الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدةد.مولاي الطاهر

كلية العلوم

قسم: الإعلام الآلي

Memory presented for obtaining

Academic Master's degree

Speciality : MICR

Theme

Prediction of cardiovascular diseases using machine learning

Presented by :

ATMANI RANA

Under the supervision of :

Pr. HAMOU REDA MOHAMED

Promotion 2023 - 2024

Thanks

First of all, I would like to address a special mention to Mr. Hamou Reda Mohamed. In their capacity as dissertation tutors, they were able to provide me with many resources, and their availability and their wise advice were able to guide my work and stimulate my thinking.

I would also like to express my deep thanks to the university's teaching team for their collaboration, all the efforts made and the quality of the teaching provided.

Finally, I would like to express all my gratitude to Taytoune Kheira for her unfailing support and without which I would not have been able to continue my journey.

Dedications

I dedicate this modest work:

*To My very dear late Papi

*To my dear Mami

*To my dear parents

*To my taytoune Kheira who contributed to my success

*To my brothers and sisters.

*To all my family.

To all those dear to me.

TABLE OF CONTENTS :

Introduction

Cardiovascular diseases (CVDs) are the major cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Four out of five CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in those under 70 years of age. Many of these deaths could have been prevented by early prediction of the disease. Predicting the heart disease in early stage is an important application of data mining as it can save lives of many patients. There are various risk factors that are associated with heart disease and are said to be its causes. Being in the domain of interest, it is easy to establish the fact that these attributes must be present in heart disease patient dataset, to carry out the deduction for presence of heart disease. Using machine learning techniques we can predict the future occurrence of heart diseases among people. In framing the problem of predicting future heart diseases among people, several factors are of interest. These are the factors that are known to be risk factors of heart diseases and occur in people's life. Coming across the presence of some of these factors in patient's life during a medical examination can lead a health care provider to suggest to the patient that he/she is at risk of having a heart disease. Now if we have patient's past and present medical history data available, then using this data we can predict the future heart diseases in patient's life. This prediction is the deduction, which health care provider had earlier made seeing the presence of risk factors in patient's life. Considering these whole events, the problem of predicting future heart diseases can be viewed as the problem of predicting heart disease in presence of some known attributes, with the ultimate aim of preventing the disease by effective treatments and/or change in lifestyle. Based on these general facts, the prediction of heart diseases based on available attributes is a worthwhile case of applying inductive inference in healthcare.

Background

This project will deliver significant benefits to CVD patients and the wider community by developing an accurate and reliable method of predicting future CVD. The results will also inform clinical decisions and provide insight into the way CVD develops from the presence of risk factors.

Machine learning and data mining have great potential to improve the prediction and diagnosis of diseases. It is often infeasible for a human to identify complex and non-linear patterns in data. Machine learning can exploit data from CVD patients to learn a model that can predict the likelihood of CVD for a patient. This is achieved using historical data to predetermine a patient's prognosis. Machine learning can also be used to discover the relationships and patterns between risk factors and CVD. This can inform clinical decisions by identifying modifiable risk factors and discovering how these risk factors influence the development of CVD.

Monitoring CVDs is crucial for diagnosing the diseases and improving patients' health. However, detection processes are often costly in terms of computation, time, and money. Therefore, a cost-effective, time-efficient, and reliable method of predicting CVD risks is needed to monitor patients more frequently and to a larger extent.

Cardiovascular diseases (CVDs) are increasingly becoming a major cause of morbidity and mortality across the world. According to the World Health Organization (WHO) estimates, CVDs are the leading cause of death, with 17.3 million people dying a year from CVDs, which is 30% of all global deaths.

1.2. Problem Statement

An individual's personal satisfaction and growth is dependent upon their health status, hence it is essential to have proactive healthcare. By considering the factors that cause diseases, it is better to prevent them than to restore health later, as the cost and effects will not be balanced. Cardiovascular disease has become a growing problem and is a global killer that affects people of all ages. Cardiovascular disease is a class of diseases that involve the heart or blood vessels. It is well known for including ischemic heart disease (IHD), cerebrovascular disease, hypertension, and peripheral artery disease. The morbidity rate and high treatment costs make this disease a serious threat. In 2007, a study conducted by 36 countries showed that IHD was the highest burden disease, accounting for 10.4% of total burden diseases, followed by stroke at 7.2%. A methods of predicting a higher risk of cardiovascular disease is by using supervised and unsupervised learning, which can separate individuals into higher risk and lower risk groups. This methods are also useful for planning better treatments or interventions to reduce the risk of developing these diseases. This project will focus on constructing a decision support system for predicting cardiovascular disease risk using a supervised and unsupervised learning methods.

1.3. Objective

this project for the prediction of cardiovascular diseases using machine learning because cardiovascular diseases are becoming a major cause of death throughout the world. Early and accurate prediction of cardiovascular diseases can aid in making decisions for lifestyle changes in high-risk patients and can prevent the occurrence of cardiovascular diseases. So, through this project, we are aiming to predict the possibility of cardiovascular diseases in a patient. This can be used by hospitals to identify patients having a high risk of cardiovascular diseases. The healthcare can then be provided to these patients so that the cardiovascular diseases can be prevented at an early stage. Our aim is to build a model with high accuracy and desired performance so that the cardiovascular diseases can be predicted at an early stage. We have healthcare data along with the cardiovascular diseases dataset. This data contains various information about the patients. We will use this data to train our model. Our project will go through the following phases: 1. Data preprocessing: This includes the cleaning of data, handling the missing values in the data, and normalizing the data so that the data can be used for the training of the model. 2. Data Analysis: This phase involves the analysis of data to identify patterns and relationships between variables. 3. Training the model: This will include the application of machine learning algorithms to the training data. 4. Testing the model: This will allow us to evaluate the performance of various machine learning algorithms on the same data. The algorithm will be trained and tested multiple times using different techniques, and performance will be compared after applying each technique. This will help us find the best technique that can be used on the new patient data to predict the diseases.

Chapter 01 :

Cardiovascular illness

1.1 Introduction

Cardiovascular disease (CVD), which is the leading cause of death globally, has become a significant problem in public health all over the world. As a result, patients, their families, and the governments of these countries have incurred substantial socioeconomic expenses. Patients at high risk for CVD can be identified by prediction models that use risk stratification. After that, measures that are tailored to this group, such as dietary changes and the use of statins, can help reduce that risk and contribute to the primary prevention of CVD.

1.2 What is cardiovascular disease?

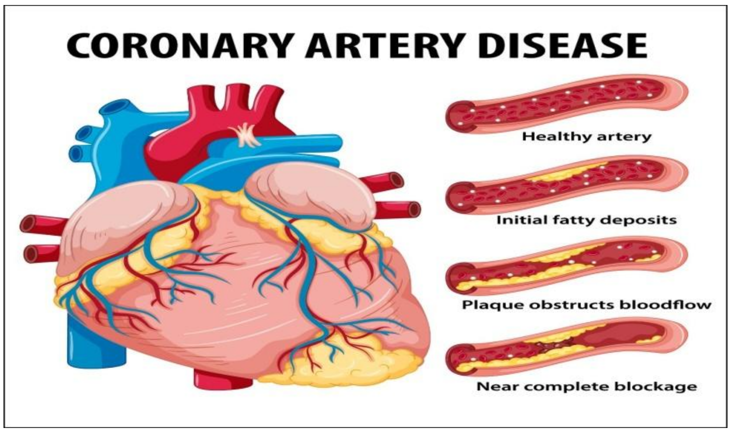
Cardiovascular disease is a group of diseases affecting your heart and blood vessels. These diseases can affect one or many parts of your heart and/or blood vessels. A person may be symptomatic (physically experiencing the disease) or asymptomatic (not feeling anything at all).
Cardiovascular disease includes heart or blood vessel issues, including:

- Narrowing of the blood vessels in your heart, other organs or throughout your body.
- Heart and blood vessel problems present at birth.
- Heart valves that aren't working right.
- Irregular heart rhythms.[1]

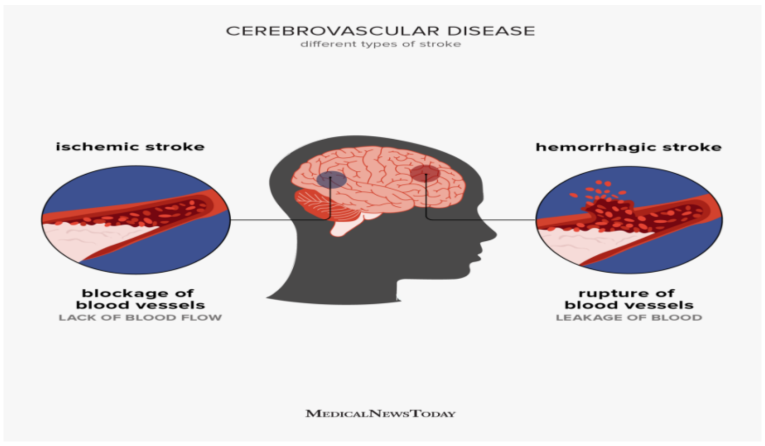
1.3 Type cardiovascular diseases

Cardiovascular diseases (CVDs) are a group of disorders of the heart and blood vessels. They include:

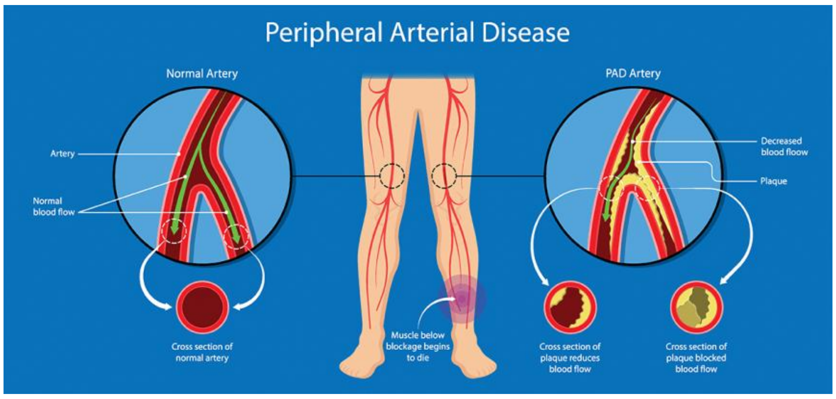
- coronary heart disease – a disease of the blood vessels supplying the heart muscle.[19]



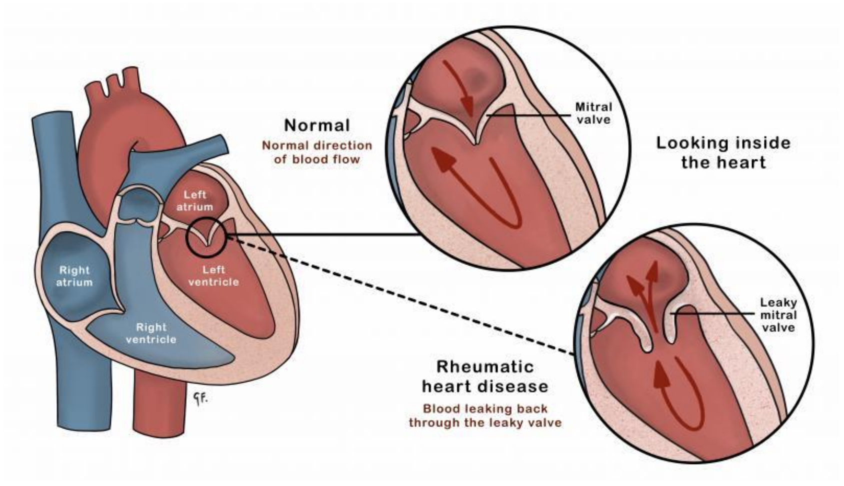
- Cerebrovascular disease – a disease of the blood vessels supplying the brain.[20]



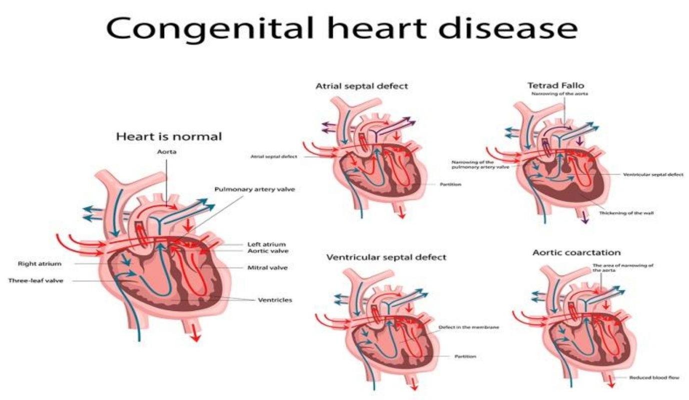
- peripheral arterial disease – a disease of blood vessels supplying the arms and legs.[21]


Peripheral Arterial Disease
Normal Artery
Artery
Normal blood flow
Cross section of normal artery
Muscle below blockage begins to die
PAD Artery
Decreased blood floow
Plaque
Cross section of plaque reduces blood flow
Cross section of plaque blocked blood flow

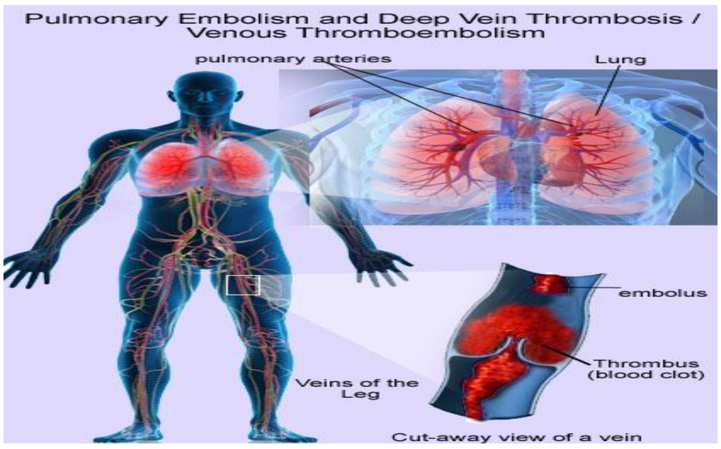
- rheumatic heart disease – damage to the heart muscle and heart valves from rheumatic fever, caused by streptococcal bacteria.[22]



- congenital heart disease – birth defects that affect the normal development and functioning of the heart caused by malformations of the heart structure from birth.[23]



- Deep vein thrombosis and pulmonary embolism – blood clots in the leg veins, which can dislodge and move to the heart and lungs.[24]



Heart attacks and strokes are usually acute events and are mainly caused by a blockage that prevents blood from flowing to the heart or brain. The most common reason for this is a build-up of fatty deposits on the inner walls of the blood vessels that supply the heart or brain. Strokes can be caused by bleeding from a blood vessel in the brain or from blood clots.[2]

1.4 Symptoms

1.4.1 Symptoms of heart issues:

Chest pain (angina).

Chest pressure, heaviness or discomfort, sometimes described as a "belt around the chest" or a "weight on the chest."

Shortness of breath (dyspnea).

Dizziness or fainting.

Fatigue or exhaustion.

1.4.2 Symptoms of blockages in blood vessels throughout your body:

Pain or cramps in your legs when you walk.

Leg sores that aren't healing.

Cool or red skin on your legs.

Swelling in your legs.

Numbness in your face or a limb. This may be on only one side of your body.

Difficulty with talking, seeing or walking.[1]

1.5 Cause and risk factor :

What causes cardiovascular disease?

The causes of cardiovascular disease can vary depending on the specific type. For example, atherosclerosis (plaque buildup in your arteries) causes coronary artery disease and peripheral artery disease. Coronary artery disease, scarring of your heart muscle, genetic problems or medications can cause arrhythmias. Aging, infections and rheumatic disease can cause valve diseases.

What are cardiovascular disease risk factors?
You may be more likely to develop cardiovascular disease if you have risk factors such as:

- ✓ High blood pressure (hypertension).
- ✓ High cholesterol (hyperlipidemia).
- ✓ Tobacco use (including vaping).
- ✓ Type 2 diabetes.
- ✓ Family history of heart disease.
- ✓ Lack of physical activity.
- ✓ Having excess weight or obesity.
- ✓ Diet high in sodium, sugar and fat.
- ✓ Overuse of alcohol.
- ✓ Misuse of prescription or recreational drugs.
- ✓ Preeclampsia or toxemia.
- ✓ Gestational diabetes.
- ✓ Chronic inflammatory or autoimmune conditions.
- ✓ Chronic kidney disease.[3]

1.6 Diagnosis and Tests :

- ✓ How is cardiovascular disease diagnosed?

Your healthcare provider will perform a physical exam and ask questions about your symptoms, personal health and family health history. They may also order tests to help diagnose cardiovascular disease.

- ✓ What tests might I have for cardiovascular disease?

Some common tests to diagnose cardiovascular disease include:

Blood work measures substances that indicate cardiovascular health, such as cholesterol, blood sugar levels and specific proteins. A provider can use a blood test to check for blood clotting issues as well.

Ankle brachial index (ABI) compares the blood pressure in your ankles and arms to diagnose peripheral artery disease.

Electrocardiogram (EKG) records your heart's electrical activity.

Ambulatory monitoring uses wearable devices that track your heart rhythm and rates.
Echocardiogram uses sound waves to create an image of your heartbeat and blood flow.

Ultrasound uses sound waves to check blood flow in your legs or neck.

Cardiac computerized tomography (CT) uses X-rays and computer processing to create 3D images of your heart and blood vessels.

Cardiac magnetic resonance imaging (MRI) uses magnets and radio waves to create highly detailed images of your heart.
MR angiogram or CT angiogram uses an MRI or CT, respectively, to see blood vessels in your legs, head and neck.

Stress tests analyze how physical activity affects your heart in a controlled setting, using exercise or medications, to determine how your heart responds. This type of test can involve EKGs and/or imaging tests.

Cardiac catheterization uses a catheter (thin, hollow tube) to measure pressure and blood flow in your heart.[3]

1.7 Conclusion:

In conclusion, this chapter has provided a comprehensive overview of cardiovascular diseases (CVDs), encompassing their types, prevalence, risk factors, and management strategies. CVDs represent a significant global health burden, contributing to a substantial proportion of morbidity and mortality worldwide.
Moreover, the chapter has delved into the diagnostic modalities and treatment options available for individuals with CVDs.

Chapter 02 :
Approaches used

2.1 Introduction :

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to learn (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. Machine learning is the design and study of software artifacts that use past experience to make future decisions. It is the study of programs that learn from data. The fundamental goal of machine learning is to generalize, or to induce an unknown rule from examples of the rule's application. Machine learning systems are often described as learning from experience either with or without supervision from humans.

2.2 Artificiel intelligent :

2.2.1 bDéfinition :

The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—such as discovering proofs for mathematical theorems or playing chess—with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match full human flexibility over wider domains or in tasks requiring much everyday knowledge.

On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, voice or handwriting recognition, and chatbots.

(Read Ray Kurzweil's Britannica essay on the future of "Nonbiological Man.")

2.2.2 History of artificial intelligence (AI) :

Began in antiquity, with myths, stories and rumors of artificial beings endowed with intelligence or consciousness by master craftsmen. The seeds of modern AI were planted by philosophers who attempted to describe the process of human thinking as the mechanical manipulation of symbols. This work culminated in the invention of the programmable digital computer in the 1940s, a machine based on the abstract essence of mathematical reasoning. This device and the ideas behind it inspired a handful of scientists to begin seriously discussing the possibility of building an electronic brain.

The field of AI research was founded at a workshop held on the campus of Dartmouth College, USA during the summer of 1956. Those who attended would become the leaders of AI research for decades. Many of them predicted that a machine as intelligent as a human being would exist in no more than a generation, and they were given millions of dollars to make this vision come true.

Eventually, it became obvious that researchers had grossly underestimated the difficulty of the project. In 1974, in response to the criticism from James Lighthill and ongoing pressure from congress, the U.S. and British Governments stopped funding undirected research into artificial intelligence, and the difficult years that followed would later be known as an "AI winter". Seven years later, a visionary initiative by the Japanese Government inspired governments and industry to provide AI with billions of dollars, but by the late 1980s the investors became disillusioned and withdrew funding again.

