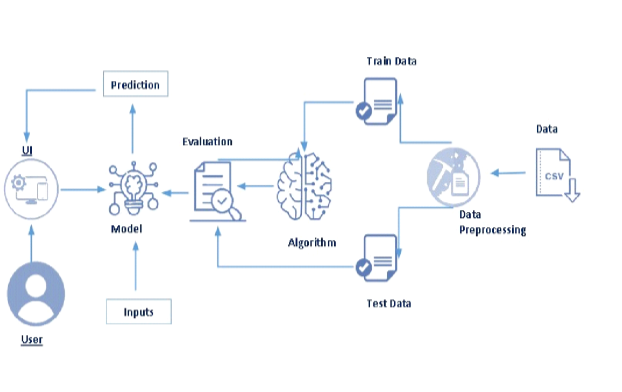
**PROJECT TITLE**

**THYROID DISEASE CLASSIFICATION USING ML**

**PROJECT DESCRIPTION**

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**

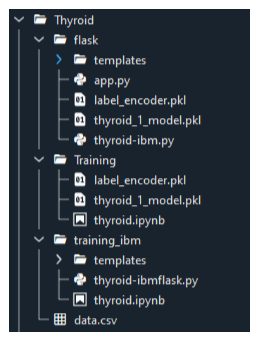
* The user interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once the model analyses the input the prediction is showcased on the UI

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

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

**PROJECT STRUCTURE**

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**

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.

**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**

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 following figure.

|  |
| --- |
| 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 |

**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.

**data=pd.read\_csv("D:\\NMDS\\drug200.csv")**

**data.head()**

| **Age** | **Sex** | **BP** | **Cholesterol** | **Na\_to\_K** | **Drug** |
| --- | --- | --- | --- | --- | --- |
| **0** | 23 | F | HIGH | HIGH | 25.355 | DrugY |
| **1** | 47 | M | LOW | HIGH | 13.093 | drugC |
| **2** | 47 | M | LOW | HIGH | 10.114 | drugC |
| **3** | 28 | F | NORMAL | HIGH | 7.798 | drugX |
| **4** | 61 | F | LOW | HIGH | 18.043 | DrugY |

**data.columns()**

Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na\_to\_K', 'Drug'], dtype='object')

**data.Drug**

0 DrugY

1 drugC

2 drugC

3 drugX

4 DrugY

...

195 drugC

196 drugC

197 drugX

198 drugX

199 drugX

Name: Drug, Length: 200, dtype: object

**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.

1. Handling missing values
2. Descriptive analysis
3. Splitting the dataset as x and y
4. Handling Categorical Values
5. Checking Correlation
6. Converting Data Type
7. Splitting dataset into training and test set
8. Handled Imbalanced Data
9. 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.

**data.isnull()**

| **Age** | **Sex** | **BP** | **Cholesterol** | **Na\_to\_K** | **Drug** |
| --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... |
| **195** | False | False | False | False | False | False |
| **196** | False | False | False | False | False | False |
| **197** | False | False | False | False | False | False |
| **198** | False | False | False | False | False | False |
| **199** | False | False | False | False | False | False |

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.

**data.isnull().sum()**

Age 0

Sex 0

BP 0

Cholesterol 0

Na\_to\_K 0

Drug 0

dtype: int64

**Milestone 3: Exploratory Data Analysis**

**Activity 1: Descriptive analysis**

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this described function we can find mean, std, min, max and percentile values of continuous features.

Checking info about data by using data.info()

**data.info()**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 200 entries, 0 to 199

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Age 200 non-null int64

1 Sex 200 non-null object

2 BP 200 non-null object

3 Cholesterol 200 non-null object

4 Na\_to\_K 200 non-null float64

5 Drug 200 non-null object

dtypes: float64(1), int64(1), object(4)

memory usage: 9.5+ KB

**Activity 2: Visual analysis**

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

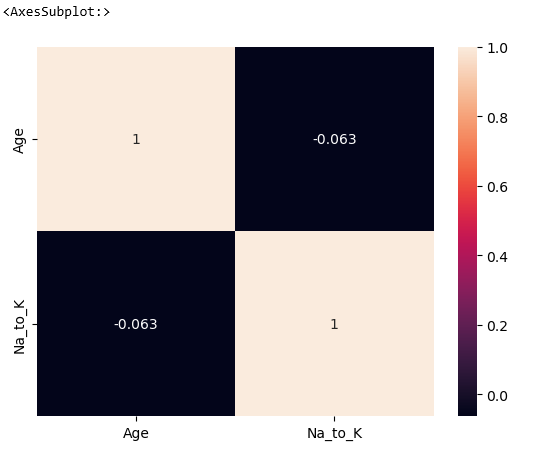
**Activity 2.1: Checking Correlation.**

Here, I'm finding the correlation using HeatMap. It visualizes the data in 2-D coloured maps making use of colour variations. It describes the related variables in the form of colours instead of numbers; it will be plotted on both axes.

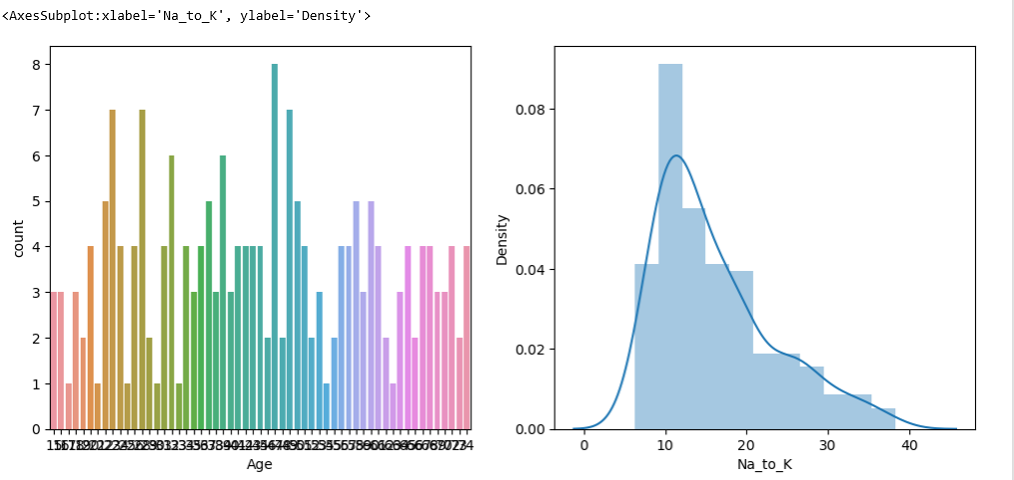
Here, there is no correlation between columns.

**import seaborn as sns**

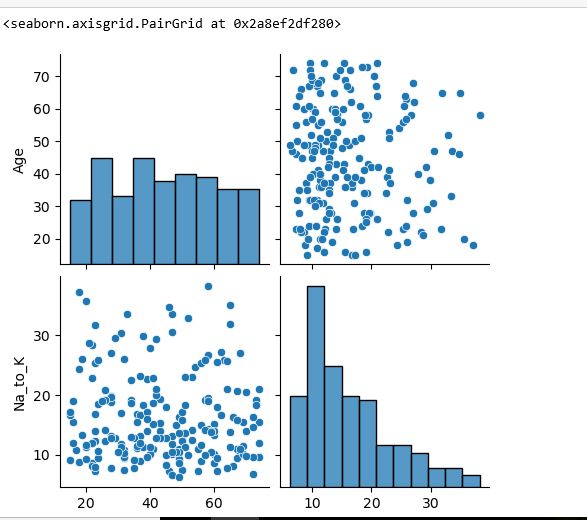
**sns.heatmap(data.corr(),annot=True)**



|  |
| --- |
| **plt.figure(figsize=(12,5))**  **plt.subplot(1,2,1)**  **sns.countplot(data['Age'])**  **plt.subplot(1,2,2)**  **sns.distplot(data['Na\_to\_K'])** |

****

**sns.pairplot(data=data,markers=["^","v"],palette="Inferno")**

****

**from sklearn.model\_selection import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_resamble,y\_resamble,test\_size=0.2,random\_state=0)**

**Milestone 4: Model Building**

**Activity 1: Training the model in multiple algorithms**

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

**Activity 1.1: Random Forest Classifier Model**

A function named Random Forest Classifier Model is created and train and test data are passed as the parameters. Inside the function, the Random Forest Classifier algorithm is initialized and training data is passed to the model with the .fit() function.

Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, accuracy\_score and classification report is done.

|  |
| --- |
| import sklearn  from sklearn.preprocessing import LabelEncoder, OneHotEncoder  from sklearn.ensample import RandomForestClassifier  rfr1=RandomForestClassifier().fit(x\_os,y\_os.values.ravel())  y\_pred=rfr1.pred(x\_test\_os) |

**Activity 1.2: XGBClassifier model**

A function named XGBClassifier model is created and train and test data are passed as the parameters. Inside the function, the XGBClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, the accuracy score and classification report is done.

|  |
| --- |
| from xgboost import XGBClassifier  xgb1=XGBClassifier()  xgb1.fit(x\_os,y\_os) |

**Activity 1.3: SVC model**

A function named SVC model is created and train and test data are passed as the parameters. Inside the function, the SVC algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, the accuracy score and classification report is done.

|  |
| --- |
| from sklearn.svm import SVC  from sklearn.metrics import accuracy\_score,classification\_report  sv=SVC  sv.fit(x\_bal,y\_bal) |

**Activity 1.4 ANN Model**

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers

**Activity 2: Testing the model**

**Milestone 5: Performance Testing & Hyperparameter Tuning**

**Activity 1: Testing model with multiple evaluation metrics**

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

**Activity 1.1: Compare the model**

For comparing the above four models, the compareModel function is defined.

**Activity 2:Comparing model accuracy before & after applying hyperparameter tuning**

From sklearn, accuracy is used to evaluate the score of the model. On the parameters, we have given xgb1 (model name), x, y, cv (as 3 folds). Our model is performing well. So, we are saving the model by pickle.dump().

**Milestone 6: Model Deployment**

**Activity 1:Save the best model**

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

Activity 2: Integrate with Web Framework In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

• Building HTML Pages

• Building server side script

**Activity 2.1: Building Html Pages:**

For this project create three HTML files namely

• home.html

• predict.html

• submit.html and save them in the templates folder.

**Milestone 7: Project Demonstration & Documentation**

Below mentioned deliverables to be submitted along with other deliverables Activity 1:- Record explanation Video for the project end to end solution

**Activity 2:- Project Documentation-Step by step project development procedure**