SUPERVISED LEARNING ALGORITHM FOR PREDICTION OF TYPE 2 DIABETES MELLITUS DISEASE

Chenreddy Gouthami, Venkat Vankunavath, Krishna Reddy Seelam
Assistance professor, Assistance professor, Assistance professor
Department Of CSE
Sree Dattha Institute Of Engineering and Science

ABSTRACT

Diabetes is a chronic metabolic disorder characterized by high blood sugar levels over an extended period. It often leads to various health complications, known as comorbidities, which can include cardiovascular disease, kidney disease, neuropathy, and more. Detecting and predicting comorbidities in individuals with diabetes is crucial for timely intervention and improved patient outcomes. Early efforts to address comorbidities of diabetes were primarily based on clinical observations and statistical analyses of patient data. These approaches, while valuable, had limitations in their ability to handle the complexity and variability of comorbidity patterns. The advent of machine learning brought about a paradigm shift in this domain, enabling the development of more sophisticated and accurate prediction models. The problem of detecting and predicting comorbidities of diabetes using machine learning techniques involves utilizing data from individuals with diabetes to build models that can identify the likelihood of developing specific comorbid conditions. This task requires processing diverse datasets, including patient demographics, medical history, laboratory results, and potentially genetic information, to provide actionable insights for healthcare providers. In the past, the detection and prediction of comorbidities in diabetes were primarily reliant on manual analysis by healthcare professionals. While they relied on their expertise and experience, this approach was limited by human subjectivity and the inability to process vast amounts of data efficiently. It also lacked the ability to discern intricate patterns that machine learning models are adept at uncovering. The need for accurate and timely detection and prediction of comorbidities in individuals with diabetes is paramount. Comorbidities significantly impact the quality of life for those with diabetes, and early intervention can lead to more effective treatment strategies. Machine learning techniques offer the potential to analyze large and complex datasets, identifying subtle patterns and relationships that may be overlooked by human observers. This can lead to more personalized and effective healthcare interventions.

Keywords: Diabetes Mellitus Disease, Chronic Metabolic Disorder, High Blood Sugar Levels, Comorbidities, Cardiovascular Disease, Kidney Disease, Neuropathy, Detecting, Predicting, Clinical Observations, Statistical Analyses, Machine Learning, Prediction Models, Diverse Datasets, Patient Demographics, Medical History, Laboratory Results, Genetic Information.

1. INTRODUCTION

The historical trajectory of addressing comorbidities in diabetes reveals a progression from early clinical observations and statistical analyses to the transformative integration of machine learning techniques. In the initial stages, efforts to detect and predict comorbidities relied on the expertise of healthcare professionals who manually analyzed patient data. However, this approach had inherent limitations, including subjectivity, inefficiency in processing vast datasets, and an inability to discern intricate patterns. With the advent of machine learning, a paradigm shift occurred, enabling the development of sophisticated and accurate prediction models. This marked a significant departure from traditional methods, as machine learning techniques could leverage diverse datasets, encompassing patient demographics, medical history, laboratory results, and even genetic information. The problem evolved into utilizing these datasets to build models capable of identifying the likelihood of specific comorbid conditions in individuals with diabetes The motivation driving this research is deeply rooted in the imperative to enhance the healthcare outcomes of individuals with diabetes by addressing the complexities of comorbidities through advanced machine learning techniques. Diabetes, as a chronic metabolic disorder, often leads to various comorbid conditions, impacting the overall well-being of affected individuals. Early efforts relying on clinical observations and statistical analyses faced limitations in their ability to handle the intricate and variable nature of comorbidity patterns. The advent of machine learning offers a promising avenue to overcome these limitations, presenting an opportunity to construct sophisticated prediction models that can analyze diverse datasets comprehensively. The challenge addressed by this research revolves around the detection and prediction of comorbidities in individuals with diabetes, a chronic metabolic disorder characterized by elevated blood sugar levels. While clinical observations and statistical analyses were early methods employed for this purpose, they faced limitations in handling the complexity and variability of comorbidity patterns. The introduction of machine learning techniques marked a transformative shift, enabling the development of more sophisticated prediction models. The problem statement involves utilizing diverse datasets, including patient demographics, medical history, laboratory results, and potentially genetic information, to construct models that can identify the likelihood of developing specific comorbid conditions in individuals with diabetes. Historically, the detection and prediction of diabetes-related comorbidities heavily relied on manual analysis by healthcare professionals, leading to subjective outcomes, inefficiencies in processing extensive data, and a failure to uncover intricate patterns that machine learning models excel at discerning, the adaptability and precision offered by machine learning applications in this context have the potential to significantly improve patient care, enhance healthcare efficiency, and contribute to the broader goal of promoting individual well-being in the face of chronic metabolic disorders.

2. LITERATURE SURVEY

Kandhasamy and Balamurali [1] proposed multiple classifiers SVM, J48, K-Nearest Neighbors (KNN), and Random Forest. The classification was performed on a dataset taken from the UCI repository. The results of the classifiers were compared based on the values of the accuracy, sensitivity, and specificity. The classification was done in two cases, when the dataset is pre-processed and without preprocessing by using 5-fold cross validation. The authors didn't explain the pre-processing step applied on the dataset, they just mentioned that the noise was removed from the data. They reported that the decision tree J48 classifier has the highest accuracy rate being 73.82% without pre-processing, while the classifiers KNN (k = 1) and Random Forest showed the highest accuracy rate of 100% after pre-processing the data.

Yuvaraj and Sripreethaa [2] presented an application for diabetes prediction using three different ML algorithms including Random Forest, Decision Tree, and the Naïve Bayes. The Pima Indian Diabetes dataset (PID) was used after pre-processing it. The authors didn't mention how the data was pre-processed; however they discussed the Information Gain method used for feature selection to extract the relevant features. They used only eight main attributes among 13 In addition, they divided the dataset into 70% for training and 30% for testing. The results showed that the random forest algorithm had the highest accuracy rate of 94%.

Furthermore, Tafa [3] proposed a new integrated improved model of SVM and Naïve Bayes for predicting the diabetes. The model was evaluated using a dataset collected from three different locations in Kosovo. The dataset contains eight attributes and 402 patients where 80 patients had type 2 diabetes. Some attributes utilized in this study have not been investigated before, including the regular diet, physical activity, and family history of diabetes. The authors didn't mention whether the data was pre-processed or not. For the validation test, they split the dataset into 50% for each of the training and testing sets. The proposed combined algorithms have improved the accuracy of the prediction to reach 97.6%. This value was compared with the performance of SVM and Naïve Bayes achieving 95.52% and 94.52%, respectively.

