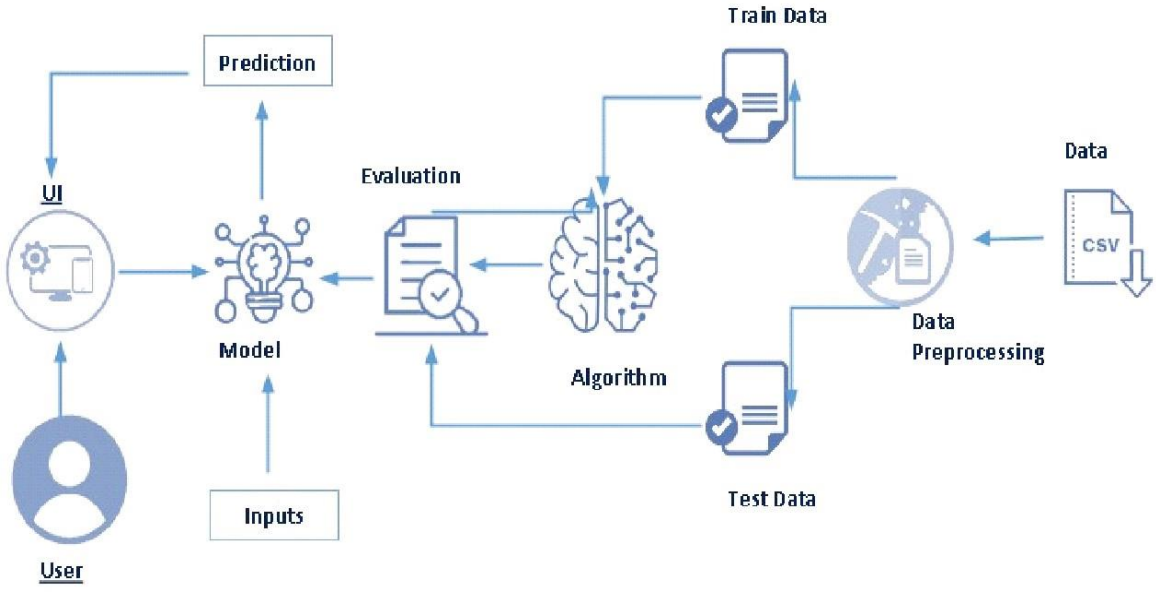
Thyroid Disease Classification Using ML

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 - 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.

Technical Architecture:



Project Flow:

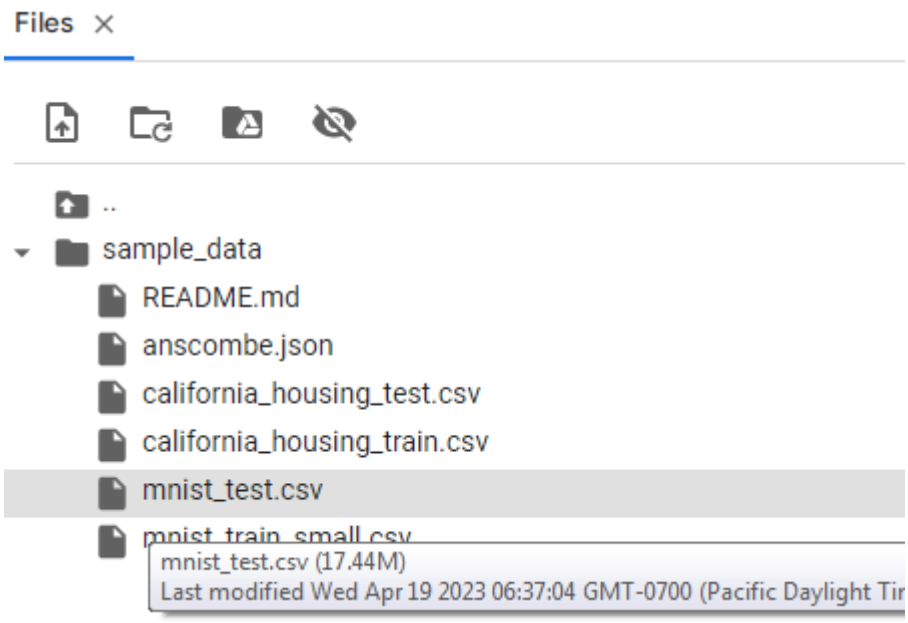
- The user interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- thyroid_1_model.pkl is our saved model. Further, we will use this model for flask integration.
- Training folder contains model training files and the training_ibm folder contains IBM deployment files.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer to Project Description

Activity 2: Business requirements

The business requirements for a machine learning model to predict thyroid disease include the ability to accurately predict thyroid disease based on the scan results, Minimise the number of false positives (wrong thyroid disease confirmations) and false negatives (thyroid is there but got as not thyroid disease). Provide an explanation for the model's decision, to comply with regulations and improve transparency.