Investment and interest in AI boomed in the 2020s when machine learning was successfully applied to many problems in academia and industry due to new methods, the application of powerful computer hardware, and the collection of immense data sets.

2.3 Static learning:

2.3.1 Type of learning :

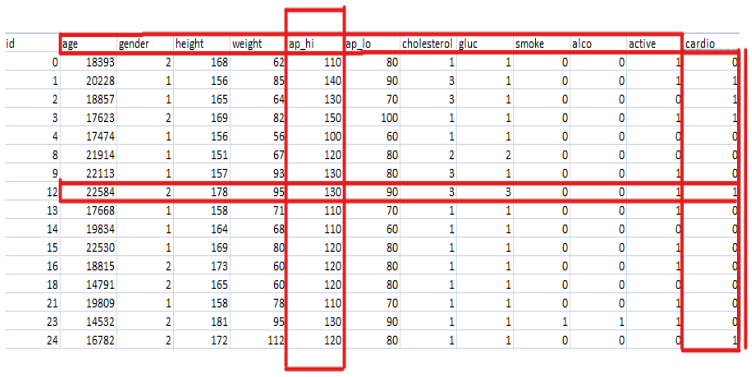
The term ‘Supervise’ means to *observe* and *direct* the execution of a task or an activity. So here we are supervising the machine to do the tasks. Let’s have a look at these supervised and unsupervised learning.

2.3.1.1 Supervised learning:

Supervised learning is a category of machine learning that uses labeled datasets to train algorithms to predict outcomes and recognize patterns.

Supervised machine learning algorithms make it easier for organizations to create complex models that can make accurate predictions. As a result, they are widely used across various industries and fields, including healthcare, marketing, financial services, and more.

Here we teach the model by loading the model with knowledge i.e by training it with some data from a labeled dataset. Consider the following example of cancer dataset :



id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	18393	2	168	62	110	80	1	1	0	0	1	0
1	20228	1	156	85	140	90	3	1	0	0	1	1
2	18857	1	165	64	130	70	3	1	0	0	0	1
3	17623	2	169	82	150	100	1	1	0	0	1	1
4	17474	1	156	56	100	60	1	1	0	0	0	0
8	21914	1	151	67	120	80	2	2	0	0	0	0
9	22113	1	157	93	130	80	3	1	0	0	1	0
12	22584	2	178	95	130	90	3	3	0	0	1	1
13	17668	1	158	71	110	70	1	1	0	0	1	0
14	19834	1	164	68	110	60	1	1	0	0	0	0
15	22530	1	169	80	120	80	1	1	0	0	1	0
16	18815	2	173	60	120	80	1	1	0	0	1	0
18	14791	2	165	60	120	80	1	1	0	0	0	0
21	19809	1	158	78	110	70	1	1	0	0	1	0
23	14532	2	181	95	130	90	1	1	1	1	1	0
24	16782	2	172	112	120	80	1	1	0	0	0	1

Labelled dataset

The columns are called features which include the data. And there are two kinds of data: The first is numerical.When dealing with machine learning, the most commonly used data is numeric.The second is categorical,that is its non-numeric because it contains characters rather than numbers.In this case, it’s categorical because this dataset is made for classification.

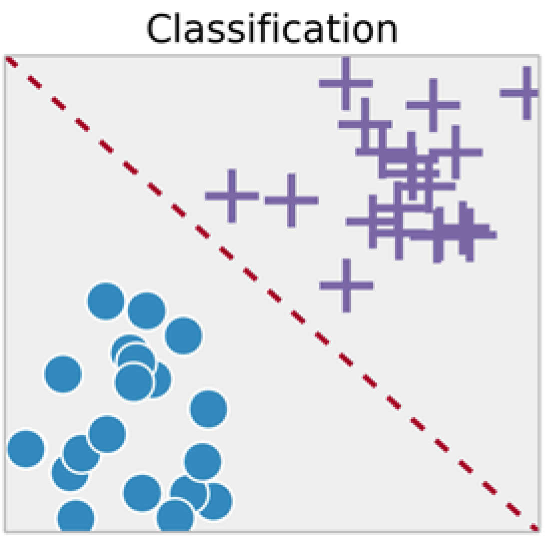
- **Types of supervised learning :**

Supervised learning in machine learning is generally divided into two categories: classification and regression.

- Classification :

Classification algorithms are used to group data by predicting a categorical label or output variable based on the input data. Classification is used when output variables are categorical, meaning there are two or more classes.

One of the most common examples of classification algorithms in use is the spam filter in your email inbox. Here, a supervised learning model is trained to predict whether an email is spam or not with a dataset that contains labeled examples of both spam and legitimate emails. The algorithm extracts information about each email, including the sender, the subject line, body copy, and more. It then uses these features and corresponding output labels to learn patterns and assign a score that indicates whether an email is real or spam.

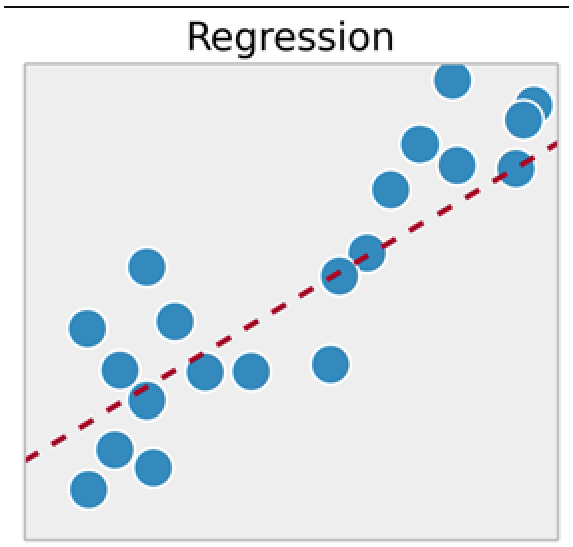


In classification, you will n eed to categorize data into predefined classes.

- Regression :

Regression algorithms are used to predict a real or continuous value, where the algorithm detects a relationship between two or more variables.

A common example of a regression task might be predicting a salary based on work experience. For instance, a supervised learning algorithm would be fed inputs related to work experience (e.g., length of time, the industry or field, location, etc.) and the corresponding assigned salary amount. After the model is trained, it could be used to predict the average salary based on work experience.



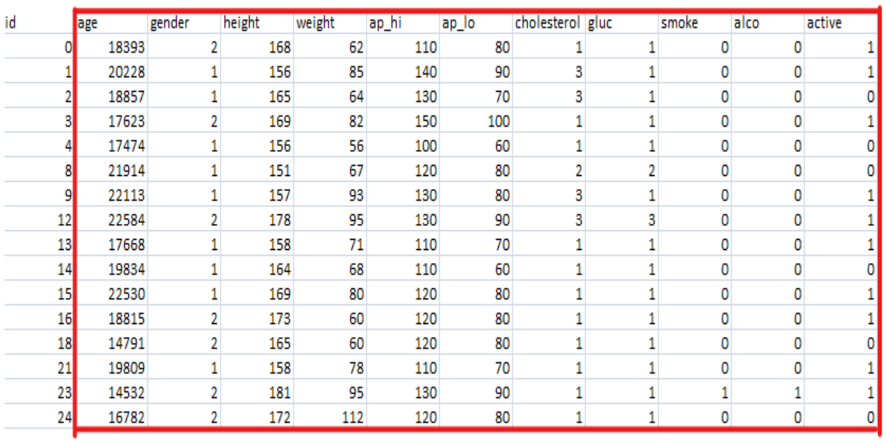
Regression techniques are used when the output is real-valued based on continuous variables. For example, any time series data. This technique involves fitting a line.

2.3.2 Unsupervised learning:

A model is prepared by deducing structures present in the input data. This may be to extract general rules. It may be through a mathematical process to systematically reduce redundancy, or it may be to organize data by similarity. A program does not learn from labeled data. Instead, it attempts to discover patterns in the data.

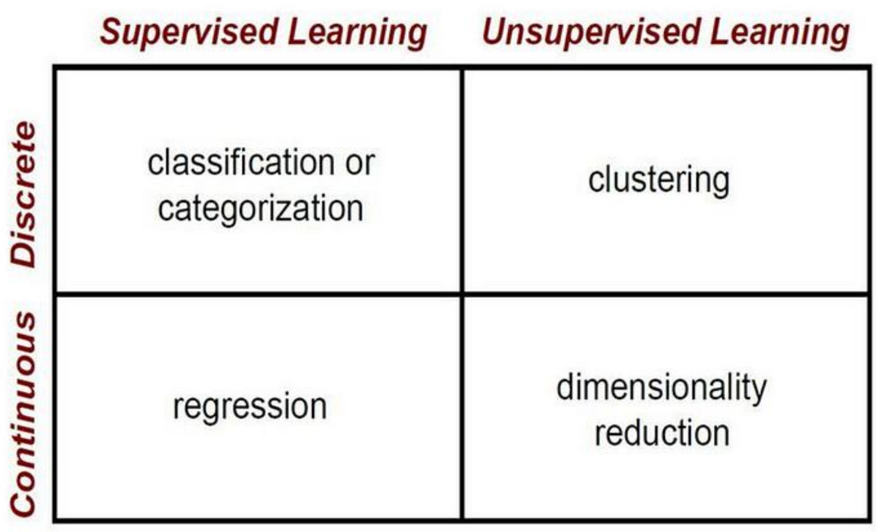
1. Input data is not labeled and does not have a known result.
2. Example problems are clustering, dimensionality reduction and association rule learning.
3. Example algorithms include: the Apriori algorithm and k-Means.

Here we let the model work on its own to discover information i.e the unsupervised algorithm trains on the dataset and draws conclusions on *unlabeled data*. Dimension reduction, density estimation,market basket analysis, and clustering are the most widely used unsupervised machine learning techniques.



id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	18393	2	168	62	110	80	1	1	0	0	1
1	20228	1	156	85	140	90	3	1	0	0	1
2	18857	1	165	64	130	70	3	1	0	0	0
3	17623	2	169	82	150	100	1	1	0	0	1
4	17474	1	156	56	100	60	1	1	0	0	0
8	21914	1	151	67	120	80	2	2	0	0	0
9	22113	1	157	93	130	80	3	1	0	0	1
12	22584	2	178	95	130	90	3	3	0	0	1
13	17668	1	158	71	110	70	1	1	0	0	1
14	19834	1	164	68	110	60	1	1	0	0	0
15	22530	1	169	80	120	80	1	1	0	0	1
16	18815	2	173	60	120	80	1	1	0	0	1
18	14791	2	165	60	120	80	1	1	0	0	0
21	19809	1	158	78	110	70	1	1	0	0	1
23	14532	2	181	95	130	90	1	1	1	1	1
24	16782	2	172	112	120	80	1	1	0	0	0

Supervised and Unsupervised Learning



So, the biggest difference between supervised and unsupervised learning is that supervised learning deals with labeled data while unsupervised learning deals with unlabeled data. In supervised learning, we have machine learning algorithms for classification and regression.In unsupervised learning, we have methods such as clustering.

In comparison to supervised learning, unsupervised learning has fewer models and fewer evaluation methods that can be used to ensure that the outcome of the model is accurate.

2.3.3 Semi supervised learning:

There is a desired prediction problem but the model must learn the structures to organize the data as well as make predictions. Semi-supervised learning problems, make use of both supervised and unsupervised data; these problems are located on the spectrum between supervised and unsupervised learning

- Input data is a mixture of labeled and unlabeled examples.
- Example problems are classification and regression.
- Example algorithms are extensions to other flexible methods that make assumptions about how to model the unlabeled data.

2.3.4 Reinforcement learning:

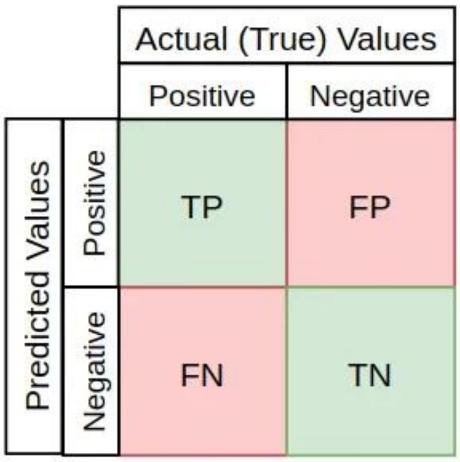
Reinforcement learning is a machine learning training method based on rewarding desired behaviors and punishing undesired ones. In general, a reinforcement learning agent -- the entity being trained -- is able to perceive and interpret its environment, take actions and learn through trial and error.

Reinforcement learning is one of several approaches developers use to train machine learning systems. What makes this approach important is that it empowers an agent, whether it's a feature in a video game or a robot in an industrial setting, to learn to navigate the complexities of the environment it was created for. Over time, through a feedback system that typically includes rewards and punishments, the agent learns from its environment and optimizes its behaviors.

2.4 Confusion matrix :

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions. It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance.

Confusion matrix is sometimes used to illustrate classifier performance based on the above four values (TP, FP, TN, FN). These are plotted against each other to show a confusion matrix:[5]



Confusion Matrix. Image by Author.

The matrix displays the number of instances produced by the model on the test data.
True positives (TP): occur when the model accurately predicts a positive data point.
True negatives (TN): occur when the model accurately predicts a negative data point.
False positives (FP): occur when the model predicts a positive data point incorrectly.
False negatives (FN): occur when the model mispredicts a negative data point.[7]

2.5 Accuracy :

Accuracy is the most straightforward metric. It's the ratio of correctly predicted instances to total instances in the data set. Accuracy is useful for balanced data sets when all classes are equally important.[6]

The base metric used for model evaluation is often *Accuracy*, describing the number of correct predictions over all predictions: [5]

$$\frac{\text{True Positives + True Negatives}}{\text{True Positives + True Negatives + False Positives + False Negatives}} = \frac{\text{N. of Correct Predictions}}{\text{N. of all Predictions}} = \frac{\text{N. of Correct Predictions}}{\text{Size of Dataset}}$$

2.6 Précision :

Precision is a measure of how accurate a model's positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.

Precision is a measure of how many of the positive predictions made are correct (true positives). The formula for it is: [5]

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Predictions you Made}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People you Predicted to have Cancer}}$$

2.7 Recall :

Recall is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data. It is sometimes also referred to as se. The formula for it is: [5]

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Instances in the Dataset}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People with Cancer in the Dataset}}$$

2.8 F1-score :

F1-score is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

F1-Score is a measure combining both precision and recall. It is generally described as the harmonic mean of the two. Harmonic mean is just another way to calculate an "average" of values, generally described as more suitable for ratios (such as precision and recall) than the traditional arithmetic mean. The formula used for F1-score in this case is: [5]

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.9 Spesificity:

Specificity is another important metric in the evaluation of classification models, particularly in binary classification. It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate.

Specificity is a measure of how many negative predictions made are correct (true negatives). The formula for it is: [5]

$$\frac{\text{True Negatives}}{\text{True Negatives + False Positives}} = \frac{\text{N. of Correctly Predicted Negative Instances}}{\text{N. of Total Negative Instances in the Dataset}} = \frac{\text{N. of Correctly Predicted People with no Cancer}}{\text{N. of People with no Cancer in the Dataset}}$$

2.10 PREDICTION ALGORITHMS:

Predictive modeling algorithms are a set of mathematical equations and statistical techniques used to predict an outcome or future behavior based on historical data. These algorithms are used to build predictive models that can forecast future trends, identify patterns in data, and make data-driven decisions. Predictive modeling algorithms are used in a wide range of applications, including finance, healthcare, marketing, and fraud detection.

Some of the popular predictive modeling algorithms have been mentioned above in our examination of problem types and include:

2.10.1 Support Vector machine:

This algorithm is used to classify data into two or more categories by finding the best separating hyperplane. It is a popular algorithm for image classification and text classification.[4]

2.10.2 Decision Tree:

This algorithm uses a tree-like structure to represent decisions and their possible consequences. It is a popular algorithm for classification and regression problems.[4]

2.10.3 Random Forest:

This algorithm is an ensemble model that combines multiple decision trees to improve accuracy and avoid overfitting.[4]

2.10.4 Naive bayes:

This algorithm is based on Bayes' theorem and is used for classification. It assumes that the features are independent of each other and calculates the probability of each class based on the input features.[4]

It is a classification technique based on Bayes' Theorem with an independence assumption among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The Naïve Bayes classifier is a popular supervised machine learning algorithm used for classification tasks such as text classification. It belongs to the family of generative learning algorithms, which means that it models the distribution of inputs for a given class or category. This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately.

In statistics, naive Bayes are simple probabilistic classifiers that apply Bayes' theorem. This theorem is based on the probability of a hypothesis, given the data and some prior knowledge. The naive Bayes classifier assumes that all features in the input data are independent of each other, which is often not true in real-world scenarios. However, despite this simplifying assumption, the naive Bayes classifier is widely used because of its efficiency and good performance in many real-world applications.

Moreover, it is worth noting that naive Bayes classifiers are among the simplest Bayesian network models, yet they can achieve high accuracy levels when coupled with kernel density estimation. This technique involves using a kernel function to estimate the probability density function of the input data, allowing the classifier to improve its performance in complex scenarios where the data distribution is not well-defined. As a result, the naive Bayes classifier is a powerful tool in machine learning, particularly in text classification, spam filtering, and sentiment analysis, among others. Look at the equation below:

$$\underbrace{P(c\mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x\mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c\mid \mathrm{X}) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of the predictor given class.
- P(x) is the prior probability of the predictor.

2.10.5 Kmeans:

K-means is a centroid-based clustering algorithm, where we calculate the distance between each data point and a centroid to assign it to a cluster. The goal is to identify the K number of groups in the dataset.

"K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster." –

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster.

The term 'K' is a number. You need to tell the system how many clusters you need to create. For example, K = 2 refers to two clusters. There is a way of finding out what is the best or optimum value of K for a given data.

For a better understanding of k-means, let's take an example from cricket. Imagine you received data on a lot of cricket players from all over the world, which gives information on the runs scored by the player and the wickets taken by them in the last ten matches. Based on this information, we need to group the data into two clusters, namely batsman and bowlers.

2.10.6 k-nearest neighbors (KNN):

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

This algorithm is used for classification and regression by finding the k-nearest neighbors of a data point. It is a popular algorithm for text classification and image classification.

Step-by-Step explanation of how KNN works is discussed below:

Step 1: Selecting the optimal value of K

K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

In the classification problem, the class labels of are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.

In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector and Y be the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point in X using a distance metric, such as Euclidean distance:

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2]}$$

The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x.

2.10.7 Adaboost :

AdaBoost algorithm, introduced by Freund and Schapire in 1997, revolutionized ensemble modeling. Since its inception, AdaBoost has become a widely adopted technique for addressing binary classification challenges. This powerful algorithm enhances prediction accuracy by transforming a multitude of weak learners into robust, strong learners.

The principle behind boosting algorithms is that we first build a model on the training dataset and then build a second model to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized and the dataset is predicted correctly. Boosting algorithms work in a similar way, it combines multiple models (weak learners) to reach the final output (strong learners).

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. [11]

2.10.8 Logistical regression :

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given data set of independent variables.

This type of statistical model (also known as *logit model*) is often used for classification and

predictive analytics. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas.[8]

Chapter 03 :

Experimentation.

3.1 Introduction :

A Dataset is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. And each row corresponds to a given member of the data set, as per the given question. This is a part of data management. Data sets describe values for each variable for unknown quantities such as height, weight, temperature, volume, etc., of an object or values of random numbers. The values in this set are known as a datum. The data set consists of data of one or more members corresponding to each row. In this article, let us learn the definition of the dataset, different types of datasets, properties, and so on with many solved examples.

