**Week 6 Assignment**

**A close up of a piece of paper

Description automatically generated**

**Costa Rican Household Poverty Level Prediction Using R**

A picture containing drawing

Description automatically generated

**Course**: [ALY6040 Data Mining Applications](about:blank)

**Instructor:**

**Kasun Samarasinghe**

**Submission date:**

March 26, 2020

By,

SUVARNA TAMBE

**INTRODUCTION**

Data analysis is a method by which raw data are gathered and turned into usable information for consumer decision making. Data is obtained and used for problems study, check theory, or hypotheses rejection. There are six stages, such as data necessity, data acquisition, data preparation, data cleaning, exploratory data analysis, and modeling & algorithms, according to the cross-industry standard method for data mining. Data cleaning is the process of preparing data for processing by extracting or changing the wrong data set, missing or outliers and data cleaning is not merely about deleting information to make room for new data but instead finding a way to improve the output of a data set without automatically deleting the information. The data collection includes multiple tools to define and handle missing values and outliers. Using a range of methods referred to as exploratory data analysis, after data is cleaned up, is the essential method of conducting a qualitative data review to identify correlations or observations and to test conclusions with support in descriptive statistics and graphical representations. Models are key to predicting the outcomes of business decisions. Modeling involves selecting the right data sets, algorithms, and variables with the right techniques to format data for a particular business problem. An analytical model forecasts or classifies data values by simply drawing a line through data points when applying to study data sets, and a model may forecast effects based on historical trends. Model building journey follows key components like creating hypothesis, loading and transforming data, identifying features, choosing the right model, and evaluating the model. Optimizing the model for predicting the target variable for more accurately and improving the model performance. Mode selection refers to the problem of selecting the best model from the set models for decision making or optimization under certainty. Methods like data transformation, exploratory analysis, and model specification will assist in choosing the set of candidate models. The Accuracy of the predictive model can be boosted in two ways, either by embracing feature engineering or by applying boosting.

**ABOUT THE DATASET**

For this project, we have used a dataset of Inter-American Development Bank, which consists of socio-economic conditions of Costa Rican Households such as education background, family details, home, electricity details, etc. Inter-American Development Bank gives this data set on the Kaggle forum. This data set contains 143 variables like home rent, house conditions, education details, household information, etc. and it contains 9558 rows of data. We need to predict the poverty level (Target variable) of household-based information. The target variable is an ordinal variable indicating groups of income level, i.e., 1= extreme poverty, 2=moderate poverty, 3= vulnerable households, 4=non-vulnerable households. For this project, we have used statistical analysis tool R for analysis and visualizing insights. **BUSINESS PROBLEM**

By analyzing socio-economic conditions of household's characteristics, governments can identify which households have the highest need for social welfare assistance. By accurately predicting the household poverty level, we can help other countries beyond the Costa Rican to assess the social need.

**PROJECT SYNOPSIS**

In this project paper, we have used various regularization and regression techniques to analyze the Costa Rican Household Poverty Level Prediction. We have focused on which households have the highest need for social welfare assistance. The project paper comprises of exploratory data analysis with visualizations, modeling techniques with regularization techniques, and lastly hyperparameter tuning with visualizations. In the current scenario, the government needs to identify where and what kind of needs of social welfare assistance are required. A brief about the dataset, so it is split into two, i.e., train and test set. If you have a look at both the set, the test dataset doesn't comprise of target column, which refers to the poverty level. Thus, selecting the train set, we proceeded with exploratory data analysis. Some columns which need to be considered on a priority basis are extreme poverty, moderate poverty, vulnerable households, and non-vulnerable households. ‘Idhogar’ is a unique identifier that is used for each household.

**CLEANING DATASET**

For handling the missing values, we have used two methods based on variables.

1. Replacing missing values with the mode of column or zero.

2. We observed that several family members do not have the same target variable. We solve this by giving all household members the same Target as that of the head of the house.

3. For the different variables, the yes and no values replaced by the numeric no's 0 and 1 based on their squared column.

4. All the eight squared columns have the squared values of the other columns. This indicates that these columns will be highly correlated with their original columns. So, we removed these squared columns. Now the columns are mainly categorized into 4 ways as ID columns, Individual, Household, and square columns.

