EDX Capstone Project - Titanic Survival Prediction

Thiago do Couto

13/06/2019

R Capstone - Standalone Project

Titanic Survival Prediction

This project aims to predict if a certain passenger would survive the Titanic disaster and show the importance of each variable.

1 - Load required libraries

We'll use Random Forest algorithm in this prediction project, the Grammar of Graphics to the Importance Plot and Dplyr to use the Glimpse function.

```
# Load required libraries
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

2 - Load datasets

```
# Load datasets
train <- read.csv("train.csv", stringsAsFactors = TRUE)
test <- read.csv("test.csv",  stringsAsFactors = TRUE)
```

3 - Exploratory Data Analysis

Verify NAs in both sets.

```
# Verify NAs
colSums(is.na(train))
```

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

```
colSums(is.na(test))
```

```
## PassengerId      Pclass        Name         Sex         Age       SibSp
##           0           0           0           0          86           0
##       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           1           0           0
```

Create the Target variable (Survival) in Test Set.

```
# Create target variable in Test set
test$Survived <- NA
```

Create variable 'IsTrainSet' to track if the observation is from Test or Train set.

```
# Create variable to track if the observation is from Test or Train set
train$IsTrainSet <- TRUE
test$IsTrainSet <- FALSE
```

Group datasets so that we can work with it.

```
# Group datasets
full_df <- rbind(train, test)
```

Let’s take a macro view of the set.

```
glimpse(full_df)
```

```
## Observations: 1,309
## Variables: 13
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1…
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1…
## $ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2…
## $ Name        <fct> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradl…
## $ Sex         <fct> male, female, female, female, male, male, male, male…
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39…
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0…
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0…
## $ Ticket      <fct> A/5 21171, PC 17599, STON/O2. 3101282, 113803, 37345…
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51…
## $ Cabin       <fct> , C85, , C123, , , E46, , , , G6, C103, , , , , , , …
## $ Embarked    <fct> S, C, S, S, S, Q, S, S, S, C, S, S, S, S, S, S, Q, S…
## $ IsTrainSet  <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE…
```

As we cah see, there are variables that have data types that hold us to work with them. We’ll treat them soon.

So, lets make a summary analysis.

```
# Dataframe summary
summary(full_df)
```

```
##   PassengerId      Survived          Pclass
##  Min.   :   1   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 328   1st Qu.:0.0000   1st Qu.:2.000
##  Median : 655   Median :0.0000   Median :3.000
##  Mean   : 655   Mean   :0.3838   Mean   :2.295
##  3rd Qu.: 982   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1309   Max.   :1.0000   Max.   :3.000
##                 NA's   :418
##                                  Name          Sex           Age
##  Connolly, Miss. Kate               :   2   female:466   Min.   : 0.17
##  Kelly, Mr. James                   :   2   male  :843   1st Qu.:21.00
##  Abbing, Mr. Anthony                :   1                Median :28.00
##  Abbott, Mr. Rossmore Edward        :   1                Mean   :29.88
##  Abbott, Mrs. Stanton (Rosa Hunt):   1                3rd Qu.:39.00
##  Abelson, Mr. Samuel                :   1                Max.   :80.00
##  (Other)                            :1301                NA's   :263
##      SibSp            Parch            Ticket          Fare
##  Min.   :0.0000   Min.   :0.000   CA. 2343:  11   Min.   :  0.000
##  1st Qu.:0.0000   1st Qu.:0.000   1601    :   8   1st Qu.:  7.896
##  Median :0.0000   Median :0.000   CA 2144 :   8   Median : 14.454
##  Mean   :0.4989   Mean   :0.385   3101295 :   7   Mean   : 33.295
##  3rd Qu.:1.0000   3rd Qu.:0.000   347077  :   7   3rd Qu.: 31.275
##  Max.   :8.0000   Max.   :9.000   347082  :   7   Max.   :512.329
##                                   (Other) :1261   NA's   :1
##              Cabin      Embarked IsTrainSet
##                 :1014    :  2    Mode :logical
##  C23 C25 C27    :   6   C:270    FALSE:418
##  B57 B59 B63 B66:   5   Q:123    TRUE :891
##  G6             :   5   S:914
##  B96 B98        :   4
##  C22 C26        :   4
##  (Other)        : 271
```

As we can see, there is 1 NA in Fare, 418 NAs in Survived (because of the Test set), 263 NAs in Age and 2 NAs in Embarked. We'll deal with them later.

Let's analyse specifically the NAs.

