Income Qualification

February 8, 2022

1 Name: Sunil Pradhan

1.1 Project: 2

1.2 Project Name: Income Qualification

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
#reading the train and test dataset
df_train=pd.read_csv("train.csv", index_col=0)
df_test=pd.read_csv("test.csv", index_col=0)
```

[3]:
```
#displaying the train dataset
df_train.head()
```

[3]:

Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	\
ID_279628684	190000.0	0	3	0	1	1	0	NaN	
ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	
ID_68de51c94	NaN	0	8	0	1	1	0	NaN	
ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	
ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	

Id	r4h1	r4h2	…	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	\
ID_279628684	0	1	…	100	1849	1	100	
ID_f29eb3ddd	0	1	…	144	4489	1	144	
ID_68de51c94	0	0	…	121	8464	1	0	
ID_d671db89c	0	2	…	81	289	16	121	
ID_d56d6f5f5	0	2	…	121	1369	16	121	

Id	SQBhogar_nin	SQBovercrowding	SQBdependency	SQBmeaned	agesq	\

```
ID_279628684             0         1.000000            0.0      100.0   1849
ID_f29eb3ddd             0         1.000000           64.0      144.0   4489
ID_68de51c94             0         0.250000           64.0      121.0   8464
ID_d671db89c             4         1.777778            1.0      121.0    289
ID_d56d6f5f5             4         1.777778            1.0      121.0   1369

              Target
Id
ID_279628684       4
ID_f29eb3ddd       4
ID_68de51c94       4
ID_d671db89c       4
ID_d56d6f5f5       4

[5 rows x 142 columns]
```

[4]:
```
#dsiplaying the test dataset
df_test.head()
```

[4]:

Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	…	r4h1	r4h2	…	age	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nin	SQBovercrowding	SQBdependency	SQBmeaned	agesq
ID_2f6873615	NaN	0	5	0	1	1	0	NaN	…	1	1	…	4	0	16	9	0	1	2.25	0.25	272.25	16
ID_1c78846d2	NaN	0	5	0	1	1	0	NaN	…	1	1	…	41	256	1681	9	0	1	2.25	0.25	272.25	1681
ID_e5442cf6a	NaN	0	5	0	1	1	0	NaN	…	1	1	…	41	289	1681	9	0	1	2.25	0.25		
ID_a8db26a79	NaN	0	14	0	1	1	1	1.0	…	0	1	…	59	256	3481	1	256	0	1.00	0.00		
ID_a62966799	175000.0	0	4	0	1	1	1	1.0	…	0	0	…	18	121	324	1	0	1	0.25	64.00		

```
ID_e5442cf6a     272.25   1681
ID_a8db26a79     256.00   3481
ID_a62966799        NaN    324

