# Can Dai - HW4

2022-05-25

## HW5

Libraries used for this assignment:

```
library(tidyverse)
library(caret)
library(e1071)
library(ggplot2)
library(stats)
library(factoextra)
library(cluster)
library(rpart)
library(kknn)
library(pROC)
library(readr)

set.seed(123)
```

### Data gathering and integration

For this assignment, I chose to use Titanic - Machine Learning from Disaster Dataset from Keggle (https://www.kaggle.com/competitions/titanic/data). I used the Training set for the machine learning model and testing, since the seperate testing set doesn't include the right predictions to compare the outcome of the predictions.

The data set consists of both categorical and numeric variables. Only the training dataset has the determined survival ordinal variable, 0 = No, 1 = Yes. In addition, The dataset have the following attributes: Name (name of the passenger: char), Sex (male or female: char), Ticket (ticket number: char), Cabin (cabin number), Embarked (Port of embarkation: C= Cherbourg, Q= Queenstown, S= Southampton: char), PassengerId (unique ID assigned: dbl), Pclass (Ticket class, 1= 1st, 2= 2nd, 3= 3rd: dbl), Age (age of the passenger: dbl), SibSp (# of siblings/spouses aboard: dbl), Parch (# of parents/children aboard: dbl) and Fare (passanger fare: dbl).

The aim of this dataset is to train a ML model using the training set that can predict the outcome of passengers.

```
titanic_train <- read_csv("/Users/candai/Desktop/Fundamentals of Data Science/HW5_Titanic/train.csv")

head(titanic_train)
```

```
## # A tibble: 6 x 12
##   PassengerId Survived Pclass Name    Sex     Age SibSp Parch Ticket  Fare Cabin
##         <dbl>    <dbl>  <dbl> <chr>   <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
## 1           1        0      3 Braund~ male     22     1     0 A/5 2~  7.25 <NA>
```

```
## 2           2        1      1 Cuming~ fema~    38     1     0 PC 17~ 71.3  C85
## 3           3        1      3 Heikki~ fema~    26     0     0 STON/~  7.92 <NA>
## 4           4        1      1 Futrel~ fema~    35     1     0 113803 53.1  C123
## 5           5        0      3 Allen,~ male     35     0     0 373450  8.05 <NA>
## 6           6        0      3 Moran,~ male     NA     0     0 330877  8.46 <NA>
## # ... with 1 more variable: Embarked <chr>
```

```
str(titanic_train)
```

```
## 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 Th
##  $ 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()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
summary(titanic_train)
```

```
##   PassengerId       Survived          Pclass          Name          
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891        
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character  
##  Median :446.0   Median :0.0000   Median :3.000   Mode  :character  
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309                     
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000                     
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000                     
##                                                                     
##      Sex                 Age            SibSp           Parch       
##  Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000  
##  Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000  
##  Mode  :character   Median :28.00   Median :0.000   Median :0.0000  
```

```
##                    Mean   :29.70   Mean   :0.523   Mean   :0.3816
##                    3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                    Max.   :80.00   Max.   :8.000   Max.   :6.0000
##                    NA's   :177
##     Ticket              Fare            Cabin              Embarked
##  Length:891         Min.   :  0.00   Length:891         Length:891
##  Class :character   1st Qu.:  7.91   Class :character   Class :character
##  Mode  :character   Median : 14.45   Mode  :character   Mode  :character
##                     Mean   : 32.20
##                     3rd Qu.: 31.00
##                     Max.   :512.33
##
```

### Data Exploration, Cleaning, and Visualization

As we look at the missing value count table for Training Data, we see that there are 177 Age, 687 Cabin and 2 Embarked missing values.For the Test data, there are 86 Age, 1 Fare, 327 Cabin missing values. For the Cabin entries, most reasonable thing to do in a case like this is to remove the column because of the high number of NA values, the number of total rows is 891 for training, NA values of 687 is quite high and replacing it with a mode or median is not logical this case. Therefore the best case scenario is to remove the Cabin column.

For the missing Embarked entries in training set, I decided to use the most common Embarking location (mode) and replace the missing values with that, because I think this is the most reasonable choice to choose from the embarked locations.

For the missing fare entry in testing dataset, I decided to replace this NA value with the median value of fares. The 3rd quartile of the fare data is less than the mean, for this reason we can conclude that most of the fare values are less than the mean, therefore it is more logical to use the median instead of the mean.

In addition, I will replace the missing Age values with their mean.

```r
#finding Na values
colSums(is.na(titanic_train))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         177
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0         687           2
```

```r
#mode function
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}

# remove Cabin column
titanic_train <- titanic_train %>% select(-c("Cabin"))

#replace NA Embarked in training dataset with mode
mode_embarked_train = getmode(titanic_train$Embarked)
titanic_train$Embarked[is.na(titanic_train$Embarked)==TRUE] <- "S"
print(mode_embarked_train)
```

```
## [1] "S"
```

```r
#replace the NA Fare in dataset with median
titanic_train$Fare[is.na(titanic_train$Fare)] = median(titanic_train$Fare, na.rm = TRUE)

#replace all NA Age values with their mean
titanic_train$Age[is.na(titanic_train$Age)] = median(titanic_train$Age, na.rm = TRUE)
```

In addition to cleaning of the dataset, I believe adding additional attributes using the existing attributes is a good way to enhance the decision making process of our model. For this reason, I decided to add a column of age groups, which labels the entries regarding their age group into 5 categories such as children, teenager, young adult, middle_age, and old.

Also we can try to conclude the marrital_state of entries using their names: Master (referred to men younger than 18 - not married), Miss (referred to women younger than 18 - not married), Ms. (not married women), Mrs. (married women). The use of Mr is a little complicated. Since mr is referred to men who are older than 18, it is not for sure that these men are married or not. For this case, I will assume that all Mr's are married. All the other prefixes will be ignored and the marital_state will be set to unknown.

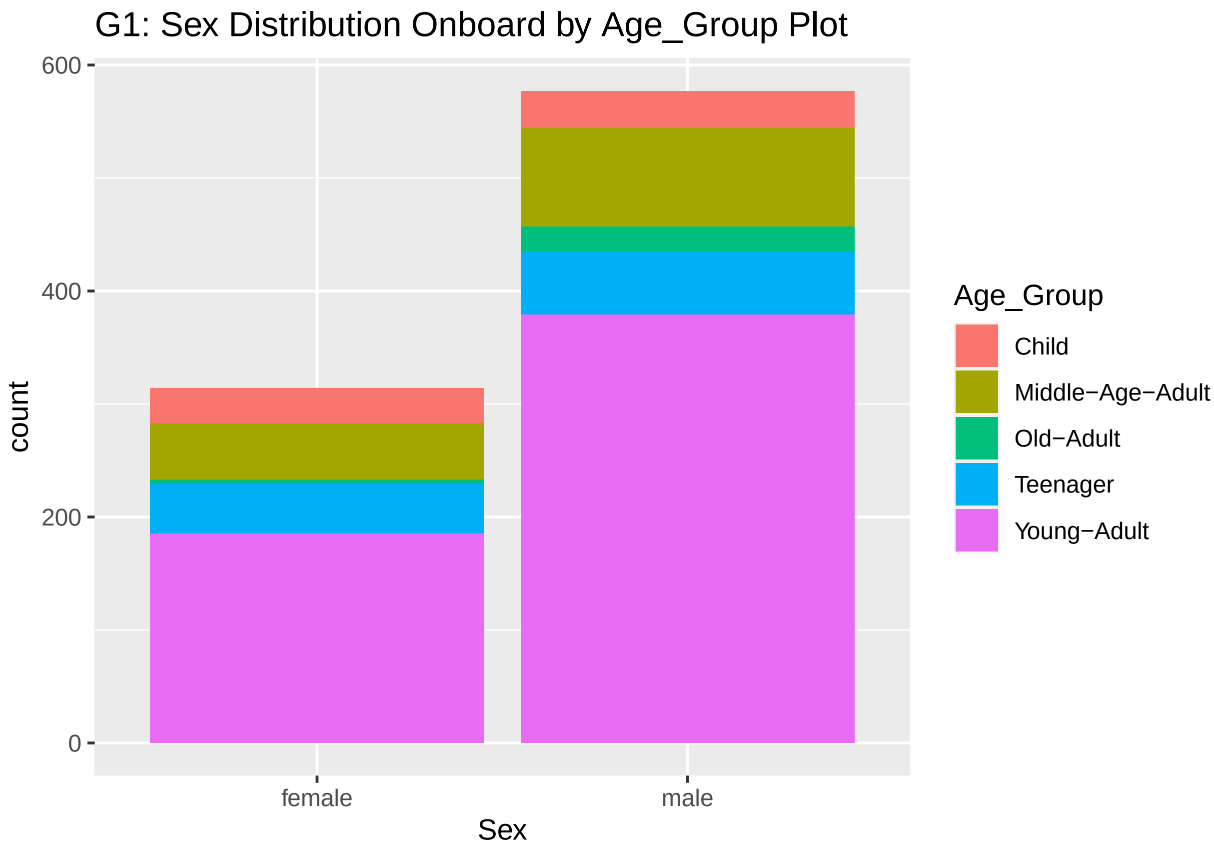
```r
# For Training Dataset
# Marital_Status attribute addition
titanic_train$Prefix <- str_extract(string = titanic_train$Name,
                                    pattern = "(Mr|Master|Mrs|Miss|Ms|Col|Don|Sir|Rev|Capt|Dr|Major|Lad
titanic_train$Prefix[is.na(titanic_train$Prefix)==TRUE] <- "unkown"

titanic_train$Marital_State <- ifelse(titanic_train$Prefix %in% c("Mr.", "Mrs.", "Countess."), "Married
                               ifelse(titanic_train$Prefix %in% c("Miss.", "Ms.", "Master."), "Not-Marr
                               "Unknown"))

# Age Grouping
titanic_train$Age_Group <- ifelse(titanic_train$Age<=10, "Child",
                           ifelse(titanic_train$Age>=10 & titanic_train$Age<20, "Teenager",
                           ifelse(titanic_train$Age>=20 & titanic_train$Age<40, "Young-Adult",
                           ifelse(titanic_train$Age>=40 & titanic_train$Age<60, "Middle-Age-Adult",
                           "Old-Adult"))))
```

Moreover, there are some irrelevant attributes in the dataset such as the parch which indicates the number of parents/children aboard, name and ticket number. These attributes are irrelevant to the model and therefore will be removed from the dataset.

```r
#remove parch, name and ticket attributes from dataset
titanic_train <- titanic_train %>% select(-c("Name", "Ticket", "Parch"))
```

