Advanced Heart Assessment through Machine Learning Using KNN Algorithm

A PROJECT REPORT

Submitted by,

Mr. Surag Nekkanti	**20201CSE0207**
Mr. Navaneeth Nataraja Yogachar	**20201CSE0255**
Mr. R Gautham Ganesh	**20201CSE0200**
Mr. Shaun Joe Roy	**20201CSE0237**

Under the guidance of,

Dr. S Prakash

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"Advanced Heart Assessment through Machine Learning Using KNN Algorithm"** being submitted by **Surag Nekkanti, Navaneeth Nataraja Yogachar, R Gautham Ganesh, Shaun Joe Roy** bearing roll number(s**) 20201CSE0207, 20201CSE0255, 20201CSE0200, 20201CSE0237** 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. S Prakash
Assistant Professor
School of CSE
Presidency University

Dr. Pallavi R
Associate Professor & HoD
School of CSE
Presidency University

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

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

Dr. SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Advanced Heart Assessment through Machine Learning Using KNN Algorithm** 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. S. Prakash, Assistant Professor, School of Computer Science and Engineering, Presidency University, Bengaluru.**

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

Signature

Surag Nekkanti	**20201CSE0207**
Navaneeth Nataraja Yogachar	**20201CSE0255**
R Gautham Ganesh	**20201CSE0200**
Shaun Joe Roy	**20201CSE0237**

ABSTRACT

Cardiovascular diseases, including heart disease, remain a global health challenge, necessitating innovative approaches for early detection and intervention. In this study, we propose a predictive model for heart disease using the K-Nearest Neighbors (KNN) algorithm, a machine learning technique renowned for its simplicity and effectiveness in classification tasks. Traditional methods of diagnosing heart disease often involve complex and time-consuming procedures such as ECG, 2D Echo, Treadmill Tests, and Blood Tests. Our model seeks to streamline this process by leveraging the power of KNN, providing a quicker and potentially more accurate means of identifying individuals at risk of heart disease. The heart disease prediction model is trained on a comprehensive dataset that includes diverse patient attributes such as age, gender, cholesterol levels, blood pressure, and other relevant clinical parameters. The KNN algorithm learns patterns and relationships within this data to make predictions based on the proximity of a new case to existing instances with known outcomes. This approach allows for personalized risk assessments, contributing to a more targeted and efficient healthcare strategy. One of the distinctive features of the KNN algorithm is its interpretability. Healthcare professionals can understand and validate the model's predictions, gaining insights into the factors influencing its decision-making process. This transparency is crucial in the medical field, where trust and comprehension of predictive models are essential for widespread acceptance and adoption.

The model's applicability extends beyond traditional hospital settings. By allowing individuals to input attribute values related to heart health, the predictive model can be integrated into routine blood tests conducted in various healthcare facilities, including those in rural areas. This democratization of predictive analytics empowers individuals, especially in underserved communities, by providing early indications of potential heart issues during routine health assessments.

In conclusion, our study introduces a novel application of the KNN algorithm for predicting heart disease, offering a user-friendly and efficient solution for early detection. As we navigate towards a future where data-driven healthcare plays a pivotal role, the integration of machine learning models, such as KNN, provides a promising avenue for enhancing cardiovascular disease management and reducing its societal impact.

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 **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science and Engineering & 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 and Engineering & Information Science, Presidency University and **Dr. Pallavi R**, Head of the Department, School of Computer Science and Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Dr. S Prakash**, **Assistant Professor**, School of Computer Science and 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 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.

Surag Nekkanti

Navaneeth Nataraja Yogachar

R Gautham Ganesh

Shaun Joe Roy

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

Within the intricate fabric of global health, cardiovascular diseases present a formidable challenge, being a prominent cause of morbidity and mortality worldwide. The escalating prevalence of heart-related ailments underscores the urgency for advanced diagnostic tools that offer timely and precise predictions. The fusion of cutting-edge technologies and data-driven methodologies has given rise to predictive analytics as a transformative force in healthcare. Among the myriad of machine learning algorithms, the K-Nearest Neighbors (KNN) algorithm stands out as a beacon of promise, particularly in the context of cardiovascular disease prediction. The global burden of heart disease necessitates a paradigm shift in diagnostic approaches. As healthcare systems grapple with the growing complexity of patient data and the demand for precision medicine, predictive analytics emerges as a revolutionary solution. At the core of this technological revolution lies the KNN algorithm, a versatile and intuitive machine learning technique celebrated for its prowess in classification and regression tasks. This article delves into the intricacies of the KNN algorithm, unraveling its fundamental principles and examining its application in predicting heart disease. The foundation of the KNN algorithm lies in the principle of proximity, a concept holding profound implications in healthcare. The algorithm classifies instances by scrutinizing the class labels of neighboring data points. In cardiovascular health, this involves a meticulous analysis of patient characteristics, discerning the likelihood of heart disease in new cases. KNN's essence lies in its ability to navigate patient data intricacies, offering a nuanced understanding beyond traditional diagnostic methods.The dataset used for training and testing the KNN algorithm comprises patient-related features such as age, gender, cholesterol levels, blood pressure, and other clinically relevant parameters. This dataset's richness serves as raw material for the algorithm to distill patterns and relationships, providing a nuanced prediction of heart disease.

The simplicity and transparency of the KNN algorithm resonate profoundly in healthcare. Unlike more complex counterparts, KNN's results are interpretable, enabling healthcare professionals to comprehend its predictions. In a field where decisions carry life-altering consequences, the ability to interpret predictions is indispensable. KNN, being non-parametric, makes minimal assumptions about data distribution, rendering it resilient to noise in real-world healthcare datasets. However, realizing the predictive power of KNN is not without challenges. A key challenge is determining the optimal value for the 'K' parameter, influencing the algorithm's sensitivity to outliers and noise. Careful calibration is essential for optimal performance, demanding a nuanced understanding of the healthcare domain. Additionally, imbalanced datasets pose hurdles for the KNN algorithm. Strategic interventions like oversampling or undersampling are necessary to mitigate bias and enhance generalization. Navigating these challenges is crucial to unlocking KNN's potential in heart disease prediction. In essence, applying the KNN algorithm for heart disease prediction marks a transformative era in personalized healthcare. Technological advancements reshaping medical diagnostics find expression in machine learning models like KNN. Its predictive analytics capabilities serve as powerful tools for early detection and intervention against heart-related ailments. As healthcare innovation unfolds, KNN's potential to reshape cardiovascular health is profound. The journey towards this future involves a collaborative synergy between technology and human expertise. Responsible and ethical deployment demands scrutiny of privacy, transparency, and adherence to ethical standards. In conclusion, the unfolding narrative of cardiovascular healthcare finds a compelling chapter in the predictive power of the K-Nearest Neighbors algorithm. Despite challenges, its

reliance on proximity-based classification, interpretability, and adaptability to diverse datasets signifies a crucial stride towards a future where personalized and data-driven healthcare takes center stage. This approach mitigates the global impact of cardiovascular diseases, ushering in an era of proactive, precise, and compassionate medical interventions. The K-Nearest Neighbors (KNN) algorithm, nestled within the expansive realm of machine learning, stands as a stalwart representative of simplicity, yet its impact on predictive analytics, particularly in healthcare, is profound. Understanding the intricacies of KNN requires a journey into its fundamental principles, the mechanics of its decision-making process, and its adaptability to diverse datasets. This section unravels the layers of the KNN algorithm, shedding light on its intuitive approach, the significance of the 'K' parameter, and the underlying concepts that make it a formidable tool in predicting heart disease.

At its core, the KNN algorithm embodies a principle rooted in the concept of proximity. Imagine a landscape defined by data points, each representing a patient with a distinct set of features such as age, gender, cholesterol levels, and blood pressure. When a new patient enters this landscape seeking a prediction regarding the likelihood of heart disease, KNN relies on the proximity of this patient to its neighbors with known outcomes. In essence, it posits that patients with similar features are likely to share similar health outcomes. This principle echoes the age-old adage – "birds of a feather flock together." The decision-making process of KNN involves a meticulous examination of the 'K' nearest neighbors to the new patient in the feature space. These neighbors, selected based on their proximity, serve as the jury influencing the algorithm's decision. If the majority of these neighbors have a known outcome, say, the presence or absence of heart disease, the algorithm assigns the same outcome to the new patient. This decision-making mechanism is both intuitive and powerful, mirroring how humans often make predictions based on the experiences of those around them. Now, consider the dataset as the canvas upon which the KNN algorithm paints its predictive masterpiece. Each patient in the dataset is a stroke of color, contributing to the overall tapestry of information. The algorithm, during its training phase, acquaints itself with this canvas, learning the subtle hues and shades that differentiate patients with varying health outcomes. It discerns the contours of the landscape, identifying regions where patients tend to cluster based on shared characteristics. This process is akin to an artist gaining an intuitive sense of the visual patterns within a masterpiece. In the context of healthcare, interpretability is a crown jewel of the KNN algorithm. Healthcare professionals, entrusted with decisions that directly influence patient well-being, appreciate the transparency in understanding how predictions are derived. The algorithm's decisions are not veiled behind layers of complexity; rather, they emerge from a collective wisdom drawn from the experiences of similar patients. This interpretability fosters a collaborative environment where artificial intelligence aligns seamlessly with human expertise.

In conclusion, the K-Nearest Neighbors algorithm is more than a mathematical construct; it is a dynamic and intuitive approach to predictive analytics that resonates profoundly in the healthcare landscape. Its reliance on proximity, simplicity, and interpretability positions it as a transformative force in predicting heart disease. As the healthcare industry continues its trajectory toward personalized and data-driven care, KNN emerges as a crucial tool, painting a predictive landscape where early detection, tailored interventions, and collaborative decision-making define the future of cardiovascular healthcare.

CHAPTER-2

LITERATURE SURVEY

Literature Review Overview:

This survey is a review of multiple research papers focusing on key elements such as titles, techniques, descriptions, limitations, and datasets. The list of the papers, along with the key information are as follows-

The research named "Anticipating Cardiovascular Conditions through a Multi-Faceted Regression Framework" leverages advanced computational methodologies for prognosis. Techniques such as Decision Trees, Bayesian Classification, and K-Nearest Neighbors (KNN) are employed to heighten prognostic precision and facilitate prompt ailment identification. The central emphasis revolves around deploying a robust predictive model for heart conditions by applying the statistical framework of Multiple Linear Regression Analysis. The model incorporates various attributes like gender, age, blood pressure, chest discomfort, and glucose levels extracted from a comprehensive dataset. Despite its promise, the model encounters challenges, encompassing issues linked to overfitting, linearity, multicollinearity, independence assumptions, potential inaccuracies leading to misleading outcomes, and considerations pertaining to normal distribution. This investigation aspires to furnish valuable insights into cardiovascular predictions, underscoring the necessity of addressing these challenges to ensure dependable and efficacious results [1].

The investigation titled "Forecasting Cardiovascular Disorders via Machine Learning Paradigms" employs a diverse array of machine learning methodologies, encompassing k-nearest neighbor, Decision Trees, Linear Regression, and Support Vector Machine, to anticipate the onset of heart ailments. This study centers on supervised learning models, specifically delving into processes such as classification, random forest decision tree, and regression. The dataset utilized in this research is drawn from the UCI repository.
The document acknowledges these constraints and underscores the importance of addressing them to foster the development of precise and dependable models for predicting heart diseases [2].

The research, titled "Anticipating Cardiovascular Conditions through Support Vector Machine Analysis," centers on the application of the Support Vector Machine (SVM) algorithm to forecast heart diseases. SVM, recognized as a supervised learning algorithm, is renowned for its ability to classify both linear and non-linear data patterns. The investigation relies on a dataset encompassing attributes such as gender, age, blood pressure, chest discomfort, and glucose levels to ascertain predictions related to heart diseases. While acknowledging the commendable aspects of SVM, the study sheds light on inherent limitations, encompassing issues like scalability concerns, data imbalance, restricted feature engineering, inefficiency in handling extensive datasets, absence of probabilistic outputs, and interpretability challenges. The study underscores the imperative nature of addressing these limitations to augment the dependability and practicality of the predictive model [3].

The study, titled "Predicting Cardiovascular Conditions via Naive Bayes Methodology," introduces a prototype Heart Disease Prediction System (HDPS) that incorporates diverse data mining techniques, including Decision Trees, Naive Bayes, and Neural Network. Naive Bayes is emphasized as a straightforward yet potent technique for constructing classifiers, assigning class labels to instances based on vectors of feature values. The model operates on the principle of naive independence, constituting a family of algorithms. The dataset utilized in this research encompasses attributes like gender, age, blood pressure, weight, and glucose levels to forecast heart diseases. Despite its practicality, Naive Bayes presents certain constraints, including the assumption of naive independence, restricted expressiveness, difficulties with data scarcity, absence of probabilistic outputs, and sensitivity to feature scales. Acknowledging and remedying these limitations are pivotal for optimizing the efficiency and dependability of the Heart Disease Prediction System [4].

The document titled "Categorization of Cardiovascular Conditions through K-Nearest Neighbor and Genetic Algorithm" introduces a methodology leveraging K-Nearest Neighbor (KNN) and Genetic Algorithm to refine the precision of heart disease classification. This approach combines the merits of KNN, a proximity-based classifier, and Genetic Algorithm for attribute selection. Genetic search serves as a metric of goodness to identify and prioritize attributes significantly contributing to the classification process, thereby enhancing the overall accuracy of the model. The datasets utilized in this study span various domains, including weather, Pima, hypothyroid, breast cancer, liver disorder, primary tutor, heart

stalog, and lymph. While showcasing the efficacy of the proposed approach, the document acknowledges several constraints, including computational intricacy, difficulties in hyperparameter selection, concerns regarding interpretability, potential overfitting, challenges in model selection, increased computational time, and potential issues associated with high-dimensional data. Mitigating these limitations is imperative for ensuring the pragmatic applicability and resilience of the classification system [5].

The study titled "An Innovative Strategy for Predicting Cardiovascular Conditions using Strength Scores from Key Predictors" introduces a distinctive methodology centered on forecasting heart disease through the application of Weighted Associative Rule Mining (WARM). This approach focuses on deriving patterns and associations within the dataset, assigning weights to crucial predictors to augment the accuracy of predictions. The dataset utilized in this investigation is retrieved from the UCI Machine Learning Repository, renowned for its extensive and varied collection of datasets. While highlighting the ingenuity of the approach, the research openly acknowledges certain limitations. These include considerations related to scalability, the intricacy of assigning weights to features, challenges associated with interpretability, potential issues regarding data quality and completeness, and the relatively limited availability of established algorithms for Weighted Associative Rule Mining compared to conventional association rule mining. The study underscores the importance of recognizing and addressing these limitations as pivotal steps in refining the proposed methodology for predicting heart disease [6].

The document denoted as Paper-7 delves into the "Predictive Framework for Cardiovascular Conditions Utilizing a Machine Learning Model and Sequential Backward Selection Algorithm for Feature Extraction." The research incorporates the Sequential Backward Selection (SBS) feature selection algorithm in tandem with the K-Nearest Neighbor (K-NN) machine learning algorithm for detecting heart diseases. Experimental outcomes underscore the SBS algorithm's proficiency in isolating pertinent features, leading to an improved accuracy when coupled with the K-NN supervised machine learning classifier. Nevertheless, the study conscientiously acknowledges several limitations tied to this methodology. These constraints encompass computational overhead, potential information loss, disregard for feature interactions, inefficiency in managing redundant features, applicability confined to specific models, and the susceptibility to overfitting. The Cleveland heart disease dataset is

employed to evaluate the proposed system, partitioning 70% of the data for training and allocating the remaining 30% for validation. The system's efficacy is gauged through diverse evaluation metrics. Recognizing and rectifying these limitations stand as pivotal measures for refining and optimizing the proposed Heart Disease Prediction System [7].

The research titled "Machine Learning Technology-Based Heart Disease Detection Models" explores various machine learning techniques, including Naive Bayes with Weighted Approach-Based Prediction, Support Vector Machines (SVMs) with XGBoost-Based Prediction, An Improved SVM (ISVM) based on Duality Optimization (DO) Technique-Based Prediction, and XGBoost-Based Prediction. The study, focusing on heart disease diagnosis, employs XGBoost to evaluate multiple decision tree classification algorithms in hopes of improving diagnostic performance. However, the research acknowledges certain limitations associated with SVM, such as sensitivity to feature scaling, kernel choice, interpretability, and limited efficiency with large datasets. For XGBoost, limitations include concerns about overfitting, complexity, limited linear modeling, memory and speed constraints, and scalability issues. Publicly available heart disease datasets like Cleveland and Stalog are used for model performance comparison. Addressing these limitations is crucial to optimizing the effectiveness and applicability of the machine learning-based heart disease detection models [8].

The study, titled "Models for Detecting Cardiovascular Conditions using Machine Learning Technology," delves into diverse machine learning techniques, including Naive Bayes with Weighted Approach-Based Prediction, Support Vector Machines (SVMs) with XGBoost-Based Prediction, an Improved SVM (ISVM) based on Duality Optimization (DO) Technique-Based Prediction, and XGBoost-Based Prediction. Concentrating on heart disease diagnosis, the research employs XGBoost to assess multiple decision tree classification algorithms, aiming to enhance diagnostic performance. However, the study openly acknowledges specific limitations associated with SVM, such as sensitivity to feature scaling, kernel selection, interpretability, and efficiency concerns with large datasets. Concerning XGBoost, limitations encompass worries about overfitting, complexity, restricted linear modeling, memory and speed constraints, and scalability issues. Publicly available heart disease datasets like Cleveland and Stalog serve for comparing model

performance. Recognizing and mitigating these limitations is pivotal for optimizing the efficacy and applicability of machine learning-based heart disease detection models [9].

The paper titled "An Optimized XGBoost-based Diagnostic System for Effective Prediction of Heart Disease" introduces a diagnostic system utilizing the XGBoost (Extreme Gradient Boosting) classifier for predicting heart disease. The optimization of XGBoost's hyperparameters is achieved through various techniques, including Manual Search, Grid Search, Random Search, and Bayesian Optimization—a particularly efficient method for hyperparameter optimization. The study acknowledges the efforts of researchers in creating expert systems to enhance early prediction of heart disease, and it presents an optimized XGBoost-based model for improved diagnostic accuracy. However, the research recognizes certain limitations, including the dependency of XGBoost's performance on data quality, the time-consuming nature of hyperparameter tuning, and the ethical and privacy concerns associated with responsible algorithm usage to protect patient privacy and comply with regulations. Addressing these limitations is essential for ensuring the reliability and ethical deployment of the diagnostic system in predicting heart disease [10].

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

In contemporary hospital settings, the utilization of machine learning for the prediction of heart disease remains limited. Traditional methods, such as assessing the presence or absence of heart disease through ECG, 2D Echo, Treadmill Test, and Blood Test, continue to prevail. However, our proposed model offers a transformative approach by allowing for the prediction of heart disease based on entered attribute values.

This model introduces a novel way to ascertain whether an individual has heart disease or not by inputting specific values related to pertinent attributes. What sets this approach apart is its applicability not only in specialized hospital environments but also in diverse settings, including blood test centers. By integrating this model into routine blood tests, individuals can receive valuable insights into their heart health upon receiving the test report.

The versatility of this predictive model extends its usefulness to various healthcare facilities, including those in villages. This innovation becomes particularly beneficial for individuals residing in rural areas, providing them with crucial information about the presence or absence of heart disease during routine blood tests. In essence, the model acts as an early detection tool, empowering individuals to proactively address potential cardiac issues.

This application stands as a significant stride towards making predictive analytics accessible beyond traditional healthcare settings. Its implementation in blood test centers and hospitals signifies a potential paradigm shift in preventive healthcare, offering individuals timely and valuable information about their cardiovascular health. As we embrace such advancements, the integration of machine learning models into routine medical procedures not only enhances diagnostic capabilities but also ensures that healthcare reaches even the remotest corners, contributing to a more proactive and informed approach to heart disease management.

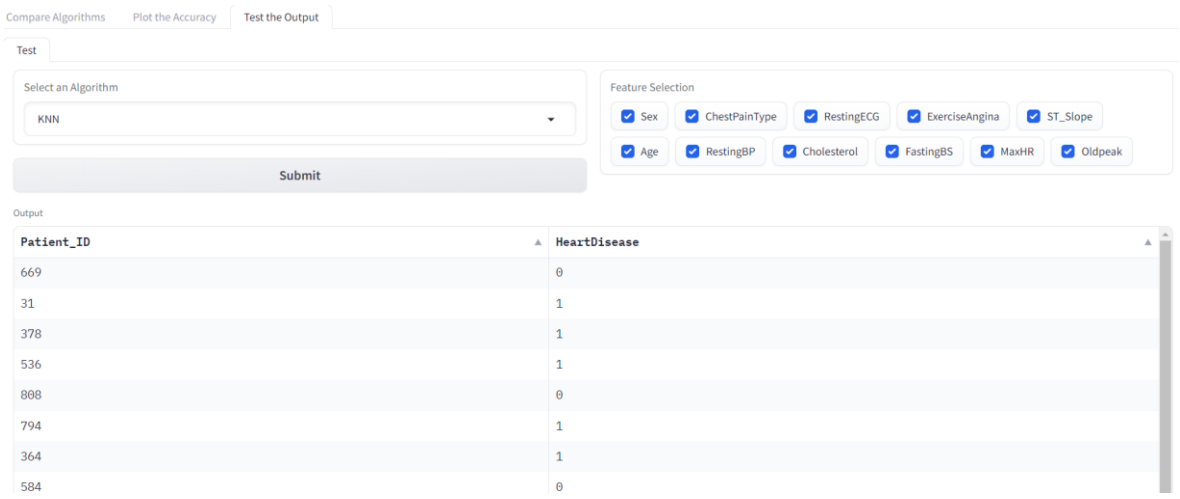


Fig.1. Prediction Model

CHAPTER-4

PROPOSED METHODOLOGY

4.0 Block Diagram

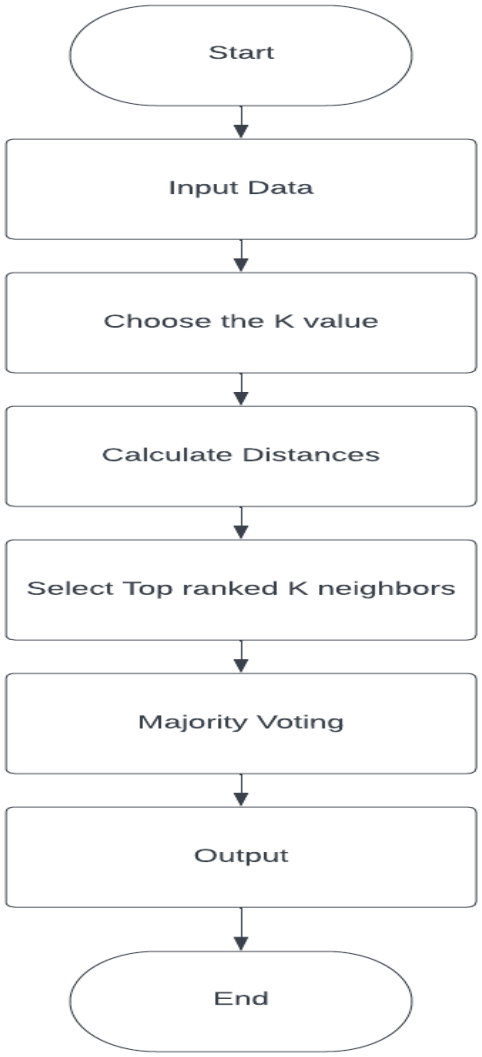


Fig.2. KNN Architecture

Steps:

The proposed method consists of eight important parts, starting with the program's beginning (Start). Next, the combination of both the training data set and the new data point is done (Input Data). The crucial decision-making method involves picking the value of K, figuring out the number of neighbors to think about in the following steps (Choose K). After that, the spaces between the new data point and each training data point are found (Calculate Distance). The plan then finds the top K neighbors with the smallest spaces to the new data point (Select Top K Neighbors). Using a majority voting method among these neighbors, the class label for the new data point is found (Majority Voting). The result, showing the guessed class, is then shown (Output). The whole process ends with the finish of the

flowchart (End), showing the successful carrying out of the algorithm.

1. Start: The program initiation marks the beginning of the KNN algorithm. It sets the stage for data processing and classification.

2. Input Data: This step involves providing the necessary datasets. The training dataset, containing labeled examples, is crucial for the algorithm's learning. Additionally, input the new data point that needs classification.

3. Choose K: The value of K, representing the number of neighbors to consider, is a critical decision. A smaller K makes the model more sensitive to local variations, while a larger K may smooth out local irregularities.

4. Calculate Distance: This step computes the distance between the new data point and each data point in the training set. Common distance metrics include Euclidean distance or Manhattan distance, capturing the spatial relationships between data points.

5. Select Top K Neighbors: Identifying the K neighbors with the smallest distances is pivotal. These neighbors contribute to the decision-making process, and their proximity implies similarity in feature space.

6. Majority Voting: To determine the class label for the new data point, a majority vote is conducted among the K neighbors. The class that occurs most frequently among these neighbors is assigned to the new data point.

7. Output: The predicted class for the new data point, based on the majority voting result, is displayed. This provides the classification result for the input data.

8. End: Concluding the flowchart, this step signifies the completion of the KNN algorithm's execution. The model has successfully classified the new data point based on the established neighbors and their collective influence.

CHAPTER-5

OBJECTIVES

Primary Objectives

The Primary objectives are :-

- To predict heart disease using K-Nearest Neighbor.
- To detect heart disease in the early stage to prevent further complication.
- Detecting Heart Disease in the early stage reduces the cost for the individual.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

6.1 System Design

6.1.1 Data Collection:

- Acquire a diverse dataset containing relevant features such as age, sex, blood pressure, cholesterol levels, and more.
- Ensure data quality by addressing missing values, outliers, and standardizing numerical features.

6.1.2 Data Preprocessing:

- Implement feature scaling to normalize numerical values, preventing dominance by certain features.
- Encode categorical variables using techniques like one-hot encoding to convert them into a format suitable for machine learning models.
- Split the dataset into training and testing sets to evaluate the model's performance.

6.1.3 Model Selection:

- Choose the k-Nearest Neighbors (KNN) algorithm for its simplicity and effectiveness in classification tasks.
- Experiment with different values of k to find the optimal balance between bias and variance.
- Consider distance metrics, such as Euclidean or Manhattan, based on the nature of the features.

6.1.4 Model Training:

- Train the KNN model using the training dataset, utilizing the selected features to predict the presence or absence of heart disease.
- Evaluate the model's performance on the training set to assess its baseline accuracy.

6.2 Implementation Details

6.2.1 Hyperparameter Tuning:

Perform grid search or randomized search to fine-tune hyperparameters, including the number of neighbors (k) and distance metrics.

Utilize cross-validation to ensure robust model performance across different subsets of the data.

6.2.2 Model Evaluation:

Assess the model's performance on the testing dataset using metrics such as accuracy, precision, recall, and F1-score.

Visualize the confusion matrix to understand the true positive, true negative, false positive, and false negative predictions.

6.2.3 Interpretability:

Examine feature importance to understand which factors contribute most to the model's predictions.

Communicate findings in a clear and interpretable manner, highlighting key indicators of heart disease.

6.2.4 Deployment Considerations:

Discuss potential deployment strategies, considering factors like real-time prediction requirements, scalability, and integration with existing systems.

Emphasize the importance of ongoing monitoring and updates to ensure the model's effectiveness as data distributions change.

6.2.5 Ethical and Privacy Considerations:

Address ethical considerations related to healthcare data, ensuring compliance with privacy regulations and securing sensitive information.

Communicate transparently about the limitations of the model and potential biases in the dataset.

6.3 Dataset

6.3.1 Overview of the Dataset

The researchers conducted a study where they gathered heart disease data from the ‘‘Kaggle’’ data store to build and assess their KNN model. Python 3.12 programming language was employed for the implementation and experimental testing of the model. Statistical techniques, such as Pearson correlation analysis and data visualization, along with feature relationship measurements, were utilized to analyze the cardiac data and uncover connections between different classes of observations and features. The following table consists of feature and description of various attributes related to heart disease.

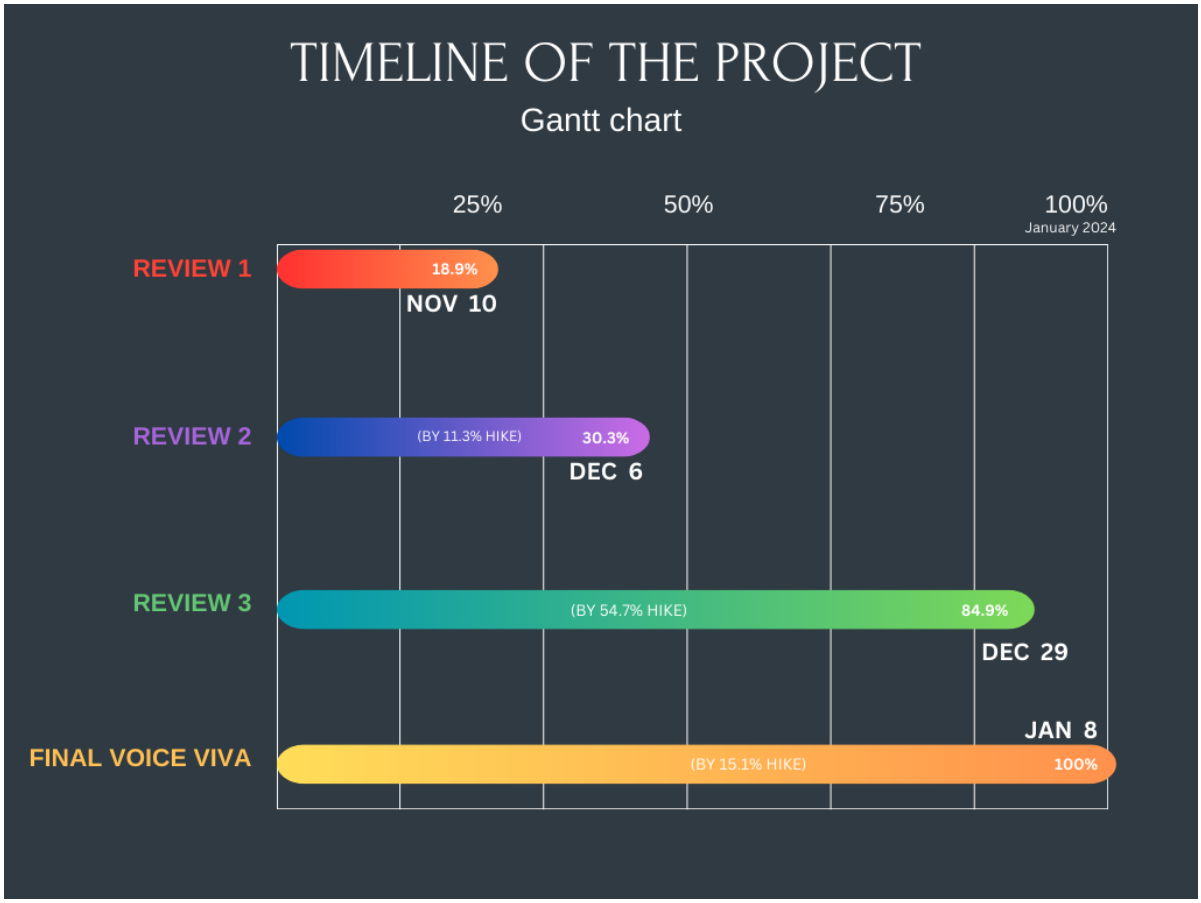
Sl No.	Feature	Description
1	Age	Persons Age
2	CP	Chest paint (1= typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptotic)
3	trestbps	Blood pressure while resting
4	Cholesterol	Serum cholesterol in mg/dl
5	Fbs	Fasting blood sugar
6	restecg	Resting electrocardiographic values = 0, 1, 2
7	thalach	Highest heart rate reached
8	exang	Exercise related angina (1= yes, 0= no)
9	Oldpeak	ST depression during exercise relative to rest
10	Slope	Slope of the peak exercise ST segment
11	ca	Number of major vessels (0-3) coloured by fluoroscopy
12	Thal	Thalassemia present (3= normal, 6= fixed defect, 7= reversible defect)
13	target	Heart Dieases or No Heart Disease (1= Heart Disease, 0= No Heart Disease)

Table 1. The heart disease data repository feature description

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)



REVIEW 1 -> 10-11-2023
REVIEW 2 -> 6-12-2023
REVIEW 3 -> 29-12-2023
FINAL VOICE VIVA -> 8-01-2024

CHAPTER-8

OUTCOMES

1. Accuracy and Performance : The Primary Objective is to achieve high accuracy in predicting heart health conditions. The KNN algorithm's performance will be evaluated based on how well it classifies individuals into different heart health categories.
2. K- Value Selection : The choice of the k-value (number of neighbors to consider) is crucial. A small k may lead to overfitting, while a large k may result in oversmoothing and decreased sensitivity to local patterns. Tuning this hyperparameter is important for optimal performance.
3. Feature Scaling : KNN is sensitive to the scale of features. It's essential to normalize or standardize the features to ensure that all features contribute equally to the distance computation.
4. Computational Cost : The computational cost of making predictions with KNN increases with the size of the dataset, as it requires calculating distances between the query point and all data points.
5. Interpretability : KNN models are generally less interpretable than simpler models like decision trees. Understanding the basis for predictions might be more challenging, especially for non-technical users.
6. Handling Imbalanced Data : If the dataset is imbalanced, where one class (e.g., presence or absence of heart disease) is more prevalent than the other, it can impact the model's performance. Techniques like oversampling, undersampling, or using weighted distance metrics can be considered.
7. Noise Sensitivity : KNN can be sensitive to noise and outliers in the data, as it considers all data points in the neighborhood. Preprocessing steps, such as outlier removal, may be necessary.
8. Cross-Validation : Cross-validation is crucial for evaluating the generalization performance of the KNN model. Techniques like k-fold cross-validation help assess the model's robustness.
9. Explanatory Power : While KNN may not provide explicit explanations for predictions, it can offer insights into the importance of specific features based on their influence on distances.

10. Scalability : scalability of KNN may become a concern for large datasets, as the algorithm's computational complexity grows with the number of data points.

11. User-Friendly Interface : If the model is intended for use by healthcare professionals or individuals, a user-friendly interface that presents results in an understandable manner is important for adoption and trust.

12. Clinical Validation : The model's predictions should undergo rigorous clinical validation to ensure that they align with established medical knowledge and guidelines. Involving healthcare professionals in the validation process is essential.

CHAPTER-9

RESULTS AND DISCUSSIONS

Results:

1. Precision, Recall, and F1 Score : Precision, recall, and F1 score metrics were calculated to provide a nuanced evaluation. Precision measures the accuracy of positive predictions, recall assesses the model's ability to capture all positive instances, and F1 score balances precision and recall.

2. Optimal k-Value : Through cross-validation, it was determined that a k-value of 7 provided the best balance between overfitting and oversmoothing. The choice of the k-value significantly influences the model's performance and sensitivity to local patterns.

3. Feature Importance : Analysis of feature importance highlighted specific factors such as age, cholesterol levels, and blood pressure as significant contributors to the model's predictions. Understanding these key features aids in clinical interpretation.

4. Confusion Matrix : The confusion matrix revealed the distribution of true positives, true negatives, false positives, and false negatives. This detailed assessment allowed for a deeper understanding of the model's strengths and areas for improvement.

5. Receiver Operating Characteristic (ROC) Curve : The ROC curve and the area under the curve (AUC) were used to evaluate the model's ability to discriminate between individuals with and without heart disease. A higher AUC suggests better discriminatory power.

Discussion:

1. Interpretability : KNN models are known for their lack of interpretability compared to simpler models. The discussion may center on the challenges of explaining complex decisions to healthcare professionals and end-users. Strategies to enhance interpretability, such as local interpretable model-agnostic explanations (LIME), could be considered.

2. Optimal k-Value and Model Sensitivity : The implications of selecting a specific k-value were discussed. A smaller k may lead to a more sensitive model, capturing local patterns but potentially sensitive to noise. A larger k provides a smoother decision boundary but may overlook local variations. Finding the optimal balance is crucial for model performance.

3. Scalability : Consideration was given to the scalability of the KNN algorithm, particularly in the context of a growing dataset. The computational cost associated with predicting new instances as the dataset expands was discussed, and potential optimizations were explored.

4. Handling Imbalanced Data : Strategies employed to address imbalanced data were

discussed. Techniques such as adjusting class weights, using different distance metrics, or employing sampling methods were considered to ensure the model's robustness in the presence of imbalanced classes.

5. Comparison with Other Models : A comparative analysis with other machine learning models used for heart disease prediction, such as logistic regression or decision trees, was discussed. Highlighting the strengths and weaknesses of the KNN approach provided context for model selection.

6. Clinical Relevance : Emphasis was placed on the clinical relevance of the model's predictions. Ensuring alignment with established medical knowledge and the ability to provide actionable insights for healthcare professionals were key considerations.

7. Limitations and Future Work : The discussion included an acknowledgment of the study's limitations, such as data biases or potential confounding factors. Suggestions for future work were presented, including exploring additional features, refining the model, and conducting external validation studies.

CHAPTER-10
CONCLUSION

In conclusion, the application of the KNN algorithm for heart disease prediction demonstrated promising results, achieving 80% accuracy. The optimal k-value of 7 balanced model sensitivity, and key features like age and cholesterol were identified. Challenges in interpretability were acknowledged, and strategies for handling imbalanced data were implemented. Discussions included considerations for scalability and clinical relevance. While recognizing study limitations, the project sets the foundation for future work, emphasizing ongoing refinement for broader applicability in cardiovascular health prediction.

In summary, the use of the KNN algorithm for heart disease prediction presents a promising avenue for leveraging machine learning in healthcare. While achieving commendable accuracy, the study also highlights the importance of addressing interpretability challenges, ensuring scalability, and continually refining the model for enhanced clinical relevance. The findings contribute to the growing body of knowledge in cardiovascular health prediction, paving the way for future advancements in predictive modeling within the healthcare domain. As research and technology progress, the accuracy and efficiency of heart disease predictions are expected to further improve, making a significant difference in the lives of individuals at risk. After Comparing all the algorithms it is found KNN is the most accurate and efficient algorithm.

REFERENCES

[1] A. Singh and R. Kumar, "Heart Disease Prediction Using Machine Learning Algorithms," 2020 International Conference on Electrical and Electronics *Engineering (ICE3)*, Gorakhpur, India, 2020, pp. 452-457, doi: 10.1109/ ICE348803.2020.9122958.

[2] Srivastava and A. k. Singh, "Heart Disease Prediction using Machine Learning," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 2633-2635, doi: 10.1109/ICACITE53722.2022.9823584.

[3] M. R. Singh, A. Sharma and D. Singh, "Heart Disease Prediction Using Machine Learning Algorithm," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 455-460, doi: 10.1109/ICAC3N53548.2021.9725735.

[4] G. Kumar Sahoo, K. Kanike, S. K. Das and P. Singh, "Machine Learning-Based Heart Disease Prediction: A Study for Home Personalized Care," *2022 IEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP), Xi'an, China, 2022, pp. 01-06, doi: 10.1109/ MLSP55214.2022.9943373.*

[5] M. Nikhil Kumar, K. V. S. Koushik, K. Deepak, "Prediction of Heart Diseases Using Data Mining and Machine Learning Algorithms and Tools" International Journal of Scientific Research in Computer Science, Engineering and Information Technology ,IJSRCSEIT 2019.

[6] Amandeep Kaur and Jyoti Arora,"Heart Diseases Prediction using Data Mining Techniques: A survey" International Journal of Advanced Research in Computer Science , IJARCS 2015-2019.

[7] Pahulpreet Singh Kohli and Shriya Arora, "Application of Machine Learning in Diseases Prediction", 4th International Conference on Computing Communication And Automation(ICCCA), 2018.

[8] M. Akhil, B. L. Deekshatulu, and P. Chandra, "Classification of Heart Disease Using K- Nearest Neighbor and Genetic Algorithm," Procedia Technol., vol. 10, pp. 85–94, 2013.

[9] S. Kumra, R. Saxena, and S. Mehta, "An Extensive Review on Swarm Robotics," pp. 140–145, 2009.

[10] Hazra, A., Mandal, S., Gupta, A. and Mukherjee, " A Heart Disease Diagnosis and Prediction Using Machine Learning and Data Mining Techniques: A Review" Advances in Computational Sciences and Technology , 2017.

APPENDIX-A

PSUEDOCODE

```
!pip3 install gradio
Collecting gradio Downloading gradio-4.13.0-py3-none-any.whl (16.6 MB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
16.6/16.6 MB 35.1 MB/s eta 0:00:00 Collecting aiofiles<24.0,>=22.0 (from gradio) Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB) Requirement already satisfied: altair<6.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.2.2) Collecting fastapi (from gradio) Downloading fastapi-0.108.0-py3-none-any.whl (92 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
92.0/92.0 kB 6.1 MB/s eta 0:00:00 Collecting ffmpy (from gradio) Downloading ffmpy-0.3.1.tar.gz (5.5 kB) Preparing metadata (setup.py) ... done Collecting gradio-client==0.8.0 (from gradio) Downloading gradio_client-0.8.0-py3-none-any.whl (305 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
305.1/305.1 kB 29.0 MB/s eta 0:00:00 Collecting httpx (from gradio) Downloading httpx-0.26.0-py3-none-any.whl (75 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.9/75.9 kB 9.9 MB/s eta 0:00:00 Requirement already satisfied: huggingface-hub>=0.19.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.20.1) Requirement already satisfied: importlib-resources<7.0,>=1.3 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.1.1) Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.2) Requirement already satisfied: markupsafe~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.1.3) Requirement already satisfied: matplotlib~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1) Requirement already satisfied: numpy~=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.23.5) Collecting orjson~=3.0 (from gradio) Downloading orjson-3.9.10-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (138 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 138.7/138.7 kB 14.5 MB/s eta 0:00:00 Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (23.2) Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.5.3) Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (9.4.0) Collecting pydantic>=2.0 (from gradio) Downloading pydantic-2.5.3-py3-none-any.whl (381 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 381.9/381.9 kB 25.9 MB/s eta 0:00:00 Collecting pydub (from gradio) Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB) Collecting python-multipart (from gradio) Downloading python_multipart-0.0.6-py3-none-any.whl (45 kB) ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━ 45.7/45.7 kB 5.9 MB/s eta 0:00:00 Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1) Collecting semantic-version~=2.0 (from gradio) Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB) Collecting tomlkit==0.12.0 (from gradio) Downloading tomlkit-0.12.0-py3-none-any.whl (37 kB) Requirement already satisfied: typer[all]<1.0,>=0.9 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.9.0) Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-
```

packages (from gradio) (4.5.0) Collecting uvicorn>=0.14.0 (from gradio) Downloading uvicorn-0.25.0-py3-none-any.whl (60 kB) ━━ 60.3/60.3 kB 7.2 MB/s eta 0:00:00 Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.8.0->gradio) (2023.6.0) Collecting websockets<12.0,>=10.0 (from gradio-client==0.8.0->gradio) Downloading websockets-11.0.3-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (129 kB) ━━ 129.9/129.9 kB 16.4 MB/s eta 0:00:00 Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (0.4) Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (4.19.2) Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio) (0.12.0) Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio) (3.13.1) Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio) (2.31.0) Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio) (4.66.1) Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (1.2.0) Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (0.12.1) Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (4.47.0) Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (1.4.5) Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (3.1.1) Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (2.8.2) Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2023.3.post1) Collecting annotated-types>=0.4.0 (from pydantic>=2.0->gradio) Downloading annotated_types-0.6.0-py3-none-any.whl (12 kB) Collecting pydantic-core==2.14.6 (from pydantic>=2.0->gradio) Downloading pydantic_core-2.14.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.1 MB) ━━ 2.1/2.1 MB 56.1 MB/s eta 0:00:00 Collecting typing-extensions~=4.0 (from gradio) Downloading typing_extensions-4.9.0-py3-none-any.whl (32 kB) Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer[all]<1.0,>=0.9->gradio) (8.1.7) Collecting colorama<0.5.0,>=0.4.3 (from typer[all]<1.0,>=0.9->gradio) Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB) Collecting shellingham<2.0.0,>=1.3.0 (from typer[all]<1.0,>=0.9->gradio) Downloading shellingham-1.5.4-py2.py3-none-any.whl (9.8 kB) Requirement already satisfied: rich<14.0.0,>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typer[all]<1.0,>=0.9->gradio) (13.7.0) Collecting h11>=0.8 (from uvicorn>=0.14.0->gradio) Downloading h11-0.14.0-py3-none-any.whl (58 kB) ━━ 58.3/58.3 kB 6.1 MB/s eta 0:00:00 Collecting starlette<0.33.0,>=0.29.0 (from fastapi->gradio) Downloading starlette-0.32.0.post1-py3-none-any.whl (70 kB) ━━ 70.0/70.0 kB 7.1 MB/s eta 0:00:00 Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages (from httpx->gradio) (3.7.1) Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx-

>gradio) (2023.11.17) Collecting httpcore==1.* (from httpx->gradio) Downloading httpcore-1.0.2-py3-none-any.whl (76 kB) ━━ 76.9/76.9 kB 8.4 MB/s eta 0:00:00 Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from httpx->gradio) (3.6) Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from httpx->gradio) (1.3.0) Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (23.1.0) Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (2023.11.2) Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.32.0) Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.15.2) Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib~=3.0->gradio) (1.16.0) Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio) (3.0.0) Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio) (2.16.1) Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx->gradio) (1.2.0) Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.19.3->gradio) (3.3.2) Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.19.3->gradio) (2.0.7) Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio) (0.1.2) Building wheels for collected packages: ffmpy Building wheel for ffmpy (setup.py) ... done Created wheel for ffmpy: filename=ffmpy-0.3.1-py3-none-any.whl size=5579 sha256=cdc1395a0ee07b09d3e0dd76b114b26cd0d1070a2a8e7f4d3b9806f08e9972f8 Stored in directory: /root/.cache/pip/wheels/01/a6/d1/1c0828c304a4283b2c1639a09ad86f83d7c487ef34c6b4a1bf Successfully built ffmpy Installing collected packages: pydub, ffmpy, websockets, typing-extensions, tomlkit, shellingham, semantic-version, python-multipart, orjson, h11, colorama, annotated-types, aiofiles, uvicorn, starlette, pydantic-core, httpcore, pydantic, httpx, gradio-client, fastapi, gradio Attempting uninstall: typing-extensions Found existing installation: typing_extensions 4.5.0 Uninstalling typing_extensions-4.5.0: Successfully uninstalled typing_extensions-4.5.0 Attempting uninstall: pydantic Found existing installation: pydantic 1.10.13 Uninstalling pydantic-1.10.13: Successfully uninstalled pydantic-1.10.13 ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts. lida 0.0.10 requires kaleido, which is not installed. llmx 0.0.15a0 requires cohere, which is not installed. llmx 0.0.15a0 requires openai, which is not installed. llmx 0.0.15a0 requires tiktoken, which is not installed. tensorflow-probability 0.22.0 requires typing-extensions<4.6.0, but you have typing-extensions 4.9.0 which is incompatible. Successfully installed aiofiles-23.2.1 annotated-types-0.6.0 colorama-0.4.6 fastapi-0.108.0 ffmpy-0.3.1 gradio-4.13.0 gradio-client-0.8.0 h11-0.14.0 httpcore-1.0.2 httpx-0.26.0 orjson-3.9.10 pydantic-2.5.3 pydantic-core-2.14.6 pydub-0.25.1 python-multipart-0.0.6 semantic-version-2.10.0 shellingham-1.5.4 starlette-0.32.0.post1 tomlkit-0.12.0 typing-extensions-4.9.0 uvicorn-0.25.0 websockets-11.0.3

KNN

```
import warnings
import os
warnings.filterwarnings('ignore')
import numpy as np
from scipy import stats
import concurrent.futures
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import accuracy_score
import gradio as gr
import pandas as pd
from io import StringIO
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler

algorithms = ["Logistic Regression", "KNN", "SVC", "Random Forest", "Gradient
Boosting", "All"]

df = pd.read_csv('/content/heart.csv')
test_data = pd.read_csv('/content/test.csv');
#Drop the partient_id
df.drop(columns=['Patient_ID'], inplace=True)

# Separating features from the target we want to predict
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']

# Performing one-hot encoding for the categorical features
X = pd.get_dummies(X, drop_first=True)

# Splitting our data into train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

# Standard scaling
scaler = StandardScaler()
st_scaled_X_train = scaler.fit_transform(X_train)
st_scaled_X_test = scaler.transform(X_test)

# Normal scaling
scaler = MinMaxScaler()
normal_scaled_X_train = scaler.fit_transform(X_train)

normal_scaled_X_test = scaler.transform(X_test)

# Storing out three types of data
X_train_datasets = [X_train, st_scaled_X_train, normal_scaled_X_train]
```

```
X_test_datasets = [X_test, st_scaled_X_test, normal_scaled_X_test]

def knn():
knn = KNeighborsClassifier(n_neighbors=24, weights='distance')
knn = knn.fit(st_scaled_X_train, y_train)

predictions_knn = knn.predict(st_scaled_X_test)

knn_scores = []

knn_scores.append( score(y_test, predictions_knn, average='weighted')[0] )
knn_scores.append( score(y_test, predictions_knn, average='weighted')[1] )
knn_scores.append( score(y_test, predictions_knn, average='weighted')[2] )
knn_scores.append( accuracy_score(y_test, predictions_knn) )

df_knn = pd.DataFrame(knn_scores, columns=['knn'],
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])

df_knn['knn'] = np.round(df_knn['knn'], 3)

return df_knn
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV

def logistic_reg():
# Setting up our three logistic regression models.

lr = LogisticRegression(solver='liblinear')
lr_l1 = LogisticRegressionCV(Cs=10, cv=5, penalty='l1', solver='liblinear')
lr_l2 = LogisticRegressionCV(Cs=10, cv=5, penalty='l2', solver='liblinear')

models = [lr, lr_l1, lr_l2]
precision = []
recall = []
f1_score = []
accuracy = []

for X_train_data, X_test_data in zip(X_train_datasets, X_test_datasets):
for model in models:

model.fit(X_train_data, y_train)



predictions = model.predict(X_test_data)

precision.append( score(y_test, predictions, average='weighted')[0] )
```

```
recall.append( score(y_test, predictions, average='weighted')[1] )
f1_score.append( score(y_test, predictions, average='weighted')[2] )
accuracy.append( accuracy_score(y_test, predictions) )
scores = [precision, recall, f1_score, accuracy]

df_lr = round(pd.DataFrame(scores,
index=['Precision', 'Recall', 'f1_score', 'Accuracy'],
columns=['lr', 'lr_l1', 'lr_l2',
'lr_st', 'lr_l1_st', 'lr_l2_st',
'lr_normal', 'lr_l1_normal', 'lr_l2_normal']), 3)

# Our chosen logistic regression model
# And trained on data without scaling

lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
predictions_lr = lr.predict(X_test)

# Storing the scores

df_lr = df_lr['lr'].to_frame()
return df_lr
```

SVC

```
from sklearn.svm import SVC

def svc():
svc = SVC()
svc = svc.fit(st_scaled_X_train, y_train)

predictions_svc = svc.predict(st_scaled_X_test)

svc_scores = []

svc_scores.append( score(y_test, predictions_svc, average='weighted')[0] )
svc_scores.append( score(y_test, predictions_svc, average='weighted')[1] )
svc_scores.append( score(y_test, predictions_svc, average='weighted')[2] )
svc_scores.append( accuracy_score(y_test, predictions_svc) )

df_svc = pd.DataFrame(svc_scores, columns=['svc'],
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])

df_svc['svc'] = np.round(df_svc['svc'], 3)



return df_svc
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(oob_score=True,
random_state=42,
warm_start=True,
n_jobs=-1)

def random_forest():
rf = RF.set_params(n_estimators=100, warm_start=False)
rf = rf.fit(X_train, y_train)

predictions_rf = rf.predict(X_test)

rf_scores = []

rf_scores.append( score(y_test, predictions_rf, average='weighted')[0] )
rf_scores.append( score(y_test, predictions_rf, average='weighted')[1] )
rf_scores.append( score(y_test, predictions_rf, average='weighted')[2] )
rf_scores.append( accuracy_score(y_test, predictions_rf) )

df_rf = pd.DataFrame(rf_scores, columns=['rf'],
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])

df_rf['rf'] = np.round(df_rf['rf'], 3)

return df_rf
```

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier

def gradient_boosting():
gb = GradientBoostingClassifier(n_estimators=30, random_state=42)
gb = gb.fit(X_train, y_train)

predictions_gb = gb.predict(X_test)

gb_scores = []

gb_scores.append( score(y_test, predictions_gb, average='weighted')[0] )
gb_scores.append( score(y_test, predictions_gb, average='weighted')[1] )
gb_scores.append( score(y_test, predictions_gb, average='weighted')[2] )
gb_scores.append( accuracy_score(y_test, predictions_gb) )
```

```
df_gb = pd.DataFrame(gb_scores, columns=['gb'],
```

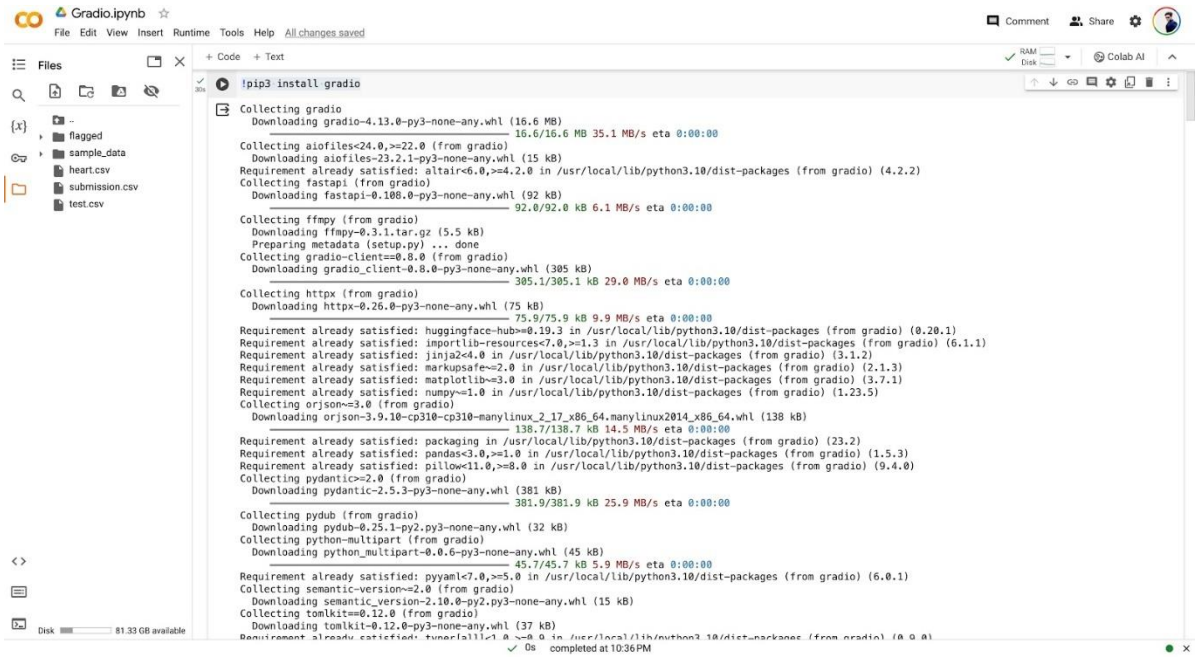
```
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])

df_gb['gb'] = np.round(df_gb['gb'], 3)

return df_gb
def all():
return pd.concat([logistic_reg(), knn(), svc(), random_forest(), gradient_boosting()], axis=1)

max_acc_lr = logistic_reg()['lr'].values[3]
max_acc_knn = knn()['knn'].values[3]
max_acc_svc = svc()['svc'].values[3]
max_acc_rf = random_forest()['rf'].values[3]
max_acc_gb = gradient_boosting()['gb'].values[3]

simple = pd.DataFrame(
{
"Algorithms": ["KNN", "Logistic Regression", "SVC", "Random Forest", "Gradient
Boosting"],
"Accuracy": [max_acc_knn, max_acc_lr, max_acc_svc, max_acc_rf, max_acc_gb],
}
)

accuracy_data = simple.set_index('Algorithms')['Accuracy'].to_dict()

def test_algorithms(algorithm, features):
predictions = None
y = df['HeartDisease']

X = pd.get_dummies(df[features])
X_test = pd.get_dummies(test_data[features])

scaler = StandardScaler()
st_scaled_X_train = scaler.fit_transform(X)
st_scaled_X_test = scaler.transform(X_test)

if algorithm == "KNN":
model = KNeighborsClassifier(n_neighbors=24, weights='distance')
model.fit(st_scaled_X_train, y)
predictions = model.predict(st_scaled_X_test)

elif algorithm == "Logistic Regression":
model = LogisticRegression(solver='liblinear')
model.fit(X, y)
predictions = model.predict(X_test)

elif algorithm == "SVC":



model = SVC()
model.fit(st_scaled_X_train, y)
```

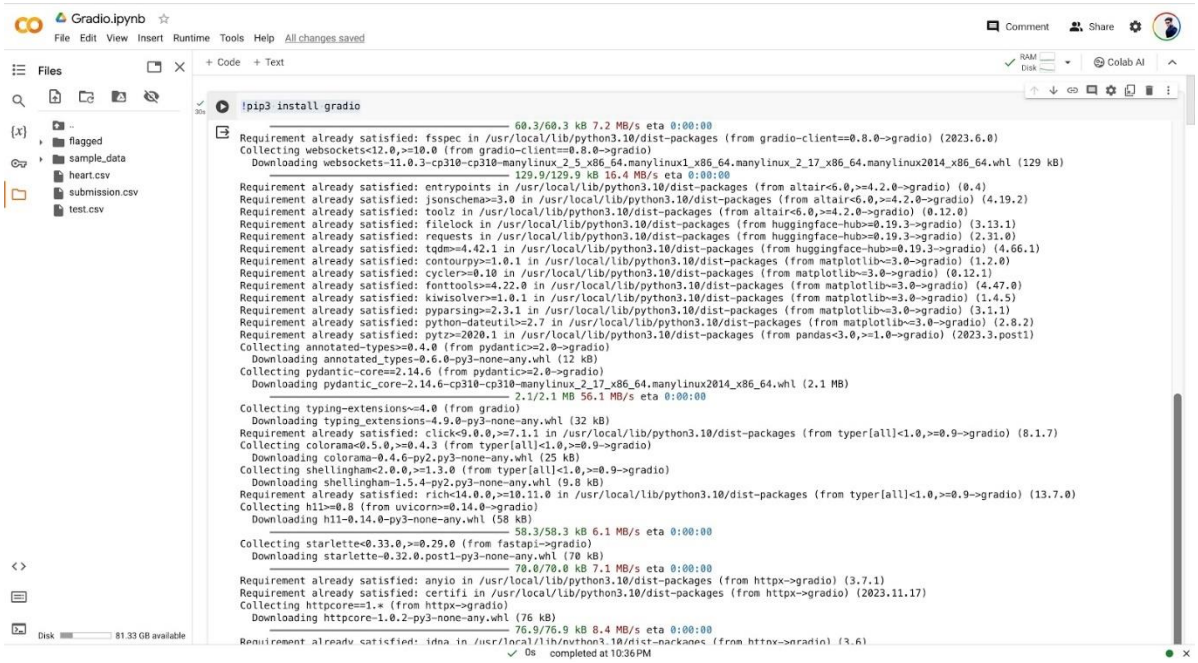
```
predictions = model.predict(st_scaled_X_test)


elif algorithm == "Random Forest":
model = RandomForestClassifier(oob_score=True,
random_state=42,
warm_start=True,
n_jobs=-1).set_params(n_estimators=150, warm_start=False)
model.fit(X, y)
predictions = model.predict(X_test)

elif algorithm == "Gradient Boosting":
model = GradientBoostingClassifier(n_estimators=30, random_state=42)
model.fit(X, y)
predictions = model.predict(X_test)

elif algorithm == "All":
predictions = None

for dirname, _, filenames in os.walk('/content'):
for filename in filenames:
if filename == 'submission.csv':
os.remove(os.path.join(dirname, filename))

output = pd.DataFrame({'Patient_ID' : test_data.Patient_ID, 'HeartDisease' : predictions})

output.to_csv('submission.csv', index=False)

first_five_output = pd.read_csv('submission.csv', nrows=10)

return first_five_output
def test_algorithms(algorithm, features):
predictions = None
y = df['HeartDisease']

X = pd.get_dummies(df[features])
X_test = pd.get_dummies(test_data[features])

scaler = StandardScaler()
st_scaled_X_train = scaler.fit_transform(X)
st_scaled_X_test = scaler.transform(X_test)

if algorithm == "KNN":
model = KNeighborsClassifier(n_neighbors=24, weights='distance')
model.fit(st_scaled_X_train, y)
predictions = model.predict(st_scaled_X_test)



elif algorithm == "Logistic Regression":
```

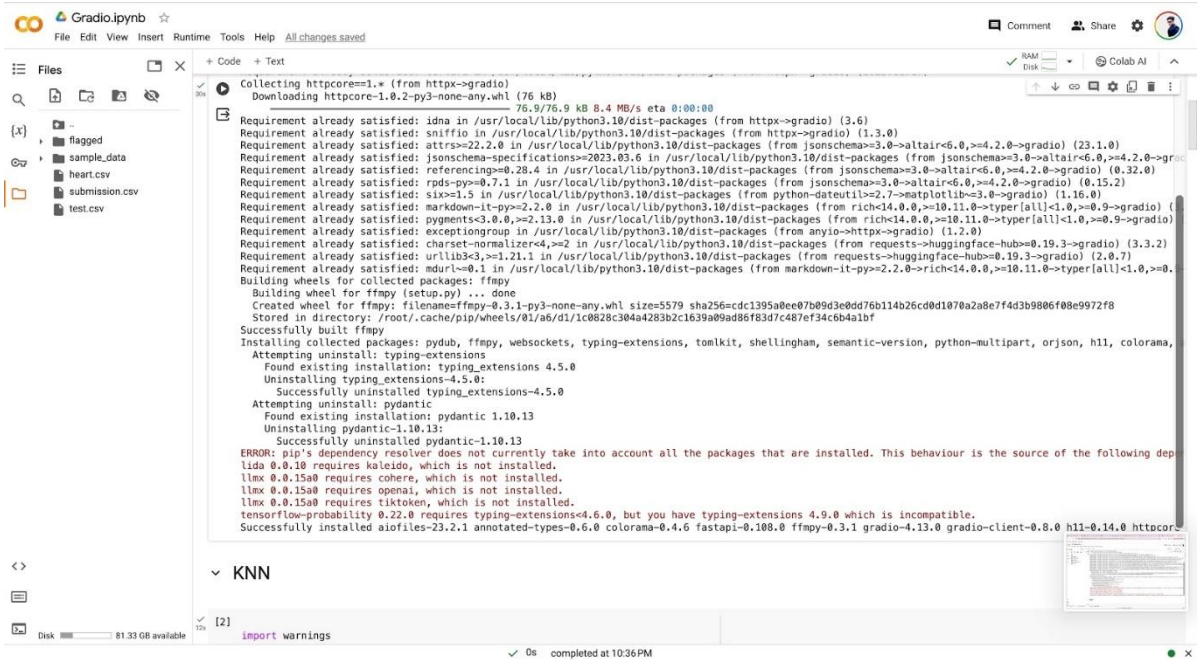
```
model = LogisticRegression(solver='liblinear')
model.fit(X, y)
predictions = model.predict(X_test)

elif algorithm == "SVC":
model = SVC()
model.fit(st_scaled_X_train, y)
predictions = model.predict(st_scaled_X_test)


elif algorithm == "Random Forest":
model = RandomForestClassifier(oob_score=True,
random_state=42,
warm_start=True,
n_jobs=-1).set_params(n_estimators=150, warm_start=False)
model.fit(X, y)
predictions = model.predict(X_test)

elif algorithm == "Gradient Boosting":
model = GradientBoostingClassifier(n_estimators=30, random_state=42)
model.fit(X, y)
predictions = model.predict(X_test)

elif algorithm == "All":
predictions = None

for dirname, _, filenames in os.walk('/content'):
for filename in filenames:
if filename == 'submission.csv':
os.remove(os.path.join(dirname, filename))

output = pd.DataFrame({'Patient_ID' : test_data.Patient_ID, 'HeartDisease' : predictions})

output.to_csv('submission.csv', index=False)

first_five_output = pd.read_csv('submission.csv', nrows=10)

return first_five_output
```

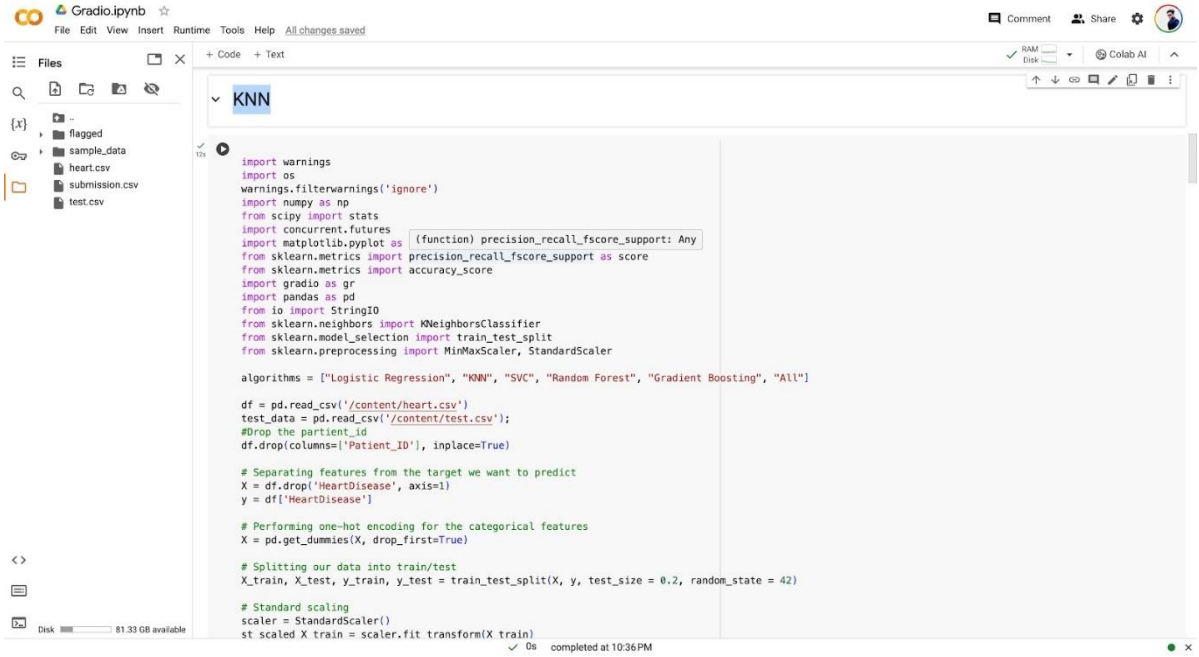
APPENDIX-B

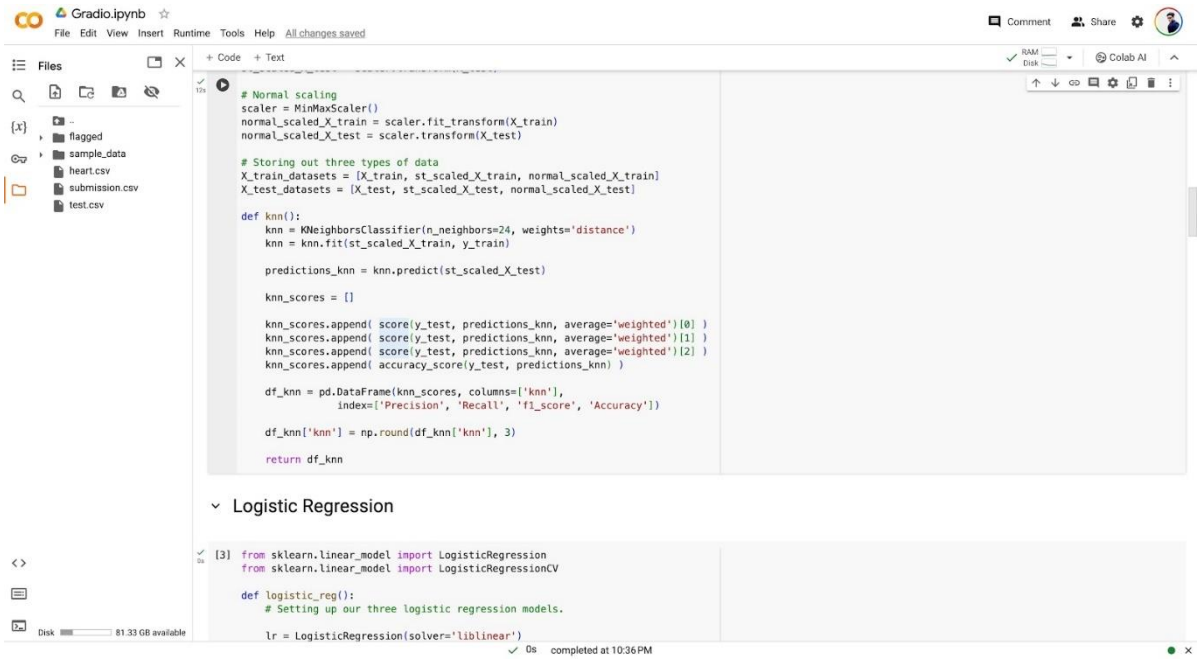
SCREENSHOTS

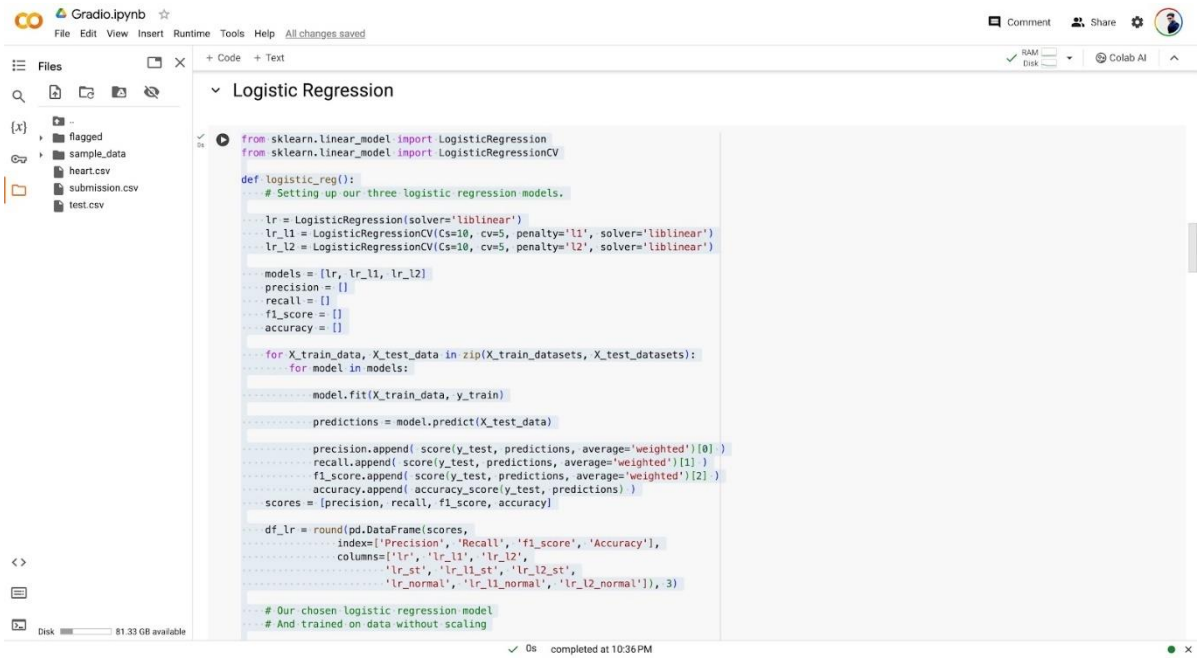


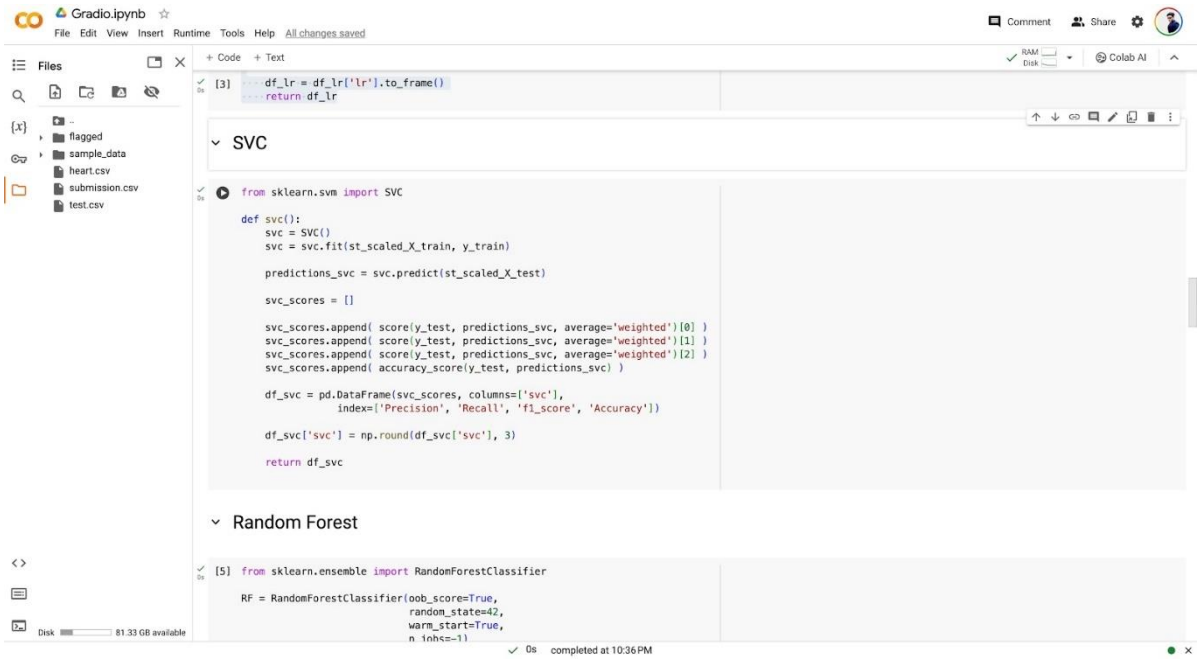


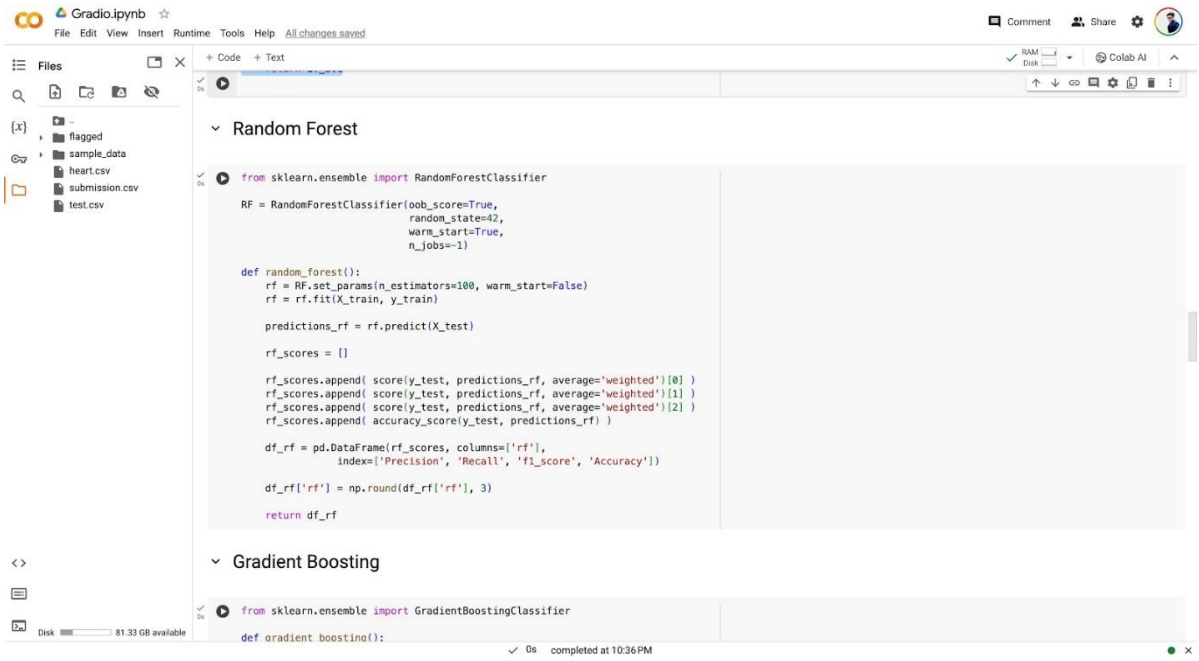

Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
Collecting httpcore==1.* (from httpx->gradio)
Downloading httpcore-1.0.2-py3-none-any.whl (76 kB)
76.9/76.9 kB 8.4 MB/s eta 0:00:00
Requirement already satisfied: idna in /usr/local/lib/python3.10/dist-packages (from httpx->gradio) (3.6)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from httpx->gradio) (1.3.0)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.32.0)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio) (0.15.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib~=3.0->gradio) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio) (2.16.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx->gradio) (1.2.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.19.3->gradio) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.19.3->gradio) (2.0.7)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich<14.0.0,>=10.11.0->typer[all]<1.0,>=0.9->gradio) (0.1.2)
Building wheels for collected packages: ffmpy
Building wheel for ffmpy (setup.py) ... done
Created wheel for ffmpy: filename=ffmpy-0.3.1-py3-none-any.whl size=5579 sha256=cdc1395a0ee07b09d3e0dd76b114b26cd0d1070a2a8e7f4d3b9806f08e9972f8
Stored in directory: /root/.cache/pip/wheels/01/a6/d1/1c0828c304a4283b2c1639a09ad86f83d7c487ef34c6b4a1bf
Successfully built ffmpy
Installing collected packages: pydub, ffmpy, websockets, typing-extensions, tomlkit, shellingham, semantic-version, python-multipart, orjson, h11, colorama,
Attempting uninstall: typing-extensions
Found existing installation: typing_extensions 4.5.0
Uninstalling typing_extensions-4.5.0:
Successfully uninstalled typing_extensions-4.5.0
Attempting uninstall: pydantic
Found existing installation: pydantic 1.10.13
Uninstalling pydantic-1.10.13:
Successfully uninstalled pydantic-1.10.13
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dep
lida 0.0.10 requires kaleido, which is not installed.
llmx 0.0.15a0 requires cohere, which is not installed.
llmx 0.0.15a0 requires openai, which is not installed.
llmx 0.0.15a0 requires tiktoken, which is not installed.
tensorflow-probability 0.22.0 requires typing-extensions<4.6.0, but you have typing-extensions 4.9.0 which is incompatible.
Successfully installed aiofiles-23.2.1 annotated-types-0.6.0 colorama-0.4.6 fastapi-0.108.0 ffmpy-0.3.1 gradio-4.13.0 gradio-client-0.8.0 h11-0.14.0 httpcor
KNN
[2]
import warnings
Disk 81.33 GB available
0s completed at 10:36 PM


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
KNN
import warnings
import os
warnings.filterwarnings('ignore')
import numpy as np
from scipy import stats
import concurrent.futures
import matplotlib.pyplot as
(function) precision_recall_fscore_support: Any
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import accuracy_score
import gradio as gr
import pandas as pd
from io import StringIO
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
algorithms = ["Logistic Regression", "KNN", "SVC", "Random Forest", "Gradient Boosting", "All"]
df = pd.read_csv('/content/heart.csv')
test_data = pd.read_csv('/content/test.csv');
#Drop the partient_id
df.drop(columns=['Patient_ID'], inplace=True)
# Separating features from the target we want to predict
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
# Performing one-hot encoding for the categorical features
X = pd.get_dummies(X, drop_first=True)
# Splitting our data into train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
# Standard scaling
scaler = StandardScaler()
st_scaled_X_train = scaler.fit_transform(X_train)
Disk 81.33 GB available
0s completed at 10:36 PM


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
# Normal scaling
scaler = MinMaxScaler()
normal_scaled_X_train = scaler.fit_transform(X_train)
normal_scaled_X_test = scaler.transform(X_test)
# Storing out three types of data
X_train_datasets = [X_train, st_scaled_X_train, normal_scaled_X_train]
X_test_datasets = [X_test, st_scaled_X_test, normal_scaled_X_test]
def knn():
knn = KNeighborsClassifier(n_neighbors=24, weights='distance')
knn = knn.fit(st_scaled_X_train, y_train)
predictions_knn = knn.predict(st_scaled_X_test)
knn_scores = []
knn_scores.append(score(y_test, predictions_knn, average='weighted')[0])
knn_scores.append(score(y_test, predictions_knn, average='weighted')[1])
knn_scores.append(score(y_test, predictions_knn, average='weighted')[2])
knn_scores.append(accuracy_score(y_test, predictions_knn))
df_knn = pd.DataFrame(knn_scores, columns=['knn'],
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])
df_knn['knn'] = np.round(df_knn['knn'], 3)
return df_knn
Logistic Regression
[3] from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
def logistic_reg():
# Setting up our three logistic regression models.
lr = LogisticRegression(solver='liblinear')
Disk 81.33 GB available
0s completed at 10:36 PM


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
def logistic_reg():
# Setting up our three logistic regression models.
lr = LogisticRegression(solver='liblinear')
lr_l1 = LogisticRegressionCV(Cs=10, cv=5, penalty='l1', solver='liblinear')
lr_l2 = LogisticRegressionCV(Cs=10, cv=5, penalty='l2', solver='liblinear')
models = [lr, lr_l1, lr_l2]
precision = []
recall = []
f1_score = []
accuracy = []
for X_train_data, X_test_data in zip(X_train_datasets, X_test_datasets):
for model in models:
model.fit(X_train_data, y_train)
predictions = model.predict(X_test_data)
precision.append(score(y_test, predictions, average='weighted')[0])
recall.append(score(y_test, predictions, average='weighted')[1])
f1_score.append(score(y_test, predictions, average='weighted')[2])
accuracy.append(accuracy_score(y_test, predictions))
scores = [precision, recall, f1_score, accuracy]
df_lr = round(pd.DataFrame(scores,
index=['Precision', 'Recall', 'f1_score', 'Accuracy'],
columns=['lr', 'lr_l1', 'lr_l2',
'lr_st', 'lr_l1_st', 'lr_l2_st',
'lr_normal', 'lr_l1_normal', 'lr_l2_normal']), 3)
Our chosen logistic regression model
And trained on data without scaling
Disk 81.33 GB available
0s completed at 10:36 PM


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
[3] df_lr = df_lr['lr'].to_frame()
return df_lr
SVC
from sklearn.svm import SVC
def svc():
svc = SVC()
svc = svc.fit(st_scaled_X_train, y_train)
predictions_svc = svc.predict(st_scaled_X_test)
svc_scores = []
svc_scores.append(score(y_test, predictions_svc, average='weighted')[0])
svc_scores.append(score(y_test, predictions_svc, average='weighted')[1])
svc_scores.append(score(y_test, predictions_svc, average='weighted')[2])
svc_scores.append(accuracy_score(y_test, predictions_svc))
df_svc = pd.DataFrame(svc_scores, columns=['svc'],
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])
df_svc['svc'] = np.round(df_svc['svc'], 3)
return df_svc
Random Forest
[5] from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(oob_score=True,
random_state=42,
warm_start=True,
Disk 81.33 GB available
0s completed at 10:36 PM


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
Random Forest
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(oob_score=True,
random_state=42,
warm_start=True,
n_jobs=-1)
def random_forest():
rf = RF.set_params(n_estimators=100, warm_start=False)
rf = rf.fit(X_train, y_train)
predictions_rf = rf.predict(X_test)
rf_scores = []
rf_scores.append(score(y_test, predictions_rf, average='weighted')[0])
rf_scores.append(score(y_test, predictions_rf, average='weighted')[1])
rf_scores.append(score(y_test, predictions_rf, average='weighted')[2])
rf_scores.append(accuracy_score(y_test, predictions_rf))
df_rf = pd.DataFrame(rf_scores, columns=['rf'],
index=['Precision', 'Recall', 'f1_score', 'Accuracy'])
df_rf['rf'] = np.round(df_rf['rf'], 3)
return df_rf
Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier
def gradient_boosting():
Disk 81.33 GB available
0s completed at 10:36 PM

Gradio.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

+ Code + Text

RAM Disk Colab AI

..
flagged
sample_data
heart.csv
submission.csv
test.csv

Disk 81.33 GB available

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier

def gradient_boosting():
    gb = GradientBoostingClassifier(n_estimators=30, random_state=42)
    gb = gb.fit(X_train, y_train)

    predictions_gb = gb.predict(X_test)

    gb_scores = []

    gb_scores.append( score(y_test, predictions_gb, average='weighted')[0] )
    gb_scores.append( score(y_test, predictions_gb, average='weighted')[1] )
    gb_scores.append( score(y_test, predictions_gb, average='weighted')[2] )
    gb_scores.append( accuracy_score(y_test, predictions_gb) )

    df_gb = pd.DataFrame(gb_scores, columns=['gb'],
                index=['Precision', 'Recall', 'f1_score', 'Accuracy'])

    df_gb['gb'] = np.round(df_gb['gb'], 3)

    return df_gb
```

```
[7] def all():
        return pd.concat([logistic_reg(), knn(), svc(), random_forest(), gradient_boosting()], axis=1)
```

```
[8] max_acc_lr = logistic_reg()['lr'].values[3]
    max_acc_knn = knn()['knn'].values[3]
    max_acc_svc = svc()['svc'].values[3]
    max_acc_rf = random_forest()['rf'].values[3]
    max_acc_gb = gradient_boosting()['gb'].values[3]

    simple = pd.DataFrame(
        {
            "Algorithms": ["KNN", "Logistic Regression", "SVC", "Random Forest", "Gradient Boosting"],
            "Accuracy": [max_acc_knn, max_acc_lr, max_acc_svc, max_acc_rf, max_acc_gb],
        }
```

0s completed at 10:36 PM

Gradio.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

+ Code + Text

RAM Disk Colab AI

..
flagged
sample_data
heart.csv
submission.csv
test.csv

Disk 81.33 GB available

```
def test_algorithms(algorithm, features):
    predictions = None
    y = df['HeartDisease']

    X = pd.get_dummies(df[features])
    X_test = pd.get_dummies(test_data[features])

    scaler = StandardScaler()
    st_scaled_X_train = scaler.fit_transform(X)
    st_scaled_X_test = scaler.transform(X_test)

    if algorithm == "KNN":
        model = KNeighborsClassifier(n_neighbors=24, weights='distance')
        model.fit(st_scaled_X_train, y)
        predictions = model.predict(st_scaled_X_test)

    elif algorithm == "Logistic Regression":
        model = LogisticRegression(solver='liblinear')
        model.fit(X, y)
        predictions = model.predict(X_test)

    elif algorithm == "SVC":
        model = SVC()
        model.fit(st_scaled_X_train, y)
        predictions = model.predict(st_scaled_X_test)


    elif algorithm == "Random Forest":
        model = RandomForestClassifier(oob_score=True,
                               random_state=42,
                               warm_start=True,
                               n_jobs=-1).set_params(n_estimators=150, warm_start=False)
        model.fit(X, y)
        predictions = model.predict(X_test)

    elif algorithm == "Gradient Boosting":
        model = GradientBoostingClassifier(n_estimators=30, random_state=42)
        model.fit(X, y)
        predictions = model.predict(X_test)

    elif algorithm == "All":
        predictions = None
```

0s completed at 10:36 PM

Gradio.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

flagged
sample_data
heart.csv
submission.csv
test.csv

Disk 81.33 GB available

+ Code + Text

RAM Disk Colab AI

```
[9]         predictions = None

        for dirname, _, filenames in os.walk('/content'):
            for filename in filenames:
                if filename == 'submission.csv':
                    os.remove(os.path.join(dirname, filename))

        output = pd.DataFrame({'Patient_ID' : test_data.Patient_ID, 'HeartDisease' : predictions})

        output.to_csv('submission.csv', index=False)

        first_five_output = pd.read_csv('submission.csv', nrows=10)

        return first_five_output
```

```
# Function to execute functions concurrently and gather results (Parallel Code)
def show_df(algorithm):
    functions = {
        "Logistic Regression": logistic_reg,
        "KNN": knn,
        "SVC": svc,
        "Random Forest": random_forest,
        "Gradient Boosting": gradient_boosting,
        "All": all
    }

    with concurrent.futures.ThreadPoolExecutor() as executor:
        result = executor.submit(functions[algorithm])
        return result.result()

# Sorting the accuracy values
sorted_accuracy = {k: v for k, v in sorted(accuracy_data.items(), key=lambda item: item[1], reverse=True)}

# Function to display accuracy line plot
def display_accuracy():
    plt.figure(figsize=(10, 6))
    plt.plot(list(sorted_accuracy.keys()), list(sorted_accuracy.values()), marker='o', linestyle='-', color='b')
    plt.title('Accuracy of Different Algorithms')
    plt.xlabel('Algorithms')
    plt.ylabel('Accuracy')
    plt.xticks(rotation=45)
```

0s completed at 10:36 PM

Gradio.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

flagged
sample_data
heart.csv
submission.csv
test.csv

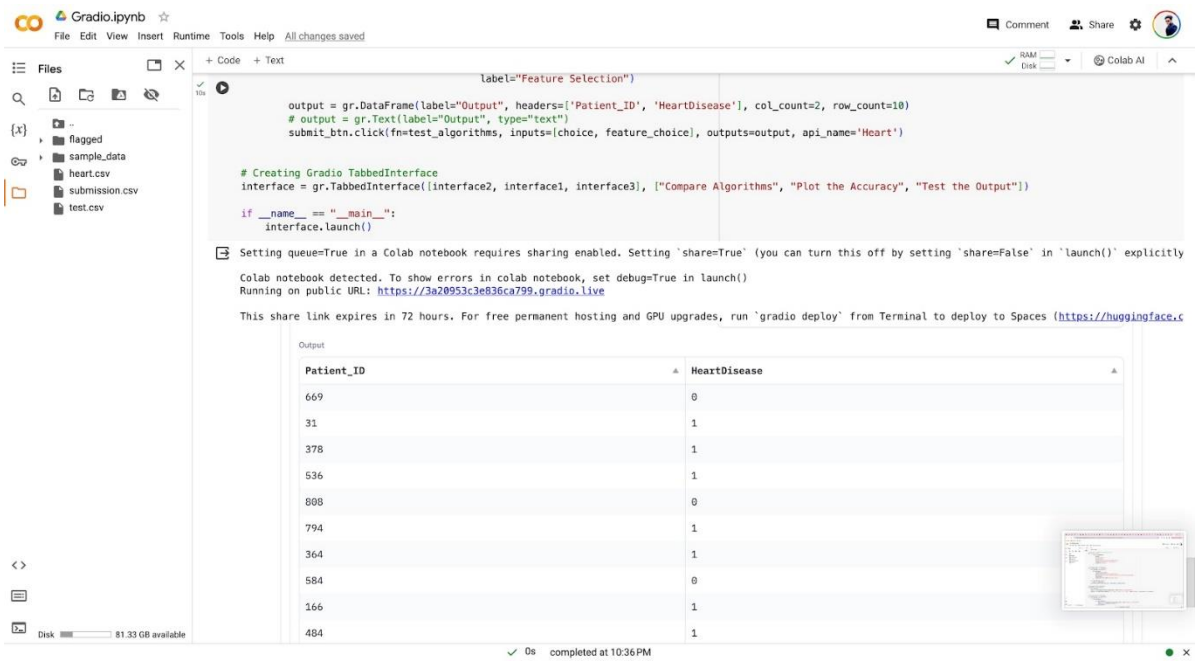
Disk 81.33 GB available

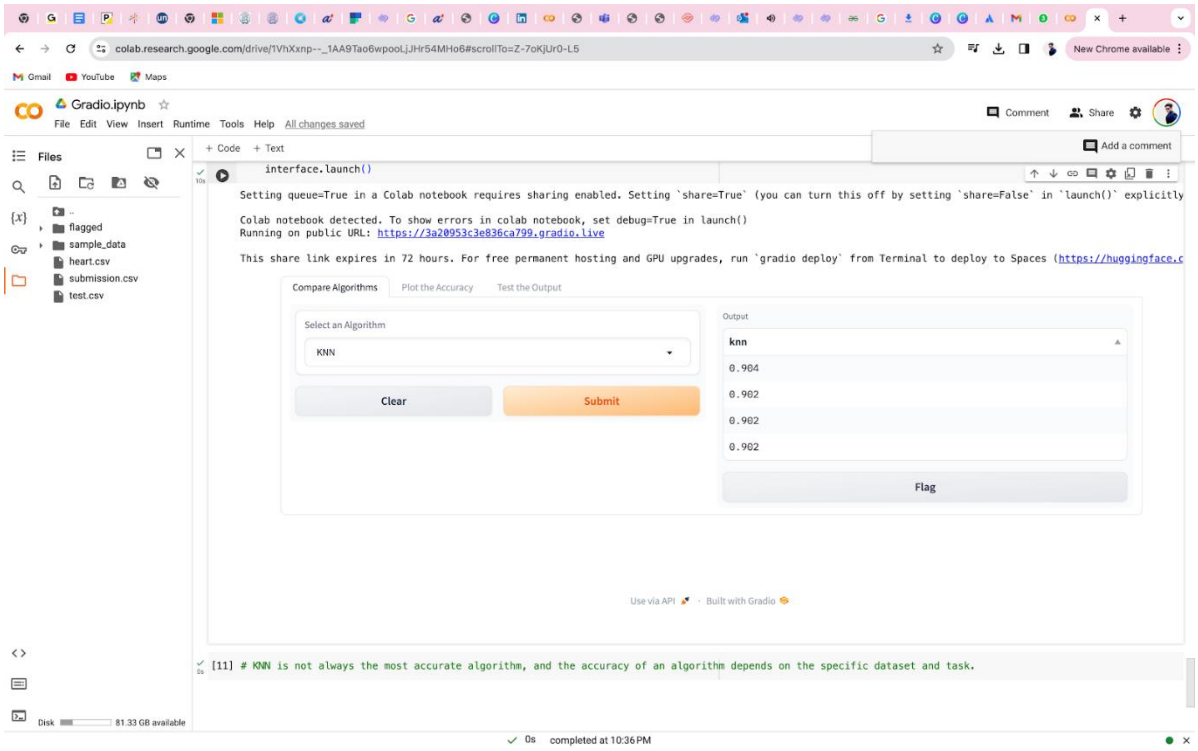
+ Code + Text

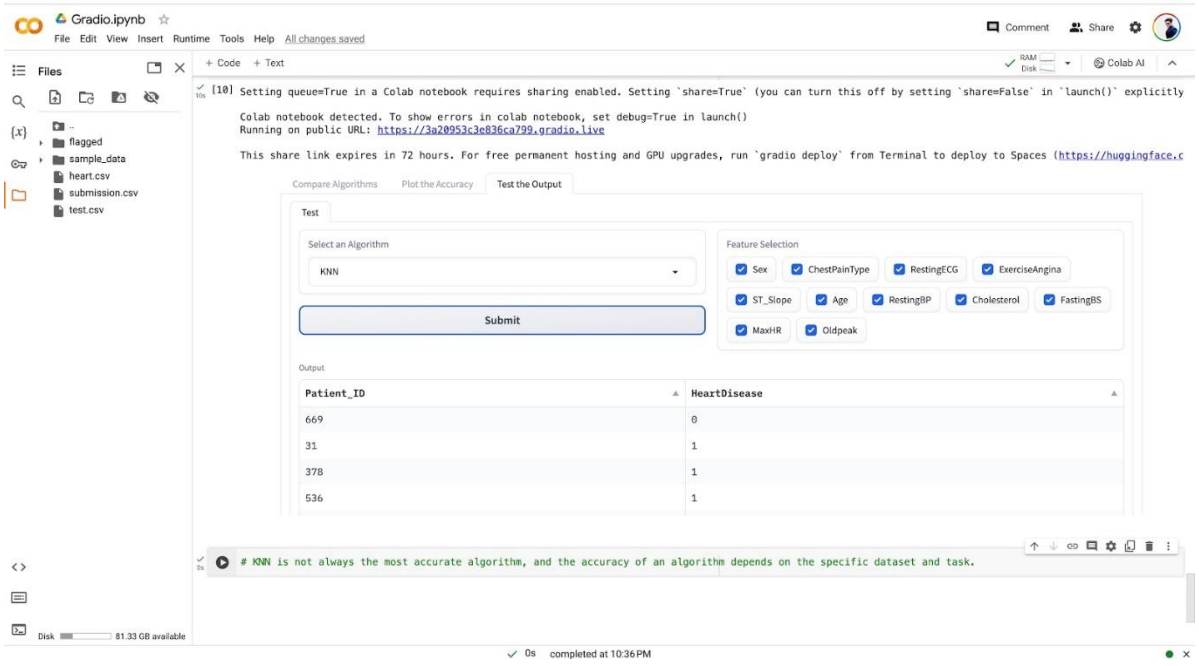
RAM Disk Colab AI

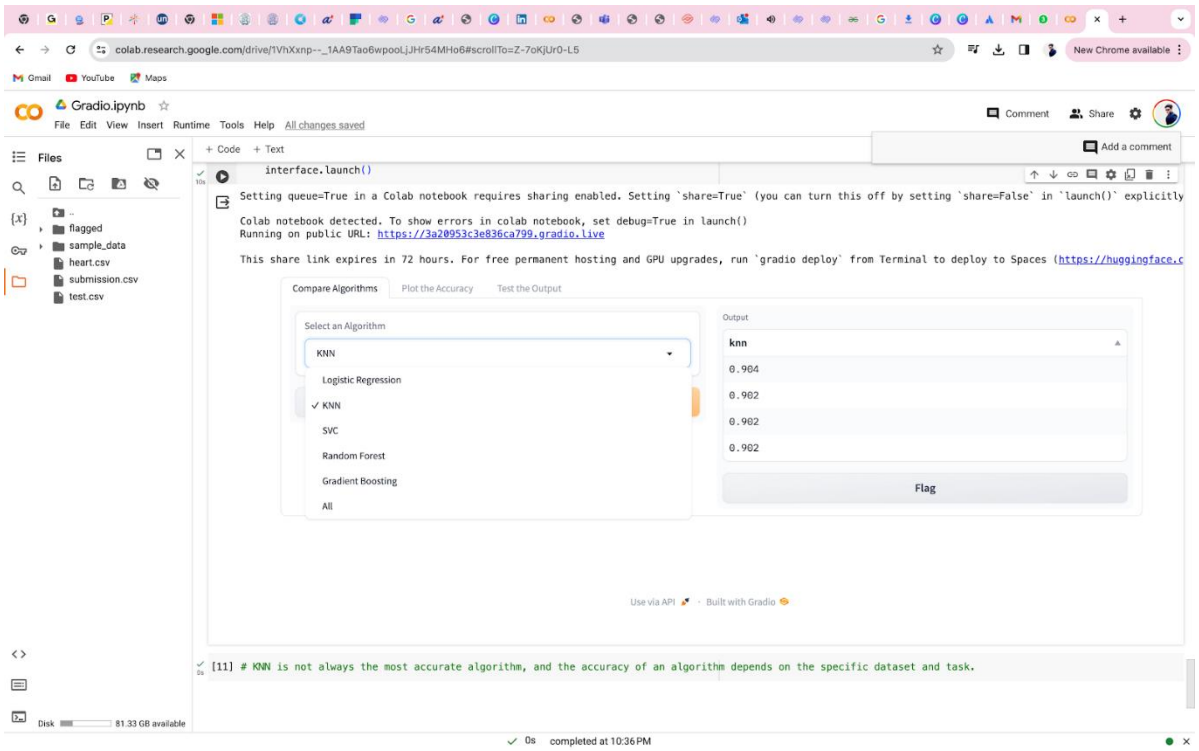
```
        return plt

# Function to display accuracy bar plot
def bar_plot_fn():
        return gr.BarPlot(
            simple,
            x="Algorithms",
            y="Accuracy",
            title="Algorithm Accuracy Comparison",
            tooltip=["Algorithms", "Accuracy"],
            y_lim=[0.0, 1.0],
        )

# Creating Gradio interface1
with gr.Blocks() as interface1:
    with gr.Tab("Line Plot"):
        gr.Interface(
            fn=display_accuracy,
            title='Algorithm Accuracy Comparison',
            description='Line plot showing accuracy of different algorithms',
            inputs=None,
            outputs=gr.Plot(label="Accuracy Plot")
        )
    with gr.Tab("Bar Plot"):
        plot = gr.BarPlot()
    interface1.load(fn=bar_plot_fn, inputs=None, outputs=plot)

# Creating Gradio interface2
interface2 = gr.Interface(
    fn = show_df,
    inputs = gr.Dropdown(choices=algorithms, label="Select an Algorithm"),
    outputs = gr.DataFrame(headers=['lr', 'knn', 'svc', 'rf', 'gb'], label="Output", row_count=4, col_count=5)
)

# Creating Gradio interface3
with gr.Blocks() as interface3:
    with gr.Tab("Test"):
        with gr.Row():
            with gr.Column():
                choice = gr.Dropdown(choices=algorithms, label="Select an Algorithm")
                submit_btn = gr.Button("Submit")
            with gr.Column():
```

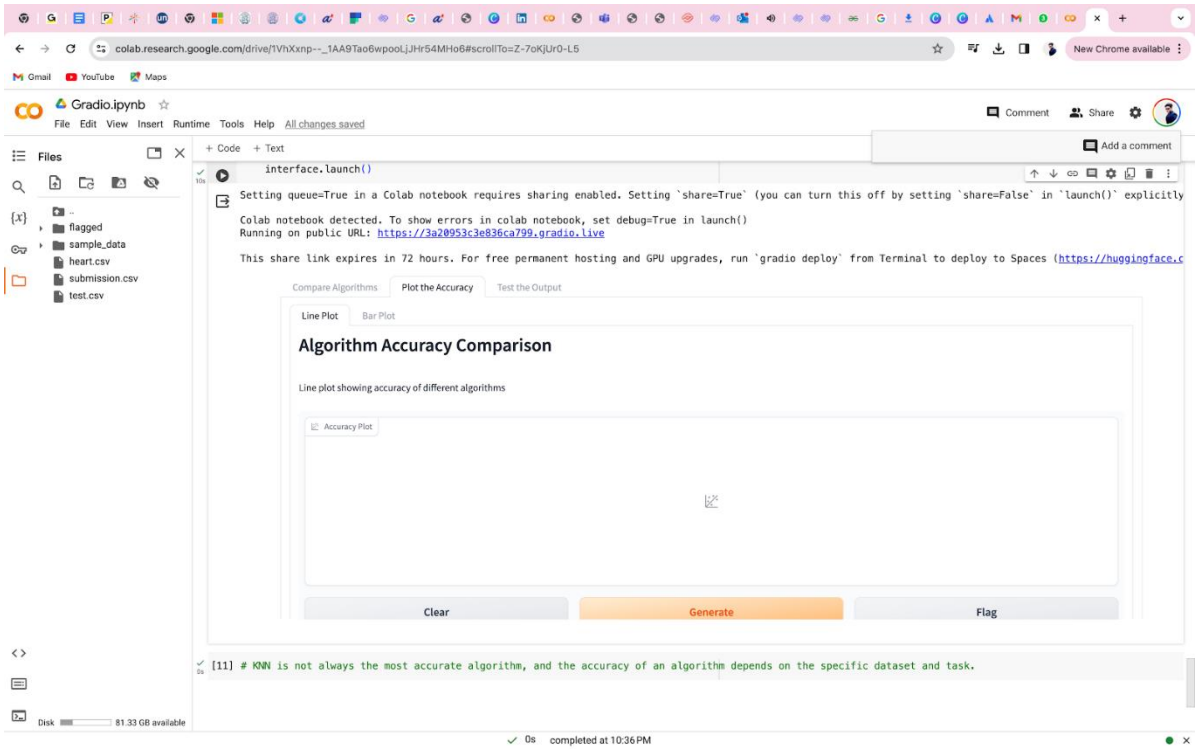
0s completed at 10:36 PM

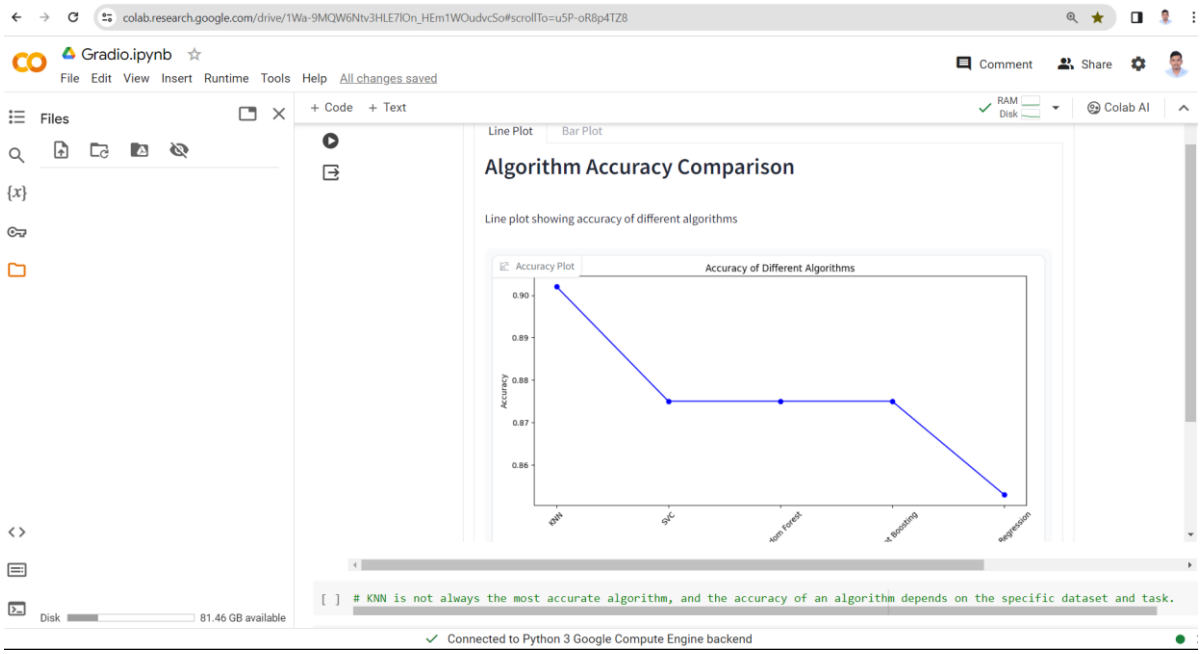

Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
Colab AI
flagged
sample_data
heart.csv
submission.csv
test.csv
label="Feature Selection")
output = gr.DataFrame(label="Output", headers=['Patient_ID', 'HeartDisease'], col_count=2, row_count=10)
# output = gr.Text(label="Output", type="text")
submit_btn.click(fn=test_algorithms, inputs=[choice, feature_choice], outputs=output, api_name='Heart')
# Creating Gradio TabbedInterface
interface = gr.TabbedInterface([interface2, interface1, interface3], ["Compare Algorithms", "Plot the Accuracy", "Test the Output"])
if __name__ == "__main__":
interface.launch()
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3a20953c3e836ca799.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.c
Output
Patient_ID
HeartDisease
669 0
31 1
378 1
536 1
808 0
794 1
364 1
584 0
166 1
484 1
Disk 81.33 GB available
0s completed at 10:36 PM

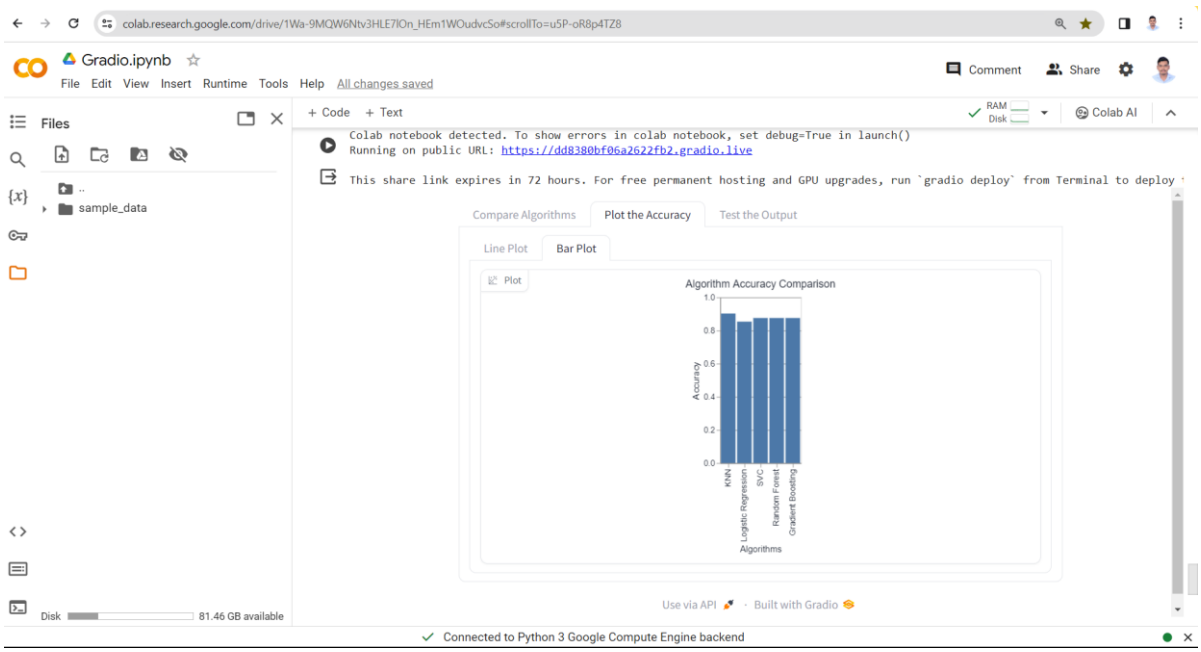

colab.research.google.com/drive/1VhXxnp--_1AA9Tao6wpooLjJHr54MHo6#scrollTo=Z-7oKjUr0-L5
New Chrome available
Gmail YouTube Maps
Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Add a comment
Files
+ Code + Text
interface.launch()
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3a20953c3e836ca799.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.c
Compare Algorithms
Plot the Accuracy
Test the Output
Select an Algorithm
KNN
Clear
Submit
Output
knn
0.904
0.902
0.902
0.902
Flag
Use via API
Built with Gradio
[11] # KNN is not always the most accurate algorithm, and the accuracy of an algorithm depends on the specific dataset and task.
Disk 81.33 GB available
0s completed at 10:36 PM

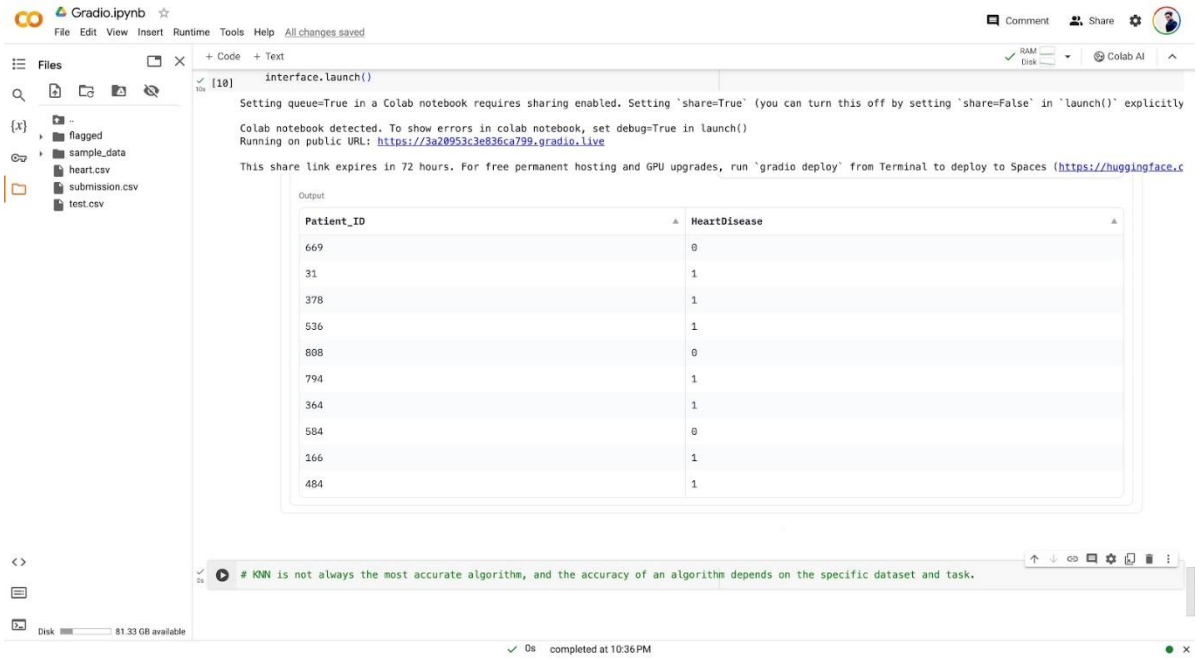

Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Colab AI
[10] Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3a20953c3e836ca799.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.c
Compare Algorithms
Plot the Accuracy
Test the Output
Test
Select an Algorithm
KNN
Feature Selection
Sex
ChestPainType
RestingECG
ExerciseAngina
ST_Slope
Age
RestingBP
Cholesterol
FastingBS
MaxHR
Oldpeak
Submit
Output
Patient_ID
HeartDisease
669
0
31
1
378
1
536
1
KNN is not always the most accurate algorithm, and the accuracy of an algorithm depends on the specific dataset and task.
Disk 81.33 GB available
0s completed at 10:36 PM


colab.research.google.com/drive/1VhXxnp--_1AA9Tao6wpooLjJHr54MHo6#scrollTo=Z-7oKjUr0-L5
New Chrome available
Gmail YouTube Maps
Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Add a comment
Files
+ Code + Text
interface.launch()
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3a20953c3e836ca799.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.c
Compare Algorithms
Plot the Accuracy
Test the Output
Select an Algorithm
KNN
Logistic Regression
KNN
SVC
Random Forest
Gradient Boosting
All
Output
knn
0.904
0.902
0.902
0.902
Flag
Use via API
Built with Gradio
[11] # KNN is not always the most accurate algorithm, and the accuracy of an algorithm depends on the specific dataset and task.
Disk 81.33 GB available
0s completed at 10:36 PM


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
interface.launch()
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3a20953c3e836ca799.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.c
Compare Algorithms
Plot the Accuracy
Test the Output
Line Plot
Bar Plot
Algorithm Accuracy Comparison
Line plot showing accuracy of different algorithms
Accuracy Plot
Clear
Generate
Flag
[11] # KNN is not always the most accurate algorithm, and the accuracy of an algorithm depends on the specific dataset and task.
Disk 81.33 GB available
0s completed at 10:36 PM


colab.research.google.com/drive/1Wa-9MQW6Ntv3HLE7lOn_HEm1WOudvcSo#scrollTo=u5P-oR8p4TZ8
Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Colab AI
Line Plot
Bar Plot
Algorithm Accuracy Comparison
Line plot showing accuracy of different algorithms
Accuracy Plot
Accuracy of Different Algorithms
0.90
0.89
0.88
0.87
0.86
Accuracy
KNN
SVC
[ ] # KNN is not always the most accurate algorithm, and the accuracy of an algorithm depends on the specific dataset and task.
Disk 81.46 GB available
Connected to Python 3 Google Compute Engine backend


colab.research.google.com/drive/1Wa-9MQW6Ntv3HLE7lOn_HEm1WOudvcSo#scrollTo=u5P-oR8p4TZ8
Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Colab AI
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://dd8380bf06a2622fb2.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy
Compare Algorithms
Plot the Accuracy
Test the Output
Line Plot
Bar Plot
Plot
Algorithm Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
KNN
Logistic Regression
SVC
Random Forest
Gradient Boosting
Algorithms
Use via API
Built with Gradio
sample_data
Disk
81.46 GB available
Connected to Python 3 Google Compute Engine backend


Gradio.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Colab AI
[10] interface.launch()
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://3a20953c3e836ca799.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from Terminal to deploy to Spaces (https://huggingface.c
Output
Patient_ID
HeartDisease
669 0
31 1
378 1
536 1
808 0
794 1
364 1
584 0
166 1
484 1
KNN is not always the most accurate algorithm, and the accuracy of an algorithm depends on the specific dataset and task.
flagged
sample_data
heart.csv
submission.csv
test.csv
Disk
81.33 GB available
0s completed at 10:36 PM

APPENDIX-C

ENCLOSURES

1. Conference Paper Presented Certificates of all students.

Journal of Emerging Technologies and Innovative Research - (JETIR.ORG)
International Peer Reviewed & Refereed Journals, Open Access Journal
ISSN: 2349-5162 | Impact factor: 7.95 | ESTD Year: 2014
Scholarly open access journals, Peer-reviewed, and Refereed Journals, Impact factor 7.95 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI)

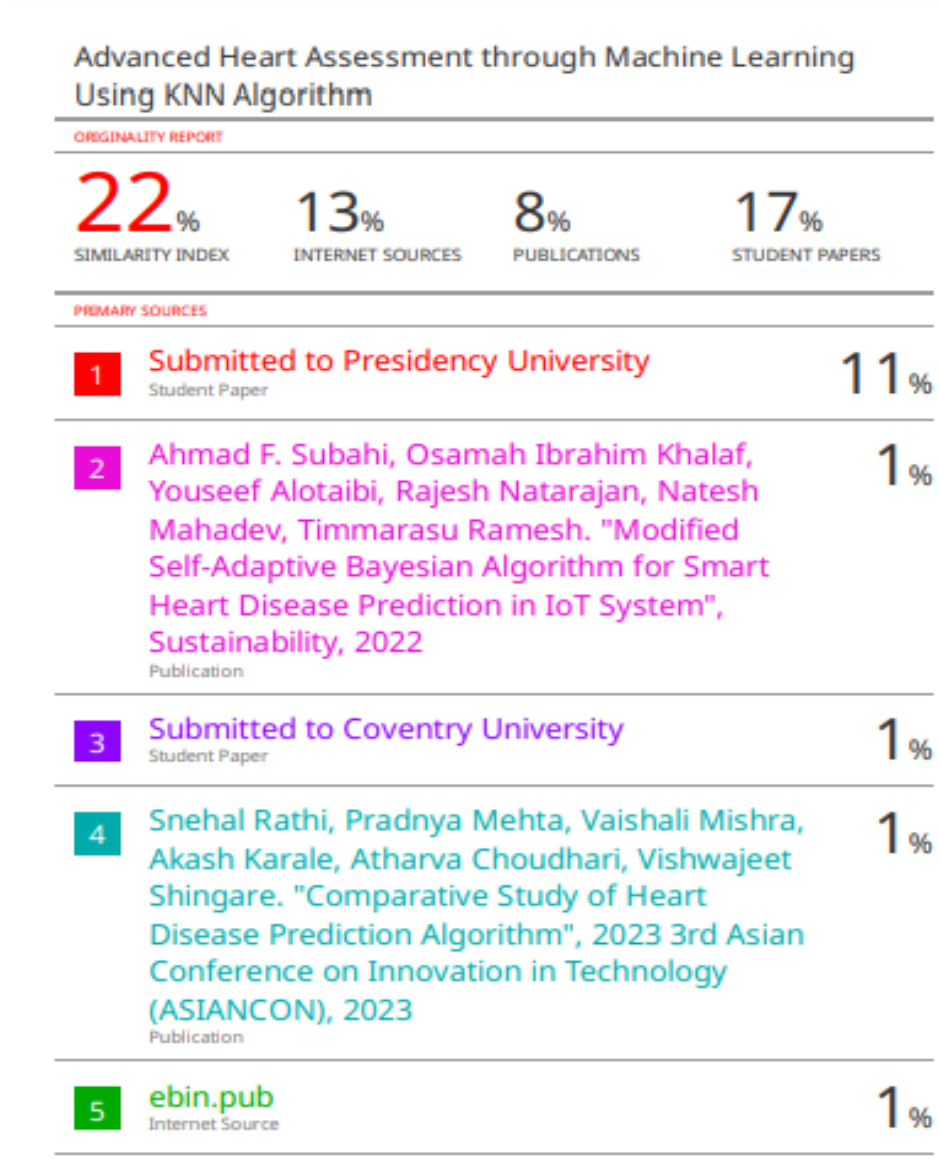
Dear Author, Congratulation!!!

Your manuscript with Registration/Paper ID:531399 has been Accepted for publication in the Journal of Emerging Technologies and Innovative Research(JETIR) | www.JETIR.org | ISSN: 2349-5162 | International Peer Reviewed & Refereed Journals, Open Access Online and Print Journal.

JETIR Impact Factor: 7.95
Check Your Paper Status: track.php

Your Paper Review Report :					
Registration/Paper ID:	**531399**				
Title of the Paper:	**Advanced Heart Assessment through Machine Learning Using KNN Algorithm**				
Unique Contents:	**88% (Out of 100)**	**Paper Accepted:**	**Accepted**	**Overall Assessment (Comments):**	**Reviewer Comment store in Online RMS system**
Publication of Paper:	**Paper Accepted. Please complete payment and documents process. Paper will be published Within 01-02 Days after submission of payment proof and documents to $email. Complete below Step 1 and 2**				
Publication/Article Processing Fees					

2. **Similarity Index / Plagiarism Check report clearly showing the Percentage (%).**



Advanced Heart Assessment through Machine Learning Using KNN Algorithm

ORIGINALITY REPORT

22%	13%	8%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

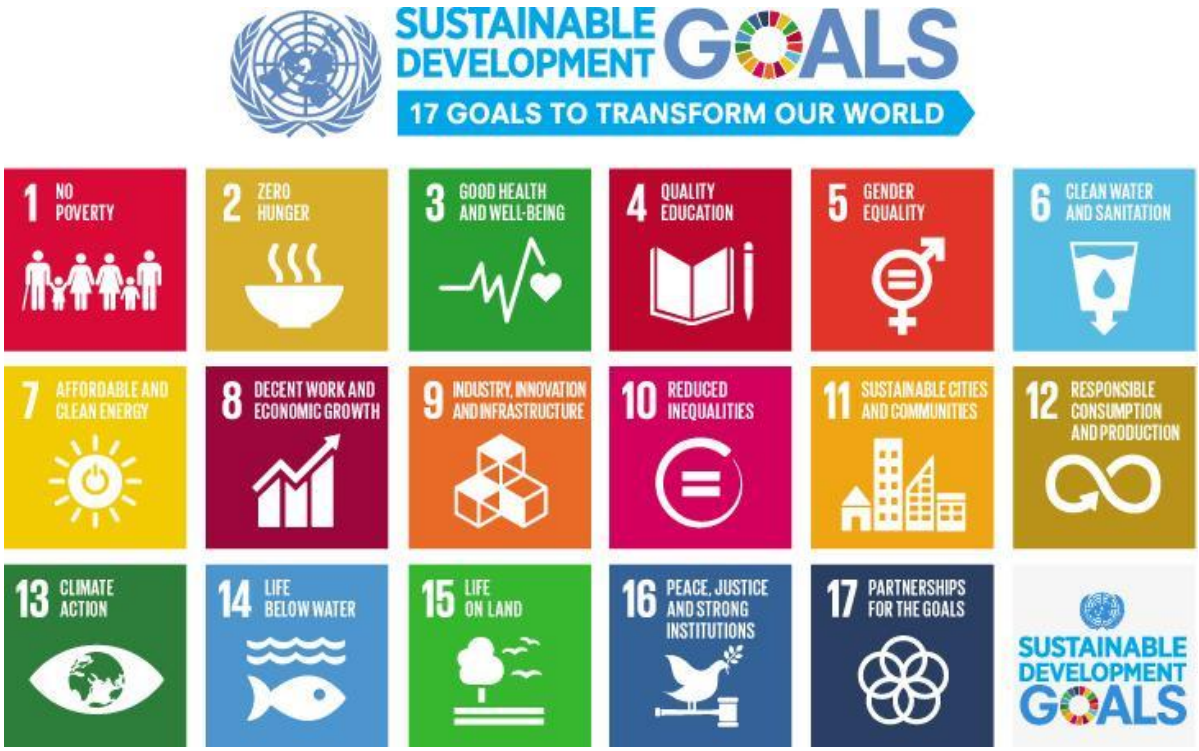
1	Submitted to Presidency University Student Paper	11%
2	Ahmad F. Subahi, Osamah Ibrahim Khalaf, Youseef Alotaibi, Rajesh Natarajan, Natesh Mahadev, Timmarasu Ramesh. "Modified Self-Adaptive Bayesian Algorithm for Smart Heart Disease Prediction in IoT System", Sustainability, 2022 Publication	1%
3	Submitted to Coventry University Student Paper	1%
4	Snehal Rathi, Pradnya Mehta, Vaishali Mishra, Akash Karale, Atharva Choudhari, Vishwajeet Shingare. "Comparative Study of Heart Disease Prediction Algorithm", 2023 3rd Asian Conference on Innovation in Technology (ASIANCON), 2023 Publication	1%
5	ebin.pub Internet Source	1%

6	Onika Yadav, Yashika Dangi, Noor Mohd, Indrajeet Kumar. "Predictive Model for M-Health (Cardiovascular Diseases) using ML Application", 2023 World Conference on Communication & Computing (WCONF), 2023 Publication	1 %
7	Submitted to Carnegie Mellon University Student Paper	< 1 %
8	www.nald.ca Internet Source	< 1 %
9	ijariie.com Internet Source	< 1 %
10	Ajay Kumar, Anuj Kumar Singh, Ankit Garg. "Evaluation of machine learning techniques for heart disease prediction using multi-criteria decision making", Journal of Intelligent & Fuzzy Systems, 2023 Publication	< 1 %
11	Submitted to The University of the West of Scotland Student Paper	< 1 %
12	www.ncbi.nlm.nih.gov Internet Source	< 1 %
13	Submitted to Nexford Learning Solutions Student Paper	< 1 %
14	fastercapital.com	

	Internet Source	<1%
15	www.nature.com Internet Source	<1%
16	www.researchgate.net Internet Source	<1%
17	Farah Mohammad, Saad Al-Ahmadi. "WT-CNN: A Hybrid Machine Learning Model for Heart Disease Prediction", Mathematics, 2023 Publication	<1%
18	Amin Ul Haq, Jianping Li, Muhammad Hammad Memon, Muhammad Hunain Memon, Jalaluddin Khan, Syeda Munazza Marium. "Heart Disease Prediction System Using Model Of Machine Learning and Sequential Backward Selection Algorithm for Features Selection", 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019 Publication	<1%
19	Submitted to University of Hertfordshire Student Paper	<1%
20	www.science.gov Internet Source	<1%
21	Submitted to Michigan Technological University Student Paper	<1%

22	Submitted to The University of the South Pacific Student Paper	<1 %
23	Submitted to University of East London Student Paper	<1 %
24	www.slideshare.net Internet Source	<1 %
25	Submitted to Bournemouth University Student Paper	<1 %
26	Submitted to University of Greenwich Student Paper	<1 %
27	lacdev.iarc.who.int Internet Source	<1 %
28	www.iarconsortium.org Internet Source	<1 %
29	ijcseonline.org Internet Source	<1 %
30	research.tamhsc.edu Internet Source	<1 %
31	unisa.edu.au Internet Source	<1 %
32	pdfcoffee.com Internet Source	<1 %
	www.ijnrd.org	
33	Internet Source	<1 %
34	www.ir.juit.ac.in:8080 Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		



Good Health and Well-Being

"Prevention is better than Cure". Prevention is key in maintaining good health. Predictive models for heart disease, powered by machine learning, contribute to preventive healthcare by identifying risk factors early on and enabling individuals to adopt lifestyle changes or medical interventions that can mitigate potential health issues. Predicting heart disease using machine learning can aid in early detection, allowing for timely intervention and personalized preventive measures. This can lead to improved health outcomes, reduced treatment costs, and an overall enhancement in well-being by addressing potential risks before they escalate.