STATISTICAL LEARNING PROJECT

SUPERVISED LEARNING

DECISION TREES, RANDOM FOREST AND ARTIFICIAL NEURAL NETWORK APPLIED TO TITANIC CASE

Report by

Amina Khalfa

ID 941835

DSE MSc Student

Titanic
case

Table of Contents

ABSTRACT

Titanic is one of a British passenger liner that sunk in the North Atlantic Ocean, in the early hours of the morning. Becoming one of the worst commercial maritime catastrophes in history. The RMS Titanic sunk on April 15, 1912, on her maiden voyage, after striking with an iceberg. Unfortunately, there were not enough lifeboats for everyone onboard, resulting in the deaths of 1502 passengers and crew members out of a total of 2224.

While survival required a certain amount of chance, it appears that some groups of individuals were more likely to live than others. The aim of this project is to estimate whether Titanic passengers would survive based on demographic, socioeconomic, and familial characteristics. First, we will do some analysis to have an overview of the type of data we are dealing with, as well as, any possible correlations, explaining step by step, what we are doing.

The aim of the project is on analysis to have an overview of the different characteristics and how they could affect the passenger survivability, as well as the creation of predictive models to forecast survivals, from the Titanic, using the following learning techniques:

- Decision tree
- Random Forest
- ANN

ALGORITHMS

DECISION TREES: When the connection between characteristics and outcomes is nonlinear or when features interact with each other, linear regression and logistic regression models fail. It's time to put your best foot forward for the decision tree! Tree-based models divide the data numerous times based on particular feature cut-off values. Different subsets of the dataset are produced as a result of the splitting, with each instance belonging to one of them. Terminal or leaf nodes are the ultimate subsets, whereas internal nodes or split nodes are the intermediate subsets. The average outcome of the training data in this node is used to predict the outcome in each leaf node. Trees can be used to classify and predict outcomes. A tree may be grown using a variety of algorithms. They differ in terms of the tree's potential structure (e.g., the number of splits per node), the criterion for locating splits, when to cease splitting, and how to estimate basic models within leaf nodes. The classification and regression trees (CART) technique is the most widely used tree induction algorithm. We'll focus on CART, although most other tree types have similar interpretations.

RANDOM FOREST: Random Forest is a supervised learning method that may be used to classify and predict data. However, it is mostly employed to solve categorization issues. A forest, as we all know, is made up of trees, and more trees equals a stronger forest. Similarly, the random forest method constructs decision trees from data samples, extracts predictions from each, and then votes on the best answer. It's an ensemble approach that's superior than a single decision tree since it averages the results to reduce over-fitting. The following stages will help us understand how the Random Forest algorithm works. Begin by selecting random samples from a specified dataset. Following that, this algorithm will create a decision tree for each sample. The forecast result from each decision tree will then be obtained. Voting will be place in this stage for each projected outcome. Finally, choose the prediction result with the most votes as the final prediction result. By averaging or integrating the outputs of several decision trees, it avoids the problem of overfitting. Random forests perform better than a single decision tree for a wide range of data items. The variance of a random forest is lower than that of a single decision tree. Random forests are extremely adaptable and have a high level of accuracy.

ARTIFICIAL NEURAL NETWORK: In the disciplines of AI, machine learning, and deep learning, neural networks mimic the activity of the human brain, allowing computer programmers to identify patterns and solve common problems. Artificial neural networks (ANNs), often known as neural networks, are a subset of machine learning that are at the heart of deep learning techniques. Their name and structure are derived from the human brain, and they resemble the way real neurons communicate with one another. A node layer contains an input layer, one or more hidden layers, and an output layer in artificial neural networks (ANNs). Each node, or artificial neuron, is connected to the others and has a weight and threshold linked with it. The data input is processed through several layers of multiplication with the learnt weights and via non-linear transformations to produce predictions with a neural network. Depending on the design of the neural network, a single prediction might entail millions of mathematical operations. We humans will never be able to follow the exact mapping from data intake to prediction. To comprehend a neural network prediction, we would have to examine millions of weights that interact in a complicated fashion.

