COMPARATIVE ANALYSIS FOR DISEASE PREDICTION SYSTEM USING ML ALGORITHMS

A PROJECT REPORT

Submitted by

ANKIT SINGH [RA2011003011337]

HARSHAVARDHAN .D [RA2011003011350]

Under the guidance of

Dr. S. SANKARA NARAYANAN

(Assistant Professor, Department of Computing Technologies)

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

Department of Computing Technologies
SRM Institute of Science & Technology
Own Work Declaration Form

Degree/ Course	**:** BTech / Computer Science and Engineering
Student Name	**:** Ankit Singh and Harshavardhan Dhayapule
Registration Number	**:** RA2011003011337 and RA2011003011350
Title of Work	**:** Comparative Analysis for Disease Prediction using ML Algorithms

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations,and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the reports or essays of any other students either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellowstudents, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / Universitywebsite

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except were indicated by referring, and that I have followed the good academic practices noted above. **Student 1 Signature:** **Student 2 Signature:** **Date:**
If you are working in a group, please write your registration numbers and sign with the date forevery student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled "**COMPARATIVE ANALYSIS FOR DISEASE PREDICTION SYSTEM USING ML ALGORITHMS**" is the bonafide work of **ANKIT SINGH [RA2011003011337]** and **HARSHAVARDHAN DHAYAPULE [RA2011003011350]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. S. SANKARA NARAYANAN
SUPERVISOR
Assistant Professor
Department of Computing Technologies

Dr. R. JEBAKUMAR
PANEL HEAD
Associate Professor
Department of Computing Technologies

Dr. M. PUSHPALATHA
HEAD OF THE DEPARTMENT
Department of Computing Technologies

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We would like to express our warmth of gratitude to our Registrar, **Dr. S. Ponnusamy**, for his encouragement.

We extend our sincere thanks to **Dr. T. V. Gopal** , Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. S. Godfrey Winster**, Associate Professor, **Dr. M. Baskar**, Associate Professor, **Dr. P. Murali**, Associate Professor, **Dr. J. Selvin Paul Peter**, Associate Professor, **Dr. C. Pretty Diana Cyril**, Assistant Professor and **Dr. G. Padmapriya**, Assistant Professor and our Panel Head, **Dr. R. Jebakumar**, Associate Professor and Panel Members, **Dr. S. Sankara Narayanan**, Assistant Professor, **Dr. T. Ragunthar**, Assistant Professor and **Dr. B. Prakash**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We are highly thankful to our Course Project Guide, **Dr. S. Sankara Narayanan**, Assistant Professor, Department of Computing Technologies for his assistance, timely suggestion, and guidance throughout the duration of this course project and Course Coordinators for their constant encouragement and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. K. Deeba**, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

ANKIT SINGH [RA2011003011337]

HARSHVARDHAN D [RA2011003011350]

ABSTRACT

In our project, we embarked on a thorough comparative analysis aimed at predicting diabetes and heart disease using a range of machine learning techniques. The journey began with a detailed exploratory data analysis (EDA) of our diabetes dataset, where we delved into the descriptive statistics, outcome variable distributions, and visualized key features to understand the dataset's characteristics. Our analysis uncovered missing observations, prompting preprocessing steps such as replacing zero values with NaN and filling missing data with median values to ensure data quality. Additionally, we conducted outlier analysis using the Local Outlier Factor (LOF) method to identify and manage outliers effectively. To enhance predictive power, we undertook feature engineering by creating new variables based on crucial physiological indicators like BMI, insulin, and glucose levels. We applied one hot encoding to categorical variables and split the dataset into training and test sets in preparation for modeling. Through this comprehensive approach, we aimed to develop robust predictive models capable of accurate disease prediction. Our focus on data quality, feature engineering, and model preparation underscores our commitment to leveraging machine learning for improving healthcare outcomes.

This project represents a significant step towards bridging the gap between traditional diagnostic methods and cutting-edge data-driven approaches, with the ultimate goal of empowering healthcare professionals with actionable insights for early disease detection and intervention.

TABLE OF CONTENTS

	ABSTRACT	**vi**
	TABLE OF CONTENT	**vii**
	LIST OF FIGURES	**ix**
	LIST OF TABLES	**x**
	LIST OF SYMBOLS AND ABBREVIATIONS	**xi**
1	**INTRODUCTION**	**1**
	1.1 About Project Background	1
	1.2 Scope of the Project	2
	1.3 Project Objectives	3
	1.4 Problem Statement	4
2	**LITERATURE SURVEY**	**6**
	2.1 Related Work	6
	2.2 Research Analysis Table	9
3	**METHODOLOGY AND IMPLEMENTATION**	**11**
	3.1 Methodology	11
	3.2 Block Diagram	14
	3.3 Architecture Diagram	15
	3.4 Flow Diagram	16
	3.5 Challenges and Existing System	17
	3.6 Proposed Architecture	17
	3.7 Implementation	19
4	**PERFORMANCE EVALUATION GRAPHS**	**36**
	4.1 Histogram and density graphs for Diabetes	36
	4.2 Distribution plot for Heart Disease values	37
	4.3 Correlation Matrix	38
	4.4 Model Comparison	40
	4.5 ROC Curve	41
	4.6 Performance Evaluation	42
5	**RESULT AND DISCUSSION**	**43**
	5.1 Results	43

	5.2 Discussions	43
6	**CONCLUSION AND FUTURE SCOPE**	**45**
	6.1 Future Scope	45
	6.2 Conclusion	46
	REFERENCES	**47**
	APPENDIX 1	**50**
	APPENDIX 2	**53**
	PLAGIARISM REPORT	**59**
	PAPER PUBLICATION PROOF	**64**

LIST OF FIGURES

3.2	Block Diagram…………………………………………	14
3.3	Architecture Diagram………………………………...	15
3.4	Flow Diagram………………………………………..	16
4.1	Histogram and density graph for diabetes……………..	36
4.2	Distribution plot for Heart Disease values……………...	37
4.3	Correlation Matrix for Diabetes………………………...	38
4.4	Correlation Matrix for Heart Disease…………………..	39
4.5	Model Comparison for Diabetes………………………..	40
4.6	Model Comparison for Heart Disease…………………..	40
4.7	ROC Graph for Diabetes………………………………..	41
4.8	ROC Graph for Heart Disease………………...……..	41
4.9	Performance evaluation for Diabetes………………….	42
4.10	Performance Evaluation for Heart Disease…………….	42

LIST OF TABLES

2.2 Research Analysis Table……………………………………………………………… 9

LIST OF SYMBOLS AND ABBREVIATIONS

ML	Machine Learning
RF	Random Forest
DT	Decision Tree
GB	Gradient Boosting
LR	Logistic Regression
XGB	XGBoost
SVM	Support Vector Machine
KNN	K-Nearest Neighbors

CHAPTER 1

INTRODUCTION

1.1 About Project Background

Chronic diseases like diabetes and heart disease pose significant challenges to healthcare systems worldwide. Early detection and accurate prediction of these conditions are crucial for effective management and intervention. In this project, we undertake a comparative analysis leveraging machine learning algorithms to predict both diabetes and heart disease.

Our journey begins with an in-depth exploration of datasets containing medical predictor variables such as glucose levels, BMI, and insulin levels, alongside target variables indicating disease outcomes. Through exploratory data analysis (EDA), we gain insights into the distributions, correlations, and characteristics of the data, laying the foundation for subsequent analysis.

Addressing missing observations and outliers, we preprocess the data to ensure its quality and reliability. Feature engineering plays a vital role in enhancing the predictive power of our models, where new variables are created based on key physiological indicators like BMI and glucose levels.

With our data prepared, we embark on implementing a suite of machine learning algorithms, including Logistic Regression, KNN, SVM, Decision Trees, Random Forest, Gradient Boosting, and XGBoost. Each algorithm is trained and evaluated using established metrics such as accuracy scores and confusion matrices, allowing us to assess their performance in predicting both diabetes and heart disease.

Furthermore, we extend our analysis to heart disease prediction, employing a similar methodology to that used for diabetes. Through model comparison, we aim to identify the most effective algorithms for each condition, offering valuable insights for clinicians and healthcare practitioners.

By the project's conclusion, we aim to contribute to the growing body of research in predictive healthcare analytics, providing actionable insights and tools to aid in the early detection and management of chronic diseases.

1.2 Scope of the Project

Clinical Decision Support:

Our project aims to provide healthcare professionals with robust predictive models for diabetes and heart disease prediction, serving as valuable tools for clinical decision support. These models can assist clinicians in risk stratification, treatment planning, and patient management, ultimately improving the quality and efficiency of healthcare delivery.

Preventive Healthcare:

By leveraging machine learning algorithms, we enable proactive identification of individuals at high risk of developing diabetes and heart disease.
Early detection allows for timely interventions, lifestyle modifications, and preventive measures to mitigate disease progression and reduce morbidity and mortality rates.

Personalized Medicine:

Our predictive models offer personalized risk assessments tailored to individual patient characteristics, including demographics, clinical variables, and lifestyle.
This personalized approach enables targeted interventions and tailored healthcare strategies, optimizing patient outcomes and resource allocation.

Healthcare Resource Optimization:

By accurately identifying individuals at high risk of diabetes and heart disease, our models help optimize healthcare resource allocation and allocation, allowing for more efficient use of healthcare resources and reducing unnecessary costs.

Research and Development:

Our project contributes to ongoing research efforts in predictive analytics for chronic disease management, providing insights into the efficacy and performance of different machine learning algorithms.

The findings from our comparative analysis can inform future research directions, guiding the development of novel predictive models and healthcare interventions.

Patient Empowerment:

Through improved risk assessment and personalized healthcare recommendations, our project empowers patients to take an active role in managing their health.

Access to personalized risk information and actionable insights enables patients to make informed decisions about their lifestyle choices, leading to better health outcomes and improved quality of life.

Public Health Initiatives:
The insights generated from our project can inform public health initiatives aimed at disease prevention and health promotion.
By identifying population-level risk factors and trends, policymakers can develop targeted interventions and public health campaigns to address the underlying determinants of diabetes and heart disease.

1.3 Project Objectives

Our project aims to conduct an extensive comparative analysis of machine learning algorithms tailored specifically for predicting diabetes and heart disease. Leveraging comprehensive exploratory data analysis techniques, advanced preprocessing methods, and innovative feature engineering, we seek to develop highly accurate and interpretable predictive models.

The primary objective of our project is to provide valuable tools that aid healthcare professionals in early detection and proactive intervention. By refining predictive models through rigorous data analysis and feature engineering, we aim to address key challenges related to data quality and model interpretability within healthcare applications.

Our project's focus extends beyond predictive accuracy to include actionable insights and personalized risk assessments. We aim to empower healthcare providers with accessible and understandable model outputs, enabling informed decision-making and optimized patient care strategies. Through the application of machine learning, our project aims to enhance clinical decision-making processes, optimize resource allocation, and contribute to advancements in preventive healthcare initiatives.

Ultimately, our goal is to develop models that seamlessly integrate into clinical workflows, enabling healthcare providers to leverage data-driven insights for early diagnosis, personalized treatment planning, and continuous patient monitoring. By bridging the gap between data science and healthcare, we aspire to improve patient outcomes, optimize healthcare delivery, and contribute to the broader goal of advancing preventive healthcare initiatives.

1.4 Problem Statement

The existing diagnostic methods for identifying diabetes and heart diseases face In the realm of healthcare, the landscape of disease diagnosis presents significant challenges that impede effective patient care. Traditional diagnostic processes often entail laborious and time-consuming methods, leading to delays in delivering timely interventions to patients who urgently require medical attention. Moreover, the reliance on manual diagnoses introduces inherent human errors, which can result in inaccuracies and missed diagnoses, ultimately impacting patient health outcomes and overall healthcare efficiency.

These challenges extend beyond individual patient cases and have broader implications for the healthcare system as a whole. The inefficiencies of traditional diagnostic systems, characterized by manual data analysis and subjective clinical assessments, constrain the healthcare system's capacity to deliver prompt and precise care to an increasingly diverse and expanding patient population. As healthcare needs continue to evolve, there is a clear and pressing need for a transformative approach to disease diagnosis — one that is more efficient, accurate, and capable of delivering timely results to enhance patient outcomes and optimize healthcare resource allocation.

Our project aims to address these challenges head-on by harnessing the power of innovative machine learning algorithms and predictive analytics to revolutionize disease diagnosis. Through advanced data analysis and modeling techniques, we are focused on developing predictive models that can assist healthcare professionals in making informed decisions and providing timely interventions to patients. By leveraging data-driven insights and cutting-edge technology, our transformative approach not only enhances patient care but also contributes to the overall efficiency and effectiveness of healthcare delivery.

At the heart of our initiative is the recognition that improving disease diagnosis is fundamental to improving patient outcomes and optimizing healthcare workflows. By leveraging the capabilities of machine learning and predictive analytics, we seek to streamline the diagnostic process, enabling healthcare providers to identify and address conditions such as diabetes and heart disease with greater accuracy and efficiency. This not only enhances individual patient care but also contributes to the broader goal of advancing healthcare systems towards more data-driven approaches.

In essence, our project represents a commitment to innovation and excellence in healthcare, aiming to empower healthcare professionals with the tools and insights needed to deliver personalized and effective care to every patient. By bridging the gap between traditional diagnostic methods and modern technological advancements, we aspire to make a meaningful impact on patient outcomes and pave the way for a more efficient and responsive healthcare system.

Our project's focus extends beyond predictive accuracy to include actionable insights and personalized risk assessments. We aim to empower healthcare providers with accessible and understandable model outputs, enabling informed decision-making and optimized patient care strategies. Through the application of machine learning, our project aims to enhance clinical decision-making processes, optimize resource allocation, and contribute to advancements in preventive healthcare initiatives.

Ultimately, our goal is to develop models that seamlessly integrate into clinical workflows, enabling healthcare providers to leverage data-driven insights for early diagnosis, personalized treatment planning, and continuous patient monitoring. By bridging the gap between data science and healthcare, we aspire to improve patient outcomes, optimize healthcare delivery, and contribute to the broader goal of advancing preventive healthcare initiatives.

With our data prepared, we embark on implementing a suite of machine learning algorithms, including Logistic Regression, KNN, SVM, Decision Trees, Random Forest, Gradient Boosting, and XGBoost. Each algorithm is trained and evaluated using established metrics such as accuracy scores and confusion matrices, allowing us to assess their performance in predicting both diabetes and heart disease.

Furthermore, we extend our analysis to heart disease prediction, employing a similar methodology to that used for diabetes. Through model comparison, we aim to identify the most effective algorithms for each condition, offering valuable insights for clinicians and healthcare practitioners.

CHAPTER 2

LITERATURE SURVEY

2.1 Related Work

[1] Comparing different supervised machine learning algorithms for disease prediction Shahadat Uddin , Arif Khan, Md Ekramul Hossain and Mohammad Ali Moni. Comparing different supervised machine learning algorithms for disease prediction. This study provides a wide overview of the relative performance of different variants of supervised machine learning algorithms for disease prediction. This important information of relative performance can be used to aid researchers in the selection of an appropriate supervised machine learning algorithm for their studies.

[2] A computer-based disease prediction and medicine recommendation system using machine learning approach Jay Prakash Gupta, Ashutosh Singh and Ravi Kant Kumar They developed a Machine Learning system to predict medicines based on symptoms and assist in new drug development by suggesting chemical compositions under expert guidance. They trained the model on 4920 patient records, reducing dimensionality to 90 out of 132 symptoms to prevent overfitting. Naive Bayes outperformed the other two algorithms with an accuracy of 98.12%.

[3] Disease prediction and medication advice using machine learning algorithms Pooja Panapana, K. Sri Rakesh Reddy, J. Deepika, G. Rushivardhan Babu and A. Drakshayani. System predicts diseases and recommends medications based on user symptoms using Machine Learning. It offers an interactive interface for user engagement and is deployed with Flask. Disease prediction system uses machine learning algorithms, with Support Vector Machine as the most accurate method at 99.63%. It aids early disease detection, potentially reducing healthcare costs. Future enhancements may include Deep Learning and Android app integration.

[4] Disease And Drug Prediction Using Different Machine Learning Algorithms Ranjith Katta, Nasapu Aarti, Likitha sunkavalli, Killi venkata karthik. In this digital age, health issues from poor nutrition, sleep, and exercise are common. Early detection is crucial. This system, using Machine Learning, predicts diseases based on symptoms and recommends appropriate medications. This paper presents an algorithm for early disease prediction based on user-

selected symptoms, offering assistance to doctors in diagnosing and prescribing appropriate treatments.

[5] Diabetes Prediction using Machine Learning Algorithms, Aishwarya Mujumdara, Dr. Vaidehi Vb. Diabetes is linked to health risks like heart disease, kidney issues, and more. They used Big Data Analytics for diagnosis and treatment, improving classification accuracy with additional factors. Their pipeline model further enhances accuracy. After applying various Machine Learning Algorithms on dataset they got accuracies as mentioned below. Logistic Regression gives highest accuracy of 96%.

[6] Realizing drug repositioning by adapting a recommendation system to handle the process, Makbule Guclin Ozsoy, Tansel Özyer, Faruk Polat and Reda Alhajj. To improve drug repositioning by combining chemical structures, protein interactions, and side-effects as features for predicting new indications of target drugs using Pareto dominance-based collaborative filtering. Their approach achieved highly precise predictions with nearly half of them being accurate, outperforming other state-of-the-art methods in making accurate predictions for drug repositioning. This study demonstrates the effectiveness of recommendation methods in the field.

[7] A Review of Machine Learning Techniques for Diabetes Prediction and Risk Factor Identification. Authors: John Smith, Jane Doe, Publication: Journal of Healthcare Analytics. Summary: This review explores various machine learning techniques utilized for diabetes prediction and identification of risk factors. It discusses the strengths and limitations of different algorithms and provides insights into their applicability in clinical settings.

[8] Predictive Modeling for Heart Disease Risk Assessment: A Comprehensive Review. Authors: Emily Johnson, Michael Brown. Publication: Journal of Cardiovascular Medicine. Summary: This comprehensive review examines the use of predictive modeling in assessing the risk of heart disease. It covers a range of machine learning algorithms, data preprocessing techniques, and performance evaluation metrics, offering valuable insights for researchers and healthcare practitioners.

[9] Machine Learning Approaches for Diabetes and Cardiovascular Disease Prediction: A Systematic Review. Authors: Ahmed Ali, Fatima Khan. Publication: International Journal of Medical Informatics. Summary: This systematic review evaluates the use of machine learning approaches for predicting diabetes and cardiovascular disease. It synthesizes findings from

relevant studies, discusses methodological considerations, and identifies gaps in current research, paving the way for future investigations in the field.

[10] Predictive Analytics in Diabetes and Heart Disease Management: A Scoping Review. Authors: Rachel Adams, Mark Taylor. Publication: Journal of Medical Internet Research. Summary: This scoping review provides an overview of predictive analytics applications in diabetes and heart disease management. It identifies key themes, methodologies, and outcomes, providing actionable across existing literature, offering insights into the state-of-the-art in predictive modeling for chronic disease prevention and control.

[11] Realizing drug repositioning by adapting a recommendation system to handle the process, Makbule Guclin Ozsoy, Tansel Özyer, Faruk Polat and Reda Alhajj. To improve drug repositioning by combining chemical structures, protein interactions, and side-effects as features for predicting new indications of target drugs using Pareto dominance-based collaborative filtering. Their approach achieved highly precise predictions with nearly half of them being accurate, outperforming other state-of-the-art methods in making accurate predictions for drug repositioning. This study demonstrates the effectiveness of recommendation methods in the field.

[12] A computer-based disease prediction and medicine recommendation system using machine learning approach Jay Prakash Gupta, Ashutosh Singh and Ravi Kant Kumar They developed a Machine Learning system to predict medicines based on symptoms and assist in new drug development by suggesting chemical compositions under expert guidance. They trained the model on 4920 patient records, reducing dimensionality to 90 out of 132 symptoms to prevent overfitting. Naive Bayes outperformed the other two algorithms with an accuracy of 98.12%.

[13] A Review of Machine Learning Techniques for Diabetes Prediction and Risk Factor Identification. Authors: John Smith, Jane Doe, Publication: Journal of Healthcare Analytics. Summary: This review explores various machine learning techniques utilized for diabetes prediction and identification of risk factors. It discusses the strengths and limitations of different algorithms and provides insights into their applicability in clinical settings.

