Heart Disease Prediction Using Machine Learning Techniques

A PROJECT REPORT

Submitted by,

Ms. Kakarla Mehrunnisa	**- 20201CSE0665**
Ms. Latthika S	**- 20201CSE0675**
Ms. Pallavi	**- 20201CSE0685**

Under the guidance of,

Dr. Aarif Ahamed S

in partial fulfillment for the award of the

degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AT

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"Heart Disease Prediction Using Machine Learning Techniques"** being submitted by "Kakarla Mehrunnisa, Latthika S and Pallavi" bearing roll number(s) "20201CSE0665, 20201CSE0675 and 20201CSE0685" in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Dr. Aarif Ahamed S
Assistant Professor
School of CSE
Presidency University

Dr. Pallavi R
Head of Department
School of CSE
Presidency University

Dr. C. KALAIARASAN
Associate Dean
School of CSE and IS
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE and IS
Presidency University

Dr. Md. SAMEERUDDIN KHAN
Dean
School of CSE and IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Heart Disease Prediction using Machine Learning Techniques** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Dr. Aarif Ahamed S, Assistant Professor, School of Computer Science Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Kakarla Mehrunnisa (20201CSE0665)

Latthika S (20201CSE0675)

Pallavi (20201CSE0685)

ABSTRACT

Heart diseases including heart attacks, cause about 31% of global deaths, remaining a significant health threat despite preventability. Limited tech advancements and awareness, especially in developing nations, amplify this challenge. Machine learning offers promise in tackling this issue, with studies advocating ensemble methods for accurate predictive models. These models analyze extensive medical data to efficiently predict heart diseases, undergoing stages like data exploration, feature selection, model implementation, and comparative analysis. A project using Logistic Regression, Naive Bayes and Random Forest initially identified top-performing models, later refined to CatBoost, RandomForest and XGBoost through cross-validation and tuning. A hybrid model, combining Logistic Regression, CatBoost and RandomForest achieved a 97% accuracy, showcasing improved precision, recall, F1 score and ROC-AUC. This underscores machine learning's potential in enhancing predictive accuracy and refining strategies to combat heart diseases effectively.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science and Engineering and School of Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science Engineering and School of Information Science, Presidency University and **Dr. Pallavi R**, Head of the Department, School of Computer Science Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Dr. Aarif Ahamed S**, Assistant Professor, School of Computer Science Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators **Mr. Mohammed Zia Ur Rahman, Mr. Peniel John Whistely**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Latthika S

Kakarla Mehrunnisa

Pallavi

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Introduction

In our modern, fast-paced world, the demands of daily life often overshadow the need for self-care, resulting in increased stress and health negligence. Conditions such as heart disease, cancer, and tuberculosis continue to claim numerous lives yearly, with cardiovascular disease (CVD) standing out as the leading cause of global mortality, responsible for nearly 31% of deaths worldwide (World Health Organization, WHO). Over a span of 15 years, WHO reported approximately 15.2 million deaths attributed to heart-related diseases, highlighting the persistent high mortality rates associated with these conditions. In India alone, heart-related ailments caused a staggering economic loss of about $237 billion between 2005 and 2015.

The heart, a pivotal organ orchestrating blood circulation, supplies vital oxygen and nutrients to every part of the body. Any dysfunction within this vital organ significantly affects the functionality of other organs, presenting a substantial challenge. Unhealthy dietary patterns and the rapid pace of modern lifestyles significantly contribute to the elevated risk of heart-related diseases.

Harnessing the power of machine learning and deep learning techniques to analyze diverse patient data within the medical field presents a promising avenue for assessing risks, identifying symptoms, and predicting heart-related diseases. Factors such as diabetes, smoking, excessive alcohol consumption, high cholesterol, high blood pressure, and obesity significantly heighten the likelihood of heart issues. While managing these factors can mitigate the risk, heart diseases can manifest irrespective of gender or age.

This study embarks on a comprehensive analysis of these factors with the aim of predicting heart disease using machine learning and deep learning techniques. Such predictive capabilities hold immense potential for revolutionizing healthcare and positively impacting

lives. The study further delves into defining various types of heart diseases, such as Coronary Artery Disease, Heart Failure, Congenital Heart Disease, and Cardiomyopathy, each with its distinct characteristics and impact on the cardiovascular system.
In the initial phase of the project, the dataset was loaded, and several machine learning algorithms were employed, including Logistic Regression, K-Nearest Neighbors, RandomForest, CatBoost, XgBoost, Naive Bayes, Support Vector Machines, and Stochastic Gradient Descent. The performance metrics were computed, and the top-performing models were identified based on Accuracy, Precision, Recall, F1 Score, and ROC AUC.

The models performed differently across many measures prior to hyperparameter adjustment. While KNN, SVM, and SGD displayed lower scores across several metrics, other models, such as Logistic Regression, Random Forest, CatBoost, and XGBoost, had reasonably strong performance in terms of accuracy, precision, recall, and F1 scores.

Following this, the dataset underwent cross-validation and hyperparameter tuning, where the same set of algorithms was applied. The performance metrics were re-evaluated, revealing another set of top-performing models based on their enhanced predictive capabilities.The majority of models performed noticeably better across a range of criteria following hyperparameter adjustment. Improved scores in accuracy, precision, recall, and F1 score were attained by models such as Random Forest, CatBoost, XGBoost, and Logistic Regression; this suggests that their parameters have been adjusted more effectively and that their predictive power has increased.

Finally, a hybrid model combining Logistic Regression, CatBoost and Random Forest was constructed using a voting classifier. This model exhibited remarkable predictive performance, achieving significantly high scores across all metrics. It achieved a high accuracy of 97% and demonstrated impressive precision and recall scores, both above 0.97. The F1 score also reflects a balanced trade-off between precision and recall, and the ROC AUC of 0.9984 suggests outstanding overall model performance.

This holistic approach utilizing machine learning techniques showcases the potential to accurately predict heart disease, offering a promising stride toward early identification and intervention in combating cardiovascular ailments. The amalgamation of various algorithms and methodologies underscores the potential for impactful advancements in healthcare and improved patient outcomes.

CHAPTER-2
LITERATURE SURVEY

2.1 Logistic Regression

A real-time system for predicting heart disease from medical data streams is presented in the paper "Heart disease identification from patients' social posts, machine learning solution on spark" by H. Ahmed, E.M.G. Younis, A. Hendawi, and A.A. Ali. It uses Apache Spark and Apache Kafka for processing and machine learning algorithms like Decision Tree and Logistic Regression Classifier. The paper's technique consists of streaming data processing, hyperparameter tweaking and cross-validation, machine learning algorithms, and feature selection methods. has limits about the quality of the data, sample size, and generalizability to other populations.[1]

The paper "Latest trends on heart disease prediction using machine learning and image fusion" by L. Sharan Monica et al. The aim of the study is to develop a program that provides reliable and instant disease diagnosis, which can save lives and improve healthcare. The features of the dataset were analyzed using exploratory data analysis (EDA) techniques, and various machine learning algorithms were applied to find the best performer. The performance evaluation metrics used include sensitivity, specificity, accuracy, kappa statistics, precision, recall, f-measure, Matthew's correlation coefficient (MCC), receiver operating characteristic (ROC), and precision-recall (PRC). The methodology used in the study includes data preprocessing, selection of attributes, and application of machine learning algorithms such as naive Bayes, decision tree, support vector machine (SVM), Logistic Regression and artificial neural network (ANN). The disadvantage of the study is that it is based on a single dataset, and the results may not be generalizable to other datasets. [4]

The paper "Early prediction model for coronary heart disease using genetic algorithms, hyper-parameter optimization and machine learning techniques". Aims to develop an efficient early prediction model for coronary heart disease using machine learning techniques. It features advanced feature selection using genetic algorithms, hyper-parameter optimization, and various classification algorithms such as Decision Trees, Adaptive Boosting, Gaussian Naive Bayes, K-Nearest Neighbors, XGBClassifier, and Gradient Boosting Classifier. The proposed system helps to identify the best set of features for diagnosis using traditional machine learning algorithms along with modern Gradient Boosting approaches.[10]

2.2 Support Vector Machine

The paper "Machine learning for the evaluation of the presence of heart disease" by Ivan Miguel Pires et al. explores the use of various machine learning methods, such as Neural Network, k-Nearest Neighbor, Combined nomenclature rule inducer, Support Vector Machine, and Stochastic Gradient Descent to identify the presence of heart disease. The paper's methodology includes Stratified 20-fold, 10-fold, and 5-fold Cross validations. The research reports that the Decision Tree and Support Vector Machine methods achieved the best accuracy (87.69%) in both 20-fold and 10-fold Cross-validation. The paper's disadvantage is that it lacks a detailed description of the methodology used for feature extraction, feature selection, and model training. [5]

The aim of the paper “Early prediction model for coronary heart disease using genetic algorithms, hyper-parameter optimization and machine learning techniques” by Jinny, S. V., & Mate, Y. V. Aims to identify heart diseases using machine learning methods and various features of heart rate. The methodology includes using Python libraries such as NumPy, pandas, and matplotlib for data processing and visualization. The paper's disadvantage is the lack of a detailed description of the feature selection process. The proposed methods include Random Forest,K-Nearest Neighbor (kNN and Decision Tree Classifier which are evaluated using a dataset from the UCI Machine Learning repository. The paper's aim is to develop a reliable and accurate heart disease detection system using machine learning techniques, which can be useful for healthcare applications and early detection of heart diseases. [6]

The paper "Machine learning model matters its accuracy: a comparative study of ensemble learning and AutoML using heart disease prediction" by Yagyanath Rimal, Siddhartha Paudel, Navneet Sharma, and Abeer Alsadoon compares the accuracy of different machine learning models for heart disease prediction. The paper evaluates eighteen machine learning models, including support vector, logistic regression, and neural network models, and recommends the generalized linear model, gradient boosting model, distributed random forest model, and extra tree model for heart disease classification. The paper's methodology includes using a dataset from the UCI Machine Learning repository and using Stratified 20-fold, 10-fold, and 5-fold Cross validations. The paper's disadvantage is the lack of a detailed description of the feature selection process.The authors emphasize the importance of model accuracy for effective heart disease prediction[8]

The paper "Machine Learning Techniques for Heart Disease Prediction: A Comparative Study and Analysis" by Katarya, R., & Meena, S. K. delves into the application of machine learning for heart disease prediction, emphasizing the increasing prevalence of heart disease and the need for efficient data analysis in the medical sector. The methodology used in the study includes network infrastructure, specification of success parameters, specification of training algorithm It reviews various risk factors such as cholesterol, blood pressure, blood sugar, and heart rate, and their influence on heart disease. The methodology includes the use of algorithms such as Logistic Regression, K-Nearest Neighbour, Support Vector Machine, Naïve Bayes, and Decision Trees for prediction and classification.[9]

2.3 Random Forest

The paper "Heart disease classification based on ECG using machine learning models" by S. Matin Malakouti presents a study on the automated categorization of Electrocardiography (ECG) data using machine learning algorithms such as Gaussian NB, Random Forest, Logistic Regression, Linear Discriminant Analysis. The paper discusses the advantages and disadvantages of these methods and the use of 10-fold cross-validation to reduce prediction variance and avoid biased assessment. The methodology involves data preprocessing, algorithm hyperparameter adjustment, and 10-fold cross-validation to evaluate the performance of the classification algorithms. The work's disadvantage lies in the limitations of the machine learning and deep learning methods in accurately distinguishing between healthy and sick individuals.The study focuses on the classification of healthy and ill persons based on ECG data, with an emphasis on the advantages and disadvantages of the employed machine learning algorithms. [2]

The paper "Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison" by Md Mamun Ali et al. investigates the use of various machine learning classifiers for heart disease prediction. The study found that the random forest method achieved 100% accuracy, sensitivity, and specificity when applied to a heart disease dataset collected from Kaggle. The methodology used in the study includes network infrastructure, specification of success parameters, specification of training algorithm, and evaluation of the model. The disadvantage of the study is that it is based on a single dataset, and the results may not be generalizable to other datasets. [3]

2.4 Hyper-parameter Tuning

The aim of the literature review paper "Machine learning and deep learning techniques for the analysis of heart disease: a systematic literature review, open challenges and future directions" by Katarya, R., & Meena, S. K. is to explore the use of machine learning and deep learning techniques for the analysis of heart diseases. The analysis of various heart diseases using machine learning and deep learning algorithms such as Convolutional Neural Networks (CNNs) and Hyper-parameter Tuning recurrent neural networks, deep belief networks, long short-term memory, and others investigated by different researchers over the time span. The paper provides a systematic review of the existing literature to guide future research in the healthcare industry. The methodology involves a thorough analysis of 63 articles from 2018 to 2022, using a systematic literature review approach. The paper concludes by discussing the open issues and challenges faced by the numerous researchers [7]

The paper "Data-efficient machine learning methods in the ME-TIME study: Rationale and design of a longitudinal study to detect atrial fibrillation and heart failure from wearables" by Naseri, A., Tax, D., van der Harst, P., Reinders, M., & van der Bilt. Aims to detect atrial fibrillation and heart failure from wearables using machine learning methods. The study involves 200 participants wearing Fitbit smartwatches for 3 months, with heart rate, step counter, and sleep time series data extracted from the data platform. Hyper-parameter Tuning recurrent neural networks, deep belief networks, long short-term memory, and others investigated by different researchers over the time span. The study uses self-supervised and weakly supervised learning techniques to design an ML model that can detect AF and HF from wearable data in a data-efficient manner. The study uses multiple-instance learning (MIL) to predict the cardiovascular outcome of a bag of instances. The data is pseudonymized to protect the data privacy of the participants, and only the researchers have access to the sensor data. [11]

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

While existing methodologies in heart disease prediction using machine learning have showcased advancements, several critical research gaps persist, hindering the optimization and accuracy of predictive models. Key areas requiring further exploration include:

3.1 Limited Feature Exploration

The current models often rely on a predefined set of risk factors such as diabetes, cholesterol levels, and smoking habits. However, these models may overlook other potential influential factors crucial for accurate heart disease prediction. A comprehensive exploration of a broader array of clinical, lifestyle, and genetic features is necessary to enhance the predictive power of models.

3.2 Lack of Model Adaptability

Many existing models demonstrate robust performance within specific datasets or demographic groups, yet they lack adaptability and generalizability across diverse populations or healthcare systems. The need for models that can seamlessly adapt to varying patient demographics and medical settings is critical for their practical applicability.

3.3 Handling Imbalanced Datasets

The imbalance between positive and negative cases of heart disease in datasets poses a significant challenge. Existing models may exhibit bias towards the majority class, impacting their predictive accuracy. Addressing imbalanced datasets to improve the model's performance on minority classes is essential for more reliable predictions.

3.4 Interpretability of Models

Complex machine learning algorithms often lack interpretability, presenting a barrier to their acceptance in clinical settings. Understanding the reasoning behind predictions is crucial for healthcare practitioners and patients. Models with explainable AI techniques are needed to enhance trust and acceptance.

3.5 Standardization of Evaluation Metrics

The variability in evaluation metrics across different studies and models poses challenges in comparing and selecting the most effective models. Establishing standardized evaluation metrics specific to heart disease prediction can facilitate fair comparisons and drive advancements in the field.

CHAPTER-4
PROPOSED METHODOLOGY

4.1 General

Before implementing cross-validation and hyperparameter tuning, Logistic Regression was the leading model, exhibiting commendable accuracy. However, following cross-validation and hyperparameter tuning, CatBoost emerged as the top-performing model, showcasing superior accuracy. Throughout these processes, the Random Forest algorithm consistently demonstrated strong performance both before and after tuning.

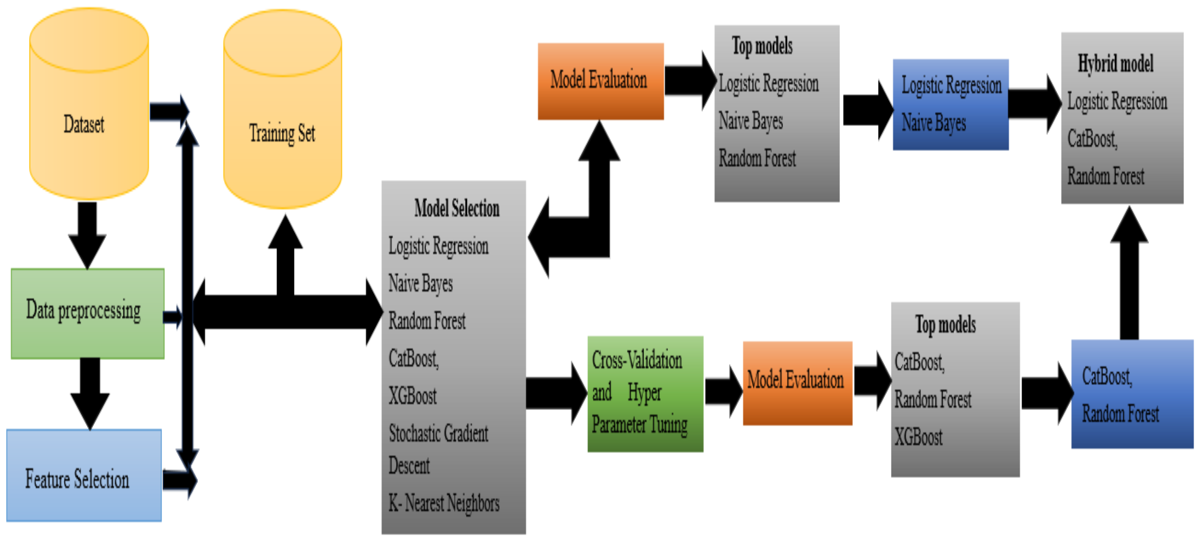
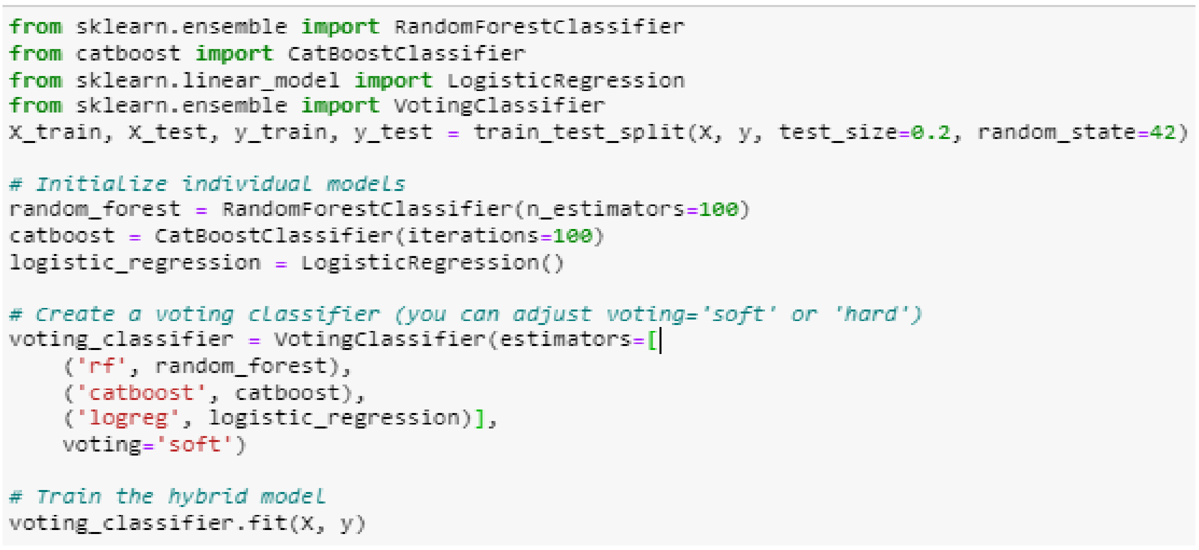


Fig. 4.1 Proposed Architecture

Given the robust performances of Logistic Regression, CatBoost, and Random Forest individually, a hybrid model was crafted using a voting classifier, leveraging the strengths of these three algorithms.



```
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize individual models
random_forest = RandomForestClassifier(n_estimators=100)
catboost = CatBoostClassifier(iterations=100)
logistic_regression = LogisticRegression()

# Create a voting classifier (you can adjust voting='soft' or 'hard')
voting_classifier = VotingClassifier(estimators=[
    ('rf', random_forest),
    ('catboost', catboost),
    ('logreg', logistic_regression)],
    voting='soft')

# Train the hybrid model
voting_classifier.fit(X, y)
```

Fig. 4.2 Code Snippet of Hybrid Model

The code snippet of Fig. 4.2 demonstrates the creation and evaluation of a Voting Classifier ensemble, amalgamating three distinct algorithms: RandomForestClassifier, CatBoostClassifier, and LogisticRegression. The 'voting' parameter is set to 'soft', indicating that the final prediction is determined by the weighted average probability of each classifier.
After training the Voting Classifier on the given dataset, it's evaluated using various metrics. The achieved performance metrics are impressive: an accuracy of 97%, with a precision of 99%, recall of 95%, and an F1 score of 97%. Additionally, the Receiver Operating Characteristic (ROC) curve showcases an Area Under the Curve (AUC) of 99.85%, signifying exceptional model discrimination ability across different thresholds.
The 'soft' voting method considers the probabilities predicted by each model, weighing them and making predictions based on these weighted probabilities. This tends to offer more nuanced decisions by taking into account the confidence levels of individual models. In contrast, 'hard' voting considers only the class labels predicted by each model and selects the majority class as the final prediction. The 'soft' approach can often lead to improved performance when models are well-calibrated and have reliable probability estimates.

Addressing Research Gaps in the existing system the project aims to bridge these research gaps by**:**

4.2 Using Cross-Validation

By using cross-validation we have achieved the following:

Better Model Assessment: Implementing cross-validation enables a more robust assessment of model performance by iteratively splitting the dataset into training and validation sets. This iterative process provides reliable estimates of the model's performance on unseen data, improving its predictive capabilities.

Overfitting Reduction: Leveraging techniques like k-fold cross-validation helps mitigate overfitting by evaluating the model on various subsets of the data. This approach ensures that the model's performance is more generalized and less susceptible to fitting noise in the dataset.

Utilization of Available Data: In scenarios where data is limited, cross-validation maximizes the use of available data by iteratively using portions for training and validation. This efficient utilization optimizes the model's learning process.

4.3 Using Hyperparameter Tuning

By using hyperparameter tuning we have achieved the following:

Improved Model Performance: Hyperparameter tuning optimizes the model's hyperparameters, significantly enhancing its accuracy, precision, and robustness. This process ensures that the model is finely tuned for optimal performance.

Enhanced Generalization: Fine-tuning hyperparameters enables the model to find the optimal configuration, improving its ability to generalize to new, unseen data. This enhances the model's adaptability and predictive power.

Efficient Model Selection: Through hyperparameter tuning, our project aims to select the most suitable model configuration among various options. This ensures that the final model is well-suited for heart disease prediction, enhancing its overall predictive capabilities.

4.4 Developing a Hybrid Model

By developing a hybrid model we have achieved the following:

Complementary Strengths: Our project aims to create a hybrid model that harnesses the strengths of different algorithms or approaches. This strategic fusion aims to improve predictive power by leveraging the unique advantages of each individual model component.

Enhanced Robustness: By integrating multiple models, the hybrid model is expected to be more robust against variations in data, minimizing the weaknesses inherent in individual models. This robustness ensures more reliable predictions.

Improved Accuracy: The strategic design of our hybrid model aims to outperform individual models by effectively capturing diverse patterns present in the data. This amalgamation leads to higher predictive accuracy, crucial for effective heart disease prediction.

By leveraging these techniques, our project seeks to revolutionize heart disease prediction models, addressing critical gaps and paving the way for more accurate, adaptable, and impactful healthcare solutions.

CHAPTER-5
OBJECTIVE

The primary aim of this project is to harness the capabilities of machine learning methodologies to significantly enhance the accuracy and reliability in predicting heart disease. By leveraging advanced algorithms and predictive models, it seeks to analyze diverse datasets comprising patient health metrics, historical records, and pertinent risk factors associated with cardiovascular ailments. Through the strategic application of machine learning techniques such as Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, and Gradient Boosting, our goal is to develop a predictive framework that not only identifies potential cardiac issues but also delineates nuanced patterns and relationships within the data. The emphasis lies in creating a sophisticated predictive model capable of early detection, precise risk assessment, and personalized prognosis, ultimately contributing to proactive interventions and improved patient outcomes. The integration of machine learning in this domain aims to revolutionize conventional diagnostic approaches, empowering healthcare professionals with a powerful tool for accurate and timely identification of individuals at risk of heart disease.

CHAPTER-6
SYSTEM DESIGN & IMPLEMENTATION

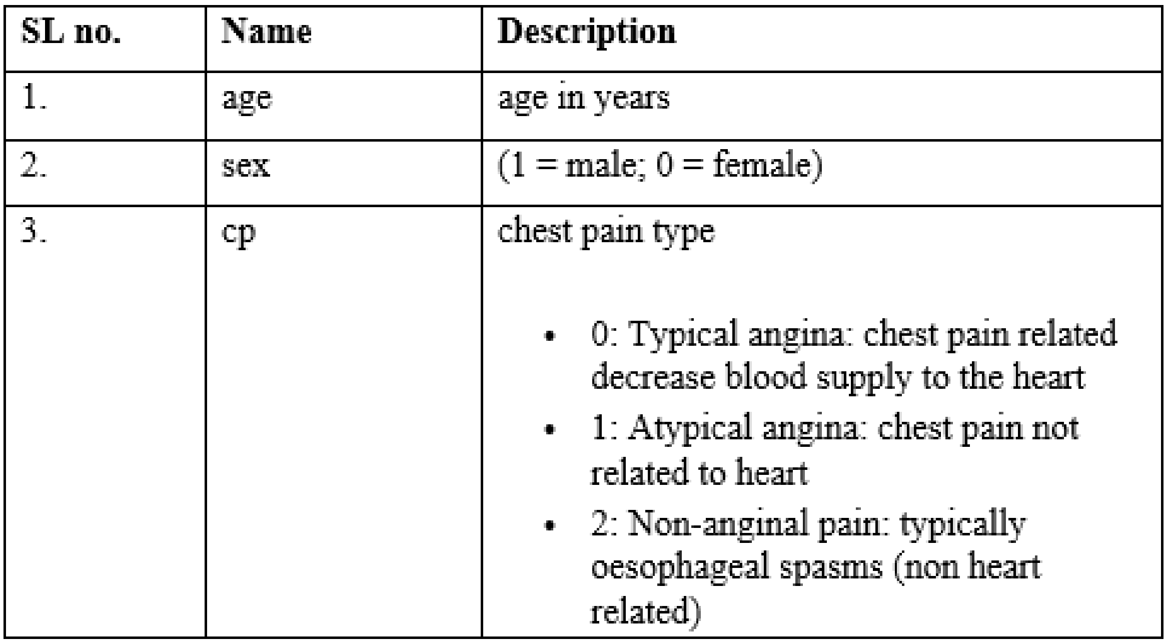
Our proposed methodology combines advanced machine learning techniques, strategic data preprocessing, and model evaluation to develop a robust predictive model for heart disease. The methodology consists of the following steps:

6.1 Data Collection and Preprocessing

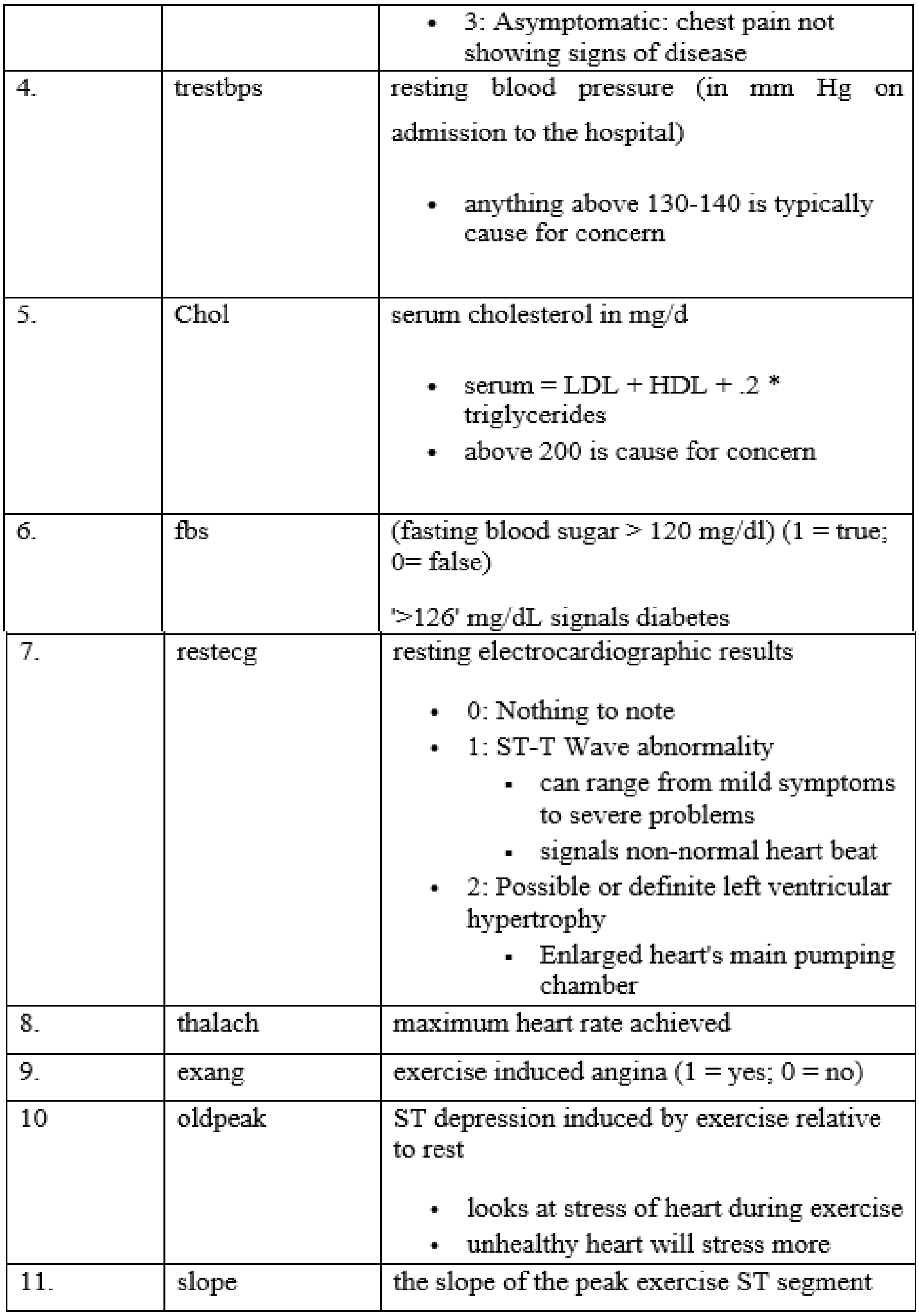
6.1.1 Data Sourcing

Obtaining a comprehensive dataset involves sourcing diverse patient information from various sources, including hospitals, research databases, or healthcare institutions. This dataset should encompass:

- Demographics: Age, gender, ethnicity, etc.
- Medical History: Pre-existing conditions (diabetes, hypertension), medication history.
- Vital Signs: Blood pressure, heart rate, BMI.
- Lab Results: Cholesterol levels, blood glucose, etc.



SL no.	Name	Description
1.	age	age in years
2.	sex	(1 = male; 0 = female)
3.	cp	chest pain type • 0: Typical angina: chest pain related decrease blood supply to the heart • 1: Atypical angina: chest pain not related to heart • 2: Non-anginal pain: typically oesophageal spasms (non heart related)



		• 3: Asymptomatic: chest pain not showing signs of disease
4.	trestbps	resting blood pressure (in mm Hg on admission to the hospital) • anything above 130-140 is typically cause for concern
5.	Chol	serum cholesterol in mg/d • serum = LDL + HDL + .2 * triglycerides • above 200 is cause for concern
6.	fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0= false) '>126' mg/dL signals diabetes
7.	restecg	resting electrocardiographic results • 0: Nothing to note • 1: ST-T Wave abnormality ▪ can range from mild symptoms to severe problems ▪ signals non-normal heart beat • 2: Possible or definite left ventricular hypertrophy ▪ Enlarged heart's main pumping chamber
8.	thalach	maximum heart rate achieved
9.	exang	exercise induced angina (1 = yes; 0 = no)
10	oldpeak	ST depression induced by exercise relative to rest • looks at stress of heart during exercise • unhealthy heart will stress more
11.	slope	the slope of the peak exercise ST segment

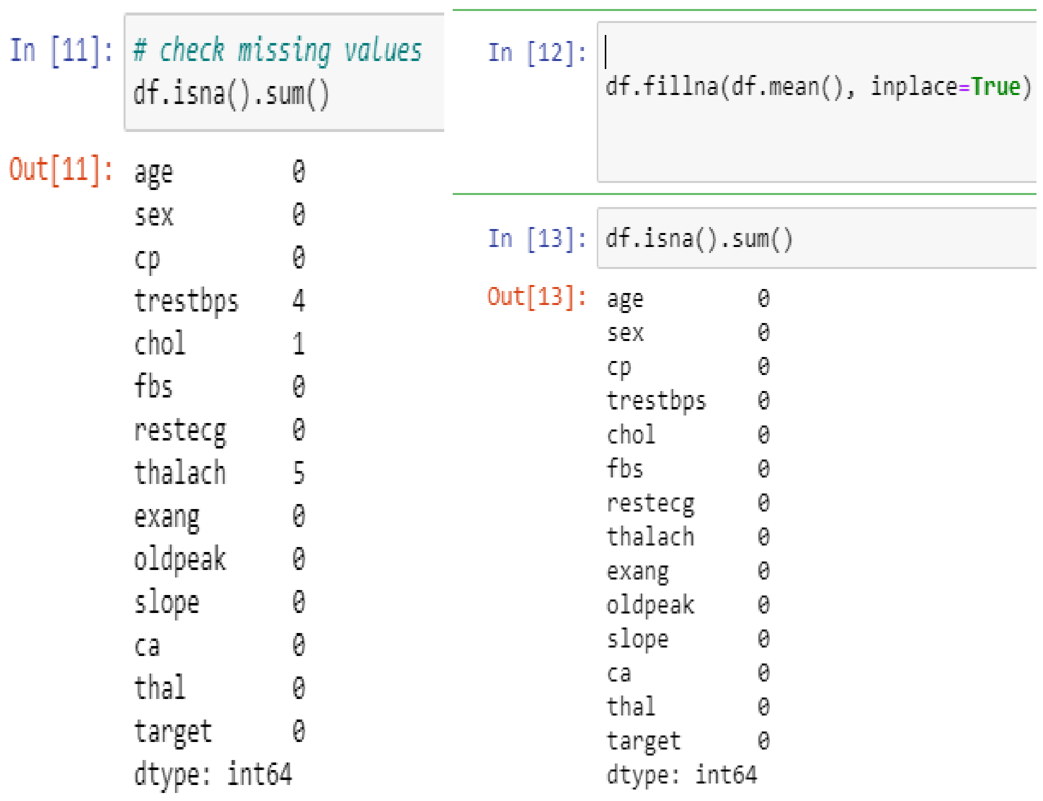
		• 0: Upsloping: better heart rate with exercise (uncommon) • 1: Flatsloping: minimal change (typical healthy heart) • 2: Downslopins: signs of unhealthy heart
12.	ca	A. number of major vessels (0-3) colored by flourosopy • colored vessel means the doctor can see the blood passing through • the more blood movement the better (no clots)
13.	Thal	thalium stress result • 1,3: normal • 6: fixed defect: used to be defect but ok now • 7: reversable defect: no proper blood movement when exercising
14.	target	have disease or not (1=yes, 0=no) (= the predicted attribute)

Table 6.1 Description of Heart Disease Dataset

6.1.2 Data Cleaning

Cleaning the dataset is essential to ensure data quality and consistency:

- Handling Missing Values: Address missing values through imputation (mean, median, mode) or deletion based on the extent of missingness.
- Outlier Treatment: Identify and handle outliers using statistical methods (e.g., Z-score, IQR) to prevent skewing of results.
- Normalization/Standardization: To improve model performance and convergence, normalise or standardise numerical features to bring them to a common scale.



```
In [11]: # check missing values
         df.isna().sum()

Out[11]: age         0
         sex         0
         cp          0
         trestbps    4
         chol        1
         fbs         0
         restecg     0
         thalach     5
         exang       0
         oldpeak     0
         slope       0
         ca          0
         thal        0
         target      0
         dtype: int64
```

```
In [12]: df.fillna(df.mean(), inplace=True)

In [13]: df.isna().sum()

Out[13]: age         0
         sex         0
         cp          0
         trestbps    0
         chol        0
         fbs         0
         restecg     0
         thalach     0
         exang       0
         oldpeak     0
         slope       0
         ca          0
         thal        0
         target      0
         dtype: int64
```

Fig. 6.1 Handling Missing Values

6.1.3 Data Split

Partitioning the dataset into test, validation, and training sets is essential for building and assessing models:

Training set: The predictive model is trained using the training set.

Validation Set: Used to evaluate model performance during training and adjust hyperparameters.

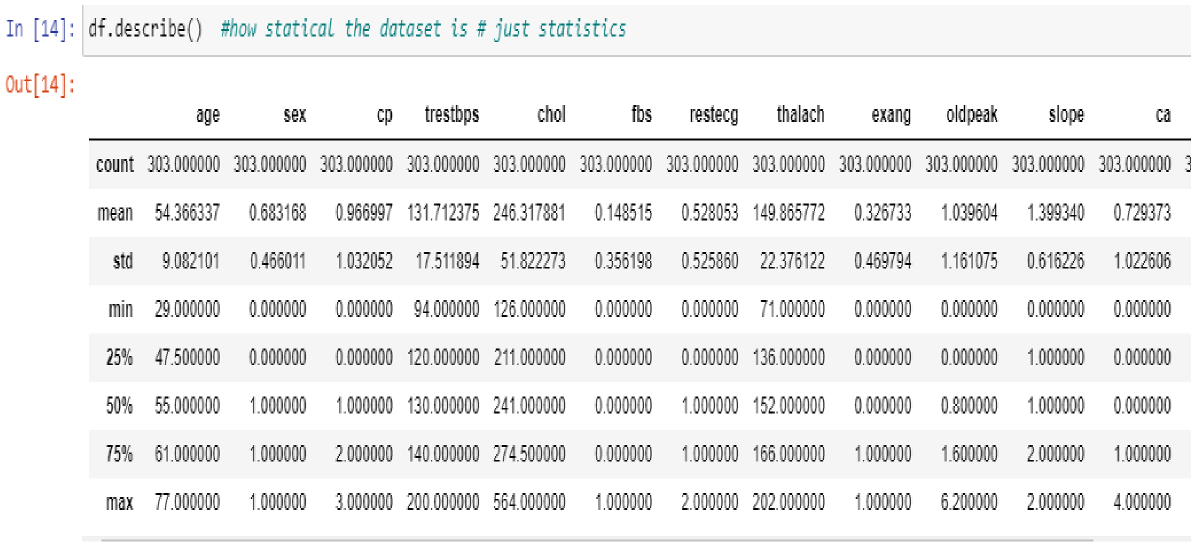
Test Set: Used to assess the performance of the finished model on unobserved data.

6.2 Exploratory Data Analysis (EDA)

6.2.1 Descriptive Analysis:

Understand the dataset's characteristics, distributions, and statistical summaries:

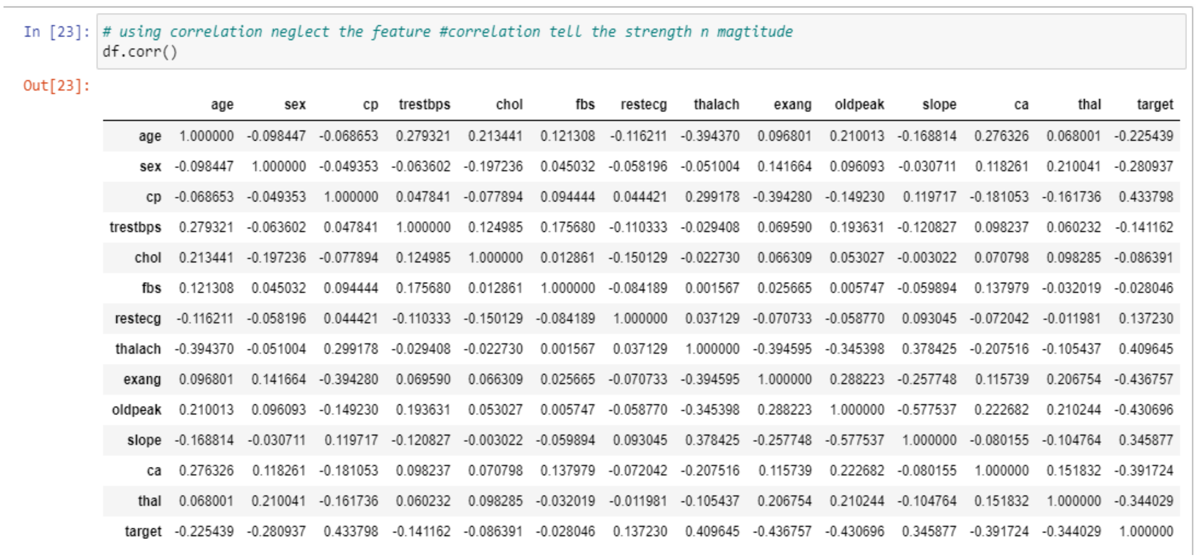
- Central Tendency: Mean, median, mode of features.
- Dispersion: Standard deviation, range, interquartile range (IQR).
- Correlation Analysis: Identify relationships between variables (e.g., correlation matrix) to understand feature importance.



```
In [14]: df.describe()  #how statical the dataset is # just statistics
Out[14]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.712375	246.317881	0.148515	0.528053	149.865772	0.326733	1.039604	1.399340	0.729373
std	9.082101	0.466011	1.032052	17.511894	51.822273	0.356198	0.525860	22.376122	0.469794	1.161075	0.616226	1.022606
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	136.000000	0.000000	0.000000	1.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	241.000000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

Fig. 6.2 Description Of The Data In The Dataframe



```
In [23]: # using correlation neglect the feature #correlation tell the strength n magtitude
         df.corr()
Out[23]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279321	0.213441	0.121308	-0.116211	-0.394370	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.063602	-0.197236	0.045032	-0.058196	-0.051004	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047841	-0.077894	0.094444	0.044421	0.299178	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279321	-0.063602	0.047841	1.000000	0.124985	0.175680	-0.110333	-0.029408	0.069590	0.193631	-0.120827	0.098237	0.060232	-0.141162
chol	0.213441	-0.197236	-0.077894	0.124985	1.000000	0.012861	-0.150129	-0.022730	0.066309	0.053027	-0.003022	0.070798	0.098285	-0.086391
fbs	0.121308	0.045032	0.094444	0.175680	0.012861	1.000000	-0.084189	0.001567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.110333	-0.150129	-0.084189	1.000000	0.037129	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.394370	-0.051004	0.299178	-0.029408	-0.022730	0.001567	0.037129	1.000000	-0.394595	-0.345398	0.378425	-0.207516	-0.105437	0.409645
exang	0.096801	0.141664	-0.394280	0.069590	0.066309	0.025665	-0.070733	-0.394595	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193631	0.053027	0.005747	-0.058770	-0.345398	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.120827	-0.003022	-0.059894	0.093045	0.378425	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.098237	0.070798	0.137979	-0.072042	-0.207516	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.060232	0.098285	-0.032019	-0.011981	-0.105437	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.141162	-0.086391	-0.028046	0.137230	0.409645	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

Fig. 6.3 Correlation Analysis

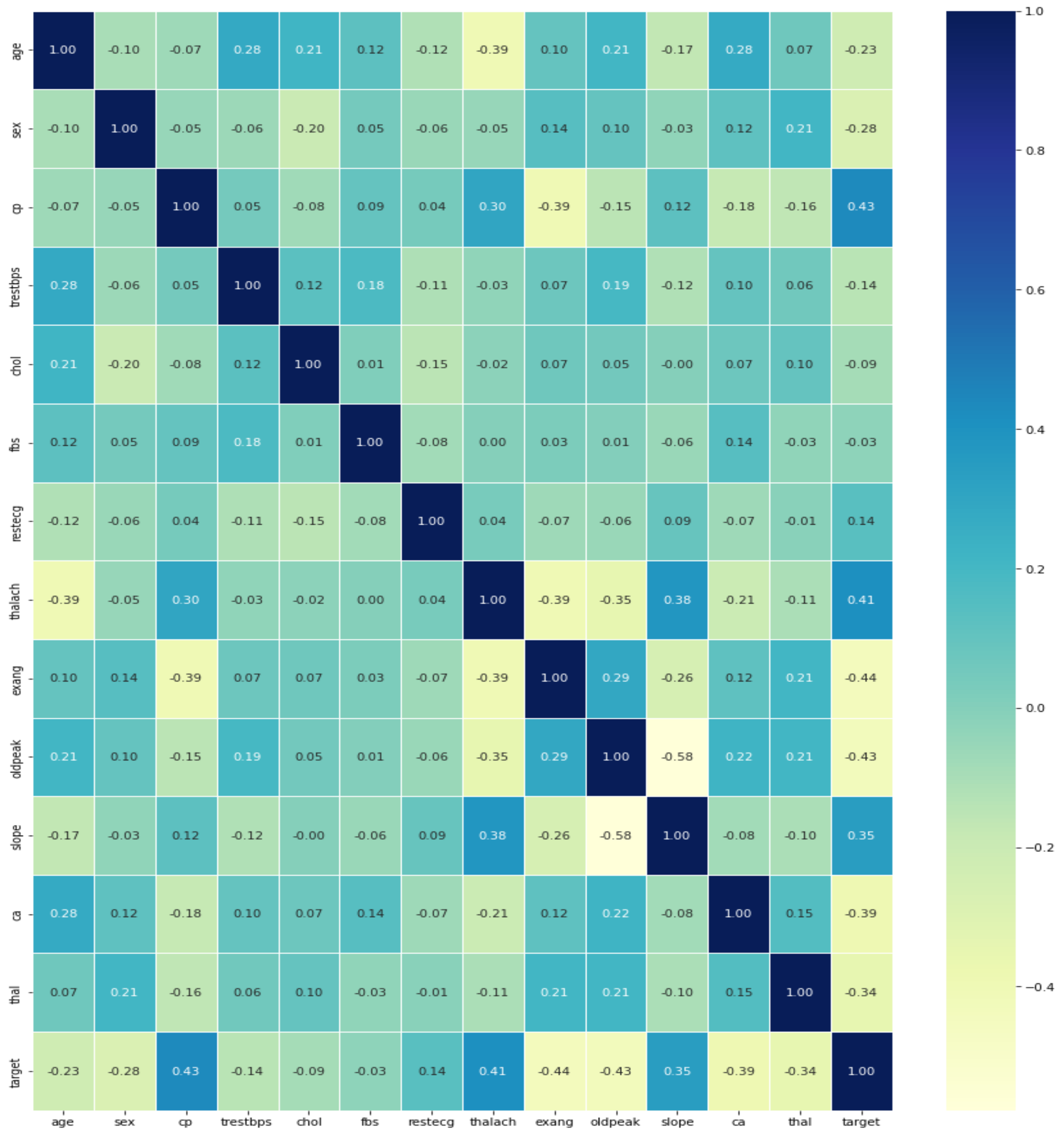
6.2.2 Visualization

Utilize visual tools to gain deeper insights and identify potential patterns related to heart disease:

Histograms: Display distributions of numerical variables.

Heatmaps: Visualize correlations between features.

Scatter Plots: Explore relationships between two numerical variables.



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.00	-0.10	-0.07	0.28	0.21	0.12	-0.12	-0.39	0.10	0.21	-0.17	0.28	0.07	-0.23
sex	-0.10	1.00	-0.05	-0.06	-0.20	0.05	-0.06	-0.05	0.14	0.10	-0.03	0.12	0.21	-0.28
cp	-0.07	-0.05	1.00	0.05	-0.08	0.09	0.04	0.30	-0.39	-0.15	0.12	-0.18	-0.16	0.43
trestbps	0.28	-0.06	0.05	1.00	0.12	0.18	-0.11	-0.03	0.07	0.19	-0.12	0.10	0.06	-0.14
chol	0.21	-0.20	-0.08	0.12	1.00	0.01	-0.15	-0.02	0.07	0.05	-0.00	0.07	0.10	-0.09
fbs	0.12	0.05	0.09	0.18	0.01	1.00	-0.08	0.00	0.03	0.01	-0.06	0.14	-0.03	-0.03
restecg	-0.12	-0.06	0.04	-0.11	-0.15	-0.08	1.00	0.04	-0.07	-0.06	0.09	-0.07	-0.01	0.14
thalach	-0.39	-0.05	0.30	-0.03	-0.02	0.00	0.04	1.00	-0.39	-0.35	0.38	-0.21	-0.11	0.41
exang	0.10	0.14	-0.39	0.07	0.07	0.03	-0.07	-0.39	1.00	0.29	-0.26	0.12	0.21	-0.44
oldpeak	0.21	0.10	-0.15	0.19	0.05	0.01	-0.06	-0.35	0.29	1.00	-0.58	0.22	0.21	-0.43
slope	-0.17	-0.03	0.12	-0.12	-0.00	-0.06	0.09	0.38	-0.26	-0.58	1.00	-0.08	-0.10	0.35
ca	0.28	0.12	-0.18	0.10	0.07	0.14	-0.07	-0.21	0.12	0.22	-0.08	1.00	0.15	-0.39
thal	0.07	0.21	-0.16	0.06	0.10	-0.03	-0.01	-0.11	0.21	0.21	-0.10	0.15	1.00	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.09	-0.03	0.14	0.41	-0.44	-0.43	0.35	-0.39	-0.34	1.00

Fig. 6.4 Correlation Matrix With Heat Map

6.3 Feature Selection and Engineering

6.3.1 The Significance of Features: Utilise methods (such as correlation matrices and statistical tests) to identify pertinent features linked to heart disease.

6.3.2 Feature Engineering: To improve the model's capacity for prediction, add new

features or modify current ones.4. Algorithm Selection:
Explore diverse machine learning algorithms suited for heart disease classification tasks.

6.4 Algorithm Selection

6.4.1. Logistic Regression

Logistic regression and other supervised machine learning algorithms can be used for both regression and classification problems. In logistic regression, probability is utilised to predict the categories into which categorical data will be divided. A sigmoid or logistic function, along with coefficient values, can be used to blend input numbers linearly in order to foresee the outcome. The most likely data are utilised to estimate maximum likelihood using the sigmoid function, and an event's probability, which ranges from 0 to 1, is provided. There is a categorising problem when the decision threshold is used. Binary (0 or 1), Multinomial (three or more classes without any ordering), and Ordinal (three or more classifications with ordering) are some of its variants. This model is simple to apply and has good predictive power. This logistic regression formula establishes the probability that input X belongs in class 1:

$$P(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)}$$

Here β_0 is bias and β_1 is the weight that is multiplied by input X .

6.4.2 K-Nearest Neighbors

A straightforward yet powerful supervised machine learning method for both regression and classification applications is the K-Nearest Neighbors (KNN) algorithm. Based on the similarity principle, a KNN finds the majority class of a sample's k nearest neighbors in the feature space to determine the class of the sample. When classifying a new data point, KNN first determines the k-nearest neighbors by calculating the distances (also known as the Euclidean distance) between the new point and every other point in the training set. A majority vote among its k neighbors determines the class for the new data point. When using regression, KNN predicts the value for the new data point by calculating the average (or weighted average) of the target values of its k nearest neighbors. In a feature space of n dimensions, the formula for the Euclidean distance between two points, p and q, is:

Euclidean distance = $\sqrt{(\sum_{i=1}^{N} (p_i - q_i)^2)}$

This distance metric helps identify the proximity of data points and forms the basis for KNN's decision-making process

6.4.3 Random Forest

Known for its excellent accuracy and resilience, Random Forest is an ensemble learning technique that is frequently used for both regression and classification applications. It is composed of many decision trees; it builds many during training and outputs the mean prediction for regression or the mode for classification, respectively. To encourage variety among the trees, each tree is constructed using a bootstrapped sample of the dataset and a random selection of the characteristics. By aggregating the predictions from these diverse trees, Random Forest mitigates overfitting and improves generalization, making it less susceptible to variance and noise in the data. Its ability to handle large datasets and capture intricate relationships between features makes it a popular choice for various machine learning applications, delivering reliable and robust predictions.

6.4.4 CatBoost

CatBoost is a powerful gradient boosting algorithm designed for handling categorical variables in machine learning tasks. Its name, "CatBoost," derives from its ability to effectively handle categorical features without the need for extensive pre-processing, reducing the risk of overfitting. Developed by Yandex, CatBoost employs an innovative method to handle categorical data, utilizing a variant of gradient boosting that integrates an advanced algorithm for handling categorical variables. It employs a symmetric tree structure and utilizes novel strategies like ordered boosting and oblivious trees to optimize model performance while minimizing overfitting. CatBoost also incorporates robust handling of missing data and provides excellent accuracy by default, requiring minimal hyperparameter tuning, making it an efficient and user-friendly choice for predictive modeling tasks, especially in scenarios with complex datasets containing categorical features.

6.4.5 XGBoost

XGBoost, short for eXtreme Gradient Boosting, is an ensemble learning method renowned for its efficiency and accuracy in handling structured/tabular data. It belongs to the gradient boosting family, building models sequentially by emphasizing the weaknesses of preceding models. XGBoost minimizes loss by optimizing a predefined objective function through the addition of weak learners (decision trees) while penalizing complex models. It employs a gradient descent algorithm, calculating gradients and using them to update model parameters. The algorithm minimizes a regularized objective comprised of the loss function and a penalty term, preventing overfitting and improving generalization. The model's final prediction is a weighted sum of predictions from all individual trees in the ensemble. The formula for XGBoost's objective function involves both a loss function (L) measuring errors and a regularization term (Ω) to control model complexity:

Objective=L(predictions , targets)+Ω(complexity)

6.4.6 Naive Bayes

The Bayes theorem is the foundation of the straightforward yet powerful Naive Bayes probabilistic classifier, which operates under the premise of predictor independence. The algorithm works by evaluating the combined probability of an instance's attributes to determine the likelihood that it belongs to a specific class. In text categorization and spam filtering, Naive Bayes frequently outperforms other models despite its simplicity and the "naive" assumption of feature independence. The formula for Naive Bayes involves Bayes' theorem, expressed as:

$P(S|R) = P(R|S)P(S) / P(R)$

Where P(S|R) is s the posterior probability of class S given predictor R.

P(R|S) is the likelihood, the probability of predictor R given class S.

P(S) is the prior probability of the class S.

P(R) is the probability of predictor R.

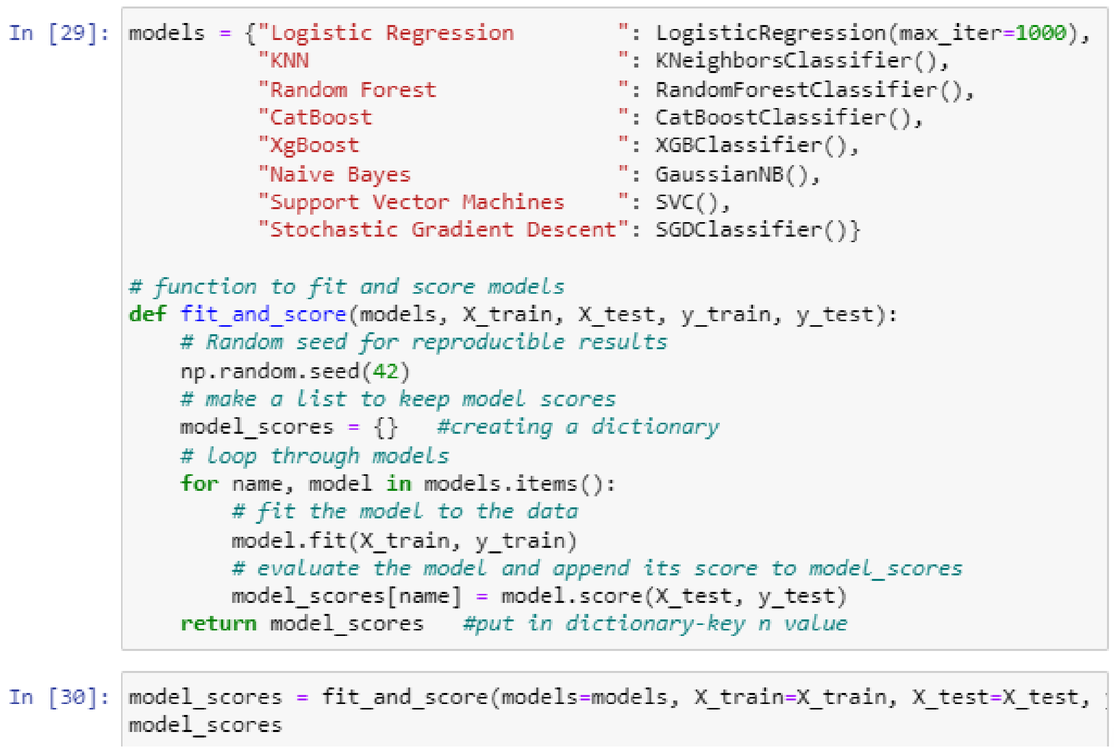
6.4.7 Stochastic Gradient Descent (SGD)

An iterative optimization technique called stochastic gradient descent (SGD) trains machine learning models, especially on big datasets, to minimize the loss function and determine the ideal parameters. SGD is computationally efficient since it changes the model's parameters using a single randomly selected data point or a tiny subset (mini-batch) of data, as opposed to traditional Gradient Descent, which uses the complete dataset for each iteration. The formula for updating the parameters in SGD involves calculating the gradient of the loss function with respect to the current parameters using the randomly chosen data point or mini-batch and adjusting the parameters in the opposite direction of the gradient to minimize the loss. The formula for updating the parameters θ at each iteration t in SGD can be represented as: $\theta t+1=\theta t-\alpha\cdot\nabla f(\theta t;xi,yi)$, where α is the learning rate, $\nabla f(\theta t;xi,yi)$ represents the gradient of the loss function f at parameters θt with respect to a randomly chosen data point (xi,yi).

6.4.8 Model Implementation

Develop and train multiple models using the selected algorithms on the training dataset:

- Model Development: Implement the selected algorithms using appropriate libraries (e.g., scikit-learn) to create predictive models.
- Training: Train each model on the training dataset using appropriate parameters.



```
In [29]: models = {"Logistic Regression          ": LogisticRegression(max_iter=1000),
                    "KNN                          ": KNeighborsClassifier(),
                    "Random Forest                ": RandomForestClassifier(),
                    "CatBoost                     ": CatBoostClassifier(),
                    "XgBoost                      ": XGBClassifier(),
                    "Naive Bayes                  ": GaussianNB(),
                    "Support Vector Machines      ": SVC(),
                    "Stochastic Gradient Descent": SGDClassifier()}

         # function to fit and score models
         def fit_and_score(models, X_train, X_test, y_train, y_test):
             # Random seed for reproducible results
             np.random.seed(42)
             # make a list to keep model scores
             model_scores = {}   #creating a dictionary
             # loop through models
             for name, model in models.items():
                 # fit the model to the data
                 model.fit(X_train, y_train)
                 # evaluate the model and append its score to model_scores
                 model_scores[name] = model.score(X_test, y_test)
             return model_scores   #put in dictionary-key n value

In [30]: model_scores = fit_and_score(models=models, X_train=X_train, X_test=X_test,
         model_scores
```

Fig. 6.5 Code Snippet of Model Implementation

6.5 Model Evaluation

A crucial first step in evaluating the efficacy, precision, and resilience of machine learning models for heart disease prediction is model evaluation. A few crucial elements of model evaluation are as follows:

Accuracy:

Formula: (TP + TN) / (TP + TN + FP + FN)
Measures the proportion of correct predictions out of the total predictions made.
In heart disease prediction, it reflects the overall correctness of identifying both healthy individuals and those with heart disease.

Precision:

Formula: TP / (TP + FP)
Indicates the accuracy of positive predictions. In heart disease prediction, it measures the proportion of correctly identified individuals with heart disease among all predicted positive cases.
High precision means fewer false positives, reducing unnecessary interventions or treatments for individuals who are actually healthy.

Recall (Sensitivity):

Formula: TP / (TP + FN)
Measures the model's ability to identify all positive instances correctly. In heart disease prediction, it quantifies the proportion of correctly identified individuals with heart disease among all actual positive cases.
High recall signifies the model's effectiveness in capturing individuals with heart disease, reducing the chance of missing those who require intervention.

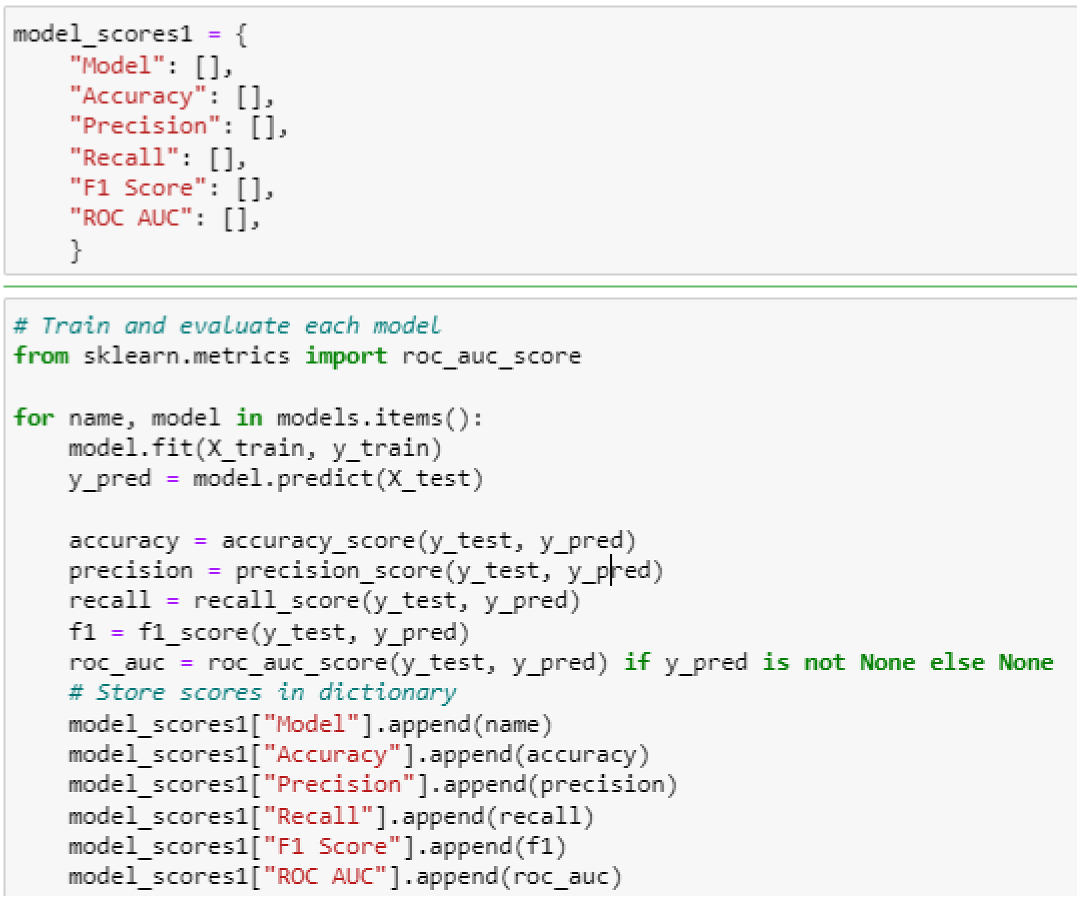
F1-Score:

Formula: 2 * (Precision * Recall) / (Precision + Recall)

The harmonic mean of precision and recall. It represents a balanced measure between precision and recall.
Useful when there's an uneven class distribution (e.g., imbalanced datasets) in heart disease prediction.

ROC-AUC (Receiver Operating Characteristic - Area Under Curve)

ROC Curve plots True Positive Rate (Sensitivity) against False Positive Rate (1 - Specificity). AUC calculates the area under the ROC curve, indicating the model's ability to distinguish between classes (heart disease vs. no heart disease).
In heart disease prediction, a higher AUC suggests better discrimination between individuals with and without heart disease.

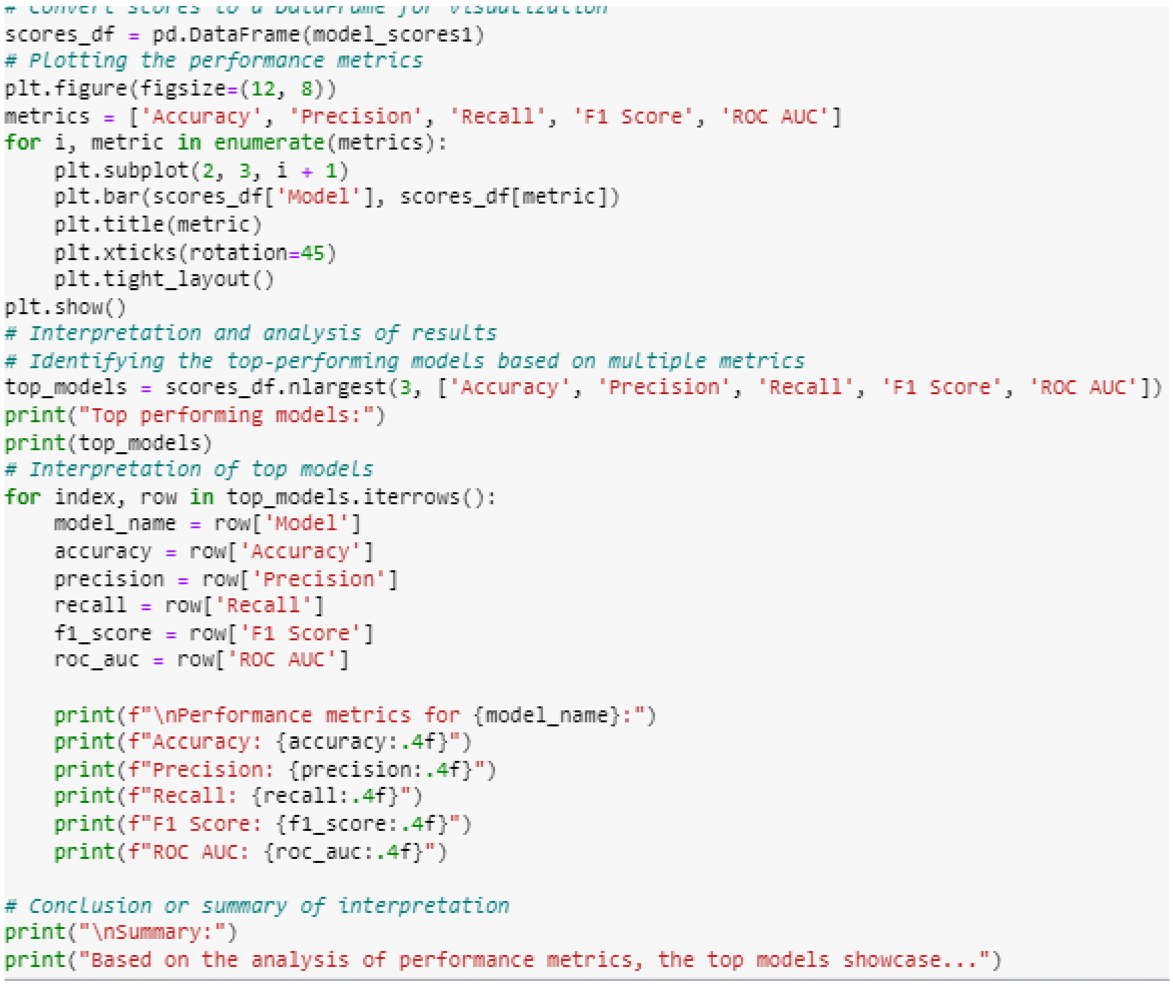


```
model_scores1 = {
    "Model": [],
    "Accuracy": [],
    "Precision": [],
    "Recall": [],
    "F1 Score": [],
    "ROC AUC": [],
    }
```

```
# Train and evaluate each model
from sklearn.metrics import roc_auc_score

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred) if y_pred is not None else None
    # Store scores in dictionary
    model_scores1["Model"].append(name)
    model_scores1["Accuracy"].append(accuracy)
    model_scores1["Precision"].append(precision)
    model_scores1["Recall"].append(recall)
    model_scores1["F1 Score"].append(f1)
    model_scores1["ROC AUC"].append(roc_auc)
```

Fig. 6.6 Code Snippet of Model Evaluation

This code initializes an empty dictionary model_scores1 to store evaluation metrics for various machine learning models. It then iterates through a dictionary of models, where each model is trained on X_train and y_train data and evaluated on X_test and y_test. For each model, it calculates evaluation metrics such as accuracy, precision, recall, F1 score, and ROC AUC using the respective functions from scikit-learn. These metrics are then appended to the model_scores1 dictionary along with the model's name. This process creates a structured collection of evaluation scores for each model, allowing easy comparison of their performance.



```
# Convert scores to a DataFrame for visualization
scores_df = pd.DataFrame(model_scores1)
# Plotting the performance metrics
plt.figure(figsize=(12, 8))
metrics = ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC']
for i, metric in enumerate(metrics):
    plt.subplot(2, 3, i + 1)
    plt.bar(scores_df['Model'], scores_df[metric])
    plt.title(metric)
    plt.xticks(rotation=45)
    plt.tight_layout()
plt.show()
# Interpretation and analysis of results
# Identifying the top-performing models based on multiple metrics
top_models = scores_df.nlargest(3, ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC'])
print("Top performing models:")
print(top_models)
# Interpretation of top models
for index, row in top_models.iterrows():
    model_name = row['Model']
    accuracy = row['Accuracy']
    precision = row['Precision']
    recall = row['Recall']
    f1_score = row['F1 Score']
    roc_auc = row['ROC AUC']

    print(f"\nPerformance metrics for {model_name}:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1_score:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}")

# Conclusion or summary of interpretation
print("\nSummary:")
print("Based on the analysis of performance metrics, the top models showcase...")
```

Fig. 6.7 Code Snippet of Model Evaluation Visualization

This code snippet employs Python libraries like matplotlib, pandas, and scikit-learn to visualize and analyze the performance metrics of multiple machine learning models for classification tasks. Initially, it imports necessary modules for plotting, data manipulation, and model evaluation. Assuming the existence of a populated DataFrame model_scores1 containing model performance metrics (Accuracy, Precision, Recall, F1 Score, ROC AUC) for various models, it converts this data into a pandas DataFrame scores_df. The subsequent section uses matplotlib to create a 2x3 subplot grid, plotting bar graphs for each metric (Accuracy, Precision, Recall, F1 Score, ROC AUC) against different model names on separate subplots, enabling visual comparison of model performances. It then identifies and prints the top-performing models based on each metric and displays their individual performance metrics like accuracy, precision, recall, F1 score, and ROC AUC. The code concludes by summarizing the overall analysis of top models' performances, aiming to provide insights into the most effective models for the classification task at hand. The code's layout enables a comprehensive analysis and comparison of multiple models' performances, aiding in model selection and decision-making processes based on key evaluation metrics.

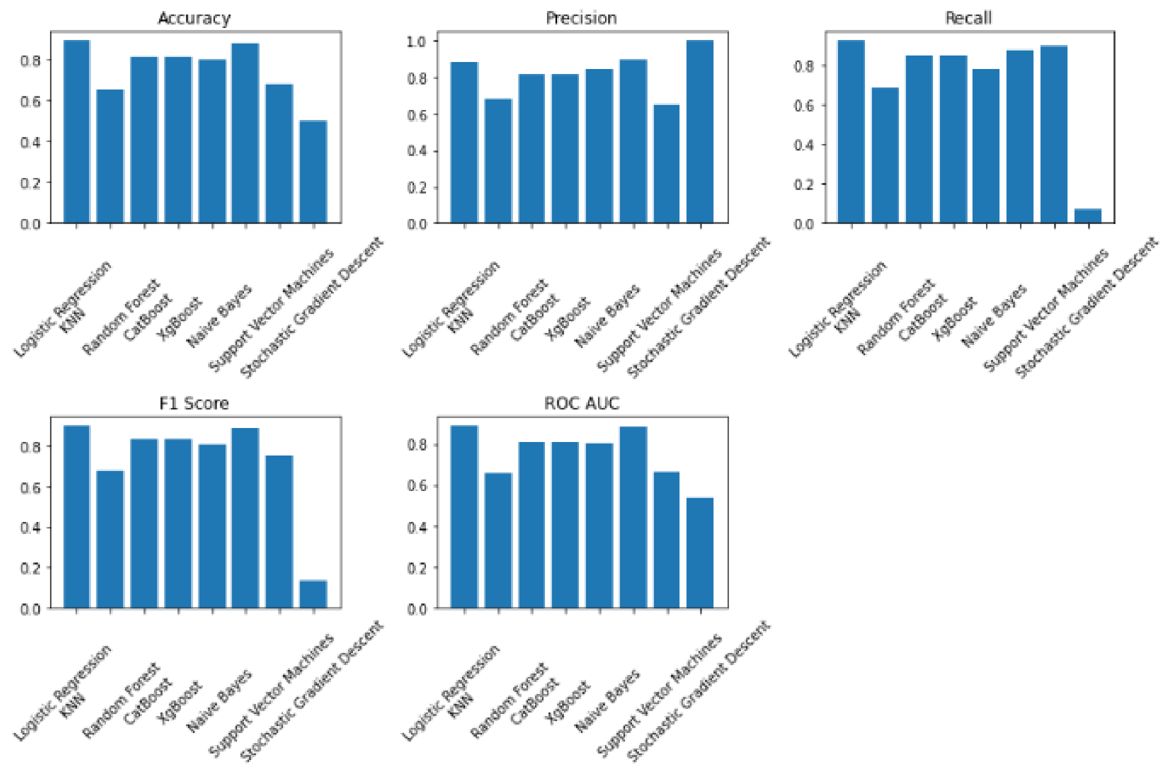
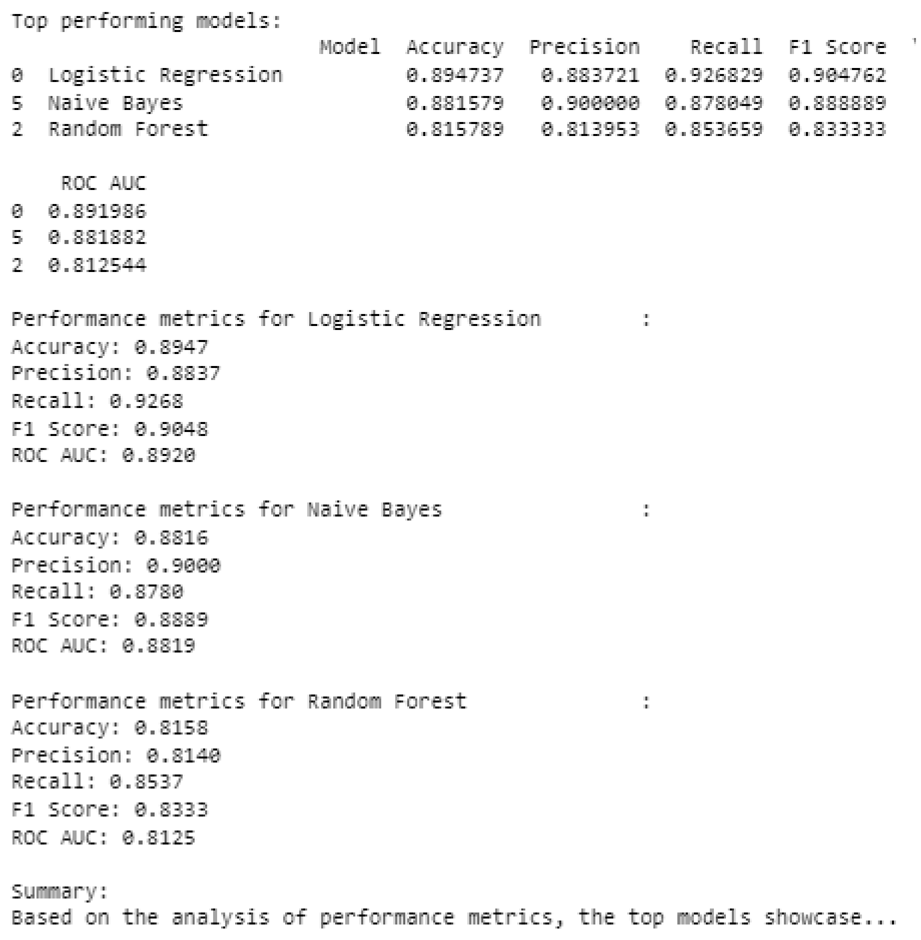


Fig. 6.8 Model Evaluation Result 1



```
Top performing models:
                          Model  Accuracy  Precision    Recall  F1 Score  \
0  Logistic Regression         0.894737   0.883721  0.926829  0.904762
5  Naive Bayes                 0.881579   0.900000  0.878049  0.888889
2  Random Forest               0.815789   0.813953  0.853659  0.833333

    ROC AUC
0  0.891986
5  0.881882
2  0.812544

Performance metrics for Logistic Regression        :
Accuracy: 0.8947
Precision: 0.8837
Recall: 0.9268
F1 Score: 0.9048
ROC AUC: 0.8920

Performance metrics for Naive Bayes                :
Accuracy: 0.8816
Precision: 0.9000
Recall: 0.8780
F1 Score: 0.8889
ROC AUC: 0.8819

Performance metrics for Random Forest              :
Accuracy: 0.8158
Precision: 0.8140
Recall: 0.8537
F1 Score: 0.8333
ROC AUC: 0.8125

Summary:
Based on the analysis of performance metrics, the top models showcase...
```

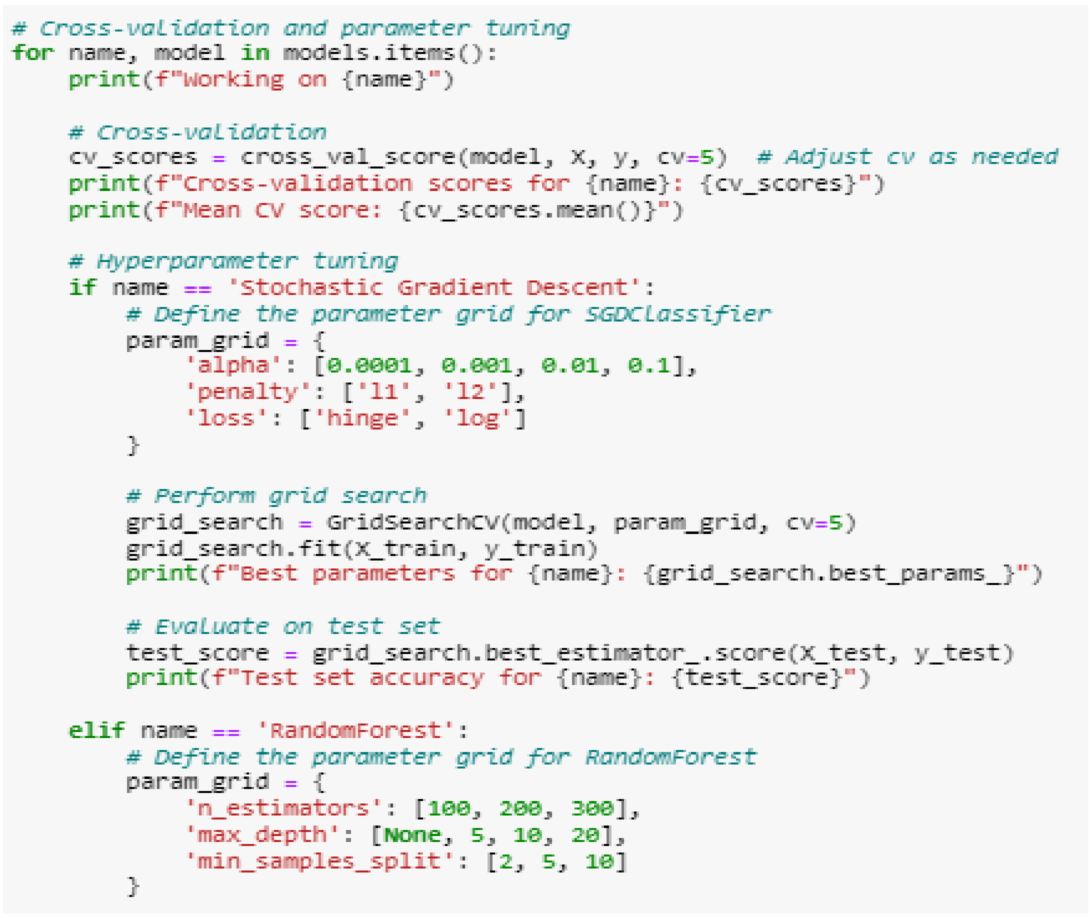
Fig. 6.9 Model Evaluation Result 2

6.6 Cross-Validation and Parameter Tuning

A key method in machine learning for evaluating the performance and generalizability of models is cross-validation. The process entails splitting the dataset into several subsets, or folds, from which the model is iteratively validated on the remaining folds after being trained on a subset of the data. One popular technique is k-fold cross-validation, in which the dataset is split into k subsets. The model is trained on the remaining k-1 folds while each fold is used once as a validation set. To make sure that every data point appears in the validation set precisely once, this process is done k times. Cross-validation reduces biases that can arise from a single train-test split, resulting in a more reliable prediction of a model's performance.By evaluating a model's performance on hypothetical data, we can detect possible problems such as overfitting or underfitting and adjust parameters to improve the model's accuracy and generalization.enhance model accuracy and generalization.

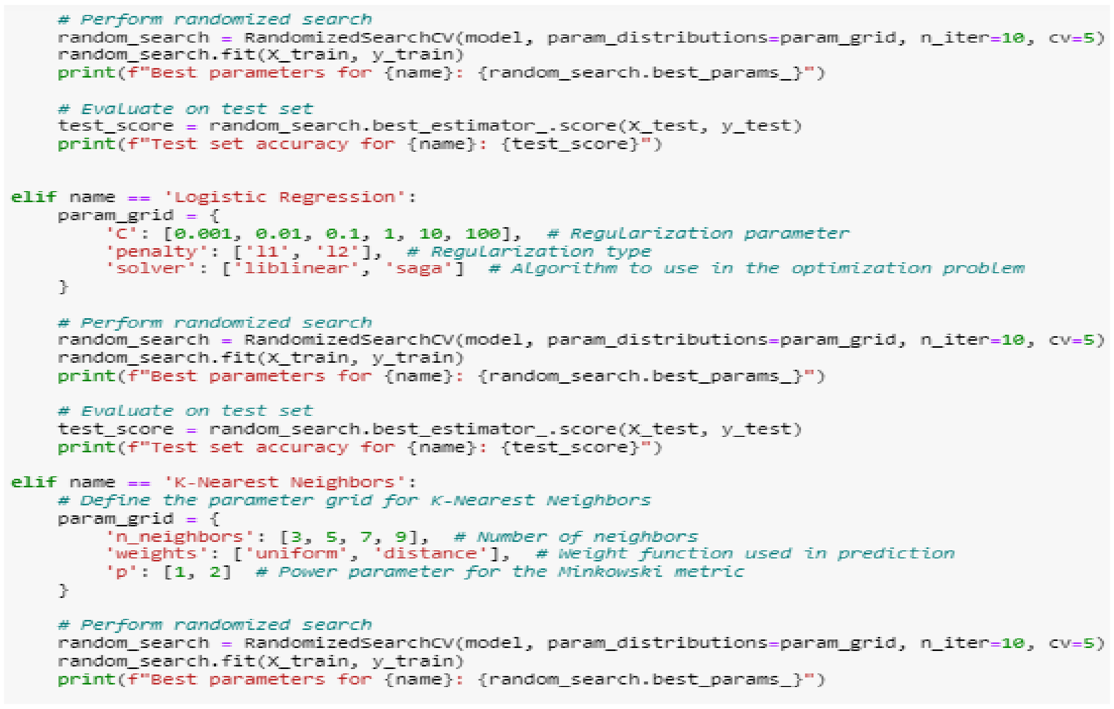
Parameter tuning, often referred to as hyperparameter optimization, is the process of selecting the best set of hyperparameters for a machine learning algorithm. Hyperparameters are configurations external to the model that control its learning process, such as the learning rate in gradient descent or the depth of a decision tree. Grid search and randomized search are common techniques employed for parameter tuning. Grid search exhaustively evaluates all combinations of specified hyperparameter values, while randomized search randomly selects combinations from predefined ranges. By fine-tuning these hyperparameters through cross-validation, models can achieve better performance, improving metrics like accuracy, precision, and recall. The goal is to find the optimal set of hyperparameters that maximizes a model's predictive power and generalizability on unseen data, enhancing its effectiveness in real-world applications.

Fig. 6.10, 6.11, 6.12 depicts the code cross-validation and hyper parameter tuning. This Python code snippet performs model building, cross-validation, and hyperparameter tuning for various machine learning models using Scikit-learn and CatBoost libraries. It first generates a sample dataset, preprocesses it using StandardScaler, and splits it into training and test sets. The code iterates through different models, conducts cross-validation to assess model performance, and tunes hyperparameters to optimize each model's accuracy on the test set. It utilizes GridSearchCV or RandomizedSearchCV to search for the best hyperparameters for each model type, such as Logistic Regression, K-Nearest Neighbors, Naive Bayes, Support Vector Machines, XGBoost, and CatBoost, helping to enhance their predictive capabilities. Finally, it prints the best hyperparameters found for each model and their respective accuracy scores on the test set.



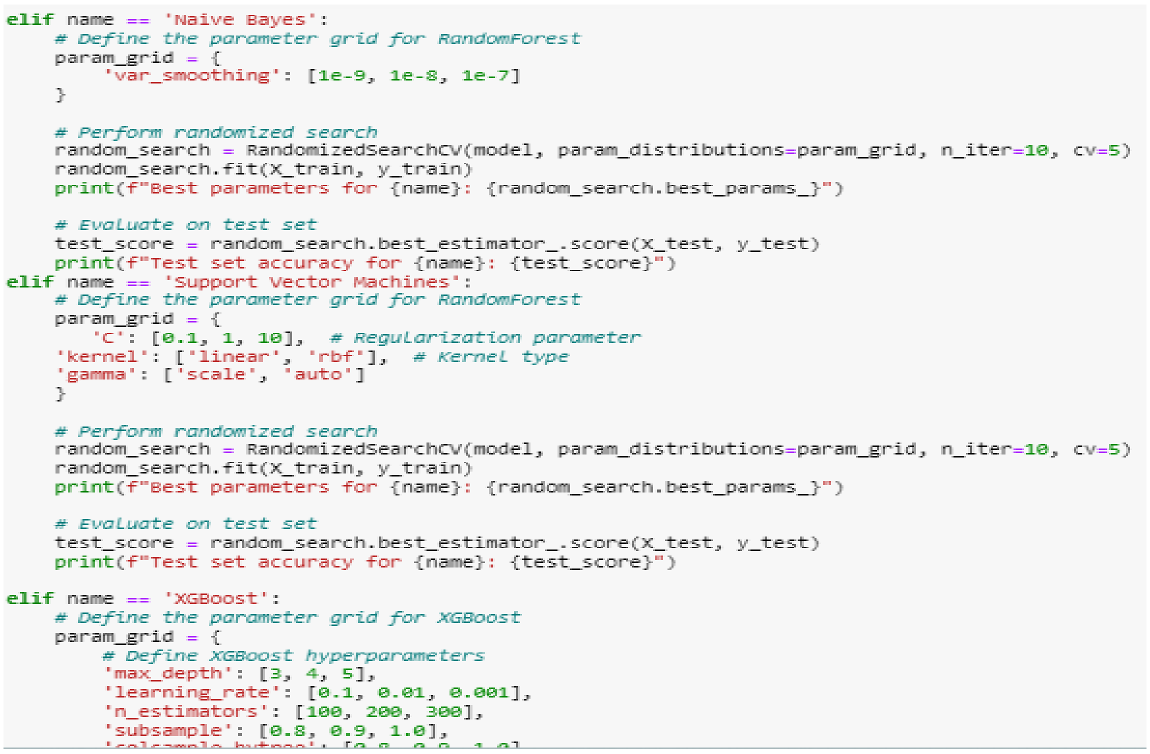
```
# Cross-validation and parameter tuning
for name, model in models.items():
    print(f"Working on {name}")

    # Cross-validation
    cv_scores = cross_val_score(model, X, y, cv=5)  # Adjust cv as needed
    print(f"Cross-validation scores for {name}: {cv_scores}")
    print(f"Mean CV score: {cv_scores.mean()}")

    # Hyperparameter tuning
    if name == 'Stochastic Gradient Descent':
        # Define the parameter grid for SGDClassifier
        param_grid = {
            'alpha': [0.0001, 0.001, 0.01, 0.1],
            'penalty': ['l1', 'l2'],
            'loss': ['hinge', 'log']
        }

        # Perform grid search
        grid_search = GridSearchCV(model, param_grid, cv=5)
        grid_search.fit(X_train, y_train)
        print(f"Best parameters for {name}: {grid_search.best_params_}")

        # Evaluate on test set
        test_score = grid_search.best_estimator_.score(X_test, y_test)
        print(f"Test set accuracy for {name}: {test_score}")

    elif name == 'RandomForest':
        # Define the parameter grid for RandomForest
        param_grid = {
            'n_estimators': [100, 200, 300],
            'max_depth': [None, 5, 10, 20],
            'min_samples_split': [2, 5, 10]
        }
```

Fig. 6.10 Code Snippet of Cross-Validation and Hyper Parameter Tuning On SGD And Random



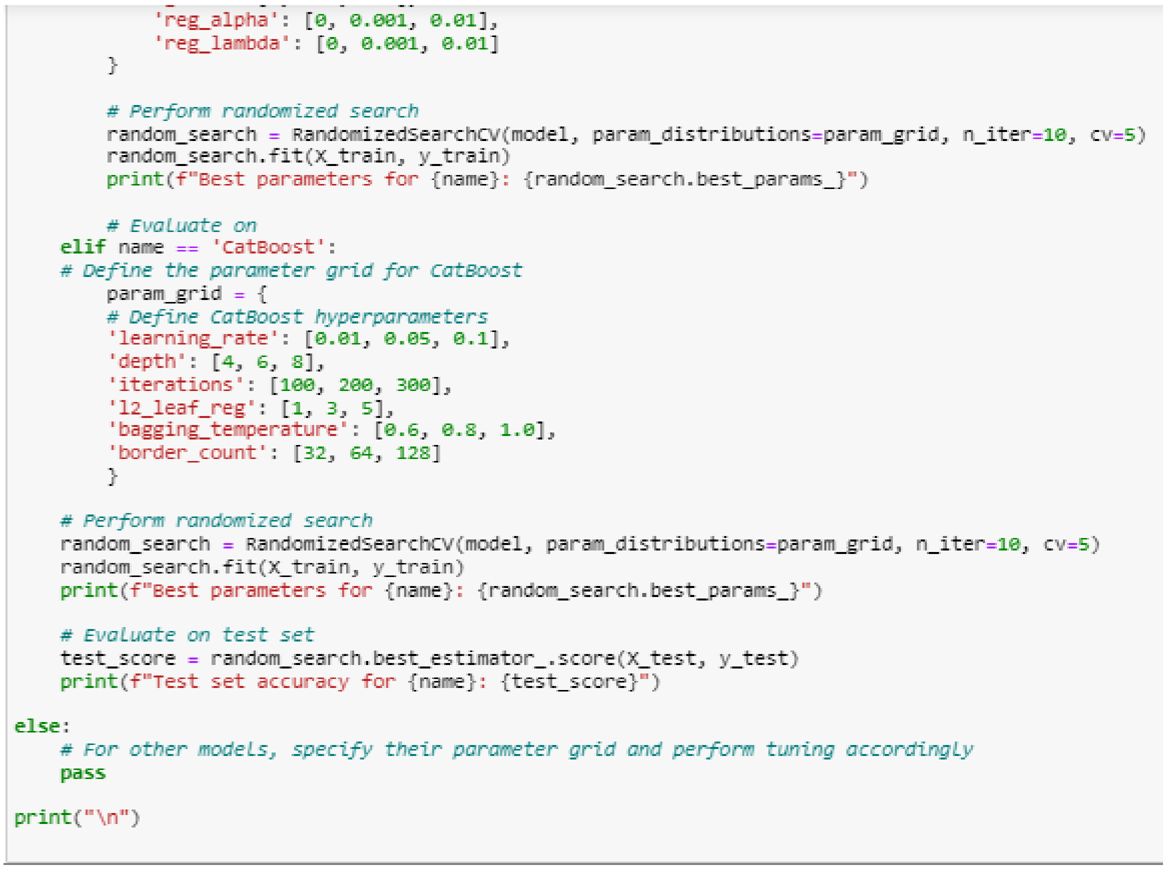
```
    # Perform randomized search
    random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
    random_search.fit(X_train, y_train)
    print(f"Best parameters for {name}: {random_search.best_params_}")

    # Evaluate on test set
    test_score = random_search.best_estimator_.score(X_test, y_test)
    print(f"Test set accuracy for {name}: {test_score}")


elif name == 'Logistic Regression':
    param_grid = {
        'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization parameter
        'penalty': ['l1', 'l2'],  # Regularization type
        'solver': ['liblinear', 'saga']  # Algorithm to use in the optimization problem
    }

    # Perform randomized search
    random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
    random_search.fit(X_train, y_train)
    print(f"Best parameters for {name}: {random_search.best_params_}")

    # Evaluate on test set
    test_score = random_search.best_estimator_.score(X_test, y_test)
    print(f"Test set accuracy for {name}: {test_score}")

elif name == 'K-Nearest Neighbors':
    # Define the parameter grid for K-Nearest Neighbors
    param_grid = {
        'n_neighbors': [3, 5, 7, 9],  # Number of neighbors
        'weights': ['uniform', 'distance'],  # Weight function used in prediction
        'p': [1, 2]  # Power parameter for the Minkowski metric
    }

    # Perform randomized search
    random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
    random_search.fit(X_train, y_train)
    print(f"Best parameters for {name}: {random_search.best_params_}")
```

Fig. 6.11 Code Snippet of Hyper Parameter Tuning on Logistic Regression And K–Nearest Neighbors



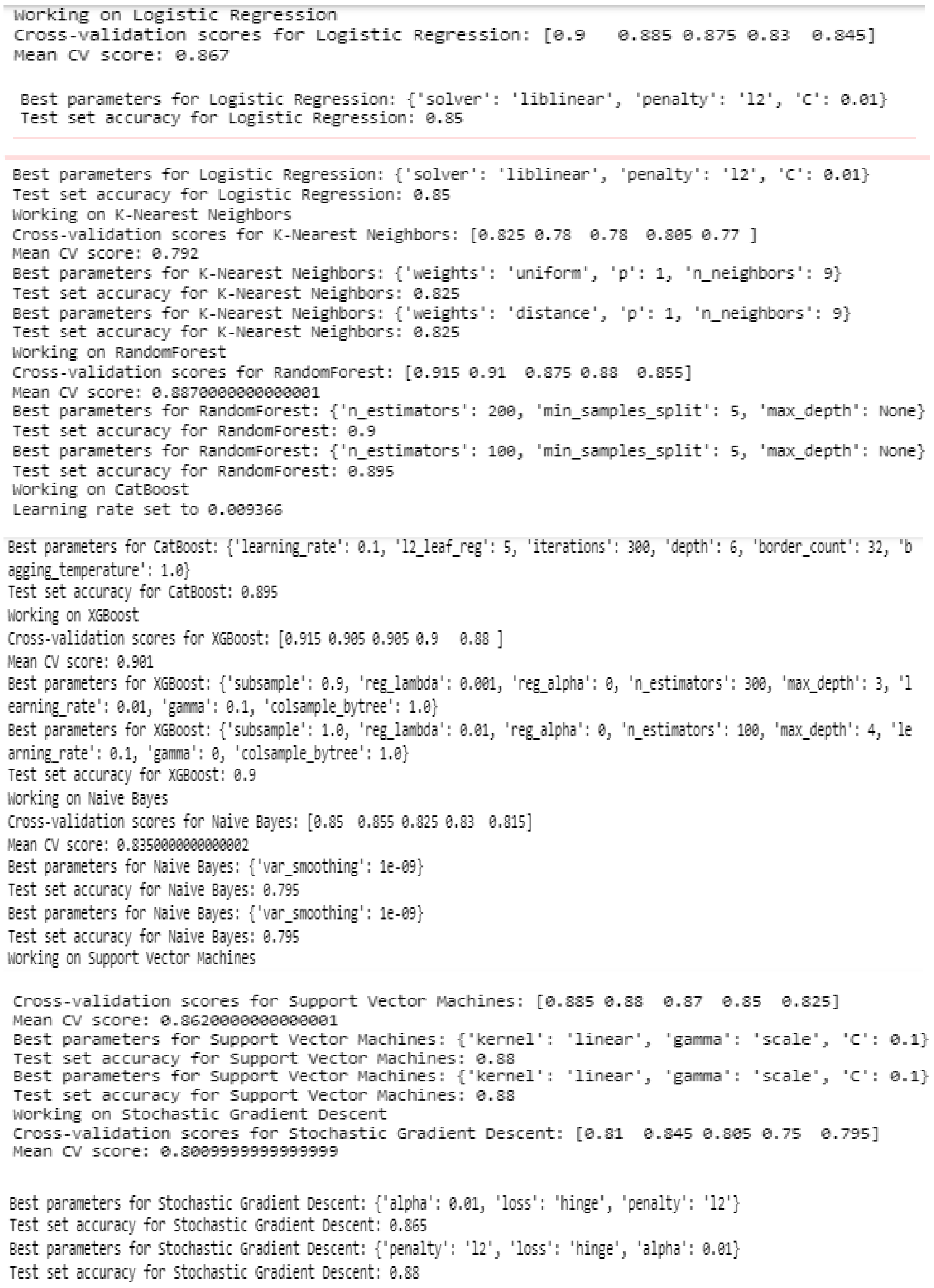
```
elif name == 'Naive Bayes':
    # Define the parameter grid for RandomForest
    param_grid = {
        'var_smoothing': [1e-9, 1e-8, 1e-7]
    }

    # Perform randomized search
    random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
    random_search.fit(X_train, y_train)
    print(f"Best parameters for {name}: {random_search.best_params_}")

    # Evaluate on test set
    test_score = random_search.best_estimator_.score(X_test, y_test)
    print(f"Test set accuracy for {name}: {test_score}")
elif name == 'Support Vector Machines':
    # Define the parameter grid for RandomForest
    param_grid = {
       'C': [0.1, 1, 10],  # Regularization parameter
    'kernel': ['linear', 'rbf'],  # Kernel type
    'gamma': ['scale', 'auto']
    }

    # Perform randomized search
    random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
    random_search.fit(X_train, y_train)
    print(f"Best parameters for {name}: {random_search.best_params_}")

    # Evaluate on test set
    test_score = random_search.best_estimator_.score(X_test, y_test)
    print(f"Test set accuracy for {name}: {test_score}")

elif name == 'XGBoost':
    # Define the parameter grid for XGBoost
    param_grid = {
        # Define XGBoost hyperparameters
        'max_depth': [3, 4, 5],
        'learning_rate': [0.1, 0.01, 0.001],
        'n_estimators': [100, 200, 300],
        'subsample': [0.8, 0.9, 1.0],
        'colsample_bytree': [0.8, 0.9, 1.0]
```

Fig. 6.12 Code Snippet of Hyper Parameter Tuning on Naïve Bayes, SVM And XGBoost



```
            'reg_alpha': [0, 0.001, 0.01],
            'reg_lambda': [0, 0.001, 0.01]
        }

        # Perform randomized search
        random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
        random_search.fit(X_train, y_train)
        print(f"Best parameters for {name}: {random_search.best_params_}")

        # Evaluate on
    elif name == 'CatBoost':
    # Define the parameter grid for CatBoost
        param_grid = {
        # Define CatBoost hyperparameters
        'learning_rate': [0.01, 0.05, 0.1],
        'depth': [4, 6, 8],
        'iterations': [100, 200, 300],
        'l2_leaf_reg': [1, 3, 5],
        'bagging_temperature': [0.6, 0.8, 1.0],
        'border_count': [32, 64, 128]
        }

    # Perform randomized search
    random_search = RandomizedSearchCV(model, param_distributions=param_grid, n_iter=10, cv=5)
    random_search.fit(X_train, y_train)
    print(f"Best parameters for {name}: {random_search.best_params_}")

    # Evaluate on test set
    test_score = random_search.best_estimator_.score(X_test, y_test)
    print(f"Test set accuracy for {name}: {test_score}")

else:
    # For other models, specify their parameter grid and perform tuning accordingly
    pass

print("\n")
```

Fig. 6.13 Code Snippet of Hyper Parameter Tuning On Catboost



```
Working on Logistic Regression
Cross-validation scores for Logistic Regression: [0.9   0.885 0.875 0.83  0.845]
Mean CV score: 0.867

 Best parameters for Logistic Regression: {'solver': 'liblinear', 'penalty': 'l2', 'C': 0.01}
 Test set accuracy for Logistic Regression: 0.85
```

```
Best parameters for Logistic Regression: {'solver': 'liblinear', 'penalty': 'l2', 'C': 0.01}
Test set accuracy for Logistic Regression: 0.85
Working on K-Nearest Neighbors
Cross-validation scores for K-Nearest Neighbors: [0.825 0.78  0.78  0.805 0.77 ]
Mean CV score: 0.792
Best parameters for K-Nearest Neighbors: {'weights': 'uniform', 'p': 1, 'n_neighbors': 9}
Test set accuracy for K-Nearest Neighbors: 0.825
Best parameters for K-Nearest Neighbors: {'weights': 'distance', 'p': 1, 'n_neighbors': 9}
Test set accuracy for K-Nearest Neighbors: 0.825
Working on RandomForest
Cross-validation scores for RandomForest: [0.915 0.91  0.875 0.88  0.855]
Mean CV score: 0.8870000000000001
Best parameters for RandomForest: {'n_estimators': 200, 'min_samples_split': 5, 'max_depth': None}
Test set accuracy for RandomForest: 0.9
Best parameters for RandomForest: {'n_estimators': 100, 'min_samples_split': 5, 'max_depth': None}
Test set accuracy for RandomForest: 0.895
Working on CatBoost
Learning rate set to 0.009366
```

```
Best parameters for CatBoost: {'learning_rate': 0.1, 'l2_leaf_reg': 5, 'iterations': 300, 'depth': 6, 'border_count': 32, 'b
agging_temperature': 1.0}
Test set accuracy for CatBoost: 0.895
Working on XGBoost
Cross-validation scores for XGBoost: [0.915 0.905 0.905 0.9   0.88 ]
Mean CV score: 0.901
Best parameters for XGBoost: {'subsample': 0.9, 'reg_lambda': 0.001, 'reg_alpha': 0, 'n_estimators': 300, 'max_depth': 3, 'l
earning_rate': 0.01, 'gamma': 0.1, 'colsample_bytree': 1.0}
Best parameters for XGBoost: {'subsample': 1.0, 'reg_lambda': 0.01, 'reg_alpha': 0, 'n_estimators': 100, 'max_depth': 4, 'le
arning_rate': 0.1, 'gamma': 0, 'colsample_bytree': 1.0}
Test set accuracy for XGBoost: 0.9
Working on Naive Bayes
Cross-validation scores for Naive Bayes: [0.85  0.855 0.825 0.83  0.815]
Mean CV score: 0.8350000000000002
Best parameters for Naive Bayes: {'var_smoothing': 1e-09}
Test set accuracy for Naive Bayes: 0.795
Best parameters for Naive Bayes: {'var_smoothing': 1e-09}
Test set accuracy for Naive Bayes: 0.795
Working on Support Vector Machines
```

```
Cross-validation scores for Support Vector Machines: [0.885 0.88  0.87  0.85  0.825]
Mean CV score: 0.8620000000000001
Best parameters for Support Vector Machines: {'kernel': 'linear', 'gamma': 'scale', 'C': 0.1}
Test set accuracy for Support Vector Machines: 0.88
Best parameters for Support Vector Machines: {'kernel': 'linear', 'gamma': 'scale', 'C': 0.1}
Test set accuracy for Support Vector Machines: 0.88
Working on Stochastic Gradient Descent
Cross-validation scores for Stochastic Gradient Descent: [0.81  0.845 0.805 0.75  0.795]
Mean CV score: 0.8009999999999999
```

```
Best parameters for Stochastic Gradient Descent: {'alpha': 0.01, 'loss': 'hinge', 'penalty': 'l2'}
Test set accuracy for Stochastic Gradient Descent: 0.865
Best parameters for Stochastic Gradient Descent: {'penalty': 'l2', 'loss': 'hinge', 'alpha': 0.01}
Test set accuracy for Stochastic Gradient Descent: 0.88
```

Fig. 6.14 Cross-Validation and Hyper Parameter Tuning Result

6.7 Re- model Evaluation After Cross-Validation and Hyper parameter tuning

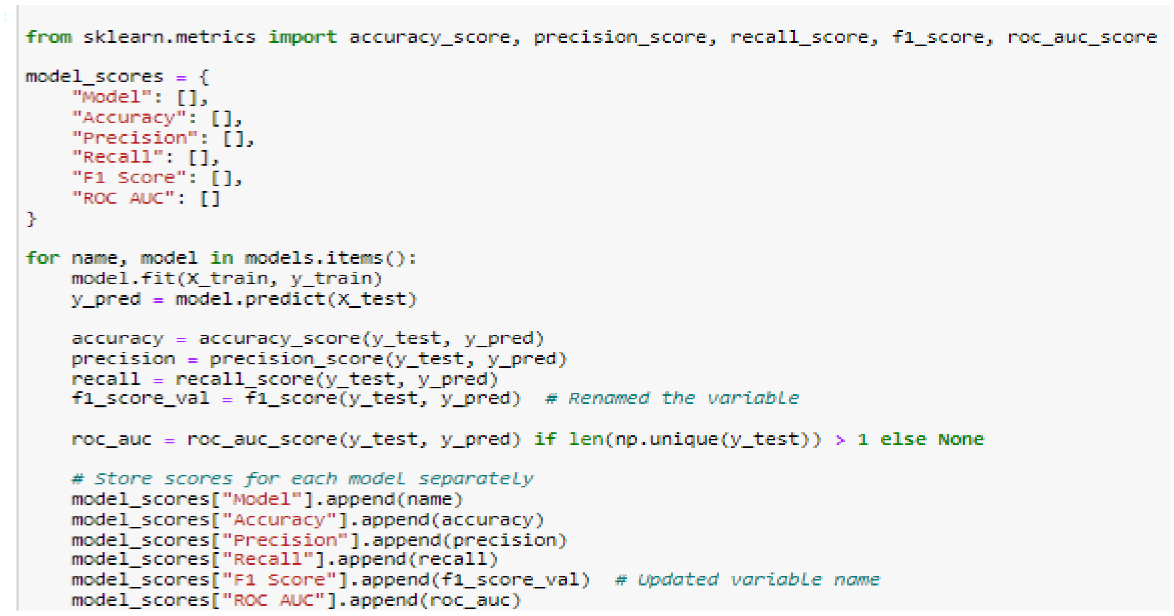
We are evaluating the model performance on the test set using various metrics like accuracy, precision, recall, F1 score, and ROC AUC again after cross-validation and hyper parameter tuning. The reason why we performing model evaluation gain after cross validation are as follows:

Performance Evaluation on Test Set: The initial assessment you performed before any tuning provides a baseline. However, after cross-validation and hyperparameter tuning, the model might have changed significantly. Hence, it's essential to evaluate the tuned models on an unseen dataset (the test set) to get a realistic estimate of how well our models generalize to new, unseen data.

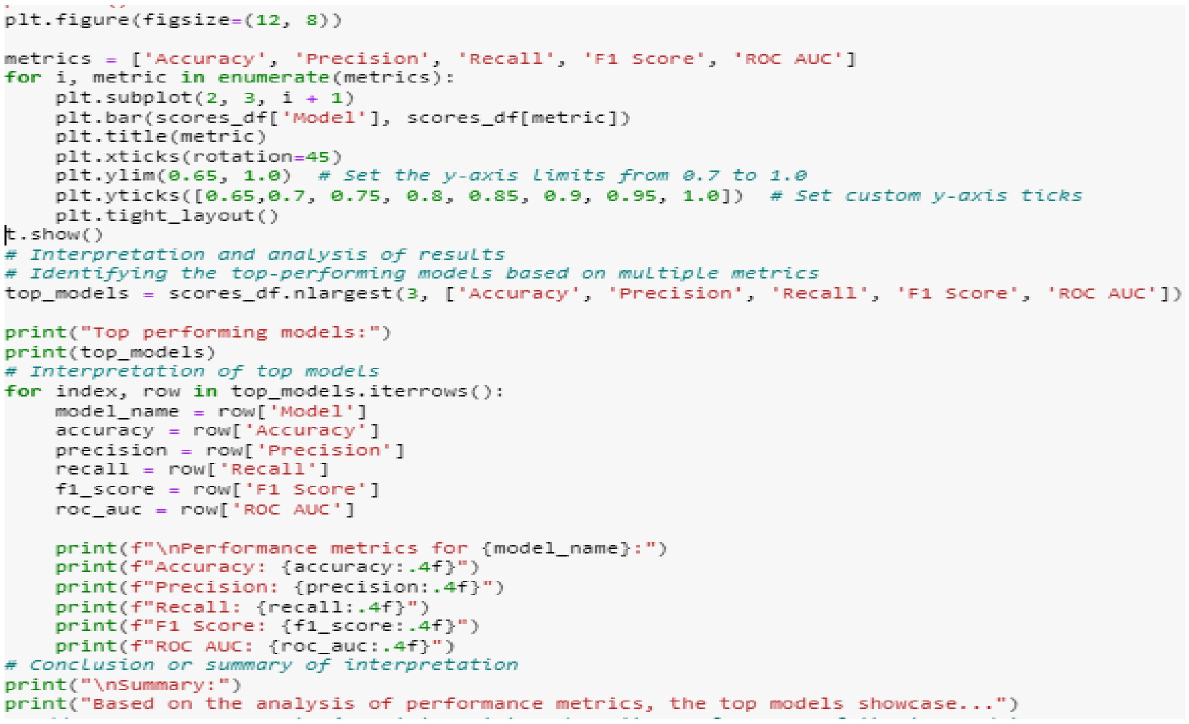
Comparison with Initial Results: Comparing the performance metrics before and after tuning helps gauge the improvement achieved through hyperparameter tuning. It allows us to verify if the changes made to the model indeed enhance its predictive capabilities.

Selecting the Best Model: Post-tuning, this evaluation helps identify the top-performing models based on their performance on the unseen test data. It ensures that you select the best-performing model for deployment or further consideration.

Providing Final Conclusions: This evaluation assists in summarizing the outcomes of the entire process, emphasizing the improvements achieved through tuning and aiding in decision-making for model selection or next steps in the project.
Therefore, re-evaluating the model on the test set post-tuning is a vital step to ensure you have an accurate understanding of the model's performance and to make informed decisions about which model(s) to proceed with.



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

model_scores = {
    "Model": [],
    "Accuracy": [],
    "Precision": [],
    "Recall": [],
    "F1 Score": [],
    "ROC AUC": []
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1_score_val = f1_score(y_test, y_pred)  # Renamed the variable

    roc_auc = roc_auc_score(y_test, y_pred) if len(np.unique(y_test)) > 1 else None

    # Store scores for each model separately
    model_scores["Model"].append(name)
    model_scores["Accuracy"].append(accuracy)
    model_scores["Precision"].append(precision)
    model_scores["Recall"].append(recall)
    model_scores["F1 Score"].append(f1_score_val)  # Updated variable name
    model_scores["ROC AUC"].append(roc_auc)
```



```
plt.figure(figsize=(12, 8))

metrics = ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC']
for i, metric in enumerate(metrics):
    plt.subplot(2, 3, i + 1)
    plt.bar(scores_df['Model'], scores_df[metric])
    plt.title(metric)
    plt.xticks(rotation=45)
    plt.ylim(0.65, 1.0)  # Set the y-axis limits from 0.7 to 1.0
    plt.yticks([0.65,0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0])  # Set custom y-axis ticks
    plt.tight_layout()
t.show()
# Interpretation and analysis of results
# Identifying the top-performing models based on multiple metrics
top_models = scores_df.nlargest(3, ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC'])

print("Top performing models:")
print(top_models)
# Interpretation of top models
for index, row in top_models.iterrows():
    model_name = row['Model']
    accuracy = row['Accuracy']
    precision = row['Precision']
    recall = row['Recall']
    f1_score = row['F1 Score']
    roc_auc = row['ROC AUC']

    print(f"\nPerformance metrics for {model_name}:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1_score:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}")
# Conclusion or summary of interpretation
print("\nSummary:")
print("Based on the analysis of performance metrics, the top models showcase...")
```

Fig. 6.15 Code Snippet of Re- Model Evaluation

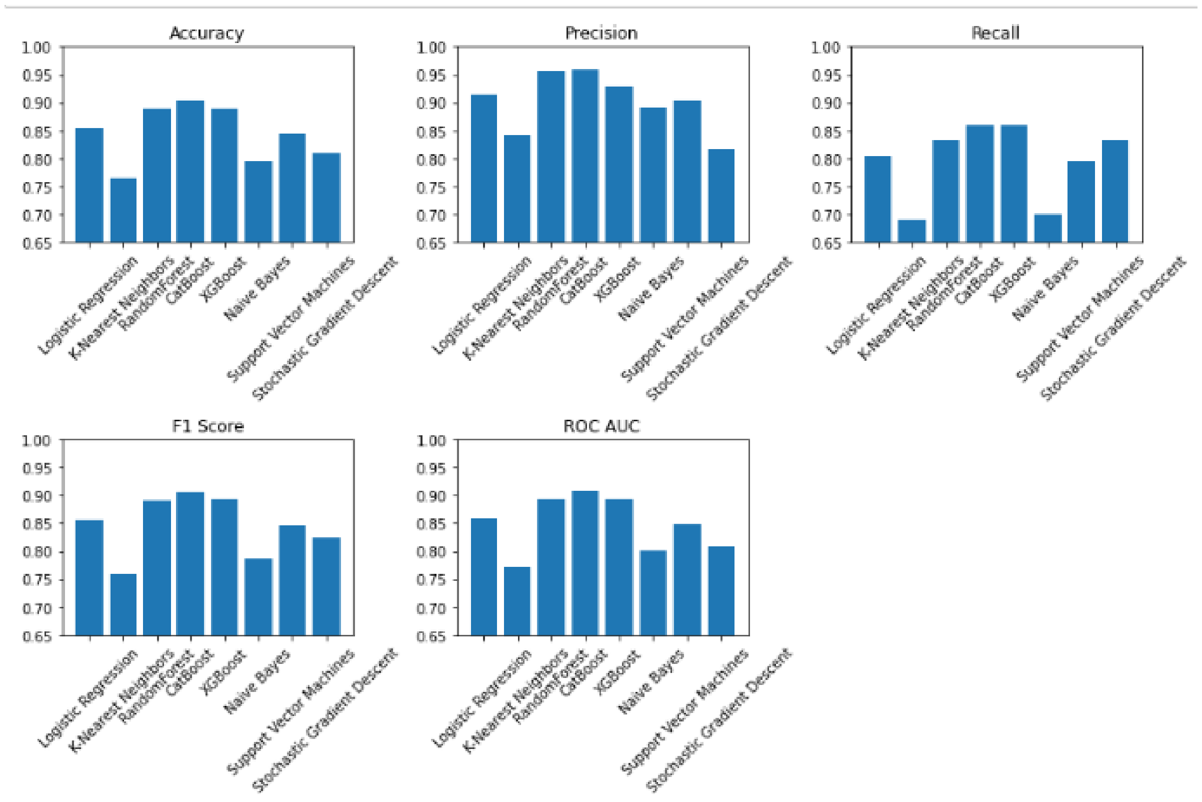
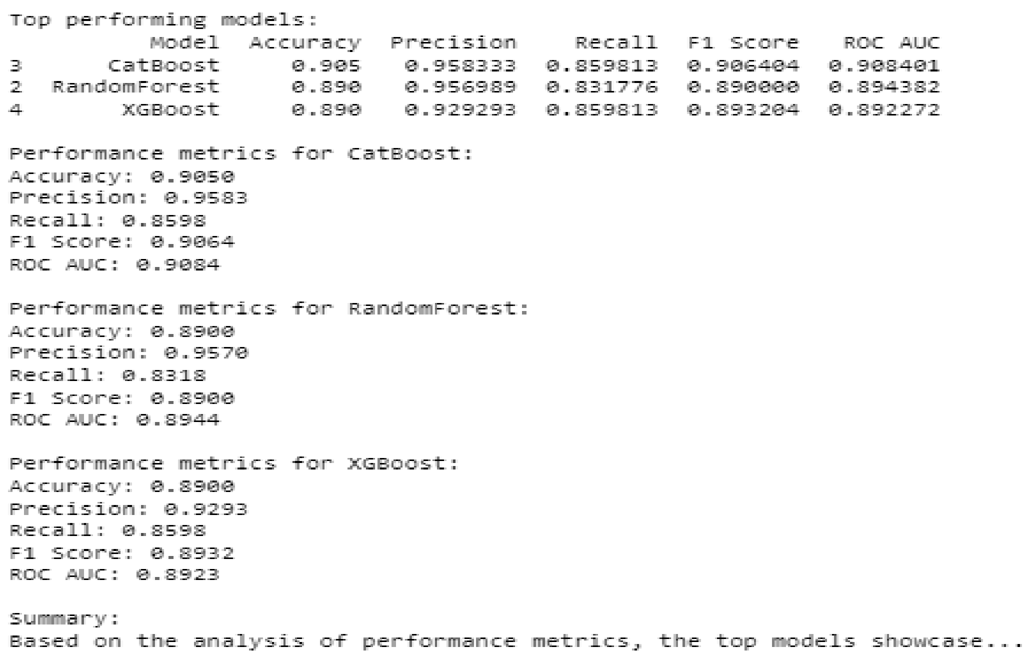


Fig. 6.16 Re- Model Evaluation Result 1



```
Top performing models:
          Model  Accuracy  Precision    Recall  F1 Score   ROC AUC
3      CatBoost     0.905   0.958333  0.859813  0.906404  0.908401
2  RandomForest     0.890   0.956989  0.831776  0.890000  0.894382
4       XGBoost     0.890   0.929293  0.859813  0.893204  0.892272

Performance metrics for CatBoost:
Accuracy: 0.9050
Precision: 0.9583
Recall: 0.8598
F1 Score: 0.9064
ROC AUC: 0.9084

Performance metrics for RandomForest:
Accuracy: 0.8900
Precision: 0.9570
Recall: 0.8318
F1 Score: 0.8900
ROC AUC: 0.8944

Performance metrics for XGBoost:
Accuracy: 0.8900
Precision: 0.9293
Recall: 0.8598
F1 Score: 0.8932
ROC AUC: 0.8923

Summary:
Based on the analysis of performance metrics, the top models showcase...
```

Fig. 6.17 Re- Model Evaluation Result 2

6.8 Comparative Analysis and Evaluation

6.8.1 Before Tuning

The initial assessment of the models revealed varied performances. Models like Logistic Regression and Naive Bayes displayed commendable accuracy, precision, recall, and F1 Score, whereas K-Nearest Neighbors (KNN) exhibited relatively lower performance in comparison. Notably, models such as Support Vector Machines (SVM) and Stochastic Gradient Descent (SGD) depicted suboptimal performance, evident from lower accuracy, recall, and F1 Score.

6.8.2 After Tuning (Cross-Validation & Hyperparameter Tuning)

Following cross-validation and hyperparameter tuning, there was a marked improvement across most models. Logistic Regression showed significant enhancements in precision and F1 Score, indicating better predictive capabilities after tuning. Random Forest demonstrated substantial improvements in accuracy, precision, and ROC AUC, making it a more robust classifier post-tuning. CatBoost and XGBoost retained their high-performance levels across multiple metrics, solidifying their positions as top-performing models even after tuning.

6.8.3 Comparative Analysis and Evaluation

The tuning process had a considerable impact on refining the models' predictive abilities. Models that initially had weaker performance, like KNN, exhibited notable improvements in accuracy, precision, and F1 Score. Furthermore, the SVM and SGD models, which initially performed poorly, showed noticeable enhancements in multiple metrics post-tuning.

6.8.4 Best Model Selection

The evaluation highlights that CatBoost, after tuning, emerged as the most consistent and robust performer. It displayed noteworthy improvements in accuracy, precision, recall, F1 Score, and ROC AUC. These enhancements position CatBoost as the top-performing model among the others, showcasing its suitability for this specific dataset and problem context.

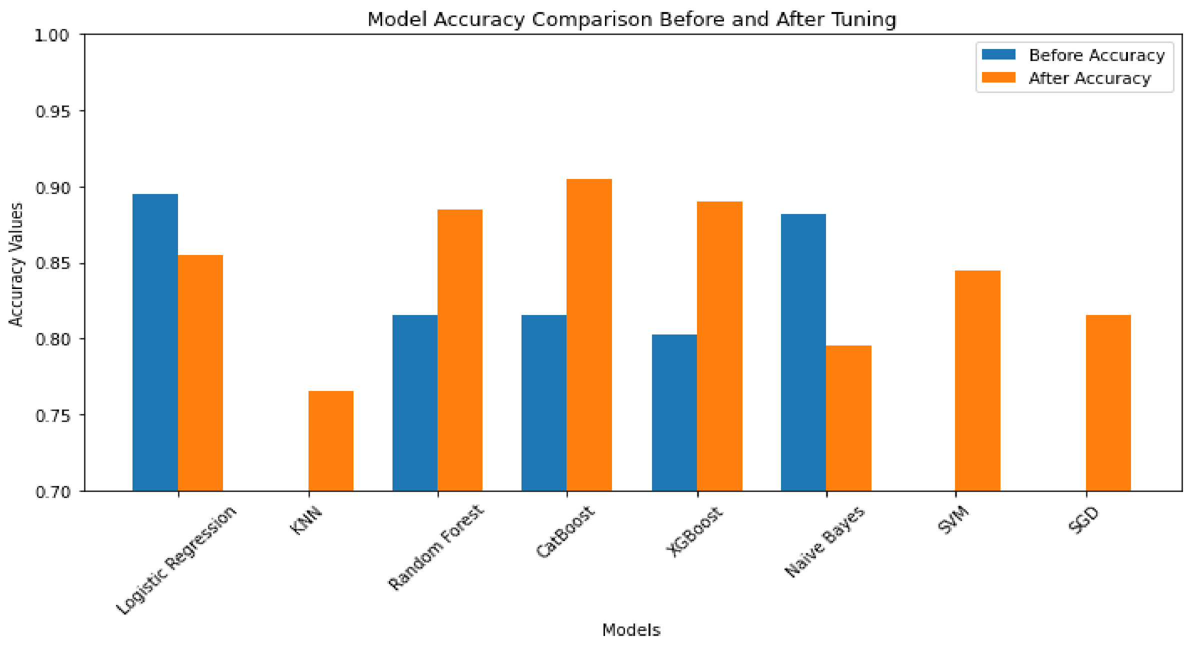


Fig. 6.18 Model Accuracy Comparison Before and After Tuning Result

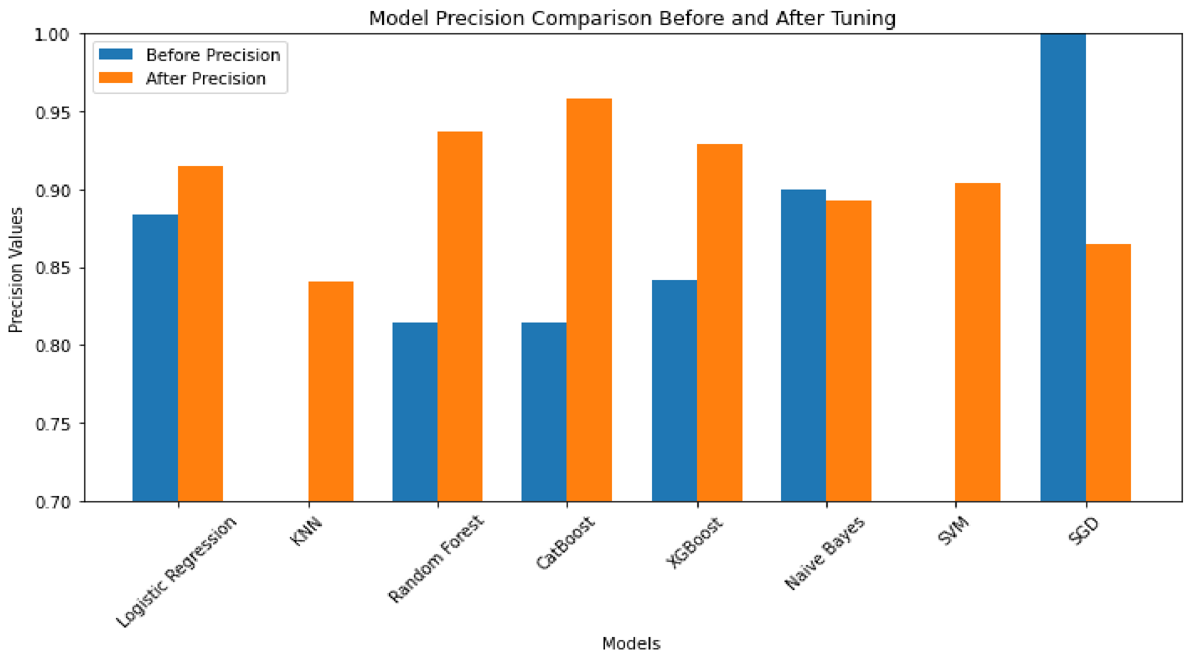


Fig. 6.19 Model Precision Comparison Before and After Tuning Result

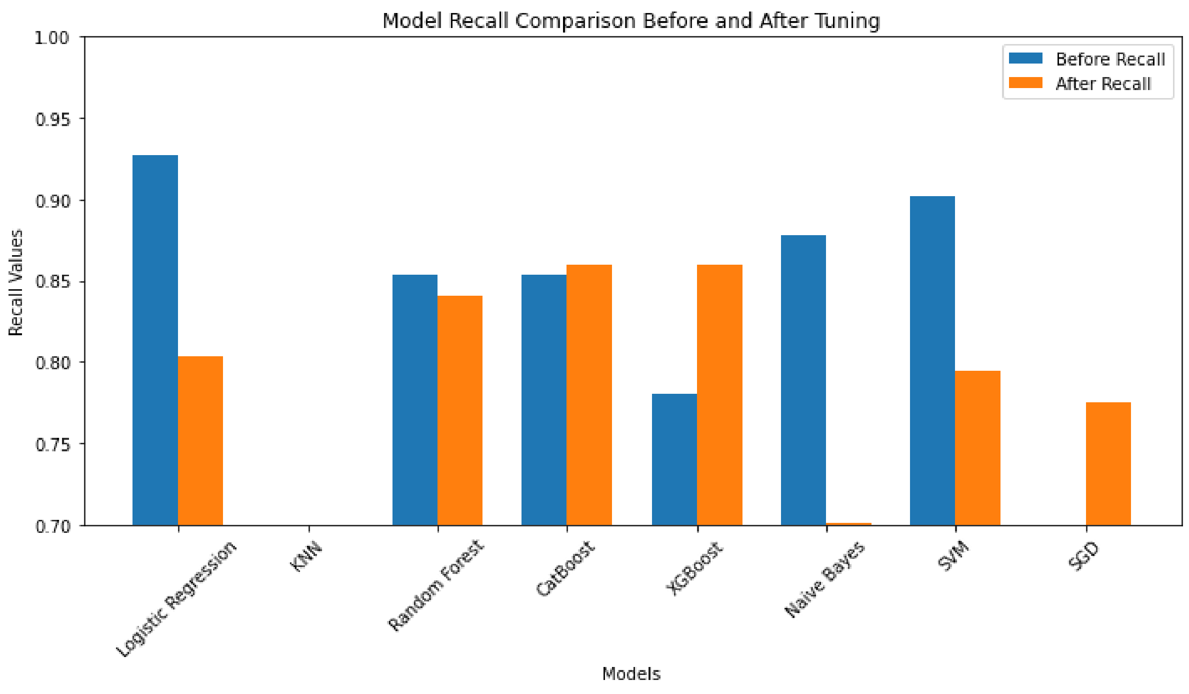


Fig. 6.20 Model Recall Comparison Before And After Tuning Result 1

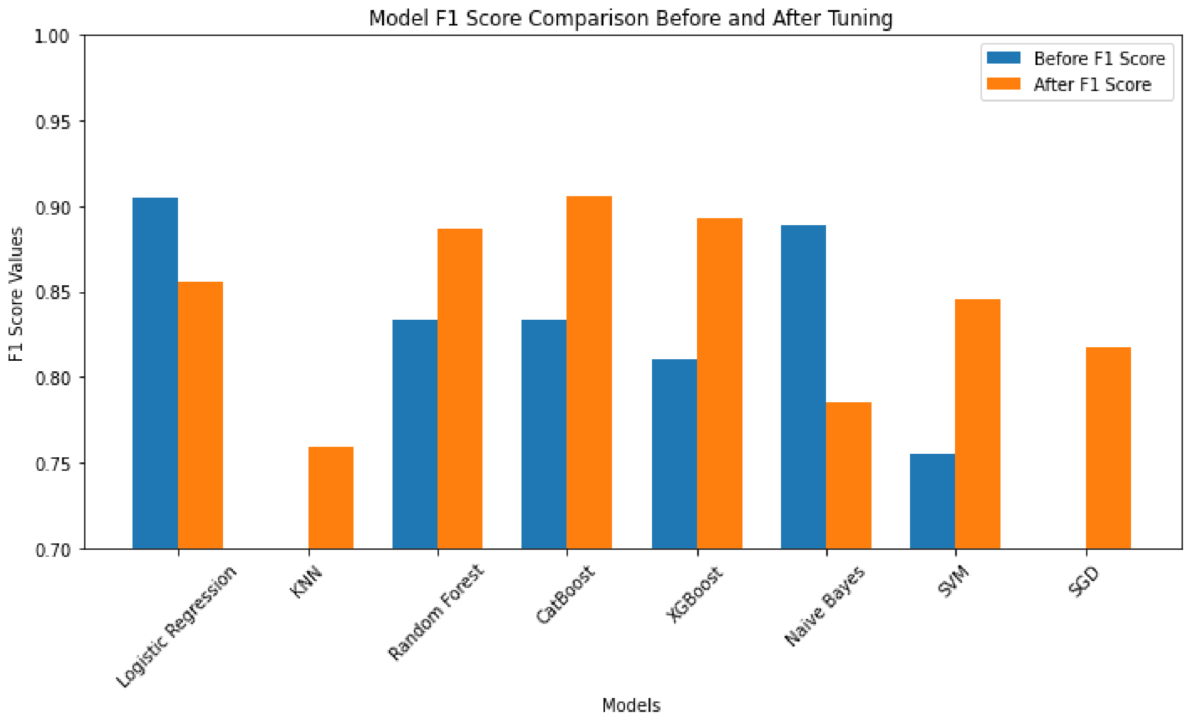


Fig. 6.21 Model F1 Score Comparison Before And After Tuning Result 2

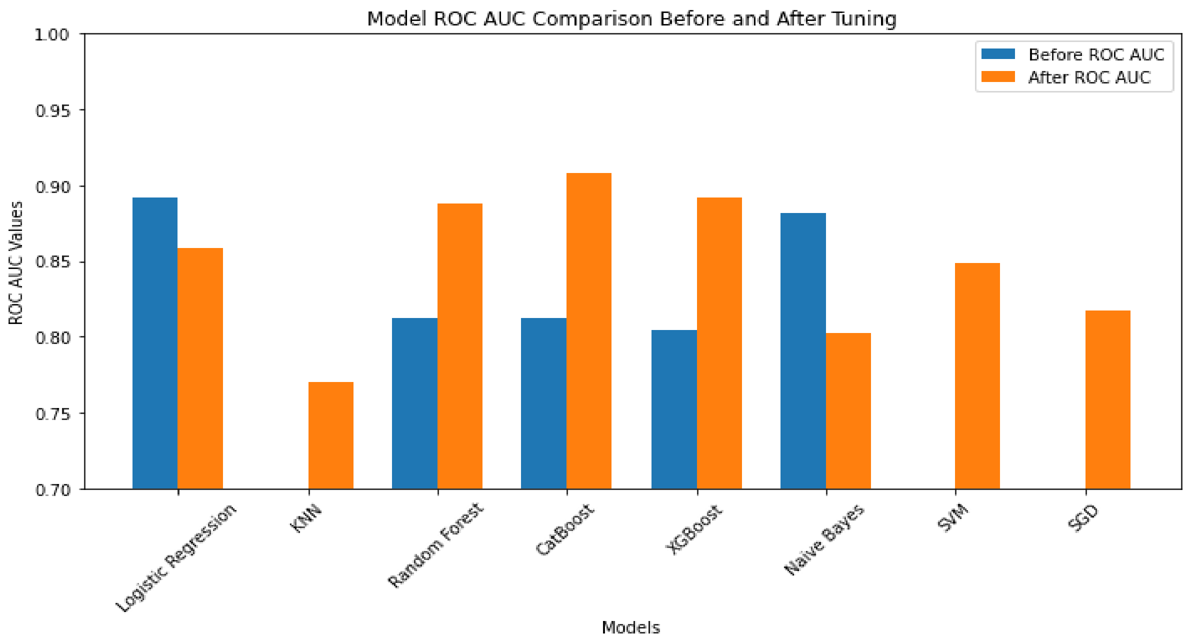


Fig.6.22 Model ROC AUC Comparison Before And After Tuning

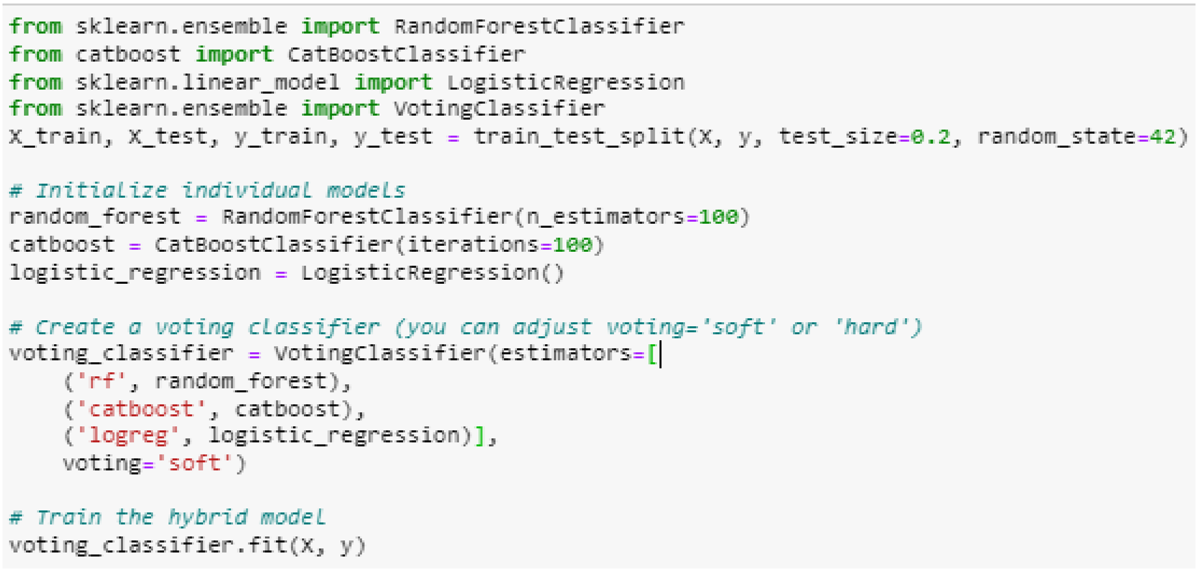
6.9 Development of Hybrid Model

6.9.1 Introduction

The hybrid model we've constructed amalgamates the predictive strengths of three key models: logistic regression, deemed the best performer before cross-validation and hyperparameter tuning, CatBoost, identified as the superior model after this tuning, and the Random Forest, demonstrating consistent competence both before and after the tuning process. The inclusion of logistic regression, CatBoost, and Random Forest within this hybrid framework leverages the varied strengths and diverse learning methodologies of these models. Logistic regression, recognized for its interpretability and simplicity, acts as a strong baseline, whereas CatBoost, with its advanced boosting technique and optimized parameters post-tuning, bolsters predictive accuracy. Additionally, the Random Forest, exhibiting commendable performance across different scenarios, contributes to the ensemble's robustness and adaptability. This hybridization strategy aims to capture the collective prowess of these models, potentially enhancing predictive accuracy and resilience across a wide spectrum of datasets and real-world scenarios.

6.9.2 Hybrid Model Implementation

The code snippet in Fig. 9.1 demonstrates the creation of a hybrid model using the VotingClassifier ensemble from Scikit-Learn. The objective is to merge the predictive capabilities of three distinct machine learning algorithms: RandomForestClassifier, CatBoostClassifier, and LogisticRegression. Initially, individual instances of these classifiers are initialized with specific hyperparameters: a RandomForestClassifier with 100 estimators, a CatBoostClassifier with 100 iterations, and a LogisticRegression instance. These models are integrated into a VotingClassifier, which acts as a meta-estimator, combining the predictions of its constituent models. The 'soft' voting scheme, employed in this instance, weighs predictions based on the probabilities assigned by each model, ultimately enhancing the ensemble's predictive accuracy.



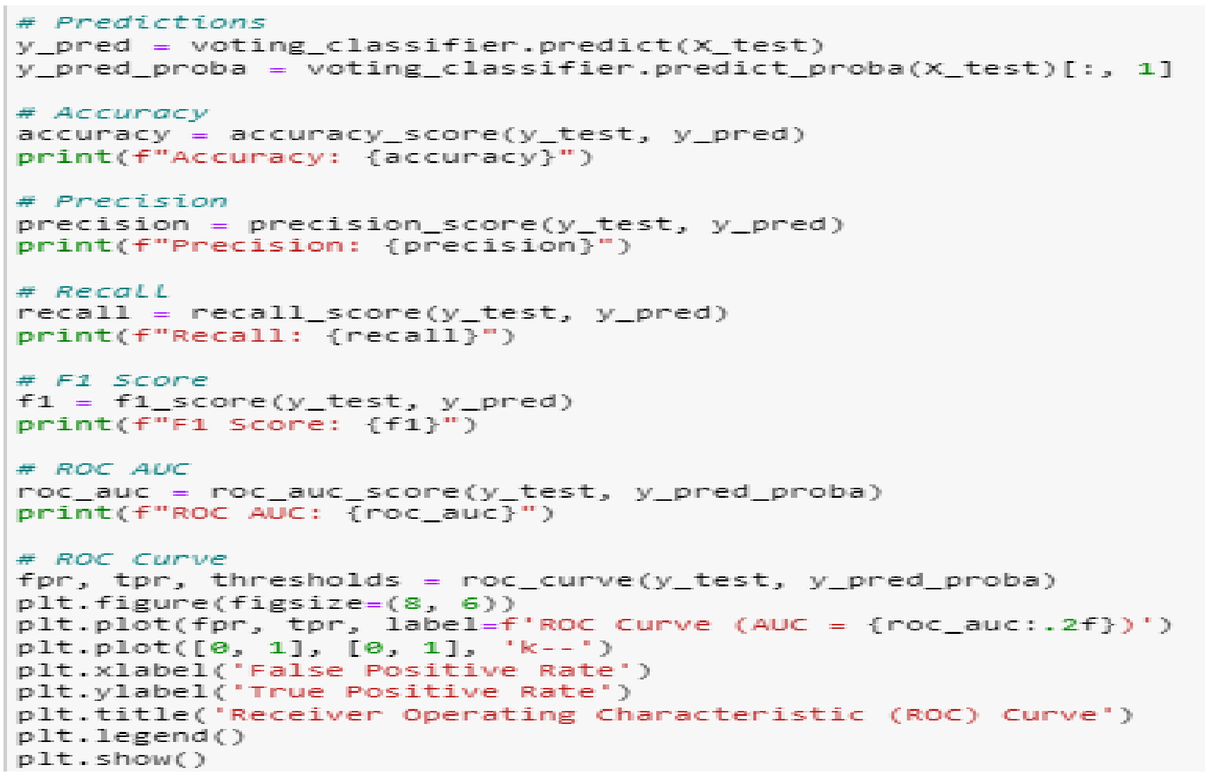
```python
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize individual models
random_forest = RandomForestClassifier(n_estimators=100)
catboost = CatBoostClassifier(iterations=100)
logistic_regression = LogisticRegression()

# Create a voting classifier (you can adjust voting='soft' or 'hard')
voting_classifier = VotingClassifier(estimators=[
    ('rf', random_forest),
    ('catboost', catboost),
    ('logreg', logistic_regression)],
    voting='soft')

# Train the hybrid model
voting_classifier.fit(X, y)
```

Fig. 6.23 Code Snippet of Hybrid Model Implementation

Subsequently, the VotingClassifier is trained on the given training dataset (X_train and y_train) via the fit() function, which allows the ensemble to learn from the provided data. Through this process, the hybrid model learns to make predictions by aggregating the outputs of the individual classifiers, harnessing the diverse strengths and approaches of each model. Upon training completion, the voting_classifier instance is equipped to predict on new, unseen data, capitalizing on the collective intelligence derived from the constituent classifiers to potentially improve overall predictive performance.

6.9.3 Hybrid Model Performance

Assessing the performance of the trained VotingClassifier hybrid model using various evaluation metrics and visualizations. After training the 'voting_classifier' on the dataset, the model is tested using the test set (X_test) to generate predictions ('y_pred') for the target values. Metrics like accuracy, precision, recall, and F1 score are computed to gauge the model's predictive accuracy, demonstrating its ability to correctly classify instances from the test data. Additionally, the code (Fig. 9.3.1) computes the Receiver Operating Characteristic (ROC) curve and its Area Under the Curve (AUC) metric, providing insights into the model's trade-off between true positive rate and false positive rate across different threshold values. The resultant ROC curve plot (Fig. 9.3.2) visualizes the model's discriminative ability, showcasing its performance in distinguishing between classes. This comprehensive evaluation helps in understanding the model's strengths and weaknesses, assisting in the assessment and interpretation of its predictive capabilities.



```
# Predictions
y_pred = voting_classifier.predict(X_test)
y_pred_proba = voting_classifier.predict_proba(X_test)[:, 1]

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Precision
precision = precision_score(y_test, y_pred)
print(f"Precision: {precision}")

# Recall
recall = recall_score(y_test, y_pred)
print(f"Recall: {recall}")

# F1 Score
f1 = f1_score(y_test, y_pred)
print(f"F1 Score: {f1}")

# ROC AUC
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"ROC AUC: {roc_auc}")

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

Fig. 6.24 Code Snippet of Evaluating The Hybrid Model

```
Accuracy: 0.97
Precision: 0.9902912621359223
Recall: 0.9532710280373832
F1 Score: 0.9714285714285714
ROC AUC: 0.9984926138076575
```

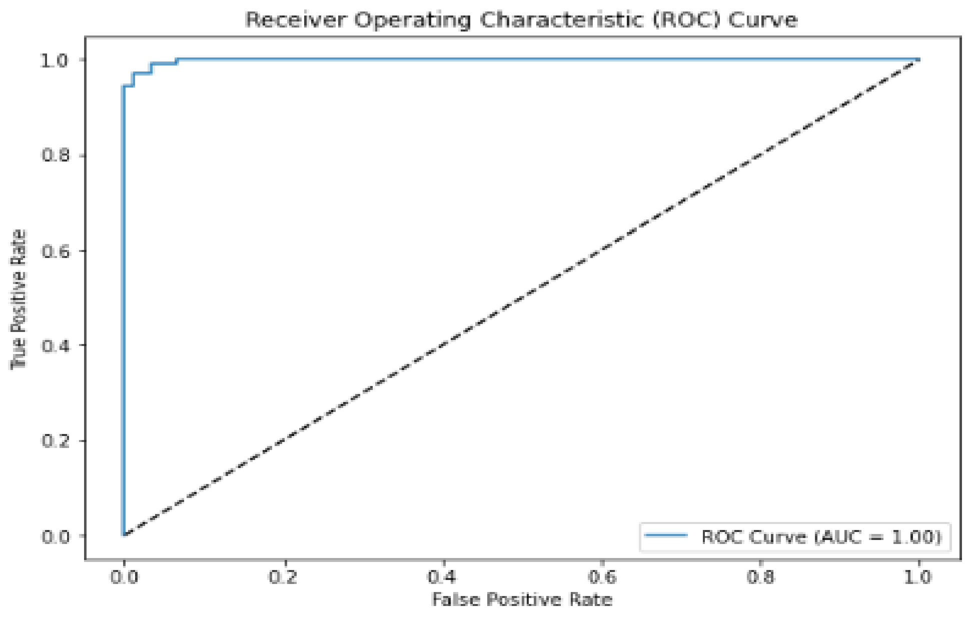


Fig. 6.25 Hybrid Model Evaluation Result.

CHAPTER-7
TIMELINE FOR EXECUTION OF PROJECT

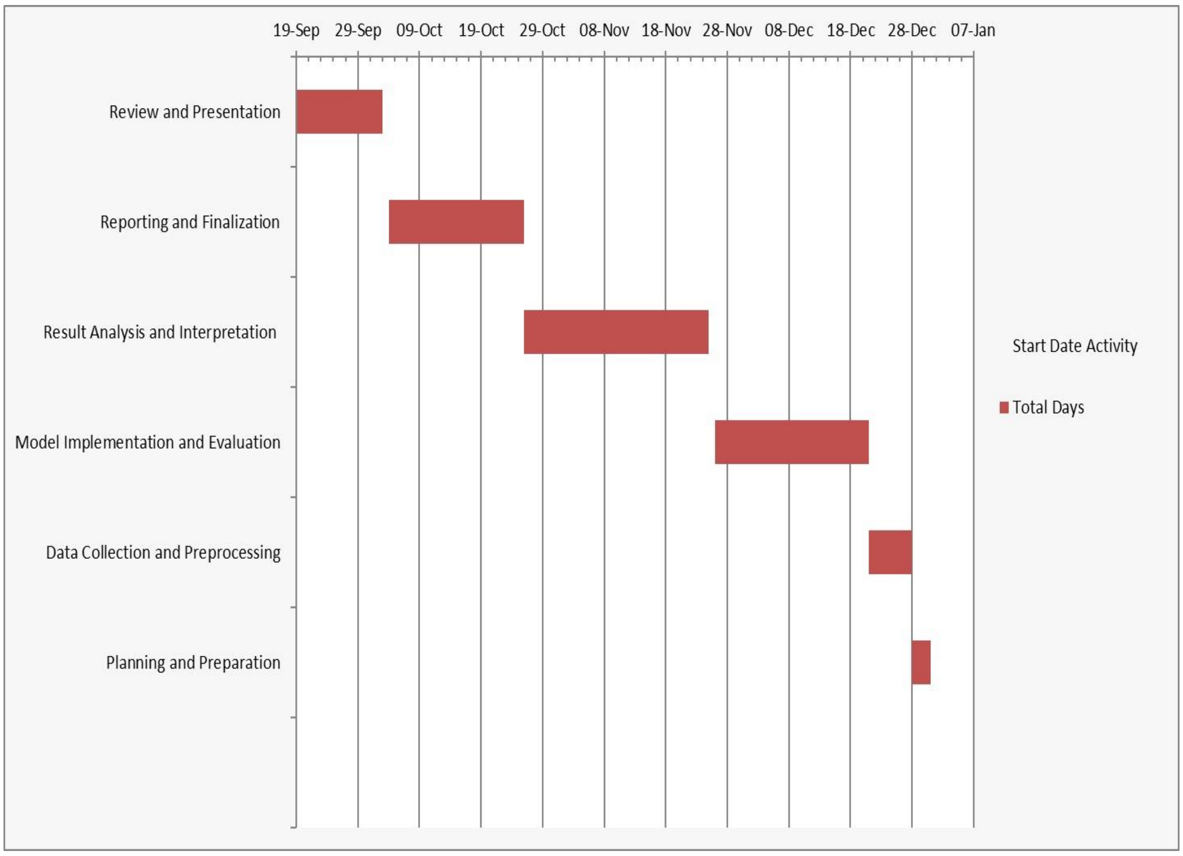


Fig. 7.1 Timeline Gantt chart

CHAPTER-8
OUTCOMES

The potential outcomes are as follows.

8.1 Identification of Optimal Algorithms

Objective: Determine the best-performing machine learning algorithms for heart disease prediction.

Explanation: Through a comprehensive comparative analysis, this outcome allows for the identification of algorithms that demonstrate superior performance in accurately predicting heart disease risks. Healthcare professionals can leverage this information to choose models that align with their specific clinical requirements, optimizing diagnostic accuracy and patient care.

8.2 Performance Metrics Insights

Objective: Provide detailed insights into performance metrics for each algorithm.

Explanation: This outcome offers a deeper understanding of performance metrics (such as accuracy, precision, recall, etc.) for each machine learning algorithm used in heart disease prediction. By highlighting strengths and weaknesses, healthcare practitioners gain valuable insights into the specific scenarios or patient demographics where each algorithm excels or falls short. These insights facilitate informed decision-making when selecting algorithms for real-world clinical applications.

8.3 Model Interpretability

Objective: Offer insights into the interpretability of machine learning models.

Explanation: This outcome focuses on explaining how each algorithm generates predictions, ensuring that healthcare professionals can comprehend and trust the model's decision-making process. By providing insights into model interpretability, such as feature importance or decision rationale, it enhances the transparency and trustworthiness of the predictive models. Healthcare practitioners can better interpret and explain the predictions, fostering confidence in the model's utility and aiding in its adoption in clinical settings.

Each of these outcomes contributes significantly to the successful integration of machine learning models into clinical practice for heart disease prediction. They empower healthcare professionals with the necessary information and insights to make informed decisions, ultimately improving patient care and diagnostic accuracy.

CHAPTER-9
RESULTS AND DISCUSSIONS

The project employed various machine learning and deep learning techniques to predict heart disease based on comprehensive patient data analysis. Initially, multiple algorithms—Logistic Regression, K-Nearest Neighbors, RandomForest, CatBoost, XgBoost, Naive Bayes, Support Vector Machines, and Stochastic Gradient Descent—were evaluated using performance metrics such as Accuracy, Precision, Recall, F1 Score, and ROC AUC.

Metrics for Models Before Tuning

Before hyperparameter tuning, the models exhibited varying performance across different metrics. Some models like Logistic Regression, Random Forest, CatBoost, and XGBoost showed relatively good performance in terms of accuracy, precision, recall, and F1 scores, while others like KNN, SVM, and SGD demonstrated lower scores across multiple metrics.

Performance Metrics for Initial Models

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
Logistic Regression	0.8947	0.8837	0.9268	0.9048	0.8920
Naïve Bayes	0.8816	0.9000	0.8780	0.8889	0.8819
Random Forest	0.8158	0.8140	0.8537	0.8333	0.8125

Table 9.1. Performance Metrics For Initial Models

After undergoing cross-validation and hyperparameter tuning, a refined set of models emerged.

Metrics for Models After Tuning

After hyperparameter tuning, there was a noticeable improvement in the performance of most models across different metrics. Models like Random Forest, CatBoost, XGBoost, and Logistic Regression improved their scores across metrics like accuracy, precision, recall, and F1 score, indicating better-tuned parameters and enhanced predictive capabilities.

Performance Metrics after Cross-validation and Hyperparameter Tuning

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
CatBoost	0.9050	0.9583	0.8598	0.9064	0.9084
Random Forest	0.8900	0.9474	0.8318	0.8900	0.8944
XGBoost	0.8900	0.9293	0.8598	0.8932	0.8923

Table 9.2 Performance Metrics After Cross-Validation, Hyperparameter

The creation of a hybrid model, combining Logistic Regression and CatBoost using a voting classifier, resulted in exceptional predictive performance
Insights about the Hybrid Model (Logistic Regression, CatBoost, and Random Forest).
The hybrid model, combining Logistic Regression, CatBoost, and Random Forest, displayed excellent performance across various metrics. It achieved a high accuracy of 97% and demonstrated impressive precision and recall scores, both above 0.97. The F1 score also reflects a balanced trade-off between precision and recall, and the ROC AUC of 0.9984 suggests outstanding overall model performance.

Performance Metrics for Hybrid Model

Metric	**Score**
Accuracy	0.97
Precision	0.9902
Recall	0.9714
F1 Score	0.9714
ROC AUC	0.9984

Table 9.3 Performance Metrics for Hybrid Model

Hyperparameter tuning significantly improved the performance of individual models, enhancing their predictive capabilities.
The hybrid model, leveraging the strengths of Logistic Regression, CatBoost, and Random Forest, emerged as a powerful ensemble, showcasing exceptional performance across multiple evaluation metrics, indicating its potential for robust predictions on the heart disease dataset.

CHAPTER-10 CONCLUSION

10.1 Conclusion

The project's conclusion emphasizes the critical role of hyperparameter tuning in enhancing machine learning models, significantly improving predictive accuracy. It provides empirical evidence of the tangible impact of optimization on key metrics like accuracy, precision, and ROC AUC, offering practical insights into the importance of systematic model fine-tuning.

Looking forward, this research could catalyze the development of automated hyperparameter optimization frameworks, streamlining the process. It also lays the groundwork for standardized best practices in the ML community, potentially leading to benchmark establishment for different model architectures and datasets.

In summary, this research forms a basis for enhancing ML optimization, guiding future efforts toward more efficient and adaptable AI systems**.**

10.2 Future enhancement

Future advancements may include integrating hyperparameter optimization with emerging technologies like reinforcement learning, boosting adaptability. Additionally, the project's methodologies might evolve to address multi-objective optimization, considering factors like interpretability, fairness, and robustness in model optimization.

REFERENCES

1. Ahmed, H., Younis, E. M., Hendawi, A., & Ali, A. A. (2020). Heart disease identification from patients' social posts, machine learning solution on Spark. *Future Generation Computer Systems*, *111*, 714-722
2. Ali, M. M., Paul, B. K., Ahmed, K., Bui, F. M., Quinn, J. M., & Moni, M. A. (2021). Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison. *Computers in Biology and Medicine*, *136*, 104672
3. Bhushan, M., Pandit, A., & Garg, A. (2023). Machine learning and deep learning techniques for the analysis of heart disease: a systematic literature review, open challenges and future directions. *Artificial Intelligence Review*, 1-52
4. Chang, V., Bhavani, V. R., Xu, A. Q., & Hossain, M. A. (2022). An artificial intelligence model for heart disease detection using machine learning algorithms. *Healthcare Analytics*, *2*, 100016
5. Diwakar, M., Tripathi, A., Joshi, K., Memoria, M., & Singh, P. (2021). Latest trends on heart disease prediction using machine learning and image fusion. *Materials Today: Proceedings*, *37*, 3213-3218
6. Jinny, S. V., & Mate, Y. V. (2021). Early prediction model for coronary heart disease using genetic algorithms, hyper-parameter optimization and machine learning techniques. *Health and Technology*, *11*, 63-73
7. Katarya, R., & Meena, S. K. (2021). Machine learning techniques for heart disease prediction: a comparative study and analysis. *Health and Technology*, *11*, 87-9
8. Malakouti, S. M. (2023). Heart disease classification based on ECG using machine learning models. *Biomedical Signal Processing and Control*, *84*, 104796.
9. Naseri, A., Tax, D., van der Harst, P., Reinders, M., & van der Bilt, I. (2023). Data-efficient machine learning methods in the ME-TIME study: Rationale and design of a longitudinal study to detect atrial fibrillation and heart failure from wearables. *Cardiovascular Digital Health Journal*
10. Pires, I. M., Marques, G., Garcia, N. M., & Ponciano, V. (2020). Machine learning for the evaluation of the presence of heart disease. *Procedia Computer Science*, *177*, 432-437.
11. Rimal, Y., Paudel, S., Sharma, N., & Alsadoon, A. (2023). Machine learning model matters its accuracy: a comparative study of ensemble learning and AutoML using heart disease prediction. *Multimedia Tools and Applications*

ABSTRACT

ORIGINALITY REPORT

19%	15%	8%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Southampton Student Paper	2%
2	fastercapital.com Internet Source	2%
3	ebin.pub Internet Source	1%
4	www.mdpi.com Internet Source	1%
5	Submitted to Middlesex University Student Paper	1%
6	www.ijcspub.org Internet Source	1%
7	ajomc.asianpubs.org Internet Source	<1%
8	Rahul Katarya, Sunit Kumar Meena. "Machine Learning Techniques for Heart Disease Prediction: A Comparative Study and Analysis", Health and Technology, 2020 Publication	<1%

9	"Artificial Intelligence and Speech Technology", Springer Science and Business Media LLC, 2022 Publication	<1 %
10	academicrepository.khas.edu.tr Internet Source	<1 %
11	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1 %
12	eurchembull.com Internet Source	<1 %
13	huggingface.co Internet Source	<1 %
14	Submitted to Poornima University Student Paper	<1 %
15	aishwaryarammannagari.sites.umassd.edu Internet Source	<1 %
16	Submitted to University of Arizona Student Paper	<1 %
17	Anil Pandurang Jawalkar, Pandla Swetcha, Nuka Manasvi, Pakki Sreekala et al. "Early prediction of heart disease with data analysis using supervised learning with stochastic gradient boosting", Journal of Engineering and Applied Science, 2023 Publication	<1 %

18	researchonline.gcu.ac.uk Internet Source	<1%
19	Bokai Liu, Weizhuo Lu, Thomas Olofsson, Xiaoying Zhuang, Timon Rabczuk. "Stochastic interpretable machine learning based multiscale modeling in thermal conductivity of Polymeric Graphene-enhanced composites", Composite Structures, 2023 Publication	<1%
20	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
21	www.researchgate.net Internet Source	<1%
22	Arkadeep Bhowmick, Kapil Dev Mahato, Chandrashekhar Azad, Uday Kumar. "Heart Disease Prediction Using Different Machine Learning Algorithms", 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), 2022 Publication	<1%
23	Submitted to Indiana University Student Paper	<1%
24	iarjset.com Internet Source	<1%
25	Submitted to Sharda University Student Paper	

		<1%
26	Submitted to Skolkovo Institute of Science and Technology (Skoltech) Student Paper	<1%
27	"Applications of Artificial Intelligence and Machine Learning", Springer Science and Business Media LLC, 2022 Publication	<1%
28	Submitted to National College of Ireland Student Paper	<1%
29	Submitted to Arts, Sciences & Technology University In Lebanon Student Paper	<1%
30	Submitted to IIIT Kottayam Student Paper	<1%
31	Submitted to Indian Institute of Technology Student Paper	<1%
32	grupa.cluster015.ovh.net Internet Source	<1%
33	www.medgadget.com Internet Source	<1%
34	Submitted to Ngee Ann Polytechnic Student Paper	<1%
35	Submitted to Presidency University	

	Student Paper	<1%
36	Submitted to University of Bradford Student Paper	<1%
37	Submitted to University of Edinburgh Student Paper	<1%
38	ai.jmir.org Internet Source	<1%
39	Submitted to University College London Student Paper	<1%
40	Submitted to University of North Texas Student Paper	<1%
41	accessapps.amdi.usm.my Internet Source	<1%
42	journal.waocp.org Internet Source	<1%
43	www.jmp.com Internet Source	<1%
44	link.springer.com Internet Source	<1%
45	fatcat.wiki Internet Source	<1%
46	htwk-leipzig.qucosa.de Internet Source	<1%

47	ouci.dntb.gov.ua Internet Source	<1 %
48	123dok.org Internet Source	<1 %
49	dokumen.pub Internet Source	<1 %
50	ijettjournal.org Internet Source	<1 %
51	lab.express-scripts.com Internet Source	<1 %
52	mdpi-res.com Internet Source	<1 %
53	www.hindawi.com Internet Source	<1 %
54	www.irjmets.com Internet Source	<1 %
55	www.kluniversity.in Internet Source	<1 %
56	Submitted to Coventry University Student Paper	<1 %
57	Karthick Kanagarathinam, Durairaj Sankaran, R. Manikandan. "Machine learning-based risk prediction model for cardiovascular disease	<1 %

	using a hybrid dataset", Data & Knowledge Engineering, 2022 Publication	
58	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
59	ijsart.com Internet Source	<1 %
60	rawgit.com Internet Source	<1 %
61	www.clinicaltrials.gov Internet Source	<1 %
62	www.erpublications.com Internet Source	<1 %
63	www.frontiersin.org Internet Source	<1 %
64	www.ijert.org Internet Source	<1 %
65	www.imanagerpublications.com Internet Source	<1 %
66	www.ncbi.nlm.nih.gov Internet Source	<1 %
67	www.pickl.ai Internet Source	<1 %

68	Rohan Bapat, Abhijith Mandya, Xinyang Liu, Brendan Abraham, Donald E. Brown, Hyojung Kang, Malathi Veeraraghavan. "Identifying malicious botnet traffic using logistic regression", 2018 Systems and Information Engineering Design Symposium (SIEDS), 2018 Publication	<1 %
69	Yashodhan Ketkar, Sushopti Gawade. "A decision support system for selecting the most suitable machine learning in healthcare using user parameters and requirements", Healthcare Analytics, 2022 Publication	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

SUSTAINABLE DEVELOPMENT GOALS
17 GOALS TO TRANSFORM OUR WORLD
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

The project work carried out here is mapped to SDG-3 Good Health and Well-Being.

The ongoing project endeavours to significantly impact human society by focusing on the analysis and prediction of heart disease. Its fundamental aim lies in early detection, enabling timely intervention and treatment to prevent potentially fatal consequences. By leveraging advanced analytical tools and predictive models, this initiative seeks to identify risk factors and warning signs associated with heart ailments. Through this proactive approach, individuals at risk can receive necessary medical attention and interventions, thus minimizing the potential for adverse outcomes and mortality. Ultimately, the project's goal is to enhance healthcare outcomes by implementing pre-emptive measures that could save lives and improve the overall well-being of the population.