DATA CLEANING AND ANALYSIS

We start by reading in csv the datasets, which R converts to data frames. Text strings are factors in csvs; the following argument transforms them to characters in data frames. The "na.strings" parameter instructs R how to handle NA's and blank. Test contains 418 passengers, compared to 891 in the train data frame. Numbers, integers, and characters are among the data kinds.

```
> str(train)
'data.frame':   891 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs
 Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...
 $ Sex        : chr  "male" "female" "female" "female" ...
 $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : chr  NA "C85" NA "C123" ...
 $ Embarked   : chr  "S" "C" "S" "S" ...
> str(test)
'data.frame':   418 obs. of  11 variables:
 $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
 $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
 $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myles, Mr. Thom
as Francis" "Wirz, Mr. Albert" ...
 $ Sex        : chr  "male" "female" "male" "male" ...
 $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
 $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
 $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
 $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
 $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
 $ Cabin      : chr  NA NA NA NA ...
 $ Embarked   : chr  "Q" "S" "Q" "S" ...
```

Figure 1: Dataset Structure

Below are some summary stats on trains and these numerical features: Pclass, Age, and Fare. In the original csv, Sex and Embarked were factor variables (categorical quantities). When the datasets were read into R as data frames, they were transformed to characters. After converting the characters back to factors to view the summary statistics. A brief glance at the train data frame's summary data reveals:

- Males made up almost two-thirds of the passengers.
- The average age was in their late twenties.
- The majority of the passengers were in third class.
- The majority of travelers paid around $14 for their ticket.
- The vast majority of passengers departed from Southampton.

```
> summary(train[, c(3,6,10)])
     Pclass          Age             Fare
 Min.   :1.000   Min.   : 0.42   Min.   :  0.00
 1st Qu.:2.000   1st Qu.:20.12   1st Qu.:  7.91
 Median :3.000   Median :28.00   Median : 14.45
 Mean   :2.309   Mean   :29.70   Mean   : 32.20
 3rd Qu.:3.000   3rd Qu.:38.00   3rd Qu.: 31.00
 Max.   :3.000   Max.   :80.00   Max.   :512.33
                 NA's   :177
> summary(as.factor(train$Sex))
female   male
   314    577
> summary(as.factor(train$Embarked))
   C    Q    S NA's
 168   77  644    2
```

Figure 2: Data Summary Statistics

MISSING DATA ANALYSIS

To begin, we make a variable called "Set" and code each data frame as "train" or "test." We'll be able to keep track of which data is which this way. Then we make both data frames' features the same. Now we can look at the total number of missing values in the dataset. "Cabin" has the most missing data of all the characteristics (77 percent). "Age" contains a large number of missing values (20%), but "Fare" and "Embarked" only have one and two values missing, respectively.

```
> sapply(All_Data, function (x) {sum(is.na(x))})
PassengerId    Survived      Pclass        Name
          0         418           0           0
        Sex         Age       SibSp       Parch
          0         263           0           0
     Ticket        Fare       Cabin    Embarked
          0           1        1014           2
        Set
          0
```

Figure 3: Missing Values

EXPLORATORY DATA ANALYSIS

The survival discrepancy by sex is depicted in the grouped bar plot below. The sex-based population count has nothing to do with sex-based survival. On the Titanic, male passengers outnumbered female passengers three to one. However, 81 percent of men died, while just 26 percent of females died. Males made up 53% of the total number of people that died, while females made up 9%. As a result, sex had a significant impact in survival rates.