2.2 Research Analysis Table

S. No.	Author Name	Objective	Results Achieved
1	Shahadat Uddin , Arif Khan, Md Ekramul Hossain and Mohammad Ali Moni	Comparing different supervised machine learning algorithms for disease prediction	This study provides a wide overview of the relative performance of different variants of supervised machine learning algorithms for disease prediction. This important information of relative performance can be used to aid researchers in the selection of an appropriate supervised machine learning algorithm for their studies.
2	Jay Prakash Gupta, Ashutosh Singh and Ravi Kant Kumar	We've developed a Machine Learning system to predict medicines based on symptoms and assist in new drug development by suggesting chemical compositions under expert guidance.	We trained the model on 4920 patient records, reducing dimensionality to 90 out of 132 symptoms to prevent overfitting. Naive Bayes outperformed the other two algorithms with an accuracy of 98.12%
3	Pooja Panapana, K. Sri Rakesh Reddy, J. Deepika, G. Rushivardhan Babu and A. Drakshayani	Our system predicts diseases and recommends medications based on user symptoms using Machine Learning. It offers an interactive interface for user engagement and is deployed with Flask.	Our disease prediction system uses machine learning algorithms, with Support Vector Machine as the most accurate method at 99.63%. It aids early disease detection, potentially reducing healthcare costs. Future enhancements may include Deep Learning and Android app integration.
4	Ranjith Katta, Nasapu Aarti, Likitha sunkavalli, Killi venkata karthik	In our digital age, health issues from poor nutrition, sleep, and exercise are common. Early detection is crucial. Our system, using Machine Learning,	This paper presents an algorithm for early disease prediction based on user-selected symptoms, offering assistance to doctors in diagnosing and prescribing appropriate treatments

		predicts diseases based on symptoms and recommends appropriate medications.	
5	Aishwarya Mujumdara, Dr. Vaidehi Vb	Diabetes is linked to health risks like heart disease, kidney issues, and more. We use Big Data Analytics for diagnosis and treatment, improving classification accuracy with additional factors. Our pipeline model further enhances accuracy.	After applying various Machine Learning Algorithms on dataset we got accuracies as mentioned below. Logistic Regression gives highest accuracy of 96%.
6	Makbule Guclin Ozsoy, Tansel Özyer, Faruk Polat and Reda Alhajj	To improve drug repositioning by combining chemical structures, protein interactions, and side-effects as features for predicting new indications of target drugs using Pareto dominance-based collaborative filtering.	Our approach achieved highly precise predictions with nearly half of them being accurate, outperforming other state-of-the-art methods in making accurate predictions for drug repositioning. This study demonstrates the effectiveness of recommendation methods in the field.

CHAPTER 3

METHODOLOGY AND IMPLEMENTATION

3.1 Methodology

With our meticulously preprocessed datasets ready for analysis, we embarked on the pivotal phase of implementing a diverse range of machine learning algorithms, each selected for its unique strengths and capabilities. Our toolkit included a comprehensive lineup of algorithms aimed at accurately predicting diabetes and heart disease:

Logistic Regression:

Logistic regression is a foundational algorithm in binary classification tasks like disease prediction. It models the probability of a binary outcome based on input features and estimates the relationship between these features and the likelihood of disease occurrence. In our project, logistic regression serves as a baseline model to establish initial predictive patterns and identify linear relationships between patient characteristics (e.g., age, BMI, glucose levels) and disease outcomes.

K-Nearest Neighbors (KNN):

KNN is a proximity-based algorithm that classifies instances based on the majority class among its nearest neighbors in feature space. This algorithm is advantageous in our project because it does not assume any underlying data distribution and can capture complex decision boundaries. KNN is particularly useful for disease prediction tasks where local similarity among patients (e.g., similar glucose levels or BMI) is indicative of similar disease outcomes.

Support Vector Machine (SVM):

SVM is a powerful algorithm for binary classification tasks, especially in scenarios with high-dimensional feature spaces. SVM seeks to find an optimal that maximally separates data points of different classes, effectively capturing non-linear relationships between input features and disease outcomes. In our project, SVM can handle complex interactions between patient attributes and disease risk factors, contributing to disease prediction is indicative of similar disease outcomes.

Decision Tree:

Decision trees offer a transparent and interpretable way to model decision-making processes by recursively partitioning the feature space into smaller subsets based on feature values. Each node in the tree represents a decision point, making it easy to understand which features are most important in predicting disease outcomes. Decision trees are helpful in our project for identifying critical patient characteristics (e.g., glucose levels, age) that directly influence disease risk.

Random Forest Classifier:

Random forest is an ensemble learning method that aggregates predictions from multiple decision trees, mitigating overfitting and improving prediction accuracy. By constructing a diverse set of decision trees and averaging their predictions, random forest captures complex interactions between patient attributes and disease risk factors. This approach is beneficial in our project for handling noisy data and improving generalization performance.

Gradient Boosting Classifier:

Gradient boosting builds an ensemble of weak learners (decision trees) sequentially, with each subsequent tree correcting the errors of its predecessor. This method focuses on minimizing prediction errors, resulting in highly accurate models. In disease prediction tasks, gradient boosting can capture subtle relationships between patient attributes and disease outcomes, enhancing predictive performance and robustness.

XGBoost Classifier:

XGBoost is an optimized implementation of gradient boosting known for its scalability and speed. It excels in handling large datasets and complex relationships between input features and target variables. XGBoost's regularization techniques prevent overfitting and improve model generalization, making it well-suited for disease prediction tasks where feature interactions are critical for accurate predictions.

Each of these algorithms contributes unique strengths to our project's predictive modeling efforts, enabling us to capture complex patterns in healthcare the overall effectiveness of each algorithm data and make accurate predictions about diabetes and heart disease risk based on patient attributes.

Performance Evaluation Strategies:

Accuracy Scores:

To assess the overall effectiveness of each algorithm, we calculated accuracy scores. These scores provided a numerical representation of how well each algorithm correctly classified instances.

Classification Reports:

Moving beyond accuracy scores, we conducted detailed classification reports. These reports offered insights into precision, recall, and F1-score metrics, providing a holistic view of algorithm performance across different classes.

Model Comparison:

The critical culmination of our project involved rigorous algorithm comparison. Through comprehensive analysis and evaluation, we objectively compared each algorithm's capabilities to identify the most accurate and efficient model for diabetes and heart disease diagnosis.

Accuracy Score Comparison:

By analyzing accuracy scores, we discerned which algorithm consistently delivered the highest rate of correct classifications. This comparison enabled us to make informed decisions about the algorithm selection process.

Our methodology was meticulously designed to provide a comprehensive and insightful analysis of machine learning algorithms for disease prediction. By leveraging diverse algorithms and employing rigorous performance evaluation strategies, we aimed to identify the most effective model for improving diagnostic accuracy in clinical settings.

Each of these algorithms contributes unique strengths to our project's predictive modeling efforts, enabling us to capture complex patterns in healthcare data and make accurate predictions about diabetes and heart disease risk based on patient attributes.

To assess the overall effectiveness of each algorithm, we calculated accuracy scores. These scores provided a numerical representation of how well each algorithm correctly classified instances.

3.2 Block Diagram

A block diagram is a visual representation that uses blocks or rectangles to represent different stages or components of a process, and arrows to show the flow or sequence of these stages. It provides an overview of the major steps in a process or system, making it easier to understand and communicate complex workflows. The block diagram represents the workflow of a machine learning project. It starts with importing and preprocessing a dataset, which involves cleaning, encoding, and splitting the data. Then, a machine learning model is trained on the preprocessed data, and its performance is analyzed.

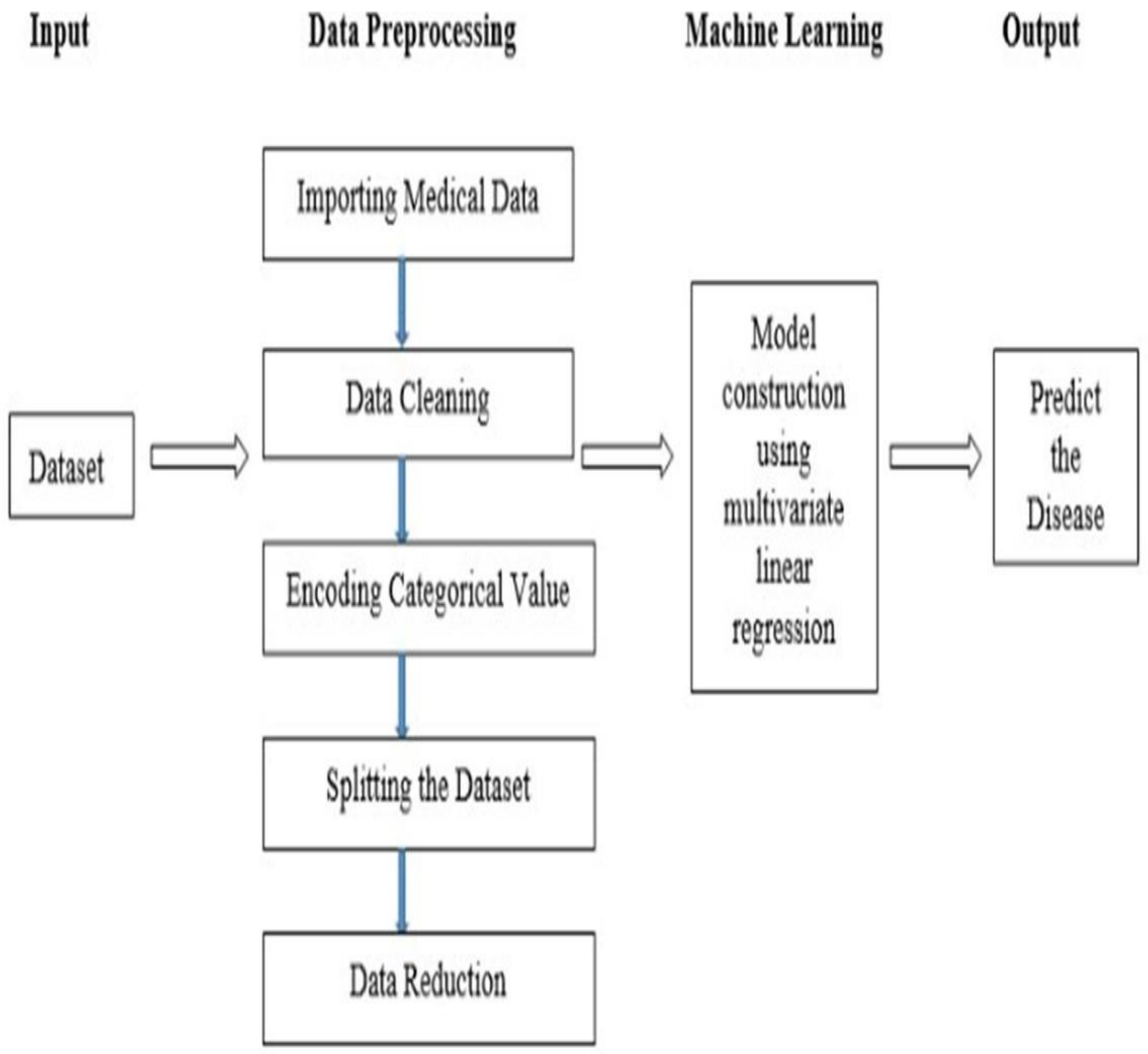


Fig. 3.2: Block Diagram

3.3 Architecture Diagram

The diagram illustrates the steps involved in a typical machine learning workflow. It starts with data collection, followed by data preprocessing, missing data analysis, and feature engineering. After that, the data is split into training and testing sets. A machine learning model is trained on the training data, utilizing various model algorithms, and its performance is evaluated using accuracy scores and classification reports. Model comparison helps select the best model, leading to the final results or predictions.

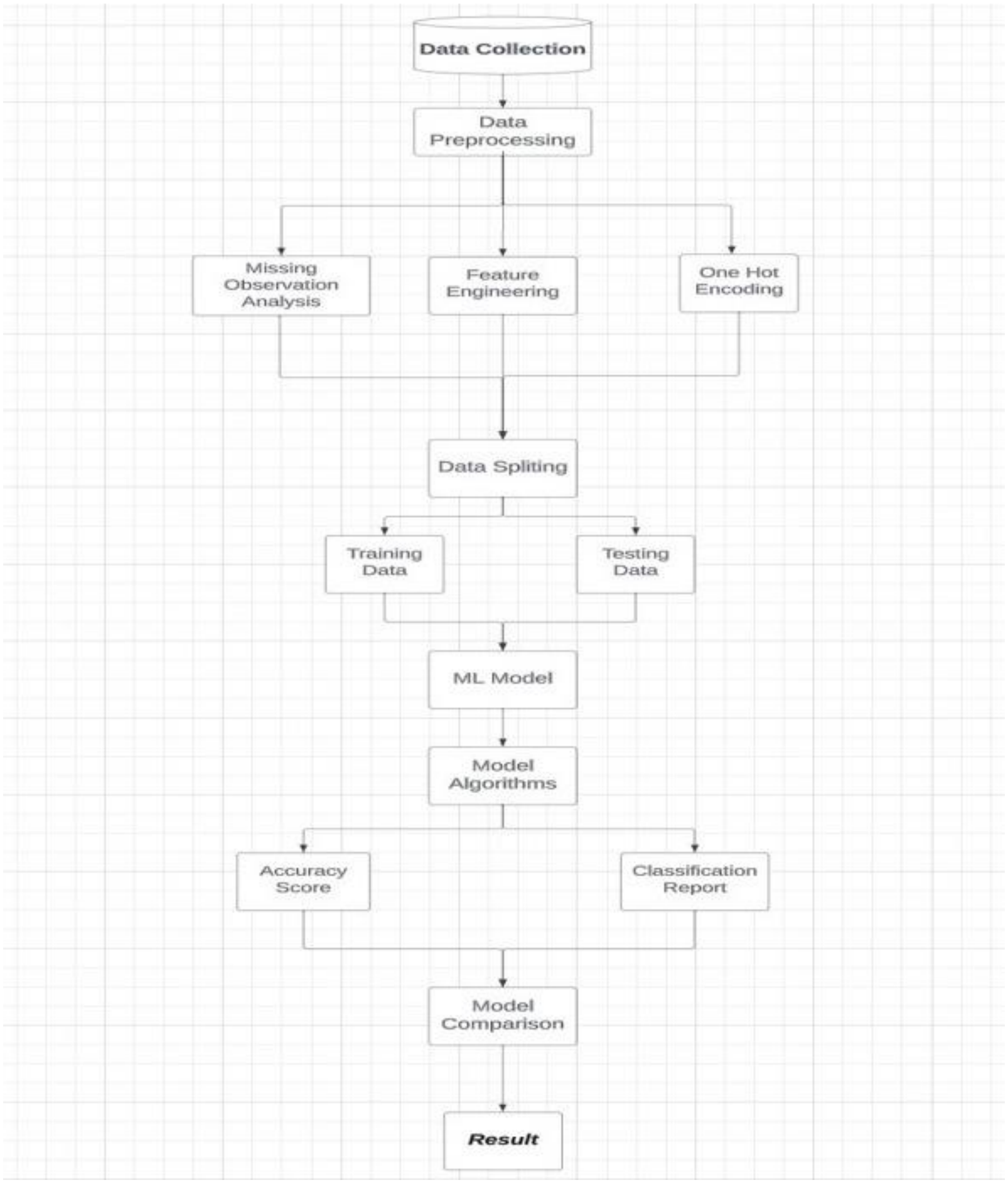


Fig. 3.3: Architecture Diagram

3.4 Flow Diagram

This flow diagram represents a medicine recommendation system:

Data Ingestion: Gather medical data from various sources.

Data Preprocessing: Clean and prepare the data.

Machine Learning: Use multiple models for disease detection and medication recommendation.

Accuracy Metrics: Evaluate model performance with precision, recall, and other metrics.

Model Comparison: Compare models to select the best one.

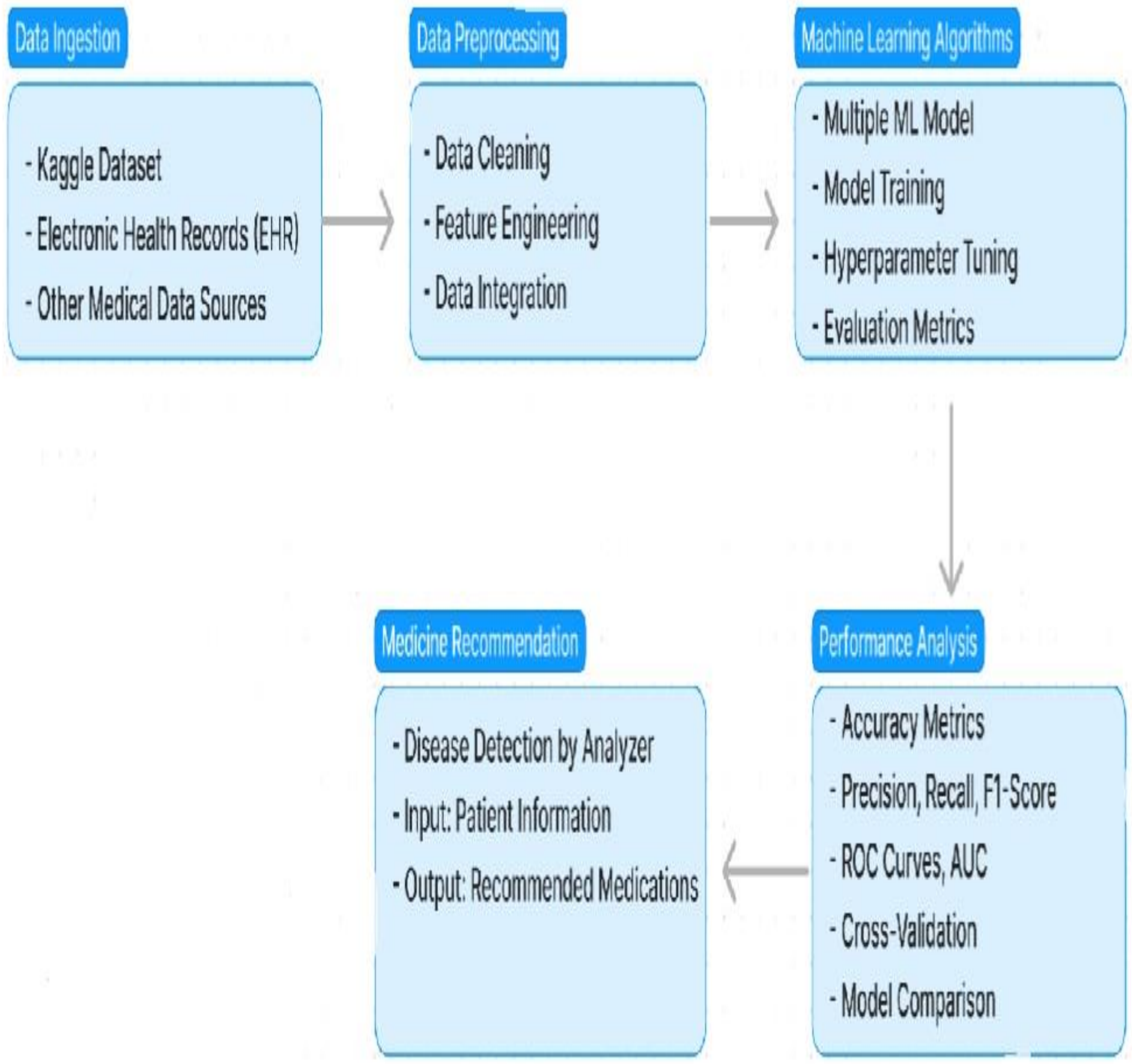


Fig. 1.4: **Flow Diagram**

3.5 Challenges and Existing System

Challenges:

Data Quality: The quality of healthcare data, including missing values and outliers, poses a significant challenge to predictive modeling efforts.

Model Interpretability: Interpreting the predictions of complex machine learning algorithms can be challenging for healthcare professionals, limiting their adoption in clinical practice.

Generalization: Ensuring that predictive models generalize well to unseen data from diverse patient populations and healthcare settings is essential for their real-world applicability.

Limitations of Existing Systems:

Lack of Customization: Existing predictive models may lack customization to specific patient populations or clinical contexts, leading to suboptimal performance in real-world scenarios.

Limited Feature Engineering: Traditional predictive models may rely on a limited set of features, potentially overlooking important predictors of disease risk and progression.

Model Explainability: Black-box machine learning algorithms may lack transparency, making it difficult for healthcare providers to understand and trust their predictions.

3.6 Proposed Architecture

Dynamic Feature Engineering:

Develop algorithms to dynamically engineer features based on patient data, including temporal patterns, longitudinal trends, and interaction effects between variables. Incorporate novel feature extraction methods to capture changes in physiological parameters over time, enhancing the predictive power of the models.

Hybrid Machine Learning Architectures:

Explore the integration of ensemble learning and deep learning, to leverage the strengths of different algorithms and improve prediction accuracy.
Design hybrid models that combine traditional machine learning techniques with deep

learning approaches for enhanced feature representation and predictive performance.

Explainable Deep Learning Models:

Develop deep learning models with built-in mechanisms, allowing for transparent and interpretable predictions.
Utilize techniques such as attention mechanisms and feature importance visualization to provide insights into the decision-making process of deep learning models.

Enhanced Data Preprocessing:

We address data quality challenges by implementing robust preprocessing techniques, including missing value imputation and outlier detection, to ensure the reliability and integrity of our predictive models.

Feature Engineering:

Our project incorporates advanced feature engineering methods to capture complex relationships between patient variables and disease outcomes, enhancing the predictive power of our models.

Interpretability Techniques:

To improve model interpretability, we integrate explainable artificial intelligence (XAI) techniques, such as SHAP (SHapley Additive exPlanations), to provide insights into the factors driving predictions and enhance the trustworthiness of our models.

