COSTA RICA POVERTY PREDICTION USE CASE STUDY REPORT

Group No.: Group 01

Student Names: Omkar Kalange and Vineeth Reddy

Executive Summary: There are a huge number of social programs, having a hard time finding the right set of people to address. It becomes tricky when it comes to the segmentation of poverty levels and focusing on the poorest segment available. The goal is devising a model for more appropriate means of classifying household into different levels of poverty for Costa Rica. It is challenging to decide on factors contributing to the poverty line, many such programs are currently going on, and still, the problem of determining the right threshold for poverty line remains unsolved. The objective is to predict poverty on a household level. Moreover, we must predict for every individual, but we should relate them to each household to predict poverty on a household basis. An exploratory data analysis was done with the data set. Categorial and non-categorical data columns were studied in detail. Redundant columns were removed, and highly correlated variables were also removed from data set. The data was prepared for mining and desired solutions were developed. The performance of these solutions was evaluated, and best model was selected. The best method of classification was undoubtedly the random forest because of its highly auc. The number of trees used for training and averaging were 300 which served a good purpose for giving an AUC score of 0.97. A data mining solution was developed and formulated to classify the household into different levels of poverty

I. Background and Introduction

Problem Statement

It is challenging to decide on factors contributing to the poverty line, many such programs are currently going on, and still, the problem of determining the right threshold for poverty line remains unsolved. It is not often easy to segregate poverty levels for different households and it is challenging to keep these results biased for equality. The problem here is to find an optimum solution over the traditional econometrics which might improve the overall process. Using the data given by IDB (Inter-American Development Board) for Costa Rica, we need to address the problem by categorizing households into four categories – extreme, moderate, vulnerable, and non-vulnerable. Apart from Costa Rica, the problem accounts for generalizing a machine learning algorithm that needs to address the poverty issues for other countries as well Provide a brief introduction to the opportunity and significance (why you study the particular use case).

Goal of Study

Recently, many initiatives are being taken for the eradication of poverty. There are a huge number of social programs, having a hard time finding the right set of people to address. It becomes tricky when it comes to the segmentation of poverty levels and focusing on the poorest segment available. The poorest typically can't prove whether they qualify, as they can't provide the necessary income and expense records. At present, one of the most viable methods to verify the income level qualification is the Proxy Means Test (PMT). Through PMT, many agencies include a family's noticeable household attributes like the material of their walls and ceiling or the assets found in the home for their models to classify them and predict their level of need. The goal is devising a model for more appropriate means of classifying household into different levels of poverty.

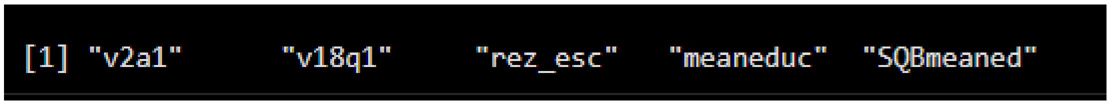
Possible Solution

The objective is to predict poverty on a household level. Moreover, we must predict for every individual, but we should relate them to each household to predict poverty on a household basis. As we are segmenting on the level of poverty, this will be a supervised multi-class classification machine learning problem:
Supervised: provided with the labels for the training data
Multi-class classification: Labels are discrete values with 4 classes.
First, we get introduced to the problem, then perform a thorough Exploratory Data Analysis of the dataset, work on feature engineering, try out multiple machine learning models, select a model, work to optimize the model, and evaluate its performance. Finally, inspect the outputs of the model and draw conclusions.

II. Data Exploration and Visualization

As a part of data exploration and visualization, the process was lengthy and complicated. We had to overcome the challenge of analyzing 142 columns to such a level that significance of them had to be kept in mind. Few of the columns had null values which had to be given importance as well. Instead of simply ignoring such values, an optimum solution was to be given a thought. As a result the data exploration and visualization was done in the following manner.

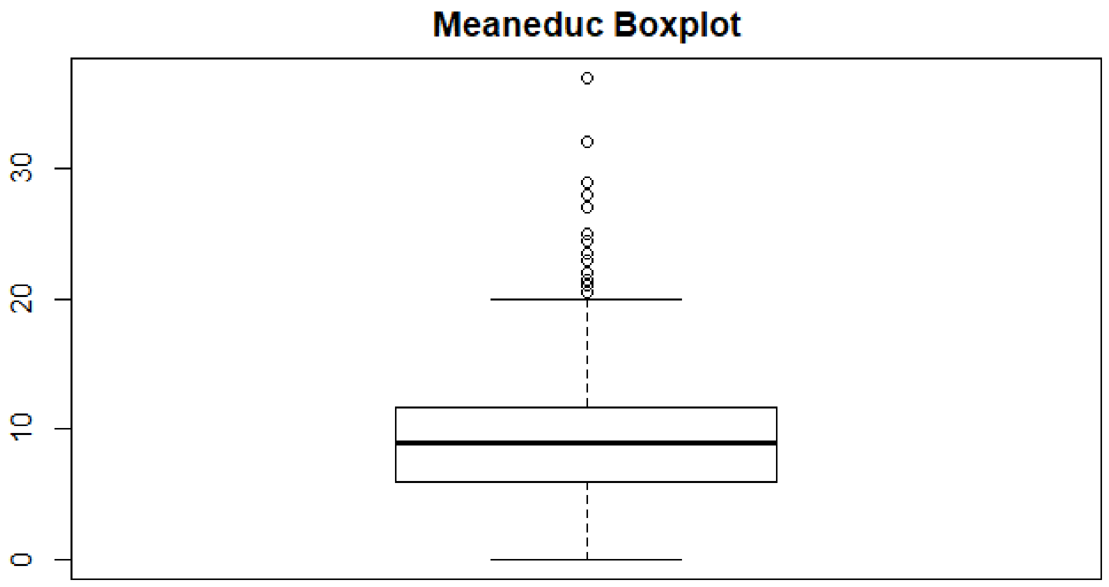
Null Analysis*:* After running analysis following columns contained null values



