Cost-sensitive Classification of German Credit Data

1 Problem

The problem is a bank's uncertainty in deciding to approve loans. The bank can use applicants' demographics and socio-economic profile to assess their "Credit Risk".

An applicant with:

1. Good-risk, is likely to repay the loan, not approving the loan would cause loss of business
2. Bad-risk, is unlikely to repay, approving the loan would cause financial loss to the bank.

As such, our objective is to predict Credit Risk in order to minimise financial cost to the bank.

<u>Data Source</u>

This report uses the German Credit dataset sourced from the UC Irvine Machine Learning Repository. The data was originally collected from regional bank in Southern Germany from 1973-1975. Being from a reputable and heavily cited source, the data is likely to be reliable.

<u>Target Attribute</u>

The target attribute is the applicant's Credit *Risk*. Since it can be classified as Good or Bad, this is a classification analysis.

2 Understanding the Data

<u>Nature & Size of Dataset</u>

The dataset contains 20 attributes and 1000 instances with 17 categorical and 3 numerical attributes (AppendixC). Each instance contains information of one customer applying for one loan. 700 instances were of Good-Risk while 300 were Bad-Risk.

The attributes can be generally categorised into Demographics, Credit Information, Wealth and Stability.

Visualisation

Firstly, we look at demographics.

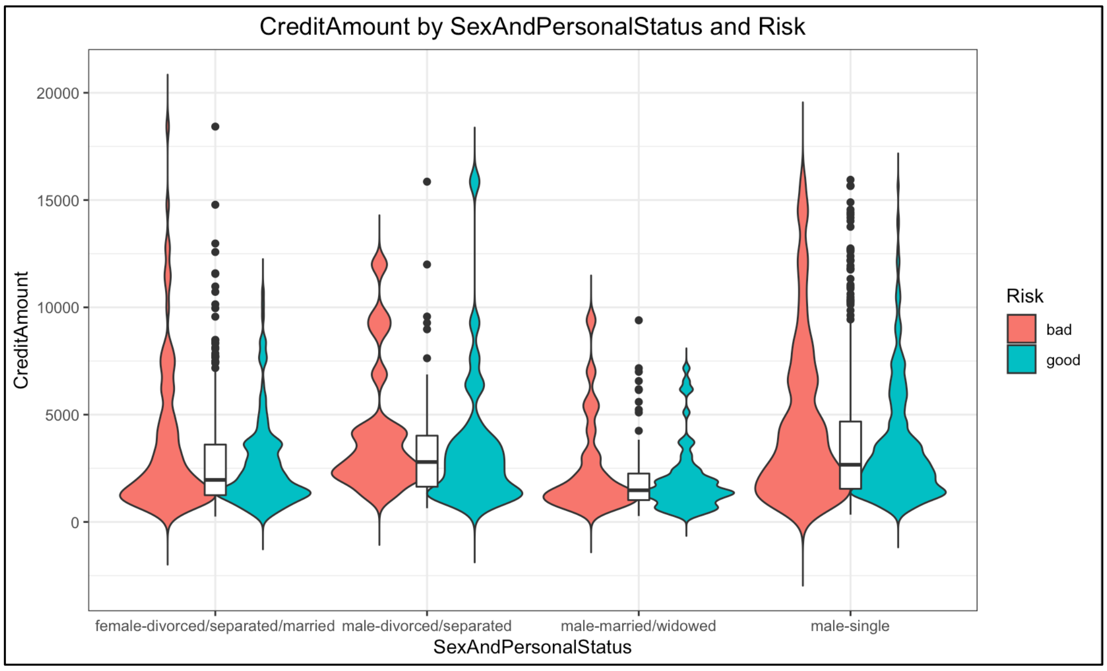


Figure 1: Violinplot of CreditAmount-SexAndPersonalStatus/Risk

Figure 1 shows varying distributions of *CreditAmount* per *SexAndPersonalStatus* and densities of Good-Risk/Bad-Risk. For "*male-single*", we observe a higher probability of Bad than Good-Risk at higher *CreditAmount*. We also note that there are no records of "*female-single*" due to regulations at the time of data collection requiring women to apply for loans in their husband's name.

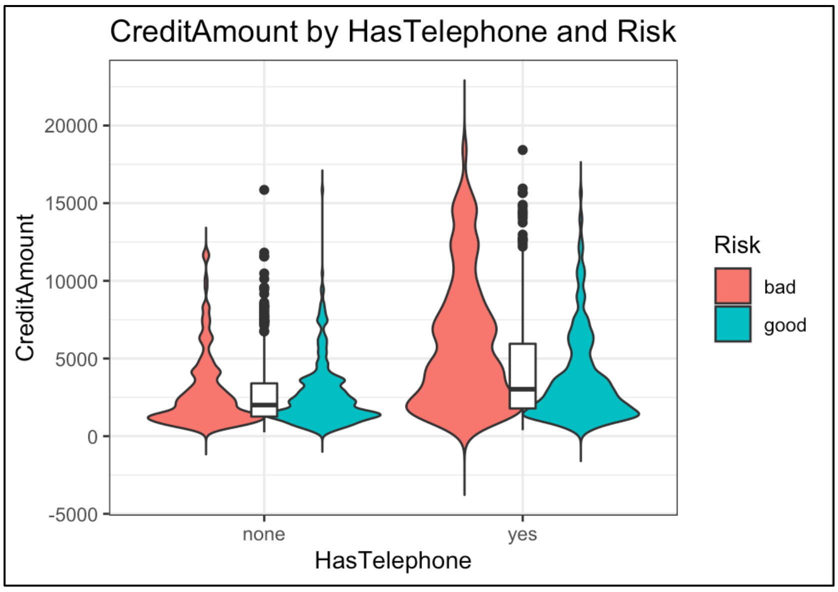


Figure 2: Violinplot of *CreditAmount-HasTelephone/Risk*

Figure 2 also shows a significantly higher probability of Bad-Risk at higher *CreditAmount* for customers with a telephone.

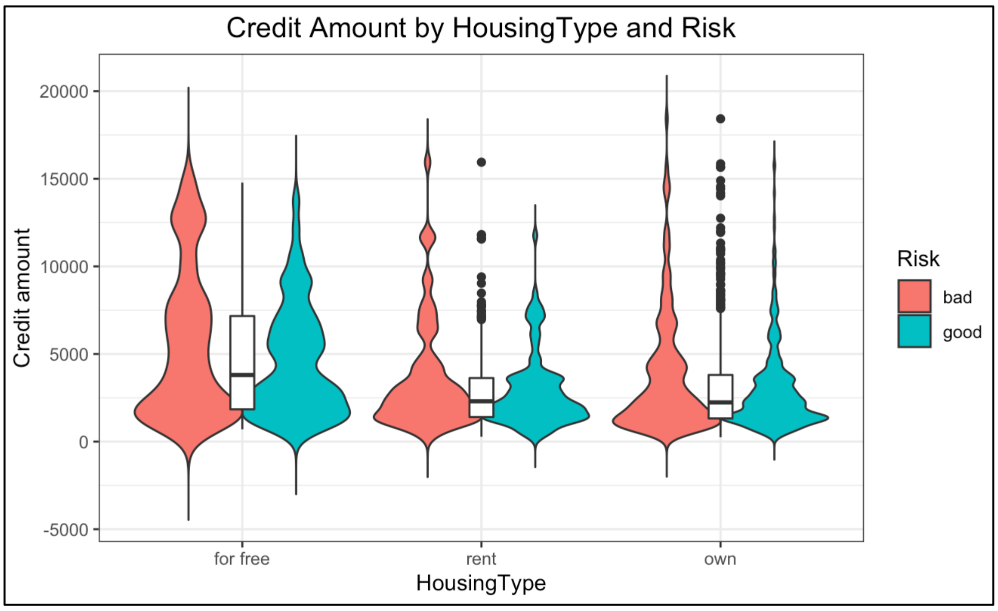


Figure 3: Violinplot of *CreditAmount-HousingType*

Figure 3 shows more customers who live in free housing applying for loans. We also observe higher probabilities of Bad-Risk at higher *CreditAmount* for customers in this group. Seeing varying effects of *CreditAmount* on Risk prompted us to further investigate attributes under Credit Information.

