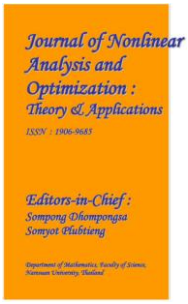
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1 : 2024
ISSN : **1906-9685**



EFFECTIVE FEATURE ENGINEERING TECHNIQUE FOR HEART DISEASE PREDICTION WITH MACHINE LEARNING

CH. KIRAN BABU, Department of Information Technology, NRI Institute of Technology, Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212
M. ISWARYA, Department of Information Technology, NRI Institute of Technology, Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212
R. MANIKANTA KUMAR Department of Information Technology, NRI Institute of Technology, Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212
M. PAVAN SAI Department of Information Technology, NRI Institute of Technology, Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212

Abstract:
Heart failure is a chronic ailment that affects millions of individuals worldwide. An efficient machine learning-based approach is needed to forecast early heart failure health status and take the appropriate actions to address this pervasive issue. Even while medication is still the primary treatment for heart failure, exercise is increasingly acknowledged as an effective adjunct therapy. In this work, we created a machine learning-based approach to improve the identification of heart failure based on patient health parameter data. Our research contributes to better early detection of heart failure, which helps save patients' lives. In order to pick the most salient features to improve performance, we used nine machine learning-based methods for comparison and introduced a novel Principal Component Heart Failure (PCHF) feature engineering technique. To attain the maximum accuracy scores, we innovated by developing a new feature set to optimise the suggested PCHF process. The eight best-fit characteristics form the foundation of the recently generated dataset. To evaluate the effectiveness of various algorithms, we carried out in-depth studies. It is impressive that the suggested decision tree approach achieved a high accuracy score, outperforming other state-of-the-art studies and applied machine learning models. The cross-validation methodology was successfully used to validate all applied procedures. There will be major scientific contributions to the medical community from our suggested research endeavour.

1. Introduction

One of the leading causes of death worldwide is coronary heart disease. One of the hardest things to do in clinical data analysis is to diagnose a heart condition. Machine learning (ML) is helpful for diagnosis in terms of prediction and decision-making based on data generated by the global healthcare industry. We have also observed the use of ML approaches in the field of disease prediction in medicine. Numerous studies on the use of ML classifiers for the prediction of heart disease have been presented in this respect. Eleven machine learning classifiers were employed in this work to uncover critical features, hence improving the predictability of heart disease. Several feature combinations and well-known classification methods were employed to introduce the prediction model. With gradient boosted trees and multilayer perceptron’s, we were able to attain 95% accuracy in the heart disease prediction model. With an accuracy rate of 96%, the Random Forest model performs better in the prediction of heart disease. The medical sciences are very diverse due to advancements in technology and computational power, particularly in the diagnosis of heart disorders in humans. It is currently one of the deadliest heart conditions that affect people and has a major impact on life expectancy. In this sense, machine learning algorithms are effective and trustworthy resources for identifying and classifying individuals with heart disease and those in good condition. The recommended study states

that we used a range of machine learning algorithms to identify and predict human heart disease. We then used the heart disease dataset to assess the model's performance using various metrics, including sensitivity, specificity, F-measure, and classification accuracy. In order to do this, we applied nine machine learning classifiers—AB, LR, ET, MNB, CART, SVM, LDA, RF, and XGB—on the final dataset both before and after the hyperparameter tuning of the machine learning classifiers. Additionally, we do specific preprocessing, dataset standardisation, and hyperparameter tuning on the standard heart disease dataset to verify their accuracy. Additionally, we used the conventional K-fold cross-validation method to train and validate the machine learning algorithms. Finally, the experimental result indicated that the accuracy of the prediction classifiers with hyperparameter tuning improved and achieved notable results with data standardization and the hyperparameter tuning of the machine learning classifiers. Cardiovascular diseases are responsible for twenty five per cent of the deaths in India and across the world. Indians are affected by CVDs at a much earlier age compared to the other demographics. Given the severity of the situation an early prediction of heart disease is of utmost importance. Heart diseases can be determined using a combination of clinical and pathological data. Hence, we use machine learning algorithms to predict heart disease in human beings using the above mentioned data. The goal of this research is to compare the different models and choose the most appropriate model for the forecast. This article employs the following models: Random Forest, K-Nearest Neighbours, SVM, Naive Bayes, Decision Tree, and Logistic Regression. With an accuracy of 85.25%, logistic regression was shown to work the best [1-15].

2. Proposed System

We created a machine learning-based method to improve the identification of heart failure based on patient health parameter data. Our research contributes to better early detection of heart failure, which helps save patients' lives. Nine machine learning based algorithms, including decision trees, logistic regression, random forests, support vector machines, naive bases, k-nearest neighbours, multilayer perceptrons, and extreme gradient boosting, were used for comparison. Additionally, a novel Principal Component Heart Failure (PCHF) feature engineering technique was proposed to identify the most salient features in order to improve performance. To attain the maximum accuracy scores, we innovated by developing a new feature set to optimise the suggested PCHF process. The eight best-fit characteristics form the foundation of the recently generated dataset. To evaluate the effectiveness of various algorithms, we carried out in-depth studies. The cross-validation method was used to validate every applied methodology.

ADVANTAGES OF PROPOSED SYSTEM

1. To choose the most noticeable features, we employ a brand-new Principal Component Heart Failure (PCHF) feature engineering method.
2. On the other hand, our work uses nine machine learning techniques, which covers a wider range.
3. To increase accuracy, we are working to optimise the PCHF method to choose the most crucial features
4. Using a novel strategy, we produced a new dataset optimised for accuracy enhancement, based on the eight best-fit characteristics.

2.1 MODULES

1. Data loading: using this module we are going to import the dataset.
2. Description of the Dataset: Although there are 76 attributes in this database, only 14 of them are used in the published experiments. Specifically, the Cleveland database is the only one that ML researchers have utilised thus far. The patient's heart disease status is indicated in the "goal" field. Its values range from 0 (no presence) to 4 in integers. The Cleveland database has been used in experiments primarily to try and separate presence (values 1, 2, 3, 4) from absence (value 0). Recently, dummy values were inserted in place of the patients' names and social security numbers in the database.
3. LOGISTIC REGRESSION: Also referred to as a logit model, this kind of statistical model is frequently utilised in predictive analytics and categorization. Logistic regression uses a dataset of independent variables to estimate the likelihood of an event occurring, such as voting or not.

4. DECISION TREE: Applied to both classification and regression applications, a decision tree is a non-parametric supervised learning technique. With a root node, branches, internal nodes, and leaf nodes, it has a hierarchical tree structure.
5. RANDOM FOREST: Developed by Leo Breiman and Adele Cutler, Random Forest is a popular machine learning technique that aggregates the output of several decision trees to produce a single conclusion. Its versatility and ease of use, combined with its ability to handle both regression and classification problems, have driven its popularity.
6. SUPPORT VECTOR MACHINE: Support Vector Machines (SVM) are strong supervised algorithms that perform best on complex but smaller datasets. Although Support Vector Machines, often known as SVMs, are useful for classification as well as regression applications, their performance is generally greatest in the former.
7. KNN: The k-nearest neighbours algorithm, sometimes referred to as KNN or k-NN, is a non-parametric supervised learning classifier that classifies or predicts how a single data point will be grouped based on closeness.
8. MULTILAYER PERCEPTRON: The term "multilayer perceptron" (MLP) refers to a misnomer for a contemporary feedforward artificial neural network. It is made up of fully connected neurons arranged in at least three layers and having a nonlinear type of activation function. It is notable for its capacity to discern nonlinearly separable data. It is a misnomer because, unlike modern networks, the original perceptron employed a nonlinear type of activation function—the Heaviside step function.
9. NAIVE BAYES: The Naïve Bayes Classifier is a straightforward and highly efficient classification technique that facilitates the rapid development of machine learning models capable of making accurate predictions. Being a probabilistic classifier, it makes predictions based on the likelihood that an object will occur.
10. XGBoost is a distributed gradient boosting library that has been optimised for training machine learning models in a scalable and effective manner. It is an ensemble learning technique that generates a stronger prediction by aggregating the predictions of several weak models.
11. GRADIENT BOOSTING: For classification and regression tasks, gradient boosting is a well-liked boosting strategy in machine learning. One type of ensemble learning technique is called "boosting," in which the model is trained successively, with each new model attempting to improve upon the one before it. It turns a number of ineffective learners into effective ones.
12. STACKING CLASSIFIER: A stacking classifier is an ensemble learning technique that builds a single "super" model by merging several classification models. As a result, performance can frequently be enhanced because the merged model can benefit from each unique model's advantages.

2.2 SYSTEM ARCHITECTURE

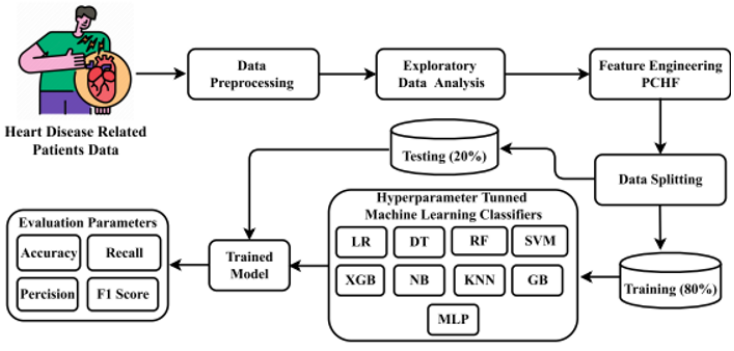


Figure.1. System architecture

2.3 DATA FLOW DIAGRAM

1. Another name for the DFD is a bubble chart. A system can be represented using this straightforward graphical formalism in terms of the input data it receives, the different operations it performs on that data, and the output data it generates.
2. One of the most crucial modelling tools is the data flow diagram (DFD). The components of the system are modelled using it. These elements consist of the system's procedure, the data it uses, an outside party that communicates with it, and the information flows within it.

3. DFD illustrates the flow of information through the system and the various changes that alter it. This method uses graphics to show how information flows and the changes made to data as it goes from input to output.

4. Another name for DFD is a bubble chart. Any level of abstraction can be utilised to portray a system using a DFD. DFD can be divided into phases that correspond to escalating functional detail and information flow.

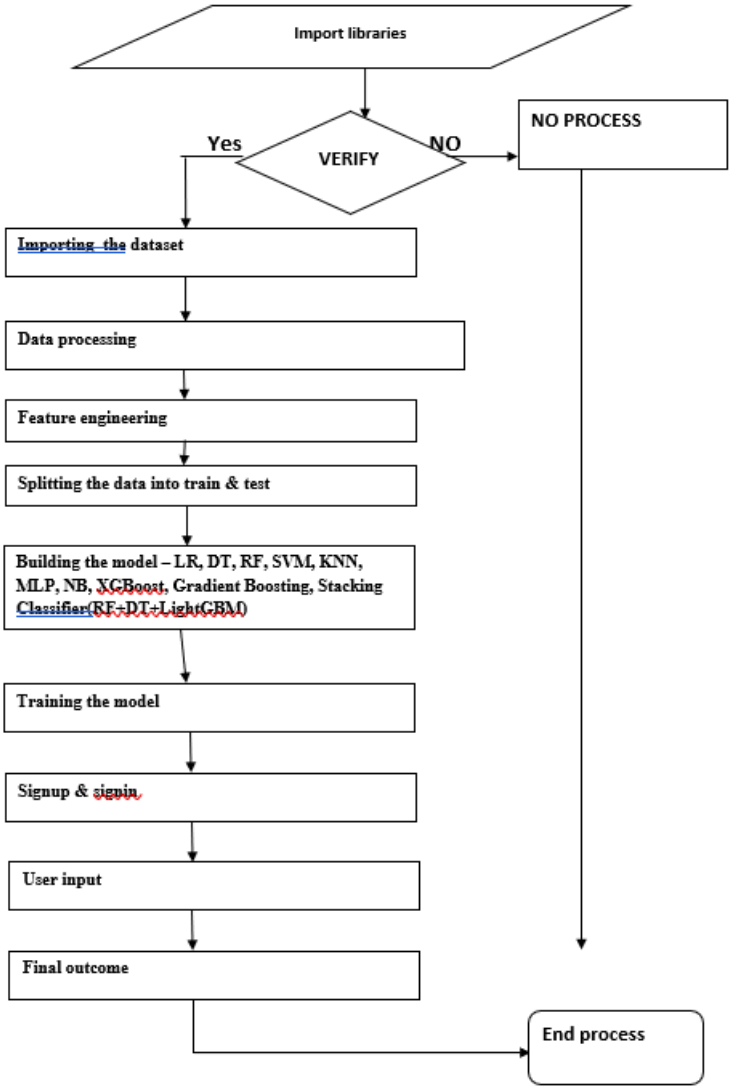


Figure.2. Data Flow Diagram

2.4 UML DIAGRAMS

Unified Modelling Language is known as UML. An industry-standard general-purpose modelling language used in object-oriented software engineering is called UML. The Object Management Group developed and oversees the standard. The intention is for UML to spread as a standard language for modelling object-oriented software. The two main parts of UML as it exists now are a notation and a meta-model. In the future, UML may also include other processes or methods that are connected to it. A common language for business modelling and other non-software systems, as well as for defining, visualising, building, and documenting software system artefacts, is called the Unified Modelling Language.

The UML is an assembly of top engineering techniques that have been successfully applied to the modelling of complicated and sizable systems.

Creating objects-oriented software and the software development process both heavily rely on the UML. The UML primarily expresses software project design through graphical notations.

The following are the main objectives of the UML design:

1. Give users access to an expressive, ready-to-use visual modelling language so they can create and share valuable models.

2. To expand the fundamental ideas, offer methods for specialisation and extendibility.
3. Be unaffected by specific development processes or programming languages.
4. Offer an official foundation for comprehending the modelling language.
5. Promote the market expansion for OO tools.
6. Encourage the use of higher level development ideas like components, frameworks, partnerships, and patterns.
7. Combine the finest techniques.

2.5 Use case diagram

According to the Unified Modelling Language (UML), a use case diagram is a particular kind of behavioural diagram that is produced from and defined by a use case study. Its objective is to provide a graphical summary of the functionality that a system offers in terms of actors, use cases (representations of their goals), and any interdependencies among those use cases. A use case diagram's primary goal is to display which actors receive which system functionalities. It is possible to illustrate the roles of the system's actors.

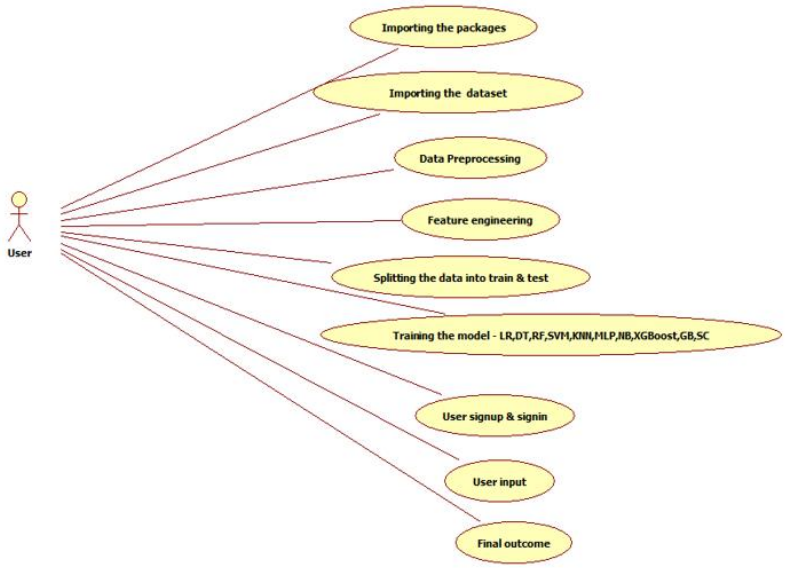


Figure.3. Use Case Diagram

2.6 Class diagram

The use case diagram and the system's comprehensive design are both improved by the class diagram. The actors identified in the use case diagram are categorised into a number of related classes by the class diagram. There are two types of relationships that can exist between the classes: "is-a" relationships and "has-a" relationships. It's possible that every class in the class diagram can perform certain functions. The "methods" of the class refer to these features that it offers. In addition, every class might possess specific "attributes" that allow for class uniqueness.

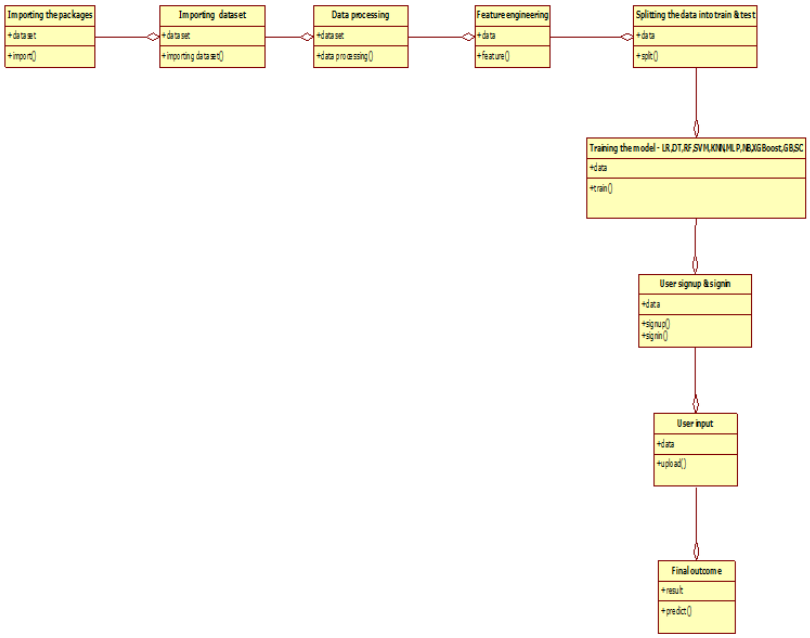


Figure.4. Class Diagram

2.7 Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

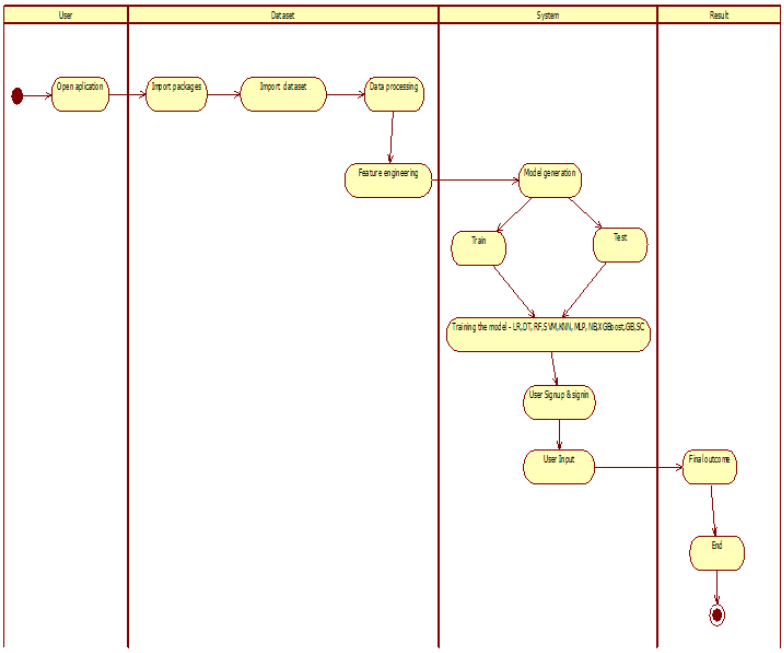


Figure.5. Activity Diagram

2.8 Sequence diagram

The way various system items interact with one another is depicted in a sequence diagram. A sequence diagram's time-ordering is one of its key features. This indicates that a step-by-step representation of the precise order in which the items interacted is provided. In the sequence diagram, various objects communicate with one another by sending "messages".

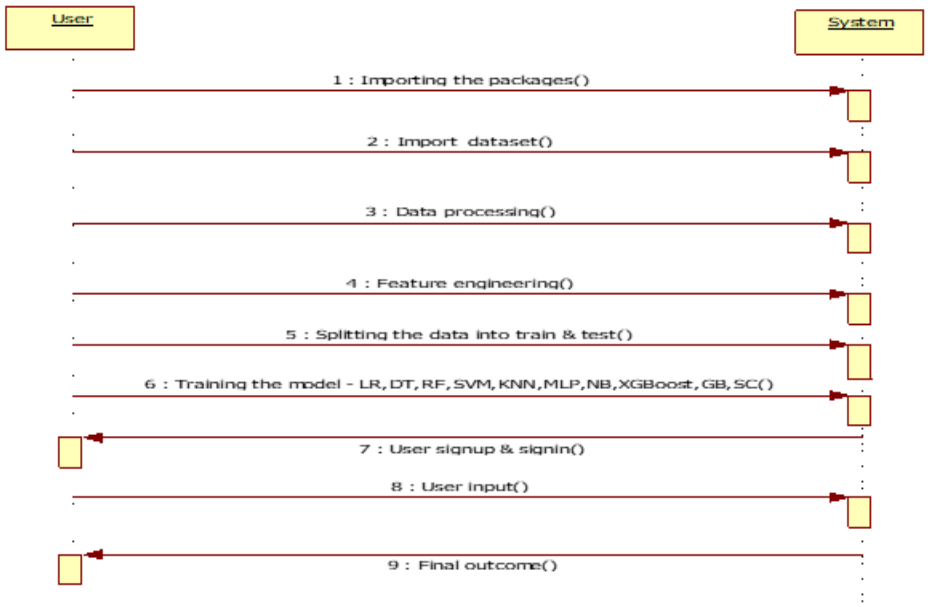


Figure.6. Sequence Diagram

2.9 Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

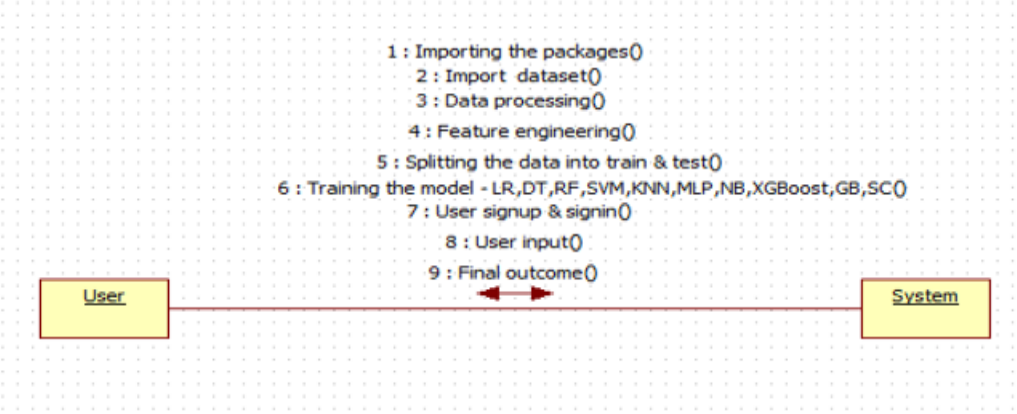


Figure.7. Collaborative Diagram

2.10. Component diagram

The high-level components that comprise the system are represented in the component diagram. A high-level representation of the system's components and their relationships is shown in this diagram. The parts removed from the system after it has completed the development or manufacturing stage are shown in a component diagram.

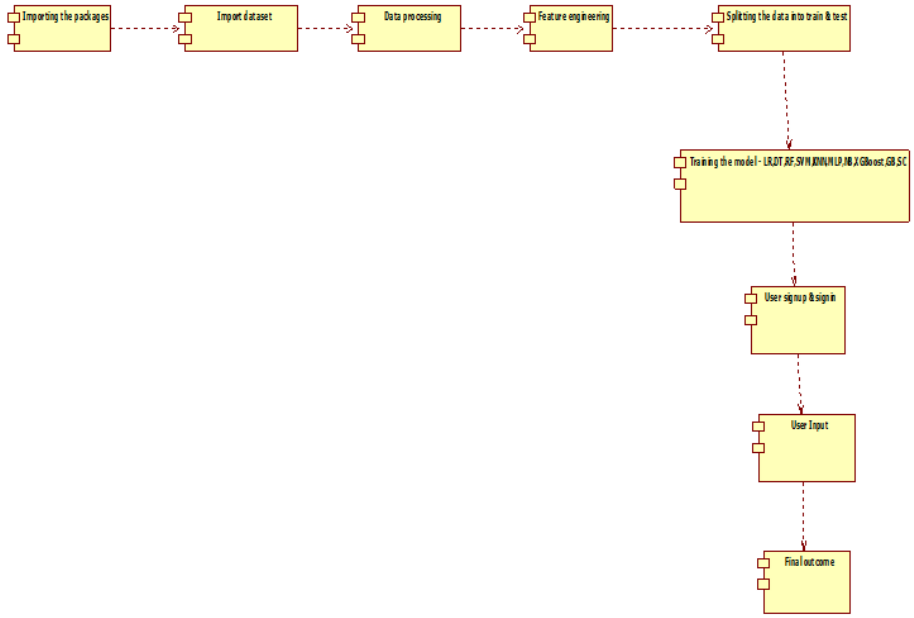


Figure.8. Component Diagram

2.11 Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

Figure.9. Deployment Diagram

2.12 Software Testing Strategies:

The greatest strategy to make software engineering testing more effective is to optimise the approach. A software testing plan outlines the steps that must be taken in order to produce a high-quality final product, including what, when, and how. To accomplish this main goal, the following software testing techniques—as well as their combinations—are typically employed:

Static Examination:

Static testing is an early-stage testing approach that is carried out without really operating the development product. In essence, desk-checking is necessary to find errors and problems in the code itself. This kind of pre-deployment inspection is crucial since it helps prevent issues brought on by coding errors and deficiencies in the software's structure.

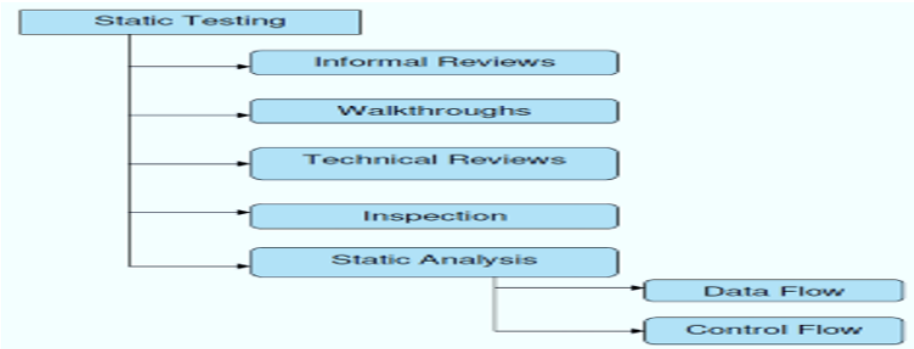


Figure.10. Static Testing

2.13 Structural Testing

Software cannot be tested efficiently unless it is run. White-box testing, another name for structural testing, is necessary to find and correct flaws and faults that surface during the pre-production phase

of the software development process. Regression testing is being used for unit testing depending on the programme structure. To expedite the development process at this point, it is typically an automated procedure operating inside the test automation framework. With complete access to the software's architecture and data flows (data flows testing), developers and quality assurance engineers are able to monitor any alterations (mutation testing) in the behaviour of the system by contrasting the test results with those of earlier iterations (control flow testing).

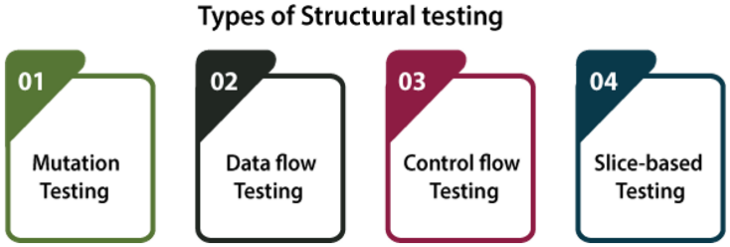


Figure.11. Structural Testing

2.14 Behavioural Testing

Rather than the mechanics underlying these reactions, the final testing phase concentrates on how the programme responds to different activities. Put differently, behavioural testing, commonly referred to as black-box testing, relies on conducting multiple tests, the majority of which are manual, in order to examine the product from the perspective of the user. In order to perform usability tests and respond to faults in a manner similar to that of ordinary users of the product, quality assurance engineers typically possess specialised information about a company or other purposes of the software, sometimes known as "the black box." If repetitive tasks are necessary, behavioural testing may also involve automation (regression tests) to remove human error. To see how the product handles an activity like filling out 100 registration forms on the internet, for instance, it would be better if this test were automated.

Figure.12. Behavioural Testing

3. Results and Discussion

DATA SET

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
52	1	0	125	212	0	1	168	0	1	2	2	3	0
53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
61	1	0	148	203	0	1	161	0	0	2	1	3	0
62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
58	0	0	100	248	0	0	122	0	1	1	0	2	1
58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
54	1	0	122	286	0	0	116	1	3.2	1	2	2	0
71	0	0	112	149	0	1	125	0	1.6	1	0	2	1
43	0	0	132	341	1	0	136	1	3	1	0	3	0
34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
51	1	0	140	298	0	1	122	1	4.2	1	3	3	0
52	1	0	128	204	1	1	156	1	1	1	0	0	0
34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
54	1	0	124	266	0	0	109	1	2.2	1	1	3	0
50	0	1	120	244	0	1	162	0	1.1	2	0	2	1
58	1	2	140	211	1	0	165	0	0	2	0	2	1
60	1	2	140	185	0	0	155	0	3	1	0	2	0
67	0	0	106	223	0	1	142	0	0.3	2	2	2	1
45	1	0	104	208	0	0	148	1	3	1	0	2	1
63	0	2	135	252	0	0	172	0	0	2	0	2	1
42	0	2	120	209	0	1	173	0	0	1	0	2	1
61	0	0	145	307	0	0	146	1	1	1	0	3	0

Figure.13. Data Set

OUTPUTS

```
data = pd.read_csv("heart.csv")
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Figure.14 Reading The Data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
age         1025 non-null int64
sex         1025 non-null int64
cp          1025 non-null int64
trestbps    1025 non-null int64
chol        1025 non-null int64
fbs         1025 non-null int64
restecg     1025 non-null int64
thalach     1025 non-null int64
exang       1025 non-null int64
oldpeak     1025 non-null float64
slope       1025 non-null int64
ca          1025 non-null int64
thal        1025 non-null int64
target      1025 non-null int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Figure.15. Data Info

```
sns.countplot(x=data['target'])
```

```
<AxesSubplot:xlabel='target', ylabel='count'>
```

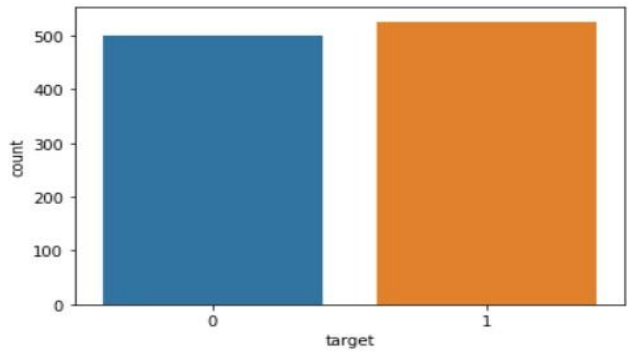


Figure.16. Count Plot Bar Graph

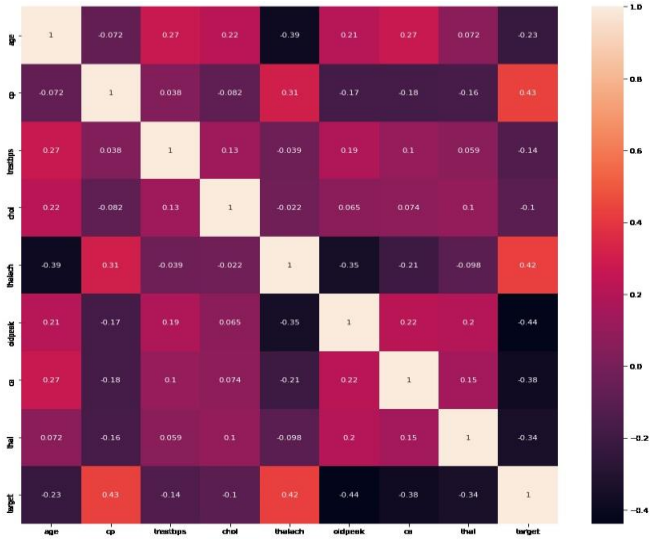


Figure.17. Heat Map

```
rfecv.fit(X,y)
```

```
RFECV(cv=11, estimator=RandomForestClassifier(random_state=0),
      scoring='accuracy')
```

```
#to get number of the features
display('Number of features:', rfecv.n_features_)

#to get feature names
list(X.columns[rfecv.support_])
```

```
'Number of features:'

8

['age', 'cp', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca', 'thal']
```

Figure.18. Calculating No. of Features

```
x_pca.shape
```

```
(1025, 8)
```

```
x_pca
```

```
array([[-34.6134893 , -18.61896596,  -2.95938327, ...,  -1.26361176,
          0.18405391,  -0.49047147],
       [-42.62662174,  -4.18089964,  10.99496101, ...,   0.82126433,
         -1.57509571,  -0.41531675],
       [-70.20160302,  29.34943055,  16.30737271, ...,   0.56332711,
         -1.67353233,  -0.55010646],
       ...,
       [ 27.99632937,  26.6437281 , -27.15221246, ...,  -0.53656787,
          0.08199777,   0.40548151],
       [  6.58865617, -12.52011982, -21.19461765, ...,  -0.79270423,
         -0.93684794,   0.26001829],
       [-57.96015356,  35.82239743, -12.38472877, ...,  -0.45106542,
         -0.23535768,  -0.61374537]])
```

Figure.19. Principle Component Analysis

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x_pca, y, test_size = 0.2, random_state = 42)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((820, 8), (820,), (205, 8), (205,))
```

Figure.20. Train, Test and Split

```
result
```

	ML Model	Accuracy	f1_score	Recall	Precision
0	Logistic Regression	0.751	0.816	0.724	0.767
1	Decision Tree	0.985	0.971	1.000	0.985
2	Random Forest	0.971	0.942	1.000	0.970
3	Support Vector Machine	0.639	0.660	0.636	0.648
4	KNN	0.902	0.913	0.895	0.904
5	MLP	0.590	0.184	1.000	0.311
6	Naive Bayes	0.751	0.777	0.741	0.758
7	XGBoost	0.912	0.913	0.913	0.913
8	Gradient Boosting	0.922	0.922	0.922	0.922

Figure.21. Result Of Each Model

```
#creating dataframe
result1 = pd.DataFrame({ 'ML Model' : ML_Model1,
                         'Accuracy' : mean,
                       })
result1
```

	ML Model	Accuracy
0	Logistic Regression K-Fold	0.833
1	Decision Tree K-Fold	0.985
2	Random Forest K-Fold	0.989
3	SVM K-Fold	0.720
4	KNN K-Fold	0.872
5	MLP K-Fold	0.727
6	Naive Bayes K-Fold	0.828
7	XGBoost K-Fold	0.955
8	Gradient Boosting K-Fold	0.959

Figure.22. Accuracy of Each Model

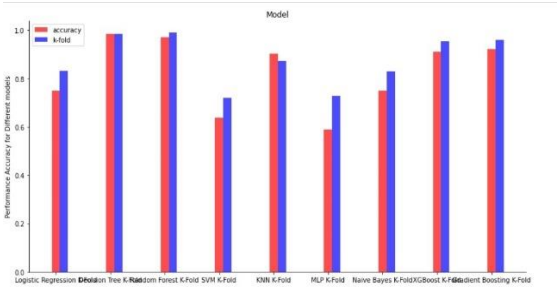


Figure.23. Graph

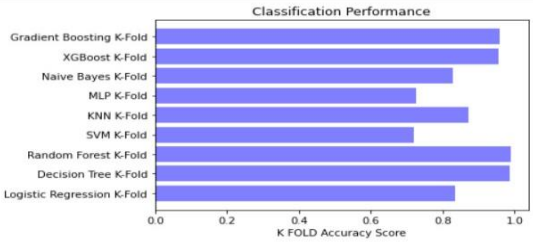


Figure.24. K- Fold Accuracy Score Comparison

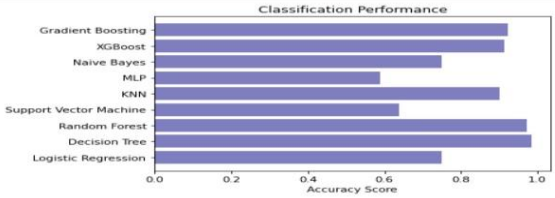


Figure.25. Accuracy

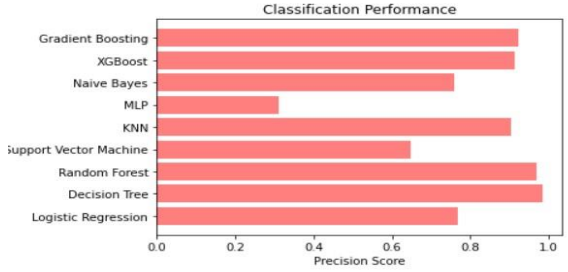


Figure.26. Precision

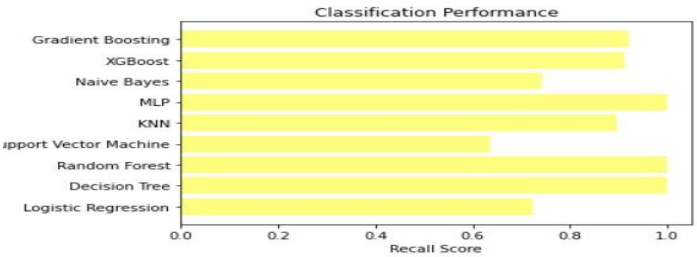


Figure.27. Recall

Figure.28. F1- Score

	Unnamed: 0	age	cp	trestbps	chol	thalach	oldpeak	ca	thal	target
0	0	52	0	125	212	168	1.0	2	3	0
1	1	53	0	140	203	155	3.1	0	3	0
2	2	70	0	145	174	125	2.6	0	3	0
3	3	61	0	148	203	161	0.0	1	3	0
4	4	62	0	138	294	106	1.9	3	2	0

Figure.29. Modelling for frontend with selective features

```
import joblib
filename = 'model.sav'
joblib.dump(tree, filename)
```

```
['model.sav']
```

Figure.30. Model Save

4. Conclusion

This study suggests utilising machine learning techniques to predict cardiac failure. The applied models are constructed using the dataset consisting of 1025 patient records. To improve performance, a new PCHF feature engineering method is suggested that picks the eight most noticeable characteristics. The applied machine learning algorithms in comparison are the logistic regression, random forest, support vector machine, decision tree, extreme gradient boosting, naive base, k-nearest neighbours, multilayer perceptron, and gradient boosting. It took 0.005 runtime computations to obtain 100% accuracy with the suggested DT technique. Each learning model is subjected to the cross-validation procedure based on 10-fold data in order to validate its performance. Our suggested approach is applicable to the detection of heart failure and exceeded the most recent research.

References

[1] M. Gjoreski, M. Simjanoska, A. Gradišek, A. Peterlin, M. Gams, and G. Poglajen, ‘‘Chronic heart failure detection from heart sounds using a stack of machine-learning classifiers,’’ in Proc. Int. Conf. Intell. Environments (IE), Aug. 2017, pp. 14–19.

[2] G. Savarese and L. H. Lund, ‘‘Global public health burden of heart failure,’’ Cardiac Failure Rev., vol. 3, no. 1, p. 7, 2017.

[3] E. J. Benjamin et al., ‘‘Heart disease and stroke statistics—2019 update: A report from the American heart association,’’ Circulation, vol. 139, no. 10, pp. e56–e528, 2019.

[4] A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha, ‘‘Secure and robust machine learning for healthcare: A survey,’’ IEEE Rev. Biomed. Eng., vol. 14, pp. 156–180, 2021.

[5] C. A. U. Hassan, J. Iqbal, R. Irfan, S. Hussain, A. D. Algarni, S. S. H. Bukhari, N. Alturki, and S. S. Ullah, ‘‘Effectively predicting the presence of coronary heart disease using machine learning classifiers,’’ Sensors, vol. 22, no. 19, p. 7227, Sep. 2022.

[6] R. Katarya and S. K. Meena, ‘‘Machine learning techniques for heart disease prediction: A comparative study and analysis,’’ Health Technol., vol. 11, no. 1, pp. 87–97, Jan. 2021.

[7] P. Rani, R. Kumar, N. M. O. S. Ahmed, and A. Jain, ‘‘A decision support system for heart disease prediction based upon machine learning,’’ J. Reliable Intell. Environments, vol. 7, no. 3, pp. 263–275, Sep. 2021.

[8] N. S. Mansur Huang, Z. Ibrahim, and N. Mat Diah, ‘‘Machine learning techniques for early heart failure prediction,’’ Malaysian J. Comput. (MJoC), vol. 6, no. 2, pp. 872–884, 2021.

[9] T. Amarbayasgalan, V. Pham, N. Theera-Umpon, Y. Piao, and K. H. Ryu, ‘‘An efficient prediction method for coronary heart disease risk based on two deep neural networks trained on well-ordered training datasets,’’ IEEE Access, vol. 9, pp. 135210–135223, 2021.
[10] R. Bharti, A. Khamparia, M. Shabaz, G. Dhiman, S. Pande, and P. Singh, ‘‘Prediction of heart disease using a combination of machine learning and deep learning,’’ Comput. Intell. Neurosci., vol. 2021, pp. 1–11, Jul. 2021.
[11] F. S. Alotaibi, ‘‘Implementation of machine learning model to predict heart failure disease,’’ Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 6, pp. 1–8, 2019.
[12] D. K. Plati, E. E. Tripoliti, A. Bechlioulis, A. Rammos, I. Dimou, L. Lakkas, C. Watson, K. McDonald, M. Ledwidge, R. Pharithi, J. Gallagher, L. K. Michalis, Y. Goletsis, K. K. Naka, and D. I. Fotiadis, ‘‘A machine learning approach for chronic heart failure diagnosis,’’ Diagnostics, vol. 11, no. 10, p. 1863, Oct. 2021.
[13] A. Saboor, M. Usman, S. Ali, A. Samad, M. F. Abrar, and N. Ullah, ‘‘A method for improving prediction of human heart disease using machine learning algorithms,’’ Mobile Inf. Syst., vol. 2022, pp. 1–9, Mar. 2022.
[14] S. Sarah, M. K. Gourisaria, S. Khare, and H. Das, ‘‘Heart disease prediction using core machine learning techniques—A comparative study,’’ in Advances in Data and Information Sciences. Springer, 2022, pp. 247–260.
[15] C. Trevisan, G. Sergi, and S. Maggi, ‘‘Gender differences in brain-heart connection,’’ Brain and Heart Dynamics. 2020, pp. 937–951.