ALY 6015: INTERMEDIATE ANALYTICS

Final Project: CHURN MODELING

Submitted to

Prof. Fatemeh Ahmadi Abkenari

Submitted by

Abhilash Dikshit

Mrityunjay Gupta

Siddharth Alashi

Smit Parmar

Final Project: Churn Modeling

Abhilash Dikshit, Siddharth Alashi ,Mrityunjay Gupta, Smit Parmar
College of Professional Studies
Northeastern University
Vancouver, Canada

I. Abstract:

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. The paper describes the finding of various methodologies, Model predictions, Regression Methods, and Classification methods to check the accuracy of the models and best fit model to solve the questions of "Banking Customers." The methods used are : ANOVA TEST, Logistic Regression, Hypothesis Testing, LASSO MODEL, KNN model, NAÏVE BAYES, Decision Tree, and Random Forest. At the end the model accuracies are showcased and categorized.

II. Introduction

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.

I.a Dataset Description

We have 10000 rows and 14 columns, and we are considering 10 columns for our data analysis i.e., Credit Score, Gender, Age, Tenure, Balance, Number of Products, Has Credit Card, is active member, Estimated Salary, exited; Data cleanup will be performed to remove the NA values if any

The following dataset has 10,000 samples. The dataset and it's head(), tail() function values are shown below. We have also shown a summary of the dataset to get a better

	RowNumber <int>	CustomerId <int>	Surname <chr>	CreditScore <int>	Geography <chr>	Gender <chr>	Age <int>	Tenure <int>	Balance <dbl>	NumOfProducts <int>	HasCrCard <int>	IsActiveMember <int>	EstimatedSalary <dbl>	Exited <int>
1	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
2	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
9998	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9999	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
10000	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

6 rows

understanding of the variables in our dataset. Dataset has been taken from Kaggle.

```
'data.frame':   10000 obs. of  14 variables:
 $ RowNumber      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ CustomerId     : int  15634602 15647311 15619304 15701354 15737888 15574012 15592531 15656148 15792365 15592389 ...
 $ Surname        : chr  "Hargrave" "Hill" "Onio" "Boni" ...
 $ CreditScore    : int  619 608 502 699 850 645 822 376 501 684 ...
 $ Geography      : chr  "France" "Spain" "France" "France" ...
 $ Gender         : chr  "Female" "Female" "Female" "Female" ...
 $ Age            : int  42 41 42 39 43 44 50 29 44 27 ...
 $ Tenure         : int  2 1 8 1 2 8 7 4 4 2 ...
 $ Balance        : num  0 83808 159661 0 125511 ...
 $ NumOfProducts  : int  1 1 3 2 1 2 2 4 2 1 ...
 $ HasCrCard      : int  1 0 1 0 1 1 1 1 1 1 ...
 $ IsActiveMember : int  1 1 0 0 1 0 1 0 1 1 ...
 $ EstimatedSalary: num  101349 112543 113932 93827 79084 ...
 $ Exited         : int  1 0 1 0 0 1 0 1 0 0 ...
```

```
   RowNumber       CustomerId         Surname           CreditScore
 Min.   :    1   Min.   :15565701   Length:10000       Min.   :350.0
 1st Qu.: 2501   1st Qu.:15628528   Class :character   1st Qu.:584.0
 Median : 5000   Median :15690738   Mode  :character   Median :652.0
 Mean   : 5000   Mean   :15690941                      Mean   :650.5
 3rd Qu.: 7500   3rd Qu.:15753234                      3rd Qu.:718.0
 Max.   :10000   Max.   :15815690                      Max.   :850.0
  Geography            Gender                Age             Tenure
 Length:10000       Length:10000       Min.   :18.00   Min.   : 0.000
 Class :character   Class :character   1st Qu.:32.00   1st Qu.: 3.000
 Mode  :character   Mode  :character   Median :37.00   Median : 5.000
                                       Mean   :38.92   Mean   : 5.013
                                       3rd Qu.:44.00   3rd Qu.: 7.000
                                       Max.   :92.00   Max.   :10.000
    Balance       NumOfProducts    HasCrCard      IsActiveMember
 Min.   :     0   Min.   :1.00   Min.   :0.0000   Min.   :0.0000
 1st Qu.:     0   1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:0.0000
 Median : 97199   Median :1.00   Median :1.0000   Median :1.0000
 Mean   : 76486   Mean   :1.53   Mean   :0.7055   Mean   :0.5151
 3rd Qu.:127644   3rd Qu.:2.00   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :250898   Max.   :4.00   Max.   :1.0000   Max.   :1.0000
 EstimatedSalary        Exited
 Min.   :    11.58   Min.   :0.0000
 1st Qu.: 51002.11   1st Qu.:0.0000
 Median :100193.91   Median :0.0000
 Mean   :100090.24   Mean   :0.2037
 3rd Qu.:149388.25   3rd Qu.:0.0000
 Max.   :199992.48   Max.   :1.0000
```

I.b. Variable Description

The following are the variables in our dataset.

