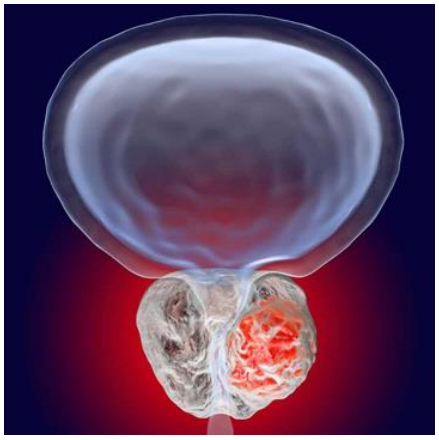
ST. PAUL THE APOSTLE UNIVERSITY OF INFORMATION SCIENCE AND TECHNOLOGY

Diploma Work (Bachelor Thesis)

Topic: Performance Analysis of the Machine Learning Algorithms in Prostate Cancer Prediction



Mentor: Professor Dijana Capeska Bogatinoska, Ph.D.

Student: Dimitar Mitrevski – Computer Science and Engineering

Statement

I, student Dimitar Mitrevski, enrolled at the University for Information Science and Technology - "St. Paul the Apostle" - Ohrid in Faculty of Computer Science and Engineering with index number 1057, under full material, moral and criminal responsibility declare that I am the author of this paper, titled "Performance Analysis of the Machine Learning Algorithms in Prostate Cancer Prediction", which is an original work that I have written under the supervision of Dijana Capeska Bogatinoska, Ph.D.

Place   Signature

Abstract

Prostate cancer is a prevalent and complex disease that requires accurate detection for effective management and treatment. This research paper focuses on analyzing the performance of various machine learning (ML) algorithms in predicting prostate cancer. The study aims to improve the accuracy and reliability of prostate cancer detection by evaluating the effectiveness of different ML algorithms.

The research methodology involves acquiring a dataset of prostate cancer cases and healthy controls. The dataset is then preprocessed and feature-engineered to ensure optimal input for the ML algorithms. Several popular ML algorithms, including logistic regression, random forest classification, and k-nearest neighbors (KNN) and etc., are implemented and evaluated using appropriate evaluation measures such as accuracy, sensitivity, specificity, and precision.

The results of the study demonstrate the potential of ML algorithms in prostate cancer prediction. Logistic regression exhibits promising performance, achieving high accuracy and balanced sensitivity and specificity. Random forest classification, with its ability to handle complex relationships, also shows competitive performance. KNN, relying on proximity-based classification, demonstrates strengths in specific scenarios but has limitations in others.

Keywords: algorithms, analysis, machine learning, prostate cancer

Table of Contents

List of figures

List of abbreviations

AUC – Area Under Curve

CNN – Convolutional Neural Network

DT – Decision Tree

FN – False negatives

FP – False positives

KNN – “K” nearest neighbors

ML – Machine Learning

MRI - Magnetic Resonance Image

PC – Prostate Cancer

PHI - Prostate Health Index

PIA – Proliferative inflammatory atrophy

PIN – Prostate Intraepithelial neoplasia

PSA - Prostate-Specific Antigen

RF- Random Forest

ROC – Receiver Operating Characteristic

SVM- Support Vector Machine

TN – True negatives

TP – True positives

XGB – Extreme Gradient Boosting

STD – Standard Deviation

Tables

Acknowledgments

I am immensely grateful to my supervisor Dijana C. Bogatinoska whose guidance, expertise, and invaluable feedback have been instrumental in shaping the direction and quality of this research. Your continuous support and encouragement throughout the project have been truly invaluable.

I am indebted to the researchers whose work in the field of prostate cancer detection has laid the foundation for this study. Their dedication and contribution to the field have been a source of inspiration and motivation.

Furthermore, I would like to extend my appreciation to my friends and family for their unwavering support, understanding, and encouragement throughout this research journey. Your belief in me and constant encouragement have been vital in overcoming challenges and staying motivated.

While the list above captures the key individuals and entities, I am aware that many others have contributed to this research endeavor in different ways. To everyone who has been a part of this journey, no matter how big or small, please accept my deepest appreciation for your contributions.

Thank you all for being a part of this research and for your sincere support.

1. Introduction

Cancer starts when cells in the body begin to grow out of control. Cells in nearly any part of the body can become cancer cells, and can then spread to other areas of the body.

1.1 Problem overview - What is prostate cancer?

When cells in the prostate gland begin to grow out of control, prostate cancer develops. Only men have the prostate gland in their bodies. It produces a portion of the fluid that is found in semen.

The prostate is located in front of the rectum, the final segment of the intestines, and underneath the bladder, the hollow organ where pee is stored. Seminal vesicles, a group of glands just behind the prostate, provide the majority of the fluid for semen. The urethra, the tube that exits the body through the penis and delivers urine and sperm, passes through the middle of the prostate.

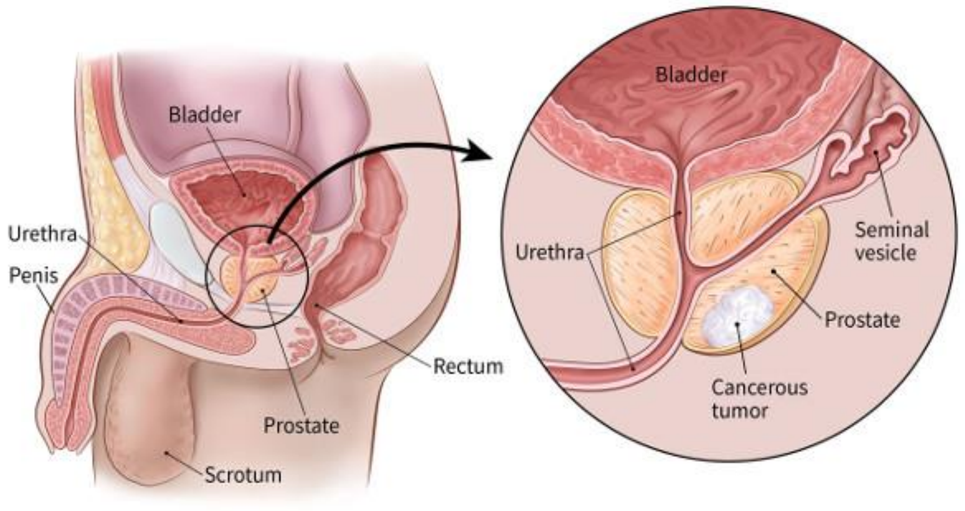


Figure 1 – Physiology of the male reproductive system. Source: https://www.cancer.org/cancer/types/prostate-cancer/about/what-is-prostate-cancer.html

The size of the prostate can change as a man ages. In younger men, it is about the size of a walnut, but it can be much larger in older men.

1.1.1 Types of prostate cancer

Adenocarcinomas make up almost all cases of prostate cancer. The gland cells—the cells that produce the prostate fluid that is mixed with the semen—are where these malignancies grow.

The following cancers can also develop in the prostate:

- Small cell carcinomas
- Neuroendocrine tumors (other than small cell carcinomas)
- Transitional cell carcinomas
- Sarcomas

These further forms of prostate cancer are uncommon. It is almost guaranteed that the prostate cancer you have is an adenocarcinoma if you are diagnosed with it.

1.1.2 Possible pre-cancerous conditions of the prostate

Although it is not yet known for sure, some research shows that prostate cancer initially manifests as a pre-cancerous disease. When a man undergoes a prostate biopsy (the removal of tiny portions of the prostate to check for cancer), these diseases may occasionally be discovered.

Prostate Intraepithelial neoplasia (PIN)

PIN causes alterations in the appearance of the prostate gland cells under a microscope, but the aberrant cells don't appear to be spreading to other regions of the prostate like cancer cells would. Cell pattern abnormalities are categorized into the following groups:

Low-grade PIN: Prostate cell patterns almost seem normal.

High-grade PIN: The cell patterns appear more atypical.

It is not believed that low-grade PIN increases a man's chance of developing prostate cancer. On the other hand, prostate cancer is hypothesized to have a potential precursor in high-grade PIN. Prostate cancer is more likely to occur in the future if high-grade PIN is discovered during a prostate biopsy.

Some men get PIN in their prostates as early as their 20s. However, a lot of PIN guys never have prostate cancer.

Proliferative inflammatory atrophy (PIA)

The prostate cells seem smaller than usual in PIA, and the region exhibits symptoms of inflammation. Although PIA is not cancer, experts think it can occasionally cause high-grade PIN or perhaps prostate cancer.

1.1.3 Key statistics for prostate cancer

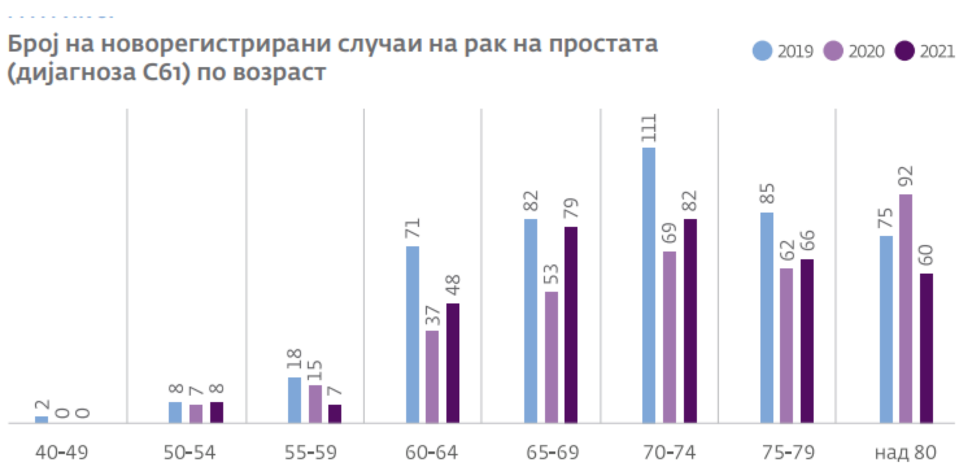


Figure 2 – Overview of new cases diagnosed with prostate cancer in Macedonia during the period 2019 – 2021
Source: https://www.stat.gov.mk/publikacii/2023/ZeniteMazite_2023.pdf

Risk of prostate cancer

About 1 man in 8 will be diagnosed with prostate cancer during his lifetime.[1]

Prostate cancer is more likely to develop in older men and in non-Hispanic Black men. About 6 cases in 10 are diagnosed in men who are 65 or older, and it is rare in men under 40. The average age of men when they are first diagnosed is about 66.[1]

The number of newly-diagnosed prostate cancer cases in men varies between 65 and 79 years in Macedonia. [2]

Deaths from prostate cancer

Prostate cancer is the second leading cause of cancer death in American men, behind only lung cancer. About 1 man in 41 will die of prostate cancer. [1]

Prostate cancer can be a serious disease, but most men diagnosed with prostate cancer do not die from it. In fact, more than 3.1 million men in the United States who have been diagnosed with prostate cancer at some point, are still alive today. [1]

The prostate cancer death rate declined by about half from 1993 to 2013, most likely due to earlier detection and advances in treatment. Since then, however, the pace of decline has slowed, likely reflecting the rise in cancers being found at an advanced stage. [1]

1.1.4 Detection

Early detection

Doctors agree that the prostate-specific antigen (PSA) blood test is not a perfect test for finding prostate cancer early. It misses some cancers, and it sometimes finds cancers that probably never need to be treated. Researchers are working on strategies to address these issues.

Alternative tests:

- The Prostate Health Index (PHI), which combines the results of total PSA, free PSA, and proPSA to help determine how likely it is that a man has prostate cancer that might need treatment
- The 4Kscore test, which combines the results of total PSA, free PSA, intact PSA, and human kallikrein 2 (hK2), along with some other factors, to help determine how likely a man is to have prostate cancer that might need treatment
- Tests (such as Progensa) that look at the level of prostate cancer antigen 3 (PCA3) in the urine after a digital rectal exam (DRE). The DRE pushes some of the prostate cells into the urine. The higher the level, the more likely that prostate cancer is present.
- Tests that look for an abnormal gene change called *TMPRSS2:ERG* in prostate cells in urine collected after a DRE. This gene change is found in some prostate cancers, but it is rarely found in the cells of men without prostate cancer.
- ConfirmMDx, which is a test that looks at certain genes in the cells from a prostate biopsy sample.

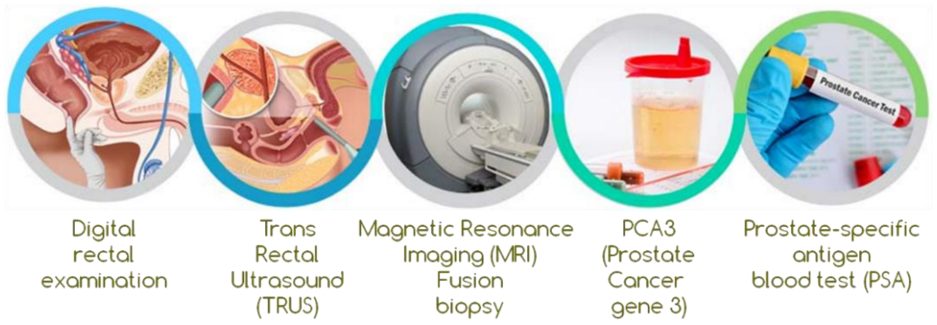

Digital rectal examination
Trans Rectal Ultrasound (TRUS)
Magnetic Resonance Imaging (MRI) Fusion biopsy
PCA3 (Prostate Cancer gene 3)
Prostate-specific antigen blood test (PSA)
Prostate Cancer Test

Figure 3 - Alternative test for detecting prostate cancer. Source: oncodiag.fr/m-302-prostate-cancer.html

1.2 Thesis overview

This thesis focuses on analyzing the performance of various machine learning (ML) algorithms in the context of prostate cancer detection. Dataset of 100 patients with prostate cancer is used with 10 parameters.

The research aims to identify the most effective ML approach for accurate and reliable prediction of prostate cancer. The study involves the evaluation of different algorithms using a comprehensive dataset and appropriate evaluation measures.

In the following sections, there will be presented a detailed explanation of all ML algorithms used and its characteristics as well as their implementation in the problem. The ultimate goal is to contribute to the advancement of prostate cancer diagnosis and improve patient outcomes.

1.3 What has been achieved in the field

Several achievements have been accomplished, including:

Improved Diagnostic Accuracy: ML algorithms have shown promising results in improving the accuracy of prostate cancer diagnosis. It can be concluded that by combining machine learning techniques, improved GrowCut algorithm and Zernik feature selection algorithm, we can detect lesions on MRI images of prostate cancer in an efficient manner.[3] Studies have reported higher sensitivity and specificity rates compared to traditional diagnostic methods like PSA testing, reducing the number of false negatives and false positives.

Automated Image Analysis: ML models, particularly deep learning approaches such as convolutional neural networks (CNNs), have been successfully applied to analyze medical images, including magnetic resonance.

2. State of the art

- Unique aspect(s)

The traditional methods used for prostate cancer detection such as biopsy, PSA testing, and imaging modalities (MRI, ultrasound) – are the foundations of cancer prediction. We want to go one step further and to automatize the results of one's screening making it at the same time accurate and reliable. Biopsies can sometimes be painful or uncomfortable for the patient, so by using the ML-based approaches we can avoid the biopsies.

Many researches have been done regarding various ML algorithms. Each of them works better in different scenarios, and each one has its own limitations and advantages. In the section *4. Methods used – general definition of the used algorithms*, an overview of each algorithm will be given in order to have a better insight of the algorithm itself and how it operates on the dataset that is provided. Contemporary algorithms that have a massive role in the digital world currently such as logistic regression, support vector machines, random forest, k-nearest neighbors, artificial neural networks will be tested and their application in prostate cancer detection tasks evaluated hence.

Integration of Multi-Modal Data has been conducted by several researchers such as combining clinical, genetic, and imaging data, to improve prostate cancer detection accuracy. There are many datasets for prostate cancer detection and prediction, and researchers that integrated multi-modal data, set a different and unique approach to the cancer prediction. [4]

Transfer Learning and pre-trained models are yet, another different approach where pre-trained ML models are used in the prostate cancer prediction. Leveraging knowledge from pre-existing models (for example ImageNet), can be adapted to medical tasks and improve the performance of the model. This model is often used with images and libraries such as opencv are used to advance the results obtained.

My pathway

The importance of comparative studies to identify the strengths and weaknesses of different models is crucial in this case because someone's life is affected in such cases. In my pathway of researching, I am going to rely to researches that have tried many different ML-models, but other techniques used in prostate cancer detection should also be taken into account in order this study to be more objective and competitive in comparison to other studies.

Challenges and Future Directions: the current challenges in ML-based prostate cancer detection, including dealing with data heterogeneity, class imbalance, and generalizability are most frequent. By presenting potential future directions and areas for improvement, integrating multi-omics data, exploring novel ML techniques, and validating models on larger and more diverse datasets, for instance, can help overcome or diminish the challenges that are present. Worth noting is that the prostate cancer itself is a difficult task to predict by nature, so there is no expectation to have a 100% reliable model of any of the mentioned algorithms. [5]

3. Project overview

3.1 Motivation

Many men worldwide spend their life being unaware they have prostate cancer, and as mentioned before, it is the second cancer that affects men after lung cancer. With my study I want to motivate men to do regular check-ups and take care of their body and wellbeing.

The driving force behind this project lies in the aspiration to contribute to the advancement of medical science and improve the lives of prostate cancer patients worldwide. By exploring the performance of various ML algorithms, I seek to identify the most effective approach for accurate and reliable prostate cancer detection, ultimately paving the way for a future of more effective and personalized prostate cancer care.

3.2 Impact on health

Accurate and reliable ML-based prostate cancer detection models can facilitate early diagnosis of the disease. Early detection allows for timely initiation of treatment, which can lead to better treatment outcomes and increased survival rates. Detecting prostate cancer at an early stage when it is localized and more treatable can potentially reduce mortality and improve overall patient health.

3.3 Limitations

Numerous limitations of the existing literature are also described, including biases in model validation, heterogeneity in reporting of performance metrics, and lack of sufficient evidence of clinical translation.

3.3.1 Data Availability and Quality:

Access to high-quality and comprehensive datasets for training and evaluating ML algorithms can be challenging. Limited data availability or small sample sizes may affect the generalizability of the results. Additionally, data quality issues such as missing values or noisy data may impact the performance of the ML models.

Class Imbalance: Prostate cancer detection datasets often suffer from class imbalance, where the number of malignant (cancerous) cases is significantly smaller than the number of benign (non-cancerous) cases. Class imbalance can lead to biased model performance, favoring the majority class and affecting the accuracy of cancer detection.

Lack of Longitudinal Data: Longitudinal data, which tracks changes in patients' health over time, can provide valuable insights into disease progression and treatment outcomes. However, the lack of longitudinal data in many datasets can limit the ability to capture the dynamic nature of prostate cancer and its response to treatments.

Generalization to Diverse Populations: ML algorithms trained on specific datasets may lack generalizability to populations with different demographics, ethnicities, or geographic locations. The models may exhibit bias or perform differently on data from diverse patient populations.

Overfitting: ML models that are excessively complex or trained on limited data may suffer from overfitting. Overfitting occurs when a model performs well on the training data but fails to generalize to new, unseen data. Addressing overfitting is crucial to ensure the reliability and robustness of the ML algorithms.

Interpretability: Many advanced ML models, such as deep learning neural networks, lack interpretability, making it challenging to understand how they arrive at their predictions. Interpretable models are crucial in the medical domain, where decisions need to be explainable and transparent.

Limitations of ML Algorithms: Different ML algorithms have specific strengths and weaknesses, and no single approach may be optimal for all scenarios. Selecting the most appropriate algorithm for a given task requires careful consideration and experimentation.

Validation and Clinical Implementation: While ML models may show promising results in research settings, their real-world implementation in clinical practice requires rigorous validation and regulatory approval. Transitioning from research findings to practical clinical tools may present additional challenges.

3.4. Project requirements

In order to fulfil the project's needs and requirements, several guidelines are being followed throughout this project paper. Some of them include the following:

- Data Collection: Obtain a comprehensive dataset of prostate cancer cases and healthy controls. The dataset should include relevant clinical, imaging, and genetic information to facilitate ML model training and evaluation.
- Literature Review: Conduct a thorough literature review to understand the state of the art in ML-based prostate cancer detection. Identify relevant studies, ML algorithms, evaluation metrics, and challenges in the field.
- ML Algorithm Selection: Choose a set of diverse ML algorithms suitable for prostate cancer detection, such as logistic regression, random forest, support vector machines, and deep learning models (e.g., CNNs).