In addition, Deepti and Dilip [4] used Decision Tree, SVM, and Naive Bayes classifiers to detect diabetes. The aim was to identify the classifier with the highest accuracy. The Pima Indian dataset was used for this study. The partition of the dataset is done by means of 10-folds cross-validation. The authors didn't discuss the data preprocessing. The performance was evaluated using the measures of the accuracy, the precision, recall, and the F-measure. The highest accuracy was obtained by the Naive Bayes, which reached 76.30%.

Mercaldo [5] proposed six different classifiers. The classifiers are J48, Multilayer Perceptron, HoeffdingTree, JRip, BayesNet, and Random Forest. The Pima Indian dataset was also utilized for this study. The authors didn't mention a preprocessing step, however, they employed two algorithms, GreedyStepwise and BestFirst, to determine the discriminatory attributes that help in increase the classification performance. Four attributes have been selected, namely body mass index, plasma glucose concentration, diabetes pedigree function, and age. A 10 fold-cross validation is applied to the dataset. The comparison between the classifiers was made based on the value of the precision, the recall, and the F-Measure. The result showed the precision value equals to 0.757, recall equals to 0.762, and F-measure equals to 0.759 using the Hoeffding Tree algorithm. This is the highest performance compared to the others.

In addition to the other studies, Negi and Jaiswal [5] aimed to apply the SVM to predict diabetes. The Pima Indians and Diabetes 130-US datasets were used as a combined dataset. The motivation of this study was to validate the reliability of the results as other researchers often used a single dataset. The dataset contains 102,538 samples and 49 attributes where 64,419 were positive samples and 38,115 were negative samples. The authors didn't discuss the attributes used in this study. The dataset is pre-processed by replacing the missing values and out of range data by zero, the non-numerical values are changed to numerical values, and finally the data is normalized between 0 and 1. Different feature selection methods were used prior to the application of the SVM model. The Fselect script from LIBSVM package selected four attributes, while Wrapper and Ranker methods (from Weka Tool) selected nine and 20 attributes, respectively. For the validation process, the authors used 10-fold cross validation technique. By using a combined dataset, the diabetes prediction might be more reliable, with an accuracy of 72%.

Moreover, Olaniyi and Adnan [6] used a Multilayer Feed-Forward Neural Network. The back-propagation algorithm was used for training the algorithm. The aim was to improve the accuracy of diabetes prediction. The Pima Indian Diabetes database was used the authors normalized the dataset before processing to the classification in order to obtain a numerical stability. It consisted of dividing each sample attributes by their corresponding amplitude to make all the dataset values between 0 and 1. After that, the dataset is divided into 500 samples for a training set and 268 for the testing set. The accuracy obtained was 82% which is considered as a high accuracy rate.

Soltani and Jafarian [7] used the Probabilistic Neural Network (PNN) to predict diabetes. The algorithm was applied to the Pima Indian dataset. The authors didn't apply any pre-processing technique. The dataset is divided into 90% for the training set and 10% for the testing set. The proposed technique achieved accuracies of 89.56%, 81.49% for the training and testing data, respectively.

Rakshit [8] pre-processed the dataset by normalizing all the sample attributes values using the mean and the standard deviation of each attribute in order to obtain a numerical stability. In addition, they extracted the relevant features using the correlation. However, the authors didn't mention these discriminatory features. The dataset was split into a training set containing 314 samples and a testing set comprising 78 samples. The result of this model achieved the highest accuracy of 83.3% when compared to other accuracies obtained from the previous studies.

3.PROPOSED SYSTEM

3.1 Overview

The Python code outlines a machine learning project for the detection and prediction of comorbidities of diabetes using various techniques.

Here's a detailed explanation of the steps involved:

- Dataset Loading: The project begins by allowing the user to upload the comorbidity diabetes dataset. The Tkinter GUI provides a file dialog, enabling the user to select the dataset file. Once loaded, the file path is displayed on the GUI, and the initial few rows of the dataset are presented.
- Dataset Preprocessing: After loading the dataset, the user initiates the preprocessing step. This involves handling null values by replacing them with zeros. The dataset is then converted into NumPy arrays for further processing. Features and labels are separated, and the dataset is normalized using standard scaling.
- Train-Test Split: The normalized dataset is split into training and testing sets using the train_test_split function from Scikit-Learn. The script provides information about the total number of records in the dataset and the proportions used for training and testing.
- Naive Bayes Model: The project incorporates the Naive Bayes algorithm for disease prediction. The Gaussian Naive Bayes model is trained on the training set, and predictions are made on the test set. Various performance metrics, such as accuracy, precision, recall, and F1-score, are calculated and displayed. Additionally, a confusion matrix is generated for result visualization.
- Artificial Neural Network (ANN) Model: An Artificial Neural Network (ANN) is implemented for disease prediction. The script uses Keras to build a neural network with multiple layers. The model is trained on the training set, and the weights are saved for future use. If the model weights exist, they are loaded instead. The ANN model is evaluated on the test set, and performance metrics are displayed similarly to the Naive Bayes model.
- Performance Metrics and Comparison Graph: The script calculates and displays various performance metrics for both Naive Bayes and ANN models, including accuracy, precision, recall, and F1-score. A comparison graph is generated to visually represent the performance of the two algorithms.
- Prediction on Test Data: Finally, the project allows the user to predict diseases from test data. The user can upload new test data, and the ANN model predicts the disease outcomes for each entry. The results, including the test data and predicted classes, are displayed on the GUI.

