Multi-Model Titanic Classifier

Author: Miguel Hidalgo

Version: 1.0

Purpose:

The purpose of this notebook is to demonstrate the process to construct a Multi-Model Classifier for the Titanic. A step by step will be presented and multiple models will be run. Because of the small data set of the Titanic, the entire data set will be used to train the models. The Caret package will be used for modeling.

Note: Comments in Red are my assessment.

The following Models will be implemented:

1- Random Forest

2- Logistic Regression

3- Support Vector Machines

4- Naive Bayes

5- Neural Network with Caret

The Process:

1-Extract Data (Provided by Kaggle in CSV format)

2-Explore Data (Summarize it)

3-Visualize Data

4-Complete/Transform/Add new Features (Feature Engineering — Consider in my opinion the more difficult part of the process)

5-Split Data (A random 70/30 - Train/Test but, using Crossvalidation during the training with Caret Package)

6-Train Model (Using all the Data available because of poor performance due to not enough samples or need for more Feature Engineering)

7-Validate Model

8-Predict with Trained Model (Using Blind data provide by Kaggle)

Load Required Package Libraries:

```
library(GGally) # Plotting-Extends ggplot
library(tidyverse) # Use to manipulate Tible data sets
library(tidyquant)
library(readxl) # Read Excel/CSV files
library(forcats)
library(stringr) # Manipulate string
library(skimr) # Explore/summarize data
library(caret) # Many Models with the same similar configuration
```

Load Data Sets, merge them and saved target labels and Ids

```
# Load data as tibbles
train_tbl <- read_csv("train.csv", col_names = TRUE) # Train Data
test_tbl <- read_csv("test.csv", col_names = TRUE)   # Blind Data

# Save Blind Data IDs
test.Id<-test_tbl$PassengerId

# Save Train data targets (Labels)
target<-train_tbl$Survived

# Combine Dataset to Clean/Complete/Transform - Remove the Target column
data_set_tbl <- rbind(train_tbl[, -2], test_tbl)
```

Summarize Data:

```
# Glimpse the whole data set, review it to determine the type of features to
select.
# Check type, Check Unique Values, Count Values and calculate proportions
# Provides Types, Hist and min, max, n_unique & n
skim(data_set_tbl)
```

```
Skim summary statistics
 n obs: 1309
 n variables: 11

Variable type: character
 variable missing complete    n min max empty n_unique
    Cabin     1014      295 1309   1  15     0      186
 Embarked        2     1307 1309   1   1     0        3
     Name        0     1309 1309  12  82     0     1307
      Sex        0     1309 1309   4   6     0        2
   Ticket        0     1309 1309   3  18     0      929

Variable type: integer
    variable missing complete    n   mean     sd p0 p25 p50 p75 p100     hist
       Parch       0     1309 1309   0.39   0.87  0   0   0   0    9
 PassengerId       0     1309 1309 655    378.02  1 328 655 982 1309
      Pclass       0     1309 1309   2.29   0.84  1   2   3   3    3
       SibSp       0     1309 1309   0.5    1.04  0   0   0   1    8

Variable type: numeric
 variable missing complete    n  mean    sd    p0  p25   p50   p75    p100     hist
      Age     263     1046 1309 29.88 14.41 0.17 21    28    39     80
     Fare       1     1308 1309 33.3  51.76 0     7.9 14.45 31.27 512.33
```

```r
# Glimpse the character data
data_set_tbl %>%
    select_if(is.character) %>%
    glimpse()
```

```
Observations: 1,309
Variables: 5
$ Name     <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Florence Briggs Thayer)", "Heikkinen, Mi
ss. Laina", "Futr...
$ Sex      <chr> "male", "female", "female", "female", "male", "male", "male", "male", "female", "female", "femal
e", "female", "ma...
$ Ticket   <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "373450", "330877", "17463", "349909", "3
47742", "237736",...
$ Cabin    <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, "G6", "C103", NA, NA, NA, NA, NA, NA, NA, NA,
NA, "D56", NA, "A...
$ Embarked <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S", "S", "S", "S", "S", "Q", "S", "S",
"C", "S", "S", "Q"...
```

```r
# Glimpse the Integer data
data_set_tbl %>%
    select_if(is.integer) %>%
    glimpse()
```

```
Observations: 1,309
Variables: 4
$ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 2
6, 27, 28, 29, 30,...
$ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3, 2, 2, 3, 1, 3, 3, 3, 1, 3, 1,
1, 3, 2, 1, 1, 3,...
$ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 3, 1, 0, 3, 0, 0, 0,
1, 0, 0, 1, 1, 0,...
$ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 5, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 0,...
```

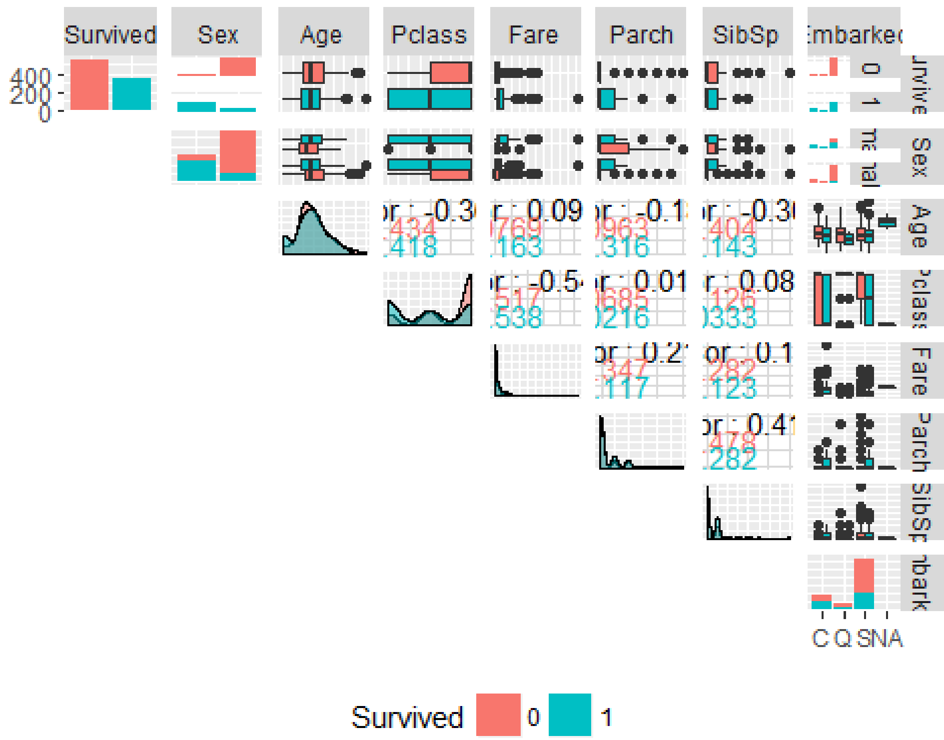
```r
# Glimpse the Double data
data_set_tbl %>%
    select_if(is.double) %>%
    glimpse()
```

```
Observations: 1,309
Variables: 2
$ Age  <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, 55, 2, NA, 31, NA, 35, 34, 15, 28, 8, 38,
NA, 19, NA, NA, 4...
$ Fare <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625, 21.0750, 11.1333, 30.0708, 16.7000, 26.55
00, 8.0500, 31.27...
```

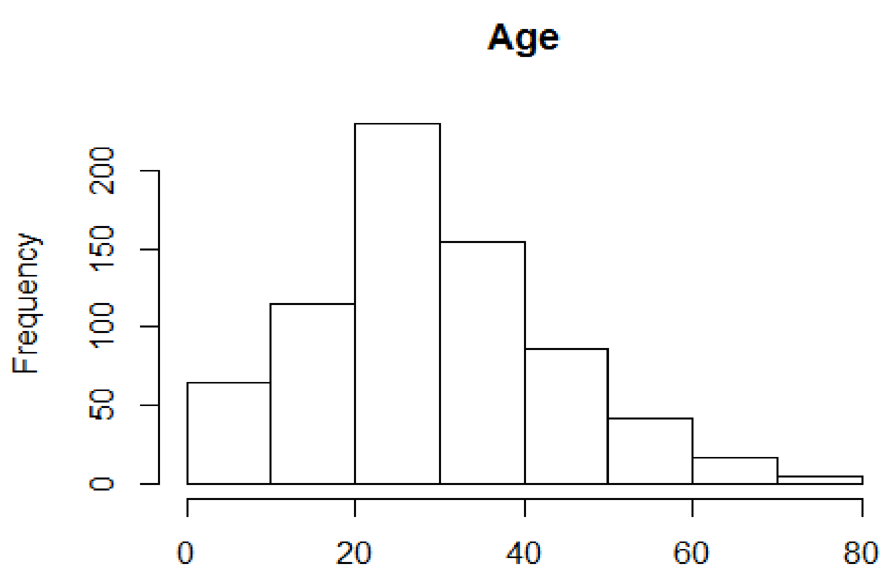
```r
# Check Numeric variables, determine the number of uniques values, filter for
hire than X & less than X to determine if it is numeric or factor.  This sele
ction of X really depends on the data.

# Save it as a dataframe.
feature_Length_df<-data_set_tbl %>%
  select_if(is.numeric) %>%
  map_df(~ unique(.)%>% length()) %>%
  gather() %>%
  arrange(value)

# Save it as a dataframe.
feature_factors_df<-data_set_tbl %>%
  select_if(is.numeric) %>%
  map_df(~ unique(.)%>% length()) %>%
  gather() %>%
  arrange(value) %>%
  filter(value<=10)
# Save it as a dataframe.
feature_numeric_df<-data_set_tbl %>%
  select_if(is.numeric) %>%
  map_df(~ unique(.)%>% length()) %>%
  gather() %>%
  arrange(value) %>%
  filter(value>5)
```

Initial Data Visualization

```r
# Convert the target to factor
train_tbl$Survived<-as.factor(train_tbl$Survived)

train_tbl %>%
  select(Survived, Sex, Age, Pclass, Fare, Parch, SibSp,Embarked) %>%
  ggpairs(aes(color=Survived), lower="blank", legend=1,
          diag=list(continuous=wrap("densityDiag", alpha=0.5)))+
  theme(legend.position="bottom")
```

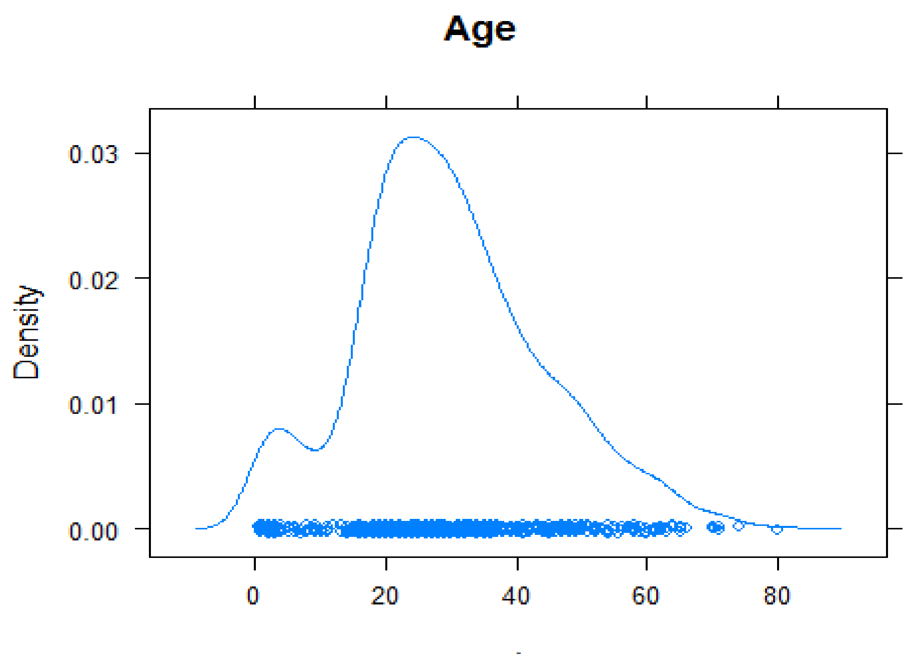



```
# Explore Numeric Features Distributions
train_tbl$Age%>%
  hist(main="Age")
```

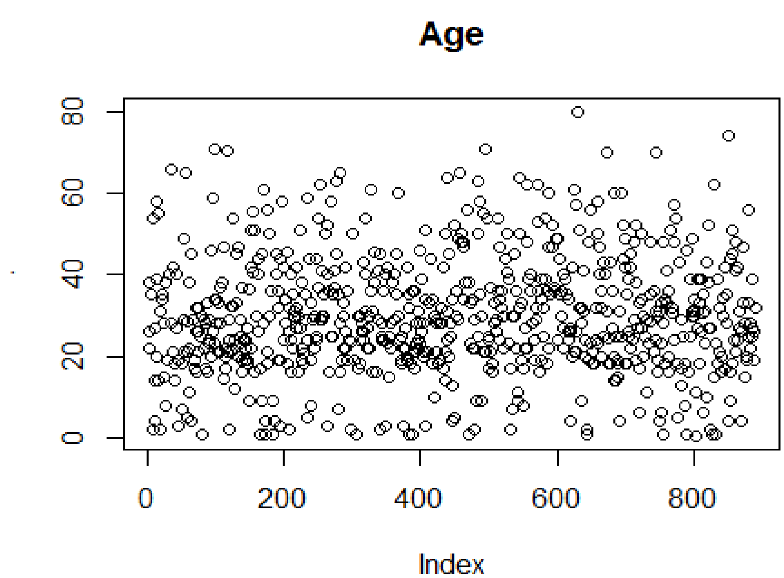



```
# Explore Numeric Features Density
```

