Smart Internz

PROJECT DOCUMENTATIONT

Team Members:

Team Leader Name: M.GOWSALYA

Team Member One: S.UMA MAHESHWARI

Team Member Two :S.MARY SHAHAYA SHINY

Team Member Three:K.KRISHNASUMITHA.

THYROID DISEASE CLASSIFICATION USING ML

1. Introduction:

My project is Thyroid Disease Classification using ML . ML is the subset of artificial intelligence. First we define problem and find ideas to solve the problem using empathy map and brainstorm.The problem of my project is classify the thyroid disease based on given input using ML. Next step of my project is import libraries files .

Second collect the data set and pre-process the data set. Remove redundant,null values .

Split the dateset x ,x_train,y , y_train and handling categorical values and perform features importance .

Next , to perform Exploratory data analysis and Training the model using multiple algorithms like RandomForestClassifier, Xgb Classifier, SVC model, ANN model.After train the model next test the model.

Save the best model insert.And Create a three web pages namely home,predict,submit. Build python code and insert the saved model Run the application.And finally, The resulting page is show the result based on given inputs.

INTRO OF THYROID DISEASE

Thyroid disease is Affected by both male and female.Two type thyroids are Hypothyroid and Hyperthyroid.Hyper function **hyperthyroidism** and **hypothyroidism** affect about 2% and 1% of individuals, respectively.Early identification and differential diagnosis raises the odds of good treatment]. The thyroid gland is a butterfly-shaped gland situated at the base of the throat. It comprises two active thyroid hormones, hyperthyroid (T4) and hypothyroidism (T3), which are involved in brain functions such as body temperature control, blood pressure management, and heart rate regulation The thyroid gland is an endocrine gland that secretes hormones and passes them through the bloodstream. It is situated in the middle of the front of the body. Thyroid gland hormones are responsible for aiding in digestion as well as maintaining the body moist, balanced, and so on. Thyroid gland treatments such as T3 (hypothyroidism), T4 (thyroid hormone), and **TSH** (thyroid stimulating hormone) are used to assess thyroid activity (thyroid stimulating hormone). Thyroid disorder is classified into two types: hypothyroidism and hyperthyroidism. Data mining is a semi-automated method of looking for correlations in massive datasets. Machine learning algorithms are one of the best solutions to many problems that are difficult to solve Classification is a data extraction technique (machine learning) used to predict and identify many diseases, such as thyroid disease, which we researched and classified here because machine learning algorithms play a significant role in classifying thyroid disease and because these algorithms are high performing and efficient and aid in classify Hyperthyroidism is a disorder in which the thyroid gland releases so many thyroid hormones. Hyperthyroidism is caused by an increase in thyroid hormone levels .

SYMPTOMS:

Dry skin, elevated temperature sensitivity, hair thinning, weight loss, increased heart rate, high blood pressure, heavy sweating, neck enlargement, nervousness, menstrual cycles shortening, irregular stomach movements, and hands shaking are some of the signs [11]. Hypothyroidism is a condition in which the thyroid gland is under active Hypothyroidism is caused by a decline in thyroid hormone production. Hypo means deficient or less in medical terms. Inflammation and thyroid gland injury are the two primary causes of hypothyroidism. Obesity, low heart rate, increased temperature sensitivity, neck swelling, dry skin, hand numbness, hair issues, heavy menstrual cycles, and intestinal problems are some of the symptoms.

OVERVIEW:

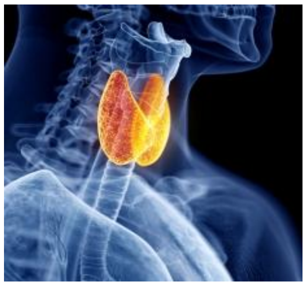
- ✓ My project is helpful for predict thyroid disease I am used google co lab to type my project coding and run all codes correctly next ,save the model and create web page and designing using my own HTML coding.
- ✓ Three web pages are created they are: Home.HTML,predict.HTML,submit.HTML.
- ✓ And the python coding to insert the saved model and HTML forms type the python coding and run the application and finally the submit page is show my result is miscellaneous.

Small description of thyroid disease

- ✓ The thyroid gland is a vascular gland and one of the most organs of the human body.The thyroid gland is the butterfly shaped organs composed of bulbous right and left lobes connected in the midlife by a thin structure called the isthmus.The two types of thyroid disorders are

  1. **Hyperthyroidism**
  2. **Hypothyroidism**

- ✓ When this disorder occur in the body , they release certain types of hormones into the body which imbalances in the body's metabolism.A thyroid-related Blood test is used to detect this disease but it is often blurred and noise will be preens.Data cleansing methods where used to make the data primitive enough for the analytic to show the risk of patient getting this disease .Machine Learning place a very deciding role in disease prediction.Machine Learning algorithms,SIM-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.

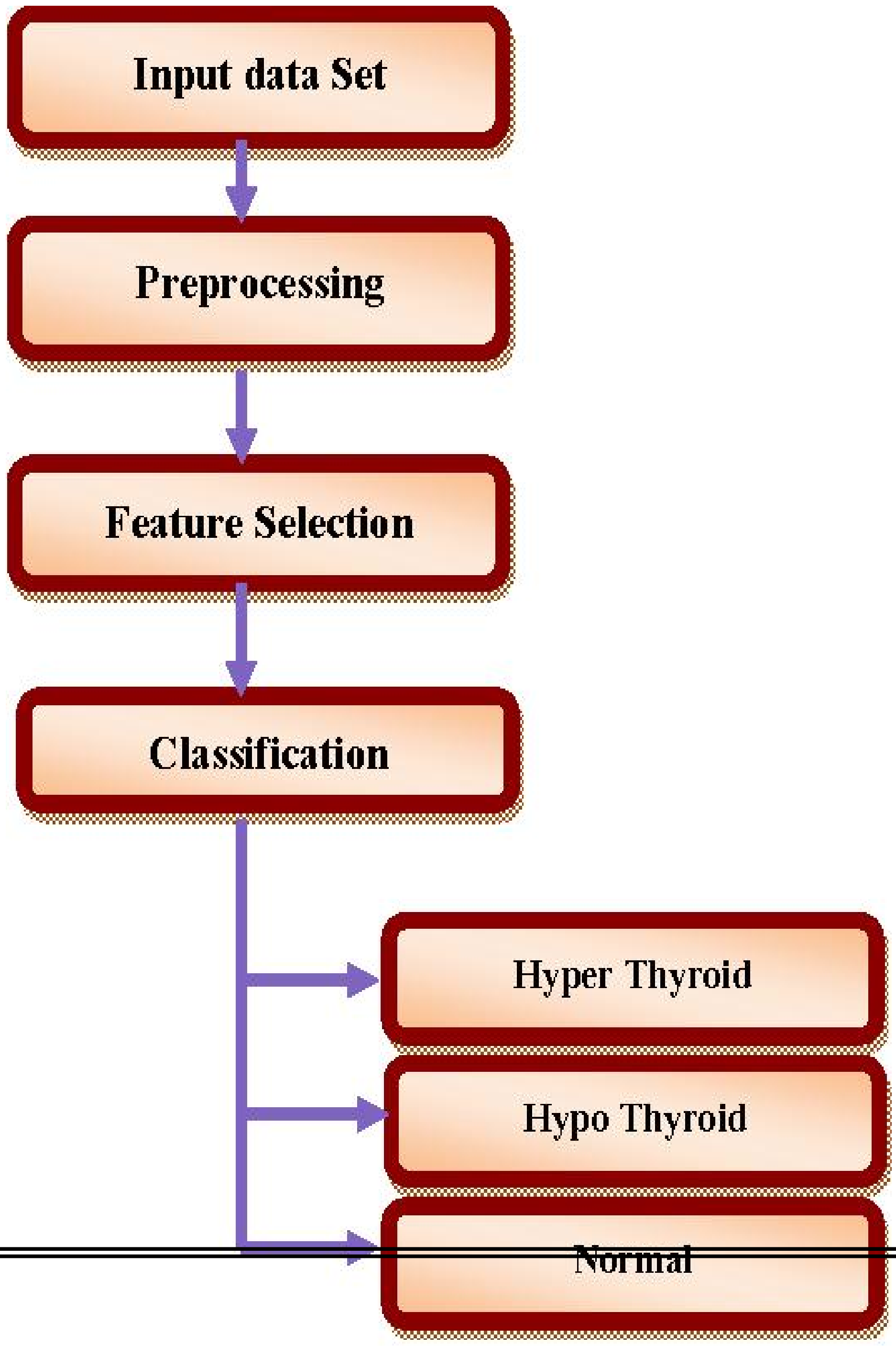


THYROID DISEASE

PURPOSE:

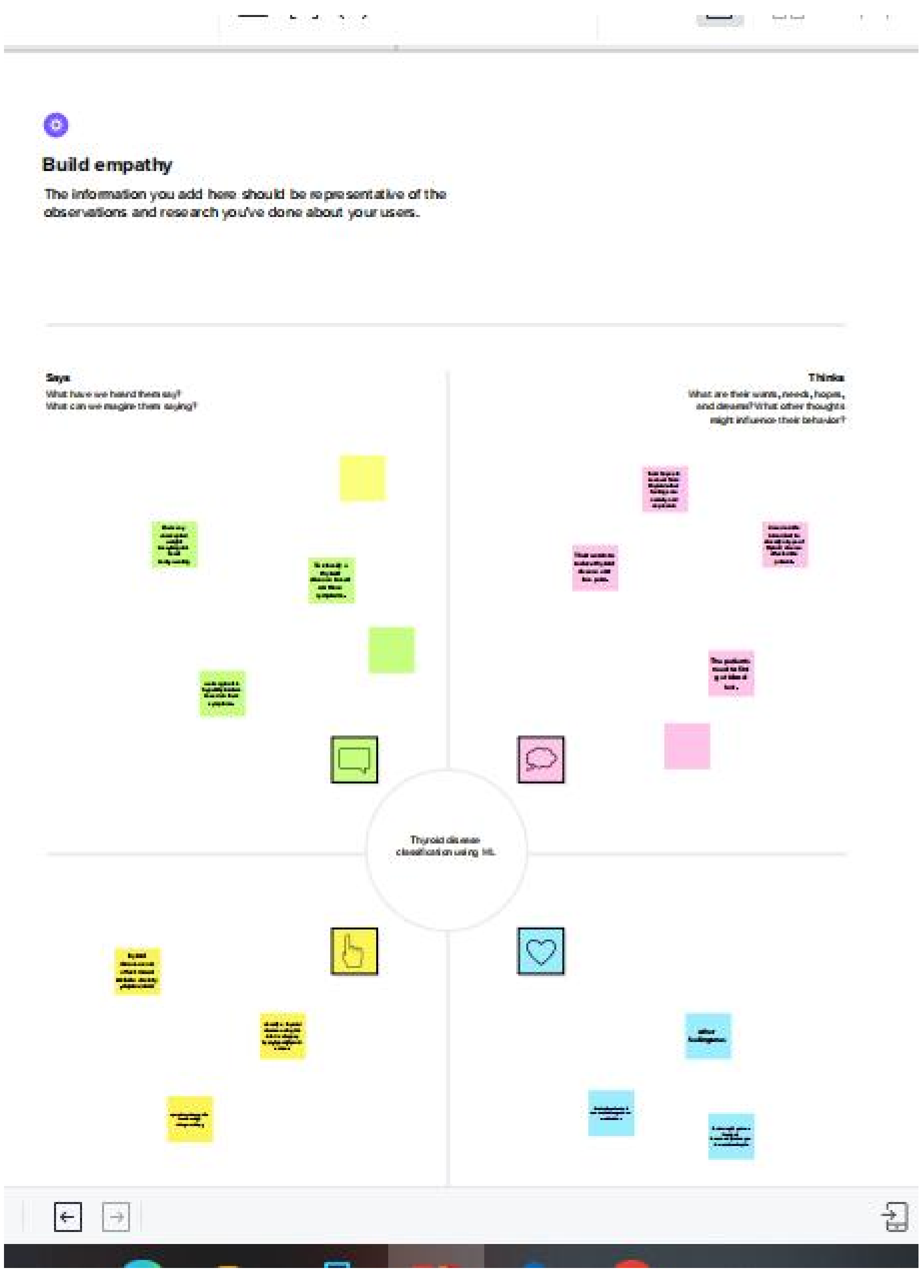
- The main purpose is to search the best classification approach for thyroid disease diagnosis by making the comparison of decision tree algorithms.In the line of this purpose, the experiments are conducted to compare different kinds of decision tree algorithms given in the previous section.It makes hormones that control the way the body uses energy. These hormones affect nearly every organ in your body and control many of your body's most important functions. For example, they affect your breathing, heart rate, weight.
- The purpose of predicting thyroid classification is to evaluate what type of thyroid disease affect the patient to find easily. The machine learning algorithms have been employed to model the prediction and diagnosis of thyroid patients.A variety of these algorithms including ,Random forest ,Decision

trees,Support vector machine,Artificial neural network and logistic regression have been widely used in development of predictive models of thyroid disease.The paper presents a review recent ML algorithms applied in the prediction and diagnosis of thyroid detection.The proposed system is used for thyroid disease prediction of patients,based on various symptoms and reports of thyroid.Among these decision algorithm is found to be better with the accuracy of 99.46%.



Problem Definition and design Thinking:

1. EMPATHY MAP:



Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says

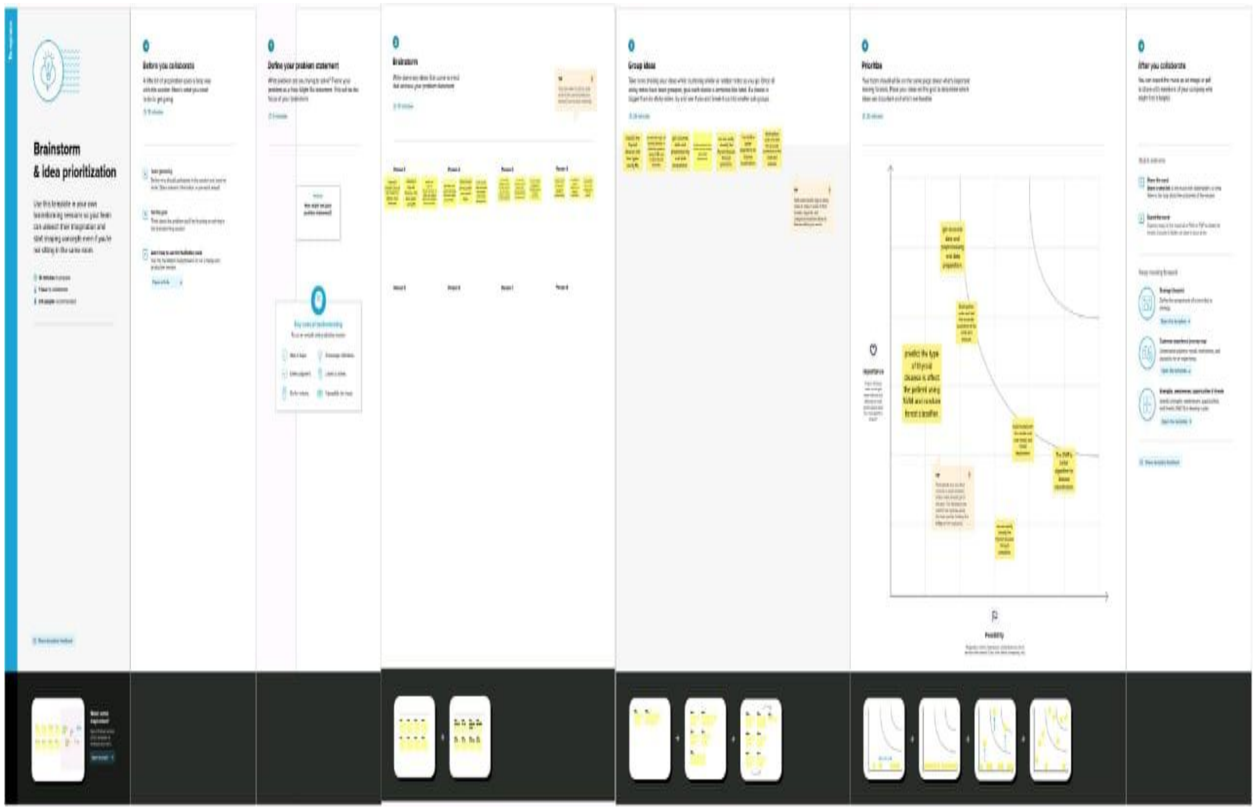
What have we heard them say?
What can we imagine them saying?

