CHURN MODELLING

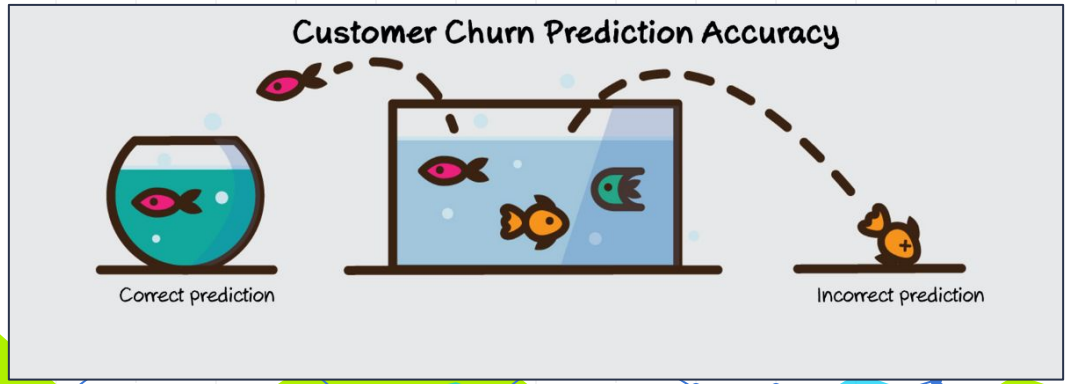
ALY 6015 : INTERMEDIATE ANALYTICS
FINAL PROJECT
GROUP 1

Contributors:
Abhilash Dikshit
Mrityunjay Gupta
Siddharth Alashi
Smit Parmar

Introduction

What is Churn Modelling?

- When a customer (player, subscriber, user, etc.) breaks off contact with a business, this is referred to as customer churn. Once a certain length of time has passed since a client's last interaction with a website or service, online firms commonly describe that customer as having "churned."
- By examining some of the key qualities and using statistical measures and findings such as regression models, one may predict which group of customers will leave the company.
- A technique called a predictive churn model outlines the phases and stages of customer churn, or the process by which a customer leaves your service or product. However, you may battle for retention with a dynamic churn model by responding to the indicators as they change.



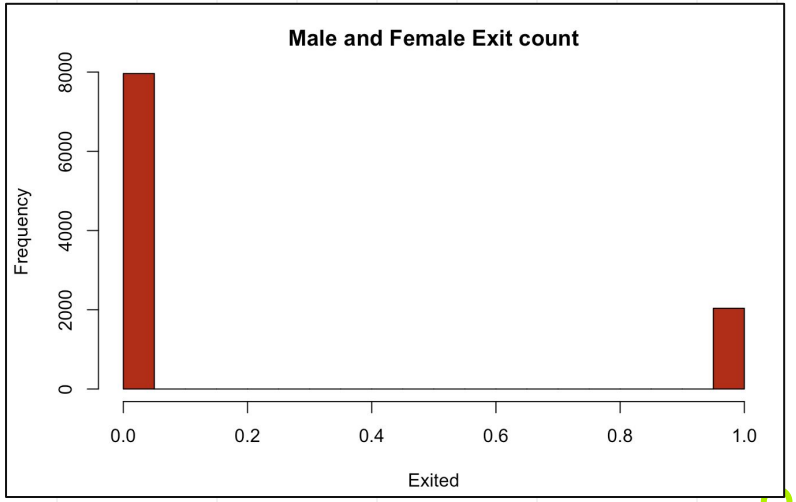
Exploratory Data Analysis

Initially, we read the required Libraries and our Dataset as well as displayed the number of rows and columns before and after the cleanup. We also used functions such as dim(), str(), summary() and unique() functions to understand the data.

```
# Load libraries
```{r}
my_packages = c("plyr", "corrplot", "plotly", "ggplot2", "psych", "tidyr", "tidyverse", "gridExtra", "caret", "MASS", "randomForest", "party", "readr", "class",
"randomForest", "rpart", "rpart.plot", "Metrics", "easypackages","leaps","e1071","factoextra","glmnet","dplyr","BSDA")
#install.packages(my_packages)
lapply(my_packages, require, character.only = T)
```
```

```
> Churn_Modelling <- read.csv("Churn_Modelling-3.csv", header = T)
> cat("Number of Rows before cleanup:", nrow(Churn_Modelling), "\n") # Printing string and variable row count
 on the same line
Number of Rows before cleanup: 10000
> cat("Number of Columns before cleanup:", ncol(Churn_Modelling), "\n")
Number of Columns before cleanup: 14
> cat("Blank cells count before cleanup:", sum(!complete.cases(Churn_Modelling))) # Displaying Blank Cells Co
unt for uncleaned data
Blank cells count before cleanup: 0
```

For Data Cleaning, we converted the values of our Target Variables as 0 for Females and 1 for Males. The histogram for the following is shown here.
Count of Male: 2037
Count of Female: 7693




```
# Data Cleaning
```{r}
Churn_Modelling[,c('RowNumber', 'CustomerId', 'Surname', 'Geography')] <- NULL
Churn_Modelling$Gender[Churn_Modelling$Gender=='Male'] <- 1
Churn_Modelling$Gender[Churn_Modelling$Gender=='Female'] <- 0
table(Churn_Modelling$Exited)
hist(Churn_Modelling$Exited, col = "#c00000", main = "Male and Female Exit count", xlab = "Exited")
y <- Churn_Modelling$Exited
x_data <- Churn_Modelling
```
```

Training and Testing Data Split(70:30); also on normal data (70:30)

- We split the data into train and test data with 70:30 split for the following project.
- We split the training and test data with 70:30 split on normal data.
- The results are as follows:

```
# Training and Testing Data Split (70:30)
```{r}
set.seed(1)
row.number <- sample(x=1:nrow(x), size=0.7*nrow(x))
x_train = x[row.number,]
x_test = x[-row.number,]
headTail(x_train, top = 4, bottom = 4, ellipsis = F)
headTail(x_test, top = 4, bottom = 4, ellipsis = F)
```

Training and Testing Data Split on normal data
```{r}
set.seed(1)
row.number <- sample(x=1:nrow(x_data), size=0.7*nrow(x_data))
train = x_data[row.number,]
test = x_data[-row.number,]
headTail(train, top = 4, bottom = 4, ellipsis = F)
headTail(test, top = 4, bottom = 4, ellipsis = F)
```
```

