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

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


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

print('Shape of train dataset is {}'.format(train.shape))
print('Shape of test dataset is {}'.format(test.shape))
```

```
Shape of train dataset is (9557, 143)
Shape of test dataset is (23856, 142)
```

```
for i in train.columns:
    if i not in test.columns:
        print("Our Target variable is {}".format(i))
```

```
Our Target variable is Target
```

```
print(train.dtypes.value_counts())
```

```
int64      130
float64      8
object       5
dtype: int64
```

```
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 143 entries, Id to Target
dtypes: float64(8), int64(130), object(5)
memory usage: 10.4+ MB
None
```

```
print(train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 143 entries, Id to Target
dtypes: float64(8), int64(130), object(5)
memory usage: 10.4+ MB
None
```

```
#lets explore each different types of datasets
for i in train.columns:
    a=train[i].dtype
```

```
    if a == 'object':
        print(i)
```

```
Id
idhogar
dependency
edjefe
edjefa
```

```
# lets drop Id variable.

train.drop(['Id','idhogar'],axis=1,inplace=True)
```

```
train['dependency'].value_counts()
```

```
yes          2192
no           1747
.5           1497
2             730
1.5           713
.33333334     598
.66666669     487
8             378
.25           260
3             236
4             100
.75            98
.2             90
1.3333334      84
.40000001      84
2.5            77
5              24
3.5            18
1.25           18
.80000001      18
2.25           13
.71428573      12
.22222222      11
1.75           11
1.2            11
.83333331      11
.2857143        9
.60000002       8
1.6666666       8
6               7
.16666667       7
Name: dependency, dtype: int64
```

```
def map(i):

    if i=='yes':
```

```
        return(float(1))
    elif i=='no':
        return(float(0))
    else:
        return(float(i))

train['dependency']=train['dependency'].apply(map)

for i in train.columns:
    a=train[i].dtype
    if a == 'object':
        print(i)
```

```
edjefe
edjefa
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 141 entries, v2a1 to Target
dtypes: float64(9), int64(130), object(2)
memory usage: 10.3+ MB
```

```
train['edjefe']=train['edjefe'].apply(map)
train['edjefa']=train['edjefa'].apply(map)

train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 141 entries, v2a1 to Target
dtypes: float64(11), int64(130)
memory usage: 10.3 MB
```

```
var_df=pd.DataFrame(np.var(train,0),columns=['variance'])
var_df.sort_values(by='variance').head(15)
print('Below are columns with variance 0.')
col=list((var_df[var_df['variance']==0]).index)
print(col)
```

```
Below are columns with variance 0.
['elimbasu5']
```

```
contingency_tab=pd.crosstab(train['r4t3'],train['hogar_total'])
Observed_Values=contingency_tab.values
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_tab)
Expected_Values = b[3]
no_of_rows=len(contingency_tab.iloc[0:2,0])
no_of_columns=len(contingency_tab.iloc[0,0:2])
df=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",df)
```

```
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in
zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
alpha=0.05
critical_value=chi2.ppf(q=1-alpha,df=df)
print('critical_value:',critical_value)
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',df)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical
variables")
else:
    print("Retain H0,There is no relationship between 2 categorical
variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical
variables")
else:
    print("Retain H0,There is no relationship between 2 categorical
variables")
```

```
Degree of Freedom:- 1
chi-square statistic:- 17022.072400560897
critical_value: 3.841458820694124
p-value: 0.0
Significance level:  0.05
Degree of Freedom:  1
chi-square statistic: 17022.072400560897
critical_value: 3.841458820694124
p-value: 0.0
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

```
contingency_tab=pd.crosstab(train['tipovivi3'],train['v2a1'])
Observed_Values=contingency_tab.values
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_tab)
Expected_Values = b[3]
no_of_rows=len(contingency_tab.iloc[0:2,0])
no_of_columns=len(contingency_tab.iloc[0,0:2])
df=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",df)
from scipy.stats import chi2
```

```
chi_square=sum([(o-e)**2./e for o,e in
zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
alpha=0.05
critical_value=chi2.ppf(q=1-alpha,df=df)
print('critical_value:',critical_value)
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',df)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical
variables")
else:
    print("Retain H0,There is no relationship between 2 categorical
variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical
variables")
else:
    print("Retain H0,There is no relationship between 2 categorical
variables")
```

```
Degree of Freedom:- 1
chi-square statistic:- 54.04781105990782
critical_value: 3.841458820694124
p-value: 1.9562129693895258e-13
Significance level:  0.05
Degree of Freedom:  1
chi-square statistic: 54.04781105990782
critical_value: 3.841458820694124
p-value: 1.9562129693895258e-13
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

