# Low Level Design (LLD)

# SHIPMENT PRICING PREDICTION


Project By: Shivanand Satyappa Nashi
Technologies: Machine Learning Technology
Domian: Supply Chain Management

## Contents

# ABSTRACT

**The goal of this project is to develop a model that can accurately predict supply chain shipment pricing based on a variety of factors. The market for supply chain analytics is expected to experience significant growth over the next few years, as organizations increasingly recognize the benefits of being able to forecast future events with a high degree of certainty. By accurately predicting supply chain pricing, supply chain leaders can address challenges, reduce costs, and improve service levels.**

# 1. Introduction

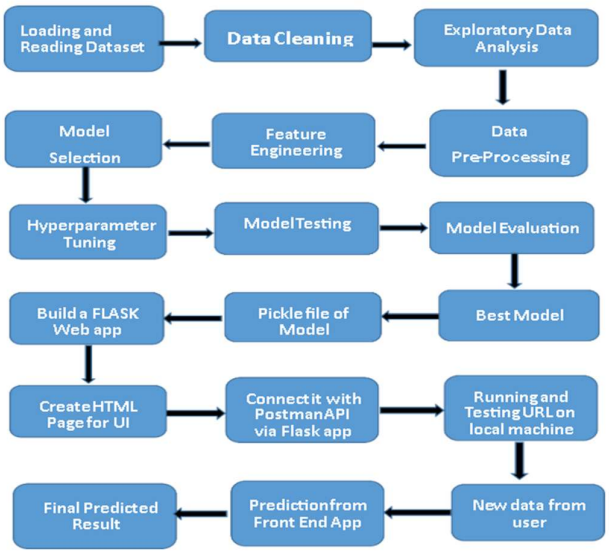
## 1.1 Why this Low-Level Design Document?

The goal of the Low-level design document (LLDD) is to give the internal logic design of the actual program code for the Shipment Pricing Prediction. 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. This 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.

# 2. Architecture



# 3. Architecture Description

## 3.1 Loading and Reading Dataset

The primary source of data for this project from Kaggle. The dataset is comprised of 10,324 records with 33 attributes. The data is in structured format. The data is a stored locally and is exported as a CSV file to be used for Data Pre-processing and Model Training.

| ID | Project Code | PQ # | PO / SO # | ASN/DN # | Country | Managed By | Fulfill Via | Vendor INCO Term | Shipment Mode | PQ First Sent to Client Date | PO Sent to Vendor Date | Scheduled Delivery Date | Delivered to Client Date | Delivery Recorded Date | Product Group | Sub Classification |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100-CI-T01 | Pre-PQ Process | SCMS-4 | ASN-8 | Côte d'Ivoire | PMO - US | Direct Drop | EXW | Air | Pre-PQ Process | Date Not Captured | 2-Jun-06 | 2-Jun-06 | 2-Jun-06 | HRDT | HIV test |
| 3 | 108-VN-T01 | Pre-PQ Process | SCMS-13 | ASN-85 | Vietnam | PMO - US | Direct Drop | EXW | Air | Pre-PQ Process | Date Not Captured | 14-Nov-06 | 14-Nov-06 | 14-Nov-06 | ARV | Pediatric |
| 4 | 100-CI-T01 | Pre-PQ Process | SCMS-20 | ASN-14 | Côte d'Ivoire | PMO - US | Direct Drop | FCA | Air | Pre-PQ Process | Date Not Captured | 27-Aug-06 | 27-Aug-06 | 27-Aug-06 | HRDT | HIV test |
| 15 | 108-VN-T01 | Pre-PQ Process | SCMS-78 | ASN-50 | Vietnam | PMO - US | Direct Drop | EXW | Air | Pre-PQ Process | Date Not Captured | 1-Sep-06 | 1-Sep-06 | 1-Sep-06 | ARV | Adult |
| 16 | 108-VN-T01 | Pre-PQ Process | SCMS-81 | ASN-55 | Vietnam | PMO - US | Direct Drop | EXW | Air | Pre-PQ Process | Date Not Captured | 11-Aug-06 | 11-Aug-06 | 11-Aug-06 | ARV | Adult |