	CreditScore <dbl>	Gender <dbl>	Age <dbl>	Tenure <dbl>	Balance <dbl>	NumOfProducts <dbl>	HasCrCard <dbl>	IsActiveMember <dbl>	EstimatedSalary <dbl>
1	619	0	42	2	0.00	1	1	1	101348.88
10	684	1	27	2	134603.88	1	1	1	71725.73
11	528	1	31	6	102016.72	2	0	0	80181.12
12	497	1	24	3	0.00	2	1	0	76390.01
9995	800	0	29	2	0.00	2	0	0	167773.55
9996	771	1	39	5	0.00	2	1	0	96270.64
9999	772	1	42	3	75075.31	2	1	0	92888.52
10000	792	0	28	4	130142.79	1	1	0	38190.78

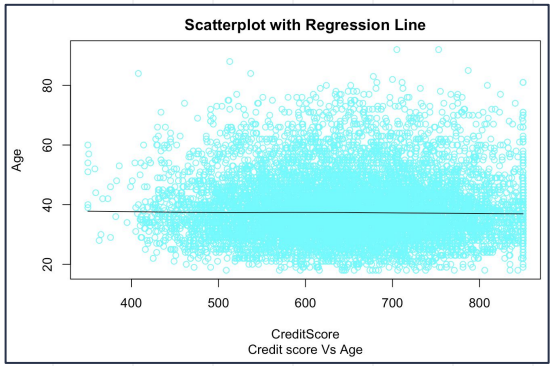
CreditScore <dbl>	Gender <dbl>	Age <dbl>	Tenure <dbl>	Balance <dbl>	NumOfProducts <dbl>	HasCrCard <dbl>	IsActiveMember <dbl>	EstimatedSalary <dbl>	Exited <dbl>
619	0	42	2	0.00	1	1	1	101348.88	1
684	1	27	2	134603.88	1	1	1	71725.73	0
528	1	31	6	102016.72	2	0	0	80181.12	0
497	1	24	3	0.00	2	1	0	76390.01	0
800	0	29	2	0.00	2	0	0	167773.55	0
771	1	39	5	0.00	2	1	0	96270.64	0
772	1	42	3	75075.31	2	1	0	92888.52	1
792	0	28	4	130142.79	1	1	0	38190.78	0

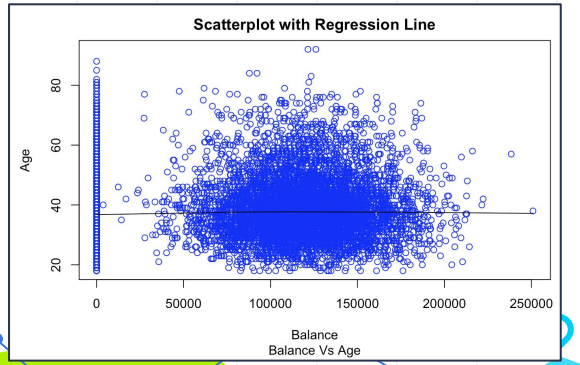
Scatter Plots

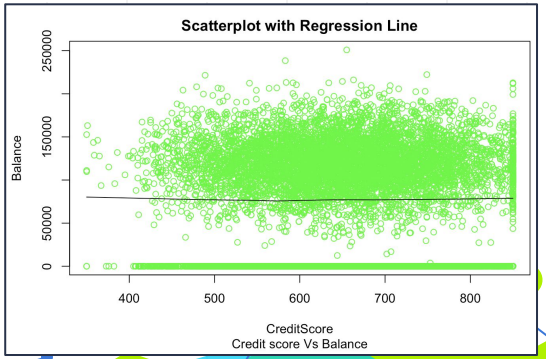
```
> scatter.smooth(x = x_data$Balance, y = x_data$EstimatedSalary,  main="Scatterplot with Regression Line", sub= "Balance Vs Estimated Salary", xlab="Balance", ylab="EstimatedSalary", col= "coral")
> scatter.smooth(x = x_data$CreditScore, y = x_data$Age,  main="Scatterplot with Regression Line", sub= "Credit score Vs Age", xlab="CreditScore", ylab="Age", col= "cyan")
> scatter.smooth(x = x_data$Balance, y = x_data$Age,  main="Scatterplot with Regression Line", sub= "Balance Vs Age",xlab="Balance", ylab="Age", col= "blue")
> scatter.smooth(x = x_data$CreditScore, y = x_data$Balance,  main="Scatterplot with Regression Line",sub= "Credit score Vs Balance", xlab="CreditScore", ylab="Balance", col= "green")
```

The plots show minimal correlation between the attributes of below data:

- Balance Vs Estimated Salary
- Credit score Vs Age
- Balance Vs Age
- Credit score Vs Balance



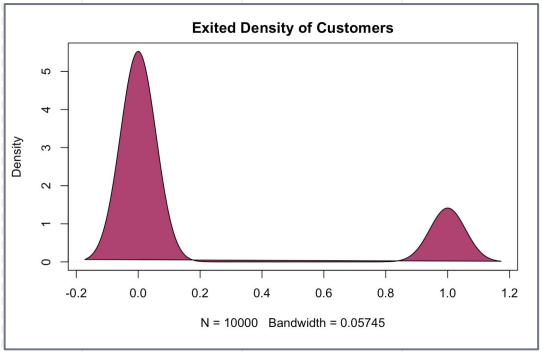


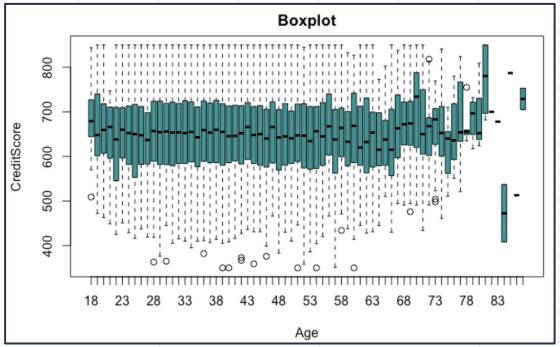


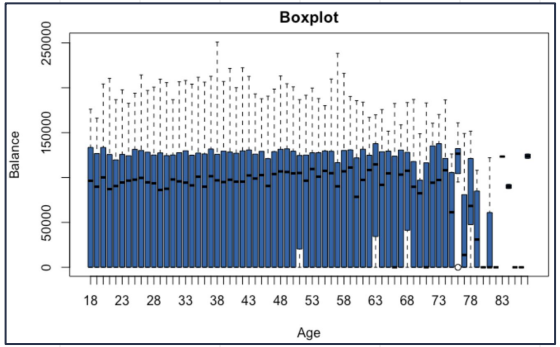
Density Plot and Boxplot

```
> density.Exited <- density(x_data$Exited)
> plot(density.Exited, main="Exited Density of Customers")
> polygon(density.Exited, col="#AE4371")
```

```
> boxplot(x_data$CreditScore ~ x_data$Age, data = x_data, main="Boxplot", xlab="Age", ylab="CreditScore", col="#009999")
> boxplot(x_data$Balance ~ x_data$Age, data = x_data, main="Boxplot", xlab="Age", ylab="Balance",col="#2166AC")
```

