Structures Energy Efficiency Project Architecture

Overview

The Structures Energy Efficiency project aims to develop a machine learning model for predicting heating and cooling loads in residential buildings. The architecture is designed to facilitate data processing, model training, deployment, and continuous integration/continuous deployment (CI/CD) using AWS services and Docker containers.

Document Version Control

Version	Date	Author	Comments
1.1	14/02/2024	Hasham Akram	

Content

Abstract

Machine Learning is a category of algorithms that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build models and employ algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. These models can be applied in different areas and trained to match management expectations so that proper steps can be taken to achieve the organization's target. In this project, we will estimate the Energy Efficiency based on various external and physical factors. Taking various aspects of a dataset collected from people and the methodology followed for building a predictive model.

1 Introduction

1.1 What is Architecture Design?

The goal of Architecture Design (AD) is to give the internal design of the actual program code for the `Energy Efficiency Prediction`. AD describes the class diagrams with the methods and relation between classes and program specification. It describes the modules so that the programmer can directly Code the program from the document.

1.2 Scope

Architecture Design (AD) is a component-level design process that follows a step by-step refinement process. This process can be used for designing data structures, required software, architecture, source code, and ultimately, performance algorithms. Overall, the data organization may be defined

during requirement analysis and then refined during data design work. And the complete workflow.

1.3 Constraints

Constraints related to overfitting (model capturing noise instead of underlying patterns) or underfitting (model oversimplified to capture complex patterns)could impact model generalization ability.

2. Architecture Components

2.1 Data Collection and Ingestion

- Data Source: Energy Efficiency Dataset Download or you can gather data from various sources such as databases, APIs, files, or streaming platforms.
- Data Ingestion: Data is downloaded and stored in a Cassandra database. Python scripts integrate with the Cassandra database to import data from a remote repository into the local machine. Preprocessing scripts then prepare the data for model training.
- Splitting: Making storage directories for train, test and original dataset.

2.2 Data Preprocessing:

- Data Cleaning: Handle missing values, duplicate records, outliers and renaming the columns.
- Feature Engineering: Create new features, drop least relevant, encode categorical variables, and scale numerical features.
- Data Transformation: Normalize or standardize features, perform dimensionality reduction using Standard scaling fit_transformation.
- Saving paths: Accessing and saving the preprocessed file paths for later use in model training.

2.3 Model Training

- Data Splitting: Prepare and split the dataset into training and testing sets.
- Model Initialization: Initialize multiple regression models such as Random Forest, Gradient Boosting, and XGBoost.
- Model Evaluation: Evaluate each model's performance using metrics like R2 score.Use techniques like grid search or random search to find the best combination of hyperparameters.
- Hyperparameter Tuning: Fine-tune the selected model's hyperparameters for optimal performance.
- Validation and Threshold Check: Validate the model's performance on testing data and ensure it meets the desired threshold (e.g., R2 score > 0.8).
- Save Best Model: Save the best-performing model for future use in making predictions during deployment.

2.4 Github

- Pushing your trained model with relevant directories and files into github repository through Git CLI.

2.5 Model Deployment:

- Containerization: Containerization: Package the trained models along with necessary dependencies into Docker containers to ensure consistency and portability across different environments. This allows for easy deployment and reproducibility.

- Scalability: Deploy the containerized models on scalable platforms like AWS ECS (Elastic Container Service) or Kubernetes. These platforms provide auto-scaling capabilities to handle increased workloads efficiently.
- API Development: Expose the trained models as RESTful APIs using a lightweight framework like Flask or FastAPI. This allows users to interact with the models easily over HTTP requests, enabling integration with other systems or applications.
- Cloud Deployment: Deploy the containerized models on cloud platforms such as AWS, leveraging services like Amazon ECR (Elastic Container Registry) for storing Docker images and AWS Lambda for serverless execution. This approach ensures easy scalability, reliability, and management of the deployed models.

2.6 User Interface (Optional):

- Dashboard: Develop interactive dashboards or web interfaces for users to interact with the model predictions.

2.7 Documentation and Reporting:

- Documentation: Document the entire process including data sources, preprocessing steps, model development, and deployment.
- Reporting: Generate reports summarizing model performance, insights, and recommendations for stakeholders.

3. Technology Stack

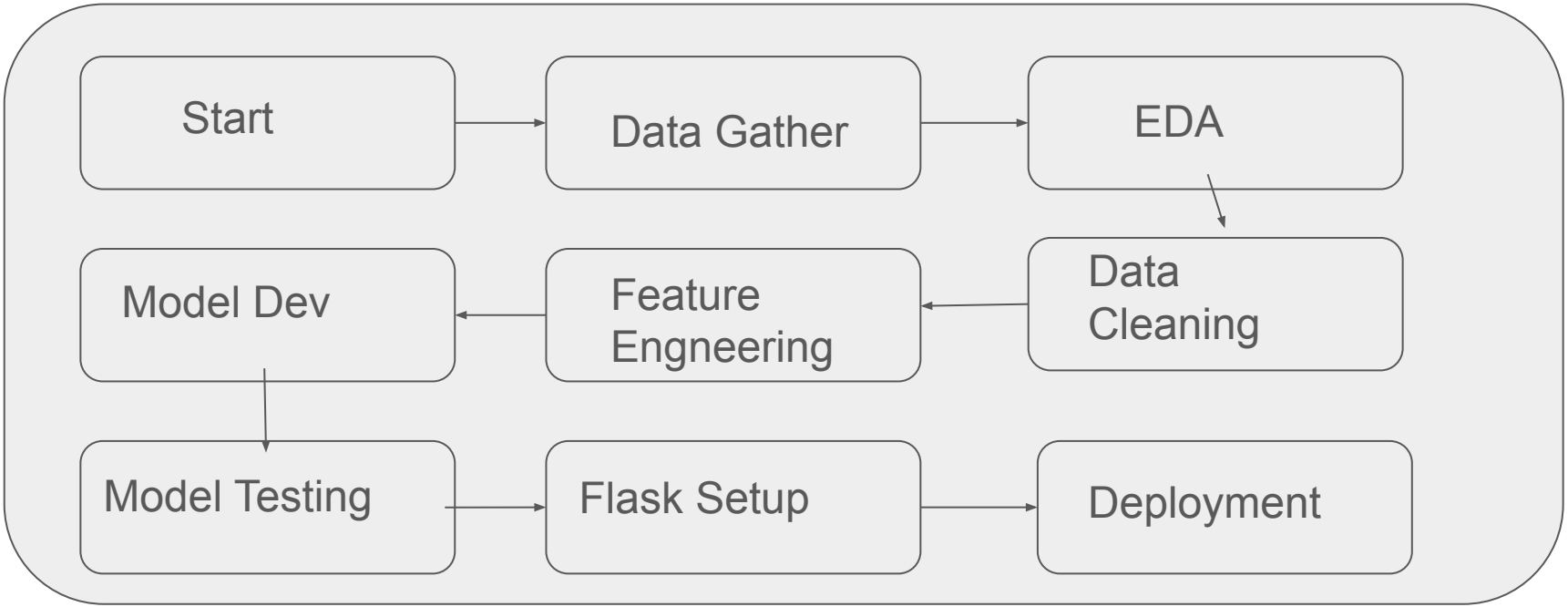
Structure and Framework

Front End	HTML/CSS
Backend	Python/Flask
Deployment	AWS/EC2

4. Proposed Solution

We will use performed EDA to find the important relation between different attributes and will use a machine-learning algorithm to estimate the cost of expenses. The client will be filled the required feature as input and will get results through the web application. The system will get features and it will be passed into the backend where the features will be validated and preprocessed and then it will be passed to a hyper parameter tuned machine learning model to predict the final outcome

5. Architecture



6.User I/O Workflow