**SUMMERINTERNSHIPREPORT**

***HEART DISEASE DETECTION***

*A report submitted in partial fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

in

**Artificial Intelligence and Data Science**

**by**

**Bonu Pujitha**

**Regd.no:21B91A5426**

**Duration**

**3rd June, 2024 to 31st July, 2024**

**Under supervision of**

**Mr. Vuday Krishna**

**BlackBuck Engineers Pvt.Ltd**

**Tirupati**

**&**

**Dr.D. RatnaGiri**

**Dept. of IT**

**SRKR Engineering College(A)**



## DEPARTMENT OF INFORMATION TECHNOLOGY

## (AI&DS)

## S R K R ENGINEERING COLLEGE (Autonomous)

Approved by AICTE, permanently affiliated to JNTUK, Kakinada

## China Amiram, Bhimavaram, Andhra Pradesh-534204

## November-2024

## S R K R ENGINEERING COLLEGE

## (Autonomous)

China Amiram::Bhimavaram

## DEPARTMENT OF INFORMATION TECHNOLOGY

## (AI&DS)



***CERTIFICATE***

This is to certify that the Internship reportwork titled “Heart Disease Detection”, submitted by **B.Pujitha(Regd. No.: 21B91A5426)** during 2024 – 2025 academic year, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in Artificial Intelligence And Data Science a**t **IIDT BlackBucks.**

**Internship Mentor**

**Dr. D. RatnaGiri**

Associate Professor, IT Dept.

**Internship Coordinator**

**Dr. K. Kishore Raju**

Associate Professor, IT Dept.

**Prof. P. Ravi Kiran Varma**

**Head of AIDS Department**

**DECLARATION**

This Internship Work report entitled “Heart Disease Detection” has been carried out by us in the partial fulfillment of the requirements for the award of the degree of B.Tech Artificial Intelligence and Data Science, S.R.K.R Engineering College(A). We hereby declare this Internship work report has not been submitted to any of the other university/Institute for the award of any other degree/diploma.

|  |  |
| --- | --- |
| **Name :**  **Register No :**  **Signature :** |  |

**CERTIFICATION OF EXAMINATION**

#### This to certify that I have examined the concept and hereby accord my approval of it as the Internship Work entitled “Heart Disease Detection” carried out and presented in a manner required for its acceptance on partial fulfillments for the award of the degree of BACHELOR OF TECHNOLOGY in Artificial Intelligence And Data Science for which it has been submitted.

This approval does not necessarily endorse or accept every statement made opinion expressed or conclusions drawn as recorded in the Internship report it only signifies the acceptance of the report for the purpose for which submitted.

**Departmental Committee:**

**Signature**

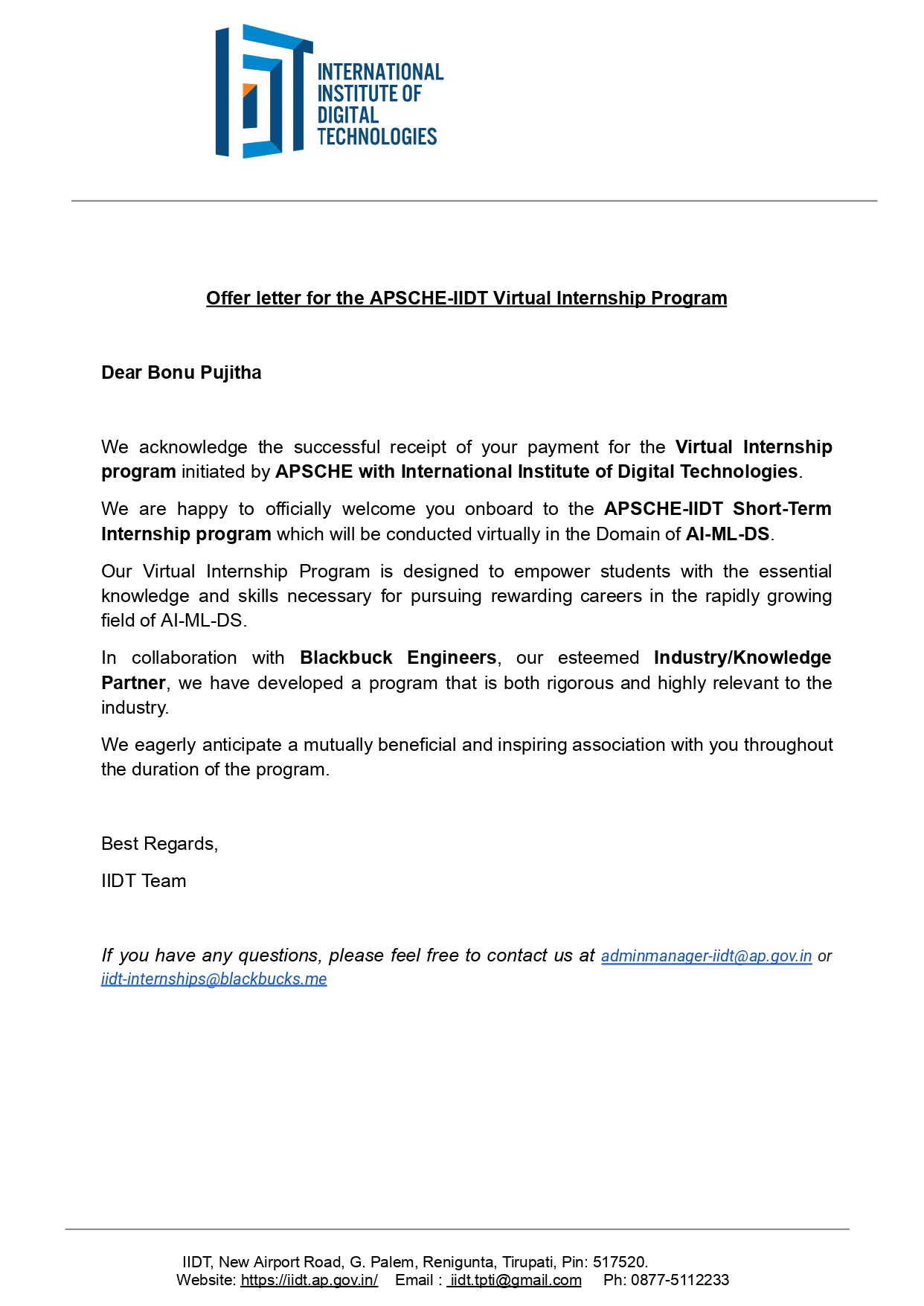
**Internship Mentor :**

**Senior Faculty :**

**External Examiner :**

**HOD :**

# 



# ACKNOWLEDGEMENT

Firstly, I would like to thank Ms. Anuradha Thota, HR, Head, of **IIDT BLACK BUCKS** for giving me the opportunity to do this internship within the organization.

I also would like to thank all the people of **IIDT BLACK BUCKS** who worked along with me with their patience and openness and created an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am highly indebted to our beloved Principal **Prof. K.V. Murali Krishnam Raju** for the facilities provided to accomplish this internship.

I would like to thank my Head of the Department **Prof. P. Ravi Kiran Varma** for his constructive criticism throughout my internship.

And also I would like to thank **Dr. K. Kishore Raju** internship coordinator, Department of IT for the support and advices he has given me to get and complete the internship in the above said organization.

I am extremely grateful to my department staff members and friends who have helped me in the successful completion of this internship.

**B. Pujitha**

**(21B91A5426)**

**ABSTRACT:**

Data mining has been used for many purposes, especially for prediction system. In healthcare, data mining algorithms are always used in disease diagnosis. Heart disease is known as a primary cause of death over the years. Nowadays, it is one of the world’s most dangerous human heart diseases and has very serious effects the human life. Accurate and timely identification of human heart disease can be very helpful in preventing heart failure in its early stages and will improve the patient’s survival. Many studies have been performed in heart disease diagnosis using data mining methods. In this paper there are some 10 popular machine learning algorithms that can be used in heart disease prediction, for example, Logistic Regression, Naïve Bayes, SVM, Random Forest, Decision Trees, Extra Trees Classifier, Bagging Classifier, AdaBoost, Gradient Boosting, XG Boost were implemented at various evaluation stages to predict heart diseases. In this paper ETC is the best model with an Accuracy is 82% with a precision of 92% and recall of 88%. Then optimized the classification results using one of the hyperparameters optimization techniques random search that shows the best accuracy was 87%.