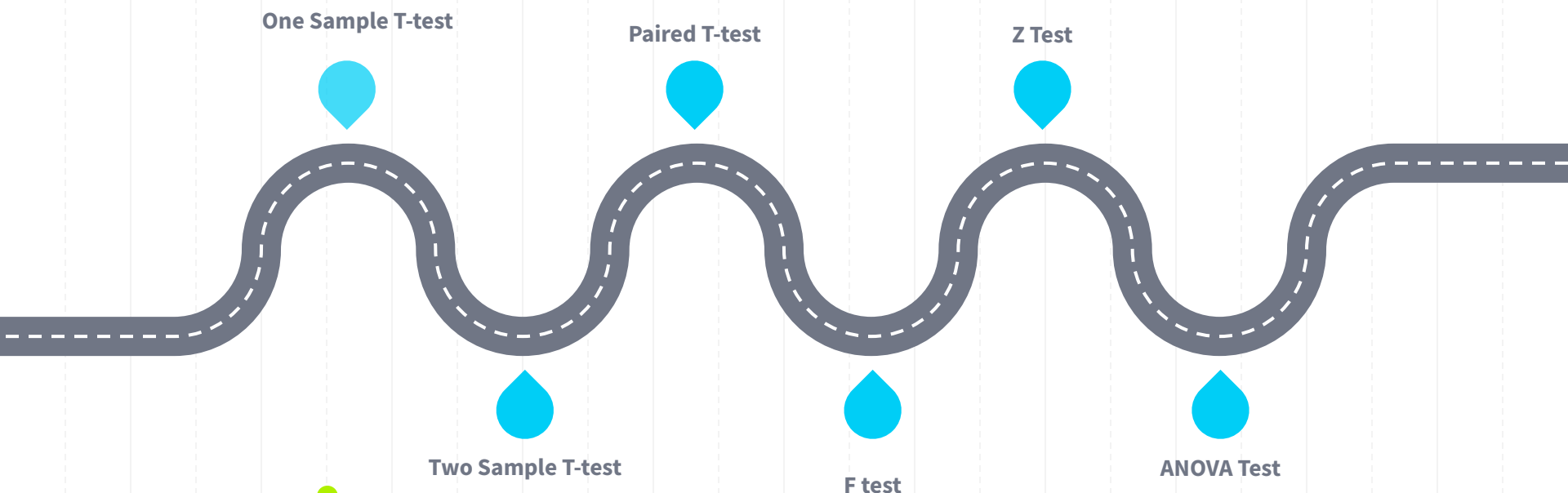






Notches in the boxplots do not coincide and we assume that their true medians differ.

HYPOTHESIS TESTING



Hypothesis Testing

To trust our model and make predictions, we utilize hypothesis testing. When we will use sample data to train our model, we make assumptions about our population. By performing hypothesis testing, we validate these assumptions for a desired significance level.

```
# Hypothesis Testing
# One Sample t-test - Is the credit score greater than 500?
#Null Hypothesis : The credit score is equal to 500.
#Alternate Hypothesisis : The credit score is not equal to 500
```{r}
set.seed(1)
onesample <- t.test(Churn_Model$CreditScore, mu = 500, alternative = "greater")
onesample
```
```

```
        One Sample t-test

data:  Churn_Model$CreditScore
t = 155.74, df = 9999, p-value < 0.00000000000000022
alternative hypothesis: true mean is greater than 500
95 percent confidence interval:
 648.9388      Inf
sample estimates:
mean of x
 650.5288
```

Hypothesis Testing of one t -test:

- **Null hypothesis**: Credit score =500
- **Alternate hypothesis**: Credit score greater than 500
- **Output**: We reject null hypothesis stating that credit score is greater than 500 as p value < 0.05.

```
# Two Sample t-test - Do males and females have the same credit scores?
#Null Hypothesis - Males and Females have the same Credit Score
#Alternate Hypothesis - Males and Females do not have the same Credit Score.
```{r}
male_cust <- subset(Churn_Model, subset = (Churn_Model$Gender == 'Male'))
female_cust <- subset(Churn_Model, subset = (Churn_Model$Gender == 'Female'))
twosample <- t.test(male_cust$CreditScore, female_cust$CreditScore, data = Churn_Model, var.equal = TRUE)
twosample
```
```

```
        Two Sample t-test

data:  male_cust$CreditScore and female_cust$CreditScore
t = -0.28563, df = 9998, p-value = 0.7752
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.359797  3.250804
sample estimates:
mean of x mean of y
 650.2769  650.8314
```

Hypothesis Testing of two t -test

- **Null hypothesis:** Male credit score = Female credit score
- **Alternate hypothesis**: Male credit score != Female credit score
- **Output**: We accept the null hypothesis that Male credit score is equal to Female Credit score as p value > 0.05.

Paired T-Test

Paired T-test to check if more males are being churned in comparison to the females.

- The paired t-test captures the correlated nature of the measurements/outcomes.
- **Null hypothesis**: Male customer exited=Female customer exited
- **Alternate hypothesis**: Male customer exited greater than Female customer exited
- **Output**: We don't have enough evidence to reject the null hypothesis stating that male customers exited is equal to female customers exited as p value > 0.05.

```
> pairedtest <- t.test(male_cust$Exited, female_cust$Exited, alternative = "greater")
> pairedtest

        Welch Two Sample t-test

data:  male_cust$Exited and female_cust$Exited
t = -10.561, df = 8985.7, p-value = 1
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -0.0995767        Inf
sample estimates:
mean of x mean of y
0.1645593 0.2507154
```

F-Test

```
> ftest <- var.test(male_cust$EstimatedSalary, female_cust$EstimatedSalary, data = Churn_Model, alternative = "less")
> ftest

        F test to compare two variances

data:  male_cust$EstimatedSalary and female_cust$EstimatedSalary
F = 1.009, num df = 5456, denom df = 4542, p-value = 0.6232
alternative hypothesis: true ratio of variances is less than 1
95 percent confidence interval:
 0.000000 1.057188
sample estimates:
ratio of variances
          1.008983
```

Z-Test

```
#Z-test - Considering the Active Customers, is the mean of Exited Clients more than the ones who chose to stay?
```{r}
active_cust <- subset(Churn_Model, subset = (Churn_Model$IsActiveMember == '1'))
inactive_cust <- subset(Churn_Model, subset = (Churn_Model$IsActiveMember == '0'))
ztest <- z.test(active_cust$Exited, inactive_cust$Exited, alternative = "greater", mu = 0 , sigma.x = sd(active_cust$Exited), sigma.y = sd(inactive_cust$Exited))
ztest
```
```

```
        Two-sample z-Test

data:  active_cust$Exited and inactive_cust$Exited
z = -15.695, p-value = 1
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -0.1390045          NA
sample estimates:
mean of x mean of y
0.1426907 0.2685090
```

F-test to Test the variance of Estimated Salary in males and females:

- **Null hypothesis**: Male customer estimated salary = Female customer estimated salary.
- **Alternate hypothesis**: Male customer estimated salary less than Female customer estimated salary
- **Output**: We don't have enough evidence to reject null hypothesis stating that male customer estimated salary less than female customer estimated salary as p value > 0.05.

