DA5030: Data Mining and Machine Learning

Ghada M. Alsebayel

This project is submitted in partial fulfillment of the requirements to complete DA5030 Data Mining and Machine Learning at Khoury College of Computer Science- Northeastern University.
Spring 2021.

Introduction:

Family planning is essential to women’s health, and access to contraceptives is a key enabler to achieve family planning. Access to contraceptives however can vary greatly among different countries, socioeconomic classes, religious and ethnic groups. Furthermore, with the overwhelmingly variant choices, sometimes even when granted access to contraceptives, women can find it difficult to choose the most appropriate contraceptive method that addresses their individual needs.
In this project, machine learning is used as a tool to improve the delivery of family planning consultations. This is achieved by the following:
- Identifying the demographic factors that contributes to lack of access to contraceptives. This in turn can aid in delivering better targeted educational campaigns.
- Predict the preferred method of contraception for women based on the preferences of others in the same demographic, socioeconomic group, which can potentially lower the confusion when choosing among different methods.

CRISP-DM Framework:

This project will follow the CRISP-DM standard. CRISP-DM stands for Cross-industry standard process for data mining. It is a standard model for processes in data mining projects. It consists of the following phases.

Buisness Understanding:

Indonesia is the world’s largest island. It ended the year 1987 with a population of 169,149,000. The Indonesian government established a plan to achieve its developmental goals by announcing a population policy, which includes reducing the rate of population growth, achieving a redistribution of the population, adjusting economic factors, and creating prosperous families.
The National Indonesia Contraceptive Prevalence Survey(NICPS) was an initial step to achieving these goals.

Data Understanding:

The National Indonesia Contraceptive Prevalence Survey (NICPS) collected data on fertility and family planning in order to provide policy makers and program managers with information useful for evaluating and improving Indonesia’s Family Planning Program. The data obtained for this project is a subset of the NICPS 1987. (https://dhsprogram.com/pubs/pdf/SR9/SR9.pdf) It was obtained from the machine learning repository of The University of California- Irvine UCI.

To understand the data I am working with, I will begin by examining the the first couple of rows:

```
## 'data.frame':    1473 obs. of  10 variables:
##  $ V1 : int  24 45 43 42 36 19 38 21 27 45 ...
##  $ V2 : int  2 1 2 3 3 4 2 3 2 1 ...
##  $ V3 : int  3 3 3 2 3 4 3 3 3 1 ...
##  $ V4 : int  3 10 7 9 8 0 6 1 3 8 ...
##  $ V5 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V6 : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ V7 : int  2 3 3 3 3 3 3 3 3 2 ...
##  $ V8 : int  3 4 4 3 2 3 2 2 4 2 ...
##  $ V9 : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ V10: int  1 1 1 1 1 1 1 1 1 1 ...
```

```
## Checking the first couple of rows
head(indo.contraceptives)
```

	w....	w.education	h.education	num.children	w.religion	w.w...	h.job	living.index	me...
	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	24	2	3	3	1	1	2	3	0
2	45	1	3	10	1	1	3	4	0
3	43	2	3	7	1	1	3	4	0
4	42	3	2	9	1	1	3	3	0
5	36	3	3	8	1	1	3	2	0
6	19	4	4	0	1	1	3	3	0

6 rows | 1-10 of 11 columns

The data consists of 1473 observations with 10 variables. The semantics of each variable are shown in the following table:

Attributes Information:

Attribute	**Type**	**Meaning**
w.age	Numerical	The age of the wife
w.education	Categorical	The educational level of the wife. It has 4 levels where: 1=Low, 2, 3, 4=High
h.education	Categorical	The educational level of the husband. It follows the same standard as the wife.
num.children	Numerical	The number of children ever born for the wife
w.religion	Binary	The religion of the wife. It is a binary variable where: 0=Non-Islam, 1=Islam

w.work	Binary	The status of the wife in the job market. It is a binary variable where: 0=Working, 1=Not-Working
h.job	Categorical	The husband's occupation
living.index	Categorical	Standard-of-living index. It is a categorical variable where: 1=Low, 2, 3, 4=High
media	Binary	The exposure to media. It is a Binary variable where: 0=Good exposure, 1=Not good
contraceptive	Categorical	The type of contraceptive method a women uses. Where:1=No-use, 2=Long-term, 3=Short-term

Now that I have seen the structure of the data. I will dig a little bit deeper and do some data exploration with descriptive statistics. First we look into the age factor.

1.Age:

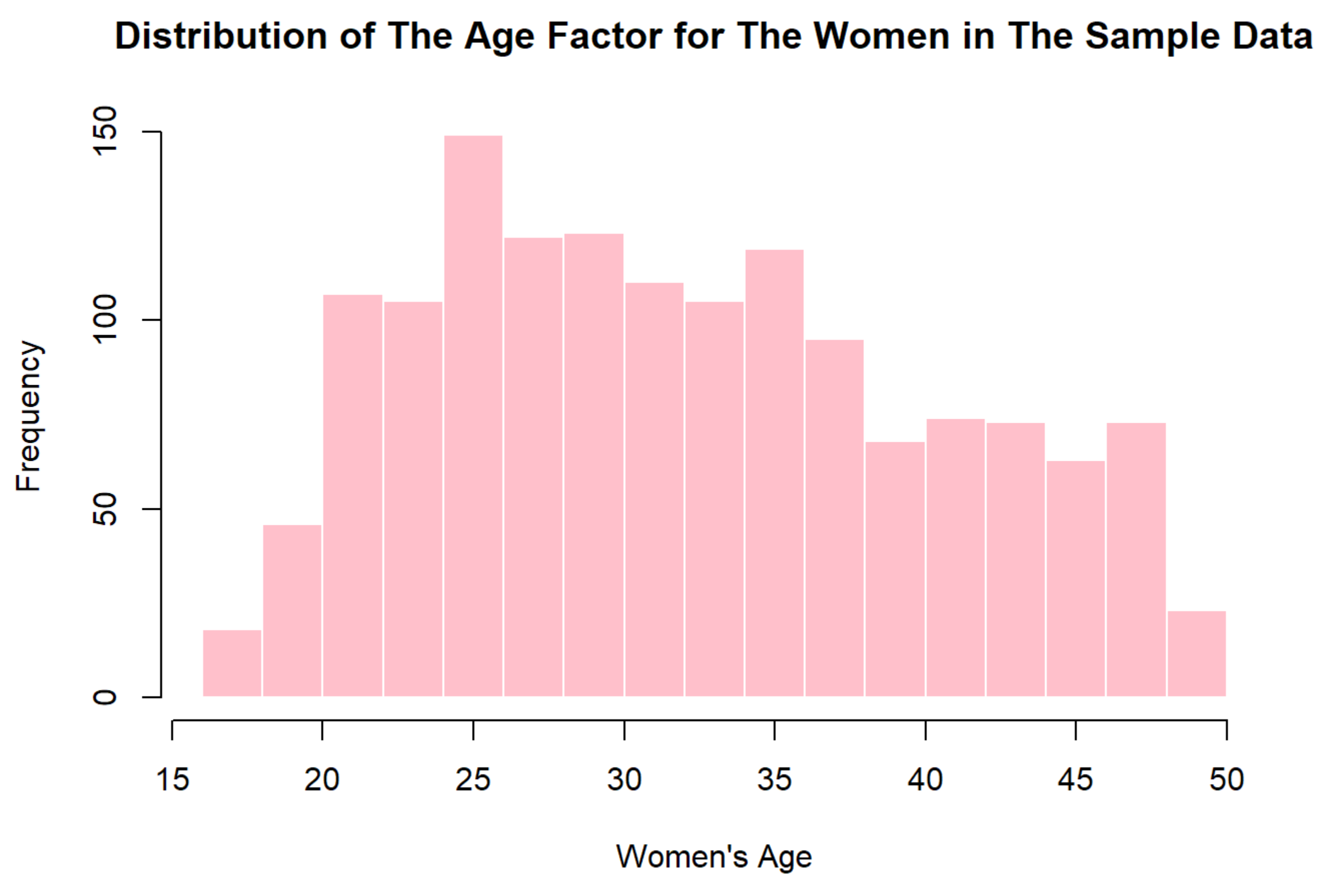
```
summary(indo.contraceptives$w.age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   26.00   32.00   32.54   39.00   49.00
```

```
sd(indo.contraceptives$w.age)
```

```
## [1] 8.227245
```

```
## Plotting the age
hist(indo.contraceptives$w.age,main="Distribution of The Age Factor for The Women in The Sample
 Data",
xlab="Women's Age",
col="pink", border = FALSE)
```



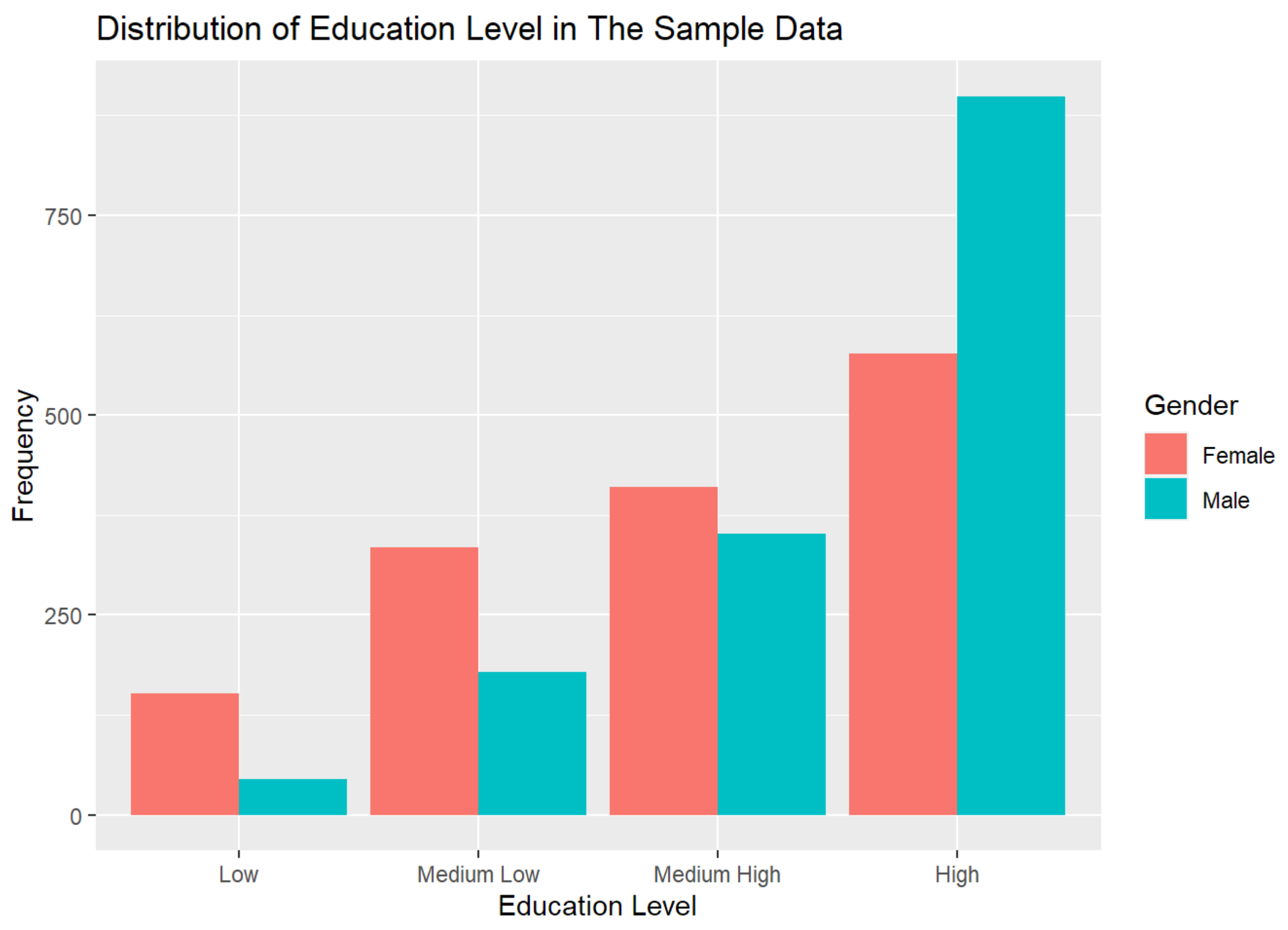
The average age of the women who participated in the study is 32.54 while the range is between 16 and 49 years old. Given that women older than 49 years old are typically in menapose, that age group was excluded as it will probably skew the data. Women younger than 16 were not included either, this could be because teenage pregnancy is a different phenomenon that requires specifically designed analysis.
The standard deviation, a measure of data spread, is 8.2 years.
The diagram above shows the distribution of the age factor for the women in the study.

2.Education Level:
When looking into the level of education for both the participating women and their spouses, we can see that both are quite educated

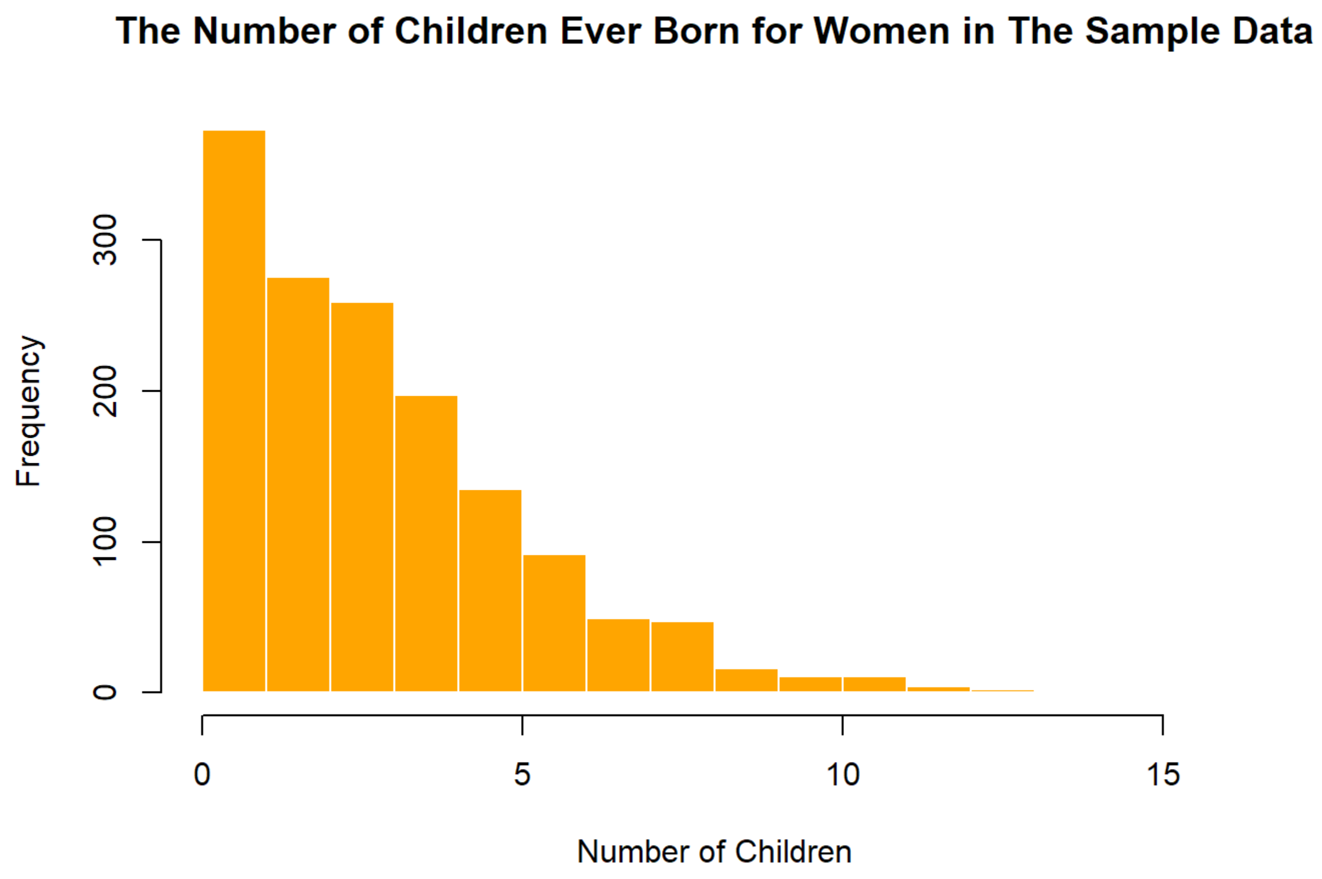
```
head(education)
```

	Education level <fct>	Women <int>	Men <int>
1	Low education	152	44
2	Medium low education	334	178
3	Medium high education	410	352
4	High education	577	899

