In [ ]:

```
'''
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:

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

In [323]:

```
#importing Amazon movie ratings dataset
import pandas as pd
income_train_df=pd.read_csv("E:/Education/PGP Simplilearn-Purdue/PGP in Data Science/Machine Learning/Machine-Learning--Projects-master/Projects/Projects for Submission/Project 2 - Income Qualification/Dataset for the project/train.csv")
income_test_df=pd.read_csv("E:/Education/PGP Simplilearn-Purdue/PGP in Data Science/Machine Learning/Machine-Learning--Projects-master/Projects/Projects for Submission/Project 2 - Income Qualification/Dataset for the project/test.csv")
```

In [324]:

```
#Exploring the datasets
income_train_df
```

Out[324]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nir
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	100	1849	1	100	(
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	144	4489	1	144	(
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	...	121	8464	1	0	(
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	...	81	289	16	121	4
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	...	121	1369	16	121	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
9552	ID_d45ae367d	80000.0	0	6	0	1	1	0	NaN	0	...	81	2116	25	81	'
9553	ID_c94744e07	80000.0	0	6	0	1	1	0	NaN	0	...	0	4	25	81	'
9554	ID_85fc658f8	80000.0	0	6	0	1	1	0	NaN	0	...	25	2500	25	81	'
9555	ID_ced540c61	80000.0	0	6	0	1	1	0	NaN	0	...	121	676	25	81	'
9556	ID_a38c64491	80000.0	0	6	0	1	1	0	NaN	0	...	64	441	25	81	'

9557 rows × 143 columns

In [325]:

```
income_train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 143 entries, Id to Target
dtypes: float64(8), int64(130), object(5)
memory usage: 10.4+ MB
```

In [326]:

```
income_test_df
```

Out[326]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	age	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBho
0	ID_2f6873615	NaN	0	5	0	1	1	0	NaN	1	...	4	0	16	9	0	
1	ID_1c78846d2	NaN	0	5	0	1	1	0	NaN	1	...	41	256	1681	9	0	
2	ID_e5442cf6a	NaN	0	5	0	1	1	0	NaN	1	...	41	289	1681	9	0	
3	ID_a8db26a79	NaN	0	14	0	1	1	1	1.0	0	...	59	256	3481	1	256	
4	ID_a62966799	175000.0	0	4	0	1	1	1	1.0	0	...	18	121	324	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
23851	ID_a065a7cad	NaN	1	2	1	1	1	0	NaN	0	...	10	9	100	36	25	
23852	ID_1a7c6953b	NaN	0	3	0	1	1	0	NaN	0	...	54	36	2916	16	36	
23853	ID_07dbb4be2	NaN	0	3	0	1	1	0	NaN	0	...	12	16	144	16	36	
23854	ID_34d2ed046	NaN	0	3	0	1	1	0	NaN	0	...	12	25	144	16	36	
23855	ID_34754556f	NaN	0	3	0	1	1	0	NaN	0	...	51	36	2601	16	36	

23856 rows × 142 columns

In [327]:

```
income_test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23856 entries, 0 to 23855
Columns: 142 entries, Id to agesq
dtypes: float64(8), int64(129), object(5)
memory usage: 25.8+ MB
```

In [328]:

```
#Analysis Tasks
#Identify the output variable.
income_train_df['Target'].value_counts()
#Observations:
#Output variable is 'Target'
#Target is an ordinal variable with 4 levels(1,2,3,4) indicating different levels of poverty as mentioned below
#1 = extreme poverty
#2 = moderate poverty
#3 = vulnerable households
#4 = non vulnerable households
```

Out[328]:

```
4    5996
2    1597
3    1209
1     755
Name: Target, dtype: int64
```

In [329]:

```
#Understand the type of data.
income_train_df.info()
#Observations:
#Training data has 142 feature variables of which 129 are integer type,8 are float and 5 are of string type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 143 entries, Id to Target
dtypes: float64(8), int64(130), object(5)
memory usage: 10.4+ MB
```

In [330]:

```
#Check if there are any biases in your dataset.
income_train_df['Target'].value_counts()
#Observations:
#Target variable has 5996(~62.57%) non vulnerable cases out of 9557 data points
#Dataset is biased towards non vulnerable cases
```

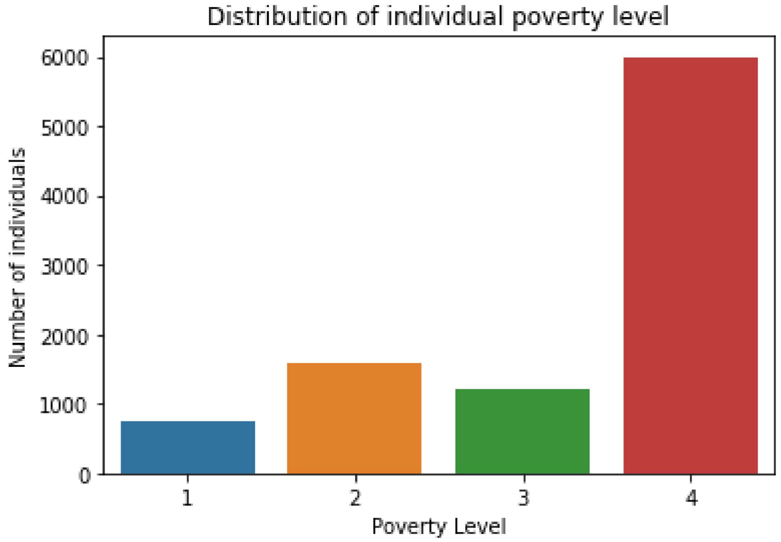
Out[330]:

```
4    5996
2    1597
3    1209
1     755
Name: Target, dtype: int64
```

In [331]:

```
#Target variable visualization
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(income_train_df['Target'])
plt.title('Distribution of individual poverty level')
plt.xlabel('Poverty Level')
plt.ylabel('Number of individuals')
plt.show()
#Observation:
#It can be clearly seen from the visualization that the dataset is biased towards non vulnerable house holds
```



In [332]:

```
#Check whether all members of the house have the same poverty level.
households_df=income_train_df['idhogar'].value_counts()
print(households_df)
print(households_df[households_df>1])
#Observations:
#There are a total of 2988 households and the largest household has a total of 13 people
#There are 2590 households which has atleast 2 people=> rest(398) are single person households
```

```
fd8a6d014    13
0c7436de6    12
ae6cf0558    12
6b35cdcf0    11
b7a0b59d7    11
             ..
a9b2a46ba     1
1e84a2ac8     1
9eb450831     1
5492ff7f2     1
dd7adf3ea     1
Name: idhogar, Length: 2988, dtype: int64
fd8a6d014    13
0c7436de6    12
ae6cf0558    12
6b35cdcf0    11
b7a0b59d7    11
             ..
894a54bdb     2
a874b7ce7     2
375d5dff6     2
1587d9070     2
17e26c0f0     2
Name: idhogar, Length: 2590, dtype: int64
```

In [333]:

```
#Check whether all members of the house have the same poverty level.
def is_unique(h):
    t=h.to_numpy()
    return (t[0]==t).all()

h_list=[]
for c in households_df[households_df>1].index:
    if(is_unique(income_train_df[income_train_df['idhogar']==c]['Target'])==False):
        h_list.append(c)

len(h_list)
#Observations:
#All the members of the household does not have the same poverty level for 85 households
```

Out[333]:

```
85
```

In [334]:

```
#Check if there is a house without a family head.
#Households with a family head
print("Households with family head:",len(income_train_df.idhogar[income_train_df['parentesco1']==1].unique()))

#Households without family head
x=[x for x in list(households_df.index) if x not in list(income_train_df.idhogar[income_train_df['parentesco1']==1])]
print("Households without family head:",len(x))
print(x)
#Observation:
#There are 15 households without a family head
```

```
Households with family head: 2973
Households without family head: 15
['6b1b2405f', 'c0c8a5013', 'f2bfa75c4', '03c6bdf85', 'bfd5067c2', '09b195e7a', '1367ab31d', '61c10e099', 'a0812ef17', 'b1f4d89d
7', '896fe6d3e', '374ca5a19', '1bc617b23', 'd363d9183', 'ad687ad89']
```

In [335]:

```
#Check whether all members of the house does not have the same poverty level and without family head
y=[x for x in x if x in h_list]
len(y)
#Observation:
#There are 0 households where all members of the house does not have the same poverty level and without family head
```

Out[335]:

```
0
```

In [336]:

```
#Set poverty level of the members and the head of the house within a family.
#Setting the poverty level of members same as head of the family
for h in h_list:
    income_train_df.loc[income_train_df['idhogar']==h,'Target']=int(income_train_df.Target[(income_train_df['idhogar']==h) & (income_train_df['parentesco1']==1)])
```

In [337]:

```
#Check whether all members of the house have the same poverty level.
h_list1=[]
for c in households_df[households_df>1].index:
    if(is_unique(income_train_df[income_train_df['idhogar']==c]['Target'])==False):
        h_list1.append(c)

len(h_list1)
#Observations:
#All the members of the household does not have the same poverty level has been set
```

Out[337]:

```
0
```

In [338]:

```
#Count how many null values are existing in columns.
income_train_df.isna().sum()[income_train_df.isna().sum()>0]
#Observations:
#There are 5 columns with null values
#3(Monthly rent Payment, number of tablets household owns and Years behind in school) columns have high proportion of nulls
#2 columns(average years of education for adults (18+) and square of the mean years of education of adults (>=18) in the household)-
#have negligible null values

#Inferences:
#Monthly rent payment could be null as the household might own a house and don't have to pay rent
#Number of tablets owned by a household could be null as they don't own any tablets
#Years behing in school could be null as these individuals were never behind in school
#Average years of education could be null as they might be uneducated
#These values can be imputed by '0'
```

Out[338]:

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

In [339]:

```
#Imputing null values with 0's for first 3 columns and deleting null records for last 2 columns
income_train_df['v2a1'].fillna(value=0,inplace=True)
income_train_df['v18q1'].fillna(value=0,inplace=True)
income_train_df['rez_esc'].fillna(value=0,inplace=True)
income_train_df['meaneduc'].fillna(value=0,inplace=True)
income_train_df['SQBmeaned'].fillna(value=0,inplace=True)
```

In [340]:

```
#Count how many null values are existing in columns after imputing
income_train_df.isna().sum()[income_train_df.isna().sum()>0]
```

Out[340]:

```
Series([], dtype: int64)
```

In [341]:

```
#Same exercise should be repeated for test dataset
income_test_df.isna().sum()[income_test_df.isna().sum()>0]
```

Out[341]:

```
v2a1         17403
v18q1        18126
rez_esc      19653
meaneduc        31
SQBmeaned       31
dtype: int64
```

In [342]:

```
#Imputing null values with 0's for first 3 columns and deleting null records for last 2 columns
income_test_df['v2a1'].fillna(value=0,inplace=True)
income_test_df['v18q1'].fillna(value=0,inplace=True)
income_test_df['rez_esc'].fillna(value=0,inplace=True)
income_test_df['meaneduc'].fillna(value=0,inplace=True)
income_test_df['SQBmeaned'].fillna(value=0,inplace=True)
```

In [343]:

```
#Count how many null values are existing in columns after imputation
income_test_df.isna().sum()[income_test_df.isna().sum()>0]
```

Out[343]:

```
Series([], dtype: int64)
```

In [344]:

```
#Remove null value rows of the target variable.
income_train_df['Target'].isna().sum()
#No null values are present in target column
```

Out[344]:

```
0
```

In [345]:

```
#Checking for columns with string data type
income_train_df.columns[income_train_df.dtypes=='object']
#Observations:
#ID and Idhogar represents individuals and households
#Rest of the columns should be numeric and will be replaced by 1 for yes and 0 for no as per the data dictionary
```

Out[345]:

```
Index(['Id', 'idhogar', 'dependency', 'edjefe', 'edjefa'], dtype='object')
```

In [346]:

```
#Replacing strings with numeric values
income_train_df.loc[income_train_df['dependency']=='yes','dependency']=1
income_train_df.loc[income_train_df['dependency']=='no','dependency']=0

income_train_df.loc[income_train_df['edjefe']=='yes','edjefe']=1
income_train_df.loc[income_train_df['edjefe']=='no','edjefe']=0

income_train_df.loc[income_train_df['edjefa']=='yes','edjefa']=1
income_train_df.loc[income_train_df['edjefa']=='no','edjefa']=0
```

In [356]:

```
#Changing object datatype to numeric
income_train_df['dependency']=pd.to_numeric(income_train_df['dependency'])
income_train_df['edjefe']=pd.to_numeric(income_train_df['edjefe'])
income_train_df['edjefa']=pd.to_numeric(income_train_df['edjefa'])
```

In [359]:

```
income_train_df.columns[income_train_df.dtypes=='object']
```

Out[359]:

```
Index(['Id', 'idhogar'], dtype='object')
```

In [360]:

```
#Repeating same steps for test dataset
income_test_df.columns[income_test_df.dtypes=='object']
```

Out[360]:

```
Index(['Id', 'idhogar', 'dependency', 'edjefe', 'edjefa'], dtype='object')
```

In [361]:

```
#Replacing strings with numeric values
income_test_df.loc[income_test_df['dependency']=='yes','dependency']=1
income_test_df.loc[income_test_df['dependency']=='no','dependency']=0

income_test_df.loc[income_test_df['edjefe']=='yes','edjefe']=1
income_test_df.loc[income_test_df['edjefe']=='no','edjefe']=0

income_test_df.loc[income_test_df['edjefa']=='yes','edjefa']=1
income_test_df.loc[income_test_df['edjefa']=='no','edjefa']=0
```

In [362]:

```
#Changing object datatype to numeric
income_test_df['dependency']=pd.to_numeric(income_test_df['dependency'])
income_test_df['edjefe']=pd.to_numeric(income_test_df['edjefe'])
income_test_df['edjefa']=pd.to_numeric(income_test_df['edjefa'])
```

In [363]:

```
income_test_df.columns[income_test_df.dtypes=='object']
```

Out[363]:

```
Index(['Id', 'idhogar'], dtype='object')
```

In [367]:

```
#Creating train and test sets
x_train=income_train_df.drop(columns=['Id','idhogar','Target'])
y_train=pd.DataFrame(income_train_df['Target'])
x_test=income_test_df.drop(columns=['Id','idhogar'])
#Check the accuracy using random forest with cross validation.
```

In [388]:

```
import warnings
warnings.filterwarnings("ignore")

#Building random forest classifier model
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=20,random_state=12)
income_rfc_model=rfc.fit(x_train,y_train)
y_pred=income_rfc_model.predict(x_test)
```

In [389]:

```
#Check the accuracy using random forest with cross validation.
from sklearn.model_selection import cross_val_score
cross_val_score(rfc,x_train,y_train,cv=20,scoring='accuracy').mean()
#Observation:
#Mean accuracy score:64.45%
```

Out[389]:

```
0.644543345350561
```