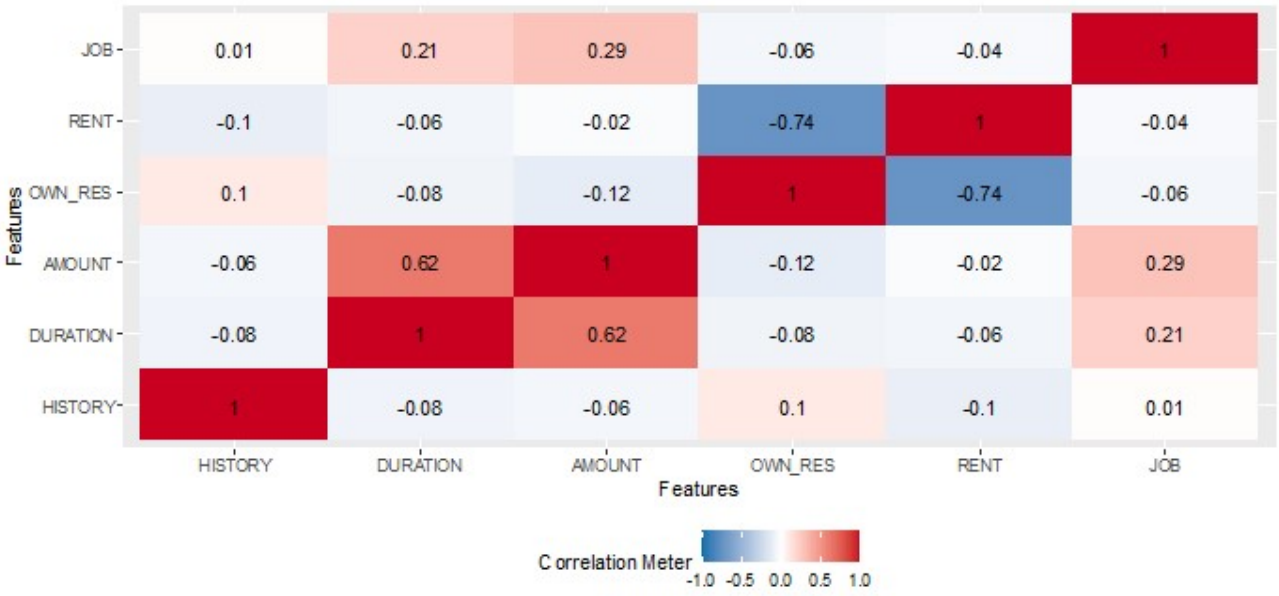


It is custom to transform such variables so that their distribution becomes more symmetric (e.g., through a log-transformation). Of the models considered later, most should be able to handle this distribution well. In logistic regression, the marginal effect of a predictor on the response is assumed to converge to zero in the extremes, making it relatively stable against predictor distributions like this(Hammouri et al., 2020). Tree-based methods consider only binary partitions of the predictors. The only concern here would be that there could be too little observations in the high ranges, making the splitting more unreliable. Concerning the categorical predictors, most have two factor levels except for *checking account status*, *credit history*, *average balance in savings account*, *present employment since* and *nature of job*. For modeling, dummy variables are created such that the alphabetically first category is left out.

Due to always leaving out one category when decomposing categorical variables into dummies, there are no perfectly collinear predictors in the dataset. Only a few variable pairs have an absolute Pearson correlation r bigger than 0.5.

[3]https://expatrist.com/can-foreigners-get-a-loan-in-germany/

Figure 0.4. Correlogram of Predictors with $rho > 0.5$



Most of those pairs represent different levels of the same categorical predictor, the only exceptions being *duration of credit* with *credit amount* ($r = 0.62$) and *whether the applicant rents* with *whether he owns a residence* ($r = -0.74$) (Note that Pearson correlation coefficient,r, between two categorical variables is equivalent to the ϕ coefficient from the analysis of contingency tables). Highly correlated predictors are thus not a problem we expect for modeling.

While some rare factor levels of categorical variables could be merged meaningfully, we decided to keep the data as raw as possible, avoiding too many assumptions on what predictors will provide no meaningful information. All models we consider can perform variable selection themselves. It should still be noted that the number of variables we pass to the models would be considered quite high by some. In the context of inference in logistic regression, a "**1 in 10**" heuristic is sometimes invoked, stating there should be at least 10 observations in the rarer category of the response per predictor included in the model (Peduzzi et al., 1996). In a real business setting, we would likely merge some variables with the help of expert judgement within the company. Here, we note that our goal is prediction, **not inference**. Including even more predictors in the models could make them unstable, however. Especially considering the binary predictors with very imbalanced distributions, it will always be desirable to have more observations available.

An important requirement for building an effective prediction model is that it is trained with only the information on the predictors available when the outcome is still unknown. A common challenge with applying Machine Learning algorithms in a business setting is *leakage*: when extracting data out of a database, it might have been updated after the information of the target variable was entered. If this updating is in any way related to the target variable, we end up (partly) predicting this target variable with itself when using the updated data for

modeling. This may result in poor performance of a model once it is deployed (but not during model building). Since we did not collect the data themselves, we can only assume there are no leakage issues with this dataset.

