PROJECT REPORT

Data Mining and Knowledge Discovery

IM672A

Project: **Costa Rican Household Poverty Level Prediction**

Investigator: Prof. Faiz Hamid

Prepared by: Subham Goyal (13714), Lingam Deepak (14354), Apoorv Krishna (160142) **- Group 1**

Problem and Data Explanation

The data for this competition is provided in two files: `train.csv` and `test.csv`. The training set has 9557 rows and 143 columns while the testing set has 23856 rows and 142 columns. Each row represents **one individual** and each column is a **feature, either unique to the individual, or for the household of the individual.** The training set has one additional column, `Target`, which represents the poverty level on a 1-4 scale and is the label for the competition. A value of 1 is the most extreme poverty.

This is 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

Objective

The objective is to predict poverty on a **household level**. We are given data on the individual level with each individual having unique features but also information about their household. In order to create a dataset for the task, we'll have to perform some aggregations of the individual data for each household. Moreover, we have to make a prediction for every individual in the test set, but *"ONLY the heads of household are used in scoring"* which means we want to predict poverty on a household basis.

The `Target` values represent poverty levels as follows:

```
1 = extreme poverty
2 = moderate poverty
3 = vulnerable households
4 = non vulnerable households
```

Core Data fields

- **Id**: a unique identifier for each individual, this should not be a feature that we use!
- **idhogar**: a unique identifier for each household. This variable is not a feature but will be used to group individuals by the household as all individuals in a household will have the same identifier.
- **parentesco1**: indicates if this person is the head of the household.
- **Target**: the label, which should be equal for all members in a household

When we make a model, we'll train on a household basis with the label for each household *the poverty level of the head of household*. The raw data contains a mix of both household and individual characteristics and for the individual data, we will have to find a way to aggregate this for each household. Some of the individuals belong to a household with *no head of the household* which means that unfortunately, we can't use this data for training. These issues with the data are completely typical of **real-world** data and hence this problem is great preparation for the datasets you'll encounter in a data science job!

Preprocessing

Imports

We'll use a familiar stack of data science libraries: `Pandas`, `numpy`, `matplotlib`, `seaborn`, and eventually `sklearn` for modeling.

In [1]:
```
import pandas as pd
import numpy as np
```

Read in Data and Look at Summary Information

In [2]:
```python
pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', 500)
```

In [3]:
```python
train=pd.read_csv('train.csv')
train
```

1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	1	1	0	0	0
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	0	0	0	1	1
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	2	2	1	1	2
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	2	2	1	1	2
5	ID_ec05b1a7b	180000.0	0	5	0	1	1	1	1.0	0	2	2	1	1	2
6	ID_e9e0c1100	180000.0	0	5	0	1	1	1	1.0	0	2	2	1	1	2
7	ID_3e04e571e	130000.0	1	2	0	1	1	0	NaN	0	1	1	2	1	3
8	ID_1284f8aad	130000.0	1	2	0	1	1	0	NaN	0	1	1	2	1	3
9	ID_51f52fdd2	130000.0	1	2	0	1	1	0	NaN	0	1	1	2	1	3
10	ID_db44f5c59	130000.0	1	2	0	1	1	0	NaN	0	1	1	2	1	3
11	ID_de822510c	100000.0	0	3	0	1	1	0	NaN	0	0	0	0	2	2
12	ID_d94071d7c	100000.0	0	3	0	1	1	0	NaN	0	0	0	0	2	2

Missing Variables

One of the most important steps of exploratory data analysis is finding missing values in the data and determining how to handle them. Missing values have to be filled in before we use a machine learning model and we need to think of the best strategy for filling them in based on the feature: this is where we'll have to start digging into the data definitions.

First, we can look at the percentage of missing values in each column.