Activity 3: Literature Survey (Student Will Write)

The thyroid gland is one of the body's most visible endocrine glands. Its size is determined by the individual's age, gender, and physiological states, such as pregnancy or lactation. It is divided into two lobes (right and left) by an isthmus (a band of tissue). It is imperceptible in everyday life yet can be detected when swallowing. The thyroid hormones T4 and T3 are needed for normal thyroid function. These hormones have a direct effect on the body's metabolic rate. It contributes to the stimulation of glucose, fatty acid, and other molecule consumption. Additionally, it enhances oxygen consumption in the majority of the body's cells by assisting in the processing of uncoupling proteins, which contributes to an improvement in the rate of cellular respiration. Thyroid conditions are difficult to detect in test results, and only trained professionals can do so. However, reading such extensive reports and predicting future results is difficult. Assume a machine learning model can detect the thyroid disease in a patient. The thyroid disease can then be easily identified based on the symptoms in the patient's history. Currently, models are evaluated using accuracy metrics on a validation dataset that is accessible.

Activity 4: Social or Business Impact.

Social Impact:- Untreated/undetected thyroid disease is more dangerous at times it can lead to fatal of the person. So, we can detect it at the earliest then people can get treatment and get cured.

Business Model/Impact:- We can make this application public, offer services as a subscription based or can collaborate with healthcare centres or specialists.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project, we have used drug200.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/prathamtripathi/drug-classification

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report,f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pickle
import warnings
warnings.filterwarnings('ignore')
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas, we have a function called read_csv() to read the dataset. As a parameter, we have to give the directory of the csv file.

	age	sex	on thyroxine	query on thyroxine	on antithyroid medication	sick	pregnant	thyroid surgery	I131 treatment	query hypothyroid	...	TT4 measured	TT4	T4U measured	T4U	FTI measured	FTI	TBG measured	TBG	r
0	41	F	f	f	f	f	f	f	f	f	...	t	125	t	1.14	t	109	f	?	
1	23	F	f	f	f	f	f	f	f	f	...	t	102	f	?	f	?	f	?	
2	46	M	f	f	f	f	f	f	f	f	...	t	109	t	0.91	t	120	f	?	
3	70	F	t	f	f	f	f	f	f	f	...	t	175	f	?	f	?	f	?	
4	70	F	f	f	f	f	f	f	f	f	...	t	61	t	0.87	t	70	f	?	

rows × 30 columns

Activity 2: Data Pre-processing

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Descriptive analysis
- Splitting the dataset as x and y
- Handling Categorical Values
- Checking Correlation
- Converting Data Type
- Splitting dataset into training and test set
- Handled Imbalanced Data
- Applying StandardScaler

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Checking for null values

- For checking the null values, data.isnull() function is used. To sum those null values we use the .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3772 entries, 0 to 3771
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              3771 non-null   object
 1   sex              3622 non-null   object
 2   sick             3772 non-null   object
 3   pregnant         3772 non-null   object
 4   thyroid surgery  3772 non-null   object
 5   I131 treatment   3772 non-null   object
 6   lithium          3772 non-null   object
 7   goitre           3772 non-null   object
 8   tumor            3772 non-null   object
 9   TSH              3403 non-null   object
 10  T3               3003 non-null   object
 11  TT4              3541 non-null   object
 12  T4U              3385 non-null   object
 13  FTI              3387 non-null   object
 14  Label            3772 non-null   object
dtypes: object(15)
memory usage: 442.2+ KB
```

Removing the Redundant attributes from the dataset

Removing the Redundant attributes from the dataset.

