T00710118_Shohada Sharmin_Final Term Project

2023-04-11

```
pkg_list <- c("tidyverse","MASS", "ISLR","ISLR2", "dplyr", "caret","ModelMetrics",
              "ggplot2", "corrplot" , "glmnet", "pwr")

# Install packages if needed
for (pkg in pkg_list)
{
  # Try loading the library.
  if ( ! library(pkg, logical.return=TRUE, character.only=TRUE) )
    {
         # If the library cannot be loaded, install it; then load.
        install.packages(pkg)
        library(pkg, character.only=TRUE)
  }
}
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
## Warning: package 'tidyverse' is in use and will not be installed
```

```
## Warning: package 'pwr' was built under R version 4.2.3
```

```
my_train_data <- read.csv("C:/Users/sharm/Documents/train_titan.csv", stringsAsFactors = FALSE)
my_test_data <- read.csv("C:/Users/sharm/Documents/test_titan.csv", stringsAsFactors = FALSE)


# Define custom function to check for missing values
my_nas <- function(x) {
  sum(is.na(x))
}

# Check for missing values in the training data before preprocess
sapply(my_train_data, my_nas)
```

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

```
# Check for missing values in the test data before preprocess
sapply(my_test_data, my_nas)
```

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

```
library(ggplot2)
# Visualize the survival rate by Passenger Class and Sex
ggplot(my_train_data, aes(x = factor(Pclass), fill = factor(Survived))) +
geom_bar(position = "fill", color = "black") +
facet_wrap(~Sex, nrow = 1) +
scale_fill_manual(values = c("#009E73", "#F0E442"), name = "Survived") +
labs(title = "Survival by Ticket Class and Gender", x = "Ticket Class")
```


```
# Summary statistics for numerical variables
summary(my_train_data[, c("Fare", "Parch", "SibSp", "Age")])
```

```
##       Fare             Parch            SibSp            Age
##  Min.   :  0.00   Min.   :0.0000   Min.   :0.000   Min.   : 0.42
##  1st Qu.:  7.91   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:20.12
##  Median : 14.45   Median :0.0000   Median :0.000   Median :28.00
##  Mean   : 32.20   Mean   :0.3816   Mean   :0.523   Mean   :29.70
```

```
##  3rd Qu.: 31.00   3rd Qu.:0.0000   3rd Qu.:1.000   3rd Qu.:38.00
##  Max.   :512.33   Max.   :6.0000   Max.   :8.000   Max.   :80.00
##                                                    NA's   :177
```

```
# Frequency table for categorical variables
table(my_train_data$Sex)
```

```
##
## female   male
##    314    577
```

```
table(my_train_data$Embarked)
```

```
##
##       C   Q   S
##   2 168  77 644
```

```
# Correlation matrix
cor(my_train_data[, c("Fare", "Parch", "SibSp", "Age")])
```

```
##            Fare     Parch     SibSp Age
## Fare  1.0000000 0.2162249 0.1596510  NA
## Parch 0.2162249 1.0000000 0.4148377  NA
## SibSp 0.1596510 0.4148377 1.0000000  NA
## Age          NA        NA        NA   1
```

```
# Remove unnecessary columns
my_train_data <- select(my_train_data, -c(Cabin, Ticket, Name, PassengerId))
my_test_data <- select(my_test_data, -c(Cabin, Ticket, Name, PassengerId))

# Fill in missing values for Age and Fare columns with the median value
my_train_data$Age[which(is.na(my_train_data$Age))] <- median(my_train_data$Age[!is.na(my_train_data$Age
my_test_data$Age[which(is.na(my_test_data$Age))] <- median(my_test_data$Age[!is.na(my_test_data$Age)])
my_test_data$Fare[which(is.na(my_test_data$Fare))] <- median(my_test_data$Fare[!is.na(my_test_data$Fare

# Convert all non-numeric variables to factor variables in both train and test datasets
my_train_data <- my_train_data %>% mutate_if(.predicate = function(x) !is.numeric(x), .funs = factor)
my_test_data <- my_test_data %>% mutate_if(.predicate = function(x) !is.numeric(x), .funs = factor)


# Convert Survived column in my_train_data to factor (0/1)
my_train_data$Survived <- as.factor(my_train_data$Survived)

# Impute missing values in the training and test data
preproc <- preProcess(my_train_data %>% select(-Survived), method = c("center", "scale", "knnImpute"))
train_data_proc <- predict(preproc, my_train_data %>% select(-Survived))
train_data_proc$Survived <- my_train_data$Survived
```