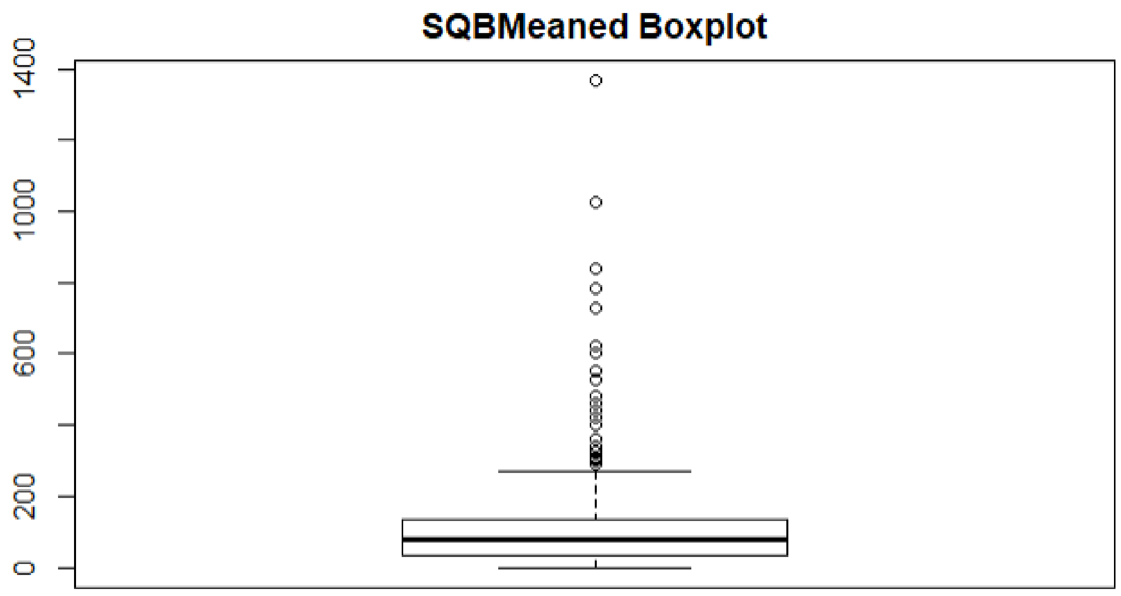
```
[1] "v2a1"      "v18q1"     "rez_esc"    "meaneduc"   "SQBmeaned"
```

Detailed exploration was required to ensure what can be done to get rid of such values. We observed each of the attribute and presented the findings according.

Distribution of meaneduc and SQBmeaned : The two attributes had 5 missing values.

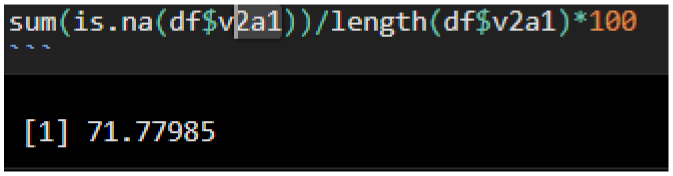


As we can observe that this particular attribute has several outliers. An optimum value should be chosen to fill these missing values, considering mean would be a bad idea. For the same reason, we chose to replace these 5 values with median of the column *Meaneduc*.



Same goes for the SQBMeaned column, 5 missing values were replaced by median of SQBMeaned.

V21a1 :

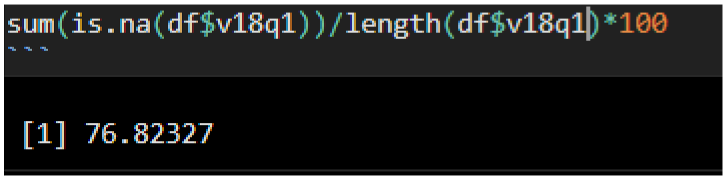


```
sum(is.na(df$v2a1))/length(df$v2a1)*100
```

```
[1] 71.77985
```

The column can be ignored since it contains 72% missing values. Replacing the values can overfit the model so the best idea is to get rid of the column.

V18q1 :

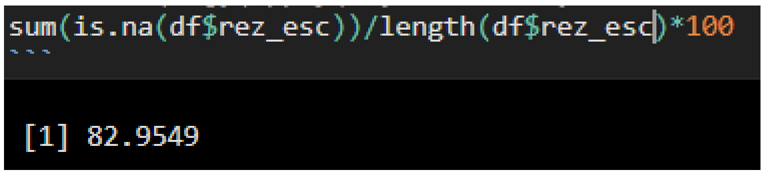


```
sum(is.na(df$v18q1))/length(df$v18q1)*100
```

```
[1] 76.82327
```

Again this column can be ignored since it contains 77% missing values. Replacing the values can overfit the model so the best idea is to get rid of the column.

Rez_esc :
This column also has 83% missing values, got rid of this column too.

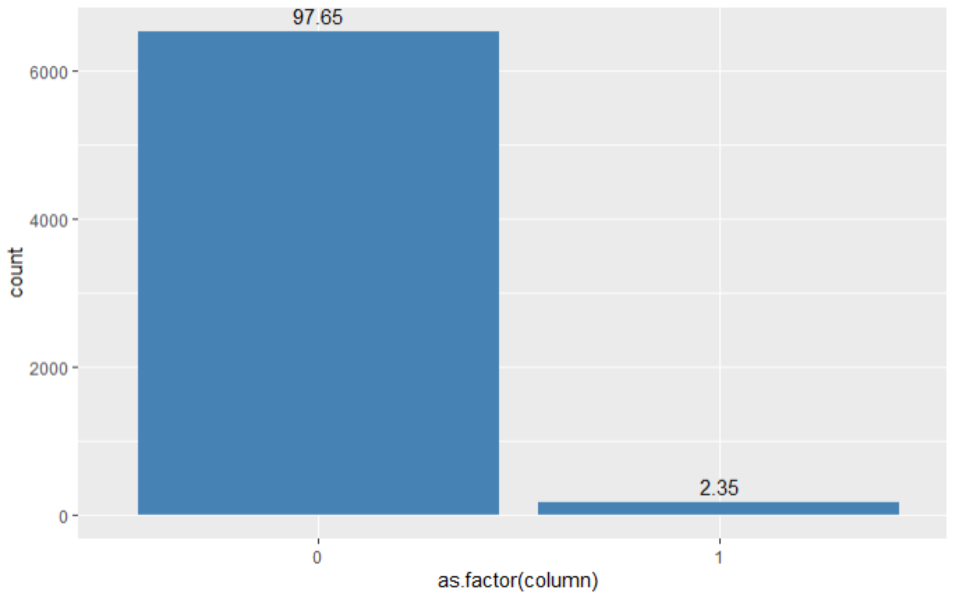


```
sum(is.na(df$rez_esc))/length(df$rez_esc)*100
```

```
[1] 82.9549
```

Exploring Binary Columns : There were 102 Binary columns (categorical) in the dataset. All the columns were one hot encoded based. There was challenge of analyzing the distribution of the columns containing 1s and 0s. User defined functions were formulated to visualize the histogram of these columns. The task of reducing and optimizing the model would have achieved by removing columns containing 97% or more 1s or 0s. Such columns were removed from the data set. Some of the columns had distribution like :

