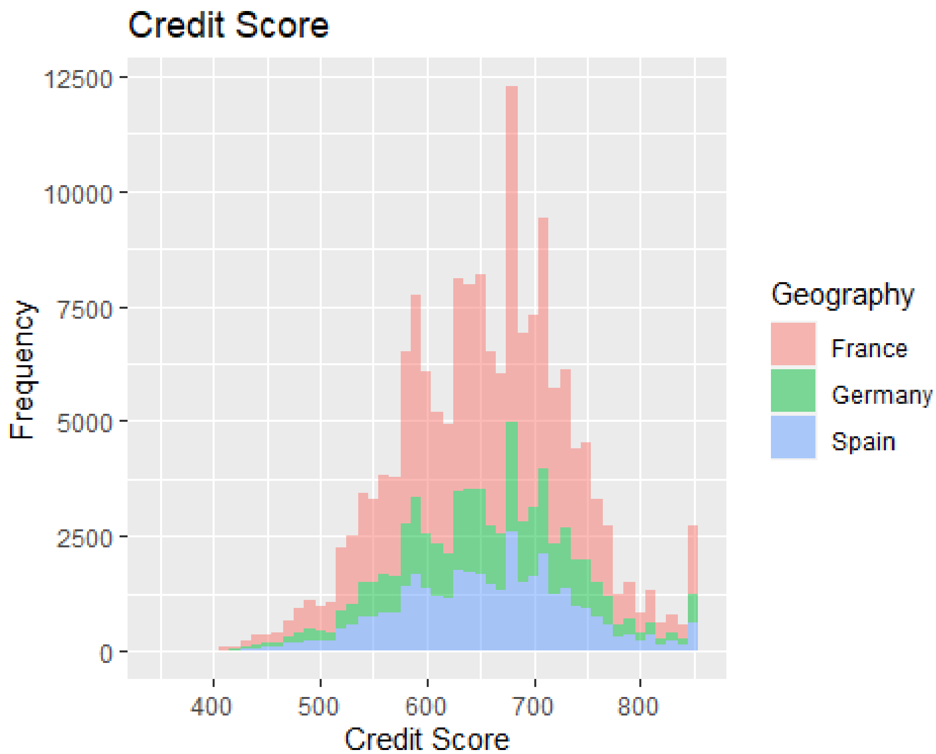
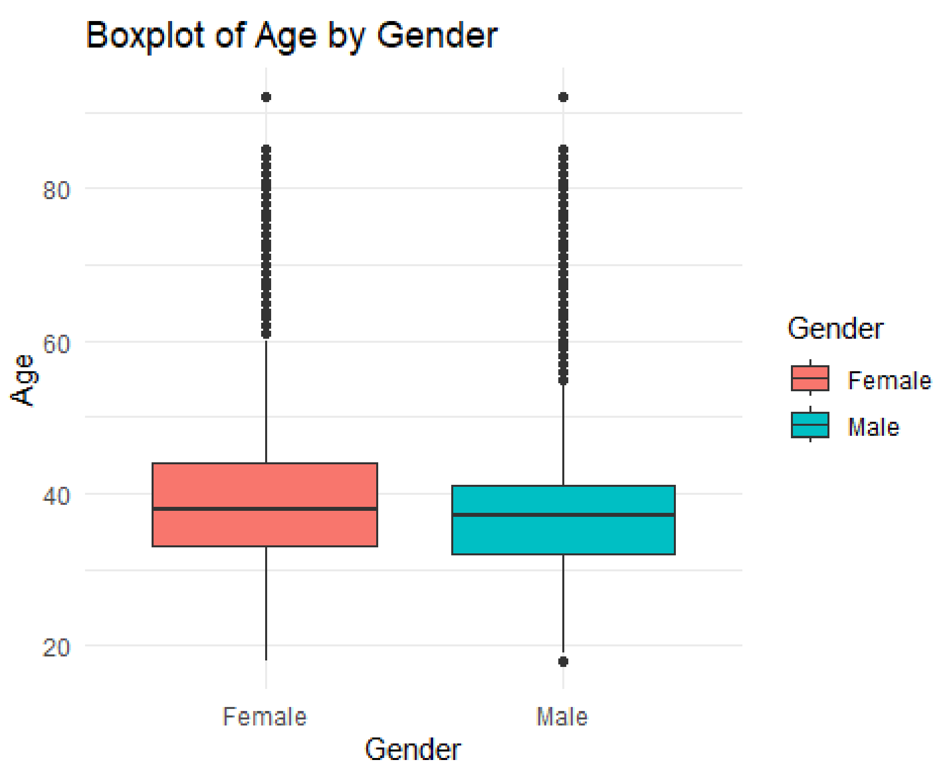
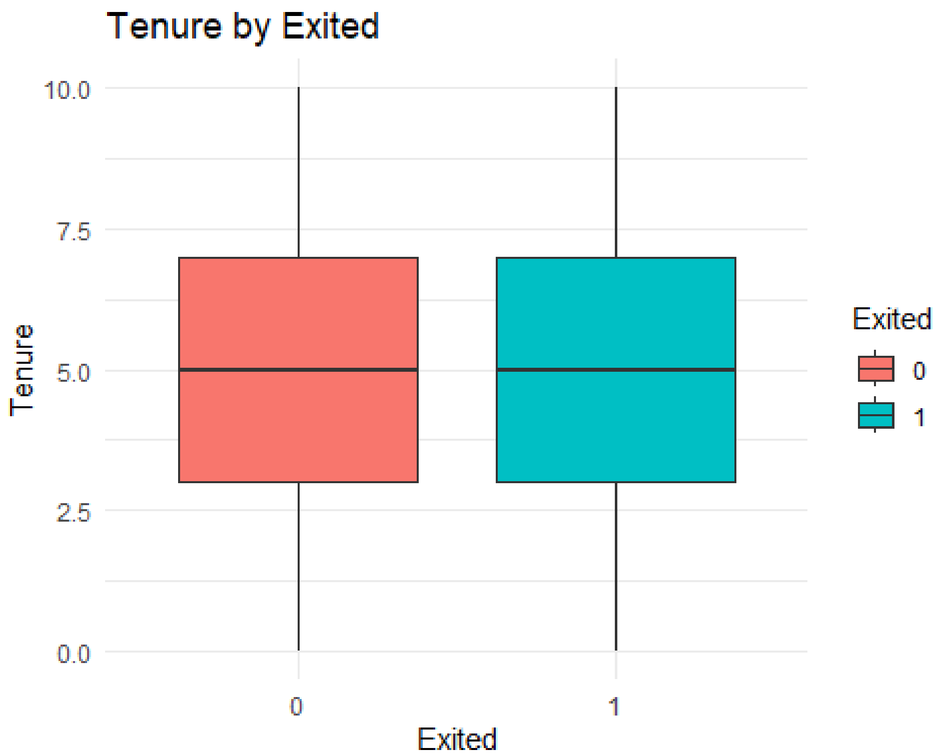
```r
# set current working directory
setwd ("C:/Programming/R/Classification")

# read data
mydata <- read.csv("BankChurnDataset.csv")

# give me information about the data
str(mydata)

## 'data.frame':    165034 obs. of  14 variables:
##  $ id             : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ CustomerId     : int  15674932 15749177 15694510 15741417 15766172
15771669 15692819 15669611 15691707 15591721 ...