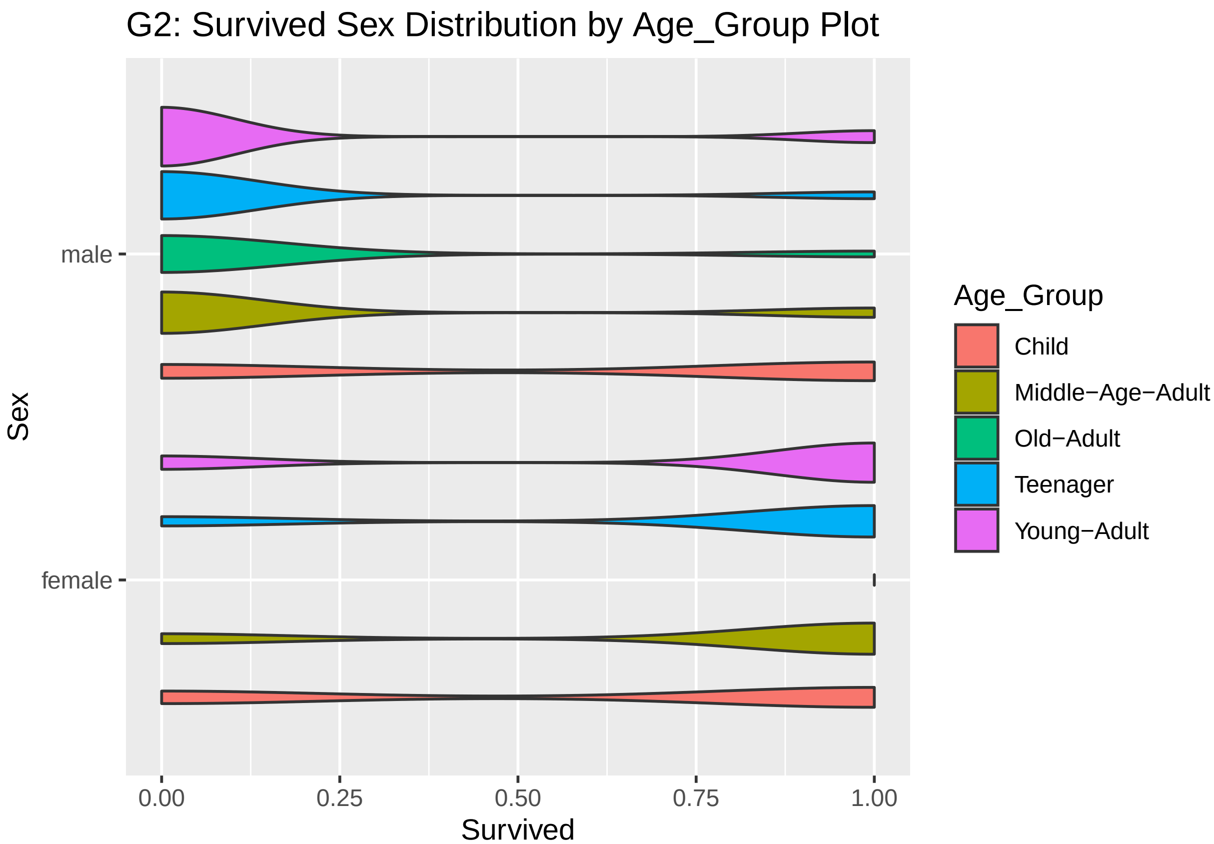
Now, lets try to visualize and analyze the relationships between pairs of variables.

```r
########### PLOTS
# sex vs survived stack bar graph (age_group)
g1 <- ggplot(titanic_train, aes(x= Sex, fill= Age_Group))
g1 + geom_bar(position="stack")+ ggtitle("G1: Sex Distribution Onboard by Age_Group Plot")
```



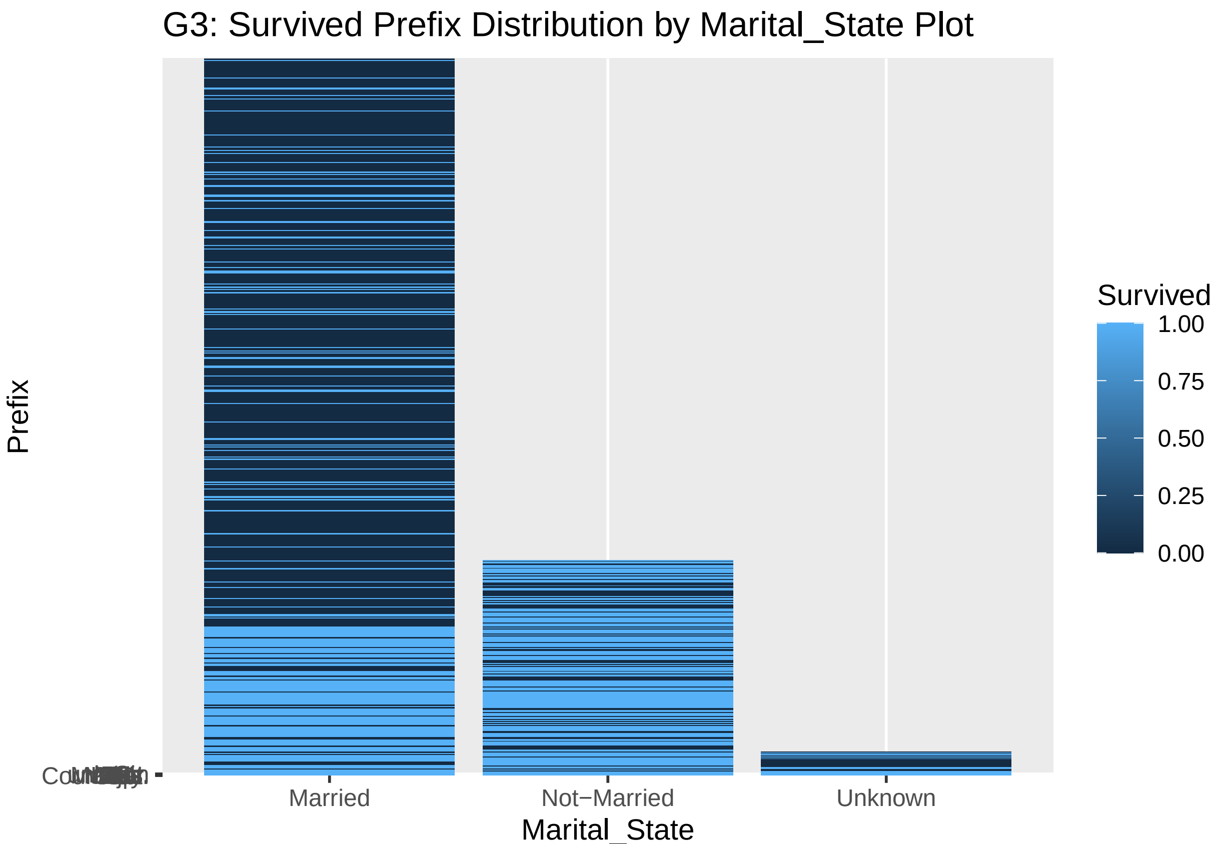

The graph above shows the sex distribution onbord the ship with a fill of age groups. Number of females is almost the half of number of male passangers. In both sexes, the number of young-adult (20<age<40) is higher than the sum of number of other age groups. Middle age adults of ages between 40 and 60 are higher in males than women. There close to 600 males and 300 women onboard Titanic.

```r
# age_group vs survived
g2 <- ggplot(titanic_train, aes(x= Survived, y= Sex, fill= Age_Group))
g2 + geom_violin()+ ggtitle("G2: Survived Sex Distribution by Age_Group Plot")
```




G2 plot is a violin graph. This graph type is very useful in our case since there are two opposite outcomes of survival (either 1 or 0). The G2 grpah shows the distubution of sexes by age groups. Starting with males, we see that the number of females that survived are greater than that of men. Most of the men couldn't survive. In addition, the highest number of survival is for women in young-adult, teenager, and middle-age-adult. As it was shown in the famous movie Titanic, when the emergency evacuations started, the first ones to be offloaded to espace vessels were women and children. In males, the highest number of survival is in children. The above table proves this point.

```
# prefix vs survived
g3 <- ggplot(titanic_train, aes(x= Marital_State, y= Prefix, fill= Survived))
g3 + geom_col()+ ggtitle("G3: Survived Prefix Distribution by Marital_State Plot")
```




G3 plot shows the Distribution of Prefix vs Marital_State grouped by Survival. Most number of deaths are shown in married people, and the highest number of survival is in not-married category. This is interesting since you would expect married men to secure a place on the boats for their wives, but this is not the case.

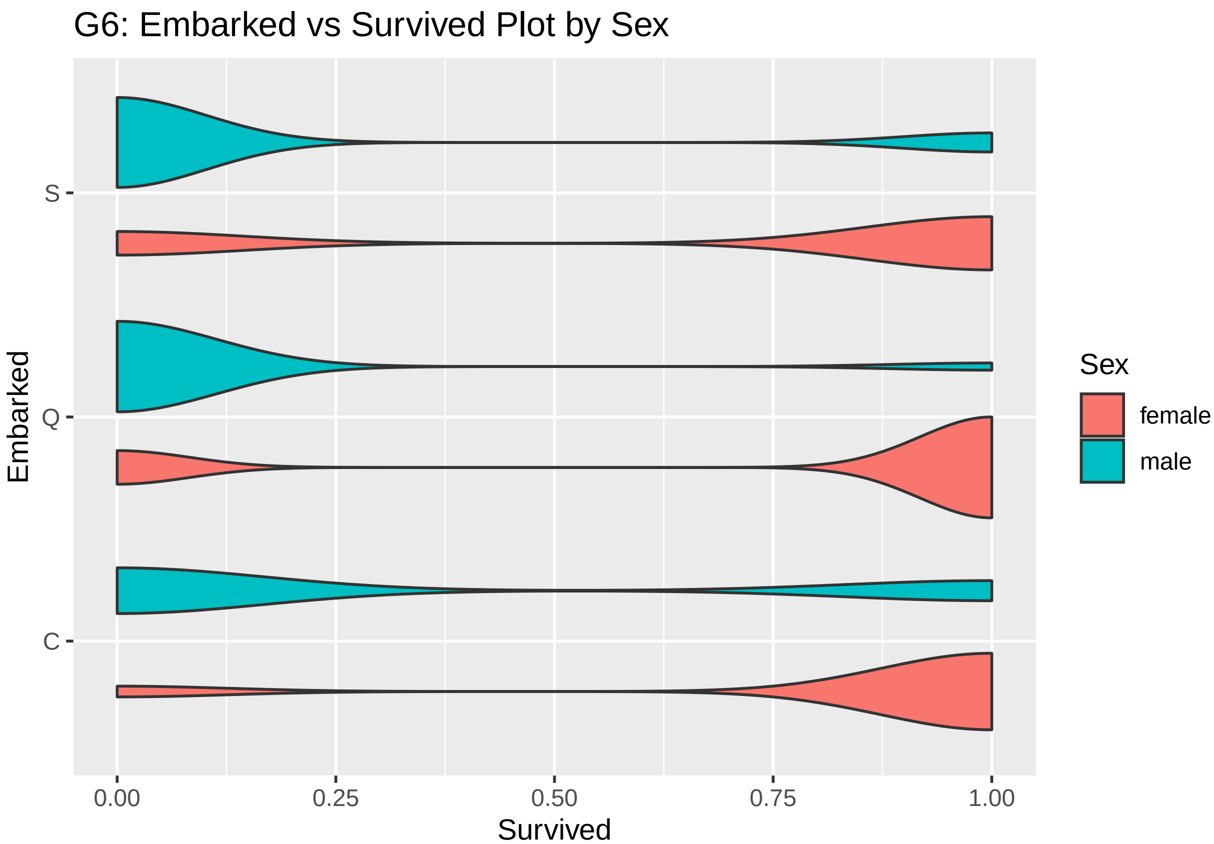
```r
# pclass vs survived (age_group)
g4 <- ggplot(titanic_train, aes(x= Pclass, y = Survived, fill= Age_Group))
g4 + geom_col()+ ggtitle("G4: Survived Pclass Distribution by Age_Group Plot")
```


