Name: Saloni Vishwakarma

Roll no: C1-13

▼ Logistic_Regression_Classification

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
data=pd.read_csv('Diabetes.csv')
print(data)
         Pregnancies Glucose BloodPressure SkinThickness Insulin
                   6
                   8
                          183
     3
                          89
                                         66
                   1
                  0
     4
                          137
                                         40
```

10

2

5

1

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1
763	0.171	63	0
764	0.340	27	0
765	0.245	30	0
766	0.349	47	1
767	0.315	23	0

101

122

121

126

93

[768 rows x 9 columns]

data.describe()

763

764

765

766

767

	Dunamanaiaa	61	BloodPressure	SkinThickness	Insulin	DMT	
	Pregnancies	Glucose	Bioodpressure	Skininickness	Insulin	BMI	
coun	t 768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mear	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	
7							

BMI \

33.6

23.3 94 28.1

168 43.1

180 32.9

112 26.2

0 30.1

0 36.8

0 30.4

23

35

48

27

23

0

31

· · · 76

70

72

60

70

data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 768 entries, 0 to 767 Data columns (total 9 columns):

200	COLUMNIS (COCUL) COLUMNIS)	•	
#	Column	Non-Null Count	Dtype
0	Pregnancies	768 non-null	int64
1	Glucose	768 non-null	int64
2	BloodPressure	768 non-null	int64
3	SkinThickness	768 non-null	int64
4	Insulin	768 non-null	int64
5	BMI	768 non-null	float64
6	DiabetesPedigreeFunction	768 non-null	float64
7	Age	768 non-null	int64

dtypes: float64(2), int64(7)
memory usage: 54.1 KB

data.corr()

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insuli
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.07353
Glucose	0.129459	1.000000	0.152590	0.057328	0.33135
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.08893
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.43678
Insulin	-0.073535	0.331357	0.088933	0.436783	1.00000
ВМІ	0.017683	0.221071	0.281805	0.392573	0.19785
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.18507
Age	0.544341	0.263514	0.239528	-0.113970	-0.04216
Outcome	0.221898	0.466581	0.065068	0.074752	0.13054

→ Dataframe 1: data (768 rows)

Dataframe 2: d (392 rows)- average of Glucose, BP, ST, Insulin and BMI

d.describe()

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI I
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	3.301020	122.627551	70.663265	29.145408	156.056122	33.086224
std	3.211424	30.860781	12.496092	10.516424	118.841690	7.027659
min	0.000000	56.000000	24.000000	7.000000	14.000000	18.200000
25%	1.000000	99.000000	62.000000	21.000000	76.750000	28.400000
50%	2.000000	119.000000	70.000000	29.000000	125.500000	33.200000
75%	5.000000	143.000000	78.000000	37.000000	190.000000	37.100000
max	17.000000	198.000000	110.000000	63.000000	846.000000	67.100000
4						

d.info()

#	Column	Non-Null Count	Dtype
0	Pregnancies	392 non-null	int64
1	Glucose	392 non-null	int64
2	BloodPressure	392 non-null	int64
3	SkinThickness	392 non-null	int64
4	Insulin	392 non-null	int64
5	BMI	392 non-null	float64
6	DiabetesPedigreeFunction	392 non-null	float64
7	Age	392 non-null	int64
8	Outcome	392 non-null	int64
dtvn	es: float64(2), int64(7)		

memory usage: 30.6 KB

#data.replace(data['Glucose']==0,value=d['Glucose'].mean(),inplace=True)
data['Glucose'].replace(0,d['Glucose'].mean(),inplace=True)
data['BloodPressure'].replace(0,d['BloodPressure'].mean(),inplace=True)

