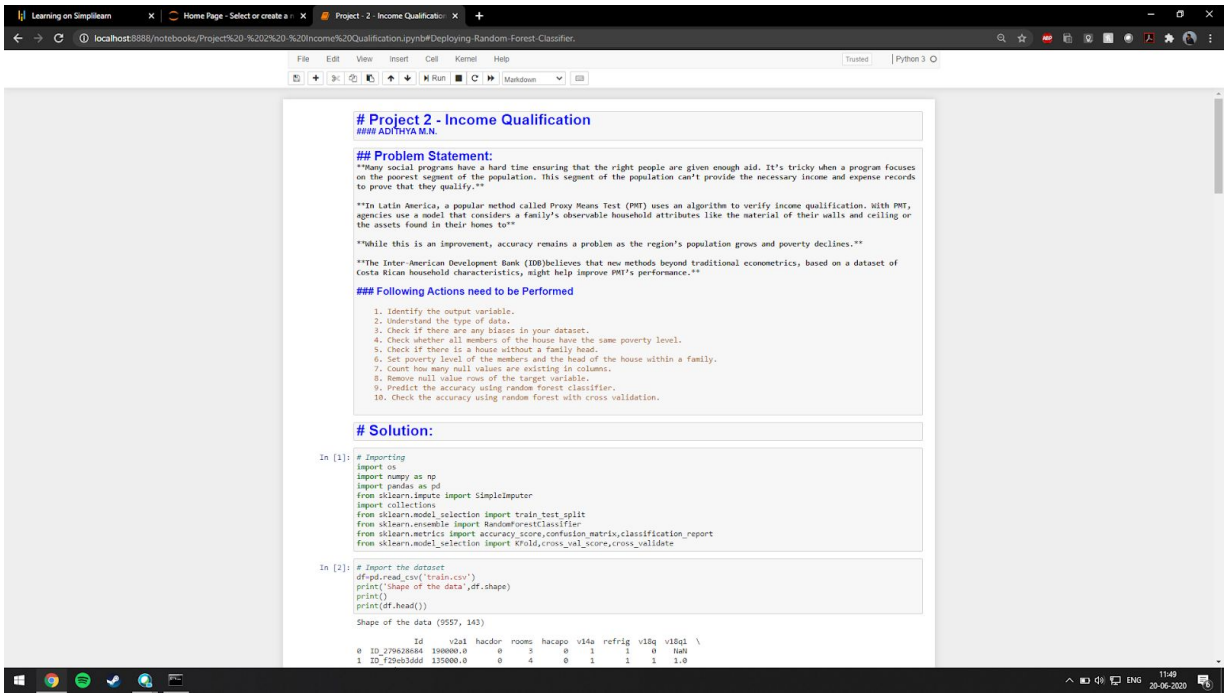
PROJECT - 2 INCOME QUALIFICATION SCREENSHOTS

SOURCE CODE - SCREENSHOTS



Project 2 - Income Qualification

ADITHYA M.N.

Problem Statement:

Many social programs have a hard time ensuring that the right people are given enough aid. It's tricky when a program focuses on the poorest segment of the population. This segment of the population can't provide the necessary income and expense records to prove that they qualify.

In Latin America, a popular method called Proxy Means Test (PMT) uses an algorithm to verify income qualification. With PMT, agencies use a model that considers a family's observable household attributes like the material of their walls and ceiling or the assets found in their homes to

While this is an improvement, accuracy remains a problem as the region's population grows and poverty declines.

The Inter-American Development Bank (IDB)believes that new methods beyond traditional econometrics, based on a dataset of Costa Rican household characteristics, might help improve PMT's performance.

Following Actions need to be Performed

1. Identify the output variable.
2. Understand the type of data.
3. Check if there are any biases in your dataset.
4. Check whether all members of the house have the same poverty level.
5. Check if there is a house without a family head.
6. Set poverty level of the members and the head of the house within a family.
7. Count how many null values are existing in columns.
8. Remove null value rows of the target variable.
9. Predict the accuracy using random forest classifier.
10. Check the accuracy using random forest with cross validation.

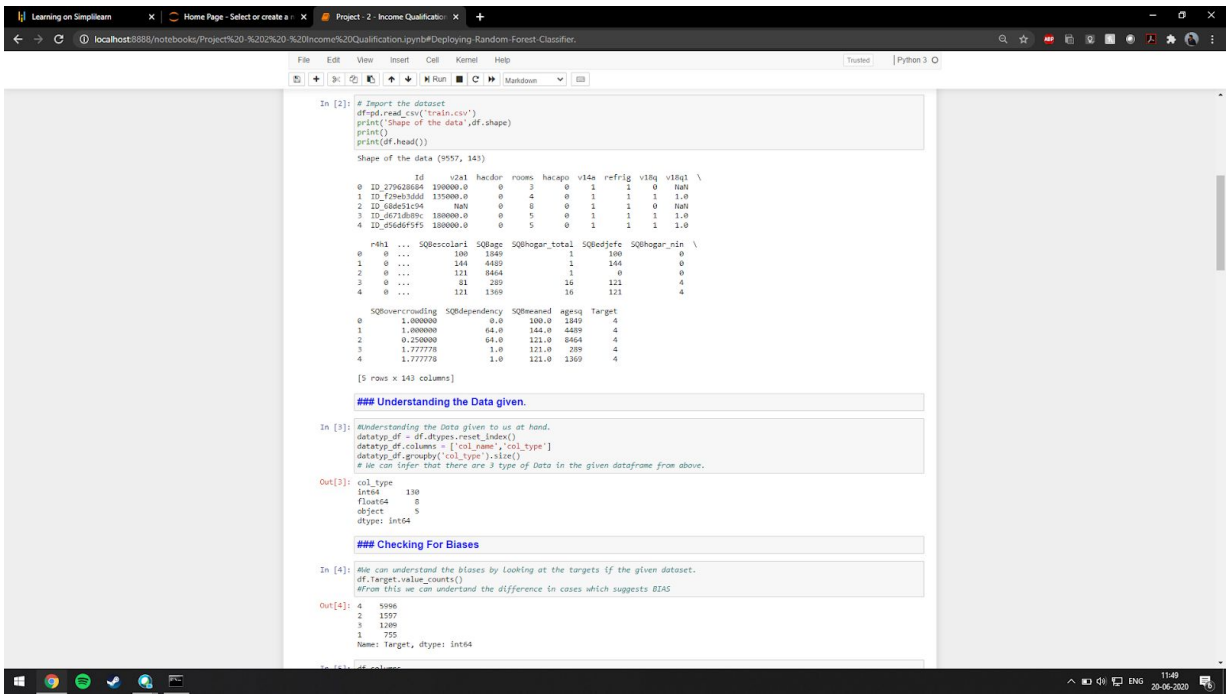
Solution:

In [1]:
```
# Importing
import os
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
import collections
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import KFold,cross_val_score,cross_validate
```

In [2]:
```
# Import the dataset
df=pd.read_csv('train.csv')
print('Shape of the data',df.shape)
print()
print(df.head())
```

```
Shape of the data (9557, 143)

        Id     v2a1  hacdor  rooms  hacapo  v14a  refrig  v18q  v18q1  \
0  ID_279628684  190000.0       0      3       0     1       1     0    NaN
1  ID_f29eb3ddd  135000.0       0      4       0     1       1     1    1.0
```



In [2]:
```
# Import the dataset
df=pd.read_csv('train.csv')
print('Shape of the data',df.shape)
print()
print(df.head())
```

```
Shape of the data (9557, 143)

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

   r4h1  ...  SQBescolari  SQBage  SQBhogar_total  SQBedjefe  SQBhogar_nin  \
0     0  ...          100    1849               1        100             0
1     0  ...          144    4489               1        144             0
2     0  ...          121    8464               1          0             0
3     0  ...           81     289              16        121             4
4     0  ...          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 143 columns]
```


Understanding the Data given.

In [3]:
```
#Understanding the Data given to us at hand.
datatyp_df = df.dtypes.reset_index()
datatyp_df.columns = ['col_name','col_type']
datatyp_df.groupby('col_type').size()
# We can infer that there are 3 type of Data in the given dataframe from above.
```

Out[3]:
```
col_type
int64      130
float64      8
object       5
dtype: int64
```

Checking For Biases

In [4]:
```
#We can understand the biases by looking at the targets if the given dataset.
df.Target.value_counts()
#From this we can understand the difference in cases which suggests BIAS
```

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

Learning on Simplilearn | Home Page - Select or create a | Project - 2 - Income Qualificatio

localhost:8888/notebooks/Project%20-%202%20-%20Income%20Qualification.ipynb#Deploying-Random-Forest-Classifier.

File Edit View Insert Cell Kernel Help — Trusted — Python 3

Run — Markdown

Checking For Biases

In [4]:
```
#We can understand the biases by looking at the targets if the given dataset.
df.Target.value_counts()
#From this we can undertand the difference in cases which suggests BIAS
```

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

In [5]:
```
df.columns
#From this we can infer that the total columns in this dataset is 143
```

Out[5]:
```
Index(['Id', 'v2a1', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q',
       'v18q1', 'r4h1',
       ...
       'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe', 'SQBhogar_nin',
       'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq', 'Target'],
      dtype='object', length=143)
```

Pre - Processing

In [6]:
```
#Finding all the columns which have null values in them.
#The Goal is to remove them because sklearn does not accept null values to train any dataset.
null_columns=df.columns[df.isnull().any()]
df[null_columns].isnull().sum()
```

Out[6]:
```
v2a1            6860
v18q1           7342
rez_esc         7928
meaneduc           5
SQBmeaned          5
dtype: int64
```

In [7]:
```
print ('Percentage of null values in v2a1 : ', df['v2a1'].isnull().sum()/df.shape[0]*100)
print ('Percentage of null values in v18q1 : ', df['v18q1'].isnull().sum()/df.shape[0]*100)
print ('Percentage of null values in rez_esc : ', df['rez_esc'].isnull().sum()/df.shape[0]*100)
print ('Percentage of null values in meaneduc : ', df['meaneduc'].isnull().sum()/df.shape[0]*100)
print ('Percentage of null values in SQBmeaned : ', df['SQBmeaned'].isnull().sum()/df.shape[0]*100)
```

```
Percentage of null values in v2a1 :  71.7798472323951
Percentage of null values in v18q1 :  76.82327090091033
Percentage of null values in rez_esc :  82.9549021659516?
Percentage of null values in meaneduc :  0.05231767290990897
Percentage of null values in SQBmeaned :  0.05231767290990897
```

In [8]:
```
#Removing the Columns which have more than 50% null values
#We can say that these are not temportant to the data.
df= df.drop(['v2a1','v18q1','rez_esc'],axis=1)
print(df.shape)
#Notice the shape has become 140 from 143
#This is because we removed the columnns with more than 50% Null values
```

```
(9557, 140)
```

Learning on Simplilearn | Home Page - Select or create a | Project - 2 - Income Qualificatio

localhost:8888/notebooks/Project%20-%202%20-%20Income%20Qualification.ipynb#Deploying-Random-Forest-Classifier.

