In [188]:

```
#importing the required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [189]:

```
#reading our dataset

df = pd.read_csv("train_data(Income-qualification(MLProject)).csv")
df
```

Out[189]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	100	1849	1	100	
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	144	4489	1	144	
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	...	121	8464	1	0	
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	...	81	289	16	121	
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	...	121	1369	16	121	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9552	ID_d45ae367d	80000.0	0	6	0	1	1	0	NaN	0	...	81	2116	25	81	
9553	ID_c94744e07	80000.0	0	6	0	1	1	0	NaN	0	...	0	4	25	81	
9554	ID_85fc658f8	80000.0	0	6	0	1	1	0	NaN	0	...	25	2500	25	81	
9555	ID_ced540c61	80000.0	0	6	0	1	1	0	NaN	0	...	121	676	25	81	
9556	ID_a38c64491	80000.0	0	6	0	1	1	0	NaN	0	...	64	441	25	81	

9557 rows × 143 columns

In [190]:

```
#Checking for null values

df.isna().sum().where(df.isna().sum()>0).dropna()
```

Out[190]:

```
v2a1        6860.0
v18q1       7342.0
rez_esc     7928.0
meaneduc       5.0
SQBmeaned      5.0
dtype: float64
```

In [191]:

```
#As, we can see that columns=['v2a1' , 'v18q1' , 'rez_esc'] conttains mostly NAN values , so we should get rid of these columns.

df = df.drop(columns=['v2a1' , 'v18q1' , 'rez_esc'] , axis=1)
df
```

Out[191]:

	Id	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	...	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nin
0	ID_279628684	0	3	0	1	1	0	0	1	1	...	100	1849	1	100	0
1	ID_f29eb3ddd	0	4	0	1	1	1	0	1	1	...	144	4489	1	144	0
2	ID_68de51c94	0	8	0	1	1	0	0	0	0	...	121	8464	1	0	0
3	ID_d671db89c	0	5	0	1	1	1	0	2	2	...	81	289	16	121	4
4	ID_d56d6f5f5	0	5	0	1	1	1	0	2	2	...	121	1369	16	121	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9552	ID_d45ae367d	0	6	0	1	1	0	0	2	2	...	81	2116	25	81	1
9553	ID_c94744e07	0	6	0	1	1	0	0	2	2	...	0	4	25	81	1
9554	ID_85fc658f8	0	6	0	1	1	0	0	2	2	...	25	2500	25	81	1
9555	ID_ced540c61	0	6	0	1	1	0	0	2	2	...	121	676	25	81	1
9556	ID_a38c64491	0	6	0	1	1	0	0	2	2	...	64	441	25	81	1

9557 rows × 140 columns

In [192]:

```
#now let's have a look at our remaining null values

df.isna().sum().where(df.isna().sum()>0).dropna()
```

Out[192]:

```
meaneduc      5.0
SQBmeaned     5.0
dtype: float64
```

In [193]:

```
# droping the rows which consist of these NAN values.

df = df.dropna(axis=0)
```

In [194]:

```
#Again , having a look at nul values if any.

df.isna().sum().where(df.isna().sum()>0).dropna()
```

Out[194]:

```
Series([], dtype: float64)
```

In [196]:

```
#now let's look at our df

df
#Pretty Cool.
```

Out[196]:

	Id	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	...	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nin
0	ID_279628684	0	3	0	1	1	0	0	1	1	...	100	1849	1	100	0
1	ID_f29eb3ddd	0	4	0	1	1	1	0	1	1	...	144	4489	1	144	0
2	ID_68de51c94	0	8	0	1	1	0	0	0	0	...	121	8464	1	0	0
3	ID_d671db89c	0	5	0	1	1	1	0	2	2	...	81	289	16	121	4
4	ID_d56d6f5f5	0	5	0	1	1	1	0	2	2	...	121	1369	16	121	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9552	ID_d45ae367d	0	6	0	1	1	0	0	2	2	...	81	2116	25	81	1
9553	ID_c94744e07	0	6	0	1	1	0	0	2	2	...	0	4	25	81	1
9554	ID_85fc658f8	0	6	0	1	1	0	0	2	2	...	25	2500	25	81	1
9555	ID_ced540c61	0	6	0	1	1	0	0	2	2	...	121	676	25	81	1
9556	ID_a38c64491	0	6	0	1	1	0	0	2	2	...	64	441	25	81	1

9552 rows × 140 columns

In [197]:

```
#Checking the description of our dataset for taking a look at the statistical values to get a basic understanding of our dataset

df.describe()
```

Out[197]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescola
count	9552.000000	9552.000000	9552.000000	9552.000000	9552.000000	9552.000000	9552.000000	9552.000000	9552.000000	9552.000000	...	9552.00000
mean	0.038107	4.956554	0.023660	0.994765	0.957601	0.231889	0.386097	1.559359	1.945456	0.399393	...	74.21555
std	0.191465	1.467227	0.151995	0.072164	0.201509	0.422060	0.680899	1.036672	1.188918	0.692581	...	76.78724
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
25%	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.000000	...	16.00000
50%	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	2.000000	0.000000	...	36.00000
75%	0.000000	6.000000	0.000000	1.000000	1.000000	0.000000	1.000000	2.000000	3.000000	1.000000	...	121.00000
max	1.000000	11.000000	1.000000	1.000000	1.000000	1.000000	5.000000	8.000000	8.000000	6.000000	...	441.00000

8 rows × 135 columns

In [198]:

```
#df['Target'] column will be our output varible of this dataset.

df['Target']
```

Out[198]:

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

In [199]:

```
#So, let's see what does our Target column contains.

df['Target'].value_counts()
```

Out[199]:

```
4    5991
2    1597
3    1209
1     755
Name: Target, dtype: int64
```

In [200]:

```
#Ok , so our Target column consist of 4 values = 1,2,3,4.
#These no.s must be representing the poverty levels of each household.
#But how do we get to know , which no. represent the most poor and which means the least poor?
#For that , we will divide our dataset with respect to each Target value and analyse which group reperesents which povrty categ.

g = df.groupby('Target')
```

In [201]:

```
g1 = g.get_group(1)
g2 = g.get_group(2)
g3 = g.get_group(3)
g4 = g.get_group(4)
```

In [202]:

```
g1.describe()
```

Out[202]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescolari	SQBa
count	755.00000	755.000000	755.000000	755.000000	755.000000	755.000000	755.000000	755.000000	755.000000	755.000000	...	755.000000	755.000
mean	0.14702	4.327152	0.075497	0.988079	0.887417	0.079470	0.796026	1.160265	1.956291	0.800000	...	37.505960	1293.280
std	0.35436	1.260601	0.264366	0.108600	0.316292	0.270651	1.041950	0.852215	1.314860	0.914661	...	45.152802	1650.502
min	0.00000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000
25%	0.00000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.000000	...	1.000000	100.000
50%	0.00000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	2.000000	1.000000	...	25.000000	576.000
75%	0.00000	5.000000	0.000000	1.000000	1.000000	0.000000	1.000000	2.000000	3.000000	1.000000	...	49.000000	1936.000
max	1.00000	8.000000	1.000000	1.000000	1.000000	1.000000	5.000000	3.000000	7.000000	3.000000	...	256.000000	8649.000

8 rows × 135 columns

In [203]:

```
g2.describe()
```

Out[203]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescola
count	1597.000000	1597.000000	1597.000000	1597.000000	1597.000000	1597.000000	1597.000000	1597.000000	1597.000000	1597.000000	...	1597.00000
mean	0.067627	4.483406	0.047589	0.986850	0.928616	0.078272	0.560426	1.474640	2.035066	0.628053	...	41.20162
std	0.251183	1.293131	0.212962	0.113951	0.257546	0.268683	0.761470	0.978536	1.233420	0.945595	...	46.40832
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
25%	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.000000	...	4.00000
50%	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	2.000000	0.000000	...	36.00000
75%	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	1.000000	2.000000	3.000000	1.000000	...	49.00000
max	1.000000	9.000000	1.000000	1.000000	1.000000	1.000000	3.000000	7.000000	7.000000	6.000000	...	289.00000

8 rows × 135 columns

In [204]:

```
g3.describe()
```

Out[204]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescola
count	1209.000000	1209.000000	1209.000000	1209.000000	1209.000000	1209.000000	1209.000000	1209.000000	1209.000000	1209.000000	...	1209.00000
mean	0.048801	4.729529	0.030604	0.991729	0.961952	0.118280	0.405294	1.692308	2.097601	0.442514	...	48.54673
std	0.215540	1.263522	0.172313	0.090607	0.191391	0.323073	0.650653	1.255300	1.311085	0.674914	...	50.28817
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
25%	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.000000	...	9.00000
50%	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	2.000000	0.000000	...	36.00000
75%	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	1.000000	2.000000	3.000000	1.000000	...	64.00000
max	1.000000	9.000000	1.000000	1.000000	1.000000	1.000000	4.000000	7.000000	7.000000	3.000000	...	289.00000

8 rows × 135 columns

In [205]:

```
g4.describe()
```

Out[205]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	SQBescola
count	5991.000000	5991.000000	5991.000000	5991.000000	5991.000000	5991.000000	5991.000000	5991.000000	5991.000000	5991.000000	...	5991.00000
mean	0.014355	5.207812	0.009347	0.998331	0.973293	0.314972	0.284093	1.605408	1.889501	0.279252	...	92.82223
std	0.118959	1.510579	0.096237	0.040825	0.161238	0.464544	0.568287	1.010726	1.128834	0.529562	...	84.38831
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000
25%	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.000000	...	36.00000
50%	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	2.000000	0.000000	...	64.00000
75%	0.000000	6.000000	0.000000	1.000000	1.000000	1.000000	0.000000	2.000000	2.000000	0.000000	...	121.00000
max	1.000000	11.000000	1.000000	1.000000	1.000000	1.000000	5.000000	8.000000	8.000000	3.000000	...	441.00000

8 rows × 135 columns

In [206]:

```
#By looking at the stats of eaach group we came to know that:
#g1 --> represents most poor
#g4 --. represents least poor
```

In [208]:

```
#There are in total 135 columns we have in our dataset.
#This is a huge no. of dimensions, so we must filter the imp columns.

column_categories={'col_rooms' : [i for i in df.columns if i.startswith("rooms")],'col_bathroom' : [i for i in df.columns if i.startswith
column_categories
```

Out[208]:

```
{'col_rooms': ['rooms'],
 'col_bathroom': ['v14a'],
 'col_edu': ['escolari'],
 'col_water': ['abastaguadentro', 'abastaguafuera', 'abastaguano'],
 'col_elec': ['public', 'planpri', 'noelec', 'coopele'],
 'col_toilets': ['sanitario1',
  'sanitario2',
  'sanitario3',
  'sanitario5',
  'sanitario6'],
 'col_walls': ['epared1', 'epared2', 'epared3'],
 'cols_roofs': ['etecho1', 'etecho2', 'etecho3'],
 'cols_floor': ['pisomoscer',
  'pisocemento',
  'pisoother',
  'pisonatur',
  'pisonotiene',
  'pisomadera'],
 'col_disable': ['dis'],
 'col_marital_status': ['estadocivil1',
  'estadocivil2',
  'estadocivil3',
  'estadocivil4',
  'estadocivil5',
  'estadocivil6',
  'estadocivil7'],
 'col_familymember': ['parentesco1',
  'parentesco2',
  'parentesco3',
  'parentesco4',
  'parentesco5',
  'parentesco6',
  'parentesco7',
  'parentesco8',
  'parentesco9',
  'parentesco10',
  'parentesco11',
  'parentesco12'],
 'col_houseownership': ['tipovivi1',
  'tipovivi2',
  'tipovivi3',
  'tipovivi4',
  'tipovivi5'],
 'col_computer': ['computer'],
 'col_mobilephone': ['qmobilephone'],
 'col_refrig': ['refrig'],
 'col_tablet': ['v18q']}
```

In [209]:

```
cols=[]
for k,v in column_categories.items():
    cols.append(v)
cols
```

Out[209]:

```
[['rooms'],
 ['v14a'],
 ['escolari'],
 ['abastaguadentro', 'abastaguafuera', 'abastaguano'],
 ['public', 'planpri', 'noelec', 'coopele'],
 ['sanitario1', 'sanitario2', 'sanitario3', 'sanitario5', 'sanitario6'],
 ['epared1', 'epared2', 'epared3'],
 ['etecho1', 'etecho2', 'etecho3'],
 ['pisomoscer',
  'pisocemento',
  'pisoother',
  'pisonatur',
  'pisonotiene',
  'pisomadera'],
 ['dis'],
 ['estadocivil1',
  'estadocivil2',
  'estadocivil3',
  'estadocivil4',
  'estadocivil5',
  'estadocivil6',
  'estadocivil7'],
 ['parentesco1',
  'parentesco2',
  'parentesco3',
  'parentesco4',
  'parentesco5',
  'parentesco6',
  'parentesco7',
  'parentesco8',
  'parentesco9',
  'parentesco10',
  'parentesco11',
  'parentesco12'],
 ['tipovivi1', 'tipovivi2', 'tipovivi3', 'tipovivi4', 'tipovivi5'],
 ['computer'],
 ['qmobilephone'],
 ['refrig'],
 ['v18q']]
```

In [210]:

```
imp_cols=[]
for i in cols:
    for a in i:
        imp_cols.append(a)
imp_cols
```

Out[210]:

```
['rooms',
 'v14a',
 'escolari',
 'abastaguadentro',
 'abastaguafuera',
 'abastaguano',
 'public',
 'planpri',
 'noelec',
 'coopele',
 'sanitario1',
 'sanitario2',
 'sanitario3',
 'sanitario5',
 'sanitario6',
 'epared1',
 'epared2',
 'epared3',
 'etecho1',
 'etecho2',
 'etecho3',
 'pisomoscer',
 'pisocemento',
 'pisoother',
 'pisonatur',
 'pisonotiene',
 'pisomadera',
 'dis',
 'estadocivil1',
 'estadocivil2',
 'estadocivil3',
 'estadocivil4',
 'estadocivil5',
 'estadocivil6',
 'estadocivil7',
 'parentesco1',
 'parentesco2',
 'parentesco3',
 'parentesco4',
 'parentesco5',
 'parentesco6',
 'parentesco7',
 'parentesco8',
 'parentesco9',
 'parentesco10',
 'parentesco11',
 'parentesco12',
 'tipovivi1',
 'tipovivi2',
 'tipovivi3',
 'tipovivi4',
 'tipovivi5',
 'computer',
 'qmobilephone',
 'refrig',
 'v18q']
```

In [211]:

```
other_imp_cols = ['meaneduc', 'cielorazo', 'escolari', 'SQBescolari', 'eviv3',
       'epared3', 'pisomoscer', 'SQBmeaned', 'paredblolad', 'etecho3',
       'SQBedjefe', 'rooms', 'instlevel8', 'qmobilephone', 'computer',
       'lugar1', 'bedrooms' , 'hogar_nin', 'r4t1', 'SQBhogar_nin', 'overcrowding', 'SQBovercrowding',
       'r4m1', 'r4h1', 'eviv1', 'pisocemento', 'epared1']
```

In [212]:

```
for i in other_imp_cols:
    if i  not in imp_cols:
         imp_cols.append(i)
```

In [214]:

```
#This is our final list of important columns on which we will be performing our model training.

imp_cols
```

Out[214]:

```
['rooms',
 'v14a',
 'escolari',
 'abastaguadentro',
 'abastaguafuera',
 'abastaguano',
 'public',
 'planpri',
 'noelec',
 'coopele',
 'sanitario1',
 'sanitario2',
 'sanitario3',
 'sanitario5',
 'sanitario6',
 'epared1',
 'epared2',
 'epared3',
 'etecho1',
 'etecho2',
 'etecho3',
 'pisomoscer',
 'pisocemento',
 'pisoother',
 'pisonatur',
 'pisonotiene',
 'pisomadera',
 'dis',
 'estadocivil1',
 'estadocivil2',
 'estadocivil3',
 'estadocivil4',
 'estadocivil5',
 'estadocivil6',
 'estadocivil7',
 'parentesco1',
 'parentesco2',
 'parentesco3',
 'parentesco4',
 'parentesco5',
 'parentesco6',
 'parentesco7',
 'parentesco8',
 'parentesco9',
 'parentesco10',
 'parentesco11',
 'parentesco12',
 'tipovivi1',
 'tipovivi2',
 'tipovivi3',
 'tipovivi4',
 'tipovivi5',
 'computer',
 'qmobilephone',
 'refrig',
 'v18q',
 'meaneduc',
 'cielorazo',
 'SQBescolari',
 'eviv3',
 'SQBmeaned',
 'paredblolad',
 'SQBedjefe',
 'instlevel8',
 'lugar1',
 'bedrooms',
 'hogar_nin',
 'r4t1',
 'SQBhogar_nin',
 'overcrowding',
 'SQBovercrowding',
 'r4m1',
 'r4h1',
 'eviv1']
```

In [216]:

```
#Let's have a look at our final dataset.

