#Loading the data

# To load our data we re going to use `read.csv()` function.

# Since it is quite tricky in the middle of the work to spot a space " ",

# we re going to specify to `read.csv()` function that "", " " and NAs should be treated as NAs.

train\_url <- "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"

train <- read.csv (train\_url, stringsAsFactors = F,na.strings=c("","NA"," "))

# Import the testing set: test

test\_url <- "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"

test <- read.csv(test\_url, stringsAsFactors = F,na.strings=c("","NA"," "))

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

#Data Cleaning and preparation

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

# Combine train and test data for Data Cleaning

Full <- rbind(train,test)

test$Survived <- 0

Full <- rbind(train,test)

# Structure of the Full data

str(Full)

summary(Full)

#before substitutuing age get a new subset of data

# Subsetting the data - removing Passenger id, Name, Ticket No

Full <- subset(Full,select=c(2,3,5,6,7,8,10,12))

str(Full)

# Survival rates in absolute numbers

table(Full$Survived)

# Survival rates in proportions

prop.table(table(Full$Survived))

#detect missing values - Approach 1 (lengthy)

sum(is.na(Full$Age)==TRUE)/length(Full$Age)

sum(is.na(Full$Fare)==TRUE)/length(Full$Fare)

#get percentage of missing value of the attributes - - Approach 2 (Function)

sapply(Full, function(df)

{

sum(is.na(df)==T)/length(df)

})

# Missing Value Imputation - Age

# Substitute the missing values with the average value

Full$Age[is.na(Full$Age)] <- mean(Full$Age,na.rm=T)

sum(is.na(Full$Age))

# Missing Value Imputation - Fare

# Substitute the missing values with the average value

Full$Fare[is.na(Full$Fare)] <- mean(Full$Fare,na.rm=T)

sum(is.na(Full$Fare))

# Missing Value Imputation - Embarked

# Substitute the missing values with the mode value

table(Full$Embarked)

#Mode = S

Full$Embarked[is.na(Full$Embarked)] <- 'S'

sum(is.na(Full$Embarked))

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

#Data Exploration on cleaned data

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

str(Full)

# Survival rates in absolute numbers

table(Full$Survived)

# Create the column child, and indicate whether child or no child

Full$Child <- NA

Full$Child[Full$Age < 18] <- 1

Full$Child[Full$Age >= 18] <- 0

str(Full$Child)

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

# Model Building - Logistic Regression

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

# Data Cleaning is done, now we will again split back the data into train and test

# Train test splitting - Why do we need it?

train\_cleaned <- Full[1:891,]

test\_cleaned <- Full[892:1309,]

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

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

# Divide training data into train and test (8:2)

train.data <- train\_cleaned[1:712,]

test.data <- train\_cleaned[713:891,]

# You can use K-Fold Cross Validation for better result

str(train.data)

str(test.data)

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

# Model fitting - Try 1

logit\_model1 <- glm(Survived ~ .,family=binomial(link='logit'),data = train.data)

fitted.results <- predict(logit\_model1,newdata = test.data,type='response')

summary(logit\_model1) # AIC : 661.31

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

# Model fitting - Try 2

logit\_model2 <- glm(Survived ~ Pclass+Sex+Age+SibSp+Child,family=binomial(link='logit'),data = train.data)

fitted.results <- predict(logit\_model2,newdata=test.data,type='response')

summary(logit\_model2) # AIC : 659.19 -> Improvement

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

# Model fitting - Try 3

logit\_model3 <- glm(Survived ~ Pclass+Sex+Age+SibSp,family=binomial(link='logit'),data = train.data)

fitted.results <- predict(logit\_model3,newdata=test.data,type='response')

summary(logit\_model3)# AIC : 659.8 -> No Improvement -> Traiin with model2

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

# If prob > 0.5 then 1, else 0. Threshold can be set for better results

fitted.results <- ifelse(fitted.results > 0.5,1,0)

# Model Evaluation

# Confusion matrix

table(test.data$Survived, fitted.results)

misClasificError <- mean(fitted.results != test.data$Survived)

print(paste('Accuracy =',1-misClasificError))

# ROC-AUC Curve

install.packages("ROCR")

library(ROCR)

ROCRPred <- prediction(fitted.results, test.data$Survived)

ROCRPerf <- performance(ROCRPred, measure ="tpr", x.measure ="fpr")

plot(ROCRPerf)

plot(ROCRPerf, colorize = TRUE, print.cutoffs.at = seq(0.1,by=0.1),main = "ROC CURVE")

abline(a=0, b=1)

auc <- performance(ROCRPred, measure = "auc")

auc <- auc@y.values[[1]]

auc

auc <- round(auc, 4)

legend (.6,.4,auc, title = "AUC", cex =1)

