12-2023

HYPOTHYROID DISEASE ANALYSIS BY USING MACHINE LEARNING

SANJANA SEELAM

HYPOTHYROID DISEASE ANALYSIS BY USING MACHINE LEARNING

_____

A Project

Presented to the

Faculty of

California State University,

San Bernardino

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

Sanjana Seelam

December 2023

HYPOTHYROID DISEASE ANALYSIS BY USING MACHINE LEARNING

_____

A Project

Presented to the

Faculty of

California State University,

San Bernardino

_____

by

Sanjana Seelam

December 2023

Approved by:

Dr. Qingquan Sun, Advisor, Computer Science and Engineering

Dr. Jennifer Jin, Committee Member

Dr. Yan Zhang, Committee Member

ABSTRACT

Thyroid illness frequently manifests as Hypothyroidism. It is evident that people with hypothyroidism are primarily female. Because the majority of people are unaware of the illness, it is quickly becoming more serious. It is crucial to catch it early on so that medical professionals can treat it more effectively and prevent it from getting worse. Machine Learning illness prediction is a challenging task. Disease prediction is aided greatly by machine learning. Once more, unique feature selection strategies have made the process of disease assumption and prediction easier.

To properly monitor and cure this illness, accurate detection is essential. In order to build models that can forecast the development of hypothyroidism. In this project, we utilized machine learning approaches such Logistic Regression, Decision Trees, and Naive Bayes. Here, we used thyroid function-related measures and characteristics from a UCI Machine learning repository dataset.

The main goals were to properly assess each machine learning model's performance and fine-tune its hyperparameters. With an accuracy rate of 99.87%, the findings of this study generated the model's ability to predict hypothyroidism were pretty remarkable.

This high degree of accuracy shows how useful these machine learning algorithms are as diagnostic and therapeutic tools for hypothyroid patients early on. This experiment demonstrates the potential of machine learning in healthcare and has an impact on diagnosis.

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Background

The approach of using machine learning techniques to diagnose hypothyroidism leverages data analysis capabilities in a new way. It has implications for the diagnosis and management of hypothyroidism. Hypothyroidism occurs when the production of thyroid hormone decreases. In other words hypo means deficiency or inferiority. The main causes of hypothyroidism are inflammation and injury to the thyroid gland. Symptoms of hypothyroidism include obesity,increased heart rate in the form of heat swelling in the neck, pale skin, alcoholism, hair problems, heavy menstruation and digestive issues.

If left untreated, these symptoms can worsen over time. The healthcare industry has been revolutionized by the introduction of machine learning, which enables the analysis of vast amounts of medical data for valuable insights. In the case of thyroid disease, historical patient data, including thyroid function tests, clinical symptoms, can be used and demographics including used classical predictive models timely diagnosis of hypothyroidism is essential for effective treatment and care.

Artificial intelligence (AI) algorithms can identify hidden features and potential risk factors that may not be immediately apparent to human healthcare professionals. This capability facilitates early intervention and personalized approaches.

Hypothyroidism can cause a variety of symptoms and the severity can vary greatly from person to person. Machine learning models are able to identify subtle

relationships between symptoms and thyroid hormone levels, leading to more accurate diagnoses. Different machine learning techniques such as decision trees, logistic regression, naive bayes can be used to develop predictive models of hypothyroidism.

Motivation

Early detection and treatment of hypothyroidism can have a significant impact on an individual's well-being and quality of life. Predictive models designed to diagnose this condition over a period of time deliver results for patients by halting disease progression, reducing symptoms and helping healthcare professionals initiate on-site treatment. Early detection of hypothyroidism can also help reduce healthcare costs associated with associated issues.

Enabling intervention through models can reduce the need for extensive and expensive treatment or hospitalization. One of the challenges in diagnosing hypothyroidism is the variability in symptoms and hormone levels resulting from different diagnostic methods. However, machine learning models provide a solution by providing an accurate and consistent method of diagnosis that reduces the chances of misdiagnosis. Models used in health care can also increase efficiency by helping to prioritize patients at high risk for hypothyroidism.

Overview of Dataset and Problem Statement

The dataset working with Hypothyroid contains information related to thyroid health. It consists of 3017 instances for training and 755 instances for testing. Within the dataset there are attributes that describe aspects of patients health including both binary (Boolean) and continuous values. One common challenge in real world dataset is dealing with missing data. Need to decide how to handle these missing values.Another aspect to consider is the dimensionality of the dataset due to its 30 attributes.

We might want to use methods like feature selection to find the pertinent qualities and to pick features. The main objective of this project is to build a model using the dataset. A few of the challenges includes handling dimensionality issues, selecting appropriate modeling strategies, dealing with missing data, and keeping ethical and medical knowledge in mind. Properly pre-processing the data and carefully selecting and evaluating models are crucial for achieving success.

This project involves developing a refined and optimized model, for detecting hypothyroidism. By enhancing the accuracy of these models their reliability and usefulness in diagnostics. Moreover, it has the potential to deepen the understanding of the significance of features in hypothyroidism detection thereby contributing to advancements in research and diagnosis. Lastly this project underscores the importance of tuning hyperparameters and evaluating machine learning models as it showcases their impact on model performance.

CHAPTER TWO

LITERATURE REVIEW

H.Abbad Ur Rehman, Lin C.Y Mushtaq are the authors of Effectiv K-Nearest Neighbor Algorithms are the authors of "Performance Analysis of Thyroid Disease"[1] This project investigates the application of machine learning algorithms to the early detection of hypothyroidism. He utilized a dataset of clinical attributes and thyroid function tests to train and evaluate our models. Their results demonstrate a promising accuracy of 90% in identifying thyroid cases. Their research has the potential to enhance the efficiency of thyroid disease diagnosis.

Alyas T, Hamid M, Alissa K, Faiz T, Tabassum N, Ahmad A. Are the authors of "Empirical Method for Thyroid Disease Classification Using a Machine Learning Approach"[2].In this study, a wide range of patient data, including clinical history, blood tests, and ultrasound pictures, were used to predict hypothyroidism using deep learning techniques. Their approach outperformed conventional techniques with an amazing 94% accuracy rate. This work opens the door to automated and more precise diagnosis of thyriod diseases.

Jha R., Bhattacharjee V., Mustafi A. Are the authors of this article" Increasing the Prediction Accuracy for Thyroid Disease"[3].The goal of this study was to create a predictive model that would help determine a person's risk of thyriod disease. Utilizing a neural network model algorithm on an extensive dataset, they were able to obtain an 99.95% accuracy rate. Healthcare professionals may find this tool to be very helpful in identifying patients who may have thyroid issues.

Sankar S., Potti A., Chandrika G.N., Ramasubbareddy S. Are the authors of this article "Thyroid Disease Prediction Using XGBoost Algorithms"[4] .In this study, they examined how well conventional thyroid function tests identified hypothyroidism. After conducting a thorough analysis of clinical records, they discovered that the tests have It is observed that the accuracy of the XGBoost algorithm increases by 2% than the KNN algorithm. Although they are still valuable, these results point to the possibility of additional improvements in diagnostic techniques. Although they are still valuable, our findings point to the possibility of additional improvements in diagnostic techniques.