File Edit View Insert Cell Kernel Help — Trusted — Python 3

Run — Markdown

In [9]:
```
#Imputing the meaneduc & SQBmeaned columns
#We are imputing the data because as already mentioned we have to change the null values  to zero.
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(df[['meaneduc','SQBmeaned']])
df[['meaneduc','SQBmeaned']]=imp.transform(df[['meaneduc','SQBmeaned']])
df[['meaneduc','SQBmeaned']].isnull().sum()
```

Out[9]:
```
meaneduc     0
SQBmeaned    0
dtype: int64
```

In [10]:
```
df= df.drop(['Id'],axis=1)
df.describe(include='O')
```

Out[10]:

	idhogar	dependency	edjefe	edjefa
count	9557	9557	9557	9557
unique	2988	31	22	22
top	68a60014	yes	no	no
freq	13	2192	3762	6230

In [11]:
```
df.dependency = df.dependency.replace(to_replace=['yes','no'],value=[0.5,0]).astype('float')
median1=np.median(df.edjefe[df.edjefe.isin(['yes','no'])==False].astype('float'))
df.edjefe = df.edjefe.replace(to_replace=['yes','no'],value=[median1,0]).astype('float')
median2 = np.median(df.edjefa[df.edjefa.isin(['yes','no'])==False].astype('float'))
df.edjefa = df.edjefa.replace(to_replace=['yes','no'],value=[median2,0]).astype('float')
```

In [12]:
```
df.describe(include='O')
```

Out[12]:

	idhogar
count	9557
unique	2988
top	68a60014
freq	13

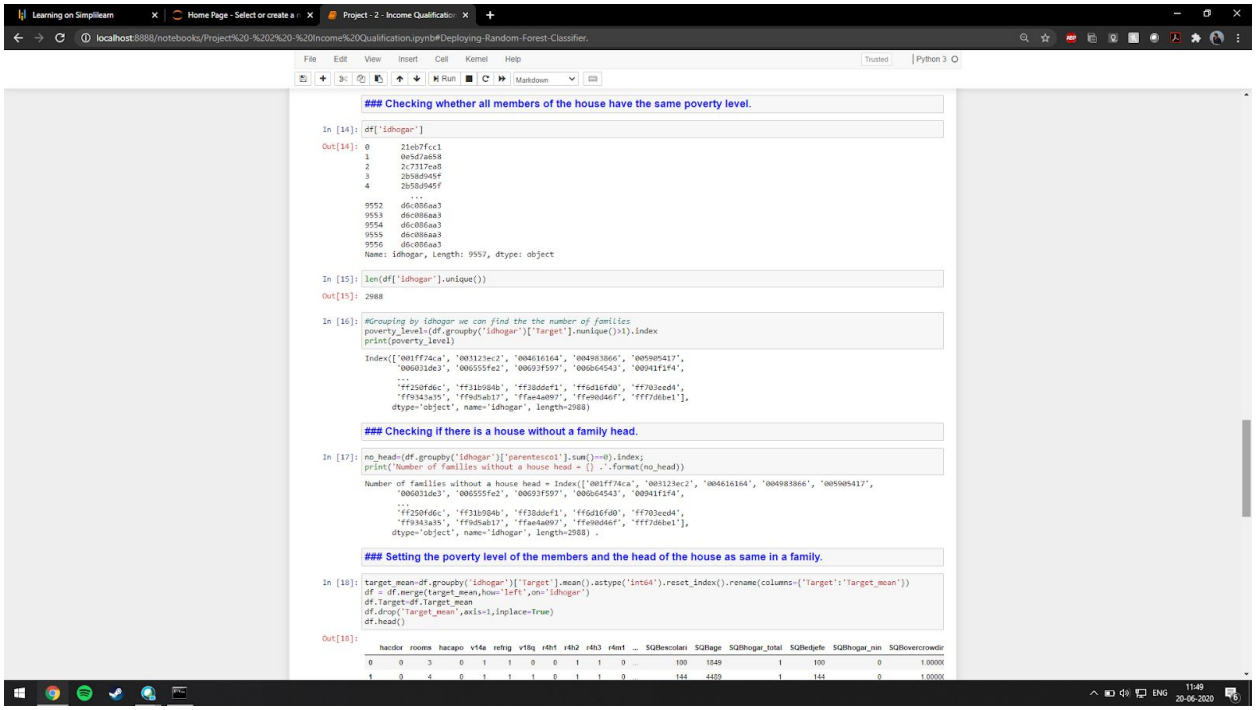
In [13]:
```
print(df.idhogar.nunique()) #Returns the number of unique values in the colum idhogar
```

```
2988
```

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

In [14]:
```
df['idhogar']
```

Out[14]:
```
0       21eb7fcc1
1       0e5d7a658
2       2c7317ea8
3       2b58d945f
4       2b58d945f
          ...
9552    d6c086aa3
9553    d6c086aa3
9554    d6c086aa3
```



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

```
In [14]: df['idhogar']
```

```
Out[14]: 0       21eb7fcc1
         1       0e5d7a658
         2       2c7317ea8
         3       2b58d945f
         4       2b58d945f
                   ...
         9552    d6c086aa3
         9553    d6c086aa3
         9554    d6c086aa3
         9555    d6c086aa3
         9556    d6c086aa3
         Name: idhogar, Length: 9557, dtype: object
```

```
In [15]: len(df['idhogar'].unique())
```

```
Out[15]: 2988
```

```
In [16]: #Grouping by idhogar we can find the the number of families
         poverty_level=(df.groupby('idhogar')['Target'].nunique()>1).index
         print(poverty_level)
```

```
Index(['001ff74ca', '003123ec2', '004616164', '004983866', '005905417',
       '006031de3', '006555fe2', '00693f597', '006b64543', '00941f1f4',
       ...
       'ff250fd6c', 'ff31b984b', 'ff38ddef1', 'ff6d16fd0', 'ff703ecd4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be1'],
      dtype='object', name='idhogar', length=2988)
```