**Keywords**: Machine learning (ML), Decision tree (DT), Random Forest, Extra Trees Classifier, AdaBoost with Base Classifier as ETC, Hyperparameters*.*

**Learning Objectives/Internship Objectives**

* Internships are generally thought of to be reserved for college students looking to gain experience in a particular field. However, a wide array of people can benefit from Training Internships in order to receive real world experience and develop their skills.
* An objective for this position should emphasize the skills you already possess in the area and your interest in learning more
* Internships are utilized in a number of different career fields, including architecture, engineering, healthcare, economics, advertising and many more.
* Some internship is used to allow individuals to perform scientific research while others are specifically designed to allow people to gain first-hand experience working.
* Utilizing internships is a great way to build your resume and develop skills that can be emphasized in your resume for future jobs. When you are applying for a Training Internship, make sure to highlight any special skills or talents that can make you stand apart from the rest of the applicants so that you have an improved chance of landing the position.

**WEEKLY OVERVIEW OF INTERNSHIP ACTIVITIES**

|  |  |  |  |
| --- | --- | --- | --- |
| **1st WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 03/06/24 | Monday | Introduction to python |
| 05/06/24 | Wednesday | Foundations of data science |
| 07/06/24 | Friday | Mathematics for data science |

|  |  |  |  |
| --- | --- | --- | --- |
| **2nd WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 10/06/24 | Monday | Exploratory data analysis |
| 12/06/24 | Wednesday | Linear regression |
| 14/06/24 | Friday | Logistic regression |

|  |  |  |  |
| --- | --- | --- | --- |
| **3rd WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 17/06/24 | Monday | K-Nearest neighbors |
| 19/06/24 | Wednesday | Decision tree and model evaluation |
| 21/06/24 | Friday | Random forest and bagging |

|  |  |  |  |
| --- | --- | --- | --- |
| **4th WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 24/06/24 | Monday | Model evaluation techniques |
| 26/06/24 | Wednesday | Advanced algorithms |
| 28/06/24 | Friday | Unsupervised learning |

|  |  |  |  |
| --- | --- | --- | --- |
| **5h WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 01/07/24 | Monday | Hyper parameter tuning techniques |
| 03/07/24 | Wednesday | project |
| 05/07/24 | Friday | Project session |

|  |  |  |  |
| --- | --- | --- | --- |
| **6th WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 08/07/24 | Monday | K-means clustering |
| 10/07/24 | Wednesday | Hierarchical clustering |
| 12/07/24 | Friday | DBSCAN |

|  |  |  |  |
| --- | --- | --- | --- |
| **7th WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 15/07/24 | Monday | Dimensionality reduction |
| 17/07/24 | Wednesday | feature selection |
| 19/07/24 | Friday | Project session |

|  |  |  |  |
| --- | --- | --- | --- |
| **8th WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 22/07/24 | Monday | Introduction to neural networks |
| 24/07/24 | Wednesday | Training neural networks |
| 26/07/24 | Friday | Evaluation of neural networks |

|  |  |  |  |
| --- | --- | --- | --- |
| **9th WEEK** | **DATE** | **DAY** | **NAME OF THE TOPIC/MODULE COMPLETED** |
| 28/07/24 | Monday | Final project selection and setup |
| 29/07/24 | Wednesday | Project implementation and tuning |
| 31/07/24 | Friday | Final presentation and documentation |

**Index**

**TABLE OF CONTENTS Pg.no**

1.0 Introduction ………………………………………………………………………………..2

2.0 System Analysis…………………………………………………………………………..3

3.0 System Architecture and Module Description………………………………………….4

4.0 Software Requirements Specifications…………………………………………………5

5.0 Technology…………………………………………………………………………….6-26

5.1. Machine Learning

5.2 Prediction of Heart Disease

5.3 AI / ML Modelling and Results

5.4 Handling of Outliers

5.5 Models Used for Development

6.0 System Implementation and Results………………………………………………27-38

7.0 Output screens………………………………………………………………………….39

8.0 Conclusions……………………………………………………………………………..40

9.0 Biblography……………………………………………………………………………...41

**1.INTRODUCTION:**

Heart attacks are serious and often fatal health problems that affect many people worldwide. Being able to predict who is at high risk of having a heart attack early is very important because it allows doctors to help these individuals before it's too late. Now, there's a tool called machine learning that can help with this prediction. Machine learning uses computers to analyze lots of information, like age, medical history, and lifestyle, to make predictions. It's like a smart detective that looks at many clues to figure out if someone might have a heart attack.One specific machine learning method we're looking at is called logistic regression. It's good at saying if someone is likely or unlikely to have a heart attack. We're using this method to study real data and see how good it is at predicting heart attacks. Our goals are to create accurate models, use numbers to measure how good they are, and find out which factors are most important for predicting heart attacks. This research could help doctors save lives and teach people how to lower their risk of heart attacks. It's a way that computers and numbers can make healthcare better.

Top of Form

**2. SYSTEM ANALYSIS**

**2.1 Existing System:**

In the existing system for heart disease detection, traditional approaches primarily rely on rule-based systems or decision tree algorithms to predict the likelihood of heart disease based on patient data. These systems generally follow predefined rules for evaluating risk factors such as age, blood pressure, cholesterol levels, and medical history. While these methods can offer reasonable accuracy, they often suffer from limitations like lack of adaptability to new data or evolving trends in patient information. The existing design involves static models that are retrained periodically based on historical data, which may not reflect the most up-to-date medical findings or patient-specific nuances. Moreover, the decision-making process is often based on oversimplified risk factors, potentially leading to less personalized treatment recommendations.

**2.2 Proposed System:**

The proposed system for heart disease detection leverages machine learning algorithms to provide more accurate and adaptive predictions based on an extensive range of features. We extend traditional models by incorporating advanced classification techniques, such as Support Vector Machines (SVM), Random Forest, and Deep Neural Networks (DNN), to better capture complex relationships in the data. This system includes the following steps:

Data Preprocessing: Clean and transform raw patient data (e.g., age, sex, cholesterol, ECG, family history) into usable formats.

Feature Engineering: Automatically select and extract relevant features that significantly impact prediction accuracy, such as blood sugar levels, resting blood pressure, and cholesterol ratio.

Model Training: Using classification models like SVM and Random Forest, the system incrementally learns from patient data, ensuring it adapts to new patterns in heart disease indicators over time.

Model Evaluation and Tuning: Models are continually evaluated and tuned using cross-validation to achieve high accuracy, sensitivity, and specificity, ensuring robust predictions for a wide range of patients.

Integration with Real-time Data: The proposed system accommodates real-time patient data streams, allowing it to process and predict heart disease risk in a dynamic, evolving environment.

In contrast to existing systems, the proposed system provides more personalized and adaptive decision-making, offering improved precision in predicting heart disease and recommending preventative measures.

**3. SYSTEM ARCHITECTURE AND MODULE DESCRIPTION**

**3.1 System Architecture:**

The system architecture for the Heart Disease Detection project is designed to manage the entire workflow of data processing, model training, evaluation, and prediction. The architecture includes components for data ingestion, preprocessing, model building, evaluation, and prediction delivery. It leverages machine learning algorithms and various preprocessing techniques to ensure accurate predictions of heart disease risk based on input data.

**3.2 Module Description:**

**Data Collection and Preprocessing:**

Collect and process the data, ensuring it includes relevant features like age, gender, blood pressure, cholesterol levels, and other cardiovascular risk factors.

Handle missing values and outliers, standardize or normalize data where necessary, and encode categorical features to be used in the machine learning models.

**Model Training:**

Implement multiple machine learning algorithms, including Logistic Regression, Decision Trees, Random Forest, Extra Trees Classifier, K-Nearest Neighbors, Naïve Bayes, Support Vector Machine, and XGBoost, to build predictive models.

Each model is trained using the training dataset, with evaluation metrics such as accuracy, precision, recall, and F1 score recorded to assess performance.

**Evaluation and Model Selection:**

Evaluate each model's performance on the test dataset, focusing on metrics like ROC-AUC score, confusion matrix, and MCC (Matthews Correlation Coefficient) to determine the most suitable model for heart disease prediction.

Hyperparameter optimization is conducted on the selected model to enhance accuracy and generalizability.

**Prediction and Risk Assessment:**

The selected model (Logistic Regression in this case) is deployed to predict heart disease risk for new input data.

Output risk predictions with probabilities and categorization to assist healthcare providers in early diagnosis and interven

**4.SOFTWARE REQUIREMENTS SPECIFICATIONS**

## System configurations

The Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for the Heart Disease Detection system. This includes detailing the software functions, performance criteria, system behavior, design constraints, and validation requirements necessary for implementing a robust heart disease prediction model using machine learning..

## 4.2 Software Requirements:

* Operating System: Windows 10 or later / Linux (Ubuntu 18.04 or later) / macOS.
* Programming Language: Python 3.8+.
* Libraries and Frameworks:
* Data Processing: Pandas, NumPy
* Machine Learning: Scikit-Learn, XGBoost, LightGBM
* Visualization: Matplotlib, Seaborn
* Development Environment: Jupyter Notebook, Visual Studio Code, or PyCharm.
* Database: SQLite or MySQL for storing processed datasets and model results (if needed).
* Version Control: Git for version management.

