Data analysis and prediction of Titanic survivors

Arjun Nischalkumar Sanchala
Master of Information and Communications Technology
Western Sydney University
Sydney, Australia
Email: arjun.sanchala@gmail.com

***Abstract*—In 1912, a deadly incident occurred, the Titanic sank. Enormous number of people lost their lives and family members. The aim of this research paper is to analyze the data of people who went aboard the Titanic, and to find the hidden pattern to predict the survival probability of unseen data. Furthermore, decision tree and support vector machine have been used to build prediction models. At the end, these models have been compared after parameter optimization.**

Keywords—Data analysis, predicition, probability, Support Vector Machine (SVM), Decision Tree, Data preprocessing, Cross validation, Model testing, Accuracy.

I. INTRODUCTION

The Data science field has helped data scientists to make crucial decisions based on historical data. Machine learning is one of the predominant branches of data science. Machine learning techniques like classification and regression have been proven significantly useful to predict future events of unseen data. The Titanic dataset has all the information of passengers who aboard the titanic ship in 1911. To find the relation between survived passengers and other attributes like Age, sex etc., number of algorithms have been applied. Algorithms such as logistic regression, Decision tree and support vector machine have been proven best with great accuracy on test data. The dataset is available on the internet [1]. In this research paper, decision tree algorithm and support vector machine have been used to build a machine learning model. Subsequently, predictions have been made using those models. At last, model comparison will be there to determine the performance of the models.

II. DATA EXPLORATION AND PREPROCESSING

A. Data Exploration

The dataset has 891 observations and 12 attributes. The information on the characteristics of each attribute is given below.

1. PassengerId (int) - Unique id of each passenger.
2. Survived (int) – Passenger survived? (0 = No; 1 = Yes).
3. Pclass (int) - Passenger class (1 = 1st ; 2 = 2nd ; 3 = 3rd).
4. Name (Chr) – Name of the passenger.
5. Sex (chr) – Sex of the passenger.
6. Age (num) – Age of the passenger.
7. SibSp (int) - Number of siblings or spouses of the passenger.
8. Parch - Number of Parents or Children of the passenger.
9. Ticket (chr) - Ticket number of the passenger.
10. Fare (num) - Fare rate of the passenger.
11. Cabin (chr) - Cabin number of the passenger.
12. Embarked (chr) – Passenger embarkation port. (C = Cherbourg; Q = Queenstown; S = Southampton)

```
> dim(titanic)
[1] 891  12
> str(titanic)
'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
 $ 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  "" "C85" "" "C123" ...
 $ Embarked   : chr  "S" "C" "S" "S" ...
```

Figure 1: Data exploration

B. Data preprocessing

Data preprocessing is an essential step before building our model. In addition, it directly affects the accuracy of the model. First and foremost step to take in data preprocessing is to remove unnecessary columns. The dataset has unnecessary attributes like PassengerId, Cabin, Name, Ticket. These attributes are not as important as other attributes are in model building. So, these attributes have been removed. The next step is to convert the Pclass and Survived attributes from int to factor. This will be helpful in visualization of model output. Pclass have been converted from values 1, 2, 3 to ‘upper’, ‘middle’ and ‘lower’ respectively. And, Survived attribute have been converted from values 0 and 1 to ‘died’ and ‘survived’ respectively. At last, column ‘Age’ has 177 missing values. There are several ways to handle missing values, like replacing them with their mean or median or remove rows which has missing values. Here, I have selected the second approach and deleted the rows with missing values. The density and frequency graph is shown in figure 2. R code of data preprocessing step is attached in the appendix[Appendix A].

