```
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


df=pd.read_csv('train.csv')


print('Shape of the data',df.shape)
```

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

```
print(df.head())
```

```
                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]
```

```
data_train_info = pd.DataFrame(columns=['Name of Col', 'Num of Null', 'Dtype', 'N_Unique' , 'Null Perc'])

for i in range(0, len(df.columns)):
    data_train_info.loc[i] = [df.columns[i],
                        df[df.columns[i]].isnull().sum(),
                        df[df.columns[i]].dtypes,
                        df[df.columns[i]].nunique(),
                        df[df.columns[i]].isnull().sum()*100/df.shape[0]]
data_train_info
```

	Name of Col	Num of Null	Dtype	N_Unique	Null Perc
0	Id	0	object	9557	0.000000
1	v2a1	6860	float64	157	71.779847
2	hacdor	0	int64	2	0.000000
3	rooms	0	int64	11	0.000000
4	hacapo	0	int64	2	0.000000
...	...	...	...	...	...
138	SQBovercrowding	0	float64	38	0.000000
139	SQBdependency	0	float64	31	0.000000
140	SQBmeaned	5	float64	155	0.052318

```
data_train_info[data_train_info['Num of Null']>0]
```

	Name of Col	Num of Null	Dtype	N_Unique	Null Perc
1	v2a1	6860	float64	157	71.779847
8	v18q1	7342	float64	6	76.823271
21	rez_esc	7928	float64	6	82.954902
103	meaneduc	5	float64	155	0.052318
140	SQBmeaned	5	float64	155	0.052318

```
#Percentage of null values in v2a1, v18q1, rez_esc is more than 50%. So, these columns are dropped
df= df.drop(['v2a1','v18q1','rez_esc'],axis=1)
print(df.shape)
```

```
(9557, 140)
```

```
#Imputing the meaneduc & SQBmeaned coumns
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()
```

```
meaneduc     0
SQBmeaned    0
dtype: int64
```

```
#Find Column with mixed values
df= df.drop(['Id'],axis=1)
df.describe(include='O')
```

	idhogar	dependency	edjefe	edjefa
count	9557	9557	9557	9557
unique	2988	31	22	22

```
# Dependency replace yes with 0.5 and no with 0
df.dependency = df.dependency.replace(to_replace=['yes','no'],value=[0.5,0]).astype('float')
```

```
# edjefe replace yes with median and no with zero
med_1=np.median(df.edjefe[df.edjefe.isin(['yes','no'])==False].astype('float'))
df.edjefe= df.edjefe.replace(to_replace=['yes','no'],value=[med_1,0]).astype('float')


# edjefa replace yes with median and no with zero
med_2=np.median(df.edjefa[df.edjefa.isin(['yes','no'])==False].astype('float'))
df.edjefa= df.edjefa.replace(to_replace=['yes','no'],value=[med_2,0]).astype('float')


df.describe(include='O')
```

	idhogar
count	9557
unique	2988
top	fd8a6d014
freq	13

```
print(df.idhogar.nunique())
```

```
2988
```

3. Finding biasness in the dataset

```
df.Target.value_counts()
import collections
print(df.shape)
collections.Counter(df['Target'])
```

```
(9557, 139)
Counter({4: 5996, 2: 1597, 3: 1209, 1: 755})
```

4. Checking whether all members of the house have the same poverty level. **bold text**

```
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', 'ff703eed4',
       'ff9343a35', 'ff9d5ab17', 'ffae4a097', 'ffe90d46f', 'fff7d6be1'],
      dtype='object', name='idhogar', length=2988)
```

5. Checking if there is a house without a family head.

```
no_head=(df.groupby('idhogar')['parentesco1'].sum()==0).index
display(no_head)
```

```
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)
```

6. Set poverty level of the members and the head of the house same in a family.

```
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()
```

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescolari	SQBage	SQBhogar
0	0	3	0	1	1	0	0	1	1	0	...	100	1849	
1	0	4	0	1	1	1	0	1	1	0	...	144	4489	
2	0	8	0	1	1	0	0	0	0	0	...	121	8464	
3	0	5	0	1	1	1	0	2	2	1	...	81	289	
4	0	5	0	1	1	1	0	2	2	1	...	121	1369	

