Analysis

Elodie Kwan and Katia Voltz

2022-04-26

Methodology

In this section we will talk about the methodology that has been used and the different models analysis that has been conducted.

Traning set and Test set

First of all we started by splitting our dataset into 2 sets: **training set** (`German_credit.tr`) and **test set** (`German_credit.te`). We do not forget to take the first variable **OBS.** out as it represents the index number for each observation. These two sets will allow us to train some models on the **training set** and then test the accuracy of the model fit on the **test set**.

Balancing the dataset

Then, we applied the balancing data technique in order to improve the predictions of **Good Credit** and **Bad Credit**, since we have more observations on the **Good Credit**.

Indeed, we observe that the "Good Credit" (1) response appears **527** times in the training set and "Bad Credit" (0) **223**, two times less. Since there are many more "Good Credit" than "Bad Credit", any model favors the prediction of the "Good Credit". It results a good accuracy but the specificity is low, as well as the balanced accuracy.

Sub-sampling Balancing using sub-sampling consists of taking all the cases in the smallest class (here "Bad Credit") and extract at random the same amount of cases in the largest category (here "Good").

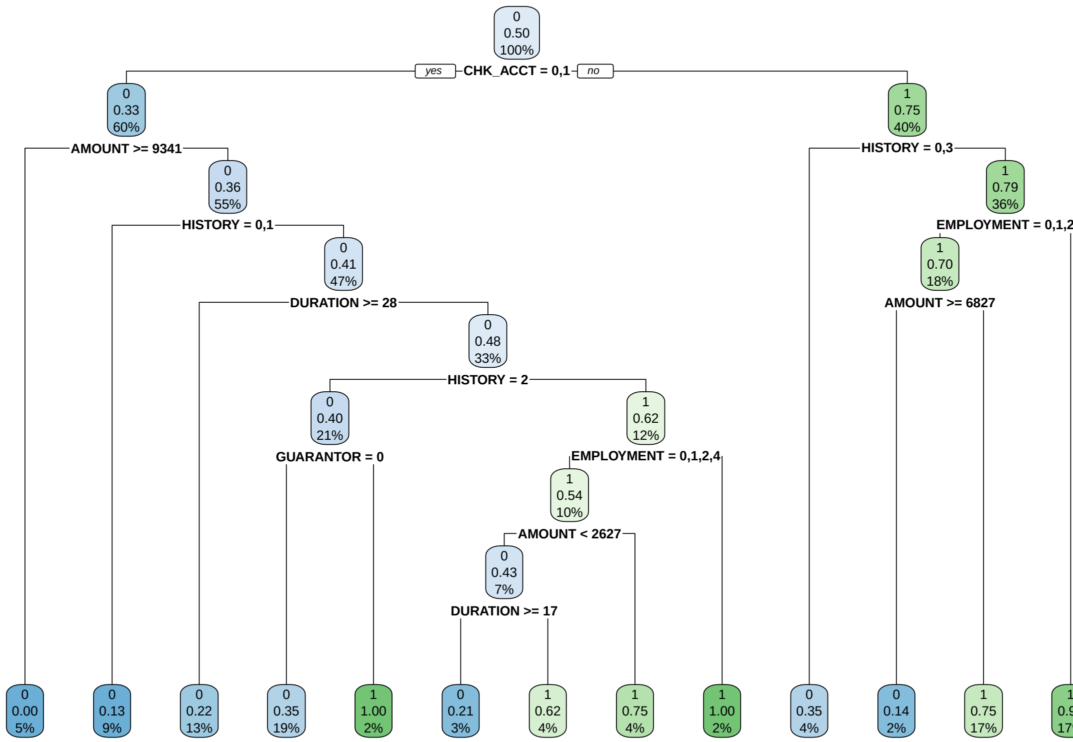
The **training set** is now balanced, we have 223 observations for both "Good Credit" (1) and "Bad Credit" (0). The new balanced training set is called `German_Credit.tr.subs`.

Models Fitting

Once we had our training set and test set, we could fit some models and compare them with together to choose the best model.

1. Classification Tree (Decision Tree) We first started with a decision tree and more specifically we chose the classification tree as we want to classify the applicants. The model was build on our previously balanced training set `German_Credit.tr.subs`. We used the R function `rpart`.

We obtained the following large tree.



We could see that among the 31 explanatory variables, this model uses 6 of them: **CHK_ACCT**, **AMOUNT**, **HISTORY**, **DURATION**, **GUARANTOR** and **EMPLOYMENT**.

Table 1: Confusion Matrix of the Big classification tree

	Bad credit risk	Good credit risk
Bad credit risk	58	70
Good credit risk	19	103

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7532468            0.5953757            0.4531250
##       Neg Pred Value            Precision               Recall
##            0.8442623            0.4531250            0.7532468
##                   F1           Prevalence       Detection Rate
##            0.5658537            0.3080000            0.2320000
## Detection Prevalence    Balanced Accuracy
##            0.5120000            0.6743112
```

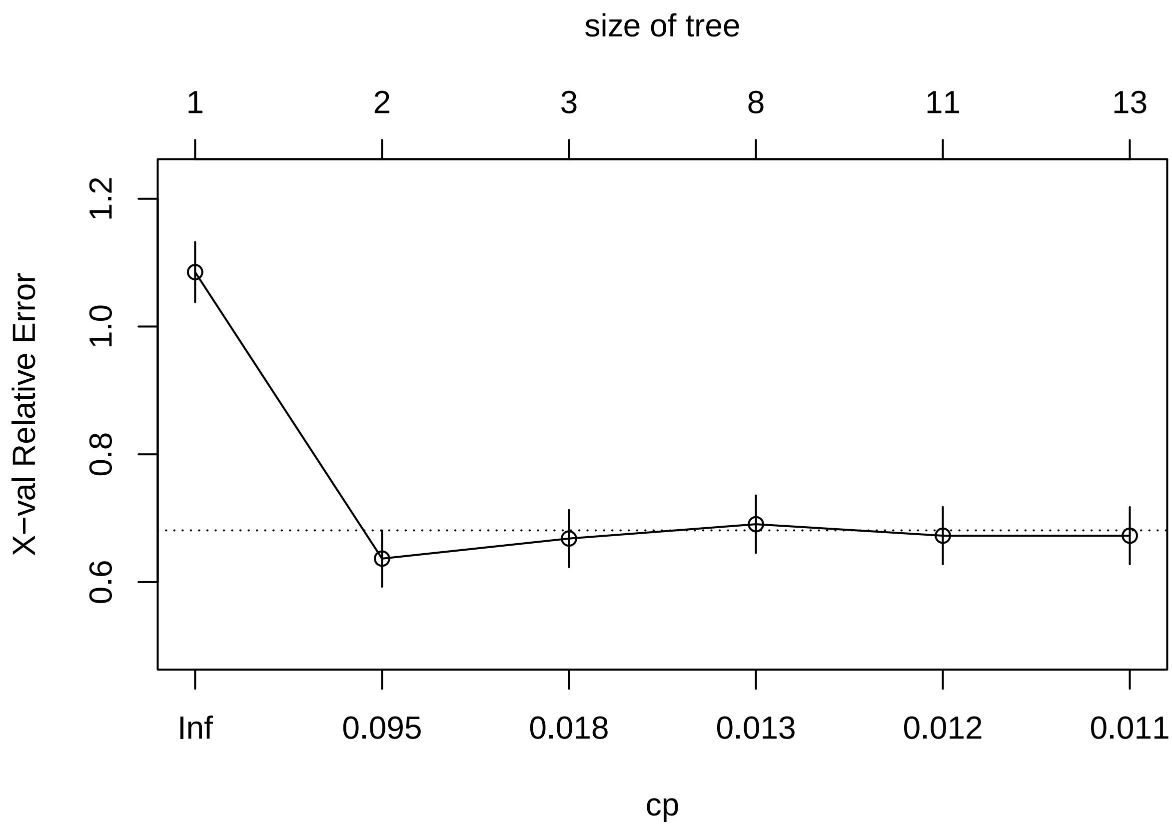
We first have an insight on how well it predict the test set (`German_credit.te`). We recall that 0 means a "Bad Credit" risk and 1 means a "Good Credit" risk. It seems that this model has difficulty to predict the "Bad Credit" risk applicants. Indeed from the table have 70 observations that were misclassified as being "Bad credit" while it was in fact a "Good credit".

As the tree is quite big and we want to know if we can prune it. To do so, we decided to use the `printcp` and `plotcp` commands and choose the best **cp** (complexity parameter) value to prune our tree.

Pruning the tree