df_imp = pd.DataFrame(data=df , columns=imp_cols)
df_imp
```

Out[216]:

	rooms	v14a	escolari	abastaguadentro	abastaguafuera	abastaguano	public	planpri	noelec	coopele	...	lugar1	bedrooms	hogar_nin	r4t1	SQI
0	3	1	10	1	0	0	1	0	0	0	...	1	1	0	0	
1	4	1	12	1	0	0	1	0	0	0	...	1	1	0	0	
2	8	1	11	1	0	0	1	0	0	0	...	1	2	0	0	
3	5	1	9	1	0	0	1	0	0	0	...	1	3	2	1	
4	5	1	11	1	0	0	1	0	0	0	...	1	3	2	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9552	6	1	9	1	0	0	0	0	0	1	...	0	4	1	1	
9553	6	1	0	1	0	0	0	0	0	1	...	0	4	1	1	
9554	6	1	5	1	0	0	0	0	0	1	...	0	4	1	1	
9555	6	1	11	1	0	0	0	0	0	1	...	0	4	1	1	
9556	6	1	8	1	0	0	0	0	0	1	...	0	4	1	1	

9552 rows × 74 columns

In [217]:

```
x = df_imp
y = df['Target']
```

In [218]:

```
#for splitting our data into training and testing datsets , we will import the required library.

from sklearn.model_selection import train_test_split
```

In [220]:

```
#so now let's split our data into datasets --> xtrain , xtest , ytrain , ytest

xtrain , xtest , ytrain , ytest = train_test_split(x,y , test_size=0.2 , random_state=0)
xtrain
```

Out[220]:

	rooms	v14a	escolari	abastaguadentro	abastaguafuera	abastaguano	public	planpri	noelec	coopele	...	lugar1	bedrooms	hogar_nin	r4t1	SQI
2760	5	1	2	1	0	0	1	0	0	0	...	1	2	0	0	
4477	7	1	14	1	0	0	1	0	0	0	...	1	5	0	0	
3653	7	1	0	1	0	0	1	0	0	0	...	1	4	2	2	
1890	4	1	3	1	0	0	1	0	0	0	...	1	2	2	2	
4459	8	1	8	1	0	0	1	0	0	0	...	1	5	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7896	3	1	6	1	0	0	1	0	0	0	...	0	2	0	0	
9230	4	1	2	1	0	0	0	0	0	1	...	0	2	1	0	
4864	5	1	6	1	0	0	1	0	0	0	...	1	3	0	0	
3269	5	1	6	1	0	0	1	0	0	0	...	1	3	1	0	
2737	4	1	6	1	0	0	1	0	0	0	...	1	2	0	0	

7641 rows × 74 columns

In [222]:

```
#Importing the required libraries for our Machine Learning Model prepration.

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [224]:

```
#Creating a parameters grid that contains the values of hyperparameters on which we want GridSearchCV to fit RandomForestClassif
#than creating our model on which we will train further on our dateset.
rfc = RandomForestClassifier()
param_grid = {'max_depth':[5,8,10,11,12,14,16,17,18,19,20,21,22] , 'min_samples_leaf':[1,2,3,4]}
gscv = GridSearchCV(rfc , param_grid=param_grid , cv=10 , verbose=3 , return_train_score=True  )
```

In [183]:

```
#fitting our model with training dataset

gscv.fit(xtrain,ytrain)
```

```
[CV 9/10] END max_depth=22, min_samples_leaf=3;, score=(train=0.929, test=0.847) total time=   0.5s
[CV 10/10] END max_depth=22, min_samples_leaf=3;, score=(train=0.931, test=0.857) total time=   0.5s
[CV 1/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.894, test=0.812) total time=   0.5s
[CV 2/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.897, test=0.813) total time=   0.6s
[CV 3/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.898, test=0.813) total time=   0.5s
[CV 4/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.899, test=0.817) total time=   0.6s
[CV 5/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.901, test=0.822) total time=   0.5s
[CV 6/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.903, test=0.825) total time=   0.6s
[CV 7/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.898, test=0.795) total time=   0.5s
[CV 8/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.901, test=0.806) total time=   0.5s
[CV 9/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.897, test=0.818) total time=   0.6s
[CV 10/10] END max_depth=22, min_samples_leaf=4;, score=(train=0.897, test=0.835) total time=   0.5s
```

Out[183]:

```
GridSearchCV(cv=10, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [5, 8, 10, 11, 12, 14, 16, 17, 18, 19, 20,
                                       21, 22],
                         'min_samples_leaf': [1, 2, 3, 4]},
             return_train_score=True, verbose=3)
```

In [186]:

```
#Checking for the best scores that were found in our model for training


gscv.best_score_
```

Out[186]:

```
0.902498545666085
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: