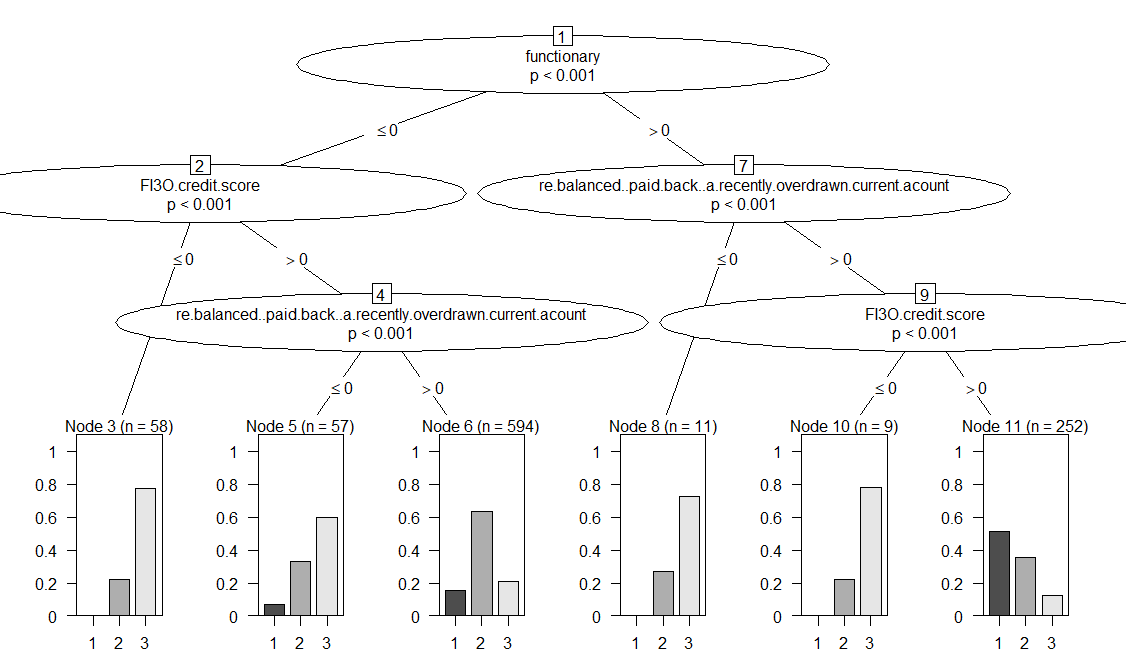
1. The new dataset are now based on the dataset order with the 50/50 training to testing dataset.

This can produce the

1. For building the Decision Tree “party” package was installed.
2. For building the Decision Tree “party” package was installed. The ctree function after the assignment of target variable.

cw.train$credit.rating <-factor(cw.train$credit.rating)



1) functionary <= 0; criterion = 1, statistic = 129.964

2) FI3O.credit.score <= 0; criterion = 1, statistic = 73.752

3)\* weights = 58

2) FI3O.credit.score > 0

4) re.balanced..paid.back..a.recently.overdrawn.current.acount <= 0; criterion = 1, statistic = 41.428

5)\* weights = 57

4) re.balanced..paid.back..a.recently.overdrawn.current.acount > 0

6)\* weights = 594

1) functionary > 0

7) re.balanced..paid.back..a.recently.overdrawn.current.acount <= 0; criterion = 1, statistic = 25.972