```
##
## Classification tree:
## rpart(formula = RESPONSE ~ ., data = German_Credit.tr.subs, method = "class")
```

```
##
## Variables actually used in tree construction:
## [1] AMOUNT     CHK_ACCT   DURATION   EMPLOYMENT GUARANTOR  HISTORY
##
## Root node error: 223/446 = 0.5
##
## n= 446
##
##         CP nsplit rel error  xerror     xstd
## 1 0.399103      0   1.00000 1.08520 0.047179
## 2 0.022422      1   0.60090 0.63677 0.044117
## 3 0.014574      2   0.57848 0.66816 0.044668
## 4 0.011958      7   0.48430 0.69058 0.045028
## 5 0.011211     10   0.44843 0.67265 0.044742
## 6 0.010000     12   0.42601 0.67265 0.044742
```

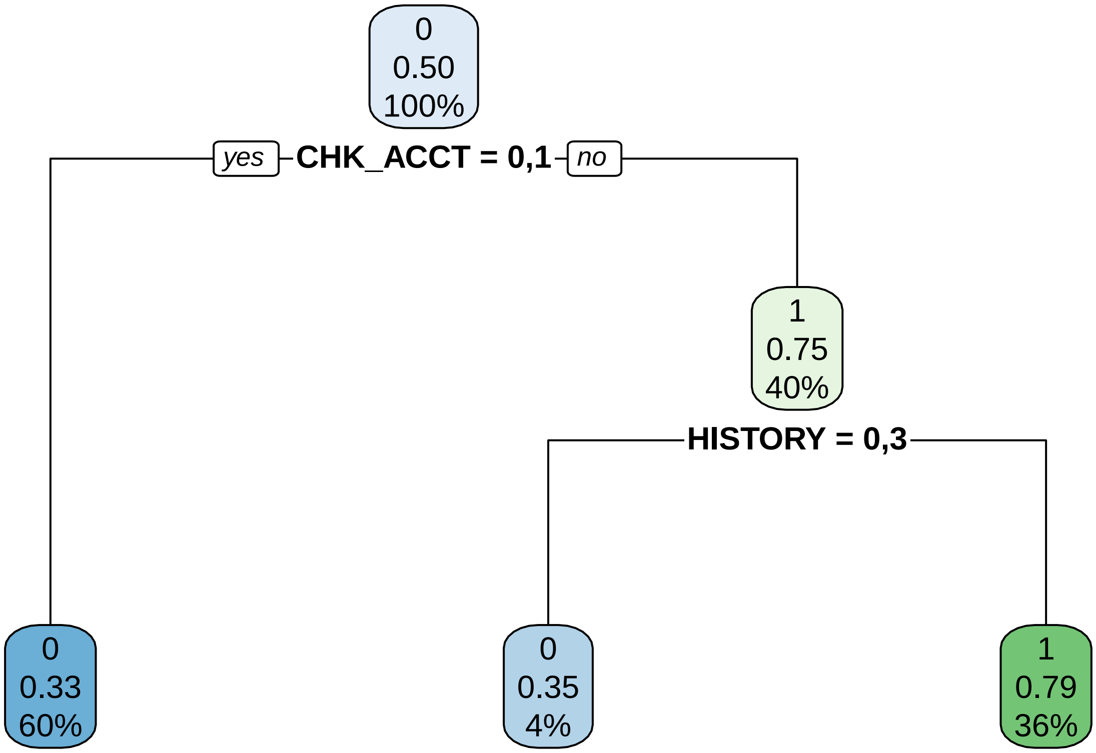


From the list of **cp** (complexity parameter), we would choose the line that has the lowest cross validation error. This can be seen on the column **xerror**. So the best cp would be 0.022422 with one split.

From the graph, we can identify that, according to the 1-SE rule, the tree with 2 and 3 are equivalent. The tree with 3 nodes should be preferred. It appears below the dotted-line.

The value of cp can be chosen arbitrarily between 0.018 and 0.095. So we decided to go with the suggested cp of 0.022 from the summary.

With these value, we obtain a very small tree.



This pruned decision tree with a cp of 0.022 uses the variables **CHK_ACCT** and **HISTORY**.

Using this pruned tree, we can computed the prediction and build a confusion matrix to see the performance of the model.

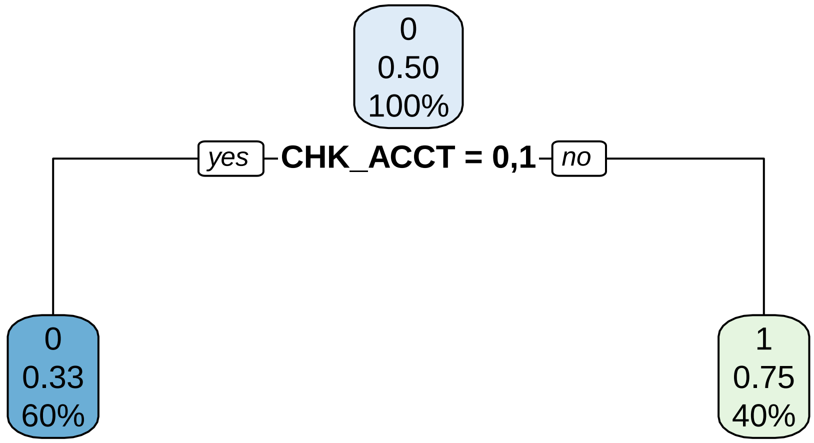
Table 2: Confusion Matrix of the Pruned classification tree

	Bad credit risk	Good credit risk
Bad credit risk	63	95
Good credit risk	14	78

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8181818            0.4508671            0.3987342
##       Neg Pred Value            Precision               Recall
##            0.8478261            0.3987342            0.8181818
##                   F1           Prevalence       Detection Rate
##            0.5361702            0.3080000            0.2520000
## Detection Prevalence    Balanced Accuracy
##            0.6320000            0.6345244
```

Here again, this model has difficulty to predict the "Bad credit" cases. The model also decreased in its precision values.

