In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sea
```

In [2]:

```
pmt = pd.read_csv('./Dataset/train.csv')
pmt.sample(10)
```

Out[2]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQ
7583	ID_9ab4f44e8	NaN	0	5	0	1	1	0	NaN	0	...	
4476	ID_0889dfdb6	NaN	0	6	0	1	1	0	NaN	0	...	
8461	ID_c3e0cca84	NaN	0	4	0	1	1	0	NaN	0	...	
4661	ID_b18bcff05	250000.0	0	4	0	1	1	0	NaN	0	...	
4962	ID_e7eba7cbf	NaN	0	4	0	1	1	1	1.0	0	...	
6384	ID_148241b63	NaN	0	4	0	1	1	0	NaN	0	...	
2706	ID_0faefcf6b	NaN	0	8	0	1	1	1	1.0	0	...	
2525	ID_3161e947d	NaN	0	6	0	1	1	0	NaN	1	...	
7557	ID_4e8f627c2	NaN	0	5	0	1	1	0	NaN	0	...	
2867	ID_de53e1c61	150000.0	0	5	0	1	1	0	NaN	0	...	

10 rows × 143 columns

In [3]:

```
pmt.columns
```

Out[3]:

```
Index(['Id', 'v2a1', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q',
       'v18q1', 'r4h1',
       ...
       'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe', 'SQBhogar_ni
n',
       'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq', 'Target'],
      dtype='object', length=143)
```

Count how many null values are existing in columns.

In [4]:

```
null = pmt.isna().sum()
null
```

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.sum()
```

Out[5]:

```
22140
```

Remove null value rows of the target variable.

In [6]:

```
target_null_count = pmt['Target'].isnull().sum()
print(f'target variable has {target_null_count} NULL values')
```

```
target variable has 0 NULL values
```

target variable has 0 NULL values

22140 null values are existing for all columns

In [7]:

```
pmt.shape
```

Out[7]:

```
(9557, 143)
```

In [8]:

```
pmt.describe()
```

Out[8]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	
count	2.697000e+03	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	9557.00
mean	1.652316e+05	0.038087	4.955530	0.023648	0.994768	0.957623	0.23
std	1.504571e+05	0.191417	1.468381	0.151957	0.072145	0.201459	0.42
min	0.000000e+00	0.000000	1.000000	0.000000	0.000000	0.000000	0.00
25%	8.000000e+04	0.000000	4.000000	0.000000	1.000000	1.000000	0.00
50%	1.300000e+05	0.000000	5.000000	0.000000	1.000000	1.000000	0.00
75%	2.000000e+05	0.000000	6.000000	0.000000	1.000000	1.000000	0.00
max	2.353477e+06	1.000000	11.000000	1.000000	1.000000	1.000000	1.00

8 rows × 138 columns

In [9]:

```
pmt_test = pd.read_csv('./Dataset/test.csv')
pmt_test.sample(10)
```

Out[9]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	ag
7873	ID_ab8894239	NaN	0	5	0	1	1	0	NaN	0	...	2
19693	ID_0d79180e4	NaN	0	4	0	1	1	1	1.0	0	...	2
1174	ID_c2f5e220b	NaN	0	6	0	1	1	1	1.0	0	...	3
718	ID_1399fc0b4	NaN	0	7	0	1	1	0	NaN	0	...	3
9872	ID_caea5e62e	NaN	0	5	0	1	1	0	NaN	0	...	8
6892	ID_c65e98643	NaN	0	4	0	1	0	0	NaN	0	...	1
20540	ID_26eca8a92	NaN	0	4	0	1	1	0	NaN	0	...	6
9739	ID_1383e3871	NaN	1	4	0	1	1	0	NaN	0	...	2
14584	ID_6d6274c9f	50000.0	0	3	0	1	1	0	NaN	0	...	1
5823	ID_dadcddbce	NaN	0	8	0	1	1	0	NaN	0	...	2

10 rows × 142 columns

In [10]:

```
pmt_test.columns
```

Out[10]:

```
Index(['Id', 'v2a1', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q',
       'v18q1', 'r4h1',
       ...
       'age', 'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
       'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned',
       'agesq'],
      dtype='object', length=142)
```

Identify the output variable.

The column 'TARGET' in the training dataset is the output variable

Understand the type of data.