Figure G4 shows the Survived Pclass distribution. Pclass is also an socio-economic measurement for the data. Pclass1 corresponds to 1st class. The highest number of people surviving is in Pclass 1 with majority are young-adults. This is the case in all Pclasses. The middle-age-adults are the second highest compared to number of other survuved classes.

```
# Fare vs survived stack bar (sex)
g5 <- ggplot(titanic_train, aes(x= Fare, y = Survived, fill = Sex))
g5 + geom_violin(position = "dodge")+ ggtitle("G5: Fare vs Survived Plot by Sex")+ facet_wrap(~Sex)
```


The violin graph of G5 shows the Fare vs Survived plot grouped by sex. This chart is insightful since it clearly shows that while women from all fare ranges have survived, only the men of fares between 200-300 have survived. Men from all fare levels have died, while the women of fares 150 and 350 couldn't survive. This is another example showing that most of the surviving people are women.

```
# embarked vs survived
g6 <- ggplot(titanic_train, aes(x= Survived, y = Embarked, fill = Sex))
g6 + geom_violin()+ ggtitle("G6: Embarked vs Survived Plot by Sex")
```




The final plot of G6 displays the Embarked locations and survival in a violin graph. Overall, the highest number of survivals have been from Q Embarkment, then C, and finally B. Most of the survivals are again women. Most non-survivals are from Embarkments S and Q. Least number of death of women and men is from Embarkment C.

### Data Preprocessing

In this part, I will create dummy variables for the categorical attributes such as: 1. Sex 2. Embarked 3. Marital_State 4. Age_Group 5. Prefix

```
########## Dummy variables: Sex, Embarked, Marutal_State, Age_Group, Prefix
library(caret)
############ For Training Dataset
#categorical values: Sex, Embarked, Martial_State, Age_Group, Prefix
#numerical values: PassangerId, Survived, Pclass, Age, SibSp, Fare
PassangerId = titanic_train$PassengerId
Survived = titanic_train$Survived
Pclass = titanic_train$Pclass
Age = titanic_train$Age
SibSp = titanic_train$SibSp
Fare = titanic_train$Fare

#dummyVars(~gender, data= ...) => dummies only gender
#dummyVars( gender ~., data = ...) => exludes gender
dummies <- dummyVars(~Sex + Embarked + Marital_State + Age_Group + Prefix, data = titanic_train)

titanic_train_dummy <- as.data.frame(predict(dummies, newdata = titanic_train))
```

```r
#head(titanic_train_dummy)

#insert #numerical values: PassangerId, Survived, Pclass, Age, SibSp, Fare back to titani_train_dummy
titanic_train_dummy$PassengerId = PassangerId
titanic_train_dummy$Survived = Survived
titanic_train_dummy$Pclass = Pclass
titanic_train_dummy$Age = Age
titanic_train_dummy$SibSp = SibSp
titanic_train_dummy$Fare = Fare

head(titanic_train_dummy)
```

```
##   Sexfemale Sexmale EmbarkedC EmbarkedQ EmbarkedS Marital_StateMarried
## 1         0       1         0         0         1                    1
## 2         1       0         1         0         0                    1
## 3         1       0         0         0         1                    0
## 4         1       0         0         0         1                    1
## 5         0       1         0         0         1                    1
## 6         0       1         0         1         0                    1
##   Marital_StateNot-Married Marital_StateUnknown Age_GroupChild
## 1                        0                    0              0
## 2                        0                    0              0
## 3                        1                    0              0
## 4                        0                    0              0
## 5                        0                    0              0
## 6                        0                    0              0
##   Age_GroupMiddle-Age-Adult Age_GroupOld-Adult Age_GroupTeenager
## 1                         0                  0                 0
## 2                         0                  0                 0
## 3                         0                  0                 0
## 4                         0                  0                 0
## 5                         0                  0                 0
## 6                         0                  0                 0
##   Age_GroupYoung-Adult PrefixCapt. PrefixCol. PrefixCountess. PrefixDon.
## 1                    1           0          0               0          0
## 2                    1           0          0               0          0
## 3                    1           0          0               0          0
## 4                    1           0          0               0          0
## 5                    1           0          0               0          0
## 6                    1           0          0               0          0
##   PrefixDr. PrefixLady. PrefixMajor. PrefixMaster. PrefixMiss. PrefixMr.
## 1         0           0            0             0           0         1
## 2         0           0            0             0           0         0
## 3         0           0            0             0           1         0
## 4         0           0            0             0           0         0
## 5         0           0            0             0           0         1
## 6         0           0            0             0           0         1
##   PrefixMrs. PrefixMs. PrefixRev. PrefixSir. Prefixunkown PassengerId Survived
## 1          0         0          0          0            0           1        0
## 2          1         0          0          0            0           2        1
## 3          0         0          0          0            0           3        1
## 4          1         0          0          0            0           4        1
## 5          0         0          0          0            0           5        0
```

```
## 6          0          0           0           0             0            6         0
##   Pclass Age SibSp    Fare
## 1      3  22     1  7.2500
## 2      1  38     1 71.2833
## 3      3  26     0  7.9250
## 4      1  35     1 53.1000
## 5      3  35     0  8.0500
## 6      3  28     0  8.4583
```

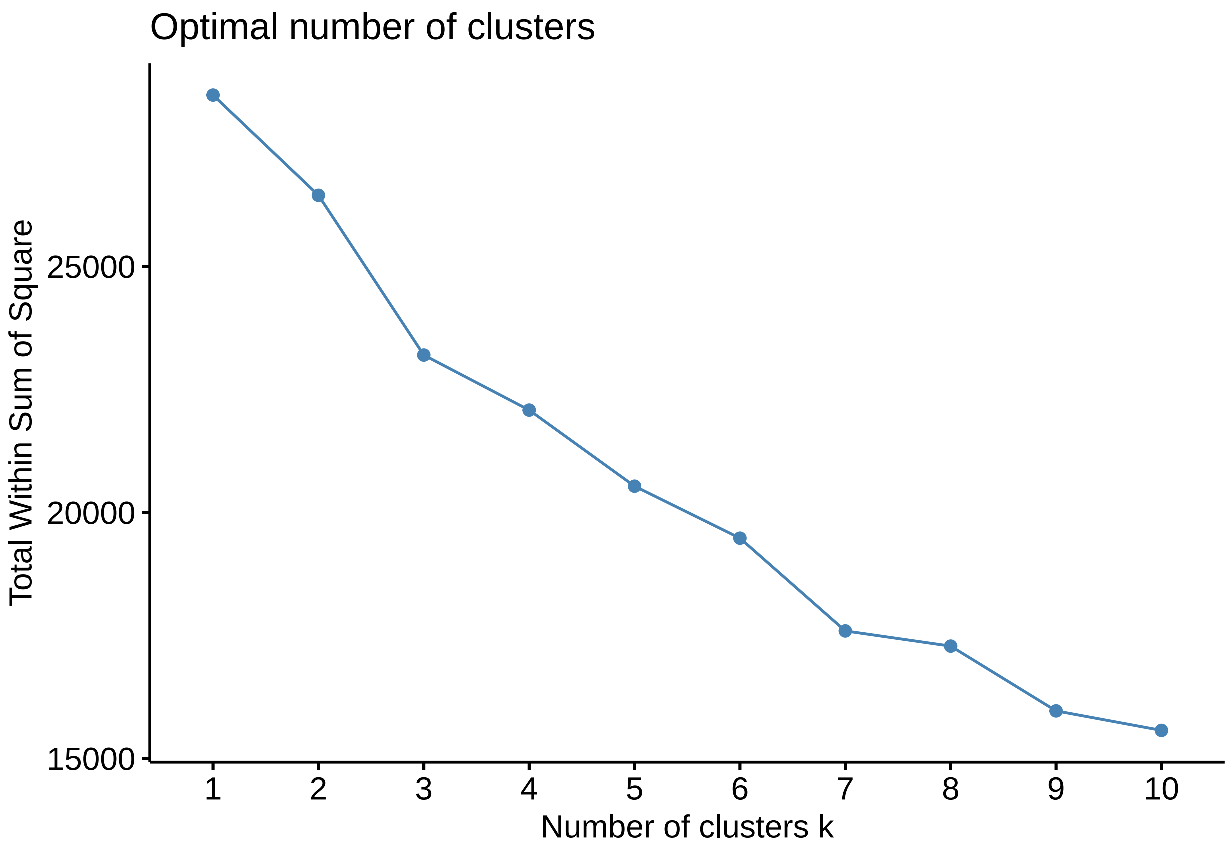
Now the whole data is numerical, and easier to work with, especially using PCA.

```
titanic_train_dummy <- titanic_train_dummy %>% select(-c("PassengerId"))
```

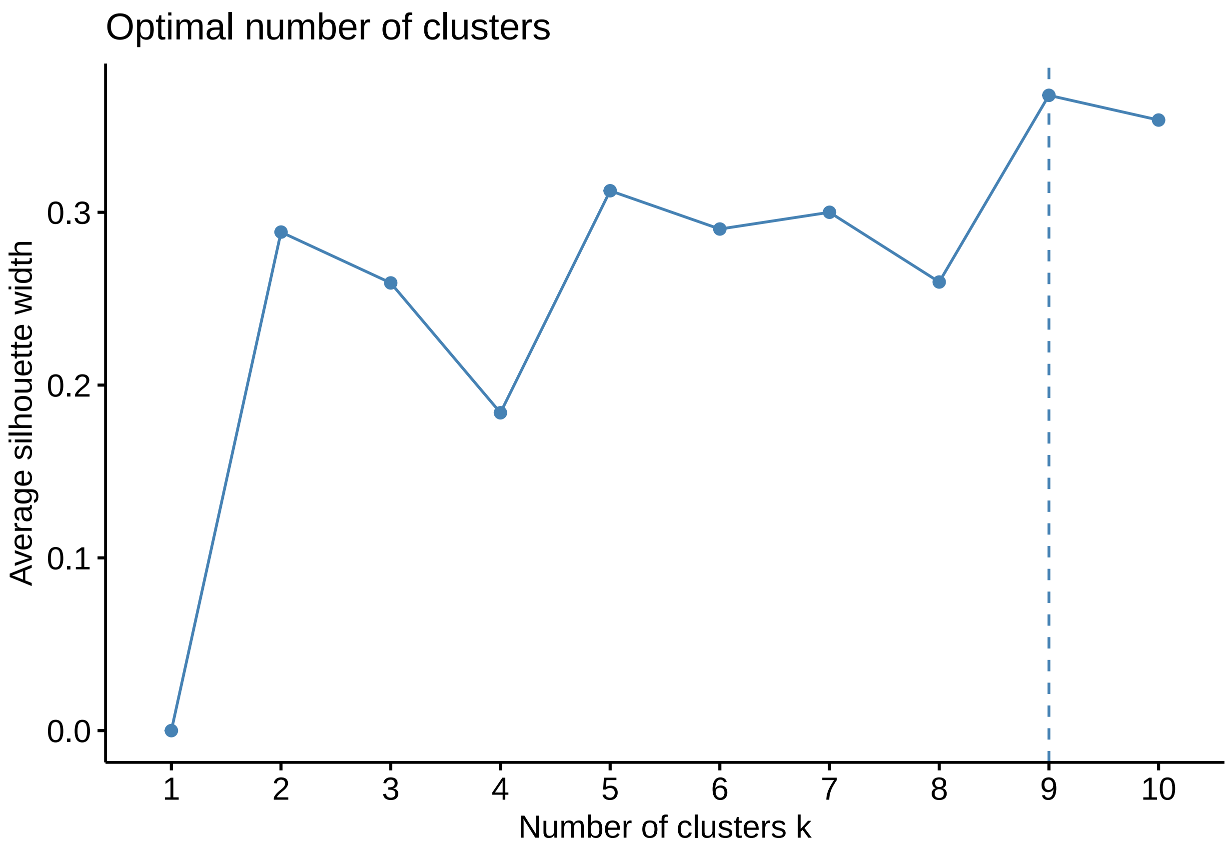
### Data Clustering using Kmeans and PCA