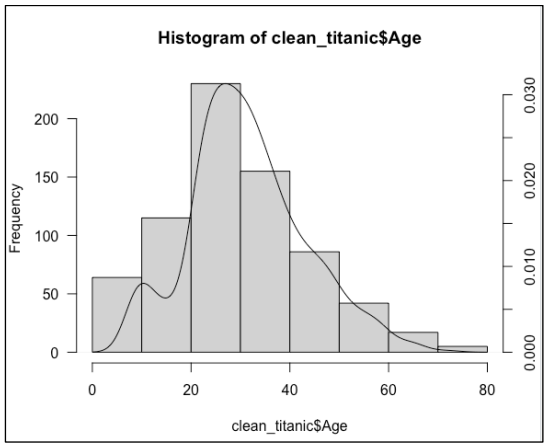

Figure 2: Density plot of Age attribute

III. Aim and Objective

The main objective of this research is to build classification models as the target class is Survived. In addition, comparison of accuracies of both the models will be compared after parameter optimization. The aim will be maximize the accuracy and minimize the misclassification rate on the test dataset.

IV. Model 1: Decision Tree

A. What is decision tree

A decision tree algorithm is a machine learning algorithm to classify the data based on the correlation between attributes. There are three main things in the decision tree: root node, branches and leaf node. The tree-like structure represent the best predictor and splits according the probability of an outcome.

B. Building the model

- Before building the model, splitting the dataset into training data and testing data is necessary.
- In most cases, 75% data of the original dataset need to be allocated to training data. And, remaining data should be used at testing stage.
- If the dataset is large enough, 85-90% data of the original dataset should be used to build a better model.
- Here, I am using 77% of the data as training data. Figure 3 is showing the percentage split of the original data vs accuracy can be achieved by that data (without parameter optimization).

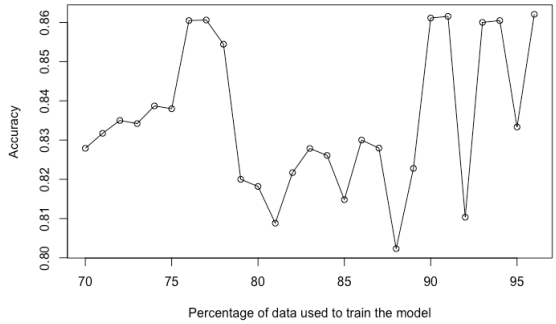

Figure 3: Training data split vs accuracy.

After splitting the training and testing data, 'rpart' library was used to fit the model on training data. The accuracy of 83.8% was achieved and 29 sample was misclassified. The visualization is given in figure 4. The model have given the list of significant attributes: Age, Fare, Pclass, Sex and SibSp[Appendix B].

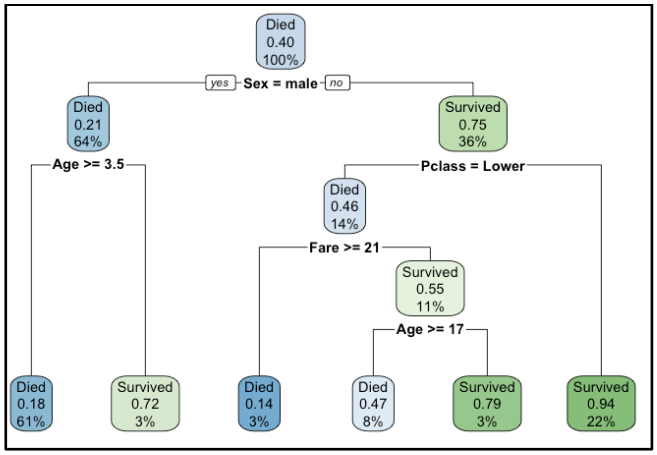

Figure 4: Decision tree with maximum depth of 4

C. Model optimization

The 'rpart' library has option to optimize the decision tree model: control. I have set the minimum split of the tree to 3, maximum depth of the tree to 6 and complexity to 0. This parameters are the inputs of 'control' method in rpart. We can say from the figure 5 that as complexity parameter closes to 0, the error reduces.