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

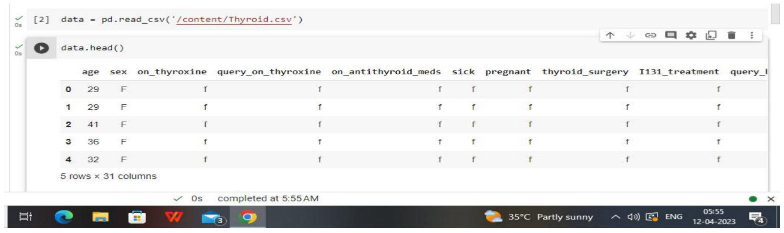
Thyroid disease classification using ML

2.BRAINSTORMING MAP:



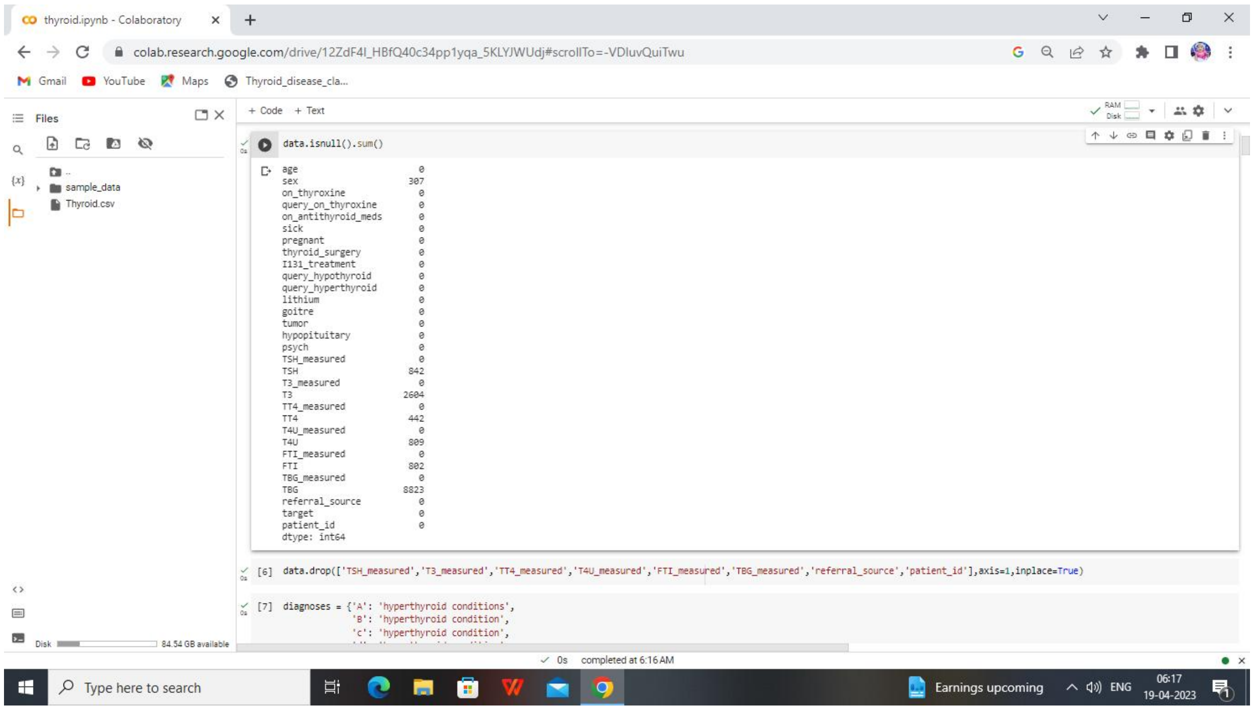
3.RESULT:

Read the data set



```
[2] data = pd.read_csv('/content/Thyroid.csv')

data.head()

   age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  I131_treatment  query_
0   29    F             f                   f                    f     f         f                f               f
1   29    F             f                   f                    f     f         f                f               f
2   41    F             f                   f                    f     f         f                f               f
3   36    F             f                   f                    f     f         f                f               f
4   32    F             f                   f                    f     f         f                f               f
5 rows × 31 columns

0s   completed at 5:55 AM
```

Checking null values



thyroid.ipynb - Colaboratory

colab.research.google.com/drive/12ZdF4l_HBfQ40c34pp1yqa_5KLYJWUdj#scrollTo=-VDluvQuiTwu

```
data.isnull().sum()
```

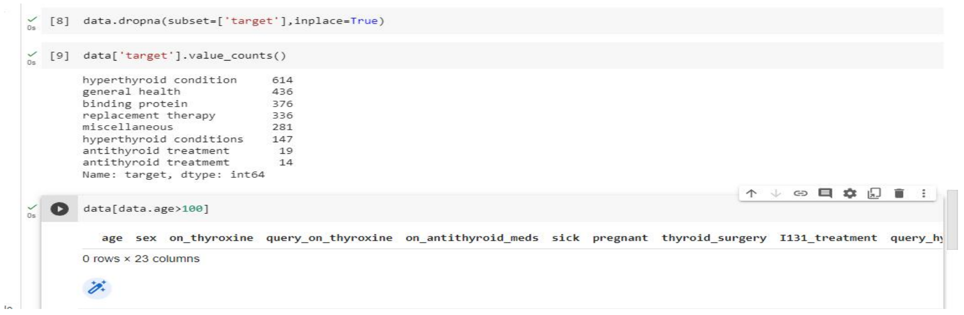
```
age                       0
sex                     307
on_thyroxine              0
query_on_thyroxine        0
on_antithyroid_meds       0
sick                      0
pregnant                  0
thyroid_surgery           0
I131_treatment            0
query_hypothyroid         0
query_hyperthyroid        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
```

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

```
[7] diagnoses = {'A': 'hyperthyroid conditions',
                 'B': 'hyperthyroid condition',
                 'c': 'hyperthyroid condition',
```

completed at 6:16 AM

Drop null values



```
[8] data.dropna(subset=['target'],inplace=True)
```

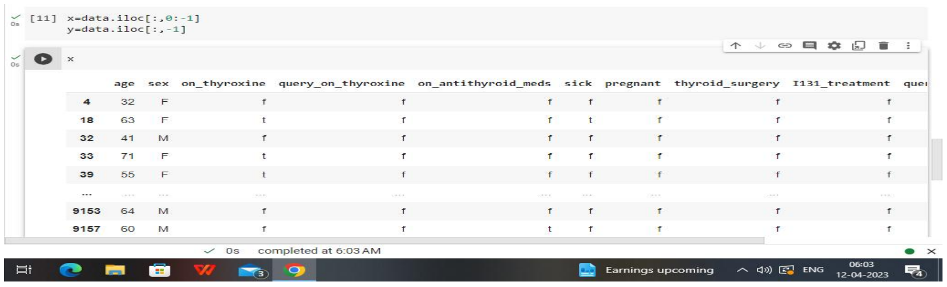
```
[9] data['target'].value_counts()
```

```
hyperthyroid condition     614
general health             436
binding protein            376
replacement therapy        336
miscellaneous              281
hyperthyroid conditions    147
antithyroid treatment       19
antithyroid treatmemt       14
Name: target, dtype: int64
```

```
data[data.age>100]
```

```
   age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  I131_treatment  query_hy
0 rows × 23 columns
```

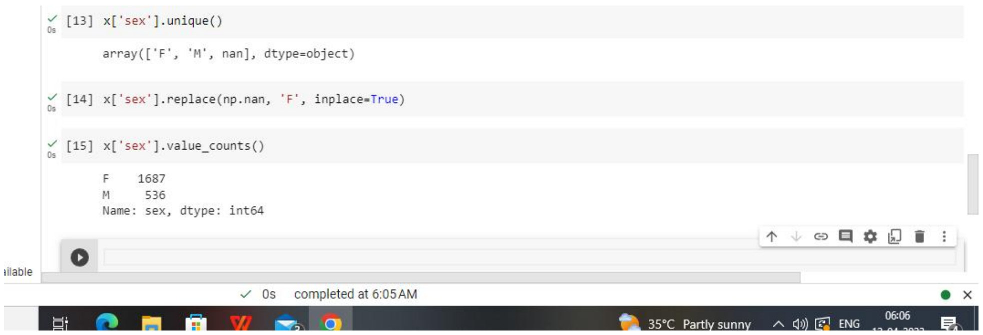
Splitting the data x and y



```
[11] x=data.iloc[:,0:-1]
     y=data.iloc[:,-1]
```

```
x
```

```
      age  sex  on_thyroxine  query_on_thyroxine  on_antithyroid_meds  sick  pregnant  thyroid_surgery  I131_treatment  que
   4   32   F        f               f                    f              f       f            f                f
  18   63   F        t               f                    f              t       f            f                f
  32   41   M        f               f                    f              f       f            f                f
  33   71   F        t               f                    f              f       f            f                f
  39   55   F        t               f                    f              f       f            f                f
 ...  ...  ...      ...             ...                  ...            ...     ...          ...              ...
9153   64   M        f               f                    f              f       f            f                f
9157   60   M        f               f                    t              f       f            f                f
```



```
[13] x['sex'].unique()
```

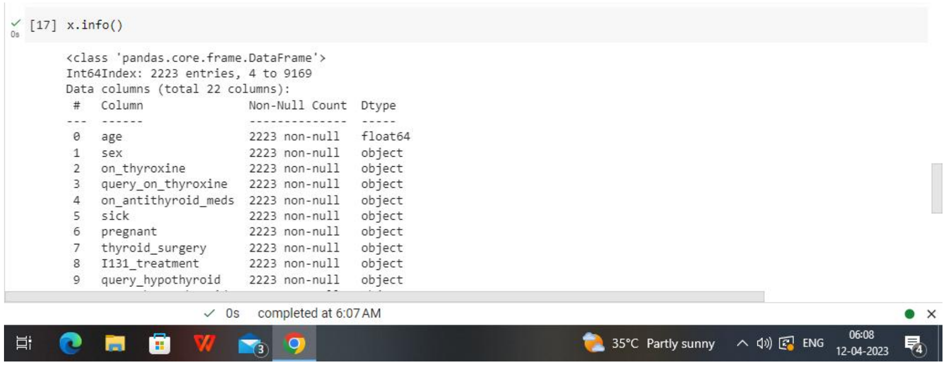
```
array(['F', 'M', nan], dtype=object)
```

```
[14] x['sex'].replace(np.nan, 'F', inplace=True)
```

```
[15] x['sex'].value_counts()
```

```
F    1687
M     536
Name: sex, dtype: int64
```

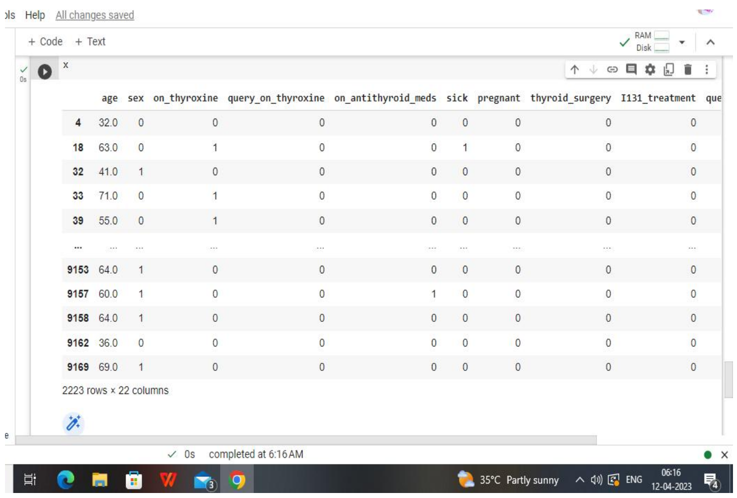
Convert the data type



```
[17] x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2223 entries, 4 to 9169
Data columns (total 22 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  2223 non-null   float64
 1   sex                  2223 non-null   object 
 2   on_thyroxine         2223 non-null   object 
 3   query_on_thyroxine   2223 non-null   object 
 4   on_antithyroid_meds  2223 non-null   object 
 5   sick                 2223 non-null   object 
 6   pregnant             2223 non-null   object 
 7   thyroid_surgery      2223 non-null   object 
 8   I131_treatment       2223 non-null   object 
 9   query_hypothyroid    2223 non-null   object 
```

Handling categorical values

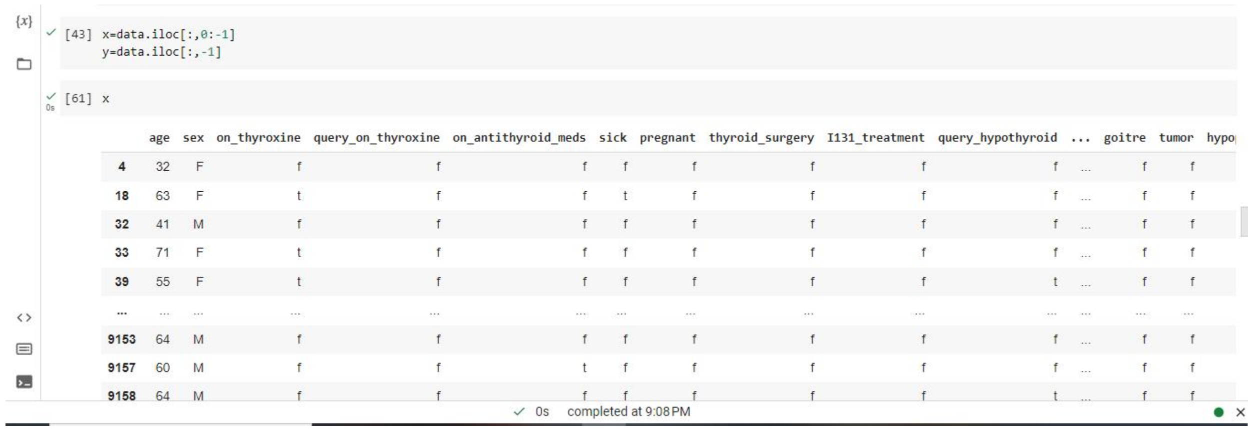


x

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

2223 rows × 22 columns

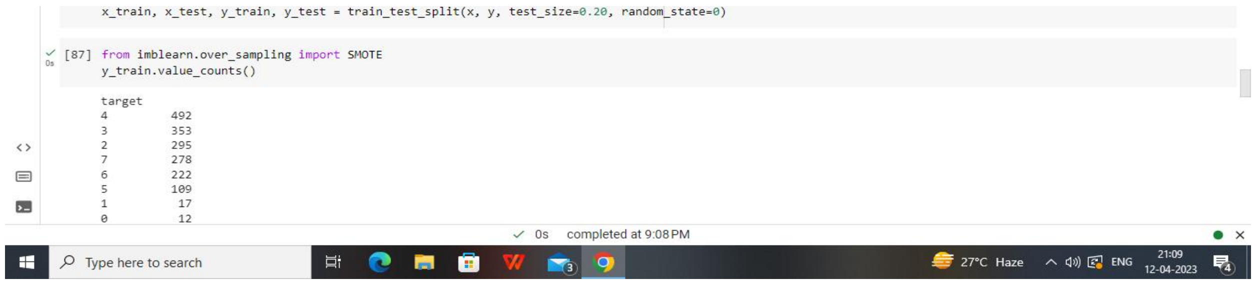
Splitting data into train and test



```
[43] x=data.iloc[:,0:-1]
     y=data.iloc[:,-1]
```

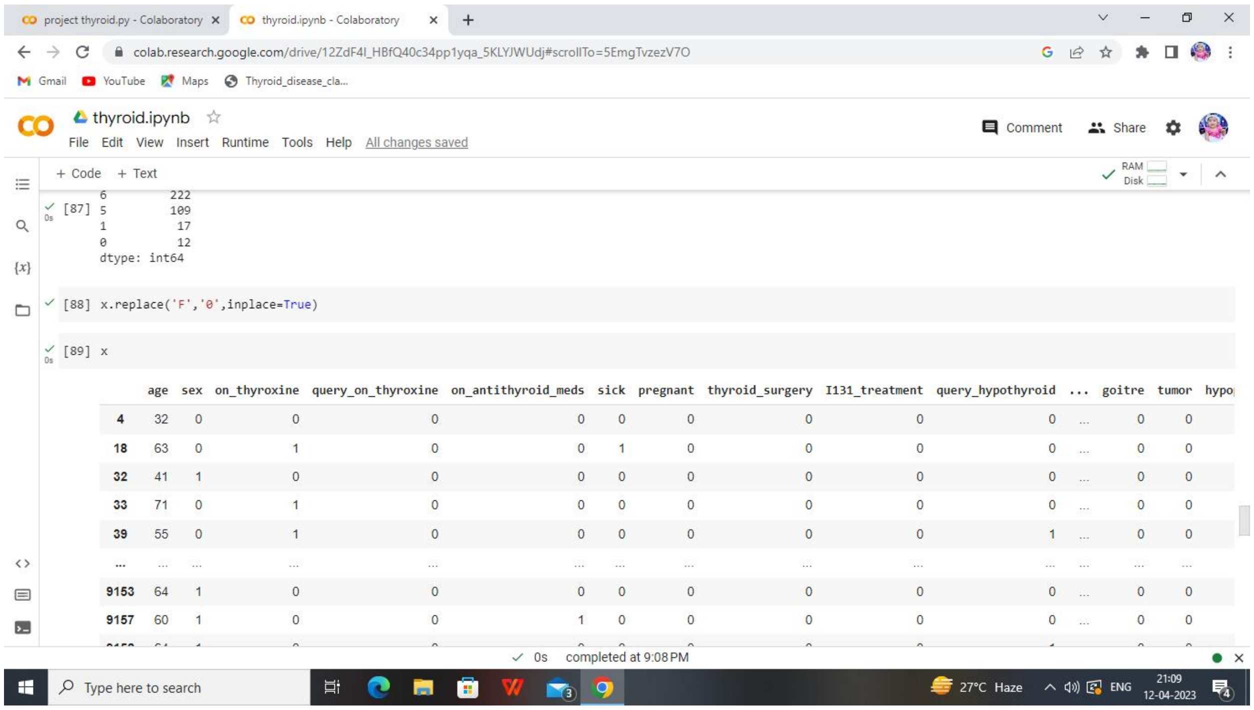
```
[61] x
```

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



```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=0)
```

```
[87] from imblearn.over_sampling import SMOTE
     y_train.value_counts()

     target
     4      492
     3      353
     2      295
     7      278
     6      222
     5      109
     1       17
     0       12
```