**4.3 Hardware Requirements:**

* Processor: Intel i5 7th generation or equivalent (i7 recommended for faster processing).
* Hard Disk: Minimum 256 GB SSD (512 GB recommended) or 1 TB HDD for data storage.
* RAM: 8 GB (16 GB recommended for handling larger datasets and smoother processing).
* Graphics: GPU (NVIDIA GTX 1050 or higher) for model training acceleration (optional but recommended for deep learning models).

# **5. TECHNOLOGY**

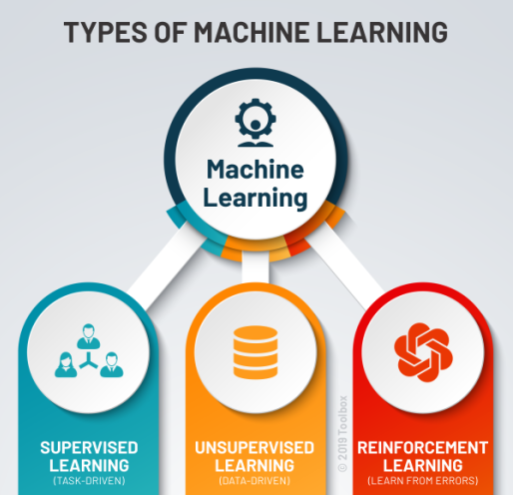
**5.1 What are the different types of Machine Learning?**

## How Machine Learning Works

Machine learning uses two types of techniques: Supervised learning, which trains a model on known input and output data so that it can predict

Future outputs and Unsupervised learning, which finds hidden patterns or intrinsic structuresin input data.

**.**



**Supervised Learning :**

Supervised machine learning builds a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data. Use supervised learning if you have known data for the output you are trying to predict.

**Supervised learn**ing uses **Regression** and **Classification** techniques to develop predictive models.

**Regression techniques** predict continuous responses - for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.Use regression techniques if you are working with a data range or if the nature of your response is a real number, such as temperature or the time until failure for a piece of equipment. Common regression algorithms include linear model, nonlinear model, stepwise regression, Gradient Descent Regression, Support Vector Regression, Ridge and Lasso Regressions.

**Classification techniques** predict discrete responses - for example, whether an email is genuine or spam, or whether a tumour is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring. Use classification if your data can be tagged, categorized, or separated into specific groups or classes.

For example, applications for hand-writing recognition use classification to recognize letters and numbers. In image processing and computer vision, unsupervised pattern recognition techniques are used for object detection and image segmentation.

Common algorithms for performing classification include support vector machine (SVM), boosted and bagged decision trees, k-nearest neighbour, Naïve Bayes, discriminant analysis, logistic regression, and neural networks.

Using Supervised Learning to Predict Heart Attacks: Suppose clinicians want to predict whether someone will have a heart attack within a year. They have data on previous patients, including age, weight, height, and blood pressure. They know whether the previous patients had heart attacks within a year. So, the problem is combining the existing data into a model that can predict whether a new person will have a heart attack within a year.

**Unsupervised Learning**

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without

labelled responses.

**Clustering** is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Applications for cluster analysis include gene sequence analysis and market research.

For example, if a cell phone company wants optimize the locations where they build cell phone towers, they can use machine learning to estimate the number of clusters of people relying on their towers. A phone can only talk to one tower at a time, so the team uses clustering algorithms to design the best placement of cell towers to optimize signal reception for groups, or clusters, of their customers.

Common algorithms for performing clustering include k-means and k-medoids, Apriori algorithms, hierarchical clustering, Gaussian mixture models and hidden Markov models.

**5.2 Benefits of Using Machine Learning in Heart disease prediction:**

Globally, cardiovascular disease (CVDs) is the primary cause of morbidity and mortality, accounting for more than 70% of all fatalities. According to the 2017 Global Burden of Disease research, cardiovascular disease is responsible for about 43% of all fatalities. Common risk factors for heart disease in high-income nations include lousy diet, cigarette use, excessive sugar consumption, and obesity or excess body fat. However, low- and middle-income nations also see a rise in chronic illness prevalence.

Cardiovascular disease is a leading cause of global mortality, impacting both high- and low-income countries. Expensive diagnostic technologies and a lack of early detection contribute to a high mortality rate.

Using machine learning for heart disease prediction offers benefits like improved accuracy, early detection, personalized risk assessment, and faster diagnosis. It can identify new risk factors, reduce human error, and aid in continuous monitoring. Machine learning can also optimize resource allocation, save costs, and support research. However, challenges like data quality and ethical considerations must be addressed in its implementation in healthcare.

**5.3 About Industry:**

The heart is a kind of muscular organ which pumps blood into the body and is the central part of the body’s cardiovascular system which also contains lungs. Cardiovascular system also comprises a network of blood vessels, for example, veins, arteries, and capillaries. These blood vessels deliver blood all over the body. Abnormalities in normal blood flow from the heart cause several types of heart diseases which are commonly known as cardiovascular diseases (CVD). Heart diseases are the main reasons for death worldwide. According to the survey of the World Health Organization (WHO), 17.5 million total global deaths occur because of heart attacks and strokes. More than 75% of deaths from cardiovascular diseases occur mostly in middle-income and low-income countries. Also, 80% of the deaths that occur due to CVDs are because of stroke and heart attack . Therefore, detection of cardiac abnormalities at the early stage and tools for the prediction of heart diseases can save a lot of life and help doctors to design an effective treatment plan which ultimately reduces the mortality rate due to cardiovascular diseases.

Machine learning in heart disease prediction is a rapidly expanding sector that employs advanced data analytics and artificial intelligence to enhance the detection and diagnosis of heart diseases. This industry includes healthcare startups, providers, medical imaging companies, data analytics service providers, pharmaceutical firms, medical device manufacturers, research institutions, health insurers, government regulators, data scientists, and even patients. The focus is on creating more accurate risk assessments and diagnostic tools to improve patient care and reduce the global impact of heart diseases.

**5.4 AI/ML Role in Heart Disease Prediction:**

Artificial Intelligence (AI) and Machine Learning (ML) play a crucial role in heart disease prediction by leveraging data analysis and predictive modeling.

Data Analysis: AI and ML excel at analyzing vast amounts of patient data, including medical records, diagnostic tests, lifestyle information, and genetic profiles. This comprehensive data analysis aids in risk assessment.

Pattern Recognition: AI/ML can identify intricate patterns and relationships within the data that may not be apparent to human analysts. This aids in the discovery of hidden risk factors for heart disease.

Early Detection: One of the most significant roles is the ability to detect heart disease at an early stage. AI/ML models can identify individuals at risk even before symptoms manifest, enabling timely intervention and improved outcomes.

Personalized Risk Assessment: AI/ML provides highly individualized risk assessments by considering a person's unique combination of risk factors, medical history, genetics, and lifestyle. This leads to tailored treatment and prevention plans.

Data Integration: AI/ML models can integrate data from various sources, such as electronic health records, wearable devices, and genetic profiles. This holistic approach offers a more accurate view of a patient's health and cardiovascular risk.

Predictive Analytics: Beyond predicting the likelihood of heart disease, AI/ML models can estimate the probability of future cardiovascular events like heart attacks, strokes, or arrhythmias. This information guides proactive medical interventions.

Automation and Efficiency: Automation streamlines the diagnostic process, reducing the time needed for risk assessment. This not only improves efficiency but also ensures that healthcare professionals can dedicate more time to patient care.

Identification of Novel Risk Factors: AI/ML can identify previously unrecognized risk factors for heart disease by analyzing large datasets. This data-driven approach enhances our understanding of the condition and contributes to the development of better prevention strategies.

Reduction in Human Error: By automating aspects of diagnosis and risk assessment, AI/ML reduces the potential for human error, ensuring consistent and objective evaluations.

Continuous Monitoring: AI and ML enable ongoing monitoring of patients' health, allowing for real-time detection of changes and emerging risk factors, facilitating timely interventions.

Resource Optimization: AI/ML models help healthcare systems allocate resources more efficiently. They identify high-risk patients in need of immediate attention, reducing unnecessary tests and procedures for low-risk individuals.

Cost Savings: Early detection and prevention facilitated by AI/ML can potentially reduce healthcare costs associated with heart disease treatment, hospitalization, and long-term care.

The application of AI and ML in heart disease prediction is a promising approach to addressing the global burden of cardiovascular conditions. It holds the potential to enhance patient outcomes and reduce the overall healthcare system's burden by enabling earlier diagnosis and more effective preventive measures.