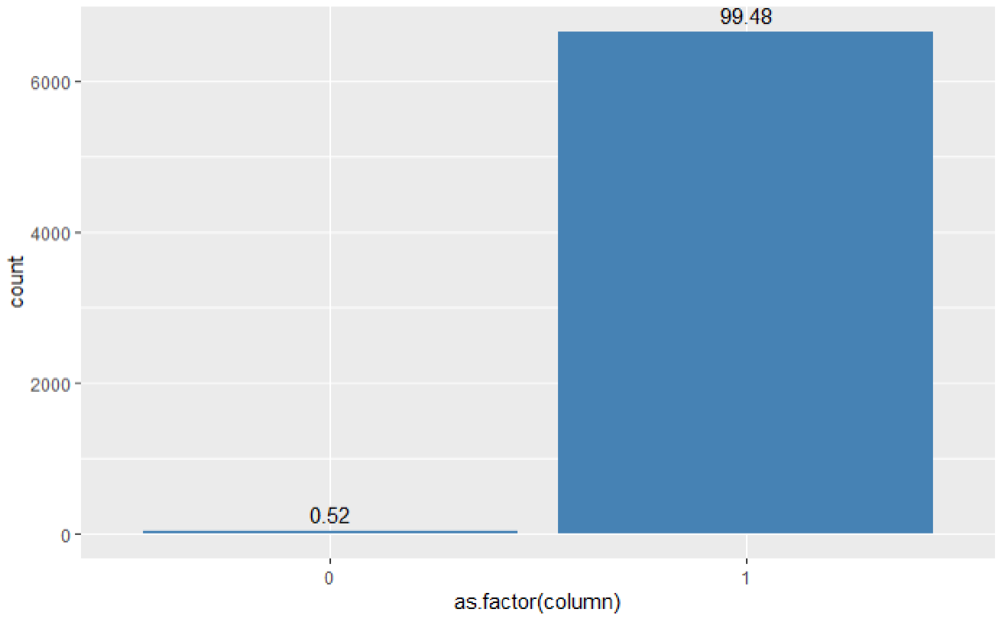
Hacapo :



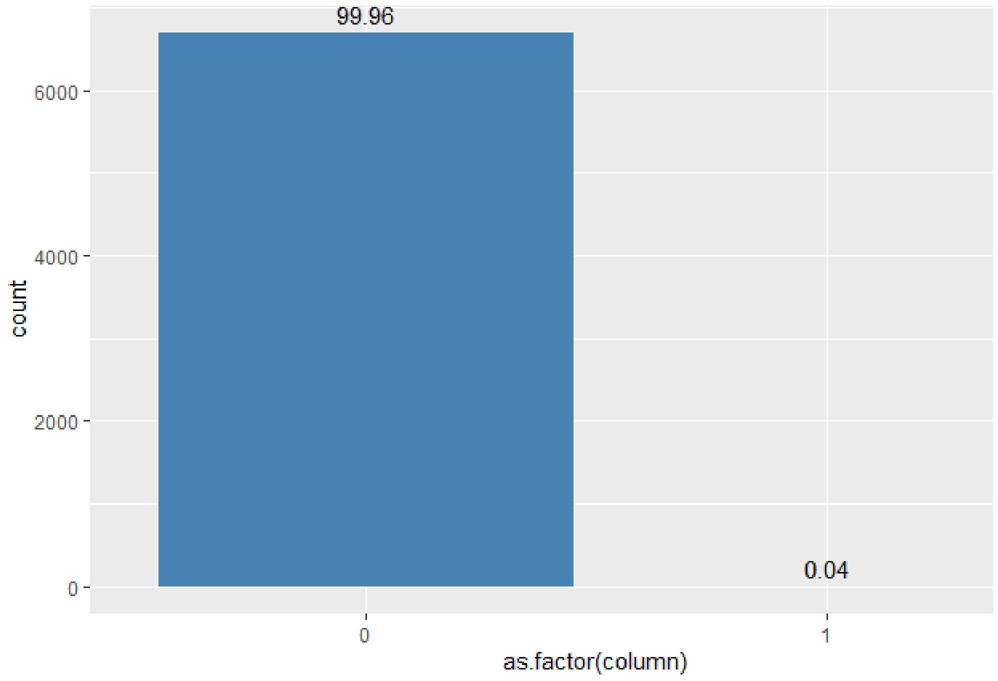
This particular column as seen as 97% 0s in it. This particular column can be eradicted from the dataset.

V14a :

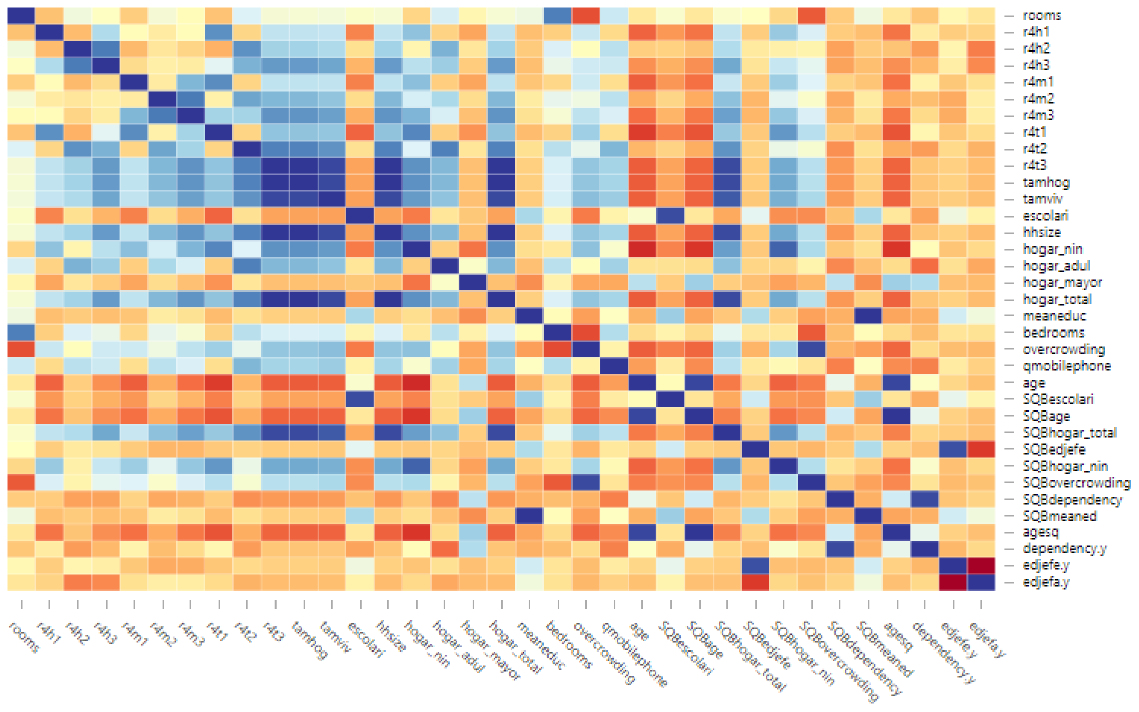
This column as shown on next page has 99.48% of 1s and hence is removed from the data set.



Planpri :



Exploring Numerical Columns : The analysis of 35 numerical columns was to be performed to get valuable insights. This was carried out by visualizing the dependence of every column on other. For this, correlation came into picture which stated how these 35 numerical columns interacted with each other. Heat map was used to visualize the correlations among these numerical columns.