CustomerId	Holds an individual customer identification number.
Surname	Has the Last Name of the Customer.
CreditScore	Provides the credit score of Customer.
Geography	Location of Customer
Gender	To identify the gender of Customer.
Age	Provides Age
Tenure	No of times Visited.
Balance	Balance of the Customer in Company
NumOfProducts	Counts the total products used.
HasCrCard	To know if the customer has credit card or not. (1=Yes, 0=No)
IsActiveMember	To identify if the customer is an active customer or not. (1=Yes, 0=No)
EstimatedSalary	Approximate Salary of Customer
Exited	If the customer is retained or has left (1 = Left, 0 = Retained)

III. Exploratory Data Analysis & Visualizations

The *fig***.** below illustrates that there are 10000 entries in the dataset and the there are total 14 variables the dataset has surveyed, there are 0 missing values / entries and therefore, the dataset doesn't requires Cleaning procedure to get normal data.

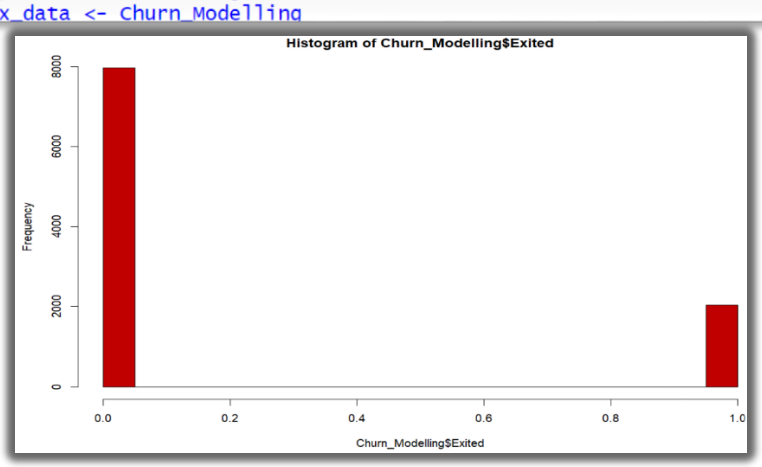
```
> Churn_Modelling <- read.csv("Churn_Modelling-3.csv", header = T)
> cat("Number of Rows before cleanup:", nrow(Churn_Modelling), "\n") #
Printing string and variable row count on the same line
Number of Rows before cleanup: 10000
> cat("Number of Columns before cleanup:", ncol(Churn_Modelling),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

The Fig. besides draws the following Conclusion

For converting the categorical variable “Gender” having attributes “Female and Male” into Numerical / Binomial variable we have counted 0 as Male and 1 as Females.

The histogram for the following is shown here.
Count of Male: 2037
Count of Female: 7693

```
> Churn_Modelling[,c('RowNumber', 'CustomerId', 'Surname', 'Geograph
y')] <- NULL
> Churn_Modelling$Gender[Churn_Modelling$Gender=='Male'] <- 1
> Churn_Modelling$Gender[Churn_Modelling$Gender=='Female'] <- 0
> count(Churn_Modelling$Exited)
> hist(Churn_Modelling$Exited, col = "#c00000",xlim=c(0,1))
> y <- Churn_Modelling$Exited
> x_data <- Churn_Modelling
```




