Metro Interstate Traffic Volume Prediction System

Architecture Documentation

- ***Fazal Rehman***

Contents

Abstract

The Metro Interstate Traffic Volume Prediction System aims to build a machine learning model that can accurately predict traffic volume on interstate highways based on various factors such as weather conditions, time, and holidays. By forecasting traffic volume, city planners and individuals can make informed decisions to reduce congestion and improve commuting experiences. The dataset used for this project has been sourced from the UCI Machine Learning Repository and contains traffic volume recorded hourly, along with corresponding weather and time features.

Introduction

What is Architecture Design?

Architectural design is the process of defining a structured solution that meets technical and operational requirements while optimizing common quality attributes like performance, security, and manageability. For this system, architectural design lays out how data flows through different system components such as frontend, backend, ML models, and deployment pipelines.

Scope

This architecture design document describes the Metro Interstate Traffic Volume Prediction System, covering its data processing, modelling, prediction workflow, user interaction, and final deployment on a local server environment.

Constraints

- Predictions are dependent on historical data trends and may slightly vary during unseen extreme conditions.
- Deployment is on a localhost environment.
- Logs are maintained for debugging and operational transparency.

Technical Specifications

Dataset

- **Source:** UCI Machine Learning Repository
- **Description:** Contains hourly traffic volume data along with corresponding weather conditions, time details (hour, weekday, month), holidays, etc.
- **Format:** CSV file, structured tabular data.

Logging

- Every prediction request is logged.
- Errors and system activities are logged for easier debugging.
- Logging does not impact real-time performance.

Deployment

- Deployed locally using the Flask framework.
- The prediction model is stored as a pickle (.pkl) file.
- Flask routes handle user form submissions and model predictions.

Architecture

Data Gathering

- Download the dataset from the UCI Repository.

Data Cleaning

- Handle missing values using SimpleImputer.
- Standardize numeric columns.

Exploratory Data Analysis

- Analyse correlations.
- Identify significant features.

Feature Engineering

- Encoding for categorical variables using OrdinalEncoder .
- Scaling for numerical variables using StandardScaler.

Model Selection

- Train multiple models like Random Forest, CatBoost, etc.
- Select the model based on the R2 score.

Model Evaluation

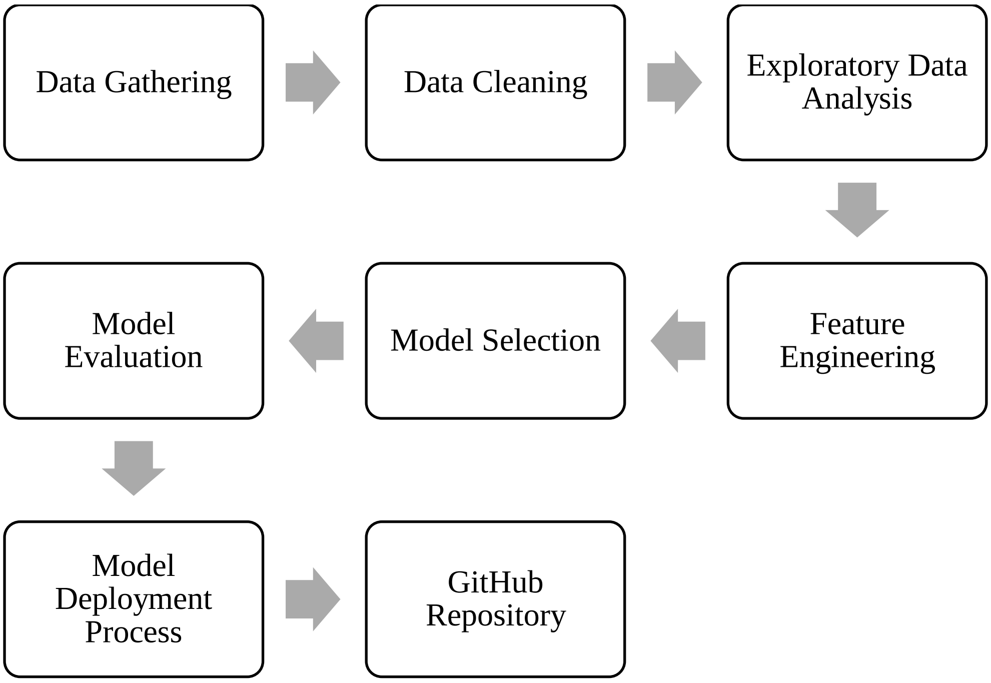
- Evaluate selected model with R2-score.

Model Deployment Process

- Save trained model as a .pkl file.
- Build a Flask server for model interaction.
- Serve prediction results through frontend forms.

GitHub Repository

- Push the entire project, including model, code, and documentation, to GitHub.



User Input-Output Workflow

Start

- The user opens the application.

User Input

- User enters weather, time, and holiday details.

Submit Details

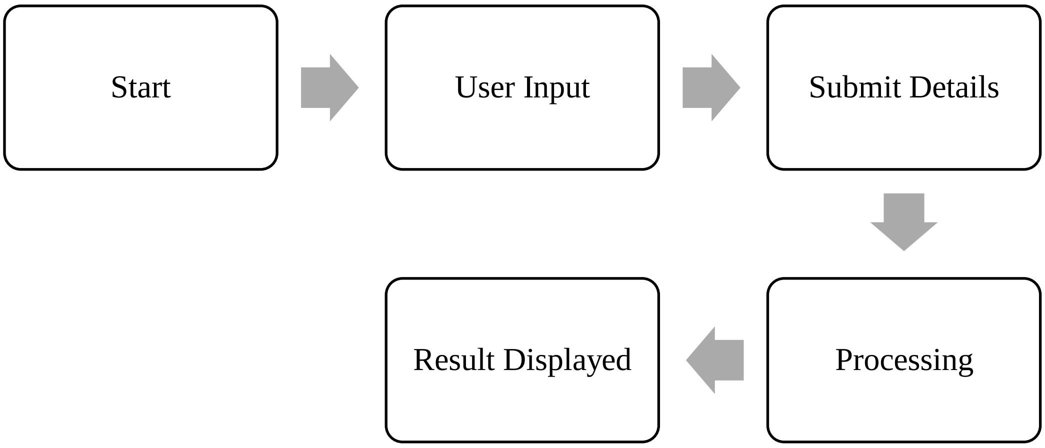
- User submits the form.

Processing

- Backend processes input and predict traffic volume.

Result Displayed

- The prediction result is shown to the user.


Start
User Input
Submit Details
Processing
Result Displayed