# Titanic

```
library(titanic)    # loads titanic_train data frame
library(caret)
```

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

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

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.2     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(rpart)
# 3 significant digits
options(digits = 3)
# Clean the data
titanic_clean <- titanic_train %>%
  mutate(Survived = factor(Survived),
         Embarked = factor(Embarked),
         Age = ifelse(is.na(Age), median(Age, na.rm = TRUE), Age), # NA age to median age
         FamilySize = SibSp + Parch + 1) %>%    # count family members
  select(Survived,  Sex, Pclass, Age, Fare, SibSp, Parch, FamilySize, Embarked)
```

1. Training and Test sets: Use the caret package to create a 20% data partition based on the Survived column. Assign the 20% partition to test_set and the remaining 80% partition to train_set.

```
test_index <- createDataPartition(titanic_clean$Survived, times = 1, p = 0.2, list = FALSE)
test <- titanic_clean[test_index,]
train <- titanic_clean[-test_index,]
```

2. Baseline Prediction by Guessing the Outcome: The simplest prediction method is randomly guessing the outcome without using additional predictors. These methods will help determine whether the machine learning algorithm performs better than chance.

```r
guess_pred <- sample(c(0, 1), size = nrow(test), replace = TRUE)
#accuracy of this guessing method
mean(guess_pred == test$Survived)
```

```
## [1] 0.475
```

3. Predicting Survival by Sex

```r
library(broom)
#a. Use the training set to determine whether members of a given sex were more likely to survive or die
train %>%
  group_by(Sex) %>%
  summarize(Survived = mean(Survived == 1))
```

```
## # A tibble: 2 x 2
##   Sex    Survived
##   <chr>     <dbl>
## 1 female    0.729
## 2 male      0.190
```

```r
#b. Predict survival using sex on the test set: if the survival rate for a sex is over 0.5, predict sur
sex_model <- train %>%
  group_by(Sex) %>%
  summarize(Survived_predict = ifelse(mean(Survived == 1) > 0.5, 1, 0))
test_set1 <- test %>%
  inner_join(sex_model, by = 'Sex')
cm1 <- confusionMatrix(data = factor(test_set1$Survived_predict), reference = factor(test_set1$Survived
cm1 %>% tidy() %>% filter(term == "accuracy")
```

```
## # A tibble: 1 x 6
##   term     class estimate conf.low conf.high      p.value
##   <chr>    <chr>    <dbl>    <dbl>     <dbl>        <dbl>
## 1 accuracy <NA>     0.810    0.745     0.865 0.0000000135
```

4. Predicting survival by Passenger Class

```r
pclass_model <- train %>%
  group_by(Pclass) %>%
  summarize(Survived_predict = ifelse(mean(Survived == 1) > 0.5, 1, 0))
test_set2 <- test %>%
  inner_join(pclass_model, by = 'Pclass')
cm2 <- confusionMatrix(data = factor(test_set2$Survived_predict), reference = factor(test_set2$Survived
cm2 %>% tidy() %>%
  filter(term == 'accuracy')
```

```
## # A tibble: 1 x 6
##   term     class estimate conf.low conf.high p.value
##   <chr>    <chr>    <dbl>    <dbl>     <dbl>   <dbl>
## 1 accuracy <NA>     0.665    0.591     0.733  0.0951
```

Use the training set to group passengers by both sex and passenger class. Which sex and class combinations were more likely to survive than die (i.e. >50% survival)?

```
combined_model <- train %>%
  group_by(Pclass, Sex) %>%
  summarize(Survived_predict = ifelse(mean(Survived == 1) > 0.5, 1, 0))
```

```
## `summarise()` has grouped output by 'Pclass'. You can override using the `.groups` argument.
```

```
test_set3 <- test %>%
  inner_join(combined_model, by = c('Pclass', 'Sex'))
cm3 <- confusionMatrix(data = factor(test_set3$Survived_predict), reference = factor(test_set3$Survived
cm3 %>% tidy() %>%
  filter(term == 'accuracy')
```

```
## # A tibble: 1 x 6
##   term     class estimate conf.low conf.high   p.value
##   <chr>    <chr>    <dbl>    <dbl>     <dbl>     <dbl>
## 1 accuracy <NA>     0.760    0.690     0.820 0.0000270
```

5. Confusion Matrix: create confusion matrices for the combined sex and class model and inspect sensitivity, specificity and balanced accuracy.