```
contingency_tab=pd.crosstab(train['v18q'],train['v18q1'])
Observed_Values=contingency_tab.values
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_tab)
Expected_Values = b[3]
no_of_rows=len(contingency_tab.iloc[0:2,0])
no_of_columns=len(contingency_tab.iloc[0,0:2])
df=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",df)
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in
```

```
zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
alpha=0.05
critical_value=chi2.ppf(q=1-alpha,df=df)
print('critical_value:',critical_value)
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',df)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical
variables")
else:
    print("Retain H0,There is no relationship between 2 categorical
variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical
variables")
else:
    print("Retain H0,There is no relationship between 2 categorical
variables")
```

```
Degree of Freedom:- 0
chi-square statistic:- 0.0
critical_value: nan
p-value: nan
Significance level:  0.05
Degree of Freedom:  0
chi-square statistic: 0.0
critical_value: nan
p-value: nan
Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables
```

```
train.drop('r4t3',axis=1,inplace=True)
```

```
train.parentesco1.value_counts()
```

```
0    6584
1    2973
Name: parentesco1, dtype: int64
```

```
pd.crosstab(train['edjefa'],train['edjefe'])
```

```
edjefe  0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   8.0
9.0   ...  12.0  \
```

```
edjefa                                                                  ..
.
0.0      435   123   194   307   137   222  1845   234   257
486  ...   113
1.0       69     0     0     0     0     0     0     0     0
0  ...     0
2.0       84     0     0     0     0     0     0     0     0
0  ...     0
3.0      152     0     0     0     0     0     0     0     0
0  ...     0
4.0      136     0     0     0     0     0     0     0     0
0  ...     0
5.0      176     0     0     0     0     0     0     0     0
0  ...     0
6.0      947     0     0     0     0     0     0     0     0
0  ...     0
7.0      179     0     0     0     0     0     0     0     0
0  ...     0
8.0      217     0     0     0     0     0     0     0     0
0  ...     0
9.0      237     0     0     0     0     0     0     0     0
0  ...     0
10.0      96     0     0     0     0     0     0     0     0
0  ...     0
11.0     399     0     0     0     0     0     0     0     0
0  ...     0
12.0      72     0     0     0     0     0     0     0     0
0  ...     0
13.0      52     0     0     0     0     0     0     0     0
0  ...     0
14.0     120     0     0     0     0     0     0     0     0
0  ...     0
15.0     188     0     0     0     0     0     0     0     0
0  ...     0
16.0     113     0     0     0     0     0     0     0     0
0  ...     0
17.0      76     0     0     0     0     0     0     0     0
0  ...     0
18.0       3     0     0     0     0     0     0     0     0
0  ...     0
19.0       4     0     0     0     0     0     0     0     0
0  ...     0
20.0       2     0     0     0     0     0     0     0     0
0  ...     0
21.0       5     0     0     0     0     0     0     0     0
0  ...     0

edjefe  13.0  14.0  15.0  16.0  17.0  18.0  19.0  20.0  21.0
edjefa
0.0      103   208   285   134   202    19    14     7    43
```

