Project 2 - Income Qualification

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')


# Dataset :- train.csv and  test.csv
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
print(df_train.shape)
print(df_test.shape)
#print(df_train.dtypes)
#print(df_test.dtypes)
```

```
(9557, 143)
(23856, 142)
```

```
df_train.head()
```

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4l
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	

5 rows × 143 columns

```
df_test.head()
```

```
# check missing value/NULL in training data
df_train.isnull().sum()
```

```
Id                     0
v2a1                6860
hacdor                 0
rooms                  0
hacapo                 0
                    ...
SQBovercrowding        0
SQBdependency          0
SQBmeaned              5
agesq                  0
Target                 0
Length: 143, dtype: int64
```

```
# check missing value/NULL in testing data
df_test.isnull().sum()
```

```
Id                     0
v2a1                17403
hacdor                 0
rooms                  0
hacapo                 0
                    ...
SQBhogar_nin           0
SQBovercrowding        0
SQBdependency          0
SQBmeaned             31
agesq                  0
Length: 142, dtype: int64
```

```
# descriptive analysis
df_train.describe()
```

	v2a1	hacdor	rooms	hacapo	v14a	refrig	
count	2.697000e+03	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	9
mean	1.652316e+05	0.038087	4.955530	0.023648	0.994768	0.957623	
std	1.504571e+05	0.191417	1.468381	0.151957	0.072145	0.201459	
min	0.000000e+00	0.000000	1.000000	0.000000	0.000000	0.000000	
25%	8.000000e+04	0.000000	4.000000	0.000000	1.000000	1.000000	
50%	1.300000e+05	0.000000	5.000000	0.000000	1.000000	1.000000	
75%	2.000000e+05	0.000000	6.000000	0.000000	1.000000	1.000000	
max	2.353477e+06	1.000000	11.000000	1.000000	1.000000	1.000000	

8 rows × 138 columns

```
# as we can see training dataset has very less number of rows/observation
# than testing dataset.
# also testing dataset don't have the target column.
# So the splitted data is not very impressive.
# we will create a new target column (same as training) in testing dataset
# and then append testing dataset after training dataset
 df_test['Target'] = np.nan
 #df_test.shape
 df_test.isnull().sum()
```

```
Id                  0
v2a1            17403
hacdor              0
rooms               0
hacapo              0
                  ...
SQBovercrowding     0
SQBdependency       0
SQBmeaned          31
agesq               0
Target          23856
Length: 143, dtype: int64
```

```
# append testing dataset after training dataset
df_appended = df_train.append(df_test)
df_appended.shape
#df_appended.isnull().sum()
#df_appended.head()
```

```
(33413, 143)
```

```
# NULL value checking. Replace NULL/Nan with mean value
df_appended.fillna(df_appended.mean(),inplace = True)
df_appended.isnull().sum()
#df_appended.head()
```

```
Id                0
v2a1              0
hacdor            0
rooms             0
hacapo            0
                 ..
SQBovercrowding   0
SQBdependency     0
SQBmeaned         0
agesq             0
Target            0
Length: 143, dtype: int64
```

```
# We will only consider some important columns as below :
#df_appended_new = df_appended[['idhogar','hacapo','v14a','rooms','v2a1','parentesco1','te
df_appended_new = df_appended[['idhogar','hacapo','v14a','rooms', \
                               'v2a1','parentesco1','television','computer', \
                               'v18q','refrig','tipovivi3','tipovivi2','tipovivi1', \
                               'paredzocalo','paredblolad','cielorazo','Target']]

# convert column "Target" to Integer (int64)
df_appended_new['Target'] = df_appended_new['Target'].astype(np.int64)
df_appended_new
```

	idhogar	hacapo	v14a	rooms	v2a1	parentesco1	television	comput
0	21eb7fcc1	0	1	3	190000.000000	1	0	
1	0e5d7a658	0	1	4	135000.000000	1	0	
2	2c7317ea8	0	1	8	172030.845574	1	0	
3	2b58d945f	0	1	5	180000.000000	0	0	
4	2b58d945f	0	1	5	180000.000000	0	0	
...	...	...	...	...	...	...	...	
23851	3aa78c56b	1	1	2	172030.845574	0	0	
23852	d237404b6	0	1	3	172030.845574	1	0	
23853	d237404b6	0	1	3	172030.845574	0	0	
23854	d237404b6	0	1	3	172030.845574	0	0	
23855	d237404b6	0	1	3	172030.845574	0	0	

33413 rows × 17 columns

