

Thyroid Disease Prediction

Submitted by,

**Alen K Joseph
Kishan M S
Group : 34
Union Christian College, Aluva**

TABLE OF CONTENTS: -

1. INTRODUCTION.....	2
OVERVIEW.....	2
PURPOSE.....	2
2. LITERATURE SURVEY.....	3-4
EXISTINGPROBLEM.....	3
PROPOSED SYSTEM	4
3. THEORITICAL ANALYSIS.....	4
BLOCK DIAGRAM.....	4
HARDWARE / SOFTWARE DESIGNING.....	4
4. EXPERIMENTAL INVESTIGATIONS	4
5. FLOW CHART.....	5
6. RESULT.....	6-7
7. ADVANTAGES & DISADVANTAGES.....	8
8. APPLICATIONS	8
9. CONCLUSION	8
10.FUTURE SCOPE.....	9
11. BIBILOGRAPHY	9
12.APPENDIX.....	9-15

INTRODUCTION

OVERVIEW

Thyroid diseases, such as hypothyroidism and hyperthyroidism, are common endocrine disorders that affect the function of the thyroid gland. These diseases can have a significant impact on a patient's health and quality of life. Early and accurate diagnosis of thyroid diseases is important for effective treatment.

In recent years, machine learning techniques have been applied to the classification of thyroid diseases. The goal of these studies is to develop models that can accurately diagnose thyroid diseases based on clinical and laboratory data.

PURPOSE

There are several machine learning algorithms that have been used for thyroid disease classification, including decision trees, random forests, k-nearest neighbors (KNN), support vector machines (SVM), artificial neural networks (ANN), and deep learning algorithms such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

The input data for these models can include clinical features, such as age, gender, and symptoms, as well as laboratory test results, such as thyroid-stimulating hormone (TSH) levels and levels of thyroxine (T4) and triiodothyronine (T3).

The performance of these models is usually evaluated using metrics such as accuracy, precision, recall, and F1 score. In general, deep learning algorithms have shown better performance than other machine learning algorithms in thyroid disease classification tasks.

LITERATURE SURVEY

EXISTING PROBLEM

The current existing system includes:

Clinical examination: This involves a physical examination of the neck to check for any visible signs of thyroid enlargement or nodules.

Blood tests: Blood tests are used to measure the levels of hormones produced by the thyroid gland and to check for antibodies that may indicate autoimmune diseases such as Hashimoto's thyroiditis.

Ultrasound: An ultrasound scan can provide images of the thyroid gland and help to identify any nodules or other abnormalities.

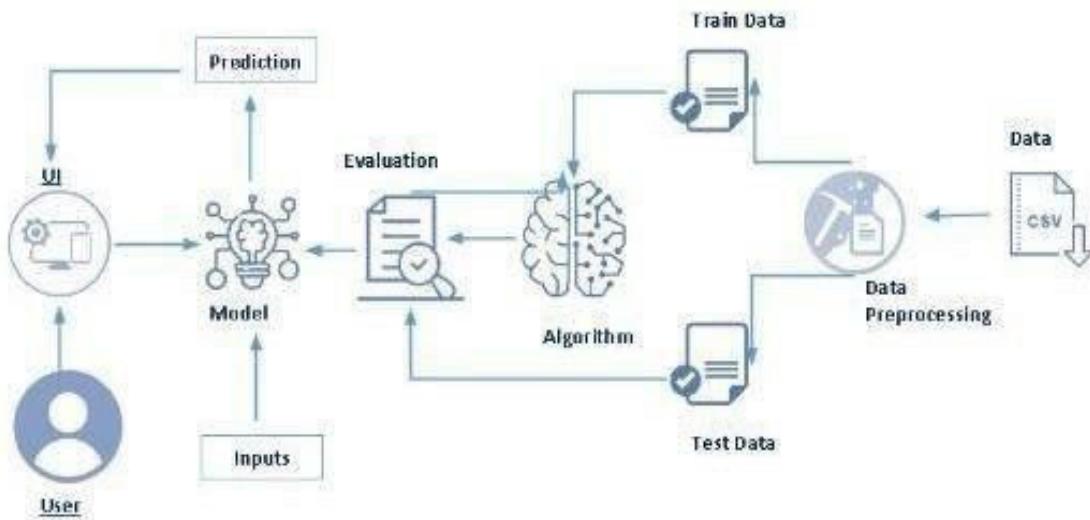
Fine needle aspiration biopsy (FNAB): This is a procedure in which a small sample of tissue is taken from a thyroid nodule using a fine needle, which is then examined under a microscope to check for cancer.

PROPOSED SYSTEM

The proposed system is by using Artificial Intelligence (AI) and Machine Learning (ML). In recent years, there have been several studies exploring the use of AI and ML algorithms for predicting thyroid diseases. These systems are trained on large datasets of patient data and use various features such as demographic information, blood test results, and ultrasound images to make prediction.

THEORETICAL ANALYSIS

BLOCK DIAGRAM



HARDWARE AND SOFTWARE DESIGNING

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum, and first released on February 20, 1991. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Anaconda Navigator

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross platform, package management system. Anaconda comes with

so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder.

Jupyter Notebook

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use

Spyder

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. Spyder is extensible with first-party and third party plugins includessupport for interactive tools for data inspection and embeds Python specific code. Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

Flask

Web frame work used for building. It is a web application framework written in python which will be running in local browser with a user interface. In this application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the use.

Hardware Requirements:

- o Operating system: window 7 and above with 64bit
- o Processor Type -Intel Core i3-3220
- o RAM: 4Gb and above
- o Hard disk: min 100GB

1.