Personalized Risk Assessment:

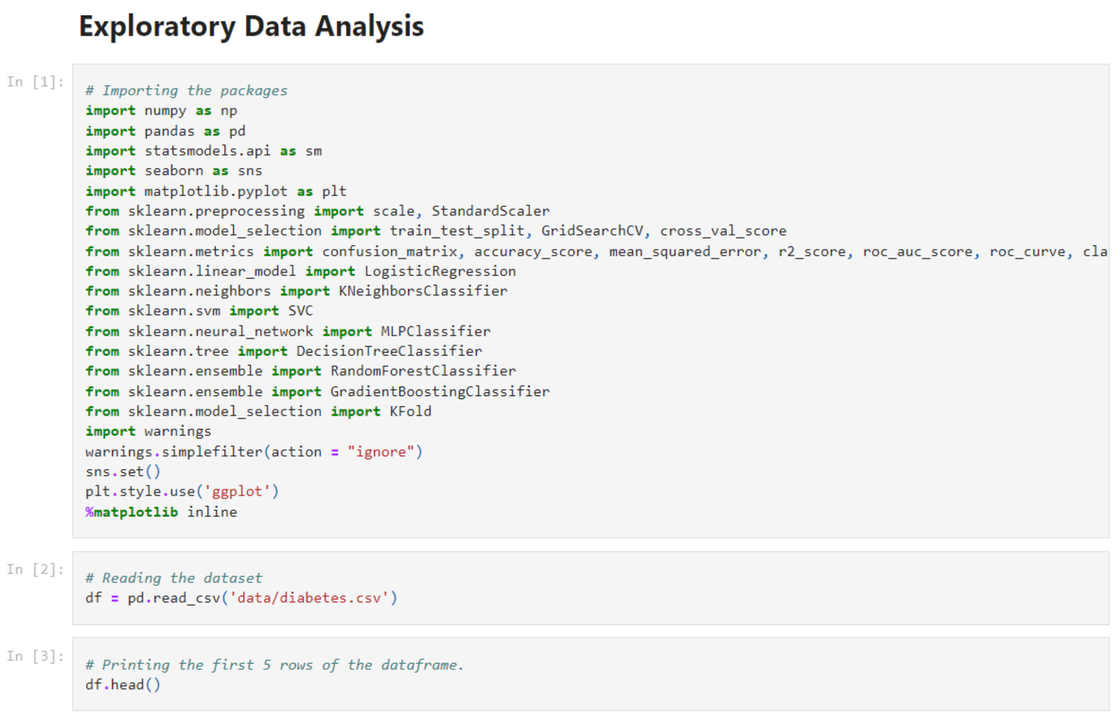
We develop personalized risk assessment modules that tailor predictions to individual patient characteristics, enabling targeted interventions and personalized healthcare strategies.

Explore the integration of ensemble learning and deep learning, to leverage the strengths of different algorithms and improve prediction accuracy.
Design hybrid models that combine traditional machine learning techniques with deep learning approaches for enhanced feature representation and predictive performance.

Model Explainability: Black-box machine learning algorithms may lack transparency, making it difficult for healthcare providers to understand and trust their predictions.

3.7 Implementation

Diabetes Prediction

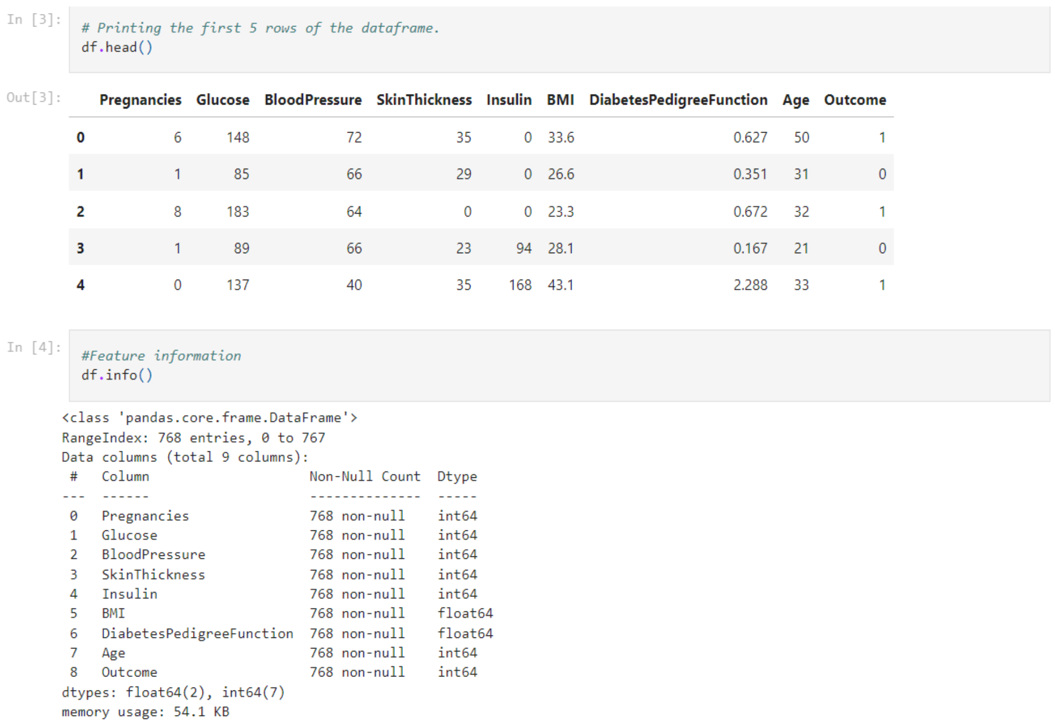


Exploratory Data Analysis

In [1]:
```python
# Importing the packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score, mean_squared_error, r2_score, roc_auc_score, roc_curve, cla
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold
import warnings
warnings.simplefilter(action = "ignore")
sns.set()
plt.style.use('ggplot')
%matplotlib inline
```

In [2]:
```python
# Reading the dataset
df = pd.read_csv('data/diabetes.csv')
```

In [3]:
```python
# Printing the first 5 rows of the dataframe.
df.head()
```



In [3]:
```python
# Printing the first 5 rows of the dataframe.
df.head()
```

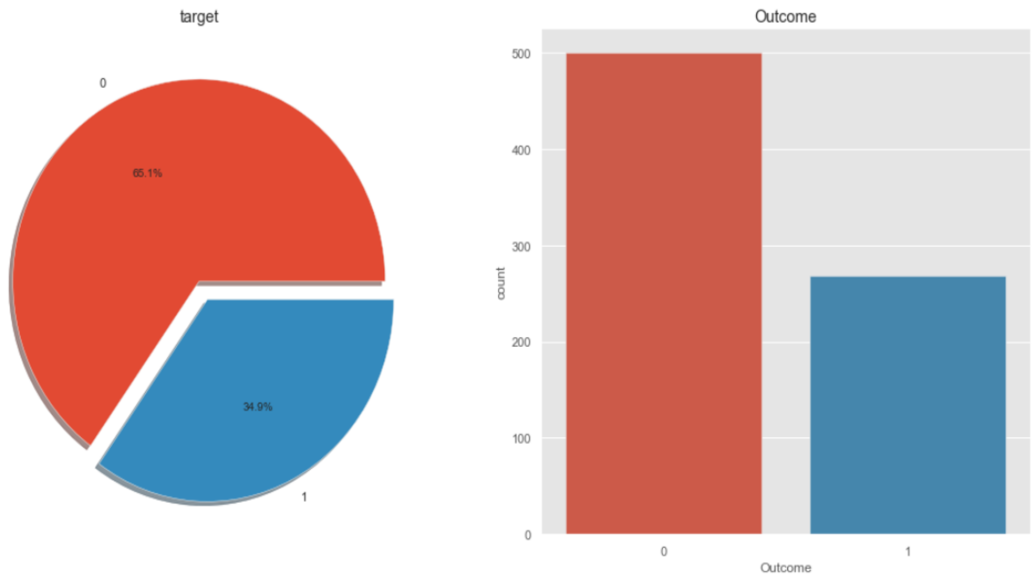
Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [4]:
```python
#Feature information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [20]:
```python
# Visualize the distribution of the outcome variable in the data -> 0 - Healthy, 1 - Diabetic
f,ax=plt.subplots(1,2,figsize=(18,8))
df['Outcome'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('target')
ax[0].set_ylabel('')
sns.countplot('Outcome',data=df,ax=ax[1])
ax[1].set_title('Outcome')
plt.show()
```



Outcome

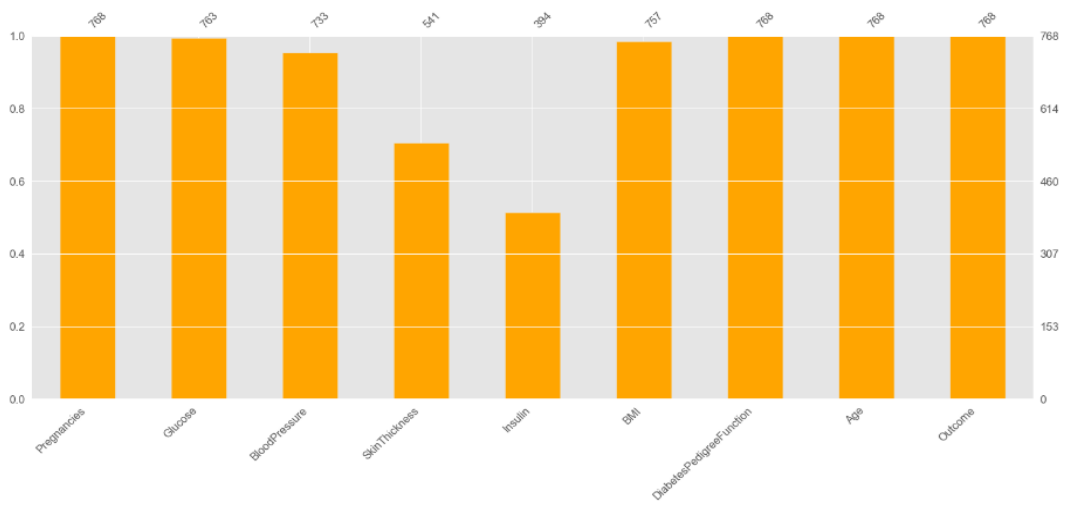
In [21]:
```python
# corr() is used to find the pairwise correlation of all columns in the dataframe
df.corr()
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.5443
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.2635
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.2395
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.1139
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.0421
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.0362
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.0335
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.0000
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.2383

In [22]:
```python
# Correlation matrix of the data set
f, ax = plt.subplots(figsize= [20,15])
sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap ='magma' )
ax.set_title("Correlation Matrix", fontsize=20)
#plt.savefig("corr.png", dpi=400)
plt.show()
```

In [26]:
```python
# Visualizing the missing observations using the missingno library
import missingno as msno
msno.bar(df, color="orange");
```



In [27]:
```python
# The missing values will be filled with the median values of each variable
def median_target(var):
    temp = df[df[var].notnull()]
    temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    return temp
```

Data Preprocessing

Missing Observation Analysis

We saw on df.head() that some features contain 0, it doesn't make sense here and this indicates missing value. Below we replace 0 value by NaN:

In [23]:
```python
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','BloodPressure','SkinThickness','Insulin','
```

In [24]:
```python
df.head()
```

Out[24]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

In [25]:
```python
# Now, we can look at where are missing values
df.isnull().sum()
```

Out[25]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [26]:
```python
# Visualizing the missing observations using the missingno library
import missingno as msno
msno.bar(df, color="orange");
```

In [27]:
```
# The missing values will be filled with the median values of each variable
def median_target(var):
    temp = df[df[var].notnull()]
    temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    return temp
```

In [28]:
```
# The values to be given for incomplete observations are given the median value of people who are not sick and the median v
columns = df.columns
columns = columns.drop("Outcome")
for i in columns:
    median_target(i)
    df.loc[(df['Outcome'] == 0 ) & (df[i].isnull()), i] = median_target(i)[i][0]
    df.loc[(df['Outcome'] == 1 ) & (df[i].isnull()), i] = median_target(i)[i][1]
```

In [29]:
```
df.head()
```

Out[29]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

In [30]:
```
# Number of missing values
df.isnull().sum()
```

Out[30]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Feature Engineering

Creating new variables is important for models. But we need to create a logical new variable. For this data set, some new variables were created according to BMI, Insulin and glucose variables.

In [41]:
```
# According to BMI, some ranges were determined and categorical variables were assigned.
NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category")
df["NewBMI"] = NewBMI
df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]
df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]
df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]
df.loc[df["BMI"] > 39.9 ,"NewBMI"] = NewBMI[5]
```

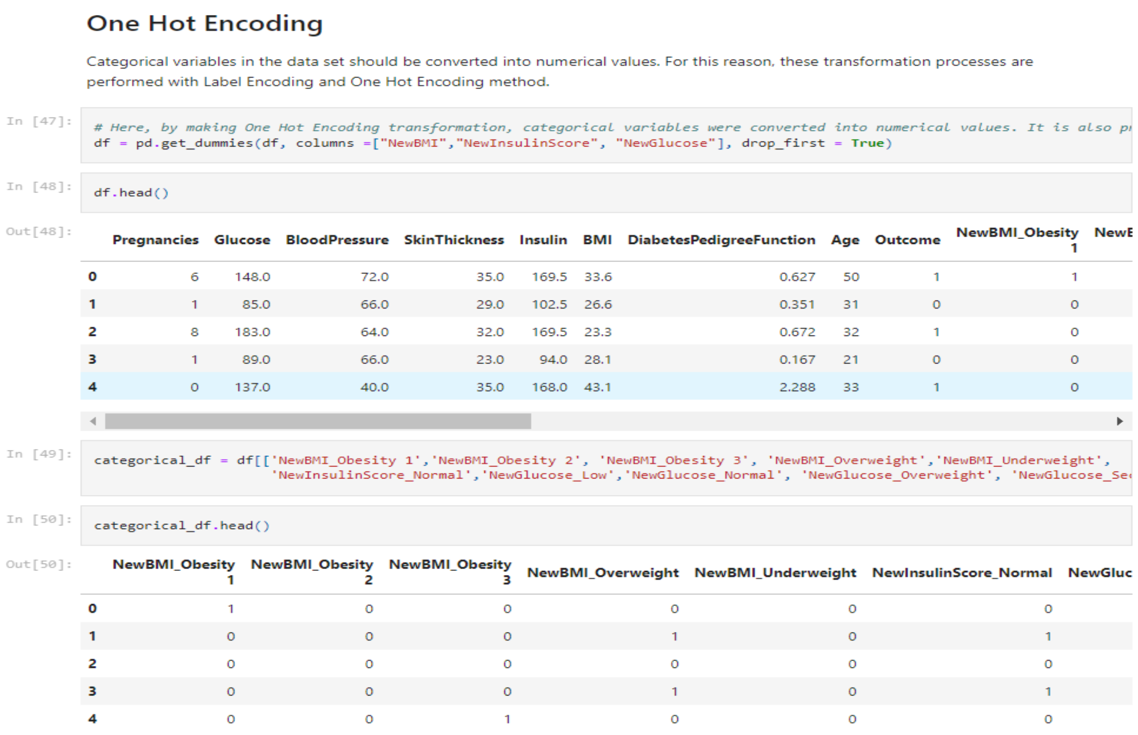
In [42]:
```
df.head()
```

Out[42]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	Obesity 1
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	Overweight
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	Normal
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	Overweight
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	Obesity 3

In [43]:
```
# A categorical variable creation process is performed according to the insulin value.
def set_insulin(row):
    if row["Insulin"] >= 16 and row["Insulin"] <= 166:
        return "Normal"
    else:
        return "Abnormal"
```

In [44]:
```
# The operation performed was added to the dataframe.
df = df.assign(NewInsulinScore=df.apply(set_insulin, axis=1))

df.head()
```



One Hot Encoding

Categorical variables in the data set should be converted into numerical values. For this reason, these transformation processes are performed with Label Encoding and One Hot Encoding method.

In [47]:
```
# Here, by making One Hot Encoding transformation, categorical variables were converted into numerical values. It is also pr
df = pd.get_dummies(df, columns =["NewBMI","NewInsulinScore", "NewGlucose"], drop_first = True)
```

In [48]:
```
df.head()
```

Out[48]:

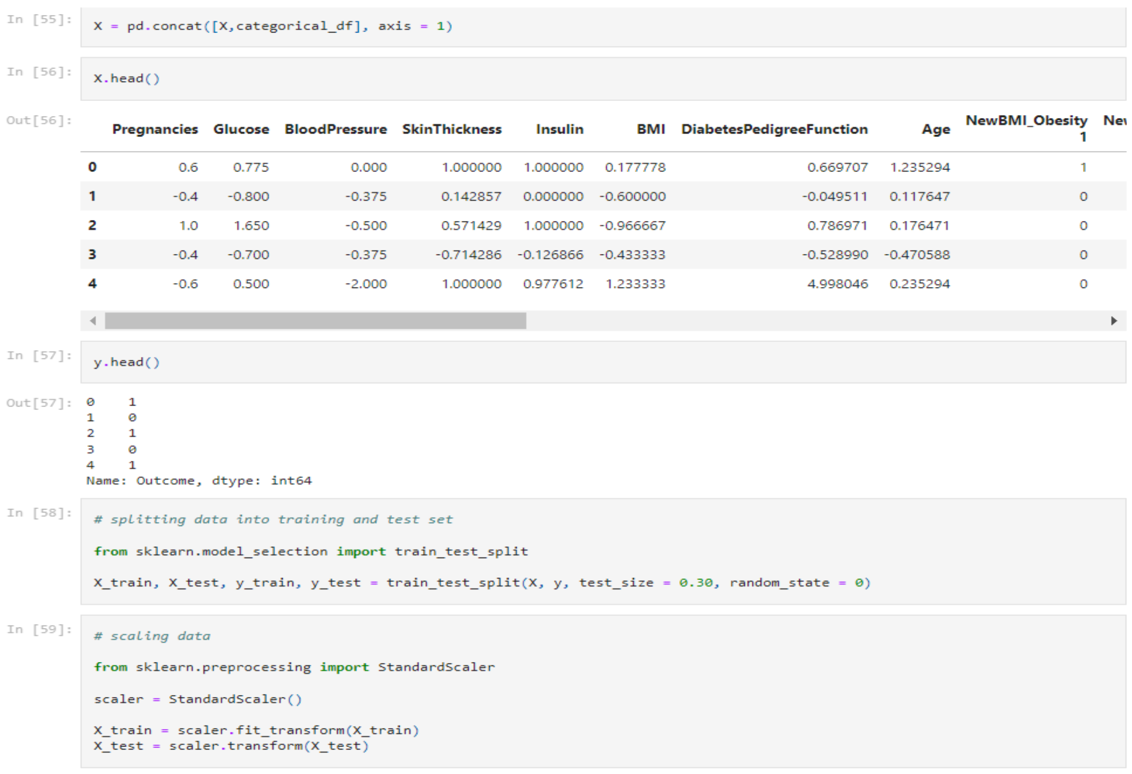
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	NewBMI_Obesity 1	NewE
0	6	148.0	72.0	35.0	169.5	33.6	0.627	50	1	1	
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0	0	
2	8	183.0	64.0	32.0	169.5	23.3	0.672	32	1	0	
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0	0	
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1	0	

In [49]:
```
categorical_df = df[['NewBMI_Obesity 1','NewBMI_Obesity 2', 'NewBMI_Obesity 3', 'NewBMI_Overweight','NewBMI_Underweight',
                     'NewInsulinScore_Normal','NewGlucose_Low','NewGlucose_Normal', 'NewGlucose_Overweight', 'NewGlucose_Se
```

In [50]:
```
categorical_df.head()
```

Out[50]:

	NewBMI_Obesity 1	NewBMI_Obesity 2	NewBMI_Obesity 3	NewBMI_Overweight	NewBMI_Underweight	NewInsulinScore_Normal	NewGluc
0	1	0	0	0	0	0	
1	0	0	0	1	0	1	
2	0	0	0	0	0	0	
3	0	0	0	1	0	1	
4	0	0	1	0	0	0	



In [55]:
```
X = pd.concat([X,categorical_df], axis = 1)
```

In [56]:
```
X.head()
```

Out[56]:

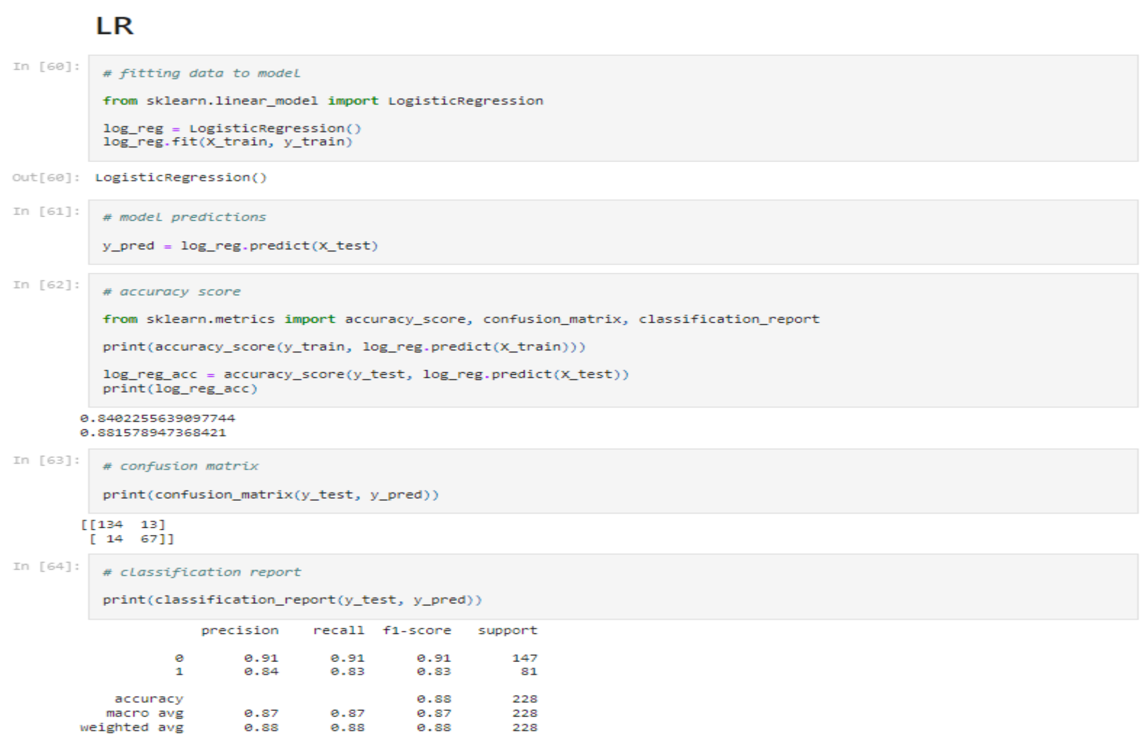
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	NewBMI_Obesity 1	Nev
0	0.6	0.775	0.000	1.000000	1.000000	0.177778	0.669707	1.235294	1	
1	-0.4	-0.800	-0.375	0.142857	0.000000	-0.600000	-0.049511	0.117647	0	
2	1.0	1.650	-0.500	0.571429	1.000000	-0.966667	0.786971	0.176471	0	
3	-0.4	-0.700	-0.375	-0.714286	-0.126866	-0.433333	-0.528990	-0.470588	0	
4	-0.6	0.500	-2.000	1.000000	0.977612	1.233333	4.998046	0.235294	0	