```
cm3 %>% tidy() %>%
  filter(term == 'sensitivity' | term == 'specificity' | term == 'balanced_accuracy')
```

```
## # A tibble: 3 x 6
##   term              class estimate conf.low conf.high p.value
##   <chr>             <chr>    <dbl>    <dbl>     <dbl>   <dbl>
## 1 sensitivity       0        1           NA        NA      NA
## 2 specificity       0        0.377       NA        NA      NA
## 3 balanced_accuracy 0        0.688       NA        NA      NA
```

6. Calculate scores for the sex model, class model, and combined sex and class model.

```
F_meas(data = factor(test_set1$Survived_predict), reference = factor(test_set1$Survived))
```

```
## [1] 0.852
```

```
F_meas(data = factor(test_set2$Survived_predict), reference = factor(test_set2$Survived))
```

```
## [1] 0.756
```

```
F_meas(data = factor(test_set3$Survived_predict), reference = factor(test_set3$Survived))
```

```
## [1] 0.837
```

7. Survival by Fare - LDA and QDA

```r
#Train a model using linear discriminant analysis (LDA)
fit_lda <- train(Survived ~ Fare, data=train, method = 'lda')
survived_hat <- predict(fit_lda, test)
#accuracy
mean(survived_hat == test$Survived)
```

```
## [1] 0.682
```

```r
#qda model
fit_qda <- train(Survived ~ Fare, data=train, method = 'qda')
survived_hat <- predict(fit_qda, test)
#accuracy
mean(survived_hat == test$Survived)
```

```
## [1] 0.67
```

8. Logistic Regression Models

```r
#Train a logistic regression model using age as the only predictor
fit_sex <- glm(Survived ~ Sex, data=train, family = binomial)
survived_hat <- predict(fit_sex, test)
survived_hat <- ifelse(survived_hat >= 0, 1, 0)
mean(survived_hat == test$Survived)
```

```
## [1] 0.81
```

```r
#Train a logistic regression model using all predictors
fit_all <- glm(Survived ~ ., data=train, family = binomial)
survived_hat <- predict(fit_all, test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
survived_hat <- ifelse(survived_hat >= 0.5, 1, 0)
mean(survived_hat == test$Survived)
```

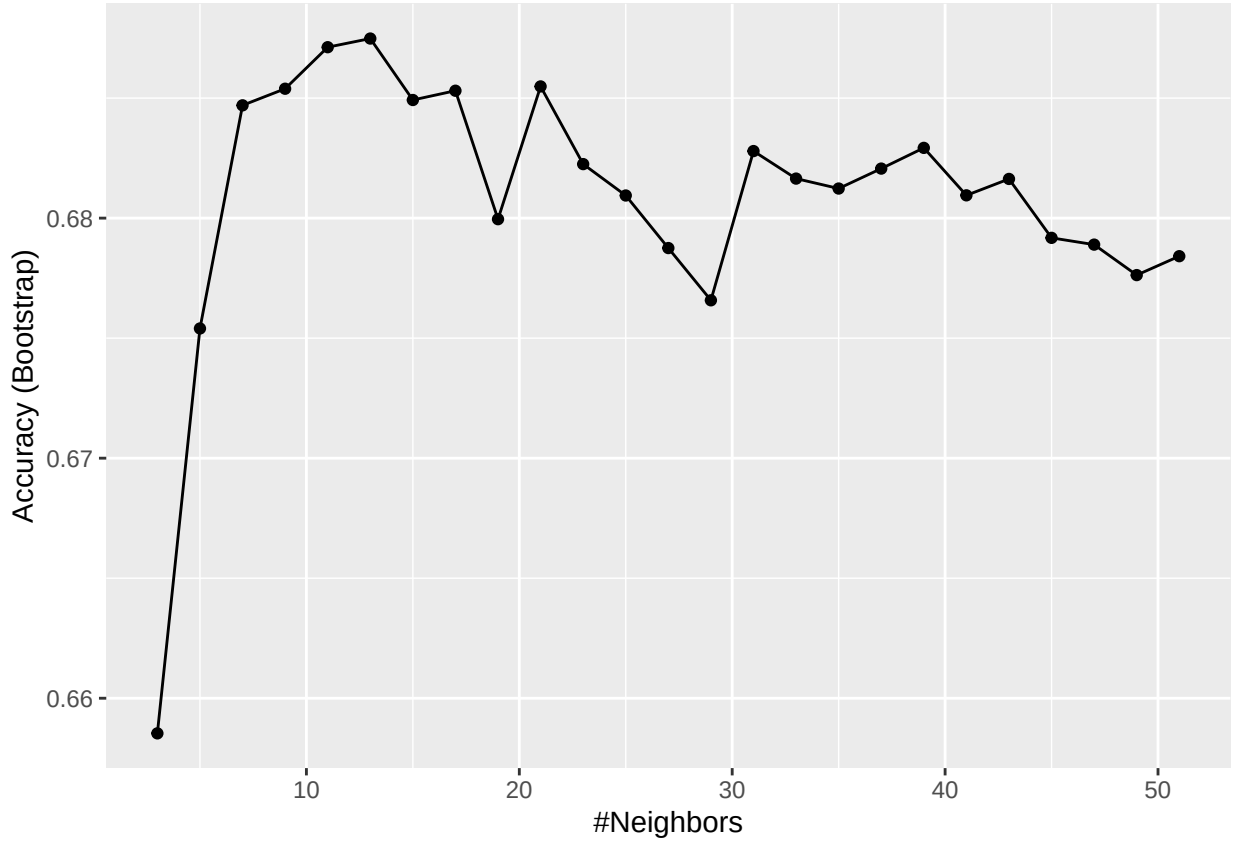
```
## [1] 0.816
```

9.kNN model