Checking if there is a house without a family head.

```
In [17]: no_head=(df.groupby('idhogar')['parentesco1'].sum()==0).index;
         print('Number of families without a house head = {} .'.format(no_head))
```

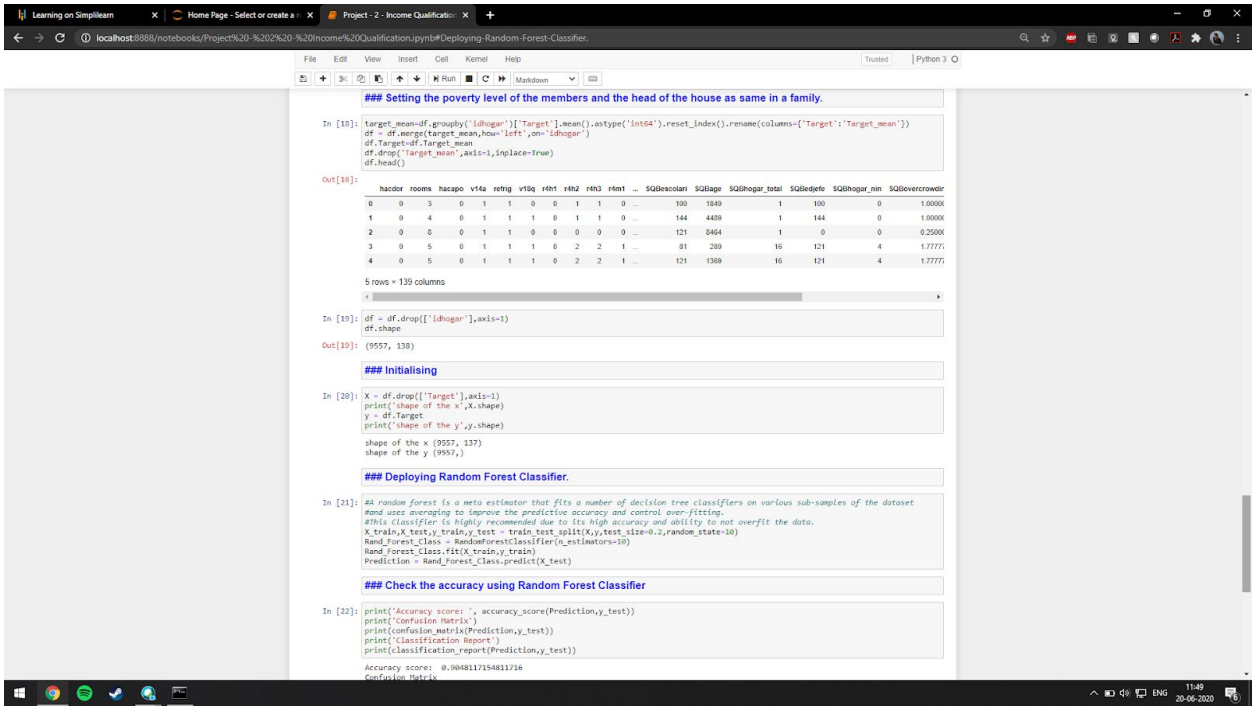
```
Number of families without a house head = Index(['001ff74ca', '003123ec2', '004616164', '004983866', '005905417',
       '006031de3', '006555fe2', '00693f597', '006b64543', '00941f1f4',
       ...
       'ff250fd6c', 'ff31b984b', 'ff38ddef1', 'ff6d16fd0', 'ff703ecd4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be1'],
      dtype='object', name='idhogar', length=2988) .
```

Setting the poverty level of the members and the head of the house as same in a family.

```
In [18]: target_mean=df.groupby('idhogar')['Target'].mean().astype('int64').reset_index().rename(columns={'Target':'Target_mean'})
         df = df.merge(target_mean,how='left',on='idhogar')
         df.Target=df.Target_mean
         df.drop('Target_mean',axis=1,inplace=True)
         df.head()
```

```
Out[18]:
```

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nin	SQBovercrowdir
0	0	3	0	1	1	0	0	1	1	0	...	100	1849	1	100	0	1.0000(
1	0	4	0	1	1	1	0	1	1	0	...	144	4489	1	144	0	1.0000(



Setting the poverty level of the members and the head of the house as same in a family.

```
In [18]: target_mean=df.groupby('idhogar')['Target'].mean().astype('int64').reset_index().rename(columns={'Target':'Target_mean'})
         df = df.merge(target_mean,how='left',on='idhogar')
         df.Target=df.Target_mean
         df.drop('Target_mean',axis=1,inplace=True)
         df.head()
```

```
Out[18]:
```

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nin	SQBovercrowdir
0	0	3	0	1	1	0	0	1	1	0	...	100	1849	1	100	0	1.0000(
1	0	4	0	1	1	1	0	1	1	0	...	144	4489	1	144	0	1.0000(
2	0	8	0	1	1	0	0	0	0	0	...	121	8464	1	0	0	0.2500(
3	0	5	0	1	1	1	0	2	2	1	...	81	289	16	121	4	1.7777(
4	0	5	0	1	1	1	0	2	2	1	...	121	1369	16	121	4	1.7777(

5 rows × 139 columns

```
In [19]: df = df.drop(['idhogar'],axis=1)
         df.shape
```

```
Out[19]: (9557, 138)
```

Initialising

```
In [20]: X = df.drop(['Target'],axis=1)
         print('shape of the x',X.shape)
         y = df.Target
         print('shape of the y',y.shape)
```

```
shape of the x (9557, 137)
shape of the y (9557,)
```

