# Low Level Design (HLD)

# PETROL PRICE FORECASTING

Revision Number: 2.0

Last date of revision: 22nd June 2022

VIJAYA

CHANDAN KUMAR

# Contents

## Document Version Control

| Date Issued | Version | Description | Author |
|---|---|---|---|
| 14-06-2022 | 1.0 | First Version of Complete HLD | Vijaya |
| 22-06-2022 | 2.0 | First Version of Complete HLD | Chandan Kumar |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# 1 Introduction

## 1.1 Why this Low-Level Design Document?

The goal of the Low-level design document (LLDD) is to give the internal logic design of the actual program code for the Food Sales Analysis dashboard. LLDD describes the class diagrams with the methods and relations between classes and programs specs. It describes the modules so that the programmer can directly code the program from the document.

## 1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step- by-step refinement process. The 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.

## 1.3 Project Introduction

Oil prices are determined by global supply and demand, rather than any country's domestic production level. There have been a number of structural drivers of global oil prices historically, including oil supply, demand, and storage shocks, and shocks to global economic growth affecting oil prices. Forecasting of the prices of petrol is important for Business decisions like airlines, auto manufacturers, utilities, Predicting carbon emissions and climate change ,Designing regulatory policies such as fuel standards and petrol taxes.

This study forecast the prices of Petrol for future dates. Different Regression algorithms have been tested and compared to predict the better outcome of the model. Also, MLOPs on the cloud gives the leverage of carrying the automation of ML projects on a various cloud platforms.

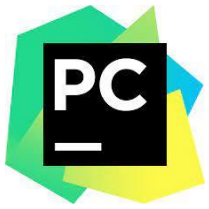
# 2 General Description

## 2.1 Problem Statement & Product Perspective

The dataset contains price information supplied on a weekly basis from 9$^{th}$June 2003 to 31$^{st}$December 2018, since it is a time series data. The main goal is to forecast crude oil prices for the following years and months. Multiple services provided by a cloud provider will be used to trigger the Continuous Integration, Continuous Deployment including Continuous training of the model when any drift in data or pattern is observed.

## 2.2 Tools used

- Python programming language.
- Libraries such as Pandas, Numpy, Scikit-Learn, Matplotlib
- Framework: Flask
- IDE: Jupyter Notebook, PyCharm.
- Cloud service: Heroku
- CI/CD pipeline : Circleci
- Git and Github.




- Pycharm and Jupyter notebook is used as IDE.
- For Visualization of the plots Matplotlib is used.
- Heroku is used for deployment of the model.
- Front-end development is done using HTML/CSS.
- Python Flask is used for backend development.
- Circleci is used for CI/CD pipeline.
- GitHub is used for version control system

## 2.3 Constraints

MLOPs on the cloud must be fully automated in consideration of continuous integration, continuous deployment with retraining approach of model, and archiving the data over time. Users can easily use the application and not needed to know any of the workings.

# 3 Dataset Information

- The dataset consists of 813 rows and 2 columns.
- Date – The date on which petrol price is recorded, given on a weekly basis from June 2003 to 31st December 2018.
- Petrol (USD) – Petrol price in USD.

| | Date | Petrol (USD) |
|---|---|---|
| 0 | 6/9/2003 | 74.59 |
| 1 | 6/16/2003 | 74.47 |
| 2 | 6/23/2003 | 74.42 |
| 3 | 6/30/2003 | 74.35 |
| 4 | 7/7/2003 | 74.28 |

# 4 Architecture

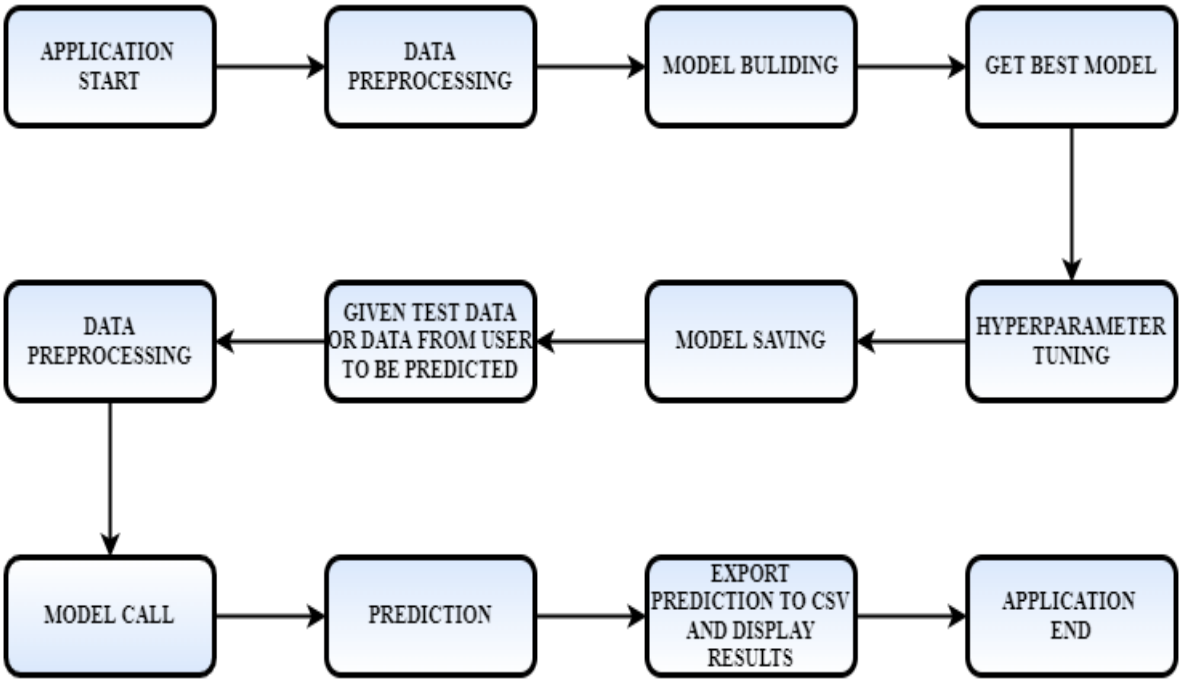



Figure 1: Process flow

## 4.1 Architecture Description

1. Raw Data Collection

   The Data set was taken from kaggle Provided Project Description Document.

2. Data Pre-Processing

   Before building any model, it is crucial to perform data pre-processing to feed the correct data to the model to learn and predict. Model performance depends on the quality of data fed to the model to train. This Process includes.

   a) Handling Null/Missing Values

   b) Outliers Detection and Removal

### 3. Data Cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a data set.

a) Remove duplicate or irrelevant observations

b) Filter unwanted outliers

c) Renaming required attributes

### 4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

### 5. Reporting

Reporting is a most important and underrated skill of a data analytic field. Because being a Data Analyst you should be good in easy and self-explanatory report because your model will be used by many stakeholders who are not from technical background. a) High Level Design Document (HLD)

b) Low Level Design Document (LLD)

c) Architecture

d) Wireframe

e) Detailed Project Report

f) Power Point Presentation

### 6. Modelling

Data Modelling is the process of analyzing the data objects and their relationship to the other objects. It is used to analyze the data requirements that are required for the business processes. The data models are created for the data to be stored in a database. The Data Model's main focus is on what data is needed and how we have to organize data rather than what operations we have to perform.

### 7. Deployment

The final model is deployed on Heroku by dockerizing using dockerhub, and using the tool Circleci.

## 4.2 Error Handling

The system should identify the errors encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

### 4.3 Optimization

Data strategy derives performance:

1. Filling missing values.
2. Replacing outliers.
3. Creating new features from datetime feature.
4. Removing correlated features by checking VIF score.
5. Validating score.
6. Hyperparameter tuning
7. Validating score again

## 4.4 Reusability

The code written and the components used should have the ability to be reused with no problems.

## 4.5 Application compatibility

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

### 4.6 Deployment


## 5 Conclusion

In this project, almost 5 machine learning models were built and each models gave different MAPE scores. The best among these models was the Random Forest which gave the least MAPE score of 0.87.