##  $ Surname        : chr  "Okwudilichukwu" "Okwudiliolisa" "Hsueh" "Kao" ...
##  $ CreditScore    : int  668 627 678 581 716 588 593 678 676 583 ...
##  $ Geography      : chr  "France" "France" "France" "France" ...
##  $ Gender         : chr  "Male" "Male" "Male" "Male" ...
##  $ Age            : num  33 33 NA 34 33 36 30 37 43 40 ...
##  $ Tenure         : int  3 1 10 2 5 4 8 1 4 4 ...
##  $ Balance        : num  0 0 0 148883 0 ...
##  $ NumOfProducts  : int  2 2 2 1 2 1 1 1 2 1 ...
##  $ HasCrCard      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ IsActiveMember : int  0 1 0 1 1 0 0 0 0 1 ...
##  $ EstimatedSalary: num  181450 49504 184867 NA 15069 ...
##  $ Exited         : int  0 0 0 0 0 1 0 0 0 0 ...

# convert string to categorical variables
mydata$Geography <- factor(mydata$Geography)
mydata$Gender <- factor(mydata$Gender)

# convert dummy int types to categorical as well
mydata$HasCrCard <- factor(mydata$HasCrCard)
mydata$IsActiveMember <- factor(mydata$IsActiveMember)
mydata$Exited <- factor(mydata$Exited, levels = c(0, 1), labels = c("0", "1"))


# delete rows that contains missing values
mydata <- na.omit(mydata)

# visualize variables
library(ggplot2)
ggplot(mydata, aes(x = CreditScore, fill=Geography))+
  geom_histogram(binwidth = 10,  alpha = 0.5)+
  labs(title = "Credit Score",
       x = "Credit Score",
       y = "Frequency")
```



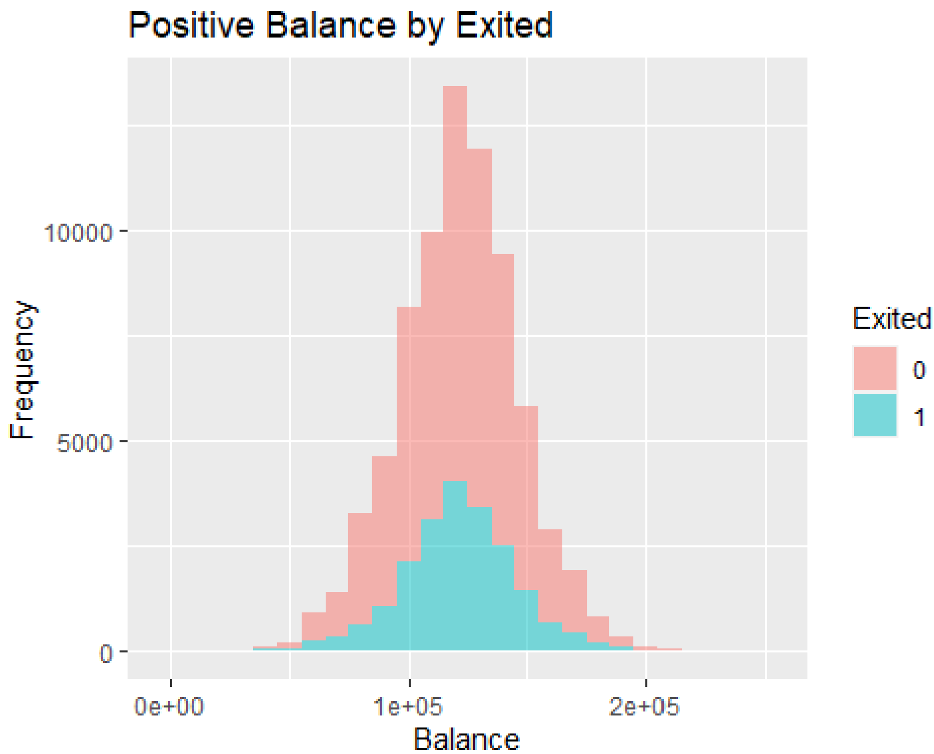

```r