Correlation of Numerical Columns

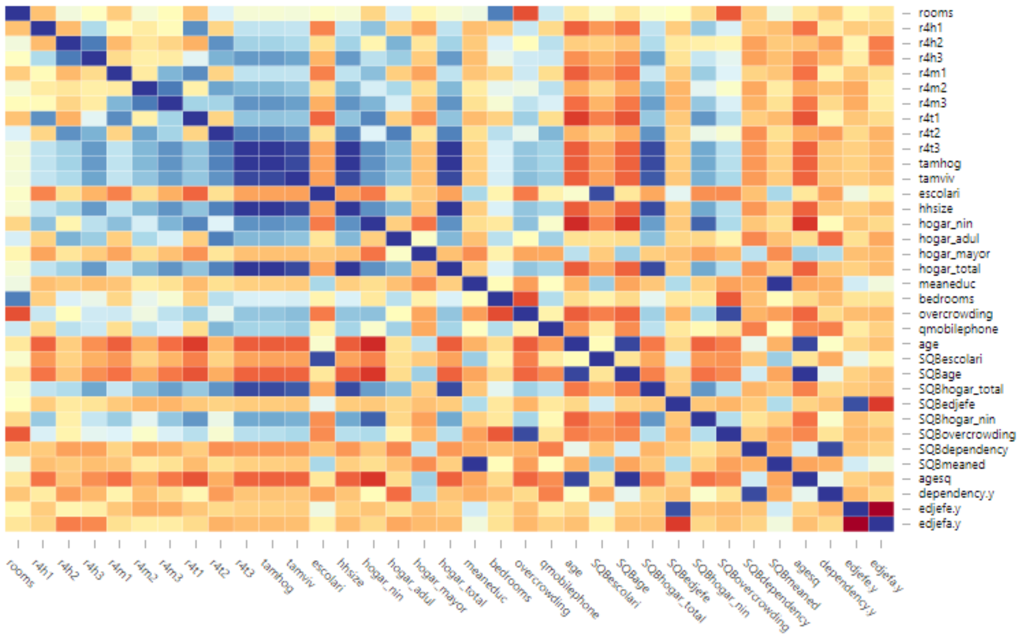
As a part of data preparation, we needed to analyze the correlations. Set of highly correlated both positively and negatively was needed to study further details. Highly correlated variable should be removed as a part of data cleaning process. These variables are redundant while applying them to algorithm.

III. Data Preparation and Preprocessing

As a part of data preparation, we explored different aspects of the dataset. Starting from breaking down of data to segregating them in form of binary and numerical, the dataset was broken into such level of details.

When we observed that few of the binary columns contained more than 97% of 1s or 0s we got rid of such attributes. This was the very first step of data preparation for our project. We created user defined functions to optimize our work and instead of visualizing and analyzing plot of 80 different columns we wrote mathematical equations and subsequently removed the columns.

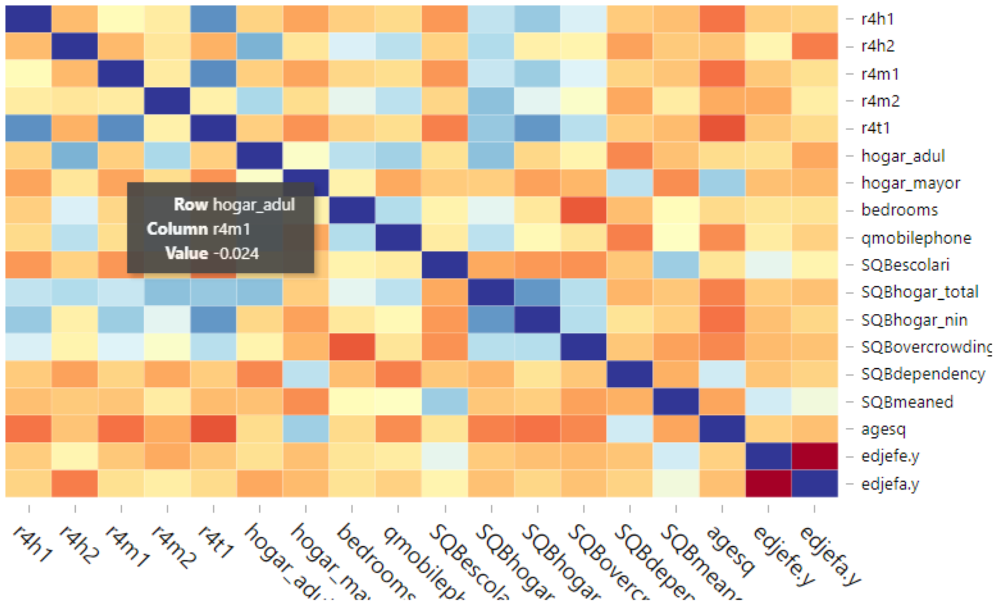
Before data preprocessing :



Correlation of Numerical Columns

While preparing the numerical data for analysis, we studied the correlations and removed the variables which had high level of correlation with other variables. This was a major step to prepare numerical data and process it. The earlier heatmap of correlations showed that there were many numerical columns having strong correlations. We removed them and the optimized heatmap with fewer variables was obtained. The following figures illustrate the same.

After processing and removing highly correlated variables :

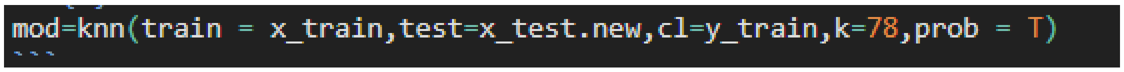


In this way, the data was processed and we reduced the columns from 142 to 85 which led to massive reduction in the data set and the dataset was ready for applying mining solutions.

IV. Data Mining Techniques and Implementation

The problem suggested was of classification, more precisely multi class classification. We explored various data mining techniques as appropriate to our solution. The following algorithms were applied for the dataset :

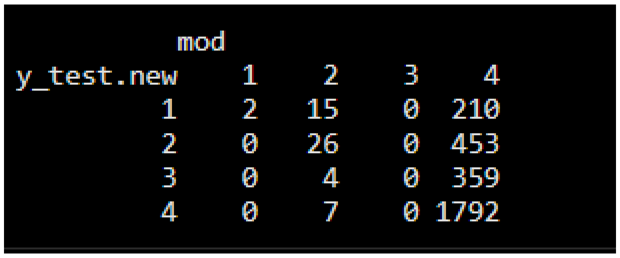
1. **KNN (K-Nearest Neighbor Algorithm) :**
   KNN was applied to our dataset to classify the household based on 85 attributes. While applying KNN, we needed to split the data set into training and testing data set in order to validate its accuracy. It was equally challenging to find a good value of K which gave a maximum accuracy for KNN. Using the testing data set we predicted values for the poverty levels of different testing household data.



```
mod=knn(train = x_train,test=x_test.new,cl=y_train,k=78,prob = T)
```

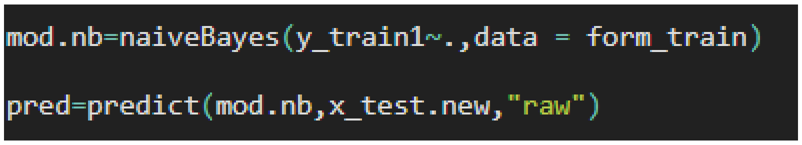
   After performing dry run, we got maximum accuracy for k=78

   Confusion Matrix



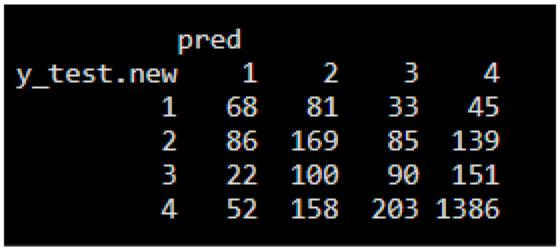
```
          mod
y_test.new    1    2    3    4
         1    2   15    0  210
         2    0   26    0  453
         3    0    4    0  359
         4    0    7    0 1792
```

2. **Naïve Bayes Classifier** : We trained our datset on Naïve Bayes classifier as well. We wanted to verify whether all of the attributes are dependent on each other or not. Even though this was a try, lot of interpretations were made from this considering the dependency of household properties on each other. We assumed that these attributes were strongly dependent on each other however they weren't from the accuracy that we got from this classifier.