A data set is an ordered collection of data. As we know, a collection of information obtained through observations, measurements, study, or analysis is referred to as data. It could include information such as facts, numbers, figures, names, or even basic descriptions of objects. For our study, data can be organized in the form of graphs, charts, or tables. Through data mining, data scientists assist in the analysis of gathered data.

A dataset is a set of numbers or values that pertain to a specific topic. A dataset is, for example, each student’s test scores in a certain class. Datasets can be written as a list of integers in a random order, a table, or with curly brackets around them. The data sets are normally labelled so you understand what the data represents, however, while dealing with data sets, you don’t always know what the data stands for, and you don’t necessarily need to realize what the data represents to accomplish the problem.

3.2 Used tools :

3.2.1 Jupyter:

What is Jupyter Notebook?

Jupyter Notebook (formerly known as IPython Notebook) is an interactive web application for creating and sharing computational documents. The project was first named IPython and later renamed Jupyter in 2014. It is a fully open-source product, and users can use every functionality available for free. It supports more than 40 languages including Python, R, and Scala.

A notebook is a mutable file saved in ipynb format. Jupyter Notebook has a notebook dashboard to help users manage different notebooks. Kernels are also part of Jupyter notebooks. Kernels are processes that run interactive code in a particular programming language and return output to the user. Kernels also respond to tab completion and introspection requests. Jupyter notebooks can be converted to an open standard format such as HTML LaTeX, PDF, Markdown, and Python by using the "Download As" function in the web interface. The conversion process can also be automated via tools like nbconvert. [9]

3.2.2 Python:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[35] Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular programming languages, and has gained widespread use in the machine learning community.[10]

Python is one of the most popular, open source programming language widely adopted by machine learning community. It was designed by Guido van Rossum and was first released in 1991. The reference implementation of Python, i.e. CPython, is managed by Python Software Foundation, which is a nonprofit organization. Python has very strong libraries for advanced mathematical functionalities (NumPy), algorithms and mathematical tools (SciPy) and numerical plotting (matplotlib). Built on these libraries, there is a machine learning library named scikitlearn, which has various classification, regression, and clustering algorithms embedded in it.[25]

3.2.3 Kaggel :

Kaggle is a data science competition platform and online community of data scientists and machine learning practitioners under Google LLC. Kaggle enables users to find and publish datasets, explore and build models in a web-based data science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges

Kaggle was founded by Anthony Goldbloom and Ben Hamner in April 2010.Jeremy Howard, one of the first Kaggle users, joined in November 2010 and served as the President and Chief Scientist.[3] Also on the team was Nicholas Gruen serving as the founding chair.In 2011, the company raised $12.5 million and Max Levchin became the chairman.[5] On 8 March 2017, Fei-Fei Li, Chief Scientist at Google, announced that Google was acquiring Kaggle.

In June 2017, Kaggle surpassed 1 million registered users, and as of October 2023, it has over 15 million users in 194 countries.

In 2022, founders Goldbloom and Hamner stepped down from their positions and D. Sculley became the CEO.

In February 2023, Kaggle introduced Models which allowed users to discover and use pre-trained models through deep integrations with the rest of Kaggle’s platform[14].

3.2.4 Pandas:

pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.[15]

3.2.5 Numpy :

NumPy is a community-driven open source project developed by a diverse group of contributors. The NumPy leadership has made a strong commitment to creating an open, inclusive, and positive community. Please read the NumPy Code of Conduct for guidance on how to interact with others in a way that makes our community thrive.

The NumPy project welcomes your expertise and enthusiasm!

Small improvements or fixes are always appreciated. If you are considering larger contributions to the source code, please contact us through the mailing list first.

Writing code isn’t the only way to contribute to NumPy. You can also:

- review pull requests
- help us stay on top of new and old issues
- develop tutorials, presentations, and other educational materials
- maintain and improve our website
- develop graphic design for our brand assets and promotional materials
- translate website content
- help with outreach and onboard new contributors
- write grant proposals and help with other fundraising efforts[16]

3.2.6 Matplotlib:

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter. Since then it has had an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswel. Matplotlib is a NumFOCUS fiscally sponsored project.

3.2.7 Streamlit :

Streamlit is a free and open-source framework for quickly creating and sharing visually appealing machine learning and data science web apps. It is a Python-based library created primarily for machine learning developers. Data scientists and machine learning engineers are not web developers, and they are not interested in spending weeks learning how to construct web apps using these frameworks. They choose a tool that is simpler to learn and use, as long as it can display data and collect necessary information for modeling. With Streamlit, you can create a visually appealing application with just a few lines of code.[17]

3.3 Datasets:

3.3.1 datasets:

A machine learning dataset is, quite simply, a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes. This means that the data collected should be made uniform and understandable for a machine that doesn't see data the same way as humans do. For this, after collecting the data, it's important to preprocess it by cleaning and completing it, as well as annotate the data by adding meaningful tags readable by a computer.

3.3.2 types of datsets :

In Statistics, we have different types of data sets available for different types of information. They are:

- Numerical data sets
- Bivariate data sets
- Multivariate data sets
- Categorical data sets
- Correlation data sets

3.3.2.1 Numerical Datasets:

The numerical data set is a data set, where the data are expressed in numbers rather than natural language. The numerical data is sometimes called quantitative data. The set of all the quantitative data/numerical data is called the numerical data set. The numerical data is always in the numbers form, such that we can perform arithmetic operations on it.

- Weight and height of a person
- The count of RBC in a medical report
- Number of pages present in a book

3.3.2.2 Bivariate Datasets:

A data set that has two variables is called a Bivariate data set. It deals with the relationship between the two variables. Bivariate dataset usually contains two types of related data.

Example: To find the percentage score and age of the students in a class. Score and age can be considered as two variables

The sales of ice cream versus the temperature on that day. Here the two variables used are ice cream and temperature.

(Note: In case, if you have one set of data alone say, temperature, then it is called the univariate dataset)

3.3.2.3 Multivariate Datasets:

A data set with multiple variables. When the dataset contains three or more than three data types (variables), then the data set is called a multivariate dataset. In other words, the multivariate dataset consists of individual measurements that are acquired as a function of three or more than three variables.

Example: If we have to measure the length, width, height, volume of a rectangular box, we have to use multiple variables to distinguish between those entities.

3.3.2.4 Categorical Datasets:

Categorical data sets represent features or characteristics of a person or an object. The categorical dataset consists of a categorical variable also called the qualitative variable, that can take exactly two values. Hence, it is termed as a dichotomous variable. Categorical data/variables with more than two possible values are called polytomous variables. The qualitative/categorical variables are often assumed to be polytomous variable unless otherwise specified.

Example:

- A person's gender (male or female)
- Marital status (married/unmarried)

3.4 Exemples of Datasets :

3.4.1 Heart Disease Prediction:

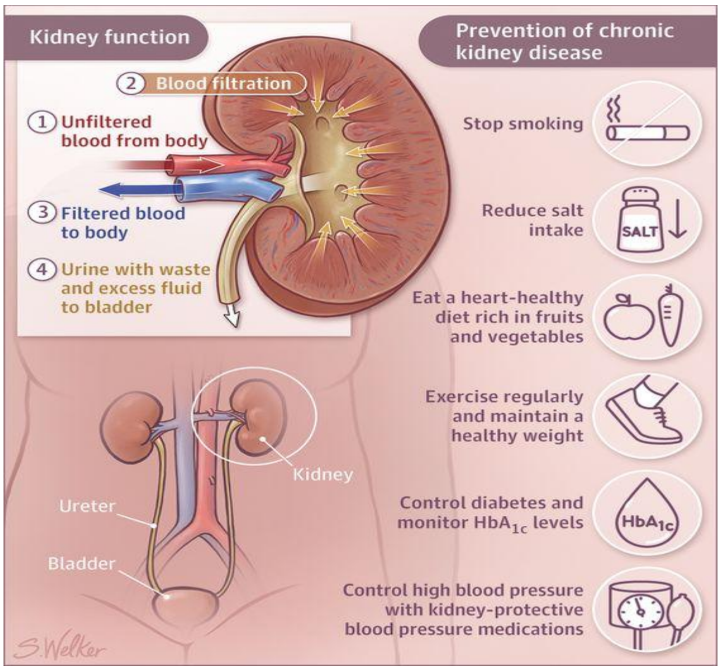
https://www.kaggle.com/datasets/arezaei81/heartcsv

we intend to analyze data related to cardiac features of patients from the "heart.csv" dataset. This dataset provides various information about patients, including age, gender, blood pressure, cholesterol levels, electrocardiographic (ECG) features, and more.
Dataset Information:
This dataset includes the following features:
age: The age of the patient.
sex: Gender of the patient (0: female, 1: male).
cp: Type of chest pain.
trestbps: Resting blood pressure.
chol: Serum cholesterol.
fbs: Fasting blood sugar > 120 mg/dl.
restecg: Resting electrocardiographic results.
thalach: Maximum heart rate achieved.
exang: Exercise induced angina.
oldpeak: ST depression induced by exercise relative to rest

Result :
the SVM model demonstrates a commendable capability in recognizing potential heart patients. With a recall of 0.97 for class 1, it's evident that almost all patients with heart disease are correctly identified. This is of paramount importance in a medical setting.

However, the model's balanced performance ensures that while aiming for high recall, it doesn't compromise on precision, thereby not overburdening the system with unnecessary alerts.[13]

3.4.2 Chronic Kidney Disease Prediction :


Kidney function
2 Blood filtration
1 Unfiltered blood from body
3 Filtered blood to body
4 Urine with waste and excess fluid to bladder
Kidney
Ureter
Bladder
S.Welker
Prevention of chronic kidney disease
Stop smoking
Reduce salt intake
SALT
Eat a heart-healthy diet rich in fruits and vegetables
Exercise regularly and maintain a healthy weight
Control diabetes and monitor HbA1c levels
HbA1c
Control high blood pressure with kidney-protective blood pressure medications

Data Description:
We use the following representation to collect the dataset

1. age - age
2. bp - blood pressure
3. sg - specific gravity
4. al - albumin
5. su - sugar
6. rbc - red blood cells
7. pc - pus cell
8. pcc - pus cell clumps
9. ba - bacteria
10. bgr - blood glucose random
11. bu - blood urea
12. sc - serum creatinine
13. sod - sodium
14. pot - potassium
15. hemo - hemoglobin
16. pcv - packed cell volume
17. wc - white blood cell count
18. rc - red blood cell count

19. htn - hypertension
20. dm - diabetes mellitus
21. cad - coronary artery disease
22. appet - appetite
23. pe - pedal edema
24. ane - anemia
25. class – class[12]

3.4.3 cardiovasculaire disease prediction :(used)

https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

- Data description
- cardio_train.csv (2.94 MB)

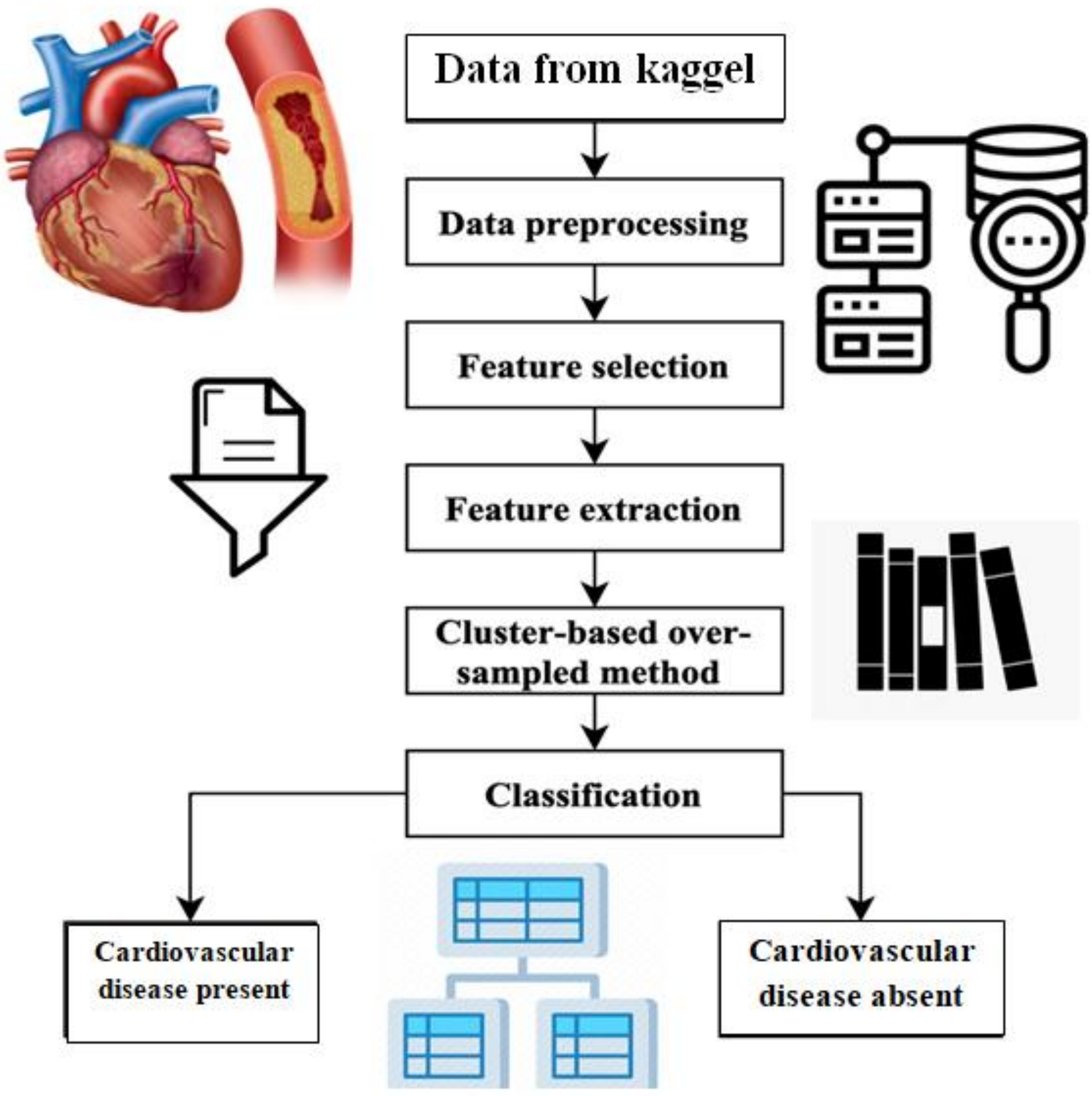
- There are 3 types of input features:
- Objective: factual information;
- Examination: results of medical examination;
- Subjective: information given by the patient.

Features:

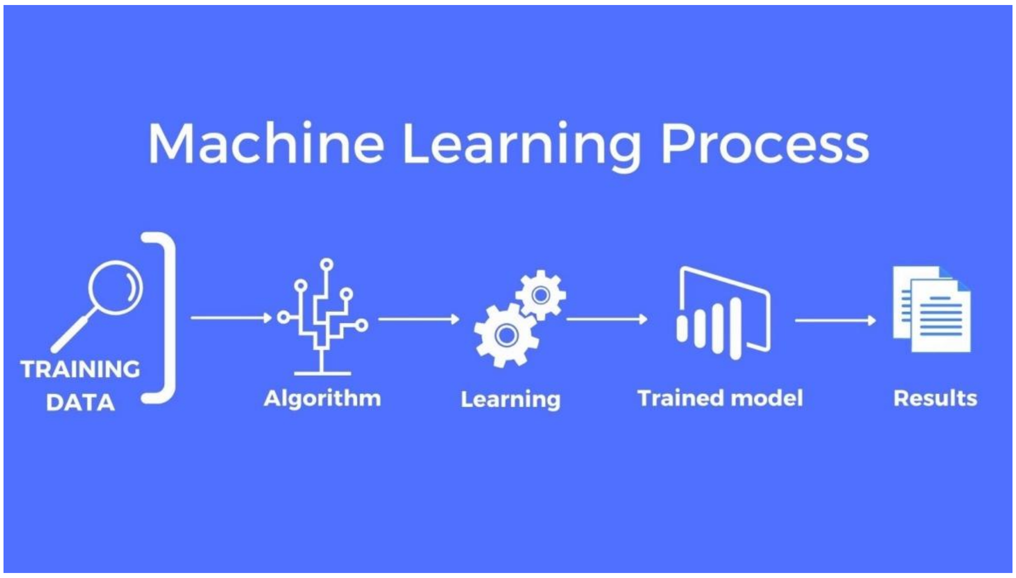
1. Age | Objective Feature | age | int (days)
2. Height | Objective Feature | height | int (cm) |
3. Weight | Objective Feature | weight | float (kg) |
4. Gender | Objective Feature | gender | categorical code |
5. Systolic blood pressure | Examination Feature | ap_hi | int |
6. Diastolic blood pressure | Examination Feature | ap_lo | int |
7. Cholesterol | Examination Feature | cholesterol | 1: normal, 2: above normal, 3: well above normal |
8. Glucose | Examination Feature | gluc | 1: normal, 2: above normal, 3: well above normal |
9. Smoking | Subjective Feature | smoke | binary |
10. Alcohol intake | Subjective Feature | alco | binary |
11. Physical activity | Subjective Feature | active | binary |
12. Presence or absence of cardiovascular disease | Target Variable | cardio | binary |

26.