Modelling

To achieve realistic assessments of prediction quality, we reserve 20% of the data as the test set, the rest is for training the models. The models we consider have hyperparameters, i.e., parameters not estimated in the fitting process but rather fixed before. To find a good set of hyperparameters, we must evaluate models with different combinations of hyperparameter values. This is called *hyperparameter tuning*. Since we reserved the test set for the final evaluation, hyperparameter tuning must take place on the training set. To determine which hyperparameter combinations are best, we again need a reliable means of evaluating the fitted models with different hyperparameter combinations. For this reason, we cannot use the whole training set for hyperparameter tuning. Instead, 5-fold cross validation will be used and hyperparameters selected according to the average model performance over the folds. Finally, the models with the selected hyperparameter combinations are fitted on the whole training set and then evaluated on the test set. Sometimes, a larger proportion of the original dataset (e.g., 30%) is assigned for the final evaluation. This is avoided here because of the low observations to predictor ratio of the full dataset, which also contains categorical variables with very rare levels. To achieve a reliable assessment of different hyperparameter combinations on each fold of the training set, both the number of folds for cross-validation and the proportion of data assigned for final model assessment are set to the lowest custom values. Increasing the number of folds would also significantly increase the computational burden.

For the whole modeling and evaluation process, we use the *tidymodels* R package (Kuhn and Wickham, 2020), which acts as a "meta-package", accessing different packages for different models. It offers a unified framework for tuning, fitting and evaluating many different models at once with the same commands, reducing the need for lengthy code.

We will try four different types of models implemented in *tidymodels*: **Elastic Net Logistic Regression** (in what follows $ElasticNetLogit$), **Random Forests**, **Bayesian Additive Regression Trees (**$BART$**)** and **Multivariate Adaptive Regression Splines (**$MARS$**)**. The model choice is motivated by the nature of the dataset and computational considerations :

- The data is not very big, but has a high number of variables and contains many binary candidate predictors.
- Classifiers like *Support Vector Machines* or *nearest neighbor methods* are not expected to perform well, as these are designed for continuous predictors(Boser et al., 1992).
- On the other hand, *boosted tree models* and *neural networks* may do well. However, with such a relatively small dataset and very complex models, the danger of overfitting is large and these models would be sensitive to small changes in the hyperparameters. Hence, these models would require extensive tuning. As the number of hyperparameters of these models is relatively large, finding a good specification of these models would be very computationally intensive.

Hence we proceed with the four models aforementioned.

- **Elastic Net**. Elastic Net logistic regression (Zou and Hastie, 2005) is a penalized logistic regression model with a mixture of the Ridge and LASSO penalty, which can also reduce to pure Ridge or pure LASSO, depending on the mixture coefficient. This model can perform shrinkage of the coefficients and variable selection, while retaining the robustness and low computational burden of a parametric regression model. All predictors are standardized before model fitting. Predicted probabilities are obtained the same way as in standard logistic regression.
- **Random Forests**. Random Forests (Breiman, 2001) are ensembles of decision trees fitted on different bootstrap samples. This technique avoids the usual overfitting problem of single decision trees and is known for its high prediction accuracy while still having a tolerable amount of model complexity. As an additional safeguard against overfitting, only a randomly sampled subset of the predictors is considered at each split of the individual trees. Predicted probabilities can be obtained by averaging the class predictions across all trees.
- **Bayesian Additive Regression Trees (BART)**. BART (Chipman et al., 2010) is a more complex tree ensemble model. Different trees are grown on a number of bootstrap samples as in random forests. However, each tree is edited iteratively to fit the yet unexplained variation in the target variable (an idea similar to boosting). To avoid overfitting, the way in which the single trees are changed over the iterations is regularized by a parametric Bayesian model, the prior of which is determined by tuning parameters. To obtain predictions from each tree in the ensemble, the trees gotten over the iterations are averaged discarding the first ones ("burn-in"). The individual tree predictions are finally combined from the final trees. BART is known for good robustness properties, e.g. not needing excessive hyperparameter tuning.
- **Multivariate Adaptive Regression Splines (MARS)**. MARS Friedman (1991a) is a non-parametric regression spline model which builds piece-wise linear functions. It is constructed in a stepwise fashion and includes a pruning procedure to avoid overfitting. MARS automatically tries all possible products between the predictors up to some degree d, where $d = 1$ yields a purely additive model. In this way, MARS can also account for interactions and nonlinearities. To get predicted probabilities, the model is passed through a logistic regression. Since non-parametric regression techniques like MARS can be sensitive to small changes in the data, MARS models are fitted to several bootstrap samples before averaging their predictions, yielding a bagged (Breiman, 1996a) MARS model. This reduces the variance of its predictions. For keeping computational load on a tolerable level, we use 20 bootstrap samples.

The models chosen work well without extensive tuning. In all models, 2 hyperparameters will be tuned. Some have more tuning parameters, but, in our trials, the models do not seem to be sensitive to a change in those. These results are also supported by the literature. For

instance, (Oshiro et al., 2012), we find that increasing the number of trees in random forests above a few hundred does not significantly improve predictive accuracy. An overview of the tuning parameters is given in Table 0.1 below. The R packages used by *tidymodels* and the range of values covered by the grid search are also indicated. The general approach is to search an equally spaced grid with max. 10 values for each tuning parameter, covering a wide range around the defaults implemented by the R packages used for model fitting.