For the data clustering part of the assignment, I decided to use K means since the method is less computationally intensive and more suited for large datasets. I created both the kmeans plot to find the knee (best k value) and the silhouette scores. I decided to go with 9 clusters since it has the maximum silhouette score. After that I used PCA projection to color and visualize the points by cluster assignment based on Survival data. Survived 1 shows in blue and 0 shows red (Not survived).

```
########## KMEANS

predictors <- titanic_train_dummy %>% select(-c("Survived"))

#Normalize Data
# Center scale allows us to standardize the data
preproc_kmeans <- preProcess(predictors, method=c("center", "scale"))
# We have to call predict to fit our data based on preprocessing
predictors <- predict(preproc_kmeans, predictors)

# Find the knee
fviz_nbclust(predictors, kmeans, method = "wss")
```




```
# compare average silhouette scores of different K values
fviz_nbclust(predictors, kmeans, method = "silhouette")
```




```r
# Fit the data
fit <- kmeans(predictors, centers = 9, nstart = 25)
# Display the kmeans object information
fit
```

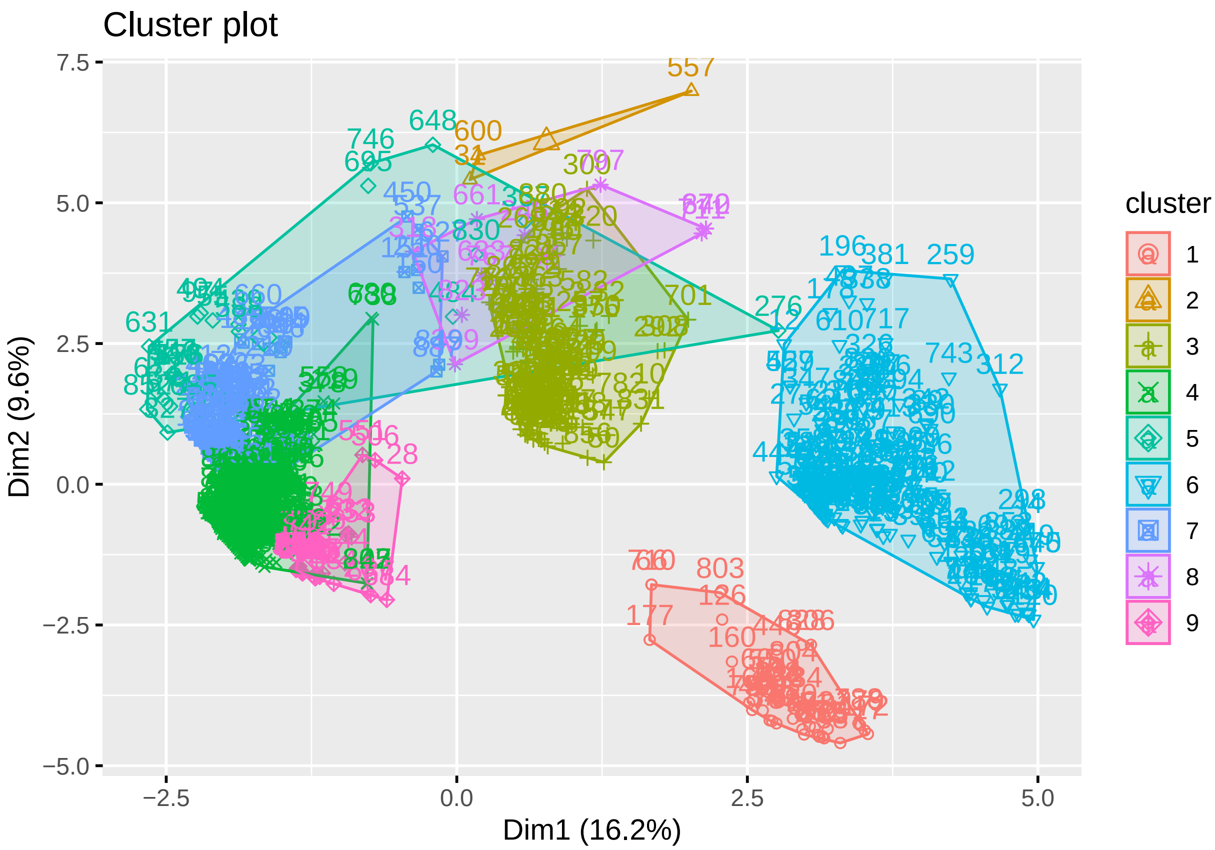
```
## K-means clustering with 9 clusters of sizes 40, 3, 123, 369, 27, 182, 83, 11, 53
##
## Cluster means:
##     Sexfemale     Sexmale   EmbarkedC   EmbarkedQ   EmbarkedS
## 1 -0.73728105  0.73728105 -0.16238301  0.04830410  0.11185638
## 2 -0.03991649  0.03991649  2.07334063 -0.30738970 -1.62289106
## 3  1.35481262 -1.35481262  0.14142613 -0.22063512  0.01495074
## 4 -0.73728105  0.73728105 -0.03860891 -0.01820775  0.04528115
## 5 -0.42734124  0.42734124  0.08603073 -0.04391281 -0.04773209
## 6  1.35481262 -1.35481262  0.05171298  0.33754960 -0.25772292
## 7 -0.73728105  0.73728105 -0.11235821 -0.17882568  0.21096555
## 8  0.02348029 -0.02348029  0.67964278  0.01596830 -0.60544388
## 9 -0.73728105  0.73728105 -0.24072373 -0.24027766  0.36209080
##   Marital_StateMarried Marital_StateNot-Married Marital_StateUnknown
## 1           -1.6092955                1.7297843           -0.1698115
## 2           -1.6092955               -0.5774579            5.8822719
## 3            0.6206925               -0.5774579           -0.1698115
## 4            0.6206925               -0.5774579           -0.1698115
## 5            0.2903239               -0.4920045            0.5026422
## 6           -1.6092955                1.7297843           -0.1698115
## 7            0.4057539               -0.5774579            0.4135218
## 8           -1.6092955               -0.5774579            5.8822719
```

```
## 9           0.6206925              -0.5774579          -0.1698115
##   Age_GroupChild Age_GroupMiddle-Age-Adult Age_GroupOld-Adult Age_GroupTeenager
## 1      2.9153082                -0.4260208         -0.1732745        -0.1178905
## 2     -0.2780311                 2.3446690         -0.1732745        -0.3553595
## 3     -0.2780311                 0.4074388         -0.1732745        -0.1236824
## 4     -0.2780311                -0.4260208         -0.1732745        -0.3553595
## 5     -0.2780311                -0.3234026          5.5447855        -0.3553595
## 6      0.3812664                -0.2585615         -0.1732745         0.2535353
## 7     -0.2780311                 2.2779054         -0.1732745        -0.3553595
## 8     -0.2780311                 0.5815028         -0.1732745        -0.3553595
## 9     -0.2780311                -0.4260208         -0.1732745         2.8108935
##   Age_GroupYoung-Adult PrefixCapt.  PrefixCol. PrefixCountess.  PrefixDon.
## 1         -1.105210221 -0.03350126 -0.04740458     -0.03350126 -0.03350126
## 2         -1.312568045 -0.03350126 -0.04740458     -0.03350126  9.91637311
## 3         -0.014474349 -0.03350126 -0.04740458      0.20917860 -0.03350126
## 4          0.761010196 -0.03350126 -0.04740458     -0.03350126 -0.03350126
## 5         -1.312568045  1.07204034  1.51694644     -0.03350126 -0.03350126
## 6         -0.116272906 -0.03350126 -0.04740458     -0.03350126 -0.03350126
## 7         -1.262602304 -0.03350126 -0.04740458     -0.03350126 -0.03350126
## 8          0.006981745 -0.03350126 -0.04740458     -0.03350126 -0.03350126
## 9         -1.312568045 -0.03350126 -0.04740458     -0.03350126 -0.03350126
##    PrefixDr. PrefixLady. PrefixMajor. PrefixMaster. PrefixMiss.  PrefixMr.
## 1 -0.0889363 -0.03350126  -0.04740458     4.6098940  -0.5063709 -1.1750751
## 2 -0.0889363  9.91637311  -0.04740458    -0.2166813  -0.5063709 -1.1750751
## 3 -0.0889363 -0.03350126  -0.04740458    -0.2166813  -0.5063709 -1.1750751
## 4 -0.0889363 -0.03350126  -0.04740458    -0.2166813  -0.5063709  0.8500543
## 5 -0.0889363 -0.03350126  -0.04740458    -0.2166813  -0.4145564  0.3250208
## 6 -0.0889363 -0.03350126  -0.04740458    -0.2166813   1.9589998 -1.1750751
## 7 -0.0889363 -0.03350126   0.46148069    -0.2166813  -0.5063709  0.6548611
## 8  7.1149037 -0.03350126  -0.04740458    -0.2166813  -0.5063709 -1.1750751
## 9 -0.0889363 -0.03350126  -0.04740458    -0.2166813  -0.5063709  0.8500543
##    PrefixMrs.   PrefixMs.  PrefixRev.  PrefixSir. Prefixunkown       Pclass
## 1 -0.40373535 -0.03350126 -0.08229248 -0.03350126  -0.06711573  0.378386445
## 2 -0.40373535 -0.03350126 -0.08229248  9.91637311  -0.06711573 -1.565227831
## 3  2.45069325 -0.03350126 -0.08229248 -0.03350126  -0.06711573 -0.369157507
## 4 -0.40373535 -0.03350126 -0.08229248 -0.03350126  -0.06711573  0.233739729
## 5 -0.08397695 -0.03350126 -0.08229248 -0.03350126  -0.06711573 -0.945043219
## 6 -0.40373535  0.13050766 -0.08229248 -0.03350126  -0.06711573  0.005435946
## 7 -0.40373535 -0.03350126  0.80111232 -0.03350126  -0.06711573 -0.412388965
## 8 -0.40373535 -0.03350126 -0.08229248 -0.03350126   5.36925863 -1.347760500
## 9 -0.40373535 -0.03350126 -0.08229248 -0.03350126  -0.06711573  0.420700254
##           Age       SibSp        Fare
## 1 -1.72391364  1.61142845  0.05028667
## 2  1.25233980  0.13027401  0.18538585
## 3  0.36812602  0.15976439  0.26223577
## 4 -0.08095384 -0.23098317 -0.18822975
## 5  2.71877894 -0.27276121  0.22072602
## 6 -0.50296070  0.16847378  0.22612146
## 7  1.34040532 -0.26669140 -0.07950637
## 8  0.46798041 -0.14452273  0.32570085
## 9 -0.91467203  0.02191077 -0.17855161
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
```

