ST JOSEPH COLLEGE OF ENGINEERING

TITLE : AI-BASED DIABETICS PREDICTION MODEL
PHASE-5

PROJ_227128_TEAM_1

TITLE : AI-BASED DIABETICS PREDICTION MODEL

C
O
D
E

Step 1 – Importing Modules

Now, let's import the necessary Python libraries into our notebook.

Keras API already includes Python's TensorFlow deep learning package, which is critical in the diabetes prediction challenge

CODE:

```
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.layers import Dense,Dropout
from sklearn.model_selection import
train_test_split
import matplotlib as mlp
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import StandardScaler
```

C
O
D
E

Step 2 – Loading the Dataset
We are now ready to begin importing the dataset. In the next piece of code, we import the
dataset and use the head() method to get the top five data points.

CODE:

```
data=pd.read_csv("pima-indians-diabetes.csv")
data.head()
```

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0

the diabetics and their related documented data are taken from the exact code provided

C O D E

Step 3 – Renaming the Columns You've probably realized that the columns are meaningless

CODE:
head() in Pandas

```
data = data.rename(index=str, columns={"6":"preg"})
data = data.rename(index=str,
columns={"148":"gluco"})data =
data.rename(index=str, columns={"72":"bp"}) data =
data.rename(index=str,
columns={"35":"stinmm"})data =
data.rename(index=str, columns={"0":"insulin"})
data = data.rename(index=str,
columns={"33.6":"mass"})data
=data.rename(index=str, columns={"0.627":"dpf"})
data = data.rename(index=str,
columns={"50":"age"}) data =
```

```
data.rename(index=str, columns={"1":"target"})
data.head()
```

Step 3:

	preg	gluco	bp	stinmm	insulin	mass	dpf	age	target
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0

the previous schedule of this table is modified by adding some names to the rows andf columns to easily recognise what happens exactly and whats the data given to it

Step 4 – Separating Inputs and Outputs

C O D E

Code:

```
X = data.iloc[:, :-1]
Y = data.iloc[:,8]
```

The X and Y values look somewhat like this:
We separated our dataset into input and target datasets, which implies that the first eight columns will serve as input features for our model and the last column will serve as the target class.

```
print(X)
```

```
     preg  gluco  bp  stinmm  insulin  mass    dpf  age
0       1     85  66      29        0  26.6  0.351   31
1       8    183  64       0        0  23.3  0.672   32
2       1     89  66      23       94  28.1  0.167   21
3       0    137  40      35      168  43.1  2.288   33
4       5    116  74       0        0  25.6  0.201   30
..    ...    ...  ..     ...      ...   ...    ...  ...
762    10    101  76      48      180  32.9  0.171   63
763     2    122  70      27        0  36.8  0.340   27
764     5    121  72      23      112  26.2  0.245   30
765     1    126  60       0        0  30.1  0.349   47
766     1     93  70      31        0  30.4  0.315   23

[767 rows x 8 columns]
```

```
print(Y)
```

```
0      0
1      1
2      0
3      1
4      0
      ..
762    0
763    0
764    0
765    1
766    0
Name: target, Length: 767, dtype: int64
```

the collected data is now allocated with x,y valuse in the represented table which allocates the source of the csv file

Step 5 – Train-Test Split of the Data

The next step involves the training and

C O D E

testing split into data and then standardizing the data to make computations simpler later on **code:**

```
X_train_full, X_test, y_train_full, y_test =
train_test_split(X, Y, random_state=42)
X_train, X_valid, y_train, y_valid =
train_test_split(X_train_full, y_train_full,
random_state from sklearn.preprocessing
import StandardScaler scaler =
StandardScaler()

X_train = scaler.fit_transform(X_train)X_valid
= scaler.transform(X_valid)
X_test = scaler.transform(X_test)
```

C O D E

Step 6 – Building the Model We start off by using a random seed to generate a pseudo-random number and
setting it to the tf graph. Then, we will be using a sequential model, and also some dropout layers in the model to avoid overfitting of the data. code:

```
np.random.seed(42) tf.random.set_seed(42)
model=Sequential()
model.add(Dense(15,input_dim=8, activation='relu'))
model.add(Dense(10,activation='relu'))
model.add(Dense(8,activation='relu'))
model.add(Dropout(0.25)) model.add(Dense(1,
activation='sigmoid'))
```

C
O
D
E

Step 7 – Training and Testing of the Model Now, let’s move forward to train our model and then fit the model on the testing dataset. **code:**

model.compile(loss="binary_crossentropy", optimizer="SGD", metrics=['accuracy'])
model_history = model.fit(X_train, y_train, epochs=200, validation_data=(X_valid, y_valid))You will realize that will train the model for 200 epochs and use binary-cross entropy loss function and SGD optimizer.

step:8

The conclusion:

The AI-based diabetes prediction system represents a promising advancement in the field of healthcare technology. By leveraging sophisticated algorithms and machine learning techniques, it has demonstrated its potential to accurately forecast the likelihood of diabetes onset in individuals. This system holds the potential to revolutionize early intervention and preventative care strategies, ultimately improving the quality of life for those at risk of developing diabetes. However, it is essential to continue refining the model, validating its predictions through extensive clinical trials, and ensuring its seamless integration into existing healthcare workflows. With further development and implementation, this AI system has the potential to significantly impact public health outcomes and contribute to a more proactive approach in managing diabetes.

THANK YOU !

REPORTED BY SANJAY.A
212921104044

PROJ_227128_TEAM_1