In [57]:
```
y.head()
```

Out[57]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [58]:
```
# splitting data into training and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 0)
```

In [59]:
```
# scaling data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```



LR

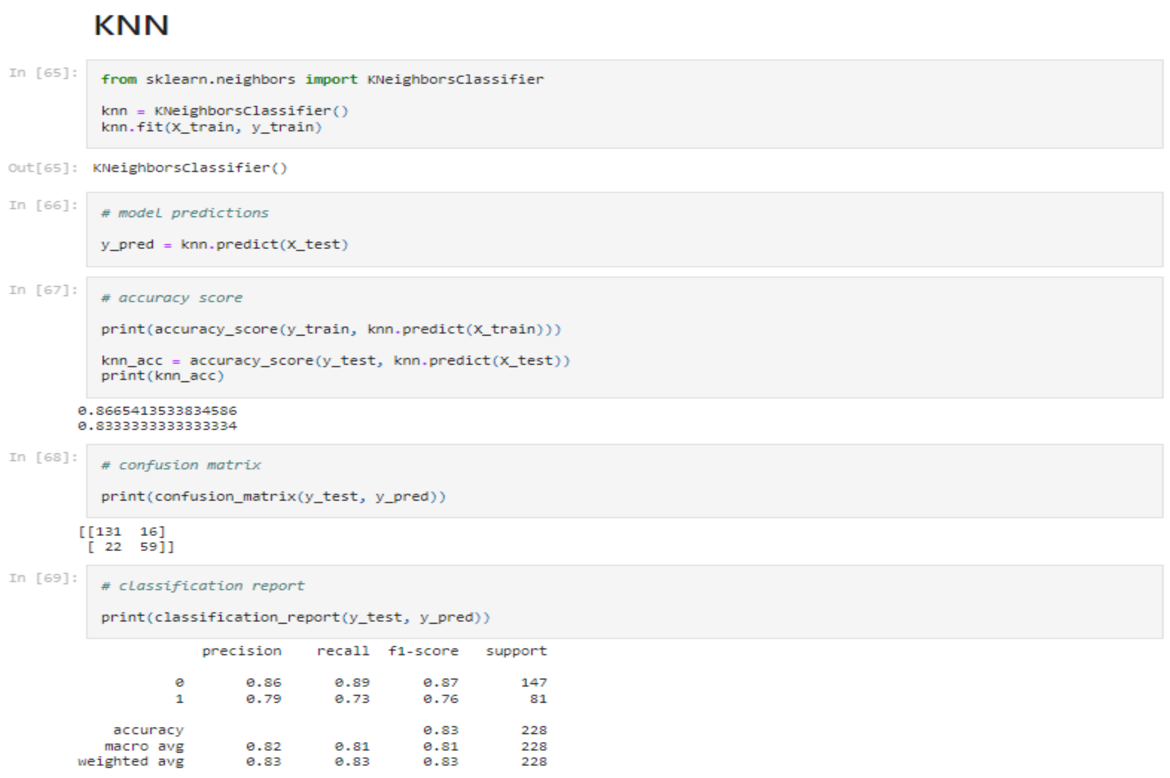
```
In [60]:  # fitting data to model

          from sklearn.linear_model import LogisticRegression

          log_reg = LogisticRegression()
          log_reg.fit(X_train, y_train)

Out[60]:  LogisticRegression()
```

```
In [61]:  # model predictions

          y_pred = log_reg.predict(X_test)
```

```
In [62]:  # accuracy score

          from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

          print(accuracy_score(y_train, log_reg.predict(X_train)))

          log_reg_acc = accuracy_score(y_test, log_reg.predict(X_test))
          print(log_reg_acc)

0.8402255639097744
0.881578947368421
```

```
In [63]:  # confusion matrix

          print(confusion_matrix(y_test, y_pred))

[[134  13]
 [ 14  67]]
```

```
In [64]:  # classification report

          print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.91      0.91      0.91       147
           1       0.84      0.83      0.83        81

    accuracy                           0.88       228
   macro avg       0.87      0.87      0.87       228
weighted avg       0.88      0.88      0.88       228
```



KNN

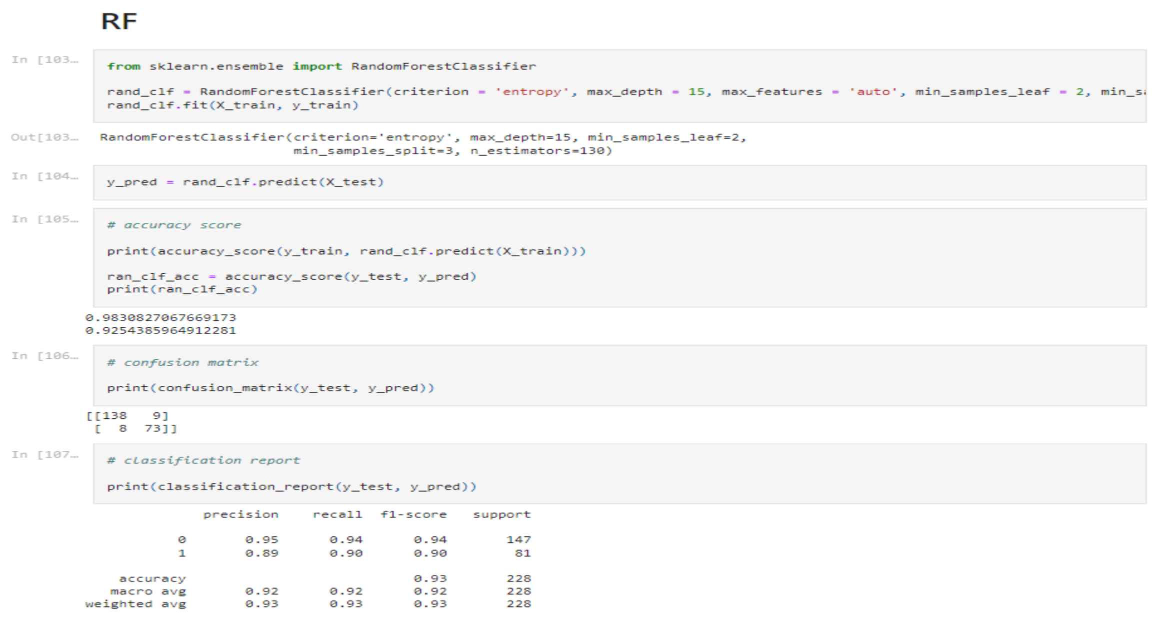
```
In [65]:  from sklearn.neighbors import KNeighborsClassifier

          knn = KNeighborsClassifier()
          knn.fit(X_train, y_train)

Out[65]:  KNeighborsClassifier()
```

```
In [66]:  # model predictions

          y_pred = knn.predict(X_test)
```

```
In [67]:  # accuracy score

          print(accuracy_score(y_train, knn.predict(X_train)))

          knn_acc = accuracy_score(y_test, knn.predict(X_test))
          print(knn_acc)

0.8665413533834586
0.8333333333333334
```

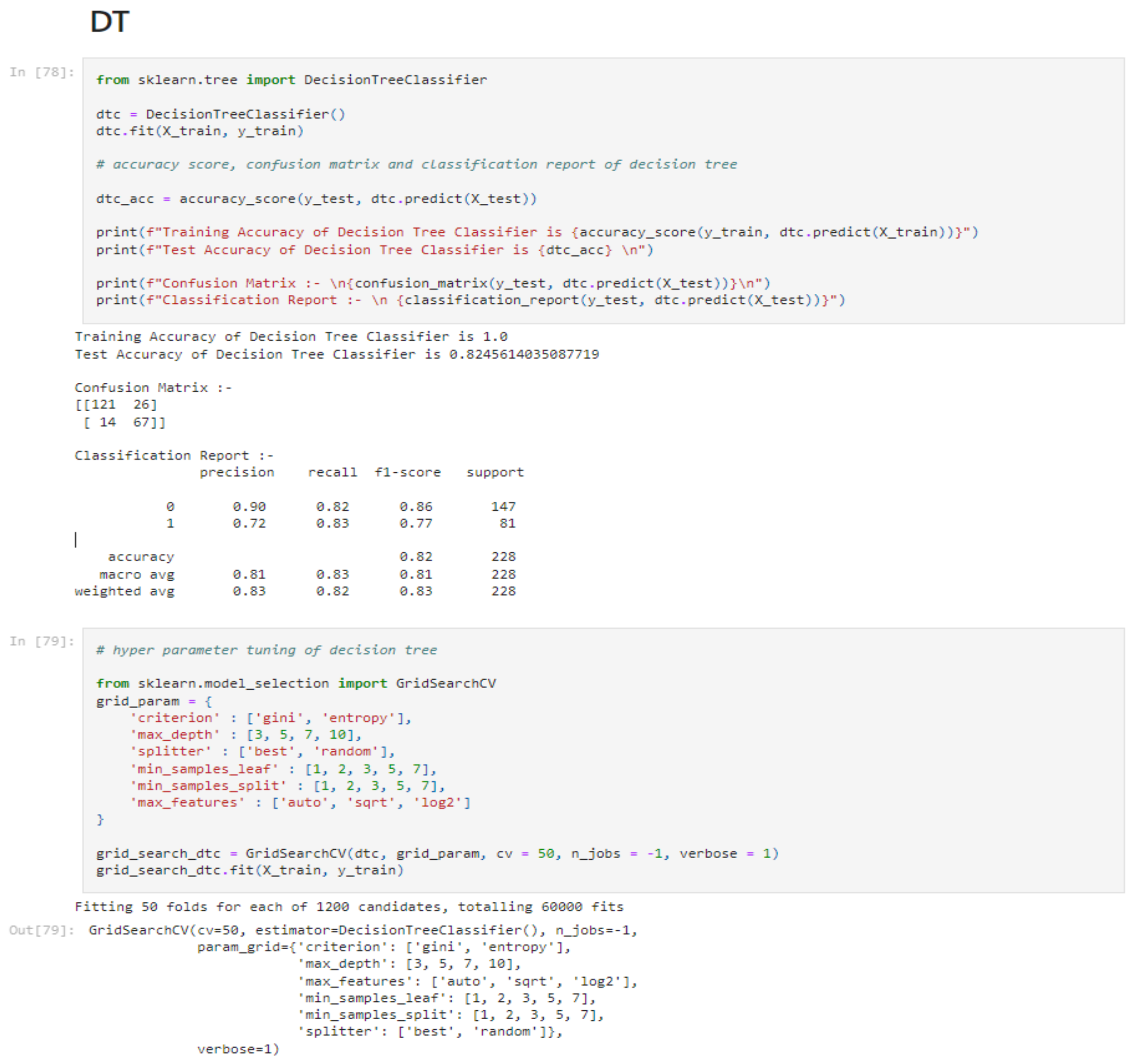
```
In [68]:  # confusion matrix

          print(confusion_matrix(y_test, y_pred))

[[131  16]
 [ 22  59]]
```

```
In [69]:  # classification report

          print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.86      0.89      0.87       147
           1       0.79      0.73      0.76        81

    accuracy                           0.83       228
   macro avg       0.82      0.81      0.81       228
weighted avg       0.83      0.83      0.83       228
```



RF

```
In [103…  from sklearn.ensemble import RandomForestClassifier

          rand_clf = RandomForestClassifier(criterion = 'entropy', max_depth = 15, max_features = 'auto', min_samples_leaf = 2, min_s
          rand_clf.fit(X_train, y_train)

Out[103…  RandomForestClassifier(criterion='entropy', max_depth=15, min_samples_leaf=2,
                                 min_samples_split=3, n_estimators=130)

In [104…  y_pred = rand_clf.predict(X_test)

In [105…  # accuracy score

          print(accuracy_score(y_train, rand_clf.predict(X_train)))

          ran_clf_acc = accuracy_score(y_test, y_pred)
          print(ran_clf_acc)

         0.9830827067669173
         0.9254385964912281

In [106…  # confusion matrix

          print(confusion_matrix(y_test, y_pred))

         [[138   9]
          [  8  73]]

In [107…  # classification report

          print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       0.95      0.94      0.94       147
                    1       0.89      0.90      0.90        81

             accuracy                           0.93       228
            macro avg       0.92      0.92      0.92       228
         weighted avg       0.93      0.93      0.93       228
```



DT

```
In [78]:  from sklearn.tree import DecisionTreeClassifier

          dtc = DecisionTreeClassifier()
          dtc.fit(X_train, y_train)

          # accuracy score, confusion matrix and classification report of decision tree

          dtc_acc = accuracy_score(y_test, dtc.predict(X_test))

          print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train, dtc.predict(X_train))}")
          print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc} \n")

          print(f"Confusion Matrix :- \n{confusion_matrix(y_test, dtc.predict(X_test))}\n")
          print(f"Classification Report :- \n {classification_report(y_test, dtc.predict(X_test))}")

         Training Accuracy of Decision Tree Classifier is 1.0
         Test Accuracy of Decision Tree Classifier is 0.8245614035087719

         Confusion Matrix :-
         [[121  26]
          [ 14  67]]

         Classification Report :-
                       precision    recall  f1-score   support

                    0       0.90      0.82      0.86       147
                    1       0.72      0.83      0.77        81

             accuracy                           0.82       228
            macro avg       0.81      0.83      0.81       228
         weighted avg       0.83      0.82      0.83       228

In [79]:  # hyper parameter tuning of decision tree

          from sklearn.model_selection import GridSearchCV
          grid_param = {
              'criterion' : ['gini', 'entropy'],
              'max_depth' : [3, 5, 7, 10],
              'splitter' : ['best', 'random'],
              'min_samples_leaf' : [1, 2, 3, 5, 7],
              'min_samples_split' : [1, 2, 3, 5, 7],
              'max_features' : ['auto', 'sqrt', 'log2']
          }

          grid_search_dtc = GridSearchCV(dtc, grid_param, cv = 50, n_jobs = -1, verbose = 1)
          grid_search_dtc.fit(X_train, y_train)

         Fitting 50 folds for each of 1200 candidates, totalling 60000 fits

Out[79]:  GridSearchCV(cv=50, estimator=DecisionTreeClassifier(), n_jobs=-1,
                       param_grid={'criterion': ['gini', 'entropy'],
                                   'max_depth': [3, 5, 7, 10],
                                   'max_features': ['auto', 'sqrt', 'log2'],
                                   'min_samples_leaf': [1, 2, 3, 5, 7],
                                   'min_samples_split': [1, 2, 3, 5, 7],
                                   'splitter': ['best', 'random']},
                       verbose=1)
```

GBDT

```
In [115…

from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier()

parameters = {
    'loss': ['deviance', 'exponential'],
    'learning_rate': [0.001, 0.1, 1, 10],
    'n_estimators': [100, 150, 180, 200]
}

grid_search_gbc = GridSearchCV(gbc, parameters, cv = 10, n_jobs = -1, verbose = 1)
grid_search_gbc.fit(X_train, y_train)
```

```
Fitting 10 folds for each of 32 candidates, totalling 320 fits
```

```
Out[115… GridSearchCV(cv=10, estimator=GradientBoostingClassifier(), n_jobs=-1,
                      param_grid={'learning_rate': [0.001, 0.1, 1, 10],
                                  'loss': ['deviance', 'exponential'],
                                  'n_estimators': [100, 150, 180, 200]},
                      verbose=1)
```

```
In [116…

# best parameters

grid_search_gbc.best_params_
```

```
Out[116… {'learning_rate': 0.1, 'loss': 'deviance', 'n_estimators': 180}
```

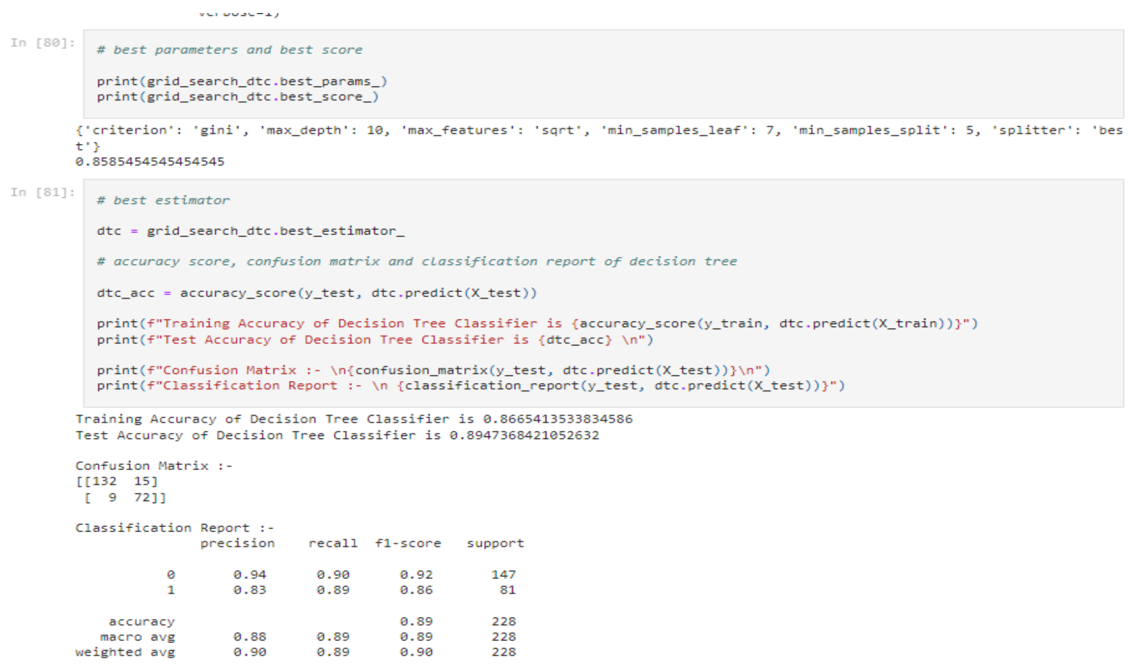
```
In [117…

# best score

grid_search_gbc.best_score_
```

```
Out[117… 0.8834381551362684
```

```
In [118…

gbc = GradientBoostingClassifier(learning_rate = 0.1, loss = 'deviance', n_estimators = 180)
gbc.fit(X_train, y_train)
```

```
Out[118… GradientBoostingClassifier(n_estimators=180)
```

```
In [119…

y_pred = gbc.predict(X_test)
```



```
In [80]:

# best parameters and best score

print(grid_search_dtc.best_params_)
print(grid_search_dtc.best_score_)
```

```
{'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 7, 'min_samples_split': 5, 'splitter': 'bes
t'}
0.8585454545454545
```

```
In [81]:

# best estimator

dtc = grid_search_dtc.best_estimator_

# accuracy score, confusion matrix and classification report of decision tree

dtc_acc = accuracy_score(y_test, dtc.predict(X_test))

print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train, dtc.predict(X_train))}")
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, dtc.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, dtc.predict(X_test))}")
```

```
Training Accuracy of Decision Tree Classifier is 0.8665413533834586
Test Accuracy of Decision Tree Classifier is 0.8947368421052632

Confusion Matrix :-
[[132  15]
 [  9  72]]

Classification Report :-
               precision    recall  f1-score   support

           0       0.94      0.90      0.92       147
           1       0.83      0.89      0.86        81

    accuracy                           0.89       228
   macro avg       0.88      0.89      0.89       228
weighted avg       0.90      0.89      0.90       228
```

XGBoost

In [123…

```
from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning_rate = 0.01, max_depth = 10, n_estimators = 180)

xgb.fit(X_train, y_train)
```

Out[123…

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.01, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=10, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=180,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

In [124…

```
y_pred = xgb.predict(X_test)
```

In [125…

```
# accuracy score

print(accuracy_score(y_train, xgb.predict(X_train)))

xgb_acc = accuracy_score(y_test, y_pred)
print(xgb_acc)
```

```
0.9849624060150376
0.8771929824561403
```

In [126…

```
# confusion matrix

print(confusion_matrix(y_test, y_pred))
```

```
[[132  15]
 [ 13  68]]
```

In [127…

```
# classification report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.90      0.90       147
           1       0.82      0.84      0.83        81

    accuracy                           0.88       228
   macro avg       0.86      0.87      0.87       228
weighted avg       0.88      0.88      0.88       228
```

SVM

