Strategic Data Mining for Early Diabetes Risk Prediction: A Public Health Initiative

Kajal Waghmare -1002070226

Abhishek Joshi (ID:1002050821)

2238-IE-6318-006-Data Mining and Analytics

Data Mining Project Report

Abstract

The goal of this research project is to use data mining techniques to predict an individual's risk of developing diabetes based on vital health indicators like age, blood pressure, BMI, and glucose levels using Pima Diabetes Dataset[3]. The data mining techniques used in this research, which include model validation, feature selection, and algorithm development through hyperparameter tuning and algorithm comparison for optimal algorithm using ROC curve, are its main components. This research addresses the chronic and widespread issue of diabetes within the Pima Indian community, with the goal of contributing to targeted interventions using advanced algorithms.

Introduction & Background

Diabetes is a common chronic illness that presents major problems for public health systems around the globe. The impact of diabetes on people and society must be minimized by early detection and intervention. The burgeoning prevalence of diabetes poses a significant health challenge, particularly within specific communities such as the Pima Indians. This research project seeks to address this chronic issue by employing data mining techniques to predict an individual's risk of developing diabetes. By employing the Pima Diabetes Dataset, the objective is to reveal trends and insights hidden within vital health indicators, including age, blood pressure, BMI, and glucose levels.

The motivation behind this research stems from the urgent need to proactively manage and mitigate the impact of diabetes [1][2]. This topic's data mining becomes vital because it provides me with a means of utilizing predictive modelling, which helps to identify people who are at risk early on. In addition to accurately predicting diabetes risk, my goal is to develop models that support evidence-based decisionmaking for healthcare interventions by comprehending the interaction of various health indicators.

Data Description

The University of California, Irvine Machine Learning Repository (UCI MLR) is the custodian of the wellknown Pima Indians Diabetes Database, which can be accessed on Kaggle. This dataset, which includes health-related data from the Pima Indian community, is a result of research done by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). With 768 cases and 9 features, the dataset is designed to support predictive modelling for diabetes risk early detection. Pima Indian community particularly focuses on individuals aged 21 and above.

- Pregnancies: Number of times pregnant.
- Glucose: Plasma glucose concentration after a 2-hour oral glucose tolerance test.
- Blood Pressure: Diastolic blood pressure (mm Hg).
- Skin Thickness: Triceps skinfold thickness (mm).
- Insulin: 2-Hour serum insulin (mu U/ml).
- BMI (Body Mass Index): Body mass index, a measure of body fat based on height and weight.
- Diabetes Pedigree Function: A function that scores the likelihood of diabetes based on family history.
- Age: Age in years.
- Outcome (Target Variable): Binary variable indicating whether the person has diabetes or not (1 for diabetes, 0 for no diabetes).

These features encompass the number of pregnancies, plasma glucose concentration post a 2-hour oral glucose tolerance test, diastolic blood pressure, triceps skin thickness, 2-hour serum insulin level, Body Mass Index (BMI), a hereditary risk score for diabetes, and age. The dataset culminates with a binary target variable, denoting the presence (1) or absence (0) of diabetes. This comprehensive set of attributes facilitates the exploration of relationships between pregnancy history, physiological metrics, and age in predicting the likelihood of diabetes. Using the recommended glucose level and BMIs, new features like OGTT interpretation and nutritional status features are incorporated in the data to predict the outcome.

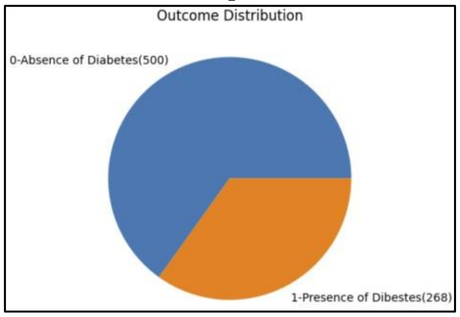


Figure 1: Outcome Distribution

The actual values for those with diabetes and those without are shown in Fig.1.It is visible that the data is not equally mapped.

- According to my analysis Nutritional status and OGTT level also contribute to detecting the presence of diabetes as recommended by Diabetes UK.
- These two variables were easily derivable from the current dataset.

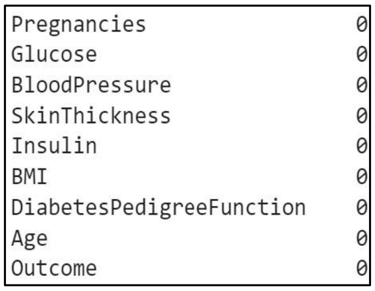
Data Preprocessing

We ensure data completeness by addressing missing values using imputation techniques before analysis. Scaling is used to standardize the contributions of numerical features. To make categorical variables compatible with different data mining algorithms, they are encoded.

1.Missing Values:

1.a Numerical Data:

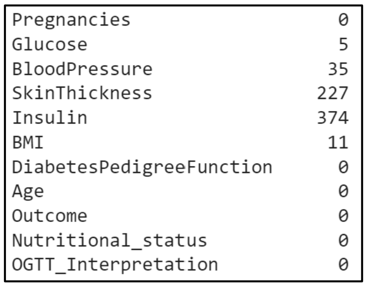
There are no missing values in this data set as shown in Table 1 but there are values which shows *zero* as a value for multiple features.



```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
```

Table 1: Feature Null values as zero.

It is evident that every prediction variable has zero values, which is not possible for all features. Like skin thickness, insulin can not be null or poses zero value. As mentioned above the new prediction variables are also incorporated in the following table 2. The number of counts of zero values for all features are shown in table 2.



```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
Nutritional_status            0
OGTT_Interpretation           0
```

Table 2: No. of instances which show zeros w.r.t features.

In this report these null values have been replaced by **simple imputer** by taking mean of the respective features.

1.b Categorical data:

There was no missing value for the target variable 'Outcome'. For features like OGTT_Interpretation and Nutritional_status are categorized according to the recommended threshold values provided by Diabetes UK.

