**DIABETES PREDICTION USING MACHINE LEARNING**

au622021243025 : John J Jasho



**Abstract:**

* Globally, diabetes affects 537 million people, making it the deadliest and the most common non‐communicable disease. Many factors can cause a person to get affected by diabetes, like excessive body weight, abnormal cholesterol level, family history, physical inactivity, bad food habit etc.
* Increased urination is one of the most common symptoms of this disease. People with diabetes for a long time can get several complications like heart disorder, kidney disease, nerve damage, diabetic retinopathy etc. But its risk can be reduced if it is predicted early.

**Introduction**

* For predicting blood pressure status, they used conditional decision making and for predicting diabetes, they used SVM, KNN, and decision tree. Among these models, SVM worked better as they got 75% accuracy which is better than other classifier algorithms.
* Random forest is a machine learning system that averages the predictions of several decision trees. As a result, the random forest can be considered an ensemble learning model

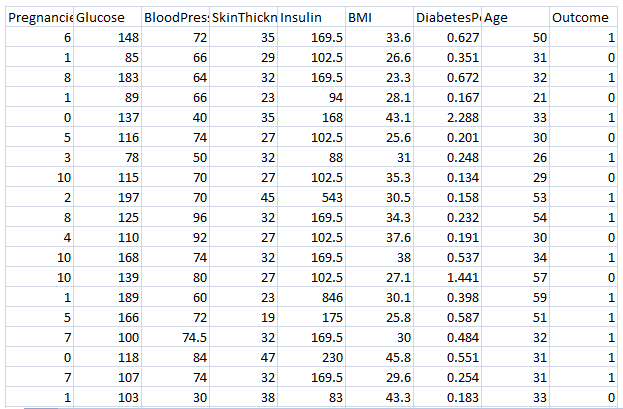
**Content for Project Phase 2:**

Consider exploring advanced regression techniques like Gradient Boosting or XGBoost for improved Prediction accuracy.

**Data source**

The datasets consists of several medical predictor variables and one target variable, outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

**Dataset link:** ( <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database> )



**Steps involved in Diabetes Prediction Project**

* Collection of data
* Exploring the data
* Splitting the data
* Training the model
* Evaluating the model
* Deploying the model

**Data collection:**

The very first step is to choose the dataset for our model. We can get a lot of different datasets from Kaggle. You just need to sign in to Kaggle and search for any dataset you need for the project. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict whether a patient has diabetes based on diagnostic measurements. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

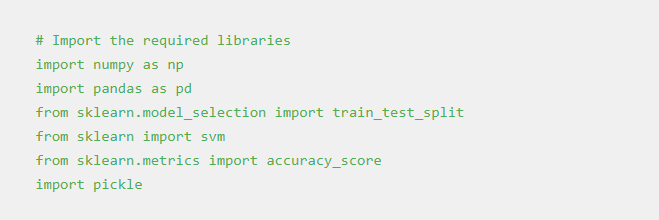
The data contains 9 columns which are as follows

* **Pregnancies**: Number of times pregnant
* **Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
* **BloodPressure**: Diastolic blood pressure (mm Hg)
* **SkinThickness**: Triceps skin fold thickness (mm)
* **Insulin**: 2-Hour serum insulin (mu U/ml)
* **BMI**: Body mass index (weight in kg/(height in m)^2)
* **DiabetesPedigreeFunction**: Diabetes pedigree function
* **Age**: Age (years)
* **Outcome**: Class variable (0 or 1)

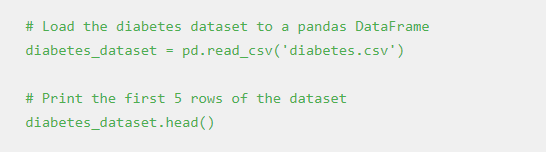
### Exploring the Data

Now we have to set the development environment to build our project. For this project, we are going to build this Diabetes prediction using Machine Learningin [Google Colab](http://colab.research.google.com/). You can also use Jupyter Notebook.

