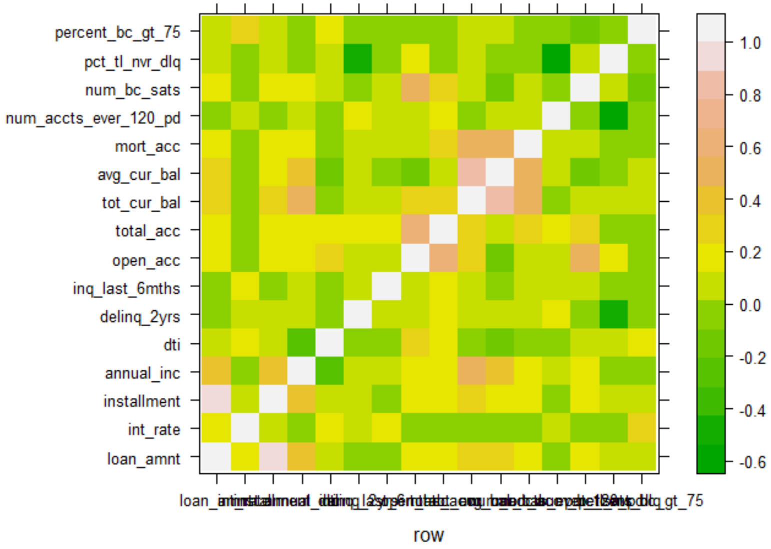
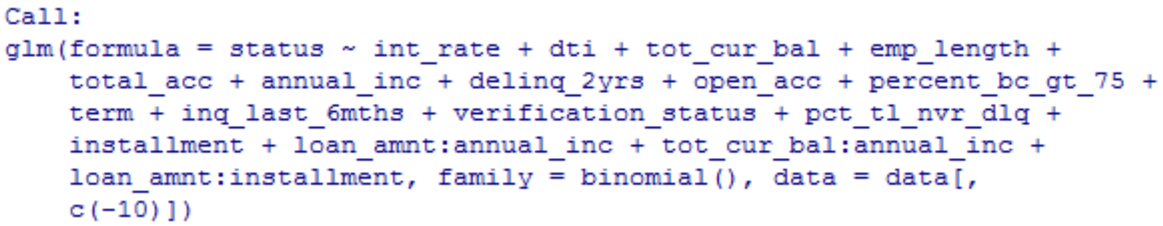
1st: Method: GLM

In cleaned data, there are 16 numerical predictors and 6 categorical predictors.

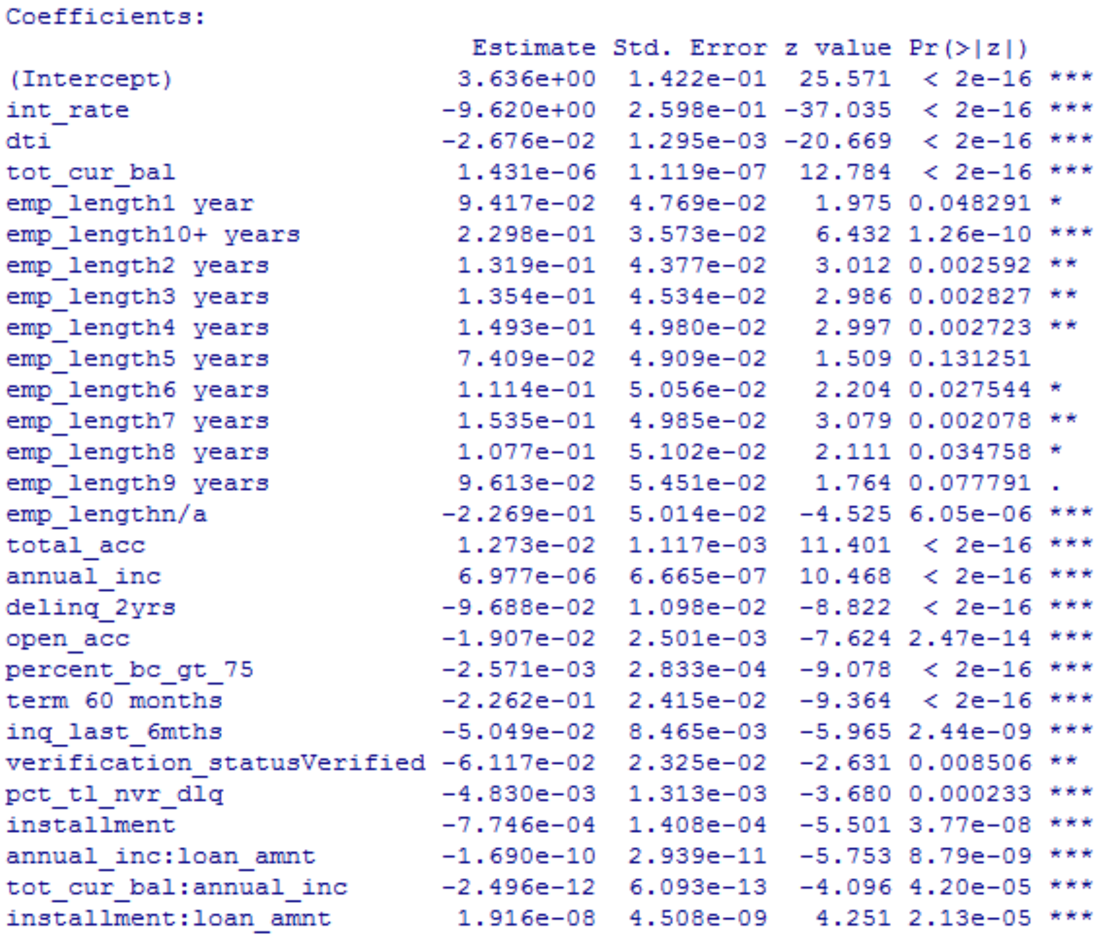
1. Use correlation plot of numerical predictors to seek interactions with abs(correlation) >0.5
2. Build GLM based only on numerical predictors and drop insignificant terms
3. Combine numerical terms and their interaction with each categorical predictor in different GLM, e.g. numerical terms+numerical terms: some categorical, and use drop-in deviance test to determine whether a categorical predictor is useful
4. Relevel categorical predictor and combine all interactions into the full model
5. Forward selection of 20 terms based on AIC starting from intercept-only model.
6. Determine the classifier to obtain categorical fitted value.