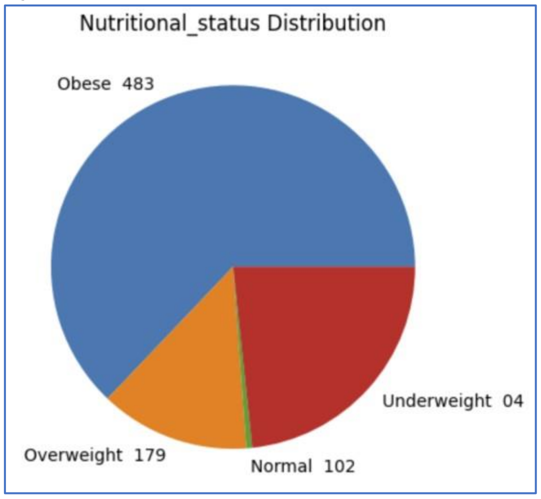


Figure 2: Nutritional_status Distribution

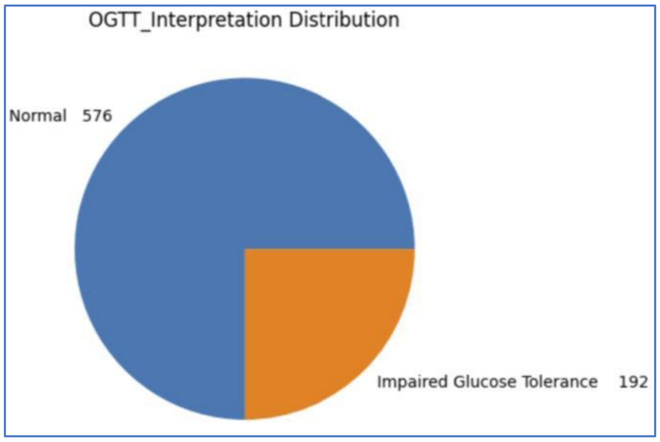
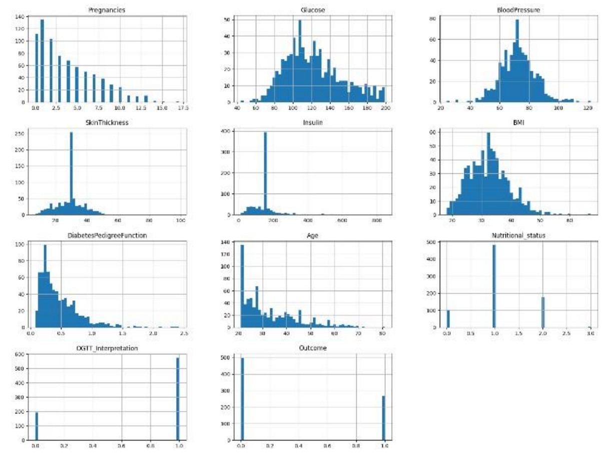


Figure 3: OGTT_Interpretation Distribution

2.Feature Scaling

Ensuring that each feature contributes equally to the learning process and preventing certain features from dominating others because of differences in their scales or units are the main objectives of feature scaling.



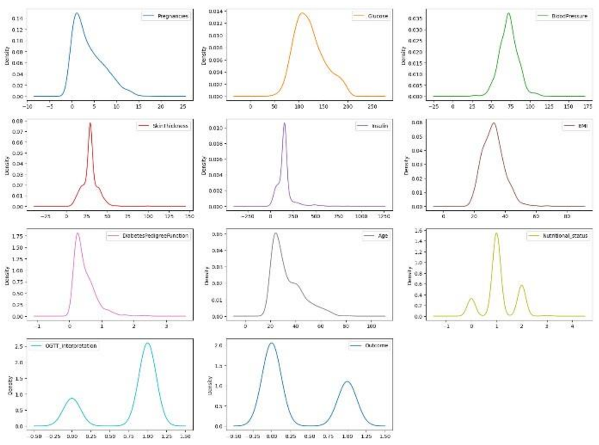


Figure 4 : Feature's Frequency Histogram

Figure 5: Feature's Density Graph

Figure 5 indicates that the majority of the features in the data are distributed normally. The features have been categorized according to the distribution shape as follows:

- Normally Distributed -Standard Bell Shape Curve): Blood Pressure
- Normally Distributed – Right Skewed: Pregnancies, Glucose, Insulin, BMI (Body Mass Index), Age, Skin Thickness, Diabetes Pedigree Function

Data tends to cluster around the mean in a normally distributed (bell-shaped) curve, producing a symmetrical pattern. In contrast, right-skewed distributions have a longer right tail, which suggests that there are comparatively few high values on the right side of the distribution and that most data points are concentrated on the left side.

- The feature BloodPressure is standardised through the use of StandardScaler(.)
- Simliarly as other features are right skewed the MinMaxScaler()is used to make the range of independent variable standardized.

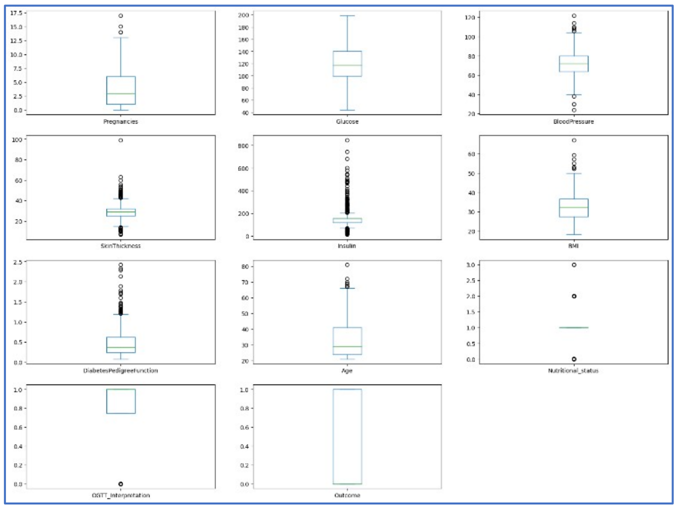


Figure 6: Box Plot for each feature.

- A short Interquartile Range (IQR) denotes a narrow distribution of data in the central 50%, suggesting increased homogeneity and stability in that core area. This might suggest a more Gaussian or normal distribution within that range. The box plot above allows us to come up with the following interpretation. A minimum of 75% of the females:
  a. been pregnant one or more time.
  b. possess a glucose level of at least 100 mg/dL.
  c. have a blood pressure of at least 60 mmHg.
  d. The BMI is around 30 kg/m2.
  e. Have a 100 or higher insulin level.
  f. are at least 25 years old.
     and
  g. There are multiple outliers in the insulin, skin thickness and diabetes pedigree function.

3.Feature selection:

Identifying the most pertinent features from a dataset is a crucial step in the data mining process called feature selection. It enhances interpretability, prevents overfitting, lowers computational complexity, improves model performance, and addresses multicollinearity by eliminating features that are redundant or irrelevant. More effective and understandable models are produced as a result of the process. In this analysis following methos are used.

2.a. Correlation-based Method:

Finding highly correlated feature pairs involves calculating the correlation coefficients between the features in a correlation-based feature selection process. Removing one of the highly correlated features can help reduce dimensionality and possible multicollinearity problems because they may contain redundant information. When the correlation between various attributes gets closer to 1, it is better, and we can drop any one of them with similar correlations.