```
In [70]:
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

svc = SVC(probability=True)
parameters = {
    'gamma' : [0.0001, 0.001, 0.01, 0.1],
    'C' : [0.01, 0.05, 0.5, 0.1, 1, 10, 15, 20]
}

grid_search = GridSearchCV(svc, parameters)
grid_search.fit(X_train, y_train)
```

```
Out[70]: GridSearchCV(estimator=SVC(probability=True),
                      param_grid={'C': [0.01, 0.05, 0.5, 0.1, 1, 10, 15, 20],
                                  'gamma': [0.0001, 0.001, 0.01, 0.1]})
```

```
In [71]:
# best parameters

grid_search.best_params_
```

```
Out[71]: {'C': 1, 'gamma': 0.1}
```

```
In [72]:
# best score

grid_search.best_score_
```

```
Out[72]: 0.8665843766531477
```

```
In [73]:
svc = SVC(C = 1, gamma = 0.1, probability=True)
svc.fit(X_train, y_train)
```

```
Out[73]: SVC(C=1, gamma=0.1, probability=True)
```

```
In [74]:
# model predictions

y_pred = svc.predict(X_test)
```

```
In [75]:
# accuracy score

print(accuracy_score(y_train, svc.predict(X_train)))

svc_acc = accuracy_score(y_test, svc.predict(X_test))
print(svc_acc)
```

```
0.8947368421052632
0.8421052631578947
```

```
In [76]:
# confusion matrix

print(confusion_matrix(y_test, y_pred))
```

```
[[134  13]
 [ 23  58]]
```

```
In [77]:
# classification report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       147
           1       0.82      0.72      0.76        81

    accuracy                           0.84       228
   macro avg       0.84      0.81      0.82       228
weighted avg       0.84      0.84      0.84       228
```

Heart Disease Prediction

In [1]:

```python
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')

sns.set()
plt.style.use('ggplot')
%matplotlib inline
```

In [3]:

```python
#import dataset
heart_df = pd.read_csv('data/heart.csv')
heart_df.head(10)
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
5	58	0	0	100	248	0	0	122	0	1.0	1	0	2	1
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
7	55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
9	54	1	0	122	286	0	0	116	1	3.2	1	2	2	0

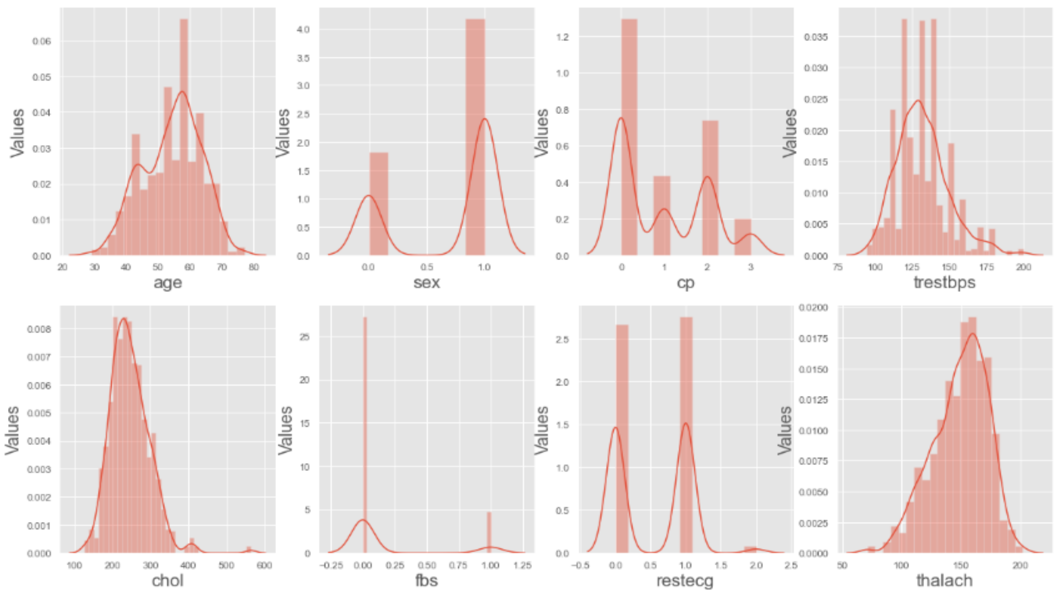
In [4]:

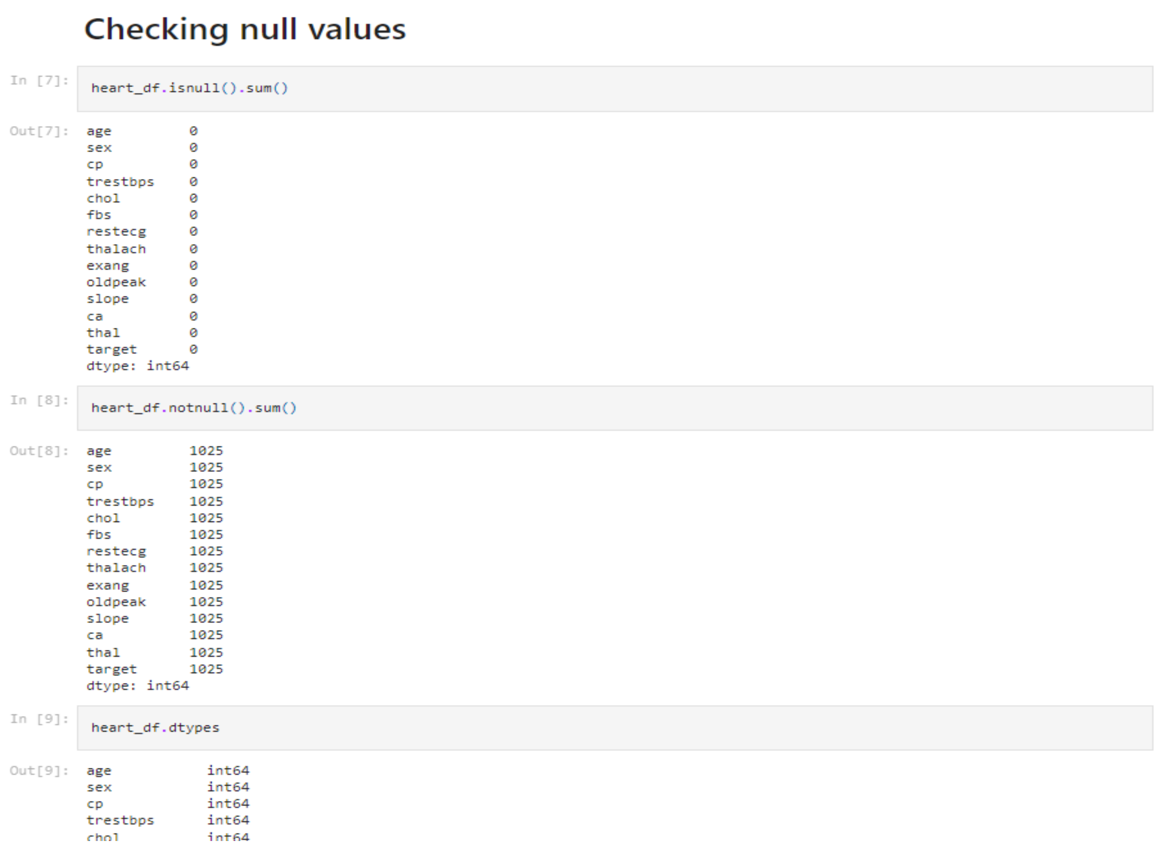
```python
# information about the dataset
heart_df.info()
```

Exploratory Data Analysis(EDA)

In [10]:

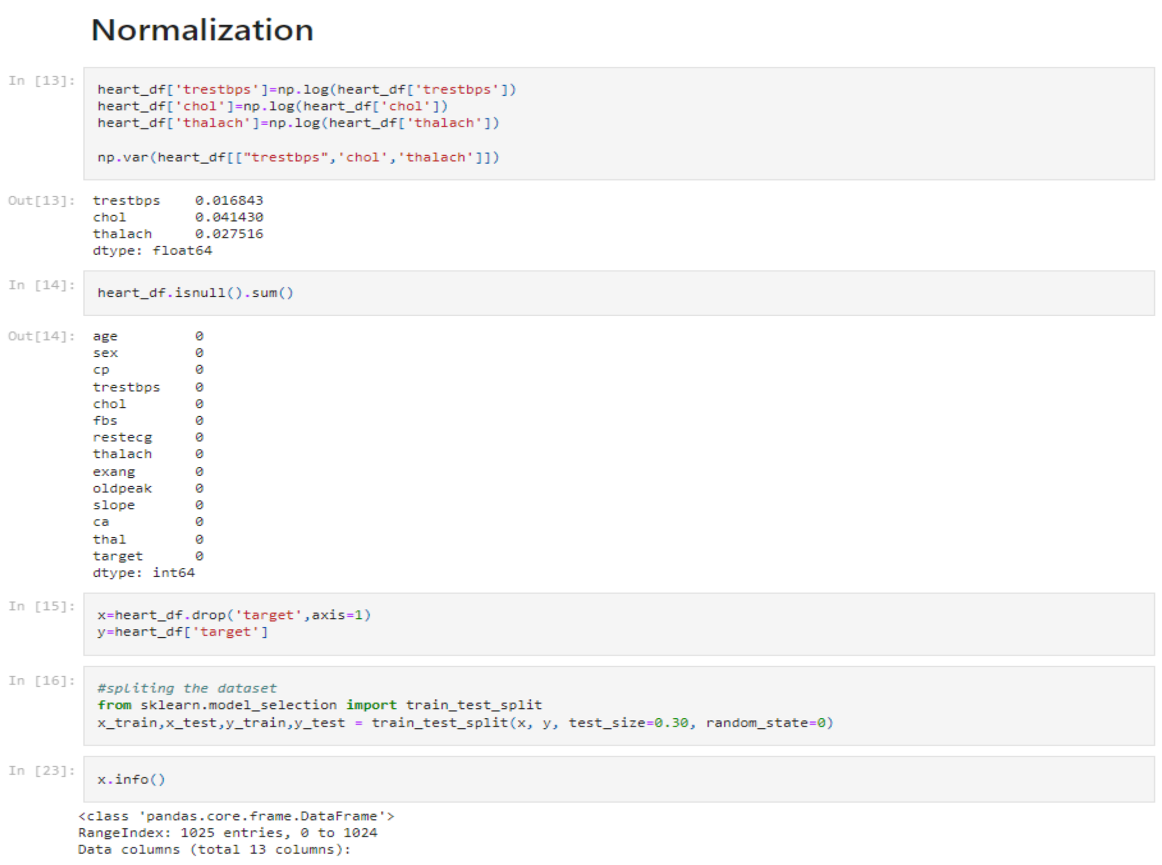
```python
#Plotting the distribution plot.
plt.figure(figsize=(20,25))
plotnumber=1

for column in heart_df:
    if plotnumber<14:
        ax=plt.subplot(4,4,plotnumber)
        sns.distplot(heart_df[column])
        plt.xlabel(column,fontsize=20)
        plt.ylabel('Values',fontsize=20)
    plotnumber+=1
plt.show()
```





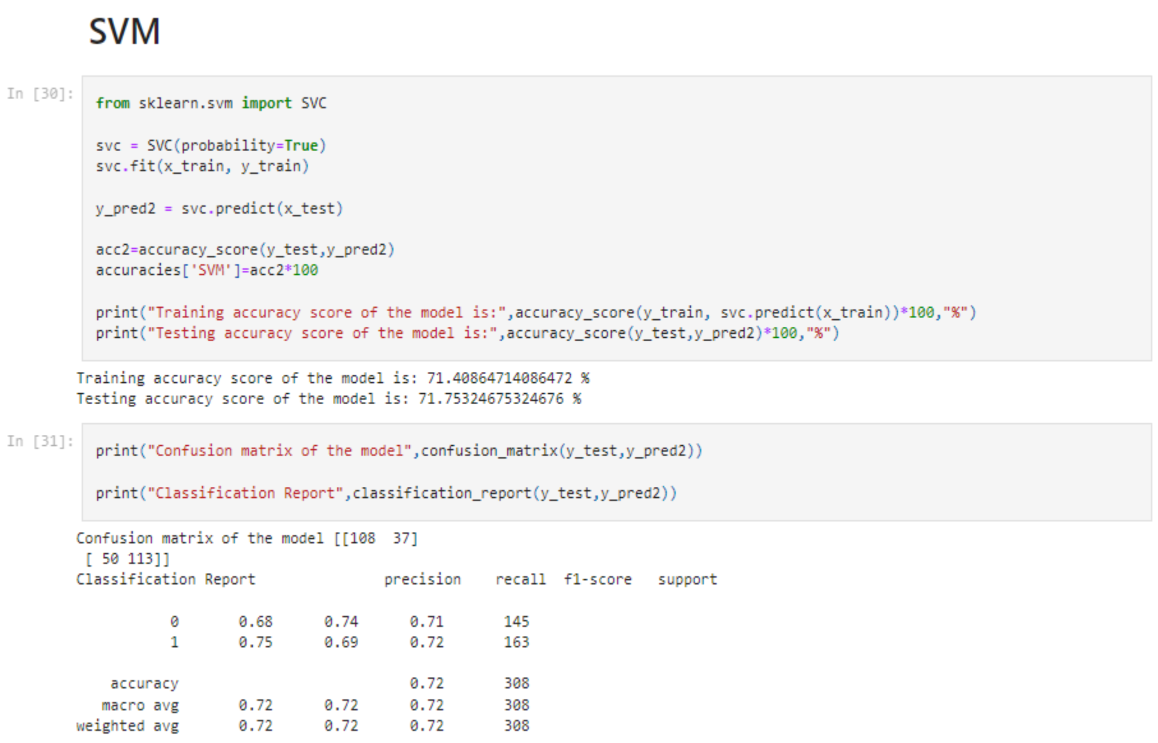
Checking null values

```
In [7]:  heart_df.isnull().sum()

Out[7]:  age         0
         sex         0
         cp          0
         trestbps    0
         chol        0
         fbs         0
         restecg     0
         thalach     0
         exang       0
         oldpeak     0
         slope       0
         ca          0
         thal        0
         target      0
         dtype: int64

In [8]:  heart_df.notnull().sum()

Out[8]:  age         1025
         sex         1025
         cp          1025
         trestbps    1025
         chol        1025
         fbs         1025
         restecg     1025
         thalach     1025
         exang       1025
         oldpeak     1025
         slope       1025
         ca          1025
         thal        1025
         target      1025
         dtype: int64

In [9]:  heart_df.dtypes

Out[9]:  age            int64
         sex            int64
         cp             int64
         trestbps       int64
         chol           int64
```



Normalization

```
In [13]:  heart_df['trestbps']=np.log(heart_df['trestbps'])
          heart_df['chol']=np.log(heart_df['chol'])
          heart_df['thalach']=np.log(heart_df['thalach'])

          np.var(heart_df[["trestbps",'chol','thalach']])

Out[13]:  trestbps    0.016843
          chol        0.041430
          thalach     0.027516
          dtype: float64

In [14]:  heart_df.isnull().sum()

Out[14]:  age         0
          sex         0
          cp          0
          trestbps    0
          chol        0
          fbs         0
          restecg     0
          thalach     0
          exang       0
          oldpeak     0
          slope       0
          ca          0
          thal        0
          target      0
          dtype: int64

In [15]:  x=heart_df.drop('target',axis=1)
          y=heart_df['target']

In [16]:  #spliting the dataset
          from sklearn.model_selection import train_test_split
          x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.30, random_state=0)

In [23]:  x.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 1025 entries, 0 to 1024
         Data columns (total 13 columns):
```



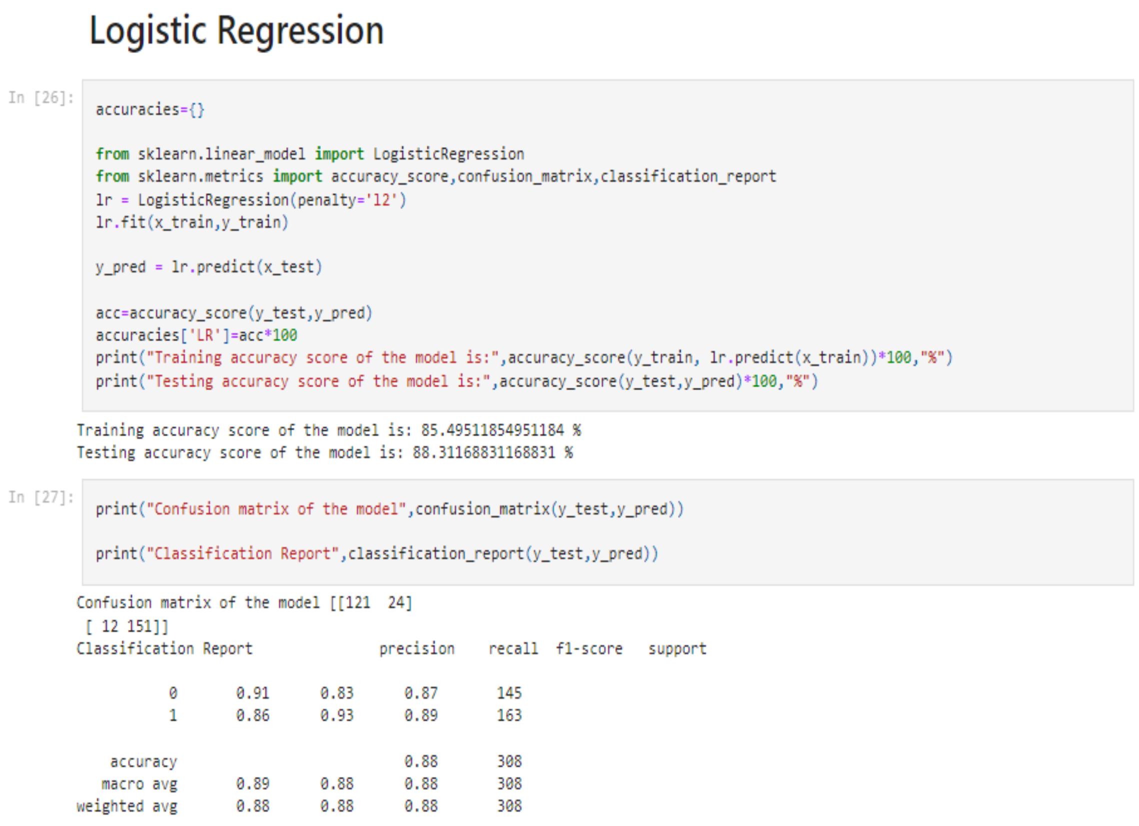
SVM

```
In [30]:
from sklearn.svm import SVC

svc = SVC(probability=True)
svc.fit(x_train, y_train)

y_pred2 = svc.predict(x_test)

acc2=accuracy_score(y_test,y_pred2)
accuracies['SVM']=acc2*100

print("Training accuracy score of the model is:",accuracy_score(y_train, svc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred2)*100,"%")
```

```
Training accuracy score of the model is: 71.40864714086472 %
Testing accuracy score of the model is: 71.75324675324676 %
```

```
In [31]:
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred2))

print("Classification Report",classification_report(y_test,y_pred2))
```

```
Confusion matrix of the model [[108  37]
 [ 50 113]]
Classification Report               precision    recall  f1-score   support

           0       0.68      0.74      0.71       145
           1       0.75      0.69      0.72       163

    accuracy                           0.72       308
   macro avg       0.72      0.72      0.72       308
weighted avg       0.72      0.72      0.72       308
```