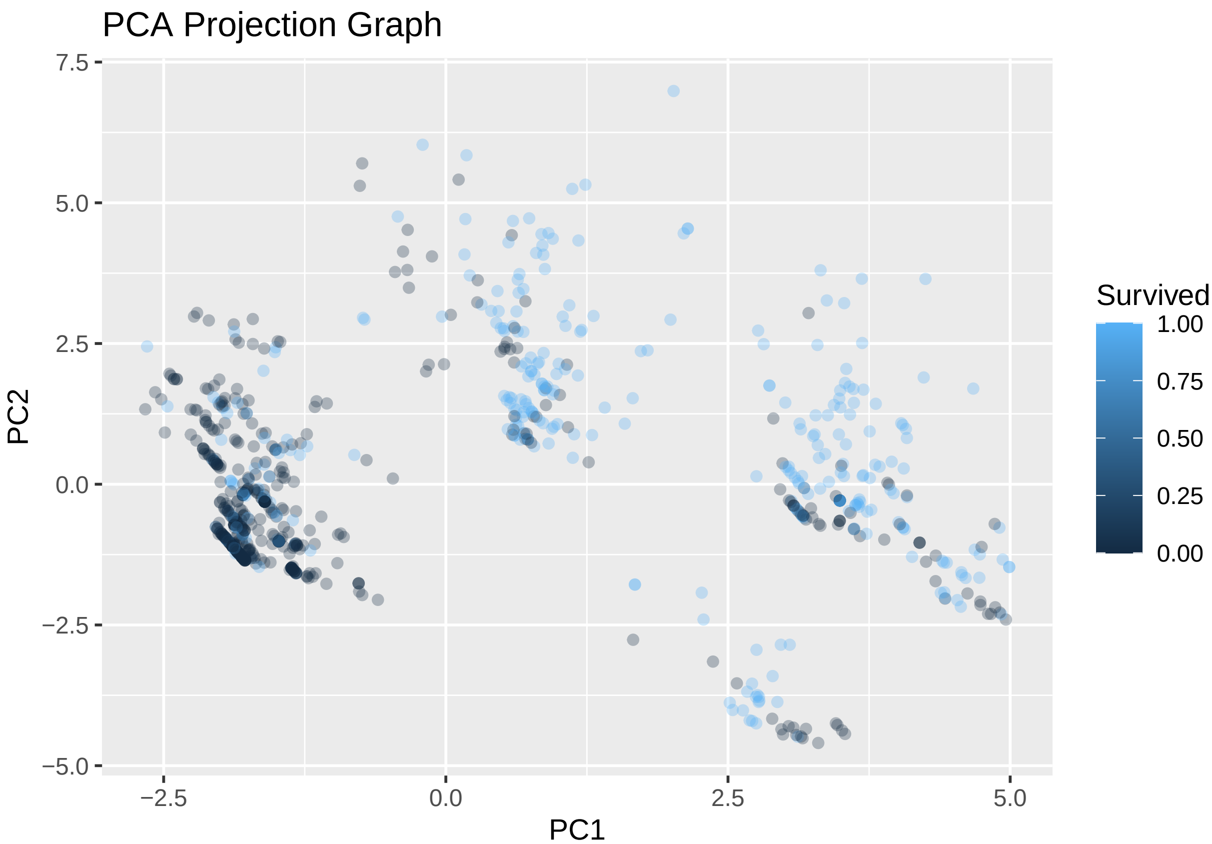
```
##   4   3   6   3   4   4   7   1   3   3   6   6   4   4   6   3   1   4   3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   4   4   6   4   6   3   4   9   6   4   2   3   6   5   4   7   4   4   6   6
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   4   6   6   4   4   6   4   3   1   4   3   3   5   4   6   4   6   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   4   6   7   1   4   1   3   9   6   4   4   6   4   4   4   4   4   4   1   6
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   4   4   6   4   6   3   9   4   6   4   4   4   7   4   7   4   5   4   3   4
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   6   4   4   4   4   4   6   4   4   6   7   6   4   6   6   4   5   4   4   6
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   4   4   4   6   7   1   4   4   6   7   4   4   3   3   4   4   6   4   9   4
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   3   6   3   9   9   9   4   6   4   7   7   3   7   7   4   7   6   4   4   1
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   7   3   4   9   1   1   3   3   4   4   5   1   6   4   7   9   1   6   4   4
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   6   4   1   1   6   4   3   7   7   4   3   9   6   1   3   6   4   7   6   6
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   4   4   4   7   9   6   4   4   6   7   4   6   4   4   4   6   6   7   6   4
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   9   4   7   4   4   4   9   4   9   6   3   4   7   6   4   6   7   6   9   4
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   6   6   4   4   4   8   6   3   4   7   4   3   5   4   3   3   3   6   6   3
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   4   1   7   7   6   4   9   4   3   6   4   4   3   4   6   5   6   4   1   3
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   5   4   9   9   4   4   4   4   7   6   6   3   4   6   4   4   4   6   4   3
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   6   4   9   6   4   1   6   3   4   6   6   6   3   4   7   6   3   8   6   3
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   4   4   6   3   4   6   5   3   3   6   6   7   4   9   3   4   4   6   7   7
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   1   6   4   4   4   6   6   3   1   7   4   4   9   4   4   4   6   6   6   6
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
##   7   4   3   4   4   4   5   3   6   8   4   9   9   4   6   3   6   4   4   9
## 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
##   6   6   4   3   4   9   1   6   4   6   4   4   4   6   3   4   6   7   8   3
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420
##   4   4   6   4   6   4   7   1   4   6   4   4   6   4   7   3   3   6   4   6
## 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440
##   4   4   4   3   9   4   3   6   4   4   4   3   3   9   7   6   6   3   5   4
## 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460
##   3   4   4   6   4   1   6   4   6   7   4   4   4   7   4   4   5   3   6   4
## 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
##   7   4   7   7   4   4   4   7   4   6   4   4   3   3   6   4   4   4   4   6
## 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500
##   1   4   7   5   4   6   3   7   4   1   4   4   7   5   4   4   6   4   3   4
## 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520
##   9   6   6   6   6   9   3   4   4   4   4   4   4   3   4   7   3   4   3   4
## 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540
##   6   4   4   3   4   7   6   4   4   4   6   4   9   3   6   6   7   6   4   6
## 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560
```

```
##   6   6   6   4   7   5   3   4   4   1   9   4   4   4   6   5   2   4   3   3
## 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580
##   4   7   4   4   6   4   9   3   4   4   5   3   4   6   9   9   6   3   3   4
## 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600
##   6   3   7   4   4   6   7   5   4   4   4   3   7   6   4   4   6   7   4   2
## 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
##   3   4   4   7   4   4   4   4   3   6   3   4   6   4   4   6   4   3   6   4
## 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640
##   4   7   4   4   4   5   7   6   4   4   5   7   8   4   6   6   4   4   3   4
## 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660
##   4   8   6   4   6   7   9   5   4   6   4   6   4   6   6   4   4   3   4   7
## 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680
##   8   7   7   4   4   4   4   4   7   3   3   4   5   4   4   9   4   6   3   4
## 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700
##   6   4   4   9   5   4   9   9   9   6   4   6   4   4   5   7   7   6   7   7
## 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720
##   3   4   6   4   4   4   3   7   6   1   8   4   7   4   7   9   6   6   4   4
## 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740
##   6   9   4   7   4   4   3   6   4   6   6   9   4   4   4   4   3   4   4   4
## 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760
##   4   4   6   4   4   5   9   6   9   4   6   1   4   4   3   1   4   9   4   3
## 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780
##   4   7   4   3   9   3   8   6   4   4   4   7   3   4   3   9   4   6   4   3
## 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800
##   6   3   4   4   4   4   6   1   1   7   4   9   6   4   4   4   8   3   4   3
## 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820
##   4   3   1   1   4   4   4   6   4   3   4   4   4   6   4   4   6   4   7   1
## 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840
##   3   4   8   3   1   4   4   1   4   5   3   1   4   4   9   6   4   4   4   4
## 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860
##   4   9   6   4   9   7   4   4   7   3   1   5   6   6   3   3   3   7   3   4
## 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880
##   7   4   3   6   4   3   6   4   4   1   4   3   4   7   3   6   4   9   4   3
## 881 882 883 884 885 886 887 888 889 890 891
##   3   4   6   4   4   3   7   6   6   4   4
##
## Within cluster sum of squares by cluster:
## [1]  404.7627 1788.8459 1936.6325 1930.2170 2064.9620 3537.7420 2368.2752
## [8] 1013.4165  186.2176
##  (between_SS / total_SS =  46.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
# Display the cluster plot
fviz_cluster(fit, data = predictors)
```




```
# Calculate PCA
pca = prcomp(predictors)
# Save as dataframe
rotated_data = as.data.frame(pca$x)
# Add original labels as a reference
rotated_data$Survived <- titanic_train_dummy$Survived
# Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Survived)) + geom_point(alpha = 0.3) + ggtitle(
```




**Data Classification using SVM and Decision Tree Models**

```
######################### SVM - works only with numerical variables (will use titanic_train_dummy)

titanic_train_dummy$Survived = as.factor(titanic_train_dummy$Survived)
#Evaluation method parameter
train_control = trainControl(method = "cv", number = 10)
# Scaling method
preproc_svm = c("center", "scale")
#Grid search
grid <- expand.grid(C= 10^seq(-5,2,0.5))

# Fit the model
svm <- train(Survived ~., data = titanic_train_dummy,
             method = "svmLinear", trControl = train_control, tuneGrid = grid)

svm
```

```
## Support Vector Machines with Linear Kernel
##
## 891 samples
##  32 predictor
##   2 classes: '0', '1'
##
## No pre-processing
```