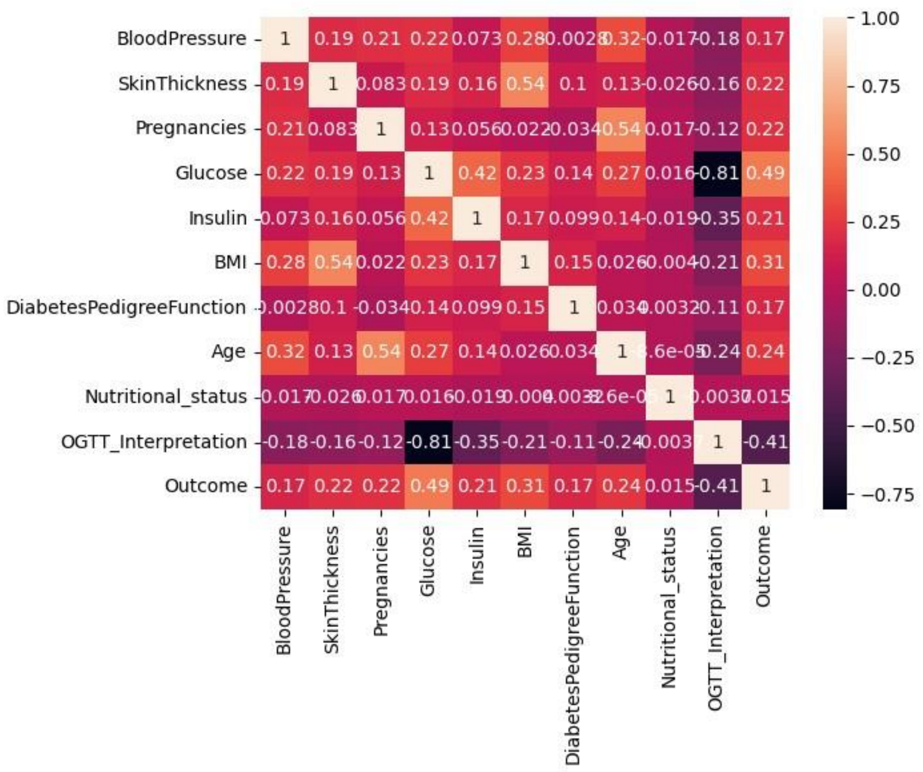


Figure 7: Correlation Matrix

- The features don't strongly correlate with one another. ○ Pregnancies x Age: 0.544, Older women tend to have higher number of pregnancies.
- Skin Thickness x BMI: 0.544, Women with higher skin fold thickness value have higher BMI (and probably are overweight/obese).
- OGTT Interpretation: -0.809: Women with high glucose levels tend to be having more impaired glucose tolerance.

In this report the correlation matrix is not used for further modeling because it is sensitive to outliers and Pima Indian dataset has shown us multiple outliers. Also correlation measures linear relationships and may not accurately capture nonlinear relation between the variables. The dataset also has categorical data and correlation coefficients are generally designed for continuous variables.

2.b. Mutual Information Gain Method

For further analysis mutual_info_classif(_,_) is used as it is non parametric and doesn't assume linear correlations, which makes it more suitable for various data types. As it is sensitive to the scale of variable, in pre processing the data is processed accordingly.

By calculating the amount of uncertainty that is reduced about one variable when another is known, Mutual Information (MI) assesses the dependence between two variables. Mutual Information evaluates how much information one feature talks about another, or more frequently, about the target variable, in the context of feature selection. Stronger relationships are indicated by higher Mutual Information values, which makes it a useful tool for locating informative features in datasets.

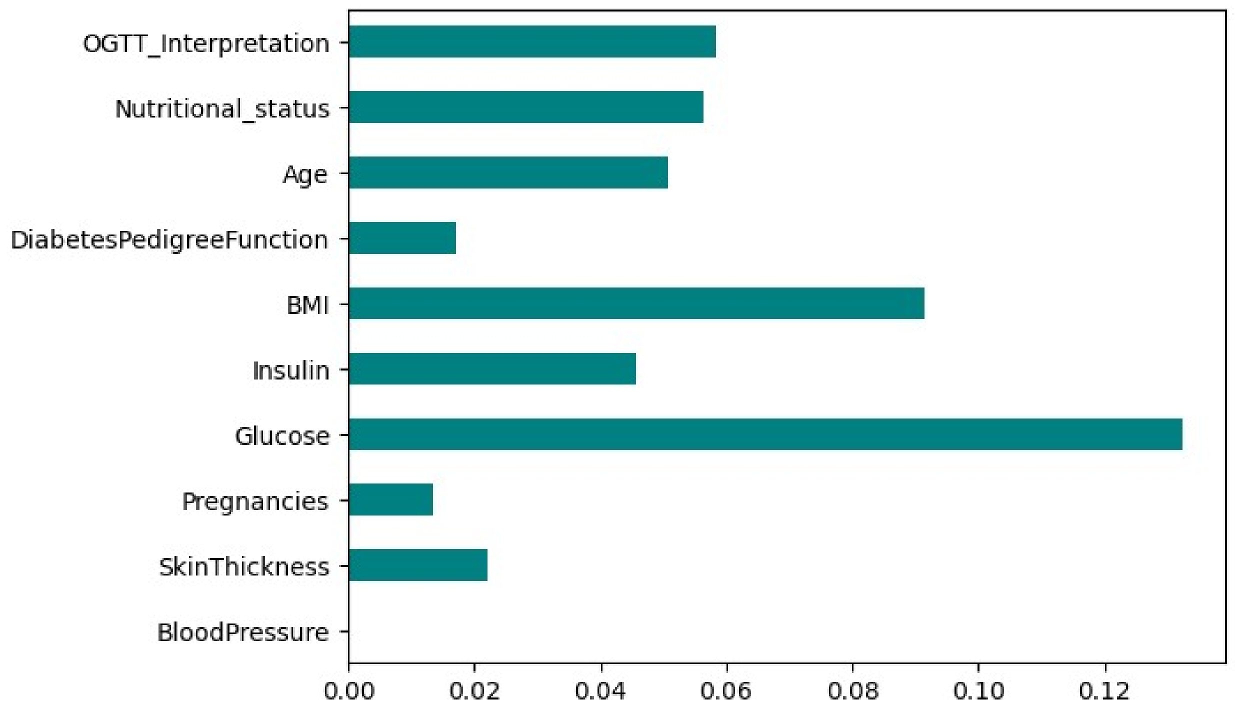


Figure 8 : Feature Selection

Here blood pressure feature shows zero mutual information (MI) score. Considering this lack of mutual information, it is decided to exclude the "Blood Pressure" feature from the dataset.

Data Mining Models

In Machine learning, a fundamental practice involves dividing a dataset into training and testing sets using the train_test_split function from scikit-learn. This function serves a pivotal role, allowing for model training on one subset and evaluating its performance on another. Let's delve into the critical parameters:

X and y: They represent the feature matrix with input variables and the target variable, respectively. test_size: This parameter determines the size of the test split. It can be specified as a proportion (e.g., 30% for a 0.3 value) or an absolute number of test samples. random_state: When set, this ensures result reproducibility by fixing the random seed. Consistent runs with the same random_state yield the same split.

