1. **Project overview:**

The project aims to develop a system that can detect thyroid conditions (normal, hyperfunction, subnormal) using a machine learning model based on patient data .

**->Data source:**

Primary data source:

UCI machine learning repository:

Contains thyroid dataset with features such as T3,TSH,T4U, and thyroid function

Classifications.

Format: CSV file

Secondary data source:

Research papers and studies:

Provides additional context and validation for the data used

Format: PDF or online publications

1. **Folder and Module Description**:

**Src/components/data\_ingestion/:**

**Purpose:** contains code for collection and loading data from MongoDB Atlas

Src/components/data\_transformation/:

**Purpose**: handles data cleaning,normalization, and feature engineering

Src/components/data\_validation/:

**Purpose**: Validates the integrity and structure of the dataset using schema definitions

Src/components/model\_trainer/:

**Purpose**: trains the machine learning model on the validated dataset.

Src/constant:

**Purpose**: it contains the constants which will be used throughout the project

Src/configurations:

**Purpose**: it is used to to configure the mongodb atlas cluster

Src/data\_access:

**Purpose**: to access data from mangodb cluster

Src/utils:

**Purpose**: it contains utility functions that can be used in project

Src/pipeline:

**Purpose**:it contains train\_pipeline.py -> to manage the workflow for model\_training , including data\_ingestion, transformation, validation and training

Prediction\_pipeline.py-> to manage workflow for prediction process

Logger.py:

**Purpose:** Handles logging of operations

Exception.py :

**Purpose:** Manages custom exceptions in the project

c**onfig/**

**model.yaml**: Configuration file containing model hyperparameters and other settings.

**train\_schema.json**: JSON schema for validating the dataset structure.

Notebook\_implementaion/:

**Purpose**: for data analysis

Upload\_data\_to\_db:

**Purpose**: to upload the data in mangodb cluster

**Other Files**

**Dockerfile:** Containerizes the Flask application for deployment.

**requirements.txt:** Lists the dependencies required for the project.

**setup.py:** Python package setup script.

**.env:** Stores environment variables like MongoDB credentials.

**HARDWARE AND SOFTWARE REQUIREMENTS:**

.Development Environment:

* Os: windows/Linux/Macos
* RAM : Minimum 8GB
* Storage : Minimum 50GB

.Software:

* Python 3.8
* MongoDB Atlas
* Tools and Utilities:

. vs code or pycharm: for code development

. jupyter notebook: for interactive data analysis

. git : version control system

. Docker : for containerization of the application

* Aws services:

. AWS CLI -> to interact with aws with command line interface

. AWS ECR-> to store and manage docker containers

. AWS S3 -> for model storage

. AWS app runner-> for application deployment

* Libraries/Frameworks:

.Flask(for the web application)

. sckikit-learn(for machine learning models)

. pandas (for data manipulation)

. boto3(for aws interaction)

. pymongo (for MongoDB interaction)

**FRONTEND AND BACKEND**

**Frontend:**

**Purpose:** The frontend serves as the user interface of the application, enabling users to interact with the system. For a machine learning project like thyroid detection, the frontend might include the following:

**User Interface:** A web-based interface developed using HTML, CSS..

**User Input:** Forms or fields where can input the file for which he want the result of thyroid\_detection

**Results Display:** Displays predictions about thyroid health based on user input in the file that contains the result column to show the prediction result

**BACKEND:**

* **Purpose:** The backend handles all the logic, data processing, machine learning predictions, and database interactions. It includes:
  + **Web Server:** Flask is used as the web server to handle HTTP requests and manage API routes.
  + **Data Processing:** Python scripts within the src folder handle data ingestion, transformation, validation, and model training.
  + **Machine Learning Model:** The trained machine learning model (handled by the model\_trainer module) makes predictions based on the input data.
  + **Database Interaction:** The backend interacts with MongoDB Atlas to store and retrieve data, such as patient records and prediction results.
  + **Deployment:** The backend is deployed on AWS using services like ECR, App Runner, and is integrated with other AWS services like S3 for data storage and access.