THYROID DISEASE CLASSIFICATION USING ML

AN INDUSTRY ORIENTED MINI REPORT
Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING (AI&ML)

Submitted By

DEEKSHITH KUCHANA	**21UK1A6619**
SREEJA VANGA	**21UK1A6607**
RUCHITHA MACHARLA	**21UK1A6643**
MAHESH ANKAM	**21UK1A6649**

Under the guidance of

MRS. G.NEERAJA

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARITIFICAL INTELLGENCE AND MACHINE LEARNING)

VAAGDEVI ENGINEERING COLLEGE

(Affilted to JNTUH, Hyderbad

BOLLIKUNTA, WARNAGAL (T.S)506005)

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled **"THYROID DISEASE CLASSIFICATION USING MACHINE LEARNING"** is being submitted by DEEKSHITH KUCHNA (21UK1A6619), SREEJAVANGA (21UK1A6607),

RUCHITHA MACHARLA (21UK1A6649), MAHESH ANKAM (21UK1A6649) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023- 2024.

Project Guide   **HOD**

MRS.G.NEERAJA   **DR.K.SHARMILA**

(Assistant Professor)   (Professor)
Department of CSE (AIML)   Department of CSE (AIML)

ACKNOWLEDGEMENT

DEEKSHITH KUCHANA   **(21UK1A6619)**
SREEJA VANGA   **(21UK1A6607)**
RUCHITHA MACHARLA   **(21UK1A6643)**
MAHESH ANKAM   **(21UK1A6649)**

ABSTRACT

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, multi layer 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.

TABLE OF CONTENTS:-

1.INTRODUCTION

1.1. OVERVIEW

➤ The thyroid gland is an endocrine gland divided into the right and left lobes, which are situated on opposite sides of the trachea in the throat, with an isthmus connecting them .

➤ It has the appearance of a butterfly gland and weighs around 25 grams in adults.

➤ The thyroid gland. It has two types of secretions within its functioning system, triiodothyronine (T3) and another is thyroxine (T4).

➤ The element used for these two hormones is iodine in the blood because iodine is the main component to building these two hormones, T3 and T4. As for (T3), this label came because it consists of three atoms of iodine, while (T4) this label came because it consists of four atoms of iodine, and the critical role of these hormones is to control the metabolism process.

➤ TRH (TSH Releasing Hormone) is generated in the hypothalamus,located in the upper part of the brain and secreted by the thyroid gland. Thyroid hormone levels in the blood affect the volume of TRH hormone secreted by the hypothalamus.

➤ T3 and T4 development occur until the body's requirements decide the optimum blood level .

➤ Thyroid disease is classified into three important categories:
  1. Hyperthyroidism.
   2. Hypothyroidism.
  3. Normal.

➤ Create a system that can identify whether a person has a thyroid disease based on their medical data. Medical datasets available online (like from the UCI Machine Learning Repository or Kaggle) Medical records from hospitals or clinics. Patient information (age, gender, etc.). Medical test results (like TSH, T3, T4 levels)

➤ Split your data into a training set (to teach the model) and a testing set (to evaluate the model) Train the model using the training set. Fine-tune the model for better performance (this is called hyperparameter tuning).

➤ Create an API (a way for other programs to interact with your model) using tools like Flask or FastAPI. Host this API on a server (like AWS, Google Cloud, or Azure).
   Build a simple user interface (UI) for doctors or patients to use.

Location of thyroid gland.

Thyroid hormones
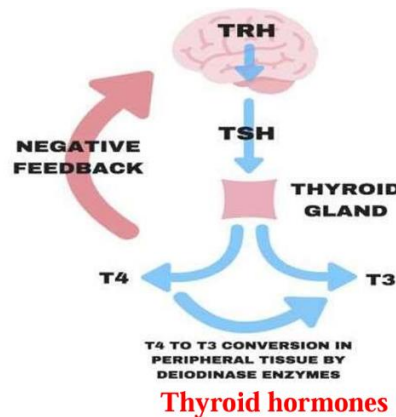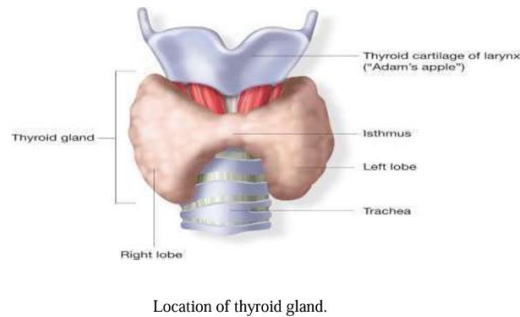
1.2. PURPOSE

- ➢ The purpose of this article is to demonstrate how a prototype that uses text mining and machine learning approaches to detect strokes may be employed. Machine learning may be a significant tracker when correctly trained machine learning algorithms are used in surveillance, nursing, and data processing.
- ➢ The semantic and syntactic analysis of information monitoring is provided by the data mining methods utilized in this work. At Sugam Multispecialty Hospital in Kumbakonam, Tamil Nadu, India, 507 patient case sheets were gathered using the data collecting approach.
- ➢ Machine learning methods utilized to analyze the data included artificial neural networks, support vector machines, boosting and bagging, and random forests. Using a classification accuracy of 95% and a standard deviation of 14.69, artificial neural networks trained with a stochastic gradient descent approach outperformed the other techniques.
- ➢ Thyroid diseases are becoming increasingly common around the world. Hypothyroidism, hyperthyroidism, or thyroid cancer affect one out of every eight women.
- ➢ Thyroid categorization is important for medical researchers because medical reports show major thyroid dysfunctions among the population, with women being the most affected.
- ➢ The literature mentions several studies in thyroid classification that use various machine learning techniques to develop robust classifiers.
- ➢ The literature mentions several studies in thyroid classification that use various machine learning techniques to develop robust classifiers. The goal we want to reach or the primary goal:
- ➢ Comparison of the performance of the eight machine learning algorithms in predicting thyroid disease.
  - ✓ Extract useful patterns from large and complex clinical data.
  - ✓ Make the study work to show the following results.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

Some existing problems in thyroid disease classification using Machine Learning (ML) include:

1. **Limited datasets**: Availability of large, diverse, and high-quality datasets for thyroid disease classification is limited.

2. **Class imbalance**: Thyroid disease classes have unequal distributions, with benign nodules being more common than malignant tumors.

3. **Feature selection**: Selecting relevant features from various data sources (e.g., imaging, lab tests, clinical data) is challenging.

4. **Data quality**: Noisy, missing, or inconsistent data can affect ML model performance.

5. **Model interpretability**: ML models can be difficult to interpret, making it challenging to understand the reasoning behind predictions.

6. **Overfitting**: ML models can overfit the training data, resulting in poor generalization performance on new data.

7. **Lack of standardization**: Different datasets and models make it difficult to compare and reproduce results.

8. **Limited clinical validation**: ML models may not be extensively validated in clinical settings.

9. **Ethical concerns**: Bias in datasets or models can lead to unfair outcomes, and patient privacy must be ensured.

10. **Regulatory approval**: ML models must meet regulatory requirements for medical device approval.

11. **Explainability**: ML models need to provide clear explanations for their predictions to ensure trust in the model.

12. **Handling rare cases**: ML models may struggle to accurately classify rare thyroid disease cases.

2.2 PROPOSED SOLLUTION

To address the challenges in thyroid disease classification using machine learning (ML), the following solutions can be proposed:

1. **Enhanced Data Collection and Management**:

   Larger and Diverse Datasets: Collaborate with multiple healthcare institutions to gather more extensive and diverse datasets that capture a wide range of patient demographics and conditions.

   - *Data Augmentation*: Use data augmentation techniques to artificially increase the size of the dataset, thereby improving model robustness and generalization.

2. **Advanced Feature Selection and Engineering**:

   Feature Importance Analysis: Utilize techniques like Recursive Feature Elimination (RFE) and SHAP (SHapley Additive exPlanations) values to identify and prioritize the most important features.

   Domain Expertise: Incorporate domain knowledge from endocrinologists and other medical experts to guide feature selection and engineering.

3. **Improved Model Interpretability**:

   Explainable AI (XAI): Implement explainable AI techniques to make model decisions more transparent and interpretable for healthcare professionals. Techniques like LIME (Local Interpretable Model-agnostic Explanations) and SHAP can provide insights into model predictions.

   Rule-Based Systems: Combine ML models with rule-based systems that use medical knowledge to enhance interpretability and trustworthiness.

4. **Robust Model Training and Validation**:

   Cross-Validation: Use cross-validation techniques to ensure the model generalizes well to new data. This helps in detecting and mitigating overfitting.

   Ensemble Methods: Implement ensemble methods (e.g., random forests, gradient boosting) to combine multiple models and improve overall accuracy and robustness.

5. **Standardization and Normalization**:

   Data Preprocessing: Apply standardization and normalization techniques to reduce variability in the data and ensure consistency across different datasets.

   Transfer Learning: Use transfer learning to leverage pre-trained models on similar tasks, which can improve performance with smaller datasets.

3. THEORITICAL ANALYSIS
3.1. BLOCK DIAGRAM
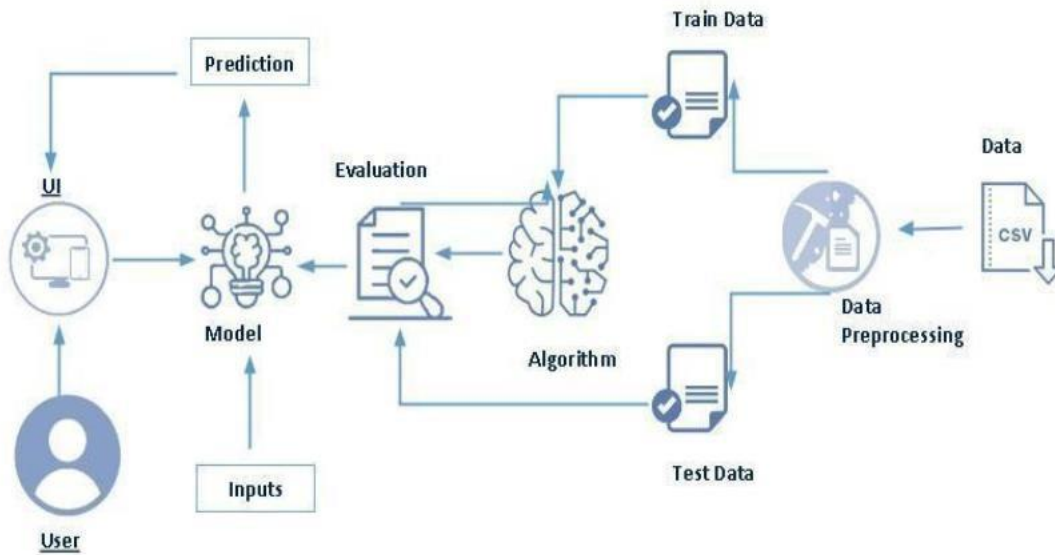
3.2. SOFTWARE DESIGNING

The following is the Software required to complete this project:

○ **Google Colab**: Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

○ **Dataset (CSV File)**: The dataset in CSV format is essential for training and testing your predictive model.

○ **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.

- **Feature Selection/Drop**: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.

- **Model Training Tools**: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the thyroid disease classification.

- **Model Accuracy Evaluation**: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to thyroid disease categories based on historical data.

- **UI Based on Flask Environment**: Flask, a Python web framework, will be used to develop the user interface (UI) for the system.

- Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the thyroid disease, allowing users to rely on the thyroid disease and associated health information.

4. EXPERIMENTAL INVESTIGATION

In this project, we have used thyroid classification Dataset. This dataset is a csv file consisting of labelled data and having the following columns-

- **Age**: Patient's age.
- **Sex**: Patient's sex (male/female).
- **TSH (Thyroid Stimulating Hormone):** Blood test value indicating the amount of TSH.
- **T3 (Triiodothyronine):** Blood test value indicating the amount of T3.
- **T4U (Thyroxine uptake):** Blood test value indicating the amount of T4U.
- **FTI (Free Thyroxine Index):** Blood test value indicating the amount of FTI.
- **TT4 (Total Thyroxine):** Blood test value indicating the amount of TT4.
- **Query Hypothyroid**: Boolean indicating if hypothyroidism is suspected.
- **Query Hyperthyroid:** Boolean indicating if hyperthyroidism is suspected.
- **On Thyroxine:** Boolean indicating if the patient is on thyroxine medication.
  On Antithyroid Medication: Boolean indicating if the patient is on antithyroid medication.
- **Sick:** Boolean indicating if the patient is currently sick.
- **Pregnant:** Boolean indicating if the patient is pregnant.
- **Thyroid Surgery:** Boolean indicating if the patient has had thyroid surgery.
- I131 Treatment: Boolean indicating if the patient has had I131 treatment.
- **Lithium:** Boolean indicating if the patient is on lithium medication.
- **Goiter:** Boolean indicating if the patient has a goiter.
- **Tumor:** Boolean indicating if the patient has a tumor.
- **Hypopituitary:** Boolean indicating if the patient has hypopituitarism.

5. FLOWCHART
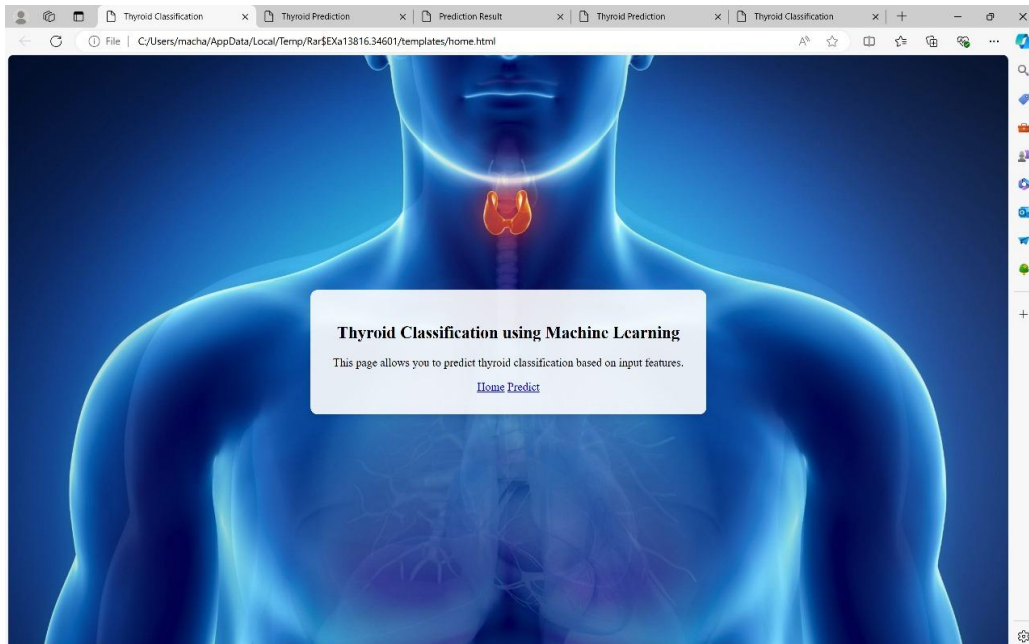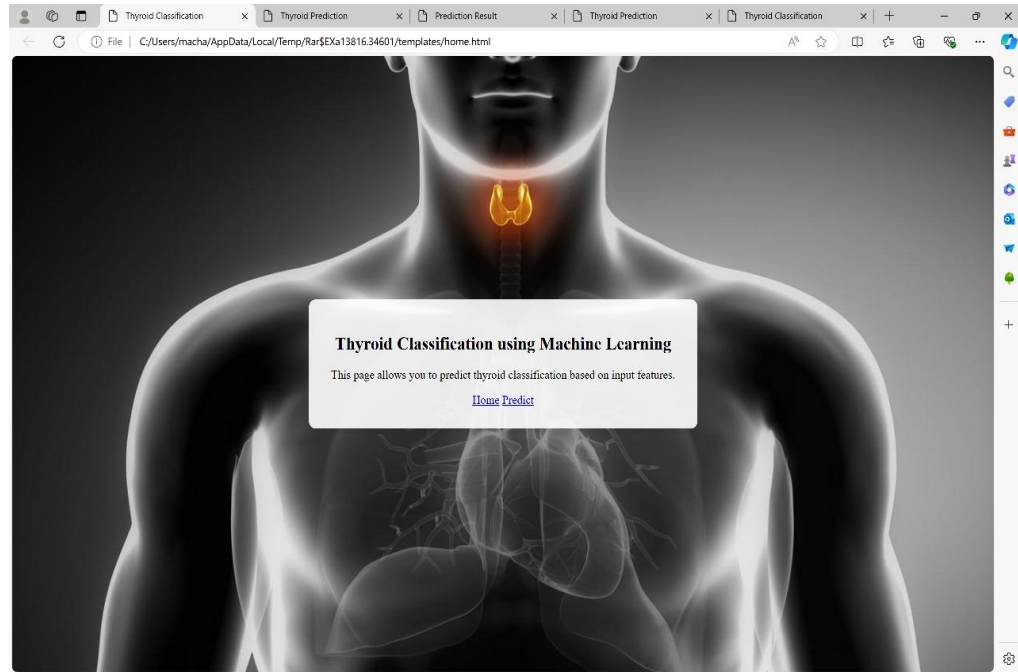
Thyroid disease datasets

↓

Data preprocessing (data encoding, data cleaning, data resampling - SMOTE and data normalization)

↓

Feature- target arrangement, feature statistics computation, feature heatmap visualization

↓

Feature-label export, Data division/ splitting

↓

Training data   Test data

↓

Statistical analysis of features (Dist plot, join plot etc.)

↓

K-fold cross validation, hyper parameter tuning

↓

Machine learning algorithms (SVM, NN, KNN, DT, LR etc.) & Ensemble Classifiers

↓

Final classification model

↓

Thyroid diagnosis &

Confusion Metrics, Performance Measures, ROC Plot etc.

6. RESULT

HOME PAGE :

PREDICTION PAGE :

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

❖ **Improved accuracy**: ML algorithms can analyze complex data patterns and make predictions more accurately than human clinicians.

❖ **Increased efficiency**: ML can analyze large amounts of data quickly, reducing the time spent by clinicians on diagnosis and treatment planning.

❖ Automated diagnosis: ML can automate the diagnosis process, reducing the need for manual analysis and interpretation.

❖ **Personalized medicine**: ML can help personalize treatment plans based on individual patient characteristics and disease patterns.

❖ **Identification of high-risk patients**: ML can identify patients at high risk of developing thyroid cancer or other severe diseases, enabling early intervention.

❖ **Reduced costs**: ML can reduce healthcare costs by minimizing the need for repeated tests, biopsies, and other invasive procedures.

❖ **Improved patient stratification**: ML can group patients into subcategories based on disease characteristics, leading to more targeted treatment approaches.

❖ **Faster diagnosis**: ML can provide faster diagnosis compared to traditional methods.

DISADVANTAGES:

❖ **Data quality issues**: ML algorithms require high-quality data, but thyroid disease data can be noisy, missing, or inconsistent.

❖ **Limited datasets**: Availability of large, diverse, and high-quality datasets for thyroid disease classification is limited.

❖ **Lack of interpretability**: ML models can be difficult to interpret, making it challenging to understand the reasoning behind predictions.

❖ **Dependence on technology**: ML models require significant technological infrastructure and expertise, which can be a barrier for some healthcare institutions.

❖ **Limited clinical validation**: ML models may not be extensively validated in clinical settings, leading to concerns about their reliability.

❖ **Cybersecurity risks**: ML models can be vulnerable to cyber attacks and data breaches, compromising patient data and confidentiality.

❖ **Limited transparency**: ML models can be complex and difficult to understand, making it challenging to identify and address errors

8.APPLICATIONS

❖ **Clinical Decision Support Systems (CDSSs)**: ML models can be integrated into CDSSs to provide clinicians with data-driven insights for diagnosis and treatment.
❖ **Personalized Medicine**: ML can help personalize treatment plans based on individual patient characteristics and disease patterns.
❖ **Drug Discovery and Development**: ML can aid in identifying potential drug targets and predicting drug efficity and toxicity.
❖ **Predictive Analytics**: ML models can predict patient outcomes, disease progression, and treatment response.
❖ **Electronic Health Records (EHRs) Analysis**: ML can help analyze EHRs to identify patterns and predict patient outcomes.
❖ **Telemedicine**: ML can enable remote diagnosis and monitoring of thyroid diseases.
❖ Medical Device Development: ML can aid in developing new medical devices, such as wearable sensors, for thyroid disease diagnosis and monitoring.
❖ **Patient Stratification**: ML can group patients into subcategories based on disease characteristics, leading to more targeted treatment approaches.
❖ **Healthcare Management**: ML can help optimize resource allocation and patient flow in healthcare systems.

9.CONCLUSION

Thyroid disease is one of the diseases that afflict the world's population, and the number of cases of this disease is increasing. Because of medical reports that show serious imbalances in thyroid diseases, our study deals with the classification of thyroid disease between hyperthyroidism and hypothyroidism. This disease was classified using algorithms. Machine learning showed us good results using several algorithms and was built in the form of two models. In the first model, all the characteristics consisting of 16 inputs and one output were taken, and the result of the accuracy of the random forest algorithm was 98.93, which is the highest accuracy among the other algorithms. In the second embodiment, the following characteristics were omitted based on a previous study. The removed attributes were 1- query_thyroxine 2- query_hypothyoid 3-query_hyperthyroid. Here we have included the increased accuracy of some algorithms, as well as the retention of the accuracy of others.

10.FUTURE SCOPE

➢ We predicted and classified thyroid disease by applying machine learning techniques to a data set consisting of 1250 actual samples. We divided the dataset as follows: 30% of the data were used for training, and 70% were used for testing.

➢ After applying these techniques to dataset one that consists of all the characteristics, the random forest algorithm obtained an accuracy rate of 98.93%. In the second step, and based on a previous study, we deleted a set of features which are 1- query_thyroxine 2- query_hypothyorid 3- query_hyperthyroid.

➢ We applied machine learning techniques to this data, and the MLP algorithm got the highest accuracy of 95.73%. The results obtained in this study help us in the rapid prediction of thyroid disease. And the classification of the disease (Hyperthyroidism or Hypothyroidism).

➢ Future work should focus on improving the performance of classification algorithms and using different approaches from feature selection methods to obtain better results.

11.BIBILOGRAPHY

1. A. T. Azar, A. E. Hassanien, and T. Kim, AI, arXiv:1403.0522, (2012)

2. K. Salman, E. Sonuc, J. Phys, 1963, 012140, (2021)

3. A. C. C. Heuck, "World Health Organization," 2000, https://www.who.int/

4. A. Tyagi and R. Mehra, Interactive Thyroid Disease Prediction System using Machine

Learning Techniques, in the Proceedings of the 5th IEEE International Conference on

Parallel, Distributed Grid Computing (PDGC-2018), 20-22 Dec, 2018, Solan, India,

(2018)

5. Y. F. Wang, Comparison Study of Radiomics and Deep-Learning Based on Methods for

Thyroid Nodules Classification Using Ultrasound Images, 8, IEEE Access, (2020)

6. R. Chandan, M. S. Chethan, C. Vasan, H. S. Devikarani, Int. J. Eng. App. Sci. Tech. 5, 9, (2021)

7. S. Godara, Intl. J. Elect. Engg. 10, 2, (2018)

8. P. Pavani, P. P. Sadu Naik, Int. J. Inno. Res. Tech. 9, 3, (2022)

9. Sunila, R. Singh, and Sanjeeev Kumar, Ind. J. Sci. and Tech. 9, (2016)

10. S. Godara and R. Singh, Ind. J. Sci. and Tech. 910, (2016)

11. Z. Obermeyer, E. Manuel, N. Engl. 375, (2016)

12. A.K. Pandey, P. Pandey, and K.L. Jaiswal, IUP J. Comp. Sci., 7, 3, (2013)

13. S. Ismaeel, A. Miri, and D. Chourishi, Using the Extreme Learning Machine (ELM) technique for heart disease diagnosis, in Proceedings of the IEEE Canada International Humanitarian Technology Conference, (2015)

14. Thyroid dataset, "UCI Machine Learning Repository", https://archive.ics.uci.edu/ml/datasets/thyroid+disease

15. Evaluation metrics, "Basic measurements derived from confusion matrix", https://classeval.wordpress.com/introduction/basic-evaluation-measures.

12.APPENDIX

Model building :

1)Dataset
2)Google colab and VS code Application Building
    1. HTML file (Home file, Predict file)
    1. CSS file
    2. Models in pickle format

1.SOURCE CODE:

HOME.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Thyroid Classification</title>
  <link rel="stylesheet" href="../static/style.css">
</head>
<body>
  <div class="container">
    <h1>Thyroid Classification using Machine Learning</h1>
    <p>This page allows you to predict thyroid classification based on input features.</p>
    <a href='{{url_for("home")}}'>Home</a>
    <a href='{{url_for("predict")}}'>Predict</a>
  </div>
</body>
</html>
```

PREDICT.HTML

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Thyroid Prediction</title>
    <link rel="stylesheet" href="../static/style.css">
</head>
```

```html
<body>
    <div class="container" id="a1" style="padding-top:1200px ">
        <form id="prediction-form" action="{{url_for('predict')}}" method="POST">
            <div class="form-group">
                <label for="age">Age</label>
                <input type="number" id="age" name="age" value="35" required>
            </div>
            <div class="form-group">
                <label for="sex">Sex</label>
                <select id="sex" name="sex" required>
                    <option value="F">Female</option>
                    <option value="M">Male</option>
                </select>
            </div>
            <div class="form-group">
                <label for="on_thyroxine">On Thyroxine</label>
                <select id="on_thyroxine" name="on_thyroxine" required>
                    <option value="f">False</option>
                    <option value="t">True</option>
                </select>
            </div>
            <div class="form-group">
                <label for="query_on_thyroxine">Query on Thyroxine</label>
                <select id="query_on_thyroxine" name="query_on_thyroxine" required>
                    <option value="f">False</option>
                    <option value="t">True</option>
                </select>
            </div>
            <div class="form-group">
                <label for="on_antithyroid_medication">On Antithyroid
Medication</label>
                <select id="on_antithyroid_medication" name="on_antithyroid_medication"
required>
                    <option value="f">False</option>
                    <option value="t">True</option>
                </select>
            </div>
            <div class="form-group">
                <label for="sick">Sick</label>
                <select id="sick" name="sick" required>
                    <option value="f">False</option>
                    <option value="t">True</option>
                </select>
            </div>
            <div class="form-group">
                <label for="pregnant">Pregnant</label>
```

```html
        <select id="pregnant" name="pregnant" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="thyroid_surgery">Thyroid Surgery</label>
        <select id="thyroid_surgery" name="thyroid_surgery" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="I131_treatment">I131 Treatment</label>
        <select id="I131_treatment" name="I131_treatment" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="query_hypothyroid">Query Hypothyroid</label>
        <select id="query_hypothyroid" name="query_hypothyroid" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="query_hyperthyroid">Query Hyperthyroid</label>
        <select id="query_hyperthyroid" name="query_hyperthyroid" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="lithium">Lithium</label>
        <select id="lithium" name="lithium" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="goitre">Goitre</label>
        <select id="goitre" name="goitre" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
```

```
</div>
<div class="form-group">
    <label for="tumor">Tumor</label>
    <select id="tumor" name="tumor" required>
        <option value="f">False</option>
        <option value="t">True</option>
    </select>
</div>
<div class="form-group">
    <label for="hypopituitary">Hypopituitary</label>
    <select id="hypopituitary" name="hypopituitary" required>
        <option value="f">False</option>
        <option value="t">True</option>
    </select>
</div>
<div class="form-group">
    <label for="psych">Psych</label>
    <select id="psych" name="psych" required>
        <option value="f">False</option>
        <option value="t">True</option>
    </select>
</div>
<div class="form-group">
    <label for="TSH_measured">TSH Measured</label>
    <select id="TSH_measured" name="TSH_measured" required>
        <option value="f">False</option>
        <option value="t">True</option>
    </select>
</div>
<div class="form-group">
    <label for="TSH">TSH</label>
    <input type="number" step="any" id="TSH" name="TSH" required>
</div>
<div class="form-group">
    <label for="T3_measured">T3 Measured</label>
    <select id="T3_measured" name="T3_measured" required>
        <option value="f">False</option>
        <option value="t">True</option>
    </select>
</div>
<div class="form-group">
    <label for="T3">T3</label>
    <input type="number" step="any" id="T3" name="T3" required>
</div>
<div class="form-group">
    <label for="TT4_measured">TT4 Measured</label>
```

```html
        <select id="TT4_measured" name="TT4_measured" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="TT4">TT4</label>
        <input type="number" step="any" id="TT4" name="TT4" required>
    </div>
    <div class="form-group">
        <label for="T4U_measured">T4U Measured</label>
        <select id="T4U_measured" name="T4U_measured" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="T4U">T4U</label>
        <input type="number" step="any" id="T4U" name="T4U" required>
    </div>
    <div class="form-group">
        <label for="FTI_measured">FTI Measured</label>
        <select id="FTI_measured" name="FTI_measured" required>
            <option value="f">False</option>
            <option value="t">True</option>
        </select>
    </div>
    <div class="form-group">
        <label for="FTI">FTI</label>
        <input type="number" step="any" id="FTI" name="FTI" required>
    </div>

    <button type="submit">Submit</button>
</form>
<h3>{{ prediction_text }}</h3>
    </div>
</body>
</html>
```

APP.PY

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

```python
# Load the model
model = pickle.load(open('model.pkl', 'rb'))

app = Flask(_name_)

# Define columns for one-hot encoding
columns_to_encode = [
    'sex', 'on_thyroxine', 'on_antithyroid_medication', 'sick', 'pregnant',
    'thyroid_surgery', 'I131_treatment', 'query_on_thyroxine', 'query_hypothyroid',
    'query_hyperthyroid', 'lithium', 'goitre', 'tumor', 'hypopituitary', 'psych',
    'TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured'
]

# Assuming you have a list of all feature names used during training
feature_names = [
    'age', 'TSH', 'TT4', 'T4U', 'FTI', 'T3',
    'sex_F', 'sex_M', 'on_thyroxine_f', 'on_thyroxine_t',
    'on_antithyroid_medication_f', 'on_antithyroid_medication_t', 'sick_f', 'sick_t',
    'pregnant_f', 'pregnant_t', 'thyroid_surgery_f', 'thyroid_surgery_t',
    'I131_treatment_f', 'I131_treatment_t', 'query_on_thyroxine_f', 'query_on_thyroxine_t',
    'query_hypothyroid_f', 'query_hypothyroid_t', 'query_hyperthyroid_f', 'query_hyperthyroid_t',
    'lithium_f', 'lithium_t', 'goitre_f', 'goitre_t', 'tumor_f', 'tumor_t',
    'hypopituitary_f', 'hypopituitary_t', 'psych_f', 'psych_t',
    'TSH_measured_f', 'TSH_measured_t', 'T3_measured_f', 'T3_measured_t',
    'TT4_measured_f', 'TT4_measured_t', 'T4U_measured_f', 'T4U_measured_t',
    'FTI_measured_f', 'FTI_measured_t'
]

@app.route("/", methods=['GET'])
def home():
    return render_template('home.html')

# Route for serving the prediction form
@app.route('/pred', methods=['GET'])
def form():
    return render_template('predict.html')

# Route for handling form submission and making predictions
@app.route("/pred", methods=['POST'])
def predict():
    # Collect form data
    form_data = {
```

```python
    'age': float(request.form['age']),
    'TSH': float(request.form['TSH']),
    'TT4': float(request.form['TT4']),
    'T4U': float(request.form['T4U']),
    'FTI': float(request.form['FTI']),
    'T3': float(request.form['T3']),  # Add 'T3' to form data collection
    'sex': request.form['sex'],
    'on_thyroxine': request.form['on_thyroxine'],
    'on_antithyroid_medication': request.form['on_antithyroid_medication'],
    'sick': request.form['sick'],
    'pregnant': request.form['pregnant'],
    'thyroid_surgery': request.form['thyroid_surgery'],
    'I131_treatment': request.form['I131_treatment'],
    'query_on_thyroxine': request.form['query_on_thyroxine'],
    'query_hypothyroid': request.form['query_hypothyroid'],
    'query_hyperthyroid': request.form['query_hyperthyroid'],
    'lithium': request.form['lithium'],
    'goitre': request.form['goitre'],
    'tumor': request.form['tumor'],
    'hypopituitary': request.form['hypopituitary'],
    'psych': request.form['psych'],
    'TSH_measured': request.form['TSH_measured'],
    'T3_measured': request.form['T3_measured'],
    'TT4_measured': request.form['TT4_measured'],
    'T4U_measured': request.form['T4U_measured'],
    'FTI_measured': request.form['FTI_measured'],
}

print("Form Data:")
print(form_data)



# Create DataFrame from form data
input_data = pd.DataFrame([form_data])

# Perform one-hot encoding for categorical columns
input_data_encoded = pd.get_dummies(input_data, columns=columns_to_encode)

# Ensure that all expected columns are present and in correct order
input_data_encoded = input_data_encoded.reindex(columns=feature_names, fill_value=0)

# Check for 'target' column
```

```
    if 'target' in input_data_encoded.columns:
        input_data_encoded.drop('target', axis=1, inplace=True)  # Drop 'target' if present

    # Make predictions
    pred = model.predict(input_data_encoded)

    return render_template('predict.html', prediction_text=str(pred[0]))

if _name_ == "_main_":
    app.run(debug=True)
```

CODE SNIPPETS

MODEL BUILDING

Welcome_To_Colab (2).ipynb

File Edit View Insert Runtime Tools Help   All changes saved

Comment   Share

+ Code   + Text

Files

```
df.describe()
df1=df[df['age']=='455']
df1
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid
2336	455	F	f		f	f	f		f	f

1 rows × 30 columns

[6] `df.sex.unique()`

```
array(['F', 'M', '?'], dtype=object)
```

[7] `#df=df.dropna(subset=['sex'])`

[8] `#df.describe()`

[9] `df.sex.unique()`

```
array(['F', 'M', '?'], dtype=object)
```

[10] `#df=df[df.sex!='?']`

[11] `df.shape`

Disk   80.28 GB available

✓ 0s   completed at 1:29 PM

Welcome_To_Colab (2).ipynb

File Edit View Insert Runtime Tools Help   All changes saved

Comment   Share

+ Code   + Text

Files

[11] `df.shape`

```
(4744, 30)
```

[12]
```
import sklearn
from sklearn.impute import SimpleImputer
```

`df.nunique()`

```
age                          94
sex                           3
on_thyroxine                  2
query_on_thyroxine            2
on_antithyroid_medication     2
sick                          2
pregnant                      2
thyroid_surgery               2
I131_treatment                2
query_hypothyroid             2
query_hyperthyroid            2
lithium                       2
goitre                        2
tumor                         2
hypopituitary                 2
psych                         2
TSH_measured                  2
TSH                         288
T3_measured                   2
T3                           70
TT4_measured                  2
```

Disk   80.28 GB available

✓ 0s   completed at 1:29 PM

colab.research.google.com/drive/1MfTysxO4Gwe5AP9Nq0p3gNoWwjAipT5f#scrollTo=9R27Tt0gpOHf

Welcome_To_Colab (2).ipynb ☆

File Edit View Insert Runtime Tools Help   All changes saved

+ Code + Text

```python
df.nunique()
```

```
age                          94
sex                           3
on_thyroxine                  2
query_on_thyroxine            2
on_antithyroid_medication     2
sick                          2
pregnant                      2
thyroid_surgery               2
I131_treatment                2
query_hypothyroid             2
query_hyperthyroid            2
lithium                       2
goitre                        2
tumor                         2
hypopituitary                 2
psych                         2
TSH_measured                  2
TSH                         288
T3_measured                   2
T3                           70
TT4_measured                  2
TT4                         242
T4U_measured                  2
T4U                         147
FTI_measured                  2
FTI                         235
TBG_measured                  1
TBG                           1
referral_source               5
target                       10
dtype: int64
```

✓ 0s   completed at 1:29 PM

---

colab.research.google.com/drive/1MfTysxO4Gwe5AP9Nq0p3gNoWwjAipT5f#scrollTo=9R27Tt0gpOHf

Welcome_To_Colab (2).ipynb ☆

File Edit View Insert Runtime Tools Help   All changes saved

+ Code + Text

```python
[14] impute=SimpleImputer(strategy='most_frequent')
     impute1=SimpleImputer(strategy='median')
```

```python
[15] df.replace('?',np.nan,inplace=True)
     df[['sex']]=impute.fit_transform(df[['sex']])
```

```python
[16] df['sex'].nunique()
```
```
    2
```

```python
[17] df['sex'].shape
```
```
    (4744,)
```

```python
[18] df.info()
     df['age'].unique()
```
```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 4744 entries, 0 to 4743
    Data columns (total 30 columns):
     #   Column                     Non-Null Count  Dtype
    ---  ------                     --------------  -----
     0   age                        4743 non-null   object
     1   sex                        4744 non-null   object
     2   on_thyroxine               4744 non-null   object
     3   query_on_thyroxine         4744 non-null   object
     4   on_antithyroid_medication  4744 non-null   object
```

✓ 0s   completed at 1:29 PM

```python
[18]  df.info()
      df['age'].unique()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4744 entries, 0 to 4743
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     4743 non-null   object
 1   sex                     4744 non-null   object
 2   on_thyroxine            4744 non-null   object
 3   query_on_thyroxine      4744 non-null   object
 4   on_antithyroid_medication  4744 non-null   object
 5   sick                    4744 non-null   object
 6   pregnant                4744 non-null   object
 7   thyroid_surgery         4744 non-null   object
 8   I131_treatment          4744 non-null   object
 9   query_hypothyroid       4744 non-null   object
 10  query_hyperthyroid      4744 non-null   object
 11  lithium                 4744 non-null   object
 12  goitre                  4744 non-null   object
 13  tumor                   4744 non-null   object
 14  hypopituitary           4744 non-null   object
 15  psych                   4744 non-null   object
 16  TSH_measured            4744 non-null   object
 17  TSH                     4290 non-null   object
 18  T3_measured             4744 non-null   object
 19  T3                      3791 non-null   object
 20  TT4_measured            4744 non-null   object
 21  TT4                     4466 non-null   object
 22  T4U_measured            4744 non-null   object
 23  T4U                     4267 non-null   object
```

completed at 1:29 PM

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     4743 non-null   object
 1   sex                     4744 non-null   object
 2   on_thyroxine            4744 non-null   object
 3   query_on_thyroxine      4744 non-null   object
 4   on_antithyroid_medication  4744 non-null   object
 5   sick                    4744 non-null   object
 6   pregnant                4744 non-null   object
 7   thyroid_surgery         4744 non-null   object
 8   I131_treatment          4744 non-null   object
 9   query_hypothyroid       4744 non-null   object
 10  query_hyperthyroid      4744 non-null   object
 11  lithium                 4744 non-null   object
 12  goitre                  4744 non-null   object
 13  tumor                   4744 non-null   object
 14  hypopituitary           4744 non-null   object
 15  psych                   4744 non-null   object
 16  TSH_measured            4744 non-null   object
 17  TSH                     4290 non-null   object
 18  T3_measured             4744 non-null   object
 19  T3                      3791 non-null   object
 20  TT4_measured            4744 non-null   object
 21  TT4                     4466 non-null   object
 22  T4U_measured            4744 non-null   object
 23  T4U                     4267 non-null   object
 24  FTI_measured            4744 non-null   object
 25  FTI                     4269 non-null   object
 26  TBG_measured            4744 non-null   object
 27  TBG                     0 non-null      float64
 28  referral_source         4744 non-null   object
 29  target                  4744 non-null   object
dtypes: float64(1), object(29)
memory usage: 1.1+ MB
```

completed at 1:29 PM

colab.research.google.com/drive/1MfTysxO4Gwe5AP9Nq0p3gNoWwjAipT5f#scrollTo=BPgxr73svGED

Welcome_To_Colab (2).ipynb ☆

File Edit View Insert Runtime Tools Help   All changes saved

+ Code   + Text

```
[19] df['age']=pd.to_numeric(df['age'],errors='coerce')
     df=df[df['age']!=455]
     mean_age=df['age'].mean()
     df['age']=df['age'].fillna(mean_age) #df['age']=impute1.fit_transform(df[['age']])
     df['age']=df['age'].round(0).astype('int')
     df['age'].unique()
```

```
<ipython-input-19-17bd11929f6f>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['age']=df['age'].fillna(mean_age) #df['age']=impute1.fit_transform(df[['age']])
<ipython-input-19-17bd11929f6f>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['age']=df['age'].round(0).astype('int')
array([35, 63, 25, 53, 92, 67, 60, 48, 27, 73, 19, 72, 16, 54, 39, 38, 33,
       45, 21, 51, 64, 40, 71, 49, 79, 20, 59, 37, 42, 46, 50, 69, 30, 31,
       89, 77, 68, 65, 24, 75, 80, 23, 11, 18, 62, 76, 22, 70, 47, 56, 26,
       28, 83, 74, 78, 58, 61, 55, 41, 85, 86, 32, 43, 17, 57, 66, 34, 14,
       52, 93, 36, 81, 84, 15, 12, 44, 29, 82, 87, 88,  7,  1, 13, 10, 90,
       94,  4,  8,  5,  2, 91,  6])
```

```
[20] #df['sex'].unique()
```

```
[21] #row=df[df[['age']]!=int]
```

✓ 0s   completed at 1:29 PM

---

colab.research.google.com/drive/1MfTysxO4Gwe5AP9Nq0p3gNoWwjAipT5f#scrollTo=ILiVgZ2t4JU1

Welcome_To_Colab (2).ipynb ☆

File Edit View Insert Runtime Tools Help   All changes saved

+ Code   + Text

```
[22] #row
```

```
    df_values=['negative','compensated hypothyroid','primary thyroid']
    df= df[df['target'].isin(df_values)]
    df['target'].value_counts()
```

```
target
negative                    4427
compensated hypothyroid      154
Name: count, dtype: int64
```

```
[24] df
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	TT4_measured	TT4	T4U_measured
0	35	F	f	f	f	f	f	f	f	f	...	f	NaN	f
1	63	M	f	f	f	f	f	f	f	f	...	t	108	t
2	25	F	f	f	f	f	f	f	f	f	...	t	61	t
3	53	F	f	f	f	f	f	f	f	t	...	t	145	t
4	92	F	f	f	f	f	f	f	f	f	...	t	120	t
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4739	30	F	f	f	f	f	f	f	f	f	...	f	NaN	f
4740	68	F	f	f	f	f	f	f	f	f	...	t	124	t

✓ 0s   completed at 1:29 PM

colab.research.google.com/drive/1MfTysxO4Gwe5AP9Nq0p3gNoWwjAipT5f#scrollTo=ILiVgZ2t4JU1

Welcome_To_Colab (2).ipynb ☆

File Edit View Insert Runtime Tools Help   All changes saved

+ Code + Text

```
[25] df['TSH']=pd.to_numeric(df['TSH'],errors='coerce')
     df['T3']=pd.to_numeric(df['T3'],errors='coerce')
     df['TT4']=pd.to_numeric(df['TT4'],errors='coerce')
     df['FTI']=pd.to_numeric(df['FTI'],errors='coerce')
     df['T4U']=pd.to_numeric(df['T4U'],errors='coerce')
```

```
df['TSH']
```

```
0        NaN
1        3.50
2        4.60
3        0.25
4        0.70
         ...
4739     NaN
4740     1.00
4741     5.10
4742     0.70
4743     1.00
Name: TSH, Length: 4581, dtype: float64
```

```
[27] df['TSH']=impute1.fit_transform(df[['TSH']])
     df['TSH']
```

```
0        1.30
1        3.50
2        4.60
```

✓ 0s   completed at 1:29 PM

```
[26] df['TSH']
```

```
0        NaN
1        3.50
2        4.60
3        0.25
4        0.70
         ...
4739     NaN
4740     1.00
4741     5.10
4742     0.70
4743     1.00
Name: TSH, Length: 4581, dtype: float64
```

```
[27] df['TSH']=impute1.fit_transform(df[['TSH']])
     df['TSH']
```

```
0        1.30
1        3.50
2        4.60
3        0.25
4        0.70
         ...
4739     1.30
4740     1.00
4741     5.10
4742     0.70
4743     1.00
Name: TSH, Length: 4581, dtype: float64
```

✓ 0s   completed at 1:29 PM

32

Welcome_To_Colab (2).ipynb

File Edit View Insert Runtime Tools Help   All changes saved

+ Code   + Text

```python
df['TSH'].unique()
```

```
array([1.30e+00, 3.50e+00, 4.60e+00, 2.50e-01, 7.00e-01, 8.10e-01,
       1.20e+00, 2.70e+01, 2.80e+00, 2.60e+00, 4.40e+00, 3.10e+00,
       1.10e+00, 4.50e+00, 1.40e-01, 6.00e-02, 7.40e+00, 2.30e+00,
       2.00e+00, 5.80e+00, 7.80e-01, 4.00e+00, 1.00e+00, 5.20e-01,
       4.00e-01, 3.00e-01, 1.00e-01, 1.90e+00, 1.70e+01, 2.00e-01,
       1.50e-02, 1.20e+01, 1.50e+00, 2.90e+00, 5.00e-02, 4.20e-01,
       1.40e+00, 5.00e+00, 5.00e-01, 9.00e-01, 2.10e+00, 1.50e-01,
       1.80e+00, 7.60e+00, 1.60e+00, 7.90e-01, 3.70e+00, 7.70e-01,
       6.40e-01, 2.70e+00, 9.80e-01, 2.50e-02, 9.30e-01, 5.80e-01,
       3.40e-01, 3.40e+00, 8.60e-01, 4.70e-01, 1.70e+00, 9.50e-01,
       3.40e+01, 5.00e-03, 1.44e+01, 4.80e+00, 8.20e-01, 2.80e+01,
       4.40e-01, 2.40e+00, 3.20e+00, 5.80e+01, 7.40e-01, 1.00e-02,
       6.70e-01, 2.10e+01, 8.00e-01, 2.00e-02, 1.65e+02, 4.20e+00,
       8.90e+00, 7.00e+00, 4.70e+00, 5.40e+00, 4.50e-01, 3.00e+00,
       8.30e-01, 8.40e-01, 4.90e+00, 6.30e-01, 3.30e+00, 2.50e+00,
       9.20e+00, 2.40e+01, 3.10e+00, 9.60e+00, 2.20e+01, 7.00e-02,
       3.00e-02, 2.20e+00, 4.30e-01, 3.70e-01, 9.50e+00, 6.00e-01,
       9.40e-01, 8.00e-01, 6.50e-01, 5.90e+00, 4.50e-02, 5.60e-01,
       4.00e-02, 1.60e+02, 3.90e-01, 6.60e-01, 2.10e-01, 5.30e-01,
       5.70e-01, 1.45e+02, 2.50e-01, 3.90e+00, 3.05e+01, 9.90e-01,
       2.00e+01, 3.50e-02, 1.90e+01, 5.90e-01, 5.10e+00, 7.90e+00,
       5.10e-01, 7.80e+00, 6.10e-01, 2.80e-01, 3.60e+00, 4.10e+00,
       3.20e-01, 4.60e-01, 9.70e-01, 3.60e-01, 6.30e+00, 4.70e+01,
       6.80e-01, 5.70e+00, 3.80e+00, 6.20e+00, 6.70e+00, 8.50e-01,
       3.30e-01, 1.80e-01, 2.60e-01, 1.80e+01, 8.90e-01, 9.60e-01,
       6.90e-01, 5.00e+01, 6.50e+00, 7.50e-01, 2.36e+02, 9.20e-01,
       1.40e+01, 2.30e+01, 1.00e+01, 4.40e+02, 9.10e-01, 6.00e+00,
       1.39e+02, 7.30e-01, 8.00e+00, 7.60e-01, 5.30e+02, 1.90e-01,
       7.50e+00, 5.50e-01, 5.30e-01, 3.50e-01, 1.60e+01, 2.60e+01,
       1.20e-01, 1.09e+02, 3.00e+01, 8.00e-02, 7.00e+01, 4.30e+00,
       2.40e-01, 3.10e-01, 6.80e+00, 4.10e-01, 1.30e+01, 8.20e+00,
```

0s completed at 1:29PM

```
       3.80e-01, 9.90e+00, 9.70e+00, 5.40e-01, 5.50e+00, 9.30e+00,
       1.30e-01, 8.30e+00, 2.30e-01, 2.70e-01, 1.01e+00, 1.84e+01,
       4.10e+01, 1.70e-01, 1.21e+01, 4.30e+01, 7.70e+00, 8.40e+00,
       5.50e-02, 3.50e+01, 7.10e-01, 9.80e+00, 1.11e+01, 3.90e+01,
       7.60e+01, 3.20e+01, 2.64e+01, 4.90e-01, 3.60e+01, 1.16e+02,
       7.80e+01])
```

```python
df['T3'].unique()
```

```
array([ nan, 2.5 , 2.4 , 2.1 , 1.3 , 2.6 , 1.8 , 2.2 , 1.2 ,
       2.8 , 3.4 , 1.4 , 1.7 , 2.3 , 1.1 , 1.6 , 3.7 , 3.  ,
       1.5 , 1.9 , 2.7 , 0.8 , 4.2 , 2.  , 1.  , 4.1 , 3.2 ,
       2.9 , 0.4 , 0.7 , 3.8 , 0.6 , 0.9 , 0.3 , 3.1 , 1.44,
       3.3 , 7.6 , 3.6 , 5.7 , 4.5 , 0.2 , 0.5 , 3.5 , 4.7 ,
       3.9 , 5.5 , 4.8 , 4.3 , 0.05, 5.4 , 4.  , 5.2 , 5.  ,
       6.  , 5.3 , 4.6 , 7.3 , 6.7 , 6.1 , 0.1 , 4.9 , 10.6 ,
       5.1 , 7.  , 6.2 , 4.4 , 7.1 , 8.5 , 6.6 ])
```

```python
df['T3']=impute1.fit_transform(df[['T3']])
df['T3'].unique()
```

```
array([ 2.  , 2.5 , 2.4 , 2.1 , 1.3 , 2.6 , 1.8 , 2.2 , 1.2 ,
       2.8 , 3.4 , 1.4 , 1.7 , 2.3 , 1.1 , 1.6 , 3.7 , 3.  ,
       1.5 , 1.9 , 2.7 , 0.8 , 4.2 , 1.  , 4.1 , 3.2 , 2.9 ,
       0.4 , 0.7 , 3.8 , 0.6 , 0.9 , 0.3 , 3.1 , 1.44, 3.3 ,
       7.6 , 3.6 , 5.7 , 4.5 , 0.2 , 0.5 , 3.5 , 4.7 , 3.9 ,
       5.5 , 4.8 , 4.3 , 0.05, 5.4 , 4.  , 5.2 , 5.  , 6.  ,
       5.3 , 4.6 , 7.3 , 6.7 , 6.1 , 0.1 , 4.9 , 10.6 , 5.1 ,
       7.  , 6.2 , 4.4 , 7.1 , 8.5 , 6.6 ])
```

0s completed at 1:29PM

Screenshot 1:

```
[31]  df['TT4'].unique()
      df['TT4']=impute1.fit_transform(df[['TT4']])
      df['TT4'].unique()
```

```
array([104. , 108. ,  61. , 145. , 120. ,  84. , 117. ,  65. , 112. ,
        94. ,  95. , 131. , 105. , 148. ,  81. , 132. ,  82. ,  92. ,
       133. , 261. , 106. ,  90. , 111. ,  96. , 103. ,  66. , 162. ,
        87. , 135. ,  89. , 140. , 125. ,  68. , 178. , 114. ,  98. ,
        76. , 188. , 122. , 141. ,  88. ,  99. , 119. ,  93. , 164. ,
       210. , 123. , 160. , 110. , 100. ,  83. , 107. , 158. , 183. ,
       118. , 163. , 102. ,  69. , 137. , 161. ,  73. ,  85. , 248. ,
       152. , 155. ,  78. , 139. , 129. ,  74. ,  51. , 113. ,  97. ,
        72. ,  86. , 109. ,  75. ,  77. ,  25. , 121. , 124. , 171. ,
        53. ,  46. ,  67. , 203. ,  17. ,  71. ,  79. , 151. , 147. ,
       115. , 138. , 157. , 150. , 146. , 220. , 197. , 134. , 136. ,
       116. ,  91. , 149. ,  41. , 126. ,  45. , 179. ,   9.5, 127. ,
        52. ,  19. ,  31. , 175. , 101. , 130. , 159. , 212. ,  57. ,
        32. , 222. , 168. , 142. , 186. ,  10. , 153. ,  44. ,  70. ,
        49. ,  80. , 372. , 180. , 128. , 166. , 214. , 144. , 143. ,
        64. , 154. ,  60. ,  16. , 156. ,  56. ,  28. , 184. ,  33. ,
        24. ,  36. , 165. ,  50. ,   4. ,  35. ,   3. ,  62. ,  30. ,
        59. ,  63. ,   2.9, 174. , 172. ,  54. ,  55. ,  11. , 216. ,
        15. ,  38. , 258. , 199. , 237. ,  39. , 217. , 205. , 225. ,
       219. , 177. , 189. ,  58. , 193. , 257. , 167. , 176. , 244. ,
       187. , 250. , 181. , 223. , 272. , 213. , 235. , 231. , 191. ,
        48. , 169. ,  40. , 232. , 204. , 430. , 198. , 230. , 170. ,
       194. , 192. , 182. , 246. , 196. , 207. , 200. , 226. , 201. ,
       233. , 206. , 255. , 239. ,  22. , 195. , 289. , 240. , 209. ,
        43. , 252. , 263. , 301. , 211. , 253. , 173. , 256. , 273. ])
```

✓ 0s   completed at 1:29 PM

Screenshot 2:

```
[31]  Start coding or generate with AI.
```

```
df['T4U'].unique()
df['T4U']=impute1.fit_transform(df[['T4U']])
df['T4U'].unique()
```

```
array([0.97 , 0.96 , 0.82 , 1.03 , 0.84 , 0.83 , 1.31 , 0.99 , 0.92 ,
       0.89 , 1.11 , 0.95 , 0.5  , 1.07 , 1.23 , 1.08 , 0.87 , 0.88 ,
       1.16 , 2.32 , 1.02 , 0.69 , 0.91 , 1.   , 1.47 , 1.29 , 1.01 ,
       0.62 , 1.26 , 1.06 , 1.1  , 0.8  , 0.9  , 0.98 , 1.09 , 1.05 ,
       1.12 , 0.86 , 1.55 , 0.93 , 1.63 , 0.74 , 0.94 , 0.65 , 0.79 ,
       1.5  , 1.4  , 1.14 , 0.71 , 0.77 , 0.68 , 1.27 , 0.6  , 0.67 ,
       1.21 , 1.36 , 1.17 , 1.2  , 0.76 , 1.19 , 1.18 , 0.75 , 0.78 ,
       1.15 , 0.85 , 1.04 , 0.72 , 1.71 , 1.25 , 1.68 , 1.65 , 0.81 ,
       1.52 , 1.13 , 0.46 , 0.73 , 0.63 , 1.24 , 1.82 , 1.57 , 0.64 ,
       1.41 , 1.7  , 0.48 , 0.7  , 0.54 , 1.42 , 1.22 , 1.75 , 1.53 ,
       1.45 , 1.51 , 1.32 , 1.54 , 1.48 , 0.47 , 0.59 , 1.35 , 1.83 ,
       1.38 , 1.3  , 1.34 , 0.36 , 2.01 , 1.28 , 0.25 , 0.56 , 1.58 ,
       1.33 , 0.53 , 1.44 , 0.61 , 1.43 , 1.46 , 1.66 , 0.52 , 1.39 ,
       1.93 , 0.58 , 1.67 , 0.66 , 1.77 , 1.59 , 1.97 , 1.69 , 1.74 ,
       2.03 , 1.73 , 0.57 , 1.84 , 1.37 , 1.79 , 1.8  , 1.62 , 1.76 ,
       1.56 , 0.31 , 1.94 , 2.12 , 0.944, 0.49 , 1.88 , 0.38 , 1.49 ,
       0.41 , 1.61 ])
```

```
[33]  df['FTI']=impute1.fit_transform(df[['FTI']])
      df['FTI'].unique()
```

```
array([107. , 113. ,  75. , 141. , 143. , 101. ,  90. ,  66. , 121. ,
       106. ,  85. , 138. , 265. , 118. , 100. ,  94. , 152. ,  92. ,
       111. , 115. ,  78. , 109. , 140. , 114. ,  88. ,  65. , 110. ,
```

✓ 0s   completed at 1:29 PM

Welcome_To_Colab (2).ipynb ☆

File Edit View Insert Runtime Tools Help   All changes saved

+ Code   + Text

```
<class 'pandas.core.frame.DataFrame'>
Index: 4581 entries, 0 to 4743
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   age                        4581 non-null   int64
 1   sex                        4581 non-null   object
 2   on_thyroxine               4581 non-null   object
 3   query_on_thyroxine         4581 non-null   object
 4   on_antithyroid_medication  4581 non-null   object
 5   sick                       4581 non-null   object
 6   pregnant                   4581 non-null   object
 7   thyroid_surgery            4581 non-null   object
 8   I131_treatment             4581 non-null   object
 9   query_hypothyroid          4581 non-null   object
 10  query_hyperthyroid         4581 non-null   object
 11  lithium                    4581 non-null   object
 12  goitre                     4581 non-null   object
 13  tumor                      4581 non-null   object
 14  hypopituitary              4581 non-null   object
 15  psych                      4581 non-null   object
 16  TSH_measured               4581 non-null   object
 17  TSH                        4581 non-null   float64
 18  T3_measured                4581 non-null   object
 19  T3                         4581 non-null   float64
 20  TT4_measured               4581 non-null   object
 21  TT4                        4581 non-null   float64
 22  T4U_measured               4581 non-null   object
 23  T4U                        4581 non-null   float64
 24  FTI_measured               4581 non-null   object
 25  FTI                        4581 non-null   float64
 26  target                     4581 non-null   object
dtypes: float64(5), int64(1), object(21)
memory usage: 1002.1+ KB
```
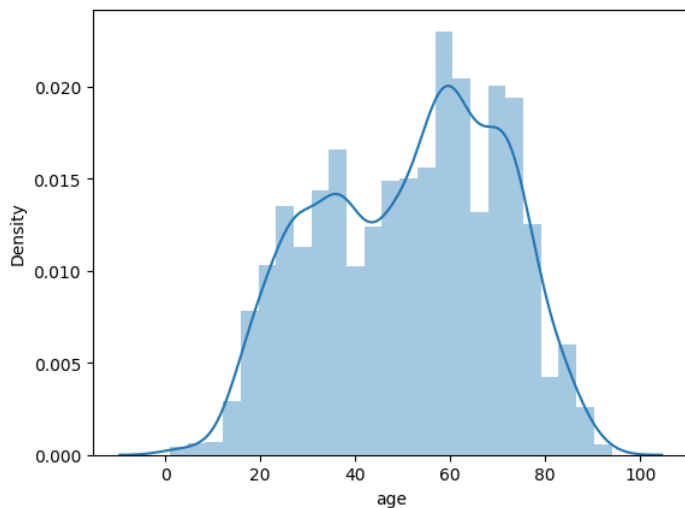
✓ 0s   completed at 1:29PM
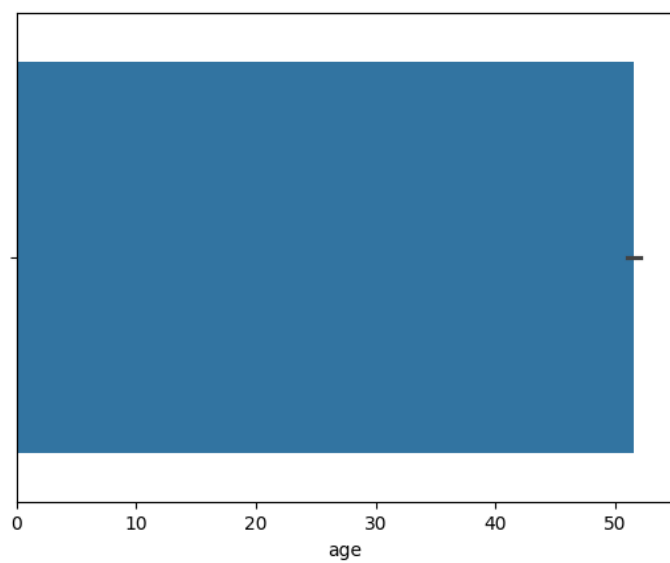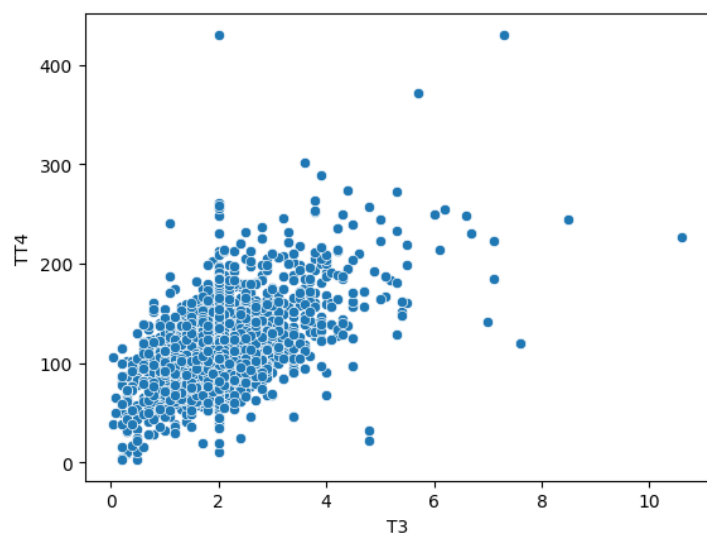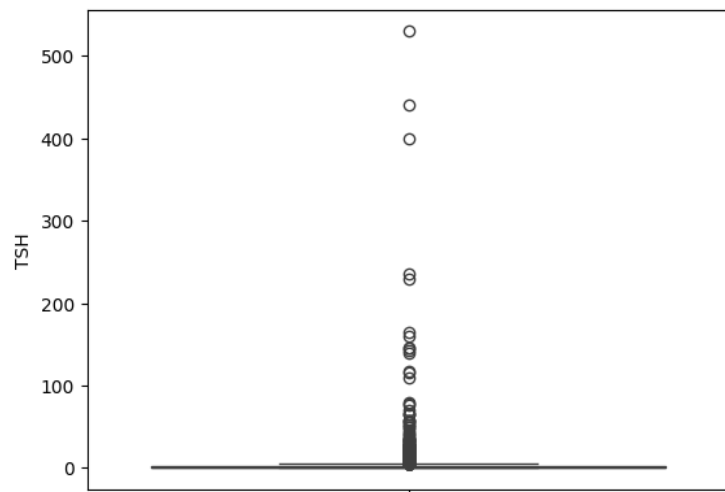
```python
import matplotlib.pyplot as plt

# 1. Distribution of age
sns.distplot(df['age'])
plt.show()

# 2. Boxplot of TSH
sns.boxplot(df['TSH'])
plt.show()

# 3. Scatter plot of T3 vs TT4
sns.scatterplot(x='T3', y='TT4', data=df)
plt.show()

# 4. Bar plot of target variable
sns.barplot(x='age', data=df)
plt.show()
```
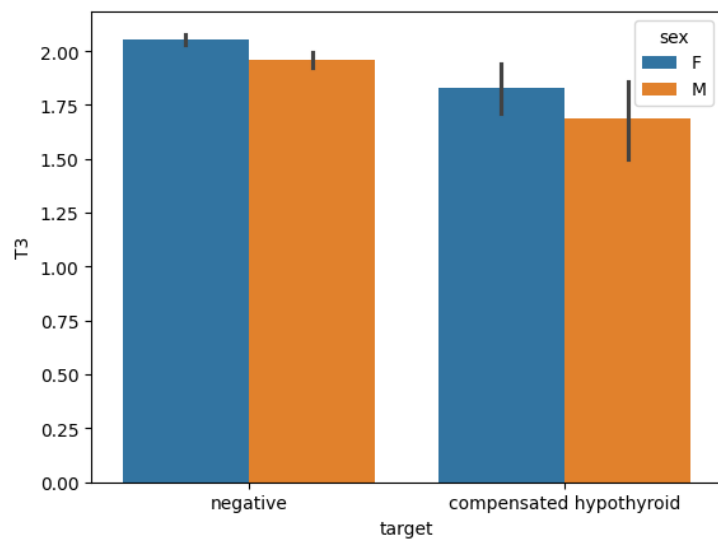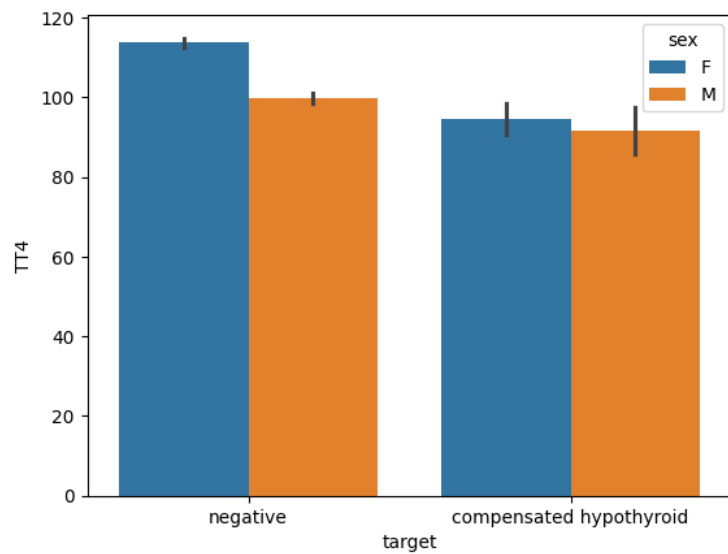
```
# prompt: visualizing between t3 and target

sns.barplot(x='target', y='T3', data=df,hue='sex')
plt.show()
```
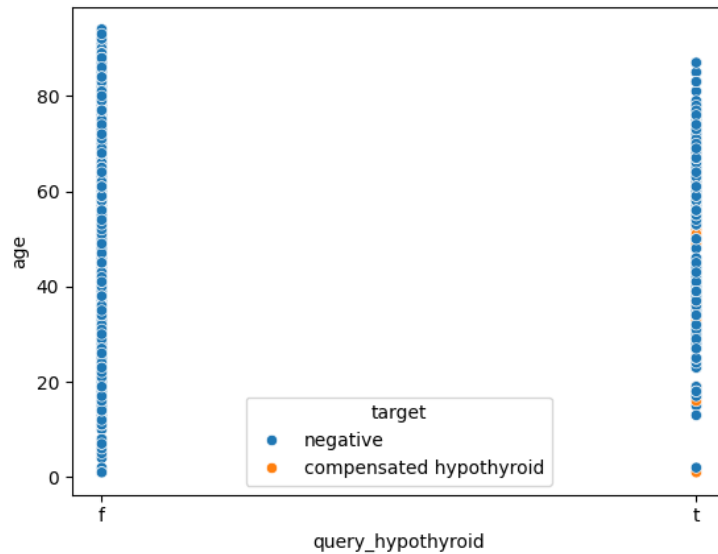


```
# prompt: graph between tt4 and target by coluring sex

sns.barplot(x='target', y='TT4', data=df, hue='sex')
plt.show()
```

```
# prompt: plot graph between query_thyroid and target

sns.scatterplot(x='query_hypothyroid', y='age', data=df, hue='target')
plt.show()
```

Welcome_To_Colab (2).ipynb

File Edit View Insert Runtime Tools Help   All changes saved

+ Code + Text

```
  9  on_thyroxine_f             4581 non-null   bool
 10  on_thyroxine_t             4581 non-null   bool
 11  on_antithyroid_medication_f 4581 non-null   bool
 12  on_antithyroid_medication_t 4581 non-null   bool
 13  sick_f                     4581 non-null   bool
 14  sick_t                     4581 non-null   bool
 15  pregnant_f                 4581 non-null   bool
 16  pregnant_t                 4581 non-null   bool
 17  thyroid_surgery_f          4581 non-null   bool
 18  thyroid_surgery_t          4581 non-null   bool
 19  I131_treatment_f           4581 non-null   bool
 20  I131_treatment_t           4581 non-null   bool
 21  query_on_thyroxine_f       4581 non-null   bool
 22  query_on_thyroxine_t       4581 non-null   bool
 23  query_hypothyroid_f        4581 non-null   bool
 24  query_hypothyroid_t        4581 non-null   bool
 25  query_hyperthyroid_f       4581 non-null   bool
 26  query_hyperthyroid_t       4581 non-null   bool
 27  lithium_f                  4581 non-null   bool
 28  lithium_t                  4581 non-null   bool
 29  goitre_f                   4581 non-null   bool
 30  goitre_t                   4581 non-null   bool
 31  tumor_f                    4581 non-null   bool
 32  tumor_t                    4581 non-null   bool
 33  hypopituitary_f            4581 non-null   bool
 34  hypopituitary_t            4581 non-null   bool
 35  psych_f                    4581 non-null   bool
 36  psych_t                    4581 non-null   bool
 37  TSH_measured_f             4581 non-null   bool
 38  TSH_measured_t             4581 non-null   bool
 39  T3_measured_f              4581 non-null   bool
 40  T3_measured_t              4581 non-null   bool
 41  TT4_measured_f             4581 non-null   bool
```

✓ 0s   completed at 1:29 PM

```
 20  I131_treatment_t           4581 non-null   bool
 21  query_on_thyroxine_f       4581 non-null   bool
 22  query_on_thyroxine_t       4581 non-null   bool
 23  query_hypothyroid_f        4581 non-null   bool
 24  query_hypothyroid_t        4581 non-null   bool
 25  query_hyperthyroid_f       4581 non-null   bool
 26  query_hyperthyroid_t       4581 non-null   bool
 27  lithium_f                  4581 non-null   bool
 28  lithium_t                  4581 non-null   bool
 29  goitre_f                   4581 non-null   bool
 30  goitre_t                   4581 non-null   bool
 31  tumor_f                    4581 non-null   bool
 32  tumor_t                    4581 non-null   bool
 33  hypopituitary_f            4581 non-null   bool
 34  hypopituitary_t            4581 non-null   bool
 35  psych_f                    4581 non-null   bool
 36  psych_t                    4581 non-null   bool
 37  TSH_measured_f             4581 non-null   bool
 38  TSH_measured_t             4581 non-null   bool
 39  T3_measured_f              4581 non-null   bool
 40  T3_measured_t              4581 non-null   bool
 41  TT4_measured_f             4581 non-null   bool
 42  TT4_measured_t             4581 non-null   bool
 43  T4U_measured_f             4581 non-null   bool
 44  T4U_measured_t             4581 non-null   bool
 45  FTI_measured_f             4581 non-null   bool
 46  FTI_measured_t             4581 non-null   bool
dtypes: bool(40), float64(5), int64(1), object(1)
memory usage: 465.3+ KB
```

[44] #spliting

✓ 0s   completed at 1:29 PM

CO Welcome_To_Colab (2).ipynb ☆
File Edit View Insert Runtime Tools Help   All changes saved

+ Code + Text

```python
[44] #spliting
```

```python
[45] x=df.drop(['target'],axis=1)
     y=df['target']
     df=df.drop_duplicates()
```

Start coding or generate with AI.

```python
[46] from sklearn.model_selection import train_test_split
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```python
[47] !pip install imblearn --quiet
     from imblearn.over_sampling import SMOTE
```

```python
[48] from sklearn.preprocessing import StandardScaler
     sc=StandardScaler()
     x_train=sc.fit_transform(x_train)
     x_test=sc.transform(x_test)
```

```python
[49] df['target'].value_counts()
```

```
target
negative                3488
```

✓ 0s   completed at 1:29 PM

CO Welcome_To_Colab (2).ipynb ☆
File Edit View Insert Runtime Tools Help   All changes saved

+ Code + Text

```python
[49] df['target'].value_counts()
```

```
target
negative                  3488
compensated hypothyroid    154
Name: count, dtype: int64
```

```python
[49] Start coding or generate with AI.
```

```python
[49] Start coding or generate with AI.
```

```python
[50] os=SMOTE(random_state=42,k_neighbors=5)
     x_train,y_train=os.fit_resample(x_train,y_train)
```

```python
[51] y_train.value_counts()
```

```
target
negative                  3534
compensated hypothyroid   3534
Name: count, dtype: int64
```

```python
[52] from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
     from sklearn.model_selection import GridSearchCV
```

✓ 0s   completed at 1:29 PM

40

```
[52] from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
     from sklearn.model_selection import GridSearchCV
```

```
[53] '''xgb=XGBClassifier(random_state=42,max_depth=4,n_estimators=100,bootstrap=True,max_leaf_nodes=10,objective='multi:softmax', num_class=len(set(y_train)))
     xgb.fit(x_train,y_train)'''
```

```
'xgb=XGBClassifier(random_state=42,max_depth=4,n_estimators=100,bootstrap=True,max_leaf_nodes=10,objective='multi:softmax', num_class=len(set(y_train)))\nxgb.fit(x_train,y_train)'
```

```
[54] '''rf=RandomForestClassifier(random_state=42,'bootstrap': False, 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50)
     rf.fit(x_train,y_train)'''
```

```
'rf=RandomForestClassifier(random_state=42,'bootstrap': False, 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50)\nrf.fit(x_train,y_train)'
```

```
[54] Start coding or generate with AI.
```

```
[55] rf = RandomForestClassifier(random_state=42, bootstrap=False, max_depth=None,
                                  max_features='sqrt', min_samples_leaf=2,
                                  min_samples_split=2, n_estimators=100)
     rf.fit(x_train,y_train)
```

```
        RandomForestClassifier
RandomForestClassifier(bootstrap=False, min_samples_leaf=2, random_state=42)
```

```
rf = RandomForestClassifier(random_state=42, bootstrap=False, max_depth=None,
                            max_features='sqrt', min_samples_leaf=2,
                            min_samples_split=2, n_estimators=100)
rf.fit(x_train,y_train)
```

```
        RandomForestClassifier
RandomForestClassifier(bootstrap=False, min_samples_leaf=2, random_state=42)
```

```
[56] x_pred=rf.predict(x_train)
     y_pred=rf.predict(x_test)
```

```
[56] Start coding or generate with AI.
```

```
[57]
     print(classification_report(y_test,y_pred))
```

```
                          precision    recall  f1-score   support

compensated hypothyroid       0.84      0.88      0.86        24
               negative       1.00      1.00      1.00       893

               accuracy                           0.99       917
              macro avg       0.92      0.94      0.93       917
           weighted avg       0.99      0.99      0.99       917
```

colab.research.google.com/drive/1MfTysxO4Gwe5AP9Nq0p3gNoWwjAipT5f#scrollTo=D4wAHjaE3VUa

Welcome_To_Colab (2).ipynb ☆

File   Edit   View   Insert   Runtime   Tools   Help   All changes saved

💬 Comment   👥 Share ⚙

+ Code + Text

```
[60] train_accuracy = accuracy_score(y_train, x_pred)
     print(f'Training Accuracy: {train_accuracy * 100:.2f}%')

     # Calculate accuracy for testing set
     test_accuracy = accuracy_score(y_test, y_pred)
     print(f'Testing Accuracy: {test_accuracy * 100:.2f}%')
```

```
Training Accuracy: 99.86%
Testing Accuracy: 99.24%
```

[60] Start coding or generate with AI.

[60] Start coding or generate with AI.

[60] Start coding or generate with AI.

```
[61] import pandas as pd

     # Define the data as a list
     data = [
         [65, 'M', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 'f', 't', '14.8', 't', '1.5', 't', '61', 't', '0.85', 't', '72', 'f']
     ]

     # Define column names based on the structure you provided
     columns = [
         'age', 'sex', 'on_thyroxine', 'on_antithyroid_medication', 'sick', 'pregnant',
         'thyroid_surgery', 'I131_treatment', 'query_on_thyroxine', 'query_hypothyroid',
```

✓ 0s   completed at 1:29 PM