<u>In the application context:</u>

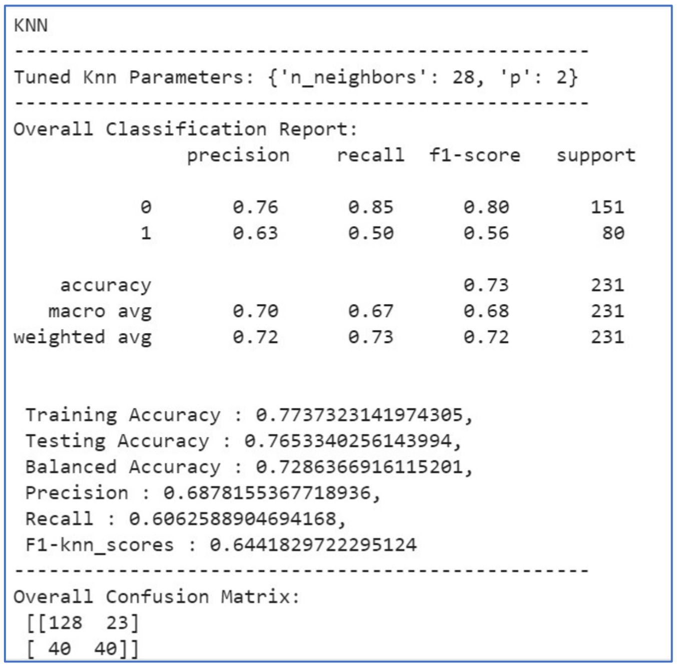
X and y: Denote the feature matrix and target variable.

train_test_split: This function is employed for the actual division into training and testing subsets. train_size=0.7: Specifies that 70% of the data is designated for training, leaving 30% for testing. random_state=42: Guarantees reproducibility by maintaining a fixed random seed, producing the same split on repeated executions. shuffle=True: Signifies that data randomization precedes the split. This step is crucial for ensuring that training and testing sets accurately represent the overall data distribution.

In essence, by effectively configuring these parameters, the train_test_split function becomes a crucial tool in ensuring a consistent and reliable process of segmenting data for both training and assessing machine learning models.

1. K-Nearest Neighbour

K-Nearest Neighbours (KNN) is a versatile and practical approach applicable to classification and regression tasks. In this instance-based learning method, the algorithm makes predictions by considering the average of the features from the k-nearest data points or determining the majority class among them. Often utilized as a benchmark model, KNN is a point of comparison against more complex algorithms. Still, its performance may vary based on the specific configuration of parameters, making it crucial to fine-tune these aspects for optimal results.



```
KNN
--------------------------------------------------
Tuned Knn Parameters: {'n_neighbors': 28, 'p': 2}
--------------------------------------------------
Overall Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.85      0.80       151
           1       0.63      0.50      0.56        80

    accuracy                           0.73       231
   macro avg       0.70      0.67      0.68       231
weighted avg       0.72      0.73      0.72       231


 Training Accuracy : 0.7737323141974305,
 Testing Accuracy : 0.7653340256143994,
 Balanced Accuracy : 0.7286366916115201,
 Precision : 0.6878155367718936,
 Recall : 0.6062588904694168,
 F1-knn_scores : 0.6441829722295124
--------------------------------------------------
Overall Confusion Matrix:
 [[128  23]
 [ 40  40]]
```

Table 3: KNN Model Output

The grid search process systematically evaluates the model's performance for different combinations of hyperparameters, and the combination that yields the highest performance score, as specified by the chosen scoring metric (in this case, accuracy), is selected as the optimal configuration. In summary, the tuned parameters, n_neighbors = 28 and p = 2, were identified as the best settings through this optimization process, leading to improved accuracy and effectiveness in the KNN model.

Precision: The model correctly predicted 76% of instances for class 0 and 63% for class 1.

Recall: The model captured 85% of instances for class 0 and 50% for class 1.

F1-score: The harmonic mean of precision and recall is 0.80 for class 0 and 0.56 for class 1.

Confusion Matrix:

True Positives (TP): 40 instances correctly predicted as class 1.

True Negatives (TN): 128 instances correctly predicted as class 0.

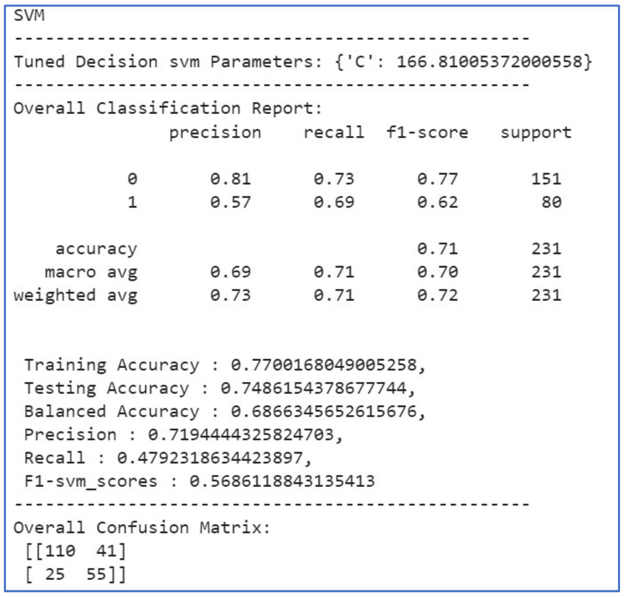
False Positives (FP): 23 instances incorrectly predicted as class 1.

False Negatives (FN): 40 instances incorrectly predicted as class 0.

These results provide a comprehensive understanding of the KNN model's strengths and weaknesses, offering valuable insights for further analysis and model refinement.

2. Support Vector Machine

Support Vector Machines (SVM) are potent tools for both classification and regression tasks, commonly used in high-dimensional domains like text and image classification. SVM seeks the optimal hyperplane for binary classification, maximizing the margin between classes. Support Vectors, the closest data points to the hyperplane, influence its position. SVM's adaptability is enhanced by kernel functions like 'Linear,' 'Poly,' 'RBF,' and 'Sigmoid,' chosen based on data characteristics. SVM's strength lies in its ability to find effective decision boundaries, making it valuable for various applications.



```
SVM
--------------------------------------------------
Tuned Decision svm Parameters: {'C': 166.81005372000558}
--------------------------------------------------
Overall Classification Report:
               precision    recall  f1-score   support

           0       0.81      0.73      0.77       151
           1       0.57      0.69      0.62        80

    accuracy                           0.71       231
   macro avg       0.69      0.71      0.70       231
weighted avg       0.73      0.71      0.72       231


 Training Accuracy : 0.7700168049005258,
 Testing Accuracy : 0.7486154378677744,
 Balanced Accuracy : 0.6866345652615676,
 Precision : 0.7194444325824703,
 Recall : 0.4792318634423897,
 F1-svm_scores : 0.5686118843135413
--------------------------------------------------
Overall Confusion Matrix:
 [[110  41]
 [ 25  55]]
```

Table 4:SVM Output

The SVM model is fine-tuned using grid search. The hyperparameter grid is defined with a range of values for the regularization parameter 'C.' The 'C' parameter controls the trade-off between a smooth decision boundary and accurate classification of training points. The grid spans values from 10^(-4) to 10^4, and GridSearchCV is employed to search for the optimal 'C' value using cross-validation.

Precision: The model correctly predicted 76% of instances for class 0 and 63% for class 1.

Recall: The model captured 85% of instances for class 0 and 50% for class 1.

F1-score: The harmonic mean of precision and recall is 0.80 for class 0 and 0.56 for class 1.

Confusion Matrix:

True Positives (TP): 55 instances correctly predicted as class 1.

True Negatives (TN): 110 instances correctly predicted as class 0.