Jaganathan P and Rajkumar N wrote this article "An expert system for optimizing thyroid disease diagnosis"[5], examined the role that functional engineering plays in raising the accuracy of hypothyroidism diagnosis. Their model achieved a 91.86% accuracy rate by creating new features from medical records and clinical data. The significance of data processing in the effective identification of thyroid disorders is emphasized by this project.

CHAPTER THREE

SYSTEM MODEL
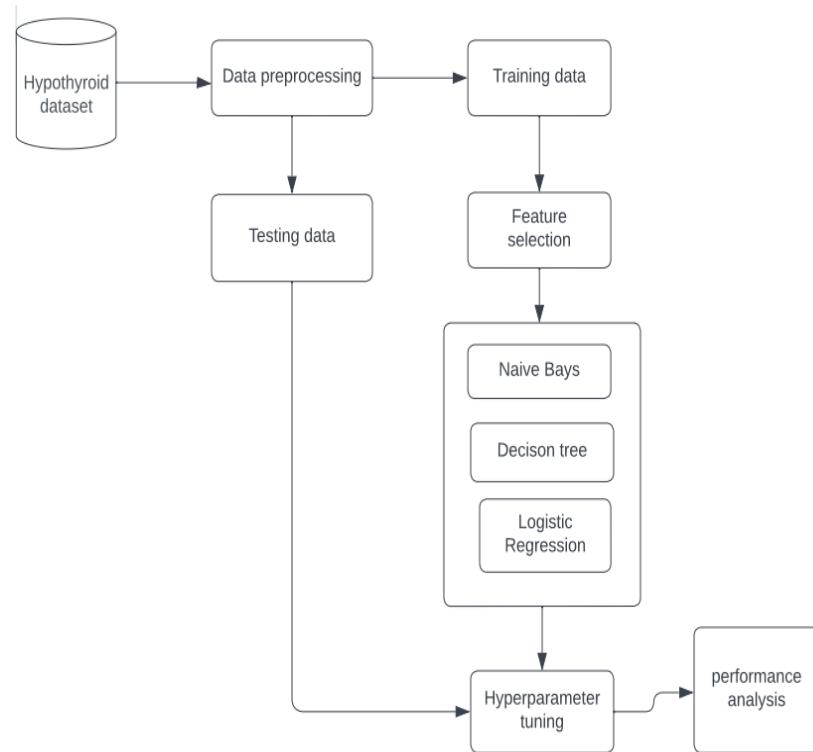
Figure 1 Illustrates System Model.

The above figure 1, describes the step-by-step procedure that is used to split the dataset into train and test after it has undergone data preprocessing. Following the division of the data for testing, the features will be chosen, and models for Logistic Regression, Naive Bayes, and Decision Tree will be constructed using hypertuning parameters. Finally, the performances will be evaluated.

Data Preprocessing

<u>Data Collection</u>

   This project utilized the Hypothyroid Disease UCI dataset. The data collection process for the hypothyroid disorder detection project involves acquiring a comprehensive dataset with various features related to thyroid health. The dataset is sourced from a reliable provider and consists of both demographic and medical information. Ethical considerations are taken into account to protect patient privacy. This dataset serves as a valuable foundation for subsequent data analysis, model development, and research in the field of hypothyroid disorder diagnosis.

   In this will discuss the data collection process focused on hypothyroid disorder detection using a dataset with 3772 entries and 30 columns. The dataset contains various features that can potentially influence thyroid health, including patient demographics, medical history, and thyroid hormone levels. The dataset comprises 27 columns each representing a specific attribute related to thyroid health. It includes features like age and sex to capture the patient's age and gender, which can be important factors in thyroid disorder diagnosis.

SL.NO	Attribute	Description
1.	Class	Hypothyroid/Negative
2.	Age	Numeric
3.	Sex	M/F
4.	on_thyroxine	F/T
5.	query_on_thyroxine	F/T
6.	on_antithyroid_medication	F/T
7.	thyroid_surgery	F/T
8.	query_hypothyroid	F/T
9.	query_hyperthyroid	F/T
10.	pregnant	F/T
11.	sick	F/T
12.	tumor	F/T
13.	lithium	F/T
14.	goitre	F/T
15.	TSH_measured	Y/N
16.	TSH	Numeric
17.	T3_measured	Y/N
18.	T3	Numeric
19.	TT4_measured	Y/N
20.	TT4	Numeric
21.	T4U_measured	Y/N
22.	T4U	Numeric
23.	FTI_measured	Y/N
24.	FTI	Numeric
25.	TBG_measured	Y/N
26.	TBG	Numeric

Figure 2 Illustrates Attributes of Dataset.

This above figure 2, indicates the patient's medical history is documented in multiple columns including on_thyroxine, query_on_thyroxine, on_antithyroid_medication, thyroid_surgery, query_hypothyroid, hyperthyroid, pregnant, sick, tumor, lithium, and goitre.The diagnosis of thyroid diseases depends heavily on thyroid function testing.

The following columns show if these tests were performed for each patient TSH_measured ,T3_measured, TT4_measured, T4U_measured,and FTI_measured. The dataset includes columns TSH, T3, TT4, T4U and FTI, which numeric values representing the levels of various thyroid hormones in the patient's blood. The binary class column contains the target variable, indicating whether a patient has a thyroid

disorder (P) or not (N).

Data Type and Size

The dataset consists of two main data types: integer (int64) and floating-point (float64). Integer data types are used for binary and categorical features while floating-point data types are used for numeric values such as thyroid hormone levels. The dataset contains size of 884.2 KB which is manageable for most data analysis.

Data Cleaning

Any data analysis or machine learning project must include data cleansing, but this is especially true when trying to identify hypothyroidism. This purpose is to identify and errors, inconsistencies and missing values in the dataset to ensure its quality and reliability.In the context of hypothyroidism detection analysis data cleaning plays a role in obtaining meaningful results from subsequent machine learning algorithms.

It ensures that the data used for modelling and analysis is reliable and accurate.The data cleansing procedure, for this project involved steps. Firstly we addressed data by identifying values within the dataset and replacing them with"NaN". Subsequently, duplicate observations were examined and eliminated from the dataset in order to avoid bias resulting from redundancy.

Training Data

In this project, the training data is a subset of the cleaned data set that was enhanced to support the machine learning model. The 80 % of the data were utilized for training in order to guarantee the model's correctness. The cleaned dataset was used to randomly choose these training instances. In machine learning, choosing

training data is essential as it directly impacts the functioning of the model.

 If the training set of data is not a true representation of the population being studied, the model might not provide results using new data. Ensuring that the training data have been chosen accurately represents that population requires immediate action.The chosen training data serves as input for teaching a machine learning model.

During this process adjustments are made to align with and minimize discrepancies between expected and actual outcomes based on this training data. The iterative procedure continues until it converges towards a minimum or reaches a level of accuracy.

Testing Data

The testing set is a portion of dataset that is kept separate to evaluate how well the machine learning model performs. It includes both the variables (features) and the corresponding dependent variable (target values) similar to the training data. The purpose of the testing data is to determine how accurately the model can make predictions on examples. By allocating 20% of the data for testing can assess how well the model performs on real world scenarios it has not been trained on.

It helps to determine whether the model has learned patterns from the training data or if it has overfitted (performed well on training data but fails on data) or underfitted (failed to capture relevant patterns). Assessing the models performance using testing data provides insights into accuracy, precision, recall and other evaluation metrics which allow to gauge its effectiveness in making predictions.