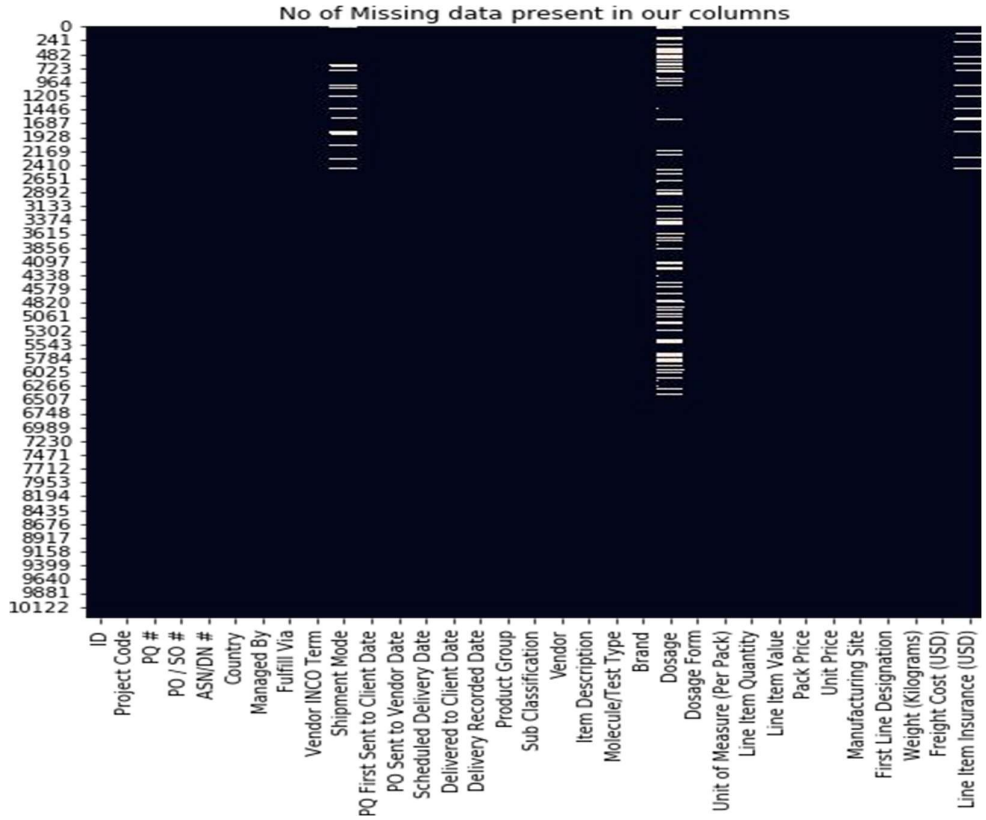
| | Vendor | Item Description | Molecule/Test Type | Brand | Dosage | Dosage Form | Unit of Measure (Per Pack) | Line Item Quantity | Line Item Value | Pack Price | Unit Price | Manufacturing Site | First Line Designation | Weight (Kilograms) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RANBAXY Fine Chemicals LTD. | HIV, Reveal G3 Rapid HIV-1 Antibody Test, 30 T... | HIV, Reveal G3 Rapid HIV-1 Antibody Test | Reveal | NaN | Test kit | 30 | 19 | 551.0 | 29.00 | 0.97 | Ranbaxy Fine Chemicals LTD | Yes | 13 |
| | Aurobindo Pharma Limited | Nevirapine 10mg/ml, oral suspension, Bottle, 2... | Nevirapine | Generic | 10mg/ml | Oral suspension | 240 | 1000 | 6200.0 | 6.20 | 0.03 | Aurobindo Unit III, India | Yes | 358 |
| | Abbott GmbH & Co. KG | HIV 1/2, Determine Complete HIV Kit, 100 Tests | HIV 1/2, Determine Complete HIV Kit | Determine | NaN | Test kit | 100 | 500 | 40000.0 | 80.00 | 0.80 | ABBVIE GmbH & Co.KG Wiesbaden | Yes | 171 |
| | SUN PHARMACEUTICAL INDUSTRIES LTD (RANBAXY LAB... | Lamivudine 150mg, tablets, 60 Tabs | Lamivudine | Generic | 150mg | Tablet | 60 | 31920 | 127360.8 | 3.99 | 0.07 | Ranbaxy, Paonta Shahib, India | Yes | 1855 |
| | Aurobindo Pharma Limited | Stavudine 30mg, capsules, 60 Caps | Stavudine | Generic | 30mg | Capsule | 60 | 38000 | 121600.0 | 3.20 | 0.05 | Aurobindo Unit III, India | Yes | 7590 |

### 3.2 Data Cleaning

Data cleaning is the process of detecting and correcting errors, inconsistencies, or missing values in a dataset. It is an essential step in data pre-processing and is often the most time-consuming part of the data analysis process.

