Name: XieJiangFeng

UOW ID: 7358453

Question 1

Import necessary libraries

```
library(kohonen)
library(ggplot2)
library(MASS)
library(Hmisc)
library(caret)
library(rpart)
library(randomForest)
library(e1071)
library(ROCR)
```

```
> # Load the dataset
> cw.all <- read.csv('creditworthiness.csv')
>
> # check for null values
> sum(is.na(cw.all))
[1] 0
>
> # Classified the data with and without credit.rating
> cw.known <- subset(cw.all, credit.rating > 0)
> cw.unknow <- subset(cw.all, credit.rating == 0)
>
> set.seed(123)
> # splite the data into testing and training
> cw.train <- cw.known[1:(nrow(cw.known)/2),]
> cw.test <- cw.known[-(1:(nrow(cw.known)/2)),]
```

```
> # Create a data frame for the hypothetical customer
> medianCust = data.frame()
> newData =c(0,1,1,0,3,0,3,3,0,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3)
> medianCust = rbind(medianCust , newData)
> colnames(medianCust) = names(cw.known)[-46]
```

Question 2

decision tree & random forest

a)

```
> train.rpart=rpart(factor(credit.rating)~., data=cw.train)
> print(train.rpart)
n= 981

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 981 478 2 (0.23037717 0.51274210 0.25688073)
   2) functionary>=0.5 272 142 1 (0.47794118 0.34926471 0.17279412)
     4) re.balanced..paid.back..a.recently.overdrawn.current.acount>=0.5 261 131 1 (0.49808429 0.35249042 0.14942529)
       8) FI3O.credit.score>=0.5 252 122 1 (0.51587302 0.35714286 0.12698413) *
       9) FI3O.credit.score< 0.5 9   2 3 (0.00000000 0.22222222 0.77777778) *
     5) re.balanced..paid.back..a.recently.overdrawn.current.acount< 0.5 11   3 3 (0.00000000 0.27272727 0.72727273) *
   3) functionary< 0.5 709 301 2 (0.13540197 0.57545839 0.28913963)
     6) FI3O.credit.score>=0.5 651 256 2 (0.14746544 0.60675883 0.24577573)
      12) re.balanced..paid.back..a.recently.overdrawn.current.acount>=0.5 594 218 2 (0.15488215 0.63299663 0.21212121) *
      13) re.balanced..paid.back..a.recently.overdrawn.current.acount< 0.5 57  23 3 (0.07017544 0.33333333 0.59649123) *
     7) FI3O.credit.score< 0.5 58  13 3 (0.00000000 0.22413793 0.77586207) *
>
```

b)

```
> # To predict the credit rating of a hypothetical "median" customer.
> cust.pred <- predict(train.rpart, medianCust, type = 'class')
> cust.pred
1
2
Levels: 1 2 3
```

c)

```
> # Make predictions on the test set using the decision tree model
> test.pred <- predict(train.rpart, cw.test, type = 'class')
>
> # Create the confusion matrix
> confusionDT <- table(test.pred, cw.test$credit.rating)
> print(confusionDT)

test.pred   1   2   3
        1 162  85  37
        2  90 361 143
        3   5  21  77
> # Calculate the overall accuracy rate
> accuracyDT <- sum(diag(confusionDT)) / sum(confusionDT)
> print(accuracyDT)
[1] 0.6116208
```

d)

```
> # get the count of all classes in credit.rating using the table() function
> beforeCountFreq = table(cw.train$credit.rating)
>
> #find the probability of each class
> beforeClassProb = beforeCountFreq /sum(beforeCountFreq)
>
> #calculate entropy (before split)
> beforeEntropy = sum( beforeClassProb * log2(beforeClassProb))
>
> # Functionary == 0
> countFreq0 = table(cw.train$credit.rating[cw.train$functionary == 0])
> classProb0 = countFreq0/sum(countFreq0)
> (functionaryEnt0 = -sum(classProb0 * log2(classProb0)))
[1] 1.366963
>
> # Functionary == 1
> countFreq1 = table(cw.train$credit.rating[cw.train$functionary == 1])
> classProb1 = countFreq1/sum(countFreq1)
> (functionaryEnt1 = -sum(classProb1 * log2(classProb1)))
[1] 1.476765
>
> ent = (beforeEntropy - (
+   functionaryEnt0 * sum(countFreq0) +
+     functionaryEnt1 * sum(countFreq1)
+ ) /
+   sum(sum(countFreq0) + sum(countFreq1)))
> print(ent)
[1] -2.883157
```

