Thyroid Prediction Using Machine Learning Techniques

Machine Learning (22AIE213)

Submitted by:

K. Sachin (AV.EN. U4AIE22124)

K. Shanmukha Balaji (AV. EN. U4AIE22125)

K. Rupesh Sai Ram (AV. EN. U4AIE22126)

L. Raghu Ram Sai (AV. EN. U4AIE22127)

Guided By

DR. MADHUSUDHANA RAO

[CSE ASC-AMARAVATHI]

Abstract:

Thyroid conditions can result from elevated TSH levels or thyroid organ infections. Machine learning (ML) algorithms, such as Decision trees, Random forest, Support vector machines, Artificial Neural Networks, and Logistic regression, have been applied to predict and diagnose thyroid diseases. This paper reviews recent ML techniques used for thyroid prediction based on symptoms and lab reports. The proposed system employs multiple ML methods for improved accuracy in thyroid disease prediction. Among these, the Decision tree algorithm shows the best performance with an accuracy of 99.46%.

Introduction:

Thyroid disease is a prevalent health issue affecting millions of people worldwide. The thyroid gland, located in the neck, produces hormones that regulate various functions in the body, including metabolism, growth, and development. However, when the thyroid gland fails to function correctly, it can lead to an excess production of thyroid hormones, causing a range of symptoms and health complications.

According to recent estimates, one out of every 38,000 individuals worldwide is affected by thyroid disease on average. However, in developing countries like India, the prevalence rate is much higher, with approximately 42 million people suffering from this condition. In some regions, such as Mumbai, the ratio can be as high as one person out of every 2640.

Thyroid disease can cause a range of symptoms that impact an individual's quality of life. These include fatigue, weight gain, low energy levels, intolerance to cold, dry skin, slow heart rate, and swelling in the neck. In severe cases, the body goes into an "auto safe mode," generating hormones that can damage the thyroid organs irreversibly.

The impact of thyroid disease on individuals can be significant, both physically and emotionally. Early detection and prevention are crucial as the disease can be challenging to cure once it reaches advanced stages. Traditional methods for detecting and treating thyroid disease involve clinical and medical studies using classical analysis and statistical tests. However, given the widespread nature of this condition, there is a need for more innovative approaches to improve detection, prevention, and Treatment.

One potential solution is the use of technology, such as machine learning algorithms, to analyze patient data and identify patterns that may indicate the presence of thyroid disease. Another approach is public health campaigns and awareness initiatives to promote early screening and healthy lifestyle choices. These efforts can help reduce the prevalence of thyroid disease by identifying cases earlier and preventing new ones from Developing.

Literature Survey Table:

We prepared a table below that contains literature survey based on various other research papers we found online. The table includes Authors, the year it published, models used and results.

Authors	Year	Model	Evaluation Metrics	Results
Chaganti et al. [1]	2022	RF, GBM, ADA, LR, SVM	Accuracy, Precision, Recall, F1 Score	RF achieved the highest performance with 0.99 for all evaluation metrics.

Tahir Alyas et al. [2]	2022	DT, RF, KNN, ANN, NB	Accuracy, Sensitivity, Specificity	RF achieved the highest accuracy of 94.8% and a specificity of 91%.
Ji [3]	2024	RF, GBM, ADA, LR, SVC, LSTM, CNN, CNN-LSTM, SVEC-H, SVEC-S	Accuracy, Precision, Recall, F1-score	Proposed SSC (Self-Stacking Classifier) model with RF as base and meta-learner achieved the highest accuracy.
Mir and Mittal [4]	N/A	SVM, NB, J48, Bagging, Boosting	Accuracy, Sensitivity, Specificity, Precision	Bagging achieved the highest accuracy of 98.56% on the primary dataset.
Aversano et al. [5]	2021	DTC, NBC, KNNC, RFC, EXTC, MLPC, XGBC, CBC, ABC, GBC	Accuracy, Precision, Recall, F1-score	Extra-Tree Classifier (EXTC) showed the highest accuracy reaching 84% on the discretized and balanced dataset.
Chaubey et al. [6]	2020	LR, DT, NB	Not specified	Study focused on providing a reference for future research in thyroid disease prediction.
Ritesh Jha et al. [7]	2021	PCA, SVD, DT	Accuracy, precision, recall, f1-score	Used data Augmentation to create 10, 000 samples. It showed the results go as high as 98.75 in DT.
Aravind Selwal et al [8]	2019	MLP	Accuracy	Collected a real dataset from SKIMS Soura. Used cross validation and achieved result of 99.2% with 7 attributes.
Emmanuel Togor at al [9]	2023	SVM, RF	Accuracy, recall, precision, f1-score	Predicted Types of Thyroid such as hyperthyroid, hypothyroid and Euthyroid. Used feature extraction and achieved result of 99.3%.