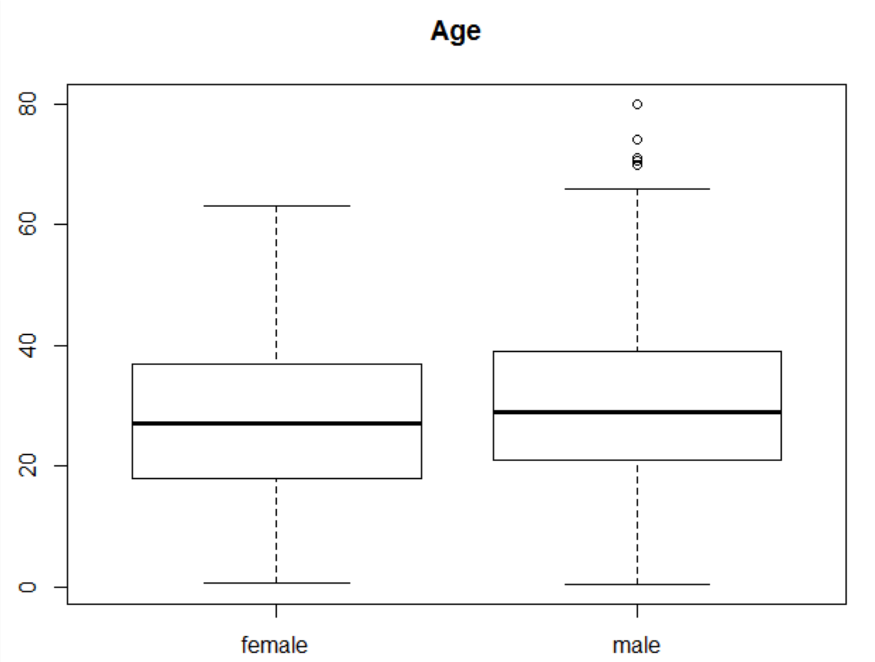
```
train_tbl$Age%>%
  densityplot(main = "Age")
```




```
# Explore Numeric Features - Scatter Plot

train_tbl$Age%>%
  plot(main = "Age")
```

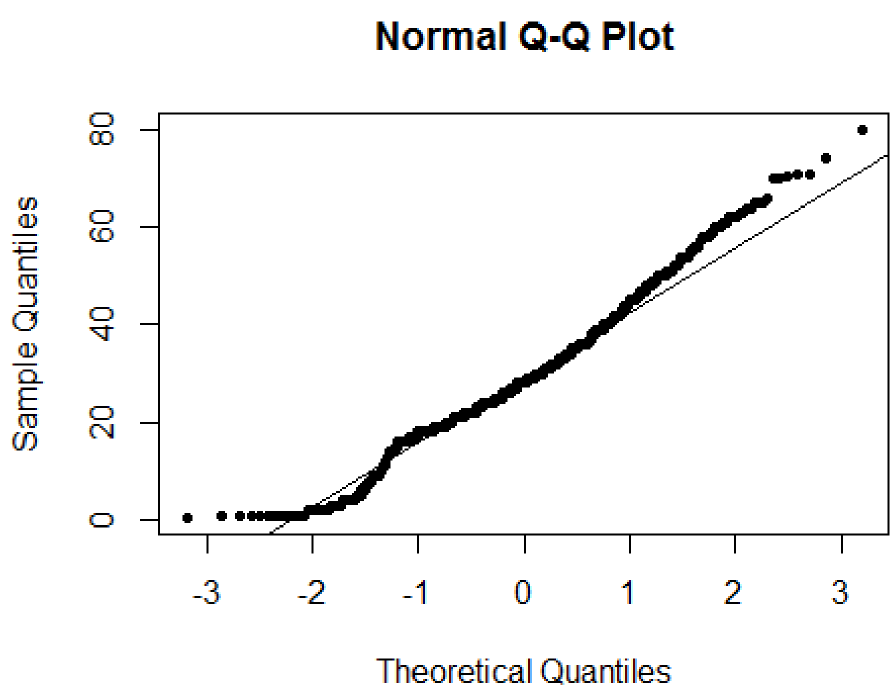



```
# Explore Numeric Features Distributions
  boxplot(train_tbl$Age~train_tbl$Sex,main = "Age")
```




```
## train_tbl$Age ~ train_tbl$Sex %>% boxplot(main = "Age")
```

```
# Check Normality with QQ-Norm plots
qqnorm(train_tbl$Age,pch=20);qqline(train_tbl$Age, color="blue")
```



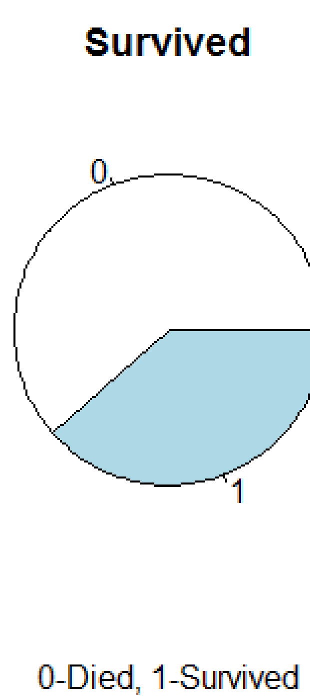

```
# Normality test, if p-value >than 0.05, we accept the null hypothesis-Is Nor
mal
```

```r
# Test for normality
train_tbl$Age%>%
  shapiro.test()
```

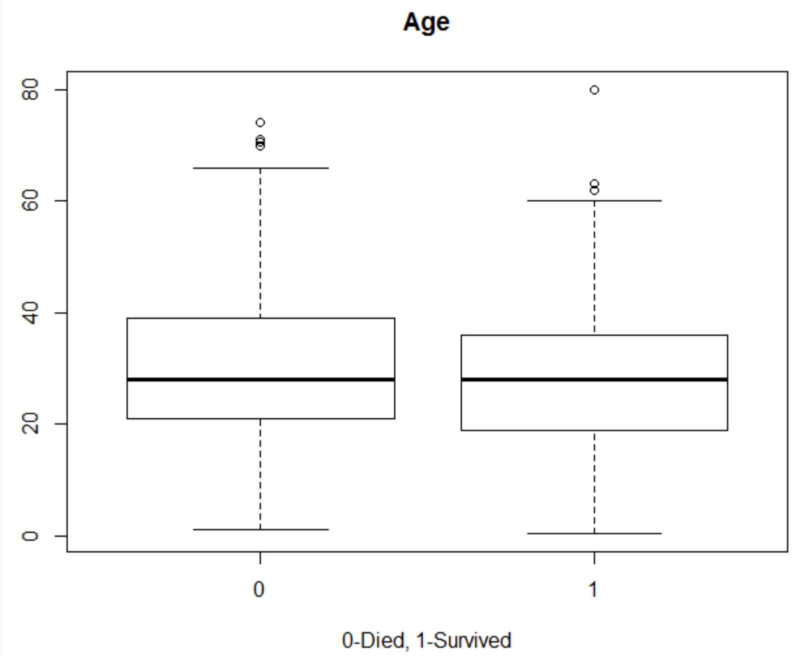
```
##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.98146, p-value = 7.337e-08
```

```r
# Not Normal, P-Value < 0.05
```

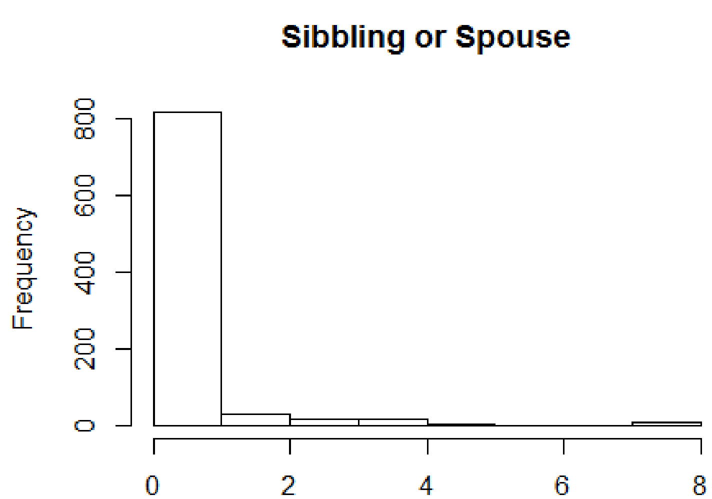
```r
# Plot Survived vs Died
table(train_tbl$Survived)%>%
  pie( main = "Survived", xlab="0-Died, 1-Survived")
```



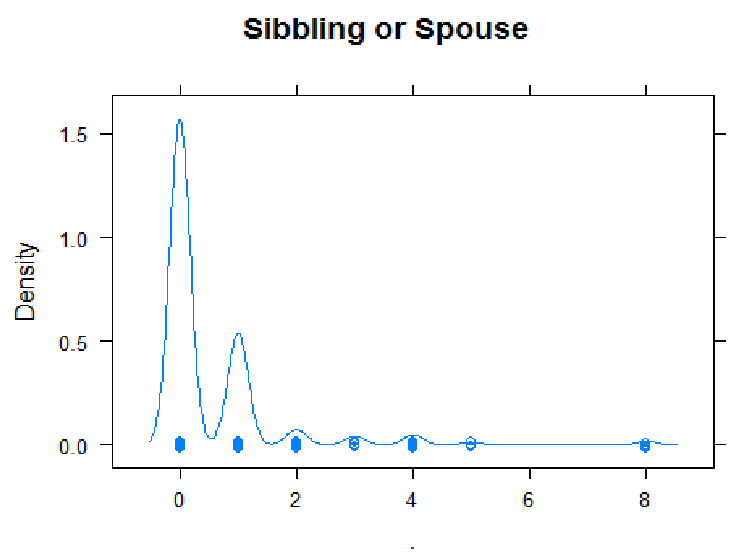

```r
# BoxPlot Survived vs Died by Age
  boxplot(train_tbl$Age~train_tbl$Survived, main="Age", xlab="0-Died, 1-Survi
ved")
```



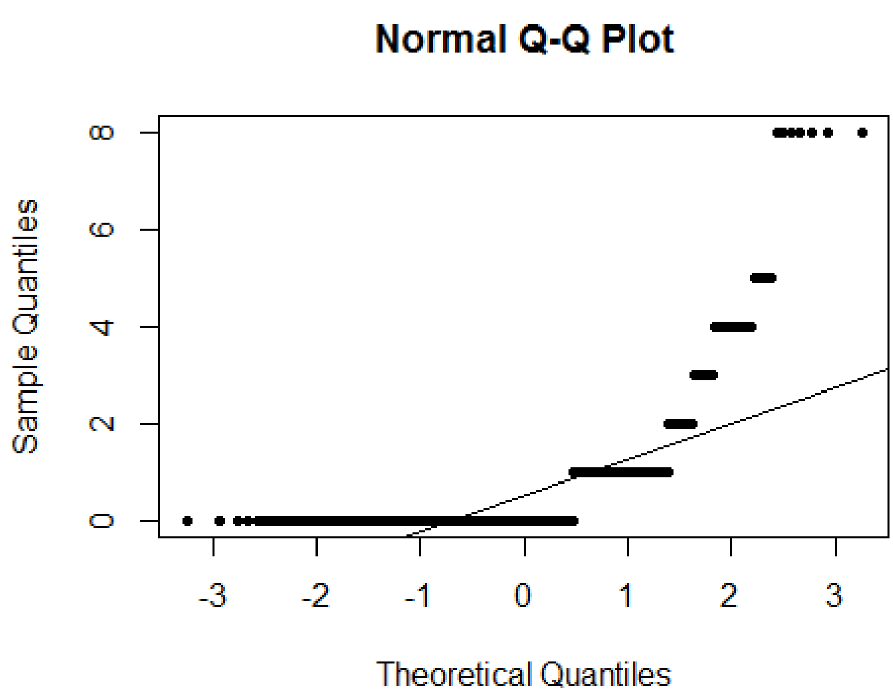

```
# Histogram - Distribution

train_tbl$SibSp%>%
  hist(main = "Sibbling or Spouse")
```




```
# Density Plot-

train_tbl$SibSp%>%
  densityplot(main = "Sibbling or Spouse")

# Appear that the majority of the people travel without Sibling or Spouse
```




```
# Check Normality with QQ-Norm plots
qqnorm(train_tbl$SibSp,pch=20);qqline(train_tbl$SibSp, color="blue")
```

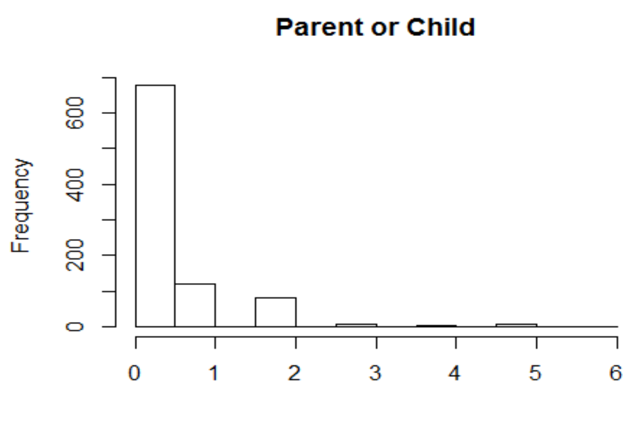



```
# Normality test, if p-value >than 0.05, we accept the null hypothesis-Is Nor
mal
# Test for normality
train_tbl$SibSp%>%
  shapiro.test()
```