```
# Check if there is a house without a family head.
# column :- #parentesco1, =1 if household head

df_appended_new_1 = df_appended_new
filter = df_appended_new_1['parentesco1'] != 1
df_appended_new_1 = df_appended_new_1[filter]
df_appended_new_1
```

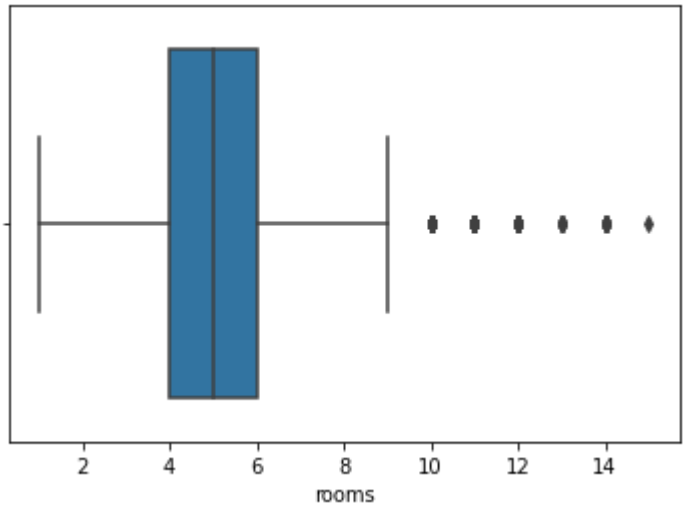
	idhogar	hacapo	v14a	rooms	v2a1	parentesco1	television	comput
3	2b58d945f	0	1	5	180000.000000	0	0	
4	2b58d945f	0	1	5	180000.000000	0	0	
6	2b58d945f	0	1	5	180000.000000	0	0	
7	d6dae86b7	0	1	2	130000.000000	0	0	
9	d6dae86b7	0	1	2	130000.000000	0	0	
...	...	...	...	...	...	...	...	
23850	3aa78c56b	1	1	2	172030.845574	0	0	
23851	3aa78c56b	1	1	2	172030.845574	0	0	
23853	d237404b6	0	1	3	172030.845574	0	0	
23854	d237404b6	0	1	3	172030.845574	0	0	
23855	d237404b6	0	1	3	172030.845574	0	0	

23106 rows × 17 columns

```
# check outliers of the feature/independent columns
#df_appended_new_filtered = df_appended_new[['hacapo','v14a','rooms','v2a1','refrig','v18c
df_appended_new_filtered = df_appended_new[['hacapo','v14a','rooms','v2a1', \
                                            'refrig','v18q','television', \
                                            'computer','tipovivi3','tipovivi2', \
                                            'tipovivi1','cielorazo','paredzocalo', \
                                            'paredblolad','Target']]

# here the boxplot of the column 'v2a1' don't give satisfied result,
# so we can ignore this.
# NOTE : also other than rooms, rest of the independent
# columns having 0 or 1 value, so we can ignore them also.
#sns.boxplot(x=df_appended_new_filtered['v2a1'])
sns.boxplot(x=df_appended_new_filtered['rooms'])
```

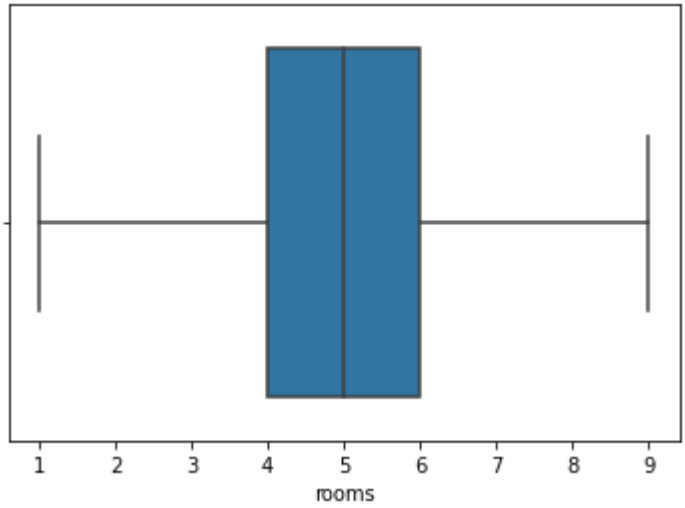
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66ff07b550>
```




```
# The above boxplot output shows the outliers where rooms > 9 ,
# so we can remove the outliers.
# The outlier is something which is separate/different from the crowd.
```

```
# remove outliers using filter the data where rooms <=9
filter = df_appended_new_filtered['rooms']<=9
df_appended_new_filtered = df_appended_new_filtered[filter]
df_appended_new_filtered
# now again check the outliers using boxplot
sns.boxplot(x=df_appended_new_filtered['rooms'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66ff2336a0>
```