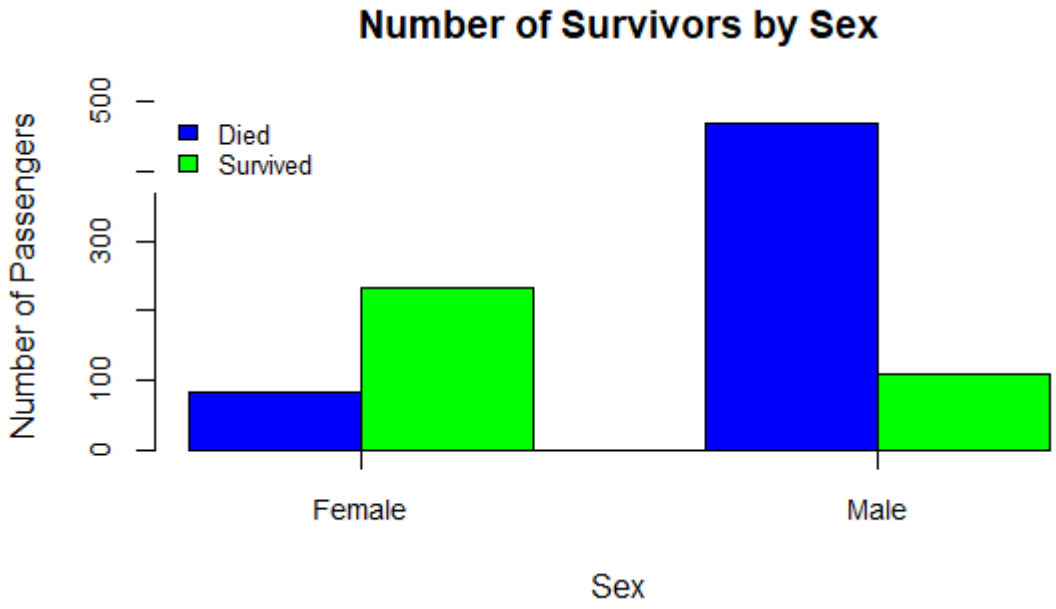

Figure 4: Number of Survivors by Sex

Next, we'd like to look at the influence of social class on survival. I make a table with the three classes in the Pclass feature.

```
> prop.table(table(All_Data$Pclass))

        1         2         3
0.2467532 0.2116119 0.5416348
```

Figure 5: Proportion of each passenger class

The first and second classes each include around a fourth of the passengers (25 percent in first and 21 percent in second). Third-class passengers, on the other hand, make up half of all passengers (54 percent). Now, we can verify survival rates by class using simply train data.

```
> prop.table(table(train$Pclass, train$Survived),
 1)

          0         1
1 0.3703704 0.6296296
2 0.5271739 0.4728261
3 0.7576375 0.2423625
```

Figure 6: Passenger class by Survived

The results are startling: survival rates drop dramatically as one's social status rises. Two-thirds (63%) of first-class passengers made it out alive. Half of the passengers in second class (47%) survived. Only a quarter of third-graders (24%) made it out alive.

Class had a significant influence on survival, as shown in the grouped bar plot below. In reality, the huge number of third-class passengers may have contributed to the Titanic's overall high death toll.

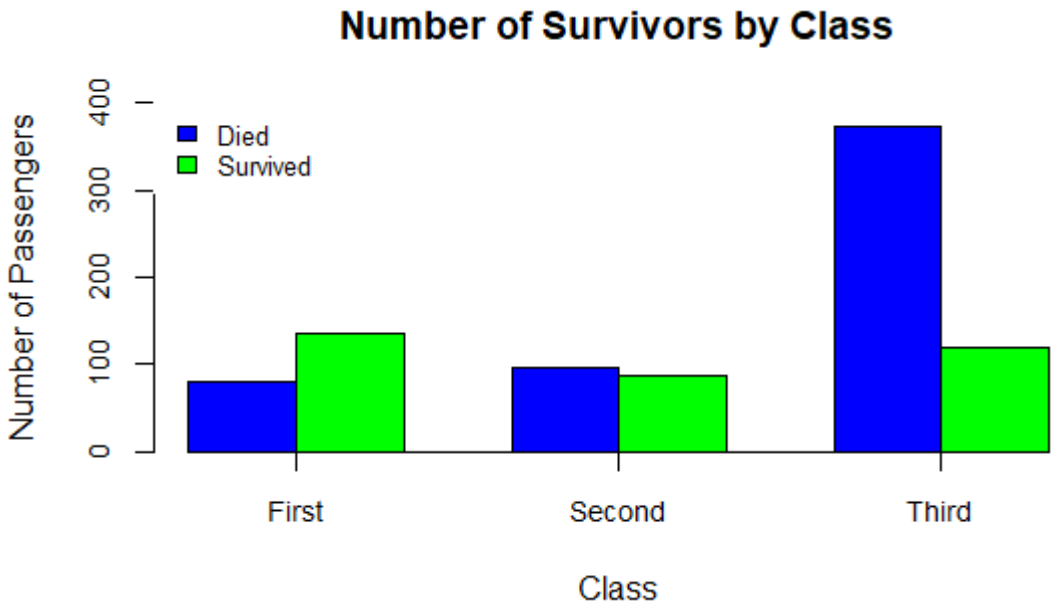

Figure 7: Number of Survivors by Class

RESEARCH

APPLYING DECISION TREE TO THE PROBLEM

We'll use the training set to build a decision tree in this section. For classification issues, decision trees are employed, particularly for binary classifiers. To grow a tree, we must first import the rpart library. Before using the decision tree approach, the train and test sets are already created which we use. Test data includes 418 observations while training data includes 891 observations.

