Problem set 7

Matt He

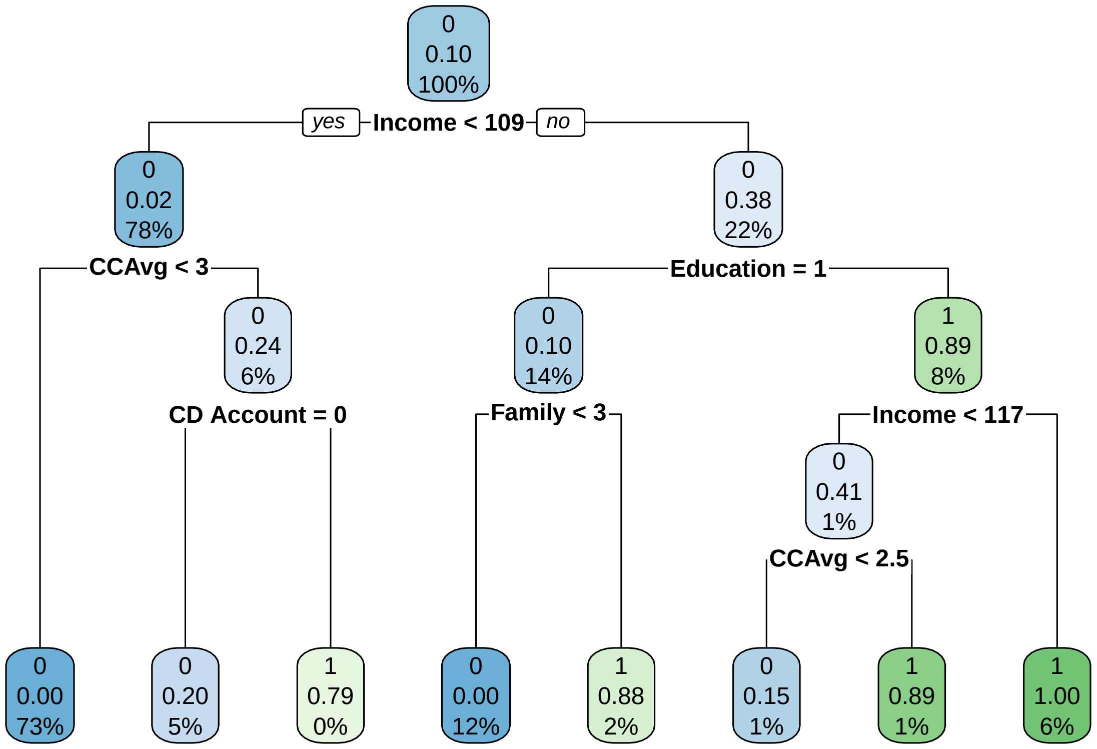
2022-10-17

1. Data Partitioning

```r
set.seed(1122)
df <- read_csv('~/Downloads/UniversalBank.csv', col_types = 'nnnnnnnfnfffff')
df <- df %>% dplyr::select(-ID)
colnames(df)[9] <- 'PL'
sample_set <- sample(nrow(df), nrow(df)*0.7)
training <- df[sample_set,]
testing <- df[-sample_set,]
```

2.Build the best tuned decision tree you can

```r
tree_1 <- rpart(PL~., data = training,
            cp = 0.01,
            minsplit = 1,
            minbucket = 1,
            maxdepth = 5)
rpart.plot::rpart.plot(tree_1)
```