```
# Check for invalid data
colSums(is.na(full_df))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0         418           0           0           0         263
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           1           0           0
##  IsTrainSet
##           0
```

So, let's treat them accordingly.

4 - Data Transformations

Lets initially treat the NAs observations.

As there are some occurences for ordinal data, we'll use the MEDIAN value to fill the fields.

```
# As there are some occurences for ordinal data, we'll use the MEDIAN value to fill t
he fields.
full_df$Age[is.na(full_df$Age)] <- median(full_df$Age, na.rm = TRUE)
full_df$Fare[is.na(full_df$Fare)] <- median(full_df$Fare, na.rm = TRUE)
```

As there are 2 occurrences of NAs in Embarked, we'll use the most common value to fill the fields.

```
# As there are 2 occurrences of NAs in Embarked, we'll use the most common value to f
ill the fields.
full_df$Embarked[full_df$Embarked==""] <-"S"
```

As foretold, there are some classes that can avoid us to work accordingly with the data.

Coerce data types to factor (when categorical) and to numeric (when ordinal).

```
# Coerce data types to factor (when categorical) and to numeric (when ordinal).
full_df$Survived <- as.factor(full_df$Survived)
full_df$Pclass <- as.factor(full_df$Pclass)
full_df$SibSp <- as.numeric(full_df$SibSp)
full_df$Parch <- as.numeric(full_df$Parch)
full_df$Embarked <- as.factor(as.character(full_df$Embarked))
```

5 - The Random Forest Model

Now that we have the dataset treated, let's build the model.

```
# Building the model
train_set <- full_df[full_df$IsTrainSet == TRUE, ]
test_set <- full_df[full_df$IsTrainSet == FALSE, ]
rf_model <- randomForest(formula = as.formula("Survived ~ Sex + Pclass + Age + SibSp
 + Parch + Fare + Embarked"), data = train_set, ntree = 50, importance = TRUE)
```

Let's visualize the model results.

```
# Visualizing the model
rf_model
```

```
##
## Call:
##  randomForest(formula = as.formula("Survived ~ Sex + Pclass + Age + SibSp + Parch
+ Fare + Embarked"),      data = train_set, ntree = 50, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 17.4%
## Confusion matrix:
##     0   1 class.error
## 0 494  55   0.1001821
## 1 100 242   0.2923977
```

```
plot(rf_model)
```

There we can sse the model error and accuracy.

Let's generate the importance Matrix of the variables.