The above output shows there is no outlier present in the dataset.

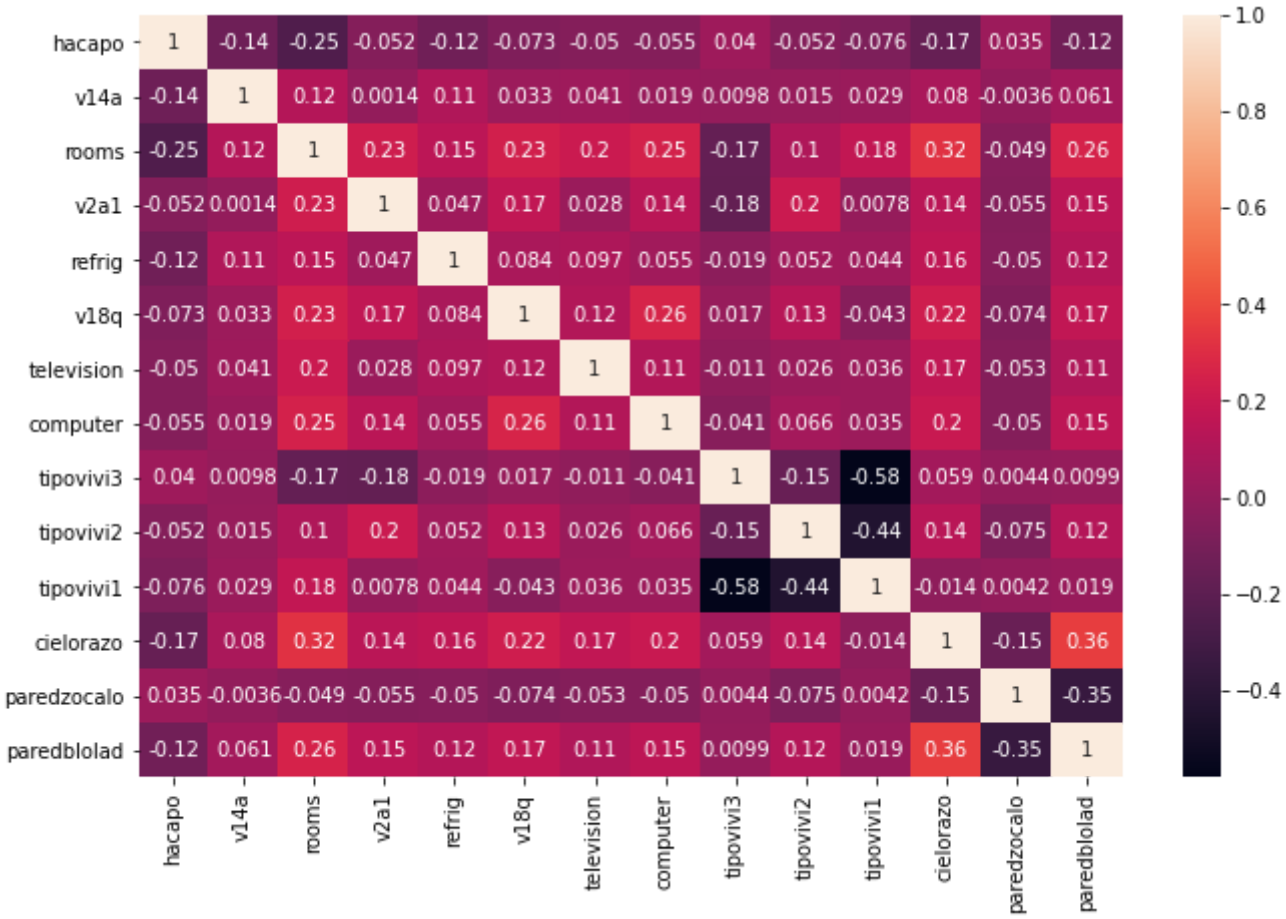
```
df_appended_new_filtered
```

	hacapo	v14a	rooms	v2a1	refrig	v18q	television	computer	tipovi
0	0	1	3	190000.000000	1	0	0	0	
1	0	1	4	135000.000000	1	1	0	0	
2	0	1	8	172030.845574	1	0	0	0	
3	0	1	5	180000.000000	1	1	0	0	
4	0	1	5	180000.000000	1	1	0	0	
...	...	...	...	...	...	...	...	...	
23851	1	1	2	172030.845574	1	0	0	0	
23852	0	1	3	172030.845574	1	0	0	0	
23853	0	1	3	172030.845574	1	0	0	0	
23854	0	1	3	172030.845574	1	0	0	0	
23855	0	1	3	172030.845574	1	0	0	0	

33118 rows × 15 columns

```
# Check coorelation using heatmap
# here just removed Target
#df_appended_new_filtered_1 = df_appended_new_filtered[['hacapo','v14a','rooms','v2a1','re