```
1.0        0     0     0     0     0     0     0     0     0
2.0        0     0     0     0     0     0     0     0     0
3.0        0     0     0     0     0     0     0     0     0
4.0        0     0     0     0     0     0     0     0     0
5.0        0     0     0     0     0     0     0     0     0
6.0        0     0     0     0     0     0     0     0     0
7.0        0     0     0     0     0     0     0     0     0
8.0        0     0     0     0     0     0     0     0     0
9.0        0     0     0     0     0     0     0     0     0
10.0       0     0     0     0     0     0     0     0     0
11.0       0     0     0     0     0     0     0     0     0
12.0       0     0     0     0     0     0     0     0     0
13.0       0     0     0     0     0     0     0     0     0
14.0       0     0     0     0     0     0     0     0     0
15.0       0     0     0     0     0     0     0     0     0
16.0       0     0     0     0     0     0     0     0     0
17.0       0     0     0     0     0     0     0     0     0
18.0       0     0     0     0     0     0     0     0     0
19.0       0     0     0     0     0     0     0     0     0
20.0       0     0     0     0     0     0     0     0     0
21.0       0     0     0     0     0     0     0     0     0

[22 rows x 22 columns]
```

```
train.isna().sum().value_counts()
```

```
0       135
5         2
7928      1
6860      1
7342      1
dtype: int64
```

```
train['Target'].isna().sum()
```

```
0
```

```
float_col=[]
for i in train.columns:
    a=train[i].dtype
    if a == 'float64':
        float_col.append(i)
print(float_col)
```

```
['v2a1', 'v18q1', 'rez_esc', 'dependency', 'edjefe', 'edjefa',
'meaneduc', 'overcrowding', 'SQBovercrowding', 'SQBdependency',
'SQBmeaned']
```

```
train[float_col].isna().sum()
```

```
v2a1               6860
v18q1              7342
```

```
rez_esc            7928
dependency            0
edjefe                0
edjefa                0
meaneduc              5
overcrowding          0
SQBovercrowding       0
SQBdependency         0
SQBmeaned             5
dtype: int64
```

```
train['v18q1'].value_counts()
```

```
1.0    1586
2.0     444
3.0     129
4.0      37
5.0      13
6.0       6
Name: v18q1, dtype: int64
```

```
pd.crosstab(train['tipovivi1'],train['v2a1'])
```

```
v2a1       0.0        12000.0    13000.0    14000.0    15000.0
16000.0    \
tipovivi1

0                 29          3          4          3          3
2

v2a1       17000.0    20000.0    23000.0    25000.0    ...  570540.0
\
tipovivi1                                                ...

0                  4         22          5         21  ...         25


v2a1       600000.0   620000.0   684648.0   700000.0   770229.0
800000.0   \
tipovivi1

0                 11          3          3          7          3
4

v2a1       855810.0   1000000.0  2353477.0
tipovivi1
0                 11          7          2

[1 rows x 157 columns]
```

```
pd.crosstab(train['v18q1'],train['v18q'])
```

```
v18q      1
v18q1
1.0    1586
2.0     444
3.0     129
4.0      37
5.0      13
6.0       6
```

```
train['v2a1'].fillna(0,inplace=True)
train['v18q1'].fillna(0,inplace=True)

train.drop(['tipovivi3',
'v18q','rez_esc','elimbasu5'],axis=1,inplace=True)

train['meaneduc'].fillna(np.mean(train['meaneduc']),inplace=True)
train['SQBmeaned'].fillna(np.mean(train['SQBmeaned']),inplace=True)
print(train.isna().sum().value_counts())
```

```
0    136
dtype: int64
```

```
int_col=[]
for i in train.columns:
    a=train[i].dtype
    if a == 'int64':
        int_col.append(i)
print(int_col)
```