We also decided to look at what would an automatically pruned using 1-SE rule would give us and whether or not it is better than the pruned tree we made by looking at the cp.



Here, only the variable **CHK__ACCT** is used. As we prune the tree more information are lost.

Table 3: Confusion Matrix of the Autoruned classification tree

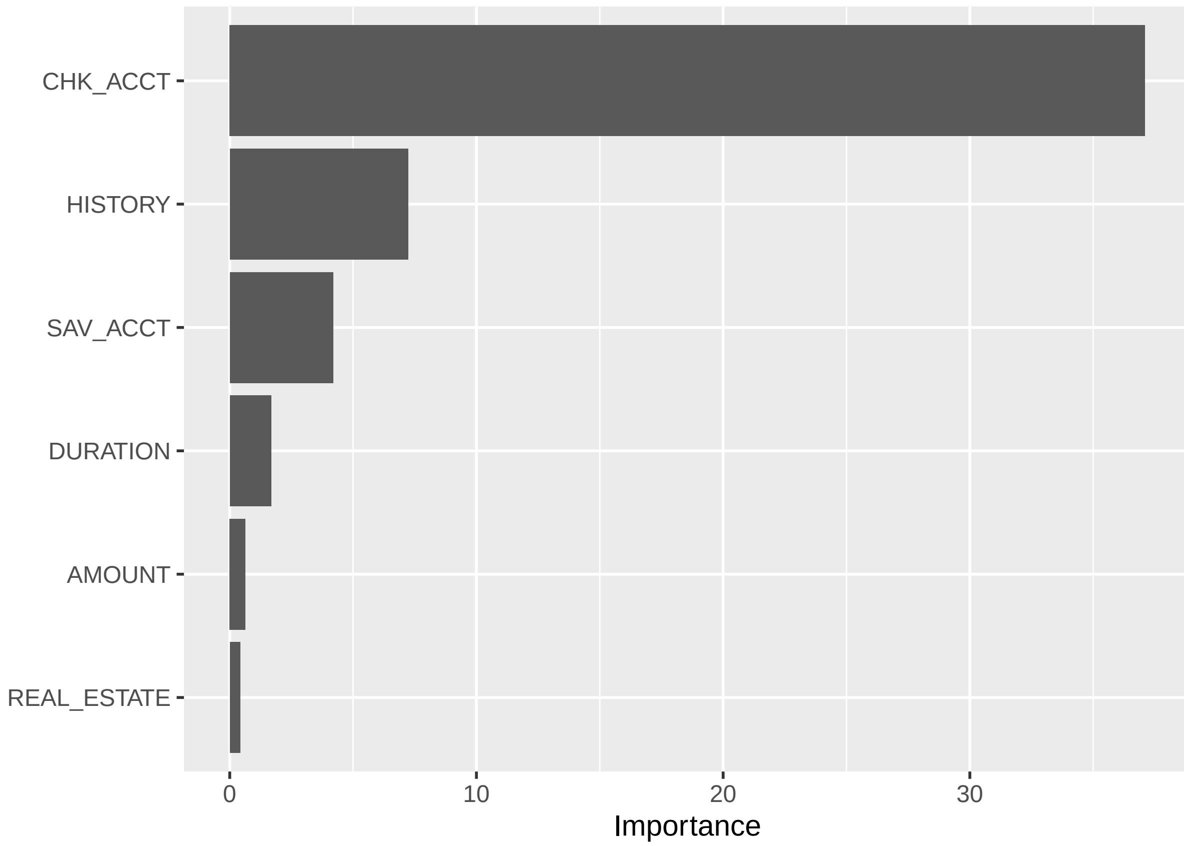
	Bad credit risk	Good credit risk
Bad credit risk	61	88
Good credit risk	16	85

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7922078            0.4913295            0.4093960
##       Neg Pred Value            Precision               Recall
##            0.8415842            0.4093960            0.7922078
##                   F1           Prevalence       Detection Rate
##            0.5398230            0.3080000            0.2440000
## Detection Prevalence    Balanced Accuracy
##            0.5960000            0.6417686
```

Again, it seems that in general the classification trees we perfomed have difficulty to predict the "Bad credit" cases.

Variable importance of the classification tree Then we analysed the variable importance of one of the models. We decided to compute the variable importance of the **pruned classisication tree** `german.ct.prune` by applying the function `varImp` on the model.

It is summarized below on the plot.



From this plot, we see that the variables that influences the most are **CHK_ACCT**, **HISTORY**, **SAV_ACCT**, **DURATION**, **AMOUNT** and **REAL_ESTATE**. They are not exactly the same as the one we saw above.

The variable **CHK_ACCT** and **HISTORY** were already noticed though.

2. Logistic Regression The next model we performed is a logistic regression.

```
##
## Call:
## glm(formula = RESPONSE ~ ., family = binomial, data = German_Credit.tr.subs)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.34578  -0.68043   0.00049   0.65178   2.74937
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.1911402  1.7958756   0.663 0.507161
## CHK_ACCT1          0.5692882  0.3363406   1.693 0.090533 .
## CHK_ACCT2          0.8404451  0.5339512   1.574 0.115485
## CHK_ACCT3          2.4337691  0.3770606   6.455 1.09e-10 ***
## DURATION          -0.0123731  0.0142153  -0.870 0.384078
## HISTORY1          -1.0734853  0.8514386  -1.261 0.207384
## HISTORY2           0.0865599  0.6747882   0.128 0.897930
## HISTORY3          -0.0598560  0.7410028  -0.081 0.935619
## HISTORY4           1.1072483  0.6576414   1.684 0.092246 .
## NEW_CAR1          -0.4538649  0.5853211  -0.775 0.438096
```

