THYROID DISEASE CLASSIFICATION USING ML

TEAM ID : NM2023TMID19513

TEAM SIZE : 5

TEAM LEADER : VIGNESH.B (20UCS2549)

TEAM MEMBERS : VIJAY.B (20UCS2550)

SANDHIYA.S (20UCS2551)

SANTHI.V (20UCS2552)

MANISHA(20UCS2553)

THYROID DISEASE CLASSIFICATION USING ML

1.Introduction:

1.1.Overview:

THYROID DISEASE ABSTRACTION

The thyroide gland is a small organ that's located in the front of the neck ,wrapped around the windpipe(trachea).It's shaped like a butterfly ,smaller in the middle with two wide wings that extend around the side of your throat. The thyroid is a gland. You have glands throughout your body, where they create and release substances that help your body do a specific thing. Your thyroid makes hormones that help control many vital functions of your body.

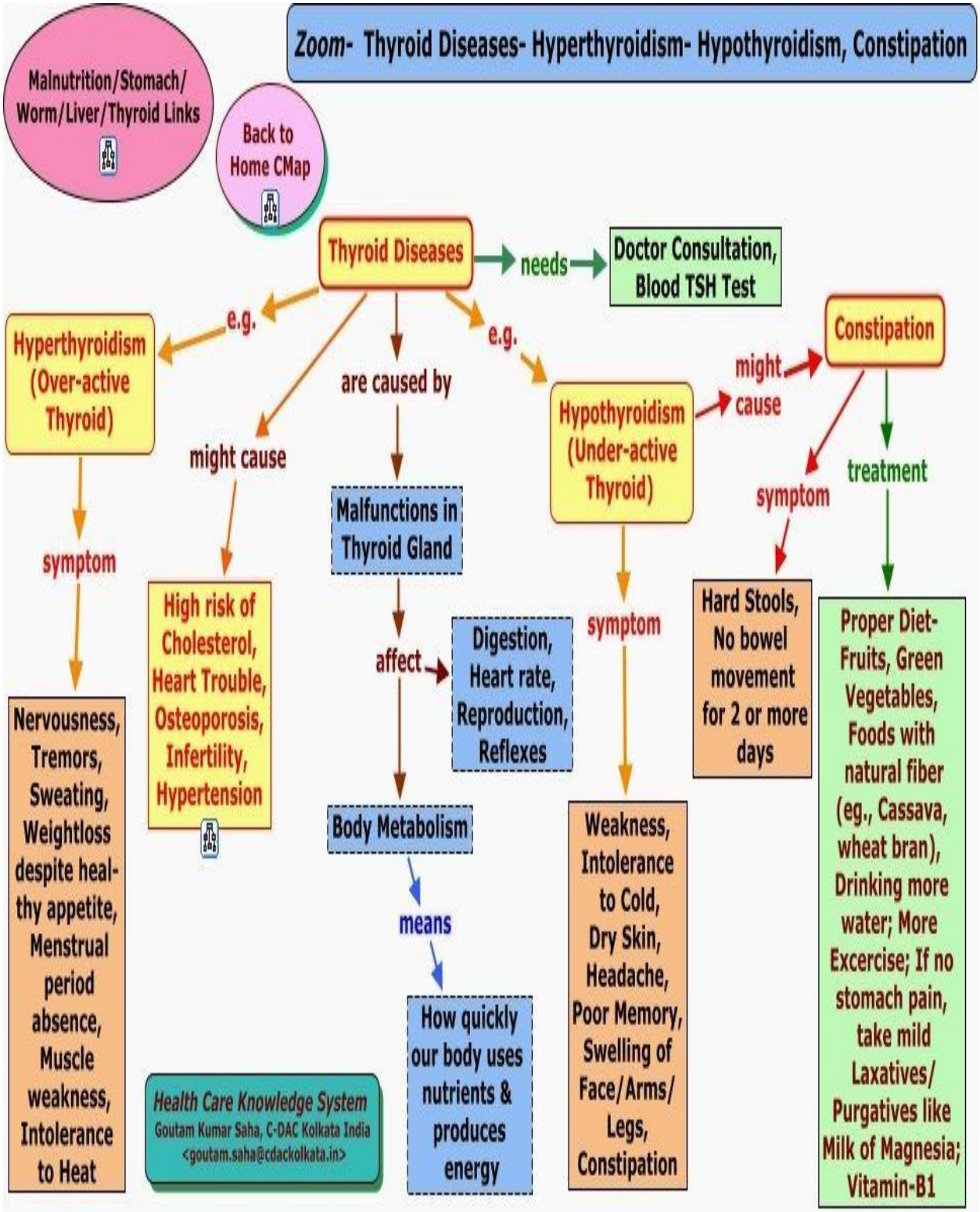
When your thyroid doesn't work properly, it can impact your entire body. If your body makes too much thyroid hormone, you can develop a condition called hyperthyroidism. If your body makes too little thyroid hormone, it's called hypothyroidism. Both conditions are serious and need to be treated by your healthcare provider.