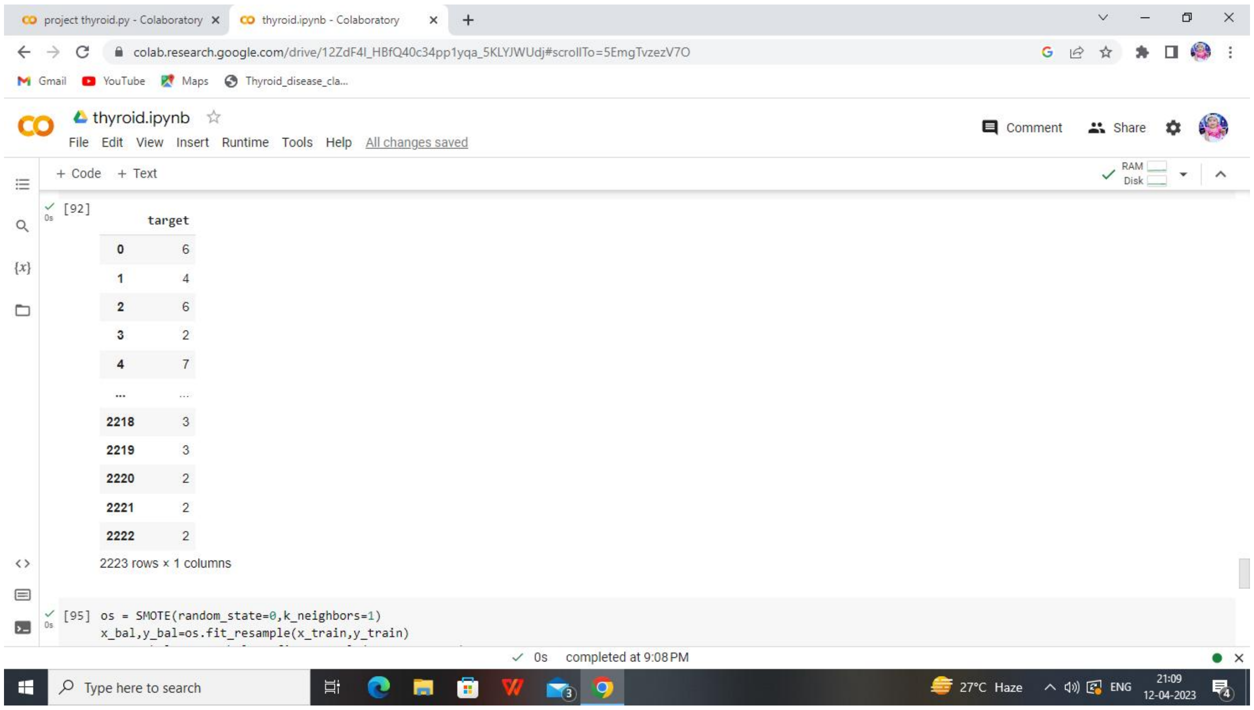


```
[87] 6        222
     5        109
     1         17
     0         12
     dtype: int64
```

```
[88] x.replace('F','0',inplace=True)
```

```
[89] x
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	goitre	tumor	hypo
4	32	0	0	0	0	0	0	0	0	0	...	0	0	
18	63	0	1	0	0	1	0	0	0	0	...	0	0	
32	41	1	0	0	0	0	0	0	0	0	...	0	0	
33	71	0	1	0	0	0	0	0	0	0	...	0	0	
39	55	0	1	0	0	0	0	0	0	1	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9153	64	1	0	0	0	0	0	0	0	0	...	0	0	
9157	60	1	0	0	1	0	0	0	0	0	...	0	0	



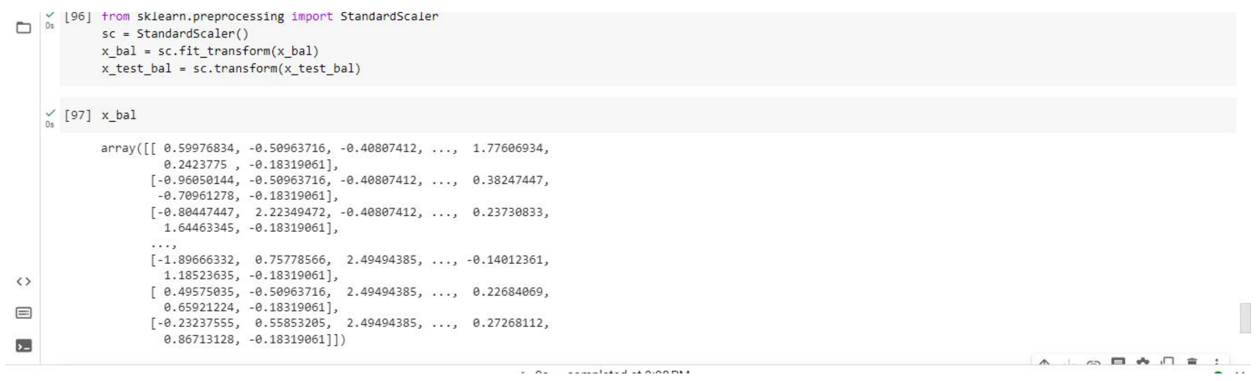
```
[92]
```

	target
0	6
1	4
2	6
3	2
4	7
...	...
2218	3
2219	3
2220	2
2221	2
2222	2

2223 rows × 1 columns

```
[95] os = SMOTE(random_state=0,k_neighbors=1)
     x_bal,y_bal=os.fit_resample(x_train,y_train)
```

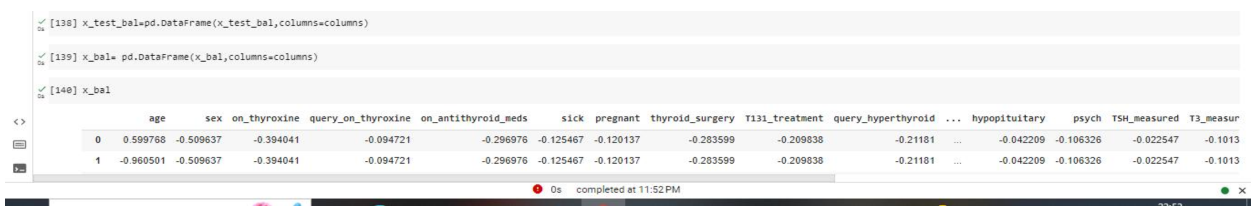
Apply StandardScaler



```
[96] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     x_bal = sc.fit_transform(x_bal)
     x_test_bal = sc.transform(x_test_bal)
```

```
[97] x_bal
```

```
array([[ 0.59976834, -0.50963716, -0.40807412, ...,  1.77606934,
         0.2423775 , -0.18319061],
       [-0.96050144, -0.50963716, -0.40807412, ...,  0.38247447,
        -0.70961278, -0.18319061],
       [-0.80447447,  2.22349472, -0.40807412, ...,  0.23730833,
         1.64463345, -0.18319061],
       ...,
       [-1.89666332,  0.75778566,  2.49494385, ..., -0.14012361,
         1.18523635, -0.18319061],
       [ 0.49575035, -0.50963716,  2.49494385, ...,  0.22684069,
         0.65921224, -0.18319061],
       [-0.23237555,  0.55853205,  2.49494385, ...,  0.27268112,
         0.86713128, -0.18319061]])
```



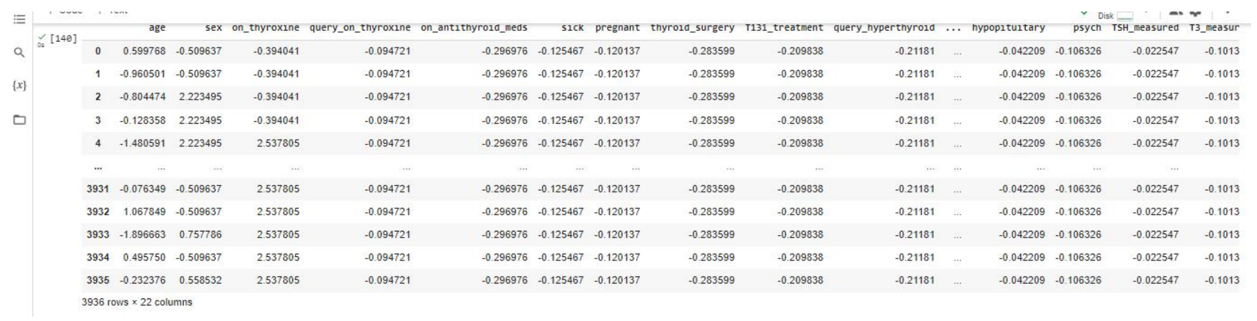
```
[138] x_test_bal=pd.DataFrame(x_test_bal,columns=columns)
```

```
[139] x_bal= pd.DataFrame(x_bal,columns=columns)
```

```
[140] x_bal
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hyperthyroid	...	hypopituitary	psych	TSH_measured	T3_measur
0	0.599768	-0.509637	-0.394041	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
1	-0.960501	-0.509637	-0.394041	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013

0s completed at 11:52 PM

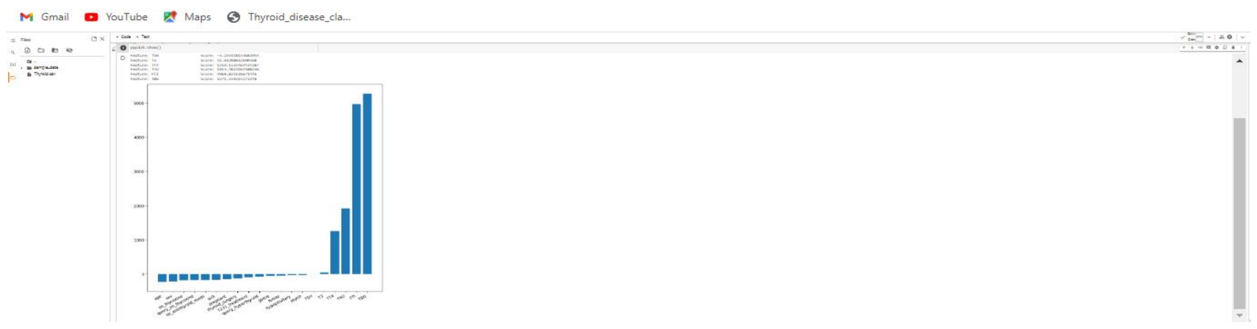


```
[140]
```

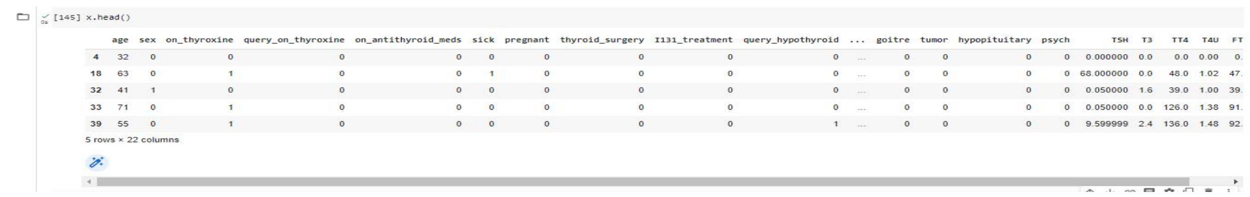
	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hyperthyroid	...	hypopituitary	psych	TSH_measured	T3_measur
0	0.599768	-0.509637	-0.394041	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
1	-0.960501	-0.509637	-0.394041	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
2	-0.804474	2.223495	-0.394041	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
3	-0.128358	2.223495	-0.394041	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
4	-1.480591	2.223495	2.537805	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3931	-0.076349	-0.509637	2.537805	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
3932	1.067849	-0.509637	2.537805	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
3933	-1.896663	0.757786	2.537805	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
3934	0.495750	-0.509637	2.537805	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013
3935	-0.232376	0.558532	2.537805	-0.094721	-0.296976	-0.125467	-0.120137	-0.283599	-0.209838	-0.21181	...	-0.042209	-0.106326	-0.022547	-0.1013

3936 rows × 22 columns

Performing Feature Importance



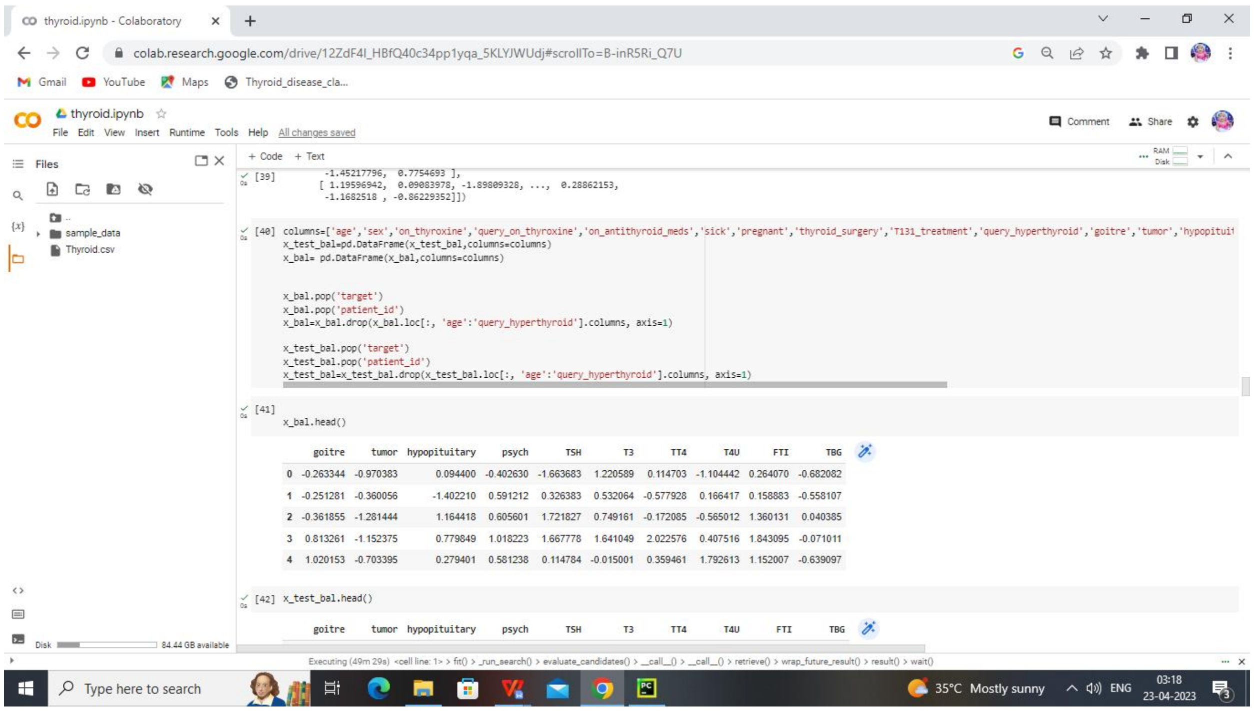
```
pyplot.show()
Feature: TSH          Score: [illegible]
Feature: T3           Score: [illegible]
Feature: TT4          Score: [illegible]
Feature: T4U          Score: [illegible]
Feature: FTI          Score: [illegible]
Feature: TBG          Score: [illegible]
```



```
[145] 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	T4U	FT
4	32	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0.000000	0.0	0.0	0.00	0.
18	63	0	1	0	0	1	0	0	0	0	...	0	0	0	0	68.000000	0.0	48.0	1.02	47.
32	41	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0.050000	1.6	39.0	1.00	39.
33	71	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0.050000	0.0	126.0	1.38	91.
39	55	0	1	0	0	0	0	0	0	1	...	0	0	0	0	9.599999	2.4	136.0	1.48	92.

5 rows × 22 columns



```
[39]        -1.45217796,  0.7754693 ],
          [ 1.19596942,  0.09083978, -1.89809328, ...,  0.28862153,
           -1.1682518 , -0.86229352]])
```

```
[40] columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','thyroid_surgery','T131_treatment','query_hyperthyroid','goitre','tumor','hypopitui
     x_test_bal=pd.DataFrame(x_test_bal,columns=columns)
     x_bal= pd.DataFrame(x_bal,columns=columns)


     x_bal.pop('target')
     x_bal.pop('patient_id')
     x_bal=x_bal.drop(x_bal.loc[:, 'age':'query_hyperthyroid'].columns, axis=1)

     x_test_bal.pop('target')
     x_test_bal.pop('patient_id')
     x_test_bal=x_test_bal.drop(x_test_bal.loc[:, 'age':'query_hyperthyroid'].columns, axis=1)
```

```
[41]
     x_bal.head()
```

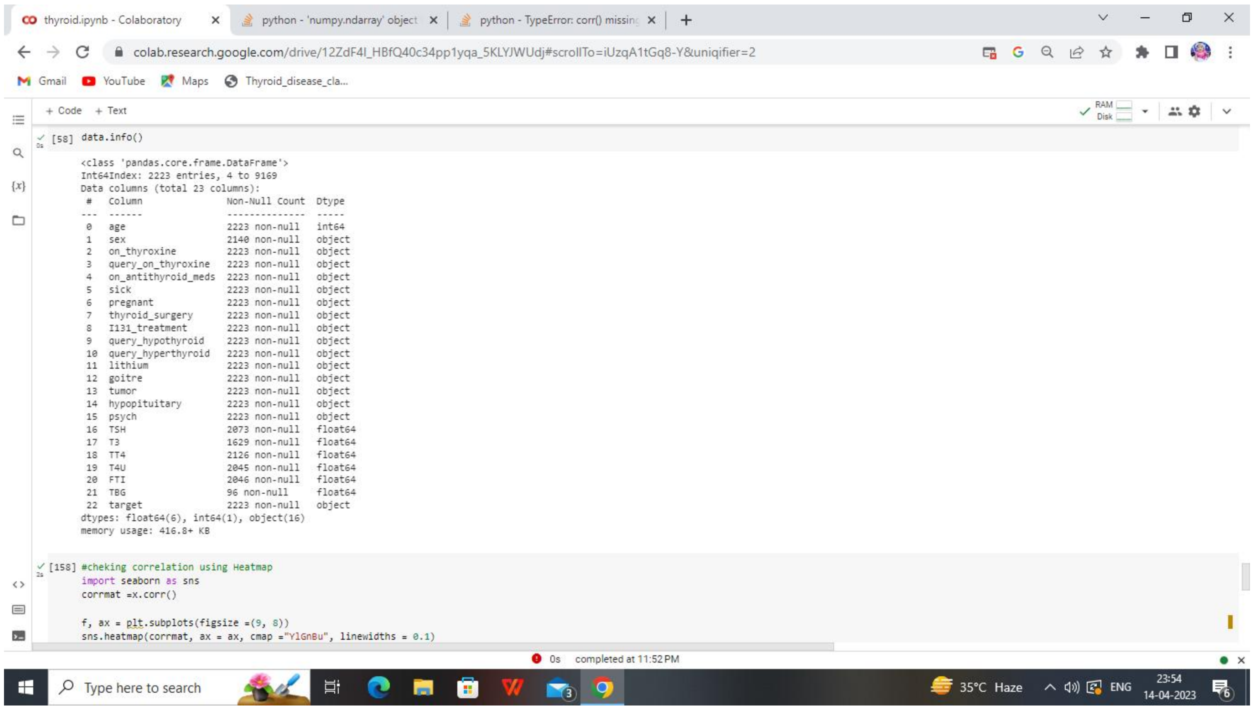
	goitre	tumor	hypopituitary	psych	TSH	T3	TT4	T4U	FTI	TBG
0	-0.263344	-0.970383	0.094400	-0.402630	-1.663683	1.220589	0.114703	-1.104442	0.264070	-0.682082
1	-0.251281	-0.360056	-1.402210	0.591212	0.326383	0.532064	-0.577928	0.166417	0.158883	-0.558107
2	-0.361855	-1.281444	1.164418	0.605601	1.721827	0.749161	-0.172085	-0.565012	1.360131	0.040385
3	0.813261	-1.152375	0.779849	1.018223	1.667778	1.641049	2.022576	0.407516	1.843095	-0.071011
4	1.020153	-0.703395	0.279401	0.581238	0.114784	-0.015001	0.359461	1.792613	1.152007	-0.639097