Table 0.1. Description of Hyperparameters and Tuning Grids

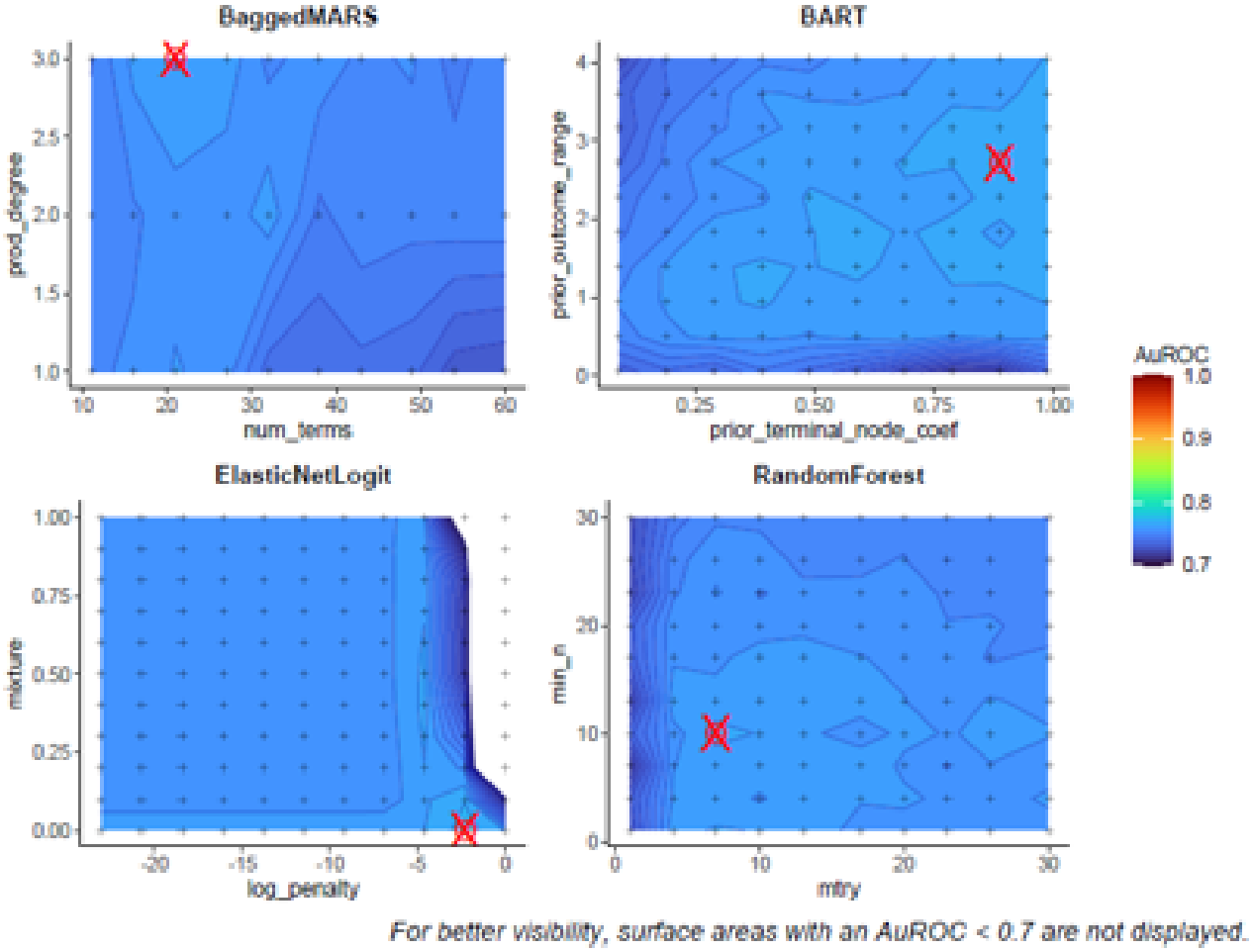
Model	*Hyperparameter*	**Explanation**	**Range**	*Package*
BaggedMARS	*num_terms*	max. number of terms before pruning	10-60	*earth*
	prod_degree	max. degree of interaction	1-3	
BART	*prior_terminal_node_coef*	prior on probability that tree node is terminal	0.09-0.99	*dbarts*
	prior_outcome_range	prior on range of predicted outcomes	0.05-4.05	
ElasticNetLogit	*mixture*	mixture of Ridge (0) and LASSO (1) penalty	0-1	*glmnet*
	log_penalty	Natural Log of regularization parameter	(-25)-0	
RandomForest	*mtry*	# of predictors considered for tree splits	1-30	*randomForest*
	min_n	min. # of observations in terminal nodes	1-30	

Model Evaluation

To tune and evaluate the models, we first need some quantity that illustrates their predictive quality. All the models considered give predicted probabilities in addition to class predictions. Predicted probabilities are preferred since they contain more information than a simple class prediction. To set up a confusion matrix and calculate measures derived from it, one then needs to choose a classification rule, i.e. a cutoff value determining which predicted probability marks the border between being classified as good or bad credit. Often, a value of 0.5 is chosen as cutoff. This is rather arbitrary and not reasonable if the class distribution in the target variable is very unequal. Moreover, since the costs of misclassification are unequal (falsely classifying someone with bad credit as good credit is likely to cost the company more), choosing 0.5 as the cutoff may not sensible even if the class probabilities are equal. Without knowledge of the cost function of the bank, a suitable cutoff value cannot be derived. It is thus desirable to choose a metric which is not dependent on the choice of a cutoff value. Instead, we can rely on a more general metric: The area under the ROC curve ($AuROC$) considers true and false positive rates (TPR and FPR) at different cutoff values. From this, a curve is constructed with the FPR on the horizontal and TPR on the vertical axis. The area under this curve gives a general indication of the predictive quality which is robust to class imbalance and does not require an arbitrary choice of cutoff. Possible values of the AuROC range between 0.5 (random classification) and 1 (perfect classification).