❖ Prediction accuracy = 0.804



```
Call:
glm(formula = status ~ int_rate + dti + tot_cur_bal + emp_length +
    total_acc + annual_inc + delinq_2yrs + open_acc + percent_bc_gt_75 +
    term + inq_last_6mths + verification_status + pct_tl_nvr_dlq +
    installment + loan_amnt:annual_inc + tot_cur_bal:annual_inc +
    loan_amnt:installment, family = binomial(), data = data[,
    c(-10)])
```



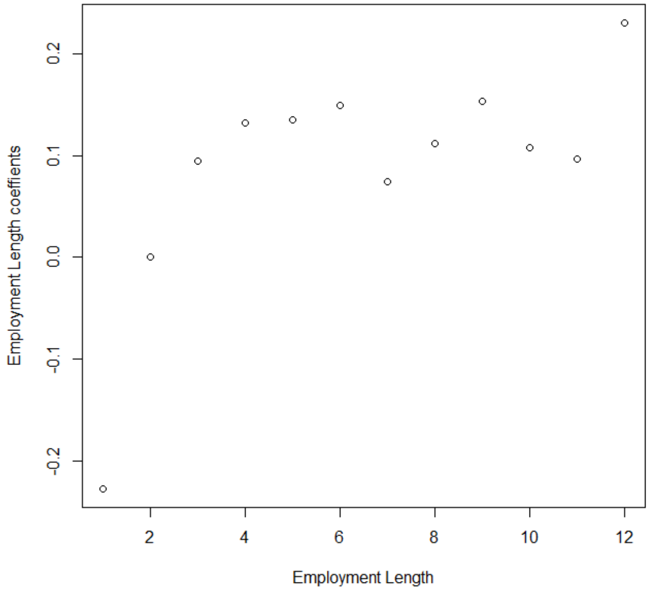
```
Coefficients:
                                Estimate Std. Error z value Pr(>|z|)    
(Intercept)                    3.636e+00  1.422e-01  25.571  < 2e-16 ***
int_rate                      -9.620e+00  2.598e-01 -37.035  < 2e-16 ***
dti                           -2.676e-02  1.295e-03 -20.669  < 2e-16 ***
tot_cur_bal                    1.431e-06  1.119e-07  12.784  < 2e-16 ***
emp_length1 year               9.417e-02  4.769e-02   1.975 0.048291 *  
emp_length10+ years            2.298e-01  3.573e-02   6.432 1.26e-10 ***
emp_length2 years              1.319e-01  4.377e-02   3.012 0.002592 ** 
emp_length3 years              1.354e-01  4.534e-02   2.986 0.002827 ** 
emp_length4 years              1.493e-01  4.980e-02   2.997 0.002723 ** 
emp_length5 years              7.409e-02  4.909e-02   1.509 0.131251    
emp_length6 years              1.114e-01  5.056e-02   2.204 0.027544 *  
emp_length7 years              1.535e-01  4.985e-02   3.079 0.002078 ** 
emp_length8 years              1.077e-01  5.102e-02   2.111 0.034758 *  
emp_length9 years              9.613e-02  5.451e-02   1.764 0.077791 .  
emp_lengthn/a                 -2.269e-01  5.014e-02  -4.525 6.05e-06 ***
total_acc                      1.273e-02  1.117e-03  11.401  < 2e-16 ***
annual_inc                     6.977e-06  6.665e-07  10.468  < 2e-16 ***
delinq_2yrs                   -9.688e-02  1.098e-02  -8.822  < 2e-16 ***
open_acc                      -1.907e-02  2.501e-03  -7.624 2.47e-14 ***
percent_bc_gt_75              -2.571e-03  2.833e-04  -9.078  < 2e-16 ***
term 60 months                -2.262e-01  2.415e-02  -9.364  < 2e-16 ***
inq_last_6mths                -5.049e-02  8.465e-03  -5.965 2.44e-09 ***
verification_statusVerified -6.117e-02  2.325e-02  -2.631 0.008506 ** 
pct_tl_nvr_dlq                -4.830e-03  1.313e-03  -3.680 0.000233 ***
installment                   -7.746e-04  1.408e-04  -5.501 3.77e-08 ***
annual_inc:loan_amnt          -1.690e-10  2.939e-11  -5.753 8.79e-09 ***
tot_cur_bal:annual_inc        -2.496e-12  6.093e-13  -4.096 4.20e-05 ***
installment:loan_amnt          1.916e-08  4.508e-09   4.251 2.13e-05 ***
```