ggplot(mydata, aes(x = Gender , y = Age, fill = Gender)) +
  geom_boxplot() +
  labs(title = "Boxplot of Age by Gender", x = "Gender", y = "Age") +
  theme_minimal()
```



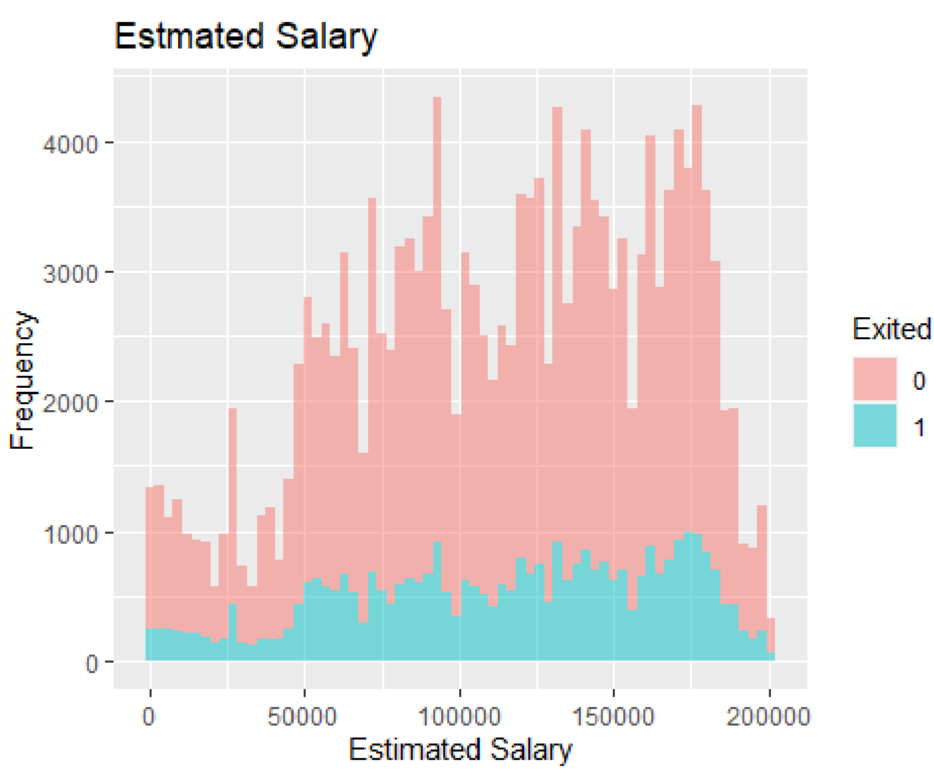
```r
ggplot(mydata, aes(x = Exited, y = Tenure, fill = Exited)) +
  geom_boxplot() +
  labs(title = "Tenure by Exited", x = "Exited", y = "Tenure") +
  theme_minimal()
```



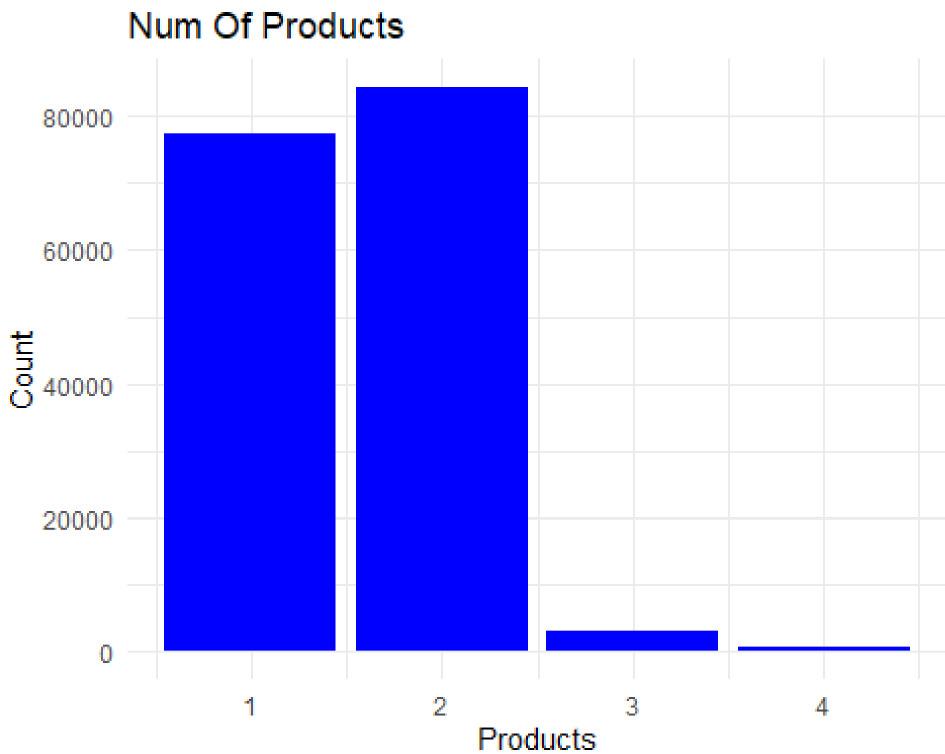

```r
ggplot(mydata, aes(x = Balance, fill=Exited))+
  geom_histogram(binwidth = 10000,  alpha = 0.5, data = subset(mydata, Balance >
0))+
  labs(title = "Positive Balance by Exited",
       x = "Balance",
       y = "Frequency")
```



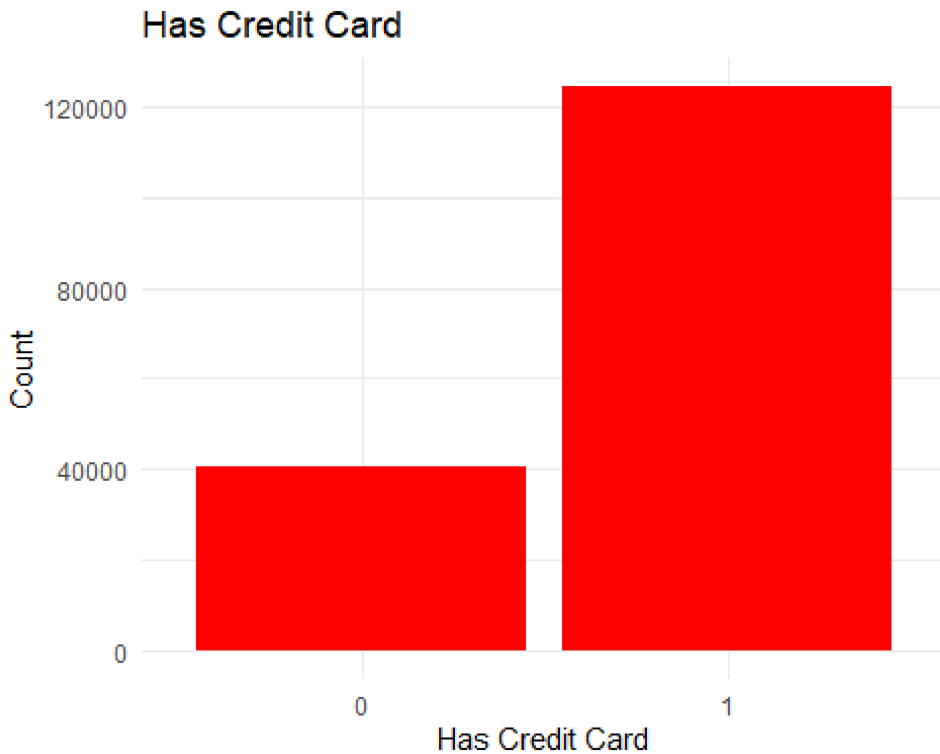

```r
ggplot(mydata, aes(x = EstimatedSalary, fill=Exited))+
  geom_histogram(binwidth = 3000,  alpha = 0.5)+
  labs(title = "Estmated Salary",
       x = "Estimated Salary",
       y = "Frequency")
```



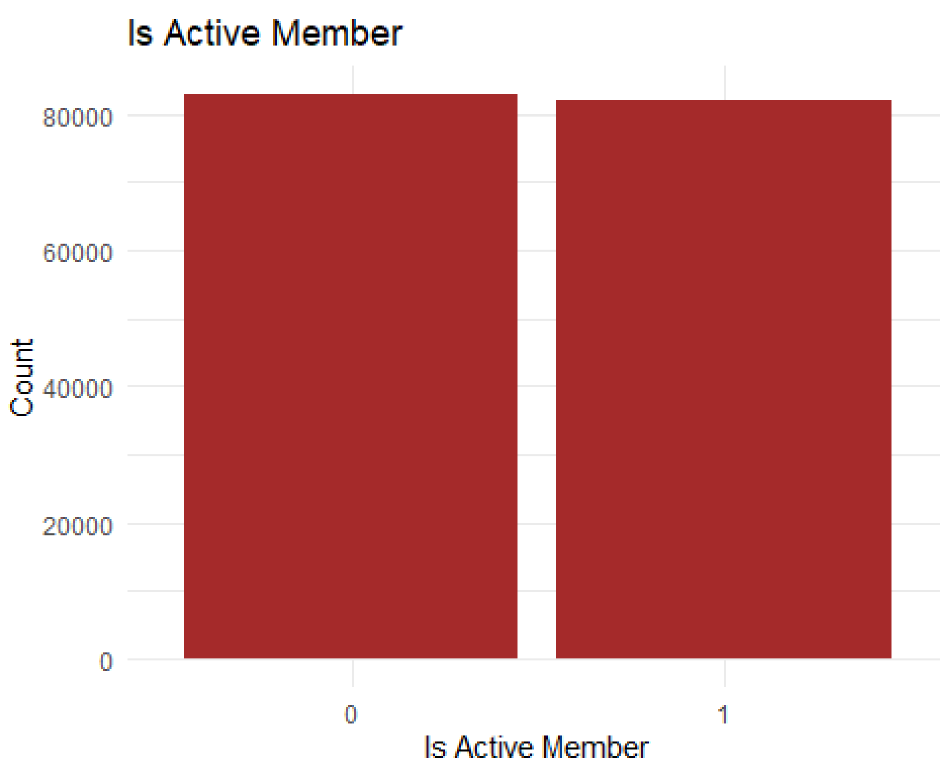
```r
ggplot(mydata, aes(x = NumOfProducts)) +
  geom_bar(stat = "count", fill = 'blue') +
  labs(title = "Num Of Products", x = "Products", y = "Count") +
  theme_minimal()
```



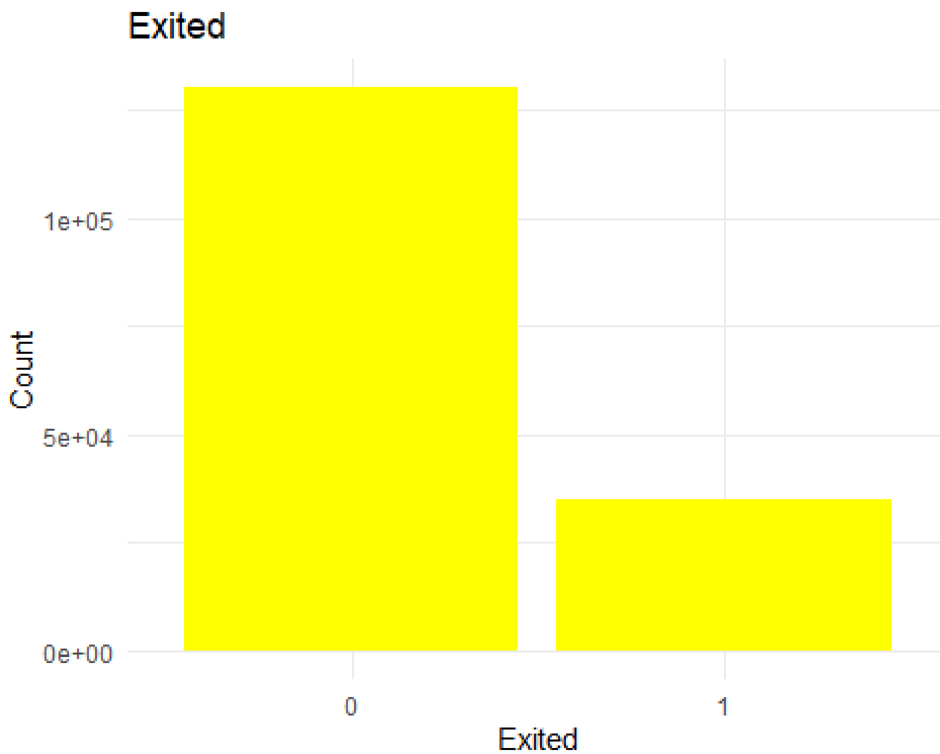

```r
ggplot(mydata, aes(x = HasCrCard)) +
  geom_bar(stat = "count", fill = 'red') +
  labs(title = "Has Credit Card", x = "Has Credit Card", y = "Count") +
  theme_minimal()
```




```
ggplot(mydata, aes(x = IsActiveMember)) +
  geom_bar(stat = "count", fill = 'brown') +