**ANALYSIS**

**Correlation**:

For the data containing only heads were separated to check their behaviour, we plotted a correlation plot.

1. Code:

*corr1 <- c('r4t3', 'tamhog', 'tamviv', 'hhsize', 'hogar\_total')*

*nums <- unlist(lapply(full[,c('r4t3', 'tamhog', 'tamviv', 'hhsize', 'hogar\_total')], is.numeric))*

*c\_corr <- full[,corr1]*

*corr\_c <- cor(c\_corr)*

*corrplot(corr\_c, method = "square", tl.cex = 0.6, tl.offset = 0.4,tl.srt = 90, cl.ratio = 0.3)*

*ggcorr(full[, corr1])*

A picture containing white

Description automatically generated

As mentioned above to separate out the heads from the data. Here’s, the output below for the same:

A screenshot of a cell phone

Description automatically generated

From the above correlation plot we can make out that the columns r4t3, tamhog, tamviv, hhsize and hogar\_total is highly correlated but hhsize is highly correlated with tamhog and hogar\_total, thus we can remove them. As these represent the total number of persons in households, size of household, number of individuals in household and household size, thus as it is above the line and highly correlated, we need to remove them. Now, as there are two areas i.e. area 1 and area 2 i.e. urban or rural areas, we can remove any one of them. Refer code for the same below:

Code:

*full[,c('tamhog', 'hogar\_total', 'r4t3')] <- NULL*

1. After, we removed one area we divided the other into two i.e. negatives and positives of the house where negatives and positives consists of:

Code:

*full$area2 <- NULL*

*Negatives:* If there is no toilet, No electricity, No floor at house, If no water provision, and house with no ceiling.

Code:

*full$negative <- 1 \* (full$sanitario1 + (full$noelec == 1) + full$pisonotiene +*

*full$abastaguano + (full$cielorazo == 0))*

*Positives:* If there is refrigerator, if they own tablet, if they have television and computer.

Code:

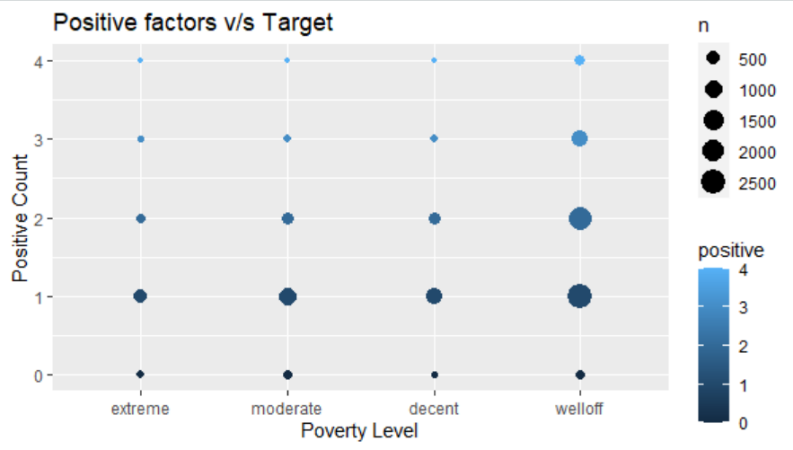
*full$positive <- 1 \* (full$refrig + (full$v18q1 > 0) + full$computer +full$television)*

Below are the interpretations for the same.

Code:

*ggplot(full, aes(x = Target, y = positive)) + geom\_count(aes(col = positive)) + labs(title = "Positive factors v/s Target", x = "Poverty Level", y = "Positive Count")*