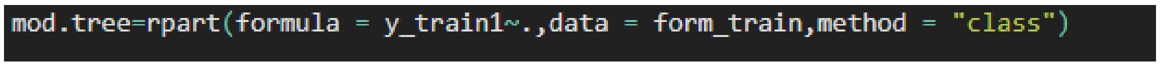
```
mod.nb=naiveBayes(y_train1~.,data = form_train)

pred=predict(mod.nb,x_test.new,"raw")
```

   Confusion Matrix



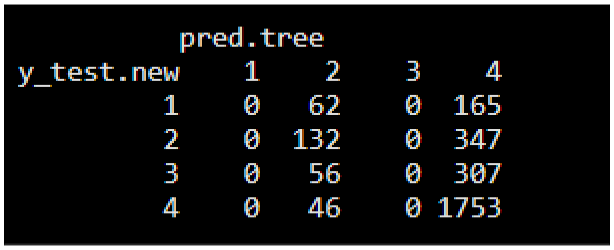
```
          pred
y_test.new    1    2    3    4
         1   68   81   33   45
         2   86  169   85  139
         3   22  100   90  151
         4   52  158  203 1386
```

3. **Decision Tree** : While classifying a particular household, we normally in real life might use decision tree itself. So studying property of this model was equally important for us. Training a decision tree for classification model was done and results were obtained.



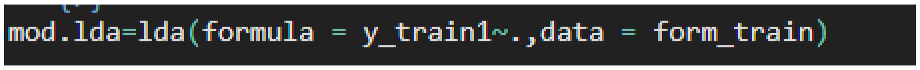
```
mod.tree=rpart(formula = y_train1~.,data = form_train,method = "class")
```

Confusion Matrix

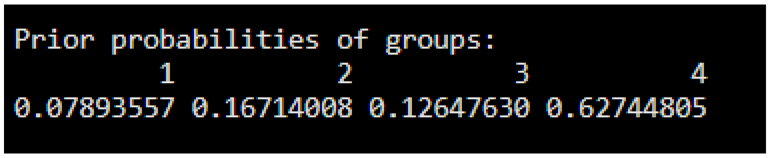


```
          pred.tree
y_test.new    1    2    3    4
         1    0   62    0  165
         2    0  132    0  347
         3    0   56    0  307
         4    0   46    0 1753
```

4. **Linear Discriminant Analysis** : Solving a classification problem while reducing the dimensions of 89 would surely be a good idea to approach the scenario. We performed LDA on our dataset to check if the results were good by reducing the dimension.

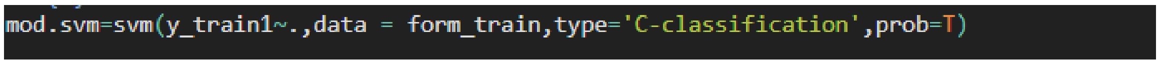


```
mod.lda=lda(formula = y_train1~.,data = form_train)
```

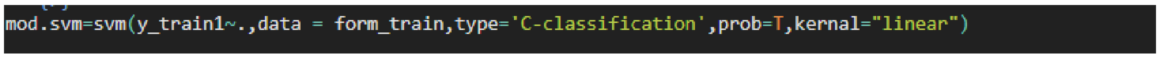


```
Prior probabilities of groups:
         1          2          3          4 
0.07893557 0.16714008 0.12647630 0.62744805 
```

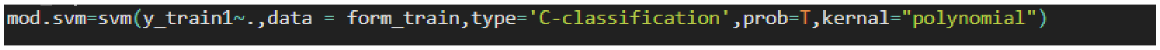
5. **Support Vector Machines** : Implementation of support vector machines with different kernals was done. Using SVMs ensured that the data was separated linearly by projecting them to different levels of axes.



```
mod.svm=svm(y_train1~.,data = form_train,type='C-classification',prob=T)
```

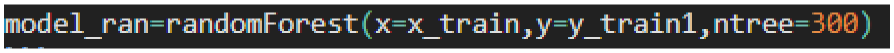


```
mod.svm=svm(y_train1~.,data = form_train,type='C-classification',prob=T,kernal="linear")
```



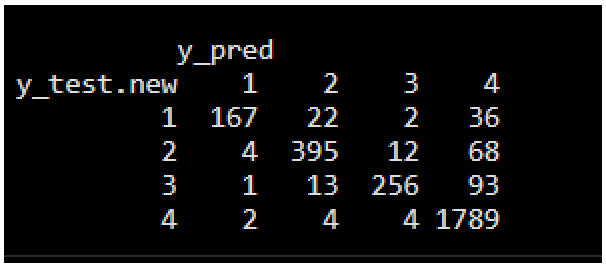
```
mod.svm=svm(y_train1~.,data = form_train,type='C-classification',prob=T,kernal="polynomial")
```

6. **Random Forest Classification** : As we observed, during decision tree few classes were not predicted. It is advisable to use many trees i.e ensemble of trees to proceed for the solution. Random forest, trees were varied from 200 to 500 with gaps fo 50 and best results were obtained for trees =300.



```
model_ran=randomForest(x=x_train,y=y_train1,ntree=300)
```

Confusion Matrix



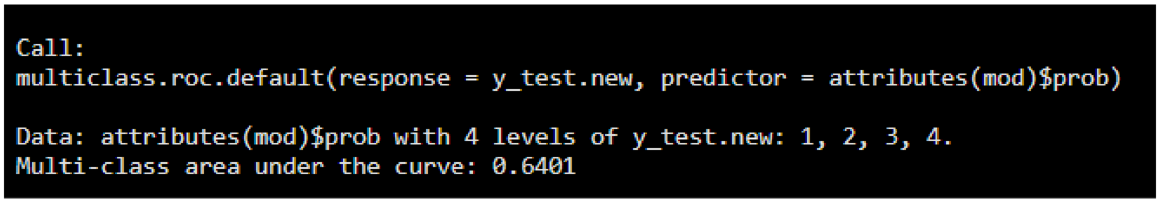
```
          y_pred
y_test.new    1    2    3    4
         1  167   22    2   36
         2    4  395   12   68
         3    1   13  256   93
         4    2    4    4 1789
```

V. Performance Evaluation

As a result of asymmetrical distributions of classes a more strong performance evaluator was needed to ensure the solution isn't biased. We didn't make use of accuracy calculated from the confusion table. The reason being that the sample had more predictions of class 4 and we didn't want to move towards best accuracy with a false behavior. For the same, we evaluated the models based on multi class auc score. It was not possible to visualize the ROC curve because we had 4 outcome variables.