e), f)

```
> # Confusion matrix
> confusionRF = with(cw.test, table(rf.pred,credit.rating))
> # Overall accuracy rate
> accuarcyRF = sum(diag(confusionRF))/sum(confusionRF)
> accuarcyRF
[1] 0.5300714
```

```
> # Fit to a model using randomForest after the tuning
> RFTuned.cw.train = randomForest(factor(credit.rating)~.,data=cw.train, mtry=12,ntree=
500,stepFactor=2,improve=0.2)
>
> RFTuned.pred = predict(RFTuned.cw.train, cw.test[,-46])
>
> # Confusion matrix
> confusionRFTuned = with(cw.test, table(RFTuned.pred, credit.rating))
> confusionRFTuned
             credit.rating
RFTuned.pred   1   2   3
           1 111  61  27
           2 143 389 161
           3   3  17  69
>
> # Overall accuracy rate
> accuarcyTunedRF = sum(diag(confusionRFTuned))/sum(confusionRFTuned)
> accuarcyTunedRF
[1] 0.5800204
```

Question 3

```
# SVM
```

a)

```
> svmfit = svm(factor(credit.rating)~.,data=cw.train, kernel='radial')
> print(svmfit)

Call:
svm(formula = factor(credit.rating) ~ ., data = cw.train, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  937

>
> # Predict the credit rating of a hypothetical "median“ customer
> predict(svmfit, medianCust, decision.values = TRUE)
1
2
attr(,"decision.values")
       2/1      2/3         1/3
1 1.021296 1.511396 -0.04938262
Levels: 1 2 3
```

b)

```
> # Predict the crefusion matrix for predicting the credit rating from the SVM on the t
est set
> svm.pred = predict(svmfit, cw.test[,-46])
>
> # Generate the confusion matrix
> confusionSVM = with(cw.test, table(svm.pred, credit.rating))
> confusionSVM
        credit.rating
svm.pred   1   2   3
       1 109  56  22
       2 143 393 162
       3   5  18  73
> # Overall accuracy rate
> accuarcySVM = sum(diag(confusionSVM))/sum(confusionSVM)
> accuarcySVM
[1] 0.5861366
```

c)

```
> summary(tune.svm(credit.rating~., data=cw.train, kernel='radial',
+                  cost= 10^c(0:2), gamma = 10^c(-4:-1)))

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
  0.01    1

- best performance: 0.3947356

- Detailed performance results:
   gamma cost     error dispersion
1  1e-04    1 0.4630384 0.06311624
2  1e-03    1 0.4073955 0.06827859
3  1e-02    1 0.3947356 0.07448703
4  1e-01    1 0.4840183 0.06919811
5  1e-04   10 0.4068753 0.06827540
6  1e-03   10 0.4008249 0.07365823
7  1e-02   10 0.5065298 0.08184796
8  1e-01   10 0.4760369 0.06414239
9  1e-04  100 0.4049369 0.07406197
10 1e-03  100 0.4188515 0.07853741
11 1e-02  100 0.5473722 0.08664445
12 1e-01  100 0.4760369 0.06414239

> # Fit a model using SVM
> svmTuned = svm(factor(credit.rating)~.,data = cw.train, kernel='radial',
+                cost=100, gamma=0.0001)
>
> # Predict the values on test set
> svmTuned.pred = predict(svmTuned, cw.test[, -46])
>
> # Produce confusion matrix
> confusionTunedSVM = with(cw.test, table(svmTuned.pred, credit.rating))
>
> # Overall accuracy rate
> accuarcyTunedSVM = sum(diag(confusionTunedSVM))/sum(confusionTunedSVM)
> accuarcyTunedSVM
[1] 0.6034659
```

