ML Project 2 - Income Qualification

February 10, 2023

1 ML Project 2 - Income Qualification

[1]:
```
import pandas as pd
import numpy as np
```

2 Understanding the Data

- Identify the output variable.
- Understand the type of data.

[2]:
```
train = pd.read_csv('ml2train.csv')
test = pd.read_csv('ml2test.csv')
```

[3]:
```
train
```

[3]:
```
                Id      v2a1  hacdor  rooms  hacapo  v14a  refrig  v18q  \
0     ID_279628684  190000.0       0      3       0     1       1     0
1     ID_f29eb3ddd  135000.0       0      4       0     1       1     1
2     ID_68de51c94       NaN       0      8       0     1       1     0
3     ID_d671db89c  180000.0       0      5       0     1       1     1
4     ID_d56d6f5f5  180000.0       0      5       0     1       1     1
…              …       …     …      …    …      …   …       …   …
9552  ID_d45ae367d   80000.0       0      6       0     1       1     0
9553  ID_c94744e07   80000.0       0      6       0     1       1     0
9554  ID_85fc658f8   80000.0       0      6       0     1       1     0
9555  ID_ced540c61   80000.0       0      6       0     1       1     0
9556  ID_a38c64491   80000.0       0      6       0     1       1     0

      v18q1  r4h1  …  SQBescolari  SQBage  SQBhogar_total  SQBedjefe  \
0       NaN     0  …          100    1849               1        100
1       1.0     0  …          144    4489               1        144
2       NaN     0  …          121    8464               1          0
3       1.0     0  …           81     289              16        121
4       1.0     0  …          121    1369              16        121
…     …   …   …  …          …      …                …         …
9552    NaN     0  …           81    2116              25         81
9553    NaN     0  …            0       4              25         81
```

```
9554    NaN     0  …           25    2500              25         81
9555    NaN     0  …          121     676              25         81
9556    NaN     0  …           64     441              25         81

      SQBhogar_nin  SQBovercrowding  SQBdependency  SQBmeaned  agesq  Target
0                0         1.000000         0.0000   100.0000   1849       4
1                0         1.000000        64.0000   144.0000   4489       4
2                0         0.250000        64.0000   121.0000   8464       4
3                4         1.777778         1.0000   121.0000    289       4
4                4         1.777778         1.0000   121.0000   1369       4
…              …                 …              …         …     …     …
9552             1         1.562500         0.0625    68.0625   2116       2
9553             1         1.562500         0.0625    68.0625      4       2
9554             1         1.562500         0.0625    68.0625   2500       2
9555             1         1.562500         0.0625    68.0625    676       2
9556             1         1.562500         0.0625    68.0625    441       2

[9557 rows x 143 columns]
```

[4]:
```
train.shape
```

[4]:
```
(9557, 143)
```

[5]:
```
train.dtypes
```

[5]:
```
Id                 object
v2a1              float64
hacdor              int64
rooms               int64
hacapo              int64
                    …
SQBovercrowding   float64
SQBdependency     float64
SQBmeaned         float64
agesq               int64
Target              int64
Length: 143, dtype: object
```

