High Level Design Document:

Petrol Price Prediction

Contents

Abstract

The volatility and unpredictability of petrol prices have a significant impact on various sectors, including transportation, logistics, and consumer spending. Accurate prediction of petrol prices can provide valuable insights for businesses and individuals to make informed decisions. This project aims to develop and compare machine learning models for petrol price prediction, leveraging historical data and relevant features. In recent times the crude oil prices are significantly increased due various reasons which also impact refined fuel products. So putting this in mind this project's outcomes can have practical implications for a wide range of stakeholders, including fuel retailers, transportation companies, government bodies, and individual consumers. By leveraging machine learning techniques, this project contributes to enhancing understanding and prediction of petrol price trends, enabling better planning, resource allocation, and cost management strategies in the face of fluctuating fuel prices.

1. Introduction

1.1 Why this High-level design document?

Writing a High-Level Design (HLD) document for data science projects can be beneficial for several reasons:

- Clarity and Organization
- Communication and
- Planning and Resource Allocation
- Scalability and Modularity
- Risk Assessment and Mitigation
- Documentation and Future Maintenance

1.2 Scope

The HLD documentation presents the system overview, design architecture, dataflow, models and algorithms, training and evaluation, performance considerations, integration and deployment and many more. It's important to note that the scope of the HLD document may vary depending on the complexity and scale of the data science project. It should capture the key aspects of the project's design and provide a clear understanding of the system architecture, data flow, and the overall approach for achieving the desired outcomes.

2. General description

2.1 Project perspective

The petrol price prediction system is a machine learning based prediction model which will help in accurately predicting petrol prices for businesses, consumers, and the economy.

2.2 Problem Statement

The ONGCF is an organisation dedicated to the exploration and production of oil and natural gas. Price information is supplied on a weekly basis. It seeks to forecast crude oil prices for the following 16 months, from January 1, 2019 to April 1, 2020. The main goal is to predict the forecast the prices based upon the best model as per your choice.

It is important to acknowledge that challenges and limitations exist, such as data availability, data quality, and uncertainties in external factors that influence fuel prices. By addressing these challenges and leveraging machine learning techniques, this project aims to provide a reliable and valuable solution to the fuel price prediction problem, empowering stakeholders to make informed decisions in an uncertain market environment.

2.3 Proposed solution

Following steps represent the solution for project:

- Data collection and preprocessing
- Feature engineering
- Model development
- Model training and evaluation
- Model selection
- Future price prediction
- Visualization and reporting
- Continuous improvement

Data requirements

Data requirements are according to problem statement

2.4 Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn are used.

- Vscode is used as IDE
- For visualization of the plots, Matplotlib, Seaborn are used
- AWS is used for deployment of the model

- Front end development is done using HTML/CSS
- GitHub is used as version control system

2.5 Constraints

Constraints for a fuel price prediction project can include various factors that may impact the project's execution, such as:

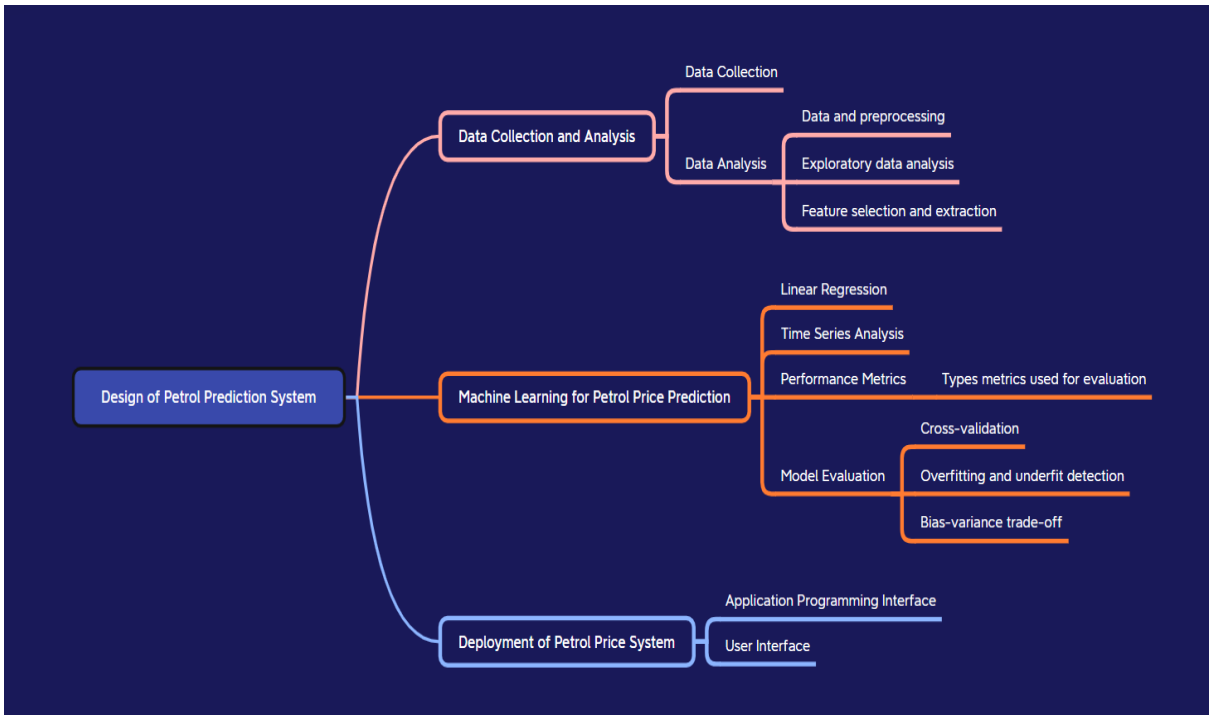
- Data Availability: Limited or incomplete data can affect the accuracy and reliability of the prediction model.
- Data Quality: Inaccurate or inconsistent data, outliers, or missing values can adversely impact the performance of the prediction model.
- Data Timeliness: Delays in data availability or outdated information may limit the model's ability to capture recent trends or changes in the fuel market.
- Resource Constraints: Delays in data availability or outdated information may limit the model's ability to capture recent trends or changes in the fuel market.
- Time Constraints: The project may have time constraints for model development, training, and evaluation.
- External Factors: Fuel prices are influenced by various external factors, such as geopolitical events, weather conditions, or unexpected market fluctuations. These factors introduce inherent uncertainties and may limit the accuracy of the prediction model.
- Model Interpretability
- Scalability: Ensuring the scalability of the solution may require architectural considerations or computational optimizations.
- Stakeholder Requirements: Aligning the project outputs with the specific needs and requirements of stakeholders can be a constraint. Balancing accuracy, interpretability, and usability according to stakeholder expectations may pose challenges.

