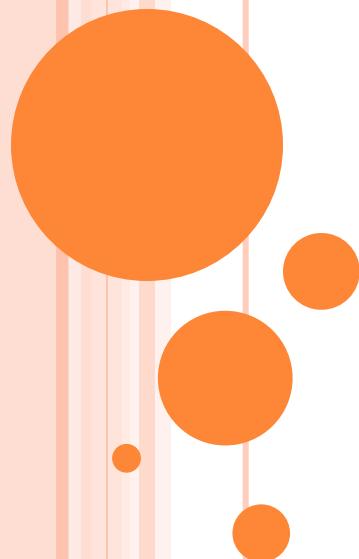


THYROID DISEASE CLASSIFICATION USING MACHINE LEARNING ALGORITHMS



DONE BY

K.AGALYA(20326ER039)

A.AMEERA BANU(20326ER040)

P.ANUPRIYA(20326ER041)

A.ARUL AKILA(20326ER042)

CONTENT

- INTRODUCTION
- PROBLEM DEFINITION & DESIGN THINKING
- RESULT
- ADVANTAGES
- APPLICATIONS
- CONCLUSION
- FUTURE SCOPE



ABSTRACT

- ❖ The Thyroid gland is a vascular gland and one of the most important organs of the human body. This gland secretes two hormones which help in controlling the metabolism of the body.
- ❖ The two types of Thyroid disorders are Hyperthyroidism and Hypothyroidism. When this disorder occurs in the body, they release certain types of hormones into the body which imbalances the body's metabolism.
- ❖ Machine Learning plays a very deciding role in disease prediction. Machine Learning algorithms, SVM - support vector machine, Random Forest Classifier, XGB Classifier and ANN - Artificial Neural Networks are used to predict the patient's risk of getting thyroid disease. The web app is created to get data from users to predict the type of disease.

INTRODUCTION

Overview

- ❖ The Thyroid gland is a vascular gland and one of the most important organs of the human body. This gland secretes two hormones which help in controlling the metabolism of the body. The two types of Thyroid disorders are Hyperthyroidism and Hypothyroidism.
- ❖ When this disorder occurs in the body, they release certain types of hormones into the body which imbalances the body's metabolism.
- ❖ Machine Learning algorithms, SVM - support vector machine, Random Forest Classifier, XGB Classifier and ANN - Artificial Neural Networks are used to predict the patient's risk of getting thyroid disease. The web app is created to get data from users to predict the type of disease.

PURPOSE

- ❖ Machine learning algorithms can help improve the accuracy and efficiency of thyroid disease diagnosis by analyzing large amounts of data from diverse sources, such as medical records, laboratory tests, and imaging studies.
- ❖ These algorithms can learn from the data and develop models that can accurately classify different types of thyroid diseases based on their features and characteristics.
- ❖ The use of machine learning algorithms for thyroid disease classification can also help reduce the time and cost involved in diagnosis, leading to faster and more efficient patient care.



PROBLEM DEFINITION AND DESIGN THINKING

Empathy map

The empathy map is a tool used in design thinking and customer experience design to better understand the needs and experiences of customers or users. It is a simple framework that helps teams develop empathy for their users by visualizing their thoughts, feelings, and behaviors.



EMPATHY MAP

Template



Empathy map canvas

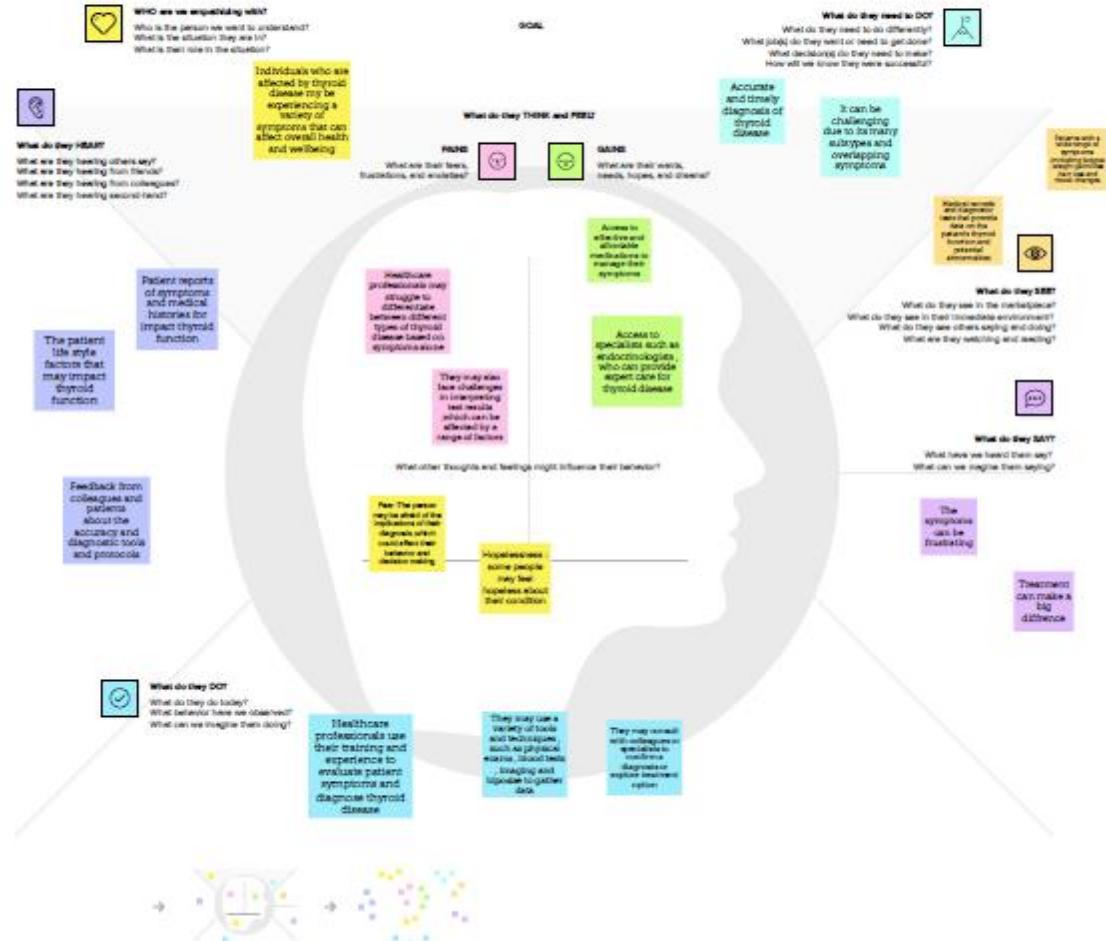
Empathy map canvas for Thyroid
disease classification using machine
learning

Originally created by Dave Gray at



Develop shared understanding and empathy

Summarize the data you have gathered related to the people that are impacted by your work. It will help you generate ideas, prioritize features, or discuss decisions.



IDEATION & BRAINSTORMING MAP

- ❖ Brainstorming is a process of generating creative and innovative ideas through group discussion and collaboration. It is typically used in the early stages of a project or problem-solving process to generate a wide range of ideas without judgment or criticism.
- ❖ Set clear goals and objectives: Before starting the brainstorming session, it's essential to define the problem or challenge you want to address and set clear goals and objectives for the session.
- ❖ Encourage participation: Encourage all participants to contribute their ideas and opinions, no matter how unconventional or outlandish they may seem. The goal is to generate as many ideas as possible.
- ❖ Avoid criticism and judgment: Criticism and judgment can stifle creativity and prevent participants from sharing their ideas. Encourage participants to build on each other's ideas and avoid negative feedback.





Brainstorm & Idea prioritization

Thyroid Disease Classification Using Machine Learning

- 10 minutes to prepare
- 1 hour to collaborate
- 3-6 people recommended



[Share template feedback](#)





Before collaborate

This project present thyroid disease prediction using various machine learning algorithms based on the

⌚ 10 minutes

atributes to obtain high accuracy.

A Team gathering

Totally, Four participation are there in this session . we invite members through mural link and gathered in this session.

B Set the goal

This project present thyroid disease prediction using various machine learning algorithms based on the atributes to obtain high accuracy.

C Learn how to use the facilitation tools

Facilitation tools can be very helpful for guiding discussions and brainstorming sessions .

Open article →



1

Problem statement

1.The Thyroid Disease can be easily Identify based on symptoms In the patient's history.

⌚ 5 minutes

2.Thyroid disease Is a medical condition that affects the function of the thyroid gland

3.This project present thyroid disease prediction using various machine learning algorithms based on the attributes to obtain high accuracy

4.That dataset create for this project Is apply from the kaggle thyroid disease dataset.

5.The Important of feature show be estimated to select the option number of features thyroid disease classification.



2

Brainstorm

Here some Ideas

⌚ 10 minutes

Person 1

Hardware Requirement are used	16 GB of RAM is used
use the machine learning to identify the different types of thyroid	Artificial neural networks are used

Person 2

Machine Learning algorithms are used	SVM algorithms is used
Python Libraries are Imported	python language is used to increase the accuracy of thyroid disease

Person 3

Google colab it used for creashare and run the program	XGB classifier is used
Operating system is required	Internet and storage are needed

Person 4

Random forest algorithms are used	Build a HTML pages
Thyroid disease dataset is used	



4

Prioritize

Prioritize the ideas

30 minutes



Feasibility

Regardless of their impressiveness, which tools are more feasible than others? Cost, time, effort, complexity, etc.





After collaborate

we can export the mural as pdf to share . It is helpful to getting information.



Importing the libraries and data preprocessing

The screenshot shows a Google Colab notebook titled "thyroid.ipynb". The code cell contains imports for pandas, numpy, matplotlib.pyplot, tensorflow, and tensorflow.keras. The next cell shows the command to read a CSV file named "thyroidDF.csv" into a DataFrame named "data". The final cell displays the first five rows of the DataFrame using the head() method.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Layer,Dense,Dropout

data=pd.read_csv("/content/sample_data/thyroidDF.csv")

data.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	TT4	T4U_measured	T4	
0	29	F	f	f	f	f	f	f	f	f	t	...	NaN	f	Na
1	29	F	f	f	f	f	f	f	f	f	f	...	128.0	f	Na
2	41	F	f	f	f	f	f	f	f	f	f	...	NaN	f	Na
3	36	F	f	f	f	f	f	f	f	f	f	...	NaN	f	Na
4	32	F	f	f	f	f	f	f	f	f	f	...	NaN	f	Na

5 rows x 31 columns

archive (1)

File Home Share View

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=2N1VEKxt_J1i

Gmail YouTube Maps

+ Code + Text

data.shape
(9172, 31)

data.isnull().sum()

archive (1)

File Home Share View

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=d5aDgul2_Nyg

Gmail YouTube Maps

+ Code + Text

data.isnull().sum()

os	age 0
{x}	sex 387
	on_thyroxine 0
	query_on_thyroxine 0
	on_antithyroid_meds 0
	sick 0
	pregnant 0
	thyroid_surgery 0
	I131_treatment 0
	query_hypothyroid 0
	query_hyperthyroid 0
	lithium 0
	goitre 0
	tumor 0
	hypopituitary 0
	psych 0
	TSH_measured 0
	TSH 842
	T3_measured 0
	T3 2664
	TT4_measured 0
	TT4 442
	T4U_measured 0
	T4U 889
	FTI_measured 0
	FTI 882
	TBG_measured 0
	TRG 8873

Type here to search

36°C ENG 12:55 09-04-2023

archive (1)

File Home Share View

thyroid.ipynb - Colaboratory x +

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=tZb6luOLAS6q

Gmail YouTube Maps

+ Code + Text

✓ [15] data.dropna(subset=['target'],inplace=True)

(x) ✓ [16] data['target'].value_counts()

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

✓ [17] data=data[data.age<=100]

(x) ✓ [18] x=data.iloc[:,0:-1]
y=data.iloc[:, -1]

archive (1)

File Home Share View

thyroid.ipynb - Colaboratory x +

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=GEdDQyzn_nzG

Gmail YouTube Maps

+ Code + Text

✓ [13] data.drop(['TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG_measured','referral_source','patient_id'],axis=1,inplace=True)

(x) ✓ [14] diagnoses={
'A': 'hyperthyroid conditions',
'B': 'hyperthyroid conditions',
'C': 'hyperthyroid conditions',
'D': 'hyperthyroid conditions',
'E': 'hyperthyroid conditions',
'F': 'hypothyroid conditions',
'G': 'hypothyroid conditions',
'H': 'hypothyroid conditions',
'I': 'binding protein',
'J': 'binding protein',
'K': 'general health',
'L': 'replacement therapy',
'M': 'replacement therapy',
'N': 'replacement therapy',
'O': 'antithyroid treatment',
'P': 'antithyroid treatment',
'Q': 'antithyroid treatment',
'R': 'miscellaneous',
'S': 'miscellaneous',
'T': 'miscellaneous'}
data['target']=data['target'].map(diagnoses)

Type here to search

36°C ENG 12:55 09-04-2023

Splitting the data x and y

The screenshot shows a Jupyter Notebook interface on Google Colab. The code cell contains the following:

```
[15] data.dropna(subset=['target'], inplace=True)  
[16] data['target'].value_counts()  
hypothyroid conditions    593  
general health            436  
binding protein             376  
replacement therapy         336  
miscellaneous                281  
hyperthyroid conditions     182  
antithyroid treatment          33  
Name: target, dtype: int64  
[17] data=data[data.age<=100]  
[18] x=data.iloc[:, :-1]  
y=data.iloc[:, -1]
```

The output pane shows the result of the value counts command from cell 16.

The screenshot shows a Jupyter Notebook interface on Google Colab. The code cell contains the following:

```
[18] x,y = data.iloc[:, :-1], data.iloc[:, -1]
```

The output pane displays a portion of the thyroid dataset as a DataFrame:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor	hypop
4	32	F	f	f	f	f	f	f	f	f	...	f	f	f
18	63	F	t	f	f	t	f	f	f	f	...	f	f	f
32	41	M	f	f	f	f	f	f	f	f	...	f	f	f
33	71	F	t	f	f	f	f	f	f	f	...	f	f	f
39	55	F	t	f	f	f	f	f	f	f	...	f	f	f
...
9153	64	M	f	f	f	f	f	f	f	f	...	f	f	f
9157	60	M	f	f	t	f	f	f	f	f	...	f	f	f
9158	64	M	f	f	f	f	f	f	f	f	...	f	f	f
9162	36	F	f	f	f	f	f	f	f	f	...	f	f	f
9169	69	M	f	f	f	f	f	f	f	f	...	f	f	f

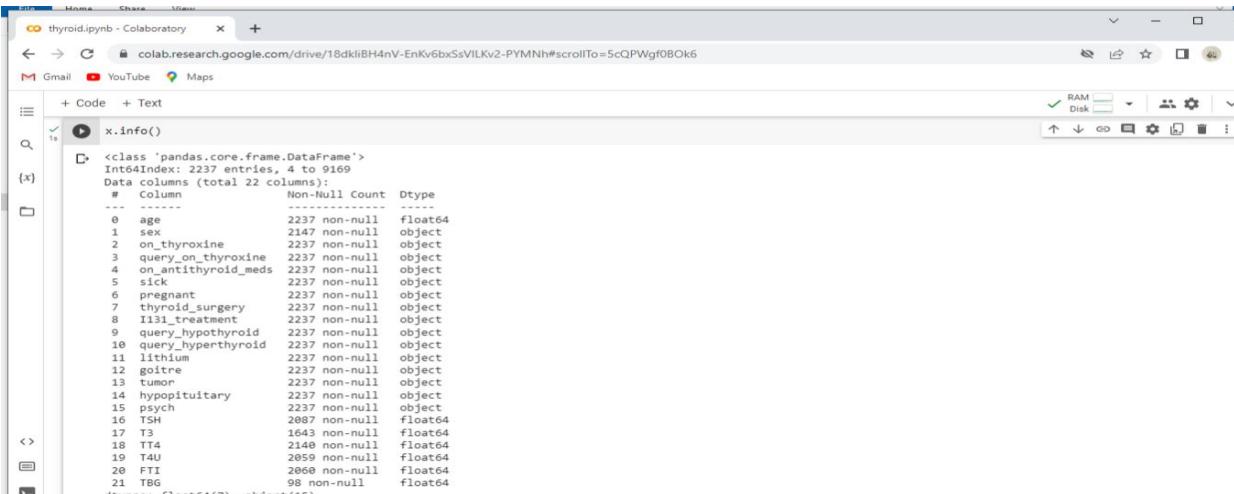
2237 rows × 22 columns

Converting the data types



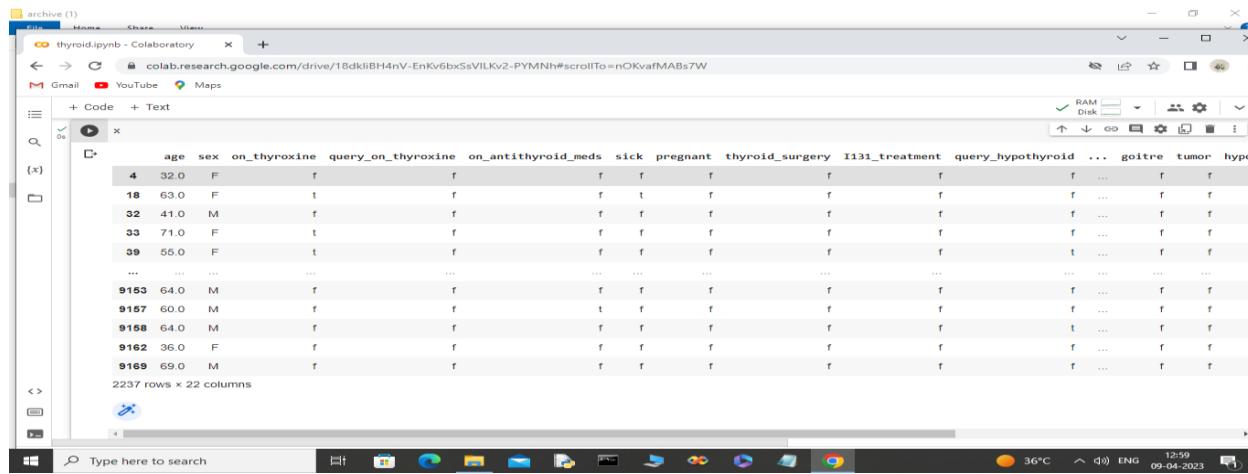
```
[20] x['sex'].value_counts()
F    1611
M     536
Name: sex, dtype: int64
```

```
x['age']=x['age'].astype('float')
x['TSH']=x['TSH'].astype('float')
x['T3']=x['T3'].astype('float')
x['T4']=x['T4'].astype('float')
x['T4U']=x['T4U'].astype('float')
x['FTI']=x['FTI'].astype('float')
x['TBG']=x['TBG'].astype('float')
```



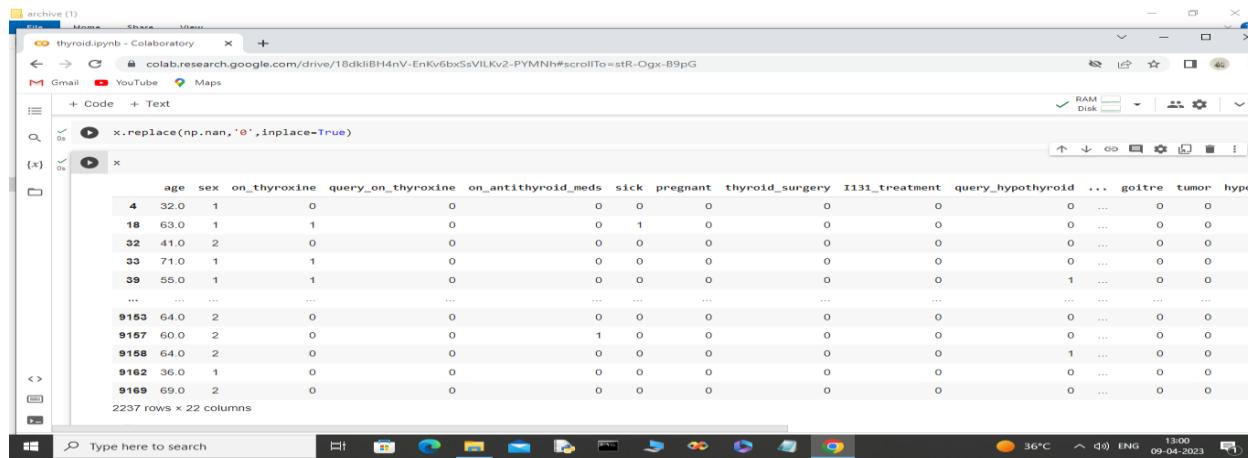
```
x.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2237 entries, 4 to 9169
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype  
--- 
 0   age            2237 non-null    float64
 1   sex            2147 non-null    object 
 2   on_thyroxine   2237 non-null    object 
 3   query_on_thyroxine 2237 non-null    object 
 4   on_antithyroid_meds 2237 non-null    object 
 5   sick           2237 non-null    object 
 6   pregnant       2237 non-null    object 
 7   thyroid_surgery 2237 non-null    object 
 8   I131_treatment 2237 non-null    object 
 9   query_hypothyroid 2237 non-null    object 
 10  query_hyperthyroid 2237 non-null    object 
 11  ltm             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  T4              2149 non-null    float64
 19  T4U             2059 non-null    float64
 20  FTI             2060 non-null    float64
 21  TBG             98 non-null     float64
dtypes: float64(7), object(15)
```

Handling categorical values



A screenshot of a Google Colab notebook titled "thyroid.ipynb - Colaboratory". The notebook displays a Pandas DataFrame with 2237 rows and 22 columns. The columns represent various medical and demographic variables, many of which are categorical. The first few rows show:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor	hypo
4	32.0	F	f	f	f	f	f	f	f	f	...	f	f	f
18	63.0	F	t	f	f	f	t	f	f	f	...	f	f	f
32	41.0	M	f	f	f	f	f	f	f	f	...	f	f	f
33	71.0	F	t	f	f	f	f	f	f	f	...	f	f	f
39	55.0	F	t	f	f	f	f	f	f	f	...	f	f	f
...
9153	64.0	M	f	f	f	f	f	f	f	f	...	f	f	f
9157	60.0	M	f	f	t	f	f	f	f	f	...	f	f	f
9158	64.0	M	f	f	f	f	f	f	f	f	...	f	f	f
9162	36.0	F	f	f	f	f	f	f	f	f	...	f	f	f
9169	69.0	M	f	f	f	f	f	f	f	f	...	f	f	f



A screenshot of a Google Colab notebook titled "thyroid.ipynb - Colaboratory". The notebook displays the same Pandas DataFrame as the previous screenshot, but with all categorical values replaced by numerical values. The code cell above the DataFrame contains the command `x.replace(np.nan, '0', inplace=True)`. The first few rows show the transformed data:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor	hypo
4	32.0	1	0	0	0	0	0	0	0	0	...	0	0	0
18	63.0	1	1	0	0	0	1	0	0	0	...	0	0	0
32	41.0	2	0	0	0	0	0	0	0	0	...	0	0	0
33	71.0	1	1	0	0	0	0	0	0	0	...	0	0	0
39	55.0	1	1	0	0	0	0	0	0	0	...	1	0	0
...
9153	64.0	2	0	0	0	0	0	0	0	0	...	0	0	0
9157	60.0	2	0	0	1	0	0	0	0	0	...	0	0	0
9158	64.0	2	0	0	0	0	0	0	0	0	...	1	0	0
9162	36.0	1	0	0	0	0	0	0	0	0	...	0	0	0
9169	69.0	2	0	0	0	0	0	0	0	0	...	0	0	0

A screenshot of a Jupyter Notebook interface running on Google Colab. The notebook has two cells:

```
[30]: from sklearn.preprocessing import OrdinalEncoder,LabelEncoder  
ordinal_encoder=OrdinalEncoder(dtype='int64')  
x.iloc[:, 1:16]=ordinal_encoder.fit_transform(x.iloc[:, 1:16])
```

The output of the first cell is a preview of a DataFrame:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor	hypo	
4	32.0	1	0	0	0	0	0	0	0	0	0	...	0	0	0
18	63.0	1	1	0	0	0	1	0	0	0	0	0	...	0	0
32	41.0	2	0	0	0	0	0	0	0	0	0	0	...	0	0
33	71.0	1	1	0	0	0	0	0	0	0	0	0	...	0	0
39	55.0	1	1	0	0	0	0	0	0	0	1	...	0	0	0
...
9153	64.0	2	0	0	0	0	0	0	0	0	0	0	...	0	0
9157	60.0	2	0	0	0	1	0	0	0	0	0	0	...	0	0
9158	64.0	2	0	0	0	0	0	0	0	0	1	...	0	0	0
9162	36.0	1	0	0	0	0	0	0	0	0	0	0	...	0	0
9169	69.0	2	0	0	0	0	0	0	0	0	0	0	...	0	0

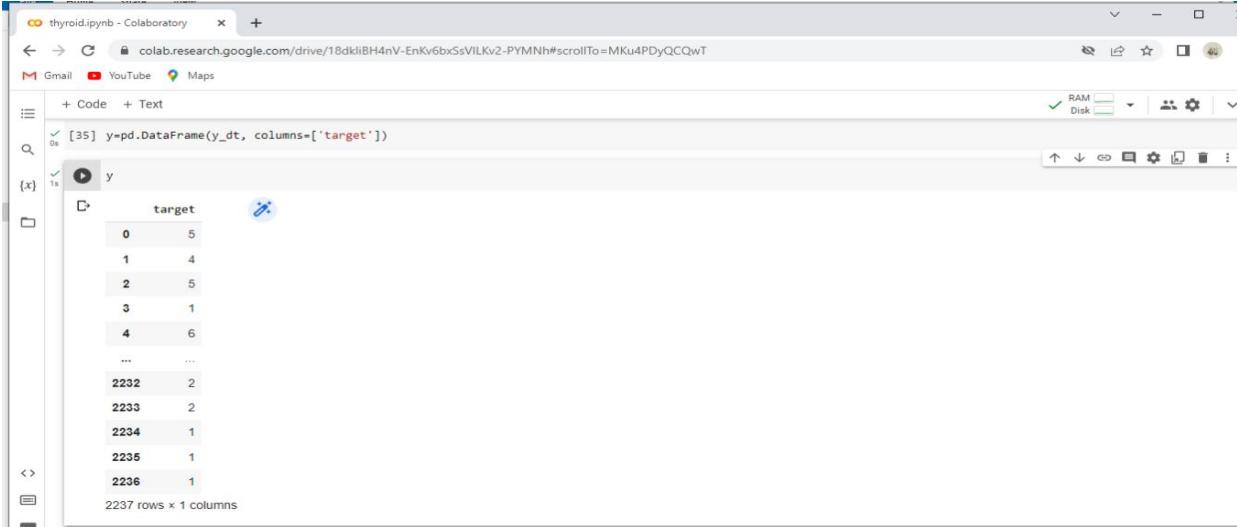
2237 rows x 22 columns

A screenshot of a Jupyter Notebook interface running on Google Colab. The notebook has two cells:

```
[34]:  
label_encoder=LabelEncoder()  
y_dt=label_encoder.fit_transform(y)
```

```
[35]:  
y=pd.DataFrame(y_dt, columns=['target'])
```

Splitting data into train and test

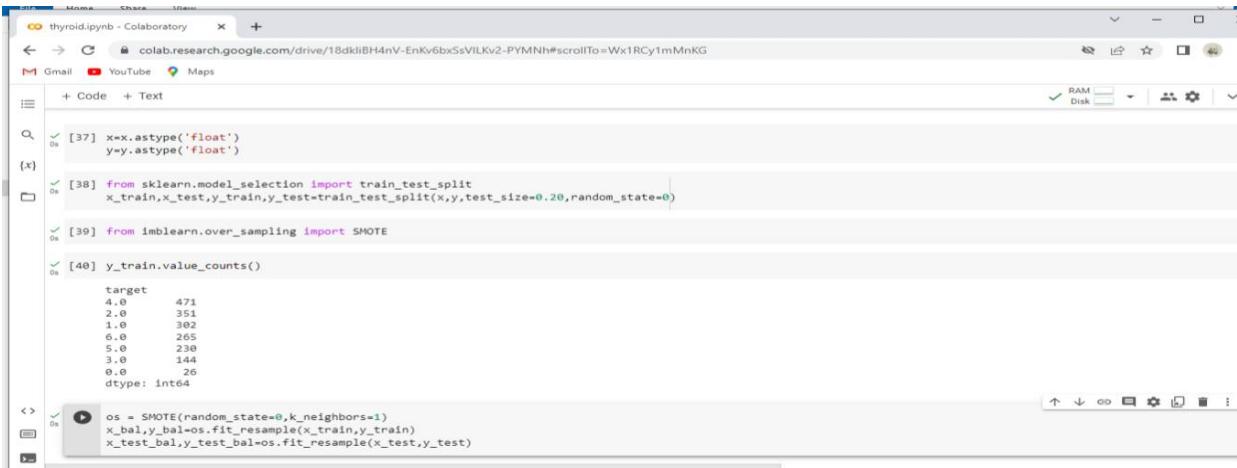


thyroid.ipynb - Colaboratory

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

{x} {y}

	target
0	5
1	4
2	5
3	1
4	6
...	...
2232	2
2233	2
2234	1
2235	1
2236	1
2237	rows x 1 columns



thyroid.ipynb - Colaboratory

```
[37]: x=x.astype('float')
       y=y.astype('float')
```

{x}

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

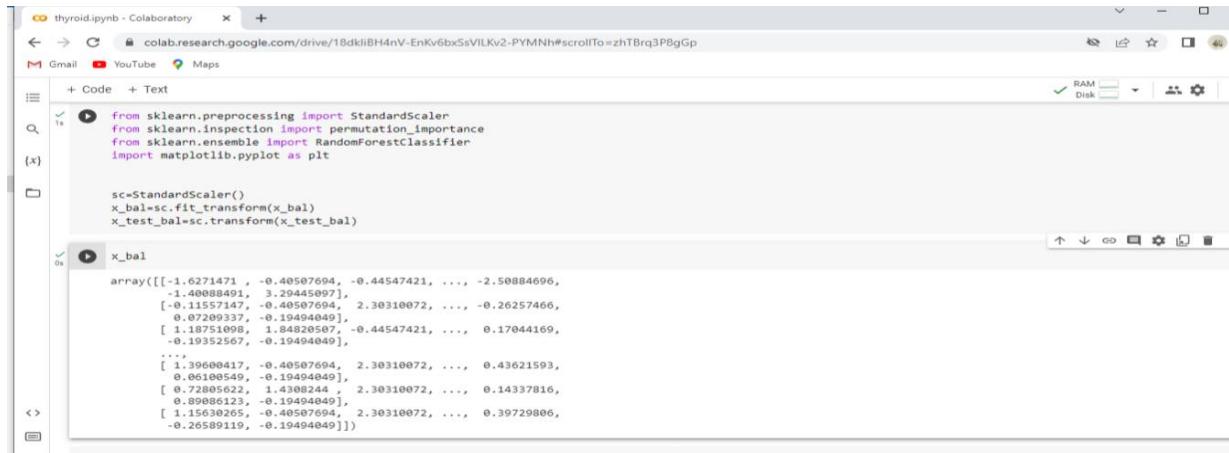
```
[39]: from imblearn.over_sampling import SMOTE
```

```
[40]: y_train.value_counts()
```

target	count
4.0	471
2.0	351
1.0	302
6.0	265
5.0	230
3.0	144
0.0	26
	dtype: int64

```
[41]: os = SMOTE(random_state=0,k_neighbors=1)
       x_bal,y_bal=os.fit_resample(x_train,y_train)
       x_test_bal,y_test_bal=os.fit_resample(x_test,y_test)
```

Handling imbalanced data

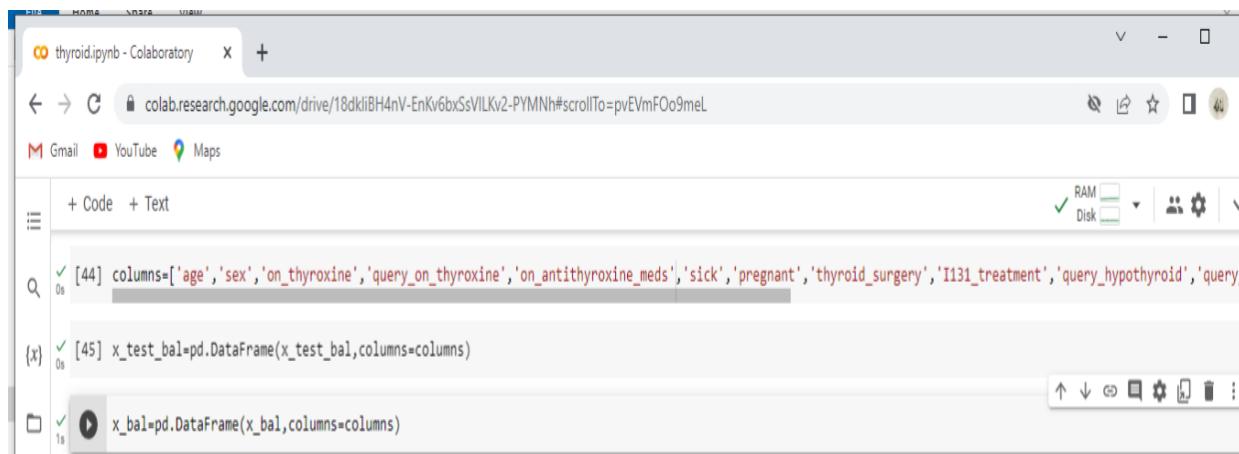


```
from sklearn.preprocessing import StandardScaler
from sklearn.inspection import permutation_importance
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

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

x_bal
```

```
array([[-1.6271471, -0.40507694, -0.44547421, ..., -2.50884696,
       -1.39600417, -0.40507694, ..., -0.44547421, ..., -2.50884696,
       [-1.1157147, -0.40507694, 2.30310072, ..., -0.26257466,
       0.07209337, -0.19494049],
       [ 1.18751098, -1.84820507, -0.44547421, ..., 0.17044169,
       -0.19352567, -0.19494049],
       ...,
       [ 1.39600417, -0.40507694, 2.30310072, ..., 0.43621593,
       0.05180549, -0.19494049],
       [ 0.72080562, -1.4308244, 2.30310072, ..., 0.14337816,
       0.89086123, -0.19494049],
       [ 1.15630265, -0.40507694, 2.30310072, ..., 0.39729806,
       -0.26589119, -0.19494049]])
```



```
[44] columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroxine_meds','sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid','query_pregnant']

{x} [45] x_test_bal=pd.DataFrame(x_test_bal,columns=columns)

{x} [46] x_bal=pd.DataFrame(x_bal,columns=columns)
```

thyroid.ipynb - Colaboratory

[colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=TF0Br-nt94Dv](#)

Gmail YouTube Maps

RAM Disk

x_bal

(x) 3297 rows × 22 columns

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroxine_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...
0	-1.627147	-0.405077	-0.445474	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
1	-0.115571	-0.405077	2.303101	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
2	1.187511	1.848205	-0.445474	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
3	-1.366531	-0.405077	-0.445474	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
4	-0.167695	-0.405077	-0.445474	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
...
3292	0.546954	0.1047459	2.303101	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
3293	0.383096	-0.405077	2.303101	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
3294	1.396004	-0.405077	2.303101	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
3295	0.728056	1.430824	2.303101	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	-0.304166	...
3296	1.156303	-0.405077	2.303101	-0.141477	-0.257609	-0.177156	-0.174387	-0.267762	-0.224295	1.322228	...

thyroid.ipynb - Colaboratory

[colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=sDCG55iBeO2i](#)

Gmail YouTube Maps

RAM Disk

```
rfr=RandomForestClassifier()
rfr.fit(x_bal,y_bal)

<ipython-input-48-571bdcc8bfaa:2> DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), (n_samples, 1) or (n_samples, n_outputs)!
```

```
[RandomForestClassifier
RandomForestClassifier()]

[1]: from sklearn.inspection import permutation_importance
results=permutation_importance(rfr,x_bal,y_bal,scoring='accuracy')
feature_importance=[`age`,`sex`,`on_thyroxine`,`query_on_thyroxine`,`on_antithyroxine_meds`,`sick`,`pregnant`,`thyroid_surgery`,`I131_treatment`,`query_hypothyroid`]
importance=results['importances_mean']
importance=np.sort(importance)
#summarize feature importance
for i,v in enumerate(importance):
    i-feature_importance[1]
    print('feature: ({<20}) Score: {}'.format(i,v))
plt.figure(figsize=(10,10))
plt.bar(x=feature_importance, height=importance)
plt.xticks(rotation=30, ha='right')
plt.show()
```

thyroid.ipynb - Colaboratory

File Home Share View

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILkv2-PYMNh#scrollTo=g4umFmvxNu3w

Gmail YouTube Maps

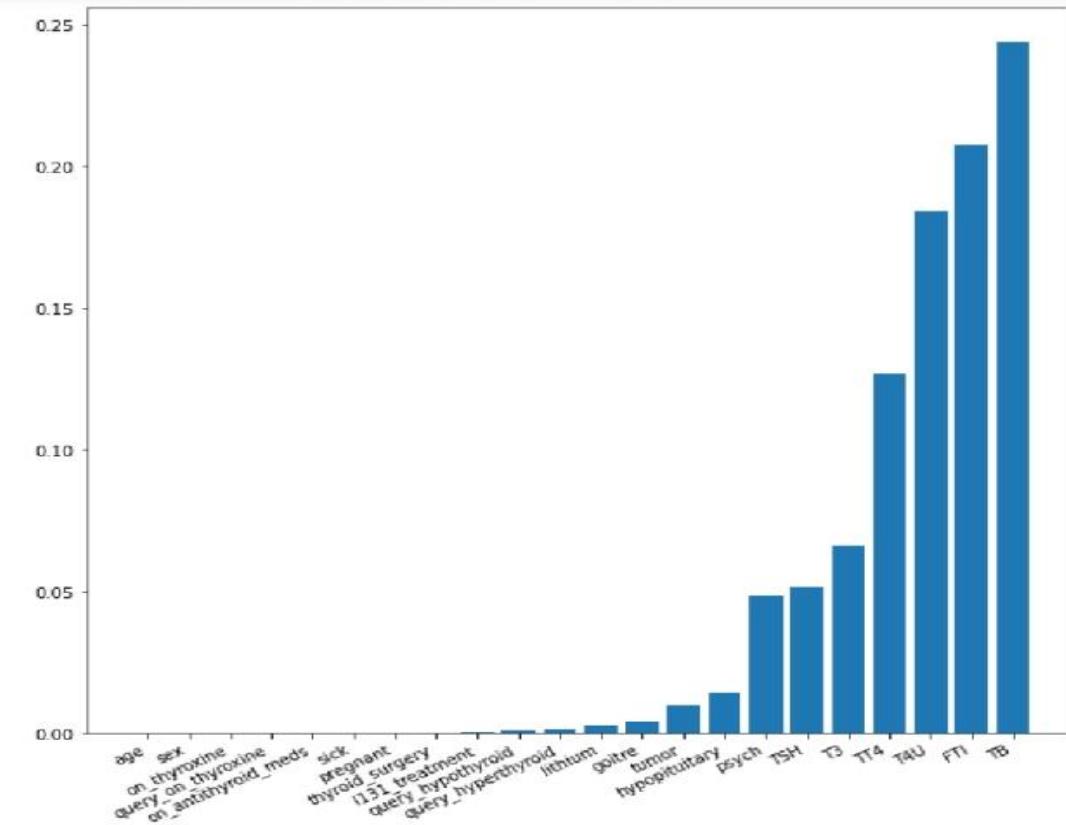
+ Code + Text

8s

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

{x}

feature	Score
feature: age	0.0
feature: sex	0.0
feature: on_thyroxine	0.0
feature: query_on_thyroxine	0.0
feature: on_antithyroxine_meds	0.00012132241431603852
feature: sick	0.0003033060357900963
feature: pregnant	0.0003033060357900963
feature: thyroid_surgery	0.006666120715801926
feature: I131_treatment	0.0087279344858962311
feature: query_hypothyroid	0.0012738853503185155
feature: query_hyperthyroid	0.0018198362147406888
feature: lithium	0.0026690931149529586
feature: goitre	0.00357901122232336
feature: tumor	0.016560509554140124
feature: hypopituitary	0.016681831968456164
feature: psych	0.019654231119199263
feature: TSH	0.05538368213527449
feature: T3	0.060843190779496494
feature: TT4	0.08905065210797694
feature: T4U	0.1575978161965423
feature: FTI	0.17488626023657872
feature: TBG	0.22942068547164088



Selecting Output Columns

x.head()

	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	FTI	TBG
4	32.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
18	63.0	1.0	1.0	0.0	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	0.0	0.0	
32	41.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
33	71.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
39	55.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows x 22 columns

[50]

```
[51] x_bal=x_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid'])
```

[52] x_test_bal=x_test_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surgery','I131_treatment','query_hypothyroid'])

x_bal.head()

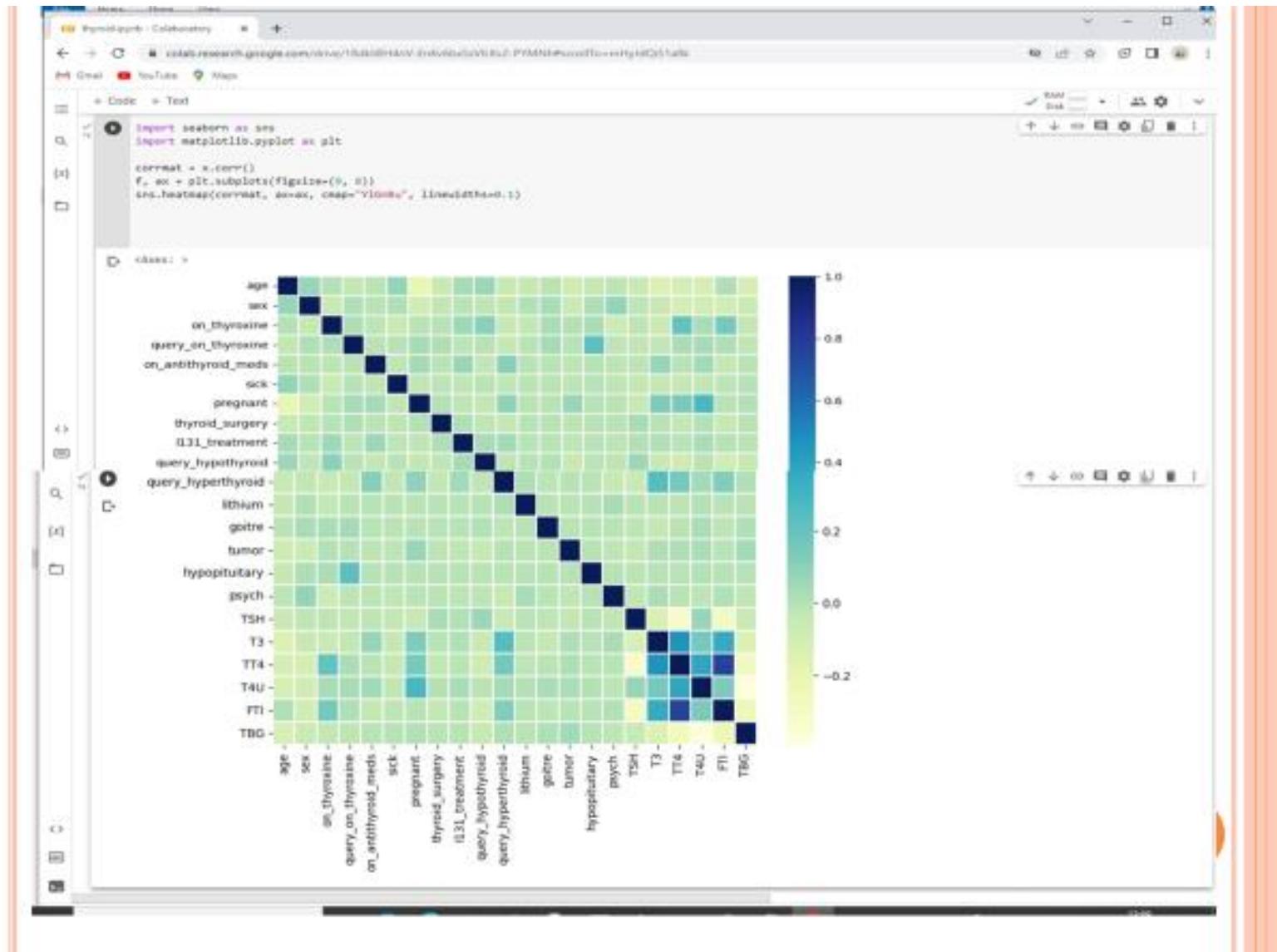
	goitre	tumor	hypopituitary	psych	TSH	T3	TT4	T4U	FTI	TBG
0	-0.073367	-0.185723	-0.037741	-0.144603	-0.315459	-1.035381	-1.704953	-2.508847	-1.400885	3.294451
1	-0.073367	-0.185723	-0.037741	-0.144603	-0.090056	0.154862	-0.197213	-0.262575	0.072093	-0.194940
2	-0.073367	-0.185723	-0.037741	-0.144603	-0.278907	-0.471581	-0.227069	0.170442	-0.193526	-0.194940
3	-0.073367	6.169673	-0.037741	-0.144603	-0.284999	0.969239	0.041637	0.495204	-0.133158	-0.194940
4	-0.073367	-0.185723	-0.037741	-0.144603	-0.306321	4.539969	1.459807	-0.127257	1.496777	-0.194940

Exploratory Data Analysis

```
+ Code + Text
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    int64  
 1   sex            2237 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            2887 non-null    float64 
 17  T3              1643 non-null    float64 
 18  TT4             2140 non-null    float64 
 19  T4U             2059 non-null    float64 
 20  FTI             2868 non-null    float64 
 21  TBG             98 non-null     float64 
 22  target          2237 non-null    object  
dtypes: float64(6), int64(1), object(16)
memory usage: 419.44 KB
```

```
+ Code + Text
[54]: x.head()
      age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  I131_treatment  query_hypothyroid  ...  goitre  tumor  hypopi
 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
18  63.0  0.0        1.0            0.0            0.0    1.0      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
33  71.0  0.0        1.0            0.0            0.0    0.0      0.0        0.0            0.0            0.0  ...  0.0    0.0
39  55.0  0.0        1.0            0.0            0.0    0.0      0.0        0.0            0.0            0.0  ...  1.0    0.0
```

Visual Analysis



Model building

A screenshot of a Google Colab notebook titled "thyroid.ipynb - Colaboratory". The code cell contains the following Python code:

```
[59] rfr1 = RandomForestClassifier()
      rfr1.fit(x_bal, y_bal)

<ipython-input-59-813284120830>:2: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), (n_samples, 1) or (n_samples, n_features). See the documentation for more information.
      rfr1.fit(x_bal, y_bal)
  * RandomForestClassifier
  RandomForestClassifier()

0s y_pred = rfr1.predict(x_test_bal)
```

The code uses the `RandomForestClassifier` from scikit-learn to fit a model on the training data `x_bal` and `y_bal`. A warning is displayed about the shape of the `y` variable. The prediction is stored in `y_pred`.

A screenshot of a Google Colab notebook titled "thyroid.ipynb - Colaboratory". The code cell contains the following Python code:

```
from sklearn.metrics import classification_report
print(classification_report(y_test_bal, y_pred))
```

The output shows a classification report for the model's performance:

	precision	recall	f1-score	support
0.0	0.75	0.07	0.13	122
1.0	0.82	0.95	0.88	122
2.0	0.93	0.98	0.96	122
3.0	0.77	0.84	0.80	122
4.0	0.45	0.86	0.59	122
5.0	0.91	0.71	0.80	122
6.0	0.59	0.53	0.56	122
accuracy		0.71	0.71	854
macro avg	0.74	0.71	0.67	854
weighted avg	0.74	0.71	0.67	854

XGB Classifier model

0s [62] from sklearn.metrics import accuracy_score
train_score = accuracy_score(y_bal, rfr1.predict(x_bal))
(x) train_score
1.0

2s [63] from xgboost import XGBClassifier
xgb1 = XGBClassifier()
xgb1.fit(x_bal,y_bal)

* XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None, feature_types=None,
gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
interaction_constraints=None, learning_rate=None, max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan, monotone_constraints=None,
n_estimators=100, n_jobs=None, num_parallel_tree=None,
objective='multi:softprob', predictor=None, ...)

thyroid.ipynb - Colaboratory

← → ⌂ colab.research.google.com/drive/18dkIiBH4nV-EkV6bxSsVlKv2-PYMNh#scrollTo=4Hzv4cCSxy5B

Gmail YouTube Maps

+ Code + Text

RAM Disk

0s [64] y_pred = xgb1.predict(x_test_bal)

{x} 0s [65] print(classification_report(y_test_bal,y_pred))

	precision	recall	f1-score	support
0.0	0.80	0.29	0.42	122
1.0	0.82	0.93	0.87	122
2.0	0.96	1.00	0.98	122
3.0	0.77	0.84	0.88	122
4.0	0.52	0.84	0.64	122
5.0	0.84	0.72	0.78	122
6.0	0.60	0.52	0.56	122
accuracy		0.74	0.74	854
macro avg	0.76	0.74	0.72	854
weighted avg	0.76	0.74	0.72	854

0s [66] accuracy_score(y_test_bal,y_pred)
C 0.7353629976580797

SVC Model

```
[67] from sklearn.svm import SVC
      from sklearn.metrics import accuracy_score, classification_report

      svm = SVC()
      svm.fit(x_bal, y_bal)

      /usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
      y = column_or_1d(y, warn=True)
      SVC()
      SVC()

0s  y_pred = svm.predict(x_test_bal)
```

```
[69] print(classification_report(y_test_bal, y_pred))

          precision    recall  f1-score   support

          0.0       0.70      0.85      0.77      122
          1.0       0.78      0.81      0.80      122
          2.0       0.88      0.92      0.90      122
          3.0       0.73      0.70      0.72      122
          4.0       0.73      0.63      0.68      122
          5.0       0.83      0.53      0.65      122
          6.0       0.51      0.62      0.56      122

      accuracy                           0.72      854
      macro avg       0.74      0.72      0.72      854
      weighted avg    0.74      0.72      0.72      854

0s  train_score=accuracy_score(y_bal,svm.predict(x_bal))
      train_score
      0.7209584470730968
```

ANN Model

The screenshot shows a Google Colab notebook titled "thyroid.ipynb". The code cell contains the following Python code for creating a Sequential model:

```
[71] model= Sequential()
[72] model.add(Dense(units=128,activation='relu',input_shape=(10,)))
[73] model.add(Dense(units=128,activation='relu',kernel_initializer='random_uniform'))
model.add(Dropout(0.2))
model.add(Dense(units=256,activation='relu',kernel_initializer='random_uniform'))
model.add(Dropout(0.2))
model.add(Dense(units=128,activation='relu',kernel_initializer='random_uniform'))
[74] model.add(Dense(units=1,activation='sigmoid'))
```

The cell [74] is currently selected and has a play button icon next to it, indicating it is ready to be run.

The screenshot shows the output of the `model.summary()` command, which displays the architecture of the Sequential model:

```
Model: "sequential"
-----  
Layer (type)      Output Shape       Param #  
-----  
dense (Dense)    (None, 128)        1408  
dense_1 (Dense)  (None, 128)        16512  
dropout (Dropout) (None, 128)        0  
dense_2 (Dense)  (None, 256)        33024  
dropout_1 (Dropout) (None, 256)        0  
dense_3 (Dense)  (None, 128)        32896  
dense_4 (Dense)  (None, 1)          129  
-----  
Total params: 83,969  
Trainable params: 83,969  
Non-trainable params: 0
```

A screenshot of a Jupyter Notebook interface running in a browser window. The notebook is titled "thyroid.ipynb - Colaboratory". The code cell at the top contains the command `model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])`. Below the code cell, the output shows the compilation of the model. The interface includes a sidebar with "Code" and "Text" tabs, a toolbar with various icons, and a status bar at the bottom.

A screenshot of a Jupyter Notebook interface running in a browser window. The notebook is titled "thyroid.ipynb - Colaboratory". The code cell at the top contains the command `model.fit(x_bal,y_bal,validation_data=[x_test_bal,y_test_bal],epochs=15)`. Below the code cell, the output shows the training progress over 15 epochs. The output text is very long and repetitive, showing the loss and accuracy for each epoch from 1 to 15. The interface includes a sidebar with "Code" and "Text" tabs, a toolbar with various icons, and a status bar at the bottom.

archive (1)

File Home Share View

∞ thyroid.ipynb - Colaboratory x +

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=4L1-qjW9OGrl

Gmail YouTube Maps

+ Code + Text

✓ RAM Disk

10s

```
model.fit(x_bal,y_bal,validation_data=[x_test_bal,y_test_bal],epochs=15)

Epoch 1/15
104/104 [=====] - 2s 7ms/step - loss: -18190.3691 - accuracy: 0.1429 - val_loss: -144140.0000 - val_accuracy: 0.1429
Epoch 2/15
104/104 [=====] - 0s 4ms/step - loss: -2581241.7500 - accuracy: 0.1429 - val_loss: -10009973.0000 - val_accuracy: 0.1429
Epoch 3/15
104/104 [=====] - 0s 4ms/step - loss: -41421256.0000 - accuracy: 0.1429 - val_loss: -110557968.0000 - val_accuracy: 0.1429
Epoch 4/15
104/104 [=====] - 0s 5ms/step - loss: -278188128.0000 - accuracy: 0.1429 - val_loss: -577503168.0000 - val_accuracy: 0.1429
Epoch 5/15
104/104 [=====] - 0s 4ms/step - loss: -1107758336.0000 - accuracy: 0.1429 - val_loss: -1990348288.0000 - val_accuracy: 0.1429
Epoch 6/15
104/104 [=====] - 0s 4ms/step - loss: -3223061504.0000 - accuracy: 0.1429 - val_loss: -5254848512.0000 - val_accuracy: 0.1429
Epoch 7/15
104/104 [=====] - 0s 4ms/step - loss: -7711963136.0000 - accuracy: 0.1429 - val_loss: -11729270784.0000 - val_accuracy: 0.1429
Epoch 8/15
104/104 [=====] - 1s 7ms/step - loss: -16032121856.0000 - accuracy: 0.1429 - val_loss: -22895458304.0000 - val_accuracy: 0.1429
Epoch 9/15
104/104 [=====] - 1s 7ms/step - loss: -29828251648.0000 - accuracy: 0.1429 - val_loss: -41082445824.0000 - val_accuracy: 0.1429
Epoch 10/15
104/104 [=====] - 1s 6ms/step - loss: -51529527296.0000 - accuracy: 0.1429 - val_loss: -68933918720.0000 - val_accuracy: 0.1429
Epoch 11/15
104/104 [=====] - 1s 7ms/step - loss: -83809009664.0000 - accuracy: 0.1429 - val_loss: -109124739072.0000 - val_accuracy: 0.1429
Epoch 12/15
104/104 [=====] - 0s 4ms/step - loss: -128870727680.0000 - accuracy: 0.1429 - val_loss: -164966252544.0000 - val_accuracy: 0.1429
Epoch 13/15
104/104 [=====] - 0s 4ms/step - loss: -191465357312.0000 - accuracy: 0.1429 - val_loss: -240482549760.0000 - val_accuracy: 0.1429
Epoch 14/15
104/104 [=====] - 0s 4ms/step - loss: -273940873216.0000 - accuracy: 0.1429 - val_loss: -340156907520.0000 - val_accuracy: 0.1429
```

Type here to search

36°C ENG 13:12 09-04-2023

Testing the model

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell at the top contains three predictions using different classifiers:

```
[78] rfr1.predict([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])  
[79] svm.predict([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])  
[80] col= ['goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']  
da = [[0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]]  
da1 = pd.DataFrame(data = da, columns=col)  
xgb1.predict(da1)
```

The output cell below shows the prediction results:

```
array([4])  
model.predict([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])  
1/1 [=====] - 0s 121ms/step  
array([[1.]], dtype=float32)
```

The status bar at the bottom indicates the system temperature is 36°C, the date is 09-04-2023, and the time is 13:12.

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell at the top contains a prediction:

```
[81] model.predict([[0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])
```

The output cell below shows the prediction result:

```
array([[1.]], dtype=float32)
```

The code cell at the bottom prints a classification report:

```
[82] print(classification_report(y_test_bal,y_pred))
```

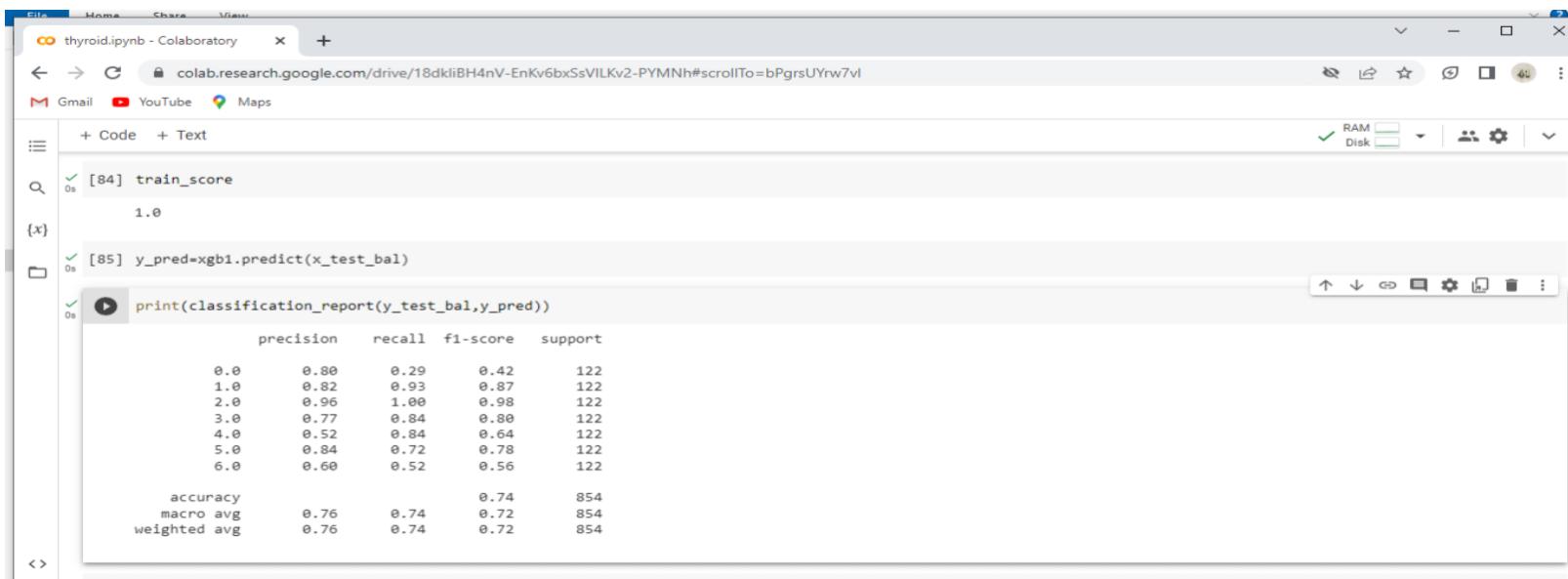
	precision	recall	f1-score	support
0.0	0.70	0.85	0.77	122
1.0	0.78	0.81	0.80	122
2.0	0.88	0.92	0.90	122
3.0	0.73	0.70	0.72	122
4.0	0.73	0.63	0.68	122
5.0	0.83	0.53	0.65	122
6.0	0.51	0.62	0.56	122

	accuracy	macro avg	weighted avg	
accuracy		0.72	0.72	854
macro avg	0.74	0.72	0.72	854
weighted avg	0.74	0.72	0.72	854

The final code cell at the bottom calculates the train score:

```
[83] train_score = accuracy_score(y_bal,rfr1.predict(x_bal))
```

Compare Model



thyroid.ipynb - Colaboratory

File Home Share View

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=bPgrsUYrw7vl

Gmail YouTube Maps

+ Code + Text

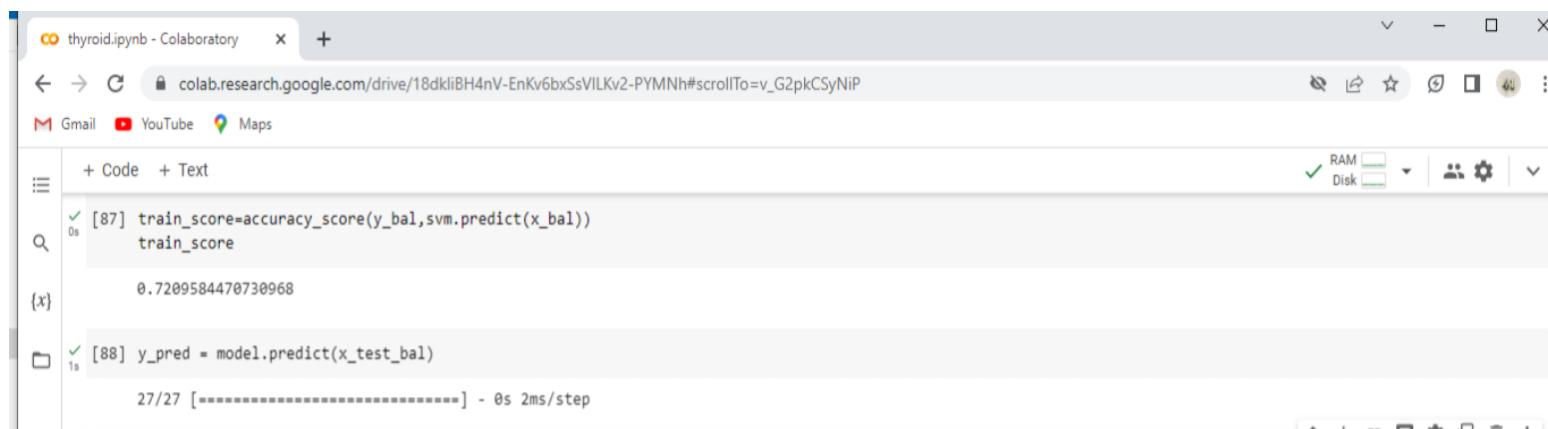
[84] train_score
1.0

{x}

[85] y_pred=xgb1.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

	precision	recall	f1-score	support
0.0	0.80	0.29	0.42	122
1.0	0.82	0.93	0.87	122
2.0	0.96	1.00	0.98	122
3.0	0.77	0.84	0.80	122
4.0	0.52	0.84	0.64	122
5.0	0.84	0.72	0.78	122
6.0	0.60	0.52	0.56	122
accuracy			0.74	854
macro avg	0.76	0.74	0.72	854
weighted avg	0.76	0.74	0.72	854



thyroid.ipynb - Colaboratory

File Home Share View

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=v_G2pkCSyNiP

Gmail YouTube Maps

+ Code + Text

[87] train_score=accuracy_score(y_bal,svm.predict(x_bal))
train_score

{x}

0.7209584470730968

[88] y_pred = model.predict(x_test_bal)

27/27 [=====] - 0s 2ms/step

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=t02gWWj2yazR

Gmail YouTube Maps

+ Code + Text

RAM Disk

0s

```
print(classification_report(y_test_bal,y_pred))

precision    recall   f1-score   support
0.0          0.00     0.00     0.00      122
1.0          0.14     1.00     0.25      122
2.0          0.00     0.00     0.00      122
3.0          0.00     0.00     0.00      122
4.0          0.00     0.00     0.00      122
5.0          0.00     0.00     0.00      122
6.0          0.00     0.00     0.00      122

accuracy                           0.14      854
macro avg       0.02     0.14     0.04      854
weighted avg    0.02     0.14     0.04      854

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
    _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
    _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
    _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
    _warn_prf(average, modifier, msg_start, len(result))

accuracy_score(y_test_bal,y_pred)

0.14285714285714285
```

Type here to search



36°C ENG 13:14
09-04-2023

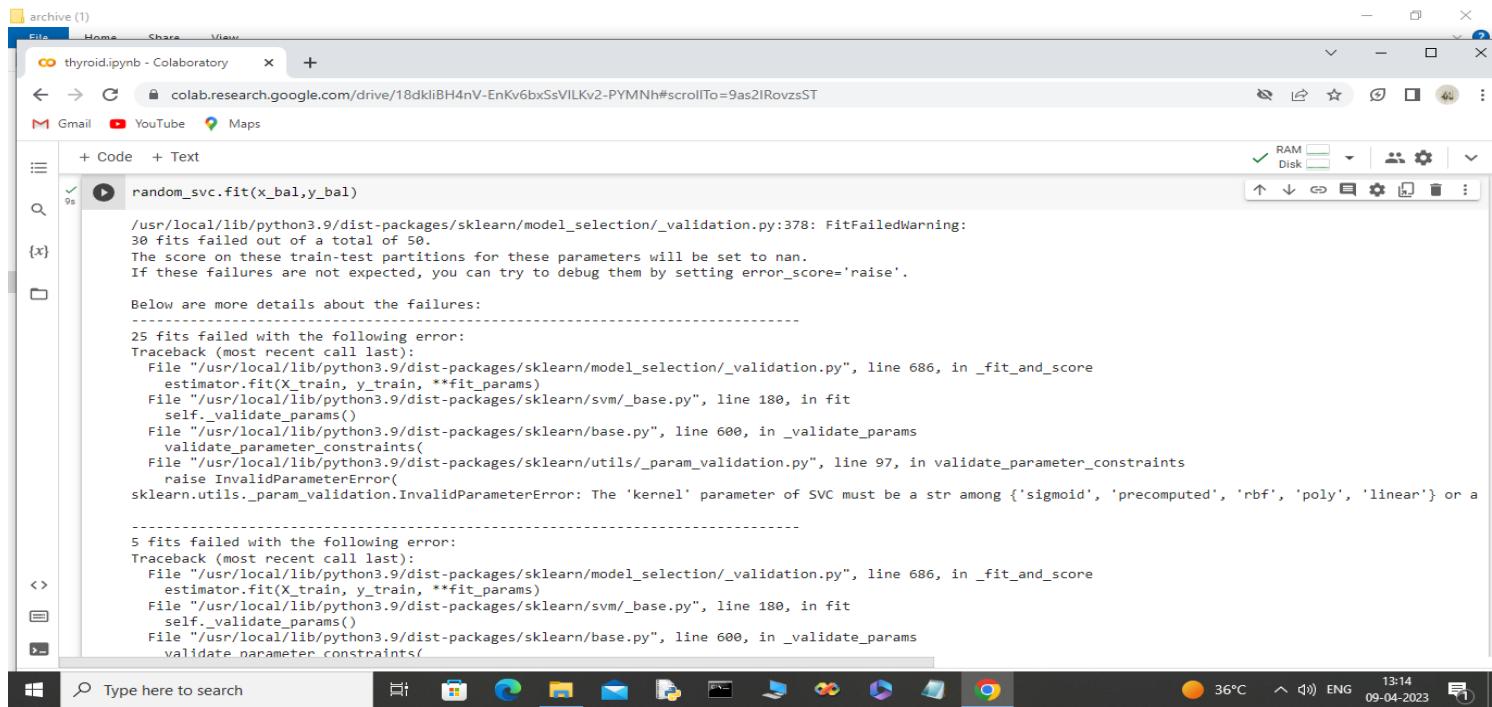


Comparing model accuracy before & after applying hyper parameter tuning



```
[x]
1s [91] params = {
    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf', 'sqrt']
}

1s ✓ from sklearn.model_selection import RandomizedSearchCV
random_svc = RandomizedSearchCV(svm, params, scoring='accuracy', cv=5, n_jobs=-1)
```



```
+ Code + Text
9s random_svc.fit(x_bal,y_bal)

/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py:378: FitFailedWarning:
30 fits failed out of a total of 50.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
-----
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py", line 180, in fit
    self._validate_params()
  File "/usr/local/lib/python3.9/dist-packages/sklearn/base.py", line 600, in _validate_params
    validate_parameter_constraints()
  File "/usr/local/lib/python3.9/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'kernel' parameter of SVC must be a str among {'sigmoid', 'precomputed', 'rbf', 'poly', 'linear'} or a

-----
5 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py", line 180, in fit
    self._validate_params()
  File "/usr/local/lib/python3.9/dist-packages/sklearn/base.py", line 600, in _validate_params
    validate_parameter_constraints()
```

archive (1)

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=8_6KjnjDz8pc

Gmail YouTube Maps

+ Code + Text

[93] Traceback (most recent call last):
File "/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py", line 686, in _fit_and_score
estimator.fit(X_train, y_train, **fit_params)
File "/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py", line 180, in fit
self._validate_params()
File "/usr/local/lib/python3.9/dist-packages/sklearn/base.py", line 600, in _validate_params
validate_parameter_constraints()
File "/usr/local/lib/python3.9/dist-packages/sklearn/utils/_param_validation.py", line 97, in validate_parameter_constraints
raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'kernel' parameter of SVC must be a str among {'poly', 'sigmoid', 'linear', 'precomputed', 'rbf'} or a
warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non-finite: [0.69730446 0.75917414
nan nan 0.7576515 nan]
warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
y = column_or_1d(y, warn=True)
► RandomizedSearchCV
► estimator: SVC
► SVC

random_svc.best_params_

```
{'kernel': 'rbf', 'gamma': 1, 'C': 10}
```

Type here to search

RAM Disk

36°C ENG 13:14 09-04-2023

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=Vbu3YCJw0a5y

Gmail YouTube Maps

+ Code + Text

[95] sv1=SVC(kernel= 'rbf', gamma= 0.1, C=100)

[96] sv1.fit(x_bal,y_bal)

/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
y = column_or_1d(y, warn=True)
► SVC
SVC(C=100, gamma=0.1)

y_pred= sv1.predict(x_test_bal)

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=Q2Be1VTr1CNL

Gmail YouTube Maps

+ Code + Text

[98] `print(classification_report(y_test_bal,y_pred))`

	precision	recall	f1-score	support
0.0	0.74	0.75	0.75	122
1.0	0.77	0.87	0.82	122
2.0	0.97	0.91	0.94	122
3.0	0.73	0.67	0.70	122
4.0	0.66	0.72	0.69	122
5.0	0.71	0.70	0.71	122
6.0	0.57	0.52	0.54	122
accuracy			0.74	854
macro avg	0.74	0.74	0.73	854
weighted avg	0.74	0.74	0.73	854

[99] `train_score= accuracy_score(y_bal,sv1.predict(x_bal))`

train_score

0.8180163785259327

thyroid.ipynb - Colaboratory

colab.research.google.com/drive/18dkliBH4nV-EnKv6bxSsVILKv2-PYMNh#scrollTo=WZm7blxV23fV

Gmail YouTube Maps

+ Code + Text

[100] `# saving the model`

[100] `import pickle`

[100] `pickle.dump(sv1,open('thyroid_1_model.pkl','wb'))`

[101] `features = np.array([[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]])`

[101] `print(label_encoder.inverse_transform(xgb1.predict(features)))`

[101] `['hypothyroid conditions']`

[102] `pickle.dump(label_encoder, open('label_encoder.pkl', 'wb'))`

[103] `data['target'].unique()`

[103] `array(['miscellaneous', 'hypothyroid conditions', 'binding protein', 'replacement therapy', 'general health', 'hyperthyroid conditions', 'antithyroid treatment'], dtype=object)`

[104] `y['target'].unique()`

[104] `array([5., 4., 1., 6., 2., 3., 0.])`

[105] `import pickle`

[105] `pickle.dump(sv1,open('thyroid_1_model.pkl','wb'))`

Type here to search

36°C ENG 09-04-2023 13:17

ADVANTAGES

- ❖ **Accuracy:** Machine learning algorithms can accurately classify thyroid diseases based on patterns and relationships found in large datasets, which can improve accuracy compared to traditional diagnostic methods.
- ❖ **Speed:** Machine learning algorithms can process large amounts of data much faster than humans, reducing the time required for diagnosis and treatment.
- ❖ **Personalization:** Machine learning algorithms can be trained to personalize diagnosis and treatment based on individual patient characteristics, such as age, gender, and medical history.
- ❖ **Consistency:** Machine learning algorithms can provide consistent results across different medical professionals and institutions, reducing the risk of misdiagnosis and improving patient outcomes.

APPLICATION

- ❖ **Automated diagnosis:** Machine learning models can be trained on medical images (such as ultrasound or MRI) to automatically diagnose thyroid disease. This could potentially reduce the need for invasive diagnostic procedures, such as biopsies.
- ❖ **Predictive modeling:** Machine learning models can be used to predict the likelihood of thyroid disease in patients based on their medical history, lifestyle factors, and genetic data. This could help clinicians identify high-risk patients and provide targeted preventive care.
- ❖ **Treatment optimization:** Machine learning models can be used to predict the most effective treatment for individual patients based on their medical history and other factors. This could help optimize treatment outcomes and reduce the risk of adverse effects.

CONCLUSION

In conclusion, machine learning algorithms can be effective in classifying thyroid disease based on various features and characteristics. The accuracy of the classification models depends on the quality and quantity of data used for training, the selection of appropriate features, and the choice of suitable algorithms. Different machine learning algorithms such as Random Forest Classifier, SVM classifier, XGB classifier and neural networks have been used in the literature to classify thyroid diseases. Ensemble methods machine learning algorithms have the potential to assist clinicians in the diagnosis and treatment of thyroid diseases by providing accurate and timely predictions.

FUTURE SCOPE

- ❖ **Feature engineering:** One of the critical aspects of machine learning is feature selection or engineering. Researchers can explore the creation of new features or the extraction of existing features from thyroid disease data. This may include demographic data, medical history, lab results, and imaging data, among others.
- ❖ **Deep learning models:** Deep learning models, such as convolution neural networks (CNNs) and recurrent neural networks (RNNs), can learn high-level representations of the data, leading to more accurate classification. Researchers can explore the use of these models for thyroid disease classification
- ❖ **Explainable AI:** Explainable AI (XAI) aims to make machine learning models more interpretable to humans. Researchers can explore the use of XAI techniques such as LIME, SHAP