I have always wondered what attributes are considered when providing aid. I personally required the help of grants and student loans in order to complete my degree in Information Systems. The Income Qualification project proved to be the most appealing.

Let’s jump right in!

First, we need to import the necessary libraries. Then, create data frames from the provided data files so we can manipulate the data as needed.

A picture containing rectangle

Description automatically generated

No output for these codes.

Now we are ready to begin the process required to build a machine learning model.

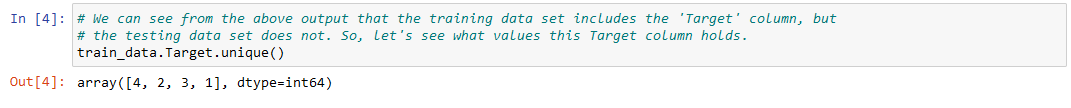
**Q1: Identify the output variable.**

First, we need to identify our output variable, which is the value that indicates how each record is classified based on the other values in the record. In this case, this is the value that will determine whether or not a family’s income, or actually the lack thereof, will qualify them for aid.

Graphical user interface, text, application

Description automatically generated

We can see from the above output that the training data set includes the ‘Target’ column, but the testing data does not. So, let’s see what unique values this column holds.



The limited values for the output variable indicate the learning model will be based on classification.

**Q2: Understand the type of data.**

To confirm our guess that the model will be a classification model, we need to further explore the data and data types included in our data sets.

A picture containing text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

It looks like we have a mix of data types, and as we saw when we pulled the info for each data set, we know we have integers, floats and objects. However, we don't know which columns are which. Let's get that info now.

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated

We can also see we have null values in some of the columns, but we'll address those later.

**Q3: Check if there are any biases in your dataset.**

Since our target variable is an explicit set of values, we will look at how our data is distributed across these values. If our train data is not evenly distributed across all target values, the data set is biased.

Text

Description automatically generated

It appears we may have biased data just from looking at the numbers, but let's look at this visually just to confirm.

Chart

Description automatically generated

We can see how skewed the distribution is across the different target values. Our data is biased.

**Q4: Check whether all members of the house have the same poverty level. (Code included with 5 & 6)**

**Q5: Check if there is a house without a family head. (Code included with 4 & 6)**

**Q6: Set poverty level of the members and the head of the house within a family.**

We could loop through the houses 3 different times to answer questions 4 – 6. However, to be more efficient with the code, we will only loop through once and get all 3 values. While we’re at it, we will store the unique household identifier as we may need to use it later.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

**Q7: Count how many null values are existing in columns.**

We need to check both train and test data since test data is just as important as the train data. Starting with the train data, we will find the nulls and determine what to do with them: fill or drop. Then, we'll do the same with the test data to clean it up as well.

Text

Description automatically generated Graphical user interface, application

Description automatically generated

Let's explore the columns with null values and determine the best way to handle them. First, check the data dictionary to understand what information each column contains.

* v2a1 - Monthly rent payment
* v18q1 - number of tablets household owns
* rez\_esc - Years behind in school
* meaneduc - average years of education for adults (18+)
* SQBmeaned - square of the mean years of education of adults (>=18) in the household
* Target - our output variable that indicates poverty level of the household

Next, let's see if any of these columns correspond with other columns in a way that would explain the null values. This will help us decide how to handle them.

A picture containing text

Description automatically generated

From the results, we see that rent is null whenever the home is owned (tipovivi1 = 1), or ownership is precarious (tipovivi1 = 1), or the house is assigned or borrowed (tipovivi1 = 1). So, it's safe to replace these null values with 0.

Text

Description automatically generated

Second column with null values is v18q1 (number of tablets household owns). Only other column related to tablets is v18q (member of household owns tablet). Since v18q does not have any null values, let's pull both columns for all rows where v18q1 is null.

Text

Description automatically generated with medium confidence

The results show us that a household member tablet indicator is only null when there are 0 tablets in the household. So, it is safe to replace these null values with 0.

Text

Description automatically generated

The third column with null values is rez\_esc (Years behind in school). Since there are many reasons why a person may be behind in school, let's peek at the raw data where rez\_esc is null to check for obvious patterns.

Table

Description automatically generated

The results vary widely. So, let's see if rez\_esc is ever 0.

Graphical user interface, application

Description automatically generated

There are 1,211 records where the rez\_esc value is 0, and again, no obvious patterns. So, let's look at age for null records to see if there seems to be a pattern there.

Graphical user interface, text, application

Description automatically generated

It seems the records apply to household members who are too young or old to attend primary or secondary. Let's test this by looking at the key indicators for age where rez\_esc is not null.

Graphical user interface, text, application

Description automatically generated

When rez\_esc is populated, the age range is between 7 and 17. So, let's see how many records we have where rez\_esc is null and age is between 7 and 17.

Graphical user interface, text, application

Description automatically generated

From previous queries, we know there are 7,928 records where rez\_esc is null. However, the above check only returned 7,927. We know that these records represent people who are too young or old for primary/secondary school. So, it is logical that the rez\_esc should be 0 instead of null. Let's go ahead and update these specific records.

Text

Description automatically generated with medium confidence

As expected only 1 null rez\_esc record is left to clean up. Let's look at the age of the person represented in this row and grab the household id and info about other members as well.

Graphical user interface

Description automatically generated with low confidence

Now we know the age of the member is 10 and have the household id and know there are also 2 other members in the house. So, we can check those records and pull a bit more info also. Since the other records where rez\_esc was null represented people that aren't in school, it would be logical to infer this person is also not in school. Let's look at some other columns that may explain why this 10-year-old child is not in school... or if the null value should be 0 also.

A picture containing text

Description automatically generated

The child is disabled. So, this is why they are not in school. We can safely set rez\_esc to 0.

Graphical user interface, text, application

Description automatically generated

Moving on to meaneduc (average years of education for adults (18+)). We know there are 5 records with null values for this column. We also know we have 5 records where the square of mean education () is null. However, we do not know if they are the same 5 records in both cases. Let's check.

Text

Description automatically generated with medium confidence

As expected, both columns are null on the same 5 records. Next, we'll look at other columns that relate to education to see if there is a reason the mean education is null. We know which columns edjefe and edjefa are derived from, so we will include those in the data set also.

A picture containing table

Description automatically generated

Finally, we can clean up meaneduc and SQBmeaned null values. We can see from the above results that we have three unique households that have null meaned columns. We can also see that the entire household is represented in the results as well since the number of records for each household matches the value in hogar\_total (total individuals in the household). We will simply calculate the meaneduc value for each of the households. Then, do the same for the squared value. No output for this code.

A picture containing text

Description automatically generated

Let’s double check the updated households to make sure records are populated as expected.

Text

Description automatically generated with low confidence

It is apparent from the results above we have mixed values for edjefe and edjefa, some values are string values, and some are numeric. So, we need to fix that, so the data is consistent. The data dictionary defines the columns as follows:

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

We will set the columns to 0 for 'no' values, and 1 for 'yes' values. No output for this code.

Scatter chart

Description automatically generated

Let’s double check everything for edjefe and edjefa.

Graphical user interface, text, application

Description automatically generated

Now, we can see that there are values other than 0 and 1. We know from previous queries that these values match the number of years of schooling for the household member. So, it is safe to set these values to 1. Then, double check again to make sure the update was successful.

Graphical user interface, text, application, chat or text message

Description automatically generated

Let’s do one last check for columns with mixed values. We are looking for fields that are strings and are not unique identifier type fields. We could generate dummy values for these fields, but I prefer seeing what exactly I’m dealing with first.

Graphical user interface, text, application, Word

Description automatically generated

The only column still containing string data is the dependency column. Let's see what data it contains.

Chart, scatter chart

Description automatically generated

The data is mostly float values, but there are also yes and no values. We need to fix this. So, we will update the 'no' values to 0 and the 'yes' values to 1.

Chart, scatter chart

Description automatically generated

Now, let's take the same steps to clean up our test data. First looking for columns with null values for all three data types we observed in the training data set.

Graphical user interface, text

Description automatically generated

Replace the null values for v2a1 and double check.

Graphical user interface, text, application

Description automatically generated

Replace the null values for v18q1.

Text

Description automatically generated

Replace the null values for rez\_esc for people who are not school aged or disabled.

Text

Description automatically generated

Replace the null values for meaneduc and SQBmeaned by calculating the values.

Text

Description automatically generated with medium confidence

Clean up the mixed data for edjefe and edjefa.

Text

Description automatically generated

A picture containing text

Description automatically generated

Clean up the mixed data for dependency.

