MA5832-Assessment 2

Student: Sacha Schwab

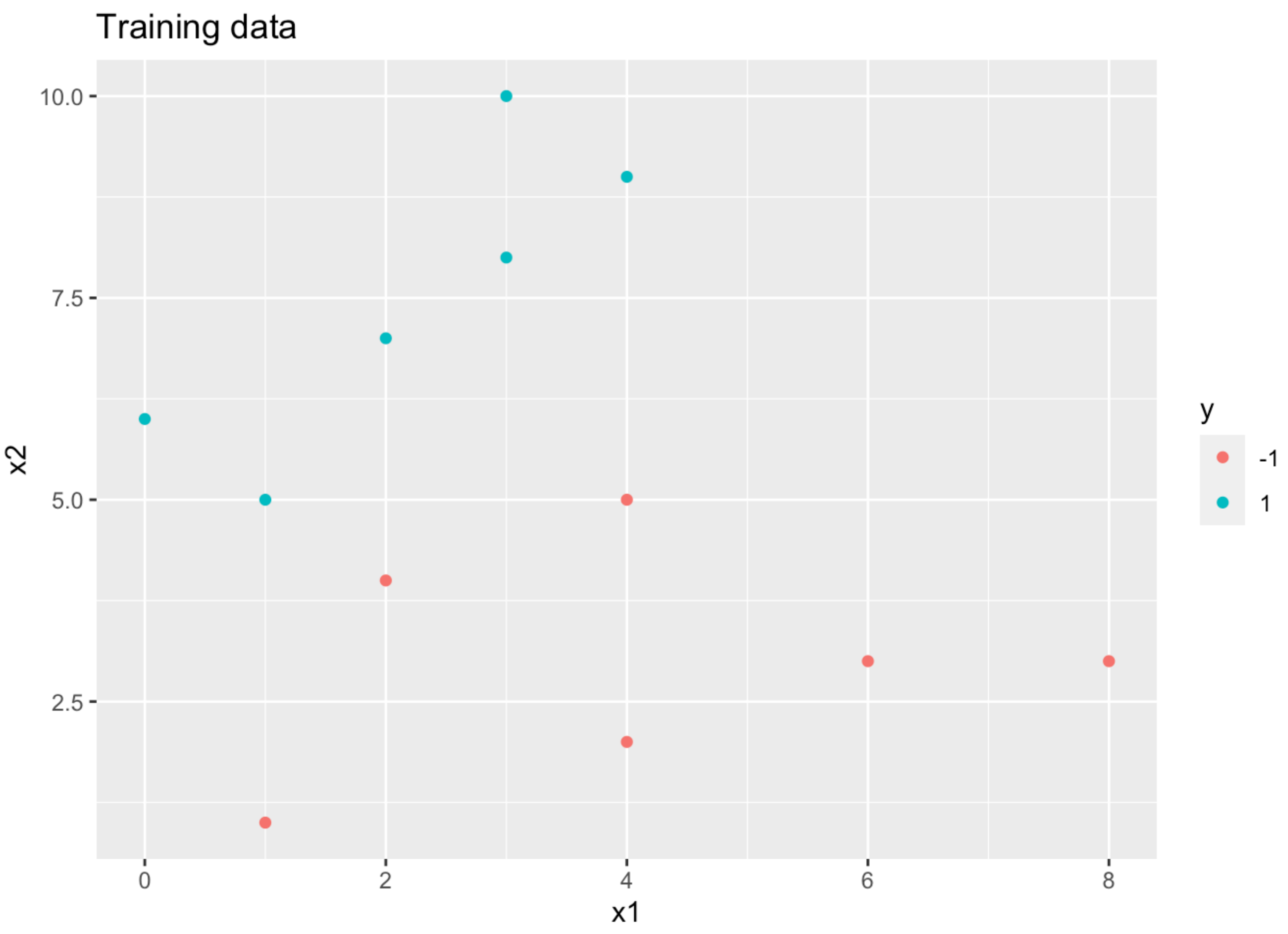
1 June 2020

Part I

Point 1

Question / Task: Students are asked to draw a scatter plot to represent the points of the table provided, with Red colour for the class Y = 1 and Blue colour for Y = –1. X1 is on the vertical axis while X2 is on the horizontal axis.

Execution:



Point 2

Question / Task: Students are asked to provide (a) the equation for the optimal separating hyperplane as well as (b) the values of β by using the function solve.QP() from quadprog package, and (c) to sketch the optimal separating hyperplane in the scatter plot.

a) Thoughts & Execution: There are different connotations of the term “optimal separating hyperplane”. Two versions may apply: i) The optimal separating hyperplane is the surface that maximizes the margin between the support vectors (see e.g. Posik 2015), whereas, in a 2-dimensional space, the middle line is $\beta_0 + \beta_1 x = 0$, the vector above is described by $\beta_0 + \beta_1 x = 1$, and the vector below, $\beta_0 + \beta_1 x = -1$, or,

$$\beta_0 + \beta_1 x(+) = 1$$
$$\beta_0 + \beta_1 x(-) = -1$$

Thus,

$$x^- + \lambda = x^+$$

However, if the margin M between the separating lines is $2/||w||$ we can provide the surface by multiplying the margin with the divider of the hyperplanes, which gives

$$\beta_0 m + \beta_1 x m = 0$$

ii) If however the background of the question is that the optimal separating hyperplane is equal to the largest separation or margin between two classes (see e.g. Liu et al. 2020), then, in a two-dimensional problem, the optimal separating hyperplane is described by $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$

b) Thoughts: Quadprog solves

$$\min_x d^T x + 1/2 x^T D x$$
$$\text{such that } A^T x \geq x_0$$

In other words: From the explanatory paper shared in the assessment 2 forum by student Adrian Langley, the quadprog package in R solves the quadratic programming problem to minimise the sum of the squared vertical distances between observed points (the support vectors in our case) and the dividing line itself. Or: solve.QP() from quadprog package aims at minimizing 1/2 * ||w||^2 subject to y(wx + b) >= 1, since minimizing 2 / ||w|| (where ||w|| is the magnitude of the support vector w) is minimizing 1/2 * ||w||^2 [see Alpha Coder 2017].

Execution: Input to solve.QP (see Cichosz 2015, p. 467): - Dmat represents the matrix specifying the coefficients of the quadratic term of the optimization $X^T X$.

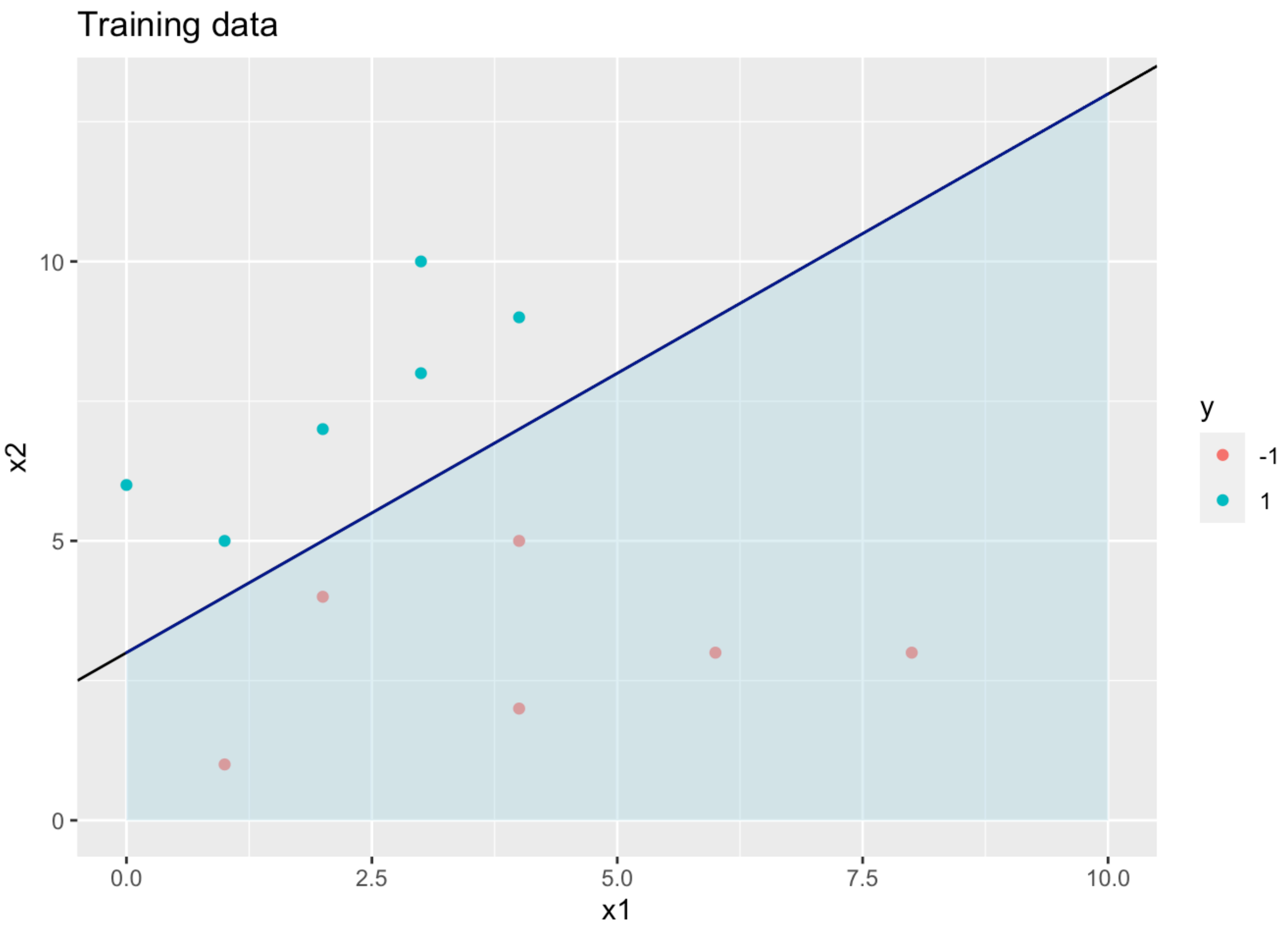
- dvec a vector containing the coefficients of the decision variables, here a T-element vector of 1’s
- Amat is obtained prepending a vector of $c\_(x)$ for each $x \in T$ as the first column to the $|T|$ x $|T|$ identity matrix
- bvec: a ($|T|$ + 1)-element vector of 0’s

