CYO Machine Learning Project: Titanic - Machine Learning from Disaster

Eddwin Cheteni

2021-03-19

Overview

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set will be used to build machine learning models. For the training set, the outcomes are provided (also known as the "ground truth") for each passenger. The model will be based on "features" like passengers' gender and class. The test set will be used to see how well our model performs on unseen data. For the test set, the ground truth is not provided for each passenger. The objective of this project is to predict these outcomes. For each passenger in the test set, we use the trained model to predict whether or not they survived the sinking of the Titanic.

The datasets also include gender_submission.csv, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

Loading data

```
#loading train set
train_set <- read_csv("~/Data Science Projects/train.csv")
#converting the column names into lowercase
names(train_set) <- tolower(names(train_set))

#loading test set
test_set <- read_csv("~/Data Science Projects/test.csv")
#converting the column names into lowercase
names(test_set) <- tolower(names(test_set))
```

Data Dictionary

```
## [1] "Dimensions of the train set are:"

## [1] 891  12

## [1] "Dimensions of the test set are:"

## [1] 418  11
```

This confirms that the train set has 891 rows and 12 columns while the test set has 418 rows and 11 columns.

```
## # A tibble: 6 x 12
##   passengerid survived pclass name
sex      age sibsp parch ticket            fare cabin embarked
##         <dbl>    <dbl>  <dbl> <chr>
<chr>  <dbl> <dbl> <dbl> <chr>            <dbl> <chr> <chr>
## 1           1        0      3 Braund, Mr. Owen Harris
male      22     1     0 A/5 21171         7.25 <NA>  S
## 2           2        1      1 Cumings, Mrs. John Bradley (Florence Briggs Thayer)
female    38     1     0 PC 17599         71.3  C85   C
## 3           3        1      3 Heikkinen, Miss. Laina
female    26     0     0 STON/O2. 3101282  7.92 <NA>  S
## 4           4        1      1 Futrelle, Mrs. Jacques Heath (Lily May Peel)
female    35     1     0 113803           53.1  C123  S
## 5           5        0      3 Allen, Mr. William Henry
male      35     0     0 373450            8.05 <NA>  S
## 6           6        0      3 Moran, Mr. James
male      NA     0     0 330877            8.46 <NA>  Q
```

```
## # A tibble: 6 x 11
##   passengerid pclass name                                        sex      age
sibsp parch ticket   fare cabin embarked
##         <dbl>  <dbl> <chr>                                       <chr>  <dbl>
<dbl> <dbl> <chr>   <dbl> <chr> <chr>
## 1         892      3 Kelly, Mr. James                            male    34.5
0     0 330911   7.83 <NA>  Q
## 2         893      3 Wilkes, Mrs. James (Ellen Needs)            female  47
1     0 363272   7    <NA>  S
## 3         894      2 Myles, Mr. Thomas Francis                   male    62
0     0 240276   9.69 <NA>  Q
## 4         895      3 Wirz, Mr. Albert                            male    27
0     0 315154   8.66 <NA>  S
## 5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female  22
1     1 3101298 12.3  <NA>  S
## 6         897      3 Svensson, Mr. Johan Cervin                  male    14
0     0 7538     9.22 <NA>  S
```

Data Exploratory Analysis

Visualize train_set

We need to explore on our data so as to get read of unnecessary features and some anomalies in the data such as missing values and outliers.

```
## spec_tbl_df [891 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ passengerid: num [1:891] 1 2 3 4 5 6 7 8 9 10 ...
##  $ survived   : num [1:891] 0 1 1 1 0 0 0 0 1 1 ...
##  $ pclass     : num [1:891] 3 1 3 1 3 3 1 3 3 2 ...
##  $ name       : chr [1:891] "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley
(Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath
(Lily May Peel)" ...
##  $ sex        : chr [1:891] "male" "female" "female" "female" ...
##  $ age        : num [1:891] 22 38 26 35 35 NA 54 2 27 14 ...
##  $ sibsp      : num [1:891] 1 1 0 1 0 0 0 3 0 1 ...
##  $ parch      : num [1:891] 0 0 0 0 0 0 0 1 2 0 ...
##  $ ticket     : chr [1:891] "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
```

```
##  $ fare       : num [1:891] 7.25 71.28 7.92 53.1 8.05 ...
##  $ cabin      : chr [1:891] NA "C85" NA "C123" ...
##  $ embarked   : chr [1:891] "S" "C" "S" "S" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   PassengerId = col_double(),
##   ..   Survived = col_double(),
##   ..   Pclass = col_double(),
##   ..   Name = col_character(),
##   ..   Sex = col_character(),
##   ..   Age = col_double(),
##   ..   SibSp = col_double(),
##   ..   Parch = col_double(),
##   ..   Ticket = col_character(),
##   ..   Fare = col_double(),
##   ..   Cabin = col_character(),
##   ..   Embarked = col_character()
##   .. )
```