Some interpretation of coefficients:

Annual income has positive coefficient, meaning higher income indicates larger capacity to repay;

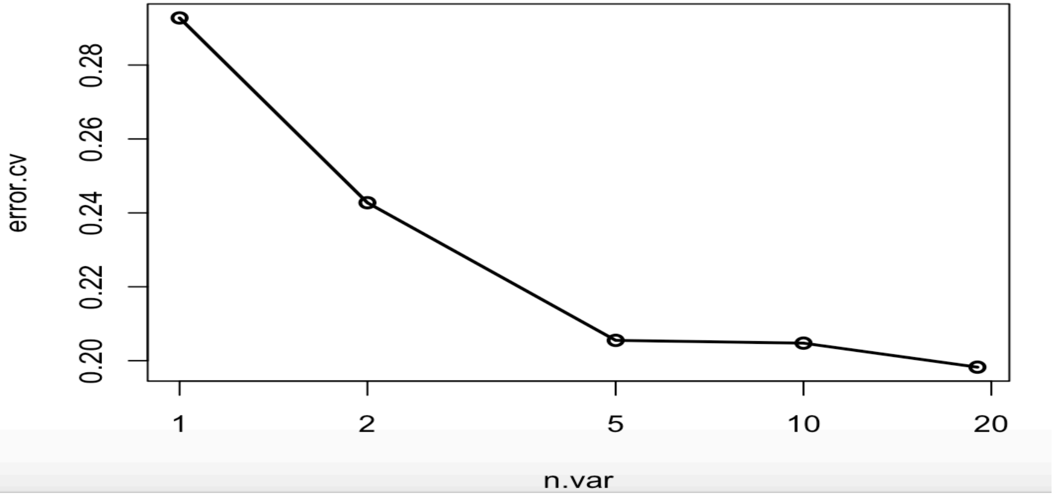
Lower interest rate indicates larger capacity to repay;

Employment length:



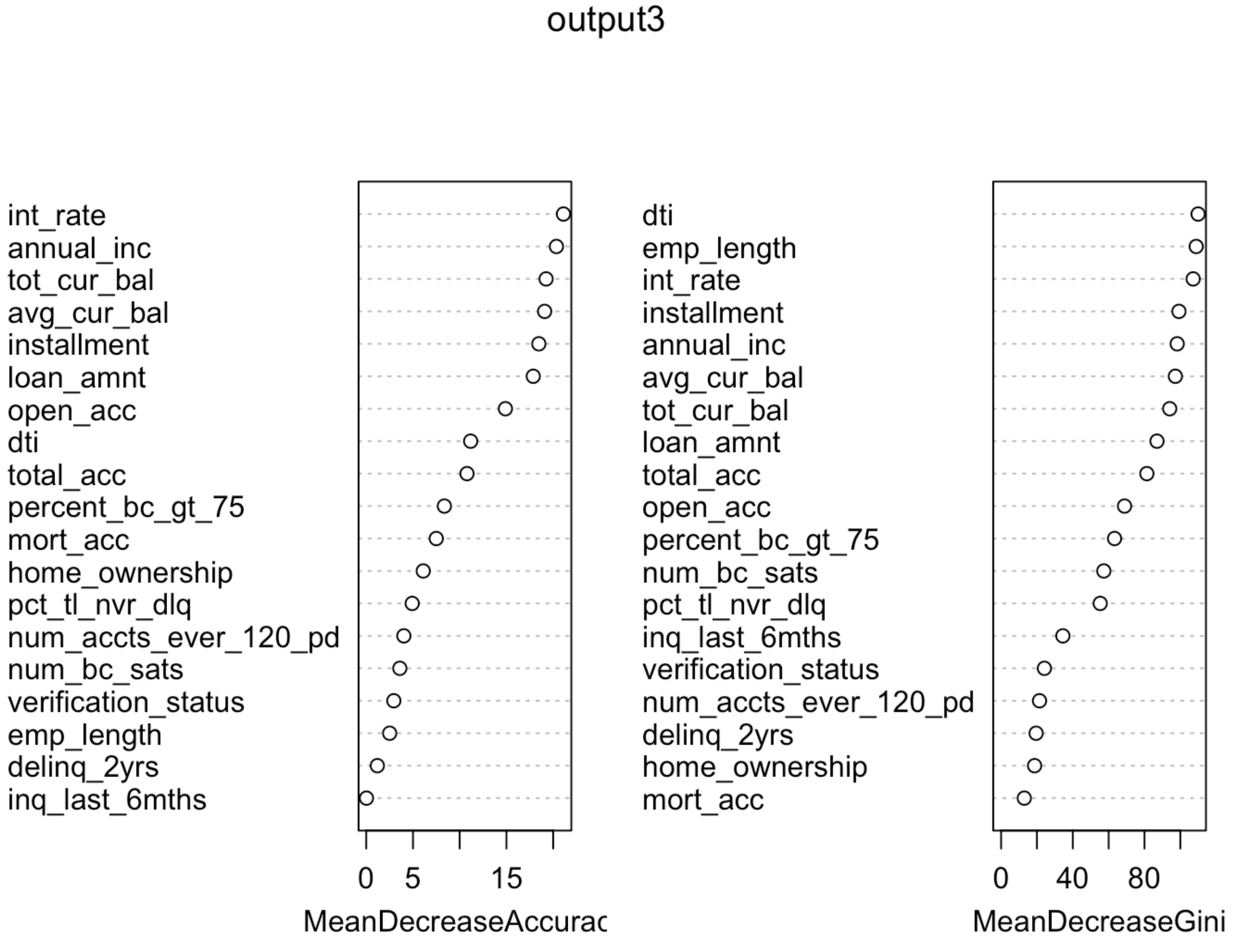
2nd Method: RandomForest

- ❖ 5 fold Cross-Validations for feature selection.

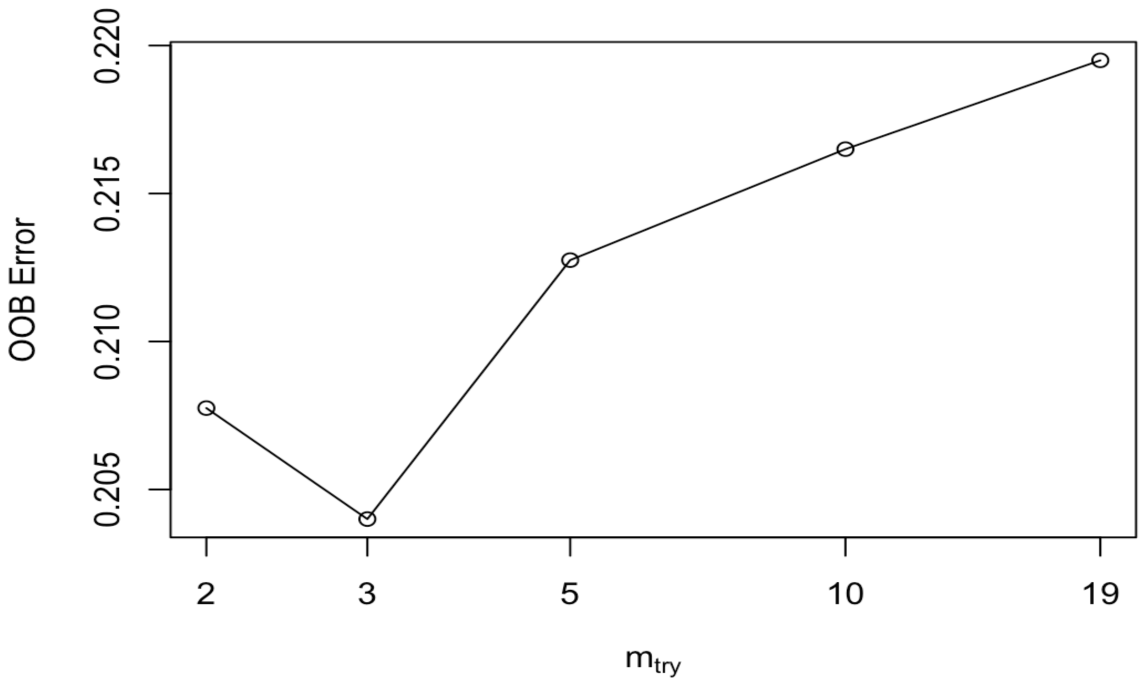


- ❖ Check importance plot and remove non-significant predictors.

- ❖ Rebuild model with 12 predictors



- Tune RandomForest for the optimal mtry parameter



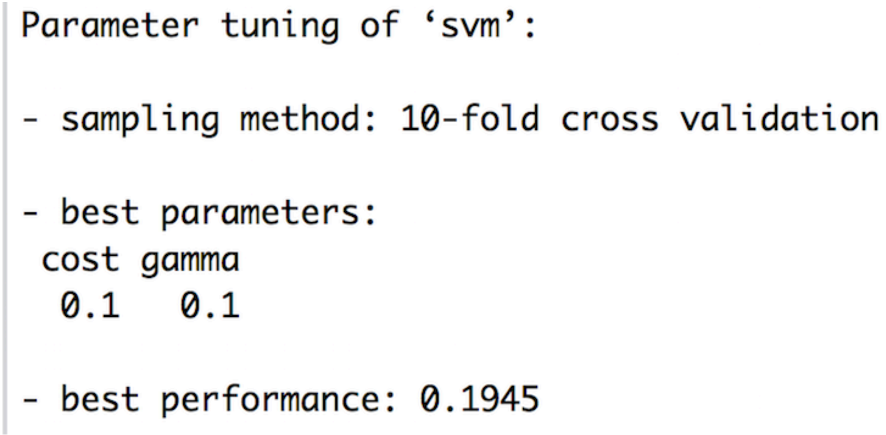
Optimal Parameter :

- ntree = 500, number of trees to grow
- mtry = 3, number of variables randomly sampled as candidates at each split.
- Prediction accuracy = 0.804

3rd Method: SVM with RBF kernel

- ❖ We normalized the dataset (training and testing) for SVM. SVMs assume that the data it works with is in a standard range (0 to 1 roughly). So the normalization of feature vectors prior to feeding them to the SVM is very important. We want to make sure that for each dimension, the values are scaled to lie roughly within this range.

