DIABETES PREDICTION USING MACHINE LEARNING

A PROJECT REPORT
Submitted by

SULTANA KHATUN, ROLL NO- 390
ENROLLMENT NO- 12023054006394 OF 2023-26
SWARNA MAJUMDER, ROLL- 400
ENROLLMENT NO- 12023054006405 OF 2023-26
UNMILITA GHOSHAL, ROLL- 417
ENROLLMENT NO- 12023054006422 OF 2023-26

In fulfilment of the requirements for the award of the degree of
BACHELOR OF COMPUTER APPLICATIONS
IN
COMPUTER APPLICATION

IEM
श्रद्धावान लभते ज्ञानम्
INSTITUTE OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

INSTITUTE OF ENGINEERING & MANAGEMENT,
SECTOR-V, KOLKATA - 700091

ACKNOWLEDGEMENT

We feel immense pleasure to introduce " Diabetes Prediction using Machine Learning" as our project.

We would like to express our special thanks to our teacher **PARTHA KOLEY** Sir who has been a constant source of knowledge and inspiration to us, and who gave us the opportunity to do this project. We would also like to express our gratitude to our beloved parents for their review and many helpful comments and enlightening us and guiding us throughout the finalization of this project within the limited time frame.

Last but not the least, we thank all our teachers and as well as friends who have given us that much strength to keep moving on forward every time. We are greatly thankful to one and all and have no words to express our gratitude to them.

Name of the Students:

- SULTANA KHATUN
- SWARNA MAJUMDER
- UNMILITA GHOSHAL

SIGNATURE OF THE STUDENTS:

SELF CERTIFICATE

I hereby certify that the work presented in this **Machine Learning Project on Diabetes Prediction** is original and was carried out by me. All tasks, including data collection, preprocessing, model implementation, evaluation, and reporting, were completed in dependently. I acknowledge the sources used in the project and confirm that no unauthorized assistance was obtained during its completion.

NAME: SULTANA KHATUN
ROLL NO: 390
ENROLLMENT NO: 12023054006394 OF 2023-26

SELF CERTIFICATE

I hereby certify that the work presented in this **Machine Learning Project on Diabetes Prediction** is original and was carried out by me. All tasks, including data collection, preprocessing, model implementation, evaluation, and reporting, were completed in dependently. I acknowledge the sources used in the project and confirm that no unauthorized assistance was obtained during its completion.

NAME: SWARNA MAJUMDER
ROLL NO: 400
ENROLLMENT NO: 12023054006405 OF 2023-26

SELF CERTIFICATE

I hereby certify that the work presented in this **Machine Learning Project on Diabetes Prediction** is original and was carried out by me. All tasks, including data collection, preprocessing, model implementation, evaluation, and reporting, were completed in dependently. I acknowledge the sources used in the project and confirm that no unauthorized assistance was obtained during its completion.

NAME: UNMILITA GHOSHAL
ROLL NO: 417
ENROLLMENT NO: 12023054006422 OF 2023-26

ABSTRACT

Diabetes prediction refers to the use of data-driven models, often powered by machine learning (ML) and Artificial intelligence (AI), to forecast the likelihood of an individual developing diabetes. These models analyze various factors such as age, body mass index (BMI), blood glucose levels, family history, and lifestyle choices to identify patterns and risk factors associated with diabetes onset. By predicting the risk, healthcare providers can implement preventive measures and personalized interventions to manage or delay the disease's progression.

Diabetes mellitus, particularly Type 2 diabetes, poses a significant global health challenge, necessitating early detection and intervention. Machine learning (ML) offers promising avenues for predicting diabetes onset by analyzing clinical and demographic data. This study evaluates various ML algorithms, including Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, and Neural Networks, to develop predictive models for diabetes diagnosis. The models were trained on datasets such as the Pima Indians Diabetes Database, encompassing attributes like age, BMI, glucoselevels, and blood pressure. Performance metrics such as accuracy, precision, recall, and F1-score were utilized to evaluate model efficacy.

Additionally, data preprocessing techniques like Synthetic Minority Over-sampling Technique (SMOTE) were employed to address class imbalance. The findings indicate that ensemble methods, particularly Random Forest, achieved high accuracy rates, with some models reaching up to 97.33% accuracy and an AUC of 0.975. Furthermore, integrating Explainable AI (XAI) tools, such as SHAP and LIME, enhanced model interpretability, allowing healthcare professionals to understand the decision-making process behind predictions. This research underscores the potential of ML in developing transparent and accurate tools for early diabetes detection, facilitating timely interventions and improved patient outcomes.

TABLE OF CONTENTS

INTRODUCTION

Diabetes mellitus, particularly Type 2 diabetes, has become a significant global health concern due to its increasing prevalence and associated complications such as cardiovascular diseases, kidney failure, and nerve damage. Early detection and intervention are crucial to managing the disease effectively and preventing long-term health issues.

Machine learning (ML) has emerged as a powerful tool in the healthcare domain, enabling the development of predictive models that can identify individuals at risk of developing diabetes. By analysing complex datasets containing various health parameters, ML algorithms can uncover patterns and relationships that may not be immediately apparent through traditional statistical methods.

OBJECTIVES

Early Detection: Allow health professionals to detect high-risk people as early as possible.

Cost-Effective Screening: Reduce the need for invasive and expensive tests by providing accurate predictions using non-invasive inputs.

Improved Decision-Making: Empower clinicians with data-driven insights to enhance diagnostic and treatment decisions.

Approach:

The project involves building a machine learning model trained on a dataset containing various health-related features, such as:

- Age
- Gender
- Glucose
- Blood Pressure
- Skin Thickness
- Insulin
- Body Mass Index (BMI)
- Pregnancies

SIGNIFICANCE

This proposed system is bridging the gap between traditional healthcare and modern technology, as it uses machine learning to save lives. A prediction model on the chances of Diabetes Prediction causing death will support proactive approaches to healthcare at the social level.

KEYWORDS

- Diabetes Prediction
- Machine Learning (ML)
- Prediction Model
- Healthcare Analytics
- Data-Driven Insights
- Risk Assessment

HEADINGS

- To use ML Techniques to predict diabetetes Prediction
- To provide an easy to use User interface.
- To increase the accuracy of heart condition disease and analyse different parameters

LITERATURE SURVEY

Several studies emphasize the significance of early detection in reducing the mortality and morbidity associated with Diabetes prediction. Diabetes prediction leverages machine learning algorithms like Logistic Regression, SVM, Random Forest, and Neural Networks to identify individuals at risk. Studies indicate that Neural Networks achieve up to 78.57% accuracy, while Random Forests reach 76.30% accuracy in early detection. These models utilize datasets such as the Pima Indians Diabetes Database to predict diabetes onset effectively.

- **Reference:** **WHO Urgent Action Needed as Global Diabetes Cases Increase (2024)**

MATERIALS

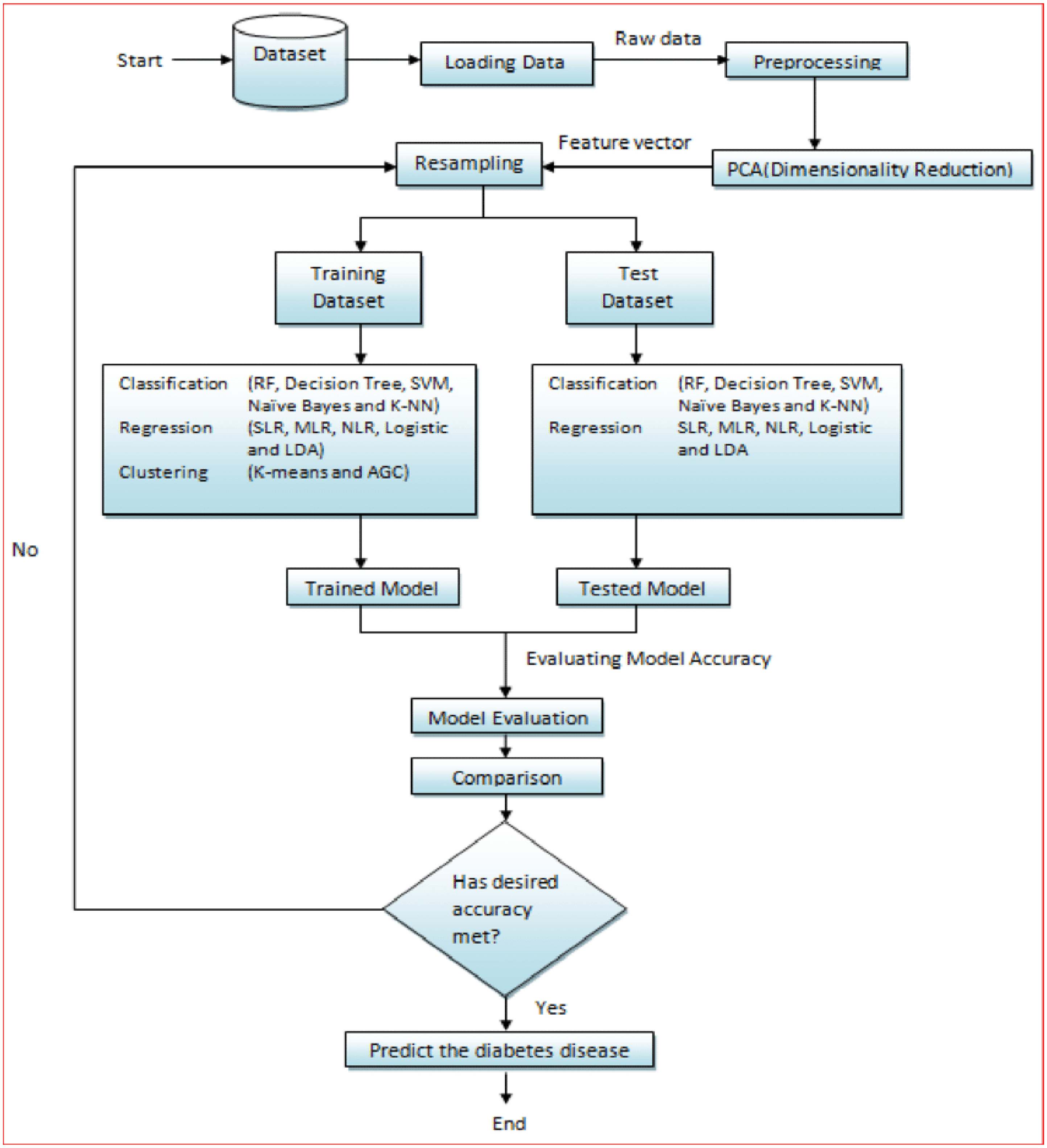
The data collection procedure refers to the programmer acquiring, and quantifying information based on variables relevant to the research. The data used was obtained from a variety of sources, including some official websites of the Government. The data of the state’s most harvested Diabetes prediction, which include wheat, rapeseed & mustard, barley, bajra, jowar onion, and Maize, have been gathered and recorded from the state’s 33 districts (Table 1). The data from 1997 to 2019 was obtained from the official Rajasthan Government website.

DATASET

Diabetes Prediction

CREDENTIALS
Age
Sex
Blood Pressure
Plasma Glucose
Pregnancies
Triceps Thickness
Serum Insulin
BMI
Diabetes Pedigree

PROPOSED SYSTEM


Start
Dataset
Loading Data
Raw data
Preprocessing
PCA(Dimensionality Reduction)
Feature vector
Resampling
Training Dataset
Test Dataset
Classification (RF, Decision Tree, SVM, Naïve Bayes and K-NN)
Regression (SLR, MLR, NLR, Logistic and LDA)
Clustering (K-means and AGC)
Classification (RF, Decision Tree, SVM, Naïve Bayes and K-NN)
Regression SLR, MLR, NLR, Logistic and LDA
Trained Model
Tested Model
Evaluating Model Accuracy
Model Evaluation
Comparison
Has desired accuracy met?
No
Yes
Predict the diabetes disease
End

The system employs a structured approach to predict diabetes, integrating various machine

learning techniques and data processing steps.

1. Data Collection

- **Sources:** Collect data from medical records, health surveys, and wearable devices.
- **Features:**Include attributes like age, BMI, glucose levels, insulin levels, blood pressure and family history.

2. Data Preprocessing

- **Cleaning:** Handle missing values and remove duplicates.
- **Normalization:** Scale numerical features to a standard range.
- **Encoding:** Convert categorical variables into numerical formats.

3. Feature Selection

- **Techniques:** Apply methods like Recursive Feature Elimination (RFE) or Principal Component Analysis (PCA) to identify the most relevant features.

4. Model Selection

- **Algorithms:** Evaluate various machine learning models such as Logistic Regression, Decision Trees,

Random Forest, Support Vector Machines (SVM), and Neural Networks.

5. Model Training

- **Training:** Split the dataset into training and testing subsets.
- **Validation:** Use techniques like k-fold cross-validation to assess model performance.

6. Model Evaluation

- **Metrics:** Assess models using accuracy, precision, recall, F1-score, and Area Under the Curve (AUC).
- **Comparison:** Compare performance across different models to select the best one.