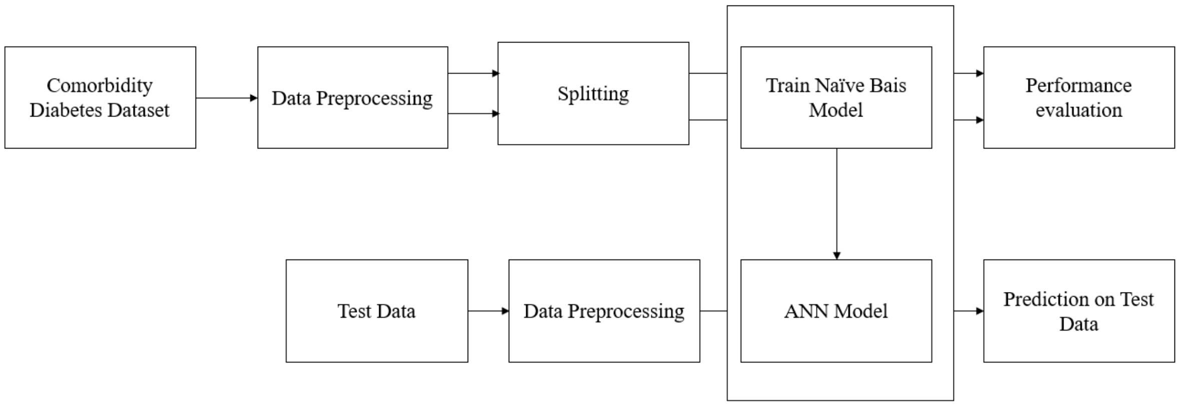
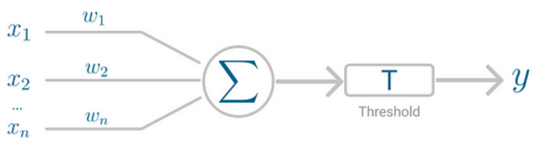


Figure 3.1: Block Diagram of Proposed Model.

3.2 ANN Classifier

Although today the Perceptron is widely recognized as an algorithm, it was initially intended as an image recognition machine. It gets its name from performing the human-like function of perception, seeing, and recognizing images. Interest has been centered on the idea of a machine which would be capable of conceptualizing inputs impinging directly from the physical environment of light, sound, temperature, etc. — the "phenomenal world" with which we are all familiar — rather than requiring the intervention of a human agent to digest and code the necessary information. Rosenblatt's perceptron machine relied on a basic unit of computation, the neuron. Just like in previous models, each neuron has a cell that receives a series of pairs of inputs and weights. The major difference in Rosenblatt's model is that inputs are combined in a weighted sum and, if the weighted sum exceeds a predefined threshold, the neuron fires and produces an output.



$$y = \begin{cases} 1, \text{if } \overbrace{\sum_i w_i x_i}^{\text{weighted sum}} - \underset{\text{threshold}}{T} > 0 \\ 0, \text{otherwise} \end{cases}$$

Fig. 3.2: Perceptron neuron model (left) and threshold logic (right).

Threshold T represents the activation function. If the weighted sum of the inputs is greater than zero the neuron outputs the value 1, otherwise the output value is zero.

Perceptron for Binary Classification

With this discrete output, controlled by the activation function, the perceptron can be used as a binary classification model, defining a linear decision boundary.

It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary. The perceptron loss function is defined as below:

$$\underbrace{D(w,c)}_{\text{distance}} = -\sum_{\underset{\text{misclassified observations}}{i \in M}} \overset{\text{output}}{y_i} (x_i w_i + c)$$

To minimize this distance, perceptron uses stochastic gradient descent (SGD) as the optimization function. If the data is linearly separable, it is guaranteed that SGD will converge in a finite number of steps. The last piece that Perceptron needs is the activation function, the function that determines if the neuron will fire or not. Initial Perceptron models used sigmoid function, and just by looking at its shape, it makes a lot of sense! The sigmoid function maps any real input to a value that is either 0 or 1 and encodes a non-linear function. The neuron can receive negative numbers as input, and it will still be able to produce an output that is either 0 or 1.

But, if you look at Deep Learning papers and algorithms from the last decade, you'll see the most of them use the Rectified Linear Unit (ReLU) as the neuron's activation function. The reason why ReLU became more adopted is that it allows better optimization using SGD, more efficient computation and is scale-invariant, meaning, its characteristics are not affected by the scale of the input. The neuron receives inputs and picks an initial set of weights random. These are combined in weighted sum and then ReLU, the activation function, determines the value of the output.

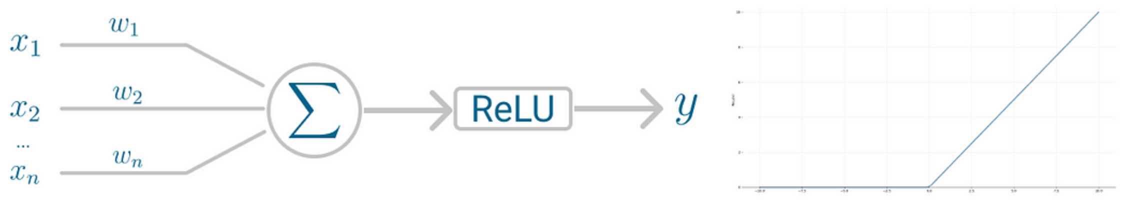


Fig. 3.3: Perceptron neuron model (left) and activation function (right).

Perceptron uses SGD to find, or you might say learn, the set of weight that minimizes the distance between the misclassified points and the decision boundary. Once SGD converges, the dataset is separated into two regions by a linear hyperplane. Although it was said the Perceptron could represent any circuit and logic, the biggest criticism was that it couldn't represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different. This was proved almost a decade later and highlights the fact that Perceptron, with only one neuron, can't be applied to non-linear data.

3.2.2 ANN

The ANN was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear. A ANN has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a ANN can use any arbitrary activation function. ANN falls under the category of feedforward algorithms because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer. If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning. This is where Backpropagation comes into play.

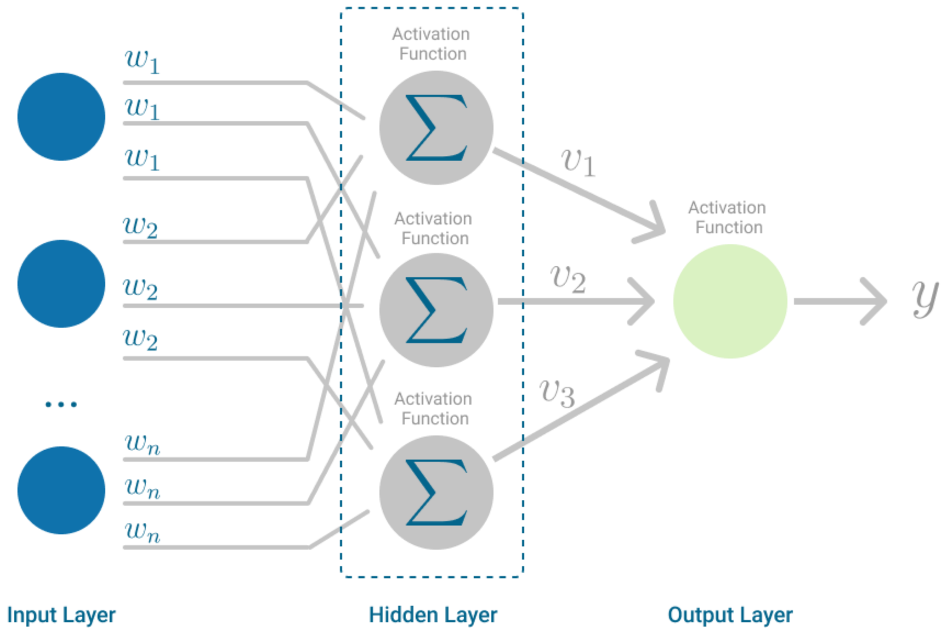


Fig. 3.4: Architecture of ANN.

Backpropagation: Backpropagation is the learning mechanism that allows the ANN to iteratively adjust the weights in the network, with the goal of minimizing the cost function. There is one hard requirement for backpropagation to work properly. The function that combines inputs and weights in a neuron, for instance the weighted sum, and the threshold function, for instance ReLU, must be differentiable. These functions must have a bounded derivative because Gradient Descent is typically the optimization function used in ANN. In each iteration, after the weighted sums are forwarded through all layers, the gradient of the Mean Squared Error is computed across all input and output pairs. Then, to propagate it back, the weights of the first hidden layer are updated with the value of the gradient. That's how the weights are propagated back to the starting point of the neural network. One iteration of Gradient Descent is defined as follows:

$$\underbrace{\Delta_w(t)}_{\text{Gradient Current Iteration}} = -\overset{\text{Bias}}{\varepsilon}\,\frac{\overset{\text{Error}}{dE}}{\underbrace{dw_{(t)}}_{\text{Weight vector}}} + \overset{\text{Learning Rate}}{\alpha}\underbrace{\Delta_{w(t-1)}}_{\text{Gradient Previous Iteration}}$$

This process keeps going until gradient for each input-output pair has converged, meaning the newly computed gradient hasn't changed more than a specified convergence threshold, compared to the previous iteration.

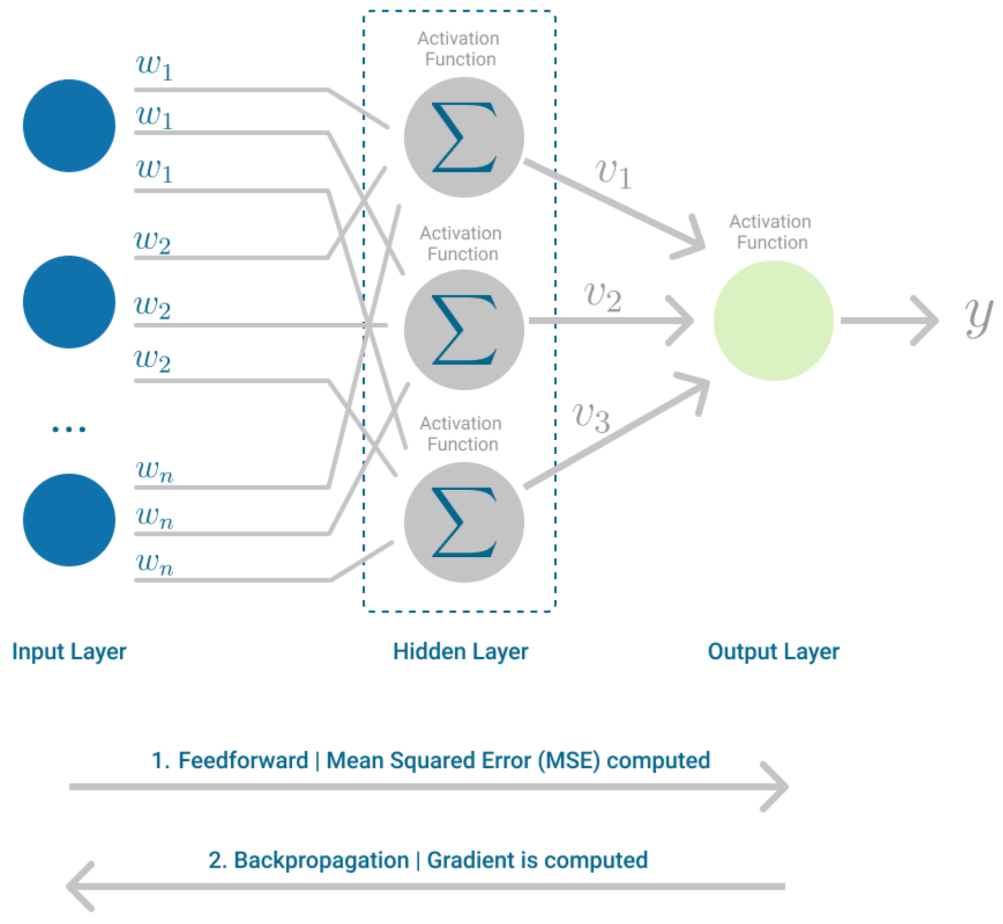


Fig. 3.5: ANN, highlighting the Feedforward and Backpropagation steps.

3.3 Advantages

Comprehensive Data Analysis: The research begins with a thorough exploration and analysis of the dataset, providing descriptive statistics and visualizations. This step aids in understanding the characteristics and distribution of the data.

Preprocessing for Model Readiness: Null value checks and preprocessing techniques, such as standard scaling, are applied to the dataset. This ensures that the data is suitable for training machine learning models by addressing missing values and normalizing feature scales.

Visualization of Decision Classes: The use of a count plot to visualize the distribution of decision classes enhances the understanding of the dataset's class balance or imbalance. This insight is crucial for selecting appropriate modeling techniques, particularly in the context of imbalanced datasets.

Logistic Regression Model: The inclusion of a logistic regression model provides a baseline for binary classification. Logistic regression is a simple yet effective algorithm for such tasks, making it suitable for comparison with more complex models.

Artificial Neural Network (ANN) Model: The research introduces an ANN model, a more sophisticated and flexible approach capable of capturing complex patterns in the data. ANNs are well-suited for tasks with non-linear relationships between features and outcomes.

Handling Imbalanced Data with SMOTE: The utilization of the Synthetic Minority Over-sampling Technique (SMOTE) during the training of the ANN model addresses potential class imbalance. SMOTE generates synthetic samples of the minority class, promoting a more balanced representation and potentially improving model performance.

Model Evaluation and Comparison:

The research evaluates and compares the performance of both the logistic regression and ANN models. This comparison allows for an assessment of whether the added complexity of the ANN model results in significant performance improvements.

Confusion Matrix and Classification Report: The inclusion of confusion matrices and classification reports provides a detailed breakdown of model performance, including metrics such as precision, recall, and F1-score. This information is crucial for understanding the model's strengths and weaknesses.

ROC Curve Analysis: The research incorporates Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) analysis. These metrics offer a visual representation of model performance, especially in terms of the trade-off between true positive rate and false positive rate.

Educational Value:

The research is educational and serves as a practical example of building, training, and evaluating machine learning models. It introduces users to fundamental concepts such as data preprocessing, model training, and performance evaluation.

Flexibility and Extensibility:

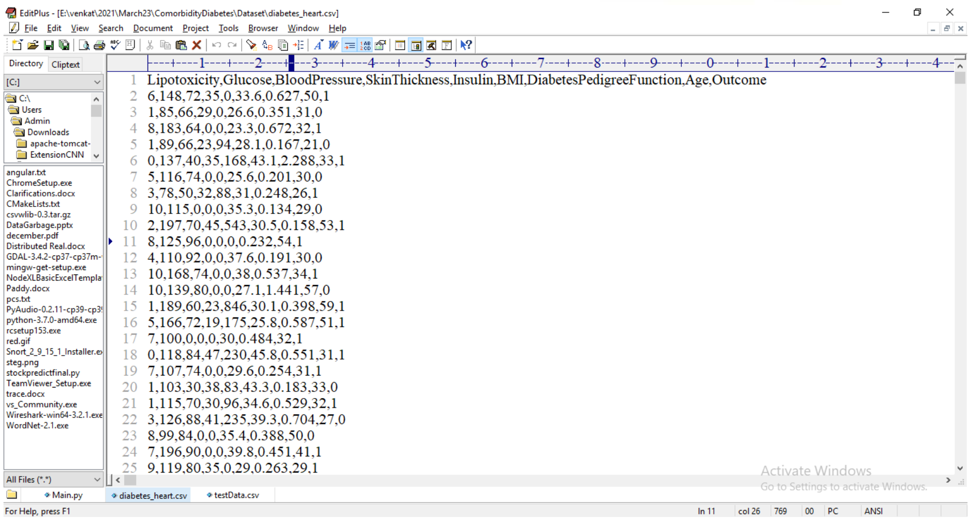
The modular structure of the code allows for flexibility and extensibility. Users can easily modify the code to experiment with different models, hyperparameters, or additional preprocessing steps to tailor the research to specific needs.

4. RESULTS

Figure 1: Presents a visual representation of the dataset. It includes various data points and their respective features. Each data point represented in a graphical format, such as points on a scatter plot or lines on a line graph, depending on the nature of the data.

Figure 2: Displays the graphical user interface (GUI) designed for the Comorbidities of Diabetes system. The interface includes different sections or tabs for performing various tasks related to analyzing diabetic comorbidities, such as data uploading, model selection, and result visualization.

Figure 3: Showcases the process of selecting a dataset to upload within the GUI interface. It has options for browsing files or directories to locate the desired dataset file. Once selected, the dataset is loaded into memory for further analysis.



```
Lipotoxicity,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
2,197,70,45,543,30.5,0.158,53,1
8,125,96,0,0,0,0.232,54,1
4,110,92,0,0,37.6,0.191,30,0
10,168,74,0,0,38,0.537,34,1
10,139,80,0,0,27.1,1.441,57,0
1,189,60,23,846,30.1,0.398,59,1
5,166,72,19,175,25.8,0.587,51,1
7,100,0,0,0,30,0.484,32,1
0,118,84,47,230,45.8,0.551,31,1
7,107,74,0,0,29.6,0.254,31,1
1,103,30,38,83,43.3,0.183,33,0
1,115,70,30,96,34.6,0.529,32,1
3,126,88,41,235,39.3,0.704,27,0
8,99,84,0,0,35.4,0.388,50,0
7,196,90,0,0,39.8,0.451,41,1
9,119,80,35,0,29,0.263,29,1
```

Figure 1: Displays the dataset.

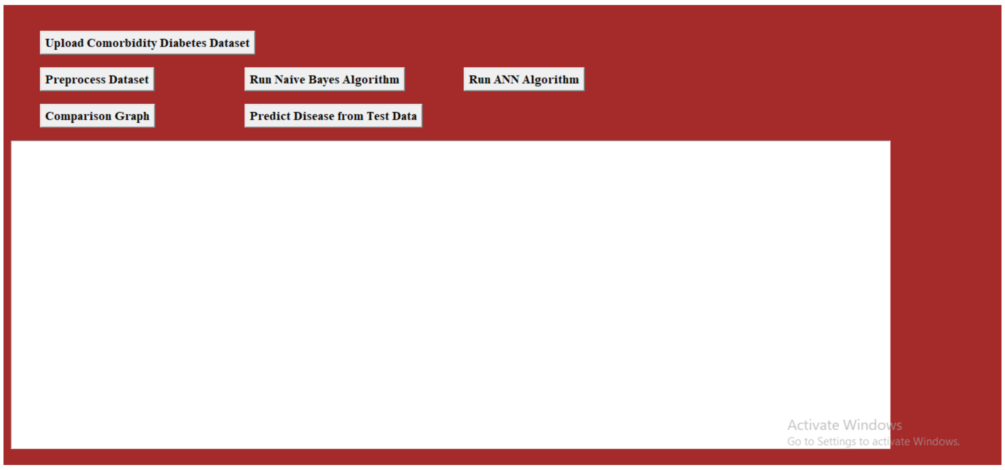

Upload Comorbidity Diabetes Dataset
Preprocess Dataset
Run Naive Bayes Algorithm
Run ANN Algorithm
Comparison Graph
Predict Disease from Test Data

Figure 2: Displays the GUI interface of Comorbidities of Diabetes.

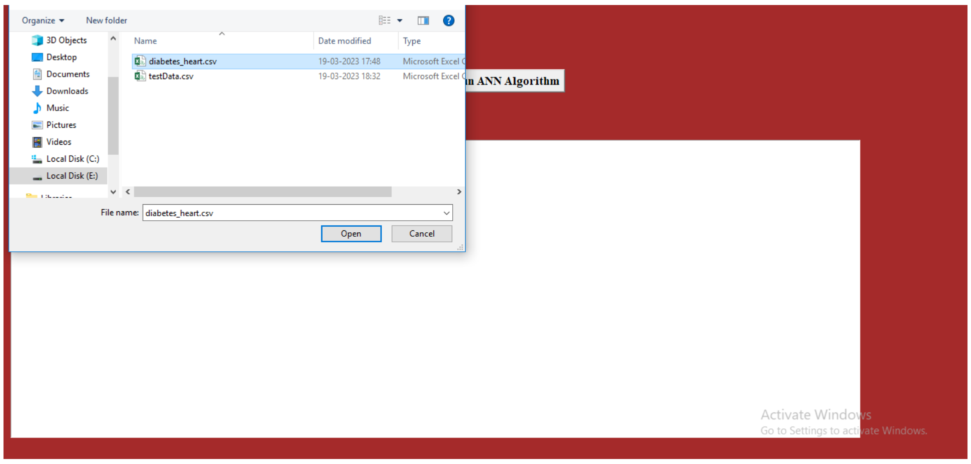


Figure 3: Displays the selection of dataset to upload in the GUI.

Figure 4: Illustrates a count plot representing the distribution of classes within the dataset. Each class, possibly related to different types of diabetic comorbidities, is depicted along the x-axis, while the frequency or count of occurrences of each class is displayed on the y-axis.

Figure 5: Demonstrates the preprocessing and normalization steps applied to the dataset before model training. Preprocessing techniques has handling missing values, encoding categorical variables, and removing outliers, while normalization ensures that all features are on a similar scale to aid in model convergence.

Figure 6: Presents the confusion matrix generated from the predictions of the Naïve Bayes model. A confusion matrix is a table used to evaluate the performance of a classification model, showing the number of true positives, true negatives, false positives, and false negatives.

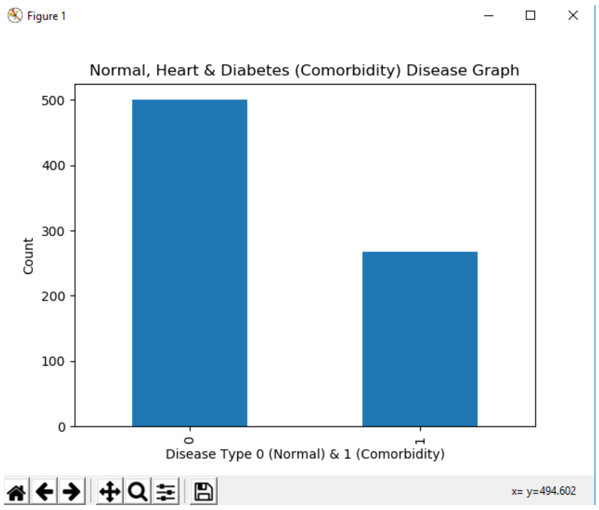
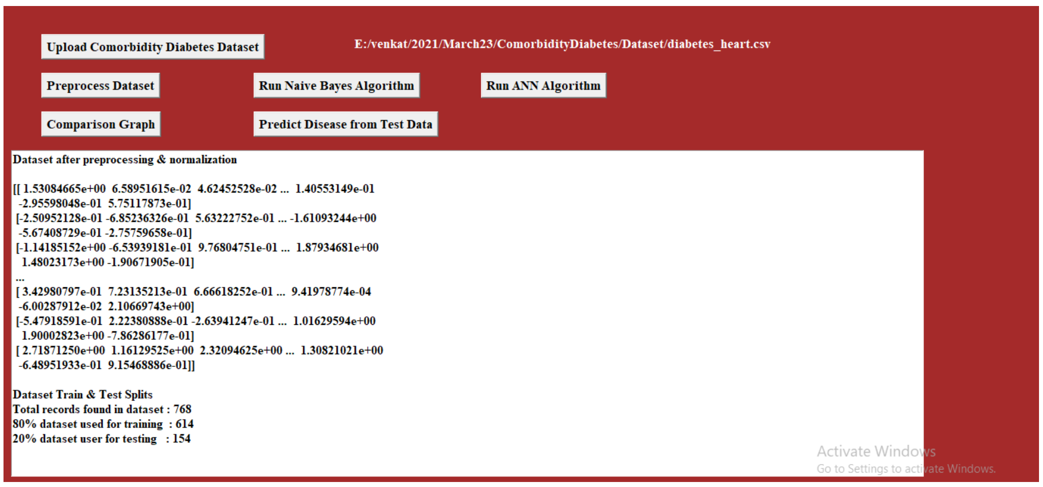


Figure 4: Shows the count plot of each class in the dataset.


Upload Comorbidity Diabetes Dataset
E:/venkat/2021/March23/ComorbidityDiabetes/Dataset/diabetes_heart.csv
Preprocess Dataset
Run Naive Bayes Algorithm
Run ANN Algorithm
Comparison Graph
Predict Disease from Test Data

```
Dataset after preprocessing & normalization

[[ 1.53084665e+00  6.58951615e-02  4.62452528e-02 ...  1.40553149e-01
  -2.95598048e-01  5.75117873e-01]
 [-2.50952128e-01 -6.85236326e-01  5.63222752e-01 ... -1.61093244e+00
  -5.67408729e-01 -2.75759658e-01]
 [-1.14185152e+00 -6.53939181e-01  9.76804751e-01 ...  1.87934681e+00
   1.48023173e+00 -1.90671905e-01]
 ...
 [ 3.42980797e-01  7.23135213e-01  6.66618252e-01 ...  9.41978774e-04
  -6.00287912e-02  2.10669743e+00]
 [-5.47918591e-01  2.22380888e-01 -2.63941247e-01 ...  1.01629594e+00
   1.90002823e+00 -7.86286177e-01]
 [ 2.71871250e+00  1.16129525e+00  2.32094625e+00 ...  1.30821021e+00
  -6.48951933e-01  9.15468886e-01]]

Dataset Train & Test Splits
Total records found in dataset : 768
80% dataset used for training  : 614
20% dataset user for testing   : 154
```

Figure 5: Displays the dataset preprocessing and normalization.

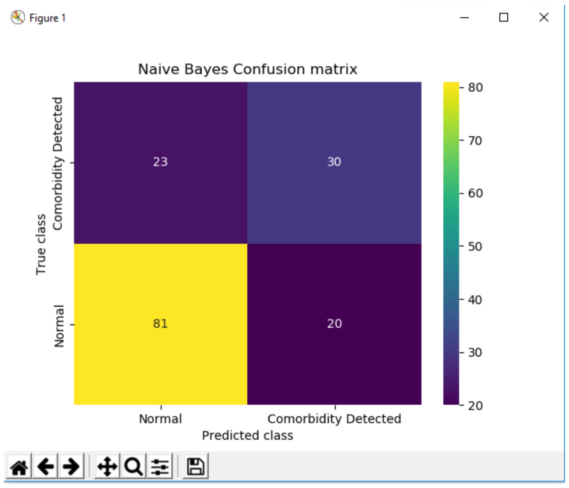


Figure 6: Presents the confusion matrix of Naïve Bayes Model.

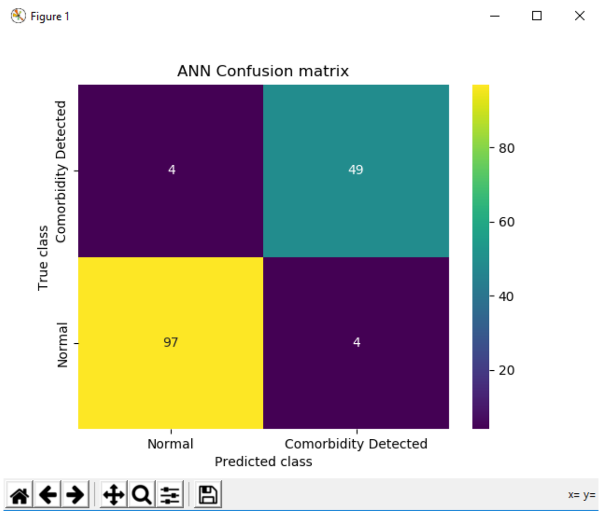


Figure 7: Displays the Confusion matrix of ANN model.

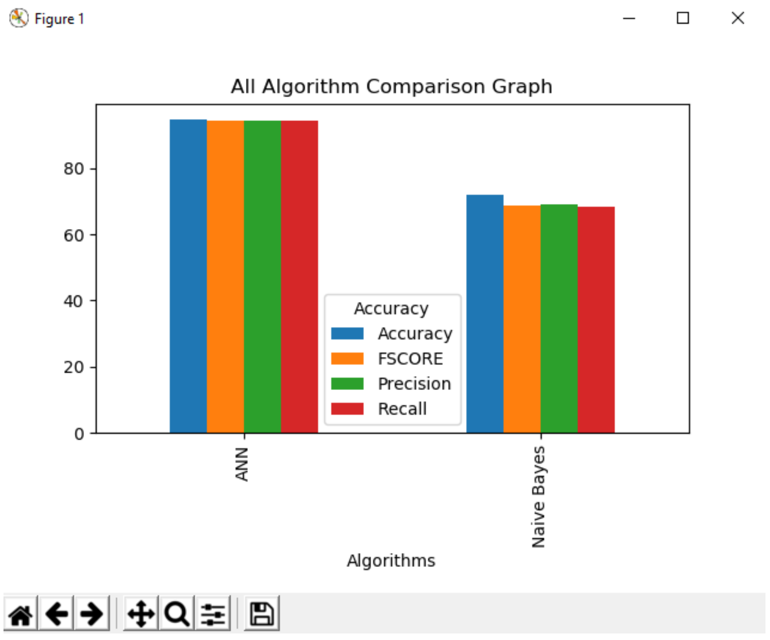


Figure 8: Represents the performance comparison graph of each model.

Figure 7: Displays the confusion matrix specifically for the Artificial Neural Network (ANN) model. The confusion matrix helps assess the ANN model's performance in classifying diabetic comorbidities based on its predictions compared to the actual labels.

Figure 8: Provides a graphical comparison of the performance metrics of different models. It has metrics such as accuracy, precision, recall, or F1-score for each model, allowing for easy comparison of their effectiveness in predicting diabetic comorbidities.

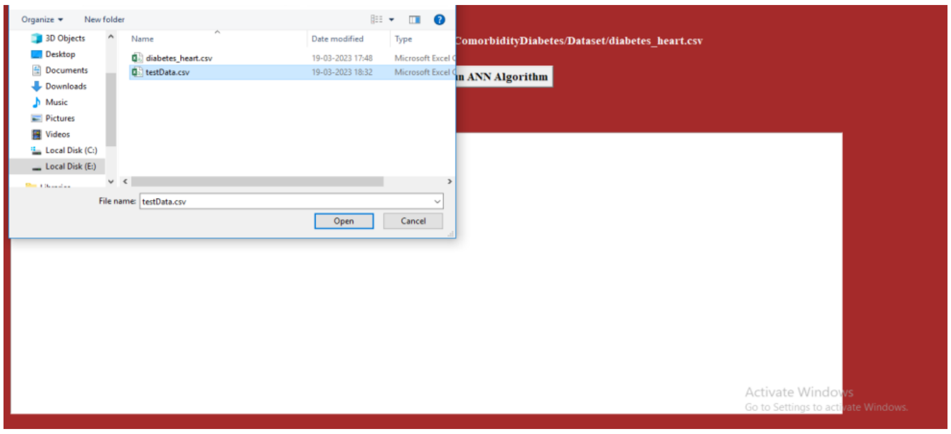
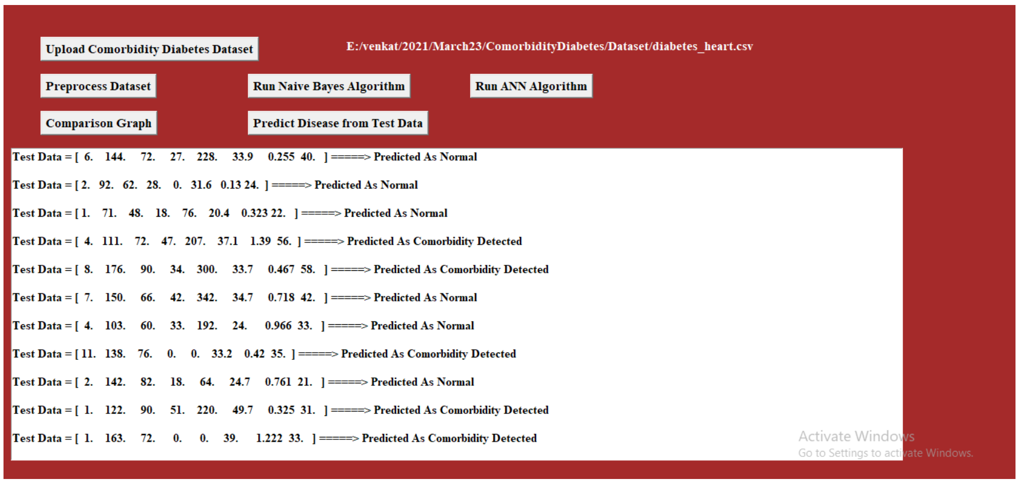


Figure 9: Displays the uploading of test data for model prediction.

Figure 9: Depicts the process of uploading test data for model prediction within the GUI interface. We can select a separate dataset containing test instances to evaluate the trained models' performance.

Figure 10: Shows the model's predictions on the test data uploaded in Figure 9. Has visualizations of the predicted outcomes compared to the actual labels, allowing for a qualitative assessment of the model's performance on unseen data.


Upload Comorbidity Diabetes Dataset
E:/venkat/2021/March23/ComorbidityDiabetes/Dataset/diabetes_heart.csv
Preprocess Dataset
Run Naive Bayes Algorithm
Run ANN Algorithm
Comparison Graph
Predict Disease from Test Data
Activate Windows
Go to Settings to activate Windows.

```
Test Data = [ 6.   144.   72.   27.   228.   33.9   0.255  40.  ] =====> Predicted As Normal
Test Data = [ 2.   92.   62.   28.   0.   31.6   0.13 24.  ] =====> Predicted As Normal
Test Data = [ 1.   71.   48.   18.   76.   20.4   0.323 22.  ] =====> Predicted As Normal
Test Data = [ 4.   111.   72.   47.   207.   37.1   1.39  56.  ] =====> Predicted As Comorbidity Detected
Test Data = [ 8.   176.   90.   34.   300.   33.7   0.467  58.  ] =====> Predicted As Comorbidity Detected
Test Data = [ 7.   150.   66.   42.   342.   34.7   0.718  42.  ] =====> Predicted As Normal
Test Data = [ 4.   103.   60.   33.   192.   24.   0.966  33.  ] =====> Predicted As Normal
Test Data = [ 11.   138.   76.   0.   0.   33.2   0.42  35.  ] =====> Predicted As Comorbidity Detected
Test Data = [ 2.   142.   82.   18.   64.   24.7   0.761  21.  ] =====> Predicted As Normal
Test Data = [ 1.   122.   90.   51.   220.   49.7   0.325  31.  ] =====> Predicted As Comorbidity Detected
Test Data = [ 1.   163.   72.   0.   0.   39.   1.222  33.  ] =====> Predicted As Comorbidity Detected
```

Figure 10: Displays the model prediction on test data.

Table 1: Performance comparison of quality metrics obtained using Naïve bayes

And ANN model.

Model	Naïve bayes	ANN model
Accuracy (%)	72.0	94.8
Precision (%)	68.9	94.2
Recall (%)	68.4	89
F1-score (%)	68.6	89

For the Naïve bayes model:

- The Accuracy is 72.0, indicating the accuracy between the actual and predicted values.
- The Precision is 68.9, suggesting that, on average Precision between the actual and predicted values.
- The Recall is 68.4, suggesting that, on average Recall between the actual and predicted values.
- The F1-score is 68.6, representing the average F1-score between the actual and predicted values.

For the ANN model:

- The Accuracy is 94.8, indicating the accuracy between the actual and predicted values.
- The Precision is 94.2, suggesting that, on average Precision between the actual and predicted values.
- The Recall is 94.2, suggesting that, on average Recall between the actual and predicted values.
- The F1-score is 94.2, representing the average F1-score between the actual and predicted values.

5. CONCLUSION

In conclusion, the integration of machine learning in the detection and prediction of comorbidities in individuals with diabetes marks a significant advancement in healthcare. While traditional approaches relied on manual analysis and clinical observations, machine learning models offer a more sophisticated and data-driven methodology. These models, trained on diverse datasets encompassing patient demographics, medical history, and laboratory results, can uncover intricate patterns and relationships that may elude human observation. The imperative for accurate and timely detection of comorbidities in diabetes is underscored by the potential to enhance treatment strategies and improve patient outcomes. Machine learning techniques, by analyzing large and complex datasets, provide a pathway to more personalized healthcare interventions, ensuring early and effective management of comorbid conditions.

REFERENCE

1) Deo, R.C. Machine Learning in Medicine. Circulation 2015, 132, 1920–1930.

2) Yuvaraj, N.; SriPreethaa, K.R. Diabetes prediction in healthcare systems using machine learning algorithms on Hadoop cluster. Clust. Comput. 2017, 22, 1–9.
3) Tafa, Z.; Pervetica, N.; Karahoda, B. An intelligent system for diabetes prediction. In Proceedings of the 2015 4th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 14–18 June 2015; pp. 378–382
4) Sisodia, D.; Sisodia, D.S. Prediction of Diabetes using Classification Algorithms. Procedia Comput. Sci. 2018, 132, 1578–1585
5) Mercaldo, F.; Nardone, V.; Santone, A. Diabetes Mellitus Affected Patients Classification and Diagnosis through Machine Learning Techniques. Procedia Comput. Sci. 2017, 112, 2519–2528
6) Negi, A.; Jaiswal, V. A first attempt to develop a diabetes prediction method based on different global datasets. In Proceedings of the 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 22–24 December 2016; pp. 237–241
7) Olaniyi, E.O.; Adnan, K. Onset diabetes diagnosis using artificial neural network. Int. J. Sci. Eng. Res. 2014, 5, 754–759.
8) Soltani, Z.; Jafarian, A. A New Artificial Neural Networks Approach for Diagnosing Diabetes Disease Type II. Int. J. Adv. Comput. Sci. Appl. 2016, 7, 89–94