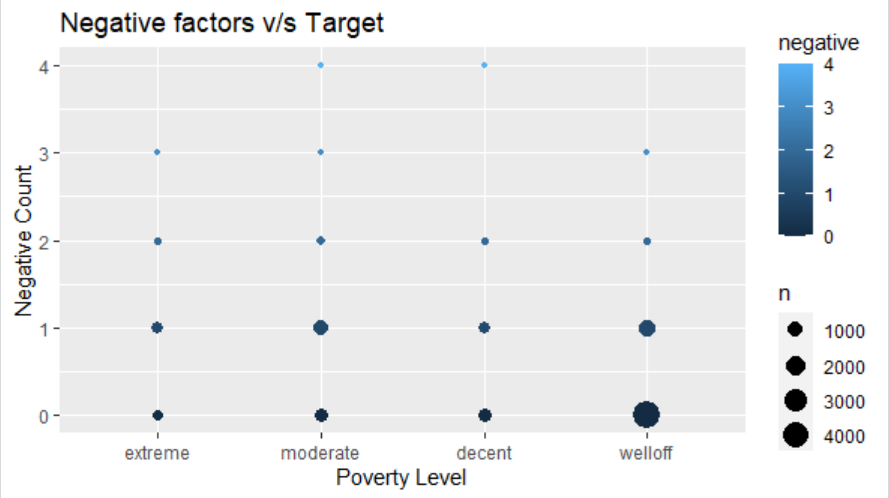
Output:



From the above graph, as you can see, the dots represent the number of families, while the color legend represents the level of poverty, where the level of poverty is divided into four i.e. extreme, moderate, decent, well off. If the count on the color legend is above 3, it means family lies in the positive section and has all the necessary things i.e. refrigerator, tablet, computer and television. Also, for reference, we can have a look at the last column, which demonstrates that the count is more than 2, so those families lie in the well-off poverty region.

Code:

ggplot(full, aes(x = Target, negative)) + geom\_count(aes(col = negative)) + labs(title = "Negative factors v/s Target", x = "Poverty Level", y = "



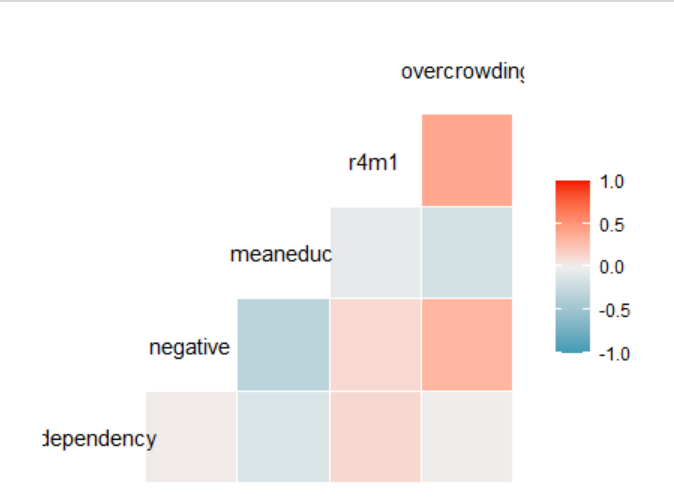
As mentioned above that the dots represent the number of families, and colour legend is for the level of poverty. So, in this case if the colour legend is above 3 it indicates that those families have the lesser chances of having no toilet, no electricity, no floor at house, no water provision or no ceiling in house. Thus, we have only considered the positive area and families. Also, if we want to consider it as a whole i.e. positive, negative, dependency and overcrowding we have designed a correlation matrix.

Code:

variables = c('dependency', 'negative', 'meaneduc', 'r4m1', 'overcrowding')

ggcorr(full[, variables])

Output:



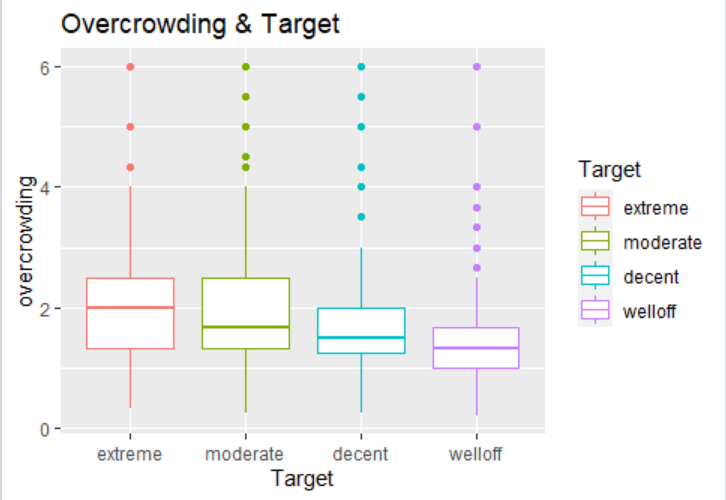
From the above correlation plot we can interpret that the positive families which has all the necessities or amenities lies above 0.5 while on the other hand the negative families without the amenities lies below -0.5. As mentioned above we are only considering the positive families. Now, the families consist male and female for which there are separate column in the dataset and, we are here removing the male column and considering only the female column.