```
## USED_CAR1          1.6322817  0.7540134   2.165 0.030404 *
## FURNITURE1         0.0509645  0.6182782   0.082 0.934305
## RADIO.TV1          0.5261147  0.5896893   0.892 0.372291
## EDUCATION1         0.5441469  0.7499724   0.726 0.468111
## RETRAINING1       -0.4293160  0.6787931  -0.632 0.527080
## AMOUNT            -0.0002155  0.0000739  -2.916 0.003550 **
## SAV_ACCT1          0.6181742  0.4475399   1.381 0.167195
## SAV_ACCT2         -0.2531524  0.5541205  -0.457 0.647776
## SAV_ACCT3          0.7292579  0.6813687   1.070 0.284492
## SAV_ACCT4          1.4221687  0.4243610   3.351 0.000804 ***
## EMPLOYMENT1        0.7574673  0.7956778   0.952 0.341108
## EMPLOYMENT2        1.4785839  0.7640267   1.935 0.052959 .
## EMPLOYMENT3        1.9691166  0.7947873   2.478 0.013229 *
## EMPLOYMENT4        1.8560330  0.7511387   2.471 0.013475 *
## INSTALL_RATE      -0.3367533  0.1411404  -2.386 0.017035 *
## MALE_DIV1         -0.5653453  0.5705857  -0.991 0.321775
## MALE_SINGLE1       0.1618525  0.3327207   0.486 0.626647
## MALE_MAR_or_WID1  -0.5551862  0.5312986  -1.045 0.296041
## CO.APPLICANT1     -0.6994379  0.6920599  -1.011 0.312179
## GUARANTOR1         1.7126786  0.6556150   2.612 0.008993 **
## PRESENT_RESIDENT2 -1.1195205  0.4773294  -2.345 0.019008 *
## PRESENT_RESIDENT3 -0.2590309  0.5313455  -0.487 0.625904
## PRESENT_RESIDENT4 -0.9082582  0.4793144  -1.895 0.058104 .
## REAL_ESTATE1      -0.0137202  0.3384983  -0.041 0.967669
## PROP_UNKN_NONE1   -1.4578770  0.6505748  -2.241 0.025032 *
## AGE                0.0167050  0.0141041   1.184 0.236255
## OTHER_INSTALL1    -0.6758552  0.3404321  -1.985 0.047113 *
## RENT1             -1.2066453  0.8244600  -1.464 0.143315
## OWN_RES1          -0.4707135  0.7665544  -0.614 0.539173
## NUM_CREDITS       -0.3634820  0.3011721  -1.207 0.227474
## JOB1              -0.7402802  1.1619781  -0.637 0.524069
## JOB2              -1.2142377  1.1317833  -1.073 0.283337
## JOB3              -1.4358446  1.1604352  -1.237 0.215964
## NUM_DEPENDENTS     0.1270172  0.3832474   0.331 0.740325
## TELEPHONE1         0.6259633  0.3143236   1.991 0.046430 *
## FOREIGN1           1.2496315  0.8543880   1.463 0.143576
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.29  on 445  degrees of freedom
## Residual deviance: 390.97  on 400  degrees of freedom
## AIC: 482.97
##
## Number of Fisher Scoring iterations: 5
```

We see that a lot of variables are not statistically significant for the model so we can think of a model reduction.

Before doing a reduction of the model, we fitted the model and predicted on the test set.

Table 4: Confusion Matrix of the Logistic Regression

	Bad credit risk	Good credit risk
Bad credit risk	49	46
Good credit risk	28	127

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6363636            0.7341040            0.5157895
##       Neg Pred Value            Precision               Recall
##            0.8193548            0.5157895            0.6363636
##                   F1           Prevalence       Detection Rate
##            0.5697674            0.3080000            0.1960000
## Detection Prevalence    Balanced Accuracy
##            0.3800000            0.6852338
```

From the confusion matrix, we see again the the model has difficulty to predict the "Bad credit" althought the wronged classified amount of observation is lower when it comes to the "Bad credit" one.

Variable selection and interpretation with step method (AIC criteria) In order to reduce the logistic regression we used a stepwise variable selection. This has been done with the command `step`.

The final reduced model is as follow.

```
##
## Call:
## glm(formula = RESPONSE ~ CHK_ACCT + HISTORY + NEW_CAR + USED_CAR +
##     RETRAINING + AMOUNT + SAV_ACCT + EMPLOYMENT + INSTALL_RATE +
##     GUARANTOR + PRESENT_RESIDENT + PROP_UNKN_NONE + AGE + OTHER_INSTALL +
##     RENT + TELEPHONE + FOREIGN, family = binomial, data = German_Credit.tr.subs)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.39343  -0.68768  -0.02628   0.71315   2.60726
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.6339113  1.1570626  -0.548 0.583786
## CHK_ACCT1          0.5970566  0.3291383   1.814 0.069678 .
## CHK_ACCT2          1.1123874  0.5042812   2.206 0.027392 *
## CHK_ACCT3          2.4109175  0.3597629   6.701 2.06e-11 ***
## HISTORY1          -0.6393459  0.8007632  -0.798 0.424626
## HISTORY2           0.3153810  0.6295178   0.501 0.616379
## HISTORY3           0.0245812  0.7411154   0.033 0.973541
## HISTORY4           1.0638624  0.6476870   1.643 0.100475
## NEW_CAR1          -0.7159178  0.3141611  -2.279 0.022678 *
## USED_CAR1          1.3489217  0.5579072   2.418 0.015614 *
## RETRAINING1       -0.8489518  0.4619050  -1.838 0.066072 .
## AMOUNT            -0.0002631  0.0000595  -4.421 9.81e-06 ***
## SAV_ACCT1          0.5675624  0.4173990   1.360 0.173906
## SAV_ACCT2         -0.0693577  0.5399345  -0.128 0.897788
## SAV_ACCT3          0.5603771  0.6437202   0.871 0.384011
## SAV_ACCT4          1.3592948  0.4069584   3.340 0.000837 ***
## EMPLOYMENT1        0.5542570  0.6967423   0.795 0.426324
## EMPLOYMENT2        1.2338686  0.6524020   1.891 0.058588 .
```

```
## EMPLOYMENT3        1.7999683  0.6887566   2.613 0.008966 **
## EMPLOYMENT4        1.5521376  0.6518729   2.381 0.017264 *
## INSTALL_RATE      -0.3278020  0.1249721  -2.623 0.008716 **
## GUARANTOR1         1.6927223  0.6068573   2.789 0.005282 **
## PRESENT_RESIDENT2 -1.1117822  0.4641005  -2.396 0.016595 *
## PRESENT_RESIDENT3 -0.3408387  0.5041109  -0.676 0.498966
## PRESENT_RESIDENT4 -0.7613632  0.4531619  -1.680 0.092935 .
## PROP_UNKN_NONE1   -1.0532655  0.3848454  -2.737 0.006203 **
## AGE                0.0181856  0.0128738   1.413 0.157769
## OTHER_INSTALL1    -0.6281982  0.3256821  -1.929 0.053747 .
## RENT1             -0.8736712  0.3412119  -2.560 0.010452 *
## TELEPHONE1         0.5251823  0.2863172   1.834 0.066614 .
## FOREIGN1           1.2896516  0.8049248   1.602 0.109111
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 618.29  on 445  degrees of freedom
## Residual deviance: 403.37  on 415  degrees of freedom
## AIC: 465.37
##
## Number of Fisher Scoring iterations: 5
```

The variables that have been removed are : **FURNITURE**, **RADIO.TV**, **EDUCATION**, **RETRAINING**, **MALE_DIV**, **MALE_SINGLE**, **MALE_MAR_or_WID**, **CO.APPLICANT**, **REAL_ESTATE**, **OWN_RES**, **NUM_CREDITS**, **JOB** and **NUM_DEPENDENTS**

In the end, we get the most significant model:

$$RESPONSE = -0.6339113 + 0.5970566 * CHK_{ACCT1} + 1.1123874 * CHK_{ACCT2} + 2.4109175 * CHK_{ACCT3} - 0.6393459 * HIST$$

$$p = (e^{RESPONSE})/(1 + e^{RESPONSE})$$

It means that :

- The predicted probability of being a good applicant for **CHCK_ACCT3** is higher than for **CHK_ACCT0** (and also higher than for **CHK_ACCT1** and **CHK_ACCT2**).
- The predicted probability of being a good applicant for **HISTORY1** is lower than for **HISTORY0**.
- The predicted probability of being a good applicant for **HISTORY4** is higher than for **HISTORY0** (and also higher than for **HISTORY2** and **HISTORY3**).
- The predicted probability of being a good applicant for **NEW_CAR1** is lower than for **NEW_CAR0**.
- The predicted probability of being a good applicant for **USED_CAR1** is higher than for **USED_CAR0**.
- The predicted probability of being a good applicant for **RETRAINING1** is lower than for **RETRAINING0**.
- **AMOUNT** is negatively associated with **RESPONSE**.
- The predicted probability of being a good applicant for **SAV_ACCT4** is higher than for **SAV_ACCT0** (and also higher than for **SAV_ACCT1** and **SAV_ACCT3**).
- The predicted probability of being a good applicant for **SAV_ACCT2** lower than for **SAV_ACCT0**.
- The predicted probability of being a good applicant for **EMPLOYMENT3** is higher than for **Employment0** (and also higher than for **EMPLOYMENT1**, **EMPLOYMENT2** and **EMPLOYMENT4**).
- **INSTALL_RATE** is negatively associated with **RESPONSE**.