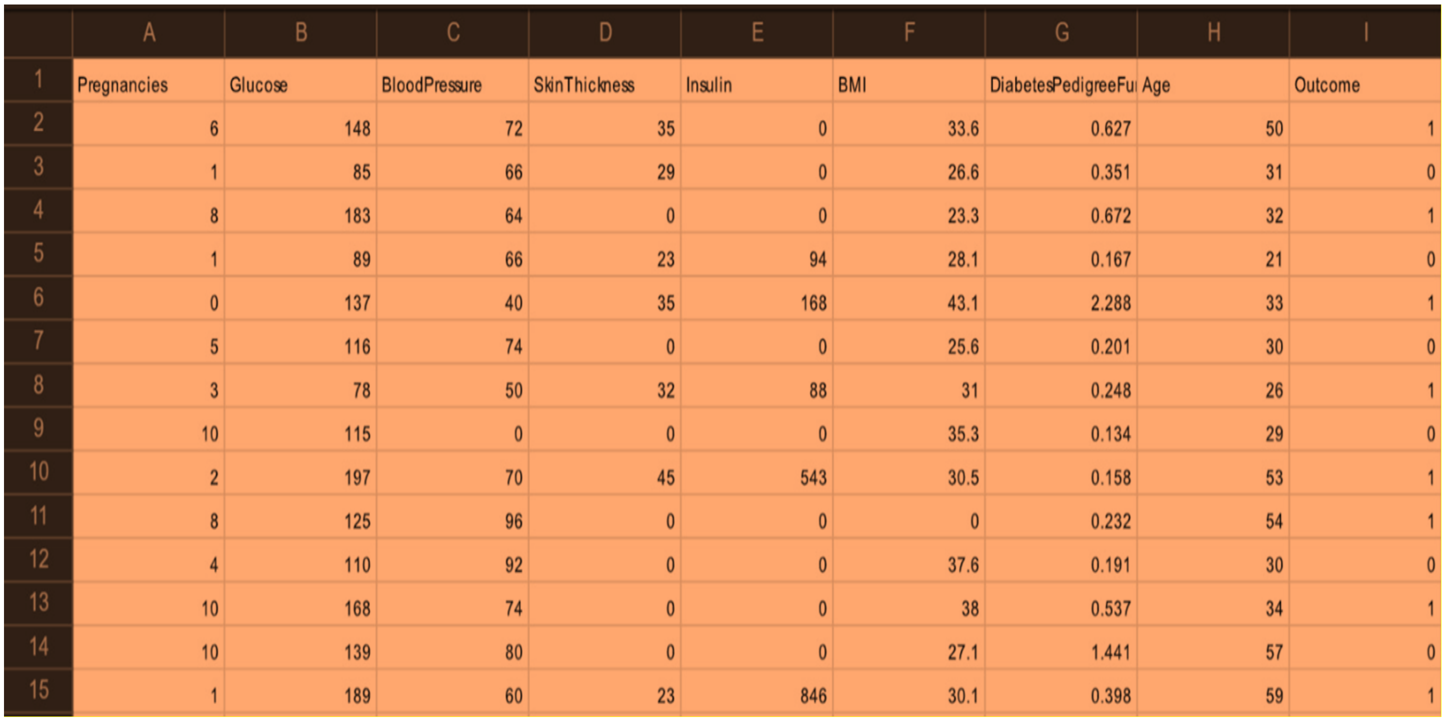
7. Deployment

- **Integration:** Implement the chosen model into a clinical decision support system or mobile application.
- **Monitoring:** Regularly update the model with new data to maintain accuracy.

The Diabetes Prediction and every step is finished we move on out final phase i.e. predicting the crop

PROPOSED SYSTEM FOLLOWS:

Collection of Data



	A	B	C	D	E	F	G	H	I
1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1

Analysing of Data: Analyzing data for diabetes prediction is a crucial preparatory step for machine learning. It begins by understanding the dataset's structure, examining column types, and getting initial insights from descriptive statics.

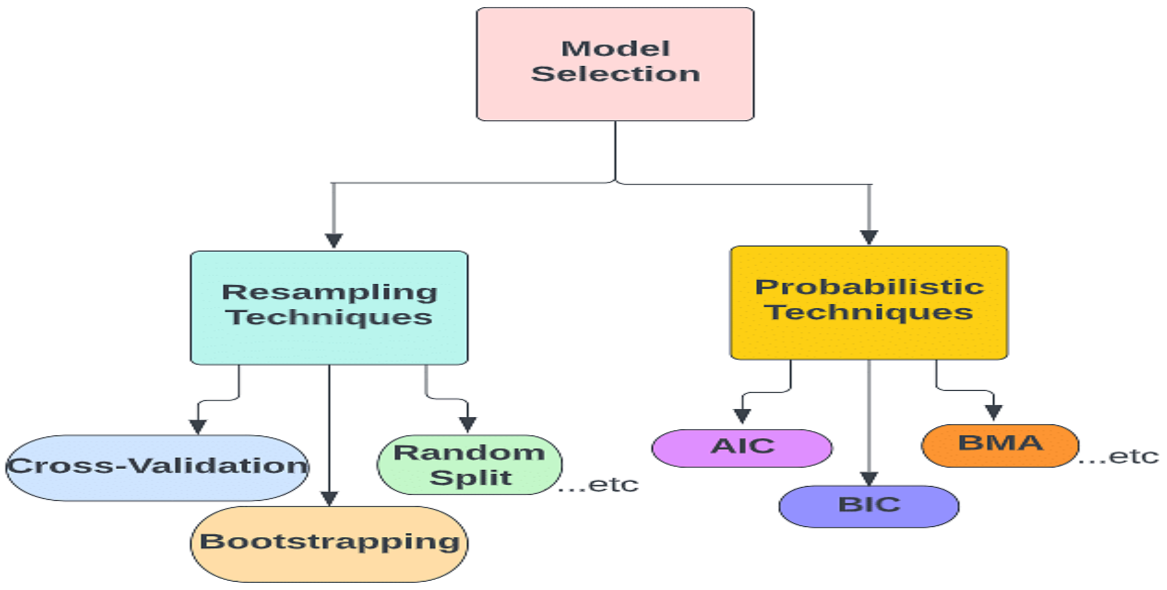
A critical phase is handling missing values, as common medical datasets often encode '0' for unrecorded measurements in fields like Glucose or BMI. Those zeros must be replaced to prevent skewed models. Outlier detection ensures extreme, potentially erroneous values don't distort analysis.

DATA PREPROCESSING: Firstly, handling missing values is paramount. Often, datasets contain gaps or placeholder values which need to be imputed or removed. Secondly, outlier detection and treatment are performed to identify and manage extreme data points that could negatively impact model performance. These might be capped, transformed, or removed if deemed erroneous. Thirdly, features contribute equally to the model, preventing features with larger ranges from dominating. Finally, encoding categorical variables converts non numeric data into a numerical format. This comprehensive preprocessing ensures data quality, improves model accuracy, and prevents issues during the learning phase.

FEATURE SCALING: We can change the data so that each feature is on a comparable scale by using feature scaling. Several techniques, including normalization, standardization, and min-max scaling, can be used to accomplish this.We can increase the data's suitability for machine accuracy and dependability by scaling the features.

After Data Preprocessing we have to scale the dataset into two parts: training and testing data which help to find out the accuracy of the model.

MODEL SELECTION:



Model selection in machine learning is the process of choosing the most appropriate algorithm from a set of   candidates based on its performance on a given dataset. This involves evaluating models using techniques like cross-validation and metrics such as accuracy, precision, recall, or mean squared error, depending on the task (classification or regression). The goal is to select a model that generalizes well to unseen data, balancing complexity and performance.

DIABETES PREDICTION: The system will suggest the best crop for cultivation based on the amount of anticipated rainfall, the composition of the soil, and weather parameters. This approach also displays the amount of seed needed to cultivate a recommended crop in parts per million.
 weather parameters. This approach also is plays the amount of seed needed to cultivate a recommended crop in parts per million. Confusion Matrix and Classification Report Confusion Matrix and Classification Report are the methods imported from the metrics module in the scikit learn library that are calculate using the actual labels of test datasets and predicted values. Confusion Matrix gives the matrix of frequency of true negatives, false negatives, true positives and false positives. Classification Report is a metric used for evaluating the performance of a classification algorithm's predictions. It gives three things: Precision, Recall and f1-score of the model. Precision refers to a classifier's ability to identify the number of positive predictions which are relatively correct. It is calculated as the ratio of true positives to the sum of true and false positives for each class.

It gives three things: Precision, Recall and f1-score of the model.

Precision refers to a classifier's ability to identify the number of positive predictions which are relatively correct. It is calculated as the ratio of true positives to the sum of true and false positives for each class.

$$Precision = \frac{TP}{TP + FP}$$

Where Precision-Positive Prediction Accuracy; TP-True Positive; FP-False Positive.

Recall is the capability of a classifier to discover all positive cases from the confusion matrix. It is calculated as the ratio of true positives to the sum of true positives and false negatives for each class.

$$Recall = \frac{TP}{TP + FN}$$

Where Recall- The percentage of positives that were correctly identified; FN-False Negative. F1 score is a weighted harmonic mean of precision and recall, with 0.0 being the worst and 1.0 being the best. Since precision and recall are used in the computation, F1 scores are often lower than accuracy

$$\text{F1 score} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

Where P-Precision; R-Recall Accuracy : The number of correct predictions divided by the total number of predictions accuracy. The accuracy of the model is calculated using the accuracy_score method of scikit learn module

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})}$$

Where TP-True Positive; FP-False Positive; TN: True Negative; FN: False Negative Proposed Work

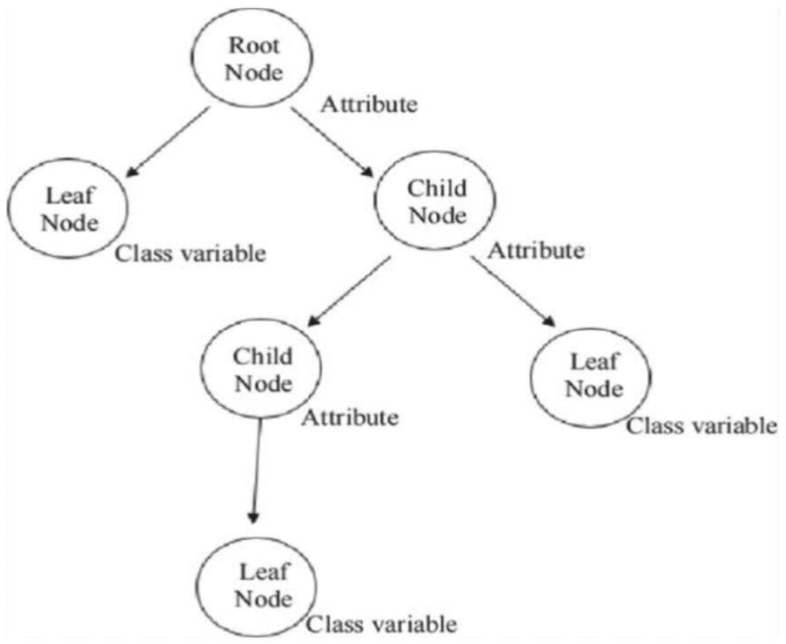
K-NEAREST NEIGHBOUR CLASSIFIER: The K-Nearest Neighbour algorithm is based on the supervised learning technique and is simple machine learning algorithm. The K-NN ◦ technique saves al possible and classifies the incoming data depending on how similar they are to the actual data. This means that the K-NN technique can swiftly classify new instances into a precisely defined category. The KNN technique can be used in both regression and classification problems but it is most likely to be used in classification. KNN technique has two properties. First, the model is based on the dataset or, it is not required to identify parameters for the distribution. Hence it is referred to as non-parametric. Second, there is no learning taking place; instead, it just stores the training data. The classification of the dataset happens during the testing phase, due to which the testing phase becomes time-consuming and takes a lot of memory. This property is known as lazy-learner.

Step 1: K, i.e., the number of neighbours is selected. The primary deciding factor is the number of neighbours.
Step 2: Using distance measures, determine the distance between two points like Euclidean Distance.

$$Euclidean\ distance = d(b,a) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$$

Step 3: K-nearest neighbours are taken into account according to the calculated Euclidean distance.
Step 4: Figure the number of data points in each class surrounded by these
Step 5: The class with the highest number of neighbours is assigned to the new data points.
Step 6: The label is voted on, and the model is ready.

DECISION TREE CLASSIFIER: Decision Tree is a supervised learning technique used where each path is a set of decision instances from the training set. The non attributes. The decision is made by comparing the instance with the split and jumping to the next node. This splitting continues, generating sub nodes which determine class labels for that instance. It divides recursive partitioning. With high accuracy, decision trees are capable of handling high data. It's a flowchart diagram-style representation that closely parallels human result, decision trees are simple to explain and apprehend.



Step 1: <u>Root Node:</u>

- The "Root Node" would represent the initial set of all patient data.
- The first "Attribute" would be a key feature (e.g., Blood Glucose Level, BMI, Age, Insulin Level, etc.) that the decision tree identifies as the most significant for splitting the data,

Step 2: <u>Child Nodes (Internal Nodes):</u>

- If the root node splits based on an attribute (e.g., "Is Blood Glucose Level > 140 mg/ dL?"), it would lead to "Child Nodes."
- Each "Child Node' represents a subset of the data based on the outcome of the previous attribute's test.
- These child nodes can further split based on other "Attributes" (e.g., "Is BMI > 30?" "Is Age > 50?"). This process continues, forming branches.

Step 3: <u>Leaf Nodes (Terminal Nodes):</u>

- The "Leaf Nodes" are the final outcomes of the classification. In the context of diabetes prediction, these "Leaf Nodes' would represent the predicted "Class variable":

- "Diabetic"'
- "Non-Diabetic"

• Each path from the root node to a leaf node represents a set of rules or conditions that lead to a specific prediction.

Steps to Split.

The dataset used in the project has with the numerical values in the following ways:
Step 1: Sorting all the values.
Step 2: It will consider a threshold value from
Step 3: Feature value will split into two parts such that threshold value, and the right node contains feature values greater than
Step 4: The next feature value will be considered as a threshold value and again create the same split
Step 5: Entropy/Gini and Information Gain are split with better information gain is considered.

$$I(Attribute) = \frac{\sum p_i + n_i}{p + n}$$

Where pi denotes the number of yes values; n attribute; p and n are the numbers of yeses and noes of the entire sample, respectively.

$$Information\ Gain = Entropy(S) - I(Attribute)$$

Where Entropy(S) denotes entropy of sample S; I(Attribute) denotes Average Information of the particular attribute.

Step 6: Repeat from Step 2 to
Step 5. In this way it will get branches for the decision tree.