Dmat: Gramian matrix expressed with coefficients of y (as its outer product)

We find the two β-values in the resulting vector that are close to 1, i.e. 2 and 7. The slope is calculated as $(y2 - y1)/(x2 - x1)$, i.e., multiplied by the coefficient, $(1 - 2)/(5 - 4) = 1$. The intercept will be the the mean of $b1$ and $b2$ in $mx + b = y$, i.e., for the two points identified, X[2] = [2,1] and X[7] = [1,5], the mean of the two b’s, 2 and for, equals 3.

```
slope <- 1
intercept <- 3
```

c) Sketch the optimal separating hyperplane in the scatter plot obtained in Question 1



Point 3

Question / Task:

Students are asked to describe the classification rule for the maximal margin classifier.

My solution:

The classification rule for the maximal margin classifier is “green if x1−x2+3<0, and red otherwise.”

Point 4

Question / Task: Students are asked to compute the margin for the maximal margin hyperplane.

Execution: The margin is $2/||w||$ where $||w|| = \sqrt{\sum w^2}$, i.e. $2/\sqrt{2}$

```
2/sqrt(2)
```

```
## [1] 1.414214
```

Point 5

Question / Task: Students are asked to indicate the support vectors for the maximal margin classifier.

Solution: As stated above, the support vectors are [2,4] and [1,5].

Part II: An Application

2.2.1 Data

Question / Task:

Select a random sample of 70% of the full dataset as the training data, retain the rest as test data. Provide the code and print out the dimensions of the training data.

Prior execution we need to preprocess the data in order to start with a clean version.

Data preprocessing:

The response variable ('Y') provides whether the client relationship resulted in a default (1) or not (0), i.e. whether the client is credible or not. The distribution is:

```
table(data$Y)
```

```
##
##                         0                          1
##                     23364                       6636
## default payment next month
##                         1
```

This indicates the necessity to remove the "default payment next month" value. Frequency after this clean-up:

```
##
##      0      1
## 0.7788 0.2212
```

Further, the numeric variables should contain numeric values, which is not the case for X5 and X12 through X23. I therefore convert from text to numeric.

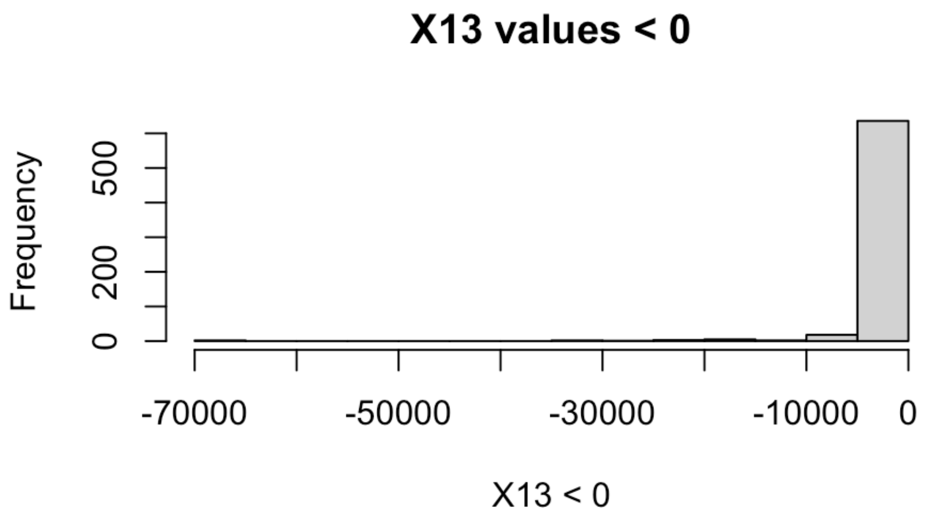
After this step, a glimpse to the variables displays:

```
print(str(data.clean))
```

```
## 'data.frame':    30000 obs. of  25 variables:
##  $ X  : chr  "1" "2" "3" "4" ...
##  $ X1 : num  20000 120000 90000 50000 50000 50000 500000 100000 140000 20000 ...
##  $ X2 : Factor w/ 2 levels "1","2": 2 2 2 2 1 1 1 2 2 1 ...
##  $ X3 : Factor w/ 7 levels "0","1","2","3",..: 3 3 3 3 3 2 2 3 4 4 ...
##  $ X4 : Factor w/ 4 levels "0","1","2","3": 2 3 3 2 2 3 3 3 2 3 ...
##  $ X5 : num  24 26 34 37 57 37 29 23 28 35 ...
##  $ X6 : Factor w/ 11 levels "-1","-2","0",..: 5 1 3 3 1 3 3 3 3 2 ...
##  $ X7 : Factor w/ 11 levels "-1","-2","0",..: 5 5 3 3 3 3 3 1 3 2 ...
##  $ X8 : Factor w/ 11 levels "-1","-2","0",..: 1 3 3 3 1 3 3 1 5 2 ...
##  $ X9 : Factor w/ 11 levels "-1","-2","0",..: 1 3 3 3 3 3 3 3 3 2 ...
##  $ X10: Factor w/ 10 levels "-1","-2","0",..: 2 3 3 3 3 3 3 3 3 1 ...
##  $ X11: Factor w/ 10 levels "-1","-2","0",..: 2 4 3 3 3 3 3 1 3 1 ...
##  $ X12: num  3913 2682 29239 46990 8617 ...
##  $ X13: num  3102 1725 14027 48233 5670 ...
##  $ X14: num  689 2682 13559 49291 35835 ...
##  $ X15: num  0 3272 14331 28314 20940 ...
##  $ X16: num  0 3455 14948 28959 19146 ...
##  $ X17: num  0 3261 15549 29547 19131 ...
##  $ X18: num  0 0 1518 2000 2000 ...
##  $ X19: num  689 1000 1500 2019 36681 ...
##  $ X20: num  0 1000 1000 1200 10000 657 38000 0 432 0 ...
##  $ X21: num  0 1000 1000 1100 9000 ...
##  $ X22: num  0 0 1000 1069 689 ...
##  $ X23: num  0 2000 5000 1000 679 ...
##  $ Y  : int  1 1 0 0 0 0 0 0 0 0 ...
## NULL
```

Another observation is that there are negative values in the bill statement amounts in variables X12-X17,

see e.g. for X13:



Prima vista this does not make sense. However, a logical reason is that the clients paid too much in previous months, and the negative billing amounts represents refunds due to the resulting positive balance. Question is whether this conclusion shall lead to dropping the respective rows as they might have no relevant information to the classification (as these instances would probably not lead to a default). However, the latter thought indicates that the values actually provide information to the category.

Further, there appear to be no NA values or other values to be removed (or replaced) for execution of algorithms that are sensitive to missing data.

