High Level Design

1.1.

Written By	Shubham Rawat
Document Version	0.1
Last Revised Date	11-04-24

Document Control

Change Record:

Version	Date	Author	Comments
0.1	11-04-2024	Shubham Rawat	Initial HLD

Contents

Abstract

Thyroid disease prediction has advanced significantly with the integration of machine learning techniques. Researchers have developed various models using algorithms like logistic regression, decision trees, and k-nearest neighbors to accurately predict thyroid disorders.

These models are trained on large datasets and can assist healthcare professionals in early diagnosis and treatment planning. The use of such predictive tools is a testament to the ongoing innovation in medical technology, aiming to improve patient outcomes and streamline diagnostic processes.

1. Introduction

1.1. What is High-Level design document?

The goal of HLD or a high-level design document (HLDD) is to give the internal logical design of the actual program code for Adult Census Income System. HLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

The HLD document presents the structure of the system, such as the database architecture, application. Architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2. General Description

2.1. Product Perspective

To Build a Thyroid Disease Prediction Model which will help in identifying the Possibility of Disease

2.2. Problem Statement

The goal of thyroid disease prediction is to accurately identify the presence of thyroid dysfunction using various machine learning techniques. By analyzing patterns in patient data, including hormone levels and clinical symptoms, the aim is to develop a model that can predict thyroid disorders, aiding in early diagnosis and effective treatment. This predictive analysis is crucial for timely intervention and management of thyroid-related health issues.

2.3. Proposed Solution

In this approach, we identify users who have similar preferences or behaviors.

Suppose User A and User B have similar tastes in movies. If User A has liked certain films, we can recommend those films to User B.

The system looks for patterns in user interactions (ratings, views, etc.) to make recommendations.

2.4. Further Improvements

We can Store our data into some database like MongoDB or MySql. Currently our data is running dynamically stored into Ram. We Can use some Cloud Platform like AWS to store our trained model so that we can use it again and again. UI can be improved as I don’t have much UI skill (CSS). User to Software connection can be improved.

2.5. Tools Used

- PyCharm is used as IDE.
- For Visualization Matplotlib, Seaborn Libraries used
- Heroku for Deployment
- Front end development is done using HTML/CSS
- Flask is used back-end development
- GitHub is used as version control system

2.6. Constraints

Its Fully Automated so User doesn't need to know any prior knowledge before using it.

Just Fill the values and Predict the result.

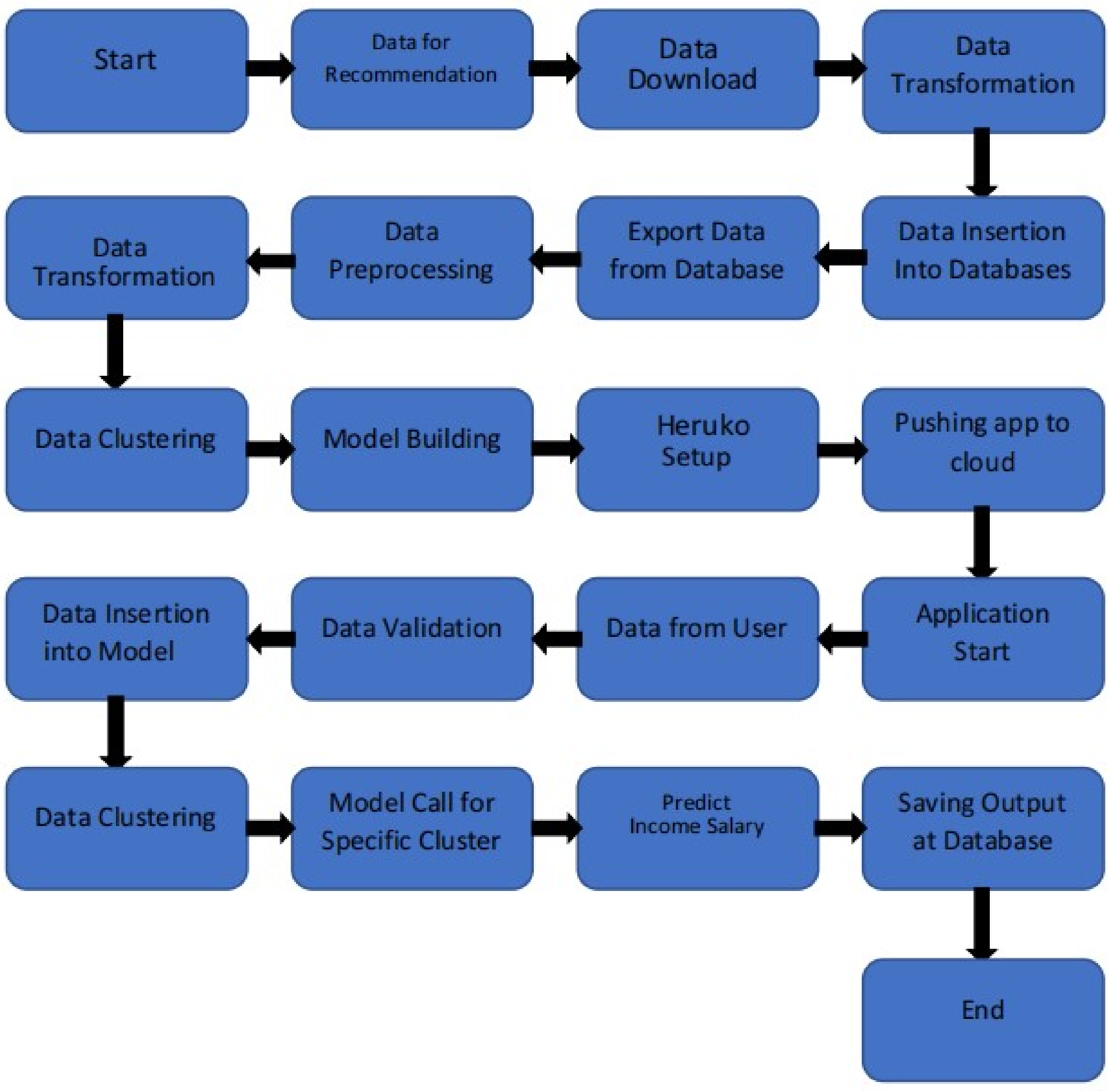
2.7. Assumptions

User are authentic and books review are genuine.

3. Design Details

3.1. Process Flow

2. Architecture



3.2. Event log

The system should log event so that the user will know what process is running internally.

1. The System identifies at what step logging required
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3. Error Handling

We have Applied Standard Processor to Prevent Error. Try and Exception is used to prevent error in system. Every error message is stored into Log file.

4. Performance

4.1. Re-usability

The Code is Written in the modular format so ıt is easy to understand and modified.We don’t need to modified full code instead we can only touch segment of code.

4.2. Application Compatibility

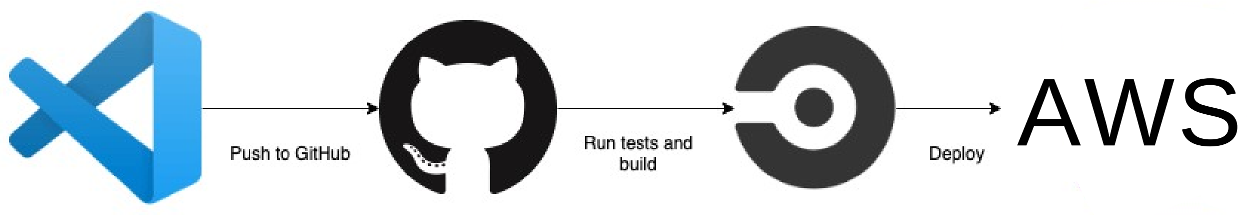
The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

4.3. Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

4.4. Deployment

It can be deploy over any cloud platform.It is deploy over AWS Continuous Integration and Deployment (CI/CD) pipelines using Git as a single source of truth.



We used Docker to deploy our flask application. Docker provides the flexibility to ship the application easily to cloud platforms.

5. Conclusion