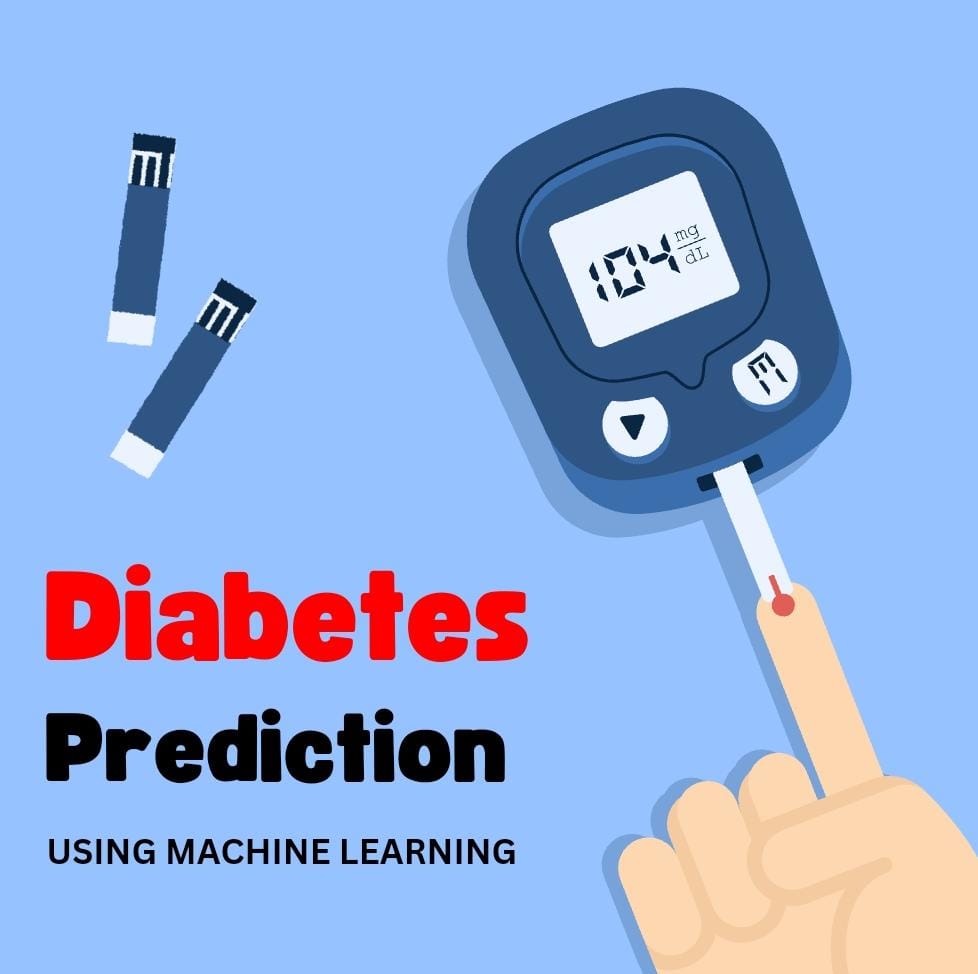
**DIABETES PREDICTION USING MACHINE LEARNING**

au620121243025 : KARTHIKA.S

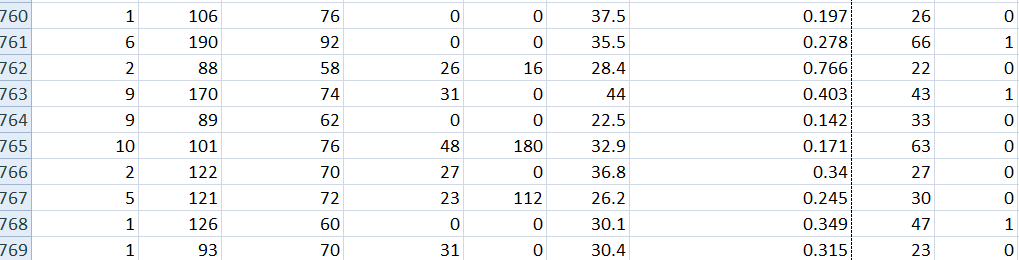
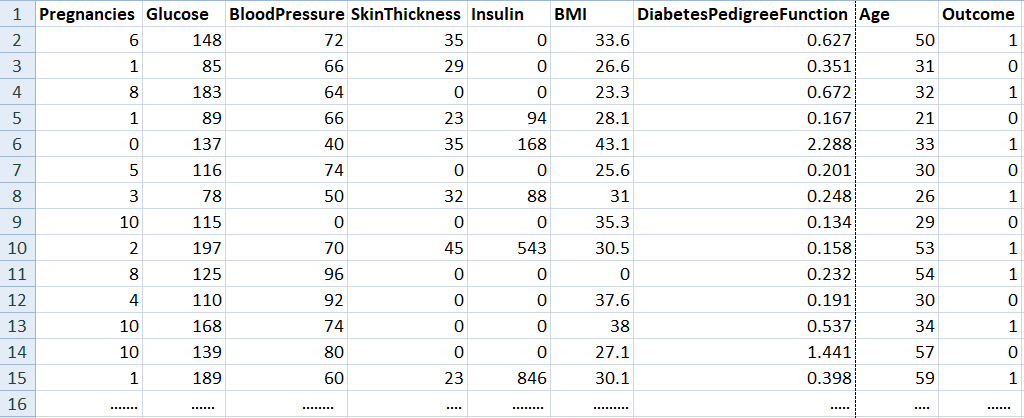
Phase 3 submission document



**Introduction**

* Diabetes is a chronic metabolic disorder characterized by elevated blood sugar levels, impacting millions of individuals worldwide. Early detection and management of diabetes are crucial for preventing complications and improving the quality of life for those affected. Machine learning (ML) offers a promising approach to predict the risk of diabetes based on various factors, providing valuable insights for healthcare professionals**.**
* Diabetes prediction using machine learning involves the application of computational techniques to analyze and interpret complex datasets related to patient health. Traditional methods often rely on manual analysis of risk factors, but ML algorithms can efficiently
* Process large amounts of data to identify patterns and relationships that might be challenging for human experts to discern.