2.6 Assumptions

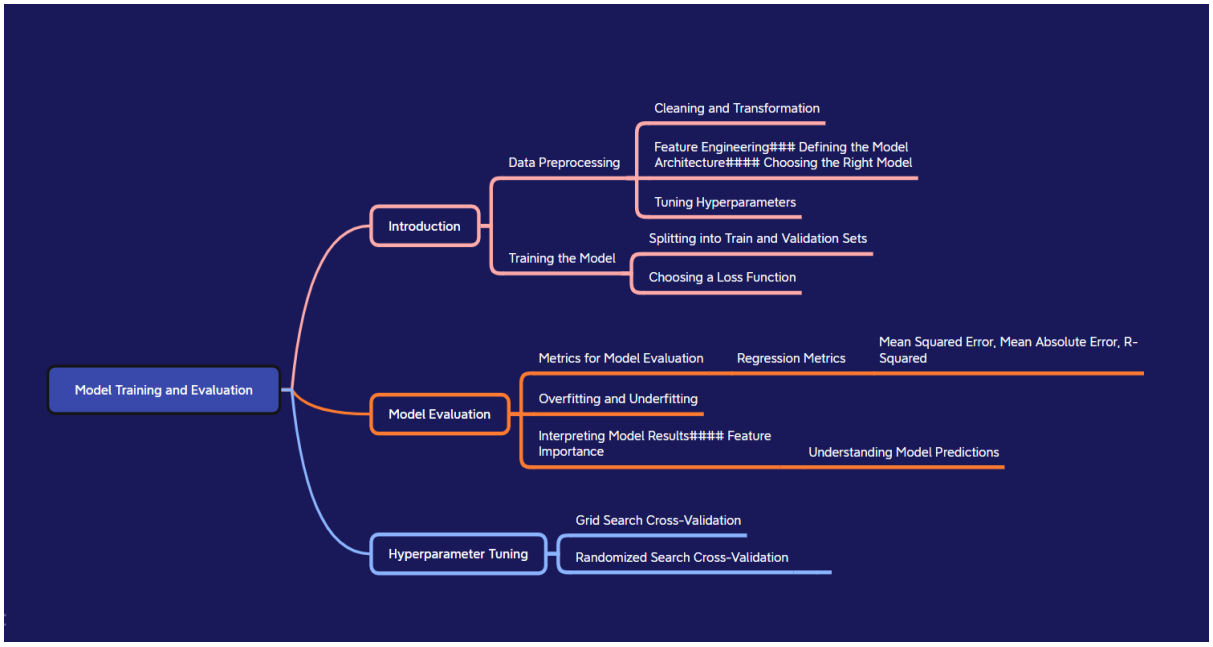
1. It is assumed that the historical fuel price data, crude oil price data, and other relevant factors used in the project are accurate and reliable.
2. It is assumed that the statistical properties of the data remain constant over time.
3. It is assumed that the necessary data for fuel price prediction, including fuel price data, crude oil price data, and other relevant factors, will be available throughout the prediction period without significant interruptions or data gaps.
4. The assumption is made that there will be no unforeseen catastrophic events or disruptions that could drastically impact fuel prices, such as natural disasters, major geopolitical conflicts, or extreme market shocks.

3. Design details

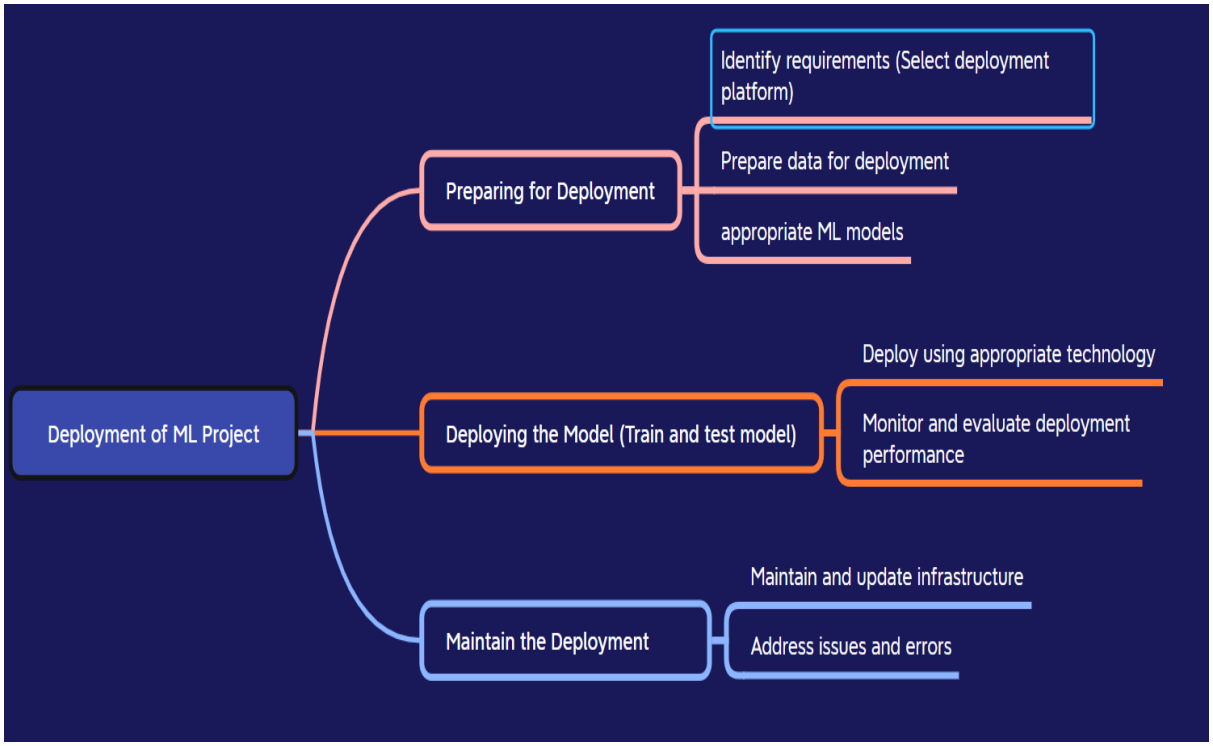
3.1 Proposed methodology:



3.2 Model training and evaluation



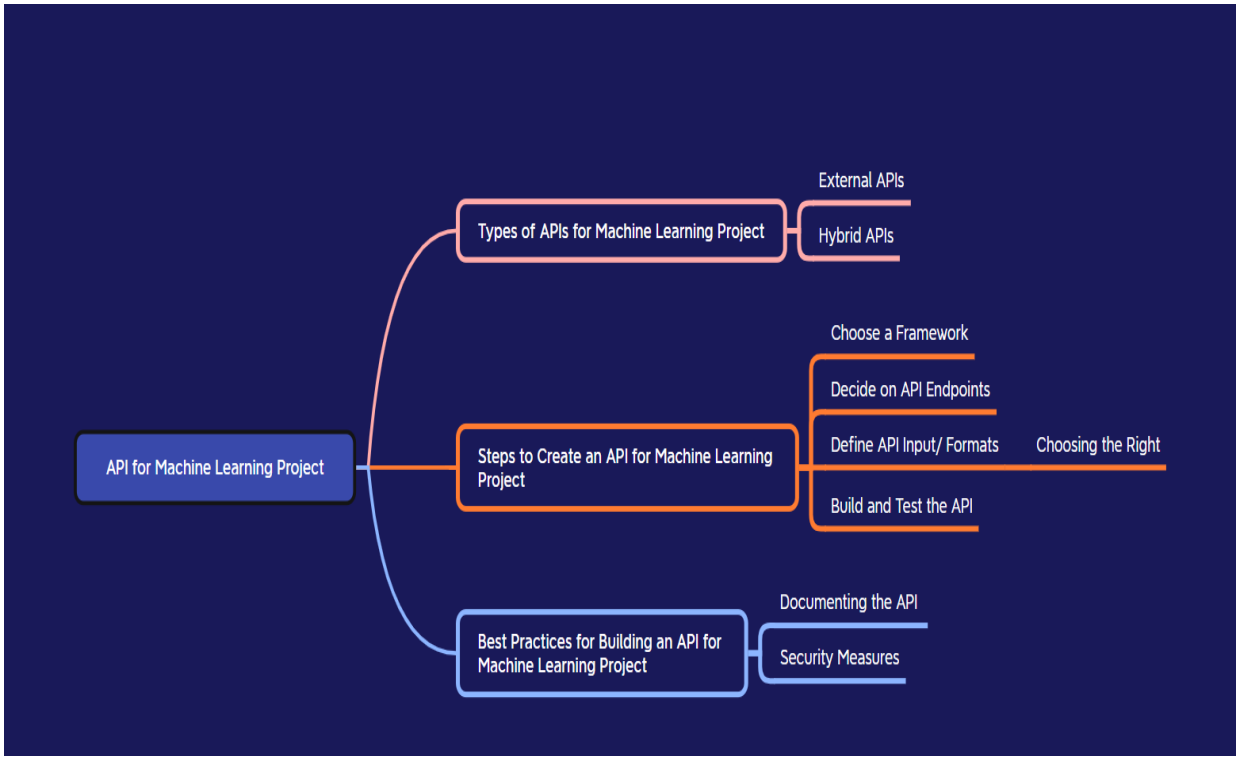
3.3 Deployment:



Apps can be used:

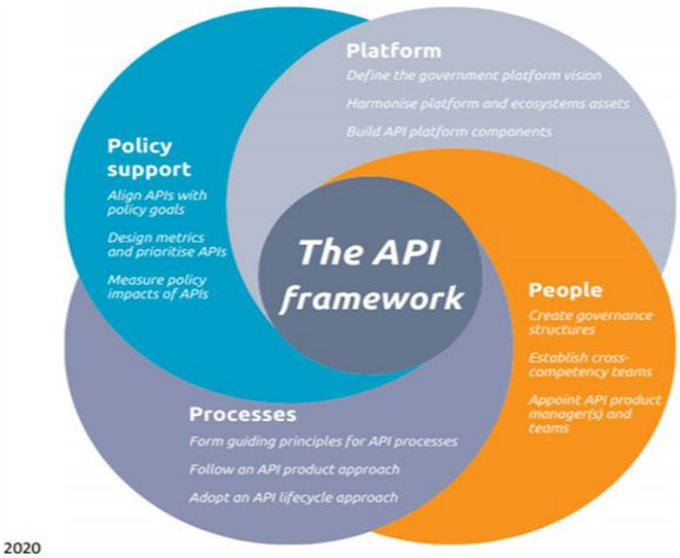
- Azure
- Amazon web services
- Google cloud

3.4 APIs (Application Programming Interface)



API Frameworks we can use:

- Flask
- Django
- FastAPI



3.4 Event logs

The system should log every event so that the user will know what process is running internally. Logging just because we can easily debug issues so logging is mandatory to do.

Steps on which log file can be created:

- Data ingestion
- Data transformation and evaluation
- Data prediction
- Completion

Logging method can be chosen by us. The System should be able to log each and every system flow.

3.6 Exception handling

Should errors be 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. Use appropriate exception handling techniques to catch and handle anticipated errors and exceptions. Implement try-catch blocks to capture exceptions and gracefully handle them. Handle different types of exceptions appropriately, considering specific error scenarios and providing informative error messages or notifications.

4. Performance Description

The performance of the fuel price prediction project can be assessed based on several key aspects:

1. Accuracy: It is evaluated by comparing the predicted fuel prices with the actual prices. Metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared can be used to quantify the model's accuracy. A lower MAE and RMSE and a higher R-squared indicate better prediction accuracy.
2. Reliability: The reliability of the prediction model(s) is determined by their consistency and stability.
3. Generalization: Cross-validation techniques are employed to assess the models' generalization ability. If the model performs well on unseen data or during cross-validation, it suggests that the model can effectively capture patterns and make accurate predictions beyond the training data.
4. Interpretability: The interpretability of the model(s) is important, particularly if stakeholders require understanding the reasoning behind the predictions. Linear regression models offer interpretability by providing coefficients for each input feature.
5. Scalability: The project's scalability is evaluated in terms of the model's ability to handle larger datasets or accommodate additional features. The model should be capable of scaling efficiently to process real-time or high-throughput prediction requests, ensuring that it can handle increased data volumes without significant performance degradation.
6. Timeliness: The model's training and prediction processes should be efficient enough to generate timely predictions, enabling stakeholders to make informed decisions and respond to market changes promptly.
7. User Satisfaction: User feedback, surveys, or usability testing can help assess stakeholders' satisfaction with the predictions and the overall project outcomes.
8. Documentation and Maintenance: Well-documented project materials enable future maintenance, knowledge transfer, and potential enhancements, ensuring the project's long-term value.

