Student Name: Om Yadav

Assignment: PwSkills

Domain: Healthcare

Project: Thyroid Disease Prediction Using Machine Learning

Tech Stack: Python, Machine Learning

```
#Importing libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import KNNImputer
from imblearn.over_sampling import SMOTENC,RandomOverSampler,KMeansSMOTE
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curve,auc,RocCurveDisplay
from sklearn.model_selection import cross_val_score
from sklearn.utils import resample
pd.set_option('display.max_columns', None)
import pickle
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

```
#Loading data as dataframe
#df = pd.read_csv(r"/content/hypothyroid.csv")
df = pd.read_csv(r"/content/hypothyroid.csv")
```

```
#reading first 5 row of dataframe
df.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_hyp
0	41	F	f	f	f	f	f	f	f	
1	23	F	f	f	f	f	f	f	f	
2	46	M	f	f	f	f	f	f	f	
3	70	F	t	f	f	f	f	f	f	
4	70	F	f	f	f	f	f	f	f	

```
#Checking information about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3772 entries, 0 to 3771
Data columns (total 30 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   age                        3772 non-null   object
 1   sex                        3772 non-null   object
 2   on_thyroxine               3772 non-null   object
 3   query_on_thyroxine         3772 non-null   object
 4   on_antithyroid_medication  3772 non-null   object
 5   sick                       3772 non-null   object
 6   pregnant                   3772 non-null   object
 7   thyroid_surgery            3772 non-null   object
 8   I131_treatment             3772 non-null   object
 9   query_hypothyroid          3772 non-null   object
 10  query_hyperthyroid         3772 non-null   object
 11  lithium                    3772 non-null   object
 12  goitre                     3772 non-null   object
 13  tumor                      3772 non-null   object
 14  hypopituitary              3772 non-null   object
 15  psych                      3772 non-null   object
 16  TSH_measured               3772 non-null   object
 17  TSH                        3772 non-null   object
 18  T3_measured                3772 non-null   object
```

```
 19  T3                        3772 non-null   object
 20  TT4_measured              3772 non-null   object
 21  TT4                       3772 non-null   object
 22  T4U_measured              3772 non-null   object
 23  T4U                       3772 non-null   object
 24  FTI_measured              3772 non-null   object
 25  FTI                       3772 non-null   object
 26  TBG_measured              3772 non-null   object
 27  TBG                       3772 non-null   object
 28  referral_source           3772 non-null   object
 29  Class                     3772 non-null   object
dtypes: object(30)
memory usage: 884.2+ KB
```

```
#Create a copy for better practice
data = df.copy()
```

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

```
(3772, 30)
```

```
# Display all Columns
data.columns
```

```
Index(['age', 'sex', 'on_thyroxine', 'query_on_thyroxine',
       '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', 'TBG_measured', 'TBG', 'referral_source',
       'Class'],
      dtype='object')
```

```
data.describe()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	qu
count	3772	3772	3772	3772	3772	3772	3772	3772	3772	
unique	94	3	2	2	2	2	2	2	2	
top	59	F	f	f	f	f	f	f	f	
freq	95	2480	3308	3722	3729	3625	3719	3719	3713	

```
#Checking for null values
data.isnull().sum()
```

	0
age	0
sex	0
on_thyroxine	0
query_on_thyroxine	0
on_antithyroid_medication	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	0
T3_measured	0
T3	0
TT4_measured	0
TT4	0
T4U_measured	0
T4U	0
FTI_measured	0
FTI	0
TBG_measured	0
TBG	0
referral_source	0
Class	0

dtype: int64

```
data.columns
```

```
Index(['age', 'sex', 'on_thyroxine', 'query_on_thyroxine',
       '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', 'TBG_measured', 'TBG', 'referral_source',
       'Class'],
      dtype='object')
```

```
#Checking for unique values in class column

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

```
array(['negative', 'compensated_hypothyroid', 'primary_hypothyroid',
       'secondary_hypothyroid'], dtype=object)
```

```
#

n = len(data[data['Class'] == 'secondary_hypothyroid'])
print("No of secondary_hypothyroid in Dataset:",n)

n = len(data[data['Class'] == 'primary_hypothyroid'])
print("No of primary_hypothyroid in Dataset:",n)

n = len(data[data['Class'] == 'compensated_hypothyroid'])
print("No of compensated_hypothyroid in Dataset:",n)

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

```
No of secondary_hypothyroid in Dataset: 2
No of primary_hypothyroid in Dataset: 95
No of compensated_hypothyroid in Dataset: 194
No of negative in Dataset: 3481
```

```
#For checking unique value of every column
for column in data.columns:
    print(column,'--->', (data[column].unique()))
```