Execution of the task to separate training and test dataset and to display the dimensions for the training data:

```
data.clean$Y <- as.factor(data.clean$Y)
# set the seed to make your partition reproducible
set.seed(123)
train.ind <- sample(seq_len(nrow(data.clean)), size = floor(0.70 * nrow(data.clean)
))
train <- data.clean[train.ind, ]
test <- data.clean[-train.ind, ]
dim(train)
```

```
## [1] 21000    25
```

2.2.2 Tree based algorithms

Question / Task: The task is to classify credible and non-credible clients using a tree-based method, and to justify the model and hyperparameter selection.

Execution:

- Basic thoughts:

  Thinking in practical ‘dimensions’ when reflecting the present dataset: The financial institute providing the credits intends to detect default clients beforehand. This may be achieved through in-depth customer data (e.g., besides age and education: profession, income, past credit history etc.). Such data is however not present in the dataset.

  Also: The data rather represents a time series approach of payment behaviour over a very limited amount of time (6 months). The aim for a practical implementation of the model can therefore be to apply it based on monthly reports looking back for a 6 months time period. All of the data groups (history of past payments, amount of billing statements, amount of previous payment) may provide relevant informtion to the model. I therefore decide to include all the data and to not exclude any variables.

- Model selection: The model selection may be driven by practical and legal/regulatory considerations, depending on what the actual action taken by the institute is:

  If e.g. the action is to raise the payment amounts of those clients that are predicted as default, then probably the underlying contract has the payment amounts installed, with a legal clause that the institue may adapt the amount. Legally, this may only be undertaken if the institute can provide proof for the decision base. This is only possible when the model is interpretable. However, due to the nice feature of decision trees to have the separation rules visualisable, the institute will be able to explain the decision basis.

Another factor is IT resources required, i.e. required computation time: There, I would assume that, if the model is successful, the institue may want to invest considerable amounts of money, i.e. omitting the required computation power or only considering if it is 'enormous'.

Overfitting: In order to assure interpretability also towards the regulator, the institue may want to avoid overfitting.

A very important, if not most important factor, is the accuracy of the classifier, as the institute's reputation also depends on 'chasing' those clients that are (or will become) actual 'bad' i.e. default clients. Depending on the local regulations, practices to cut e.g. credit limits based on mere assumptions might even be illegal.

The following provides a comparison of pros and cons of a number of decision tree models:

  - Random forest: According to Libermann 2017 and other authors, random forests have the ability to limit overfitting without significantly increasing error due to bias. The result is (as usual for decision trees) interpretable, however the computation to achieve the result has only very limited interpretability.
- Parameter selection: For the sake of interpretability and to avoid overfitting, pruning appears to be an important aspect.

In the end, it comes down to accuracy being the top criteria. Computational efforts and complexity do not appear to play a major role. I will therefore explore more sophisticated models and choose, for a start, random forests and boosted trees.

Random Forest: - Parameter selection: According to Bhalla 2015, mtry is the only adjustable parameter to which random forests is somewhat sensitive, as reducing mtry reduces both the correlation and the strength. An 'optimal' range is stated to be "somewhere in between".

Implementation in R: Random Forest

The following code produces 24 random forest models, with varying bagging "mtry" parameter for experimentation whether the indication in the literature has an effect on the performance of the model.

```
library(randomForest)
library(reprtree)
set.seed(1234)
# Standard mtry = sqrt(number of variables)
rand.tree.0 <- randomForest(x = train[2:24],
                            y = train$Y, xtest = train[2:24], ytest = train$Y,
                            keep.forest=TRUE)
# Try different mtry settings
for(i in 2:23) {
  assign(paste0("rand.tree.", i), randomForest(x = train[2:24],
                                               y = train$Y,
                                               mtry=i,
                                               xtest = train[2:24],
                                               ytest = train$Y))
}
```

b) Summary and discussions

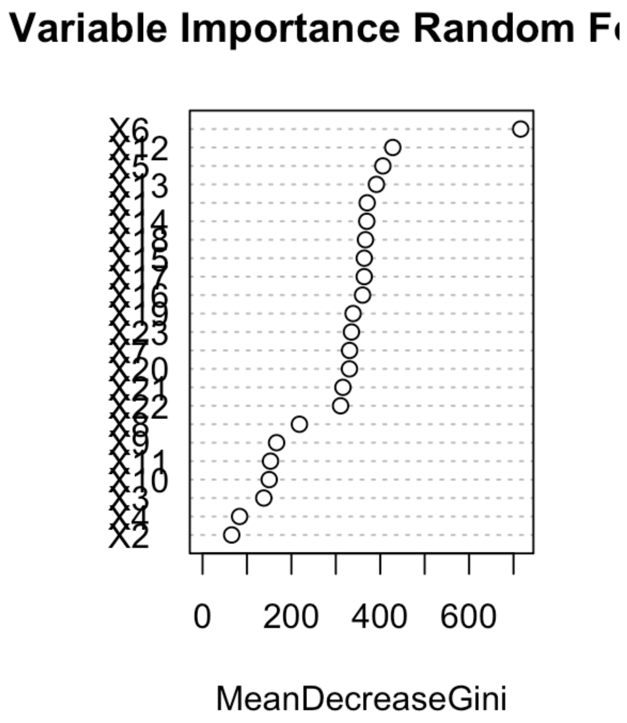
Confusion matrix:

```
rand.tree.0$confusion
```

```
##       0    1 class.error
## 0 15410  911  0.05581766
## 1  2923 1756  0.62470613
```

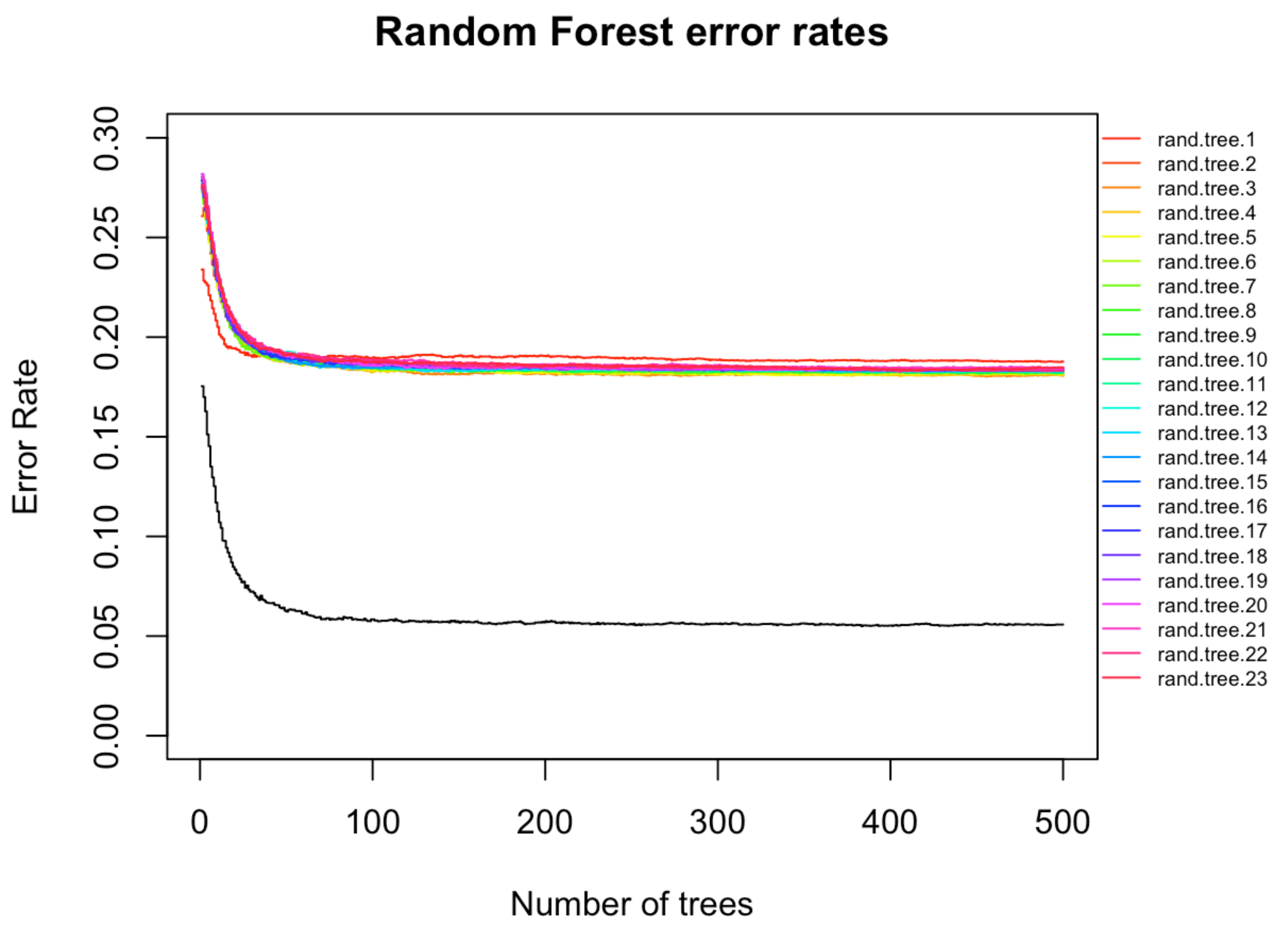
Variable Importance Plot:

```
varImpPlot(rand.tree.0, main = "Variable Importance Random Forest")
```



Discussion: The confusion matrix provides acceptable results, i.e. the feature selection provides appropriate response compared to the labels.

b) Error rates:



The black line represents the error rates of the tree using the basic mtry parameter. Hence, this one is selected for further evaluation.