df_appended_new_filtered_1 = df_appended_new_filtered[['hacapo','v14a', \
                                                       'rooms','v2a1', \
                                                       'refrig','v18q', \
                                                       'television','computer', \
                                                       'tipovivi3','tipovivi2', \
                                                       'tipovivi1','cielorazo', \
                                                       'paredzocalo','paredblolad']]

plt.figure(figsize = (11,7))
correlations = df_appended_new_filtered_1.corr()
sns.heatmap(data = correlations,annot=True)
plt.show()
```

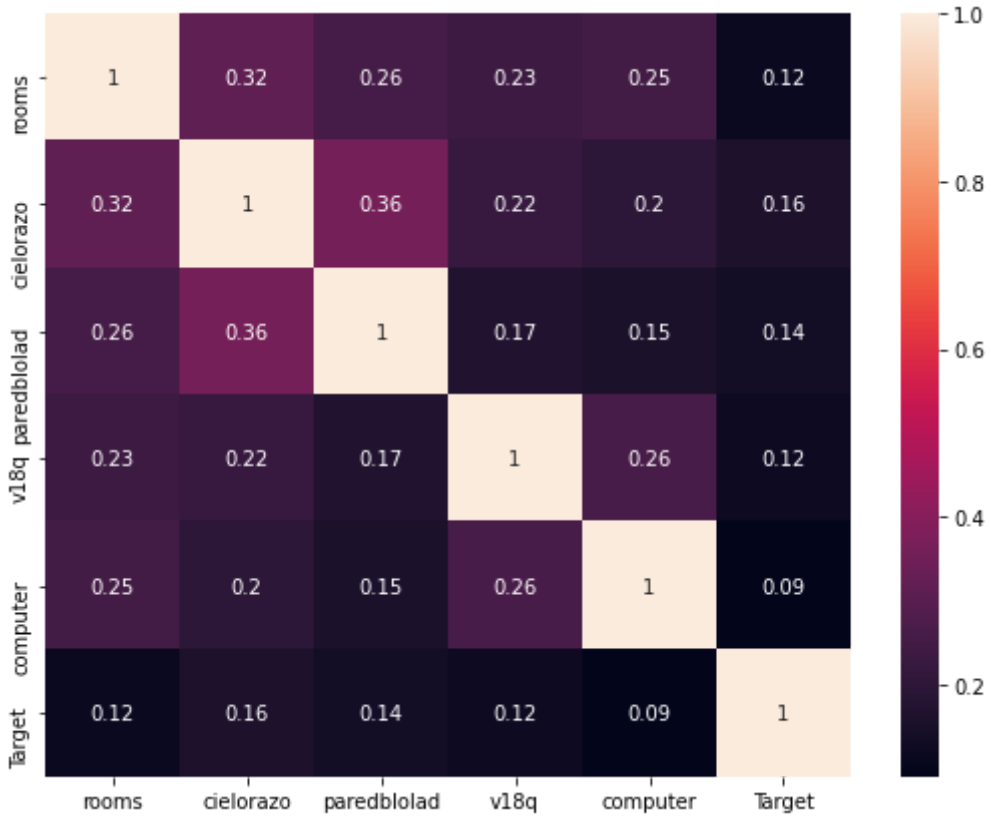


The above output shows there has been moderate positive correlation between the below :