```
thyroid_df.drop(['T3 measured','TSH measured','TT4 measured','T4U measured','FTI measured','TBG measured','TBG','referral source','on thyroxine','q
```

- Re-mapping the 'target' values to the diagnostic Group

```
#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) #remapping
```

Dropping Null Values

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

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

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

```
cols = ['age','FTI','TSH','T3','TT4','T4U']
for i in cols:
    thyroid_df[i] = pd.to_numeric(thyroid_df[i])
```

```
thyroid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3772 entries, 0 to 3771
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   age              3771 non-null   float64
 1   sex              3622 non-null   object 
 2   sick             3772 non-null   object 
 3   pregnant         3772 non-null   object 
 4   thyroid surgery  3772 non-null   object 
 5   I131 treatment   3772 non-null   object 
 6   lithium          3772 non-null   object 
 7   goitre           3772 non-null   object 
 8   tumor            3772 non-null   object 
 9   TSH              3403 non-null   float64
 10  T3               3003 non-null   float64
 11  TT4              3541 non-null   float64
 12  T4U              3385 non-null   float64
 13  FTI              3387 non-null   float64
 14  Label            3772 non-null   object 
dtypes: float64(6), object(9)
memory usage: 442.2+ KB
```

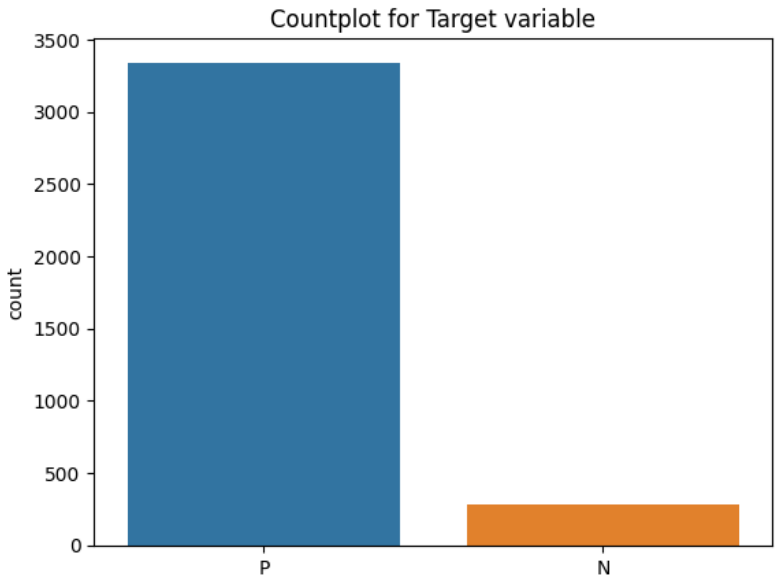
```
thyroid_df.isnull().sum()
```

```
age                0
sex                0
sick               0
pregnant           0
thyroid surgery    0
I131 treatment     0
lithium            0
goitre             0
tumor              0
TSH                0
T3                 0
TT4                0
T4U                0
FTI                0
Label              0
dtype: int64
```

```
thyroid_df = thyroid_df.drop(1364)
```

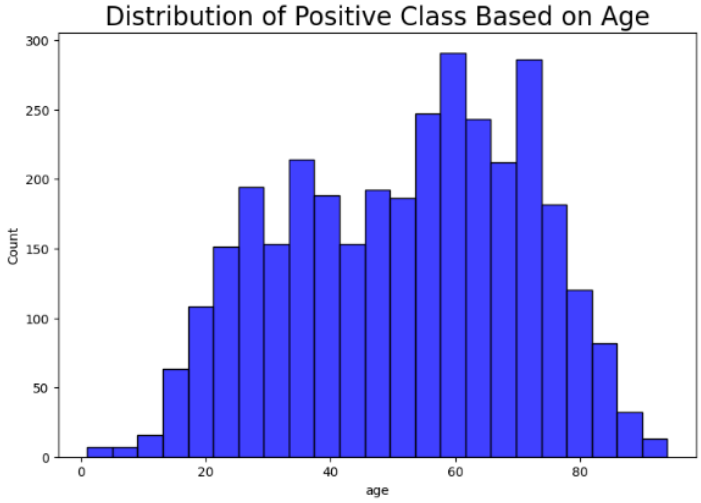
```
thyroid_df.TT4 = thyroid_df.TT4.astype(int)
thyroid_df.FTI = thyroid_df.FTI.astype(int)
thyroid_df.age = thyroid_df.age.astype(int)
```