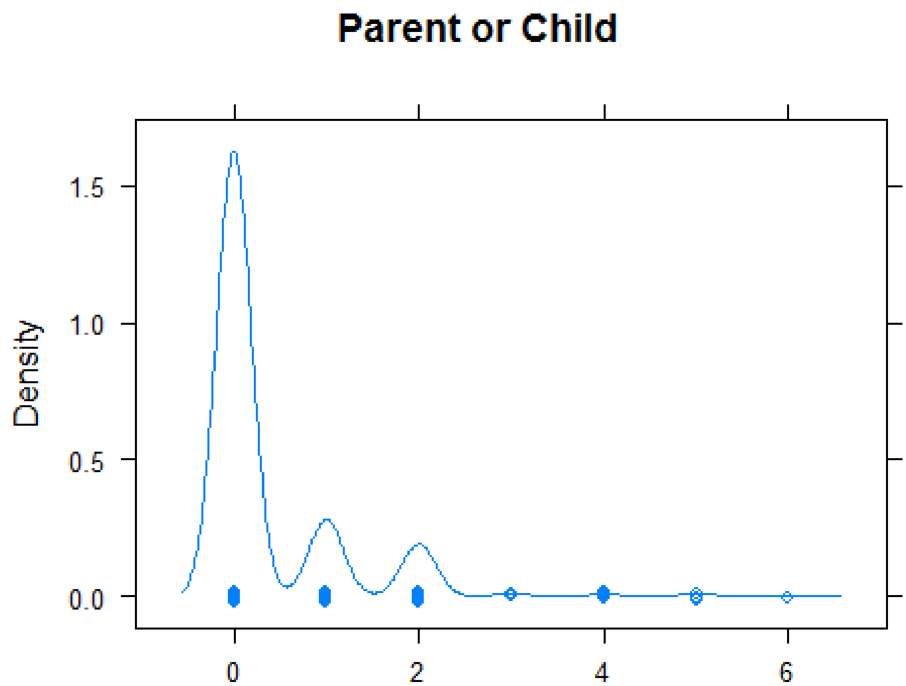
```
##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.51297, p-value < 2.2e-16
```

```
# Not Normal, P-Value < 0.05
```

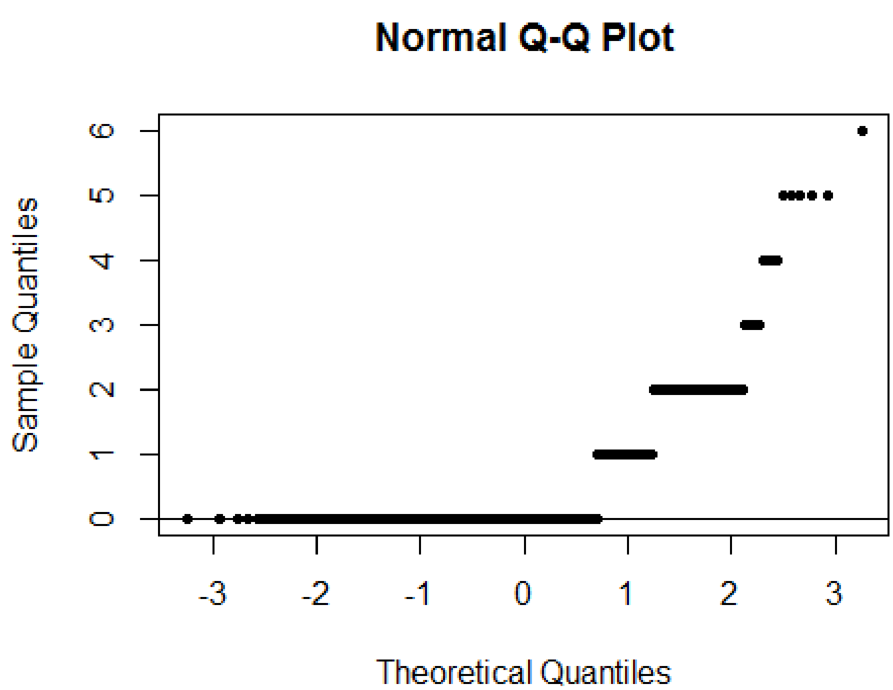
```
# Histogram - Distribution

Train_tbl$Parch%>%
  hist(main = "Parent or Child")
```




```
# Density PLot

train_tbl$Parch%>%
  densityplot(main = "Parent or Child")

# Appear that the majority of the people travel without Parent or Child
```




```r
# Check Normality with QQ-Norm plots
qqnorm(train_tbl$Parch,pch=20);qqline(train_tbl$Parch, color="blue")
```



```r
# Normality test, if p-value >than 0.05, we accept the null hypothesis-Is Nor
mal
# Test for normality
train_tbl$Parch%>%
  shapiro.test()
```

```
##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.53281, p-value < 2.2e-16
```

```
# Not Normal, P-Value < 0.05
```

```
# Although by just observing the Histograms or Density plots, you can see that the d
ata is not Normal. The use of QQNORM plots and Shapiro Normality Test was done with t
he purpose to emphasize that a consistent method is required. If I would only use the
Histogram for Age, I could conclude that is was relatively normal.
```

Feature Engineering & Data Transformation

```
Before making decision about transforming the data, you need a hypothesis:

Based on historical events, culture/human factors, the survivor's principal f
actor (Who get in the lifeboat) was prioritized as follow:
```

```
1- Women and Children (By First, Second and third class tickets)

2- People with upper level society titles (Countess, Colonel, etc.)

Based on the factors above I will use the titles and Sex to determine the
missing age of the passengers.
```

```r
# drop Features with not enough data or that you believe do not provide insig
hts

# Copy data set in case you need it later as is
data_set_orig_tbl<-data_set_tbl

# Drop PassengerID, Ticket, Cabin features
data_set_tbl<-data_set_tbl%>%
  select(-c(PassengerId, Ticket, Cabin))


# Create new features Travelling WithFamily and FamilyID
data_set_tbl<-data_set_tbl%>%
  mutate(WithFamily="N", Title="", FamilyID="")
```

```r
# Calculate the size of the family including the traveller+Parent or Childrin
+ Sibling or/and Spouse
data_set_tbl$FamilySize<-1+ data_set_tbl$SibSp + data_set_tbl$Parch

# Set value to Yes if travelling with someone
data_set_tbl[data_set_tbl$FamilySize>1,]$WithFamily<-"Y"

# Convert to factor
data_set_tbl$WithFamily<-as.factor(data_set_tbl$WithFamily)


# Construct Title category feature by extracting titles from names feature
data_set_tbl$Title <- sapply(data_set_tbl$Name, FUN=function(x) {strsplit(x,
split='[,.]')[[1]][2]})
# Remove spaces
data_set_tbl$Title <- sub(' ', '', data_set_tbl$Title)



# View Count of groups by Titles
data_set_tbl$Title%>%
  table()

## .
##         Capt          Col          Don         Dona           Dr
##            1            4            1            1            8
##     Jonkheer         Lady        Major       Master         Miss
##            1            1            2           61          260
##         Mlle          Mme           Mr          Mrs           Ms
##            2            1          757          197            2
##          Rev          Sir the Countess
##            8            1            1

# Combined titles variations to reduce it into a manageable category
data_set_tbl$Title[data_set_tbl$Title %in% c('Ms', 'Miss')] <- 'Ms'
data_set_tbl$Title[data_set_tbl$Title %in% c('Capt', 'Don', 'Major', 'Sir', '
Col','Dr', 'Rev', 'Mr')] <- 'Sir'
data_set_tbl$Title[data_set_tbl$Title %in% c('Dona', 'Lady', 'the Countess',
'Jonkheer','Mme', 'Mlle', 'Mrs')] <- 'Lady'
# Convert to factor
data_set_tbl$Title <- as.factor(data_set_tbl$Title)

# Remove Name Feature no longer needed at this time
data_set_tbl<-data_set_tbl%>%
  select(-c(Name))

#  Get median of children <13
child_median_age<-median(data_set_tbl[data_set_tbl$Age<13 & !(is.na(data_set_
tbl$Age)) & (data_set_tbl$Title=='Master' |data_set_tbl$Title=='Ms'),]$Age)
```

```r
# Replace missing Age values with Medians by title
data_set_tbl[is.na(data_set_tbl$Age) & (data_set_tbl$Title=='Master' | data_set_tbl$Title=='Ms') ,]$Age<-child_median_age

# Get Female Median older than 13
Adult_female_median_age<-median(data_set_tbl[data_set_tbl$Age>=13 & !(is.na(data_set_tbl$Age)) & data_set_tbl$Sex=="female",]$Age)

# Replace missing Age values with Medians by title
data_set_tbl[is.na(data_set_tbl$Age) & (data_set_tbl$Title=='Lady' | data_set_tbl$Title=='Ms'),]$Age<-Adult_female_median_age

# Get male Median older than 13
Adult_male_median_age<-median(unlist(data_set_tbl[data_set_tbl$Age>=13 & !(is.na(data_set_tbl$Age)) & data_set_tbl$Sex=="male","Age"]))

# Replace missing Age values with Medians by title
data_set_tbl[is.na(data_set_tbl$Age) & (data_set_tbl$Title=='Sir') & data_set_tbl$Sex=="male",]$Age<-Adult_male_median_age

# View Count for Embarkment
 data_set_tbl$Embarked%>%
  table()

## .
##   C   Q   S
## 270 123 914

 # Replace the missing Values with the highest count of Embarkemnt (Mode) in this case S
data_set_tbl[is.na(data_set_tbl$Embarked),]$Embarked<-"S"

# Assign Values to identify Large (>2) & Small families (<=2) -- This will need to be Tuned better as you evaluate Models
data_set_tbl$FamilyID<-"Large"

# Categorize by size names Small
data_set_tbl$FamilyID[data_set_tbl$FamilySize <= 2] <- "Small"

# Convert features to factors
data_set_tbl$FamilyID <- as.factor(data_set_tbl$FamilyID)

# Convert features to factors
data_set_tbl$Sex<-as.factor(data_set_tbl$Sex)

# Convert features to factors
data_set_tbl$Pclass<-as.factor(data_set_tbl$Pclass)
```

```
# Convert features to factors
data_set_tbl$Embarked<-as.factor(data_set_tbl$Embarked)

# Re-check Missing Values -- 1 Missing
MissingFare<-sum(is.na(data_set_tbl$Fare))

# Replace missing data with median for the Missing Fare value
data_set_tbl[is.na(data_set_tbl$Fare),]$Fare<-median(!(is.na(data_set_tbl$Far
e)))

# Re-check Missing Values --- 0 Missing
MissingFare<-sum(is.na(data_set_tbl$Fare))

# Extract the features types
feature_classes <- sapply(names(data_set_tbl),function(x){class(data_set_tbl[
[x]])})

# Extract the factors heading names
categorical_feats <- names(feature_classes[feature_classes == "factor"])

# Extract the numeric heading names
numeric_feats <-names(feature_classes[feature_classes != "factor" ])

# Hot 1 encoding. Create list with all factors and its levels
dummies <- dummyVars(~.,data_set_tbl[categorical_feats])

# Create new features for each level of each categorical feature and place 0
and 1 in accordance to levels
categorical_1_hot <- predict(dummies,data_set_tbl[categorical_feats])

# Ensure any level with an NA is replaced with a 0
categorical_1_hot[is.na(categorical_1_hot)] <- 0  #for any level that was NA,
set to zero

# Display top of set
head(categorical_1_hot)
```

```
##   Pclass.1 Pclass.2 Pclass.3 Sex.female Sex.male Embarked.C Embarked.Q
## 1        0        0        1          0        1          0          0
## 2        1        0        0          1        0          1          0
## 3        0        0        1          1        0          0          0
## 4        1        0        0          1        0          0          0
## 5        0        0        1          0        1          0          0
## 6        0        0        1          0        1          0          1
##   Embarked.S WithFamily.N WithFamily.Y Title.Lady Title.Master Title.Ms
## 1          1            0            1          0            0        0
## 2          0            0            1          1            0        0
## 3          1            1            0          0            0        1
## 4          1            0            1          1            0        0
```

```
## 5         1            1            0           0            0        0
## 6         0            1            0           0            0        0
##   Title.Sir FamilyID.Large FamilyID.Small
## 1         1              0              1
## 2         0              0              1
## 3         0              0              1
## 4         0              0              1
## 5         1              0              1
## 6         1              0              1
```

```
# Create data Frame with Numeric Data
df.num<-data_set_tbl[,numeric_feats]
# Convert to numeric -- Required by preProcess Transformations
df.num$SibSp<-as.numeric(df.num$SibSp)
# Convert to numeric -- Required by preProcess Transformations
df.num$Parch<-as.numeric(df.num$Parch)
# Convert to numeric -- Required by preProcess Transformations
df.num$Age<-as.numeric(df.num$Age)
# Convert to numeric -- Required by preProcess Transformations
df.num$Fare<-as.numeric(df.num$Fare)
# Convert to numeric -- Required by preProcess Transformations
df.num$FamilySize<-as.numeric(df.num$FamilySize)

# Verify they were converted to dbl
glimpse(df.num)
```

```
## Observations: 1,309
## Variables: 5
## $ Age        <dbl> 22, 38, 26, 35, 35, 30, 54, 2, 27, 14, 4, 58, 20, 3...
## $ SibSp      <dbl> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, ...
## $ Parch      <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, ...
## $ Fare       <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 5...
## $ FamilySize <dbl> 2, 2, 1, 2, 1, 1, 1, 5, 3, 2, 3, 1, 1, 7, 1, 1, 6, ...
```

```
# Convert to Dataframe only
df.num<-df.num %>%
  as.matrix() %>%
  unlist() %>%
  as.data.frame()
# Verify it was converted to Data Frame
glimpse(df.num)
```

```
## Observations: 1,309
## Variables: 5
## $ Age        <dbl> 22, 38, 26, 35, 35, 30, 54, 2, 27, 14, 4, 58, 20, 3...
## $ SibSp      <dbl> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, ...
## $ Parch      <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, ...
## $ Fare       <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 5...
## $ FamilySize <dbl> 2, 2, 1, 2, 1, 1, 1, 5, 3, 2, 3, 1, 1, 7, 1, 1, 6, ...
```

```r
# Transform Data by Scaling it, Center it and linearize with a BoxCox Transfo
rmation
preProcValues <- preProcess(df.num, method = c("center","scale", "BoxCox"))

# Apply PreProcessing to Data Frame
df.num <- predict(preProcValues, df.num)

# View Numeric Data Frame transformed
glimpse(df.num)

## Observations: 1,309
## Variables: 5
## $ Age        <dbl> -0.48088731, 0.66536080, -0.19432529, 0.45043928, 0...
## $ SibSp      <dbl> 0.4811039, 0.4811039, -0.4789037, 0.4811039, -0.478...
## $ Parch      <dbl> -0.4448295, -0.4448295, -0.4448295, -0.4448295, -0....
## $ Fare       <dbl> -0.50285078, 0.73458949, -0.48980644, 0.38319814, -...
## $ FamilySize <dbl> 0.9102699, 0.9102699, -0.7959567, 0.9102699, -0.795...

# Merge Numerical data frame with Hot-1 encode categorical features and recon
struct the data set
df<-cbind(df.num,categorical_1_hot)

# View Data Frame transformed
glimpse(df)

## Observations: 1,309
## Variables: 21
## $ Age            <dbl> -0.48088731, 0.66536080, -0.19432529, 0.4504392...
## $ SibSp          <dbl> 0.4811039, 0.4811039, -0.4789037, 0.4811039, -0...
## $ Parch          <dbl> -0.4448295, -0.4448295, -0.4448295, -0.4448295,...
## $ Fare           <dbl> -0.50285078, 0.73458949, -0.48980644, 0.3831981...
## $ FamilySize     <dbl> 0.9102699, 0.9102699, -0.7959567, 0.9102699, -0...
## $ Pclass.1       <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,...
## $ Pclass.2       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,...
## $ Pclass.3       <dbl> 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0,...
## $ Sex.female     <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,...
## $ Sex.male       <dbl> 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,...
## $ Embarked.C     <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,...
## $ Embarked.Q     <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Embarked.S     <dbl> 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,...
## $ WithFamily.N   <dbl> 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,...
## $ WithFamily.Y   <dbl> 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,...
## $ Title.Lady     <dbl> 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,...
## $ Title.Master   <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Title.Ms       <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,...
## $ Title.Sir      <dbl> 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,...
## $ FamilyID.Large <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0,...
## $ FamilyID.Small <dbl> 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,...
```