```r
#tuning model with k = seq(3, 51, 2)
k <- seq(3, 51, 2)
fit_knn <- train(Survived ~ ., data = train, method = "knn", tuneGrid = data.frame(k))
fit_knn$bestTune
```

```
##    k
## 6 13
```

```r
#plotting k values
ggplot(fit_knn)
```




```r
#of 7, 11, 17 and 21, which yields the highest accuracy
fit_knn$results %>% filter(k %in% c(7, 11, 17, 21)) %>% pull(Accuracy)
```

```
## [1] 0.685 0.687 0.685 0.685
```

```r
#accuracy
survived_hat <- predict(fit_knn, test)
mean(survived_hat == test$Survived)
```

```
## [1] 0.715
```

10. Cross-Validation: Instead of the default training control, use 10-fold cross-validation where each partition consists of 10% of the total.

```r
#tuning model with k = seq(3, 51, 2)
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn_cv <- train(Survived ~ ., method = "knn",
                      data = train,
                      tuneGrid = data.frame(k = seq(3, 51, 2)),
                      trControl = control)
#optimal value of k
train_knn_cv$bestTune
```

```
##   k
## 3 7
```

```
#accuracy of test set
survived_hat <- predict(train_knn_cv, test)
mean(survived_hat == test$Survived)
```

```
## [1] 0.737
```

11. Classification Tree Model

a) Tune the complexity parameter with cp = seq(0, 0.05, 0.002)

```
fit_tree <- train(Survived ~ ., method = "rpart",
                  data = train,
                  tuneGrid = data.frame(cp = seq(0, 0.05, 0.002)))
fit_tree$bestTune
```