There are many different techniques for cleaning data, depending on the nature of the dataset and the types of errors it contains. Some common techniques include:

- Identifying and removing duplicates
- Handling missing values (e.g., replacing them with a default value or dropping them)
- Correcting data formatting issues (e.g., inconsistent date formats)
- It is important to carefully inspect the data and carefully consider the appropriate cleaning techniques to apply, as these can significantly affect the quality and usefulness of the data.

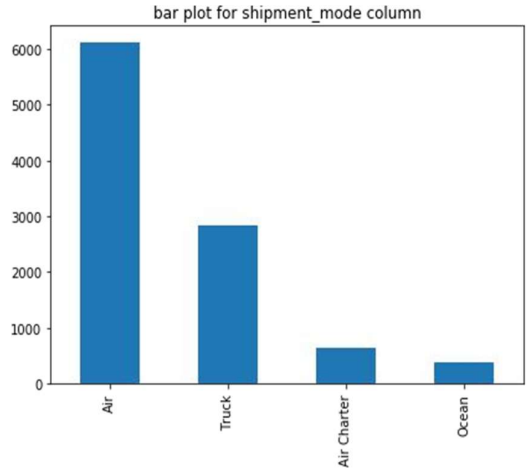


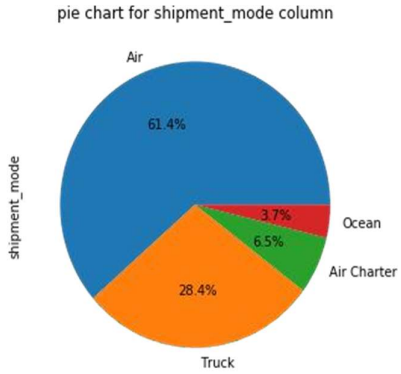

### 3.3 Exploratory Data Analysis (EDA)

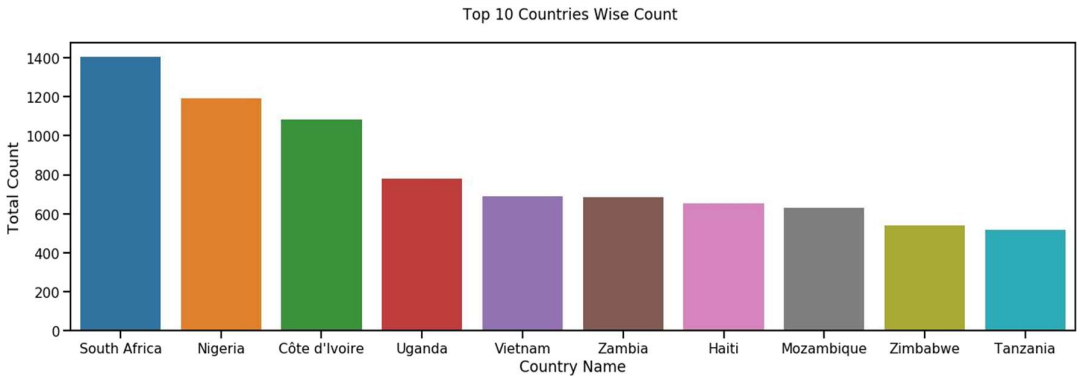
Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

- **Bar charts**: Shows the distribution of a categorical variable or the comparison of multiple variables.
- **Histograms**: Shows the distribution of a continuous variable.

- **Scatter plots**: Shows the relationship between two continuous variables.
- **Box plots**: Shows the distribution of a continuous variable by displaying the median, quartiles, and outliers.
- **Heat maps**: Shows the relationship between two categorical variables or the distribution of a continuous variable over two categorical variables.
- **Pie charts**: Shows the proportion of a categorical variable.