Z-test considering Active customers:

- **Null hypothesis**: Mean of Exited customers = Mean of customers stayed.
- **Alternate hypothesis**: Mean of Exited customers is not equal to the Mean of customers stayed.
- **Output**: We observe that the mean of Exited customers is the same as Mean of Customers Stayed as p value > 0.05..

ANOVA (Analysis of variance)

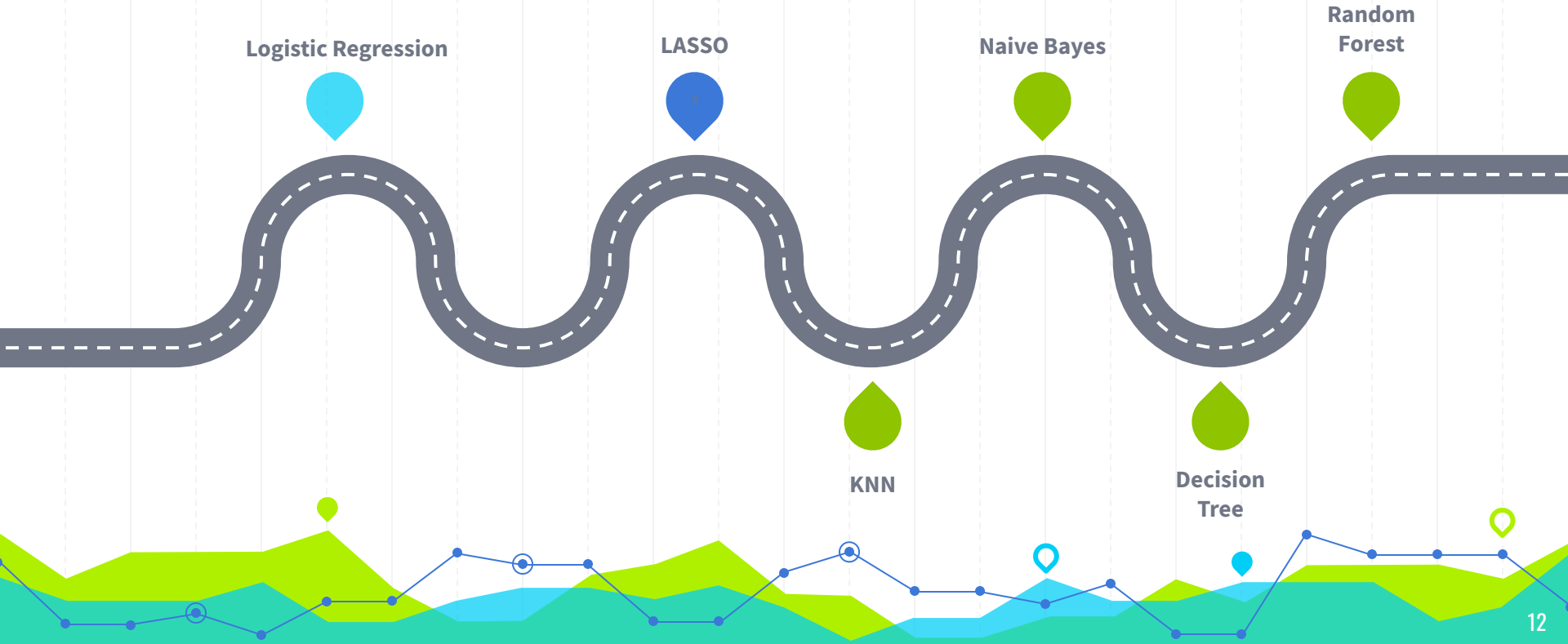
- This code appears to be fitting a multiple linear regression model using the aov function in R.
- The dependent variable y i.e. Exited is being regressed on three independent variables: CreditScore, Balance, and Estimated Salary. These variables are being extracted from a data frame called x_train.
- The aov function is typically used to perform analysis of variance, but in this case it is being used to fit a linear regression model. This is because aov is a general function for fitting linear models, and can be used for both ANOVA and regression.
- The results of the linear regression model are being stored in an object called res.aov. This object contain information about the coefficients, standard errors, and other diagnostic information about the model. This information can be used to make predictions and interpret the results of the regression.
- **Null hypothesis**: The regression coefficient for a given independent variable is equal to zero, meaning that the independent variable has no effect on the dependent variable.
- **Alternate hypothesis**: The regression coefficient for a given independent variable is not equal to zero, meaning that the independent variable has a significant effect on the dependent variable.

```
> res.aov <- aov(y ~ x$CreditScore+x$Balance+x$EstimatedSalary, data = x_train)
> summary(res.aov)
                    Df Sum Sq Mean Sq F value               Pr(>F)
x$CreditScore        1    1.2   1.191   7.449              0.00636 **
x$Balance            1   22.9  22.856 142.988 < 0.0000000000000002 ***
x$EstimatedSalary    1    0.2   0.180   1.127              0.28837
Residuals         9996 1597.8   0.160
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

REGRESSION METHODS

REGULARIZATION TECHNIQUE

CLUSTERING TECHNIQUE



Logistic Regression Model

After this we tried a Logistic Regression Model where Exited is our Response Variable and the Predictor variables are all the variables in our data. The no. of Fisher Scoring Iterations observed were 17 and we predicted whether the customer continues with the credit card services or not. The model achieves 78.93 accuracy.