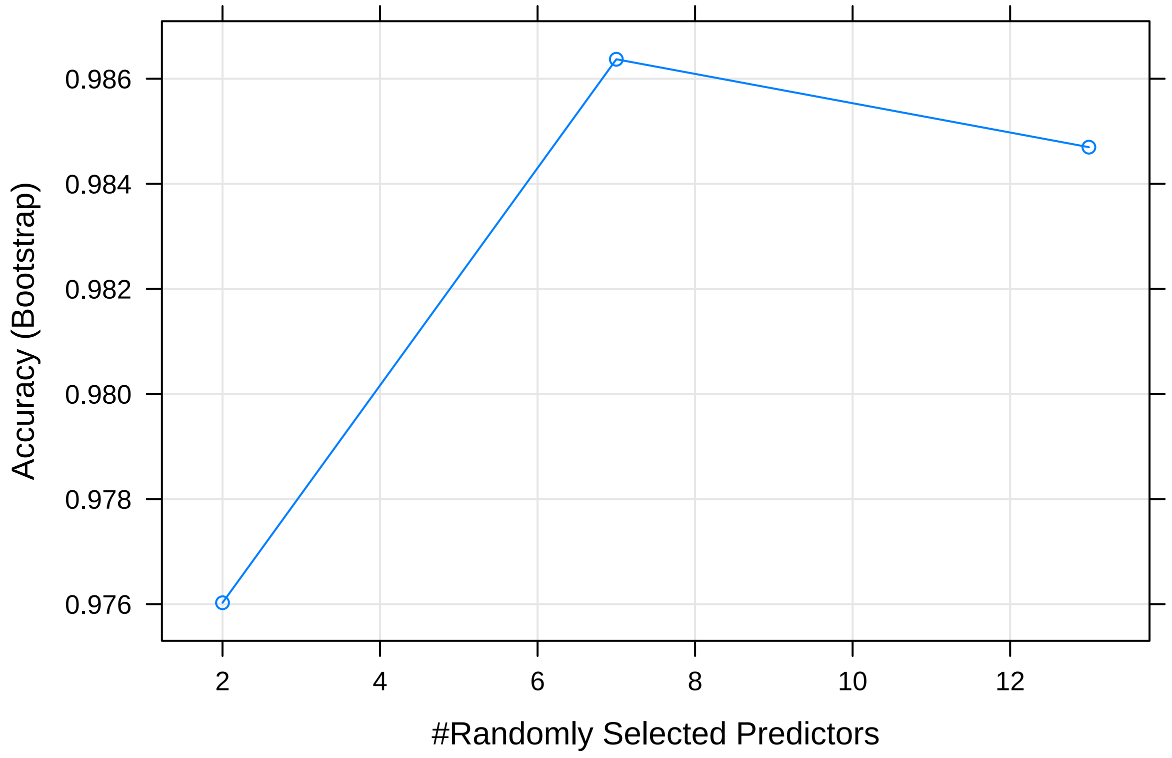


3. Build the best tuned random forest you can

```r
ctrl <- caret::trainControl(method = 'cv', number = 10)
rf_1 = caret::train(PL~.,
                      data = training,
                      method = 'rf',
                      trainControl = ctrl,
                      turneGrid = expand.grid(mtry = c(0,0.1,0.003)))
rf_1
```

```
## Random Forest
##
## 3500 samples
##   12 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3500, 3500, 3500, 3500, 3500, 3500, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9760266  0.8429485
##    7    0.9863712  0.9163403
##   13    0.9846978  0.9061514
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 7.
```

```r
plot(rf_1)
```



4.Build the best tuned gradient boosting machine you can (feel free to engineer features if you'd like)

```
gb_testing <- sample(nrow(training), nrow(training)*(1500/3500))
gb_training <- training[gb_testing,]
```

When I use training as my gradient boosting model, it gave me 3500 prediction, i have no idea why. So I saperate the training again.

```
ctrl_2 <- caret::trainControl(method = 'cv', number = 10)
gb_1 = caret::train(PL ~.,
                  data = gb_training,
                  method = 'xgbTree',
                  metric = 'Accuracy',
                  trControl = trainControl(method = 'none'),
                  tuneGrid = expand.grid(
                    nrounds = 100,
                    max_depth = 6,
                    eta = 0.3,
                    gamma = 0.01,
                    colsample_bytree = 1,
                    min_child_weight = 1,
                    subsample = 1
                  ))
```

5. Compare the precision and sensitivity of all three models on the testing data

```
tree_pred <- predict(tree_1, newdata = testing,type = 'class')
tree_prob <- predict(tree_1, newdata = testing, type = 'prob')[,1]
```

```r
tree_confusionMatrix = confusionMatrix(tree_pred, testing$PL, positive = '1')
tree_confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1346   16
##          1    8  130
##
##                Accuracy : 0.984
##                  95% CI : (0.9763, 0.9897)
##     No Information Rate : 0.9027
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9067
##
##  Mcnemar's Test P-Value : 0.153
##
##             Sensitivity : 0.89041
##             Specificity : 0.99409
##          Pos Pred Value : 0.94203
##          Neg Pred Value : 0.98825
##              Prevalence : 0.09733
##          Detection Rate : 0.08667
##    Detection Prevalence : 0.09200
##       Balanced Accuracy : 0.94225
##
##        'Positive' Class : 1
##
```

```r
rf_pred <- predict(rf_1, newdata = testing)
rf_prob <- predict(rf_1, newdata = testing, type = 'prob')[,1]
rf_confusionMatrix = confusionMatrix(rf_pred, testing$PL, positive = '1')
rf_confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1349   11
##          1    5  135
##
##                Accuracy : 0.9893
##                  95% CI : (0.9827, 0.9939)
##     No Information Rate : 0.9027
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9382
##
##  Mcnemar's Test P-Value : 0.2113
##
##             Sensitivity : 0.92466
##              Specificity : 0.99631
##           Pos Pred Value : 0.96429
##           Neg Pred Value : 0.99191
##               Prevalence : 0.09733
##           Detection Rate : 0.09000
##     Detection Prevalence : 0.09333
##        Balanced Accuracy : 0.96048
##
##         'Positive' Class : 1
##
```

```
gb_pred <- predict(gb_1, new_data = testing,type = 'raw')
gb_prob <- predict(gb_1, new_data = testing, type = 'prob')[,1]
confusionMatrix(gb_pred, testing$PL, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1226  131
##          1  128   15
##
##                Accuracy : 0.8273
##                  95% CI : (0.8072, 0.8461)
##     No Information Rate : 0.9027
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0083
##
##  Mcnemar's Test P-Value : 0.9011
##
##             Sensitivity : 0.10274
##             Specificity : 0.90547
##          Pos Pred Value : 0.10490
##          Neg Pred Value : 0.90346
##              Prevalence : 0.09733
##          Detection Rate : 0.01000
##    Detection Prevalence : 0.09533
##       Balanced Accuracy : 0.50410
##
##        'Positive' Class : 1
##
```

