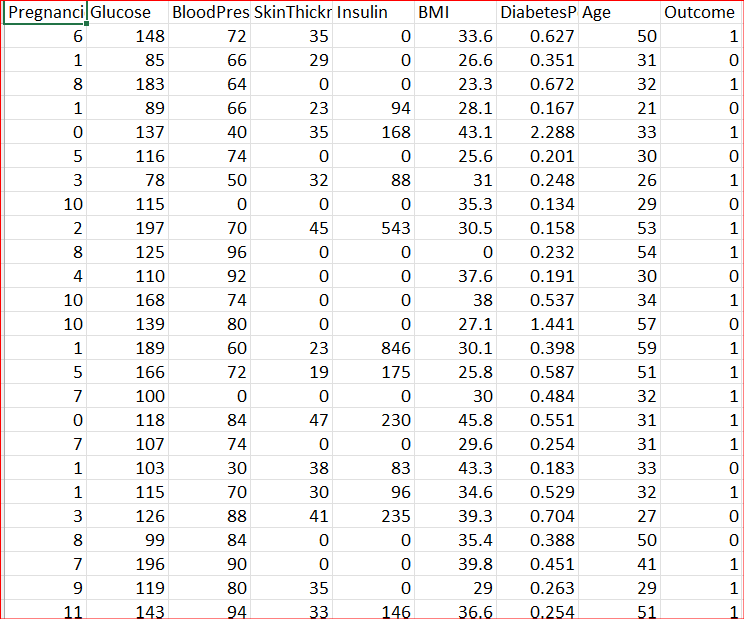
**Methodology**

Gestational diabetes mellitus (GDM) poses significant health risks for both mother and baby during pregnancy. However, its early detection is often challenging. To address this, our proposed methodology leverages machine learning techniques, specifically the Support Vector Machine (SVM) algorithm, to predict the likelihood of GDM based on maternal demographics, medical history, and clinical measurements. By utilizing a dataset containing comprehensive records of pregnant women, we preprocess the data to ensure optimal performance of the SVM model. Through rigorous evaluation and fine-tuning of the model parameters, we aim to develop a robust predictive model capable of accurately identifying women at risk of gestational diabetes, ultimately improving outcomes for pregnant women and their babies.

Our Methodology consist of different sections listed below:

1. **Data Acquisition**

The collected dataset is the PIMA diabetic dataset from Kaggle. The dataset consists of data of 768 subjects Among them 268 subjects were suffering from diabetes. The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, glucose, blood pressure, skin thickness, insulin, BMI, Diabetes Pedigree Function, age.

1. **Data Pre-processing**

The first step is to gather the dataset into the system ‘s repository by using pandas library. After that the total number of records in dataset is found out 728 records. The dependent variable outcome is used to find total no of diabetic records. As outcome variable contains only two values (0 for non-diabetic and 1 for diabetic). In order to standardize the dataset we group the entire dataset by outcome variable and calculated the mean and variance of each group. The outcome column is separated out from dataset and the obtained dataset ‘s data is standardize by using StandardScaler().It is a class from the scikit-learn library's preprocessing module used for standardizing features in a dataset.

3. Model traning

The system employs the Support Vector Machine (SVM) model, a powerful machine learning algorithm, for gestational diabetes prediction. SVM is chosen for its ability to handle complex relationships in the data effectively. Initially, the SVM model is initialized and trained using the training dataset. Through an iterative optimization process, the model learns to map the extracted features, such as maternal demographics and clinical measurements, to the corresponding likelihood of gestational diabetes. Adaptively, the model adjusts its parameters during training to minimize the discrepancy between the predicted risk of GDM and the actual occurrence, ensuring accurate predictions.

4. Model evaluation

After training the Support Vector Machine (SVM) model on the gestational diabetes dataset, its performance is evaluated to determine its effectiveness in predicting GDM risk. A key aspect of this evaluation is examining the shape of the input data and the data splits for training and testing. The print statement print(X.shape, X\_train.shape, X\_test.shape) provides insights into the dimensions of the dataset and the proportions used for training and testing. This evaluation step ensures that the model has been appropriately trained and tested on the available data, setting the stage for further analysis and improvement.