4.1 KPIs (Key performance indicators)

Key Performance Indicators (KPIs) for the Petrol Price Prediction Project can be used to measure and assess the success and effectiveness of the project. Here are some important KPIs to consider:

- Prediction Accuracy
- Model Robustness:
- Timeliness of Prediction
- User Satisfaction
- Adoption Rate
- Cost Efficiency
- Scalability
- Maintenance and Updates
- Documentation Quality
- Business Impact

5. Conclusion

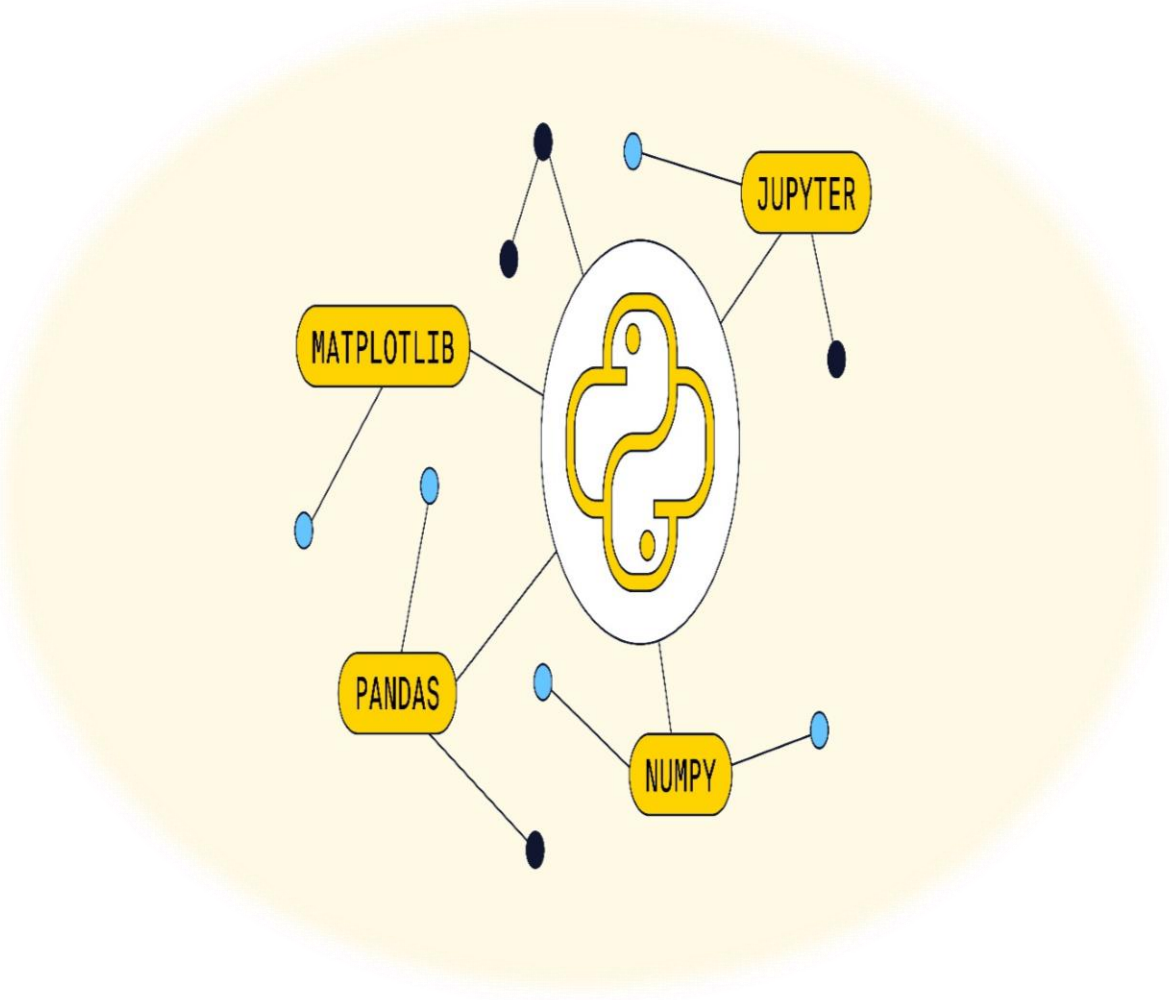
The fuel price prediction project aims to develop a machine learning solution that accurately predicts future fuel prices based on historical data and relevant factors. By successfully implementing this project, stakeholders such as fuel retailers, transportation companies, government agencies, and individual consumers can make informed decisions and plan their operations effectively in response to fuel price fluctuations. Throughout the project, key steps were undertaken, including data collection, preprocessing, feature engineering, model development, training, and evaluation. The performance of the prediction models was assessed based on accuracy, reliability, generalization, interpretability, scalability, timeliness, user satisfaction, and documentation quality.

The project's success lies in achieving high prediction accuracy, demonstrated by low mean absolute error (MAE), root mean squared error (RMSE), and high R-squared values. The reliability and generalization of the models were monitored to ensure consistent performance over time and on unseen data. The system's timeliness in generating predictions and its usability through an intuitive user interface contributed to stakeholder satisfaction and adoption. Additionally, the project accounted for constraints and assumptions, such as data availability, quality, and the influence of external factors. Documentation and maintenance practices were implemented to facilitate knowledge transfer, system maintenance, and potential enhancements in the future.

By successfully deploying the fuel price prediction system, stakeholders can access and utilize the predicted fuel prices to optimize pricing strategies, plan routes and manage costs, monitor market trends, and budget expenses effectively. The project's impact can be measured through tangible benefits realized by stakeholders, such as improved decision-making, cost optimization, and risk management in the fuel industry.