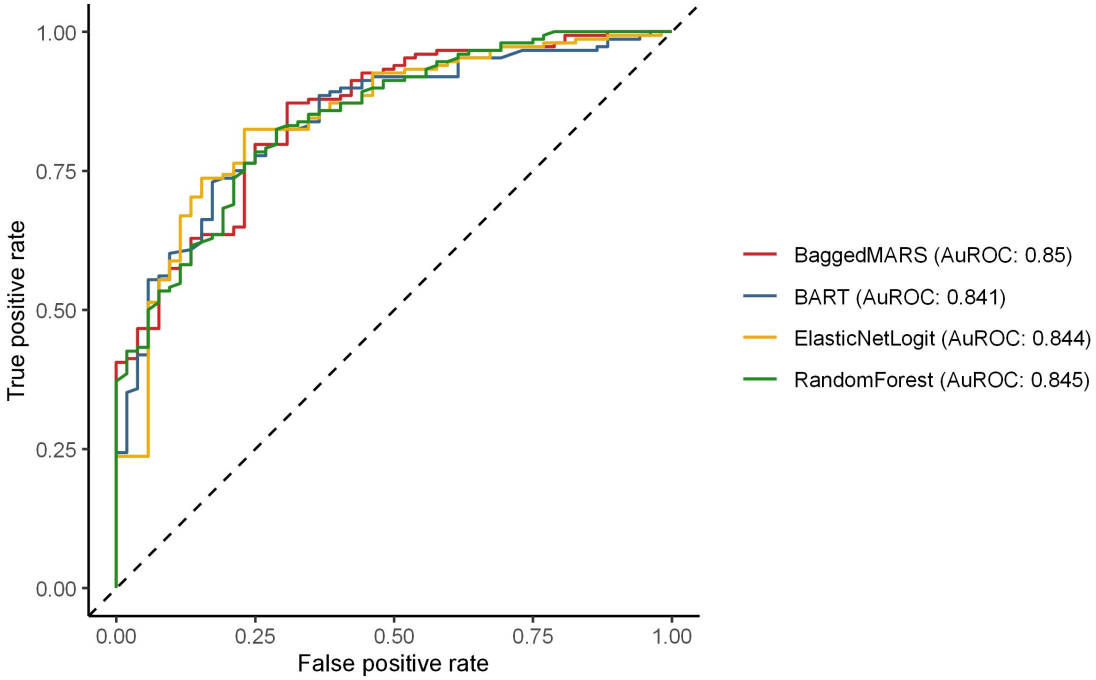
After hyperparameter tuning, to make sure we chose sensible values we check visually how sensitive the AuROC is to small changes in these parameters. As only 2 parameters are tuned for each model, we can produce a contour plot of the average AuROC values obtained across the 5 cross validation folds of the training set, for each hyperparameter combination considered as seen below on Figure 0.5 .

Figure 0.5. AuROC surfaces of Hyperparameter Search Grids (red X marks the selected model)



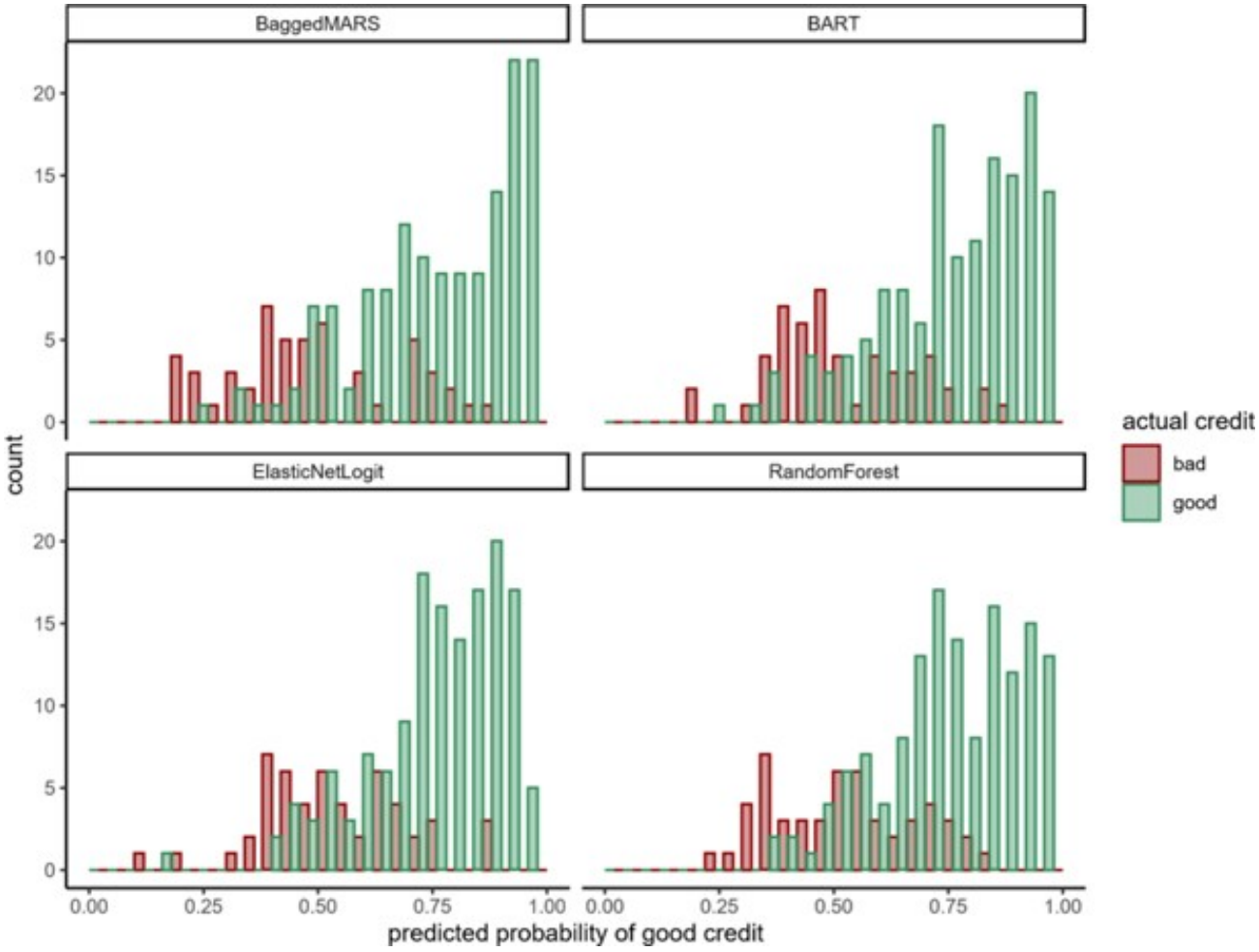
We can see that the selected tuning parameter values all lie in regions where the AuROC is generally higher than elsewhere. The model should thus be stable against small departures in the tuning parameters and can be expected to perform similarly on the test set. We can then proceed with fitting the tuned models on the whole training set before the final evaluation on the test set. Figure 0.6 shows the ROC curves with AuROC values of the final models on the test set.