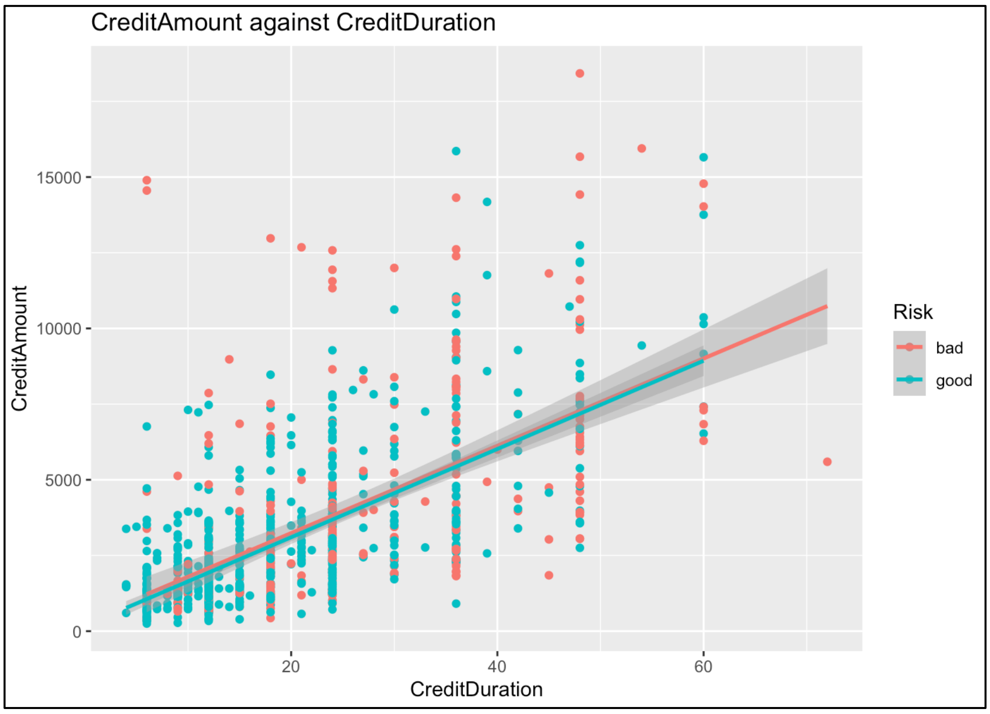


Figure 4: Scatterplot of *CreditAmount-CreditDuration*

Figure 4 shows similarly positive correlations between *CreditAmount* and *CreditDuration* for both customers with Good-Risk and Bad-Risk. This is as expected since larger *CreditAmounts* are paid off over a longer *CreditDuration*. There are also more scattered points of Bad-Risk at higher *CreditAmounts*. To further see the distributions of *CreditAmount* and *CreditDuration* individually, we create density plots.

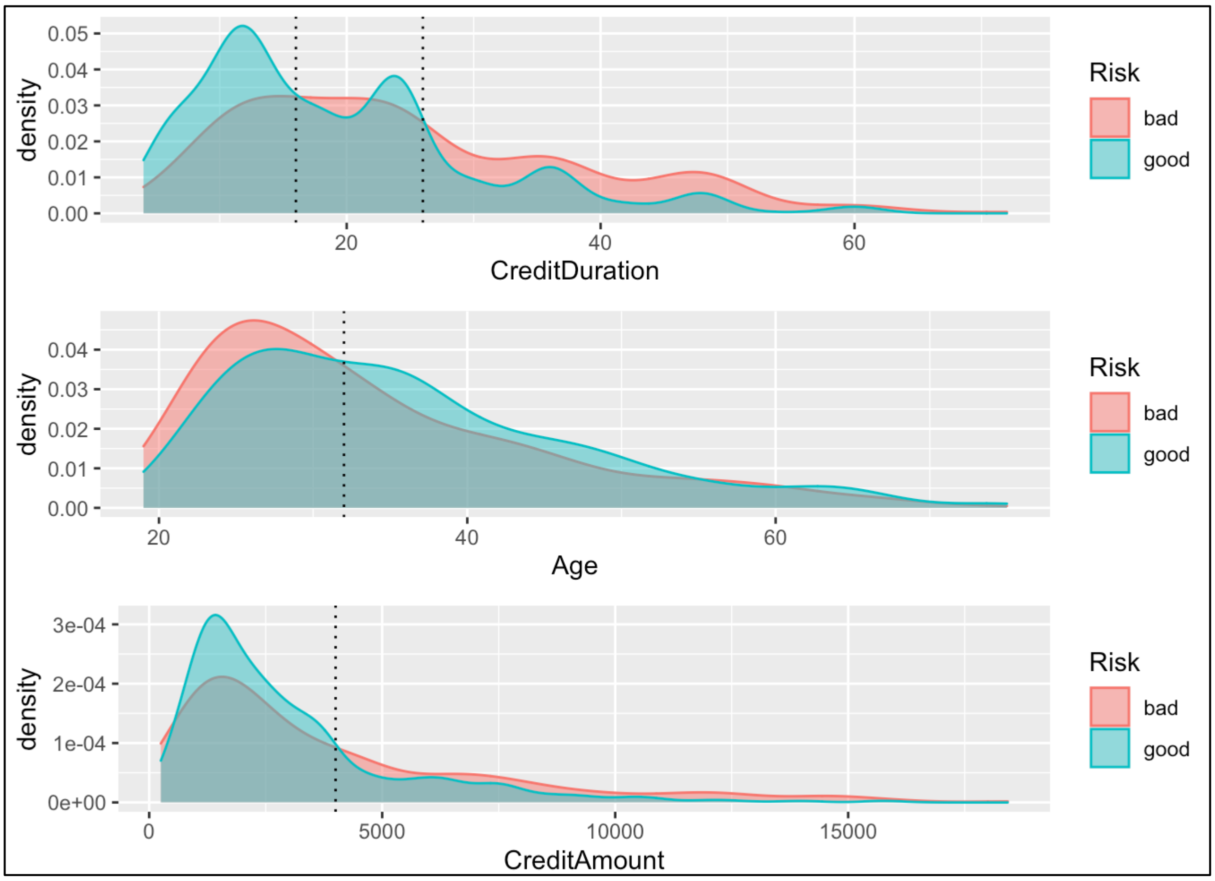
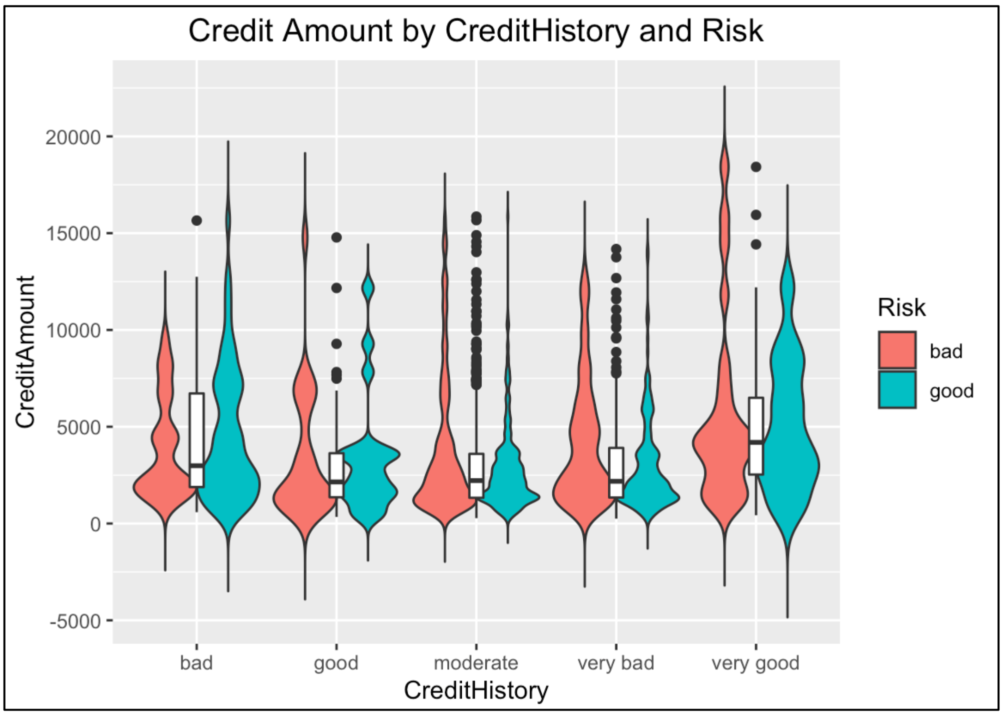