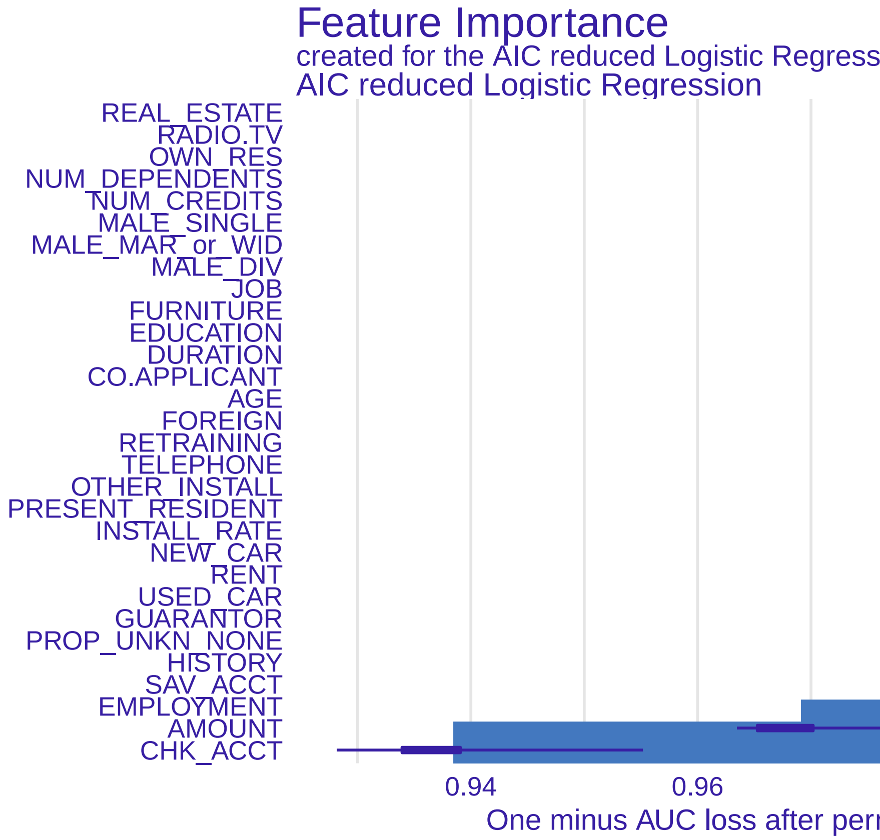
- The predicted probability of being a good applicant for **GUARANTOR1** is higher than for **GUARANTOR0**.
- The predicted probability of being a good applicant for **PRESENT_RESIDENT2** is lower than for **PRESENT_RESIDENT0** (and also lower than **PRESENT_RESIDENT3** and **PRESENT_RESIDENT4**).
- The predicted probability of being a good applicant for **PROP_UNKN_NONE1** is lower than for **PROP_UNKN_NONE0**.
- **AGE** is positively associated with **RESPONSE**.
- The predicted probability of being a good applicant for **OTHER_INSTALL1** is lower than for **OTHER_INSTALL0**.
- The predicted probability of being a good applicant for **RENT1** is lower than for **RENT0**.
- The predicted probability of being a good applicant for **TELEPHONE1** is higher than for **TELEPHONE0**.
- The predicted probability of being a good applicant for **FOREIGN1** is higher than for **FOREIGN0**.

Table 5: Confusion Matrix of the AIC reduced Logistic regression

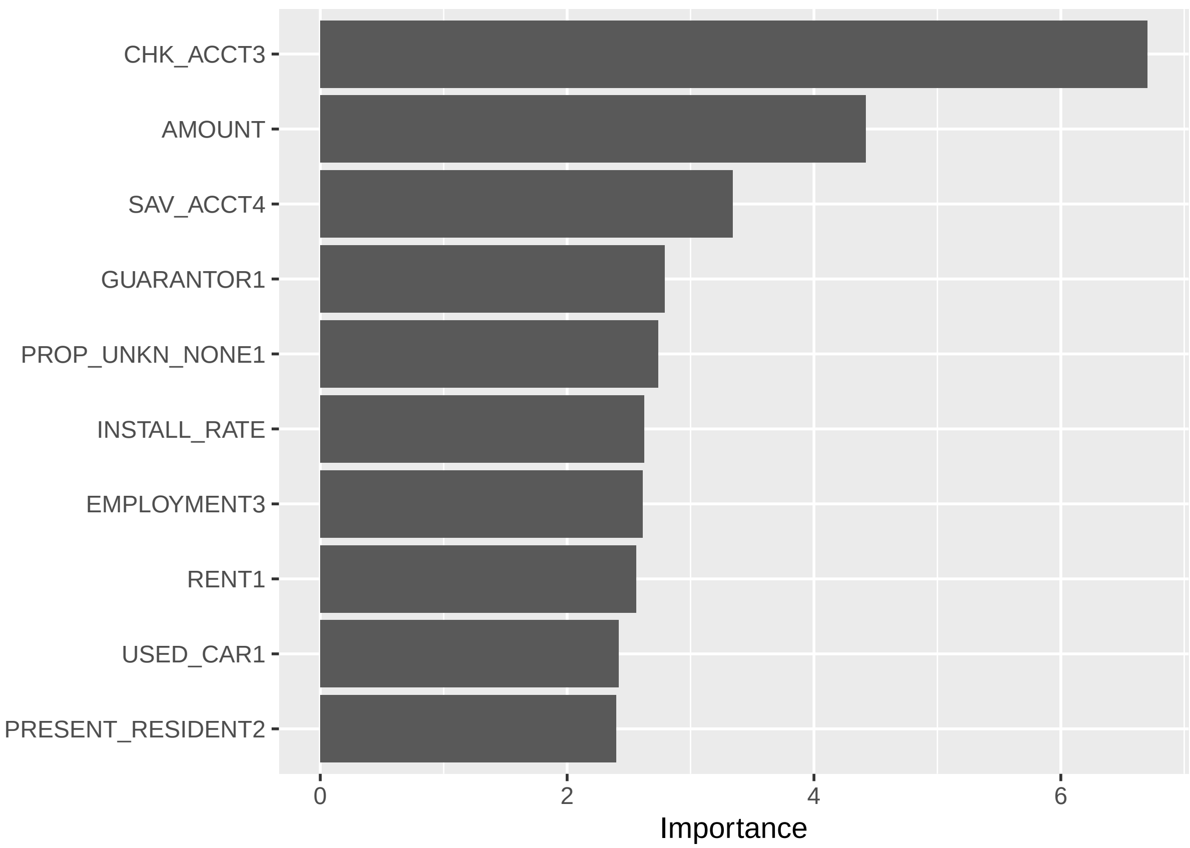
	Bad credit risk	Good credit risk
Bad credit risk	50	52
Good credit risk	27	121

