

Introduction

1.1 Overview

The Thyroid gland is a vascular gland and one of the most important organs of the human body. This gland secretes two hormones which help in controlling the metabolism of the body. The two types of Thyroid disorders are Hyperthyroidism and Hypothyroidism. When this disorder occurs in the body, they release certain types of hormones into the body which imbalances the body's metabolism. A thyroid-related Blood test is used to detect this disease but it is often blurred and noise will be present. Data cleansing methods were used to make the data primitive enough for the analytics to show the risk of patients getting this disease. Machine Learning plays a very deciding role in disease prediction. Machine Learning algorithms, SVM , NB, Decision Tree, 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.

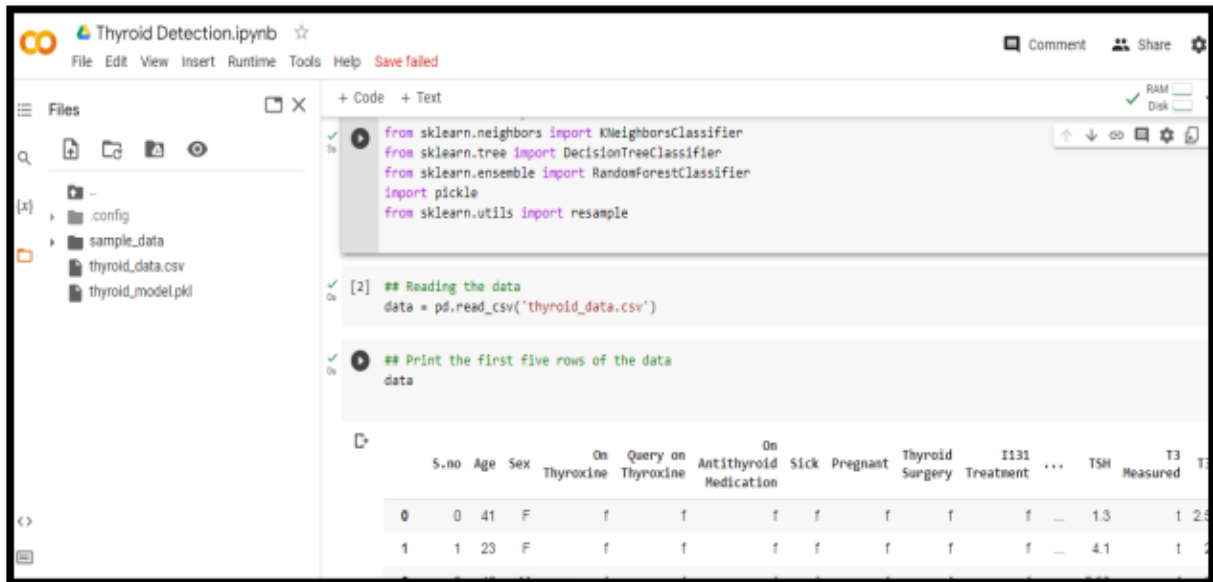
1.2 Purpose

The purpose of this project is to early prediction of Thyroid

2. Problem Definition & Design Thinking



3. Results

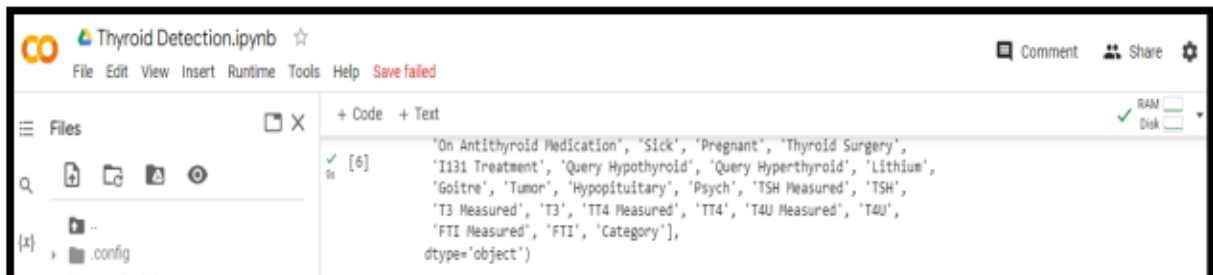


```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import pickle
from sklearn.utils import resample
```