```
[42] x_test_bal.head()
```

	goitre	tumor	hypopituitary	psych	TSH	T3	TT4	T4U	FTI	TBG

```
Executing (49m 29s) <cell line: 1> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > retrieve() > wrap_future_result() > result() > wait()
```

Exploratory Data Analysis

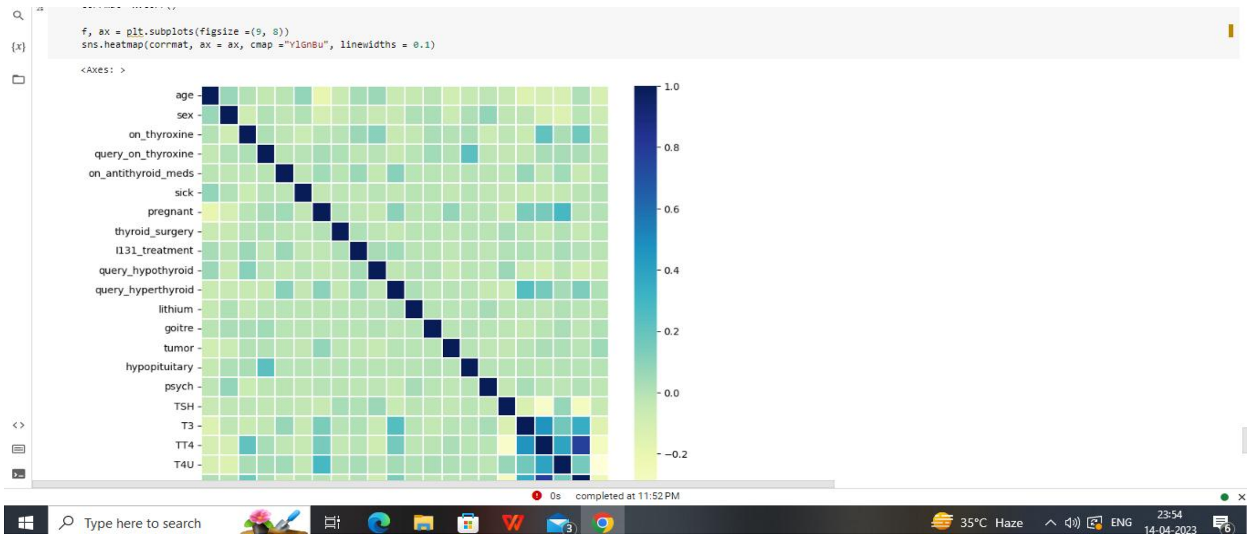


```
[58] data.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 2223 entries, 4 to 9169
     Data columns (total 23 columns):
      #   Column               Non-Null Count  Dtype
     ---  ------               --------------  -----
      0   age                  2223 non-null   int64
      1   sex                  2140 non-null   object
      2   on_thyroxine         2223 non-null   object
      3   query_on_thyroxine   2223 non-null   object
      4   on_antithyroid_meds  2223 non-null   object
      5   sick                 2223 non-null   object
      6   pregnant             2223 non-null   object
      7   thyroid_surgery      2223 non-null   object
      8   I131_treatment       2223 non-null   object
      9   query_hypothyroid    2223 non-null   object
      10  query_hyperthyroid   2223 non-null   object
      11  lithium              2223 non-null   object
      12  goitre               2223 non-null   object
      13  tumor                2223 non-null   object
      14  hypopituitary        2223 non-null   object
      15  psych                2223 non-null   object
      16  TSH                  2073 non-null   float64
      17  T3                   1629 non-null   float64
      18  TT4                  2126 non-null   float64
      19  T4U                  2045 non-null   float64
      20  FTI                  2046 non-null   float64
      21  TBG                  96 non-null     float64
      22  target               2223 non-null   object
     dtypes: float64(6), int64(1), object(16)
     memory usage: 416.8+ KB
```

```
[158] #cheking correlation using Heatmap
      import seaborn as sns
      corrmat =x.corr()

      f, ax = plt.subplots(figsize =(9, 8))
      sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)
```

Visual analysis:

Checking correlation

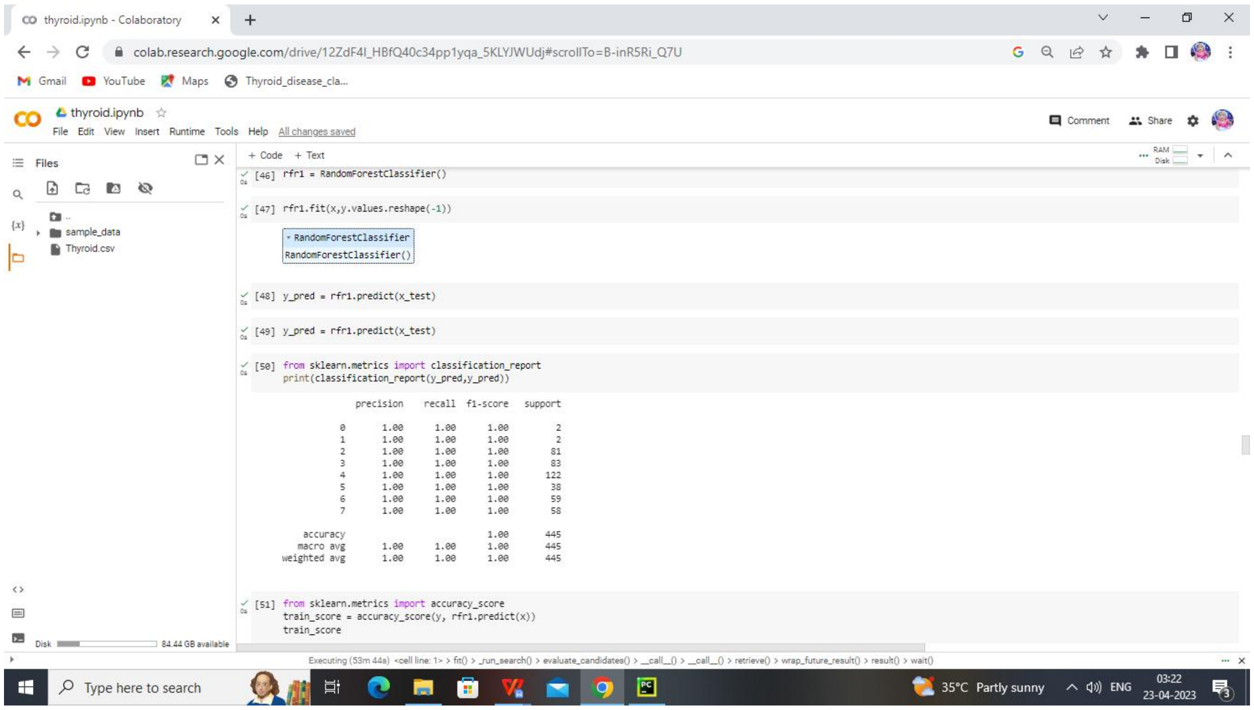


```
f, ax = plt.subplots(figsize =(9, 8))
sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)
```

```
<Axes: >
```

Model building

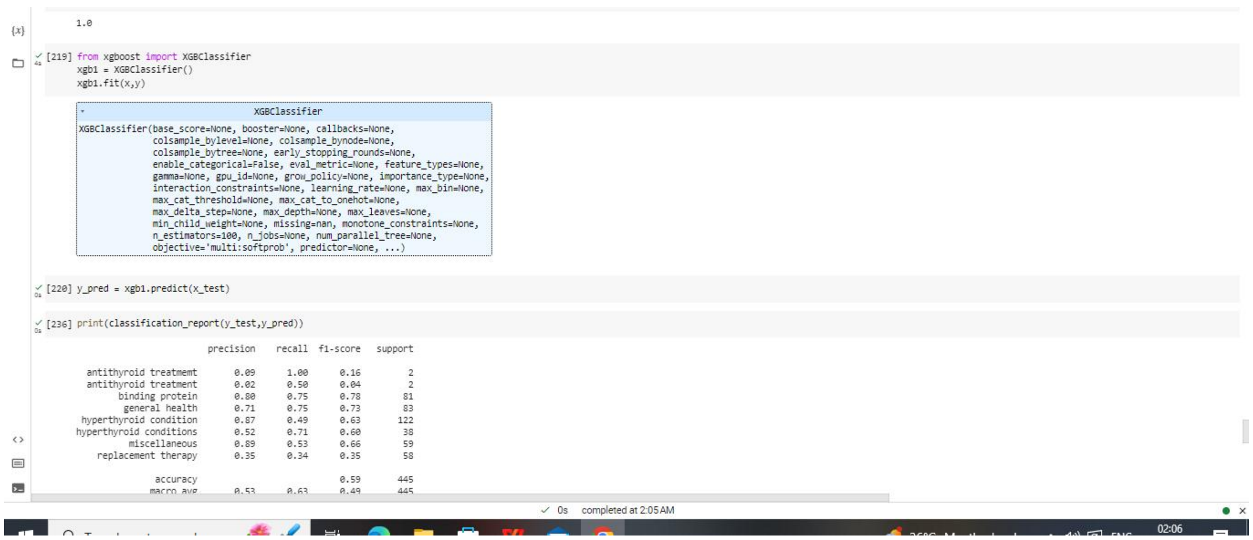
Random Forest Classifier Model



```
[46] rfr1 = RandomForestClassifier()

[47] rfr1.fit(x,y.values.reshape(-1))

     RandomForestClassifier
     RandomForestClassifier()

[48] y_pred = rfr1.predict(x_test)

[49] y_pred = rfr1.predict(x_test)

[50] from sklearn.metrics import classification_report
     print(classification_report(y_pred,y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00        81
           3       1.00      1.00      1.00        83
           4       1.00      1.00      1.00       122
           5       1.00      1.00      1.00        38
           6       1.00      1.00      1.00        59
           7       1.00      1.00      1.00        58

    accuracy                           1.00       445
   macro avg       1.00      1.00      1.00       445
weighted avg       1.00      1.00      1.00       445

[51] from sklearn.metrics import accuracy_score
     train_score = accuracy_score(y, rfr1.predict(x))
     train_score
```

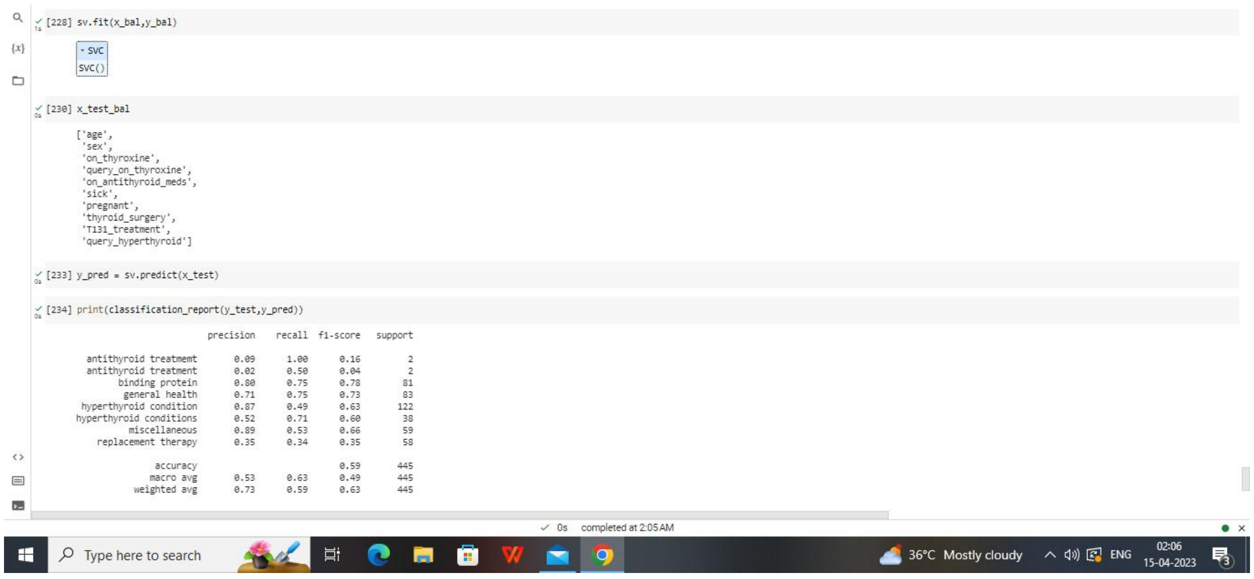
thyroid.ipynb - Colaboratory
colab.research.google.com/drive/12ZdF4I_HBfQ40c34pp1yqa_5KLYJWUdj#scrollTo=B-inR5Ri_Q7U
thyroid.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Files
sample_data
Thyroid.csv
Executing (53m 44s) <cell line: 1> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > retrieve() > wrap_future_result() > result() > wait()
35°C Partly sunny
03:22
23-04-2023

Xgb Classifier Model

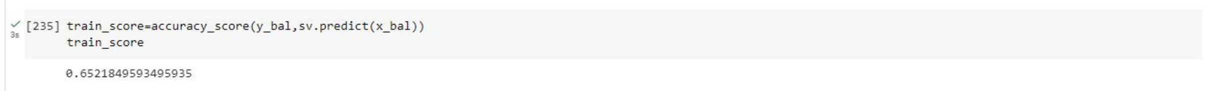


```
1.0

[219] from xgboost import XGBClassifier
      xgb1 = XGBClassifier()
      xgb1.fit(x,y)

      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, ...)

[220] y_pred = xgb1.predict(x_test)

[236] print(classification_report(y_test,y_pred))

                          precision    recall  f1-score   support

   antithyroid treatmemt       0.09      1.00      0.16         2
   antithyroid treatment       0.02      0.50      0.04         2
         binding protein       0.80      0.75      0.78        81
          general health       0.71      0.75      0.73        83
    hyperthyroid condition      0.87      0.49      0.63       122
  hyperthyroid conditions      0.52      0.71      0.60        38
           miscellaneous       0.89      0.53      0.66        59
     replacement therapy       0.35      0.34      0.35        58

                accuracy                           0.59       445
               macro avg       0.53      0.63      0.49       445
```

SVC model

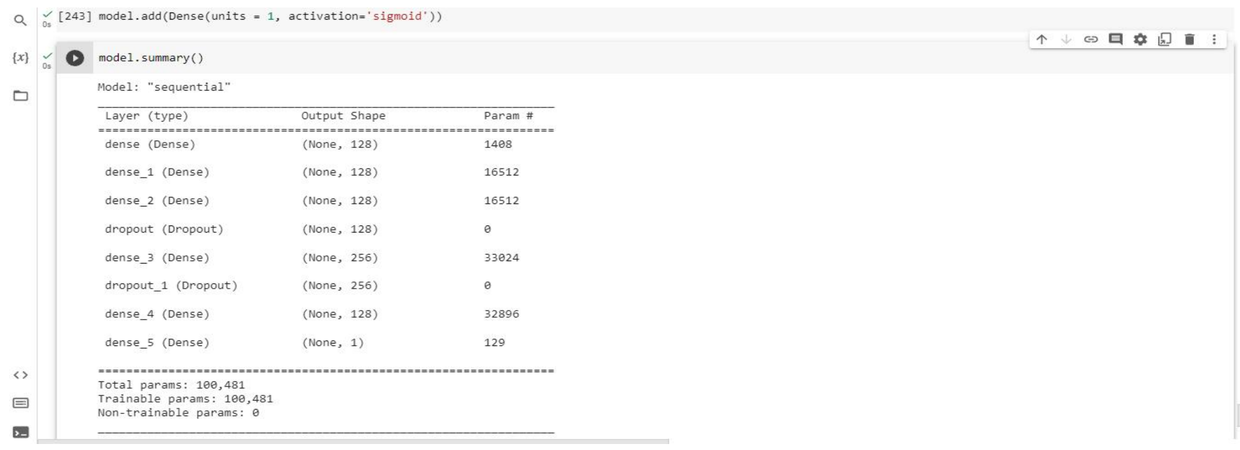


```
[228] sv.fit(x_bal,y_bal)

      SVC
      SVC()

[230] x_test_bal

      ['age',
       'sex',
       'on_thyroxine',
       'query_on_thyroxine',
       'on_antithyroid_meds',
       'sick',
       'pregnant',
       'thyroid_surgery',
       'T131_treatment',
       'query_hyperthyroid']

[233] y_pred = sv.predict(x_test)

[234] print(classification_report(y_test,y_pred))