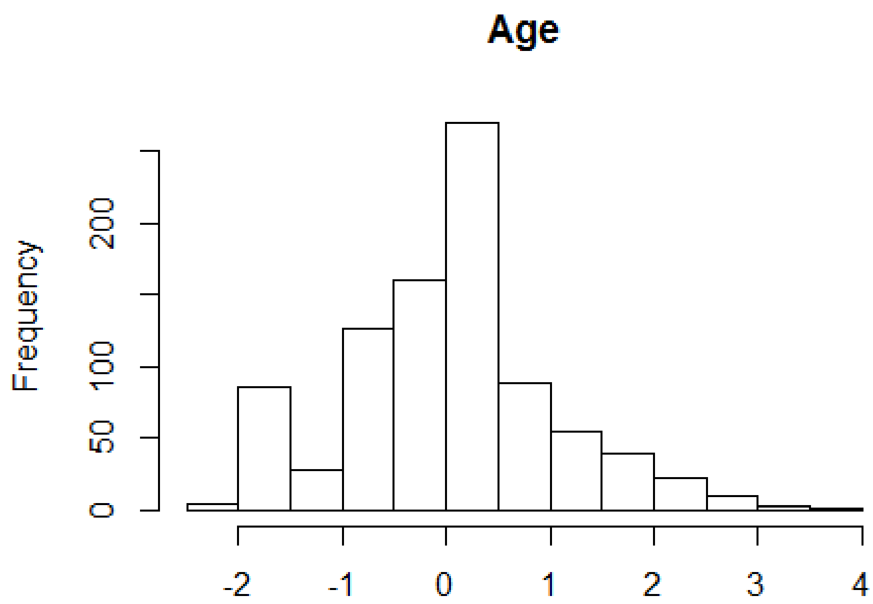
```r
# Save previously processed data
data.set.orig<-data_set_tbl

# Replace data with tranformed data
data_set_tbl<-df

# Recheck for Missing Values - No Surprises
zeroMissing<-as.character(sum(is.na(data_set_tbl)))

#Convert Names - Remove symbols
# Get Names Total New Features 21
vnames<-names(data_set_tbl)
# Remove .
vnames<-gsub('\\.', '', vnames)
# Rename Data Set Columns
names(data_set_tbl)<-vnames

# Save data set for Deep Learning Neural Network Keras Model
DLNN_data_set_tbl<-data_set_tbl



# Re-convert all categorical data transformed to factors - Required by Caret
Models but not Keras NN
data_set_tbl$Pclass1<-as.factor(data_set_tbl$Pclass1)
data_set_tbl$Pclass2<-as.factor(data_set_tbl$Pclass2)
data_set_tbl$Pclass3<-as.factor(data_set_tbl$Pclass3)
data_set_tbl$Sexfemale<-as.factor(data_set_tbl$Sexfemale)
data_set_tbl$Sexmale<-as.factor(data_set_tbl$Sexmale)
data_set_tbl$EmbarkedC<-as.factor(data_set_tbl$EmbarkedC)
data_set_tbl$EmbarkedQ<-as.factor(data_set_tbl$EmbarkedQ)
data_set_tbl$EmbarkedS<-as.factor(data_set_tbl$EmbarkedS)
data_set_tbl$FamilyIDSmall<-as.factor(data_set_tbl$FamilyIDSmall)
data_set_tbl$FamilyIDLarge<-as.factor(data_set_tbl$FamilyIDLarge)
data_set_tbl$TitleSir<-as.factor(data_set_tbl$TitleSir)
data_set_tbl$TitleLady<-as.factor(data_set_tbl$TitleLady)
data_set_tbl$TitleMaster<-as.factor(data_set_tbl$TitleMaster)
data_set_tbl$TitleMs<-as.factor(data_set_tbl$TitleMs)
data_set_tbl$TitleSir<-as.factor(data_set_tbl$TitleSir)
data_set_tbl$WithFamilyY<-as.factor(data_set_tbl$WithFamilyY)
data_set_tbl$WithFamilyN<-as.factor(data_set_tbl$WithFamilyN)



# Caret package Models requires this step. (I did not create a function for r
eadability)
# Encode Category clasess for Models - Train Set
feature.names=names(data_set_tbl)

for (f in feature.names) {
```

```
  if (class(data_set_tbl[[f]])=="factor") {
    levels <- unique(c(data_set_tbl[[f]]))
    data_set_tbl[[f]] <- factor(data_set_tbl[[f]],
                   labels=make.names(levels))
  }
}
# Glimpse Data
glimpse(data_set_tbl)

# Notice the X1/X2 below, they replaced the 0/1 values, this is done to meet
a requirement of R/Caret package.  All categorical values are now X1/X2

## Observations: 1,309
## Variables: 21
## $ Age           <dbl> -0.48088731, 0.66536080, -0.19432529, 0.45043928...
## $ SibSp         <dbl> 0.4811039, 0.4811039, -0.4789037, 0.4811039, -0....
## $ Parch         <dbl> -0.4448295, -0.4448295, -0.4448295, -0.4448295, ...
## $ Fare          <dbl> -0.50285078, 0.73458949, -0.48980644, 0.38319814...
## $ FamilySize    <dbl> 0.9102699, 0.9102699, -0.7959567, 0.9102699, -0....
## $ Pclass1       <fct> X1, X2, X1, X2, X1, X1, X2, X1, X1, X1, X1, X2, ...
## $ Pclass2       <fct> X1, X1, X1, X1, X1, X1, X1, X1, X1, X2, X1, X1, ...
## $ Pclass3       <fct> X1, X2, X1, X2, X1, X1, X2, X1, X1, X2, X1, X2, ...
## $ Sexfemale     <fct> X1, X2, X2, X2, X1, X1, X1, X1, X2, X2, X2, X2, ...
## $ Sexmale       <fct> X1, X2, X2, X2, X1, X1, X1, X1, X2, X2, X2, X2, ...
## $ EmbarkedC     <fct> X1, X2, X1, X1, X1, X1, X1, X1, X1, X2, X1, X1, ...
## $ EmbarkedQ     <fct> X1, X1, X1, X1, X1, X2, X1, X1, X1, X1, X1, X1, ...
## $ EmbarkedS     <fct> X1, X2, X1, X1, X1, X2, X1, X1, X1, X2, X1, X1, ...
## $ WithFamilyN   <fct> X1, X1, X2, X1, X2, X2, X2, X1, X1, X1, X1, X2, ...
## $ WithFamilyY   <fct> X1, X1, X2, X1, X2, X2, X2, X1, X1, X1, X1, X2, ...
## $ TitleLady     <fct> X1, X2, X1, X2, X1, X1, X1, X1, X2, X2, X1, X1, ...
## $ TitleMaster   <fct> X1, X1, X1, X1, X1, X1, X1, X2, X1, X1, X1, X1, ...
## $ TitleMs       <fct> X1, X1, X2, X1, X1, X1, X1, X1, X1, X1, X2, X2, ...
## $ TitleSir      <fct> X1, X2, X2, X2, X1, X1, X1, X2, X2, X2, X2, X2, ...
## $ FamilyIDLarge <fct> X1, X1, X1, X1, X1, X1, X1, X2, X2, X1, X2, X1, ...
## $ FamilyIDSmall <fct> X1, X1, X1, X1, X1, X1, X1, X2, X2, X1, X2, X1, ...

# Notice the Target (Label Class) is now named target and not Survived.

# Split data back out. -
train <- data_set_tbl[1:891,] # Train Data
# Create a target column in the data frame and make sure is a factor type
train$target <- as.factor(ifelse(target==0, "X1", "X2"))

# Separate Blind data
test <- data_set_tbl[892:1309,]

# To ensure repeatability
set.seed(12345)
```

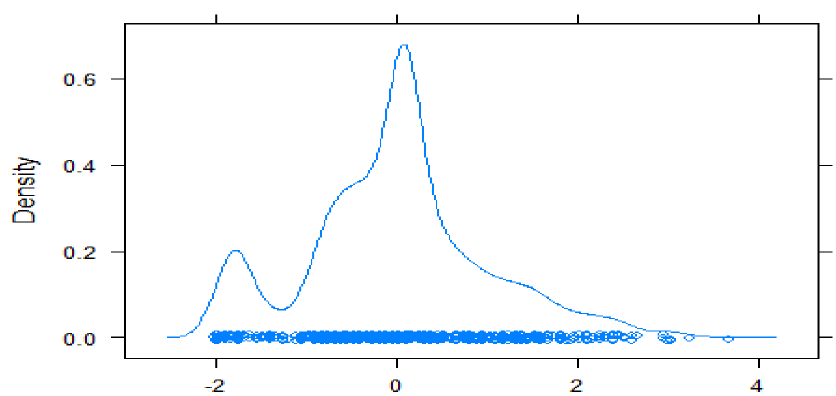
```r
# Create Indexes- Stratified proportional random samples based on your label
(Survived--target) 70/30
indexes <- createDataPartition(train$target, # Create proportion based on Lab
el
                               times = 1, # Number of Splits
                               p = 0.7,
                               list = FALSE)


# Create a train set with all data.  Normally we use the split and train the
model with the train set and validate with a validation set
# In this case because of low samples I will train with the entire set and va
lidate with data previouly used for training.  This is not the
# norm
X_train <- as.data.frame(train) # X_train <- as.data.frame(train[indexes,]) <
== This is what normally happens
# Extract Test set 30%
X_test <- as.data.frame(train[-indexes,])

# Convert to Data Frame the Blind folded data
test <- as.data.frame(test)
```

Data Visualization After Cleaning/FE/Transforming

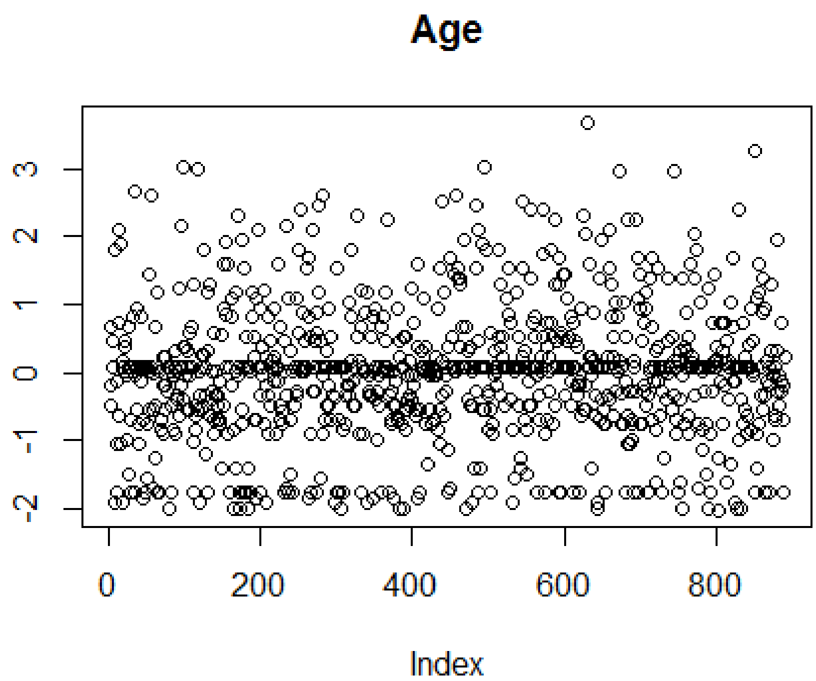
```r
# Note, because I performed a HOT-1 encoding in all categorical data, the cat
egorical features names changed.

#Explore Numeric Features Distributions
train$Age%>%
  hist(main="Age")
```