Lets have a look at the summary of our train set:

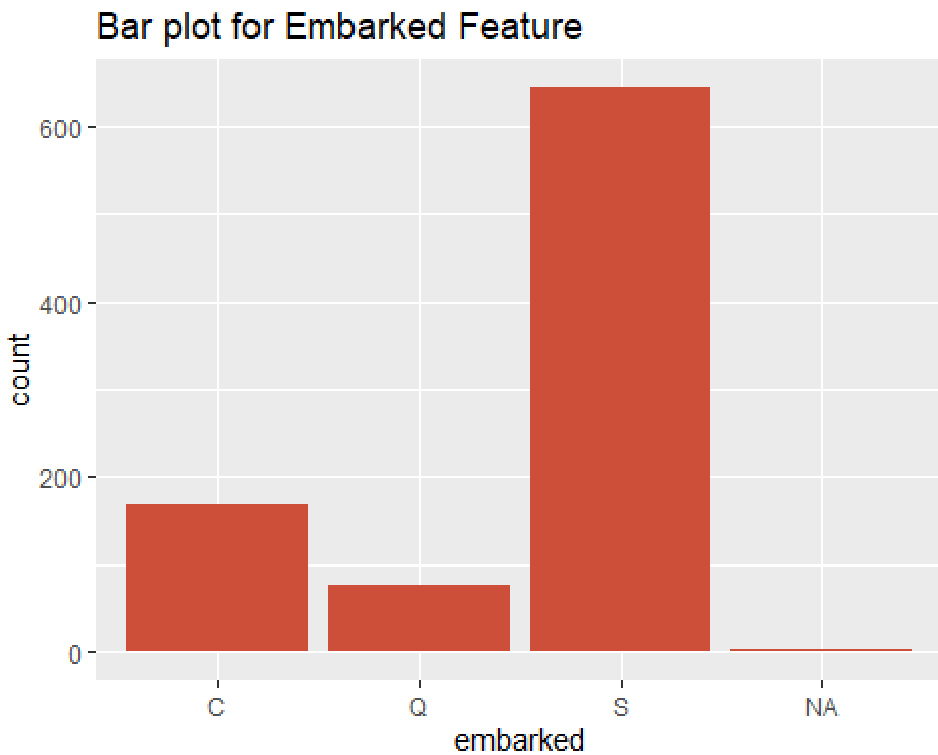
```
##   passengerid     survived         pclass         name                sex
age            sibsp           parch          ticket               fare
##  Min.   :  1   Min.   :0.000   Min.   :1.00   Length:891          Length:891
Min.   : 0.42   Min.   :0.000   Min.   :0.000   Length:891         Min.   :  0.00
##  1st Qu.:224   1st Qu.:0.000   1st Qu.:2.00   Class :character   Class :character
1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.000   Class :character   1st Qu.:  7.91
##  Median :446   Median :0.000   Median :3.00   Mode  :character   Mode  :character
Median :28.00   Median :0.000   Median :0.000   Mode  :character   Median : 14.45
##  Mean   :446   Mean   :0.384   Mean   :2.31
Mean   :29.70   Mean   :0.523   Mean   :0.382                      Mean   : 32.20
##  3rd Qu.:668   3rd Qu.:1.000   3rd Qu.:3.00
3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.000                      3rd Qu.: 31.00
##  Max.   :891   Max.   :1.000   Max.   :3.00
Max.   :80.00   Max.   :8.000   Max.   :6.000                      Max.   :512.33
##
NA's   :177
##     cabin              embarked
##  Length:891         Length:891
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

It indicates that age has 177 missing values.