- Data Preprocessing: Preprocess the dataset to handle missing values, perform feature engineering, and ensure compatibility with the selected ML algorithms. Implement techniques like feature scaling and dimensionality reduction as needed.
- Model Training and Evaluation: Implement and train the selected ML algorithms on the preprocessed dataset. Use appropriate evaluation metrics (e.g., accuracy, sensitivity, specificity, precision, recall) to assess the performance of each algorithm.
- Comparative Analysis: Conduct a comparative analysis of the ML algorithms to identify the most effective approach for prostate cancer detection. Compare their strengths, weaknesses, and overall performance.
- Interpretability: Consider the interpretability of the ML models, especially when using complex algorithms like deep learning. Explore methods to explain the model's predictions, especially in a medical context.
- Generalization: Ensure that the ML models generalize well to new, unseen data beyond the training dataset. Use techniques like cross-validation and testing on an independent dataset to validate the models' performance.
- Ethical Considerations: Address ethical and privacy concerns related to handling patient data. Ensure compliance with relevant ethical guidelines and data protection regulations.
- Impact Assessment: Evaluate the potential impact of the ML-based prostate cancer detection models on clinical practice, patient outcomes, and healthcare resources.
- Reporting and Documentation: Prepare a well-structured research paper detailing the methodology, experimental results, analysis, and discussion of findings. Include clear visualizations and tables to present the results effectively.
- Future Directions: Discuss the limitations of the research and propose potential future directions for improving ML-based prostate cancer detection, such as integrating multi-omics data or exploring advanced ML techniques.
- Collaboration and Consultation: Collaborate with domain experts, clinicians, and researchers in the field of oncology and ML to gather valuable insights and ensure the research is clinically relevant.

3.5. Project setup

After some research and comparison, for the purposes of this project, I decided to work with a web IDE – Google Colab, since it turned out it would fit great for this type of a project.

“Google Colaboratory”, popularly known as “Colab”, is a web IDE for python that was released by Google in 2017. “Colab” is an excellent tool for data scientists to execute Machine Learning and Deep Learning projects with cloud storage capabilities.
Colab is basically a cloud-based Jupyter notebook environment that requires no setup. What’s more, it provides its users free access to high compute resources such as GPUs and TPUs that are essential to training models quickly and more efficiently.

Its GPU runtime comes with Intel Xeon CPU @2.20 GHz, 13 GB RAM, Tesla K80 accelerator, and 12 GB GDDR5 VRAM.

The TPU runtime consists of an Intel Xeon CPU @2.30 GHz, 13 GB RAM, and a cloud TPU with 180 teraflops of computational power.

3.6 Analysis (performance analysis)

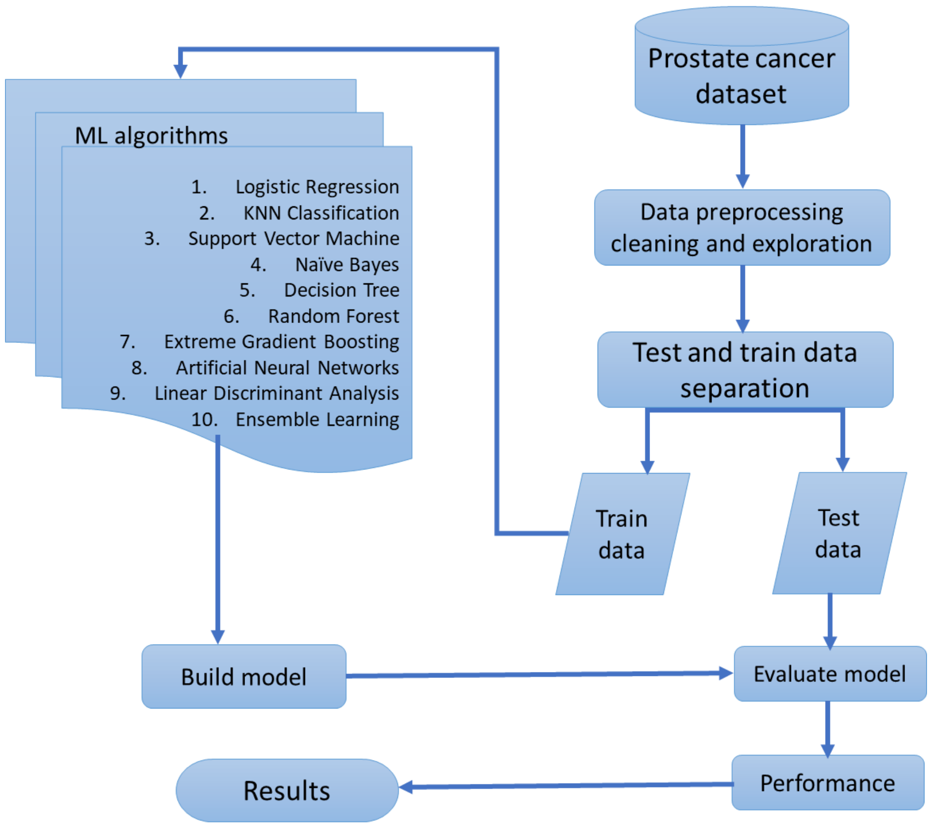


Figure 4 – General performance analysis

The design flow diagram of this study, which was carried out for the prediction of prostate cancer through various supervised machine learning methods, is given in Figure 4.

After obtaining the dataset online, we move to the next step which is preprocessing of the data; a phase which deals with cleaning, reshaping and scaling the data.

After the raw data is cleaned, we divide the data into two sets: training and testing set. We then, train the data using miscellaneous Machine Learning algorithms then we test the data based on the training.

That is how we evaluate the data and develop an evaluation model. Finally, we examine the performance of the algorithms and observe which one did give the best result, least errors, and in order to make a decision, we will create various visualizations and use comparative techniques.

4. Methods used – general definition of the used algorithms

The methods employed in this study encompass a series of systematic steps to address the research objectives. We began by acquiring a comprehensive dataset comprising clinical, imaging, and genetic information of prostate cancer cases and healthy controls. This dataset forms the foundation for training and evaluating the ML models.

First of all, let's take a dive into the algorithms that are being used to predict the type of cancer in prostate.

- Logistic Regression Classification
- KNN Classification
- Support Vector Machine (SVM) Classification
- Naive Bayes Classification
- Decision Tree Classification
- Random Forest Classification
- XGB Classifier
- Artificial Neural Network
- Linear Discriminant Analysis
- Ensemble learning (Voting Classifier)

4.1 Logistic Regression Classification

Early in the 20th century, the biological sciences began to employ logistic regression. Then, it was put to many different social sciences uses. When the dependent variable (target) is categorical, logistic regression is utilized.

For example,

- To predict whether an email is spam (1) or not (0)
- Whether the tumor is malignant (1) or not (0) [as in our case]

This would be the case where we are predicting whether or not a patient is suffering from a malignant or benign tumor, using information on their prostate parameters, age, smoothness, and other continuous variables. [6]

Working Principle:

The fundamental idea behind logistic regression is to model the relationship between the input features and the probability of belonging to the positive class using a logistic (sigmoid) function. The logistic function maps any real-valued input to a value between 0 and 1, which can be interpreted as the probability of the positive class. The logistic function is defined as:

$$P(y=1)=\frac{1}{1+e^{-z}} \qquad (1)$$

where:

- $P(y = 1)$ is the probability of the positive class (having prostate cancer).
- e is the base of the natural logarithm (approximately 2.71828).
- z is the linear combination (2) of input features and their corresponding weights.

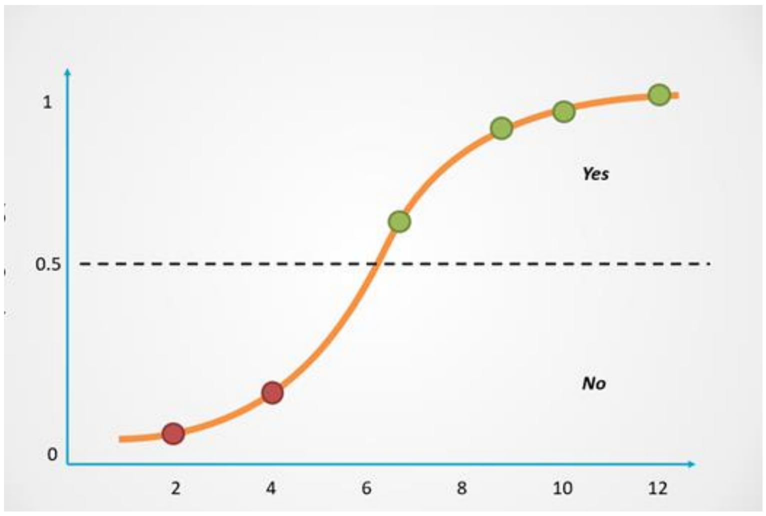


Figure 5 – Graphical representation of Logistic Regression algorithm

Source: www.natasshaselvaraj.com

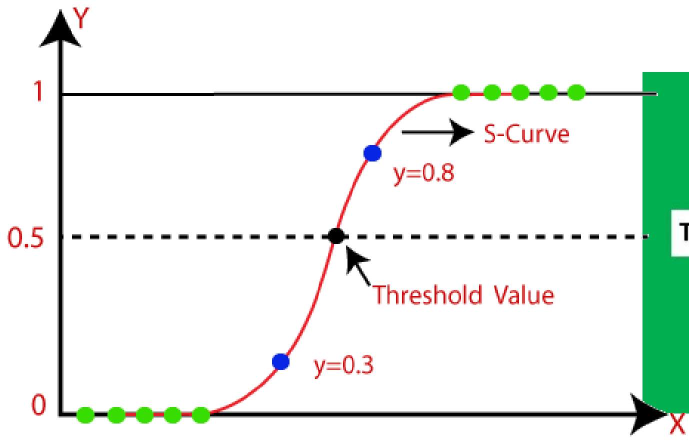
The linear combination z is calculated as:

$z=\beta0+\beta1x1+\beta2x2+\ldots+\beta nxn$ (2)

where:

- β0 is the intercept (bias term).
- β1,β2,…,βn are the coefficients (weights) corresponding to the input features x1 ,x2,…,xn.

Logistic regression can be a useful method for determining the chance of prostate cancer based on clinical, genetic, or imaging information in the context of prostate cancer detection. It is a popular option because of its readability and simplicity, especially when trying to understand the connections between the input data and the existence of prostate cancer. The use of more sophisticated ML methods, such as random forests or deep learning models, can be more effectively used to handle the data's very complicated and non-linear interactions.



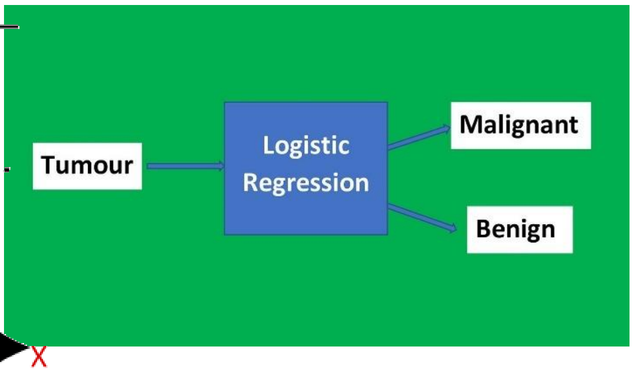


Figure 6 – “S” curve of Logistic Regression

Figure 7 – Output diagram for Logistic Regression

Source: javatpoint.com/logistic-regression-in-machine-learning

4.2 K-nearest neighbor (KNN)

The grouping method proposed by Cover and Hart, in which the group containing the sample data point and the closest neighbor to this data point are determined according to the value of k, is called the K-NN algorithm. K-NN is a supervised learning algorithm that solves the grouping problem. [7]

A straightforward and understandable machine learning technique called K-Nearest Neighbors (KNN) is employed for both classification and regression problems. KNN may be used as a classification algorithm to determine if a patient has prostate cancer (positive class) or does not have prostate cancer (negative class) based on a collection of input features in the context of prostate cancer detection.

Working concept: The proximity-based categorization concept underlies the operation of the KNN algorithm. The technique locates the K closest data points from the training dataset based on a distance measure (for example, Euclidean distance) given a new data point (sample) with an unknown class. The majority class among the new data point's K closest neighbors then determines its class. The new data point is labeled as positive, for instance, if K=5 and three of the five closest neighbors are in the positive category (prostate cancer patients).

KNN's benefits and drawbacks

Advantages: KNN is simple to comprehend, use, and interpret. Due to the lack of complicated model training requirements, it is computationally effective for small to medium-sized datasets. Furthermore, KNN can adjust to non-linear decision limits and is resilient to noisy data.

Cons: Because KNN must determine the distances between each data point, it can be computationally costly for huge datasets. The choice of K and the distance measure can have an impact on how well the algorithm performs. The "curse of dimensionality" is a phenomena that may cause KNN to underperform on high-dimensional data.

When the correlations between the input characteristics and the target variable are locally smooth and non-linear, KNN can be useful in the context of the identification of prostate cancer. However, the existence of irrelevant or noisy features may restrict its performance, highlighting the need of feature selection and data pretreatment when employing KNN.

Steps in KNN

1. Determine the value of K
2. Calculate distances
3. Select K-nearest neighbors
4. Majority voting

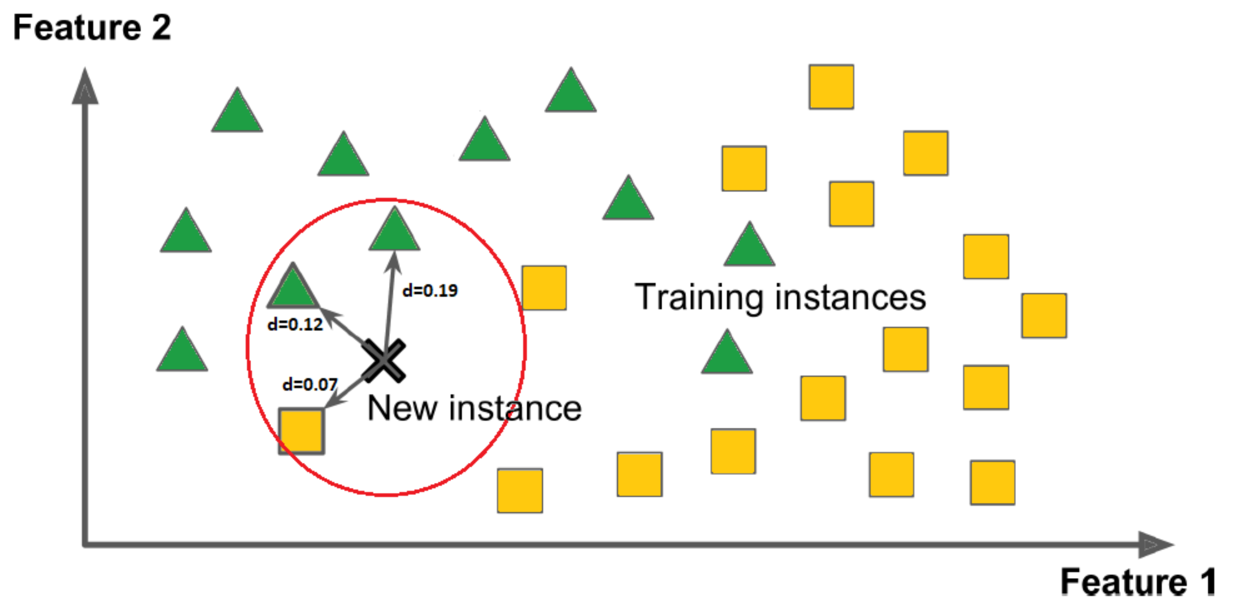


Figure 8 – Graphical representation of K-NN algorithm. Source: mlarchive.com/machine-learning/k-nearest-neighbor-knn-explained/

4.3 Support Vector Machine (SVM) Classification

A powerful and versatile machine learning algorithm that is mainly used for classification and regression tasks. When referring to prostate cancer prediction, SVM could be utilized as a classification algorithm which can predict whether one patient has a malignant tumor (cancerous), or benign (not cancerous). SVM does all this based on a set of input features

Working Principle: The SVM algorithm looks for the optimum hyperplane in the feature space to divide data points into distinct classes. The hyperplane is a decision border that optimizes the margin between the two classes in a binary classification situation. Support vectors, which are the data points nearest to the decision border, are very important in determining the decision boundary.

The fundamental objective of SVM is to maximize the margin between the support vectors and the decision border while achieving the best separation between the classes. The margin maximization characteristic enhances the generalization performance of the model on unobserved data.

Advantages and disadvantages

Advantages: SVM works well in high-dimensional spaces and can process datasets with few training samples. It is adaptable and may be enhanced using the kernel method to handle non-linearly separable data. SVM is less prone to overfitting and has solid theoretical underpinnings.

Disadvantages: When utilizing non-linear kernels and huge datasets, SVM training time can be rather lengthy. It might be difficult to select the right kernel and adjust the hyperparameters. Compared to simpler methods like logistic regression, it might be more difficult to interpret the decision boundaries and model predictions.

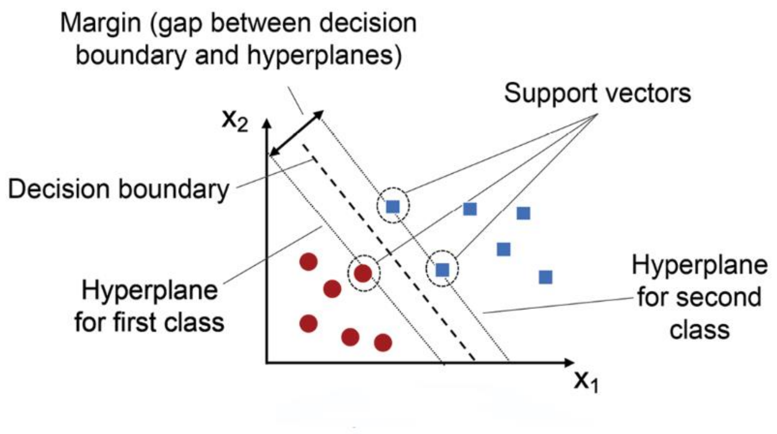

Margin (gap between decision boundary and hyperplanes)
Support vectors
x_2
Decision boundary
Hyperplane for first class
Hyperplane for second class
x_1

Figure 9 – The hyperplane of SVM. Source: vitalflux.com/classification-model-svm-classifier-python-example/

In reference to PC, SVM can be an intuitively a valuable tool for identifying complex and non-linear relationships between input features and the presence of prostate cancer. It has the ability to establish decision thresholds that effectively distinguish between malignant and non-cancerous situations, enhancing forecast precision. To get the best results, however, SVM's effectiveness is dependent on good data preparation, feature selection, and hyperparameter optimization.

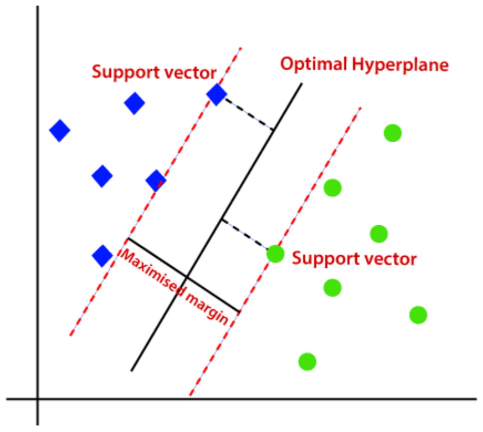


Figure 10 – Graphical representation of SVM

Source: javatpoint.com/machine-learning-support-vector-machine-algorithm

4.4 NaïveBayes Classification

Another simple but popular ML algorithm used for classification purposes, is the Naïve Bayes Algorithm. This algorithm is based on the Bayes' theorem, being especially well-suited for classification of text and problems regarding high-dimensional data. In the field of prostate cancer prediction, the algorithm works as follows:

Working principle

The algorithm is based on the assumption of conditional independence between the features given in the class label. Hence the term "naïve", as it assumes that all features are independent of each other once the class label is known. No matter this assumption may not hold true for every dataset, Naïve Bayes can perform surprisingly well with categorical data anyway.