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

# Make predictions on the test set

my\_prediction <- predict(logit\_model2, test, type = "response")

my\_prediction <- predict(logit\_model2, test\_cleaned, type = "response")

# If prob > 0.5 then 1, else 0. Threshold can be set for better results

my\_prediction <- ifelse(my\_prediction > 0.5,1,0)

# Finish the data.frame() call

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_prediction)

# Use nrow() on my\_solution - Should be 418

nrow(my\_solution)

str(my\_solution)

# Finish the write.csv() call

write.csv(my\_solution, file = "my\_solution\_logit.csv", row.names = FALSE)

Using Decision Tree

install.packages('rattle')

install.packages('rpart')

install.packages('rpart.plot')

install.packages('RColorBrewer')

library(rattle)

library(rpart.plot)

library(RColorBrewer)

library(rpart)

# Recreate the gender model

fit1 <- rpart(Survived ~ Sex, data=train\_cleaned, method="class")

fancyRpartPlot(fit1)

my\_Prediction <- predict(fit1, test\_cleaned, type = "class")

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "myfirstdtree.csv", row.names = FALSE)

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

# Build a deeper tree

fit2 <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Child,

data=train\_cleaned,

method="class")

fancyRpartPlot(fit2)

my\_Prediction <- predict(fit2, test\_cleaned, type = "class")

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "myDeepertree.csv", row.names = FALSE)

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

fit3 <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Child,

data=train\_cleaned,

method="class",

control=rpart.control(minsplit=2, cp=0))

fancyRpartPlot(fit3)

# Overfitting Problem

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

fit4 <- rpart(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,

data=train\_cleaned,

method="class",

control=rpart.control( minsplit=2, cp=0 ))

new.fit <- prp(fit4,snip=TRUE)$obj

fancyRpartPlot(new.fit)

my\_Prediction <- predict(new.fit, test, type = "class")

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "myDeepertree2.csv", row.names = FALSE)

Using Random Forest

# Install and load required packages for decision trees and forests

install.packages('randomForest')

install.packages('party')

library(randomForest)

library(rpart)

library(party)

str(train\_cleaned)

train\_cleaned$Survived <- as.factor(train\_cleaned$Survived)

train\_cleaned$Sex <- as.factor(train\_cleaned$Sex)

train\_cleaned$Embarked <- as.factor(train\_cleaned$Embarked)

test\_cleaned$Sex <- as.factor(test\_cleaned$Sex)

test\_cleaned$Embarked <- as.factor(test\_cleaned$Embarked)

# Build Random Forest Ensemble

set.seed(415)

fit <- randomForest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked+ Child,

data= train\_cleaned, importance=TRUE, ntree=2000)

# Look at variable importance

varImpPlot(fit)

my\_Prediction <- predict(fit, test\_cleaned)

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "firstforest.csv", row.names = FALSE)

# Build condition inference tree Random Forest

set.seed(415)

fit <- cforest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked+ Child,

data = train\_cleaned, controls=cforest\_unbiased(ntree=2000, mtry=3))

# Now let's make a prediction and write a submission file

my\_Prediction <- predict(fit, test\_cleaned, OOB=TRUE, type = "response")

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "ciforest.csv", row.names = FALSE)

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

Feature Engineering

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

#Data Cleaning and preparation

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

# Combine train and test data for Data Cleaning

test$Survived <- 0

Full <- rbind(train,test)

# Structure of the Full data

str(Full)

summary(Full)

# Survival rates in absolute numbers

table(Full$Survived)

# Survival rates in proportions

prop.table(table(Full$Survived))

#detect missing values - Approach 1 (lengthy)

sum(is.na(Full$Age)==TRUE)/length(Full$Age)

sum(is.na(Full$Fare)==TRUE)/length(Full$Fare)

#get percentage of missing value of the attributes - - Approach 2 (Function)

sapply(Full, function(df)

{

sum(is.na(df)==T)/length(df)

})

# Missing Value Imputation - Age

# Substitute the missing values with the average value

Full$Age[is.na(Full$Age)] <- mean(Full$Age,na.rm=T)

sum(is.na(Full$Age))

# Missing Value Imputation - Fare

# Substitute the missing values with the average value

Full$Fare[is.na(Full$Fare)] <- mean(Full$Fare,na.rm=T)

sum(is.na(Full$Fare))

# Missing Value Imputation - Embarked

# Substitute the missing values with the mode value

table(Full$Embarked)

#Mode = S

Full$Embarked[is.na(Full$Embarked)] <- 'S'

sum(is.na(Full$Embarked))

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

#Data Exploration on cleaned data

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

str(Full)

# Survival rates in absolute numbers

table(Full$Survived)

# Create the column child, and indicate whether child or no child

Full$Child <- NA

Full$Child[Full$Age < 18] <- 1

Full$Child[Full$Age >= 18] <- 0

str(Full$Child)

# Data Cleaning is done, now we will again split back the data into train and test

# Train test splitting - Why do we need it?

train\_cleaned <- Full[1:891,]

test\_cleaned <- Full[892:1309,]

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

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

# Divide training data into train and test (8:2)

train.data <- train\_cleaned[1:712,]

test.data <- train\_cleaned[713:891,]

str(train.data)

str(test.data)

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

# Feature Engineering

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

# Convert to a string

Full$Name <- as.character(Full$Name)

# Engineered variable 1: Title

# Extract the title - Mr, Mrs, Miss

Full$Title <- sapply(Full$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][2]})

Full$Title <- sub(' ', '', Full$Title) # Remove the white space or blank

table(Full$Title)

# Combine small title groups

Full$Title[Full$Title %in% c('Mme', 'Mlle')] <- 'Mlle'

Full$Title[Full$Title %in% c('Capt', 'Don', 'Major', 'Sir')] <- 'Sir'

Full$Title[Full$Title %in% c('Dona', 'Lady', 'the Countess', 'Jonkheer')] <- 'Lady'

# Convert to a factor

Full$Title <- factor(Full$Title)

table(Full$Title)

# Engineered variable 2: Family size

Full$FamilySize <- Full$SibSp + Full$Parch + 1

table(Full$FamilySize)

# Engineered variable 3: Family

Full$Surname <- sapply(Full$Name, FUN=function(x) {strsplit(x, split='[,.]')[[1]][1]})

Full$FamilyID <- paste(as.character(Full$FamilySize), Full$Surname, sep="")

Full$FamilyID[Full$FamilySize <= 2] <- 'Small'

# Delete erroneous family IDs

famIDs <- data.frame(table(Full$FamilyID))

famIDs <- famIDs[famIDs$Freq <= 2,]

Full$FamilyID[Full$FamilyID %in% famIDs$Var1] <- 'Small'

# Convert to a factor

Full$FamilyID <- factor(Full$FamilyID)

# Fill in Age NAs

summary(Full$Age)

Agefit <- rpart(Age ~ Pclass + Sex + SibSp + Parch + Fare + Embarked + Title + FamilySize,

data=Full[!is.na(Full$Age),], method="anova")

Full$Age[is.na(Full$Age)] <- predict(Agefit, Full[is.na(Full$Age),])

# Check what else might be missing

summary(Full)

# Fill in Embarked blanks

summary(Full$Embarked)

which(Full$Embarked == '')

Full$Embarked[c(62,830)] = "S"

Full$Embarked <- factor(Full$Embarked)

# Fill in Fare NAs

summary(Full$Fare)

which(is.na(Full$Fare))

Full$Fare[1044] <- median(Full$Fare, na.rm=TRUE)

# New factor for Random Forests, only allowed <32 levels, so reduce number

Full$FamilyID2 <- Full$FamilyID

# Convert back to string

Full$FamilyID2 <- as.character(Full$FamilyID2)

Full$FamilyID2[Full$FamilySize <= 3] <- 'Small'

# And convert back to factor

Full$FamilyID2 <- factor(Full$FamilyID2)

# Split back into test and train sets

train\_Featured <- Full[1:891,]

test\_Featured <- Full[892:1309,]

train\_Featured$Survived <- as.factor(train\_Featured$Survived)

train\_Featured$Sex <- as.factor(train\_Featured$Sex)

train\_Featured$Embarked <- as.factor(train\_Featured$Embarked)

test\_Featured$Sex <- as.factor(test\_Featured$Sex)

test\_Featured$Embarked <- as.factor(test\_Featured$Embarked)

# Build Random Forest Ensemble

set.seed(415)

fit <- randomForest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Child + Title + FamilySize + FamilyID2,

data=train\_Featured, importance=TRUE, ntree=2000)

# Look at variable importance

varImpPlot(fit)

# Now let's make a prediction and write a submission file

my\_Prediction <- predict(fit, test\_Featured)

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "Featuredfirstforest.csv", row.names = FALSE)

# Build condition inference tree Random Forest

set.seed(415)

fit <- cforest(as.factor(Survived) ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked + Child + Title + FamilySize + FamilyID,

data = train\_Featured, controls=cforest\_unbiased(ntree=2000, mtry=3))

# Now let's make a prediction and write a submission file

my\_Prediction <- predict(fit, test\_Featured, OOB=TRUE, type = "response")

my\_solution <- data.frame(PassengerId = test$PassengerId, Survived = my\_Prediction)

write.csv(my\_solution, file = "Featuredciforest.csv", row.names = FALSE)