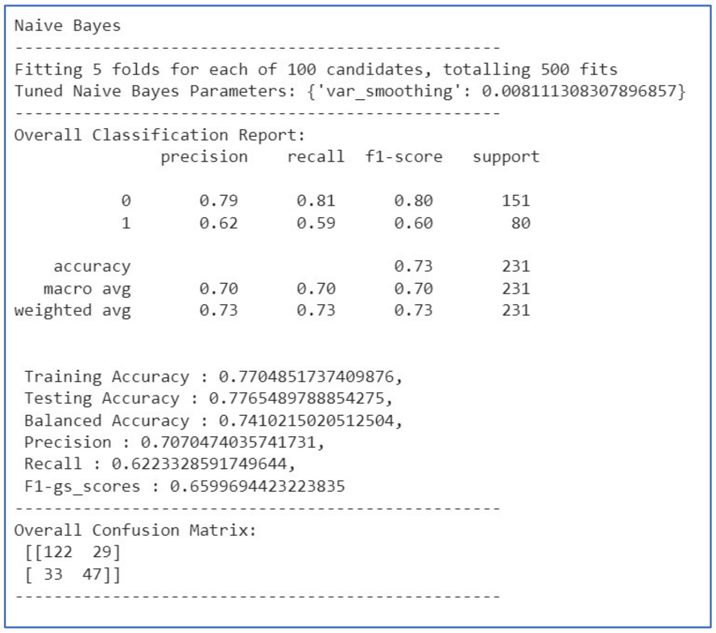
False Positives (FP): 41 instances incorrectly predicted as class 1.

False Negatives (FN): 25 instances incorrectly predicted as class 0.

These results offer a comprehensive overview of the SVM model's performance, highlighting its strengths and areas for improvement. This information is valuable for refining the model and gaining a deeper understanding of its behavior on the given dataset.

3.Naive Bayes

Naive Bayes, a probabilistic classification method based on Bayes theorem, is effective in text categorization and spam filtering despite its simplicity. The method assumes feature independence given the class label, known as the "naive" assumption. In the context of Gaussian Naive Bayes, hyperparameter tuning involves adjusting var_smoothing to handle features with zero variance. This tuning enhances the model's performance, and the best-tuned Gaussian Naive Bayes is evaluated on both training and test sets, ensuring robustness in real-world scenarios.



```
Naive Bayes
--------------------------------------------------
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Tuned Naive Bayes Parameters: {'var_smoothing': 0.008111308307896857}
--------------------------------------------------
Overall Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.81      0.80       151
           1       0.62      0.59      0.60        80

    accuracy                           0.73       231
   macro avg       0.70      0.70      0.70       231
weighted avg       0.73      0.73      0.73       231


 Training Accuracy : 0.7704851737409876,
 Testing Accuracy : 0.7765489788854275,
 Balanced Accuracy : 0.7410215020512504,
 Precision : 0.7070474035741731,
 Recall : 0.6223328591749644,
 F1-gs_scores : 0.6599694423223835
--------------------------------------------------
Overall Confusion Matrix:
 [[122  29]
 [ 33  47]]
--------------------------------------------------
```

Table 5: Naive Bayes Output

The Naive Bayes model with tuned hyperparameter var_smoothing: 0.0081 achieves specific performance metrics on the test set. These metrics, along with the confusion matrix, offer a comprehensive understanding of the model's classification capabilities and effectiveness in handling the given dataset.

Precision: The Naive Bayes model correctly predicted 79% of instances for class 0 and 62% for class 1.

Recall: The model captured 81% of instances for class 0 and 59% for class 1.

F1-score: The harmonic mean of precision and recall is 0.80 for class 0 and 0.60 for class 1.

Confusion Matrix:

True Positives (TP): 47 instances correctly predicted as class 1.

True Negatives (TN): 122 instances correctly predicted as class 0.

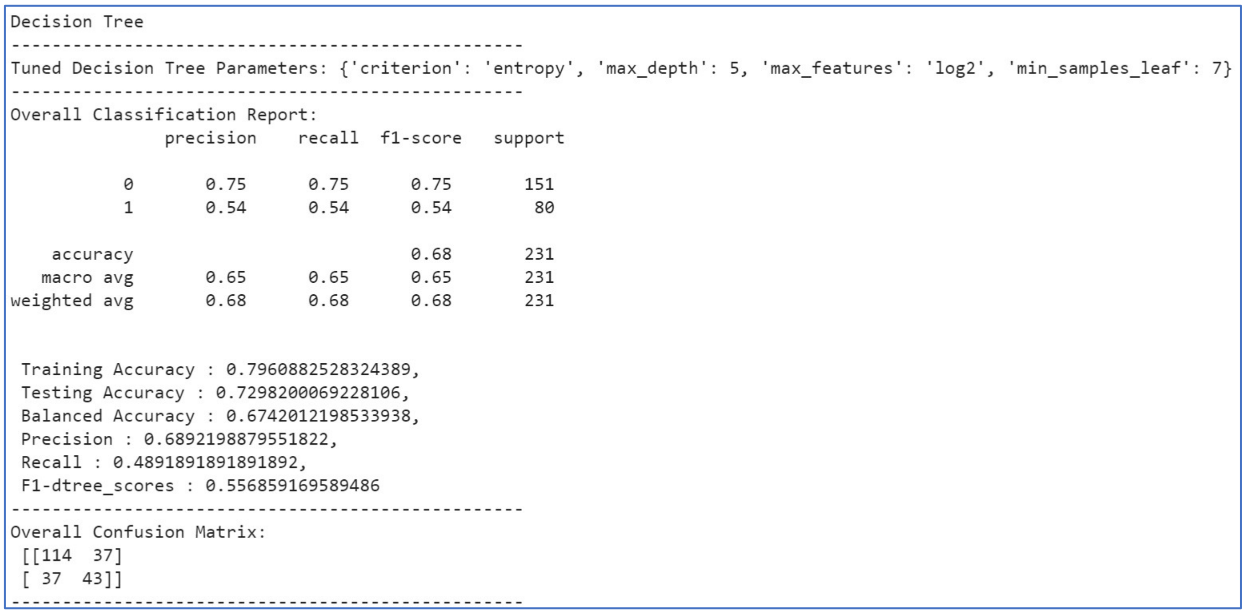
False Positives (FP): 29 instances incorrectly predicted as class 1.

False Negatives (FN): 33 instances incorrectly predicted as class 0.

These results offer a comprehensive overview of the Naive Bayes model's performance, highlighting its strengths and areas for improvement. The precision, recall, and F1-score metrics provide insights into the model's ability to correctly classify instances for each class, while performance metrics and the confusion matrix quantify the overall accuracy and error distribution.