```
# Gerenating importance matrix
importance_var <- importance(rf_model, type = 1)
importance_var
```

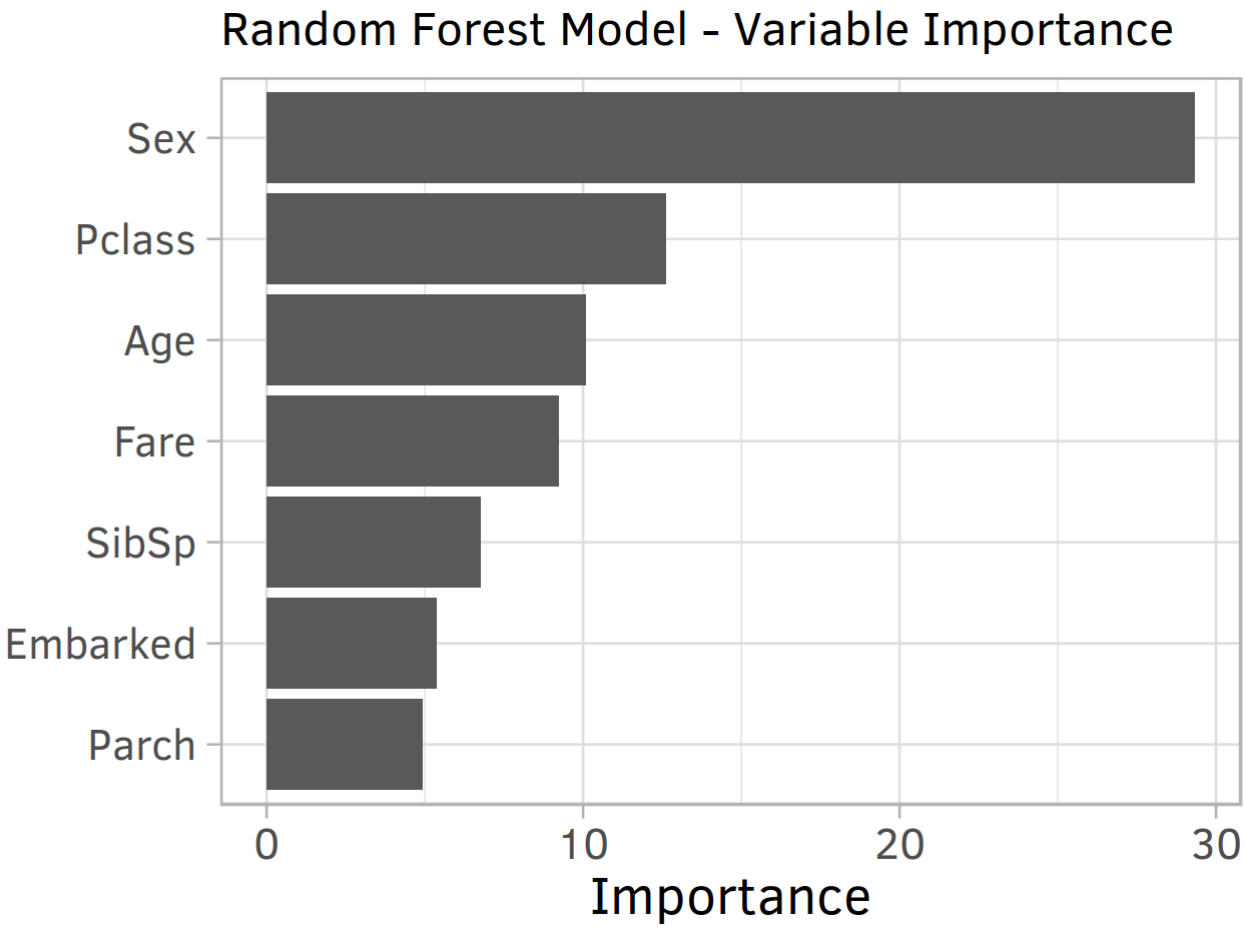
```
##          MeanDecreaseAccuracy
## Sex                 29.335926
## Pclass              12.616504
## Age                 10.076700
## SibSp                6.782148
## Parch                4.936870
## Fare                 9.224315
## Embarked             5.363491
```

Let's plot the graph of the Importance attributes. The higher the Importance the most it impacts the possibility of Survivability .

```
# Generating importance graph
importance_df <- data.frame(variables = row.names(importance_var), relevancy = import
ance_var[,1]);importance_df
```

```
##          variables relevancy
## Sex            Sex 29.335926
## Pclass      Pclass 12.616504
## Age            Age 10.076700
## SibSp        SibSp  6.782148
## Parch        Parch  4.936870
## Fare          Fare  9.224315
## Embarked  Embarked  5.363491
```

```
importance_graph <- ggplot(importance_df, aes(x=reorder(variables, relevancy), y = im
portance_var)) +
  geom_bar(stat="identity") +
  coord_flip() +
  theme_light(base_size = 20) +
  xlab("") +
  ylab("Importance") +
  ggtitle("Random Forest Model - Variable Importance") +
  theme(plot.title = element_text(size = 18))
importance_graph
```



We'll then generate the model versus data in the test set, removing the previous sent NAs with the correct prediction values.

```
# Create a Data Frame with PassengerID
final_df <- data.frame(PassengerId = test$PassengerId,
                       Survived = predict(rf_model, newdata = test_set))
View(final_df)
```