

Thyroid Disease Classification Using ML

Greetings From _NM2023TNID32126,

	Name of the student	NM ID
TEAM LEADER	T.Kavitha	BDEFC696FD4B053A2ED4D3EE9CCE8F09
TEAM MEMBER	T.Esakkiammal	71E744DF5D1710990FA49F6761C80541
	P.Kavipakkiya	C6F3B9B0E41E8198EB9402585E3B0F65
	C.Narmatha	7E1700B9EEE6342BF6A4F13B11EE47D9

**GOVERNMENT ARTS AND SCIENCE COLLEGE,
KADAYANALLUR**

Index

S.No	Content	Page No
1	Introduction	3
2	Problem Definition&Design Thinking	6
3	Result	9
4	Advantages&Disadvantages	13
5	Applications	15
6	Conclusion	16
7	Future Scope	17
8	Appendix	18

1.Introduction

1.1Overview:

Thyroid disease classification using machine learning involves using algorithms and models to analyze data related to the thyroid gland and to classify patients into different categories based on their thyroid function. The aim of this classification is to aid in diagnosis and treatment of thyroid disorders.

Machine learning techniques can be applied to a wide range of data sources including patient medical records, lab results, and imaging studies. These algorithms can be trained using labeled data that have been previously diagnosed by physicians or experts. Various supervised and unsupervised machine learning techniques such as decision trees, support vector machines, neural networks, and clustering algorithms can be used for this classification.

The classification process can involve identifying patients with normal thyroid function, hyperthyroidism, hypothyroidism, thyroid nodules, or thyroid cancer. Machine learning models can help to identify patterns and features in the data that are most relevant to the classification task, and can provide accurate predictions for new patients based on these features.

Overall, machine learning can be a powerful tool for thyroid disease classification, providing clinicians with a more accurate and efficient way to diagnose and manage thyroid disorders.

1.2Purpose:

Thyroid disease classification using machine learning has several potential applications in clinical practice. Some of the ways in which this technology can be used include:

- **Early detection and diagnosis:** Machine learning models can help identify patients who are at risk of developing thyroid disease or who have early-stage thyroid disease. This can lead to earlier diagnosis and treatment, which can improve patient outcomes.
- **Personalized Treatment:** Different thyroid disorders require different treatments, and

machine learning models can help identify the most appropriate treatment for each patient based on their specific thyroid function and other factors.

- **Improved accuracy and efficiency:** Machine learning can help improve the accuracy and efficiency of thyroid disease diagnosis, reducing the need for invasive procedures such as biopsies and reducing the risk of misdiagnosis.
- **Resource allocation:** By accurately identifying patients with thyroid disease, machine learning can help healthcare providers allocate resources more efficiently, ensuring that patients receive the appropriate level of care.
- **Research:** Machine learning can also be used to analyze large datasets related to thyroid disease, helping researchers to identify new patterns and insights that can improve our understanding of thyroid function and disease.

Achieved

Thyroid disease classification using machine learning can help achieve several goals, including:

- **More accurate diagnosis:** Machine learning models can help clinicians accurately identify patients with thyroid disorders, which can lead to earlier diagnosis and appropriate treatment.
- **Improved patient outcomes:** By providing more accurate and timely diagnoses, machine learning can help improve patient outcomes, reducing the risk of complications and improving overall quality of life.
- **Personalized treatment:** Machine learning can help identify the most appropriate treatment for each patient, based on their specific thyroid function and other factors. This can improve treatment outcomes and reduce the risk of side effects.
- **More efficient resource allocation:** By accurately identifying patients with thyroid disease, machine learning can help healthcare providers allocate resources more efficiently,

ensuring that patients receive the appropriate level of care.

- **Reduced healthcare costs:** Early diagnosis and appropriate treatment can help reduce healthcare costs associated with thyroid disease, such as hospitalizations, surgeries, and other procedures.
- **Advancements in research:** Machine learning can also be used to analyze large datasets related to thyroid disease, helping researchers to identify new patterns and insights that can improve our understanding of thyroid function and disease.
- Overall, thyroid disease classification using machine learning has the potential to improve patient care, reduce healthcare costs, and advance our understanding of thyroid disease.

2.Problem Definition&Design Thinking

Thyroid disease classification using machine learning (ML) involves training a model to classify different types of thyroid disorders based on certain features or characteristics. These features could include patient demographics, medical history, thyroid function test results, imaging studies, and other clinical data.

The classification of thyroid disease can be done into various types such as hypothyroidism, hyperthyroidism, thyroid nodules, goiter, thyroiditis, thyroid cancer, etc.