References

1. Kaggle for dataset
2. XMind Copilot AI for creating flowcharts
3. Google.com for images



Low Level Design:
Petrol Price Prediction

Contents

1. Introduction

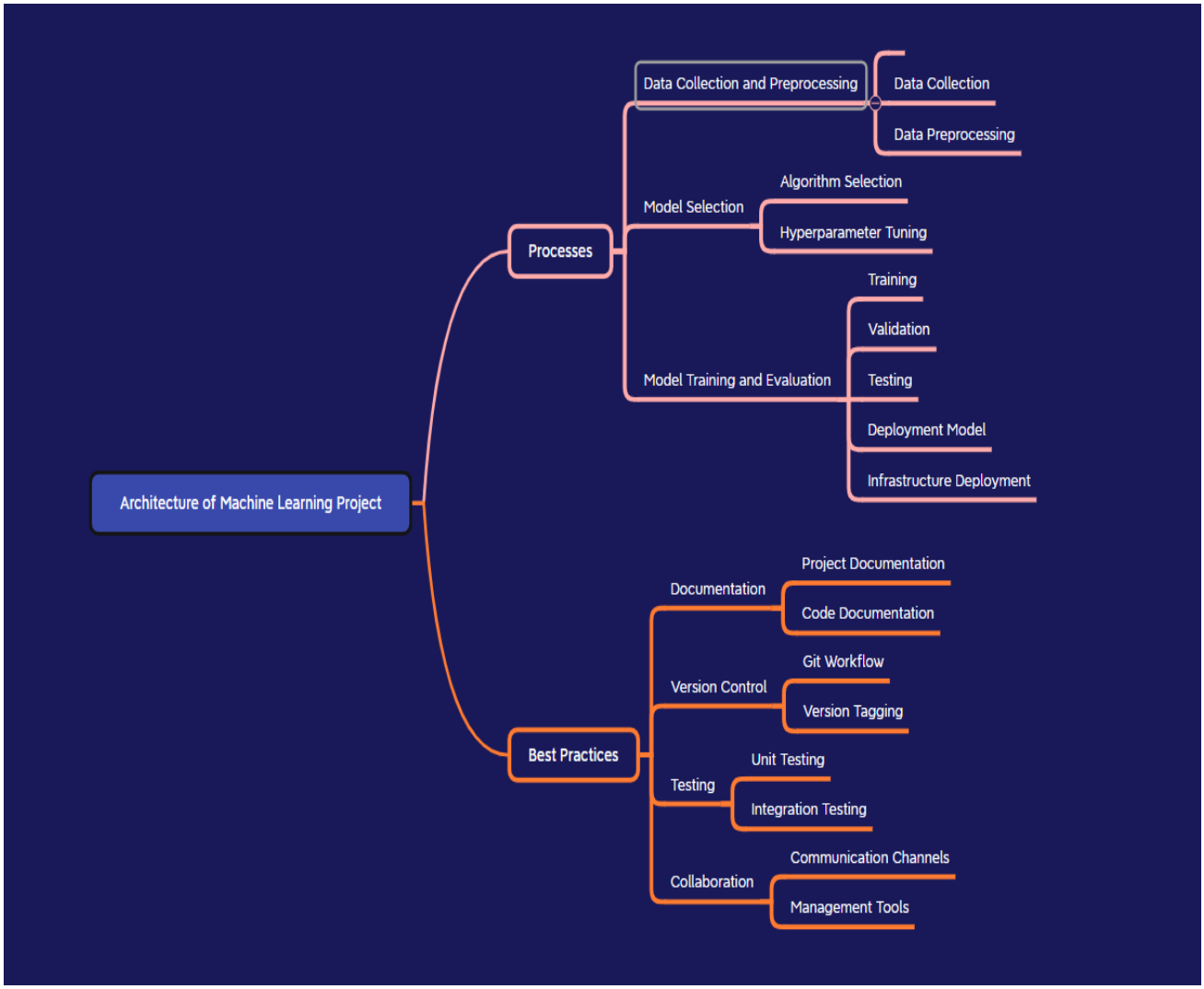
1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 Scope

The scope of the low-level design document for the Fuel Price Prediction Project encompasses the detailed design and implementation aspects of the system. It focuses on the specific components, modules, and functionalities that make up the project, providing in-depth guidance for developers and stakeholders involved in the implementation phase.

2. Architecture



2.1 Components of architecture

1. Data Collection:

- This component focuses on gathering the necessary historical fuel price data, crude oil price data, economic indicators, and other relevant data from reliable sources
- It involves identifying and accessing various data repositories, APIs, or databases to retrieve the required data.
- The collected data is then securely stored and made available for further processing.

2. Data Preprocessing:

- The data preprocessing component performs necessary tasks to clean, transform, and prepare the collected data for analysis and model training.
- It handles tasks such as handling missing values, removing outliers, normalizing or scaling data, and resolving any inconsistencies in the dataset.
- Data quality checks and validation processes are applied to ensure the integrity and reliability of the data.

3. Feature Engineering:

- The feature engineering component focuses on extracting meaningful features from the pre-processed data that can influence fuel prices.
- It involves analysing the data and identifying relevant lag variables, statistical measures, or incorporating external data sources like economic indicators, tax rates, exchange rates, or geopolitical events.

4. Model Development:

- The model development component focuses on building and training machine learning models to predict fuel prices based on the input features.
- Various models such as linear regression, support vector regression, random forest regression, or deep learning models like LSTM networks may be explored.
- Model architecture, hyperparameters, and training methodologies are defined during this phase.

5. Model Training:

- The model training component involves splitting the dataset into training and validation sets.
- The selected model(s) are trained using the training data, and hyperparameters are tuned to optimize the model's performance.
- Iterative training and validation cycles are conducted to refine the models and ensure they generalize well to unseen data.