**5.5 Prediction of Heart Disease:**

Predicting heart disease involves using machine learning and artificial intelligence to assess a person's risk of developing cardiovascular problems. This process includes collecting and preparing medical data, selecting relevant features, training a machine learning model, and evaluating its performance. If successful, the model can be used to assist healthcare professionals in making more accurate predictions, but it should always be used in conjunction with clinical expertise for diagnosis and treatment decisions**.**

**2.1 Internship Project-Data Link:**

The internship project data has taken from kaggle and the link is

“<https://www.kaggle.com/code/besenturk/logistic-regression-to-predict-heart-disease>”

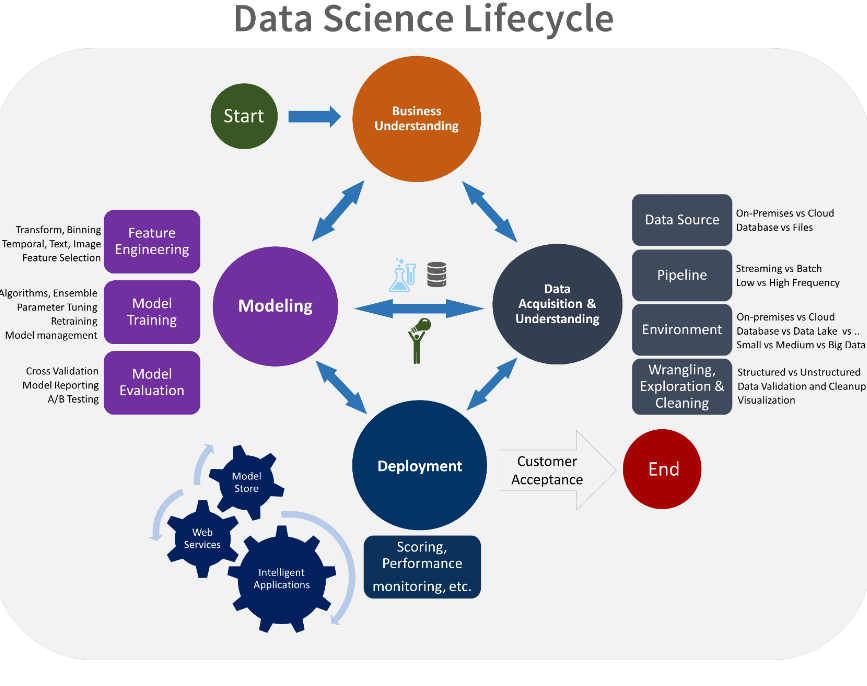
**3.0 AI/ML Modelling and Results:**

**3.1 Problem Statement:**

To predict heart diseases by using different algorithms of AI/ML and to accurate and best algorithm with evaluation.

**3.2 Data Science Project Life Cycle:**

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



**3.2.1 Data Exploratory Analysis:**

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable. The exploratory analysis is done for heart disease prediction with different parameters and all the charts are presented in Appendices 6.2 - List of charts(6.2.1 to 6.2.9)

**3.2.2 Data Pre-processing:**

We removed variables which does not affect our target variable(Status Target)as they may add noise and also increase our computation time we checked the data for anomalous data points and outliers.We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

**3.2.2.1 Check the Duplicate and low variance data:**

Duplicate Values: When two features have the same set of values. Duplicate Index: When the value of two features are different, but they occur at the same index.

Steps to delete duplicates:

1.Use the get duplicate features functions to get all the constant features. 2. Store all the duplicate features as a list for removing from the dataset. 3.Drop all such features from the dataset.

HDdata\_dup=HDdata[HDdata.duplicated(keep='last')]

HDdata\_dup

**OUTPUT:**

No values found,Since it doesn’t has any duplicate values no need of deleting them.

**3.2.2.2 Identify and address the missing variables:**

**How to Identify Missing Values?**

We can check for null values in a derived dataset. But, sometimes, it might

not be this simple to identify missing values. One needs to use the domain

knowledge and look at the data description to understand the variables. There are variables that have a minimum value of zero. On some columns, a value of zero does not make sense and indicates an invalid or missing value. Quick Classification of Missing Data.

There are three types of missing data as below:

**Missing Completely At Random (MCAR):** It is the highest level of

randomness. This means that the missing values in any features are not

dependent on any other feature’s values. This is the desirable scenario in

case of missing data.

**Missing At Random (MAR):** This means that the missing values in any feature are dependent on the values of other features.

**Missing Not At Random (MNAR):** Missing not at random data is a more

serious issue and, in this case, it might be wise to check the data gathering

process further and try to understand why the information is missing.

**What to Do with the Missing Values?**

We identified the missing values in a derived dataset, next we should decide the further course of action.

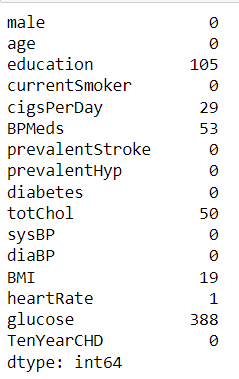
* Ignore the missing values
* Drop the missing values
* Case Deletion
* Imputation
* None: None is a Python singleton object that is often used for missing data in Python code.
* NaN: Nan (an acronym for Not a Number), is a special floating-point

value recognized by all systems that use the standard IEEE

floating- point representation.

HDdata.isnull().sum()

**OUTPUT:**



**To handle missing values:**

# By using KNNImputer

from sklearn.impute import KNNImputer

Imputer\_knn=KNNImputer(missing\_values=np.nan)

HDdata['education']=Imputer\_knn.fit\_transform(HDdata[['education']])

HDdata['cigsPerDay']=Imputer\_knn.fit\_transform(HDdata[['cigsPerDay']])

HDdata['BPMeds']=Imputer\_knn.fit\_transform(HDdata[['BPMeds']])

HDdata['totChol']=Imputer\_knn.fit\_transform(HDdata[['totChol']])

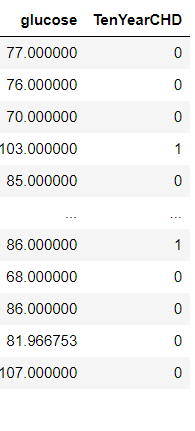
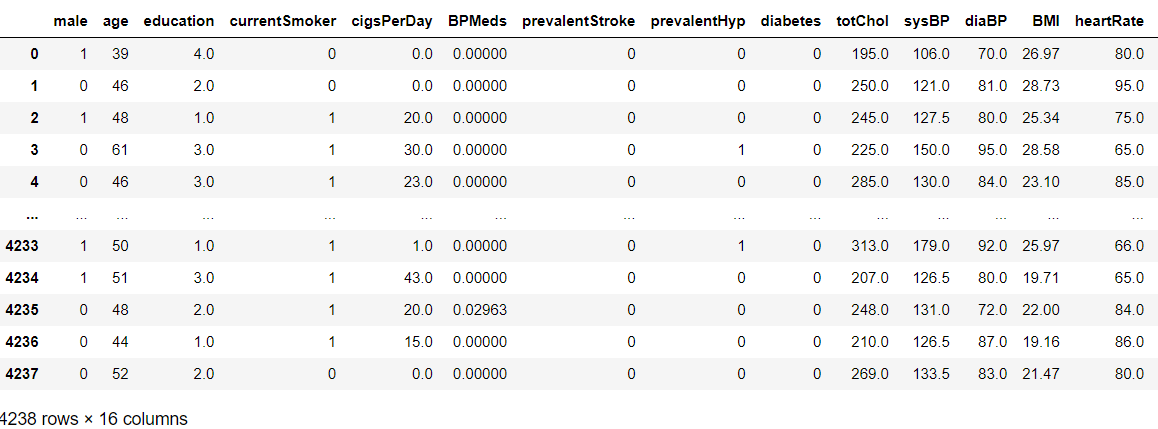
HDdata['BMI']=Imputer\_knn.fit\_transform(HDdata[['BMI']])

HDdata['heartRate']=Imputer\_knn.fit\_transform(HDdata[['heartRate']])

HDdata['glucose']=Imputer\_knn.fit\_transform(HDdata[['glucose']])

HDdata

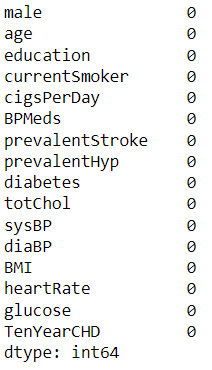
**OUTPUT:**



Missing values:

HDdata.isnull().sum()

OUTPUT:



**3.3 Handling of Outliers:**

An outlier is a data point in a data set that is distant from all other

observations. A data point that lies outside the overall distribution of

dataset (95% of customer behaviour / claims / spending nature of

customer).Detecting outliers or anomalies is one of the core problems in

data mining. The emerging expansion and continued growth of data and

the spread of IoT devices, make us rethink the way we approach

anomalies and the use cases that can be built by looking at those

anomalies. We now have smart watches and wristbands that can detect

our heartbeats every few minutes. Detecting anomalies in the heartbeat

data can help in predicting heart diseases. Anomalies in traffic patterns

can help in predicting accidents. It can also be used to identify

bottlenecks in network infrastructure and traffic between servers.