Code:

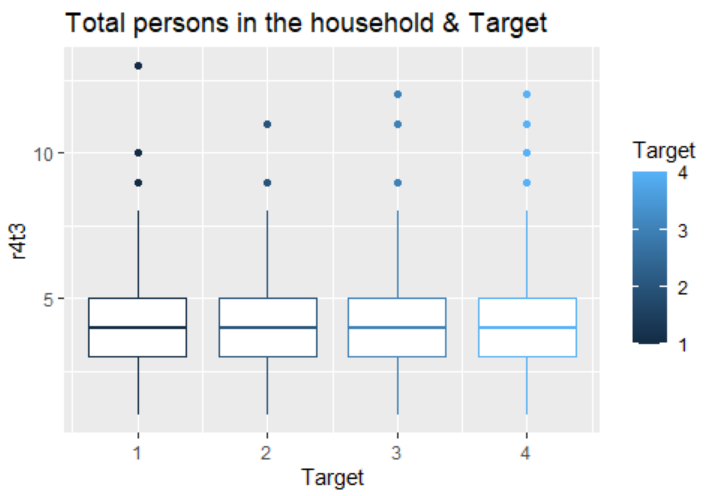
*full$male <- NULL*

*ggplot(full, aes(x = Target, y = overcrowding)) + geom\_boxplot(aes(group = Target, col = Target))+labs(title = "Overcrowding & Target")*

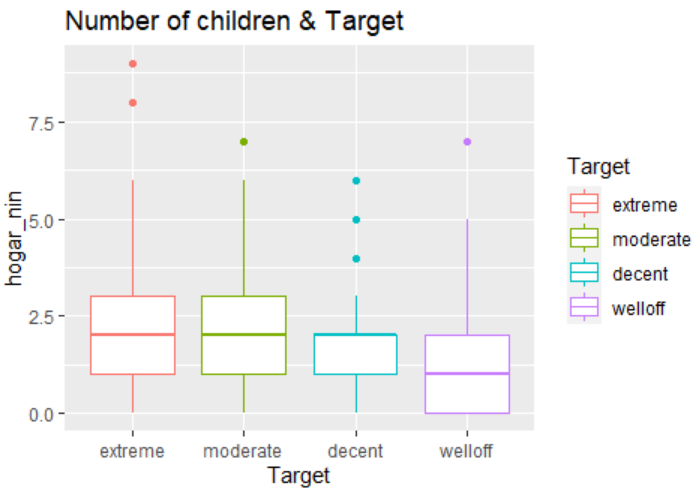
Output:



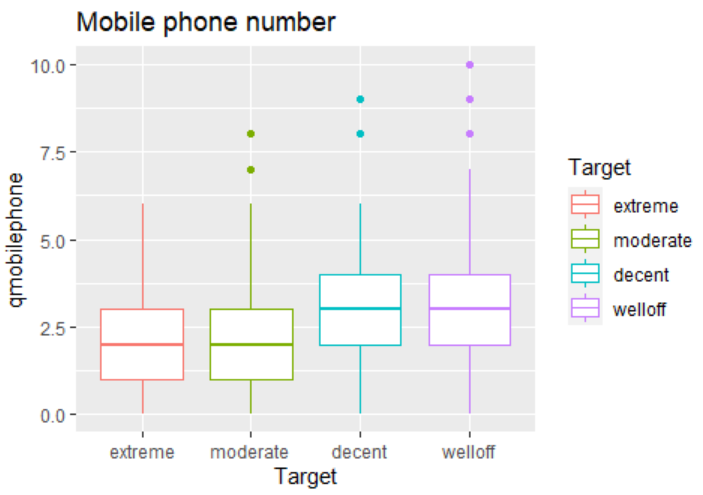
*ggplot(mytrain, aes(x = Target, y = r4t3)) + geom\_boxplot(aes(group = Target, col = Target))+labs(title = "Total persons in the household & Target")*

