

**CODE:**

**data=pd.read\_csv("pima-indians-**

**diabetes.csv")**

**data.head()**

**Step 2**

**– Loading the Dataset**

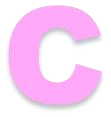
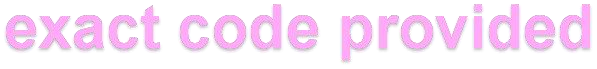
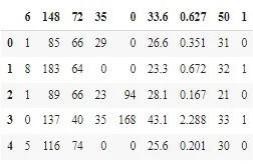
**We are now ready to begin importing the**

**dataset. In the next piece of code, we import**

**the**

**dataset and use the head() method to get the**

**top five data points.**



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

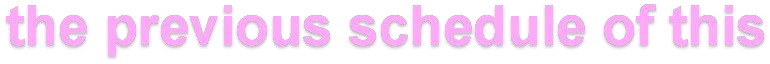
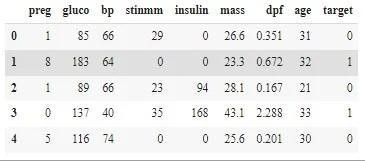
**CODE:**

**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**



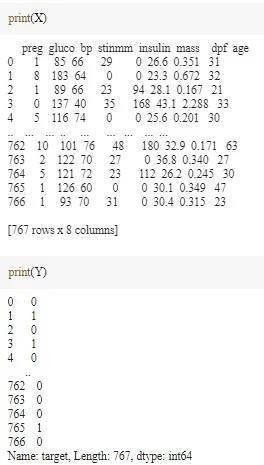
**=data.rename(index=str, columns={"0.627":"dpf"}) data = data.rename(index=str, columns={"50":"age"}) data = data.rename(index=str, columns={"1":"target"}) data.head()**



**Step 4 – Separating Inputs and Outputs**

**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.**



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

**Step 5 –** **of the Data**

**The next step involves the training and**

**testing**  **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)**

**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'))**

**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:**



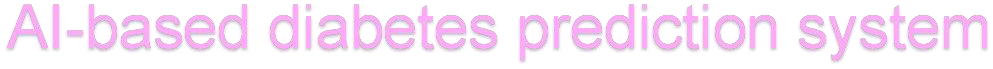
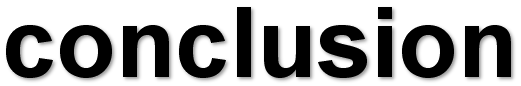
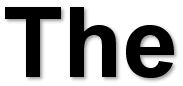
**model.compile(loss="binary\_crossentropy", optimizer="SGD", metrics=['accuracy']) model\_history = model.fit(X\_train, y\_train, epochs=200, validation\_data=(X\_valid, y\_valid))You will realize that will train the model for 200 epochs and use binary-cross entropy loss function and SGD optimizer.**

**step:8**

represents a promising

T

he



advancement in the field of healthcare technology. By leveraging sophisticated algorithms and machine learning techniques, it has demonstrated



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 

 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.