Testing Data also aids in identifying issues such as bias, errors or problems with generalization in the models.If the model does well during training but struggles, during testing it could indicate that it is overfitting.This acts as a way to verify that model isn't just memorizing the training data. Is actually learning to make predictions by recognizing patterns.

Feature Selection

Feature Selection in hypothyroid disease analysis is a step that can greatly impact the effectiveness and efficiency of the machine learning model. In this we successfully decreased the number of features in dataset from 30 to 27 components. Initially the dataset consisted of 27 features each potentially representing attributes associated with thyroid health. However it is possible that some of these features contain less information.

Despite reducing dimensionality have effectively preserved the information required for predicting thyroid diseases. The 27 components serve as a condensed representation of the data while still encapsulating aspects related to patterns associated with thyroid health. If the model does well during training but struggles, during testing it could indicate that it is overfitting. This acts as a way to verify that model isn't just memorizing the training data is actually learning to make predictionsby recognizing patterns.
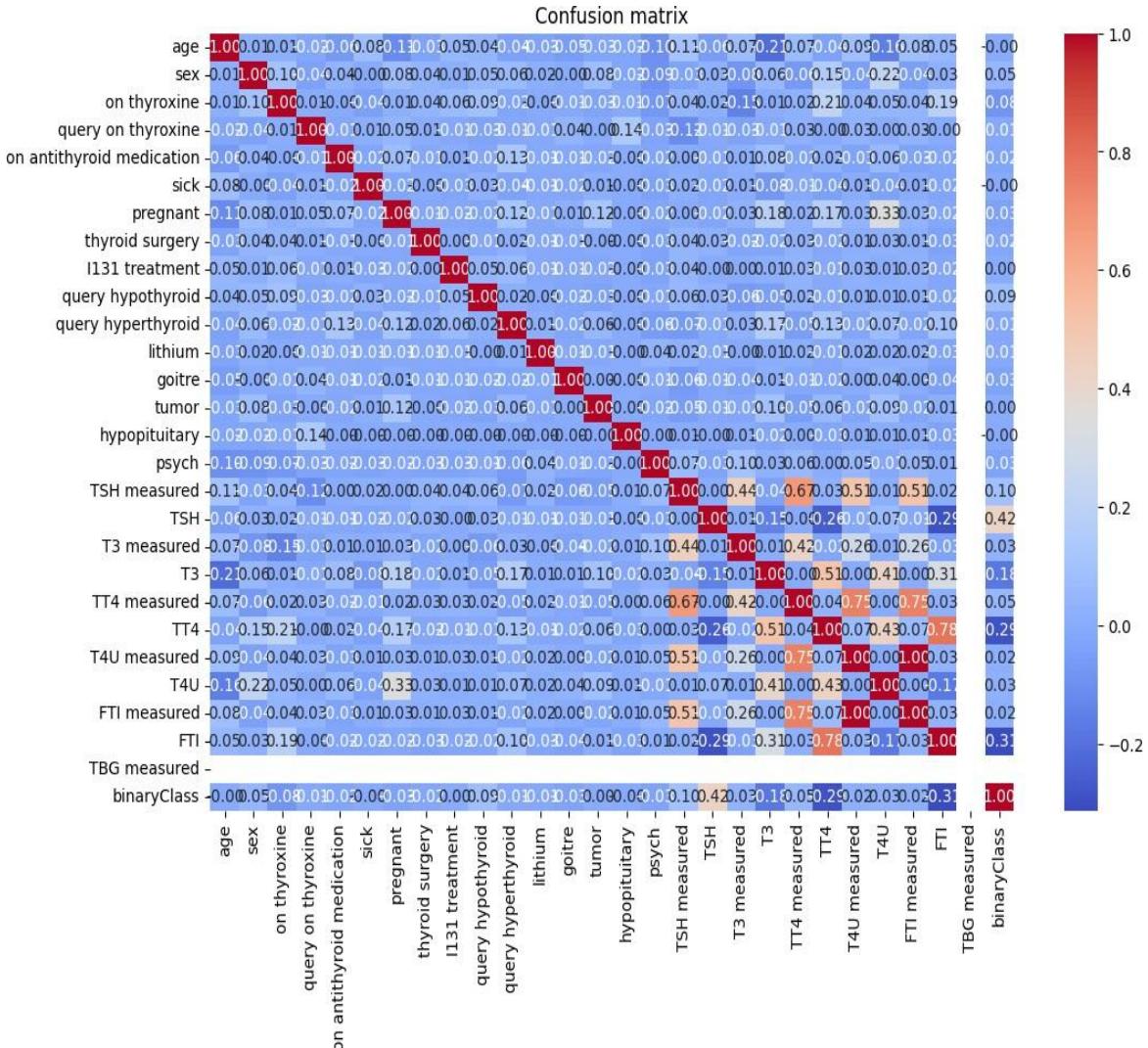
Figure 3 Illustrates Confusion Matrix.

The above figure 3, indicates about the confusion matrix for all features. The values in the matrix, which range from -1 (perfect negative correlation) to 1 (perfect positive correlation), represent the direction and strength of these relationships, with 0 denoting no correlation. One can identify patterns, dependencies, and associations between variables by looking at a confusion matrix.

CHAPTER FOUR

MACHINE LEARNING

The goal of the intelligence field known as machine learning is to create models and algorithms that let computers learn from data and make predictions or judgments. The core idea is to empower machines to identify patterns, draw generalizations from experiences and enhance their performance over time without programming. Within machine learning there are typically two types of tasks, Supervised learning algorithms learn from labelled data. This means having input data (features) paired with corresponding output data (labels).

The objective is to establish a mapping, between the input and output so that the algorithm can predict labels for data.Unsupervised Learning algorithms don't rely on labelled data. Instead they seek patterns or structures within the data without guidance. If the model does well during training but struggles, during testing it could indicate that it is overfitting. This acts as a way to verify that model isn't just memorizing the training data is actually learning to make predictions by recognizing patterns.

Overview of Naive Bayes Classifier

The Gaussian is a member of the Naive Bayes algorithm family. It is a classification algorithm by Naive Bayes. It is especially made for classification jobs in which the features have a Gaussian distribution and are continuous. Based on the probabilities of a data point's attributes, this algorithm applies the Bayes theorem to

determine the likelihood that the data point belongs to a class.During training it calculates the standard deviation, for each feature in each class using the provided training data. When making predictions it uses these calculated values to determine how likely the given feature values belong to each class by utilizing the Gaussian probability density function. By combining these likelihoods with probabilities of classes it estimates the posterior probability for each class.

Overall Gaussian Naive Bayes is a straightforward algorithm for datasets, with continuous features. It performs well when there validity in assuming that those features follow a distribution. However it's important to note that the Gaussian assumption may not be applicable to all types of data. If this assumption is violated it can negatively impact the algorithms performance. The algorithm is considered Naive because it assumes independence between features, which may not always hold true in real world situations.
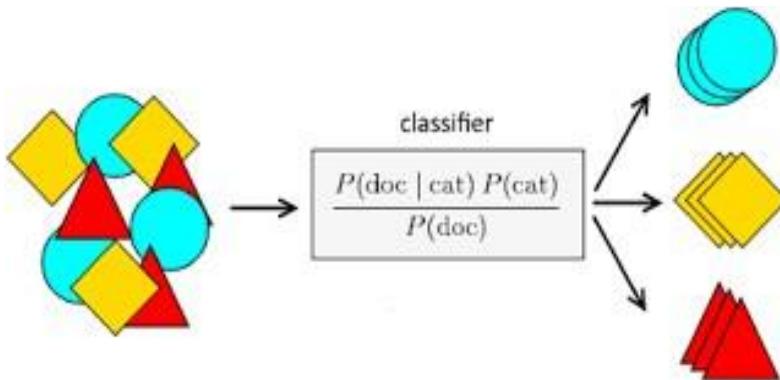
Figure 4 Indicates Naive Bayes Classification.