4 rows



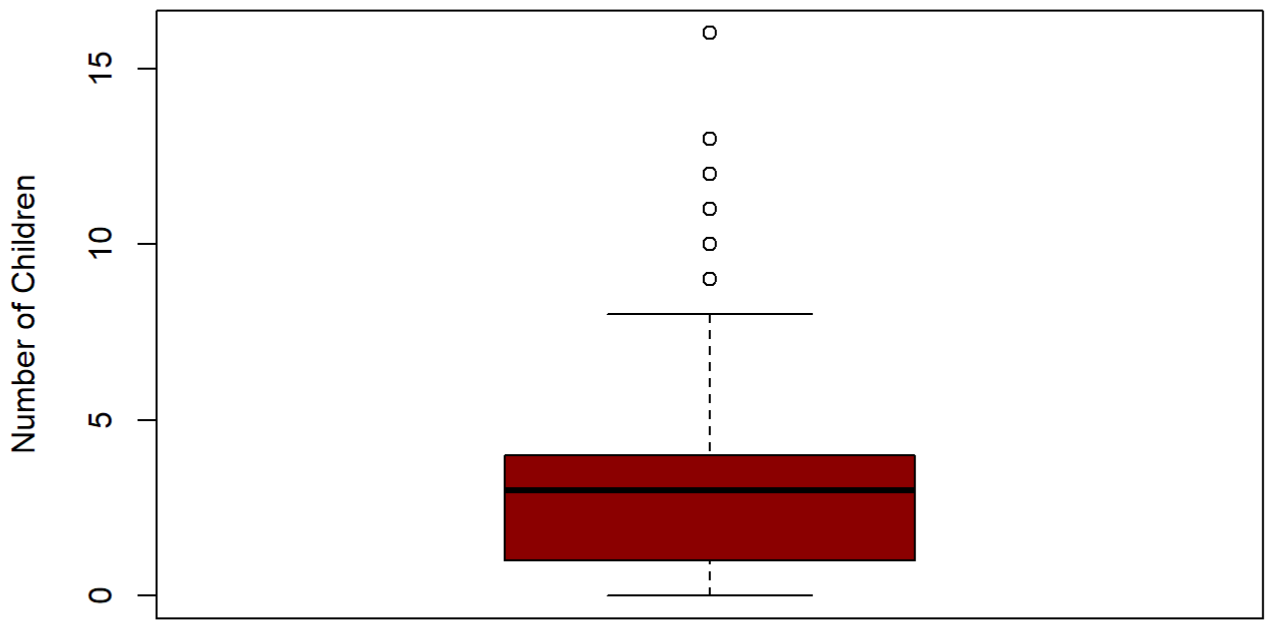
3.The Number of Children Born per Women:



```
summary(indo.contraceptives$num.children)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   1.000   3.000   3.261   4.000  16.000
```

The mean number of children born per women = 3.261. The maximum number is 16. This is probably an outlier. To double check I will be Visualizing the data points which helps in detecting outliers.

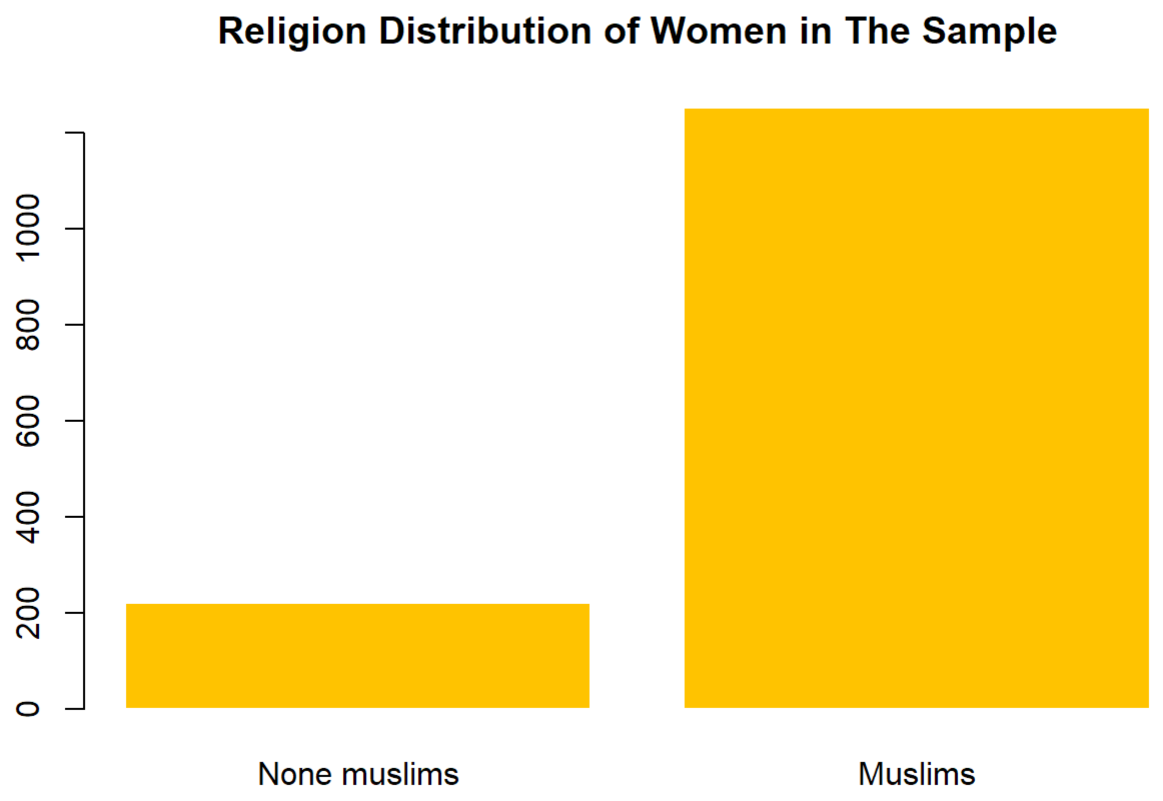


As anticipated, having more than 10 children is not the norm. I will be dealing with outliers in the next phase. For now, I will be looking into the next variable.

4.Religion of the wife

	Religion <chr>	Count <int>
1	None Muslim	220
2	Muslim	1253

2 rows



The majority of the women who participated in the study were Muslims. They represent the following percentage of the whole population:

```
religion[2,2]/sum(religion$Count)*100
```

```
## [1] 85.06449
```

5.Employment Status of Women in The Sample:

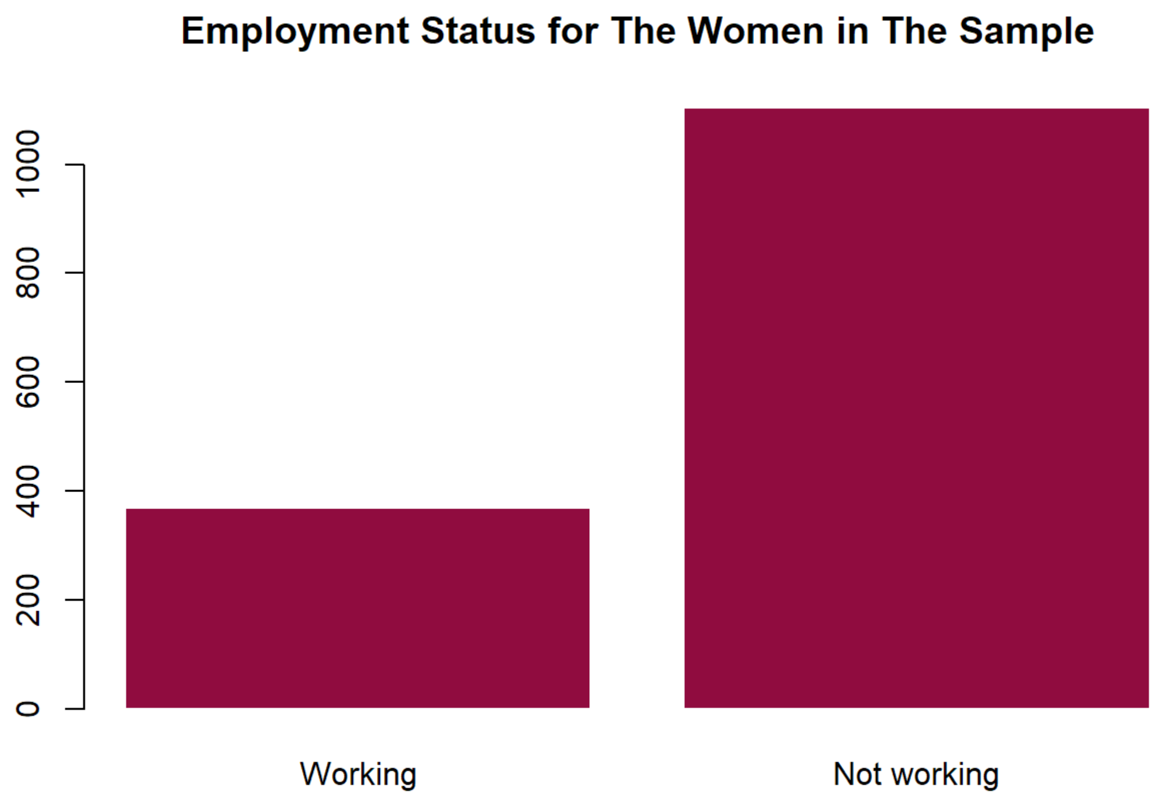
	Working_Status <chr>	Count <int>
1	Working	369
2	Not Working	1104

2 rows

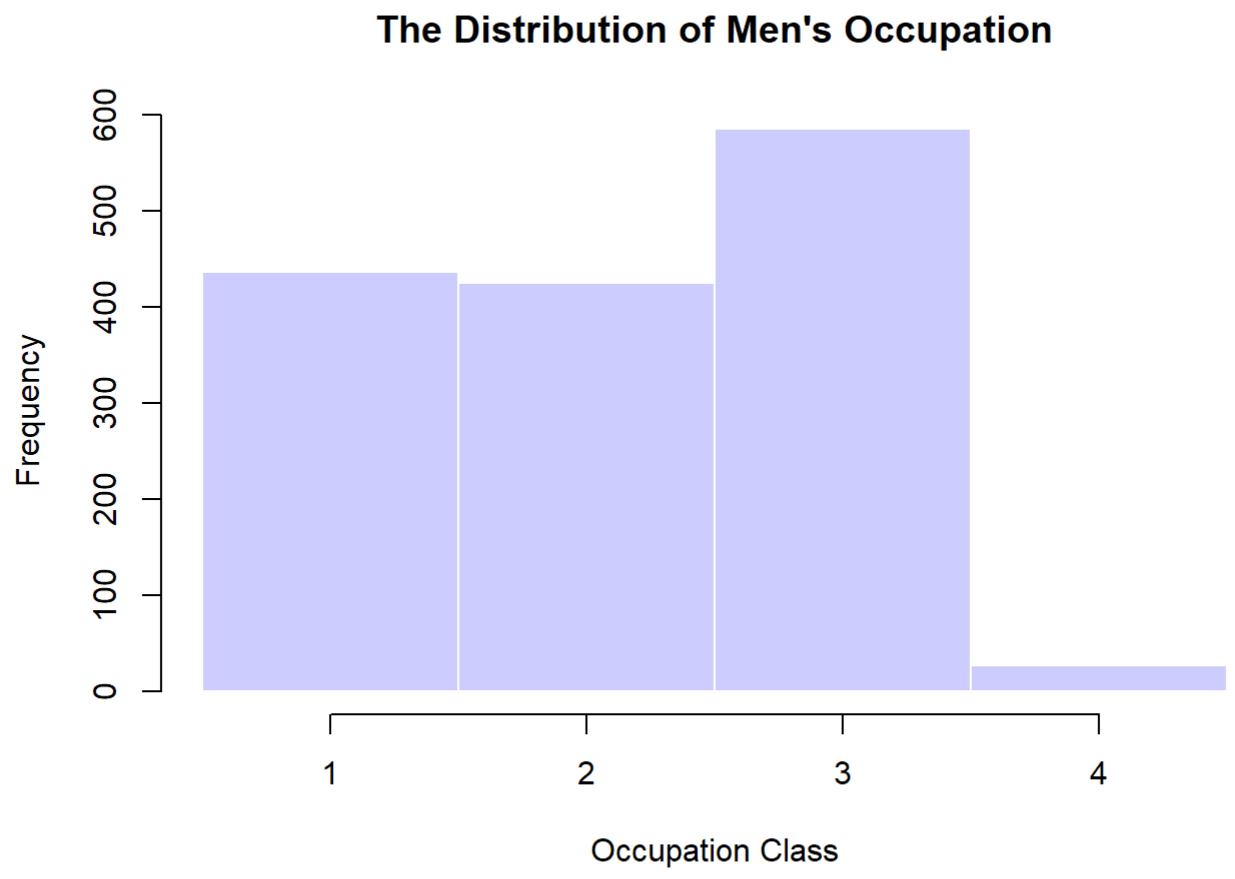
The majority of women in the sample are not working. They represent the following percentage of the whole population:

```
work[2,2]/sum(work$Count)*100
```