3.5 Stepes of this project :


Data from kaggel
Data preprocessing
Feature selection
Feature extraction
Cluster-based over-sampled method
Classification
Cardiovascular disease present
Cardiovascular disease absent

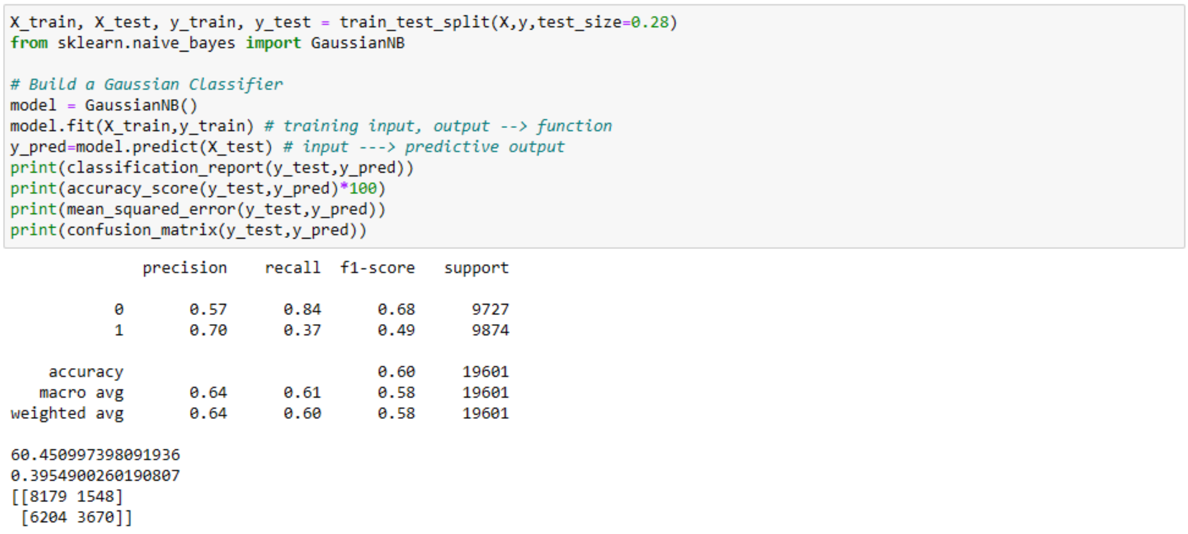
<u>3.6 machine learning process :</u>



- The Experiments below demonstrates this and shows that two separate splits of the data result in the same result:

3.7 Experiment : Naïve_bayes model :

Experiment 1 :

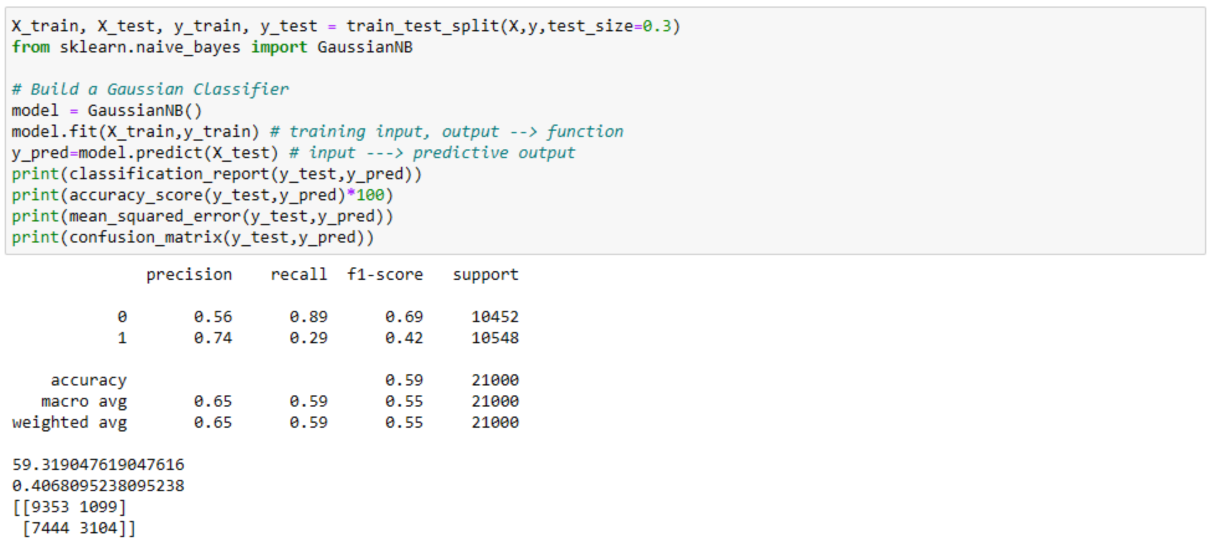


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.28)
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.57      0.84      0.68      9727
           1       0.70      0.37      0.49      9874

    accuracy                           0.60     19601
   macro avg       0.64      0.61      0.58     19601
weighted avg       0.64      0.60      0.58     19601

60.450997398091936
0.3954900260190807
[[8179 1548]
 [6204 3670]]
```

- Parameters indicates that 28% of the data will be used for testing and 72% used for training .
- The Naïve_bayes classifier achieved 60.45% accuracy and 0.58 , 0.61 and 0.64 F1 score , recall and precision respectively.

Experiment 2 :

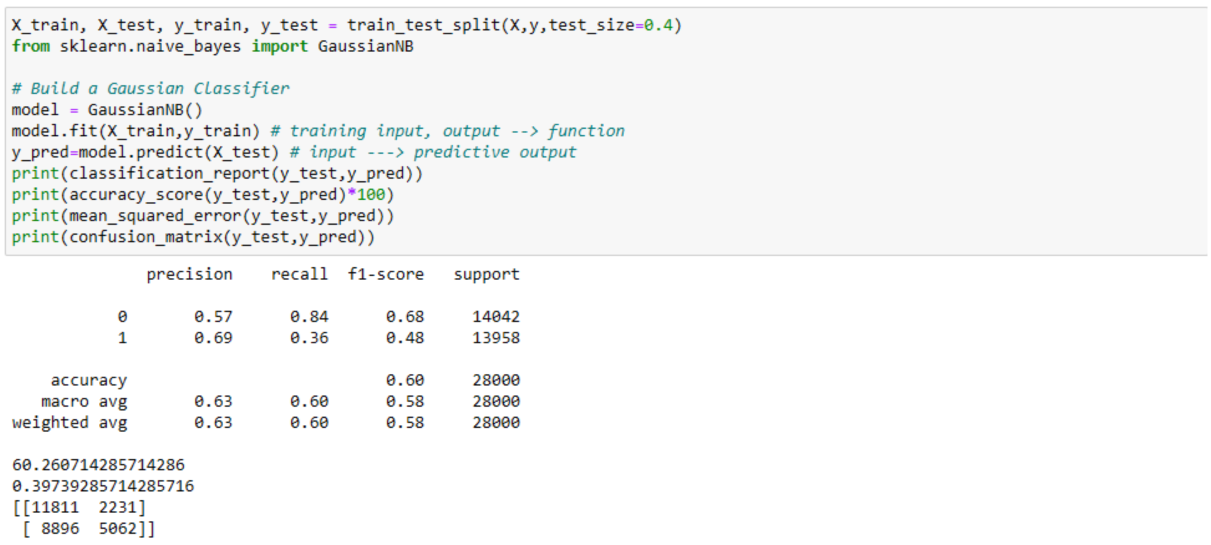


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.56      0.89      0.69     10452
           1       0.74      0.29      0.42     10548

    accuracy                           0.59     21000
   macro avg       0.65      0.59      0.55     21000
weighted avg       0.65      0.59      0.55     21000

59.319047619047616
0.4068095238095238
[[9353 1099]
 [7444 3104]]
```

- Parameters indicates that 30% of the data will be used for testing and 70% used for training .
- The Naïve_bayes classifier achieved 59.31% accuracy and 0.55 , 0.59 and 0.65 F1 score , recall and precision respectively.

<u>Experiment 3 :</u>

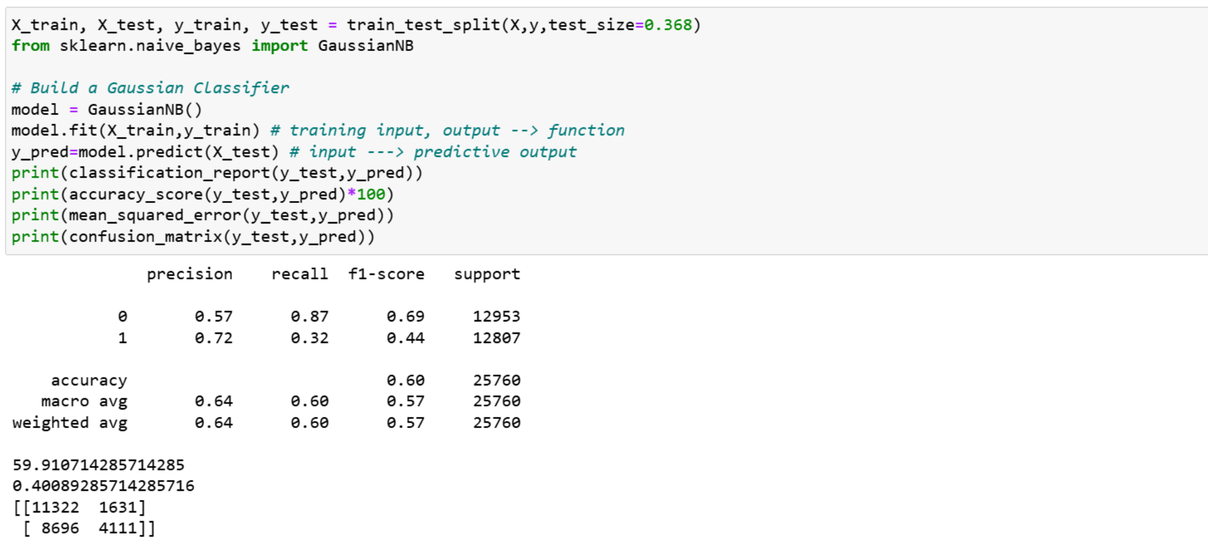


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4)
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.57      0.84      0.68     14042
           1       0.69      0.36      0.48     13958

    accuracy                           0.60     28000
   macro avg       0.63      0.60      0.58     28000
weighted avg       0.63      0.60      0.58     28000

60.260714285714286
0.39739285714285716
[[11811  2231]
 [ 8896  5062]]
```

- Parameters indicates that 40% of the data will be used for testing and 60% used for training .
- The Naïve_bayes classifier achieved 60.26% accuracy and 0.58 , 0.60 and 0.63 F1 score , recall and precision respectively.

<u>Experiment 4 :</u>

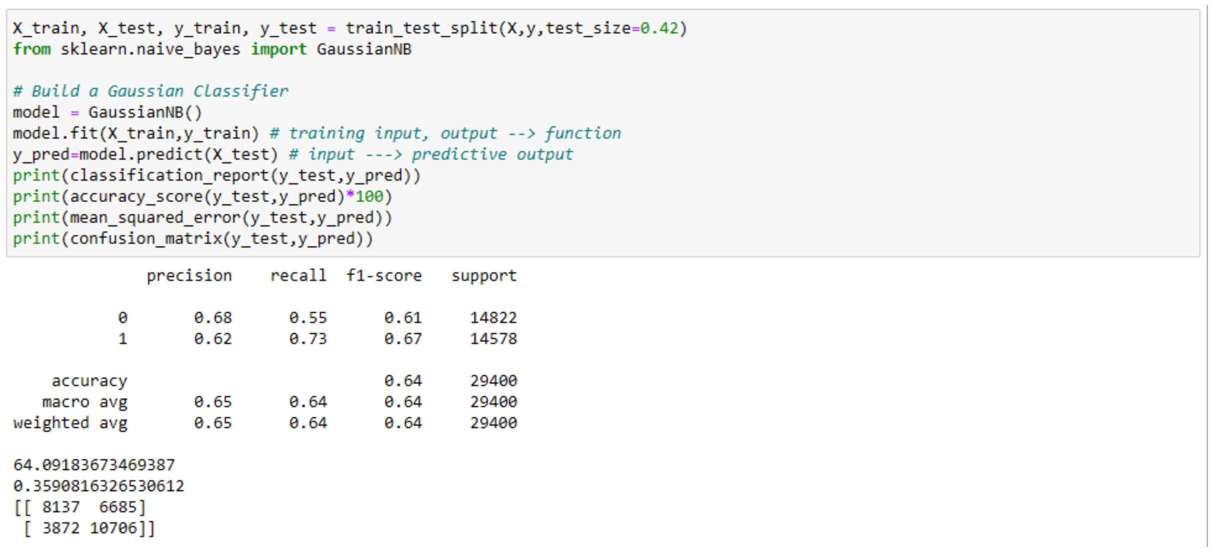


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.368)
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.57      0.87      0.69     12953
           1       0.72      0.32      0.44     12807

    accuracy                           0.60     25760
   macro avg       0.64      0.60      0.57     25760
weighted avg       0.64      0.60      0.57     25760

59.910714285714285
0.40089285714285716
[[11322  1631]
 [ 8696  4111]]
```

- Parameters indicates that 63.2% of the data will be used for testing and 36.8% used for training (Boostrap).
- The Naïve_bayes classifier achieved 59.91 % accuracy and 0.57 , 0.60 and 0.64 F1 score , recall and precision respectively.

Experiment 5 :

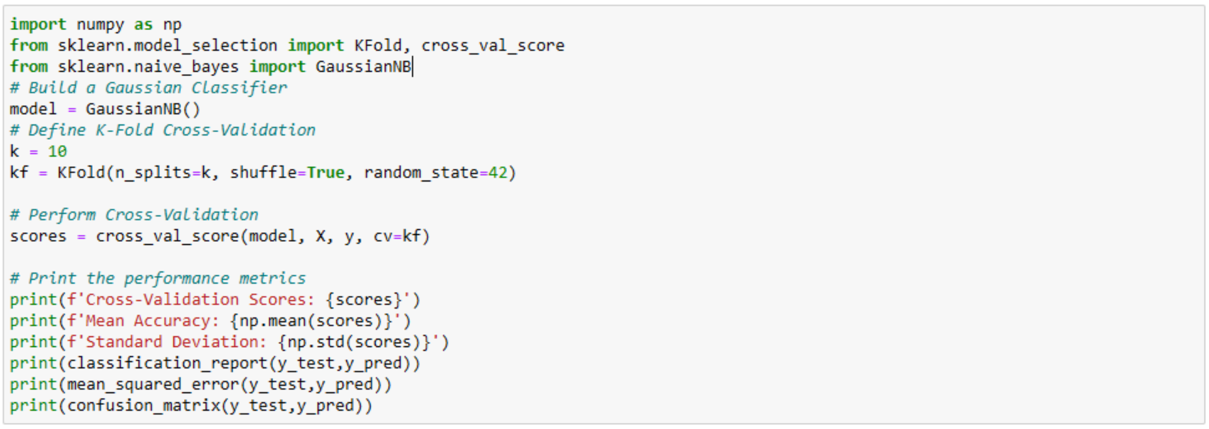


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.42)
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

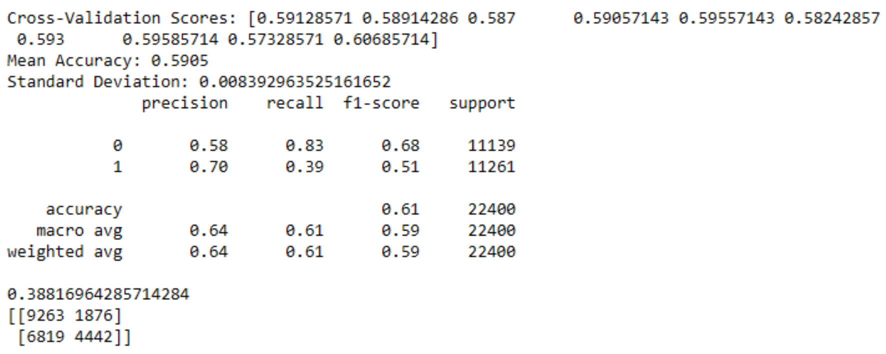
```
              precision    recall  f1-score   support

           0       0.68      0.55      0.61     14822
           1       0.62      0.73      0.67     14578

    accuracy                           0.64     29400
   macro avg       0.65      0.64      0.64     29400
weighted avg       0.65      0.64      0.64     29400

64.09183673469387
0.3590816326530612
[[ 8137  6685]
 [ 3872 10706]]
```

- Parameters indicates that 42% of the data will be used for testing and 58% used for training .
- The Naïve_bayes classifier achieved 64.9% accuracy and 0.64 , 0.64 and 0.65 F1 score , recall and precision respectively.

Experiment 6 :



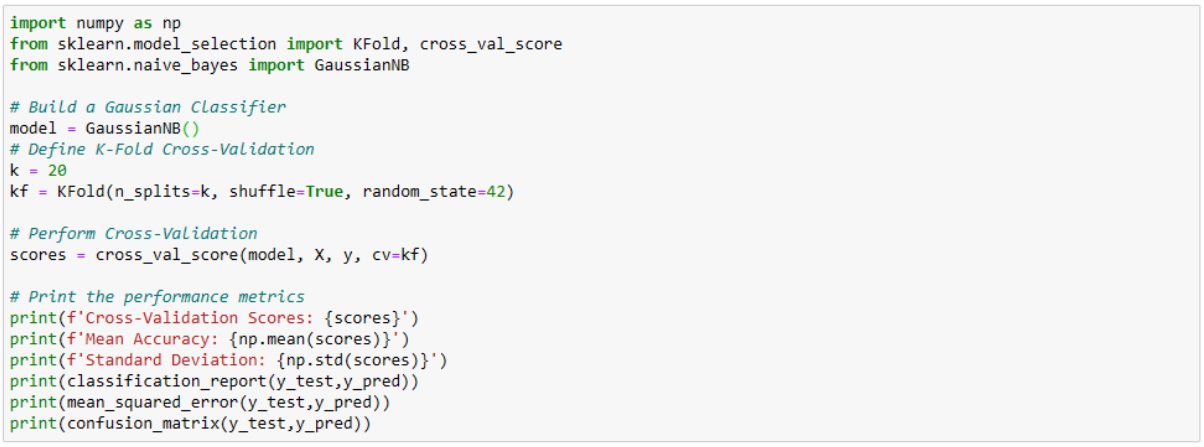
```python
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.naive_bayes import GaussianNB
# Build a Gaussian Classifier
model = GaussianNB()
# Define K-Fold Cross-Validation
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



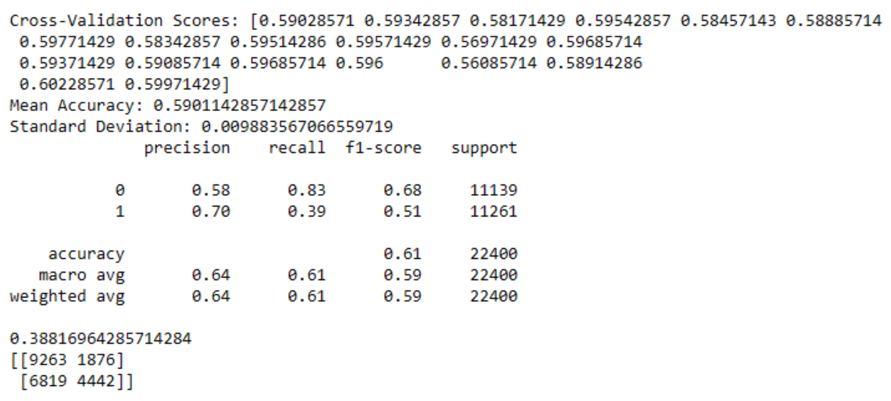
```
Cross-Validation Scores: [0.59128571 0.58914286 0.587      0.59057143 0.59557143 0.58242857
 0.593      0.59585714 0.57328571 0.60685714]
Mean Accuracy: 0.5905
Standard Deviation: 0.008392963525161652
              precision    recall  f1-score   support

           0       0.58      0.83      0.68     11139
           1       0.70      0.39      0.51     11261

    accuracy                           0.61     22400
   macro avg       0.64      0.61      0.59     22400
weighted avg       0.64      0.61      0.59     22400

0.38816964285714284
[[9263 1876]
 [6819 4442]]
```

- Parameters :cross validation=10
- The Naïve_bayes classifier achieved 61% accuracy ,mean accuracy 59% and 0.59 , 0.61 and 0.64 F1 score , recall and precision respectively.

<u>Experiment 7 :</u>



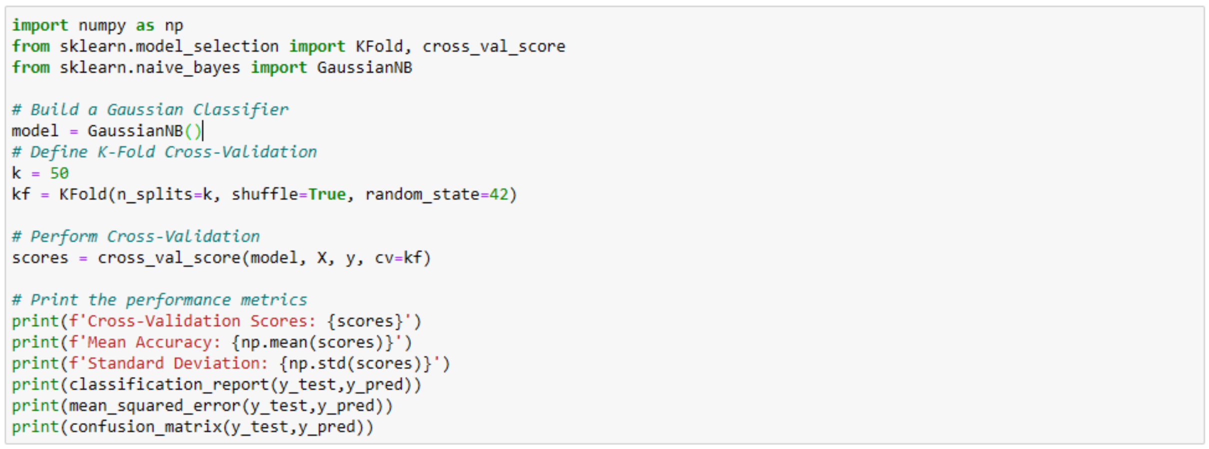
```python
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
# Define K-Fold Cross-Validation
k = 20
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



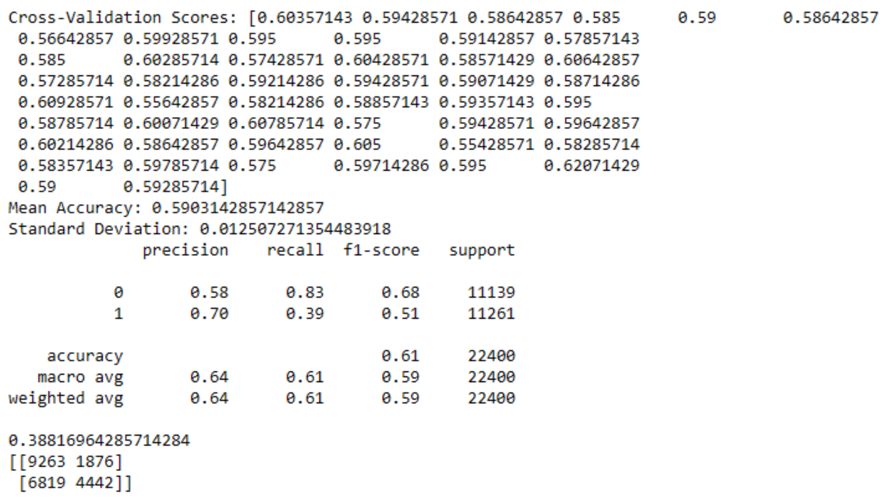
```
Cross-Validation Scores: [0.59028571 0.59342857 0.58171429 0.59542857 0.58457143 0.58885714
 0.59771429 0.58342857 0.59514286 0.59571429 0.56971429 0.59685714
 0.59371429 0.59085714 0.59685714 0.596      0.56085714 0.58914286
 0.60228571 0.59971429]
Mean Accuracy: 0.5901142857142857
Standard Deviation: 0.009883567066559719
              precision    recall  f1-score   support

           0       0.58      0.83      0.68     11139
           1       0.70      0.39      0.51     11261

    accuracy                           0.61     22400
   macro avg       0.64      0.61      0.59     22400
weighted avg       0.64      0.61      0.59     22400

0.38816964285714284
[[9263 1876]
 [6819 4442]]
```

- Parameters :cross validation=20
- The Naïve_bayes classifier achieved 61% accuracy ,mean accuracy 59.01% and 0.59 , 0.61 and 0.64 F1 score , recall and precision respectively.