```
##       cp
## 10 0.018
```

```
fit_tree$results
```

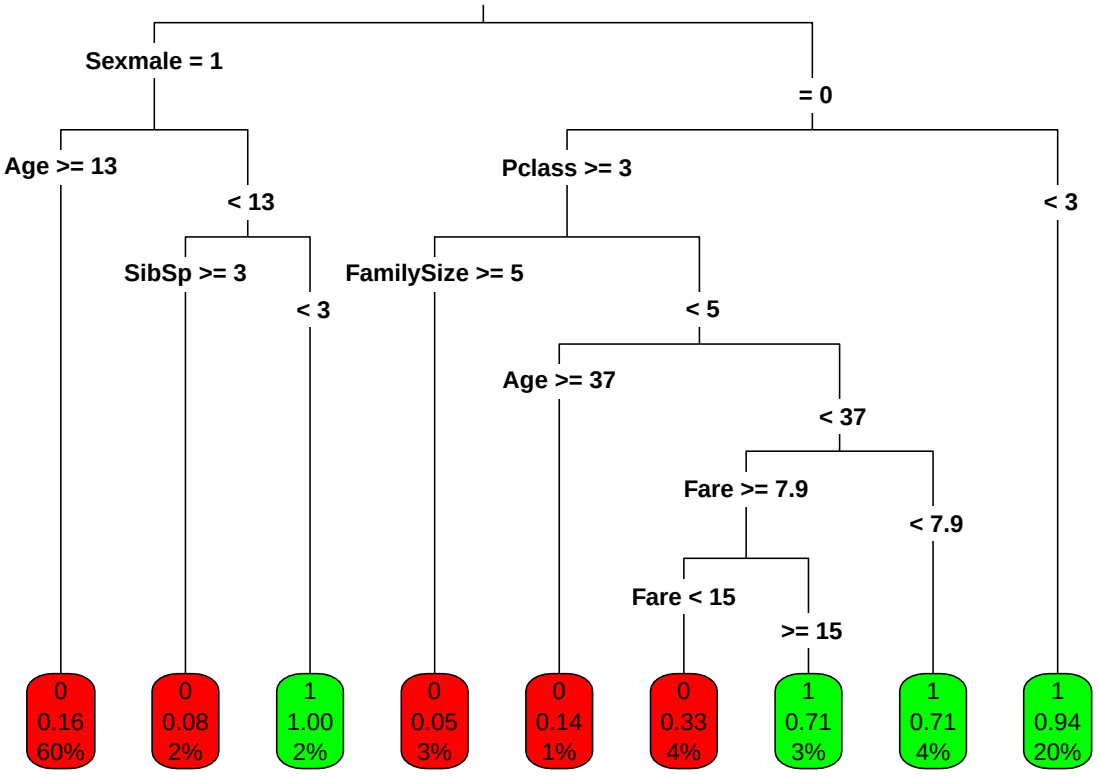
```
##       cp Accuracy Kappa AccuracySD KappaSD
## 1  0.000    0.779 0.528     0.0258  0.0518
## 2  0.002    0.788 0.545     0.0266  0.0544
## 3  0.004    0.793 0.554     0.0293  0.0591
## 4  0.006    0.796 0.559     0.0266  0.0558
## 5  0.008    0.798 0.563     0.0250  0.0522
## 6  0.010    0.803 0.570     0.0250  0.0531
## 7  0.012    0.805 0.569     0.0189  0.0411
## 8  0.014    0.805 0.569     0.0177  0.0381
## 9  0.016    0.805 0.568     0.0177  0.0386
## 10 0.018    0.807 0.572     0.0158  0.0381
## 11 0.020    0.806 0.567     0.0163  0.0374
## 12 0.022    0.806 0.566     0.0162  0.0372
## 13 0.024    0.805 0.565     0.0153  0.0369
## 14 0.026    0.802 0.561     0.0184  0.0440
## 15 0.028    0.798 0.553     0.0217  0.0495
## 16 0.030    0.796 0.547     0.0181  0.0418
## 17 0.032    0.794 0.543     0.0190  0.0415
## 18 0.034    0.792 0.540     0.0191  0.0411
## 19 0.036    0.789 0.534     0.0214  0.0446
## 20 0.038    0.789 0.532     0.0203  0.0434
## 21 0.040    0.787 0.530     0.0191  0.0409
## 22 0.042    0.787 0.530     0.0191  0.0409
## 23 0.044    0.787 0.530     0.0192  0.0403
## 24 0.046    0.787 0.531     0.0188  0.0407
## 25 0.048    0.787 0.531     0.0188  0.0407
## 26 0.050    0.786 0.529     0.0194  0.0411
```

```
#accuracy with test set
survived_hat <- predict(fit_tree, test)
mean(survived_hat == test$Survived)
```

```
## [1] 0.832
```

b. Inspect the final model and plot the decision tree.

```
library(rpart.plot)
rpart.plot(fit_tree$finalModel, type = 3, box.palette = c("red", "green"), fallen.leaves = TRUE)
```




12. Random Forest Model

```
#Test values of mtry = seq(1:7)
#Set ntree to 100
fit_rf <- train(Survived ~ ., data = train, method = "rf",
                tuneGrid = data.frame(mtry = seq(1:7)), ntree = 100)
fit_rf$bestTune
```

```
##   mtry
## 2    2
```

```r
#accuracy
survived_hat <- predict(fit_rf, test)
confusionMatrix(data = factor(survived_hat), reference = factor(test$Survived))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 108  26
##          1   2  43
##
##                Accuracy : 0.844
##                  95% CI : (0.782, 0.893)
##     No Information Rate : 0.615
##     P-Value [Acc > NIR] : 1.79e-11
##
##                   Kappa : 0.647
##
##  Mcnemar's Test P-Value : 1.38e-05
##
##             Sensitivity : 0.982
##             Specificity : 0.623
##          Pos Pred Value : 0.806
##          Neg Pred Value : 0.956
##              Prevalence : 0.615
##          Detection Rate : 0.603
##    Detection Prevalence : 0.749
##       Balanced Accuracy : 0.803
##
##        'Positive' Class : 0
##
```

```r
mean(survived_hat == test$Survived)
```

```
## [1] 0.844
```

```r
#determine the importance of various predictors to the random forest model
varImp(fit_rf$finalModel)
```

```
##            Overall
## Sexmale      66.38
## Pclass       24.64
## Age          29.74
## Fare         37.04
## SibSp         8.71
## Parch         6.99
## FamilySize   16.77
## EmbarkedC     3.64
## EmbarkedQ     1.77
## EmbarkedS     3.62
```