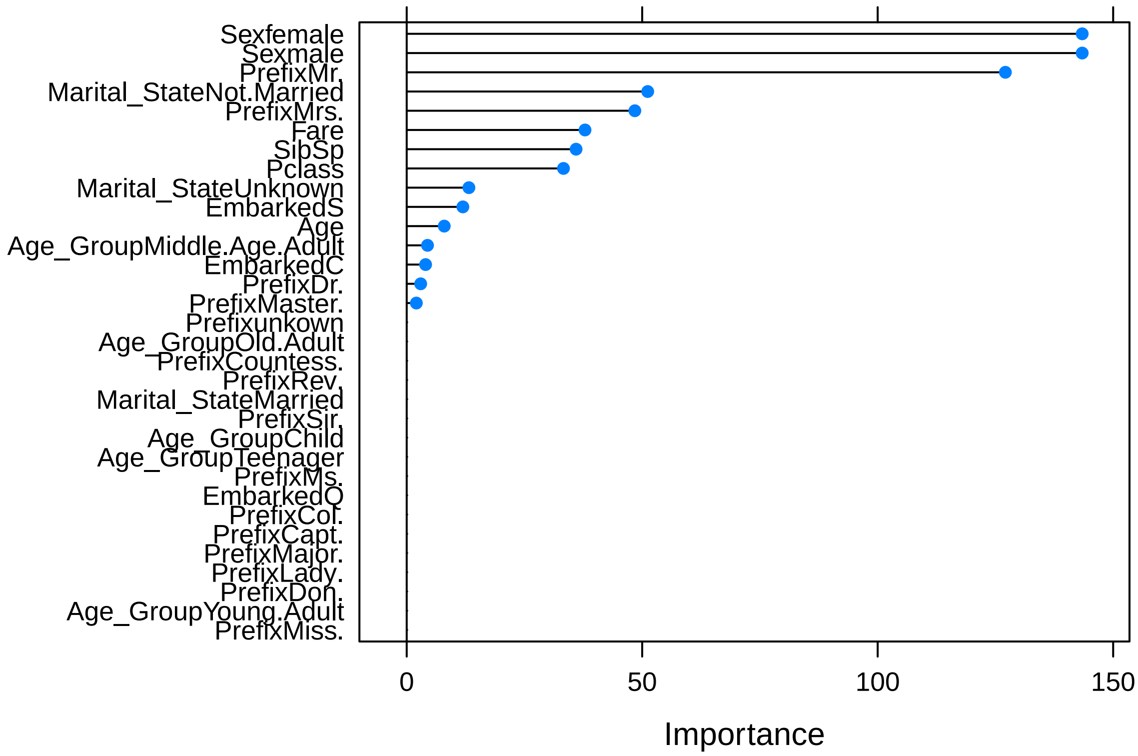
```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 801, 802, 803, 802, 801, 802, ...
## Resampling results across tuning parameters:
##
##   C             Accuracy   Kappa
##   1.000000e-05  0.6375695  0.06964330
##   3.162278e-05  0.6398167  0.08180914
##   1.000000e-04  0.6432130  0.09179074
##   3.162278e-04  0.6745988  0.19396100
##   1.000000e-03  0.7529636  0.44298553
##   3.162278e-03  0.7934650  0.55933141
##   1.000000e-02  0.8024166  0.57873228
##   3.162278e-02  0.8169984  0.60845637
##   1.000000e-01  0.8237910  0.62396170
##   3.162278e-01  0.8260257  0.62803964
##   1.000000e+00  0.8237785  0.62355056
##   3.162278e+00  0.8249021  0.62607640
##   1.000000e+01  0.8091462  0.59410212
##   3.162278e+01  0.8125170  0.60203082
##   1.000000e+02  0.8147898  0.60543895
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.3162278.
```

```
################### Decision Tree

# Make Valid Column Names
colnames(titanic_train_dummy) <- make.names(colnames(titanic_train_dummy))

# First lets check the relevance score of the decision tree

# BASE MODEL - Tree1: Fit the model
tree_base <- train(Survived ~., data = titanic_train_dummy,
                   method = "rpart1SE", trControl = train_control)
# View the variable importance scores
var_imp <- varImp(tree_base, scale = FALSE)
# Estimate variable importance
importance <- varImp(tree_base, scale=FALSE)
# Summarize importance
print(importance)
```

```
## rpart1SE variable importance
##
##   only 20 most important variables shown (out of 32)
##
##                             Overall
## Sexmale                     143.429
## Sexfemale                   143.429
## PrefixMr.                   127.119
## Marital_StateNot.Married     51.172
## PrefixMrs.                   48.445
## Fare                         37.858
## SibSp                        35.963
## Pclass                       33.289
```

```
## Marital_StateUnknown        13.212
## EmbarkedS                   11.927
## Age                          7.974
## Age_GroupMiddle.Age.Adult    4.423
## EmbarkedC                    4.014
## PrefixDr.                    2.971
## PrefixMaster.                2.051
## PrefixMiss.                  0.000
## PrefixCol.                   0.000
## PrefixLady.                  0.000
## Age_GroupOld.Adult           0.000
## PrefixDon.                   0.000
```

```r
# Visualize
plot(importance)
```




```r
# Using the scores from variable importance analysis, we can reduce the size of our table and only keep
decision_train_data <- titanic_train_dummy %>% select("Sexmale","Sexfemale","PrefixMr.","Marital_StateN
                                                       "SibSp","Pclass","Marital_StateUnknown","Embarked
                                                       "Age","Age_GroupMiddle.Age.Adult","EmbarkedC","Pr
head(decision_train_data)
```

```
##   Sexmale Sexfemale PrefixMr. Marital_StateNot.Married PrefixMrs.    Fare SibSp
## 1       1         0         1                        0          0  7.2500     1
## 2       0         1         0                        0          1 71.2833     1
## 3       0         1         0                        1          0  7.9250     0
## 4       0         1         0                        0          1 53.1000     1
## 5       1         0         1                        0          0  8.0500     0
## 6       1         0         1                        0          0  8.4583     0
```

```
##   Pclass Marital_StateUnknown EmbarkedS Age Age_GroupMiddle.Age.Adult EmbarkedC
## 1      3                    0         1  22                         0         0
## 2      1                    0         0  38                         0         1
## 3      3                    0         1  26                         0         0
## 4      1                    0         1  35                         0         0
## 5      3                    0         1  35                         0         0
## 6      3                    0         0  28                         0         0
##   PrefixDr. PrefixMaster. Survived
## 1         0             0        0
## 2         0             0        1
## 3         0             0        1
## 4         0             0        1
## 5         0             0        0
## 6         0             0        0
```

```r
# I will use the decision_train_data dataset to create the decision tree model
set.seed(123)

# General Model Comparison & Visualization: I will create 10 different trees with different hyper param

# Partition the data
index_decision = createDataPartition(y=decision_train_data$Survived, p=0.7, list=FALSE)

# Everything in the generated index list
train_set_q3 = decision_train_data[index_decision,]

# Everything except the generated indices
test_set_q3 = decision_train_data[-index_decision,]
# Initialize cross validation
train_control = trainControl(method = "cv", number = 10)

# Tree 1
hypers = rpart.control(minsplit = 5, maxdepth = 1, minbucket = 5)
tree1 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree1, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree1$finalModel$frame)
```

```r
# Form the table
comp_tbl <- data.frame("Nodes" = nodes, "TrainAccuracy" = a_train,
                       "TestAccuracy" = a_test, "MaxDepth" = 1, "Minsplit" = 5, "Minbucket" = 5)

######################

# Tree 2
hypers = rpart.control(minsplit = 10, maxdepth = 2, minbucket = 10)
tree2 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree2, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree2, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree2$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 2, 10, 10))

######################

# Tree 3
hypers = rpart.control(minsplit = 30, maxdepth = 3, minbucket = 30)
tree3 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree3, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree3, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)
cfm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 143  21
##          1  32  70
##
##                Accuracy : 0.8008
##                  95% CI : (0.7476, 0.847)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 2.11e-07
##
##                   Kappa : 0.5698
##
##  Mcnemar's Test P-Value : 0.1696
##
##             Sensitivity : 0.8171
##             Specificity : 0.7692
##          Pos Pred Value : 0.8720
##          Neg Pred Value : 0.6863
##              Prevalence : 0.6579
##          Detection Rate : 0.5376
##    Detection Prevalence : 0.6165
##       Balanced Accuracy : 0.7932
##
##        'Positive' Class : 0
##
```

```
# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree3$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 30, 30))

######################
# Tree 4
hypers = rpart.control(minsplit = 50, maxdepth = 3, minbucket = 50)
tree4 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree4, test_set_q3)
# Confusion Matrix
```

```r
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree4$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 50, 50))

######################
# Tree 5
hypers = rpart.control(minsplit = 100, maxdepth = 3, minbucket = 100)
tree5 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree5, train_set_q3)
# Confusion Matrix
cfm_train_4 <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree5, test_set_q3)
# Confusion Matrix
cfm_test_4 <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train_4$overall[1]
# Get testing accuracy
a_test <- cfm_test_4$overall[1]
# Get number of nodes
nodes <- nrow(tree5$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 3, 100, 100))

######################
# Tree 6
hypers = rpart.control(minsplit = 100, maxdepth = 4, minbucket = 100)
tree6 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree6, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
```

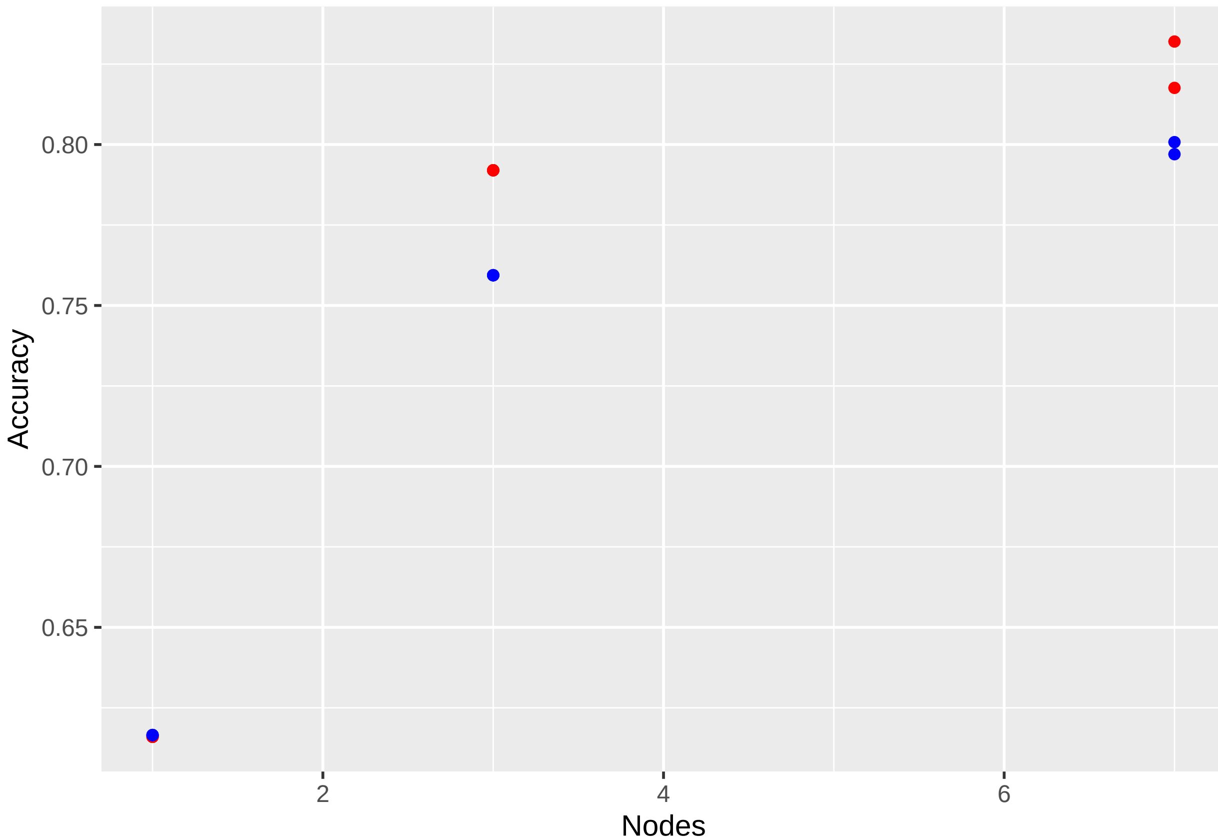
```
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree6, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree6$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 100, 100))

######################
# Tree 7
hypers = rpart.control(minsplit = 1000, maxdepth = 4, minbucket = 1000)
tree7 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree7, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree7, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree7$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 4, 1000, 1000))

######################
# Tree 8
hypers = rpart.control(minsplit = 1000, maxdepth = 5, minbucket = 1000)
tree8 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree8, train_set_q3)
# Confusion Matrix
```

```r
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree8, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree8$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 5, 1000, 1000))

######################
# Tree 9
hypers = rpart.control(minsplit = 3000, maxdepth = 6, minbucket = 3000)
tree9 <- train(Survived ~., data = train_set_q3, control = hypers,
               trControl = train_control, method = "rpart1SE")

# Training Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree9, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree9, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree9$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 6, 3000, 3000))

######################
# Tree 10
hypers = rpart.control(minsplit = 5000, maxdepth = 7, minbucket = 5000)
tree10 <- train(Survived ~., data = train_set_q3, control = hypers,
                trControl = train_control, method = "rpart1SE")

# Training Set
```