[6]:
```
test
```

[6]:
```
                  Id      v2a1  hacdor  rooms  hacapo  v14a  refrig  v18q  \
0      ID_2f6873615       NaN       0      5       0     1       1     0
1      ID_1c78846d2       NaN       0      5       0     1       1     0
2      ID_e5442cf6a       NaN       0      5       0     1       1     0
3      ID_a8db26a79       NaN       0     14       0     1       1     1
4      ID_a62966799  175000.0       0      4       0     1       1     1
…              …       …     …    …     …    …   …     …   …
```

```
23851  ID_a065a7cad       NaN       1      2       1     1       1     0
23852  ID_1a7c6953b       NaN       0      3       0     1       1     0
23853  ID_07dbb4be2       NaN       0      3       0     1       1     0
23854  ID_34d2ed046       NaN       0      3       0     1       1     0
23855  ID_34754556f       NaN       0      3       0     1       1     0

       v18q1  r4h1  …  age  SQBescolari  SQBage  SQBhogar_total  SQBedjefe  \
0        NaN     1  …    4            0      16               9          0
1        NaN     1  …   41          256    1681               9          0
2        NaN     1  …   41          289    1681               9          0
3        1.0     0  …   59          256    3481               1        256
4        1.0     0  …   18          121     324               1          0
…        …   …   …  …    …            …       …               …          …
23851    NaN     0  …   10            9     100              36         25
23852    NaN     0  …   54           36    2916              16         36
23853    NaN     0  …   12           16     144              16         36
23854    NaN     0  …   12           25     144              16         36
23855    NaN     0  …   51           36    2601              16         36

       SQBhogar_nin  SQBovercrowding  SQBdependency  SQBmeaned  agesq
0                 1             2.25           0.25   272.2500     16
1                 1             2.25           0.25   272.2500   1681
2                 1             2.25           0.25   272.2500   1681
3                 0             1.00           0.00   256.0000   3481
4                 1             0.25          64.00        NaN    324
…               …                …              …         …    …
23851             4            36.00           0.25    33.0625    100
23852             4             4.00           1.00    36.0000   2916
23853             4             4.00           1.00    36.0000    144
23854             4             4.00           1.00    36.0000    144
23855             4             4.00           1.00    36.0000   2601

[23856 rows x 142 columns]
```

[7]:
```
test.shape
```

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

[8]:
```
test.dtypes
```

[8]:
```
Id                 object
v2a1              float64
hacdor              int64
rooms               int64
hacapo              int64
                    …
SQBhogar_nin        int64
```

```
SQBovercrowding    float64
SQBdependency      float64
SQBmeaned          float64
agesq                int64
Length: 142, dtype: object
```

