# Hypothyroidism Disease Prediction Using Neural Networks And Supervised Learning Techniques

Arwa Alqahtani - 1805926

# CONTENTS OF THIS TEMPLATE

1. Brief descriptions of the problem
2. pre-processing steps
3. features used for training and why you choose them
4. brief descriptions of all trained models
5. evaluation of all trained models
6. Model Tuning
7. error analysis or suggestion for improvements

# Brief descriptions of the problem

**Hypothyroidism** is a disorder in which the thyroid produces and releases insufficient thyroid hormone into the circulation. This slows down your metabolism. Hypothyroidism, also known as an underactive thyroid, causes fatigue, weight gain, and an inability to tolerate cold temperatures. Hypothyroidism can be difficult to diagnose since the symptoms are often mistaken with those of other diseases.

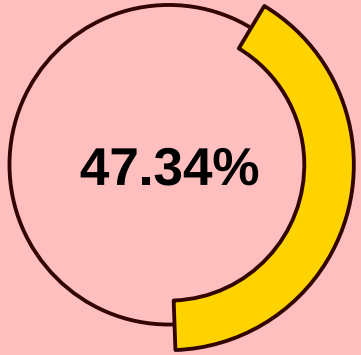
**The prevalence of hypothyroidism was 47.34% in Saudi Arabia**

**according to a study on the thyroid diseases in the Arab world.**

**From the results of this study,**

**it shows us the importance of finding a quick and accurate solution**

**to diagnosing hypothyroidism.**



# pre-processing steps

1. change the first column name to "target"
2. convert target values to 0 for negative and 1 for hypothyroid
3. Check the List of columns that containing null values
4. changing the categorical values into binary values

# change the first column name to "target"

```
# change the first column name to "target"
dataset = dataset.rename(columns = {dataset.columns[0]:"target"})
```

# convert target values to 0 for negative and 1 for hypothyroid

```
# convert target values to 0 for negative and 1 for hypothyroid
dataset["target"] = dataset["target"].map({"negative":0,"hypothyroid":1})
```

# Check the List of columns that containing null values

```
# Check the List of columns that containing null values
dataset.columns[dataset.isnull().any()].tolist()
```

```
# Replacing Age and T4U null values by mean
dataset['Age'].fillna(dataset['Age'].mean(), inplace = True)
dataset['T4U'].fillna(dataset['T4U'].mean(), inplace = True)

# Replacing TSH, T3, TT4, FTI null values by median
dataset['TSH'].fillna(dataset['TSH'].mean(), inplace = True)
dataset['T3'].fillna(dataset['T3'].median(), inplace = True)
dataset['TT4'].fillna(dataset['TT4'].median(), inplace = True)
dataset['FTI'].fillna(dataset['FTI'].median(), inplace = True)

# The gender data looks to be imbalanced with 0 lesser than 1
# Replacing null values with 0
dataset['Sex'].fillna(0, inplace = True)
```

# changing the categorical values into binary values

```
# changing the categorical values into binary values
dataset = dataset.replace({'f':0,'t':1, 'y':1, 'n':0, 'M':0, 'F':1})
```

# features used for training and why you choose them

These are important features based on previous scientific papers:

| Attributes | Description |
|---|---|
| Age | In years |
| Sex | Male or female |
| TSH | Thyroid-Stimulating Hormone |
| T3 | Triiodothyronine |
| T4U | Thyroxin utilization rate |
| TT4 | Total Thyroxin |
| FTI | Free Thyroxin Index |

# brief descriptions of all trained models

## 1- Artificial Neural Network

we are defining a neural network containing a sequential model with one hidden layer and one output layer. We have 24 feature columns due to which we have specified the input dimension as input_dim=24. We are using the ReLU activation function in the hidden layer, and sigmoid function for the output layer to return a binary value of output, i.e. hypothyroid (1) or negative (0). then I will Compiling the model with 'adam' optimizer and loss function as 'binary_crossentropy'.

```
# Input
model = Sequential()
# Hidden layer
model.add(Dense(64, kernel_initializer='uniform', input_dim=24, activation='relu'))
# Output layer
model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))
```

**Compiling the model with 'adam' optimizer and loss function as 'binary_crossentropy'.**

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

**5. Training the model**

We will now train the model with a validation split of 0.2 for 100 epochs.

```
# Training the model
result = model.fit(X_train, y_train, epochs=100, validation_split=0.2, batch_size=40,  verbose=2)
```

# brief descriptions of all trained models

2- Random Forest

## 7. Building the Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rf_model= RandomForestClassifier(random_state=10)
rf_model.fit(X_train,y_train)
y_pred_rf= rf_model.predict(X_test)
```

# brief descriptions of all trained models

3- decision tree

## 9. Building the decision tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dt_model= DecisionTreeClassifier(random_state=10)
dt_model.fit(X_train,y_train)
y_pred_dt= dt_model.predict(X_test)
```

# evaluation of all trained models

The evaluation metrics I used is accuracy and F1 score. the following table shows the result of three different models:

| Algorithm Used | Accuracy | F1 score |
|---|---|---|
| ANN | 98.42 | 85.71 |
| Random forest | 98.57 | 86.56 |
| DT | 98.57 | 86.95 |

# Model Tuning

The best model based on the accuracy and F1-score is decision tree . now , we will fine tune the chosen model using grid search with at least one important parameter tuned with at least 3 different values:

```
#import 'Grid search' and other metrics
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, r2_score, fbeta_score

#create parameters list
parameters = {
    "criterion":['gini','entropy'],
    "max_depth":[1,2,3,4,5,6,7,None]
}
f1=make_scorer(f1_score)
grid=GridSearchCV(dt_model, param_grid= parameters, cv=10, n_jobs=-1, scoring= "f1")

grid.fit(X_train,y_train)
```

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=10),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, None]},
             scoring='f1')
```

```
grid.best_estimator_
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=10)
```

```
best_pred=grid.predict(X_test)
```

# The result after model tuning

The table below shows an improvement in the optimized model.

| Metrics | Unoptimized model | Optimized model |
|---|---|---|
| accuracy | 98.57 | 99.37 |
| F1 score | 86.59 | 94.44 |

# suggestion for improvements

Undoubtedly, the field of machine learning is vast and large. And medical problems increase with progression of time. So, in the future work it is possible to increase the size of the data and choose other algorithms to see which one is better.

# THANKS!

DO YOU HAVE ANY QUESTIONS?