```
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.preprocessing import OneHotEncoder
```

```
data = pd.read_csv("thyroidDF.csv")
```

```
data.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant
0	29	F	f	f	f	f	f
1	29	F	f	f	f	f	f
2	41	F	f	f	f	f	f
3	36	F	f	f	f	f	f
4	32	F	f	f	f	f	f

5 rows × 31 columns

```
data.shape
```

```
(9172, 31)
```

```
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
```

```
data.drop(['TSH_measured','T3_measured','TT4_measured','T4U','FTI_measured','TBG_measured'
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnan
0	29	F	f	f	f	f	
1	29	F	f	f	f	f	
2	41	F	f	f	f	f	
3	36	F	f	f	f	f	
4	32	F	f	f	f	f	
...	...	...	...	...	...	...	.
9167	56	M	f	f	f	f	
9168	22	M	f	f	f	f	
9169	69	M	f	f	f	f	
9170	47	F	f	f	f	f	
9171	31	M	f	f	f	f	

9172 rows × 23 columns

```
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',
            '0': '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()
```

```
hypothyroid conditions     593
general health             436
binding protein            376
replacement therapy        336
miscellaneous              281
hyperthyroid conditions    182
antithyroid treatment       19
Name: target, dtype: int64
```

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

age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	tl

0 rows × 31 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2223 entries, 4 to 9169
Data columns (total 31 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   age                  2223 non-null   int64
 1   sex                  2133 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_measured         2223 non-null   object
 17  TSH                  2073 non-null   float64
 18  T3_measured          2223 non-null   object
 19  T3                   1629 non-null   float64
 20  TT4_measured         2223 non-null   object
 21  TT4                  2126 non-null   float64
 22  T4U_measured         2223 non-null   object
 23  T4U                  2045 non-null   float64
 24  FTI_measured         2223 non-null   object
 25  FTI                  2046 non-null   float64
 26  TBG_measured         2223 non-null   object
 27  TBG                  98 non-null     float64
 28  referral_source      2223 non-null   object
 29  target               2223 non-null   object
 30  patient_id           2223 non-null   int64
dtypes: float64(6), int64(2), object(23)
memory usage: 555.8+ KB
```

```
x.isnull().sum()
```

```
age                    0
sex                    0
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                    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
target                 0
dtype: int64
```

Filling Null Values

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

```
x
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnan
4	32	F	f	f	f	f	
18	63	F	t	f	f	t	
32	41	M	f	f	f	f	
33	71	F	t	f	f	f	
39	55	F	t	f	f	f	
...	...	...	...	...	...	...	.
9153	64	M	f	f	f	f	
9157	60	M	f	f	t	f	
9158	64	M	f	f	f	f	
9162	36	F	f	f	f	f	
9169	69	M	f	f	f	f	