```
#checking variables with missing values
list_na <- colnames(train_set)[ apply(train_set, 2, anyNA) ]
list_na

## [1] "age"      "cabin"    "embarked"
```

```
#checking missing values by use of a barplot on embarked feature
train_set%>%
        select(embarked)%>% #select embarked column
        ggplot()+
        geom_bar(aes(embarked), stat = "count", na.rm=TRUE, position = "stack", fill
=I("tomato3"))+
        ggtitle("Bar plot for Embarked Feature") #title of the chart
```



Based on the data source provided, there is 20% missing values for age, 77% for cabin and almost 0% for embarked. From this observation, we can drop the cabin column, replace with median on the age column and mode in the embarked column.

```
#since Southampton has the most passengers
embarked_mode<-"S"
#computing the median age
age_median<-median(train_set$age,na.rm = TRUE)
#replacing the missing values, age by median and embarked by the most frequent one
i.e. Southampton
train_set_replaced <- train_set %>%
            mutate(age  = ifelse(is.na(age), age_median, age),embarked  =
ifelse(is.na(embarked),embarked_mode, embarked))
head(train_set_replaced)

## # A tibble: 6 x 12
##   passengerid survived pclass name
sex      age sibsp parch ticket            fare cabin embarked
##         <dbl>    <dbl>  <dbl> <chr>
<chr>  <dbl> <dbl> <dbl> <chr>            <dbl> <chr> <chr>
## 1           1        0      3 Braund, Mr. Owen Harris
```

```
male       22     1     0 A/5 21171         7.25 <NA>  S
## 2           2      1       1 Cumings, Mrs. John Bradley (Florence Briggs Thayer)
female     38     1     0 PC 17599         71.3  C85   C
## 3           3      1       3 Heikkinen, Miss. Laina
female     26     0     0 STON/O2. 3101282  7.92 <NA>  S
## 4           4      1       1 Futrelle, Mrs. Jacques Heath (Lily May Peel)
female     35     1     0 113803           53.1  C123  S
## 5           5      0       3 Allen, Mr. William Henry
male       35     0     0 373450            8.05 <NA>  S
## 6           6      0       3 Moran, Mr. James
male       28     0     0 330877            8.46 <NA>  Q
```

Dropping unncessary columns

The variable column Cabin is dropped since 77% of the observations are missing therefore, including this in our model might compromise our model performance.

```
colnames(train_set_replaced)[apply(train_set_replaced, 2, anyNA)]

## [1] "cabin"

# Data pre-processing on train set
df<- train_set_replaced %>%
   mutate(survived = as.factor(survived),#convert to a factor
          pclass = as.factor(pclass),#convert to a factor
          sex = as.factor(sex),#convert to a factor
          embarked = as.factor(embarked),#convert to a factor
          fare = log1p(fare),#log-transform fare variable
          nbr_family = as.integer(sibsp + parch),#combine siblings and parents
together
          age = scale(age)) %>%#convert to an integer
   select(survived, pclass, sex, age, nbr_family, fare, embarked)#selecting the
required columns
# Add cleaner factor levels
levels(df$survived) <- c('Perished','Survived')
levels(df$embarked) <- c('Cherbourg','Queenstown','Southampton')
levels(df$sex) <- c('Female','Male')
df%>%head()

## # A tibble: 6 x 7
##   survived pclass sex    age[,1] nbr_family  fare embarked
##   <fct>    <fct>  <fct>    <dbl>      <int> <dbl> <fct>
## 1 Perished 3      Male    -0.565          1  2.11 Southampton
## 2 Survived 1      Female   0.663          1  4.28 Cherbourg
## 3 Survived 3      Female  -0.258          0  2.19 Southampton
## 4 Survived 1      Female   0.433          1  3.99 Southampton
## 5 Perished 3      Male     0.433          0  2.20 Southampton
## 6 Perished 3      Male    -0.105          0  2.25 Queenstown

df%>%str()

## tibble [891 x 7] (S3: tbl_df/tbl/data.frame)
##  $ survived  : Factor w/ 2 levels "Perished","Survived": 1 2 2 2 1 1 1 1 2 2 ...
##  $ pclass    : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ sex       : Factor w/ 2 levels "Female","Male": 2 1 1 1 2 2 2 2 1 1 ...
```