* The primary objective of this project is to develop a predictive model that can assess the likelihood of individual developing diabetes based on specific health indicators. By leveraging machine learning, we aim to create a tool that can assist healthcare professionals in early diagnosis and risk stratification.
* Our analysis is based on a comprehensive dataset that includes various features such as age, body mass index (BMI), blood pressure, family history, and glucose levels. This dataset is collected from a diverse population to ensure the model's generalizability across different demographic groups.

**Using Data set**

****

**769 rows X 9 columns**

**Necessary step to follow:**

1. Import libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

from sklearn.metrics import accuracy\_score

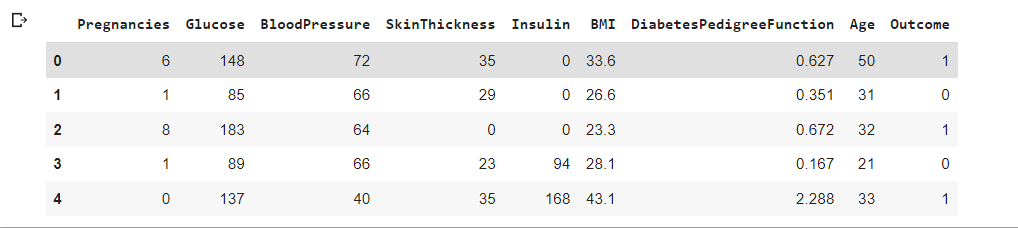
import pickle

# Load the diabetes dataset to a pandas DataFrame

diabetes\_dataset = pd.read\_csv('diabetes.csv')

# Print the first 5 rows of the dataset

diabetes\_dataset.head()



# To get the number of rows and columns in the dataset

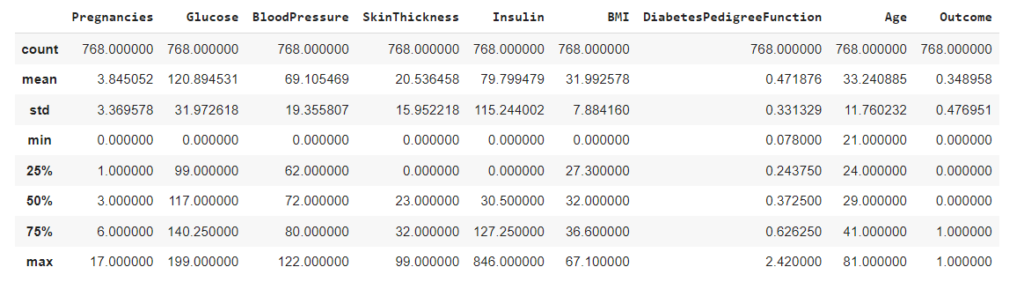
diabetes\_dataset.shape

#prints (768, 9)

# To get the statistical measures of the data

diabetes\_dataset.describe()

**Output:**

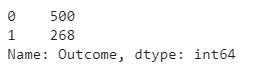


And, it is clear that the Outcome column is the output variable. So let us explore more details about that column.

# To get details of the outcome column

diabetes\_dataset['Outcome'].value\_counts()

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.

# separating the data and labels

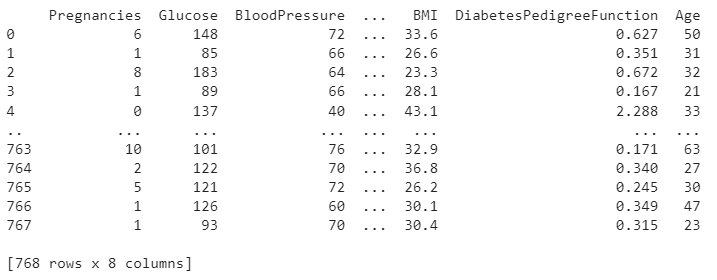
X = diabetes\_dataset.drop(columns = 'Outcome', axis=1)

Y = diabetes\_dataset['Outcome']

# To print the independent variables

print(X)

**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)

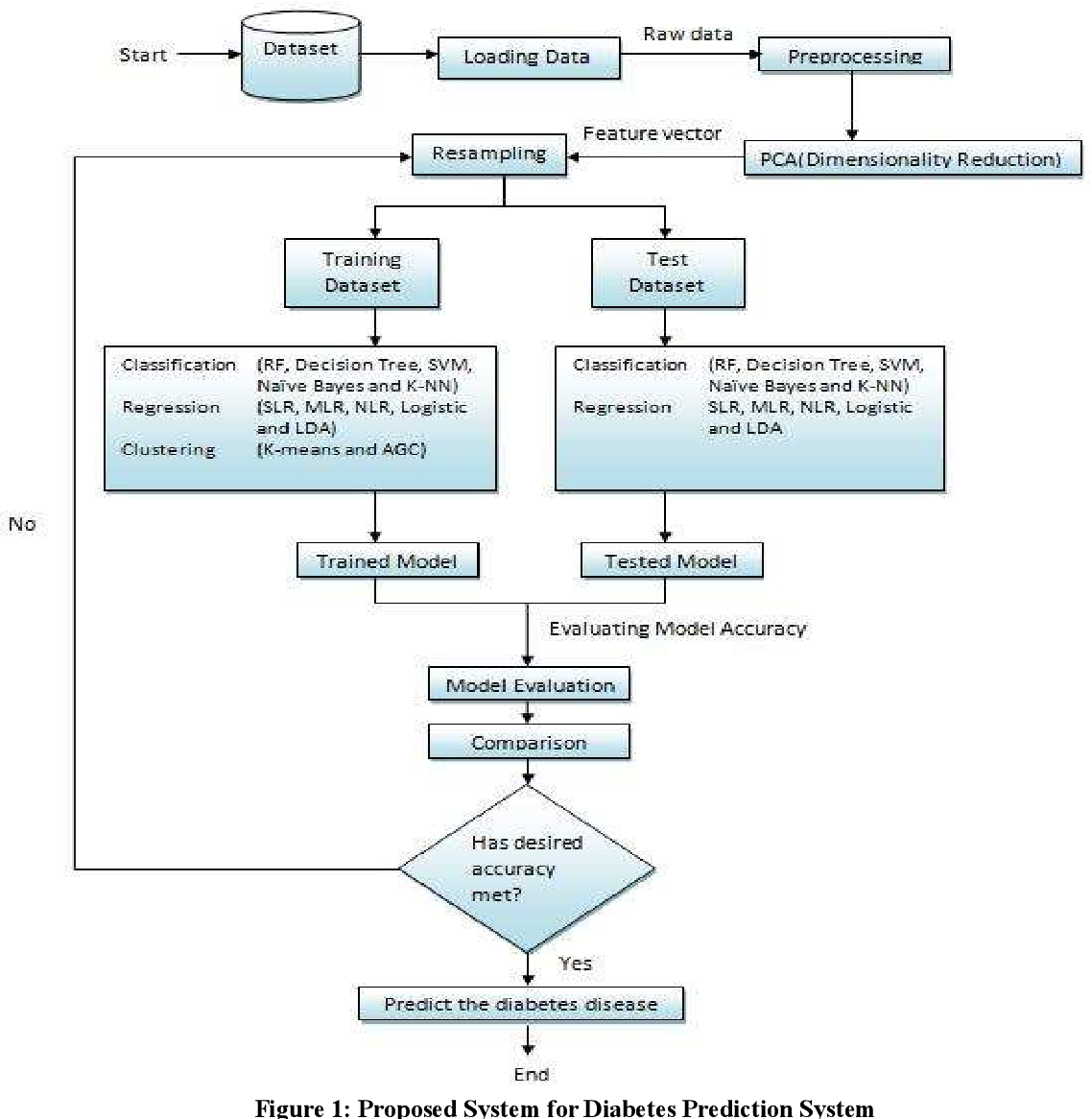
print(X.shape, X\_train.shape, X\_test.shape)

**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.

# Build the model

classifier = svm.SVC(kernel='linear')

# Train the support vector Machine Classifier

classifier.fit(X\_train, Y\_train)

# Accuracy score on the training data

X\_train\_prediction = classifier.predict(X\_train)

training\_data\_accuracy = accuracy\_score(X\_train\_prediction, Y\_train)

print('Accuracy score of the training data : ', training\_data\_accuracy)

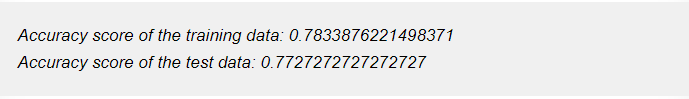
# Accuracy score on the test data

X\_test\_prediction = classifier.predict(X\_test)

test\_data\_accuracy = accuracy\_score(X\_test\_prediction, Y\_test)

print('Accuracy score of the test data : ', test\_data\_accuracy)

**Output:**



**5. Evaluating the model**

input\_data = (5,166,72,19,175,25.8,0.587,51)

# Change the input\_data to numpy array

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

# Reshape the array for one instance

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

prediction = classifier.predict(input\_data\_reshaped)

print(prediction)

if (prediction[0] == 0):

print('The person is not diabetic')

else:

print('The person is diabetic')

**Output**:

The person is diabetic

**Saving the file**

# Save the trained model

filename = 'trained\_model.sav'

pickle.dump(classifier, open(filename, 'wb'))

# Load the saved model

loaded\_model = pickle.load(open('trained\_model.sav', 'rb'))

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.

pip install streamlit

Open a new Python file and put the following code.

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\_\_':

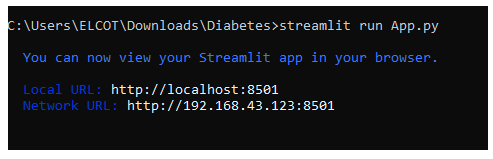
main()

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

streamlit run App.py

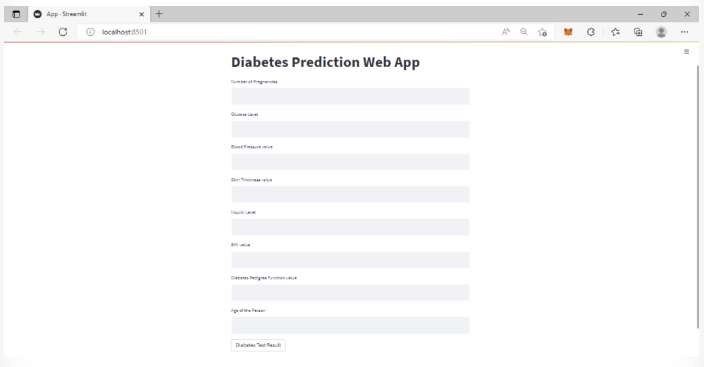
(or)

streamlit run filename.py



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