[9]:
```
for col in train.columns:
    print("Column '{}' data type: {}".format(col,train[col].dtypes))
```

```
Column 'Id' data type: object
Column 'v2a1' data type: float64
Column 'hacdor' data type: int64
Column 'rooms' data type: int64
Column 'hacapo' data type: int64
Column 'v14a' data type: int64
Column 'refrig' data type: int64
Column 'v18q' data type: int64
Column 'v18q1' data type: float64
Column 'r4h1' data type: int64
Column 'r4h2' data type: int64
Column 'r4h3' data type: int64
Column 'r4m1' data type: int64
Column 'r4m2' data type: int64
Column 'r4m3' data type: int64
Column 'r4t1' data type: int64
Column 'r4t2' data type: int64
Column 'r4t3' data type: int64
Column 'tamhog' data type: int64
Column 'tamviv' data type: int64
Column 'escolari' data type: int64
Column 'rez_esc' data type: float64
Column 'hhsize' data type: int64
Column 'paredblolad' data type: int64
Column 'paredzocalo' data type: int64
Column 'paredpreb' data type: int64
Column 'pareddes' data type: int64
Column 'paredmad' data type: int64
Column 'paredzinc' data type: int64
Column 'paredfibras' data type: int64
Column 'paredother' data type: int64
Column 'pisomoscer' data type: int64
Column 'pisocemento' data type: int64
Column 'pisoother' data type: int64
Column 'pisonatur' data type: int64
Column 'pisonotiene' data type: int64
Column 'pisomadera' data type: int64
Column 'techozinc' data type: int64
Column 'techoentrepiso' data type: int64
```

```
Column 'techocane' data type: int64
Column 'techootro' data type: int64
Column 'cielorazo' data type: int64
Column 'abastaguadentro' data type: int64
Column 'abastaguafuera' data type: int64
Column 'abastaguano' data type: int64
Column 'public' data type: int64
Column 'planpri' data type: int64
Column 'noelec' data type: int64
Column 'coopele' data type: int64
Column 'sanitario1' data type: int64
Column 'sanitario2' data type: int64
Column 'sanitario3' data type: int64
Column 'sanitario5' data type: int64
Column 'sanitario6' data type: int64
Column 'energcocinar1' data type: int64
Column 'energcocinar2' data type: int64
Column 'energcocinar3' data type: int64
Column 'energcocinar4' data type: int64
Column 'elimbasu1' data type: int64
Column 'elimbasu2' data type: int64
Column 'elimbasu3' data type: int64
Column 'elimbasu4' data type: int64
Column 'elimbasu5' data type: int64
Column 'elimbasu6' data type: int64
Column 'epared1' data type: int64
Column 'epared2' data type: int64
Column 'epared3' data type: int64
Column 'etecho1' data type: int64
Column 'etecho2' data type: int64
Column 'etecho3' data type: int64
Column 'eviv1' data type: int64
Column 'eviv2' data type: int64
Column 'eviv3' data type: int64
Column 'dis' data type: int64
Column 'male' data type: int64
Column 'female' data type: int64
Column 'estadocivil1' data type: int64
Column 'estadocivil2' data type: int64
Column 'estadocivil3' data type: int64
Column 'estadocivil4' data type: int64
Column 'estadocivil5' data type: int64
Column 'estadocivil6' data type: int64
Column 'estadocivil7' data type: int64
Column 'parentesco1' data type: int64
Column 'parentesco2' data type: int64
Column 'parentesco3' data type: int64
Column 'parentesco4' data type: int64
```

```
Column 'parentesco5' data type: int64
Column 'parentesco6' data type: int64
Column 'parentesco7' data type: int64
Column 'parentesco8' data type: int64
Column 'parentesco9' data type: int64
Column 'parentesco10' data type: int64
Column 'parentesco11' data type: int64
Column 'parentesco12' data type: int64
Column 'idhogar' data type: object
Column 'hogar_nin' data type: int64
Column 'hogar_adul' data type: int64
Column 'hogar_mayor' data type: int64
Column 'hogar_total' data type: int64
Column 'dependency' data type: object
Column 'edjefe' data type: object
Column 'edjefa' data type: object
Column 'meaneduc' data type: float64
Column 'instlevel1' data type: int64
Column 'instlevel2' data type: int64
Column 'instlevel3' data type: int64
Column 'instlevel4' data type: int64
Column 'instlevel5' data type: int64
Column 'instlevel6' data type: int64
Column 'instlevel7' data type: int64
Column 'instlevel8' data type: int64
Column 'instlevel9' data type: int64
Column 'bedrooms' data type: int64
Column 'overcrowding' data type: float64
Column 'tipovivi1' data type: int64
Column 'tipovivi2' data type: int64
Column 'tipovivi3' data type: int64
Column 'tipovivi4' data type: int64
Column 'tipovivi5' data type: int64
Column 'computer' data type: int64
Column 'television' data type: int64
Column 'mobilephone' data type: int64
Column 'qmobilephone' data type: int64
Column 'lugar1' data type: int64
Column 'lugar2' data type: int64
Column 'lugar3' data type: int64
Column 'lugar4' data type: int64
Column 'lugar5' data type: int64
Column 'lugar6' data type: int64
Column 'area1' data type: int64
Column 'area2' data type: int64
Column 'age' data type: int64
Column 'SQBescolari' data type: int64
Column 'SQBage' data type: int64
```

```
Column 'SQBhogar_total' data type: int64
Column 'SQBedjefe' data type: int64
Column 'SQBhogar_nin' data type: int64
Column 'SQBovercrowding' data type: float64
Column 'SQBdependency' data type: float64
Column 'SQBmeaned' data type: float64
Column 'agesq' data type: int64
Column 'Target' data type: int64
```

3 Missing Values

- Count how many null values are existing in columns.
- Remove null value rows of the target variable.