```
##  $ age       : num [1:891, 1] -0.565 0.663 -0.258 0.433 0.433 ...
##   ..- attr(*, "scaled:center")= num 29.4
##   ..- attr(*, "scaled:scale")= num 13
##  $ nbr_family: int [1:891] 1 1 0 1 0 0 0 4 2 1 ...
##  $ fare      : num [1:891] 2.11 4.28 2.19 3.99 2.2 ...
##  $ embarked  : Factor w/ 3 levels "Cherbourg","Queenstown",..: 3 1 3 3 3 2 3 3 3 1
...
```

Exploring the test set:

```
#checking missing values on test set
colnames(test_set)[ apply(test_set, 2, anyNA) ]

## [1] "age"   "fare"  "cabin"

#computing the median age
age_median<-median(test_set$age,na.rm = TRUE)
#computing the median fare
fare_median<-median(test_set$fare,na.rm = TRUE)
#replacing the missing values in age and fare column so that validation can be
performed
test_set_replaced <- test_set %>%
            mutate(age  = ifelse(is.na(age), age_median, age),fare  =
ifelse(is.na(fare),fare_median, fare))
head(test_set_replaced)

## # A tibble: 6 x 11
##   passengerid pclass name                                     sex      age
sibsp parch ticket   fare cabin embarked
##         <dbl>  <dbl> <chr>                                    <chr>  <dbl>
<dbl> <dbl> <chr>   <dbl> <chr> <chr>
## 1         892      3 Kelly, Mr. James                         male    34.5
0     0 330911   7.83 <NA>  Q
## 2         893      3 Wilkes, Mrs. James (Ellen Needs)         female  47
1     0 363272   7    <NA>  S
## 3         894      2 Myles, Mr. Thomas Francis                male    62
0     0 240276   9.69 <NA>  Q
## 4         895      3 Wirz, Mr. Albert                         male    27
0     0 315154   8.66 <NA>  S
## 5         896      3 Hirvonen, Mrs. Alexander (Helga E Lindqvist) female  22
1     1 3101298 12.3  <NA>  S
## 6         897      3 Svensson, Mr. Johan Cervin               male    14
0     0 7538     9.22 <NA>  S

#checking missing values on test set
colnames(test_set_replaced)[ apply(test_set_replaced, 2, anyNA) ]

## [1] "cabin"
```

Let us create the validation set from the test set:

```
# Data pre-processing on test set to create the validation set
validation<- test_set_replaced %>%
   mutate(pclass = as.factor(pclass),#convert to a factor
          sex = as.factor(sex),#convert to a factor
          embarked = as.factor(embarked),#convert to a factor
```

```
          fare = log1p(fare),#log-transform variable fare
          nbr_family = as.integer(sibsp + parch),#combine siblings and parents
together
          age = scale(age)) %>% #Normalize age
   select(pclass, sex, age, nbr_family, fare, embarked)#selecting important features
# Add cleaner factor levels
levels(validation$embarked) <- c('Cherbourg','Queenstown','Southampton')
levels(validation$sex) <- c('Female','Male')
validation%>%head()

## # A tibble: 6 x 6
##   pclass sex    age[,1] nbr_family  fare embarked
##   <fct>  <fct>    <dbl>      <int> <dbl> <fct>
## 1 3      Male     0.386          0  2.18 Queenstown
## 2 3      Female   1.37           1  2.08 Southampton
## 3 2      Male     2.55           0  2.37 Queenstown
## 4 3      Male    -0.205          0  2.27 Southampton
## 5 3      Female  -0.598          2  2.59 Southampton
## 6 3      Male    -1.23           0  2.32 Southampton

#########################################################
# Pre-Processing
#########################################################

#Create dummy variables on the categorical features in df set
df <- dummy_cols(df,
                        select_columns=c('pclass','sex', 'embarked'),
                       remove_most_frequent_dummy = TRUE) %>%
   select(-pclass,-sex,-embarked)
#Create dummy variables on the categorical features in validation set
validation <- dummy_cols(validation,
                   select_columns=c('pclass','sex', 'embarked'),
                   remove_most_frequent_dummy = TRUE) %>%
   select(-pclass,-sex,-embarked)
```

The tables below shows the datasets (df and validation set) ready for model training process after dummies were created in the above procedure.

```
#Have a look at the df set ready for splitting
df%>%head()%>%knitr::kable()
```

survived	age	nbr_family	fare	pclass_1	pclass_2	sex_Female	embarked_Cherbourg	embarked_Queenstown
Perished	-0.56542	1	2.1102	0	0	0	0	0
Survived	0.66349	1	4.2806	1	0	1	1	0
Survived	-0.25819	0	2.1889	0	0	1	0	0
Survived	0.43307	1	3.9908	1	0	1	0	0
Perished	0.43307	0	2.2028	0	0	0	0	0
Perished	-0.10458	0	2.2469	0	0	0	0	1

```r
#Have a look at the validation set ready for final model validation
validation%>%head()%>%knitr::kable()
```

age	nbr_family	fare	pclass_1	pclass_2	sex_Female	embarked_Cherbourg	embarked_Queenstown
0.38577	0	2.1781	0	0	0	0	1
1.36973	1	2.0794	0	0	1	0	0
2.55048	0	2.3691	0	1	0	0	1
-0.20461	0	2.2683	0	0	0	0	0
-0.59819	2	2.5868	0	0	1	0	0
-1.22793	0	2.3248	0	0	0	0	0

Data partition

We create this partition on the df set so that the validation set will be used for validation of the model.

```
## [1] "Dimensions of the train set are:"

## [1] 714   9

## [1] "Dimensions of the test set are:"

## [1] 177   9
```

Model building

Model 1: Logistic Regression Model (glm)

```r
#############################################################
# Model 1: Logistic Regression Model ('glm') all selected variables
#############################################################

# Fit model with all variables
(fit_glm <- glm(survived ~ .,
               data=train,
               family=binomial))

##
## Call:  glm(formula = survived ~ ., family = binomial, data = train)
##
## Coefficients:
##         (Intercept)                  age           nbr_family                 fare
pclass_1             pclass_2           sex_Female     embarked_Cherbourg
##              -3.303               -0.519               -0.288                0.405
1.690                1.107                2.749                0.338
## embarked_Queenstown
##               0.184
##
```

```
## Degrees of Freedom: 713 Total (i.e. Null);  705 Residual
## Null Deviance:       951
## Residual Deviance: 617   AIC: 635
```

```
# Generate prediction
p_hat_glm <- predict(fit_glm, test, type='response')
y_hat_glm <- ifelse(p_hat_glm > 0.5, "Survived", "Perished") %>% factor()
# Compute the accuracy
acc <- confusionMatrix(y_hat_glm,test$survived, positive='Survived')
#Generate table of accuracy
accuracy_results <- tibble(method='Model 1: Logistic Regression Model',
                           accuracy = acc$overall['Accuracy'])
accuracy_results%>%knitr::kable()
```

method	accuracy
Model 1: Logistic Regression Model	0.78531

Model 2: Random Forest Model (rf)

```
##########################################################
# Model 2: Random Forest Model ('rf')
##########################################################

set.seed(1234)

# Set control parameters
control <- trainControl(method='repeatedcv',
                        number=10,
                        repeats=5,
                        search = 'random')

# Determine baseline mtry
mtry <- sqrt(ncol(train))
tunegrid = expand.grid(.mtry=mtry)

#Train RF model
#Random generate 15 mtry values with tuneLength = 15
train_rf <- train(survived ~ .,
                  data=train,
                  method='rf',
                  tuneLength=15,
                  trControl=control,
                  importance=TRUE,
                  localImp=TRUE)

# Explain final RF model
(fit_rf <- train_rf$finalModel)

##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE,      localImp =
TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 16.67%
## Confusion matrix:
##          Perished Survived class.error
## Perished      388       52     0.11818
## Survived       67      207     0.24453
```

```
#Generate predictions
y_hat_rf <- predict(fit_rf, test)
#Compute the accuracy
acc <- confusionMatrix(y_hat_rf,test$survived, positive='Survived')
acc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Perished Survived
##   Perished       94       20
##   Survived       15       48
##
##                Accuracy : 0.802
##                  95% CI : (0.736, 0.858)
##     No Information Rate : 0.616
##     P-Value [Acc > NIR] : 7.43e-08
##
##                   Kappa : 0.576
##
##  Mcnemar's Test P-Value : 0.499
##
##             Sensitivity : 0.706
##             Specificity : 0.862
##          Pos Pred Value : 0.762
##          Neg Pred Value : 0.825
##              Prevalence : 0.384
##          Detection Rate : 0.271
##    Detection Prevalence : 0.356
##       Balanced Accuracy : 0.784
##
##        'Positive' Class : Survived
##
```