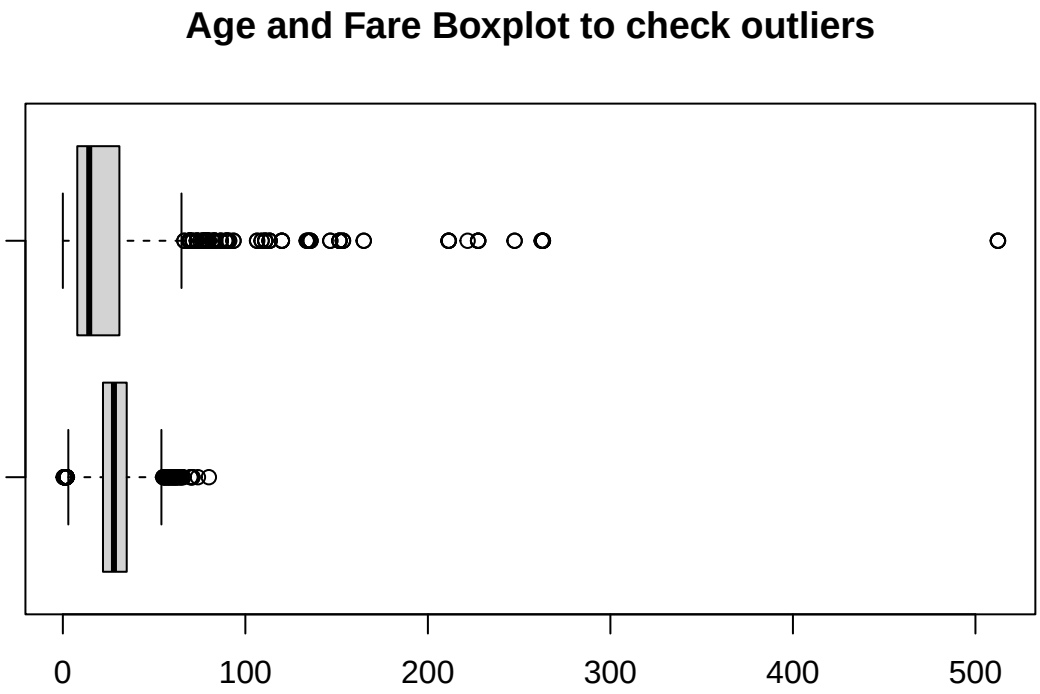
```
# Check for missing values after process missing value
colSums(is.na(my_train_data))
```

```
## Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##        0        0        0        0        0        0        0        0
```

```
colSums(is.na(my_test_data))
```

```
##   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##        0        0        0        0        0        0        0
```

```
# Check for outliers before process for numerical variable Age and Fare
boxplot(my_train_data$Age, my_train_data$Fare, horizontal = TRUE, main = "Age and Fare Boxplot to check
```




```
# Remove outliers
my_train_data <- my_train_data[my_train_data$Fare < quantile(my_train_data$Fare, 0.99) & my_train_data$A


# Set up cross-validation
trctrl <- trainControl(method = "cv", number = 10)

# Train logistic regression model using cross-validation
logit_fit <- train(Survived ~ ., data = train_data_proc, method = "glm", trControl = trctrl)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
# Print results
print(logit_fit)
```

```
## Generalized Linear Model
##
## 891 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 802, 801, 802, 801, 802, 802, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7978725  0.5676445
```

```r
# Train a random forest model using k-fold cross-validation.
rf_fit <- train(Survived ~ ., data = train_data_proc, method = "rf", trControl = trctrl)

# Print results
print(rf_fit)
```

```
## Random Forest
##
## 891 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 801, 802, 803, 802, 801, 802, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.8282224  0.6189999
##   5     0.8316434  0.6375825
##   9     0.8216187  0.6191792
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

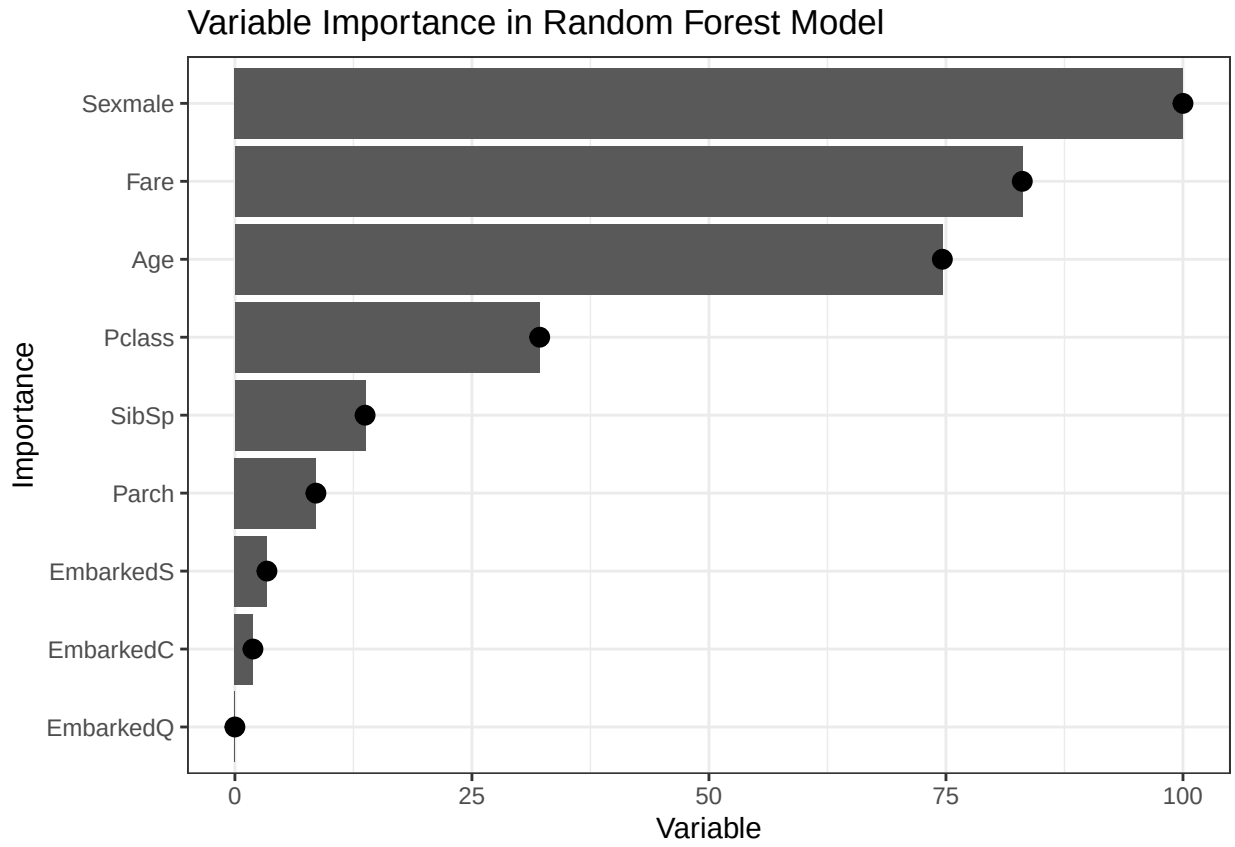
```r
library(lattice)