The core of the algorithm is based on the formula (3):

$$P(y|x_1, x_2, \ldots, x_n) = \frac{P(\, x_1, x_2, \ldots, x_n|y) * P(y)}{P(x_1, x_2, \ldots, x_n)} \quad (3)$$

where:

- P(y|x1, x2,…,xn) is the posterior probability of class y given the features *x*1 ,*x*2,…,*xn*.
- P(x1,x2,…,xn|y) is the likelihood of observing the features given class y.
- P(y) is the prior probability of class y.
- P(x1,x2,…,xn) is the evidence, the probability of observing the features.

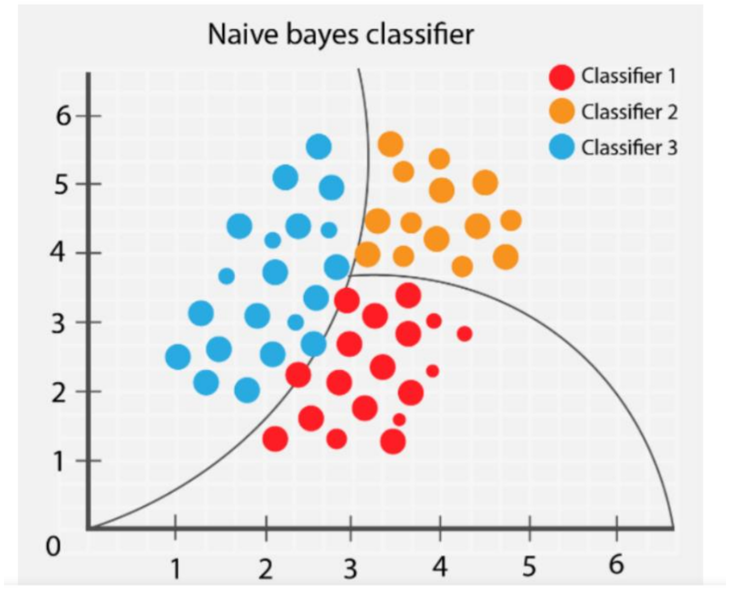


Figure 11 – Graphical representation of Naïve Bayes algorithm. Source: analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/

It can be said that NB is particularly better than other approaches in different cases for especially medical datasets. [8]

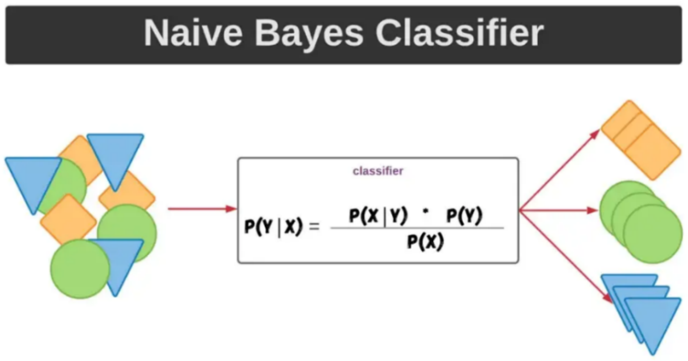


Figure 12 – Output diagram for Naïve Bayes algorithm. Source: thinkinfi.com/naive-bayes-algorithm-in-machine-learning-with-python/

4.5 Decision Tree Classification

The supervised learning class of algorithms, which includes the Decision Tree (DT) method, is mostly used to solve classification issues. It comprises of inner nodes that reflect the branch structures, dataset, indicating the algorithm's decision, and each leaf node that represents a result. Two nodes are present: the decision node, which is used to make decisions and contains a variety of branches, and the leaf node, which is the result of decision nodes and does not have any more branches. Due of the form resemblance, it shares a name with a tree.

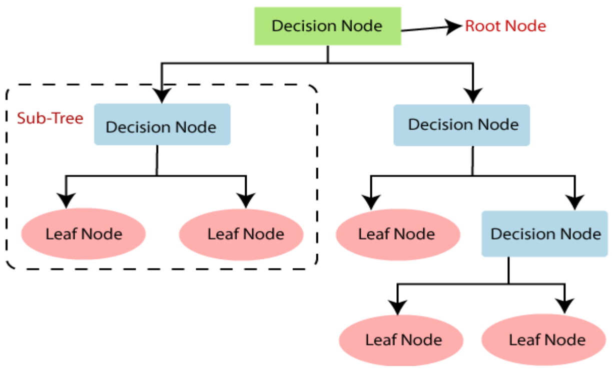


Figure 13 – Representation of Decision Tree algorithm with the root and the nodes. Source: javatpoint.com/machine-learning-decision-tree-classification-algorithm

It has a tree-like structure since the root node serves as a beginning point and eventually spreads to multiple branches. If the response to the query is yes or no, the decision tree simply branches into sub-trees.

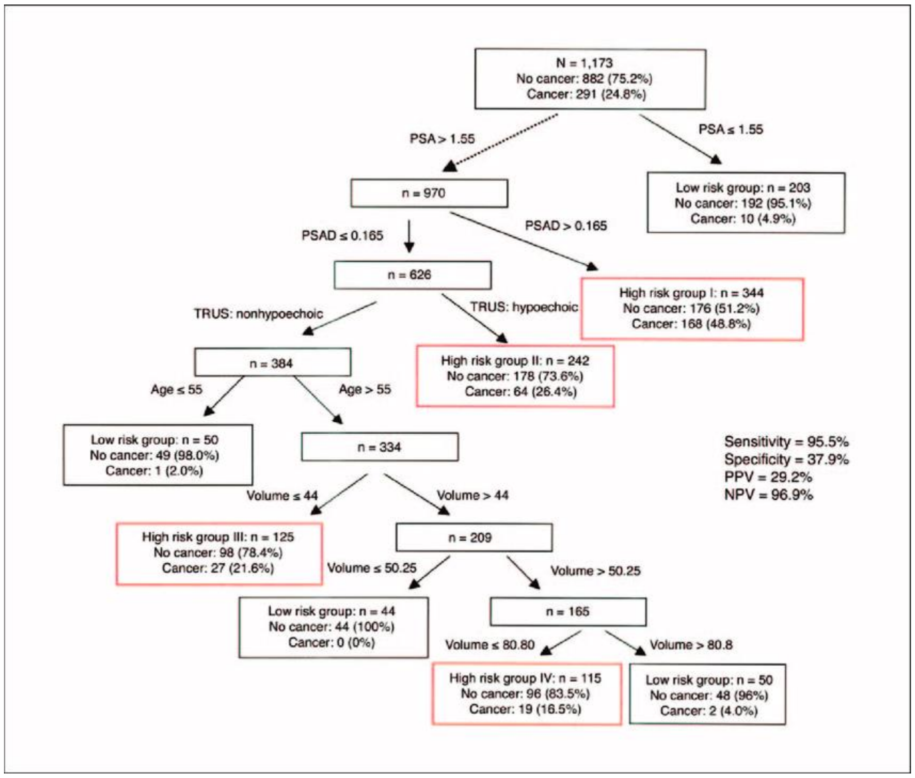


Figure 14 – Decision Tree for prostate cancer example. Source: researchgate.com

To address overfitting, ensemble methods like Random Forest and Gradient Boosting can be used. These methods create multiple Decision Trees and combine their predictions to achieve better generalization performance.

In the context of prostate cancer detection, Decision Tree Classification can provide insights into the relevant features and their importance in predicting the presence of prostate cancer. However, careful pruning and regularization techniques are necessary to mitigate overfitting and ensure the generalization of Decision Trees to new data.

4.6. Random Forest Classification

Random forest (RF) is a new algorithm that integrates multiple trees based on the idea of ensemble learning, and it can handle thousands of input variables without reduction and evaluate the importance of input variables.

It is good at processing non-parametric and high-dimensional data, and has a solid anti-overfitting ability and a high computational efficiency. It has been widely used in biology, medicine, remote sensing, and other fields. [9]

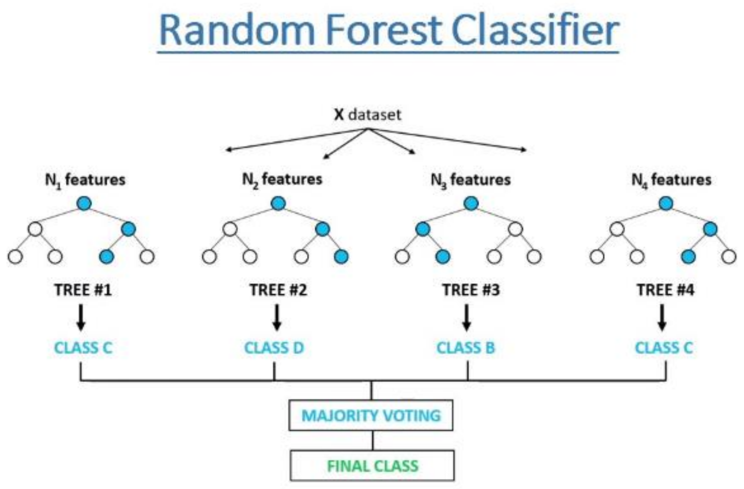


Figure 15 – How random forest classifier work.

Source: medium.com/analytics-vidhya/random-forest-classifier-and-its-hyperparameters-8467bec755f6

The Random Forest algorithm involves the following steps:

Bootstrapped Sampling: A random subset (sample with replacement) of the training data is chosen for each decision tree. Multiple datasets are produced in order to build different trees.

Feature subsetting: A random subset of features is taken into account for splitting at each node of each tree. This enhances the variability of their forecasts and decorrelates the trees.

Tree Growth: Each decision tree is developed using a method akin to conventional decision tree growth, where nodes are divided according to a chosen criterion (for example, Gini impurity or entropy).

Aggregation: Each decision tree in the forest individually predicts the class label for a brand-new data point during prediction. In order to get the final forecast, the individual tree projections are then combined, frequently using majority vote.

Among the benefits of using Random Forest Classification is the overfitting addressing that is commonly found in individual decision trees by averaging out their predictions.

However, RF training process can be computationally more expensive due to construction of multiple trees.

By utilizing the strength of several decision trees, Random Forest Classification can offer reliable and accurate predictions in the context of prostate cancer diagnosis. It is appropriate for categorical and numerical data, and it is especially helpful when working with high-dimensional datasets that need for careful feature selection and dimensionality reduction. Random Forest gives a dependable method to increase the accuracy of prostate cancer prediction models by combining the predictions of many trees.

4.7 XGB Classifier

XGBoost (Extreme Gradient Boosting) is an advanced and highly effective machine learning algorithm known for its exceptional predictive performance in various classification and regression tasks. XGBoost is particularly popular because it's so fast, and that speed comes at no cost to accuracy.

Regularization is a feature of XGBoost that enables you to prevent overfitting by imposing L1/L2 penalties on each tree's weights and biases. Many other gradient boosting solutions do not provide this functionality.

Using the weighted quantile sketch technique, XGBoost also has the capacity to handle sparse data sets. While maintaining the same level of computational complexity as previous techniques like stochastic gradient descent, this algorithm enables us to cope with feature matrices that include non-zero elements.

Additionally, XGBoost provides a block structure for concurrent learning. It makes scaling up on multicore computers or clusters simple. Additionally, it makes advantage of cache awareness, which lowers memory consumption while training models using sizable datasets.

Last but not least, XGBoost provides out-of-core computing capability by leveraging disk-based data structures rather than in-memory ones while doing computations.

XGBoost has already been applied alone in many bioinformatics and biomedical applications while the use of metaheuristics for optimizing their hyperparameters has also been successfully performed in many other applications. [10]

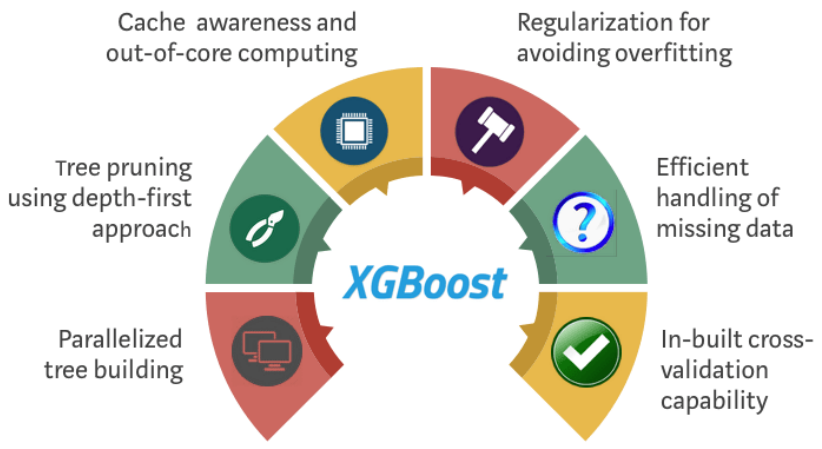

Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

Figure 16 – Principles of XGB classifier. Source: [linkedin.com/pulse/xgboost-classifier-algorithm-machine-learning-kavya-kumar]()

CPU-powered machine learning tasks with XGBoost can literally take hours to run. That's because creating highly accurate, state-of-the-art prediction results involves the creation of thousands of decision trees and the testing of large numbers of parameter combinations. Graphics processing units, or GPUs, with their massively parallel architecture consisting of thousands of small efficient cores, can launch thousands of parallel threads simultaneously to supercharge compute-intensive tasks.

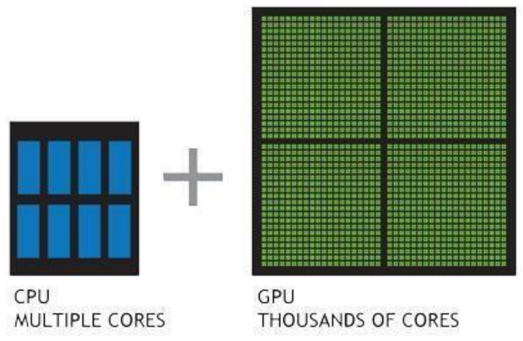


Figure 17 – How GPU makes the computation faster.

Source: [nvidia.com/en-us/glossary/data-science/xgboost/]()

XGB's gradient boosting approach leverages the strength of multiple weak learners to collectively improve predictive accuracy. The iterative nature of boosting focuses on correcting errors made by previous models, resulting in a highly adaptive and accurate ensemble.

In the field of ML prostate cancer detection, XGB Classifier can provide exceptional accuracy and predictive power. It is particularly useful when striving for the highest predictive performance and when dealing with complex and high-dimensional datasets. However, due to its complexity, adequate hyperparameter tuning and cross-validation are essential to ensure optimal results.

4.8 Artificial Neural Network

An Artificial Neural Network (ANN) is a complex machine learning model inspired by the structure and functioning of the human brain. ANNs are highly versatile and can be applied to a wide range of tasks, including classification, regression, and pattern recognition.

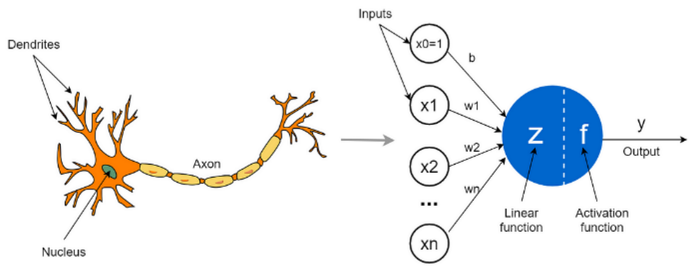


Figure 18 – Analogy of the neurons and the neural networks. Source: towardsdatascience.com

In artificial neural networks, the accurate updating of weights is crucial to the learning process. After calculating the error, the weights are updated according to the obtained error. [11]

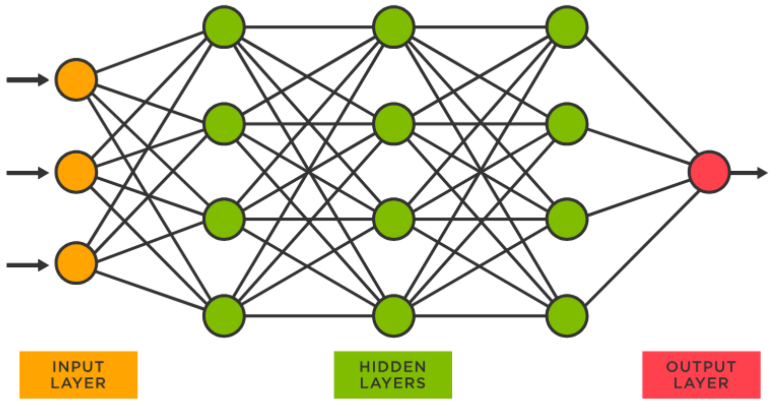
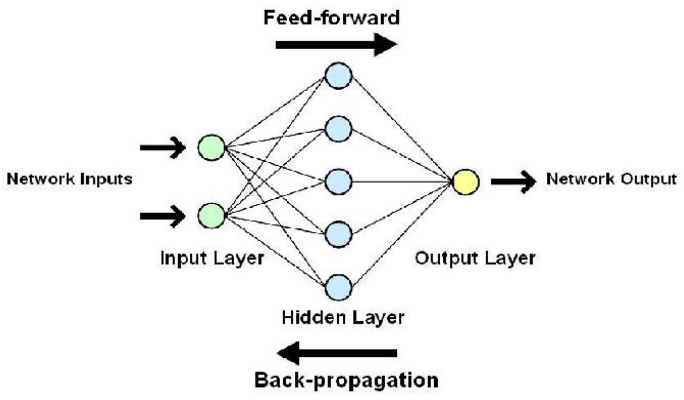


Figure 19 – Graphical representation of ANN. Source: tibco.com/reference-center/what-is-a-neural-network



The training process involves the steps:

1. Initialization: Use random weights and a suitable architecture for the model (number of layers and neurons per layer).

Figure 20 – The training process in ANN Source: analyticsvidhya.com

2. Feedforward pass: It should be used to generate predictions for the supplied data

3. Calculate Loss: Using a suitable loss function (e.g., mean squared error for regression, cross-entropy for classification), determine the discrepancy between projected outputs and actual objectives.

4. Backpropagation: The error gradient is backpropagated from the input layer to the output layer. To reduce the loss, update the weights using optimization procedures (such as gradient descent).

5. Iterative optimization: Repeat the forward and backward passes over a number of epochs while changing the weights to incrementally lower the prediction error.

Imaging and diagnostics in medicine have been transformed by deep learning, a branch of machine learning that uses deep neural networks. It is very useful for applications like medical image analysis since it excels at feature extraction from pictures and other complicated data sources.

Deep neural networks, in particular, can use their capacity to understand complicated associations to increase classification accuracy in the context of prostate cancer diagnosis. However, because to their complexity, obtaining the best results requires a substantial amount of training data, processing power, and hyperparameter tuning skill.

4.9 Linear Discriminant Analysis

Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.

Linear Discriminant Analysis (LDA) is a supervised learning algorithm used for classification tasks in machine learning. It is a technique used to find a linear combination of features that best separates the classes in a dataset.

In order for LDA to function, the data are projected onto a lower-dimensional space with a maximum degree of class separation. Finding a group of linear discriminants that optimize the proportion of between-class variation to within-class variance is how it does this. In other words, it determines the directions in the feature space that effectively distinguish the various data classes.

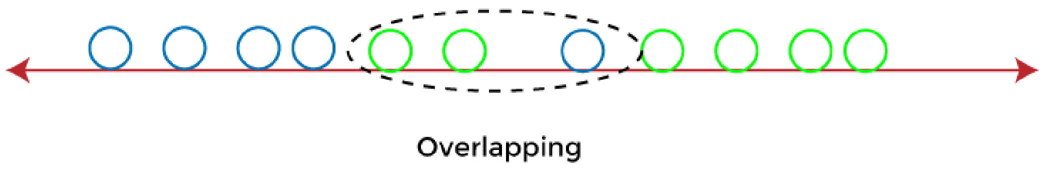
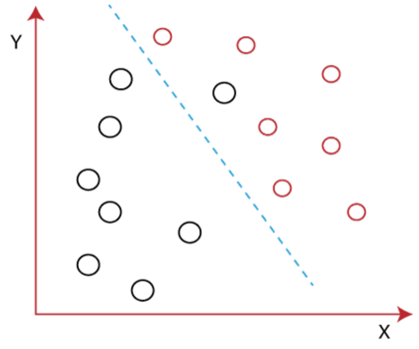


Figure 21 – How LDA overcomes the overlapping issue.
Source: javatpoint.com/linear-discriminant-analysis-in-machine-learning



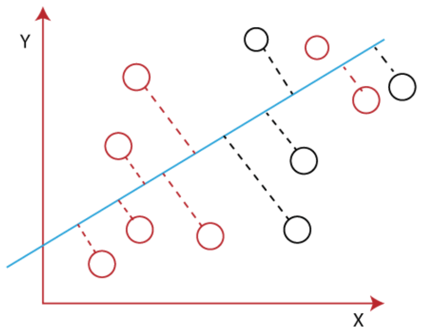
It is impossible to draw a straight line in a 2-d plane that can separate these data points efficiently but using linear Discriminant analysis; we can dimensionally reduce the 2-D plane into the 1-D plane. Using this technique, we can also maximize the separability between multiple classes.