ML algorithms like decision trees, support vector machines, logistic regression, and neural networks can be used to classify thyroid disease accurately. These algorithms use statistical models and patterns to learn from the data and classify the thyroid disease.

The classification accuracy of the ML model depends on the quality and quantity of the data used for training. Hence, it is essential to collect and preprocess the data before training the model. Additionally, it is important to validate the model using a separate dataset to ensure that the model performs well on unseen data.

Overall, thyroid disease classification using ML can assist clinicians in making faster and more accurate diagnoses and treatment decisions, potentially leading to better patient outcomes.

2.1 Empathy Map:

Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

[Share template feedback](#)

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 service might they require?

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

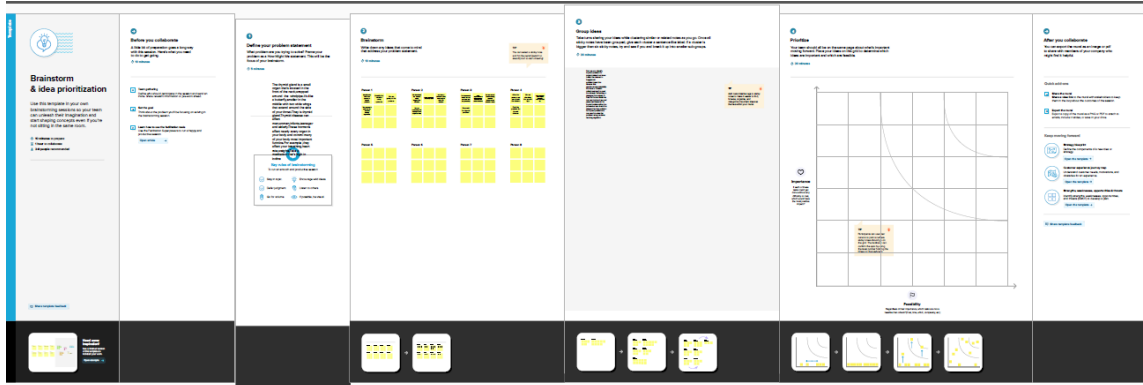
Does
What behavior have we observed?
What actions might they take?

Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Give them a name and a personality associated with your persona.

Need some inspiration?
See a filtered version of the template to kickstart your work.
[Open example](#)

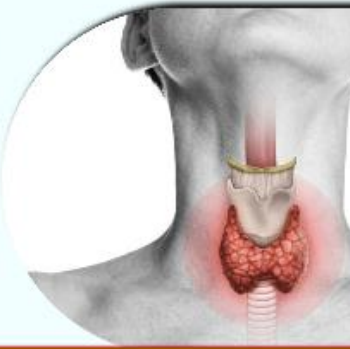
2.2 Ideation & Brainstorming Map:



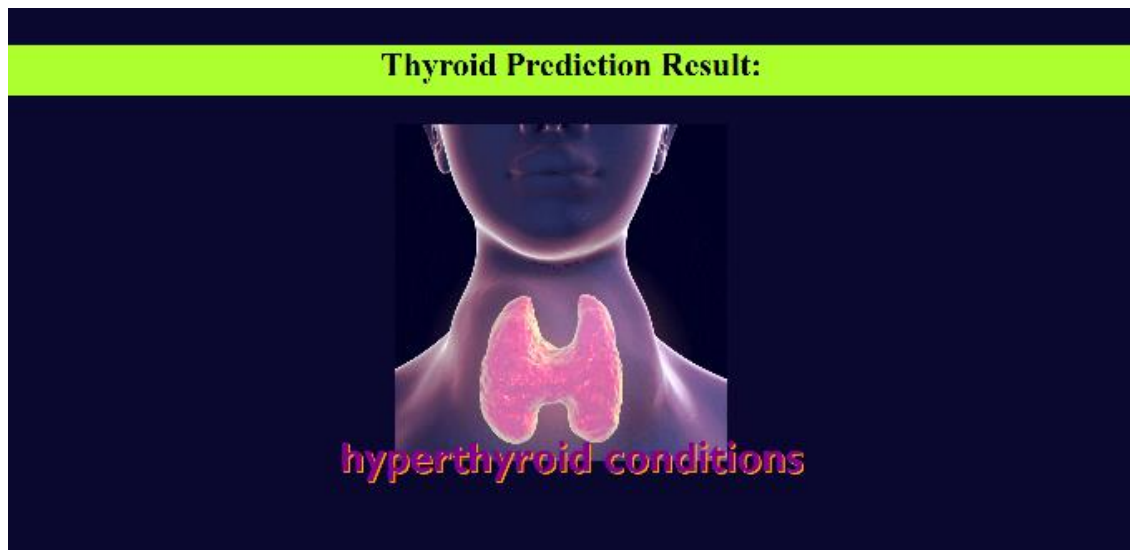
3.Result:

Welcome to Thyroid Disease Prediction

Goitre	<input type="text"/>
Tumor	<input type="text"/>
Hypopituitary	<input type="text"/>
Psych	<input type="text"/>
TSH	<input type="text"/>
T3	<input type="text"/>
TT4	<input type="text"/>
T4U	<input type="text"/>
FTI	<input type="text"/>
TBG	<input type="text"/>




Predict



Welcome to Thyroid Disease Prediction

Goitre	<input type="text"/>
Tumor	<input type="text"/>
Hypopituitary	<input type="text"/>
Psych	<input type="text"/>
TSH	<input type="text"/>
T3	<input type="text"/>
TT4	<input type="text"/>
T4U	<input type="text"/>
FTI	<input type="text"/>
TBG	<input type="text"/>



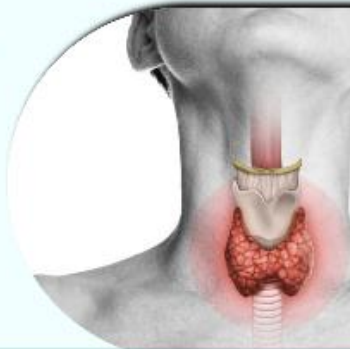
Predict

Thyroid Prediction Result:



Welcome to Thyroid Disease Prediction

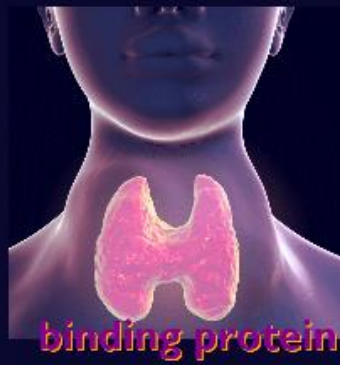
Goitre
Tumor
Hypopituitary
Psych
TSH
T3
TT4
T4U
FTI
TDB



0	<input type="text"/>
2	<input type="text"/>
40	<input type="text"/>
5	<input type="text"/>
4	<input type="text"/>
23	<input type="text"/>
34	<input type="text"/>
57	<input type="text"/>
30	<input type="text"/>
54	<input type="text"/>

Predict

Thyroid Prediction Result:



Welcome to Thyroid Disease Prediction

Goitre	<input type="text"/>
Tumor	<input type="text"/>
Hypopituitary	<input type="text"/>
Psych	<input type="text"/>
TSII	<input type="text"/>
T3	<input type="text"/>
T4	<input type="text"/>
T4U	<input type="text"/>
FTI	<input type="text"/>
TBG	<input type="text"/>

Predict

4.Advantages & Disadvantages

Advantages of using machine learning for thyroid disease classification include:

- **Improved accuracy:** Machine learning algorithms can analyze large amounts of data and identify patterns that may not be evident to human clinicians, leading to more accurate diagnoses and treatment recommendations.
- **Early detection:** Machine learning can help identify patients with early-stage thyroid disease, leading to earlier treatment and better outcomes.
- **Personalized treatment:** Machine learning can help identify the most appropriate treatment for each patient based on their specific thyroid function and other factors, leading to more effective treatment and fewer side effects.
- **Efficient use of resources:** By accurately identifying patients with thyroid disease, machine learning can help healthcare providers allocate resources more efficiently, ensuring that patients receive the appropriate level of care.
- **Advancements in research:** Machine learning can be used to analyze large datasets related to thyroid disease, helping researchers to identify new patterns and insights that can improve our understanding of thyroid function and disease.

- **Disadvantages of using machine learning for thyroid disease classification include:**

- **Reliance on data quality:** Machine learning algorithms rely on high-quality data for accurate analysis and diagnosis. Poor-quality data can lead to inaccurate results and misdiagnosis.
- **Need for training data:** Machine learning models require large amounts of labeled training data to be accurate, which can be time-consuming and expensive to collect.

- **Lack of transparency:** Machine learning algorithms can be difficult to interpret, which can make it challenging for clinicians to understand how a diagnosis was reached.
- **Limited generalizability:** Machine learning models may not be generalizable to other populations or settings, which can limit their usefulness in clinical practice.
- **Potential for bias:** Machine learning algorithms can perpetuate existing biases in the data, leading to inaccurate or unfair diagnoses for certain groups of patients.
- Overall, while machine learning has the potential to improve thyroid disease classification, careful attention must be paid to data quality, training data, transparency, generalizability, and potential bias.