In [50]:
```
#code for number of null values in a column
for i in list(final.columns):
    count=0
    for j in  list(final[i].isnull()):
        if j is True:
            count=count+1
    if count != 0:
        print(i,'-',count)
```

```
edu_avg_above_18 - 5
```

In [51]:
```
final.loc[(final['edu_avg_above_18'].isnull() ),['ID','age','gender','household_size','Household_le
```

Out[51]:

	ID	age	gender	household_size	Household_level_identifier	education_level	edu_male_head_years	edu_
1291	ID_bd8e11b0f	18	2	1	1b31fd159	4	0	
1840	ID_46ff87316	18	2	2	a874b7ce7	3	4	
1841	ID_69f50bf3e	18	1	2	a874b7ce7	2	4	
2049	ID_db3168f9f	19	1	2	faaebf71a	13	12	
2050	ID_2a7615902	19	1	2	faaebf71a	13	12	

In [52]:
```
final.loc[(final['ID']=='ID_bd8e11b0f' )]=final.loc[((final['household_size']==1) & (final['Target'
final.loc[(final['ID']=='ID_46ff87316' )]=final.loc[((final['household_size']==2) & (final['Target'
final.loc[(final['ID']=='ID_69f50bf3e' )]=final.loc[((final['household_size']==2) & (final['Target'
final.loc[(final['ID']=='ID_db3168f9f' )]=final.loc[((final['household_size']==2) & (final['Target'
final.loc[(final['ID']=='ID_2a7615902' )]=final.loc[((final['household_size']==2) & (final['Target'
```

In [53]:
```
final.loc[[1291,1840,1841,2049,2050],'edu_avg_above_18']
```

Out[53]:
```
1291    10.0
1840    10.0
1841    10.0
2049    10.0
2050    10.0
Name: edu_avg_above_18, dtype: float64
```

In [54]:
```
#code for number of null values in a column
for i in list(final.columns):
    count=0
    for j in  list(final[i].isnull()):
        if j is True:
            count=count+1
    if count != 0:
        print(i,'-',count)
```

In [55]:
```
final.to_csv('final.csv',index=False)
```

As we can see different readings for final people age.

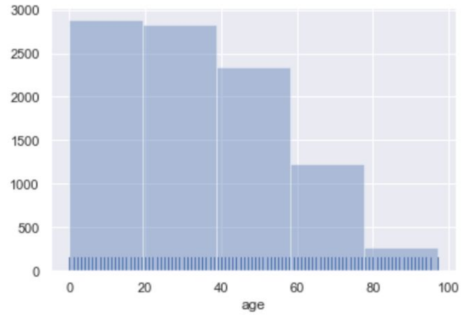
In [127]:
```
final['age'].describe()
```

Out[127]:
```
count    9557.000000
mean       34.299152
std        21.616346
min         0.000000
25%        17.000000
50%        31.000000
75%        51.000000
max        97.000000
Name: age, dtype: float64
```

To get a much better idea of this data, we plot the distribution as well as a box plot of people age.

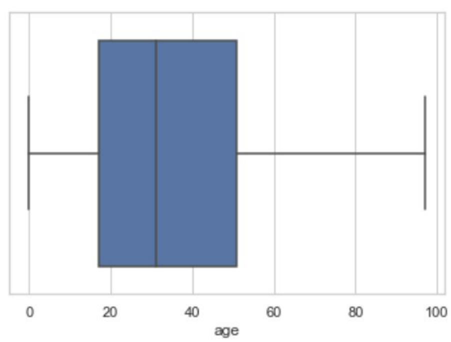
Distribution plot

```
In [128]:  sns.distplot(final['age'], bins=5, kde=False, rug=True);
```



Box plot

```
In [60]:  sns.set(style="whitegrid")
          ax = sns.boxplot(x=final['age'])
```



Similary, we plot the graphs for other columns like `‘years_of_schooling’`, `‘disable_person’`, `‘marital_status’`, `'in_house_position'`, `'education_level'`, `'region'`, `'area'`, `'house_owned_status'`, `'monthly_rent_payment'`, `'bathrooms'`, `'bedrooms'`, `'condition_walls'`, `'condition_floor'`, `'condition_roof'`, `'electricity'`, `'household_size'`, `'males_younger_12'` etc