<u>Experiment 8 :</u>



```python
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.naive_bayes import GaussianNB

# Build a Gaussian Classifier
model = GaussianNB()
# Define K-Fold Cross-Validation
k = 50
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

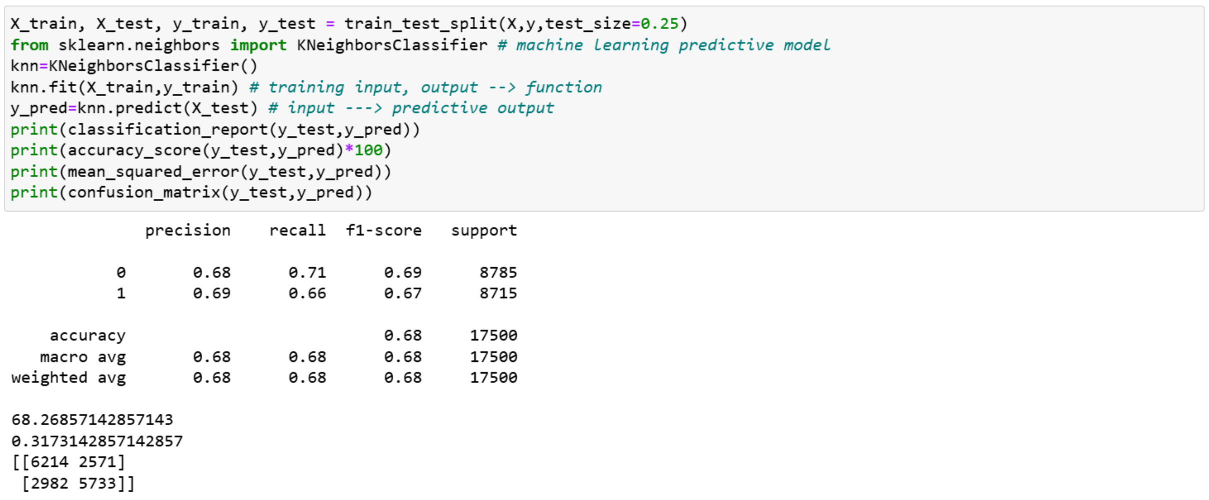


```
Cross-Validation Scores: [0.60357143 0.59428571 0.58642857 0.585      0.59       0.58642857
 0.56642857 0.59928571 0.595      0.595      0.59142857 0.57857143
 0.585      0.60285714 0.57428571 0.60428571 0.58571429 0.60642857
 0.57285714 0.58214286 0.59214286 0.59428571 0.59071429 0.58714286
 0.60928571 0.55642857 0.58214286 0.58857143 0.59357143 0.595
 0.58785714 0.60071429 0.60785714 0.575      0.59428571 0.59642857
 0.60214286 0.58642857 0.59642857 0.605      0.55428571 0.58285714
 0.58357143 0.59785714 0.575      0.59714286 0.595      0.62071429
 0.59       0.59285714]
Mean Accuracy: 0.5903142857142857
Standard Deviation: 0.012507271354483918
              precision    recall  f1-score   support

           0       0.58      0.83      0.68     11139
           1       0.70      0.39      0.51     11261

    accuracy                           0.61     22400
   macro avg       0.64      0.61      0.59     22400
weighted avg       0.64      0.61      0.59     22400

0.38816964285714284
[[9263 1876]
 [6819 4442]]
```

- Parameters :cross validation=50
- The Naïve_bayes classifier achieved 61% accuracy ,mean accuracy 59% and 0.59 , 0.61 and 0.64 F1 score , recall and precision respectively.

3.8 Experiment : KNN model :

Experiment1:

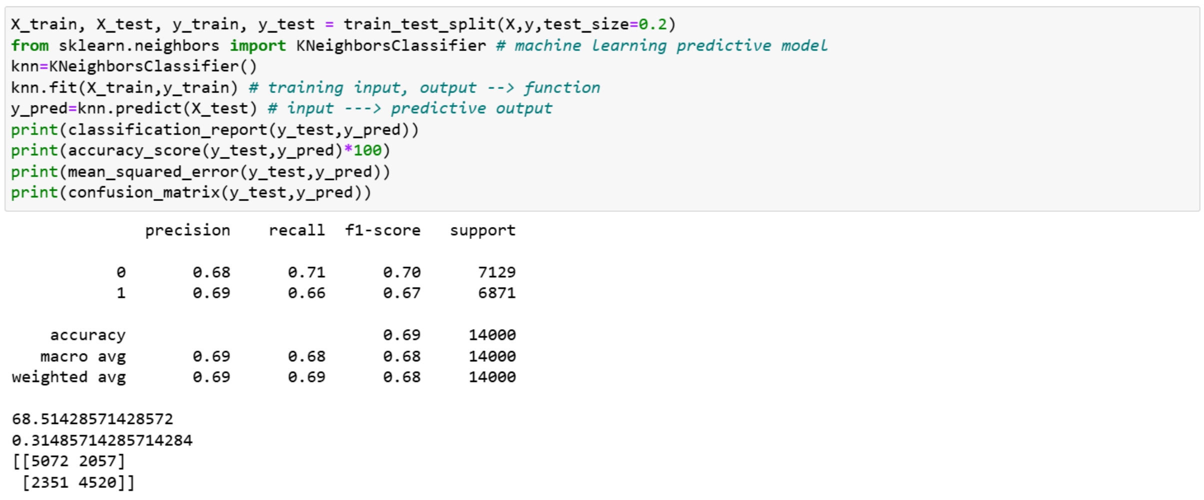


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25)
from sklearn.neighbors import KNeighborsClassifier # machine learning predictive model
knn=KNeighborsClassifier()
knn.fit(X_train,y_train) # training input, output --> function
y_pred=knn.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.68      0.71      0.69      8785
           1       0.69      0.66      0.67      8715

    accuracy                           0.68     17500
   macro avg       0.68      0.68      0.68     17500
weighted avg       0.68      0.68      0.68     17500

68.26857142857143
0.3173142857142857
[[6214 2571]
 [2982 5733]]
```

- Parameters indicates that 25% of the data will be used for testing and 75% used for training .
- The KNN classifier achieved 68.26% accuracy and 0.68 , 0.68 and 0.68 F1 score , recall and precision respectively.

Experiment 2:

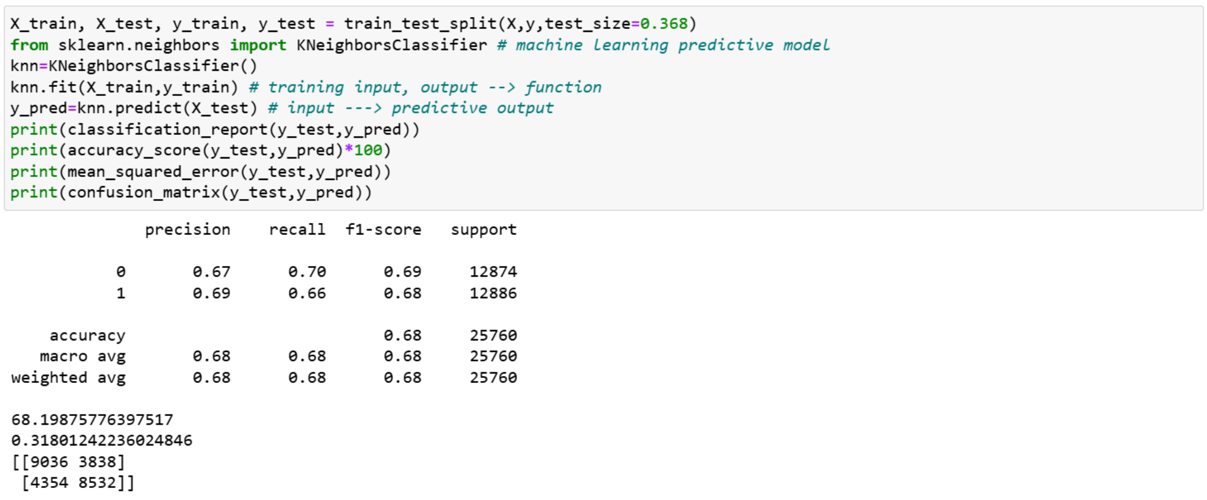


```python
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
from sklearn.neighbors import KNeighborsClassifier # machine learning predictive model
knn=KNeighborsClassifier()
knn.fit(X_train,y_train) # training input, output --> function
y_pred=knn.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.68      0.71      0.70      7129
           1       0.69      0.66      0.67      6871

    accuracy                           0.69     14000
   macro avg       0.69      0.68      0.68     14000
weighted avg       0.69      0.69      0.68     14000

68.51428571428572
0.31485714285714284
[[5072 2057]
 [2351 4520]]
```

- Parameters indicates that 20% of the data will be used for testing and 80% used for training .
- The KNN classifier achieved 68.26% accuracy and 0.68 , 0.68 and 0.68 F1 score , recall and precision respectively.

Experiment 3:

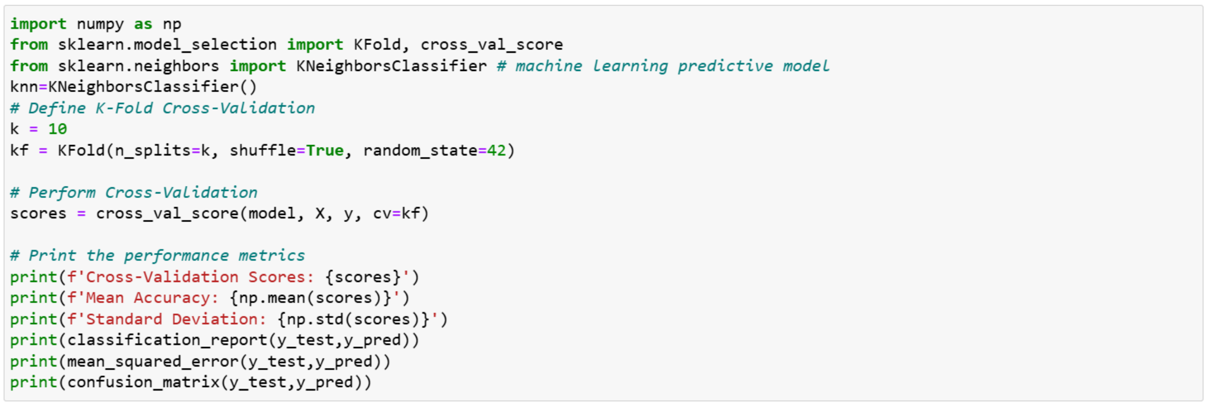


```python
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.368)
from sklearn.neighbors import KNeighborsClassifier # machine learning predictive model
knn=KNeighborsClassifier()
knn.fit(X_train,y_train) # training input, output --> function
y_pred=knn.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

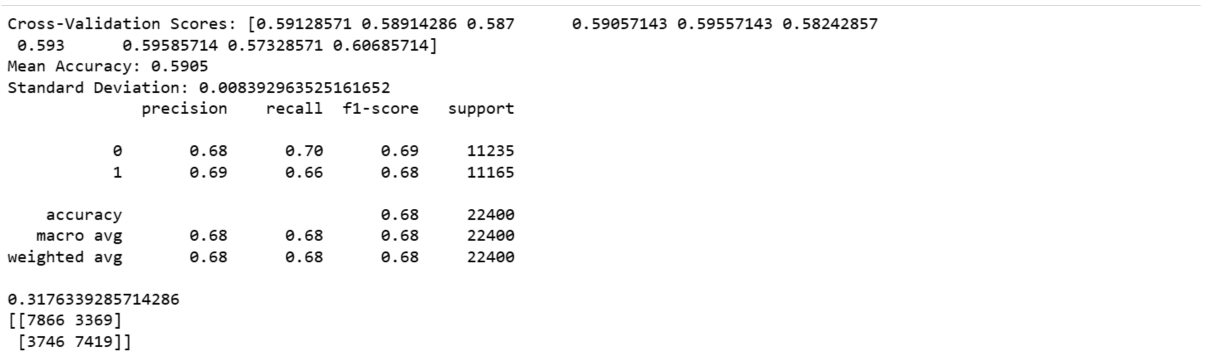
```
              precision    recall  f1-score   support

           0       0.67      0.70      0.69     12874
           1       0.69      0.66      0.68     12886

    accuracy                           0.68     25760
   macro avg       0.68      0.68      0.68     25760
weighted avg       0.68      0.68      0.68     25760

68.19875776397517
0.31801242236024846
[[9036 3838]
 [4354 8532]]
```

- Parameters indicates that 36.8% of the data will be used for testing and 63.2% used for training (Boostrap) .
- The KNN classifier achieved 68.19% accuracy and 0.68 , 0.68 and 0.68 F1 score , recall and precision respectively.

Experiment 4:



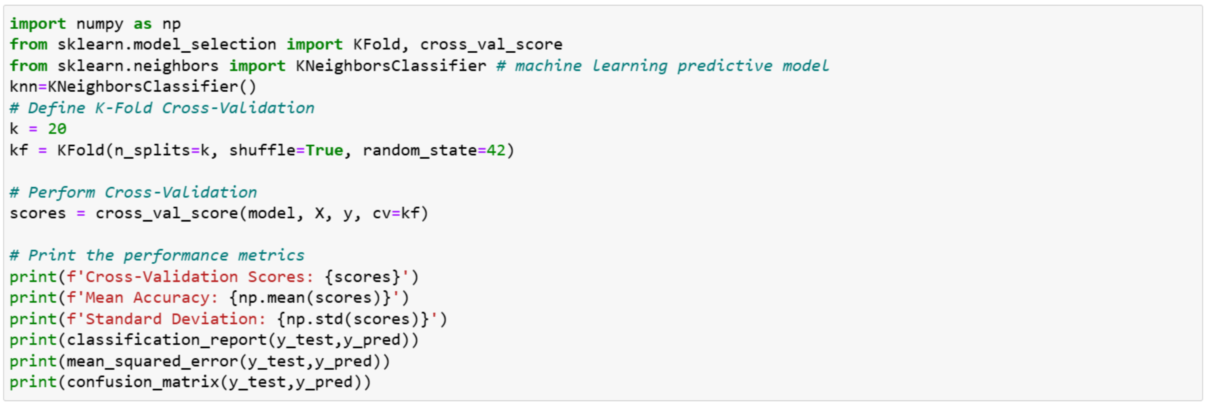
```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier # machine learning predictive model
knn=KNeighborsClassifier()
# Define K-Fold Cross-Validation
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



```
Cross-Validation Scores: [0.59128571 0.58914286 0.587      0.59057143 0.59557143 0.58242857
 0.593      0.59585714 0.57328571 0.60685714]
Mean Accuracy: 0.5905
Standard Deviation: 0.008392963525161652
              precision    recall  f1-score   support

           0       0.68      0.70      0.69     11235
           1       0.69      0.66      0.68     11165

    accuracy                           0.68     22400
   macro avg       0.68      0.68      0.68     22400
weighted avg       0.68      0.68      0.68     22400

0.3176339285714286
[[7866 3369]
 [3746 7419]]
```

- Parameters :cross validation=10.
- The KNN classifier achieved 68% accuracy ,mean accuracy 59% and 0.68 , 0.68 and 0.68 F1 score , recall and precision respectively.

Experiment 5:



```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier # machine learning predictive model
knn=KNeighborsClassifier()
# Define K-Fold Cross-Validation
k = 20
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
Cross-Validation Scores: [0.59028571 0.59342857 0.58171429 0.59542857 0.58457143 0.58885714
 0.59771429 0.58342857 0.59514286 0.59571429 0.56971429 0.59685714
 0.59371429 0.59085714 0.59685714 0.596      0.56085714 0.58914286
 0.60228571 0.59971429]
Mean Accuracy: 0.5901142857142857
Standard Deviation: 0.009883567066559719
              precision    recall  f1-score   support

           0       0.68      0.70      0.69     11235
           1       0.69      0.66      0.68     11165

    accuracy                           0.68     22400
   macro avg       0.68      0.68      0.68     22400
weighted avg       0.68      0.68      0.68     22400

0.3176339285714286
[[7866 3369]
 [3746 7419]]
```

- Parameters :cross validation=20.
- The KNN classifier achieved 68% accuracy ,mean accuracy 59% and 0.68 , 0.68 and 0.68 F1 score , recall and precision respectively.

3.9 Experiment : Tree Desions model :

Experiment 1:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.44)
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.64      0.63      0.64     15448
           1       0.63      0.64      0.64     15352

    accuracy                           0.64     30800
   macro avg       0.64      0.64      0.64     30800
weighted avg       0.64      0.64      0.64     30800

63.59415584415584
0.3640584415584416
[[9787 5661]
 [5552 9800]]
```

- Parameters indicates that 44% of the data will be used for testing and 56% used for training .
- The decision Tree classifier achieved 63.59% accuracy and 0.64 , 0.64 and 0.64 F1 score , recall and precision respectively.

Experiment 2 :

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.26)
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.63      0.64      0.64      9109
           1       0.63      0.63      0.63      9091

    accuracy                           0.63     18200
   macro avg       0.63      0.63      0.63     18200
weighted avg       0.63      0.63      0.63     18200

63.34065934065934
0.3665934065934066
[[5808 3301]
 [3371 5720]]
```

- Parameters indicates that 26% of the data will be used for testing and 74% used for training .
- The decision Tree classifier achieved 63.34% accuracy and 0.63 , 0.63 and 0.63 F1 score , recall and precision respectively.