[10]:
```
for col in train.columns:
    nullcount = train[col].isnull().sum()
    if nullcount != 0:
        print('Column {} missing count {}'.format(col,nullcount))
```

```
Column v2a1 missing count 6860
Column v18q1 missing count 7342
Column rez_esc missing count 7928
Column meaneduc missing count 5
Column SQBmeaned missing count 5
```

[11]:
```
for df in [train,test]:
    df['v2a1'].fillna(value=0,inplace=True)
```

[12]:
```
train['v2a1'].isnull().sum()
```

[12]: 0

[13]:
```
for df in [train,test]:
    df['v18q1'].fillna(value=0,inplace=True)

train['v18q1'].isnull().sum()
```

[13]: 0

[14]:
```
for df in [train,test]:
    df['rez_esc'].fillna(value=0,inplace=True)

train['rez_esc'].isnull().sum()
```

[14]: 0

[15]:
```
for df in [train,test]:
    df['meaneduc'].fillna(value=0,inplace=True)
```

```
train['meaneduc'].isnull().sum()
```

[15]: 0

[16]:
```
for df in[train,test]:
    df['SQBmeaned'].fillna(value=0,inplace=True)

train['SQBmeaned'].isnull().sum()
```

[16]: 0

[17]:
```
train.select_dtypes(object)
```

[17]:

	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 x 5 columns]

[18]:
```
mapping = {'yes':1,'no':0}
```

[19]:
```
for df in [train,test]:
    df['dependency'] = df['dependency'].replace(mapping).astype(np.float64)
    df['edjefe'] = df['edjefe'].replace(mapping).astype(np.float64)
    df['edjefa'] = df['edjefa'].replace(mapping).astype(np.float64)

train[['dependency','edjefe','edjefa']].describe()
```

[19]:

	dependency	edjefe	edjefa
count	9557.000000	9557.000000	9557.000000
mean	1.149550	5.096788	2.896830
std	1.605993	5.246513	4.612056
min	0.000000	0.000000	0.000000
25%	0.333333	0.000000	0.000000
50%	0.666667	6.000000	0.000000
75%	1.333333	9.000000	6.000000
max	8.000000	21.000000	21.000000

4 Check for household members poverty level consistency

- 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.

[20]:
```
all_equal = train.groupby('idhogar')['Target'].apply(lambda x:x.nunique()==1)
```

[21]:
```
not_equal = all_equal[all_equal != True]
print('{} households with inconsistent poverty level'.format(len(not_equal)))
```

```
85 households with inconsistent poverty level
```

[22]:
```
household_head = train.groupby('idhogar')['parentesco1'].sum()
```

[23]:
```
household_no_head = train.loc[train['idhogar'].
 ↪isin(household_head[household_head == 0].index),:]
```

[24]:
```
print('{} Household without head'.format(household_no_head['idhogar'].
 ↪nunique()))
```

```
15 Household without head
```

[25]:
```
household_no_head_equal = household_no_head.groupby('idhogar')['Target'].
 ↪apply(lambda x:x.nunique()==1)
print('{} Household with no head have different poverty levels'.
 ↪format(sum(household_no_head_equal==False)))
```

```
0 Household with no head have different poverty levels
```

[26]:
```
for household in not_equal.index:
    true_target = int(train[(train['idhogar']==household) &␣
 ↪(train['parentesco1']==1.0)]['Target'])
    train.loc[train['idhogar']==household,'Target'] = true_target
```

[27]:
```
all_equal = train.groupby('idhogar')['Target'].apply(lambda x:x.nunique()==1)
```

[28]:
```
not_equal = all_equal[all_equal != True]
print('{} household with inconsistent poverty level'.format(len(not_equal)))
```

```
0 household with inconsistent poverty level
```

5 Bias

- Check if there are any biases in your dataset.

