Data Analysis Project

Xiwen Hao | Student ID #0020326855 | STAT 4440 – 1001 | Dr. Shuchismita Sarkar

May, 2020

Introduction

This report will utilize logistic regression model, kNN, decision tree, Naive Bayes, support vector machine and neural network as the classifiers to classify the German data set. The best model of each classifier is selected by cross-validation algorithm. To compare the performance of prediction, accuracy, specificity, ROC curve and AUC are used as the criterions.

First, load all the libraries:

ggplot2, caret,rpart,rpart.plot,tree,e1071,kernlab,neuralnet,NeuralNetTools,ROCR,pROC.

Import Data

The following program import the data and convert some variables' attributes.

```
G.data<-read.csv("/Users/GERMAN_DATA.csv",
                 header=TRUE)
G.data$DUR<-as.numeric(G.data$DUR)
G.data$CRED_AMT<-as.numeric(G.data$CRED_AMT)
G.data$INST_RT_PER_DISP_INCM<-as.numeric(G.data$INST_RT_PER_DISP_INCM)
G.data$DUR_RES<-as.numeric(G.data$DUR_RES)
G.data$AGE<-as.numeric(G.data$AGE)
G.data$NUM_CRED<-as.numeric(G.data$NUM_CRED)
G.data$NUM_PEOP_LIABL<-as.numeric(G.data$NUM_PEOP_LIABL)
G.data$Y<-as.factor(G.data$Y)
```

Check if there is any missing data:

```
sapply(G.data, function(x) sum(is.na(x)))
```

There is no missing data.

Preparing Data

Split the dataset into a training dataset (66.7%) and a testing dataset (33.3%). The minimum and maximum of the numeric variables in the training dataset are used to normalize the datasets to the [0,1] range.

Logistic Regression Model

Create the first model without cross-validation:

```
log_fit1 <- train(Y~., data=train_set, method="glm", family="binomial")
# Prediction
pred<-predict(log_fit1, newdata=test_set)
confusionMatrix(pred,test_set$Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 203  54
##          2  29  47
##
##                Accuracy : 0.7508
##                  95% CI : (0.7007, 0.7963)
##             Sensitivity : 0.8750
##             Specificity : 0.4653
```

10-Fold Cross Validation:

```
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TR
UE)
log_fit2 <- train(Y~.,data=train_set, method="glm", family="binomial",
                 trControl = ctrl, tuneLength = 5)
# Prediction
pred <- predict(log_fit2, newdata=test_set)
confusionMatrix(data=pred, test_set$Y)
```

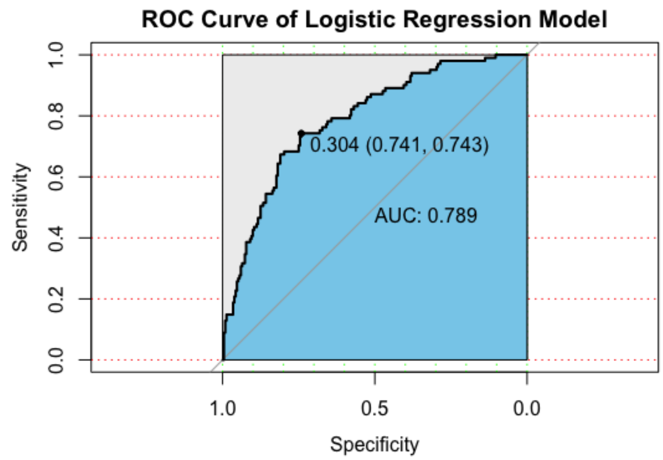
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 203  54
##          2  29  47
##
##                Accuracy : 0.7508
##                  95% CI : (0.7007, 0.7963)
##             Sensitivity : 0.8750
##             Specificity : 0.4653
```

The 10-fold cross-validation algorithm can not improve the accuracy of model.

Draw the ROC curve with AUC value:

```
# Logistic ROC curve
log.predd <- predict(log_fit2, type='prob',test_set, probability = TRUE)
```

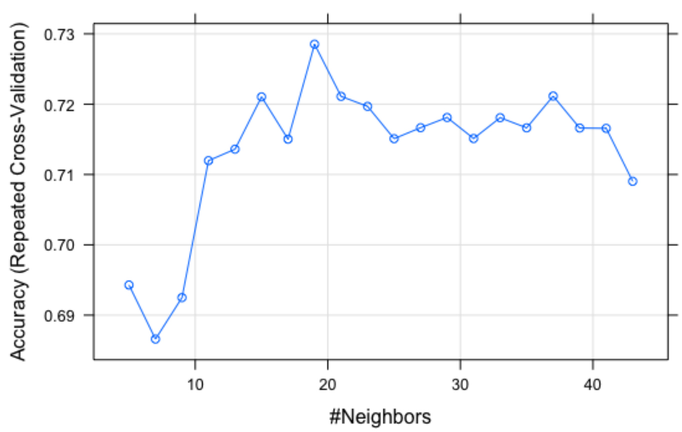
```r
modelroc <- roc(test_set$Y,log.predd[,2])
plot(modelroc, type="S",print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE, main="ROC Curve of Logistic
 Regression Model")
```



kNN

Create the KNN training model:

```r
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TR
UE)
knn_fit <- train(Y ~ ., data = train_set, method = "knn",
                 trControl = ctrl, preProcess = c("center","scale"), tuneLeng
th = 20)
# Plotting yields Number of Neighbours Vs accuracy (based on repeated cross v
alidation)
plot(knn_fit)
```



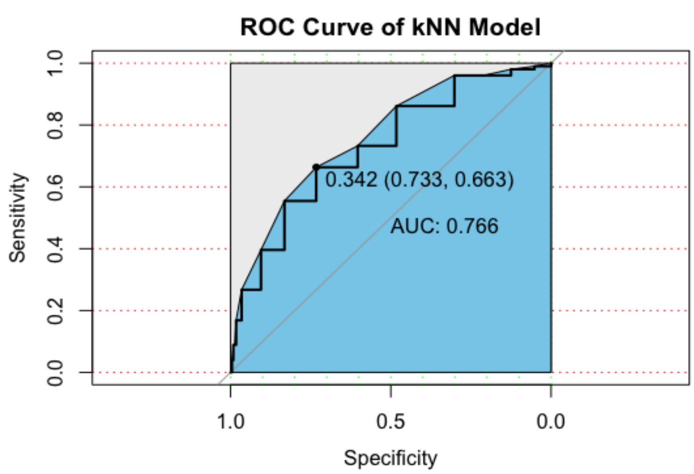
Prediction results and confusion matrix:

```r
knnPredict <- predict(knn_fit,newdata = test_set )
#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(knnPredict, test_set$Y )
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   1   2
##          1 224  74
##          2   8  27
##
##                Accuracy : 0.7538
##                  95% CI : (0.7038, 0.7991)
##             Sensitivity : 0.9655
##             Specificity : 0.2673
```

Draw the ROC curve with AUC value:

```r
# kNN ROC curve
knn.predd <- predict(knn_fit, type='prob',test_set, probability = TRUE)
modelroc <- roc(test_set$Y,knn.predd[,2])
plot(modelroc, type="S",print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE, main="ROC Curve of kNN Mode
l")
```



Decision Tree

Create tree model

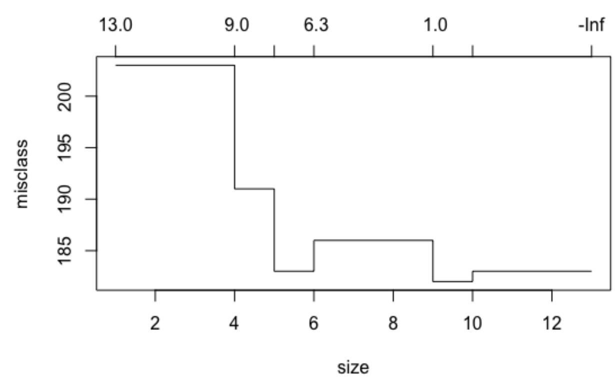
```r
trees <- tree(Y~., train_set)