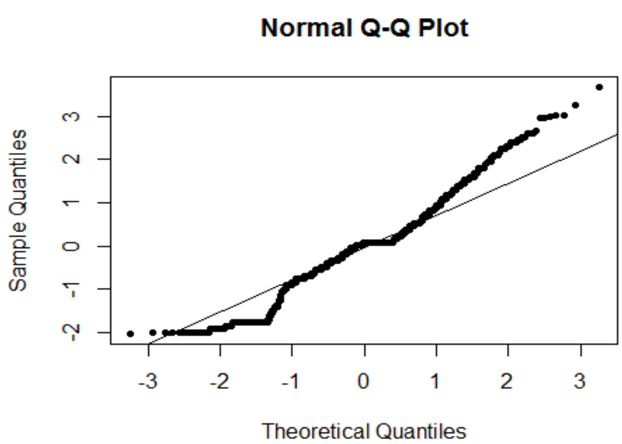

```
train$Age%>%
  densityplot()
```




```
train$Age%>%
  plot(main = "Age")
```

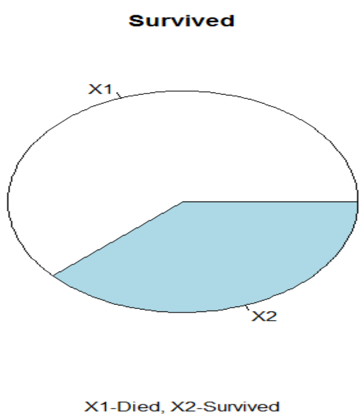



```r
# Check Normality with QQ-Norm plots
qqnorm(train$Age,pch=20);qqline(train$Age, color="blue")
```

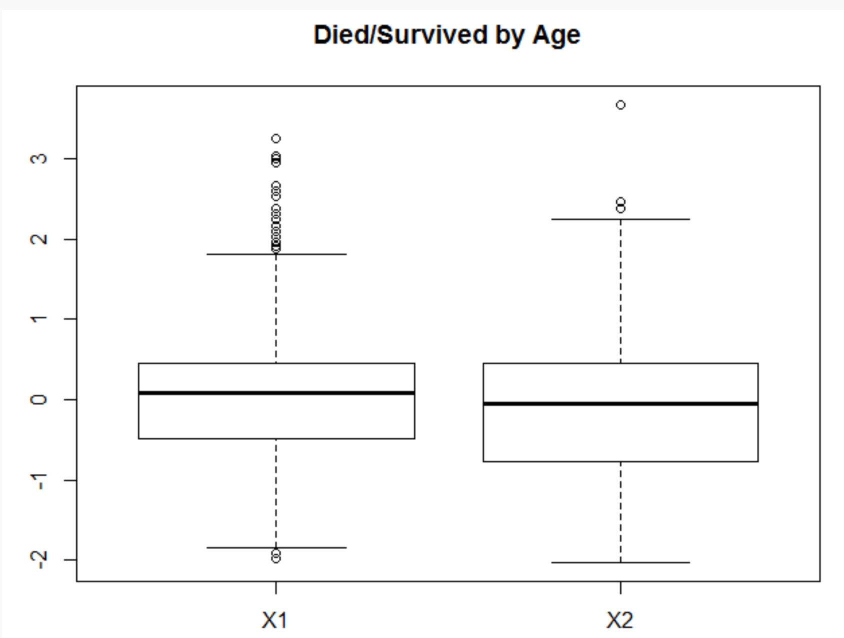



```r
# Normality test, if p-value >than 0.05, we accept the null hypothesis-Is Nor
mal
# Test for normality
train$Age%>%
  shapiro.test()

##
##  Shapiro-Wilk normality test
```

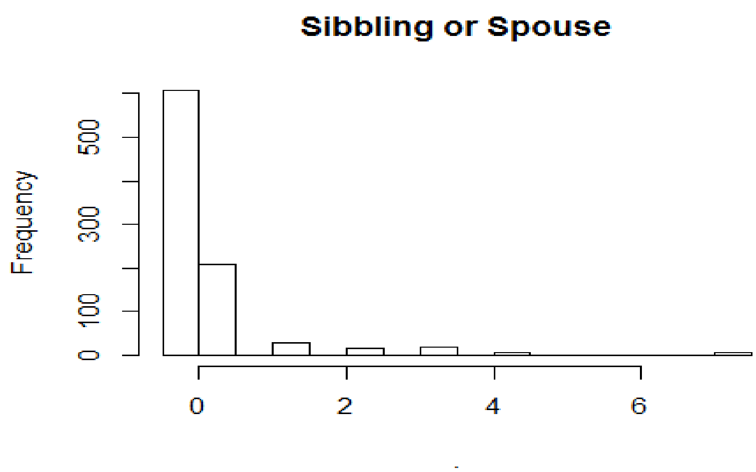
```
##
## data:  .
## W = 0.96781, p-value = 3.979e-13

# Not Normal, P-Value < 0.05

# Plot Survived vs Died
table(train$target)%>%
  pie( main = "Survived", xlab="X1-Died, X2-Survived")
```



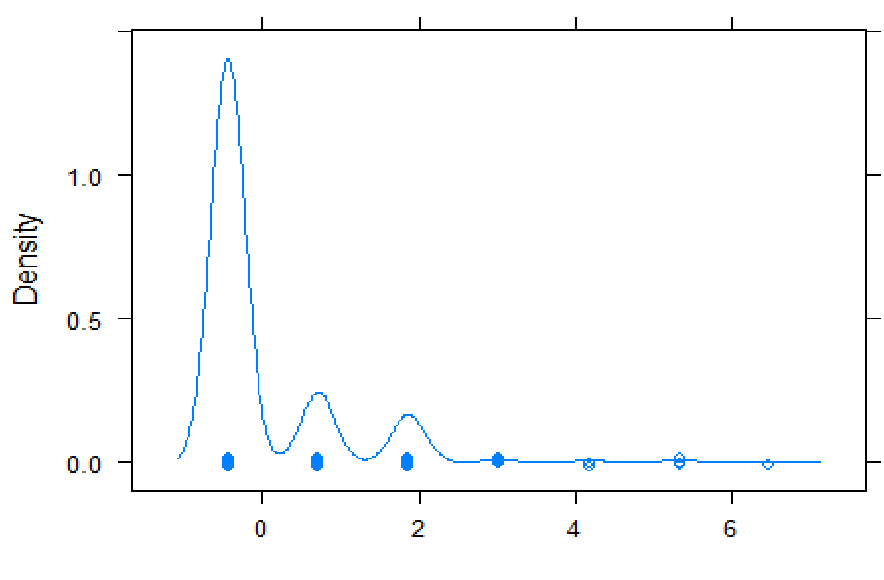

```
train$Age~train$target%>%

      boxplot( main="Age", xlab="X1-Died, X2-Survived")
```



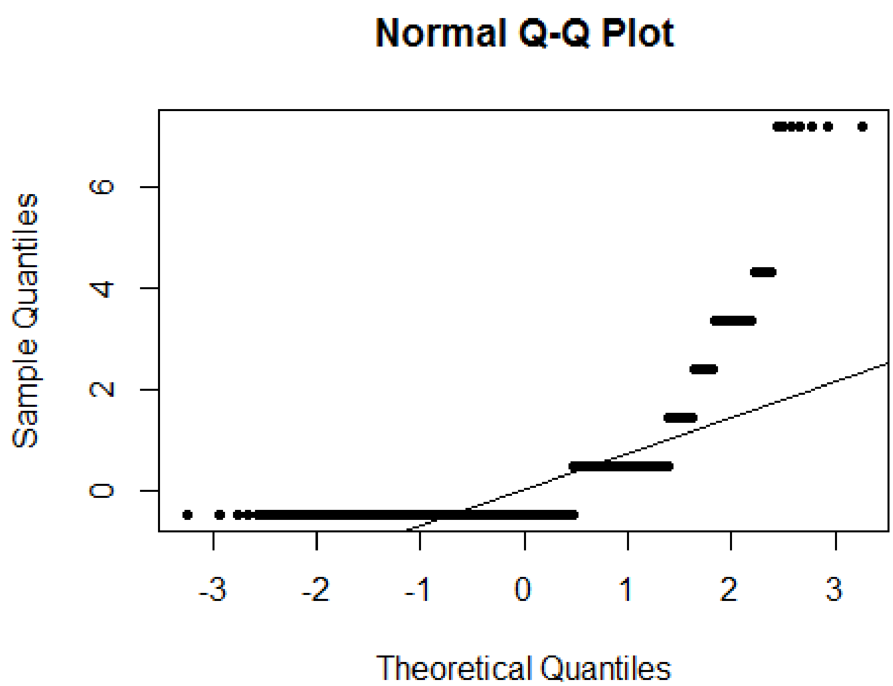
```
train$SibSp%>%
  hist(main = "Sibbling or Spouse")
```




```
train$Parch%>%
  densityplot()
```

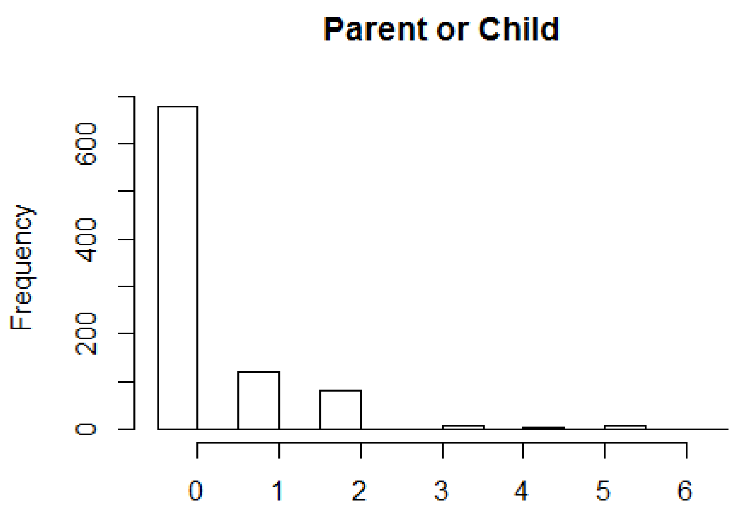


```r
# Check Normality with QQ-Norm plots
qqnorm(train$SibSp,pch=20);qqline(train$SibSp, color="blue")
```

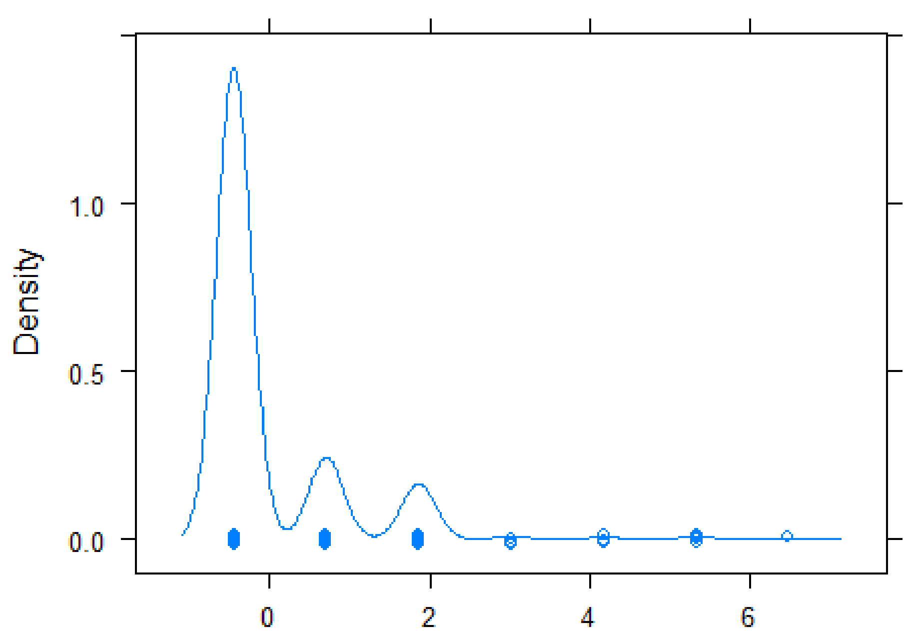



```r
# Normality test, if p-value >than 0.05, we accept the null hypothesis-Is Nor
mal
# Test for normality
train$SibSp%>%
  shapiro.test()

##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.51297, p-value < 2.2e-16
```