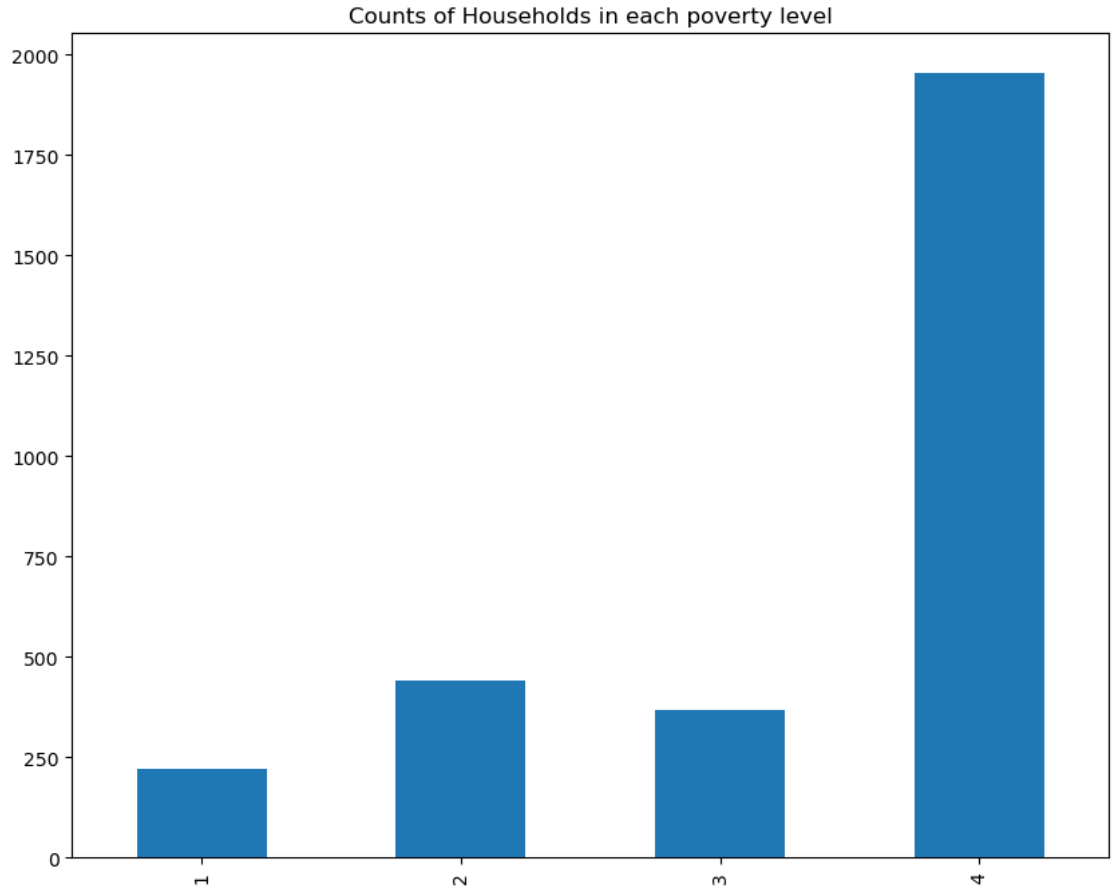
[29]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[30]:
```
target_counts = train.groupby('idhogar')['Target'].first().value_counts().
↪sort_index()
```

[31]:
```
target_counts
```

[31]:
```
1     222
2     442
3     369
4    1955
Name: Target, dtype: int64
```

[32]:
```
plot1 = target_counts.plot(kind='bar',figsize=(10,8), linewidth=2,␣
↪title='Counts of Households in each poverty level')
```



[33]:
```
train.shape
```

[33]:
```
(9557, 143)
```

[34]:
```
cols =␣
↪['SQBescolari','SQBage','SQBhogar_total','SQBedjefe','SQBhogar_nin','SQBovercrowding','SQBd
```

[35]:
```
for df in [train,test]:
    df.drop(columns=cols,inplace=True)
```

[36]:
```
train.shape
```

[36]: (9557, 126)

6 Prediction using Random Forest Classifier

- Predict the accuracy using random forest classifier.
- Check the accuracy using random forest with cross validation.