```
#Generate table for results
accuracy_results <- accuracy_results %>%
   bind_rows(tibble(method='Model 2: Random Forest Model',
                    accuracy = acc$overall['Accuracy']))
accuracy_results %>% knitr::kable()
```

method	accuracy
Model 1: Logistic Regression Model	0.78531
Model 2: Random Forest Model	0.80226

Model 3: Gradient Boosting Model (gbm)

```
###########################################################
# Model 3: Gradient Boosting Model ('gbm')
###########################################################

set.seed(1234)

# Set control parameters
control <- trainControl(method='repeatedcv',
                        number=4,
                        repeats=4)

#Train gbm model
fit_gbm <- train(survived ~ .,
                  data=train,
                  method='gbm',
                  trControl=control,
                  verbose = FALSE)
fit_gbm

## Stochastic Gradient Boosting
##
## 714 samples
##   8 predictor
##   2 classes: 'Perished', 'Survived'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold, repeated 4 times)
## Summary of sample sizes: 536, 536, 535, 535, 535, 536, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa
##   1                   50      0.80917   0.58808
##   1                  100      0.81234   0.59836
##   1                  150      0.80955   0.59232
##   2                   50      0.81619   0.60545
##   2                  100      0.81514   0.60403
##   2                  150      0.81619   0.60663
##   3                   50      0.81934   0.61134
##   3                  100      0.82423   0.62067
##   3                  150      0.82353   0.61979
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 100, interaction.depth = 3,
shrinkage = 0.1 and n.minobsinnode = 10.

#Generate predictions
y_hat_gbm <- predict(fit_gbm, test, type = "raw")

#Compute the accuracy
```

```
acc <- confusionMatrix(y_hat_gbm,test$survived, positive='Survived')
acc

## Confusion Matrix and Statistics
##
##           Reference
## Prediction Perished Survived
##   Perished       96       24
##   Survived       13       44
##
##                Accuracy : 0.791
##                  95% CI : (0.724, 0.848)
##     No Information Rate : 0.616
##     P-Value [Acc > NIR] : 4.55e-07
##
##                   Kappa : 0.544
##
##  Mcnemar's Test P-Value : 0.1
##
##             Sensitivity : 0.647
##             Specificity : 0.881
##          Pos Pred Value : 0.772
##          Neg Pred Value : 0.800
##              Prevalence : 0.384
##          Detection Rate : 0.249
##    Detection Prevalence : 0.322
##       Balanced Accuracy : 0.764
##
##        'Positive' Class : Survived
##

#Generate table for results
accuracy_results <- accuracy_results %>%
   bind_rows(tibble(method='Model 3: Gradient Boosting Model',
                    accuracy = acc$overall['Accuracy']))
accuracy_results %>% knitr::kable()
```

method	accuracy
Model 1: Logistic Regression Model	0.78531
Model 2: Random Forest Model	0.80226
Model 3: Gradient Boosting Model	0.79096

Model 4: Support Vector Machine Model (svm)

```
#########################################################
# Model 4: Support Vector Machine Model ('svm')
#########################################################
set.seed(1234)

#setting control parameters
cv_opts = trainControl(method="cv", number=10)

#Train the model
```

```r
results_svm = train(survived~.,
                    data=train,
                    method="svmLinear",
                    preProcess="range",
                    trControl=cv_opts,
                    tuneLength=5)
results_svm

## Support Vector Machines with Linear Kernel
##
## 714 samples
##   8 predictor
##   2 classes: 'Perished', 'Survived'
##
## Pre-processing: re-scaling to [0, 1] (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 643, 643, 642, 643, 643, 643, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.7928    0.5562
##
## Tuning parameter 'C' was held constant at a value of 1

#Generate predictions
y_hat_svm = predict(results_svm, test)
#Compute the accuracy
acc <- confusionMatrix(y_hat_svm, test$survived, positive='Survived')
acc

## Confusion Matrix and Statistics
##
##           Reference
## Prediction Perished Survived
##   Perished       92       25
##   Survived       17       43
##
##                Accuracy : 0.763
##                  95% CI : (0.693, 0.823)
##     No Information Rate : 0.616
##     P-Value [Acc > NIR] : 2.39e-05
##
##                   Kappa : 0.487
##
##  Mcnemar's Test P-Value : 0.28
##
##             Sensitivity : 0.632
##             Specificity : 0.844
##          Pos Pred Value : 0.717
##          Neg Pred Value : 0.786
##              Prevalence : 0.384
##          Detection Rate : 0.243
##    Detection Prevalence : 0.339
##       Balanced Accuracy : 0.738
##
```

