THYROID DETECTION USING MACHINE LEARNING

Done by

Nikhila Baby (MAC23MCA-2044)

Under the guidance

Of

Prof. Nisha Markose

ABSTRACT

The thyroid gland has one of the most important functions in regulating metabolism of our body. Hypothyroidism and hyperthyroidism are two critical conditions caused by insufficient thyroid hormone production and excessive thyroid hormone production, respectively. Accurate and timely detection of these diseases is crucial for effective treatment and management. The “Thyroid Detection Using Machine Learning” project is focussed on detecting and diagnosing thyroid disease.

From the three papers, we get to know that different models were used for the detection of thyroid disease. First paper is the detection and classification of thyroid disease using selective features. Second paper focuses on machine learning approach for thyroid disease detection using optimized model. Third paper aims at classification of hypo and hyper thyroidism.

The system is the comparative study of three algorithms Random Forest, Logistic Regression and Support Vector Machine. The models will classify thyroid disease under three classes which are no thyroid, hyperthyroid and hypothyroid. By comparing these algorithms, the project aims to determine which model offers the highest accuracy and reliability in diagnosing thyroid conditions.

The dataset is taken from the Kaggle repository. The dataset contains 9172 sample observations and has 31 columns including 1 identifier, 1 class variable and 29 features. The dataset contains numeric values and Boolean values.

Dataset: https://www.kaggle.com/datasets/emmanuelfwerr/thyroid-disease-data

References

- Chaganti R, Rustam F, De La Torre Díez I, Mazón JL, Rodríguez CL, Ashraf I. Thyroid disease prediction using selective features and machine learning techniques. Cancers. 2022 Aug 13;14(16):3914.
- Gupta P, Rustam F, Kanwal K, Aljedaani W, Alfarhood S, Safran M, Ashraf I. Detecting thyroid disease using optimized machine learning model based on differential evolution. International Journal of Computational Intelligence Systems. 2024 Jan 3;17(1):3.
- Hossain MB, Shama A, Adhikary A, Raha AD, Uddin KA, Hossain MA, Islam I, Murad SA, Munir MS, Bairagi AK. An explainable artificial intelligence framework for the predictive analysis of hypo and hyper thyroidism using machine learning algorithms. Human-Centric Intelligent Systems. 2023 Sep;3(3):211-31.

INTRODUCTION

The thyroid gland has one of the most important functions in regulating metabolism of our body. When the function of the thyroid gland is affected, it leads to inappropriate production of the thyroid hormone. Hypothyroidism and hyperthyroidism are two critical conditions caused by insufficient thyroid hormone production and excessive thyroid hormone production, respectively. Accurate and timely detection of these diseases is crucial for effective treatment and management. Traditional diagnostic methods, such as clinical evaluations and laboratory tests, which are effective, can sometimes be time-consuming and expensive.

The "Thyroid Detection Using Machine Learning" project is focussed on detecting and diagnosing thyroid disease. The system is the comparative study of three algorithms. The performance of three machine learning algorithms such as Random Forest, Logistic Regression, Support Vector Machine are compared to classify Thyroid disease into no thyroid, hypothyroidism, and hyperthyroidism categories. The models are evaluated on a separate test dataset using metrics such as accuracy, precision, recall, F1-score etc.

The dataset is taken from the Kaggle repository. The dataset contains 9172 sample observations and has 31 columns including 1 identifier, 1 class variable and 29 features. The dataset contains numeric and Boolean values.

A blood test is one of the ways to diagnose thyroid disease. But after a lab blood test, a medical expert needs to examine the test stats of hormones and other parameters of the patient to diagnose the disease. There is very little difference in the blood test stats, which refer to different thyroid hormone levels. Such minor differences can lead to the wrong diagnosis even by medical experts as human error is expected. Incorrect diagnosis may lead to wrong medication and further complexities. So, an automated system can be very helpful to assist medical experts and even make automated disease predictions without any human mistakes. Patients can also dragonise their condition without the assistance of a medical expert.

LITERATURE REVIEW

Paper 1: Chaganti R, Rustam F, De La Torre Díez I, Mazón JL, Rodríguez CL, Ashraf I. Thyroid disease prediction using selective features and machine learning techniques. Cancers. 2022 Aug 13;14(16):3914.

The primary focus of the paper is to improve the prediction accuracy of thyroid disease by utilizing selective features and various machine learning techniques. It highlights the importance of accurate and early detection of thyroid diseases to ensure timely and effective treatment.

This paper works on a five-class thyroid disease prediction problem, which includes primary hypothyroid, increased binding protein, compensated hypothyroid, and concurrent non-thyroidal illness. It uses four feature engineering methods such as forward feature selection (FFS), backward feature elimination (BFE), bidirectional feature elimination (BiDFE), and ML-based feature selection using extra tree classifiers. The paper evaluates five machine learning models such as Random Forest, Logistic Regression, Support Vector Machine, AdaBoost and GBM.

The extra tree classifier-based feature selection yielded the best accuracy of 99% when combined with the RF classifier.

Title of the paper	Chaganti R, Rustam F, De La Torre Díez I, Mazón JL, Rodríguez CL, Ashraf I. Thyroid disease prediction using selective features and machine learning techniques. Cancers. 2022 Aug 13;14(16):3914.
Area of work	Detection and classification of thyroid disease using selective features.
Dataset	Dataset was taken from UCI repository. The dataset contains 9172 sample observations and each sample is represented by 31 features.
Methodology / Strategy	The dataset consists of several thyroid-related disease records and many target classes. Four feature engineering approaches which includes forward feature selection (FFS), backward feature elimination (BFE), bidirectional feature elimination (BiDFE), and machine learning-based feature selection using an extra tree classifier are applied for feature selection. The aim is to classify thyroid disease to a multi-class problem.
Algorithm	RF, LR, SVM, ADA, GBM
Result/Accuracy	RF – 99 % GBM – 98 % ADA – 97 % LR – 87 % SVM – 92 %

Paper 2: Gupta P, Rustam F, Kanwal K, Aljedaani W, Alfarhood S, Safran M, Ashraf I. Detecting thyroid disease using optimized machine learning model based on differential evolution. International Journal of Computational Intelligence Systems. 2024 Jan 3;17(1):3.

The paper focuses on the issues of class imbalance. It considers ten types of thyroid diseases and uses a differential evolution (DE)-based optimization algorithm to fine-tune machine learning models parameters. Conditional generative adversarial networks (CTGAN) are employed for data augmentation.

The paper evaluates several machine learning models including Random Forest, Logistic Regression, Support Vector Machine, AdaBoost and GBM for classification. Performance is evaluated using accuracy, recall, precision, and F1 score metrics.

AdaBoost with DE optimization achieved the highest accuracy of 99.8%.

Title of the paper	Gupta P, Rustam F, Kanwal K, Aljedaani W, Alfarhood S, Safran M, Ashraf I. Detecting thyroid disease using optimized machine learning model based on differential evolution. International Journal of Computational Intelligence Systems. 2024 Jan 3;17(1):3.
Area of work	Machine learning approach for thyroid disease detection using optimized model.
Dataset	The datasets used in this study are taken from the Kaggle repository. The thyroid disease dataset comprises 9172 samples and every sample have 31 features.
Methodology / Strategy	The dataset contains 25 target classes, of which the top 10 target classes are selected for experiments. The selected targeted dataset is imbalanced, so to make the dataset balanced, CTGAN augmentation technique is used. Train machine learning models with a training set and perform hyperparameter optimization using DE optimizer which helps to select the best hyperparameter setting for models.
Algorithm	RF, SVM, LR, AdaBoost, GBM
Result/Accuracy	RF – 99.5 % GBM – 99.6 % AdaBoost – 99.8 % LR – 64.3 % SVM – 96.6 %

Paper 3: Hossain MB, Shama A, Adhikary A, Raha AD, Uddin KA, Hossain MA, Islam I, Murad SA, Munir MS, Bairagi AK. An explainable artificial intelligence framework for the predictive analysis of hypo and hyper thyroidism using machine learning algorithms. Human-Centric Intelligent Systems. 2023 Sep;3(3):211-31.