Hence, the use cases and solution built on top of detecting anomalies

are limitless. Another reason why we need to detect anomalies is that

when preparing datasets for machine learning models, it is really

important to detect all the outliers and either get rid of them or analyse

them to know why you had them there in the first place.

**Method 1 - Standard Deviation:**

In statistics, if a data distribution is approximately normal then about

68% of the data values lie within one standard deviation of the mean and

about 95% are within two standard deviations, and about 99.7% lie

within three standard deviations. Therefore, if you have any data point

that is more than 3 times the standard deviation, then those points are

very likely to be anomalous or outliers.

**Z score**

Z score indicates how many standard deviation away a data point.

Calculate the **Z score = (X - m)/Sigma**,

where m = mean, Sigma = standard deviation

**Method 2 - Boxplots**:

Box plots are a graphical depiction of numerical data through their

quantiles. It is a very simple but effective way to visualize outliers. Think

about the lower and upper whiskers as the boundaries of the data

distribution. Any data points that show above or below the whiskers, can

be considered outliers or anomalous.

HDdata.shape

**OUTPUT:**

(4238, 16)

This doesn’t have any outliers.

**3.3.1.1 Categorical data and Encoding Techniques:**

A categorical variable is one that has two or more categories (values).

There are two types of categorical variable, nominal and ordinal. A

nominal variable has no intrinsic ordering to its categories. For example,

gender is a categorical variable having two categories (male and

female) with no intrinsic ordering to the categories. An ordinal variable

has a clear ordering.

Many ML algorithms are unable to operate on categorical or label data

directly. However, Decision tree can directly learn from such data.

Hence, they require all input variables and output variables to be

numeric. This means that categorical data must be converted to a

numerical form. Few types of categorical variable encoding are:

1.One hot encoding: Encoding each categorical variable with different

Boolean variables (also called dummy variables) which take values 0 or

1, indicating if a category is present in an observation.

2.Integer Encoding / Label Encoding: Replace the categories by a

number from 1 to n (or 0 to n-1, depending the implementation), where n

is the number of distinct categories of the variable.

3.Count or frequency encoding: Replace the categories by the count of

the observations that show that category in the dataset. Similarly, we

can replace the category by the frequency -or percentage- of

observations in the dataset. That is, if 10 of our 100 observations show

the colour blue, we would replace blue by 10 if doing count encoding, or

by 0.1 if replacing by the frequency.

4. Ordered Integer Encoding: Categories are replaced by integer 1 to k,

where k is the distinct categories in variable, but this numbering is

decided by mean of target of each category.

HDdata.info()

**OUTPUT :**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4238 entries, 0 to 4237

Data columns (total 16 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 male 4238 non-null int64

1 age 4238 non-null int64

2 education 4133 non-null float64

3 currentSmoker 4238 non-null int64

4 cigsPerDay 4209 non-null float64

5 BPMeds 4185 non-null float64

6 prevalentStroke 4238 non-null int64

7 prevalentHyp 4238 non-null int64

8 diabetes 4238 non-null int64

9 totChol 4188 non-null float64

10 sysBP 4238 non-null float64

11 diaBP 4238 non-null float64

12 BMI 4219 non-null float64

13 heartRate 4237 non-null float64

14 glucose 3850 non-null float64

15 TenYearCHD 4238 non-null int64

dtypes: float64(9), int64(7)

memory usage: 529.9 KB

**3.3.1.2 Feature Scaling:**

**Why Feature Scaling?**

Real Life Datasets have many features with a wide range of values like

for example let’s consider the house price prediction dataset. It will have

many features like no. of. bedrooms, square feet area of the house, etc.

As you can guess, the no. of bedrooms will vary between 1 and 5, but

the square feet area will range from 500-2000. This is a huge difference

in the range of both features. Many machine learning algorithms that

are using Euclidean distance as a metric to calculate the similarities will

fail to give a reasonable recognition to the smaller feature, in this case,

the number of bedrooms, which in the real case can turn out to be an

important metric. E.g.: Logistic Regression, KNN,EM Results. There are

several ways to do feature scaling. I will be discussing the top 5 of the

most used feature scaling techniques.

# Scaling the features by using MinMaxScaler

from sklearn.preprocessing import MinMaxScaler

mmscaler = MinMaxScaler(feature\_range=(0, 1))

x\_train = mmscaler.fit\_transform(x\_train)

x\_train = pd.DataFrame(x\_train)

x\_test = mmscaler.fit\_transform(x\_test)

x\_test = pd.DataFrame(x\_test)

**3.3.2 Selection of Dependent and Independent variables**

The dependent or target variable here Status Target which tells us a

particular policy holder has filed a claim or not the target variable is

selected based on our business problem and what we are trying to

predict.

The independent variables are selected after doing exploratory data

analysis and we used Boruta to select which variables are most affecting

our target variable.

**3.3.3 Data Sampling Methods**

The data we have is highly unbalanced data so we used some sampling

methods which are used to balance the target variable so we our model

will be developed with good accuracy and precision. We used three

Sampling methods

**3.3.3.1 Stratified sampling**

Stratified sampling randomly selects data points from majority class so

they will be equal to the data points in the minority class. So, after the

sampling both the class will have same no of observations.It can be

performed using strata function from the library sampling.

**3.3.3.2 Simple random sampling**

Simple random sampling is a sampling technique where a set

percentage ofthe data is selected randomly. It is generally done to

reduce bias in thedataset which can occur if data is selected manually

without randomizing thedataset. We used this method to split the

dataset into train dataset whichcontains 70% of the total data and test

dataset with the remaining 30% of thedata.

**3.3.4 Models Used for Development**

We built our predictive models by using the following ten algorithms

**3.3.4.1 Model 01**

Logistic uses logit link function to convert the likelihood values to

probabilities so we can get a good estimate on the probability of a

particular observation to be positive class or negative class.The also

gives us p-value of the variables which tells us about significance of

each independent variable.

**3.3.4.2 Model 02**

Random forest is an algorithm that consists of many decision trees. It

was first developed by Leo Breiman and Adele Cutler. The idea behind it

is to build several trees, to have the instance classified by each tree, and

to give a "vote" at each class. The model uses a "bagging" approach and

the random selection of features to build allocation of decision trees with

controlled variance. The instance's class is to the class with the highest

number of votes,the class that occurs the most within the leaf in which

the instance is placed.

The error of the forest depends on:

* Trees correlation: the higher the correlation, the higher the forest

error rate.

* The strength of each tree in the forest. A strong tree is a tree

with low error. By using trees that classify the instances with low

error the error rate of the forest

decreases.

**3.3.4.3 Model 03(Random forest)**

Random forest is an algorithm that consists of many decision trees. It

was first developed by Leo Bierman and Adele Cutler. The idea behind it

is to build several trees, to have the instance classified by each tree, and

to give a "vote" at each class. The model uses a "bagging" approach and

the random selection of features to build a collection of decision trees

with controlled variance. The instance's class is to the class with the

highest number of votes, the class that occurs the most within the leaf in

which the instance is placed.

The error of the forest depends on:

* Trees correlation: the higher the correlation, the higher the forest

error rate.

* The strength of each tree in the forest. A strong tree is a tree with low

error. By using trees that classify the instances with low error the error rate of the forest.

**3.3.4.4 Model 04(Extra Tree Classifier)**

Extremely Randomized Trees Classifier (Extra Trees Classifier) is a type

of ensemble learning technique which aggregates the results of multiple

de-correlated decision trees collected in a “forest” to output its

classification result. In concept, it is very similar to a Random Forest

Classifier and only differs from it in the manner of construction of the

decision trees in the forest.

**3.3.4.5 Model 05(KNN Classifier)**

K-Nearest Neighbour is one of the simplest Machine Learning algorithms

based on Supervised Learning technique. K-NN algorithm assumes the

similarity between the new case/data and available cases and put the

new case into the category that is most similar to the available

categories. K-NN algorithm stores all the available data and classifies a

new data point based on the similarity. This means when new data

appears then it can be easily classified into a well suite category by

using K- NN algorithm. K-NN algorithm can be used for Regression as

well as for Classification but mostly it is used for the Classification

problems. K- NN is a non-parametric algorithm, which means it does not

make any assumption on underlying data. It is also called a lazy learner

algorithm because it does not learn from the training set immediately

instead it stores the dataset and at the time of classification, it performs

an action on the dataset. KNN algorithm at the training phase just stores

the dataset and when it gets new data, then it classifies that data into a

category that is much similar to the new data.

**3.3.4.6 Model 06(GaussianNB)**

Naïve Bayes is a probabilistic machine learning algorithm used for

many classification functions and is based on the Bayes theorem.

Gaussian Naïve Bayes is the extension of naïve Bayes. While other

functions are used to estimate data distribution, Gaussian or normal

distribution is the simplest to implement as you will need to calculate the

mean and standard deviation for the training data.