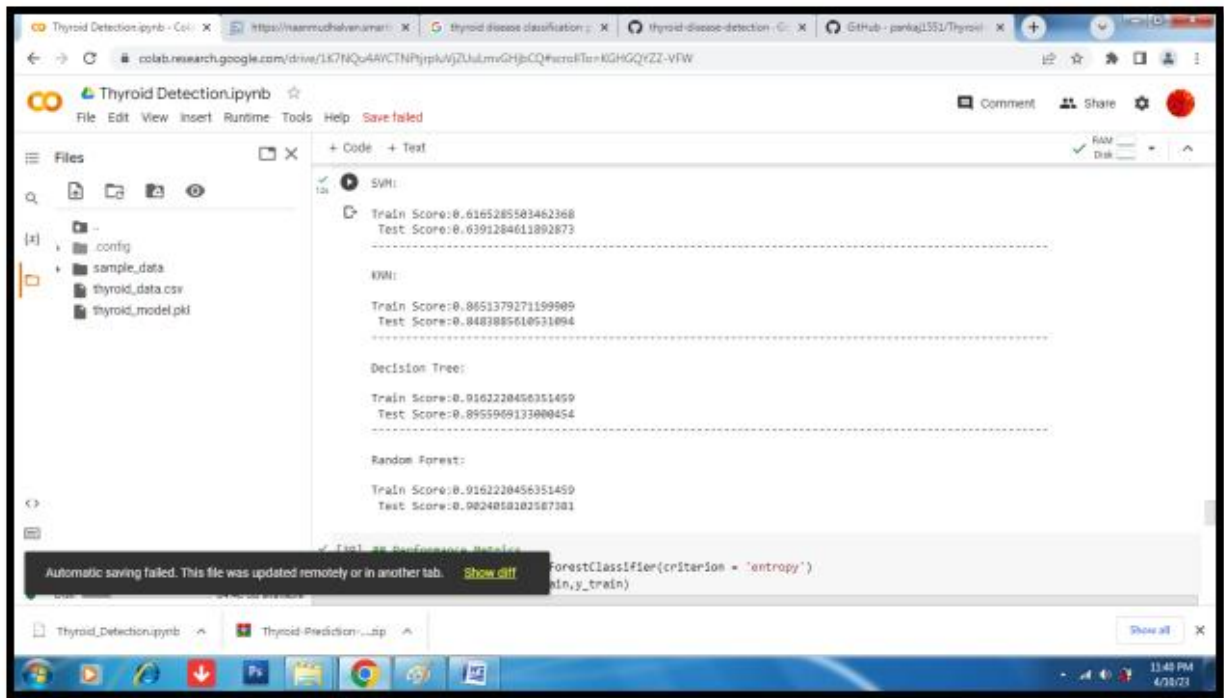
```
## Reading the data
data = pd.read_csv('thyroid_data.csv')
```

```
## Print the first five rows of the data
data
```

S.no	Age	Sex	On Thyroxine	Query on Thyroxine	On Antithyroid Medication	Sick	Pregnant	Thyroid Surgery	I131 Treatment	TSH Measured	T3 Measured
0	0	41	F	f	f	f	f	f	f	1.3	1.25
1	1	23	F	f	f	f	f	f	f	4.1	1.25



```
'On Antithyroid Medication', 'Sick', 'Pregnant', 'Thyroid Surgery',  
'I131 Treatment', 'Query Hypothyroid', 'Query Hyperthyroid', 'Lithium',  
'Goitre', 'Tumor', 'Hypopituitary', 'Psych', 'TSH Measured', 'TSH',  
'T3 Measured', 'T3', 'TT4 Measured', 'TT4', 'T4U Measured', 'T4U',  
'FTI Measured', 'FTI', 'Category'],  
dtype='object')
```



4. Advantages and Disadvantages

- Produce high accuracy
- Easy to use
- Early Detection

5. Applications

It can be applied in medical field.

6. Conclusion

Thyroid disease was classified using algorithm several algorithms SVM, NB, Decision Tree. The Decision Tree Algorithm Model are Produced high accuracy.

7. Future Scope

For this Model will develop Mobile Applications

8. Appendix

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
# %matplotlib inline
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import pickle
from sklearn.utils import resample
```

```
## Reading the data
data = pd.read_csv('thyroid_data.csv')
```

```
## Print the first five rows of the data
data
```

```
## Shape of the data
data.shape
```

```
n = len(data[data['Category'] == 'hyperthyroid'])
print("No of hyperthyroid in Dataset:",n)
```

```
n1 = len(data[data['Category'] == 'hypothyroid'])
print("No of hypothyroid in Dataset:",n1)
```

```
n2 = len(data[data['Category'] == 'sick'])
print("No of sick in Dataset:",n2)
```