```
 '119' '84' '81' '95' '66' '101' '147' '120' '69' '?' '39' '87' '63' '133'
 '86' '163' '162' '103' '96' '151' '112' '82' '138' '71' '77' '93' '107'
 '237' '110' '67' '88' '160' '118' '136' '114' '116' '94' '161' '11' '32'
 '124' '137' '92' '135' '105' '150' '126' '146' '91' '217' '141' '159'
 '122' '100' '111' '140' '205' '225' '85' '90' '74' '219' '127' '132'
 '128' '106' '144' '131' '56' '79' '142' '98' '177' '139' '78' '189' '180'
 '73' '145' '184' '38' '156' '75' '148' '14' '76' '54' '58' '27' '65'
 '193' '13' '143' '12' '64' '257' '164' '59' '167' '18' '41' '176' '37'
 '33' '44' '45' '154' '174' '203' '244' '62' '158' '60' '187' '250' '181'
 '157' '223' '272' '166' '213' '235' '10' '68' '231' '191' '48' '5.8'
 '169' '149' '210' '40' '155' '232' '42' '204' '430' '198' '230' '15'
 '170' '165' '47' '168' '194' '89' '52' '179' '192' '172' '4.8' '50' '182'
 '197' '214' '246' '196' '207' '19' '153' '22' '46' '200' '35' '226' '201'
 '233' '206' '31' '255' '178' '239' '195' '6' '36' '2' '3' '289' '240'
 '209' '43' '34' '252' '29' '263' '301' '23' '188' '211' '253' '21' '173'
 '261' '248' '51' '25' '53' '17' '220' '256' '9.5' '212' '273' '222' '186'
 '49' '372' '16' '28' '24' '4' '30' '2.9' '55' '216' '258']
T4U_measured ---> ['t' 'f']
T4U ---> ['1.14' '?' '0.91' '0.87' '1.3' '0.92' '0.7' '0.93' '0.89' '0.95' '0.99'
 '1.13' '0.86' '0.96' '0.94' '0.9' '1.02' '1.05' '0.62' '1.06' '1.55'
 '0.83' '1.09' '1.07' '1.27' '0.76' '1.16' '1' '0.56' '0.81' '0.68' '0.78'
 '0.85' '1.35' '1.15' '0.82' '1.03' '1.58' '0.79' '1.17' '0.71' '0.72'
 '0.88' '1.11' '1.2' '1.1' '1.33' '0.77' '1.24' '0.53' '1.44' '1.63'
 '1.51' '1.42' '1.23' '1.01' '0.98' '0.61' '1.12' '1.43' '1.25' '1.41'
 '1.68' '0.97' '0.84' '0.8' '1.04' '0.73' '1.08' '1.26' '1.46' '1.29'
 '1.34' '1.66' '1.21' '1.19' '0.75' '0.52' '1.83' '1.39' '1.5' '1.93'
 '1.18' '0.74' '0.58' '1.82' '0.6' '1.67' '1.22' '0.66' '0.67' '1.31'
 '0.54' '1.77' '1.59' '1.97' '1.69' '1.38' '1.28' '1.4' '0.69' '0.65'
 '1.74' '2.03' '1.73' '1.65' '1.36' '1.52' '0.57' '1.53' '1.84' '1.57'
 '1.75' '1.32' '1.37' '0.64' '1.79' '1.8' '0.48' '1.71' '1.62' '1.76'
 '1.56' '1.48' '0.59' '0.31' '1.94' '2.12' '1.47' '0.63' '0.944' '0.49'
 '1.88' '0.5' '0.38' '1.49' '0.41' '1.61' '1.7' '2.32' '0.46' '1.45'
 '1.54' '0.47' '0.36' '2.01' '0.25']
FTI_measured ---> ['t' 'f']
FTI ---> ['109' '?' '120' '70' '141' '78' '115' '132' '93' '121' '153' '151' '107'
 '119' '87' '81' '104' '130' '106' '116' '131' '190' '92' '102' '76' '98'
 '90' '61' '94' '129' '95' '91' '33' '113' '148' '140' '171' '155' '186'
 '122' '136' '110' '111' '97' '72' '100' '88' '67' '84' '103' '135' '203'
 '112' '117' '180' '142' '145' '156' '96' '134' '8.9' '60' '139' '41' '99'
 '89' '146' '124' '105' '85' '157' '143' '71' '221' '28' '108' '137' '83'
 '74' '170' '65' '101' '127' '274' '154' '114' '62' '86' '126' '125' '64'
 '172' '162' '79' '118' '73' '152' '163' '149' '14' '51' '165' '77' '32'
 '69' '80' '11' '54' '164' '123' '144' '10' '214' '200' '160' '53' '16'
 '138' '169' '56' '47' '133' '43' '68' '179' '224' '220' '82' '362' '182'
 '75' '66' '161' '57' '58' '312' '63' '128' '147' '158' '281' '207' '216'
 '251' '194' '46' '7' '42' '174' '395' '185' '13' '201' '48' '173' '167'
 '188' '150' '235' '175' '159' '5.4' '189' '59' '166' '34' '228' '232'
 '217' '177' '176' '195' '219' '17' '210' '168' '205' '39' '187' '50'
 '349' '52' '206' '253' '242' '244' '213' '178' '247' '215' '198' '19'
 '237' '37' '7.6' '24' '2' '3' '191' '223' '9' '29' '222' '204' '26' '218'
 '197' '49' '209' '183' '265' '199' '196' '20' '283' '36' '249' '181'
 '8.4' '291' '55' '245' '18' '40' '8.5' '184' '4' '21' '280' '2.8' '9.1'
 '27' '15' '35' '227']
TBG_measured ---> ['f']
TBG ---> ['?']
referral_source ---> ['SVHC' 'other' 'SVI' 'STMW' 'SVHD']
Class ---> ['negative' 'compensated_hypothyroid' 'primary_hypothyroid'
 'secondary_hypothyroid']
```