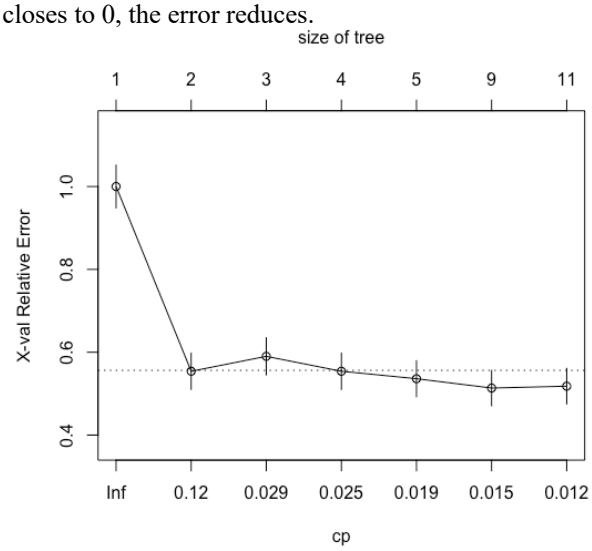

Figure 5: Complexity parameter vs error

The optimized model has increased the accuracy to ~87% and has reduced the misclassification samples to 23.

V. Model 2: Support Vector Machine

A. What is Support vector machine

A support vector machine algorithm is a supervise machine learning method. The model uses the decision boundary (line or

hyperplane) to differentiate the data into various classes. It can solve linear and non-linear problems. In addition, not only classification problems but also regression problems can be solved by SVM.

B. Building and tuning the model

To build a support vector machine model, R studio has a package called 'e1071'. It has optimization parameters like 'gamma' and 'cost'. The gamma parameter decides the behavior of the redial kernel SVM model. Usually, low gamma values give the best decision rule and high cost value does not let the model to overfit[Figure 6] [2]. I have used tune out method provided by the package and set the list of values for gamma and cost[Appendix C]. A summary of model was generated and suggested best optimization parameters values(best gamma and cost values) [Appendix D]. Furthermore, 10-fold cross validation was applied to the data to validate the model.

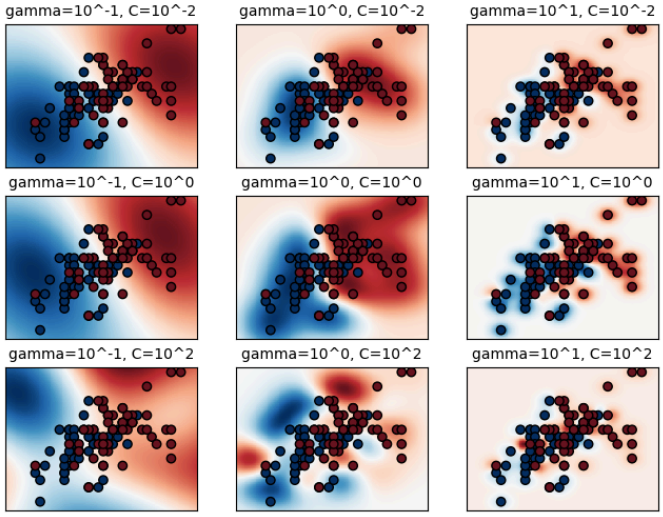

Figure 6: Effect of gamma and cost values on decision rule

By building and tuning the model based on given gamma and cost values in summary, gamma = 0.5 and cost = 1, accuracy was 84.81% with 24 misclassify samples.

VI. Model comparison

Decision tree models perform well for classification tasks. The models find the hidden pattern in training dataset. The reason behind the best performance is, decision tree algorithm calculates probability of occurrence and learn faster from training data [3]. This is exactly true for this dataset.

After optimization, the decision tree model achieved ~87% accuracy. Despite been optimized, the support vector machine achieved ~85% accuracy which is quite less than expected. The decision tree model performed great compared to support vector machine.

VII. Future work

As misclassification rate is higher in support vector machine, one can minimize this by applying several preprocessing steps like replacing missing values by their mean or median, applying one hot encoding and transform the data before training the model etc. The performance of decision tree can be boosted by adjusting the tree depth and split of the branches. This will be the input of 'control' variable of 'rpart' function.

VIII. Reference

[1] K. B. Hosseini, "kaggle," [Online]. Available: https://www.kaggle.com/heptapod/titanic. [Accessed 12 10 2020].

[2] N. S. Chauhan, "KDnuggets," [Online]. Available: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html#:~:text=The%20decision%20criteria%20are%20different,two%20or%20more%20sub%2Dnodes.&text=The%20decision%20tree%20splits%20the,in%20most%20homogeneous%20sub%2Dnodes..