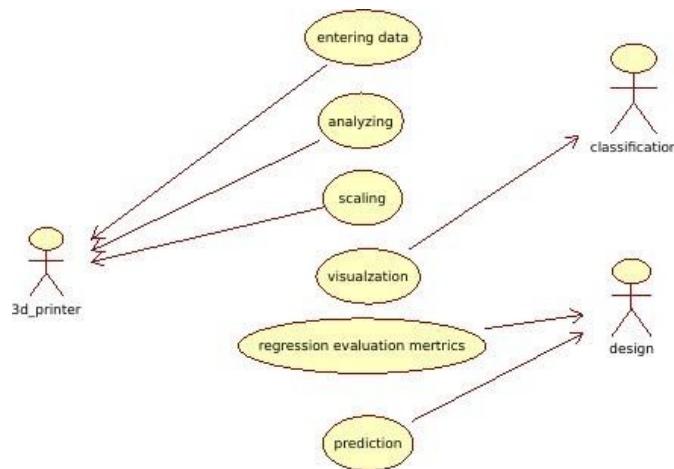
EXPERIMENTAL INVESTIGATION

Here we are going to build a machine learning model that predicts whether the given message is a spam or not, based on these parameters a supervised machine learning model is built to predict the best

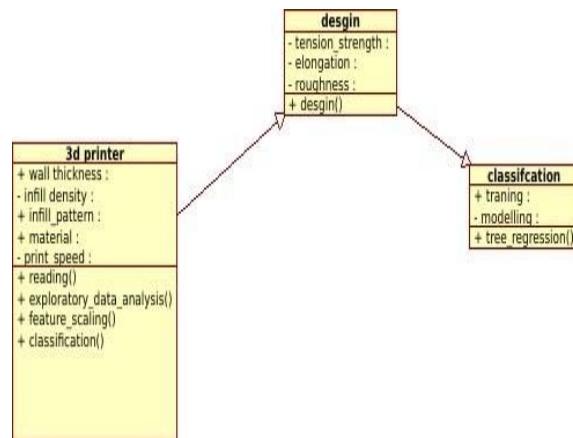
material to be used for building 3D models. A web application is build so that the user can type in the mentioned part a meters and the material which suits the best is showcased on UI.

2.

FLOWCHART

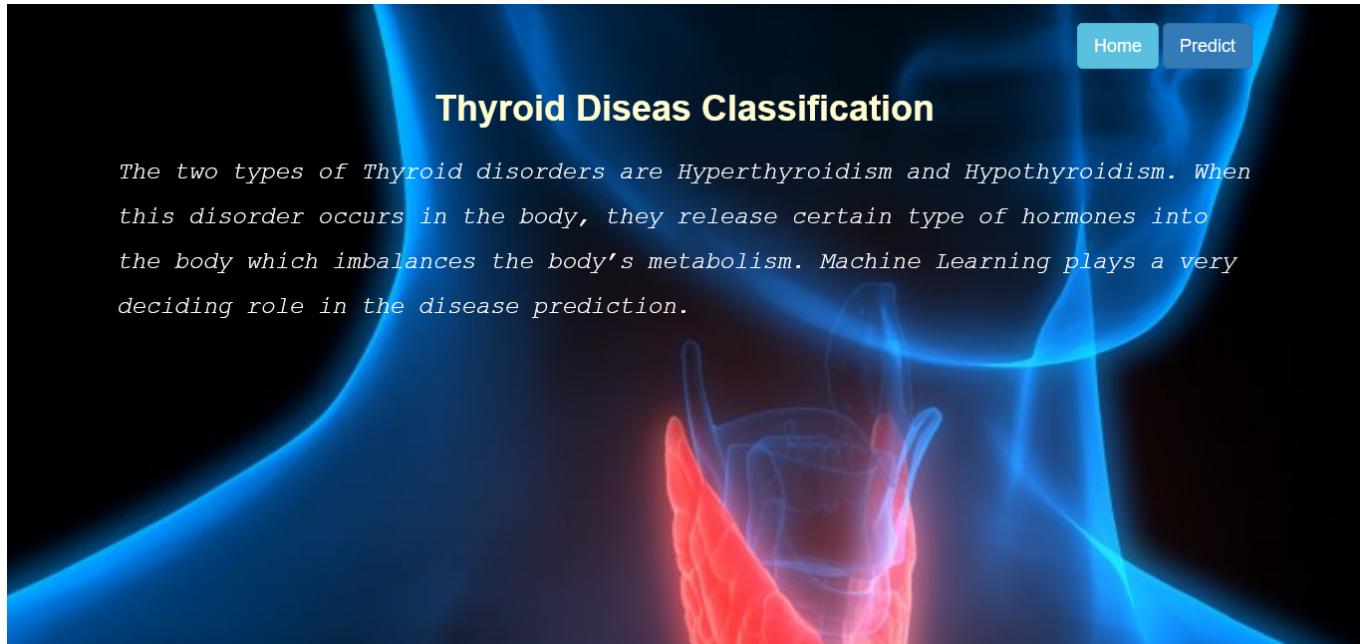


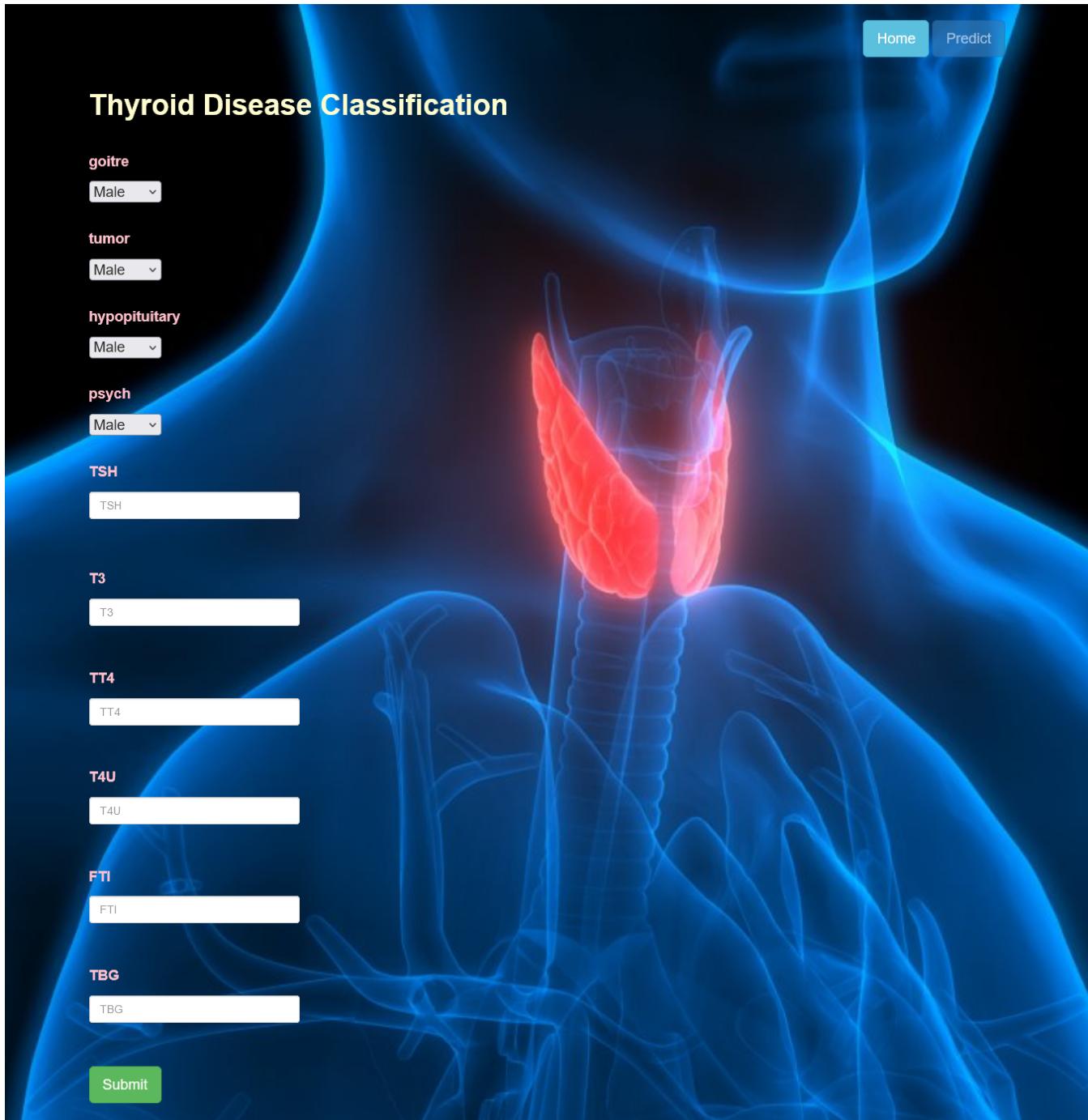
USE CASE DIAGRAM



3.

RESULT





The background of the form features a blue-toned anatomical illustration of a human neck. The thyroid gland is highlighted in a bright red color, showing its two lobes and central isthmus. The surrounding structures, including the trachea, esophagus, and major blood vessels, are depicted in a translucent blue.

Thyroid Disease Classification

goitre

Male

tumor

Male

hypopituitary

Male

psych

Male

TSH

T3

TT4

T4U

FTI

TBG

Home



Home Predict

Thyroid Disease Classification

Based on the given input, it predicts Thyroid disease for your body condition is {{prediction_text}}.

4. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Easy to use
- Cost efficient
- Time efficient

DISADVANTAGE

1. Initial costs of printer
2. Post processing
3. Printing time
4. Special skill required for 3D models
5. Manufacturing Job Losses

8. Applications

3D printing has gone through a number of changes over the years. In the early days, 3D printing was time-consuming and costly, and not very practical for applications outside of industry. However, with the advent of today's more flexible and cost-effective 3D printing methods, there are areas where 3D printing has become a practical tool.

It is applicable in different sectors such as

- Engineering And Design
- Consumer products
- Manufacturing
- Education
- Aerospace
- Medical
- Movies / Theatres
- Architectures

9. CONCLUSION

3D printing technology could revolutionize and re-shape the world. Advance in 3D technology can significantly change and improve the way we manufacture products worldwide.

If the last industrial revolution brought us mass production and the advent of economics of scale – the digital 3D printing revolution could bring mass manufacturing back a full of circle – to an era of mass personalization, and return to individual craftsmanship.

10. FUTURE SCOPE

Future applications for 3D printing might include creating open-source scientific equipment to create opensource labs

Science-based applications like reconstructing fossils in palaeontology . Replicating ancient and priceless artifacts in archaeology

Reconstructing bones and body parts in forensic pathology. The technology currently being researched for building construction.

11. BIBILOGRAPHY

- <http://mashable.com/2014/03/06/3d-printed-blood-vessels/>
- <http://www.3dprinter.net/>

12. APPENDIX

```
In [1]: import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return iter(self)

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_3055a99af3464697994102c4e129439e = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='ENLjs91Tkkb2D19X6CwgXNYtpJMCpNEelE6j8RyJ754',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url="https://s3.private.us.cloud-object-storage.appdomain.cloud")

body = client_3055a99af3464697994102c4e129439e.get_object(Bucket='thyroid-donotdelete-pr-qkstuhbfy1zjo',Key='data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

data = pd.read_csv(body)
data.head()
data.shape

Out[1]: (9172, 31)

In [13]: data.info()

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

```
In [3]: data.isnull().sum()
Out[3]:
age           0
sex          307
on_thyroxine   0
query_on_thyroxine 0
on_antithyroid_meds 0
sick          0
pregnant      0
thyroid_surgery 0
I131_treatment 0
query_hypothyroid 0
lithium        0
goitre         0
tumor          0
hypopituitary   0
psych          0
TSH_measured    0
TSH            842
T3_measured     0
T3             2604
TT4_measured    0
TT4            442
T4U_measured    0
T4U            809
FTI_measured    0
FTI            802
TBG_measured    0
TBG            8823
referral_source 0
target         0
patient_id      0
dtype: int64

In [4]: #Removing Redundant attributes from dataset
#The columns listed below were removed because of redundancy.
#They are boolean and state whether or not a value has been recorded for their respective blood tests.
#TSH_measured
#T3_measured
#TT4_measured
#T4U_measured
#FTI_measured
#TBG_measured
data.drop(['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG_measured','referral_source','patient_id'], axis=1, inplace = True)

In [5]: data.head()
Out[5]:
   age sex on_thyroxine query_on_thyroxine on_antithyroid_meds sick pregnant thyroid_surgery I131_treatment query_hypothyroid ... tumor hypopituitary psych TSH T3 TT4 T4U FTI TBG target
0  29   F          f              f           f       f       f          f          f          f          t ... f          f          f          f  0.3  NaN  NaN  NaN  NaN  NaN  -
1  29   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  1.6  1.9  128.0  NaN  NaN  NaN  -
2  41   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  NaN  NaN  NaN  NaN  NaN  11.0  -
3  36   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  NaN  NaN  NaN  NaN  NaN  26.0  -
4  32   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  NaN  NaN  NaN  NaN  NaN  36.0  S
```

5 rows × 23 columns

```
In [6]: data['target']
Out[6]:
0   -
1   -
2   -
3   -
4   S
...
9167  -
9168  -
9169  I
9170  -
9171  -
Name: target, Length: 9172, dtype: object

In [7]: #re-mapping target values to diagnostic group
diagnoses = {'A': 'hyperthyroid conditions',
 'B': 'hyperthyroid conditions',
 'C': 'hyperthyroid conditions',
 'D': 'hyperthyroid conditions',
 'E': 'hypothyroid conditions',
 'F': 'hypothyroid conditions',
 'G': 'hypothyroid conditions',
 'H': 'hypothyroid conditions',
 'I1': 'binding protein',
 'J1': 'binding protein',
 'K': 'general health',
 'L': 'replacement therapy',
 'M': 'replacement therapy',
 'N': 'replacement therapy',
 'O': 'antithyroid treatment',
 'P': 'antithyroid treatment',
 'Q': 'antithyroid treatment',
 'R': 'miscellaneous',
 'S': 'miscellaneous',
 'T': 'miscellaneous'}
data['target'] = data['target'].map(diagnoses) #remapping

In [8]: data
Out[8]:
   age sex on_thyroxine query_on_thyroxine on_antithyroid_meds sick pregnant thyroid_surgery I131_treatment query_hypothyroid ... tumor hypopituitary psych TSH T3 TT4 T4U FTI TBG target
0  29   F          f              f           f       f       f          f          f          f          t ... f          f          f          f  0.3  NaN  NaN  NaN  NaN  NaN  NaN
1  29   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  1.6  1.9  128.0  NaN  NaN  NaN  NaN
2  41   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  NaN  NaN  NaN  NaN  NaN  11.0  NaN
3  36   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  NaN  NaN  NaN  NaN  NaN  26.0  -
4  32   F          f              f           f       f       f          f          f          f          f ... f          f          f          f  NaN  NaN  NaN  NaN  NaN  36.0  miscellaneous
...
9167  56   M          f              f           f       f       f          f          f          f          f ... f          f          f          f  64.0  0.83  77.0  NaN  NaN  NaN  NaN
9168  22   M          f              f           f       f       f          f          f          f ... f          f          f          f  91.0  0.92  99.0  NaN  NaN  binding protein
9169  69   M          f              f           f       f       f          f          f          f ... f          f          f          f  113.0  1.27  89.0  NaN  NaN  -
9170  47   F          f              f           f       f       f          f          f          f ... f          f          f          f  75.0  0.85  88.0  NaN  NaN  NaN
9171  31   M          f              f           f       f       f          f          f          f ... f          f          f          f  66.0  1.02  65.0  NaN  NaN  NaN
```

9172 rows × 23 columns

```
In [9]: data.isnull().sum()
Out[9]:
age          0
sex          307
on_thyroxine 0
query_on_thyroxine 0
on_antithyroid_meds 0
sick          0
pregnant     0
thyroid_surgery 0
lithium       0
goitre        0
tumor          0
hypopituitary 0
psych          0
TSH           842
T3            2604
TT4           442
T4U           809
FTI           802
TBG           8823
target        6935
dtype: int64

In [10]: data.dropna(subset=['target'], inplace=True)

In [11]: data['target'].value_counts()
Out[11]:
hypothyroid conditions    593
general health             436
binding protein              376
replacement therapy          335
miscellaneous                201
hyperthyroid conditions     182
antithyroid treatment        33
Name: target, dtype: int64

In [12]: data.head()
Out[12]:
   age sex on_thyroxine query_on_thyroxine on_antithyroid_meds sick pregnant thyroid_surgery l131_treatment query_hypothyroid ... tumor hypopituitary psych      TSH      T3      TT4      T4U      FTI      TBG      target
0  42   F         f             f           f       f       f       f       f       f   ...   f       f       f       f       NaN      NaN      NaN      NaN      NaN      36.0  miscellaneous
1  63   F         t             f           f       t       f       f       f       f   ...   f       f       f       f       68.000000  NaN      48.0      1.02     47.0      NaN  hypothyroid conditions
2  41   M         f             f           f       f       f       f       f       f   ...   f       f       f       f       0.050000  1.6      39.0      1.00     39.0      NaN
3  71   F         t             f           f       f       f       f       f       f   ...   f       f       f       f       0.050000  NaN     126.0      1.38     91.0      NaN  binding protein
4  55   F         t             f           f       f       f       f       f       f   ...   f       f       f       f       9.599999  2.4      136.0      1.48     92.0      NaN  replacement therapy

5 rows × 23 columns
```



```
In [13]: data.describe()
Out[13]:
   age      TSH      T3      TT4      T4U      FTI      TBG
count  2237.000000  2087.000000  1643.000000  2140.000000  2059.000000  2060.000000  98.000000
mean   52.792579  14.930791  1.961875  116.390495  1.015439  120.363369  47.717347
std    19.677450  46.204092  1.452238  60.351600  0.280222  70.996728  32.398750
min    1.000000  0.050000  2.000000  0.170000  1.000000  9.289999
25%   36.000000  0.285000  1.000000  76.000000  0.850000  83.000000  32.000000
50%   56.000000  2.000000  1.700000  109.000000  0.960000  109.000000  36.000000
75%   69.000000  8.799999  2.500000  156.000000  1.120000  157.000000  46.750000
max   95.000000  630.000000  18.000000  600.000000  2.330000  881.000000  200.000000
```



```
In [14]: #Checking whether the age above 100
data[data.age>100]
Out[14]:
   age sex on_thyroxine query_on_thyroxine on_antithyroid_meds sick pregnant thyroid_surgery l131_treatment query_hypothyroid ... tumor hypopituitary psych      TSH      T3      TT4      T4U      FTI      TBG      target
0      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...      ...
9153  64.0   M         f             f           f       f       f       f       f       f   ...   f       f       f       f       0.810000  NaN      31.0      0.58      56.0      NaN  general health
9157  60.0   M         f             f           f       t       f       f       f       f   ...   f       f       f       f       0.180000  NaN      28.0      0.87     32.0      NaN  general health
9158  64.0   M         f             f           f       f       f       f       f       f   ...   f       f       f       f       NaN      NaN      44.0      0.53     83.0      NaN  binding protein
9162  36.0   F         f             f           f       f       f       f       f       f   ...   f       f       f       f       NaN      NaN      84.0      1.26     67.0      NaN  binding protein
9169  69.0   M         f             f           f       f       f       f       f       f   ...   f       f       f       f       NaN      NaN      113.0     1.27     89.0      NaN  binding protein

0 rows × 23 columns
```



```
In [15]: import numpy as np
#changing age of observation with(age>100) to null
data['age']=np.where((data.age>100), np.nan, data.age)
In [16]: data
Out[16]:
   age      TSH      T3      TT4      T4U      FTI      TBG
count  2237.000000  2087.000000  1643.000000  2140.000000  2059.000000  2060.000000  98.000000
mean   52.792579  14.930791  1.961875  116.390495  1.015439  120.363369  47.717347
std    19.677450  46.204092  1.452238  60.351600  0.280222  70.996728  32.398750
min    1.000000  0.050000  2.000000  0.170000  1.000000  9.289999
25%   36.000000  0.285000  1.000000  76.000000  0.850000  83.000000  32.000000
50%   56.000000  2.000000  1.700000  109.000000  0.960000  109.000000  36.000000
75%   69.000000  8.799999  2.500000  156.000000  1.120000  157.000000  46.750000
max   95.000000  630.000000  18.000000  600.000000  2.330000  881.000000  200.000000
```



```
2237 rows × 23 columns
```

```
#splitting the data values as x and y
In [19]: #splitting the data values as x and y
x=data.iloc[:,0:-1]
y = data.iloc[:,-1]

In [20]: data.isnull().sum()

Out[20]:
age          0
sex          98
on_thyroxine 0
query_on_thyroxine 0
on_antithyroid_meds 0
sick          0
pregnant     0
thyroid_surgery 0
I131_treatment 0
query_hypothyroid 0
query_hyperthyroid 0
lithium      0
goitre        0
tumor          0
hypopituitary 0
psych          0
TSH          159
T3           594
TT4          97
T4U          178
FTI          177
TBG          2139
target        0
dtype: int64

In [21]: x['sex'].unique()

Out[21]: array(['F', 'M', nan], dtype=object)

In [22]: x['sex'].replace(np.nan, 'F', inplace=True)

In [23]: x['sex'].value_counts()

Out[23]:
F    1701
M    536
Name: sex, dtype: int64
```

```
In [24]: x.isnull().sum()

Out[24]:
age          0
sex          0
on_thyroxine 0
query_on_thyroxine 0
on_antithyroid_meds 0
sick          0
pregnant     0
thyroid_surgery 0
I131_treatment 0
query_hypothyroid 0
query_hyperthyroid 0
lithium      0
goitre        0
tumor          0
hypopituitary 0
psych          0
TSH          158
T3           594
TT4          97
T4U          178
FTI          177
TBG          2139
dtype: int64

In [25]: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2237 entries, 4 to 9169
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype  
 ____ _____
 0   age              2237 non-null   float64
 1   sex              2141 non-null   object 
 2   on_thyroxine     2237 non-null   object 
 3   query_on_thyroxine 2237 non-null   object 
 4   on_antithyroid_meds 2237 non-null   object 
 5   sick             2237 non-null   object 
 6   pregnant         2237 non-null   object 
 7   thyroid_surgery  2237 non-null   object 
 8   I131_treatment   2237 non-null   object 
 9   query_hypothyroid 2237 non-null   object 
 10  query_hyperthyroid 2237 non-null   object 
 11  lithium          2237 non-null   object 
 12  goitre           2237 non-null   object 
 13  tumor            2237 non-null   object 
 14  hypopituitary    2237 non-null   object 
 15  psych             2237 non-null   object 
 16  TSH              2088 non-null   float64
 17  T3               1643 non-null   float64
 18  TT4              2140 non-null   float64
 19  T4U              2089 non-null   float64
 20  FTI              2060 non-null   float64
 21  TBG              98 non-null    float64
 22  target            2237 non-null   object 
dtypes: float64(7), object(16)
memory usage: 419.4+ KB
```

```
In [26]: x['age']=x['age'].astype('float')
x['TSH']=x['TSH'].astype('float')
x['TT4']=x['TT4'].astype('float')
x['T4U']=x['T4U'].astype('float')
x['FTI']=x['FTI'].astype('float')
x['TBG']=x['TBG'].astype('float')

converting categorical to numerical values

In [27]: #applying ordinal_encoding to x values
#Encoding the categorical data
#Encoding the independent(output) variable
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
#categorical data

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

In [28]: x.head()

Out[28]:
   age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  I131_treatment  query_hypothyroid ... goitre  tumor  hypopituitary  psych  TSH  T3  TT4  T4U  FTI  TBG
4  32.0  0           0                  0          0       0       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       36.0
18 63.0  0           1                  0          0       1       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
32 41.0  1           0                  0          0       0       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
33 71.0  0           1                  0          0       0       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
39 55.0  0           1                  0          0       0       0        0        0        0 ... 1       0       0       0       0       0       0       0       0       0       0       0       0       0       0

5 rows × 22 columns

In [29]: x.replace(np.nan, '0', inplace=True)
x.head()

Out[29]:
   age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  I131_treatment  query_hypothyroid ... goitre  tumor  hypopituitary  psych  TSH  T3  TT4  T4U  FTI  TBG
4  32.0  0           0                  0          0       0       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       36.0
18 63.0  0           1                  0          0       1       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
32 41.0  1           0                  0          0       0       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
33 71.0  0           1                  0          0       0       0        0        0        0 ... 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
39 55.0  0           1                  0          0       0       0        0        0        0 ... 1       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0

5 rows × 22 columns

In [30]: #applying label_encoding to y values
label_encoder = LabelEncoder()
y_dt=label_encoder.fit_transform(y)


```

```
In [31]: y=pd.DataFrame(y_dt, columns=['target'])

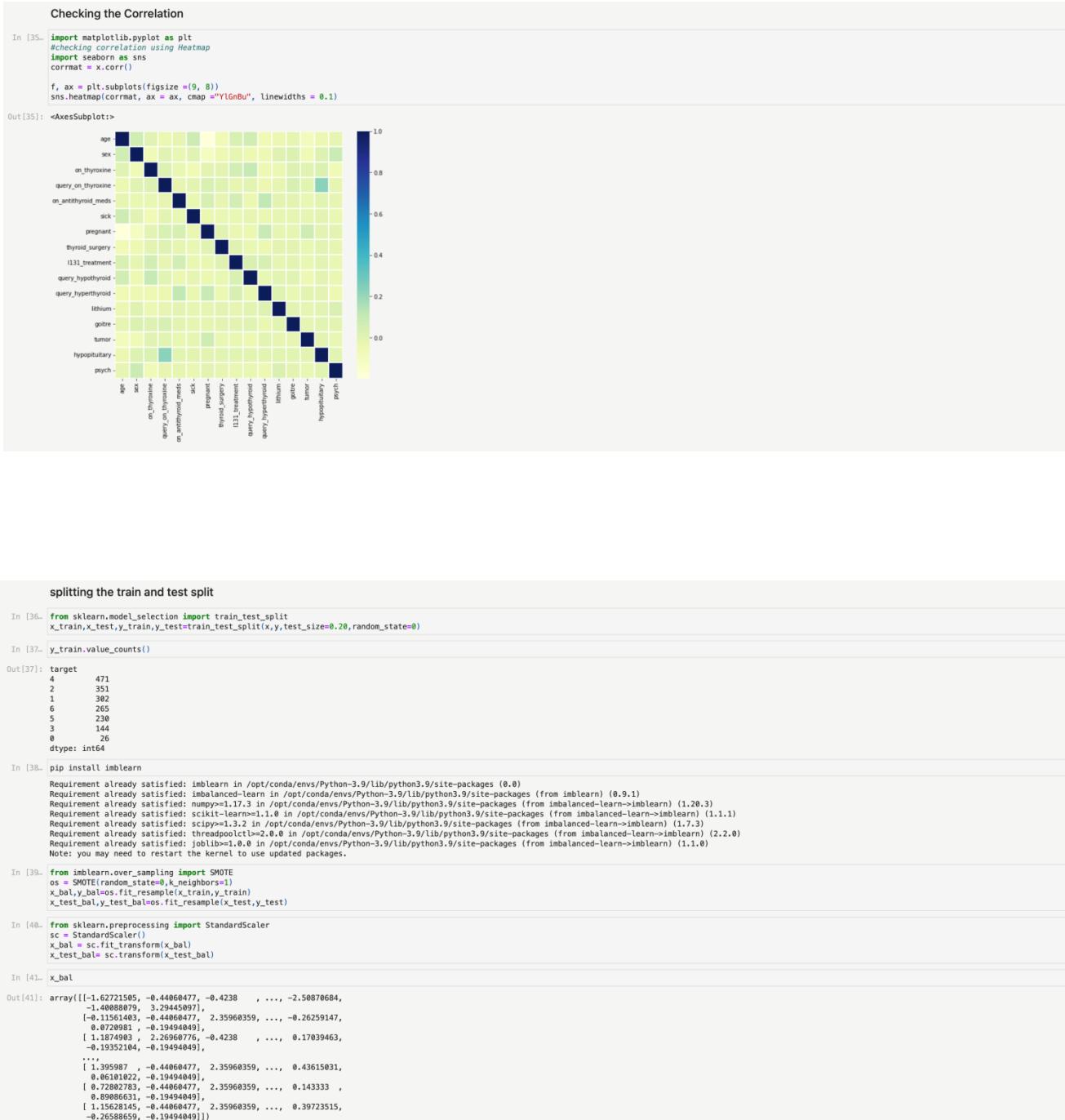
Out[31]:
   target
0      5
1      4
2      5
3      1
4      6
...
2232    2
2233    2
2234    1
2235    1
2236    1

2237 rows × 1 columns

In [32]: y.value_counts(normalize=True)

Out[32]:
target
0    0.265987
2    0.194984
1    0.168882
6    0.150201
5    0.125615
3    0.0811359
0    0.014752
dtype: float64

In [ ]:
```



```
In [42]: x_test_bal
Out[42]: array([-1.5229667, -0.44060477, -0.4238, ..., 1.06342846,
   0.3974753, -0.19494049], [-0.3974753, -0.44060477, -0.4238, ..., 1.76703086,
   -0.30218342, -0.19494049], [-0.9496898, 2.26968776, -0.4238, ..., -0.39789962,
   -0.98586329, -0.19494049], ...,
   [1.39013447, -0.44060477, 2.35960359, ..., 0.81835453,
   0.700094189, -0.19494049], [1.33846247, -0.44060477, 2.35960359, ..., 0.81987378,
   0.67327619, -0.19494049], [-0.19842352, -0.44060477, -0.4238, ..., 0.24830842,
   0.37610348, -0.19494049]])
In [43]: y_bal.value_counts()
Out[43]: target
0    471
1    471
2    471
3    471
4    471
5    471
6    471
dtype: int64
In [44]: columns=['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','T3','TT4','T4U','FTI','TBG']
In [45]: x_test_bal=pd.DataFrame(x_test_bal,columns=columns)
In [46]: x_bal=pd.DataFrame(x_bal,columns=columns)
In [47]: x_bal
Out[47]:
   age   sex on_thyroxine query_on_thyroxine on_antithyroid_meds  sick pregnant thyroid_surgery I131_treatment query_hypothyroid ... goitre tumor hypopituitary psych      TSH      T3      TT4      T4U     FTI      TBG
0   0.1627215 -0.440605 -0.423800 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.315458 -1.035358 -1.704935 -2.508707 -1.400881 3.2944
1   0.115614 -0.440605  2.359604 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.090056 0.155233 -0.197223 -0.262591 0.072098 -0.19494
2   0.187490  2.269608 -0.423800 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.278907 -0.471394 -0.227079 0.170398 -0.193521 -0.19494
3   -0.1366594 -0.440605 -0.423800 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 7.283487 -0.024637 -0.107982 -0.284999 0.969848 0.041622 0.495134 -0.133153 -0.19494
4   -0.167738 -0.440605 -0.423800 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.306321 4.541622 1.459767 -0.127283 1.496783 -0.19494
... ...
3292  0.546923 -0.440605  2.359604 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.114424 0.343221 -0.148122 -0.146517 0.040168 -0.19494
3293  0.383062 -0.440605  2.359604 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.309176 -0.856640 0.565143 -0.513902 1.085434 -0.19494
3294  0.1395897 -0.440605  2.359604 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.095452 -0.172405 0.248906 0.436150 0.061010 -0.19494
3295  0.728028 -0.440605  2.359604 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.311566 0.087864 1.071643 0.143333 0.890866 -0.19494
3296  1.156281 -0.440605  2.359604 -0.105069 -0.158703 -0.141815 -0.137297 -0.239601 -0.162675 -0.230986 ... -0.052319 -0.137297 -0.024637 -0.107982 -0.072439 0.079407 -0.200359 0.397235 -0.265887 -0.19494
3297 rows x 22 columns
```

```
In [48]: from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
rfr = RandomForestClassifier(n_estimators=100)
rfr.fit(x_bal,y_bal)
y_pred = rfr.predict(x_test_bal)
accuracy_score(y_bal,y_pred)
x_bal.shape,y_bal.shape,x_test_bal.shape,y_test_bal.shape

/tmp/wuser/ipykernel_1028/2696972469.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rfr = RandomForestClassifier().fit(x_bal,y_bal)
Out[48]: ((3297, 22), (3297, 1), (854, 22), (854, 1))

In [49]: test_score=accuracy_score(y_test_bal,y_pred)
test_score

Out[49]: 0.9988360655737705

In [50]: train_score = accuracy_score(y_bal,rfr.predict(x_bal))
train_score

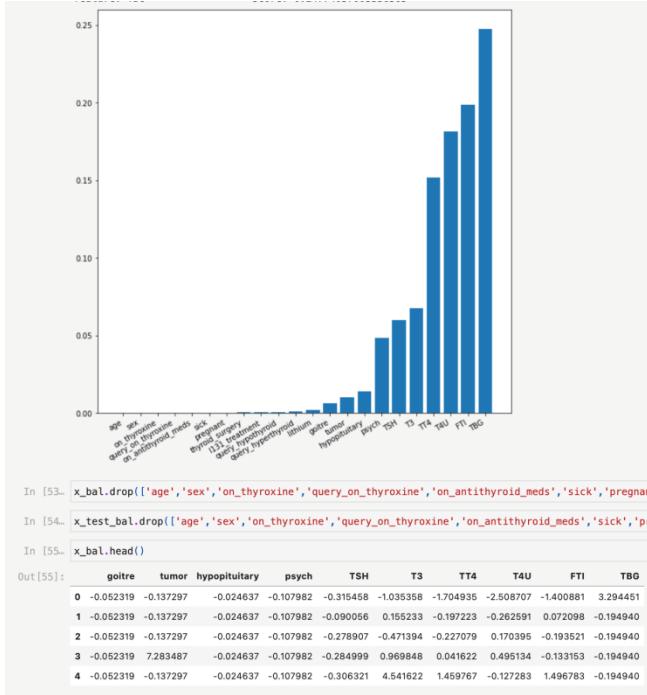
Out[50]: 1.0
```

performing feature importance

```
In [51]: #perform feature importance
from sklearn.inspection import permutation_importance
results = permutation_importance(rfr,x_bal,y_bal, scoring='accuracy')

In [52]: #gets importance
feature_importances['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','T3','TT4','T4U','FTI','TBG']
importance = results.importances_mean
importance = np.sort(importance)
#summarize feature importance
for i, in enumerate(importance):
  if feature_importances[i]:
    print('feature: {} Score: {}'.format(i,v))
plt.figure(figsize=(10,10))
plt.bar(*feature_importance, height = importance)
plt.xticks(rotation=90, ha='right')
plt.show()

feature: age          Score: 0.0
feature: sex          Score: 0.0
feature: on_thyroxine Score: 0.0
feature: query_on_thyroxine Score: 0.0
feature: on_antithyroid_meds Score: 0.0
feature: sick          Score: 0.0
feature: pregnant      Score: 6.066120715801926e-05
feature: thyroid_surgery Score: 0.008363967242948115541
feature: I131_treatment Score: 0.00048528965726415141
feature: query_hypothyroid Score: 0.0006066120715801926
feature: query_hyperthyroid Score: 0.0012132241431684962
feature: lithium        Score: 0.001819836214746688
feature: goitre         Score: 0.006308765544434333
feature: tumor          Score: 0.010138421595389748
feature: hypopituitary  Score: 0.010138421595389748
feature: psych          Score: 0.0465289572641796
feature: TSH             Score: 0.05987261146496816
feature: T3              Score: 0.0678192296266989
feature: TT4             Score: 0.1519563239308462
feature: T4U             Score: 0.181619654311192
feature: FTI             Score: 0.1988474370639975
feature: TBG             Score: 0.247740370033336363
```



```

In [56]: x_test_bal.head()

Out[56]:
      goitre   tumor  hypopituitary    psych     TSH      T3      TT4     T4U     FTI     TBG
0 -0.052319 -0.137297 -0.024637 -0.107982 -0.312412  0.593872  0.788014  1.063428  0.132466 -0.19494
1 -0.052319 -0.137297 -0.024637 -0.107982 -0.314240  0.781860  0.444674  1.767031 -0.302183 -0.19494
2 -0.052319 -0.137297 -0.024637 -0.107982  1.298911 -0.408731 -1.227244 -0.397900 -0.905863 -0.19494
3 -0.052319 -0.137297 -0.024637 -0.107982 -0.168205 -0.471394 -0.227079 -0.397900  0.132466 -0.19494
4 -0.052319 -0.137297 -0.024637 -0.107982 -0.227125 -0.346068 -0.301718 -0.830886  0.434306 -0.19494

RandomForest Model-1

In [57]: rf1 = RandomForestClassifier()
rf1.fit(x_bal,y_bal)
y_pred=rf1.predict(x_test_bal)

/tmp/wuser/pykerne_1020/1228087459.py:2: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
rf1.fit(x_bal,y_bal)

In [58]: print(classification_report(y_test_bal,y_pred))
      precision    recall  f1-score   support
          0       0.81      0.17      0.28      122
          1       0.81      0.93      0.87      122
          2       0.93      0.98      0.96      122
          3       0.76      0.84      0.88      122
          4       0.48      0.87      0.61      122
          5       0.87      0.69      0.77      122
          6       0.58      0.50      0.54      122

      accuracy                           0.71      854
     macro avg       0.75      0.71      0.69      854
  weighted avg       0.75      0.71      0.69      854

In [59]: train_score = accuracy_score(y_bal,rf1.predict(x_bal))

In [60]: train_score

Out[60]: 1.0

```

SVC Model-3

```
In [62]: from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

sv=SVC()

In [63]: sv.fit(x_bal,y_bal)

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)

Out[63]: SVC()
SVC()

In [64]: y_pred=sv.predict(x_test_bal)

In [65]: print(classification_report(y_test_bal,y_pred))

precision    recall   f1-score   support
          0       0.70      0.85      0.77     122
          1       0.76      0.81      0.79     122
          2       0.88      0.93      0.90     122
          3       0.71      0.65      0.68     122
          4       0.71      0.63      0.67     122
          5       0.76      0.54      0.63     122
          6       0.49      0.57      0.52     122

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

In [66]: train_score=accuracy_score(y_bal,sv.predict(x_bal))
train_score

Out[66]: 0.7154989384288747
```

Random_Search for SVC

```
In [67]: params = {

    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf','sqrt']

}

In [68]: from sklearn.model_selection import RandomizedSearchCV
random_svc = RandomizedSearchCV(sv,params, scoring='accuracy',cv=5,n_jobs=-1)

In [69]: random_svc.fit(x_bal,y_bal)

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
y = column_or_1d(y, warn=True)
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/sklearn/utils/validation.py:1111: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example by using ravel().
```



```

In [11]: from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "4a9YB_3PvNDBV0mekwXaJuLoP_kIRhjOYcUVwFBQzv"
}
client = APIClient(wml_credentials)

In [11]: def guid_from_space_name(client, space_name):
    space = client.spaces.get_details()
    #print(space)
    return(next(item for item in space['resources'] if item['entity']['name'] == space_name) ['metadata']['id'])

In [12]: space_uid = guid_from_space_name(client, space_name="thyroid_deploy")
print("Space UID = " + space_uid)
Space UID = 77c033bf-40dc-4e25-8afb-5bea7f65de10

In [12]: client.set.default_space(space_uid)

Out[12]: 'SUCCESS'

In [12]: client.software_specifications.list()

NAME          ASSET_ID          TYPE
default_py3.6 0062bb9c-8b7d-44a0-a9b9-46c416adcb9  base
kernel-spark3.2-scala2.12 020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt 069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6   09c5a1d8-9c1e-4473-a344-eb7b65ff687  base
spark-mllib_3.0-scala2.12 09d0a03a-5a0a-4a0a-8a0a-0a0a0a0a0a0a  base
pytorch-omnix_rtt22.1-py3.9 0b3484dd-e681-5590-9ed1-05f16fccc6471  base
ai-function_0.1-py3.6   0cd8b91e-537e-4f4d-02dd-dab6b29a99d8  base
shiny-3.6        0e6e79df-875e-4f24-8ae9-62a2cc21a48386  base
tensorflow_2.4-py3.7-horovod 1092590a-307d-453d-9b62-4eb7d6403722  base
pytorch_1.1-py3.6   10a0c126e-6039-4cc0-8392-3e922c096892  base
tensorflow_1.15-py3.6-ddl 111e41b3-dc24-5422-a4d6-f1776828ab7  base
runtime-22.1-py3.6   12b83a17-24db-5082-900f-9a31fbfd3cb  base
scikit-learn_0.22-py3.6 154010fa-5b3b-412aaf-4d9ee5abbc85  base
default_3.6       1b70a0ec3-a034-4a87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6 1b853719e-cf01-5a0a-98e7-39c3880dbb7  base
pytorch-omnix_rtt22.1-py3.9-edt 1b925984-d661-5d0e-5b65-3f1df1665666  base
tensorflow_2.1-py3.6   20047f72-0498-58c7-0ff5-77b012eb9f5  base
tensorflow_2.4-py3.8-horovod 217c15f6-0781-56bf-f23a-119f20564c49  base
runtime-22.1-py3.9-cuda 26215f05-08c3-5a41-1b10-d66306ce658  base
do_py3.8         295addb5-9ef9-547e-9b74-92ae3563e720  base
autoai-ts_3.8-py3.8 2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6 2b73a275-7cbf-4200-a912-eae77436e6bc  base
pytorch_1.2-py3.6   2b8ef57d-2687-4b7d-acce-01194976da1  base
spark-mllib_2.3   2e51f700-bca0-4b0d-88d5-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt 32983cea-3f32-440b-8965-dde874a86d7e  base
spark-mllib_3.0-py37 36591beb-8710-56bf-92a0-7e7a0500e9  base
spark-mllib_3.0-py37 389071e0-5580-4f5c-9e5-07c7e056503  base
xgboost_0.62-py3.6 39e31a0c-5130-41dc-aed4-60233c09306e  base
pytorch-onnx_1.1-py3.6-edt 4058940e-7010-4e28-8da0-f0b3b614fe12  base
default_3.6-py3.8 41c247d3-4519-5a71-9b65-8580229facf0  base
autoai-ts_rtt22.1-py3.9 4269d26e-07ba-5d40-8f66-2d495ba0c71f7  base
autoai-obm_3.0   42b92c18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0.4.3    493cb095-16f1-5bc5-bee8-81ba8f80e9e7  base
spark-mllib 2.4-r 3.6 49403dff-92e9-4c87-a3d7-a42d0021c095  base

```

```

In [12]: import sklearn
sklearn.__version__
Out[12]: '1.1.1'

In [12]: software_spec_uid= client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
software_spec_uid

Out[12]: '12b83a17-24d8-5082-900f-0ab31fbfd3cb'

In [13]: model_details = client.repository.store_model(model='thyroid_disease_new.tgz',
meta_props={client.repository.ModelMetaNames.NAME:"thyroiddeploy",
client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid
},
training_data=x_train,
training_target=y_train)

In [13]: model_id = client.repository.get_model_id(model_details)
model_id

Out[13]: '2a67d325-0fcf-471c-9365-70472e8446a4'

In [13]: # Deploy
deployment = client.deployments.create(
artifact_uid=model_id,
meta_props={client.deployments.ConfigurationMetaNames.NAME:"thyroiddeploy_deploy",
client.deployments.ConfigurationMetaNames.ONLINE: {}})
#####
# Synchronous deployment creation for uid: '2a67d325-0fcf-471c-9365-70472e8446a4' started
#####

initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.
ready

Successfully finished deployment creation, deployment_uid='93c0fb0f-c7b4-4dbd-a280-8fb0bcee2d8d'

```