```
##        'Positive' Class : Survived
##

#Generate table for results
accuracy_results <- accuracy_results %>%
   bind_rows(tibble(method='Model 4: Support Vector Machine Model',
                    accuracy = acc$overall['Accuracy']))
accuracy_results %>% knitr::kable()
```

method	accuracy
Model 1: Logistic Regression Model	0.78531
Model 2: Random Forest Model	0.80226
Model 3: Gradient Boosting Model	0.79096
Model 4: Support Vector Machine Model	0.76271

Model 5: Neural Network Model (nnet)

```
#########################################################
# Model 5: Neural Network Model ('nnet')
#########################################################
results_nnet = train(survived~.,
                     data=train,
                     method="avNNet",
                     trControl=cv_opts,
                     preProcess="range",
                     tuneLength=5,
                     trace=F,
                     maxit=1000)
results_nnet

## Model Averaged Neural Network
##
## 714 samples
##   8 predictor
##   2 classes: 'Perished', 'Survived'
##
## Pre-processing: re-scaling to [0, 1] (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 643, 643, 643, 643, 642, 642, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy  Kappa
##   1     0e+00  0.80379   0.58087
##   1     1e-04  0.79832   0.56023
##   1     1e-03  0.80115   0.56106
##   1     1e-02  0.80395   0.56580
##   1     1e-01  0.80256   0.56428
##   3     0e+00  0.82349   0.61770
##   3     1e-04  0.82079   0.60906
##   3     1e-03  0.81230   0.59284
##   3     1e-02  0.81939   0.60990
##   3     1e-01  0.80677   0.57740
##   5     0e+00  0.82218   0.61347
```

```
##   5     1e-04  0.82218   0.61836
##   5     1e-03  0.82494   0.62700
##   5     1e-02  0.81379   0.60170
##   5     1e-01  0.80675   0.57662
##   7     0e+00  0.81800   0.60660
##   7     1e-04  0.80266   0.58016
##   7     1e-03  0.80959   0.59288
##   7     1e-02  0.80959   0.59284
##   7     1e-01  0.80816   0.58010
##   9     0e+00  0.82076   0.61236
##   9     1e-04  0.81097   0.59699
##   9     1e-03  0.80117   0.57996
##   9     1e-02  0.80115   0.57520
##   9     1e-01  0.80816   0.58010
##
## Tuning parameter 'bag' was held constant at a value of FALSE
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 5, decay = 0.001 and bag = FALSE.
```

```
#Generate predictions
y_hat_nnet <- predict(results_nnet, test)
#Compute the accuracy
acc <- confusionMatrix(y_hat_nnet,test$survived, positive="Survived")
acc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Perished Survived
##   Perished       97       21
##   Survived       12       47
##
##                Accuracy : 0.814
##                  95% CI : (0.748, 0.868)
##     No Information Rate : 0.616
##     P-Value [Acc > NIR] : 1.06e-08
##
##                   Kappa : 0.596
##
##  Mcnemar's Test P-Value : 0.164
##
##             Sensitivity : 0.691
##             Specificity : 0.890
##          Pos Pred Value : 0.797
##          Neg Pred Value : 0.822
##              Prevalence : 0.384
##          Detection Rate : 0.266
##    Detection Prevalence : 0.333
##       Balanced Accuracy : 0.791
##
##        'Positive' Class : Survived
##
```

```
#Generate table for results
accuracy_results <- accuracy_results %>%
```

```
  bind_rows(tibble(method='Model 5: Neural Network Model',
                   accuracy = acc$overall['Accuracy']))
accuracy_results %>% knitr::kable()
```

method	accuracy
Model 1: Logistic Regression Model	0.78531
Model 2: Random Forest Model	0.80226
Model 3: Gradient Boosting Model	0.79096
Model 4: Support Vector Machine Model	0.76271
Model 5: Neural Network Model	0.81356

Neural Network model gives the best accuracy of 0.81356 on the test set, so we will consider it in predicting survivors in the validation set to be submitted on Kaggle site for evalution.

Final model - Neural network (nnet)