Experiment 3:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.63      0.64      0.63      7000
           1       0.63      0.63      0.63      7000

    accuracy                           0.63     14000
   macro avg       0.63      0.63      0.63     14000
weighted avg       0.63      0.63      0.63     14000

63.43571428571428
0.36564285714285716
[[4452 2548]
 [2571 4429]]
```

- Parameters indicates that 36.8% of the data will be used for testing and 63.2% used for training (Boostrap) .
- The decision Tree classifier achieved 63.54% accuracy and 0.64 , 0.64 and 0.64 F1 score , recall and precision respectively.

Experiment 4:

```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
# Define K-Fold Cross-Validation
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
Cross-Validation Scores: [0.63457143 0.62757143 0.637      0.63742857 0.63842857 0.63471429
 0.627      0.63471429 0.63057143 0.63042857]
Mean Accuracy: 0.6332428571428571
Standard Deviation: 0.0038835340522124845
              precision    recall  f1-score   support

           0       0.63      0.64      0.63      7000
           1       0.63      0.63      0.63      7000

    accuracy                           0.63     14000
   macro avg       0.63      0.63      0.63     14000
weighted avg       0.63      0.63      0.63     14000

0.36564285714285716
[[4452 2548]
 [2571 4429]]
```

- Parameters :cross validation=10.
- The decision tree classifier achieved 63% accuracy ,mean accuracy 63.332% and 0.63 , 0.63 and 0.63 F1 score , recall and precision respectively.

3.10 Experiment : logistical regression model :

Experiment 1:

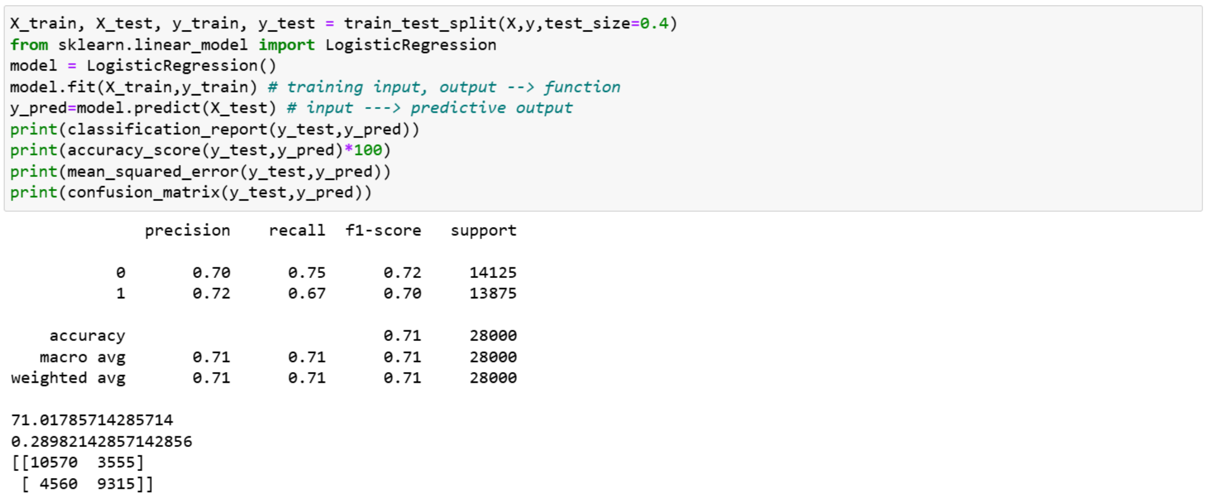
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.28)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.69      0.75      0.72      9786
           1       0.72      0.66      0.69      9815

    accuracy                           0.70     19601
   macro avg       0.71      0.70      0.70     19601
weighted avg       0.71      0.70      0.70     19601

70.36885873169736
0.29631141268302635
[[7293 2493]
 [3315 6500]]
```

- Parameters indicates that 28% of the data will be used for testing and 72% used for training .

- The logistical regression classifier achieved 70.36% accuracy and 0.70 , 0.70 and 0.71 F1 score , recall and precision respectively.

Experiment 2 :

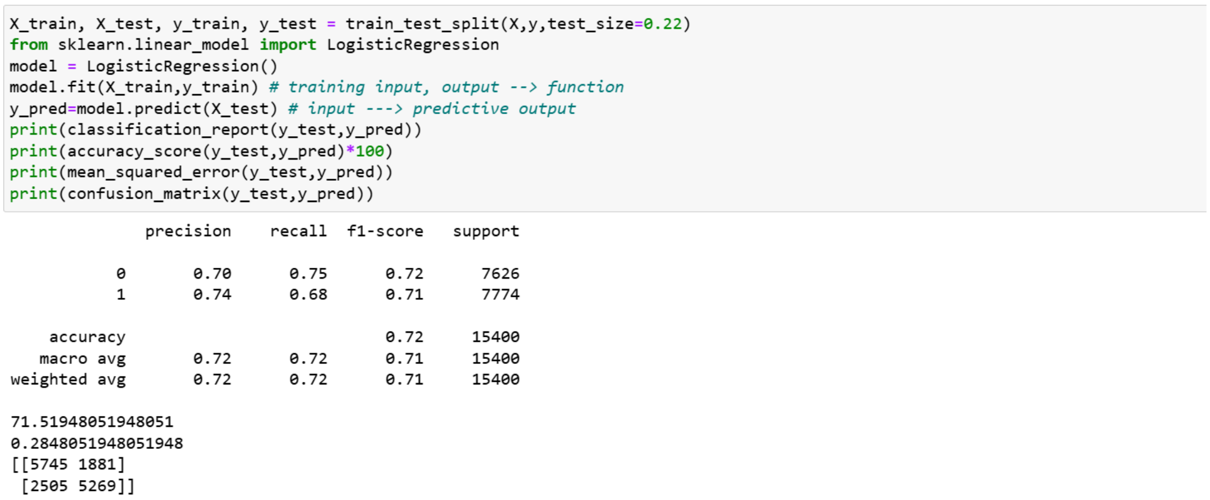


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.75      0.72     14125
           1       0.72      0.67      0.70     13875

    accuracy                           0.71     28000
   macro avg       0.71      0.71      0.71     28000
weighted avg       0.71      0.71      0.71     28000

71.01785714285714
0.28982142857142856
[[10570  3555]
 [ 4560  9315]]
```

- Parameters indicates that 40% of the data will be used for testing and 60% used for training .
- The logistical regression classifier achieved 71.01% accuracy and 0.71 , 0.71 and 0.71 F1 score , recall and precision respectively.

Experiment 3:

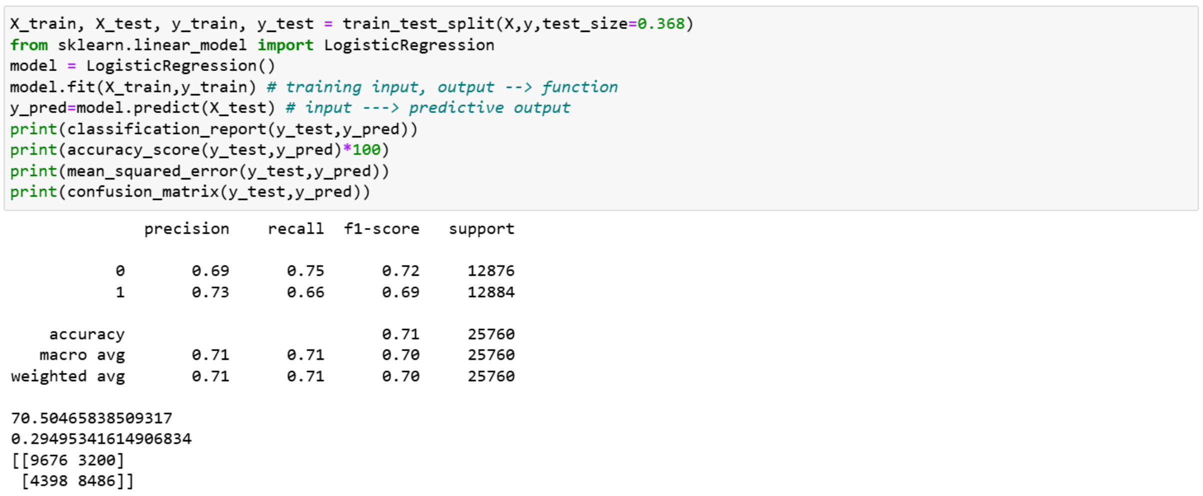


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.22)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.75      0.72      7626
           1       0.74      0.68      0.71      7774

    accuracy                           0.72     15400
   macro avg       0.72      0.72      0.71     15400
weighted avg       0.72      0.72      0.71     15400

71.51948051948051
0.2848051948051948
[[5745 1881]
 [2505 5269]]
```

- Parameters indicates that 22% of the data will be used for testing and 78% used for training .
- The logistical regression classifier achieved 71.51% accuracy and 0.71 , 0.72 and 0.72 F1 score , recall and precision respectively.

Experiment 4:

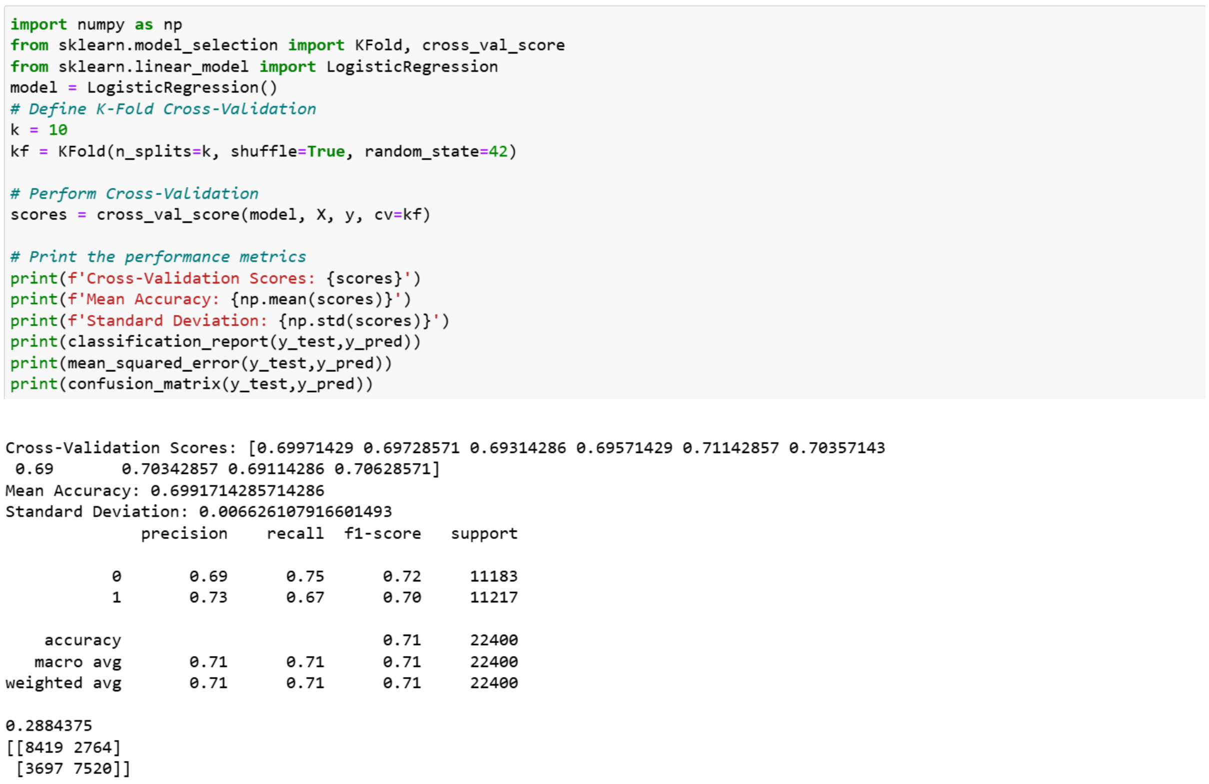


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.368)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.69      0.75      0.72     12876
           1       0.73      0.66      0.69     12884

    accuracy                           0.71     25760
   macro avg       0.71      0.71      0.70     25760
weighted avg       0.71      0.71      0.70     25760

70.50465838509317
0.29495341614906834
[[9676 3200]
 [4398 8486]]
```

- Parameters indicates that 36.8% of the data will be used for testing and 63.2% used for training (Boostrap) .
- The logistical regression classifier achieved 70.50% accuracy and 0.70 , 0.71 and 0.71 F1 score , recall and precision respectively.

Experiment 5:



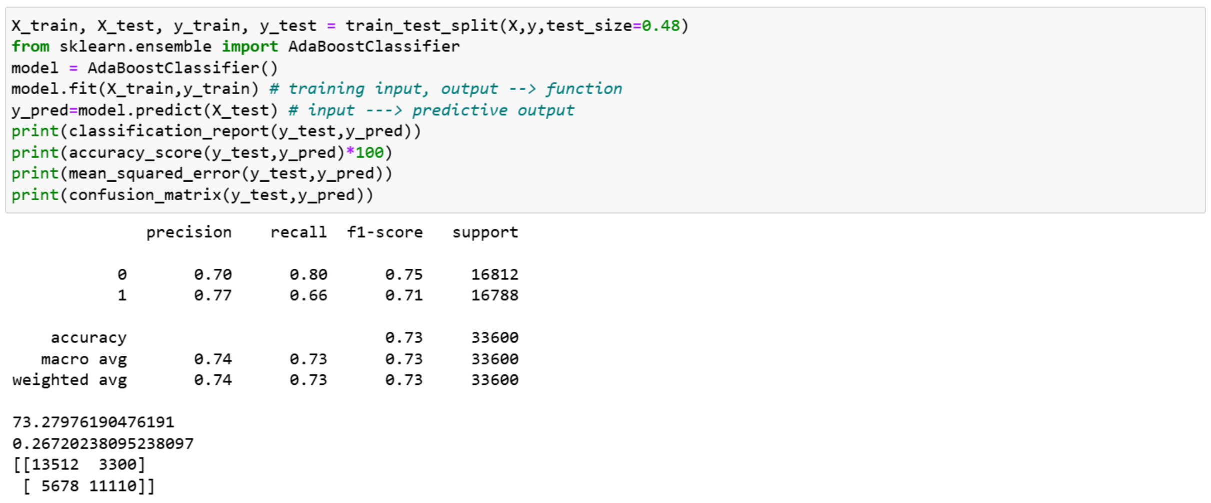
```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
# Define K-Fold Cross-Validation
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
Cross-Validation Scores: [0.69971429 0.69728571 0.69314286 0.69571429 0.71142857 0.70357143
 0.69       0.70342857 0.69114286 0.70628571]
Mean Accuracy: 0.69917142857l4286
Standard Deviation: 0.006626107916601493
              precision    recall  f1-score   support

           0       0.69      0.75      0.72     11183
           1       0.73      0.67      0.70     11217

    accuracy                           0.71     22400
   macro avg       0.71      0.71      0.71     22400
weighted avg       0.71      0.71      0.71     22400

0.2884375
[[8419 2764]
 [3697 7520]]
```

- Parameters :cross validation=10.
- The logistical regression classifier achieved 71% accuracy ,mean accuracy 69.91% and 0.71 , 0.71 and 0.71 F1 score , recall and precision respectively.

3.11 Experiment : AdaBoost model :

Experiment 1:



```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.48)
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.80      0.75     16812
           1       0.77      0.66      0.71     16788

    accuracy                           0.73     33600
   macro avg       0.74      0.73      0.73     33600
weighted avg       0.74      0.73      0.73     33600

73.27976190476191
0.26720238095238097
[[13512  3300]
 [ 5678 11110]]
```

- Parameters indicates that 48% of the data will be used for testing and 52% used for training .
- The AdaBoost classifier achieved 73.27% accuracy and 0.73 , 0.73 and 0.74 F1 score , recall and precision respectively.

Experiment 2:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30)
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier(n_estimators=50, learning_rate=1, random_state=0)
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.71      0.79      0.75     10525
           1       0.76      0.67      0.72     10475

    accuracy                           0.73     21000
   macro avg       0.74      0.73      0.73     21000
weighted avg       0.74      0.73      0.73     21000

73.3047619047619
0.26695238095238094
[[8351 2174]
 [3432 7043]]
```

- Parameters indicates that 30% of the data will be used for testing and 70% used for training .
- The AdaBoost classifier achieved 71.30% accuracy and 0.73 , 0.73 and 0.74 F1 score , recall and precision respectively.

Experiment 3:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.368)
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.80      0.74     12841
           1       0.77      0.66      0.71     12919

    accuracy                           0.73     25760
   macro avg       0.73      0.73      0.73     25760
weighted avg       0.73      0.73      0.73     25760

72.7445652173913
0.27255434782608695
[[10243  2598]
 [ 4423  8496]]
```

- Parameters indicates that 36.8% of the data will be used for testing and 63.2% used for training (Boostrap) .
- The AdaBoost classifier achieved 72.57 % accuracy and 0.73 , 0.73 and 0.72 F1 score , recall and precision respectively.

Experiment 4:

```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.ensemble import AdaBoostClassifier
model = AdaBoostClassifier()
k = 20
kf = KFold(n_splits=k, shuffle=True, random_state=42)
# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)
# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
Cross-Validation Scores: [0.74228571 0.73771429 0.71971429 0.74057143 0.72885714 0.74314286
 0.73457143 0.72342857 0.73685714 0.74057143 0.72371429 0.722
 0.73742857 0.70714286 0.73685714 0.72628571 0.72142857 0.72285714
 0.74       0.73085714]
Mean Accuracy: 0.7308142857142857
Standard Deviation: 0.009430854701890127

              precision    recall  f1-score   support

           0       0.70      0.79      0.74     11197
           1       0.76      0.66      0.71     11203

    accuracy                           0.73     22400
   macro avg       0.73      0.73      0.73     22400
weighted avg       0.73      0.73      0.73     22400

0.2726339285714286
[[8850 2347]
 [3760 7443]]
```

- Parameters :cross validation=20.
- The AdaBoost classifier achieved 73% accuracy ,mean accuracy 73.081% and 0.73 , 0.73 and 0.73 F1 score , recall and precision respectively.

3.12 Experiment : SVM model :

Experiment 1:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
from sklearn.svm import SVC # machine learning predictive model

model = SVC()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.59      0.64      0.62     10562
           1       0.61      0.55      0.58     10438

    accuracy                           0.60     21000
   macro avg       0.60      0.60      0.60     21000
weighted avg       0.60      0.60      0.60     21000

59.84761904761905
0.4015238095238095
[[6789 3773]
 [4659 5779]]
```

- Parameters indicates that 30% of the data will be used for testing and 70% used for training .
- The SVM classifier achieved 59.84% accuracy and 0.60 , 0.60 and 0.60 F1 score , recall and precision respectively.

Experiment 2:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
from sklearn.svm import SVC # machine learning predictive model

model = SVC()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.60      0.65      0.62      7035
           1       0.61      0.55      0.58      6965

    accuracy                           0.60     14000
   macro avg       0.60      0.60      0.60     14000
weighted avg       0.60      0.60      0.60     14000

60.25
0.3975
[[4596 2439]
 [3126 3839]]
```

- Parameters indicates that 20% of the data will be used for testing and 80% used for training .
- The SVM classifier achieved 60.25% accuracy and 0.60 , 0.60 and 0.60 F1 score , recall and precision respectively.

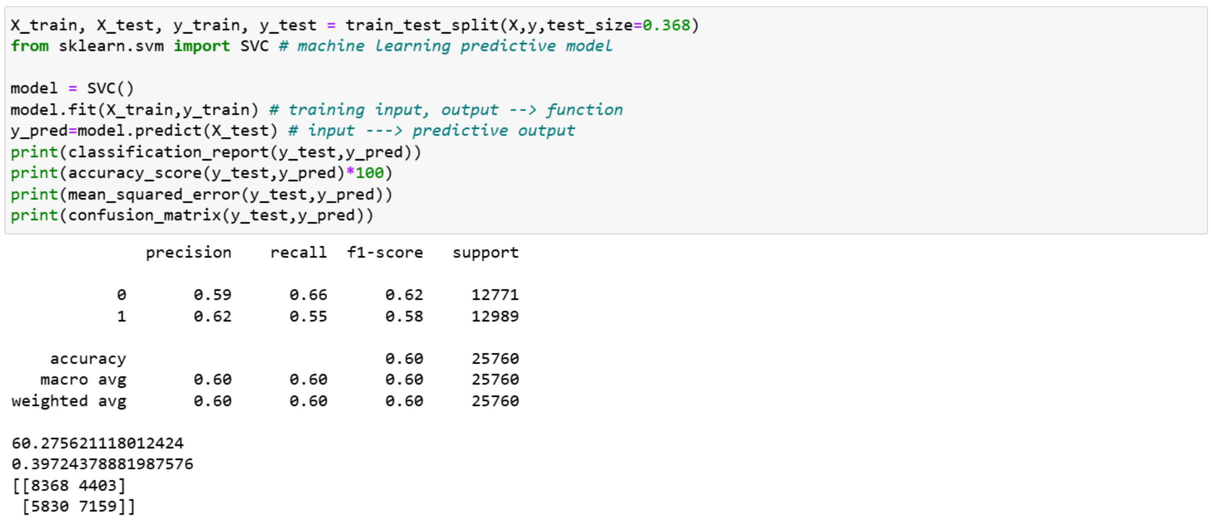
Experiment 3:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.4)
from sklearn.svm import SVC # machine learning predictive model

model = SVC()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.59      0.66      0.62     13940
           1       0.62      0.55      0.58     14060

    accuracy                           0.61     28000
   macro avg       0.61      0.61      0.60     28000
weighted avg       0.61      0.61      0.60     28000

60.517857142857146
0.39482142857142855
[[9171 4769]
 [6286 7774]]
```

- Parameters indicates that 40% of the data will be used for testing and 60% used for training .
- The SVM classifier achieved 60.51% accuracy and 0.60 , 0.61 and 0.61 F1 score , recall and precision respectively.

