Thoughts and motivations:

In this famous titanic dataset in Kaggle competition, I randomly select 70% as Training Sample, the rest as Test Sample. Also, I included predict dataset at the beginning to manipulate the two datasets together.

In titanic dataset, each passenger has its own attributes such as if survived, class level, name, age, gender, fare and so on. I did some transformation before splitting titanic dataset into training and test samples.

1. Group individual entry. Each row represents one passenger, Travel by ship could be in Family base, then we could infer those passengers are family members from Family Name and Family Size.
2. Get Mean Fare and work out the difference to Actual Fare. Calculate Mean fare on 3 different class levels, we could use other methods like calculate Median or put age also into consideration to get weighting.
3. Unite Embarked value. Replace Na value with empty Text (“”).

Rpart and mice could be used for categorized variables, and rpart only needs one predictor variable to be non-NA. There are age value missing in original titanic dataset, which could be simply avoid by rPart, however the age boundary will be set as 28 instead of further partition. Imputing data before classify age into four categories.

Set Itinerary as 50 (could be set more) , impute missing age value 5 times ( get 5 datasets), then we get 50\*5 models, the correction\_percentage\_matrix is list as below, which is tested for this method(rpart)’s stability.

Correction\_Percentage\_Matrix (50\*5)

[,1] [,2] [,3] [,4] [,5]

[1,] 0.8507463 0.8507463 0.8507463 0.8507463 0.8507463

[2,] 0.7835821 0.7835821 0.7910448 0.7835821 0.7835821

[3,] 0.8694030 0.8694030 0.8694030 0.8694030 0.8694030

[4,] 0.7910448 0.7910448 0.7910448 0.7910448 0.7910448

[5,] 0.8134328 0.8134328 0.8134328 0.8134328 0.8134328

[6,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[7,] 0.8358209 0.8358209 0.8358209 0.8358209 0.8358209

[8,] 0.7723881 0.7723881 0.7723881 0.7723881 0.7723881

[9,] 0.8097015 0.8097015 0.8097015 0.8134328 0.8097015

[10,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[11,] 0.8694030 0.8694030 0.8694030 0.8694030 0.8694030

[12,] 0.7910448 0.7910448 0.7910448 0.7910448 0.7910448

[13,] 0.8134328 0.8134328 0.8134328 0.8134328 0.8134328

[14,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[15,] 0.8358209 0.8358209 0.8358209 0.8358209 0.8358209

[16,] 0.7723881 0.7723881 0.7723881 0.7723881 0.7723881

[17,] 0.8097015 0.8097015 0.8097015 0.8134328 0.8097015

[18,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[19,] 0.8694030 0.8694030 0.8694030 0.8694030 0.8694030

[20,] 0.7910448 0.7910448 0.7910448 0.7910448 0.7910448

[21,] 0.8134328 0.8134328 0.8134328 0.8134328 0.8134328

[22,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[23,] 0.8358209 0.8358209 0.8358209 0.8358209 0.8358209

[24,] 0.7723881 0.7723881 0.7723881 0.7723881 0.7723881

[25,] 0.8097015 0.8097015 0.8097015 0.8134328 0.8097015

[26,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[27,] 0.8694030 0.8694030 0.8694030 0.8694030 0.8694030

[28,] 0.7910448 0.7910448 0.7910448 0.7910448 0.7910448

[29,] 0.8134328 0.8134328 0.8134328 0.8134328 0.8134328

[30,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[31,] 0.8358209 0.8358209 0.8358209 0.8358209 0.8358209

[32,] 0.7723881 0.7723881 0.7723881 0.7723881 0.7723881

[33,] 0.8097015 0.8097015 0.8097015 0.8134328 0.8097015

[34,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[35,] 0.8694030 0.8694030 0.8694030 0.8694030 0.8694030

[36,] 0.7910448 0.7910448 0.7910448 0.7910448 0.7910448

[37,] 0.8134328 0.8134328 0.8134328 0.8134328 0.8134328

[38,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[39,] 0.8358209 0.8358209 0.8358209 0.8358209 0.8358209