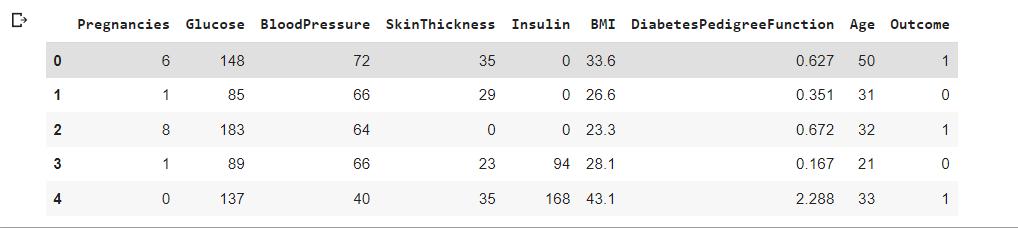
After downloading the dataset, import the necessary libraries to build the model.

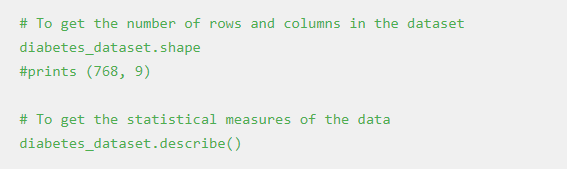


Load the data using the *read\_csv* method in the pandas library. Then the *head()* method in the pandas library is used to print the rows up to the limit we specify. The default number of rows is five.

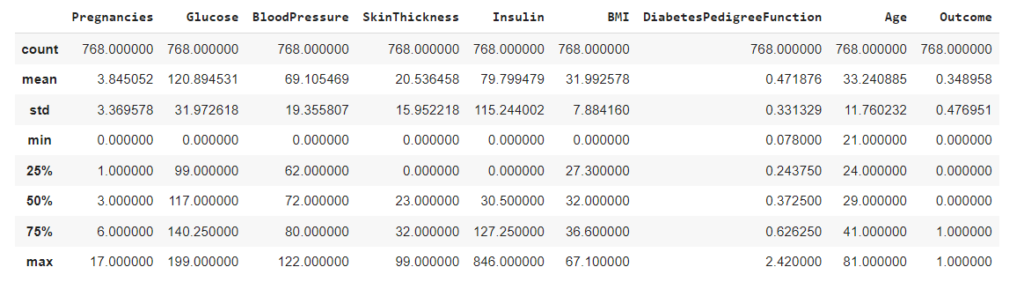


**Output:**

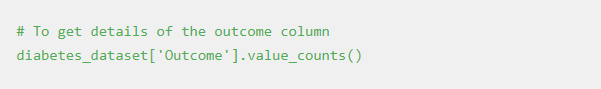




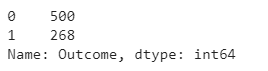
**Output:**



And, it is clear that the Outcome column is the output variable. So let us explore more details about that column.

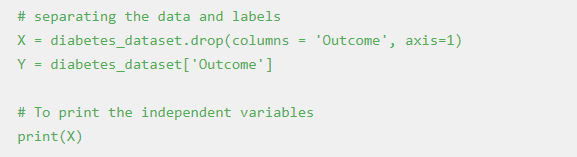


In the output, the value 1 means the person is having Diabetes, and 0 means the person is not having Diabetes. We can see the total count of people with and without Diabetes.

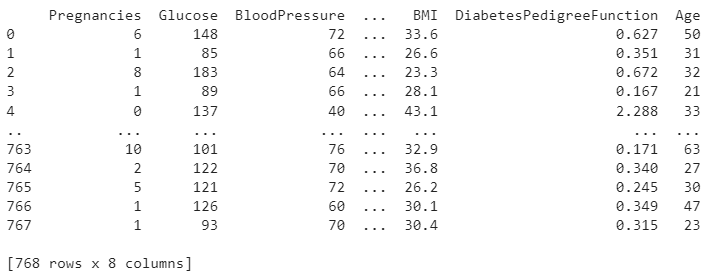


### Splitting the data

The next step in the building of the Machine learning model is splitting the data into training and testing sets. The training and testing data should be split in a ratio of 3:1 for better prediction results.