8)\* weights = 11

7) re.balanced..paid.back..a.recently.overdrawn.current.acount > 0

9) FI3O.credit.score <= 0; criterion = 1, statistic = 29.608

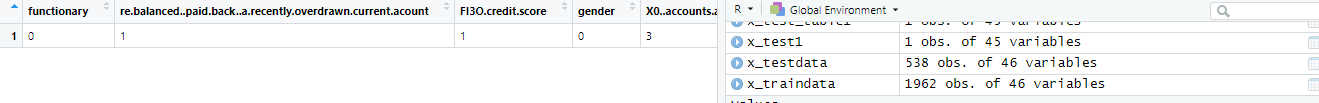
10)\* weights = 9

9) FI3O.credit.score > 0 11)\* weights = 252

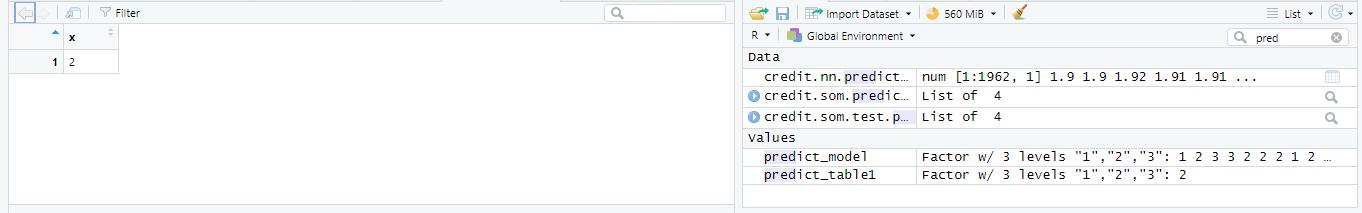
1. The table1 was imported as csv into R.

predict\_table1<-predict(tree, x\_test1)

view(predict\_table1)



For predicting by the decision tree “tree” the class predict was used. The predict function after the assignment predict model returned predict\_table1 as follow which contains one row of data



1. The cw.test was imported into predict as anargument for tree model “tree”. The table view of thepredic\_model is the confusion matrix

predict\_model<-predict(tree, cw.test)

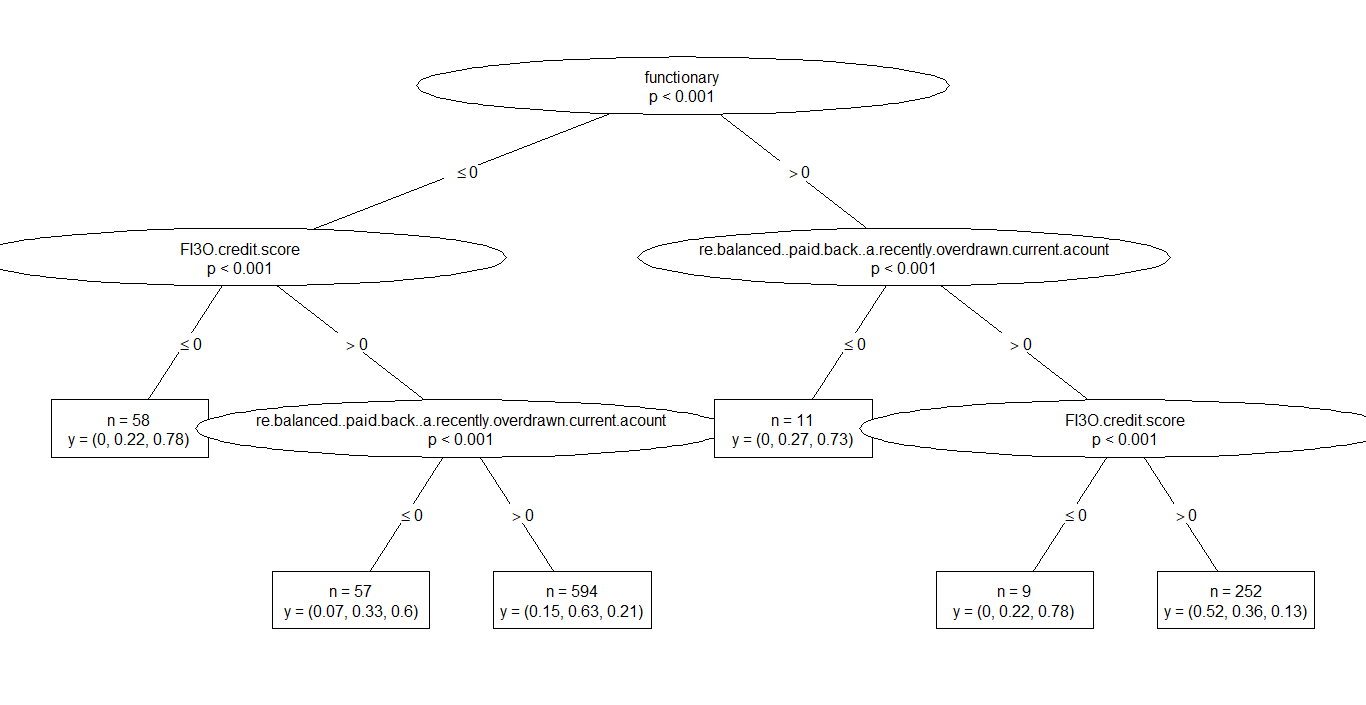
table(cw.train$credit.rating, predict\_model)