Question 4

```
# Naive Bayes
```

a)

```
> nb = naiveBayes(credit.rating~., data=cw.train)
> predict(nb, medianCust, type='class')
[1] 1
Levels: 1 2 3
> predict(nb, medianCust, type='raw')
             1          2            3
[1,] 0.9850729 0.01393277 0.0009942948
```

b)

```
> nb.pred = predict(nb, cw.test[,-46])
>
> confusionNB = with(cw.test, table(nb.pred, credit.rating))
> confusionNB
       credit.rating
nb.pred   1   2   3
      1 252 439 173
      2   0   4   6
      3   5  24  78
>
> accuarcyNB = sum(diag(confusionNB))/sum(confusionNB)
> accuarcyNB
[1] 0.3404689
>
> nb
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        1         2         3
0.2303772 0.5127421 0.2568807

Conditional probabilities:
   functionary
Y        [,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
Y        [,1]      [,2]
  1 0.9823009 0.1321481
  2 0.9542744 0.2090974
  3 0.8095238 0.3934582

   FI3O.credit.score
Y        [,1]      [,2]
  1 1.0000000 0.0000000
  2 0.9701789 0.1702628
  3 0.7936508 0.4054894

   gender
Y        [,1]      [,2]
  1 0.5265487 0.5004030
  2 0.4015905 0.4907079
  3 0.3531746 0.4789075

   XO..accounts.at.other.banks
Y       [,1]     [,2]
  1 2.898230 1.370579
  2 3.079523 1.410560
  3 3.047619 1.433004

   credit.refused.in.past.
Y         [,1]      [,2]
  1 0.05752212 0.2333544
  2 0.09940358 0.2995010
  3 0.21428571 0.4111425

   years.employed
Y       [,1]     [,2]
  1 3.013274 1.409429
  2 2.972167 1.412530
  3 3.039683 1.314345

   savings.on.other.accounts
Y       [,1]     [,2]
  1 3.442478 1.854427
  2 3.626243 1.802905
  3 3.420635 1.748548

   self.employed.
Y        [,1]      [,2]
  1 0.1637168 0.3708398
  2 0.2206759 0.4151152
  3 0.2142857 0.4111425

   max..account.balance.12.months.ago
Y       [,1]     [,2]
  1 2.955752 1.453819
  2 2.978131 1.425968
  3 2.940476 1.408719

   min..account.balance.12.months.ago
Y       [,1]     [,2]
  1 2.973451 1.391787
  2 2.956262 1.412126
  3 3.115079 1.393872

   avrg..account.balance.12.months.ago
Y       [,1]     [,2]
  1 3.225664 1.450657
  2 2.982107 1.366094
  3 3.015873 1.414124

   max..account.balance.11.months.ago
Y       [,1]     [,2]
  1 2.884956 1.390458
  2 2.992048 1.412782
  3 3.059524 1.436723

   min..account.balance.11.months.ago
Y       [,1]     [,2]
  1 2.827434 1.442636
  2 2.990060 1.370543
  3 2.984127 1.405647

   avrg..account.balance.11.months.ago
Y       [,1]     [,2]
  1 3.017699 1.391928
  2 2.978131 1.361654
  3 2.996032 1.407147

   max..account.balance.10.months.ago
Y       [,1]     [,2]
  1 3.026549 1.454252
  2 3.001988 1.445558
  3 2.968254 1.413856

   min..account.balance.10.months.ago
Y       [,1]     [,2]
  1 2.792035 1.419273
  2 3.031809 1.434833
  3 2.904762 1.416626

   avrg..account.balance.10.months.ago
Y       [,1]     [,2]
  1 2.946903 1.325582
  2 2.998012 1.410686
  3 3.027778 1.470569

   max..account.balance.9.months.ago
Y       [,1]     [,2]
  1 3.110619 1.467022
  2 2.912525 1.414320
  3 2.980159 1.440591

   min..account.balance.9.months.ago
Y       [,1]     [,2]
  1 2.920354 1.363926
  2 2.914513 1.419362
  3 3.067460 1.385545

   avrg..account.balance.9.months.ago
Y       [,1]     [,2]
  1 2.915929 1.432019
  2 2.926441 1.446440
  3 3.158730 1.370798
```