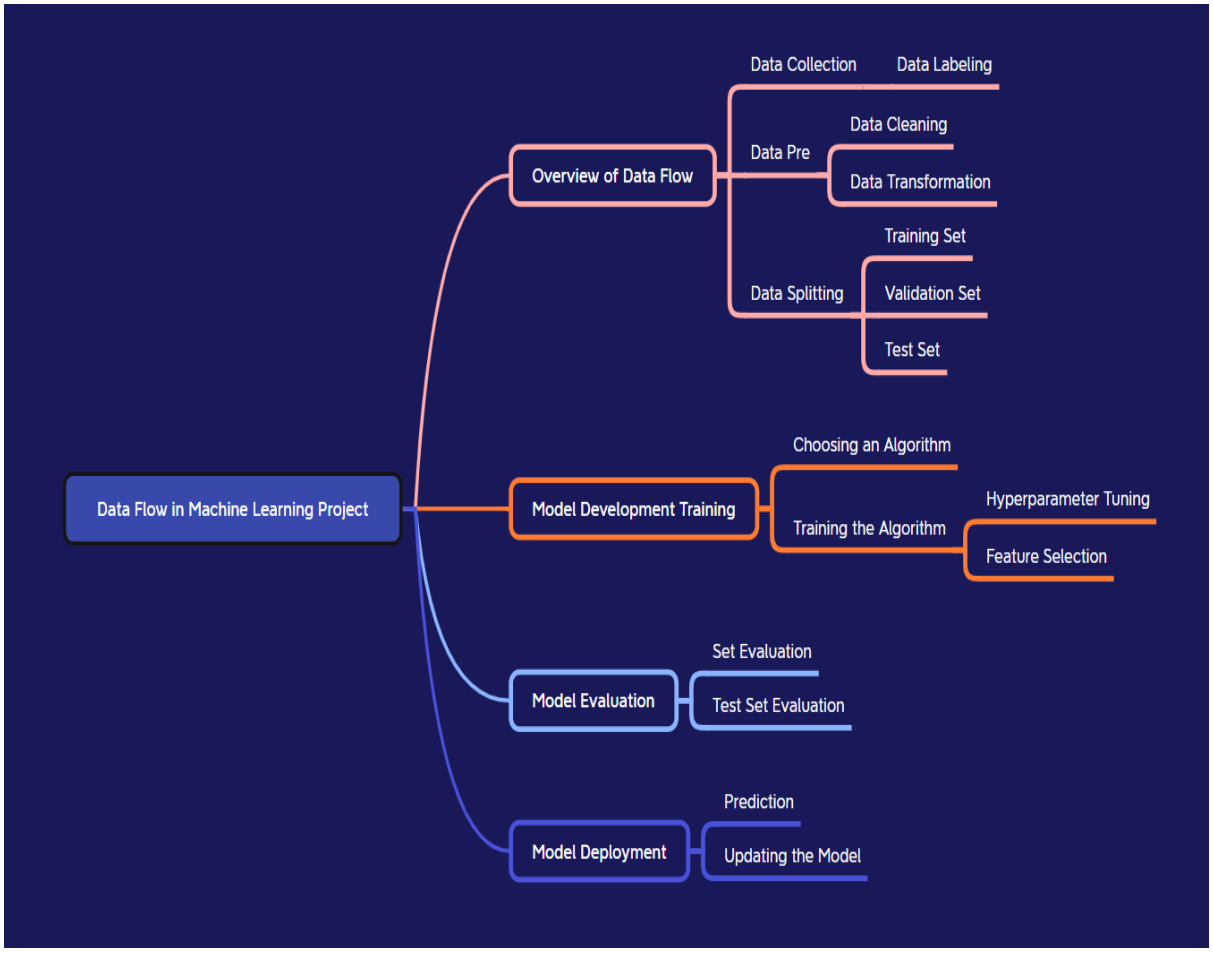
6. Model Evaluation:

- The model evaluation component assesses the performance of the trained models using evaluation metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared.
- Cross-validation techniques may be employed to evaluate the models' generalization ability and robustness.
- Model performance summaries and insights are documented for analysis and comparison.

7. Future Price Prediction:

- The future price prediction component utilizes the trained model(s) to generate predictions for future fuel prices based on the latest available data and selected features.
- Predictions are made for a specified time horizon, considering the identified factors that influence fuel prices.
- The predictions are generated using the trained model(s) and are made available for further analysis, visualization, or integration with other systems.

3. Data Flow



Input Data Sources:

1. Historical Fuel Price Data
2. Crude Oil Price Data
3. Economic Indicators

Data Preprocessing Steps:

1. Data Cleaning
2. Data Integration
3. Feature Scaling/Normalization
4. Feature Engineering
5. Data Splitting

Final Prediction Output:

The final prediction output is the forecasted fuel prices for the specified time period. The predictions are typically represented as a time series of fuel prices for different fuel types. The output can be in various formats, such as a list of predicted prices, a time series plot, or a downloadable file. The predicted fuel prices provide insights into future price trends, allowing stakeholders to make informed decisions related to fuel pricing, budgeting, and strategy planning.

4. API Design

1. Endpoint: `/predict`

Method: POST

Description: This endpoint receives input data for fuel price prediction and returns the predicted fuel prices.

Request Format:

```
{
  "historical_prices": [10.2, 9.8, 11.3, ...],
  "crude_oil_prices": [60.5, 62.1, 64.8, ...],
  "economic_indicators": {
    "gdp_growth": 2.5,
    "unemployment_rate": 5.2,
    ...}
}
```

Response Format:

```
{
  "predicted_prices": [12.1, 11.8, 13.2, ...]
}
```

Example API Request:

```
POST /predict
Content-Type: application/json
{
  "historical_prices": [10.2, 9.8, 11.3, ...],
  "crude_oil_prices": [60.5, 62.1, 64.8, ...],
  "economic_indicators": {
    "gdp_growth": 2.5,
    "unemployment_rate": 5.2,
    ...
  }
```

}```

Example API Response:

```HTTP/1.1 200 OK

Content-Type: application/json

{

"predicted_prices": [12.1, 11.8, 13.2, ...]

}```

## 2. Endpoint: `/info`

**Method: GET**

**Description: This endpoint provides additional information about the fuel price prediction model.**

**Response Format:**

``` {

"model_name": "FuelPricePredictor",

"model_version": "1.0.0",

"description": "A machine learning model for predicting fuel prices based on historical data and economic indicators."

}```

**Example API Request:**

```GET /info```

**Example API Response:**

``` HTTP/1.1 200 OK

Content-Type: application/json

{

"model_name": "FuelPricePredictor",

"model_version": "1.0.0",

"description": "A machine learning model for predicting fuel prices based on historical data and economic indicators."