```
n3 = len(data[data['Category'] == 'negative'])
```

```
print("No of negative in Dataset:",n3)
```

```
"""##### The data has 3772 rows and 30 columns."""
```

```
## Columns
```

```
data.columns
```

```
## Checking the missing values
```

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

```
"""##### We can see that there are no missing values. But if we see the dataset the missing values are replaced with the invalid values like '?'. Let's replace such values with 'nan' and check for the missing values again."""
```

```
for column in data.columns:
```

```
    count=data[column][data[column]=='?'].count()
```

```
    if count != 0:
```

```
        print(column,data[column][data[column]=='?'].count())
```

```
## Let's drop some unnecessary columns
```

```
data=data.drop(['S.no','On Thyroxine', 'Query on Thyroxine',
```

```
               'On Antithyroid Medication',
```

```
               'I131 Treatment', 'Query Hypothyroid', 'Query Hyperthyroid', 'Lithium',
```

```
               'TSH Measured','Hypopituitary','Psych',
```

```
               'T3 Measured', 'TT4 Measured', 'T4U Measured',
```

```
               'FTI Measured'],axis=1)
```

```
## Now replace the '?' values with numpy nan
```

```
for column in data.columns:
```

```
    count=data[column][data[column]=='?'].count()
```

```
    if count!=0:
```

```
data[column]=data[column].replace('?',np.nan)
```

```
for column in data.columns:
```

```
    count=data[column][data[column]=='?'].count()
```

```
    if count == 0:
```

```
        print(column,data[column][data[column]=='?'].count())
```

```
"""##### So, we have replaced all such values with 'nan' values."""
```

```
## Unique Values
```

```
for column in data.columns:
```

```
    print(column,(data[column].unique()))
```

```
data.dtypes
```

```
"""### Handling Missing Values"""
```

```
data['Age'].fillna((data['Age'].median()), inplace = True)
```

```
data['TSH'].fillna((data['TSH'].median()), inplace = True)
```

```
data['T3'].fillna((data['T3'].median()), inplace = True)
```

```
data['TT4'].fillna((data['TT4'].median()), inplace = True)
```

```
data['T4U'].fillna((data['T4U'].median()), inplace = True)
```

```
data['FTI'].fillna((data['FTI'].median()), inplace = True)
```

```
"""### Handling nominal categorical variables"""
```

```
## We will perform one hot encoding for nominal categorical variable.
```

```
sex = data[["Sex"]]
```

```
sex = pd.get_dummies(sex, drop_first= True)
```

```
sick = data[["Sick"]]
```

```
sick = pd.get_dummies(sick, drop_first= True)
```

```
pregnant = data[["Pregnant"]]
```

```
pregnant = pd.get_dummies(pregnant, drop_first= True)
```

```
thyroid_surgery = data[["Thyroid Surgery"]]
```

```
thyroid_surgery = pd.get_dummies(thyroid_surgery, drop_first= True)
```

```
goitre = data[["Goitre"]]
```

```
goitre = pd.get_dummies(goitre, drop_first= True)
```

```
tumor = data[["Tumor"]]
```

```
tumor = pd.get_dummies(tumor, drop_first= True)
```

```
data.columns
```

```
final_df = pd.concat([data,sex, sick, pregnant, thyroid_surgery,  
    goitre, tumor,], axis = 1)
```

```
final_df.columns
```

```
final_df.drop(['Sex', 'Sick', 'Pregnant', 'Thyroid Surgery',  
    'Goitre', 'Tumor'], axis = 1, inplace = True)
```

```
## Checking the missing values
```

```
final_df.isnull().sum()
```

```
"""##### Great! Now the data has no missing values."""
```

```
## Let's check the distribution for our continuous data in the dataset.
```

```
columns = ['Age','TSH','T3','TT4','T4U','FTI']
```

```
plt.figure(figsize=(10,15),facecolor='white')
plotnumber = 1
```

```
for column in columns:
```

```
    ax = plt.subplot(3,2,plotnumber)
    sns.distplot(final_df[column])
    plt.xlabel(column,fontsize=10)
    plotnumber+=1
plt.show()
```

```
"""##### The graph TSH heavily skewed towards left. We drop it."""
```

```
final_df = final_df.drop(['TSH'], axis = 1)
```

```
from sklearn.preprocessing import LabelEncoder
```

```
lblEn=LabelEncoder()
```

```
final_df['Category']=lblEn.fit_transform(final_df['Category'])
```

```
sns.countplot(final_df['Category'])
```

```
"""##### We can see that the dataset is highly imbalanced.
```

```
"""
```

```
X=final_df.drop(['Category'],axis=1)
```

```
y=final_df.Category
```

```
# 42 input
```

```
from imblearn.over_sampling import SMOTENC,RandomOverSampler,KMeansSMOTE
```



```
rdsmple=RandomOverSampler()  
X_sampled,y_sampled=rdsmple.fit_resample(X,y)
```