                          precision    recall  f1-score   support

   antithyroid treatmemt       0.09      1.00      0.16         2
   antithyroid treatment       0.02      0.50      0.04         2
         binding protein       0.80      0.75      0.78        81
          general health       0.71      0.75      0.73        83
    hyperthyroid condition      0.87      0.49      0.63       122
  hyperthyroid conditions      0.52      0.71      0.60        38
           miscellaneous       0.89      0.53      0.66        59
     replacement therapy       0.35      0.34      0.35        58

                accuracy                           0.59       445
               macro avg       0.53      0.63      0.49       445
            weighted avg       0.73      0.59      0.63       445
```



```
[235] train_score=accuracy_score(y_bal,sv.predict(x_bal))
      train_score

      0.6521849593495935
```

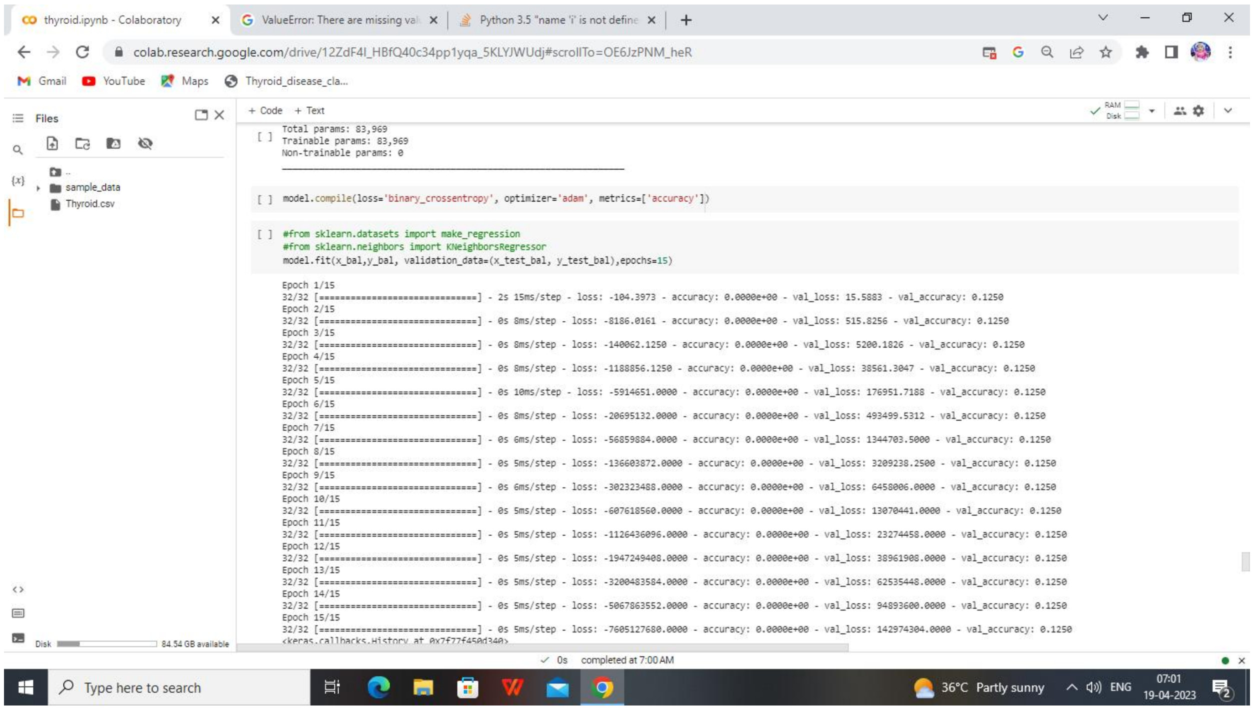
ANN Model



```
[243] model.add(Dense(units = 1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               1408

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dropout (Dropout)           (None, 128)               0

 dense_3 (Dense)             (None, 256)               33024

 dropout_1 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 128)               32896

 dense_5 (Dense)             (None, 1)                 129

=================================================================
Total params: 100,481
Trainable params: 100,481
Non-trainable params: 0
_________________________________________________________________
```



thyroid.ipynb - Colaboratory
ValueError: There are missing val
Python 3.5 "name 'i' is not define
colab.research.google.com/drive/12ZdF4I_HBfQ40c34pp1yqa_5KLYJWUdj#scrollTo=OE6JzPNM_heR
Gmail
YouTube
Maps
Thyroid_disease_cla...
Files
sample_data
Thyroid.csv
+ Code
+ Text
Disk 84.54 GB available

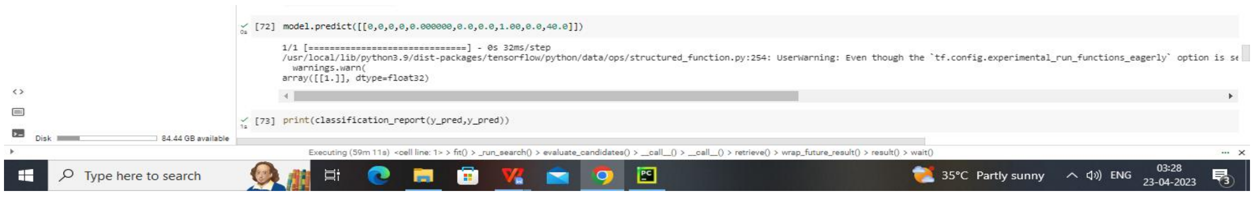
```
Total params: 83,969
Trainable params: 83,969
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
#from sklearn.datasets import make_regression
#from sklearn.neighbors import KNeighborsRegressor
model.fit(x_bal,y_bal, validation_data=(x_test_bal, y_test_bal),epochs=15)
```

```
Epoch 1/15
32/32 [==============================] - 2s 15ms/step - loss: -104.3973 - accuracy: 0.0000e+00 - val_loss: 15.5883 - val_accuracy: 0.1250
Epoch 2/15
32/32 [==============================] - 0s 8ms/step - loss: -8186.0161 - accuracy: 0.0000e+00 - val_loss: 515.8256 - val_accuracy: 0.1250
Epoch 3/15
32/32 [==============================] - 0s 8ms/step - loss: -140062.1250 - accuracy: 0.0000e+00 - val_loss: 5200.1826 - val_accuracy: 0.1250
Epoch 4/15
32/32 [==============================] - 0s 8ms/step - loss: -1188856.1250 - accuracy: 0.0000e+00 - val_loss: 38561.3047 - val_accuracy: 0.1250
Epoch 5/15
32/32 [==============================] - 0s 10ms/step - loss: -5914651.0000 - accuracy: 0.0000e+00 - val_loss: 176951.7188 - val_accuracy: 0.1250
Epoch 6/15
32/32 [==============================] - 0s 8ms/step - loss: -20695132.0000 - accuracy: 0.0000e+00 - val_loss: 493499.5312 - val_accuracy: 0.1250
Epoch 7/15
32/32 [==============================] - 0s 6ms/step - loss: -56859884.0000 - accuracy: 0.0000e+00 - val_loss: 1344703.5000 - val_accuracy: 0.1250
Epoch 8/15
32/32 [==============================] - 0s 5ms/step - loss: -136603872.0000 - accuracy: 0.0000e+00 - val_loss: 3209238.2500 - val_accuracy: 0.1250
Epoch 9/15
32/32 [==============================] - 0s 6ms/step - loss: -302323488.0000 - accuracy: 0.0000e+00 - val_loss: 6458006.0000 - val_accuracy: 0.1250
Epoch 10/15
32/32 [==============================] - 0s 5ms/step - loss: -607618560.0000 - accuracy: 0.0000e+00 - val_loss: 13070441.0000 - val_accuracy: 0.1250
Epoch 11/15
32/32 [==============================] - 0s 5ms/step - loss: -1126436096.0000 - accuracy: 0.0000e+00 - val_loss: 23274458.0000 - val_accuracy: 0.1250
Epoch 12/15
32/32 [==============================] - 0s 5ms/step - loss: -1947249408.0000 - accuracy: 0.0000e+00 - val_loss: 38961908.0000 - val_accuracy: 0.1250
Epoch 13/15
32/32 [==============================] - 0s 5ms/step - loss: -3200483584.0000 - accuracy: 0.0000e+00 - val_loss: 62535448.0000 - val_accuracy: 0.1250
Epoch 14/15
32/32 [==============================] - 0s 5ms/step - loss: -5067863552.0000 - accuracy: 0.0000e+00 - val_loss: 94893600.0000 - val_accuracy: 0.1250
Epoch 15/15
32/32 [==============================] - 0s 5ms/step - loss: -7605127680.0000 - accuracy: 0.0000e+00 - val_loss: 142974304.0000 - val_accuracy: 0.1250
<keras.callbacks.History at 0x7f77f450d340>
```

Testing the model



```
[72] model.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])
```

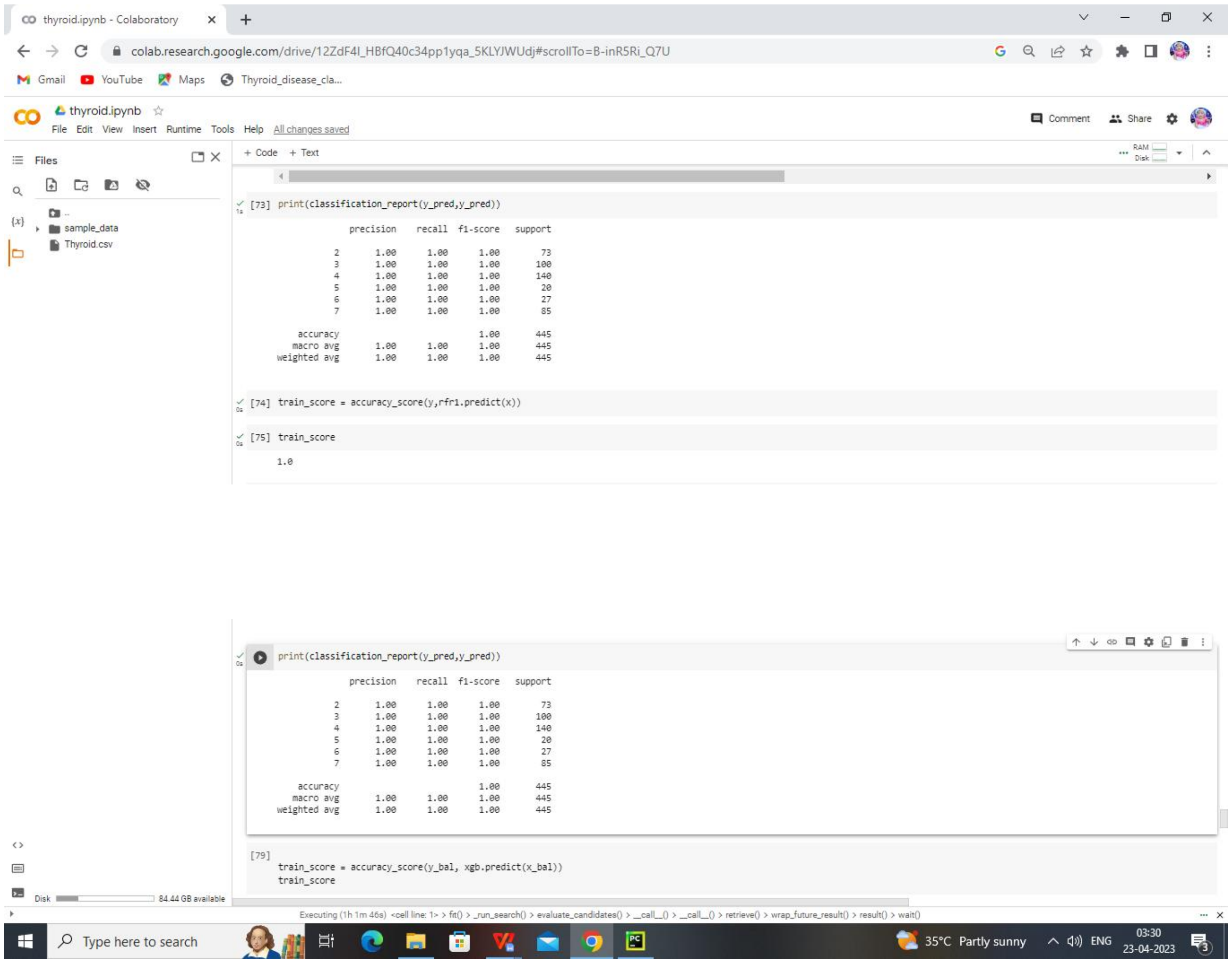
```
1/1 [==============================] - 0s 32ms/step
/usr/local/lib/python3.9/dist-packages/tensorflow/python/data/ops/structured_function.py:254: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly` option is se
  warnings.warn(
array([[1.]], dtype=float32)
```

```
[73] print(classification_report(y_pred,y_pred))
```

Disk 84.44 GB available
Executing (59m 11s) <cell line: 1> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > retrieve() > wrap_future_result() > result() > wait()
Type here to search
35°C Partly sunny
ENG
03:28
23-04-2023

Performance Testing & Hyperparameter Tuning:

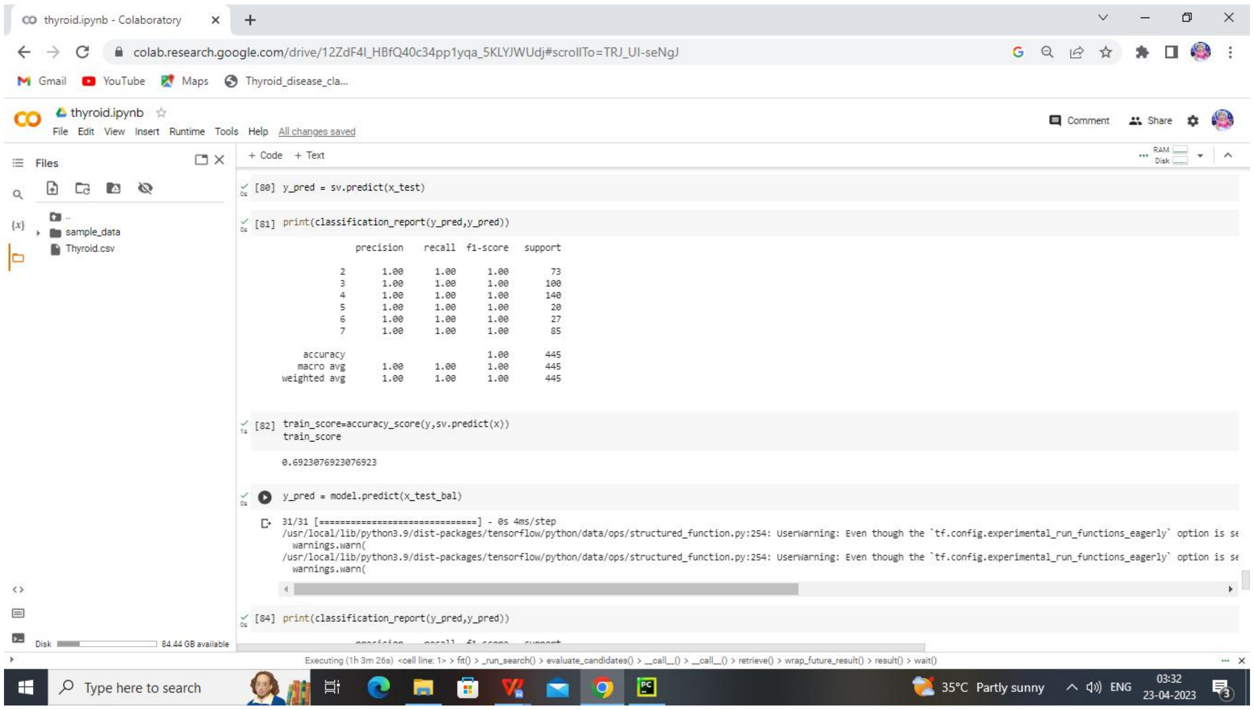
Compare the model:



```
[73] print(classification_report(y_pred,y_pred))

              precision    recall  f1-score   support

           2       1.00      1.00      1.00        73
           3       1.00      1.00      1.00       100
           4       1.00      1.00      1.00       140
           5       1.00      1.00      1.00        20
           6       1.00      1.00      1.00        27
           7       1.00      1.00      1.00        85

    accuracy                           1.00       445
   macro avg       1.00      1.00      1.00       445
weighted avg       1.00      1.00      1.00       445

[74] train_score = accuracy_score(y,rfr1.predict(x))

[75] train_score

1.0
```

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

              precision    recall  f1-score   support

           2       1.00      1.00      1.00        73
           3       1.00      1.00      1.00       100
           4       1.00      1.00      1.00       140
           5       1.00      1.00      1.00        20
           6       1.00      1.00      1.00        27
           7       1.00      1.00      1.00        85

    accuracy                           1.00       445
   macro avg       1.00      1.00      1.00       445
weighted avg       1.00      1.00      1.00       445

[79]
train_score = accuracy_score(y_bal, xgb.predict(x_bal))
train_score
```

Executing (1h 1m 46s) <cell line: 1> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > retrieve() > wrap_future_result() > result() > wait()



```
[80] y_pred = sv.predict(x_test)
```

```
[81] print(classification_report(y_pred,y_pred))
```

```
              precision    recall  f1-score   support

           2       1.00      1.00      1.00        73
           3       1.00      1.00      1.00       100
           4       1.00      1.00      1.00       140
           5       1.00      1.00      1.00        20
           6       1.00      1.00      1.00        27
           7       1.00      1.00      1.00        85

    accuracy                           1.00       445
   macro avg       1.00      1.00      1.00       445
weighted avg       1.00      1.00      1.00       445
```

```
[82] train_score=accuracy_score(y,sv.predict(x))
     train_score
```

