This is the main submission document. ***Save and rename this document filename with your <u>registered full name</u> as Prefix before submission.***

Class / Seminar Grp	Seminar 4
Full Name	Ng Jun Long
Matriculation Number	U2110010D

** : Delete and replace as appropriate.*

Declaration of Academic Integrity

By submitting this assignment for assessment, I declare that this submission is my own work, unless otherwise quoted, cited, referenced or credited. I have read and understood the Instructions to CBA.PDF provided and the Academic Integrity Policy.

I am aware that failure to act in accordance with the University's Academic Integrity Policy may lead to the imposition of penalties which may include the requirement to revise and resubmit an assignment, receiving a lower grade, or receiving an F grade for the assignment; suspension from the University or termination of my candidature.

I consent to the University copying and distributing any or all of my work in any form and using third parties to verify whether my work contains plagiarised material, and for quality assurance purposes.

Please insert an "X" within the square brackets below to indicate your selection.

[X] **I have read and accept the above.**

Table of Contents

For each question, please start your answer in a new page.

Answer to Q1:

Answer to Q1a):

```r
## Import packages
library(data.table)
library(rpart)
library(rpart.plot)
library(stringr)
library(ggplot2)
library(caret)
library(skimr)
library(dplyr)
library(tidyr)
library(tidyverse)
library(class)
library(RANN)
library(VIM)
library(ggpubr)
library(MLmetrics)
library(purrr)
library(ggpubr)
library(mice)
library(caTools)
library(ROSE)
```

We start by importing all the relevant packages into our environment

```r
## Import Data
setwd("/Users/junlongng/Desktop/NTU/Year 2/Semester 1/BC2406 Analytics 1/AY22 BC2406 CBA")
df1 = fread("/Users/junlongng/Desktop/NTU/Year 2/Semester 1/BC2406 Analytics 1/AY22 BC2406 CBA/homeloan2.csv", na.strings = c("NA", "missing", "N/A", "", "m", "M", "na", ".",NA))
dt1 = data.table(df1)
dim(dt1)
summary(dt1)
```

We then import the data and set na.strings = c(….) this allows us to catch all potential values of NA or missing data as NA such that it is easier for analysis later on.

We create a data table using dt1 = data.table(df1)

```r
sapply(dt1, class)
```

```
> sapply(dt1, class)
          Loan_ID            Gender           Married        Dependents 
      "character"       "character"       "character"       "character" 
        Education     Self_Employed   ApplicantIncome CoapplicantIncome 
      "character"       "character"         "integer"         "numeric" 
       LoanAmount  Loan_Amount_Term      Credit_Score     Property_Area 
        "integer"         "integer"         "integer"       "character" 
      Loan_Status 
      "character" 
> 
```

This shows us the class of each data. Most of the data is currently stored as a character and we have to recast them into the appropriate data type for further analysis.

```
factorcolumns = c("Gender","Married","Self_Employed","Credit_Score","Loan_Status"
,"Education","Property_Area")

## For Dependents, it's abit strange since there is 3+. Shall make it into ordered
factor
dt1$Dependents = factor(df1$Dependents, ordered = T, levels = c("0","1","2","3+"
,""))
```

From these two lines of codes, I've decided to make Gender, Married, Self_Employed, Credit_Score, Loan_Status, Education and property area into a factor data type

I've also made dependents into an ordered factor type.

```
dt1[, (factorcolumns):= lapply(.SD, factor), .SDcols = factorcolumns]
sapply(dt1, class)
```

I then recast them to dt1 using data table functions.

```
## Changing Applicant Income to numeric for cents value
dt1$ApplicantIncome = as.numeric(dt1$ApplicantIncome)
summary(dt1)
```

I also changed Applicant Income to a numeric value to allow for decimals.

```
> sapply(dt1, class)
$Loan_ID
[1] "character"

$Gender
[1] "factor"

$Married
[1] "factor"

$Dependents
[1] "ordered" "factor"

$Education
[1] "factor"

$Self_Employed
[1] "factor"

$ApplicantIncome
[1] "numeric"

$CoapplicantIncome
[1] "numeric"

$LoanAmount
[1] "integer"

$Loan_Amount_Term
[1] "integer"

$Credit_Score
[1] "factor"

$Property_Area
[1] "factor"

$Loan_Status
[1] "factor"
```

These are the new data types of each column.

Answer to Q1b):

Based on my observation, will be using Mode to replace all categorical variables with NA values

As such, we have to develop a mode function in R due to a lack of it in the base R package.

```
calc_mode = function(x){

  # List the distinct / unique values
  distinct_values = unique(x)

  # Count the occurrence of each distinct value
  distinct_tabulate= tabulate(match(x, distinct_values))

  # Return the value with the highest occurrence
  distinct_values[which.max(distinct_tabulate)]
}
```

```r
## Replacing NA values in categorical variables with mode
dt1$Gender[is.na(dt1$Gender)] = calc_mode(dt1$Gender)
dt1$Dependents[is.na(dt1$Dependents)] = calc_mode(dt1$Dependents)
dt1$Married[is.na(dt1$Married)] = calc_mode(dt1$Married)
dt1$Self_Employed[is.na(dt1$Self_Employed)] = calc_mode(dt1$Self_Employed)
dt1$Credit_Score[is.na(dt1$Credit_Score)] = calc_mode(dt1$Credit_Score)
```