Scatter chart

Description automatically generated

The last thing we need to do is eliminate columns that provide no real value when creating a training model. Columns which have unique IDs or index values can be eliminated, as well as columns which essentially duplicate data found in other columns. After reviewing the data dictionary, I was able to classify the following:

* Number of People in household to compare and drop all but one: r4t3, tamviv, hogar\_total, tamhog, hhsize

Graphical user interface

Description automatically generated with medium confidence

From the results above, we can see that all of the columns related to number of people in the household or household size all contain the exact same data. So, we can drop all but one. Let's keep hogar\_total since we used that field previously.

Graphical user interface, text, application, email

Description automatically generated

Next, we'll drop the id and squared values since they provide no real value as far as training a machine learning model is concerned.

* id values: Id, idhogar
* squared values: SQBescolari, SQBage, SQBhogar\_total, SQBedjefe, SQBhogar\_nin, SQBovercrowding, SQBdependency, SQBmeaned, agesq

Text

Description automatically generated with medium confidence

**Q8: Remove null value rows of the target variable.**

Next, we will check for any records that have a null Target value in our train data and remove them. We will not need to do this step with our test data since our test data does not have the Target column. First, get a count of affected rows to see if this is even an issue for us.

Application, table

Description automatically generated with medium confidence

We do have 250 records with a Target value of null. So, we’ll drop these records, and then double check the drop was successful.

Text

Description automatically generated

**Q9: Predict the accuracy using random forest classifier.**

We are finally done wrangling and cleaning our data sets. The next step is to create our train and test samples. While you may be thinking, "But we already have train and test data sets". Let me explain. When we create a learning model, we build it on the training data set using X and Y, where X is the data frame with all columns except for the target variable and Y is the data frame with only the target variable. Then, the test data set is split off of that to create our X and Y for testing the model against. Our test data does not have the target variable. Therefore, we must split our training data into training and testing. Only after this is complete, and the learning algorithm prediction score satisfactory, can we run our model against the test data to predict the target variable value for each record in the testing data set. We will first split our data set and then create separate data frames. One for all columns except the target variable and one with only the target variable. No output for this code.

**Text

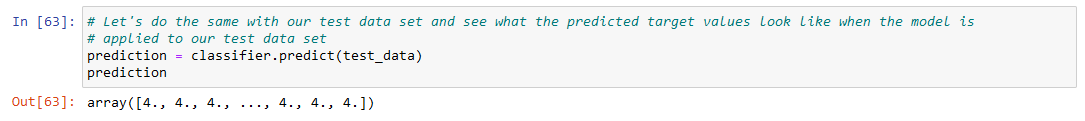
Description automatically generated**

Now, finally, let's create our learning model using random forest classifier. Random forest model is used because it is more accurate than a decision tree model. This is because the random forest classifier uses an ensemble of decision tree models and compiles the results from all to generate a more accurate prediction.

Graphical user interface, text, application, email

Description automatically generated

Just for fun, let’s also look at the target values predicted when running the model against our test data.



**Q10: Check the accuracy using random forest with cross validation.**

Lastly, we will use random forest with cross validation. To explain this concept, think back to how we set up our data by splitting the data into train and test data sets for our random forest classification learning model. Random forest only splits the data into one single train/test data split. Random forest with cross validation splits the main data set into the given number of folds or randomly selected, but equal sized subsets of data. Then, each subset of data is split into train/test models. This provides even more accuracy to the predictions. Since our data set is biased, random forest with cross validation learning model will most likely provide the greatest accuracy of predictions. We will first use the default number of splits and 10 trees. Then, check the mean accuracy of the folds.

Text

Description automatically generated

Our accuracy with the default number of folds is 90.7%. Let's see if we achieve a higher degree of accuracy with the same number of splits and 100 trees. Then, check the mean accuracy of the folds so we can compare to the accuracy with only 10 trees.

Text

Description automatically generated with medium confidence

**Final thoughts:**

Our prediction accuracy dropped when comparing random forest classification to random forest classification with cross validation. Both iterations of testing the latter resulted in basically the same accuracy level. However, I believe the similarity between these two is likely attributed to the extreme bias in our original data set. I also believe that while the prediction accuracy is slightly lower with the cross validation, it is likely more of a true accuracy due to the cross validation and random nature of the record selections.