This paper focuses on using machine learning algorithms to predict hypothyroidism and hyperthyroidism and identifying significant features for accurate detection. It identifies the most important features for detecting thyroid disease.

The algorithms used include Decision Tree Classifier, Random Forest Classifier, Naive Bayes Classifier, Gradient Boosting Classifier, Logistic Regression Classifier, K- Nearest Neighbor, Support Vector Machine.

Random Forest provided the best performance with an accuracy of 91.42%.

Title of the paper	Hossain MB, Shama A, Adhikary A, Raha AD, Uddin KA, Hossain MA, Islam I, Murad SA, Munir MS, Bairagi AK. An explainable artificial intelligence framework for the predictive analysis of hypo and hyper thyroidism using machine learning algorithms. Human-Centric Intelligent Systems. 2023 Sep;3(3):211-31.
Area of work	Classification of hypo and hyper thyroidism
Dataset	The data was taken from the UCI machine learning repository. Dataset contains 3221 instances with a total of 30 features.
Methodology / Strategy	The selected targeted dataset is imbalanced, so to make the dataset balanced, resampling techniques are used. Feature selection methods like univariate feature selection approach and the feature importance method is implemented and the best set of characteristics for building effective models are selected.
Algorithm	Decision Tree Classifier, Random Forest Classifier, Naive Bayes Classifier, Gradient Boosting Classifier, Logistic Regression Classifier, K- Nearest Neighbor, Support Vector Machine
Result/Accuracy	DT – 90.43 % RF – 91.42 % GBM – 90.5 % NB – 67.86 % KNN – 86.22 % LR – 73.15 % SVM – 73.7 %

SUMMARY

The first paper “Chaganti R, Rustam F, De La Torre Díez I, Mazón JL, Rodríguez CL, Ashraf I. Thyroid disease prediction using selective features and machine learning techniques. Cancers. 2022 Aug 13;14(16):3914” focuses on using feature engineering techniques for thyroid disease prediction. Forward feature selection, backward feature elimination, bidirectional feature elimination, and machine learning-based feature selection using extra tree classifiers are adopted. The extra tree classifier based selected feature yields the best results with 99% accuracy when used with the random forest classifier.

The second paper “Gupta P, Rustam F, Kanwal K, Aljedaani W, Alfarhood S, Safran M, Ashraf I. Detecting thyroid disease using optimized machine learning model based on differential evolution. International Journal of Computational Intelligence Systems. 2024 Jan 3;17(1):3” uses a differential evolution (DE)-based optimization algorithm to fine-tune the parameters of machine learning models and conditional generative adversarial networks are used for data augmentation. AdaBoost with DE optimization obtained an accuracy of 99.8%.

The third paper “Hossain MB, Shama A, Adhikary A, Raha AD, Uddin KA, Hossain MA, Islam I, Murad SA, Munir MS, Bairagi AK. An explainable artificial intelligence framework for the predictive analysis of hypo and hyper thyroidism using machine learning algorithms. Human-Centric Intelligent Systems. 2023 Sep;3(3):211-31” focuses on different machine-learning algorithms to predict hypothyroidism and hyperthyroidism and identified the most significant features, which can be used to detect thyroid diseases more precisely. Random Forest provides the best accuracy of 91.42%.

PROJECT PROPOSAL

From the above three papers, we get to know that different models were used for the detection of thyroid disease. First paper is the detection and classification of thyroid disease using selective features. Second paper focuses on machine learning approach for thyroid disease detection using optimized model. Third paper aims at classification of hypo and hyper thyroidism.

Accurate and timely detection of thyroid diseases is crucial for effective treatment and management. The proposed system is the comparative study of three algorithms Random Forest, Logistic Regression and Support Vector Machine. The models will classify thyroid disease under three classes which are no thyroid, hyperthyroid and hypothyroid. By comparing these algorithms, the project aims to determine which model offers the highest accuracy and reliability in diagnosing thyroid conditions. Random Forest is known for its robustness and handling of complex datasets, Logistic Regression for its simplicity and interpretability, and SVM for its effectiveness in high-dimensional spaces.

Dataset is collected from Kaggle. The dataset is then pre-processed to handle missing values, normalize data, and select relevant features. The dataset is split into training and testing dataset. Three different machine learning models such as Random Forest (RF), Logistic Regression (LR), and Support Vector Machine (SVM) are trained using the prepared training dataset. The models are evaluated on a separate test dataset using metrics such as accuracy, precision, recall, F1-score etc.

An automated system based on these machine learning models can significantly benefit both medical professionals and patients. Incorrect diagnosis may lead to wrong medication and further complexities. So, an automated system can be very helpful to assist medical experts and even make automated disease predictions without any human mistakes. Patients can also diagnose their condition without the assistance of a medical expert.

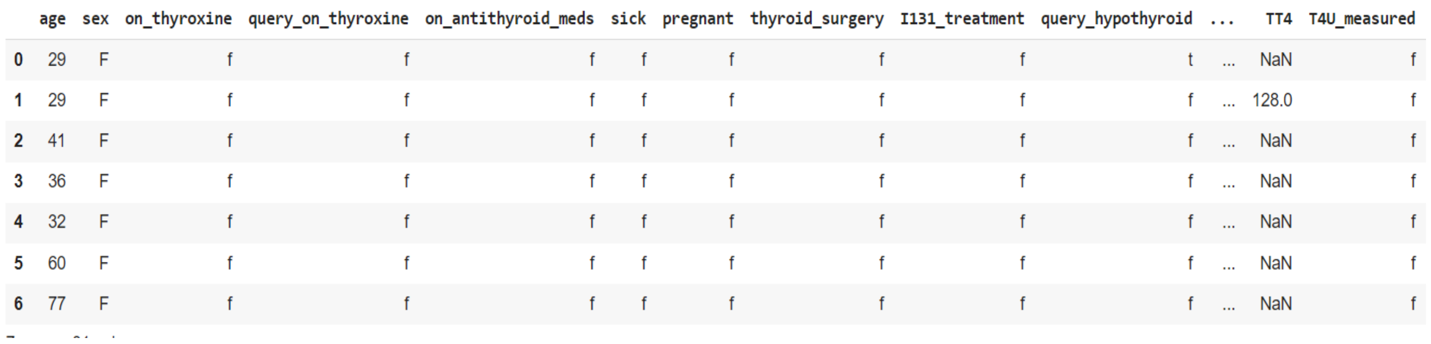
DATASET

The dataset is taken from the Kaggle repository. The dataset contains 9172 sample observations and has 31 columns including 1 identifier, 1 class variable and 29 features. The dataset contains numeric values and Boolean values. There are missing values in the dataset.

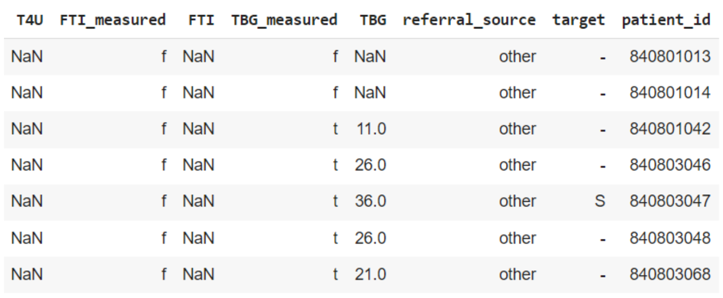
The identifier is the patient_id. The features are age, sex, on_thyroxine, query_on_thyroxine, on_antithyroid_meds, sick, pregnant, thyroid_surgery, I131_treatment, query_hypothyroid, query_hyperthyroid, lithium, goitre, tumor, hypopituitary, psych, TSH_measured, TSH, T3_measured, T3, TT4_measured, TT4, T4U_measured, T4U, FTI_measured, FTI, TBG_measured, TBG, and referral_source.