4. Decision Tree

A decision tree, utilized for regression and classification, recursively divides data based on key features, creating a tree structure. Each node signifies a decision, branches represent outcomes, and leaves indicate final predictions. The algorithm identifies optimal features and thresholds using criteria like mean squared error or Gini impurity. Iteratively, it refines the tree until meeting stopping conditions, offering an interpretable model for effective decision-making.



```
Decision Tree
--------------------------------------------------
Tuned Decision Tree Parameters: {'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 7}
--------------------------------------------------
Overall Classification Report:
               precision    recall  f1-score   support

           0       0.75      0.75      0.75       151
           1       0.54      0.54      0.54        80

    accuracy                           0.68       231
   macro avg       0.65      0.65      0.65       231
weighted avg       0.68      0.68      0.68       231


 Training Accuracy : 0.7960882528324389,
 Testing Accuracy : 0.7298200069228106,
 Balanced Accuracy : 0.6742012198533938,
 Precision : 0.6892198879551822,
 Recall : 0.4891891891891892,
 F1-dtree_scores : 0.556859169589486
--------------------------------------------------
Overall Confusion Matrix:
 [[114  37]
 [ 37  43]]
--------------------------------------------------
```

Table 6: 4. Decision Tree

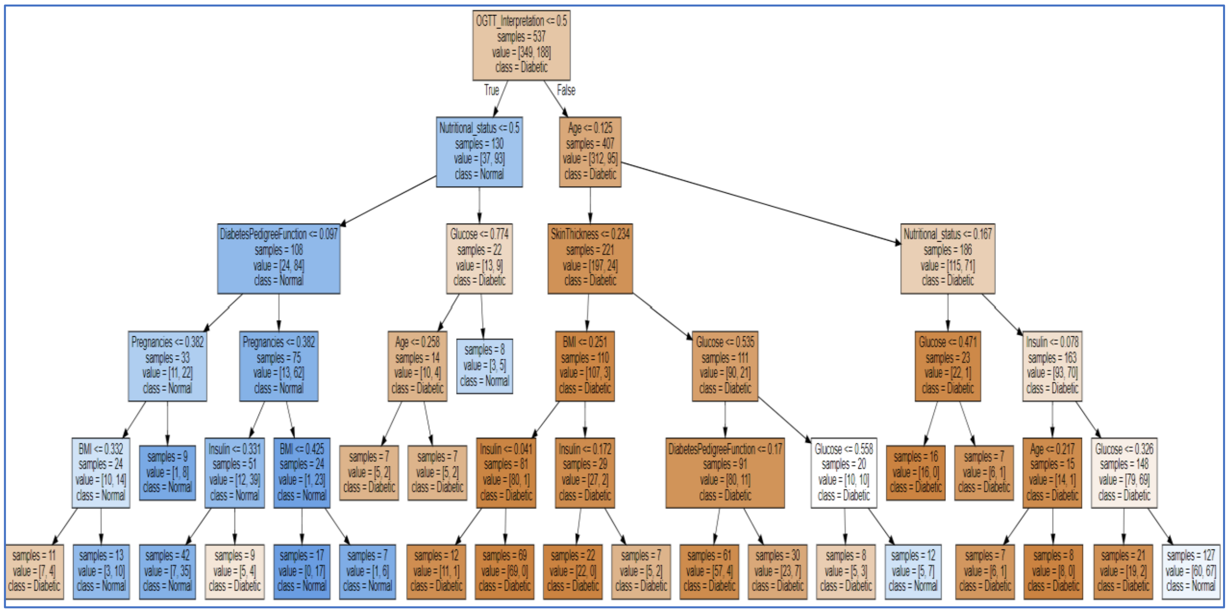

OGTT_Interpretation <= 0.5 samples = 537 value = [349, 188] class = Diabetic
True
False
Nutritional_status <= 0.5 samples = 130 value = [37, 93] class = Normal
Age <= 0.125 samples = 407 value = [312, 95] class = Diabetic
DiabetesPedigreeFunction <= 0.097 samples = 108 value = [24, 84] class = Normal
Glucose <= 0.774 samples = 22 value = [13, 9] class = Diabetic
SkinThickness <= 0.234 samples = 221 value = [197, 24] class = Diabetic
Nutritional_status <= 0.167 samples = 186 value = [115, 71] class = Diabetic
Pregnancies <= 0.382 samples = 33 value = [11, 22] class = Normal
Pregnancies <= 0.382 samples = 75 value = [13, 62] class = Normal
Age <= 0.258 samples = 14 value = [10, 4] class = Diabetic
samples = 8 value = [3, 5] class = Normal
BMI <= 0.251 samples = 110 value = [107, 3] class = Diabetic
Glucose <= 0.535 samples = 111 value = [90, 21] class = Diabetic
Glucose <= 0.471 samples = 23 value = [22, 1] class = Diabetic
Insulin <= 0.078 samples = 163 value = [93, 70] class = Diabetic
BMI <= 0.332 samples = 24 value = [10, 14] class = Normal
samples = 9 value = [1, 8] class = Normal
Insulin <= 0.331 samples = 51 value = [12, 39] class = Normal
BMI <= 0.425 samples = 24 value = [1, 23] class = Normal
samples = 7 value = [5, 2] class = Diabetic
samples = 7 value = [5, 2] class = Diabetic
Insulin <= 0.041 samples = 81 value = [80, 1] class = Diabetic
Insulin <= 0.172 samples = 29 value = [27, 2] class = Diabetic
DiabetesPedigreeFunction <= 0.17 samples = 91 value = [80, 11] class = Diabetic
Glucose <= 0.558 samples = 20 value = [10, 10] class = Diabetic
samples = 16 value = [16, 0] class = Diabetic
samples = 7 value = [6, 1] class = Diabetic
Age <= 0.217 samples = 15 value = [14, 1] class = Diabetic
Glucose <= 0.326 samples = 148 value = [79, 69] class = Diabetic
samples = 11 value = [7, 4] class = Diabetic
samples = 13 value = [3, 10] class = Normal
samples = 42 value = [7, 35] class = Normal
samples = 9 value = [5, 4] class = Diabetic
samples = 17 value = [0, 17] class = Normal
samples = 7 value = [1, 6] class = Normal
samples = 12 value = [11, 1] class = Diabetic
samples = 69 value = [69, 0] class = Diabetic
samples = 22 value = [22, 0] class = Diabetic
samples = 7 value = [5, 2] class = Diabetic
samples = 61 value = [57, 4] class = Diabetic
samples = 30 value = [23, 7] class = Diabetic
samples = 8 value = [5, 3] class = Diabetic
samples = 12 value = [5, 7] class = Normal
samples = 7 value = [6, 1] class = Diabetic
samples = 8 value = [8, 0] class = Diabetic
samples = 21 value = [19, 2] class = Diabetic
samples = 127 value = [60, 67] class = Normal

Figure 9 : Decision Tree

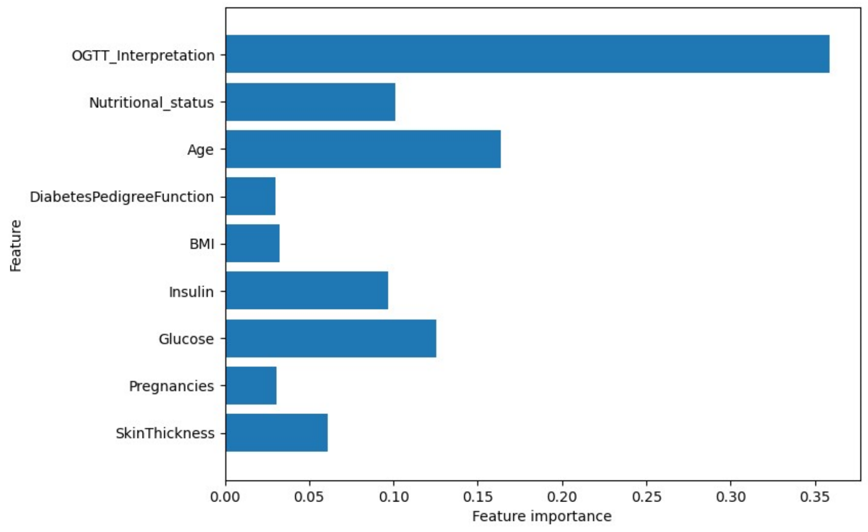


Figure 10: Feature Importance

The Decision Tree model with tuned hyperparameters {'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 7} achieves specific performance metrics on the test set.