Figure 22 – Impossibility to draw a 2D plane that will effectively separate the two classes. Source: javatpoint.com

To create a new axis, Linear Discriminant Analysis uses the following criteria:

- It maximizes the distance between means of two classes.
- It minimizes the variance within the individual class.

Using the above two conditions, LDA generates a new axis in such a way that it can maximize the distance between the means of the two classes and minimizes the variation within each class.



LDA reduces the number of dimensions to prevent learning and memorization. Therefore, it can provide a high advantage in feature extraction as well as the application as a classification algorithm. [12]

Figure 23 – New axis created by LDA. Source: javatpoint.com

4.10 Ensemble Learning – voting classifier

Ensemble learning in machine learning is a powerful technique that combines the predictions of multiple individual models (base learners) to create a more accurate and robust predictive model. The idea behind ensemble learning is that by aggregating the predictions of diverse models, the collective wisdom of these models can often outperform any single model. Ensemble methods are widely used in both classification and regression tasks. Here's a detailed explanation of ensemble learning:

Key Concepts in Ensemble Learning:

1. Base Learners: These are the individual models used in ensemble learning. Base learners can be of the same type (homogeneous ensemble) or different types (heterogeneous ensemble). They should be trained independently and have some diversity in their predictions.

2. Ensemble Methods: These are techniques used to combine the predictions of base learners to form a final prediction. Common ensemble methods include averaging, voting, bagging, boosting, and stacking.

3. Aggregation: Ensemble methods involve aggregating the predictions of base learners. The way predictions are aggregated depends on the specific ensemble method being used.

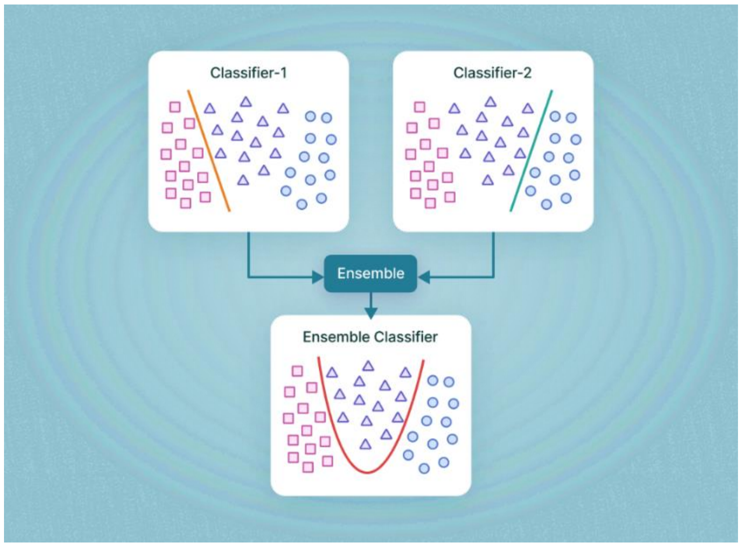


Figure 24 – Basic example of an ensemble learning method. Source: v7labs.com/blog/ensemble-learning

Ensemble learning is a valuable tool in machine learning, as it often leads to improved model performance, better generalization, and increased robustness. It's particularly effective when base learners are diverse, making different types of errors. The diversity enables the ensemble to correct errors and make more accurate predictions.

In this project we will use a voting classifier. A Voting Classifier is an ensemble machine learning model that combines the predictions of multiple base models (individual classifiers) to make a final prediction or decision. It is a type of ensemble learning technique used in both classification and regression tasks. The idea behind a Voting Classifier is to leverage the collective wisdom of multiple models to improve predictive accuracy and robustness.

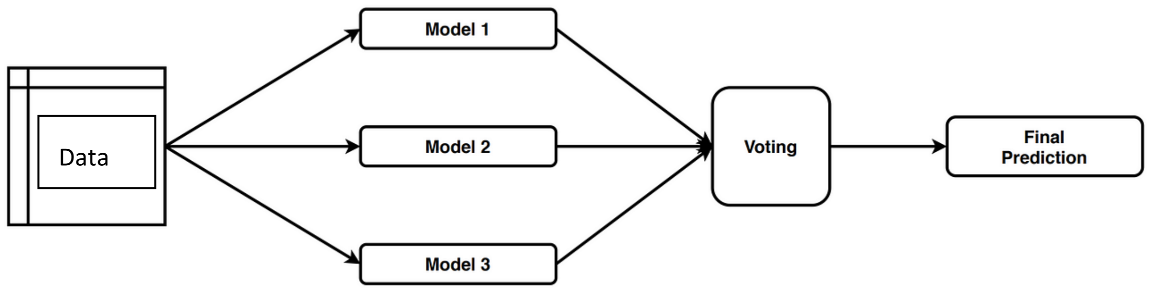


Figure 25: Flowchart of an Ensemble Learning model

5. Analysis of the data

Data analysis involves a series of systematic steps to transform raw data into meaningful insights and actionable conclusions. After we comprehensively discussed our relevant problems and understood the methods we are going to use in this research, our next step is the data mining step.

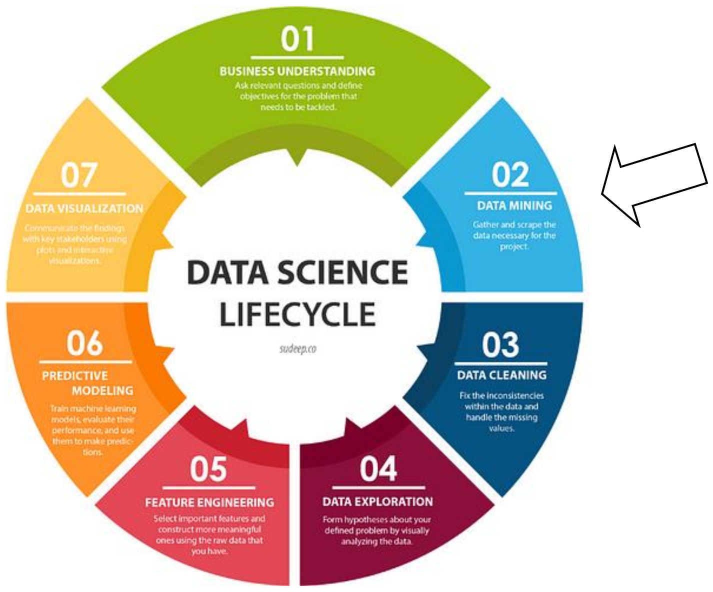

DATA SCIENCE LIFECYCLE
sudeep.co
01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.

Figure 26 – Steps of a data science project

Source: sudeep.co/data-science/2018/02/09/Understanding-the-Data-Science-Lifecycle.html

5.1 Understanding the dataset – description of the dataset

In order to analyze the performance of the methods evaluated in this study, a prostate cancer dataset was used as shown in first step of the design flow. Prostate cancer dataset is an open-access data source that can be obtained through the Kaggle platform. The dataset consists of observations from 100 patients. The dataset consists of eight independent variables (radius, texture, area, perimeter, compactness, smoothness, fractal dimension, symmetry) and one dependent variable (diagnosis result). The variables used as predictors are as follows:

1-Radius,

2-Perimeter,

3-Texture,

4- Smoothness,

5- Area,

6- Compactness,

7-Symmetry,

8-Fractal dimension and

9-Diagnosis result.

The output response takes two values: “B” for benign tumors and “M” for malignant tumors. A statistical analysis containing detailed information about the data is given in Table 1.

	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Symmetry	Fractal_dimension
count	100	100	100	100	100	100	100	100
mean	16.85	18.23	96.78	702.88	0.102730	0.126700	0.193170	0.064690
std	4.87	5.19	23.67	319.71	0.014642	0.061144	0.030785	0.008151
min	9	11	52	202	0.07	0.038	0.135	0.053
max	25	27	172	1878	0.143	0.345	0.304	0.097

Table 1 – Statistical analysis of the dataset

5.2 First operations on the dataset – EDA

Any data analysis or research effort must begin with an exploratory data analysis (EDA). It entails the process of highlighting, displaying, and analyzing the key traits, patterns, and trends found in a dataset graphically and quantitatively. EDA aids researchers in better comprehending the data and locating possible connections that might direct future analysis, modeling, and decision-making. EDA can offer useful insights into the dataset's structure, quality, and relevant predictive aspects while studying data for the diagnosis of prostate cancer.

Importing the necessary libraries:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_selection import VarianceThreshold
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn import preprocessing
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from sklearn import metrics
```

After importing the libraries that are going to be used in this project, the very first step we have do is to import the dataset we are going to work with. After that, we are calling the function *info()*. By printing the *shape* of the dataset, we observe how many columns and rows does this dataset contain.

```
cancer_data = pd.read_csv('Prostate_Cancer.csv')
cancer_data.info()
print('Dataset: ',cancer_data.shape)
```

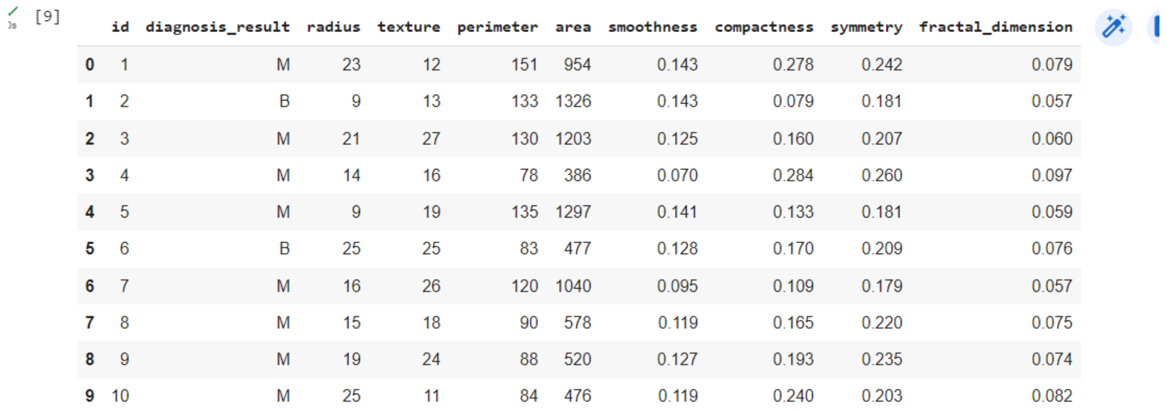
Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 100 non-null    int64
 1   diagnosis_result   100 non-null    object
 2   radius             100 non-null    int64
 3   texture            100 non-null    int64
 4   perimeter          100 non-null    int64
 5   area               100 non-null    int64
 6   smoothness         100 non-null    float64
 7   compactness        100 non-null    float64
 8   symmetry           100 non-null    float64
 9   fractal_dimension  100 non-null    float64
dtypes: float64(4), int64(5), object(1)
memory usage: 7.9+ KB

Dataset:  (100, 10)
```

Also, in order to have a glance of the data, we want to see the first 10 and last 5 rows of the dataset:

```
cancer_data.head(10)
```

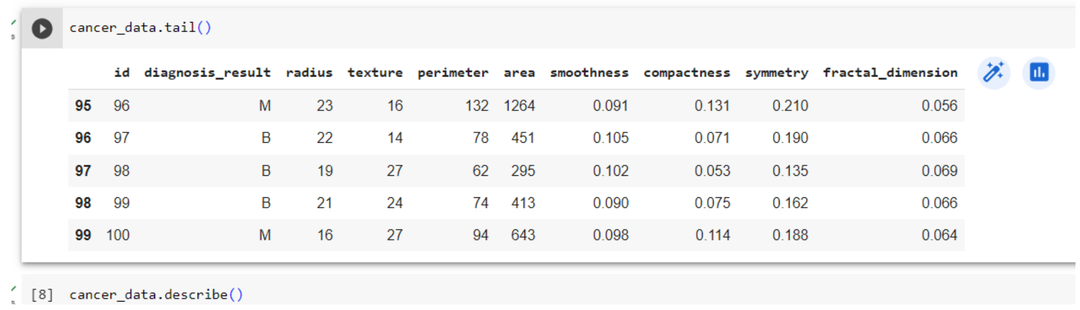


```
[9]
```

	id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
0	1	M	23	12	151	954	0.143	0.278	0.242	0.079
1	2	B	9	13	133	1326	0.143	0.079	0.181	0.057
2	3	M	21	27	130	1203	0.125	0.160	0.207	0.060
3	4	M	14	16	78	386	0.070	0.284	0.260	0.097
4	5	M	9	19	135	1297	0.141	0.133	0.181	0.059
5	6	B	25	25	83	477	0.128	0.170	0.209	0.076
6	7	M	16	26	120	1040	0.095	0.109	0.179	0.057
7	8	M	15	18	90	578	0.119	0.165	0.220	0.075
8	9	M	19	24	88	520	0.127	0.193	0.235	0.074
9	10	M	25	11	84	476	0.119	0.240	0.203	0.082

Table 2 – first 10 observations of the data in the dataset

```
cancer_data.tail()
```



```
cancer_data.tail()
```

	id	diagnosis_result	radius	texture	perimeter	area	smoothness	compactness	symmetry	fractal_dimension
95	96	M	23	16	132	1264	0.091	0.131	0.210	0.056
96	97	B	22	14	78	451	0.105	0.071	0.190	0.066
97	98	B	19	27	62	295	0.102	0.053	0.135	0.069
98	99	B	21	24	74	413	0.090	0.075	0.162	0.066
99	100	M	16	27	94	643	0.098	0.114	0.188	0.064

```
[8] cancer_data.describe()
```

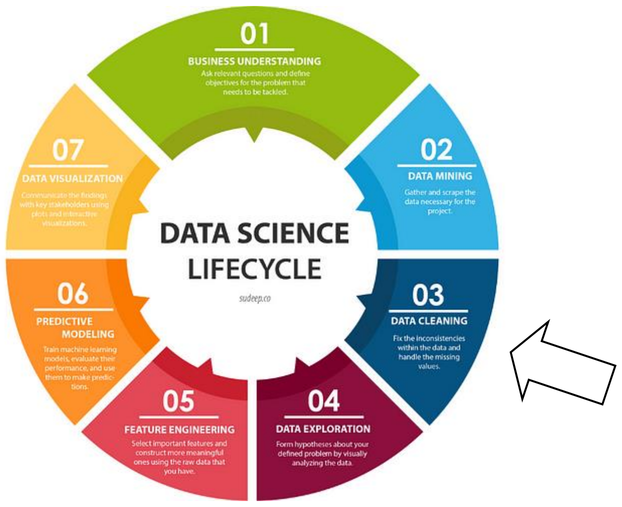
table 3 – last 5 observations of the data in the dataset

5.3 Data Preprocessing – Data Cleaning

Data preprocessing is the phase where raw data is cleaned, transformed, and made ready for analysis. The primary objective is to improve data quality, consistency, and compatibility with the analytical tools and models you plan to use.

Data preprocessing steps include handling missing values, dealing with outliers, standardizing or normalizing data, and converting categorical data into a suitable format (e.g., one-hot encoding). Proper data preprocessing ensures that the data is in a form that can be effectively analyzed without introducing biases.

We can notice that *diagnosis_result* has data type ‘object’. In order for the program to deal with our task, we have to transform the *object* data type into a type that is processable for the program; in this case - integer


DATA SCIENCE LIFECYCLE
sudeep.co
01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.

```
cancer_data.diagnosis_result = [1 if each == 'M' else 0 for each in
cancer_data.diagnosis_result]


cancer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   diagnosis_result   100 non-null    int64  
 1   radius             100 non-null    int64  
 2   texture            100 non-null    int64  
 3   perimeter          100 non-null    int64  
 4   area               100 non-null    int64  
 5   smoothness         100 non-null    float64
 6   compactness        100 non-null    float64
 7   symmetry           100 non-null    float64
 8   fractal_dimension  100 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 7.2 KB
```

```
cancer_data.diagnosis_result.value_counts()
```

```
Output:

1  62

0  38

 Name: diagnosis_result, dtype: int64
```

Plotting the aforementioned data using bar plot:

```
value_counts = cancer_data.diagnosis_result.value_counts()[0:30]

colors = ['blue', 'orange']

value_counts.plot(kind='bar', color=colors)
plt.title('Diagnosis Result Distribution')
plt.xlabel('Diagnosis Result')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

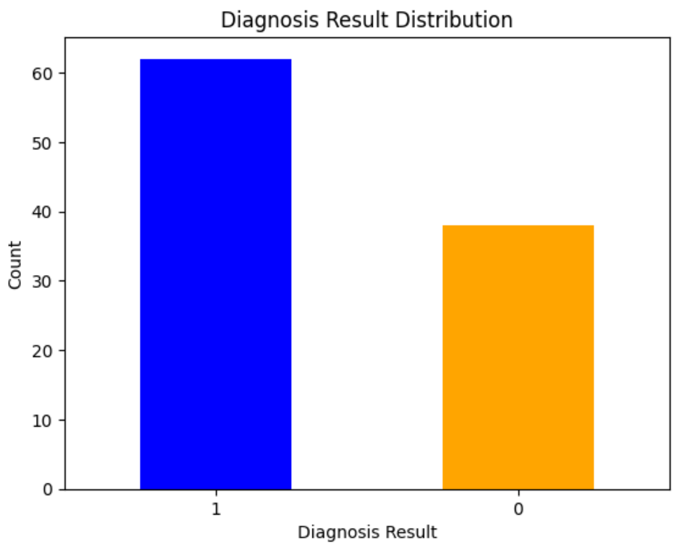
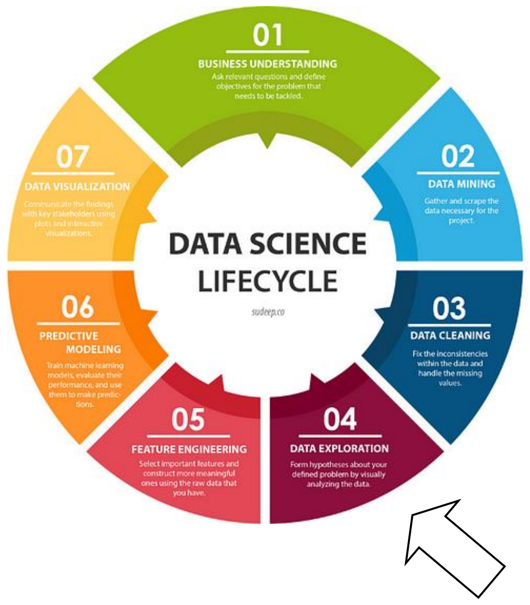


Figure 27 – Number of cases with cancerous tumor (blue) vs number of cases with non-cancerous tumor (yellow)

5.4 Data Exploration

Data exploration is a broader term that encompasses both EDA and data preprocessing.

It refers to the process of investigating data to uncover hidden patterns, trends, and relationships, as well as preparing the data for analysis.


DATA SCIENCE LIFECYCLE
sudeep.co
01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.

Data exploration includes understanding the data's structure, identifying anomalies or inconsistencies, and conducting initial analysis to make informed decisions about data cleaning, transformation, and model building.

Essentially, data exploration involves both understanding the data through visualization and summary statistics (EDA) and preparing it for analysis (data preprocessing).

