Chronic Disease Prediction through Supervised Learning Techniques

Kasarapu Ramani[1], Irala Suneetha[2], Nainaru Pushpalatha[3], P. Harish[4], P. Yugandhar[5]

[1]Professor, Department of Information Technology, Sree Vidyanikethan Engineering College, Mohan Babu University, Tirupati, India.
E-mail: ramanidileep@yahoo.com

[2]Professor, Department of Electronics and Communication Engineering, Annamacharya Institute of Technology and Science, Tirupati, India.
E-mail: iralasuneetha.aits@gmail.com

[3]Assoc Professor, Department of Electronics and Communication Engineering, Annamacharya Institute of Technology and Science, Tirupati, India.
E-mail: pushpalatha825@gmail.com

[4]Associate Professor, Department of Electronics and Communication Engineering, Annamacharya Institute of Technology and Science, Tirupati, India.
E-mail: harishpasupulati@gmail.com

[5]Assistant Manager, Ernst & Young Pvt. Ltd, Bangalore. E-mail: yugandhar84@gmail.com

Abstract

About 10.5 percent of global adult population is living with diabetes. India has 77 million diabetic patients, it is the second highest in the world. Developing countries such as India face a huge burden of diabetes and its complications. Even children at the age of five are suffering from this disease. It is high time that people understand the gravity of the situation and make themselves fit to fight the disease than to suffer with it. If diabetes is not identified and treated in right time, it may lead to chronical health issues. In this paper a machine learning based prediction model is built to find the factors leading to complicated health issues such as cardio vascular disease. This model identifies the attributes that highly contribute to cardio vascular disease and compare various machine learning algorithms to predict cardio vascular disease among diabetic patients. It identifies the best algorithm from a set of supervised learning algorithms such as KNN, Decision Tree, Random Forest, Naïve Bayes and Gradient Boosting for prediction based on several performance metrics. The algorithms are compared based on the performance metrics such as accuracy, precision, recall, F1 score, time taken to train and time taken to test. We identified that Decision Tree with entropy as the split criterion achieved the highest accuracy.

Keywords: K-Nearest Neighbours, Decision Tree with Gini Index, Decision Tree with Entropy, Random Forest, Naïve Bayes and Gradient Boosting.

DOI: 10.47750/pnr.2022.13.04.134

INTRODUCTION

As per [1] about 530 millions of people in the age group of 20-79 are living with diabetes and it is projected to rise around 650 million by 2030. Two-Thirds of Urban people are living with diabetes [2]. Diabetes affects different organs of human body if is not diagnosed in early stage and not treated with proper medication, then it may lead to several life threatening diseases such as heart stroke, kidney damage and nerve damage etc.,. Therefore, an awareness on the factors leading to the occurrence of diabetes should be provided to the public. It is important to identify the chronic diseases that a patient may suffer after being diagnosed with diabetes. This work focuses on Machine learning models that identify the key attributes that highly effect the diabetic patients in acquiring chronic diseases. This paper aims at developing machine learning models that help in predicting chronic diseases due to diabetes. It aims at predicting the heart disease of a diabetic patient that highly affect the person to suffer from heart disease after being diagnosed of diabetes.

RELATED WORK

In[3], the authors used supervised learning techniques such as Random Forest (RF), Decision Tree (DT), and Logistic Regression (LR) to predict cardiac illness on UCI Cleveland database of heart disease patients. The findings show that logistic regression achieves the best accuracy score. In this only three algorithm's performance is compared and also used for Cardiovascular Disease but not on exclusively on diabetic patients. In[4] the ML algorithms like ANN, SVM, LASSO, and Ensemble learning are used for diagnosis of diabetic and Heart disease. But the dataset used is binary class labeled and does not speak of risk of cardio vascular disease among diabetic people. In[6] the authors used Naive Bayes (NB) Machine Learning technique for predicting diabetes. The Bayesian algorithm considers the attributes are independent of each other, but many correlated attributes may result in diabetes. Hence, this technique may not be the

most appropriate method for diabetes prediction. In[7] the authors applied KNN algorithm for diabetes prediction, but algorithm cost increases with number of attributes also if equal weights are assigned to attributes, then it will result in poor classification. Another difficulty of this method is it won't decide the appropriate feature which will result in successful classification. In[7] the authors used SVM technique for prediction of diabetes. But, for noisy data this algorithm gives poor results. Also, interpretation of appropriate feature for successful classification is not possible. Its computational cost increases with size of the dataset.

From the literature it is observed that most of the research work focused on predicting diabetes if dataset is given. But, the research work in this paper focuses on the prediction of cardiovascular in patients affected with diabetes using Machine Learning Techniques.