```
> 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)
> cat("Number of Rows in train dataset:", nrow(x_train), "\n") # Printi
ng string and variable row count on the same line
Number of Rows in train dataset: 7000
> cat("Number of Rows in Test dataset:", nrow(x_test), "\n")
Number of Rows in Test dataset: 3000
```

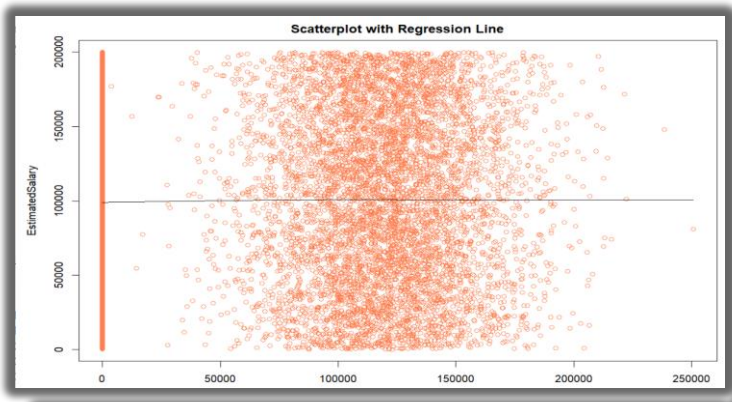
We have differentiated the whole Main dataset into “Train and Test dataset”. The Train dataset holds 70% of total volume of the Main data, whereas the Test Dataset holds the rest 30%

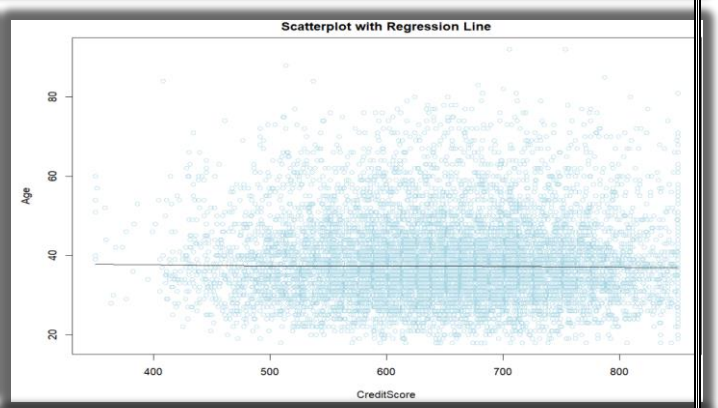
	CreditScore <dbl>	Gender <dbl>	Age <dbl>	Tenure <dbl>	Balance <dbl>	NumOfProducts <dbl>	HasCrCard <dbl>	IsActiveMember <dbl>	EstimatedSalary <dbl>	Exited <dbl>
1	619	0	42	2	0.00	1	1	1	101348.88	1
10	684	1	27	2	134603.88	1	1	1	71725.73	0
11	528	1	31	6	102016.72	2	0	0	80181.12	0
12	497	1	24	3	0.00	2	1	0	76390.01	0
9995	800	0	29	2	0.00	2	0	0	167773.55	0

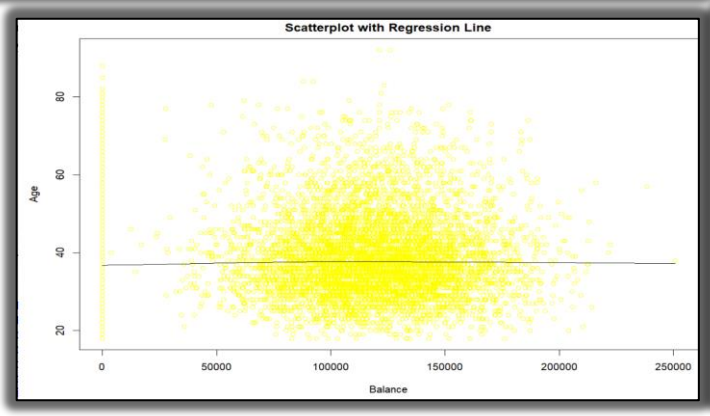
	CreditScore <dbl>	Gender <dbl>	Age <dbl>	Tenure <dbl>	Balance <dbl>	NumOfProducts <dbl>	HasCrCard <dbl>	IsActiveMember <dbl>	EstimatedSalary <dbl>	Exited <dbl>
1017	541	1	40	7	95710.11	2	1	0	49063.42	0
8004	603	1	57	6	105000.85	2	1	1	87412.24	1
4775	811	0	35	7	0.00	1	1	1	178.19	0
9725	485	1	41	2	100254.76	2	1	1	12706.67	0
3559	765	1	41	4	124182.21	1	0	0	100153.43	0
1672	594	1	41	2	122545.65	2	1	1	42050.24	0
9979	774	1	40	9	93017.47	2	1	0	191608.97	0
5842	676	0	49	1	0.00	1	1	0	79342.31	1

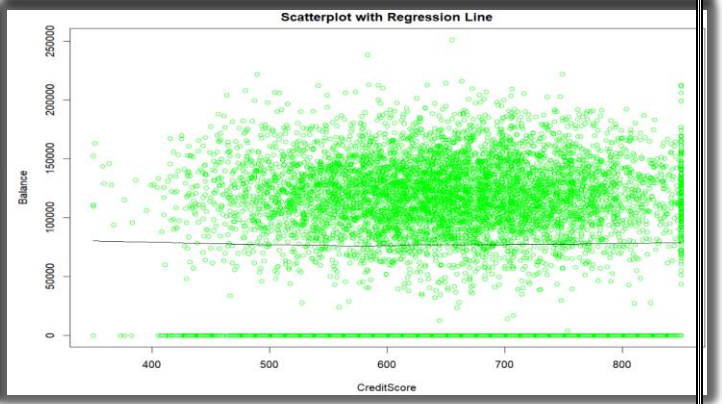
A scatter-plot is used to find the linear relationship between dependent and independent variable. We've concluded that there is very minimal co-relationship between the variables. The targeted variables chosen are EstimatedSalary, Age, Balance, CreditScore.