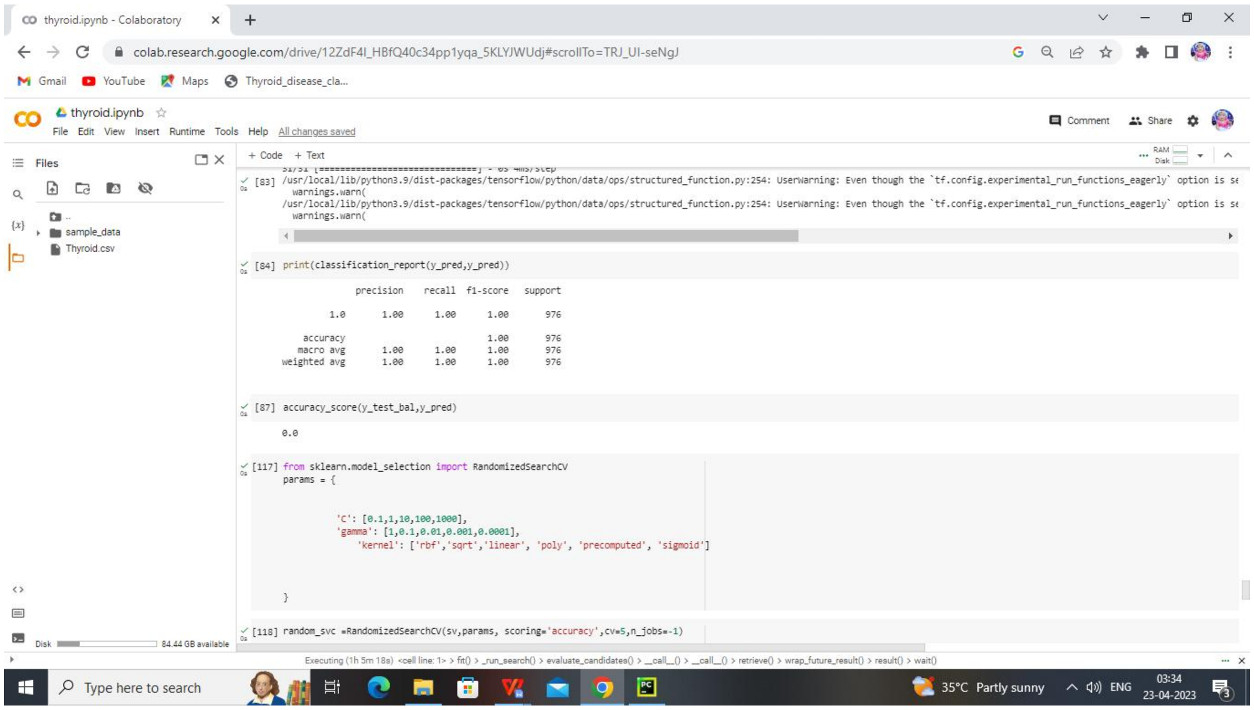
```
0.6923076923076923
```

```
y_pred = model.predict(x_test_bal)
```

```
31/31 [==============================] - 0s 4ms/step
/usr/local/lib/python3.9/dist-packages/tensorflow/python/data/ops/structured_function.py:254: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly` option is se
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/tensorflow/python/data/ops/structured_function.py:254: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly` option is se
  warnings.warn(
```

```
[84] print(classification_report(y_pred,y_pred))
```

Executing (1h 3m 26s) <cell line: 1> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > retrieve() > wrap_future_result() > result() > wait()



```
[83] /usr/local/lib/python3.9/dist-packages/tensorflow/python/data/ops/structured_function.py:254: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly` option is se
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/tensorflow/python/data/ops/structured_function.py:254: UserWarning: Even though the `tf.config.experimental_run_functions_eagerly` option is se
  warnings.warn(
```

```
[84] print(classification_report(y_pred,y_pred))
```

```
              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00       976

    accuracy                           1.00       976
   macro avg       1.00      1.00      1.00       976
weighted avg       1.00      1.00      1.00       976
```

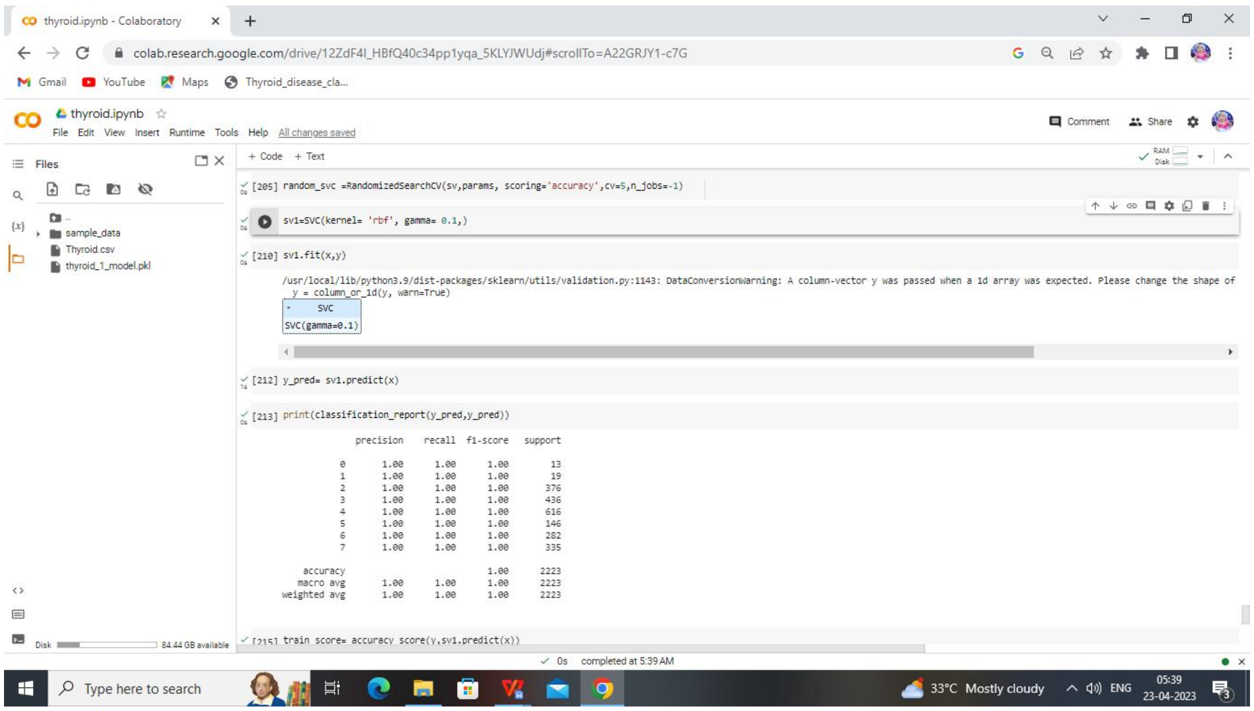
```
[87] accuracy_score(y_test_bal,y_pred)
```

```
0.0
```

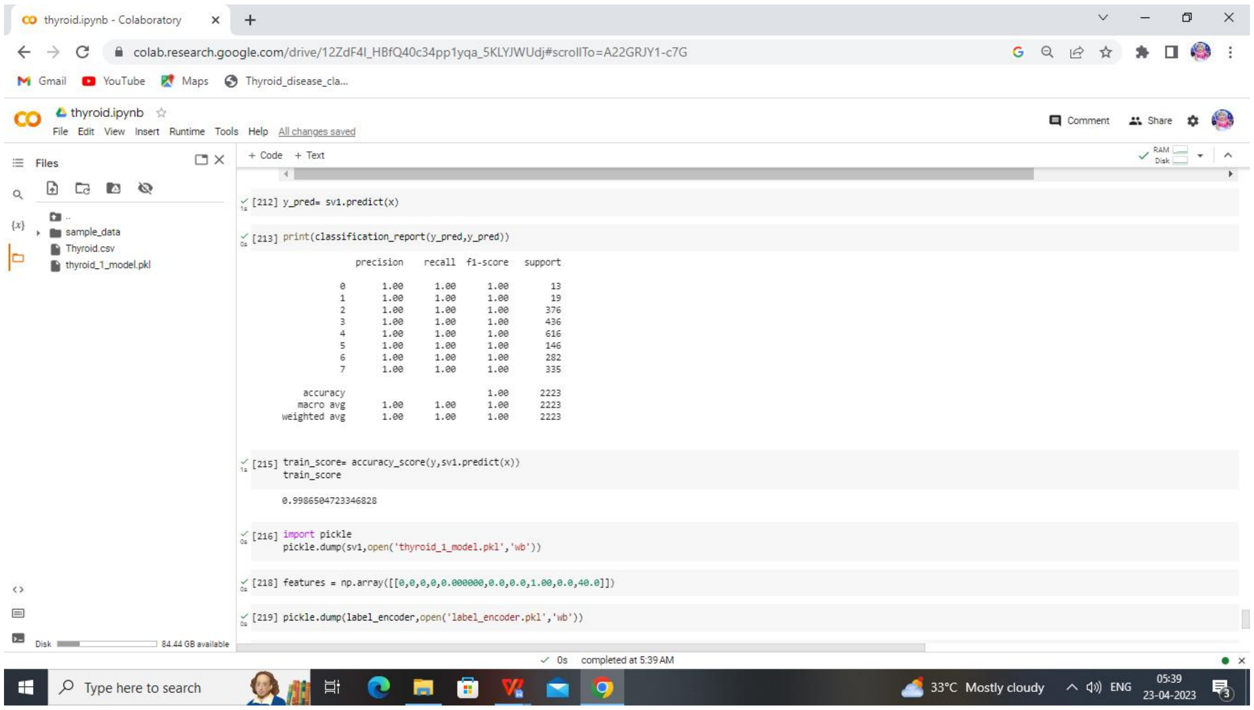
```
[117] from sklearn.model_selection import RandomizedSearchCV
      params = {


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



      }
```

```
[118] random_svc =RandomizedSearchCV(sv,params, scoring='accuracy',cv=5,n_jobs=-1)
```

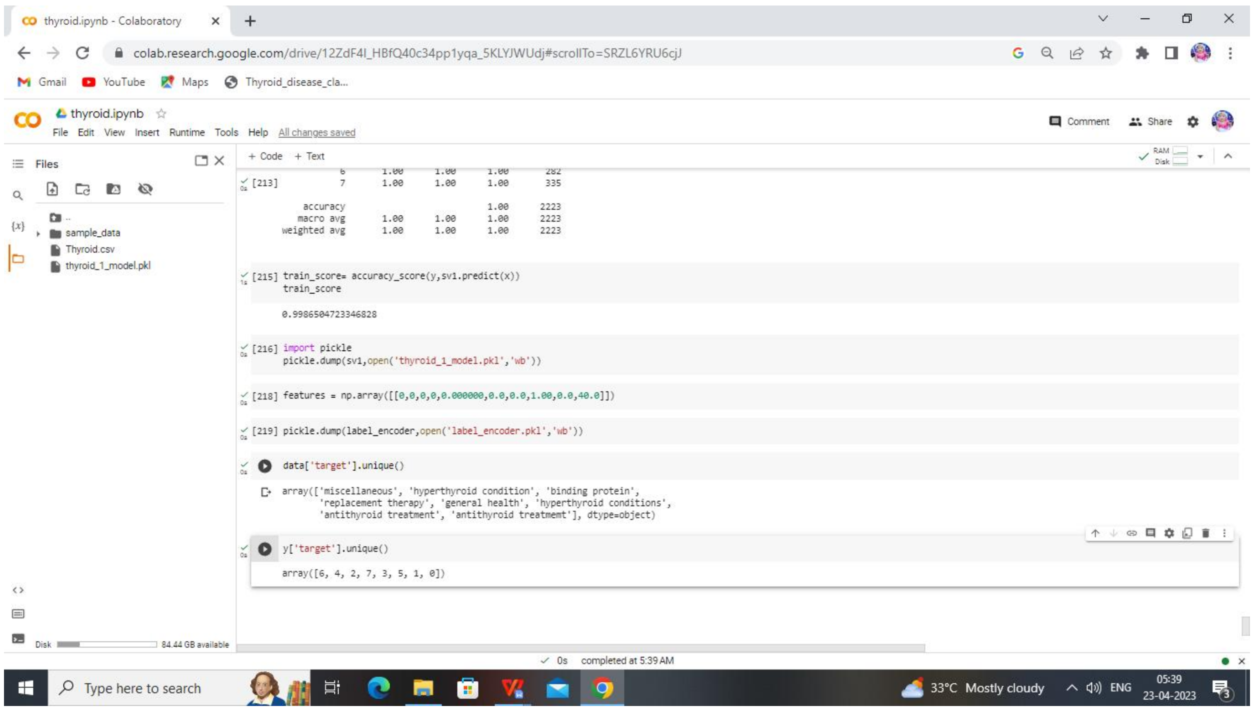
Executing (1h 5m 18s) <cell line: 1> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > retrieve() > wrap_future_result() > result() > wait()



```
[205] random_svc =RandomizedSearchCV(sv,params, scoring='accuracy',cv=5,n_jobs=-1)

      sv1=SVC(kernel= 'rbf', gamma= 0.1,)

[210] sv1.fit(x,y)

      /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 change the shape of
        y = column_or_1d(y, warn=True)
      SVC
      SVC(gamma=0.1)

[212] y_pred= sv1.predict(x)

[213] print(classification_report(y_pred,y_pred))

                    precision    recall  f1-score   support

                 0       1.00      1.00      1.00        13
                 1       1.00      1.00      1.00        19
                 2       1.00      1.00      1.00       376
                 3       1.00      1.00      1.00       436
                 4       1.00      1.00      1.00       616
                 5       1.00      1.00      1.00       146
                 6       1.00      1.00      1.00       282
                 7       1.00      1.00      1.00       335

          accuracy                           1.00      2223
         macro avg       1.00      1.00      1.00      2223
      weighted avg       1.00      1.00      1.00      2223

[215] train_score= accuracy_score(y,sv1.predict(x))
```



```
[212] y_pred= sv1.predict(x)

[213] print(classification_report(y_pred,y_pred))

                    precision    recall  f1-score   support

                 0       1.00      1.00      1.00        13
                 1       1.00      1.00      1.00        19
                 2       1.00      1.00      1.00       376
                 3       1.00      1.00      1.00       436
                 4       1.00      1.00      1.00       616
                 5       1.00      1.00      1.00       146
                 6       1.00      1.00      1.00       282
                 7       1.00      1.00      1.00       335

          accuracy                           1.00      2223
         macro avg       1.00      1.00      1.00      2223
      weighted avg       1.00      1.00      1.00      2223

[215] train_score= accuracy_score(y,sv1.predict(x))
      train_score

      0.9986504723346828

[216] import pickle
      pickle.dump(sv1,open('thyroid_1_model.pkl','wb'))

[218] features = np.array([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

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



```
              6       1.00      1.00      1.00       282
[213]         7       1.00      1.00      1.00       335

       accuracy                           1.00      2223
      macro avg       1.00      1.00      1.00      2223
   weighted avg       1.00      1.00      1.00      2223
```

```
[215] train_score= accuracy_score(y,sv1.predict(x))
      train_score

      0.9986504723346828
```

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

```
[218] features = np.array([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])
```

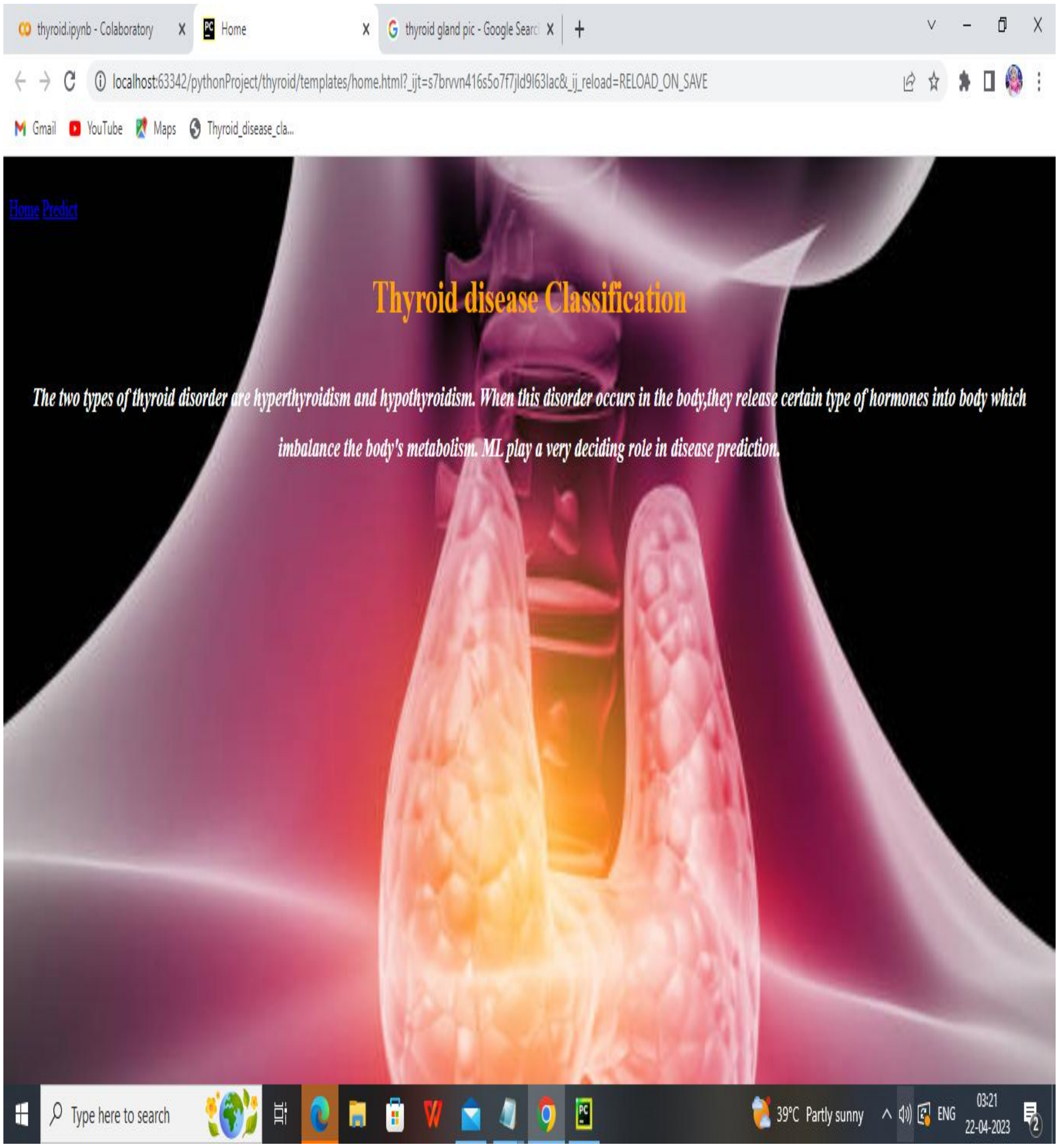
```
[219] pickle.dump(label_encoder,open('label_encoder.pkl','wb'))
```

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

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

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

array([6, 4, 2, 7, 3, 5, 1, 0])
```