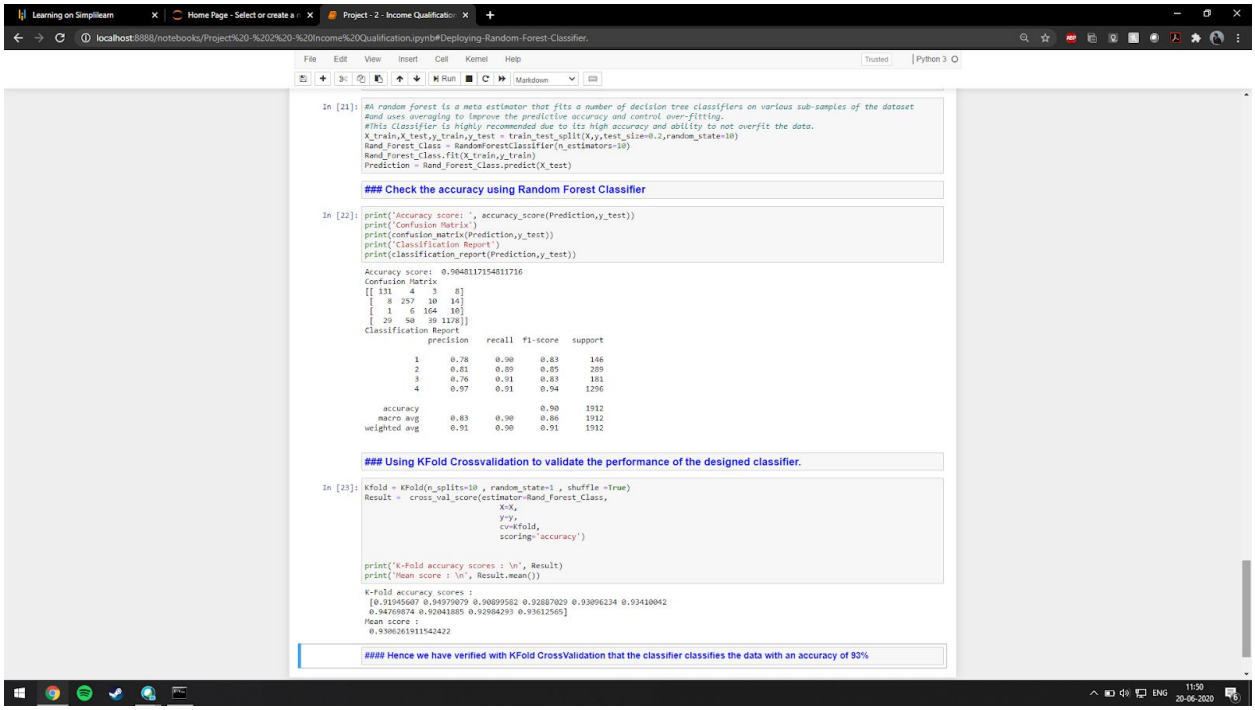
Deploying Random Forest Classifier.

```
In [21]: #A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset
         #and uses averaging to improve the predictive accuracy and control over-fitting.
         #This Classifier is highly recommended due to its high accuracy and ability to not overfit the data.
         X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=10)
         Rand_Forest_Class = RandomForestClassifier(n_estimators=10)
         Rand_Forest_Class.fit(X_train,y_train)
         Prediction = Rand_Forest_Class.predict(X_test)
```

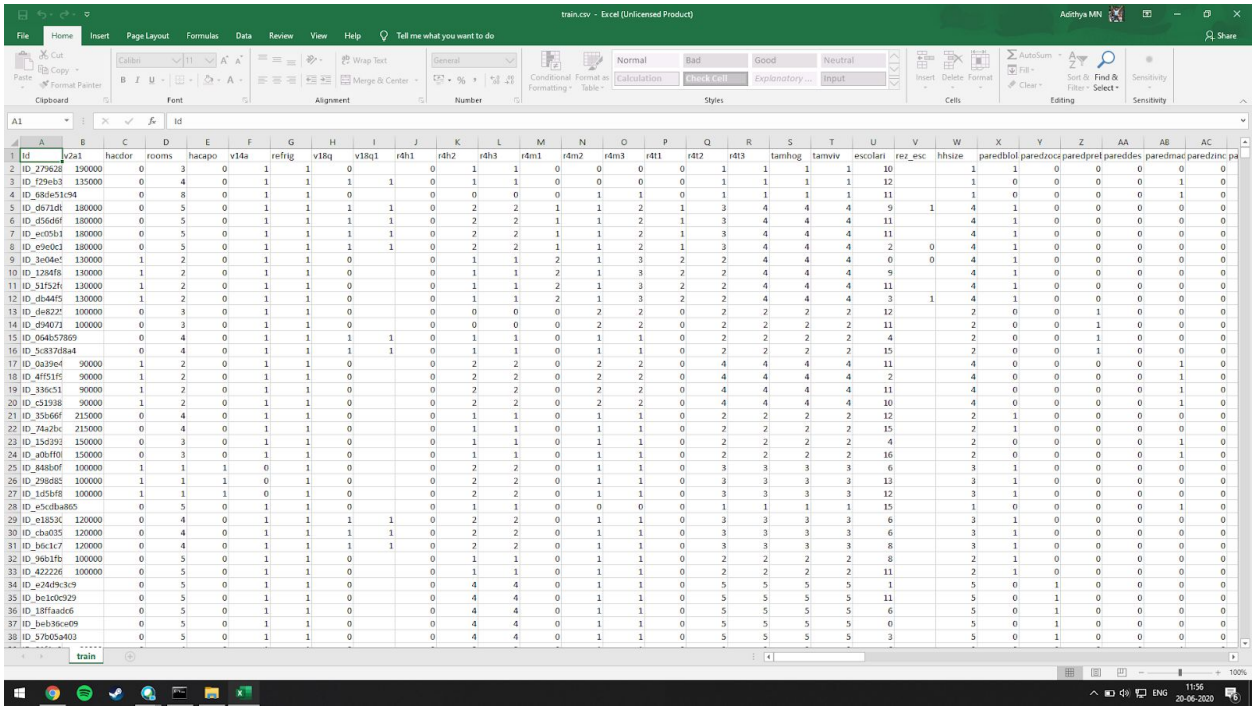
Check the accuracy using Random Forest Classifier

```
In [22]: print('Accuracy score: ', accuracy_score(Prediction,y_test))
         print('Confusion Matrix')
         print(confusion_matrix(Prediction,y_test))
         print('Classification Report')
         print(classification_report(Prediction,y_test))
```