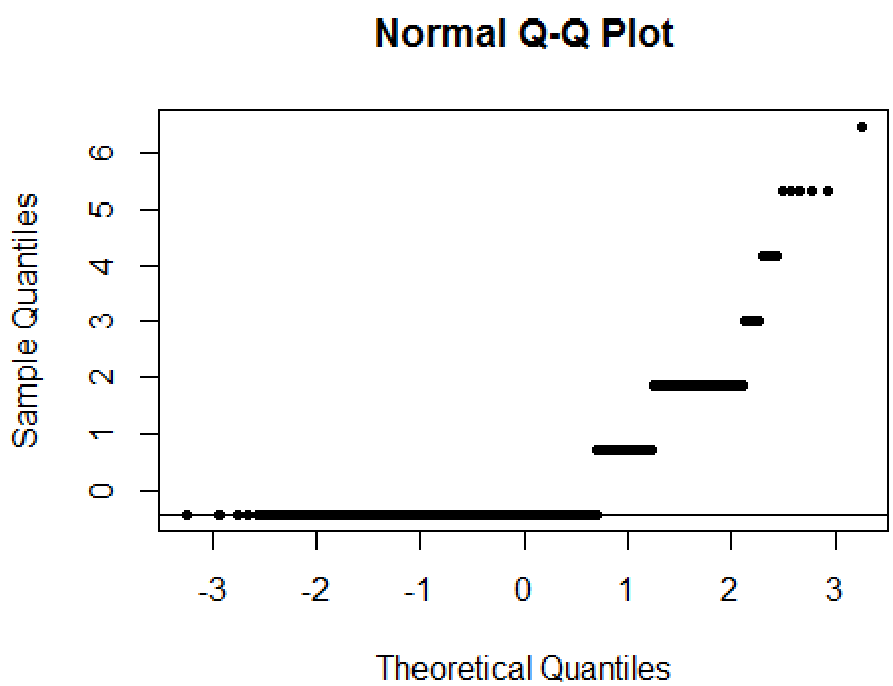
```
train$Parch%>%
  hist(main = "Parent or Child")
```




```
train$Parch%>%
  densityplot()
```



```r
# Check Normality with QQ-Norm plots
qqnorm(train$Parch,pch=20);qqline(train$Parch, color="blue")
```




```r
# Normality test, if p-value >than 0.05, we accept the null hypothesis-Is Nor
mal
# Test for normality
train$Parch%>%
  shapiro.test()

##
##  Shapiro-Wilk normality test
##
## data:  .
## W = 0.53281, p-value < 2.2e-16
```

Multiple Machine Learning Models will be executed

Consider that there has not been any fine tunning of the algorithm, therefore the results may no be the best. After submitting them the first time is time to fine tune them and also review if the amount of feature engineering that was done briefly in this example can be improved.

In addition to Accuracy, the Kappa Metric is utilized. Below is a guideline to interpret the Kappa KPI:

Here is one possible interpretation of Kappa.

- Poor agreement = Less than 0.20
- Fair agreement = 0.20 to 0.40
- Moderate agreement = 0.40 to 0.60
- Good agreement = 0.60 to 0.80
- Very good agreement = 0.80 to 1.00

Load libraries required by Models

```
# There is a conflict between the preview loaded libraries and these librarie
s, So, this order is required.  Once Notebook is completely run, if we want t
o run it completely again, we must unload all libraries and start from the be
ggining.

library(doSNOW) # Parallel processes of the model - Speeds the training of th
e Caret Models
library(klaR)
library(ElemStatLearn)
library(mlbench)
library(DMwR)
```

Train Random Forest Model 0.79904 Kaggle

```
b_time<-Sys.time()
print(b_time)

## [1] "2018-05-12 07:56:43 CDT"

# Set seed to ensure reproducibility between runs
set.seed(12345)


# Set up caret to perform 10-fold cross validation repeated 5 times
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              classProbs=TRUE,
                              search = "grid",
                              verboseIter = FALSE,
                              savePredictions = TRUE)

# Hyperparameter - search 1-10
tune.grid <- expand.grid(mtry=c(1:10))

# Create clusters for parallel processing
cl <- makeCluster(10, type = "SOCK") #10 parallel processes-RStudio running a
```

```r
t the same time


# Register cluster so that caret will know to train in parallel.
registerDoSNOW(cl)


# Train model with 150 trees and 7 hyperparameter -- The reader need to learn
how to fine tune these values
rf.cv <- train(target ~ .,
               data=X_train,
               method =  "rf",
               tuneGrid = tune.grid,
               ntree=150,
               trControl = caret.control,
               tuneLength = 7)

# Stop cluster
stopCluster(cl)

# Display the results of the cross validation run -
# mean accuracy!
rf.cv

## Random Forest
##
## 891 samples
##  21 predictor
##   2 classes: 'X1', 'X2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 802, 803, 802, 802, 802, 802, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.8184153  0.6014209
##    2    0.8282932  0.6278072
##    3    0.8197484  0.6065893
##    4    0.8174960  0.6009647
##    5    0.8206371  0.6082885
##    6    0.8253261  0.6204934
##    7    0.8266869  0.6249984
##    8    0.8240030  0.6205315
##    9    0.8230789  0.6186809
##   10    0.8203721  0.6134225
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
All Kappas are at the beginning of the GOOD Range

# Make predictions- Validate with the validation test set
preds <- predict(rf.cv, X_test)

# Use caret's confusionMatrix() function to estimate the
# effectiveness of this model on unseen, new data.
confusion.matrix<-confusionMatrix(preds, X_test$target)

print(confusion.matrix)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   X1  X2  How to interpret Results:
##           X1 146  33   (146 People that died was classified correctly.
##           X2  18  69    69 Survivors were classified correctly.  18 People
##                         that died were classified as survivor.  33 people
##                         that survived were classified as dead.)

##                Accuracy : 0.8083
##                  95% CI : (0.7557, 0.8538)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 1.164e-11
##
##                   Kappa : 0.5829
##  Mcnemar's Test P-Value : 0.04995
##
##             Sensitivity : 0.8902
##             Specificity : 0.6765
##          Pos Pred Value : 0.8156
##          Neg Pred Value : 0.7931
##              Prevalence : 0.6165
##          Detection Rate : 0.5489
##    Detection Prevalence : 0.6729
##       Balanced Accuracy : 0.7834
##
##        'Positive' Class : X1
##

# What is the standard deviation?
cat(paste("\nCross validation standard deviation:",
          sd(rf.cv$resample$Accuracy), "\n", sep = " "))

##
## Cross validation standard deviation: 0.0409932084754562

cat(paste("\nCross validation mean:",
          mean(rf.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation mean: 0.82829315628192
```

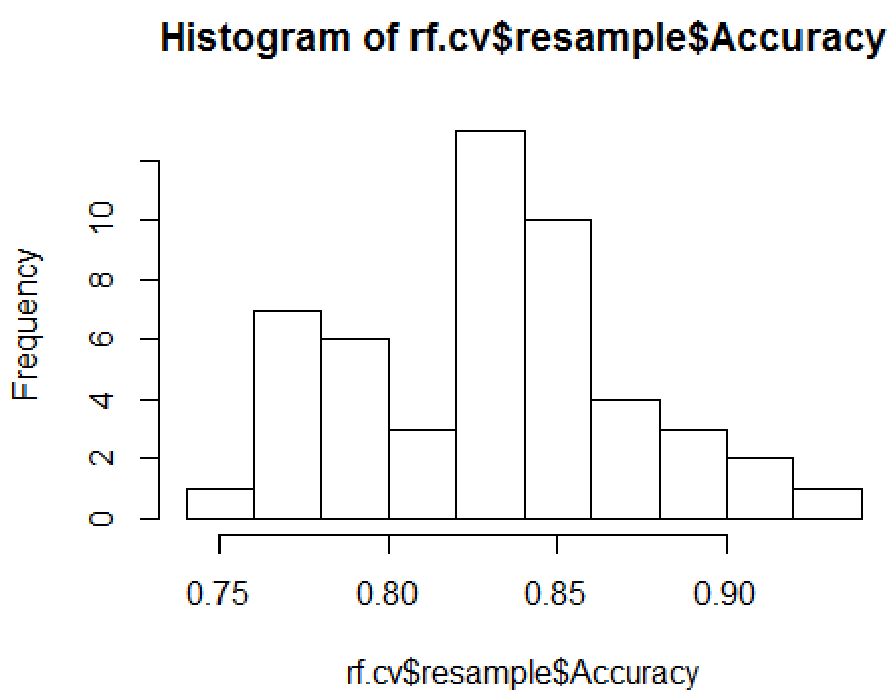
```r
cat(paste("\nMin : ", min(rf.cv$resample$Accuracy), "Max: ", max(rf.cv$resamp
le$Accuracy), "\n", sep = " "))
```

```
##
## Min :  0.752808988764045 Max:  0.921348314606742
```

```r
cat(paste("\nCross validation Folds:",
          rf.cv$resample$Accuracy,  sep = " "))
```

```
##
## Cross validation Folds: 0.766666666666667
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.829545454545455
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.855555555555556
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.788888888888889
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.911111111111111
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.840909090909091
## Cross validation Folds: 0.852272727272727
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.811111111111111
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.8
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.921348314606742
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.866666666666667
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.820224719101124
```

```
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.833333333333333
## Cross validation Folds: 0.752808988764045
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.818181818181818
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.863636363636364
## Cross validation Folds: 0.910112359550562
## Cross validation Folds: 0.797752808988764

# Display histogram with crossvalidation means to see varaibility and the dis
tribution
hist(rf.cv$resample$Accuracy)
```




```
# Pull out the the trained model using the best parameters on
rf.best <- rf.cv$finalModel

# Look at the model - this model is trained on 100% of the
# training data!
print(rf.best)

##
## Call:
##  randomForest(x = x, y = y, ntree = 150, mtry = param$mtry)
```

```
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 17.28%
## Confusion matrix:
##     X1  X2 class.error
## X1 491  58   0.1056466
## X2  96 246   0.2807018
```

```r
# Predict with the blindfold data for submission to Kaggle
Prediction <- predict(rf.cv, test, OOB=TRUE, type = "raw")

# Verify the number of prediction (418)
length(Prediction)
```

```
## [1] 418
```

```r
# glympse preds
glimpse(Prediction)
```

```
##  Factor w/ 2 levels "X1","X2": 1 2 1 1 2 1 2 1 2 1 ...
```

```r
# Save results
result<-Prediction


# Save Model to disk
rf.Model<-saveRDS(rf.cv, "rfTitanicModel.RDS")

# String to help identify the type of model than the output file has
selModel<-"RF"

# COnvert Predictions to the required Kaggle submission
Prediction<-ifelse(Prediction=="X1", 0, 1)

# Create dataframe shaped for Kaggle
submission <- data.frame(PassengerId = test.Id,
                          Survived = Prediction)
# Create string with date to be part of the name
dateCreated<-gsub(':', '_', Sys.time())

# Construct File Name
kaggleFilename<-paste0(selModel,"_kaggle_submission-", "-",dateCreated ,".csv
")

# Write out a .CSV suitable for Kaggle submission
write.csv(submission, file = kaggleFilename, row.names = FALSE)
```

```
e_time<-Sys.time()
print(e_time)

## [1] "2018-05-12 07:57:33 CDT"

t_time<-e_time-b_time
print (paste("Total time :", t_time))

## [1] "Total time : 49.8676919937134"
```