5.Application:

Thyroid disease classification using machine learning can be applied in various areas, including:

- **Clinical practice:** Machine learning models can be used by clinicians to aid in the diagnosis and treatment of thyroid disease. By analyzing patient data, machine learning algorithms can provide accurate diagnoses and treatment recommendations, helping to improve patient outcomes.
- **Medical research:** Machine learning can be used to analyze large datasets related to thyroid disease, helping researchers to identify new patterns and insights that can improve our understanding of thyroid function and disease.
- **Healthcare management:** Machine learning can help healthcare providers manage resources more efficiently by accurately identifying patients with thyroid disease and ensuring they receive the appropriate level of care.
- **Public health:** Machine learning can help public health officials monitor trends in thyroid disease incidence and prevalence, identify at-risk populations, and develop targeted interventions.
- **Telemedicine:** Machine learning can be used to develop telemedicine tools that allow patients to receive remote diagnosis and treatment for thyroid disease. This can be particularly useful for patients in rural or underserved areas.
- Overall, thyroid disease classification using machine learning has broad applications in healthcare, medical research, and public health. By improving our ability to diagnose and treat thyroid disease, machine learning has the potential to improve patient outcomes, reduce healthcare costs, and advance our understanding of thyroid function and disease.

6.Conclusion:

In conclusion, thyroid disease classification using machine learning has the potential to significantly improve patient care and advance our understanding of thyroid function and disease. By analyzing patient data, machine learning algorithms can provide accurate diagnoses, identify at-risk populations, and recommend personalized treatment options. This can lead to improved patient outcomes, reduced healthcare costs, and more efficient resource allocation. However, the use of machine learning in thyroid disease classification also has potential drawbacks, such as the reliance on high-quality data, the need for large amounts of training data, and the potential for bias. Despite these challenges, thyroid disease classification using machine learning represents a promising area of research and development in healthcare, with broad applications in clinical practice, medical research, healthcare management, public health, and telemedicine.

7.Future Scope

- There are several potential enhancements that can be made in the future for thyroid disease classification using machine learning, including:
- **Integration with electronic health records (EHRs):** Machine learning algorithms can be integrated with EHRs to facilitate the collection and analysis of patient data, improving the accuracy and efficiency of thyroid disease classification.
- **Use of multiple data sources:** Machine learning models can be trained on multiple data sources, such as genomic data, imaging data, and patient-reported outcomes, to improve the accuracy and generalizability of thyroid disease classification.
- **Development of explainable AI:** Explainable AI techniques can be developed to enhance the interpretability of machine learning algorithms, allowing clinicians to better understand how a diagnosis was reached.
- **Improved data quality and bias mitigation:** Efforts can be made to improve the quality of data used in machine learning models, and to develop methods to mitigate potential biases in the data.
- **Development of novel biomarkers:** Machine learning can be used to identify novel biomarkers for thyroid disease, which can be used to improve diagnosis and treatment.
- **Integration with telemedicine:** Machine learning can be integrated with telemedicine tools to provide remote diagnosis and treatment for thyroid disease, improving access to care for patients in underserved areas.
- Overall, there are many potential enhancements that can be made in the future for thyroid disease classification using machine learning, and continued research and development in this area has the potential to significantly improve patient care and outcomes.

8.Appendix

A.Source Code:

The all.py:

```
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

from sklearn.model_selection import train_test_split

from imblearn.over_sampling import SMOTE

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification_report

from sklearn.model_selection import RandomizedSearchCV

import warnings

warnings.filterwarnings("ignore", category=UserWarning)

data = pd.read_csv("C:/Users/ELCOT/Desktop/PROJECT for ES/thyroidDF.csv")

print(data.head())

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

```

# Drop redundant attributes and modify the original dataframe

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

#re-mapping target values to diagnostic group

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

               'B': 'hyperthyroid conditions',

               'C': 'hyperthyroid conditions',

               'D': 'hyperthyroid conditions',

               'E': 'hypothyroid conditions',

               'F': 'hypothyroid conditions',

               'G': 'hypothyroid conditions',

               'H': 'hypothyroid conditions',

               '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)

print(data['target'].value_counts())

print(data[data.age>100])

#splitting the data

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

y= data.iloc[:, -1]

print(x)

print(x['sex'].unique())

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

print(x['sex'].value_counts())

#converting the data

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')

print(x.info())

from sklearn.preprocessing import OrdinalEncoder, LabelEncoder

import numpy as np

# create an OrdinalEncoder object

ordinal_encoder = OrdinalEncoder(dtype='int64')