[40,] 0.7723881 0.7723881 0.7723881 0.7723881 0.7723881

[41,] 0.8097015 0.8097015 0.8097015 0.8134328 0.8097015

[42,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[43,] 0.8694030 0.8694030 0.8694030 0.8694030 0.8694030

[44,] 0.7910448 0.7910448 0.7910448 0.7910448 0.7910448

[45,] 0.8134328 0.8134328 0.8134328 0.8134328 0.8134328

[46,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

[47,] 0.8358209 0.8358209 0.8358209 0.8358209 0.8358209

[48,] 0.7723881 0.7723881 0.7723881 0.7723881 0.7723881

[49,] 0.8097015 0.8097015 0.8097015 0.8134328 0.8097015

[50,] 0.8283582 0.8283582 0.8283582 0.8283582 0.8283582

Just take one training model to get list of variable importance:

arImp(fit)

Overall

AgeClass 10.205949

FamilyID 111.482955

Fare 57.808843

FareDiff 30.928204

Parch 10.248593

Pclass 75.589263

Sex 85.510496

SibSp 9.213854

Embarked 0.000000

Based on 250 training model, we get mean percentage and standard deviation:

Correction\_mean : 0.8171642

Standard Deviation: 0.032043962756

To get confusion matrix of model, take one model as example:

confusion\_matrix\_df

Survived(Test) Not Survived(Test)

Survived(Train) 0.2574627 0.04850746

Not Survived(Train) 0.1455224 0.54850746

Scripts:

library("rpart")

library("rattle")

library("rpart.plot")

library("RColorBrewer")

library("mice")

library("VIM")

library(caret)

##recursive partitioning and regression trees

input\_dataset <- "C:/Users/jennyc.wang/Downloads/R Task/titanic.csv"

predict\_dataset <- "C:/Users/jennyc.wang/Downloads/R Task/predict.csv"

input\_df <- read.csv(input\_dataset , stringsAsFactors = F )

predict\_df <- read.csv(predict\_dataset)

#combine them

predict\_df$Survived <- NA

predict\_df <- predict\_df[ , names(input\_df)]

df\_cmb <- rbind( input\_df , predict\_df )

###--------------input missing value------------------

#train.mis <- subset( train , select = -c(Name , Sex , Embarked ))

#md.pattern( train.mis )

#mice\_plot <- aggr( train.mis , col=c( "navyblue", "yellow") ,

# numbers = TRUE , sortVars = TRUE ,

# labels = names(train.mis) , cex.axis = .7,

# gap = 3 , ylab=c( "Missing data" , "Pattern" ))

#imputed\_data <- mice( train.mis , m = 5 , maxit = 50 , method = 'pmm' , seed = 500 )

#complete\_data <- complete( imputed\_data , 2 )

#---------------------------------------------------------

##Name split

df\_cmb$Title <- sapply( df\_cmb$Name , FUN = function(x) {strsplit( df\_cmb$Name , '[,[:space:]]')[[1]][3]} )

df\_cmb$FamilyName <- sapply( df\_cmb$Name , FUN = function(x) { strsplit( x, split = '[,.]')[[1]][1]})

df\_cmb$FamilySize <- df\_cmb$SibSp + df\_cmb$Parch + 1

df\_cmb$FamilyID <- paste(as.character(df\_cmb$FamilySize) , df\_cmb$FamilyName , sep = "")

famIds <- data.frame( table( df\_cmb$FamilyID ))

df\_cmb$FamilyID[ df\_cmb$FamilySize <= 2 ] <- 'Small'

df\_cmb$FamilyID <- factor( df\_cmb$FamilyID )

#Fare to idenitfy location

fareIds <- unique( df\_cmb[ , c( "Fare", "Pclass" )])

fareIds\_mean <- aggregate( Fare~Pclass , fareIds , mean )

names(fareIds\_mean) <- c( "Pclass", "meanFare")

df\_mergeedMeanFare <- merge( df\_cmb , fareIds\_mean, by = "Pclass")

df\_mergeedMeanFare$FareDiff <- df\_mergeedMeanFare$Fare - df\_mergeedMeanFare$meanFare

#clean Embarked

df\_mergeedMeanFare$Embarked[ which( is.na( df\_mergeedMeanFare$Embarked ) ) ] <- ""

Actual\_datasets <- df\_mergeedMeanFare[ -which(is.na(df\_mergeedMeanFare$Survived)), ]

Predict\_datasets <- df\_mergeedMeanFare[ which(is.na(df\_mergeedMeanFare$Survived)), ]

sampling\_count <- 50

impute\_data\_counts <- 5

correct\_percent\_list <- matrix( , nrow = sampling\_count , ncol = impute\_data\_counts )

var\_importance\_matrix <- matrix( , nrow = sampling\_count , ncol = impute\_data\_counts )

confusion\_matrix <- matrix( , nrow = 2 , ncol = 2 )

confusion\_Matrix\_matrix <- matrix( , nrow = sampling\_count , ncol = impute\_data\_counts )

###############################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

for ( i in 1 : sampling\_count ){

train <- Actual\_datasets[ sample(nrow(Actual\_datasets) , as.integer( nrow(Actual\_datasets) \* 0.7 ) ), ]

test <- Actual\_datasets [ - which( Actual\_datasets$PassengerId %in% train$PassengerId ) , ]

#------------------------------------------------

train.mis <- subset( train , select = -c(Name , Sex , Embarked ))

imputed\_data <- mice( train.mis , m = impute\_data\_counts , maxit = 50 , method = 'pmm' , seed = 500 )

for( j in 1 : impute\_data\_counts ){

complete\_data <- complete( imputed\_data , j )

train\_v2 <- merge( complete\_data , train[ , c( "PassengerId", "Name", "Sex" , "Embarked" )] , by = "PassengerId" )

#Generate age class

train\_v2$Age <- as.numeric( train\_v2$Age )

train\_v2[ which( train\_v2$Age < 10 ), 'AgeClass'] <- "Kid"

train\_v2[ which( train\_v2$Age >= 10 & train\_v2$Age < 18 ), 'AgeClass'] <- "Youth"

train\_v2[ which( train\_v2$Age >= 18 & train\_v2$Age < 65 ), 'AgeClass'] <- "Adult"

train\_v2[ which( train\_v2$Age >=65 ), 'AgeClass'] <- "Senior"

#test

test$Age <- as.numeric( test$Age )

test[ which( test$Age < 10 ), 'AgeClass'] <- "Kid"

test[ which( test$Age >= 10 & test$Age < 18 ), 'AgeClass'] <- "Youth"

test[ which( test$Age >= 18 & test$Age < 65 ), 'AgeClass'] <- "Adult"

test[ which( test$Age >=65 ), 'AgeClass'] <- "Senior"

fit <- rpart( Survived ~ Pclass + Sex + AgeClass + SibSp + Parch + Fare + Embarked + FamilyID + FareDiff,

data = train\_v2 , method = "class" )

var\_importance\_matrix[i,j] <- varImp(fit)

fancyRpartPlot(fit)

Prediction <- predict( fit, test , type = "class" )

submit <- data.frame( PassengerId = test$PassengerId , Survived = Prediction )

##

df\_merge <- merge( test , submit , by = 'PassengerId' )

##confusion matrix

confusion\_matrix[1,1] <- length( which( df\_merge$Survived.x == 1 & df\_merge$Survived.y == 1 ) )/ nrow( df\_merge )

confusion\_matrix[1,2] <- length( which( df\_merge$Survived.x == 0 & df\_merge$Survived.y == 1 ) )/ nrow( df\_merge )

confusion\_matrix[2,1] <- length( which( df\_merge$Survived.x == 1 & df\_merge$Survived.y == 0 ) )/ nrow( df\_merge )

confusion\_matrix[2,2] <- length( which( df\_merge$Survived.x == 0 & df\_merge$Survived.y == 0 ) )/ nrow( df\_merge )

confusion\_matrix\_df <- as.data.frame(confusion\_matrix)

names(confusion\_matrix\_df) <- c( "Survived(Test)", "Not Survived(Test)" )

rownames(confusion\_matrix\_df) <- c( "Survived(Train)" , "Not Survived(Train)" )

confusion\_Matrix\_matrix[ i , j ] <- confusion\_matrix\_df

correct\_percentage <- length( which( df\_merge$Survived.y == df\_merge$Survived.x ) ) / nrow( df\_merge )

correct\_percent\_list[i,j] <- correct\_percentage

}

}

#assign 0 to Na

correct\_percent\_list[ is.na(correct\_percent\_list) ] <- 0

#convert matrix to unlist

unlst\_correction <- unlist( as.list( as.data.frame( correct\_percent\_list )))