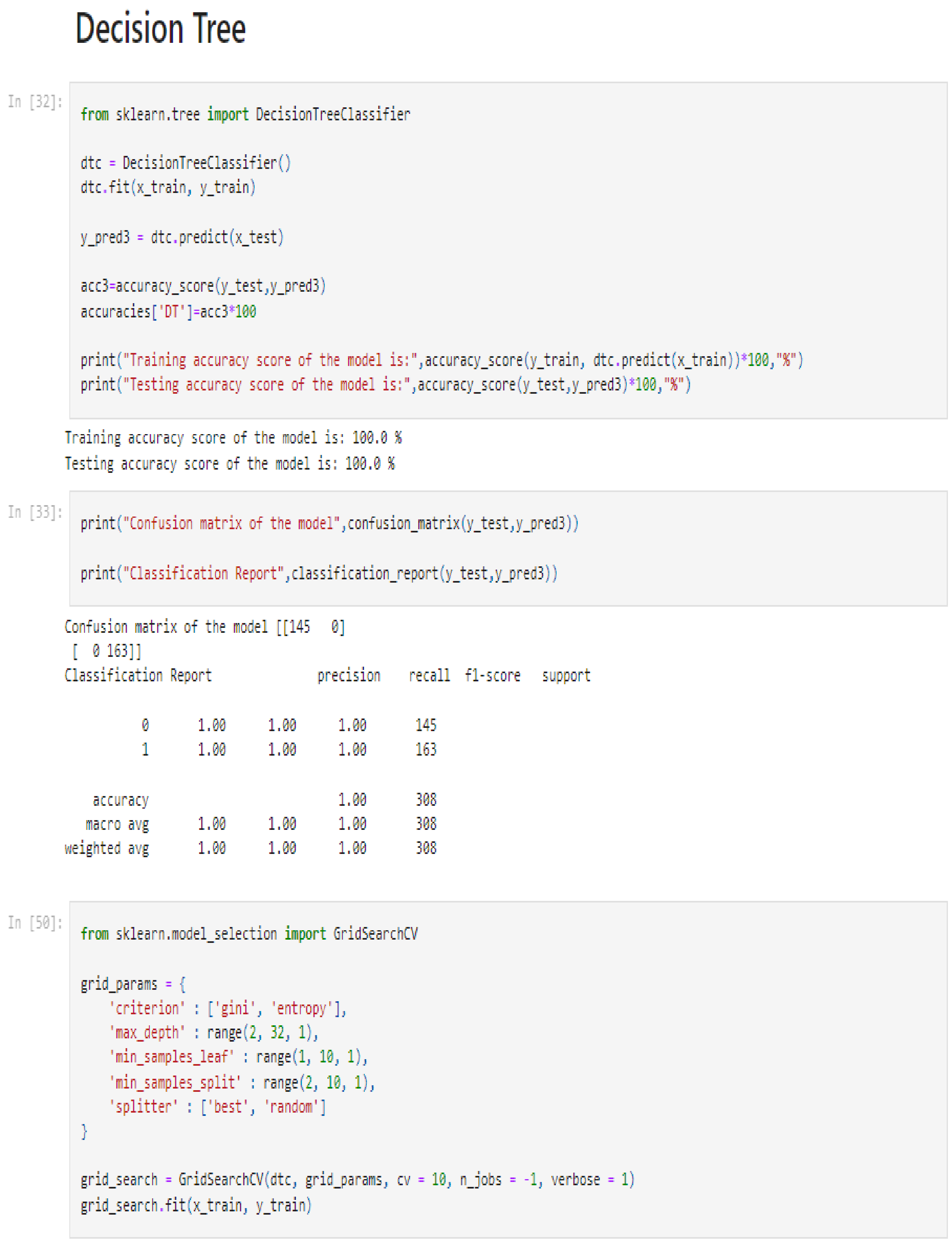
Logistic Regression

```
In [26]:
accuracies={}

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
lr = LogisticRegression(penalty='l2')
lr.fit(x_train,y_train)

y_pred = lr.predict(x_test)

acc=accuracy_score(y_test,y_pred)
accuracies['LR']=acc*100
print("Training accuracy score of the model is:",accuracy_score(y_train, lr.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred)*100,"%")
```

```
Training accuracy score of the model is: 85.49511854951184 %
Testing accuracy score of the model is: 88.31168831168831 %
```

```
In [27]:
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred))

print("Classification Report",classification_report(y_test,y_pred))
```

```
Confusion matrix of the model [[121  24]
 [ 12 151]]
Classification Report               precision    recall  f1-score   support

           0       0.91      0.83      0.87       145
           1       0.86      0.93      0.89       163

    accuracy                           0.88       308
   macro avg       0.89      0.88      0.88       308
weighted avg       0.88      0.88      0.88       308
```



Decision Tree

In [32]:
```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)

y_pred3 = dtc.predict(x_test)

acc3=accuracy_score(y_test,y_pred3)
accuracies['DT']=acc3*100

print("Training accuracy score of the model is:",accuracy_score(y_train, dtc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred3)*100,"%")
```

```
Training accuracy score of the model is: 100.0 %
Testing accuracy score of the model is: 100.0 %
```

In [33]:
```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred3))

print("Classification Report",classification_report(y_test,y_pred3))
```

```
Confusion matrix of the model [[145   0]
 [  0 163]]
Classification Report               precision    recall  f1-score   support

           0       1.00      1.00      1.00       145
           1       1.00      1.00      1.00       163

    accuracy                           1.00       308
   macro avg       1.00      1.00      1.00       308
weighted avg       1.00      1.00      1.00       308
```

In [50]:
```
from sklearn.model_selection import GridSearchCV

grid_params = {
    'criterion' : ['gini', 'entropy'],
    'max_depth' : range(2, 32, 1),
    'min_samples_leaf' : range(1, 10, 1),
    'min_samples_split' : range(2, 10, 1),
    'splitter' : ['best', 'random']
}

grid_search = GridSearchCV(dtc, grid_params, cv = 10, n_jobs = -1, verbose = 1)
grid_search.fit(x_train, y_train)
```

Random Forest

In [57]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(criterion = 'gini', max_depth = 7, max_features = 'sqrt', min_samples_leaf = 2, min_samples
rfc.fit(x_train, y_train)

y_pred5 = rfc.predict(x_test)

acc5=accuracy_score(y_test,y_pred5)
accuracies['RF']=acc5*100

print("Training accuracy score of the model is:",accuracy_score(y_train, rfc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred5)*100,"%")
```

```
Training accuracy score of the model is: 99.442119944212 %
Testing accuracy score of the model is: 98.7012987012987 %
```

In [58]:
```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred5))

print("Classification Report",classification_report(y_test,y_pred5))
```

```
Confusion matrix of the model [[141   4]
 [  0 163]]
Classification Report               precision    recall  f1-score   support

           0       1.00      0.97      0.99       145
           1       0.98      1.00      0.99       163

    accuracy                           0.99       308
   macro avg       0.99      0.99      0.99       308
weighted avg       0.99      0.99      0.99       308
```

Gradient Boosting

In [131…
```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier()

gbc = GradientBoostingClassifier(learning_rate = 0.05, loss = 'deviance', n_estimators = 180)
gbc.fit(x_train, y_train)

y_pred6 = gbc.predict(x_test)

acc6 = accuracy_score(y_test,y_pred6)
accuracies['GradientBoosting']=acc6*100

print("Training accuracy score of the model is:",accuracy_score(y_train, gbc.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred6)*100,"%")
```

```
Training accuracy score of the model is: 98.60529986052998 %
Testing accuracy score of the model is: 97.72727272727273 %
```

```
Fitting 10 folds for each of 8640 candidates, totalling 86400 fits
```

Out[50]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 32),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']},
             verbose=1)
```

In [51]:
```
grid_search.best_score_
```

Out[51]:
```
0.9860328638497652
```

In [52]:
```
grid_search.best_params_
```

Out[52]:
```
{'criterion': 'entropy',
 'max_depth': 12,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'splitter': 'random'}
```

In [53]:
```
dtc2 = DecisionTreeClassifier(criterion= 'entropy', max_depth= 12, min_samples_leaf= 1, min_samples_split= 2, splitter=
dtc2.fit(x_train, y_train)
```

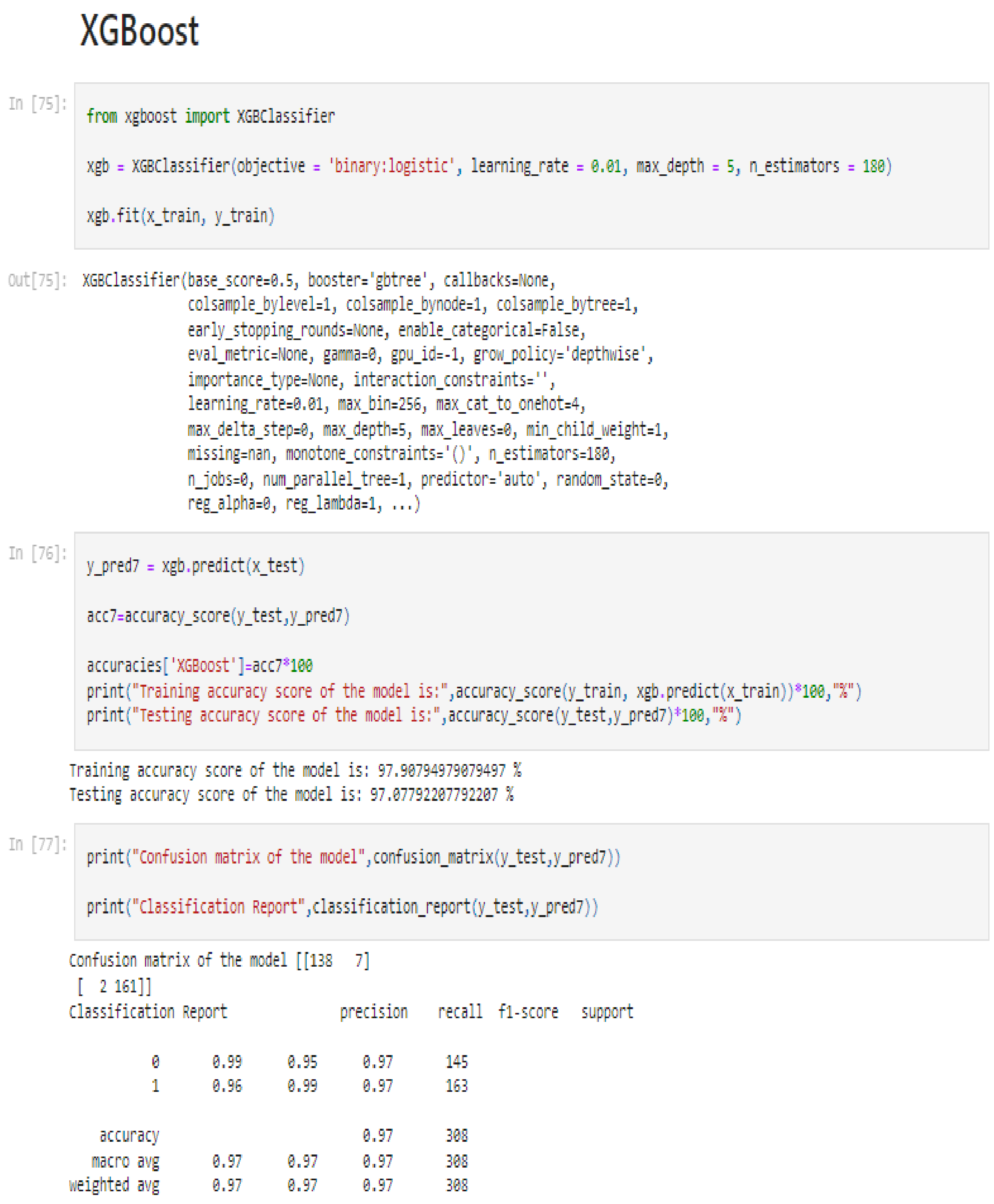
Out[53]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=12, splitter='random')
```

In [54]:
```
y_pred4 = dtc2.predict(x_test)
acc4=accuracy_score(y_test,y_pred4)
accuracies['DT2']=acc4*100

print("Training accuracy score of the model is:",accuracy_score(y_train, dtc2.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred4)*100,"%")
```

```
Training accuracy score of the model is: 99.30264993026499 %
Testing accuracy score of the model is: 97.72727272727273 %
```

In [55]:
```
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred4))

print("Classification Report",classification_report(y_test,y_pred4))
```

```
Confusion matrix of the model [[145   0]
 [  7 156]]
Classification Report               precision    recall  f1-score   support

           0       0.95      1.00      0.98       145
           1       1.00      0.96      0.98       163

    accuracy                           0.98       308
   macro avg       0.98      0.98      0.98       308
weighted avg       0.98      0.98      0.98       308
```



XGBoost

```
In [75]:
from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning_rate = 0.01, max_depth = 5, n_estimators = 180)

xgb.fit(x_train, y_train)
```

```
Out[75]: XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
                       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
                       early_stopping_rounds=None, enable_categorical=False,
                       eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
                       importance_type=None, interaction_constraints='',
                       learning_rate=0.01, max_bin=256, max_cat_to_onehot=4,
                       max_delta_step=0, max_depth=5, max_leaves=0, min_child_weight=1,
                       missing=nan, monotone_constraints='()', n_estimators=180,
                       n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
                       reg_alpha=0, reg_lambda=1, ...)
```

```
In [76]:
y_pred7 = xgb.predict(x_test)

acc7=accuracy_score(y_test,y_pred7)

accuracies['XGBoost']=acc7*100
print("Training accuracy score of the model is:",accuracy_score(y_train, xgb.predict(x_train))*100,"%")
print("Testing accuracy score of the model is:",accuracy_score(y_test,y_pred7)*100,"%")
```

```
Training accuracy score of the model is: 97.90794979079497 %
Testing accuracy score of the model is: 97.07792207792207 %
```

```
In [77]:
print("Confusion matrix of the model",confusion_matrix(y_test,y_pred7))

print("Classification Report",classification_report(y_test,y_pred7))
```

```
Confusion matrix of the model [[138   7]
 [  2 161]]
Classification Report               precision    recall  f1-score   support

           0       0.99      0.95      0.97       145
           1       0.96      0.99      0.97       163

    accuracy                           0.97       308
   macro avg       0.97      0.97      0.97       308
weighted avg       0.97      0.97      0.97       308
```

CHAPTER 4

PERFORMANCE EVALUATION GRAPHS

4.1 Histogram and Density Graphs of all Diabetes Variables

Graphical representation that provides a visual summary of the distribution of data. In the context of your disease prediction system for diabetes, each histogram represents the distribution of a specific variable (glucose, age, pregnancy, blood pressure, skin thickness, BMI, diabetes pedigree) within the dataset.

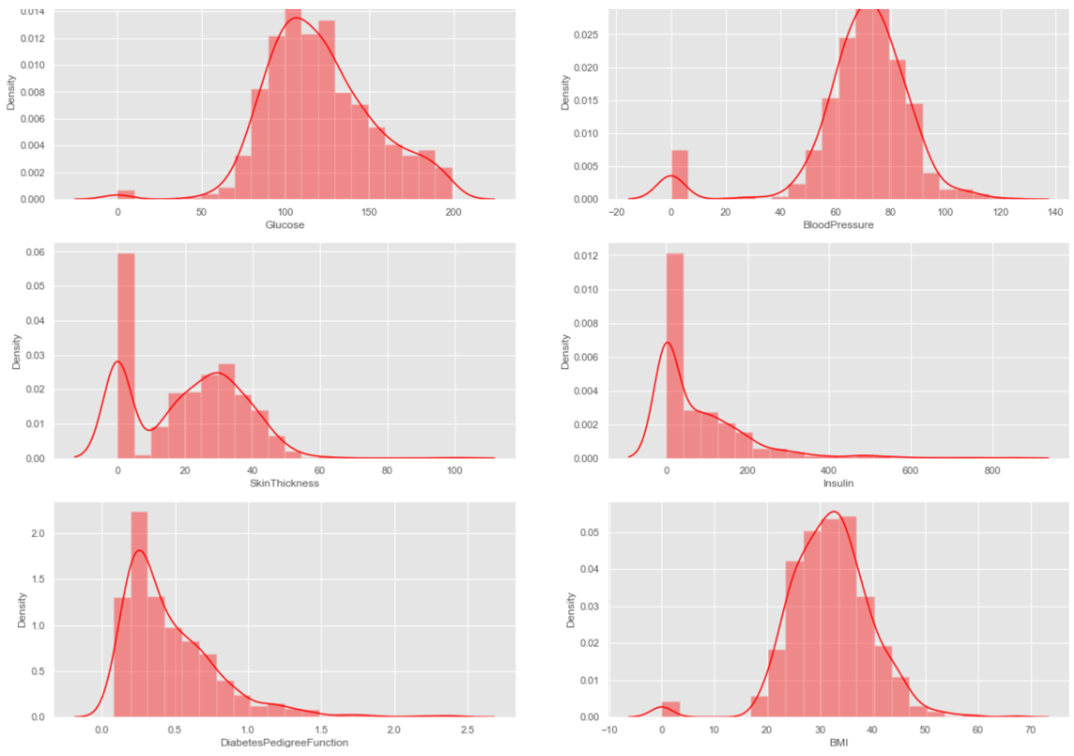


Fig. 4.1: Diabetes Histogram

4.2 Distribution Plot for Heart Disease Values

The plot displays the distribution of values for each column in the dataset. This exploratory data analysis (EDA) technique helps in understanding the underlying distribution of each feature in the dataset, which can be crucial for further analysis and modeling, particularly in the context of heart disease prediction.

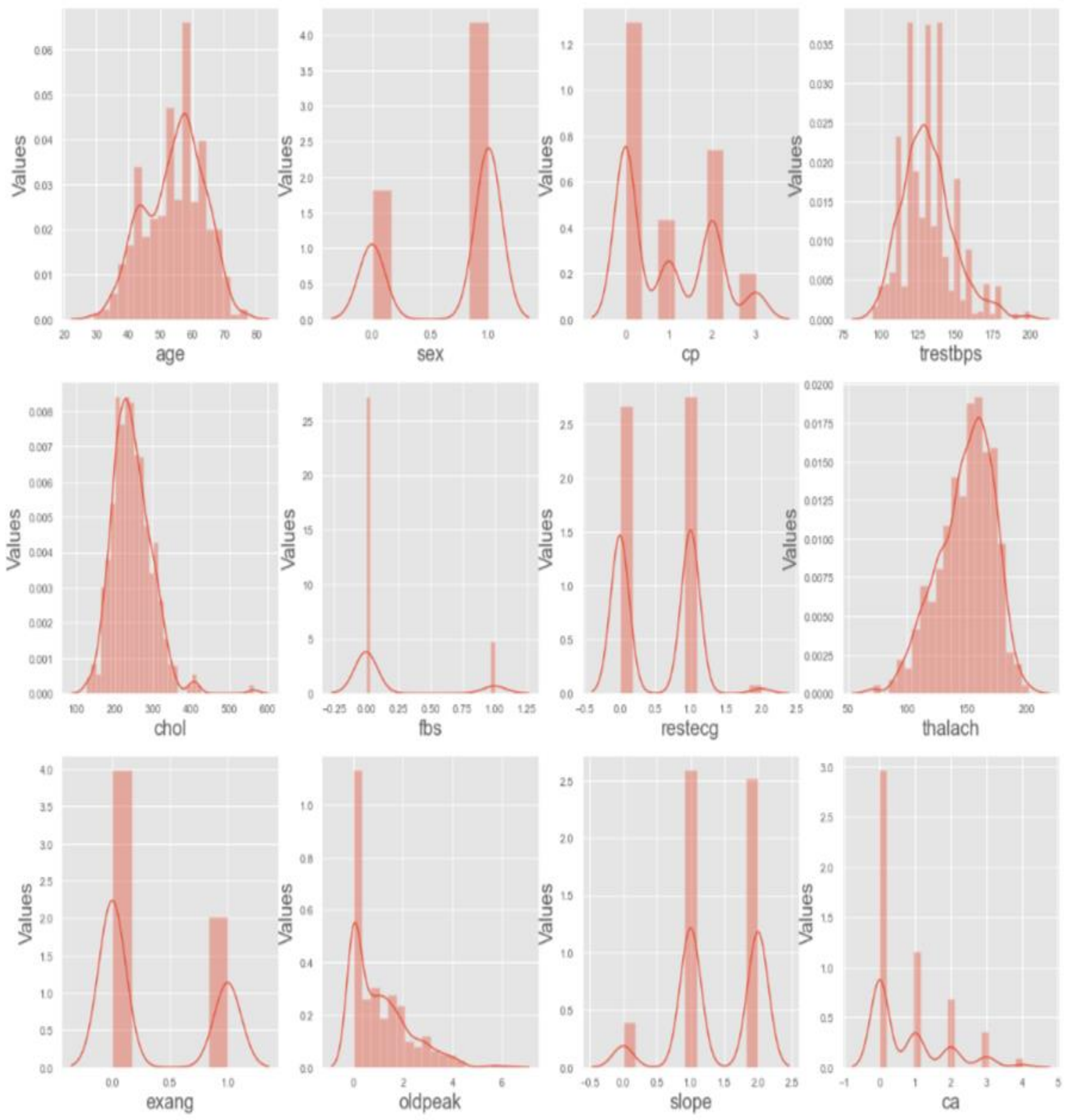


Fig. 4.2: Heart Disease Histogram

4.3 Correlation Matrix

Developing a diabetes prediction system, the correlation matrix is pivotal. It reveals relationships between features, aiding in key predictor identification and streamlined model development. This matrix guides feature selection, highlighting influential variables, and helps detect multicollinearity issues for stable model coefficients. The insights enhance overall model performance, making the disease prediction system more reliable and effective.

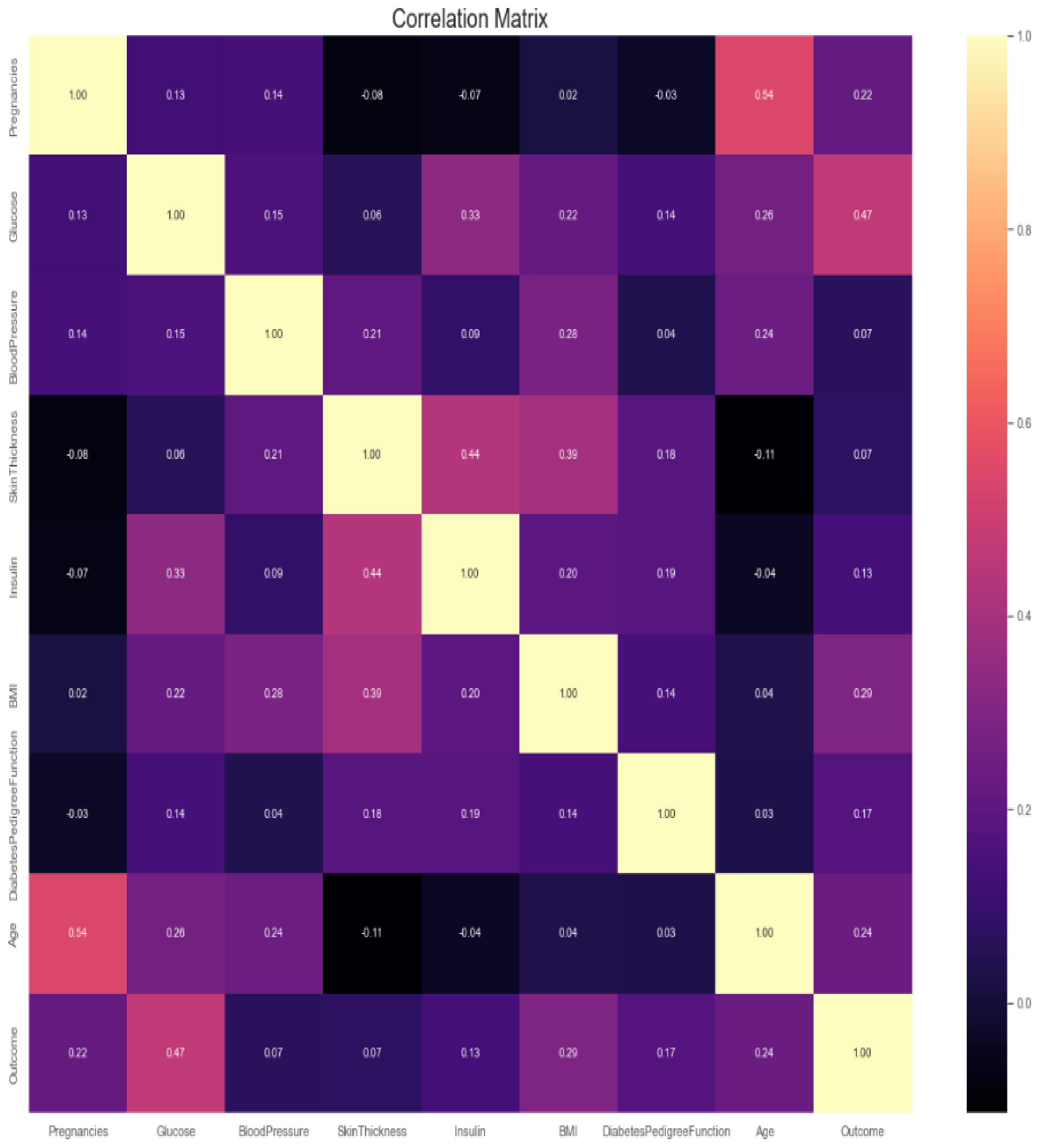


Fig. 4.3: For Diabetes

This matrix guides feature selection, highlighting influential variables, and helps detect multicollinearity issues for stable model coefficients. The insights enhance overall model performance, making the disease prediction system more reliable and effective.

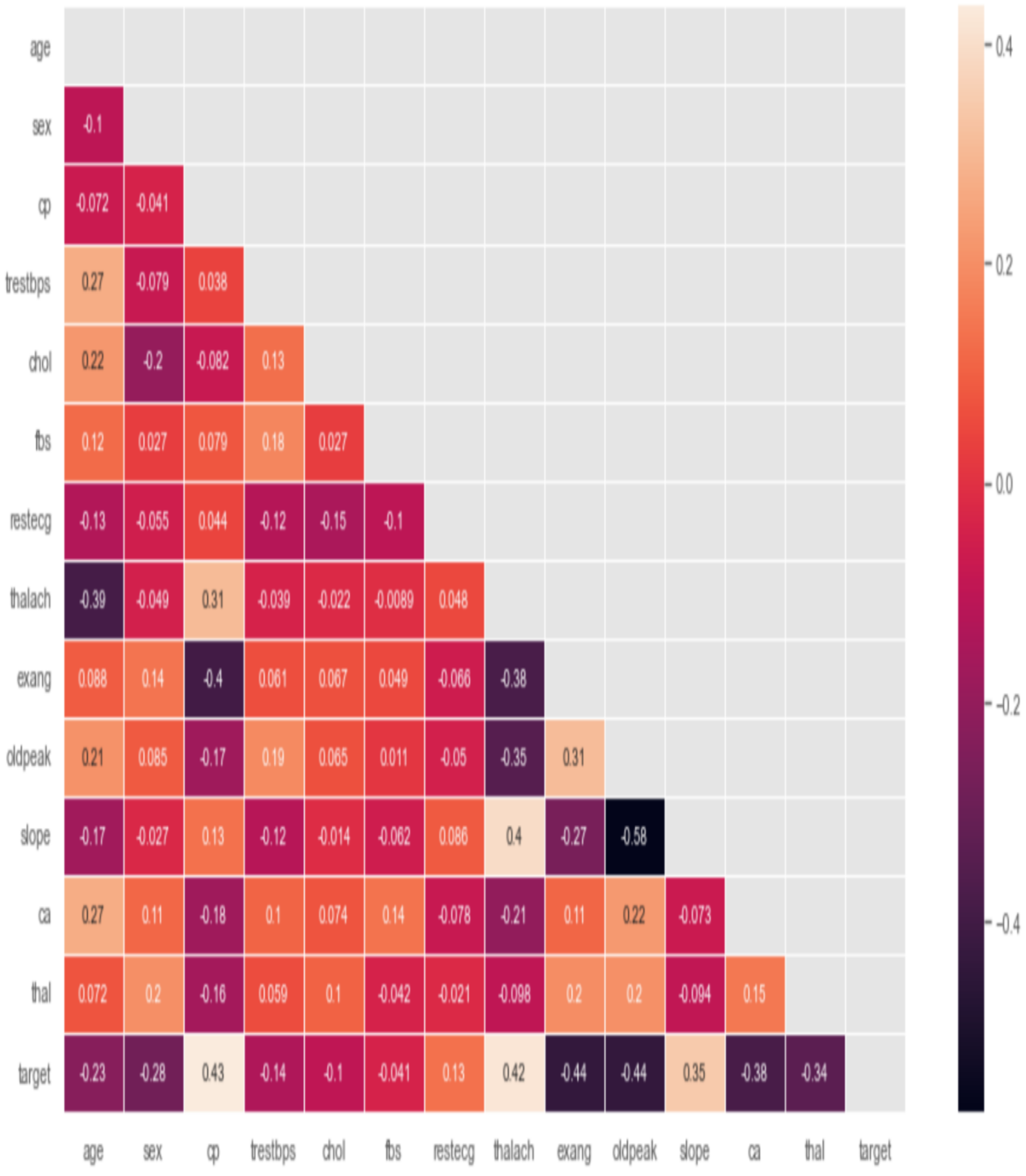
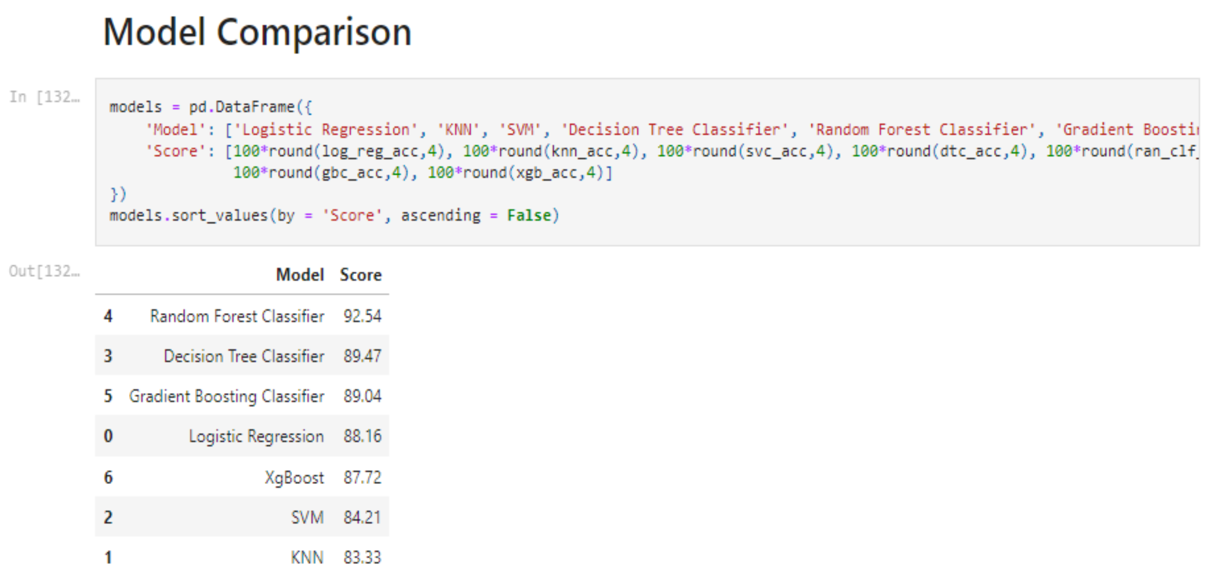


Fig. 4.4: For Heart Disease

4.4 Model Comparison

Model Comparison graph succinctly illustrates the performance of different algorithms in the disease prediction system. It streamlines algorithm selection by presenting key metrics, such as accuracy and precision, for each model. This visual aid informs decision-making, guiding optimization efforts and facilitating a concise yet comprehensive evaluation of algorithm effectiveness in the project.



Model Comparison

In [132…