```
X_sampled.shape
```

```
X_sampled=pd.DataFrame(data=X_sampled,columns=X.columns)
```

```
X_sampled
```

```
sns.countplot(y_sampled)
```

```
X_sampled.columns
```

```
"""##### Great! Our dataset is balanced now."""
```

```
from sklearn.model_selection import train_test_split  
X_train,X_test,y_train,y_test=train_test_split(X_sampled,y_sampled,test_size=0.2,random_state=0)
```

```
"""### Fitting data in various models"""
```

```
def svm_classifier(X_train,X_test,y_train,y_test):  
    classifier_svm=SVC(kernel='rbf',random_state=0)  
    classifier_svm.fit(X_train,y_train)  
    y_pred=classifier_svm.predict(X_test)
```

```
cm=confusion_matrix(y_test,y_pred)
return (f'Train Score:{classifier_svm.score(X_train,y_train)}\n Test
Score:{classifier_svm.score(X_test,y_test)}')
```

```
def knn_classifier(X_train,X_test,y_train,y_test):
    classifier_knn=KNeighborsClassifier(metric='minkowski',p=2)
    classifier_knn.fit(X_train,y_train)
    y_pred=classifier_knn.predict(X_test)
    cm=confusion_matrix(y_test,y_pred)
    return (f'Train Score:{classifier_knn.score(X_train,y_train)}\n Test
Score:{classifier_knn.score(X_test,y_test)}')
```

```
def tree_classifier(X_train,X_test,y_train,y_test):
    classifier_tree=DecisionTreeClassifier(criterion='entropy',random_state=0)
    classifier_tree.fit(X_train,y_train)
    y_pred=classifier_tree.predict(X_test)
    cm=confusion_matrix(y_test,y_pred)
    return (f'Train Score:{classifier_tree.score(X_train,y_train)}\n Test
Score:{classifier_tree.score(X_test,y_test)}')
```

```
def forest_classifier(X_train,X_test,y_train,y_test):
    classifier_forest=RandomForestClassifier(criterion='entropy',random_state=0)
    classifier_forest.fit(X_train,y_train)
    y_pred=classifier_forest.predict(X_test)
    cm=confusion_matrix(y_test,y_pred)
    return (f'Train Score:{classifier_forest.score(X_train,y_train)}\n Test
Score:{classifier_forest.score(X_test,y_test)}')
```

```
def print_score(X_train, X_test, y_train, y_test):
    print("SVM:\n")
    result1=svm_classifier(X_train, X_test, y_train, y_test)
```

```
print(result1)
print("-"*100)
print()
```

```
print("KNN:\n")
result2=knn_classifier(X_train, X_test, y_train, y_test)
print(result2)
```

```
print("-"*100)
print()
```

```
print("Decision Tree:\n")
result4=tree_classifier(X_train, X_test, y_train, y_test)
print(result4)
```

```
print("-"*100)
print()
```

```
print("Random Forest:\n")
result5=forest_classifier(X_train, X_test, y_train, y_test)
print(result5)
```

```
print_score(X_train, X_test, y_train, y_test)
```

```
## Performance Metrics
```

```
classifier_forest = RandomForestClassifier(criterion = 'entropy')
classifier_forest.fit(X_train,y_train)
y_pred = classifier_forest.predict(X_test)
cm = confusion_matrix(y_test,y_pred)
cm
```

```

## Classification Report (Accuracy, Precision, Recall and F1 Score)
from sklearn.metrics import roc_auc_score,roc_curve,classification_report

print(classification_report(y_test,y_pred))

"""#### Hyperparameter Tuning"""

from sklearn.model_selection import cross_val_score
accuracies = cross_val_score (estimator = classifier_forest, X=X_train,y=y_train,cv=10)
print(accuracies.mean())

import pickle
filename = 'thyroid_model.pkl'
pickle.dump(classifier_forest,open(filename,'wb'))

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

y_pred = forest.predict(X_test)

confusion_matrix(y_test, y_pred)

X.columns

print(forest.predict([[41,2.5,125,1.14,109,0,0,0,0,0]]))

print(forest.predict([[63,5.5,199,1.05,190,0,0,0,0,0]]))

print(forest.predict([[44,1.4,39,1.16,33,1,0,0,0,0]]))

print(forest.predict([[61,1,96,0.93,109,1,1,0,0,0]]))

```