Question 5

a)

```
> # Create a named vector with the variable names and their values
> accuarcy <- c(accuracyDT, accuarcyRF, accuarcySVM, accuarcyNB,accuarcyTunedRF, accuar
cyTunedSVM)
> names(accuarcy) <- c("accuracyDT", "accuracyRF", "accuracySVM", "accuracyNB","accuarc
yTunedRF", "accuarcyTunedSVM")
>
> # Find the name of the variable with the largest value
> largest_var <- names(accuarcy)[which.max(accuarcy)]
>
> # Print the name of the variable with the largest value
> cat("The variable with the largest value is", largest_var, "with a value of", accuarc
y[largest_var], "\n")
The variable with the largest value is accuracyDT with a value of 0.6116208
```

b)

```
> ranked_accuarcy <- accuarcy[order(accuarcy, decreasing = TRUE)]
> ranked_accuarcy
      accuracyDT accuarcyTunedSVM      accuracySVM  accuarcyTunedRF
       0.6116208        0.6034659        0.5861366        0.5800204
      accuracyRF       accuracyNB
       0.5300714        0.3404689
```

Based on the confusion matrices, all of them struggle with predicting class 3 (credit rating = 3). In all four confusion matrices, the values along the diagonal for class 3 are lower than the values for the other two classes.

Question 6

a), b)

```
> glm.fit <- glm((credit.rating==1)~., data=cw.train,family=binomial)
>
> options(width=130)
> summary(glm.fit)

Call:
glm(formula = (credit.rating == 1) ~ ., family = binomial, data = cw.train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.00215  -0.65353  -0.42668  -0.00012   2.70789

Coefficients:
                                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                                         -17.551605 429.995589  -0.041  0.96744
functionary                                           1.740533   0.183036   9.509  < 2e-16 ***
re.balanced..paid.back..a.recently.overdrawn.current.acount   1.501222   0.550965   2.725  0.00644 **
FI3O.credit.score                                    16.502759 429.993845   0.038  0.96939
gender                                                0.577104   0.178807   3.228  0.00125 **
XO..accounts.at.other.banks                          -0.027413   0.063141  -0.434  0.66417
credit.refused.in.past.                              -0.935877   0.341848  -2.738  0.00619 **
years.employed                                        0.672572   0.269126   2.499  0.01245 *
savings.on.other.accounts                            -0.548195   0.204670  -2.678  0.00740 **
self.employed.                                       -0.376394   0.236506  -1.591  0.11150
max..account.balance.12.months.ago                   -0.004444   0.062647  -0.071  0.94345
min..account.balance.12.months.ago                    0.030192   0.063737   0.474  0.63572
avrg..account.balance.12.months.ago                   0.124651   0.065028   1.917  0.05525 .
max..account.balance.11.months.ago                   -0.010150   0.063924  -0.159  0.87385
min..account.balance.11.months.ago                   -0.110469   0.064328  -1.717  0.08593 .
avrg..account.balance.11.months.ago                   0.052783   0.065196   0.810  0.41816
max..account.balance.10.months.ago                    0.019305   0.062526   0.309  0.75750
min..account.balance.10.months.ago                   -0.101696   0.063199  -1.609  0.10759
avrg..account.balance.10.months.ago                  -0.050933   0.065720  -0.775  0.43834
max..account.balance.9.months.ago                     0.096730   0.062586   1.546  0.12221
min..account.balance.9.months.ago                    -0.038009   0.064765  -0.587  0.55728
avrg..account.balance.9.months.ago                   -0.032928   0.062640  -0.526  0.59912
max..account.balance.8.months.ago                    -0.019017   0.063459  -0.300  0.76443
min..account.balance.8.months.ago                    -0.041455   0.062710  -0.661  0.50858
avrg..account.balance.8.months.ago                   -0.106852   0.063685  -1.678  0.09338 .
max..account.balance.7.months.ago                    -0.018414   0.063321  -0.291  0.77120
min..account.balance.7.months.ago                    -0.094176   0.063702  -1.478  0.13930
avrg..account.balance.7.months.ago                   -0.074021   0.061950  -1.195  0.23215
max..account.balance.6.months.ago                     0.069171   0.064686   1.069  0.28492
min..account.balance.6.months.ago                    -0.033830   0.062428  -0.542  0.58788
avrg..account.balance.6.months.ago                   -0.025278   0.062786  -0.403  0.68724
max..account.balance.5.months.ago                     0.015218   0.061902   0.246  0.80581
min..account.balance.5.months.ago                    -0.088221   0.064391  -1.370  0.17066
avrg..account.balance.5.months.ago                   -0.072089   0.063401  -1.137  0.25553
max..account.balance.4.months.ago                     0.034718   0.062889   0.552  0.58091
min..account.balance.4.months.ago                    -0.036728   0.064179  -0.572  0.56714
avrg..account.balance.4.months.ago                    0.020068   0.063954   0.314  0.75368
max..account.balance.3.months.ago                    -0.144584   0.062966  -2.296  0.02166 *
min..account.balance.3.months.ago                     0.014149   0.064191   0.220  0.82554
avrg..account.balance.3.months.ago                   -0.010770   0.064635  -0.167  0.86767
max..account.balance.2.months.ago                     0.100711   0.063196   1.594  0.11102
min..account.balance.2.months.ago                    -0.065585   0.063059  -1.040  0.29832
avrg..account.balance.2.months.ago                   -0.038225   0.064392  -0.594  0.55276
max..account.balance.1.months.ago                    -0.073012   0.065482  -1.115  0.26486
min..account.balance.1.months.ago                    -0.000658   0.062229  -0.011  0.99156
avrg..account.balance.1.months.ago                   -0.068570   0.064302  -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

>
> # Make predictions for test data
> glm.pred <- predict(glm.fit, newdata=cw.test, type="response")
>
> # Convert predicted probabilities to class labels
> glm.pred.class <- ifelse(glm.pred > 0.5, 1, 2)
>
> # Calculate accuracy
> accuracy <- mean(glm.pred.class == cw.test$credit.rating)
> accuracy
[1] 0.5107034
```