```
> 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)
> scatter.smooth(x = x_data$Balance, y = x_data$EstimatedSalary,  main
="Scatterplot with Regression Line", xlab="Balance", ylab="EstimatedSa
ary", col= "coral")
> scatter.smooth(x = x_data$CreditScore, y = x_data$Age,  main="Scatte
plot with Regression Line", xlab="CreditScore", ylab="Age", col= "ligh
blue")
> scatter.smooth(x = x_data$Balance, y = x_data$Age,  main="Scatterplo
with Regression Line", xlab="Balance", ylab="Age", col= "yellow")
> scatter.smooth(x = x_data$CreditScore, y = x_data$Balance,  main="Sc
tterplot with Regression Line", xlab="CreditScore", ylab="Balance", co
= "green")
```





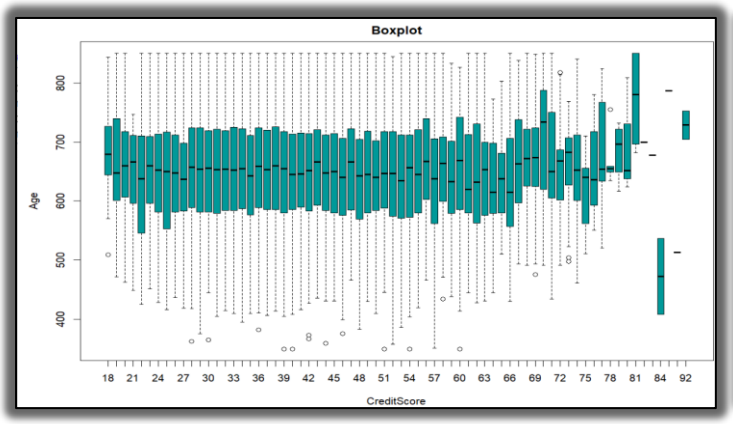


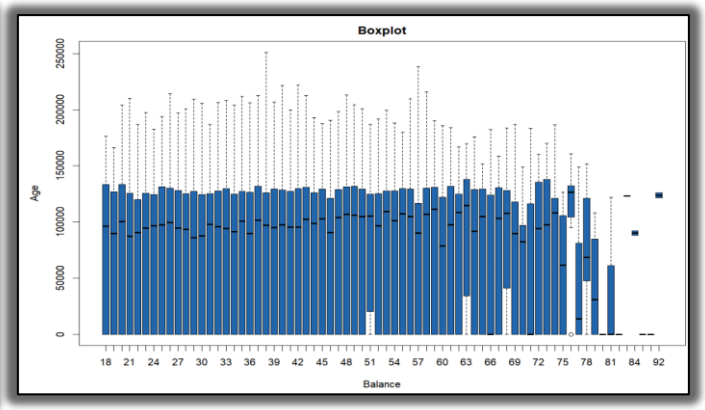


The main purpose of visualizing with ***Box plots*** are used to display the distributions of numerical data values, particularly when comparing them across various groups. They are designed to give high-level information at a glance and provide details like the symmetry, skew, variance, and outliers of a set of data.

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

We can conclude from Below Box plots that Notches in the boxplots do not coincide, we assume that their true medians differ. We've visualized as Age v/s CreditScore and the 2nd Box plot depicts Age v/s Balance.





IV. Model Implementation and Accuracies

1. 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 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.

```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 < 2.2e-16
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 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.

```r
# 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.
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  3.258804
sample estimates:
mean of x mean of y
 650.2769  650.8314
```

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 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.

```
> 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 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..

```
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$Exite
d), sigma.y = sd(inactive_cust$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
```

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 < 2e-16 ***
## 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
```

LOGISTIC REGRESSION

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
```

LASSO MODEL

Lasso is a linear model regularization method that is commonly used for feature selection. By applying the L1 penalty, it shrinks the coefficients of less important features towards zero, effectively removing them from the model. The amount of shrinkage is controlled by a hyperparameter called lambda.

In the code provided, a LASSO model is generated to reduce the number of features. The glmnet() function is used to fit a LASSO model with alpha=1 (L1 penalty) and 63 different lambda values. The coef() function is then used to obtain the coefficients for lambda value number 50. This is followed by plotting the LASSO coefficients as a function of lambda value using the plot() function.