This replaces all Na values in each of the categorical data columns

I've decided to use median to replace NA values in the continuous data columns.

```r
## Replacing NA values in continuous variables with median
dt1$LoanAmount[is.na(dt1$LoanAmount)] = median(dt1$LoanAmount, na.rm = TRUE)
dt1$Loan_Amount_Term[is.na(dt1$Loan_Amount_Term)] = median(dt1$Loan_Amount_Term, na.rm = TRUE)
```

```
> sum(is.na(dt1))
[1] 0
>
```

```
> str(dt1)
Classes ‘data.table’ and 'data.frame':  592 obs. of  12 variables:
 $ Gender           : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Married          : num  1 1 1 0 1 1 1 1 1 1 ...
 $ Dependents       : Ord.factor w/ 5 levels "0"<"1"<"2"<"3+"<..: 2 1 1 1 3 1 4 3 2 3 ...
 $ Education        : num  1 1 0 1 1 0 1 1 1 1 ...
 $ Self_Employed    : num  0 1 0 0 1 0 0 0 0 0 ...
 $ ApplicantIncome  : num  4583 3000 2583 6000 5417 ...
 $ CoapplicantIncome: num  1508 0 2358 0 4196 ...
 $ LoanAmount       : num  128 66 120 141 267 95 158 168 349 70 ...
 $ Loan_Amount_Term : num  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_Score     : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 2 ...
 $ Property_Area    : Factor w/ 3 levels "A","B","C": 3 1 1 1 1 1 2 1 2 1 ...
 $ Loan_Status      : Factor w/ 2 levels "N","Y": 1 2 2 2 2 2 1 2 1 2 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

From here we can see the full list of data and their relevant types. We can already see that there are some outliers in Loan_Amount and Coapplicant Income

Answer to Q2):

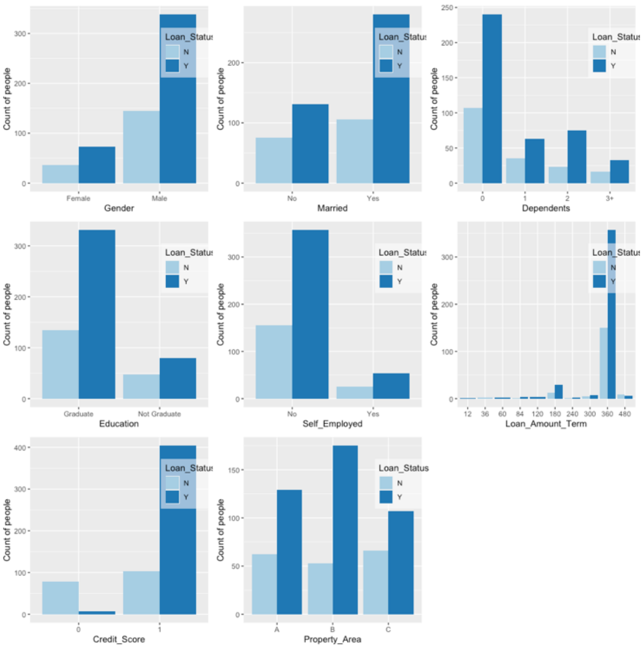
[Start your answers here…]

First step was to remove LoanID which will be explained more later.

```r
## Loan ID is not needed for analysis
dt1 = dt1[,c(2:13)]
```

```r
## Plotting graphs For Categorical Data
factor = names(keep(dt1,is.factor))
gglist = list()
for(graph in 1:(length(factor)-1)){
  gglist[graph]<-list(ggplot(data=dt1,aes_string(factor[graph],fill='Loan_Status'))+ geom_bar(position = 'dodge')+ ylab("Count of people") + theme(legend
.position = c(.9,.75),legend.background=element_rect(fill = alpha("white", 0.5)))+ scale_fill_brewer(palette = 'Paired'))
}
ggarrange(plotlist=gglist)
```

Using a for loop to plot categorical data.



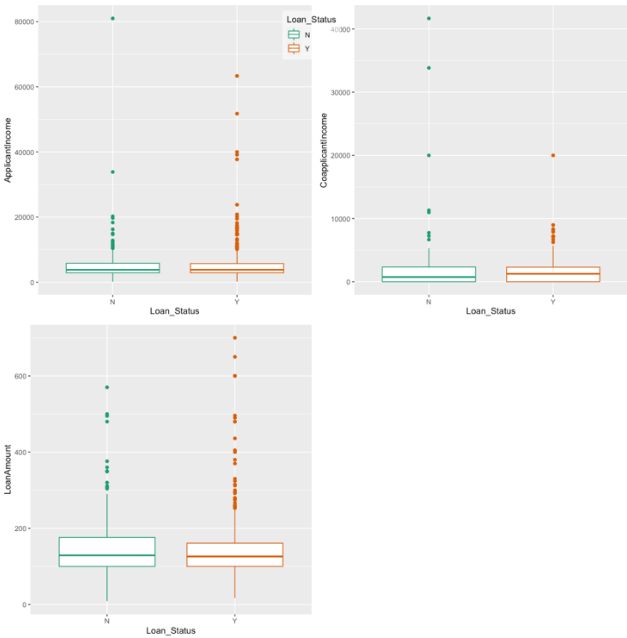
The Categorical graph tells us the following points

1) Gender V Loan Status: On average there are about a lot more males than there are females in our dataset. As a result, more males do get their loans approved. But whether or not there is a gender bias, we do not know yet
2) Married V Loan Status: We can also see that there are about twice more married applicants. Applicants that were married are also more likely to have their loans approved
3) Dependents V Loan Status: Most of our applicants do not have any dependents. Applicants with 2 applicants have a slightly higher chance of their loan being approved in terms of ratio.
4) Education V Loan Status: Applicants with “Graduated” as education seems to be correlated to a loan approval.

5) Self_Employed V Loan Status: We have more employed people applying for loans as compared to people who are self-employed. Based on the preliminary analysis, there seems to be no bias for employment status yet.
6) Property Area V Loan_Status: People in property area B have the highest amount of approvals for their loans.
7) Credit_Score V Loan Status: Most Applicants have a credit score of 1. There seems to be a strong relation with having a loan and credit score of 1.
8) Loan_Amount_Term V Loan Status: Seems that most of the loans are for 360 months.

Plotting Continuous Graphs

```
## Plotting graph for continuous variables
continuous_list = names(keep(dt1,is.numeric))
continuous_list
cont_plot1 = list()
for (i in 1:(length(continuous_list))) {
  ## Creating Box Plots using a for loop
  cont_plot1[i] = list(ggplot(data = dt1, aes_string(x ="Loan_Status", y = continuous_list[i], color = "Loan_Status")) + geom_boxplot() + theme
(legend.position = c(1,2), legend.background = element_rect(fill = alpha("white", 0.5))) + scale_color_brewer(palette = "Dark2"))
}
ggarrange(plotlist = cont_plot1)
```

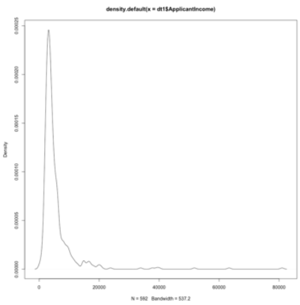


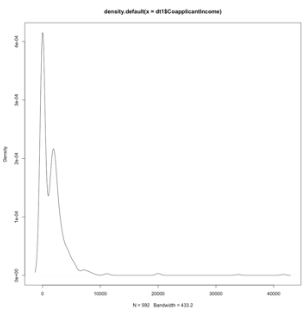
From the continuous table graph, we can see these points:

1) Applicant Income V Loan Status: Of those that were approved, there were more outliers in this column of data.
2) CoApplicant Income V Loan Status: There is a lower median for this column of data. Could suggest that coapplicant income is not significant. Need to investigate more.
3) Loan Amount V Loan Status: This reveals that the median for loan amount being approved or not is quite close. Suggests that perhaps loan amount has no effect on loan status.

We need to take a closer look into income for applicant and co-applicant as it seems like there are various outliers

```
## Both incomes seem abit strange
plot(density(dt1$ApplicantIncome))
plot(density(dt1$CoapplicantIncome))
plot(density(dt1$LoanAmount))
```





This shows us that the two data are extremely left skewed and needs to be handled.