Marissa at el [10]	2019	MLP	Accuracy	In This paper they used two classifiers, first one to predict the types of thyroid. Then the samples with hyperthyroid is given to another classifier for another classification.

Literature Survey :

The research paper by Rao and Razia [1] showcased almost all the ML techniques with their basic structure. Out of all the algorithms, ANN showed best result. The major limitation of this method was it didn't explored the Genetic Algorithm for better optimized result.

Prerana et al. [2] proposed the technique of data mining using neural networks that can be used for early prediction of thyroid. The network was trained using back propagation and gradient method working simultaneously. But the variation in layers of various network parameters were not considered during training.

Umadevi et al. [3] worked on the trial of 21 parameters to train the model using classification algorithm. The model was trained using KNN, ANN and fuzzy ANN algorithm and the accuracy was compared. It was evident that Fuzzy ANN performed better than other two classification algorithms. But it is observed that due to few samples for dis-functions of thyroid, the classification between over and under functioning thyroid was difficult.

Ammulu and Venugopal [4] have proposed data mining technique to predict hypothyroidism in the patient. In the research paper, data mining technique is applied on the hypothyroid dataset to determine the positive and the negative cases from the entire dataset. But, the algorithm works only for under functioning thyroid and thus nothing can be said about hyperthyroidism.

Ahmed et al. [5] provides Support vector machine (multi, binary) algorithm for thyroid prediction. The precision value along with confusion matrix was used for evaluation of results. Medical data cleaning was used for filling all the blank spaces. When thyroid disease goes through structural changes it becomes difficult to detect it based on variations of thyroid hormones.

Mahajan et al. [6] have provided a way to detect hypo and hyperthyroid from thermal images using Bayesian classifier. It provided 81.18% accuracy in classification. The major drawback of this algorithm was if the image were not cleared or not processed properly, the results shown differed with a high range than the actual output.It provides thyroid diagnosis at early stages and attains higher accuracy.

Saiti et al. [7] have proposed thyroid prediction using PNN and support vector machine. Feature selection was done using genetic algorithms. The fitness evaluation contained two terms: (1) accuracy and (2) the number of features selected. When the algorithms were tested without the GA, the accuracy of both the methods was around 85%, but with the help of the GA accuracy obtained was nearly 100%. The proposed system will even help the new practitioners to improve their analysis skills and predict the disease even without prior knowledge about it. The primary task is to provide thyroid diagnosis at early stages and also attain higher accuracy

Sunila Godara. [8] They have used Logistics Regression and SVM machine learning Technique to analyze Thyroid Dataset. Comparison was made between these two algorithms based on Precision, Recall, F measure, ROC, RMS error. Logistic Regression turned out has best classifier.

Umar Sidiq, Dr, Syed Mutahar Aaqib, and Rafi Ahmad Khan [9] 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.