**

*ggplot(full, aes(x = Target, y = hogar\_nin)) + geom\_boxplot(aes(group = Target, col = Target))+labs(title = "Number of children & Target")*

**

*ggplot(full, aes(x = Target, y = qmobilephone)) + geom\_boxplot(aes(group = Target, col = Target))+labs(title = "Mobile phone number")*

**

The above box plots show the households the heads of the females are very much towards the severity level of poverty. It looks like every target the females are towards the severity level, which means that maximum households with female’s heads are more likely to be on severity level of poverty. This also shows that there must be some missing values and to overcome these missing values we can eliminate the unnecessary columns before modelling and then find out the correlation between the remaining.

Code:

*full$Id <- NULL*

*full$idhogar <- NULL*

*full$agesq <- NULL*

*full$elimbasu5 <- NULL*

*mytrain <- full*

*mytest <- full[9558:33413,]*

*dim(mytrain)*

[1] 9557 128

**K means Clustering:**

We seek to identify homogeneous subgroups within the data, such that data points in each cluster are as similar as possible to similarity metrics such as correlation-based space. Kmeans algorithm is an iterative algorithm that attempts to partition the dataset into Kpredefined independent nonoverlapping subgroups (clusters) where each data point belongs to one group only. The operations of the algorithm k-means the following way: **Calculating Distance Matrix:** Calculating Distance Matrix: Classification of findings into groups involves other techniques to measure the distance or dissimilarity between each pair. Choosing distance measurements is a critical step in the clustering process. It determines the measure of the similarity of two elements (x, y) and will affect the form of the clusters.

**Code:**

**#computing a distance matrix**

**> distance <- get\_dist(Sum\_DS)**

**#visualizing a distance matrix**

**fviz\_dist(distance, gradient = list(low = "green", mid = "white", high = "orange"))**

Output:

A picture containing screenshot

Description automatically generated

Fig.1 shows the distance matrix of the dataset.

Choosing distance measurements is very significant because it has a clear effect on the effects of the clustering. The default calculation of distance for most common clustering applications is the Euclidean interval.

**Determining Optimal Clusters using Elbow Method:**

Elbow approach provides us with an understanding of what a good k number of clusters will be based on the amount of squared distance (SSE) between data points and centroids of their allocated clusters. At the spot where SSE is starting to flatten out and shape an elbow, we choose k.

Firstly, we grouped data into two clusters, then into three and then into four. We plotted each cluster for comparison.

**Code:**

**# The first step is grouping the data into 2 clusters**

**Ds\_K2 <- kmeans(Sum\_DS,centers = 2, nstart = 25)**

**str(Ds\_K2)**

**# print the output1**

**Ds\_K2**

**# visualizing the output1**

**fviz\_cluster(Ds\_K2,data = Sum\_DS)**

**# Executing the same process for 3,4,5**

**Ds\_K3 <- kmeans(Sum\_DS,centers = 3, nstart = 25)**

**Ds\_K4 <- kmeans(Sum\_DS,centers = 4, nstart = 25)**

**#plots to compare**

**p2 <- fviz\_cluster(Ds\_K2,geom = "point",data = Sum\_DS)+ ggtitle("k=2")**

**p3 <- fviz\_cluster(Ds\_K3,geom = "point",data = Sum\_DS)+ ggtitle("k=3")**

**p4 <- fviz\_cluster(Ds\_K4,geom = "point",data = Sum\_DS)+ ggtitle("k=4")**

**library(gridExtra)**

**grid.arrange(p2,p3,nrow = 2)**

**Output:**

**A close up of a map

Description automatically generated**

Fig 2. K=4 clustering

Determining Optimal Number of Clusters:

**# Determining Optimal Clusters**

**# There are tree types to determine the optimal cluster, which includes:**

**# Elbow method / Silhouette method / Gap statistic**

**# Elbow method**

**fviz\_nbclust(Sum\_DS,kmeans,method = "wss")+**

**geom\_vline(xintercept = 4,linetype = 2)+**

**labs(subtitle = "Elbow method")**

**# Extracting Result**

**# Computing k-means clustering wiht k =4**

**set.seed(1)**

**Finalresult <- kmeans (Sum\_DS, 4, nstart = 25)**

**print(Finalresult)**

**fviz\_cluster(Finalresult, data = Sum\_DS)**

**Output:**

**A close up of a map

Description automatically generated**

Fig 3 optimal no of clusters

The above graph indicates k=4 isn't a bad alternative. Often it is also difficult to work out a sufficient number of clusters to use as the curve is diminishing monotonically and does not reveal any elbow or have a clear point where the curve starts flattening out.

**DECISION TREE:**

Using the same train and test dataset, we fit a decision tree model and below is the code used in R to do so:

Code:

**#Building decision tree**

**set.seed(1)**

**tree <- rpart(rtrain$Target~., data=rtrain)**

**## myAttrition = rtrain , Attrition = Target**

**#Draw the tree**

**prp(tree, type=3, tweak=0.8, main="Target", compress=TRUE )**

**Output:**

A close up of a map

Description automatically generated

Fig 3. Shows the decision tree model

**Decision Tree Accuracy:**

**> mean((as.numeric(Predict)-as.numeric(rtest$Target))^2)**

**[1] 1.012515**

**> sqrt(mean((as.numeric(Predict)-as.numeric(rtest$Target))^2))**

**[1] 1.006238**

**> table\_mat <- table(rtest$Target, Predict)**

**> accuracy\_Test <- sum(diag(table\_mat)) / sum(table\_mat)**

**> accuracy\_Test**

**[1] 0.6785295**

The Decision tree gives us the 67% accurate model for dataset. The decision tree plot in figure 3 Is the model output that was fit on the training data.

**RANDOM FOREST:**

**Why Random Forest?**

The decision tree is always fitted on the training data. In the training data, the decision tree is always installed. Overfitting occurs when we have a highly versatile model (the model has a high capacity) that basically memorizes the training data by closely fitting it. The inflexible model is known to be highly biased because it requires training data assumptions. An inflexible model may not be able to fit only the training data because in both cases— high variance because high bias — the model cannot generalize well into new data. We use random forests to surmount this problem.

**Code:**

**RF = randomForest(rtrain$Target ~ rtrain$meaneduc, data = rtrain)**

**RF**

**plot(RF, main = "")**

A screenshot of a social media post

Description automatically generated

**Fig. Random forest of training dataset**

**Call:**

**randomForest(formula = rtrain$Target ~ rtrain$meaneduc, data = rtrain)**

**Type of random forest: classification**

**Number of trees: 500**

**No. of variables tried at each split: 1**

**OOB estimate of error rate: 36.61%**

**Confusion matrix:**

**extreme moderate decent welloff class.error**

**extreme 34 101 0 446 0.94148021**

**moderate 23 185 11 971 0.84453782**

**decent 12 80 56 722 0.93563218**

**welloff 24 143 29 4163 0.04496444**

**> plot(RF, main = "")**

**> impVar = round(randomForest::importance(RF),2)**

**> impVar[order(impVar[,1], decreasing = TRUE),]**

**[1] 594.09**

**Tuning of RF:**

**A close up of a map

Description automatically generated**

**> tunedRF = tuneRF(rtrain[ , -1], rtrain[ ,1], stepFactor = 0.5, plot = TRUE, ntreeTry = 30, improve = TRUE)**  
mtry = 42  OOB error = 1556469703   
Searching left ...  
mtry = 84 OOB error = 1460372215   
0.06174067 TRUE   
Searching right ...  
mtry = 21 OOB error = 1743770832   
-0.1203372 TRUE

> tunedRF  
   mtry   OOBError  
21   21 1743770832  
42   42 1556469703  
84   84 1460372215

> #Confusion Matrix  
> t2 = table(rtrain$Target, predRF)  
Model Accuracy

> #RandomForest Model Accuracy  
> (t2[1] + t2[4])/(nrow(rtrain))  
[1] 0.009857143

**MULTINOMIAL LOGISTIC REGRESSION:**

Multinomial logistic regression is a simple extension of binary logistic regression allowing for more than two dependent or outcome variable types. Multinomial logistic regression, like binary logistics regression, uses maximum likelihood estimation to determine the probability of categorical membership. Multinomial logistic regression is used to estimate categorical placement in a dependent variable or the probability of group membership based on several independent variables that can be d discrete or continuous.

**Code:**

**ml\_fit <- multinom(Target~.,data = rtrain)**

**summary(ml\_fit)**

**save(ml\_fit,file = "ml\_fit.rda")**

**ml\_pred <- predict(ml\_fit, rtest)**

**head(ml\_pred)**

**accuracy <- mean(ml\_pred == rtest$Target)**

**accuracy**

**cm\_ml <- confusionMatrix(ml\_pred, rtest$Target)**

**Output:**

**ml\_fit <- multinom(Target~.,data = rtrain)**

**# weights: 560 (417 variable)**

**initial value 9701.287939**

**iter 10 value 7456.584399**

**iter 20 value 6999.710784**

**iter 30 value 6535.381624**

**iter 40 value 6418.836693**

**iter 50 value 6196.805302**

**iter 60 value 5849.895990**

**iter 70 value 5663.524817**

**iter 80 value 5526.011425**

**iter 90 value 5432.890710**

**iter 100 value 5395.765607**

**final value 5395.765607**

**stopped after 100 iterations**

**> summary(ml\_fit)**

**Call:**

**multinom(formula = Target ~ ., data = rtrain)**

**Residual Deviance: 10791.53**

**>**

**Interpretation:**

**Residual Deviance**: **10791.53** - Deviance is a measure of goodness of fit of a model. Higher numbers always indicate bad fit.

**AIC: 11487.53 -** The Akaike Information Criterion (AIC) is an estimator of the out-of-sample prediction error and thus the relative accuracy of the statistical models for a given collection of data. ... AIC measures the relative amount of information lost by a particular model: the less information a model loses, the higher the model's efficiency.

> accuracy <- mean(ml\_pred == rtest$Target)  
> accuracy

**[1] 0.6922174**

The results show the logistic coefficient for each predictor variable for each alternative category of the outcome variable; alternative category meaning, not the reference category. The logistic coefficient is the expected amount of change in the logit for each one unit change in the predictor. The logit is what is being predicted; it is the odds of membership in the category of the outcome variable which has been specified. The closer a logistic coefficient is

**GRADIENT BOOSTING METHOD:**

**Running Multiple GBMS:**

Each new tree matches to an updated version of the original data set while boosting. It begins by training a decision tree that assigns an equal weight to each observation. We increase the weights of those findings that are difficult to classify and lower the weights for those that are easy to classify after evaluating the first tree. Therefore, the second tree is grown on that weighted data. The purpose here is to boost the first tree's predictions. So, our new layout is Tree 1 + Tree 2. We then calculate this new 2-tree ensemble model's classification error and develop a third tree to predict the revised residuals. For a given number of iterations, we repeat this cycle. Subsequent trees enable us to identify findings that have not been well identified by the trees before. Therefore, the predictions of the final ensemble model are the weighted sum of the predictions of the previous three models.

GBM 1: Code:

**GBM1<- gbm(**

**formula = Target ~ .,**

**distribution = "gaussian",**

**data = rtrain,**

**n.trees = 100,**

**interaction.depth = 1,**

**shrinkage = 0.001,**

**cv.folds = 5,**

**n.cores = NULL,**

**verbose = FALSE**

**)**

**# Summary gives a table of Variable Importance and a plot of Variable Importance**

**summary(GBM1)**

**#find index for n trees with minimum CV error**

**cv.error = sqrt(min(GBM1$cv.error)); cv.error**

**[1] 1.010776**

**# get MSE and compute RMSE**

**idx\_min\_MSE <- which.min(GBM1$cv.error)**

**sqrt(GBM1$cv.error[idx\_min\_MSE])**

**[1] 1.010776**

**Output:**

**A screenshot of a social media post

Description automatically generated A screenshot of a cell phone

Description automatically generated**

**GBM2:**

**GBM2 <- gbm(**

**formula = Target ~ .,**

**distribution = "gaussian",**

**data = rtrain,**

**n.trees = 1000,**

**interaction.depth = 5,**

**shrinkage = 0.001,**

**cv.folds = 5,**

**n.cores = NULL,**

**verbose = FALSE**

**)**

**summary(GBM2)**

**cv.error = sqrt(min(GBM2$cv.error)); cv.error**

**[1] 0.8807077**

**> idx\_min\_MSE <- which.min(GBM2$cv.error)**

**> sqrt(GBM2$cv.error[idx\_min\_MSE])**

**[1] 0.8807077**

**> gbm.perf(GBM2, method = "cv")**

**[1] 1000**

**Output:**

**A screenshot of a social media post

Description automatically generatedA picture containing screenshot

Description automatically generated**

**GBM 3:**

**GBM3 <- gbm(**

**formula = Target ~ .,**

**distribution = "gaussian",**

**data = rtrain,**

**n.trees = 600,**

**interaction.depth = 5,**

**shrinkage = 0.05,**

**cv.folds = 5,**

**n.cores = NULL,**

**verbose = FALSE**

**)**

**summary(GBM3)**

**Squareed error GMB3:**

**cv.error = sqrt(min(GBM3$cv.error)); cv.error**

**[1] 0.6638871**

**> idx\_min\_MSE <- which.min(GBM3$cv.error)**

**> sqrt(GBM3$cv.error[idx\_min\_MSE])**

**[1] 0.6638871**

**> gbm.perf(GBM3, method = "cv")**

**[1] 600**

**A screenshot of a social media post

Description automatically generatedA picture containing screenshot

Description automatically generated**

**Visualization of Variable Importance:** Variable Importance plot indicates the importance of the predictors in the model. From this plot we see that the dependency, SQBmeaned, meaneduc, hogar\_nin explains maximum variance in the data and the accuracy for our model is **0.9264** for the validation dataset."

**A screenshot of a cell phone

Description automatically generated**

**FINAL GBM:**

**gbm.final <- gbm(# gbm.white.final <- gbm.final**

**formula = Target ~ .,**

**distribution = "gaussian",**

**data = rtrain,**

**n.trees = 4926,**

**interaction.depth = 5,**

**shrinkage = 0.01,**

**cv.folds = 5,**

**n.minobsinnode = 5,**

**bag.fraction = .8,**

**n.cores = NULL, # will use all cores by default**

**verbose = FALSE**

**)**

**summary(gbm.final)**

**> cv.error = sqrt(min(gbm.final$cv.error)); cv.error**

**[1] 0.659589**

**> idx\_min\_MSE <- which.min(gbm.final$cv.error)**

**> idx\_min\_MSE**

**[1] 4926**

**> sqrt(gbm.final$cv.error[idx\_min\_MSE])**

**[1] 0.659589**

**> gbm.perf(gbm.final, method = "cv")**

**[1] 4926**

**Output:**

**A screenshot of a social media post

Description automatically generated A picture containing screenshot, map

Description automatically generated**

**Interpretation:** The predictions of the final ensemble model are the weighted sum of the predictions of the previous three models.In each iteration of GBM, we see that error rate is decreasing in each of the iteration of GBM. Gradient Boosting Method is a machine learning technique for classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by al-lowing optimization of an arbitrary differentiable loss function. The purpose of boosting is to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, thereby producing a sequence of weak classifiers. It inherits all the good features of 10 trees and improves on the weak features such as prediction performance.

**CONCLUSION**

Based on the accuracy rate, sensitivity, and specificity, we can propose GBM model to inter-American development banks, which will help them to analyze socio-economic conditions on household characteristics, and governments can identify which households have the highest need for social welfare assistance. They can allocate funds for such people who are below the poverty line. This solution will also help other countries beyond the Costa Rican for assessing the social need.

**REFERENCE**

* Data Analysis & Exploratory Data Analysis (EDA). (n.d.). Retrieved March 3, 2019, from <https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/data-analysis/>
* Costa Rican Household Poverty Level Prediction. (n.d.). Retrieved June 1, 2019, from

<https://www.kaggle.com/c/costa-rican-household-poverty-prediction/data>.