The training set was then fitted with a model. Except for PassengerId, and other 3 variables, we use all of the dataset's category and numerical characteristics. Using the rpart.plot() method, we plot the growing tree.

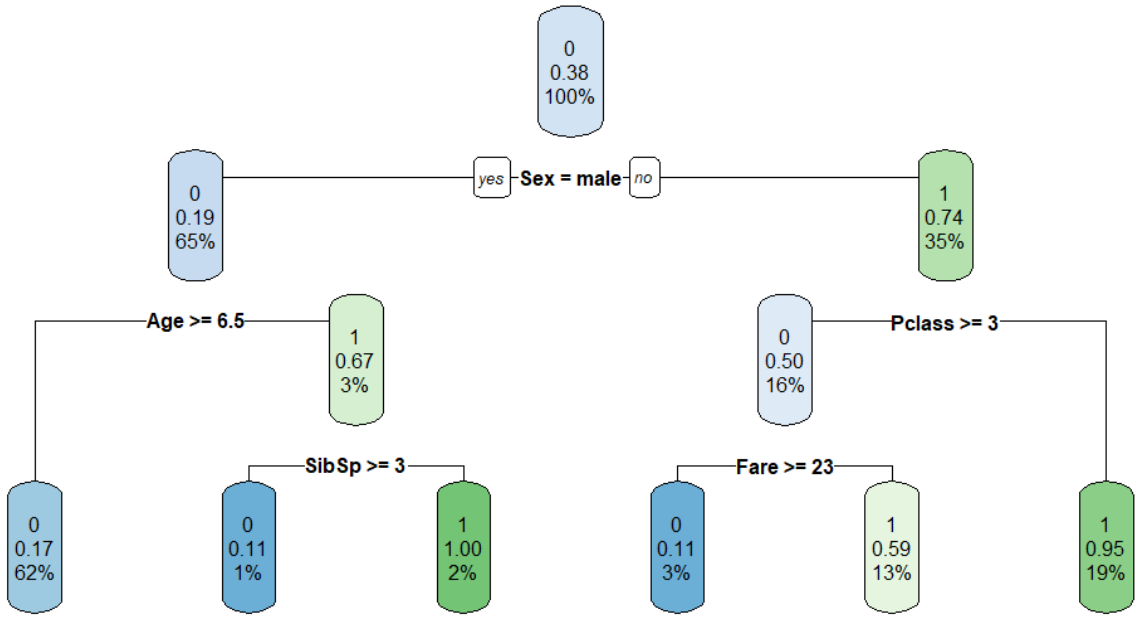

Figure 8: Decision Tree

The following are the rules followed in this model:

- If the sex is male and age is greater than 6.5 then there are 3 percent chances to not survived
- If sex is female than 35% chances to survive
- If Sibsp is more than or equal to 3 then chances survive
- If passenger class is 3 then 19% chances to survive
- If fare is more than 23 than also 13% chances to survive and 3% chances to death.

When a tree becomes overgrown (i.e., has too many branches) and complex, we "prune" it according to specific requirements. However, because our trees do not appear to be overly basic or complex, we utilize it as is. Using the growing tree, we generate a prediction for the test set. To do so, we employ the predict() method. And finally, confusion matrix is made with the help of caret library. And this model performs pretty well, as it has about 97% accuracy.

```
Confusion Matrix and Statistics

   my_prediction
      0   1
  0 260   6
  1   6 146

               Accuracy : 0.9713
                 95% CI : (0.9504, 0.9851)
    No Information Rate : 0.6364
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.938

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9774
            Specificity : 0.9605
         Pos Pred Value : 0.9774
         Neg Pred Value : 0.9605
             Prevalence : 0.6364
         Detection Rate : 0.6220
   Detection Prevalence : 0.6364
      Balanced Accuracy : 0.9690

       'Positive' Class : 0
```

Figure 9: Decision Tree Confusion Matrix

In a classification issue, a confusion matrix summarizes the predicted findings. Correct and wrong predictions are listed in a table with their respective values, divided down by class.