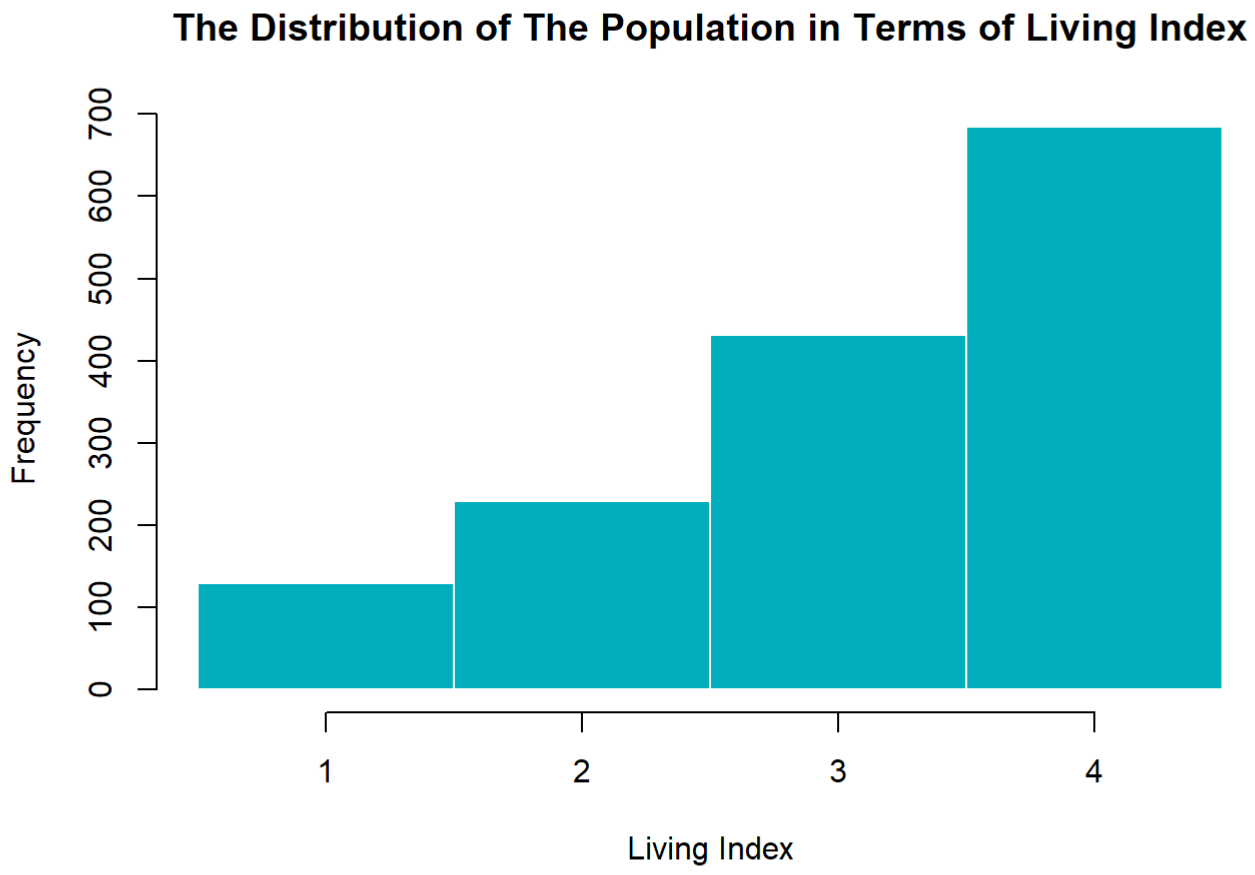
```
## [1] 74.94908
```



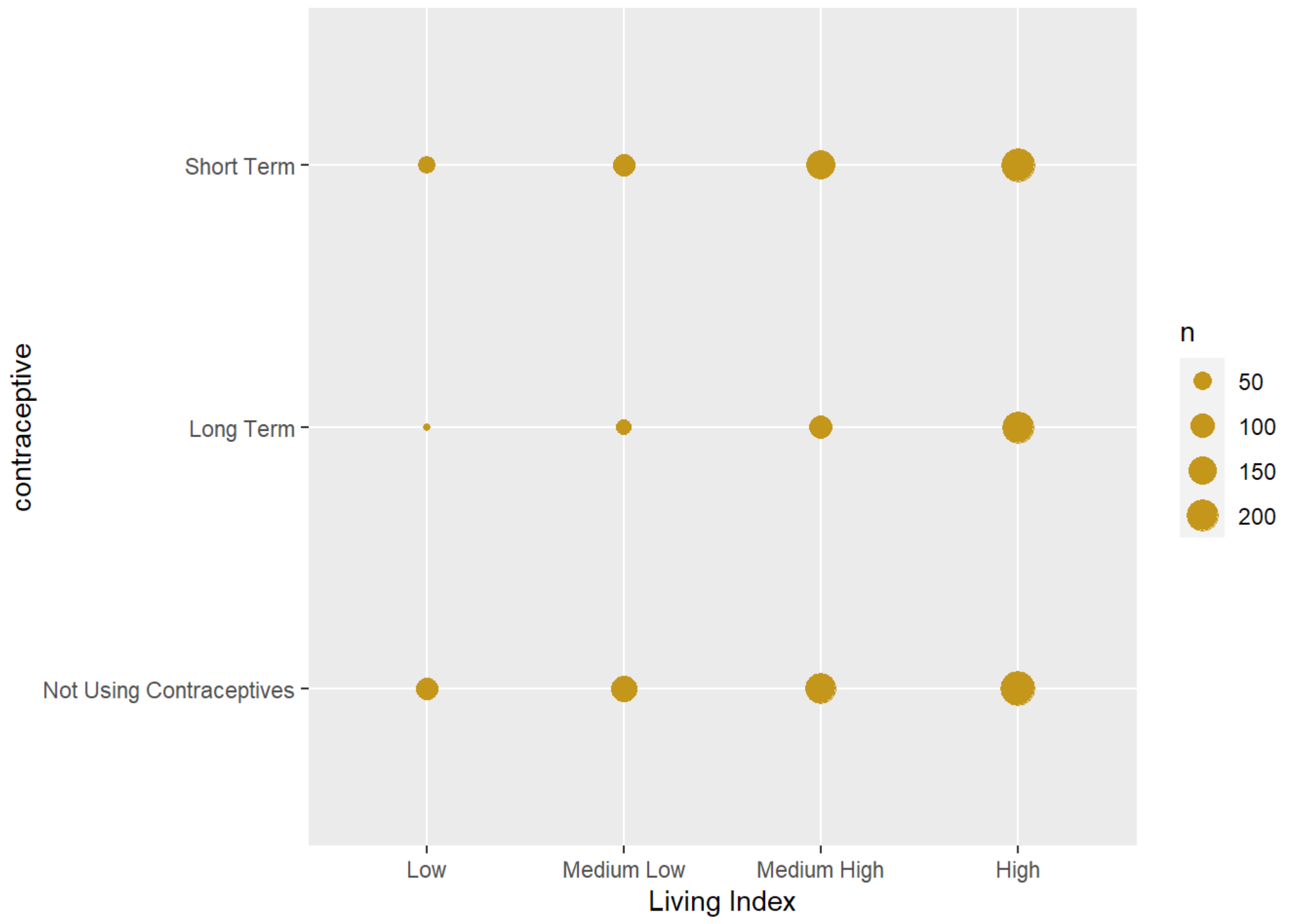
6. Occupation of The Husband:



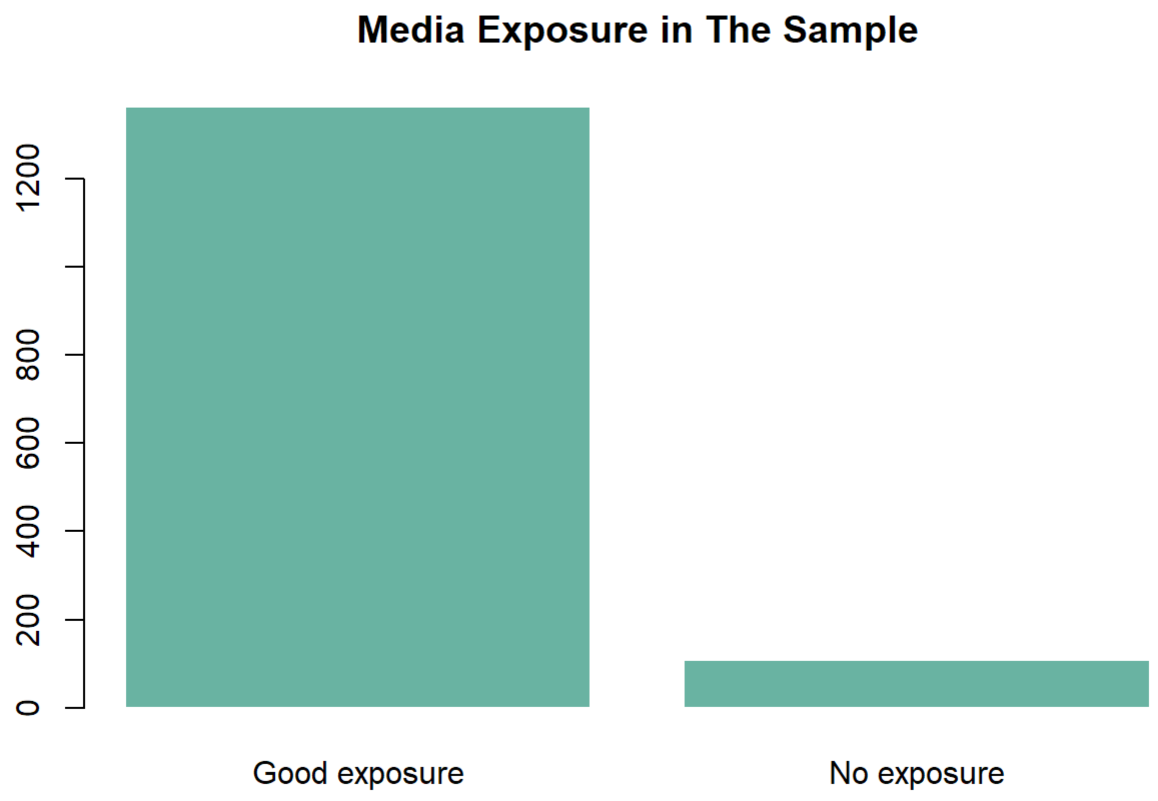
7.Living Index: For the living index, the study considers four categories: low income, medium low income, medium high income, and high income. The following diagram shows the distribution of the population based on living index. The majority of the participants are earning high income.



I was curious to know if there was a relationship between the living index and the use of contraceptives. So, I plotted the following diagram. It shows the used method of contraceptives in relation to living index.



8.Media Exposure:



The majority of the women in the study had good exposure to media.

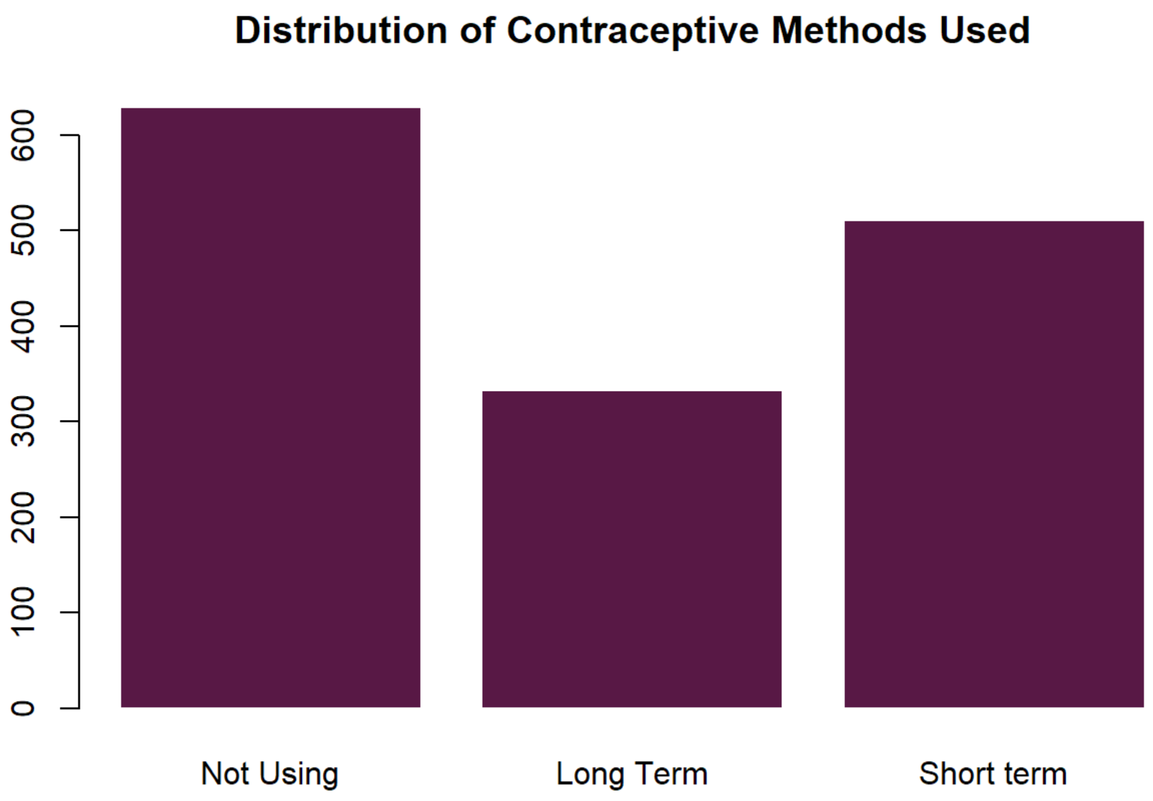
9.Contraceptive Method:

This is the target variable that I will be predicting the next sections of this report. The goal is to build a model that can predict which method of contraceptives a women uses based on her demographic data.
The data being used to build the models was collected in 1987, so new methods might have emerged. However, if the concept works with the given data, it could be extended to new data.
In this dataset, contraceptives were categorized into three classes; 1. Not using any contraceptives, 2. Using long term contraceptives, and 3. Using short term contraceptives.
Before modeling the data, it might be worthwhile to look into the target variable in relation to other variables. Since many of my variables are categorical, I will be using chi squared analysis.

```
## Investigating the distribution of contraceptive methods
table(indo.contraceptives$contraceptive)
```

```
##
##   1   2   3
## 629 333 511
```

```
barplot(table(indo.contraceptives$contraceptive), col="#581845", border = FALSE,names=c("Not Usi
ng","Long Term","Short term"), main = "Distribution of Contraceptive Methods Used")
```



```
## Probability distribution
prop.table(table(indo.contraceptives$contraceptive))
```

```
##
##         1         2         3
## 0.4270197 0.2260692 0.3469111
```

```
## CHI square analysis for different variables
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$w.education)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$w.education
## X-squared = 140.46, df = 6, p-value < 2.2e-16
```

Here, Since we have a p-value of less than the significance level of 0.05, we can reject the null hypothesis and conclude that the two variables are, indeed, independent. In other words, the use of contraceptives is not dependent on the women's education.

```
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$h.education)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$h.education
## X-squared = 73.953, df = 6, p-value = 6.305e-14
```

Here the p-value is less than the significance level of 0.05, so the husband's education and the use of contraceptives are also independent variables.

```
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$w.religion)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$w.religion
## X-squared = 21.622, df = 2, p-value = 2.018e-05
```

The same applies the religion of the wife.

```
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$w.work)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$w.work
## X-squared = 5.1872, df = 2, p-value = 0.07475
```

Here the p-value is greater than the significance level of 0.05, so we can assume that the women's choice of contraceptive is defendant on her work status

```
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$h.job)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$h.job
## X-squared = 65.401, df = 6, p-value = 3.573e-12
```

The use of contraceptives and the job of the husband are independant variables.

```
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$living.index)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$living.index
## X-squared = 62.199, df = 6, p-value = 1.607e-11
```

The use of contraceptives and the living index of the family are independent.

```
chisq.test(indo.contraceptives$contraceptive, indo.contraceptives$media)
```

```
##
##  Pearson's Chi-squared test
##
## data:  indo.contraceptives$contraceptive and indo.contraceptives$media
## X-squared = 31.572, df = 2, p-value = 1.394e-07
```

The use of contraceptives and the media exposure are also independent.

Data Preparation:

Now that I have completed the initial exploration of the data, I will start preparing my data for modeling. Data preparation involves handling missing values, standardizing the data, encoding categorical variables, data transformation, and feature re-engineering.

1. Detecting and handling missing values:

```
print(sprintf("The number of missing values= %i",
sum(is.na(indo.contraceptives))))
```

```
## [1] "The number of missing values= 0"
```

It looks like the original data does not have any null values. However, for the purpose of demonstration, I will remove some value at random then impute the missing value.

```
## Generating random column number
rc<- sample(1:10, 1)
## Generating random row number
rr<- sample(1:1473, 1)
## Now I will set the value to null
indo.contraceptives[rr,rc]<- NA
## Checking if it worked
sum(is.na(indo.contraceptives))
```

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

```
print(sprintf("The number of missing values= %i",
sum(is.na(indo.contraceptives))))
```

```
## [1] "The number of missing values= 1"
```

It looks like the trick worked and now we have a null value in the dataset. I need to identify it then impute it. For imputation, I will use the mode for categorical variables and the mean for continues variables.

```
# Creating get mode function.
getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

```
## To identify the index of the missing value, I will loop
## through the entire dataset.
for (i in c(1:10)) {
  for (j in c(1:1473)) {
    if(is.na(indo.contraceptives[j,i])){
     ## At this point we found a missing value
     print("The row number")
     print(j)
     print("The column number")
     print(i)
     ## Now that we have identified the index
     ## I will call a method to impute the missing value
     ## if it is categorical
     if(is.factor(indo.contraceptives[,i])){
       indo.contraceptives[j,i]<- getmode(indo.contraceptives[,i])
     }
     ## else, I will use R's built in mean function
     else {
       indo.contraceptives[j,i]<- mean(indo.contraceptives[,i])       }
    }
  }
}
```