```
col_name = ['on_thyroxine', 'query_on_thyroxine',
       'on_antithyroid_medication', 'sick', 'pregnant', 'thyroid_surgery',
       'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH_measured']
```

```
for col in col_name:
    print(f"No.of 'f' and 't' values are in {col} column")
    print(f"No.of f value in {col} column : {len(data[data[col] == 'f'])}")
    print(f"No.of t value in {col} column : {len(data[data[col] == 't'])}", '\n', '---'*20)
```

```
No.of t value in on_thyroxine column : 464
 ------------------------------------------------------------
No.of 'f' and 't' values are in query_on_thyroxine column
No.of f value in query_on_thyroxine column : 3722
No.of t value in query_on_thyroxine column : 50
 ------------------------------------------------------------
No.of 'f' and 't' values are in on_antithyroid_medication column
No.of f value in on_antithyroid_medication column : 3729
No.of t value in on_antithyroid_medication column : 43
 ------------------------------------------------------------
No.of 'f' and 't' values are in sick column
No.of f value in sick column : 3625
No.of t value in sick column : 147
 ------------------------------------------------------------
No.of 'f' and 't' values are in pregnant column
No.of f value in pregnant column : 3719
No.of t value in pregnant column : 53
 ------------------------------------------------------------
No.of 'f' and 't' values are in thyroid_surgery column
No.of f value in thyroid_surgery column : 3719
No.of t value in thyroid_surgery column : 53
 ------------------------------------------------------------
No.of 'f' and 't' values are in I131_treatment column
No.of f value in I131_treatment column : 3713
No.of t value in I131_treatment column : 59
 ------------------------------------------------------------
No.of 'f' and 't' values are in query_hypothyroid column
No.of f value in query_hypothyroid column : 3538
No.of t value in query_hypothyroid column : 234
 ------------------------------------------------------------
No.of 'f' and 't' values are in query_hyperthyroid column
No.of f value in query_hyperthyroid column : 3535
No.of t value in query_hyperthyroid column : 237
 ------------------------------------------------------------
No.of 'f' and 't' values are in lithium column
No.of f value in lithium column : 3754
No.of t value in lithium column : 18
 ------------------------------------------------------------
No.of 'f' and 't' values are in goitre column
No.of f value in goitre column : 3738
No.of t value in goitre column : 34
 ------------------------------------------------------------
No.of 'f' and 't' values are in tumor column
No.of f value in tumor column : 3676
No.of t value in tumor column : 96
 ------------------------------------------------------------
No.of 'f' and 't' values are in hypopituitary column
No.of f value in hypopituitary column : 3771
No.of t value in hypopituitary column : 1
 ------------------------------------------------------------
No.of 'f' and 't' values are in psych column
No.of f value in psych column : 3588
No.of t value in psych column : 184
 ------------------------------------------------------------
No.of 'f' and 't' values are in TSH_measured column
No.of f value in TSH_measured column : 369
No.of t value in TSH_measured column : 3403
 ------------------------------------------------------------
```

```
#Checking for '?' value in our data which are null values

data.isin(['?']).sum()
```

	0
age	1
sex	150
on_thyroxine	0
query_on_thyroxine	0
on_antithyroid_medication	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	369
T3_measured	0
T3	769
TT4_measured	0
TT4	231
T4U_measured	0
T4U	387
FTI_measured	0
FTI	385
TBG_measured	0
TBG	3772
referral_source	0
Class	0

dtype: int64

```
# we can see that for column 'TBG' all the values are missing. So we will drop this column as it is of no use to us.

data = data.drop(['TBG'], axis=1)
```

looking to the dataset, we can see that some columns are with true and false value are just the indication that whether the next column has values or not.