```
df = cancer_data[['radius','texture','perimeter','area',
'smoothness','diagnosis_result']]
cor = df.corr()
sns.heatmap(cor, square = True)
```

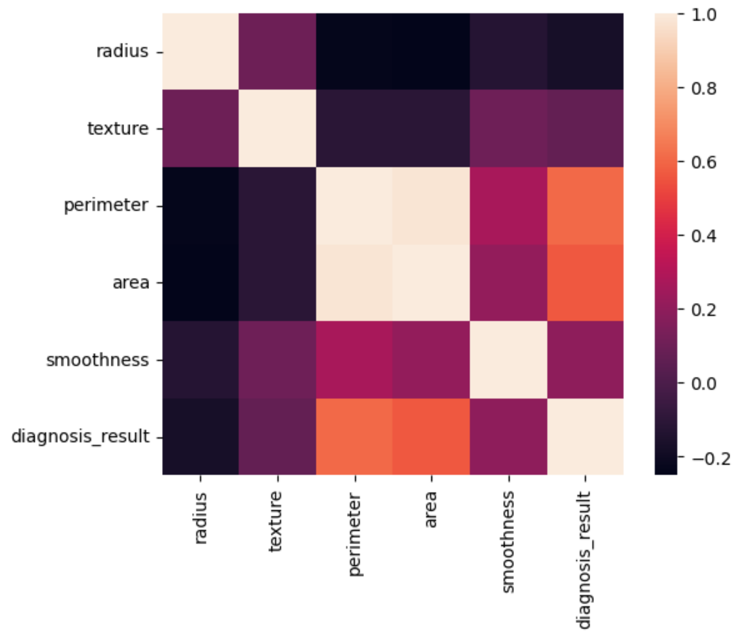


Figure 28 – Heatmap data of correlation

In the code and figure above, we are taking a look in the heatmap of correlation between the variables. In essence, the statistical concept of correlation indicates how much two variables change together. It measures the magnitude and direction of the linear connection between two variables, to put it another way. In statistics and data analysis, correlation is a fundamental concept that explains how changes in one variable may be related to changes in another.

```
sns.set_style("ticks")
pairplot = sns.pairplot(df, hue="diagnosis_result", size=3)
pairplot.add_legend()
plt.show()
```

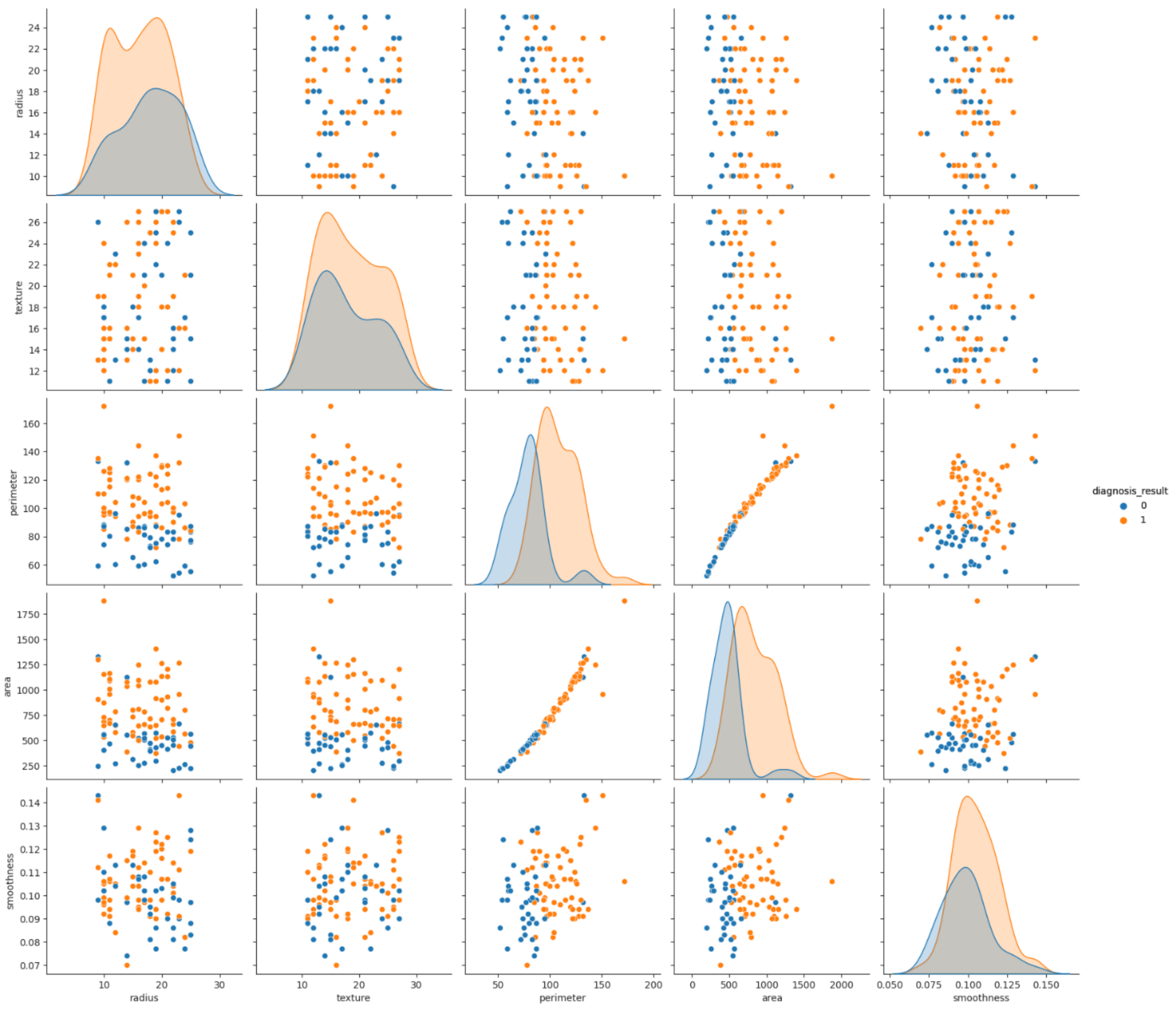
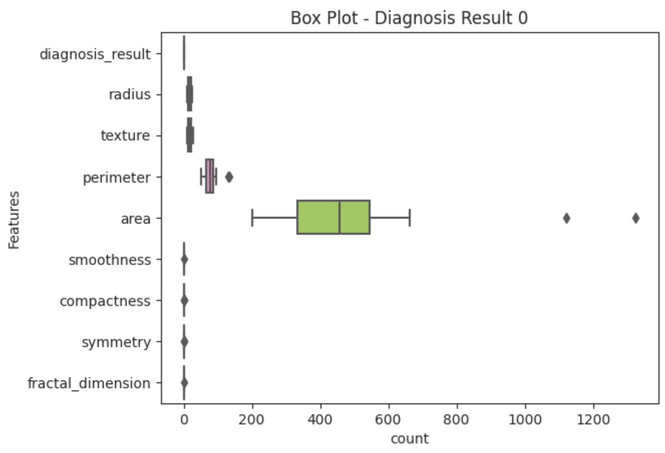


Figure 29 – Matrix of scatter plots

The latest code produces a matrix of scatterplots, where each variable in the dataframe is plotted against every other variable. The color of the data points in each scatterplot is determined by the values in the "diagnosis_result" column. This visualization helps us understand the relationships and distributions between pairs of variables and provides insights into potential patterns based on different diagnosis results.

```
boxplot1 = sns.boxplot(palette="Set2", orient="h",
data=cancer_data[cancer_data.diagnosis_result == 0])
plt.title('Box Plot - Diagnosis Result 0')
plt.xlabel('count')
plt.ylabel('Features')
plt.show()
```



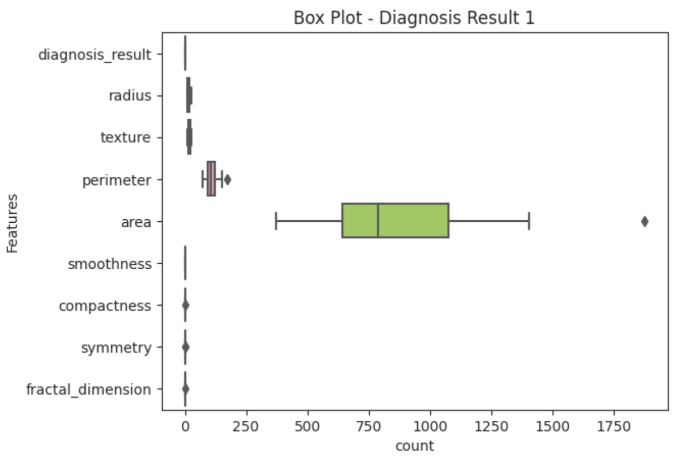


Figure 30 – Box plot of noncancerous tumor

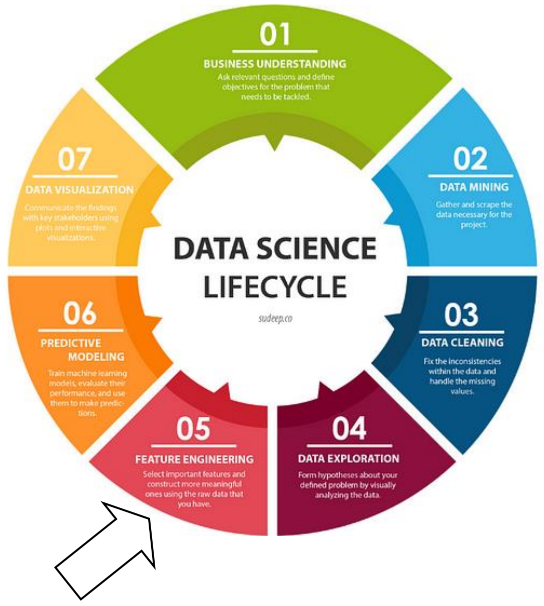
Figure 31 - Box plot of cancerous tumor

```
boxplot2 = sns.boxplot(palette="Set2", orient="h",
data=cancer_data[cancer_data.diagnosis_result == 1])
plt.title('Box Plot - Diagnosis Result 1')
plt.xlabel('count')
plt.ylabel('Features')
plt.show()
```

Using data visualization tool 'boxplot' we are able to observe the outliers in the data, so we can eliminate them later in our data i.e., before introducing the ML models.

5.5 Feature Engineering

In the step of feature engineering, your goal is to create new features or transform existing ones to improve the performance and interpretability of your predictive models. This process requires a combination of domain knowledge, creativity, and a thorough understanding of the dataset.



01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.
DATA SCIENCE LIFECYCLE
sudeep.co

Now, we have to split the data into training and testing set. In data science projects, usually 70% of the data is dedicated to the training set, 15% of the data to the testing set and the rest 15% are dedicated to the validation set. However, we won't use validation data in our project, therefore we will give 20% to the testing data set.

```
x_train, x_test, y_train, y_test=train_test_split(
    cancer_data.drop(['diagnosis_result'], axis=1),
    cancer_data[['diagnosis_result']],
    test_size=0.2,
    random_state=42)
```

This is how the shape of the training and testing dataset now looks like:

```
X train shape:  (80, 8)
Y train shape:  (80, 1)
X test shape:  (20, 8)
Y test shape:  (20, 1)
```

```
for column in x_train.columns:

    df_train1 = x_train[(y_train.diagnosis_result==0) &
(x_train[column]<np.mean(x_train.loc[y_train.diagnosis_result==0,column])+
3*np.std(x_train.loc[y_train.diagnosis_result==0,column]))]
```

```
    df_test1 = x_test[(y_test.diagnosis_result==0) &
(x_test[column]<np.mean(x_train.loc[y_train.diagnosis_result==0,column])+3
*np.std(x_train.loc[y_train.diagnosis_result==0,column]))]

    label_train1 = y_train[(y_train.diagnosis_result==0) &
(x_train[column]<np.mean(x_train.loc[y_train.diagnosis_result==0,column])+
3*np.std(x_train.loc[y_train.diagnosis_result==0,column]))]
    label_test1 = y_test[(y_test.diagnosis_result==0) &
(x_test[column]<np.mean(x_train.loc[y_train.diagnosis_result==0,column])+3
*np.std(x_train.loc[y_train.diagnosis_result==0,column]))]

    df_train2 = x_train[(y_train.diagnosis_result==1) &
(x_train[column]<np.mean(x_train.loc[y_train.diagnosis_result==1,column])+
3*np.std(x_train.loc[y_train.diagnosis_result==1,column]))]
    df_test2 = x_test[(y_test.diagnosis_result==1) &
(x_test[column]<np.mean(x_train.loc[y_train.diagnosis_result==1,column])+3
*np.std(x_train.loc[y_train.diagnosis_result==1,column]))]

    label_train2 = y_train[(y_train.diagnosis_result==1) &
(x_train[column]<np.mean(x_train.loc[y_train.diagnosis_result==1,column])+
3*np.std(x_train.loc[y_train.diagnosis_result==1,column]))]
    label_test2 = y_test[(y_test.diagnosis_result==1) &
(x_test[column]<np.mean(x_train.loc[y_train.diagnosis_result==1,column])+3
*np.std(x_train.loc[y_train.diagnosis_result==1,column]))]
```

In the previous step, we saw that we have some outliers in figure 25 and 26, so we should do removing of the outliers. The code above aims to remove outliers from the training and testing datasets within specific classes, as indicated by the filtering based on the mean and standard deviation of each column for each class. This kind of data preprocessing can help improve the robustness and accuracy of machine learning models by excluding potential outliers that might negatively impact their performance.

First, we loop through the columns; then we filter and partition the data; finally, we label the partitioned data – This is what is done in simple terms.

Also, in the previous step we can notice that something is not really okay with the features *area* and *perimeter.* Namely, they seem to be corelated too much. This kind of data may interfere our model, so it is important to remove it now. In the figure below,

this is more detailed heatmap of the correlation, we can observe that even 99% of the data between those two features are the same.

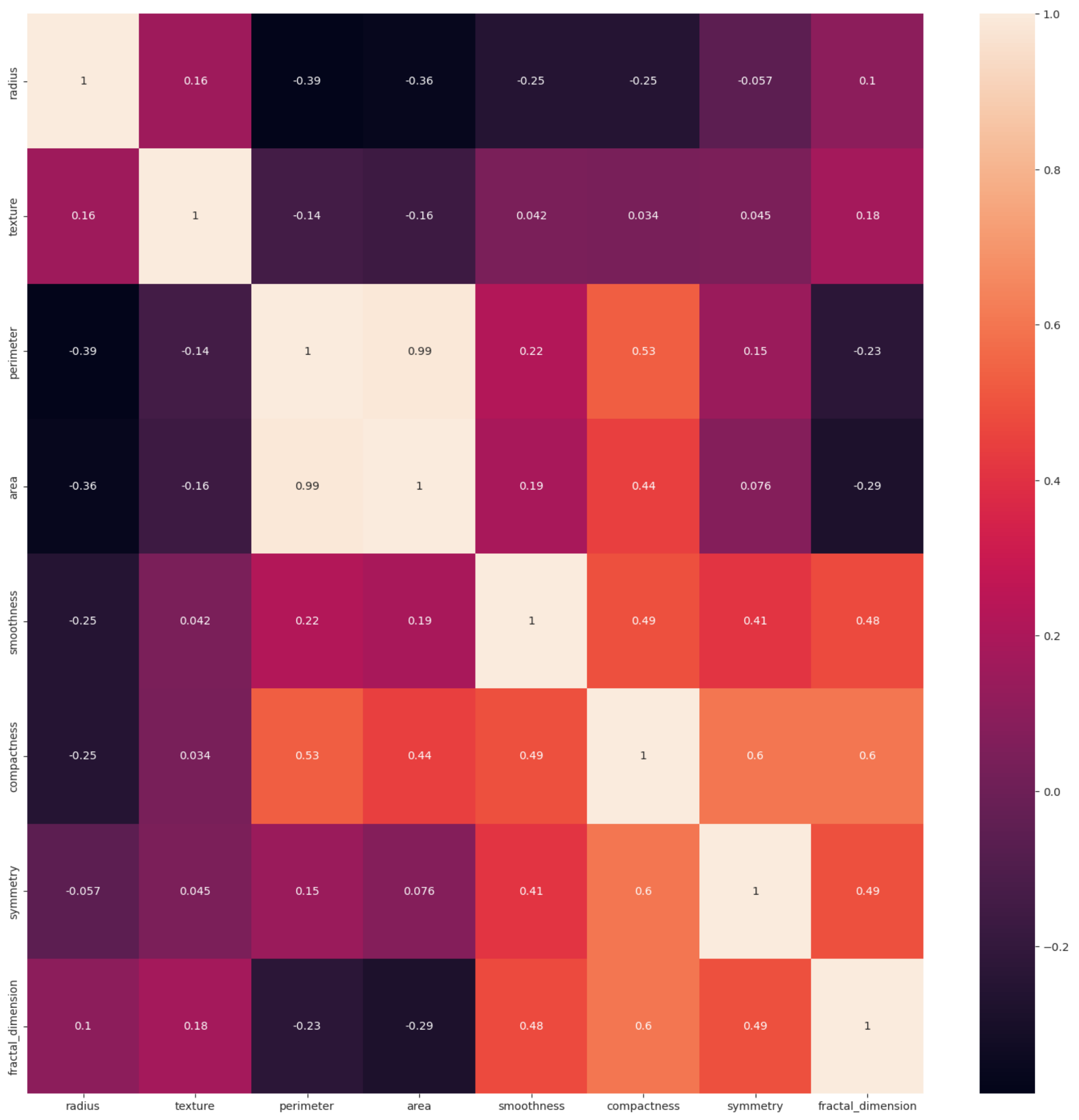


Figure 32 – Heatmap of correlation with values

Feature engineering and machine learning frequently employ the feature selection or dimensionality reduction strategy of removing corelated features. Features that are strongly linearly connected to one another are said to be correlated. While having highly correlated features can occasionally have a detrimental effect on the performance and interpretability of machine learning models, those features can sometimes negatively impact the performance and interpretability of machine learning models.