Figure 5: Density plots of *CreditDuration/Age/CreditAmount*

Figure 5 shows the densities of numerical attributes. There are significantly higher densities of Bad-Risk when *CreditDuration*>26 and *Age*<32 and significantly lower density of Bad-Risk when *CreditAmount*<4000 and *CreditDuration*<16. As such, these numerical attributes re likely important in predicting Risk.



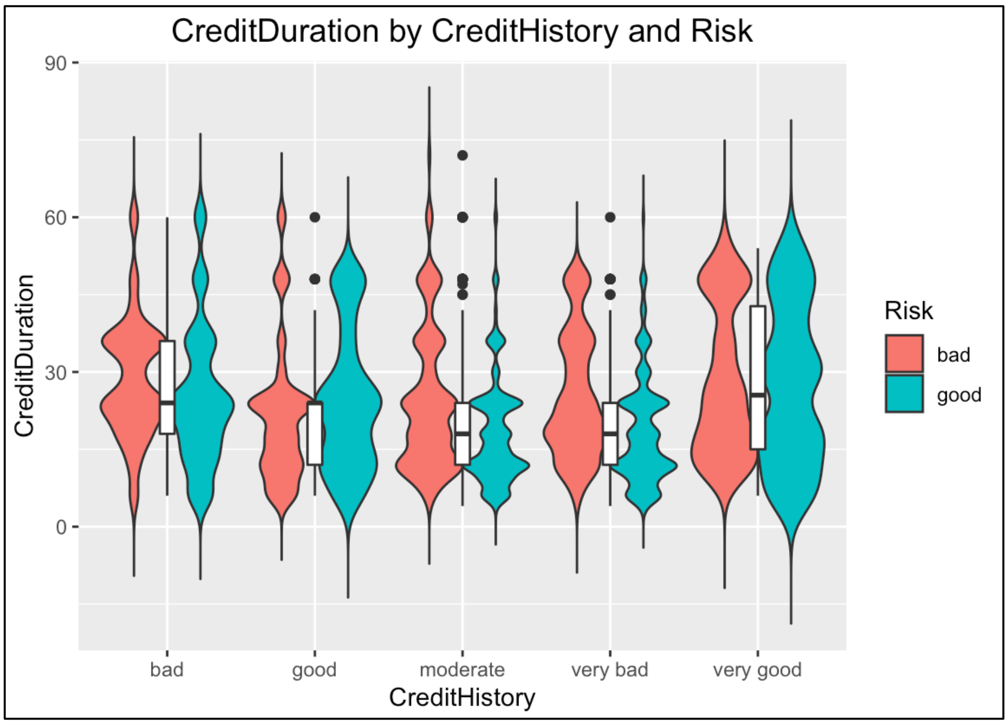


Figure 6: Violinplots of (a)*CreditAmount-CreditHistory* & (b)*CreditDuration-CreditHistory*

Figure 6 show customers with "very-good" *CreditHistory* obtaining loans of greater *CreditAmount* and longer *CreditDuration*. This group also higher probability of being Good-Risk. Contrastingly, customers with "very-bad" *CreditHistory* have significantly higher probability of being Bad-Risk regardless of *CreditAmount* or *CreditDuration*. Hence, we also expect *CreditHistory* to determine *Risk*.

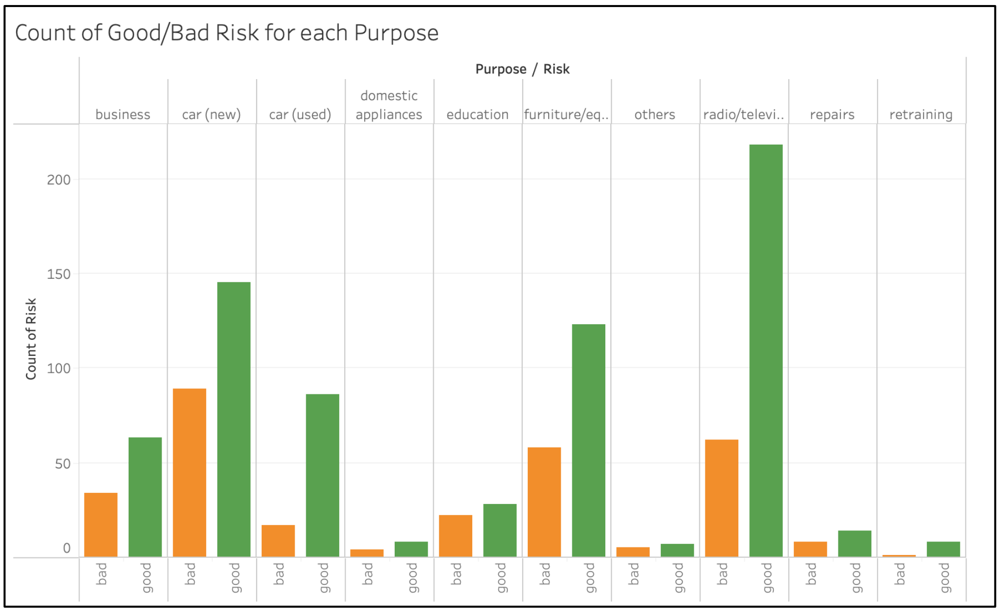


Figure 7: Barchart of Good-Risk/Bad-Risk by *Purpose*

Figure 7 shows significantly greater numbers of customers with Good-Risk who take loans for *Purpose* “car(used)” and “radio/television”. Contrastingly, loans with *Purpose* “domestic-appliances”, “repairs” and “education” are almost equally Good-Risk and Bad-Risk. However, these categories have very low frequencies and may not be representative for the larger true population.

We then investigated how Wealth and Stability relate to Risk.