```
data[['T4U_measured','T4U']]
```

	T4U_measured	T4U
0	t	1.14
1	f	?
2	t	0.91
3	f	?
4	t	0.87
...	...	...
3767	f	?
3768	t	1.08
3769	t	1.07
3770	t	0.94
3771	t	1.07

3772 rows × 2 columns

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

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


# Now let's replace the '?' values with nan value

for col in data.columns:
    count = data[col][data[col]=='?'].count()
    if count!=0:
        data[col] = data[col].replace('?',np.nan)


# For rechecking

for col in data.columns:
    count = data[col][data[col]=='?'].count()
    if count==0:
        print(col, data[col][data[col]=='?'].count())
```

```
age 0
sex 0
on_thyroxine 0
query_on_thyroxine 0
on_antithyroid_medication 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 0
T3 0
TT4 0
T4U 0
FTI 0
referral_source 0
Class 0
```

```
#Now ? is replaced with nan value. so checking for null value
data.isna().sum()
```

	0
age	1
sex	150
on_thyroxine	0
query_on_thyroxine	0
on_antithyroid_medication	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	369
T3	769
TT4	231
T4U	387
FTI	385
referral_source	0
Class	0

dtype: int64

```
#Now checking for datatypes of columns

data.dtypes
```

	0
age	object
sex	object
on_thyroxine	object
query_on_thyroxine	object
on_antithyroid_medication	object
sick	object
pregnant	object
thyroid_surgery	object
I131_treatment	object
query_hypothyroid	object
query_hyperthyroid	object
lithium	object
goitre	object
tumor	object
hypopituitary	object
psych	object
TSH	object
T3	object
TT4	object
T4U	object
FTI	object
referral_source	object
Class	object

dtype: object

As the datatype of all columns are object, so first we've to convert them.

```
# Mapping the categorical column

data['sex'] = data['sex'].map({'F' : 0, 'M' : 1})


col_name = ['on_thyroxine', 'query_on_thyroxine',
       'on_antithyroid_medication', 'sick', 'pregnant', 'thyroid_surgery',
       'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych',]

for col in col_name:
    if  len(data[col].unique())==2:
        data[col] = data[col].map({'f' : 0, 't' : 1})


data.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_hyp
0	41	0.0	0	0	0	0	0	0	0	
1	23	0.0	0	0	0	0	0	0	0	
2	46	1.0	0	0	0	0	0	0	0	
3	70	0.0	1	0	0	0	0	0	0	
4	70	0.0	0	0	0	0	0	0	0	

Encoding categorical columns

Double-click (or enter) to edit

```
# Check if 'referral_source' is in the DataFrame's columns
if 'referral_source' in data.columns:
    #Unique values of referral_source column
    print(data['referral_source'].unique())
else:
    print("Column 'referral_source' not found in the DataFrame.")

print(data.columns) # Print all column names to verify
```

```
['SVHC' 'other' 'SVI' 'STMW' 'SVHD']
Index(['age', 'sex', 'on_thyroxine', 'query_on_thyroxine',
       'on_antithyroid_medication', 'sick', 'pregnant', 'thyroid_surgery',
       'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U',
       'FTI', 'referral_source', 'Class'],
      dtype='object')
```

```
#Unique values of referral_source column

data['referral_source'].unique()
print(data.columns)
```

```
Index(['age', 'sex', 'on_thyroxine', 'query_on_thyroxine',
       'on_antithyroid_medication', 'sick', 'pregnant', 'thyroid_surgery',
       'I131_treatment', 'query_hypothyroid', 'query_hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U',
       'FTI', 'referral_source', 'Class'],
      dtype='object')
```

```
# using one-hot-encoding

data.columns = data.columns.str.strip()  # Remove leading/trailing whitespace
data = pd.get_dummies(data, columns=['referral_source'], drop_first=True)
```

Double-click (or enter) to edit

```
data.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_hyp
0	41	0.0	0	0	0	0	0	0	0	
1	23	0.0	0	0	0	0	0	0	0	
2	46	1.0	0	0	0	0	0	0	0	
3	70	0.0	1	0	0	0	0	0	0	
4	70	0.0	0	0	0	0	0	0	0	