2223 rows × 30 columns

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

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

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

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

```
F    1687
M     536
```

```
    Name: sex, dtype: int64
```

```
x['age']=x['age'].astype('int')
x['sex']=x['sex'].astype('string')
x['on_thyroxine']=x['on_thyroxine'].astype('string')
x['query_on_thyroxine']=x['query_on_thyroxine'].astype('string')
x['on_antithyroid_meds']=x['on_antithyroid_meds'].astype('string')
x['sick']=x['sick'].astype('string')
x['pregnant']=x['pregnant'].astype('string')
x['thyroid_surgery']=x['thyroid_surgery'].astype('string')
x['I131_treatment']=x['I131_treatment'].astype('string')
x['query_hypothyroid']=x['query_hypothyroid'].astype('string')
x['query_hyperthyroid']=x['query_hyperthyroid'].astype('string')
x['lithium']=x['lithium'].astype('string')
x['goitre']=x['goitre'].astype('string')
x['tumor']=x['tumor'].astype('string')
x['hypopituitary']=x['hypopituitary'].astype('string')
x['psych']=x['psych'].astype('string')
x['TSH_measured']=x['TSH_measured'].astype('string')
x['TSH']=x['TSH'].astype('float')
x['T3_measured']=x['T3_measured'].astype('string')
x['T3']=x['T3'].astype('float')
x['TT4_measured']=x['TT4_measured'].astype('string')
x['TT4']=x['TT4'].astype('float')
x['T4U_measured']=x['T4U_measured'].astype('string')
x['T4U']=x['T4U'].astype('float')
x['FTI_measured']=x['FTI_measured'].astype('string')
x['FTI']=x['FTI'].astype('float')
x['TBG_measured']=x['TBG_measured'].astype('string')
x['TBG']=x['TBG'].astype('float')
x['referral_source']=x['referral_source'].astype('string')
#x['patient_id']=x['patient_id'].astype('float')
x['target']=x['target'].astype('string')
```

```
x.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 2223 entries, 4 to 9169
    Data columns (total 30 columns):
     #   Column               Non-Null Count  Dtype
    ---  ------               --------------  -----
     0   age                  2223 non-null   int64
     1   sex                  2223 non-null   string
     2   on_thyroxine         2223 non-null   string
     3   query_on_thyroxine   2223 non-null   string
     4   on_antithyroid_meds  2223 non-null   string
     5   sick                 2223 non-null   string
     6   pregnant             2223 non-null   string
     7   thyroid_surgery      2223 non-null   string
     8   I131_treatment       2223 non-null   string
     9   query_hypothyroid    2223 non-null   string
     10  query_hyperthyroid   2223 non-null   string
     11  lithium              2223 non-null   string
     12  goitre               2223 non-null   string
     13  tumor                2223 non-null   string
```

```
 14  hypopituitary        2223 non-null   string
 15  psych                2223 non-null   string
 16  TSH_measured         2223 non-null   string
 17  TSH                  2073 non-null   float64
 18  T3_measured          2223 non-null   string
 19  T3                   1629 non-null   float64
 20  TT4_measured         2223 non-null   string
 21  TT4                  2126 non-null   float64
 22  T4U_measured         2223 non-null   string
 23  T4U                  2045 non-null   float64
 24  FTI_measured         2223 non-null   string
 25  FTI                  2046 non-null   float64
 26  TBG_measured         2223 non-null   string
 27  TBG                  98 non-null     float64
 28  referral_source      2223 non-null   string
 29  target               2223 non-null   string
dtypes: float64(6), int64(1), string(23)
memory usage: 538.4 KB
```

```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
```

```
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
ordinal_encoder = OrdinalEncoder(dtype='int64')
#x.iloc[:, 1:16] = ordinal_encoder.fit_transform(x.iloc[:, 1:16])
```

```
x.apply(lb.fit_transform)
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnan
4	29	0	0	0	0	0	
18	60	0	1	0	0	1	
32	38	1	0	0	0	0	
33	68	0	1	0	0	0	
39	52	0	1	0	0	0	
...	...	...	...	...	...	...	
9153	61	1	0	0	0	0	
9157	57	1	0	0	1	0	
9158	61	1	0	0	0	0	
9162	33	0	0	0	0	0	
9169	66	1	0	0	0	0	

2223 rows × 30 columns

```
#x.iloc[:, 16:29] = lb.fit_transform(x.iloc[:, 16:29])
```

```
x
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnan
4	32	F	f	f	f	f	
18	63	F	t	f	f	t	
32	41	M	f	f	f	f	
33	71	F	t	f	f	f	
39	55	F	t	f	f	f	
...	...	...	...	...	...	...	.
9153	64	M	f	f	f	f	
9157	60	M	f	f	t	f	
9158	64	M	f	f	f	f	
9162	36	F	f	f	f	f	
9169	69	M	f	f	f	f	

2223 rows × 30 columns

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

```
x
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnan
4	32	F	f	f	f	f	
18	63	F	t	f	f	t	
32	41	M	f	f	f	f	
33	71	F	t	f	f	f	
39	55	F	t	f	f	f	
...	...	...	...	...	...	...	.
9153	64	M	f	f	f	f	
9157	60	M	f	f	t	f	
9158	64	M	f	f	f	f	