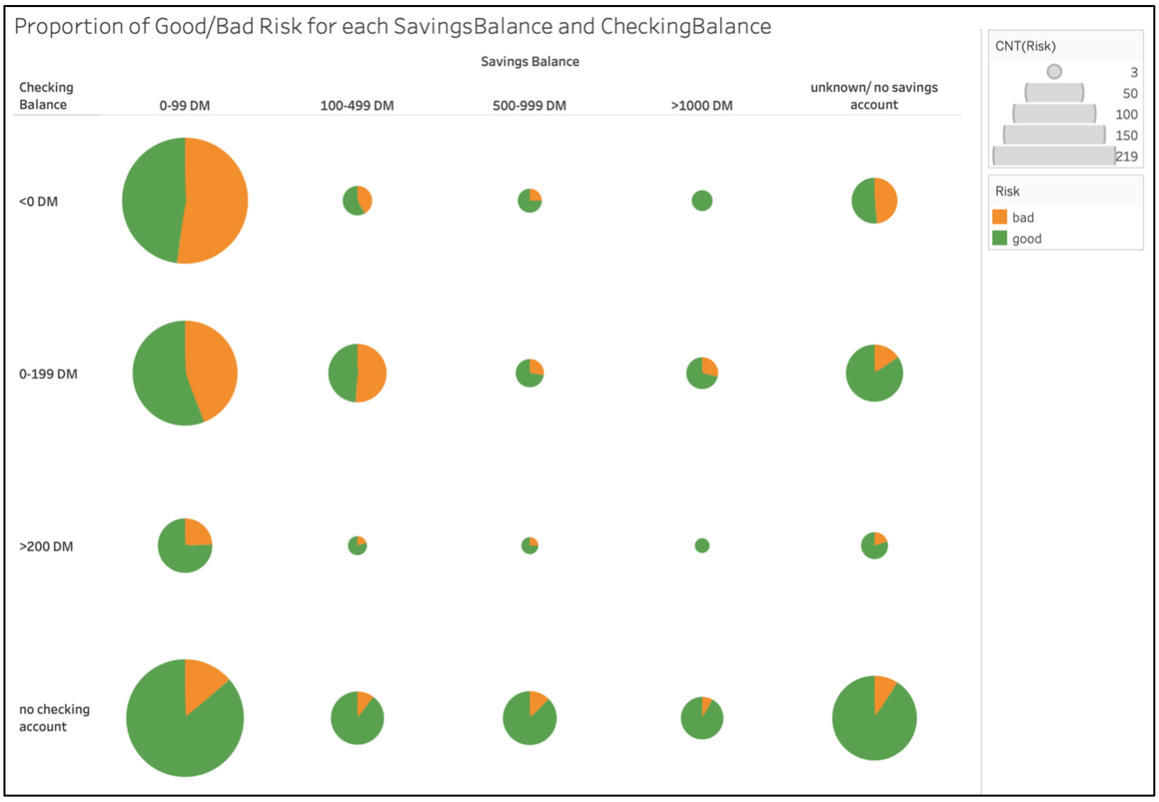
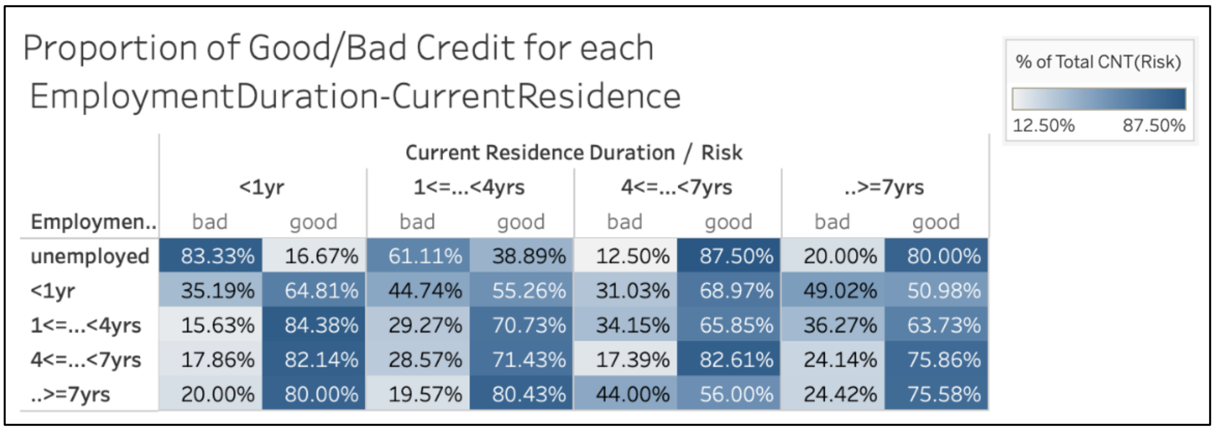


Figure 8: Piecharts of Risk by *CheckingBalance-SavingsBalance*

Figure 8 shows the low-*SavingsBalance* and low-*CheckingBalance* group containing the most instances with the highest proportion of Bad-Risk. Customers with "no checking account" have smaller proportions of Bad-Risk regardless of *SavingsBalance*. These significantly different proportions for each group illustrate the influence of both *CheckingBalance* and *SavingsBalance* in predicting *Risk*.

We observe a sizeable number of observations with "0-99DM" *SavingsBalance* and "no checking account". Thus, the Good-Risk/Bad-Risk proportions for other groups of *SavingsBalance* and *CheckingBalance* may be unrepresentative of true population



Proportion of Good/Bad Credit for each EmploymentDuration-CurrentResidence

% of Total CNT(Risk): 12.50% – 87.50%

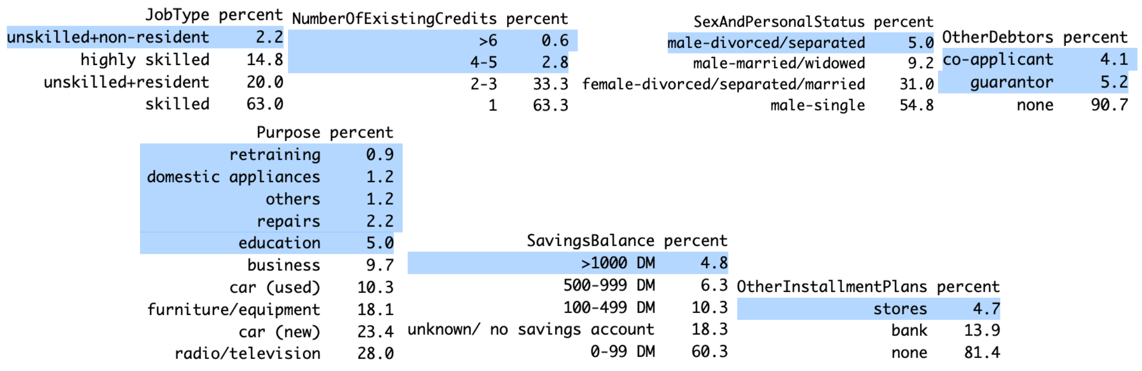
	Current Residence Duration / Risk							
	<1yr		1<=...<4yrs		4<=...<7yrs		..>=7yrs	
Employmen..	bad	good	bad	good	bad	good	bad	good
unemployed	83.33%	16.67%	61.11%	38.89%	12.50%	87.50%	20.00%	80.00%
<1yr	35.19%	64.81%	44.74%	55.26%	31.03%	68.97%	49.02%	50.98%
1<=...<4yrs	15.63%	84.38%	29.27%	70.73%	34.15%	65.85%	36.27%	63.73%
4<=...<7yrs	17.86%	82.14%	28.57%	71.43%	17.39%	82.61%	24.14%	75.86%
..>=7yrs	20.00%	80.00%	19.57%	80.43%	44.00%	56.00%	24.42%	75.58%

Figure 9: Aggregation table of Proportions *Risk* by *EmploymentDuration-CurrentResidenceDuration*

Figure 9 show "unemployed" customers with *CurrentResidenceDuration*<1yr and 1≤*CurrentResidenceDuration*<4yrs" were more likely Bad-Risk. In contrast, those with longer *EmploymentDuration* were generally more likely Good-Risk. This also shows that *EmployementDuration* may be more deterministic of *Risk* than *CurrentResidenceDuration* since it better indicates financial stability.