```
copydt = dt1
copydt$TotalIncome = copydt$ApplicantIncome + copydt$CoapplicantIncome
copydt$TotalIncome = log(copydt$TotalIncome)
plot(density(copydt$TotalIncome))

copydt$LoanRatio = copydt$LoanAmount/copydt$Loan_Amount_Term
copydt$LoanRatio = log(copydt$LoanRatio)
plot(density(copydt$LoanRatio))
```

Decided to experiment with combining both applicants' income together as well as creating a ratio of loan amount divided by loan amount terms to balance out the datasets.

```r
## Need to remove the previous incomes and the loan
copydt = copydt[ ,c("ApplicantIncome","CoapplicantIncome","LoanAmount", "Loan_Amount_Term"):=NULL]

summary(copydt)
```

Removing the previous variables if not there will be a multi-collinearity effect and the effect of combining It would bring no extra value to the analysis model. I also created dummy variables by transforming the following variables to binary split for further analysis.

```r
## Need to create dummy variables for log model later on
dt1$Gender = ifelse(dt1$Gender == "Male", 1,0)
dt1$Gender
dt1$Married = ifelse(dt1$Married == "Yes",1,0)
dt1$Married
dt1$Education = ifelse(dt1$Education == "Graduate",1,0)
dt1$Education
dt1$Self_Employed = ifelse(dt1$Self_Employed == "Yes",1,0)
dt1$Self_Employed

copydt = dt1
```

These are the final variables that I will be analysing later on.

```
    Gender          Married        Dependents   Education        Self_Employed    Credit_Score Property_Area Loan_Status  TotalIncome
 Min.   :0.0000   Min.   :0.000   0 :339      Min.   :0.0000   Min.   :0.0000   0: 93        A:191         N:181       Min.   : 7.274
 1st Qu.:1.0000   1st Qu.:0.000   1 :104      1st Qu.:1.0000   1st Qu.:0.0000   1:499        B:228         Y:411       1st Qu.: 8.335
 Median :1.0000   Median :1.000   2 :100      Median :1.0000   Median :0.0000                C:173                     Median : 8.595
 Mean   :0.8142   Mean   :0.652   3+: 49      Mean   :0.7855   Mean   :0.1334                                          Mean   : 8.672
 3rd Qu.:1.0000   3rd Qu.:1.000     :  0      3rd Qu.:1.0000   3rd Qu.:0.0000                                          3rd Qu.: 8.928
 Max.   :1.0000   Max.   :1.000               Max.   :1.0000   Max.   :1.0000                                          Max.   :11.302
   LoanRatio
 Min.   :-3.6889
 1st Qu.:-1.2611
 Median :-1.0147
 Mean   :-0.9457
 3rd Qu.:-0.6777
 Max.   : 2.2246
```

For my final dataset, I will have 592 rows of 10 columns.

```
> dim(copydt)
[1] 592  10
>
```

Answer to Q3:

Answer to Q3a):

No, Loan_ID should not be used as a predictor variable. This is because it is a unique identifier value that showcases each person's ID, and its function as a variable is to identify the person based on their existence. As a result, LoanID does not support any classification capabilities and should therefore not be a predictor X variable.

Answer to Q3b):

Setting the seed as 8 and creating a train test split

```
set.seed(8)
train_data = sample.split(Y=copydt$Loan_Status,SplitRatio = 0.7)
trainset = subset(copydt,train_data==T)
testset = subset(copydt, train_data==F)
```

There seems to be an overwhelming amount of Yes, this might cause an imbalance in the trainset. Hence, we will need to perform an over sampling of No cases of under sampling of Yes cases.

```
> summary(trainset$Loan_Status)
  N   Y
127 288
>
```

Decided to oversample No values, this will create a larger sample size for greater analysis.

```
> ## Need to rebalance the trainset such that it is more "fair"
> ## Need to achieve 50% of each occurence. Set N to 2* amount of Y
> balancetrain_data = ovun.sample(Loan_Status~., data=trainset, seed = 8, method ="over", N = 576 )$data
> table(balancetrain_data$Loan_Status)

  Y   N
288 288
>
```

Performing a logistic regression on Categorical data “Loan_Status”

```
model1=glm(Loan_Status~.,data=balancetrain_data, family="binomial")
```

```
> summary(model1)

Call:
glm(formula = Loan_Status ~ ., family = "binomial", data = balancetrain_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0814  -0.8573  -0.1113   0.9404   2.2659

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     4.25239    2.09648   2.028 0.042525 *
Gender          0.28407    0.30260   0.939 0.347849
Married        -0.82507    0.23865  -3.457 0.000546 ***
Dependents.L   -0.34386    0.28826  -1.193 0.232915
Dependents.Q    0.15911    0.29471   0.540 0.589263
Dependents.C    0.73228    0.29492   2.483 0.013029 *
Education       0.11818    0.26992   0.438 0.661499
Self_Employed   0.37772    0.31440   1.201 0.229605
Credit_Score1  -4.16648    0.45527  -9.152  < 2e-16 ***
Property_AreaB -0.53699    0.27034  -1.986 0.046994 *
Property_AreaC  0.75312    0.25429   2.962 0.003060 **
TotalIncome    -0.08533    0.22453  -0.380 0.703910
LoanRatio       0.08851    0.21582   0.410 0.681715
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 798.51  on 575  degrees of freedom
Residual deviance: 559.73  on 563  degrees of freedom
AIC: 585.73

Number of Fisher Scoring iterations: 6
```

From this, we can see that marriage, Credit_score, Property Area and dependents are significant

Using backwards elimination to figure out the best logistic regression model for predicting Loan Status

```
> model2=glm(Loan_Status~Married+Dependents+Credit_Score+Property_Area,data=balancetrain_data,family="binomial")
> summary(model2)

Call:
glm(formula = Loan_Status ~ Married + Dependents + Credit_Score +
    Property_Area, family = "binomial", data = balancetrain_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0296  -0.8625  -0.1474   0.9808   2.1877

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)      3.7381     0.4906   7.620 2.54e-14 ***
Married         -0.7716     0.2249  -3.430 0.000603 ***
Dependents.L    -0.3105     0.2809  -1.105 0.268998
Dependents.Q     0.1064     0.2898   0.367 0.713371
Dependents.C     0.7694     0.2919   2.636 0.008396 **
Credit_Score1   -4.0974     0.4496  -9.113  < 2e-16 ***
Property_AreaB  -0.5274     0.2644  -1.994 0.046113 *
Property_AreaC   0.7515     0.2514   2.990 0.002794 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 798.51  on 575  degrees of freedom
Residual deviance: 562.40  on 568  degrees of freedom
AIC: 578.4