ENTROPY AND GINI INDEX: The criteria for measuring Information Gain are the Gini index and Entropy. Information gain is a measurement of how much the reduction in entropy. Entropy and Gini Index are the metrics that measure the impurity of the nodes. A node is considered as impure if it has multiple classes, else Entropy is a metric that gives the degree of impurity in a specified attribute. The following formula can be used to compute entropy:

Entropy: is a metric that gives the degree of impurity in a specified attribute. The following formula can be used to compute entropy:

$$Entropy(S) = -P(yes)\log_2 P(yes) - P(no)\log 2\, P(no)$$

$$Entropy(S) = \sum_{i=1}^{n} -p_i \log_2 p_i$$

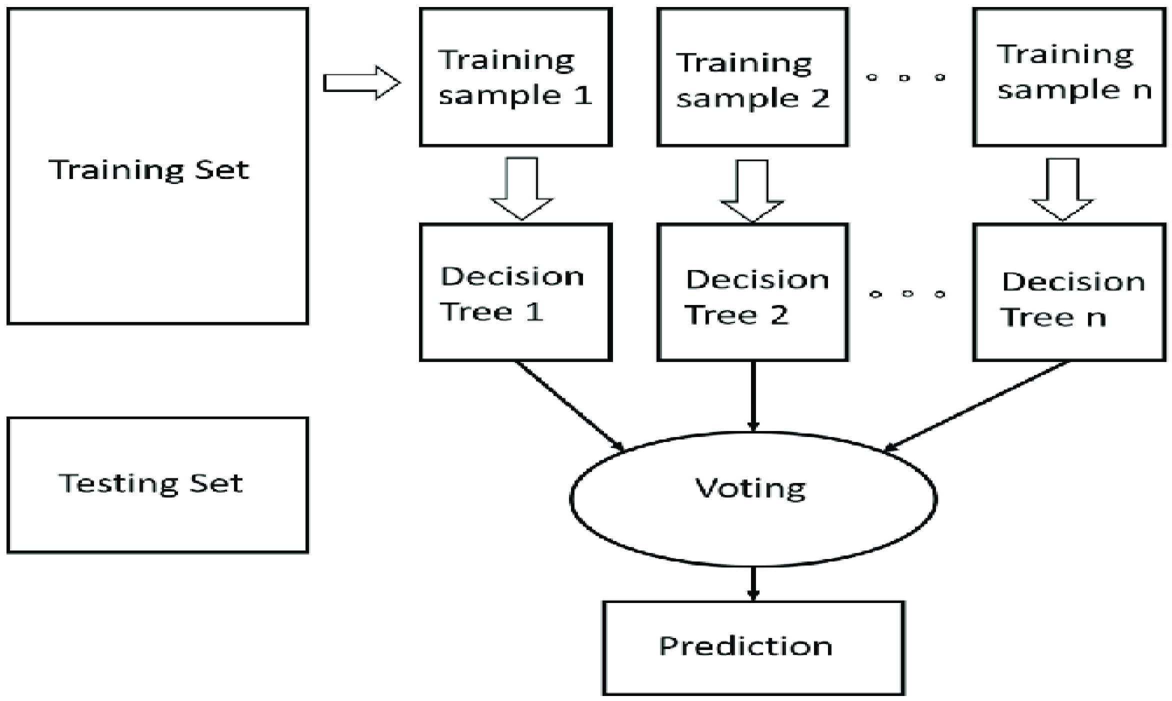
Where S denotes a complete sample, P(yes) denotes the probability of yes,P(no) denotes the probability of no

Gini Index: Gini is estimated by deducting the sum of squared probabilities of each class from one. The lower Gini Index value is preferred rather than a higher value. Scikit is the default value and supports "Gini" criteria for Gini Index.

$$Gini\ Index = 1 - \sum_{i=1} (P_i^2)$$

Where Pi denotes the probability of a tuple , say R belonging to class ci of the decision trees the model is decided by majority voting. One of the reasons for its popularity as a machine learning approach is that it can handle the issue of overfitting trees.

RANDOM FOREST CLASSIFIER:

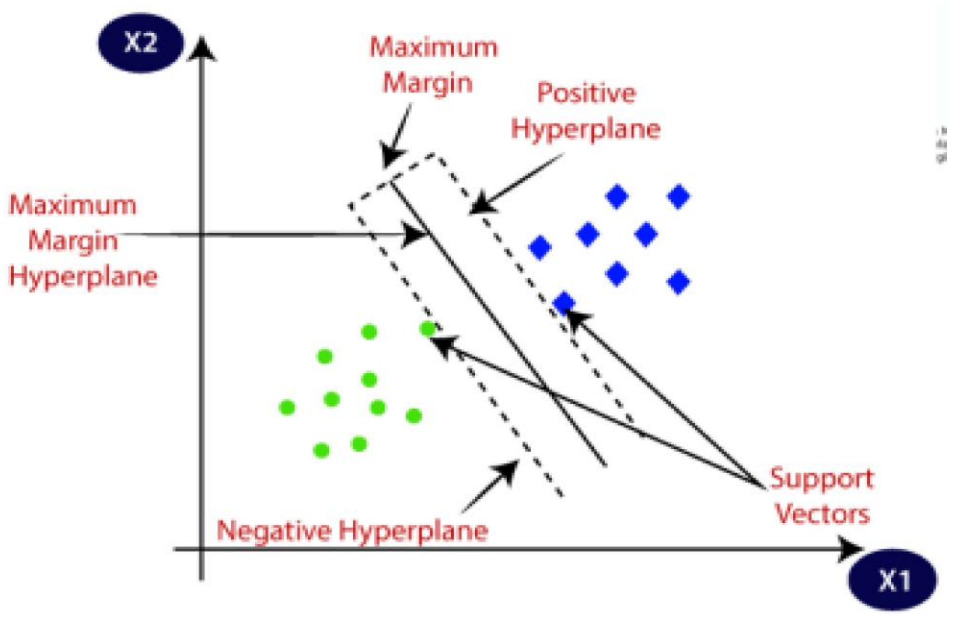


Step 1: K instances are chosen at random from the given training
Step 2: Decision trees are created for the chosen instances.
Step 3: The N is selected for the number of estimators to be created.
Step 4: Here Step 1 & Step 2 is repeated.
Step 5: For the new instance, the predictions of each estimator is determined, and the category with the Hughes votes is assigned.

RANDOM FOREST MODEL:

- **Accuracy:** 84.52%
- **Precision:** Class 1 has a very high precision of 0.97, meaning the model is very effective at correctly identifying osteoporosis patients.
- **Recall:** Class 0 has a very high recall (0.97), showing that it is excellent at identifying individuals without osteoporosis.
- **F1-Score:** Both precision and recall are balanced, providing an overall well-performing F1-score for both classes.

SVM: SVM is used for Classification as well as Regression problems .The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
SVM chooses the extreme points/vectors that help in creating the hyperplane These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.



Here is this diagram of two different categories that are classified using a decision boundary or hyperplane.

Type of SVM:

1. Linear
2. Non -Linear

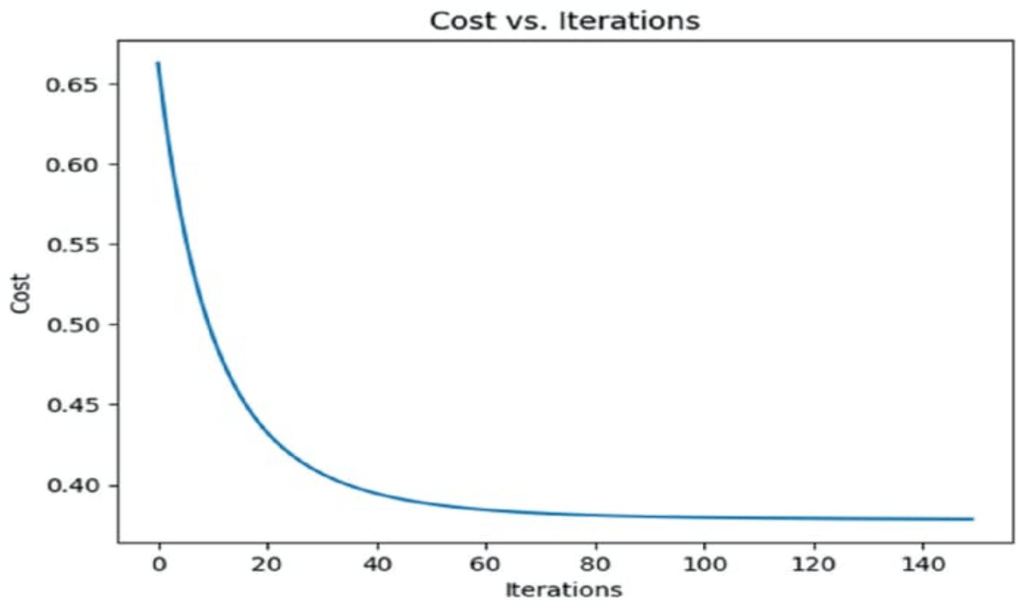
Here we will use a linear SVM for prediction

SVC MODEL:

- **Accuracy:** 83.16%
- **Precision:** Class 1 (osteoporosis) has a precision of 0.89, and class 0 has a precision of 0.79, indicating that the model is fairly good at correctly identifying both classes.
- **Recall:** Class 1 (osteoporosis) has a recall of 0.77, which means the model misses some osteoporosis cases. Class 0 has a higher recall of 0.90, showing strong performance in identifying healthy individuals.
- **F1-Score:** The F1-scores are balanced, with a slight advantage for class 0

LOGISTIC REGRESSION:

Logistic Regression is a statistical model used for binary classification (e.g., predicting "Diabetic" or "Non-Diabetic"). It models the probability of a patient having diabetes based on their features (e.g., glucose, BMI). The "Cost" plot illustrates the model learning to minimize prediction errors over training "Iterations," aiming for a lower cost and better accuracy in distinguishing diabetic from non-diabetic cases.



The equation for the Binary Cross-Entropy Loss for a single training example is:

$$L(y,(y\,)\hat{}\,) = -\left[y\,log(\hat{y}) + (1-y)\,log(1-\widehat{y)}\right]$$

Where:

y is the true label (0 or 1).

$\hat{y}$ is the predicted probability of the positive class (between 0 and 1).

The overall cost function for the entire training set (what is usually plotted on the 'Cost' axis) is the average of the loss over all m training examples:

$$j(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(\,\hat{y}^{(i)}) + (1-y^{(i)})\,log(1-\hat{y}^{(i)})$$

Where:

- θ represents the model's parameters (weights and bias).
- m is the number of training examples.
- $y^{(i)}$ is the true label for the i-th example

- $y^{(i)}$ is the predicted probability for the i-th example, calculated by the logistic regression model as $\hat{y}^{(i)} = \sigma(z^{(i)}) = \frac{1}{1+e^{-z(i)'}}$ where $z^{(i)} = \theta^T x^{(i)}\,(or\,w^T x^{(i)} + b)$.

Steps in Logistic Regression:

Step 1: Data Collection and Preprocessing: gather relevant data, handle missing values, and potentially transform features.
Step 2: Data Splitting: divide the data into training and testing sets.
Step 3: Model Training: fit the logistic regression model to the training data.
Step 4: Model Evaluation: assess the model's performance using appropriate metrics.
Step 5: Prediction and Deployment: use the trained model to predict diabetes status on new data.

Logistic Regression (LogReg) Model:

- **Accuracy:** 83.42%
- **Precision:** Class 1 (osteoporosis) has a slightly higher precision (0.88), meaning the model is effective in identifying patients with osteoporosis.
- **Recall:** The recall for class 1 is 0.78, which is decent but not as high as for class 0 (0.89), indicating the model misses some osteoporosis cases.
- **F1-Score:** The F1-scores for both classes are similar, reflecting a balanced performance in predicting both classes.

NATIVE BAYES:

- The Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- It is ○ probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Native Bayes Algorithm are spam filtration Sentimental analysis, and classifying articles.

BAYE'S THEOREM: Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

○ The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Where,
P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.,
P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
P(A) is Prior Probability: Probability of hypothesis before observing the Evidence.
P(B) is Marginal Probability: Probability of Evidence.

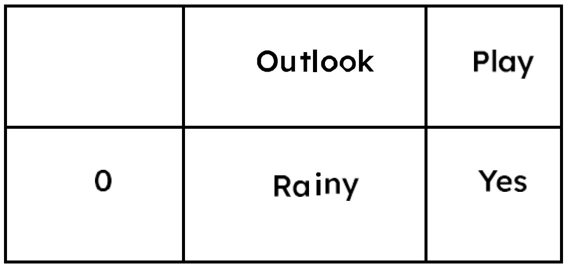
WORKING OF NATIVE BAYES' CLASSIFIER:

Working of Naive Bayes' Classifier can be understood with the help of the below example:
Suppose we have a dataset of weather conditions and corresponding target variable "Play". So using this dataset we need to decide whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

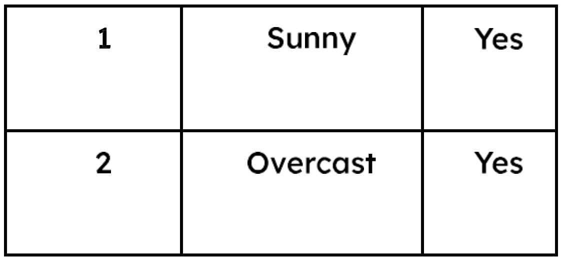
1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:



	Outlook	Play
0	Rainy	Yes



1	Sunny	Yes
2	Overcast	Yes

3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes

6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the weather Conditions:

Weather	Yes	No

Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0 29
Sunny	2	3	5/14=0 35
All	4/14=0.29	10/14=0.71	

Applying Bayes Theorem :
P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)
P(Sunny|Yes)= 3/10= 0.3
P(Sunny)= 0.35
P(Yes)=0.71

Naive Bayes (NB) Model:

- Accuracy: 86.99%
- Precision: The Gaussian Naive Bayes model shows excellent precision for class 1 (osteoporosis) at 0.97, meaning it rarely misclassifies healthy individuals as having osteoporosis.
- Recall: Class 0 (no osteoporosis) has a high recall of 0.97, showing that it effectively identifies individuals without osteoporosis.
- .F1-Score: Both classes have strong F1-scores, with class 0 having slightly better performance than class 1.

Result And Discussion:

Diabetes is a condition characterized by weakened bones that are more susceptible to fractures. Early prediction of osteoporosis can significantly improve patient care by providing timely interventions. In this project, machine learning (ML) algorithms such as K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest are used to predict the likelihood of osteoporosis based on various input features (e.g., age, gender, lifestyle factors, medical history, etc.).

The goal is to evaluate and compare the performance of each of these algorithms in terms of prediction accuracy.

K-Nearest Neighbors (KNN): KNN is a non-parametric method that classifies data points based on the majority vote of their neighbors. The number of neighbors (K) is a hyperparameter.

Logistic Regression: Logistic Regression is a linear classifier that models the probability of the binary outcome (e.g., osteoporosis or not). It uses the logistic function (sigmoid) to predict probabilities.