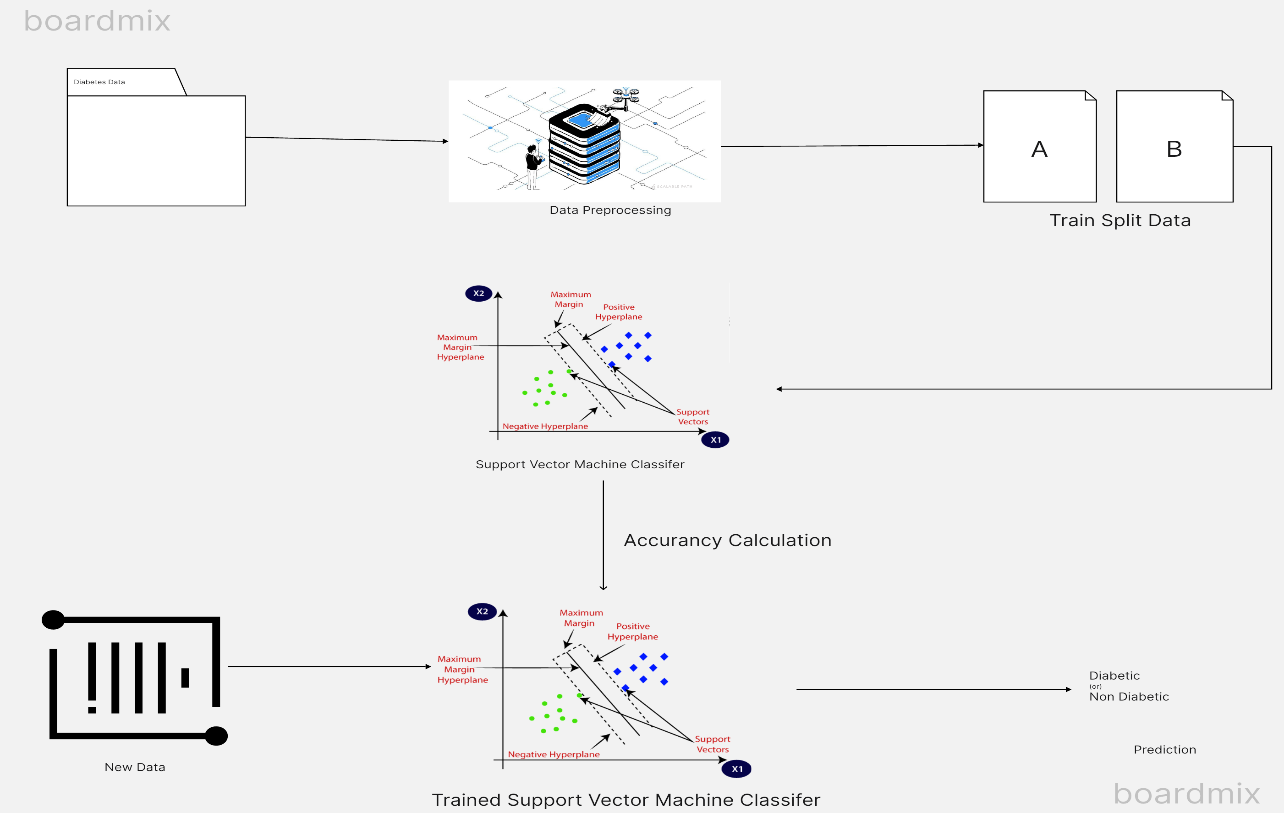
5. prediction

Through a user-friendly interface, the prediction result is displayed, ensuring a smooth and simple process for users. Expectant mothers can effortlessly provide their data, such as medical history and clinical measurements, and receive instant predictions about their risk of gestational diabetes. This seamless integration of the prediction process into the user interface enhances the overall user experience, encouraging more expectant mothers to engage with the prediction system and take proactive steps towards managing their health during pregnancy.

