AI DIABETES PREDICTION SYSTEM

NAME:SIVANESAN.G

REG NO:720421104045

INTRODUCTION :

K-Nearest Neighbours (KNN) is a popular machine learning algorithm used for classification and regression tasks. It is a **lazy learning**, non-parametric algorithm that uses data with several classes to predict the classification of the new sample point. KNN is non-parametric since it doesn't make any assumptions on the data being studied.

During the training phase, the KNN algorithm stores the entire training dataset as a reference. When implementing an algorithm, you will always need a data set. So, you start by loading the training and the test data. Then, you choose the nearest data points (the value of K). K can be any integer.

The working of KNN Algorithm in Machine Learning can be summarized in three steps:

1. Load the data
2. Choose the nearest data points $\left(\text{the value of K}\right)$
3. Do the following, for each test data –
    - Calculate the distance between test data and each row of training data
    - Sort the calculated distances in ascending order based on distance values
    - Get top K rows from sorted array
    - Get the most frequent class of these rows
    - Return this class as output.

PROCESS:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score


dataset=pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")

print(len(dataset))

print(dataset.head())
```

OUTPUT:

```
768
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   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

```
nonzero=['Glucose','BloodPressure','SkinThickness','Insulin','BMI']

for col in nonzero:
    dataset[col]=dataset[col].replace(0,np.NaN)
    mean=int(dataset[col].mean(skipna=True))
    dataset[col]=dataset[col].replace(np.NaN,mean)
print(dataset['Glucose'])
```

OUTPUT:

```
0      148.0
1       85.0
2      183.0
3       89.0
4      137.0
       ...
763    101.0
764    122.0
```

765   121.0

766   126.0

767   93.0

Name: Glucose, Length: 768, dtype: float64

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

y=dataset.iloc[:,8]

x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=
0.3)

sc=StandardScaler()

x_train=sc.fit_transform(x_train)

x_test=sc.transform(x_test)

classifier=KNeighborsClassifier(n_neighbors=15,p=2,metric='euclidean')

model=classifier.fit(x_train,y_train)
```

```
yp=classifier.predict(x_test)

yp
```

OUTPUT:

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

```
CM=confusion_matrix(y_test,yp)

print(CM)
```

OUTPUT:

$$\left[\begin{bmatrix}133 & 13\end{bmatrix}\right.$$
$$\left.\begin{bmatrix}34 & 51\end{bmatrix}\right]$$

```
print("F-Score: ",(f1_score(y_test,yp)))
```

OUTPUT:

F-Score: 0.6845637583892616

CONCLUSION:

The AI prediction system using the KNN algorithm has shown promise in making
accurate predictions. While it has its strengths, we acknowledge its limitations and recommend further research and improvements to maximize its potential. The system has the potential to contribute to data-driven decision-making and add value in real-world applications.