[5 rows x 141 columns]
```

[5]:
```
#checking info of trin dataset
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9557 entries, ID_279628684 to ID_a38c64491
Columns: 142 entries, v2a1 to Target
dtypes: float64(8), int64(130), object(4)
memory usage: 10.4+ MB
```

[6]:
```
#checking info of test dataset
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 23856 entries, ID_2f6873615 to ID_34754556f
Columns: 141 entries, v2a1 to agesq
dtypes: float64(8), int64(129), object(4)
memory usage: 25.8+ MB
```

[7]:
```
#shape of train and test dataset
df_train.shape, df_test.shape
```

[7]:
```
((9557, 142), (23856, 141))
```

[8]:
```
#checking the categorical columns in train
df_train.describe(include='object')
```

[8]:

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

[9]:
```
#checking the categorical columns in test
df_test.describe(include='object')
```

[9]:

	idhogar	dependency	edjefe	edjefa
count	23856	23856	23856	23856
unique	7352	35	22	22
top	8e9159699	yes	no	no
freq	13	5388	9056	15845

[10]:
```
#preprocessing for categorical columns
```

[11]:
```
#checking unique values for 'idhogar' column in train and test dataset
df_train['idhogar'].unique(), df_test['idhogar'].unique()
```

[11]:
```
(array(['21eb7fcc1', '0e5d7a658', '2c7317ea8', …, 'a8eeafc29',
        '212db6f6c', 'd6c086aa3'], dtype=object),
 array(['72958b30c', '5b598fbc9', '1e2fc704e', …, '2edb6f51e',
        '3aa78c56b', 'd237404b6'], dtype=object))
```

[12]:
```
#checking unique values for 'dependency' column in train and test dataset
df_train['dependency'].unique(), df_test['dependency'].unique()
```

[12]:
```
(array(['no', '8', 'yes', '3', '.5', '.25', '2', '.66666669', '.33333334',
        '1.5', '.40000001', '.75', '1.25', '.2', '2.5', '1.2', '4',
        '1.3333334', '2.25', '.22222222', '5', '.83333331', '.80000001',
        '6', '3.5', '1.6666666', '.2857143', '1.75', '.71428573',
        '.16666667', '.60000002'], dtype=object),
 array(['.5', 'no', '8', 'yes', '.25', '2', '.33333334', '.375',
        '.60000002', '1.5', '.2', '.75', '.66666669', '3', '.14285715',
        '.40000001', '.80000001', '1.6666666', '.2857143', '1.25', '2.5',
        '5', '.85714287', '1.3333334', '.16666667', '4', '.125',
        '.83333331', '2.3333333', '7', '1.2', '3.5', '2.25', '3.3333333',
        '6'], dtype=object))
```

[13]:
```
#checking unique values for 'edjefe' column in train and test dataset
df_train['edjefe'].unique(), df_test['edjefe'].unique()
```

[13]:
```
(array(['10', '12', 'no', '11', '9', '15', '4', '6', '8', '17', '7', '16',
        '14', '5', '21', '2', '19', 'yes', '3', '18', '13', '20'],
       dtype=object),
 array(['no', '16', '10', '6', '11', '8', '13', '14', '5', '3', '9', '17',
        '15', '7', '21', '4', '12', '2', '20', 'yes', '19', '18'],
       dtype=object))
```

[14]:
```
#checking unique values for 'edjefa' column in train and test dataset
df_train['edjefa'].unique(), df_test['edjefa'].unique()
```

[14]:
```
(array(['no', '11', '4', '10', '9', '15', '7', '14', '13', '8', '17', '6',
        '5', '3', '16', '19', 'yes', '21', '12', '2', '20', '18'],
       dtype=object),
 array(['17', 'no', '11', '14', '10', '15', '9', '6', '8', '3', '2', '5',
        '16', '12', 'yes', '7', '13', '21', '4', '19', '18', '20'],
       dtype=object))
```

[15]:
```
#replacing yes and no in deependency columns with mode and 0
```

```
dependency_mode=df_train['dependency'][df_train['dependency'].
 ↪isin(['no','yes'])==False].astype('float').mode()
df_train['dependency']=df_train['dependency'].replace('no',0).
 ↪replace('yes',dependency_mode[0])
df_test['dependency']=df_test['dependency'].replace('no',0).
 ↪replace('yes',dependency_mode[0])
```

[16]:
```
df_train['dependency'].unique(), df_test['dependency'].unique()
```

[16]:
```
(array([0, '8', 0.5, '3', '.5', '.25', '2', '.66666669', '.33333334',
        '1.5', '.40000001', '.75', '1.25', '.2', '2.5', '1.2', '4',
        '1.3333334', '2.25', '.22222222', '5', '.83333331', '.80000001',
        '6', '3.5', '1.6666666', '.2857143', '1.75', '.71428573',
        '.16666667', '.60000002'], dtype=object),
 array(['.5', 0, '8', 0.5, '.25', '2', '.33333334', '.375', '.60000002',
        '1.5', '.2', '.75', '.66666669', '3', '.14285715', '.40000001',
        '.80000001', '1.6666666', '.2857143', '1.25', '2.5', '5',
        '.85714287', '1.3333334', '.16666667', '4', '.125', '.83333331',
        '2.3333333', '7', '1.2', '3.5', '2.25', '3.3333333', '6'],
       dtype=object))
```

[17]:
```
#replacing  yes and no in edjefe column with median and 0
edjefe_med=df_train['edjefe'][df_train['edjefe'].isin(['no','yes'])==False].
 ↪astype('float').median()
df_train['edjefe']=df_train['edjefe'].replace('no',0).replace('yes',edjefe_med)
df_test['edjefe']=df_test['edjefe'].replace('no',0).replace('yes',edjefe_med)
```

[18]:
```
df_train['edjefe'].unique(), df_test['edjefe'].unique()
```

[18]:
```
(array(['10', '12', 0, '11', '9', '15', '4', '6', '8', '17', '7', '16',
        '14', '5', '21', '2', '19', 7.0, '3', '18', '13', '20'],
       dtype=object),
 array([0, '16', '10', '6', '11', '8', '13', '14', '5', '3', '9', '17',
        '15', '7', '21', '4', '12', '2', '20', 7.0, '19', '18'],
       dtype=object))
```

[19]:
```
#replacing  yes and no in edjefa column with median and 0
edjefa_med=df_train['edjefa'][df_train['edjefa'].isin(['no','yes'])==False].
 ↪astype('float').median()
df_train['edjefa']=df_train['edjefa'].replace('no',0).replace('yes',edjefa_med)
df_test['edjefa']=df_test['edjefa'].replace('no',0).replace('yes',edjefa_med)
```

[20]:
```
df_train['edjefa'].unique(), df_test['edjefa'].unique()
```

[20]:
```
(array([0, '11', '4', '10', '9', '15', '7', '14', '13', '8', '17', '6',
        '5', '3', '16', '19', 7.0, '21', '12', '2', '20', '18'],
       dtype=object),
```