Figure 0.6. ROC curves for all Selected models



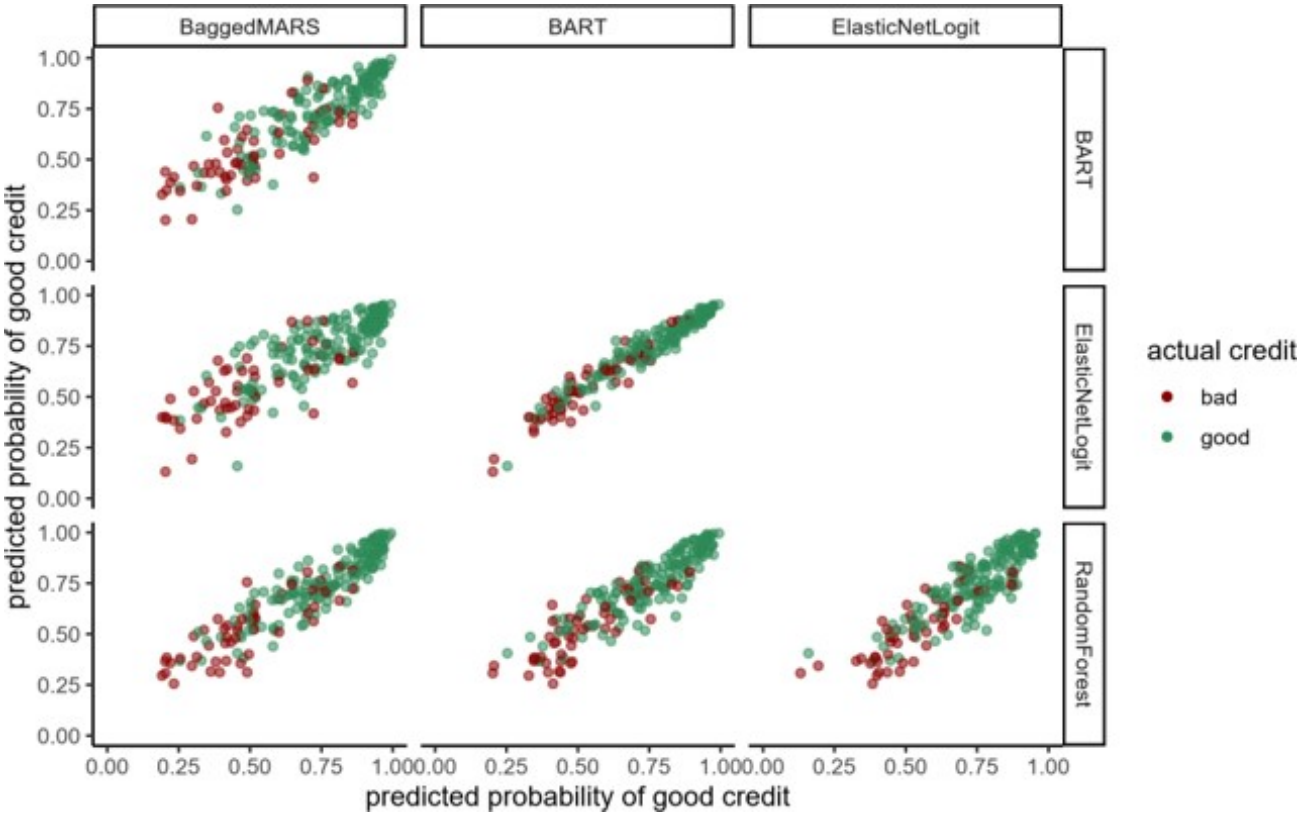
We observe that all models have a similar (and decent) overall predictive quality. For a closer diagnosis, we can have a look on the models' predictions on the test set. Figure 0.7 shows histograms of predicted probabilites of good credit by the observed value of the target variable. The more seperated the predictions of the actual classes are, the better is a given model.