3 Preparing the Data

Merging Categories



JobType	percent
unskilled+non-resident	2.2
highly skilled	14.8
unskilled+resident	20.0
skilled	63.0

NumberOfExistingCredits	percent
>6	0.6
4-5	2.8
2-3	33.3
1	63.3

SexAndPersonalStatus	percent
male-divorced/separated	5.0
male-married/widowed	9.2
female-divorced/separated/married	31.0
male-single	54.8

OtherDebtors	percent
co-applicant	4.1
guarantor	5.2
none	90.7

Purpose	percent
retraining	0.9
domestic appliances	1.2
others	1.2
repairs	2.2
education	5.0
business	9.7
car (used)	10.3
furniture/equipment	18.1
car (new)	23.4
radio/television	28.0

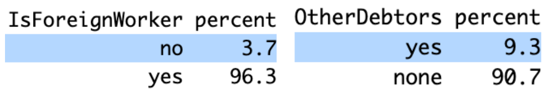
SavingsBalance	percent
>1000 DM	4.8
500-999 DM	6.3
100-499 DM	10.3
unknown/ no savings account	18.3
0-99 DM	60.3

OtherInstallmentPlans	percent
stores	4.7
bank	13.9
none	81.4

Figure 10: Categories with proportion<5%

Some categorical attributes have categories with very few observations (Figure10) which may unrepresent the true population. Hence, we merged closely related groups to create new categories to represent at least 6% of total observations.

Removing Attributes



IsForeignWorker	percent
no	3.7
yes	96.3

OtherDebtors	percent
yes	9.3
none	90.7

Figure 11: Unbalanced attributes

Despite merging, two-category attributes *IsForeignWorker* and *OtherDebtors* still had unbalanced observations (Figure11). This may not make them good predictors and were consequently removed.

Derived Attributes

We initially considered to derive separate attributes “Sex” and “PersonalStatus” from *SexAndPersonalStatus*. However, PersonalStatus only varied for males and the effects of each *SexAndPersonalStatus* with Risk were observable. Hence, we decided to keep the original attribute.

Imputing Missing Values

There were no missing values. Although, *SavingsBalance* contained the category “unknown/no-savings-account”, we decided to note impute this as it contained a sizeable number of observations and visualisations showed it to be insightful to predicting *Risk* (Figure8).

Transforming Attributes

Numerical attributes were kept as is while categorical attributes were transformed to factors.

Training & Testing Split

The data was split 75/25 into training/testing datasets.

4 Generating & Testing Prediction Models

Cost-sensitive Problem

A false-positive (predicted Good when actually Bad) results in issuing a loan that would be defaulted while a false-negative (predicted Bad when actually Good) results the opportunity cost of not earning the interests on that loan. However, financial loss of a defaulted loan significantly outweighs opportunity cost of earning interest.

```
      Bad        Good
----------------------------
  Bad    0         5
-----------------------
  Good   1         0

the rows represent the actual classification and the columns
the predicted classification.
```

Figure 12: Cost Matrix

The dataset is accompanied by a cost-matrix (Figure12). Since it is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1), our problem is cost-sensitive.

Metrics Used

In evaluating our models, we used two metrics: Cost and Mean Misclassification Error (MMCE).

We created a new metric Cost using MLR's makeCostMeasure() which measures resulting cost, weighted according to the cost matrix. MMCE is defined as mean(response!=truth), measuring the proportion of misclassifications.

With these two measures, we consider both the correctness of classification and cost resulting from misclassifications. With the objective of minimising financial loss, Cost is prioritised. Both lower Cost and MMCE implies better predictions.

Models Selected

We selected 3 models – Logistic Regression, RandomForest, and XGBoost. The Logistic Regression is more interpretable to provide insights and acts as a good baseline model. However, it may not well capture complex, non-linear relationships between predictors and target attributes. To complement this, we selected two ensemble methods – RandomForest and XGBoost – which are more robust to noise and outliers, helping provide more generalise predictions.

Since our classification problem is cost-sensitive, we try two different methods for each model – thresholding and weighting.

Thresholding

The default thresholds of classification is 0.5, meaning the instance would be classified as Good if Prob(Good)>0.5. Since false-Goods are worse than false-Bads, we should be more confident that the customer is actually Good before concluding that they are Good. Thus, we adjust thresholds used in predictions.

Firstly, we trained our 3 models using the training-set and finetuned the RandomForest and XGBoost with these parameters (AppendixD):

RandomForest

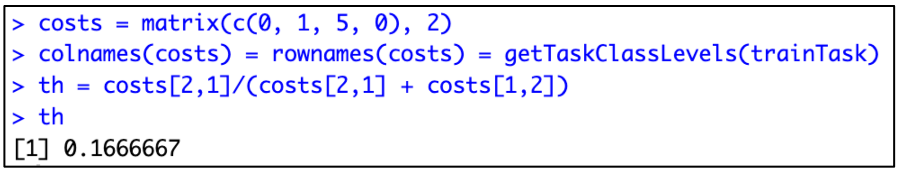
- ntree
- mtry
- nodesize

XGBoost

- nrounds
- max_depth
- lambda
- eta
- subsample
- min_child_weight
- colsample_bytree

Given the large number and range of parameters, we performed a random search, limiting to 100 maximum iterations, giving us optimal parameters (AppendixE), helping us build default models.

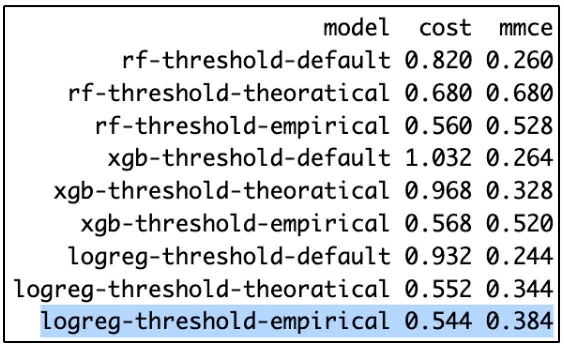
From this we created theoretical and empirical thresholds.



```
> costs = matrix(c(0, 1, 5, 0), 2)
> colnames(costs) = rownames(costs) = getTaskClassLevels(trainTask)
> th = costs[2,1]/(costs[2,1] + costs[1,2])
> th
[1] 0.1666667
```

Figure 13: Theoretical Threshold

The theoretical threshold for the Bad class is calculated to be 0.167 (Figure13). We then also created empirical thresholds which are cost-optimal thresholds for a given learning method based on training data (Sheng & Ling, 2007).

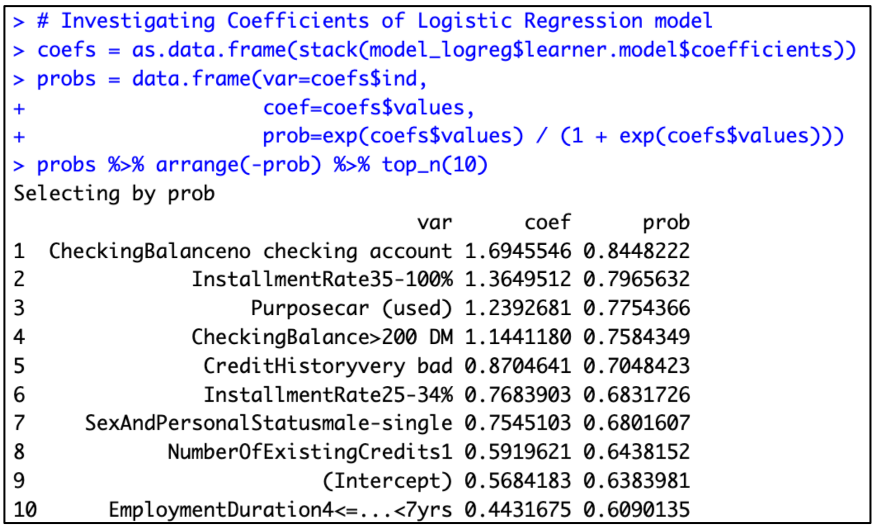


```
                      model  cost  mmce
      rf-threshold-default 0.820 0.260
  rf-threshold-theoratical 0.680 0.680
    rf-threshold-empirical 0.560 0.528
     xgb-threshold-default 1.032 0.264
 xgb-threshold-theoratical 0.968 0.328
   xgb-threshold-empirical 0.568 0.520
  logreg-threshold-default 0.932 0.244
logreg-threshold-theoratical 0.552 0.344
  logreg-threshold-empirical 0.544 0.384
```

Figure 14: Threshold Models

Using default, theoretical and empirical thresholds, we have three variation of predictions of each model (Figure14).

