CS 699 Assignment 4

Katherine Rein

```
# Import libraries
library(MASS)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'future':
##   method               from
##   all.equal.connection parallelly
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.3.3
```

```
library(tidyverse)
```

```
## Warning: package 'purrr' was built under R version 4.3.3
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.4     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 4.3.3
```

```
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.3.3
```

```
##
## Attaching package: 'xgboost'
##
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

Problem 1

(1). Generate training and holdout partitions on the data set. Use 1/3 of the data in the holdout. (2). Fit a discriminant analysis model. Use the discriminant analysis model to make predictions on the holdout data. Generate a confusion matrix and measure the model's performance using one or more appropriate metrics of your choice. (3). Fit a neural network model, applying a training grid to tune the parameters. Use the neural network model to make predictions on the holdout data. Generate a confusion matrix and measure the model's performance using one or more appropriate metrics of your choice. (4). Fit a random forest model, applying a training grid to tune the parameters. Use the random forest model to make predictions on the holdout data. Generate a confusion matrix and measure the model's performance using one or more appropriate metrics of your choice. (5). Fit a support vector machine, applying a training grid to tune the parameters. Use the support vector machine to make predictions on the holdout data. Generate a confusion matrix and measure the model's performance using one or more appropriate metrics of your choice. (6). Compare the four models. Which, if any, seems to perform better? Discuss.

```
# Read in data
acc_data = read.csv('accidents1000.csv')

# Ensure the class column is categorical
acc_data$MAX_SEV <- factor(acc_data$MAX_SEV)

# Make data usable
acc_data_scaled <- acc_data %>% mutate(across(SPD_LIM, ~ ( . - min(.)) / (max(.) - min(.))))
acc_data_scaled$MAX_SEV <- factor(make.names(acc_data_scaled$MAX_SEV))

# Split train and test
set.seed(42)
split <- initial_split(acc_data_scaled, prop = 2/3, strata = MAX_SEV)
train <- training(split)
test <- testing(split)

# Fit a DA model
da_model <- lda(MAX_SEV ~ ., data = train)
pred <- predict(da_model, test)

performance_measures <- confusionMatrix(pred$class, test$MAX_SEV)
print('DA MODEL')
```

```
## [1] "DA MODEL"
```

```
print(performance_measures)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  fatal no.injury non.fatal
##   fatal         0         0         2
##   no.injury     1        64        78
##   non.fatal     1       100        88
##
## Overall Statistics
```

```
##
##                Accuracy : 0.4551
##                  95% CI : (0.4008, 0.5102)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : 0.9646
##
##                   Kappa : -0.0788
##
##  Mcnemar's Test P-Value : 0.2559
##
## Statistics by Class:
##
##                      Class: fatal Class: no.injury Class: non.fatal
## Sensitivity              0.000000           0.3902           0.5238
## Specificity              0.993976           0.5353           0.3916
## Pos Pred Value           0.000000           0.4476           0.4656
## Neg Pred Value           0.993976           0.4764           0.4483
## Prevalence               0.005988           0.4910           0.5030
## Detection Rate           0.000000           0.1916           0.2635
## Detection Prevalence     0.005988           0.4281           0.5659
## Balanced Accuracy        0.496988           0.4628           0.4577
```

```
# Fit a neural network
ctrl <- trainControl(method = "CV", number = 10,
                     summaryFunction = defaultSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

nnetGrid <- expand.grid(size = 1:13, decay = seq(0, 2, 0.2))

nnetFit <- train(x = train[, colnames(train) != "MAX_SEV"],
                 y = train$MAX_SEV,
                 method = "nnet",
                 preProc = c("center", "scale"),
                 tuneGrid = nnetGrid,
                 trace = FALSE,
                 maxit = 100,
                 MaxNWts = 1000,
                 trControl = ctrl)

test_pred <- predict(nnetFit, newdata = test)
performance_measures <- confusionMatrix(test_pred, test$MAX_SEV)
print('NEURAL NETWORK')
```