Figure 0.7. Distributions of Predictions on test set, by Actual response



We observe that observations with very high predicted probabilities are almost always correctly classified as good credit, while predictions for truly bad credit are more spread out. There are very few observations with predicted probabilities of good credit below 0.25. We can conclude that the models do better in predicting good credit rather than bad credit. This is likely because bad credit is the much rarer category in this dataset ($\sim 30\%$). These patterns are similar across all models, indicating that the predictions for each observation should be very similar across models. To confirm this, we can examine the scatterplot matrix of the predictions on the test set given in Figure 0.8.

Figure 0.8. Scatter plot Matrix of Predictions on test set,by Actual response

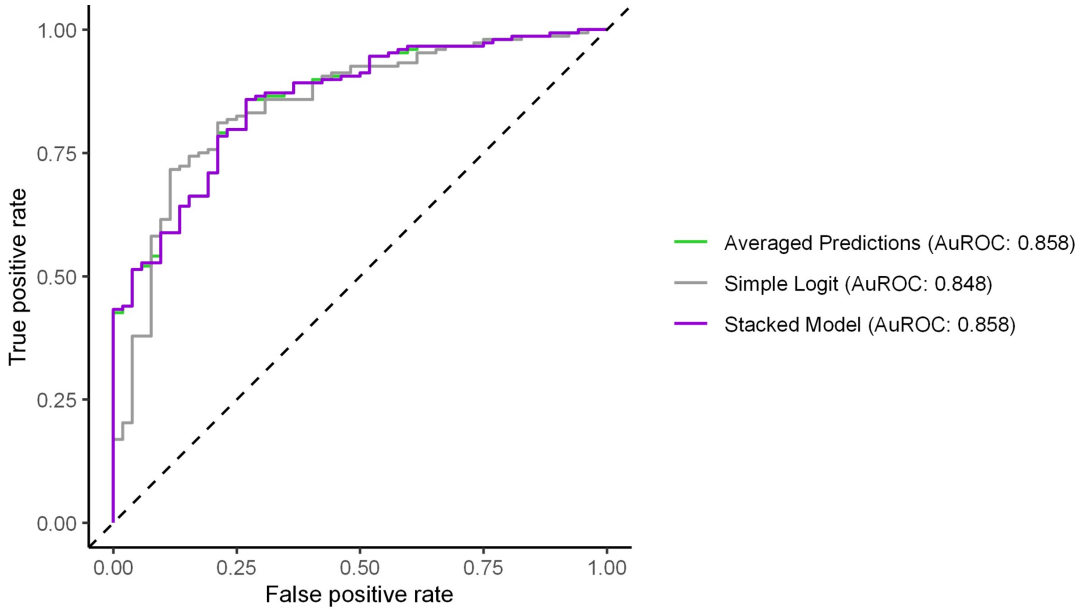


Again, we can observe a generally good separation of individuals with truly good credit and more uncertainty for bad credit. We also observe that the models are in strong agreement concerning their predictions; there are very strong positive correlations between the models. The bagged MARS model shows the biggest differences when compared to the others. There might not be much predictive gain from ensembling these models, as they contain largely the same information. Since the overall predictive accuracy indicated by ROC curves is similar across models, we may instead choose one model which is easily implemented and has the lowest running time. For implementation in a business setting, we would recommend using the Elastic Net (which here reduces to Ridge) logistic regression model. If computational load is not a huge concern, we may still ensemble the models to produce one final set of predictions. This can not only improve prediction accuracy, but also serve as a safeguard should one models' performance deteriorate on new data. A possibility to ensemble different types of models is simply taking the unweighted average of the model predictions. Another, more sophisticated approach is called *stacking* (Wolpert, 1992)(Breiman, 1996b), where a regression model is run on the out-of-fold predictions of the base models to give a linear combination of base model predictions. Due to the high correlation between the different models' predictions, it is custom to use Ridge regression to form the combination of base models. To avoid overfitting of stacking when base models are tuned, more sophisticated resampling schemes such as nested cross-validation may be required, which our data is not suited for due to its relatively small size. Moreover, stacking improves predictive accuracy the most when the predictions of base models are dissimilar; this is hardly

the case here (except maybe for the MARS model).

Nonetheless, we try both ensembling by considering the arithmetic mean of the model predictions and stacking (using the out of fold predictions of the selected models on the training set). As a benchmark, we see how well these approaches do compared to a simple, unpenalized logistic regression model. The ROC curves of these models are shown in Figure 0.9

Figure 0.9. ROCcurves of ensemble models and simple logistic regression



We see from the AuROC of our ensemble models do slightly better than the base models (see Figure 0.6). Stacking does not outperform simple averaging (the stacking weights are very similar across models). More surprisingly, the difference to a simple, unpenalized logistic regression is quite small, as its AuROC is quite high. If we are very concerned about running time, we might even propose to implement the simple Logit model, although there is still an argument to pick the Elastic Net model due to its lower variance. Nonetheless, there does not seem to be much predictive gain from using some of the sophisticated methods considered. Viewing model averaging as an "insurance" against deterioration of some base model on new data and not putting an emphasis on interpretability and low computational load, the business might still decide in favor of an ensemble model. Contrary to the single models, there is still potential of improving the ensemble by adding more base models.