[3] "scikit-learn" [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html.

APPENDIXES

APPENDIX A: DATA PREPROCESSING

```
clean_titanic <- titanic
clean_titanic <- select(clean_titanic, -c(PassengerId, Cabin, Name, Ticket))
clean_titanic <- mutate(clean_titanic, Pclass = factor(Pclass, levels = c(1, 2, 3),
                        labels = c('Upper', 'Middle', 'Lower')),
         Survived = factor(Survived, levels = c(0, 1), labels = c('Died', 'Survived')))
clean_titanic <- na.omit(clean_titanic)
```

APPENDIX B: IMPORTANT ATTRIBUTES

```
Classification tree:
rpart(formula = Survived ~ ., data = data_train, method = "class",
    control = control)

Variables actually used in tree construction:
[1] Age    Fare   Pclass Sex    SibSp

Root node error: 222/542 = 0.40959

n= 542

        CP nsplit rel error  xerror     xstd
1 0.445946      0   1.00000 1.00000 0.051570
2 0.031532      1   0.55405 0.55405 0.043925
3 0.027027      2   0.52252 0.59009 0.044896
4 0.022523      3   0.49550 0.55405 0.043925
5 0.015766      4   0.47297 0.53604 0.043410
6 0.013514      8   0.40991 0.51351 0.042739
7 0.010000     10   0.38288 0.51802 0.042876
```

APPENDIX C: MODEL TUNING

```
tune.out = tune(svm, train.x = Survived ~ Age + Pclass + Sex + SibSp + Fare, data = data_test,
                ranges = list(cost = c(0.1, 1, 5, 10), gamma = c(0.005, 0.5,1,2)), kernel = "radial")

bestmodel = tune.out$best.model
summary(tune.out)
```

APPENDIX D: GAMMA, COST AND ERROR

```
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.1943932

- Detailed performance results:
    cost gamma     error dispersion
1  1e-01   0.5 0.2486687 0.06014128
2  1e+00   0.5 0.1943932 0.05695034
3  1e+01   0.5 0.1960061 0.06324281
4  1e+02   0.5 0.2279058 0.06674157
5  1e+03   0.5 0.2454173 0.07436611
6  1e-01   1.0 0.3108295 0.09125079
7  1e+00   1.0 0.2119304 0.06550038
8  1e+01   1.0 0.2183308 0.06825704
9  1e+02   1.0 0.2421659 0.07186241
10 1e+03   1.0 0.2676651 0.06513774
11 1e-01   2.0 0.3601126 0.10515043
12 1e+00   2.0 0.2151818 0.08189103
13 1e+01   2.0 0.2278546 0.07094031
14 1e+02   2.0 0.2564772 0.06062909
15 1e+03   2.0 0.2835637 0.06094316
16 1e-01   3.0 0.3776242 0.10000345
17 1e+00   3.0 0.2358423 0.08499211
18 1e+01   3.0 0.2437020 0.07592126
19 1e+02   3.0 0.2739375 0.05471115
20 1e+03   3.0 0.2882744 0.05406507
21 1e-01   4.0 0.3903482 0.10273133
22 1e+00   4.0 0.2501024 0.06537469
23 1e+01   4.0 0.2612391 0.05412703
24 1e+02   4.0 0.2819252 0.06262726
25 1e+03   4.0 0.2977727 0.06995350
```

