Statistical Learning, LAB 3

Gustav Felländer(gufe0008), Axel Eriksson(axer0005)

2023-12-07

Task 1

After reading through the work of Megan Risdal we imported her data set "full" into the R environment and set a seed for repeatability.

```
# Load the relevant packages
library(randomForest)
library(caret)

set.seed(721)
# Load the data set
load("full")
df <- as.data.frame(full)
```

However, some clean-up is needed for the data set, and therefore we removed some of the irrelevant columns and removed rows with NA values in the survived column. Finally, we checked that the dataset has no more NA values.

```
# Clean up the data set, remove Na values

# Remove columns from data set
columns_to_remove <- c("Deck", "PassengerId", "Ticket", "Cabin", "Surname", "Family", "Child")
df <- df[, -which(names(df) %in% columns_to_remove)]

# Remove rows with Survived NA values
df <- df[-which(is.na(df$Survived)), ]
df$Survived <- as.factor(df$Survived)


# Check if df has NAs
print(any(is.na(df)))
```

```
## [1] FALSE
```

Then, we perform the train, test split of the data where we use 80% for training and 20% for validation.

```
# Train test split
passengers <- nrow(df)
id <- sample(1:passengers, 0.8*passengers)
train <- df[id, ]
test <- df[-id, ]
```

Now it is time for training, and fitting the random forest to the data. For this case we use 100 trees in our forest. Then, we show the results using the confusionMatrix() function in the caret library. Here we can see that the model seems to perform quite well with an accuracy of 88.8% and a kappa statistic of 0.762.

```
# Train the actual model
model <- randomForest(Survived~., data = train,
                      ntree = 100, importance = T)


# Try on test set
predictions <- predict(model, test)
cm <- confusionMatrix(predictions, test$Survived)
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 105  17
##          1   4  53
##
##                Accuracy : 0.8827
##                  95% CI : (0.8263, 0.9259)
##     No Information Rate : 0.6089
##     P-Value [Acc > NIR] : 3.712e-16
##
##                   Kappa : 0.7452
##
##  Mcnemar's Test P-Value : 0.008829
##
##             Sensitivity : 0.9633
##             Specificity : 0.7571
##          Pos Pred Value : 0.8607
##          Neg Pred Value : 0.9298
##              Prevalence : 0.6089
##          Detection Rate : 0.5866
##    Detection Prevalence : 0.6816
##       Balanced Accuracy : 0.8602
##
##        'Positive' Class : 0
##
```

Lastly, we show the importances of each feature rounded to three decimal digits.

```
# Show the importance of each feature
round(model$importance, 3)
```

```
##               0     1 MeanDecreaseAccuracy MeanDecreaseGini
## Pclass    0.030 0.074                0.047           23.053
## Name     -0.004 0.008                0.001           38.762
## Sex       0.083 0.106                0.092           43.345
## Age       0.001 0.012                0.005           32.773
## SibSp     0.005 0.002                0.004            5.700
```

```
## Parch     0.002 0.007              0.004          4.831
## Fare      0.032 0.049              0.038         43.662
## Embarked  0.000 0.009              0.003          6.188
## Title     0.081 0.101              0.088         53.535
## Fsize     0.017 0.009              0.013         11.627
## FsizeD    0.013 0.015              0.014          8.961
## Mother    0.001 0.001              0.001          1.566
```

Task 2

In the second task we are asked to calculate the new weights in the Adaboost algorithm and lastly give second decision stump of the final model. So first we can see that the given weights from the first iteration is

$$w_1 = \begin{bmatrix} 0.072 & 0.072 & 0.071 & 0.071 & 0.071 & 0.167 & 0.167 & 0.071 & 0.167 & 0.071 \end{bmatrix}^T.$$

Based on the figure in the specification we see that the model misclassifies the blue data points and we track the misclassified cases in the vector below as ones and correctly classified cases as zero.

$$\mathbf{1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Now we can calculate the errors for this current classifier by

$$\epsilon_{(t)} = \sum_{i=1}^{N} w_1^{(t)} \mathbf{1}_{f^{(t)}(\mathbf{x}_i) \neq y_i} = 0.355$$

and subsequently we can calculate the adjustment factor $\theta_{(t)}$ as

$$\theta_{(t)} = \sqrt{\frac{\epsilon_{(t)}}{1 - \epsilon_{(t)}}} = 0.742 < 1.$$

Lastly, we can perform the update step to obtain the new vector of weight by using

$$w_i^{(t+1)} = w_i^{(t)} \theta_{(t)}^{1 - |f^{(t)}(\mathbf{x}_i) - y_i|}$$

which yields,

$$w^{(t+1)} = \begin{bmatrix} 0.0558, 0.0558, 0.1, 0.1, 0.1, 0.1295, 0.1295, 0.1, 0.1295, 0.1 \end{bmatrix}$$

Now, we can get the weight of the second decision stump in the final model which is

$$\log\left(\frac{1}{\theta_2}\right) = \log\left(\frac{1}{0.742}\right) = 0.293.$$