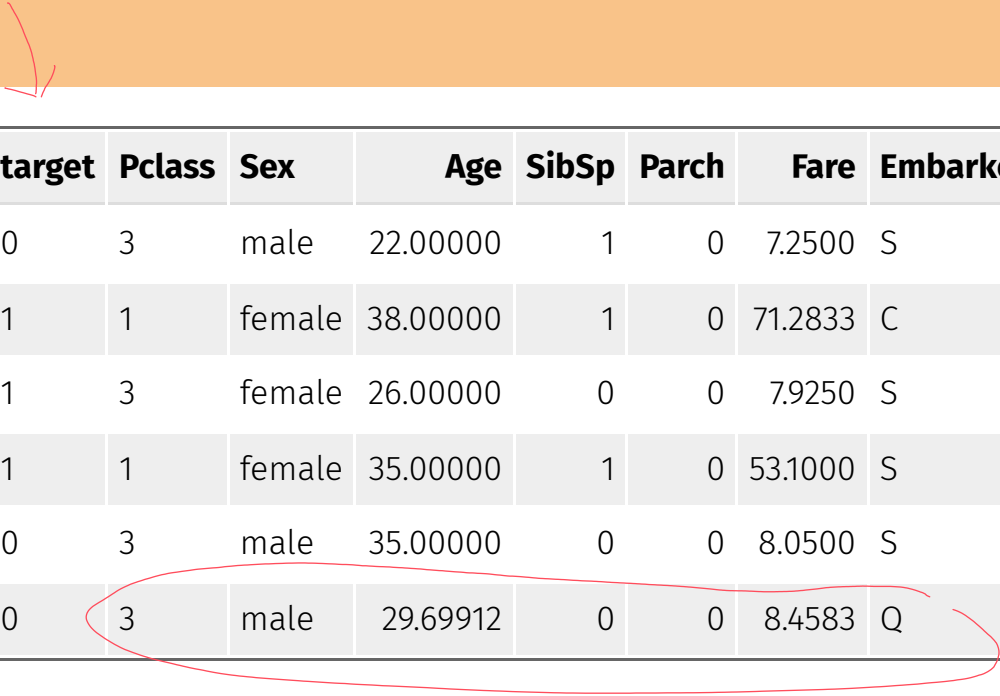


# Predictive Modeling - Part 1

Son Nguyen

# Data



target	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.00000	1	0	7.2500	S
1	1	female	38.00000	1	0	71.2833	C
1	3	female	26.00000	0	0	7.9250	S
1	1	female	35.00000	1	0	53.1000	S
0	3	male	35.00000	0	0	8.0500	S
0	3	male	29.69912	0	0	8.4583	Q

- Passengers in the Titanic
- Target = 1 means the passenger was survived
- Target = 0 means the passenger was not survived

# Prediction Problem

target	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.00000	1	0	7.2500	S
1	1	female	38.00000	1	0	71.2833	C
1	3	female	26.00000	0	0	7.9250	S
1	1	female	35.00000	1	0	53.1000	S
0	3	male	35.00000	0	0	8.0500	S
0	3	male	29.69912	0	0	8.4583	Q

- We want to predict the `target` given the information of other variables.

# Import and Clean the data

```
# Read in the data  
library(tidyverse)  
df = read_csv("https://bryantstats.github.io/math421/data/titanic.csv")
```

# Set the Target Variable

- It's a common practice that the target variable named `target`

```
# Take out some columns
df ← df %>% select(-PassengerId, -Ticket, -Name, -Cabin)

# Set the target variable
df ← df %>% rename(target=Survived)
```

# Correct Variables' Types

- Make sure all categorical variables are factors.

```
# Correct variables' types
df <- df %>%
  mutate(target = as.factor(target),
         Pclass = as.factor(Pclass),
         Embarked = as.factor(Embarked),
         Sex = as.factor(Sex)
  )
```

# Handle Missing Values

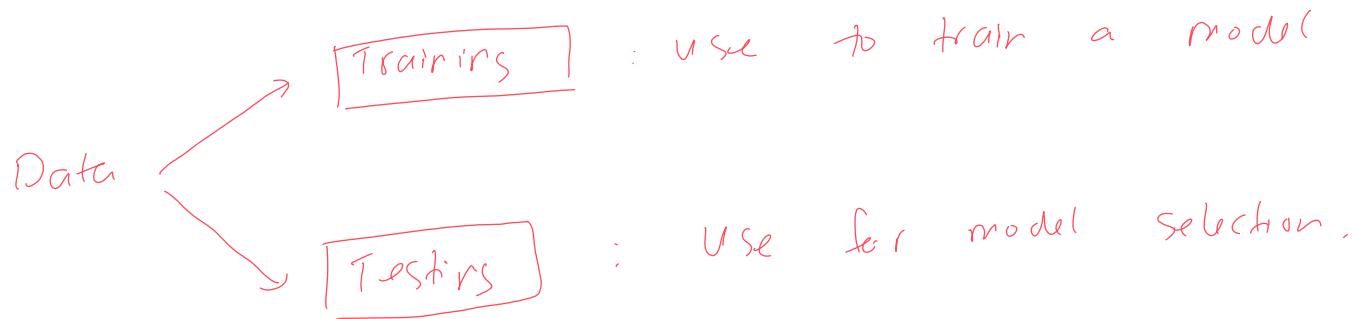
- Make sure there are no missing values

```
# Replace NA of Age by its mean  
mean_age ← mean(df$Age, na.rm=TRUE)  
df$Age ← replace_na(df$Age, mean_age)  
  
# Drop all rows that has an NA  
df = drop_na(df)
```

# Split the data to training and testing

- Make sure to set seed to that the results are reproducible.

```
library(caret)
set.seed(2020)
splitIndex <- createDataPartition(df$target, p = .70,
                                   list = FALSE)
df_train <- df[ splitIndex,]
df_test <- df[-splitIndex,]
```



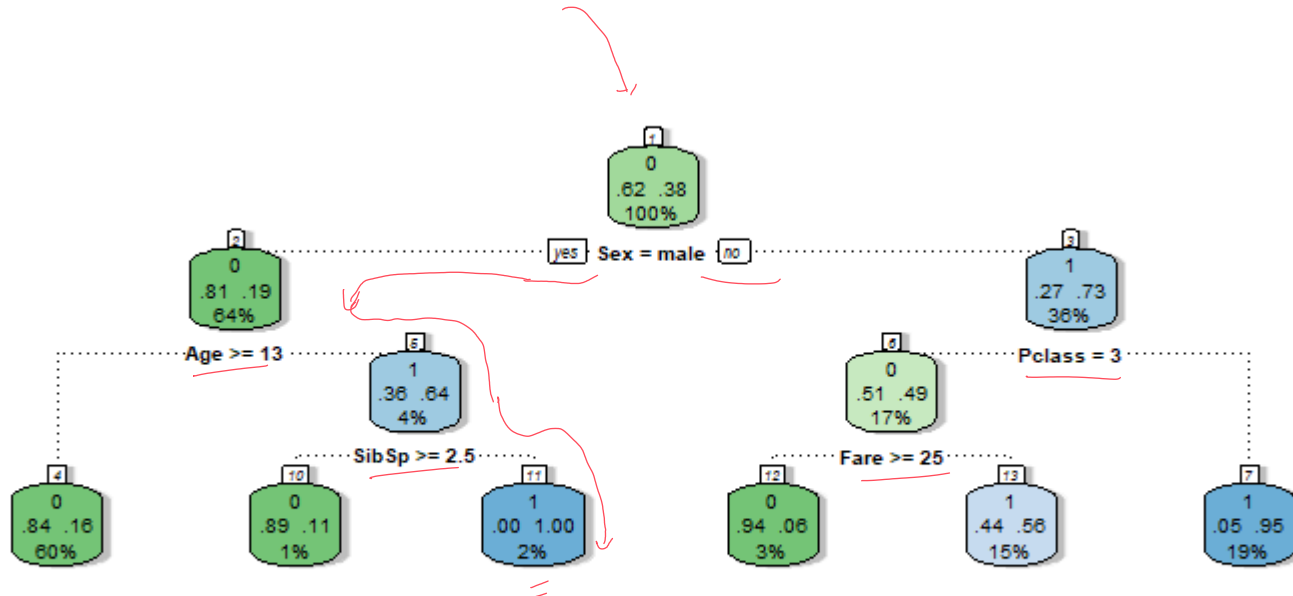




# Plot the tree

Decision Tree

```
library(rattle)
fancyRpartPlot(tree_model)
```



Rattle 2022-Oct-16 19:37:02 sonou

# Variable importances

```
tree_model$variable.importance
```

```
##           Sex           Fare          Pclass           Age           SibSp           Parch           Embarked
## 83.761904 32.854400 28.807881 19.954127 19.287469 14.431109 7.625484
```

# Variable importances

```
barplot(tree_model$variable.importance)
```



# Evaluate the tree

```
#predict on testing data  
pred ← predict(tree_model, df_test, type = "class")  
  
#Evaluate the predictions  
cm ← confusionMatrix(data = pred, reference = df_test$target, positive = "1")  
cm$overall[1]
```

```
## Accuracy  
## 0.8383459
```

# Evaluate the tree

	<b>Metric</b>
Accuracy	0.8383459
Kappa	0.6483429
AccuracyLower	0.7884950
AccuracyUpper	0.8804722
AccuracyNull	0.6165414
AccuracyPValue	0.0000000
McnemarPValue	0.0327626

# Evaluate the tree

	<b>Metric</b>
Sensitivity	0.7156863
Specificity	0.9146341
Pos Pred Value	0.8390805
Neg Pred Value	0.8379888
Precision	0.8390805
Recall	0.7156863
F1	0.7724868
Prevalence	0.3834586
Detection Rate	0.2744361
Detection Prevalence	0.3270677
Balanced Accuracy	0.8151602

# Random Forest

- Random Forest is a collection of decision trees
- Random Forest predict by the majority vote between the trees
- For example: if 51 trees in a forest of 100 trees predict passenger A survived, then the forest also predict passenger A survived
- Trees are trained only a subset of the original data
- Only random of few variables are considered at each split



# Random Forest

```
library(randomForest)
forest_model = randomForest(target ~ ., data=df_train, ntree = 500)
pred <- predict(forest_model, df_test, type = "class")

cm <- confusionMatrix(data = pred, reference = df_test$target, positive = "1")

cm$overall[1]
```

```
## Accuracy
```

```
## 0.8270677
```

# Variable importances

```
importance(forest_model)
```

##	MeanDecreaseGini
## Pclass	23.497310
## Sex	69.014066
## Age	38.661926
## SibSp	10.796182
## Parch	9.727980
## Fare	49.767591
## Embarked	7.816245

# Evaluate the Forest

	<b>Metric</b>
Accuracy	0.8270677
Kappa	0.6187212
AccuracyLower	0.7761558
AccuracyUpper	0.8705225
AccuracyNull	0.6165414
AccuracyPValue	0.0000000
McnemarPValue	0.0019596

# Evaluate the Forest

	<b>Metric</b>
Sensitivity	0.6666667
Specificity	0.9268293
Pos Pred Value	0.8500000
Neg Pred Value	0.8172043
Precision	0.8500000
Recall	0.6666667
F1	0.7472527
Prevalence	0.3834586
Detection Rate	0.2556391
Detection Prevalence	0.3007519
Balanced Accuracy	0.7967480