In the confusion matrix above for the calculation of overall accuracy, the total number of corrected classified instances is obtained:



1. For having the probabilities we need to use plot configuration as below:



E\_{before} = 1​​ *After* the first split, we had 272 functionary 1 and 709 functionary 0, so the entropy was can be calculated based on probabilities

E\_{right}for rating 1, 2 and 3

*Eright*​​ for rating 1 =- (252/272\*0.52\*log2​(252/272\*0.52​) = 0.15279891

*Eright*​​ for rating 2 =- (11/272\*0.27+2/272\*0.22+252/272\*0.36)log2 (11/272\*0.27+2/272\*0.22+252/272\*0.36) = 0.15948143

*Eright*​​ for rating 3 =- (11/272\*0.73+2/272\*0.78+252/272\*0.13)log2 (11/272\*0.73+2/272\*0.78+252/272\*0.13)= 0.12576011

E\_{left}for rating 1, 2 and 3

*Eright*​​ for rating 1 =- (57/709\*0.07\*log2​(57/709\*0.07) + 594/709\*0.15\*log2​(594/709\*0.15​) ) = 0.12585991

*Eright*​​ for rating 2 =- (58/709\*0.22\*log2​(58/709\*0.22) + 57/709\*0.33\*log2​(57/709\*0.33) + 594/709\*0.63\*log2​(594/709\*0.63​) ) = 0.21969851

*Eright*​​ for rating 3 =- (58/709\*0.78\*log2​(58/709\*0.78) + 57/709\*0.6\*log2​(57/709\*0.6) + 594/709\*0.21\*log2​(594/709\*0.21) ) = 0.2725383

Now that we have the entropies for both branches, we can determine the quality of the split by **weighting the entropy of each branch by how many elements it has**. Since Left Branch has 709 elements and Right Branch has 272, we weight them by 709/981 and 272/981, respectively:

E\_{split} = 0.7223(0.125859905+0.219698511+0.272538304) + 0.2773(0.152798912 +0.159481429+0.125760114) =

*Esplit*​​=0.56792 , as We started with E\_{before} = 1

*Ebefore*​=1 entropy before the split and now are down to 0.56792

**Information Gain = how much Entropy we removed**, so

{Gain} = 1 - 0.56792 = 0.4321

1. We have used ranger to fit random forest with full training 5 and 10 cross folding. The process time and accuracy are reported as below

user system elapsed

142.84 23.29 735903.39

+ Fold1: mtry=6, min.node.size=1, splitrule=gini

- Fold1: mtry=6, min.node.size=1, splitrule=gini

+ Fold1: mtry=6, min.node.size=1, splitrule=extratrees

- Fold1: mtry=6, min.node.size=1, splitrule=extratrees

+ Fold2: mtry=6, min.node.size=1, splitrule=gini

- Fold2: mtry=6, min.node.size=1, splitrule=gini

+ Fold2: mtry=6, min.node.size=1, splitrule=extratrees

- Fold2: mtry=6, min.node.size=1, splitrule=extratrees

+ Fold3: mtry=6, min.node.size=1, splitrule=gini

- Fold3: mtry=6, min.node.size=1, splitrule=gini

+ Fold3: mtry=6, min.node.size=1, splitrule=extratrees

- Fold3: mtry=6, min.node.size=1, splitrule=extratrees

+ Fold4: mtry=6, min.node.size=1, splitrule=gini

- Fold4: mtry=6, min.node.size=1, splitrule=gini

+ Fold4: mtry=6, min.node.size=1, splitrule=extratrees

- Fold4: mtry=6, min.node.size=1, splitrule=extratrees

+ Fold5: mtry=6, min.node.size=1, splitrule=gini

- Fold5: mtry=6, min.node.size=1, splitrule=gini

+ Fold5: mtry=6, min.node.size=1, splitrule=extratrees

- Fold5: mtry=6, min.node.size=1, splitrule=extratrees

Aggregating results

Selecting tuning parameters

Fitting mtry = 6, splitrule = extratrees, min.node.size = 1 on full training set

1. In the following tables the effect of higher cross fold can improve the accuracy around 20% . By fitting a rf the same ouput resulted. The effect of fine Tune was worse than CV folds

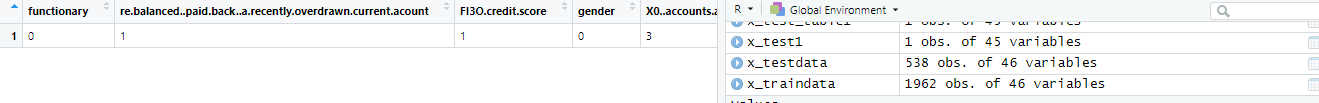


method=”ranger”, CV (n=5), Fine Tune = 1 method=”ranger”, CV (n=10), Fine Tune = 1



method=”ranger” ,CV (n=10), Fine Tune = 5 method=”rf”, CV (n=10), Fine Tune = 1

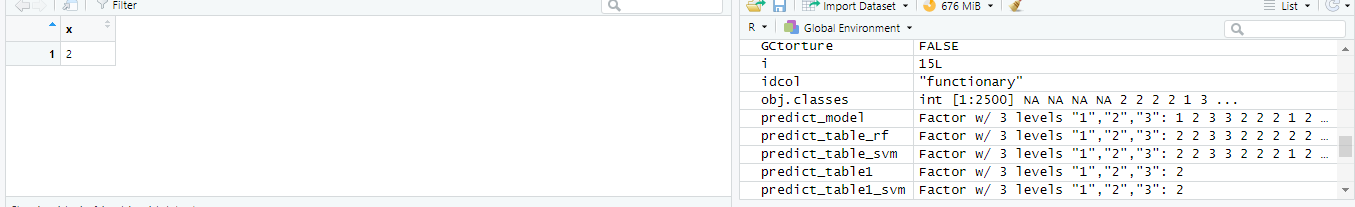
1. The hypothetical Median customer was imported into R from the x\_test\_table1



For predicting by the svm model svm\_model the class predict was used. The predict function after the assignment predict model returned predict\_table1\_svm as follow which contains one row of data

predict\_table1\_svm<-predict(svm\_model, x\_test1)

view(predict\_table1\_svm)



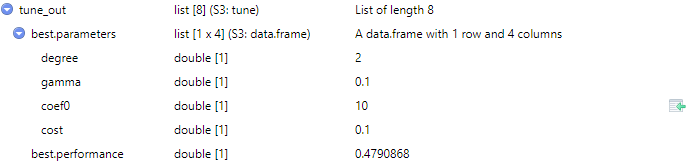
The result is Customer Grade B.

1. It is wanted to train Support Vector Machine by default setting of svm\_model<-svm(credit.rating~.,data=cw.train)

The confusion Matrix of the table(cw.train$credit.rating, predict\_table\_svm) shows the following results



1. for tuning of the svm we have used tune.svm the following results are gained.



The automatic tuning didn’t imprve the results so we have used the manual setting resulted in 10% improvement inaccuracy:

svm\_model<-svm(credit.rating~.,data=cw.train, gamma = 0.05,

coef0 = 0, cost = 1)



4) Naïve Bayes based on the embedded approximation in Bayes theory seems more efficient but the result is around 26% and just not the best.

nb\_model<-naiveBayes(cw.train,cw.train$credit.rating)

nb\_predict<-predict(nb\_default,cw.test,type="class")

view(nb\_predict)

table(cw.test$credit.rating, nb\_predict)