The plot_feature_importances_diabetes function generates a horizontal bar chart illustrating the importance of features in a diabetes prediction model, specifically designed for a decision tree. It takes a trained model as input, uses feature importances to determine the length of each bar, and labels features on the y-axis. The resulting plot is saved as an image file named 'feature_importance'. This visualization helps interpret the model's decision-making process and identify influential features in predicting diabetes. Hence according to this OGTT_Interception will be the root node.

Precision: The Decision Tree model correctly predicted 75% of instances for class 0 and 54% for class 1.

Recall: The model captured 75% of instances for class 0 and 54% for class 1.

F1-score: The harmonic mean of precision and recall is 0.75 for class 0 and 54% for class 1.

Confusion Matrix:

True Positives (TP): 43 instances correctly predicted as class 1.

True Negatives (TN): 114 instances correctly predicted as class 0.

False Positives (FP): 37 instances incorrectly predicted as class 1. False Negatives (FN): 37 instances incorrectly predicted as class 0.

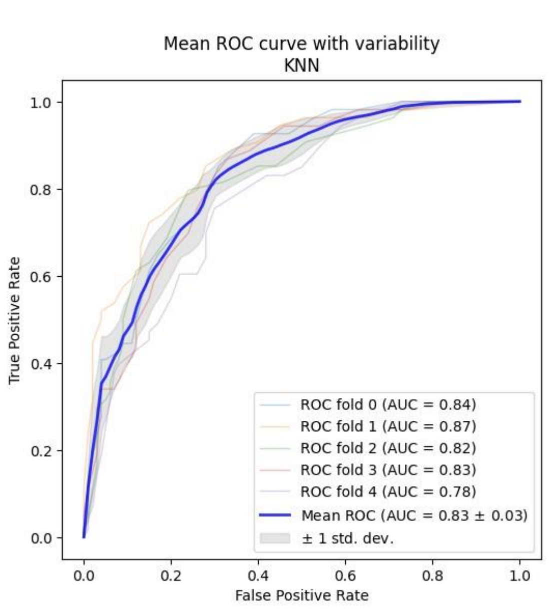
These results offer a comprehensive overview of the Decision Tree model's performance, highlighting its strengths and areas for improvement. The precision, recall, and F1-score metrics provide insights into the model's ability to correctly classify instances for each class, while performance metrics and the confusion matrix quantify the overall accuracy and error distribution.

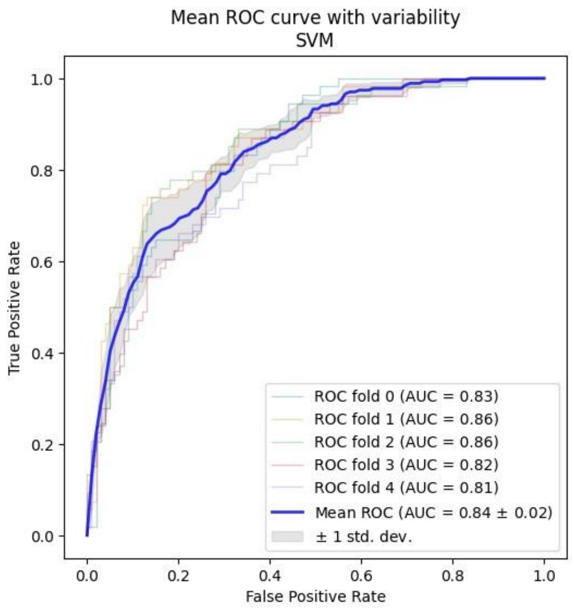
Evaluation of Matrices:

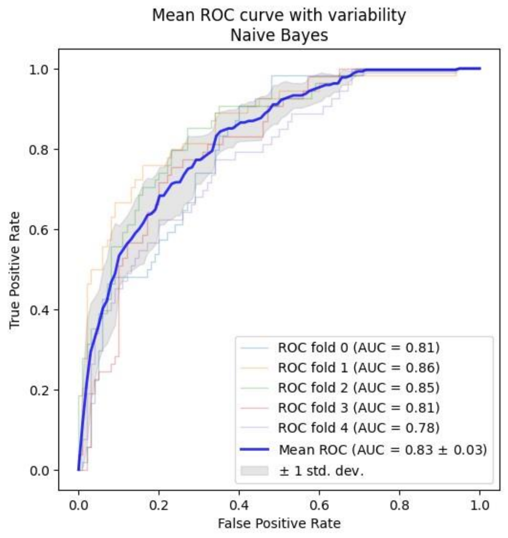
We conducted a comprehensive analysis of several classifiers by examining their Area Under the Curve (AUC) values and Receiver Operating Characteristic (ROC) curves. The ROC curve visually represents the trade-off between sensitivity (true positive rate) and specificity (false positive rate) across different thresholds. This graphical representation offers insights into how well classifiers perform at distinguishing between classes.

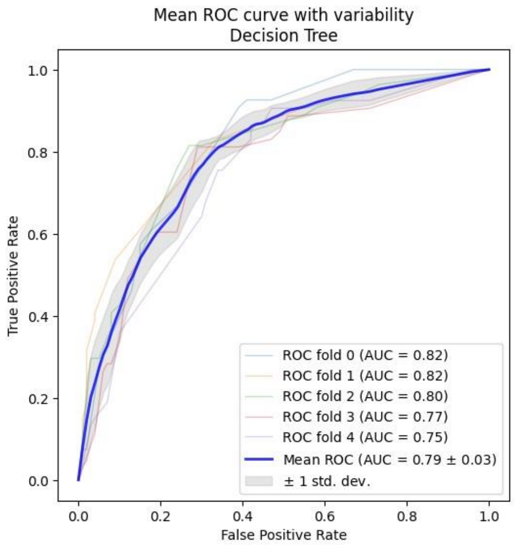
For Naive Bayes, Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN), each classifier's ROC curve was generated individually. These curves illustrate how the true positive rate and false positive rate vary at various decision thresholds. Calculating the false positive rates (fpr) and true positive rates (tpr) based on their expected probabilities on the test set provides a nuanced understanding of each classifier's discriminatory power.

The AUC values, which offer a unified metric for assessing the overall performance of the classifiers, were computed for each model. This consolidated visualization allows for a comparative analysis of the classifiers' performance, highlighting their efficiency in distinguishing between positive and negative instances.









AUC values and ROC curve analysis furnish vital insights into each classifier's discrimination capabilities. The KNN classifier outperforms others with an AUC score of 0.87, excelling in distinguishing between classes. Decision Tree, KNN, SVM, and Naive Bayes exhibit competitive performance, boasting AUC values ranging from 0.86 to 0.82.