- **True Positive:** You correctly anticipated a positive outcome, and it came true. You correctly guessed that an animal would be a cat.
- **True Negative:** You correctly anticipated negative, and it came to pass. You correctly guessed that the animal isn't a cat, and it isn't (it's a dog).
- **Untrue Positive (Type 1 Error):** You anticipated something positive, but it turned out to be false. That animal was supposed to be a cat, but it is actually a dog.
- **False Negative (Type 2 Error):** You anticipated a negative outcome, but it was incorrect.

The sensitivity/specificity trade-offs for a binary classifier are frequently described using ROC curves. The majority of machine learning classifiers generate real-valued scores that reflect the accuracy of the prediction that a particular example is positive. The AUC indicates how well the model predicts 0 classes as 0 and 1 courses as 1. The higher the AUC, the better the model predicts 0 classes as 0 and 1 classes as 1.

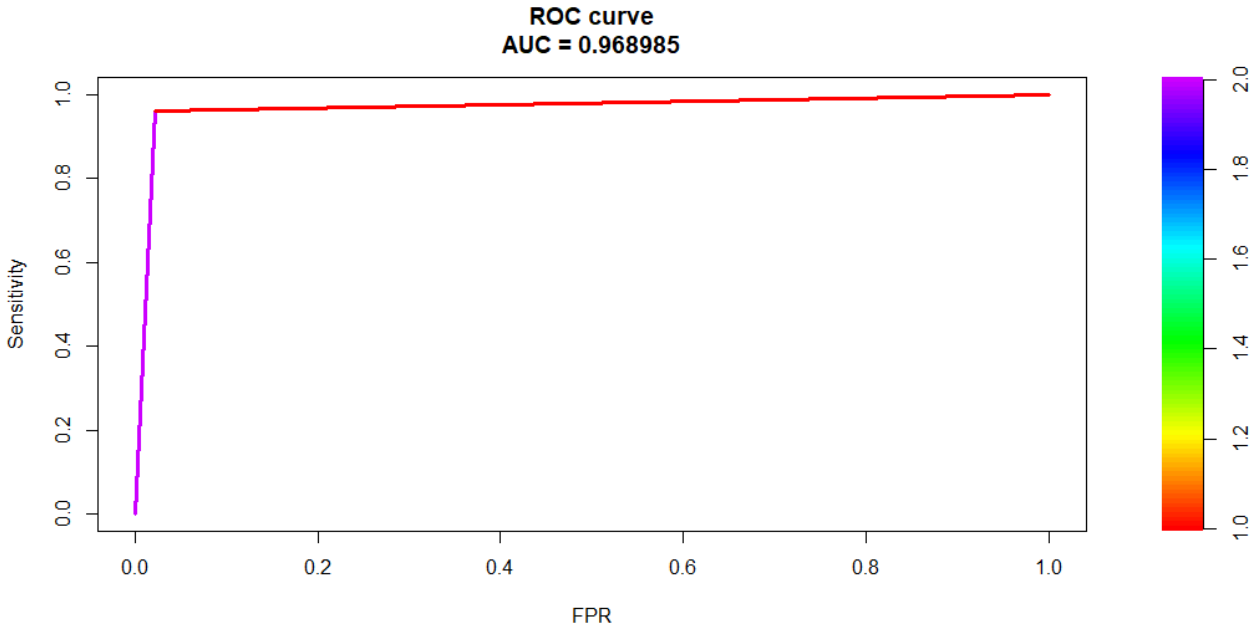

Figure 10: Decision Tree ROC Curve

APPLYING RANDOM FOREST TO THE PROBLEM

A huge number of decision trees are produced in the random forest technique. Every observation is incorporated into the decision-making process. The final result is based on the most common conclusion for each observation. A fresh observation is put into all of the trees, with each categorization model receiving a majority vote. Below figure shows the variables used in random forest. Moreover, it also includes number of trees and residuals which are very lower with more than 45% variance explained.

```
Call:
 randomForest(formula = Survived ~ Age + Fare + Sex + Pclass +
  Parch + SibSp + Embarked, data = train_rand)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 0.1322527
                    % Var explained: 45.1
```

Figure 11: Random Forest Summary

From below random forest variable importance plot we can see that sex is the most important variable. The least important variable is Embarked.

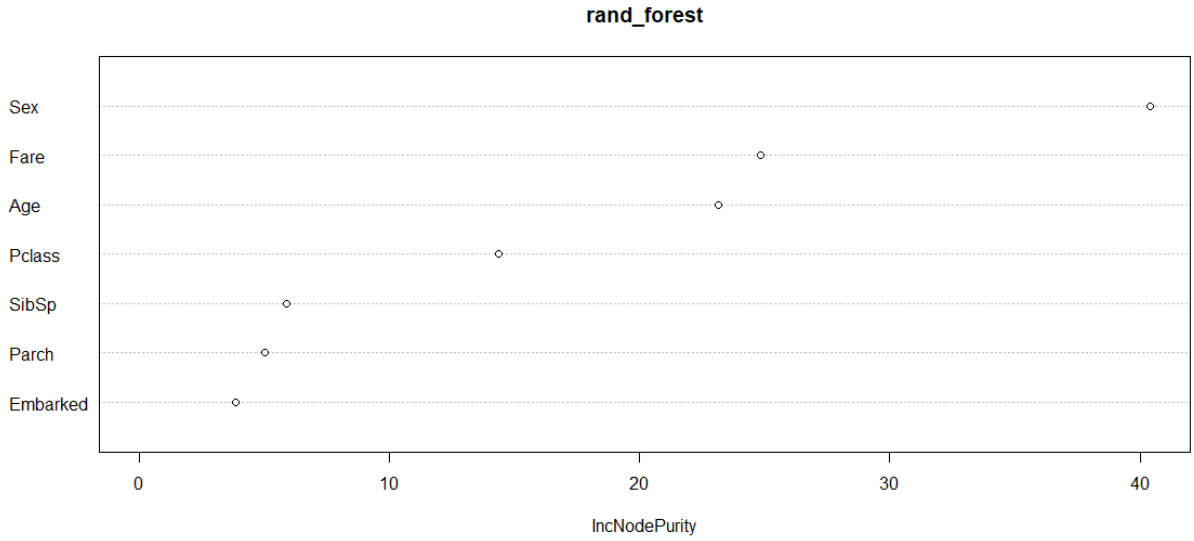

Figure 12: Random Forest Variable Importance Plot

Below given confusion matrix shows that accuracy of this model is 81. 82% with 23 false positives (FP) and 53 false negatives (FN).

```
Confusion Matrix and Statistics

   pred_rf_valid
      0   1
  0 243  23
  1  53  99

                Accuracy : 0.8182
                  95% CI : (0.7778, 0.854)
    No Information Rate : 0.7081
    P-Value [Acc > NIR] : 1.571e-07

                   Kappa : 0.5898

 Mcnemar's Test P-Value : 0.0008794

            Sensitivity : 0.8209
            Specificity : 0.8115
         Pos Pred Value : 0.9135
         Neg Pred Value : 0.6513
             Prevalence : 0.7081
         Detection Rate : 0.5813
   Detection Prevalence : 0.6364
      Balanced Accuracy : 0.8162

       'Positive' Class : 0
```

Figure 13: Random Forest Confusion Matrix

From below given ROC curve it can be interpreted that area under the curve is 0.78 which is not more than decision tree.