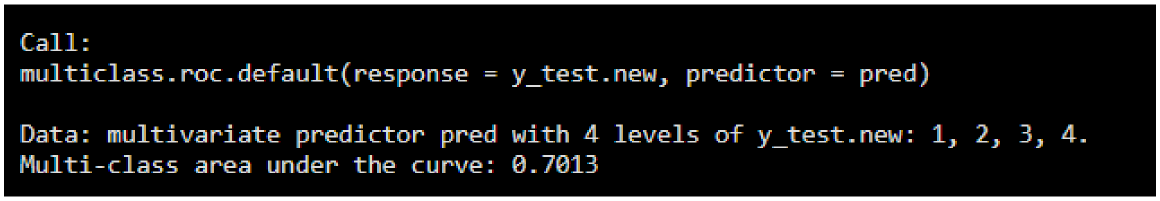
Every model, we checked the AUC score and the results were obtained in the following manner :

1. *KNN :* for the best possible k = 78 : AUC = 0.64



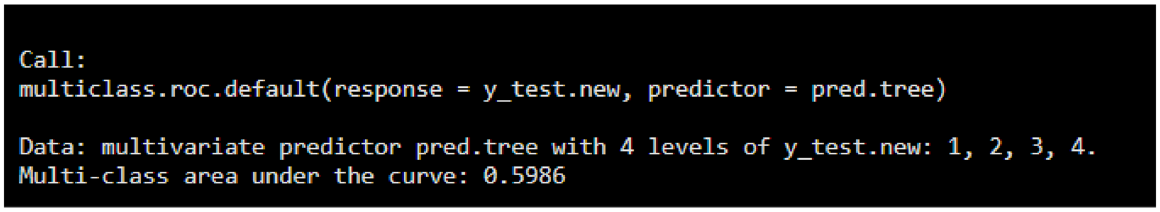
```
Call:
multiclass.roc.default(response = y_test.new, predictor = attributes(mod)$prob)

Data: attributes(mod)$prob with 4 levels of y_test.new: 1, 2, 3, 4.
Multi-class area under the curve: 0.6401
```

2. *Naïve Bayes :* The model increased the AUC score to 0.70 from KNN model



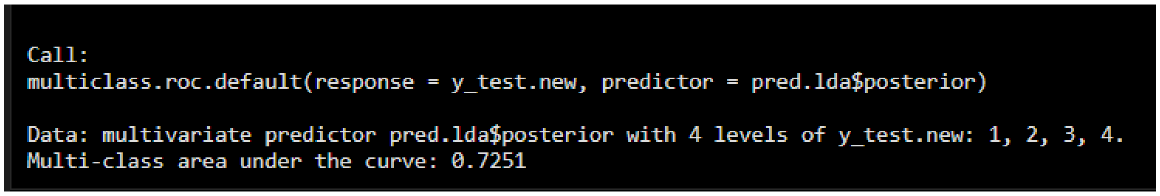
```
Call:
multiclass.roc.default(response = y_test.new, predictor = pred)

Data: multivariate predictor pred with 4 levels of y_test.new: 1, 2, 3, 4.
Multi-class area under the curve: 0.7013
```

3. *Decision Tree :* The decision tree had AUC score of 0.59



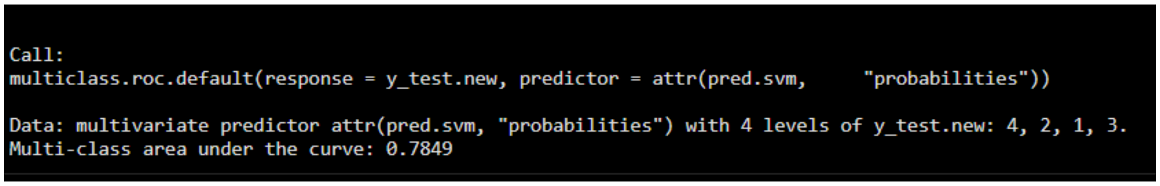
```
Call:
multiclass.roc.default(response = y_test.new, predictor = pred.tree)

Data: multivariate predictor pred.tree with 4 levels of y_test.new: 1, 2, 3, 4.
Multi-class area under the curve: 0.5986
```

4. *Linear Discriminant Analysis :* AUC Score : 0.72



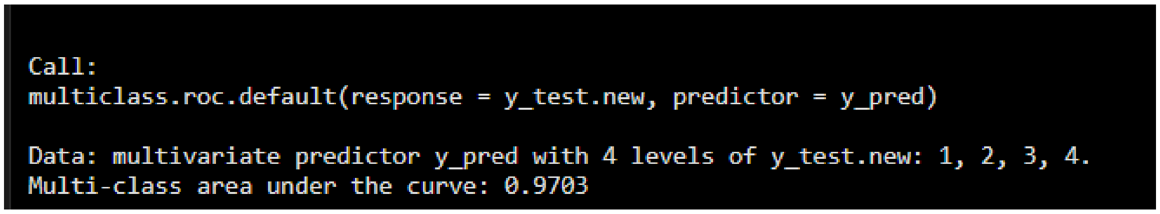
```
Call:
multiclass.roc.default(response = y_test.new, predictor = pred.lda$posterior)

Data: multivariate predictor pred.lda$posterior with 4 levels of y_test.new: 1, 2, 3, 4.
Multi-class area under the curve: 0.7251
```

5. *Support Vector Machine :* AUC Score : 0.78



```
Call:
multiclass.roc.default(response = y_test.new, predictor = attr(pred.svm,     "probabilities"))

Data: multivariate predictor attr(pred.svm, "probabilities") with 4 levels of y_test.new: 4, 2, 1, 3.
Multi-class area under the curve: 0.7849
```

6. *Random Forest :* AUC Score : 0.97



```
Call:
multiclass.roc.default(response = y_test.new, predictor = y_pred)

Data: multivariate predictor y_pred with 4 levels of y_test.new: 1, 2, 3, 4.
Multi-class area under the curve: 0.9703
```

The best method of classification was undoubtely the random forest because of its highly auc. The number of trees used for training and averaging were 300 which served a good purpose for giving an AUC score of 0.97.

VI. Discussion and Recommendation

The overall approach was to classify the hold of Costa Rica based on the information which we received from the board. The aim was to do this in an unbiased manner. There would have been a strict policy rules which to determine the poverty status as it constitutes a major part in country’s growth.

The advantages were lot of information was provided for the classification problem. Around 142 attributes of particulate household were sufficient for us to come to conclusion. The disadvantage was that the limited sample size could some way affect the generalized model and might prove wrong at times. The recommendation would be getting more data in terms of households for a more accurate model.