The class labels include letters from A to T which indicates different thyroid conditions.

Dataset: https://www.kaggle.com/datasets/emmanuelfwerr/thyroid-disease-data

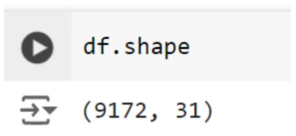


	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	TT4	T4U_measured
0	29	F	f	f	f	f	f	f	f	t	...	NaN	f
1	29	F	f	f	f	f	f	f	f	f	...	128.0	f
2	41	F	f	f	f	f	f	f	f	f	...	NaN	f
3	36	F	f	f	f	f	f	f	f	f	...	NaN	f
4	32	F	f	f	f	f	f	f	f	f	...	NaN	f
5	60	F	f	f	f	f	f	f	f	f	...	NaN	f
6	77	F	f	f	f	f	f	f	f	f	...	NaN	f



T4U	FTI_measured	FTI	TBG_measured	TBG	referral_source	target	patient_id
NaN	f	NaN	f	NaN	other	-	840801013
NaN	f	NaN	f	NaN	other	-	840801014
NaN	f	NaN	t	11.0	other	-	840801042
NaN	f	NaN	t	26.0	other	-	840803046
NaN	f	NaN	t	36.0	other	S	840803047
NaN	f	NaN	t	26.0	other	-	840803048
NaN	f	NaN	t	21.0	other	-	840803068

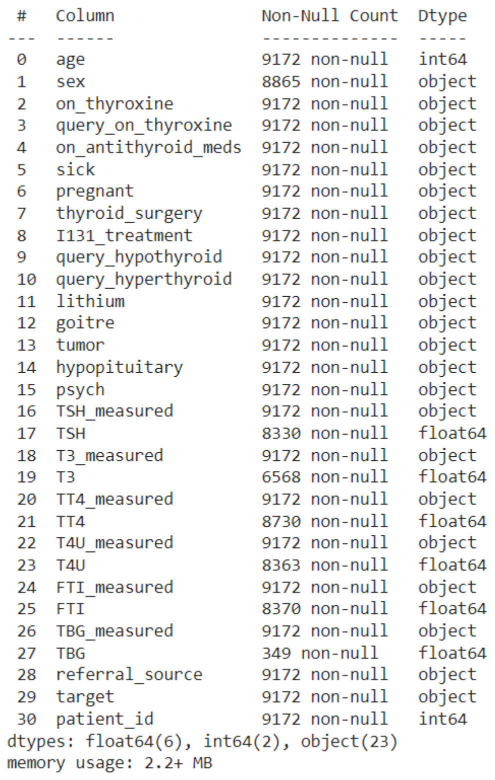
EXPLORATORY ANALYSIS



```
df.shape
```

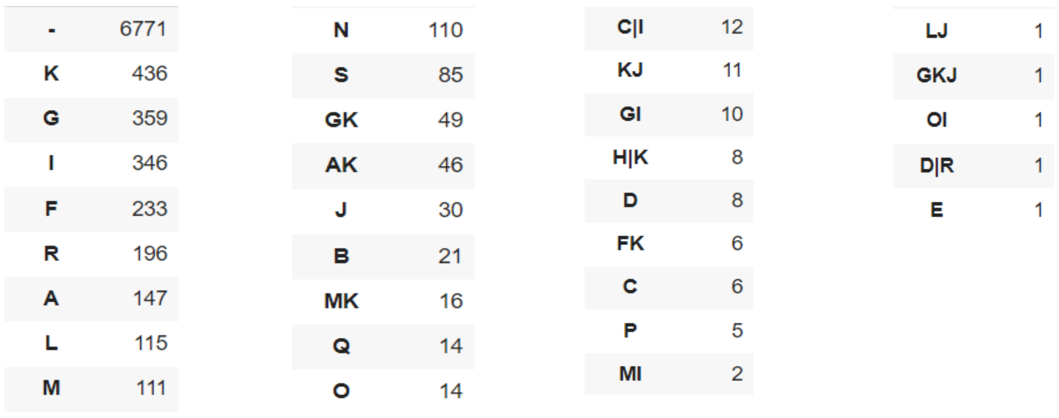
```
(9172, 31)
```

Dataset has 9172 rows and 31 columns



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

Dataset contains 8 numerical attributes and 23 non numerical attributes.



-	6771
K	436
G	359
I	346
F	233
R	196
A	147
L	115
M	111

N	110
S	85
GK	49
AK	46
J	30
B	21
MK	16
Q	14
O	14

C|I	12
KJ	11
GI	10
H|K	8
D	8
FK	6
C	6
P	5
MI	2

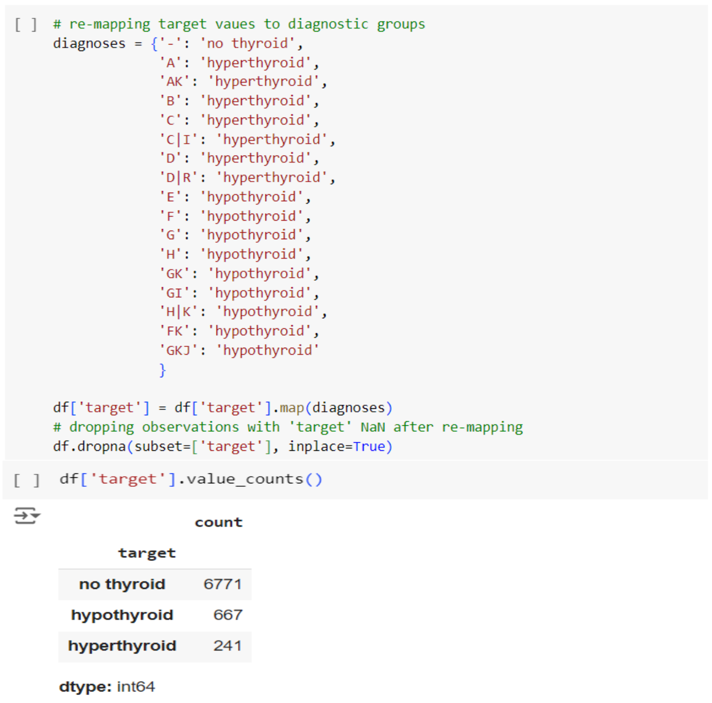
LJ	1
GKJ	1
OI	1
D|R	1
E	1

The target class contains letters from A to T which indicates different thyroid conditions.

DATA PREPROCESSING

Remapping Target Class

The proposed system will classify thyroid disease under three classes which are no thyroid, hyperthyroid and hypothyroid. The target class is remapped under no thyroid, hyperthyroid and hypothyroid.



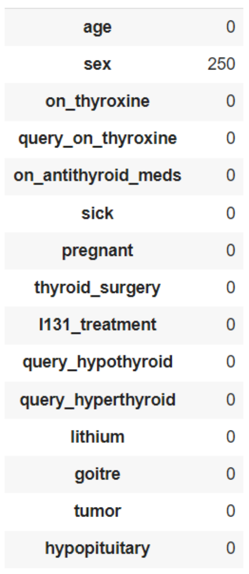
```
[ ] # re-mapping target vaues to diagnostic groups
    diagnoses = {'-': 'no thyroid',
                 'A': 'hyperthyroid',
                 'AK': 'hyperthyroid',
                 'B': 'hyperthyroid',
                 'C': 'hyperthyroid',
                 'C|I': 'hyperthyroid',
                 'D': 'hyperthyroid',
                 'D|R': 'hyperthyroid',
                 'E': 'hypothyroid',
                 'F': 'hypothyroid',
                 'G': 'hypothyroid',
                 'H': 'hypothyroid',
                 'GK': 'hypothyroid',
                 'GI': 'hypothyroid',
                 'H|K': 'hypothyroid',
                 'FK': 'hypothyroid',
                 'GKJ': 'hypothyroid'
                 }

    df['target'] = df['target'].map(diagnoses)
    # dropping observations with 'target' NaN after re-mapping
    df.dropna(subset=['target'], inplace=True)
```

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

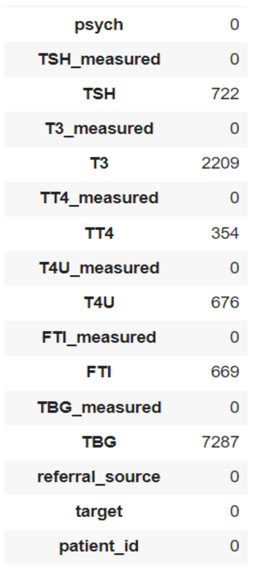
```
                 count
      target
  no thyroid      6771
 hypothyroid       667
hyperthyroid       241

dtype: int64
```

Missing Values

There are missing values in the dataset.