1. cielorazo(if the house has ceiling) &
   paredblolad(if predominant material on the outside wall is block or brick)
2. rooms(number of all rooms in the house) &
   computer(if the household has notebook or desktop computer)
3. v18q (owns a tablet) & computer(if the household has notebook or desktop computer)

```
# Check coorelation using heatmap
# here just added Target and
# taking only the above 5 variables(rooms,cielorazo,paredblolad,v18q,computer) from the ab
#df_appended_new_filtered_1 = df_appended_new_filtered[['rooms','cielorazo','paredblolad',
df_appended_new_filtered_1 = df_appended_new_filtered[['rooms','cielorazo', \
                                                       'paredblolad','v18q', \
                                                       'computer','Target']]

plt.figure(figsize = (9,7))
correlations = df_appended_new_filtered_1.corr()
sns.heatmap(data = correlations,annot=True)
plt.show()
```



The above output shows that Target (i.e. Poverty Level) mostly depends on the below :

cielorazo(if the house has ceiling) & paredblolad(if predominant material on the outside wall is blo

```
# Select X and Y
df_appended_new_filtered_1.head()
X = df_appended_new_filtered_1[['cielorazo','paredblolad']]
Y = df_appended_new_filtered_1['Target']
#X
#Y
```

```
# split the data in training and testing with 70:30 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.30, random_state =
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(23182, 2)
(9936, 2)
(23182,)
(9936,)
```

```
# Model Building - Apply Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

rfc = RandomForestClassifier()
rfc.fit(X_train,Y_train)
Y_predict = rfc.predict(X_test)
accuracy = accuracy_score(Y_predict,Y_test)
print("Accuracy of Random Forest is: " , accuracy)
```

```
Accuracy of Random Forest is:  0.7466787439613527
```

```
# Split the dataset into traing and testing using
# K-Fold Cross Validation Split
from sklearn.model_selection import KFold

#kf = KFold(n_splits = 5, shuffle=True)
kf = KFold(n_splits = 10, shuffle=True)

X = df_appended_new_filtered_1[['cielorazo','paredblolad']]
Y = df_appended_new_filtered_1['Target']

for train_index, test_index in kf.split(X,Y):
      #print("Train:", train_index, "Validation:",test_index)
      print("Train:", train_index, "Test:",test_index)
      X_train, X_test = X.iloc[train_index], X.iloc[test_index]
      Y_train, Y_test = Y.iloc[train_index], Y.iloc[test_index]
```

```
Train: [    0     1     2 ... 33115 33116 33117] Test: [   40    53    57 ... 33100 3
Train: [    0     1     2 ... 33115 33116 33117] Test: [   14    19    45 ... 33093 3
Train: [    0     1     2 ... 33114 33115 33117] Test: [    8    13    20 ... 33101 3
Train: [    1     2     3 ... 33113 33115 33116] Test: [    0     6    25 ... 33107 3
Train: [    0     1     2 ... 33115 33116 33117] Test: [    7    16    33 ... 33097 3
Train: [    0     1     2 ... 33115 33116 33117] Test: [    3     9    12 ... 33084 3
Train: [    0     1     2 ... 33115 33116 33117] Test: [    5    24    28 ... 33073 3
Train: [    0     2     3 ... 33115 33116 33117] Test: [    1    17    23 ... 33102 3
Train: [    0     1     3 ... 33114 33116 33117] Test: [    2    10    11 ... 33091 3
Train: [    0     1     2 ... 33115 33116 33117] Test: [    4    31    46 ... 33060 3
```

```
from sklearn.ensemble import RandomForestClassifier
#rf_class = RandomForestClassifier(n_estimators=10)
# n_estimators : Number of trees in random forest
#rfc = RandomForestClassifier(n_estimators=10)
rfc = RandomForestClassifier()
rfc.fit(X_train,Y_train)
Y_predict = rfc.predict(X_test)
```

```
from sklearn.model_selection import cross_val_score

accuracy = cross_val_score(rfc, X, Y, scoring='accuracy', cv = 10).mean() * 100
print("Accuracy of Random Forests with Cross Validation is: " , accuracy)
```

```
Accuracy of Random Forests with Cross Validation is:  74.94414023231009
```