  labs(title = "Is Active Member", x = "Is Active Member", y = "Count") +
  theme_minimal()
```



```r
ggplot(mydata, aes(x = Exited)) +
  geom_bar(stat = "count", fill = 'yellow') +
  labs(title = "Exited", x = "Exited", y = "Count") +
  theme_minimal()
```




```r
# load library for sampling
library(caret)

## Loading required package: lattice

# set seed for random so that each run gives same result
set.seed(321654)

# split data into train and test sets
# 80% for training, 20% for test
v <- createDataPartition(mydata$Exited, p = 0.80, list = FALSE)

# -c(1, 2, 3) exclude id, CustomerId, Surname variables
# postive vector means include those
mydata_train <- mydata[v,-c(1, 2, 3)]

# negative vector means exclude those
mydata_test <- mydata[-v,-c(1, 2, 3)]


# build logistic regression model
# model uses 80% of the data i.e. train data
model <- glm(Exited ~ ., data = mydata_train, family =  binomial("logit"))
```

```r
# show model summary
# stars in each variable means each variable is statistically significant
summary(model)
```

```
##
## Call:
## glm(formula = Exited ~ ., family = binomial("logit"), data = mydata_train)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.431e+00  8.201e-02 -29.637  < 2e-16 ***
## CreditScore      -7.509e-04  9.703e-05  -7.739 1.01e-14 ***
## GeographyGermany  1.141e+00  2.201e-02  51.827  < 2e-16 ***
## GeographySpain    3.289e-02  2.054e-02   1.602    0.109
## GenderMale       -6.756e-01  1.564e-02 -43.210  < 2e-16 ***
## Age               9.352e-02  8.808e-04 106.176  < 2e-16 ***
## Tenure           -1.398e-02  2.771e-03  -5.046 4.51e-07 ***
## Balance          -1.929e-06  1.587e-07 -12.156  < 2e-16 ***
## NumOfProducts    -9.106e-01  1.542e-02 -59.059  < 2e-16 ***
## HasCrCard1       -1.677e-01  1.781e-02  -9.419  < 2e-16 ***
## IsActiveMember1  -1.286e+00  1.676e-02 -76.691  < 2e-16 ***
## EstimatedSalary   8.889e-07  1.553e-07   5.724 1.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 136264  on 132019  degrees of freedom
## Residual deviance: 104529  on 132008  degrees of freedom
## AIC: 104553
##
## Number of Fisher Scoring iterations: 5
```

```r
# check model accuracy
train_predictions <- predict (model, newdata = mydata_train, type = "response")

# convert continuous value to binary
train_class_predictions <- as.numeric(train_predictions > 0.5)

# prediction accuracy for train data
mean(train_class_predictions == mydata_train$Exited)
```

```
## [1] 0.8337449
```

```r
# prediction accuracy for test data
test_predictions <- predict (model, newdata = mydata_test, type = "response")

# convert continuous value to binary
test_class_predictions <- as.numeric(test_predictions > 0.5)

# prediction accuracy for train data
mean(test_class_predictions == mydata_test$Exited)
```

```r
## [1] 0.8352674

# calculate confusion matrix
(confusion_matrix <- table(predicted = train_class_predictions, actual =
mydata_train$Exited))