Another possibility to improve our models consists of weighting more important observations higher in the fitting process. This could be done according to the credit amount or the response value (giving higher weight to observations with bad credit to optimize the model towards a lower misclassification rate for truly bad credit).

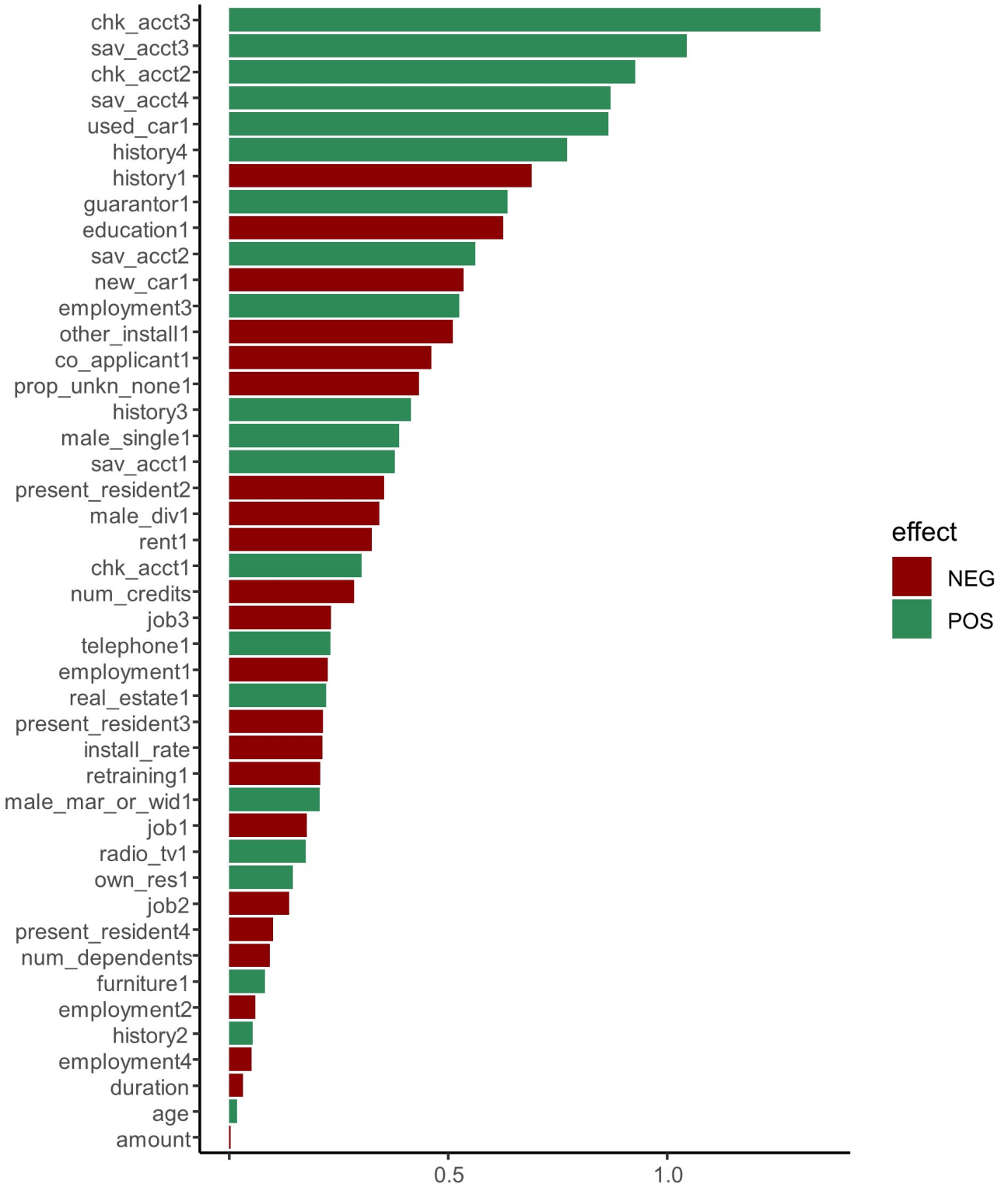
Variable importance scores were inconsistent across different models, which might be due to

the low observations to predictor ratio (Babyak, 2004). Moreover, not all models considered have variable importance functions implemented in R (Friedman et al., 2010), (Friedman, 1991b),(Kapelner and Bleich, 2013) i.e:

- BART: Lacks direct built-in variable importance functions, but the *bartMachine* package provides this functionality
- MARS: Has variable importance functions implemented in R (*earth* package).
- Random Forests: Has variable importance functions implemented in R (*randomForest* package).
- Elastic Net Logistic Regression: Does not have a built-in variable importance function but can assess importance through coefficients.

Nonetheless, having chosen the Elastic Net Logit model, we can easily obtain variable importance scores as the absolute values of the coefficients. Since all predictors are standardized before fitting the Elastic Net model, all coefficients are on the same scale and can thus be easily compared. We again caution against interpreting too much into this score, however; even this regularized model seems likely to have high variance in the coefficients, making the importance scores uncertain. Figure 0.10 below of variable importances can thus only give a very rough idea about the partial influence of the predictors on the target variable. We show the absolute value of the estimated coefficients; if the sign is negative, the importance score is shown in red, and in green otherwise.