```
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))
```

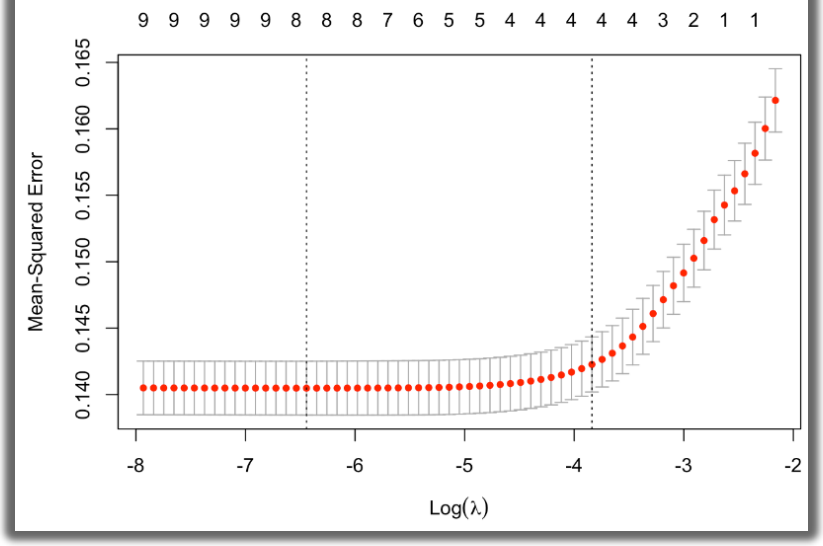
```
## [1] 9
```

```
print(ncol(test_x))
```

```
## [1] 9
```

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.**

```
Lasso1 = cv.glmnet(train_x, train_y, nfolds =10)
plot(Lasso1)
```



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.

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.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
##
##   Df  %Dev   Lambda
## 1  8 13.58 0.001591
```

```
coef(model1.min)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                               s0
## (Intercept)     -0.11414051582187
## CreditScore     -0.00007667833431
## Gender          -0.07429133452757
## Age              0.01115601330160
## Tenure          -0.00131222069497
## Balance          0.00000067560230
## NumOfProducts   -0.00221265297806
## HasCrCard        .
## IsActiveMember  -0.13987634949949
## EstimatedSalary  0.00000004386651
```

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'.

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

```
pre.fit= predict(fit1, new=x_data)
rmse(x_data$Exited, pre.fit)
```

```
## [1] 0.3743791
```

```
pre.model1.1se <- predict(model1.1se, newx=train_x)
rmse(train_y, pre.model1.1se)
```

```
## [1] 0.3786023
```

```
pre.test.model1.1se <- predict(model1.1se, newx=test_x)
rmse(test_y, pre.test.model1.1se)
```

```
## [1] 0.3786023
```

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.**

#Regularization Techniques

```
model_x <- model.matrix(x$Exited~.,x)[,-1]
Exited <- (x$Exited)
```

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)     -0.10251901594  0.03386679775  -3.027              0.00248 **
## CreditScore     -0.00009264233  0.00003877484  -2.389              0.01690 *
## Gender          -0.07732506860  0.00753148737 -10.267 < 0.0000000000000002 ***
## Age              0.01130838673  0.00035893642  31.505 < 0.0000000000000002 ***
## Tenure          -0.00184858270  0.00129640148  -1.426              0.15392
## Balance          0.00000069380  0.00000006306  11.003 < 0.0000000000000002 ***
## NumOfProducts   -0.00426240279  0.00676604181  -0.630              0.52873
## HasCrCard       -0.00295820448  0.00822205158  -0.360              0.71901
## IsActiveMember  -0.14322491471  0.00753198849 -19.016 < 0.0000000000000002 ***
## EstimatedSalary  0.00000007118  0.00000006516   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: < 0.00000000000000022
```

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%

We have created a KNN CLASSIFIER and set k=13. To recap this means that if atleast 12 outoff 13 nearest points to a new data points are not EXITED CUSTOMERS.

```
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
```

NAÏVE BAYES

Naïve bayes model is trained on training set (X_Train) usig naïve bayes function. The response variable is Exited and all other variables is predicted. The Train model is used to predict the values of the exited for the test set (X_Test) using predicted functions.

The predicted test values and actual values of exited in the test set are used to create confusion Matrix (Tab_Naive) using Table Function.

The Accuracy of Naïve Bayes classified is calculated using the accuracy function which takes confusion matrix as input and return percent as correct prediction.