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

```
array(['negative', 'compensated_hypothyroid', 'primary_hypothyroid',
       'secondary_hypothyroid'], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder

lblEn = LabelEncoder()

data['Class'] =lblEn.fit_transform(data['Class'])
```

```
data.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_hyp
0	41	0.0	0	0	0	0	0	0	0	
1	23	0.0	0	0	0	0	0	0	0	
2	46	1.0	0	0	0	0	0	0	0	
3	70	0.0	1	0	0	0	0	0	0	
4	70	0.0	0	0	0	0	0	0	0	

```
from sklearn.impute import KNNImputer

imputer=KNNImputer(n_neighbors=3, weights='uniform',missing_values=np.nan)
# For imputing the missing values
new_array=imputer.fit_transform(data)
# convert the nd-array returned in the step above to a Dataframe
new_data=pd.DataFrame(data=np.round(new_array), columns=data.columns)
```

```
new_data
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_
0	41.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	23.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	46.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	70.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	70.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
3767	30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3768	68.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3769	74.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3770	72.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3771	64.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

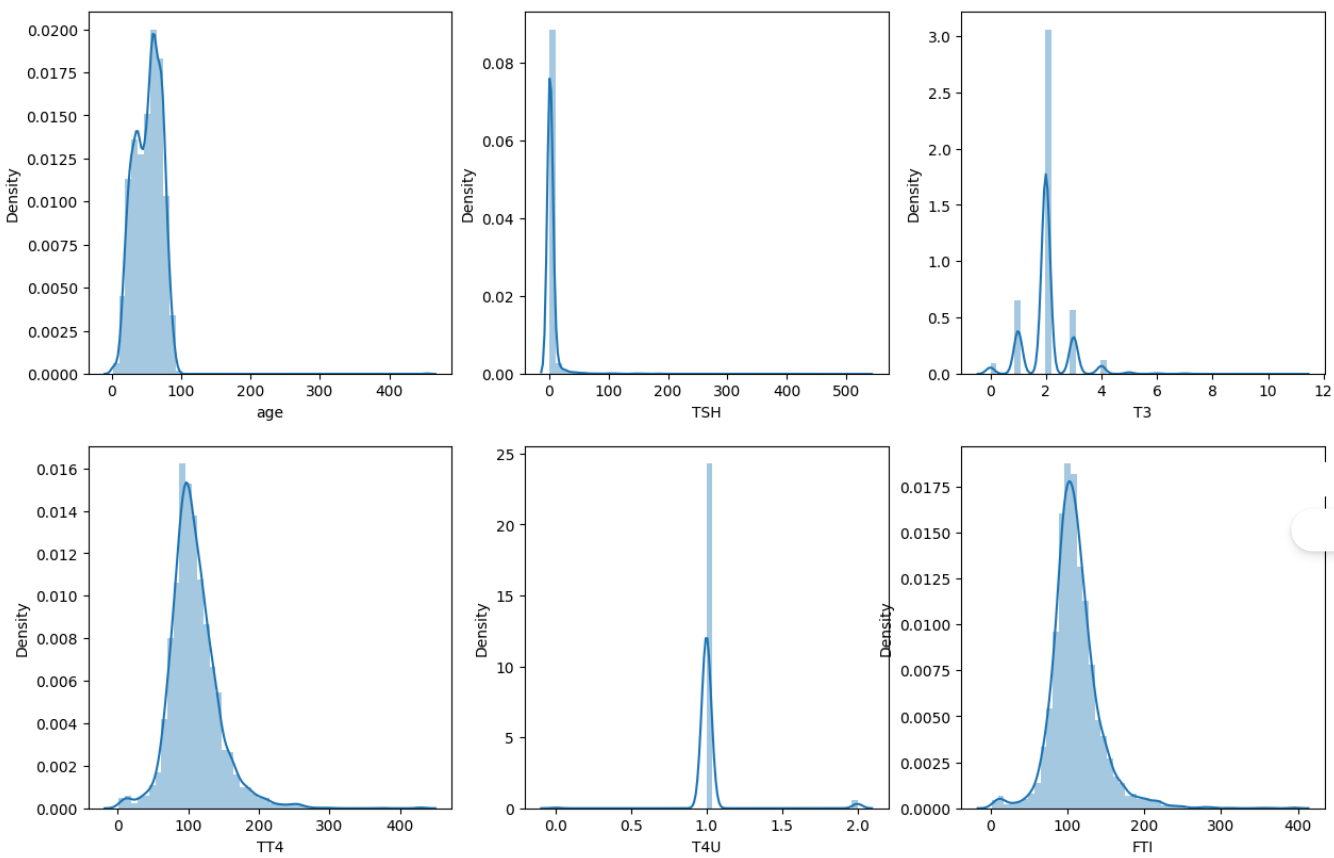
3772 rows × 26 columns

```
## For checking the distribution for our continuous data in the dataset.
```

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

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

for column in columns:
    ax = plt.subplot(3,3,plotnumber)
    sns.distplot(new_data[column])
    plt.xlabel(column,fontsize=10)
    plotnumber+=1
plt.show()
```



The graphs for age, TSH and T3 looks heavely skewed towards left.

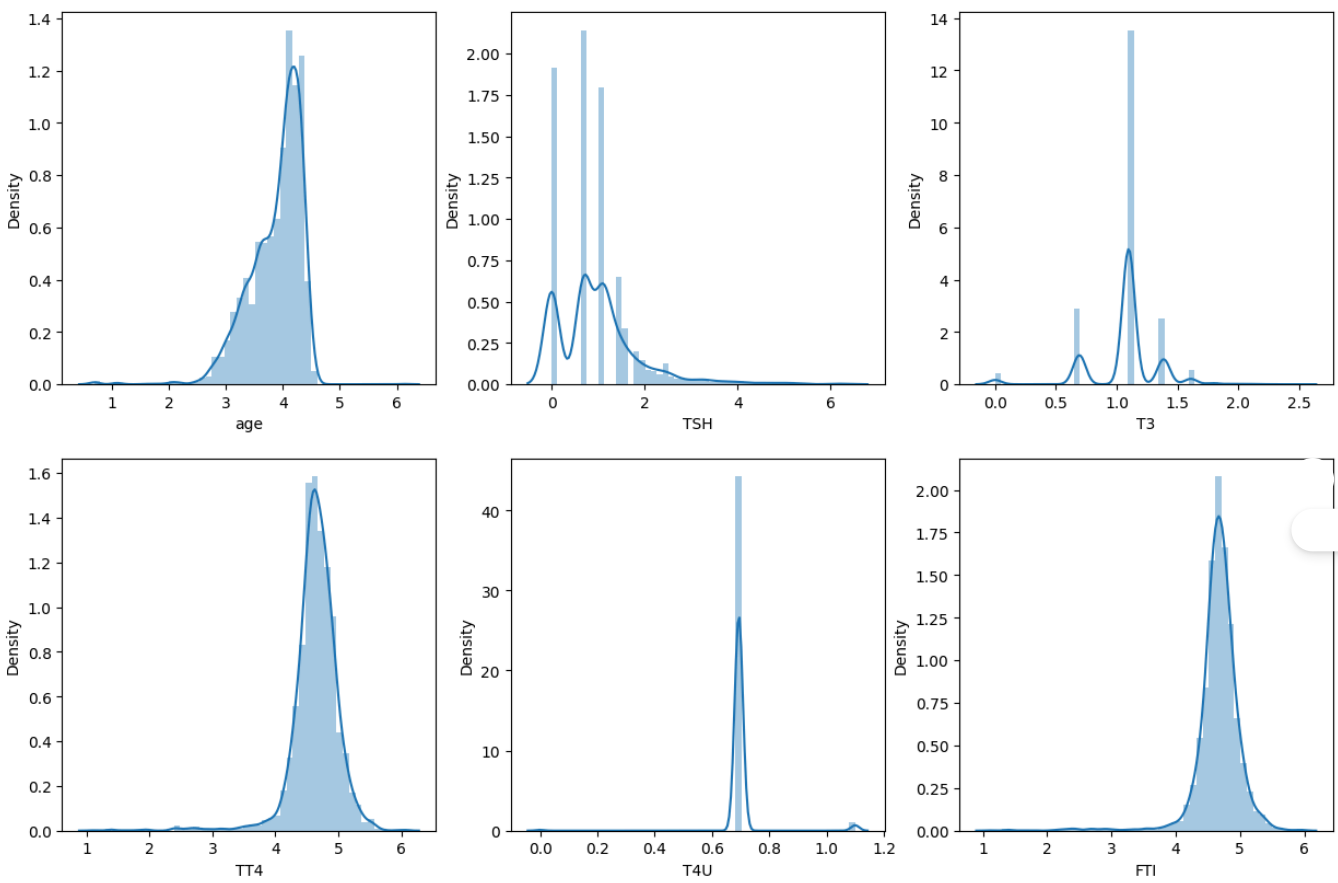
Let's do some transformations to the data and see if it improves the plot.

Before doing log transformation , let's add 1 to each valuue in the column to handle exception when we try to find log of '0'.

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

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

for column in columns:
    new_data[column]+=1
    ax = plt.subplot(3,3,plotnumber)
    sns.distplot(np.log(new_data[column]))
    plt.xlabel(column,fontsize=10)
    plotnumber+=1
plt.show()
```