```
label_encoder = LabelEncoder()
y_dt = label_encoder.fit_transform(y)
```

9169	69	M	f	f	f	f	

```
x= label_encoder.fit_transform(x)
```

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

```
y
```

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

```
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()
```

```
os = SMOTE()
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 imblearn.over_sampling import SMOTE
from imblearn.over_sampling import SMOTE
```

```
sm = SMOTE(random_state = 2)
x_train_res, y_train_res = sm.fit_resample(x_train, y_train)
```

```
from imblearn.over_sampling import RandomOverSampler
```

```
sm=RandomOverSampler(random_state=2)
```

```
x_train_res, y_train_res = sm.fit_resample(x_train, y_train)
```

```
x_train.info()
```

```
x_train.info
```

```
y_train
```

```
x.info()
```

```
x.info()
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_bal = sc.fit_transform(x_train_res)
x_test_bal= sc.transform(x_test_res)
```

```
x_bal
```

```
columns=['age","sex", "on thyroxine', 'query_on_thyroxine', 'on antithyroid meds', 'sick',
```

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

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

```
x_bal
```

```
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 seds
', 'sick', 'preg
```

```
importance= results.importances _mean importance = np.sort(importance)
```

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

```
plt.bar(x=feature_importance, height importance) plt.xticks(rotation-30, ha="right") plt.s
```

```
x.head()
```

```
x_bal.drop(['age","sex", 'on thyroxine", "query on thyroxine', 'on antithyroid meds', 'sic
```

```
x_test_bal.drop(['age', 'sex', 'on thyroxine', 'query_on_thyroxine', 'on antithyroid meds'
```

```
x_bal.head()
```

```
data.Info()
```

```
data.info()
```

```
#checking correlation using Heatmap
import seaborn as sns corrmat = x.corr()
f, ax = plt.subplots(figsize =(9, 8))
sns.heatmap (corrmat, ax = ax, cmap "Y1GnBu", linewidths = 0.1)
```

```
from sklearn.ensemble import RandomForestClassifier
rfr1 = RandomForestClassifier().fit(x_os,y_os.values.ravel())
y_pred rfr1.predict(x_test_os)
rfr1 = RandomForestClassifier()
```

```
rfr1.fit(x_os, y_os.values.ravel())
```

```
y_pred rfr1.predict(x_test_os)
```

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

```
train_score=accuracy_score (y_os, rfr1.predict(x_os))
train score
```

```
from xgboost import XGBClassifier
xgb1= XGBClassifier()
xgb1.fit(x_os,y_os)
```

```
y pred xgb1.predict(x_test_os)
```

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

```
accuracy_score (y_test_os,y_pred)


from sklearn.svm import SVC
from sk.learn.metrics import accuracy_scare,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))


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


model = Sequential()


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


[6:38 pm, 12/04/2023] Yuvaraj (Cs): model.add(Dense (units = 128, activation='relu', kerne
[6:38 pm, 12/04/2023] Yuvaraj (Cs): model.add(Dense (units = 256, activation='relu', kerne

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=15)


rfr1.predict([[0,0,0,0,0.000000, 0.0, 0.0,1.00, 0.0,48.0]])


sv.predict([[0,0,0,0,0.000000, 0.0, 0.0,1.08,8.8,48.8]])


col = ['goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4', 'T4U', 'FTI', 'TB(
da = [[0,0,0,0,0.000000, 0.0, 0.0,1.00, 0.0,40.0]]
dal = pd.DataFrame(data = da, columns=col)
gb1.predict(dal)
```

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

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

```
train_score = accuracy_score (y_bal, rfr1.predict(x_bal))
```

```
train_score
```

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

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

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

```
params={
    'C': [8.1, 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)
```

```
randon_svx.fit(x_bal,y_bal)
```

```
randon_svc.best_params_
```

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

```
svl.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_transformagt.predict(features)))
```

```
pickle.dump(label_encoder,open("label_encoder.pk","b"))
```

```
data['target']undque()
```

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