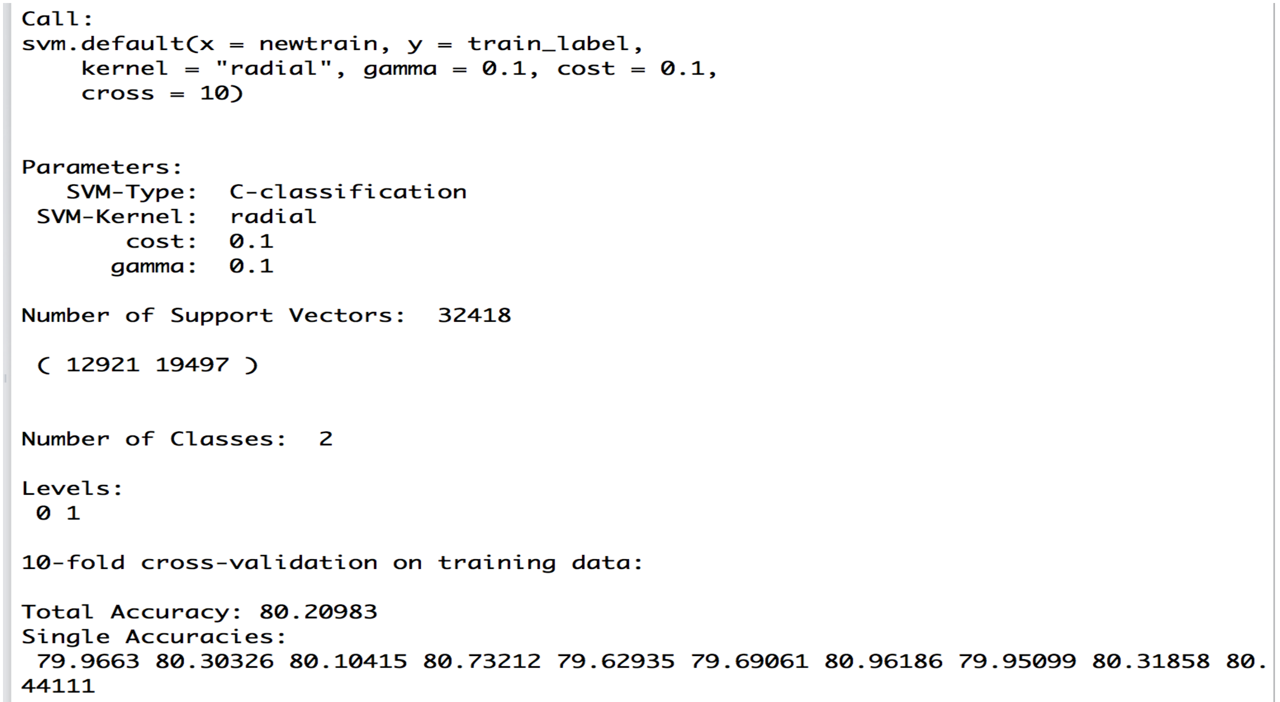
❖We selected optimal parameters (the cost and gamma) by 10-folds CV.



```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
  0.1   0.1

- best performance: 0.1945
```