[37]:
```
X_train_data = train.drop(['Target'],axis=1)
y_train_data = train['Target']
```

[38]:
```
X_train_data
```

[38]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	\
0	190000.0	0	3	0	1	1	0	0.0	0	1	
1	135000.0	0	4	0	1	1	1	1.0	0	1	
2	0.0	0	8	0	1	1	0	0.0	0	0	
3	180000.0	0	5	0	1	1	1	1.0	0	2	
4	180000.0	0	5	0	1	1	1	1.0	0	2	
…	…	…	…	…	…	…	…	…	…	…	
9552	80000.0	0	6	0	1	1	0	0.0	0	2	
9553	80000.0	0	6	0	1	1	0	0.0	0	2	
9554	80000.0	0	6	0	1	1	0	0.0	0	2	
9555	80000.0	0	6	0	1	1	0	0.0	0	2	
9556	80000.0	0	6	0	1	1	0	0.0	0	2	

	…	mobilephone	qmobilephone	lugar1	lugar2	lugar3	lugar4	lugar5	\
0	…	1	1	1	0	0	0	0	
1	…	1	1	1	0	0	0	0	
2	…	0	0	1	0	0	0	0	
3	…	1	3	1	0	0	0	0	
4	…	1	3	1	0	0	0	0	
…	…	…	…	…	…	…	…	…	
9552	…	1	3	0	0	0	0	0	
9553	…	1	3	0	0	0	0	0	
9554	…	1	3	0	0	0	0	0	
9555	…	1	3	0	0	0	0	0	
9556	…	1	3	0	0	0	0	0	

	lugar6	area1	age

```
0          0      1   43
1          0      1   67
2          0      1   92
3          0      1   17
4          0      1   37
…        …    …   …
9552       1      0   46
9553       1      0    2
9554       1      0   50
9555       1      0   26
9556       1      0   21

[9557 rows x 125 columns]
```

[39]:
```
y_train_data
```

[39]:
```
0       4
1       4
2       4
3       4
4       4
       ..
9552    2
9553    2
9554    2
9555    2
9556    2
Name: Target, Length: 9557, dtype: int64
```

[40]:
```
from sklearn.model_selection import train_test_split
```

[41]:
```
X_train,X_test,y_train,y_test =␣
↪train_test_split(X_train_data,y_train_data,test_size=0.20,random_state=123)
```

[42]:
```
from sklearn.ensemble import RandomForestClassifier
```

[43]:
```
rfc = RandomForestClassifier()
```

[44]:
```
rfc.fit(X_train,y_train)
```

[44]:
```
RandomForestClassifier()
```

[45]:
```
y_pred_train = rfc.predict(X_test)
```

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

[47]:
```
ac_train_data = accuracy_score(y_test,y_pred_train)*100
cm_train_data = confusion_matrix(y_test,y_pred_train)
cr_train_data = classification_report(y_test,y_pred_train)
```

[48]:
```
print('Accuracy Score :',ac_train_data)
print('-----------------------------------------------------------')
print('Confusion Matrix \n',cm_train_data)
print('-----------------------------------------------------------')
print('Classification Report \n',cr_train_data)
```

```
Accuracy Score : 93.82845188284519
-----------------------------------------------------------
Confusion Matrix
 [[ 125    3    1   22]
 [   0  276    1   32]
 [   0    3  186   56]
 [   0    0    0 1207]]
-----------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           1       1.00      0.83      0.91       151
           2       0.98      0.89      0.93       309
           3       0.99      0.76      0.86       245
           4       0.92      1.00      0.96      1207

    accuracy                           0.94      1912
   macro avg       0.97      0.87      0.91      1912
weighted avg       0.94      0.94      0.94      1912
```

[49]:
```
from sklearn.model_selection import cross_val_score
```

[50]:
```
from sklearn.model_selection import KFold
```

[51]:
```
kfold = KFold(n_splits=10,shuffle=True,random_state=7)
model1 = RandomForestClassifier(n_estimators=100,max_features=3)
```

[52]:
```
score_train =␣
 ↪cross_val_score(model1,X_train_data,y_train_data,cv=kfold,scoring='accuracy')
```

[53]:
```
print('Cross Validation Accuracy :',score_train.mean()*100)
```

```
Cross Validation Accuracy : 91.32557120637912
```

7 Accuracy from the Random Forest : 93.8%

8 Accuracy from Cross Validation : 91.6%

[ ]: