Classification Case

Overview

The aim of this work is to predict the gender class and evaluate the algorithms which are used. First the libraries we are going to use are installed to the environment.

```
library(ggplot2)
library(caret)
library(gridExtra)
library(randomForest)
```

The dataset is loaded and summary about the dataset is shown.

```
data <- read.csv("dataset.CSV", header = T)
str(data)
```

```
## 'data.frame':    10000 obs. of  3 variables:
##  $ Gender: Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Height: num  73.8 68.8 74.1 71.7 69.9 ...
##  $ Weight: num  242 162 213 220 206 ...
```

```
prop.table(table(data$Gender))
```

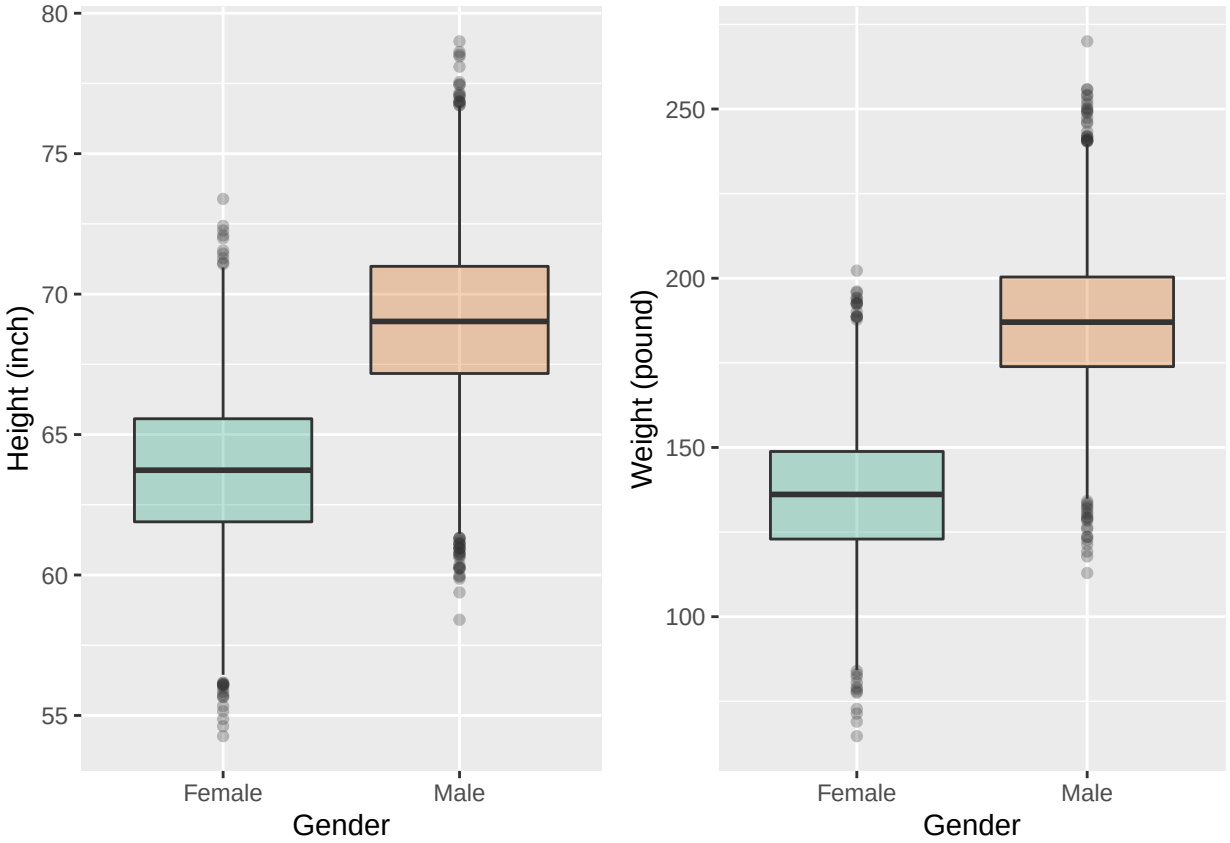
```
##
## Female   Male
##    0.5    0.5
```

Stastics of the dataset's features are shown on the boxplots below.

```
plot1 <- ggplot(data, aes(x=Gender, y=Height, fill=Gender)) +
  ylab("Height (inch)") +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none") +
  scale_fill_brewer(palette="Dark2")

plot2 <- ggplot(data, aes(x=Gender, y=Weight, fill=Gender)) +
  ylab("Weight (pound)") +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none") +
  scale_fill_brewer(palette="Dark2")

grid.arrange(plot1, plot2, ncol=2)
```



The dataset is split into train and test sets by half and half.

```
intrain <- createDataPartition(y=data$Gender,p=0.5,list=FALSE)
train <- data[intrain,]
test <- data[-intrain,]
```

Proportions of train and test sets' gender are found to be same.

```
prop.table(table(train$Gender))
```

```
##
## Female   Male
##    0.5    0.5
```

```
prop.table(table(test$Gender))
```

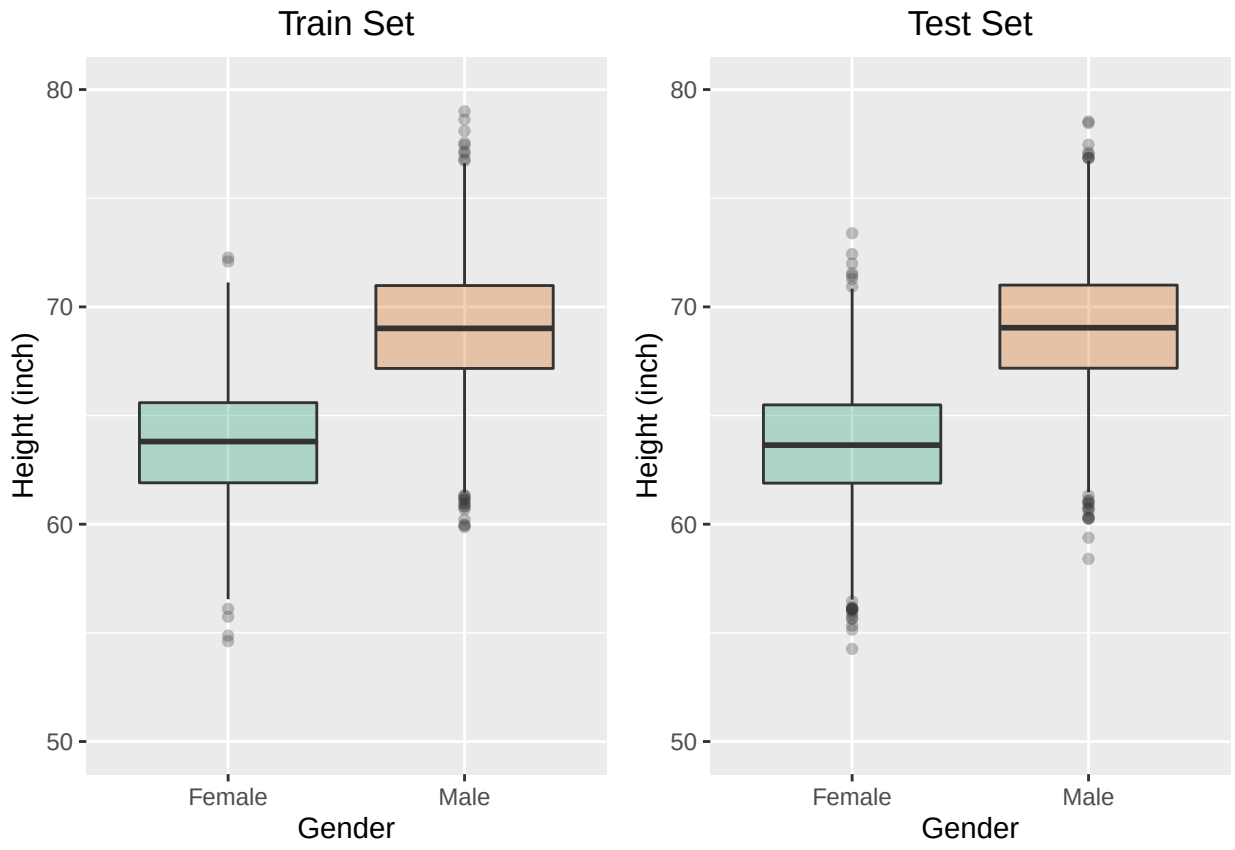
```
##
## Female   Male
##    0.5    0.5
```

Boxplots below also demonstrate that distribution of 'Height' is similar in both sets.

```
plot3 <- ggplot(train, aes(x=Gender, y=Height, fill=Gender)) +
  ggtitle("Train Set") +
  ylab("Height (inch)") +
  ylim(50,80) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5)) +
  scale_fill_brewer(palette="Dark2")

plot4 <- ggplot(test, aes(x=Gender, y=Height, fill=Gender)) +
  ggtitle("Test Set") +
  ylab("Height (inch)") +
  ylim(50,80) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5)) +
  scale_fill_brewer(palette="Dark2")

grid.arrange(plot3, plot4, ncol=2)
```



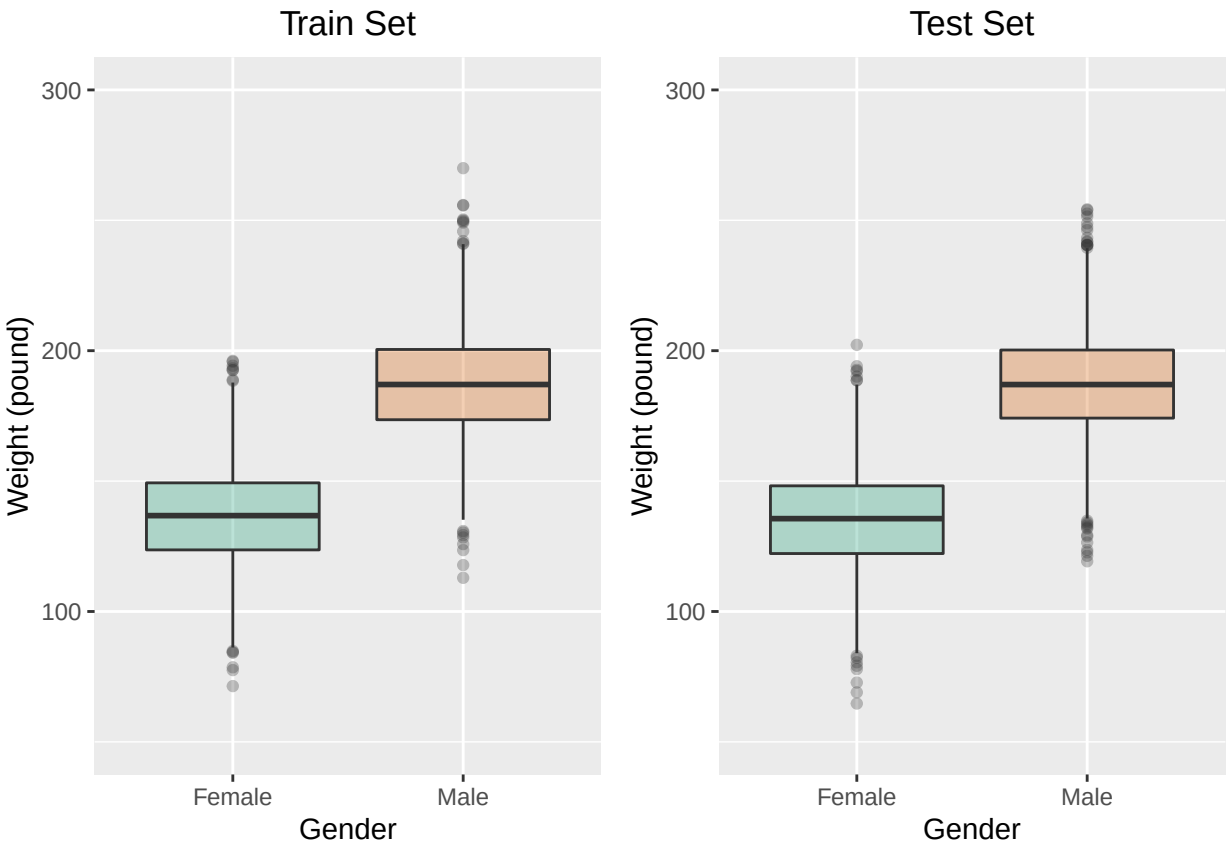
Boxplots below also demonstrate that distribution of 'Weight' is similar in both sets.

```
plot5 <- ggplot(train, aes(x=Gender, y=Weight, fill=Gender)) +
  ggtitle("Train Set") +
  ylab("Weight (pound)") +
  ylim(50,300) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5)) +
```

```
  scale_fill_brewer(palette="Dark2")

plot6 <- ggplot(test, aes(x=Gender, y=Weight, fill=Gender)) +
  ggtitle("Test Set") +
  ylab("Weight (pound)") +
  ylim(50,300) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5)) +
  scale_fill_brewer(palette="Dark2")

grid.arrange(plot5, plot6, ncol=2)
```



Genders for the test dataset using different probabilities are predicted randomly. Predictions are stored in a dataframe.

```
mat <- matrix(nrow = 5000, ncol = 0)
for (p in (1:9)/10) {
  mat <- cbind(mat,sample(c("Male", "Female"), nrow(test), replace = TRUE, prob=c(p, 1-p)))
}

df <- as.data.frame(mat)
```

Accuracy and F1 score of each prediction set is stored in a matrix.

```
accuracy <- matrix(nrow = 0, ncol = 2)
for (i in 1:9) {
  tp <- table(test$Gender,df[,i])[2,2]
  tn <- table(test$Gender,df[,i])[1,1]
  fp <- table(test$Gender,df[,i])[1,2]
  fn <- table(test$Gender,df[,i])[2,1]
  accuracy <- rbind(accuracy, c(i/10,((tp+tn)/(tp+tn+fp+fn))))
}

f1_score <- matrix(nrow = 0, ncol = 1)
for (i in 1:9) {
  tp <- table(test$Gender,df[,i])[2,2]
  tn <- table(test$Gender,df[,i])[1,1]
  fp <- table(test$Gender,df[,i])[1,2]
  fn <- table(test$Gender,df[,i])[2,1]
  precision <- tp/(tp+fp)
  recall <- tp/(tp+fn)
  f1_score <- rbind(f1_score, 2*(precision*recall)/(precision+recall))
}
```

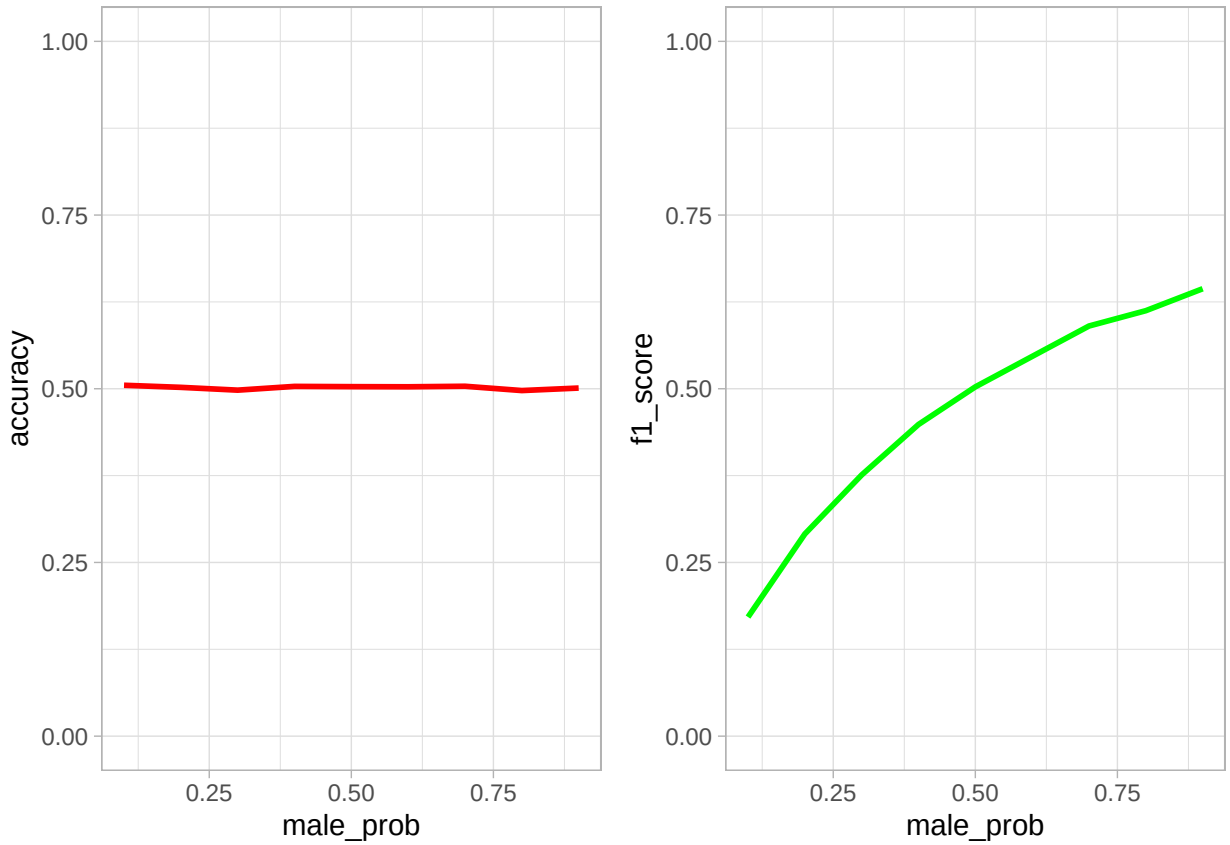
Aacuracy and F1 Score of each randomly predicted set is plotted below. Positive class is "Male". In some cases, true positive cases are more valuable than the true negative cases. Accuracy does not take that into account whereas F1 Score gives higher weight to positive cases. That's why, according to plots, accuracy does not change much throughout the different prediction sets. On the other hand, F1 score increases while probability of being male increases.

```
perf_df <- as.data.frame(cbind(accuracy,f1_score))
perf_df <- setNames(perf_df, c("male_prob","accuracy","f1_score"))

plot7 <- ggplot(perf_df, aes(x = male_prob, y = accuracy)) +
  geom_line(color = "red", size = 1) +
  ylim(0,1) +
  theme_light()

plot8 <- ggplot(perf_df, aes(x = male_prob, y = f1_score)) +
  geom_line(color = "green", size = 1) +
  ylim(0,1) +
  theme_light()

grid.arrange(plot7, plot8, ncol=2)
```



Algorithm Training

First, logistic regression algorithm is used to build a model for predicting gender. Positive class is chosen "Male" as default.

```
model <- glm(Gender ~., family=binomial(link='logit'), data=train)
model
```

```
##
## Call:  glm(formula = Gender ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Coefficients:
## (Intercept)       Height       Weight
##      1.6116      -0.5169       0.2025
##
## Degrees of Freedom: 4999 Total (i.e. Null);  4997 Residual
## Null Deviance:       6931
## Residual Deviance: 2096  AIC: 2102
```

Prediction scores are done based on the model and each observation in the train set is asssigned either male or female according to their scores.

```
pred_train <- predict.glm(model, newdata = train, type = "response")
pred_train <- as.data.frame(ifelse(pred_train >= 0.5, "Male", "Female"))
pred_train$Gender <- pred_train[,1]
pred_train[,1] <- NULL
```

F1 score of train set when we use logistic regression, is then calculated. Other performance metrics are also calculated.

```
cm <- table(train$Gender,pred_train$Gender)
tp <- cm[2,2]
tn <- cm[1,1]
fp <- cm[1,2]
fn <- cm[2,1]
precision <- tp/(tp+fp)
recall <- tp/(tp+fn)
f1_score_log <- 2*(precision*recall)/(precision+recall)

print(cm)
```

```
##
##          Female Male
##   Female   2291  209
##   Male      198 2302
```

```
print(precision)
```

```
## [1] 0.9167662
```

```
print(recall)
```

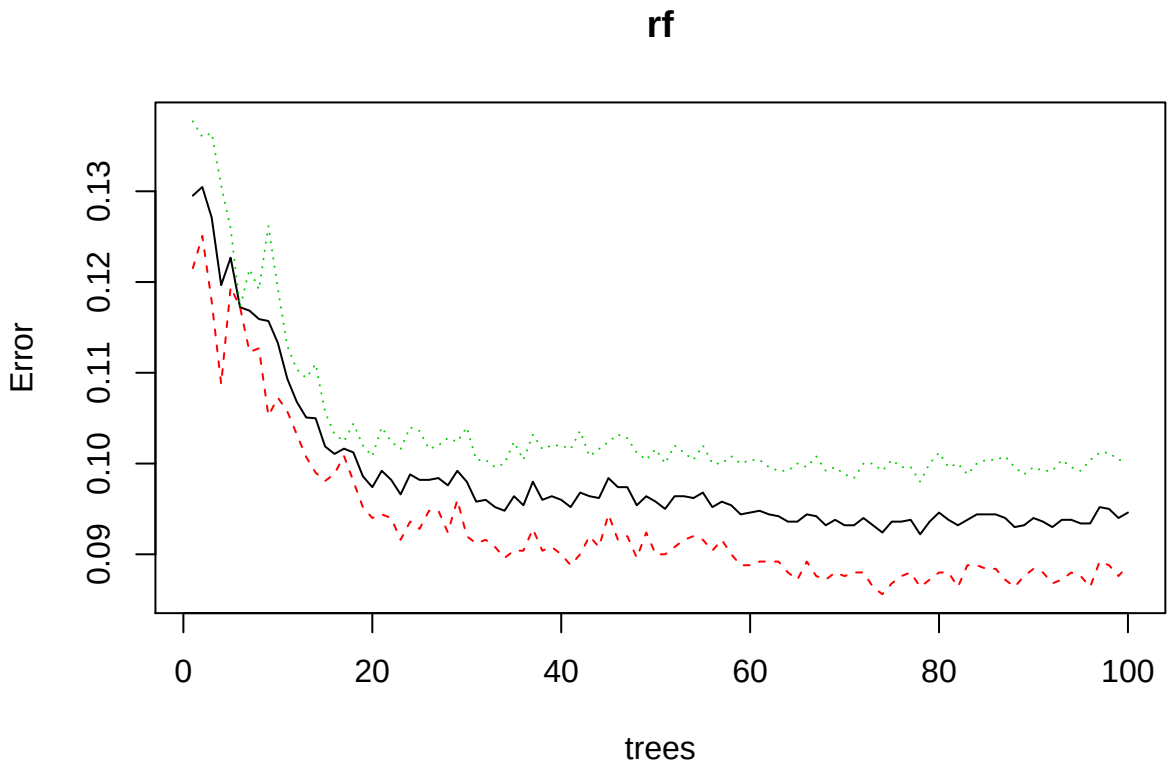
```
## [1] 0.9208
```

```
print(f1_score_log)
```

```
## [1] 0.9187787
```

Second, Random Forest Algorithm is used. Again positive class is chosen as "Male".

```
rf <- randomForest(Gender ~ ., ntree = 100, data = train)
plot(rf)
```



Error significantly decreases until 20 trees. That's why tree size is chosen as 30.

```
rf <- randomForest(Gender ~ ., ntree = 30, data = train)
```

Prediction scores are done based on the model and each observation in the train set is asssigned either male or female according to their scores.

```
pred_rf <- predict(rf, train, type = "prob")
pred_rf <- pred_rf[,2]
pred_rf <- as.data.frame(ifelse(pred_rf >= 0.5, "Male", "Female"))
pred_rf$Gender <- pred_rf[,1]
pred_rf[,1] <- NULL
```

F1 score of train set when we use Random Forest, is then calculated. Other performance metrics are also calculated.

```
cm <- table(train$Gender,pred_rf$Gender)
tp <- cm[2,2]
tn <- cm[1,1]
fp <- cm[1,2]
fn <- cm[2,1]
precision <- tp/(tp+fp)
recall <- tp/(tp+fn)
accuracy <- (tp+tn)/(tp+tn+fp+fn)
f1_score_rf <- 2*(precision*recall)/(precision+recall)
```

```
print(cm)
```

```
##
##          Female Male
##   Female   2496    4
##   Male        6 2494
```

```
print(precision)
```

```
## [1] 0.9983987
```

```
print(recall)
```

```
## [1] 0.9976
```

```
print(accuracy)
```

```
## [1] 0.998
```

```
print(f1_score_rf)
```

```
## [1] 0.9979992
```

Comparing F1 score of each model one another, Random Forest performs better.

```
ifelse(f1_score_rf >= f1_score_log,
       print("Go for Random Forest!"),
       print("Go for Logistic Regression!"))
```

```
## [1] "Go for Random Forest!"
```

```
## [1] "Go for Random Forest!"
```

Model Performance

Model that is built using Random Forest Classification Algorithm is used to predict the observations' gender.

```
pred_rf_test <- predict(rf, test, type = "prob")
pred_rf_test <- pred_rf_test[,2]
pred_rf_test <- as.data.frame(ifelse(pred_rf_test >= 0.5, "Male", "Female"))
pred_rf_test$Gender <- pred_rf_test[,1]
pred_rf_test[,1] <- NULL
```

Performance metrics of the test set are given below.

```r
cm <- table(test$Gender,pred_rf_test$Gender)
tp <- cm[2,2]
tn <- cm[1,1]
fp <- cm[1,2]
fn <- cm[2,1]
precision <- tp/(tp+fp)
recall <- tp/(tp+fn)
accuracy <- (tp+tn)/(tp+tn+fp+fn)
f1_score_test <- 2*(precision*recall)/(precision+recall)

print(cm)
```

```
##
##          Female Male
##   Female   2271  229
##   Male      268 2232
```

```r
print(precision)
```

```
## [1] 0.9069484
```

```r
print(recall)
```

```
## [1] 0.8928
```

```r
print(accuracy)
```

```
## [1] 0.9006
```

```r
print(f1_score_test)
```

```
## [1] 0.8998186
```

When Random Forest algorithm is used, training performance is seen as 99% (accuracy, F1 score etc.) which is very good. However, on the test set, performance drops at around 90% (accuracy, F1 score etc.). This ratio is also very good. However, a slight overfit may seem to occur whether we do the predictions using Random Forest Classification Algorithm. Logistic Regression algorithm should also be checked for whether there is overfitting or not.