METHODOLOGY

In order to find the highly contributing attributes for cardio vascular disease among diabetic patients and to find the best algorithm that predicts cardio vascular disease among diabetic patients the following procedure is followed:

1. To train the model, a dataset is collected from UCI and 14 key attributes are considered for cardiovascular disease prediction.
2. Pre-processing is applied to eliminate missing values from the dataset.
3. Feature extraction is performed on the data by using ExtraTreeClassifier and the feature with highest relevance is obtained and a graph is drawn between the score and the attributes.
4. The dataset is split into training set and test set with 70% and 30% proportions respectively.
5. Classification is performed using K-NN with k=13, Decision Tree with gini index, Decision Tree with entropy, Random Forest, Naïve Bayes and Gradient Boosting.
6. The trained model is tested on the test set to predict for various algorithms mentioned above and their performance is evaluated.
7. The performance metrics used include accuracy, precision, recall, F1 score, train time, test time and confusion matrix.
8. The performance metrics of all the algorithms are obtained in the form of a table for comparison.
9. Graphs are obtained for performance metrics vs algorithms.

EXPERIMENTAL RESULTS

- **Environment:** 11th Gen Intel(R) Core(TM) i5-1135G7 processor is used with 8GB RAM and 64-bit operating system. The software used is python of version 3.x with sklearn.
- UCI dataset with 14 attributes is used. The "label" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. The data is in the form of. data file.
- One of the attributes is fasting blood sugar which is the criterion for filtering diabetic data.
- The remaining 13 attributes include age, sex, chest pain type, resting blood pressure, serum cholesterol, resting electrocardiographic results, max heart rate achieved, exercise induced angina, ST depression induced by exercise relative to rest, slope of peak exercise ST segment, number of major vessels colored by fluoroscopy, thalassemia and label.
- After filtering the null values from the dataset, we obtained 43 diabetic people we performed feature selection using ExtraTreeClassifier and obtained the following result:

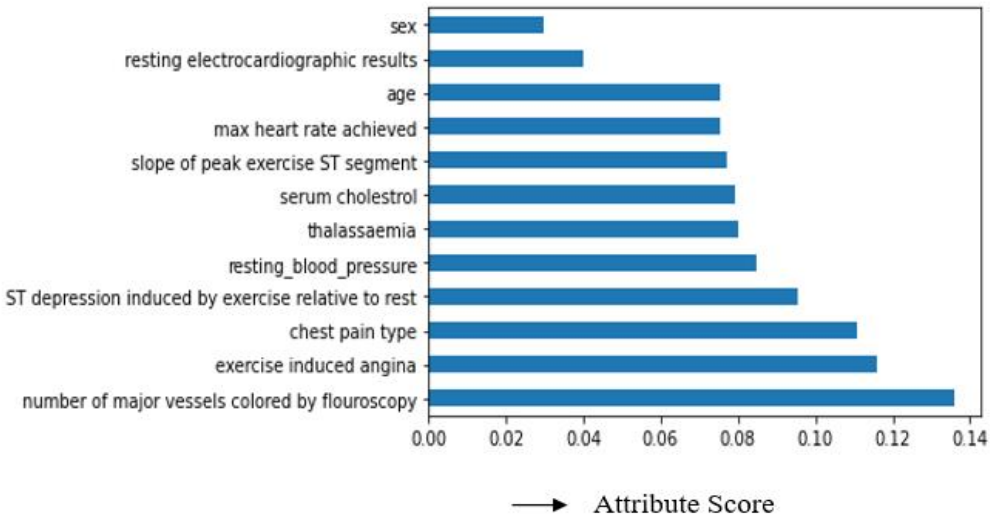

Fig. 1. The Impact of attribute on Heart disease prediction

- Various classification algorithms are used to train the model and test it [14].

Algorithm 1: KNN

The steps followed while implementing KNN is as follows:

1) Select the number K of the neighbours
2) Step-2: Calculate the Euclidean distance of K number of neighbours
3) Step-3: Take the K nearest neighbours as per the calculated Euclidean distance.

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Where, p,q = two points in Euclidean n-space, q_i,p_i = Euclidean vectors, starting from the origin of the space (initial point), and n = n-space

4) Among these k neighbours, count the number of the data points in each category.
5) Assign the new data points to that category for which the number of the neighbour is maximum.

Decision Tree with Gini Index

We perform the following while implementing the KNN using KNeighborsClassifier module from sklearn library

1) Import KNeighborsClassifier from sklearn library
2) Split the dataset into 70% for training and 30% for testing
3) For k from 1 to 25, fit the model and compare the model score to identify the k that gives the highest score.
4) By using the obtained k, test the model and apply performance metrics for further analysis.
5) Draw the confusion matrix to analyse the results further. For Fig.2 we can observe that 7 values are falsely predicted and the other values are correctly predicted.

Algorithm 2: Decision Tree with Gini Index

1) Find each feature's best split. For each feature with K different values there exist K-1 possible splits. Find the split, which maximizes the splitting criterion. The resulting set of splits contains best splits (one for each feature). Split the dataset into 70% for training and 30% for testing
2) Find the node's best split. Among the best splits from Step i find the one, which maximizes the splitting criterion.
3) Split the node using best node split from Step ii and repeat from Step i until stopping criterion is satisfied.

As splitting criterion we used Gini's impurity index, which is defined for node t as:

$$i(t) = \sum_{i,j} C(i|j)p(i|t)p(j|t) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.....(2)$$

Where, C(i|j) is cost of misclassifying a class j case as a class i case (in our case C(i|j) = 1, if i 6= j and C(i|j) = 0 if i = j), p(i|t) (p(j|t) respectively) is probability of case in class i (j) given that falls into node t.

The Gini impurity criterion is type of decrease of impurity, which is defined as:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(3)$$

Where, Δi(s, t) is decrease of impurity at node t with split s,

p_L,p_R are probabilities of sending case to the left (right) child node t_L (t_R) and

i(t_L),i(t_R) is Gini impurity measure for left (right) child node.

In decision tree algorithm, finding the attribute at each root node is a major challenge known as attribute selection. It gives the probability of classifying items incorrectly when attributes are selected randomly. Gini index or Gini Impurity is an optimum method for performing split at root node and subsequent splits [11]. If all items belong to a single class, then it is called pure and ranges between 0 and 1. 0 indicates that all items belong to one class, 0.5 indicates that items are equally distributed and 1 indicates items are randomly distributed. [12].

Classification And Regression Trees (CART) algorithm is a classification algorithm for building a decision tree based on Gini's impurity index as splitting criterion. CART is a binary tree build by splitting node into two child nodes repeatedly [13].

In order to enhance generalization of decision tree we used pruning with combination of cross-validation error rate estimation.

The algorithm for pruning works as follows:

1. Split randomly training data into 10 folds.
2. Select pruning level of tree (level 0 equals to full decision tree).
3. Use 9 folds for creation of 9 new pruned trees and estimate error on last 10th fold.
4. Repeat from Step ii until all pruning levels are used.
5. Find the smallest error and use the pruning level assigned to it.
6. Until pruning level is reached, remove all terminal nodes in the lowest tree level and assign decision class to parent node. Decision value is equal to class with higher number of cases covered by node.

This algorithm is used inherently by sklearn and we use this module to implement the decision tree classifier as follows:

1) Import DecisionTreeClassifier from sklearn library
2) Split the dataset into 70% for training and 30% for testing
3) The gini index criterion is by default available for DecisionTreeClassifier
4) Train the model with 70% of the data and calculate the time taken to train.
5) Test the model with 30% of the data and calculate the time taken to test.
6) Apply performance metrics for further analysis.
7) Use the trained model to obtain the decision tree with Gini index as the split criteria and split feature is chest pain type.

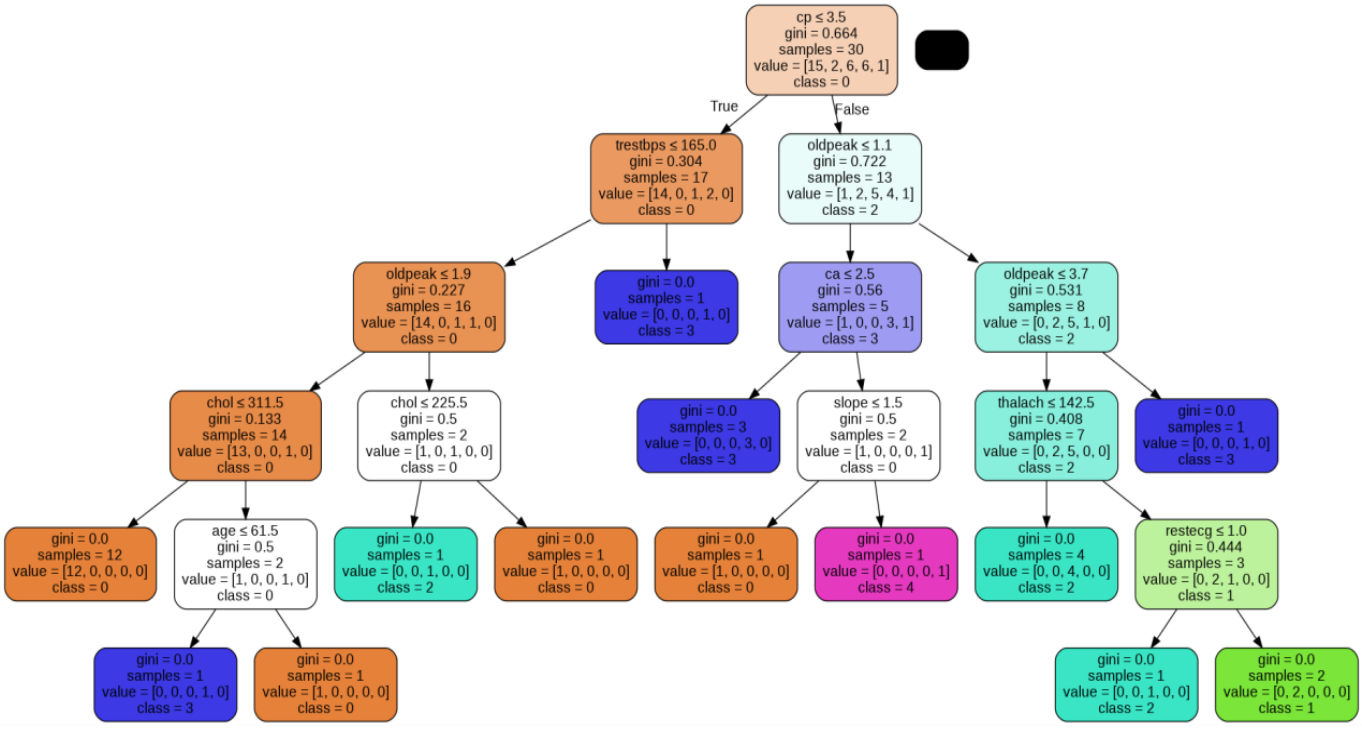

Fig. 2: Decision Tree with Gini Index as the splitting criterion

Algorithm 3: Decision Tree with Entropy

The algorithm works repeatedly in three steps:

1. Find each feature's best split. For each feature with K different values there exist K-1 possible splits. Find the split, which maximizes the splitting criterion. The resulting set of splits contains best splits (one for each feature).
2. Find the node's best split. Among the best splits from Step i find the one, which maximizes the splitting criterion.

Split the node using best node split from Step ii and repeat from Step i until stopping criterion is satisfied.

As splitting criterion we used Entropy, which is defined for node t as:

$$Entropy = -\sum_{j} p_j \log_2 p_j \qquad \text{.....(4)}$$

Where p_j is the probability of class j.

In order to enhance generalization of decision tree we used pruning with combination of cross-validation error rate estimation.

The algorithm for pruning works as follows:

1. Split randomly training data into 10 folds.
2. Select pruning level of tree (level 0 equals to full decision tree).
3. Use 9 folds for creation of 9 new pruned trees and estimate error on last 10th fold.
4. Repeat from Step ii until all pruning levels are used.
5. Find the smallest error and use the pruning level assigned to it.
6. Until pruning level is reached, remove all terminal nodes in the lowest tree level and assign decision class to parent node. Decision value is equal to class with higher number of cases covered by node.

This algorithm is used inherently by sklearn and we use this module to implement the decision tree classifier as follows:

1) Import DecisionTreeClassifier from sklearn library
2) Split the dataset into 70% for training and 30% for testing
3) Make the criterion as entropy in DecisionTreeClassifier.
4) Train the model with 70% of the data and calculate the time taken to train.
5) Test the model with 30% of the data and calculate the time taken to test.
6) Apply performance metrics for further analysis.
7) Use the trained model to obtain the decision tree with Gini index as the split criteria and split feature obtained is chest pain type.

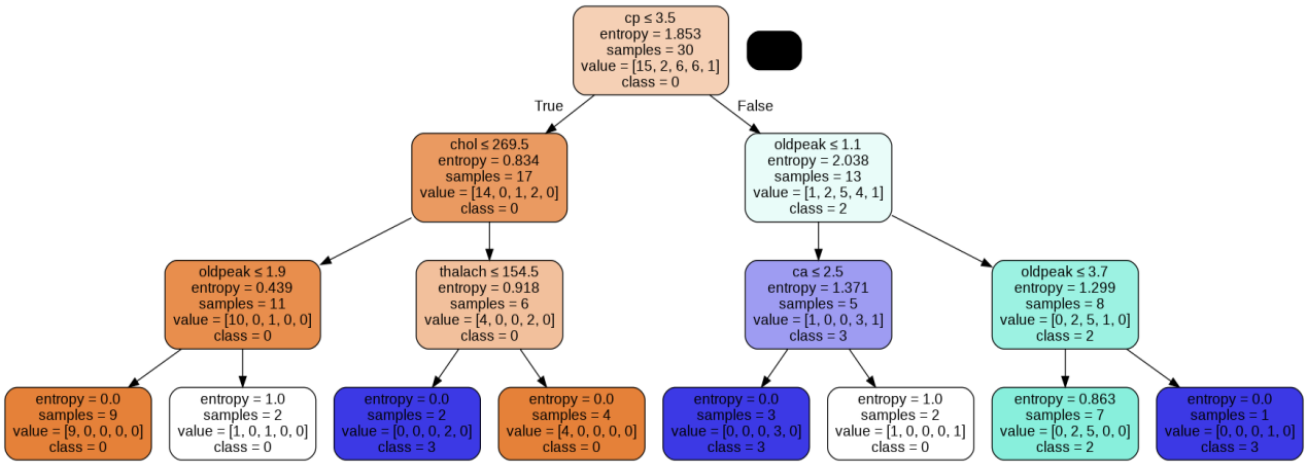

Fig. 3: Decision Tree with Entropy as the splitting criterion

Algorithm 4: Random Forest

The following are the basic steps involved in performing the random forest algorithm:

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

This algorithm in internally used by the RandomForestClassifier module of sklearn library. We use this module to train the Random Forest as show below:

1) Import RandomForestClassifier from sklearn library
2) Split the dataset into 70% for training and 30% for testing
3) Initialize model with a max depth of 3.
4) Train the model with 70% of the data and calculate the time taken to train.
5) Test the model with 30% of the data and calculate the time taken to test.
6) Apply performance metrics for further analysis.

Algorithm 5: Naïve Bayes

Bayes' 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:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)} \quad \text{……(5)}$$

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.

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.

We implement the same using sklearn library. The steps are as given below:

1) Import GaussianNB from sklearn library
2) Split the dataset into 70% for training and 30% for testing
3) Train the model with 70% of the data and calculate the time taken to train.
4) Test the model with 30% of the data and calculate the time taken to test.
5) Apply performance metrics for further analysis.

Algorithm 6: Gradient Boosting

Gradient boosting requires a differential loss function and works for both regression and classification.

The steps followed in Gradient Boosting

1. Fit estimator F^1
2. For i in [1,M]/M weak estimators

2.1. Calculate the loss in ith iteration

$$Loss^i = \sum_{j=1}^{n}\left(Y_j - F^i(X_j)\right)^2 \quad \text{……(6)}$$

2.2. Calculate negative gradient

$$-\frac{\partial L^i}{\partial X_j} = -\frac{2}{n} * \left(Y_j - F^i(X_j)\right) \forall i \quad \text{……(7)}$$

2.3. Fit a weak estimator Hi on $\left(X, \frac{\partial L}{\partial X}\right)$

3. Predict using the following formula

$$F^m(X) = F^i(X) + \rho * H^i(X) = F^1 + \rho * \sum_{i=1}^{m} H^i(X) \quad \text{……..(8)}$$

The below GradientBoostingClassifier from sklearn library is used to do the same.

1) Import GradientBoostingClassifier from sklearn library

2) Split the dataset into 70% for training and 30% for testing
3) Train the model with 70% of the data and calculate the time taken to train.
4) Test the model with 30% of the data and calculate the time taken to test.
5) Apply performance metrics for further analysis.

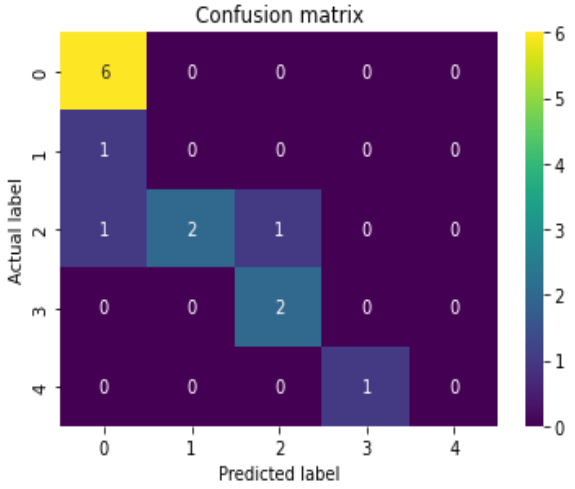

Fig 4. Confusion matrix of KNN with k = 13

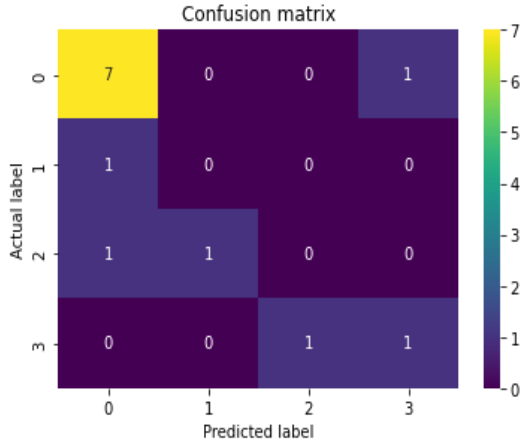

Fig 5.: Confusion matrix for decision tree classifier with Gini index

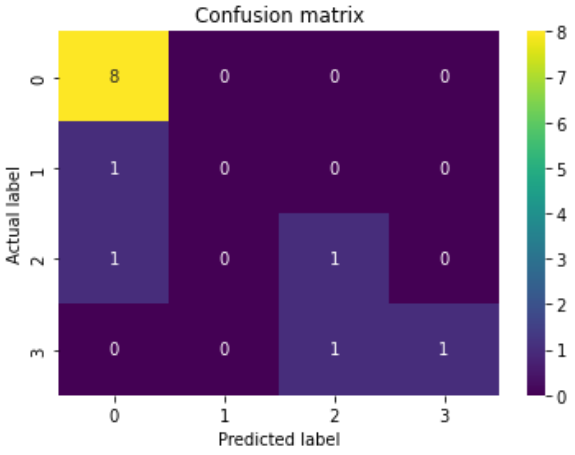

Fig.6 Confusion matrix for decision tree with Entropy

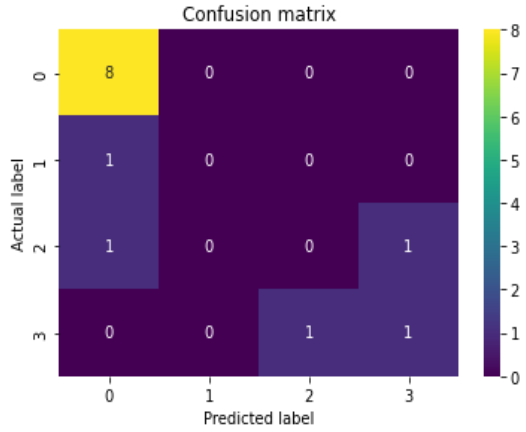

Fig 7: Confusion matrix for Random Forest

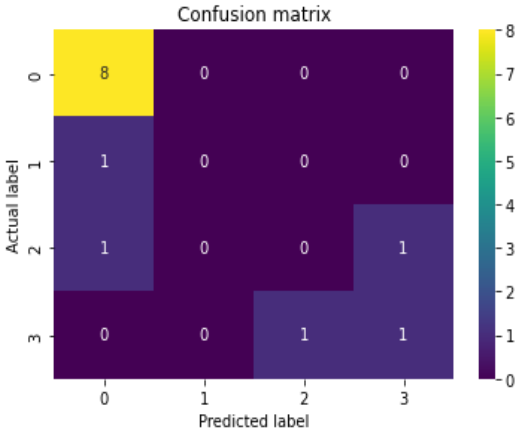

Fig 8. Confusion matrix for Naïve Bayes

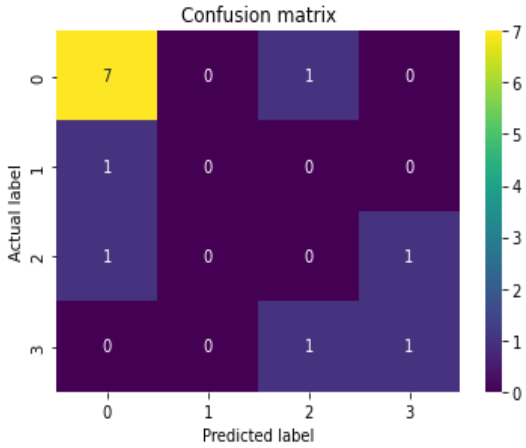

Fig 9. Confusion matrix for Gradient Boosting

Table 1. Comparison of different Machine Learning Algorithms in disease prediction

Name of the algorithm	Accuracy	Precision	Recall	F1-Score	Prediction Time(msec)
KNN	69%	0.56	0.69	0.59	44
Decision Tree with Gini Index	61%	0.55	0.61	0.58	14.8
Decision Tree with Entropy	76.9%	0.72	0.769	0.72	21
Random Forest	69%	0.57	0.69	0.62	11
Naïve Bayes	69%	0.60	0.69	0.64	27.6
Gradient Boosting	61%	0.55	0.61	0.58	24.5

Graphs for performance metrics vs algorithms are shown below:

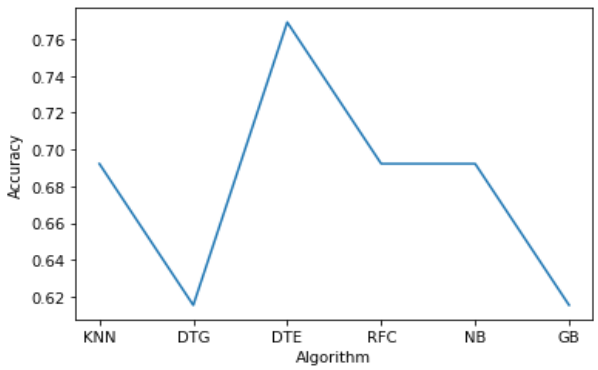

Fig. 10 A graph between Accuracy vs Algorithm

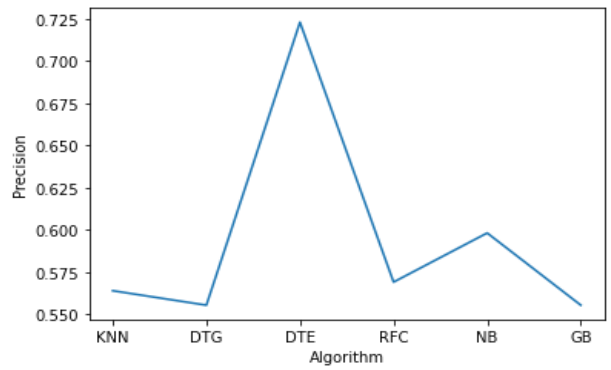

Fig.11 A graph between Precision vs Algorithm

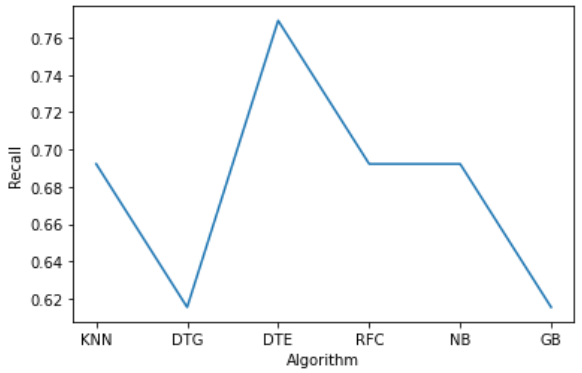

Fig. 12 A graph between Recall vs Algorithm

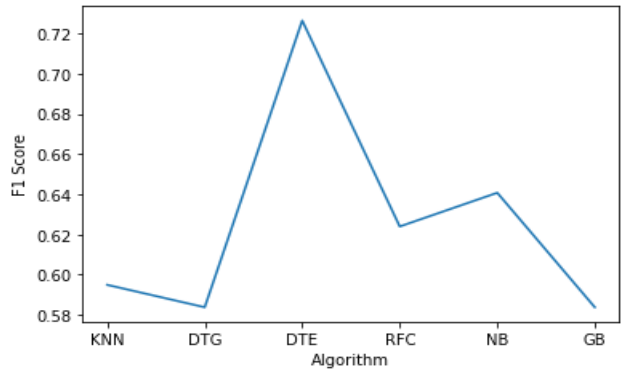

Fig.13 A graph between F1 Score vs Algorithm

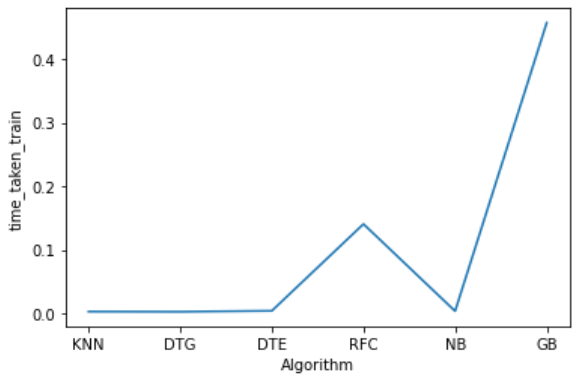

Fig.14 A graph between time taken to train vs Algorithm

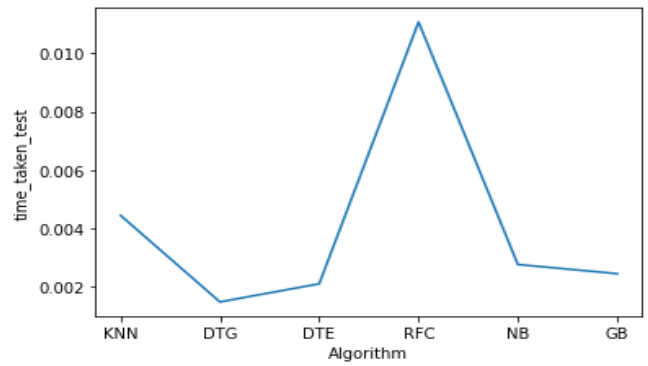

Fig 15 A graph between time taken to test vs Algorithm

Observations from the Graphs

According to the Fig.10 The Decision Tree Algorithm with entropy as the split criterion has the highest accuracy whereas Decision Tree with Gini index as the split criterion and Gradient Boosting algorithms have the least accuracy respectively.

According to the Fig.11 Decision Tree with Entropy has the highest precision whereas Decision Tree with Gini index

and Gradient Boosting have the least precision.

According to the Fig.12 Decision Tree with Entropy has the highest recall whereas Decision Tree with Gini index and Gradient Boosting have the least recall.

According to the Fig.13 Decision Tree with Entropy has the highest F1 Score whereas Decision Tree with Gini index and Gradient Boosting have the least F1 Score.

According to the Fig. 14 Decision Tree with Entropy, KNN, Decision tree with Gini index and Naïve Bayes have the least time taken to train the model whereas Gradient Boosting has taken the highest time taken to train the model among the algorithms used.

According to the Fig.15 Decision Tree with Gini index has the least time taken to test the model, followed by Decision Tree with Entropy whereas Random Forest Classifier has taken the highest time taken to test the model among all the algorithms used.

CONCLUSIONS

Diabetes is a disorder that is very hard to cure and once diagnosed there is a high risk of being subjected to chronic diseases. At this dire situation proper methodologies must be developed to identify the attributes that highly effect the risk of acquiring these chronic diseases. We in this project focused on cardio vascular disease and identified that number of major vessels colored by fluoroscopy, exercised induced angina and chest pain type were highly contributing to cardio vascular disease among diabetic patients. We also used various machine learning algorithms such as KNN, Decision Tree, Random Forest, Naïve Bayes and Gradient Boosting for classification and prediction. We compared the algorithms based on the performance metrics such as accuracy, precision, recall, F1 score, time taken to train and time taken to test. We identified that Decision Tree with entropy as the split criterion achieved the highest accuracy of 76.9%. We also visualized performance metrics vs algorithms to better analyze the algorithms.

This work is implemented based on less than 50 diabetic people. The project can be extended to group patients by using unsupervised learning methods to analyze the data further. More feature analysis methods can be used to identify the highly contributing parameters.

REFERENCES

https://idf.org/ aboutdiabetes/complications.html

Pangi Vijaya Nirmala, Mani Gudivada, and Chikkam Vijaya Lashmi, Comparative Study of the Prevalence of Type-2 Diabetes Mellitus in Various Demographic Regions of Andhra Pradesh, India: a Population based Study, International Journal of Maternal and Child Health and AIDS, Vol. 5, Issue 2, 2016, pp. 103-111.

M.D. Amzad Hossen et.al, Supervised Machine Learning-Based Cardiovascular Disease Analysis and Prediction, International Journal on Advanced Aspects of Computational Intelligence and Applications of Fuzzy Logic and Soft Computing, 2021.

Bhavesh Dhande, Kartik Bamble, Sahil Chavan, and Tabassum Maktum, Diabetes & Heart Disease Prediction Using Machine Learning, ITM Web of Conferences 44, 03057 (2022), ICACC-2022.

Abdul Saboor et. al, A Method for Improving Prediction of Human Heart Disease Using Machine Learning Algorithms, International Journal on Advanced Artificial Intelligence Technologies for Service Enhancement on the Internet of Medical Things, 2022.

M. F. Faruque, I. H. Sarker, et al., "Performance analysis of machine learning techniques to predict diabetes mellitus," in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 1–4, IEEE, 2019.

D. Vigneswari, N.K. Kumar, V.G. Raj, A. Gugan, and S. Vikash, "Machine learning tree classifiers in predicting diabetes mellitus," in 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 84–87, IEEE, 2019.

H. Abbas, L. Alic, M. Rios, M. Abdul-Ghani, and K. Qaraqe, "Predicting diabetes in healthy population through machine learning," in 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), pp. 567–570, IEEE, 2019.

Sakthivel M, Sivanantham S, Kamalraj R & Krishnamoorthy V (2022). An Analysis of Machine Learning Depend on Q-MIND for Defencing the Distributed Denial of Service Attack on Software Defined Network. International Journal of Early Childhood Special Education. 2022; 14(05): 3769 – 3776.

P. Yogendra Prasad, Dumpa Prasad, D Naga Malleswari (2022).Implementation of Machine Learning Based Google Teachable Machine in Early Childhood Education. International Journal of Early Childhood Special Education (INT-JECSE); 14(3): 1308-5581.

P. Lakshmi Sagar (2022).Resolving sets and Dimension in Bidiakis Cube and Durer Graphs. International journal of Neuro Quantology; 20: 4988-4992.

CH Prathima, R. Anusuya, M. Ram Kumar Prabhu (2022). Comprehensive Design Analysis of Digital Marketing in Agriculture Sector. International Journal of Early Childhood Special Education; 14(5):2022.

Silpa C, Ram Prakash Reddy Arava, K.K. Baseer. Agri Farm: Crop And F International Journal of Early Childhood Special Education ertilizer Recommendation System for High Yield Farming Using Machine Learning Algorithms; 14(5): 2022.

P. Dhanalakshmi et al. (2022). Application of Machine Learning in Multi-Directional Model to Follow Solar Energy Using Photo Sensor Matrix. International journal of Photoenergy; 9: 1-9.