# Prediction and confusion matrix
treesPredict <- predict(trees,newdata = test_set , type="class")
confusionMatrix(treesPredict, test_set$Y )
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   1   2
##          1 203  58
##          2  29  43
##
##                Accuracy : 0.7387
##                  95% CI : (0.6881, 0.7851)
##             Sensitivity : 0.8750
##             Specificity : 0.4257
```

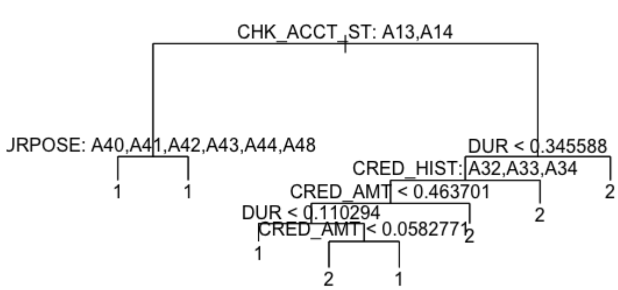
Cross validate to see whether pruning the tree will improve performance

```
# Plot the tree
cv.trees <- cv.tree(trees, FUN = prune.misclass)
plot(cv.trees)
```



It seems like the 6th or 7th sized trees result in the lowest deviance. We can then prune the tree.

```
prune.trees <- prune.tree(trees, best=6)
plot(prune.trees)
text(prune.trees, pretty=0)
```



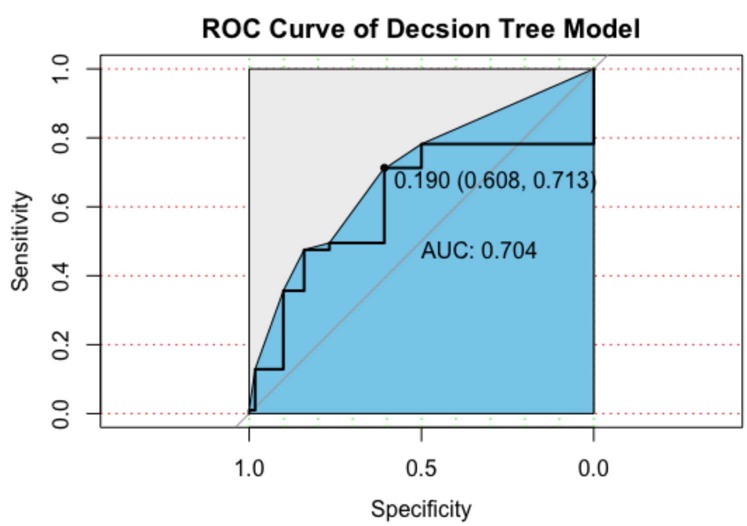
Make predictions based on new tree model

```
prune.treesPredict <- predict(prune.trees,newdata = test_set , type="class")
confusionMatrix(prune.treesPredict, test_set$Y )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 195  53
##          2  37  48
##
##                Accuracy : 0.7297
##                  95% CI : (0.6786, 0.7767)
##             Sensitivity : 0.8405
##             Specificity : 0.4752
```

Draw the ROC curve with AUC value:

```
# Decision Tree ROC curve
dt.predd <- predict(prune.trees,newdata = test_set)
modelroc <- roc(test_set$Y,dt.predd[,2])
plot(modelroc, type="S",print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE, main="ROC Curve of Decsion
Tree Model")
```



Naive Bayes

Create the Naive Bayes model with 10-fold cross validation