```
correlated_features = set()
for i in range(len(corrMatrix.columns)):
    for j in range(i):
        if abs(corrMatrix.iloc[i, j]) > 0.7:
            colname = corrMatrix.columns[i]
            correlated_features.add(colname)
print(correlated_features)
```

Output:

```
{'area'}
```

The code above confirms our statement, so we move on cutting off the correlated feature(s):

```
x_train.drop(labels=correlated_features, axis=1, inplace=True)

x_test.drop(labels=correlated_features, axis=1, inplace=True)
```

Scaling the data

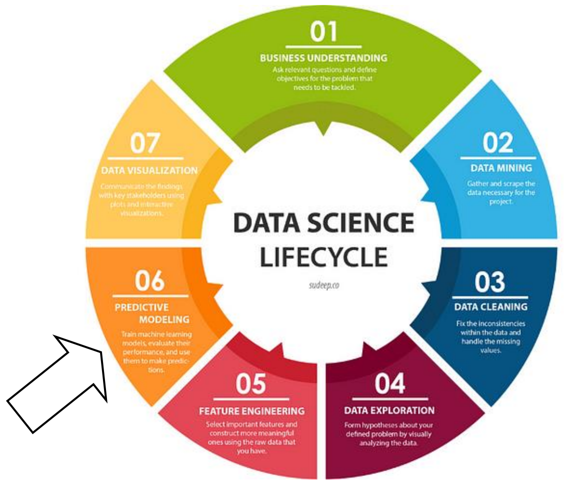
```
mm_scaler = preprocessing.StandardScaler()
x_train = pd.DataFrame(mm_scaler.fit_transform(x_train))
x_test = pd.DataFrame(mm_scaler.transform(x_test))
```

The code snippet scales the features in both the training and testing datasets using the StandardScaler. This step ensures that the features have zero mean and unit variance, which can help improve the performance and convergence of machine learning algorithms. Additionally, the scaling applied to the test data uses the mean and standard deviation computed from the training data, ensuring that the scaling is consistent across both datasets.

After executing these operations, we are free to go further – finally start the machine learning models and their confluence with our data to get our results which will be discussed later in Section 7.

5.6 Model development and analysis – solving approach

In this part we finally put the ML algorithms into action. In every implementation, the first thing we will do is to fit our data in the algorithm, print out the accuracy score, and show the confusion matrix for each algorithm with detailed information about precision, recall, f1-score and support, such as macro average and weighted average.



01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.
DATA SCIENCE LIFECYCLE
sudeep.co

Here is a brief explanation for each of these parameters:

Precision: Precision measures how many of the predicted positive instances were actually positive. It's calculated as TP / (TP + FP). High precision indicates that when the model predicts a positive class, it's likely to be correct.

Recall (Sensitivity or True Positive Rate): Recall measures how many of the actual positive instances were correctly predicted as positive. It's calculated as TP / (TP + FN). High recall indicates that the model is good at identifying positive instances.

F1-Score: The F1-score is the harmonic mean of precision and recall. It's calculated as 2 * (Precision * Recall) / (Precision + Recall). It provides a balance between precision and recall, especially when dealing with imbalanced datasets.

Support: Support is the number of actual occurrences of each class in the dataset. It represents how many samples belong to each class.

Macro Average (macro_avg): Macro average calculates the metrics (precision, recall, F1-score) separately for each class and then takes the average across all classes. It treats all classes equally, regardless of their size.

Weighted Average (weighted_avg): Weighted average calculates the metrics (precision, recall, F1-score) for each class and takes a weighted average based on the number of true instances for each class. It gives more weight to classes with larger support, making it useful for imbalanced datasets.

But the most important parameter we are looking for in such projects is the accuracy. The formula (4) and (5) tells us how we measure the accuracy of a ML model.

Accuracy measures the percentage of correctly predicted instances (both true positives and true negatives) out of the total instances in the dataset.

$$Accuracy = \frac{num_{of} correct\ predictions}{total\ num_{of}\ predictions} \qquad (4)$$

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \qquad (5)$$

Accuracy provides an overall measure of how well the model is performing across all classes. However, it may not be the best metric for imbalanced datasets, where one class significantly outnumbers the others, as the model can achieve a high accuracy by simply predicting the majority class. In such cases, it's important to consider other metrics like precision, recall, F1-score, and the area under the ROC curve (AUC) for a more comprehensive evaluation.

5.6.1. Logistic Regression

We start implementing our first algorithm. The max_iter indicates the number of iterations for the solver to converge. The ravel() method is used to convert 2D or 3D arrays to 1D.

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(max_iter=50)
logreg.fit(x_train, y_train.values.ravel())
y_pred=logreg.predict(x_test)
acc = metrics.accuracy_score(y_pred,y_test.values.ravel())*100
print("Test Accuracy of Logistic Regression Algorithm:
{:.2f}%".format(acc))
```

Output: Test Accuracy of Logistic Regression Algorithm: 95.00%

After every prediction, this sample of code will be used in order to show the confusion matrix metrics to have e better insights of the results.

```
y_pred = logreg.predict(x_test)
cnf_matrix = metrics.confusion_matrix(y_pred,y_test)
conf_matrix(cnf_matrix,y_test)
# calculate prediction
report = classification_report(y_pred,y_test)
print(report)
```

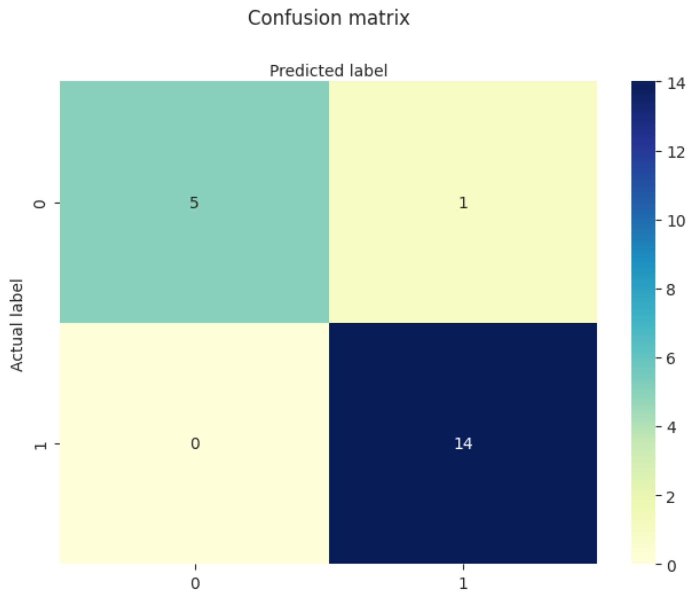


Figure 33 – Confusion matrix of the Logistic Regression model

```
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.93      1.00      0.97        14

    accuracy                           0.95        20
   macro avg       0.97      0.92      0.94        20
weighted avg       0.95      0.95      0.95        20
```

5.6.2. KNN

For the KNN algorithm, we would like to collect all of the scores for up to 20 neighbors, and use only the one(s) with the highest accuracy. We can see down below that the highest score is 85% accuracy but we are having that score multiple times, so we can choose any number of neighbors that matches the highest score.

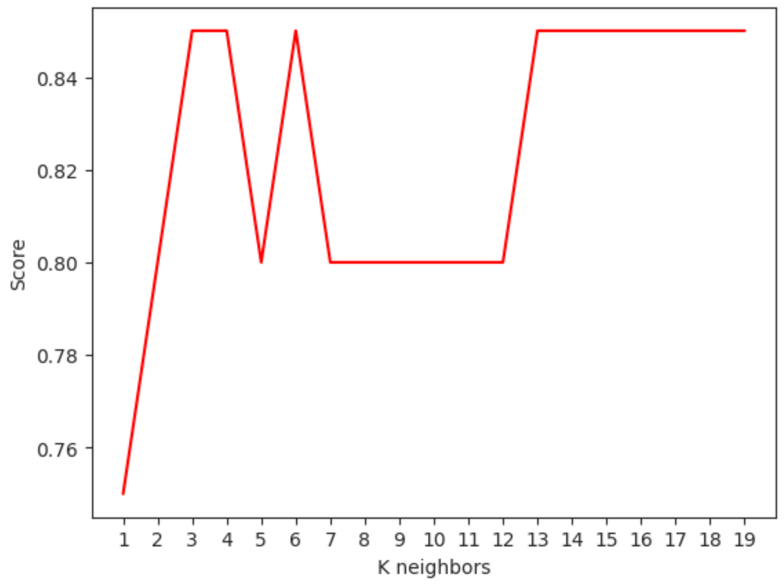
```
from sklearn.neighbors import KNeighborsClassifier

score = []

for i in range(1,20):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(x_train, y_train.values.ravel())
    score.append(knn.score(x_test, y_test.values.ravel()))

plt.plot(range(1,20), score, color='red')
plt.xticks(np.arange(1,20,1))
plt.xlabel("K neighbors")
plt.ylabel("Score")
plt.show()

acc = max(score)*100
print("Maximum KNN Score is {:.2f}%".format(acc))
```



Output: `Maximum KNN Score is 85.00%`

Since one of best value is 4 neighbors, we are further using that specific value in our metrics.

```
knn = KNeighborsClassifier(n_neighbors =4)
knn.fit(x_train, y_train.values.ravel())
```

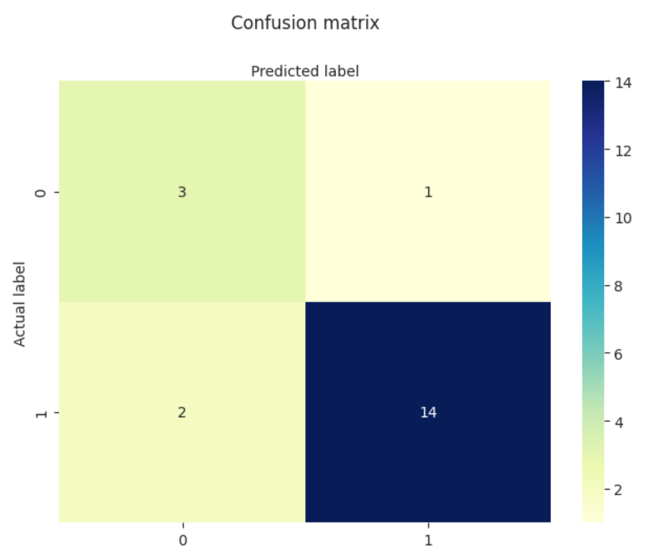


Figure 34 – Confusion matrix of the KNN model

```
              precision    recall  f1-score   support

           0       0.60      0.75      0.67         4
           1       0.93      0.88      0.90        16

    accuracy                           0.85        20
   macro avg       0.77      0.81      0.78        20
weighted avg       0.87      0.85      0.86        20
```

5.6.3. SVM

The implementation of the SVM algorithm is pretty straightforward, we just fit the train data in the model and then simply test it.

```
from sklearn.svm import SVC
svm = SVC(random_state = 41)
svm.fit(x_train, y_train.values.ravel())

y_pred=svm.predict(x_test)
acc = metrics.accuracy_score(y_pred,y_test.values.ravel())*100

print("Test Accuracy of SVM Algorithm: {:.2f}%".format(acc))
```

Output: `Test Accuracy of SVM Algorithm: 80.00%`

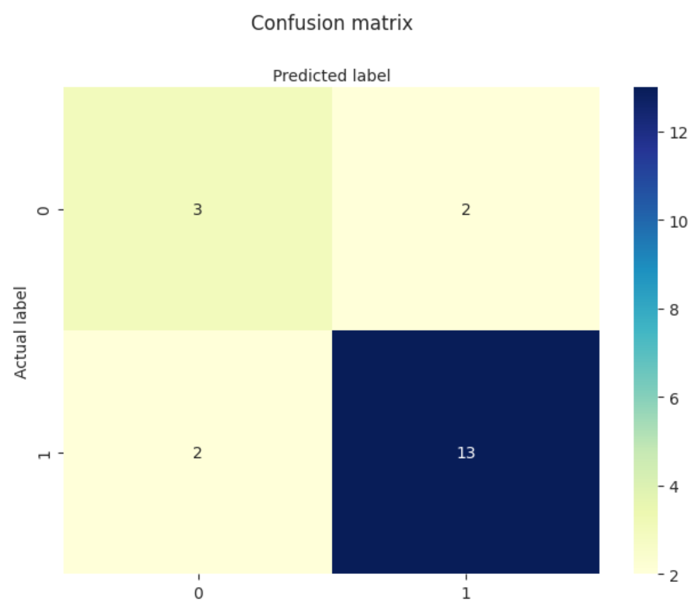


Figure 35 – Confusion matrix of the SVM model

```
              precision    recall  f1-score   support

           0       0.60      0.60      0.60         5
           1       0.87      0.87      0.87        15

    accuracy                           0.80        20
   macro avg       0.73      0.73      0.73        20

weighted avg       0.80      0.80      0.80        20
```

5.6.4 Naïve Bayes

The implementation of the Naïve Bayes algorithm is also simple, same as SVM, we just change the name of our classifier.

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train.values.ravel())

y_pred=nb.predict(x_test)
acc = metrics.accuracy_score(y_pred,y_test.values.ravel())*100
print("Accuracy of Naive Bayes: {:.2f}%".format(acc))
```

Output: `Accuracy of Naive Bayes: 85.00%`

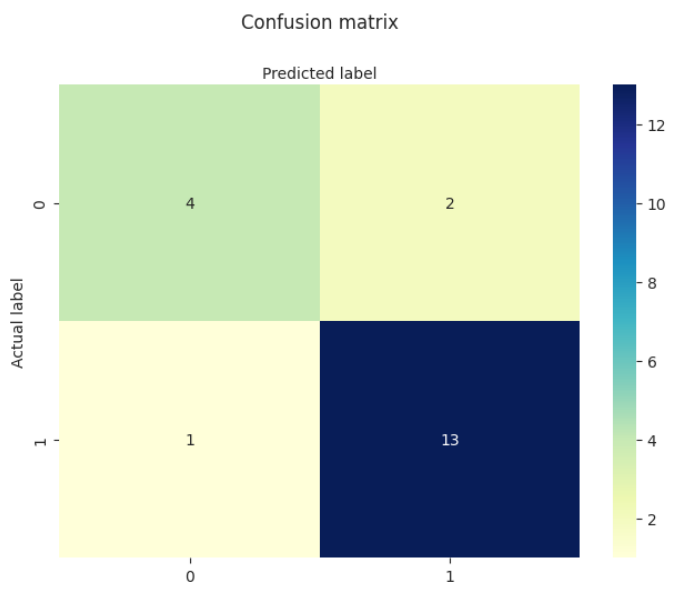


Figure 36 – Confusion matrix of the NB model

```
              precision    recall  f1-score   support

           0       0.80      0.67      0.73         6
           1       0.87      0.93      0.90        14

    accuracy                           0.85        20
   macro avg       0.83      0.80      0.81        20

weighted avg       0.85      0.85      0.85        20
```

5.6.5 Decision Tree

Same here, same steps as above. Note that in every algorithm we have to multiply the accuracy score with 100 in order to get a result in percentages. We format all of the results to only two decimal numbers.

```
from sklearn.tree import DecisionTreeClassifier
dtcla = DecisionTreeClassifier(random_state=41)

dtcla.fit(x_train, y_train.values.ravel())

y_pred = dtcla.predict(x_test)
acc = metrics.accuracy_score(y_pred,y_test.values.ravel())*100

print("Accuracy of DT : {:.2f}%".format(acc))
```

```
Accuracy of DT : 85.00%
```

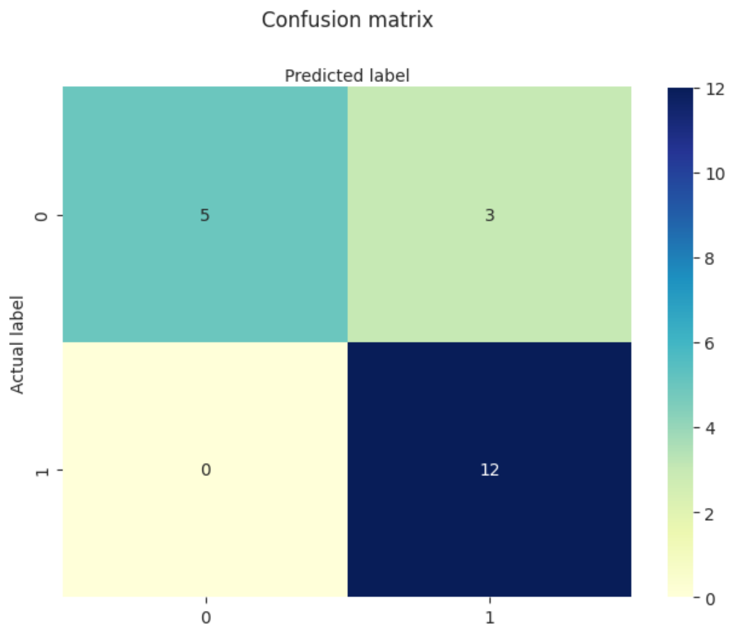


Figure 37 – Confusion matrix of the DT model

```
              precision    recall  f1-score   support

           0       1.00      0.62      0.77         8
           1       0.80      1.00      0.89        12

    accuracy                           0.85        20
   macro avg       0.90      0.81      0.83        20
weighted avg       0.88      0.85      0.84        20
```

5.6.6. Random Forest

When dealing with random forests, we actually use decision tree classification, but with many 'trees'. In the RF classifier we have the parameter class_weight = 'balanced'. This parameter specifies how class weights should be determined. Setting it to "balanced" means that scikit-learn will automatically adjust the weights of the classes based on the number of samples in each class. This is particularly useful for imbalanced datasets, as it helps the model give more importance to the minority class.

We also define the number of estimators i.e., the number of decision trees. A larger number of estimators can lead to better performance, but it also increases the computational cost. Two hundred (200) trees are used in this case.

```
from sklearn.ensemble import RandomForestClassifier
rf =
RandomForestClassifier(class_weight="balanced",n_estimators=200,random_sta
te = 41)
rf.fit(x_train, y_train.values.ravel())
y_pred=rf.predict(x_test)
acc = metrics.accuracy_score(y_pred,y_test.values.ravel())*100
print("Random Forest Algorithm Accuracy Score : {:.2f}%".format(acc))
```

```
Random Forest Algorithm Accuracy Score : 95.00%
```

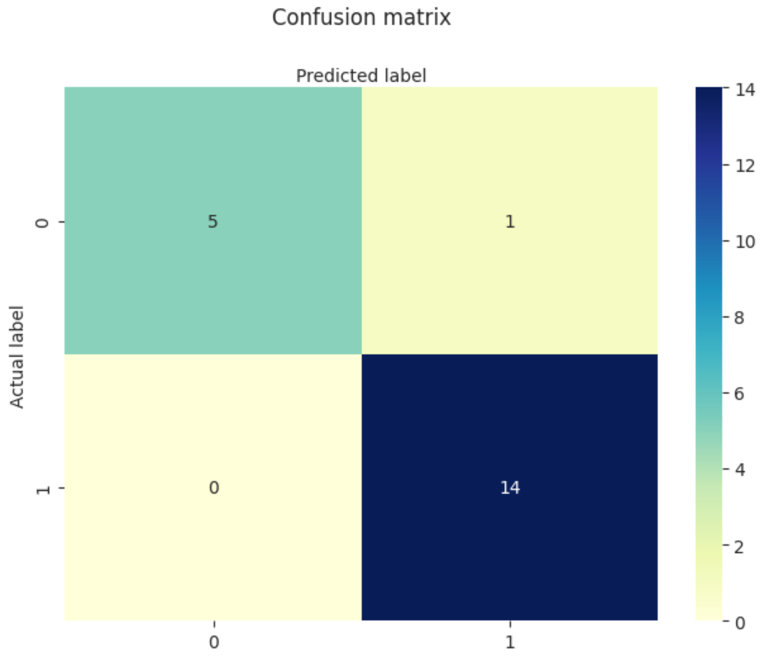


Figure 38 – Confusion matrix of the RF model

```
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.93      1.00      0.97        14

    accuracy                           0.95        20
   macro avg       0.97      0.92      0.94        20
weighted avg       0.95      0.95      0.95        20
```

5.6.7 Extreme Gradient Boosting

```
from xgboost import XGBClassifier
xgb=XGBClassifier(random_state=41)
xgb.fit(x_train, y_train,)
```

After importing the *XGBClassifier*, we fit the train data and the next step is to examine the performance. But we will spend a little more time here, first showing the performance of the training data which was 100%!

Output:

```
Training data confusion matrix
[[32  0]
 [ 0 46]]
```

This means that our train data was 100% successfully classified. However, more important in these kinds of projects is the testing data performance, so we will see that as well.

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
predictions= xgb.predict(x_train)
percentage=xgb.score(x_train,y_train)
res=confusion_matrix(y_train,predictions)
print("Training data confusion matrix")
print(res)
predictions= xgb.predict(x_test)
cnf_matrix = metrics.confusion_matrix(predictions, y_test)
conf_matrix(cnf_matrix,y_test)
report = classification_report(predictions,y_test)
print(report)
percentage=xgb.score(x_test,y_test)
res=confusion_matrix(y_test,predictions)
print("Testing data confusion matrix")
print(res)

print('training accuracy = '+str(xgb.score(x_train, y_train)*100))
print('testing accuracy = '+str(xgb.score(x_test, y_test)*100))
method_names.append("XGB")
method_scores.append(xgb.score(x_test, y_test))
```

Output:

```
Testing data confusion matrix
[[ 5  0]
 [ 1 14]]


training accuracy = 100.0
testing accuracy = 95.0
```

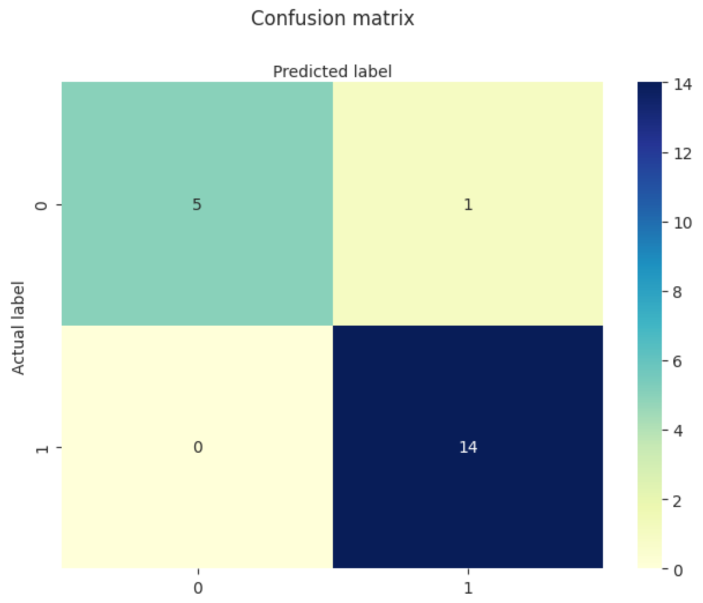


Figure 39 – Confusion matrix of the XGB model

```
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.93      1.00      0.97        14

    accuracy                           0.95        20
   macro avg       0.97      0.92      0.94        20
weighted avg       0.95      0.95      0.95        20
```

5.6.8 Artificial Neural Network

This model requires a bit more explanation. First of all, we import the necessary libraries for building and training neural networks. A sequential model is being used; that means we use a linear stack of layers. We will add layers to this model one by one. Then, we add the input layer and the first hidden layer. The 'Dense' layer represents a fully connected layer with 12 units (neurons) and ReLU (Rectified Linear Unit) activation function. The input_dim specifies the number of input features, which is inferred from the shape of x_train.