```
sns.countplot(x='Label',data=thyroid_df)
plt.title("Countplot for Target variable");
```

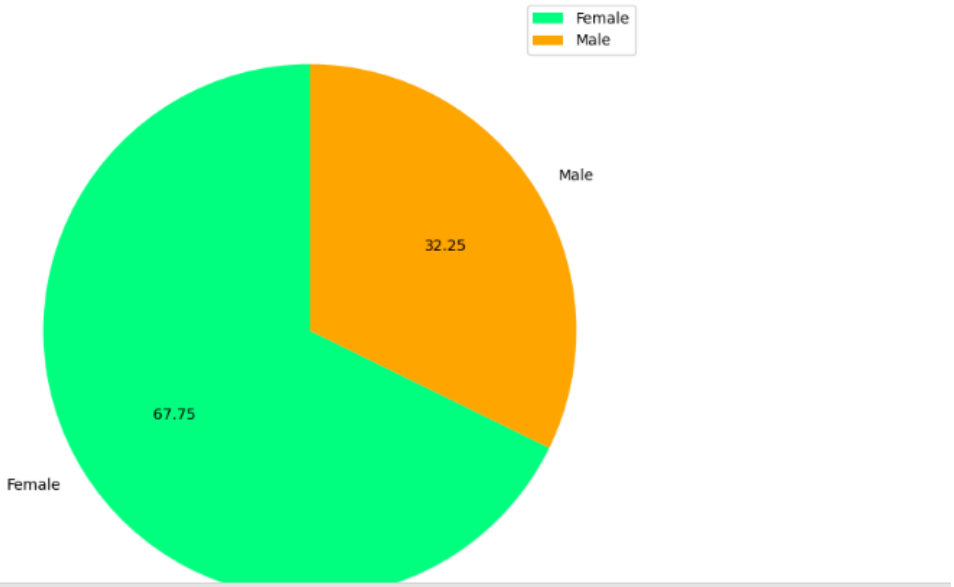



```
[ ] positive_df = thyroid_df[thyroid_df.Label=='P']
```

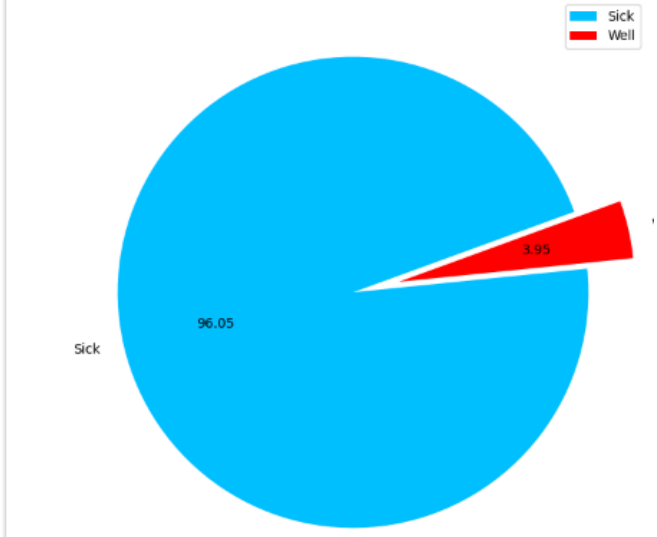
```
plt.figure(figsize=(9,6))
sns.histplot(x='age',data=positive_df,color='blue')
plt.title("Distribution of Positive Class Based on Age",{'fontsize':20});
```



```
[ ]  plt.figure(figsize=(10,8))
     plt.pie(x=positive_df.sex.value_counts(),
             labels=['Female','Male'],
             startangle = 90,
             colors=['springgreen','orange'],
             autopct='%.2f'
            );
     plt.legend();
```




```
plt.figure(figsize=(8,8))
plt.pie(x=positive_df.sick.value_counts(),
        labels=['Sick','Well'],
        startangle = 20,
        colors=['deepskyblue','red'],
        autopct='%.2f',
        explode=[0,0.2]
       );
plt.legend();
```