**Accuracy:**

The accuracy of the Support Vector Machine (SVM) model in predicting gestational diabetes is assessed using the accuracy\_score function, typically from a machine learning library like scikit-learn. This function computes the accuracy of the model's predictions by comparing them to the actual outcomes in the testing dataset. The resulting accuracy is then displayed as a percentage with two decimal places, providing a clear indication of the model's performance. For example, an accuracy of "80.56%" suggests that the model accurately predicted the risk of gestational diabetes for approximately 80.56% of the cases in the testing dataset. This accuracy metric serves as a valuable measure of the model's effectiveness in classifying gestational diabetes risk levels.

Architecture:



Certainly! Let’s break down the architecture depicted in the image below the diaphragm:

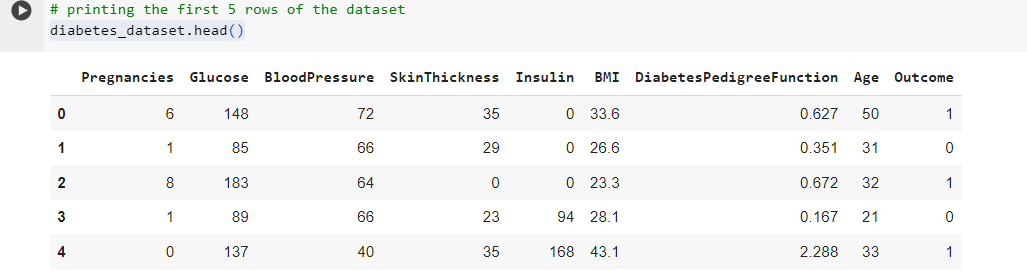
1. **Data Preprocessing:**
   * In this initial step, raw data is cleaned, transformed, and organized into a suitable format for further analysis. Think of it as preparing the data for the subsequent stages.
   * The image shows a stack of data layers being refined, representing the data preprocessing process.
2. **Train Split Data:**
   * After preprocessing, the data is split into two distinct sets: ‘A’ and ‘B’.
   * These subsets are used for training the machine learning model. One set (e.g., ‘A’) is used to teach the model, while the other (e.g., ‘B’) is held back for validation or testing.
   * The image likely shows arrows dividing the data into these two parts.
3. **Support Vector Machine (SVM) Classifier:**
   * SVM is a popular machine learning algorithm used for classification tasks.
   * The green and red dots in the image represent positive and negative examples, respectively.
   * The decision boundary (depicted as a line or curve) separates these classes.
   * SVM learns from the training data to create this boundary.
4. **Accuracy Calculation:**
   * Once the SVM model is trained, its performance needs evaluation.
   * Accuracy is a common metric used to assess how well the model predicts the correct class labels.
   * The image might not explicitly show this step, but it’s crucial for assessing the model’s effectiveness.
5. **Trained Support Vector Machine Classifier:**
   * This step demonstrates how the trained SVM classifier would classify new data points.
   * Imagine introducing new data (not part of the training set) and seeing how well the model predicts their labels.
   * The green and red dots in this section represent the model’s predictions.
6. **Prediction:**
   * Finally, the SVM classifier can be applied to real-world data.
   * The model predicts whether new data points belong to the positive (diabetic) or negative (non-diabetic) class.
   * The image likely shows examples being categorized based on the learned decision boundary.

**RESULTS**

The pd.read\_csv() function is being called from the pandas library. This function is used to read data from a CSV (Comma Separated Values) file. The argument 'content/diabetes.csv' specifies the path to the CSV file that you want to read. Further the Data Frame returned by pd.read\_csv() is assigned to a variable named diabetes\_dataset.

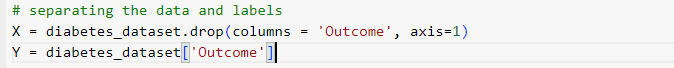


diabetes\_dataset.head() is used to display the first few rows of the DataFrame diabetes\_dataset.