Support Vector Machine (SVM): SVM aims to find the hyperplane that best separates the classes by maximizing the margin between the classes. For non- linearly separable data, the kernel trick is used to map the data to higher- dimensional space.

Decision Tree: A decision tree splits the data into subsets based on the feature that provides the best split according to a criterion like Gini impurity or entropy. It creates a tree-like structure with branches and leaves.

Random Forest: Random Forest is an ensemble learning method that builds multiple decision trees using random subsets of the data and features. It aggregates the predictions of all the trees to make a final decision. Here in this project the accuracy of K-Nearest Neighbors (KNN) is higher among other algorithms.

Table of Diabetes:

Osteoporosis Prediction- Osteoporosis prediction by 0 and 1'

Dataset - The source of the data used for the study.

Feature selection - The features or variables used to train the machine learning algorithm.

Machine learning algorithm - The name of the algorithm used for osteoporosis prediction.

Model accuracy - The accuracy of the model in predicting osteoporosis (usually in percentage).

Evaluation metric - The metric used to evaluate the performance of the model (e.g., RMSE, MAE, R^2).

Experiment setup - The parameters used for the experiment, such as training and testingsplit, hyperparameters tuning, and cross-validation.

Accuracy:

The accuracy of osteoporosis prediction using machine learning algorithms depends on various factors such as the quality and quantity of data used, the choice of machine learning algorithm, and the specific health conditions and other involving factors. Generally, the accuracy of the prediction is evaluated using evaluation metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coefficient of Determination (R^2).

Additionally, techniques such as cross-validation help ensure the model's generalizability, preventing overfitting. Factors like feature selection, data quality, and handling missing or imbalanced data also significantly impact the model's performance. In clinical settings, it is essential for the model to not only be accurate but also interpretable and reliable, ensuring that it provides actionable insights for healthcare providers. Therefore, an accurate osteoporosis prediction model must be carefully assessed through multiple metrics and validated in clinical contexts to ensure its efficacy and safety.

1. K-Nearest Neighbors (KNN): 86.39%

2. Logistic Regression (LogReg): 83.42%

3. Gaussian Naive Bayes (NB): 86.99%

4. Decision Tree (DT): 82.14%

5. Random Forest (RF): 84.52%

6. Support Vector Classifier (SVC): 83.16%

Summary:

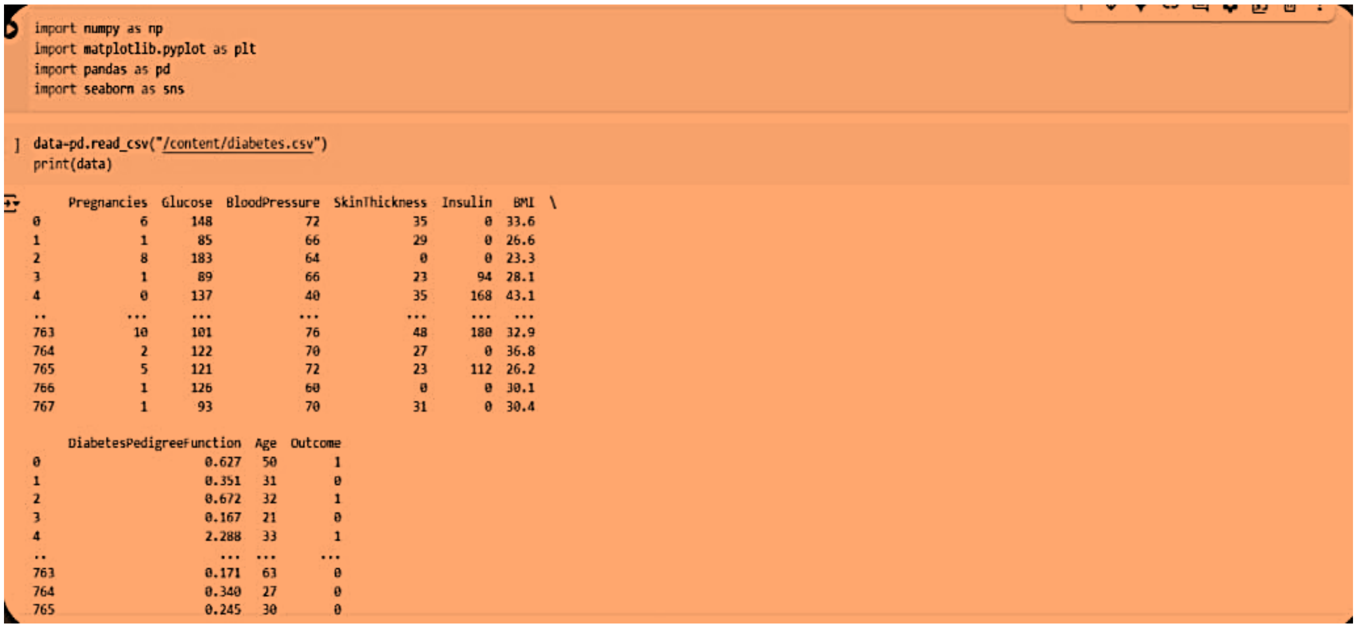
• **Best Performance:** The Gaussian Naive Bayes model performs the best with an accuracy of 86.99%, excelling in both precision and recall for class (osteoporosis).

• **Balanced Models:** KNN and Random Forest also perform strongly with high accuracy and balanced results across classes.

• **Moderate Performance:** The Logistic Regression and SVC models offer good performance but have slightly lower accuracy and recall for class compared to the other models.

• **Decision Tree:** The Decision Tree model has the lowest accuracy (82.14%) and slightly lower precision and recall compared to the other.

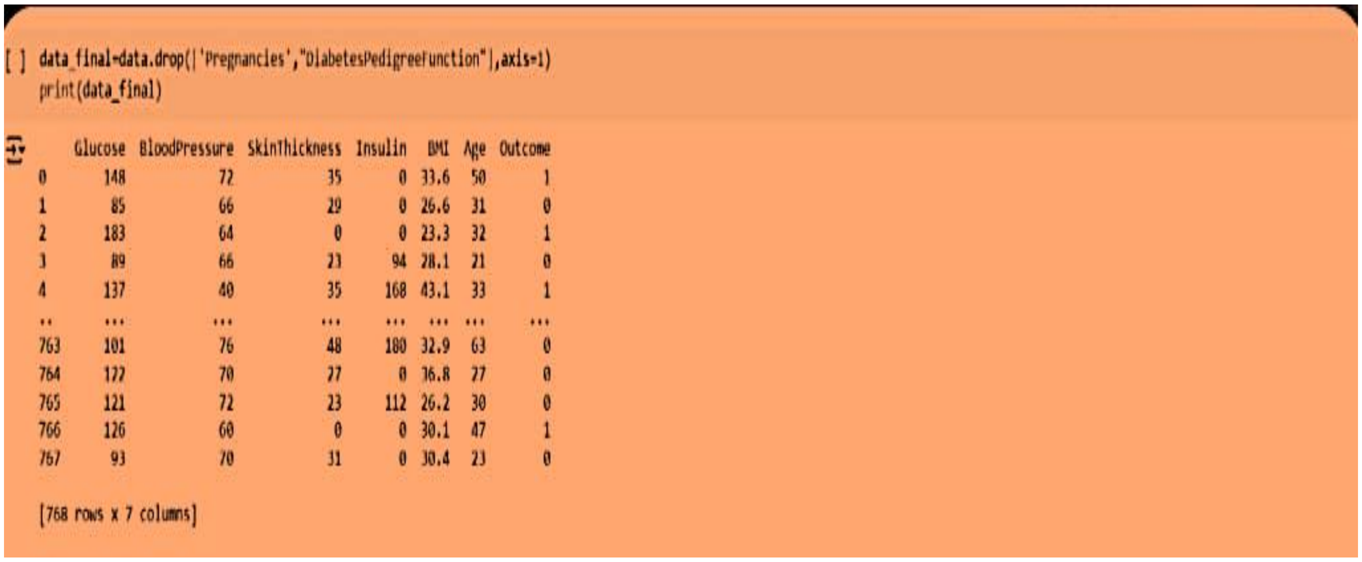
OUTPUTS



```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

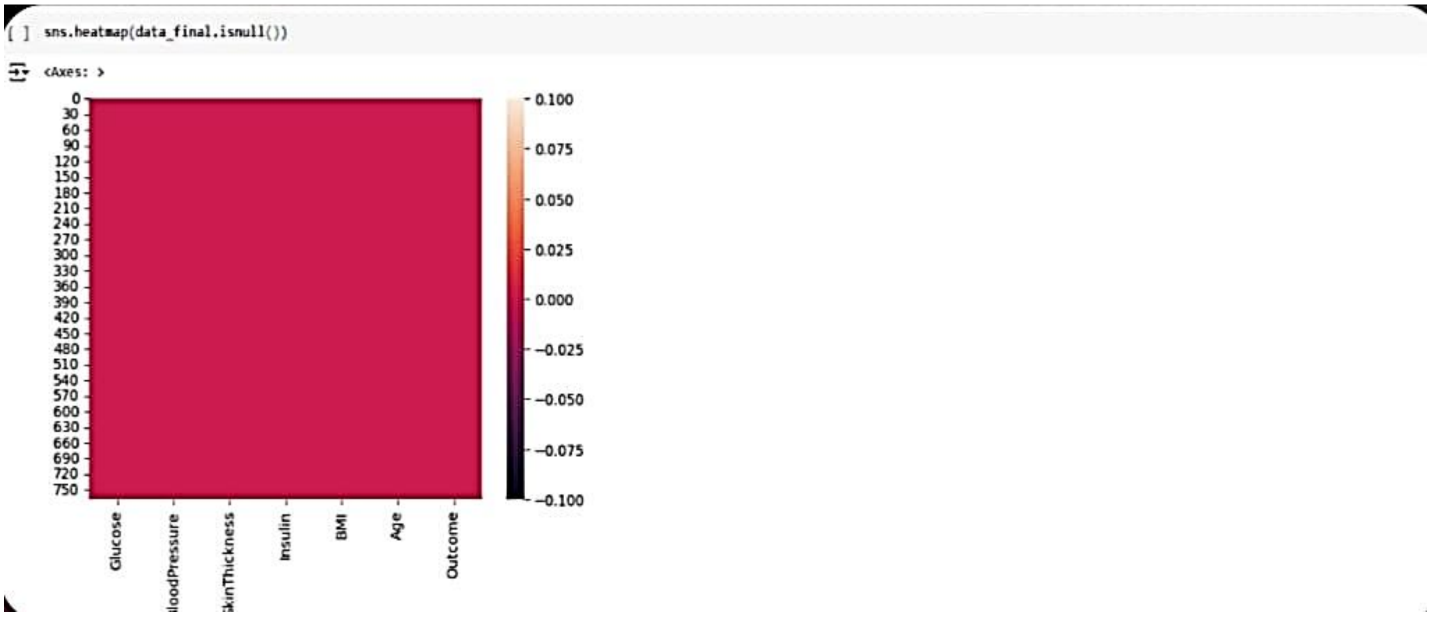
```
data=pd.read_csv("/content/diabetes.csv")
print(data)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
```



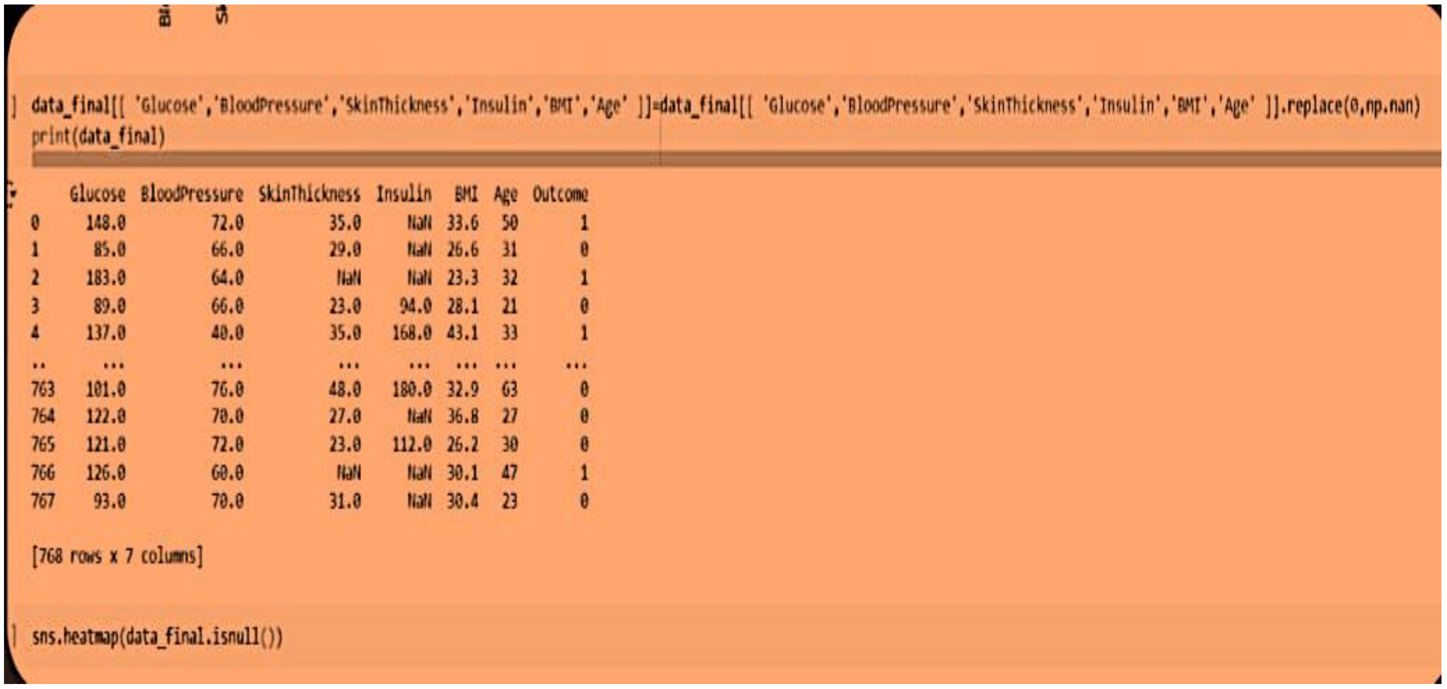
```
data_final=data.drop(['Pregnancies',"DiabetesPedigreeFunction"],axis=1)
print(data_final)
```

