

## Diabetes Prediction Using KNN

```
In [2]: #Import Libraries
import numpy as np
import pandas as pd
```

```
In [3]: #Load Dataset
df=pd.read_csv('/home/student/Downloads/diabetes.csv')
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...	...	...
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

```
In [4]: df.columns
```

```
Out[4]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',  
              'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],  
              dtype='object')
```

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

```
Out[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

```
In [6]: df.tail()
```

```
Out[6]:
```

	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
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

```
In [7]: #Summarize Dataset
df.shape
```

```
Out[7]: (768, 9)
```

```
In [8]: df.isna().sum()
```

```
Out[8]: Pregnancies      0
Glucose      0
BloodPressure  0
SkinThickness  0
Insulin      0
BMI          0
DiabetesPedigreeFunction  0
Age          0
Outcome      0
dtype: int64
```

```
In [9]: #Segregate Dataset into X(Input Data)
x=df.iloc[:, :-1].values
x
```

```
Out[9]: array([[ 6.   , 148.   , 72.   , ..., 33.6   , 0.627, 50.   ],
 [ 1.   , 85.    , 66.   , ..., 26.6   , 0.351, 31.   ],
 [ 8.   , 183.   , 64.   , ..., 23.3   , 0.672, 32.   ],
 ...,
 [ 5.   , 121.   , 72.   , ..., 26.2   , 0.245, 30.   ],
 [ 1.   , 126.   , 60.   , ..., 30.1   , 0.349, 47.   ],
 [ 1.   , 93.    , 70.   , ..., 30.4   , 0.315, 23.   ]])
```

```
In [10]: #Y(Output Data)
y=df.iloc[:, -1].values
y
```

```
Out[10]: array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1,
1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
```

In [13]: *#Splitting Dataset into Train & Test Datasets*

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30)
```

In [14]: *#Feature Scaling- Standard Scaler*

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [15]: x\_train

Out[15]: array([[ -1.14548682, -0.4745438 , -0.35879275, ..., 0.60440588,  
 0.90262253, -0.7091095 ],  
 [ -0.85046216, 0.54286864, -1.19106794, ..., -0.38669267,  
 0.58427028, -0.96302713],  
 [ -0.26041283, -1.26939727, 0.05734485, ..., -1.34015456,  
 -0.23479133, -0.7091095 ],  
 ...,  
 [ 0.62466116, 2.3233404 , 0.05734485, ..., -0.11069054,  
 -0.42333004, -0.20127425],  
 [ -0.26041283, 2.25975213, 0.05734485, ..., 0.39113151,  
 -0.69222952, -0.7091095 ],  
 [ 1.21471049, 1.43310452, 1.92996402, ..., 0.72986139,  
 -0.92713021, 0.81439627]])

In [16]: x\_test



```
In [16]: x_test
```

```
Out[16]: array([[ -1.14548682,  1.81463418,  1.09768883, ...,  1.54532222,
                  0.68317584, -0.87838792],
                [ -0.5554375 , -0.69710277, -0.25475835, ..., -0.26123716,
                  -0.2996981 , -1.04766634],
                [  0.03461183,  0.63825105,  0.88962003, ...,  1.53277666,
                  0.5564531 , -0.96302713],
                ...,
                [ -1.14548682,  0.54286864, -0.35879275, ..., -1.21469905,
                  -0.79731667, -1.04766634],
                [ -0.26041283,  1.56028107,  0.16137924, ...,  0.19040269,
                  -0.82204305, -0.79374871],
                [  1.50973515,  0.95619244,  0.47348244, ...,  0.30331265,
                  1.32297113, -0.03199583]])
```

```
In [17]: #KNN model
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
y_pred
```

```
Out[17]: array([1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,  
                0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
                0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,  
                0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,  
                1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,  
                0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,  
                0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,  
                0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
In [18]: print(classifier.predict([[0,137,40,35,168,43.1,2.288,33]]))
```

```
[1]
```

```
In [19]: #Evaluating Model -Accuracy Score and Confusion Matrix and ConfusionMatrixDisplay
from sklearn.metrics._plot.confusion_matrix import confusion_matrix
from sklearn.metrics import classification_report, accuracy_score, ConfusionMatrixDisplay
labels= [ 'Diabetic', 'not diabetic']
result=confusion_matrix(y_test,y_pred)
cm=ConfusionMatrixDisplay(result,display_labels=labels)
cm.plot()
score=accuracy_score(y_test,y_pred)
print(result)
print(score)
```

```
[[124  26]
 [ 41  40]]
0.70995670995671
```