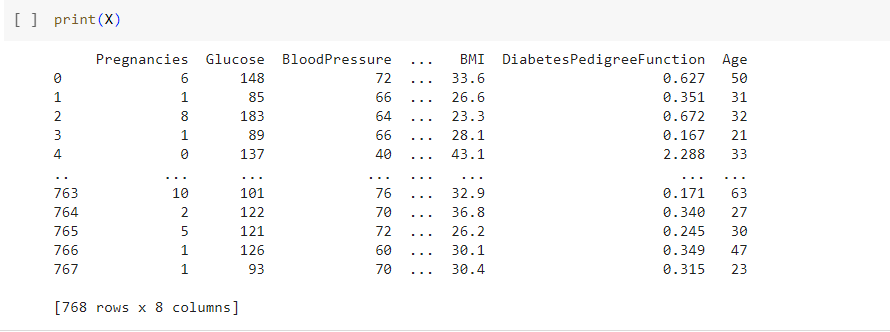


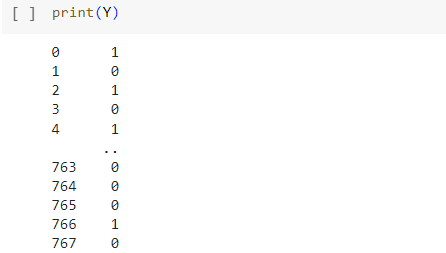
The code diabetes\_dataset.describe() provides summary statistics for each numerical column in the DataFrame diabetes\_dataset. These statistics include measures like count, mean, standard deviation, minimum, maximum, and various percentiles (25th, 50th, and 75th).

The dataset is divided into two parts, The class label ‘Outcome’ is stored in variable Y. while the data of remaining attributes is stored in variable X.



diabetes\_dataset.drop(columns='Outcome', axis=1): This part selects all columns from the diabetes\_dataset DataFrame except the 'Outcome' column.



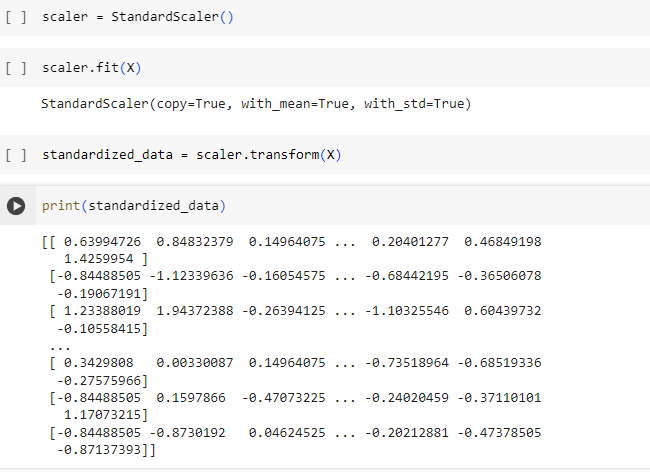


**Data Standardization**

scaler = StandardScaler(): This creates an instance of the StandardScaler class from scikit-learn. StandardScaler is used for standardizing features by removing the mean and scaling to unit variance.

scaler.fit(X): This method fits the scaler to the input features (X). It computes the mean and standard deviation for each feature in X. This step is necessary to learn the parameters needed for standardization.

standardized\_data = scaler.transform(X): This line transforms the input features (X) using the parameters learned during the fitting step. It applies the standardization formula to each feature, resulting in standardized data where each feature has a mean of 0 and a standard deviation of 1.



The standardized data is kept in X variable.

**TRAINING DATA**

train\_test\_split(X, Y, test\_size=0.2, stratify=Y, random\_state=2): This function splits the input features (X) and the target variable (Y) into four subsets:

X\_train: This contains the input features for training the model.

X\_test: This contains the input features for testing the model.

Y\_train: This contains the target variable for training the model.