Train Logistic Regression Model 0.77033 Kaggle

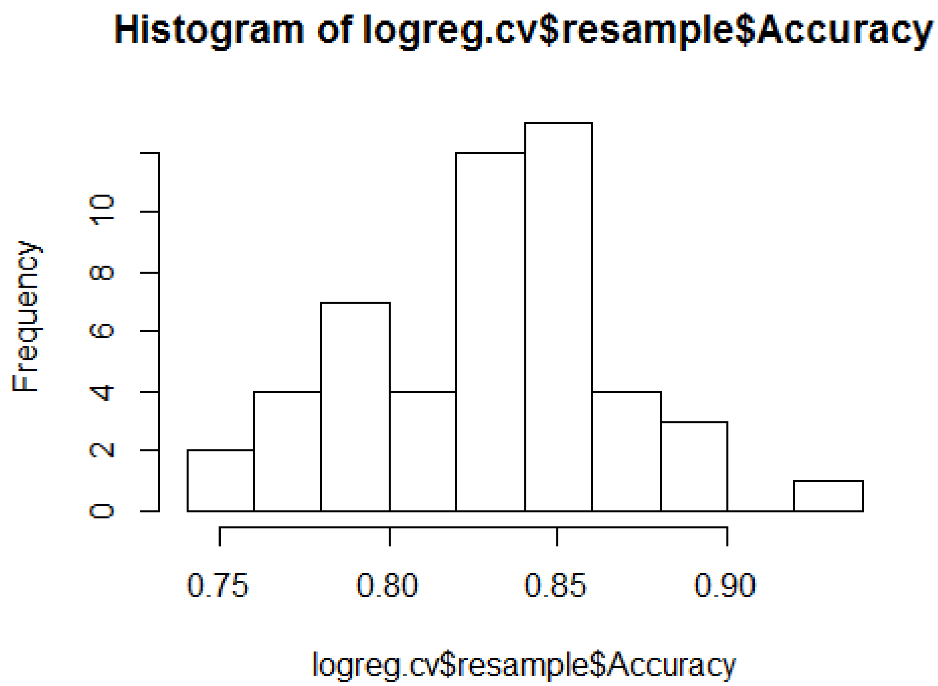
```
b_time<-Sys.time()
print(b_time)

## [1] "2018-05-12 07:57:33 CDT"

# Set seed to ensure reproducibility between runs
set.seed(12345)


# Set up caret to perform 10-fold cross validation repeated 3 times
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              classProbs=TRUE,
                              verboseIter = FALSE,
                              savePredictions = TRUE)


cl <- makeCluster(10, type = "SOCK") #10 parallel processes-RStudio running a
t the same time


# Register cluster so that caret will know to train in parallel.
registerDoSNOW(cl)


# Use caret to train a rpart decision trees using 10-fold cross
# validation repeated 3 times and use 7 values for tuning the
# cp hyperparameter. This code returns the best model trained on
# all the data using the best value of cp! Mighty!
logreg.cv <- train(target ~ .,
                  data = X_train,
                  method =  "glm",
                  family ="binomial",
                  trControl = caret.control
                  )

stopCluster(cl)
```

```
# Display the results of the cross validation run -
# mean accuracy!
logreg.cv

## Generalized Linear Model
##
## 891 samples
##  21 predictor
##   2 classes: 'X1', 'X2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 802, 803, 802, 802, 802, 802, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8276088  0.630864

# Make predictions
preds <- predict(logreg.cv, X_test)

# Use caret's confusionMatrix() function to estimate the
# effectiveness of this model on unseen, new data.
confusion.matrix<-confusionMatrix(preds, X_test$target)

print(confusion.matrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  X1  X2
##         X1 142  31
##         X2  22  71
##
##                Accuracy : 0.8008
##                  95% CI : (0.7476, 0.847)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 7.74e-11
##
##                   Kappa : 0.5715
##  Mcnemar's Test P-Value : 0.2718
##
##             Sensitivity : 0.8659
##             Specificity : 0.6961
##          Pos Pred Value : 0.8208
##          Neg Pred Value : 0.7634
##              Prevalence : 0.6165
##          Detection Rate : 0.5338
##    Detection Prevalence : 0.6504
##       Balanced Accuracy : 0.7810
```

```
##
##        'Positive' Class : X1
##

# What is the standard deviation?
cat(paste("\nCross validation standard deviation:",
          sd(logreg.cv$resample$Accuracy), "\n", sep = " "))

##
## Cross validation standard deviation: 0.037629342862058

cat(paste("\nCross validation mean:",
          mean(logreg.cv$resample$Accuracy), "\n", sep = " "))

##
## Cross validation mean: 0.827608841221201

cat(paste("\nMin : ", min(logreg.cv$resample$Accuracy), "Max: ", max(logreg.c
v$resample$Accuracy), "\n", sep = " "))

##
## Min :  0.752808988764045 Max:  0.921348314606742

cat(paste("\nCross validation Folds:",
          logreg.cv$resample$Accuracy,  sep = " "))

##
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.829545454545455
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.833333333333333
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.833333333333333
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.840909090909091
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.876404494382023
## Cross validation Folds: 0.855555555555556
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.752808988764045
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.921348314606742
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.853932584269663
```

```
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.795454545454545
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.863636363636364
## Cross validation Folds: 0.9
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.755555555555556
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.840909090909091
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.876404494382023
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.775280898876405
```

```r
dateCreated<-gsub(':', '_', Sys.time())
#pdf(paste0("logRegAccuracyHistogram-",node_name, "-",dateCreated,".pdf"))
hist(logreg.cv$resample$Accuracy)
```




```
#dev.off()
# Pull out the the trained model using the best parameters on
# all the data! Mighty!p
logreg.best <- logreg.cv$finalModel


# Look at the model - this model is trained on 100% of the
# training data!
print(logreg.best)

##
## Call:  NULL
##
## Coefficients:
##     (Intercept)              Age            SibSp            Parch
##         -0.7766          -0.3389          -0.6154          -0.4156
##            Fare       FamilySize        Pclass1X2        Pclass2X2
##          0.1748          -2.1466           2.1122           1.0323
##       Pclass3X1      SexfemaleX2        SexmaleX1       EmbarkedCX2
##              NA          16.7669               NA           0.3455
##     EmbarkedQX2      EmbarkedSX1    WithFamilyNX2    WithFamilyYX1
##          0.1456               NA          -3.8254               NA
##     TitleLadyX2    TitleMasterX2       TitleMsX2       TitleSirX1
##        -13.1771           3.4392         -13.9366               NA
## FamilyIDLargeX2  FamilyIDSmallX1
##          1.6088               NA
##
```

```
## Degrees of Freedom: 890 Total (i.e. Null);  875 Residual
## Null Deviance:       1187
## Residual Deviance: 718.5     AIC: 750.5

Prediction <- predict(logreg.cv, test, OOB=TRUE, type = "prob")



length(Prediction$X1)

## [1] 418

result<-Prediction




# Save Model

LogReg.Model<-saveRDS(logreg.cv, "LogRegTitanicModel.RDS")

selModel<-"glm"

# Convert Prediction to Kaggle requirement
Prediction<-ifelse(Prediction$X1>0.5, 0, 1)

# Create dataframe shaped for Kaggle
submission <- data.frame(PassengerId = test.Id,
                         Survived = Prediction)

dateCreated<-gsub(':', '_', Sys.time())
kaggleFilename<-paste0(selModel,"_kaggle_submission-", "-",dateCreated ,".csv
")
# Write out a .CSV suitable for Kaggle submission
write.csv(submission, file = kaggleFilename, row.names = FALSE)

e_time<-Sys.time()
print(e_time)

## [1] "2018-05-12 07:57:53 CDT"

t_time<-e_time-b_time
print (paste("Total time :", t_time))

## [1] "Total time : 20.00546002388"
```

Train Naive Bayes Classification Model 0.75598 Kaggle

```
b_time<-Sys.time()
print(b_time)
```

```
## [1] "2018-05-12 07:57:53 CDT"

# Set seed to ensure reproducibility between runs
set.seed(12345)


# Set up caret to perform 10-fold cross validation repeated 5 times
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              verboseIter = FALSE,
                              savePredictions = TRUE)


cl <- makeCluster(10, type = "SOCK") #10 parallel processes-RStudio running a
t the same time


# Register cluster so that caret will know to train in parallel.
registerDoSNOW(cl)


## Use Naive Bayes Classifier Model
nbayes.cv <- train(target ~ .,
                  data = X_train,
                  method =  "nb",
                  trControl = caret.control
                  )

stopCluster(cl)


# Make predictions
preds <- predict(nbayes.cv, X_test)

# Get & Display Confusion Matrix
confusion.matrix<-confusionMatrix(preds, X_test$target)
print(confusion.matrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  X1  X2
##         X1 135  25
##         X2  29  77
##
##                Accuracy : 0.797
##                  95% CI : (0.7436, 0.8437)
##     No Information Rate : 0.6165
```

```
##     P-Value [Acc > NIR] : 1.928e-10
##
##                   Kappa : 0.5738
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.8232
##             Specificity : 0.7549
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.7264
##              Prevalence : 0.6165
##          Detection Rate : 0.5075
##    Detection Prevalence : 0.6015
##       Balanced Accuracy : 0.7890
##
##        'Positive' Class : X1
##
```

```r
# What is the standard deviation?
cat(paste("\nCross validation standard deviation:",
          sd(nbayes.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation standard deviation: 0.0408307409632427
```

```r
cat(paste("\nCross validation mean:",
          mean(nbayes.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation mean: 0.799088298717512
```

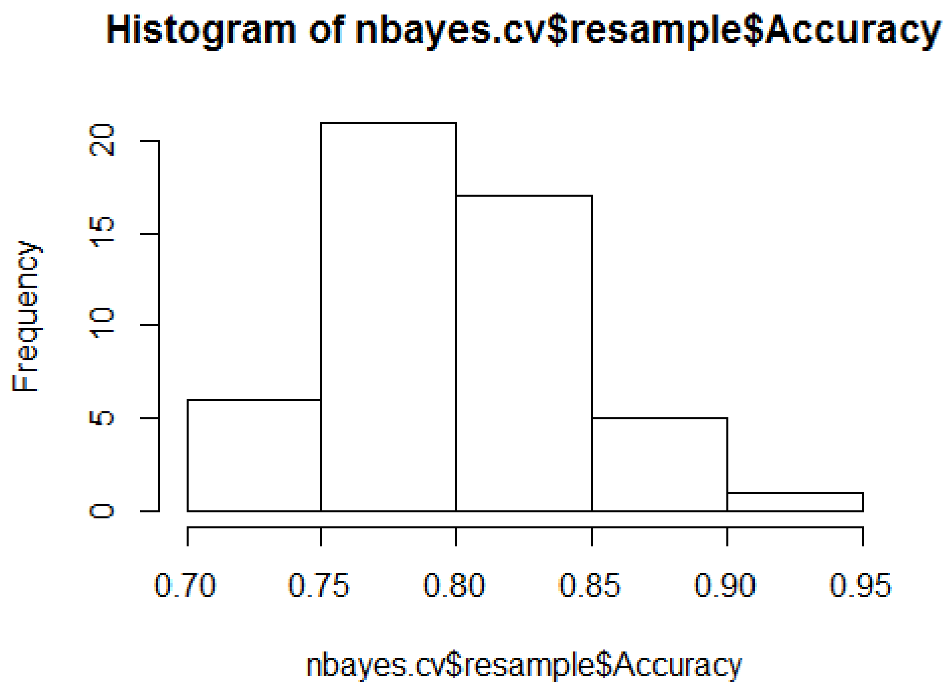
```r
cat(paste("\nMin : ", min(nbayes.cv$resample$Accuracy), "Max: ", max(nbayes.c
v$resample$Accuracy), "\n", sep = " "))
```

```
##
## Min :  0.707865168539326 Max:  0.911111111111111
```

```r
cat(paste("\nCross validation Folds:",
          nbayes.cv$resample$Accuracy,  sep = " "))
```

```
##
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.741573033707865
## Cross validation Folds: 0.811111111111111
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.75
## Cross validation Folds: 0.811111111111111
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.777777777777778
## Cross validation Folds: 0.730337078651685
```

```
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.818181818181818
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.876404494382023
## Cross validation Folds: 0.876404494382023
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.806818181818182
## Cross validation Folds: 0.752808988764045
## Cross validation Folds: 0.911111111111111
## Cross validation Folds: 0.766666666666667
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.761363636363636
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.744444444444444
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.795454545454545
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.744444444444444
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.788888888888889
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.811111111111111
## Cross validation Folds: 0.707865168539326

# Histogram of Crossvalidation
hist(nbayes.cv$resample$Accuracy)
```




```
# Use the best model
nbayes.best <- nbayes.cv$finalModel

# Predict with blind folded data for Kaggle
Prediction <- predict(nbayes.cv, test, OOB=TRUE, type = "raw")

# Verify # of predictions
length(Prediction)

## [1] 418

# Save Model to Disk
nbayes.Model<-saveRDS(nbayes.cv, "nbayesTitanicModel.RDS")

selModel<-"NB"

# Convert Prediction to Kaggle requirement
Prediction<-ifelse(Prediction=="X1", 0, 1)

# Get predictions
result<-Prediction

# Create dataframe shaped for Kaggle
submission <- data.frame(PassengerId = test.Id,
                         Survived = Prediction)
```

```r
dateCreated<-gsub(':', '_', Sys.time())
kaggleFilename<-paste0(selModel,"_kaggle_submission-", "-",dateCreated ,".csv
")
# Write out a .CSV suitable for Kaggle submission
write.csv(submission, file = kaggleFilename, row.names = FALSE)

e_time<-Sys.time()
print(e_time)

## [1] "2018-05-12 07:58:16 CDT"

t_time<-e_time-b_time
print (paste("Total time :", t_time))

## [1] "Total time : 23.0797250270844"
```

Train Support Vector Machine Linear Model 0.78468 Kaggle

```r
b_time<-Sys.time()
print(b_time)

## [1] "2018-05-12 07:58:16 CDT"

# Set seed to ensure reproducibility between runs
set.seed(12345)


# Set up caret to perform 10-fold cross validation repeated 5 times
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              search = "grid",
                              classProbs=TRUE,
                              verboseIter = FALSE,
                              savePredictions = TRUE)

# Search Grid
tune.grid <- expand.grid(mtry=c(1:10))

cl <- makeCluster(10, type = "SOCK") #10 parallel processes-RStudio running a
t the same time


# Register cluster so that caret will know to train in parallel.
registerDoSNOW(cl)


# Train Model
svm.cv <- train(target ~ .,
                data = X_train,
                method =  "svmLinear",
```

```r
                  trControl = caret.control
                  )

stopCluster(cl)

# Make predictions/Validate
preds <- predict(svm.cv, X_test)

# Get & Display COnfusion Matrix
confusion.matrix<-confusionMatrix(preds, X_test$target)
print(confusion.matrix)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  X1  X2
##         X1 144  33
##         X2  20  69
##
##                Accuracy : 0.8008
##                  95% CI : (0.7476, 0.847)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 7.74e-11
##
##                   Kappa : 0.5682
##  Mcnemar's Test P-Value : 0.09929
##
##             Sensitivity : 0.8780
##             Specificity : 0.6765
##          Pos Pred Value : 0.8136
##          Neg Pred Value : 0.7753
##              Prevalence : 0.6165
##          Detection Rate : 0.5414
##    Detection Prevalence : 0.6654
##       Balanced Accuracy : 0.7773
##
##        'Positive' Class : X1
##
```

```r
# What is the standard deviation?
cat(paste("\nCross validation standard deviation:",
          sd(svm.cv$resample$Accuracy), "\n", sep = " "))
```