**3.3.4.7 Model 07(SVC)**

Support Vector Machine(SVM) is a supervised machine learning

algorithm used for both classification and regression. Though we say

regression problems as well its best suited for classification. The

objective of SVM algorithm is to find a hyperplane in an N- dimensional

space that distinctly classifies the data points. The

dimension of the hyperplane depends upon the number of features. If

the number of input features is two, then the hyperplane is just a line.

If the number of input features is three, then the hyperplane becomes

a 2-D plane. It becomes difficult to imagine when the number of

features exceeds three.

**3.3.4.8 Model 08(XGB Classifier)**

XGB is an implementation of Gradient Boosted decision trees. This

library was written in C++. It is a type of Software library that was

designed basically to improve speed and model performance. It has

recently been dominating in applied machine learning. XGB models

majorly dominate in many Kaggle Competitions. In this algorithm, decision

trees are created in sequential form. Weights play an

important role in XGB. Weights are assigned to all the independent

variables which are then fed into the decision tree which predicts

results. The weight of variables predicted wrong by the tree is

increased and the variables are then fed to the second decision tree.

These individual classifiers/predictors then ensemble to give a strong

and more precise model. It can work on regression, classification, ranking,

and user-defined prediction problems.

**3.3.4.9 Model 09(LGBM Classifier)**

LGBM is a gradient boosting framework based on decision trees to

increases the efficiency of the model and reduces memory usage. It uses

two novel techniques: Gradient-based One Side Sampling and Exclusive

Feature Bundling (EFB) which fulfills the limitations of histogram-based

algorithm that is primarily used in all GBDT (Gradient Boosting Decision

Tree) frameworks. The two techniques of GOSS and EFB described

below form the characteristics of LGBM Algorithm. They comprise

together to make the model work efficiently and provide it a cutting edge

over other GBDT frameworks Gradient-based One Side Sampling

Technique for LGBM: Different data instances have varied roles in the

computation of information gain.

**3.4 AI / ML Models Analysis and Final Results**

We used our train dataset to build the above models and used our test

data to check the accuracy and performance of our models.We used

confusion matrix to check accuracy, Precision, Recall and F1 score of

our models and compare and select the best model for given auto

dataset of size ~ 23409 policies.

**3.4.1 Different Model codes**

The Python code for models with stratified sampling technique as

follows:

* Logistic Regression
* Decision Tree Classifier
* Random Forest Classifier
* Extra Trees Classifier
* K Neighbors Classifier
* Gaussian NB
* SVC
* Bagging Classifier
* Gradient Boosting Classifier
* Lightgbm

**6.SYSTEM IMPLEMENTATION and RESULTS**

**6.1 Implementation Details:**

# To build the 'Multi Label Decision Tree' model with random sampling

from s k learn. linearmodel import Logistic Regression

from sklearn. tree import Decision Tree Classifier

from sklearn. ensemble import Random Forest Classifier

from sklearn. ensemble import Extra Trees Classifier

from sklearn. svm import SVC

from sklearn. naivebayes import Gaussian NB

from sklearn. neighbors import KNeighbors Classifier

**CODE:**

# Build the Calssification models and compare the results

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

# Create objects of classification algorithm with default hyper-parameter

ModelLR = LogisticRegression()

ModelDC = DecisionTreeClassifier()

ModelRF = RandomForestClassifier()

ModelET = ExtraTreesClassifier()

ModelKNN=KNeighborsClassifier(n\_neighbors=5)

ModelSVM=SVC(kernel='rbf',random\_state=42,class\_weight='balanced',

probability=True)

ModelGNB = GaussianNB()

# Evalution matrix for all the algorithms

MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN, ModelSVM,

ModelGNB] for models in MM:

# Fit the model models.

fit(x\_train, y\_train)

# Prediction

y\_pred = models.predict(x\_test)

y\_pred\_prob = models.predict\_proba(x\_test)

# Print the model name

print('Model Name: ', models)

# confusion matrix in sklearn

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

# actual values

actual = y\_test

#predicted values

predicted = y\_pred

#confusion matrix matrix = confusion\_matrix(actual,predicted,

labels=[1,0],sample\_weight=None, normalize=None) print('Confusion

matrix : \n', matrix)

# outcome values order in sklearn

tp, fn, fp, tn = confusion\_matrix(actual,predicted,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy

C\_Report = classification\_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C\_Report)

# calculating the metrics

sensitivity = round(tp/(tp+fn), 3);

specificity = round(tn/(tn+fp), 3);

accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);

balanced\_accuracy = round((sensitivity+specificity)/2, 3);

precision = round(tp/(tp+fp), 3);

f1Score = round((2\*tp/(2\*tp + fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1

to +1.

# A model with a score of +1 is a perfect model and -1 is a poor model

from math import sqrt mx = (tp+fp) \* (tp+fn) \* (tn+fp) \* (tn+fn)

MCC = round(((tp \* tn) - (fp \* fn)) / sqrt(mx), 3)

print('Accuracy :', round(accuracy\*100, 2),'%')

print('Precision :', round(precision\*100, 2),'%')

print('Recall :', round(sensitivity\*100,2), '%')

print('F1 Score :', f1Score)

print('Specificity or True Negative Rate :', round(specificity\*100,2), '%')

print('Balanced Accuracy :', round(balanced\_accuracy\*100, 2),'%')

print('MCC :', MCC)

# Area under ROC curve

from sklearn.metrics import roc\_curve, roc\_auc\_score

print('roc\_auc\_score:', round(roc\_auc\_score(actual, predicted), 3))

# ROC Curve

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

logit\_roc\_auc = roc\_auc\_score(actual, predicted)

fpr, tpr, thresholds = roc\_curve(actual, models.predict\_proba(x\_test)[:,1])

plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit\_roc\_auc)

plt.plot(fpr, tpr, label= 'Classification Model' % logit\_roc\_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

plt.savefig('Log\_ROC')

plt.show()

print('-----------------------------------------------------------------------------------------------------

') #---------------------------------------------------------------------------------------------------------

new\_row = {'Model Name' : models,

'True\_Positive' : tp,

‘False\_Negative' : fn,

'False\_Positive' : fp,

'True\_Negative' : tn,

'Accuracy' : accuracy,

'Precision' : precision,

'Recall' : sensitivity,

'F1 Score' : f1Score,

'Specificity' : specificity,

'MCC':MCC,

'ROC\_AUC\_Score':roc\_auc\_score(actual, predicted),

'Balanced Accuracy':balanced\_accuracy}

EMResults = EMResults.append(new\_row, ignore\_index=True)

#---------------------------------------------------------------------------------------------------------

**KNN Classifier:**

# Build KNN Model

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

import sklearn.metrics as metrics

from sklearn.metrics import roc\_curve, roc\_auc\_score

accuracy = []

for a in range(1, 21, 1):

k = a

# Build the model

ModelKNN = KNeighborsClassifier(n\_neighbors=k)

# Train the model

ModelKNN.fit(x\_train, y\_train)

# Predict the model

y\_pred = ModelKNN.predict(x\_test)

y\_pred\_prob = ModelKNN.predict\_proba(x\_test)

print('KNN\_K\_value = ', a)

# Print the model name

print('Model Name: ', ModelKNN)

# confusion matrix in sklearn

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

# actual values

actual = y\_test

# predicted values

predicted = y\_pred

# confusion matrix

matrix = confusion\_matrix(actual,predicted, labels=[1,0],sample\_weight=None, normalize=None)

print('Confusion matrix : \n', matrix)

# outcome values order in sklearn

tp, fn, fp, tn = confusion\_matrix(actual,predicted,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy

C\_Report = classification\_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C\_Report)

# calculating the metrics

sensitivity = round(tp/(tp+fn), 3);

specificity = round(tn/(tn+fp), 3);

accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);

balanced\_accuracy = round((sensitivity+specificity)/2, 3);

precision = round(tp/(tp+fp), 3);

f1Score = round((2\*tp/(2\*tp + fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.

# A model with a score of +1 is a perfect model and -1 is a poor model

from math import sqrt

mx = (tp+fp) \* (tp+fn) \* (tn+fp) \* (tn+fn)

MCC = round(((tp \* tn) - (fp \* fn)) / sqrt(mx), 3)

print('Accuracy :', round(accuracy\*100, 2),'%')

print('Precision :', round(precision\*100, 2),'%')

print('Recall :', round(sensitivity\*100,2), '%')

print('F1 Score :', f1Score)

print('Specificity or True Negative Rate :', round(specificity\*100,2), '%' )

print('Balanced Accuracy :', round(balanced\_accuracy\*100, 2),'%')

print('MCC :', MCC)

# Area under ROC curve

from sklearn.metrics import roc\_curve, roc\_auc\_score

print('roc\_auc\_score:', round(roc\_auc\_score(actual, predicted), 3))

# ROC Curve

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

model\_roc\_auc = roc\_auc\_score(actual, predicted)

fpr, tpr, thresholds = roc\_curve(actual, ModelKNN.predict\_proba(x\_test)[:,1])

plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit\_roc\_auc)

plt.plot(fpr, tpr, label= 'Classification Model' % model\_roc\_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

#plt.savefig('Log\_ROC')

plt.show()

#------------------------------------------------------------------------------

new\_row = {'Model Name' : ModelKNN,

'KNN K Value' : a,

'True\_Positive' : tp,

'False\_Negative' : fn,

'False\_Positive' : fp,

'True\_Negative' : tn,

'Accuracy' : accuracy,

'Precision' : precision,

'Recall' : sensitivity,

'F1 Score' : f1Score,

'Specificity' : specificity,

'MCC':MCC,

'ROC\_AUC\_Score':roc\_auc\_score(actual, predicted),

'Balanced Accuracy':balanced\_accuracy}

KNN\_Results = KNN\_Results.append(new\_row, ignore\_index=True)

#------KNN\_Results------------------------------------------------------------------------

**SVM Algorithm:**

# Training the SVM algorithm

from sklearn.svm import SVC

ModelSVMGaussian = SVC(kernel='rbf', random\_state = 42, class\_weight='balanced', probability=True)

# Train the model

ModelSVMGaussian.fit(x\_train, y\_train)

# Predict the model with test data set

y\_pred = ModelSVMGaussian.predict(x\_test)

y\_pred\_prob = ModelSVMGaussian.predict\_proba(x\_test)

# Confusion matrix in sklearn

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

# Print the model name

print('Model Name: ', "SVM - Gaussian")

# actual values

actual = y\_test

# predicted values

predicted = y\_pred

# confusion matrix

matrix = confusion\_matrix(actual,predicted, labels=[1,0],sample\_weight=None, normalize=None)

print('Confusion matrix : \n', matrix)

# outcome values order in sklearn

tp, fn, fp, tn = confusion\_matrix(actual,predicted,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy

C\_Report = classification\_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C\_Report)

# calculating the metrics

sensitivity = round(tp/(tp+fn), 3);

specificity = round(tn/(tn+fp), 3);

accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);

balanced\_accuracy = round((sensitivity+specificity)/2, 3);

precision = round(tp/(tp+fp), 3);

f1Score = round((2\*tp/(2\*tp + fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.

# A model with a score of +1 is a perfect model and -1 is a poor model

from math import sqrt

mx = (tp+fp) \* (tp+fn) \* (tn+fp) \* (tn+fn)

MCC = round(((tp \* tn) - (fp \* fn)) / sqrt(mx), 3)

print('Accuracy :', round(accuracy\*100, 2),'%')

print('Precision :', round(precision\*100, 2),'%')

print('Recall :', round(sensitivity\*100,2), '%')

print('F1 Score :', f1Score)

print('Specificity or True Negative Rate :', round(specificity\*100,2), '%' )

print('Balanced Accuracy :', round(balanced\_accuracy\*100, 2),'%')

print('MCC :', MCC)

# Area under ROC curve

from sklearn.metrics import roc\_curve, roc\_auc\_score

print('roc\_auc\_score:', round(roc\_auc\_score(y\_test, y\_pred), 3))

# ROC Curve

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

logit\_roc\_auc = roc\_auc\_score(y\_test, y\_pred)

fpr, tpr, thresholds = roc\_curve(y\_test,ModelSVMGaussian.predict\_proba(x\_test)[:,1])

plt.figure()

# plt.plot

plt.plot(fpr, tpr, label= 'Classification Model' % logit\_roc\_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

plt.savefig('Log\_ROC')

plt.show()

print('-----------------------------------------------------------------------------------------------------')

#---

new\_row = {'Model Name' : "SVM - Gaussian",

'True\_Positive' : tp,

'False\_Negative' : fn,

'False\_Positive' : fp,

'True\_Negative' : tn,

'Accuracy' : accuracy,

'Precision' : precision,

'Recall' : sensitivity,

'F1 Score' : f1Score,

'Specificity' : specificity,

'MCC':MCC,

'ROC\_AUC\_Score':roc\_auc\_score(actual, predicted),

'Balanced Accuracy':balanced\_accuracy}

EMResults = EMResults.append(new\_row, ignore\_index=True)

#---------------------------------------------------------------------------------------------------------------

**Comparision:**

# Train the model training dataset

models.fit(x\_train, y\_train)

# Prediction the model with test dataset

y\_pred = models.predict(x\_test)

y\_pred\_prob = models.predict\_proba(x\_test)

# Print the model name

print('Model Name: ', models)

# confusion matrix in sklearn

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

# actual values

actual = y\_test

# predicted values

predicted = y\_pred

# confusion matrix

matrix = confusion\_matrix(actual,predicted, labels=[1,0],sample\_weight=None, normalize=None)

print('Confusion matrix : \n', matrix)

# outcome values order in sklearn

tp, fn, fp, tn = confusion\_matrix(actual,predicted,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp, fn, fp, tn)

# classification report for precision, recall f1-score and accuracy

C\_Report = classification\_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C\_Report)

# calculating the metrics

sensitivity = round(tp/(tp+fn), 3);

specificity = round(tn/(tn+fp), 3);

accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);

balanced\_accuracy = round((sensitivity+specificity)/2, 3);

precision = round(tp/(tp+fp), 3);

f1Score = round((2\*tp/(2\*tp + fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1. # A model with a score of +1 is a perfect model and -1 is a poor model

from math import sqrt

mx = (tp+fp) \* (tp+fn) \* (tn+fp) \* (tn+fn)

MCC = round(((tp \* tn) - (fp \* fn)) / sqrt(mx), 3)

print('Accuracy :', round(accuracy\*100, 2),'%')

print('Precision :', round(precision\*100, 2),'%')

print('Recall :', round(sensitivity\*100,2), '%')

print('F1 Score :', f1Score)

print('Specificity or True Negative Rate :', round(specificity\*100,2), '%' )

print('Balanced Accuracy :', round(balanced\_accuracy\*100, 2),'%')

print('MCC :', MCC)

# Area under ROC curve

from sklearn.metrics import roc\_curve, roc\_auc\_score

print('roc\_auc\_score:', round(roc\_auc\_score(actual, predicted), 3))

# ROC Curve

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

Model\_roc\_auc = roc\_auc\_score(actual, predicted)

fpr, tpr, thresholds = roc\_curve(actual, models.predict\_proba(x\_test)[:,1])

plt.figure()

#

plt.plot(fpr, tpr, label= 'Classification Model' % Model\_roc\_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

plt.savefig('Log\_ROC')

plt.show()

print('-----------------------------------------------------------------------------------------------------')

#----------------------------------------------------------------------------------------------------------