Figure 0.10. Variable importance of the Elastic Net Logit model



Variables relating directly to the financial situation of the applicant such as the balance in checking and savings account as well as credit history seem to have the strongest influence on

whether this person has good or bad credit. It should be noted that for categorical variables, the sign and relative magnitude of the coefficients are to be interpreted relative to the first factor level that was left out to avoid perfect multicollinearity.

Deployment

Having determined a model with satisfying properties for the business problem at hand (high predictive accuracy and low computational complexity), it is time to apply the model in the real business setting. The business can use the model to partially automate its credit decision process. Checking all credit applications manually requires a lot of manpower, which costs our business time and money. The idea is thus as follows: Below a certain threshold of predicted probability of good credit, credit applicants can be automatically rejected without the need for a manual check by credit clerks. At the same time, applicants with a very high estimated probability of good credit could be automatically accepted. Every applicant that has a prediction between these two cutoff values would still have to be checked manually.

Mistaking applicants with good credit for bad credit is not as costly as misclassifying individuals with truly bad credit. Hence, the cutoff value for the classifying an applicant as good credit should likely be chosen high, such that there is a very slim chance of a wrong classification. The cutoff for bad credit can be set more leniently. The exact values for the cutoffs depend on the distribution of the predictions conditional on the actual response value Figure 0.7 and the costs of misclassification, which are to be determined by the company.

Alternatively, the model may simply be used a supporting tool for credit clerks to make better decisions and maybe allowing less experienced employees to make decisions on credit applications. In this case, the model can simply be deployed through an R shiny app, which makes it easy to maintain, as both the model itself and its user interface are built in R.

Updating the model is necessary periodically (e.g. yearly) to avoid *model drift*. This refers to a common situation where a models' predictive accuracy deteriorates over time due to structural changes in the relations of predictors and the target variable. In this specific business problem, model drift might be due to changes in the client portfolio, legal frameworks or the macroeconomic situation.

Finally, it must be remembered what the exact definition of the target variable is; from a business standpoint, a prediction model might not be necessary to assess an applicants' creditworthiness. In Germany, where the data come from, banks have access to the so-called $SCHUFA$ credit score. From this score, one could derive whether an applicant has "good" or "bad" credit risk. The $SCHUFA$ score is also a tool used in evaluating credit risk. It is a credit rating score for all German residents that basically tracks an individuals' ability to pay bills. $SCHUFA$, short for $Schutzgemeinschaft\ für\ allgemeine\ Kreditsicherung$ in German, roughly translates to "protection association for General Credit Guarantees"[4]. The score measures the probability with which an individual honours their bill, credits, and contracts.The highest rating is 100% which decreases if one doesn't pay their bills and debts. Below is tier for the $SCHUFA$ score.

[4]https://feather-insurance.com/blog/how-can-you-improve-your-schufa-score/

Table 0.2. *SCHUFA* Score Tier Zones

Score Value	Risk Assessment-Level of risk
$> 97.5\%$	Very low
$95\% - 97.5\%$	Low to negligible
$90\% - 95\%$	Satisfactory to elevated
$80\% - 90\%$	Fairly elevated
$50\% - 80\%$	Very elevated
$< 50\%$	Critical

The score however does not encompass other socio-economic and demographic factors hence should not be used as standalone tool as it is mostly based on credit history and bill payments[5]. Provided that the target variable was not derived from the SCHUFA score, our model may thus still have business value. The same could be the case if the costs of using the SCHUFA system exceed the costs of building, using and maintaining our model.

[5]https://www.settle-in-berlin.com/what-is-schufa/

References

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266 – 298, 2010.

Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19 (1):1 – 67, 1991a.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 67(2):301–320, 2005.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Leo Breiman. Stacked regressions. *Machine learning*, 24:49–64, 1996b.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *IAPR International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2012.

Peter Peduzzi, John Concato, Elizabeth Kemper, Theodore R. Holford, and Alvan R. Feinstein. A simulation study of the number of events per variable in logistic regression analysis. *Journal of Clinical Epidemiology*, 49(12):1373–1379, 1996.

Max Kuhn and Hadley Wickham. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*, 2020. URL https://www.tidymodels.org.

Hanan M. Hammouri, Roy T. Sabo, Rasha Alsaadawi, and Khalid A. Kheirallah. Handling skewed data: A comparison of two popular methods. *Applied Sciences*, 10(18):6247, 2020. doi: 10.3390/app10186247.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992. doi: 10.1145/130385.130401. URL https://www.researchgate.net/publication/2376111_A_Training_Algorithm_for_Optimal_Margin_Classifier.

Michael A. Babyak. What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models. *Psychosomatic Medicine*, 66(3):411–421, 2004. doi: 10.1097/01.psy.0000127692.23278.a9. URL https://doi.org/10.1097/01.psy.0000127692.23278.a9.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. doi: 10.18637/jss.v033.i01. URL https://www.jstatsoft.org/v33/i01/.

Jerome H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1): 1–67, 1991b. doi: 10.1214/aos/1176347963. URL https://projecteuclid.org/journals/annals-of-statistics/volume-19/issue-1/Multivariate-Adaptive-Regression-Splines/10.1214/aos/1176347963.full.

Adam Kapelner and Justin Bleich. bartmachine: Machine learning with bayesian additive regression trees. *arXiv preprint arXiv:1312.2171*, 2013. URL https://arxiv.org/abs/1312.2171.