We can optionally add a dropout layer which is used for regularization to prevent overfitting. A dropout rate of 0.2 means that during training, 20% of the neurons in this layer will be randomly dropped out, preventing them from learning.

We can add more hidden layers in our model if needed. Usually, two to three layers are enough, since this is a small dataset, we don't need many layers.

The output layer has 1 unit with a sigmoid activation function. This is suitable for binary classification problems, where the output represents the probability of belonging to one class.

Next, we have to compile the model.

- optimizer='adam': This specifies the optimization algorithm. Adam is a popular optimizer for training neural networks.
- loss='binary_crossentropy': This is the loss function used for binary classification tasks.
- metrics=['accuracy']: It specifies the metrics to be monitored during training. In this case, accuracy is chosen.

The training part of the model is in the history variable.

- x_train and y_train are the training data and labels.
- epochs=200 specifies the number of training iterations.
- batch_size=8 determines how many samples are used in each update of the model's weights during training.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

tf.random.set_seed(41)

model = Sequential()

# Add input layer and first hidden layer
model.add(Dense(units=12, activation='relu', input_dim=x_train.shape[1]))
model.add(Dropout(0.2))  # Optional dropout layer
```

```
# Add more hidden layers
model.add(Dense(units=8, activation='relu'))

# Add output layer
model.add(Dense(units=1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

model.summary()

history = model.fit(x_train, y_train, epochs=200, batch_size=8)

test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

The last part of the output along with the accuracy of the model is shown below:

```
Epoch 197/200
10/10 [==============================] - 0s 5ms/step - loss: 0.2555 -
accuracy: 0.9103
Epoch 198/200
10/10 [==============================] - 0s 6ms/step - loss: 0.2224 -
accuracy: 0.8974
Epoch 199/200
10/10 [==============================] - 0s 7ms/step - loss: 0.2404 -
accuracy: 0.9231
Epoch 200/200
10/10 [==============================] - 0s 5ms/step - loss: 0.2145 -
accuracy: 0.9359
1/1 [==============================] - 0s 224ms/step - loss: 0.2060 -
accuracy: 0.9500
Test Loss: 0.2059541642665863
Test Accuracy: 0.949999988079071
```

5.6.9 Linear Discriminant Analysis

This method tries to reduce the number of dimensions to prevent learning and memorization. Therefore, it can provide a promising advantage in feature extraction as well as the application as a classification algorithm. After we create the model by calling the module, we fit the data and show the accuracy score

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score

lda = LinearDiscriminantAnalysis()

lda.fit(x_train, y_train)
y_pred = lda.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
method_names.append("LDA")
method_scores.append(lda.score(x_test, y_test))
```

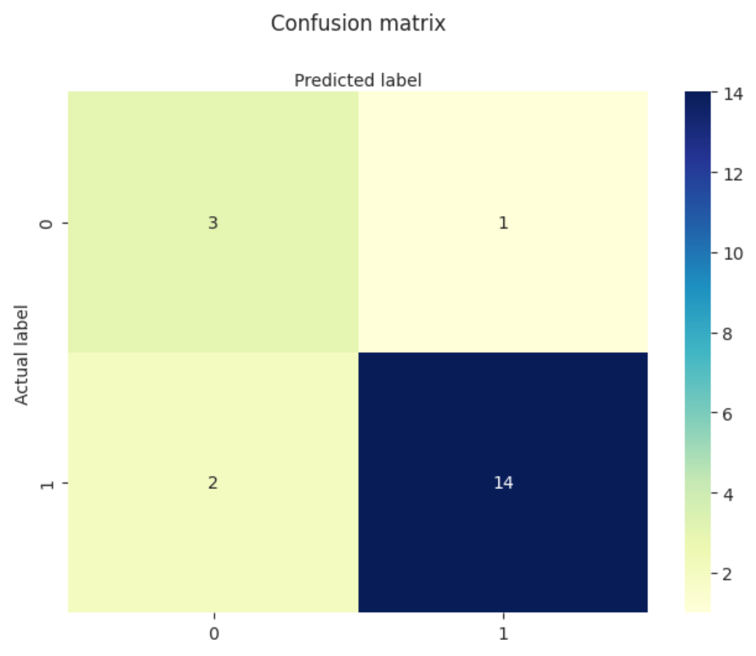


Figure 40 – Confusion matrix of the LDA model

Output: `Accuracy: 0.85`

```
              precision    recall  f1-score   support

           0       0.60      0.75      0.67         4
           1       0.93      0.88      0.90        16

    accuracy                           0.85        20
   macro avg       0.77      0.81      0.78        20
weighted avg       0.87      0.85      0.86        20
```

5.6.10 Ensemble learning (Voting Classifier)

With this algorithm we want to include actually more than one algorithm, i.e., we want to jointly use at least two algorithms and see their performance together. In our case we are using the random forest classifier and the logistic regression, since these two gave us the best results.

```
from sklearn.ensemble import VotingClassifier

eclf1 = VotingClassifier(estimators=[('rf', rf), ('lr', logreg)],
                         voting='hard')
eclf1 = eclf1.fit(x_train, y_train.values.ravel())
print(eclf1.predict(x_test))
eclf2 = VotingClassifier(estimators=[('rf', rf), ('lr',
logreg)],voting='soft')
eclf2 = eclf2.fit(x_train, y_train.values.ravel())
print(eclf2.predict(x_test))
```

Output:

```
[0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1]

[0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1]
```

There are two common voting strategies used in Voting Classifiers:

1. Hard Voting: In hard voting, each base model makes a prediction, and the final prediction is determined by a majority vote. For classification tasks, the class that receives the most votes among the base models is chosen as the final prediction. In the case of a tie, the Voting Classifier may choose randomly or based on some predefined rule. This is how our first example is done in the code, more specifically, the eclf1 classifier.

2. Soft Voting: In soft voting, each base model provides class probabilities (or regression values) as predictions. The final prediction is computed by averaging these probabilities (or values). For classification tasks, this results in a weighted average of class probabilities, and the class with the highest average probability is selected as the final prediction. This kind of voting is used in the second classifier i.e., eclf2.

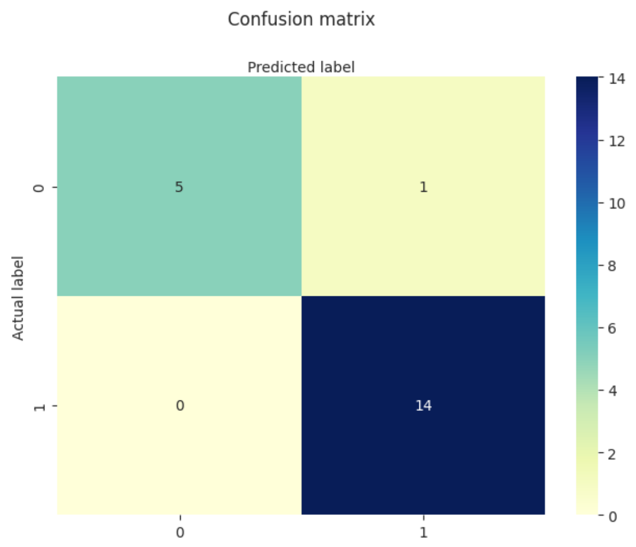


Figure 41 – CM of the Ensemble model - hard voting

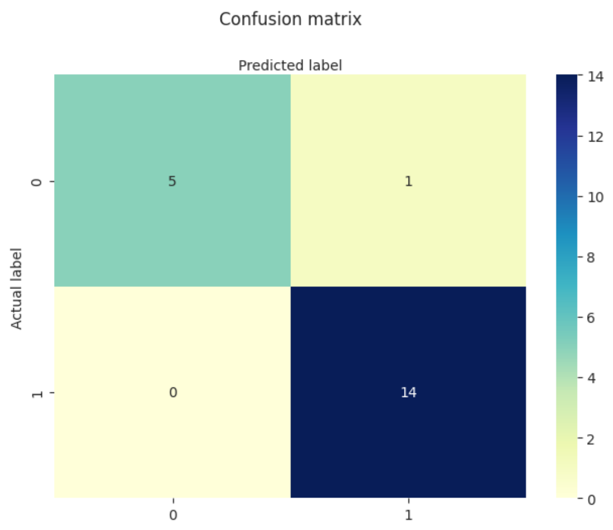


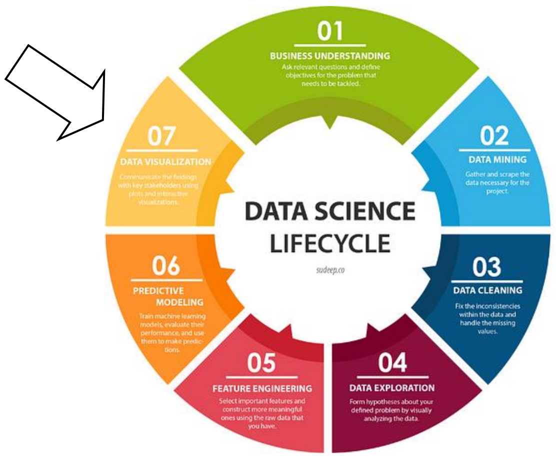
Figure 42 – CM of the Ensemble model – soft voting

```
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.93      1.00      0.97        14

    accuracy                           0.95        20
   macro avg       0.97      0.92      0.94        20
weighted avg       0.95      0.95      0.95        20
```

6. Results and discussion - Interpretation and insights


01
BUSINESS UNDERSTANDING
Ask relevant questions and define objectives for the problem that needs to be tackled.
02
DATA MINING
Gather and scrape the data necessary for the project.
03
DATA CLEANING
Fix the inconsistencies within the data and handle the missing values.
04
DATA EXPLORATION
Form hypotheses about your defined problem by visually analyzing the data.
05
FEATURE ENGINEERING
Select important features and construct more meaningful ones using the raw data that you have.
06
PREDICTIVE MODELING
Train machine learning models, evaluate their performance, and use them to make predictions.
07
DATA VISUALIZATION
Communicate the findings with key stakeholders using plots and interactive visualizations.
DATA SCIENCE LIFECYCLE
sudeep.co

We are finally done with the inspection of all our 10 algorithms. In order to show the results, we created two lists: method_names and method_scores in which we store the data of our model accuracies and plotted them in a simple bar graph:

```
plt.figure(figsize=(15,10))
plt.ylim([0.75, 1])
plt.bar(method_names,method_scores,width=0.5, color='orange')
plt.xlabel('Method Name')
plt.ylabel('Method Score')
```

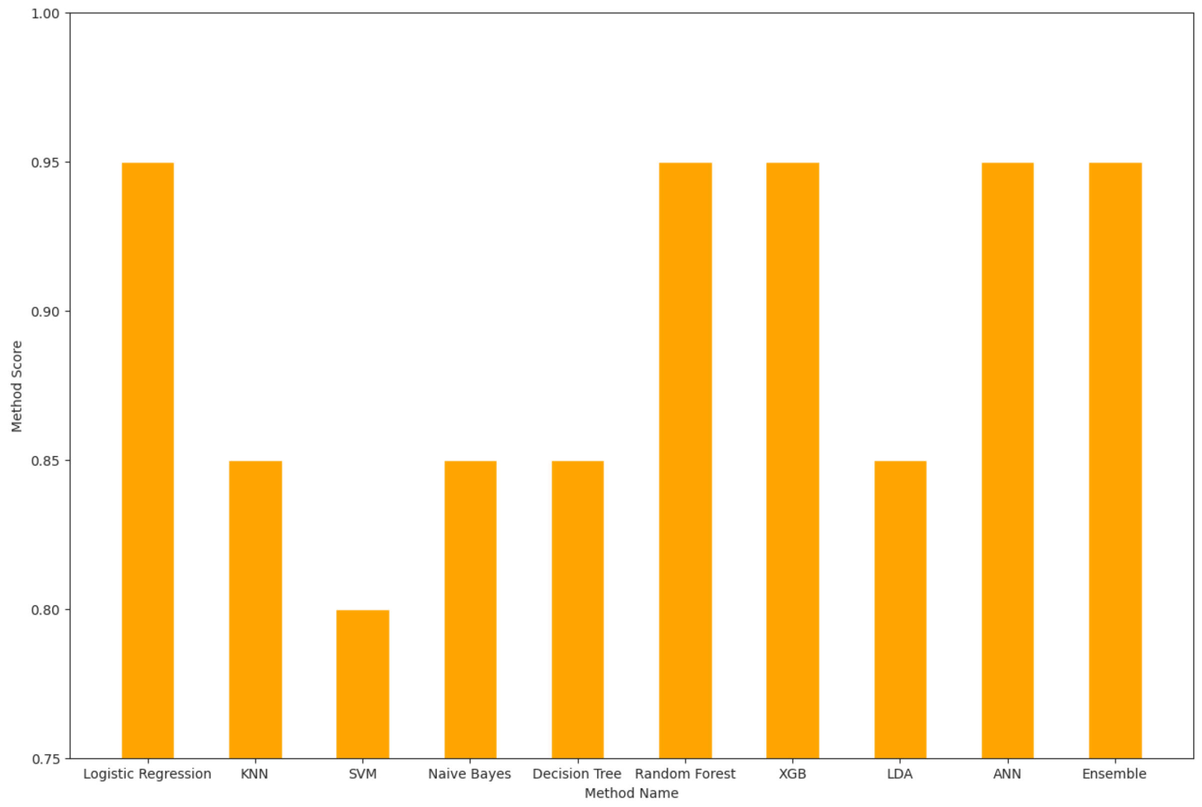


Figure 43 – Comparison of the accuracy of all algorithms applied

We can easily observe that even half of our algorithms i.e., 5 out of 10 had an accuracy of 95% - meaning on the testing dataset they successfully classified our data with a 95% score. Accuracy, however, while a commonly used metric for evaluating the performance of classification models, is not always sufficient on its own to provide a complete understanding of a model's effectiveness, especially in situations where the data is imbalanced or when certain types of errors are more costly or critical than others. Here are some reasons why accuracy alone may be insufficient:

1. Class Imbalance: In datasets where one class significantly outweighs the other(s), achieving high accuracy can be trivial if the model predicts the majority class most of the time.
2. Misclassification Costs: In many real-world applications, misclassifying one class can have more severe consequences than misclassifying another.
3. Asymmetric Errors: In some scenarios, false positives and false negatives have different implications.
4. Incomplete Information: Accuracy does not provide information about the model's ability to discriminate between different classes.

To address these limitations, other evaluation metrics are often used in combination with accuracy:

- Precision: It measures the proportion of true positive predictions among all positive predictions. Precision is valuable when minimizing false positives is critical.
- Recall (Sensitivity or True Positive Rate): It measures the proportion of true positive predictions among all actual positives. Recall is valuable when minimizing false negatives is crucial.
- F1-Score: The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance.
- Confusion Matrix: Examining the confusion matrix provides a detailed breakdown of true positives, true negatives, false positives, and false negatives, offering insights into specific error types.
- Area Under the Receiver Operating Characteristic Curve (AUC-ROC): ROC curves plot the trade-off between true positive rate (sensitivity) and false

positive rate at various thresholds. AUC-ROC quantifies the overall discriminative power of the model.

If you recall, so far, we used all other metrics (precision, recall, f1-score, confusion matrix) in each of our ML algorithms, except the last evaluation metric – AUC-ROC.

The AUC-ROC, or Area Under the Receiver Operating Characteristic Curve, is an evaluation metric commonly used to assess the performance of binary classification models. It quantifies the model's ability to discriminate between the positive and negative classes across different classification thresholds. Here's a brief explanation of AUC-ROC:

1. Receiver Operating Characteristic (ROC) Curve: The ROC curve is a graphical representation of a classifier's performance. It plots the True Positive Rate (TPR), also known as Sensitivity or Recall, on the y-axis and the False Positive Rate (FPR) on the x-axis as the classification threshold is varied.

    - TPR (Sensitivity): It measures the proportion of true positives among all actual positives, indicating how well the model correctly identifies positive instances.

    - FPR: It measures the proportion of false positives among all actual negatives, representing the rate of false alarms.

2. AUC-ROC: The AUC-ROC quantifies the area under the ROC curve. It's a single scalar value between 0 and 1, where:

    - AUC-ROC = 0.5: A random classifier that makes predictions by chance has an AUC-ROC of 0.5. It means the model cannot discriminate between the two classes.

    - AUC-ROC > 0.5: A classifier with an AUC-ROC greater than 0.5 is better than random. It indicates that the model has some level of discriminative power.

    - AUC-ROC = 1: A perfect classifier has an AUC-ROC of 1, meaning it perfectly separates the two classes with no false positives.

3. Interpretation: The AUC-ROC value represents the probability that a randomly chosen positive sample will be ranked higher (have a higher predicted probability) than a randomly chosen negative sample. In other words, it

measures how well the model ranks positive instances above negative instances across different thresholds.

4. Model Comparison: A higher AUC-ROC suggests that the model is better at distinguishing between the two classes. It is a useful metric for comparing different models or variations of a model.

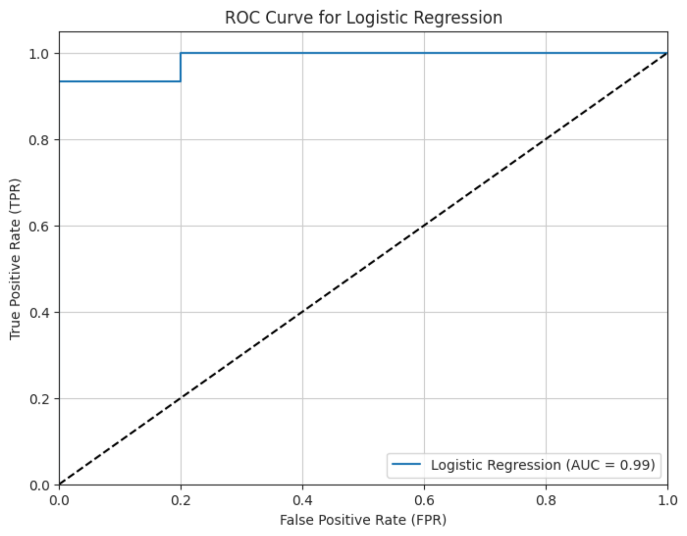


Figure 44 – ROC curve for LR

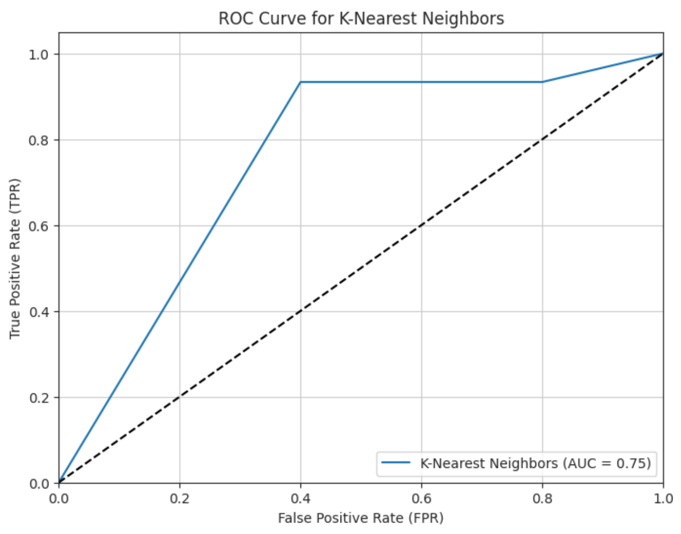


Figure 45 – ROC curve for KNN

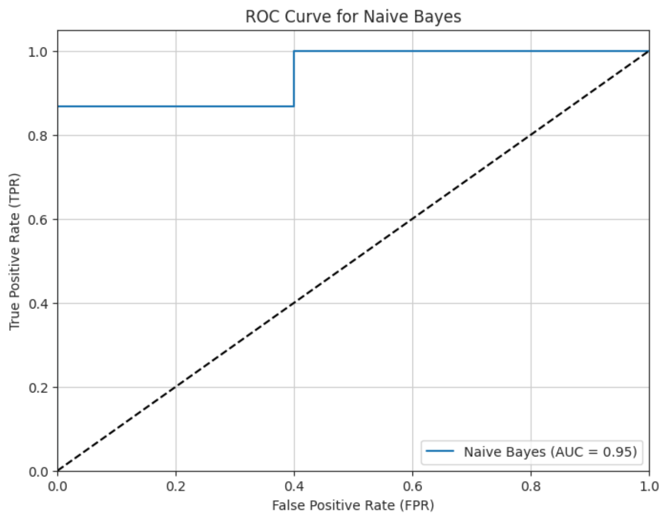


Figure 46 – ROC curve for NB

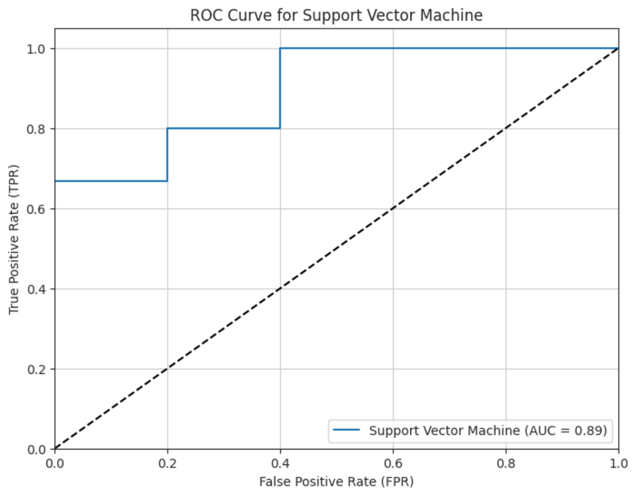


Figure 47 – ROC curve for SVM

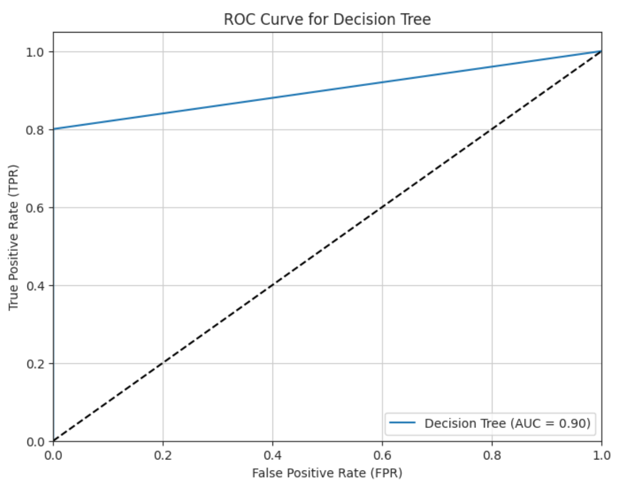


Figure 48 – ROC curve for DT

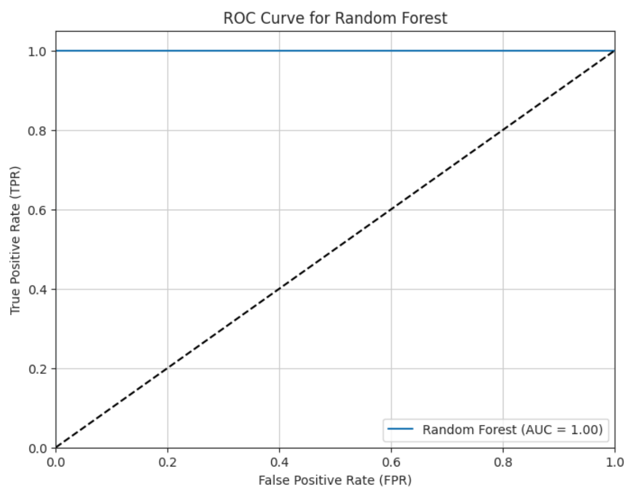


Figure 49 – ROC curve for RF

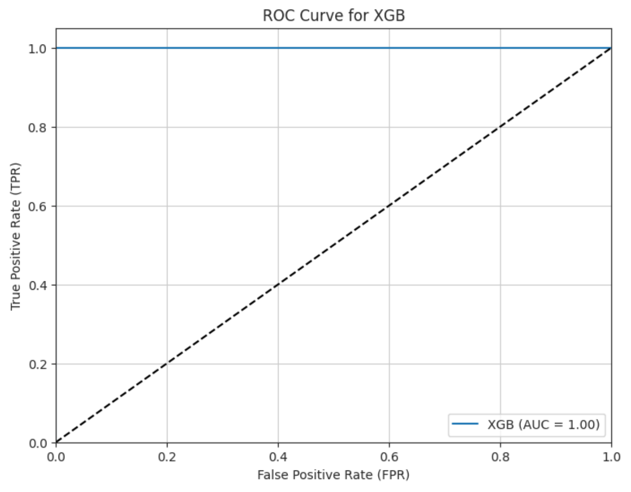


Figure 50 – ROC curve for XGB

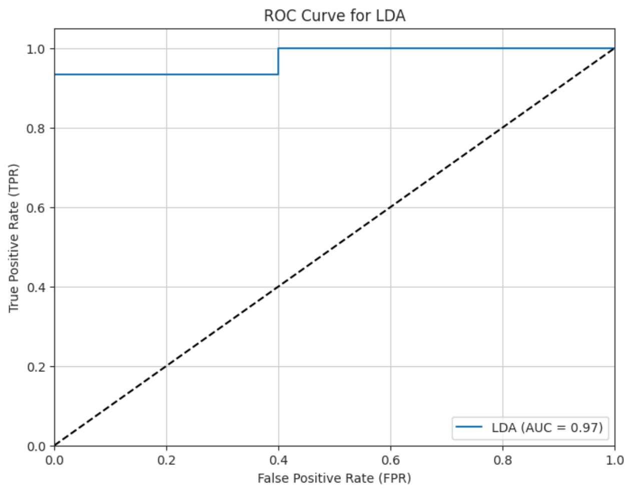


Figure 51 – ROC curve for LDA

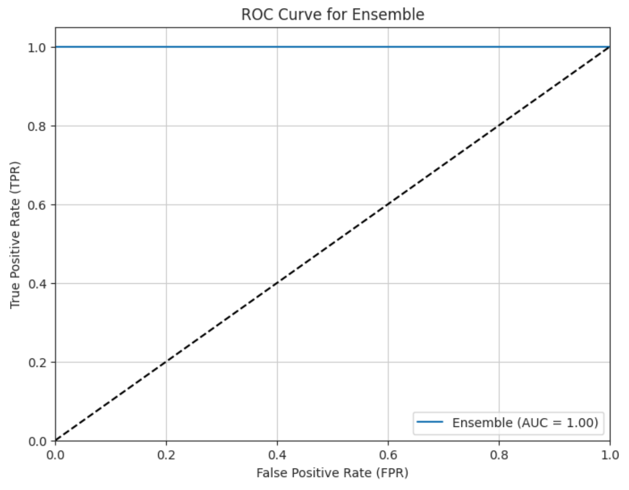


Figure 52 – ROC curve for Ensemble learning

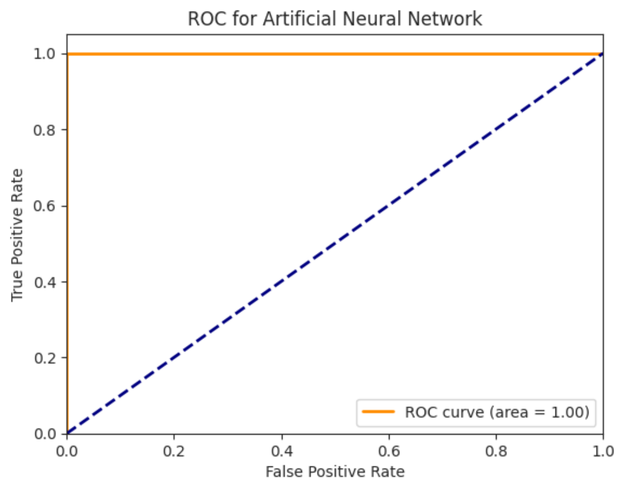


Figure 53 – ROC curve for Artificial Neural Network

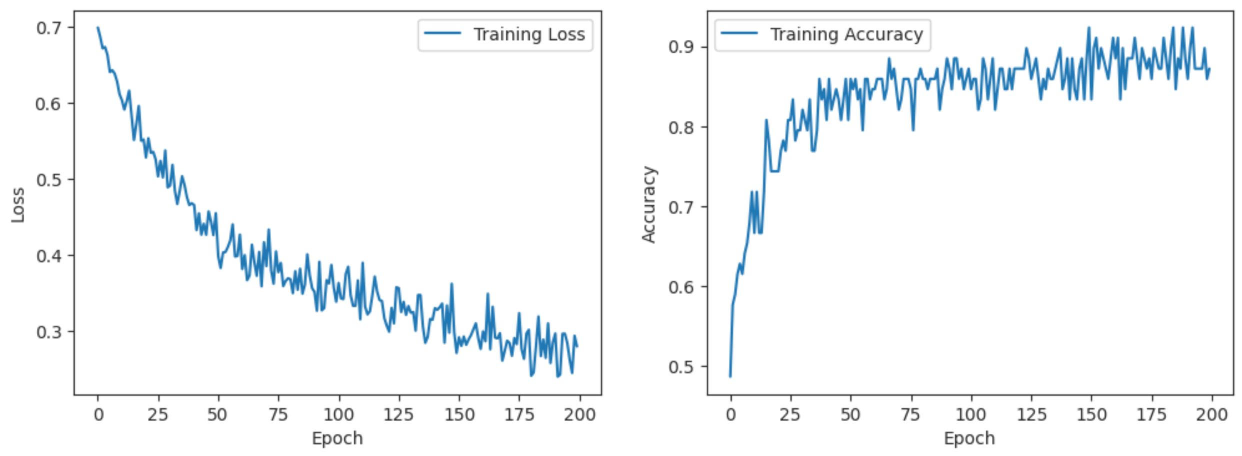


Figure 54 – Training loss and Training Accuracy of the ANN model subplots

After going through the AUC-ROC results in the graphs above, we can confirm that by combining all of the evaluation metrics mentioned earlier in this section, Random Forest, XGB, ANN and Ensemble learning gave the best results. While testing the ANN, the results were volatile. Note that ensemble learning contains random forest in itself, therefore XGB and Random forest are the best one to note.

We should not underestimate the other algorithms as well, since all of them gave pretty good results, however we will go further on with the best two and compare them.

Here's a comparison in terms of speed, computational expense, and ease of use:

Speed:

- Random Forest: Random Forest can be relatively faster to train on the Prostate Cancer dataset, which typically has a moderate number of features and samples. The dataset size is not extremely large, so Random Forest should handle it efficiently.
- XGBoost: XGBoost is generally fast and efficient, even for moderately sized datasets like the Prostate Cancer dataset. It's unlikely that training XGBoost on this dataset would be prohibitively slow.

Computational Expense:

- Random Forest: Random Forest tends to be less computationally expensive than XGBoost in terms of both training and prediction. This may be an advantage when working with limited computational resources.

- XGBoost: While XGBoost can be more computationally intensive than Random Forest, it's still manageable for datasets of this size. Its speed and efficiency often justify the computational cost.

Ease of Use:

- Random Forest: Random Forest is known for its ease of use and is suitable for users who prefer a straightforward approach with fewer hyperparameters to fine-tune. It's a good starting point for analyzing the Prostate Cancer dataset.
- XGBoost: XGBoost has a richer set of hyperparameters, but it also provides greater control and potential for optimization. If you're experienced with hyperparameter tuning and want to maximize predictive accuracy on this dataset, XGBoost is a valuable choice.

Considering the Prostate Cancer dataset's characteristics (moderate size and balanced classes), both Random Forest and XGBoost are reasonable options. If you prioritize simplicity and want to quickly develop a predictive model, Random Forest is a solid choice. However, if you aim to extract the best possible predictive performance from the dataset and are willing to invest some effort in hyperparameter tuning, XGBoost may offer an advantage in terms of accuracy.

7. Future work

After analyzing the results, we can state they were quite successful, the ML models worked out pretty fine and gave promising results. However, this was just one example of such study. We should take into consideration the possible ways one can improve such a study or which similar ways can be taken to build another ML model that may outperform the achieved results. There are other considerations not mentioned in this study, but necessary ones if the ML model will be implemented in the hospitals for example. Some potential directions that can be taken when dealing with this kind of a study are the following:

Integration of Multi-Omics Data: Multi-omics data such as genomics, transcriptomics, proteomics, and metabolomics can be incorporated into the analysis. Combining different data types could enhance the model's predictive power by capturing a more comprehensive view of the disease. In our case, we had kind of a limited amount of data and that can be improved to get deeper and more relevant results.

Longitudinal Analysis: Involving longitudinal patient data to analyze disease progression over time. This could involve methods like time-series analysis or recurrent neural networks (RNNs).

Clinical Validation: Collaborating with clinicians to validate the ML models' predictions using real-world patient data. Clinical validation is crucial for translating research findings into clinical practice.

Integration with Genetic Risk Factors: Integrating genetic risk factors, such as family history and genetic mutations, into the models can improve prediction accuracy.

Clinical Impact Assessment: Assess the clinical impact of implementing ML models in real healthcare settings. Consider factors like reduction in unnecessary biopsies, improvement in patient outcomes, and cost-effectiveness.

Ethical and Regulatory Considerations: Investigate ethical considerations surrounding the use of patient data, model transparency, and patient consent when implementing ML models in clinical settings.

8. Conclusion

This study on the performance and analysis of ML algorithms in prostate cancer prediction has shed light on the promising potential of machine learning in the field of healthcare. This research aimed to evaluate and analyze the effectiveness of various machine learning algorithms in predicting prostate cancer, a prevalent and potentially life-threatening disease.

Throughout this research, several significant insights and outcomes have been achieved:

1. Algorithm Performance: Our study extensively compared the performance of different machine learning algorithms, including Logistic Regression, Random Forest, Support Vector Machine, k-Nearest Neighbors, Naive Bayes, and more. These algorithms were rigorously evaluated based on metrics like accuracy, sensitivity, specificity, and area under the ROC curve (AUC).
2. Data Integration: Our research examined the potential benefits of integrating diverse data types, such as genomics, clinical records, and imaging data. This multi-modal approach would provide a comprehensive view of prostate cancer prediction.

3. Clinical Relevance: The machine learning models developed in this research hold promise for improving early diagnosis and risk assessment. These models can potentially assist healthcare providers in making informed decisions and tailoring treatments to individual patients.

4. Challenges and Ethical Considerations: We acknowledge the challenges related to data privacy, bias, and model interpretability in healthcare applications. Addressing these challenges is crucial to ensure the responsible and ethical deployment of machine learning in clinical settings.

In conclusion, the findings of this study underscore the potential of machine learning as a valuable tool in prostate cancer prediction. While no single algorithm emerged as universally superior, the comparative analysis provided valuable insights into algorithm performance under different scenarios and data conditions.

Ultimately, the application of machine learning in prostate cancer prediction holds the promise of early detection, improved patient outcomes, and more personalized healthcare—a significant stride towards the advancement of medicine and the well-being of individuals at risk of this disease.

I should also note that this was a very limited dataset, and given the relatively low amount of data, I still managed to obtain pretty satisfying results. During the model evaluation phase, split of the data in ratio 70:30 and 85:15 was also tried but in the case 80:20 the results were the most balanced ones.

9. References

[1] American Cancer Society. *Facts & Figures 2023*. American Cancer Society. Atlanta, Ga. 2023 *Available online here*

[2] State Statistical Office of Republic of N. Macedonia (2023), Men and Women in N. Macedonia *Available Online here*

[3] Li Zhang et al. "A new approach to diagnosing prostate cancer through magnetic resonance imaging" *Alexandria Engineering Journal, Elsevier, vol. 60, page: 897-904 (2021)*

[4] Boehm, K.M., Khosravi, P., Vanguri, R. *et al.* Harnessing multimodal data integration to advance precision oncology. *Nat Rev Cancer* 22, 114–126 (2022). *Available online here*

[5] *December 8, 2021, Artificial Intelligence for Automated Cancer Detection on Prostate MRI: Opportunities and Ongoing Challenges, From the AJR Special Series on AI Applications*
Baris Turkbey, MD and Masoom A. Haider, MD Volume 219, Issue 2

[6] Peter D. Caie BSc, MRes, PhD, Ognjen Arandjelović M.Eng. (Oxon), Ph.D. (Cantab), in Artificial Intelligence and Deep Learning in Pathology, 2021

[7] Gou, Jianping & Zhan, Yongzhao & Rao, Yunbo & Shen, Xiangjun & Wang, Xiaoming & He, Wu. (2014). Improved pseudo nearest neighbor classification. Knowledge-Based Systems. 70. 361–375. 10.1016/j.knosys.2014.07.020.

[8] Erdem, E. & Bozkurt, F. (2021). A Comparison of Various Supervised Machine Learning Techniques for Prostate Cancer Prediction. European Journal of Science and Technology, (21), 610-620.

[9] Guo, Qian, et al. "Urban Tree Classification Based on Object-Oriented Approach and Random Forest Algorithm Using Unmanned Aerial Vehicle (UAV) Multispectral Imagery." *Remote Sensing*, vol. 14, no. 16, Aug. 2022, p. 3885. *Crossref*, *Available online here*

[10] Konstantinos Panagiotopoulos, et al. "MEvA-X: a hybrid multiobjective evolutionary tool using an XGBoost classifier for biomarkers discovery on biomedical datasets", BIOINFORMATICS, Volume 39, Issue 7, July 2023, btad384,
Available online here

[11] Erdem, E. & Bozkurt, F. (2021). A Comparison of Various Supervised Machine Learning Techniques for Prostate Cancer Prediction. European Journal of Science and Technology, (21), 610-620.

[12] Aliyari Ghassabeh, Youness; Rudzicz, Frank; Moghaddam, Hamid Abrishami (2015-06-01). "Fast incremental LDA feature extraction". *Pattern Recognition*. 48 (6): 1999–2012

Other links

Article - Improved pseudo nearest neighbor classification
researchgate.net/publication/266320558_Improved_pseudo_nearest_neighbor_classification

https://www.mayoclinic.org/diseases-conditions/prostate-cancer/symptoms-causes/syc-20353087 Article – symptoms and causes of prostate cancer

https://www.sciencedirect.com/topics/computer-science/logistic-regression Logistic Regression - Exploratory study

https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc Overview of LR algorithm

https://www.researchgate.net/figure/Model-of-help-to-diagnose-of-the-prostate-cancer_fig1_270901781 Naive bayes network model to diagnose the prostate cancer

https://link.springer.com/chapter/10.1007/978-3-030-90633-7_72 Naive Bayes and Support Vector Machines overview

https://www.researchgate.net/figure/Classification-and-regression-tree-analysis-in-men-with-a-serum-PSA-level-of-10-ng-mL_fig1_7952998 Classification and regression tree analysis in men with serum PSA level of 10ng/mL

https://www.mdpi.com/2072-4292/14/16/3885 Random Forest overview

https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article#what_is_xgboost_in_machine_learning XGB algorithm overview

https://www.linkedin.com/pulse/xgboost-classifier-algorithm-machine-learning-kavya-kumar/ article about XGBoost on LinkedIn

https://www.nvidia.com/en-us/glossary/data-science/xgboost/ Nvidia article for XGB

https://medium.com/co-learning-lounge/complete-data-science-project-life-cycle-9eae6e4ed4c9 - Lifecycle of a Data Science project

https://geekflare.com/google-colab/ Article about Google Colab

https://www.youtube.com/watch?v=t4k1LwRA2KE video for Prostate Cancer

10. Appendices

Jupyter notebook/code repository and the dataset –

https://github.com/dimitarmit7/ML-algorithms-in-prostate-cancer-prediction