# Compare model performance using resamples
compare_models <- resamples(list(Logistic_Regression = logit_fit, Random_Forest = rf_fit))


# Extract variable importance from random forest model
var_imp <- varImp(rf_fit)

# Plot variable importance
```

```
ggplot(var_imp, aes(x = Importance, y = Reordered_names)) +
  geom_point(size = 3) +
  labs(title = "Variable Importance in Random Forest Model", x = "Importance", y = "Variable") +
  theme_bw()
```

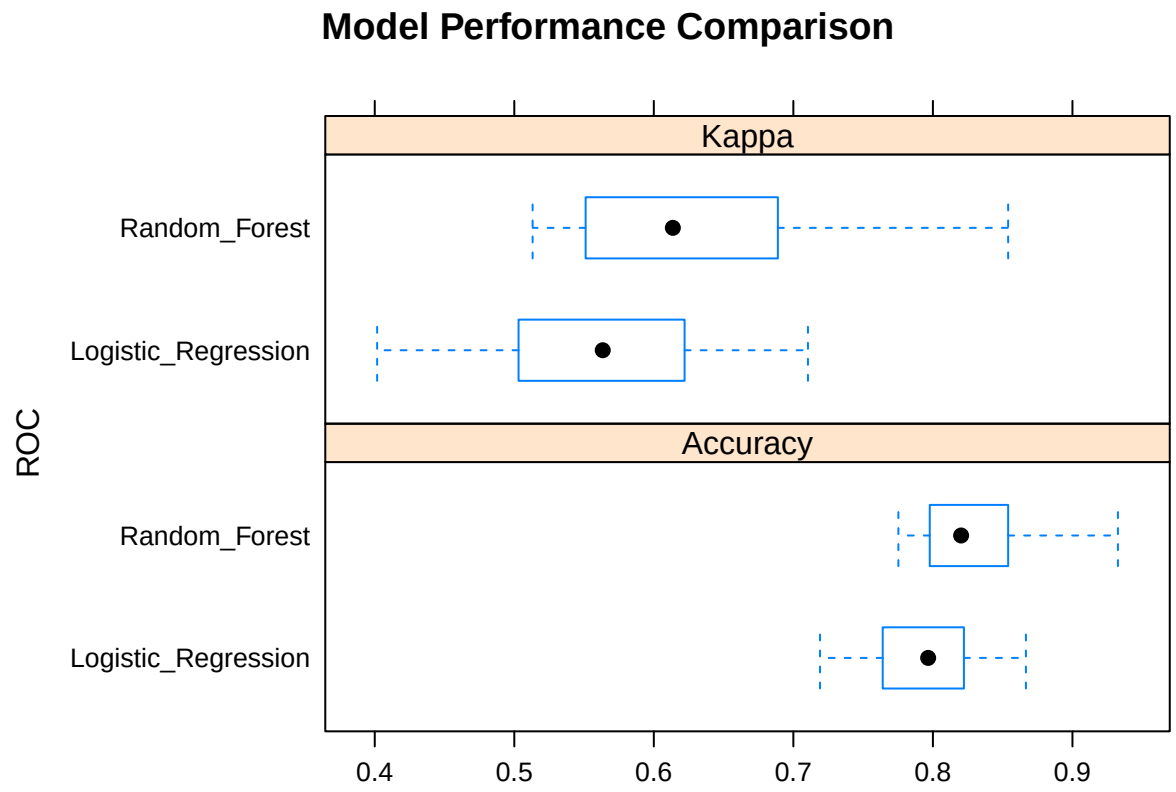



```
# Summarize results
summary(compare_models)
```

```
##
## Call:
## summary.resamples(object = compare_models)
##
## Models: Logistic_Regression, Random_Forest
## Number of resamples: 10
##
## Accuracy
##                          Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## Logistic_Regression 0.7191011 0.7696629 0.7966037 0.7978725 0.8189139 0.8666667
## Random_Forest       0.7752809 0.8028601 0.8202247 0.8316434 0.8511236 0.9325843
##                     NA's
## Logistic_Regression    0
## Random_Forest          0
##
## Kappa
##                          Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
```

```
## Logistic_Regression 0.4017209 0.5158068 0.5633882 0.5676445 0.6148280 0.7104558
## Random_Forest       0.5131291 0.5636776 0.6136346 0.6375825 0.6804786 0.8539387
##                     NA's
## Logistic_Regression    0
## Random_Forest          0
```

```
# Plot results
bwplot(compare_models, layout = c(1,2), ylab = "ROC",
       main = "Model Performance Comparison",
       auto.key = list(space = "right", columns = 2))
```




```
# Make predictions of survival on test data set
test_data_proc <- predict(preproc, my_test_data)
predictions_survive <- predict(rf_fit, test_data_proc)

# Save predictions of survival to file for test data
id_range <- seq(from = 892, to = 1309)
output <- data.frame(PassengerId = id_range, Survived = predictions_survive)
write.csv(output, file = "predictions_survive.csv", row.names = FALSE)
```