```
> set.seed(1)
> mylogit <- glm(Exited ~ ., data = x_train, family = binomial(link='logit'))
> summary(mylogit)

Call:
glm(formula = Exited ~ ., family = binomial(link = "logit"),
    data = x_train)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-0.0041345  -0.0012251  -0.0009298  -0.0006498   0.0047233

Coefficients:
                   Estimate   Std. Error z value Pr(>|z|)
(Intercept)         -15.5353     118.2225  -0.131    0.895
CreditScore        -115.6199   34169.2729  -0.003    0.997
Gender           -81117.8656 6739191.3665  -0.012    0.990
Age               12610.4362  275622.5026   0.046    0.964
Tenure            -3662.5122 1159830.4686  -0.003    0.997
Balance               0.8786      58.1856   0.015    0.988
NumOfProducts     -9207.3336 5784859.7914  -0.002    0.999
HasCrCard         -3840.7664 7335049.9281  -0.001    1.000
IsActiveMember  -195933.0375 7173295.7662  -0.027    0.978
EstimatedSalary       0.1247      58.1737   0.002    0.998

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.017985  on 6999  degrees of freedom
Residual deviance: 0.015063  on 6990  degrees of freedom
AIC: 20.011

Number of Fisher Scoring iterations: 17
```

```
> log_pred <- predict(mylogit, newdata = x_test)
> tab_log <- table(x_test$Exited,log_pred)
> tab_log
                        log_pred
 -16.1589584638518 -16.0660599366842 -16.032295867297 -15.9861238880157
                        log_pred
 -15.9347310608181 -15.9345733018718 -15.920262990442 -15.887414246602 -
                        log_pred
 -15.860401066582 -15.7988815220675 -15.7827074647267 -15.7817787454045
                        log_pred
 -15.7613126059018 -15.7598551334599 -15.758980405497 -15.7344034400907
                        log_pred
 -15.716910944618 -15.7062172918322 -15.7058411236137 -15.6998454202414
```

```
                        log_pred
 -12.4618269246084 -12.4475775495383 -12.4454106322811 -12.436458305782 -
                        log_pred
 -12.4036883378039 -12.3865035462809 -12.3738980829616 -12.3698505076689
                        log_pred
 -12.3073381573121 -12.2869211372334 -12.2177247768172 -12.1933267562672
                        log_pred
 -12.1343363919506 -12.0741616237385 -11.8799463009349 -11.8161691257951
 [ reached getOption("max.print") -- omitted 2 rows ]
> log_pred <- ifelse(log_pred>0.5,1,0)
> log_pred_res <- mean(log_pred!= x_test$Exited)
> print(paste('Accuracy', 1-log_pred_res))
[1] "Accuracy 0.789333333333333"
```

LASSO MODEL

- LASSO, short for **Least Absolute Shrinkage and Selection Operator**, is a statistical formula whose main purpose is the feature selection and regularization of data models

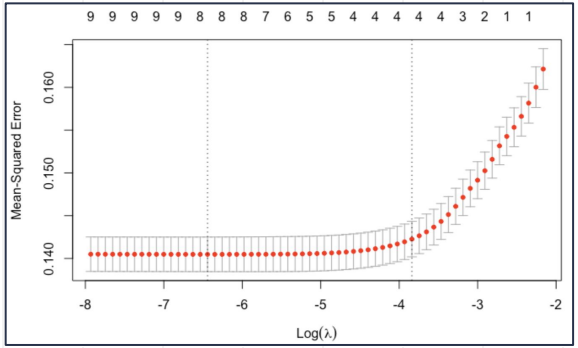
- The LASSO method regularizes model parameters by shrinking the regression coefficients and reducing some of them to zero. This method is significant in the **minimization of prediction errors that are common in statistical models**

```
> train_x = model.matrix(Exited~., x_data)[,-1]
> test_x = model.matrix(Exited~., x_data)[,-1]
> train_y = x_data$Exited
> test_y = x_data$Exited
> print(ncol(train_x))
[1] 9
> print(ncol(test_x))
[1] 9
```

LASSO MODEL

1. This code is performing Lasso regression using the cv.glmnet function in R, which is a cross-validated version of the glmnet function. **The train_x and train_y variables contain the predictor variables and response variable, respectively, for the training dataset.**
2. The nfolds = 10 argument specifies that the function should use **10-fold cross-validation, which means the dataset is divided into 10** equal parts, and the model is trained on 9 parts and tested on the remaining part. This process is repeated 10 times, with each part serving as the validation set once.

```
> Lasso1 = cv.glmnet(train_x, train_y, nfolds =10)
> plot(Lasso1)
> print(log(Lasso1$lambda.min))
[1] -6.443128
> print(log(Lasso1$lambda.1se))
[1] -3.838184
```



```
> model1.min = glmnet(train_x, train_y, alpha=1, lambda = Lasso1$lambda.min)
> model1.min

Call:  glmnet(x = train_x, y = train_y, alpha = 1, lambda = Lasso1$lambda.min)

  Df  %Dev   Lambda
1  8 13.58 0.001591
> coef(model1.min)
10 x 1 sparse Matrix of class "dgCMatrix"
                         s0
(Intercept)     -1.141405e-01
CreditScore     -7.667833e-05
Gender          -7.429133e-02
Age              1.115601e-02
Tenure          -1.312221e-03
Balance          6.756023e-07
NumOfProducts   -2.212653e-03
HasCrCard        .
IsActiveMember  -1.398763e-01
EstimatedSalary  4.386651e-08
```

```
> coef(model1.1se)
10 x 63 sparse Matrix of class "dgCMatrix"
   [[ suppressing 63 column names ‘s0’, ‘s1’, ‘s2’ ... ]]

(Intercept)     0.2037 0.1658125168 0.131290853 0.09983600 0.071175507 0.045061135  0.022218415  0.005095117 -0.010506975 -0.024723020 -3.784427e-02
CreditScore     .      .            .           .          .           .            .            .           .           .           .
Gender          .      .            .           .          .           .            .            .           .           .           .
Age             .      0.0009734258 0.001860375 0.00266853 0.003404891 0.004075836  0.004698047  0.005307137 0.005862118 0.006367795 6.828133e-03
Tenure          .      .            .           .          .           .            .            .           .           .           .
Balance         .      .            .           .          .           .            .            .           .           .           2.384941e-09
NumOfProducts   .      .            .           .          .           .            .            .           .           .           .
HasCrCard       .      .            .           .          .           .            .            .           .           .           .
IsActiveMember  .      .            .           .          .           .           -0.002669085 -0.015450318 -0.027096102 -0.037707306 -4.737209e-02
EstimatedSalary .      .            .           .          .           .            .            .           .           .           .
```

```
> model1.1se = glmnet(train_x, train_y, alpha=1, lambda = Lasso1$lambda.se)
> model1.1se

Call:  glmnet(x = train_x, y = train_y, alpha = 1, lambda = Lasso1$lambda.se)

