In [316]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

In [317]:
```
data = pd.read_csv(r"C:\Users\GUNDEEP\Desktop\Dataset for the project (1)\
Dataset for the project\train.csv")
data.head(5)
```

Out[317]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBescolar
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	100
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	144
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	...	121
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	...	81
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	...	121

5 rows × 143 columns

In [318]:
```
data.describe()
```

Out[318]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q
count	2.697000e+03	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000
mean	1.652316e+05	0.038087	4.955530	0.023648	0.994768	0.957623	0.231767
std	1.504571e+05	0.191417	1.468381	0.151957	0.072145	0.201459	0.421983
min	0.000000e+00	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	8.000000e+04	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000
50%	1.300000e+05	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000
75%	2.000000e+05	0.000000	6.000000	0.000000	1.000000	1.000000	0.000000
max	2.353477e+06	1.000000	11.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 138 columns

In [319]:
```
data.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 [320]:
```
# dataFr = pd.DataFrame(data)
```

In [321]:
```
newdata  = data.loc[:,['idhogar','Id','Target']]
# print(newdata)
```

In [322]:
```
# creation of csv to check for wether all the family members have same/dif
ferent level of poverty
newdata.to_csv(r'Desktop/checkkro2.csv')
```

In [323]:
```
# the target value is not same for all the members and thus needs to be ma
de same
#making the target variables same for all the family members..


#finding the household in which all the family members have the same pover
ty levels
household_unique = data.groupby('idhogar')['Target'].nunique() == 1
#finding the household in which all the family members do not have the sam
e poverty levels
household_not_unique = household_unique[household_unique != True ]

#correcting the labels
for family in household_not_unique.index:
    #select the actual label
    actual_label = int(data[(data['idhogar'] == family) & (data['parentesc
o1'] == 1)]['Target'])
    #correct the label
    data.loc[data['idhogar'] == family, 'Target'] = actual_label
```

In [324]:
```
# to show that there exists households without head
data.loc[:,['parentesco1']].head(5)
```

Out[324]:

	parentesco1
0	1
1	1
2	1
3	0
4	0

we know that 'parentesco1' column signifies the household head and has value equal to 0 for no head and 1 if there exists a head of the household.

Since in the above output we have a lot of 0 values for parentesco1 column, hence we can certainly say that yes there exist houses without the household head.

In [325]:
```
# to check the null values of all the columns
data.isnull().sum(axis=0).sort_values(ascending = False).head(9)
```

Out[325]:
```
rez_esc            7928
v18q1              7342
```

```
v2a1               6860
meaneduc              5
SQBmeaned             5
techozinc             0
techoentrepiso        0
techocane             0
techootro             0
dtype: int64
```

In [326]:
```
# databn.to_csv(r'Desktop/nullcal.csv')
```

In [327]:
```
male_education = data.loc[((data.edjefe == 'no') | (data.edjefe == 'yes'))
 & (data.male == 1 ) & (data.parentesco1 == 1) ,
          [ 'edjefe','escolari']]



male_education[(male_education.escolari == 1)]['edjefe'].value_counts()

data.loc[(data.edjefe  == 'yes') , 'edjefe'] = 1

data.loc[(data.edjefe  == 'no'), 'edjefe'] = 0

data.edjefe = data.edjefe.astype(float)
```

In [328]:
```
data.loc[(data.edjefa  == 'yes') , 'edjefa'] = 1

data.loc[(data.edjefa  == 'no'), 'edjefa'] = 0

data.edjefa = data.edjefa.astype(float)
```

In [329]:
```
#dependency

data.loc[(data.dependency  == 'yes') , 'dependency'] = 1

data.loc[(data.dependency  == 'no'), 'dependency'] = 0

data.dependency = data.dependency.astype(float)
# data.Target = data.Target.astype(float)
```

In [330]:
```
# since we dont need idhogar for the prediction of level of poverty level
hence we just drop it before training.
data = data.drop("idhogar", axis = 1)
```

In [331]:
```
# to see that idhogar is removed and now only 142 columns exist
data.info()
# data.loc[:,['idhogar']]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 142 entries, Id to Target
dtypes: float64(11), int64(130), object(1)
memory usage: 10.4+ MB
```

In [332]:
```
# dropping the columns that contain null values
data = data.dropna(axis='columns')
```

In [333]:
```
# to ensure that the columns are dropped.//only 137 columns left more
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 137 entries, Id to Target
dtypes: float64(6), int64(130), object(1)
memory usage: 10.0+ MB
```

In [334]:
```
# just for self checking
checkdata = data.iloc[:,136]
# checkdata.to_csv(r'Desktop/checkkro3.csv')
```

In [335]:
```
arr = np.array(data)
```

In [336]:
```
X = arr[:,1:136]
y= arr[:,-1]
```

In [337]:
```
print(X)
print(y)
```

```
[[0 3 0 ... 1.0 0.0 1849]
 [0 4 0 ... 1.0 64.0 4489]
 [0 8 0 ... 0.25 64.0 8464]
 ...
 [0 6 0 ... 1.5625 0.0625 2500]
 [0 6 0 ... 1.5625 0.0625 676]
 [0 6 0 ... 1.5625 0.0625 441]]
[4 4 4 ... 2 2 2]
```

In [338]:
```
# using cross validation (train_test_split_) with 20% test and 80% trainin
g data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
andom_state = 0)
y_train=y_train.astype('int')
y_test=y_test.astype('int')
```

In [351]:
```
# random forest with maximum depth as 2
clf = RandomForestClassifier(max_depth=2, random_state=0)
```

In [352]:
```
clf.fit(X_train,y_train)
```

```
C:\Users\GUNDEEP\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:24
6: FutureWarning: The default value of n_estimators will change from 10 in
 version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[352]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini'
,
                       max_depth=2, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
```

```
                       min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                       oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [353]:
```
# training accuracy
clf.score(X_train,y_train)
```

Out[353]: 0.6334859385219097

In [354]:
```
# testing accuracy
clf.score(X_test,y_test)
```

Out[354]: 0.6239539748953975

In [ ]:
```

```