VijiyaKumar, K., et al [10] 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

Research Gaps and Proposed solutions:

Research Gaps:

1). Imbalanced Data:

Gap: Many machine learning models perform poorly on imbalanced datasets because they tend to be biased towards the majority class. This can lead to inaccurate predictions and a lack of generalizability.
Proposed Solution: To address this, we implemented several steps to handle the imbalance. Initially, we removed null values based on a threshold to ensure that some amount of minor class data was retained, thus slightly improving the balance between classes.

2). Handling Null Values:

Gap: Null values in the dataset can lead to incomplete analysis and poor model performance if not handled properly. Simply removing null values can sometimes lead to significant loss of data, particularly for minor classes.
Proposed Solution: We set a threshold for removing null values to minimize data loss from the minor class, which helped in retaining more representative data for training.

3). Outlier Detection and Removal:

Gap: Outliers can skew the results and negatively impact the performance of machine learning models. Identifying and handling outliers is crucial for creating a robust model.
Proposed Solution: We applied the Interquartile Range (IQR) method to detect and remove outliers. This method helps in improving the dataset quality by

ensuring that the data is within a reasonable range, which aids in better model training.

4). Data Imbalance Handling with SMOTE:

Gap: While handling imbalanced data is critical, traditional methods like undersampling and oversampling have their limitations, such as loss of valuable data or overfitting.
Proposed Solution: We utilized Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for the minority class. This approach helped in balancing the dataset without losing information from the majority class, thus enhancing the model's ability to generalize better.

5). Feature Selection:

Gap: Not all features contribute equally to the predictive power of a model. Irrelevant or redundant features can lead to overfitting and increased computational cost.
Proposed Solution: We performed feature selection using a correlation matrix to identify and retain only the most relevant features. This step helped in reducing the dimensionality of the dataset, improving model performance, and speeding up the training process.

6). Model Validation:

Gap: Single train-test splits can lead to biased evaluation results due to the randomness of the split. It does not provide a reliable measure of model performance across different subsets of data.
Proposed Solution: We employed k-fold cross-validation to ensure that the model performance is consistent and generalizable across different data splits. This method divides the dataset into k subsets, trains the model on k-1 subsets, and tests it on the remaining subset, repeating this process k times to provide a robust evaluation metric.

Materials and Methods:

a). Details of the Algorithms Processed:

Random Forests

Purpose: Used for classification due to its ability to handle large datasets with higher dimensionality and its robustness against overfitting.

Process:

- **Data Preparation:** Clean and preprocess the data, handling missing values and encoding categorical variables.
- **Tree Construction:** Build multiple decision trees using different subsets of the data and features.
- **Voting Mechanism:** Aggregate the predictions from all trees to determine the final classification output.

Decision Trees

Purpose: Employed for its simplicity and interpretability in classification tasks.

Process:

- **Data Preparation:** Similar preprocessing to Random Forests.
- **Tree Construction:** Split the data at each node based on the feature that provides the maximum information gain.
- **Leaf Nodes:** Assign class labels based on the majority class of the samples at each leaf.

Feedforward Neural Networks

Purpose: Used for classification tasks due to its ability to model complex non-linear relationships.

Process:

- **Input Layer:** Accepts the input features of the dataset.
- **Hidden Layers:** One or more layers with neurons that apply weighted sums and activation functions.

- **Activation Function (ReLU):** Introduces non-linearity.
- **Output Layer:** Produces the final class probabilities.

Logistic Regression

Purpose: Applied for binary classification due to its simplicity and efficiency.

- **Process:**
- **Data Preparation:** Standardize the input features.
- **Model Training:** Fit the logistic regression model to the training data.
- **Prediction:** Calculate the probability of the positive class and apply a threshold to determine the final class.

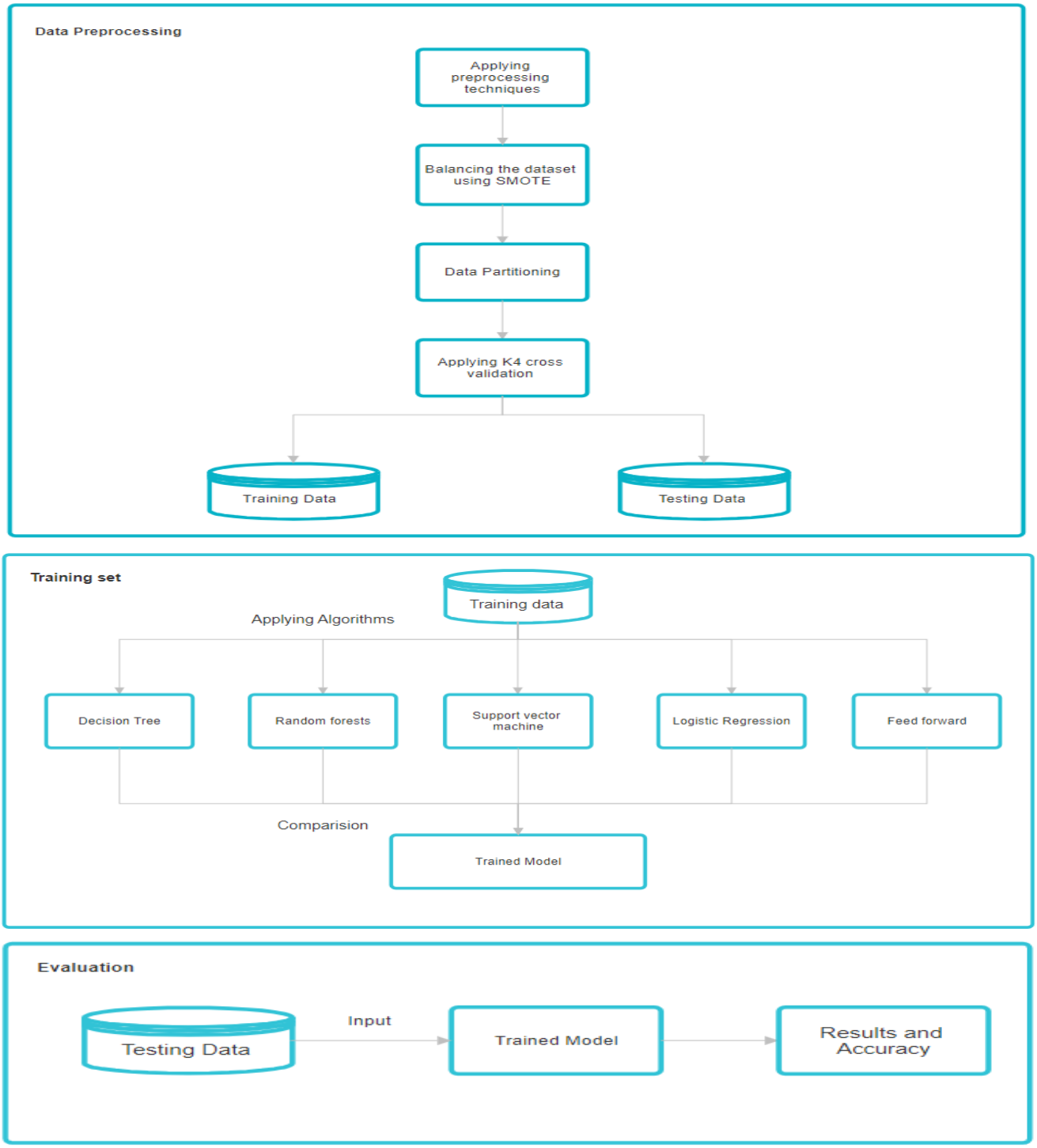
Support Vector Machines (SVM)

Purpose: Used for classification by finding the optimal hyperplane that maximizes the margin between classes.

Process:

- **Data Preparation:** Standardize the input features.
- **Kernel Selection:** Choose a kernel function (e.g., linear, RBF) based on the data characteristics.
- **Model Training:** Fit the SVM model to the training data.
- **Prediction:** Classify new instances based on which side of the hyperplane they fall on.

b). Flow Chart



c). Python Program:

Dependencies and libraries:

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix
from imblearn.over_sampling import SMOTE
```

Data Preprocessing:

1).Handling Missing Values and Splitting Data:

The first step involves handling missing values and splitting the data into features (X) and the target variable (y).

```
# Drop the target column to create feature set X
x = model_df.drop('target', axis=1)

# Separate the target variable
y = model_df['target']

# Impute missing values using the mean strategy
imputer = SimpleImputer(strategy='mean')
x = imputer.fit_transform(x)
```

2).Handling Class Imbalance with SMOTE:

Synthetic Minority Over-sampling Technique (SMOTE) is used to address class imbalance by generating synthetic samples.

```
# Apply SMOTE to handle class imbalance
x_smot, y_smot = SMOTE(k_neighbors=2).fit_resample(x, y)
```

3). K-Fold Cross-Validation:

K-Fold Cross-Validation is used to evaluate the model performance by splitting the data into 4 folds.

```
kf = KFold(n_splits=4, shuffle=True, random_state=42)

# Iterate through each split
for train_index, test_index in kf.split(x):
    # Split data into training and testing sets based on the K-Fold split
    x_train, x_test = x[train_index], x[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

Model Training and Evaluation

1). Support Vector Machine (SVM):

Train and evaluate a Support Vector Machine with a linear kernel.

```
# Initialize and train the SVM model
SVM = svm.SVC(kernel='linear')
SVM.fit(x_train, y_train)

# Make predictions
y_pred = SVM.predict(x_test)

# Calculate and print evaluation metrics
svm_accuracy = accuracy_score(y_test, y_pred)
svm_precision = precision_score(y_test, y_pred, average='weighted')
svm_recall = recall_score(y_test, y_pred, average='weighted')
svm_f1 = f1_score(y_test, y_pred, average='weighted')

print(f'SVM Accuracy: {svm_accuracy}')
print(f'SVM Precision: {svm_precision}')
print(f'SVM Recall: {svm_recall}')
print(f'SVM F1 Score: {svm_f1}')
```

2). Decision Tree Classifier:

Train and evaluate a Decision Tree Classifier.

```
# Initialize and train the Decision Tree model
dt = DecisionTreeClassifier(random_state=42)
dt.fit(x_train, y_train)

# Make predictions
y_pred = dt.predict(x_test)
```

```
# Calculate and print evaluation metrics
dt_accuracy = accuracy_score(y_test, y_pred)
dt_precision = precision_score(y_test, y_pred, average='weighted')
dt_recall = recall_score(y_test, y_pred, average='weighted')
dt_f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Decision Tree Accuracy: {dt_accuracy}')
print(f'Decision Tree Precision: {dt_precision}')
print(f'Decision Tree Recall: {dt_recall}')
print(f'Decision Tree F1 Score: {dt_f1}')
```

3). Random Forest Classifier:

Train and evaluate a Random Forest Classifier.

```
# Initialize and train the Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(x_train, y_train)

# Make predictions
y_pred = rf.predict(x_test)

# Calculate and print evaluation metrics
rf_accuracy = accuracy_score(y_test, y_pred)
rf_precision = precision_score(y_test, y_pred, average='weighted')
rf_recall = recall_score(y_test, y_pred, average='weighted')
rf_f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Random Forest Accuracy: {rf_accuracy}')
print(f'Random Forest Precision: {rf_precision}')
print(f'Random Forest Recall: {rf_recall}')
print(f'Random Forest F1 Score: {rf_f1}')
```

4).Logistic Regression:

Train and evaluate Logistic regression model.

```
# Initialize and train the Logistic Regression model
log_reg = LogisticRegression(random_state=42, solver='saga')
log_reg.fit(x_train, y_train)
```

```
# Make predictions
y_pred = log_reg.predict(x_test)

# Calculate and print evaluation metrics
lr_accuracy = accuracy_score(y_test, y_pred)
lr_precision = precision_score(y_test, y_pred, average='weighted')
lr_recall = recall_score(y_test, y_pred, average='weighted')
lr_f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Logistic Regression Accuracy: {lr_accuracy}')
print(f'Logistic Regression Precision: {lr_precision}')
print(f'Logistic Regression Recall: {lr_recall}')
print(f'Logistic Regression F1 Score: {
```

5). Multi-Layer Perceptron (MLP):

Train and evaluate a Multi-Layer Perceptron (Neural Network)

```
# Initialize and train the MLP model
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300, random_state=42)
mlp.fit(x_train, y_train)

# Make predictions
y_pred = mlp.predict(x_test)

# Calculate and print evaluation metrics
mlp_accuracy = accuracy_score(y_test, y_pred)
mlp_precision = precision_score(y_test, y_pred, average='weighted')
mlp_recall = recall_score(y_test, y_pred, average='weighted')
mlp_f1 = f1_score(y_test, y_pred, average='weighted')

print(f'MLP Accuracy: {mlp_accuracy}')
print(f'MLP Precision: {mlp_precision}')
print(f'MLP Recall: {mlp_recall}')
print(f'MLP F1 Score: {mlp_f1}')
```

Results and Discussion:

a). Description of the Proposed Datasets:

The dataset used in this project is the Thyroid Disease dataset obtained from Kaggle. This dataset is designed to predict whether a patient has a thyroid disease based on various medical features. The dataset includes the following attributes

thyroidDF.csv - 9172 observations x 31 attributes

1. age - age of the patient (int)
2. sex - sex patient identifies (str)
3. on_thyroxine - whether patient is on thyroxine (bool)
4. query on thyroxine - *whether patient is on thyroxine (bool)
5. on antithyroid meds - whether patient is on antithyroid meds (bool)
6. sick - whether patient is sick (bool)
7. pregnant - whether patient is pregnant (bool)
8. thyroid_surgery - whether patient has undergone thyroid surgery (bool)
9. I131_treatment - whether patient is undergoing I131 treatment (bool)
10. query_hypothyroid - whether patient believes they have hypothyroid (bool)
11. query_hyperthyroid - whether patient believes they have hyperthyroid (bool)
12. lithium - whether patient * lithium (bool)
13. goitre - whether patient has goitre (bool)
14. tumor - whether patient has tumor (bool)
15. hypopituitary - whether patient * hyperpituitary gland (float)
16. psych - whether patient * psych (bool)
17. TSH_measured - whether TSH was measured in the blood (bool)
18. TSH - TSH level in blood from lab work (float)
19. T3_measured - whether T3 was measured in the blood (bool)
20. T3 - T3 level in blood from lab work (float)
21. TT4_measured - whether TT4 was measured in the blood (bool)
22. TT4 - TT4 level in blood from lab work (float)
23. T4U_measured - whether T4U was measured in the blood (bool)
24. T4U - T4U level in blood from lab work (float)
25. FTI_measured - whether FTI was measured in the blood (bool)
26. FTI - FTI level in blood from lab work (float)
27. TBG_measured - whether TBG was measured in the blood (bool)

28.TBG - TBG level in blood from lab work (float)
29.referral_source - (str)
30.target - hyperthyroidism medical diagnosis (str)
31.patient_id - unique id of the patient (str)

Target Metadata:

The diagnosis consists of a string of letters indicating diagnosed conditions. A diagnosis "-" indicates no condition requiring comment. A diagnosis of the form "X|Y" is interpreted as "consistent with X, but more likely Y". The conditions are divided into groups where each group corresponds to a class of comments.

Letter Diagnosis

------ ---------

hyperthyroid conditions:

A Hyperthyroid

B T3 toxic

C toxic goitre

D secondary toxic

hypothyroid conditions:

E hypothyroid

F primary hypothyroid

G compensated hypothyroid

H secondary hypothyroid

binding protein:

I increased binding protein

J decreased binding protein

general health:

K concurrent non-thyroidal illness

replacement therapy:

L consistent with replacement therapy

M underreplaced

N overreplaced

antithyroid treatment:

O antithyroid drugs

P I131 treatment

Q surgery

miscellaneous:

R discordant assay results

S elevated TBG

T elevated thyroid hormones

b). Details of the Hardware and Software Used:

Hardware:

- Processor: Intel Core i5
- RAM: 16 GB
- Storage: 512 GB SSD

Software:

- Operating System: Windows 11 64-bit
- Programming Language: Python 3.x
- IDE: Google Colab
-
- **Libraries:**
- Pandas: For data manipulation and analysis
- NumPy: For numerical computations
- Scikit-learn: For machine learning algorithms and data preprocessing
- Matplotlib: For data visualization

- Seaborn: For statistical data visualization
- Imbalanced-learn: For handling imbalanced datasets (SMOTE)

c). Literature Results in Terms of the Evaluation Metrics:

The performance of various machine learning models is evaluated based on standard metrics including accuracy, precision, recall, and F1 score. These metrics provide a comprehensive understanding of the model's performance, particularly in the context of imbalanced datasets.

- **Accuracy:** The proportion of true results (both true positives and true negatives) among the total number of cases examined. It is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  where TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively.

- **Precision:** The proportion of true positive results among all positive results predicted by the model. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity):** The proportion of true positive results among all actual positive cases. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 Score:** The harmonic mean of precision and recall, providing a balance between the two metrics. It is calculated as:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

d). Data Split-up:

In machine learning, it is crucial to evaluate the performance of models on unseen data to ensure that they generalize well. Splitting data in a dataset typically involves dividing it into training, validation, and test sets. The training set is used to train the model, the validation set is used to tune the model's hyperparameters and select the best model, and the test set is used to evaluate the model's performance. A common split ratio is 70-80% for training, 10-15% for validation, and 10-15% for testing.

We in our paper used 70 – 30 division of our dataset to evaluate model i.e 70% of the data we used for the training purpose and the remaining 30%, we used for testing purpose.

Steps involve in doing this are

- Loading the data:

  Load your dataset into a suitable format, usually as arrays or dataframes.

- Splitting the data:

  Use a method to randomly split the data into training and test sets.

- Training the model:

  Train your machine learning model using the training set.

- Evaluating the model:

  Evaluate the trained model's performance using the test set.

e).Comparision of the proposed model's performance with the state-of-the-art results interms of tables and figures.

Result Analysis of our proposed model:

Model	Precision (%)	Accuracy (%)	F1 Score (%)	Recall (%)
Decision Tree	99.383	99.0148	99.1778	99.0148
Logistic Regression	99.0172	99.5074	99.2617	99.5074
Random Forest	99.5074	99.5074	99.5074	99.5074
SVM	99.0172	99.5074	99.2617	99.5074
Feed Forward	99.016	99.2611	99.1384	99.2611

Comparing **accuracies** of our results with other papers

paper	Decision Tree	Logistic Regression	Random Forest	SVM
Paper 1	87.5	81.5	-	-
Paper 2	-	84	94	91
Paper 3	92.3	96.92	-	95.38
Paper 4	61	-	58	-
Paper 5	98.7	-	-	-
Paper 6	98.4	91.47	98.93	92.27
Paper 7	-	-	94.8	-
Paper 8	75.76	-	-	99.63
Paper 9	90.13	91.73	91.2	92.53
Paper 10	-	-	99.3	98.6
ours	99.46	97.5	99.3	96.25

f). Observations:

We found a few important factors in our machine learning research on liver illness prediction that had a big impact on the study's results. First, there was an imbalance in the dataset: more non-thyroid patients than thyroid patients. Then we used smote to balance the dataset.

Several important findings emerged from our investigation on the prediction of thyroid illness using different machine learning algorithms. These observations offer a thorough explanation of the behavior and performance of the models that were tested, in addition to offering insights on the qualities of the data and their significance.

- The **Random Forest** model achieved the highest overall accuracy of 99.5074%, demonstrating superior performance across all metrics with a precision of 99.5074%, recall of 99.5074%, and an F1-score of 99.5074%.
- The **Support Vector Machine (SVM)** model achieved an accuracy of 99.5074%, with a precision of 99.0172%, recall of 99.5074%, and an F1-score of 99.2617%.
- The **Decision Tree** model, while interpretable, showed least accuracy at 99.0148%, with a precision of 99.383%, recall of 99.0148%, and an F1-score of 99.1778%.

Conclusion and Future works:

This research paper explored the effectiveness of various machine learning techniques in predicting thyroid disease. By employing a range of algorithms, including Decision Trees, Random Forests, Support Vector Machines (SVM), we aimed to determine the most accurate and reliable method for early detection and diagnosis of thyroid conditions.

Our experimental results indicate that ensemble methods, particularly SVM, Logistic Regression, Random Forests, demonstrated superior performance in terms of accuracy compared to individual classifiers. This highlights the strength of

ensemble learning in capturing the complexities and nuances of thyroid disease data.

In conclusion, the application of machine learning techniques to thyroid disease prediction holds substantial promise for enhancing diagnostic accuracy and facilitating early intervention.

Future work should focus on refining these models through larger and more diverse datasets, integrating advanced techniques such as Neural Networks, deep learning, and exploring the potential of hybrid models that combine the strengths of various algorithms. Additionally, real-world implementation and validation in clinical settings are essential steps towards translating these technological advancements into practical healthcare solutions.

References:

1. Chaganti, R.; Rustam, F.; De La Torre Díez, I.; Mazón, J.L.V.; Rodríguez, C.L.; Ashraf, I. Thyroid Disease Prediction Using Selective Features and Machine Learning Techniques. Cancers 2022, 14, 3914. https://doi.org/10.3390/cancers14163914

2. Alyas, T.; Hamid, M.; Alissa, K.; Faiz, T.; Tabassum, N.; Ahmad, A. Empirical Method for Thyroid Disease Classification Using a Machine Learning Approach. BioMed Res. Int. 2022, 2022, 9809932.

3. Ji, Shengjun. "SSC: The novel self-stack ensemble model for thyroid disease prediction." *Plos one* 19, no. 1 (2024): e0295501.

4. Mir, Yasir Iqbal, and Sonu Mittal. "Thyroid disease prediction using hybrid machine learning techniques: An effective framework." *International Journal of Scientific & Technology Research* 9.2 (2020): 2868-2874.

5. Aversano, L., Bernardi, M. L., Cimitile, M., Iammarino, M., Macchia, P. E., Nettore, I. C., & Verdone, C. (2021). Thyroid disease treatment prediction with machine learning approaches. *Procedia Computer Science*, *192*, 1031-1040.

6. Chaubey, G., Bisen, D., Arjaria, S., & Yadav, V. (2021). Thyroid disease prediction using machine learning approaches. *National Academy Science Letters*, *44*(3), 233-238.

7. Jha, Ritesh, Vandana Bhattacharjee, and Abhijit Mustafi. "Increasing the prediction accuracy for thyroid disease: a step towards better health for society." *Wireless Personal Communications* 122.2 (2022): 1921-1938.

8. Selwal, Arvind, and Ifrah Raoof. "A Multi-layer perceptron based intelligent thyroid disease prediction system." *Indonesian Journal of Electrical Engineering and Computer Science* 17.1 (2020): 524-532.

9. Togor, Terlumun Emmanuel, Joshua Abah, and Dekera Kenneth Kwaghtyo. "Development of Thyroid Disease Prediction Model in Nigeria." *Development* 12.41 (2023).

10. De Ataide, Marissa Lourdes, and Amita Dessai. "Thyroid disease detection using soft computing techniques." *International Research journal of Engineering and Technology* 6.5 (2019): 8015-8016.