**Output:**



#Split the data into train and test

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y, test\_size = 0.2, stratify=Y, random\_state=2)

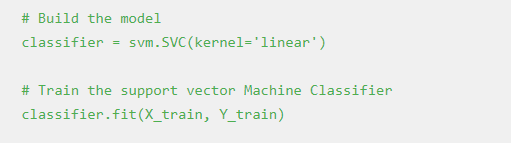
print(X.shape, X\_train.shape, X\_test.shape)

**Output:**

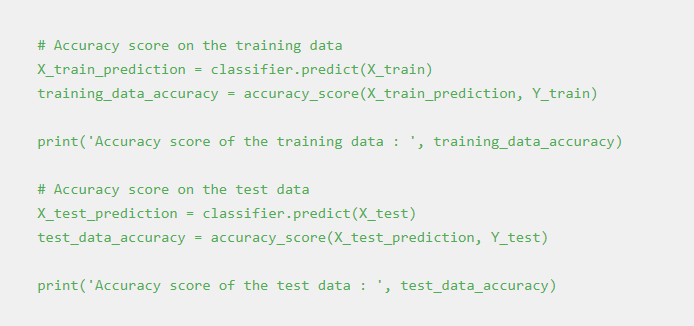
*(768, 8) (614, 8) (154, 8)*

### Training the model

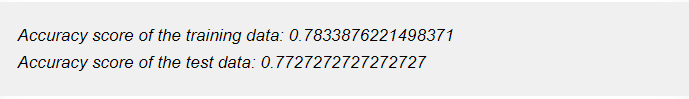
The next step is to build and train our model. We are going to use a Support vector classifier algorithm to build our model.



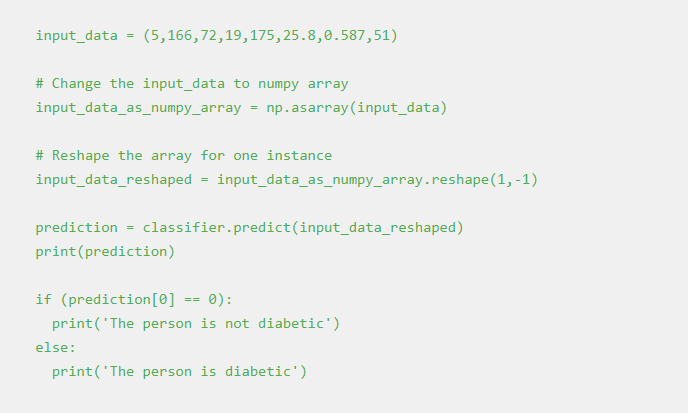
After building the model, the model has to predict output with test data. After the prediction of the outcome with test data, we can calculate the accuracy score of the prediction results by the model.



**Output:**



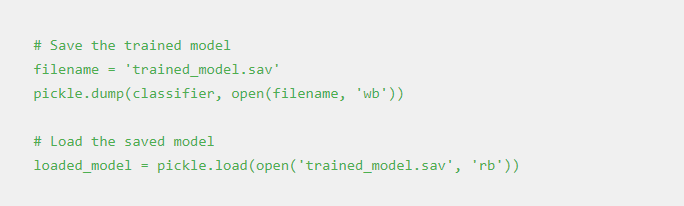
**5. Evaluating the model**



**Output**:

*The person is diabetic*

**Saving the file**



Once you run this code a new file named trained\_model.sav will be saved in the project folder.

## **Deploying the model**

One of the most important and final steps in building a Machine Learning project is Model deployment. There are many frameworks available for deploying the Machine learning model on the web. Some of the most used Python frameworks are Django and Flask. But these frameworks require a little knowledge of languages such as HTML, CSS, and JavaScript.