```
['hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'r4h1', 'r4h2',
'r4h3', 'r4m1', 'r4m2', 'r4m3', 'r4t1', 'r4t2', 'tamhog', 'tamviv',
'escolari', 'hhsize', 'paredblolad', 'paredzocalo', 'paredpreb',
'pareddes', 'paredmad', 'paredzinc', 'paredfibras', 'paredother',
'pisomoscer', 'pisocemento', 'pisoother', 'pisonatur', 'pisonotiene',
'pisomadera', 'techozinc', 'techoentrepiso', 'techocane', 'techootro',
'cielorazo', 'abastaguadentro', 'abastaguafuera', 'abastaguano',
'public', 'planpri', 'noelec', 'coopele', 'sanitario1', 'sanitario2',
'sanitario3', 'sanitario5', 'sanitario6', 'energcocinar1',
'energcocinar2', 'energcocinar3', 'energcocinar4', 'elimbasu1',
'elimbasu2', 'elimbasu3', 'elimbasu4', 'elimbasu6', 'epared1',
'epared2', 'epared3', 'etecho1', 'etecho2', 'etecho3', 'eviv1',
'eviv2', 'eviv3', 'dis', 'male', 'female', 'estadocivil1',
'estadocivil2', 'estadocivil3', 'estadocivil4', 'estadocivil5',
'estadocivil6', 'estadocivil7', 'parentesco1', 'parentesco2',
'parentesco3', 'parentesco4', 'parentesco5', 'parentesco6',
'parentesco7', 'parentesco8', 'parentesco9', 'parentesco10',
'parentesco11', 'parentesco12', 'hogar_nin', 'hogar_adul',
'hogar_mayor', 'hogar_total', 'instlevel1', 'instlevel2',
'instlevel3', 'instlevel4', 'instlevel5', 'instlevel6', 'instlevel7',
'instlevel8', 'instlevel9', 'bedrooms', 'tipovivi1', 'tipovivi2',
```

```
'tipovivi4', 'tipovivi5', 'computer', 'television', 'mobilephone',
'qmobilephone', 'lugar1', 'lugar2', 'lugar3', 'lugar4', 'lugar5',
'lugar6', 'area1', 'area2', 'age', 'SQBescolari', 'SQBage',
'SQBhogar_total', 'SQBedjefe', 'SQBhogar_nin', 'agesq', 'Target']
```

```
train[int_col].isna().sum().value_counts()
```

```
0    126
dtype: int64
```

```
train.Target.value_counts()
```

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

```
Poverty_level=train[train['v2a1'] !=0]
```

```
Poverty_level.shape
```

```
(2668, 136)
```

```
poverty_level=Poverty_level.groupby('area1')['v2a1'].apply(np.median)
```

```
poverty_level
```

```
area1
0     80000.0
1    140000.0
Name: v2a1, dtype: float64
```

```
def povert(x):
    if x<8000:
        return('Below poverty level')

    elif x>140000:
        return('Above poverty level')
    elif x<140000:
        return('Below poverty level: Ur-ban ; Above poverty level :
Rural ')
```

