```
# Appendix 1

# Libraries
library(tree)
library(e1071)
library(MASS)

# Setup
dataframe = read.csv("creditscoring.csv", dec=',')
n = dim(dataframe)[1]
set.seed(12345)
ids = sample(1:n, n) # sample random order of data

# Task 1 - Split the data into training/validation/training (50/25/25)
training = dataframe[ids[1:floor(n/2)],]                  # 50% of the data
validation = dataframe[ids[(floor(n/2)+1):floor(3*n/4)],] # 25% of data
test = dataframe[ids[(floor(3*n/4)+1):n],]                # 25% of the data


# Task 2 - Deviance vs. Gini
tree.deviance = tree(good_bad~., data=training, split="deviance")
tree.gini = tree(good_bad~., data=training, split="gini")

# plot(tree.deviance)
# text(tree.deviance, pretty=0)
# plot(tree.gini)
# text(tree.gini, pretty=0)

# Fit Test Data
fitted.deviance.test = predict(tree.deviance, test, type="class")
fitted.gini.test = predict(tree.gini, test, type="class")

# Confusion matrixes
cm.deviance.test = table(fitted.deviance.test, test[,20])
cm.gini.test = table(fitted.gini.test, test[,20])

# Missclassification rates
mcr.deviance.test = 1-sum(diag(cm.deviance.test))/sum(cm.deviance.test) # 0.248
mcr.gini.test = 1-sum(diag(cm.gini.test))/sum(cm.gini.test) # 0.304
# mcr.deviance.training = 0.2105   from summary(tree.deviance)
# mcr.gini.training = 0.2368       from summary(tree.gini)
# mcr.deviance < mcr.gini --> Will use Deviance as the measure of impurity



# Task 3 - Tree Pruning
# Empty vectors of score
score.training = rep(0,14)
score.validation = rep(0,14)

# Compute the deviance for the pruned tree's prediction on train set and on validation set, save in score vectors
for(i in 2:14){
  pruned.tree = prune.tree(tree.deviance, best=i)
  prediction = predict(pruned.tree, newdata=validation, type="tree")
  score.training[i] = deviance(pruned.tree)
  score.validation[i] = deviance(prediction)
}

# Plot the deviances for each number of leaves
plot(2:14, score.training[2:14], type="b", col="green",
     xlab="# of leaves", ylab="deviance", ylim=c(250, 600), main="Deviance based on # of leaves")
points(2:14, score.validation[2:14], type="b", col="blue")
legend("topright", legend=c("Train Score", "Validation Score"), lty=1, col=c("green","blue"))

# Optimal leaves = 4, depth=3, var=savings, duration, history
pruned.tree = prune.tree(tree.deviance, best=4)
plot(pruned.tree, main="Pruned tree with 4 leaves")
text(pruned.tree, pretty=0)

# Predictions, Confusion matrix and lastly Missclassification rate for pruned tree on test data
fitted.pruned.tree.test = predict(pruned.tree, newdata=test, type="class")
cm.pruned.tree.test = table(fitted.pruned.tree.test, test[,20])
mcr.pruned.tree.test = 1-sum(diag(cm.pruned.tree.test)/sum(cm.pruned.tree.test)) #0.248



# Task 4 - Naive Bayes
naive.bayes = naiveBayes(good_bad~., data=training)

# Make predictions
```

```
fitted.naive.bayes.training = predict(naive.bayes, newdata=training, type="class")
fitted.naive.bayes.test = predict(naive.bayes, newdata=test, type="class")

# Confusion matrixes
cm.naive.bayes.training = table(fitted.naive.bayes.training, training[,20])
cm.naive.bayes.test = table(fitted.naive.bayes.test, test[,20])

# Missclassification rates
mcr.naive.bayes.training = 1-sum(diag(cm.naive.bayes.training))/sum(cm.naive.bayes.training) # 0.3
mcr.naive.bayes.test = 1-sum(diag(cm.naive.bayes.test))/sum(cm.naive.bayes.test) # 0.3



# Task 5 - Naive Bayes with Loss Matrix
fitted.naive.bayes.training.raw = predict(naive.bayes, newdata=training, type="raw")
fitted.naive.bayes.test.raw = predict(naive.bayes, newdata=test, type="raw")

# Confusion matrixes
cm.naive.bayes.training.loss.matrix = table(fitted.naive.bayes.training.raw[,2]>10/11, training[,20])
cm.naive.bayes.test.loss.matrix = table(fitted.naive.bayes.test.raw[,2]>10/11, test[,20])
```