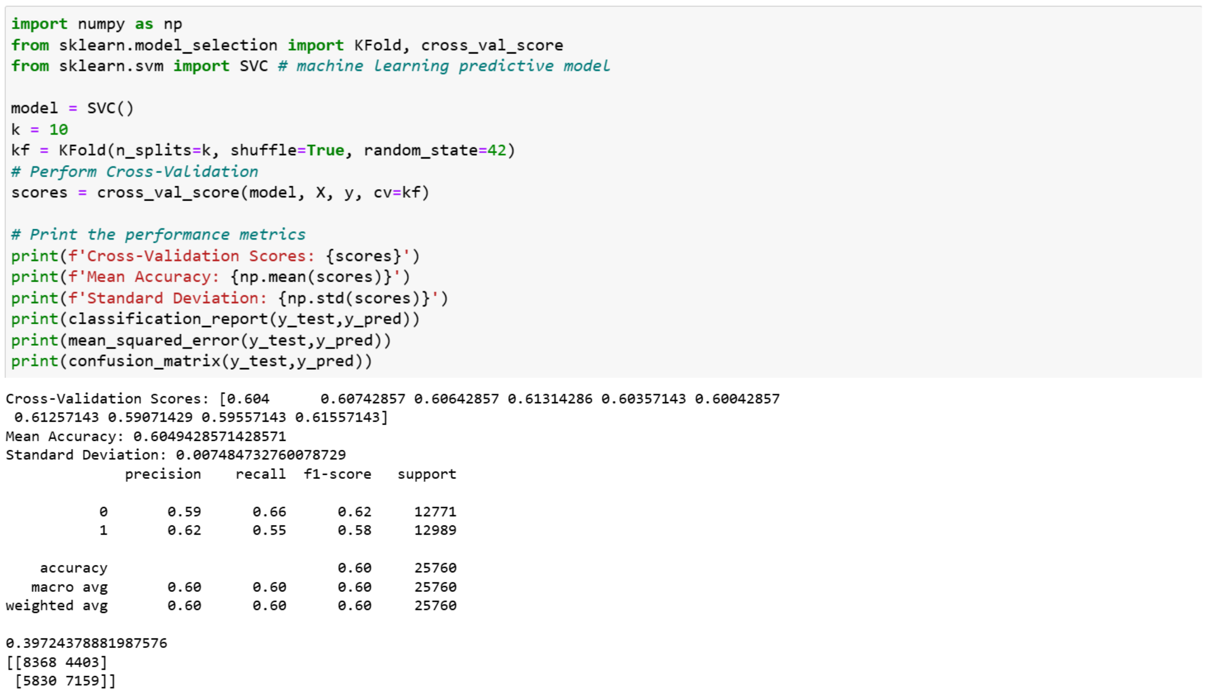
Experiment 4:


```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.368)
from sklearn.svm import SVC # machine learning predictive model

model = SVC()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.59      0.66      0.62     12771
           1       0.62      0.55      0.58     12989

    accuracy                           0.60     25760
   macro avg       0.60      0.60      0.60     25760
weighted avg       0.60      0.60      0.60     25760

60.275621118012424
0.397243788881987576
[[8368 4403]
 [5830 7159]]
```

- Parameters indicates that 36.8% of the data will be used for testing and 63.2% used for training (Boostrap) .
- The SVM classifier achieved 60.27% accuracy and 0.60 , 0.60 and 0.60 F1 score , recall and precision respectively.

Experiment 5:

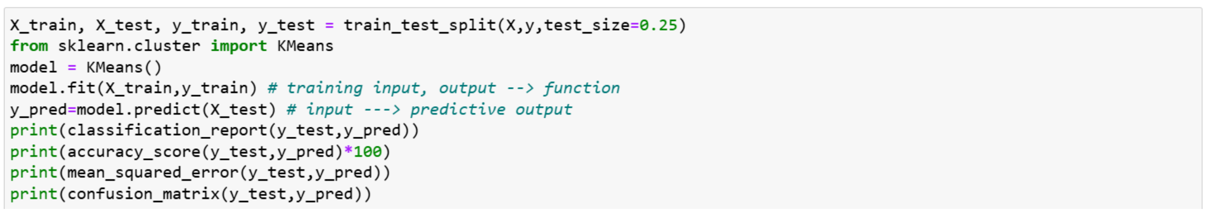

```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.svm import SVC # machine learning predictive model

model = SVC()
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)
# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
Cross-Validation Scores: [0.604      0.60742857 0.60642857 0.61314286 0.60357143 0.60042857
 0.61257143 0.59071429 0.59557143 0.61557143]
Mean Accuracy: 0.6049428571428571
Standard Deviation: 0.007484732760078729
              precision    recall  f1-score   support

           0       0.59      0.66      0.62     12771
           1       0.62      0.55      0.58     12989

    accuracy                           0.60     25760
   macro avg       0.60      0.60      0.60     25760
weighted avg       0.60      0.60      0.60     25760

0.397243788881987576
[[8368 4403]
 [5830 7159]]
```

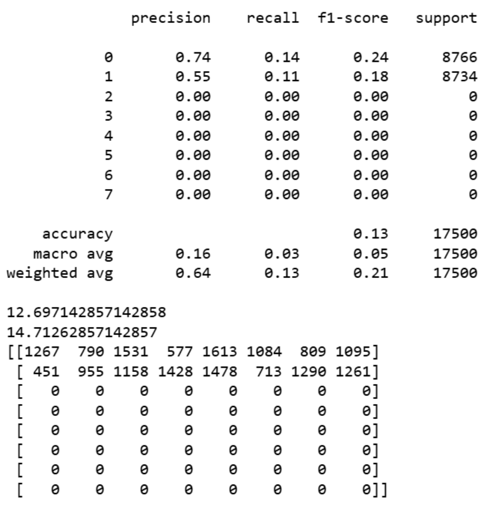
- Parameters :cross validation=10.
- The SVM classifier achieved 60% accuracy ,mean accuracy 60.49% and 0.73 , 0.73 and 0.73 F1 score , recall and precision respectively.

3.13 <u>Experiment : Kmeans model :</u>

Experiment1:



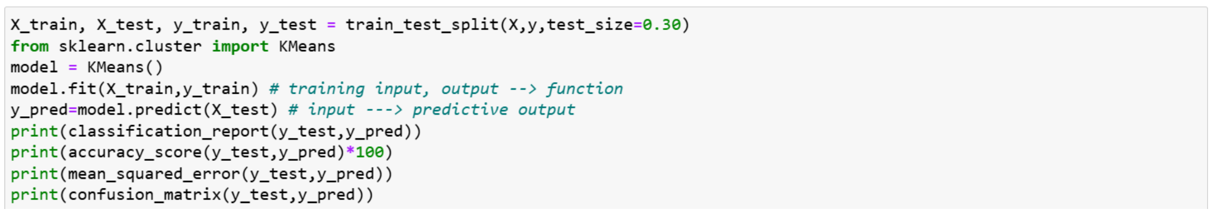
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25)
from sklearn.cluster import KMeans
model = KMeans()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



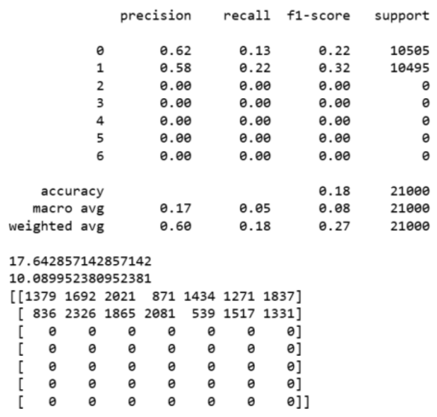
```
              precision    recall  f1-score   support

           0       0.74      0.14      0.24      8766
           1       0.55      0.11      0.18      8734
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0
           7       0.00      0.00      0.00         0

    accuracy                           0.13     17500
   macro avg       0.16      0.03      0.05     17500
weighted avg       0.64      0.13      0.21     17500

12.697142857142858
14.71262857142857
[[1267  790 1531  577 1613 1084  809 1095]
 [ 451  955 1158 1428 1478  713 1290 1261]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]]
```

- Parameters indicates that 25% of the data will be used for testing and 75% used for training .
- The Kmeans classifier achieved 12.69% accuracy and 0.21 , 0.13 and 0.64 F1 score , recall and precision respectively.

Experiment 2:



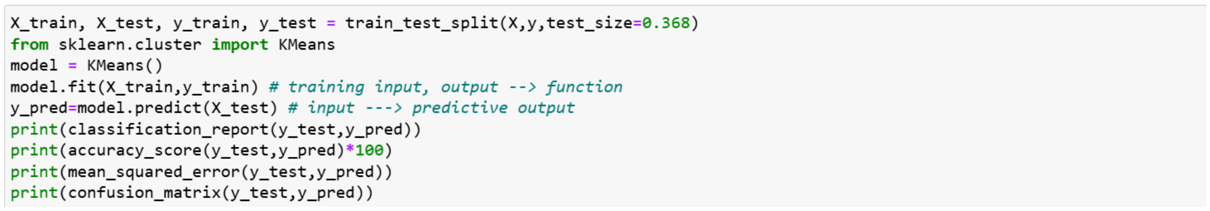
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30)
from sklearn.cluster import KMeans
model = KMeans()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



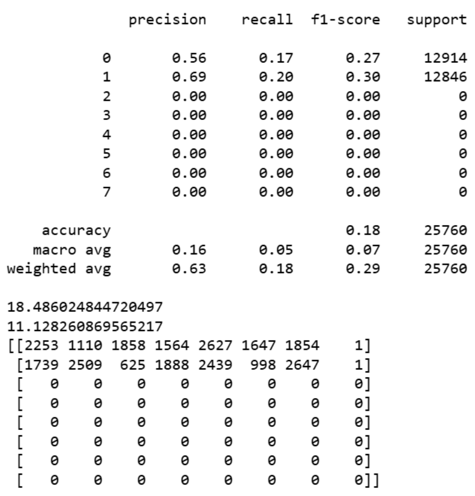
```
              precision    recall  f1-score   support

           0       0.62      0.13      0.22     10505
           1       0.58      0.22      0.32     10495
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0

    accuracy                           0.18     21000
   macro avg       0.17      0.05      0.08     21000
weighted avg       0.60      0.18      0.27     21000

17.642857142857142
10.089952380952381
[[1379 1692 2021  871 1434 1271 1837]
 [ 836 2326 1865 2081  539 1517 1331]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0]]
```

- Parameters indicates that 30% of the data will be used for testing and 70% used for training .
- The Kmeans classifier achieved 17.64% accuracy and 0.27 , 0.18 and 0.60 F1 score , recall and precision respectively.

Experiment 3:



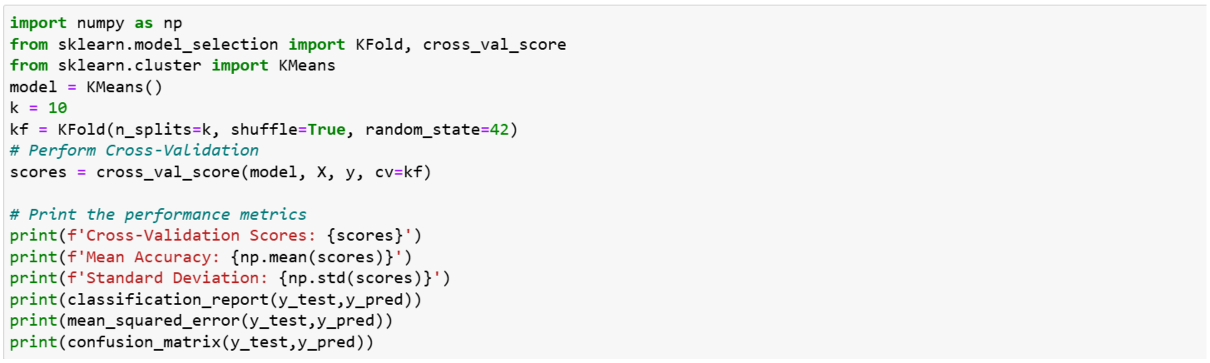
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.368)
from sklearn.cluster import KMeans
model = KMeans()
model.fit(X_train,y_train) # training input, output --> function
y_pred=model.predict(X_test) # input ---> predictive output
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test,y_pred)*100)
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



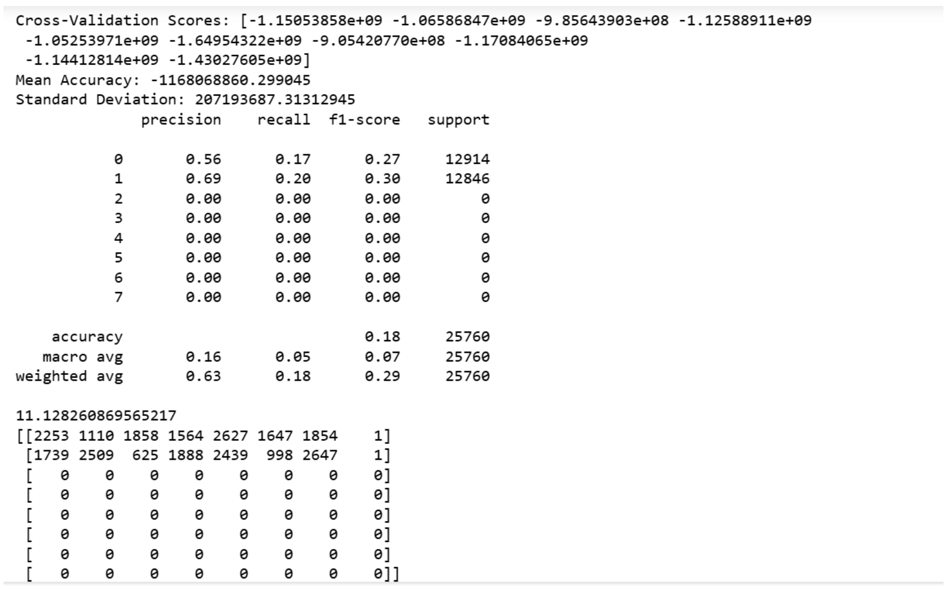
```
              precision    recall  f1-score   support

           0       0.56      0.17      0.27     12914
           1       0.69      0.20      0.30     12846
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0
           7       0.00      0.00      0.00         0

    accuracy                           0.18     25760
   macro avg       0.16      0.05      0.07     25760
weighted avg       0.63      0.18      0.29     25760

18.486024844720497
11.128260869565217
[[2253 1110 1858 1564 2627 1647 1854    1]
 [1739 2509  625 1888 2439  998 2647    1]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]]
```

- Parameters indicates that 36.8% of the data will be used for testing and 63.2% used for training (Boostrap) .
- The KNN classifier achieved 18.48% accuracy and 0.29 , 0.18 and 0.63 F1 score , recall and precision respectively.

Experiment 5:



```
import numpy as np
from sklearn.model_selection import KFold, cross_val_score
from sklearn.cluster import KMeans
model = KMeans()
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=42)
# Perform Cross-Validation
scores = cross_val_score(model, X, y, cv=kf)

# Print the performance metrics
print(f'Cross-Validation Scores: {scores}')
print(f'Mean Accuracy: {np.mean(scores)}')
print(f'Standard Deviation: {np.std(scores)}')
print(classification_report(y_test,y_pred))
print(mean_squared_error(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```



```
Cross-Validation Scores: [-1.15053858e+09 -1.06586847e+09 -9.85643903e+08 -1.12588911e+09
 -1.05253971e+09 -1.64954322e+09 -9.05420770e+08 -1.17084065e+09
 -1.14412814e+09 -1.43027605e+09]
Mean Accuracy: -1168068860.299045
Standard Deviation: 207193687.31312945
              precision    recall  f1-score   support

           0       0.56      0.17      0.27     12914
           1       0.69      0.20      0.30     12846
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0
           7       0.00      0.00      0.00         0

    accuracy                           0.18     25760
   macro avg       0.16      0.05      0.07     25760
weighted avg       0.63      0.18      0.29     25760