```
c=Poverty_level['v2a1'].apply(povert)
```

```
c.shape
```

```
(2668,)
```

```
pd.crosstab(c,Poverty_level['area1'])
```

```
area1                                                  0     1
v2a1
```

```
Above poverty level                                  139  1103
Below poverty level: Ur-ban ; Above poverty lev...  306  1081
```

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

X_data=train.drop('Target',axis=1)
Y_data=train.Target

X_data_col=X_data.columns

from sklearn.preprocessing import StandardScaler
SS=StandardScaler()
X_data_1=SS.fit_transform(X_data)
X_data_1=pd.DataFrame(X_data_1,columns=X_data_col)

X_train,X_test,Y_train,Y_test=train_test_split(X_data_1,Y_data,test_si
ze=0.25,stratify=Y_data,random_state=0)

from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

rfc=RandomForestClassifier(random_state=0)
parameters={'n_estimators':[10,50,100,300],'max_depth':[3,5,10,15]}
grid=zip([rfc],[parameters])

best_=None

for i, j in grid:
    a=GridSearchCV(i,param_grid=j,cv=3,n_jobs=1)
    a.fit(X_train,Y_train)
    if best_ is None:
        best_=a
    elif a.best_score_>best_.best_score_:
        best_=a


print ("Best CV Score",best_.best_score_)
print ("Model Parameters",best_.best_params_)
print("Best Estimator",best_.best_estimator_)
```

```
Best CV Score 0.8507046183898423
Model Parameters {'max_depth': 15, 'n_estimators': 300}
Best Estimator RandomForestClassifier(max_depth=15, n_estimators=300,
random_state=0)
```

```
RFC=best_.best_estimator_
Model=RFC.fit(X_train,Y_train)
pred=Model.predict(X_test)

print('Model Score of train data :
{}'.format(Model.score(X_train,Y_train)))
```

```
print('Model Score of test data :
{}'.format(Model.score(X_test,Y_test)))
```

```
Model Score of train data : 0.9831170643225896
Model Score of test data : 0.8824267782426778
```

```
Important_features=pd.DataFrame(Model.feature_importances_,X_data_col,
columns=['feature_importance'])
```

```
Top50Features=Important_features.sort_values(by='feature_importance',a
scending=False).head(50).index
```

```
Top50Features
```

```
Index(['SQBmeaned', 'meaneduc', 'SQBdependency', 'dependency',
'overcrowding',
       'SQBovercrowding', 'qmobilephone', 'SQBhogar_nin', 'SQBedjefe',
       'edjefe', 'hogar_nin', 'rooms', 'cielorazo', 'r4t1', 'v2a1',
'edjefa',
       'agesq', 'r4m3', 'r4h2', 'SQBage', 'age', 'escolari', 'r4t2',
'r4h3',
       'hogar_adul', 'SQBescolari', 'eviv3', 'bedrooms', 'r4m1',
'epared3',
       'r4m2', 'tamviv', 'paredblolad', 'v18q1', 'SQBhogar_total',
'tamhog',
       'hhsize', 'hogar_total', 'pisomoscer', 'etecho3', 'r4h1',
'lugar1',
       'eviv2', 'tipovivi1', 'energcocinar2', 'energcocinar3',
'epared2',
       'television', 'area2', 'area1'],
      dtype='object')
```

```
for i in Top50Features:
    if i not in X_data_col:
        print(i)
```

```
X_data_Top50=X_data[Top50Features]
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X_data_Top50,Y_data,tes
t_size=0.25,stratify=Y_data,random_state=0)
```

```
Model_1=RFC.fit(X_train,Y_train)
pred=Model_1.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix,f1_score,accuracy_score
```

```
confusion_matrix(Y_test,pred)
```

```
array([[ 143,   17,    0,   29],
       [   8,  324,    4,   63],
       [   1,   12,  214,   75],
       [   2,   10,    3, 1485]])
```

```
f1_score(Y_test,pred,average='weighted')
```

```
0.9026906492316511
```

```
accuracy_score(Y_test,pred)
```

```
0.906276150627615
```

```
# lets drop Id variable.
test.drop('r4t3',axis=1,inplace=True)
test.drop(['Id','idhogar'],axis=1,inplace=True)
test['dependency']=test['dependency'].apply(map)
test['edjefe']=test['edjefe'].apply(map)
test['edjefa']=test['edjefa'].apply(map)

test['v2a1'].fillna(0,inplace=True)
test['v18q1'].fillna(0,inplace=True)

test.drop(['tipovivi3',
'v18q','rez_esc','elimbasu5'],axis=1,inplace=True)

train['meaneduc'].fillna(np.mean(train['meaneduc']),inplace=True)
train['SQBmeaned'].fillna(np.mean(train['SQBmeaned']),inplace=True)

test_data=test[Top50Features]

test_data.isna().sum().value_counts()
```

```
0     48
31     2
dtype: int64
```

```
test_data.SQBmeaned.fillna(np.mean(test_data['SQBmeaned']),inplace=Tru
e)
```

```
/usr/local/lib/python3.7/site-packages/pandas/core/series.py:4536:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  downcast=downcast,
```

```
test_data.meaneduc.fillna(np.mean(test_data['meaneduc']),inplace=True)

Test_data_1=SS.fit_transform(test_data)
X_data_1=pd.DataFrame(Test_data_1)

test_prediction=Model_1.predict(test_data)

test_prediction
```

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