**AUTHOR**: Gifty.G

**PROJECT TITTLE:** AI Based Diabetes Prediction System

**COMPREHENSIVE OVERVIEW:** The "AI Based Diabetes Prediction System" is a project deels with analysis of person’s medical condition. The main contributions of this project is to develop an automated remote monitoring framework that consolidates patient vital data from different multi‐vendor personal health devices, smartphones and smartwatches to enhance diagnostic decision making. Integrate a machine learning model within framework and use a subset of the data collected to provide live diabetes risk predictions. This project is more like an End-to-End Data Analytics project which integrates a collection of skills and tools such as:

-NumPy and Pandas for Data Cleaning. It could have been better to perform Initial Data Cleaning/Data Pre-processing more faster if I used Spreadsheet. But the only reason why i used Python libraries for Data Cleaning/Data pre-processing is that i can clearly demonstrate each and every step of data cleaning using it. For an instance, how the data has been converted from it's original format into the correct format (if the original data is wrong).

-Geophy for Feature Engineering (created a new field using information from already existing fields, which can have scopes of usage in future.)

-Matplotlib, Seaborn and Follum for general Exploratory Data Analysis (EDA) and for some basic Data Visualizations within the Jupyter Notebook.

-Tableau for Visualizations of Key Findings / Insights and for creating the final Dashboard.

NOTE: This Notebook includes only Data Cleaning, Data Transformation (creating a new DataFrame using records from scopeful rows), Feature Engineering, few EDA and some visualizations

**PROJECT DETAILS:**

Data Source:The project extracts data from pima-indias-diabetes.csv

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.

Updated DataSet: As the original dataset can not also be clean enough in order take that dataset into Data Analysis phase, I have cleaned the dataset and stored the cleaned dataset as an "updated\_dataset" in ".csv" file extension.

Export to Spreadsheet: The project includes functionality to export the extracted DataFrame to a spreadsheet. Users have the option to save the data in CSV or Excel format, enabling further analysis and visualization using various data analysis tools.

Documentation: Detailed Markdown cells provide comprehensive documentation for the entire web scraping process Explanations of the project's objectives, web scraping methodology, code explanations, and data output are included

Sample Output: displaying cleaned and transformed data.

Future Plans: Even though the Data is a Real-time data, no real-time audience or no real-time clients are involved to showcase and present my Dashboard as this is an educational purpose project. I consider every user going through my project is my client/audience.

***Python Program***:

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

**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.

***Python Program***:

data=pd.read\_csv("pima-indians-diabetes.csv")

data.head()

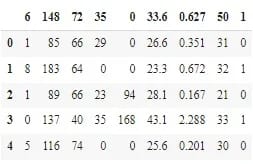


Fig: Diabetes Dataset Top5

**Renaming the Columns:**

You’ve probably realized that the columns are meaningless, right? Let us now rename the column names.

Also read: head() in Pandas

***Python Program***:

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()

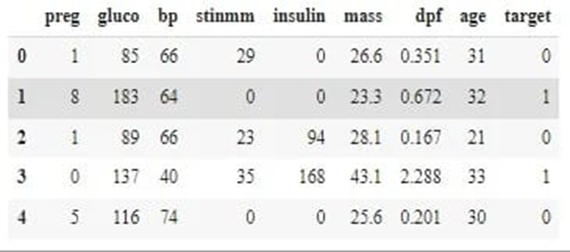


Fig: Renamed Columns Diabetes Dataset Top5

**Separating Inputs and Outputs:**

***Python Program***:

X = data.iloc[:, :-1]

Y = data.iloc[:,8]

The X and Y values look somewhat like this:

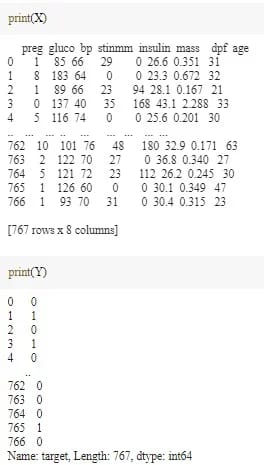


Fig: Input N Output Diabetes Dataset

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.

**Train-Test Split of the Data:**

The next step involves the training and testing split into data and then standardizing the data to make computations simpler later on.

***Python Program***:

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=42)

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)

**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.

***Python Program***:

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'))

**Training and Testing of the Model:**

Now, let’s move forward to train our model and then fit the model on the testing dataset.

***Python Program***:

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.

**Conclusion**

Contribution of the Explainable AI in Diabetes Prediction system makes it easy for the end-user to understand the AI systems' complex working. It provides a human-centered interface to the user. Explainability is a key to producing a transparent, proficient, and accurate AI system that can help the healthcare practitioner, patients, and researcher understand and use the system.