   Df  %Dev   Lambda
1   0  0.00 0.114900
2   1  1.38 0.104700
3   1  2.53 0.095400
4   1  3.48 0.086930
5   1  4.27 0.079210
```

mode1.min: Negative coefficients indicate that an **increase in the corresponding predictor variable is associated with a decrease in the response variable**, w**hile positive coefficients indicate that an increase in the corresponding predictor variable is associated with an increase in the response variable**.The magnitude of the coefficients represents the strength of the effect of each predictor variable on the response variable, while taking into account the other predictor variables in the model.

Model1.se: Lasso has chosen to exclude the 'CreditScore' and 'Gender' features from the model. The other 8 features have non-zero coefficients, with the largest coefficients corresponding to 'Age' and 'NumOfProducts'.

LASSO

Root mean square error: These lines of code appear to be calculating the **root mean squared error (RMSE) between the actual and predicted values** of the response variable for different sets of data using different models

It appears that the **RMSE for the training set and the testing set are the same**, which suggests that the **model is performing similarly on both sets**. However, **the RMSE for the predictions made on the new x_data set is slightly lower,** which could suggest that the **model is better at predicting on new, unseen data.**

fit1: The output also shows the residuals, which are the differences between the predicted and actual values of the response variable. The residual standard error is a measure of the variability of the residuals, and it provides an estimate of the standard deviation of the errors.

The multiple **R-squared value of 0.1359 indicates that the model explains 13.59% of the variance in the response variable**, and the Adjusted R-squared of 0.1351 is adjusted for the number of predictors in the model. The F-statistic and its associated p-value test whether the model as a whole is significant. In this case, **the p-value is very small, indicating that the model is significant.**

```
> fit1 = lm(Exited~.,data=x_data)
> summary(fit1)

Call:
lm(formula = Exited ~ ., data = x_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.77456 -0.23648 -0.12527  0.02732  1.20241

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1.025e-01  3.387e-02  -3.027  0.00248 **
CreditScore     -9.264e-05  3.877e-05  -2.389  0.01690 *
Gender          -7.733e-02  7.531e-03 -10.267  < 2e-16 ***
Age              1.131e-02  3.589e-04  31.505  < 2e-16 ***
Tenure          -1.849e-03  1.296e-03  -1.426  0.15392
Balance          6.938e-07  6.306e-08  11.003  < 2e-16 ***
NumOfProducts   -4.262e-03  6.766e-03  -0.630  0.52873
HasCrCard       -2.958e-03  8.222e-03  -0.360  0.71901
IsActiveMember  -1.432e-01  7.532e-03 -19.016  < 2e-16 ***
EstimatedSalary  7.118e-08  6.516e-08   1.092  0.27474
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3746 on 9990 degrees of freedom
Multiple R-squared:  0.1359,    Adjusted R-squared:  0.1351
F-statistic: 174.6 on 9 and 9990 DF,  p-value: < 2.2e-16
```

Root mean square error

```
> pre.fit= predict(fit1, new=x_data)
> rmse(x_data$Exited, pre.fit)
[1] 0.3743791
>
> pre.model1.1se <- predict(model1.1se, new
> rmse(train_y, pre.model1.1se)
[1] 0.3786023
>
> pre.test.model1.1se <- predict(model1.1se
> rmse(test_y, pre.test.model1.1se)
[1] 0.3786023
```

KNN Model

KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. The Accuracy of this KNN Model was 79.9%

```
> set.seed(1)
> x.train.target<- x[1:7000,10]
> x.test.target<- x[7001:10000,10]
> knn_model <- knn(x_train,x_test,cl=x.train.target,k=13)
> summary(knn_model)
                     0 0.00000398568199542691
                  2977                       23
> tab <- table(knn_model,x.test.target)
> tab
                        x.test.target
knn_model                   0 0.00000398568199542691
  0                      2390                      587
  0.00000398568199542691   16                        7
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(tab)
[1] 79.9
```

Naive Bayes

The output of the code shows the trained Naive Bayes model, which includes the a-priori probabilities of the two classes (0 and 3.98568199542691e-06), and the conditional probabilities of each predictor variable given the class. The output also shows the confusion matrix (tab_naive), which is a table that shows the number of true positives, false positives, true negatives, and false negatives. The accuracy of the Naive Bayes classifier is 81.63333%.

```
> tab_naive <- table(NB_Predictions,x_test$Exited)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(tab_naive)
[1] 81.63333
```

```
> NBclassfier=naive_bayes(as.factor(Exited) ~ ., data=x_train)
> print(NBclassfier)

================================== Naive Bayes ==================================

 Call:
naive_bayes.formula(formula = as.factor(Exited) ~ ., data = x_train)

---------------------------------------------------------------------------------

Laplace smoothing: 0

---------------------------------------------------------------------------------

 A priori probabilities:

                  0 0.00000398568199542691
          0.7992857               0.2007143

---------------------------------------------------------------------------------

 Tables:

---------------------------------------------------------------------------------
 ::: CreditScore (Gaussian)
---------------------------------------------------------------------------------

CreditScore            0 0.00000398568199542691
       mean 0.0025984467           0.0025681749
       sd   0.0003804919           0.0004015496

---------------------------------------------------------------------------------
 ::: Gender (Gaussian)
---------------------------------------------------------------------------------

Gender              0 0.00000398568199542691
  mean 0.000002293103          0.000001835399
  sd   0.000001970267          0.000001987319

---------------------------------------------------------------------------------
 ::: Age (Gaussian)
---------------------------------------------------------------------------------

Age                0 0.00000398568199542691
  mean 0.00014822463           0.00017815573
  sd   0.00003923128           0.00003780397

---------------------------------------------------------------------------------
 ::: Tenure (Gaussian)
---------------------------------------------------------------------------------

Tenure             0 0.00000398568199542691
  mean 0.00002009724           0.00001956530
  sd   0.00001144682           0.00001162519

---------------------------------------------------------------------------------

Balance          0 0.00000398568199542691
   mean 0.2900137               0.3632354
   sd   0.2503230               0.2313600

---------------------------------------------------------------------------------