```
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TR
UE)
```

```
nb_fit = train(train_set[,1:20],train_set[,21],'nb',
              trControl=ctrl)
```

Prediction results and confusion matrix:

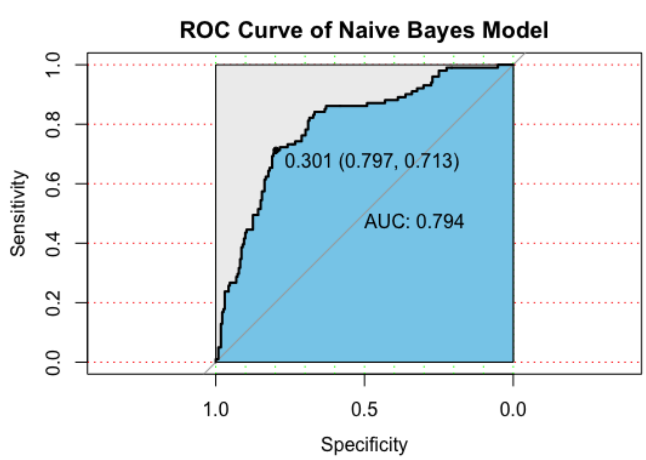
```
nbPredict <- predict(nb_fit$finalModel,newdata = test_set[,1:20] )$class
confusionMatrix(nbPredict, test_set$Y )

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 203  55
##          2  29  46
##
##                Accuracy : 0.7477
##                  95% CI : (0.6975, 0.7935)
##             Sensitivity : 0.8750
##             Specificity : 0.4554
```

Draw the ROC curve with AUC value:

```
# Naive Bayes ROC curve
nb.predd <- predict(nb_fit$finalModel, type='prob',test_set, probability = TR
UE)

modelroc <- roc(test_set$Y,nb.predd$posterior[,2])
plot(modelroc, type="S",print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE, main="ROC Curve of Naive Ba
yes Model")
```



Support Vector Machine

Create the SVM model with 10-fold cross validation and tuning:

```
tc <- tune.control(cross = 10)
tune.out <- tune(svm, Y~.,
                 data = train_set, kernel = "radial",
                 ranges = list(cost = 10^(-1:2),
                               gamma = c(0.25,0.5,1,2,5)),
                 tunecontrol = tc)
print(tune.out) # best parameters: cost=1, gamma=0.25
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1  0.25
##
## - best performance: 0.2638851
```

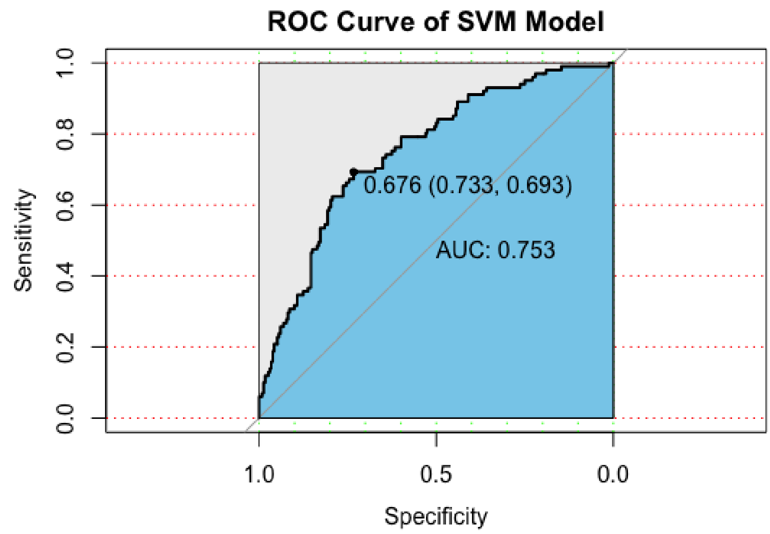
Choose C=1 and gamma=0.25 based on the cross-validation results above to train svm model. Make prediction and show the confusion matrix