We observe significant improvements in cost from default, theoretical to empirical thresholds while MMCE increases as adjusting thresholds reduce false-Goods but result in more misclassifications overall. With thresholding, we see that the logistic regression model generally performs best in both Cost and MMCE.



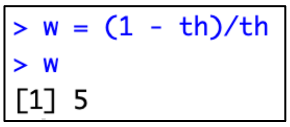
```
> # Investigating Coefficients of Logistic Regression model
> coefs = as.data.frame(stack(model_logreg$learner.model$coefficients))
> probs = data.frame(var=coefs$ind,
+                    coef=coefs$values,
+                    prob=exp(coefs$values) / (1 + exp(coefs$values)))
> probs %>% arrange(-prob) %>% top_n(10)
Selecting by prob
                                 var      coef      prob
1  CheckingBalanceno checking account 1.6945546 0.8448222
2              InstallmentRate35-100% 1.3649512 0.7965632
3                   Purposecar (used) 1.2392681 0.7754366
4              CheckingBalance>200 DM 1.1441180 0.7584349
5               CreditHistoryvery bad 0.8704641 0.7048423
6               InstallmentRate25-34% 0.7683903 0.6831726
7     SexAndPersonalStatusmale-single 0.7545103 0.6801607
8            NumberOfExistingCredits1 0.5919621 0.6438152
9                         (Intercept) 0.5684183 0.6383981
10      EmploymentDuration4<=...<7yrs 0.4431675 0.6090135
```

Figure 15: Probabilities of BadRisk

Coefficients from the Logistic Regression shows customers with CheckingBalance "no-checking-account", InstallmentRate "35-100%" and Purpose "car-(used)" more likely to be Bad-Risk (Figure15).

Weighting

Another cost-sensitive approach is weighting which factors the imbalanced-cost during the training of the models.

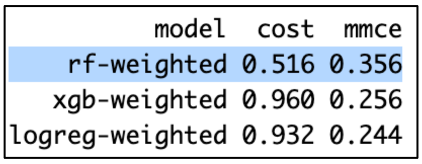


```
> w = (1 - th)/th
> w
[1] 5
```

Figure 16: Theoretical Weight

We can derive the theoretical weight from the theoretical threshold (Figure16).

From this, we create three weighted versions of the above models using the same tuning parameters with an addition of weight. Even though the theoretical weight is 5, we may find other weights to provide better performance depending on each model. As such, in tuning we try weights from 4-8.



```
           model  cost  mmce
     rf-weighted 0.516 0.356
    xgb-weighted 0.960 0.256
logreg-weighted 0.932 0.244
```

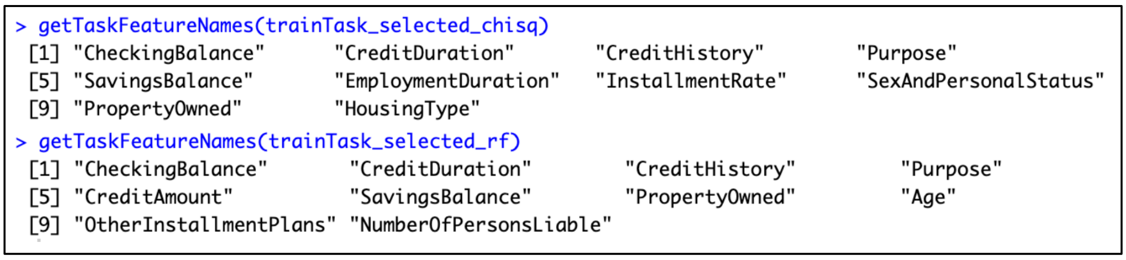
Figure 17: Weighted Models

The RandomForest gave the lowest cost out of the weighted models although it resulted in higher MMCE (Figure17).

Further Model Improvements

RandomForest and XGBoost selects features automatically in creating trees during training. However, the Logistic Regression does not. Given that logistic regression already performed the best with Thresholding, feature selection may further improve its performance.

Using MLR's filterFeartures(), we selected 10 features with the highest chi-squared p-values. However, seeing that the weighted RandomForest had very low Cost, we suspect that feature selection by RandomForest Importance may also be helpful.

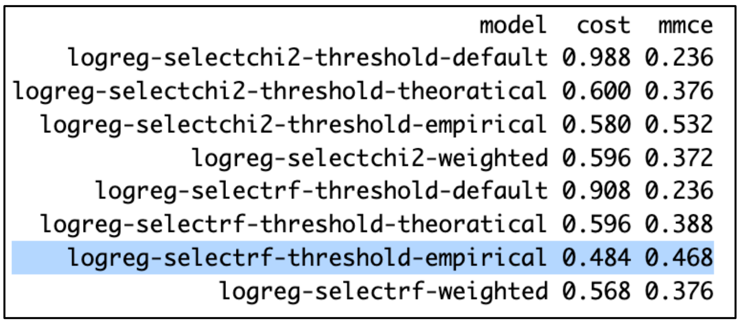


```
> getTaskFeatureNames(trainTask_selected_chisq)
 [1] "CheckingBalance"      "CreditDuration"      "CreditHistory"       "Purpose"
 [5] "SavingsBalance"       "EmploymentDuration"  "InstallmentRate"     "SexAndPersonalStatus"
 [9] "PropertyOwned"        "HousingType"
> getTaskFeatureNames(trainTask_selected_rf)
 [1] "CheckingBalance"       "CreditDuration"       "CreditHistory"        "Purpose"
 [5] "CreditAmount"          "SavingsBalance"       "PropertyOwned"        "Age"
 [9] "OtherInstallmentPlans" "NumberOfPersonsLiable"
```

Figure 18: Feature Selection

As such, we created two variations of feature selected data with different selected attributes (Figure18).

We then retrain Logistic Regression models on these two feature-selected training-sets, using both thresholding and weighting with the same tuning parameters as above.

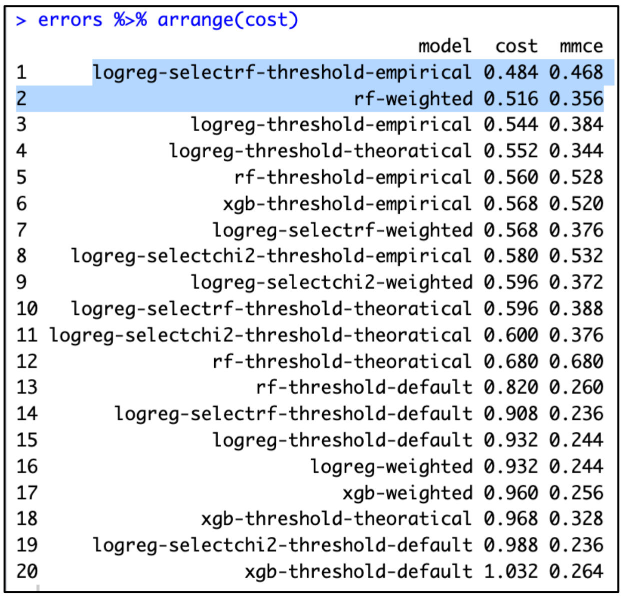


```
                                 model  cost  mmce
   logreg-selectchi2-threshold-default 0.988 0.236
logreg-selectchi2-threshold-theoratical 0.600 0.376
  logreg-selectchi2-threshold-empirical 0.580 0.532
             logreg-selectchi2-weighted 0.596 0.372
      logreg-selectrf-threshold-default 0.908 0.236
  logreg-selectrf-threshold-theoratical 0.596 0.388
    logreg-selectrf-threshold-empirical 0.484 0.468
               logreg-selectrf-weighted 0.568 0.376
```

Figure 19: Models with Feature Selection

Models with features selected with RandomForest Importance indeed resulted in lowest cost and MMCE (Figure19). This model used weight=4.5, ntree=430, mtry=6, nodesize=20.