```
array(['17', 0, '11', '14', '10', '15', '9', '6', '8', '3', '2', '5',
       '16', '12', 7.0, '7', '13', '21', '4', '19', '18', '20'],
      dtype=object))
```

[21]:
```
#checking for missing values for train dataset
np.sum(df_train.isnull().sum())
```

[21]:
```
22140
```

[22]:
```
#null columns in the train dataset
null_columns=df_train.columns[df_train.isnull().any()]
df_train[null_columns].isnull().sum()
```

[22]:
```
v2a1          6860
v18q1         7342
rez_esc       7928
meaneduc         5
SQBmeaned        5
dtype: int64
```

[23]:
```
#checking the percentage of null values in the above columns of train
v2a1_null1=(df_train['v2a1'].isnull().sum()/df_train.shape[0])*100
v18q1_null1=(df_train['v18q1'].isnull().sum()/df_train.shape[0])*100
rez_esc_null1=(df_train['rez_esc'].isnull().sum()/df_train.shape[0])*100

print("Null Value Percentage in v2a1:",v2a1_null1)
print("Null Value Percentage in v18q1:",v18q1_null1)
print("Null Value Percentage in rez_esc:",rez_esc_null1)
```

```
Null Value Percentage in v2a1: 71.7798472323951
Null Value Percentage in v18q1: 76.82327090091033
Null Value Percentage in rez_esc: 82.95490216595167
```

As the above 3 columns have null value more than 50%, so we have to drop that column

[24]:
```
#dropping the above columns from train dataset
df_train.drop(columns=['v2a1','v18q1','rez_esc'], axis=1, inplace=True)
```

[25]:
```
#reaplcing nan of 'meaneduc' and 'SQBmeaned' with median value of respective␣
↪columns
df_train['meaneduc'].fillna(df_train['meaneduc'].median(),inplace=True)
df_train['SQBmeaned'].fillna(df_train['SQBmeaned'].median(),inplace=True)
```

[26]:
```
#rechecking for null value in train data set
np.sum(df_train.isnull().sum())
```

[26]:
```
0
```

Now there is no nuyll values in train dataset

[27]:
```
#checking for missing values for test dataset
np.sum(df_test.isnull().sum())
```

[27]:
```
55244
```

[28]:
```
#null columns in the test dataset
null_columns_test=df_test.columns[df_test.isnull().any()]
df_test[null_columns_test].isnull().sum()
```

[28]:
```
v2a1          17403
v18q1         18126
rez_esc       19653
meaneduc         31
SQBmeaned        31
dtype: int64
```

[29]:
```
#checking the percentage of null values in the above columns of test
v2a1_null2=(df_test['v2a1'].isnull().sum()/df_test.shape[0])*100
v18q1_null2=(df_test['v18q1'].isnull().sum()/df_test.shape[0])*100
rez_esc_null2=(df_test['rez_esc'].isnull().sum()/df_test.shape[0])*100

print("Null Value Percentage in v2a1:",v2a1_null2)
print("Null Value Percentage in v18q1:",v18q1_null2)
print("Null Value Percentage in rez_esc:",rez_esc_null2)
```