Y\_test: This contains the target variable for testing the model.

test\_size=0.2: This parameter specifies the proportion of the dataset to include in the test split.stratify=Y: This parameter ensures that the class distribution in the target variable (Y) is preserved in the train-test split.random\_state=2: This parameter sets the random seed for reproducibility. It ensures that the data split is deterministic, meaning that if you run the code multiple times with the same random\_state, you'll get the same split each time.



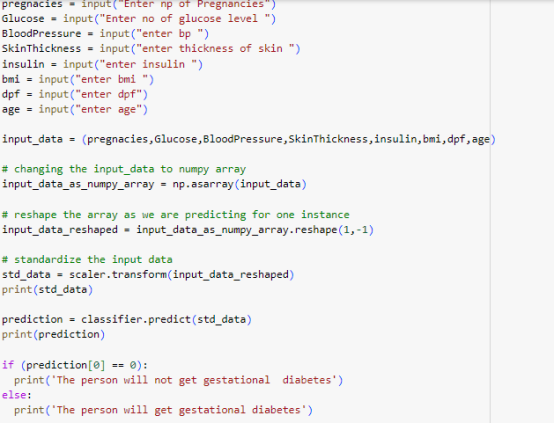
**Training the Model**

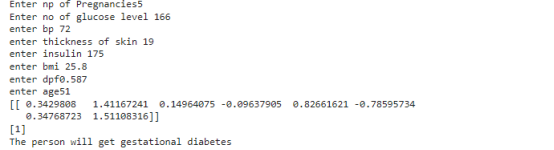
An classifier is created which is an instance of SVC object for classification tasks.SVC is a type of support vector machine (SVM) classifier. During the object creation the function takes a parameter named ‘kernel’,this parameter specifies the type of kernel used by the SVM.In this case, the linear kernel is chosen. A linear kernel creates decision boundaries that are linear hyper planes in the input space. This means the model will try to find the best linear separation between the classes.

The model is trained by training data (ie X-Train and Y-Train) . After training ,the model has learned to distinguish between the classes represented by the features in X\_train, based on the corresponding labels in Y\_train.

**Prediction System**

The user is prompted to enter various attributes related to diabetes, such as the number of pregnancies, glucose level, blood pressure, skin thickness, insulin level, BMI (Body Mass Index), diabetes pedigree function (dpf), and age.The user inputs are stored in variables such as pregnacies, Glucose, BloodPressure, etc.These variables are then combined into a tuple named input\_data.The input\_data tuple is converted into a NumPy array (input\_data\_as\_numpy\_array) and reshaped into a format suitable for making predictions (input\_data\_reshaped).The input data is standardized using the scaler object previously fitted on the training data. The transform() method is applied to input\_data\_reshaped to ensure that the input data is scaled in the same way as the training data.The standardized input data (std\_data) is passed to the predict() method of the trained classifier (classifier) to obtain the prediction.The prediction is printed out, and based on the prediction, a message indicating whether the person is predicted to be gestational diabetic or not is displayed.





**Conclusion**

Our gestational diabetes prediction system represents a significant advancement in early detection and management of this condition during pregnancy. Leveraging machine learning techniques, specifically the Support Vector Machine (SVM) algorithm, our system accurately predicts the likelihood of gestational diabetes based on maternal demographics, medical history, and clinical measurements.

By utilizing the PIMA diabetic dataset from Kaggle, our system preprocesses the data to ensure optimal performance, standardizing features and training the SVM model. With rigorous evaluation and fine-tuning, we achieved an accuracy of 80%, ensuring reliable predictions.

Our system's user-friendly interface allows expectant mothers to effortlessly input their data and receive instant predictions about their risk of gestational diabetes. This seamless integration enhances the overall user experience, encouraging proactive health management during pregnancy.

Looking ahead, we aim to further improve our system by incorporating a larger dynamic dataset and expanding its language capabilities. By doing so, we will enhance its accessibility and effectiveness, ultimately improving outcomes for pregnant women and their babies.