<u>Best Models & Evaluation</u>



```
> errors %>% arrange(cost)
                                     model  cost  mmce
1      logreg-selectrf-threshold-empirical 0.484 0.468
2                              rf-weighted 0.516 0.356
3                logreg-threshold-empirical 0.544 0.384
4              logreg-threshold-theoratical 0.552 0.344
5                    rf-threshold-empirical 0.560 0.528
6                   xgb-threshold-empirical 0.568 0.520
7                 logreg-selectrf-weighted 0.568 0.376
8    logreg-selectchi2-threshold-empirical 0.580 0.532
9               logreg-selectchi2-weighted 0.596 0.372
10   logreg-selectrf-threshold-theoratical 0.596 0.388
11 logreg-selectchi2-threshold-theoratical 0.600 0.376
12                rf-threshold-theoratical 0.680 0.680
13                     rf-threshold-default 0.820 0.260
14       logreg-selectrf-threshold-default 0.908 0.236
15                logreg-threshold-default 0.932 0.244
16                         logreg-weighted 0.932 0.244
17                            xgb-weighted 0.960 0.256
18               xgb-threshold-theoratical 0.968 0.328
19     logreg-selectchi2-threshold-default 0.988 0.236
20                   xgb-threshold-default 1.032 0.264
```

Figure 20: All Models

The models with the top 2 lowest Costs were the logistic regression with feature selection using RandomForest importance and empirical threshold (*logreg-selectrf-threshold-empirical*) and the weighted RandomForest (*rf-weighted*) (Figure20). We also generally observe that models with low Cost tend to have high MMCE.

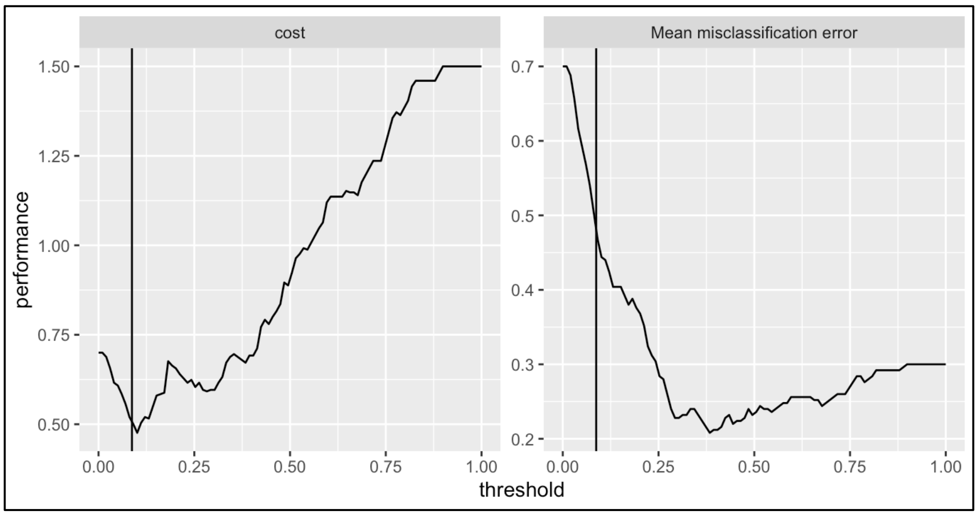
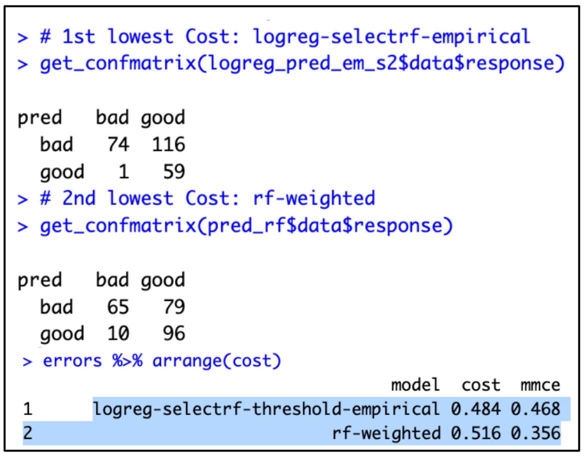


Figure 21: Cost/MMCE-threshold of *logreg-selectrf-threshold-empirical*

When using the Thresholding approach, there is a direct trade-off between Cost and MMCE as chosen threshold decreases (Figure21). Hence for *logreg-selectrf-threshold-empirical*, even though Cost is minimised, MMCE was still high at 0.468, meaning that almost half of the classifications were incorrect.



```
> # 1st lowest Cost: logreg-selectrf-empirical
> get_confmatrix(logreg_pred_em_s2$data$response)

pred   bad good
  bad   74  116
  good   1   59
> # 2nd lowest Cost: rf-weighted
> get_confmatrix(pred_rf$data$response)

pred   bad good
  bad   65   79
  good  10   96
> errors %>% arrange(cost)
                                model  cost  mmce
1      logreg-selectrf-threshold-empirical 0.484 0.468
2                          rf-weighted 0.516 0.356
```

Figure 22: Confusion Matrix

Comparing the confusion matrix (Figure22), we see that *logreg-selectrf-threshold-empirical* is overly-biased towards predicting false-Bads. This resulted in slightly lower cost but significantly higher misclassifications for *logreg-selectrf-threshold-empirical.*

Although *rf-weighted* resulted in more false-Goods, this issue would be less significant in the real population as the dataset was described to have over-sampled instances with Bad-Risk (Groemping, 2019). Therefore, we conclude rf-weighted to be the best model given its low Cost and more reasonable MMCE.

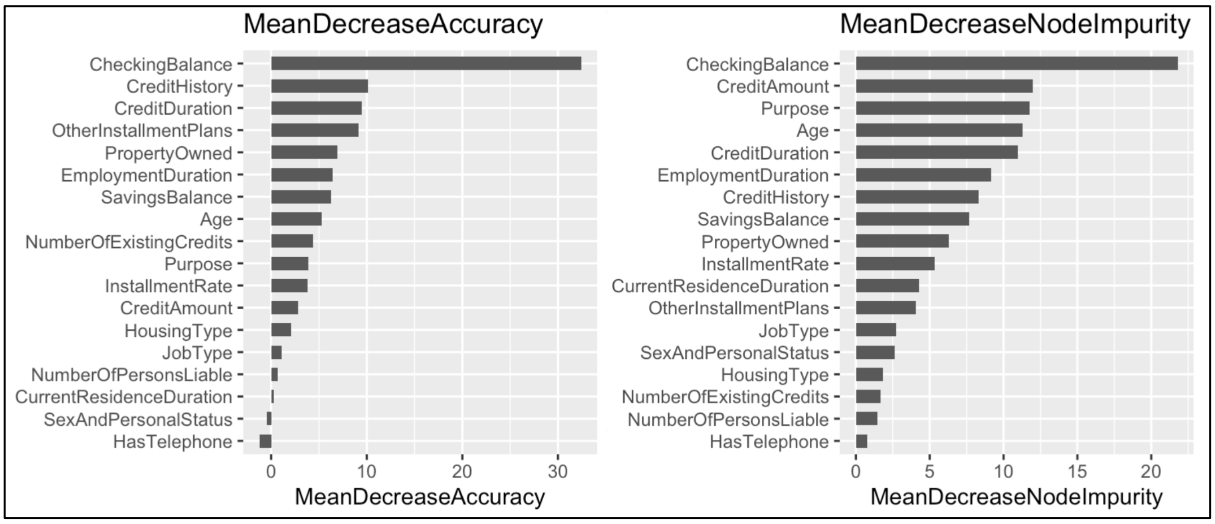


Figure 23: Variable Importance

Variable importance from *rf-weighted* (Figure23) shows CheckingBalance to be most important in predicting Risk. Removing it would cause a 33% mean decrease in accuracy.

5 Problem Conclusions & Recommendations

By implementing the *rf-weighted*, the bank can predict Credit Risk with 64.4% correct classifications. Even though there are models which are more accurate, those result in higher costs. For example, the model most accurate model with 76.4% correct classifications resulted in twice as much cost as our chosen model.

Important Features

Both Logistic Regression and RandomForest model showed *CheckingBalance* to be the most important in predicting Risk. Several other factors such as *CreditHistory*, *CreditDuration*, *OtherInstallmentPlans*, *CreditAmount*, *Purpose* and *Age* were also important. Consequently, the bank should consider to gather and vet this information more rigorously in order to most accurately predict Risk.

Considerations

The objective of minimising cost also results in more false-Bads since we only predict GoodRisk with higher confidence. In practicality, this results in more people being denied loans. This may affect the bank’s reputation to be seen as a strict lender, potentially resulting in fewer applicants.

References

Groemping, U., 2019. South German credit data: Correcting a widely used data set. *Rep. Math., Phys. Chem., Berlin, Germany, Tech. Rep*, *4*, p.2019.

Sheng, V.S. and Ling, C.X., 2007. Roulette sampling for cost-sensitive learning. In *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18* (pp. 724-731). Springer Berlin Heidelberg.

Appendices

Appendix A: Data Source

http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

Appendix B: Data Description & Further Info

http://www1.beuth-hochschule.de/FB_II/reports/Report-2019-004.pdf

Appendix C: Categorical Attributes

Variable name	Level	P2	bad	good
status (A1, M9)	no checking account	1	45.00	19.86
	... < 0 DM	2	35.00	23.43
	0<= ... < 200 DM	3	4.67	7.00
	... >= 200 DM / salary for at least 1 year	4	15.33	49.71
credit_history (A3, M15)	delay in paying off in the past	0	8.33	2.14
	critical account/other credits elsewhere	1	9.33	3.00
	no credits taken/all credits paid back duly	2	56.33	51.57
	existing credits paid back duly till now	3	9.33	8.57
	all credits at this bank paid back duly	4	16.67	34.71
purpose (A4, M16)	others	0	29.67	20.71
	car (new)	1	5.67	12.29
	car (used)	2	19.33	17.57
	furniture/equipment	3	20.67	31.14
	radio/television	4	1.33	1.14
	domestic appliances	5	2.67	2.00
	repairs	6	7.33	4.00
	education	7	0.00	0.00
	vacation	8	0.33	1.14
	retraining	9	11.33	9.00
	business	10	1.67	1.00
savings (A6, M7)	unknown/no savings account	1	72.33	55.14
	... < 100 DM	2	11.33	9.86
	100 <= ... < 500 DM	3	3.67	7.43
	500 <= ... < 1000 DM	4	2.00	6.00
	... >= 1000 DM	5	10.67	21.57
employment_duration (A7, M6)	unemployed	1	7.67	5.57
	< 1 yr	2	23.33	14.57
	1 <= ... < 4 yrs	3	34.67	33.57
	4 <= ... < 7 yrs	4	13.00	19.29
	>= 7 yrs	5	21.33	27.00
installment_rate (A8, M8)	>= 35	1	11.33	14.57
	25 <= ... < 35	2	20.67	24.14
	20 <= ... < 25	3	15.00	16.00
	< 20	4	53.00	45.29
personal_status_sex (A9, M17)	male : divorced/separated	1	6.67	4.29
	female : non-single or male : single	2	36.33	28.71
	male : married/widowed	3	48.67	57.43
	female : single	4	8.33	9.57
other_debtors (A10, M10)	none	1	90.67	90.71
	co-applicant	2	6.00	3.29
	guarantor	3	3.33	6.00
present_residence (A11, M5)	< 1 yr	1	12.00	13.43
	1 <= ... < 4 yrs	2	32.33	30.14
	4 <= ... < 7 yrs	3	14.33	15.14
	>= 7 yrs	4	41.33	41.29

Variable name	Level	P2	bad	good
property (A12, M12)	unknown / no property	1	20.00	31.71
	car or other	2	23.67	23.00
	building soc. savings agr./life insurance	3	34.00	32.86
	real estate	4	22.33	12.43
other_installment_plans (A14, M13)	bank	1	19.00	11.71
	stores	2	6.33	4.00
	none	3	74.67	84.29
housing (A15, M14)	for free	1	23.33	15.57
	rent	2	62.00	75.43
	own	3	14.67	9.00
number_credits (A16, M4)	1	1	66.67	61.86
	2-3	2	30.67	34.43
	4-5	3	2.00	3.14
	>= 6	4	0.67	0.57
job (A17, M11)	unemployed/unskilled - non-resident	1	2.33	2.14
	unskilled - resident	2	18.67	20.57
	skilled employee/official	3	62.00	63.43
	manager/self-empl./highly qualif. employee	4	17.00	13.86
people_liable (A18, M20)	3 or more	1	15.33	15.57
	0 to 2	2	84.67	84.43
telephone (A19, M19)	no	1	62.33	58.43
	yes (under customer name)	2	37.67	41.57
foreign_worker (A20, M18)	yes	1	1.33	4.71
	no	2	98.67	95.29

Appendix D: Tuning Parameters

```
rf_param = makeParamSet(
  makeIntegerParam("ntree",lower = 50, upper = 500),
  makeIntegerParam("mtry", lower = 3, upper = 15),
  makeIntegerParam("nodesize", lower = 10, upper = 50)
)
```

```
xg_ps = makeParamSet(
  makeIntegerParam("nrounds",lower=200,upper=600),
  makeIntegerParam("max_depth",lower=3,upper=20),
  makeNumericParam("lambda",lower=0.55,upper=0.60),
  makeNumericParam("eta", lower=0.001, upper=0.5),
  makeNumericParam("subsample", lower=0.10, upper=0.80),
  makeNumericParam("min_child_weight",lower=1,upper=5),
  makeNumericParam("colsample_bytree",lower=0.2,upper=0.8)
)
```

Appendix E: Optimal Tuning Parameters

```
> rf_tune$y
cost.test.mean mmce.test.mean
         0.576          0.400
> rf_tune$x
$wcw.weight
[1] 4.5

$ntree
[1] 430

$mtry
[1] 6

$nodesize
[1] 20
```

```
> xg_tune$y
cost.test.mean mmce.test.mean
         0.856          0.264
> xg_tune$x
$wcw.weight
[1] 5.5

$nrounds
[1] 275

$max_depth
[1] 5

$lambda
[1] 0.5686219

$eta
[1] 0.1608161

$subsample
[1] 0.770873

$min_child_weight
[1] 3.786923

$colsample_bytree
[1] 0.7768955
```