High 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

Abstract

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.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) document is to add the necessary detail to the project description to represent a suitable model and coding for application. This document is also intended to help detect contradictions before coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface is implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology stack. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2 General Description

2.1 Problem Statement & Product Perspective

The dataset contains price information supplied on a weekly basis from 9thJune 2003 to 31stDecember 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 Design Details

3.1 Process flow

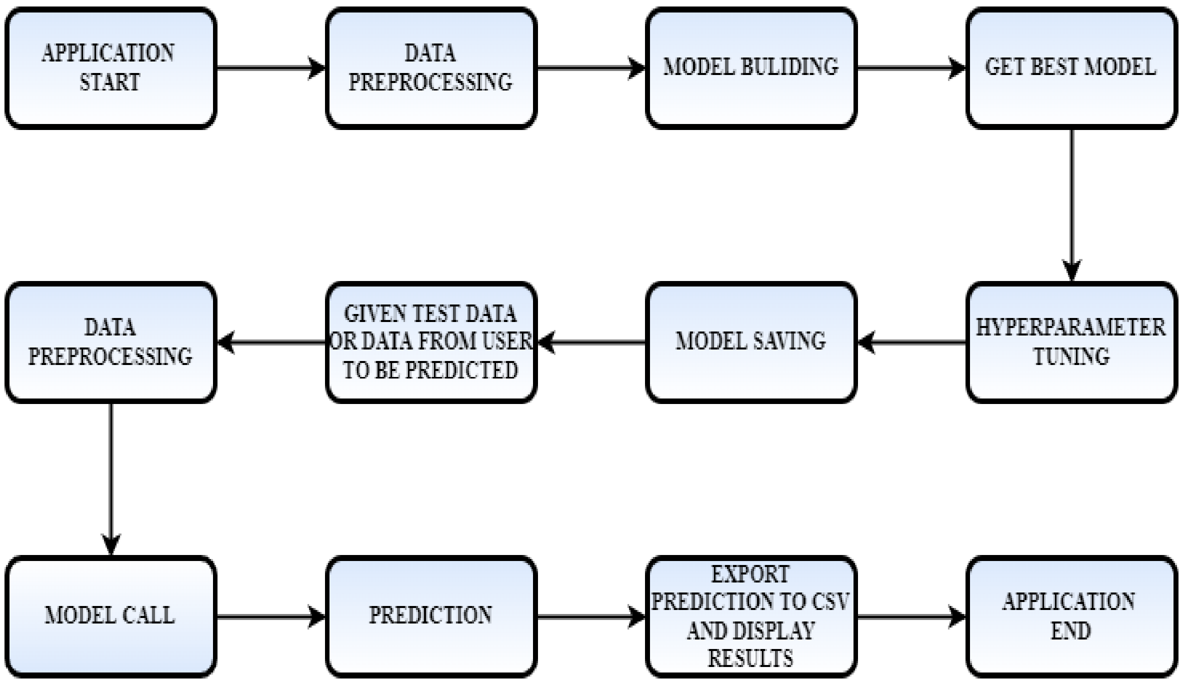

Figure 1: Process flow

3.2 Event log

The system should log every event so that the track of every detail will be known and what process is running currently could be seen.

Initial Step-By-Step Description:

1. The System identifies at what step logging is required.
2. The System should be able to log each and every system flow.
3. Developer can choose logging method. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3 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.

3.4 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

3.5 Reusability

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

3.6 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.

3.7 Deployment

4 Conclusion

In this projects, 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.