c)

The output displays the coefficients for each independent variable, along with the standard error, z-value, and p-value. The coefficients represent the estimated change in the log odds of having good credit for a one unit increase in the corresponding independent variable, holding all other variables constant. The z-value and p-value indicate the statistical significance of each coefficient, with smaller p-values indicating stronger evidence against the null hypothesis that the coefficient is equal to zero.

Predictors with low p-values are considered statistically significant. In the given output, we can see that functionary, re.balanced..paid.back..a.recently.overdrawn.current.account, gender, age and recently.defaulted all have very low p-values, indicating that they are significant predictors of credit rating.

However, some of the variables have very large standard errors, such as Intercept and FI3O.credit.score, which could indicate that they are not statistically significant.

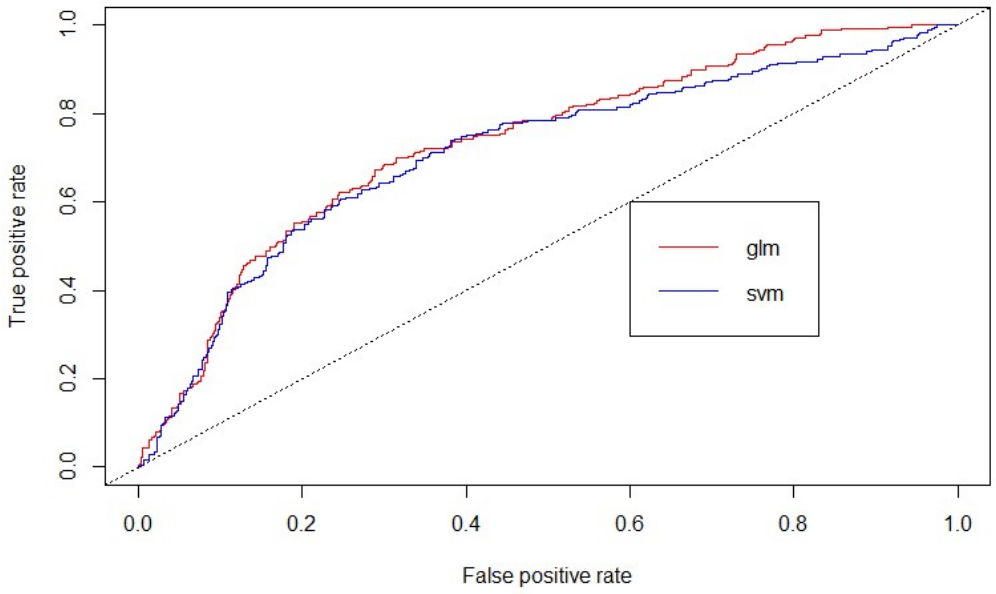
d)

```
> summary(tune.svm((credit.rating==1)~., data=cw.train,kernel='radial',cost=10^c(-2:2),gamma=10^c
(-4:1),type='C'))

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 0.001  100

- best performance: 0.2232323

- Detailed performance results:
   gamma  cost     error dispersion
1  1e-04 1e-02 0.2303649 0.04949816
2  1e-03 1e-02 0.2303649 0.04949816
3  1e-02 1e-02 0.2303649 0.04949816
4  1e-01 1e-02 0.2303649 0.04949816
5  1e+00 1e-02 0.2303649 0.04949816
6  1e+01 1e-02 0.2303649 0.04949816
7  1e-04 1e-01 0.2303649 0.04949816
8  1e-03 1e-01 0.2303649 0.04949816
9  1e-02 1e-01 0.2303649 0.04949816
10 1e-01 1e-01 0.2303649 0.04949816
11 1e+00 1e-01 0.2303649 0.04949816
12 1e+01 1e-01 0.2303649 0.04949816
13 1e-04 1e+00 0.2303649 0.04949816
14 1e-03 1e+00 0.2303649 0.04949816
15 1e-02 1e+00 0.2323954 0.04958832
16 1e-01 1e+00 0.2303649 0.04949816
17 1e+00 1e+00 0.2303649 0.04949816
18 1e+01 1e+00 0.2303649 0.04949816
19 1e-04 1e+01 0.2303649 0.04949816
20 1e-03 1e+01 0.2364873 0.04425234
21 1e-02 1e+01 0.2375180 0.05093962
22 1e-01 1e+01 0.2303649 0.04949816
23 1e+00 1e+01 0.2303649 0.04949816
24 1e+01 1e+01 0.2303649 0.04949816
25 1e-04 1e+02 0.2344465 0.04780915
