DIABETES PREDICTION USING LOGISTIC REGRESSION

In this project we try to predict diabetes based on diagnostic measures using Logistic regression(classification). Here Pima Indians Diabetes Database is used.

The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, glucose level, blood pressure, their BMI, insulin level, skin thickness, age, diabetespedigreefunction and result.

CODE

```
from google.colab import files

uploaded = files.upload()

import pandas as pd


df=pd.read_csv("diabetes.csv")

df

#EDA

df.info()

df.shape

df.size

df.head(5)

df.tail(5)

import seaborn as sns

sns.countplot(x='Outcome',data=df)

x=df.iloc[:,0:8].values

y=df.iloc[:,8].values

from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
print(x.shape)
print(x_train.shape)
print(x_test.shape)
#applying classifier
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()

model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
#calculating accuracy of model
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100

x=model.predict([[6      ,148,    72,      35,      0,       33.6     ,0.627   ,50       ]])

if x==1:
  print("Diabetic")
else:
  print("Not diabetic")
x=model.predict([[1,     89,      66,      23,      94,      28.1,    0.167    ,21]])
if x==1:
  print("Diabetic")
else:
  print("Not diabetic")
```

OUTPUT SCREENSHOTS

```
[ ] from google.colab import files
    uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving diabetes.csv to diabetes.csv
```

```
[ ] import pandas as pd

    df=pd.read_csv("diabetes.csv")
    df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

```
[ ] #EDA
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   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  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
[ ] df.shape
```

```
(768, 9)
```

```
[ ] df.head(5)
```

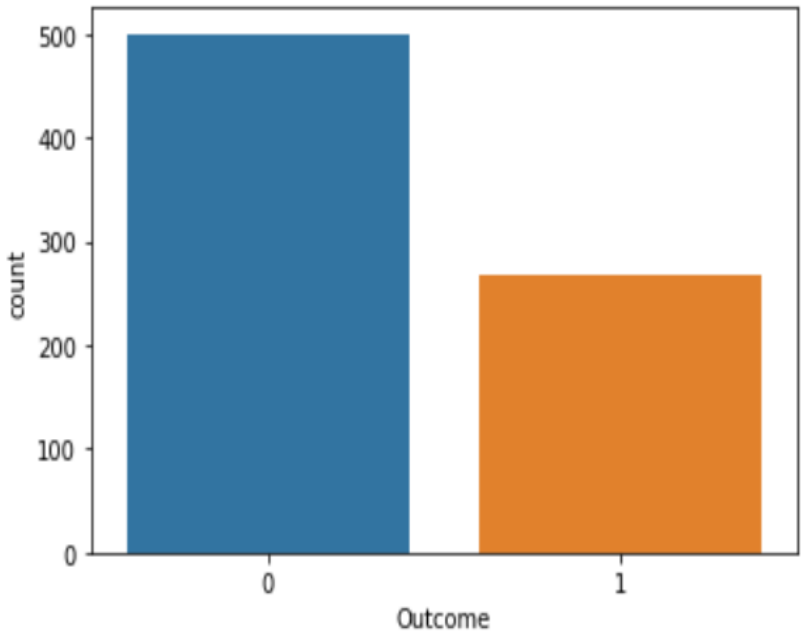
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
[ ] df.tail(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0

```
[ ] import seaborn as sns
    sns.countplot(x='Outcome',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f86aa5e6fd0>
```




```
[ ] x=df.iloc[:,0:8].values
    y=df.iloc[:,8].values
```

```
[ ] from sklearn.model_selection import train_test_split
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

```
[ ] print(x.shape)
    print(x_train.shape)
    print(x_test.shape)
```

```
(768, 8)
(576, 8)
(192, 8)
```

```
[ ] #applying classifier
    from sklearn.linear_model import LogisticRegression
    model=LogisticRegression()
```

```
[ ] y_pred=model.predict(x_test)
```

```
[ ] y_pred
```

```
array([1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0])
```

```
#calculating accuracy of model
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100
```

```
79.16666666666666
```

```
[ ]
x=model.predict([[6 ,148, 72, 35, 0,  33.6  ,0.627  ,50 ]])

if x==1:
  print("Diabetic")
else:
  print("Not diabetic")
```

```
Diabetic
```

```
[ ] x=model.predict([[1,  89, 66, 23, 94, 28.1, 0.167 ,21]])
if x==1:
  print("Diabetic")
else:
  print("Not diabetic")
```

```
Not diabetic
```