VII. Summary

The objective to categorize the households of Costa Rica into four levels of poverty was achieved by designing the model. When the data had different attributes from families our aim is to reduce the redundant variables and get a optimum dataset of the development of machine learning model. An exploratory data analysis made us to work on feature engineering of various attributes of household and then making data ready for data mining solution. After designing models, it was important for us to evaluate the performance of model so that it doesn’t perform in unbiased manner. Therefore, a data mining solution was developed and formulated to classify the household into different levels of poverty.

Appendix: R Code for use case study

~~~
```{r warning=F,echo=F}
library(ggplot2)
library(tidyverse)
library(caTools)
library(d3heatmap)
library(dostats)
library(gplots)
library(caret)
library(class)
library(randomForest)
library(pROC)
library(e1071)
library(rpart)
library(MASS)
library(neuralnet)
```
~~~

~~~
```{r}
df=read.csv("costarica.csv", header=T,stringsAsFactors = F)
#colnames(df)
```
~~~

Null Analysis of entire dataframe -

Analyzing null valued columns -

Checking % of nulls present in null valued columns :

~~~
```{r}
#colnames(df[,(apply(df,MARGIN = 2,FUN=function(x)
{sum(is.na(x))/length(x)*100})>0)])
sum(is.na(df$rez_esc))/length(df$rez_esc)*100
```
~~~

Handling meaneduc and SQBmeaned fields - We have 5 observations of NaN, instead of dropping the rows : We can visualize the distribution and decide how these values can be filled

i) Meaneduc :

~~~
```{r}
boxplot(x=df$SQBmeaned,main="SQBMeaned Boxplot")
```
~~~

Due to outliers, it is advisable to use median to fill null values rather than using mean.

```{r}
med_educ=median(df$meaneduc,na.rm = T)
df$meaneduc=replace_na(df$meaneduc,replace = med_educ)

sqmed_educ=median(df$SQBmeaned,na.rm=T)
df$SQBmeaned=replace_na(df$meaneduc,replace = sqmed_educ)

```

We can drop these three columns because they contain more than 70% Null values.

```{r}
df1=subset(df,select=-c(v2a1,v18q1,rez_esc),drop=F)
```

**Removing unnecessary column - "idhogar" **

```{r}
df.new=subset(df1,select = -c(idhogar),drop=F)

#df.num=subset(df,select = -c(Id,dependency,edjefe,edjefa))


```

Columns of String type

```{r}
df.num2=unlist(lapply(df.new,is.numeric))
df.str=df.new[,!df.num2]
nrow(df.str)
```

Edjefe : years of education of male head of household, based on the interaction of escolari (years of education), head of household and gender, yes=1 and no=0

Replace yes with 1 and no with 0

Checking nulls -
```{r}
apply(df.str,MARGIN=2, function(z) sum(is.na(z)))
```

~~~
```{r}

df.str$dependency=ifelse(df.str$dependency=="yes",1,df.str$dependency)
df.str$dependency=ifelse(df.str$dependency=="no",0,df.str$dependency)

df.str$edjefe=ifelse(df.str$edjefe=="yes",1,df.str$edjefe)
df.str$edjefe=ifelse(df.str$edjefe=="no",0,df.str$edjefe)

df.str$edjefa=ifelse(df.str$edjefa=="yes",1,df.str$edjefa)
df.str$edjefa=ifelse(df.str$edjefa=="no",0,df.str$edjefa)
```



```{r}
df.str$dependency=as.numeric(df.str$dependency)

df.str$edjefe=as.numeric(df.str$edjefe)

df.str$edjefa=as.numeric(df.str$edjefa)
```

```{r}
df.new1=merge.data.frame(x=df.new,y=df.str,by = "Id")
head(df.new1)
```

```{r}
df.new2=subset(df.new1,select=-c(dependency.x,edjefe.x,edjefa.x),drop=F)
head(df.new2)
```



**Splitting Data into Test and Train**

```{r}
set.seed(12345)

split=sample.split(df.new2$Target,SplitRatio = 0.7)

df_train=subset(df.new2,subset = split)

df_test=subset(df.new2, subset = !split)

(df_train)
~~~

~~~
```

```{r}
df.num1=unlist(lapply(df_train,is.numeric))
df.num=df_train[,df.num1]

#head(df.num)
(df.num)
```




**Function to find binary columns from the dataframe**

```{r}
bin.col=list()
is.binary<-function(col){
  x<-unique(col)
  if( sum(x,na.rm=T) == 0 || sum(x,na.rm=T) ==1){
    return(T)
  }else{
    return(F)
  }
}
```

**Make separate dataframe for Binary Columns**

```{r}
df.bin1=unlist(lapply(df.num,is.binary))
df.bin=df.num[,df.bin1]
length(df.bin)
```

**Visualizing the categorial Variables to see the count of 1s and 0s**

```{r}
p=list()
i=1
check.dist<-function(column){
    ggplot(df.bin,aes(x=as.factor(column)))+stat_count(binwidth =
1,fill="steelblue")+stat_count(binwidth=1,geom =
"text",aes(label=round((..count../sum(..count..))*100,digits = 2)),vjust=-
0.5)+labs(colnames(column))
}
~~~

```
#for (i in colnames(df.bin)){
check.dist(df.bin$planpri)
#}


```

Removing binary columns which have more than 97% of 0s or 1s because they might be redundant while designing data mining model

```{r}
train.size=nrow(df.bin)

check.majority<-function(col){
  if ((sum(col)/train.size)>=.97 || (sum(col)/train.size)<=0.03) return (F) else return(T)
}

df.bin.t=unlist(lapply(df.bin,check.majority))

df.bincols=df.bin[,df.bin.t]
#colnames(df.bincols)
```

Length of Binary columns reduced from 102 to 67, 35 colummns either had 97% of 1s or 0s so it was bad idea considering them to determine target variable

```{r}
df.bincols.new=as.factor(df.bincols)
```

Segregating numerical and string columns -

```{r}


df.nume=df.num[,!df.bin1]
#length(df.nume)
df.nums=subset(df.nume,select = -c(Target))

(df.nums)
```

Data Analysis of Numerical Columns

Checking for numerical columns containing null values:

~~~
```{r}

check.na<-function(col1){
  y<-unique(col1)
  return(y %contains% NA)
}


df.null1<-unlist(lapply(df.nums,check.na))
df.null<-df.nums[,df.null1]
df.notnull<-df.nums[,!df.null1]

(df.notnull)
```
~~~

Correlation of not null numeric columns :

~~~
```{r}
df.cor=cor(df.notnull)

d3heatmap(df.cor,dendrogram = "none",main=paste("Heatmap for Numeric Variables"))

#heatmap.2(cor(df.notnull),dendrogram = "none")

#length(df.notnull)

```
~~~

Removing highly correlated variables

~~~
```{r}
x=findCorrelation(df.cor,cutoff=0.8,names=T)

#subset(df.notnull,select=-c(findCorrelation(df.cor,cutoff=0.8,names=T),drop=F))
df.numeric=df.notnull[,!colnames(df.notnull) %in% x]

d3heatmap(cor(df.numeric),dendrogram = "none",main=paste("Heatmap for Numeric
Variables"))

```
~~~

Numerical = 18 columns (earlier = 35)

Binary = 67 columns   (earlier = 102)

Total X = 85

~~~
```{r}
dfbinary=apply(df.bincols,MARGIN=2,function(a) as.factor(a))

dfbinary1=data.frame(dfbinary)

```
~~~

~~~
```{r}
x_train=cbind.data.frame(df.numeric,dfbinary1)
y_train=(df_train$Target)

colnames(x_train)
```
~~~

~~~
```{r}
x_test=df_test[c("r4h1"          , "r4h2"           ,"r4m1"           ,"r4m2"           ,"r4t1"
,"hogar_adul"    ,
"hogar_mayor"    , "bedrooms"       ,"qmobilephone"    ,"SQBescolari"
,"SQBhogar_total"  ,"SQBhogar_nin"  ,
"SQBovercrowding", "SQBdependency"   ,"SQBmeaned"       ,"agesq"          ,"edjefe.y"
,"edjefa.y"      ,
"hacdor"         , "refrig"          ,"v18q"           ,"paredblolad"     ,"paredzocalo"
,"paredpreb"     ,
"paredmad"       , "pisomoscer"      ,"pisocemento"     ,"pisomadera"      ,"techozinc"
,"cielorazo"     ,
"abastaguadentro", "abastaguafuera"  ,"public"          ,"coopele"         ,"sanitario2"
,"sanitario3"    ,
"energcocinar2"  , "energcocinar3"   ,"energcocinar4"   ,"elimbasu1"       ,"elimbasu2"
,"elimbasu3"     ,
"epared1"        , "epared2"         ,"epared3"         ,"etecho1"         ,"etecho2"
,"etecho3"       ,
"eviv1"          , "eviv2"           ,"eviv3"           ,"dis"             ,"male"            ,"female"        ,
"estadocivil1"   , "estadocivil2"    ,"estadocivil3"    ,"estadocivil4"    ,"estadocivil5"
,"estadocivil6"  ,
"estadocivil7"   , "parentesco1"     ,"parentesco2"     ,"parentesco3"     ,"parentesco6"
,"instlevel1"    ,
"instlevel2"     , "instlevel3"      ,"instlevel4"      ,"instlevel5"      ,"instlevel8"
,"tipovivi1"     ,
~~~

```
"tipovivi2"      , "tipovivi3"       ,"tipovivi5"       ,"computer"        ,"television"      ,"lugar1"
,
"lugar2"         , "lugar3"          ,"lugar4"          ,"lugar5"          ,"lugar6"          ,"area1"
,
"area2"          )]
```

```{r}
dftest.bin1=unlist(lapply(x_test,is.binary))
df.bintest=x_test[,dftest.bin1]

dftestbinary=apply(df.bintest,MARGIN=2,function(a) as.factor(a))

dftestbinary1=data.frame(dftestbinary)

dftestbinary1
```

```{r}
#dft.num1=unlist(lapply(x_test,is.numeric))
dft.num=x_test[,!dftest.bin1]

dft.num
```

```{r}
x_test.new=cbind.data.frame(dft.num,dftestbinary1)
y_test=(df_test$Target)

(x_test.new)
```

```{r}

y_train.new=(as.factor(y_train))
y_test.new=(as.factor(y_test))

#knn(train = x_train,test=x_test.new,cl=y_train,k=15)
```

Training KNN model :

~~~
```{r}
mod=knn(train = x_train,test=x_test.new,cl=y_train,k=78,prob = T)
```

```{r warning=F}
knntable = table(y_test.new,mod)
#sum(diag(knntable))/sum(knntable)

multiclass.roc(y_test.new,attributes(mod)$prob)

#knntable
```
```{r}
#cal_err <- function(actual,pred){
 # mean(actual!=pred)
#}
```
~~~

Error:

~~~
```{r}
#cal_err(y_test.new,mod)

```
~~~

Training Naive Bayes Model :

~~~
```{r}
y_train1=as.factor(y_train)

form_train=cbind.data.frame(x_train,y_train1)

mod.nb=naiveBayes(y_train1~.,data = form_train)

pred=predict(mod.nb,x_test.new,"raw")
#pred=predict(mod.nb,x_test.new)

multiclass.roc(y_test.new,pred)

#table(y_test.new,pred)
```
~~~

Training Decision Tree :

~~~
```{r}
~~~

```
mod.tree=rpart(formula = y_train1~.,data = form_train,method = "class")

pred.tree=predict(mod.tree,x_test.new)

multiclass.roc(y_test.new,pred.tree)

#table(y_test.new,pred.tree)

```

Linear Discriminant Analysis :

```{r}
mod.lda=lda(formula = y_train1~.,data = form_train)

pred.lda=predict(mod.lda,x_test.new,prob=T)

#pred.lda=predict(mod.lda,x_test.new)

multiclass.roc(y_test.new,pred.lda$posterior)

mod.lda
```

Neural Network :

```{r}

#form_train1=as.data.frame(mutate_all(form_train, function(x)
if(as.numeric(as.character(x)))))

form_neu=as.data.frame(mutate_all(form_train[,-86],function(x)
(as.numeric(as.character(x)))))

form_neu1=cbind.data.frame(form_neu,y_train1)

n=names(form_neu1)

f<-as.formula(paste("y_train1~",paste(n[!n%in%"y_train1"],collapse="+")))

#mod.neu=neuralnet(formula = f, form_neu1, hidden = c(3,2),threshold =
0.01,linear.output = F,act.fct = "logistic")

#predict(mod.neu,mutate_all(x_test.new, function(x)
as.numeric(as.character(x))),type="raw")
```

~~~
#prob=compute(x = mod.neu, as.data.frame(mutate_all(x_test.new, function(x)
as.numeric(as.character(x))) ))

#prob$net.result

```

Support Vector Machines -

```{r}
mod.svm=svm(y_train1~.,data = form_train,type='C-classification',prob=T)

pred.svm=predict(mod.svm,x_test.new,probability = T)

multiclass.roc(y_test.new,attr(pred.svm, "probabilities"))
```



Training Random Forest Model :

```{r}
#mod_ran=randomForest(x = x_train,y = y_train,xtest = x_test.new,ytest
=y_test.new,ntree = 20)

#form=cbind.data.frame(x_train,y_train1)
#model_ran=randomForest(y_train~.,data = form_train,ntree=)
model_ran=randomForest(x=x_train,y=y_train1,ntree=300)
```

```{r}
#predict(model_ran,x_test.new)
#model_ran
#form_valid=cbind.data.frame(x_test.new,y_test.new)
y_pred=predict(model_ran,x_test.new)
conf=table(y_test.new,y_pred)
#conf
multiclass.roc(y_test.new,y_pred)
```

```{r}
#(conf[1,1]+conf[2,2]+conf[3,3]+conf[4,4])/sum(conf)
```

```{r}
~~~

```
```