
Vehicle Fuel Prediction Pipeline

PROJECT DOCUMENTATION

August 25, 2023
Shalini Priya

1 Introduction

The goal of a prediction pipeline is to streamline and automate the process of making predictions from the data retrieved from the InfluxDB while ensuring consistency and reproducibility. This project focusses on retrieving data for only one IoT device.

1.1 Dataset

The following dataset has been taken for the machine learning model to make predictions.

- **Timestamp** : The timestamp indicates the date and time when the fuel reading was taken
- **Vehicle Id** : It represents unique id of each vehicle
- **Fuel Reading** : This is the measurement of the amount of fuel present in the vehicle's fuel tank at the time of reading
- **Distance Travelled** : This feature represents the distance the vehicle has traveled since the last reading. Longer distances requires more fuel
- **Speed** : speed refers to the rate at which the vehicle is moving. It can impact fuel consumption

1.2 Features of the Project

- Retrieves data from the database based on a predefined schedule. Data from each IoT device is collected by querying the database using specific measurements, tags, and buckets corresponding to each device. This allows for organized and periodic data retrieval for further analysis and processing
- Data is then preprocessed and transformed into the data which can be used by different models
- Important features are extracted based on the machine learning model being used
- The data is then split into train, test and validation set for model training , measuring the performance and predictions
- This result is then showed on the Flask UI based on the input from the user

2 Architecture

The architecture of a prediction pipeline involves multiple components working together to perform data processing, model training, and prediction. Here's a high-level overview of the architecture **Figure 4** shows the flow of data and interactions among the components

- **Data Extraction** : Data is extracted from the database using queries
- **Preprocessing and Feature Extraction**: Data is preprocessed to get the clean data and relevant features are extracted
- **Model Training**: Preprocessed data is split into train data, test data and validation data, two models are then trained using this training data. Trained model is then stored in the project path
- **Prediction**: Prediction result is generated using the test data
- **Flask UI** Based on the selected model, result is displayed on the UI
- **Retraining** New data retrieved from the database (based on scheduler) follows the same process from data extraction to prediction of results

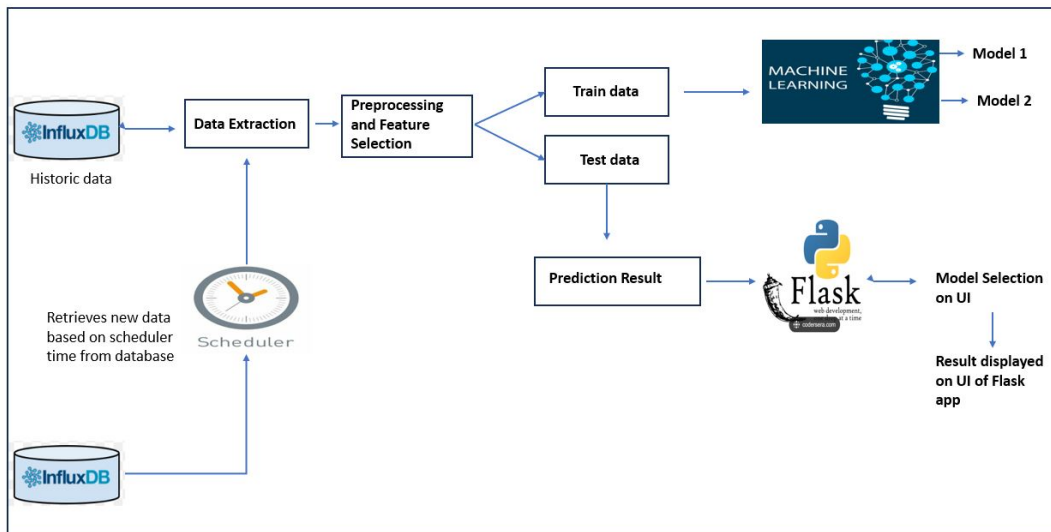


FIGURE 1: Project Architecture

3 Project Structure

Figure 2 shows the structure of the project which has been developed in PyCharm IDE. It has following components:

- **app**: app directory contains Flask UI code which accepts model type from a drop down and predicts result based on the selection. It contains related html and css files
- **data** : this directory contains code to retrieve data from the InfluxDB, scheduler which retrieves new data based on time paramer and stores new data retrieved from the database for further steps
- **models**: it holds the implementations of machine learning models xgboost and linear regression

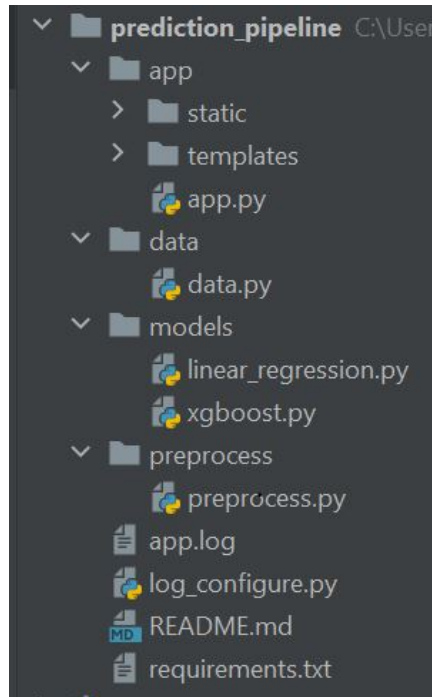


FIGURE 2: Project Structure

- **preprocess:** provides functions for data preprocessing tasks like cleaning, normalization, and encoding along with feature selection
- **README.md:** documentation that provides an overview of the project,
- **requirements.txt** file consisting of libraries that were used in the project with relevant version
- **log_config.py:** sets up the logging configuration for better troubleshooting
- **app.log:** provides the logging details for further investigation into the issues faced during running the project

4 Instruction

4.1 Instruction

- **Set Up Environment** Open a terminal and navigate to the project directory. Run the following command to install the necessary dependencies:

```
pip install -r requirements.txt
```

- **Database Query:** Modify query based on project needs
- **Execute model:** Run the files: `xgboost.py` and `liner_regression.py` for storing the respective prediction results

- **View Predictions:** Run app.py, go to the `http://127.0.0.1:5000`, select model from the drop down and click on predict to view the prediction result

Troubleshooting: If any issue arise during pipeline's execution , check the log app.log for detailed information

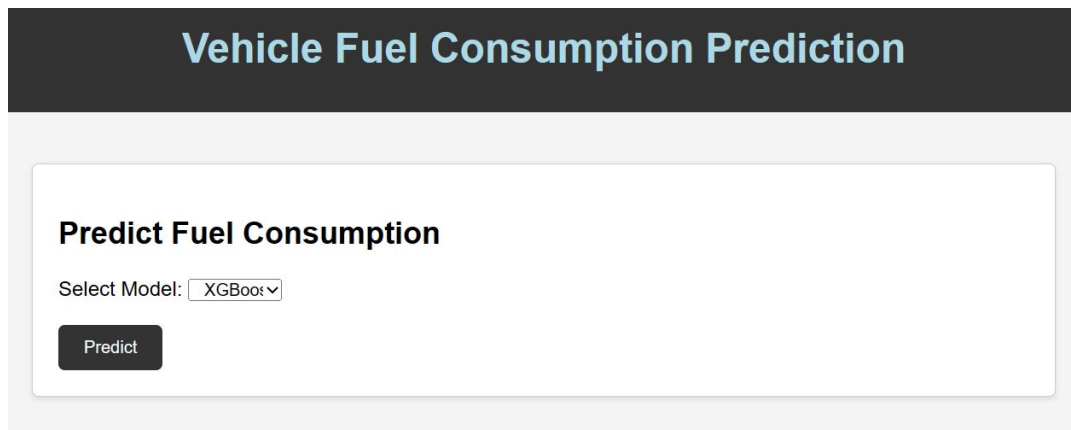
5 Factors to Consider Regarding Data

- **Data Source:** This could be a database, a data stream, webservice or any source that provides the input data for predictions
- **Data Formats:** The data could be in different formats and types(structured and unstructured)
- **Data Volume:** Analyze the volume of data available for training the model or data streaming from different types of sources
- **Data Frequency:** Data pipelines should be able to handle high frequency data streams
- **Data Quality:** Data can come with noises so we need to make sure that our data is clean and free of errors
- **Data Security and Privacy:** To protect sensitive information
- **Data Drift:** We should monitor the changes seen in data distribution or patterns and implement strategies to adapt the model to these data distributions
- **Balanced data:** There should be a balance between using historical data for model training and incremental learning with new data coming in continuously
- **Data Dependencies:** Identify relation and dependencies across the data
- **Feature Selection :** Choose relevant features that contribute to the predictive power of the model or engineer new features to capture valuable information from the data

6 Limitations

- Not all scenarios of the data pipeline has been covered due to the simulated data in use
- Data from only one IoT device has been taken into account to showcase the data flow

7 Output



The screenshot shows a web application titled "Vehicle Fuel Consumption Prediction". Below the title is a form with the heading "Predict Fuel Consumption". Inside the form, there is a label "Select Model:" followed by a dropdown menu showing "XGBoost" with a downward arrow. Below the dropdown is a dark button labeled "Predict".

FIGURE 3: Flask UI

Prediction Results

Prediction
61.474754

[Back to Home](#)

FIGURE 4: Prediction Result