# apply ordinal encoding to the categorical features in x

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

# replace the nan values with 0

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

# print the transformed x dataframe

print(x)

# Replace missing values in x with 0

x.fillna(0, inplace=True)

# Use OrdinalEncoder to handle categorical values in x

ordinal_encoder = OrdinalEncoder(dtype='int64')

```

```

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

# Use LabelEncoder to encode y values

label_encoder = LabelEncoder()

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

# Split the data into train and test sets

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

# Scale the data

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)

# Handling Imbalanced data

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)

print(y_train.value_counts())

print(x_bal)

# Convert arrays to dataframes

columns=['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','t
hyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid','lithium','goitre','tumor
','hypopituitary','psych','TSH','T3','TT4','T4U','FTI','TBG']

```

```

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

y_train_bal = pd.DataFrame(y_bal, columns=['target'])

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

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

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

#Random Forest Classifier Model

from sklearn.ensemble import RandomForestClassifier

rfr = RandomForestClassifier().fit(x_bal,y_bal.values.ravel())

y_pred = rfr.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

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_antithyroid_meds',
'sick', 'pregnant', 'thyroid_surgery','I131_treatment', 'query_hypothyroid', 'query_hyperthyroid',
'lithium', 'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']

importance = results.importances_mean

importance = np.sort(importance)

for i,v in enumerate(importance):

    i=feature_importance[i]

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

```

```

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

plt.bar(feature_importance, importance.astype(float))

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

plt.show()

print(x.head())

x_bal =
x_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','pregnant','
thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid','lithium'], axis=1)

x_test_bal =
x_test_bal.drop(['age','sex','on_thyroxine','query_on_thyroxine','on_antithyroid_meds','sick','preg
nant','thyroid_surgery','I131_treatment','query_hypothyroid','query_hyperthyroid','lithium'],
axis=1)

print(x_bal.head())

print(data.info())

import seaborn as sns

corrmatrix = x.corr()

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

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

plt.show()

# select features for model

cols = ['goitre','tumor','hypopituitary','psych','TSH','T3','TT4','T4U','FTI','TBG']

```



```

# create balanced training set

x_bal = x_train_bal[cols]

y_bal = y_train_bal

# create balanced test set

x_test_bal = x_test_bal[cols]

y_test_bal = y_test_bal

#Random Forest Classifier Model

from sklearn.ensemble import RandomForestClassifier

rfr1 = RandomForestClassifier().fit(x_bal, y_bal.values.ravel())

y_pred = rfr1.predict(x_test_bal)

print(classification_report(y_test_bal, y_pred))

from xgboost import XGBClassifier

xgb1 = XGBClassifier()

xgb1.fit(x_bal, y_bal)

y_pred = xgb1.predict(x_test_bal)

print(classification_report(y_test_bal, y_pred))

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report

sv = SVC()

```

```

sv.fit(x_bal,y_bal)

y_pred = sv.predict(x_test_bal)

print(classification_report(y_test_bal,y_pred))

#ANN Model

model = Sequential()

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

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

model.add(Dropout(0.2))

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

model.add(Dropout(0.2))

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

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

model.summary()

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

model.fit(x_bal,y_bal, validation_data=[x_test_bal, y_test_bal], epochs=0)

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

model.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]])

print(y_pred)

```

```

from sklearn.svm import SVC

from sklearn.model_selection import RandomizedSearchCV

import warnings

params = {

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

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

    'kernel': ['rbf']

}

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

with warnings.catch_warnings():

    warnings.filterwarnings("ignore", category=FutureWarning)

    random_svc.fit(x_bal, y_bal)

print(random_svc.best_params_)

# Output: {'kernel': 'rbf', 'gamma': 0.1, 'C': 100}

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

sv1.fit(x_bal, y_bal)

y_pred = sv1.predict(x_test_bal)

#saving the model

import pickle

```

```
pickle.dump(xgb1, 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'))

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

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

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

app.py:

```
from flask import Flask, render_template, request

import numpy as np

import pickle

import pandas as pd


app = Flask(__name__)

model = pickle.load(open('thyroid_1_model.pkl', 'rb'))

le = pickle.load(open("label_encoder.pkl", 'rb'))


@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', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TBG']

    x = pd.DataFrame(x, columns=col)

    print(x)

    pred = model.predict(x)

    pred = le.inverse_transform(pred)

    print(pred[0])

    if pred == 0:

        return render_template('submit.html', prediction_text=pred[0])

    else:

        return render_template('submit.html', prediction_text=pred[0])

```

```
if __name__ == "__main__":
```

```
    app.run(debug=False)
```