❖ Prediction accuracy = 0.8055

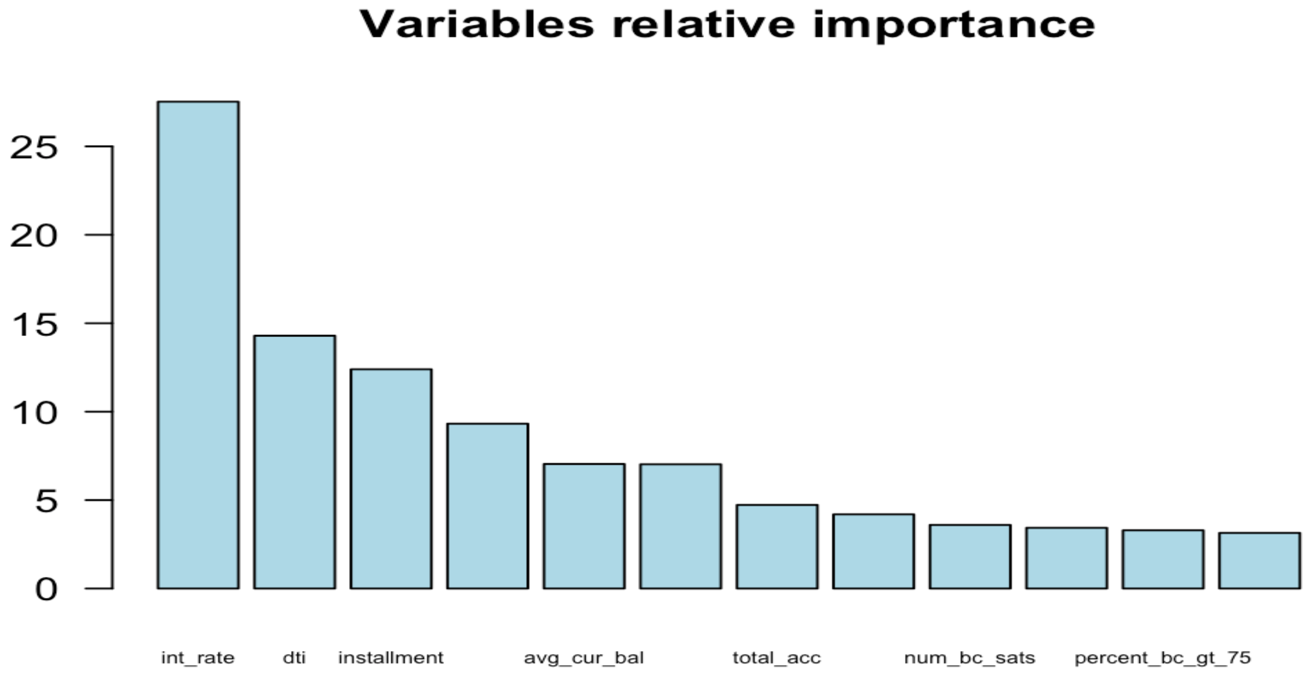
❖Train the model with optimal parameters(cost = 0.1, gamma = 0.1) with 10-folds CV.

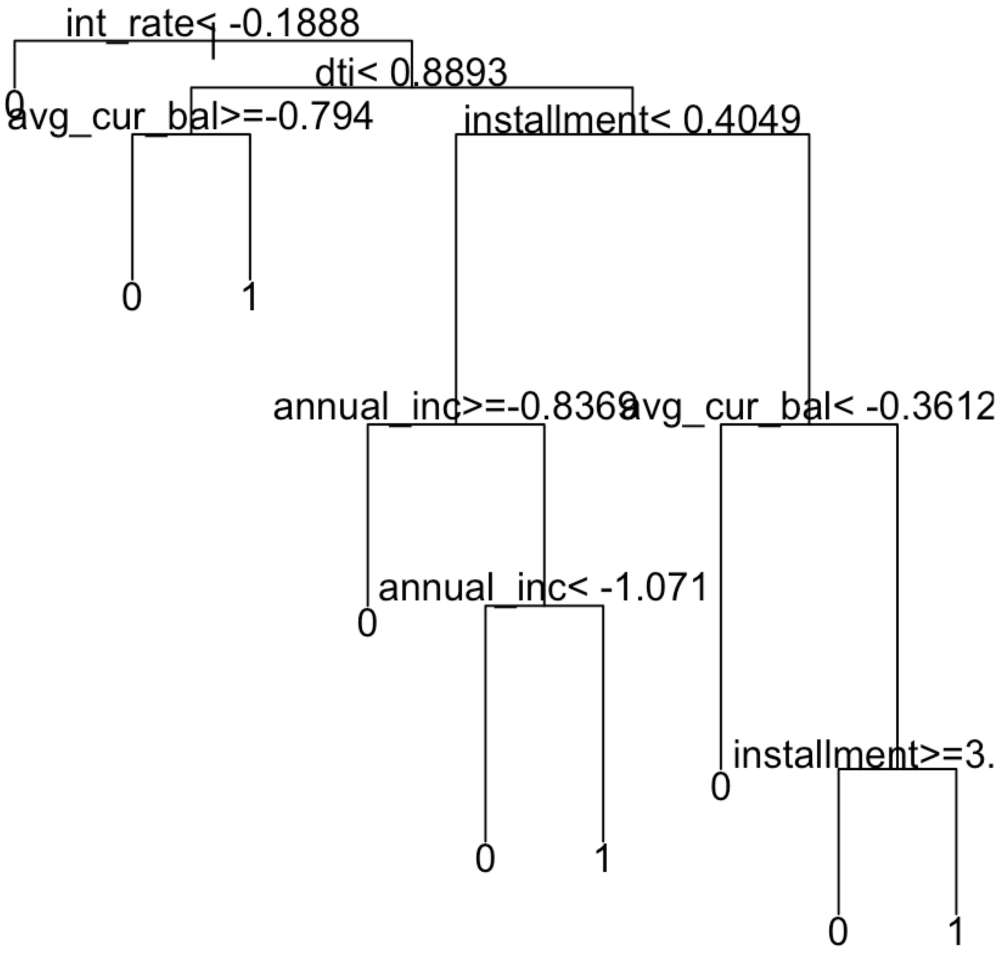


```
Call:
svm.default(x = newtrain, y = train_label,
    kernel = "radial", gamma = 0.1, cost = 0.1,
    cross = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  0.1
      gamma:  0.1

Number of Support Vectors:  32418

 ( 12921 19497 )


Number of Classes:  2

Levels:
 0 1

10-fold cross-validation on training data:

Total Accuracy: 80.20983
Single Accuracies:
 79.9663 80.30326 80.10415 80.73212 79.62935 79.69061 80.96186 79.95099 80.31858 80.
44111
```