##          actual
## predicted     0     1
##         0 99381 17246
##         1  4703 10690

# extract TP, TN, FP, FN
TP <- confusion_matrix[2, 2]
TN <- confusion_matrix[1, 1]
FP <- confusion_matrix[1, 2]
FN <- confusion_matrix[2, 1]

# calculate sensitivity and specificity
sensitivity <- TP / (TP + FN)
specificity <- TN / (TN + FP)

# show results
print(paste("Sensitivity:", sensitivity))

## [1] "Sensitivity: 0.694471513025401"

print(paste("Specificity:", specificity))

## [1] "Specificity: 0.85212686599158"

# sensitivity is 70%. It shows how well model predicts exited cases
# specificity is 85%. It shows how well model predicts not exited cases

# now predict new data set from another file

# read data
mydata_new <- read.csv("NewCustomerDataset.csv")

# convert string to categorical variables
mydata_new$Geography <- factor(mydata_new$Geography)
mydata_new$Gender <- factor(mydata_new$Gender)

# convert dummy int types to categorical as well
mydata_new$HasCrCard <- factor(mydata_new$HasCrCard)
mydata_new$IsActiveMember <- factor(mydata_new$IsActiveMember)

# make prediction using the model
predictions <- predict (model, newdata = mydata_new, type = "response")

# convert continuous value to binary
class_predictions <- as.numeric(predictions > 0.5)

# add prediction to new data as new column
```

```r
mydata_new$PredictedExited <- class_predictions

# save new data along with prediction to file
write.csv(mydata_new,"PredictedExited.csv")
```