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6493506            0.6994220            0.4901961
##       Neg Pred Value            Precision               Recall
##            0.8175676            0.4901961            0.6493506
##                   F1           Prevalence       Detection Rate
##            0.5586592            0.3080000            0.2000000
## Detection Prevalence    Balanced Accuracy
##            0.4080000            0.6743863
```

From this point we might think that the problem for the difficulty to predict the “Bad Credit” cases is not due to the models but rather the data itself. We might still want to perfom other models to make sure of our intuition.



Variable importance for logistic regression



Listed above are the most important variables for the logarithmic regression we reduced. Here again, the **CHK_ACCT** variable differentiate itself from the others in term of importance in the model prediction.

3.a K-Nearest Neighbor (K=2) To perform a k-nearest neighbor method, we do not need to balance the data so we will use the unbalanced training set.

We first try to model it using a 2-NN (with Euclidean distance). Note that the model is fitting on the training set and the predictions are computed on the test set.

Table 6: Confusion Matrix of the 2-Nearest neighbor

	Bad credit risk	Good credit risk
Bad credit risk	21	45
Good credit risk	56	128

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.2727273            0.7398844            0.3181818
##       Neg Pred Value            Precision               Recall
##            0.6956522            0.3181818            0.2727273
##                   F1           Prevalence       Detection Rate
##            0.2937063            0.3080000            0.0840000
## Detection Prevalence    Balanced Accuracy
##            0.2640000            0.5063058
```

The table is read as follow :

- We predicted 21 Bad credits and there were indeed 21 observed Bad credits. But the prediction misjudges 45 good credits by predicting them as bad credits.
- We predicted 128 Good credits as it was in fact a Good credits but 56 where predicted as Good while it was in fact Bad.

So again, this model also have difficulties to predict the "Bad Credit" cases.

The prediction is not perfect. We need to try to improve the prediction by changing K at that point. Therefore, we use K=3.

3.b K-Nearest Neighbor (K=3)

Table 7: Confusion Matrix of the 3-Nearest neighbor

	Bad credit risk	Good credit risk
Bad credit risk	58	70
Good credit risk	19	103

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.1818182            0.8381503            0.3333333
##       Neg Pred Value            Precision               Recall
##            0.6971154            0.3333333            0.1818182
##                   F1           Prevalence       Detection Rate
##            0.2352941            0.3080000            0.0560000
## Detection Prevalence    Balanced Accuracy
##            0.1680000            0.5099842
```

The table is read as follow :

- We predicted 58 Bad credits and they were indeed observed Bad credits. But the prediction misjudges 70 good credits by predicting them as being bad credits.
- We predicted 103 Good credits as it was in fact a Good credits but 19 where predicted as Good while it was in fact Bad.

Again it seems not to have improved anything, the **F-measure** even seems to have decreased a little bit.

4. Linear Support Vector Machine The next model is the linear Support Vector Machine.

```
##
## Call:
## svm(formula = RESPONSE ~ ., data = German_Credit.tr.subs, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  246
```

Table 8: Confusion Matrix of the Linear support vector machine

	Bad credit risk	Good credit risk
Bad credit risk	50	50
Good credit risk	27	123

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6493506            0.7109827            0.5000000
##       Neg Pred Value            Precision               Recall
##            0.8200000            0.5000000            0.6493506
##                   F1           Prevalence       Detection Rate
##            0.5649718            0.3080000            0.2000000
## Detection Prevalence    Balanced Accuracy
##            0.4000000            0.6801667
```

This model seems to have improved a little bit our ability to predict the "Bad Credit" cases. As half of the "Bad Credit" predicted by the model are correct.

Tunning the hyperparameters of Linear SVM We want to select the good hyperparameters for our linear SVM.

```
## Support Vector Machines with Linear Kernel
##
## 446 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 402, 400, 402, 401, 402, 402, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7264361  0.4530209
```

```
##
## Tuning parameter 'C' was held constant at a value of 1
```

We see that we have a good accuracy (0.72).

```
## Support Vector Machines with Linear Kernel
##
## 446 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 402, 400, 402, 401, 402, 402, ...
## Resampling results across tuning parameters:
##
##   C      Accuracy   Kappa
##   1e-02  0.7264339  0.4532044
##   1e-01  0.7286056  0.4575791
##   1e+00  0.7264361  0.4530209
##   1e+01  0.7108278  0.4216992
##   1e+02  0.7130501  0.4261940
##   1e+03  0.7197672  0.4397558
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.1.
```

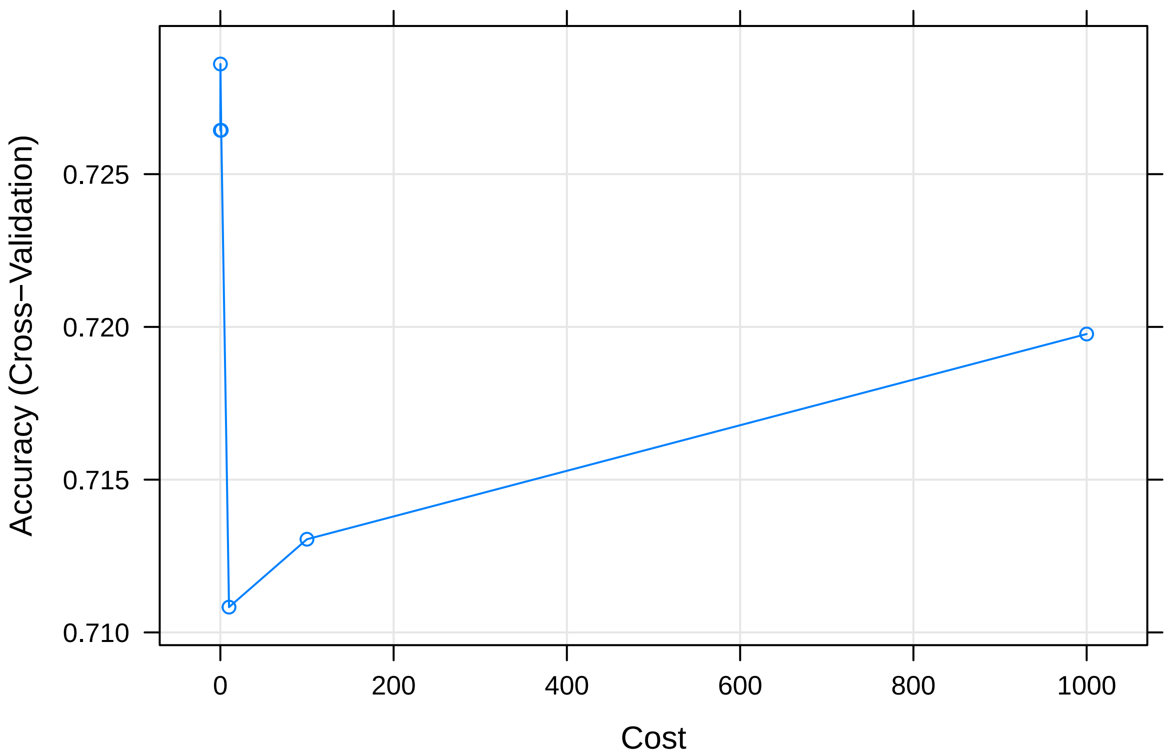


Table 9: Confusion Matrix of the Tuned linear support vector machine

	Bad credit risk	Good credit risk
Bad credit risk	49	49
Good credit risk	28	124

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6363636            0.7167630            0.5000000
##       Neg Pred Value            Precision               Recall
##            0.8157895            0.5000000            0.6363636
##                   F1           Prevalence       Detection Rate
##            0.5600000            0.3080000            0.1960000
## Detection Prevalence    Balanced Accuracy
##            0.3920000            0.6765633
```

Again, half of the "Bad Credit" predicted observations are indeed "Bad credit".

The Linear support vector models are not too bad in the sense that they are better that the models we have seen so far.

5. Radial Basis Support Vector Machine We try now with a radial basis kernel (the default).