```
[ ] X = thyroid_df.drop('Label',axis=1)
    y = thyroid_df.Label
```

```
[ ] s_encoder = LabelEncoder()
    si_encoder = LabelEncoder()
    preg_encoder = LabelEncoder()
    th_encoder = LabelEncoder()
    treat_encoder = LabelEncoder()
    lith_encoder = LabelEncoder()
    g_encoder= LabelEncoder()
    tu_encoder = LabelEncoder()
```

```
[ ] X['sex'] = s_encoder.fit_transform(X.sex)
    X['I131 treatment'] = treat_encoder.fit_transform(X['I131 treatment'])
    X['sick'] = si_encoder.fit_transform(X.sick)
    X['pregnant'] = preg_encoder.fit_transform(X.pregnant)
    X['thyroid surgery'] = th_encoder.fit_transform(X['thyroid surgery'])
    X['lithium'] = lith_encoder.fit_transform(X['lithium'])
    X['goitre'] = g_encoder.fit_transform(X['goitre'])
    X['tumor'] = tu_encoder.fit_transform(X['tumor'])
```

```
[ ] # output1 = open('sex_lbl.pkl', 'wb')
    # pickle.dump(s_encoder, output1)
    # output1.close()
    # output2 = open('surgery.pkl', 'wb')
    # pickle.dump(th_encoder, output2)
    # output2.close()
```

```
[ ] def func(df):
        if df == 'P':
            return 1
        else:
            return 0
```

```
[ ] y = y.apply(func)
```

```
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=11)
```

```
    smote = SMOTE(random_state=11)

    x_smote, y_smote = smote.fit_resample(X_train, y_train)
```

```
smote = SMOTE(random_state=11)

x_smote, y_smote = smote.fit_resample(X_train, y_train)
```

```
print("Shape before the Oversampling : ",X_train.shape)
print("Shape after the Oversampling : ",x_smote.shape)
```

```
Shape before the Oversampling :  (2896, 14)
Shape after the Oversampling :  (5340, 14)
```

```
scalr = MinMaxScaler()
x_smote.TT4 = scalr.fit_transform(x_smote[['TT4']])
x_smote.age = scalr.fit_transform(x_smote[['age']])
x_smote.FTI = scalr.fit_transform(x_smote[['FTI']])
```

```
# with open('scalar.pkl','wb') as f:
#     pickle.dump(scalr,f)
```

```
X_test.TT4 = scalr.transform(X_test[['TT4']])
X_test.age = scalr.transform(X_test[['age']])
X_test.FTI = scalr.transform(X_test[['FTI']])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-63-49a241e93e4c> in <cell line: 1>()
----> 1 X_test.TT4 = scalr.transform(X_test[['TT4']])
      2 X_test.age = scalr.transform(X_test[['age']])
      3 X_test.FTI = scalr.transform(X_test[['FTI']])

3 frames
/usr/local/lib/python3.9/dist-packages/sklearn/base.py in _check_feature_names(self, X, reset)
    479                 )
    480
--> 481             raise ValueError(message)
    482
    483     def _validate_data(

ValueError: The feature names should match those that were passed during fit.
Feature names unseen at fit time:
- TT4
Feature names seen at fit time, yet now missing:
- FTI
```

SEARCH STACK OVERFLOW

```
models = {
    LogisticRegression(max_iter=500):'Logistic Regression',
    SVC():"Support Vector Machine",
    RandomForestClassifier():'Random Forest'
}
for m in models.keys():
    m.fit(x_smote,y_smote)
for model,name in models.items():
     print(f"Accuracy Score for {name} is : ",model.score(X_test,y_test)*100,"%")
```