26 1e-03 1e+02 0.2232323 0.03606685
27 1e-02 1e+02 0.2568955 0.05183418
28 1e-01 1e+02 0.2303649 0.04949816
29 1e+00 1e+02 0.2303649 0.04949816
30 1e+01 1e+02 0.2303649 0.04949816
```

```
> (svm2 = svm(I(credit.rating==1)~., data=cw.train, type='C'))

Call:
svm(formula = I(credit.rating == 1) ~ ., data = cw.train, type = "C")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  664

>
> # Predict the values on test set SVM
> svm.fit.pred = predict(svm2, cw.test[,-46], decision.values= TRUE)
> # Confusion matrix
> confusionSVM = prediction(-attr(svm.fit.pred,'decision.values'),cw.test$credit.rating==1)
```

e)

```
> # Create rocs curve based on prediction
> rocsSVM <- performance(confusionSVM, 'tpr','fpr')
>
> # Predict the values on test set[GLM]
> glm.fit.pred = predict(glm.fit, cw.test[,-46])
> # Confusion matrix
> confusionGLM = prediction(glm.fit.pred, cw.test$credit.rating==1)
> # Create rocs curve based on prediction
> rocsGLM <- performance(confusionGLM, 'tpr', 'fpr')
>
> # Plot the graph
> plot(rocsGLM, col='red')
> plot(rocsSVM, col='blue', add=TRUE)
> abline(0,1,lty=3)
>
> # Add the legend to the graph
> legend(0.6, 0.6, c('glm', 'svm'), col=c("red", "blue"), lty=1:1)
```



The graph is a Receiver Operating Characteristic (ROC) curve that compares the performance of two models, a generalized linear model (GLM) and a support vector machine (SVM), in predicting credit ratings. The x-axis represents the false positive rate (FPR), which is the proportion of actual negative instances (non-defaults) that are incorrectly predicted as positive (defaults). The y-axis represents the true positive rate (TPR), which is the proportion of actual positive instances (defaults) that are correctly predicted as positive.

The graph shows the ROC curves for two models: GLM (in red) and SVM (in blue). The plot indicates that the GLM model has a slightly higher curve compared to the SVM model, which suggests that the GLM model performs slightly better in terms of discrimination accuracy. The dotted line in the plot represents a random classifier, and the closer the ROC curve is to the top left corner, the better the classifier. The legend in the plot shows the corresponding color and label for each model.