Number of Fisher Scoring iterations: 5
```

```
> model3=glm(Loan_Status~Married+Credit_Score+Property_Area,data=balancetrain_data, family="binomial")
> summary(model3)

Call:
glm(formula = Loan_Status ~ Married + Credit_Score + Property_Area,
    family = "binomial", data = balancetrain_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0251  -0.7684  -0.2301   0.9719   1.8927

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)      3.8401     0.4782   8.030 9.76e-16 ***
Married         -0.8484     0.2086  -4.067 4.76e-05 ***
Credit_Score1   -4.0604     0.4409  -9.209  < 2e-16 ***
Property_AreaB  -0.5400     0.2614  -2.066  0.03882 *
Property_AreaC   0.7251     0.2448   2.962  0.00306 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 798.51  on 575  degrees of freedom
Residual deviance: 570.70  on 571  degrees of freedom
AIC: 580.7

Number of Fisher Scoring iterations: 5
```

```
> model4 =glm(Loan_Status~Credit_Score+Married,data=balancetrain_data, family="binomial")
> summary(model4)

Call:
glm(formula = Loan_Status ~ Credit_Score + Married, family = "binomial",
    data = balancetrain_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7218  -0.8068  -0.2918   1.2342   1.6005

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.6790     0.4457   8.254  < 2e-16 ***
Credit_Score1  -3.8115     0.4326  -8.811  < 2e-16 ***
Married        -0.8229     0.2017  -4.081 4.49e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 798.51  on 575  degrees of freedom
Residual deviance: 596.93  on 573  degrees of freedom
AIC: 602.93

Number of Fisher Scoring iterations: 5
```

Using AIC to find the best logistic regression to use to predict Loan Status.

```
> ## Checking for the AIC values
> model_list = c(AIC(model1), AIC(model2), AIC(model3),AIC(model4))
> AIC(model1)
[1] 585.7275
> AIC(model2)
[1] 578.3966
> AIC(model3)
[1] 580.704
> AIC(model4)
[1] 602.9299
> ## Lowed AIC is best model
> min(model_list)
[1] 578.3966
> levels(balancetrain_data$Loan_Status)
[1] "Y" "N"
>
```

From here, we can see that model 2 has the lowest AIC value which means that it is the most accurate in predicting loan status through logistic regression. We also use levels function to discover the base reference value for loan_status for further analysis.

```
> ## Accuracy of Training on trainset
> prob = predict(model2, type = 'response')
> classifier = ifelse(prob>0.5, "N", "Y")
> classifier=as.factor(classifier)
> confusionMatrix(classifier,balancetrain_data$Loan_Status,positive = 'Y')
Confusion Matrix and Statistics

          Reference
Prediction   Y   N
         Y 250 110
         N  38 178

               Accuracy : 0.7431
                 95% CI : (0.7053, 0.7783)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4861

 Mcnemar's Test P-Value : 5.342e-09

            Sensitivity : 0.8681
            Specificity : 0.6181
         Pos Pred Value : 0.6944
         Neg Pred Value : 0.8241
             Prevalence : 0.5000
         Detection Rate : 0.4340
   Detection Prevalence : 0.6250
      Balanced Accuracy : 0.7431

       'Positive' Class : Y
```

Using model 2, we are able to have a 74% accuracy in predicting Loan Status through logistic regression on the trainset data.

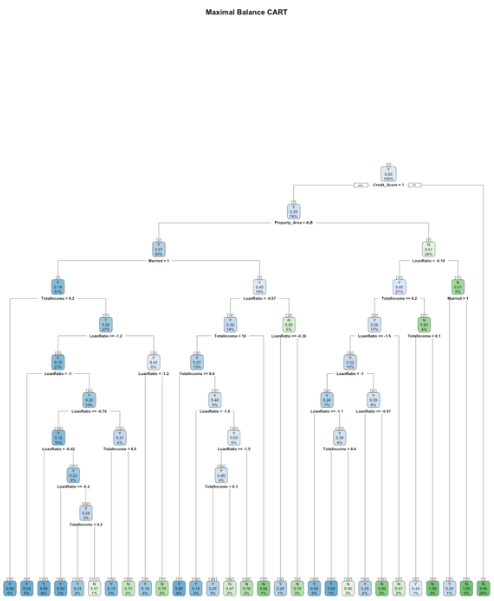
We will now use it on the testsetdata

```
> ## Testing out on the testset with log model trained on balanced data
> test_prob = predict(model4, newdata=testset, type="response")
> test_classifier = ifelse(test_prob>0.5,"N","Y")
> test_classifier = as.factor(test_classifier)
> confusionMatrix(test_classifier,testset$Loan_Status,positive = 'Y')
Confusion Matrix and Statistics

          Reference
Prediction   N   Y
         N  26   1
         Y  28 122

               Accuracy : 0.8362
                 95% CI : (0.7732, 0.8874)
    No Information Rate : 0.6949
    P-Value [Acc > NIR] : 1.227e-05

                  Kappa : 0.5506

 Mcnemar's Test P-Value : 1.379e-06

            Sensitivity : 0.9919
            Specificity : 0.4815
         Pos Pred Value : 0.8133
         Neg Pred Value : 0.9630
             Prevalence : 0.6949
         Detection Rate : 0.6893
   Detection Prevalence : 0.8475
      Balanced Accuracy : 0.7367

       'Positive' Class : Y

> log_model_acc = mean(test_classifier == testset$Loan_Status) * 100
> log_model_acc
[1] 83.61582
```

The logistic model on testset has an accuracy of 83%

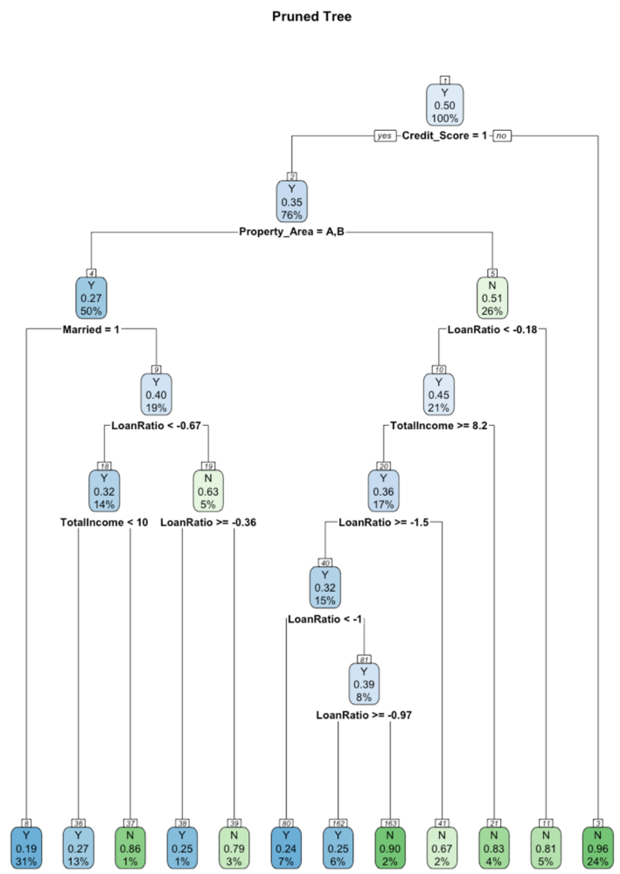
We weill now begin building a CART model for the categorical variable Loan_Status using the balanced train set data.

```
> ## Do Cart on balanced Data
> bcart = rpart(Loan_Status ~. , data = balancetrain_data, method="class",control = rpart.control(cp=0))
> printcp(bcart)

Classification tree:
rpart(formula = Loan_Status ~ ., data = balancetrain_data, method = "class",
    control = rpart.control(cp = 0))

Variables actually used in tree construction:
[1] Credit_Score  LoanRatio     Married       Property_Area TotalIncome

Root node error: 288/576 = 0.5

n= 576

          CP nsplit rel error  xerror     xstd
1  0.4479167      0   1.00000 1.07639 0.041545
2  0.0358796      1   0.55208 0.55208 0.037253
3  0.0138889      4   0.44444 0.46528 0.035209
4  0.0121528      7   0.40278 0.47917 0.035569
5  0.0104167     11   0.34722 0.45486 0.034931
6  0.0092593     12   0.33681 0.44792 0.034741
7  0.0057870     15   0.30903 0.44097 0.034548
8  0.0046296     19   0.28125 0.43750 0.034450
9  0.0034722     22   0.26736 0.43403 0.034351
10 0.0011574     25   0.25694 0.41667 0.033843
11 0.0000000     28   0.25347 0.40972 0.033633
```



Once we've grown the full CART tree, we will need to prune the tree and subsequently plot it again.

```
## Pruning the tree
## Acknowledgements to Professor Neumann of BC2406 for the pruning code.
cverrorcap = bcart$cptable[which.min(bcart$cptable[,"xerror"]),"xerror"] + bcart$cptable[which.min(bcart$cptable[,"xerror"]),"xstd"]

i <- 1; j<-4
while (bcart$cptable[i,j] > cverrorcap) {
  i = i+1
}

optimal_cp = ifelse(i > 1, sqrt(bcart$cptable[i,1] * bcart$cptable[i-1,1]),1)
optimal_cp


bcart2 = prune(bcart, cp = optimal_cp)
rpart.plot(bcart2,nn=T,main="Pruned Tree")
```



Now we shall begin calculating the accuracy of the CART model based on the balanced train set data!

```
> ## Testing CART on trainset data
> predictcart1 = predict(bcart2, newdata = balancetrain_data, type = "class")
> result1 = data.frame(balancetrain_data$Loan_Status, predictcart1)
> cart1_accuracy = mean(predictcart1 == balancetrain_data$Loan_Status)
> table(actual = balancetrain_data$Loan_Status, predictcart1)
      predictcart1
actual   Y   N
     Y 263  25
     N  75 213
> cart1_accuracy = cart1_accuracy*100
> cart1_accuracy
[1] 82.63889
```

This gives us a CART model accuracy of 82.6% for the balanced train set data. We will now begin testing on the test set data!

```
> ## Using it on the testset
> predictcart2 = predict(bcart2, newdata = testset, type="class")
> result2 = data.frame(testset$Loan_Status, predictcart2)
> cart2accuracy = mean(predictcart2 == testset$Loan_Status)
> table(actual = testset$Loan_Status, predict=predictcart2)
      predict
actual   Y   N
     N  19  35
     Y 114   9
> cart2accuracy = cart2accuracy * 100
> cart2accuracy
[1] 84.18079
```

The CART model on test set accuracy is rounded up as 84.2%

Presenting it in a table format.

```
> ## Comparing accuracy of log model and cart model
> accuracy_table = data.frame("Model_Table" = 2:1)
> rownames(accuracy_table) = c("CART on Balanced Data", "Logistic Regression on Balanced Data")
> colnames(accuracy_table) = "Model Accuracy In Percentage"
> accuracy_table[1,1] = cart2accuracy
> accuracy_table[2,1] = log_model_acc
> accuracy_table
                                     Model Accuracy In Percentage
CART on Balanced Data                                    84.18079
Logistic Regression on Balanced Data                     82.48588
> ## From this, we can see that the CART model has a higher accuracy on balanced data.
```

Answer to 3c):

```
> ## --------- 3C ------------- ##
> bcart2$variable.importance
 Credit_Score     LoanRatio   TotalIncome Property_Area       Married        Gender    Dependents     Education 
  78.13822450   24.42009728   21.39240236   11.74354102    6.24895107    1.08215133    0.87645064    0.02297602 
> ## This shows us that credit score is the most important variable
> ## Third most important variable would be the total income of both applicants
> ## Second most important variable would be the LoanRatio consisting on Loan Amount/ Loan Amount Term
> ## Fourth important would be the property area
> ## Least important would be marriage status
> ## The rest of the variables are only marginally important and not really significant
```

Answer to Q3d):

To answer this question, we have to look at the base reference level for our analysis.

```
> levels(balancetrain_data$Loan_Status)
[1] "Y" "N"
> ## This shows us that our base reference level is Yes. This will be our positive value as can be seen.
> ## A type one error is a false positive error.
> ## Hence a type one error in our loan status analysis would be that a applicant is falsely predicted to be a future positive loan status.
> ## What this means is that the applicant is actually not worthy of a loan, but the prediction algorithm classifies them as a worthy loan applicant.
>
> ## A type two error in this is a false negative error.
> ## What this means is that our applicant is truly a worthy loan applicant, but the prediction algorithm falsely deems them as unworthy.
>
> ## Therefore, for a bank, the more serious error would be a type 1 error where the loan applicant is truly unworthy of receiving a loan, but the prediction models deemed them worthy.
> ## As a result, the bank would have loaned money to an unworthy person which increases the chance that the loan applicant might not pay back the money they loaned.
```

Our base level is Yes. This will be our positive value for type 1 and type 2 error comparisons.

A type one error is a false positive error.

Hence a type one error in our loan status analysis would be that an applicant is falsely predicted to be a future positive loan status. What this means is that the applicant is actually not worthy of a loan, but the prediction algorithm classifies them as a worthy loan applicant.

A type two error in this is a false negative error.

What this means is that our applicant is truly a worthy loan applicant, but the prediction algorithm falsely deems them as unworthy. The cost of a type 2 error in this case would be lost potential revenue if the applicant is actually worthy of paying loan interest and the principal amount back

Therefore, for a bank, **the more serious error would be a type 1 error** where the loan applicant is truly unworthy of receiving a loan, but the prediction models deemed them worthy. As a result, the bank would have loaned money to an unworthy person which increases the chance that the loan applicant might not pay back the money they loaned. This error suggests that the bank will lose money instantly if the applicant defaults on the loan and is unable to pay the loan back. Making it the more serious error.

Answer to Q4:

To reduce the chance of a type 1 error, the bank could run hypothesis testing with a smaller significance level. However, this method required some fine tuning as lowering the significance level by too much would mean that the bank’s researcher has failed to capture the true parameter of the hypothesis tests.

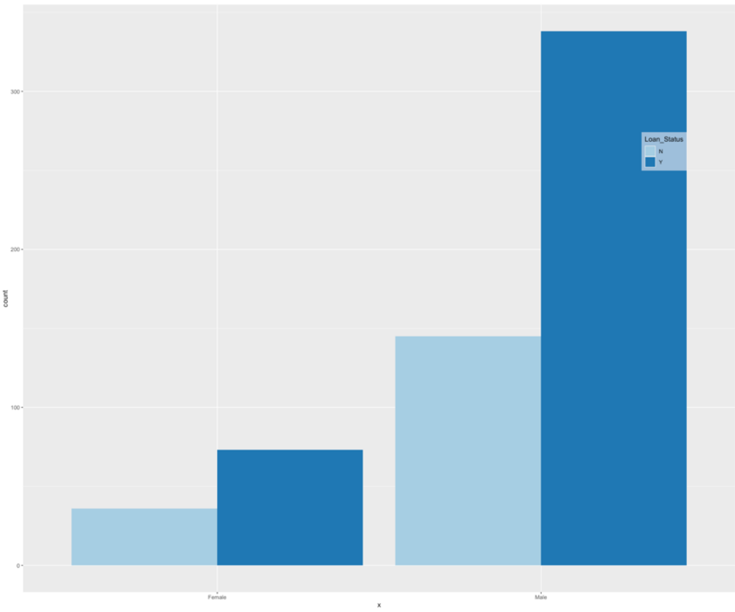
Answer to Q5:

We will start by analysing the final dataset. From the code below, we can see that there are slightly more than 4 times the number of males that applied for a loan with the bank.

```
> ## ----------- Q5 -------------- ##
> ## Final Dataset
> GDE1 = copydt
> GDE1$Gender = ifelse(GDE1$Gender == 1, "Male","Female")
> GDE1$Gender = as.factor(GDE1$Gender)
> summary(GDE1$Gender)
Female   Male
   109    483
```

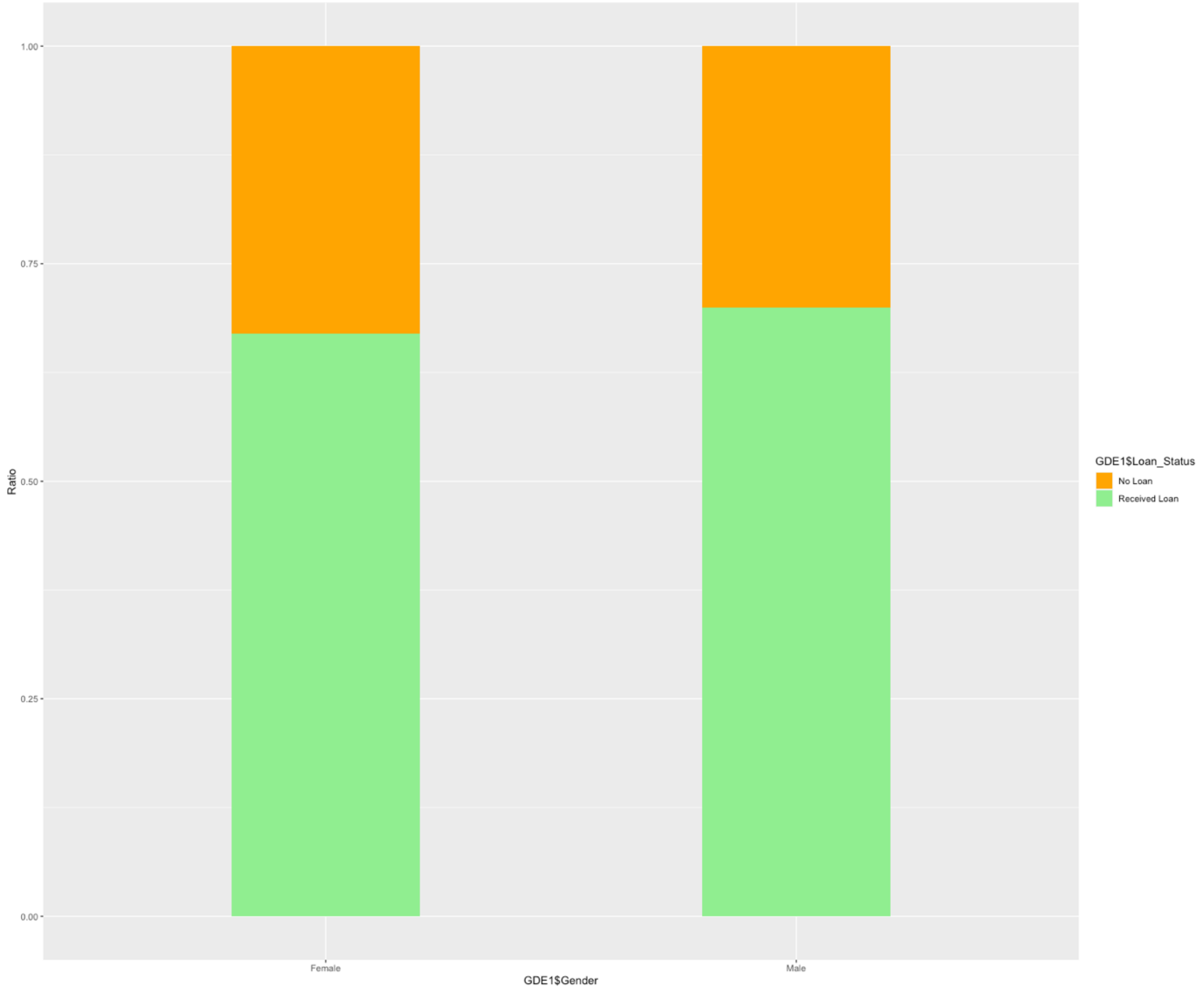
We've also plotted out the graph to showcase the proportion of males and females who managed to successfully receive a loan from the bank.