```
## [1] "The row number"
## [1] 533
## [1] "The column number"
## [1] 7
```

```
## Checking if it worked
sum(is.na(indo.contraceptives))
```

```
## [1] 0
```

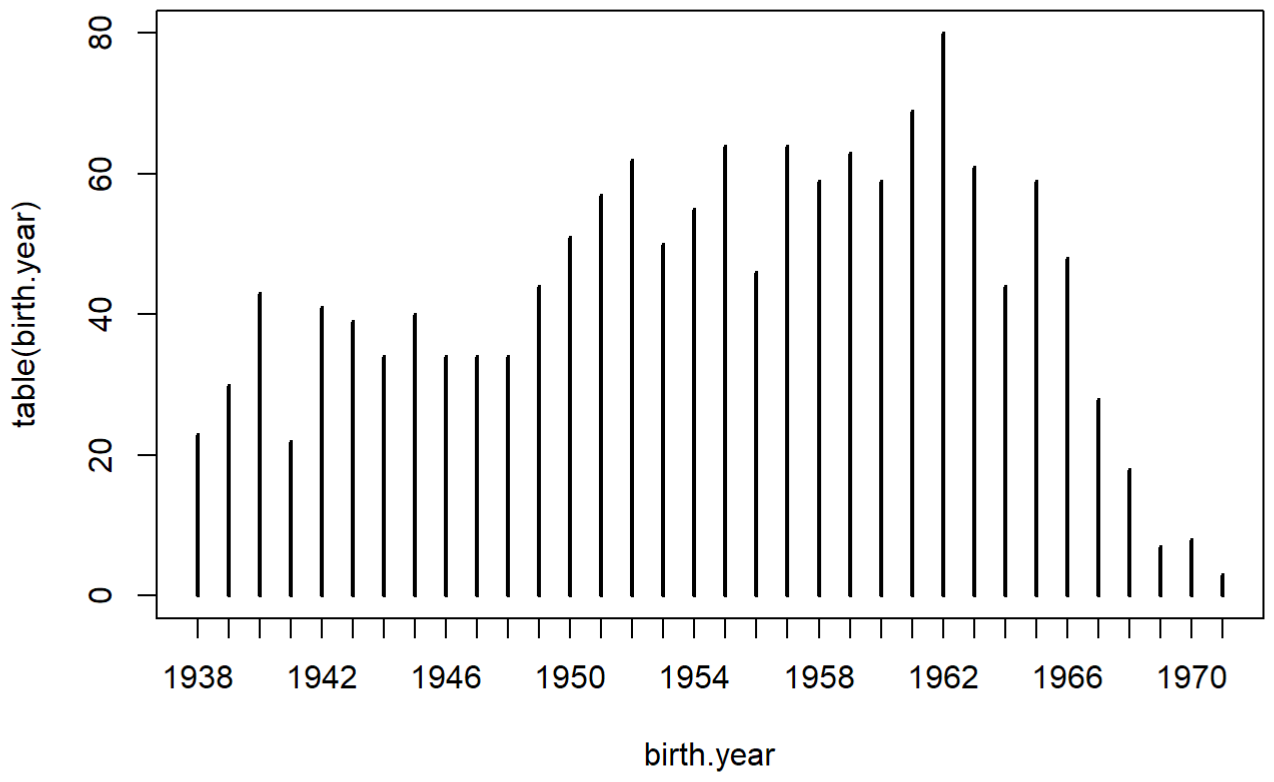
It looks like the null value has been imputed successfully.

2.feature engineering: new derived features

```
## Calculating the date year for women in the sample
birth.year<- 1987 - indo.contraceptives$w.age
```

I could not think of any derived variable that can improve the classification process. But, for purposes of demonstration, I calculated the date of birth based on the age.

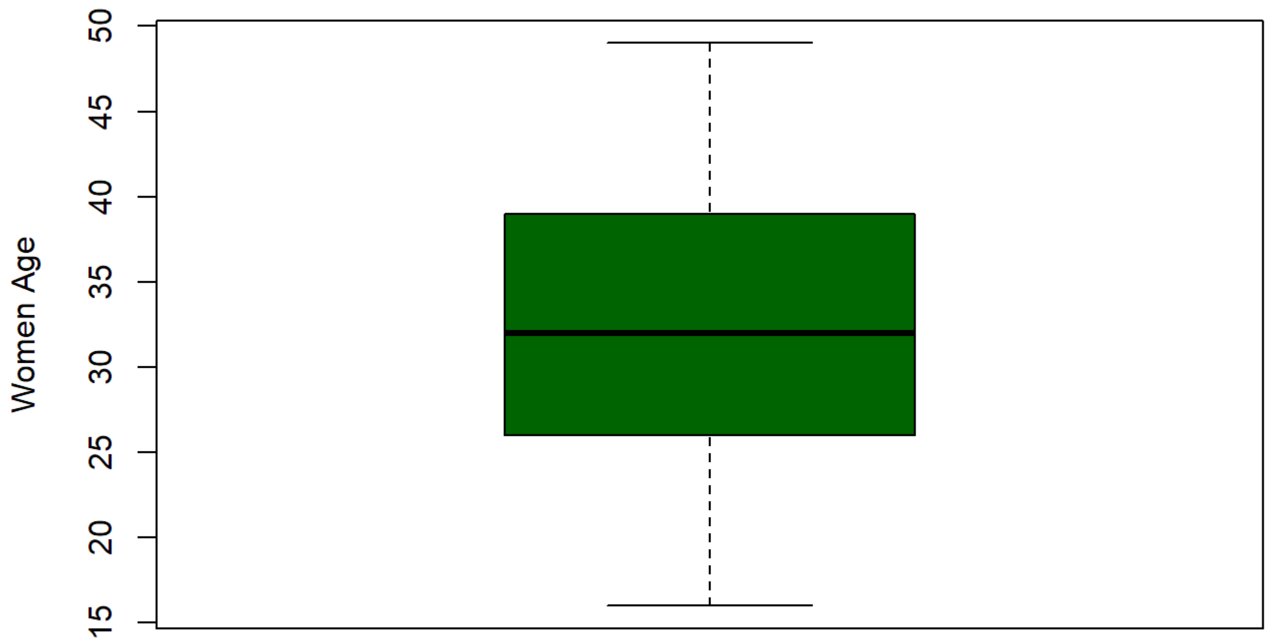
```
## Plotting the year of birth
plot(table(birth.year))
```



3. Detecting outliers:

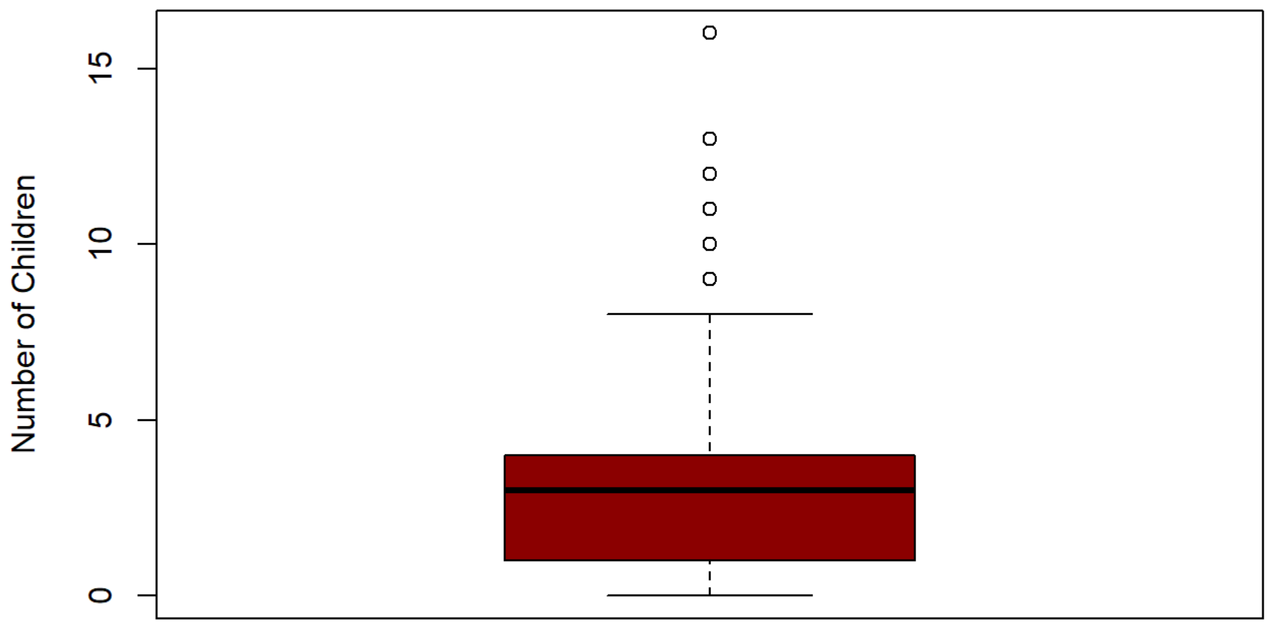
The continues variables in this dataset are only : w.age and num.childern. So, I will check these two variables for outliers.

```
boxplot(indo.contraceptives$w.age,
  col = "darkgreen",
  ylab = "Women Age"
)
```



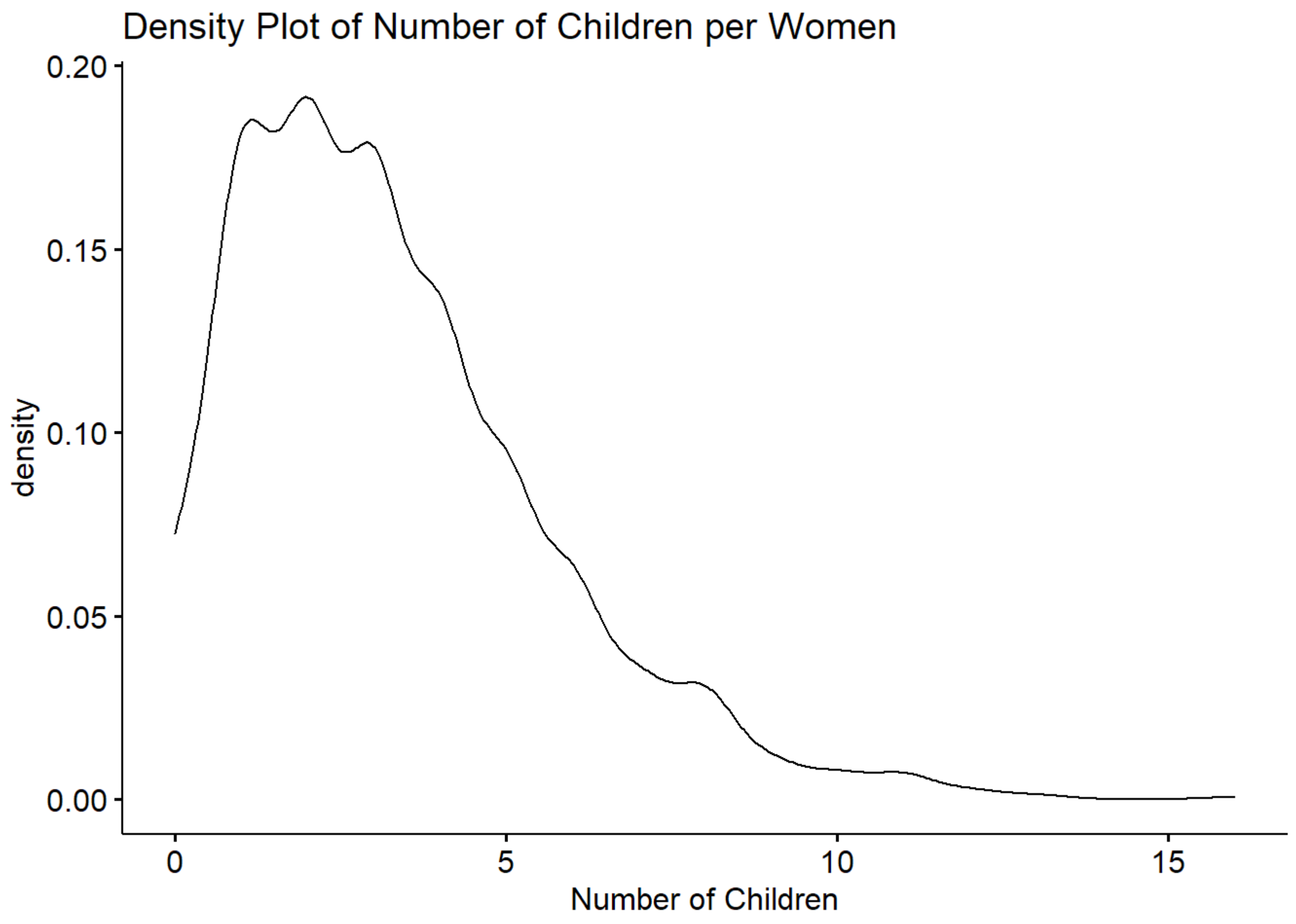
It looks like there are no outliers in the age feature. So, let us re-examine the number of children born.

```
boxplot(indo.contraceptives$num.children,
  col = "darkred",
  ylab = "Number of Children"
)
```



It is clear from the boxplot that there are some high outliers in the number of children born per women. To confirm, I will look at the data distribution.

```
## Visualizing the density plot
ggdensity(indo.contraceptives$num.children,
          main = "Density Plot of Number of Children per Women",
          xlab = "Number of Children")
```



```
## Performing shapiro test
shapiro.test(indo.contraceptives$num.children)
```

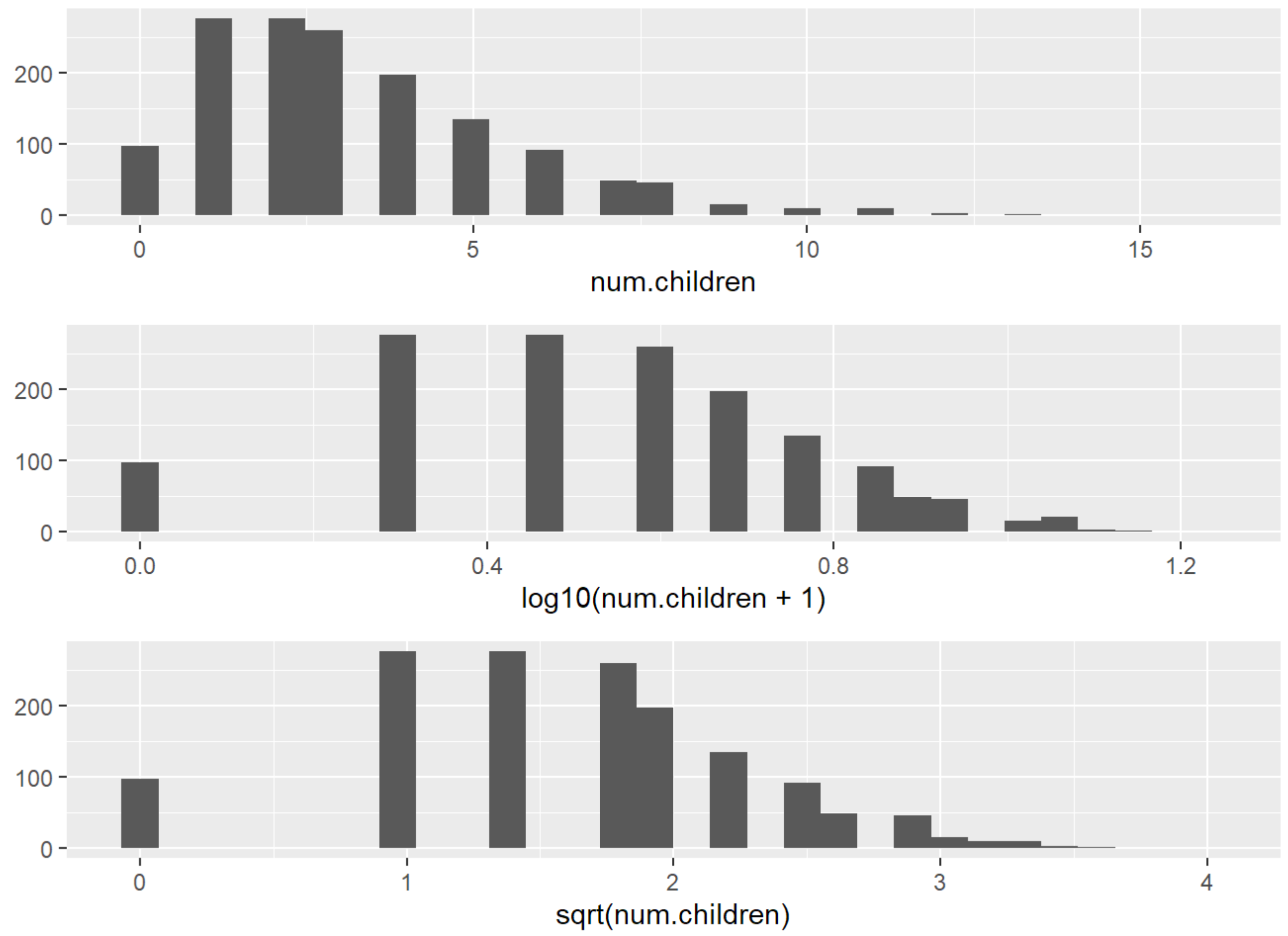
```
##
##  Shapiro-Wilk normality test
##
## data:  indo.contraceptives$num.children
## W = 0.91266, p-value < 2.2e-16
```

It appears like we have a decent right-skew in our data. This is also confirmed with the Shapiro Wilk normality test. In Shapiro test p-value > 0.05 implies that the distribution of the data is not significantly different from normal distribution. In other words, we can assume the normality. Otherwise, we can not assume normal distribution. In our case, it is quite obvious that we can not assume normality since the p-value is much smaller than 0.05.

Let us see if we can transform the feature to adjust the distribution

```
## Original data
plot.1<- qplot(x=num.children, data = indo.contraceptives)
## Log10 transformation
plot.2<- qplot(x= log10(num.children + 1), data = indo.contraceptives)
## sqrt transformation
plot.3<- qplot(x= sqrt(num.children), data = indo.contraceptives)
grid.arrange(plot.1,plot.2,plot.3, ncol=1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There are many methods that can be used to transform the features to adjust distribution including; log and square root transformation. The above diagram shows the distribution before and after applying each of these techniques. It does not appear to do much help in adjusting the transformation. In the following section, I will examine data normalization and outliers removal.