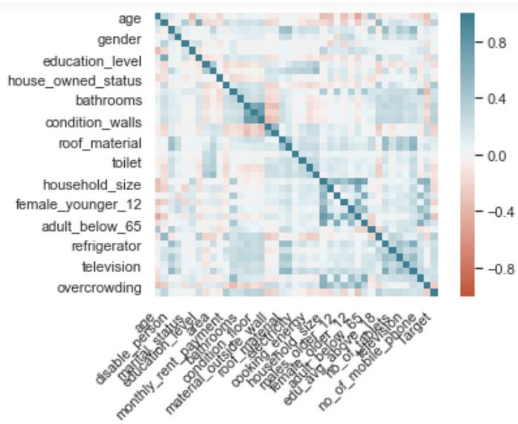
Correlation matrix of redundant household variables

Let's take a look at the correlations between all of the household variables. If there are any that are too highly correlated, then we might want to remove one of the pair of highly correlated variables. The following code identifies any variables with a greater than 0.95 absolute magnitude correlation.

```
In [221]: corr_matrix=final.corr()
```

```
In [225]: corr_matrix['adult_above_65'].sort_values(ascending=False)
```

```
marital_status          0.054232
roof_material           0.031588
female_older_12         0.027880
condition_roof          0.027477
refrigerator            0.024403
rubbish_disposal        0.010218
water_provision         0.008099
cooking_energy          0.007510
condition_walls         0.001337
Target                 -0.005159
gender                 -0.005445
electricity            -0.013530
material_outside_wall  -0.021471
education_level        -0.024308
area                   -0.035689
condition_floor        -0.040639
computer               -0.044713
years_of_schooling     -0.047996
monthly_rent_payment   -0.048172
no_of_tablets          -0.048894
floor_material         -0.049290
```




```
In [133]: final['condition_walls'].corr(final['roof_material'])
Out[133]: 0.21872491561999677
```

Data Modelling

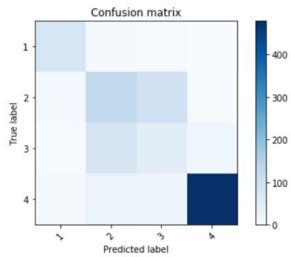
Now that we have a good set of features, it's time to get into the modelling.

1. **Classification Method** - We used two classification method Gini and entropy to predict poverty on a **household level**.

a. **Gini-** In this case, our accuracy is 72.7%.

```
In [6]:  clf_entropy = DecisionTreeClassifier( criterion = "entropy", max_depth = None)
         clf_entropy.fit(x_train, y_train)
         result=clf_entropy.predict(x_test)
         cal_accuracy(y_test, result)
```

```
Confusion Matrix:
 [[ 75   5   2   0]
 [  8 125  96   2]
 [  3  82  49  17]
 [  6  28  24 478]]
```

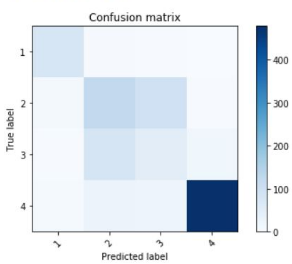



```
Accuracy :  72.7
Sensitivity for 1 :  91.46341463414635
Sensitivity for 2 :  54.112554112554115
Sensitivity for 3 :  32.450331125827816
Sensitivity for 4 :  89.17910447761194
```

b. **Entropy:** In this case, our accuracy is 72.7%, Which is not good as compare to Gini.

```
In [6]:  clf_entropy = DecisionTreeClassifier( criterion = "entropy", max_depth = None)
         clf_entropy.fit(x_train, y_train)
         result=clf_entropy.predict(x_test)
         cal_accuracy(y_test, result)
```

```
Confusion Matrix:
 [[ 75   5   2   0]
 [  8 125  96   2]
 [  3  82  49  17]
 [  6  28  24 478]]
```

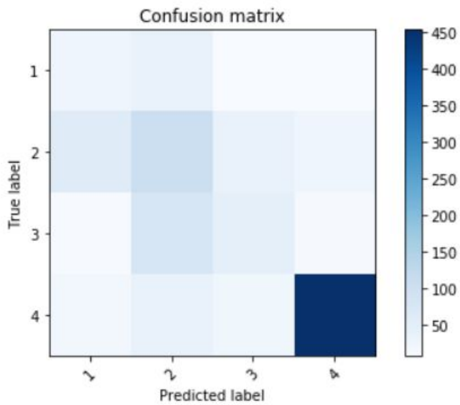



```
Accuracy :  72.7
Sensitivity for 1 :  91.46341463414635
Sensitivity for 2 :  54.112554112554115
Sensitivity for 3 :  32.450331125827816
Sensitivity for 4 :  89.17910447761194
```

2. **Logistic Regression:** In this case, our prediction accuracy came down to 63.5% then the classification methods we used above. We applied this method twice and observe a

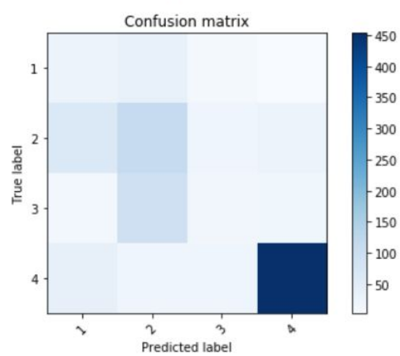
drop in the accuracy level.

```
Confusion Matrix:
 [[ 28  38   8   8]
 [ 62 104  39  26]
 [ 10  79  49  13]
 [ 20  38  24 454]]
```




```
Accuracy :  63.5
Sensitivity for 1 :  34.146341463414636
Sensitivity for 2 :  45.02164502164502
Sensitivity for 3 :  32.450331125827816
Sensitivity for 4 :  84.70149253731343
```

```
Confusion Matrix:
 [[ 28  38  13   3]
 [ 66 115  21  29]
 [ 16  97  18  20]
 [ 39  23  21 453]]
```




```
Accuracy :  61.4
Sensitivity for 1 :  34.146341463414636
Sensitivity for 2 :  49.78354978354979
Sensitivity for 3 :  11.920529801324504
Sensitivity for 4 :  84.51492537313433
```

3. **Neural Network:** In this case, our prediction accuracy was varying after observing this model for 1000 times. The accuracy that we observe is 68.4%.

Results

Here we can see the accuracy of all the methods used for modelling data.

Decision Tree with GINI

	1	2	3	4
0	78	2	1	1
1	9	126	89	7
2	6	56	73	16
3	18	33	16	469

Accuracy : 74.6

Logistic Regression with Multi class Classfication

	1	2	3	4
0	28	38	8	8
1	62	104	39	26
2	10	79	49	13
3	20	38	24	454

Accuracy : 63.5

Decision Tree with Entropy

	1	2	3	4
0	75	5	2	0
1	8	125	96	2
2	3	82	49	17
3	6	28	24	478

Accuracy : 72.7

Nural Network with Multi Class Classfication

	1	2	3	4
0	49	17	15	1
1	12	94	71	54
2	15	28	56	52
3	19	21	11	485

Accuracy : 68.4

The poverty prediction level accuracy of households for the following methods are:

Methods	Accuracy (%)
Gini	74.6
Entropy	72.7
Logistic Regression	63.5
Neural Network	68.4

Although I observe the accuracy level in the case of the neural network sometimes come more than **Gini** method. But I would recommend using Gini method for predicting the poverty of households. As currently, according to me this method is way better than the other three with an accuracy of **74.6%**.