```
Null Value Percentage in v2a1: 72.95020120724345
Null Value Percentage in v18q1: 75.98088531187123
Null Value Percentage in rez_esc: 82.3817907444668
```

As the above 3 columns have null value more than 50%, so we have to drop that column

[30]:
```
#dropping the above columns from train dataset
df_test.drop(columns=['v2a1','v18q1','rez_esc'], axis=1, inplace=True)
```

[31]:
```
#reaplcing nan of 'meaneduc' and 'SQBmeaned' with median value of respective␣
↪columns
df_test['meaneduc'].fillna(df_test['meaneduc'].median(),inplace=True)
df_test['SQBmeaned'].fillna(df_test['SQBmeaned'].median(),inplace=True)
```

[32]:
```
#rechecking for null value in test data set
np.sum(df_test.isnull().sum())
```

[32]:
```
0
```

Now there is no nuyll values in test dataset

1.2.1 Identify the output variable

[33]:
```
#total columns in train and test dataset
print("total columns in train dataset:", df_train.shape[1])
print("total columns in train dataset:", df_test.shape[1])
```

```
total columns in train dataset: 139
total columns in train dataset: 138
```

[34]:
```
print("Columns in Train Dataset:")
df_train.columns
```

```
Columns in Train Dataset:
```

[34]:
```
Index(['hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'r4h1', 'r4h2',
       'r4h3', 'r4m1',
       …
       'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe', 'SQBhogar_nin',
       'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq', 'Target'],
      dtype='object', length=139)
```

[35]:
```
print("Columns in Test Dataset:")
df_test.columns
```

```
Columns in Test Dataset:
```

[35]:
```
Index(['hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'r4h1', 'r4h2',
       'r4h3', 'r4m1',
       …
       'age', 'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
       'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned',
       'agesq'],
      dtype='object', length=138)
```

Comparing to columns of both the dataset, 'Target' columnsumn in the train dataset is the output column

1.2.2 Understand the type of data

Already this step was done in EDA portion in above

1.2.3 Check if there are any biases in your dataset

[36]:
```
df_train['Target'].unique()
```

[36]:
```
array([4, 2, 3, 1], dtype=int64)
```

[37]:
```
df_train['Target'].value_counts()
```

[37]:
```
4    5996
2    1597
3    1209
1     755
Name: Target, dtype: int64
```

Yes, Bias in there in there dataset as the value counts of 4 is much higher than value counts of 3

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

[38]:
```
print("House Identifier with different poverty level:")
(df_train.groupby('idhogar')['Target'].nunique()>1).index
```

```
House Identifier with different poverty level:
```

[38]:
```
Index(['001ff74ca', '003123ec2', '004616164', '004983866', '005905417',
       '006031de3', '006555fe2', '00693f597', '006b64543', '00941f1f4',
       …
       'ff250fd6c', 'ff31b984b', 'ff38ddef1', 'ff6d16fd0', 'ff703eed4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be1'],
      dtype='object', name='idhogar', length=2988)
```

1.2.5 Check if there is a house without a family head

[39]:
```
print("House identifier without a family head:")
(df_train.groupby('idhogar')['parentesco1'].sum()==0).index
```

```
House identifier without a family head:
```

[39]:
```
Index(['001ff74ca', '003123ec2', '004616164', '004983866', '005905417',
       '006031de3', '006555fe2', '00693f597', '006b64543', '00941f1f4',
       …
       'ff250fd6c', 'ff31b984b', 'ff38ddef1', 'ff6d16fd0', 'ff703eed4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be1'],
      dtype='object', name='idhogar', length=2988)
```

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

[40]:
```
target_mean=df_train.groupby('idhogar')['Target'].mean().astype('int64').
 ↪reset_index().rename(columns={'Target':'Target_Mean'})
target_mean
```

[40]:

	idhogar	Target_Mean
0	001ff74ca	4
1	003123ec2	2
2	004616164	2
3	004983866	3

```
4     005905417            2
…         …          …
2983  ff9343a35            4
2984  ff9d5ab17            4
2985  ffae4a097            4
2986  ffe90d46f            1
2987  fff7d6be1            4