In [11]:

```
pmt.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
```

There are 130 integer columns(including Boolean and TARGET column), 8 float (numeric) columns, and 5 object columns (String)

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

In [12]:

```
equal = pmt.groupby('idhogar')['Target'].apply(lambda x: x.nunique() == 1)

not_equal = equal[equal != True]

not_equal_count = len(not_equal)

if not_equal_count ==0:
    print("all members of all house have the same poverty level")
else:
    print(f"all members of the {not_equal_count} house do not have the same poverty level")
```

```
all members of the 85 house do not have the same poverty level
```

all members of the 85 house do not have the same poverty level

Check if there is a house without a family head.

In [13]:

```
leader = pmt.groupby('idhogar')['parentesco1'].sum()

# Find households without a head
no_head = pmt.loc[pmt['idhogar'].isin(leader[leader == 0].index), :]

no_head_count = no_head['idhogar'].nunique()

print(f'There are {(no_head_count)} households without a head.')
```

```
There are 15 households without a head.
```

There are 15 households without a head.

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

In [14]:

```
for household in not_equal.index:

    target1 = int(pmt[(pmt['idhogar'] == household) & (pmt['parentesco1'] == 1.0)]['Target'

    # Set correct label for all members in the household
    pmt.loc[pmt['idhogar'] == household, 'Target'] = target1


# Check count again

equal = pmt.groupby('idhogar')['Target'].apply(lambda x: x.nunique() == 1)

not_equal = equal[equal != True]

not_equal_count = len(not_equal)

if not_equal_count ==0:
    print("all members of all house have the same poverty level")
else:
    print(f"all members of the {not_equal_count} house do not have the same poverty level")
```

```
all members of all house have the same poverty level
```

In [15]:

```
pmt.select_dtypes(include=['object'])
```

Out[15]:

	Id	idhogar	dependency	edjefe	edjefa
0	ID_279628684	21eb7fcc1	no	10	no
1	ID_f29eb3ddd	0e5d7a658	8	12	no
2	ID_68de51c94	2c7317ea8	8	no	11
3	ID_d671db89c	2b58d945f	yes	11	no
4	ID_d56d6f5f5	2b58d945f	yes	11	no
...	...	...	...	...	...
9552	ID_d45ae367d	d6c086aa3	.25	9	no
9553	ID_c94744e07	d6c086aa3	.25	9	no
9554	ID_85fc658f8	d6c086aa3	.25	9	no
9555	ID_ced540c61	d6c086aa3	.25	9	no
9556	ID_a38c64491	d6c086aa3	.25	9	no

9557 rows × 5 columns

In [16]:

```
pmt_copy = pmt.drop(columns=['Id','idhogar','Target'],axis=1)
```

In [17]:

```
pmt_test = pmt_test.drop(columns=['Id','idhogar'],axis=1)
```

In [18]:

```
pmt_copy['dependency'] = pd.to_numeric(pmt_copy['dependency'],errors='coerce')
pmt_copy['edjefa'] = pd.to_numeric(pmt_copy['edjefa'],errors='coerce')
pmt_copy['edjefe'] = pd.to_numeric(pmt_copy['edjefe'],errors='coerce')
```

In [19]:

```
pmt_test['dependency'] = pd.to_numeric(pmt_test['dependency'],errors='coerce')
pmt_test['edjefa'] = pd.to_numeric(pmt_test['edjefa'],errors='coerce')
pmt_test['edjefe'] = pd.to_numeric(pmt_test['edjefe'],errors='coerce')
```

In [20]:

```
X_train = pmt_copy
y_train = pmt['Target']
X_test = pmt_test
```

In [21]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, make_scorer
from sklearn.model_selection import cross_val_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
```

In [22]:

```
pipeline = Pipeline([('imputer', SimpleImputer(missing_values=np.nan, strategy="most_freque
                      ('scaler', MinMaxScaler())])
scorer = make_scorer(f1_score, greater_is_better=True, average = 'macro')
```

In [23]:

```
train_set = pipeline.fit_transform(X_train)
test_set = pipeline.transform(X_test)
```

In [24]:

```
model = RandomForestClassifier(n_estimators=100, random_state=10,
                               n_jobs = -1)
```

In [25]:

```
cv_score = cross_val_score(model, train_set, y_train, cv = 10, scoring = scorer)
```

In [26]:

```
print(f'10 Fold Cross Validation F1 Score = {round(cv_score.mean(), 4)} with std = {round(c
```

```
10 Fold Cross Validation F1 Score = 0.3629 with std = 0.0436
```

In [ ]:

```

```