4. Normalization of Feature Values:

```
## Creating a function to calculate z-score
calculate_Z_score <- function(x){
 return((x - mean(x)) / sd(x))
}
## Standardizing the numerical variables with z-score
indo.contraceptives[,c(1,4)]<- lapply(indo.contraceptives[,c(1,4)], calculate_Z_score)
range(indo.contraceptives$w.age)
```

```
## [1] -2.010194  2.000869
```

```
range(indo.contraceptives$num.children)
```

```
## [1] -1.382787  5.401045
```

The normalization has been done successfully.
After normalizing the data we can remove outliers, for example the values that are more than 3 standard deviations from the mean.
The following code does that:

```
## Identifying outliers
age_outliers <- filter(indo.contraceptives, abs(indo.contraceptives$w.age)>3)
## Checking how many outliers we have
nrow(age_outliers)
```

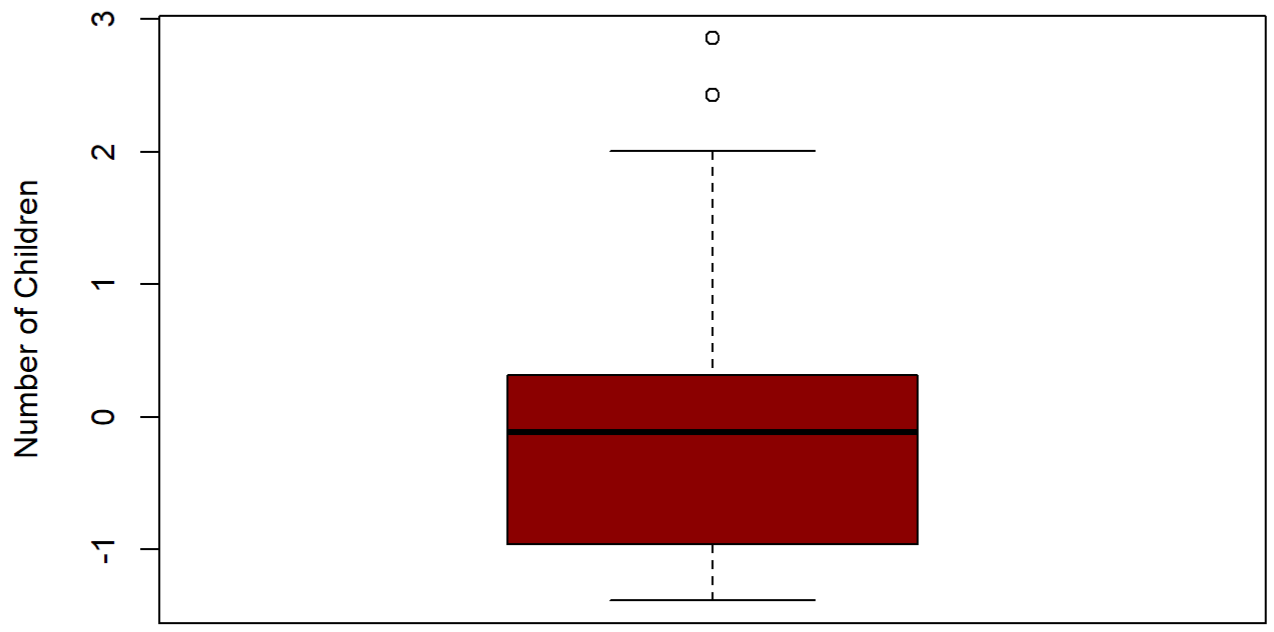
```
## [1] 0
```

```
## Identifying outliers
child_outliers <- filter(indo.contraceptives, abs(indo.contraceptives$num.children)>3)
## Checking how many outliers we have
nrow(child_outliers)
```

```
## [1] 18
```

We have 18 outliers in the number of children born per women feature. I will remove these outliers next.

```
## Removing outliers
indo.contraceptives<- filter(indo.contraceptives, abs(indo.contraceptives$num.children)<3)
## Re-exmining the box plot
boxplot(indo.contraceptives$num.children,
  col = "darkred",
  ylab = "Number of Children"
)
```



much better.

5. Dummy Coding:
We have ten variables in the dataset. Two of those are continues and have been normalized in the previous section. In this section, dummy coding will be performed on the categorical variables.

```r
## Dummy coding the w.education variable
indo.contraceptives<- indo.contraceptives%>%
  mutate(w.education.low = ifelse(w.education == "1" , 1, 0))
indo.contraceptives<- indo.contraceptives%>%
  mutate(w.education.mid = ifelse(w.education == "2" | w.education == "3", 1, 0))
## Dummy coding the h.education
indo.contraceptives<- indo.contraceptives%>%
  mutate(h.education.low = ifelse(h.education == "1" , 1, 0))
indo.contraceptives<- indo.contraceptives%>%
  mutate(h.education.mid = ifelse(h.education == "2" | h.education == "3", 1, 0))
## Since w.religion is already binary, I will not encode
## However, I will change the type to int
indo.contraceptives$w.religion<- as.integer(indo.contraceptives$w.religion)
## Same goes for the w.work
indo.contraceptives$w.work<- as.integer(indo.contraceptives$w.work)
## Dummy coding h.job
indo.contraceptives<- indo.contraceptives%>%
  mutate(h.job.first = ifelse(h.job == "1" , 1, 0))
indo.contraceptives<- indo.contraceptives%>%
  mutate(h.job.second = ifelse(h.job == "2" , 1, 0))
indo.contraceptives<- indo.contraceptives%>%
  mutate(h.job.third = ifelse(h.job == "3" , 1, 0))
## Dummy coding the living.index variable
indo.contraceptives<- indo.contraceptives%>%
  mutate(living.index.low = ifelse(living.index == "1" , 1, 0))
indo.contraceptives<- indo.contraceptives%>%
  mutate(living.index.mid = ifelse(living.index == "2" |living.index == "3" , 1, 0))
## Media is also binary, so I will just fix the type
indo.contraceptives$media<- as.integer(indo.contraceptives$media)
## Contraceptive is the target variable
indo.contraceptives$contraceptive<- as.integer(indo.contraceptives$contraceptive)
```

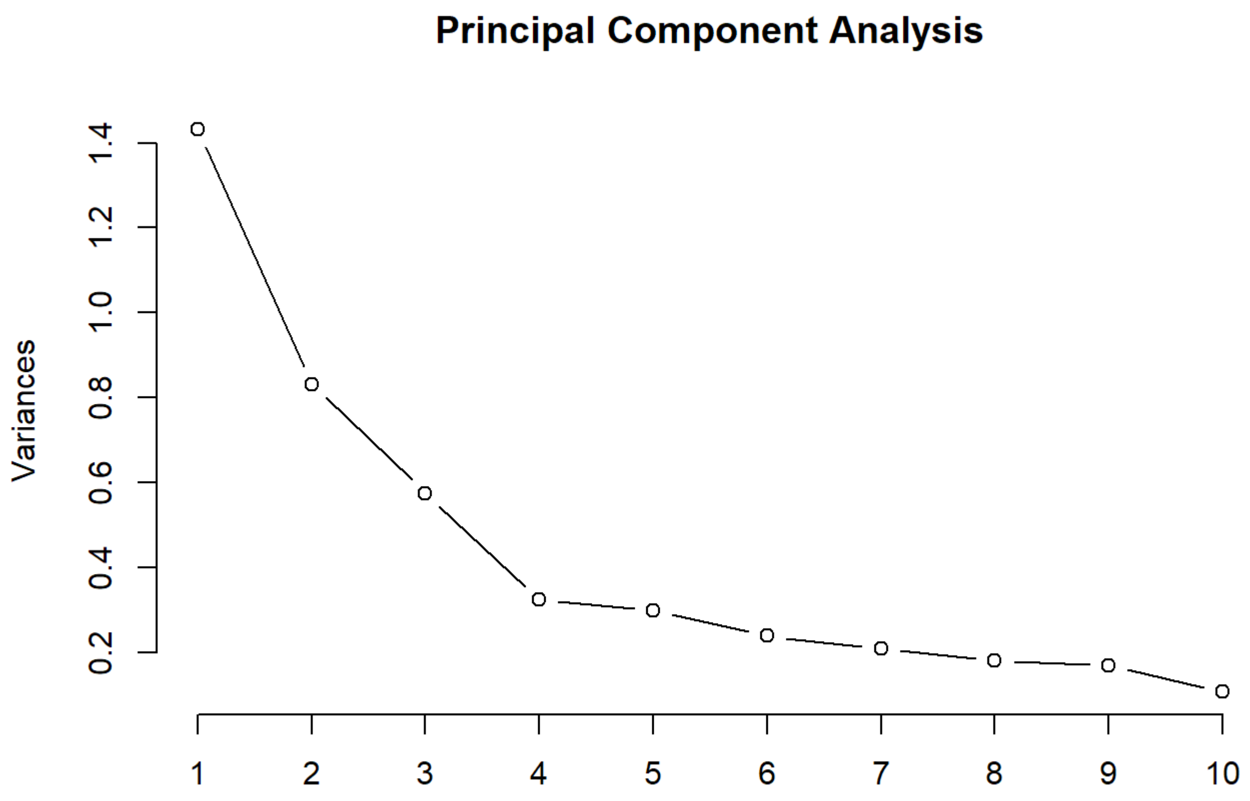
```r
## Removing un-needed columns, these are columns already encoded
for (i in 1:10) {
  if(is.factor(indo.contraceptives[,i])){
    indo.contraceptives<-indo.contraceptives[,-i]
    i<- i-1
  }
}
```

6. Identification of Principal Components(PCA):
It is not a good choice for this data set since most of the variables are categorical. For the purposes of demonstration it was implemented in the following code:

```r
## PCA on the encoded, scaled dataset.
A1 = prcomp(indo.contraceptives)
## Summary of the results
summary(A1)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6    PC7
## Standard deviation     1.1962 0.9119 0.7588 0.57037 0.54641 0.48916 0.4592
## Proportion of Variance 0.3135 0.1822 0.1262 0.07127 0.06541 0.05242 0.0462
## Cumulative Proportion  0.3135 0.4956 0.6218 0.69303 0.75844 0.81086 0.8571
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.42576 0.41200 0.32886 0.27468 0.22497 0.20062 0.14602
## Proportion of Variance 0.03971 0.03719 0.02369 0.01653 0.01109 0.00882 0.00467
## Cumulative Proportion  0.89678 0.93396 0.95765 0.97418 0.98527 0.99409 0.99876
##                           PC15
## Standard deviation     0.07535
## Proportion of Variance 0.00124
## Cumulative Proportion  1.00000
```

```
plot(A1, type="l", main = "Principal Component Analysis")
```



Modeling and Evaluation:

This is a classification problem where we are trying to classify women into one of three groups in terms of their usage of contraceptives; long term, short term, and none. Since we have more than two classes, it is a multi-nominal problem. The classification is done based on demographic data and media exposure.
Before modeling, I will partition the data for training and testing purposes.

I will devote 75% of the data for training and the remaining 25% for testing.

Training and Testing Subsets:

```
## Checking the number of rows before partitioning
nrow(indo.contraceptives)
```

```
## [1] 1455
```

```
## Partitioning
index <- sample(1:nrow(indo.contraceptives), floor(0.75 * nrow(indo.contraceptives)))
train <- indo.contraceptives[index,]
test <- indo.contraceptives[-index,]
## Checking the number of rows after partitioning
nrow(train)
```

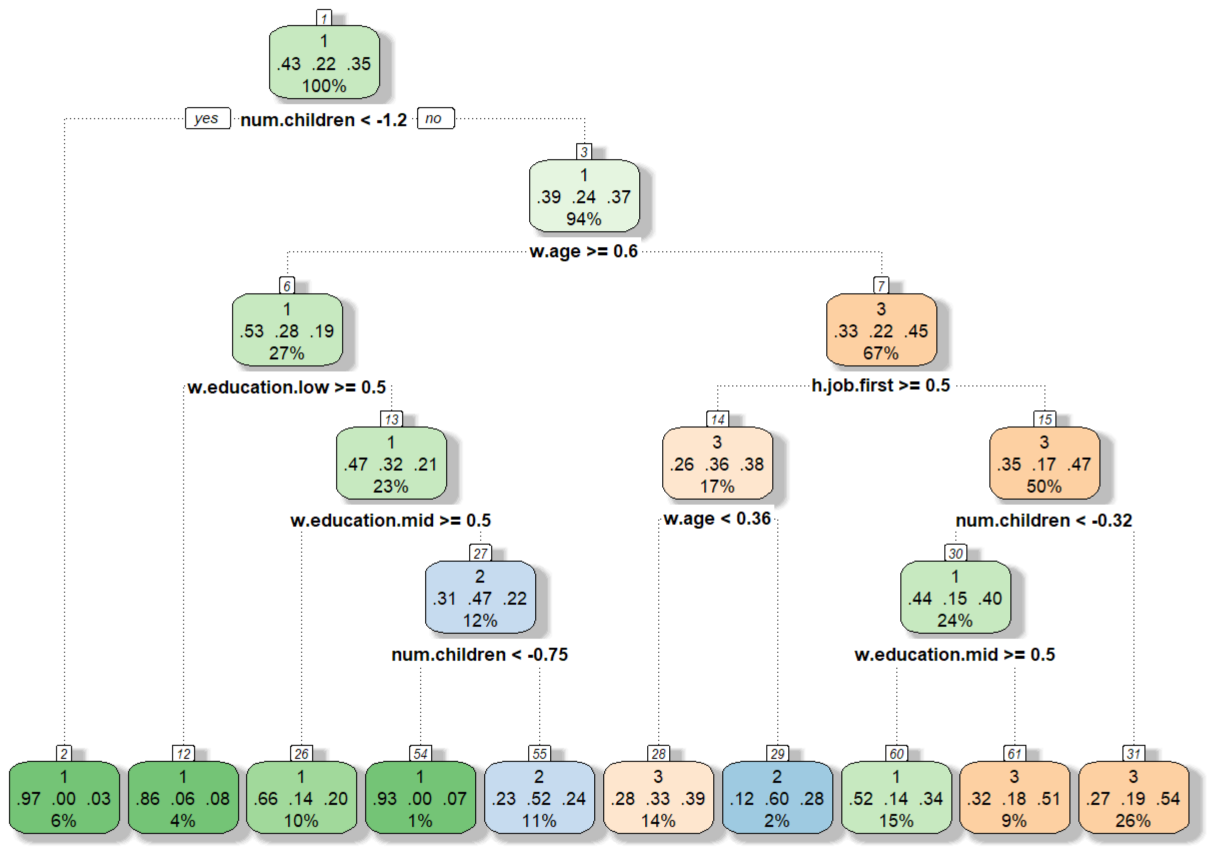
```
## [1] 1091
```

```
nrow(test)
```

```
## [1] 364
```

Looks good!
Next step, I will start building the models.
I have chosen the following three models:
1. Decision Tree:
Decision trees are a supervised machine learning technique, where the data is continuously split according to a certain parameter.
I have chosen to begin with decision trees because it has the advantage of being intuitive and easy to visualize.
## Decision Tree:

```
## Training the model
contraceptive.dt <- rpart(contraceptive ~. , data = train, method = "class")
## Visualizing the decision tree
fancyRpartPlot(contraceptive.dt)
```



Rattle 2021-Apr-28 00:54:36 Ghada

1 = no use

2 = long term

3= short term

It appears that the parameter used to make the initial split is the number of children born. In other words, the decision tree suggests that if a women has not had any children in the past, then she most likely is not using any contraceptives.

It also suggests that a women who is older than 38 is more likely to be using no contraceptives.

Wife education is the next indicator, followed by another split based on the number of children.

I will check the accuracy of the decision tree using testing data

```
## Evaluation using testing data
contraceptive.predict.1 <- predict(contraceptive.dt, test, type = "class")
## Confusiom matrix
table(contraceptive.predict.1, test$contraceptive)
```

```
##
## contraceptive.predict.1  1  2  3
##                       1 98 17 30
##                       2  6 19 12
##                       3 50 52 80
```

```
results <- contraceptive.predict.1 == test$contraceptive
## Print the accuracy
print(sprintf("The accuracy using hold out method: %.3f percent", (accuracy <- sum(results) / le
ngth(results))*100))
```

```
## [1] "The accuracy using hold out method: 54.121 percent"
```

It appears that the accuracy of the decision tree is not optimal (in the mid 50%)
The confusion matrix shows that there was a fair amount of miss-classification in the testing data.
The method used to evaluate the model is called holdout method, where a portion of the data is held out from the training and is used to evaluate the accuracy.
Now, I will try another method of validation called K-Fold Cross Validation.

```
set.seed(123)
form <- "contraceptive~ w.age+ num.children + w.religion + w.work + media + w.education.low+ w.e
ducation.mid+ h.education.low+ h.education.mid+ h.job.first+ h.job.second+ h.job.third+ living.i
ndex.low+ living.index.mid"
folds <- split(indo.contraceptives, cut(sample(1:nrow(indo.contraceptives)),10))
errs <- rep(NA, length(folds))
for (i in 1:length(folds)) {
 test <- ldply(folds[i], data.frame)
 train <- ldply(folds[-i], data.frame)
 tmp.model <- rpart(form , train, method = "class")
 tmp.predict <- predict(tmp.model, newdata = test, type = "class")
 conf.mat <- table(test$contraceptive, tmp.predict)
 errs[i] <- 1-sum(diag(conf.mat))/sum(conf.mat)
}
print(sprintf("average error using k-fold cross-validation: %.3f percent", 100*mean(errs)))
```

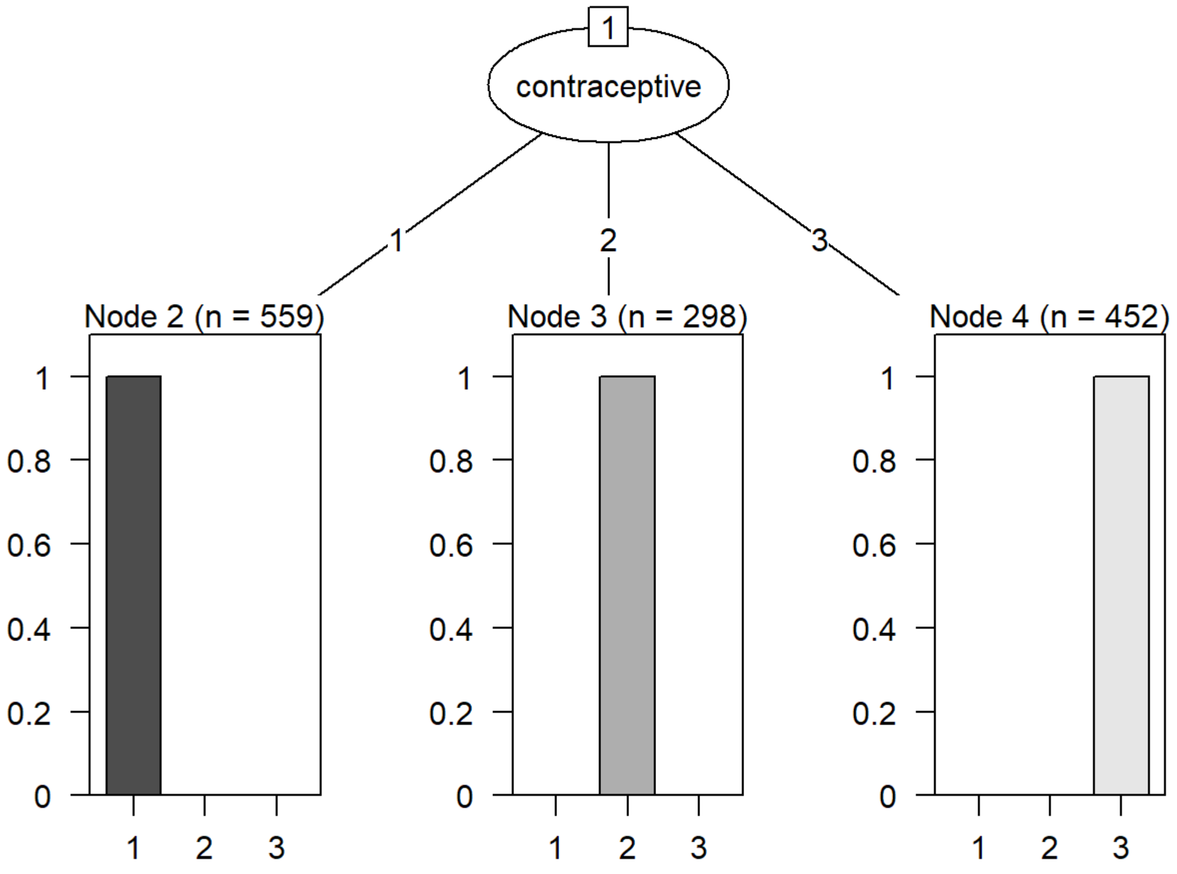
```
## [1] "average error using k-fold cross-validation: 44.811 percent"
```

The average error with k-fold cross-validation is too high which validates the previous result.
I will try a different implementation of decision tree model next.

```
## Setting the target variable as factor
train$contraceptive<-as.factor(train$contraceptive)
test$contraceptive<-as.factor(test$contraceptive)
## Training the model
dt.2 <- C5.0(train[,-1],train$contraceptive)
summary(dt.2)
```

```
##
## Call:
## C5.0.default(x = train[, -1], y = train$contraceptive)
##
##
## C5.0 [Release 2.07 GPL Edition]      Wed Apr 28 00:54:39 2021
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 1309 cases (16 attributes) from undefined.data
##
## Decision tree:
##
## contraceptive = 1: 1 (559)
## contraceptive = 2: 2 (298)
## contraceptive = 3: 3 (452)
##
##
## Evaluation on training data (1309 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       3    0( 0.0%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##     559                (a): class 1
##           298          (b): class 2
##                 452    (c): class 3
##
##
##  Attribute usage:
##
##  100.00% contraceptive
##
##
## Time: 0.0 secs
```

```
plot(dt.2)
```



```
## Evaluating the model on the testing data
dt.pred<- predict(dt.2, test[,-1])
CrossTable(test$contraceptive, dt.pred,
           prop.chisq = FALSE,
           prop.c = FALSE, prop.r = FALSE, dnn = c('actual method', 'predicted method'))
```

```
## 
## 
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
## 
## 
## Total Observations in Table:  146 
## 
## 
##               | predicted method 
## actual method |         1 |         2 |         3 | Row Total | 
## --------------|-----------|-----------|-----------|-----------|
##             1 |        60 |         0 |         0 |        60 | 
##               |     0.411 |     0.000 |     0.000 |           | 
## --------------|-----------|-----------|-----------|-----------|
##             2 |         0 |        32 |         0 |        32 | 
##               |     0.000 |     0.219 |     0.000 |           | 
## --------------|-----------|-----------|-----------|-----------|
##             3 |         0 |         0 |        54 |        54 | 
##               |     0.000 |     0.000 |     0.370 |           | 
## --------------|-----------|-----------|-----------|-----------|
##  Column Total |        60 |        32 |        54 |       146 | 
## --------------|-----------|-----------|-----------|-----------|
## 
## 
```

```
## Confusion matrix
table(test$contraceptive, dt.pred)
```

```
##    dt.pred
##      1  2  3
##   1 60  0  0
##   2  0 32  0
##   3  0  0 54
```

```
results <- dt.pred == test$contraceptive
print(sprintf("The accuracy using the new decision tree: %.3f percent", (accuracy <- sum(result
s) / length(results))*100))
```

```
## [1] "The accuracy using the new decision tree: 100.000 percent"
```

The accuracy increased to 100%. But that is not exactly good news, it raise the concern of over fitting.
Let us try k-fold cross validation next:

```
set.seed(123)
form <- "contraceptive~ w.age+ num.children + w.religion + w.work + media + w.education.low+ w.e
ducation.mid+ h.education.low+ h.education.mid+ h.job.first+ h.job.second+ h.job.third+ living.i
ndex.low+ living.index.mid"
folds <- split(indo.contraceptives, cut(sample(1:nrow(indo.contraceptives)),10))
errs <- rep(NA, length(folds))
for (i in 1:length(folds)) {
 test <- ldply(folds[i], data.frame)
 train <- ldply(folds[-i], data.frame)
 ## Setting the target variable as factor
 train[,7]<-as.factor(train[,7])
 test[,7]<-as.factor(test[,7])
 tmp.model <- C5.0(train[,-1],train$contraceptive)
 tmp.predict <- predict(tmp.model, test[,-1])
 conf.mat <- table(test$contraceptive, tmp.predict)
 errs[i] <- 1-sum(diag(conf.mat))/sum(conf.mat)
}
print(sprintf("average error using k-fold cross-validation: %.3f percent", 100*mean(errs)))
```

```
## [1] "average error using k-fold cross-validation: 0.000 percent"
```

The average error is 0 which confirms the previous finding.

KNN:

I will now try to use KNN with k = sqrt(the number of features).
I will experiment with this first, and I will tune the model in later sections.

```
## Run knn function
k.value<- round(sqrt(length(indo.contraceptives)),digits = 0)
contraceptive.knn <- knn(train[,c(-1)],test[,c(-1)], cl= train$contraceptive,k= k.value)

## Create confusion matrix
 tab <- table(contraceptive.knn,test$contraceptive)
 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
 accuracy(tab)
```

```
## [1] 95.89041
```

```
## Print the accuracy
print(sprintf("The accuracy of knn using hold out method: %.3f percent", accuracy(tab)))
```

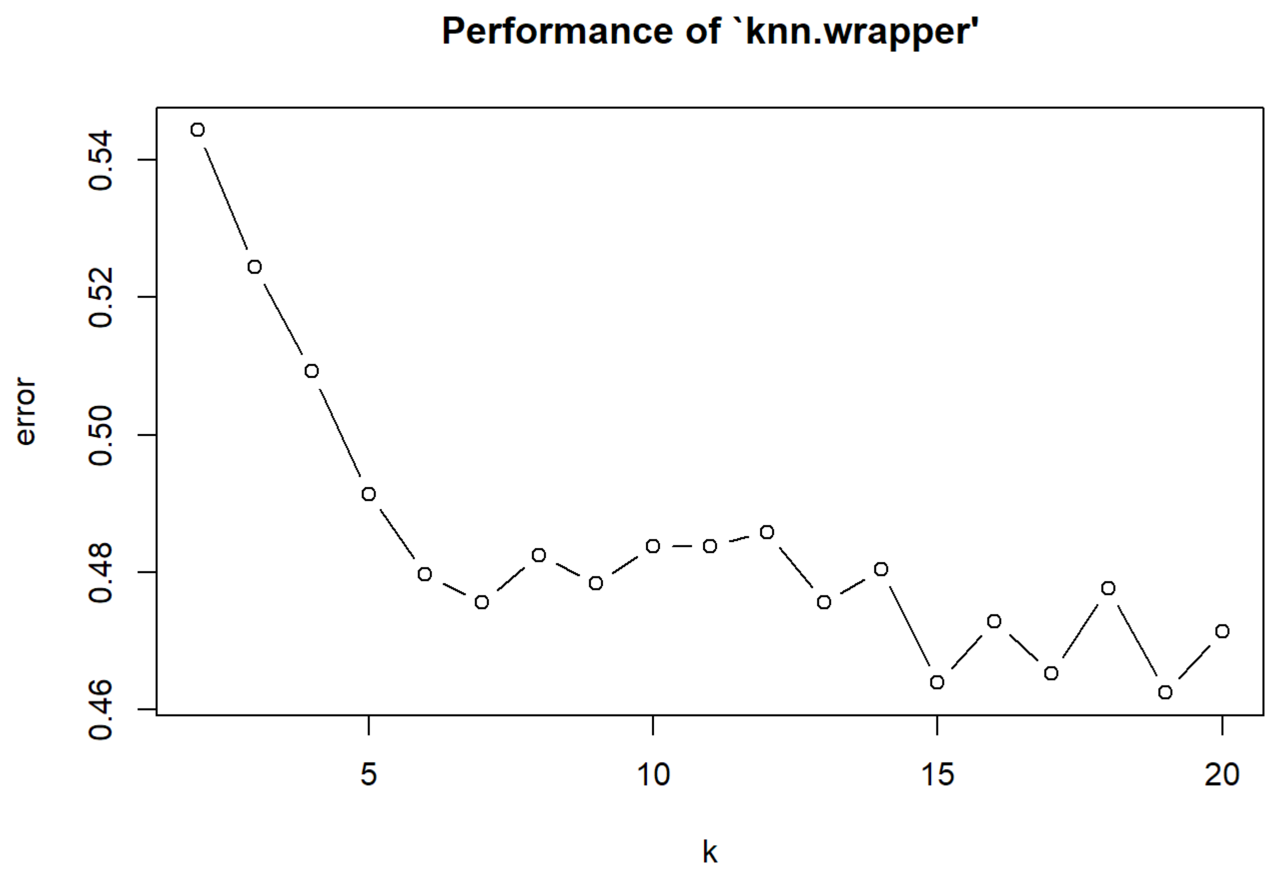
```
## [1] "The accuracy of knn using hold out method: 95.890 percent"
```

The accuracy of Knn is much better than the decision tree in the first implementation!
Let us see if we can improve the model further by tuning the parameters using cross validation.

```
## Model tuning
indo.contraceptives$contraceptive<- as.factor(indo.contraceptives$contraceptive)
knn.cross <- tune.knn(x = indo.contraceptives[,-6], y = indo.contraceptives[,6],k = 2:20,tunecon
trol=tune.control(sampling = "cross"), cross=10)
summary(knn.cross)
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   k
##  19
##
## - best performance: 0.4624846
##
## - Detailed performance results:
##     k     error dispersion
## 1   2 0.5443127 0.05037338
## 2   3 0.5243977 0.06014988
## 3   4 0.5092537 0.06664180
## 4   5 0.4913982 0.05665438
## 5   6 0.4796930 0.05731532
## 6   7 0.4755739 0.06801666
## 7   8 0.4824752 0.05587469
## 8   9 0.4783845 0.07125518
## 9  10 0.4838120 0.06384126
## 10 11 0.4838026 0.06925940
## 11 12 0.4858385 0.07853970
## 12 13 0.4755645 0.04838452
## 13 14 0.4804015 0.06527359
## 14 15 0.4639017 0.07083791
## 15 16 0.4728389 0.06292811
## 16 17 0.4652622 0.06104292
## 17 18 0.4776004 0.07429881
## 18 19 0.4624846 0.06648566
## 19 20 0.4714454 0.05174227
```

```
plot(knn.cross)
```



It looks like the optimal value for k is 18. I will run the model again, and see if improved accuracy is achieved.

```
## Training the model with k =18
contraceptive.knn <- knn(train[,c(-1)],test[,c(-1)],cl=train$contraceptive,k= 18)

## Creating confusion matrix
 tab <- table(contraceptive.knn,test$contraceptive)
 accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
 accuracy(tab)
```

```
## [1] 97.26027
```

```
print(sprintf("The accuracy of knn after tunning: %.3f percent", accuracy(tab)))
```

```
## [1] "The accuracy of knn after tunning: 97.260 percent"
```

Indeed!
The accuracy is now around 98%!
Very good, Now I will try another model

Neural Network:

```
## Training the model
contraceptive.nnet <- nnet(contraceptive ~ . , data = train[,c(-1)], size = 1)
```

```
## # weights:  21
## initial  value 1521.636434
## iter  10 value 1342.712564
## iter  20 value 1331.179750
## iter  30 value 1327.657801
## iter  40 value 1321.504922
## iter  50 value 1319.956615
## iter  60 value 1319.633058
## iter  70 value 1318.719397
## iter  80 value 1318.651017
## iter  90 value 1318.635551
## iter 100 value 1318.630909
## final  value 1318.630909
## stopped after 100 iterations
```

```
## Classify the testing data
pred.nn<-predict(contraceptive.nnet, newdata = test[,c(-1)], type = "class")
## Confusion matrix
table(pred.nn,test$contraceptive)
```

```
##
## pred.nn  1  2  3
##       1 35 25 31
##       3 25  7 23
```

```
## Calculate accuracy
tab <- table(pred.nn,test$contraceptive)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

```
## [1] 28.76712
```

```
print(sprintf("The accuracy of NN with a single node: %.3f percent", accuracy(tab)))
```

```
## [1] "The accuracy of NN with a single node: 28.767 percent"
```

The accuracy of the neural network with one hidden node is not satisfactory at all. I will try to increase the number of nodes and see if that will help improve the accuracy.

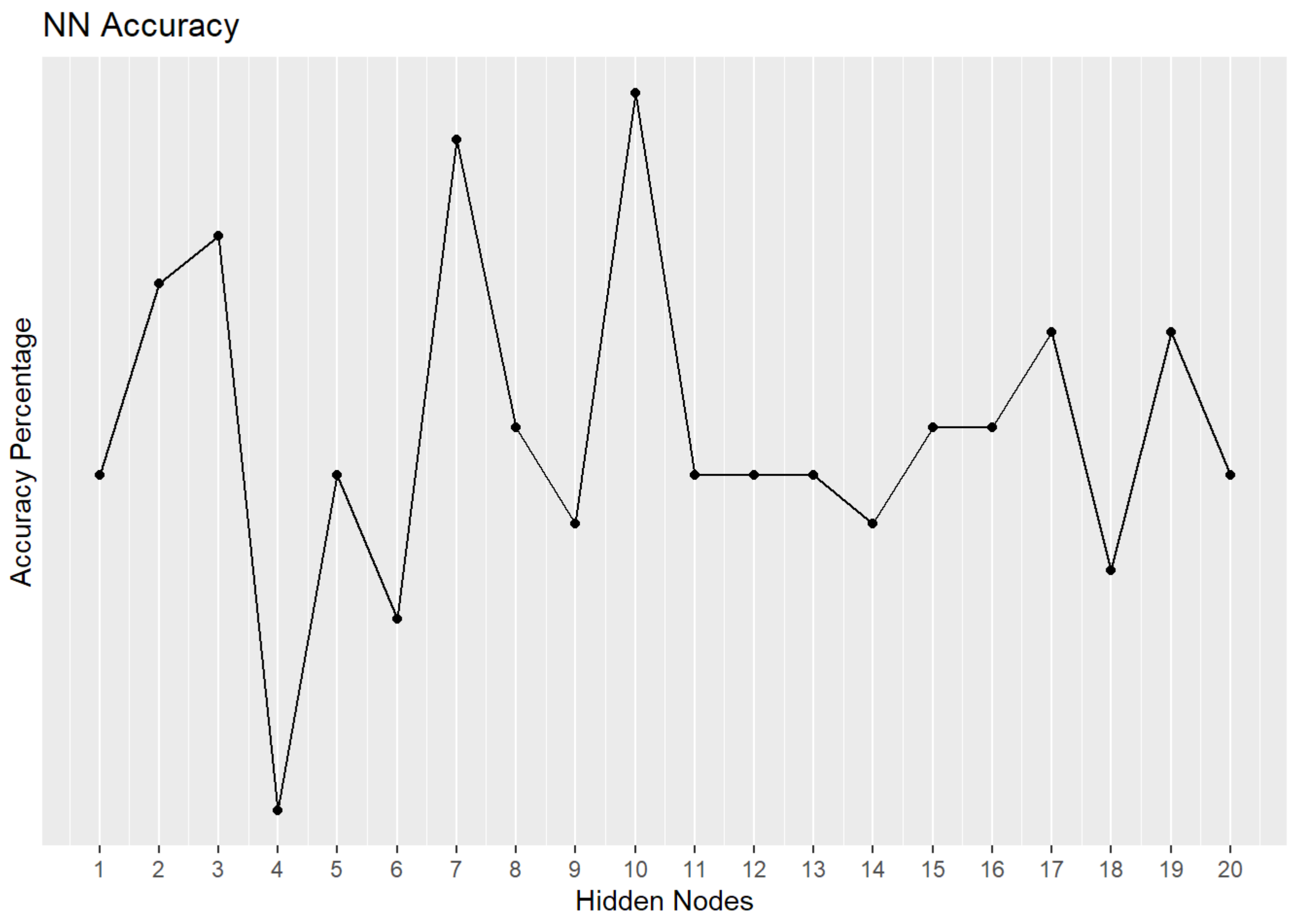
```r
## Data frame to hold NN accuracy and number of correct predictions
contraceptive_NN_accuracy <- data.frame(nodes = c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,1
9,20), correct_predictions = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0))

## Looping for 20 nodes
for (i in 1:20) {
  set.seed(250)
  contraceptive.NN.i <- nnet(contraceptive ~ ., data = train[,c(-1)], size=i)
  contraceptive.NN.Predictions.i<- predict(contraceptive.NN.i,     test[,c(-1)], type="class")
  contraceptive_NN_accuracy[i,2] <- sum(contraceptive.NN.Predictions.i == test$contraceptive)
}
```

```
## # weights:  21
## initial  value 1564.854990
## iter  10 value 1310.786095
## iter  20 value 1264.331108
## iter  30 value 1262.539290
## iter  40 value 1262.524736
## final  value 1262.524628
## converged
## # weights:  39
## initial  value 1518.934444
## iter  10 value 1339.840200
## iter  20 value 1304.081117
## iter  30 value 1281.049214
## iter  40 value 1263.226044
## iter  50 value 1262.691736
## iter  60 value 1255.984165
## iter  70 value 1230.356973
## iter  80 value 1208.207188
## iter  90 value 1201.149976
## iter 100 value 1195.899836
## final  value 1195.899836
## stopped after 100 iterations
## # weights:  57
## initial  value 1695.623040
## iter  10 value 1306.815991
## iter  20 value 1216.102972
## iter  30 value 1193.130394
## iter  40 value 1175.558865
## iter  50 value 1167.967892
## iter  60 value 1163.639842
## iter  70 value 1160.985227
## iter  80 value 1155.024868
## iter  90 value 1145.852320
## iter 100 value 1142.334267
## final  value 1142.334267
## stopped after 100 iterations
## # weights:  75
## initial  value 1461.857330
## iter  10 value 1297.533558
## iter  20 value 1188.972979
## iter  30 value 1139.514393
## iter  40 value 1121.950239
## iter  50 value 1108.913716
## iter  60 value 1104.831509
## iter  70 value 1099.565637
## iter  80 value 1092.679417
## iter  90 value 1086.774769
## iter 100 value 1070.804990
## final  value 1070.804990
## stopped after 100 iterations
## # weights:  93
## initial  value 2324.574561
## iter  10 value 1347.148257
## iter  20 value 1251.025374
## iter  30 value 1207.212549
## iter  40 value 1172.966150
## iter  50 value 1127.848553
## iter  60 value 1114.517407
## iter  70 value 1103.341786
## iter  80 value 1098.521334
## iter  90 value 1093.754843
## iter 100 value 1087.677353
## final  value 1087.677353
## stopped after 100 iterations
## # weights:  111
## initial  value 1458.047043
## iter  10 value 1241.139137
## iter  20 value 1133.573802
## iter  30 value 1097.815277
## iter  40 value 1076.845441
## iter  50 value 1065.547898
## iter  60 value 1058.296736
## iter  70 value 1050.689359
## iter  80 value 1040.659154
## iter  90 value 1034.490707
## iter 100 value 1031.515178
## final  value 1031.515178
## stopped after 100 iterations
## # weights:  129
## initial  value 1751.330898
## iter  10 value 1345.701312
## iter  20 value 1227.604283
## iter  30 value 1140.536043
## iter  40 value 1115.548644
## iter  50 value 1098.370966
## iter  60 value 1095.607105
## iter  70 value 1093.095462
## iter  80 value 1088.163888
## iter  90 value 1084.077568
## iter 100 value 1074.634976
## final  value 1074.634976
## stopped after 100 iterations
## # weights:  147
## initial  value 1761.168668
## iter  10 value 1286.508896
## iter  20 value 1178.148317
## iter  30 value 1120.132363
## iter  40 value 1095.160417
## iter  50 value 1076.276014
## iter  60 value 1060.490076
## iter  70 value 1050.209975
## iter  80 value 1039.828778
## iter  90 value 1026.269652
## iter 100 value 1012.893741
## final  value 1012.893741
## stopped after 100 iterations
## # weights:  165
## initial  value 1629.941913
## iter  10 value 1268.460601
## iter  20 value 1148.913393
## iter  30 value 1099.821700
## iter  40 value 1068.549145
## iter  50 value 1048.248118
## iter  60 value 1029.607895
## iter  70 value 1012.759790
## iter  80 value 996.834389
## iter  90 value 980.348849
## iter 100 value 964.351988
## final  value 964.351988
## stopped after 100 iterations
## # weights:  183
## initial  value 1411.965880
## iter  10 value 1210.615198
## iter  20 value 1118.054446
## iter  30 value 1068.792902
## iter  40 value 1042.358051
## iter  50 value 1019.821490
## iter  60 value 1001.353003
## iter  70 value 983.284720
## iter  80 value 963.055161
## iter  90 value 937.208608
## iter 100 value 927.751731
## final  value 927.751731
## stopped after 100 iterations
## # weights:  201
## initial  value 1668.046146
## iter  10 value 1267.832975
## iter  20 value 1173.239722
## iter  30 value 1125.625231
## iter  40 value 1091.279001
## iter  50 value 1065.940243
## iter  60 value 1045.735492
## iter  70 value 1037.055075
## iter  80 value 1026.052313
## iter  90 value 1018.358591
## iter 100 value 1012.841295
## final  value 1012.841295
## stopped after 100 iterations
## # weights:  219
## initial  value 2174.145168
## iter  10 value 1347.505334
## iter  20 value 1322.230442
## iter  30 value 1237.235469
## iter  40 value 1165.773619
## iter  50 value 1135.138880
## iter  60 value 1123.754254
## iter  70 value 1115.177780
## iter  80 value 1111.321765
## iter  90 value 1109.711786
## iter 100 value 1106.325922
## final  value 1106.325922
## stopped after 100 iterations
## # weights:  237
## initial  value 1440.228693
## iter  10 value 1196.157707
## iter  20 value 1133.767658
## iter  30 value 1097.924707
## iter  40 value 1061.395176
## iter  50 value 1033.809734
## iter  60 value 1010.440091
## iter  70 value 978.776234
## iter  80 value 956.789733
## iter  90 value 942.892698
## iter 100 value 931.024007
## final  value 931.024007
## stopped after 100 iterations
## # weights:  255
## initial  value 1579.874760
## iter  10 value 1211.249043
## iter  20 value 1139.652391
## iter  30 value 1091.428993
## iter  40 value 1043.345477
## iter  50 value 1005.008496
## iter  60 value 983.312507
## iter  70 value 958.035306
## iter  80 value 941.970585
## iter  90 value 929.003494
## iter 100 value 919.322555
## final  value 919.322555
## stopped after 100 iterations
## # weights:  273
## initial  value 1579.804271
## iter  10 value 1220.720324
## iter  20 value 1150.943155
## iter  30 value 1100.987998
## iter  40 value 1045.988773
## iter  50 value 998.161441
## iter  60 value 970.855782
## iter  70 value 938.938208
## iter  80 value 917.622416
## iter  90 value 902.411517
## iter 100 value 889.472082
## final  value 889.472082
## stopped after 100 iterations
## # weights:  291
## initial  value 2984.747159
## iter  10 value 1232.269244
## iter  20 value 1149.367387
## iter  30 value 1096.583694
## iter  40 value 1043.889599
## iter  50 value 1004.615457
## iter  60 value 978.292875
## iter  70 value 957.525555
## iter  80 value 929.992089
## iter  90 value 912.207272
## iter 100 value 898.447728
## final  value 898.447728
## stopped after 100 iterations
## # weights:  309
## initial  value 3373.429304
## iter  10 value 1366.547520
## iter  20 value 1279.676693
## iter  30 value 1164.474106
## iter  40 value 1139.408724
## iter  50 value 1129.179931
## iter  60 value 1115.956317
## iter  70 value 1102.746607
## iter  80 value 1088.258258
## iter  90 value 1078.215436
## iter 100 value 1055.161443
## final  value 1055.161443
## stopped after 100 iterations
## # weights:  327
## initial  value 1869.103171
## iter  10 value 1214.787010
## iter  20 value 1124.825038
## iter  30 value 1072.425598
## iter  40 value 1020.011830
## iter  50 value 975.966558
## iter  60 value 934.268343
## iter  70 value 901.083425
## iter  80 value 876.513138
## iter  90 value 855.018084
## iter 100 value 840.065228
## final  value 840.065228
## stopped after 100 iterations
## # weights:  345
## initial  value 1532.457261
## iter  10 value 1194.110297
## iter  20 value 1135.158341
## iter  30 value 1091.809147
## iter  40 value 1043.070055
## iter  50 value 978.248819
## iter  60 value 944.597576
## iter  70 value 919.796640
## iter  80 value 893.865781
## iter  90 value 876.811086
## iter 100 value 850.708144
## final  value 850.708144
## stopped after 100 iterations
## # weights:  363
## initial  value 1770.378863
## iter  10 value 1189.304118
## iter  20 value 1139.563329
## iter  30 value 1101.984202
## iter  40 value 1058.273771
## iter  50 value 998.776683
## iter  60 value 956.219203
## iter  70 value 910.771596
## iter  80 value 876.471435
## iter  90 value 849.062401
## iter 100 value 832.699472
## final  value 832.699472
## stopped after 100 iterations
```

```
## Plotting the different NN settings
plot.data.nn<- mutate(contraceptive_NN_accuracy, accuracy = round((correct_predictions/148)*100,
2))

## Plotting the different number of nodes
ggplot(data = plot.data.nn, aes(x=nodes, y=accuracy)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = c(1:20)) +
  scale_y_continuous(breaks = c(84:91)) +
  ggtitle("NN Accuracy") +
  xlab("Hidden Nodes") +
  ylab("Accuracy Percentage")
```



It looks like the best accuracy can be achieved with 7 nodes.

```
## Training the model
contraceptive.nnet <- nnet(contraceptive ~ . , data = train[,c(-1)], size = 7)
```

```
## # weights:  129
## initial  value 1782.213964
## iter  10 value 1389.229212
## iter  20 value 1308.748091
## iter  30 value 1262.697820
## iter  40 value 1250.699178
## iter  50 value 1213.515515
## iter  60 value 1202.193974
## iter  70 value 1170.520001
## iter  80 value 1131.918231
## iter  90 value 1124.193861
## iter 100 value 1122.346614
## final  value 1122.346614
## stopped after 100 iterations
```

```
## Classify the testing data
pred.nn<-predict(contraceptive.nnet, newdata = test[,c(-1)], type = "class")
## Confusion matrix
table(pred.nn,test$contraceptive)
```

```
##
## pred.nn  1  2  3
##       1 27  6  8
##       2  6 11 12
##       3 27 15 34
```

```
## Calculate accuracy
tab <- table(pred.nn,test$contraceptive)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

```
## [1] 49.31507
```

```
print(sprintf("The accuracy of NN with 7 nodes: %.3f percent", accuracy(tab)))
```

```
## [1] "The accuracy of NN with 7 nodes: 49.315 percent"
```

The accuracy indeed increased after adding more nodes. However, it is still not impressive. kNN wins so far.
I will try to test the neural network with cross validation.

```r
set.seed(123)
folds <- split(indo.contraceptives, cut(sample(1:nrow(indo.contraceptives)),10))
errs <- rep(NA, length(folds))
for (i in 1:length(folds)) {
 test <- ldply(folds[i], data.frame)
 train <- ldply(folds[-i], data.frame)
 ## Setting the target variable as factor
 train[,7]<-as.factor(train[,7])
 test[,7]<-as.factor(test[,7])
 tmp.model <- nnet(contraceptive~., data = train[,-1], size = 7)
 tmp.predict <- predict(tmp.model, newdata = test[,-1], type = "class")
 conf.mat <- table(test$contraceptive, tmp.predict)
 errs[i] <- 1-sum(diag(conf.mat))/sum(conf.mat)
}
```

```
## # weights:  129
## initial  value 1648.769445
## iter  10 value 1271.433002
## iter  20 value 1192.795584
## iter  30 value 1140.413640
## iter  40 value 1106.763923
## iter  50 value 1091.435337
## iter  60 value 1079.272157
## iter  70 value 1066.004182
## iter  80 value 1049.790168
## iter  90 value 1038.462999
## iter 100 value 1035.318766
## final  value 1035.318766
## stopped after 100 iterations
## # weights:  129
## initial  value 1628.857927
## iter  10 value 1271.516512
## iter  20 value 1174.282734
## iter  30 value 1121.544356
## iter  40 value 1097.383723
## iter  50 value 1082.069251
## iter  60 value 1072.320271
## iter  70 value 1062.843147
## iter  80 value 1054.015767
## iter  90 value 1048.696213
## iter 100 value 1042.837325
## final  value 1042.837325
## stopped after 100 iterations
## # weights:  129
## initial  value 1592.884791
## iter  10 value 1242.188502
## iter  20 value 1165.268725
## iter  30 value 1124.735374
## iter  40 value 1093.489879
## iter  50 value 1074.691583
## iter  60 value 1068.463735
## iter  70 value 1059.489706
## iter  80 value 1048.896629
## iter  90 value 1032.871544
## iter 100 value 1015.392355
## final  value 1015.392355
## stopped after 100 iterations
## # weights:  129
## initial  value 1571.675347
## iter  10 value 1279.201285
## iter  20 value 1182.155613
## iter  30 value 1141.713583
## iter  40 value 1115.803967
## iter  50 value 1099.189869
## iter  60 value 1090.023097
## iter  70 value 1079.846933
## iter  80 value 1066.644441
## iter  90 value 1053.437870
## iter 100 value 1039.842704
## final  value 1039.842704
## stopped after 100 iterations
## # weights:  129
## initial  value 1640.212799
## iter  10 value 1264.920532
## iter  20 value 1172.587333
## iter  30 value 1131.711648
## iter  40 value 1102.281715
## iter  50 value 1079.152265
## iter  60 value 1070.003795
## iter  70 value 1064.356912
## iter  80 value 1052.381518
## iter  90 value 1041.233703
## iter 100 value 1034.519545
## final  value 1034.519545
## stopped after 100 iterations
## # weights:  129
## initial  value 1551.101172
## iter  10 value 1294.878455
## iter  20 value 1196.568752
## iter  30 value 1140.391081
## iter  40 value 1113.294105
## iter  50 value 1093.780872
## iter  60 value 1081.701096
## iter  70 value 1071.812444
## iter  80 value 1064.217551
## iter  90 value 1060.252501
## iter 100 value 1055.164995
## final  value 1055.164995
## stopped after 100 iterations
## # weights:  129
## initial  value 1472.751698
## iter  10 value 1255.911185
## iter  20 value 1164.369596
## iter  30 value 1119.344289
## iter  40 value 1096.781320
## iter  50 value 1081.899778
## iter  60 value 1064.769634
## iter  70 value 1056.166599
## iter  80 value 1047.898052
## iter  90 value 1038.297697
## iter 100 value 1032.424999
## final  value 1032.424999
## stopped after 100 iterations
## # weights:  129
## initial  value 1517.047082
## iter  10 value 1247.837501
## iter  20 value 1188.424009
## iter  30 value 1152.089634
## iter  40 value 1132.783776
## iter  50 value 1110.375894
## iter  60 value 1093.752653
## iter  70 value 1089.622379
## iter  80 value 1085.513748
## iter  90 value 1079.807300
## iter 100 value 1070.090754
## final  value 1070.090754
## stopped after 100 iterations
## # weights:  129
## initial  value 1718.677223
## iter  10 value 1397.285283
## iter  20 value 1397.190190
## iter  30 value 1371.823613
## iter  40 value 1337.423125
## iter  50 value 1333.591857
## iter  60 value 1326.715981
## iter  70 value 1300.998120
## iter  80 value 1282.347290
## iter  90 value 1272.941262
## iter 100 value 1269.346987
## final  value 1269.346987
## stopped after 100 iterations
## # weights:  129
## initial  value 1461.803534
## iter  10 value 1234.789238
## iter  20 value 1172.238299
## iter  30 value 1124.292067
## iter  40 value 1100.429708
## iter  50 value 1083.689938
## iter  60 value 1073.324926
## iter  70 value 1061.486714
## iter  80 value 1050.849528
## iter  90 value 1046.867543
## iter 100 value 1044.764797
## final  value 1044.764797
## stopped after 100 iterations
```

```
print(sprintf("average error using k-fold cross-validation: %.3f percent", 100*mean(errs)))
```

```
## [1] "average error using k-fold cross-validation: 50.447 percent"
```

The average error is high!

Performance Improvement:

In this section, I will try to improve the performance further by using ensembles.
First I will use bagging.
Bagging:

```
## Training the model
contraceptive.bagging <- randomForest(contraceptive ~ . , data = train[,c(-1)], mtry = ncol(trai
n)- 1)
## Evaluation
predict.bagging<- predict(contraceptive.bagging, newdata = test[,c(-1)], type = "class")
## Confusion matrix
table(predict.bagging,test$contraceptive)
```

```
##
## predict.bagging  1  2  3
##               1 30 13 23
##               2  8  8 11
##               3 22 11 20
```

```
## Calculate accuracy
tab <- table(predict.bagging,test$contraceptive)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

```
## [1] 39.72603
```

```
print(sprintf("The accuracy of bagging: %.3f percent", accuracy(tab)))
```

```
## [1] "The accuracy of bagging: 39.726 percent"
```

```
##Confusion matrix
##mean(test$contraceptive == predict(contraceptive.bagging, newdata = test[,c(-1)], type = "clas
s"))
```

Next, I will try bagging with 25 bags.

```
RNGversion("3.5.2")
set.seed(300)
## Training the model
mybag <- bagging(contraceptive ~ ., data = train[,c(-1)], nbagg = 25)
## Making predictions
bagging.Prediction<- predict(mybag, test[,c(-1)])
tab<-table(bagging.Prediction, test$contraceptive)
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

```
## [1] 44.52055
```

```
print(sprintf("The accuracy of bagging with 25 bags: %.3f percent", accuracy(tab)))
```

```
## [1] "The accuracy of bagging with 25 bags: 44.521 percent"
```

Both implementations of bagging give similar result.

Ensemble:
I will now combine the models, creating an ensemble model using majority vote:

```
## Creating a function to make prediction for each model
## First decision tree
get_dt1_pred<- function (x){
  return(predict(contraceptive.dt, x , type = "class"))
}
## Second decision tree
get_dt2_pred<- function (x){
  return(predict(dt.2, x))
}
## Knn
get_knn_pred<- function (x){
 return(knn(train[,c(-1)],x,cl=train$contraceptive,k= 17))
}
## Neural network
get_nn_pred<- function (x){
 return(predict(contraceptive.nnet, newdata = x, type = "class"))
}

## Ensemble method
ensemble.model.predictions <- function(x){
  ## Getting all the predictions from the previous models
  predictions.all<- data.frame(get_dt1_pred(x),get_dt2_pred(x),get_knn_pred(x),get_nn_pred(x))
  ## Taking the majority vote
  predictions.all <- predictions.all %>%
    mutate(ensemble = apply(predictions.all[,1:length(predictions.all)], 1, mfv))
  names(predictions.all)<- c("decision tree1","decision tree2","kNN","NN","ensemble")
  return(predictions.all)
}
ensemble.model.predictions(test[,-1])
```

	decision tree1 <fct>	decision tree2 <fct>	kNN <fct>	NN <chr>	ensemble <named list>
1	1	1	1	3	<chr [1]>
2	1	1	1	1	<chr [1]>
3	1	1	1	3	<chr [1]>
4	2	1	1	2	<chr [2]>
5	1	1	1	1	<chr [1]>
6	1	1	1	1	<chr [1]>
7	1	1	1	1	<chr [1]>

	decision tree1 <fct>	decision tree2 <fct>	kNN <fct>	NN <chr>	ensemble <named list>
8	3	1	1	3	<chr [2]>
9	1	1	1	1	<chr [1]>
10	3	1	1	3	<chr [2]>

1-10 of 146 rows Previous **1** 2 3 4 5 6 ... 15 Next

I used the ensemble on the testing data. Now, I will evaluate the accuracy of the ensemble prediction against each individual model.

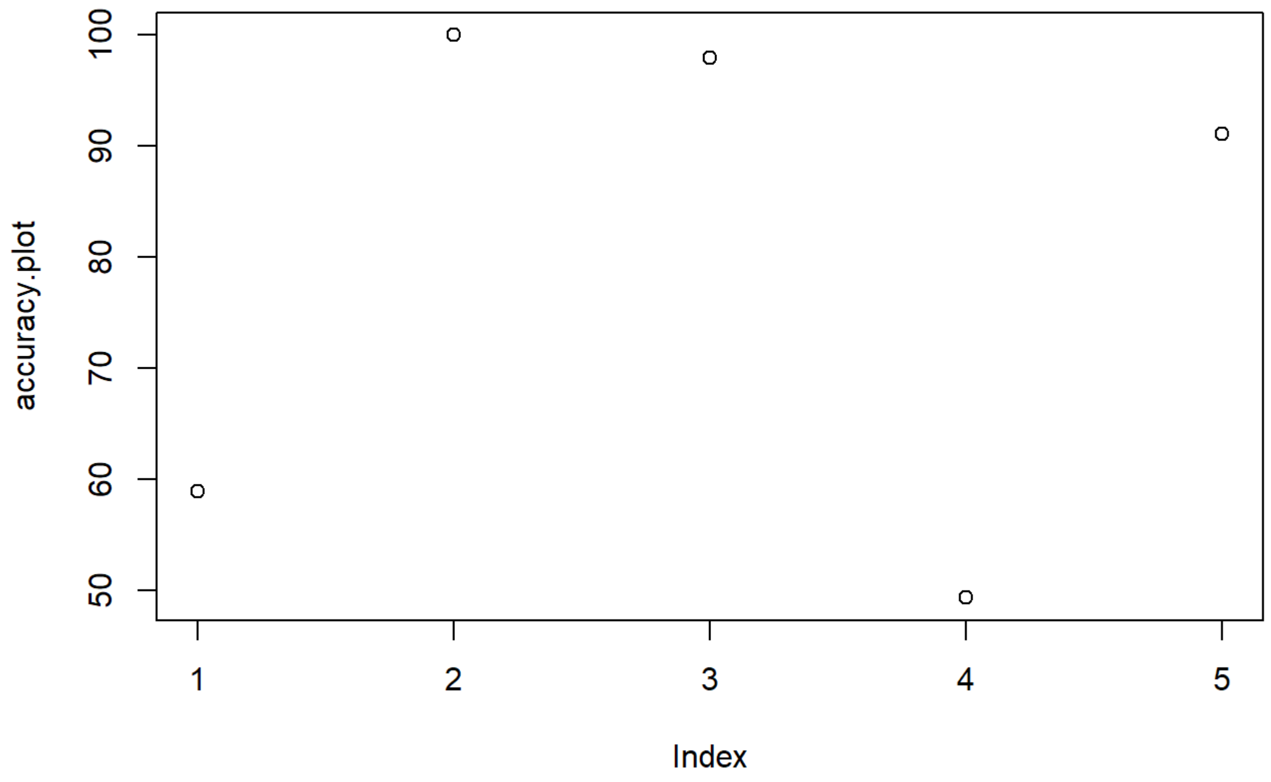
```
rep<- ensemble.model.predictions(test[,-1])
final.report<- data.frame(rep[,c(1:4)])
temp.x<- data.frame(NA)
for (i in 1:length(rep$ensemble)) {
temp.x<-rbind(temp.x,unlist(rep$ensemble[[i]][[1]]))
}
final.report<-cbind(final.report,temp.x[-1,])
final.report<-cbind(final.report,test$contraceptive)
names(final.report)<- c("decision tree1","decision tree2","kNN","NN","ensemble", "actual")
## Let us check what we have
head(final.report)
```

	decision tree1 <fct>	decision tree2 <fct>	kNN <fct>	NN <chr>	ensemble <chr>	actual <fct>
1	1	1	1	3	1	1
2	1	1	1	1	1	1
3	1	1	1	3	1	1
4	2	1	1	2	1	1
5	1	1	1	1	1	1
6	1	1	1	1	1	1

6 rows

Looks perfect!
Now we need to calculate the accuracy for each prediction.

```r
AC.1<- accuracy(table(final.report$actual, final.report$`decision tree1`))
AC.2<- accuracy(table(final.report$actual, final.report$`decision tree2`))
AC.3<-accuracy(table(final.report$actual, final.report$kNN))
AC.4<-accuracy(table(final.report$actual, final.report$NN))
AC.5<-accuracy(table(final.report$actual, final.report$ensemble))
accuracy.plot<- c(AC.1,AC.2,AC.3,AC.4,AC.5)
##colnames(accuracy.plot)<-c("dt1","dt2","kNN","NN","ensamble")
plot(accuracy.plot)
```



The worst accuracy is the NN then the decision tree in the first implementation.
The accuracy of the ensemble is actually worse than some of the individual models. This could be because the ensemble takes the majority vote. Sometimes, the we will have ties (2 models predict some class and two -better models- predict another). In this case the ensemble could go with the weaker prediction since there is no weights assigned to each model based on accuracy.

Now, I will try to make a prediction using the ensemble.
Let us say that we have a women who is 28 years old. She has 1 child. She is Muslim, exposed to media, she is working and she has middle education. Her husband has low middle education as well. Their living index is in the middle and the class of her husbands job is 3.
What will be her contraceptive method?

```r
head(test[12,c(-1)])
```

	w.age	num.children	w.religion	w.w…	me…	contraceptive	w.education.low	w.e
	<dbl>	<dbl>	<int>	<int>	<int>	<fct>	<dbl>	

	w.age	num.children	w.religion	w.w…	me…	contraceptive	w.education.low	w.e
	<dbl>	<dbl>	<int>	<int>	<int>	<fct>	<dbl>	
12	-0.3085306	-0.5348082	2	2	1	1	0	

1 row | 1-9 of 16 columns

```
## Age = 28
## num.children=1
## Religion=2
## Work=2
## Media= 1
## Contraceptive = ?
## W.education.low= 0
## W.education. mid=1
## h.education.low=0
## h.education.mid=1
## h.job.first=0
## h.job.second=0
## h.job.third=1
## living.index.low=0
## living.index.mid=1
## Let us see what the ensemble predicts for her
print(ensemble.model.predictions(test[12,c(-1)]))
```

```
##    decision tree1 decision tree2 kNN NN ensemble
## 12              1              1   1  3        1
```

```
#age<- 28 - mean(indo.contraceptives$w.age)/ sd(indo.contraceptives$w.age)
#child<- 1 - mean(indo.contraceptives$num.children)/sd(indo.contraceptives$num.children)
#case<- data.frame(age,child,2,2,1,0,0,1,0,1,0,0,1,0,1)
#case[,6]<- as.factor(case[,6])
#colnames(case)<- c("w.age","num.children","w.religion","w.work","media","contraceptive","w.educ
ation.low","w.education.mid","h.education.low","h.education.mid","h.job.first","h.job.secon
d","h.job.third","living.index.low","living.index.mid")
#print(ensemble.model.predictions(case))
```

The ensemble indeed predicts that she is not using any contraceptives.

Deployment:

This is the last step of the CRISP-DM, after reviewing the models with the business leaders, we can discuss how to deploy it.

Ref

1.Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml (http://archive.ics.uci.edu/ml)]. Irvine, CA: University of California, School of Information and Computer Science. 2. https://dhsprogram.com/pubs/pdf/SR9/SR9.pdf (https://dhsprogram.com/pubs/pdf/SR9/SR9.pdf) 3. https://en.wikipedia.org/wiki/Indonesia (https://en.wikipedia.org/wiki/Indonesia)