```
##
## Call:
## svm(formula = RESPONSE ~ ., data = German_Credit.tr.subs, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  334
```

Table 10: Confusion Matrix of the Radial base support vector machine

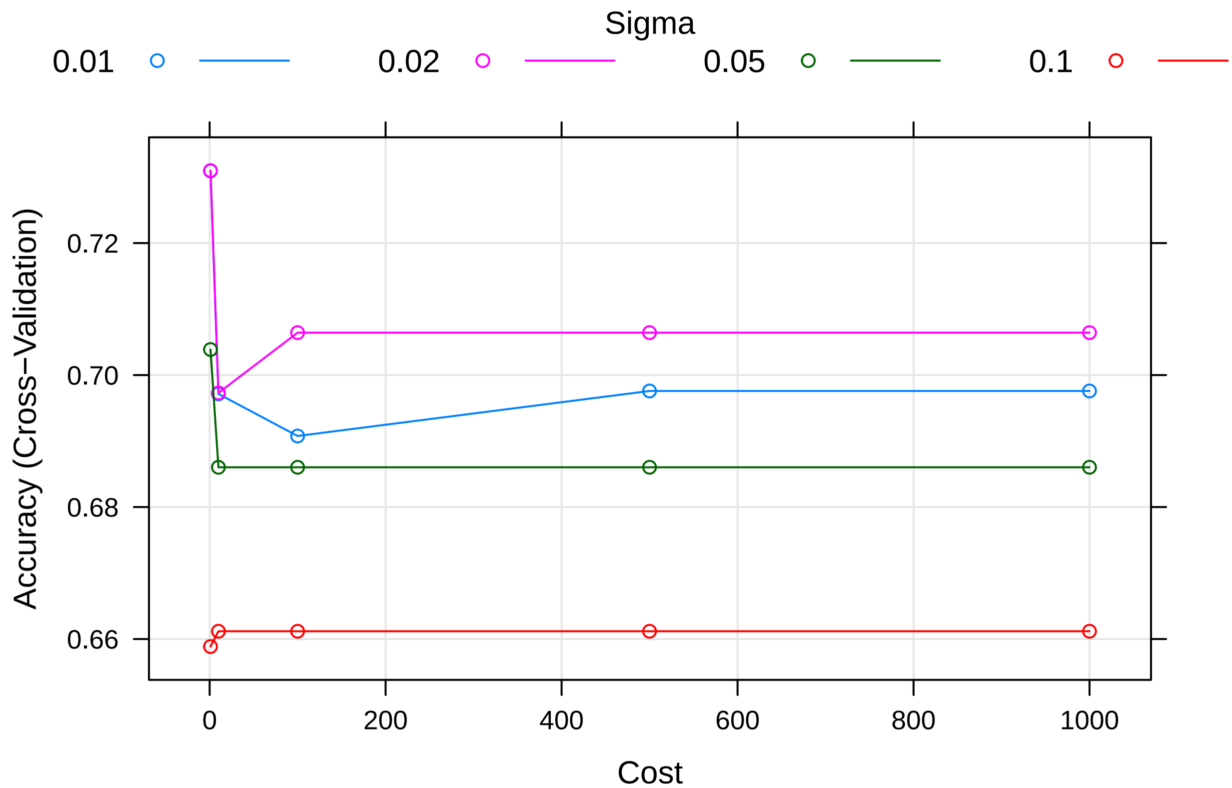
	Bad credit risk	Good credit risk
Bad credit risk	54	52
Good credit risk	23	121

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7012987            0.6994220            0.5094340
##       Neg Pred Value            Precision               Recall
##            0.8402778            0.5094340            0.7012987
##                   F1           Prevalence       Detection Rate
##            0.5901639            0.3080000            0.2160000
## Detection Prevalence    Balanced Accuracy
##            0.4240000            0.7003603
```

This model performance is very close to the linear support vector models. We need to infer more by looking at the tunned version.

Tunning the hyperparameters of Radial basis SVM

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 446 samples
##  30 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 402, 400, 402, 401, 402, 402, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   0.01      1  0.7309289  0.4618150
##   0.01     10  0.6971476  0.3942416
##   0.01    100  0.6907708  0.3814209
##   0.01    500  0.6975889  0.3950572
##   0.01   1000  0.6975889  0.3950572
##   0.02      1  0.7309816  0.4620420
##   0.02     10  0.6972925  0.3946754
##   0.02    100  0.7064273  0.4127482
##   0.02    500  0.7064273  0.4127482
##   0.02   1000  0.7064273  0.4127482
##   0.05      1  0.7038647  0.4085756
##   0.05     10  0.6860299  0.3726705
##   0.05    100  0.6860299  0.3726705
##   0.05    500  0.6860299  0.3726705
##   0.05   1000  0.6860299  0.3726705
##   0.10      1  0.6588603  0.3190546
##   0.10     10  0.6611792  0.3234506
##   0.10    100  0.6611792  0.3234506
##   0.10    500  0.6611792  0.3234506
##   0.10   1000  0.6611792  0.3234506
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.02 and C = 1.
```