1.2.Purpose:

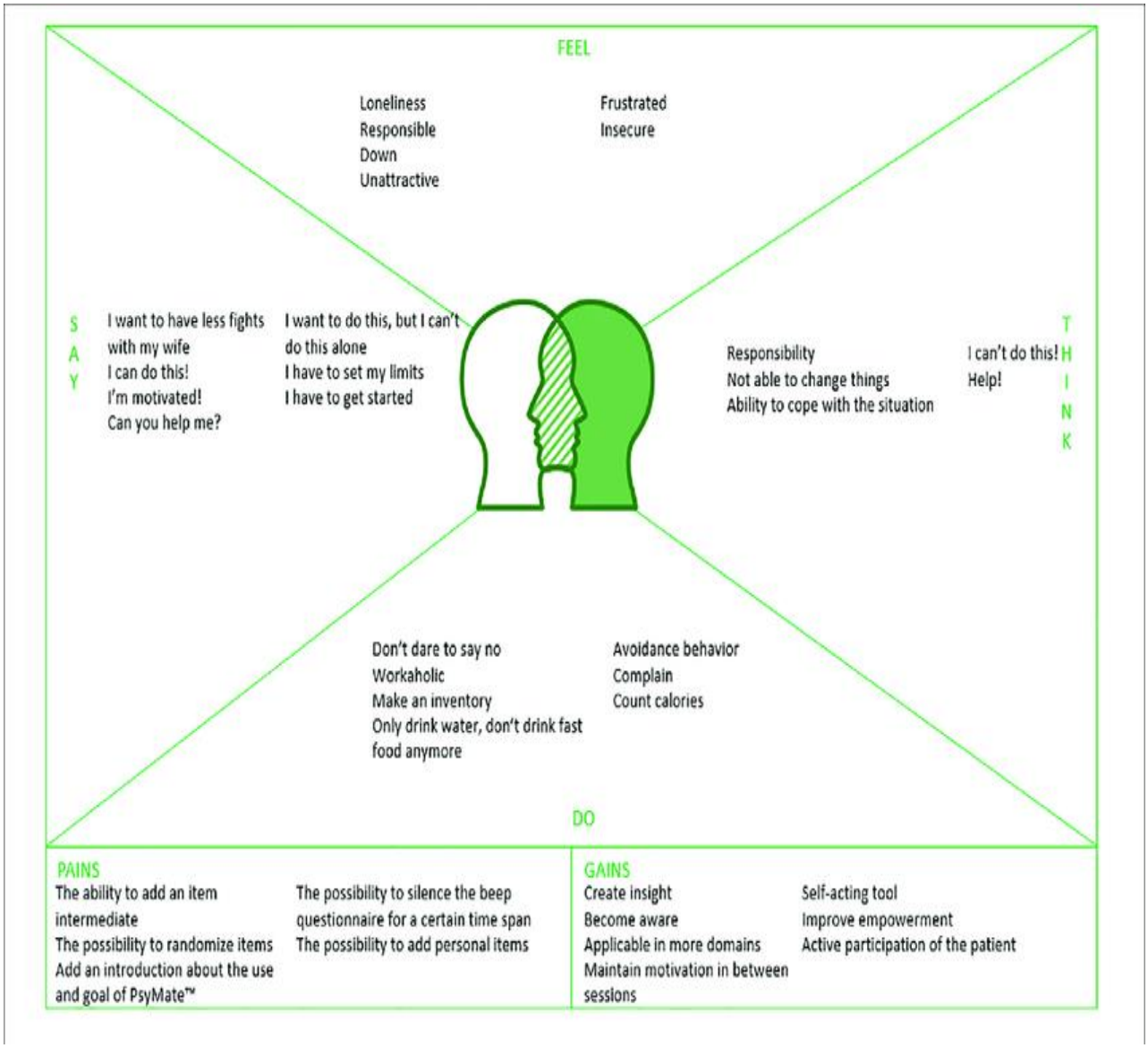
The thyroid gland is a vital hormone gland: It plays a major rolein the metabolism,growth and development of the human body. It helps to regulate many body functions by constandly releasing a steady amount of thyroid hormones into the bloodstream.