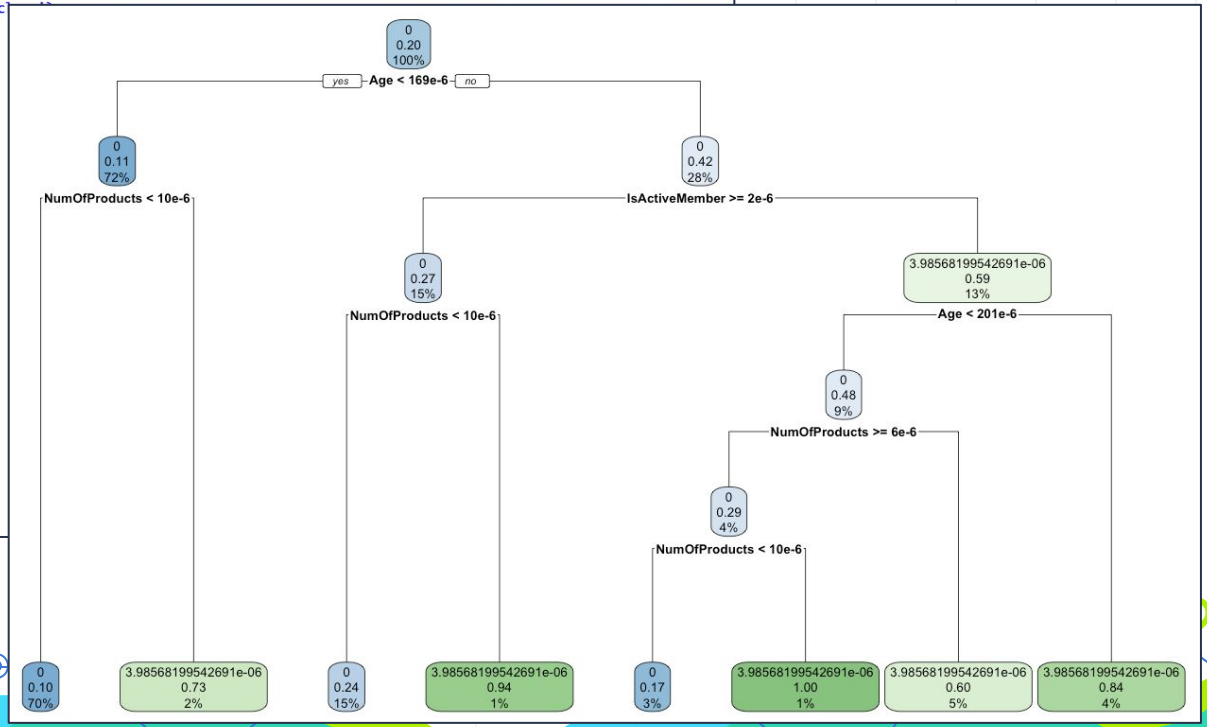
# ... and 4 more tables

---------------------------------------------------------------------------------
```

Decision Tree

Classification Decision tree, predictive analysis using train and test data, confusion matrix to calculate accuracy has been show in the given figure. The decision tree model performed better than the logistic regression and Naive Bayes models in terms of accuracy. The accuracy achieved is 84.8%.

```
> set.seed(1)
> fit <- rpart(y~x$CreditScore+x$Gender+x$Age+x$Tenure+x$Balance+x$NumOfProducts+x$HasCrCard+x$IsActiveMember+x$EstimatedSalary, data = x_train, method = 'class')
> fit <- rpart(as.factor(x_train$Exited) ~ ., data = x_train, method = 'cl
> rpart.plot(fit, extra = 106)
> predict_unseen <-predict(fit, x_test, type = 'class')
> table_mat <- table(x_test$Exited, predict_unseen)
> table_mat
                    predict_unseen
                        0 3.98568199542691e-06
  0                  2275                   93
  3.98568199542691e-06  363                  269
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(table_mat)
[1] 84.8
> library(party)
> fit <- ctree(as.factor(x_train$Exited) ~ ., data = x_train)
> plot(fit, main="Conditional Inference Tree for Churn Model")
> predict_unseen <- predict(fit, x_test)
> table_mat <- table(x_test$Exited, predict_unseen)
> table_mat
                    predict_unseen
                        0 3.98568199542691e-06
  0                  2321                   47
  3.98568199542691e-06  452                  180
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(table_mat)
[1] 83.36667
> mean(predict_unseen == x_test$Exited)
[1] 0.8336667
```



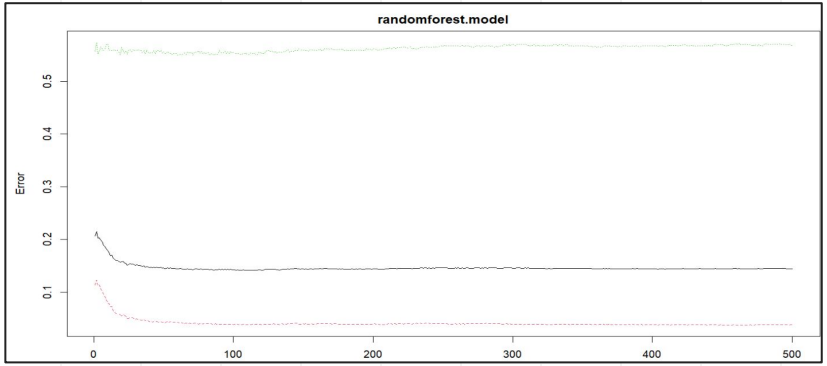
Random Forest

The number of variables at each split is 3. The important variables are shown below.

```
> set.seed(1)
> randomforest.model <- randomForest(as.factor(x_train$Exited) ~ ., data = x_train, importance = TRUE)
> randomforest.model

Call:
 randomForest(formula = as.factor(x_train$Exited) ~ ., data = x_train,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 14.44%
Confusion matrix:
                     0 3.98568199542691e-06 class.error
0                 5382                 213  0.03806971
3.98568199542691e-06  798                 607  0.56797153
```

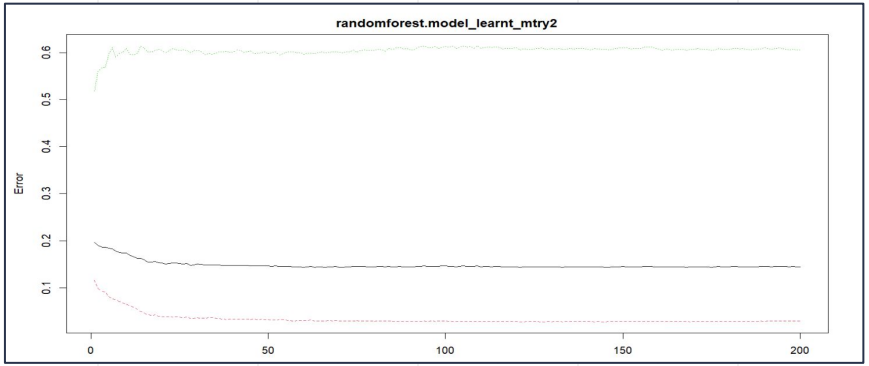


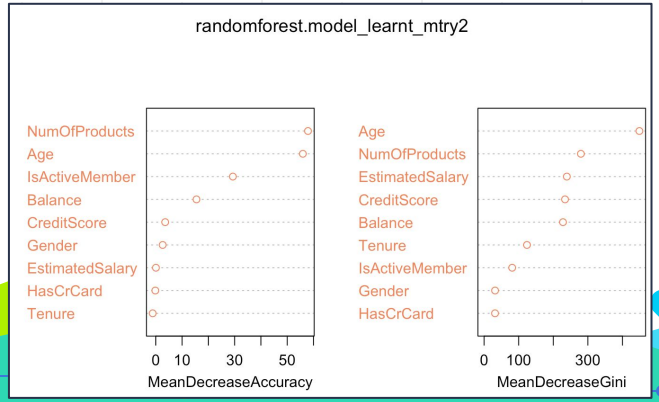