```
     Glucose  BloodPressure  SkinThickness  Insulin   BMI  Age  Outcome
0        148             72             35        0  33.6   50        1
1         85             66             29        0  26.6   31        0
2        183             64              0        0  23.3   32        1
3         89             66             23       94  28.1   21        0
4        137             40             35      168  43.1   33        1
..       ...            ...            ...      ...   ...  ...      ...
763      101             76             48      180  32.9   63        0
764      122             70             27        0  36.8   27        0
765      121             72             23      112  26.2   30        0
766      126             60              0        0  30.1   47        1
767       93             70             31        0  30.4   23        0

[768 rows x 7 columns]
```



```
[ ] sns.heatmap(data_final.isnull())
```

```
<Axes: >
```



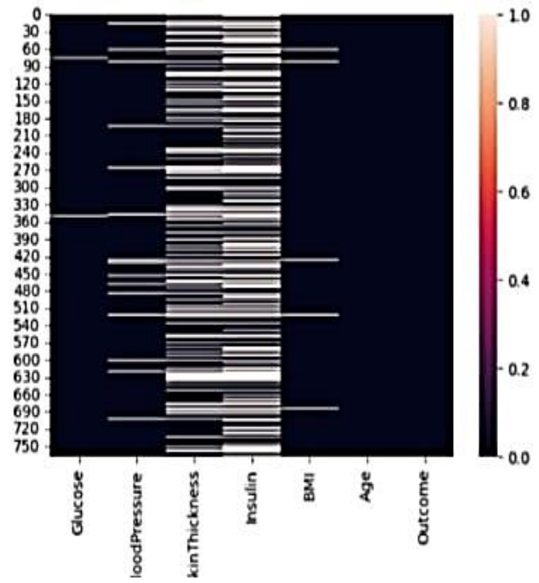
```
] data_final[[ 'Glucose','BloodPressure','SkinThickness','Insulin','BMI','Age' ]]=data_final[[ 'Glucose','BloodPressure','SkinThickness','Insulin','BMI','Age' ]].replace(0,np.nan)
  print(data_final)
```

```
     Glucose  BloodPressure  SkinThickness  Insulin   BMI  Age  Outcome
0      148.0           72.0           35.0      NaN  33.6   50        1
1       85.0           66.0           29.0      NaN  26.6   31        0
2      183.0           64.0            NaN      NaN  23.3   32        1
3       89.0           66.0           23.0     94.0  28.1   21        0
4      137.0           40.0           35.0    168.0  43.1   33        1
..       ...            ...            ...      ...   ...  ...      ...
763    101.0           76.0           48.0    180.0  32.9   63        0
764    122.0           70.0           27.0      NaN  36.8   27        0
765    121.0           72.0           23.0    112.0  26.2   30        0
766    126.0           60.0            NaN      NaN  30.1   47        1
767     93.0           70.0           31.0      NaN  30.4   23        0

[768 rows x 7 columns]
```

```
] sns.heatmap(data_final.isnull())
```

```
[ ] sns.heatmap(data_final.isnull())
```

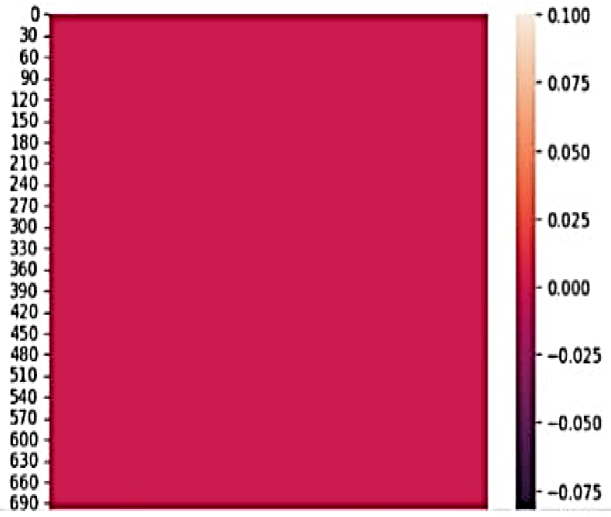
```
<Axes: >
```



```
] data_final.fillna({'Glucose':data_final['Glucose'].mean()},inplace=True)
  data_final.fillna({'BloodPressure':data_final['BloodPressure'].mean()},inplace=True)
  data_final.fillna({'SkinThickness':data_final['SkinThickness'].mean()},inplace=True)
  data_final.fillna({'Insulin':data_final['Insulin'].mean()},inplace=True)
  data_final.fillna({'Age':data_final['Age'].mean()},inplace=True)
  data_final.fillna({'BMI':data_final['BMI'].mean()},inplace=True)
```

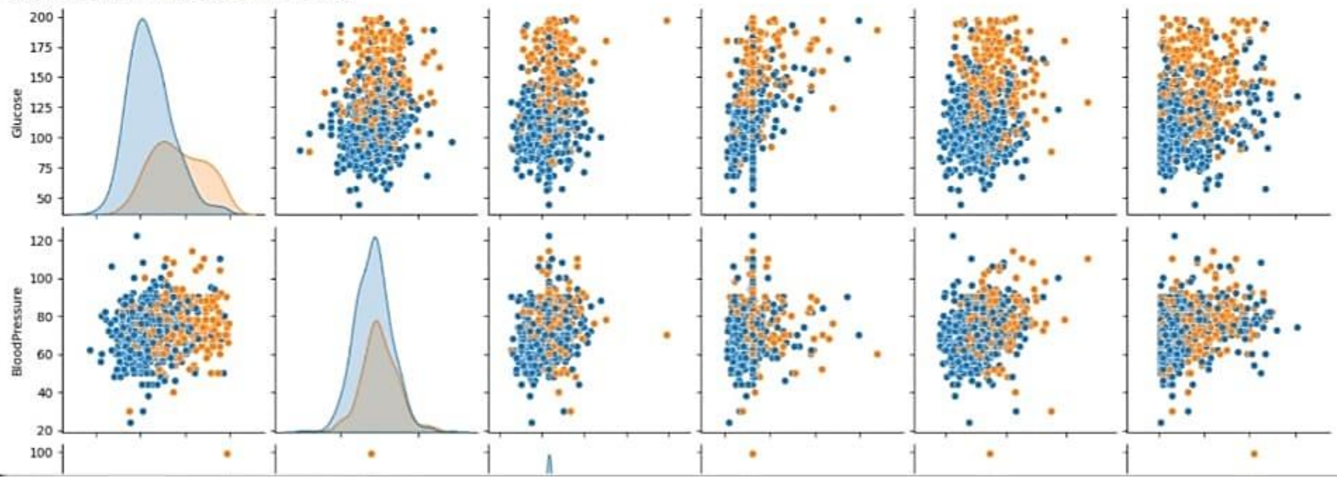
```
] sns.heatmap(data_final.isnull())
```

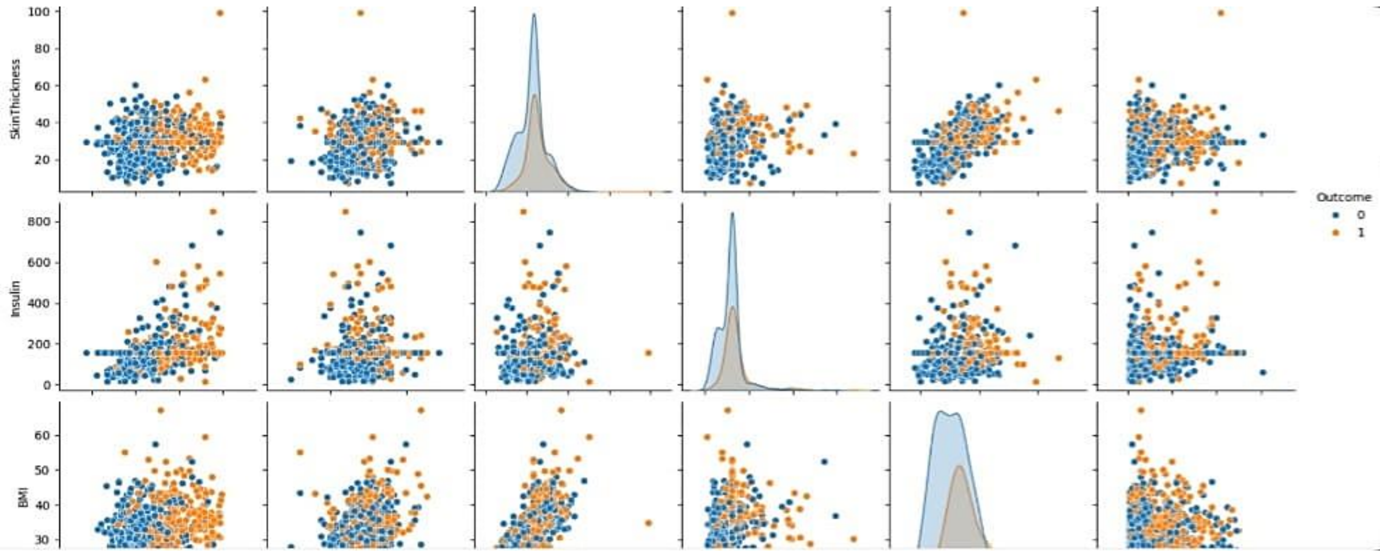
```
<Axes: >
```

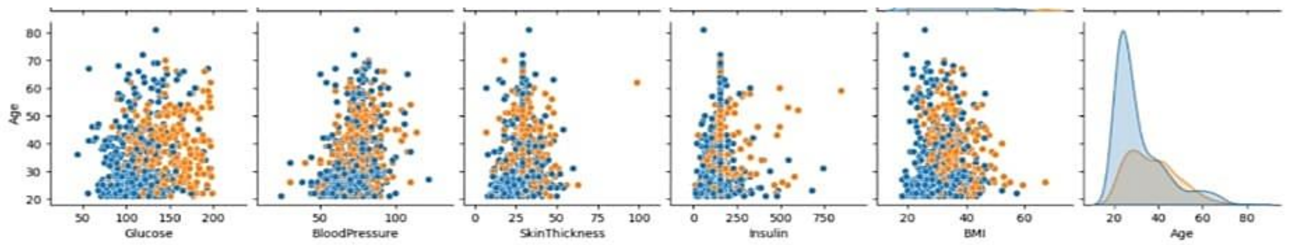


```
sns.pairplot(data_final,hue="Outcome")
```

```
<seaborn.axisgrid.PairGrid at 0x7b7e7ff12590>
```





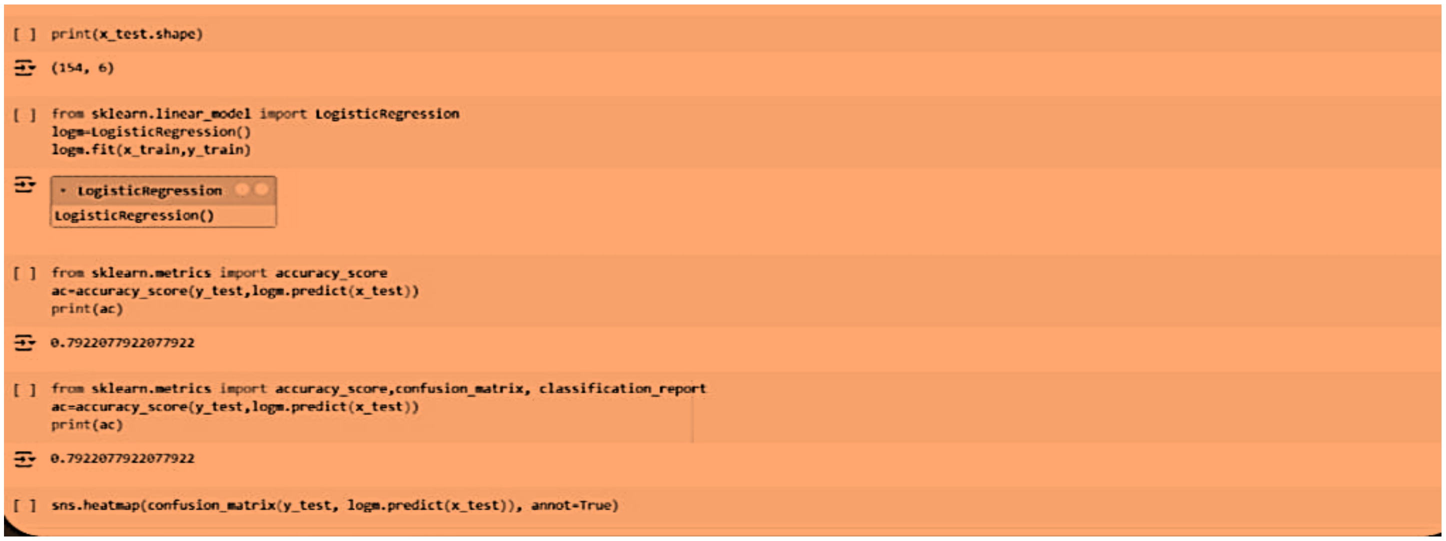


