

本科毕业论文

**A COMPETITIVE STUDY OF MACHINE LEARNING TECHNIQUES FOR CLASSIFYING THYROID DISORDERS**

|  |  |
| --- | --- |
| **学 院** | **School of Computer Science and Engineering** |
| **专 业** | **Computer Science and Technology** |
| **学生姓名** | **SARA SAJID UNDRE 映月** |
| **学生学号** | **201969990298** |
| **指导教师** | **WING W. Y. NG 吴永贤** |
| **提交日期** | **2023年4月20日** |

# ABSTRACT

Thyroid disorders are common endocrine disorders that have a significant impact on individuals' health and well-being, as well as on society as a whole. They can lead to increased healthcare costs, lost productivity, decreased quality of life, and mental health issues such as depression and anxiety.

This thesis presents a study on the use of machine learning algorithms, neural networks, and data mining techniques to classify thyroid disorders into three primary categories: hyperthyroidism, hypothyroidism, and euthyroidism. Medical datasets sourced from the UCI database repository provided by the Garavan Institute were utilized for the experiments. The methodology involves dataset acquisition, preprocessing, feature engineering, model development and evaluation. Various machine learning algorithms and neural network frameworks were employed for the classification process, and models were developed, trained, and refined using the GridSearch by finding the best hyperparameters for the machine learning models and the optimal epoch number for the neural network.

The developed models were evaluated using the testing dataset, and the best model was selected based on evaluation metrics. The selected model demonstrated an average accuracy rate exceeding 90%, validating its efficiency in accurately categorizing thyroid-related conditions. The study provides empirical evidence showcasing the high effectiveness of the proposed model in the classification of thyroid disorders.

The study offers significant insights into enhancing the precision and efficiency of thyroid disorder diagnosis through advanced technological techniques. It contributes to the understanding of underlying pathophysiological mechanisms and risk factors associated with diverse thyroid disorders. The outcomes hold promising implications for the development of innovative treatment modalities and the advancement of scientific knowledge of the disease.

Key words: Thyroid disorder, Data Mining, Machine Learning algorithms, Neural network, Classification.

# Table of Content

[ABSTRACT i](#_Toc133249216)

[Table of Content iii](#_Toc133249217)

[List of Tables v](#_Toc133249218)

[List of Figures vi](#_Toc133249219)

[1 Introduction 1](#_Toc133249220)

[1.1 Background 1](#_Toc133249221)

[1.2 Objectives 3](#_Toc133249222)

[1.3 Scope and Constraints 3](#_Toc133249223)

[2 Literature Review 6](#_Toc133249224)

[2.1 Thyroid disorder overview 6](#_Toc133249225)

[2.2 Current diagnosis methods 8](#_Toc133249226)

[2.3 Machine learning and deep learning in medical diagnosis 10](#_Toc133249227)

[2.4 Previous studies on thyroid disorder classification 11](#_Toc133249228)

[3 Methodology 13](#_Toc133249229)

[3.1 Data acquisition and pre-processing 14](#_Toc133249230)

[3.1.1 Data cleaning 15](#_Toc133249231)

[3.1.2 Data Integration 17](#_Toc133249232)

[3.1.3 Dimension reduction 17](#_Toc133249233)

[3.2 Feature Engineering 18](#_Toc133249234)

[3.2.1 Transformation of skew data 18](#_Toc133249235)

[3.2.2 Encoding 20](#_Toc133249236)

[3.2.3 Multi Collinearity 21](#_Toc133249237)

[3.3 Model Development 25](#_Toc133249238)

[3.3.1 Preparing Dataset for Model 25](#_Toc133249239)

[3.3.2 Traditional Machine Learning Classifiers 27](#_Toc133249240)

[3.3.3 Neural Network Framework 40](#_Toc133249241)

[3.4 Model Evaluation 47](#_Toc133249242)

[4 Experiment results and analysis 50](#_Toc133249243)

[4.1 Experiment results for models trained on Smote sampled dataset 51](#_Toc133249244)

[4.2 Experiment results for models trained on Kmeans Smote sampled dataset 54](#_Toc133249245)

[4.3 Experiment results for neural network models 55](#_Toc133249246)

[4.4 Analysis 56](#_Toc133249247)

[5 Conclusion 58](#_Toc133249248)

[6 References 59](#_Toc133249249)

# 

# List of Tables

[Table 3.1.1: Attributes present in the UCI Thyroid Dataset 14](#_Toc133245625)

[Table 3.1.1.1: Cumulative count of missing values in the dataset's column attributes 16](#_Toc133245626)

[Table 3.2.1.1: Selecting the Best Transformation Method for Reducing Skewness 20](#_Toc133245627)

[Table 3.2.3.1: VIF values for the attributes in the dataset 24](#_Toc133245628)

[Table 4.1.1: Performance Metric Comparison of Standard and Fine-tuned Algorithmic Models trained using Smote sampled dataset. 52](#_Toc133245629)

[Table 4.1.2: Performance Metric Comparison of Standard and Fine-tuned Algorithmic Models trained using K-means Smote sampled dataset. 55](#_Toc133245630)

[Table 4.1.3: Comparison of neural network framework model performance 56](#_Toc133245631)

# List of Figures

[Figure 3.1: Methodology flowchart 13](#_Toc133249625)

[Figure 3.1.1.1: Visualization of nullity by column before and after handling the data 16](#_Toc133249626)

[Figure 3.2.1.1: Comparison of Skewness for column attributes 19](#_Toc133249627)

[Figure 3.2.3.1: Dendrogram for the attributes in the dataset 22](#_Toc133249628)

[Figure 3.2.3.2: Initial correlation of attributes in the dataset 23](#_Toc133249629)

[Figure 3.2.3.3: Correlation plot after handling multicollinearity in the dataset 24](#_Toc133249630)

[Figure 3.3.1.1: Imbalanced dataset 26](#_Toc133249631)

[Figure 3.3.1.2: Balanced dataset 26](#_Toc133249632)

# Introduction

## Background

Thyroid disorders are a global health concern affecting millions of people worldwide. The thyroid gland, located in the neck, produces hormones that control the body's metabolism, heart rate and other essential functions. When the thyroid gland is not functioning properly, it can result in a range of disorders including hypothyroidism, hyperthyroidism, goiter (enlargement of the thyroid gland), and thyroid cancer. These disorders can have a significant effect on a person's health and quality of life, as well as economic and societal consequences. Early detection and appropriate treatment of thyroid disorders is crucial to manage the condition and improve outcomes for those affected.

According to the World Health Organization (WHO), an estimated 200 million people worldwide have some form of thyroid disorder. Women are more likely to be affected than men, and the prevalence increases with age. Certain populations may be at higher risk for thyroid disorders, such as individuals with a family history of thyroid disorders, who may develop the condition due to genetic predisposition. Studies have shown that individuals with a first-degree relative with a thyroid disorder are more likely to develop the condition themselves. Additionally, radiation exposure can increase the risk of developing a thyroid disorder. This is because radiation can damage the thyroid gland, leading to dysfunction. Research [­20][28] has shown that individuals who have received radiation therapy for cancer, or who have been exposed to high levels of radiation, such as from nuclear accidents, have a higher risk of developing a thyroid disorder.

Autoimmune diseases also increase the risk of developing a thyroid disorder. Autoimmune diseases are a group of disorders in which the immune system attacks the body's own tissues. Two autoimmune diseases that affect the thyroid are Hashimoto's thyroiditis and Graves disease. Hashimoto's thyroiditis is an autoimmune disorder in which the immune system attacks the thyroid gland, leading to inflammation and damage. This can result in hypothyroidism, which is a condition in which the thyroid gland does not produce enough hormones. Graves disease is another autoimmune disorder that affects the thyroid. It is characterized by hyperthyroidism, which is a condition in which the thyroid gland produces too much hormone. Both Hashimoto's thyroiditis and Graves' disease can lead to serious complications if left untreated.

Symptoms of thyroid disorders can vary and may include fatigue, weight fluctuation, changes in skin, hair, and in mood or mental function. Thyroid disorders can be diagnosed through blood tests and imaging tests, and treatment options include medications, surgery, and radiation therapy. People with thyroid disorders may also need to make lifestyle changes, such as adjusting their diet and exercise routine, in order to manage their condition.

In the United States, an estimated 20 million people have a thyroid disorder. The American Thyroid Association estimates that up to 60% of those with thyroid disorders are unaware of their condition[3]. In Europe, it is estimated that 1 in 20 people have a thyroid disorder. In Asia, the prevalence varies widely between countries, but it is estimated to affect at least 5% of the population in some countries. In Africa, there is limited data on the prevalence of thyroid disorders, but it is believed that the condition is often not properly diagnosed or treated. This could be attributed to a lack of access to healthcare and diagnostic facilities, as well as a lack of awareness about the condition among the population. This highlights the need for improved access to healthcare and diagnostic resources in developing countries, as well as increased awareness about the importance of early detection and management of thyroid disorders.

It is essential to consider that the true prevalence of thyroid disorders may be underestimated globally, particularly in low and middle-income countries where access to diagnosis and treatment is limited. Therefore, the actual number of people affected by thyroid disorders is likely to be higher than reported. Untreated thyroid disorders can lead to a variety of long-term health consequences, including an increased risk of heart disease, osteoporosis, and infertility. Studies have shown that individuals with untreated hypothyroidism (an underactive thyroid) have a higher risk of cardiovascular disease, while hyperthyroidism (an overactive thyroid) can lead to bone loss and an increased risk of osteoporosis. Additionally, untreated thyroid disorders can cause changes in hormone levels, which can lead to infertility and other reproductive issues.

Regular monitoring of thyroid function and prompt treatment of any issues that arise are necessary to prevent these potential long-term health consequences. To address this global health issue, there is a need for more research, improved access to diagnosis and treatment, and raising awareness among individuals about the importance of early detection and management of thyroid disorders.

## Objectives

The objective of this thesis is to explore the application of machine learning and deep learning techniques in the classification of thyroid disorders. This project aims to develop and evaluate practical models that can accurately distinguish between hypothyroidism, hyperthyroidism, and euthyroidism using real-world clinical data. Additionally, this thesis will investigate the effectiveness of feature engineering and model parameterization in improving the model’s performance, as well as the computational resources and time required for training and testing the models. The main objectives of this project are as follows:

1. Develop and evaluate ML and neural network models that accurately classify thyroid disorders using real-world clinical data.
2. Investigate the effectiveness of feature engineering and model parameterization in improving the model’s performance.
3. Analyze the computational resources and time required for training and testing the models to identify practical limitations and trade-offs of deploying the models in clinical settings.
4. Compare the performance of the ML and DL models and identify the potential benefits and challenges of incorporating the models into existing workflows.

## Scope and Constraints

The scope of this study is to evaluate and compare various machine learning and deep learning techniques for accurately classifying thyroid disorders. The study will focus on support vector machines, random forests, decision trees, naive bayes, logistic regression, linear discriminant analysis, k-nearest neighbors, multilayer perceptron, and hybrid neural network, with an emphasis on exploring their strengths, limitations, and best parameters & framework for the developed models. The dataset used in this study will be obtained from the UCI Machine Learning Repository, specifically the thyroid disease records provided by the Garavan Institute and J. Ross Quinlan. The scope of the study is outlined below:

1. Limit the classification of thyroid disorders to hypothyroidism, hyperthyroidism, and euthyroidism.
2. To not explore other medical conditions or diseases outside this scope.
3. Use the thyroid disease records provided by the Garavan Institute and J. Ross Quinlan to train and test the models.
4. Evaluate the effectiveness of various machine learning and deep learning techniques for accurately classifying thyroid disorders.
5. Focus on the strengths and limitations of machine learning and neural network framework models developed.

Despite the promising scope of the research, there are certain constraints that must be considered. The primary limitation is the reliance on clinical datasets that may not be comprehensive or of optimal quality, which could potentially limit the accuracy of the classification results. Furthermore, the collection and training procedures presented challenges, including resource constraints such as a moderately small and imbalanced dataset, which necessitated the acquisition of more relevant data.

Additionally, computational constraints arose due to the limitations of the computational resources available for training the models on a personal laptop, which included an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz and an NVIDIA Version 462.30 GeForce GTX 1650 GPU with 8.00 GB of RAM. Notably, the GeForce GTX 1650 GPU on this setup has only 4GB of video memory and 896 CUDA cores, which may not be sufficient for training larger models. Moreover, the training duration for a model often exceeded a day, necessitating the dedication of the computer to one task, resulting in significant time consumption, particularly considering the number of models that needed to be trained and fine-tuned.

To address these limitations, the utilization of Google Colab, a cloud-based platform, was decided upon. This approach significantly enhanced the efficiency of the training process by allowing training on an Intel(R) Xeon(R) CPU @ 2.20GHz and an NVIDIA Tesla T4 GPU with 12 GB of RAM. Notably, the Tesla T4 GPU on Google Colab has 16GB of dedicated video memory and 320 Turing Tensor Cores, is specifically designed for matrix operations involved in deep learning, making it ideal for accelerating the training of neural networks. Additionally, Google Colab offers the advantage of being able to use scalable cloud computing resources, allowing for significantly faster training times and the ability to work with larger datasets.

In terms of CPU specifications, the Intel(R) Xeon(R) CPU @ 2.20GHz on Google Colab is designed for high-performance computing and has two cores, which can handle multiple threads simultaneously, making it advantageous for training machine learning and deep learning models that often involve parallel processing. Furthermore, SMOTE techniques were employed to address the issue of imbalance in the dataset, which aimed to improve the accuracy and generalizability of the models developed in this study. Taken together, these steps were crucial in overcoming the constraints associated with the project and enabled the successful completion of the research.

# Literature Review

## Thyroid disorder overview

The thyroid gland, situated bilaterally in the neck and possessing a butterfly-shaped morphology, is an endocrine organ located inferiorly to the laryngeal prominence, commonly known as Adam's apple. The gland comprises two lobes, symmetrically located on either side of the trachea, connected by a slender tissue band, referred to as the isthmus. The thyroid gland produces two important hormones, namely triiodothyronine (T3) and thyroxine (T4), which are synthesized by combining iodine with the amino acid tyrosine. T3 comprises three iodine atoms and one tyrosine molecule, while T4 is composed of four iodine atoms and one tyrosine molecule. These hormones play a vital role in growth and development, metabolism, and the regulation of body temperature.

The regulation of thyroid hormone synthesis and secretion involves a complex feedback loop system, with the hypothalamus and pituitary gland playing key roles in controlling the process. The hypothalamus releases thyrotropin-releasing hormone (TRH), which stimulates the pituitary gland to secrete thyroid-stimulating hormone (TSH). TSH, in turn, stimulates the thyroid gland to synthesize and release T3 and T4 hormones.

Thyroid-binding globulin (TBG), a protein synthesized by the liver, is responsible for transporting T4 and T3 hormones in the bloodstream. TBG binds to T4 and T3, produced by the thyroid gland, and carries them to target tissues. The unbound fraction of thyroid hormones, known as "free" T3 or T4, is the active form of the hormone that can be utilized by the body's cells. Measuring free T3 and T4 levels in the blood provides a more accurate reflection of thyroid function than measuring total T3 and T4 levels, which include both bound and unbound fractions.

The hypothalamus and pituitary gland continuously monitor the levels of T3 and T4 hormones in the bloodstream. High levels of T3 and T4 hormones prompt the hypothalamus to decrease the secretion of TRH, which results in a reduction of TSH release from the pituitary gland, ultimately leading to a decrease in thyroid hormone production and release. Conversely, low levels of T3 and T4 hormones stimulate the hypothalamus to increase the release of TRH, prompting the pituitary gland to secrete more TSH, which leads to an increase in thyroid hormone synthesis and release.

The thyroid gland produces not only T3 and T4 hormones but also calcitonin, a hormone that plays a significant role in regulating calcium levels in the bloodstream. The hormone acts by inhibiting bone resorption, the process by which osteoclasts break down and remove bone, and by stimulating the activity of osteoblasts, the cells responsible for building new bone. This dual action of calcitonin ensures a balance of calcium in the blood and promotes healthy bones. The release of calcitonin is triggered by high levels of calcium in the bloodstream, and a feedback loop system regulates its release as calcium levels decrease, thus keeping the levels of calcium in the blood within a narrow range and ensuring proper functioning of the bones and other calcium-dependent body systems.

The thyroid gland is enclosed by a connective tissue capsule that houses a rich network of blood vessels, lymphatic vessels, and nerves. The superior and inferior thyroid arteries supply oxygen and nutrients to the gland, and the superior and inferior thyroid veins provide venous drainage. This extensive blood supply is critical for the proper functioning of the gland and hormone production.

Lymphatic vessels are equally important in the proper functioning of the thyroid gland. They help to eliminate waste products and cellular debris from the gland's hormone-producing cells. They also contribute to the gland's immune defence by removing any pathogens that may enter it.

The thyroid gland is innervated by nerves such as the recurrent laryngeal nerve and the external branch of the superior laryngeal nerve. These nerves provide sensory input to the gland, relaying information about the environment and the body's needs. The recurrent laryngeal nerve, in particular, plays a significant role in regulating the gland's hormone production. It can stimulate hormone synthesis by increasing blood flow to the gland or inhibit hormone production by reducing blood flow. The external branch of the superior laryngeal nerve also contributes to the regulation of hormone production by providing information about the body's metabolism and energy needs. The interplay between the blood vessels, lymphatic vessels, and nerves ensures the thyroid gland receives proper nourishment and functions optimally.

Thyroid disorders represent a common condition that can arise from a variety of underlying causes. One such disorder is hypothyroidism, a state in which the thyroid gland fails to produce an adequate amount of hormones. Autoimmune conditions, such as Hashimoto's thyroiditis, and surgical removal of the gland can precipitate this disorder. Clinical manifestations of hypothyroidism include fatigue, weight gain, cold intolerance, dry skin, constipation, and depression. Hyperthyroidism, conversely, manifests as an overproduction of thyroid hormones. Graves' disease and benign tumors on the gland are common etiologies for this condition. Symptoms of hyperthyroidism include weight loss, nervousness, tremors, sweating, heat intolerance, and irregular menstrual cycles.

Goiter, characterized by enlargement of the thyroid gland, is a clinical manifestation of various disorders such as iodine deficiency, benign tumors like thyroid nodules, or thyroid disorders. Both hypothyroidism and hyperthyroidism can induce goiter formation. In developing countries where access to iodine-rich foods and supplements may be limited, iodine deficiency can commonly cause goiter. In response to iodine deficiency, the thyroid gland increases hormone production in an attempt to maintain normal function, resulting in thyroid enlargement. Regular monitoring of iodine levels and appropriate supplementation are vital in preventing goiter from iodine deficiency.

Thyroid cancer, although rare, is a serious condition that arises from the uncontrolled proliferation of abnormal cells within the thyroid gland, culminating in the formation of a tumor or lump. Common clinical features of thyroid cancer include a lump or swelling in the neck, difficulty swallowing, and hoarseness. These symptoms can be attributable to compression of the windpipe or other neck structures by a growth on the gland. Other manifestations can include pain in the neck or throat, difficulty breathing or speaking, and alterations in voice. Advanced cases of the disease can spread to other parts of the body, such as the lymph nodes or lungs.

Euthyroidism represents the normal state of the thyroid gland in which it produces an appropriate amount of hormones, and is characterized by normal levels of TSH, T3, and T4 hormones in the blood. In contrast to hypothyroidism and hyperthyroidism, individuals with euthyroidism do not experience symptoms related to thyroid dysfunction. It is noteworthy, however, that even within the normal range of thyroid hormones, variations can occur that affect an individual's health and well-being. Thus, routine check-ups and monitoring of thyroid function are crucial for maintaining optimal health.

## Current diagnosis methods

The accurate diagnosis of thyroid disorders is imperative for the effective management and treatment of this widespread condition that affects millions of individuals worldwide. The pathophysiology of thyroid disease involves an imbalance in thyroid hormone production by the thyroid gland. To achieve an accurate diagnosis, a range of diagnostic methods are commonly utilized. The current diagnostic methods employed for the detection of thyroid disorders include the following:

1. Physical examination: The initial step in diagnosing a thyroid disorder is a comprehensive physical examination. During this examination, physicians assess for signs of a swollen thyroid gland, changes in skin, hair or nails, and any evidence of hoarseness or difficulty swallowing.
2. Blood tests: One of the most common diagnostic tests used to identify thyroid disease is a blood test. The blood test assesses the levels of thyroid hormones, including T3 (triiodothyronine), T4 (thyroxine), and TSH (thyroid-stimulating hormone). Elevated levels of TSH and decreased levels of T3 and T4 can indicate hypothyroidism, while increased levels of T3 and T4 and reduced levels of TSH can indicate hyperthyroidism.
3. Ultrasound: An ultrasound is a non-invasive imaging test that employs high-frequency sound waves to produce images of the thyroid gland. This test can assist physicians in determining the size, shape, and location of any nodules or masses on the thyroid gland.
4. Fine needle aspiration biopsy: A fine needle aspiration biopsy (FNAB) is a diagnostic procedure in which a thin needle is used to extract a small sample of tissue from the thyroid gland. The sample is subsequently examined under a microscope to identify whether the nodule is benign or cancerous.
5. Radioactive iodine uptake test: This test quantifies the amount of radioactive iodine absorbed by the thyroid gland. The test is useful in determining the function of the thyroid gland and detecting any areas of abnormal activity.
6. Thyroid scan: A thyroid scan is a non-invasive imaging test that uses a small amount of radioactive material to generate images of the thyroid gland. This test can assist physicians in determining the size, shape, and location of any nodules or masses on the thyroid gland.
7. Thyroid antibodies test: This test measures the levels of antibodies in the blood that attack the thyroid gland. The presence of these antibodies can indicate autoimmune thyroid disorders such as Hashimoto's thyroiditis or Graves' disease.

The accurate diagnosis of thyroid disorders is paramount to their effective management and treatment, given the significant impact of this global condition on millions of individuals worldwide. Despite the widespread use of traditional diagnostic methods, several limitations may compromise their accuracy and reliability.

For instance, blood tests may produce inaccurate results due to factors such as stress, illness, and medications. Although ultrasound is useful in detecting nodules or masses, it may not always be reliable in determining their nature. Additionally, achieving an accurate diagnosis may require multiple tests, which can be time-consuming and financially demanding.

Biopsy procedures, such as fine needle aspiration biopsy, may carry a risk of false negatives, leading to misdiagnosis. Moreover, tests that involve the use of radioactive material, such as the radioactive iodine uptake test and thyroid scan, may expose patients to radiation, potentially increasing the risk of cancer.

Antibody tests may not always be reliable as the levels of antibodies can fluctuate and may not detect autoimmune disorders in their early stages. Furthermore, subclinical thyroid disorders, where hormone levels are abnormal but the patient is asymptomatic, can be challenging to diagnose using traditional diagnostic methods. It is therefore critical to recognize these limitations and strive for the development of more accurate and dependable diagnostic methods in the diagnosis of thyroid disorders.

## Machine learning and deep learning in medical diagnosis

The conventional diagnosis of thyroid disorders is currently facing limitations in accuracy and dependability. Therefore, there is a need to develop more accurate and reliable diagnostic methods for the detection of thyroid disorders. Such that it can detect the condition in its early stages and provide a reliable diagnosis without the need for multiple tests. Recent advancements in medical technology, including the use of molecular diagnostic tests, imaging techniques, and artificial intelligence (AI), have the potential to revolutionize the diagnosis of thyroid disorders. Some applications of AI, ML, and DL in Medical Diagnosis are as listed below:

1. Analysis of Medical Images: Recent advancements in deep learning algorithms have made it possible to analyse medical images, including X-rays, MRI scans, and CT scans, with increased accuracy and reduced manual analysis time. This approach provides valuable information for diagnosis and treatment planning, resulting in improved treatment outcomes.
2. Predictive Maintenance of Medical Equipment: ML algorithms can be utilized to monitor medical equipment for anomalies and predict when maintenance is necessary, thus avoiding equipment breakdowns and reducing maintenance costs.
3. Personalized Medicine and Drug Discovery: By analysing patient data, personalized treatments can be identified, leading to improved patient outcomes. In drug discovery, ML algorithms can be used to analyse large amounts of clinical data to predict the efficacy and toxicity of new drugs.
4. Analysis of Electronic Health Records (EHRs): ML and DL algorithms can analyse the vast amounts of patient data stored in EHRs to identify patterns and trends that can inform better diagnoses and treatments. In predictive modelling of patient outcomes, such as hospital readmissions, disease progression, and treatment response, ML and DL have also shown great potential.
5. Wearable Devices for Early Disease Detection and Predictive Risk Assessment: The use of wearable devices for early disease detection and predictive risk assessment for hospital readmissions has been applied using ML algorithms, with the goal of reducing the cost of hospitalization through early intervention.
6. Clinical Documentation and Coding: Natural language processing (NLP) algorithms have been employed to extract information from clinical documents for analysis and reporting. ML-based decision support systems have also been developed to provide healthcare professionals with informed decision-making support, further increasing the accuracy of diagnoses in the medical field.

## Previous studies on thyroid disorder classification

Thyroid disorder classification has been the focus of several studies in recent years, with the aim of improving the accuracy and efficiency of diagnosis through automated classification systems. Machine learning and deep learning techniques have been utilized in these studies to achieve this goal.

Salman and Sonuç [21] conducted research utilizing machine learning to classify thyroid diseases using two models. In the first model, all 16 input characteristics were incorporated, and the random forest algorithm achieved the highest accuracy of 98.93%. In the second model, three attributes were excluded based on a prior investigation, resulting in improved accuracy for some algorithms, including 90.67% for Naive Bayes and 96.4% for MLP. This study highlights the potential of machine learning algorithms for the precise classification of thyroid diseases.

Tahir Alyas and Muhammad Hamid [1] also conducted a study on the classification of thyroid diseases using machine learning algorithms. Their research aimed to enhance the timely detection and treatment of thyroid diseases through automated and accurate thyroid nodule detection in ultrasound images. Various algorithms were evaluated, including decision tree, random forest, KNN, and artificial neural networks, on both manipulated and unmanipulated datasets. The random forest algorithm achieved the highest accuracy of 94.8% and specificity of 91%. The study concluded that random forest and KNN demonstrated the best outcomes for the classification of thyroid diseases.

Chaganti, Rustam, and De La Torre Díez [4] performed a study to enhance the accuracy of thyroid disease prediction through feature engineering in machine learning and deep learning models. Their proposed method utilized forward and backward feature selection, bidirectional feature elimination, and machine learning-based feature selection using extra tree classifiers. The extra tree classifier-based selected feature produced the best results, attaining 0.99 accuracy and an F1 score when combined with the random forest classifier. The study suggests that machine learning models are superior for thyroid disease detection in terms of accuracy and computational complexity. The proposed approach was validated using k-fold cross-validation and performance comparison with existing studies, demonstrating high accuracy in predicting ten thyroid diseases.

Afzal Hussain Shahid and Maheshwari Prasad Singh [22] conducted research comparing the effectiveness of three classifiers (Random Forest, Support Vector Machine, and K-Nearest Neighbours) in diagnosing thyroid disease using a dataset from the UCI machine learning repository. The study revealed that the Random Forest classifier outperformed the SVM and K-NN classifiers, with an overall classification accuracy of 98.50% compared to 97.02% and 95.81%, respectively. These outcomes indicate the potential of Random Forest as an effective tool for thyroid disease diagnosis utilizing machine learning techniques.

# Methodology

The proposed framework for the classification of thyroid disorders represents a substantial contribution to the field of medical diagnostics. This framework is designed to facilitate the identification and categorization of thyroid disorders by employing a comprehensive approach that integrates diverse clinical parameters, such as hormone levels, symptoms, and medical history. Additionally, state-of-the-art machine learning classifiers [8] and neural network architecture [6] [9] are incorporated within the framework to optimize the accuracy and efficiency of the classification process. The proposed framework for thyroid disorder classification consists of four key steps, beginning with data acquisition and pre-processing, followed by feature engineering, model development, and model evaluation. The design flowchart of the framework is illustrated in Figure 3.1. This systematic approach provides a structured framework for the accurate and efficient classification of thyroid disorders, which has significant potential to improve the diagnosis and treatment of these disorders.

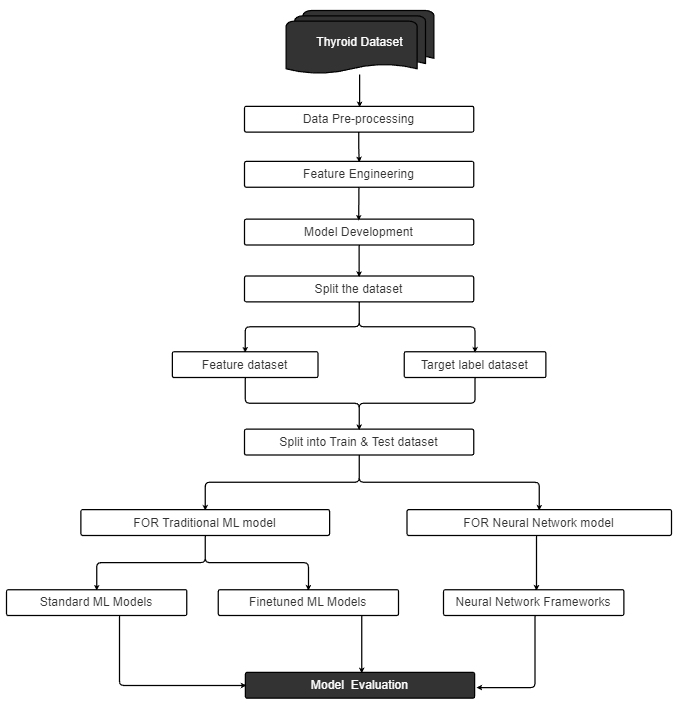


Figure 3.1: Methodology flowchart

## Data acquisition and pre-processing

The present study utilized the Thyroid dataset obtained from the UCI Machine Learning Repository [26], a reputable and widely used database for machine learning research. The dataset was collected and provided by the Garavan Institute and J. Ross Quinlan, covering several years of data collection beginning in January 1987. The study encompasses a sample of 2800 training data instances, featuring a varied population of patients with distinct ages, medical histories, and thyroid conditions. The data was procured from the medical records of patients, resulting in a dataset comprising 30 attributes, including demographic information, clinical measurements, and other significant data points, as outlined in Table 3.1.1.

Table 3.1.1: Attributes present in the UCI Thyroid Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No** | **Column Attribute** | **Data type** | **Unique values** | **Description** |
| 1 | age | Numeric | Continuous | Age of the patient |
| 2 | sex | Categorical | “M”, “F”, “nan” | Gender of the patient |
| 3 | on thyroxine | Categorical | f, t | Indicates if the patient is prescribed thyroxine medication |
| 4 | query on thyroxine | Categorical | f, t | Indicates a query has been actioned on thyroxine activity |
| 5 | on antithyroid medication | Categorical | f, t | Indicates if the patient is prescribed anti-thyroxine medication |
| 6 | sick | Categorical | f, t | Indicates sickness due to thyroxine depletion or overactivity |
| 7 | pregnant | Categorical | f, t | Indicates whether the patient is pregnant |
| 8 | thyroid surgery | Categorical | f, t | Indicates whether the patient has had thyroid surgery |
| 9 | I131 treatment | Categorical | f, t | Indicates whether the patient has had radioactive iodine treatment |
| 10 | query hypothyroid | Categorical | f, t | Indicates underactive thyroid query |
| 11 | query hyperthyroid | Categorical | f, t | Indicate overactive thyroid query |
| 12 | lithium | Categorical | f, t | Indicates of Lithium carbonate is administered to decrease the level of thyroid hormones |
| 13 | goitre | Categorical | f, t | Indicates swelling of the thyroid gland |
| 14 | tumor | Categorical | f, t | Indicates tumor |
| 15 | hypopituitary | Categorical | f, t | Indicates a diagnosed underactive thyroid |
| 16 | psych | Categorical | f, t | Indicates whether the patient has a psychological condition |
| 17 | TSH measured | Categorical | f, t | Indicates whether the patient was administrated with TSH (Thyroid stimulating hormone) blood test. |
| 18 | TSH | Numeric | f, t | The reading results of the TSH blood test |
| 19 | T3 measured | Categorical | f, t | Indicates whether the patient was administrated with T3 (Triiodothyronine) blood test. |
| 20 | T3 | Numeric | Continuous | The reading results of the T3 blood test. |
| 21 | TT4 measured | Categorical | f, t | Indicates whether the patient was administrated with TT4 (Total Thyroxine) blood test. |
| 22 | TT4 | Numeric | Continuous | The reading results of the TT4 blood test |
| 23 | T4U measured | Categorical | f, t | Indicates whether the patient was administrated with T4U (Thyroxine utilization rate) blood test. |
| 24 | T4U | Numeric | Continuous | The reading results of the T4U blood test |
| 25 | FTI measured | Categorical | f, t | Indicates whether the patient was administrated with FTI (Free Thyroxine Index) blood test. |
| 26 | FTI | Numeric | Continuous | The reading results of the FTI blood test |
| 27 | TBG measured | Categorical | f, t | Indicates whether the patient was administrated with TBG (Thyroid-binding globulin) blood test. |
| 28 | TBG | Numeric | Continuous | The reading results of the TBG blood test |
| 29 | referral source | Categorical | WEST, STMW, SVHC, SVI, SVHD, and others. | Indicates the referral source of the patients |
| 30 | Labels | Categorical | ‘negative’, ‘hypothyroid’, ‘hyperthyroid’ (FORMAT - class.|id) | Indicates the thyroid disorder patient is diagnosed with. |

However, as with any real-world dataset, the Thyroid dataset is subject to limitations and errors that may affect the accuracy of the model and analysis results. These limitations may include missing data, inaccuracies in clinical parameter measurements, and potential data bias or skewness. The imposition of constraints can exert a notable influence on both the model and the analytical findings. In this study, the datasets pertaining to hyperthyroid and hypothyroid conditions were utilized, which were procured from the allhyper.data and allhypo.data files. Subsequent scrutiny of the datasets revealed the presence of inconsistencies, absent values, and superfluous attributes, which were dealt with through the implementation of methods such as cleaning, integration, dimension reduction, normalization, and encoding. These measures were aimed at ensuring the credibility and authenticity of the results generated from the analysis.

### Data cleaning

Within the hyperthyroid and hypothyroid datasets, a total of thirty attributes are present. Notably, the final attribute, labelled as "label," indicates the diagnosed condition and comprises a string of letters in the format "X.|Y," combining both class (X) and identifier (Y) information. This amalgamation results in a lack of clarity, rendering it difficult to interpret the content. To rectify this issue, the attribute was split into two separate columns, 'class' and 'ID,' utilizing the delimiter "|" present in the data. The initial segment was assigned to the 'class' column, while the numeric value following the delimiter was stored in the 'ID' column. Furthermore, the 'class' attribute was refined by replacing the period (".") with an empty space. The 'class' column was distinguished as 'hypo\_class' for the hypothyroid dataset and 'hyper\_class' for the hyperthyroid dataset.

Upon closer examination of the cleaned dataset, it was observed that the age variable exhibited a wide range of values, spanning from 1 to 455. However, a single data point was recorded with an age of 455, exceeding the standard human lifespan range of 70s to 100s. This anomalous record was deemed implausible and, therefore, was excluded from the dataset to avoid any consequential effects on the mean calculation.

The dataset encompasses a total of six numerical attributes, including age, TSH, T3, T4, T4U, and FTI, with the remaining attributes being categorical. These categorical attributes include sex, on thyroxine, query on thyroxine, on antithyroid medication, sick, pregnant, thyroid surgery, I131 treatment, lithium, goitre, tumor, hypopituitary, psych, and labels. Of note, the 'sex' attribute was the sole categorical attribute that contained missing values (see Figure 3.1.1.1).

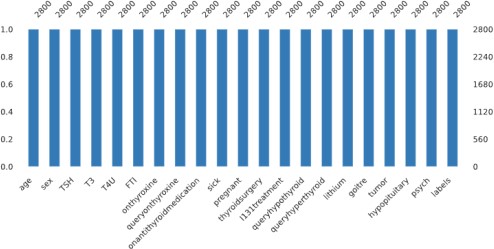
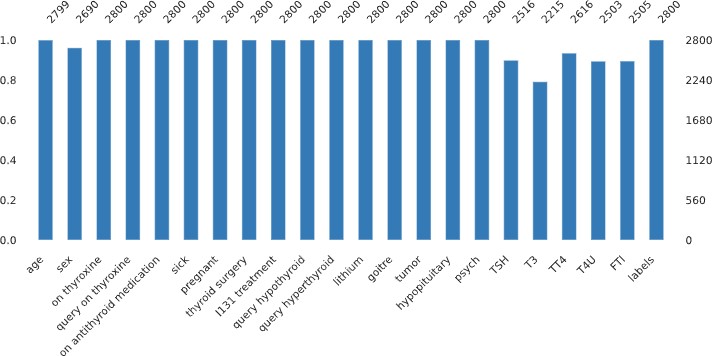


Figure 3.1.1.1: Visualization of nullity by column before and after handling the data

To address the presence of missing values, the dataset was segregated into two categories based on the type of data present: categorical or numerical. For the categorical attribute, the mode was utilized to substitute the null values. However, this method has the potential to reduce variance, as the imputed values are merely estimates and not inherently related to other variables, leading to decreased correlation between the imputed values and other variables.

Table 3.1.1.1: Cumulative count of missing values in the dataset's column attributes

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Column Attribute** | **Data Type** | **Sum of Null values** |
| 1 | Age | Numeric | 1 |
| 2 | Sex | Categorical | 110 |
| 3 | TSH | Numeric | 284 |
| 4 | T3 | Numeric | 585 |
| 5 | TT4 | Numeric | 184 |
| 6 | T4U | Numeric | 297 |
| 7 | FTI | Numeric | 295 |

The numerical features within the dataset also exhibit missing values, as illustrated in Table3.1.1.1. To mitigate this issue, two methods may be utilized. The first method involves substituting the missing values with the column's mean. Alternatively, in cases where an outlier is suspected, a different approach may be pursued, involving replacing the missing values with the column's median.

### Data Integration

The present study entailed the classification of thyroid disorders into three distinct categories: hypothyroidism, hyperthyroidism, and negative (euthyroidism). However, the datasets obtained from medical records contained several subclasses of primary thyroid disorders. In order to simplify the analysis, these subclasses were amalgamated into a single superclass. This was achieved by replacing "compensated hypothyroid," "primary hypothyroid," and "secondary hypothyroidism" with hypothyroidism, and "T3 toxic" and "goitre" with hyperthyroidism.

To streamline the dataset, the hypothyroid and hyperthyroid datasets were concatenated, since the sole difference between them was the "class" column. As a result, instead of the initial dataset's singular "label" column, there were now two "class" columns, namely hypo\_class and hyper\_class, and "ID" being the last three columns. Moreover, the hypo\_class and hyper\_class columns were merged to produce the "labels" column, which was conditioned in the following manner:

1. Negative + Negative = Negative
2. Hypothyroid + Negative = Hypothyroid
3. Hyperthyroid + Negative = Hyperthyroid

### Dimension reduction

The selection of appropriate attributes plays a crucial role in optimizing the performance of the model. To this end, several columns in the dataset were considered redundant and were eliminated, including the hypo\_class and hyper\_class columns, which were consolidated to form a new label class. Additionally, columns such as referral source, ID, TBG, and measure columns, which serve only to indicate the validity of specific values, were also excluded as they do not contribute to the process of disease classification.

As a result of this pruning process, the final dataset included only the following essential attributes: age, sex, on thyroxine, querying about thyroxine, on antithyroid medication, sick, pregnant, thyroid surgery, I131 treatment, query hypothyroidism, query hyperthyroidism, lithium, goitres, tumors, hypopituitary, psych, TSH, T3, TT4, T4U, FTI, and the labels class. These attributes were deemed sufficient for the analysis and classification of thyroid disorders.

## Feature Engineering

In the processed dataset, the presence of skewed data, categorical features, and highly correlated attributes presents a challenge for the development of effective models. To overcome these issues, the dataset undergoes a feature engineering process to simplify and prepare the features for use in modelling applications. The three primary methods employed in this process include transforming skewed data, encoding categorical features, and addressing multicollinearity. These methods are crucial for enhancing the performance and interpretability of the model.

### Transformation of skew data

In the realm of probability theory and statistics, skewness serves as a metric for measuring the asymmetry of a real-valued random variable's probability distribution with respect to its mean. Data is deemed highly skewed if its skewness value falls above 1 or below -1, while data is moderately skewed if it ranges between +0.5 and -0.5. A skewness value of zero indicates symmetrical data. The direction of skewness can be inferred by analyzing the distribution's tail, where data with a right-tail indicates positive skewness, while data with a left-tail implies negative skewness. Skewness quantifies the degree to which data deviates from a normal distribution.

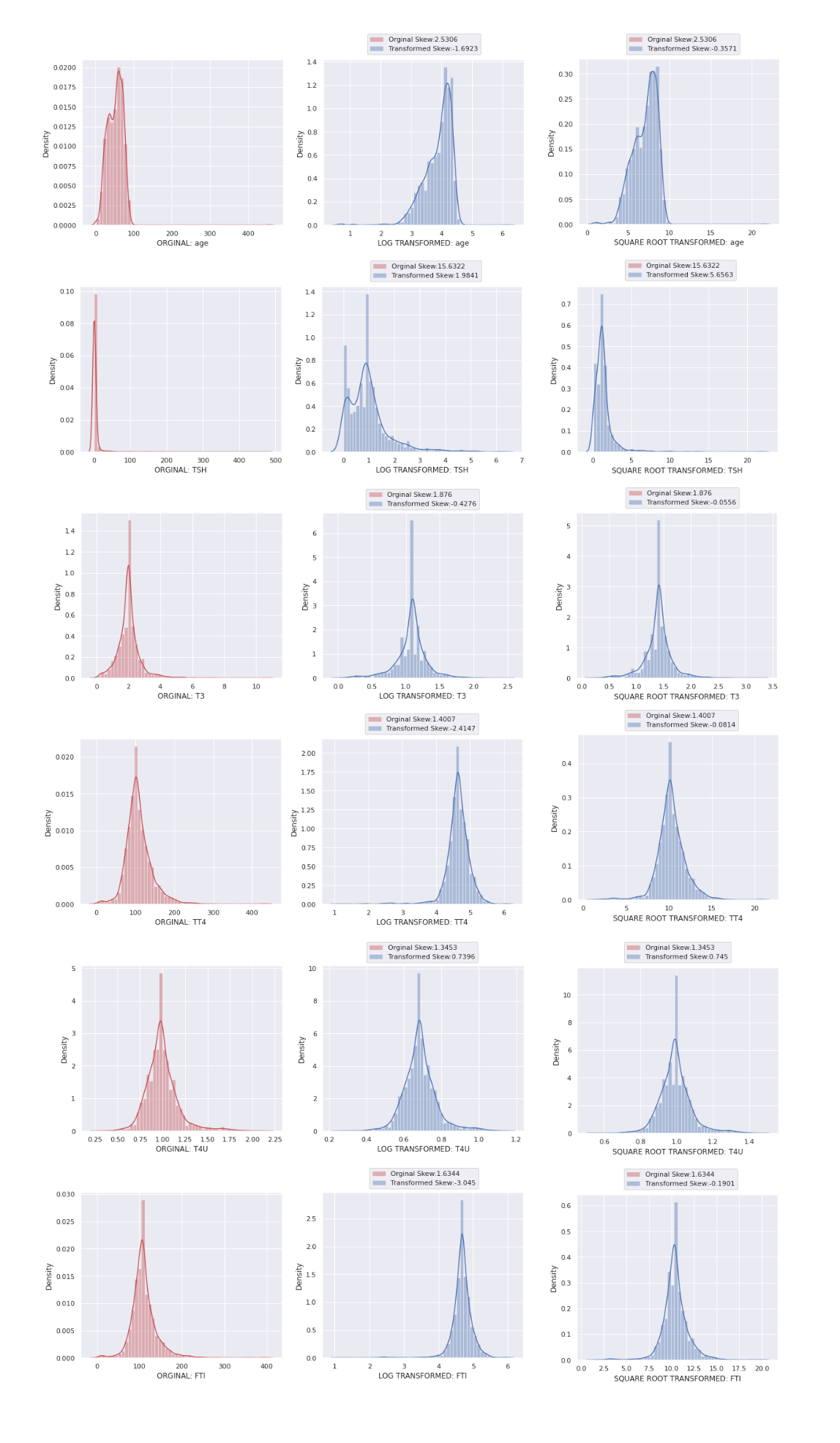


Figure 3.2.1.1: Comparison of Skewness for column attributes

To enhance prediction accuracy, log and square root transformations are employed on the numerical attribute columns of the dataset. The log transformation computes the natural logarithm of the data, expressed as log(x), where x is the original data. This transformation is used to transform positively skewed data into a more symmetrical distribution. The reduction in skewness is achieved by decreasing the influence of large values while increasing the impact of small values. The square transformation, on the other hand, involves squaring the data. The square transformation is defined as y = sqrt(x), where x denotes the original data. This transformation is applied to convert negatively skewed data into a more symmetrical distribution by decreasing the impact of small values and increasing the influence of large values (see Figure 3.2.1.1).

Table 3.2.1.1: Selecting the Best Transformation Method for Reducing Skewness

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Column Attribute** | **Original skew** | **Log Transform skew** | **Square root Transform skew** | **Best Transform method** |
| 1 | Age | 2**.**5306 | -1**.**6923 | -0**.**3571 | Square root |
| 2 | TSH | 15**.**6322 | 1**.**9841 | 5**.**6563 | Log |
| 3 | T3 | 1**.**876 | -0**.**4276 | -0**.**0556 | Square root |
| 4 | TT4 | 1**.**4007 | -2**.**4147 | -0**.**0814 | Square root |
| 5 | T4U | 1**.**3453 | 0**.**7396 | 0**.**745 | Log |
| 6 | FTI | 1**.**6344 | -3**.**045 | -0**.**1901 | Square root |

Both the log and square root transformations serve to stabilize the variance of the data. However, to determine the optimal transformation, the skewness of the original data is compared to the skewness of the transformed data. The transformation that yields the most substantial reduction in skewness is applied to the numerical attribute columns, creating separate columns for each transformed column. To achieve optimal results, only the best-transformed columns for each numerical attribute column are retained, while the remaining columns are discarded. (Refer to Table 3.2.1.1)

### Encoding

Encoding is a crucial step in feature engineering that facilitates the transformation of categorical features into numerical features. One popular encoding technique is one-hot encoding, which creates a binary variable for each category. However, the use of this approach may result in the dummy variable trap when two or more variables are highly correlated (multicollinear). In the present dataset, a significant number of attributes exhibit high correlation, which prompted me to extract categorical features initially. To avoid the dummy variable trap [14], it is recommended to have one less variable than the total number of categories in categorical data (n) . Therefore, I extracted two categorical features: one containing the diagnosis condition label column, which serves as the dependent feature, and the other containing the remaining features, which are independent and encoded using a dummy variable.

To encode the dependent feature, it was necessary to convert the labeling scheme from non-numerical to numerical. Label encoders were employed to map a word label to a specific numeric feature within the data. Subsequently, the data underwent a fitting and transforming process, during which the feature mean and distribution were computed during the fitting phase, and the feature was scaled based on these factors during the transformation phase. The label column comprises three distinct thyroid statuses: negative, hypothyroid, and hyperthyroid, which were respectively translated as 2, 1, and 0.

### Multi Collinearity

Multicollinearity refers to the existence of a high degree of interdependence and correlation among multiple independent variables within a dataset. This phenomenon is commonly manifested by perfect collinearity, which is characterized by a correlation coefficient of either 1.0 or -1.0 between two variables. The presence of multicollinearity within a dataset may lead to increased standard errors and a consequent reduction in the reliability of statistical inferences that can be drawn from the data.

In order to visually evaluate and assess the extent of multicollinearity within our original dataset, I employed an agglomerative hierarchical clustering dendrogram, which is a type of data visualization technique that can aid in understanding the interrelationships between variables within a dataset. This approach treats each variable as a distinct cluster initially and subsequently combines the two most similar clusters at each stage until all variables are in a single cluster. The result is a dendrogram that represents the interrelationships between variables.

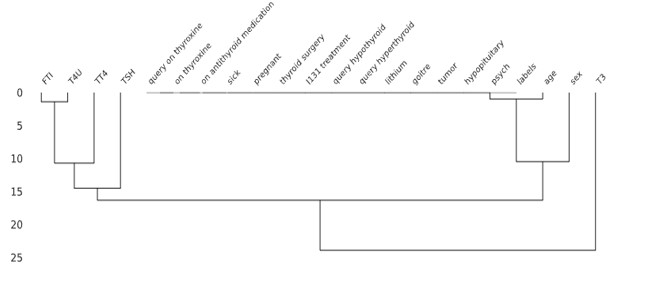


Figure 3.2.3.1: Dendrogram for the attributes in the dataset

The dendrogram analysis of the thyroid dataset revealed two major groups of variables with high correlations. The first group, comprising the FTI (Free Thyroxine Index) and T4U ( Thyroxine utilization rate) variables, formed Cluster A, indicating a strong association between the two variables. Cluster A was further combined with other hormone variables, clustering with TT4 and subsequently with TSH at some distance, resulting in the formation of a major cluster group (Cluster C) consisting of hormones.

The second group, formed by the psych and age variables, formed Cluster B with high correlation, which clustered with the attribute sex at a larger distance, forming Cluster D. Clusters C and D were found to have a slight relationship, forming another cluster (Cluster E), which was subsequently clustered with T3. Interestingly, the remaining attributes were found to be independent, indicating no significant relationship with the other variables in the dataset. These findings suggest that hormone-related variables play a more significant role in determining thyroid function than demographic factors, which may have less influence on thyroid function in the studied population.

In addition, a correlation heatmap was utilized to offer a comprehensive depiction of the underlying trends present in the data. To ascertain the strength of the correlations between features and determine the highly correlated features, a correlation matrix analysis was conducted to calculate the pairwise correlations between the features. Correlation matrix analysis is a statistical technique frequently employed to analyze the relationship between two or more variables. This method involves computing the correlation coefficient, which measures the strength and direction of the linear relationship between two variables. The correlation coefficient ranges from -1 to +1, with a value of -1 indicating a perfect negative correlation, a value of +1 indicating a perfect positive correlation, and a value of 0 indicating no correlation.

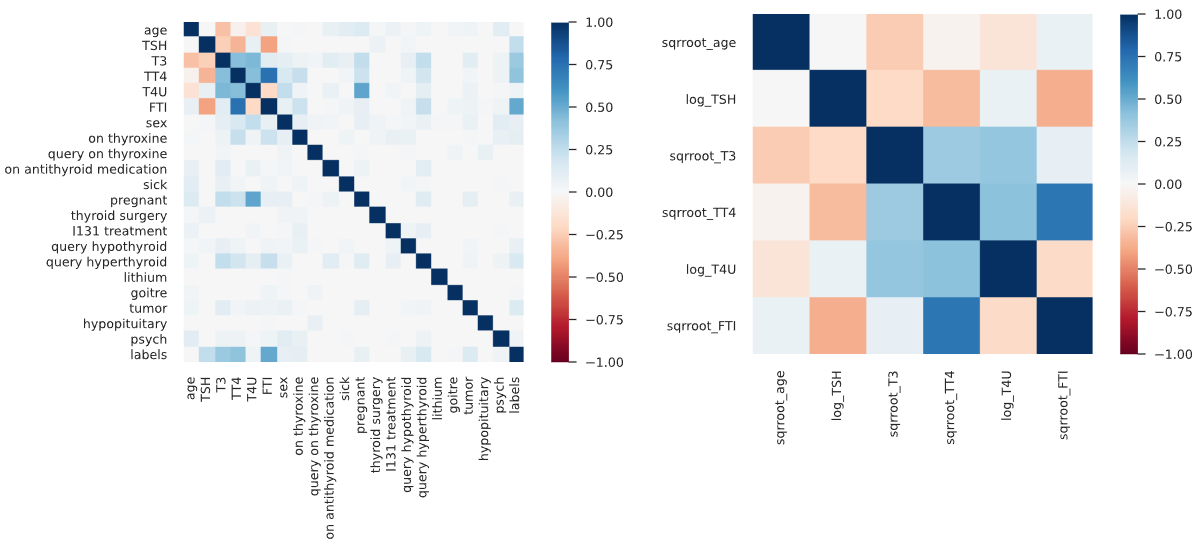


Figure 3.2.3.2: Initial correlation of attributes in the dataset

The resulting correlation matrix (see Figure 3.2.3.2) enabled the identification of variables with moderate positive or negative correlations. Additionally, the dendrogram visualization of the correlations was consistent with the matrix, reflecting the pairwise relationships between variables.

To further evaluate the impact of multicollinearity among independent features in a dataset, the Variance Inflation Factor (VIF) was employed, which is a widely accepted measure used to assess this phenomenon. The VIF quantifies the amount of increase in the variance of the estimate of a regression coefficient that occurs when all other predictors are included in the model. Features with high VIF values indicate a significant influence on multicollinearity, which can lead to unstable and unreliable results. A VIF of 1 indicates the absence of correlation, while a VIF between 1 and 5 suggests a moderate correlation. On the other hand, a VIF between 5 and 10 signifies a high degree of correlation.

Table 3.2.3.1: VIF values for the attributes in the dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Column Attribute** | **Initial VIF** | **Final VIF** |
| 1 | sqrt\_age | 1.07917 | 1.07816 |
| 2 | log\_TSH | 1.4201 | 1.39594 |
| 3 | sqrt\_T3 | 1.608135 | 1.607507 |
| 4 | sqrt\_TT4 | 17.571336 | - |
| 5 | log\_T4U | 6.915051 | 1.410238 |
| 6 | sqrt\_FTI | 14.999932 | 1.545203 |
| 7 | constant | 521.505157 | 206.999885 |

Upon analyzing the numerical features in the dataset using the VIF technique, the columns sqrt\_TT4 and sqrt\_FTI displayed coefficients of 17.571336 and 14.9999, respectively, indicating significant multicollinearity. To address this issue and minimize the potential impact of multicollinearity in the dataset, the variable sqrt\_TT4, which displayed the highest coefficient value, was removed. Following this step, the remaining columns demonstrated coefficients ranging from 1.0 to 1.5, indicating a lack of significant correlation between the variables.

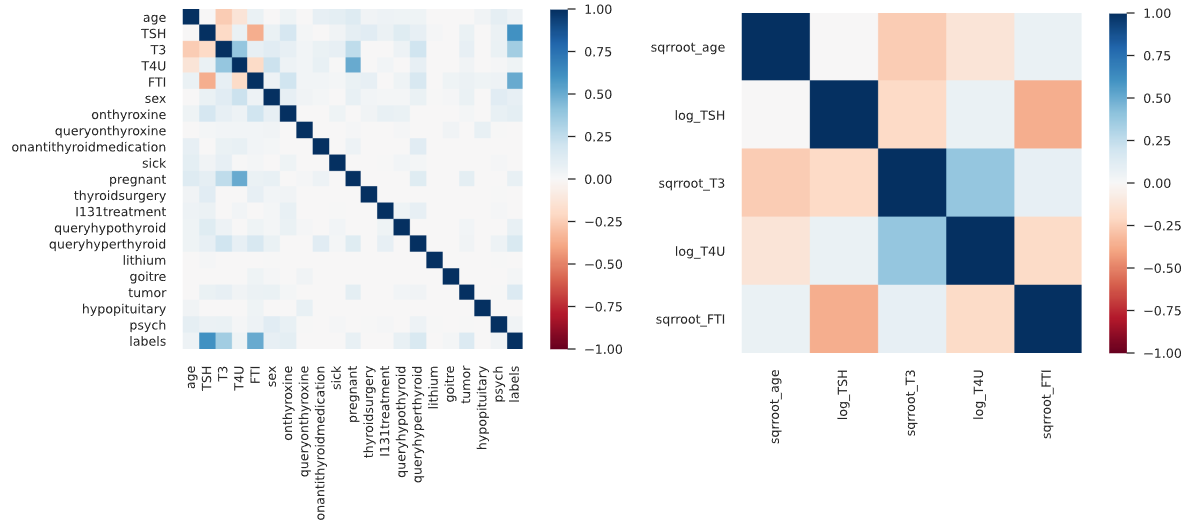


Figure 3.2.3.3: Correlation plot after handling multicollinearity in the dataset

By taking measures to reduce the presence of multicollinearity, I enhanced the validity and reliability of statistical inferences drawn from the model. It is worth noting that a constant column of ones is included in the dataset to incorporate an intercept term in the regression model, which provides stability and interpretability to the regression coefficients, despite the presence of multicollinearity.

## Model Development

This section details the development of machine learning models for the research study, which were developed using two main approaches: algorithm-based models and neural network framework-based models. The first subsection outlines the steps taken to prepare the dataset for both model types, which involved the train-test data split. Subsequently, an in-depth discussion of the machine learning approaches employed is presented, including the algorithm name, a brief overview of its functioning, and the standard and fine-tuned versions of the model. The subsequent subsection details the neural network framework approaches, describing the design of the framework and the techniques employed for training and evaluating the models. The primary aim of this section is to provide a comprehensive understanding of the development of the models, their accuracy, and their suitability for the research study.

### Preparing Dataset for Model

The study necessitated the use of two distinct types of datasets, one for the machine learning models and another for the neural network framework models. To prepare the dataset for modeling, the processed data was divided into two subsets to differentiate between independent and dependent features. This stratification allowed for the creation of separate dataframes for the dependent feature, represented by the label column, and the independent features, comprising all columns except for the label, named "Y" and "X," respectively. This approach was consistently implemented for the machine learning model dataset.

To facilitate the use of the dataset for the neural network model, the dataframes were transformed into NumPy arrays using the .values method. Furthermore, to enable classification, the target array (Y) was converted into a categorical variable. To evaluate the performance of the models, the dataset was further divided into training and testing sets, with 70% of the data allocated to the training set and 30% to the test set.



Figure 3.3.1.1: Imbalanced dataset

On examining the data distribution in the machine learning model dataset, it was determined that the dataset exhibited a significant degree of imbalance, as illustrated in Figure 3.3.1.1. This imbalanced distribution of the data highlights the necessity for additional pre-processing through data balancing techniques. To alleviate the impact of this issue, two data balancing methods were employed: the Synthetic Minority Over-Sampling Technique (SMOTE) and the K-means Synthetic Minority Over-Sampling Technique (Kmeans SMOTE). These techniques were implemented to address the class imbalance by equalizing the distribution of the label class records and were applied exclusively to the training datasets. Subsequently, the impact of these techniques on the data distribution was re-evaluated, as depicted in Figure 3.3.1.2.

Figure 3.3.1.2: Balanced dataset

SMOTE is a widely employed oversampling technique that creates synthetic samples of the minority class by interpolating between existing minority class samples. The procedure entails selecting two existing minority class samples, computing the difference between them, and multiplying this difference by a random number between 0 and 1. The result is subsequently added to the feature values of one of the selected samples to create a new synthetic sample.

In contrast, Kmeans SMOTE is an adapted version of the SMOTE technique that utilizes the K-means clustering algorithm. This technique first utilizes the K-means clustering algorithm to identify natural clusters within the minority class samples. Subsequently, synthetic samples are generated for each cluster using the SMOTE technique. This modification aims to ensure that the produced synthetic samples better represent the minority class distribution by considering the cluster structure of the data.

The current study further examines the role of pre-processing techniques in the training of deep neural networks on imbalanced datasets. While some argue that deep learning models can learn from imbalanced datasets without pre-processing through data balancing techniques, this assertion is dependent on the deep learning approaches employed and the characteristics of the dataset employed.

The pre-processed dataset utilized in this study retains moderate to weak collinearity between the column attributes. This feature, while sometimes viewed as problematic, can provide redundancy in the information captured by the features, thereby conveying similar information about the target variable. In the case of imbalanced datasets, this redundancy can be beneficial for deep learning models as it can aid the model in learning from the minority class samples more effectively.

Given the moderate multicollinearity between column data in the dataset, the model can still learn from the minority class samples indirectly through other features that are correlated with them. Thus, the model can identify relevant patterns and features required for accurate classification without the need for data balancing techniques. Therefore, the dataset for the deep learning model was not balanced and kept as such in this study.

### Traditional Machine Learning Classifiers

The precision and dependability of models are fundamental components of data analysis and prediction. Advancements in machine learning (ML) and deep learning (DL) techniques have greatly enhanced the effectiveness of models across various domains. To that end, this project employs several ML and DL algorithms to develop models capable of making precise predictions. This section details the specific ML and DL algorithms that were utilized to construct various models in this study.

1. Decision Tree Model

Decision Trees are tree-based models that use a recursive partitioning strategy to divide the feature space into smaller subspaces. The resulting tree structure represents a sequence of decisions and their corresponding outcomes. The algorithm iteratively splits the training data based on the features that result in the greatest reduction in impurity. Measures such as information gain, gain ratio, or Gini impurity are used to select the optimal feature for each split. One of the primary advantages of Decision Trees is their interpretability, which arises from the hierarchical nature of the tree structure. The branching points in the tree represent decision boundaries, and the leaf nodes represent class labels or numerical predictions. Therefore, the tree structure provides insight into the relationships between the features and the target variable.

Furthermore, they can be used for feature selection, as they provide feature importance that identify the most important features for prediction. To make predictions using a Decision Tree, the model traverses the tree from the root to a leaf node. At each node, the model checks the value of the corresponding feature and moves to the appropriate child node. The process continues until a leaf node is reached, and the prediction is made based on the class label or numerical value associated with that node.

In this study, a standard DecisionTreeClassifier() model was instantiated with default parameters. To optimize the model's performance, a custom function FT\_DT() is used to optimize the hyper-parameters of a Decision Tree model by performing a grid search [25] using the GridSearchCV class from the scikit-learn library.

First, the function initializes a Decision Tree Classifier object and defines a parameter grid over which the Grid Search algorithm searches for the best combination of hyper-parameters. The parameter grid consists of the criterion used to evaluate the quality of a split (criterion), the splitter strategy used to select the split at each node (splitter), the maximum depth of the tree (max\_depth), the maximum number of features considered for splitting a node (max\_features) and the minimum number of samples required to split an internal node (min\_samples\_splits) or to be considered at a leaf node(min\_samples\_leaf). Next, the function creates an instance of the GridSearchCV class with the following parameters:

1. estimator: The classifier object used to build models
2. param\_grid: The parameter grid to search over
3. scoring: The evaluation metric used to judge the performance of the models
4. n\_jobs: The number of CPUs to use for parallel processing (-1 means use all available CPUs)
5. cv: The cross-validation strategy used to evaluate the performance of the models
6. verbose: Controls the amount of information printed during the search
7. refit: If True, refit an estimator using the best-found parameters on the whole dataset.

The GridSearchCV object instance is created with the Decision Tree Classifier object and the parameter grid defined previously. The GridSearchCV object applies 6-fold cross-validation on the training data using a Repeated Stratified K-Fold object as a splitter. The scoring argument is set to 'accuracy' and n\_jobs is set to -1 to use all available CPU cores. The verbose argument is set to 3 to show progress of the search and refit is set to True to automatically refit the best estimator on the entire training data after finding the best hyperparameters.

Next, the function fits the GridSearchCV object to the sampled training data, records the time taken for the search and prints the number of candidate parameter settings. The function then prints the best hyper-parameters found, the best estimator and the best score of the Grid Search algorithm.

Lastly, the function applies the best estimator to the test data, computes the accuracy of the predictions, and prints the accuracy score. The best parameter found through hyperparameter tuning was then used to create the fine-tuned model, which had a maximum depth of 16 and utilized the Gini criterion with the best splitter.

1. Random Forest Model

The Random Forest is an ensemble learning technique that combines multiple decision trees to construct a more reliable and precise model. Its randomization process involves the random sampling of both the training data and features used for prediction at each tree node. This technique generates multiple decision trees that are then merged to create a single model.

The fundamental working principle of the Random Forest involves creating multiple decision trees, each trained on a randomly selected subset of the training data. The randomization process ensures that each tree has a different view of the feature space, thereby reducing overfitting and increasing the overall predictive accuracy. At each node of each decision tree, a subset of features is randomly selected for splitting the data. This process ensures that the trees are not biased towards specific features and that the final model is more robust to noisy or irrelevant features.

The final prediction is obtained by combining the predictions made by each tree using a majority voting scheme, where the prediction with the highest number of votes is selected as the final prediction. This approach improves the predictive accuracy of the model and reduces the risk of overfitting.

This study employed a standard RandomForestClassifier() model with default parameters. To improve the performance of a standard model, a custom function called FT\_RF() is used. The FT\_RF() function implements a grid search algorithm using the GridSearchCV class from the scikit-learn library to optimize the hyperparameters of the Random Forest model. The function defines a parameter grid that includes various hyperparameters such as the number of trees in the forest (n\_estimators), the minimum number of samples required to split an internal node (min\_samples\_split), the minimum number of samples required to be at a leaf node (min\_samples\_leaf), the maximum depth of the tree (max\_depth), the criterion to measure the quality of a split (criterion), the maximum number of features considered for splitting a node (max\_features), and whether bootstrap samples are used when building trees (bootstrap).

The Grid Search algorithm evaluates all possible hyperparameter combinations defined in the parameter grid to identify the combination that results in the highest model accuracy. The function creates a GridSearchCV object that takes the RandomForestClassifier() as an estimator, the param\_grid dictionary containing the hyperparameters, the scoring parameter set to 'accuracy', the number of jobs set to -1 for parallel processing, and a cv parameter that specifies the cross-validation strategy using RepeatedStratifiedKFold. The GridSearchCV object is then fitted to the training data.

The Random Forest fine-tuned model was created using the optimal parameter identified through hyperparameter tuning. This parameter included a bootstrap setting of False, a max depth of 18, 200 estimators, and a maximum feature setting of 'sqrt'.

1. Support Vector Machine Model

The Support Vector Machine (SVM) algorithm is a powerful classification method that employs a maximum margin approach to separate data into distinct categories. The underlying principle of SVM entails identifying a hyperplane that maximizes the margin, defined as the distance between the closest data points from each class and the decision boundary, to separate data into distinct classes. Support vectors, i.e., data points closest to the decision boundary, play a critical role in determining its location. Although SVM is primarily designed for binary classification, it can be extended to handle multiclass classification problems through techniques, such as the One-vs-All (OVA) and One-vs-One (OVO) approaches.

In the OVA approach, a multiclass problem is transformed into several binary classification problems. Specifically, N separate binary classifiers are trained for a problem with N classes, where each classifier distinguishes between one class and the remaining classes. During the prediction phase, each classifier is employed to estimate the probability of an instance belonging to its corresponding class, and the class with the highest likelihood is selected as the final prediction. In contrast, the OVO approach transforms the problem into N(N-1)/2 binary classification problems, where each classifier distinguishes between two different classes. For instance, for a three-class classification problem, three binary classifiers would be trained: hypothyroid vs. hyperthyroid, hypothyroid vs. negative, and hyperthyroid vs. negative. During the prediction phase, each classifier is used to make a prediction, and the class with the most votes is selected as the final prediction.

In scikit-learn, the SVC classifier utilizes the OVO approach by default for multiclass classification, whereas the LinearSVC classifier uses the OVA approach. In this study, I created a standard classifier using both OVA and OVO methods for multiclassification. The performance of SVM for multiclass classification primarily depends on the choice of kernel function, regularization parameter, and other hyperparameters. These hyperparameters are tuned using grid search techniques to improve classification accuracy.

The code comprises two functions, namely FT\_OVO\_SVM() and FT\_OVA\_SVM(), that fine-tune the hyperparameters of two different SVM models using Grid Search Cross-Validation. The FT\_OVO\_SVM() function optimizes the hyperparameters of an OVO SVM model. The function first creates an SVM classifier object using the "SVC" function, setting the "decision\_function\_shape" parameter to "ovo." Next, it instantiates a One-vs-One classifier object using the "OneVsOneClassifier" function with the previously created SVM classifier object as the parameter. The function defines a parameter grid to search over, including values for the "C," "kernel," and "gamma" parameters of the SVM model. The "GridSearchCV" function is then utilized to perform a grid search with cross-validation to determine the best combination of hyperparameters. Finally, the model is used to predict the class labels for the test data, and the accuracy of the model is calculated using the "accuracy\_score" function of the "metrics" module in Scikit-Learn.

The FT\_OVA\_SVM() function follows a similar process to the FT\_OVO\_SVM() function, except that the SVM classifier object is created using the " LinearSVC" function. The rest of the code is identical to the first function.

The optimizing function code performs hyperparameter tuning of two different SVM models using Grid Search Cross-Validation and prints the best hyperparameters, best estimator, and best score for each model. After identifying the optimal parameter through hyperparameter tuning, it was used to create the fine-tuned models for OVA SVM and OVO SVM. For OVO SVM, the parameters used were C=1000, decision\_function\_shape='ovo', gamma='auto', and kernel='rbf'. On the other hand, for OVA SVM, the parameters used were C=100, loss='hinge', multi\_class='crammer\_singer', and penalty='l1'.

1. Naive Bayes Model

Naïve Bayes is a statistical learning algorithm that is based on the Bayes theorem and is used for classification problems. It is a popular and simple method for text classification, sentiment analysis, and spam filtering. The term "Naive" refers to the assumption of independence between the features, which is often not true in real-world problems.

The working principle of Naive Bayes involves calculating the probability of each class given a set of features, and then selecting the class with the highest probability as the predicted class. The probabilities are estimated using the Bayes theorem, which states that the probability of a class given a set of features is proportional to the prior probability of the class multiplied by the likelihood of the features given the class. Mathematically, Bayes' theorem can be expressed as shown in equation 1 below:

Where is the posterior probability of class event A (target) occurring given that event B (predictor attributes) has occurred. represents the probability of event B occurring given that event A has occurred, also known as likelihood. is the prior probability of class event A occurring, while denotes the probability of event B occurring.

In Naive Bayes, the features are assumed to be independent, which means that the likelihood of the features given the class can be calculated as the product of the individual feature probabilities. This assumption allows for a computationally efficient calculation of the class probabilities, which makes Naive Bayes a fast and effective algorithm for large datasets. There are three primary types of Naive Bayes models: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. The selection of the appropriate model depends on the nature of the data being analyzed.

The present study utilized a dataset comprising both continuous and boolean variables. To accommodate this mixed data type, the Gaussian and Bernoulli Naive Bayes models were selected as standard classifiers. In order to further enhance model performance and accuracy, a hybrid Naive Bayes model was developed. This approach effectively leverages the strengths of both the Gaussian and Bernoulli models for handling continuous and binary data, respectively. By combining the predictions generated by each model, the hybrid approach offers a more comprehensive solution for handling mixed data types.

To implement the hybrid naive Bayes classifier, the HybridNB class was defined. This class inherits from BaseEstimator and ClassifierMixin, which are classes provided by the scikit-learn library. The class contains a constructor and three methods: init, fit, predict, and predict\_proba.

The constructor of the HybridNB class takes three parameters: gaussian, bernoulli, and alpha. The gaussian and bernoulli parameters represent instances of GaussianNB and BernoulliNB classifiers, respectively. The alpha parameter controls the weight assigned to the Gaussian and Bernoulli classifiers in the hybrid classifier.

The fit() method of the HybridNB class fits the Gaussian and Bernoulli classifiers to the input data. The method takes two parameters: X and y, representing the input features and labels, respectively. The method utilizes slicing to partition the input data into two parts: one consisting of continuous data, which includes the age, TSH, T3, T4U, and FTI columns; and the other consisting of Boolean data, which encompasses the remaining column attributes from the dataset. The Gaussian classifier is fitted to the continuous data, while the Bernoulli classifier is fitted to the Boolean data. The method returns self, which is the fitted HybridNB object.

The predict\_proba() method calculates the predicted probabilities using the two Naive Bayes models and the weight parameter alpha. It first predicts class probabilities for the continuous data using the Gaussian Naive Bayes model and for the Boolean data using the Bernoulli Naive Bayes model. Subsequently, it calculates the weighted average of these probabilities based on the value of alpha. The predict() method predicts the target values using the predict\_proba() method which is employed to estimate the probability distribution of the input data belonging to each class, thereby enabling the prediction of target values. Upon obtaining the probability estimates, the predict method selects the class with the highest probability and returns it as the predicted label for the input data.

In order to implement the Hybrid Naïve Bayes classifier model, a pipeline was designed to handle datasets containing both continuous and discrete features. The pipeline was constructed using the make\_pipeline() function, which comprises three stages. The first stage is a column\_transformer, which applies distinct preprocessing steps to columns based on the nature of the input data. Specifically, a StandardScaler() transformer was applied to the continuous features, while a Binarizer() transformer was applied to the discrete features. The output of the column transformer is a 2D array containing 20 columns, where five columns represent transformed continuous features and 15 columns represent transformed discrete features.

To enable the data to be fed into the next stage of the pipeline, the second stage involves a FunctionTransformer that flattens the 2D output of the column\_transformer to a 1D array. The third stage of the pipeline consists of a custom estimator named HybridNB, which combines the Gaussian Naive Bayes and Bernoulli Naive Bayes models to handle datasets containing both continuous and discrete features. The flattened data is fed into the HybridNB() estimator, which is instantiated using GaussianNB() and BernoulliNB() objects to represent the Gaussian and Bernoulli Naive Bayes models, respectively.

To optimize the performance of the HybridNB model, a custom function called FT\_HybridNB() was defined to perform hyperparameter tuning through a grid search. The function explored a range of values for two key hyperparameters: alpha and the Gaussian Naive Bayes var\_smoothing parameter. The HybridNB model combines Bernoulli and Gaussian components, with the alpha parameter serving as a weighting factor that determines the degree to which the model relies on each component for making predictions. An alpha value of 0 or 1 corresponds to a pure Gaussian Naive Bayes or pure Bernoulli Naive Bayes model, respectively.

To identify the best hyperparameters to optimize the performance of the model, the parameter space was explored using a grid search with the param\_grid parameter. The numpy logarithmically-spaced functions np.logspace(0,-9, num=100) and np.logspace(-3, 0, num=1000) were used to generate values for var\_smoothing and alpha, respectively. Specifically, 100 values were created between 1 and 0.000000001 for var\_smoothing and 1000 values were created between 0.001 and 1.0 for alpha. The aim was to determine the optimal hyperparameters to fine-tune the HybridNB model.

Ultimately, the fine-tuned HybridNB model was constructed using parameters ‘alpha’ set to 0.46415888336127775 and ‘var\_smoothing’ set to 0.0657933224657568, which were identified through the hyperparameter tuning process. This approach enables a streamlined and automated process for preprocessing the data and fitting the model, thereby providing a comprehensive solution for handling mixed data types.

1. Logistic Regression Model

Logistic Regression is a commonly used statistical method for binary classification problems, wherein the objective is to predict a binary outcome based on a set of input features. It is a specialized form of generalized linear model that models the relationship between the dependent variable and the independent variables through a logistic function. The logistic function transforms any real-valued input into a value between 0 and 1, which can be interpreted as the probability of the positive class.

The fundamental principle of Logistic Regression involves determining the optimal coefficients that minimize the difference between the predicted probabilities and the actual class labels. Maximum likelihood estimation is used to estimate these coefficients, which maximizes the probability of observing the observed data given the coefficients. These coefficients are then utilized to predict the probability of the positive class based on a given set of input features.

The decision boundary is a hyperplane that separates the positive and negative classes, and its location is determined by the coefficients. Instances positioned on one side of the boundary are classified as the positive class, while instances positioned on the other side are classified as the negative class. Predictions for the positive class are generated by applying a threshold to the predicted probability, with instances possessing a predicted probability greater than the threshold being classified as the positive class.

While Logistic Regression is primarily intended for binary classification, it can be expanded to handle multiclass classification problems using different strategies. One-vs-Rest (OvR) entails training a separate binary logistic regression model for each class, with samples belonging to the corresponding class designated as positive samples, and the remainder labeled as negative samples. During inference, the model with the highest probability score is selected as the predicted class. Alternatively, Multinomial Logistic Regression (MLR) utilizes a single logistic regression model to directly predict the probabilities of each class using a Softmax activation function. This approach optimizes a single cost function that accounts for all classes simultaneously.

In this study, a standard classifier was developed for multiclass classification using both One-vs-Rest (OVR) and Multinomial Logistic Regression (MLR) methods. The LogisticRegression classifier was employed to construct the standard classifier model for both methods by setting the multi\_class parameter value to ‘ovr’ and ‘multinomial’ for OVR and MLR, respectively. The performance of the LogisticRegression algorithm for multiclass classification is primarily dependent on the choice of hyperparameters, such as penalty, C and solver. These hyperparameters are tuned using grid search techniques to improve classification accuracy.

To optimize the hyperparameters of two logistic regression models using grid search cross-validation by RepeatedStratifiedKFold. To achieve this, two functions, namely FT\_OVR\_LR() and FT\_MLR(), have been defined for one-vs-rest (OvR) logistic regression and multinomial logistic regression (MLR) model, respectively.

The FT\_OVR\_LR() function initializes an OvR logistic regression model and defines a parameter grid to search over. The parameter grid consists of values of the C hyperparameter, which controls the strength of the regularization, along with different regularization penalties (l1, l2, and elasticnet), optimization algorithms (liblinear and saga), and maximum numbers of iterations for optimization.

Similarly, the FT\_MLR() function optimizes the hyperparameters of the MLR model to enhance its classification accuracy. The param\_grid dictionary is used to define the hyperparameters to tune using Grid Search, including exploring various values of the C hyperparameter, regularization penalties (l2 and elasticnet), optimization algorithms (newton-cg, lbfgs, sag, and saga), and maximum numbers of iterations for optimization.

On running the hyperparameter tuning process, the best hyperparameters were identified for each model. In the case of the OvR LR model, the optimal hyperparameters were C set to 200, penalty set to l1, solver set to liblinear, and max\_iter set to 1000. On the other hand, for the MLR model, the optimal hyperparameters were C set to 0.001, penalty set to 'none', 'fit\_intercept' set to True, 'max\_iter' set to 1000, and 'solver' set to 'lbfgs'.

1. Linear Discriminant Analysis Model

Linear Discriminant Analysis (LDA) [2] is a statistical method used for multi-class classification problems where the objective is to predict a categorical target outcome based on a set of input features. It achieves this by transforming the original features into a lower-dimensional space that is optimized for the separation of classes. As a dimensionality reduction technique, LDA maximizes the distance between-classes while minimizing the distance within-class.

The principle behind LDA involves the identification of the optimal linear combination of features that maximizes the separation of classes while minimizing intra-class scatter. This optimal linear combination of features is defined as the discriminant vectors and forms the basis for the lower-dimensional space. The computation of discriminant vectors involves utilizing the mean vectors and covariance matrices of each class to calculate within-class and between-class scatter matrices. The discriminant vectors are then selected as the eigenvectors of the between-class scatter matrix corresponding to the largest eigenvalues.

Once the discriminant vectors are obtained, the original features are projected onto the lower-dimensional space defined by these vectors. Subsequently, these projected features are utilized to make predictions, and a nearest neighbor class is assigned to a given instance based on the Euclidean distance between the instance and mean vectors of each class in the lower-dimensional space. This results in a decision boundary that separates the classes.

In this study, the standard Linear Discriminant Analysis (LDA) classifier is initially constructed using default parameters. However, in order to improve its performance, a custom function named FT\_LDA() is defined to optimize the hyperparameters of the LDA model by employing a Grid Search technique.

The fine-tuning process in the FT\_LDA() function is initiated by creating an instance of the LDA classifier, which is then subjected to a parameter grid search. The parameter grid search is defined as a dictionary that encapsulates various hyperparameters, such as solver, shrinkage, tolerance, and the number of components, that require optimization. Subsequently, an instance of the GridSearchCV object is created to perform the grid search technique using a six-fold cross-validation strategy. To mitigate the risk of overfitting, the RepeatedStratifiedKFold method is used with two splits and three repeats to ensure the model's performance evaluation is reliable.

Once the cross-validation strategy is established, the function logs the current time and trains the GridSearchCV object with the training data. It then outputs the best hyperparameters and the best score for the LDA model. The fine-tuned LDA model is constructed with optimized parameters, including 'n\_components' set to None, 'shrinkage' set to 'auto', 'solver' set to 'lsqr', and 'tol' set to 0.0001.

1. K-Nearest Neighbors Classifier Model

The K-Nearest Neighbors Classifier (KNN) is a classification algorithm commonly used in supervised learning[19]. It operates on the basis of a distance-based principle, where the similarity between instances is determined by their distance in the feature space. The algorithm makes predictions by computing the distances between the test instance and all training instances, selecting the K nearest neighbors, and using the majority class of those neighbors to predict the class of the test instance. The choice of distance metric in KNN is dependent on the data characteristics and the problem being addressed. The Euclidean distance is frequently used for continuous data, whereas the Manhattan distance is more appropriate for data with categorical or binary features. The Minkowski distance can handle mixed data types and can be parameterized with a power parameter.

The K parameter is an essential hyperparameter in KNN that determines the number of neighbors to consider when making predictions. The value of K affects the bias-variance trade-off. A larger value of K results in a smoother decision boundary but may cause the model to underfit, while a smaller value of K results in a more complex decision boundary and may lead to overfitting. Therefore, selecting an optimal value for K is crucial for obtaining a well-performing model.

The use of weights is another important aspect of the KNN algorithm during the prediction process. Two commonly used weight functions in KNN are "uniform" and "distance." In the "uniform" weight function, all neighbors are given equal weight in the prediction process, regardless of their distance to the test instance. In contrast, the "distance" weight function assigns weights to each neighbor based on its distance to the test instance. The choice of weight function can significantly impact the performance of the KNN algorithm, affecting the decision boundary and the accuracy of the predictions.

In this study, a standard KNN model with default hyperparameters is constructed. However, KNN can be computationally expensive for large datasets and high-dimensional feature spaces, leading to the curse of dimensionality. Therefore, it is essential to choose appropriate hyperparameter values, including the distance metric, K, and the weight function, for an accurate and high-performing KNN model. To address this issue, a customized approach called FT\_KNN() is proposed to fine-tune the KNN model using a Grid Search technique. This approach involves creating an instance of the KNN classifier and specifying a parameter grid search that encapsulates the hyperparameters to be optimized, such as the k value, power parameter, weight function, leaf size, and algorithm for computing nearest neighbor. The GridSearchCV object is then utilized with a six-fold cross-validation strategy and the RepeatedStratifiedKFold method with two splits and three repeats to evaluate the model's performance reliably and reduce the risk of overfitting.

To strike a balance between Euclidean and Manhattan distances in the presence of mixed data types (continuous and binary), the Minkowski distance is used for modeling the distance between instances. The Minkowski distance incorporates a power parameter, which can take on decimal values. This parameter is determined by generating an array of 100 numbers that are evenly spaced on a log scale between the values of 1 and 2, utilizing the np.logspace() function. By restricting the value of the power parameter within the range of 1 to 2, the resulting distance metric falls between the Manhattan distance (p=1) and the Euclidean distance (p=2), thus achieving a desired balance between these two measures.

The leaf size parameter in the KNN algorithm denotes the number of points stored in a leaf node of the K Decision tree data structure used for efficient nearest neighbor search. Selecting an appropriate leaf size is vital for obtaining optimal model performance and depends on several factors, including dataset size, input features, data dimensionality, and distribution. For a KNN model consisting of 20 input features in one dimension that comprises continuous and binary data, a small leaf size is deemed appropriate. An array of 100 numbers is generated using the np.logspace() function that is evenly spaced on a logarithmic scale between 1 and 50 to determine the optimal leaf size value for the model.

In the scikit-learn's KNeighborsClassifier, the 'algorithm' hyperparameter specifies the algorithm utilized to compute the nearest neighbors, with options such as 'auto', 'ball\_tree', 'kd\_tree', and 'brute'. The 'auto' option automatically selects the most suitable algorithm based on n\_samples and n\_features, while the 'ball\_tree' and 'kd\_tree' options use tree-based data structures, and the 'brute' option applies a brute-force search to find the nearest neighbors.

To establish the optimal number of nearest neighbors (K), a list of values is generated using the list(range(1,21)) function, which generates a list of integers from 1 to 20. The exploration of both uniform and distance weights, as well as other hyperparameters, is undertaken. Once the cross-validation strategy is established, the function records the current time and trains the GridSearchCV object utilizing the training data to identify the optimal values for K, the Minkowski distance power parameter, and the leaf size parameter that yield the best trade-off between bias and variance.

Finally, the KNN model is fine-tuned using the optimized hyperparameters 'algorithm' parameter is set to 'auto', 'n\_neighbors' parameter is set to 1, 'p' parameter is set to 1.200102719578103, and the 'weights' parameter is set to 'uniform'.

### Neural Network Framework

1. Multi-Layer Perceptron Model

The Multi-Layer Perceptron (MLP) is a widely-used artificial neural network (ANN) architecture that has been employed to solve various supervised learning problems, including regression and classification. The MLP is classified as a feedforward neural network, wherein information flows unidirectionally from the input to the output layer through multiple layers of artificial neurons or nodes.

The MLP architecture is composed of an input layer, one or more hidden layers, and an output layer. The input layer receives input features, which are then transformed through several hidden layers to generate a prediction at the output layer. Each hidden layer comprises several artificial neurons or nodes that connect to the input layer and subsequent hidden layers. The activation function or transfer function utilized by each node in the hidden layers determines the transformation of input features into predictions. During the training process, optimization algorithms such as gradient descent optimize the weights and biases of connections between nodes to minimize the prediction error between actual target values and predicted values. This optimization process, known as backpropagation, entails propagating error backward through the network to update the weights and biases.

The standard MLP classifier is constructed using default hyperparameters. To fine-tune the MLP model, a custom function called FT\_MLP() was developed to implement the Grid Search technique. This function creates a parameter grid that includes hyperparameters such as hidden layer sizes, activation function, solver, alpha, and learning rate. The GridSearchCV object is then instantiated with the classifier, parameter grid, six-fold cross-validation using the RepeatedStratifiedKFold method, and the scoring metric set to accuracy.

Another function, named FindLayerNodesLinear(), was developed to generate a list of the number of nodes in each layer of a linear neural network. This function takes three arguments: n\_layers, first\_layer\_nodes, and last\_layer\_nodes. It initializes an empty list named 'layers' and then calculates the increment in the number of nodes between consecutive layers by subtracting the number of nodes in the first layer from the number of nodes in the last layer and dividing the result by the number of layers minus one. This increment value is stored in a variable named 'nodes\_increment'.

Subsequently, the function initializes a variable named 'nodes' with the number of nodes in the first layer. Using a for loop, the function iterates over the range of 1 to n\_layers+1. During each iteration, the function appends to the 'layers' list the ceiling value of the 'nodes' variable, which represents the number of nodes in the current layer. The 'nodes' variable is then updated by adding the 'nodes\_increment' value to it. Finally, the function returns the 'layers' list, which contains the number of nodes in each layer of the linear neural network. The optimal combination of hidden layers that produces the best results is shortlisted and incorporated into the param\_grid for grid search, alongside the other relevant parameters.

The 'hidden\_layer\_sizes' hyperparameter determines the number of neurons in each hidden layer of the MLP model. Two different combinations of hidden layer sizes are specified, namely (60, 46, 32, 18, 3) and (80, 69, 58, 47, 36, 25, 14). These values are chosen based on the results of the FindLayerNodesLinear() function.

The 'activation' hyperparameter specifies the activation function used in the MLP model. Three different activation functions are considered, including 'tanh', 'relu', and 'logistic'. These activation functions transform the output of each neuron in a nonlinear way and have been shown to improve the performance of neural networks.

The 'solver' hyperparameter on the other hand controls the optimization algorithm used to train the MLP model. Three different solvers are considered, namely 'lbfgs', 'sgd', and 'adam'. The 'lbfgs' solver is an optimization algorithm that uses a quasi-Newton method to find the optimal weights of the MLP model. The 'sgd' solver uses stochastic gradient descent, a widely-used optimization algorithm that updates the weights of the MLP model based on small random subsets of the training data. The 'adam' solver combines the benefits of both lbfgs and sgd, by using adaptive learning rates and momentum to update the weights of the MLP model.

The 'alpha' hyperparameter controls the regularization strength of the MLP model. Regularization is a technique used to prevent overfitting by adding a penalty term to the loss function that discourages the weights from becoming too large. In this study, five different values of 'alpha' were considered, including 0.0001, 0.001, 0.01, 0.05, and 0.1.

The 'learning\_rate' hyperparameter controls the learning rate schedule of the MLP model. The learning rate is a hyperparameter that determines the step size taken in the direction of the negative gradient during training. Two different values of 'learning\_rate' are considered, including 'constant' and 'adaptive'. The 'constant' learning rate schedule maintains a constant learning rate throughout training, while the 'adaptive' learning rate schedule adapts the learning rate based on the validation loss.

Ultimately, after fine-tuning the MLP model, the optimized hyperparameters were determined to be a hidden layer size of (80, 69, 58, 47, 36, 25, 14), an activation function of 'tanh', a solver of 'Adam', an alpha of 0.1, and a learning rate of 'constant'.

1. **Neural framework 1 (Dense-Dropout)**

This framework utilizes a feedforward artificial neural network architecture for the classification of a dataset with 20 features and 3 classes. The architecture consists of three fully connected dense layers that employ the rectified linear unit (ReLU) activation function, followed by two dropout layers with a dropout rate of 0.2. The output layer adopts the softmax activation function to produce probabilities for each of the 3 classes. To define the model, the Sequential module is used to create a linear stack of layers, and the Dense module is used to create a fully connected layer. Additionally, the Dropout layer is imported to randomly drop some of the nodes during training, which helps to prevent overfitting.

The model is instantiated by defining a Sequential object, which is an empty linear stack of layers. The first layer added to the model is a Dense layer with 64 nodes, a ReLU activation function, and an input dimension of 20. This means that the input to the model has 20 features, and the first layer has 64 nodes that apply the ReLU activation function to the input data. The Dropout layer is then added to randomly drop 20% of the nodes in the first layer. The second layer is another Dense layer with 32 nodes and a ReLU activation function. Again, a Dropout layer is added to randomly drop 20% of the nodes in this layer. Finally, a Dense layer with 3 nodes and a softmax activation function is added to output the probabilities for each class.

To train the model, the categorical cross-entropy loss function, the Adam optimizer, and the accuracy metric were utilized to assess its performance during training. The model's summary, which includes the number of parameters and the shape of each layer, was displayed in the console. The model was then trained using the fit method with a batch size of 32 for number of epochs, with 20% of the training data employed for validation. Finally, the performance of the model is evaluated using the test dataset. The evaluation metrics used were the test loss and accuracy, which provide an estimation of the model's effectiveness on unseen data.

1. **Neural framework 2 (LSTM-Dense-Dropout)**

This proposed framework is developed using an LSTM-based classification model utilizing the Keras API. The model architecture comprises an LSTM layer, two Dropout layers, and two Dense layers. The sequential model architecture, consisting of Long Short-Term Memory (LSTM) and Dense layers, is designed for a classification problem. The initial step in the code is defining the input shape of the data as (20, 1), which means that each input sequence has 20-time steps and one feature per step. To match the expected input shape of the LSTM layer, the code reshapes the input data using np.swapaxes. This transformation is essential because the LSTM layer expects input data with three dimensions: samples, time steps, and features. By swapping the axes, the code transforms the input data from having the shape (samples, features, time steps) to (samples, time steps, features).

The model architecture consists of three layers: an LSTM layer, a Dropout layer, a Dense layer, another Dropout layer, and a final Dense layer. The first layer is an LSTM layer with 64 units and a relu activation function. The input\_shape argument specifies the input shape of the layer. The second layer is a Dropout layer with a dropout rate of 0.2, which is used to prevent overfitting in the model. The third layer is a Dense layer with 32 units and a relu activation function. The fourth layer is another Dropout layer with a dropout rate of 0.2. The final layer is a Dense layer with three units and a softmax activation function. The model's output is a probability distribution over the three classes.

To compile the model, categorical\_crossentropy is used as the loss function, adam as the optimizer, and accuracy as the evaluation metric. The model summary is printed to display the architecture and the number of trainable parameters in the model.

The training of the model is carried out using the fit() method with nX\_train and ny\_train as the input and output data, respectively. The training is performed for 50 epochs with a batch size of 32, and 20% of the training data is used as validation data. Finally, the model is evaluated on the test data using the evaluate() method, and the test loss and accuracy are printed to the console.

1. **Neural framework 3 (Sequential MLP-1DCNN)**

The proposed model architecture combines a multilayer perceptron (MLP) and a one-dimensional convolutional neural network (1D CNN) for the classification of a dataset with 20 input features and 3 output classes. The code implementation of this architecture starts by defining the input shape of the data as (20, 1), which implies that each input sequence has 20-time steps and one feature per step.

The model architecture consists of two main parts. The first part includes a stack of MLP layers, which consists of four fully connected layers with 64, 128, 128, and 3 units, respectively. The activation function used for all MLP layers is relu, except for the output layer, which uses a softmax activation function. To prevent overfitting, two Dropout layers with a rate of 0.5 are added to the MLP layers. The GlobalMaxPooling1D layer is also included in the MLP layers to reduce the dimensionality of the output.

The second part of the model comprises a sequence of 1D CNN layers. This stack of layers includes a reshape layer that aligns the output of the preceding MLP layers with the anticipated input shape of the CNN layers. The reshape layer transforms the output of the MLP layers into a three-dimensional tensor. The CNN layers consist of a convolutional layer that uses 64 filters, a kernel size of 3, and the rectified linear unit (ReLU) activation function. Furthermore, a MaxPooling1D layer with a pool size of 2 is added, which down samples the output of the convolutional layer. Finally, a flatten layer is appended to convert the output of the CNN layers into a one-dimensional vector. The output layer is a dense layer with three units and a softmax activation function to output the probability distribution over the three classes.

The model is compiled using categorical\_crossentropy as the loss function, adam as the optimizer, and accuracy as the evaluation metric. The model summary is printed to show the architecture and the number of trainable parameters. The model is trained using the fit() method with nX\_train and ny\_train as the input and output data, respectively. The training is performed for number of epochs with a batch size of 32, and 20% of the training data is used as validation data. Finally, the model is evaluated on the test data using the evaluate() method, and the test loss and accuracy are printed to the console.

1. **Neural framework 4 (Concatenated  MLP-1DCNN)**

The proposed hybrid model combines a multilayer perceptron (MLP) and a convolutional neural network (CNN) to classify dataset with 20 features into three class categories. The MLP model takes an input of shape (20,), where 20 represents the number of features in the input data. It has two hidden layers, the first with 64 units and the second with 32 units, both with the ReLU activation function. The output layer has three units with the softmax activation function, representing the probability distribution over the three categories.

The CNN model takes an input of shape (20, 1), where 20 represents the number of time steps and 1 represents the number of channels in the input data. It has two convolutional layers, the first with 32 filters of size 3 and the second with 64 filters of size 3, both with the ReLU activation function. The output of each convolutional layer is max-pooled with a pool size of 2. The output of the second convolutional layer is then flattened, and passed through a fully connected layer with three units and the softmax activation function.

The MLP and CNN models are then combined using a concatenation layer, which combines the output of the MLP and CNN models. The concatenated output is then passed through a fully connected layer with 64 units and the ReLU activation function. The final output layer has three units with the softmax activation function, representing the probability distribution over the three categories.

The hybrid model is compiled using the Adam optimizer with a learning rate of 0.001, the categorical cross-entropy loss function, and the accuracy metric. The model summary is printed, which provides details about the layers, the number of parameters, and the shape of the input and output tensors.

The hybrid model is then trained using the fit method with the training data and labels, a batch size of 32, and number of epochs. The validation split is set to 0.2, meaning that 20% of the training data is used for validation. The training history is stored in the history variable. Finally, the model is evaluated on the test data using the evaluate method, which computes the loss and accuracy metrics. The loss and accuracy are printed to the console.

## Model Evaluation

To evaluate the performance of the model on the test data, various commonly used evaluation metrics were computed. These metrics include accuracy, precision, f1-score, recall (sensitivity), and loss functions. The loss functions utilized in the evaluation process comprise of mean absolute error, mean squared error, and root mean square error.

**Accuracy**

Accuracy is a fundamental measure of model performance, expressed as the proportion of correctly predicted observations in relation to the total number of observations. The accuracy metric can be formulated mathematically as follows:

where True Positives (TP) refer to the number of correctly predicted positive instances, True Negatives (TN) represent the number of correctly predicted negative instances, False Positives (FP) denote the number of incorrectly predicted positive instances and False Negatives (FN) indicate the number of incorrectly predicted negative instances.

**Precision**

Precision measures the proportion of positive predictions that are actually correct. It is used to assess the accuracy of the model in making positive predictions. It is calculated as the ratio of True Positives (TP) to the sum of True Positives and False Positives (FP). The formula is as follows:

**Recall**

Recall, also known as Sensitivity, is a measure of the proportion of positive cases that the model correctly identifies. It is used to assess the model's ability to identify all relevant instances in the data. It is calculated as the ratio of True Positives (TP) to the sum of True Positives and False Negatives (FN). The formula is as follows:

**F1 Score**

It is a measure of the balance between precision and recall and provides a comprehensive assessment of the model's performance. Precision and recall are two important evaluation metrics that are used to assess the quality of machine learning models. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positive cases that are correctly predicted. The F1 Score is the harmonic mean of precision and recall, and it is calculated as follows:

**Mean Absolute Error (MAE)**

It is a popular loss function used in model evaluation that measures the average absolute difference between the predicted values and the actual values of the observations in the dataset. MAE is calculated using the following formula:

In this equation, yi represents the predicted value for an observation, and xi represents the true value for the same observation. The absolute value of the difference between the predicted and true values is computed and then averaged across the entire dataset to obtain the MAE score. Like other evaluation metrics, MAE cannot be negative since it is always taken in absolute terms. MAE is particularly useful when outliers in the data can significantly affect the model performance, as it is less sensitive to outliers than other loss functions like Mean Squared Error (MSE).

**Mean Squared Error (MSE)**

It measures the average of the squared differences between the predicted and true values of the observations in the dataset. The MSE is calculated using the following equation:

In this formula, yi represents the predicted value for an observation, and xi represents the true value for the same observation. The difference between the predicted and true values is squared and then averaged across the entire dataset to obtain the MSE score. The squaring component of the function ensures that the MSE value is always non-negative and prevents outlier predictions with large errors in the trained model. However, the squaring component also amplifies the error when the model makes a single incorrect prediction, making it more sensitive to outliers than other loss functions like Mean Absolute Error (MAE).

**Root Mean Squared Error (RMSE)**

This metric is used to measure the average distance between the predicted and true values of the observations in a dataset, using the Euclidean distance. RMSE is calculated by taking the square root of the average of the squared differences between the predicted and true values of the observations. The RMSE equation can be expressed as follows:

In this formula, represents the true value of the ith observation, represents the corresponding predicted value, and n represents the total number of observations. The squared differences between the predicted and true values are averaged across all observations and then the square root is taken to obtain the RMSE score. RMSE is a popular metric because it not only measures the error between predictions and true values but also considers the magnitude of the error. Additionally, RMSE is useful when the range of the target variable is large, as it provides a better understanding of the error magnitude in such cases.

# Experiment results and analysis

This study evaluates the effectiveness of various machine learning algorithms and neural network frameworks for the classification of thyroid disorders[11], utilizing the UCI thyroid disorder dataset containing both continuous and boolean variables. The algorithms under consideration include Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), Logistic Regression (LR), Linear Discriminant Analysis (LDA), K Nearest Neighbor (KNN), and Multi-Layer Perceptron (MLP). To fine-tune these models, the following parameters were employed:

1. DecisionTreeClassifier (DT) with criterion set to 'gini' and splitter set to 'best'.
2. RandomForestClassifier (RF) with bootstrap set to False, max\_features set to 'sqrt', 'max\_depth' set to 18 and n\_estimators set to 200.
3. Support Vector Classifier (SVC) for One vs One Support Vector Machine (OVO\_SVM) with C set to 1000, decision\_function\_shape set to 'ovo', kernel set to 'rbf', and gamma set to 'auto'.
4. Linear Support Vector Classifier (LinearSVC) for One vs All Support Vector Machine (OVA\_SVM) with C set to 100, dual set to False, loss set to 'squared\_hinge', and penalty set to 'l1'.
5. Hybrid Naive Bayes (HybridNB) with alpha set to 0.46415888336127775 and var\_smoothing set to 0.0657933224657568.
6. Logistic Regression for One vs Rest Logistic Regression (OVR\_LR) with C set to 1000, multi\_class set to 'ovr', penalty set to 'l1', and solver set to 'liblinear'.
7. Logistic Regression for Multinomial Logistic Regression (MLR) with C set to 0.001, max\_iter set to 1000, multi\_class set to 'multinomial', penalty set to 'none', and solver set to 'lbfgs'.
8. Linear Discriminant Analysis (LinearDA) with shrinkage set to 'auto', solver set to 'lsqr', and tol set to 0.0001.
9. k-Nearest Neighbors (KNN) with n\_neighbors set to 1, p set to 1.200102719578103, algorithm set to 'auto', and weights set to 'uniform'.
10. Multi-layer Perceptron (MLP) Classifier with activation set to 'tanh', alpha set to 0.1, hidden\_layer\_sizes set to (90, 78, 66, 53, 41, 28, 16), learning\_rate set to 'constant', and solver set to 'adam'.

The results and analysis of the experiments are discussed in the following sections.

## Experiment results for models trained on Smote sampled dataset

This section discusses the performance of several algorithmic models trained on a SMOTE sampled dataset, as detailed in Table 4.1.1. Results indicate that the Standard Random Forest (RF) model achieved the highest accuracy of 98.095%, closely followed by the Fine-tuned Random Forest (FT\_RF) model at 98.21%, while the Decision Tree (DT) model demonstrated relatively lower accuracy. Potentially due to its relatively simplistic nature, which struggles to capture the intricacies present in the dataset. The RF model is an ensemble learning algorithm that utilizes multiple decision trees to enhance accuracy while mitigating overfitting. RF models are well-suited for high-dimensional data and can effectively manage missing values, accounting for non-linear relationships between features and the target variable. However, overfitting may occur when the number of trees in the forest is excessively large or when individual trees are overly deep. To address this issue, the hyperparameters 'max\_depth' and 'max\_features' were judiciously selected, effectively constraining the model's tendency to overfit.

In contrast, SVMs are effective in high-dimensional spaces and can capture complex relationships between features, both linear and non-linear. However, the present study demonstrates that SVMs can be sensitive to kernel function selection and its corresponding hyperparameters and can be computationally expensive when applied to large datasets.

To evaluate the effectiveness of SVMs in multiclass classification tasks, the One-Vs-One SVM (OVO\_SVM) and One-Vs-All SVM (OVA\_SVM) models were employed, with the selection of appropriate hyperparameters expected to yield favorable outcomes. The 'rbf' kernel used in the OVO\_SVM approach was appropriate for capturing complex feature relationships in high-dimensional spaces, while the 'squared\_hinge' loss and 'l1' penalty selected for the OVA\_SVM approach were expected to enhance model performance in handling multiclass classification problems. The OVO and OVA SVM models exhibited accuracy rates of 87.73% and 87.38%, respectively. Fine-tuning these models yielded accuracy rates of 95.35% and 88.21%, respectively.

Table 4.1.1: Performance Metric Comparison of Standard and Fine-tuned Algorithmic Models trained using Smote sampled dataset.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **MAE** | **MSE** | **RMSE** |
| Decision Tree | DT | 96.90% | 0.85 | 0.91 | 0.97 | 0.0523 | 0.0952 | 0.3086 |
| FT\_DT | 97.38% | 0.86 | 0.91 | 0.97 | 0.0476 | 0.0904 | 0.3007 |
| Random Forest | RF | 98.095% | 0.88 | 0.96 | 0.98 | 0.0345 | 0.0654 | 0.2558 |
| FT\_RF | 98.21% | 0.90 | 0.94 | 0.98 | 0.0321 | 0.0607 | 0.2464 |
| Support vector machine | OVO\_SVM | 87.73% | 0.63 | 0.92 | 0.88 | 0.1952 | 0.3404 | 0.5835 |
| FT\_OVO\_SVM | 95.35% | 0.80 | 0.89 | 0.95 | 0.0726 | 0.125 | 0.3535 |
| OVA\_SVM | 87.38% | 0.62 | 0.89 | 0.87 | 0.1924 | 0.3190 | 0.5648 |
| FT\_OVA\_SVM | 88.21% | 0.64 | 0.90 | 0.88 | 0.1821 | 0.3107 | 0.5574 |
| Naïve bayes | GaussianNB | 31.07% | 0.40 | 0.61 | 0.31 | 1.0845 | 1.875 | 1.3693 |
| BernoulliNB | 52.85% | 0.41 | 0.47 | 0.53 | 0.5178 | 0.6107 | 0.7814 |
| HybridNB | 89.97% | 0.67 | 0.93 | 0.90 | 0.1714 | 0.3095 | 0.5563 |
| FT\_HybridNB | 90.83% | 0.69 | 0.93 | 0.91 | 0.1488 | 0.2630 | 0.5129 |
| Logistic Regression | OVR\_LR | 88.57% | 0.64 | 0.90 | 0.89 | 0.1738 | 0.2928 | 0.5411 |
| FT\_OVR\_LR | 89.52% | 0.66 | 0.91 | 0.90 | 0.1642 | 0.2833 | 0.5322 |
| MLR | 91.19% | 0.69 | 0.94 | 0.91 | 0.1440 | 0.2559 | 0.5059 |
| FT\_MLR | 92.21% | 0.71 | 0.93 | 0.92 | 0.1321 | 0.2392 | 0.4891 |
| Linear Discriminant Analysis | LDA | 88.80% | 0.66 | 0.89 | 0.89 | 0.1845 | 0.3297 | 0.5742 |
| FT\_LDA | 90.83% | 0.69 | 0.93 | 0.91 | 0.1523 | 0.2738 | 0.5232 |
| K Nearest Neighbor | KNN | 92.73% | 0.72 | 0.92 | 0.93 | 0.1047 | 0.1690 | 0.4111 |
| FT\_KNN | 93.45% | 0.76 | 0.86 | 0.93 | 0.0892 | 0.1369 | 0.3700 |
| Multi-Layer Perceptron | MLP | 91.78% | 0.69 | 0.89 | 0.92 | 0.1380 | 0.25 | 0.5 |
| FT\_MLP | 96.54% | 0.82 | 0.94 | 0.97 | 0.0607 | 0.1130 | 0.3362 |

Two Naïve Bayes models, Gaussian Naïve Bayes (GaussianNB) and Bernoulli Naïve Bayes (BernoulliNB), were found to have the lowest accuracy of 31.07% and 52.85%, respectively. However, a Hybrid Naïve Bayes (HybridNB) model, which combines both GaussianNB and BernoulliNB, exhibited superior performance by leveraging the strengths of both models in handling continuous and binary data, respectively. By optimizing the alpha parameter to balance the weights of these models, the Fine-tuned Hybrid Naïve Bayes (FT\_HybridNB) model achieved the highest accuracy of 90.83%.

Logistic Regression is a popular model for binary classification and can be extended to handle multi-class classification using various techniques. However, this model assumes a linear relationship between features and the target variable, and may be sensitive to outliers, requiring regularization. In this study, the 'l1' penalty and 'liblinear' solver were used for the One-vs-Rest Logistic Regression (OVR\_LR) model, while the 'lbfgs' solver was used for the Multinomial Logistic Regression (MLR) model, to reduce overfitting and optimize the objective function during training.

The accuracy of the Logistic Regression models ranged from 88.57% to 92.21%, with the MLR approach outperforming the OVR approach in terms of performance. Specifically, the standard MLR approach achieved accuracies of 91.19% and 92.21% for the MLR and Fine-tuned MLR (FT\_MLR) models, respectively. The superiority of the MLR approach over the OVR approach is attributed to its high accuracy and F1 scores. Thus, the results suggest that the MLR approach is more suitable for the given dataset, although the FT\_OVR\_LR approach may be a viable alternative owing to its high precision and recall values.

The LDA model is a simple and efficient approach that is particularly suitable for small to moderate-sized datasets. However, it may not perform well when the data is not normally distributed and when the classes have different covariance matrices. To address these issues, the LDA model is employed with solver set to 'lsqr', and tol set to 0.0001 to improve the accuracy of the model by further addressing the issue of multicollinearity in the data. The results indicate that the LDA model achieves an accuracy of 88.80%, which suggests that it can effectively predict the target variable. Nonetheless, the precision and recall values are relatively low, indicating that the model may have difficulty correctly classifying instances of the minority class. The FT\_LDA approach, which includes shrinkage and solver methods, improves the accuracy of the model to 90.83%, along with higher precision and recall values, suggesting that LDA can be an effective method for classification tasks.

The KNN model is a non-parametric method that is simple to implement and does not require any assumptions about the underlying data distribution. The model is employed with n\_neighbors set to 1, p set to 1.200102719578103, and weights set to 'uniform', which ensures that each neighbor has an equal contribution to the classification decision. The results indicate that the KNN model achieves an accuracy of 92.73%, indicating that it can accurately classify the target variable. The FT\_KNN approach further improves the accuracy of the model to 93.45%, along with relatively higher precision and recall values.

The Multi-layer Perceptron (MLP) is a powerful and flexible neural network model that can handle non-linear relationships between the features and the target variable. The use of the 'tanh' activation function and the 'adam' solver is aimed to help the model converge faster and improve the accuracy of the model. The MLP model achieved an accuracy of 91.78%, indicating that it can accurately classify the target variable. The FT\_MLP approach showed significant improvement, achieving an accuracy of 96.54% with high precision and recall values.

## Experiment results for models trained on Kmeans Smote sampled dataset

This section presents the experimental results obtained by training different machine learning models on the K-means SMOTE sampled dataset. Table 4.1.2 shows the performance metrics comparison of standard and fine-tuned algorithmic models, while also indicating their accuracy, precision, recall, F1 score, mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE).

The results show that the Fine-tuned Random Forest (FT\_RF) model achieved the highest accuracy score of 98.33%, followed closely by the Fine-tuned Multi-Layer Perceptron (FT\_MLP) and MLP models with accuracy scores of 96.66% and 96.07%, respectively. Conversely, the GaussianNB and BernoulliNB models achieved the lowest accuracy scores of 37.26% and 53.33%, respectively.

In addition to accuracy scores, precision and recall scores were also evaluated for each model. The Decision Tree and Random Forest models achieved the highest precision and recall scores, with all models exhibiting at least 0.91 precision and 0.92 recall. The SVM models displayed lower accuracy scores than other models, but their performance significantly improved with fine-tuning techniques.

The HybridNB and FT\_HybridNB models exhibited similar performance metrics, indicating that fine-tuning did not significantly enhance their performance. Likewise, LDA and FT\_LDA models demonstrated comparable accuracy scores, with FT\_LDA exhibiting a slight improvement in precision, recall, and F1 Score.

Table 4.1.2: Performance Metric Comparison of Standard and Fine-tuned Algorithmic Models trained using K-means Smote sampled dataset.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **MAE** | **MSE** | **RMSE** |
| Decision Tree | DT | 97.85% | 0.88 | 0.92 | 0.98 | 0.0392 | 0.075 | 0.2738 |
| FT\_DT | 98.21% | 0.91 | 0.91 | 0.98 | 0.0333 | 0.0642 | 0.2535 |
| Random Forest | RF | 98.21% | 0.91 | 0.92 | 0.98 | 0.0309 | 0.0571 | 0.2390 |
| FT\_RF | 98.33% | 0.91 | 0.93 | 0.98 | 0.0297 | 0.0559 | 0.2365 |
| Support vector machine | OVO\_SVM | 89.88% | 0.70 | 0.774 | 0.90 | 0.1476 | 0.2404 | 0.4903 |
| FT\_OVO\_SVM | 95.55% | 0.81 | 0.85 | 0.96 | 0.0690 | 0.1190 | 0.3450 |
| OVA\_SVM | 91.54% | 0.69 | 0.86 | 0.92 | 0.1238 | 0.2023 | 0.4498 |
| FT\_OVA\_SVM | 92.02% | 0.70 | 0.86 | 0.92 | 0.1154 | 0.1869 | 0.4323 |
| Naïve bayes | GaussianNB | 37.26% | 0.39 | 0.54 | 0.37 | 1.0285 | 1.8509 | 1.3531 |
| BernoulliNB | 53.33% | 0.39 | 0.43 | 0.53 | 0.5059 | 0.5845 | 0.7645 |
| HybridNB | 89.97% | 0.73 | 0.73 | 0.90 | 0.1333 | 0.2095 | 0.4577 |
| FT\_HybridNB | 90.47% | 0.73 | 0.73 | 0.90 | 0.1333 | 0.2095 | 0.4577 |
| Logistic Regression | OVR\_LR | 93.21% | 0.74 | 0.89 | 0.93 | 0.0964 | 0.1535 | 0.3918 |
| FT\_OVR\_LR | 93.33% | 0.74 | 0.87 | 0.93 | 0.0928 | 0.1452 | 0.3811 |
| MLR | 95.59% | 0.81 | 0.89 | 0.96 | 0.0702 | 0.1226 | 0.3501 |
| FT\_MLR | 95.11% | 0.80 | 0.86 | 0.95 | 0.0738 | 0.1238 | 0.3518 |
| Linear Discriminant Analysis | LDA | 90.23% | 0.69 | 0.73 | 0.90 | 0.1440 | 0.2369 | 0.4867 |
| FT\_LDA | 90.83% | 0.72 | 0.74 | 0.91 | 0.1285 | 0.2023 | 0.4498 |
| K Nearest Neighbor | KNN | 94.52% | 0.80 | 0.84 | 0.95 | 0.0761 | 0.1190 | 0.3450 |
| FT\_KNN | 94.28% | 0.81 | 0.79 | 0.94 | 0.0785 | 0.1214 | 0.3484 |
| Multi-Layer Perceptron | MLP | 96.07% | 0.83 | 0.88 | 0.96 | 0.0619 | 0.1071 | 0.3273 |
| FT\_MLP | 96.66% | 0.85 | 0.88 | 0.97 | 0.0535 | 0.0940 | 0.3066 |

In contrast, Logistic Regression models demonstrated considerable improvement, achieving a 5.67% increase in performance with the FT\_MLR model. KNN models performed well in terms of accuracy, precision, and recall, with KNN and FT\_KNN achieving accuracy scores of 94.52% and 94.28%, respectively.

## Experiment results for neural network models

This section presents the results of experiments conducted to evaluate the performance of various neural network models for the classification of thyroid disorders. One important factor that affects the accuracy and performance of machine learning algorithms is the number of epochs used in training. Table 4.1.3 shows the test loss and test accuracy of four different neural network frameworks, with varying numbers of epochs.

The results indicate that the accuracy of each model was affected by the number of epochs used in training. Neural frameworks 2 and 4, which consist of LSTM-Dense-Dropout and Concat MLP-1DCNN architectures respectively, displayed a marked improvement in accuracy as the number of epochs increased. In contrast, neural frameworks 1 and 3, which use Dense-Dropout and Sequential MLP-1DCNN architectures respectively, showed a decline in accuracy as the number of epochs increased, indicating suboptimal model optimization for the data or overfitting of the training data.

Table 4.1.3: Comparison of neural network framework model performance

|  |  |  |  |
| --- | --- | --- | --- |
| **Framework** | **Epoch** | **Test Loss** | **Test Accuracy** |
| Neural framework 1 (Dense-Dropout) | 50 | 0.10811 | 97.02 |
| 100 | 0.11695 | 96.42 |
| 200 | 0.13327 | 96.42 |
| 500 | 0.26015 | 96.19 |
| Neural framework 2 (LSTM-Dense-Dropout) | 50 | 0.15381 | 94.40 |
| 100 | 0.13208 | 95.35 |
| 200 | 0.13384 | 96.42 |
| 500 | 0.26028 | 96.30 |
| Neural framework 3 (Sequential MLP-1DCNN) | 50 | 0.35587 | 90.23 |
| 100 | 0.40131 | 90.11 |
| 200 | 0.56664 | 85.95 |
| 500 | 0.26933 | 85.23 |
| Neural framework 4 (Concat MLP-1DCNN) | 50 | 0.11071 | 95.71 |
| 100 | 0.10523 | 96.30 |
| 200 | 0.12312 | 96.42 |
| 500 | 0.19398 | 96.78 |

Notably, neural framework 4 demonstrated the highest accuracy among the tested models, with a maximum accuracy of 96.78% achieved at epoch 500. These results suggest that increasing the number of epochs can further enhance the accuracy of the model, indicating that the model can continue to learn from the data over time.

## Analysis

The study conducted an analysis of the performance metric comparison between standard and fine-tuned algorithmic models trained using SMOTE and K-means SMOTE sampled datasets. The following key findings were made:

1. The Random Forest algorithm outperformed all other models in terms of accuracy, precision, recall, F1 score, and RMSE, achieving 98.21% and 98.33% for SMOTE and K-means SMOTE sampled datasets, respectively. The ensemble nature of the algorithm made it less prone to overfitting and more robust to noise in the data.
2. The Support Vector Machine (SVM) algorithm performed significantly better after fine-tuning for both SMOTE and K-means SMOTE sampled datasets. The fine-tuned SVM models achieved an accuracy of 95.35% and 95.55% for SMOTE and K-means SMOTE sampled datasets, respectively, which is a considerable improvement from the standard SVM models' accuracy.
3. The Gaussian Naive Bayes (GNB) algorithm performed poorly, achieving only 31.07% accuracy for SMOTE sampled dataset and 38.92% accuracy for the K-means SMOTE sampled dataset. The other two variants of Naive Bayes, BernoulliNB and HybridNB, achieved better accuracy rates for both datasets, but still significantly lower than the top-performing algorithms.
4. Each algorithm has its strengths and weaknesses, and it's essential to note that while the Random Forest algorithm performed the best, it may not be the most interpretable model due to its ensemble nature. On the other hand, decision trees and logistic regression models are more interpretable but may suffer from overfitting if the hyperparameters are not well-tuned.

The study also revealed that the fine-tuned algorithmic models trained using the K-Means SMOTE sampled dataset outperformed their counterparts trained using the standard SMOTE sampled dataset. The K-Means SMOTE approach enhances the effectiveness of the SMOTE algorithm by clustering the data points into several subgroups, thereby enabling the algorithm to generate more realistic and diverse synthetic samples.

Finally, the assessment of the Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) revealed considerably higher values for Naive Bayes models in comparison to other models, suggesting a higher degree of error in the predictions made by Naive Bayes models. However, the HybridNB model displayed superior performance relative to the GaussianNB and BernoulliNB models, ameliorating their elevated error rates. The analysis of MAE, MSE, and RMSE values for the remaining models yielded relatively low values, signifying the robust predictive capabilities and high accuracy of the models assessed in this study.

# Conclusion

This thesis investigated various machine learning techniques for the classification of thyroid disorders. The results showed that the Random Forest algorithm performed better than other algorithms in terms of accuracy, precision, recall, and F1 score. These findings suggest that machine learning can be a valuable tool for precise and efficient classification of thyroid disorders, aiding healthcare professionals in making informed decisions and providing prompt treatment to patients.

Furthermore, the study emphasized the importance of data pre-processing, feature selection, and oversampling techniques in enhancing the effectiveness of machine learning models for thyroid disorder classification. Techniques such as data cleaning, integration, dimension reduction, and outlier detection were identified as valuable in improving data quality and model performance. Similarly, feature engineering techniques such as transforming skewed data, encoding, and mitigating multicollinearity were shown to enhance feature quality and relevance, leading to improved model performance.

Moreover, oversampling techniques such as SMOTE and Kmeans SMOTE were found to effectively address the issue of imbalanced data and improve model performance. The study showed that fine-tuned algorithmic models trained using the K-Means Smote sampled dataset outperformed their counterparts trained using the standard Smote sampled dataset.

Overall, the findings of this study offer valuable insights into the use of machine learning techniques for thyroid disorder classification and can serve as a basis for further research in this area. The study highlights the potential of machine learning algorithms to improve the accuracy and efficiency of thyroid disorder diagnosis, resulting in improved patient outcomes and healthcare delivery. The significance of this study lies in its ability to shed light on the potential of machine learning techniques to address the problem of accurate diagnosis and effective treatment of thyroid disorders, which is critical for improving healthcare outcomes and reducing the burden of the disease on patients and healthcare systems.

# References

1. Alyas T, Hamid M, Alissa K, Faiz T, Tabassum N, Ahmad A. Empirical Method for Thyroid Disease Classification Using a Machine Learning Approach. Biomed Res Int. 2022;2022:9809932. Published 2022 Jun 7. doi:10.1155/2022/9809932
2. Banu, Gulmohamed. (2016). Predicting Thyroid Disease using Linear Discriminant Analysis (LDA) Data Mining Technique. Communications on Applied Electronics. 4. 4-6. 10.5120/cae2016651990.
3. Beach, A. (2023, January 11). JANUARY IS THYROID AWARENESS MONTH - Delaney Radiology | Wilmington, NC. Delaney Radiology | Wilmington, NC. <https://www.delaneyrad.com/blog/january-is-thyroid-awareness-month/>
4. Chaganti R, Rustam F, De La Torre Díez I, Mazón JLV, Rodríguez CL, Ashraf I. Thyroid Disease Prediction Using Selective Features and Machine Learning Techniques. *Cancers*. 2022; 14(16):3914. https://doi.org/10.3390/cancers14163914
5. Charu C. Aggarwal. 2018. Neural Networks and Deep Learning: A Textbook (1st. ed.). Springer Publishing Company, Incorporated.
6. Dongyang Li, Dan Yang, Jing Zhang, Xuedong Zhang. AR-ANN: Incorporating Association Rule Mining in Artificial Neural Network for Thyroid Disease Knowledge Discovery and Diagnosis. IAENG International Journal of Computer Science, 47:1, IJCS\_47\_1\_04
7. Giuseppe Bonaccorso. 2017. Machine Learning Algorithms: A reference guide to popular algorithms for data science and machine learning. Packt Publishing.
8. Guleria, K., Sharma, S., Kumar, S., & Tiwari, S. (2022). Early prediction of hypothyroidism and multiclass classification using predictive machine learning and deep learning. Measurement: Sensors, 24, 100482. doi: 10.1016/j.measen.2022.100482
9. Hameed, M. (2017). Artificial Neural Network System for Thyroid Diagnosis. Journal Of University Of Babylon, 25(2). Retrieved from <https://www.iasj.net/iasj/article/121656>
10. Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). Deep Learning. MIT Press.
11. Islam SS, Haque MS, Miah MSU, Sarwar TB, Nugraha R. Application of machine learning algorithms to predict the thyroid disease risk: an experimental comparative study. PeerJ Comput Sci. 2022;8:e898. Published 2022 Mar 3. doi:10.7717/peerj-cs.898
12. K. Pavya and B. Srinivasan, "Feature selection algorithms to improve thyroid disease diagnosis," 2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT), Coimbatore, India, 2017, pp. 1-5, doi: 10.1109/IGEHT.2017.8094070.
13. Li, Z., Qin, J., Zhang, X., Wan, Y. (2019). A Hybrid Intelligent Framework for Thyroid Diagnosis. In: Ning, H. (eds) Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health. CyberDI CyberLife 2019 2019. Communications in Computer and Information Science, vol 1138. Springer, Singapore. https://doi.org/10.1007/978-981-15-1925-3\_32
14. Liaquat, I. (2020, May 16). Linear Regression — Dummy Variable Trap. Medium. https://medium.com/analytics-vidhya/linear-regression-dummy-variable-trap-8964a83516d9
15. M. Riajuliislam, K. Z. Rahim and A. Mahmud, "Prediction of Thyroid Disease (Hypothyroid) in Early Stage Using Feature Selection and Classification Techniques," *2021 International Conference on Information and Communication* Technology for Sustainable Development (ICICT4SD), Dhaka, Bangladesh, 2021, pp. 60-64, doi: 10.1109/ICICT4SD50815.2021.9397052.
16. May Sanjaya, I Made & Nugroho, Hanung Adi & Setiawan, Noor Akhmad. (2015). A Comparison of Classification Methods on Diagnosis of Thyroid Diseases. 10.1109/ISITIA.2015.7219959.
17. McKinney Wes. 2012. Python for data analysis (1st. ed.). O'Reilly Media, Inc.
18. Müller Andreas C & Guido S. (2017). Introduction to machine learning with python : a guide for data scientists (First). O'Reilly Media. Retrieved April 24 2023 from http://www.dawsonera.com/depp/reader/protected/external/AbstractView/S9781449369903.
19. Poorani, S. (2020). Supervised Machine Learning Models For Classification Of Thyroid Data . International Journal Of Scientific & Technology Research, 9(2), 1683-1685. Retrieved from https://www.ijstr.org/paper-references.php?ref=IJSTR-0120-29424
20. Radiation-induced hypothyroidism after treatment of head and neck cancer - PubMed. (2016, March 1). PubMed. <https://pubmed.ncbi.nlm.nih.gov/26931197/>
21. Salman, Khalid & Sonuç, Emrullah. (2021). Thyroid Disease Classification Using Machine Learning Algorithms. Journal of Physics: Conference Series. 1963. 012140. 10.1088/1742-6596/1963/1/012140.
22. Shahid, Afzal & Singh, Maheshwari & Raj, Rahul & Suman, Rashmi & Jawaid, Drakhshan & Alam, Muqtadir. (2019). A Study on Label TSH, T3, T4U, TT4, FTI in Hyperthyroidism and Hypothyroidism using Machine Learning Techniques. 930-933. 10.1109/ICCES45898.2019.9002284.
23. Shai Shalev-Shwartz and Shai Ben-David. 2014. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, USA.
24. Shivanee Pandey, Rohit Miri, S. R. Tandan, 2013, Diagnosis And Classification Of Hypothyroid Disease Using Data Mining Techniques, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 02, Issue 06 (June 2013),
25. Siji George C G and B.Sumathi, “Grid Search Tuning of Hyperparameters in Random Forest Classifier for Customer Feedback Sentiment Prediction” International Journal of Advanced Computer Science and Applications(IJACSA), 11(9), 2020. [http://dx.doi.org/10.14569/IJACSA.2020.0110920](https://dx.doi.org/10.14569/IJACSA.2020.0110920)
26. UCI Machine Learning Repository: Thyroid Disease Data Set. archive.ics.uci.edu/ml/datasets/thyroid+disease.
27. V. Bolón-Canedo, I. Porto-Díaz, N. Sánchez-Maroño, A. Alonso-Betanzos, A framework for cost-based feature selection, Pattern Recognition, Volume 47, Issue 7, 2014, Pages 2481-2489,ISSN 0031-3203, <https://doi.org/10.1016/j.patcog.2014.01.008>.
28. Wolny-Rokicka, E., Tukiendorf, A., Wydmański, J., Roszkowska, D., Staniul, B., & Zembroń-Łacny, A. (n.d.). Thyroid Function after Postoperative Radiation Therapy in Patients with Breast Cancer. PubMed Central (PMC). https://doi.org/10.22034/APJCP.2016.17.10.4577

# 指导教师意见

指导教师签名：