}```

## 3. Endpoint: `/status`

**Method: GET**

**Description: This endpoint returns the status of the API server.**

**Response Format**

``` {
"status": "API server is running"
} ```

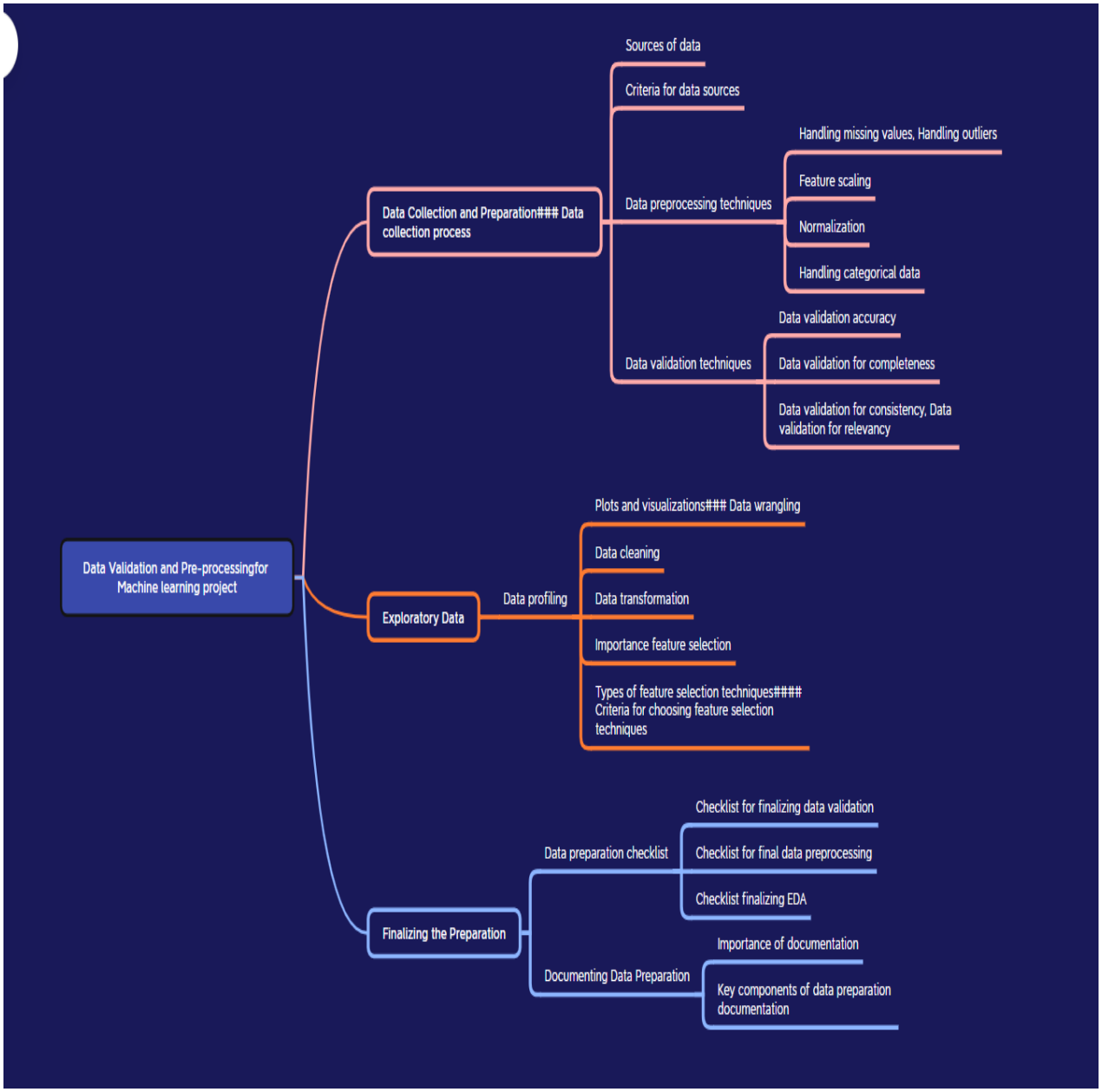
**Example API Request:**

```GET /status```

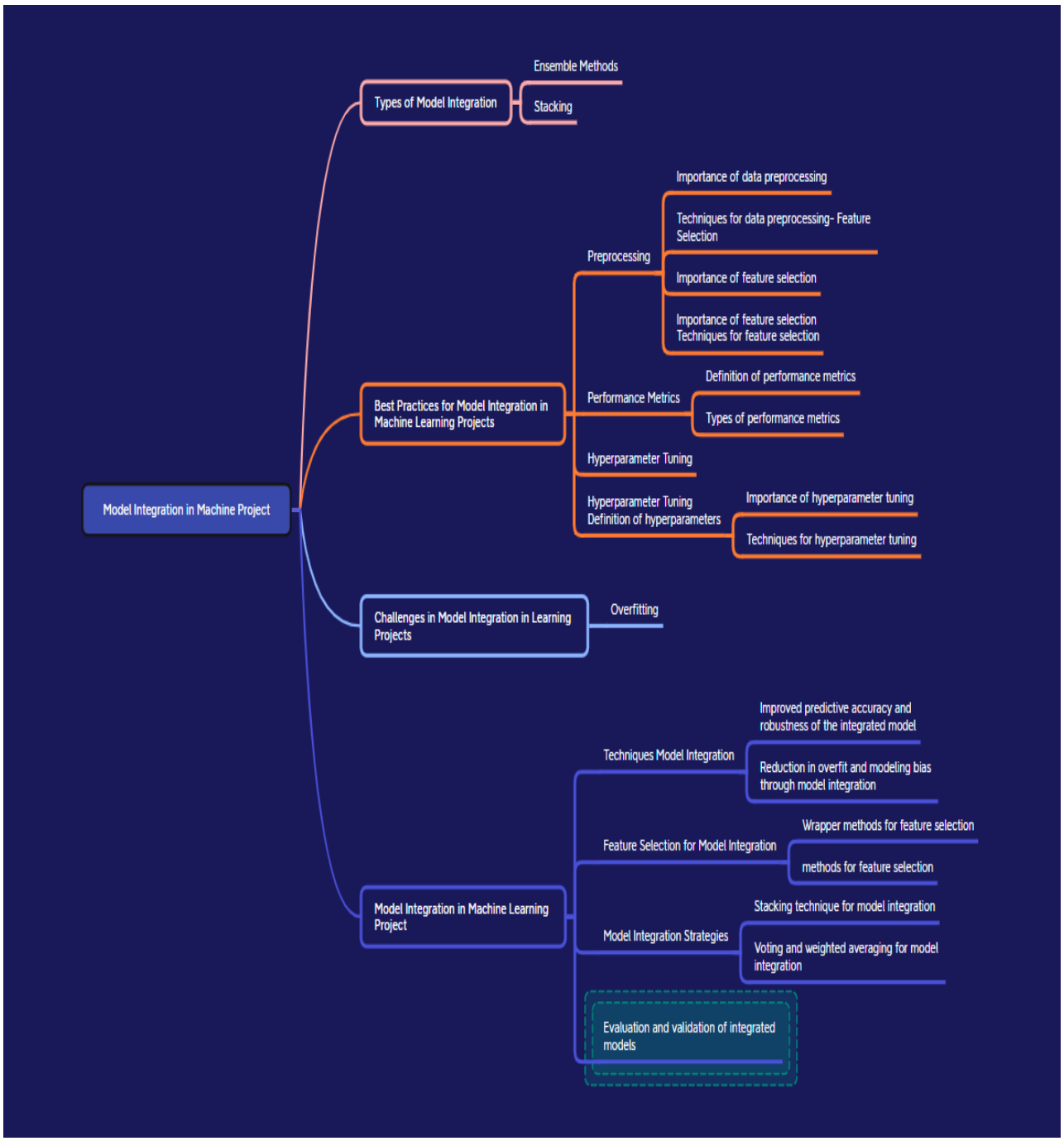
**Example API Response:**

```HTTP/1.1 200 OK
Content-Type: application/json
{
"status": "API server is running"
}```
```