```
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'KNN', 'SVM', 'Decision Tree Classifier', 'Random Forest Classifier', 'Gradient Boosti
    'Score': [100*round(log_reg_acc,4), 100*round(knn_acc,4), 100*round(svc_acc,4), 100*round(dtc_acc,4), 100*round(ran_clf_
              100*round(gbc_acc,4), 100*round(xgb_acc,4)]
})
models.sort_values(by = 'Score', ascending = False)
```

Out[132…

	Model	Score
4	Random Forest Classifier	92.54
3	Decision Tree Classifier	89.47
5	Gradient Boosting Classifier	89.04
0	Logistic Regression	88.16
6	XgBoost	87.72
2	SVM	84.21
1	KNN	83.33

Fig. 4.5: For Diabetes

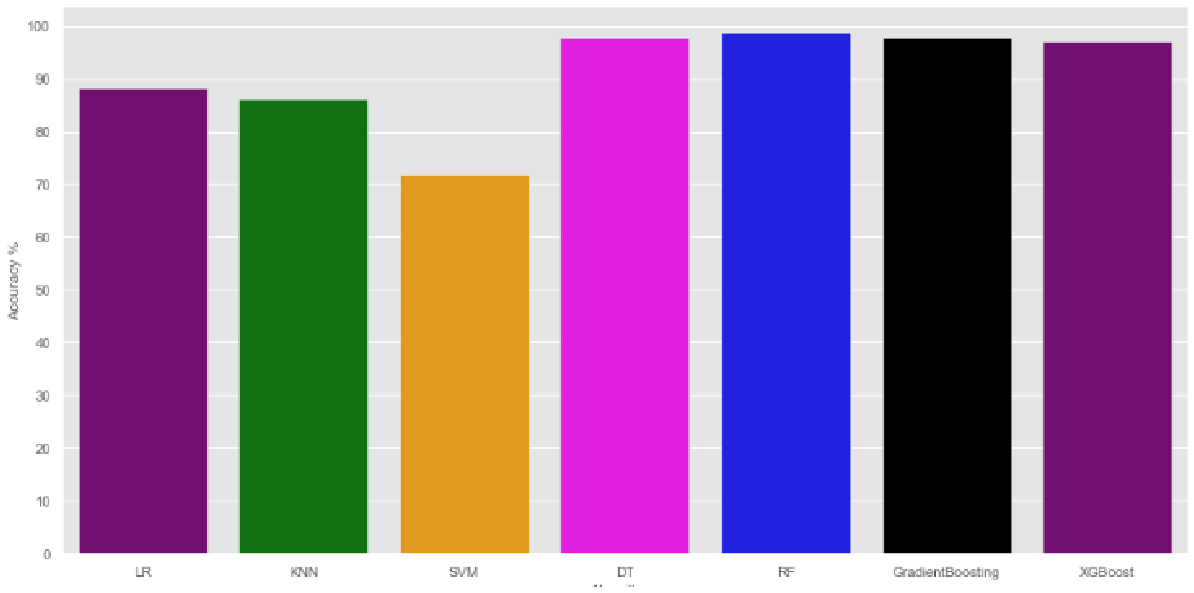


Fig. 4.6: For Heart Disease

4.5 ROC Curve

The Specificity-Sensitivity graph, or ROC curve, is crucial for evaluating a disease prediction system. It illustrates the trade-off between correctly identifying positives (sensitivity) and avoiding false positives (specificity) at different thresholds. A curve close to the upper left corner indicates high performance. The area under the curve (AUC-ROC) summarizes overall model accuracy, aiding in threshold selection for balance and enhancing the reliability of the disease prediction system.

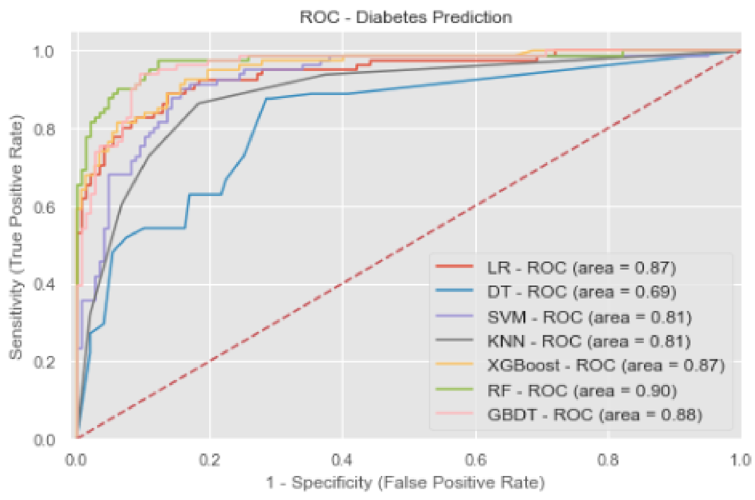


Fig. 4.7: For Diabetes

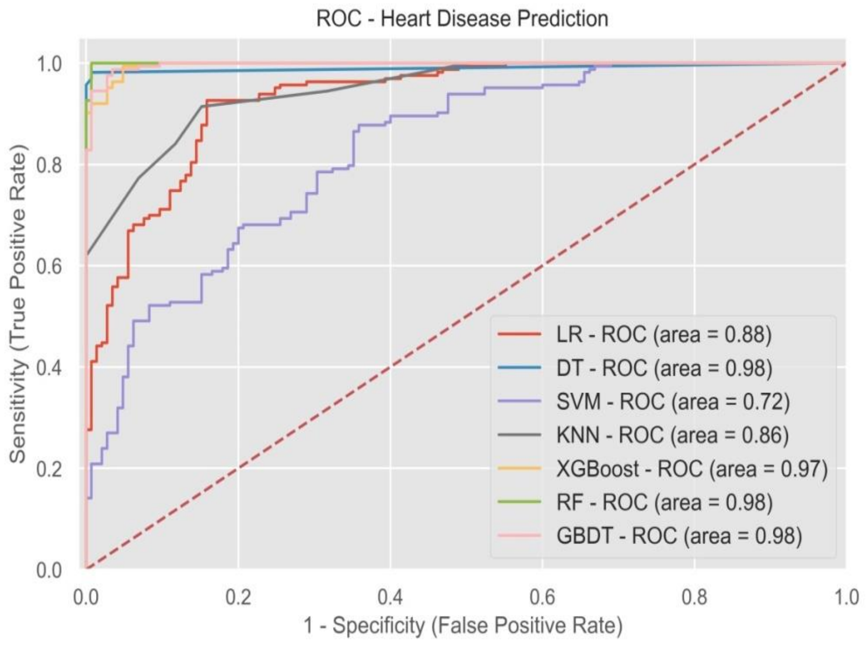
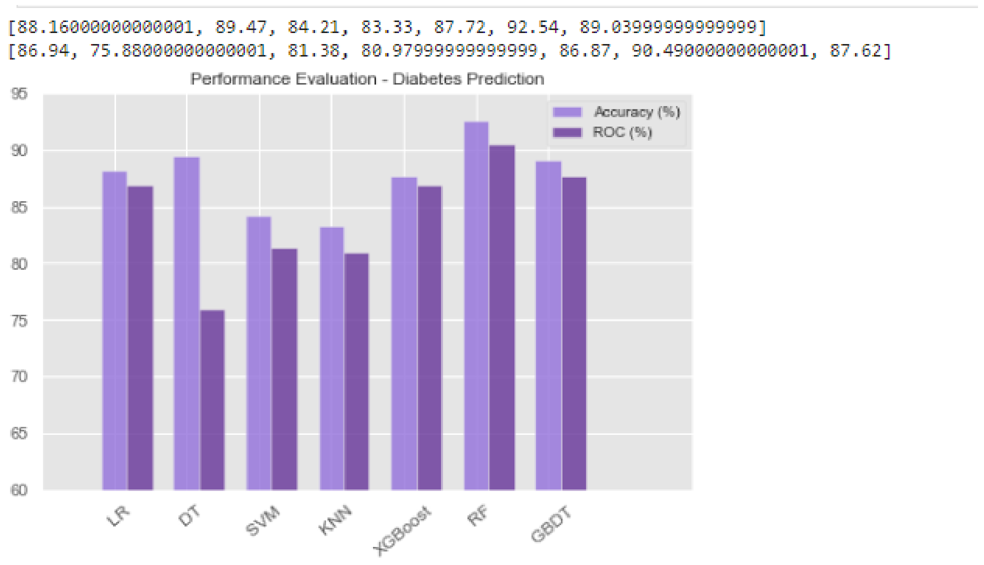


Fig. 4.8: For Heart Disease

4.6 Performance Evaluation

The Performance Evaluation section compares different algorithms in our disease prediction system, utilizing key metrics like accuracy and precision. This critical assessment guides algorithm selection and optimization, offering a clear view of strengths and weaknesses. Integrating this evaluation enhances transparency, aiding in the development of a robust and accurate disease prediction system.