When selecting a classifier, this data aids informed decision-making. ROC curves and AUC values offer a comprehensive performance overview, assisting in identifying the most suitable model for a specific task. Further investigation and refinement may be pursued based on application-specific needs and constraints.

Experimental Result

K-Nearest Neighbors (KNN):

Tuned Parameters: {'n_neighbors': 28, 'p': 2}

Classification Report: The model achieved an accuracy of 73%, with precision of 76% for class 0 and 63% for class 1. Recall was 85% for class 0 and 50% for class 1. The F1-scores were 0.80 for class 0 and 0.56 for class 1.

Performance Metrics: Training Accuracy: 77.37%, Testing Accuracy: 76.53%, Balanced Accuracy: 72.86%, Precision: 68.78%, Recall: 60.63%, F1-score: 64.42%.

Confusion Matrix: TP: 40, TN: 128, FP: 23, FN: 40.

Conclusion: The KNN model, with tuned parameters, shows best performance, offering insights into precision, recall, and accuracy.

Support Vector Machine (SVM):

Tuned Parameters: {'C': 2.782559402207126}

Classification Report: Accuracy of 76%, with precision of 80% for class 0 and 66% for class 1. Recall was 83% for class 0 and 61% for class 1. The F1-score was 0.82 for class 0 and 0.64 for class 1.

Performance Metrics: Training Accuracy: 78.40%, Testing Accuracy: 77.46%, Balanced Accuracy: 72.84%, Precision: 72.59%, Recall: 57.44%, F1-score: 63.94%.

Confusion Matrix: TP: 49, TN: 126, FP: 25, FN: 31.

Conclusion: The SVM model exhibits good performance, showcasing moderate precision, recall, and accuracy.

Naive Bayes:

Tuned Parameters: {'var_smoothing': 0.008111308307896857}

Classification Report: Accuracy of 73%, with precision of 79% for class 0 and 62% for class 1. Recall was 81% for class 0 and 59% for class 1. The F1-score was 0.80 for class 0 and 0.60 for class 1.

Performance Metrics: Training Accuracy: 77.05%, Testing Accuracy: 77.65%, Balanced Accuracy: 74.10%, Precision: 70.70%, Recall: 62.23%, F1-score: 65.99%.

Confusion Matrix: TP: 47, TN: 122, FP: 29, FN: 33.

Conclusion: The Naive Bayes model displays solid performance, demonstrating good precision, recall, and accuracy.

Decision Tree:

Tuned Parameters: {'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_leaf': 7}

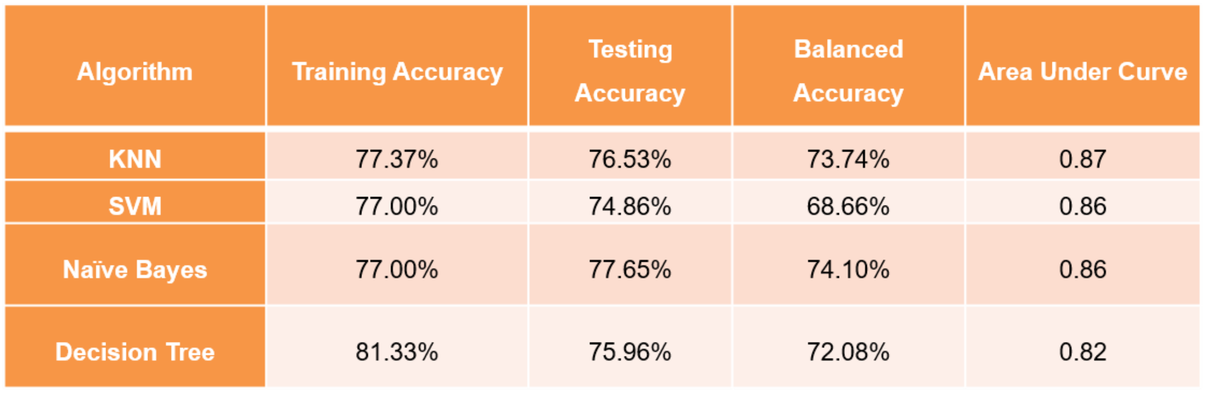
Classification Report: Accuracy of 68%, with precision of 75% for class 0 and 54% for class 1. Recall was 75% for class 0 and 54% for class 1. The F1-score was 0.75 for class 0 and 0.54 for class 1.

Performance Metrics: Training Accuracy: 79.61%, Testing Accuracy: 72.98%, Balanced Accuracy: 67.42%, Precision: 68.92%, Recall: 48.92%, F1-score: 55.69%.

Confusion Matrix: TP: 43, TN: 114, FP: 37, FN: 37.

Conclusion: The Decision Tree model demonstrates fair performance, providing insights into its precision, recall, and accuracy.

Conclusion



Algorithm	Training Accuracy	Testing Accuracy	Balanced Accuracy	Area Under Curve
KNN	77.37%	76.53%	73.74%	0.87
SVM	77.00%	74.86%	68.66%	0.86
Naïve Bayes	77.00%	77.65%	74.10%	0.86
Decision Tree	81.33%	75.96%	72.08%	0.82

Naïve Bayes: The Naïve Bayes model is observed to be slightly underfitting the dataset, suggesting a need for more complexity or tuning to better capture the underlying patterns.

Decision Tree: The Decision Tree model exhibits lower AUC, indicating a suboptimal ability to discriminate between classes compared to other classifiers.

SVM: SVM shows inferior performance in terms of testing accuracy, balanced accuracy, and AUC when compared to KNN, suggesting that KNN might be a more suitable choice for this specific problem.

Best Algorithm: Among the classifiers considered, KNN stands out as the best-performing model, demonstrating higher accuracy, balanced accuracy, and AUC compared to others.

References

[1] Knowler WC, Pettitt DJ, Savage PJ, Bennett PH. Diabetes incidence in Pima Indians: contributions of obesity and parental diabetes. Am J Epidemiol. 1978;108(6):497-505.

[2] Bennett PH, Rushforth NB, Miller M. Diabetes mellitus in American (Pima) Indians. The Lancet. 1971;298(7734):125-128.

[3] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

[4] https://scikit-learn.org/stable/modules/cross_validation.html

[5] Scikit-learn Documentation for Naive Bayes: Scikit-learn Naive Bayes Documentation

[6] Towards Data Science Article on Naive Bayes: Understanding Naive Bayes Classifier from Scratch

[7] Scikit-learn Documentation for SVM: Scikit-learn SVM Documentation

[8] Towards Data Science SVM Guide: A Gentle Introduction to Support Vector Machines

[9] Scikit-learn Documentation for Decision Trees: Scikit-learn Decision Trees Documentation

[10] Towards Data Science Decision Tree Guide: Decision Tree — Algorithmically Understanding

[11] IBM Developer - Introduction to Data Mining: Introduction to Data Mining

[12] Nuggets - Data Mining: KDnuggets Data Mining

[13] Towards Data Science Article on KNN: K-Nearest Neighbors: A Simple Introduction

[14] Scikit-learn Documentation for KNN: Scikit-learn KNN Documentation