Rahul Agrawal

TC202

Model Building , Training And Testing On pima-indians-diabetes dataset

In [1]:
```
# Importing Libreries

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler , StandardScaler ,RobustScaler
from sklearn import preprocessing

import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [2]:
```
df= pd.read_csv('pima-indians-diabetes.csv' )

df
```

Out[2]:

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)	Class variable (0 or 1)
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Number of times pregnant',
       'Plasma glucose concentration a 2 hours in an oral glucose tolerance test',
       'Diastolic blood pressure (mm Hg)', 'Triceps skin fold thickness (mm)',
       '2-Hour serum insulin (mu U/ml)',
       'Body mass index (weight in kg/(height in m)^2)',
       'Diabetes pedigree function', 'Age (years)', 'Class variable (0 or 1)'],
      dtype='object')
```

In [4]:
```
#  Get Datatypes Of each Columns
dtype={}
for col in df:
    dtype[col]=df[col].dtypes

dtype
```

Out[4]:
```
{'Number of times pregnant': dtype('int64'),
 'Plasma glucose concentration a 2 hours in an oral glucose tolerance test': dtype('int64'),
 'Diastolic blood pressure (mm Hg)': dtype('int64'),
 'Triceps skin fold thickness (mm)': dtype('int64'),
 '2-Hour serum insulin (mu U/ml)': dtype('int64'),
 'Body mass index (weight in kg/(height in m)^2)': dtype('float64'),
 'Diabetes pedigree function': dtype('float64'),
 'Age (years)': dtype('int64'),
 'Class variable (0 or 1)': dtype('int64')}
```

In [5]:
```
df_feature= df.drop(['Class variable (0 or 1)'] , axis=1)
df_label= df['Class variable (0 or 1)']
```

Performing Preprocessing

In [6]:
```
#  using standered scaler


# scaler =RobustScaler()
# scaler = MinMaxScaler()
scaler =StandardScaler()
std=scaler.fit_transform(df_feature)

std_df = pd.DataFrame(std)
std_df.columns=['Number of times pregnant',
       'Plasma glucose concentration a 2 hours in an oral glucose tolerance test',
       'Diastolic blood pressure (mm Hg)', 'Triceps skin fold thickness (mm)',
       '2-Hour serum insulin (mu U/ml)',
       'Body mass index (weight in kg/(height in m)^2)',
       'Diabetes pedigree function', 'Age (years)']
```

In [7]:
```
std_df
```

Out[7]:

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)
0	0.639947	0.848324	0.149641	0.907270	-0.692891	0.204013	0.468492	1.425995
1	-0.844885	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	-0.365061	-0.190672
2	1.233880	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	0.604397	-0.105584
3	-0.844885	-0.998208	-0.160546	0.154533	0.123302	-0.494043	-0.920763	-1.041549
4	-1.141852	0.504055	-1.504687	0.907270	0.765836	1.409746	5.484909	-0.020496
...	...	...	...	...	...	...	...	...
763	1.827813	-0.622642	0.356432	1.722735	0.870031	0.115169	-0.908682	2.532136
764	-0.547919	0.034598	0.046245	0.405445	-0.692891	0.610154	-0.398282	-0.531023
765	0.342981	0.003301	0.149641	0.154533	0.279594	-0.735190	-0.685193	-0.275760
766	-0.844885	0.159787	-0.470732	-1.288212	-0.692891	-0.240205	-0.371101	1.170732
767	-0.844885	-0.873019	0.046245	0.656358	-0.692891	-0.202129	-0.473785	-0.871374

768 rows × 8 columns

In [8]:
```
#  Making test_train split

X_train, X_test, y_train, y_test = train_test_split(std_df, df_label, test_size=0.2, 


X_train
```

Out[8]:

	Number of times pregnant	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Diastolic blood pressure (mm Hg)	Triceps skin fold thickness (mm)	2-Hour serum insulin (mu U/ml)	Body mass index (weight in kg/(height in m)^2)	Diabetes pedigree function	Age (years)
663	1.530847	0.754432	0.563223	1.597279	0.435886	0.749766	0.498693	0.575118
712	1.827813	0.253678	-0.367337	0.969998	-0.692891	1.168599	-0.093250	0.404942
161	0.936914	-0.591345	0.253036	1.220910	0.218813	0.660922	-0.809018	1.000557
509	1.233880	-0.027996	0.459827	-1.288212	-0.692891	-0.887493	-0.189894	2.617224
305	-0.547919	-0.027996	0.356432	1.032726	0.218813	0.978220	-0.775797	-0.360847
...	...	...	...	...	...	...	...	...
645	-0.547919	1.129998	0.253036	0.907270	3.127584	0.940144	-1.020427	-0.275760
715	0.936914	2.068912	-0.987710	0.781814	2.710805	0.242089	1.069496	0.064591
72	2.718712	0.159787	1.080200	-1.288212	-0.692891	1.447821	0.335607	0.745293
235	0.046014	1.568158	0.149641	-1.288212	-0.692891	1.473205	0.021514	-0.616111
37	1.530847	-0.591345	0.356432	1.032726	-0.692891	0.115169	0.583256	1.085644

614 rows × 8 columns

Performing Logistic Regression

In [9]:
```
lr=LogisticRegression()
```

In [10]:
```
lr.fit(X_train,y_train)
```

Out[10]:
```
LogisticRegression()
```

In [11]:
```
#  predicting values for test data

y_pred = lr.predict(X_test)

#  calculating total errored prediction

e=y_pred - y_test
k1=0
for i in e:
    if i!=0:
        k1=k1+1
print("total wrong predictions are :" ,k1)
```

```
total wrong predictions are : 34
```

In [12]:
```
#    calcualting Error betn actaual data and prediceted data
error = mean_squared_error(y_test,y_pred)

print('error in prediction is ',error)
```

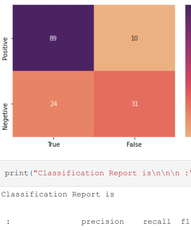
```
error in prediction is  0.22077922077922077
```

In [13]:
```
lr.score(X_test,y_test)
```

Out[13]:
```
0.7792207792207793
```

In [14]:
```
cm=confusion_matrix(y_test,y_pred)  # for stdscaled
x_axis_labels=['True','False']
y_axis_labels=['Positive','Negetive']
sns.heatmap(cm , annot=True , cmap="flare" ,xticklabels=x_axis_labels, yticklabels=y_
print(" Confusion Matrix ")
```

```
 Confusion Matrix
```



In [15]:
```
print("Classification Report is\n\n\n :",classification_report(y_test,y_pred))
```

```
Classification Report is


 :               precision    recall  f1-score   support

           0       0.79      0.90      0.84        99
           1       0.76      0.56      0.65        55

    accuracy                           0.78       154
   macro avg       0.77      0.73      0.74       154
weighted avg       0.78      0.78      0.77       154
```