R CODES

```
set.seed(678)
path <- '/Users/aj/Downloads/2dfd2de0d4f8727f8734222c5d959fff5-fa71405126017e6a37bea592440b4bee94bf7b9e/titanic.csv'
titanic <-read.csv(path)

#Data information
head(titanic)
dim(titanic)
str(titanic)
sapply(titanic, function(x) sum(is.na(x)))

#Data preprocessing
library(dplyr)
# Drop variables
clean_titanic <- titanic
clean_titanic <- select(clean_titanic, -c(PassengerId, Cabin, Name, Ticket))
clean_titanic <- mutate(clean_titanic, Pclass = factor(Pclass, levels = c(1, 2, 3),
                        labels = c('Upper', 'Middle', 'Lower')),
         Survived = factor(Survived, levels = c(0, 1), labels = c('Died', 'Survived')))
clean_titanic <- na.omit(clean_titanic)
glimpse(clean_titanic)
tail(clean_titanic)

#Density plot of Age attribute
hist(clean_titanic$Age, las=1)
par(new=TRUE)
plot(density(clean_titanic$Age), yaxt="n", xaxt="n",
             bty='n', xlab="", ylab="", main='')
axis(4)
```

```
#train test split Function
create_train_test <- function(data, size = 0.8, train = TRUE) {
  n_row = nrow(data)
  total_row = size * n_row
  train_sample <- 1: total_row
  if (train == TRUE) {
    return (data[train_sample, ])
  } else {
    return (data[-train_sample, ])
    }
}

ind <- c()
acc <- c()

# Checking accuracy for assigning 70% to 96% of original data to training data.
for (i in 70:96){
  data_train <- create_train_test(clean_titanic, (i/100), train = TRUE)
  data_test <- create_train_test(clean_titanic, (i/100), train = FALSE)
  dim(data_train)

  #install.packages("rpart.plot")

  library(rpart)
  library(rpart.plot)
  fit <- rpart(Survived~., data = data_train, method = 'class')
  #rpart.plot(fit, extra = 106)

  predict_unseen <-predict(fit, data_test, type = 'class')
  table_mat <- table(data_test$Survived, predict_unseen)
  print(table_mat)

  accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
  print(paste(i,'Accuracy of the model is', round(accuracy_Test*100, digits = 2), '% on test data.'))
  ind <- append(ind,i)
  acc <- append(acc,accuracy_Test)
}

plot(ind,acc,xlab = "Percentage of data used to train the model", ylab = "Accuracy" ,type = "o")
```

```r
#training and testing data.
data_train <- create_train_test(clean_titanic, 0.76, train = TRUE)
data_test <- create_train_test(clean_titanic, 0.76, train = FALSE)
dim(data_train)

--------------------------------------------------------------------------------------------
                      # Model 1: Decision tree
--------------------------------------------------------------------------------------------

#install.packages("rpart.plot")
library(rpart)
library(rpart.plot)

#Setting optimization criteria.
control <- rpart.control(minsplit = 3, maxdepth = 6,cp = 0)

#fitting the model
fit <- rpart(Survived~., data = data_train, method = 'class', control = control)

#Tree visualization
rpart.plot(fit, extra = 106)

#Predict test data
predict_unseen <-predict(fit, data_test[,2:8], type = 'class')

#complexity parameter
printcp(fit)
plotcp(fit)
table_mat <- table(data_test$Survived, predict_unseen)

#Accuracy
accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
print(paste('Accuracy of the model is', round(accuracy_Test*100, digits = 2), '% on test data.'))


--------------------------------------------------------------------------------------------
                    # Model 2: Support Vector Machine
--------------------------------------------------------------------------------------------

#install.packages('e1071')
library(e1071)

data_train <- create_train_test(clean_titanic, 0.78, train = TRUE)
data_test <- create_train_test(clean_titanic, 0.78, train = FALSE)

#Tuning the model with various cost and gamma values
tune.out = tune(svm, train.x = Survived ~ Age + Pclass + Sex + SibSp + Fare, data = data_train,
                ranges = list(cost = c(0.1, 1, 5, 10, 100), gamma = c(0.005, 0.5,1,2)), kernel = "radial")

#selecting the best model
bestmodel = tune.out$best.model
summary(tune.out)

#Building the model with derived values
classifier = svm(formula = Survived ~ ., data = data_train, type = 'C-classification',
                 kernel="radial", cost = 1, gamma = 0.5)

#predict the testing data
y_pred = predict(classifier, data_test)

#Accuracy
table_b = table(y_pred, data_test$Survived)
accuracy_Test <- sum(diag(table_b)) / sum(table_b)
print(paste('Accuracy of the model is', round(accuracy_Test*100, digits = 2), '% on test data.'))
summary(classifier)
```