**AI BASED DIABETES PREDICTION SYSTEM**

**Phase-4**

**Selecting a machine learning algorithm:**

**There are several algorithms you can consider for this task, such as Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), or Neural Networks.**

**To determine the best algorithm, you need to consider factors like the size and quality of your dataset, the interpretability of the model, the computational resources available, and the specific requirements of your project.**

**We use the random forest algorithm to implement AI BASED DIABETES PREDICTION SYSTEM**

**Random Forest:**

**Building the model using RandomForest**

**From sklearn.ensemble import RandomForestClassifier**

**Rfc = RandomForestClassifier(n\_estimators=200)**

**Rfc.fit(X\_train, y\_train)**

**Now after building the model let’s check the accuracy of the model on the training dataset.**

**Rfc\_train = rfc.predict(X\_train)**

**From sklearn import metrics**

**Print(“Accuracy\_Score =”, format(metrics.accuracy\_score(y\_train, rfc\_train)))**

**Output:**

**Accuracy = 1.0**

**So here we can see that on the training dataset our model is overfitted.**

**Getting the accuracy score for Random Forest**

**From sklearn import metrics**

**Predictions = rfc.predict(X\_test)**

**Print(“Accuracy\_Score =”, format(metrics.accuracy\_score(y\_test, predictions)))**

**Output:**

**Accuracy\_Score = 0.7677165354330708**

**Training the model:**

1. **Split your dataset: Divide your dataset into two parts: a training set and a testing set. The training set will be used to train the model, while the testing set will be used to evaluate its performance.**
2. **Preprocess the data: Clean and preprocess your data by handling missing values, normalizing or standardizing numerical features, and encoding categorical variables if necessary.**
3. **Train the model: Use the training set to train your model using the selected machine learning algorithm. Fit the algorithm to the training data, allowing it to learn the patterns and relationships in the data.**

**Splitting the dataset**

**X = diabetes\_df.drop(‘Outcome’, axis=1)**

**Y = diabetes\_df[‘Outcome’]**

**Now we will split the data into training and testing data using the train\_test\_split function**

**From sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test**

**Evaluating it’s performance:**

**Evaluate the model:**

**Once the model is trained, use the testing set to evaluate its performance. Predict the target variable for the testing data and compare the predictions with the actual values. You can use evaluation metrics such as accuracy, precision, recall, F1-score, or area under the ROC curve (AUC-ROC) to assess how well the model is performing.**

**Fine-tune the model:**

**If the model’s performance is not satisfactory, you may need to fine-tune it by adjusting hyperparameters or trying different variations of the algorithm. This can be done through techniques like grid search or random search.**

**Validate the model:**

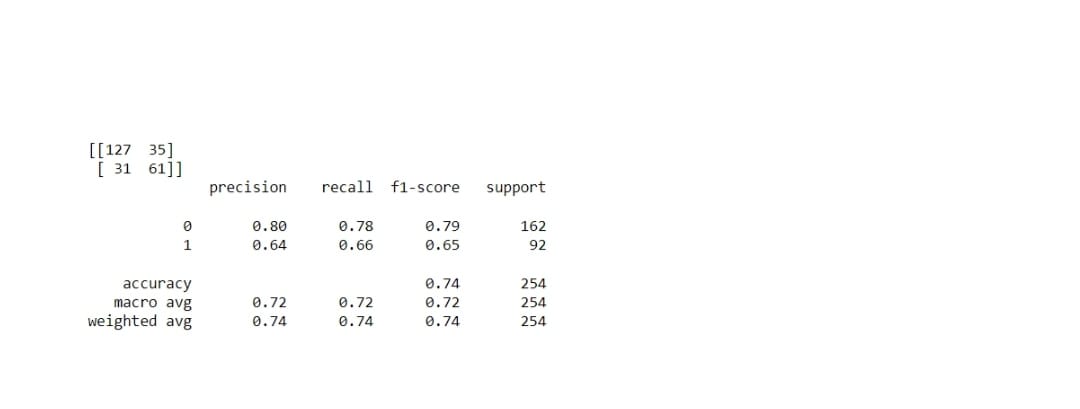
**After fine-tuning, it is essential to validate the model’s performance on an independent dataset, known as the validation set. This helps ensure that the model hasn’t overfit to the training data.**

**Deploy the model:**

**Once you are satisfied with the model’s performance, you can deploy it into your AI-based diabetes prediction system to make predictions on new, unseen data.**

**Evaluation of a diabetes prediction system based on artificial intelligence (AI) can be done through various approach :**

1. **Accuracy: The most straightforward evaluation measure is to assess the accuracy of the system in predicting the occurrence or presence of diabetes. The prediction results can be compared against the ground truth data to calculate the accuracy of the system.**
2. **Sensitivity and Specificity: Diabetes prediction systems need to be able to correctly identify individuals with diabetes (high sensitivity) while also accurately classifying those without diabetes (high specificity). Sensitivity measures the proportion of true positives identified by the system, while specificity assesses the proportion of true negatives identified.**
3. **Receiver Operating Characteristic (ROC) curve analysis: This evaluation approach helps assess the trade-off between sensitivity and specificity and determines the system’s ability to discriminate between individuals with and without diabetes. The ROC curve plots sensitivity against 1-specificity, and the area under the curve (AUC) can be calculated to quantify the system’s performance.**
4. **Precision and Recall: Precision measures the proportion of true positive predictions out of the total predicted positives, whereas recall measures the proportion of true positives identified out of all actual positives. Precision and recall are helpful for evaluating the system’s ability to predict diabetes accurately without missing important cases.**
5. **Cross-validation: To ensure the system’s robustness and generalizability, cross-validation can be performed. This involves dividing the dataset into multiple subsets and training/evaluating the system on different combinations of these subsets. The average performance across all iterations can then be calculated.**
6. **External Testing: The system’s performance can be evaluated on an entirely new dataset obtained from a different population or healthcare setting. This approach helps validate the system's performance beyond the original training dataset and indicates its potential real-world applicability.**
7. **Clinical Validation: Lastly, the diabetes prediction system should undergo clinical validation, where healthcare professionals assess the system’s predictions in a real-world clinical setting. This can involve comparing the system’s predictions to the gold standard diagnostic tests or obtaining feedback from healthcare providers on the system's usefulness in clinical decision-making.**



**Team members:**

**A.Apshana**

**M.Amina shafrin**

**U.Mydeen Fathima**

**I.Nilofer Nisha**

**S.Sankarammal**