4th Method: AdaBoosting and Bagging

- adaBoosting with total 50 iterations for which the number of trees to use. Additionally, a bootstrap sample of the training set is drawn using the weights for each observation on that iteration.
- adaBoosting Prediction accuracy = 0.797
- Bagging Prediction accuracy = 0.804



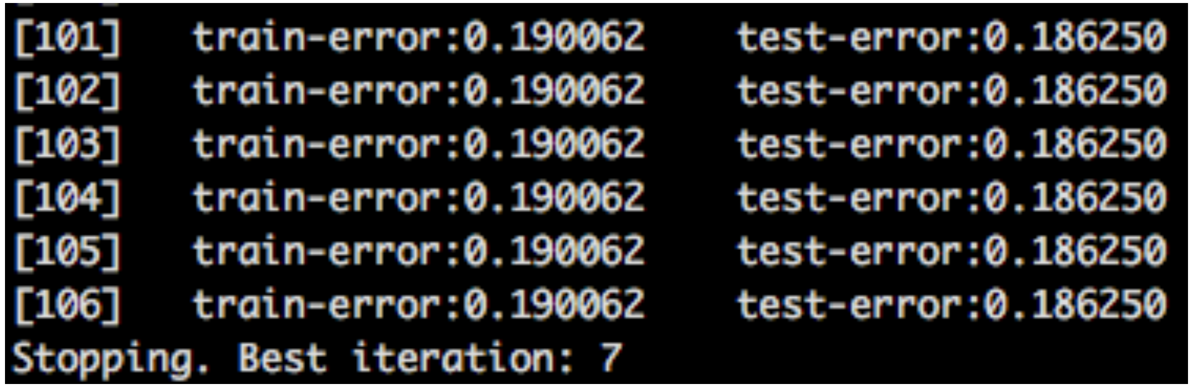


5th Method: Extreme Gradient Boosting

Optimal Parameters for Tree/Linear Booster:

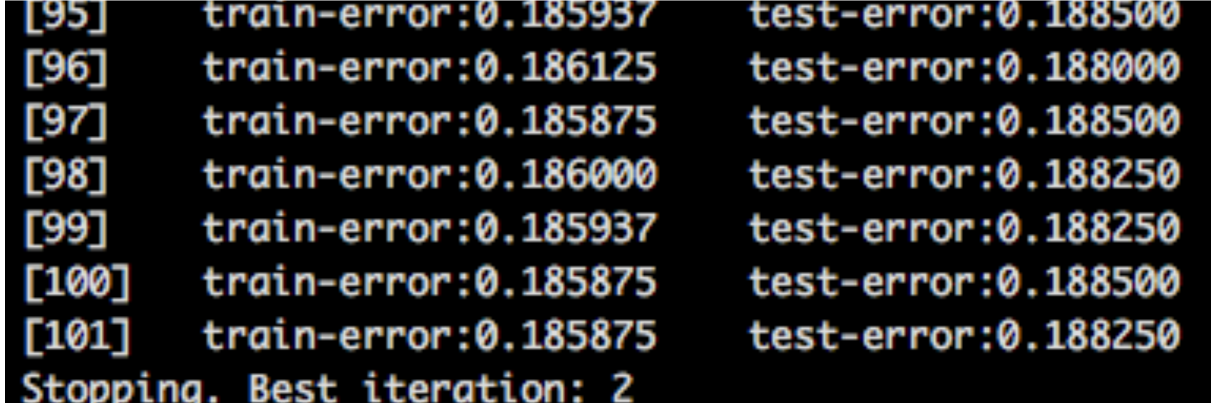
- eta = 0.001 (control the learning rate), which used to prevent overfitting by making the boosting process more conservative. (for tree only)
- max.depth = 4， maximum depth of a tree (for tree only)
- subsample = 0.8, randomly collected 80% of the data instances to grow trees and this will prevent overfitting.

Tree Booster converged at 7th iteration



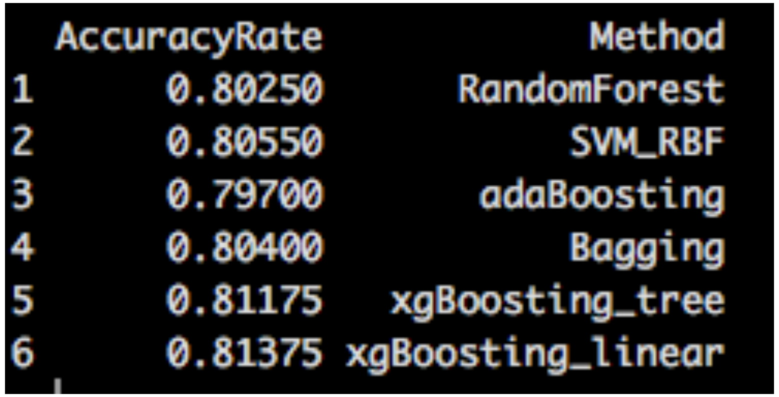
```
[101]   train-error:0.190062    test-error:0.186250
[102]   train-error:0.190062    test-error:0.186250
[103]   train-error:0.190062    test-error:0.186250
[104]   train-error:0.190062    test-error:0.186250
[105]   train-error:0.190062    test-error:0.186250
[106]   train-error:0.190062    test-error:0.186250
Stopping. Best iteration: 7
```