```
x=data_final.iloc[:,:-1].values
y=data_final.iloc[:,-1].values
```

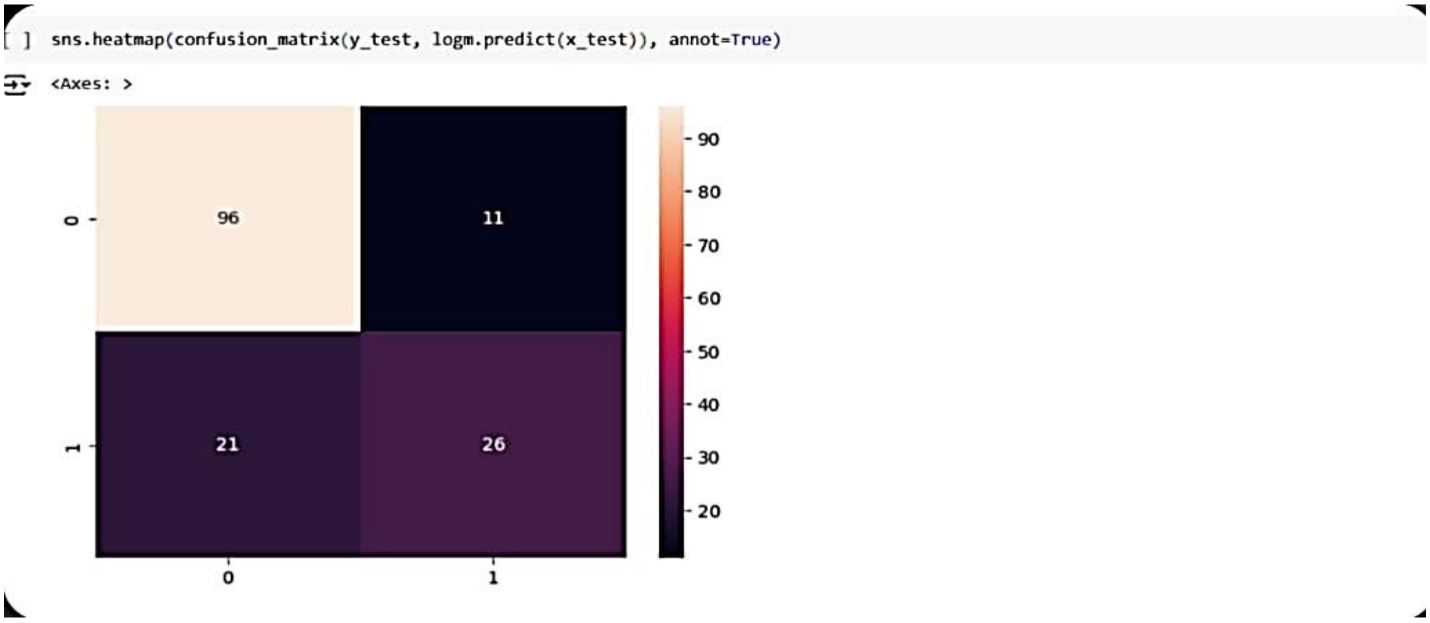
```
print(data_final.shape)
```

```
(768, 7)
```

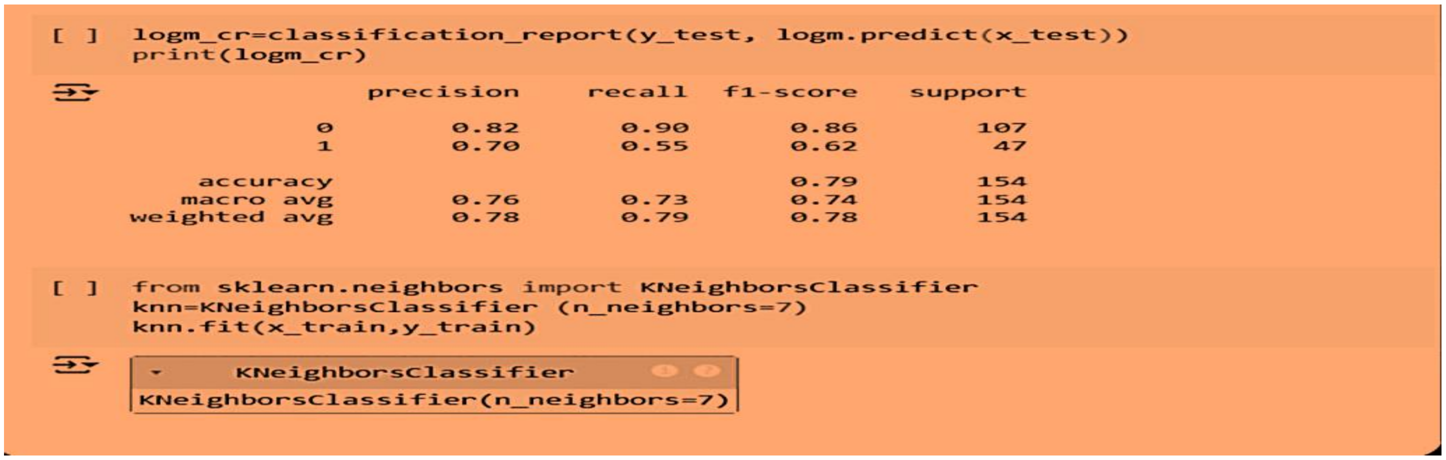
```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train,y_test=train_test_split(x, y, test_size=.2, random_state=0)
```



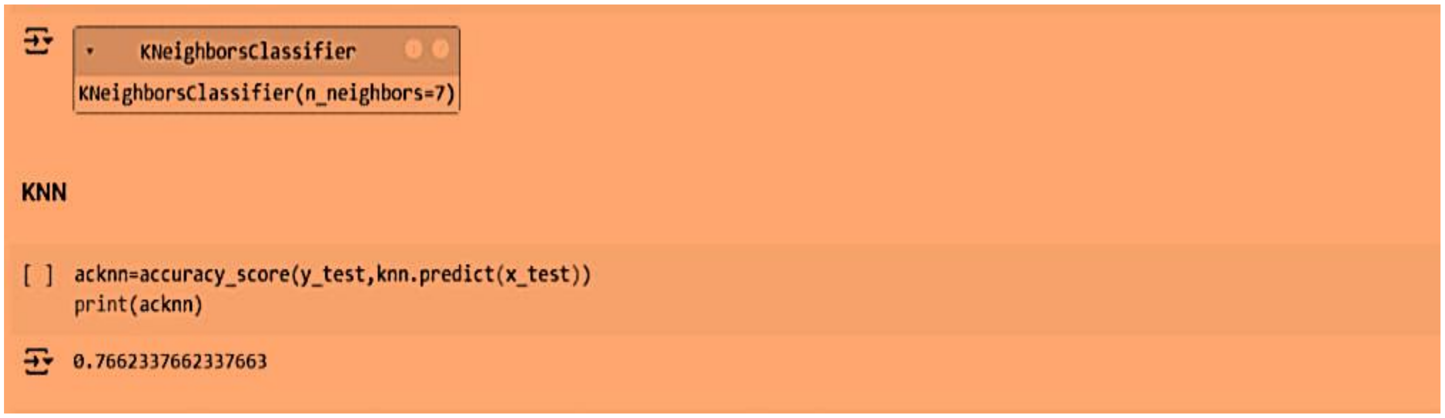
```
[ ] print(x_test.shape)

(154, 6)

[ ] from sklearn.linear_model import LogisticRegression
    logm=LogisticRegression()
    logm.fit(x_train,y_train)

    ▾ LogisticRegression
    LogisticRegression()

[ ] from sklearn.metrics import accuracy_score
    ac=accuracy_score(y_test,logm.predict(x_test))
    print(ac)

0.7922077922077922

[ ] from sklearn.metrics import accuracy_score,confusion_matrix, classification_report
    ac=accuracy_score(y_test,logm.predict(x_test))
    print(ac)

0.7922077922077922

[ ] sns.heatmap(confusion_matrix(y_test, logm.predict(x_test)), annot=True)
```



```
[ ] sns.heatmap(confusion_matrix(y_test, logm.predict(x_test)), annot=True)

<Axes: >
```



```
[ ] logm_cr=classification_report(y_test, logm.predict(x_test))
    print(logm_cr)

                  precision    recall  f1-score   support

               0       0.82      0.90      0.86       107
               1       0.70      0.55      0.62        47

        accuracy                           0.79       154
       macro avg       0.76      0.73      0.74       154
    weighted avg       0.78      0.79      0.78       154

[ ] from sklearn.neighbors import KNeighborsClassifier
    knn=KNeighborsClassifier (n_neighbors=7)
    knn.fit(x_train,y_train)

    ▾ KNeighborsClassifier
    KNeighborsClassifier(n_neighbors=7)
```



```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=7)
```

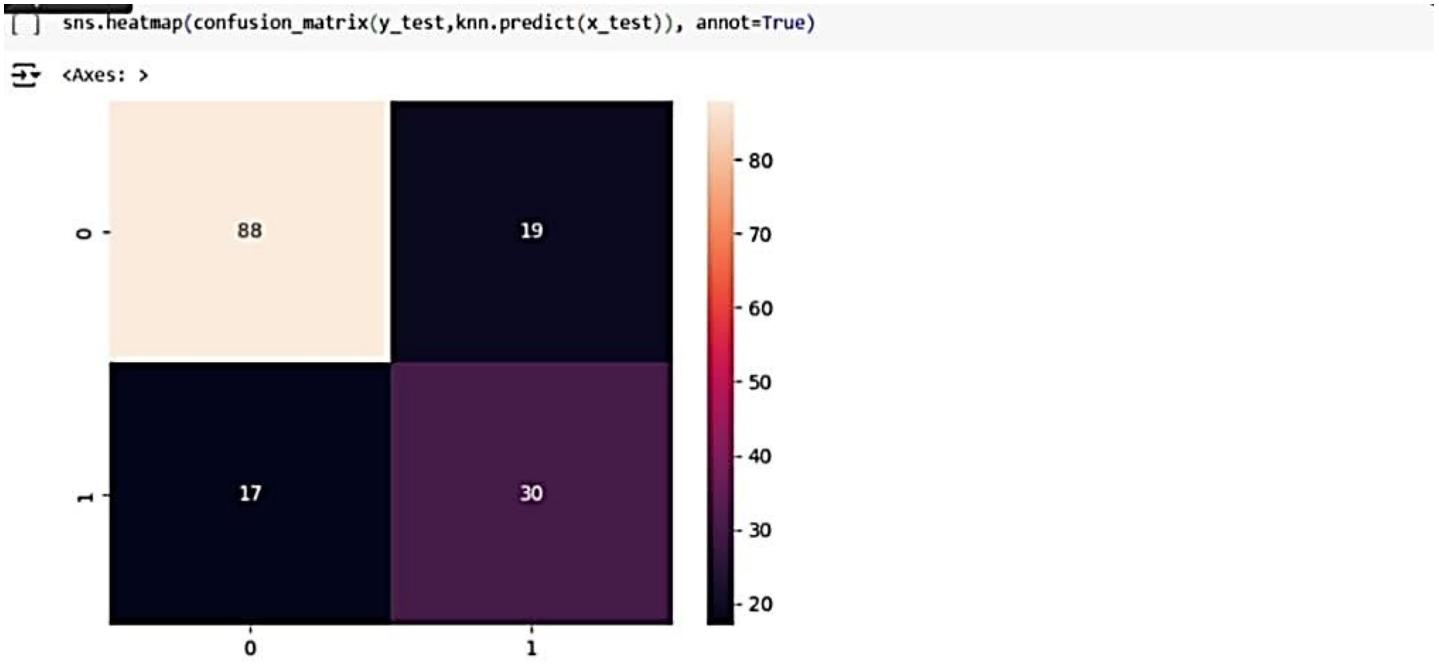
KNN

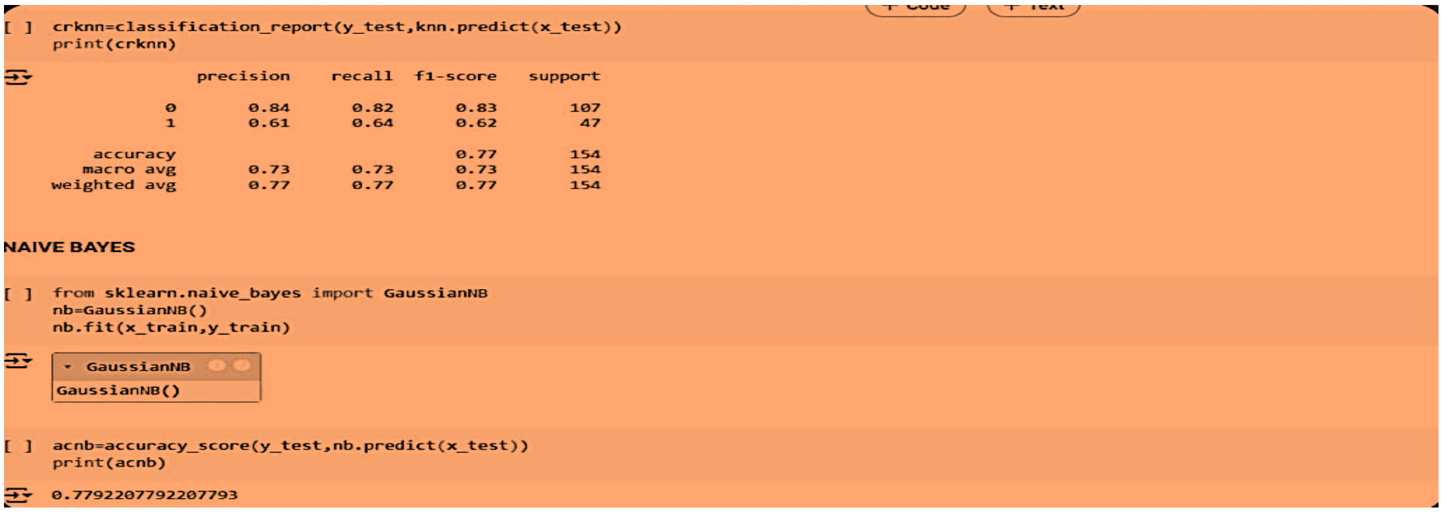
```
[ ] acknn=accuracy_score(y_test,knn.predict(x_test))
    print(acknn)
```

```
0.7662337662337663
```

```
[ ] sns.heatmap(confusion_matrix(y_test,knn.predict(x_test)), annot=True)
```

```
<Axes: >
```





```
[ ] crknn=classification_report(y_test,knn.predict(x_test))
    print(crknn)
```

```
              precision    recall  f1-score   support

           0       0.84      0.82      0.83       107
           1       0.61      0.64      0.62        47

    accuracy                           0.77       154
   macro avg       0.73      0.73      0.73       154
weighted avg       0.77      0.77      0.77       154
```

NAIVE BAYES