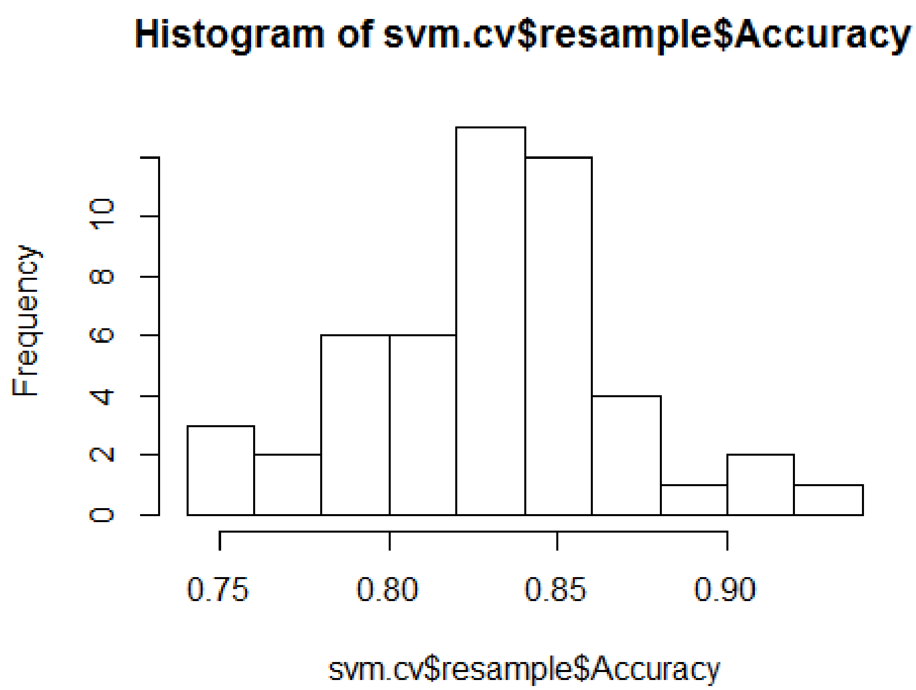
```
##
## Cross validation standard deviation: 0.0377421165066087
```

```r
cat(paste("\nCross validation mean:",
          mean(svm.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation mean: 0.828285438656225
```

```r
cat(paste("\nMin : ", min(svm.cv$resample$Accuracy), "Max: ", max(svm.cv$resa
mple$Accuracy), "\n", sep = " "))

##
## Min :  0.752808988764045 Max:  0.921348314606742

cat(paste("\nCross validation Folds:",
          svm.cv$resample$Accuracy,  sep = " "))

##
## Cross validation Folds: 0.910112359550562
## Cross validation Folds: 0.818181818181818
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.8
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.852272727272727
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.855555555555556
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.752808988764045
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.921348314606742
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.788888888888889
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.818181818181818
## Cross validation Folds: 0.822222222222222
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.840909090909091
## Cross validation Folds: 0.911111111111111
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.755555555555556
## Cross validation Folds: 0.820224719101124
```

```
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.811111111111111
## Cross validation Folds: 0.829545454545455
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.855555555555556
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.752808988764045

# Display histogram of Crossvalidation
hist(svm.cv$resample$Accuracy)
```




```
# Use best model
svm.best <- svm.cv$finalModel

# Predict with blind folded data for Kaggle
Prediction <- predict(svm.cv, test, OOB=TRUE, type = "raw")

# Verify # of predictions
length(Prediction)

## [1] 418

# Save Model to Disk
svm.Model<-saveRDS(svm.cv, "svmTitanicModel.RDS")
```

```
selModel<-"SVML"

# Convert Prediction to Kaggle requirement
Prediction<-ifelse(Prediction=="X1", 0, 1)

# Get predictions
result<-Prediction

# Create dataframe shaped for Kaggle
submission <- data.frame(PassengerId = test.Id,
                          Survived = Prediction)

dateCreated<-gsub(':', '_', Sys.time())
kaggleFilename<-paste0(selModel,"_kaggle_submission-", "-",dateCreated ,".csv
")
# Write out a .CSV suitable for Kaggle submission
write.csv(submission, file = kaggleFilename, row.names = FALSE)

e_time<-Sys.time()
print(e_time)

## [1] "2018-05-12 07:58:41 CDT"

t_time<-e_time-b_time
print (paste("Total time :", t_time))

## [1] "Total time : 25.4220559597015"
```

Train Neural Network Model 0.78947 Kaggle (Caret Package)

```
b_time<-Sys.time()
print(b_time)

## [1] "2018-05-12 07:58:41 CDT"

# Set seed to ensure reproducibility between runs
set.seed(12345)


# Set up caret to perform 10-fold cross validation repeated 5 times
caret.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              search = "grid",
                              classProbs=TRUE,
                              verboseIter = FALSE,
                              savePredictions = TRUE)

# Search Grid
tune.grid <- expand.grid(mtry=c(1:10))
```

```
cl <- makeCluster(10, type = "SOCK") #10 parallel processes-RStudio running a
t the same time


# Register cluster so that caret will know to train in parallel.
registerDoSNOW(cl)


# Use caret to train a nn using 10-fold cross
nnet.cv <- train(target ~ .,
                 data = X_train,
                 method =  "nnet",
                 trControl = caret.control
                 )

## # weights:  24
## initial  value 615.043160
## iter  10 value 461.807313
## iter  20 value 436.987866
## iter  30 value 386.955607
## iter  40 value 355.640923
## iter  50 value 354.710057
## iter  60 value 354.587495
## iter  70 value 354.565494
## iter  70 value 354.565494
## iter  70 value 354.565494
## final  value 354.565494
## converged

stopCluster(cl)

# Make predictions/Validation
preds <- predict(nnet.cv, X_test)

## Get & Display Confusion Matrix
confusion.matrix<-confusionMatrix(preds, X_test$target)
print(confusion.matrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  X1  X2
##         X1 147  37
##         X2  17  65
##
##                Accuracy : 0.797
##                  95% CI : (0.7436, 0.8437)
##     No Information Rate : 0.6165
##     P-Value [Acc > NIR] : 1.928e-10
```

```
##
##                   Kappa : 0.5541
##  Mcnemar's Test P-Value : 0.009722
##
##             Sensitivity : 0.8963
##             Specificity : 0.6373
##          Pos Pred Value : 0.7989
##          Neg Pred Value : 0.7927
##              Prevalence : 0.6165
##          Detection Rate : 0.5526
##    Detection Prevalence : 0.6917
##       Balanced Accuracy : 0.7668
##
##        'Positive' Class : X1
##
```

```
# What is the standard deviation?
cat(paste("\nCross validation standard deviation:",
          sd(nnet.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation standard deviation: 0.0389500249871733
```

```
cat(paste("\nCross validation mean:",
          mean(nnet.cv$resample$Accuracy), "\n", sep = " "))
```

```
##
## Cross validation mean: 0.82627772103053
```

```
cat(paste("\nMin : ", min(nnet.cv$resample$Accuracy), "Max: ", max(nnet.cv$re
sample$Accuracy), "\n", sep = " "))
```

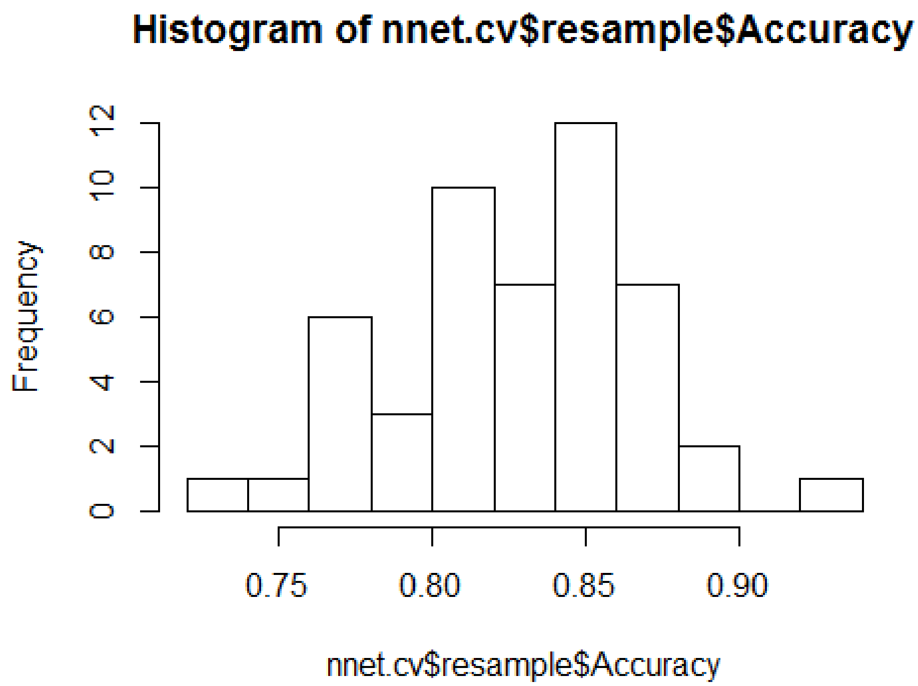
```
##
## Min :  0.733333333333333 Max:  0.921348314606742
```

```
cat(paste("\nCross validation Folds:",
          nnet.cv$resample$Accuracy,  sep = " "))
```

```
##
## Cross validation Folds: 0.829545454545455
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.811111111111111
## Cross validation Folds: 0.852272727272727
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.806818181818182
## Cross validation Folds: 0.786516853932584
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.877777777777778
```

```
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.733333333333333
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.831460674157303
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.777777777777778
## Cross validation Folds: 0.820224719101124
## Cross validation Folds: 0.853932584269663
## Cross validation Folds: 0.766666666666667
## Cross validation Folds: 0.752808988764045
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.764044943820225
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.921348314606742
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.876404494382023
## Cross validation Folds: 0.887640449438202
## Cross validation Folds: 0.833333333333333
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.844444444444444
## Cross validation Folds: 0.806818181818182
## Cross validation Folds: 0.865168539325843
## Cross validation Folds: 0.842696629213483
## Cross validation Folds: 0.775280898876405
## Cross validation Folds: 0.818181818181818
## Cross validation Folds: 0.898876404494382
## Cross validation Folds: 0.797752808988764
## Cross validation Folds: 0.808988764044944
## Cross validation Folds: 0.808988764044944
```

```
# Display Crossvalidations histogram
hist(nnet.cv$resample$Accuracy)
```




```r
# Select Best Model
nnet.best <- nnet.cv$finalModel

# Predict with Blind Folded Data
Prediction <- predict(nnet.cv, test, OOB=TRUE, type = "raw")

selModel<-"NN"

# Convert Prediction to Kaggle requirement
Prediction<-ifelse(Prediction=="X1", 0, 1)

# Verify # of predictions
length(Prediction)

## [1] 418

# Get predictions
result<-Prediction

# Save Model to Disk
nnet.Model<-saveRDS(nnet.cv, "nnetTitanicModel.RDS")

# Create dataframe shaped for Kaggle
submission <- data.frame(PassengerId = test.Id,
                         Survived = Prediction)
```

```
dateCreated<-gsub(':', '_', Sys.time())
kaggleFilename<-paste0(selModel,"_kaggle_submission-", "-",dateCreated ,".csv
")
# Write out a .CSV suitable for Kaggle submission
write.csv(submission, file = kaggleFilename, row.names = FALSE)

e_time<-Sys.time()
print(e_time)

## [1] "2018-05-12 07:59:12 CDT"

t_time<-e_time-b_time
print (paste("Total time :", t_time))

## [1] "Total time : 30.1739690303802"
```

Conclusion

Now is time to compare all models metrics Accuracy for this data set since that is what Kaggle is measuring. Look at the histograms and review Normality, standard deviation (Variation), Range of the Crossvalidation folds results, etc.

What do we want to see?

- Crossvalidation results with a tight/small range.

- Small Standard deviation

- Normalized distribution of the crossvalidation results of accuracy (Bell Shape)

- A good KAPPA (See note above about Kappa guidelines)

- Generalized Model. We want to ensure the changes to the models/data improve the Kaggle Score.

In few words, we want our models to perform consistently in new/unknown data.

- Avoid Overfitting- If the scores improve in our model with the validation data but they don't improve when submitting to Kaggle, most likely, we are overfitting

- Avoid Underfitting. Make sure you do not remove key data, you will see the Accuracy going down and the variation increasing

- Remove features that have no variation

- Remove features that are highly correlated.

Also, is time to read about the hyperparameters of each model and fine tune them. Check results and so on. Because this class appear to be unbalanced, I would suggest using SMOTE to do oversampling of the imbalance class. As I mentioned before, the feature engineering was not intensive, so that is another area to consider improving. Furthermore, in this presentation notebook I did not perform enough visualization or used features importance tools as in my second version. **In my Second Version, I achieved Position 777 (Top 6.9%), with a 0.80861 which I achieved with a Random Forest and by doing some feature engineering + SMOTE.** So, also that is something you can do to find more insights and improve the model. If you end with too many features after Hot-1 Encoding, consider using the Principal Component Analysis (PCA) package to reduce dimensionality. If you would like to learn about Stack Ensemble Models (A single Model learning from other Models), see my Stack Ensemble Titanic Classifiers Notebook coming soon.

Miguel Hidalgo is a Life Long Learner, Data Science Enthusiast, that holds a MBA in E-Commerce, a BS in Computer Science, a BS in Professional Aeronautics, PMP Certification, Agile Foundation Certification, ITIL Foundations Cert., Black Belt Lean Six Sigma, ISO9001-2015 Lead Auditor Cert., Aircraft Electrician, Hovercraft Electrician, …, I told you, LIFE LONG LEARNER. In my work, one of the many huts I wear is Data Analysis and Modeling in order to improve the organization. LinkedIn-> https://www.linkedin.com/in/mhidalgo/