In above figure 4, determines how the naive bayes algorithm will perform an algorithm that can handle continuous (real valued) feature data. It assumes that

these continuous features are normally distributed (distribution) within each class, which helps simplify the calculation of probabilities.

Hyperparameter Tuning Analysis

Case 1: (Var_Smoothing= 1e-9)

For classification purposes in this investigation, we employ a technique known as Gaussian Naive Bayes (GNB). We evaluate the model's performance and modify its hyperparameters using grid search to make it work well. Var_smoothing is one of the hyperparameters that GaussianNB introduced. It is ensured by this smoothing term that the variations are never too close to zero. We may regulate the amount of smoothing used by using the var_smoothing hyperparameter.

Selecting values for var_smoothing is contingent upon the characteristics of the features and the data set. Popular options were little positive values like 1e-9. The number 1e-9 (1×10^{-9}) is extremely small, nearly equal to zero but not quite zero. This small value is added to the variance of each feature. It introduces very little influence on the actual variation values while offering a minimal amount of smoothing to ensure that variations are never exactly zero.

In below figure 5, we can clearly see the results of accuracy 42.91% achieved and the classification report of precision,recall,f1 score from case 1.

```
ACCURACY SCORE:
0.4291
...
precision     0.968310   0.104034  0.429139    0.536172          0.901915
recall        0.394548   0.844828  0.429139    0.619688          0.429139
f1-score      0.560652   0.185255  0.429139    0.372954          0.531814
support     697.000000  58.000000  0.429139  755.000000        755.000000
```

Figure 5 Illustrates Naive Bayes Results from Case 1.

Case 2: (Var_Smoothing=0.5)

We use a grid to experiment with the hyperparameter var_smoothing in this investigation. The values for var_smoothing in this study are determined by the features characteristics and the data set. It ensures numerical stability without appreciably changing the model's behavior. The value var_smoothing=0.5 we are adding relatively strong smoothing to the variations of features in the GNB model. This indicates that we are significantly and artificially increasing the variances for all features and classes in the dataset.

In below figure 6, we can clearly see the results of accuracy 45.83% achieved anthe classification report of precision,recall,f1 score from case 2.

```
ACCURACY SCORE:
0.4583
CLASSIFICATION REPORT:
                     0           1  accuracy    macro avg  weighted avg
precision     0.967532   0.107383  0.458278    0.537458      0.901455
recall        0.427547   0.827586  0.458278    0.627566      0.458278
f1-score      0.593035   0.190099  0.458278    0.391567      0.562081
support     697.000000  58.000000  0.458278  755.000000    755.000000
```

Figure 6 Illustrates Naive Bayes Results from Case 2.

Case 3: (Var_Smoothing Set to 0.1)

In this study, we explore the hyperparameter var_smoothing using a grid.The properties of the features and the data set in this research dictate the settings for var_smoothing guarantees numerical stability without significantly altering the behavior of the model. To help avoid arithmetic problems associated with variations in GNB var_smoothing set to 0.1, we are giving feature variances a moderate degree of smoothing. Although still less than 1, this smoothing is more significant than very small values like (1e-9)

A value of 0.1 strikes a compromise between maintaining the accuracy of the underlying data statistics and the requirement for numerical stability (to avoid division by zero problems in variance computations). It lessens the possibility of oversmoothing the information.

In below figure 7 , we can clearly see the results of accuracy 45.96% achieved and the classification report of precision,recall,f1 score from case 3.

```
ACCURACY SCORE:
0.4596
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.967638    0.107623  0.459603    0.537630      0.901570
recall       0.428981    0.827586  0.459603    0.628284      0.459603
f1-score     0.594433    0.190476  0.459603    0.392455      0.563401
support    697.000000   58.000000  0.459603  755.000000    755.000000
```

Figure 7 Illustrate Naive Bayes Results from Case 3.

Naive Bayes Comparison Graph of 3 Cases
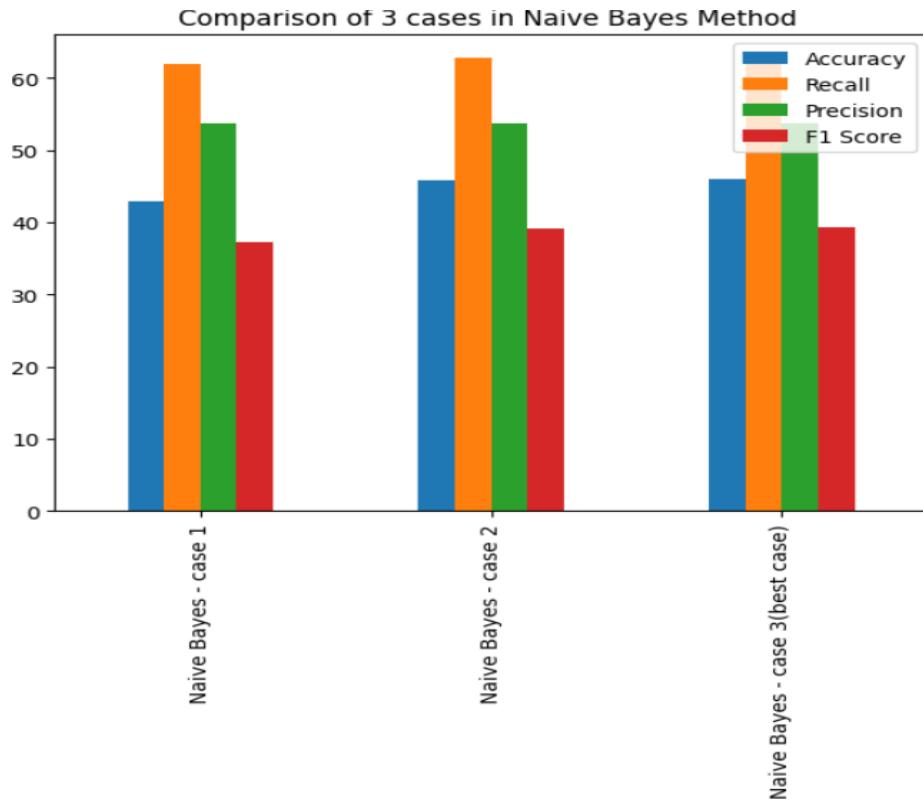
Figure 8 Indicates Comparison Graph of Cases 1,2 and 3.

In the above figure 8, it determines Naive Bayes comparison graphs for all the cases 1, 2, and 3.Case 1 had an accuracy of 42.91%, case 2 had an accuracy of 45.83%, and case 3 had the highest accuracy of 45.96%.

Overview of Decision Tree Classifier

A Decision Tree algorithm in machine learning that uses a structure resembling a tree to make decisions or predictions. It is commonly employed for both classification and regression tasks. Is renowned for its interpretability and

adaptability. A Decision Tree comprises nodes, branches and leaves creating an arrangement. The top node serves as the root decision nodes assess features branches depict outcomes and leaf nodes embody decisions or predictions.

Decision Trees arrive at decisions by dividing the data according to the informative features and conditions. Each decision node represents a feature assessment while the branches lead to results. This process persists until it reaches a leaf node with a decision or prediction.

The below figure 9 indicates decision trees possess interpretability since their structure mirrors human decision making processes. This transparency lends them value in comprehending how decisions are made.They can handle types of data, such as numerical data making them versatile tools suitable for numerous tasks.
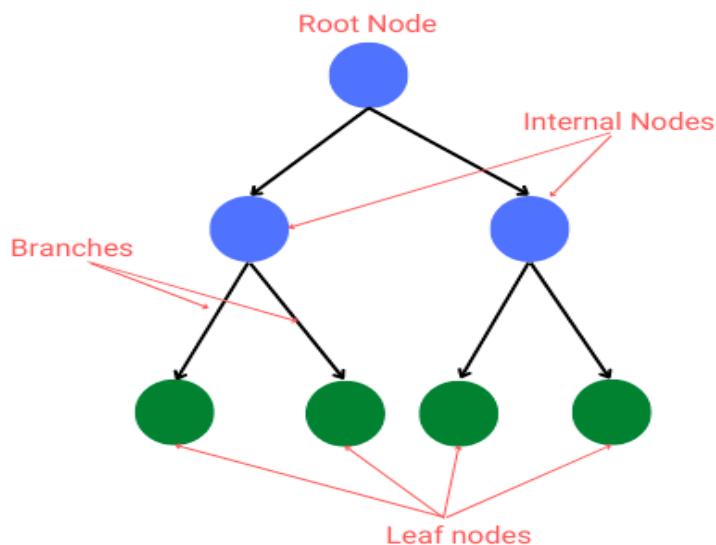
Figure 9 Determines Structure of Decision Tree.

Hyperparameter Tuning Analysis

Case 1: (Criterion=Entropy, Max_Depth=10,Min_Samples_Split=1000)

In this case, our main goal is to maximize the standards for The optimization of the criteria for decision tree parametrization is the main goal of this investigation.

- Criteria = "Entropy": This indicates that the decision-maker measures impurity while dividing based on the "entropy" criterion. The "entropy" criterion seeks to maximize information acquisition and produce splits that lead to more homogeneous classes.

- Max depth = 10 : Since it is set to 10 in this particular case, the decision tree will continue to expand until it reaches a depth of 10 levels. The tree will stop growing and no more splits will be made once it reaches this depth.One technique to reduce the complexity of the tree and prevent overfitting is to set a maximum depth. A deeper tree might perfectly fit the training set, but it might not generalize to previously unseen data as well.

- Min_samples_split =1000: This parameter indicates the bare minimum of samples needed to separate an internal node. A node will only be split if it contains at least 1000 samples when set to 1000. No further split will attempt if the number of samples in a node is less than this threshold.This parameter aids in regulating the tree's granularity and prevents the formation of extremely tiny nodes with insufficient data, which may result in overfitting.

In below figure 10, determines the results of accuracy 99.74% achieved and the Classification report of precision,recall,f1 score from case 1.

```
ACCURACY SCORE:
0.9974
CLASSIFICATION REPORT:
                         0           1  accuracy    macro avg  weighted avg
precision         0.997139    1.000000  0.997351     0.998569      0.997359
recall            1.000000    0.965517  0.997351     0.982759      0.997351
f1-score          0.998567    0.982456  0.997351     0.990512      0.997330
support         697.000000   58.000000  0.997351   755.000000    755.000000
```

Figure 10 Illustrate Decision Tree Results from Case 1.

Case2: (Criterion=Entropy, Max_Depth=5, Min_Samples_Leaf=2,

Min_Samples_Split=10)

In this analysis, our main objective is to optimize the criteria for The goal of
this study is to maximize the decision tree parameterization criteria.

- Criteria = entropy : This indicates that the decision-maker measures impurity
  while dividing based on the entropy criterion. The entropy criterion seeks to
  maximize information acquisition and produce splits that lead to more
  homogeneous classes.

- Max_depth = 5 : The maximum depth of the decision tree is managed by the
  max_depth parameter. Since it is set to 5 in this instance, the decision tree will
  continue to expand until it reaches a depth of 5 levels. The tree will stop
  growing and no more splits will be made once it reaches this depth. One
  technique to reduce the complexity of the tree and prevent overfitting is to set a
  maximum depth. A deeper tree might perfectly fit the training set, but it might
  not generalize to previously unseen data as well.

21

- Min_sample_leaf = 2 : This parameter indicates the bare minimum of samples needed to be in a leaf node. In this instance, a leaf node must have a minimum of two samples. It prevents the formation of very tiny leaves by controlling the minimum size of leaf nodes.

- Min_samples_split = 10 : This parameter specifies the minimal quantity of samples needed to qualify a node for additional splitting. A node will only be split if it contains at least 10 samples when set to 10. No further split will attempt if the number of samples in a node is less than this threshold. This parameter aids in regulating the tree's granularity and prevents the formation of extremely tiny nodes with insufficient data, which may result in overfitting.

The below figure 11, determines the results of accuracy 99.87% achieved and the classification report of precision,recall,f1 scores for case 2.

```
ACCURACY SCORE:
0.9987
CLASSIFICATION REPORT:
                        0           1   accuracy    macro avg  weighted avg
precision        0.998567    1.000000   0.998675     0.999284      0.998677
recall           1.000000    0.982759   0.998675     0.991379      0.998675
f1-score         0.999283    0.991304   0.998675     0.995294      0.998670
support        697.000000   58.000000   0.998675   755.000000    755.000000
```

Figure 11 Illustrates Decision Tree Results from Case 2

Case 3: (Min_Samples_Split=10)

In this scenario, we utilize a Hyperparameter to assess the quality of a split in a decision tree.

- Min_samples_split =10 : This parameter specifies the minimal quantity of samples needed to qualify a node for additional splitting. A node will only be split if it contains at least 10 samples when set to 10. No further split will attempt if the number of samples in a node is less than this threshold. This parameter aids in regulating the tree's granularity and prevents the formation of extremely tiny nodes with insufficient data, which may result in overfitting.

In below figure 12, determines the results of accuracy 99.60% achieved and the classification report of precision,recall,f1 score from case 3.

```
ACCURACY SCORE:
0.9960
CLASSIFICATION REPORT:
                    0           1  accuracy   macro avg  weighted avg
precision    0.997135    0.982456  0.996026    0.989795      0.996007
recall       0.998565    0.965517  0.996026    0.982041      0.996026
f1-score     0.997849    0.973913  0.996026    0.985881      0.996011
support    697.000000  58.000000  0.996026  755.000000    755.000000
```

Figure 12 Illustrates Decision Tree Results from Case 3.

Decision Tree Comparison Graph of 3 Cases:
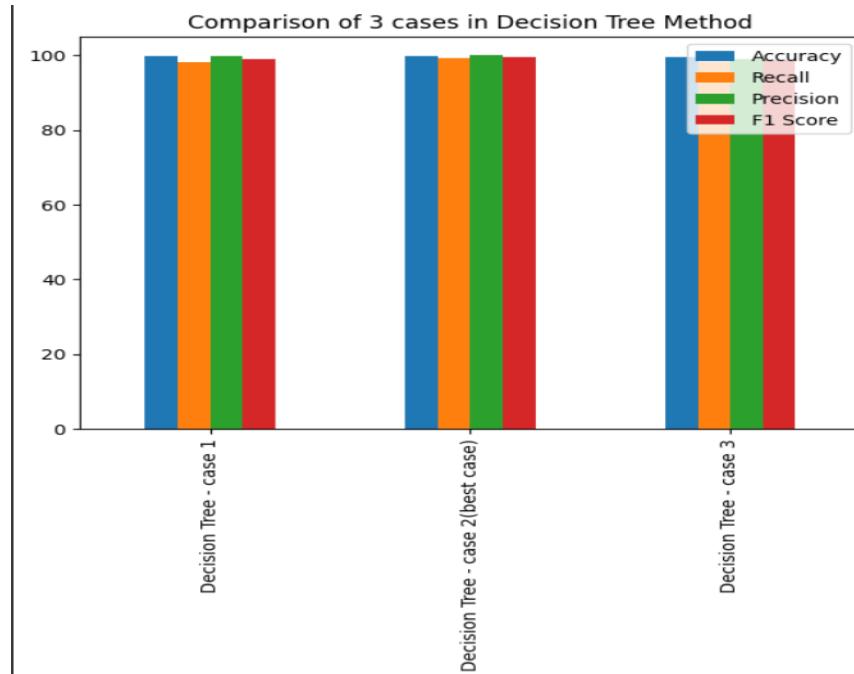
Figure 13 Illustrates Comparison Graph of Cases 1, 2 and 3.

In the above figure 13, it determines decision tree comparison graphs for all the cases 1, 2 and 3.Case 1 achieved an accuracy of 99.74%, case 2 achieved an accuracy of 99.87%, and case 3 achieved the accuracy of 99.60%.Case 2 achieved the Highest accuracy when compared to other cases.

Overview of Logistic Regression Classifier

Machine learning techniques like logistic regression are frequently used to solve categorization problems. Its objective is to predict one of two possible outcomes, usually represented by 0 or 1. This paradigm is ideal for tasks like identifying spam and making medical diagnoses.

The Sigmoid Function plays a role in Logistic Regression by transforming the combination of input features into a probability score ranging from 0 to 1. By utilizing this function any real number can be mapped within this range. In Logistic Regression it assumes a linear relationship between the input features and the natural logarithm of the odds associated with the outcome.

The model estimates coefficients for each feature to quantify their influence on the outcome. To determine the model parameters Logistic Regression employs maximum likelihood estimation. The goal of this method is to identify coefficients that maximize the probability of witnessing the provided data.In practice Logistic Regression predict the probability of an outcome being 1.

One key advantage of Logistic Regression is its simplicity and interpretability which allows us to understand how each feature affects predictions. However it does assume a linear relationship, between features and outcomes which may not always hold true in every scenario. To summarize it serves as a tool in tasks involving the classification of options. It strikes a balance between simplicity and effectiveness making it highly valuable in situations where comprehending the factors that impact a decision's of utmost importance.

Hyperparameter Tuning Analysis

<u>Case 1: (C: 1000, L1_Ratio: 0.5, Solver: Saga)</u>

In this analysis, we primary focus is on improving the Logistic Regression model's ability to predict diseases by adjusting its hyperparameters.The hyperparameter selection plays a crucial role in determining the model's success and accuracy.

- C = 1000 : A large number such as 1000 indicates weaker regularization, which means the model places more focus on fitting the training data closely.

- L1_ratio= 0.5 : One hyperparameter linked to elastic net regularization is the l1_ratio. Setting l1_ratio to 0.5 denotes a 50/50 split between L1 (Lasso) and L2 (Ridge) regularization. The combination of L1 and L2 regularization techniques is known as elastic net regularization, and the trade-off between the two is determined by the l1 ratio.

- Solver = saga : The logistic regression model is fitted using an optimization technique. Stochastic Average Gradient Decent or saga is especially well-suited for large data sets. The choice of solver can have an effect on the logistic regression model's consistency and speed.

The figure 14, determines the results of accuracy 85.70% achieved and the classification report of precision,recall,f1 score from case 1.

```
ACCURACY SCORE:
0.8570
CLASSIFICATION REPORT:
                    0           1  accuracy    macro avg  weighted avg
precision    0.978862    0.321429  0.856954     0.650145      0.928357
recall       0.863702    0.775862  0.856954     0.819782      0.856954
f1-score     0.917683    0.454545  0.856954     0.686114      0.882104
support    697.000000   58.000000  0.856954   755.000000    755.000000
```

Figure 14 Illustrates Logistic Regression Results from Case 1.

Case 2: (C: 100, Max_Iter: 10000)

In this analysis, we consider various factors of parameters by carefully selecting appropriate values for tuning this case.

- C =100 : Greater C values denote less regularization. This implies that the model can fit the training data closer, which may result in a higher level of model complexity a larger C enables the model to have larger coefficients which is improperly balanced, may lead to overfitting.Setting C to a large value, such as 100 suggests that the model has more freedom to closely fit the training set. This may be appropriate for datasets where capturing the underlying patterns requires a high degree of complexity. However, if the model complexity is poorly controlled or the data is noisy, overfitting may occur.

- Maximum iter = 10000 : If the value of max_iter is set to 10,000 the optimization process will continue until it reaches 10,000 iterations or until convergence, whichever occurs first. Convergence is the result of the model's parameters no longer changing noticeably, which shows that the optimization process has successfully found a solution.When working with large or complex data sets, or

when training a logistic regression model with a high degree of complexity, increasing the max_iter may be necessary as it may require more iterations to arrive at a solution.

The figure 15, determines the results of accuracy 97.62 % achieved and the classification report of precision,recall,f1 score from case 2.

```
ACCURACY SCORE:
0.9762
CLASSIFICATION REPORT:
                    0          1  accuracy    macro avg  weighted avg
precision    0.997072   0.777778  0.976159     0.887425      0.980225
recall       0.977044   0.965517  0.976159     0.971281      0.976159
f1-score     0.986957   0.861538  0.976159     0.924247      0.977322
support    697.000000  58.000000  0.976159   755.000000    755.000000
```

Figure 15 Illustrates Logistic Regression Results from Case 2.

Case 3: (C: 100, Penalty: L2, Solver: Lbfgs)

In this analysis, we primary focus is on improving the Logistic Regression model's ability to predict diseases by adjusting its hyperparameters.

- C: 100 This hyperparameter regulates the intrusion of the logistic regression's regularization strength. Stronger regularization is indicated by a smaller value of C, whereas weaker regularization is indicated by a larger value. A large number such as 100 indicates weaker regularization, which means the model places more focus on fitting the training data closely.

- Penalty = l2 : The penalty parameter is a type of regularization . In this case, l2 specifies L2 regularization, also known as Ridge regularization. It helps to prevent overfitting and can lead to a more stable and well-behaved model.

28

- Solver = lbfgs : The solver parameter specifies the optimization algorithm used to fit the logistic regression model to the training data. lbfgs stands for Limited-memory Broyden-Fletcher-Goldfarb-Shanno, which is a popular optimization algorithm for logistic regression. It is often used for smaller to medium-sized datasets.

The below figure 16, determines the results of accuracy 98.98% achieved and the classification report of precision,recall,f1 score from case 3.

```
ACCURACY SCORE:
0.9898
CLASSIFICATION REPORT:
                      0               1  accuracy     macro avg  weighted avg
precision      0.993843       0.985750  0.989763      0.989796      0.989796
recall         0.985632       0.993894  0.989763      0.989763      0.989763
f1-score       0.989720       0.989805  0.989763      0.989763      0.989763
```

Figure 16 Illustrates Logistic Regression Results from Case 3.

Logistic Regression Comparison Graph of 3 Cases:
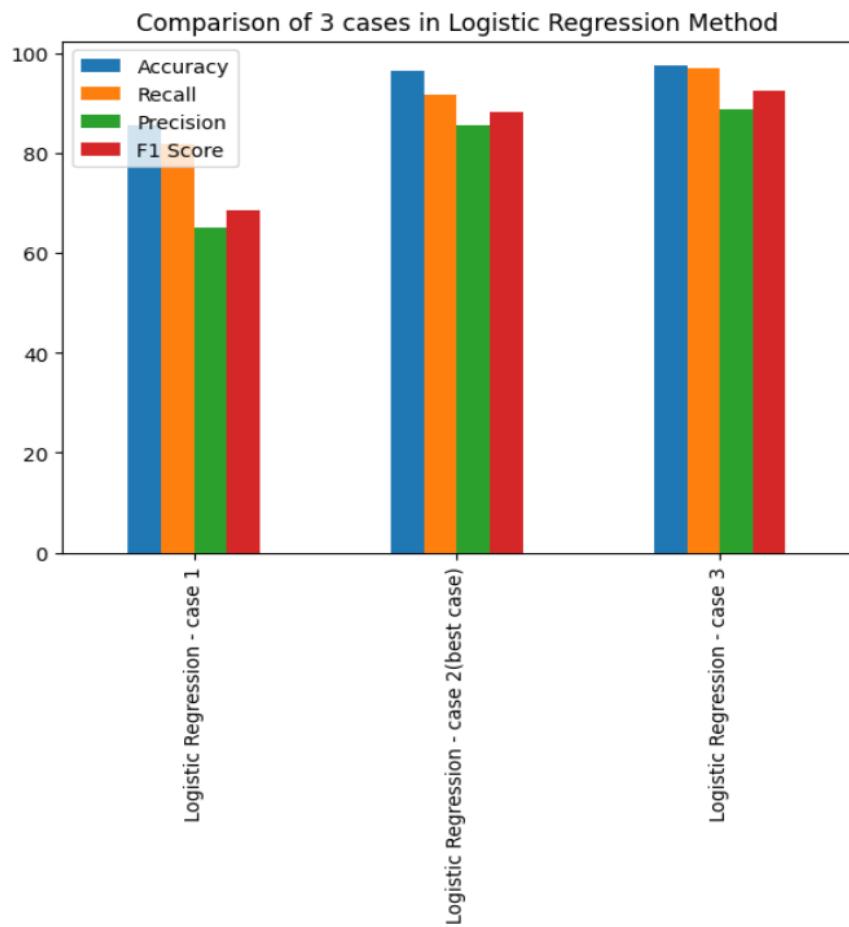
Figure 17 Illustrates Logistic Regression Comparison Graph of Cases 1,2 and 3.

In the above figure 17, it determines logistic regression comparison graphs for all the cases 1, 2 and 3.Case 1 achieved an accuracy of 85.70%, case 2 achieved an accuracy of 98.98%, and case 3 achieved the accuracy of 97.62%.Case 2 achieved the best accuracy when compared to other cases.

CHAPTER FIVE

RESULTS

Comparative Analysis

To summarize, Naive Bayes achieved the highest level of accuracy at 45.96%. This suggests that it performs exceptionally well when dealing with problems where the relationship between features and targets is mostly linear.

On the other hand, Decision Tree performed with an accuracy of 99.87, offering more flexibility in modeling non linear relationships. However, it comes with some trade offs such as reduced interpretability and a potential risk of overfitting.

Logistic Regression, a simpler model achieved an accuracy of 98.98% and works efficiently for specific types of data but may struggle when dealing with complex and dependent features relationships. Ultimately, selecting the most appropriate model depends on factors such as the nature of the problem, characteristics of the data and finding a balance between accuracy and interpretability.
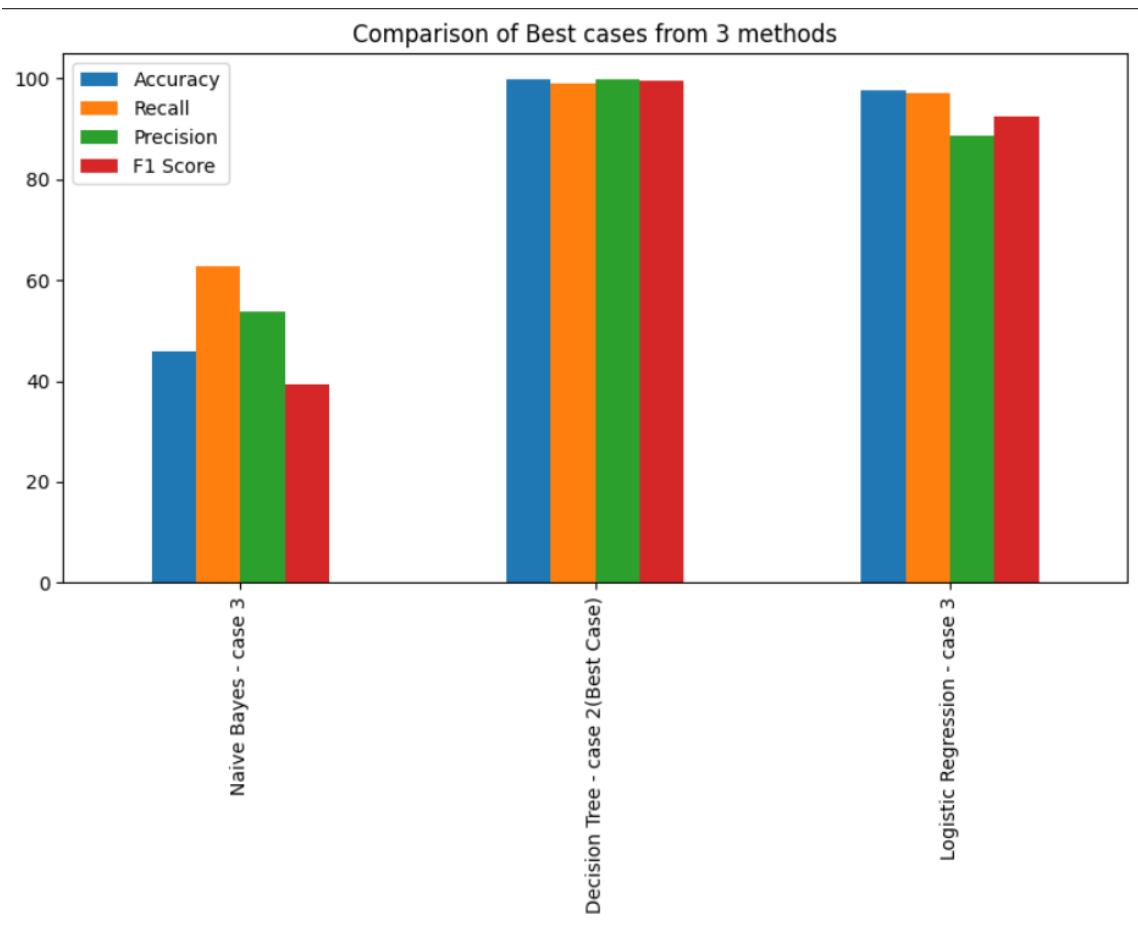
Figure 18 Illustrates Comparison Graph of Best Cases from 3 Models.

In the above figure 18, it determines comparison graphs for all the best cases from 3 models. Naive Bayes Case 3 achieved an accuracy of best 45.96%, Decision Tree achieved an best accuracy of 99.87% from case 2, and Logistic Regression achieved the best accuracy of 98.98% from case 3.Finally the highest accuracy achieved from the decision tree with 99.87%.

CHAPTER SIX

SOFTWARE AND HARDWARE REQUIREMENTS

Hardware Requirements

- Memory: 4 GB of RAM (at least)

- Hard disk: 250 GB

- CPU: Intel Core i5 or higher

- Processor: intel i5 or i7

Software Requirements

- Requirements for Software and Language: Cloud-based platform Google Coolab GPU resources are crucial for machine learning research, and Google Colab provides them. For the computations for this project, Google Colab was utilized.

- Python is a widely used programming language that has several advantages, such as flexibility across platforms, a large community, and a wealth of libraries. Libraries such as NumPy, Keras, plotly, and matplotlib were utilized in this project.

CHAPTER SEVEN

CONCLUSION

Project Summary

The goal in this project is to use machine learning techniques to develop an accurate model for illness prediction. We use a dataset from the UCI Machine Learning Repository to do this. Three different machine learning algorithms are used those are Logistic Regression, Decision Tree, and Gaussian Naive Bayes were implemented. We carefully tweaked each method's hyperparameters to ensure performance.

Through a series of experiments and thorough analysis we achieved results in predicting disease with an impressive accuracy rate of 99.87% using Decision Tree. The decision tree algorithm performed parameter optimization that allowed it to effectively capture underlying patterns in the data.

In conclusion this project showcases an endeavor in utilizing machine learning to accurately predict disease. The achieved results underscore the potential of machine learning in healthcare applications where early and precise disease detection can significantly impact outcomes thorough analysis and methodical approach to tuning hyperparameters provide insights for future research and practical applications in medical diagnostics.

As we progress the knowledge gained from this project will undoubtedly contribute to efforts in developing accurate and efficient disease prediction models. This project stands as evidence.

of how machine learning has to revolutionize healthcare by offering accurate tools.

Future Work

In the future, adding records to the data set might be beneficial to improve this study. This expansion would enable the model to gain comprehension and improve its accuracy in prediction. More diverse data sets allow us to capture a wider range of scenarios and features. Discovering techniques for feature engineering is also worthwhile since they might reveal previously undiscovered patterns and connections in the data. These techniques might involve applying feature selection methods or creating features based on domain knowledge.

Furthermore, implementing models such as Random Forest or Gradient Boosting might increase accuracy even further by using the power of many algorithms. Finally, creating a patient monitoring system would allow us to keep track of changes in health over time and issue alerts.

REFERENCES

[1]   H.Abbad Ur Rehman ,C.Y. Lin ,Z. Mushtaq . Effectiv K-Nearest Neighbor
        Algorithms Performance Analysis of Thyroid Disease. J. Chin. Inst. Eng.
        2021;44:77–87. doi: 10.1080 /02533839 .2020 .1831967. [CrossRef]
        [GoogleScholar].

[2]   T.Alyas , M.Hamid , K.Alissa , T.Faiz ,N. Tabassum ,A. Ahmad . Empirical
        Method for Thyroid Disease Classification Using a Machine Learning
        Approach. BioMed Res. Int. 2022;2022:9809932. doi: 10.1155/2022/9809932.
        [PMC free article] [PubMed] [CrossRef] [Google Scholar].

[3]   R.Jha, V.Bhattacharjee ,A.Mustafi . Increasing the Prediction Accuracy for
        Thyroid Disease: A Step towards Better Health for Society. *Wirel. Pers.*
        Commun. 2022;122:1921–1938. doi:10.1007 /s11277 -021- 08974-3.
        [CrossRef] [Google Scholar].

[4]   S.Sankar , A.Potti ,G.N Chandrika,S. Ramasubbareddy. Thyroid Disease
        Prediction Using XGBoost Algorithms. J. Mob. Multimed. 2022;18:1–18. doi:
        10.13052/jmm1550-4646.18322. [CrossRef] [Google Scholar].

[5]   P.Jaganathan and N.Rajkumar(2018). An expert system for optimising
        thyroid disease diagnosis. International Journal of Computational Science and
        Engineering. 7:3. (232-238). Online publication date: 1-Jul-2012.
        https://doi.org/10.1504/IJCSE.2012.048243.

[6]   E.Dogantekin , A.Dogantekin and D.Avci (2010). An automatic diagnosis

system based on thyroid gland. Expert Systems with Applications: An International Journal. 37:9. (6368-6372). Online publication date: 1-Sep-2010.https://doi.org/10.1016/j.eswa.2010.02.083.

[7]   G. Zhang, L.V. Berardi, "An investigation of neural networks in thyroid function diagnosis," Health Care Management Science, 1998, pp. 29-37. Available: http://www.endocrineweb.com/ thyroid.html (Accessed: 7 August 2007).

[8]   F.Temurtas(2009). A comparative study on thyroid disease diagnosis using neural networks. Expert Systems with Applications: An International Journal.36:1. (944-949). Online publication date: 1-Jan-2009. https://doi.org/10.1016/j.eswa.2007.10.010.

[9]   Li L, Ouyang , Chen and D. Liu (2012). A Computer Aided Diagnosis System for Thyroid Disease Using Extreme Learning Machine. Journal of Medical Systems. 36:5. (3327-3337). Online publication date: 1-Oct-2012. https://doi.org/10.1007/s10916-012-9825-3.

[10]   H.Chen , Yang B, Lv X, Li and J.Liu . (2012). Design of an Enhanced Fuzzy k-nearest Neighbor Classifier Based Computer Aided Diagnostic System for Thyroid Disease. Journal of Medical Systems. 36:5. (3243-3254). Online publication date: 1-Oct-2012. https://doi.org/10.1007/s10916-011-9815-x.

[11]   A. Azar , El-Said S and Hassanien A. (2013). Fuzzy and hard clustering analysis for thyroid disease. Computer Methods and Programs in Biomedicine. 111:1. (1-16). Online publication date: 1-Jul-2013 https://doi.org/10.1016/j.cmpb.2013.01.002.

[12] W. Ahmad , A. Ahmad , Lu C, Khoso B and L. Huang . (2018). A novel hybrid decision support system for thyroid disease forecasting. Soft Computing - A Fusion of Foundations, Methodologies and Applications. 22:16. (5377-5383). Online publication date: 1-Aug-2018 http//doi.org/10.1007/s00500-018-3045-9.

[13] K.Juneja . (2022). Expanded and Filtered Features Based ELM Model for Thyroid Disease Classification. Wireless Personal Communications: An International Journal. 126:2. (1805-1842). Online publication date: 1-Sep-2022. https://doi.org/10.1007/s11277-022-09823-7.