```
[ ] from sklearn.naive_bayes import GaussianNB
    nb=GaussianNB()
    nb.fit(x_train,y_train)
```

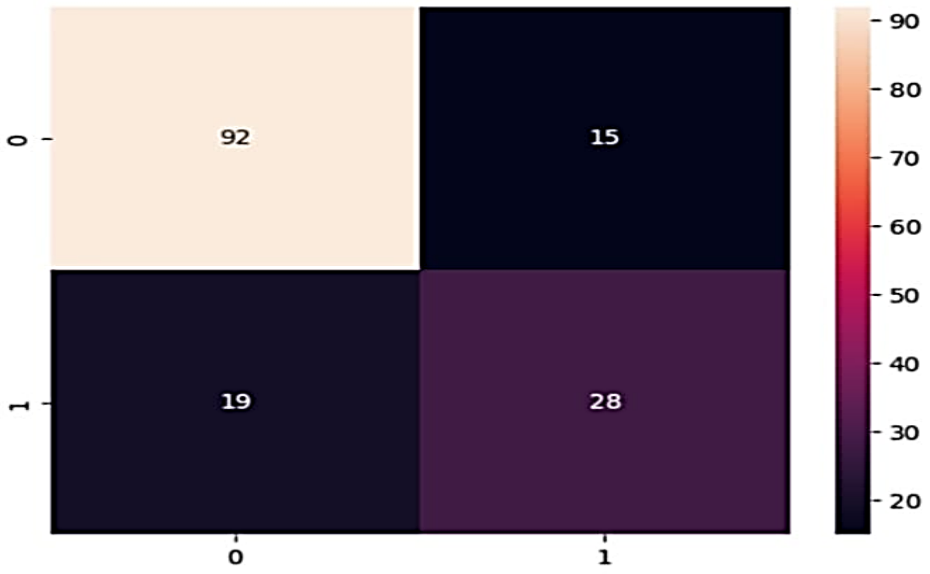
```
GaussianNB
GaussianNB()
```

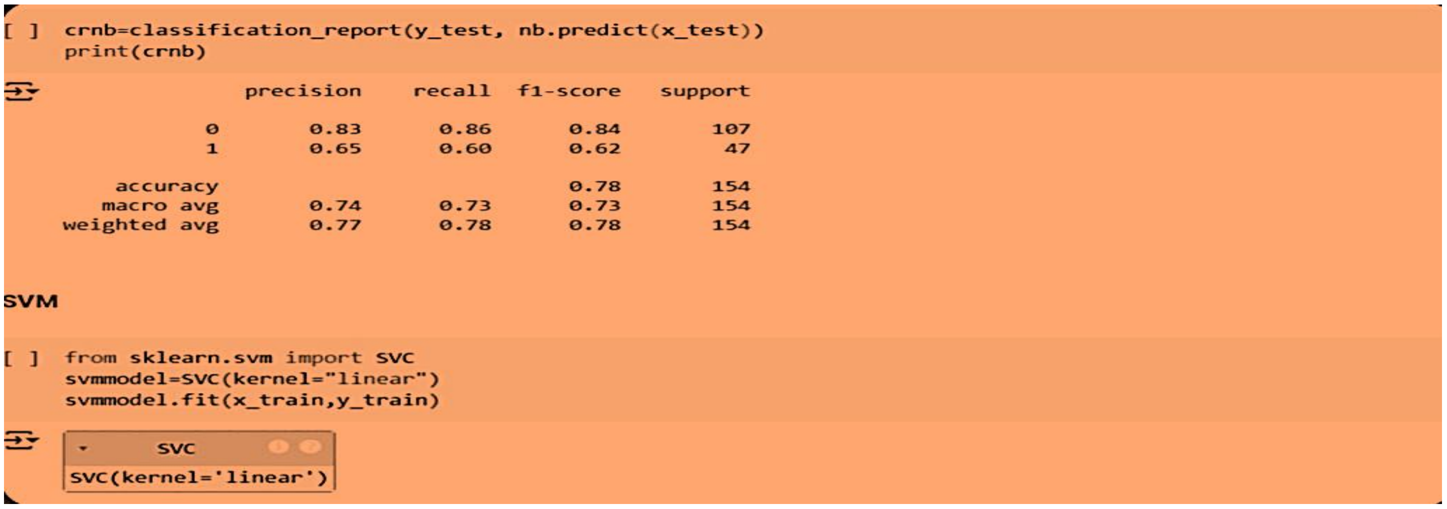
```
[ ] acnb=accuracy_score(y_test,nb.predict(x_test))
    print(acnb)
```

```
0.7792207792207793
```

```
[ ] sns.heatmap(confusion_matrix(y_test,nb.predict(x_test)), annot=True)
```

```
<Axes: >
```





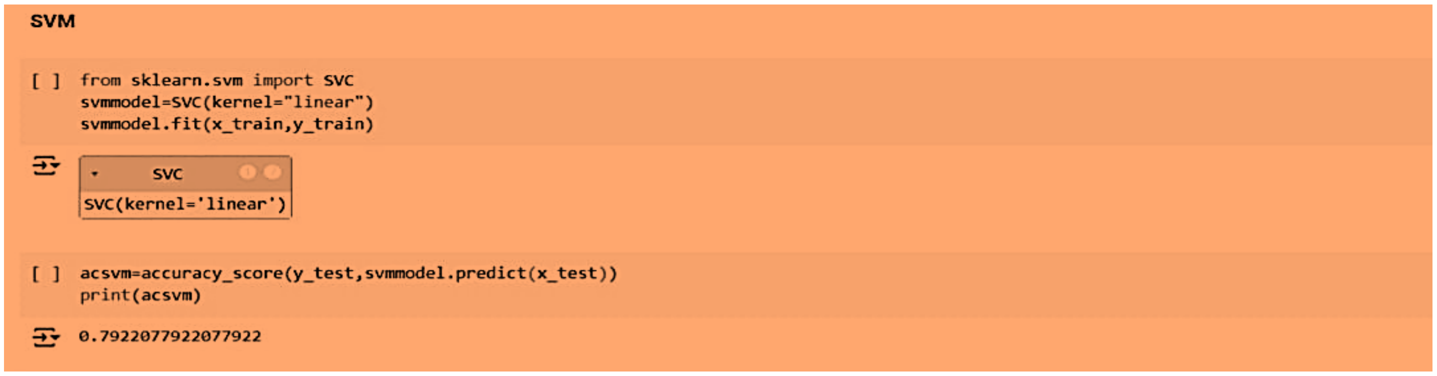
```
[ ] crnb=classification_report(y_test, nb.predict(x_test))
    print(crnb)
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       107
           1       0.65      0.60      0.62        47

    accuracy                           0.78       154
   macro avg       0.74      0.73      0.73       154
weighted avg       0.77      0.78      0.78       154
```

SVM

```
[ ] from sklearn.svm import SVC
    svmmodel=SVC(kernel="linear")
    svmmodel.fit(x_train,y_train)
```

```
SVC
SVC(kernel='linear')
```



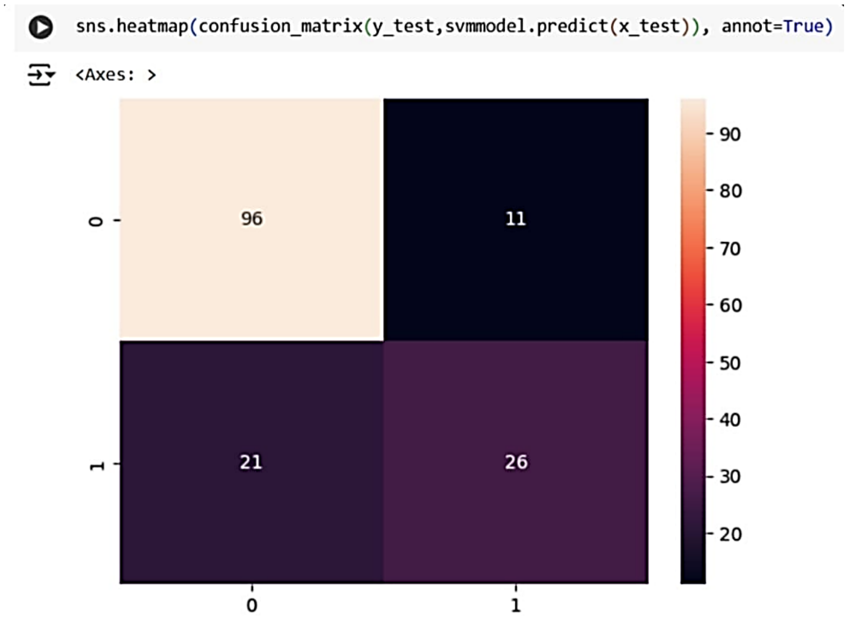
SVM

```
[ ] from sklearn.svm import SVC
    svmmodel=SVC(kernel="linear")
    svmmodel.fit(x_train,y_train)
```

```
SVC
SVC(kernel='linear')
```

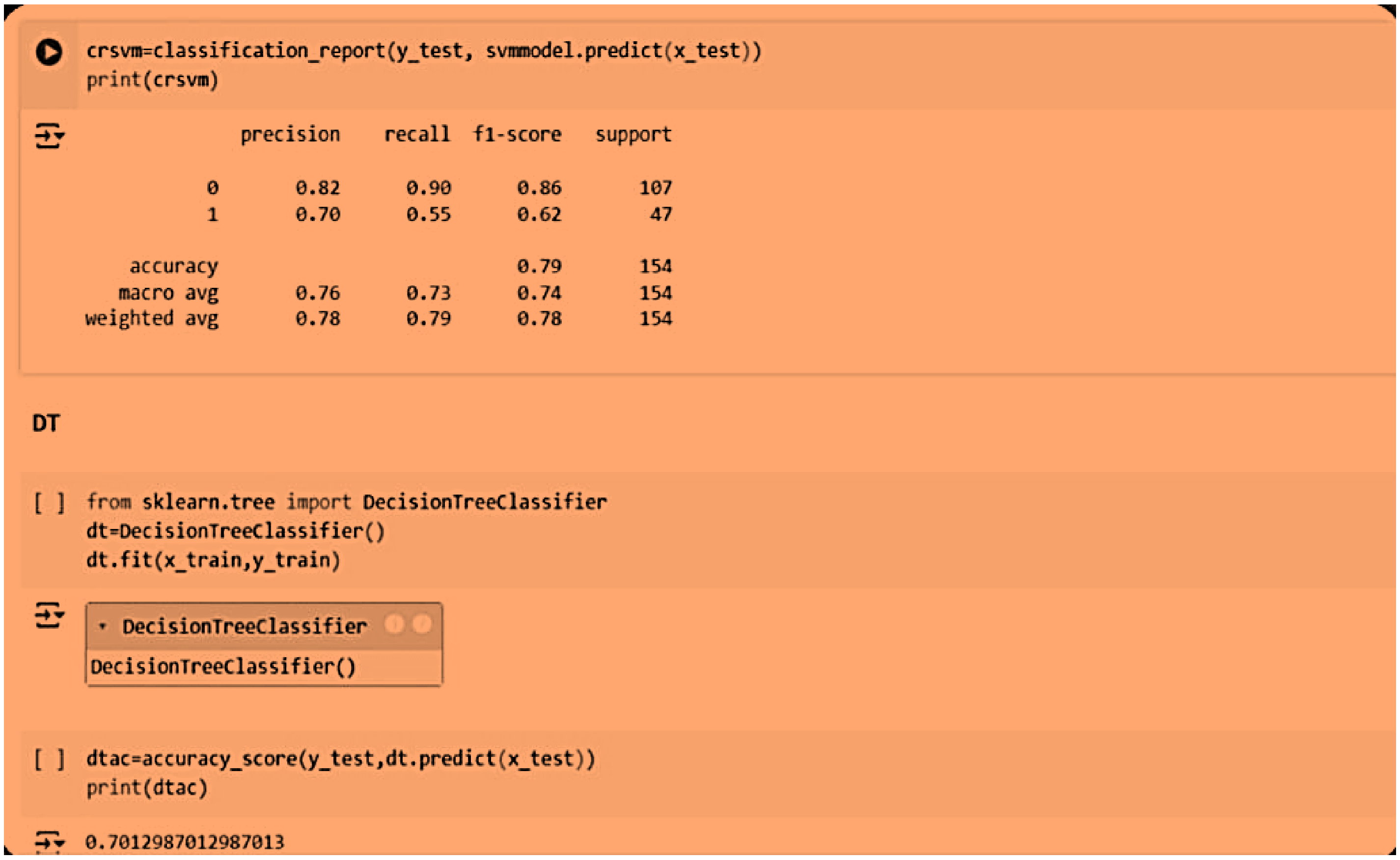
```
[ ] acsvm=accuracy_score(y_test,svmmodel.predict(x_test))
    print(acsvm)
```

```
0.7922077922077922
```



```
sns.heatmap(confusion_matrix(y_test,svmmodel.predict(x_test)), annot=True)
```

```
<Axes: >
```



```
crsvm=classification_report(y_test, svmmodel.predict(x_test))
print(crsvm)
```

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86       107
           1       0.70      0.55      0.62        47

    accuracy                           0.79       154
   macro avg       0.76      0.73      0.74       154
weighted avg       0.78      0.79      0.78       154
```