11.128260869565217
[[2253 1110 1858 1564 2627 1647 1854    1]
 [1739 2509  625 1888 2439  998 2647    1]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]
 [   0    0    0    0    0    0    0    0]]
```

- Parameters :cross validation=10.
- The KNN classifier achieved 18% accuracy ,mean accuracy 11.12% and 0.29 , 0.18 and 0.63 F1 score , recall and precision respectively.

3.13 Test Results:

In this section, the proposed method has been implemented on the test data, and the results have been compared with other ML methods such as KNN, Decision Tree, AdaBoost, SVM,Logistic regression . Also, the four different types of performance metrics, such as Sensitivity, Specificity, Accuracy, and Area Under Curve, have been calculated. Furthermore, other numbers of the training and testing data were selected and tested, but the best performance has been obtained from the mentioned percentages. The performance evaluation results of ML models without applying feature selection with different techniques (boostrap ,croos validation) .

Performance comparison of different ML models :

T :train

t :test

Supervised learning						
Model	Split of data	Accuarcy%	Precesion %	Recall %	Entropy%	F1-score%
Naive bayes						
	T=0.72/t=0.28	60.45	64	61	31	58
	T=0.7/t=0.3	59.31	65	59	30	55
	T=0.6/t=0.4	60.26	53	60	36	58
Boostrap	T=0.632/t=0.368	59.91	64	60		57
	Croos validation =10	59/61	64	61	37	59
	Croos validation	59.9/61	64	61	37	59

	=20					
	Croos validation =50	50.9/61	64	61	37	59
	T=0.58/t=0.42	64.09	65	64	42	64
	T=0.74/t=0.26	68.64	69	63	32.93	69
SVM						
	T=0.70/t=0.3	59.84	60	60	30	60
	T=0.8/t=0.2	60.25	60	60	30	60
	T=0.6/t=0.4	60.51	61	61	30.5	61
Boostrap	T=0.632/t=0.368	60.27	60	60	30	60
	Croos validation =10	60.49/60	60	60	30	60
AdaBoost						
	T=0.48/t=0.48	73.27	74	73	36.74	73
	T=0.7/t=0.3	73.30	74	73	36.74	73
	T=0.75/t=0.25	73.85	74	74	37	74
Boostrap	T=0.632/t=0.368	72.74	73	73	36.5	73
	Croos validation =10	72.57/73	73	73	36.5	72
	Croos validation =20	73.10/73	73	73	36.5	73
	Croos validation =50	73.08/73	73	73	36.5	73
Decesion tree						
	T=0.74/t=0.26	63.34	63	63	31.5	63
	T=0.56/t=0.44	63.59	64	64	32	64
	T=0.75/t=0.25	64.41	64	64	32	64
Boostrap	T=0.632/t=0.368	63.54	64	64	32	64
	Croos validation =10	63.3/63	63	63	31.5	63
	Croos validation =20	63.4/63	63	63	31.5	63
	Croos validation =50	63.24/63	63	63	31.5	63
Logistical regression						
	T=0.72/t=0.28	70.36	71	70	35.24	71
	T=0.6/t=0.4	71.01	71	71	35.5	71
Boostrap	T=0.632/t=0.368	70.50	71	71	35.5	70
	Croos validation =10	69.91/71	71	71	35.5	71
	Croos validation =20	69.73/71	71	71	35.5	71
	Croos validation =50	69.75/71	71	71	35.5	71
	Unsupervised learning					
Model	Split of data	Accuarcy%	Precesion%	Recall%	Entropy%	F1-score%
Kmeans						
	T=0.75/t=0.25	12.69	64	13	10.80	21

	T=0.7/t=0.3	17.64	60	18	13.84	27
	T=0.6/t=0.4	17.77	55	18	13.56	27
	T=0.632/t=0.368	18.48	63	18	14	29
	Croos validation =10	18/11.12	63	16	12.75	29
	Croos validation =20	18	63	18	14	29
	Croos validation =50	18	63	18	14	29
KNN						
	T=0.75/t=0.25	68.26	68	68	34	68
	T=0.8/t=0.2	68.51	69	69	34.5	68
	T=0.78/t=0.22	68.62	69	69	34.5	69
Boostrap	T=0.632/t=0.368	68.19	68	68	34	68
	Croos validation =10	59/68	68	68	34	68
	Croos validation =20	59/68	68	68	34	68
	Croos validation =50	59/68	68	68	34	68
	T=0.74/t=0.26	68.64	69	63	32.93	69

According to the results of the studies, the classification accuracy of the tree decision , Naïve bayes , AdaBoost, SVM , Kmeans , KNN and Logistic regression classifier models was 64.41%, 68.64%, 73.85%, 60.51%, 18.48%, 68.64% and 71.01%, respectively. The AdaBoost classifier model was the most accurate when compared to the other ML models, and the accuracy rose to 73.85%. [18]

The table represente the final results of models according the accuracy:

Model	accuracy
Naïve bayes	68.64%
AdaBoost	73.85%
SVM	60.51%
Kmeans	18.48%
KNN	68.64%
Logistic regression	71.01%
Tree decision	64.41%

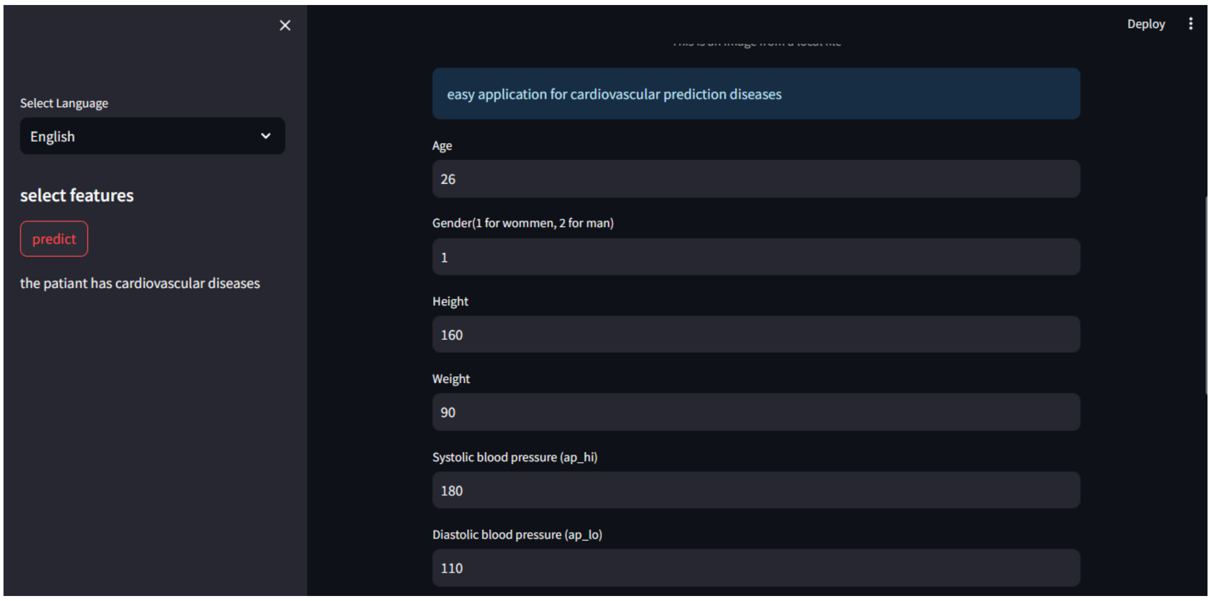
3.14 Conclusion :

The AdaBoost classifier achieved the best performance with 73.85% accuracy and an F1 score and AUC of 0.74 and 0.73, respectively. Next, the domain adaptation technique has been applied to demonstrate the versatility of the proposed prediction system. Finally, the best-performed AdaBoost framework has been deployed into an website application and to predict cardiovascular disease instantly. There are some future scopes of this work, for example, we recommend getting additional private data with a larger cohort of patients to get better results. Another extension of this work is combining machine learning models with fuzzy logic techniques and applying optimization approaches.

Annex

Annex :

Interface: application prediction cardiovascular disease

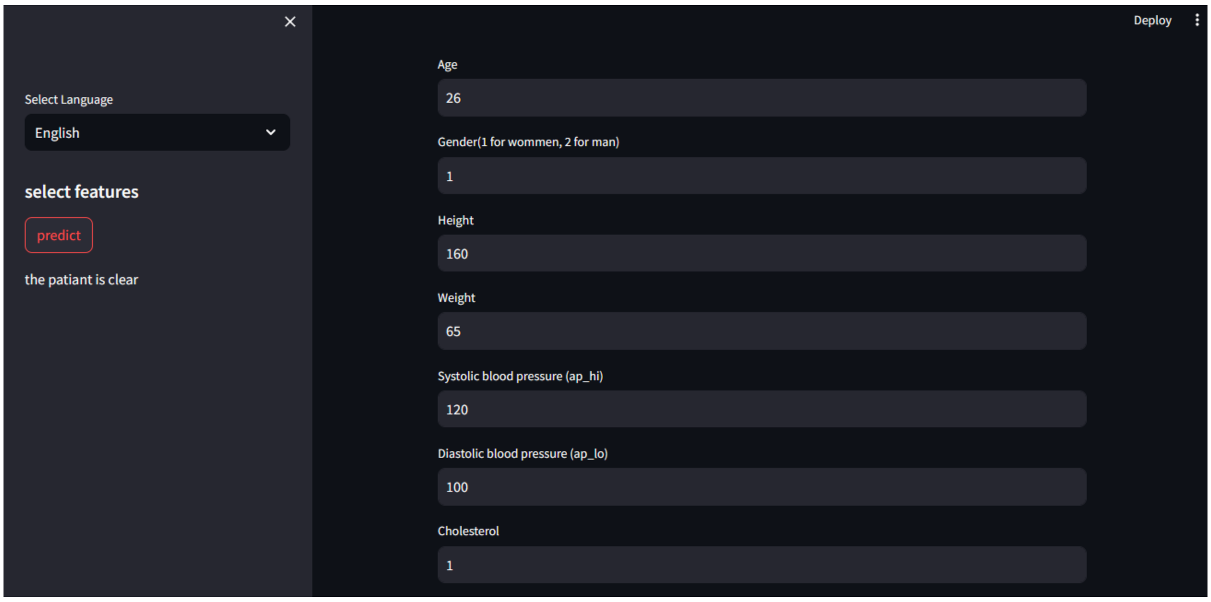


the patiant has cardiovascular diseases

easy application for cardiovascular prediction diseases

Age
26
Gender(1 for wommen, 2 for man)
1
Height
160
Weight
90
Systolic blood pressure (ap_hi)
180
Diastolic blood pressure (ap_lo)
110

- This interface represent the case where this patient has cardiovascular diseases .



the patiant is clear

Age
26
Gender(1 for wommen, 2 for man)
1
Height
160
Weight
65
Systolic blood pressure (ap_hi)
120
Diastolic blood pressure (ap_lo)
100
Cholesterol
1

- This interface represent the case where this patient hasn't cardiovascular diseases .

Generale Conclusion :

Cardiovascular diseases (CVDs) remain a leading cause of mortality worldwide, underscoring the urgent need for innovative approaches to prediction, prevention, and management. In recent years, machine learning (ML) has emerged as a powerful tool in the fight against CVDs, offering unprecedented opportunities for early detection, risk stratification, and personalized interventions.

Through predictive modeling, ML algorithms analyze vast amounts of patient data, including demographics, clinical variables, genetic markers, and imaging studies, to identify patterns, correlations, and predictive factors associated with CVDs. By leveraging advanced algorithms such as logistic regression, decision trees, and deep learning models, healthcare providers can develop accurate risk prediction models tailored to individual patients, enabling timely interventions and preventive strategies.

Moreover, ML-driven approaches enable the integration of diverse data sources, including electronic health records, wearable devices, and genetic profiles, facilitating a holistic understanding of cardiovascular health and disease trajectories. Real-time monitoring and analysis of patient data empower clinicians to proactively identify high-risk individuals, optimize treatment strategies, and improve patient outcomes.

The application of ML in cardiovascular disease management extends beyond risk prediction to encompass personalized treatment recommendations, medication adherence monitoring, and lifestyle interventions. By leveraging data-driven insights, healthcare providers can deliver targeted interventions that address the underlying drivers of CVDs, thereby improving patient adherence, satisfaction, and long-term outcomes.

However, the widespread adoption of ML in clinical practice requires addressing several challenges, including data privacy concerns, algorithm interpretability, and regulatory considerations. Collaborative efforts among healthcare professionals, data scientists, policymakers, and industry stakeholders are essential to overcome these barriers and harness the full potential of ML in cardiovascular disease management.

In conclusion, machine learning holds immense promise in transforming the landscape of cardiovascular disease management, offering unprecedented opportunities for early detection, personalized interventions, and improved patient outcomes. By harnessing the power of data-driven insights, healthcare providers can advance precision medicine initiatives and mitigate the burden of cardiovascular diseases on global health.

Reference

- [1] https://my.clevelandclinic.org/health/diseases/21493-cardiovascular-disease (2024, April 15-20)
- [2] https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)?gad_source=1&gclid=Cj0KCQjwudexBhDKARIsAI-GWYXf0RGiHMkaKnLrhn1jU0YC1gdaUmTYDEejhOoYnVM7f-YvETDq0-8aAiQmEALw_wcB (2024, April 15-20)
- [3] https://my.clevelandclinic.org/health/diseases/21493-cardiovascular-disease (2024, April 15-20)
- [4] https://stefanini.com/en/insights/news/machine-learning-models-for-precise-predictive-analytics#:~:text=Predictive%20modeling%20algorithms%20are%20a,and%20make%20data%2Ddriven%20decisions. (2024, April 15-20)
- [5] https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec
- [6] https://www.purestorage.com/knowledge/machine-learning-performance-metrics.html#:~:text=Accuracy%20is%20a%20performance%20metric,out%20of%20all%20predictions%20made. (2024, April 15-20)
- [7] https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ (2024, April 15-20)
- [8] https://www.ibm.com/topics/logistic-regression (2024, April 15-20)

- [9] https://domino.ai/data-science-dictionary/jupyter-notebook (2024, May 1-10)
- [10] https://en.wikipedia.org/wiki/Python_(programming_language) (2024, May 1-10)
- [11] https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/ (2024, May 1-10)
- [12] https://www.kaggle.com/code/anoopashware/chronic-kidney-disease-prediction-with-100-accu (2024, May 1-10)
- [13] https://www.kaggle.com/code/shtrausslearning/heart-disease-gaussian-process-models (2024, May 1-10)
- [14] https://en.wikipedia.org/wiki/Kaggle (2024, May 1-10)
- [15] https://pypi.org/project/pandas/ (2024, May 1-10)
- [16] https://pypi.org/project/numpy/ (2024, May 1-10)
- [17] https://www.linkedin.com/pulse/what-streamlit-pranav-kshirsagar-qxmkf (2024, May 1-10)
- [18] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10378171/ (2024, May 1-10)
- [19] https://longmoreclinic.org/coronary-artery-disease-a-comprehensive-overview/
- [20] https://www.medicalnewstoday.com/articles/184601#diagnosis
- [21] https://cvtsc.com/conditions/peripheral-arterial-disease/
- [22] https://www.kidshealth.org.nz/what-rheumatic-heart-disease
- [23] https://drraghu.com/resources/congenital-heart-disease/basics-of-congenital-heart-disease/
- [24] https://www.medindia.net/health/conditions/pulmonary-embolism-and-deep-vein-thrombosis.htm
- [25] Nabel, E . G. (2003). Cardiovascular disease. New England Journal of Medicine, 349(1), 60-72.
- [26] Gaziano, T., Reddy, K. S., Paccaud, F., Horton, S., & Chaturvedi, V. (2006). Cardiovascular disease. Disease Control Priorities in Developing Countries. 2nd edition.

- [27] Sin, D. D., & Man, S. P. (2005). Chronic obstructive pulmonary disease: a novel risk factor for cardiovascular disease. Canadian journal of physiology and pharmacology, 83(1), 8-13.
- [28] Python, W. (2021). Python. Python releases for windows, 24.
- [29] Dinesh, K. G., Arumugaraj, K., Santhosh, K. D., & Mareeswari, V. (2018, March). Prediction of cardiovascular disease using machine learning algorithms. In 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT) (pp. 1-7). IEEE.
- [30] Louridi, N., Amar, M., & El Ouahidi, B. (2019, October). Identification of cardiovascular diseases using machine learning. In 2019 7th mediterranean congress of telecommunications (CMT) (pp. 1-6). IEEE.
- [31] Sitar-Tăut, A., Zdrenghea, D., Pop, D., & Sitar-Tăut, D. (2009). Using machine learning algorithms in cardiovascular disease risk evaluation. Age, 1(4), 4.
- https://byjus.com/maths/data-sets/

ملخص

[Tapez un texte]

أصبحت أمراض القلب والأوعية الدموية مشكلة متنامية وهي قاتلة عالمية تؤثر على الناس من جميع الأعمار. أمراض القلب والأوعية الدموية هي فئة من الأمراض التي تشمل القلب أو الأوعية الدموية. ومن المعروف أنه يشمل أمراض القلب الإقفارية (IHD)، والأمراض الدماغية الوعائية، وارتفاع ضغط الدم، وأمراض الشرايين الطرفية. إن معدل الإصابة بالمرض وارتفاع تكاليف العلاج يجعل من هذا المرض تهديدًا خطيرًا. إحدى طرق التنبؤ بارتفاع خطر الإصابة بأمراض القلب والأوعية الدموية هي استخدام التعلم الخاضع للإشراف وغير الخاضع للإشراف، والذي يمكن أن يفصل الأفراد إلى مجموعات أكثر عرضة للخطر ومجموعات أقل عرضة للخطر.

ذكرت هذه الورقة البحثية مقاييس أداء مختلفة، وهي الدقة والاستدعاء والدقة ودرجة F1 والمساحة تحت المنحنى لمختلف تقنيات التعلم الآلي والتجميع. فيما يلي أسماء خوارزميات التصنيف التي سنقوم بتنفيذها ومقارنة الدقة في هذا البحث:

Linear Classifiers: Logistic Regression/1.
K-Nearest Neighbor2/ .
Support Vector Machine (SVM) 3/.
Decision Trees 4/.
AdaBoost5/.
Kmeans6/.

حقق مصنف AdaBoost أفضل أداء بدقة تصل إلى73.85% ودرجة F1 وAUC تبلغ 0.74و0.73 على التوالي. بعد ذلك، تم تطبيق تقنية تكييف المجال لإثبات تنوع نظام التنبؤ المقترح. وأخيرًا، تم نشر إطار عمل AdaBoost الأفضل أداءً في موقع ويب للتنبؤ بأمراض القلب والأوعية الدموية على الفور. هناك بعض المجالات المستقبلية لهذا العمل، على سبيل المثال، نوصي بالحصول على بيانات خاصة إضافية مع مجموعة أكبر من المرضى للحصول على نتائج أفضل. امتداد آخر لهذا العمل هو الجمع بين نماذج التعلم الآلي وتقنيات المنطق الغامض وتطبيق أساليب التحسين.

Abstract

Cardiovascular disease has become a growing problem and is a global killer that affects people of all ages. cardiovascular disease is a class of diseases that involve the heart or blood vessels. it is well known for including ischemic heart disease (ihd), cerebrovascular disease, hypertension, and peripheral artery disease. the morbidity rate and high treatment costs make this disease a serious threat. a methods of predicting a higher risk of cardiovascular disease is by using supervised and unsupervised learning, which can separate individuals into higher risk and lower risk groups.

This research paper reported different performance metrics, that is, precision, recall, accuracy, F1 score, and AUC for various machine learning and ensemble techniques. Here are the names of classification algorithms which we are going to implement and compare the accuracy in this research:

1) Linear Classifiers: Logistic Regression.
2) K-Nearest Neighbor .
3) Support Vector Machine (SVM) .
4) Decision Trees .
5) AdaBoost.
5) Kmeans.

The AdaBoost classifier achieved the best performance with 73.85% accuracy and an F1 score and AUC of 0.74 and 0.73, respectively. Next, the domain adaptation technique has been applied to demonstrate the versatility of the proposed prediction system. Finally, the best-performed AdaBoost framework has been deployed into an website application and to predict cardiovascular disease instantly. There are some future scopes of this work, for example, we recommend getting additional private data with a larger cohort of patients to get better results. Another extension of this work is combining machine learning models with fuzzy logic techniques and applying optimization approaches.

Résumé

Les maladies cardiovasculaires sont devenues un problème croissant et constituent une maladie mortelle à l’échelle mondiale qui touche les personnes de tous âges. Les maladies cardiovasculaires sont une classe de maladies qui touchent le cœur ou les vaisseaux sanguins. il est bien connu pour inclure les cardiopathies ischémiques (IHD), les maladies cérébrovasculaires, l'hypertension et les maladies artérielles périphériques. le taux de morbidité et les coûts de traitement élevés font de cette maladie une menace sérieuse. Une méthode permettant de prédire un risque plus élevé de maladie cardiovasculaire consiste à utiliser l'apprentissage supervisé et non supervisé, qui peut séparer les individus en groupes à risque plus élevé et à risque plus faible.

Ce document de recherche a rapporté différentes mesures de performances, à savoir la précision, le rappel, l'exactitude, le score F1 et l'AUC pour diverses techniques d'apprentissage automatique et d'ensemble. Voici les noms des algorithmes de classification que nous allons mettre en œuvre et comparer la précision dans cette recherche :

1) Classificateurs linéaires : régression logistique.
2) K-Voisin le plus proche.
3) Machine à vecteurs de support (SVM).
4) Arbres de décision.
5) AdaBoost.
5) Ksignifie.

Le classificateur AdaBoost a obtenu les meilleures performances avec une précision de 73,85 % et un score F1 et une AUC de 0,74 et 0,73, respectivement. Ensuite, la technique d'adaptation de domaine a été appliquée pour démontrer la polyvalence du système de prédiction proposé. Enfin, le framework AdaBoost le plus performant a été déployé sur une application site Web et permet de prédire instantanément les maladies cardiovasculaires. Il existe des perspectives futures pour ce travail, par exemple, nous recommandons d'obtenir des données privées supplémentaires avec une plus grande cohorte de patients pour obtenir de meilleurs résultats. Une autre extension de ce travail consiste à combiner des modèles d'apprentissage automatique avec des techniques de logique floue et à appliquer des approches d'optimisation.