Discussion: The error rates appear to be within acceptable ranges;

Implementation in R: Gradient Boost Decision Tree

```
library(caret)
test$Y <- as.factor(test$Y)
tc = trainControl(method = "repeatedcv", number = 5)
boost.tree.model.1 = train(Y ~., data=train[2:25], method="gbm", trControl=tc)
```

b) Display the model summary:

```
boost.tree.model.1
```

```
## Stochastic Gradient Boosting
##
## 21000 samples
##    23 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 16800, 16800, 16800, 16800, 16800
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8109048  0.2912195
##   1                  100      0.8140952  0.3274409
##   1                  150      0.8154762  0.3405388
##   2                   50      0.8169048  0.3559426
##   2                  100      0.8188095  0.3716381
##   2                  150      0.8187619  0.3736921
##   3                   50      0.8187143  0.3685639
##   3                  100      0.8195714  0.3767124
##   3                  150      0.8196190  0.3807912
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

Summary and feature importance plot:

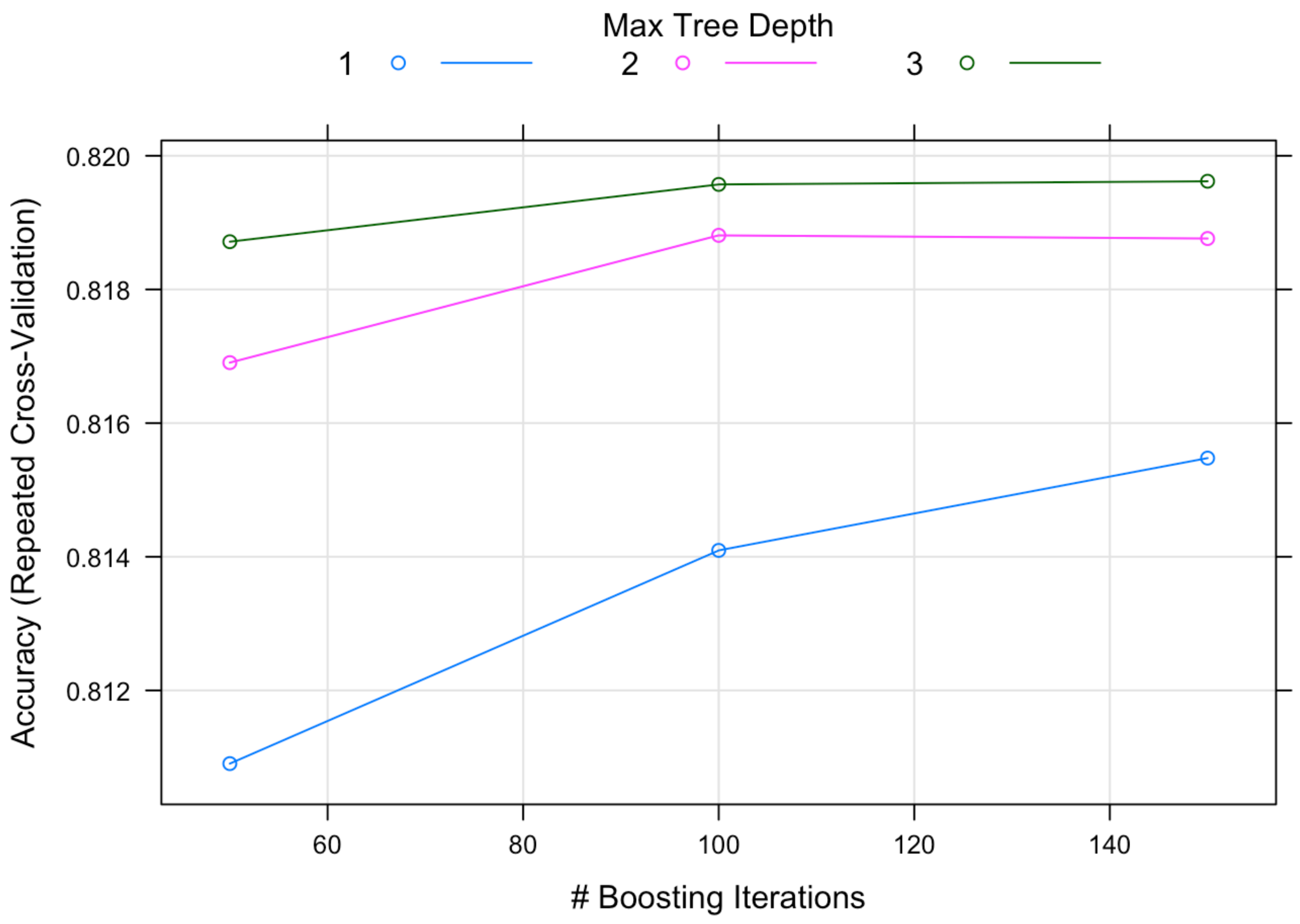
```
show.gbm(boost.tree.model.1)

# Not available in RMarkdown :-(   Showing:
# gbm(formula = Y ~ ., distribution = "gaussian", data = train[2:25],
#     n.trees = 3000, interaction.depth = 4, shrinkage = 0.01)
# A gradient boosted model with gaussian loss function.
# 3000 iterations were performed.
# There were 23 predictors of which 23 had non-zero influence.
```

Discussion

Interestingly (probably I am wrong) there is not much information to be gained from the GBM model object. The mean train error appears to be in an acceptable range. There is a feature importance plot appearing as summary, however in RMarkdown it does unfortunately not. It shows that X6 has a predominant importance over all other features, which confirms other observations in this assessment that the payment history variables bear an important amount of information to the response variable.

c) Error rates display (expressed in accuracy):