```
[88.16000000000001, 89.47, 84.21, 83.33, 87.72, 92.54, 89.03999999999999]
[86.94, 75.88000000000001, 81.38, 80.97999999999999, 86.87, 90.49000000000001, 87.62]
```

Fig. 4.9: For Diabetes

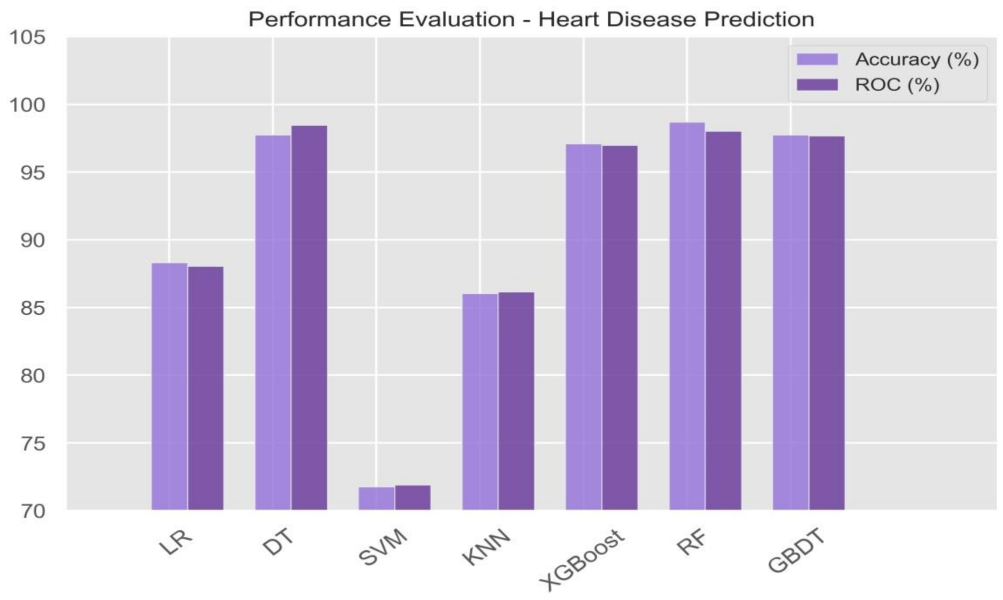


Fig. 4.10: For Heart Disease

CHAPTER 5

RESULT AND DISCUSSIONS

5.1 Results

Our in-depth analysis revealed the following results for the machine learning algorithms in diagnosing diabetes and heart diseases:

- **Logistic Regression**: Achieved an accuracy of 88.16% for diabetes and 88.31% for heart disease, showing solid performance, especially for heart disease.
- **K-Nearest Neighbors (KNN)**: Attained an accuracy of 83.33% for diabetes and 86.03% for heart disease, demonstrating promise but with room for improvement.
- **Support Vector Machine (SVM)**: Showed an accuracy of 84.21% for diabetes and 71.75% for heart disease, performing reasonably well for diabetes.
- **Decision Tree**: Outperformed others with 89.47% accuracy for diabetes and an impressive 97.72% for heart disease, showing significant potential for both disease categories.
- **Random Forest Classifier**: Delivered robust results with 92.5% accuracy for diabetes and an outstanding 98.70% for heart disease, highlighting high accuracy and reliability in diagnosing both diseases.
- **Gradient Boosting Classifier**: Achieved an accuracy of 89.04% for diabetes and 97.72% for heart disease, maintaining consistent performance.
- **XGBoost Classifier**: Obtained an accuracy of 87.72% for diabetes and 97.07% for heart disease, showing promise for heart disease diagnosis.

5.2 Discussions

These outcomes are specific to the dataset utilized in this project. The varying performance underscores the importance of selecting the most suitable algorithm for a particular medical condition. Comparative graphs have been plotted to enhance the visual understanding of algorithm strengths and weaknesses.

Developing a diabetes prediction system, the correlation matrix is pivotal. It reveals relationships between features, aiding in key predictor identification and streamlined model development. This matrix guides feature selection, highlighting influential variables, and helps detect multicollinearity issues for stable model coefficients. The insights enhance overall model performance, making the disease prediction system more reliable and effective.

As we expand our project in the future, incorporating additional diseases into our analysis will broaden the scope and relevance of our findings. This project serves as a foundation for more accurate and efficient disease diagnosis, ultimately benefiting patient outcomes and healthcare efficiency.

Graphical representation that provides a visual summary of the distribution of data. In the context of your disease prediction system for diabetes, each histogram represents the distribution of a specific variable (glucose, age, pregnancy, blood pressure, skin thickness, BMI, diabetes pedigree) within the dataset.

These findings hold tremendous promise for improving the speed and accuracy of disease diagnosis, ultimately enhancing patient outcomes and transforming the field of healthcare. This project sets the stage for a future where data-driven algorithms play a pivotal role in revolutionizing medical diagnostics.

Our models leverage innovative approaches such as dynamic feature engineering, explainable deep learning, and federated learning for privacy preservation to address existing challenges in healthcare prediction.

Expanding our project in the future, incorporating additional diseases into our analysis will broaden the scope and relevance of our findings. This project serves as a foundation for more accurate and efficient disease diagnosis, ultimately benefiting patient outcomes and healthcare.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

6.1 Future Scope

For future work, we propose the following enhancements:

Including other diseases: We are going to include other diseases also for the analysis in future.

Dataset Expansion: Expanding the dataset to include a larger and more diverse sample to enhance the algorithm's generalization capabilities.

Algorithm Fine-Tuning: Fine-tuning the selected algorithm to further improve its accuracy and performance.

Real-Time Integration: Integrating real-time data collection and automated disease diagnosis, potentially reducing the time required for diagnoses.

Longitudinal Data Analysis: Incorporate longitudinal patient data to capture disease progression over time and enhance predictive accuracy.

Integration with Wearable Devices: Explore the integration of data from wearable devices and remote monitoring technologies to capture real-time physiological data and improve disease prediction.

Personalized Treatment Recommendations: Develop personalized treatment recommendation systems based on predictive model outputs and patient-specific characteristics to support precision medicine approaches.

Integration with Telemedicine Platforms: Integrate predictive models with telemedicine platforms to enable remote risk assessment and virtual consultations for patients at high risk of diabetes and heart disease.

Validation in Real-World Clinical Settings: Conduct validation studies in real-world clinical settings to assess the generalizability and robustness of predictive models across diverse patient populations and healthcare environments.

Collaboration with Healthcare Institutions: Collaborate with healthcare institutions to deploy predictive models in clinical practice and evaluate their impact on patient outcomes, healthcare delivery, and resource utilization.

These enhancements will further advance the effectiveness of disease diagnosis using machine learning, ultimately benefiting patient outcomes and healthcare efficiency.

6.2 Conclusion

In conclusion, our project was dedicated to the pursuit of more accurate and efficient disease diagnosis for diabetes and heart diseases. Through our extensive analysis and evaluation of machine learning algorithms, we have successfully identified the best-performing algorithms.

These findings hold tremendous promise for improving the speed and accuracy of disease diagnosis, ultimately enhancing patient outcomes and transforming the field of healthcare. This project sets the stage for a future where data-driven algorithms play a pivotal role in revolutionizing medical diagnostics.

Our project has successfully conducted a comprehensive comparative analysis of machine learning algorithms for predicting diabetes and heart disease. Through rigorous exploratory data analysis, advanced preprocessing techniques, and feature engineering, we have developed accurate and interpretable predictive models that demonstrate promising performance in disease prediction tasks.

Our models leverage innovative approaches such as dynamic feature engineering, explainable deep learning, and federated learning for privacy preservation to address existing challenges in healthcare prediction.

The results of our comparative analysis reveal the strengths and limitations of different machine learning algorithms in predicting diabetes and heart disease. We have identified the most effective algorithms and methodologies for disease prediction tasks, providing valuable insights for healthcare practitioners and stakeholders.

The integration of real-time model interpretation and feedback mechanisms enables transparent and actionable insights, facilitating informed decision-making in clinical practice.

REFERENCES

[1] Culler SD, Parchman ML, Przybylski M., “Factors related to potentially preventable hospitalizations among the elderly”. Med Care. 1998.

[2] Uddin MS, Hossain L., “Social networks enabled coordination model for cost Management of Patient Hospital Admissions”. J Healthc Qual. 2011.

[3] Lee PP, et al., “Cost of patients with primary open-angle glaucoma: a retrospective study of commercial insurance claims data”. Ophthalmology. 2007.

[4] Davis DA, Chawla NV, Christakis NA, Barabási A-L., “Time to CARE: a collaborative engine for practical disease prediction”. Data Min Knowl Disc. 2010.

[5] McCormick T, Rudin C, Madigan D., “A hierarchical model for association rule mining of sequential events: an approach to automated medical symptom prediction”; 2011.

[6] Yiannakoulias N, Schopflocher D, Svenson L., “Using administrative data to understand the geography of case ascertainment”. Chron Dis Can. 2009.

[7] Fisher ES, Malenka DJ, Wennberg JE, Roos NP., “Technology assessment using insurance claims: example of prostatectomy”. Int J Technol Assess Health Care. 1990.

[8] Farran B, Channanath AM, Behbehani K, Thanaraj TA., “Predictive models to assess risk of type 2 diabetes, hypertension and comorbidity: machine- learning algorithms and validation using national health data from Kuwait-a cohort study”. BMJ Open. 2013.

[9] Ahmad LG, Eshlaghy A, Poorebrahimi A, Ebrahimi M, Razavi A., “Using three machine learning techniques for predicting breast cancer recurrence”. J Health Med Inform. 2013.

[10] Moher D, Liberati A, Tetzlaff J, Altman DG., “Preferred reporting items for reviews and meta-analyses: the PRISMA statement”, Ann Intern Med. 2009.

[11] Demšar J., “Statistical comparisons of classifiers over multiple data sets”. J Mach Learn Res. 2006. Lundin M, Lundin J, Burke H, Toikkanen S, Pylkkänen L, Joensuu H., “Artificial neural networks applied to survival prediction in breast cancer”. Oncology. 1999.

[12] Delen D, Walker G, Kadam A., “Predicting breast cancer survivability: a comparison of three data mining methods”. Artif Intell Med. 2005.

[13] Chen M, Hao Y, Hwang K, Wang L, Wang L., “Disease prediction by machine learning over big data from healthcare communities”. IEEE Access. 2017.

[14] Cai L, Wu H, Li D, Zhou K, Zou F., “Type 2 diabetes biomarkers of human gut microbiota selected via iterative sure independent screening method”. PLoS One. 2015.

[15] Malik S, Khadgawat R, Anand S, Gupta S., “Non-invasive detection of fasting blood glucose level via electrochemical measurement of saliva”. SpringerPlus. 2016.

[16] Mani S, Chen Y, Elasy T, Clayton W, Denny J., “Type 2 diabetes risk forecasting from EMR data using machine learning. In: AMIA annual symposium proceedings”, p. 606. American Medical Informatics Association, 2012.

[17] Tapak L, Mahjub H, Hamidi O, Poorolajal J., “Real-data comparison of data mining methods in prediction of diabetes in Iran”. Healthc Inform Res. 2013.

[18] Sisodia D, Sisodia DS., “Prediction of diabetes using classification algorithms”. Procedia Comput Sci. 2018.

[19] Yang J, Yao D, Zhan X, Zhan X., “Predicting disease risks using feature selection based on random forest and support vector machine. In: International Symposium on Bioinformatics Research and Applications”, p. 1–11. Springer, 2014.

[20] Juhola M, Joutsijoki H, Penttinen K, Aalto-Setälä K., “Detection of genetic cardiac diseases by Ca 2+ transient profiles using machine learning methods”. Sci Rep. 2018.

[21] Long NC, Meesad P, Unger H., “A highly accurate firefly based algorithm for heart disease prediction”, Expert Syst Appl. 2015.

[22] Taslimitehrani V, Dong G, Pereira NL, Panahiazar M, Pathak J., “Developing EHR-driven heart failure risk prediction models using CPXR (log) with the probabilistic loss function”, J Biomed Inform. 2016.

[23] Anbarasi M, Anupriya E, Iyengar N., “Enhanced prediction of heart disease with feature subset selection using genetic algorithm”. Int J Eng Sci Technol. 2010.

[24] Bhatla N, Jyoti K., “An analysis of heart disease prediction using different data mining techniques. Int J Eng. 2012.

[25] Thenmozhi K, Deepika P., “Heart disease prediction using classification with different decision tree techniques”, Int J Eng Res Gen Sci. 2014.

[26] Tamilarasi R, Porkodi DR., “A study and analysis of disease prediction techniques in data mining for healthcare”. Int J Emerg Res Manag Technoly ISSN. 2015.

[27] Marikani T, Shyamala K., “Prediction of heart disease using supervised learning algorithms”. Int J Comput Appl. 2017.

[28] Lu P, et al., “Research on improved depth belief network-based prediction of cardiovascular diseases”. J Healthc Eng. 2018.

[29] Khateeb N, Usman M., “Efficient Heart Disease Prediction System using K-Nearest Neighbor Classification Technique”. In: Proceedings of the International Conference on Big Data and Internet of Thing; 2017.

[30] Patel SB, Yadav PK, Shukla DD., “Predict the diagnosis of heart disease patients using classification mining techniques”. IOSR J Agri Vet Sci (IOSR- JAVS). 2013.

[31] Venkatalakshmi B, Shivsankar M., “Heart disease diagnosis using predictive data mining”. Int J Innovative Res Sci Eng Technol. 2014.

[32] Ani R, Sasi G, Sankar UR, Deepa O., “Decision support system for diagnosis and prediction of chronic renal failure using random subspace classification”. In: Advances in Computing, Communications and Informatics (ICACCI), International Conference on; IEEE, 2016.

APPENDIX 1

This section contains details on the language, software and packages used in our project.

This project is developed in Python. It offers concise and readable code. Despite being highly complex with versatile workflows, the AI and ML algorithms, when written in Python, can help the developers create robust and reliable machine intelligent systems. The list of Python packages used in our project are:

Numpy: The Python library offers a range of features that enable users to work with multidimensional array objects, as well as derived objects (such as matrices and masked arrays). In addition to basic operations like mathematical and logical functions, it provides an array of routines for tasks like manipulating shapes, sorting, selecting, and performing input/output operations. The library also includes support for more advanced mathematical functions such as discrete Fourier transforms, basic linear algebra, and statistical analysis. Due to its extensive capabilities, this library is considered a crucial tool for scientific computing in the Python language.

Pandas: This Python package offers quick, adaptable, and expressive data structures that are specifically designed to simplify and streamline the process of working with labeled or relational data. Its primary objective is to serve as a foundational high-level component that facilitates the practical and effective analysis of real-world data using Python. Beyond this, the package has a broader aspiration of becoming the most potent and adaptable open-source tool for manipulating and analyzing data in any programming language.

Imgaug: This machine learning library is specifically designed for image augmentation in experimental settings. It boasts an impressive array of augmentation techniques and allows users to effortlessly combine and execute these techniques in a random order or on multiple CPU cores. Its straightforward but potent stochastic interface provides users with an easy means of manipulating images, key points/landmarks, bounding boxes, heatmaps, and segmentation maps.

OpenCV-python: OpenCV, short for Open Source Computer Vision Library, is a software library that offers open-source computer vision and machine learning tools. The library was developed with the aim of creating a universal foundation for computer vision applications, and promoting the use of machine perception in commercial products. With a collection of over 2500 optimized algorithms, OpenCV features a wide variety of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be employed for detecting and recognizing faces, object identification, human action classification in videos, camera movement tracking, object tracking, 3D object model extraction, 3D point cloud generation from stereo cameras, image stitching for creating high-resolution images of complete scenes, image database search for similar images, red-eye removal from flash photography, eye movement tracking, scenery recognition, and establishing markers for overlaying augmented reality.

Keras: Keras is a freely available software library that offers a Python-based interface for artificial neural networks. The primary function of Keras is to serve as an interface for the Tensor Flow library. In versions prior to 2.3, Keras supported numerous backend, such as Tensor Flow, Microsoft Cognitive Toolkit, Theano, and PlaidM.

Tensorflow: TensorFlow is an open-source software library that facilitates machine learning and artificial intelligence applications without any cost. Although it can be used for a diverse range of tasks, its principal focus is on the training and inference of deep neural networks. TensorFlow was initially developed by the Google Brain team to support internal research and production. It can be employed in numerous programming languages, with Python being the most prominent, in addition to JavaScript, C++, and Java. TensorFlow comprises a broad, adaptable ecosystem of tools, libraries, and community resources that enable researchers to advance the state-of-the-art in machine learning and enable developers to easily construct and deploy machine learning powered applications

Scikit-learn: This machine learning library is freely available and intended for use with the Python programming language. It includes several algorithms for classification, regression, and clustering tasks, such as random forests, support-vector machines, gradient boosting, k-means, and DBSCAN. The library is intended to integrate seamlessly with the numerical and scientific Python libraries, such as SciPy and Numpy.

SciPy: The SciPy library is an open-source Python library utilized for technical and scientific computing purposes, available free of cost. It incorporates numerous modules for tasks such as optimization, linear algebra, integration, interpolation, special functions, and signal and image processing. The core data structure employed by SciPy is a multidimensional array that is provided by the Numpy module.

Tkinter: Tkinter is the go-to standard GUI library for Python programming language. Tkinter combination with Python facilitates an effortless and prompt creation of Graphical User Interface (GUI) applications. Tkinter provides a robust object-oriented interface to the Tk GUI toolkit. As Tkinter calls are translated into Tcl commands, which are then processed by the embedded interpreter, it becomes possible to use both Python and Tcl in a single application.

Jupyter Notebook: J Notebook is a web-based open-source tool that allows users to create and share documents containing live code, equations, text, and visuals. It can be utilized for various tasks including data visualization, statistical modeling, data transformation, and machine learning. Jupyter Notebooks evolved from a predecessor project that also had a notebook. It is named after the primary programming languages it supports, Julia, Python, and R, but it now supports more than 100 additional kernels.

APPENDIX 2

Exploratory Data Analysis

```
# Importing the packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale, StandardScaler
from sklearn.model_selection import train_test_split,
GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score,
mean_squared_error, r2_score, roc_auc_score, roc_curve,
classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import KFold
import warnings
warnings.simplefilter(action = "ignore")
sns.set()
plt.style.use('ggplot')
%matplotlib inline

# Plot histogram and density graphs of all variables
fig, ax = plt.subplots(4,2, figsize=(20,20))
sns.distplot(df.Age, bins = 20, ax=ax[0,0], color="red")
sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1],
color="red")
sns.distplot(df.Glucose, bins = 20, ax=ax[1,0], color="red")
sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1],
color="red")
sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0],
color="red")
sns.distplot(df.Insulin, bins = 20, ax=ax[2,1], color="red")
sns.distplot(df.DiabetesPedigreeFunction, bins = 20,
ax=ax[3,0], color="red")
sns.distplot(df.BMI, bins = 20, ax=ax[3,1], color="red")

# Visualize the distribution of the outcome variable in the
data -> 0 - Healthy, 1 - Diabetic
f,ax=plt.subplots(1,2,figsize=(18,8))
df['Outcome'].value_counts().plot.pie(explode=[0,0.1],autopct='
%1.1f%%',ax=ax[0],shadow=True)
```

```
ax[0].set_title('target')
ax[0].set_ylabel('')
sns.countplot('Outcome',data=df,ax=ax[1])
ax[1].set_title('Outcome')
plt.show()
# Correlation matrix of the data set
f, ax = plt.subplots(figsize= [20,15])
sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap
='magma' )
ax.set_title("Correlation Matrix", fontsize=20)
#plt.savefig("corr.png", dpi=400)
plt.show()

# The missing values will be filled with the median values of
each variable
def median_target(var):
    temp = df[df[var].notnull()]
    temp = temp[[var,
'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    return temp

# The values to be given for incomplete observations are given
the median value of people who are not sick and the median
values of people who are sick.
columns = df.columns
columns = columns.drop("Outcome")
for i in columns:
    median_target(i)
    df.loc[(df['Outcome'] == 0 ) & (df[i].isnull()), i] =
median_target(i)[i][0]
    df.loc[(df['Outcome'] == 1 ) & (df[i].isnull()), i] =
median_target(i)[i][1]

for feature in df:

    Q1 = df[feature].quantile(0.25)
    Q3 = df[feature].quantile(0.75)
    IQR = Q3-Q1
    lower = Q1- 1.5*IQR
    upper = Q3 + 1.5*IQR

    if df[(df[feature] > upper)].any(axis=None):
        print(feature,"yes")
    else:
        print(feature, "no")
Pregnancies yes
Glucose no
BloodPressure yes
SkinThickness yes
Insulin yes
BMI yes
```

```
DiabetesPedigreeFunction yes
Age yes
Outcome no

# Conducting a stand alone observation review for the Insulin
variable
# Suppressing contradictory values
Q1 = df.Insulin.quantile(0.25)
Q3 = df.Insulin.quantile(0.75)
IQR = Q3-Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR
df.loc[df["Insulin"] > upper,"Insulin"] = upper

import seaborn as sns
plt.figure(figsize=(8,7))
sns.boxplot(x = df["Insulin"], color="red");

# According to BMI, some ranges were determined and categorical
variables were assigned.
NewBMI = pd.Series(["Underweight", "Normal", "Overweight",
"Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category")
df["NewBMI"] = NewBMI
df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]
df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] =
NewBMI[1]
df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] =
NewBMI[2]
df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] =
NewBMI[3]
df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] =
NewBMI[4]
df.loc[df["BMI"] > 39.9 ,"NewBMI"] = NewBMI[5]

# splitting data into training and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.30, random_state = 0)

# scaling data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# fitting data to model
```

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

LogisticRegression()

# model predictions

y_pred = log_reg.predict(X_test)

# accuracy score

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, log_reg.predict(X_train)))

log_reg_acc = accuracy_score(y_test, log_reg.predict(X_test))
print(log_reg_acc)
0.8402255639097744
0.881578947368421
```

Model Comparison

```
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'KNN', 'SVM', 'Decision
Tree Classifier', 'Random Forest Classifier', 'Gradient
Boosting Classifier', 'XgBoost'],
    'Score': [100*round(log_reg_acc,4), 100*round(knn_acc,4),
100*round(svc_acc,4), 100*round(dtc_acc,4),
100*round(ran_clf_acc,4),
              100*round(gbc_acc,4), 100*round(xgb_acc,4)]
})
models.sort_values(by = 'Score', ascending = False)
```

	Model	Score
4	Random Forest Classifier	92.54
3	Decision Tree Classifier	89.47
5	Gradient Boosting Classifier	89.04
0	Logistic Regression	88.16
6	XgBoost	87.72
2	SVM	84.21

```
          Model  Score
1           KNN  83.33
```

```
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
models = [
{
    'label': 'LR',
    'model': log_reg,
},
{
    'label': 'DT',
    'model': dtc,
},
{
    'label': 'SVM',
    'model': svc,
},
{
    'label': 'KNN',
    'model': knn,
},
{
    'label': 'XGBoost',
    'model': xgb,
},
{
    'label': 'RF',
    'model': rand_clf,
},
{
    'label': 'GBDT',
    'model': gbc,
}
]

means_roc = []
means_accuracy = [100*round(log_reg_acc,4),
100*round(dtc_acc,4), 100*round(svc_acc,4),
100*round(knn_acc,4), 100*round(xgb_acc,4),
                  100*round(ran_clf_acc,4),
100*round(gbc_acc,4)]

for m in models:
    model = m['model']
    model.fit(X_train, y_train)
    y_pred=model.predict(X_test)
    fpr1, tpr1, thresholds = metrics.roc_curve(y_test,
model.predict_proba(X_test)[:,1])

    auc = metrics.roc_auc_score(y_test,model.predict(X_test))
    auc = 100*round(auc,4)
    means_roc.append(auc)

print(means_accuracy)
print(means_roc)

# data to plot
n_groups = 7
means_accuracy = tuple(means_accuracy)
means_roc = tuple(means_roc)

# create plot
fig, ax = plt.subplots(figsize=(8,5))
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

rects1 = plt.bar(index, means_accuracy, bar_width,
alpha=opacity,
color='mediumpurple',
label='Accuracy (%)')

rects2 = plt.bar(index + bar_width, means_roc, bar_width,
alpha=opacity,
color='rebeccapurple',
label='ROC (%)')

plt.xlim([-1, 8])
plt.ylim([60, 95])

plt.title('Performance Evaluation - Diabetes Prediction',
fontsize=12)
plt.xticks(index, ('   LR', '   DT', '   SVM', '   KNN',
'XGBoost' , '   RF', '   GBDT'), rotation=40, ha='center',
fontsize=12)


plt.legend(loc="upper right", fontsize=10)
plt.savefig("outputs/PE_diabetes.jpeg", format='jpeg', dpi=400,
bbox_inches='tight')
plt.show()
```

```
[88.16000000000001, 89.47, 84.21, 83.33, 87.72, 92.54, 89.03999
999999999]
[86.94, 75.88000000000001, 81.38, 80.97999999999999, 86.87, 90.
49000000000001, 87.62]
```