

“Heart Lens Disease Prediction Using Machine Learning”

KANETI. RUPA ¹, Miss. G. PRAMEELA ², Mr.K. SURESH³

M.Tech Research scholar in Department of computer science and engineering & KITS, DIVILI

Assistant professor in Department of computer science and engineering & KITS, DIVILI

Assistant professor in Department of computer science and engineering & KITS, DIVILI

Abstract - Heart Lens Disease Prediction using Machine Learning is an intelligent healthcare framework designed to support early identification of cardiovascular risks through advanced data-driven analysis. The proposed system acts as a “Heart Lens,” providing a focused view of patient health conditions by analyzing clinical, physiological, and lifestyle-related attributes. Machine learning techniques are employed to process complex medical datasets and identify patterns associated with heart disease occurrence.

To improve prediction accuracy, feature selection methods including Chi-Square, Analysis of Variance (ANOVA), and Mutual Information (MI) were utilized to extract the most significant attributes. Multiple machine learning algorithms such as Naïve Bayes, Support Vector Machine (SVM), Decision Tree, Random Forest, K-Nearest Neighbor (KNN), Logistic Regression, AdaBoost, Bagging Classifier, XGBoost, and Voting Classifier were evaluated and compared.

Experimental results demonstrated that ensemble learning approaches achieved superior performance across different feature subsets. Furthermore, a Stacking Classifier integrating Boosted Decision Tree, Extra Tree, and LightGBM models was developed to enhance predictive capability. To address class imbalance, SMOTE was applied, resulting in improved classification accuracy and reliability. The proposed Heart Lens framework achieved high prediction performance, providing a practical and efficient solution for early heart disease risk assessment.

Additionally, a Flask-based web application with secure authentication was implemented to enable real-time prediction, user-friendly interaction, and controlled access for healthcare professionals and patients. This system contributes toward intelligent, accessible, and preventive cardiovascular healthcare management

Key Words: machine learning ,python programming.

1.INTRODUCTION

Heart-related diseases continue to be one of the leading causes of mortality worldwide, creating a significant burden on healthcare systems. Early identification of cardiovascular abnormalities plays a crucial role in reducing complications and improving patient survival rates. However, traditional diagnostic procedures often depend on extensive clinical tests, medical imaging, and expert evaluation, which may not always be accessible or affordable for every patient.

To address these challenges, intelligent healthcare solutions are increasingly being developed using Machine Learning (ML) techniques. In this context, Heart Lens Disease Prediction refers to an advanced predictive framework that

acts as a “digital lens” for examining patient health data and identifying potential heart disease risks at an early stage. By analyzing various clinical parameters such as age, blood pressure, cholesterol levels, heart rate, blood sugar levels, and other medical indicators, the system provides a comprehensive assessment of cardiovascular health.

Machine Learning algorithms have demonstrated remarkable capability in discovering hidden patterns and relationships within large medical datasets. These algorithms can learn from historical patient records and generate accurate predictions regarding the likelihood of heart disease occurrence. Such predictive models assist healthcare professionals in making timely decisions and enable preventive measures before the disease progresses to critical stages.

The proposed Heart Lens Disease Prediction Using Machine Learning system aims to improve diagnostic efficiency, reduce human error, and provide reliable risk assessment through data-driven intelligence. By integrating healthcare analytics with machine learning methodologies, the system contributes to the development of smart healthcare environments where early detection, continuous monitoring, and personalized treatment recommendations become more effective and accessible.

applications. The Heart Lens framework leverages these advancements to offer a scalable and cost-effective solution for cardiovascular risk prediction. The ultimate objective of this system is to support clinicians, enhance patient awareness, and improve overall healthcare outcomes through accurate and timely disease prediction

OBJECTIVE

The primary objective of the Heart Lens project is to develop an intelligent machine learning-based system capable of analyzing patient health data and predicting the risk of heart-related diseases with improved accuracy and efficiency.

PROBLEM STATEMENT

Heart disease is difficult to detect in its early stages, yet early detection is critical for effective treatment. Diagnoses, on the other hand, are made based on medical specialists' personal experiences and perspectives of the condition, which raises the likelihood of errors, inhibits appropriate treatment, lengthens treatment timeframes, and drives up costs. A thorough consideration of these factors has led healthcare bodies like the Centers for Disease Control and

prevention (CDC) to gather massive amounts of data from patients and store them in datasets or databases, which are used to build machine learning models.

3. SOFTWARE REQUIREMENTS

performance A software requirements specification (SRS) for a Heart Disease Prediction System outlines the necessary functional capabilities, non-functional metrics, and technological components to build an accurate medical diagnostic tool.

Furthermore, the growing availability of healthcare data and advancements in artificial intelligence have accelerated the adoption of predictive healthcare

- **Operating Systems:** Windows 10/11, Ubuntu 2 Linux, (optional)
- **Programming Language:** Python 3.10 or above
- **Web Framework:** Flask
- **Development Environment:** visual studio (vs code) Pycharm ide
- **Data visualization Libraries:**
 - matplotlib
 - seaborn
- **Database:**
 - SQLite
 - MySQL(OPTIONAL)
- **FRONT-END**
 - HTML5
 - CSS3
 - JAVASCRIPT
 - BOOTSTRAP
- **Authentication module**
 - Flask-Login
 - Werkzeug Security
- **Dataset Format**
 - CSV(comma separated values)

Machine learning libraries:

- Numpy
- Pandas
- Scikit-learn
- XGboost
- Lightgbm
- Imbalanced-learn(SMOTE)
- **Browser Support:** Chrome / Edge / Firefox

Platform – The proposed Heart Lens Disease Prediction System is developed on a Python-based machine learning platform using Scikit-learn, XGBoost, and LightGBM algorithms. The system utilizes the Flask framework for web application development and runs on Windows operating systems, providing an efficient environment for data processing, model training, prediction, and user interaction through a secure web interface

4 HARDWARE REQUIREMENT

The proposed Heart Lens Disease Prediction System requires a computing device capable of handling machine learning algorithms, feature selection techniques, and web-based application deployment. The hardware configuration should support efficient data preprocessing, model training, and prediction operations. The minimum and recommended hardware configurations are as follows:

- **Processor:** Intel i3 /i5/ i7 / AMD Equivalent

- **RAM:** Minimum 4GB (8 GB recommended)
 - **Storage:** SSD with 10–50 GB of free space
 - **Display:** 1366*768 resolution or higher
1660 / RTX 3060 or higher
 - **Hard Disk** 20 GB Free Space
 - **System Architecture** 64-bit System
 - **Input Devices** Keyboard and Mouse
 - **Network Connectivity** Internet Connection
- system or application. The following sub-sections discuss the various aspects of hardware requirements.

5. LITERATURE SURVEY

Early detection allows for timely treatment and continuous monitoring by healthcare providers, which is essential but often limited by the inability of medical professionals to provide constant patient supervision. Early detection of cardiac problems and continuous patient monitoring by physicians can help reduce death rates. Doctors cannot constantly have contact with patients, and heart disease detection is not always accurate.. We have chosen features using chi-square, ANOVA, and mutual information methods. The three feature groups chosen were SF-1, SF-2, and SF-3. The study employed ten machine learning algorithms to determine the most accurate technique and feature subset fit. The classification algorithms used include support vector machines (SVM),

XGBoost, bagging, decision trees (DT), and random forests (RF). We evaluated the proposed heart disease prediction technique using a private dataset, a public dataset, and different cross-validation methods. We used the Synthetic Minority Oversampling Technique (SMOTE) to eliminate inconsistent data and discover the machine learning algorithm that achieves the most accurate heart

EXISTING SYSTEM

Existing systems for predicting cardiovascular disease (CVD) have employed various machine learning (ML) models and deep learning (DL) algorithms with significant success. One approach utilized large datasets of electronic health records linked with socio-demographic information to stratify CVD risks, achieving high accuracy in a multi-ethnic population. Another method applied deep learning to clinical data and coronary computed tomography angiography (CCTA) images, demonstrating high accuracy in predicting coronary artery disease (CAD).

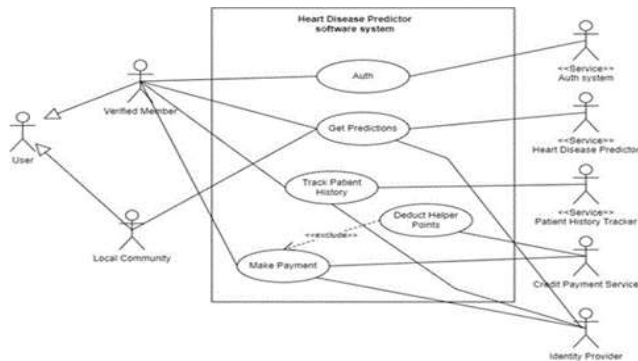
Additionally, other studies have explored ML models such as Decision Trees (DTs), K-Nearest Neighbors (KNN), and Random Forests (RFs) for predicting CVD, reporting high performance with these techniques.

6. Proposed System

The proposed system focuses on predicting heart disease using machine learning techniques applied to a dataset derived from both publicly available and privately collected records, ensuring diverse patient demographics and clinical features. To enhance prediction quality, three feature selection techniques—Chi-square, Analysis of Variance (ANOVA), and Mutual Information (MI)—were employed, producing three optimized feature subsets (SF-1, SF-2, and

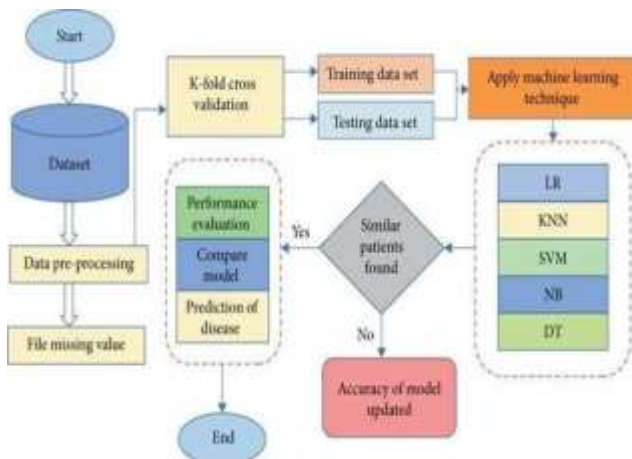
SF-3). Classification is performed using ten machine learning algorithms: Naive Bayes, Support Vector Machine (SVM), XGBoost, AdaBoost, Bagging Classifier, Decision Tree, K-Nearest Neighbor (KNN), Random Forest, Logistic Regression, and a Voting Classifier combining multiple models.

SYSTEM ARCHITECTURE



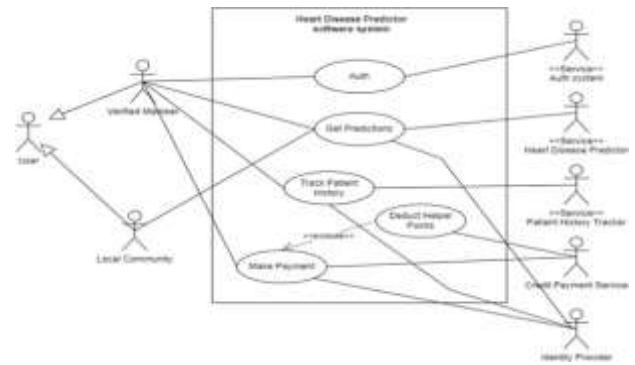
DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a graphical tool that represents the flow of data through a system, highlighting how inputs are transformed into outputs. In the context of a heart disease prediction system, a DFD helps visualize how patient data—such as biomarkers and clinical records—is collected, processed through machine learning algorithms, and produces heart disease predictions. It simplifies the understanding of data interactions and system functionality for both technical and non-technical stakeholder ensuring effective communication and system development.



Use Case diagram

A use case diagram visually represents interactions between users and the system, illustrating the functional requirements and services provided. It identifies actors, their roles, and the actions they can perform, helping developers understand system behavior. It ensures clarity of requirements, supports communication between stakeholders, and provides a blueprint for implementing user-centered functionalities effectively. By defining the scope of interactions, it helps streamline system design, reduces ambiguities, and highlights essential features for prediction, data authentication all.



The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities.

These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class

Class Diagram

A class diagram models the system's static structure by showing classes, attributes, methods, and relationships. It helps organize data structures such as patient information, feature sets, and model outputs, providing clarity on dependencies and inheritance. Developers can efficiently design and maintain modules, understand data flow, and ensure consistent implementation. By defining classes and their interactions, it improves system readability, enforces standardization, and supports future extension or integration of new algorithms, models, or modules for prediction

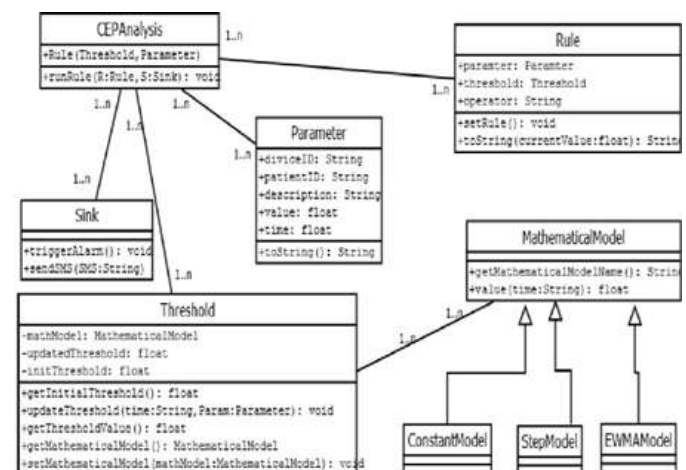


Fig. 7. Class diagram.

Class Diagram

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

IMPLEMENTATION

▪ **Data Loading:** The data loading module is responsible for importing heart disease datasets into the system, ensuring that the data is accessible

▪ **Data Preprocessing:** Data preprocessing involves preparing the heart disease dataset to enhance model accuracy and reliability. This includes cleaning the data by identifying and correcting inconsistencies, missing values.

Data Visualization: The data visualization module helps interpret complex medical datasets by providing graphical representations of relationships between patient features and outcomes

▪ **Label Encoding:** Label encoding converts categorical variables into numerical values, enabling machine learning algorithms to process non-numeric data effectively..

▪ **Feature Selection:** The feature selection module identifies the most relevant predictors of heart disease to enhance model performance. Techniques such as SelectKBest combined with the Chi-square (Chi2) statistic evaluate the relationship between each feature and the target variable..

▪ **Splitting Data into Train & Test:** This module divides the dataset into training and testing subsets to evaluate model performance effectively..

▪ **Model Generation:** The model generation module involves building and training various machine learning algorithms to predict heart disease.

▪ **User Signup & Login:** The signup and login module provides a secure mechanism for users to register and access the system.

13.Algorithms

Naive Bayes The Naive Bayes classifier offers a fast, probabilistic approach suited to datasets with categorical or discretized clinical attributes. It models the posterior probability of a cardiac outcome by applying Bayes' theorem under an assumption of conditional independence among features.

Support Vector Machine (SVM)

Support Vector Machine constructs an optimal separating hyperplane in feature space by maximizing the margin between classes. It is effective in high-dimensional settings and can capture nonlinear relationships via kernel functions (e.g., radial basis, polynomial).

XG Boost

XG Boost is an efficient gradient boosting tree ensemble that builds additive models by fitting successive decision trees to residual errors.

AdaBoost

AdaBoost is a boosting framework that sequentially trains weak learners, typically shallow decision trees, while reweighting training samples to emphasize previously misclassified instances.

Bagging Classifier

Bagging (bootstrap aggregation) reduces variance by training multiple base estimators on different bootstrap samples and aggregating their predictions

Decision Tree

the feature space via recursive binary split chosen to maximize class purity (using Gini impurity or information gain).

K-Nearest Neighbors (KNN)

K-Nearest Neighbors is an instance-based classifier that assigns labels based on the majority class among the k closest training samples according to a distance metric (commonly Euclidean). Random Forest Random Forest builds an ensemble of decision trees using bootstrap sampling and random feature selection at each split, thereby reducing correlation among trees and controlling overfitting

```
print(f"There are {df.shape[0]} patients and {df.shape[1]} features in the dataset.")
df.info()

print_heading("DATASET INFORMATION SUMMARY")

print("\n1. The dataset contains 920 entries across 16 features pertinent to heart disease studies.")

print("\n2. Complete data is available for 'id', 'age', 'sex', 'dataset', 'cp', and the outcome variable 'num'.")

print("\n3. Significant missing data is found in 'ca' with 66.41% missing.")

print("\n4. Significant missing data is found in 'thal' with 52.83% missing.")

print("\n5. 'Slope' also has a considerable amount of missing data, with only 611 non-null entries.")

print("\n6. Variables 'trestbps', 'chol', 'tbs', 'thalch', 'exang', and 'oldpeak' have some missing values but to a lesser extent.")

df.describe()
```

```
from scipy import stats

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder, QuantileTransformer,
OneHotEncoder

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder, QuantileTransformer,
OneHotEncoder

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, mean_squared_error, r2_score

import warnings

warnings.filterwarnings('ignore')

# print heading - for display purposes only

def print_heading(heading):
```

```
# Select target variable(s)
y = df_clean[target_variable]

# Initialize LabelEncoder
le = LabelEncoder()

#le = OneHotEncoder()
for col in X.columns:
    # If column is of object type
    if X[col].dtype == 'object':
        # Encode the column
        X[col] = le.fit_transform(X[col])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)

# Initialize RandomForestClassifier
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Print heading
print_heading('Model performance for {target_variable}'.upper())

# Print accuracy score
print('The accuracy of the model is: {round(accuracy_score(y_test, y_pred),2)}%')

# Print classification report
print('Classification report:\n {classification_report(y_test, y_pred)}')
```

```
encode_and_predict(df_encoded_pred, 'ca', rf_model_ca)
# Update the 'ca' column in df_missing with the predicted values
df_missing.loc[:, 'ca'] = df_encoded_pred['ca']

Predicting missing values for thal - Thalassemia category.
# Create a new dataframe and drop the target variables
df_encoded_pred = df_missing.drop(['thal'], axis=1)

# Encode categorical features and predict the 'thal' column in df_encoded_pred
encode_and_predict(df_encoded_pred, 'thal', rf_model_thal)

# Update the 'thal' column in df_missing with the predicted values
df_missing.loc[:, 'thal'] = df_encoded_pred['thal']

Next step - Merging the datasets
# merge df_clean and df_missing
df = pd.concat([df_clean, df_missing])

# List of numerical columns that have less than a certain threshold of missing values
numerical_cols = [col for col in df.columns
                  if df[col].isnull().sum() < 1000]

# List of categorical columns
cat_cols = [col for col in df.columns
            if df[col].dtype == 'object']

# Apply simple imputation to numerical and categorical columns
num_imputer = SimpleImputer(strategy='median')
cat_imputer = SimpleImputer(strategy='most_frequent')

# Apply imputation to numerical and categorical columns
for col in numerical_cols:
    df[col] = num_imputer.fit_transform(df[col])

for col in categorical_cols:
    df[col] = cat_imputer.fit_transform(df[col])
```

```
if df.isnull().sum().sum() == 0:
    print("There are no missing values in the dataset.") else:
    print("There are missing values in the dataset.")

# create a heatmap for df to check missing values
plt.figure(figsize=(16, 6))
sns.heatmap(df.isnull(), cbar=False)
plt.title('Missing values after imputation')
plt.show()

# Plotting the histogram and boxplot to check the data distribution for the numerical columns
def plot_histogram_and_boxplot(df, numerical_cols, cyan_palette=["#00FFFF"]):
    # Set to "dark" for a black background
    sns.set(style="dark", palette="dark")

    # Set the figure size to accommodate the subplots
    plt.figure(figsize=(15, len(numerical_cols) * 5))
```

DECISION TREE:

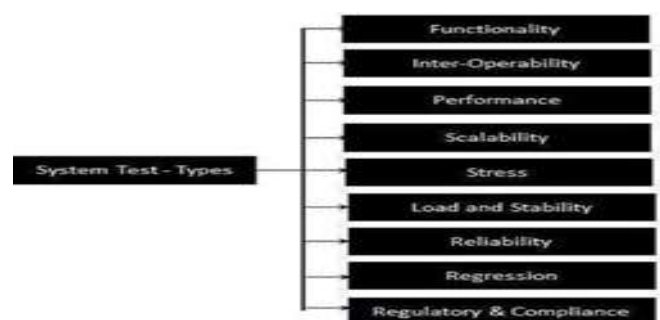
The Decision Tree algorithm provided easy-to-understand classification rules and offered high interpretability. Medical practitioners can easily understand the decision-making process generated by the model. However, the model showed a tendency toward overfitting when dealing with certain data distributions, which may affect its generalization capability.

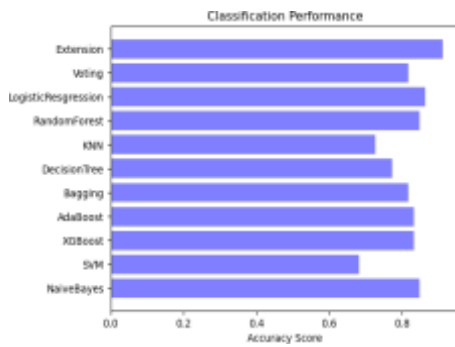
The K-Nearest Neighbors (KNN) algorithm achieved satisfactory classification performance by identifying similar instances within the dataset. Although KNN can effectively classify data points based on proximity, its computational cost increases with larger datasets, making it less efficient for real-time applications.

Support Vector Machine (SVM) demonstrated strong classification capability by identifying optimal decision boundaries between classes. The algorithm performed effectively in handling high-dimensional data and produced reliable predictions. However, the training process required greater computational resources compared to some other algorithms.

SYSTEM TESTING

The software system testing is done to check every feature and performance of the software after all its components have been integrated. It tests if the software is working properly as per the requirements and it is able to solve the customers' needs. The software system testing is performed to detect any issues occurring after integration of multiple units of the software. It finds defects in the integrated software as well as in the complete software as a whole.





ML Model	Accuracy	Precision	Recall	F1_score
NaiveBayes	0.848	0.850	0.848	0.849
SVM	0.682	0.686	0.682	0.682
XGBoost	0.833	0.834	0.833	0.833
AdaBoost	0.833	0.835	0.835	0.834
Bagging	0.818	0.826	0.818	0.819
DecisionTree	0.773	0.784	0.773	0.774
KNN	0.727	0.729	0.727	0.728
RandomForest	0.848	0.850	0.848	0.849
LogisticRegression	0.864	0.864	0.864	0.864
Voting	0.818	0.818	0.818	0.818
Extension	0.912	0.915	0.912	0.913

Performance Testing

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User signin	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

Performance Testing is performed to verify the performance, stability, reliability etc of the software

Load Testing

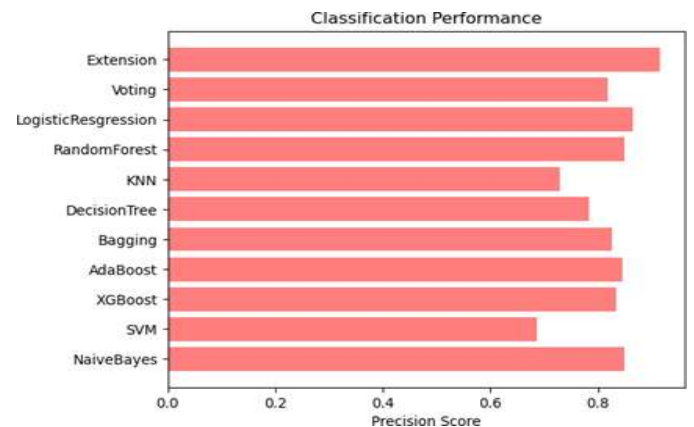
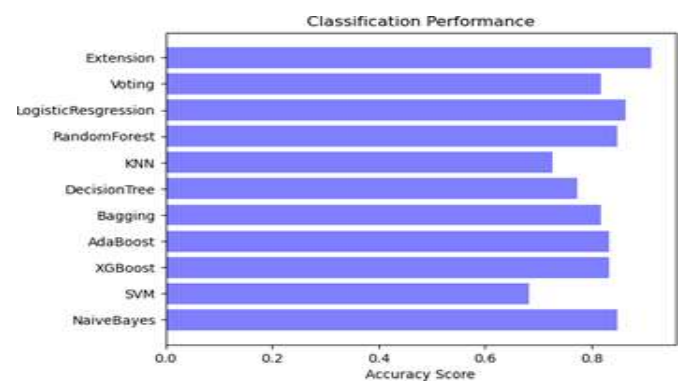
Load Testing is performed to verify the amount of load or traffic that the software can accommodate before it crashes. It finds the threshold limit of the maximum count of users that the software can bear at a time after its software undergoes a breakdown.

Stress Testing

Stress Testing is performed to verify if there exists any security problems leading to the potential scope of hacking, and other vulnerabilities

Scalability Testing

Scalability Testing is performed to verify the performance of the software with respect to its capacity to scale up or down the count of the user



Performance Evolution Table – ANOVA FS Performance
Evolution Table – Chi Square FS

CONCLUSION

In conclusion, the system demonstrates the effectiveness of machine learning techniques for predicting heart disease by leveraging a diverse dataset derived from both publicly available and privately collected records containing clinical physiological and lifestyle attributes the implementation of feature selection techniques such as Chi-square, Analysis of Variance (ANOVA), and Mutual Information (MI) allowed the creation of optimized subsets SF-1, SF-2, and SF-3, ensuring improved classification performance. Ten machine learning algorithms, including Naive Bayes, Support Vector Machine, XGBoost, AdaBoost, Bagging Classifier, Decision Tree, K-Nearest Neighbor, Random Forest, Logistic Regression, and a Voting Classifier, were applied to assess predictive accuracy. Evaluation was conducted with cross-validation, where the Voting Classifier showed consistent improvements across feature subsets, highlighting the strength of ensemble methods. Furthermore, a Stacking Classifier combining Boosted Decision Tree, ExtraTree, and LightGBM was introduced to further enhance results. SMOTE was employed exclusively in this configuration to address class

REFERENCES

1. El-Sofany, H., Bouallegue, B., & El-Latif, Y. M. A. (2024). A proposed technique for predicting heart disease using machine learning algorithms and an explainable AI method. *Scientific Reports*, 14(1), 23277.
2. Pathan, M. S., Nag, A., Pathan, M. M., & Dev, S. (2022). Analyzing the impact of feature selection on the accuracy of heart disease prediction. *Healthcare Analytics*, 2, 100060.
3. Biswas, N., Ali, M. M., Rahaman, M. A., Islam, M., Mia, M. R., Azam, S., ... & Moni, M. A. (2023). Machine learning-based model to predict heart disease in early stage employing different feature selection techniques. *BioMed Research International*, 2023(1), 6864343
4. Alshraideh, M., Alshraideh, N., Alshraideh, A., Alkayed, Y., Al Trabsheh, Y., & Alshraideh, B. (2024). Enhancing heart attack prediction with machine learning: A study at Jordan University Hospital. *Applied Computational Intelligence and Soft Computing*, 2024(1), 5080332
5. Dwivedi, A. K. (2018). Performance evaluation of different machine learning techniques for prediction of heart disease. *Neural Computing and Applications*, 29(10), 685–693.
6. Dissanayake, K., & Md Johar, M. G. (2021). Comparative study on heart disease prediction using feature selection techniques on classification algorithms. *Applied Computational Intelligence and Soft Computing*, 2021(1), 5581806
7. Ahmad, G. N., Ullah, S., Algethami, A., Fatima, H., & Akhter, S. M. H. (2022). Comparative study of optimum medical diagnosis of human heart disease using machine learning technique with and without sequential feature selection. *IEEE Access*, 10, 23808–23828.

BIOGRAPHIES (Optional not mandatory)

Miss KANETI RUPA
RESEARCH SCHOLAR
M.Tech(CSE)
KITS-DIVILI