Integrate with Web Development:

Building web page(home.htm):

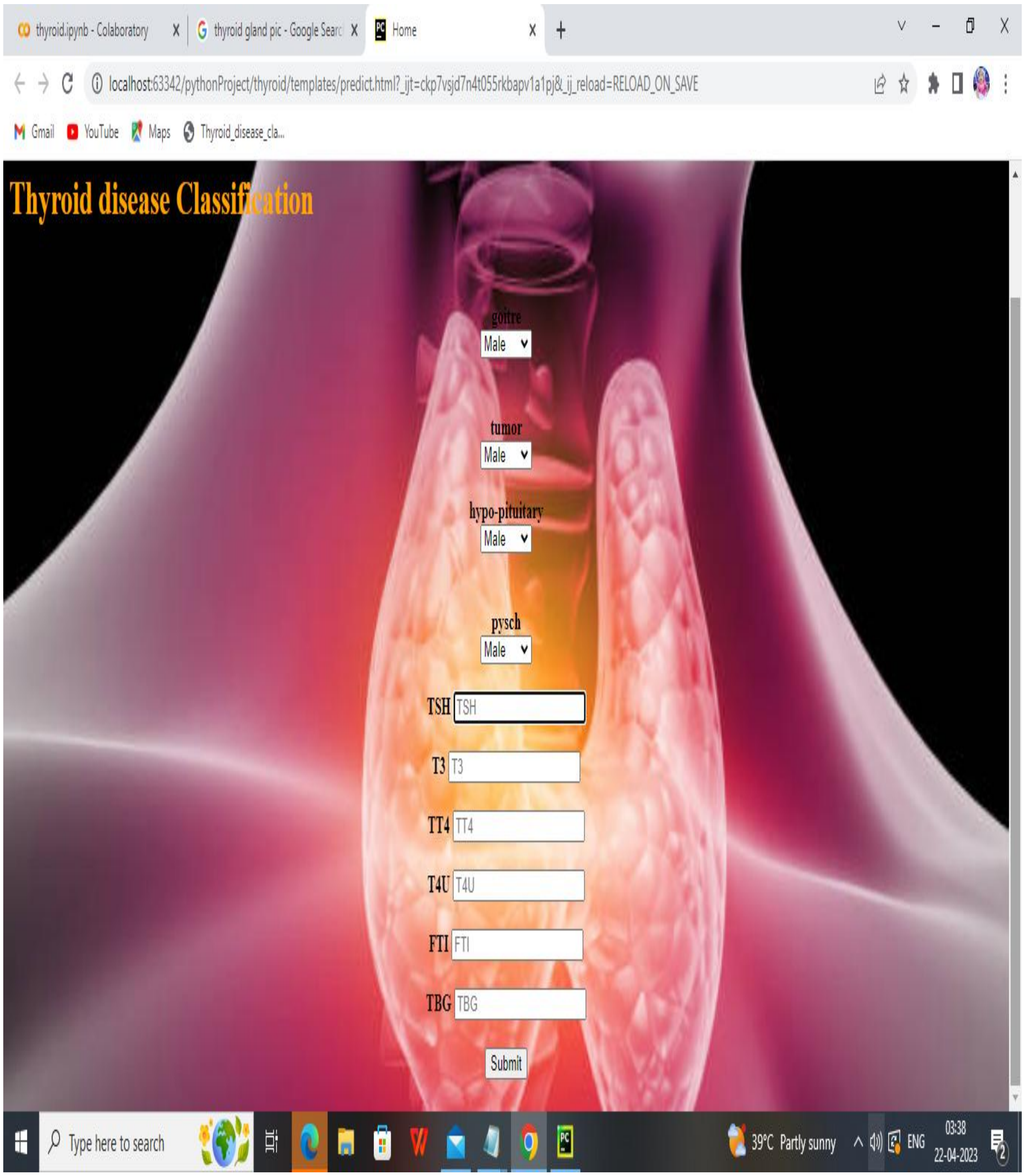

thyroid.ipynb - Colaboratory
Home
thyroid gland pic - Google Searc
localhost:63342/pythonProject/thyroid/templates/home.html?_ijt=s7brvvn416s5o7f7jld9l63lac&_ij_reload=RELOAD_ON_SAVE
Gmail
YouTube
Maps
Thyroid_disease_cla...
Home Predict

Thyroid disease Classification

The two types of thyroid disorder are hyperthyroidism and hypothyroidism. When this disorder occurs in the body,they release certain type of hormones into body which imbalance the body's metabolism. ML play a very deciding role in disease prediction.

Type here to search
39°C Partly sunny
ENG
03:21
22-04-2023

Predict.html web page:



thyroid.ipynb - Colaboratory
thyroid gland pic - Google Searc
Home
localhost:63342/pythonProject/thyroid/templates/predict.html?_ijt=ckp7vsjd7n4t055rkbapv1a1pj&_ij_reload=RELOAD_ON_SAVE
Gmail
YouTube
Maps
Thyroid_disease_cla...
Type here to search
39°C Partly sunny
ENG
03:38
22-04-2023

Thyroid disease Classification

goitre
Male

tumor
Male

hypo-pituitary
Male

pysch
Male

TSH TSH

T3 T3

TT4 TT4

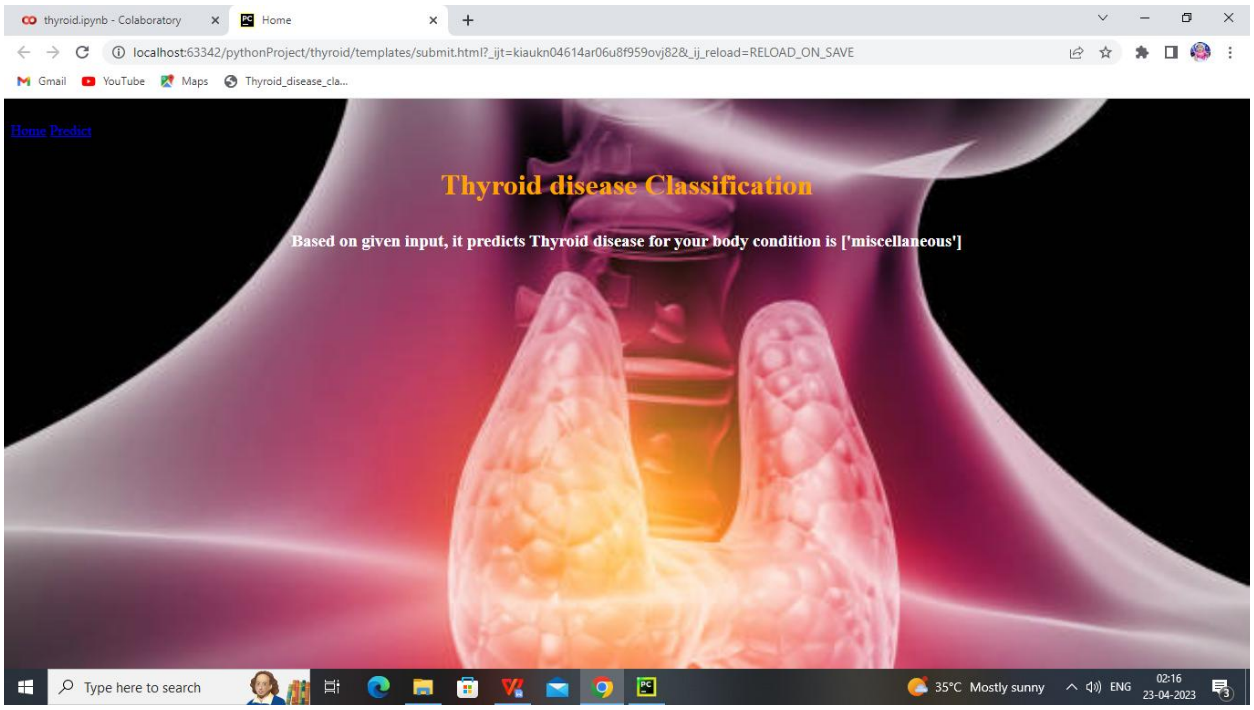
T4U T4U

FTI FTI

TBG TBG

Submit

Result(After click submit button):


thyroid.ipynb - Colaboratory
Home
localhost:63342/pythonProject/thyroid/templates/submit.html?_ijt=kiaukn04614ar06u8f959ovj82&_ij_reload=RELOAD_ON_SAVE
Gmail
YouTube
Maps
Thyroid_disease_cla...
Home Predict
Type here to search
35°C Mostly sunny
ENG
02:16
23-04-2023

Thyroid disease Classification

Based on given input, it predicts Thyroid disease for your body condition is ['miscellaneous']

Advantages and Disadvantages :

Classification Model	Advantages	Disadvantages
Logistic Regression	Probabilistic Approach, gives information about statistical significance of features.	The assumptions of logistic regression.
K – Nearest Neighbors	Simple to understand, fast and efficient.	Need to manually choose the number of neighbors 'k'.
Support Vector Machine (SVM)	Perform ant, not biased by outliers, not sensitive to over fitting.	Not appropriate for non-linear problems, not the best choice for large number of features.

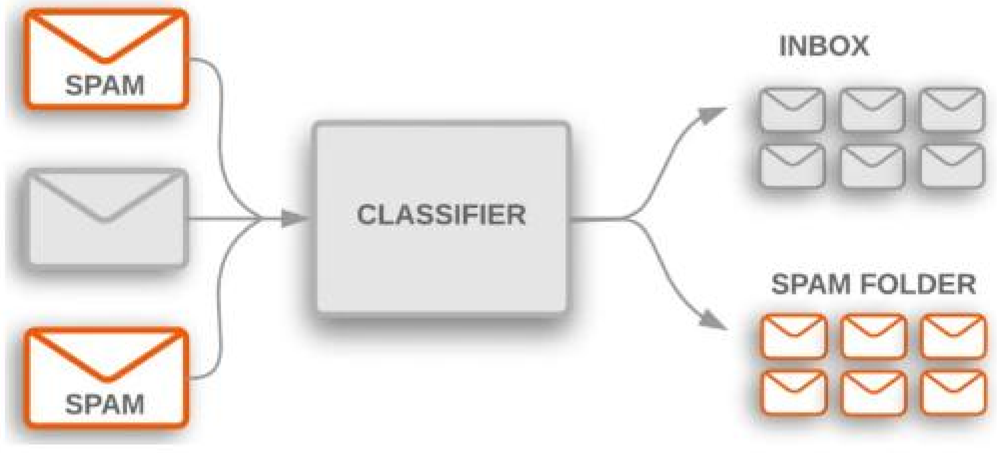
Kernel SVM	High performance on non – linear problems, not biased by outliers, not sensitive to over fitting.	Not the best choice for large number of features, more complex.
Naive Bayes	Efficient, not biased by outliers, works on non – linear problems, probabilistic approach.	Based in the assumption that the features have same statistical relevance.
Decision Tree Classification	Interpret ability, no need for feature scaling, works on both linear / non – linear problems.	Poor results on very small datasets, over fitting can easily occur.
Random Forest Classification	Powerful and accurate, good performance on many problems, including non – linear.	No interpret ability, over fitting can easily occur, need to choose the number of trees manually.

Applications:

◆ *Real-life examples of machine learning classification problems:*

❖ **Customer behavior prediction**: Customers can be classified into different categories based on their buying patterns, web store browsing patterns etc. For example, classification models can be used to determine whether a customer is likely to purchase more items or not. If the classification model predicts a greater likelihood that they are about to make more purchases, then you might want to send them promotional offers and discounts accordingly. Or if it has been determined that they will probably fall off of their purchasing habits soon, maybe save them for later by making their information readily available.

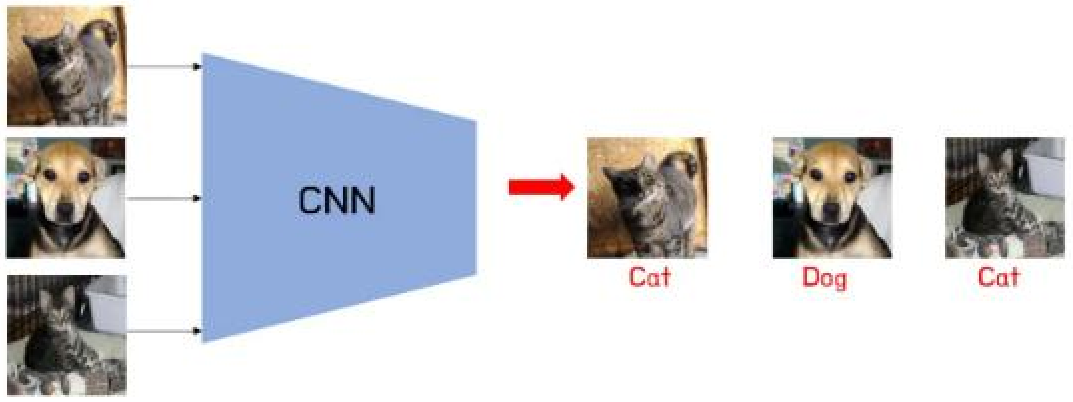
- **Document classification**: A multinational classification model can be trained to classify documents in different categories. In this case, the classification model can be thought of as a function that maps from a document to a category label. Different algorithms can be used for document classification such as Naive Bayes classifier, Support Vector Machines (SVM), or Neural Networks models. Deep learning algorithms such as Deep Boltzmann Machines (DBM), Deep Belief Networks (DBN), and Stacked Auto Encoder (SAE) give state-of-the-art classification results on different document classification datasets.
- **Spam filtering**: An algorithm is trained to recognize spam email by learning the characteristics of what constitutes spam vs non-spam email. The classification model could be a function that maps from an email text to a spam classification (or non-spam classification). Algorithms such as Naive Bayes and Support Vector Machines can be used for classification. Once the classification model is trained, it can then be used to filter new incoming emails as spam or non-spam. The picture below represents the Spam classification model depicted as Spam classifier.



- **Image classification**: One of the most popular classification problems is image classification: determining what type of object (or scene) is in a digital image. Images can be thought of as a high-dimensional vectors which we would like to classify into different classes such as cat, car, human, and airplane. A Multinomial classification model can be trained to classify images into different categories.

For example

In order to classify images of dogs and cats for use within machine vision systems, machine learning techniques can help automate this process based on per-classified images of dogs and cats.rent categories. Deep learning algorithms such as Convolution Neural Networks (CNN)-based classification models are state-of-the-art in different image classification tasks. Another use case is image segmentation, where the pixels of an image are assigned a label based on what object they belong to. Image segmentation is defined as “the process of distinguishing semantically meaningful image regions on the basis of visual features”. The picture below represents how the CNN algorithm can be used to build a classification model that classifies images such as Cat and dog.



CONCLUSION:

- ❖ Thyroid disease is one of the diseases that afflict the world's population, and the number of cases of this disease is increasing. Because of medical reports that show serious imbalances in thyroid diseases, our study deals with the classification of thyroid disease between hyperthyroidism and hypothyroidism. This disease was classified using algorithms. Machine learning showed us good results using several algorithms and was built in the form of two models. In the first model, all the characteristics consisting of 16 inputs and one output were taken, and the result of the accuracy of the random forest algorithm was 98.93, which is the highest accuracy among the other algorithms. In the second embodiment, the following characteristics were omitted based on a previous study. The removed attributes were 1- query_thyroxine 2- query_hypothyroid 3-query_hyperthyroid. Here we have included the increased accuracy of some algorithms, as well as the retention of the accuracy of others. It was observed that the accuracy of Naive Bayes algorithm increased the accuracy by 90.67. The highest precision of the MLP algorithm was 96.4 accuracy.

FUTURE SCOPE:

- ✧ The scope of machine learning is not limited to the investment sector.
- ✧ Rather, it is expanding across all fields such as banking and finance,information technology,media & entertainment, gaming, and the automotive industry.

✧ As the Machine learning scope is very high, there are some areas where researchers are working toward revolutionizing the world for the future.