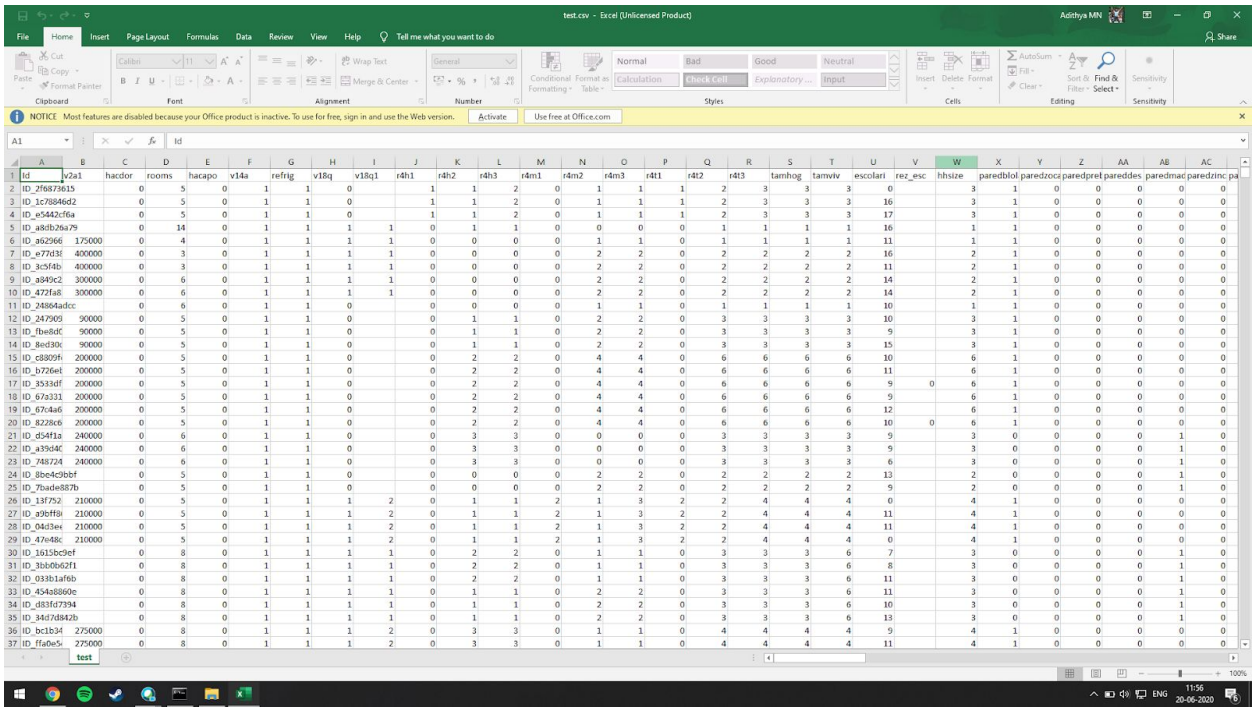
```
Accuracy score:  0.9048117154811716
Confusion Matrix
```


Learning on Simplilearn
Home Page - Select or create a
Project - 2 - Income Qualificatio
localhost:8888/notebooks/Project%20-%202%20-%20Income%20Qualification.ipynb#Deploying-Random-Forest-Classifier.
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Markdown
In [21]:
#A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset
#and uses averaging to improve the predictive accuracy and control over-fitting.
#This Classifier is highly recommended due to its high accuracy and ability to not overfit the data.
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=10)
Rand_Forest_Class = RandomForestClassifier(n_estimators=10)
Rand_Forest_Class.fit(X_train,y_train)
Prediction = Rand_Forest_Class.predict(X_test)
Check the accuracy using Random Forest Classifier
In [22]:
print('Accuracy score: ', accuracy_score(Prediction,y_test))
print('Confusion Matrix')
print(confusion_matrix(Prediction,y_test))
print('Classification Report')
print(classification_report(Prediction,y_test))
Accuracy score: 0.9048117154811716
Confusion Matrix
[[131 4 3 8]
[8 257 10 14]
[1 6 164 10]
[29 50 39 1178]]
Classification Report
precision recall f1-score support
1 0.78 0.90 0.83 146
2 0.81 0.89 0.85 289
3 0.76 0.91 0.83 181
4 0.97 0.91 0.94 1296
accuracy 0.90 1912
macro avg 0.83 0.90 0.86 1912
weighted avg 0.91 0.90 0.91 1912
Using KFold Crossvalidation to validate the performance of the designed classifier.
In [23]:
Kfold = KFold(n_splits=10 , random_state=1 , shuffle =True)
Result = cross_val_score(estimator=Rand_Forest_Class,
X=X,
y=y,
cv=Kfold,
scoring='accuracy')
print('K-Fold accuracy scores : \n', Result)
print('Mean score : \n', Result.mean())
K-Fold accuracy scores :
[0.91945607 0.94970079 0.90899582 0.92887029 0.93096234 0.93410042
0.94769874 0.92041885 0.92984293 0.93612565]
Mean score :
0.9300261911542422
Hence we have verified with KFold CrossValidation that the classifier classifies the data with an accuracy of 93%
ENG
11:50
20-06-2020

TRAIN AND TEST DATASET FILES



Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t1	r4t2	r4t3	tamhog	tamviv	escolari	rez_esc	hhsize	paredblol	paredzoca	paredpreb	pareddes	paredmad	paredzinc	pa
ID_279628	190000	0	3	0	1	1	0		0	1	1	0	0	0	0	1	1	1	1	10		1	1	0	0	0	0	0	
ID_f29eb3	135000	0	4	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	12		1	0	0	0	0	1	0	
ID_68de51c94		0	8	0	1	1	0		0	0	0	0	1	1	0	1	1	1	1	11		1	0	0	0	0	0	0	
ID_d671db	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	9	1	4	1	0	0	0	0	0	
ID_d56d6f	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	11		4	1	0	0	0	0	0	
ID_ec05b1	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	11		4	1	0	0	0	0	0	
ID_e9e0c1	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	2	0	4	1	0	0	0	0	0	
ID_3e04e5	130000	1	2	0	1	1	0		0	1	1	2	1	3	2	2	4	4	4	6	0	4	1	0	0	0	0	0	
ID_128df8	130000	1	2	0	1	1	0		0	1	1	2	1	3	2	2	4	4	4	9		4	1	0	0	0	0	0	
ID_51f52f6	130000	1	2	0	1	1	0		0	1	1	2	1	3	2	2	4	4	4	11		4	1	0	0	0	0	0	
ID_db44f5	130000	1	2	0	1	1	0		0	1	1	2	1	3	2	2	4	4	4	3	1	4	1	0	0	0	0	0	
ID_de822f	100000	0	3	0	1	1	0		0	0	0	0	2	2	0	2	2	2	2	12		2	0	0	1	0	0	0	
ID_d94071	100000	0	3	0	1	1	0		0	0	0	0	2	2	0	2	2	2	2	11		2	0	0	1	0	0	0	
ID_064b57869		0	4	0	1	1	1	1	0	1	1	0	1	1	0	2	2	2	2	4		2	0	0	1	0	0	0	
ID_5c837d8a4		0	4	0	1	1	1	1	0	1	1	0	1	1	0	2	2	2	2	15		2	0	0	1	0	0	0	
ID_0a39e4	90000	1	2	0	1	1	0		0	2	2	0	2	2	0	4	4	4	4	11		4	0	0	0	0	1	0	
ID_4ff51f5	90000	1	2	0	1	1	0		0	2	2	0	2	2	0	4	4	4	4	2		4	0	0	0	0	1	0	
ID_336c51	90000	1	2	0	1	1	0		0	2	2	0	2	2	0	4	4	4	4	11		4	0	0	0	0	1	0	
ID_c51938	90000	1	2	0	1	1	0		0	2	2	0	2	2	0	4	4	4	4	10		4	0	0	0	0	1	0	
ID_35b66f	215000	0	4	0	1	1	0		0	1	1	0	1	1	0	2	2	2	2	12		2	1	0	0	0	0	0	
ID_74a2bc	215000	0	4	0	1	1	0		0	1	1	0	1	1	0	2	2	2	2	15		2	1	0	0	0	0	0	
ID_15d393	150000	0	3	0	1	1	0		0	1	1	0	1	1	0	2	2	2	2	4		2	0	0	0	0	1	0	
ID_a0bff0f	150000	0	3	0	1	1	0		0	1	1	0	1	1	0	2	2	2	2	16		2	0	0	0	0	1	0	
ID_848b0f	100000	1	1	1	0	1	0		0	2	2	0	1	1	0	3	3	3	3	6		3	1	0	0	0	0	0	
ID_298d85	100000	1	1	1	0	1	0		0	2	2	0	1	1	0	3	3	3	3	13		3	1	0	0	0	0	0	
ID_1d5bf8	100000	1	1	1	0	1	0		0	2	2	0	1	1	0	3	3	3	3	12		3	1	0	0	0	0	0	
ID_e5cdba865		0	5	0	1	1	0		0	1	1	0	0	0	0	1	1	1	1	15		1	0	0	0	0	1	0	
ID_e1853(	120000	0	4	0	1	1	1	1	0	2	2	0	1	1	0	3	3	3	3	6		3	1	0	0	0	0	0	
ID_cba035	120000	0	4	0	1	1	1	1	0	2	2	0	1	1	0	3	3	3	3	6		3	1	0	0	0	0	0	
ID_b6c1c7	120000	0	4	0	1	1	1	1	0	2	2	0	1	1	0	3	3	3	3	8		3	1	0	0	0	0	0	
ID_96b1fb	100000	0	5	0	1	1	0		0	1	1	0	1	1	0	2	2	2	2	8		2	1	0	0	0	0	0	
ID_422226	100000	0	5	0	1	1	0		0	1	1	0	1	1	0	2	2	2	2	11		2	1	0	0	0	0	0	
ID_e24d9c3c9		0	5	0	1	1	0		0	4	4	0	1	1	0	5	5	5	5	1		5	0	1	0	0	0	0	
ID_be1c0c929		0	5	0	1	1	0		0	4	4	0	1	1	0	5	5	5	5	11		5	0	1	0	0	0	0	
ID_18ffaadc6		0	5	0	1	1	0		0	4	4	0	1	1	0	5	5	5	5	6		5	0	1	0	0	0	0	
ID_be3c4e09		0	5	0	1	1	0		0	4	4	0	1	1	0	5	5	5	5	0		5	0	1	0	0	0	0	
ID_57b05a903		0	5	0	1	1	0		0	4	4	0	1	1	0	5	5	5	5	3		5	0	1	0	0	0	0	



Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t1	r4t2	r4t3	tamhog	tamviv	escolari	rez_esc	hhsize	paredblol	paredzoca	paredpreb	pareddes	paredmad	paredzinc	pa
ID_2f6873615		0	5	0	1	1	0		1	1	2	0	1	1	1	2	3	3	3	0		3	1	0	0	0	0	0	
ID_1c78846d2		0	5	0	1	1	0		1	1	2	0	1	1	1	2	3	3	3	16		3	1	0	0	0	0	0	
ID_e5442cf6a		0	5	0	1	1	0		1	1	2	0	1	1	1	2	3	3	3	17		3	1	0	0	0	0	0	
ID_a8db26a79		0	14	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	16		1	1	0	0	0	0	0	
ID_a62966	175000	0	4	0	1	1	1	1	0	0	0	0	1	1	0	1	1	1	1	11		1	1	0	0	0	0	0	
ID_e77d3(	400000	0	3	0	1	1	1	1	0	0	0	0	2	2	0	2	2	2	2	16		2	1	0	0	0	0	0	
ID_3c5f4b	400000	0	3	0	1	1	1	1	0	0	0	0	2	2	0	2	2	2	2	11		2	1	0	0	0	0	0	
ID_a949c2	300000	0	6	0	1	1	1	1	0	0	0	0	2	2	0	2	2	2	2	14		2	1	0	0	0	0	0	
ID_472fa8	300000	0	6	0	1	1	1	1	0	0	0	0	2	2	0	2	2	2	2	14		2	1	0	0	0	0	0	
ID_24864adcc		0	6	0	1	1	0		0	0	0	0	1	1	0	1	1	1	1	10		1	1	0	0	0	0	0	
ID_247909	90000	0	5	0	1	1	0		0	1	1	0	2	2	0	3	3	3	3	10		3	1	0	0	0	0	0	
ID_fbe8d(	90000	0	5	0	1	1	0		0	1	1	0	2	2	0	3	3	3	3	9		3	1	0	0	0	0	0	
ID_8ed30(	90000	0	5	0	1	1	0		0	1	1	0	2	2	0	3	3	3	3	15		3	1	0	0	0	0	0	
ID_c8809f(	200000	0	5	0	1	1	0		0	2	2	0	4	4	0	6	6	6	6	10		6	1	0	0	0	0	0	
ID_b726e8	200000	0	5	0	1	1	0		0	2	2	0	4	4	0	6	6	6	6	11		6	1	0	0	0	0	0	
ID_3533df	200000	0	5	0	1	1	0		0	2	2	0	4	4	0	6	6	6	6	9	0	6	1	0	0	0	0	0	
ID_67a331	200000	0	5	0	1	1	0		0	2	2	0	4	4	0	6	6	6	6	9		6	1	0	0	0	0	0	
ID_67c4a6	200000	0	5	0	1	1	0		0	2	2	0	4	4	0	6	6	6	6	12		6	1	0	0	0	0	0	
ID_8228c6	200000	0	5	0	1	1	0		0	2	2	0	4	4	0	6	6	6	6	10	0	6	1	0	0	0	0	0	
ID_d54f1a	240000	0	6	0	1	1	0		0	3	3	0	0	0	0	3	3	3	3	9		3	0	0	0	0	0	1	
ID_a39d4(	240000	0	6	0	1	1	0		0	3	3	0	0	0	0	3	3	3	3	9		3	0	0	0	0	0	1	
ID_74872(	240000	0	6	0	1	1	0		0	3	3	0	0	0	0	3	3	3	3	6		3	0	0	0	0	0	1	
ID_8be4c9bbf		0	5	0	1	1	0		0	0	0	0	2	2	0	2	2	2	2	13		2	0	0	0	0	0	0	
ID_7bade887b		0	5	0	1	1	0		0	0	0	0	2	2	0	2	2	2	2	9		2	0	0	0	0	0	1	
ID_13f752(	210000	0	5	0	1	1	1	2	0	1	1	2	1	3	2	2	4	4	4	0		4	1	0	0	0	0	0	
ID_a9bff8(	210000	0	5	0	1	1	1	2	0	1	1	2	1	3	2	2	4	4	4	11		4	1	0	0	0	0	0	
ID_04d3e(	210000	0	5	0	1	1	1	2	0	1	1	2	1	3	2	2	4	4	4	11		4	1	0	0	0	0	0	
ID_47e48(	210000	0	5	0	1	1	1	2	0	1	1	2	1	3	2	2	4	4	4	0		4	1	0	0	0	0	0	
ID_1615bc9ef		0	8	0	1	1	1	1	0	2	2	0	1	1	0	3	3	3	3	6	7	3	0	0	0	0	0	1	
ID_3bb0b62f1		0	8	0	1	1	1	1	0	2	2	0	1	1	0	3	3	3	3	6	8	3	0	0	0	0	0	1	
ID_033b1af6b		0	8	0	1	1	1	1	0	2	2	0	1	1	0	3	3	3	3	6	11	3	0	0	0	0	0	1	
ID_454a8860e		0	8	0	1	1	1	1	0	1	1	0	2	2	0	3	3	3	3	6	11	3	0	0	0	0	0	1	
ID_d83fd7394		0	8	0	1	1	1	1	0	1	1	0	2	2	0	3	3	3	3	6	10	3	0	0	0	0	0	1	
ID_34d7d842b		0	8	0	1	1	1	1	0	1	1	0	2	2	0	3	3	3	3	6	13	3	0	0	0	0	0	1	
ID_bc1b34	275000	0	8	0	1	1	1	2	0	3	3	0	1	1	0	4	4	4	4	9		4	1	0	0	0	0	0	
ID_ffa0e5(	275000	0	8	0	1	1	1	2	0	3	3	0	1	1	0	4	4	4	4	11		4	1	0	0	0	0	0	