```
## [1] "NEURAL NETWORK"
```

```
print(performance_measures)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  fatal no.injury non.fatal
##   fatal         0         0         0
##   no.injury     1        74        86
##   non.fatal     1        90        82
```

```
##
## Overall Statistics
##
##                Accuracy : 0.4671
##                  95% CI : (0.4126, 0.5222)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : 0.9144
##
##                   Kappa : -0.06
##
##  Mcnemar's Test P-Value : 0.5538
##
## Statistics by Class:
##
##                      Class: fatal Class: no.injury Class: non.fatal
## Sensitivity              0.000000           0.4512           0.4881
## Specificity              1.000000           0.4882           0.4518
## Pos Pred Value                NaN           0.4596           0.4740
## Neg Pred Value           0.994012           0.4798           0.4658
## Prevalence               0.005988           0.4910           0.5030
## Detection Rate           0.000000           0.2216           0.2455
## Detection Prevalence     0.000000           0.4820           0.5180
## Balanced Accuracy        0.500000           0.4697           0.4700
```

```
# Fit a random forest
ctrl <- trainControl(method = "CV",
                     summaryFunction = defaultSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

mtryValues <- seq(2, ncol(train) - 1, by = 1)
rfFit <- caret::train(x = train[, colnames(train) != "MAX_SEV"],
                      y = train$MAX_SEV,
                      method = "rf",
                      ntree = 100,
                      tuneGrid = data.frame(mtry = mtryValues),
                      importance = TRUE,
                      metric = "Accuracy",
                      trControl = ctrl)

pred <- predict(rfFit, newdata = test)
performance_measures <- confusionMatrix(pred, test$MAX_SEV)
print('RANDOM FOREST')
```

```
## [1] "RANDOM FOREST"
```

```
print(performance_measures)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  fatal no.injury non.fatal
##   fatal         0         0         0
##   no.injury     1        60        66
##   non.fatal     1       104       102
```

```
##
## Overall Statistics
##
##                Accuracy : 0.485
##                  95% CI : (0.4303, 0.5401)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : 0.7616
##
##                   Kappa : -0.0267
##
##  Mcnemar's Test P-Value : 0.0148
##
## Statistics by Class:
##
##                      Class: fatal Class: no.injury Class: non.fatal
## Sensitivity              0.000000           0.3659           0.6071
## Specificity              1.000000           0.6059           0.3675
## Pos Pred Value                NaN           0.4724           0.4928
## Neg Pred Value           0.994012           0.4976           0.4803
## Prevalence               0.005988           0.4910           0.5030
## Detection Rate           0.000000           0.1796           0.3054
## Detection Prevalence     0.000000           0.3802           0.6198
## Balanced Accuracy        0.500000           0.4859           0.4873
```

```
# Fit a SVM
train_control <- trainControl(method = "repeatedcv", number = 10, repeats = 5,
                              summaryFunction = defaultSummary)
svmGrid <-  expand.grid(sigma = seq(0.06, 0.3, by = 0.06), C = seq(0.5, 1.5, by = 0.1))
svm.model <- train(MAX_SEV ~ ., data = train, method = "svmRadial",
                   preProc = c("center", "scale"),
                   trControl = train_control, tuneGrid = svmGrid)

pred <- predict(svm.model, test)
performance_measures <- confusionMatrix(pred, test$MAX_SEV)
print('SVM')
```

```
## [1] "SVM"
```

```
print(performance_measures)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  fatal no.injury non.fatal
##   fatal         0         0         0
##   no.injury     1        86        94
##   non.fatal     1        78        74
##
## Overall Statistics
##
##                Accuracy : 0.479
##                  95% CI : (0.4244, 0.5341)
##     No Information Rate : 0.503
##     P-Value [Acc > NIR] : 0.8239
##
```

```
##                   Kappa : -0.0347
##
##  Mcnemar's Test P-Value : 0.3223
##
## Statistics by Class:
##
##                      Class: fatal Class: no.injury Class: non.fatal
## Sensitivity              0.000000           0.5244           0.4405
## Specificity              1.000000           0.4412           0.5241
## Pos Pred Value                NaN           0.4751           0.4837
## Neg Pred Value           0.994012           0.4902           0.4807
## Prevalence               0.005988           0.4910           0.5030
## Detection Rate           0.000000           0.2575           0.2216
## Detection Prevalence     0.000000           0.5419           0.4581
## Balanced Accuracy        0.500000           0.4828           0.4823
```

Discriminant Analysis Model: This model has an accuraccy of 0.4551 which feels very low. I would argue that with such a low accuracy this model is not a good model.

Nueral Network: This model has an accuraccy of 0.4671 which is better but still not great. Overall this is a better model but I would not say it is good.

Random Forest: This model has an accuraccy of 0.485 which is also better but none of these models so far are making massive jumps in accuracy.

SVM: This model is slightly worse than random forest with an accuracy of 0.479.

Overall, Random Forest has the best accuracy so I would most likely choose it for my model of choice.

Problem 2

(1). Generate training and holdout partitions on the data set. Use 1/3 of the data in the holdout. (2). Fit an XGBoost tree model, applying a training grid to tune the parameters. Use the XGBoost tree model to make predictions on the holdout data. Measure the tree's performance using one or more appropriate metrics of your choice. (3). Fit a random forest model, applying a training grid to tune the parameters. Use the random forest model to make predictions on the holdout data. Measure the forest's performance using one or more appropriate metrics of your choice. (4.) Compare the XGBoost tree model to the random forest model. Which, if any, seems to perform better? Discuss.

```r
# Load data (predictive column = Revenue)
rest_data = read.csv('restaurantdata.csv')

# Train test split
set.seed(42)
split <- initial_split(rest_data, prop = 2/3, strata = Revenue)
train <- training(split)
test <- testing(split)

# Fit an XGBoost tree
xgb_control = trainControl(
  method = "cv", number = 10,
  summaryFunction = defaultSummary
```

```
)

xgbGrid <- expand.grid(
  nrounds = c(100, 200),
  eta = c(0.1),
  max_depth = c(2),
  gamma = c(0),
  colsample_bytree = 1,
  min_child_weight = c(1),
  subsample = c(0.7)
)

xgbModel <- train(
  Revenue ~ .,
  data = train,
  method = "xgbTree",
  trControl = xgb_control,
  tuneGrid = xgbGrid,
  metric = "RMSE",
  verbose = FALSE,
  verbosity = 0
)


pred <- predict(xgbModel, test)
performance_measures <- postResample(pred, test$Revenue)
print('XGBOOST TREE')
```

```
## [1] "XGBOOST TREE"
```

```
print(performance_measures)
```

```
##         RMSE     Rsquared          MAE
## 1.079798e+04 9.983985e-01 8.506753e+03
```

```
# Random Forest
ctrl <- trainControl(method = "CV",
                     summaryFunction = defaultSummary,
                     number = 5)

mtryValues <- c(3, 6, 9, 12)
#mtryValues <- seq(2, ncol(train) - 1, by = 1)
rfFit <- train(Revenue ~ ., data = train,
               method = "rf",
               ntree = 100,
               tuneGrid = data.frame(mtry = mtryValues),
               importance = TRUE,
               metric = "RMSE",
               trControl = ctrl)

pred <- predict(rfFit, newdata = test)
performance_measures <- postResample(pred, test$Revenue)
print('RANDOM FOREST')
```

```
## [1] "RANDOM FOREST"
```

```r
print(performance_measures)
```

```
##         RMSE     Rsquared          MAE
## 1.011641e+04 9.986573e-01 7.566912e+03
```

XGBoost Tree: The R squared is 0.9984 which is really good. This means that the model accounts for 99.8% of the variability in the data. This is a really good model.

Random Forest: The R squared is 0.9987 which is also really good.

From just looking at the R sqaured values, the Random Forest model seems to be better. This is because it has a higher R squared value. The higher the R squared value the more the model is predicting the varibility of the data. However, because they are so similar and XGBoost Tree took significantly less time to run I would probably still choose XGBoost Tree