5 rows × 139 columns

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

```
(9557, 138)
```

```
# Assigning the value for x & y
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

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=10)
rfc = RandomForestClassifier(criterion= 'gini',n_estimators=100)
rfc.fit(x_train,y_train)
pred=rfc.predict(x_test)
```

```
# Check the accuracy using random forest

print('Accuracy score: ', accuracy_score(pred,y_test))
print()
print('Confusion matrix:\n', confusion_matrix(pred,y_test))
print()
print('Classification report:\n', classification_report(pred,y_test))
```

```
Accuracy score:  0.9325313807531381

Confusion matrix:
 [[ 141    3    1    3]
 [   3  267    3    5]
 [   0    1  175    2]
 [  25   46   37 1200]]

Classification report:
               precision    recall  f1-score   support

           1       0.83      0.95      0.89       148
           2       0.84      0.96      0.90       278
           3       0.81      0.98      0.89       178
           4       0.99      0.92      0.95      1308

    accuracy                           0.93      1912
   macro avg       0.87      0.95      0.91      1912
weighted avg       0.94      0.93      0.93      1912
```

Check the accuracy using random forest with cross validation.

```
from sklearn.model_selection import KFold,cross_val_score
```

```
seed=7
kfold=KFold(n_splits=5,random_state=seed,shuffle=True)

rmclassifier=RandomForestClassifier(random_state=10)
#print(cross_val_score(rmclassifier,x_train,y_train,cv=kfold,scoring='accuracy'))
```

```
results=cross_val_score(rmclassifier,x_train,y_train,cv=kfold,scoring='accuracy')
print(results)
print(results.mean()*100)
```

```
[0.91563113 0.89535644 0.8940484  0.90843689 0.91628515]
90.595160235448
```

Predict for Test Data

```
df_test =pd.read_csv('test.csv')
```

```
#Percentage of null values in v2a1, v18q1, rez_esc is more than 50%. So, these columns are dropped
df_test= df_test.drop(['v2a1','v18q1','rez_esc'],axis=1)
print(df_test.shape)
```

```
(23856, 139)
```

```
#Imputing the meaneduc & SQBmeaned coumns
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(df_test[['meaneduc','SQBmeaned']])
df_test[['meaneduc','SQBmeaned']]=imp.transform(df_test[['meaneduc','SQBmeaned']])
df_test[['meaneduc','SQBmeaned']].isnull().sum()
```

```
meaneduc     0
SQBmeaned    0
dtype: int64
```

```
df_test.describe(include='O')
```

	Id	idhogar	dependency	edjefe	edjefa
count	23856	23856	23856	23856	23856
unique	23856	7352	35	22	22
top	ID_2f6873615	8e9159699	yes	no	no
freq	1	13	5388	9056	15845

```
# Dependency replace yes with 0.5 and no with 0
df_test.dependency = df_test.dependency.replace(to_replace=['yes','no'],value=[0.5,0]).astype('float')

# edjefe replace yes with median and no with zero
med_1=np.median(df_test.edjefe[df_test.edjefe.isin(['yes','no'])==False].astype('float'))
df_test.edjefe= df_test.edjefe.replace(to_replace=['yes','no'],value=[med_1,0]).astype('float')

# edjefa replace yes with median and no with zero
med_2=np.median(df_test.edjefa[df_test.edjefa.isin(['yes','no'])==False].astype('float'))
df_test.edjefa= df_test.edjefa.replace(to_replace=['yes','no'],value=[med_2,0]).astype('float')
```

```
df_test= df_test.drop(['idhogar'],axis=1)
df_test.shape
```

```
(23856, 138)
```

```
df_test= df_test.drop(['Id'],axis=1)
df_test.shape
```

```
(23856, 137)
```

```
y_predict_testdata = rfc.predict(df_test)
y_predict_testdata
```

```
array([4, 4, 4, ..., 4, 4, 4])
```

✓ 0s completed at 10:09 AM