new\_row = {'Model Name' : models,

'True\_Positive': tp,

'False\_Negative': fn,

'False\_Positive': fp,

'True\_Negative': tn,

'Accuracy' : accuracy,

'Precision' : precision,

'Recall' : sensitivity,

'F1 Score' : f1Score,

'Specificity' : specificity,

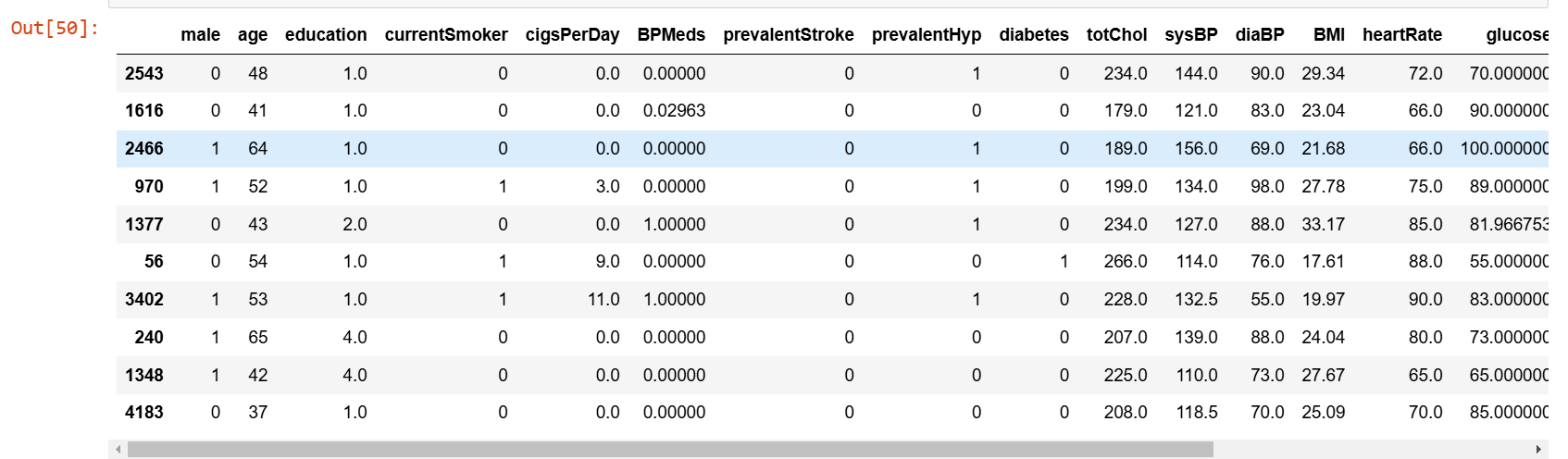
'MCC':MCC,

'ROC\_AUC\_Score':roc\_auc\_score(actual, predicted),

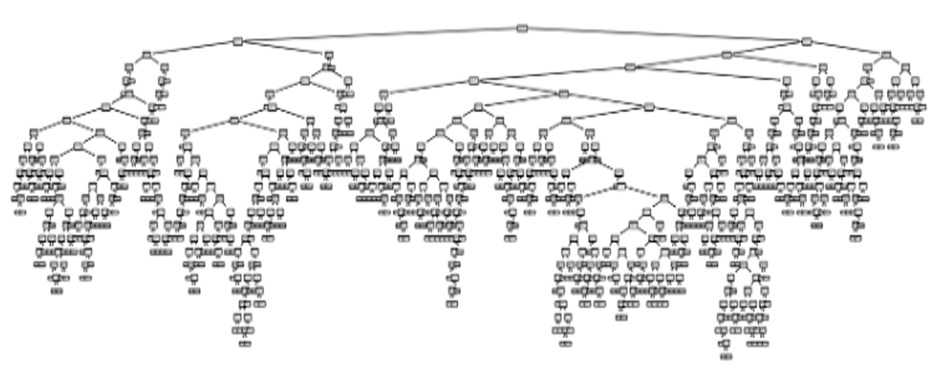
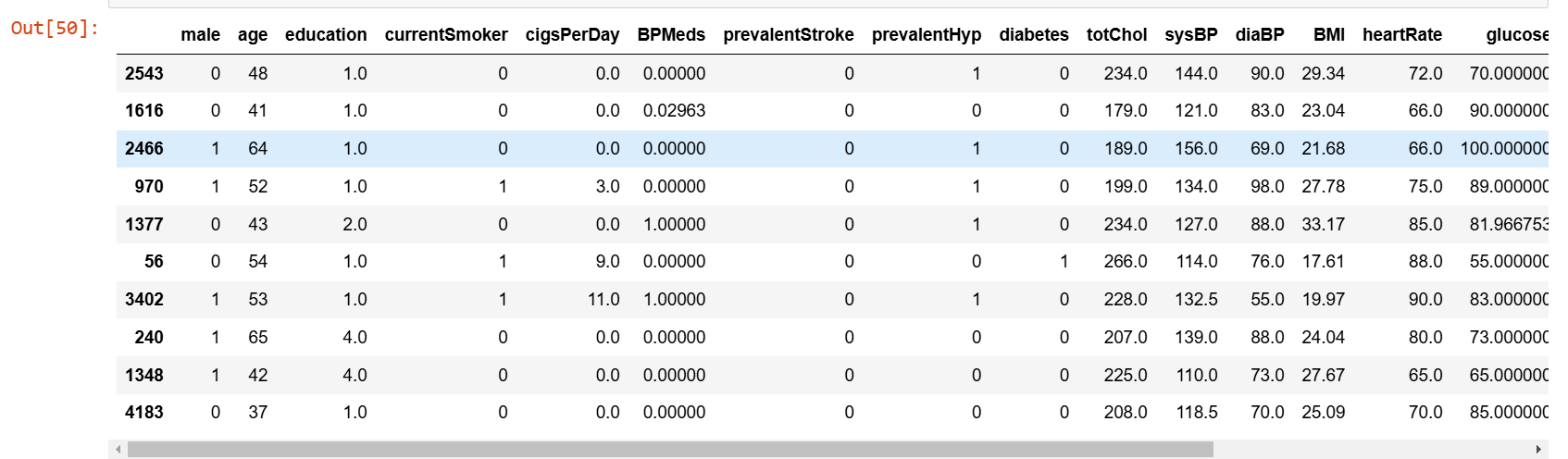
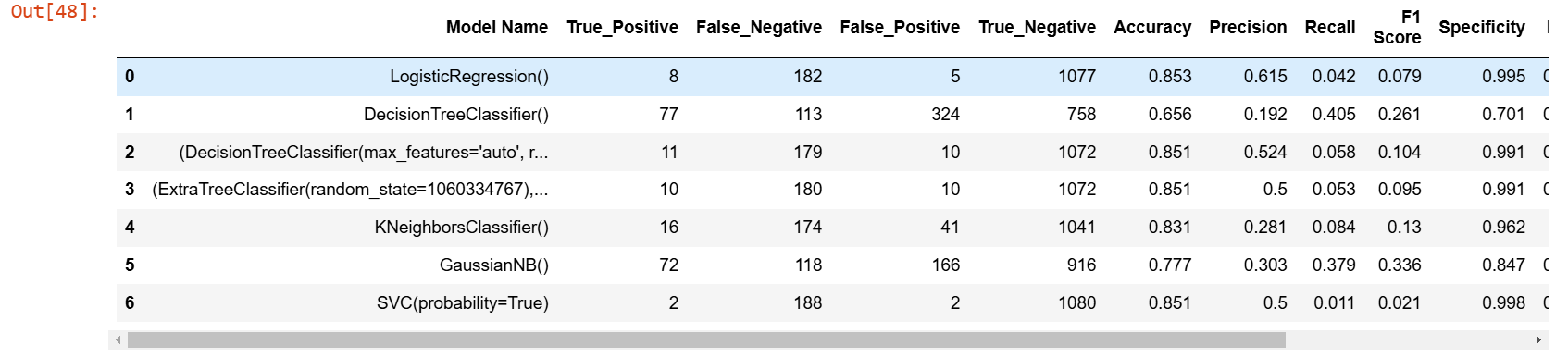
'Balanced Accuracy':balanced\_accuracy}

EMResults = EMResults.append(new\_row, ignore\_index=True)

**6.2 RESULTS**



# **7.OUTPUT SCREENS**



**8. Conclusions**

The model results in the following order by considering the model

accuracy, F1 score and RoC AUC score.

• KNeighbours Classifier

• Extra Trees Classifier

• Decision tree classifier

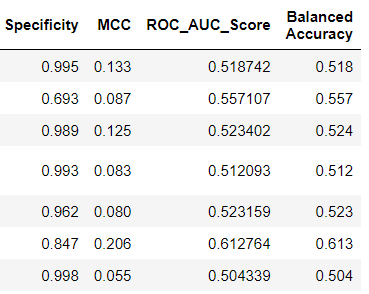
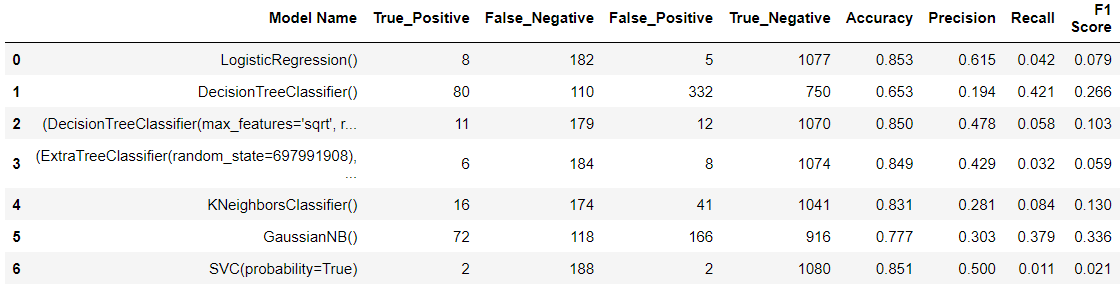
• LogisticRegression

We recommend model – Logistic Regression algorithm as a best fit for

the given dataset because it uses bootstrap aggregation which can

reduce bias and variance in the data and can leads to good predictions

with heart disease dataset.



▪ After evaluating various algorithms, we have found that Logistic

Regression algorithm achieves the highest accuracy among them.

▪ Therefore, we will utilize the Logistic Regression algorithm to predict

Heart diseases in our model.

**9. BIBLOGRAPHY**

▪ <https://www.javatpoint.com/types-of-machine-learning>

▪ <https://www.kaggle.com/>

**Appendices**

• Kernel methods

• Graphical models

• Reinforcement learning

• Convex analysis

• Optimization

• Bio informatics

• Minimal description length principle

• Topics in information theory

• Decision theory

• Network algorithms

• Computational Social Science

• Natural Language Processing Recent trends in deep learning and

representation learning

• Facial recognition

• Voice Recognition

• Virtual Personal Assistant