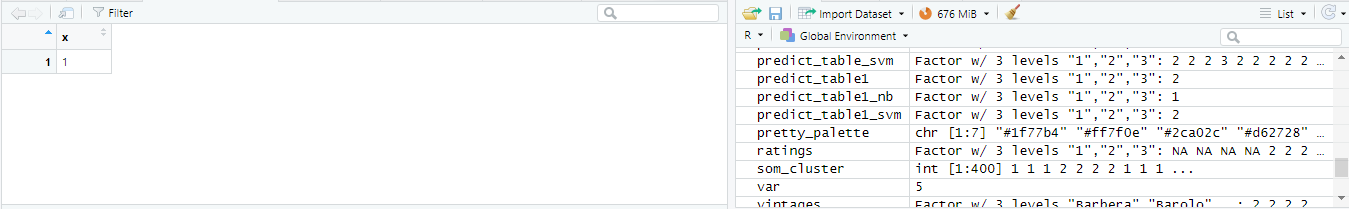


1. For predicting predict\_table1 as follow the nb\_model which resulted in a variable predict\_table1\_nb which contains one row of data

predict\_table1\_nb<-predict(nb\_model, x\_test1)

view(predict\_table1\_nb)

Naïve Base predict the value of 1 for the Mean Customer according to Table 1.



1. By printing the nb\_model the following 20 lines are exported

Naive Bayes Classifier for Discrete Predictors

Here the Naïve Bayes is requested

Call:

naiveBayes.default(x = cw.train, y = cw.train$credit.rating)

List of tables containing the a-priori and conditional probabilities of each class of the response are reported.

A-priori probabilities:

cw.train$credit.rating

1 2 3

0.2303772 0.5127421 0.2568807

The a-priori probability is the estimated probability of a particular class before observing any of the predictors.

Conditional probabilities:

functionary

cw.train$credit.rating [,1] [,2]

1 0.5752212 0.4954066

2 0.1888668 0.3917924

3 0.1865079 0.3902912

re.balanced..paid.back..a.recently.overdrawn.current.acount

cw.train$credit.rating [,1] [,2]

1 0.9823009 0.1321481

2 0.9542744 0.2090974

3 0.8095238 0.3934582

FI3O.credit.score

cw.train$credit.rating [,1] [,2]

1 1.0000000 0.0000000

2 0.9701789 0.1702628

3 0.7936508 0.4054894

gender

cw.train$credit.rating [,1] [,2]

1 0.5265487 0.5004030

2 0.4015905 0.4907079

3 0.3531746 0.4789075

X0..accounts.at.other.banks

cw.train$credit.rating [,1] [,2]

1 2.898230 1.370579

2 3.079523 1.410560

3 3.047619 1.433004

credit.refused.in.past.

cw.train$credit.rating [,1] [,2]

1 0.05752212 0.2333544

2 0.09940358 0.2995010

3 0.21428571 0.4111425

Each conditional probability table corresponds to a predictor column. This resulted in the predict which are calculated based on the above probabilities in a variable called levels.

> summary(nb\_model)

Length Class Mode

apriori 3 table numeric

tables 46 -none- list

levels 3 -none- character

isnumeric 46 -none- logical

call 3 -none- call

5. a) The best result of the previous classifications in terms of overall accuracy is belong to Support Vector Machine. The Confusion Matrix after the tuning outperformed the others(Random Forest (with and without tuning), decision Tree and Naïve Bayes)



b) The category of three is problematic in all of the classifications with more than 200 misclassified instances.







6. We have used the following code to make a binary target. It is either 1 which is A rating or 0 not A Rating

cw.train$credit.rating[cw.train$credit.rating == 2] <- 0

cw.train$credit.rating[cw.train$credit.rating == 3] <- 0

cw.train$credit.rating

1. R has a function glr which simply implement a logistic regression

lr\_model <- glm(credit.rating ~.,family=binomial(link='logit'),data=cw.train)

1. The summary report of the lr\_model

Call:

glm(formula = credit.rating ~ ., family = binomial(link = "logit"),

data = cw.train)

Deviance Residuals:

Min 1Q Median 3Q Max

-2.00215 -0.65353 -0.42668 -0.00012 2.70789

Coefficients:

Estimate Std. Error

(Intercept) -17.551605 429.995589

functionary 1.740533 0.183036

re.balanced..paid.back..a.recently.overdrawn.current.acount 1.501222 0.550965

FI3O.credit.score 16.502759 429.993845

gender 0.577104 0.178807

X0..accounts.at.other.banks -0.027413 0.063141

credit.refused.in.past. -0.935877 0.341848

years.employed 0.672572 0.269126

savings.on.other.accounts -0.548195 0.204670

self.employed. -0.376394 0.236506

max..account.balance.12.months.ago -0.004444 0.062647

min..account.balance.12.months.ago 0.030192 0.063737

avrg..account.balance.12.months.ago 0.124651 0.065028

max..account.balance.11.months.ago -0.010150 0.063924

min..account.balance.11.months.ago -0.110469 0.064328

avrg..account.balance.11.months.ago 0.052783 0.065196

max..account.balance.10.months.ago 0.019305 0.062526

min..account.balance.10.months.ago -0.101696 0.063199

avrg..account.balance.10.months.ago -0.050933 0.065720

max..account.balance.9.months.ago 0.096730 0.062586

min..account.balance.9.months.ago -0.038009 0.064765

avrg..account.balance.9.months.ago -0.032928 0.062640

max..account.balance.8.months.ago -0.019017 0.063459

min..account.balance.8.months.ago -0.041455 0.062710

avrg..account.balance.8.months.ago -0.106852 0.063685

max..account.balance.7.months.ago -0.018414 0.063321

min..account.balance.7.months.ago -0.094176 0.063702

avrg..account.balance.7.months.ago -0.074021 0.061950

max..account.balance.6.months.ago 0.069171 0.064686

min..account.balance.6.months.ago -0.033830 0.062428

avrg..account.balance.6.months.ago -0.025278 0.062786

max..account.balance.5.months.ago 0.015218 0.061902

min..account.balance.5.months.ago -0.088221 0.064391

avrg..account.balance.5.months.ago -0.072089 0.063401

max..account.balance.4.months.ago 0.034718 0.062889

min..account.balance.4.months.ago -0.036728 0.064179

avrg..account.balance.4.months.ago 0.020068 0.063954

max..account.balance.3.months.ago -0.144584 0.062966

min..account.balance.3.months.ago 0.014149 0.064191

avrg..account.balance.3.months.ago -0.010770 0.064635

max..account.balance.2.months.ago 0.100711 0.063196

min..account.balance.2.months.ago -0.065585 0.063059

avrg..account.balance.2.months.ago -0.038225 0.064392

max..account.balance.1.months.ago -0.073012 0.065482

min..account.balance.1.months.ago -0.000658 0.062229

avrg..account.balance.1.months.ago -0.068570 0.064302

z value Pr(>|z|)

(Intercept) -0.041 0.96744

functionary 9.509 < 2e-16 \*\*\*

re.balanced..paid.back..a.recently.overdrawn.current.acount 2.725 0.00644 \*\*

FI3O.credit.score 0.038 0.96939

gender 3.228 0.00125 \*\*

X0..accounts.at.other.banks -0.434 0.66417

credit.refused.in.past. -2.738 0.00619 \*\*

years.employed 2.499 0.01245 \*

savings.on.other.accounts -2.678 0.00740 \*\*

self.employed. -1.591 0.11150

max..account.balance.12.months.ago -0.071 0.94345

min..account.balance.12.months.ago 0.474 0.63572

avrg..account.balance.12.months.ago 1.917 0.05525 .

max..account.balance.11.months.ago -0.159 0.87385

min..account.balance.11.months.ago -1.717 0.08593 .

avrg..account.balance.11.months.ago 0.810 0.41816

max..account.balance.10.months.ago 0.309 0.75750

min..account.balance.10.months.ago -1.609 0.10759

avrg..account.balance.10.months.ago -0.775 0.43834

max..account.balance.9.months.ago 1.546 0.12221

min..account.balance.9.months.ago -0.587 0.55728

avrg..account.balance.9.months.ago -0.526 0.59912

max..account.balance.8.months.ago -0.300 0.76443

min..account.balance.8.months.ago -0.661 0.50858

avrg..account.balance.8.months.ago -1.678 0.09338 .

max..account.balance.7.months.ago -0.291 0.77120

min..account.balance.7.months.ago -1.478 0.13930

avrg..account.balance.7.months.ago -1.195 0.23215

max..account.balance.6.months.ago 1.069 0.28492

min..account.balance.6.months.ago -0.542 0.58788

avrg..account.balance.6.months.ago -0.403 0.68724

max..account.balance.5.months.ago 0.246 0.80581

min..account.balance.5.months.ago -1.370 0.17066

avrg..account.balance.5.months.ago -1.137 0.25553

max..account.balance.4.months.ago 0.552 0.58091

min..account.balance.4.months.ago -0.572 0.56714

avrg..account.balance.4.months.ago 0.314 0.75368

max..account.balance.3.months.ago -2.296 0.02166 \*

min..account.balance.3.months.ago 0.220 0.82554

avrg..account.balance.3.months.ago -0.167 0.86767

max..account.balance.2.months.ago 1.594 0.11102

min..account.balance.2.months.ago -1.040 0.29832

avrg..account.balance.2.months.ago -0.594 0.55276

max..account.balance.1.months.ago -1.115 0.26486

min..account.balance.1.months.ago -0.011 0.99156

avrg..account.balance.1.months.ago -1.066 0.28626

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1058.95 on 980 degrees of freedom

Residual deviance: 820.79 on 935 degrees of freedom

AIC: 912.79

Number of Fisher Scoring iterations: 16

1. Between all variables the \*marker is less than 0.05 level significant \*\* and \*\*\* etc are also significant

* functionary 9.509 < 2e-16 \*\*\*
* re.balanced..paid.back..a.recently.overdrawn.current.acount 2.725 0.00644 \*\*
* FI3O.credit.score 0.038 0.96939
* gender 3.228 0.00125 \*\*
* credit.refused.in.past. -2.738 0.00619 \*\*
* years.employed 2.499 0.01245 \*
* savings.on.other.accounts -2.678 0.00740 \*\*
* max..account.balance.3.months.ago -2.296 0.02166 \*

Intercept and the following variable are spurously

* (Intercept) -0.041 0.96744
* FI3O.credit.score 0.038 0.96939
* X0..accounts.at.other.banks -0.434 0.66417
* self.employed. -1.591 0.11150
* max..account.balance.12.months.ago -0.071 0.94345
* min..account.balance.12.months.ago 0.474 0.63572
* avrg..account.balance.12.months.ago 1.917 0.05525 .
* max..account.balance.11.months.ago -0.159 0.87385
* min..account.balance.11.months.ago -1.717 0.08593 .
* avrg..account.balance.11.months.ago 0.810 0.41816
* max..account.balance.10.months.ago 0.309 0.75750
* min..account.balance.10.months.ago -1.609 0.10759
* avrg..account.balance.10.months.ago -0.775 0.43834
* max..account.balance.9.months.ago 1.546 0.12221
* min..account.balance.9.months.ago -0.587 0.55728
* avrg..account.balance.9.months.ago -0.526 0.59912
* max..account.balance.8.months.ago -0.300 0.76443
* min..account.balance.8.months.ago -0.661 0.50858
* avrg..account.balance.8.months.ago -1.678 0.09338 .
* max..account.balance.7.months.ago -0.291 0.77120
* min..account.balance.7.months.ago -1.478 0.13930
* avrg..account.balance.7.months.ago -1.195 0.23215
* max..account.balance.6.months.ago 1.069 0.28492
* min..account.balance.6.months.ago -0.542 0.58788
* avrg..account.balance.6.months.ago -0.403 0.68724
* max..account.balance.5.months.ago 0.246 0.80581
* min..account.balance.5.months.ago -1.370 0.17066
* avrg..account.balance.5.months.ago -1.137 0.25553
* max..account.balance.4.months.ago 0.552 0.58091
* min..account.balance.4.months.ago -0.572 0.56714
* avrg..account.balance.4.months.ago 0.314 0.75368

1. svm\_model<-svm(credit.rating~.,data=cw.train, gamma=0.1)

We have fitted the svm according to the above code which resulted in

Call:

svm(formula = credit.rating ~ ., data = cw.train, gamma = 0.1)

Parameters:

SVM-Type: eps-regression

SVM-Kernel: radial

cost: 1

gamma: 0.1

epsilon: 0.1

Number of Support Vectors: 981

1. ROCR library with prediction and performance

The main difference between the above are that SVM(red) and logistic regression(orange) is for the calculation. These are based on cutoff value which is different