5. Data Validation and Preprocessing



6. Model Integration



7. Error Handling

7.1 Identifying Potential Errors, Exceptions, and Edge Cases:

- Missing Input Data:
- Invalid Data Format:
- Out-of-Range Values:
- Model Unavailability
- Server Errors

7.2 Error Messages or Response Codes:

- InputDataMissingException
- InvalidDataFormatException
- OutOfRangeValueException
- ModelUnavailableException
- ServerErrorException

7.3 Error Handling Mechanisms:

- Try-Catch Blocks
- Validation Checks
- Logging and Monitoring
- Custom Exceptions
- Graceful Error Responses

8. Security

Security Measures Implemented in the Project:

1. Authentication:

- User Authentication: Implement user authentication mechanisms, such as username and password-based authentication, to verify the identity of users accessing the API or user interface.
- Token-Based Authentication: Utilize token-based authentication methods like JSON Web Tokens (JWT) to issue and validate tokens for authorized API access.

2. Authorization:

- Role-Based Access Control: Define roles and permissions to restrict access to specific API endpoints or functionalities based on user roles (e.g., admin, regular user).
- Fine-Grained Authorization: Implement authorization checks at the endpoint level to ensure that only authorized users can access or perform certain operations.

3. Encryption:

- Data Encryption: Encrypt sensitive data at rest to protect it from unauthorized access. Utilize encryption algorithms such as Advanced Encryption Standard (AES) or RSA for secure storage of data.
- Transport Layer Security (TLS): Use HTTPS (HTTP over TLS) to encrypt data during communication between the client and the server, ensuring secure transmission of sensitive information.

4. Input Data Validation:

- Validate and sanitize user input data to prevent security vulnerabilities such as SQL injection, cross-site scripting (XSS), or other common security exploits.
- Implement input validation checks to ensure that user-provided data adheres to the expected format and does not contain malicious content.

5. API Key or Token-Based Access:

- Issue unique API keys or tokens to authorized users or applications to access the API.
- Validate the API keys or tokens for each API request to ensure that only authenticated and authorized requests are processed.

6. Logging and Monitoring:

- Implement logging mechanisms to record important events, errors, and access attempts for auditing purposes.
- Monitor logs and system activities to detect any suspicious or unauthorized activities, helping to identify and mitigate security threats.

7. Data Protection:

- Data Minimization: Collect and retain only the necessary data required for the fuel price prediction process, minimizing the potential exposure of sensitive information.
- Data Anonymization: Anonymize or pseudonymize sensitive data whenever possible to protect individual privacy.
- Secure Storage: Store data in secure databases or storage systems, ensuring appropriate access controls, backups, and redundancy measures are in place.

8. Security Audits and Updates:

- Regularly perform security audits and vulnerability assessments to identify and address potential security vulnerabilities.
- Stay updated with the latest security patches, libraries, and frameworks to ensure the project benefits from the latest security enhancements.

9. Performance Optimization

Performance Optimizations can be Implemented in the Project:

- Caching
- Parallel Processing
- Batch Processing
- Asynchronous Processing

Scalability Considerations and Techniques:

- Horizontal Scaling
- Distributed Computing
- Cloud Infrastructure
- Database Optimization
- Performance Monitoring

10. Deployment Strategy

Deployment of the Fuel Price Prediction Project in a Production Environment:

10.1 Deployment Architecture:

Server Infrastructure:

- Cloud Platform: Utilize a cloud platform such as Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure.
- Virtual Machines (VMs): Provision VM instances to host the backend API, model serving components, and other required services.
- Load Balancer: Configure a load balancer to distribute incoming requests across multiple VM instances for load balancing and high availability.
-

10.2 Steps to Deploy and Configure the Project:

1. Set up the Server Infrastructure:

- Provision the required virtual machines or cloud instances with the desired operating system and hardware specifications.
- Install necessary dependencies, such as Python, web servers (e.g., Nginx), and database clients.

2. Deploy the Backend API:

- Install and configure a web server, such as Nginx or Apache, to serve as a reverse proxy for the backend API.
- Set up the backend API codebase on the server and ensure the necessary dependencies are installed using package managers like pip.

3. Model Deployment and Integration:

- Load the trained fuel price prediction model onto the server or utilize cloud-based model serving platforms like AWS SageMaker or Google Cloud AI Platform.
- Integrate the model into the backend API, allowing the API to make predictions using the model.

4. Configure API Security:

- Set up SSL/TLS certificates for secure HTTPS communication between clients and the API, ensuring data confidentiality during transmission.
- Implement authentication and authorization mechanisms to control access to the API endpoints, utilizing techniques like token-based authentication or OAuth.

5. Set up Database and Data Storage

6. Performance Optimization and Scaling

7. Testing and Quality Assurance

8. Continuous Monitoring and Maintenance

References:

1. XMind copilot AI for flowcharts