```
########################################################
# Final Model
# Based on modeled results, apply Neural Network Model
# to 'test set' and create submission csv
########################################################
#setting control parameters
cv_opts = trainControl(method="cv", number=10)

#Train nnet model
fit_final <-  train(survived~.,
                    data=train,
                    method="avNNet",
                    trControl=cv_opts,
                    preProcess="range",
                    tuneLength=5,
                    trace=F,
                    maxit=1000)
fit_final

## Model Averaged Neural Network
##
## 714 samples
##   8 predictor
##   2 classes: 'Perished', 'Survived'
##
## Pre-processing: re-scaling to [0, 1] (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 642, 643, 643, 643, 643, 642, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy  Kappa
##   1     0e+00  0.80115   0.56649
##   1     1e-04  0.80822   0.57847
##   1     1e-03  0.80962   0.57792
##   1     1e-02  0.80962   0.57695
##   1     1e-01  0.80964   0.58004
```

```
##   3     0e+00  0.82367   0.61373
##   3     1e-04  0.81800   0.60558
##   3     1e-03  0.81941   0.60496
##   3     1e-02  0.82226   0.61135
##   3     1e-01  0.81377   0.58771
##   5     0e+00  0.81794   0.60423
##   5     1e-04  0.82222   0.61495
##   5     1e-03  0.81800   0.60842
##   5     1e-02  0.81516   0.59970
##   5     1e-01  0.81659   0.59472
##   7     0e+00  0.81939   0.61117
##   7     1e-04  0.81377   0.60043
##   7     1e-03  0.82502   0.62356
##   7     1e-02  0.81657   0.60372
##   7     1e-01  0.81659   0.59472
##   9     0e+00  0.81234   0.59305
##   9     1e-04  0.80953   0.59309
##   9     1e-03  0.81516   0.60234
##   9     1e-02  0.81520   0.60177
##   9     1e-01  0.81659   0.59472
## 
## Tuning parameter 'bag' was held constant at a value of FALSE
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 7, decay = 0.001 and bag = FALSE.
```

```
#Generate predictions
final_pred <- predict(fit_final, validation)
# Save the solution to a dataframe with two columns: PassengerId and Survived
(prediction)
gender_submission <- read_csv("~/Data Science Projects/gender_submission.csv")
results <- data.frame(PassengerID = gender_submission$PassengerId)
solution <- results %>% mutate(Survived = ifelse(final_pred=="Survived",1,0))
# Write the solution to file
write_csv(solution, path = '~/Data Science Projects/final_solution.csv', col_names =
TRUE)
```

Final Model accuracy

```
actual<-gender_submission%>%mutate(outcome = ifelse(gender_submission$Survived ==
'1', "Survived", "Perished"))
predicted <- solution%>%mutate(outcome = ifelse(solution$Survived == '1', "Survived",
"Perished"))
#Compute the accuracy
acc <- confusionMatrix(as.factor(predicted$outcome), as.factor(actual$outcome),
positive="Survived")
acc
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction Perished Survived
##   Perished      231       29
##   Survived       35      123
## 
##                Accuracy : 0.847
```

```
##                  95% CI : (0.809, 0.88)
##     No Information Rate : 0.636
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.672
##
##  Mcnemar's Test P-Value : 0.532
##
##             Sensitivity : 0.809
##             Specificity : 0.868
##          Pos Pred Value : 0.778
##          Neg Pred Value : 0.888
##              Prevalence : 0.364
##          Detection Rate : 0.294
##    Detection Prevalence : 0.378
##       Balanced Accuracy : 0.839
##
##        'Positive' Class : Survived
##
```

```
#Generate table for results
final_accuracy <- acc$overall['Accuracy']
final_accuracy
```

```
## Accuracy
##  0.84689
```

Conclusion

Looking at the outcome obtained above, an accuracy of 0.84689 was obtained when computed on the predicted survivors in the provided gender_submission dataset against the predicted survivors using the nnet model.The final model results (final_solution file) were submitted on https://www.kaggle.com/submissions/20054628/20054628.raw for evaluation and scored an overall accuracy of 0.78468 on an unseen data, which sounds to be a good model.

References

1. ***Titanic - Machine Learning from Disaster*** https://www.kaggle.com/c/titanic

2. Clark, M. (2013), ***"An Introduction to Machine Learning with Application in R"***, Center for Social Research, University of Notre Dame.