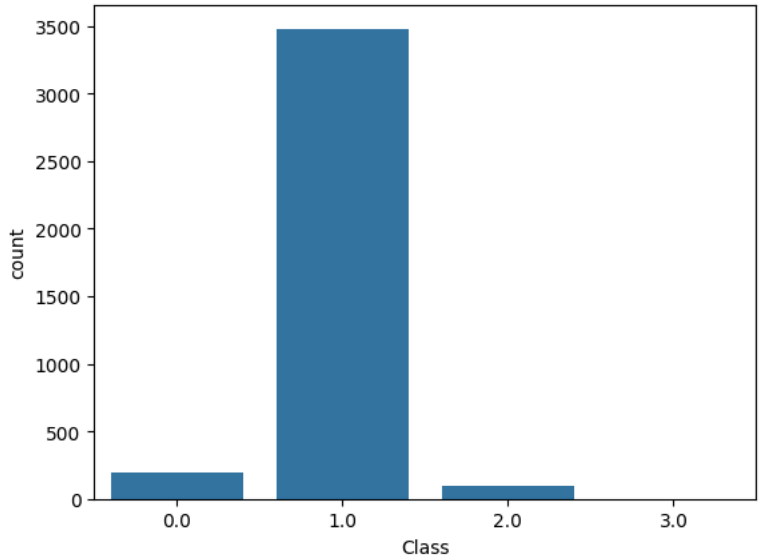


It won't give much of information so let's drop this column.

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

```
#countplot of class column to see the distribution
sns.countplot(data=new_data, x= 'Class')
```

```
<Axes: xlabel='Class', ylabel='count'>
```



We can clerly see that the dataset is highly imbalanced.

```
x = new_data.drop(['Class'],axis=1)
y = new_data['Class']
```

```
!pip install -U imbalanced-learn # Upgrade imblearn to the latest version
#For balancing the imbalance dataset

from imblearn.over_sampling import SMOTENC,RandomOverSampler,KMeansSMOTE
rdsample=RandomOverSampler()

x_sampled,y_sampled  = rdsample.fit_resample(x,y) # Use fit_resample instead of fit_sample
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.12.3)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.26.4)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.13.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.5.0)
```

```
#Checking for shape of x_sample
x_sampled.shape
```

```
(13924, 24)
```

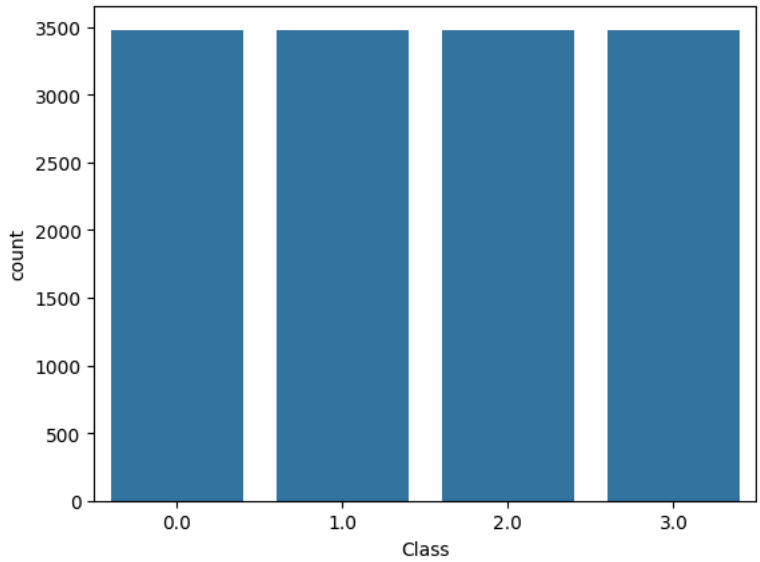
```
#creating dataframe of x_sample
x_sampled = pd.DataFrame(data = x_sampled, columns = x.columns)
x_sampled
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query
0	42.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	24.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	47.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	71.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	71.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
13919	47.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13920	42.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13921	47.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13922	42.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
13923	42.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

13924 rows × 24 columns