```
set.seed(1)
NBclassfier=naiveBayes(as.factor(Exited) ~ ., data=x_train)
print(NBclassfier)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##                      0 0.00000398568199542691
##              0.7992857              0.2007143
##
## Conditional probabilities:
##                          CreditScore
## Y                                [,1]         [,2]
##   0                       0.002598447 0.0003804919
##   0.00000398568199542691 0.002568175 0.0004015496
##
##                          Gender
## Y                                   [,1]           [,2]
##   0                       0.000002293103 0.000001970267
##   0.00000398568199542691 0.000001835399 0.000001987319
##
##                          Age
## Y                                 [,1]          [,2]
##   0                       0.0001482246 0.00003923128
##   0.00000398568199542691 0.0001781557 0.00003780397
##
##                          Tenure
```

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.

```
NB_Predictions <- predict(NBclassfier, x_test)
tab_naive <- table(NB_Predictions,x_test$Exited)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab_naive)
```

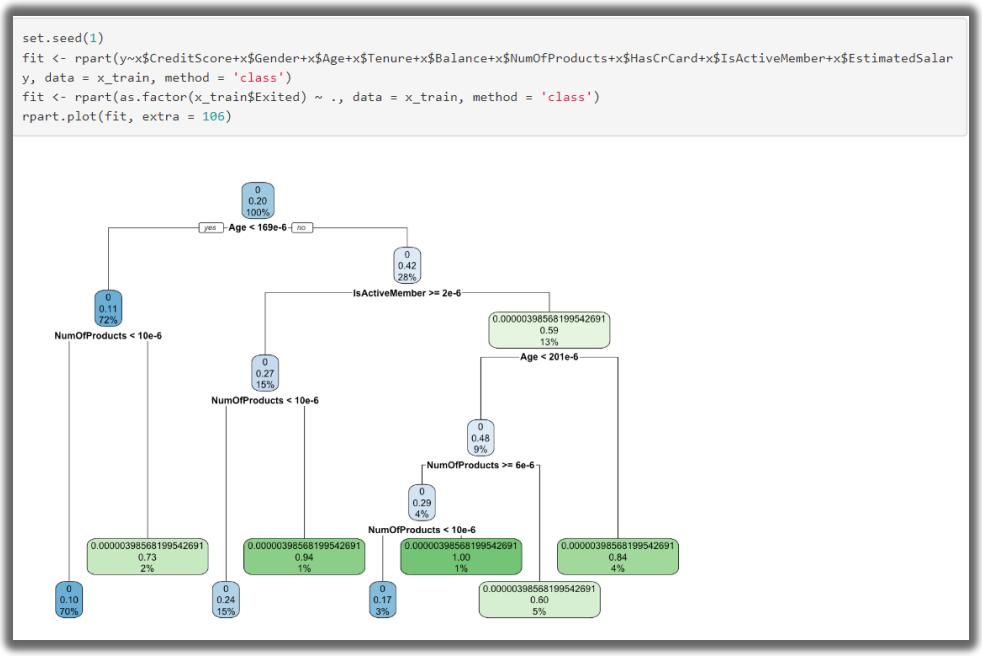
```
## [1] 81.63333
```

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%.**

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 = 'class')
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 0.00000398568199542691
##   0                      2275                      93
##   0.00000398568199542691  363                     269
```

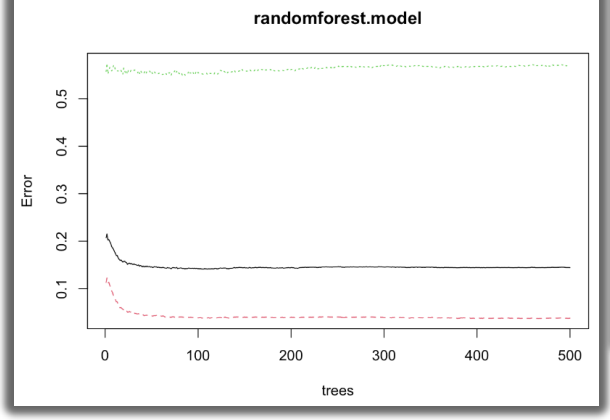
```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(table_mat)
```

```
## [1] 84.8
```

RANDOM FOREST

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

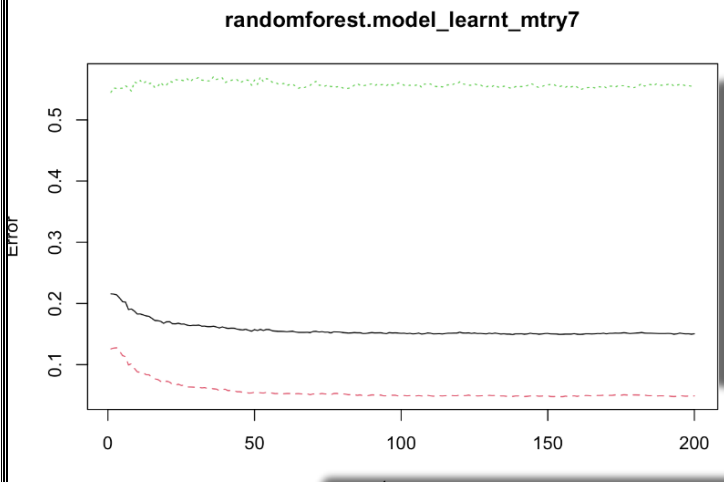
```
plot(randomforest.model)
```




```
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 0.00000398568199542691 class.error
## 0                     5382                   213  0.03806971
## 0.00000398568199542691  798                   607  0.56797153
```

```
plot(randomforest.model)
```




```
randomforest.model_learnt_mtry2 <- randomForest(as.factor(x_train$Exited) ~ ., data = x_train, ntree = 200, mtry = 2, import
ance = 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 0.00000398568199542691 class.error
## 0                     5433                   162  0.02895442
## 0.00000398568199542691  852                   553  0.60640569
```

```
plot(randomforest.model_learnt_mtry2)
```

```
predTest <- predict(randomforest.model_learnt_mtry2,x_test)
predicted_table <- table(observed=x_test$Exited,predicted=predTest)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(predicted_table)
```

```
## [1] 85
```

```
mean(predTest == x_test$Exited)
```

```
## [1] 0.85
```

V.CONCLUSION

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%.**

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.

VI. References

Bevans (2022, November 11);Datanovia (2019, December 26);*Linear Regression Example in r Using Lm() Function* (n.d.);Zach (2021, September 29);John (2023, January 25);Rithika (2022, December 29)

Bevans, R. 2022, November 11. *Hypothesis Testing | a Step-by-Step Guide with Easy Examples*. https://www.scribbr.com/statistics/hypothesis-testing/.

Datanovia. 2019, December 26. *How to Do a t-Test in r: Calculation and Reporting*. https://www.datanovia.com/en/lessons/how-to-do-a-t-test-in-r-calculation-and-reporting/.

John, B. 2023, January 25. *How to Implement Customer Churn Prediction [Machine Learning Guide for Programmers].* https://neptune.ai/blog/how-to-implement-customer-churn-prediction.

Linear Regression Example in r Using Lm() Function. n.d. https://www.learnbymarketing.com/tutorials/linear-regression-in-r/.

Rithika, S. 2022, December 29. *Building a Churn Prediction Model on Retail Data Simplified: The Ultimate Guide 101. Learn | Hevo.* https://hevodata.com/learn/churn-prediction-model/.

Zach, Z. 2021, September 29. *How to Perform Logistic Regression in r (Step-by-Step)*. https://www.statology.org/logistic-regression-in-r/.

VII. APPENDIX

```
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("/Users/abidikshit/R_Projects/Data/Churn_Model.
csv", header = T)

cat("Number of Rows before cleanup:", nrow(Churn_Modelling), "\n") # Printi
ng string and variable row count on the same line

cat("Number of Columns before cleanup:", ncol(Churn_Modelling), "\n")

cat("Blank cells count before cleanup:", sum(!complete.cases(Churn_Modellin
g))) # Displaying Blank Cells Count for uncleaned data


head(Churn_Modelling)


Churn_Model <- Churn_Modelling  #(Used for Hypothesis Testing)


Churn_Modelling[,c('RowNumber', 'CustomerId', 'Surname', 'Geography')] <- N
ULL

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


headTail(x_data, top = 4, bottom = 4, ellipsis = F)


sapply(x_data,class)

x_data$Gender <- as.numeric(x_data$Gender)

x <- (x_data - min(x_data))/(max(x_data)-min(x_data))

headTail(x, top = 4, bottom = 4, ellipsis = F)

summary(x)


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)


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)


summary(x_data)


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")


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="CreditScore", ylab="Age", col="#009999")

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


set.seed(1)

onesample <- t.test(Churn_Model$CreditScore, mu = 500, alternative = "greater")

onesample
```

```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


pairedtest <- t.test(male_cust$Exited, female_cust$Exited, alternative = "g
reater")

pairedtest


ftest <- var.test(male_cust$EstimatedSalary, female_cust$EstimatedSalary, d
ata = Churn_Model, alternative = "less")

ftest


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 = "gr
eater", mu = 0 , sigma.x = sd(active_cust$Exited), sigma.y = sd(inactive_cu
st$Exited))

ztest


res.aov <- aov(y ~ x$CreditScore+x$Balance+x$EstimatedSalary, data = x_trai
n)

summary(res.aov)


set.seed(1)

options(scipen = 100)

linear.model <- lm(y ~ x$CreditScore + x$Balance + x$EstimatedSalary, data
= x_train)

summary(linear.model)

prediction.lm <- predict(linear.model, newdata=x_test)

prediction.lm

prediction.lm <- ifelse(prediction.lm>0.5,1,0)

linear.prediction.lm <- mean(prediction.lm!=x_test$Exited)

print(paste('Accuracy',1-linear.prediction.lm))


set.seed(1)
```

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


log_pred <- predict(mylogit, newdata = x_test)
tab_log <- table(x_test$Exited,log_pred)
#tab_log
log_pred <- ifelse(log_pred>0.5,1,0)
log_pred_res <- mean(log_pred!= x_test$Exited)
print(paste('Accuracy', 1-log_pred_res))


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)
tab <- table(knn_model,x.test.target)
tab
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)


set.seed(1)
NBclassfier=naiveBayes(as.factor(Exited) ~ ., data=x_train)
print(NBclassfier)
NB_Predictions <- predict(NBclassfier, x_test)
tab_naive <- table(NB_Predictions,x_test$Exited)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab_naive)


set.seed(1)
fit <- rpart(y~x$CreditScore+x$Gender+x$Age+x$Tenure+x$Balance+x$NumOfProdu
cts+x$HasCrCard+x$IsActiveMember+x$EstimatedSalary, data = x_train, method
= 'class')
fit <- rpart(as.factor(x_train$Exited) ~ ., data = x_train, method = 'class
')
rpart.plot(fit, extra = 106)
predict_unseen <-predict(fit, x_test, type = 'class')
table_mat <- table(x_test$Exited, predict_unseen)
```

```
table_mat
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(table_mat)
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
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(table_mat)
mean(predict_unseen == x_test$Exited)


set.seed(1)
randomforest.model <- randomForest(as.factor(x_train$Exited) ~ ., data = x_
train, importance = TRUE)
randomforest.model
plot(randomforest.model)
randomforest.model_learnt_mtry2 <- randomForest(as.factor(x_train$Exited) ~
., data = x_train, ntree = 200, mtry = 2, importance = TRUE)
print(randomforest.model_learnt_mtry2)
plot(randomforest.model_learnt_mtry2)
randomforest.model_learnt_mtry4 <- randomForest(as.factor(x_train$Exited) ~
., data = x_train, ntree = 200, mtry = 4, importance = TRUE)
randomforest.model_learnt_mtry4
randomforest.model_learnt_mtry7 <- randomForest(as.factor(x_train$Exited) ~
., data = x_train, ntree = 200, mtry = 7, importance = TRUE)
randomforest.model_learnt_mtry7
plot(randomforest.model_learnt_mtry7)
predTest <- predict(randomforest.model_learnt_mtry2,x_test)
predicted_table <- table(observed=x_test$Exited,predicted=predTest)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(predicted_table)
mean(predTest == x_test$Exited)
importance(randomforest.model_learnt_mtry2, col= "coral")
varImpPlot(randomforest.model_learnt_mtry2, col= "coral")


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))
print(ncol(test_x))

Lasso1 = cv.glmnet(train_x, train_y, nfolds =10)
plot(Lasso1)
print(log(Lasso1$lambda.min))
print(log(Lasso1$lambda.1se))

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

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

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

pre.fit= predict(fit1, new=x_data)
rmse(x_data$Exited, pre.fit)

pre.model1.1se <- predict(model1.1se, newx=train_x)
rmse(train_y, pre.model1.1se)

pre.test.model1.1se <- predict(model1.1se, newx=test_x)
rmse(test_y, pre.test.model1.1se)

model_x <- model.matrix(x$Exited~.,x)[,-1]
Exited <- (x$Exited)

ridge_fit <- glmnet(model_x,Exited,alpha=0)
plot(ridge_fit,xvar="lambda",label=TRUE)
```

```
ridge_cv <- cv.glmnet(model_x,Exited,alpha=0)
plot(ridge_cv)

set.seed(141197)
train=sample(1:nrow(model_x),size=0.7*nrow(model_x))
test=(-train)
fit_training <- glmnet(model_x[train,],Exited[train],alpha=1,lambda=100)
cv_training <- cv.glmnet(model_x[train,],Exited[train],alpha=1)
par(mfrow=c(1,1))
plot(cv_training)

cv_bestlam <- cv_training$lambda.min
predict(Lasso1,type ="coefficients",s=cv_bestlam)[1:10,]

## NA
```