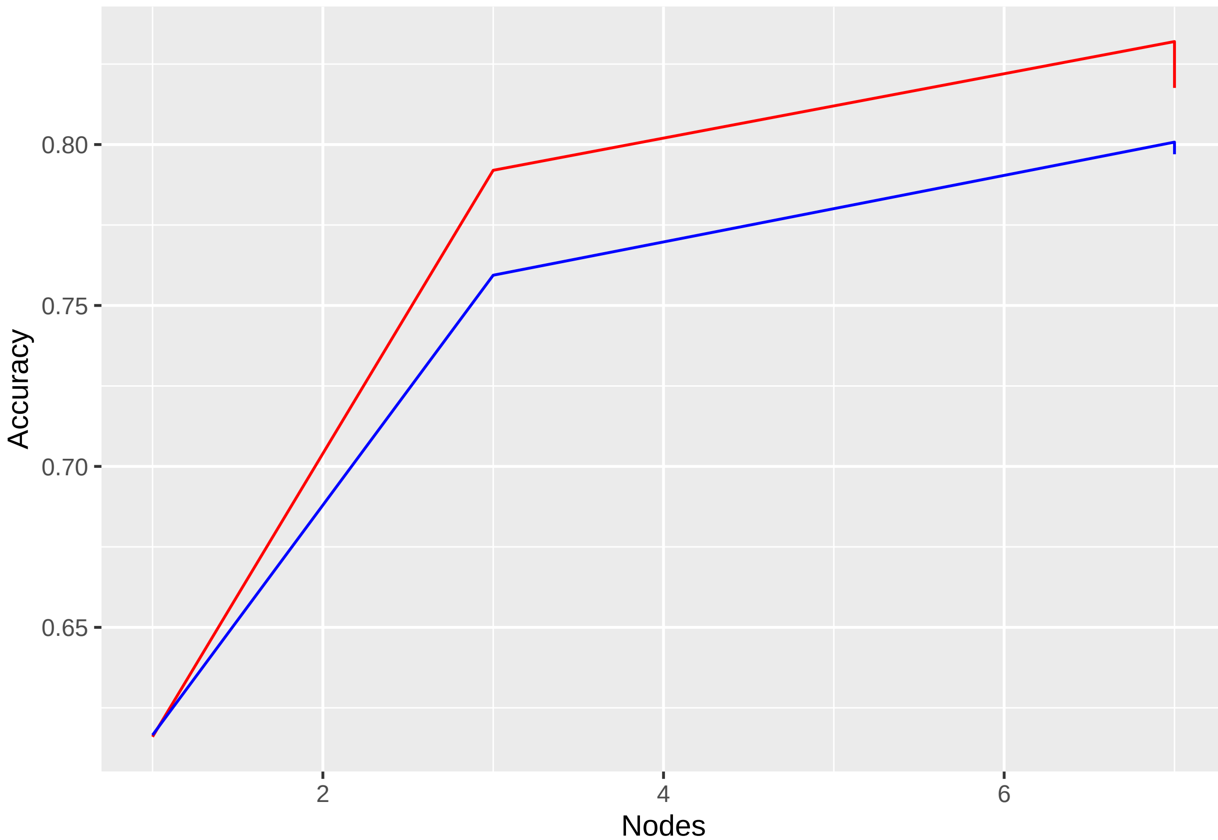
```r
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree10, train_set_q3)
# Confusion Matrix
cfm_train <- confusionMatrix(train_set_q3$Survived, pred_tree)

# Test Set
# Evaluate the fit with a confusion matrix
pred_tree <- predict(tree10, test_set_q3)
# Confusion Matrix
cfm_test <- confusionMatrix(test_set_q3$Survived, pred_tree)

# Get training accuracy
a_train <- cfm_train$overall[1]
# Get testing accuracy
a_test <- cfm_test$overall[1]
# Get number of nodes
nodes <- nrow(tree10$finalModel$frame)

# Add rows to the table - Make sure the order is correct
comp_tbl <- comp_tbl %>% rbind(list(nodes, a_train, a_test, 7, 5000, 5000))

##################### VISUALS
#table display
comp_tbl
```

```
##          Nodes TrainAccuracy TestAccuracy MaxDepth Minsplit Minbucket
## Accuracy     3        0.7920    0.7593985        1        5         5
## 1            3        0.7920    0.7593985        2       10        10
## 11           7        0.8320    0.8007519        3       30        30
## 12           7        0.8176    0.7969925        3       50        50
## 13           3        0.7920    0.7593985        3      100       100
## 14           3        0.7920    0.7593985        4      100       100
## 15           1        0.6160    0.6165414        4     1000      1000
## 16           1        0.6160    0.6165414        5     1000      1000
## 17           1        0.6160    0.6165414        6     3000      3000
## 18           1        0.6160    0.6165414        7     5000      5000
```

```r
# Visualize with scatter plot
ggplot(comp_tbl, aes(x=Nodes)) + geom_point(aes(y = TrainAccuracy), color = "red") +
  geom_point(aes(y = TestAccuracy), color="blue") + ylab("Accuracy")
```




```
# Visualize with line plot
ggplot(comp_tbl, aes(x=Nodes)) + geom_line(aes(y = TrainAccuracy), color = "red") +
  geom_line(aes(y = TestAccuracy), color="blue") + ylab("Accuracy")
```




With the SVM, we get the C = 0.1, Accuracy = 0.8249146 and Kappa = 0.62615970.

With the optimized decision tree, we get the maximum accuracy with 7 nodes, TrainAccuracy and TestAccuracy of 0.8320, 0.8007519 respectively, using MaxDepth of 3, Minsplit of 30, and MinBucket of 30.

In comparison, we see that the decision tree with 7 nodes, MaxDept =3, Minsplit=30, and MinBucket=30 has a greater training accuracy then SVM model. The Decision tree has a Kappa value of 0.5698 which is lower than that of SVM. However, it is important to state that the accuracy of both models are close. In the next section, I will use a more advanced classifier and try to enhance the model accuracy.

**Data Evaluation using kNN**

```
################### Classifier: kNN (using the tools of Week 9)

# Check target class and make sure it has 2 levels -  Correct: Survived (0,1)
str(titanic_train_dummy$Survived)
```

```
##  Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
```

```
set.seed(123)

# Partition the data
index = createDataPartition(y=titanic_train_dummy$Survived, p=0.7, list=FALSE)
# Everything in the generated index list
train_knn = titanic_train_dummy[index,]
# Everything except the generated indices
```

```r
test_knn = titanic_train_dummy[-index,]

# Set control parameter
train_control = trainControl(method = "cv", number = 10)

# setup a tuneGrid with the tuning parameters
# data has to be scaled because the distance measurements are sensitive

tuneGrid <- expand.grid(kmax = 3:7, kernel = c("rectangular", "cos"),
                        distance = 1:3) #powers of Minkowski 1 to 3

kknn_fit <- train(Survived ~ ., data = train_knn, method = 'kknn',
                  trControl = train_control, preProcess = c('center', 'scale'),
                  tuneGrid = tuneGrid)
kknn_fit
```

```
## k-Nearest Neighbors
##
## 625 samples
##  32 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (32), scaled (32)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 563, 562, 563, 563, 563, 562, ...
## Resampling results across tuning parameters:
##
##   kmax  kernel       distance  Accuracy   Kappa
##   3     rectangular  1         0.8240143  0.6248186
##   3     rectangular  2         0.8223758  0.6213521
##   3     rectangular  3         0.8064004  0.5863740
##   3     cos          1         0.8016385  0.5789039
##   3     cos          2         0.7952125  0.5662436
##   3     cos          3         0.7984639  0.5738614
##   4     rectangular  1         0.8240143  0.6248186
##   4     rectangular  2         0.8239887  0.6247697
##   4     rectangular  3         0.8191500  0.6148195
##   4     cos          1         0.8112391  0.5993842
##   4     cos          2         0.8064004  0.5887162
##   4     cos          3         0.8000768  0.5748126
##   5     rectangular  1         0.8192012  0.6110228
##   5     rectangular  2         0.8176395  0.6064434
##   5     rectangular  3         0.8128008  0.5958320
##   5     cos          1         0.8096262  0.5952021
##   5     cos          2         0.8032258  0.5813571
##   5     cos          3         0.8000768  0.5740969
##   6     rectangular  1         0.8257808  0.6233774
##   6     rectangular  2         0.8177163  0.6062648
##   6     rectangular  3         0.8128776  0.5956534
##   6     cos          1         0.8160522  0.6090814
##   6     cos          2         0.8128008  0.6012507
##   6     cos          3         0.8143881  0.6048088
##   7     rectangular  1         0.8241423  0.6200595
```

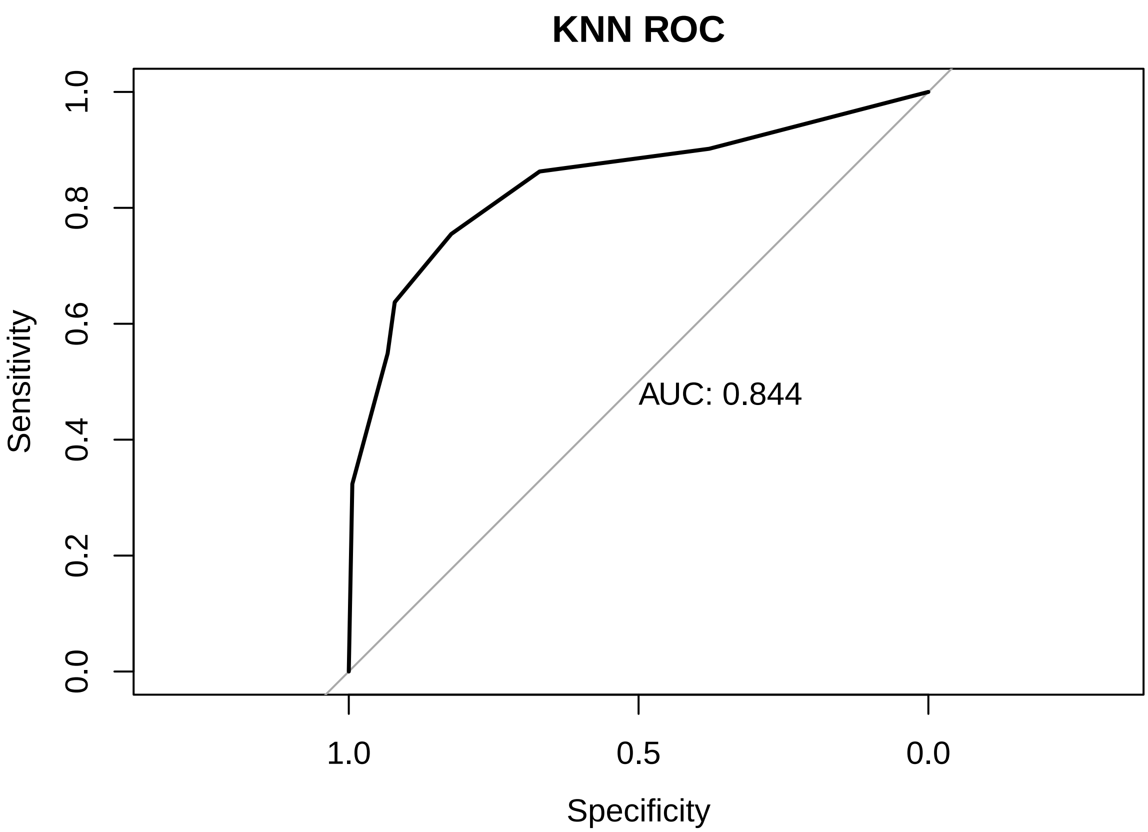
```
##   7     rectangular  2         0.8177163  0.6052179
##   7     rectangular  3         0.8128776  0.5946065
##   7     cos          1         0.8224782  0.6210140
##   7     cos          2         0.8256528  0.6267409
##   7     cos          3         0.8224014  0.6208904
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 6, distance = 1 and kernel
##  = rectangular.
```

```r
# Evaluate the fit with a confusion matrix
pred_knn <- predict(kknn_fit, test_knn)
# Confusion Matrix
confusionMatrix_kknn <- confusionMatrix(test_knn$Survived, pred_knn)
confusionMatrix_kknn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 140  24
##          1  31  71
##
##                Accuracy : 0.7932
##                  95% CI : (0.7395, 0.8403)
##     No Information Rate : 0.6429
##     P-Value [Acc > NIR] : 6.919e-08
##
##                   Kappa : 0.557
##
##  Mcnemar's Test P-Value : 0.4185
##
##             Sensitivity : 0.8187
##             Specificity : 0.7474
##          Pos Pred Value : 0.8537
##          Neg Pred Value : 0.6961
##              Prevalence : 0.6429
##          Detection Rate : 0.5263
##    Detection Prevalence : 0.6165
##       Balanced Accuracy : 0.7830
##
##        'Positive' Class : 0
##
```

```r
# Store the byClass object of confusion matrix as a dataframe
metrics <- as.data.frame(confusionMatrix_kknn$byClass)

########## ROC
# Get class probabilities for KNN
pred_prob <- predict(kknn_fit, test_knn, type = "prob")

# And now we can create an ROC curve for our model.
roc_obj <- roc((test_knn$Survived), pred_prob[,1])
```

```r
plot(roc_obj, print.auc=TRUE, main= "KNN ROC")
```



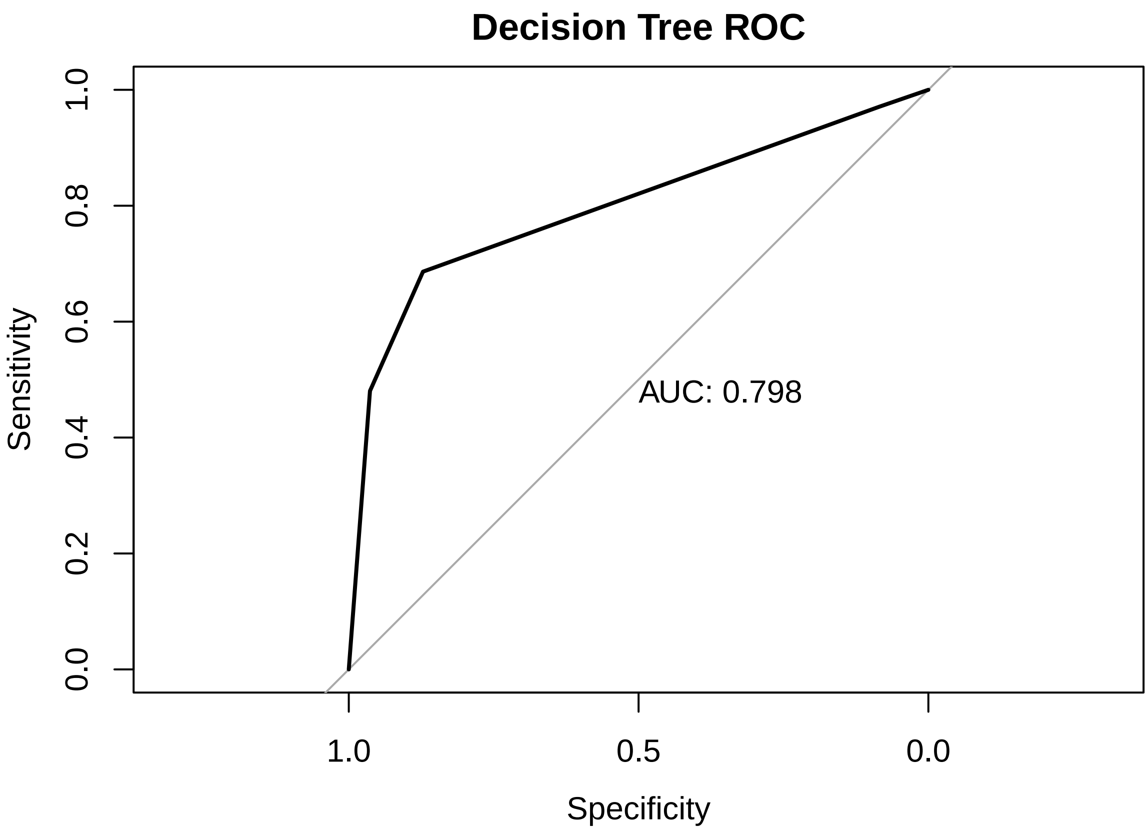

```r
# Now let's compare our knn model with a model of decision
# tree using Area Under the Curve metric from ROC Curve

# I will use the best decision tree from the previous section with k - 3, maxdepth and minsplit = 30.
tree3
```

```
## CART
##
## 625 samples
##  15 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 562, 562, 562, 563, 563, 562, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8126472  0.5938749
```

```r
# Evaluate the fit with a confusion matrix
pred_pima2 <- predict(tree3, test_knn)
# Confusion Matrix
confusionMatrix(test_knn$Survived, pred_pima2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 143  21
##          1  32  70
##
##                Accuracy : 0.8008
##                  95% CI : (0.7476, 0.847)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 2.11e-07
##
##                   Kappa : 0.5698
##
##  Mcnemar's Test P-Value : 0.1696
##
##             Sensitivity : 0.8171
##             Specificity : 0.7692
##          Pos Pred Value : 0.8720
##          Neg Pred Value : 0.6863
##              Prevalence : 0.6579
##          Detection Rate : 0.5376
##    Detection Prevalence : 0.6165
##       Balanced Accuracy : 0.7932
##
##        'Positive' Class : 0
##
```

```r
# Get class probabilities for decision tree model
pred_prob2 <- predict(tree3, test_knn, type = "prob")

# And now we can create an ROC curve for our model.
roc_obj2 <- roc((test_knn$Survived), pred_prob2[,1])

plot(roc_obj2, print.auc=TRUE, main= "Decision Tree ROC")
```




```r
# Getting Scoring Metrics of KNN model
metrics
```

```
##                        confusionMatrix_kknn$byClass
## Sensitivity                               0.8187135
## Specificity                               0.7473684
## Pos Pred Value                            0.8536585
## Neg Pred Value                            0.6960784
## Precision                                 0.8536585
## Recall                                    0.8187135
## F1                                        0.8358209
## Prevalence                                0.6428571
## Detection Rate                            0.5263158
## Detection Prevalence                      0.6165414
## Balanced Accuracy                         0.7830409
```

Using grid tuning I was able to find the maximum accuracy of using Knn with kmax = 6, distance = 1 and kernel = rectangular. Accuracy reported is 0.7932 with a Kappa value of 0.557. Even though the accuracy of kNN model is lower than optimized SVM and Decision Tree models in previous section, the ROC curve comparison shows that kNN model has an AUC of 0.844, while the best decision tree model that was trained in previous section has an AUC of 0.798. This is a clear indication that the kNN model is a superior model even though it has a slightly less accuracy compared to the best decision tree model.

**Report**

The Titanic Dataset is a very famous Keggle dataset, because it is not a very complex dataset to work with for students. However, I was suprised the effort required to even clean, process, analyze this dataset. In the clean up stage, it was interesting to realize the ratio of men to women losing their lives in this tragedy. As it was portrayed in the Titanic movie by James Cameron, the women and children were boarded to rescue vessels first before all man. In addition, it was suprisng that I couldn't achieve an accuracy above 90%. Even using grid tuning, hyper parameter tuning, I was able to reach a maximum accuracy of 0.8320 using Decision Tree training model.

**Reflection**

This course has been one of the best classes I have taken so far in my education life. Having a bachelors degree in engineering, I have never been exposed to data science part of programming, and this class has been a great tool to learn from basics to advanced clustering/classification methods. The fact that the tutorials are provided and reviewed each week, and being able to implement the taught course material on real life data has been great to practice and understand the concepts better. I have learned a lot while completing the homework. Before this class, I thought that Machine Learning was a very hard topic to grasp and learn, however this course changed my perspective in many fields such as how I look at data, how vital it is in our society, data gathering, the importance of data bias and cleaning, preprocessing methods, different ML models that have amazing mathematics behind. In addition, I was not only been able to improve myself in the field of data science, but also have learned R programming language with it.