```
sns.countplot(data=new_data, x= y_sampled)
```

```
<Axes: xlabel='Class', ylabel='count'>
```

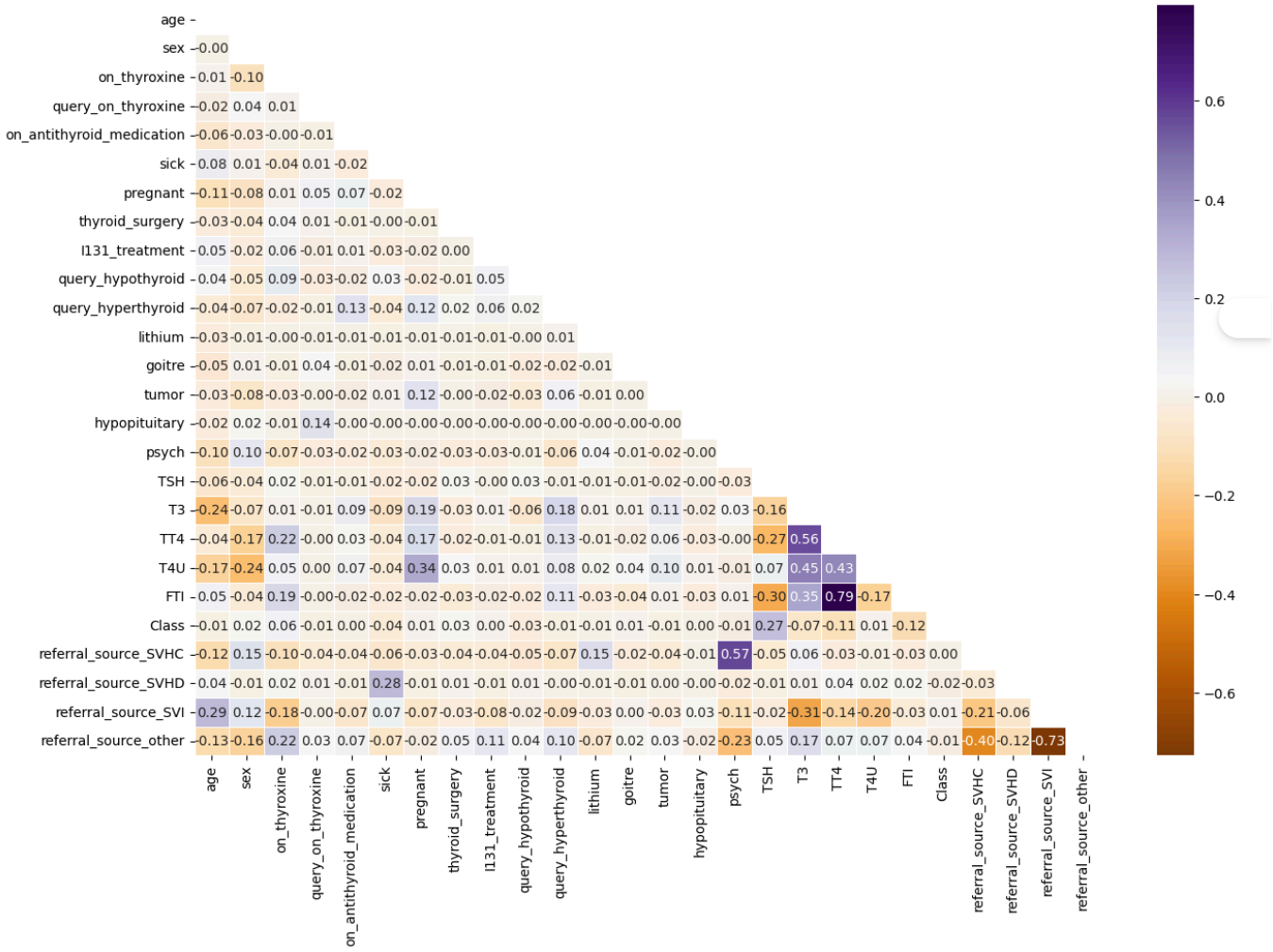


Now dataset is balanced.

```
# Correlation Analysis Using Heatmap

plt.figure(figsize=(15, 10))
mask = np.triu(np.ones_like(data.corr()))
sns.heatmap(data.corr(),mask= mask, annot=True, fmt='.2f', linewidths=0.5, cmap='PuOr')
```

```
<Axes: >
```




```
#Splitting data into train and test for model building

X_train,X_test,y_train,y_test=train_test_split(x_sampled,y_sampled,test_size=0.2,random_state=0)
```

Logistic regression

```
def log_classifier(X_train,X_test,y_train,y_test):
    log_model=LogisticRegression()
    log_model.fit(X_train,y_train)
    log_pred=log_model.predict(X_test)
    cm=confusion_matrix(y_test,log_pred)
    acc = accuracy_score(log_pred,y_test)
    return (f'Accuracy_Score: {acc}\n Train Score: {log_model.score(X_train,y_train)}\n Test Score: {log_model.score(X_test,y_test)}\n (
```

SVM

```
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)
    svm_pred=classifier_svm.predict(X_test)
    cm=confusion_matrix(y_test,svm_pred)
    acc = accuracy_score(svm_pred,y_test)
    return (f'Accuracy_Score: {acc}\n Train Score: {classifier_svm.score(X_train,y_train)}\n Test Score: {classifier_svm.score(X_test,y_
```

knn

```
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)
    knn_pred=classifier_knn.predict(X_test)
```