INTERDISCIPLINARY PROJECT REPORT

At

Sathyabama Institute of Science and Technology

(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

ALLAM ROHITH KUMAR
Reg.No. 40110068

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12 B Status
by UGC | Approved by AICTE
JEPPIAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ALLAM ROHITH KUMAR (Reg. No: 40110068)** who carried out the project entitled **"THYROID CLASSIFICATION"** under my supervision from January 2023 to April 2023.

INTERNAL GUIDE

Mr.MuraiDevakannaKamalesh.M.E.,Ph.D.

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

I, **ALLAM ROHITH KUMAR** hereby declare that the project report entitled **THYROID CLASSIFICATION** done by me under the guidance of **Mr.MurariDevakannanKamalesh.M.E.,Ph.D is** Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: **SIGNATURE OF THECANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks **to Dr. T. Sasikala M.E., Ph.D.,** Dean, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr.MurariDevakannanKamalesh.M.E.,Ph.D.**for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work report.

I wish to express my thanks to all Teaching and Non-teaching staff members of **t**he **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

This project is aimed at developing a Thyroid Classification.Thyroid disorder leading cause of medical diagnosis and prediction development, which medical science is a complicated axiom. The thyroid gland is one of our body's main organs. Thyroid hormone secretions are responsible for regulating metabolism. Hyperthyroidism and hypothyroidism are the two prominent thyroid disorders that produce thyroid hormones for the control of body metabolism. Machine learning is critical in the disease prediction process and in the study and classification models used for thyroid disease on the basis of data obtained from hospital datasets. A decent knowledge base must be ensured, built, and used as a hybrid model to solve dynamic learning tasks like medical diagnosis and prediction of tasks. Basic techniques of machine learning are used for the identification and inhibition of the thyroid. The data set is trained by using algorithms such as Random Forest Classifier, XG Boost, KNN Classifier, Logistic Regression. The Random Forest Classifier is used to predict the Thyroid of the patient. The dataset is trained by the algorithm to get the accuracy and data cleaning is done to improve the accuracy. If the patient has a risk of getting thyroid our system has to give suggestions like recommending Foods to eat and Foods to avoid, medication etc

With the vast amount of data and information difficult to deal with, especially in the health system, machine learning algorithms and data mining techniques have an important role in dealing with data. In our study, we used machine learning algorithms with thyroid disease. The goal of this study is to categorize thyroid disease into three categories: hyperthyroidism, hypothyroidism, and normal, so we worked on this study using data from Iraqi people, some of whom have an overactive thyroid gland and others who have hypothyroidism, so we used all of the algorithms. Support vector machines, random forest, decision tree, naïve bayes, logistic regression, k-nearest neighbors, multilayer perceptron (MLP), linear discriminant analysis. To classification of thyroid disease. Keywords: Machine learning, classification model, Thyroid diseases, Support vector machines, Random forest, Decision tree, Naïve bayes, logistic regression, K-nearest neighbors, Multi-layer perceptron (MLP), Linear discriminant analysis. Keywords: Machine learning, classification model, Thyroid diseases, Support vector machines, Random forest, Decision tree, Naïve bayes, logistic regression, K-nearest neighbors, Multi-layer perceptron (MLP), Linear discriminant analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Thyroid disease is a subset of endocrinology which is one of the most misunderstood and undiagnosed diseases . Thyroid gland diseases are among the most prevalent endocrine disorders in the world, second only to diabetes, according to the World Health Organization. Hyper function hyperthyroidism and hypothyroidism affect about 2% and 1% of individuals, respectively. Men have about a tenth of the prevalence of women. Hyper-and hypothyroidism may be caused by thyroid gland dysfunction, secondary to pituitary gland failure, or tertiary to hypothalamic malfunction. Due to dietary iodine deficiency, goiter or active thyroid nodules may become prevalent in some regions, with a prevalence of up to 15%. The thyroid gland can also be the location of different kinds of tumors and can be a dangerous place where endogenous antibodies wreak havoc (autoantibodies) . Early disease detection, diagnosis, and care, according to doctors, are vital in preventing disease progression and even death. For several different forms of anomalies, early identification and differential diagnosis raises the odds of good treatment. Despite multiple trials, clinical diagnosis is often thought to be a difficult task [4]. The thyroid gland is a butterfly-shaped gland situated at the base of the throat. It comprises two active thyroid hormones, levothyroxine (T4) and triiodothyronine (T3), which are involved in brain functions such as body temperature control, blood pressure management, and heart rate regulation. Likewise, thyroiddisease is one of the most prevalent diseases worldwide, and it is mostly caused by a deficiency of iodine, but it may also be caused by other factors. The thyroid gland is an endocrine gland that secretes hormones and passes them through the bloodstream. It is situated in the middle of the front of the body. Thyroid gland hormones are responsible for aiding in digestion as well as maintaining the body moist, balanced, and so on. Thyroid gland treatments such as T3 (triiodothyronine), T4 (thyroid hormone), and TSH (thyroid stimulating hormone) are used to assess thyroid activity (thyroid stimulating hormone). Thyroid disorder is classified into two types: hypothyroidism and hyperthyroidism. Data mining is a semi-automated method of looking for correlations in massive datasets. Machine learning algorithms are one of the best solutions to many problems that are difficult to solve [Classification is a data extraction technique (machine learning) used to predict and identify many diseases, such as thyroid disease, which we researched and

classified here because machine learning algorithms play a significant role in classifying thyroid disease and because these algorithms are high performing and efficient and aid in classification 7. Although the application of computer learning and artificial intelligence in medicine dates back to the early days of the field 8, there has been a new movement to consider the need for machine learning-driven healthcare solutions. As a result, analysts predict that machine learning will become commonplace in healthcare in the near future . Hyperthyroidism is a disorder in which the thyroid gland releases so many thyroid hormones. Hyperthyroidism is caused by an increase in thyroid hormone levels . Dry skin, elevated temperature sensitivity, hair thinning, weight loss, increased heart rate, high blood pressure, heavy sweating, neck enlargement, nervousness, menstrual cycles shortening, irregular stomach movements, and hands shaking are some of the signs . Hypothyroidism is a condition in which the thyroid gland is underactive Hypothyroidism is caused by a decline in thyroid hormone production. Hypo means deficient or less in medical terms. Inflammation and thyroid gland injury are the two primary causes of hypothyroidism. Obesity, low heart rate, increased temperature sensitivity, neck swelling, dry skin, hand numbness, hair issues, heavy menstrual cycles, and intestinal problems are some of the symptoms. If not treated, thesesymptoms can escalate over time

CHAPTER 2

LITERATURE REVIEW

Chandel, Khushboo [13] Thyroid disorder is classified using different classification models based on parameters such as TSH, T4U, and goiter in this study. Several grouping methods, such as K-nearest neighbor, are used to justify this argument. The Naive Bayes and support vector machines algorithms are employed. The experiment was carried out using the Rapid miner instrument, and the findings indicate that K-nearest neighbor is more effective than Naive Bayes in detecting thyroid disease. To diagnose thyroid disorder, the researchers used data mining classifiers. Thyroid disorder is a vital factor to consider when diagnosing a disease. KNN and Naive Bayes classifiers were used in this study. The Rapidminer tool is used to compare these two classifiers. The findings revealed that the K-nearest neighbor classifier is the most reliable, with a 93.44 percent accuracy, while the Naive Bayes classifier has a 22.56 percent accuracy. The proposed KNN technique improves classification accuracy, which contributes to improved results. As a result, Naive Bayes can only have a linear, elliptic, or parabolic decision boundary, so the decision boundary consistency of KNN is a huge plus. KNN outperforms most methods since the factors are interdependent. Banu, G. Rasitha [14] Thyroid disease is one of the most common illnesses that humans suffer from. The hypothyroid data used in this study came from the data repository at the University of California, Irvine (UCI). The platform Waikato Environment of Information Analysis will be used for the whole research project (WEKA). The J48 technique was found to be more effective than the decision stump tree technique. In the world of health care, disease diagnosis is a difficult challenge. In the decision-making method, a number of data mining methods are used. In this analysis, we used dimensionality reduction to pick a subset of attributes from the original results, and we used J48 and decision stump data mining classification techniques to define hypothyroidism. The uncertainty matrix is used to assess classifier output in terms of precision and error rate. The J48 Algorithm has 99.58 percent accuracy, which is higher than decision stump tree accuracy, and it also has a smaller error rate than Decision stump. Umar Sidiq, Dr, Syed Mutahar Aaqib, and Rafi Ahmad Khan [15] Classification, which is used to

characterize predefined data sets, is one of the most popular supervised learning data mining techniques. In the healthcare sector, the classification is commonly used to aid in medical decision-making, diagnosis, and administration. The information for this study was gathered from a well-known Kashmiri laboratory. The entire research project will be conducted on the ANACONDA3-5.2.0 platform. In an experimental analysis, classification methods such as k nearest neighbors, Support vector machine, Decision tree, and Nave bayes may be used. The Judgment Tree has the greatest accuracy of the other classes, at 98.89 percent. Sindhya, Mrs K [16] Thyroid disorder is a chronic illness that affects people all over the world. Data mining in healthcare is producing excellent results in the prediction of different diseases. The accuracy of data mining techniques for prediction is high, and the cost of prediction is low. Another significant benefit is that prediction takes very little time. In this study, I used classification algorithms to analyze thyroid data and came up with a result. A model's efficacy is primarily determined by two factors. The first is prediction precision, and the second is prediction time. According to our findings, Nave Bayes took just 0.04 seconds to forecast. However, it is less accurate than J48 and Random Forest. When we looked at prediction accuracy, the Random Forest model came in at 99.3 percent. However, the model's construction time is longer than the other two iterations. So we can assume that J48 is the best model for hypothyroid prediction since its accuracy is 99 percent, which is among the highest, and it takes 0.2 seconds to run, which is significantly less time than the Random Forest model. AKGÜL, Göksu, et al [17] The aim of this study is to propose a data mining-based method for enhancing the precision of hypothyroidism diagnosis by integrating patient questions with test results during the diagnosis process. Another goal is to reduce the risks that come with dialysis interventional trials. The logical conclusion It was determined if the new samples were hypothyroid using data from the UCI machine learning database, which included 3163 samples, 151 of which were hypothyroid and the others were hypothyroid. Different sampling techniques were used in the data collection to eradicate the unbalanced distribution, and models were developed to diagnose hypothyroidism using Logistic Regression, K Nearest Neighbor, and Support Vector Machine classifiers. The thesis demonstrated the impact of sampling techniques on the diagnosis of hypothyroidism in this regard. The Logistic Regression classifier produced the best results of all the models created. The precision was 97.8%, the F-Score was 82.26 percent, the region under the curve was 93.2 percent, and the Matthews correlation coefficient was 81.8 percent for this analysis,

which was trained on the data set using over-sampling techniques. VijiyaKumar, K., et al [18] The aim of this paper is to create a method that can predict diabetes in a patient early and accurately using the Random Forest algorithm in a machine learning technique. Random Forest algorithms are a type of ensemble learning system that is commonly used for classification and regression tasks. As compared to other algorithms, the performance ratio is higher. The suggested model gives the best outcomes for diabetic prediction, and the results revealed that the prediction system is capable of correctly, effectively, and most importantly, immediately forecasting diabetes disease. Chaurasia, Vikas, Saurabh Pal, and B. B. Tiwari [19] After all other cancers, breast cancer is the second most common cancer in women. The aim of this research paper is to provide a breast cancer study that incorporates cutting-edge techniques. Improving breast cancer survivability modeling models by incorporating recent research advances. We used a broad dataset and three common data mining algorithms (Nave Bayes, RBF Network, and J48) to construct prediction models (683 breast cancer cases). For accuracy comparison, we used 10-fold cross-validation approaches to measure the unbiased estimation of the three prediction models. The findings suggest that the Bay is a safe place to visit (based on an average precision Breast Cancer dataset). The RBF Network is the second-best predictor, with 93.41 percent accuracy on the holdout sample (better than any other prediction accuracy reported in the literature), and Nave Bayes is the third-best predictor, with 97.36 percent accuracy on the holdout sample (better than any other prediction accuracy reported in the literature) (better than any other prediction accuracy published in the literature). In this study, we evaluated three breast cancer survivability prediction models using two criteria: benign and malignant cancer cases. Begum, Amina, and A. Parkavi [20] The most recent research focuses on thyroid disease classification of two of the most frequent thyroid dysfunctions in the general population (hyperthyroidism and hypothyroidism). The researchers looked at and compared four different classification models: Naive Bayes, Decision Trees, Multilayer Perceptrons, and Radial Basis Function Networks. The findings reveal that all of the classification models listed above have a high degree of accuracy, with the Decision Tree model having the highest classification score. The classifier was built and validated using data from a Romanian data website and the UCI machine learning repository. KNIME Analytics Platform and Weka are two data sets. Data mining techniques were used as the foundation for developing and testing the classification models. A variety of studies in the field of thyroid

classification use various data mining techniques to construct robust classifiers, according to the literature. The authors of this research explored the use of four classification models on thyroid data (Nave Bayes, Decision Tree, MLP, and RBF Network) to help classify thyroid dysfunctions such as hyperthyroidism and hypothyroidism. In all of the **cases that were tested, the decision tree model was the correct classification model.**

Table 1: shows the literature review and the algorithms used and their accuracy.

Study number	Authors	Reference	year	Algorithms	Accuracy
1	Chandel, Khushboo	[13]	2016	KNN, Naive Bayes	KNN 93.44, Naive Bayes 22.56
2	Banu, G. Rasitha	[14]	2016	J48	J48 99.85
3	Umar Sidiq, Dr, Syed Mutahar Aaqib, and Rafi Ahmad Khan	[15]	2019	k nearest neighbors, Support vector machine, Decision tree, and Nave bayes	Nave bayes 98.89, SVM 96.30, KNN 98.89
4	Sindhya, Mrs K	[16]	2020	Nave bayes,J48 and Random Forest	J48 99 , Random Forest 99.3, Nave bayes 95
5	AKGÜL, Göksu, et al	[17]	2020	k nearest neighbors and SVM	k nearest neighbors 92, SVM 97.8
6	VijiyaKumar, K., et al	[18]	2019	Random Forest	the results revealed that the prediction system is capable of correctly
7	Chaurasia, Vikas, Saurabh Pal, and B. B. Tiwari	[19]	2018	Nave Bayes, RBF Network, and J48	J48 93.41, Nave Bayes 97.36, RBF Network 96.77
8	Begum, Amina, and A. Parkavi	[20]	2019	Nave Bayes, Decision Tree, MLP, and RBF Network	Nave Bayes 91.63, Decision Tree 96.91, MLP 95.15, and RBF Network 96.03

CHAPTER 3

METHODOLOGY

3.1. Data Collection

Machine learning algorithms are used in the rapid and early diagnosis of thyroid diseases and other diseases, as they now in a significant position in the health field and help us in diagnosing and classifying diseases for this reason we were able to collect a good amount of data on thyroid diseases and we are working in our study on the classification of diseases using this data The data that I used in our study is a set of data taken from external hospitals and laboratories specialized in analyzing and diagnosing diseases, and the sample taken from the data is the data of the Iraqi people and the type of data taken related to thyroid disease, where data were taken on 1250 people between males and females, and their ages range from 1 year to 1 year. 90 years as these samples contain people with thyroid disease who suffer from hyperthyroidism and hypothyroidism and normal people who do not suffer from thyroid disease.

The data were collected over a period of one to four months, and the main goal of collecting the data was to classify thyroid diseases using machine learning algorithms. These data include gender, age, analysis of T3 (triiodothyronine), T4 (thyroid hormone), TSH (thyroid stimulating hormone), and a host of other characteristics. As the data obtained consist of 17 variables or attributes where all the attributes were taken in our study which consist of(id, age, gender, query thyroxine, on_antithyroid_medication,sick,pregnant,thyroid_surgery,query_hypothyorid,query_hyperthyroid, TSH_M, TSH, T3_M, T3, T3, T4, Category).

Table 2: shows the features contained in the dataset.

No	Attribute Name	Value Type	Clarification
1	id	number	1,2,3…….,
12	age	number	1,10,20,50,…….,
3	gender	1,0	1=m,0=f
4	query_thyroxine	1,0	1=yes,0=no
5	on_antithyroid_medication	1,0	1=yes,0=no
6	sick	1,0	1=yes,0=no
7	pregnant	1,0	1=yes,0=no
8	thyroid_surgery	1,0	1=yes,0=no
9	query_hypothyorid	1,0	1=yes,0=no
10	query_hyperthyroid	1,0	1=yes,0=no
11	TSH measured	1,0	1=yes,0=no
12	TSH	Analysis ratio	Numeric value
13	T3 measured	1,0	1=yes,0=no
14	T3	Analysis ratio	Numeric value
15	T4 measured	1,0	1=yes,0=no
16	T4	Analysis ratio	Numeric value
17	category	0,1,2	0=normal,1=hypothyroid,2=hyperthyroid

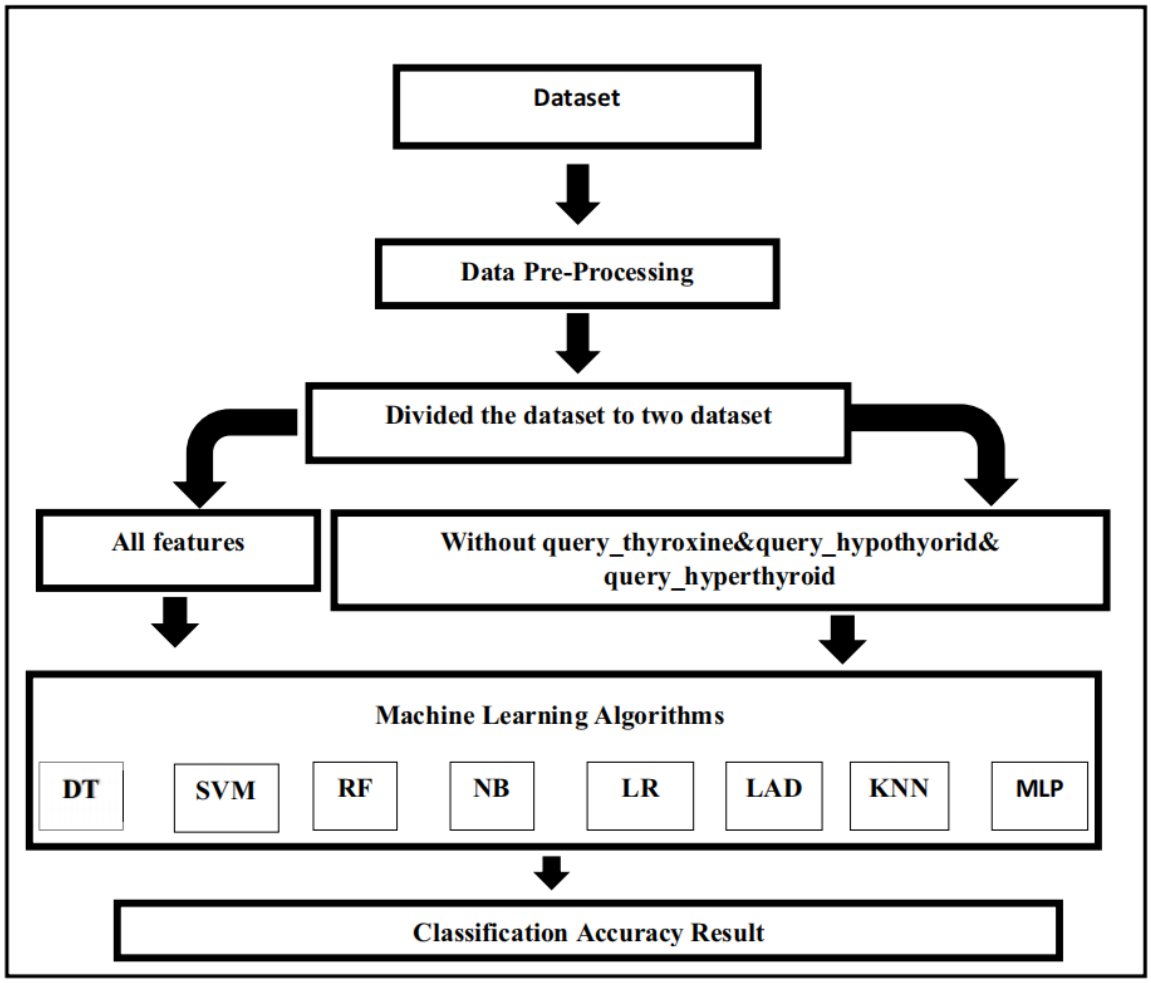


Figure 1. Shows how data is entered and the operations that take place .

3.2. Data Preprocessing

The process of pre-processing the data is very important and it is a major step in data mining, as it has a good effect on the data, as the pre-processing process is used to reveal the data through analyzing the data and discovering the lost data, as it examines the data with great care. The pre-processing process includes cleaning the data, preparing the data, etc. In this stage or step we did is to clean and arrange the data that we were able to obtain, where we identified a set of missing data in this data where the missing features were identified, and among these properties that were missing T4 by number 151 and T3 by number 112, where we were able to Processing this lost data by replacing it with the value of the mediator, and

after working in this way we were able to obtain the data in a good and better way and free from lost data, as the data became arranged and good and free from any defect or problem so that we can work on it smoothly and well. We also used normalization technical with the MLP algorithm.

3.3. Data Machine Learning Techniques

The key aim of using machine learning algorithms is to differentiate between three forms of thyroid disease. The first is hyperthyroidism, the second is hypothyroidism, and the third is stable patients who do not have any thyroid issues.

3.3.1.Support Vector Machines

The support vector machine (SVM) is a machine learning and data mining algorithm to determine the strongest predictors of this variable for energy consumption. The research used popular classification methods to answer our question: best subset selection, boosting trees, and generalized additive models. Our first approach was to use forward, backward and best subset selection to obtain a subset of predictors that most strongly predicted consumption with a linear relationship. The SVM provided an approach that was to use a tree-based method to stratify the predictor space into sample regions using recursive binary splitting. The research decided to use the boosting tree method as this is known to be one of the most powerful tree based models. SVM also has a good ability to deal with high dimensionality data.

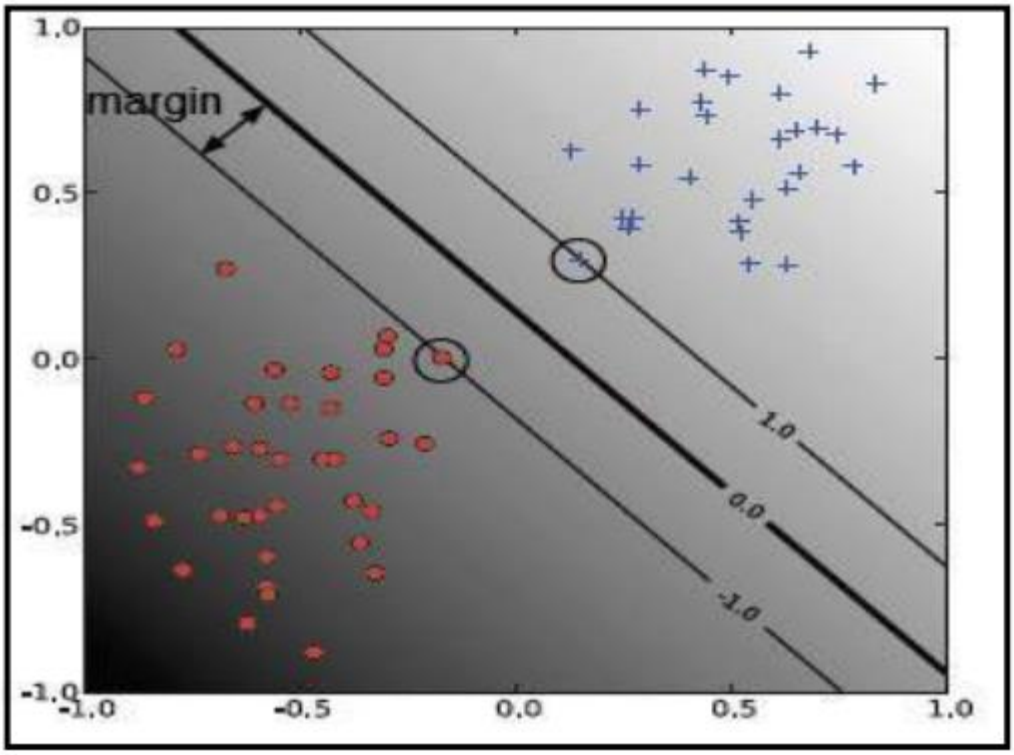


Figure 2. Example a two-class linear SVM classifier [21].

3.3.2. Random forest

The random forest computes the mean response of every predictor for energy consumption. Then, for each sample, a random forest adds the absolute distance each response was from the mean of each predictor for a total sum of the distance that each answer was from the means of the data. A high distance value will signify individuals who were consistently far away from the mean response in each sample. Detecting rates who repeatedly classify the samples was simple--a function that calculated the mode of each response was used. If the mode of a response was over 90% of the total number of questions, the research marked the response as potentially high in energy consumption. There are many responses marked. It was clear from a visual examination of these responses that the individuals had sampled with the same response.

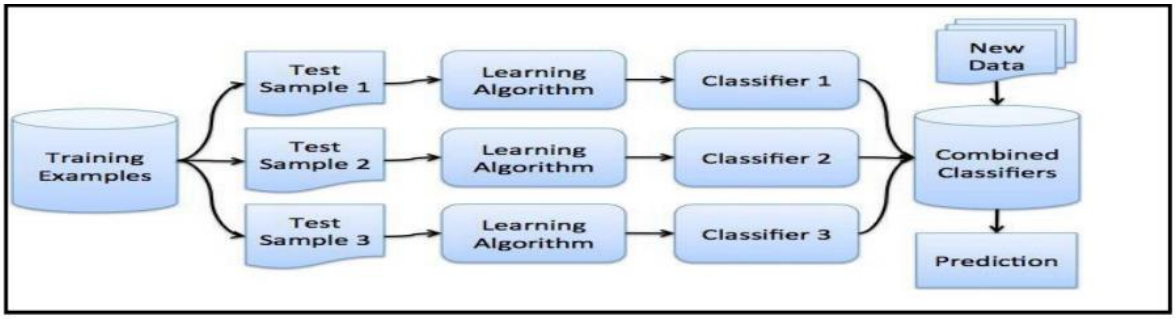


Figure 3. Phases of ensemble random forest approaches to solve classification problems [22].

3.3.3.Decision Tree

The decision tree method is based on a decision-boosting machine which is analyzed for predicting the energy consumption factor to try a tree based approach to determine the most significant predictors of consumption. To do so, used the decision tree approach as provided. The decision entails fitting thousands of trees, each of which is grown using information from the previous tree, in order for the model to improve over time. There are a few tuning parameters, including: number of trees, shrinkage parameter, number of splits in each tree.

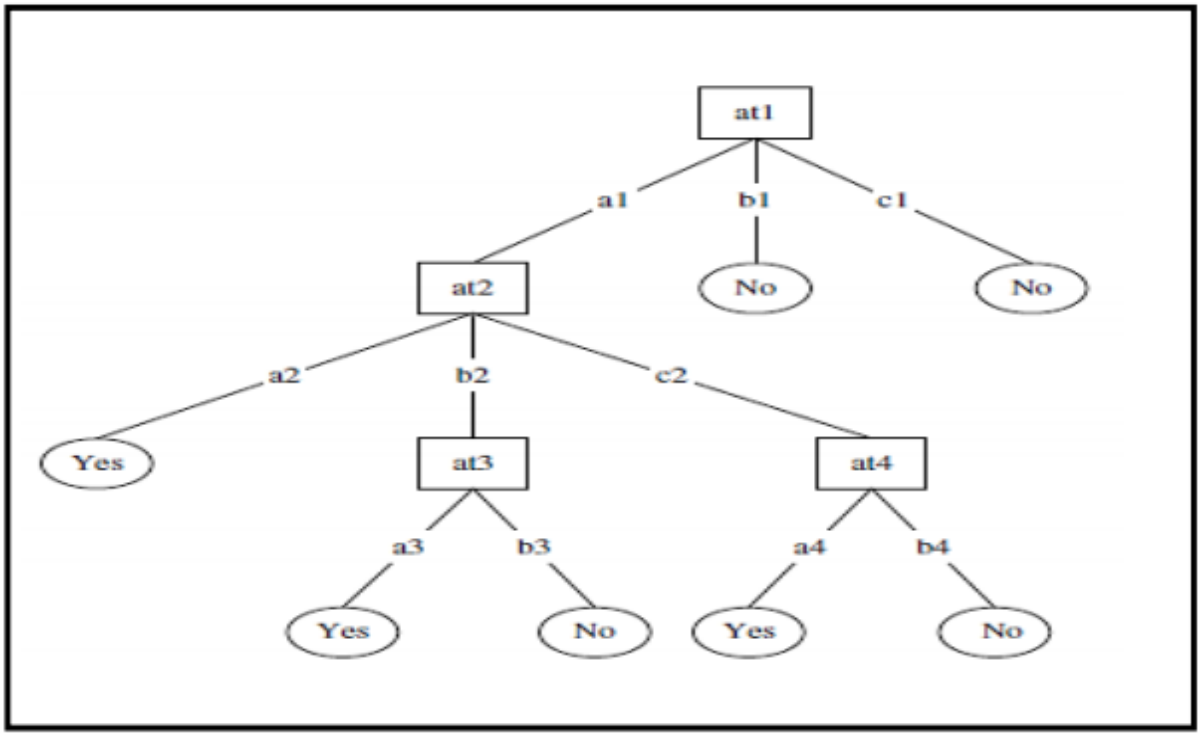


Figure 4. Decision tree algorithm structure [23].

3.3.4.Naïve Bayes

Naïve Bayesian are able to compare multiple generalized additive models featuring the output variables of subset selection, variables with the highest relative influence in the classifying and a mix of variables from both. It compared the prediction accuracy of each best model to directly compare them. By fitting naïve Bayes with various combinations of splines, 2nd degree polynomials and linear predictor variables, it narrowed down the relationships between single predictors and the response. More polynomials and splines were applied to predictors that were proved to have nonlinear relationships with our response variable.

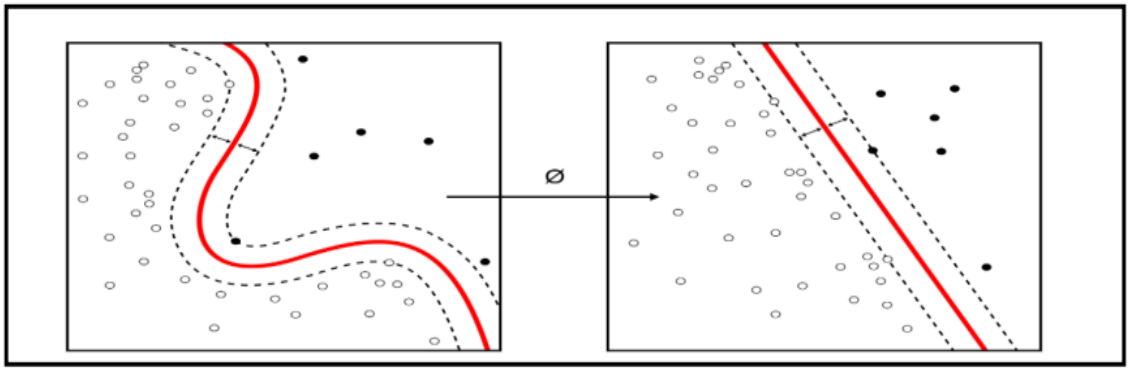


Figure 5. Naïve Bayes algorithm on left with respect to support vector machine on right side for classification structure [24].

3.3.5.Logistic Regression

Under the Supervised Learning technique, one of the most common Machine Learning algorithms is logistic regression. It's a method for estimating a categorical dependent variable from a number of independent variables. A categorical dependent variable's contribution is predicted using logistic regression. As a result, the result must be a singular or categorical value. It may be Yes or No, 0 or 1, true or false, and so on, but instead of providing exact values like 0 and 1, it gives probabilistic values that are somewhere between 0 and 1. Except for how they are used, Logistic Regression is somewhat similar to Linear Regression. Regression problems are solved using Linear Regression, although the classification problems are solved using logistic regression .

3.3.6.k-Nearest neighbors

The k-nearest neighbor algorithm differs from the other methods in that it

uses the data directly for classification rather than first building a model [26]. As a result, no special model construction is necessary, and the only variable in the model is k, the number of nearest neighbors to use in class membership estimation: the value of p(y/x) is simply the ratio of members of classy among the k nearest neighbors of x. Changing the value of k will make the model more or less stable (small or big values of k, respectively). The advantage of k-nearest neighbors over other algorithms is its ease of use. Neighbors can provide a rationale for the classification result; in cases where black-box models are incomplete, this case-based reasoning can be helpful. The key disadvantage of k-nearest neighbors is the calculation of the case neighborhood, which requires defining a metric that measures the distance between data objects.

3.3.7.Multi-Layer Perceptron's (MLP)

A multilayer perceptron is a feedforward artificial neural network that generates a series of outputs from a set of inputs (MLP). In an MLP, several layers of input nodes form a directed graph between the input and output layers. MLP employs backpropogation to train the network. MLP is a deep learning technique. A multilayer perceptron is a neural network that connects several layers in a guided graph, meaning that the signal only travels in one direction between nodes. Each node, with the exception of the input nodes, has a nonlinear activation function. An MLP uses backpropagation as a supervised learning process. MLP is a deep learning method that employs several layers of neurons. MLP is a commonly used system in supervised learning problems, computational biology, and parallel distributed processing analysis. Applications include speech recognition, image recognition, and automatic translation .

3.3.8.Linear Discriminant Analysis

It is one of the most commonly used dimensionality reduction techniques. It's used in pattern recognition systems like machine learning and other systems. The aim of LDA is to project elements from a highdimensional space into a lower-dimensional space. This is achieved in order to avoid common dimensionality issues while still lowering spatial costs and capital.

Linear discriminant analysis, a supervised classification method, is used to construct machine learning models. These dimensionality reduction models are used in a number of applications, such as ad prediction and image recognition [29].

CHAPTER 4

RESULTS:

We have applied our data to a range of machine learning algorithms (Decision Tree, SVM, Random Forest, Naive Bayes, Logistic Regression, Linear Discriminant Analysis, k-Nearest neighbors, Multi-Layer Perceptron) We divided the existing data into two parts, 30% for training and 70% for testing as this training is the first training on this data. In the first step we took all the properties in our data and applied them to a group of algorithms shown in the table below, and after the application process these results appeared to us. This practical part has been implemented on the python platform and is considered a complete and integrated platform. All attributes have been taken which are 16 inputs and one output.

Table 3. Evaluation measurements for classification models with all attribute of dataset

NO	Algorithms	Accuracy
1	Decision Tree	90.13
2	SVM	92.53
3	Random Forest	91.2
4	Naive Bayes	90.67
5	Logistic Regression	91.73
6	Linear Discriminant Analysis	83.2
7	KNeighbors Classifier	91.47
8	MLP	96.4

And as shown to us in this table, it shows us the accuracy of each algorithm, as it received an algorithm Decision Tree 98.4 accuracy SVM 92.27 accuracy Random Forest 98.93 accuracy Naive Bayes 81.33 accuracy Logistic Regression 91.47 accuracy Linear Discriminant Analysis 83.2 accuracy KNeighbors Classifier 90.93 accuracy and MLP(NN) 97.6 accuracy and through these results, this logic Random Forest algorithm has obtained high accuracyThen an algorithm followsDecision Tree. Most of the algorithms that I used to classify thyroid disease have proven their worth in diagnosing the disease, and this will help us a lot in the health system, as it will be an aid to the health sectors. In the second step, we removed 3 traits, based on a previous study Ioniţă, Irina, and Liviu Ioniţă [30] The deleted attributes were both query_thyroxine&query_hypothyorid& query_hyperthyroid. After deleting these attributes, we applied our data also to the algorithm group, and also by using the Python script, we were able to obtain these results listed below in Table (4) .

Table 4. Evaluation measures of the classification models without three attributes of the data set.

NO	Algorithms	Accuracy
1	Decision Tree	98.4
2	SVM	92.27
3	Random Forest	98.93
4	Naive Bayes	81.33
5	Logistic Regression	91.47
6	Linear Discriminant Analysis	83.2
7	KNeighbors Classifier	90.93
8	MLP	97.6

As it seems to us that the Naive Bayes algorithm has a high accuracy of 90.67 after the three traits have been omitted, the SVM algorithm, the logistic regression algorithm and the KNeighbours Classifier algorithm have increased slightly and reduced the accuracy of the other algorithms. We show here that the accuracy of the algorithms used on our data changes with the change of the characteristics used in the data, as experience has demonstrated this clear change, which obtained the accuracy of the algorithms when three of the characteristics were deleted, as the accuracy of some algorithms decreased and some of them increased.

SOURCE CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("Data.csv")
data.head()
data.shape
data.info()
data.isnull().sum()
data.drop(['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_me
asured','TBG_measured','referral_source','patient_id'], axis=1, inplace = True)
data.head()
data['target']
diagnoses = {'A': 'hyperthyroid conditions',
             'B': 'hyperthyroid conditions',
             'C': 'hyperthyroid conditions',
             'D': 'hyperthyroid conditions',
             'E': 'hypothyroid conditions',
             'F': 'hypothyroid conditions',
             'G': 'hypothyroid conditions',
             'H': 'hypothyroid conditions',
```

```
            'I': 'binding protein',
            'J': 'binding protein',
            'K': 'general health',
            'L': 'replacement therapy',
            'M': 'replacement therapy',
            'N': 'replacement therapy',
            'O': 'antithyroid treatment',
            'P': 'antithyroid treatment',
            'Q': 'antithyroid treatment',
            'R': 'miscellaneous',
            'S': 'miscellaneous',
            'T': 'miscellaneous'}
data['target'] = data['target'].map(diagnoses)
data
data.isnull().sum()
data.dropna(subset=['target'],inplace=True)
data['target'].value_counts()
data.head()
data.describe()
data[data.age>100]
data['age']=np.where((data.age>100), np.nan, data.age)
data
x=data.iloc[:,0:-1]
y= data.iloc[:,-1]
data.isnull().sum()
x['sex'].unique()
x['sex'].replace(np.nan, 'F', inplace=True)
x['sex'].value_counts()
x.isnull().sum()
data.info()
x['age']=x['age'].astype('float')
x['TSH']=x['TSH'].astype('float')
x['T3']=x['T3'].astype('float')
x['TT4']=x['TT4'].astype('float')
x['T4U']=x['T4U'].astype('float')
x['FTI']=x['FTI'].astype('float')
x['TBG']=x['TBG'].astype('float')
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
ordinal_encoder = OrdinalEncoder(dtype = 'int64')
x.iloc[:, 1:16] = ordinal_encoder.fit_transform(x.iloc[:, 1:16])
x.head()
x.replace(np.nan, '0', inplace=True)
x.head()
label_encoder = LabelEncoder()
y_dt= label_encoder.fit_transform(y)
y=pd.DataFrame(y_dt, columns=['target'])
y
```

```
y.value_counts(normalize=True)
import seaborn as sns
corrmat = x.corr()

f, ax = plt.subplots(figsize =(9, 8))
sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
y_train.value_counts()
from imblearn.over_sampling import SMOTE
os = SMOTE(random_state=0,k_neighbors=1)
x_bal,y_bal=os.fit_resample(x_train,y_train)
x_test_bal,y_test_bal=os.fit_resample(x_test,y_test)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_bal = sc.fit_transform(x_bal)
x_test_bal= sc.transform(x_test_bal)
x_bal
x_test_bal
y_bal.value_counts()
columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','
pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid','l
ithium', 'goitre', 'tumor','hypopituitary','psych','TSH','T3','TT4','T4U','FTI','TBG']
x_test_bal= pd.DataFrame(x_test_bal,columns=columns)
x_bal= pd.DataFrame(x_bal,columns=columns)
x_bal
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
rfr = RandomForestClassifier().fit(x_bal,y_bal)
y_pred = rfr.predict(x_test_bal)
accuracy_score(y_test_bal,y_pred)
x_bal.shape,y_bal.shape,x_test_bal.shape,y_test_bal.shape
test_score=accuracy_score(y_test_bal,y_pred)
test_score
train_score = accuracy_score(y_bal,rfr.predict(x_bal))
train_score
from sklearn.inspection import permutation_importance
results = permutation_importance(rfr,x_bal,y_bal, scoring='accuracy')
feature_importance=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_m
eds','sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyp
erthyroid','lithium',                                                                    'goitre',
'tumor','hypopituitary','psych','TSH','T3','TT4','T4U','FTI','TBG']
importance = results.importances_mean
importance = np.sort(importance)
for i,v in enumerate(importance):
    i=feature_importance[i]
    print('feature: {:<20} Score: {}'. format(i,v))
```

```
plt.figure(figsize=(10,10))
plt.bar(x=feature_importance, height = importance)
plt.xticks(rotation=30, ha='right')
plt.show()
x_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick',
'pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid','
lithium'],axis=1,inplace=True)
x_test_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','
sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthy
roid','lithium'],axis=1,inplace=True)
x_bal.head()
x_test_bal.head()
rfr1 = RandomForestClassifier()
rfr1.fit(x_bal,y_bal)
y_pred=rfr1.predict(x_test_bal)
print(classification_report(y_test_bal,y_pred))
train_score = accuracy_score(y_bal,rfr1.predict(x_bal))
train_score
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(x_bal,y_bal)
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(x_bal,y_bal)
print(classification_report(y_test_bal,y_pred))
train_score = accuracy_score(y_bal, xgb.predict(x_bal))
train_score

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

sv= SVC()
sv.fit(x_bal,y_bal)
y_pred = sv.predict(x_test_bal)
print(classification_report(y_test_bal,y_pred))
train_score=accuracy_score(y_bal,sv.predict(x_bal))
train_score
params={
   'n_estimators': [100, 200, 500],
    'criterion': ['gini', 'entropy'],
    'max_depth' : [x for x in range(1,20)]
}
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(rfr1, params, scoring='accuracy',cv=5,n_jobs=-1)
grid_search.fit(x_bal,y_bal)
grid_search.best_params_
rfr_gs=RandomForestClassifier(criterion=   'entropy',   max_depth=16,   n_estimators=
```

```
200
rfr_gs.fit(x_bal, y_bal)
y_pred=rfr_gs.predict(x_test_bal)
print(classification_report(y_test_bal,y_pred))
train_score= accuracy_score(y_bal,rfr_gs.predict(x_bal))
train_score
params={
    'n_estimators': [100, 200, 500],
    'learning_rate': [0.01,0.05,0.1],
    'booster': ['gbtree', 'gblinear'],
    'gamma': [0, 0.5, 1],



}
grid_xgb = GridSearchCV(xgb, params, scoring='accuracy',cv=3,n_jobs=-1
grid_xgb.fit(x_bal,y_bal)
grid_xgb.best_params_
xgb1=XGBClassifier(booster='gbtree', gamma= 0, learning_rate= 0.1, n_estimators=
500)
xgb1.fit(x_bal,y_bal)
y_pred= xgb1.predict(x_test_bal)
print(classification_report(y_test_bal,y_pred))
train_score= accuracy_score(y_bal,xgb1.predict(x_bal))
train_score
params = {
    'C' : [0.01, 0.1, 1, 10,100, 1000],
    'kernel': ['rbf', 'linear'],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
}
grid_svc = GridSearchCV(sv,params, scoring='accuracy',cv=5,n_jobs=-1)
grid_svc.fit(x_bal,y_bal)
grid_svc.best_params_
sv1=SVC(C =1000,gamma=1, kernel= 'rbf')
sv1.fit(x_bal,y_bal)
y_pred= sv1.predict(x_test_bal)
print(classification_report(y_test_bal,y_pred))
train_score= accuracy_score(y_bal,sv1.predict(x_bal))
train_score
import pickle
pickle.dump(xgb1,open('thyroid_1_model.pkl','wb'))
features = np.array([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])
print(label_encoder.inverse_transform(xgb1.predict(features)))
type(features)
pickle.dump(label_encoder,open('label_encoder.pkl','wb'))
data['target'].unique()
y['target'].unique()
```

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```python
data = pd.read_csv("Data.csv")
```

In [3]:
```python
data.head()
```

Out[3]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
0	29	F	f	f	f	f	f	f	f	t	...
1	29	F	f	f	f	f	f	f	f	f	...
2	41	F	f	f	f	f	f	f	f	f	...
3	36	F	f	f	f	f	f	f	f	f	...
4	32	F	f	f	f	f	f	f	f	f	...

5 rows × 31 columns

In [4]:
```python
data.shape
```

Out[4]:
```
(9172, 31)
```

In [6]:
```python
data.info()
```

Out[6]:
```
age                      0
sex                    307
on_thyroxine             0
query_on_thyroxine       0
on_antithyroid_meds      0
sick                     0
pregnant                 0
thyroid_surgery          0
I131_treatment           0
query_hypothyroid        0
query_hyperthyroid       0
lithium                  0
goitre                   0
tumor                    0
hypopituitary            0
psych                    0
TSH_measured             0
TSH                    842
T3_measured              0
T3                    2604
TT4_measured             0
TT4                    442
T4U_measured             0
T4U                    809
FTI_measured             0
FTI                    802
TBG_measured             0
TBG                   8823
referral_source          0
target                   0
patient_id               0
dtype: int64
```

In [5]:
```python
data.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9172 entries, 0 to 9171
Data columns (total 31 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   age                    9172 non-null   int64
 1   sex                    8865 non-null   object
 2   on_thyroxine           9172 non-null   object
 3   query_on_thyroxine     9172 non-null   object
 4   on_antithyroid_meds    9172 non-null   object
 5   sick                   9172 non-null   object
 6   pregnant               9172 non-null   object
 7   thyroid_surgery        9172 non-null   object
 8   I131_treatment         9172 non-null   object
 9   query_hypothyroid      9172 non-null   object
 10  query_hyperthyroid     9172 non-null   object
 11  lithium                9172 non-null   object
 12  goitre                 9172 non-null   object
 13  tumor                  9172 non-null   object
 14  hypopituitary          9172 non-null   object
 15  psych                  9172 non-null   object
 16  TSH_measured           9172 non-null   object
 17  TSH                    8330 non-null   float64
 18  T3_measured            9172 non-null   object
 19  T3                     6568 non-null   float64
 20  TT4_measured           9172 non-null   object
 21  TT4                    8730 non-null   float64
 22  T4U_measured           9172 non-null   object
 23  T4U                    8363 non-null   float64
 24  FTI_measured           9172 non-null   object
 25  FTI                    8370 non-null   float64
 26  TBG_measured           9172 non-null   object
 27  TBG                    349 non-null    float64
 28  referral_source        9172 non-null   object
 29  target                 9172 non-null   object
 30  patient_id             9172 non-null   int64
dtypes: float64(6), int64(2), object(23)
memory usage: 2.2+ MB
```

In [7]:

```
#Removing Redundant attributes from dataset
#The columns listed below were removed because of redundacy.
#They are boolean and state whether or not a value has been recorded for their respective blood tests.
#TSH_measured
#T3_measured
#TT4_measured
#T4U_measured
#FTI_measured
#TBG_measured
data.drop(['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG_measured','referral_s
```

In [8]:

```
data.head()
```

Out[8]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
0	29	F	f	f	f	f	f	f	f	t	...
1	29	F	f	f	f	f	f	f	f	f	...
2	41	F	f	f	f	f	f	f	f	f	...
3	36	F	f	f	f	f	f	f	f	f	...
4	32	F	f	f	f	f	f	f	f	f	...

5 rows × 23 columns

In [9]:

```
data['target']
```

Out[9]:

```
0       -
1       -
2       -
3       -
4       S
       ..
9167    -
9168    -
9169    I
9170    -
9171    -
Name: target, Length: 9172, dtype: object
```

In [10]:

```
#re-mapping target values to diagnostic group
diagnoses = {'A': 'hyperthyroid conditions',
             'B': 'hyperthyroid conditions',
             'C': 'hyperthyroid conditions',
             'D': 'hyperthyroid conditions',
             'E': 'hypothyroid conditions',
             'F': 'hypothyroid conditions',
```

```
            'G': 'hypothyroid conditions',
            'H': 'hypothyroid conditions',
            'I': 'binding protein',
            'J': 'binding protein',
            'K': 'general health',
            'L': 'replacement therapy',
            'M': 'replacement therapy',
            'N': 'replacement therapy',
            'O': 'antithyroid treatment',
            'P': 'antithyroid treatment',
            'Q': 'antithyroid treatment',
            'R': 'miscellaneous',
            'S': 'miscellaneous',
            'T': 'miscellaneous'}
data['target'] = data['target'].map(diagnoses) #remapping
```

In [11]:
```
data
```

Out[11]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid
0	29	F	f	f	f	f	f	f	f	t
1	29	F	f	f	f	f	f	f	f	f
2	41	F	f	f	f	f	f	f	f	f
3	36	F	f	f	f	f	f	f	f	f
4	32	F	f	f	f	f	f	f	f	f
...	...	...	...	...	...	...	...	...	...	...
9167	56	M	f	f	f	f	f	f	f	f
9168	22	M	f	f	f	f	f	f	f	f
9169	69	M	f	f	f	f	f	f	f	f
9170	47	F	f	f	f	f	f	f	f	f
9171	31	M	f	f	f	f	f	f	f	t

9172 rows × 23 columns

In [12]:
```
data.isnull().sum()
```

Out[12]:
```
age                      0
sex                    387
on_thyroxine             0
query_on_thyroxine       0
on_antithyroid_meds      0
sick                     0
pregnant                 0
thyroid_surgery          0
I131_treatment           0
query_hypothyroid        0
query_hyperthyroid       0
lithium                  0
goitre                   0
tumor                    0
hypopituitary            0
psych                    0
TSH                    842
T3                    2604
TT4                    442
T4U                    809
FTI                    882
TBG                   8823
target                6935
dtype: int64
```

In [13]:
```
data.dropna(subset=['target'],inplace=True)
```

In [14]:
```
data['target'].value_counts()
```

Out[14]:
```
hypothyroid conditions     593
general health             436
binding protein            376
replacement therapy        336
miscellaneous              281
hyperthyroid conditions    182
antithyroid treatment       33
Name: target, dtype: int64
```

In [15]:
```
data.head()
```

Out[15]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
4	32	F	f	f	f	f	f	f	f	f	...
18	63	F	t	f	f	t	f	f	f	f	...
32	41	M	f	f	f	f	f	f	f	f	...
33	71	F	t	f	f	f	f	f	f	f	...
39	55	F	t	f	f	f	f	f	f	t	...

5 rows × 23 columns

In [16]:
```
data.describe()
```

Out[16]:

	age	TSH	T3	TT4	T4U	FTI	TBG
count	2237.000000	2087.000000	1643.000000	2140.000000	2059.000000	2080.000000	98.000000
mean	52.782579	14.930791	1.961875	116.390495	1.013439	120.363389	47.717347
std	19.677450	46.204092	1.452238	60.351600	0.280222	70.996728	32.398750
min	1.000000	0.005000	0.050000	2.000000	0.170000	1.400000	9.299999
25%	36.000000	0.255000	1.000000	76.000000	0.850000	83.000000	32.000000
50%	56.000000	2.000000	1.700000	109.000000	0.980000	109.000000	36.000000
75%	69.000000	8.799999	2.500000	156.000000	1.120000	157.000000	46.750000
max	95.000000	530.000000	18.000000	600.000000	2.330000	881.000000	200.000000

In [17]:
```
#Checking whether the age above 100
data[data.age>100]
```

Out[17]:

age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	t

0 rows × 23 columns

In [18]:
```
#changing age of observation with(age>100) to null
data['age']=np.where((data.age>100), np.nan, data.age)
```

In [19]:
```
data
```

Out[19]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid
4	32.0	F	f	f	f	f	f	f	f	f
18	63.0	F	t	f	f	t	f	f	f	f
32	41.0	M	f	f	f	f	f	f	f	f
33	71.0	F	t	f	f	f	f	f	f	f
39	55.0	F	t	f	f	f	f	f	f	t
...	...	...	...	...	...	...	...	...	...	...
3153	64.0	M	f	f	f	f	f	f	f	f
3157	60.0	M	f	f	t	f	f	f	f	f
3158	64.0	M	f	f	f	f	f	f	f	t
3162	36.0	F	f	f	f	f	f	f	f	f
3169	69.0	M	f	f	f	f	f	f	f	f

2237 rows × 23 columns

#splitting the data values as x and y¶

In [20]:
```
#splitting the data values as x and y
x=data.iloc[:,8:-1]
y= data.iloc[:,-1]
```

In [21]:
```
data.isnull().sum()
```

Out[21]:
```
age                       0
sex                      98
on_thyroxine              0
query_on_thyroxine        0
on_antithyroid_meds       0
sick                      0
pregnant                  0
thyroid_surgery           0
I131_treatment            0
query_hypothyroid         0
query_hyperthyroid        0
lithium                   0
goitre                    0
tumor                     0
hypopituitary             0
psych                     0
TSH                     150
T3                      594
TT4                      97
T4U                     178
FTI                     177
TBG                    2139
target                    0
dtype: int64
```

In [22]:
```
x['sex'].unique()
```

Out[22]:
```
array(['F', 'M', nan], dtype=object)
```

In [23]:
```
x['sex'].replace(np.nan, 'F', inplace=True)
```

In [24]:
```
x['sex'].value_counts()
```

Out[24]:
```
F    1701
M     536
Name: sex, dtype: int64
```

In [25]:
```
x.isnull().sum()
```

Out[25]:
```
age                       0
sex                       0
on_thyroxine              0
query_on_thyroxine        0
on_antithyroid_meds       0
sick                      0
pregnant                  0
thyroid_surgery           0
I131_treatment            0
query_hypothyroid         0
query_hyperthyroid        0
lithium                   0
goitre                    0
tumor                     0
hypopituitary             0
psych                     0
TSH                     150
T3                      594
TT4                      97
T4U                     178
FTI                     177
TBG                    2139
dtype: int64
```

In [26]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2237 entries, 4 to 9169
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  2237 non-null   float64
 1   sex                  2147 non-null   object
 2   on_thyroxine         2237 non-null   object
 3   query_on_thyroxine   2237 non-null   object
 4   on_antithyroid_meds  2237 non-null   object
 5   sick                 2237 non-null   object
 6   pregnant             2237 non-null   object
 7   thyroid_surgery      2237 non-null   object
 8   I131_treatment       2237 non-null   object
 9   query_hypothyroid    2237 non-null   object
 10  query_hyperthyroid   2237 non-null   object
 11  lithium              2237 non-null   object
 12  goitre               2237 non-null   object
 13  tumor                2237 non-null   object
 14  hypopituitary        2237 non-null   object
 15  psych                2237 non-null   object
 16  TSH                  2087 non-null   float64
 17  T3                   1643 non-null   float64
 18  TT4                  2140 non-null   float64
 19  T4U                  2059 non-null   float64
 20  FTI                  2060 non-null   float64
 21  TBG                  98 non-null     float64
 22  target               2237 non-null   object
dtypes: float64(7), object(16)
memory usage: 419.4+ KB
```

In [27]:
```
x['age']=x['age'].astype('float')
x['TSH']=x['TSH'].astype('float')
x['T3']=x['T3'].astype('float')
x['TT4']=x['TT4'].astype('float')
x['T4U']=x['T4U'].astype('float')
x['FTI']=x['FTI'].astype('float')
x['TBG']=x['TBG'].astype('float')
```

converting categorical to numerical values

In [28]:
```
#applying ordinal encoding to x values
#Encoding the categorical data
#Encoding the independent(output) variable
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
#categorical data


ordinal_encoder = OrdinalEncoder(dtype = 'int64')
x.iloc[:, 1:16] = ordinal_encoder.fit_transform(x.iloc[:, 1:16])
#ordinal_encoder.fit_transform(x[['sex']])
```

```
C:\Users\Sree\AppData\Local\Temp\ipykernel_9524\277053971.py:9: DeprecationWarning: In a future version, `df.il
oc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the
old behavior, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  x.iloc[:, 1:16] = ordinal_encoder.fit_transform(x.iloc[:, 1:16])
```

In [29]:
```
x.head()
```

Out[29]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
4	32.0	0	0	0	0	0	0	0	0	0	...
18	63.0	0	1	0	0	1	0	0	0	0	...
32	41.0	1	0	0	0	0	0	0	0	0	...
33	71.0	0	1	0	0	0	0	0	0	0	...
39	55.0	0	1	0	0	0	0	0	0	1	...

5 rows × 22 columns

In [30]:
```
x.replace(np.nan, '0', inplace=True)
x.head()
```

Out[30]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
4	32.0	0	0	0	0	0	0	0	0	0	...
18	63.0	0	1	0	0	1	0	0	0	0	...
32	41.0	1	0	0	0	0	0	0	0	0	...
33	71.0	0	1	0	0	0	0	0	0	0	...
39	55.0	0	1	0	0	0	0	0	0	1	...

5 rows × 22 columns

In [31]:
```
#applying label encoding to y values
label_encoder = LabelEncoder()
y_dt= label_encoder.fit_transform(y)
```

In [32]:
```
y=pd.DataFrame(y_dt, columns=['target'])
y
```

Out[32]:

	target
0	5
1	4
2	5
3	1
4	6
...	...
2232	2
2233	2
2234	1
2235	1
2236	1

2237 rows × 1 columns

In [33]:
```
y.value_counts(normalize=True)
```

Out[33]:
```
target
4         0.265087
2         0.194904
1         0.168082
6         0.150201
5         0.125615
3         0.081359
0         0.014752
dtype: float64
```

In [ ]:

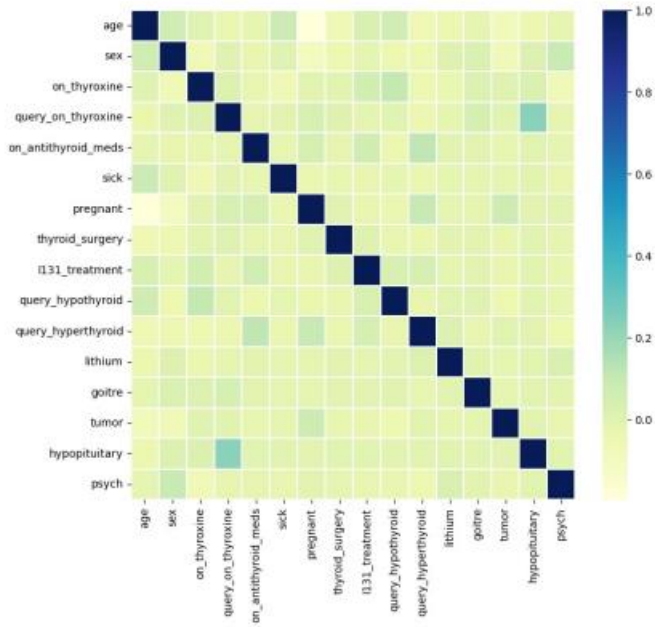
Checking the Correlation

In [34]:
```
#checking correlation using Heatmap
import seaborn as sns
corrmat = x.corr()

f, ax = plt.subplots(figsize =(9, 8))
sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)
```

```
C:\Users\Sree\AppData\Local\Temp\ipykernel_9524\113084675.py:3: FutureWarning: The default value of numeric_onl
y in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or
specify the value of numeric_only to silence this warning.
  corrmat = x.corr()
```

Out[34]:
```
<Axes: >
```



splitting the train and test split

In [35]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

In [36]:
```
y_train.value_counts()
```

Out[36]:
```
target
4         471
2         351
1         302
6         265
5         230
3         144
0          26
dtype: int64
```

In [37]:
```
from imblearn.over_sampling import SMOTE
os = SMOTE(random_state=0,k_neighbors=1)
x_bal,y_bal=os.fit_resample(x_train,y_train)
x_test_bal,y_test_bal=os.fit_resample(x_test,y_test)
```

In [38]:
```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
x_bal = sc.fit_transform(x_bal)
x_test_bal= sc.transform(x_test_bal)
```

In [39]:
```
x_bal
```

Out[39]:
```
array([[-1.62721505, -0.44060477, -0.4238    , ..., -2.50870684,
        -1.40088079,  3.29445097],
       [-0.11561403, -0.44060477,  2.35960359, ..., -0.26259147,
         0.0720981 , -0.19494049],
       [ 1.1874903 ,  2.26960776, -0.4238    , ...,  0.17039463,
        -0.19352184, -0.19494049],
       ...,
       [ 1.395987  , -0.44060477,  2.35960359, ...,  0.43615031,
         0.06101822, -0.19494049],
       [ 0.72882783, -0.44060477,  2.35960359, ...,  0.143333  ,
         0.89086631, -0.19494049],
       [ 1.15628145, -0.44060477,  2.35960359, ...,  0.39723515,
        -0.26588659, -0.19494049]])
```

In [40]:
```
x_test_bal
```

Out[40]:
```
array([[-1.5229667 , -0.44060477, -0.4238    , ...,  1.06342846,
         0.13246609, -0.19494049],
       [-0.89747663, -0.44060477, -0.4238    , ...,  1.76703886,
        -0.30218342, -0.19494049],
       [-0.9496008 ,  2.26960776, -0.4238    , ..., -0.39789962,
        -0.90586329, -0.19494049],
       ...,
       [ 1.39013447, -0.44060477,  2.35960359, ...,  0.81835453,
         0.70094189, -0.19494049],
       [ 1.33846247, -0.44060477,  2.35960359, ...,  0.81987378,
         0.67327619, -0.19494049],
       [-0.19842352, -0.44060477, -0.4238    , ...,  0.24830842,
         0.37610348, -0.19494049]])
```

In [41]:
```
y_bal.value_counts()
```

Out[41]:
```
target
0         471
1         471
2         471
3         471
4         471
5         471
6         471
dtype: int64
```

In [42]:
```
columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surge
```

In [43]:
```
x_test_bal= pd.DataFrame(x_test_bal,columns=columns)
```

In [44]:
```
x_bal= pd.DataFrame(x_bal,columns=columns)
```

In [45]:
```
x_bal
```

Out[45]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	que
0	-1.627215	-0.440605	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
1	-0.115614	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
2	1.187490	2.269608	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
3	-1.366594	-0.440605	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
4	-0.167738	-0.440605	-0.423800	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
...	...	...	...	...	...	...	...	...	...	
3292	0.546923	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
3293	0.383062	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
3294	1.395987	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
3295	0.728028	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	
3296	1.156281	-0.440605	2.359604	-0.105069	-0.158703	-0.141815	-0.137297	-0.239601	-0.162675	

3297 rows × 22 columns

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
rfr = RandomForestClassifier().fit(x_bal,y_bal)
y_pred = rfr.predict(x_test_bal)
accuracy_score(y_test_bal,y_pred)
x_bal.shape,y_bal.shape,x_test_bal.shape,y_test_bal.shape
```

```
C:\Users\Sree\AppData\Local\Temp\ipykernel_9524\2696972469.py:3: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rfr = RandomForestClassifier().fit(x_bal,y_bal)
```

Out[46]:
```
((3297, 22), (3297, 1), (854, 22), (854, 1))
```

In [47]:
```
test_score=accuracy_score(y_test_bal,y_pred)
test_score
```

Out[47]:
```
0.9110070257611241
```

In [48]:
```
train_score = accuracy_score(y_bal,rfr.predict(x_bal))
train_score
```

Out[48]:
```
1.0
```

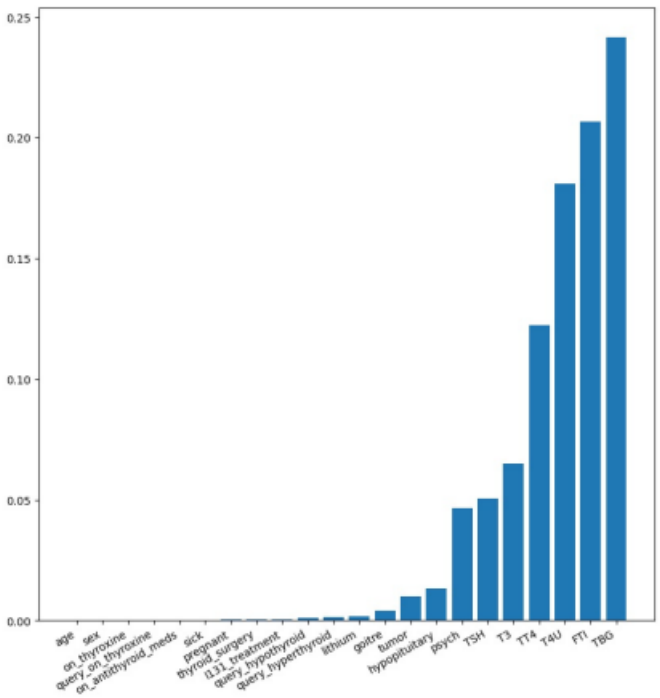
performing feature importance

In [49]:
```
#perform feature importance
from sklearn.inspection import permutation_importance
results = permutation_importance(rfr,x_bal,y_bal, scoring='accuracy')
```

In [50]:
```
#gets importance
feature_importance=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','th
importance = results.importances_mean
importance = np.sort(importance)
#summerize feature importance
for i,v in enumerate(importance):
    i=feature_importance[i]
    print('feature: {:<20} Score: {}'. format(i,v))
#plot important feature

plt.figure(figsize=(10,10))
plt.bar(x=feature_importance, height = importance)
plt.xticks(rotation=30, ha='right')
plt.show()
```

```
feature: age                 Score: 0.0
feature: sex                 Score: 0.0
feature: on_thyroxine        Score: 0.0
feature: query_on_thyroxine  Score: 0.0
feature: on_antithyroid_meds Score: 0.0
feature: sick                Score: 0.0001213224143160385 2
feature: pregnant            Score: 0.0003033068357900963
feature: thyroid_surgery     Score: 0.0003033068357900963
feature: I131_treatment      Score: 0.0006066128715801926
feature: query_hypothyroid   Score: 0.0010312405216863717
feature: query_hyperthyroid  Score: 0.0014558689717925732
feature: lithium             Score: 0.0018198362147406888
feature: goitre              Score: 0.0040036339672429449
feature: tumor               Score: 0.0101304215895389771
feature: hypopituitary       Score: 0.0132848043676069928
feature: psych               Score: 0.0465878807093 6123
feature: TSH                 Score: 0.0584094631483166 45
feature: T3                  Score: 0.0651501364877160 9
feature: TT4                 Score: 0.1224749772520473 3
feature: T4U                 Score: 0.1810130421595389 8
feature: FTI                 Score: 0.2067940552016985 2
feature: TBG                 Score: 0.2416135881104034
```




```
In [51]: x_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_su
In [52]: x_test_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyro
```

In [53]:
```
x_bal.head()
```

Out[53]:

	goitre	tumor	hypopituitary	psych	TSH	T3	TT4	T4U	FTI	TBG
0	-0.052319	-0.137297	-0.024637	-0.107982	-0.315458	-1.035358	-1.704935	-2.508707	-1.400881	3.294451
1	-0.052319	-0.137297	-0.024637	-0.107982	-0.090056	0.155233	-0.197223	-0.262591	0.072098	-0.194940
2	-0.052319	-0.137297	-0.024637	-0.107982	-0.278907	-0.471394	-0.227079	0.170395	-0.193521	-0.194940
3	-0.052319	7.283487	-0.024637	-0.107982	-0.284999	0.989848	0.041622	0.495134	-0.133153	-0.194940
4	-0.052319	-0.137297	-0.024637	-0.107982	-0.306321	4.541622	1.459767	-0.127283	1.496783	-0.194940

In [54]:
```
x_test_bal.head()
```

Out[54]:

	goitre	tumor	hypopituitary	psych	TSH	T3	TT4	T4U	FTI	TBG
0	-0.052319	-0.137297	-0.024637	-0.107982	-0.312412	0.593872	0.788014	1.063428	0.132466	-0.19494
1	-0.052319	-0.137297	-0.024637	-0.107982	-0.314240	0.781860	0.444674	1.767031	-0.302183	-0.19494
2	-0.052319	-0.137297	-0.024637	-0.107982	1.298911	-0.408731	-1.227244	-0.397900	-0.905863	-0.19494
3	-0.052319	-0.137297	-0.024637	-0.107982	-0.166205	-0.471394	-0.227079	-0.397900	0.132466	-0.19494
4	-0.052319	-0.137297	-0.024637	-0.107982	-0.227125	-0.346068	-0.301718	-0.830886	0.434306	-0.19494

RandomForest Model-1

In [55]:
```
rfr1 = RandomForestClassifier()
rfr1.fit(x_bal,y_bal)
y_pred=rfr1.predict(x_test_bal)
```

```
C:\Users\Sree\AppData\Local\Temp\ipykernel_9524\1228887459.py:2: DataConversionWarning: A column-vector y was p
assed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rfr1.fit(x_bal,y_bal)
```

In [56]:
```
print(classification_report(y_test_bal,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.11      0.20       122
           1       0.81      0.93      0.87       122
           2       0.92      0.98      0.95       122
           3       0.77      0.84      0.80       122
           4       0.47      0.89      0.62       122
           5       0.88      0.70      0.78       122
           6       0.60      0.52      0.56       122

    accuracy                           0.71       854
   macro avg       0.75      0.71      0.68       854
weighted avg       0.75      0.71      0.68       854
```

In [57]:
```
train_score = accuracy_score(y_bal,rfr1.predict(x_bal))
```

In [58]:
```
train_score
```

Out[58]:
```
1.0
```

XGBClassifier Model-2

In [59]:
```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(x_bal,y_bal)
```

Out[59]:
```
▾ XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None
,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=Non
e,
              interaction_constraints=None, learning_rate=None, max_bin=None
,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
```

In [60]:
```
y_pred=xgb.predict(x_test_bal)
```

In [61]:
```
print(classification_report(y_test_bal,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.30      0.44       122
           1       0.82      0.94      0.88       122
           2       0.96      1.00      0.98       122
           3       0.77      0.84      0.81       122
           4       0.51      0.81      0.62       122
           5       0.84      0.70      0.76       122
           6       0.59      0.54      0.56       122

    accuracy                           0.73       854
   macro avg       0.76      0.73      0.72       854
weighted avg       0.76      0.73      0.72       854
```

In [62]:
```
train_score = accuracy_score(y_bal, xgb.predict(x_bal))
train_score
```

Out[62]:
```
1.0
```

SVC Model-3

In [63]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

sv= SVC()
```

In [64]:
```
sv.fit(x_bal,y_bal)
```

```
C:\Users\Sree\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[64]:
```
▾ SVC
SVC()
```

In [65]:
```
y_pred = sv.predict(x_test_bal)
```

In [66]:
```
print(classification_report(y_test_bal,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.85      0.77       122
           1       0.76      0.81      0.79       122
           2       0.88      0.93      0.90       122
           3       0.71      0.65      0.68       122
           4       0.71      0.63      0.67       122
           5       0.76      0.54      0.63       122
           6       0.49      0.57      0.52       122

    accuracy                           0.71       854
   macro avg       0.72      0.71      0.71       854
weighted avg       0.72      0.71      0.71       854
```

In [67]:
```
train_score=accuracy_score(y_bal,sv.predict(x_bal))
train_score
```

Out[67]:
```
0.7154989384288747
```

Grid_Search for RandomForest

In [68]:
```
params={
   'n_estimators': [100, 200, 500],
    'criterion': ['gini', 'entropy'],
    'max_depth' : [x for x in range(1,20)]
}
```

In [69]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(rfrl, params, scoring='accuracy',cv=5,n_jobs=-1)
```

In [70]:
```
grid_search.fit(x_bal,y_bal)
```

```
C:\Users\Sree\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:909: DataConversionWarning: A colu
mn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example
using ravel().
  self.best_estimator_.fit(X, y, **fit_params)
```

C:\Users\Sree\thyroid disease classification using ML-20230125T133239Z-001\Thyroid\flask\app.py

app.py

```python
from flask import Flask, render_template, request
import numpy as np
import pickle
import pandas as pd

model = pickle.load(open(r"C:\Users\Sree\thyroid disease classification using ML-20230125T133239Z-001\Thyroid\Training\thyr
le = pickle.load(open("label_encoder.pkl", 'rb'))


app = Flask(__name__)


@app.route("/")
def about():
    return render_template('home.html')


@app.route("/predict")
def home1():
    return render_template('predict.html')


@app.route("/pred", methods=['POST', 'GET'])
def predict():
    x = [[float(x) for x in request.form.values()]]

    print(x)
    col = ['goitre','tumor','hypopituitary','psych','TSH','T3','TT4','T4U','FTI','TBG']
    x = pd.DataFrame(x, columns=col)


    #print(x.shape)

    print(x)
    pred = model.predict(x)
    pred = le.inverse_transform(pred)
    print(pred[0])
    return render_template('submit.html', prediction_text=str(pred))


if __name__ == "__main__":
    app.run(debug=False)

```

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

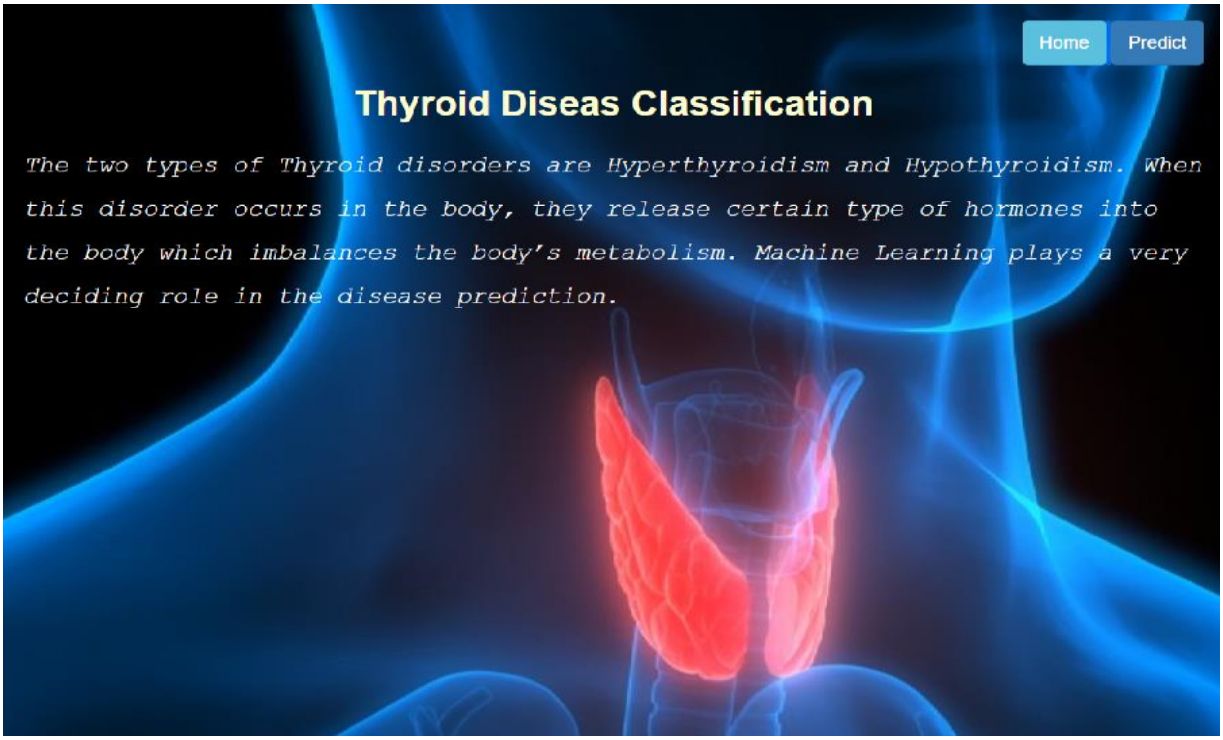
New to Spyder? Read our tutorial

Help Variable Explorer Plots Files

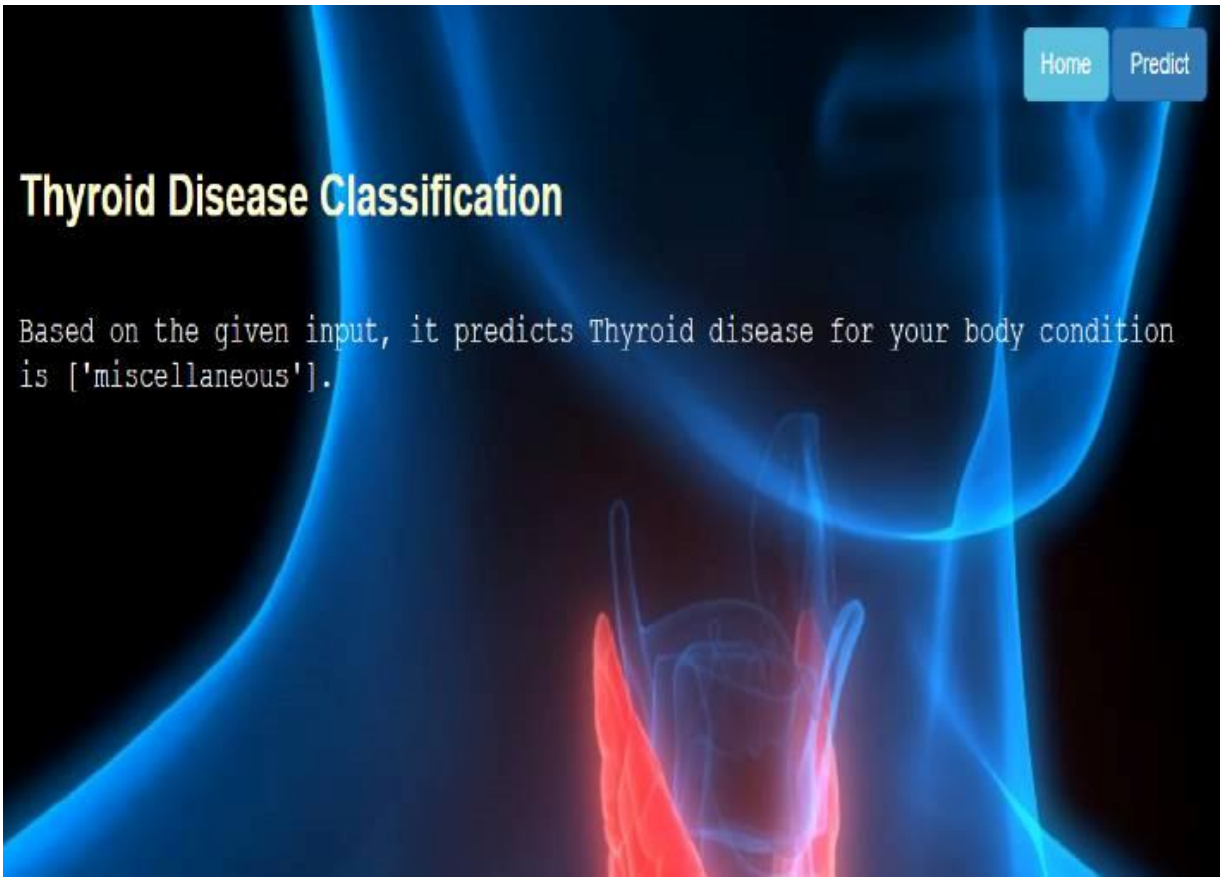
Console 1/A

```
  warnings.warn(
C:\Users\Sree\anaconda3\lib\site-packages\sklearn\base.py:318:
UserWarning: Trying to unpickle estimator LabelEncoder from version
1.1.1 when using version 1.2.2. This might lead to breaking code or
invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-
maintainability-limitations
  warnings.warn(
WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

APPLICATION RUNNING




TSH
TSH
T3
T3
TT4
TT4
T4U
T4U
FTI
FTI
TBG
TBG
Submit


Home
Predict
Thyroid Disease Classification
Based on the given input, it predicts Thyroid disease for your body condition is ['miscellaneous'].

CHAPTER 6

REFERENCES:

1. A. Shrivas and P. Ambastha, "An ensemble approach for classification of thyroid disease with feature optimization," *International Education and Research Journal*, vol. 3, no. 5, pp. 1–4, 2019.
   View at: Google Scholar
2. G. Chaubey, D. Bisen, S. Arjaria, and V. Yadav, "Thyroid disease prediction using machine learning approaches," *National Academy Science Letters*, vol. 3, pp. 128–133, 2021.
   View at: Publisher Site | Google Scholar
3. A. Dewangan, A. Shrivas, and P. Kumar, "Classification of thyroid disease with feature selection technique," *International Journal of Engineering & Technology*, vol. 2, no. 3, pp. 128–133, 2016.
   View at: Google Scholar
4. A. Begum and A. Parkavi, "Prediction of thyroid disease using data mining techniques," in *International Conference on Advanced Computing & Communication Systems (ICACCS)*, pp. 342–345, Coimbatore, India, 2019.
   View at: Google Scholar
5. J. H. Moon and S. Steinhubl, "Digital medicine in thyroidology: a new era of managing thyroid disease," *Endocrinology and Metabolism*, vol. 34, no. 2, pp. 124–131, 2019.