Project- Income Qualification with Machine Learning

DESCRIPTION

Identify the level of income qualification needed for the families in Latin America.

Problem Statement Scenario:

Many social programs have a hard time ensuring that the right people are given enough aid. It's tricky when a program focuses on the poorest segment of the population. This segment of the population can't provide the necessary income and expense records to prove that they qualify.

In Latin America, a popular method called Proxy Means Test (PMT) uses an algorithm to verify income qualification. With PMT, agencies use a model that considers a family's observable household attributes like the material of their walls and ceiling or the assets found in their homes to classify them and predict their level of need.

While this is an improvement, accuracy remains a problem as the region's population grows and poverty declines.

The Inter-American Development Bank (IDB)believes that new methods beyond traditional econometrics, based on a dataset of Costa Rican household characteristics, might help improve PMT's performance.

Following actions should be performed:

1.Identify the output variable. 2.Understand the type of data. 3.Check if there are any biases in your dataset. 4.Check whether all members of the house have the same poverty level. 5.Check if there is a house without a family head. 6.Set poverty level of the members and the head of the house within a family. 7.Count how many null values are existing in columns. 8.Remove null value rows of the target variable. 9.Predict the accuracy using random forest classifier. 10.Check the accuracy using random forest with cross validation.

Importing the libraries

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
import collections
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classificat
ion_report
```

In [3]:
```
#import the file using pandas

df=pd.read_csv('train-IQ.csv')
print('Shape of the data',df.shape)
print()
print(df.head())
```

```
Shape of the data (9557, 143)

             Id      v2a1  hacdor  rooms  hacapo  v14a  refrig  v18q  v
18q1  \
0  ID_279628684  190000.0       0      3       0     1       1     0
NaN
1  ID_f29eb3ddd  135000.0       0      4       0     1       1     1
1.0
2  ID_68de51c94       NaN       0      8       0     1       1     0
NaN
3  ID_d671db89c  180000.0       0      5       0     1       1     1
1.0
4  ID_d56d6f5f5  180000.0       0      5       0     1       1     1
1.0

   r4h1  ...  SQBescolari  SQBage  SQBhogar_total  SQBedjefe  SQBhogar_
nin  \
0     0  ...          100    1849               1        100
```

```
  0
1     0  ...          144    4489              1        144
  0
2     0  ...          121    8464              1          0
  0
3     0  ...           81     289             16        121
  4
4     0  ...          121    1369             16        121
  4

   SQBovercrowding  SQBdependency  SQBmeaned  agesq  Target
0         1.000000            0.0      100.0   1849       4
1         1.000000           64.0      144.0   4489       4
2         0.250000           64.0      121.0   8464       4
3         1.777778            1.0      121.0    289       4
4         1.777778            1.0      121.0   1369       4

[5 rows x 143 columns]
```

Check and remove the null values

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Id                     0
v2a1                6860
hacdor                 0
rooms                  0
hacapo                 0
                    ...
SQBovercrowding        0
SQBdependency          0
SQBmeaned              5
agesq                  0
Target                 0
Length: 143, dtype: int64
```

In [5]:
```
null_columns=df.columns[df.isnull().any()]
```

```
df[null_columns].isnull().sum()
```

Out[5]:
```
v2a1          6860
v18q1         7342
rez_esc       7928
meaneduc         5
SQBmeaned        5
dtype: int64
```

In [6]:
```
print ('Percentage of null values in v2a1 : ', df['v2a1'].isnull().sum
()/df.shape[0]*100)
print ('Percentage of null values in v18q1 : ', df['v18q1'].isnull().su
m()/df.shape[0]*100)
print ('Percentage of null values in rez_esc : ', df['rez_esc'].isnull
().sum()/df.shape[0]*100)
print ('Percentage of null values in meaneduc : ', df['meaneduc'].isnul
l().sum()/df.shape[0]*100)
print ('Percentage of null values in SQBmeaned : ', df['SQBmeaned'].isn
ull().sum()/df.shape[0]*100)
```

```
Percentage of null values in v2a1 :  71.7798472323951
Percentage of null values in v18q1 :  76.82327090091033
Percentage of null values in rez_esc :  82.95490216595167
Percentage of null values in meaneduc :  0.05231767290990897
Percentage of null values in SQBmeaned :  0.05231767290990897
```

In [7]:
```
#Percentage of null values in v2a1, v18q1, rez_esc is more than 50%. S
o, these columns are dropped

df= df.drop(['v2a1','v18q1','rez_esc'],axis=1)
print(df.shape)
```

```
(9557, 140)
```

In [8]:
```
#Imputing the meaneduc & SQBmeaned coumns

imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(df[['meaneduc','SQBmeaned']])
df[['meaneduc','SQBmeaned']]=imp.transform(df[['meaneduc','SQBmeaned'
```

```
]])
df[['meaneduc','SQBmeaned']].isnull().sum()
```

Out[8]:
```
meaneduc     0
SQBmeaned    0
dtype: int64
```

From the train & test dataset, the output variable is Target column

In [9]:
```
df= df.drop(['Id'],axis=1)
df.describe(include='O')
```

Out[9]:

	idhogar	dependency	edjefe	edjefa
count	9557	9557	9557	9557
unique	2988	31	22	22
top	fd8a6d014	yes	no	no
freq	13	2192	3762	6230

In [10]:
```
df.dependency = df.dependency.replace(to_replace=['yes','no'],value=[0.5,0]).astype('float')
```

In [11]:
```
med_1=np.median(df.edjefe[df.edjefe.isin(['yes','no'])==False].astype('float'))
df.edjefe= df.edjefe.replace(to_replace=['yes','no'],value=[med_1,0]).astype('float')
```

In [12]:
```
med_2=np.median(df.edjefa[df.edjefa.isin(['yes','no'])==False].astype('float'))
df.edjefa= df.edjefa.replace(to_replace=['yes','no'],value=[med_2,0]).astype('float')
```

In [13]:
```
df.describe(include='O')
```

Out[13]:

	idhogar
count	9557
unique	2988
top	fd8a6d014
freq	13

In [14]:
```
print(df.idhogar.nunique())
```

```
2988
```

Finding biasness in the dataset

In [15]:
```
df.Target.value_counts()
import collections
print(df.shape)
collections.Counter(df['Target'])
```

```
(9557, 139)
```

Out[15]:
```
Counter({4: 5996, 2: 1597, 3: 1209, 1: 755})
```

Insights : It shows the biasness in the dataset.

Checking whether all members of the house have the same poverty level.

In [16]:
```
poverty_level=(df.groupby('idhogar')['Target'].nunique()>1).index
print(poverty_level)
```

```
Index(['001ff74ca', '003123ec2', '004616164', '004983866', '005905417',
       '006031de3', '006555fe2', '00693f597', '006b64543', '00941f1f4',
       ...
```

```
       'ff250fd6c', 'ff31b984b', 'ff38ddef1', 'ff6d16fd0', 'ff703eed4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be
1'],
      dtype='object', name='idhogar', length=2988)
```

Checking if there is a house without a family head.

In [17]:
```
no_head=(df.groupby('idhogar')['parentesco1'].sum()==0).index
display(no_head)
```

```
Index(['001ff74ca', '003123ec2', '004616164', '004983866', '005905417',
       '006031de3', '006555fe2', '00693f597', '006b64543', '00941f1f4',
       ...
       'ff250fd6c', 'ff31b984b', 'ff38ddef1', 'ff6d16fd0', 'ff703eed4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be
1'],
      dtype='object', name='idhogar', length=2988)
```

Set poverty level of the members and the head of the house same in a family.

In [18]:
```
target_mean=df.groupby('idhogar')['Target'].mean().astype('int64').rese
t_index().rename(columns={'Target':'Target_mean'})
df=df.merge(target_mean,how='left',on='idhogar')
df.Target=df.Target_mean
df.drop('Target_mean',axis=1,inplace=True)
df.head()
```

Out[18]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescolari	SQBage
0	0	3	0	1	1	0	0	1	1	0	...	100	1849
1	0	4	0	1	1	1	0	1	1	0	...	144	4489
2	0	8	0	1	1	0	0	0	0	0	...	121	8464
3	0	5	0	1	1	1	0	2	2	1	...	81	289

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescolari	SQBage
4	0	5	0	1	1	1	0	2	2	1	...	121	1369

5 rows × 139 columns

In [19]:
```
df.shape
```

Out[19]:
```
(9557, 139)
```

In [20]:
```
df= df.drop(['idhogar'],axis=1)
df.shape
```

Out[20]:
```
(9557, 138)
```

Assigning the value for x & y

In [21]:
```
x=df.drop(['Target'],axis=1)
print('shape of the x',x.shape)
y=df.Target
print('shape of the y',y.shape)
```

```
shape of the x (9557, 137)
shape of the y (9557,)
```

Deploying Random Forest Classifier.

In [22]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,rand
om_state=10)
rfc = RandomForestClassifier(criterion= 'gini',n_estimators=100)
rfc.fit(x_train,y_train)
pred=rfc.predict(x_test)
```

Check the accuracy using random forest with cross validation.

In [23]:
```
print('Accuracy score: ', accuracy_score(pred,y_test))
print()
print('Confusion matrix: ', confusion_matrix(pred,y_test))
print()
print('Classification report: ', classification_report(pred,y_test))
```

```
Accuracy score:  0.9325313807531381

Confusion matrix:  [[ 138    2    1    1]
 [   3  270    2    5]
 [   1    1  173    2]
 [  27   44   40 1202]]

Classification report:                precision    recall  f1-score   s
upport

           1       0.82      0.97      0.89       142
           2       0.85      0.96      0.90       280
           3       0.80      0.98      0.88       177
           4       0.99      0.92      0.95      1313

    accuracy                           0.93      1912
   macro avg       0.87      0.96      0.91      1912
weighted avg       0.94      0.93      0.93      1912
```

END OF PROJECT

In [ ]:
```
```