```
data['SkinThickness'].replace(0,d['SkinThickness'].mean(),inplace=True)
data['Insulin'].replace(0,d['Insulin'].mean(),inplace=True)
data['BMI'].replace(0,d['BMI'].mean(),inplace=True)
```

data.describe()

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	I
768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
3.845052	121.692888	72.325800	29.151052	155.795560	32.466469	
3.369578	30.436043	12.101807	8.790943	85.021487	6.875558	
0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	
3.000000	117.000000	72.000000	29.145408	156.056122	32.400000	
6.000000	140.250000	80.000000	32.000000	156.056122	36.600000	
17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	
	768.000000 3.845052 3.369578 0.000000 1.000000 3.000000 6.000000	768.000000 768.000000 3.845052 121.692888 3.369578 30.436043 0.000000 44.000000 1.000000 99.750000 3.000000 117.000000 6.000000 140.250000	768.000000 768.000000 768.000000 3.845052 121.692888 72.325800 3.369578 30.436043 12.101807 0.000000 44.000000 24.000000 1.000000 99.750000 64.000000 3.000000 117.000000 72.000000 6.000000 140.250000 80.000000	768.000000 768.000000 768.000000 768.000000 3.845052 121.692888 72.325800 29.151052 3.369578 30.436043 12.101807 8.790943 0.000000 44.000000 24.000000 7.000000 1.000000 99.750000 64.000000 25.000000 3.000000 117.000000 72.000000 29.145408 6.000000 140.250000 80.000000 32.000000	768.000000 768.0000000 768.0000000 768.0000000 768.	768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 768.000000 32.466469 32.466469 32.466469 33.369578 30.436043 12.101807 8.790943 85.021487 6.875558 6.875558 0.000000 44.000000 24.000000 7.000000 14.000000 18.200000 1.000000 99.750000 64.000000 25.000000 121.500000 27.500000 3.000000 117.000000 72.000000 29.145408 156.056122 32.400000 6.000000 140.250000 80.000000 32.000000 156.056122 36.600000

data.corr()

lr.fit(x_train,y_train)

Pregnancies Glucose BloodPressure SkinThickness Insulir Pregnancies 1.000000 0.127849 0.208850 0.082926 0.056535 Glucose 0.127849 1.000000 0.219028 0.192985 0.419998 BloodPressure 0.208850 0.219028 1.000000 0.192796 0.072908 SkinThickness 0.082926 0.192985 0.192796 1.000000 0.158154 Insulin 0.056535 0.419998 0.072908 0.158154 1.000000 BMI 0.021589 0.230189 0.281531 0.542239 0.166212 DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366 Outcome 0.221898 0.492948 0.164509 0.215277 0.214532						
Glucose 0.127849 1.000000 0.219028 0.192985 0.419998 BloodPressure 0.208850 0.219028 1.000000 0.192796 0.072908 SkinThickness 0.082926 0.192985 0.192796 1.000000 0.158154 Insulin 0.056535 0.419998 0.072908 0.158154 1.000000 BMI 0.021589 0.230189 0.281531 0.542239 0.166212 DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366		Pregnancies	Glucose	BloodPressure	SkinThickness	Insulir
BloodPressure 0.208850 0.219028 1.000000 0.192796 0.072908 SkinThickness 0.082926 0.192985 0.192796 1.000000 0.158154 Insulin 0.056535 0.419998 0.072908 0.158154 1.000000 BMI 0.021589 0.230189 0.281531 0.542239 0.166212 DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366	Pregnancies	1.000000	0.127849	0.208850	0.082926	0.056535
SkinThickness 0.082926 0.192985 0.192796 1.000000 0.158154 Insulin 0.056535 0.419998 0.072908 0.158154 1.000000 BMI 0.021589 0.230189 0.281531 0.542239 0.166212 DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366	Glucose	0.127849	1.000000	0.219028	0.192985	0.419998
Insulin 0.056535 0.419998 0.072908 0.158154 1.000000 BMI 0.021589 0.230189 0.281531 0.542239 0.166212 DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366	BloodPressure	0.208850	0.219028	1.000000	0.192796	0.072908
BMI 0.021589 0.230189 0.281531 0.542239 0.166212 DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366	SkinThickness	0.082926	0.192985	0.192796	1.000000	0.158154
DiabetesPedigreeFunction -0.033523 0.137004 -0.001108 0.101030 0.098136 Age 0.544341 0.266453 0.325860 0.127780 0.137366	Insulin	0.056535	0.419998	0.072908	0.158154	1.000000
Age 0.544341 0.266453 0.325860 0.127780 0.137366	ВМІ	0.021589	0.230189	0.281531	0.542239	0.166212
	DiabetesPedigreeFunction	-0.033523	0.137004	-0.001108	0.101030	0.098136
Outcome 0.221898 0.492948 0.164509 0.215277 0.214532	Age	0.544341	0.266453	0.325860	0.127780	0.137366
	Outcome	0.221898	0.492948	0.164509	0.215277	0.214532

```
x=data.iloc[:,0:8]

y=data.iloc[:,-1]

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

C:\Users\acer\anaconda3\lib\site-packages\sklearn\linear_model_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (statustop: TOTAL NO. of ITERATIONS REACHED LIMIT.

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
    n_iter_i = _check_optimize_result(
LogisticRegression()
```

```
000100]
    Actual Values:
    661
    122
           0
    113
           0
    14
           1
    529
           0
    476
          1
    482
           a
    230
           1
           0
    527
    380
           0
    Name: Outcome, Length: 154, dtype: int64
y_test
    661
           1
    122
           0
           0
    113
    14
           1
    529
           0
    476
    482
           0
    230
           1
    527
    380
    Name: Outcome, Length: 154, dtype: int64
cf=confusion_matrix(y_test,y_pred)
print(cf)
print("Classification Report for Testing Dataset:")
print(classification_report(y_test,y_pred))
    [[94 13]
     [18 29]]
    Classification Report for Testing Dataset:
                           recall f1-score
                 precision
                                               support
              0
                      0.84
                               0.88
                                        0.86
                                                  107
                      0.69
                               0.62
                                        0.65
                                                   47
        accuracy
                                        0.80
                                                  154
                      0.76
                               0.75
       macro avg
                                        0.76
                                                  154
                                        0.80
    weighted avg
                      0.79
                               0.80
                                                  154
y\_train\_pred=lr.predict(x\_train)
print("Classification Report for Training Dataset:")
print(classification_report(y_train,y_train_pred))
    Classification Report for Training Dataset:
                 precision
                            recall f1-score
                                               support
              0
                      0.78
                               0.88
                                        0.83
                                                   393
                               0.57
                                        0.64
                                                   221
```

0.77

0.73

0.76

0.73

0.77

614

614

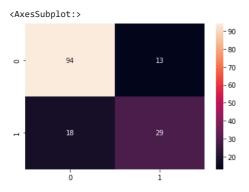
614

cf=confusion_matrix(y_test,y_pred)
sns.heatmap(cf,annot=True)

accuracy

macro avg

weighted avg



0.76

0.76