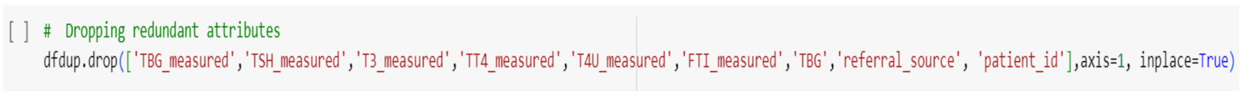


age	0
sex	250
on_thyroxine	0
query_on_thyroxine	0
on_antithyroid_meds	0
sick	0
pregnant	0
thyroid_surgery	0
I131_treatment	0
query_hypothyroid	0
query_hyperthyroid	0
lithium	0
goitre	0
tumor	0
hypopituitary	0

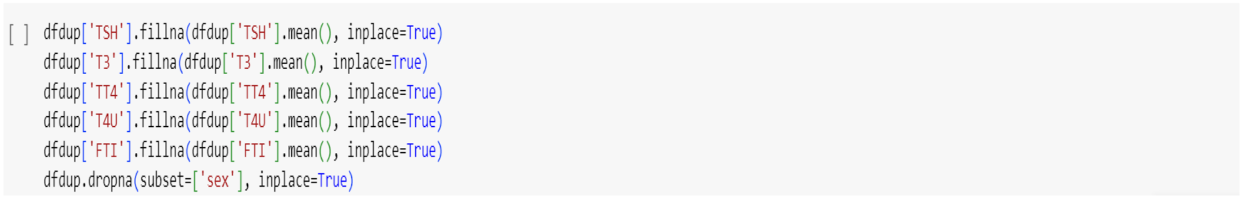


psych	0
TSH_measured	0
TSH	722
T3_measured	0
T3	2209
TT4_measured	0
TT4	354
T4U_measured	0
T4U	676
FTI_measured	0
FTI	669
TBG_measured	0
TBG	7287
referral_source	0
target	0
patient_id	0

- sex has 307 missing values - 3.35 %
- TSH has 842 missing values - 9.2 %
- T3 has 2604 missing values - 28.4 %
- TT4 has 442 missing values - 4.82 %
- T4U has 809 missing values - 8.8 %
- FTI has 802 missing values - 8.7 %
- TBG has 8823 missing values - 96.2 %



```
[ ] # Dropping redundant attributes
    dfdup.drop(['TBG_measured','TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG','referral_source', 'patient_id'],axis=1, inplace=True)
```



```
[ ] dfdup['TSH'].fillna(dfdup['TSH'].mean(), inplace=True)
    dfdup['T3'].fillna(dfdup['T3'].mean(), inplace=True)
    dfdup['TT4'].fillna(dfdup['TT4'].mean(), inplace=True)
    dfdup['T4U'].fillna(dfdup['T4U'].mean(), inplace=True)
    dfdup['FTI'].fillna(dfdup['FTI'].mean(), inplace=True)
    dfdup.dropna(subset=['sex'], inplace=True)
```

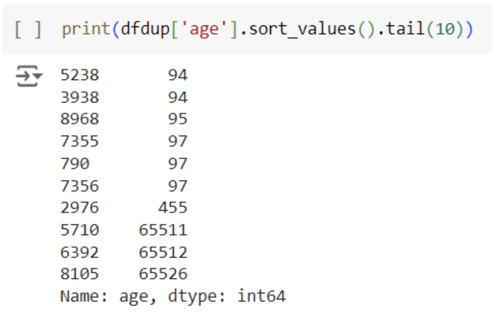
Since TBG has 96.2 % missing values, TBG is dropped.

Null values in sex are dropped.

TSH, T3, TT4, T4U and FTI are filled with their mean value.

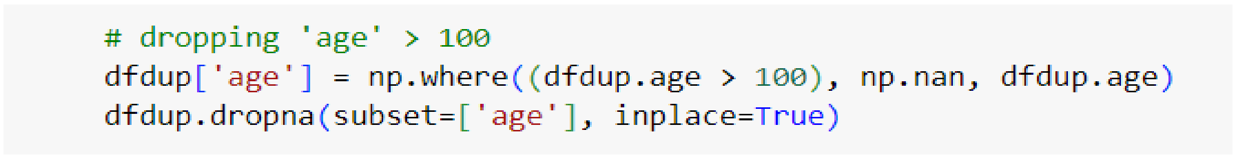
Handling Outliers

- **Age**



```
[ ] print(dfdup['age'].sort_values().tail(10))

5238       94
3938       94
8968       95
7355       97
790        97
7356       97
2976      455
5710    65511
6392    65512
8105    65526
Name: age, dtype: int64
```

Age above 100 are treated as outliers and they are removed.



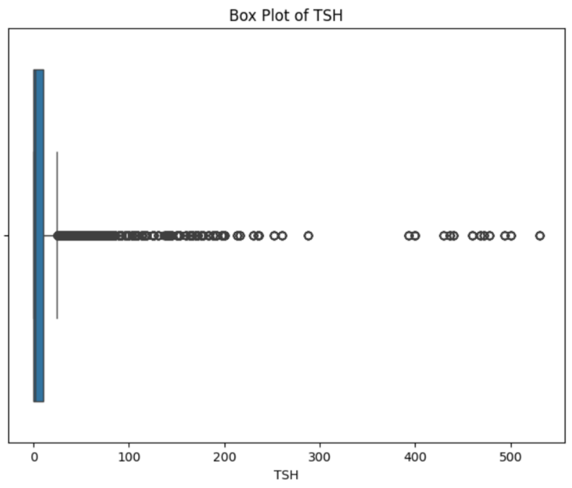
```
# dropping 'age' > 100
dfdup['age'] = np.where((dfdup.age > 100), np.nan, dfdup.age)
dfdup.dropna(subset=['age'], inplace=True)
```

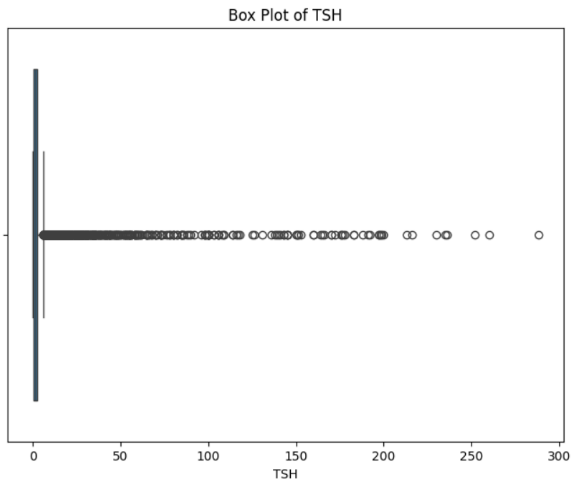
- **TSH**

TSH shows severe outliers. Most of them indicates severe thyroid condition. So, values above 300 are removed.

Before handling outlier

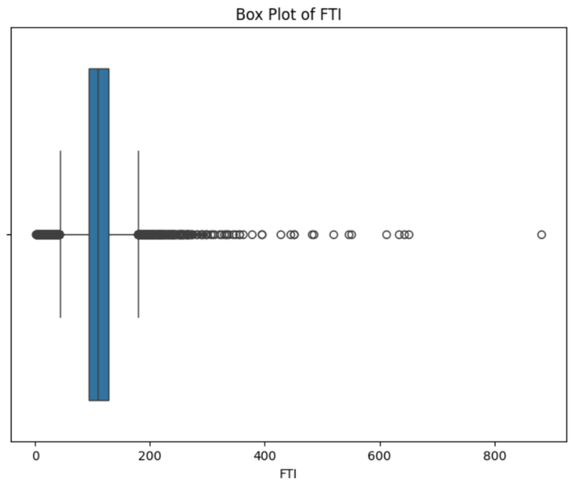
After handling outlier

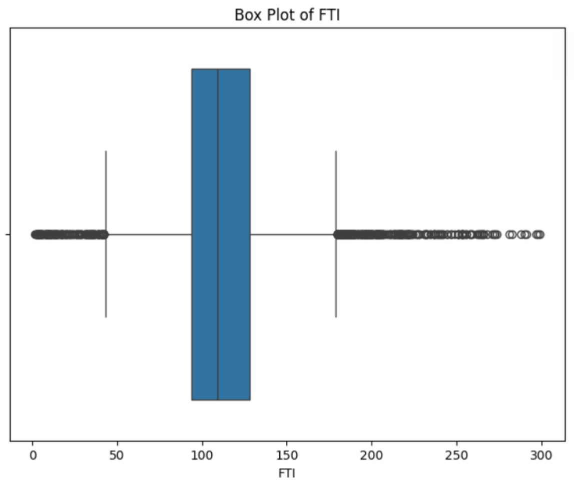




- **FTI**

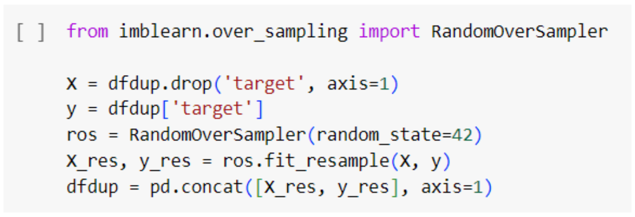
FTI is showing severe outliers. However, some of them indicate severe thyroid conditions. Values above 400 is removed.





Handling class imbalance

Target class is highly imbalanced. There are large number of people with no thyroid and comparatively small number of people with hypothyroid and hyperthyroid.

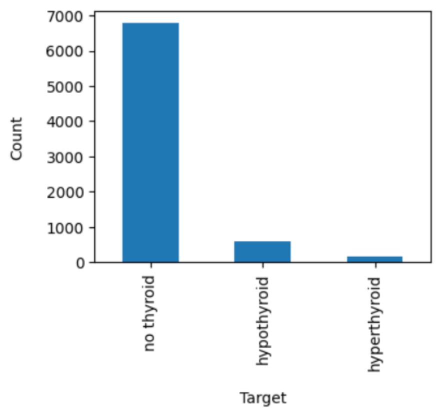


```
[ ] from imblearn.over_sampling import RandomOverSampler

    X = dfdup.drop('target', axis=1)
    y = dfdup['target']
    ros = RandomOverSampler(random_state=42)
    X_res, y_res = ros.fit_resample(X, y)
    dfdup = pd.concat([X_res, y_res], axis=1)
```

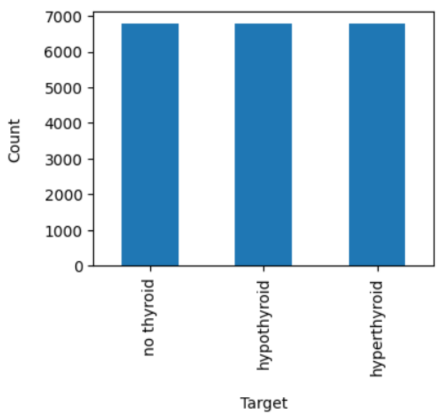
Imbalanced target class

Balanced target class

```
Text(0, 0.5, 'Count')
```



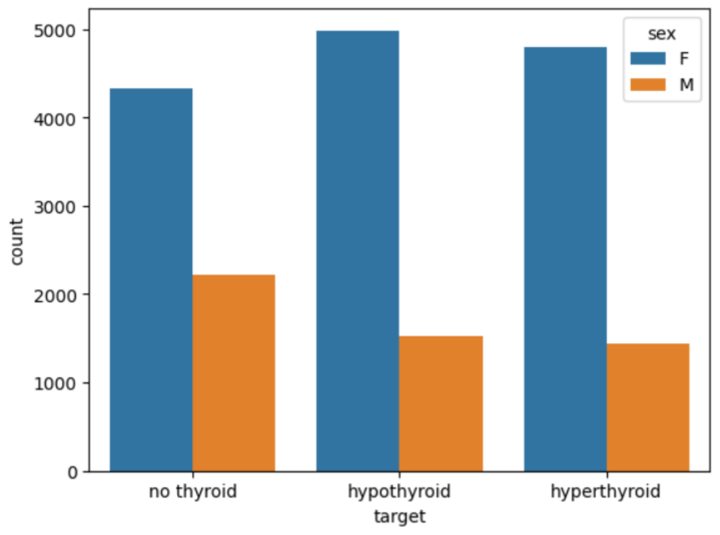
```
Text(0, 0.5, 'Count')
```



TRENDS AND PATTERNS

- **Sex**

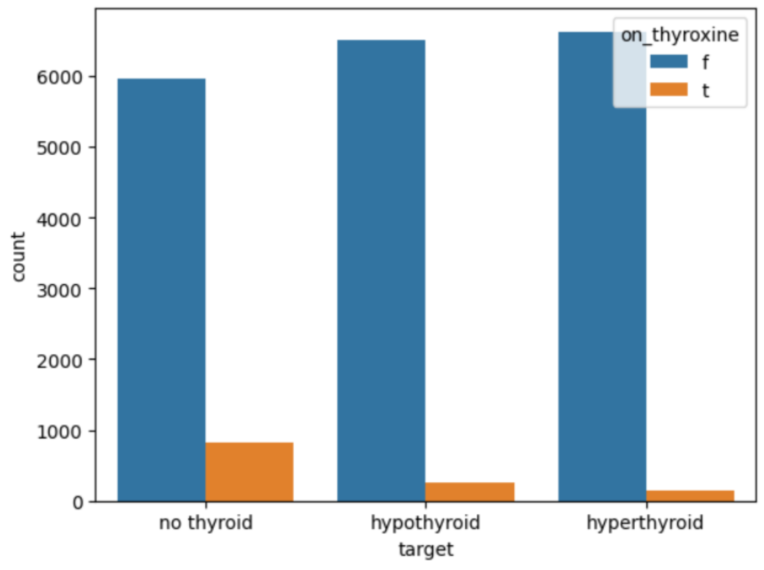
```
<Axes: xlabel='target', ylabel='count'>
```



Females are more likely to be affected by hypo and hyperthyroid.

- **On_thyroxine**

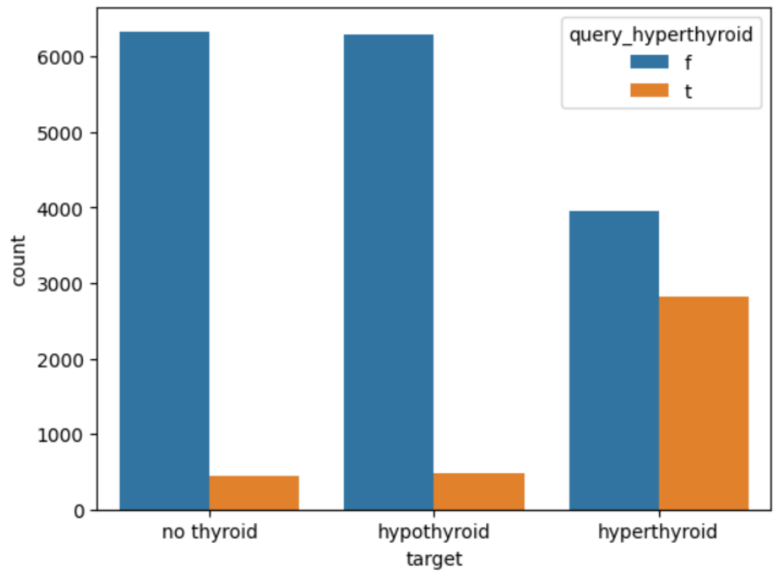
```
<Axes: xlabel='target', ylabel='count'>
```



Those who are not taking thyroxine medications are likely to be caused by hypo and hyper thyroid.

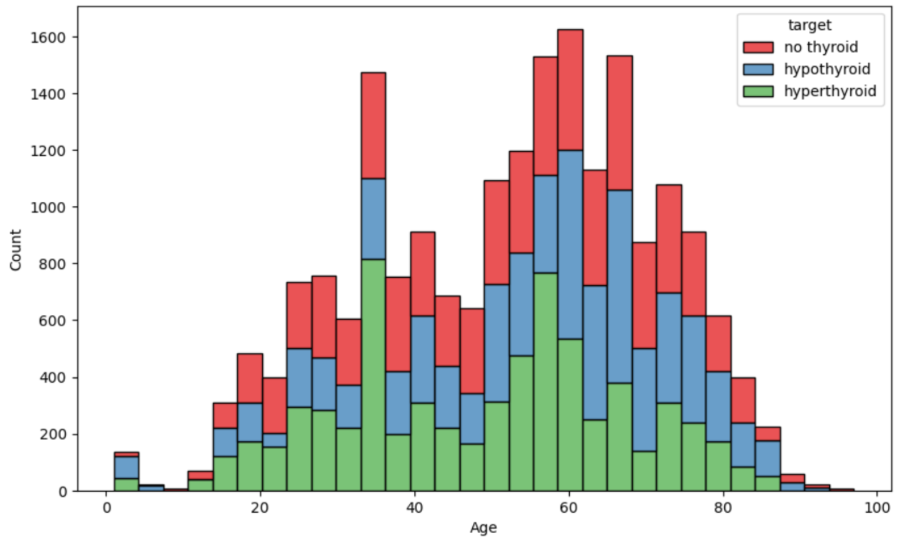
- **Query_hyperthyroid**

```
<Axes: xlabel='target', ylabel='count'>
```



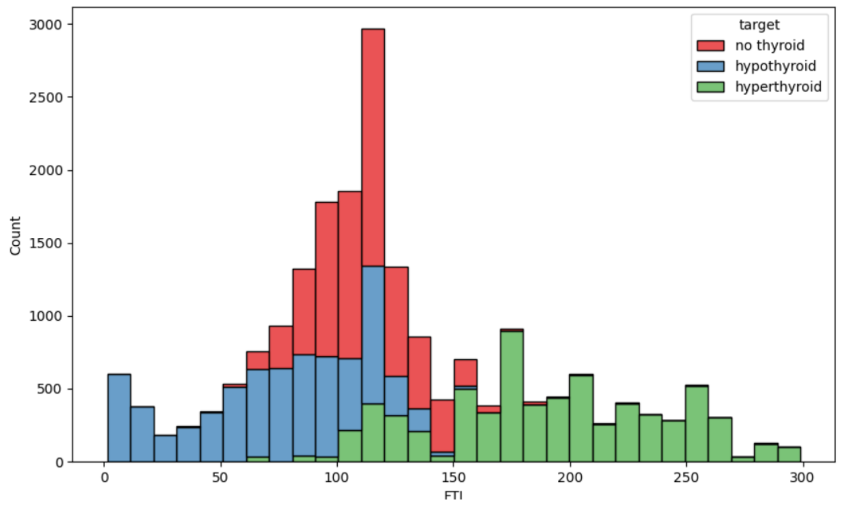
Those who queried about hyperthyroid are more likely to be affected by hyperthyroid.

- **Age**



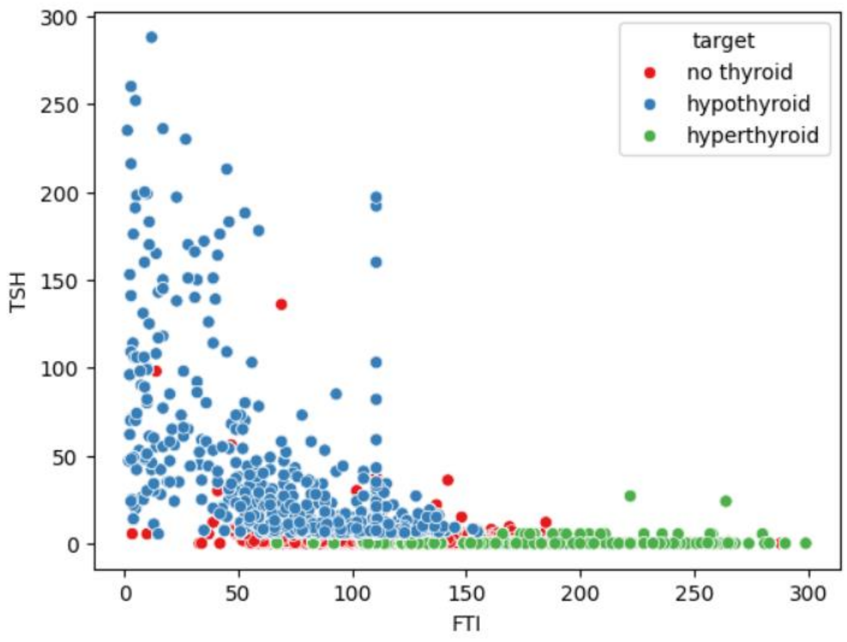
Age in the range from 18 to 80 are more likely to be affected by thyroid disease.

- **FTI**



Higher level of FTI causes hyperthyroid and lower level of FTI causes hypothyroid.

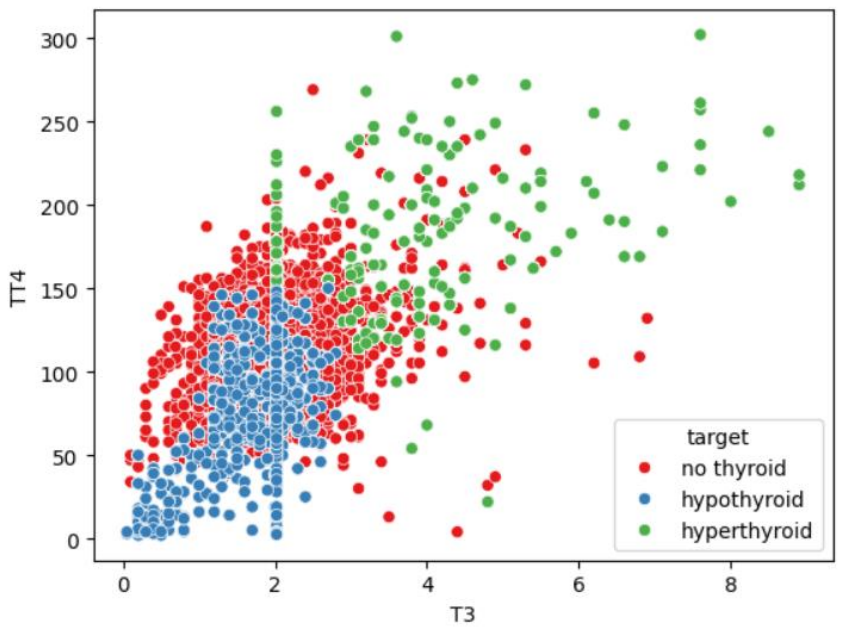
- **TSH and FTI**



Low TSH and high FTI causes hyperthyroid and high TSH and low FTI causes hypothyroid.

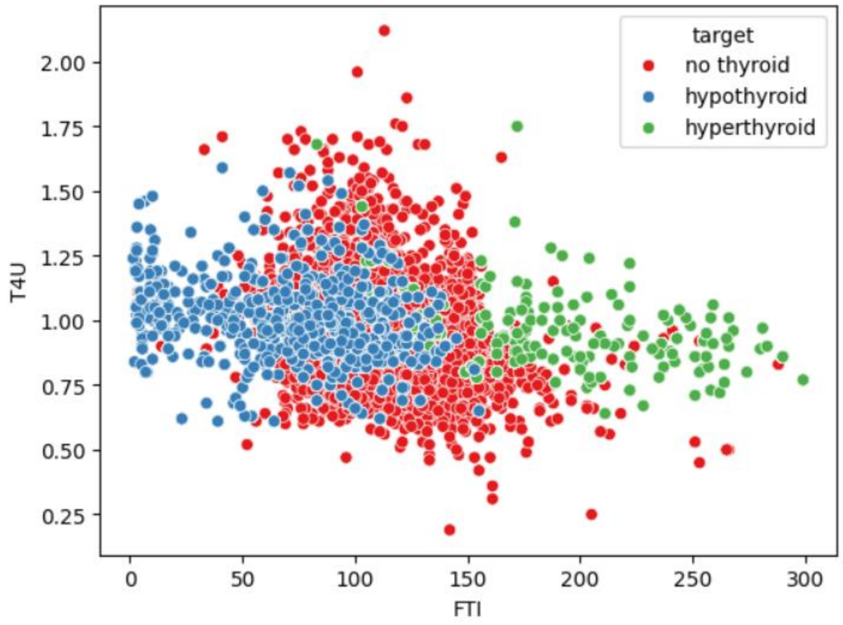
High TSH and low FTI causes hypothyroid.

- **TT4 and T3**



Low TT4 and low T3 causes hypothyroid and high TT4 and high T3 causes hyperthyroid.

- **T4U and FTI**



Increase in FTI causes hyperthyroid and decrease in FTI causes hypothyroid.

WORKING OF ALGORITHMS

Algorithms used in ‘Thyroid Detection Using Machine Learning’ are Random Forest, Logistic Regression and Support Vector Machine.

Random Forest

Random Forest is a supervised learning technique used for both Classification and Regression problems. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. Instead of relying on one decision tree, the random forest takes the output from each tree and based on the majority votes of outputs, and it takes the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Working

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Pseudocode

```
1. Input:
        - Training dataset with N samples and M features.
        - Number of trees to create: T.
        - Number of features to randomly select at each split: F.

2. Initialize an empty list called 'forest' to store the trees.

3. For each tree (from 1 to T):
   a. Create a random sample of the training data (some samples may repeat).
```

    b. Build a decision tree:
        i. At each decision point in the tree:
            - Randomly pick F features.
            - Find the best feature and split point among the F features to split the node.
            - Split the node into two child nodes based on the selected split point.
        ii. Keep splitting until the tree is fully grown or meets some stopping condition (like a max depth or minimum samples).
    c. Add this decision tree to the 'forest'.

4. To make a prediction for a new sample x:
    a. Let each tree in the forest make a prediction.
    b. The final prediction is the one that most trees agree on (majority vote).

Logistic Regression

Logistic regression comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression has several variants, including binary logistic regression, multinomial logistic regression, and ordinal logistic regression. Logistic regression is used for solving the classification problems.

<u>Logistic Function (Sigmoid Function)</u>

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.

Working

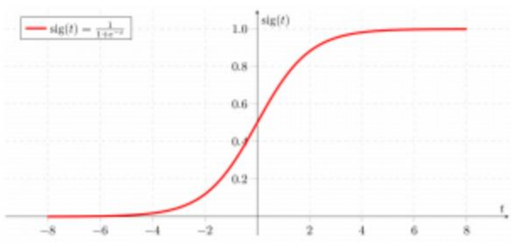
- Logistic regression works by creating a linear combination of input features, which involves multiplying each feature by a coefficient and summing the results.
  Applies the multi-linear function to the input variables

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

xi is the ith observation of X
wi=[w1,w2,w3,⋯,wm] is the weights or Coefficient
b is the bias term also known as intercept.

- This value is then passed through the logistic (sigmoid) function, which transforms it into a probability value between 0 and 1.
  Z will be input to the sigmoid function and find the probability between 0 and 1.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Sigmoid function

- Sigmoid function converts the continuous variable data into the probability i.e. between 0 and 1.
- For binary classification, this probability is compared to a threshold (usually 0.5) to decide the class label.

Multinomial logistic regression, also known as softmax regression, is used for multi-class classification tasks, where there are more than two possible outcomes for the output variable. In this case, the softmax function is used in place of the sigmoid function.
Softmax function for K classes will be:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

σ	= softmax
$\vec{z}$	= input vector
e^{z_i}	= standard exponential function for input vector
K	= number of classes in the multi-class classifier
e^{z_j}	= standard exponential function for output vector
e^{z_j}	= standard exponential function for output vector

- The calculated probabilities will be in the range of 0 to 1.
- The sum of all the probabilities is equals to 1.
- The class with highest probability will be the final output.

Pseudocode

```
1. Input:
        - Training data with N samples and M features.
       - Labels (0 or 1) for each sample.
```

```
        - Learning rate (how big of a step to take when updating weights).
        - Number of iterations (how many times to update weights).

2. Initialize:
        - Set weights and bias to 0 (or small random numbers).

3. For a set number of iterations:
   a. For each sample in the training data:
        i. Calculate the weighted sum (z):
           z=(weight_1*feature_1)+(weight_2*feature_2)+...+(weight_M*feature_M) + bias
     ii. Apply the sigmoid function to get a probability:
          probability = 1 / (1 + exp(-z))
     iii. Calculate the error:
          error = probability - actual_label
     iv. Update each weight:
          weight_j = weight_j - (learning rate * error * feature_j)
     v. Update the bias:
          bias = bias - (learning rate * error)

4. After finishing all iterations, use the weights and bias to make predictions:
   a. For a new sample:
     i. Calculate the weighted sum (z) using the final weights and bias.
     ii. Apply the sigmoid function to get a probability.
     iii. If the probability is 0.5 or higher, output class 1. Otherwise, output class 0.
```

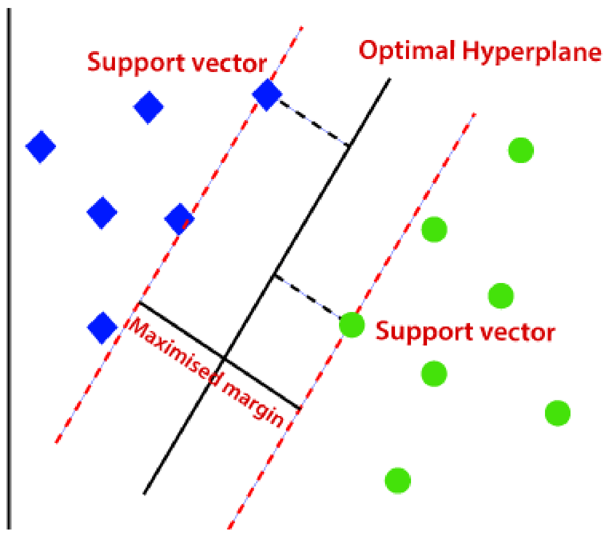
Support Vector Machine

Support Vector Machine is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features.

Types of SVM

Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

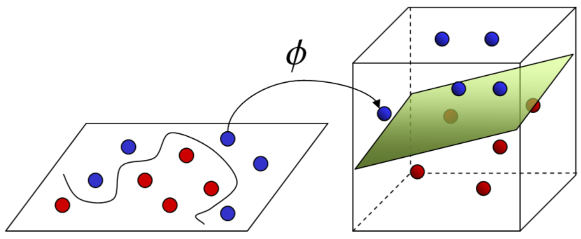
Non-Linear SVM can be used to classify data when it cannot be separated into two classes by a straight line (in the case of 2D). By using kernel functions, nonlinear SVMs can handle nonlinearly separable data. The original input data is transformed by these kernel functions into a higher-dimensional feature space, where the data points can be linearly separated. A linear SVM is used to locate a nonlinear decision boundary in this modified space.



The best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category is called a hyperplane. There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors. The distance between the vectors and the hyperplane is called as margin. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers. So, the margins in these types of cases are called soft margins.

If the data is non-linear, we cannot draw a single straight line. So, to separate these data points, we need to add one more dimension. For that, SVM kernel function is used.



The SVM kernel is a function that takes low-dimensional input space and transforms it into higher-dimensional space, i.e. it converts nonseparable problems to separable problems. It is mostly useful in non-linear separation problems. Some commonly used kernel functions are:

$$\text{Linear} : K(w,b) = w^T x + b$$
$$\text{Polynomial} : K(w,x) = (\gamma w^T x + b)^N$$
$$\text{Gaussian RBF}: K(w,x) = \exp(-\gamma ||x_i - x_j||^n$$
$$\text{Sigmoid} : K(x_i,x_j) = \tanh(\alpha x_i^T x_j + b)$$

Working

- Gather dataset which consists of feature vectors (inputs) and corresponding labels (outputs).
- Represent each data point as a vector in a multi-dimensional space, where each dimension corresponds to a feature of the data.
- Find the hyperplane that maximizes this margin, ensuring the best possible separation between the two classes.
- If the data is not linearly separable, the original feature space is mapped into a higher-dimensional space by using kernel trick where a linear hyperplane can separate the classes.
- Find the weights and bias that define the hyperplane by maximizing the margin and minimizing the classification errors.
- Once trained, the SVM model can classify new data points. For each new point, the model checks on which side of the hyperplane the point falls.

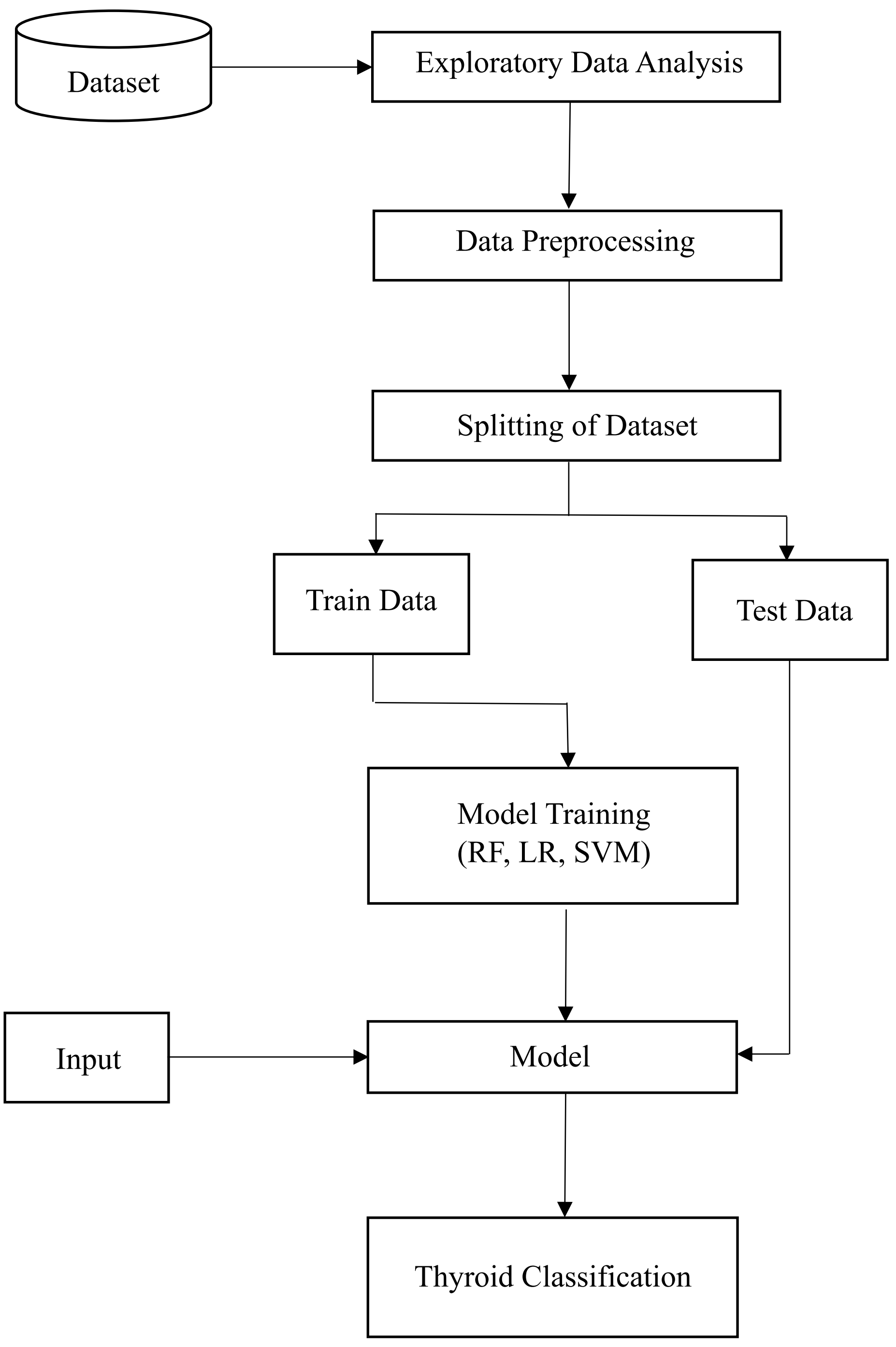
Pseudocode

```
1. Initialize Parameters:
        - Set the regularization parameter C.
        - Initialize weights w and bias b to 0 or small random values.

2. Training Process:
    - Repeat for a fixed number of iterations:
        a. For each data point (xi, yi) in the training set:
            i.   Calculate the decision function: f(xi) = w * xi + b
            ii.  If the data point is misclassified (i.e., yi * f(xi) < 1):
                    - Update weights: w = w + C * yi * xi
                    - Update bias: b = b + C * yi

3. Prediction:
    - For a new data point x_new, compute:
        f(x_new) = w * x_new + b
    - Return the predicted class:
        if f(x_new) >= 0, return +1
        else return -1
```

PACKAGES AND FUNCTIONS

- Pandas
  - read_csv()
  - head()
  - shape()
  - info()
  - drop()
  - value_counts()
  - isnull()
  - describe()
  - tail()
- Matplotlib
  - figure()
  - title()
  - show()
  - xlabel()
  - ylabel()
- Seaborn
  - boxplot()
  - countplot()
  - heatmap()
  - scatterplot()
- Numpy

PROJECT PIPELINE



TIMELINE

- Submission of project synopsis with Journal Papers - 22.07.2024
- project proposal approval - 26.07.2024
- Presenting project proposal before the

  Approval Committee - 29.07.2024 & 30.07.2024
- Initial report submission - 12.08.2024
- Analysis and design report submission - 16.08.2024
- Verification of the report and PPT by the guide - 19.08.2024
- First project presentation - 21.08.2024 & 23.08.2024
- Sprint Release I - 30.08.2024
- Sprint Release II - 26.09.2024
- Interim project presentation - 30.09.2024 & 01.10.2024
- Sprint Release III - 18.10.2024
- Project execution, submission of project report and PPT before the guide - 24.10.2024
- Submission of the project report to the guide - 28.10.2024
- Final project presentation - 28.10.2024 & 29.10.2024
- Submission of project report after corrections - 01.11.2024