ARCHITECTURE

Thyroid Disease Detection System

Written by- Santhoshkumar

- Document Version Control
- Change Record:

VERSION	DATE	AUTHOR	COMMENTS
1.0	4-07-2024	Santhoshkumar	Architecuture

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

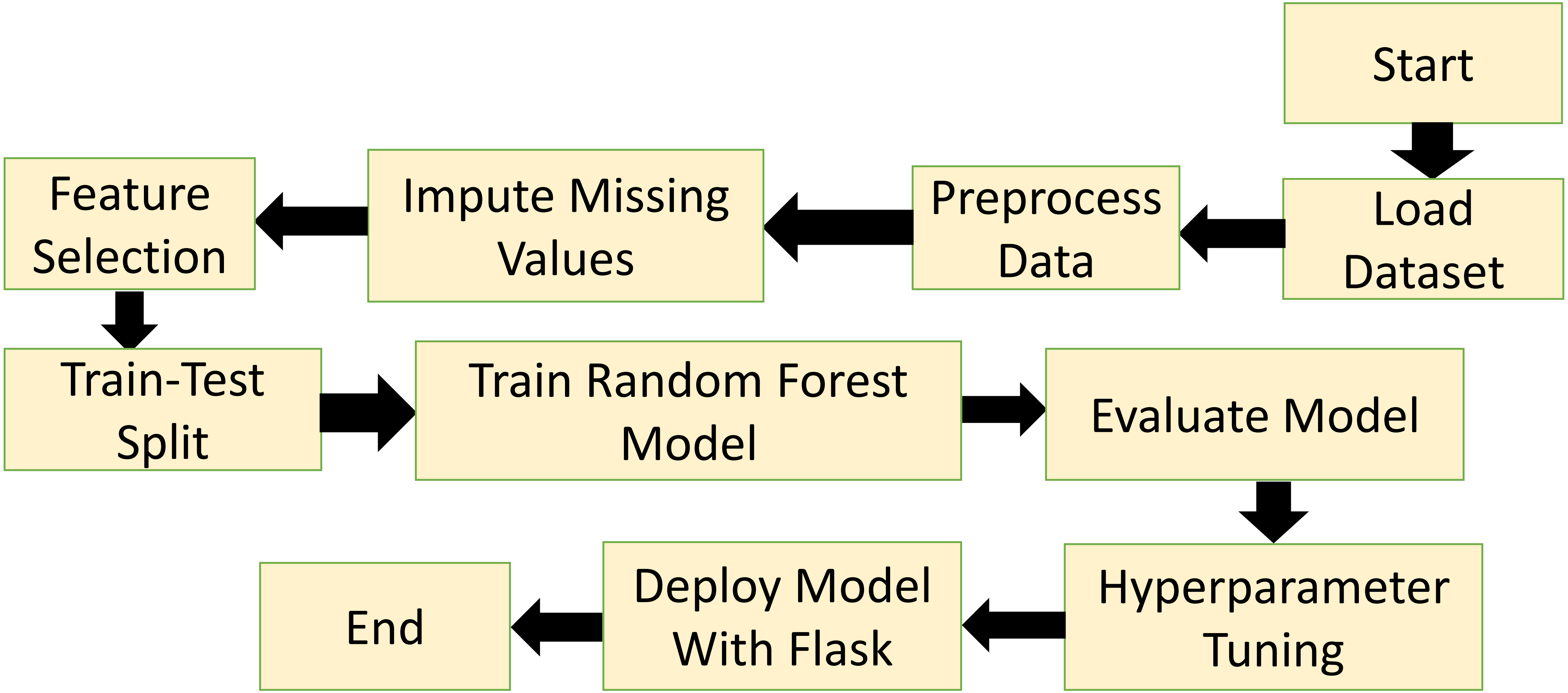
Version	Review Date	Aoorived By	Comments

contents

Introduction

- This document provides an in-depth look at the architecture of the thyroid prediction project. It outlines the system architecture, data flow, and detailed descriptions of each component and module.

Architecture



Components and Modules

- Each component and module in the system plays a crucial role in ensuring accurate and efficient predictions. Here are detailed explanations of each:

Data Collection:

- Data for this project is collected from a reputable source, which includes various medical records and features pertinent to thyroid diseases. The dataset is comprehensive and includes features such as TSH, T3, T4U, and FTI.

Data Preprocessing

- Preprocessing is crucial to ensure the data is clean and suitable for training the model.
- **Impute Missing Values:** Handle missing values using methods like mean, median, or mode imputation.
- **Normalize Data:** Scale the features to ensure uniformity.
- **Encode Categorical Variables:** Convert categorical variables into numerical ones using techniques like one-hot encoding.

Feature Selection

- Feature selection involves choosing the most relevant features for training the model. Techniques like correlation analysis and feature importance from models like Random Forest are used to select these features.

Model Training

- A Random Forest model is trained on the preprocessed data. Random Forest is chosen due to its robustness and high accuracy in classification problems. The training involves:

Model Training:

- Splitting the data into training and testing sets.
- Fitting the model on the training data.
- Using cross-validation to ensure the model is not overfitting.

Model Evaluation:

- he trained model is evaluated on the testing dataset using metrics like accuracy, precision, recall, and F1-score. The evaluation helps in understanding the model’s performance and areas of improvement.

Hyperparameter Tuning:

- Hyperparameter tuning is performed to optimize the model’s performance. Techniques like Grid Search and Random Search are used to find the best combination of hyperparameters.

Model Saving:

- The final model, along with the preprocessing steps, is saved using Python’s pickle module. This saved model is then used for making predictions in the Flask application.

Deployment with Flask

- Flask is used to create a web application that serves the predictive model. The steps include:
- Creating Flask routes for different functionalities.
- Loading the saved model.
- Setting up a user interface for input and displaying predictions.

Application Workflow

- The overall workflow of the application includes:
- User input data through the web interface.
- Data is preprocessed using the same steps as during training.
- The preprocessed data is fed to the model for prediction.
- The prediction result is displayed to the user and stored in the Cassandra database.

Predictions and Results

- The application provides accurate predictions based on the input data. The results are stored and can be retrieved for further analysis.

Conclusion:

The architecture of the thyroid prediction project integrates machine learning, web development, and database management to provide a robust and scalable solution for thyroid disease prediction.