```
## Using Data exploration to see if there is indeed sexual discrimination
ggplot(data=GDE1,aes_string(factor(GDE1$Gender),fill='Loan_Status'))+ geom_bar(position = 'dodge')+ theme(legend.position = c(.9,.75
),legend.background=element_rect(fill = alpha("white", 0.5)))+ scale_fill_brewer(palette = 'Paired')
## While at first glance, this might seem that there is a disproportionate amount of males that got loan approved.
## This shows us that there is in fact more males that even applied for the loans as compared to females
```



We will now look into the specific comparison of proportion between males and females that received a loan.

```
## Looking into the relevant specific columns
GDE_df = data.frame(GDE1$Gender, GDE1$Loan_Status)
options(repr.plot.width = 10, repr.plot.height = 7)
ggplot(GDE_df, aes(x=GDE1$Gender, fill= GDE1$Loan_Status)) + geom_bar(position ="fill", width = 0.4) + ylab("Ratio") + scale_fill_manual
(labels = c("No Loan","Received Loan"), values = c("orange", "lightgreen"))
```



From the graph, we can see that the amount of proportion of females to males differ only slightly. This potentially suggests that there isn't a gender bias but rather just that more males applied for the loan. I've developed a separate table to analyse the ratio of this observation.

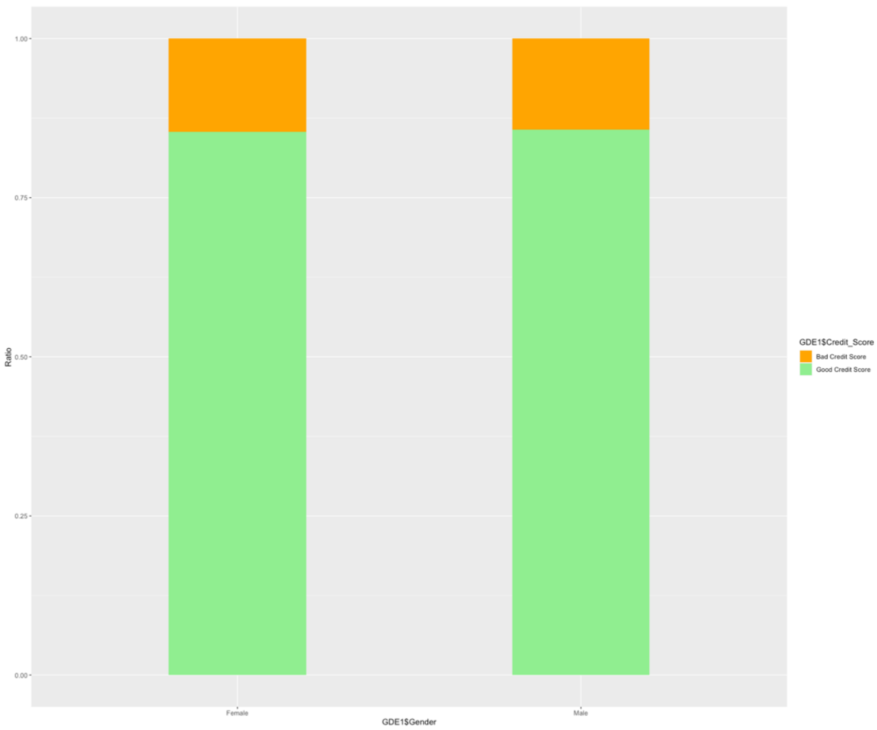
```
> ## Ratio Analuysis based purely on Gender
> females_approved = (sum(GDE1$Gender == "Female" & GDE1$Loan_Status =="Y")/sum(GDE1$Gender == "Female"))
> females_approved
[1] 0.6697248
> males_approved = (sum(GDE1$Gender == "Male" & GDE1$Loan_Status =="Y")/sum(GDE1$Gender == "Male"))
> males_approved
[1] 0.699793
>
> GDETable = data.frame("Approval Rates" = 2:1)
> rownames(GDETable) = c("Males Approved", "Females Approved")
> colnames(GDETable) = "Approval Percentages"
> GDETable[1,1] = males_approved
> GDETable[2,1] = females_approved
> GDETable
                 Approval Percentages
Males Approved              0.6997930
Females Approved            0.6697248
```

It seems that there is a very similar approval percentage for males and females based purely on their gender. This further disproves that there is a gender bias in the loan approval process.

Looking back at variable importance, the strongest variable that is correlated to loan status is credit score. Hence, one might argue that perhaps more males receive a good credit score as compared to females. As such, we will build similar ratio analysis and plot similar graphs to showcase the results of this hypothesis.

```
> ## Ratio Analysis based purely of Gender against Credit Score
> females_approved = (sum(GDE1$Gender == "Female" & GDE1$Credit_Score =="1")/sum(GDE1$Gender == "Female"))
> females_approved
[1] 0.853211
> males_approved = (sum(GDE1$Gender == "Male" & GDE1$Credit_Score =="1")/sum(GDE1$Gender == "Male"))
> males_approved
[1] 0.8571429
> GDETable = data.frame("Approval Rates" = 2:1)
> rownames(GDETable) = c("Males Approved", "Females Approved")
> colnames(GDETable) = "Approval Percentages"
> GDETable[1,1] = males_approved
> GDETable[2,1] = females_approved
> GDETable
                 Approval Percentages
Males Approved              0.8571429
Females Approved            0.8532110
```

Similarly, there are similar percentages for both males and females to receive a positive credit score.



Graphically, it seems like the hypothesis is also disproved when both males and females receive the same credit score. Hence there is no gender bias even in the variable of credit score.

We will now attempt the same methods but with a balance of genders in our analysis. Which we start by creating a balance of males and females in our test data.