So, a new framework known as Streamlit was introduced to deploy the Machine Learning model without the need to have the knowledge of Front End Languages. It is quite easy to deploy using Streamlit. So, we will use the [Streamlit](https://streamlit.io/) framework to deploy our model. Although Streamlit has many advantages over the other frameworks, lot more features are under development. If you are getting started in Machine Learning then this framework will be a perfect start to deploy your machine learning model on the web.

### Python Code to Deploy ML model using Streamlit

To install Streamlit run the following command in the command prompt or terminal.



Open a new Python file and put the following code.

***App.py***

import numpy as np

import pickle

import streamlit as st

# Load the saved model

loaded\_model = pickle.load(open('C:/Users/ELCOT/Downloads/trained\_model.sav', 'rb'))

# Create a function for Prediction

def diabetes\_prediction(input\_data):

# Change the input\_data to numpy array

input\_data\_as\_numpy\_array = np.asarray(input\_data)

# Reshape the array as we are predicting for one instance

input\_data\_reshaped = input\_data\_as\_numpy\_array.reshape(1,-1)

prediction = loaded\_model.predict(input\_data\_reshaped)

print(prediction)

if (prediction[0] == 0):

return 'The person is not diabetic'

else:

return 'The person is diabetic'

def main():

# Give a title

st.title('Diabetes Prediction Web App')

# To get the input data from the user

Pregnancies = st.text\_input('Number of Pregnancies')

Glucose = st.text\_input('Glucose Level')

BloodPressure = st.text\_input('Blood Pressure value')

SkinThickness = st.text\_input('Skin Thickness value')

Insulin = st.text\_input('Insulin Level')

BMI = st.text\_input('BMI value')

DiabetesPedigreeFunction = st.text\_input('Diabetes Pedigree Function value')

Age = st.text\_input('Age of the Person')

# Code for Prediction

diagnosis = ''

# Create a button for Prediction

if st.button('Diabetes Test Result'):

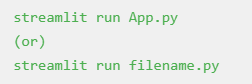
diagnosis = diabetes\_prediction([Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age])

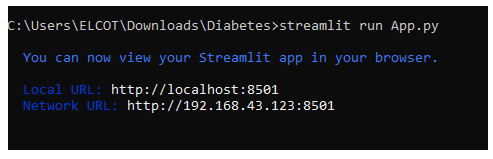
st.success(diagnosis)

if \_\_name\_\_ == '\_\_main\_\_':

main()

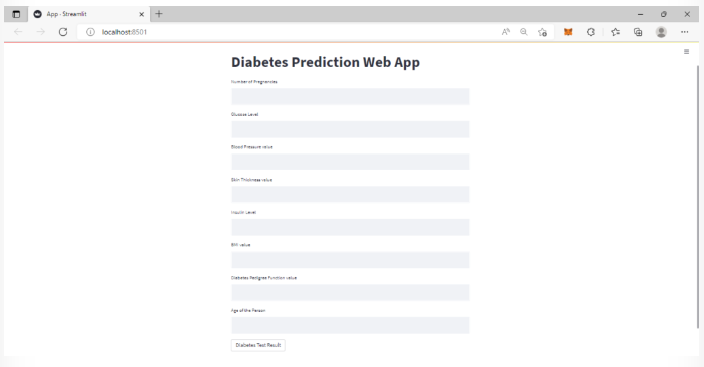
Save the file after pasting the code. And then to deploy using streamlit go to command prompt and run the following command.





After running the command the web app will open in the localhost webserver. Otherwise, go to your browser and type *localhost:8501*. The following output will be shown.

**Output**:



Sample Input data for a person does not have diabetes is {1, 85, 66, 29, 0, 26.6, 0.351, 31}. These data as input will generate the following output in the web app.



Sample input data for a person who have diabetes is {6, 148, 72, 35, 0, 33.6, 0.627, 50}. These data as input will generate the following output in the web app.



## **Conclusion**

We learned how to build a project on Diabetes Prediction using Machine Learning(with all 5 proper ML steps) and deploy it using Streamlit. Hope you enjoyed doing this project.