[2988 rows x 2 columns]
```

[41]:
```
df_train=df_train.merge(target_mean,how='left', on='idhogar')
df_train.Target=df_train.Target_Mean
df_train.drop('Target_Mean', axis=1, inplace=True)
```

[42]:
```
df_train.head()
```

[42]:
```
   hacdor  rooms  hacapo  v14a  refrig  v18q  r4h1  r4h2  r4h3  r4m1  … \
0       0      3       0     1       1     0     0     1     1     0  …
1       0      4       0     1       1     1     0     1     1     0  …
2       0      8       0     1       1     0     0     0     0     0  …
3       0      5       0     1       1     1     0     2     2     1  …
4       0      5       0     1       1     1     0     2     2     1  …

   SQBescolari  SQBage  SQBhogar_total  SQBedjefe  SQBhogar_nin  \
0          100    1849               1        100             0
1          144    4489               1        144             0
2          121    8464               1          0             0
3           81     289              16        121             4
4          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 139 columns]
```

[43]:
```
#drop the'idhogar' column
print(df_train.shape)
df_train.drop('idhogar', axis=1, inplace=True)
print(df_train.shape)
print("")
print(df_test.shape)
df_test.drop('idhogar', axis=1, inplace=True)
print(df_test.shape)
```

```
(9557, 139)
(9557, 138)

(23856, 138)
(23856, 137)
```

1.2.7 Count how many null values are existing in columns

Counting and treating null values are already idone in EDA above

1.2.8 Remove null value rows of the target variable

[44]:
```
df_train['Target'].isnull().sum()
```

[44]:
```
0
```

There is no null values in the Target column

1.2.9 Predict the accuracy using random forest classifier

[45]:
```
#Finding X and y
X=df_train.drop('Target', axis=1)
y=df_train['Target']
```

[46]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
```

[47]:
```
RF=RandomForestClassifier(min_samples_leaf=10, min_samples_split=10,␣
 ↪random_state=7)
RF.fit(X,y)
```

[47]:
```
RandomForestClassifier(min_samples_leaf=10, min_samples_split=10,
                       random_state=7)
```

[48]:
```
#predict
pred_train=RF.predict(X)
```

[49]:
```
pred_train
```

[49]:
```
array([4, 4, 4, …, 4, 4, 4], dtype=int64)
```

[50]:
```
#accuracy
accuracy=accuracy_score(y,pred_train)
print("Accuracy Score=", accuracy)
```

```
Accuracy Score= 0.8428377105786334
```

[51]:
```
print("Confusion Matrix:")
confusion_matrix(y,pred_train)
```

```
Confusion Matrix:
```

[51]:
```
array([[ 536,   23,    0,  299],
       [   2, 1062,    1,  510],
       [  10,   27,  500,  617],
       [   5,    7,    1, 5957]], dtype=int64)
```

[52]:
```
print("Classification Report:")
classification_report(y,pred_train)
```

```
Classification Report:
```

[52]:
```
'              precision    recall  f1-score   support\n\n           1
0.97      0.62      0.76       858\n           2       0.95      0.67      0.79
1575\n           3       1.00      0.43      0.60      1154\n           4
0.81      1.00      0.89      5970\n\n    accuracy
0.84      9557\n   macro avg       0.93      0.68      0.76      9557\nweighted
avg       0.87      0.84      0.83      9557\n'
```

1.2.10 Check the accuracy using random forest with cross validation

[53]:
```
#cross validation
from sklearn.model_selection import cross_val_score
```

[54]:
```
cv=cross_val_score(RF,X,y, scoring='accuracy', cv=10)
```

[55]:
```
cv
```

[55]:
```
array([0.65585774, 0.65481172, 0.66317992, 0.63493724, 0.65376569,
       0.65690377, 0.63702929, 0.62303665, 0.58638743, 0.63350785])
```

[56]:
```
print("Accuracy of Random Forest with cross validation=", np.mean(cv)*100)
```

```
Accuracy of Random Forest with cross validation= 63.9941729282131
```

[57]:
```
min(cv)
```

[57]:
```
0.5863874345549738
```

[58]:
```
max(cv)
```

[58]:
```
0.6631799163179917
```

[ ]:
```

```