```
svm.prediction = predict(tune.out$best.model,newdata=test_set,type='class')
confusionMatrix(svm.prediction,as.factor(test_set$Y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 228  89
##          2   4  12
##
##                Accuracy : 0.7207
##                  95% CI : (0.6692, 0.7683)
##             Sensitivity : 0.9828
##             Specificity : 0.1188
```

Draw the ROC curve with AUC value:

```
# SVM ROC curve
svm_fit2 <- svm(Y~., data =train_set, cost=1, gamma=0.25, probability = TRUE)
svm.predd <- predict(svm_fit2, type='prob',test_set, probability = TRUE)
modelroc <- roc(test_set$Y,attr(svm.predd, "probabilities")[,2])
plot(modelroc, type="S",print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE, main="ROC Curve of SVM Mode
l")
```



Neural Network

Creat Neural Network model with 10-fold cross validation

```
ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TR
UE)
nn_fit <- train(Y~., data = train_set,
              method = 'nnet', preProcess = c('center', 'scale'), trControl =
 ctrl,
              tuneGrid=expand.grid(size=c(10), decay=c(0.1)))
```

Prediction results and confusion matrix:

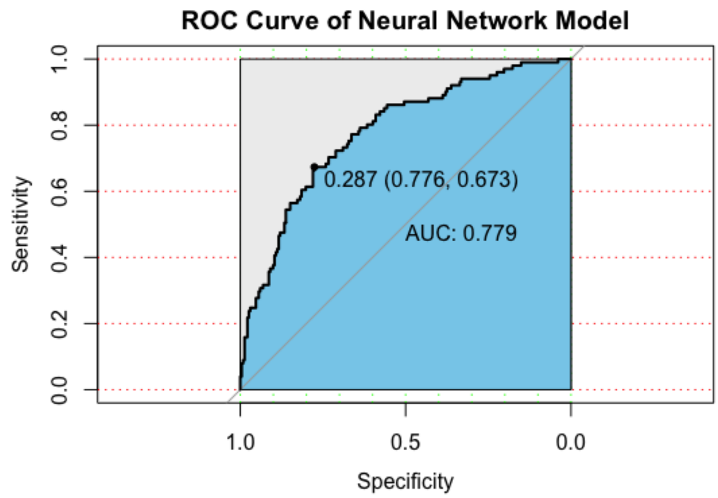
```
nnPredict <- predict(nn_fit,newdata = test_set )
#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(nnPredict, test_set$Y )

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 197  46
##          2  35  55
##
##                Accuracy : 0.7568
##                  95% CI : (0.707, 0.8019)
##             Sensitivity : 0.8491
##             Specificity : 0.5446
```

Draw the ROC curve with AUC value:

```
# Neural Network ROC curve
nn.predd <- predict(nn_fit, type='prob',test_set, probability = TRUE)
```

```r
modelroc <- roc(test_set$Y,nn.predd[,2])
plot(modelroc, type="S",print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("green", "red"), max.auc.polygon=TRUE,
     auc.polygon.col="skyblue", print.thres=TRUE, main="ROC Curve of Neural N
etwork Model")
```



Comparison and Conclusions

The accuracy and specificity of each model are:

```
##  Logistic model's accuracy:  0.7507508 ; Specificity:  0.4653465
##  kNN model's accuracy:  0.7537538 ; Specificity:  0.2673267
##  Decision tree model's accuracy:  0.7297297 ; Specificity:  0.4752475
##  Naive Bayes model's accuracy:  0.7477477 ; Specificity:  0.4554455
##  SVM model's accuracy:  0.7207207 ; Specificity:  0.1188119
##  Neural Network model's accuracy:  0.7567568 ; Specificity:  0.5445545
```

According to the results, the AUC values of most models are higher than 0.7. The neural network model works better than others because of high accuracy and high specificity, which means it is less likely to class a customer as good when they are bad than other models.