#mean

correct\_mean <- mean( setdiff( unlst\_correction , 0 ) )

std\_devi <- sd( setdiff( unlst\_correction , 0 ) , na.rm = F )

###################################################@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

actual.mis <- subset( Actual\_datasets , select = -c(Name , Sex , Embarked ))

imputed\_actual\_data <- mice( actual.mis , m = impute\_data\_counts , maxit = 50 , method = 'pmm' , seed = 500 )

submit\_list <- list()

var\_importance <- list()

for( j in 1 : impute\_data\_counts ){

complete\_actual\_data <- complete( imputed\_actual\_data , j )

actual\_v2 <- merge( complete\_actual\_data , Actual\_datasets[ , c( "PassengerId", "Name", "Sex" , "Embarked" )] , by = "PassengerId" )

#Generate age class

actual\_v2$Age <- as.numeric( actual\_v2$Age )

actual\_v2[ which( actual\_v2$Age < 10 ), 'AgeClass'] <- "Kid"

actual\_v2[ which( actual\_v2$Age >= 10 & actual\_v2$Age < 18 ), 'AgeClass'] <- "Youth"

actual\_v2[ which( actual\_v2$Age >= 18 & actual\_v2$Age < 65 ), 'AgeClass'] <- "Adult"

actual\_v2[ which( actual\_v2$Age >=65 ), 'AgeClass'] <- "Senior"

#test

Predict\_datasets$Age <- as.numeric( Predict\_datasets$Age )

Predict\_datasets[ which( Predict\_datasets$Age < 10 ), 'AgeClass'] <- "Kid"

Predict\_datasets[ which( Predict\_datasets$Age >= 10 & Predict\_datasets$Age < 18 ), 'AgeClass'] <- "Youth"

Predict\_datasets[ which( Predict\_datasets$Age >= 18 & Predict\_datasets$Age < 65 ), 'AgeClass'] <- "Adult"

Predict\_datasets[ which( Predict\_datasets$Age >=65 ), 'AgeClass'] <- "Senior"

fit\_real <- rpart( Survived ~ Pclass + Sex + AgeClass + SibSp + Parch + Fare + Embarked + FamilyID + FareDiff,

data = actual\_v2 , method = "class" )

var\_importance[[j]] <- varImp(fit\_real)

fancyRpartPlot(fit\_real)

Prediction\_real <- predict( fit\_real , Predict\_datasets , type = "class" )

submit\_real <- data.frame( PassengerId = Predict\_datasets$PassengerId , Survived = Prediction\_real )

submit\_list[[j]] <- submit\_real

}

#compare 5 outcomes

for( x in 1 : impute\_data\_counts ){ names(submit\_list[[x]])[2] <- paste0( "Survived-", x ) }

df\_combined <- as.data.frame ( Reduce( function(x,y) { merge(x , y, by = "PassengerId" ) }, submit\_list ) )

df\_combined <- as.data.frame( lapply( df\_combined , function(x) as.numeric(as.character(x)) ) )

df\_combined$vari <- rowSums( df\_combined[ , grep( 'survived' , names(df\_combined) , ignore.case = T )])

table(df\_combined$vari)

#Final(anyone is ok)

Final <- merge( predict\_df , df\_combined[ , c("PassengerId","Survived.1")] , by = "PassengerId" )

Final$Survived <- Final$Survived.1

Final <- Final[ , names(predict\_df)]

write.csv(Final , file = "C:/Users/jennyc.wang/Downloads/R Task/predictv2.0.csv" , row.names = F )