DT

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
```

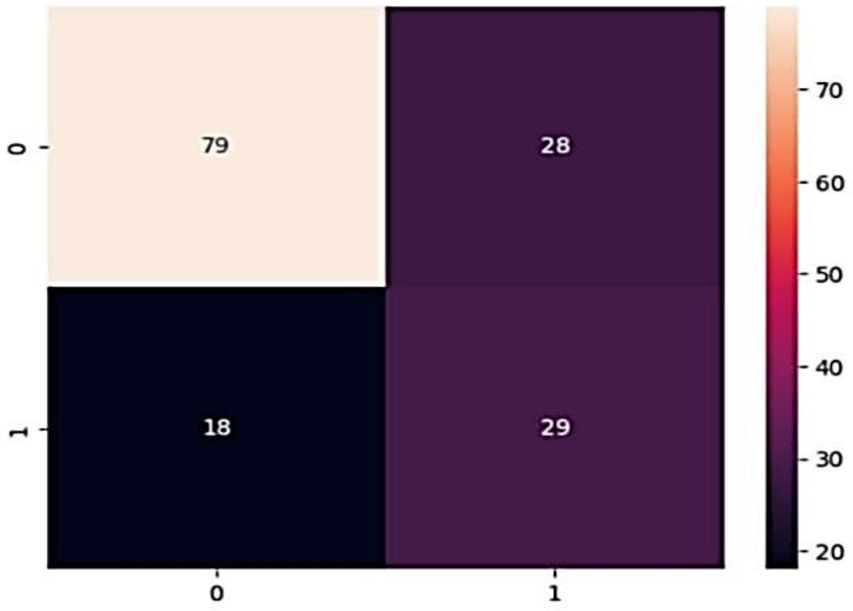
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

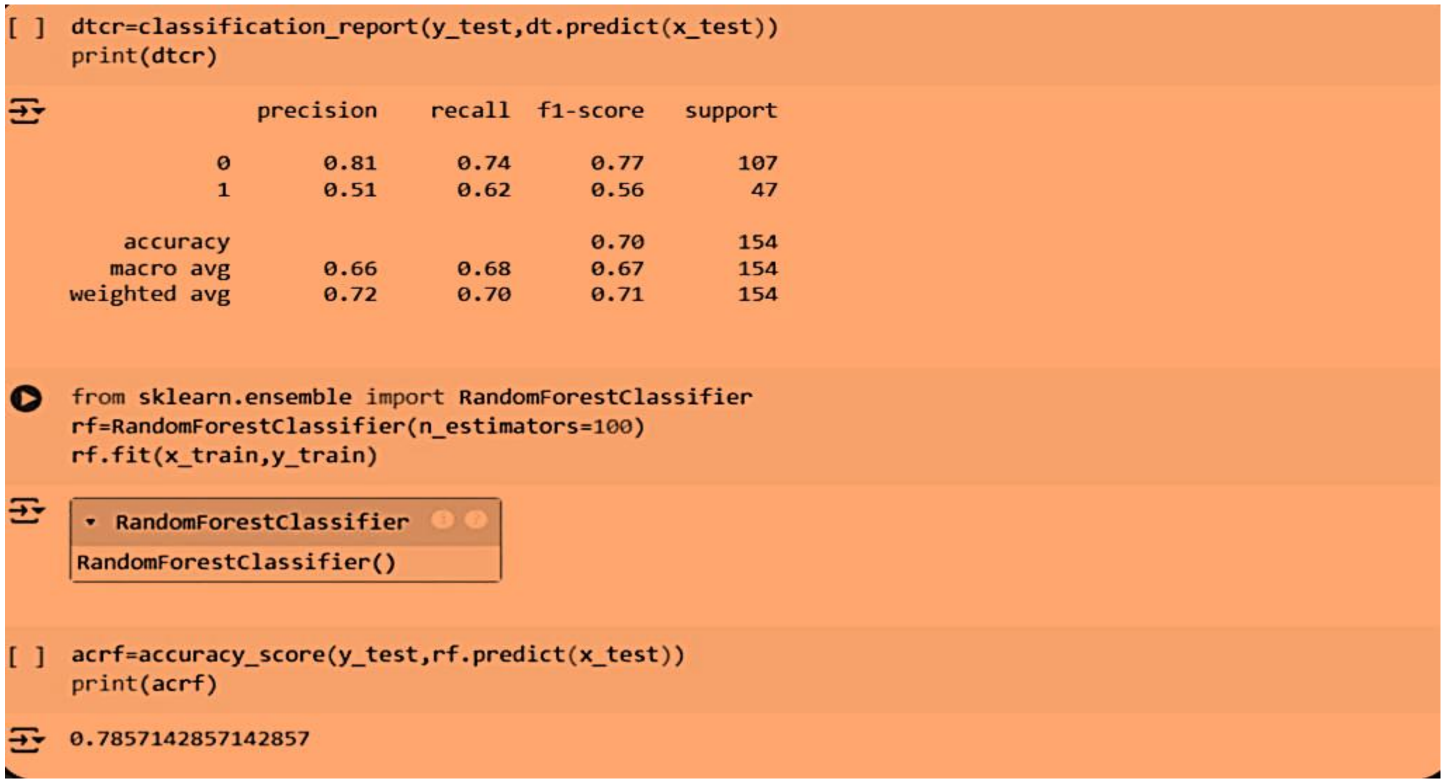
```
dtac=accuracy_score(y_test,dt.predict(x_test))
print(dtac)
```

```
0.7012987012987013
```

```
[ ] sns.heatmap(confusion_matrix(y_test,dt.predict(x_test)), annot=True)
```

```
<Axes: >
```





```
[ ] dtcr=classification_report(y_test,dt.predict(x_test))
    print(dtcr)
```

```
              precision    recall  f1-score   support

           0       0.81      0.74      0.77       107
           1       0.51      0.62      0.56        47

    accuracy                           0.70       154
   macro avg       0.66      0.68      0.67       154
weighted avg       0.72      0.70      0.71       154
```

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(x_train,y_train)
```

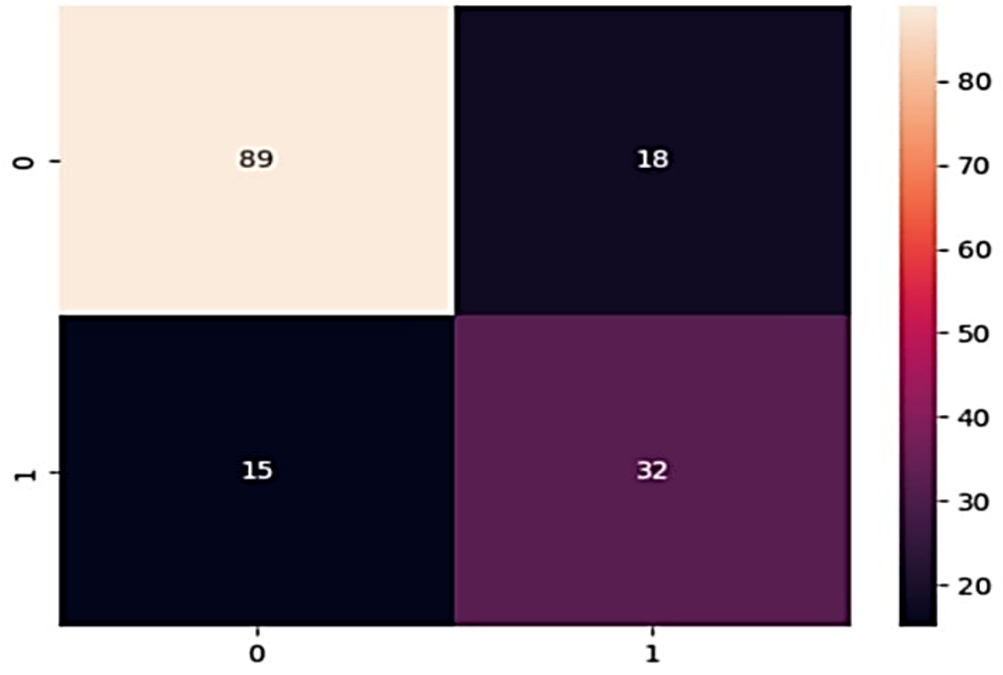
```
▾ RandomForestClassifier
RandomForestClassifier()
```

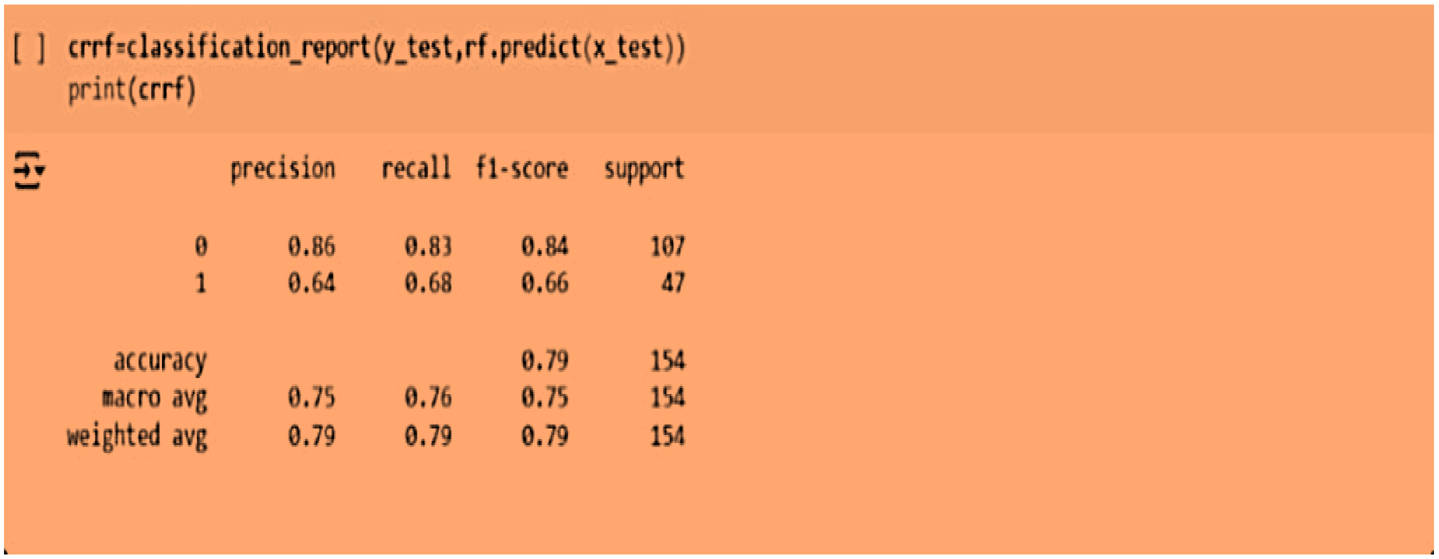
```
[ ] acrf=accuracy_score(y_test,rf.predict(x_test))
    print(acrf)
```

```
0.7857142857142857
```

```
[ ] sns.heatmap(confusion_matrix(y_test,rf.predict(x_test)),annot=True)
```

```
<Axes: >
```





```
[ ] crrf=classification_report(y_test,rf.predict(x_test))
    print(crrf)
```

```
              precision    recall  f1-score   support

           0       0.86      0.83      0.84       107
           1       0.64      0.68      0.66        47

    accuracy                           0.79       154
   macro avg       0.75      0.76      0.75       154
weighted avg       0.79      0.79      0.79       154
```

CONCLUSION

A revolutionary method for early diagnosis and individualized treatment is provided using machine learning (ML) in Diabetes prediction. Complex datasets including bone mineral density (BMD), medical imaging, genetic profiles, and patient histories are expertly analyzed by machine learning (ML) methods like support vector machines, random forests, and neural networks. In order to lower the risk of fractures and improve patient outcomes, these models allow for precise risk assessment and early identification.

ML is very useful in environments with limited resources because of its capacity to combine many data sources and offer affordable solutions. T guarantee dependability and equity, however, issues including dataset biases, algorithm generalizability, and data quality must be resolved.

To overcome these constraints, data scientists and clinicians must work together. In the future, ML developments in conjunction with genomic information and sophisticated imaging could completely transform the treatment of Diabetes Globally, ML-driven solutions hold potential for bettering diagnosis, prevention. and individualized treatment plans.

FUTURE SCOPE

The future scope of this project includes several opportunities for extension and improvement. Advanced statistical analysis can be explored by incorporating metrics such as skewness, kurtosis, and hypothesis testing to identify significant factors influencing game lengths. Predictive modeling using machine learning algorithms like regression, decision trees, or neural networks can help predict game outcomes based on key parameters. Real-time analysis capabilities can be introduced with tools like Apache Kafka or Spark, enabling live performance monitoring. Strategy optimization through reinforcement learning or optimization techniques can recommend the best moves or pathways to players. Enhancing data visualization with interactive dashboards using tools like Plotly or Power BI will offer deeper insights and dynamic analysis. Integrating this project with gaming platforms can automate insights for players and developers, while scaling the pipeline to handle larger datasets efficiently will ensure broader applicability. Benchmarking game length distributions across various platforms can uncover universal patterns, while automating data collection, cleaning, and visualization will streamline the workflow. Lastly, this methodology can be customized for similar domains such as sports analytics, simulations, or other competitive processes, making the project versatile and impactful.

REFERENCES

Accuracy of Machine Learning Classification Models for the Prediction of Type 2 Diabetes Mellitus: A Systematic Survey and Meta-Analysis
This MDPI review analyses 34 studies (121,000+ patients, 2010–2021), reporting overall ML accuracy ~86%, with decision trees at 88% and neural networks at 85% arxiv.org+15mdpi.com+15arxiv.org+15
🔗 **Link:** mdpi.com/1660-4601/19/21/14280

Advances in Artificial Intelligence for Diabetes Prediction: Insights from a Systematic Literature Review (Dec 2024)
A deep dive into datasets (Pima, NHANES, REPLACE-BG, Singapore DR screening), algorithms (CNN, SVM, XGBoost), metrics, and XAI tools like SHAP/LIME arxiv.org+1pubmed.ncbi.nlm.nih.gov+1
🔗 **Link:** arxiv.org/abs/2412.14736

Predictive Value of Machine Learning for the Progression of Gestational Diabetes to Type 2 Diabetes (Jan 2025)
A BMC article presenting a systematic review & meta-analysis of ML models forecasting progression from GDM to T2DM dmsjournal.biomedcentral.com+3mdpi.com+3pubmed.ncbi.nlm.nih.gov+3
🔗 **Link:** doi.org/10.1186/s12911-024-02848-x

A recent arXiv paper showcasing an ensemble + XAI framework achieving 92.5% accuracy (ROC-AUC 0.975). Highlights top predictors like BMI, age, income, physical activity arxiv.org+9arxiv.org+9arxiv.org+9
🔗 **Link:** arxiv.org/abs/2501.18071

A Comparative Study of Machine Learning Techniques for Early Prediction of Diabetes (Jun 2025)
Compares eight algorithms on the Pima dataset. Neural networks led at 78.6%, followed by Random Forest at 76.3% arxiv.org
🔗 **Link:** arxiv.org/abs/2506.10180

Data-Driven Machine-Learning Methods for Diabetes Risk Prediction (Sensors, 2022)
MDPI paper using SMOTE + feature ranking, showing RF and k-NN models with high accuracy (>98%) frontiersin.org+13mdpi.com+13mdpi.com+13
🔗 **Link:** mdpi.com/1424-8220/22/14/5304

NHS to begin world-first trial of AI-ECG tool (Aire-DM) to predict Type 2 Diabetes risk up to 13 years ahead (Dec 2024)
This Guardian report covers the NHS pilot starting in 2025 that uses ECG-based AI with ~70% prediction accuracy theguardian.com
🔗 **Link:** theguardian.com (UK news)