```
##   sigma C
## 6  0.02 1
```

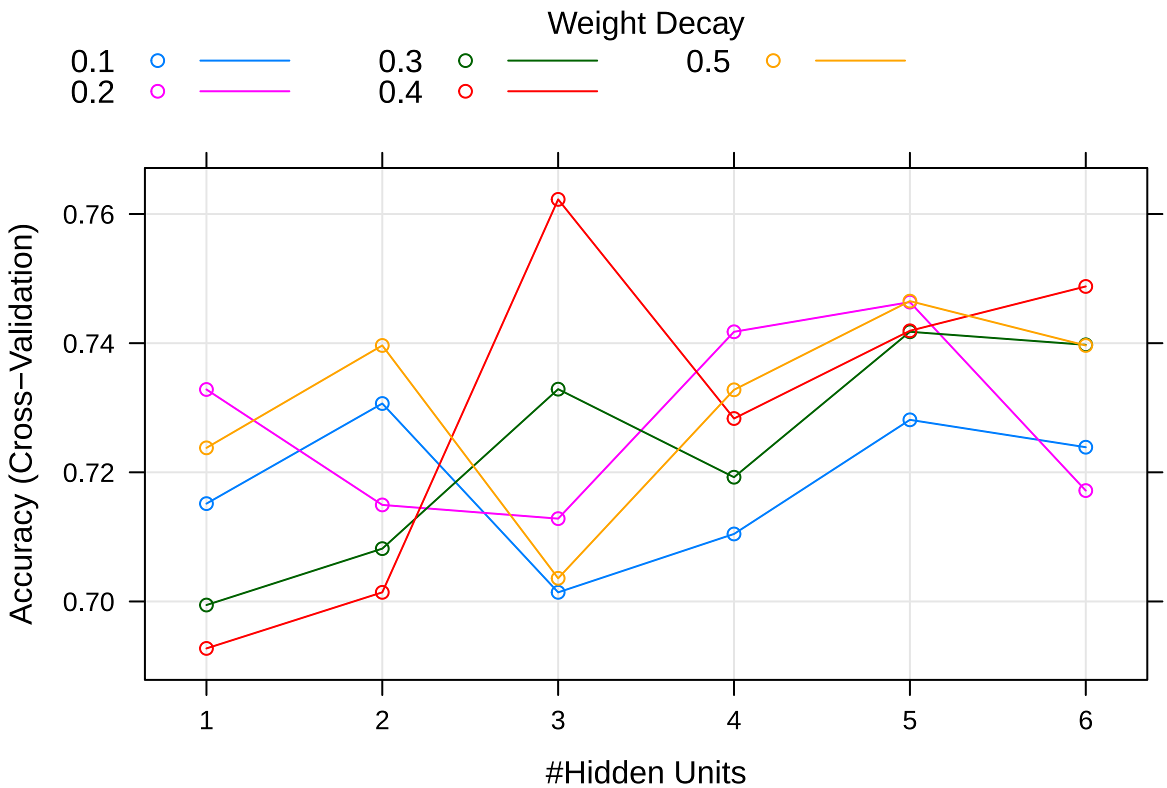
Table 11: Confusion Matrix of the Tuned radial base support vector machine

	Bad credit risk	Good credit risk
Bad credit risk	54	53
Good credit risk	23	120

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7012987            0.6936416            0.5046729
##       Neg Pred Value            Precision               Recall
##            0.8391608            0.5046729            0.7012987
##                   F1           Prevalence       Detection Rate
##            0.5869565            0.3080000            0.2160000
## Detection Prevalence    Balanced Accuracy
##            0.4280000            0.6974702
```

The model seems to be well balanced and it has good scores. The F-measure is quite high as well.

6. Neural Network - Simple hyperparameter tuning To select the good parameters, we build a search grid and fit the model with each possible value in the grid. This is brute force and time consuming. The best model is selected among all the possible choices.



The best Neural Networks parameters would be to choose 3 hidden layers, with a decay of 0.4.

Table 12: Confusion Matrix of the Hyperparameter tuned neural network 3 nodes

	Bad credit risk	Good credit risk
Bad credit risk	52	53
Good credit risk	25	120

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6753247            0.6936416            0.4952381
##       Neg Pred Value            Precision               Recall
##            0.8275862            0.4952381            0.6753247
##                   F1           Prevalence       Detection Rate
##            0.5714286            0.3080000            0.2080000
## Detection Prevalence    Balanced Accuracy
##            0.4200000            0.6844831
```

We see that this Neural Network competes quite close to the support vector machines ones. Almost half of the "Bad credit" cases that it has predicted are correct.

7. Gradient Boosting The Gradient Boosting model accepts only numerical values so we have some transformation to do on our data in order to use it.

```
## ##### xgb.Booster
## raw: 31.2 Mb
## call:
##   xgb.train(params = xgb_params, data = xgb_train, nrounds = 5000,
##     verbose = 1)
```

```
## params (as set within xgb.train):
##   booster = "gbtree", eta = "0.01", max_depth = "8", gamma = "4", subsample = "0.75", colsample_bytre
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
## # of features: 46
## niter: 5000
## nfeatures : 46
```

Here we have an accuracy of 68.4%. It is good but there is room for improvement.

Table 13: Confusion Matrix of the Gadient boosting

	Bad credit risk	Good credit risk
Bad credit risk	57	59
Good credit risk	20	114

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.7402597            0.6589595            0.4913793
##       Neg Pred Value            Precision               Recall
##            0.8507463            0.4913793            0.7402597
##                   F1           Prevalence       Detection Rate
##            0.5906736            0.3080000            0.2280000
## Detection Prevalence    Balanced Accuracy
##            0.4640000            0.6996096
```

Review of statistics

Once all the models were modelized we compared them according to their scores and metrics. Below we summarized all their accuracy into one table.

Table 14: Scores of the models (continued below)

	Big classification tree	Pruned classification tree	Autoprune classification tree	Logistic regression	AIC reduced Logistic regression	Linear support vector machine
Accuracy	0.644	0.564	0.584	0.704	0.684	0.692
Kappa	0.2945	0.2083	0.2251	0.3479	0.32	0.3328
Accuracy lower bound	0.5812	0.5001	0.5202	0.6432	0.6224	0.6307
Accuracy upper bound	0.7033	0.6264	0.6458	0.7599	0.7411	0.7486
Accuracy null	0.692	0.692	0.692	0.692	0.692	0.692
Accuracy P-value	0.9552	1	0.9999	0.369	0.6369	0.5308
Mcnemar P-value	1.158e-07	1.822e-14	3.352e-12	0.04813	0.00693	0.01217

	Big classification tree	Pruned classification tree	Autoprune classification tree	Logistic regression	AIC reduced Logistic regression	Linear support vector machine
Balanced Accuracy	0.6743	0.6345	0.6418	0.6852	0.6744	0.6802

	Tuned linear support vector machine	Radial base support vector machine	Tuned radial base support vector machine	Hyperparameter tuned neural network 3 nodes	Gradient Boosting
Accuracy	0.692	0.7	0.696	0.688	0.684
Kappa	0.3283	0.3628	0.3564	0.3352	0.35
Accuracy lower bound	0.6307	0.6391	0.6349	0.6266	0.6224
Accuracy upper bound	0.7486	0.7561	0.7524	0.7449	0.7411
Accuracy null	0.692	0.692	0.692	0.692	0.692
Accuracy P-value	0.5308	0.4219	0.4762	0.5847	0.6369
Mcnemar P-value	0.02265	0.001224	0.0008794	0.002235	1.909e-05
Balanced Accuracy	0.6766	0.7004	0.6975	0.6845	0.6996

According to these two first tables, the best model would be the "Radial base linear support vector machine" as it has the highest accuracy level of 0.7 and the best kappa value of 0.3628.

The accuracy means that out of total number of observations, the model predicted correctly 70% of them. The Cohen's Kappa Coefficient means that there is 36% of agreement, indicating that the raters agree in their classification for 36% of the cases.

We also remember that the model was almost better than the other models when it came to predict the "Bad credit" cases. Although the model remains quite poor at predicting them since only half of the predicted ones are correct.

Another table is done to compare the KNN because they were not performed on the balanced dataset.

Table 16: Scores of the KNN models

	2-Nearest neighbor	3-Nearest neighbor
Accuracy	0.596	0.636
Kappa	0.01313	0.02285
AccuracyLower	0.5323	0.573
AccuracyUpper	0.6574	0.6957
AccuracyNull	0.692	0.692
AccuracyPValue	0.9995	0.9752
McnemarPValue	0.3197	0.000365

Overall, we see that the worst model is the 'Autoprune classification tree'. This is understandable because we pruned the model so much that we lost many observations on the way.