SOURCE CODE

```
import pandas as pd

import numpy as np

import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Layer, Dense, Dropout

data = pd.read_csv('/content/Thyroid.csv')

data.head()

data.shape

data.isnull().sum()

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

diagnoses = {'A': 'hyperthyroid conditions',

             'B': 'hyperthyroid condition',

             'c': 'hyperthyroid condition',

             'd': 'hyperthyroid condition',

             'E':'hyperthyroid condition',

             'F':'hyperthyroid condition',

             'G':'hyperthyroid condition',

             'H':'hyperthyroid ciondition',

             '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)
data.dropna(subset=['target'],inplace=True)
data['target'].value_counts()
data[data.age>100]
x=data.iloc[:,0:-1]
y=data.iloc[:,-1]
X
x['sex'].unique()
x['sex'].replace(np.nan, 'F', inplace=True)
x['sex'].value_counts()
x['age']=x['age'].astype('float')
x['TSH']=x['TSH'].astype('float')
x['T3']=x['T3'].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')
x.info()
#Encoding the categorical data
```

```
#Encoding the independent(output) variable

from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

#categorical data


ordinal_encoder = OrdinalEncoder(dtype = 'int64')

x.iloc[:, 1:16] = ordinal_encoder.fit_transform(x.iloc[:, 1:16])

#ordinal_encoder.fit_transform(x[['sex']])

X

x=data.iloc[:,0:-1]

y=data.iloc[:,-1]

X

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)

from imblearn.over_sampling import SMOTE

y_train.value_counts()

x.replace(np.nan, 0, inplace=True)

x.replace('F', 0, inplace=True)

x.replace('M', 1, inplace=True)

x.replace('f', 0, inplace=True)

x.replace('t', 1, inplace=True)

X

label_encoder = LabelEncoder()

y_dt= label_encoder.fit_transform(y)

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

Y

x_test.replace('F', 0, inplace=True)

x_test.replace('M', 1, inplace=True)
x_test.replace('f', 0, inplace=True)

x_test.replace('t', 1, inplace=True)

x_test.replace(np.nan, '0', inplace=True)

x_test

x_train.replace('F', 0, inplace=True)

x_train.replace('M', 1, inplace=True)

x_train.replace('f', 0, inplace=True)

x_train.replace('t', 1, inplace=True)

x_train.replace(np.nan, '0', inplace=True)

x_train

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)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_bal = sc.fit_transform(x_bal)

x_test_bal = sc.transform(x_test_bal)

x_bal

# permutation feature importance with knn for regression

from sklearn.datasets import make_regression

from sklearn.neighbors import KNeighborsRegressor

from sklearn.inspection import permutation_importance

import numpy as np

from matplotlib import pyplot

# define dataset

x_bal, y_bal = make_regression(n_samples=1000, n_features=22, n_informative=5, ra
ndom_state=1)
```

```
# define the model

model = KNeighborsRegressor()

# fit the model

model.fit(x_bal, y_bal)

# perform permutation importance

results = permutation_importance(model, x_bal, y_bal, scoring='neg_mean_squared_e
rror')

# get importance

feature_importance=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyro
id_meds','sick','pregnant','thyroid_surgery','T131_treatment','query_hyperthyroid
','query_hyperthyroid','goitre','tumor','hypopituitary','psych','TSH','T3','TT4',
'T4U','FTI','TBG','target']

importance = results.importances_mean

importance = np.sort(importance)

# summarize feature importance

for i,v in enumerate(importance):

  i=feature_importance[i]

  print('Feature: {:<20} Score: {}'. format(i,v))

# plot feature importance

pyplot.figure(figsize=(10,10))

pyplot.bar(x=feature_importance, height = importance)

pyplot.xticks(rotation=30, ha='right')

pyplot.show()

x.head()

x_bal

columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','s
ick','pregnant','thyroid_surgery','T131_treatment','query_hyperthyroid','goitre',
'tumor','hypopituitary','psych','TSH','T3','TT4','T4U','FTI','TBG','target','pati
ent_id']

x_test_bal=pd.DataFrame(x_test_bal,columns=columns)
```

```
x_bal= pd.DataFrame(x_bal,columns=columns)


x_bal.pop('target')

x_bal.pop('patient_id')

x_bal=x_bal.drop(x_bal.loc[:, 'age':'query_hyperthyroid'].columns, axis=1)


x_test_bal.pop('target')

x_test_bal.pop('patient_id')

x_test_bal=x_test_bal.drop(x_test_bal.loc[:, 'age':'query_hyperthyroid'].columns,
 axis=1)

x_bal.head()

x_test_bal.head()

data.info()

#checking correlation using Heat map

import seaborn as sns

import matplotlib.pyplot as plt


corrmat =x.corr()


f, ax = plt.subplots(figsize =(9, 8))

sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1)

x_bal

from sklearn.ensemble import RandomForestClassifier

rfr1 =  RandomForestClassifier().fit(x,y.values.ravel())

y_pred = rfr1.predict(x_test)


rfr1 = RandomForestClassifier()
```

```
rfr1.fit(x,y.values.reshape(-1))

y_pred = rfr1.predict(x_test)

y_pred = rfr1.predict(x_test)

from sklearn.metrics import classification_report

print(classification_report(y_pred,y_pred))

from sklearn.metrics import accuracy_score

train_score = accuracy_score(y, rfr1.predict(x))

train_score

from xgboost import XGBClassifier

xgb1 = XGBClassifier()

xgb1.fit(x,y)

from xgboost import XGBClassifier

xgb1 = XGBClassifier()

xgb1.fit(x,y)

y_pred = xgb1.predict(x)

print(classification_report(y_pred,y_pred))

accuracy_score(y_pred,y_pred)

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report

sv= SVC()

sv.fit(x,y)

y_pred = sv.predict(x_test)

print(classification_report(y_pred,y_pred))

train_score=accuracy_score(y,sv.predict(x))

train_score

model = Sequential()

model.add(Dense(units = 128, activation='relu', input_shape=(10,)))
model.add(Dense(units = 128, activation='relu', kernel_initializer='random_unifor
m'))

model.add(Dropout(0.2))

model.add(Dense(units = 256, activation='relu', kernel_initializer='random_unifor
m'))

model.add(Dropout(0.2))

model.add(Dense(units = 128, activation='relu', kernel_initializer='random_unifor
m'))

model.add(Dense(units = 1, activation='sigmoid'))

model.summary()

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

#from sklearn.datasets import make_regression

#from sklearn.neighbors import KNeighborsRegressor

model.fit(x,y, validation_data=(x,y),epochs=15)

x.head()

rfr1.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

sv.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

col = ['goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'F
TI', 'TBG']

da =[[0,0,0,0,0.00000,0.0,0.0,1.00,0.0,40.0]]

da1 = pd.DataFrame(data = da, columns=col)

xgb1.predict(da1)

model.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

print(classification_report(y_pred,y_pred))

train_score = accuracy_score(y,rfr1.predict(x_bal))

train_score

y_pred=xgb.predict(x_test_bal)

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

```
train_score = accuracy_score(y_bal, xgb.predict(x_bal))

train_score

y_pred = sv.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

train_score=accuracy_score(y_bal,sv.predict(x_bal))

train_score

y_pred = model.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

accuracy_score(y_test_bal,y_pred)

from sklearn.model_selection import RandomizedSearchCV

params = {



          'c': [0.1,1,10,100,1000],

          'gamma': [1,0.1,0.01,0.001,0.0001],

              'kernel': ['rbf','sqrt']


}

random_svc = RandomizedSearchCV(sv,params, scoring='accuracy',cv=5,n_jobs=-1)

random_svc.fit(x_bal,y_bal)

random_svc.best_params_

sv1=SVC(kernel= 'rbf', gamma= 0.1, c=100)

sv1.fit(x_bal,y_bal)

y_pred= sv1.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

train_score= accuracy_score(y_bal,sv1.predict(x_bal))

train_score
```

```
import pickle

pickle.dump(sv1,open('thyroid_1_model.pkl','wb'))

features = np.array([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

print(label_encoder.inverse_transform(xgb1.predict(features)))

pickle.dump(label_encoder,open('label_encoder.pkl','wb'))

data['target'].unique()

y['target'].unique()


#model deployment

#build HTML pages

#creating and designing web page using home.html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Home</title>
<style>
body
{
background-image:url("https://media.istockphoto.com/id/1198780608/photo/human-
thyroid-
anatomy.jpg?s=612x612&w=0&k=20&c=A96BQGdc57yAqHZI3cS2xrHJTPcUBvXsXQqdNSWh
8jA=");
background-size:cover;
}
h3.big
{
line-height:1.8;
}
```

```html
<text color = white></text>
</style>
</head>
<body>
<br>
<div class="container">
<div class="row">
<div class="col-md-12 bg-light text-right">
<a href="/home" class="btn btn-info btn-lg">Home</a>
<a href="/predict" class="btn btn-primary disabled btn-lg">Predict</a>
</div>
   </div>
   <center>
   <h1><strong><p style="color: orange">Thyroid disease Classification</p></strong></h1>
<h3 class="big"><em><p style="color: white">The two types of thyroid disorder are
hyperthyroidism and hypothyroidism.
   When this disorder occurs in the body,they release certain type of hormones into body which
imbalance the body's metabolism.
   ML play a very deciding role in disease prediction.</p>
</em></h3><br>
   </center>
</div>


<script src="https://ajax.googleapix.com/ajax/libs/jquer/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</body>
</html>
```

#create predict.html file we create web page and designing

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset ="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Home</title>
<style>
body
{
background-image:url("https://media.istockphoto.com/id/1198780608/photo/human-
thyroid-
anatomy.jpg?s=612x612&w=0&k=20&c=A96BQGdc57yAqHZI3cS2xrHJTPcUBvXsXQqdNSWh
8jA=");
background-size:cover;
}
h3.big
{
line-height:1.8;
}
</style>
</head>
<body>
<br>
<div class="container">
<div class="row">
<div class="col-md-12 bg-light text-right">
<a href="/home" class="btn btn-info btn-lg">Home</a>
<a href="/predict" class="btn btn-primary disabled btn-lg">Predict</a>
</div>
</div>
<br>
<h1><strong><p style="color: orange">Thyroid disease Classification</p></strong></h1>
```

```
<h4>
<form action = "/pred", method="POST">
   <center>
<div><br>

<div class="form-group mb-3">
<div class="input-group-perpend">
   <label  class="input-group-text" for="goitre">goitre</label>
</div>
<select class="custom-select" id="goitre" name="goitre">
<option value="1">Male</option>
<option value="2">Female</option>
</select>
</div><br>

<div><br>
<div class="form-group mb-3">
<div class="input-group-perpend">
   <label  class="input-group-text" for="tumor">tumor</label>
</div>
<select class="custom-select" id="tumor" name="tumor">
<option value="1">Male</option>
<option value="2">Female</option>
</select>
</div><br>
   <div class="form-group mb-3">
<div class="input-group-perpend">
   <label  class="input-group-text" for="hypo-pituitary">hypo-pituitary</label>
</div>
<select class="custom-select" id="hypo-pituitary" name="hypo-pituitary">
<option value="1">Male</option>
<option value="2">Female</option>
</select>
```

```
</div><br>


<div><br>
<div class="form-group mb-3">
<div class="input-group-perpend">
  <label  class="input-group-text" for="psych">psych</label>
</div>
<select class="custom-select" id="psych" name="psych">
<option value="1">Male</option>
<option value="2">Female</option>
</select>
</div><br>
<div class="form-group row">
<div class="col-md-3">
<label for="TSH">TSH</label>
<input type="text" class="form-control" name="TSH" placeholder="TSH">
</div>
<div><br>

<div class="form-group row">
<div class="col-md-3">
<label for="T3">T3</label>
<input type="text" class="form-control" name="T3" placeholder="T3">
</div>
<div><br>



<div class="form-group row">
<div class="col-md-3">
<label for="TT4">TT4</label>
<input type="text" class="form-control" name="TT4" placeholder="TT4">
</div>
<div><br>
```

```html
<div class="form-group row">
<div class="col-md-3">
<label for="T4U">T4U</label>
<input type="text" class="T4U" name="T4U" placeholder="T4U">
</div>
<div><br>



<div class="form-group row">
<div class="col-md-3">
<label for="FTI">FTI</label>
<input type="text" class="form-control" name="FTI" placeholder="FTI">
</div>
<div><br>
   <div class="form-group row">
<div class="col-md-3">
<label for="TBG">TBG</label>
<input type="text" class="form-control" name="TBG" placeholder="TBG">
</div>
<div><br>

<button type="submit" class="btn btn-success btn-lg">Submit</button>
</>
<br>
</h4>
</div>
  </center>
</form>
```

```
<script src="https://ajax.googleapix.com/ajax/libs/jquer/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</body>
</html>
```

#create a file submit.html

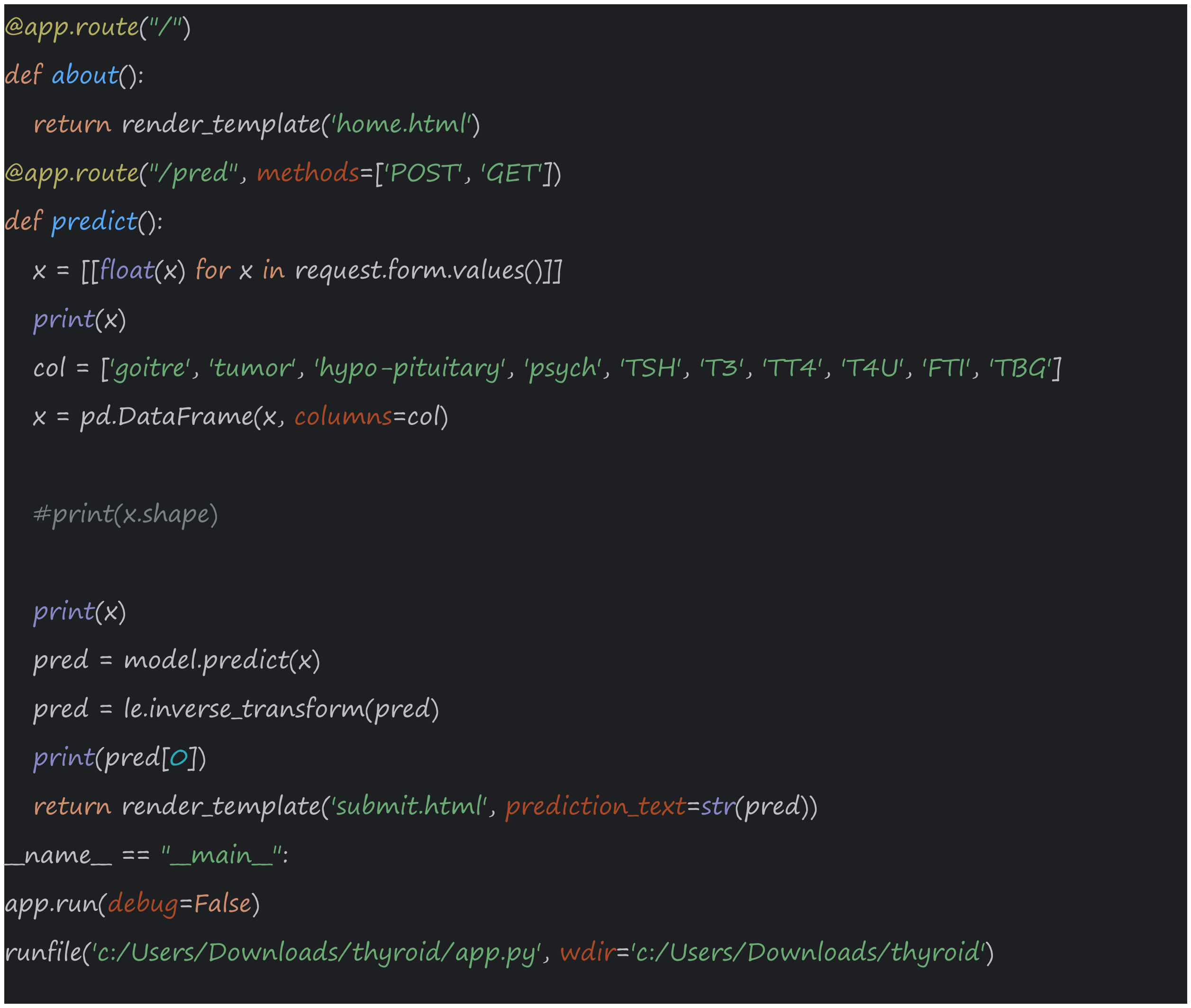
#build web page using html coding

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset ="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Home</title>
<style>
body
{
background-image:url("https://media.istockphoto.com/id/1198780608/photo/human-
thyroid-
anatomy.jpg?s=612x612&w=0&k=20&c=A96BQGdc57yAqHZI3cS2xrHJTPcUBvXsXQqdNSWh
8jA=");
background-size:cover;
}
h3.big
{
line-height:1.8;
}
</style>
</head>


<body>
```

```
<br>
<div class="container">
<div class="row">
<div class="col-md-12 bg-light text-right">
<a href="/home" class="btn btn-info btn-lg">Home</a>
<a href="/predict" class="btn btn-primary disabled btn-lg">Predict</a>
</div>
</div>

<center>
<h1><strong><p style="color: orange">Thyroid disease Classification</p></strong></h1>
<h3>miscellaneous<h3>

</div>



<script src="https://ajax.googleapix.com/ajax/libs/jquer/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</body>
</html>
```

#Build python code

```
from flask import Flask, render_template, request
import numpy as np
import pickle
import pandas as pd
model = pickle.load(open(r"c:\Users\downloads\thyroid\thyroid_1_model.pkl", 'rb'))
le = pickle.load(open("label_encoder.pkl", 'rb'))
app = Flask(__name__)
```



```
@app.route("/")
def about():
   return render_template('home.html')
@app.route("/pred", methods=['POST', 'GET'])
def predict():
   x = [[float(x) for x in request.form.values()]]
   print(x)
   col = ['goitre', 'tumor', 'hypo-pituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']
   x = pd.DataFrame(x, columns=col)

   #print(x.shape)

   print(x)
   pred = model.predict(x)
   pred = le.inverse_transform(pred)
   print(pred[0])
   return render_template('submit.html', prediction_text=str(pred))
_name_ == "_main_":
app.run(debug=False)
runfile('c:/Users/Downloads/thyroid/app.py', wdir='c:/Users/Downloads/thyroid')
```

-

- **Image classification**: One of the most popular classification problems is image classification: determining what type of object (or scene) is in a digital image. Images can be thought of as a high-dimensional vectors which we would like to classify into different classes such as cat, car, human, and airplane. A multinomial classification model can be trained to classify images into different categories. For example, in order to classify images of dogs and cats for use within machine vision systems, machine learning techniques can help automate this process based on pre-classified images of dogs and cats.rent categories. Deep learning algorithms such as Convolutional Neural Networks (CNN)-based classification models are state-of-the-art in different image classification tasks. Another use case is image segmentation, where the pixels of an image are assigned a label based on what object they belong to. Image segmentation is defined as “the process of distinguishing semantically meaningful image regions on the basis of visual features”. The picture below represents how the CNN algorithm can be used to build a classification model that classifies images such as Cat and dog.