Linear Booster converged at 2th iteration

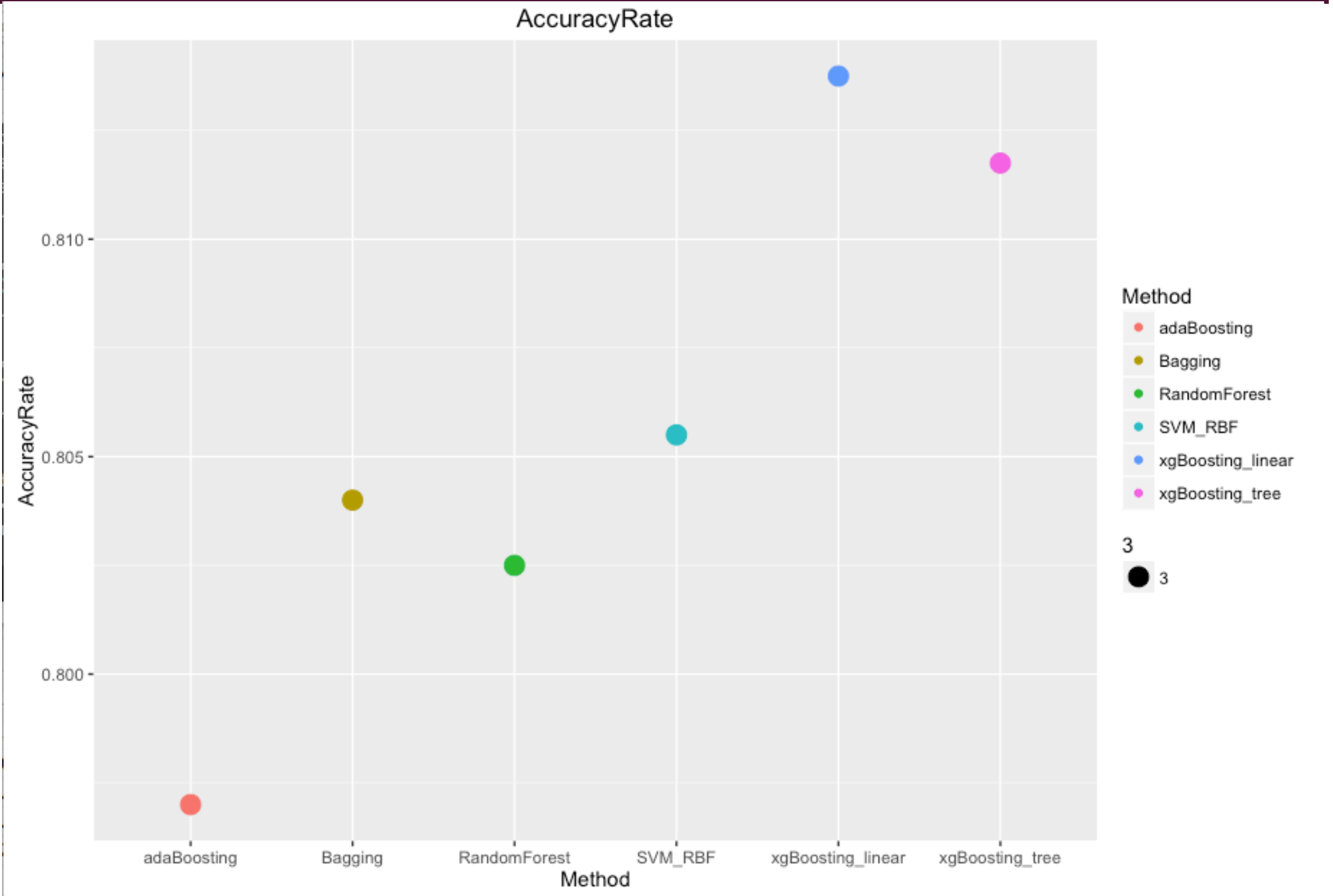


```
[95]    train-error:0.185937    test-error:0.188500
[96]    train-error:0.186125    test-error:0.188000
[97]    train-error:0.185875    test-error:0.188500
[98]    train-error:0.186000    test-error:0.188250
[99]    train-error:0.185937    test-error:0.188250
[100]   train-error:0.185875    test-error:0.188500
[101]   train-error:0.185875    test-error:0.188250
Stopping. Best iteration: 2
```

Final Result



```
  AccuracyRate            Method
1      0.80250      RandomForest
2      0.80550           SVM_RBF
3      0.79700       adaBoosting
4      0.80400           Bagging
5      0.81175   xgBoosting_tree
6      0.81375 xgBoosting_linear
```


AccuracyRate
AccuracyRate
adaBoosting
Bagging
RandomForest
SVM_RBF
xgBoosting_linear
xgBoosting_tree
Method
Method
● adaBoosting
● Bagging
● RandomForest
● SVM_RBF
● xgBoosting_linear
● xgBoosting_tree
● 3