```
> # Conducting a sampling for gender, this will give us equal representation.
> summary(GDE1$Gender)
Female   Male
   109    483
> gender_sampled = ovun.sample(Gender ~., data= GDE1, seed = 8, method ="under", N = (109*2))$data
Error in (function (formula, data, method, subset, na.action, N, p = 0.5,  :
  Too few observations.
> summary(gender_sampled$Gender)
  Male Female
   108    110
>
> females_approved = (sum(gender_sampled$Gender == "Female" & gender_sampled$Loan_Status =="Y"))/sum(gender_sampled$Gender == "Female")
> females_approved
[1] 0.6727273
> males_approved = (sum(gender_sampled$Gender == "Male" & gender_sampled$Loan_Status =="Y"))/sum(gender_sampled$Gender == "Male")
> males_approved
[1] 0.6759259
```

By removing the inflated number of males that applied, we can see the pure ratio of equal number of males and females. Similarly, we can see that males and females that are in our balanced data receive generally the same ratio of approval. Further disproving the gender bias theory.

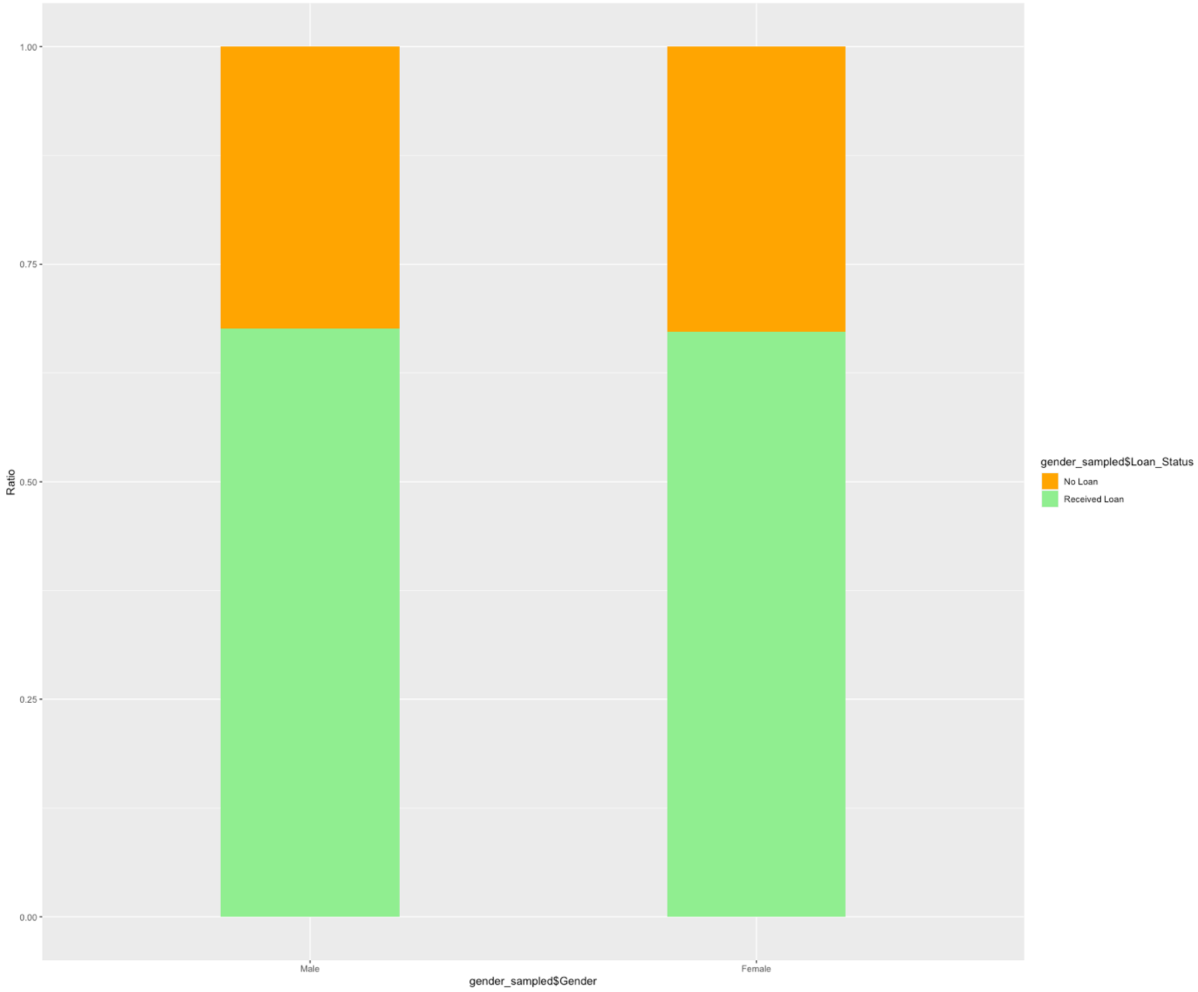
We will now run a logistic regression to see if Gender is correlated with Loan Status

```
> gender_log2 = glm(Gender~., family = binomial, data = gender_sampled)
> summary(gender_log2)

Call:
glm(formula = Gender ~ ., family = binomial, data = gender_sampled)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1752  -0.7989   0.4282   0.8590   2.1837

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     4.51118    3.37283   1.338   0.1811
Married        -1.70947    0.34734  -4.922 8.59e-07 ***
Dependents.L   -0.98407    0.59424  -1.656   0.0977 .
Dependents.Q   -0.05130    0.54026  -0.095   0.9243
Dependents.C    0.18426    0.46098   0.400   0.6894
Education       0.62247    0.39808   1.564   0.1179
Self_Employed   0.13800    0.50882   0.271   0.7862
Credit_Score1   0.17807    0.53291   0.334   0.7383
Property_AreaB  1.01340    0.39816   2.545   0.0109 *
Property_AreaC -0.37713    0.41393  -0.911   0.3622
Loan_StatusY   -0.11739    0.41287  -0.284   0.7762
TotalIncome    -0.56936    0.36997  -1.539   0.1238
LoanRatio      -0.00932    0.33259  -0.028   0.9776
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 302.19  on 217  degrees of freedom
Residual deviance: 234.06  on 205  degrees of freedom
AIC: 260.06

Number of Fisher Scoring iterations: 4
```

From this, we can see that Loan Status has no significant correlation with Gender.

We will lastly end of with calling a graph to visually depict this result.

```
> ## displaying graphs using balanced data
> GDE_df = data.frame(gender_sampled$Gender, gender_sampled$Credit_Score, gender_sampled$Loan_Status)
> options(repr.plot.width = 10, repr.plot.height = 7)
> ggplot(GDE_df, aes(x=gender_sampled$Gender, fill= gender_sampled$Loan_Status)) + geom_bar(position ="fill", width = 0.4) + ylab("Ratio") + sca
le_fill_manual(labels = c("No Loan","Received Loan"), values = c("orange", "lightgreen"))
>
> ## This shows us that Gender is not correlated with Loan Status. Hence further disproving a gender bias
```



Therefore, through our analysis, we have checked on the unbalanced data and it reveals that there isn't a gender discrimination through the loan process. We then created a balanced dataset which shares the same results that there is no gender discrimination. We've also analysed gender on credit score and there isn't a strong correlation between gender and credit score as well. Hence, through our analysis, we are unable to detect gender discrimination.

Answer to Q6:

One way that the bank can increase the accuracy of the analytics would be to deploy an ensemble model that combines base learners into a final model of a meta learner to reduce generalization errors within the bank.

However, this method relies on the assumption that each base learner model will yield a different aspect of the data and capture differing results. To do this, the bank should develop and train individual models independently of one another using various training and test splits.

With regards to data quality, the bank could also ensure that only processed data are fed into the analysis pipeline and that only quality data are recorded. This includes the enforcement of information fields that cannot be NA as can be seen in the current dataset where certain important fields such as Gender were left as NA. This increases the quality of the data and will be of greater benefit to the bank's analysis of data later on.

Another way to increase the accuracy of this model is to allow it to train on more data perhaps from cross collaboration with other banks or other branches. This allows the model to predict and train on a greater number of samples.