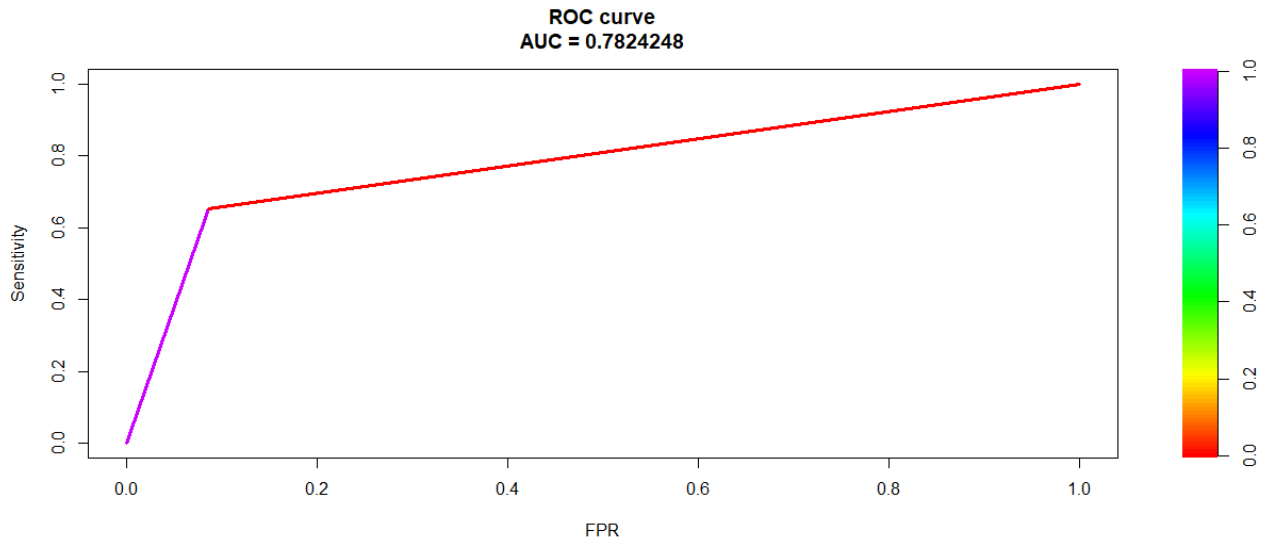

Figure 14: Random Forest ROC Curve

APPLYING ARTIFICIAL NEURAL NETWORK TO THE PROBLEM

A neural network (also known as an artificial neural network) may learn from examples. The artificial neural network (ANN) is a data processing model based on the biological neuron system. To solve issues, it is made up of a huge number of highly linked processing components called as neurons. It takes a non-linear path and processes data in parallel across all nodes. A neural network is an adaptable, sophisticated system. It is adaptive if it can modify its internal structure by adjusting the weights of inputs. Below is the output of convergence of neural network in R. Here there is need to use nnet library.

```
# weights:  21
initial  value 214.179387
iter  10 value 139.003482
iter  20 value 118.454310
iter  30 value 115.527538
iter  40 value 104.667926
iter  50 value 97.945614
iter  60 value 97.062275
iter  70 value 96.938022
final  value 96.933305
converged
```

Figure 15: ANN Convergence

Below figure shows the main neural network visualization. Let us interpret these

- Your raw data variables are the left-most nodes (i.e., input nodes).
- Your hidden nodes are the nodes in the center (i.e., anything between the input and output nodes). This is when the analogy with images comes in handy. Each node represents a component that the network is learning to identify.
- The ultimate output of your neural network is the far-right (output node(s)) node.

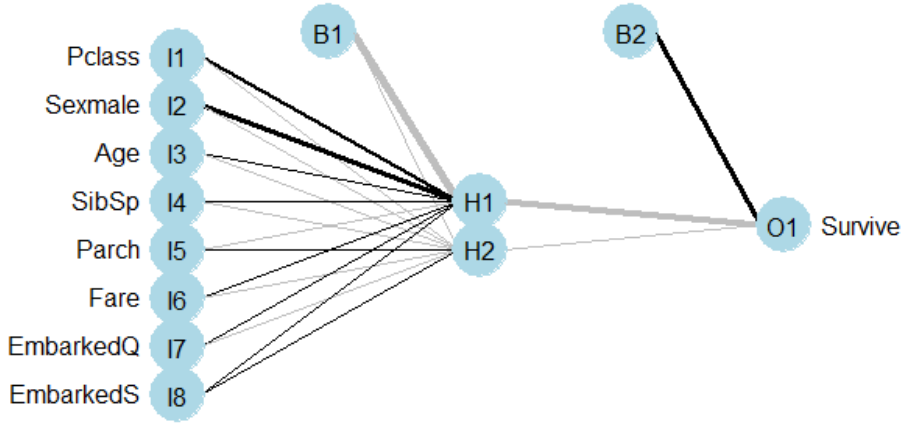

Figure 16: ANN Visualization

Below is the confusion matrics showing that ANN is giving accuracy of 83.97% with 16 false positive values (FP) and 51 false negative values (FN).

```
Confusion Matrix and Statistics

   Survived
      0   1
  0 250  16
  1  51 101

                Accuracy : 0.8397
                  95% CI : (0.801, 0.8736)
     No Information Rate : 0.7201
     P-Value [Acc > NIR] : 6.402e-09

                   Kappa : 0.6357

  Mcnemar's Test P-Value : 3.271e-05

             Sensitivity : 0.8306
             Specificity : 0.8632
          Pos Pred Value : 0.9398
          Neg Pred Value : 0.6645
              Prevalence : 0.7201
          Detection Rate : 0.5981
    Detection Prevalence : 0.6364
       Balanced Accuracy : 0.8469

        'Positive' Class : 0
```

Figure 17: ANN Confusion Matrix

The trade-off between sensitivity (or TPR) and specificity (1 – FPR) is depicted by the ROC curve. Classifiers with curves that are closer to the top-left corner perform better. Here we got a good AUC of 0.80 showing good fitting of model.

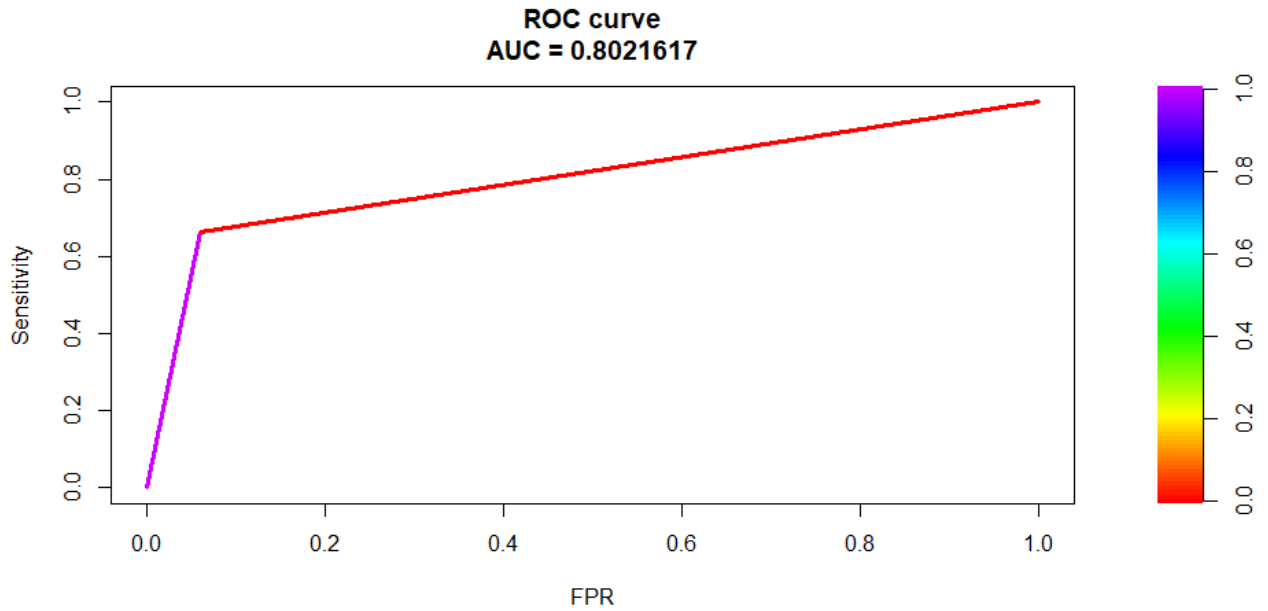

Figure 18: ANN ROC Curve

CONCLUSION

In whole analysis, our goal was to predict survivals of titanic mystery. So, first we have used exploratory data analysis to see the effect of different features. Then implemented machine learning algorithms to get the final decision about the features which are important. Here decision tree gives the best accuracy and AUC in ROC curve which is more than 95% showing the best model. It is able to distinguish between survived and non-survived persons. Moreover, other both have large number of FP and FN values effecting its accuracy of predicting. ANN is also applied for precision as we need to get great precision. But it was not possible as it gives large amount of FP values.

LIST OF FIGURES