```
plot(boost.tree.model.1)
```



Discussion:

The plot shows that with increase in depth, the accuracy increases (within moderate terms), i.e. error rates decrease. This is probably due to high dimensionality of the present dataset, i.e. deeper trees produce finer scales of nodes and leaves that in sum provide more information to the response variable, i.e. cover higher variance.

2.2.3 Support vector classifier

Disclaimer’: Due to high computational time already for tuning (> 3 hours per kerne) I could include cross-validation only for one of the kernels. However, for a first learning experience tuning the cost should be appropriate. the following function from “caret” library is run with different kernels. It uses 9 cost tuning steps and the standard “center”/“scale” preprocessing. For kernel selection, according to Raschka the rule of thumb is to use nonlinear kernels for non-linear problems, and that in practice polynomial kernel is not less useful for efficiency and prediction. This leaves the kernel for the present problem being RBF. Also, Raschka provides that based on the loss function or a performance metric the kernel selection can be made.

Even if this is a non-linear problem in higher dimensions, I all the same ran linear kernel for the experience.

For the C parameter (cost) I select a range between 0.01 and 1, which appears appropriate. Due to lack of computational resources I limit the tuning range to 5 steps.

Performance shall be measured along the error rates and ROC curve, therefore the metric "ROC" is selected. Sigma is held to a minimum number of values, i.e. 1, being 0.01.

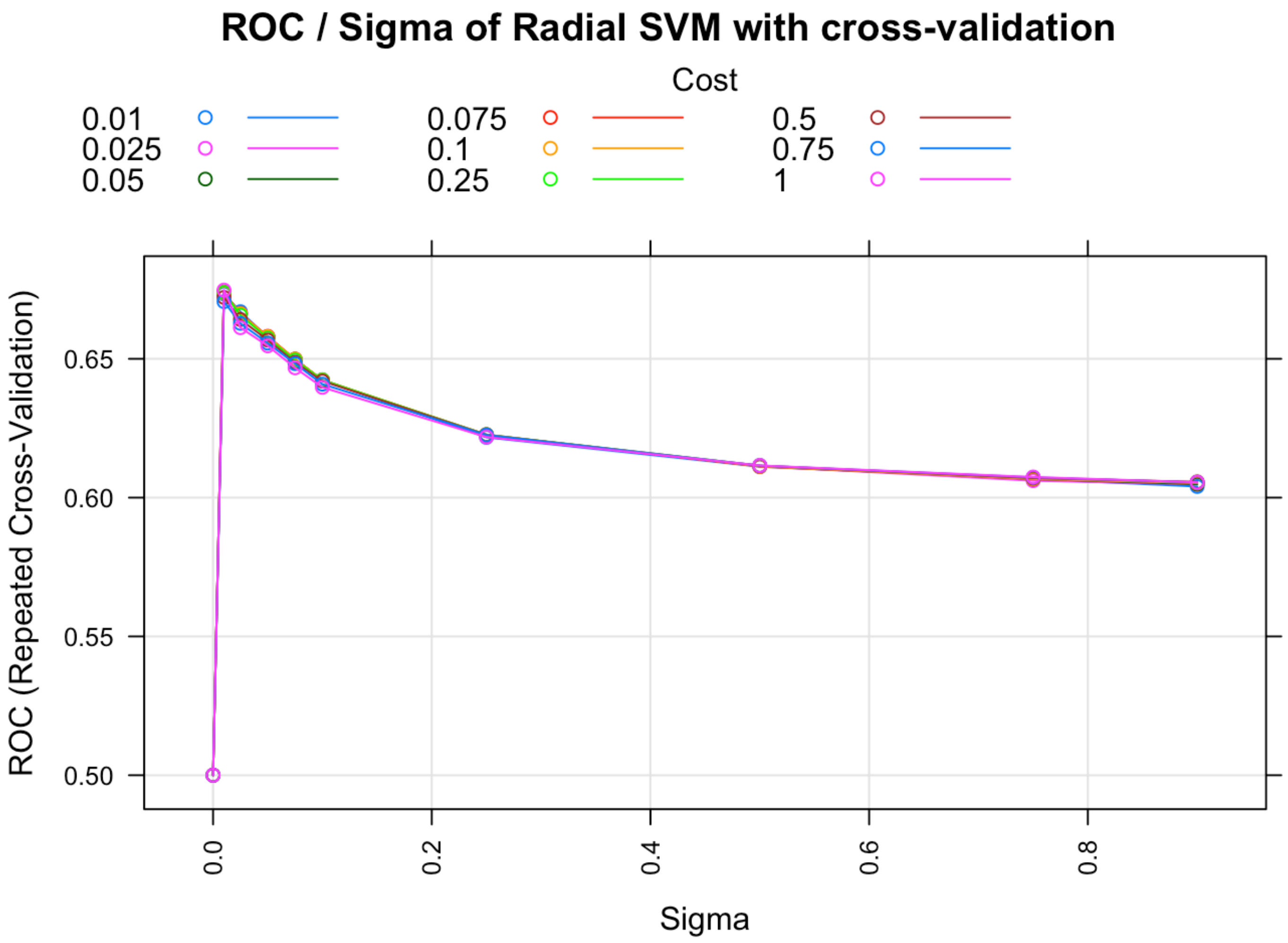
```
library(caret)
# Prepare SVM functions
levels(train$Y) <- make.names(levels(factor(train$Y)))
svm.train <- function(method, repeats, number) {
  set.seed(1492)
  trctrl <- trainControl(method="repeatedcv", repeats=repeats,
                         number = number, classProbs = TRUE, index=folds,
                         summaryFunction = twoClassSummary)

  grid_radial <- expand.grid(
    sigma = c(0,0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75,0.9),
    C = c(0.01,0.025,0.05,0.075,0.1,0.25, 0.5, 0.75, 1))

  svm.radial.noCV.tuneGrid <- train(Y ~., data = train.for.svm[2:25],
                                    method = "svmRadial",
                                    trControl=trctrl,
                                    preProcess = c("center", "scale"),
                                    tuneGrid = grid_radial,
                                    metric="ROC")
}
svm.Linear.noCV.tuneGrid <- svm.train("svmLinear", repeats = 2, number = 2) # This
would not take ROC
svm.radial.noCV.tuneGrid <- svm.train("svmRadial", repeats = 2, number = 2) # This
would not take ROC
svm.radial.CV.tuneGrid <- svm.train("svmRadial", repeats = 10, number = 5) # This w
ould take ROC
```

b. What is the relationship between the response variable and the selected features? This question pertains to the identification of the most relevant predictors of the response variable. Taking the summary plot of the the resulting models (see below) we receive a similar result to random forest, however without the dominance of payment history of the closest month (September). Also, the amount of the credit is deemed to have a greater influence in SVM than in decision trees. Further, the intuition that the amount of the previous payment bears information as to the "success" of the relationship (i.e. non-default).

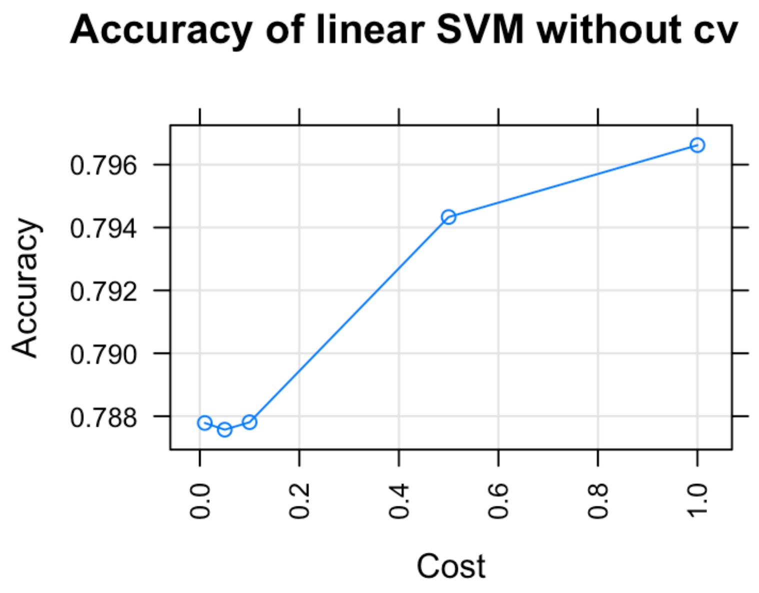
b) Display the model summary:

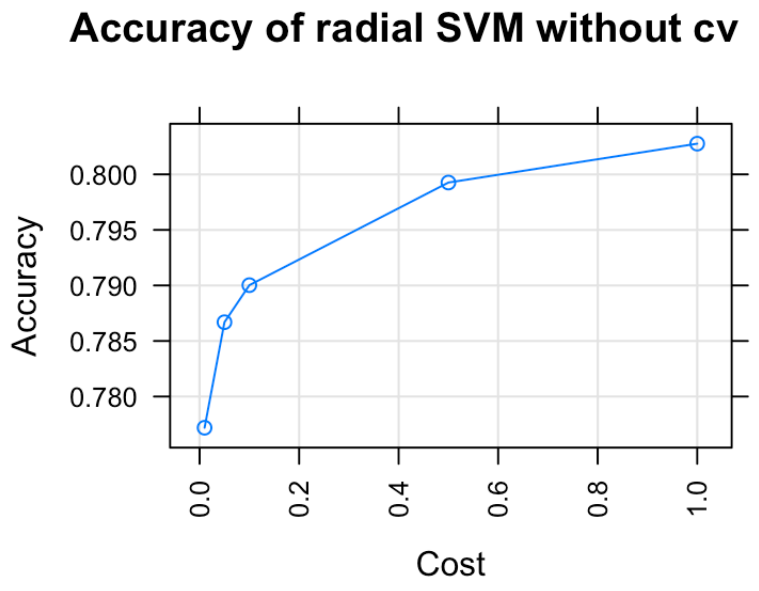
ROC / Sigma plot:

```
plot(svm.radial.CV.tuneGrid, scales = list(x = list(rot = 90)), main="ROC / Sigma o
f Radial SVM with cross-validation")
```



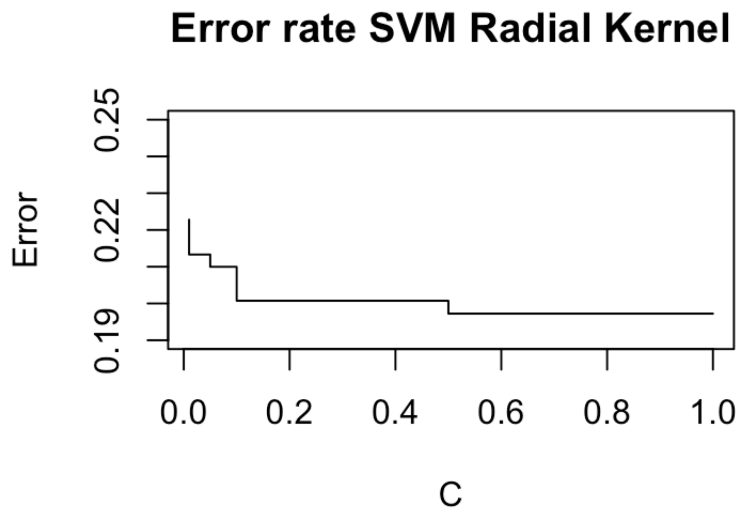
Accuracy plots:

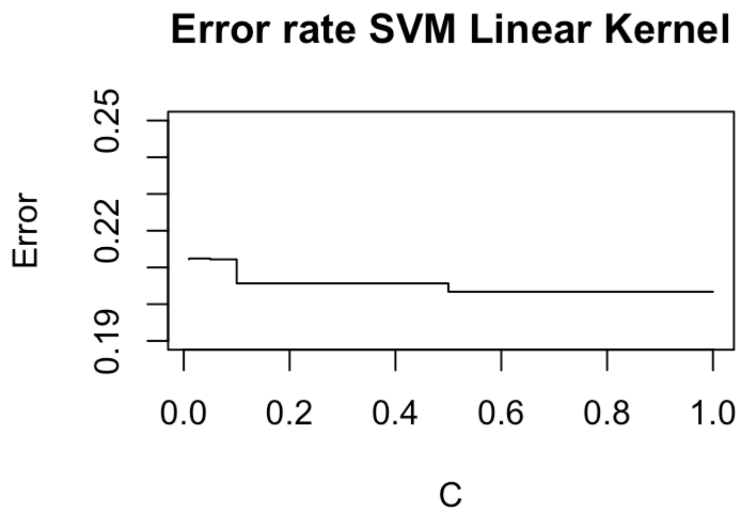




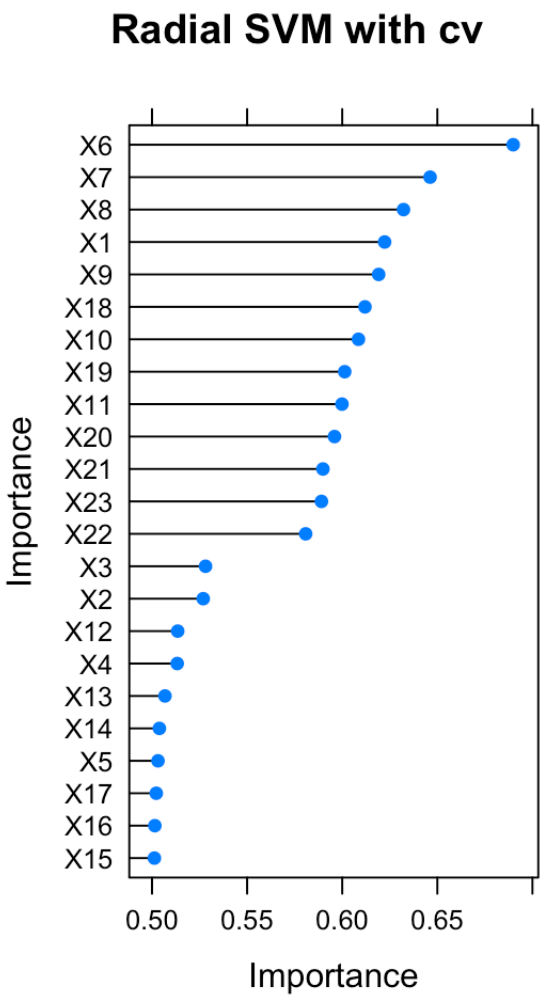
Error rate plots:

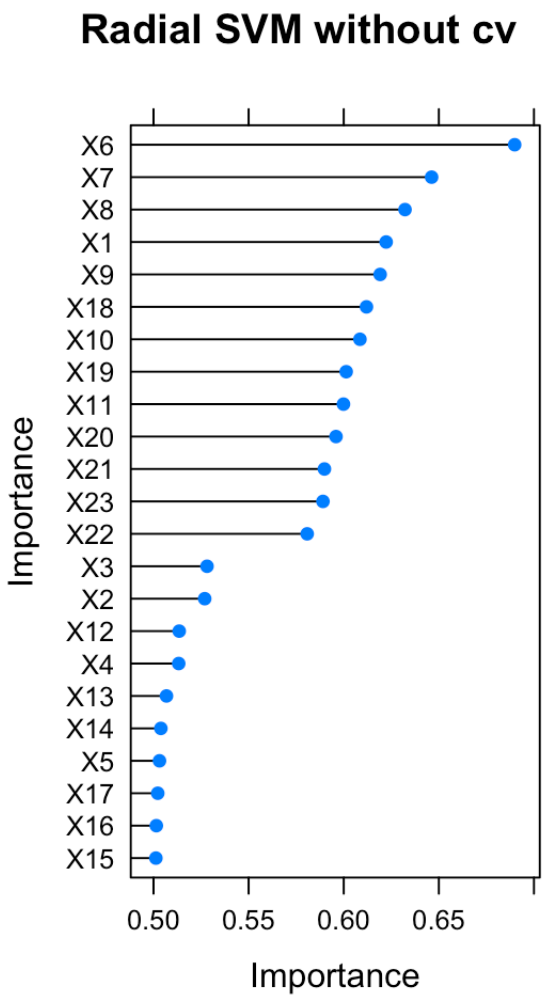
(C tunes = 0.01,0.05,0.1,0.5, 1, both kernels without cv)

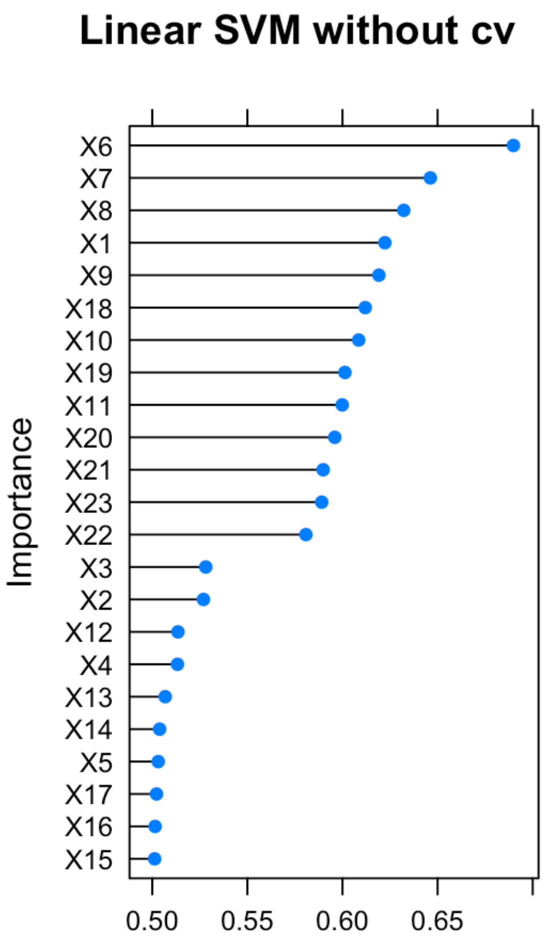




#### Variable importance plots:


Radial SVM with cv
X6
X7
X8
X1
X9
X18
X10
X19
X11
X20
X21
X23
X22
X3
X2
X12
X4
X13
X14
X5
X17
X16
X15
Importance
0.50
0.55
0.60
0.65
Importance


Radial SVM without cv
X6
X7
X8
X1
X9
X18
X10
X19
X11
X20
X21
X23
X22
X3
X2
X12
X4
X13
X14
X5
X17
X16
X15
Importance
0.50
0.55
0.60
0.65
Importance


Linear SVM without cv
X6
X7
X8
X1
X9
X18
X10
X19
X11
X20
X21
X23
X22
X3
X2
X12
X4
X13
X14
X5
X17
X16
X15
Importance
0.50
0.55
0.60
0.65

Importance

Comment:

ROC plots indicate that the SVM method may not deliver appropriate results. This may be due to the fact that the response variable is distributed among the data points (of the selected features) such