For some reason, I can not run the code at the confusionMatrix at gradient boosting because I have a length of 3500, which is the length of training data. So I used part of training set to train. And this model have surprisingly low sensitivity.

Other than that, first two models have high sensitivity around 90%, the third model only has quite low sentitivity which is only 0.10. Similarly, first two model have high precision(Pos Pred Value) while third one presicion is quite low.

6. Create an ROC plot comparing all three models on the testing data. Which has the greatest AUC?

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
par(pty="s")
tree_1_roc = roc(testing$PL ~ tree_prob,
               plot=TRUE, print.auc=TRUE, print.auc.y=0.3,
               col = "blue", lwd=3, legacy.axes=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

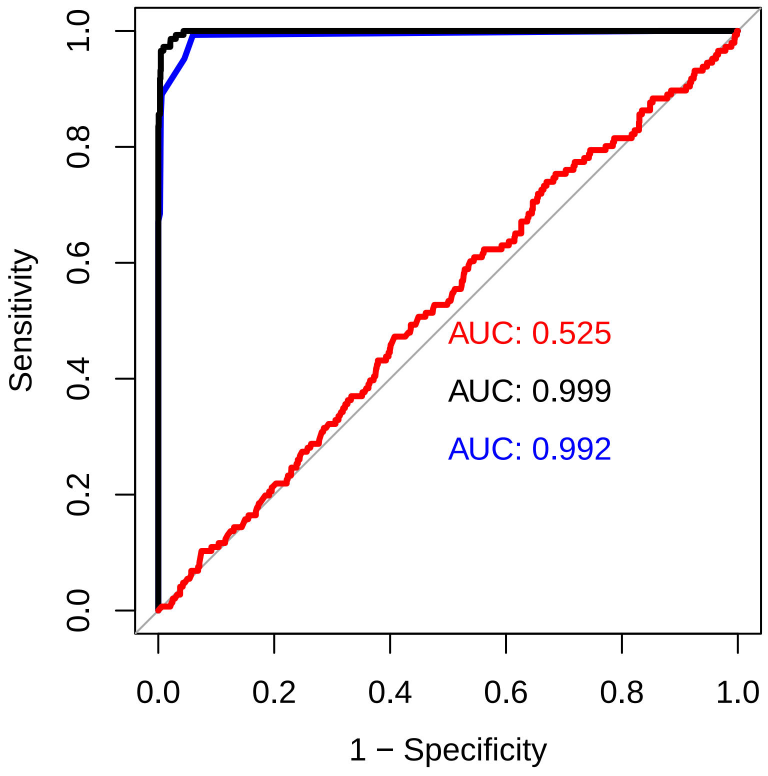
```
rf_prob_roc = roc(testing$PL ~ rf_prob,
                plot=TRUE, print.auc=TRUE, print.auc.y=0.4,
                col = "black", lwd=3, legacy.axes=TRUE, add=TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls > cases
```

```
gb_prob_roc = roc(testing$PL ~ gb_prob,
                plot=TRUE, print.auc=TRUE, print.auc.y=0.5,
                col = "red", lwd=3, legacy.axes=TRUE, add=TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



According to the plot, the Random Forrest tree model have very high AUC value.

7.Write a couple of paragraphs explaining the importance of data partitioning to a manager.

The importance of data partitioning is that we have to test our model by using existing data. We definitely want our model to be more accurate, that is why we have to test our accuracy. By splitting our data into two parts, we can use existing data to double check the model, that is why we partitioning our data. We can not using future data to check accuracy because that is totally unpredictable.

8. Write a couple of paragraphs carefully explaining to a manager how bagging and ensemble models can improve model accuracy and performance.

Bagging and ensemble model is not just limit our data as one, instead we will using multiple, countless model to improve our accuracy. Maybe a similar matephor is a single stick is easy to break, but when you hold 10 or even more sticks, it is unlikely to break those together because they are stronger. Just like our data, just one model is weak, ensemble models will be stronger than just one data. In this case, our random forest model is more accurate than single tree model.