```
Accuracy Score for Logistic Regression is :  92.95580110497238 %
Accuracy Score for Support Vector Machine is :  7.458563535911603 %
Accuracy Score for Random Forest is :  93.37016574585635 %
```

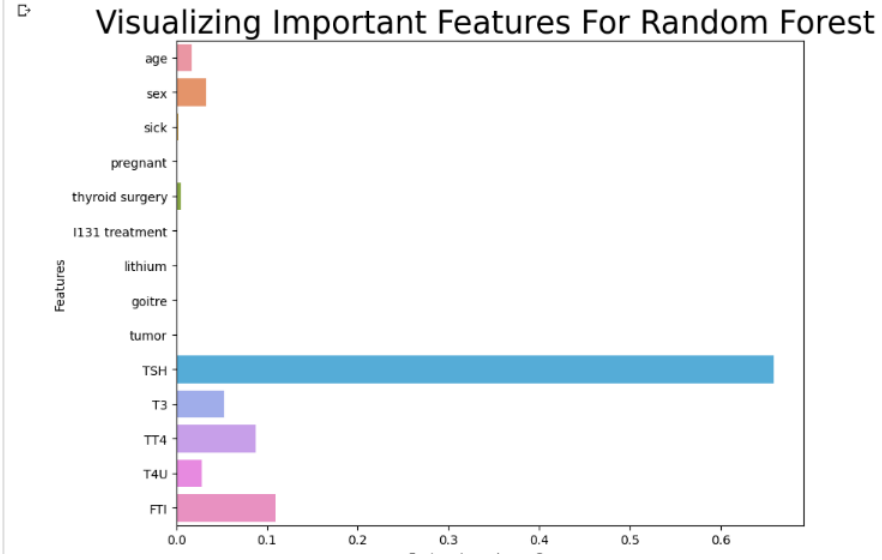
```python
for model,name in models.items():
    y_pred = model.predict(X_test)
    print(f"Classification Report for {name}")
    print("-------------------------------------------------------------")
    print(classification_report(y_test,y_pred))
    print("-------------------------------------------------------------")
```

```
Classification Report for Logistic Regression
-------------------------------------------------------------
              precision    recall  f1-score   support

           0       1.00      0.06      0.11        54
           1       0.93      1.00      0.96       670

    accuracy                           0.93       724
   macro avg       0.96      0.53      0.53       724
weighted avg       0.93      0.93      0.90       724

-------------------------------------------------------------
Classification Report for Support Vector Machine
-------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.07      1.00      0.14        54
           1       0.00      0.00      0.00       670

    accuracy                           0.07       724
   macro avg       0.04      0.50      0.07       724
weighted avg       0.01      0.07      0.01       724

-------------------------------------------------------------
Classification Report for Random Forest
-------------------------------------------------------------
              precision    recall  f1-score   support

           0       1.00      0.11      0.20        54
           1       0.93      1.00      0.97       670

    accuracy                           0.93       724
   macro avg       0.97      0.56      0.58       724
weighted avg       0.94      0.93      0.91       724
```

```python
rf = RandomForestClassifier()
rf.fit(x_smote,y_smote)
rf.score(X_test,y_test)
```

```
0.9419889502762431
```

```python
plt.figure(figsize=(9,7))
feature_imp1 = rf.feature_importances_
sns.barplot(x=feature_imp1, y=X.columns)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features For Random Forest ",{'fontsize':25})
plt.show();
```



```
[ ] x_smote.drop(['sick', 'pregnant', 'I131 treatment',
                  'lithium', 'goitre', 'tumor'], axis=1, inplace=True)
    X_test.drop(['sick', 'pregnant', 'I131 treatment',
                  'lithium', 'goitre', 'tumor'], axis=1, inplace=True)
```

```
[ ] new_rf = RandomForestClassifier()
    new_rf.fit(x_smote,y_smote)
    new_rf.score(X_test,y_test)
```

```
0.925414364640884
```

```
[ ] # with open('thyroid.pkl','wb') as f:
    #     pickle.dump(new_rf,f)
```

```
[ ] thyroid_df.head()
```

	age	sex	sick	pregnant	thyroid surgery	I131 treatment	lithium	goitre	tumor	TSH	T3	TT4	T4U	FTI	Label
0	41	F	f	f	f	f	f	f	f	1.30	2.5000	125	1.140	109	P
1	23	F	f	f	f	f	f	f	f	4.10	2.0000	102	0.995	110	P
2	46	M	f	f	f	f	f	f	f	0.98	2.0135	109	0.910	120	P
3	70	F	f	f	f	f	f	f	f	0.16	1.9000	175	0.995	110	P
4	70	F	f	f	f	f	f	f	f	0.72	1.2000	61	0.870	70	P

Colad Link:

https://colab.research.google.com/drive/1IryxOap8jk47ynJHN_rdKnBBqD95wZNX#scrollTo=k7jlqCf66lk_