that there are no clear boundaries to be identified. Also, the variable importance provides some non-intuitive information, e.g. the credit amount within the top 5, and the billing amount at position 6, without having provided any correlation to the credit amount or the amounts paid in previous months. This also indicates that (this version of) SVM may have its difficulties with the given feature complexity.

c) Commenting the error rates:

The error rates along the cost parameter are however within an acceptable range considering the rather basic set-up of tuning and (due to lack of time to compute) inferior cross-validation set-up.

In terms of error rates, the radial kernel SVM shows a higher sensibility to cost.

Of course, with a more sophisticated set-up and provided more resources, the performance could be improved.

2.2.4: Prediction

Applying the most appopriate model options to the test data, achieving:

Accuracy of prediction from random forest model:

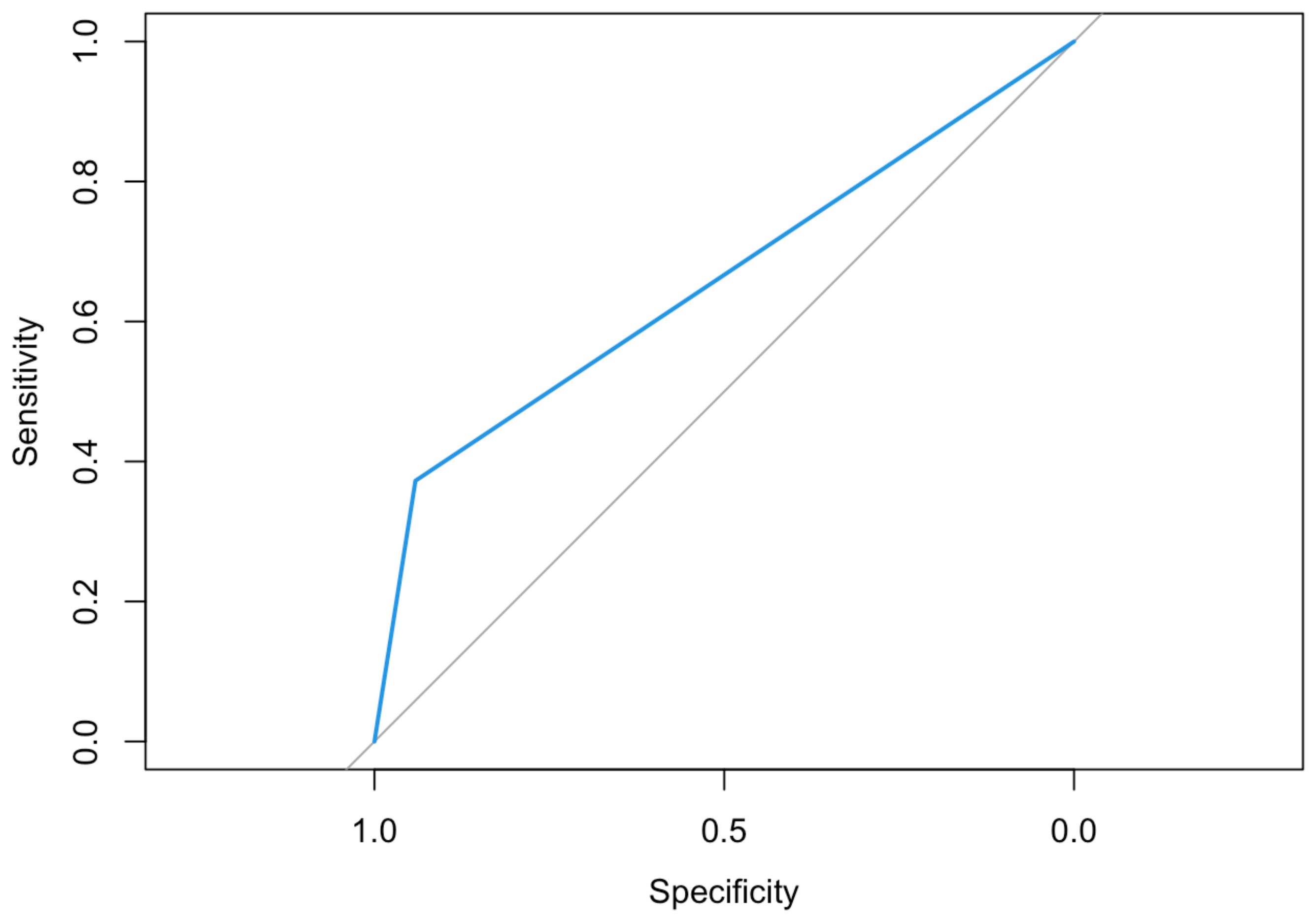
```
##
##     FALSE      TRUE
## 0.1823333 0.8176667
```

Performance metrix for random forest:

```
##           Reference
## Prediction          0          1
##          0 0.73666667 0.04588889
##          1 0.13644444 0.08100000
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.176667e-01   3.694178e-01   8.095316e-01   8.255956e-01   8.731111e-01
## AccuracyPValue  McnemarPValue
##   1.000000e+00   8.298730e-90
##          Sensitivity          Specificity       Pos Pred Value
##            0.8437261            0.6383538            0.9413602
##       Neg Pred Value            Precision               Recall
##            0.3725089            0.9413602            0.8437261
##                   F1           Prevalence       Detection Rate
##            0.8898732            0.8731111            0.7366667
## Detection Prevalence    Balanced Accuracy
##            0.7825556            0.7410400
```

ROC Curve for random forest:



Accuracy of prediction from GBM model:

```
prop.table(table(test$Y == pred.gbm))
```

```
##
##     FALSE      TRUE
## 0.1795556 0.8204444
```

Performance metrics for GBM tree:

```
##           Reference
## Prediction          0          1
##          0 0.73966667 0.04288889
##          1 0.13666667 0.08077778
```

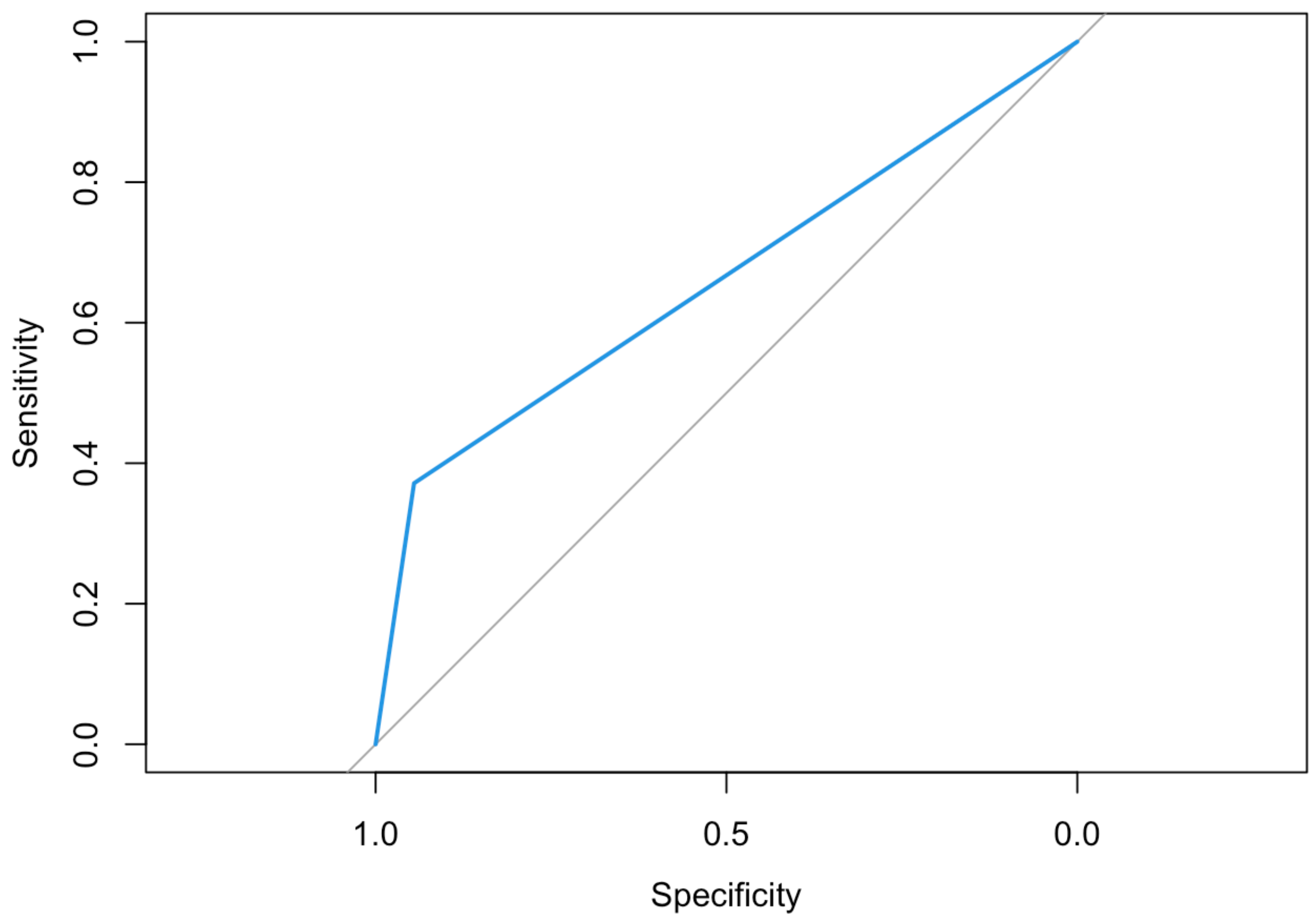
```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.204444e-01   3.750891e-01   8.123560e-01   8.283249e-01   8.763333e-01
## AccuracyPValue  McnemarPValue
##   1.000000e+00   1.222095e-97
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8440472            0.6531896            0.9451938
##       Neg Pred Value            Precision               Recall
##            0.3714870            0.9451938            0.8440472
##                   F1           Prevalence       Detection Rate
##            0.8917616            0.8763333            0.7396667
## Detection Prevalence    Balanced Accuracy
##            0.7825556            0.7486184
```

ROC Curve for GBM:

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```



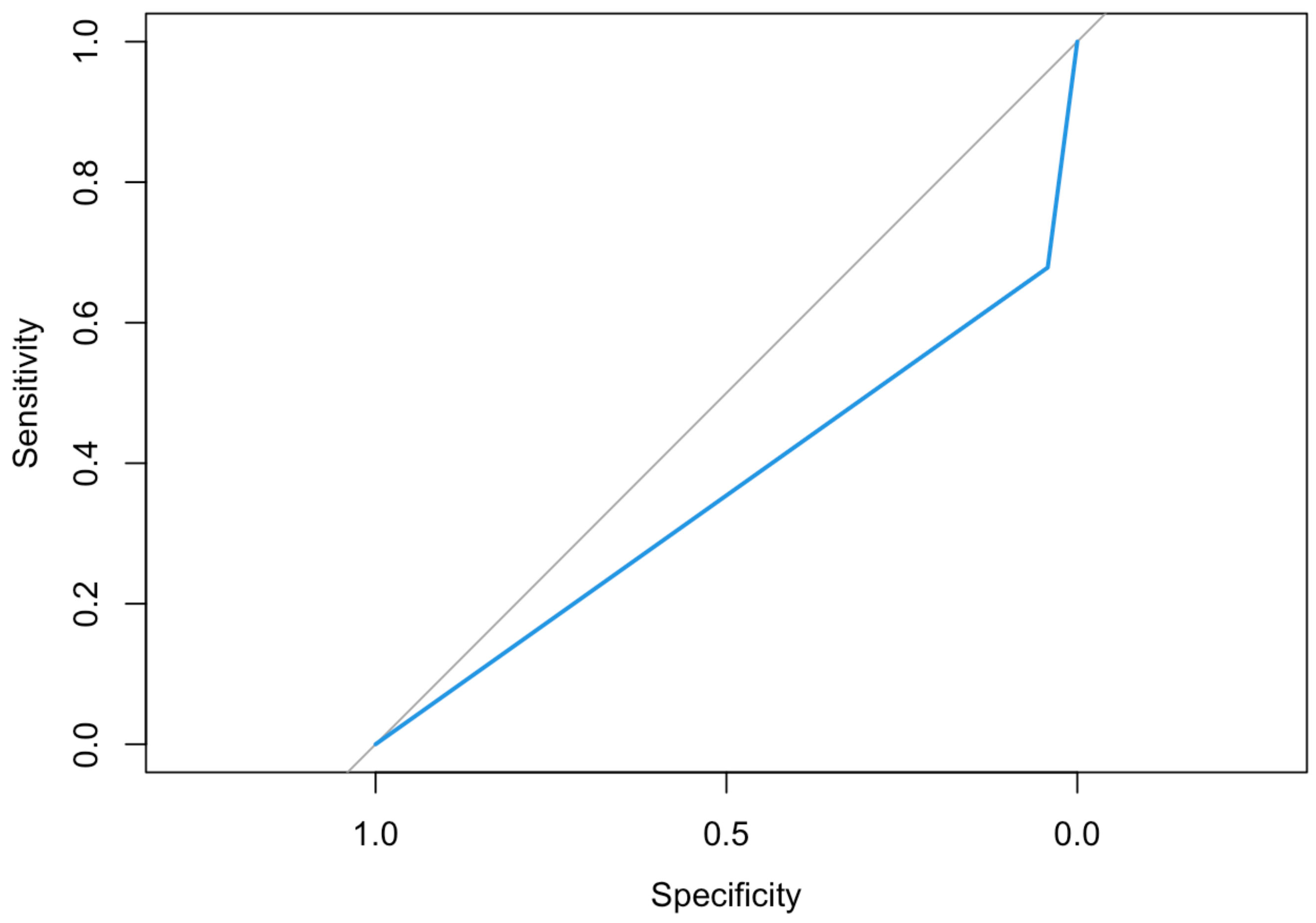
Accuracy of prediction from SVM model, and confusion matrix:

```
##
##     FALSE      TRUE
## 0.8194444 0.1805556
```

ROC Curve for SVM:

```
## Setting levels: control = 1, case = 2
```

```
## Setting direction: controls < cases
```



Discussion:

As already seen above, SVM does did perform appropriately in training, which takes its effect in prediction, where, in particular, the ROC curve is inferior of appropriate. Also, accuracy is not within expectations. I may therefore need to re-select features for SVM so to achieve a data distribution that allows for boundaries to the support vectors through any of the kernels.

In particular, as seen with decision trees, the payment history variables appear to bear most of the information to the classification. However, from a regulatory perspective (see basic considerations above), SVM is not well interpretable and would therefore not be pursued by the financial institute.

As to the decision tree algorithms, to my surprise, both algorithms (random forest and gradient boosting) provide very similar results.

The sophistication level regarding hyperparameters (due to lack of time) may be similar to both methods, however such a “spontaneous” result appears significant. The ROC curves are very similar but need to be improved as the area under the curve appears inferior, in particular compared to indications in the literature (see e.g. Pires de Oliveira 2017). The performance metrics appear close to acceptable, although sensitivity and specificity, influenced by a rather large false positive rate, could be better. I am certain that with further analysis in particular in conjunction with domain knowledge the performance can be improved. In particular I suppose that some level of feature engineering (e.g. taking the changes in payment behaviour from month to month) could help.

Overall I am however quite happy that the methods to be favoured from a regulatory standpoint (i.e. decision trees) basically look promising.

Appendix

References:

- Alpha Coder (2017). “Support Vector Machine” (blog). URL: https://alphacoder.xyz/support-vector-machine/ (https://alphacoder.xyz/support-vector-machine/). URL:
- Bhalla, Deepanshu. “A COMPLETE GUIDE TO RANDOM FOREST IN R” (blog). URL: https://www.listendata.com/2014/11/random-forest-with-r.html (https://www.listendata.com/2014/11/random-forest-with-r.html)
- Cichosz, Pawel (2015). “Data Mining Algorithms: Explained Using R”. Publisher: Wiley
- Elith, Jane and Leathwick, John (2017). “Boosted Regression Trees for ecological modeling” (CRAN tutorial). URL: https://cran.r-project.org/web/packages/dismo/vignettes/brt.pdf (https://cran.r-project.org/web/packages/dismo/vignettes/brt.pdf)
- Libermann, Neil (2017). “Decision Trees and Random Forests”. URL: https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991 (https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991)
- Liu, Hongbin et al. (2020). “Linear support vector” (subject material). URL: https://learn.jcu.edu.au/bbcswebdav/courses/20-MA5832-ONL-EXT-SP83/HTML/week4/W4_topic2.html (https://learn.jcu.edu.au/bbcswebdav/courses/20-MA5832-ONL-EXT-SP83/HTML/week4/W4_topic2.html)
- Posik, Petr (2015). “Optimal separating hyperplane. Basis expansion. Kernel trick. Support vector machine.” (university presentation). URL: https://cw.fel.cvut.cz/old/_media/courses/ae3m33ui/lectures/svm-slides.pdf (https://cw.fel.cvut.cz/old/_media/courses/ae3m33ui/lectures/svm-slides.pdf)
- Pires de Oliveira, Saulo (2017). “A very basic introduction to Random Forests using R” (scientific blog by Oxford Protein Informatics Group). URL: https://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/ (https://www.blopig.com/blog/2017/04/a-very-basic-introduction-to-random-forests-using-r/)
- Raschka, Sebastian (2016). “How to Select Support Vector Machine Kernels” (university blog). URL: https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html (https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html)