2. Problem Definition & Design Thinking:

2.1:THYROID DISEASE EMPATHY MAP:



2.2: Ideation & Brainstorming Map:



3.RESULT:

4.ADVANTAGES & DISADVANTAGES:

Advantages:

It helps to regulate many body functions by constantly releasing a steady amount of thyroid hormones into the bloodstream. If the body needs more energy in certain situations – for instance, if it is growing or cold, or during pregnancy – the thyroid gland produces more hormones.

Disadvantages:

1.Experiencing anxiety, irritability and nervousness.

2.Having trouble sleeping.

3.Losing weight.

4.Having an enlarged thyroid gland or a goiter.

5.Having muscle weakness and tremors.

6.Experiencing irregular menstrual periods or having your menstrual cycle stop.

5.APPLICATION:

The thyroid gland is a vital hormone gland: It plays a major role in the metabolism, growth and development of the human body. It helps to regulate many body functions by constantly releasing a steady amount of thyroid hormones into the bloodstream.

6.CONCLUSION:

The article performs the prediction of hypothyroid and its multiclass classification by applying ML and DL models. ML techniques utilized for model building are naïve Bayes, random forest, multiclass classification, and decision tree, along with the DL-based ANN model. The performance of predictive models has been compared using precision, recall, F1-score, ROC, and accuracy. It is observed that prediction models built up using a decision tree and random forest provide the highest accuracy, which is 99.5758% and 99.3107%, respectively. Where the error rate is also very less in both the models which concludes them as the best performing models in case of correct prediction of hypothyroidism type. Furthermore, the DL-based ANN model has also shown an accuracy of 93.82%, which is quite good and a competitive score, however, for the presented dataset, ML models decision tree, and random forest has achieved better accuracy. ANN is a good DL-based model which provides good results for text data. As it comes under DL models, hence it requires a large amount of information for training to correctly classify the data. Lastly, this work can be used for further research on the early detection of hypothyroid and its classification so that the timely treatment of the disease can be done and may lead to slowing down the enhancement in mortality rate.

In the future, the work can be used in hospitals to provide support to doctors and clinicians during the diagnosis of hypothyroidism. Further, the dataset can be increased, and academic researchers in the medical area can also use this work to identify more ML or DL based prediction models to detect hypothyroidism.

7.FUTURE SCOPE:

Thyroid disease incidences have been on the rise in recent times. The thyroid gland has one of the most important functions in regulating metabolism. Irregularities in the thyroid gland can lead to different abnormalities; two of the most common are hyperthyroidism and hypothyroidism. A large number of people are diagnosed with thyroid diseases such as hypothyroidism and hyperthyroidism yearly [1]. The thyroid gland produces levothyroxine (T4) and triiodothyronine (T3) and insufficient thyroid hormones may lead to hypothyroidism and hyperthyroidism [2]. Many approaches are proposed to detect thyroid disease diagnosis in the literature. A proactive thyroid disease prediction is essential to properly treat the patient at the right time and save human lives and medical expenses. Due to the technological advancements in data processing and computation, machine learning and deep learning techniques are applied to predict the thyroid diagnosis in the early stages and classify the thyroid disease types hypothyroidism, hyperthyroidism, etc.

8. APPENDIX :

```
import os
import numpy as np
import pandas as pd
import random
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
import seaborn as sns
from contextlib import contextmanager
from time import time
from tqdm import tqdm
import lightgbm as lgbm
import category_encoders as ce
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
data0=pd.read_csv('../input/thyroid-disease-data/thyroidDF.csv')
display(data0)
print(data0.columns.tolist())
name0=data0['target'].unique().tolist()
name=sorted(name0)
print(name)
N=list(range(len(name)))
normal_mapping=dict(zip(name,N))
reverse_mapping=dict(zip(N,name))
data0['target']=data0['target'].map(normal_mapping)
```

```
from sklearn.preprocessing import LabelEncoder

def labelencoder(df):
    for c in df.columns:
        if df[c].dtype=='object':
            df[c] = df[c].fillna('N')
            lbl = LabelEncoder()
            lbl.fit(list(df[c].values))
            df[c] = lbl.transform(df[c].values)
    return df
data1=labelencoder(data0.copy())
n=len(data1)
print(n)
N=list(range(n))
random.seed(2022)
random.shuffle(N)
target=['target']
dataY = data1[target]
dataX = data1.drop(target+['patient_id'],axis=1)

trainX=dataX.iloc[N[0:(n//5)*4]]
trainY=dataY.iloc[N[0:(n//5)*4]]

testX=dataX.iloc[N[(n//5)*4:]]
testY=dataY.iloc[N[(n//5)*4:]]
df_columns = list(trainX.columns)
print(df_columns)
def create_numeric_feature(input_df):
    use_columns = df_columns
    return input_df[use_columns].copy()
from contextlib import contextmanager
from time import time

class Timer:
    def __init__(self, logger=None, format_str='{:.3f}[s]', prefix=None,
suffix=None, sep=' '):

        if prefix: format_str = str(prefix) + sep + format_str
        if suffix: format_str = format_str + sep + str(suffix)
        self.format_str = format_str
        self.logger = logger
        self.start = None
        self.end = None

    @property
```

```
    def duration(self):
        if self.end is None:
            return 0
        return self.end - self.start

    def __enter__(self):
        self.start = time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.end = time()
        out_str = self.format_str.format(self.duration)
        if self.logger:
            self.logger.info(out_str)
        else:
            print(out_str)
from tqdm import tqdm

def to_feature(input_df):

    processors = [
        create_numeric_feature,
    ]

    out_df = pd.DataFrame()

    for func in tqdm(processors, total=len(processors)):
        with Timer(prefix='create' + func.__name__ + ' '):
            _df = func(input_df)

        assert len(_df) == len(input_df), func.__name__
        out_df = pd.concat([out_df, _df], axis=1)

    return out_df
train_feat_df = to_feature(trainX)
test_feat_df = to_feature(testX)
import lightgbm as lgbm
from sklearn.metrics import mean_squared_error

def fit_lgbm(X, y, cv,
             params: dict=None,
             verbose: int=50):

    if params is None:
        params = {}
```

```
    models = []
    oof_pred = np.zeros_like(y, dtype=np.float)

    for i, (idx_train, idx_valid) in enumerate(cv):
        x_train, y_train = X[idx_train], y[idx_train]
        x_valid, y_valid = X[idx_valid], y[idx_valid]

        clf = lgbm.LGBMRegressor(**params)

        with Timer(prefix='fit fold={} '.format(i)):
            clf.fit(x_train, y_train,
                    eval_set=[(x_valid, y_valid)],
                    early_stopping_rounds=100,
                    verbose=verbose)

        pred_i = clf.predict(x_valid)
        oof_pred[idx_valid] = pred_i
        models.append(clf)
        print(f'Fold {i} RMSLE: {mean_squared_error(y_valid, pred_i) ** .5:.4f}')
        print()

    score = mean_squared_error(y, oof_pred) ** .5
    print('-' * 50)
    print('FINISHED | Whole RMSLE: {:.4f}'.format(score))
    return oof_pred, models
params = {
    'objective': 'rmse',
    'learning_rate': .1,
    'reg_lambda': 1.,
    'reg_alpha': .1,
    'max_depth': 5,
    'n_estimators': 10000,
    'colsample_bytree': .5,
    'min_child_samples': 10,
    'subsample_freq': 3,
    'subsample': .9,
    'importance_type': 'gain',
    'random_state': 71,
    'num_leaves': 62
}
y = trainY
ydf=pd.DataFrame(y)
display(ydf)
from sklearn.model_selection import KFold
```

```
for i in range(1):
    fold = KFold(n_splits=5, shuffle=True, random_state=71)
    ydfi=ydf.iloc[:,i]
    y=np.array(ydfi)
    cv = list(fold.split(train_feat_df, y))
    oof, models = fit_lgbm(train_feat_df.values, y, cv, params=params, verbose=500)

    fig,ax = plt.subplots(figsize=(6,6))
    ax.set_title(target[i],fontsize=20)
    ax.set_ylabel('Train Predicted '+target[i],fontsize=12)
    ax.set_xlabel('Train Actual '+target[i],fontsize=12)
    ax.scatter(y,oof)
    def visualize_importance(models, feat_train_df):

    feature_importance_df = pd.DataFrame()
    for i, model in enumerate(models):
        _df = pd.DataFrame()
        _df['feature_importance'] = model.feature_importances_
        _df['column'] = feat_train_df.columns
        _df['fold'] = i + 1
        feature_importance_df = pd.concat([feature_importance_df, _df],
                                          axis=0, ignore_index=True)

    order = feature_importance_df.groupby('column')\
        .sum()[['feature_importance']]\
        .sort_values('feature_importance', ascending=False).index[:50]

    fig, ax = plt.subplots(figsize=(8, max(6, len(order) * .25)))
    sns.boxenplot(data=feature_importance_df,
                  x='feature_importance',
                  y='column',
                  order=order,
                  ax=ax,
                  palette='viridis',
                  orient='h')

    ax.tick_params(axis='x', rotation=0)
    #ax.set_title('Importance')
    ax.grid()
    fig.tight_layout()

    return fig,ax

#fig, ax = visualize_importance(models, train_feat_df)
for i in range(1):
```

```
    fold = KFold(n_splits=5, shuffle=True, random_state=71)
    ydfi=ydf.iloc[:,i]
    y=np.array(ydfi)
    cv = list(fold.split(train_feat_df, y))
    oof, models = fit_lgbm(train_feat_df.values, y, cv, params=params, verbose=500)
    fig, ax = visualize_importance(models, train_feat_df)
    ax.set_title(target[i]+' Imortance',fontsize=20)
    preds=[]
for i in range(5):
    preds += [models[i].predict(test_feat_df.values)/5]
predsT=np.array(preds).T
print(predsT.shape)
preds2=[]
for item in predsT:
    preds2+=[sum(item)]
print(np.array(preds2).shape)
for i in range(1):
    fig, ax = plt.subplots(figsize=(10,5))
    sns.histplot(oof, label='Train Predicted '+target[i], ax=ax,
color='C1',bins=60)
    sns.histplot(preds2, label='Test Predicted '+target[i], ax=ax,
color='black',bins=60)
    ax.legend()
    ax.grid()
    fig,ax = plt.subplots(figsize=(15,6))
ax.set_title('FTI vs Target',fontsize=20)
ax.set_xlabel('target',fontsize=12)
ax.set_ylabel('FTI',fontsize=12)
ax.scatter(data0['target'].map(reverse_mapping),data0['FTI'])
plt.show()
```