```
> importance(randomforest.model_learnt_mtry2, col= "coral")
                                0 3.98568199542691e-06 MeanDecreaseAccuracy MeanDecreaseGini
CreditScore          1.9848228          3.433407          3.57129033        234.00079
Gender               0.7107777          3.678227          2.61730875         31.66621
Age                 40.4685590         56.768949         55.95078409        448.98205
Tenure              -2.2933919          1.281578         -1.18151718        123.95662
Balance             14.6364979          4.660788         15.47448344        228.02122
NumOfProducts       40.2374598         47.567047         57.89393502        279.81865
HasCrCard           -0.8893712          1.086822         -0.17609371         31.58209
IsActiveMember      25.7883822         23.012815         29.32261929         81.14729
EstimatedSalary     -0.4532287          0.723849          0.04895787        239.21403
```

```
> randomforest.model_learnt_mtry2 <- randomForest(as.factor(x_train$Exited) ~ ., data = x_train, ntree = 200, mtry = 2, importance = TRUE)
> print(randomforest.model_learnt_mtry2)

Call:
 randomForest(formula = as.factor(x_train$Exited) ~ ., data = x_train,      ntree = 200, mtry = 2, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 2

        OOB estimate of  error rate: 14.49%
Confusion matrix:
                     0 3.98568199542691e-06 class.error
0                 5433                 162  0.02895442
3.98568199542691e-06  852                 553  0.60640569
```





```
> randomforest.model_learnt_mtry4 <- randomForest(as.factor(x_train$Exited) ~ ., data = x_train, ntree = 200, mtry = 4, importance = TRUE)
> randomforest.model_learnt_mtry4

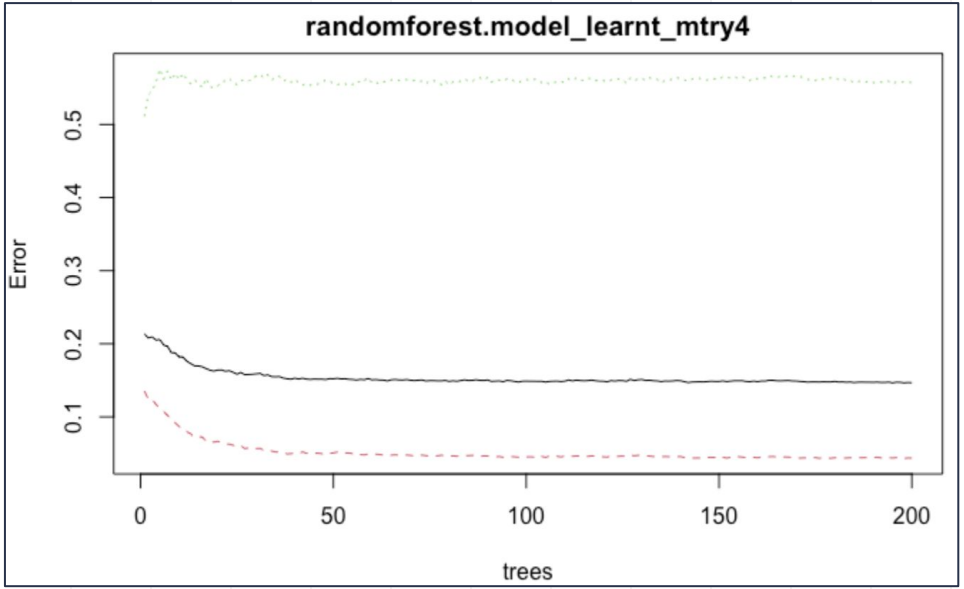
Call:
 randomForest(formula = as.factor(x_train$Exited) ~ ., data = x_train,      ntree = 200, mtry = 4, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 4

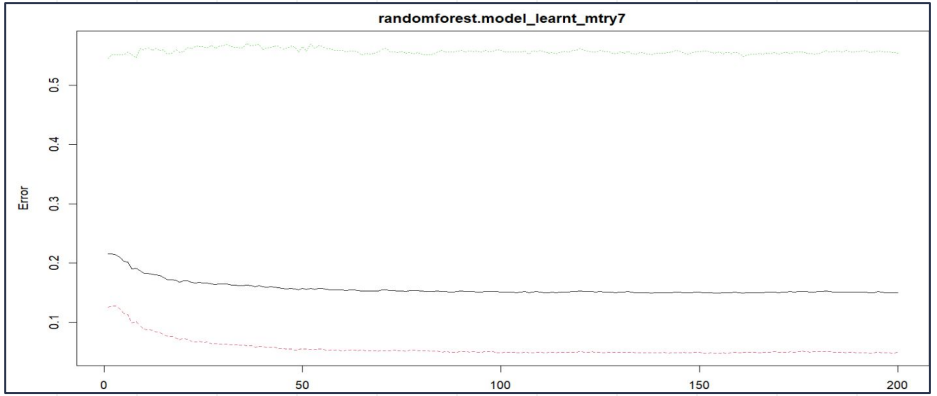
        OOB estimate of  error rate: 14.67%
Confusion matrix:
                      0 3.98568199542691e-06 class.error
0                  5351                 244  0.04361037
3.98568199542691e-06  783                 622  0.55729537
```

```
> randomforest.model_learnt_mtry7 <- randomForest(as.factor(x_train$Exited) ~ ., data = x_train, ntree = 200, mtry = 7, importance = TRUE)
> randomforest.model_learnt_mtry7

Call:
 randomForest(formula = as.factor(x_train$Exited) ~ ., data = x_train,      ntree = 200, mtry = 7, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 200
No. of variables tried at each split: 7

        OOB estimate of  error rate: 15.04%
Confusion matrix:
                      0 3.98568199542691e-06 class.error
0                  5321                 274   0.0489723
3.98568199542691e-06  779                 626   0.5544484
```





The accuracy is 85.05%, which is highest as compared to other models.

```
> accuracy(predicted_table)
[1] 85
> mean(predTest == x_test$Exited)
[1] 0.85
```

INTERPRETATIONS

1. Hypothesis testing and descriptive statistics are used to establish the minimum credit score needed by the consumer to keep their account open. **We discovered that the minimum criterion for the customer was 500 credit or above**.
2. We estimate whether a customer will leave the bank's service based on their balance, estimated salary, and credit score. **We were able to attain a target variable accuracy of about 85% using the classification techniques.**
3. We can tell whether a consumer has a credit card by looking at their balance, location, and estimated salary. To do this, we utilise classification and clustering techniques to assess the likelihood. **Our accuracy in identifying the target variables was about 70%.**

Techniques	Accuracy
Logistic Modelling	78.93%
KNN	79.90%
Naïve Bayes	81.63%
Decision Tree	83.36%
Random Forest	85.00%

CONCLUSION

We conclude that our churn modelling can be classified and clustered using various different algorithms and clustering techniques. However, since the dataset is huge and to avoid the overfitting of data, **Random Forest would be the best option to predict the target variables and determine the customer's outcome,** also it fitted the data with minimum error.

THANK YOU