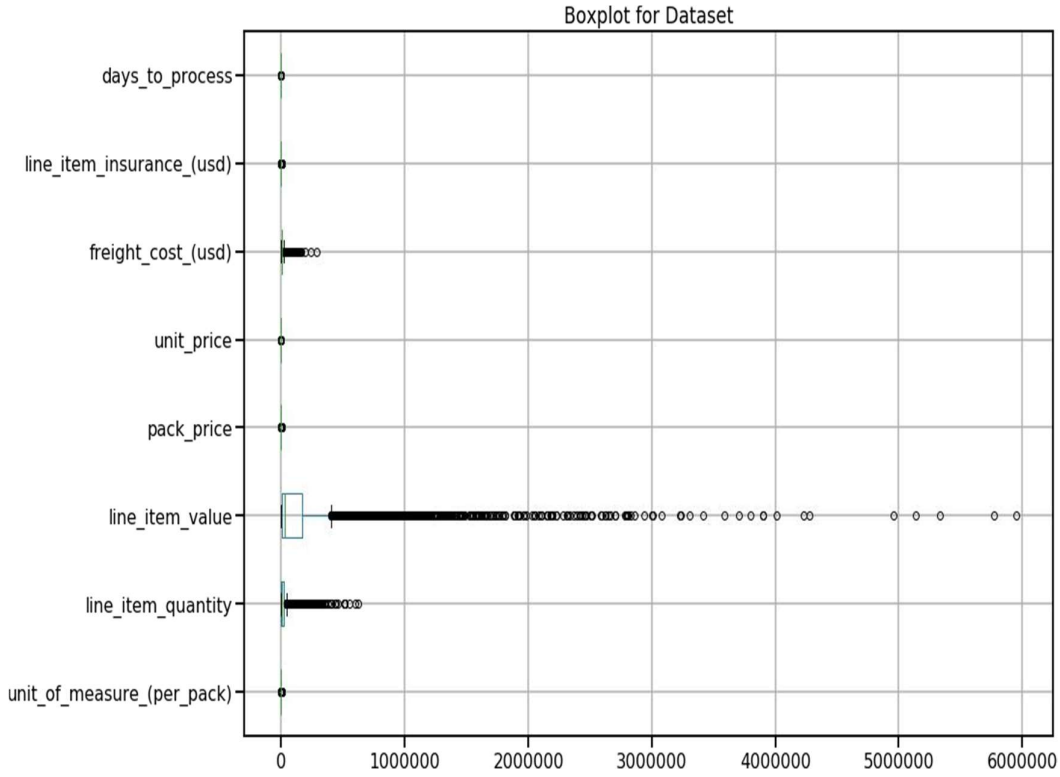

### 3.4 Data Pre-Processing

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:

- Data transformation: Modifying the data to fit the requirements of the analysis or model being used.
- Data scaling: Normalizing the data by scaling it to a specific range.
- Data reduction: Reducing the number of features or samples in the dataset to improve the efficiency of the analysis or model.

- Outlier Treatment: Outliers Detection and Removal




### 3.5 Feature Engineering

Feature engineering is the process of designing and creating new features or variables from existing data to improve the performance of a machine learning model. It is a key step in the data preprocessing process, as the quality and relevance of the features can significantly affect the model's ability to learn and generalize.

Some common techniques for feature engineering include:

- Feature selection: Selecting a subset of the most relevant features from the data.
- Feature extraction: Creating new features from existing data by applying transformations or combining existing features.
- Feature creation: Generating new features based on domain knowledge or by applying algorithms to the data.

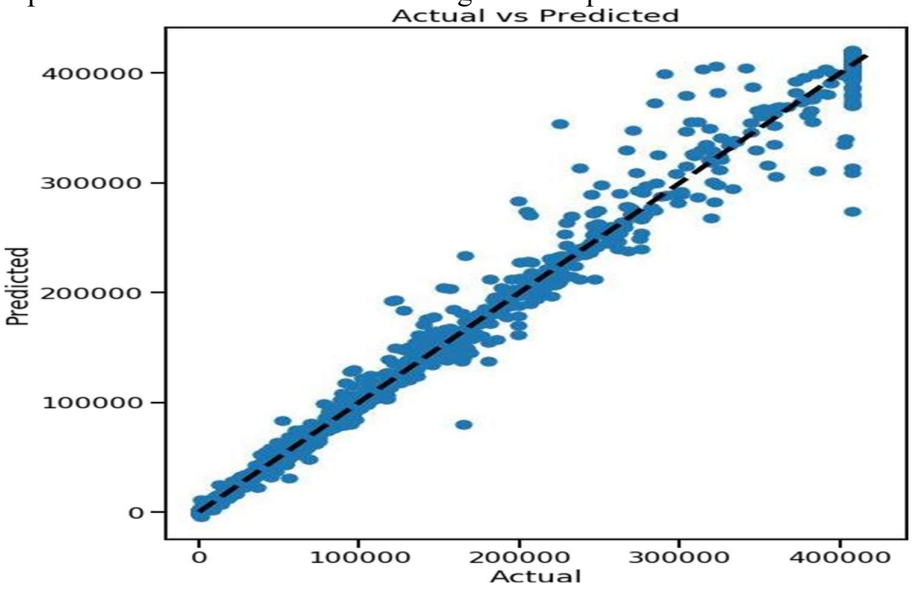
### 3.6 Model Selection

In This step we apply different machine learning algorithms to our processed data and select the model with best results for further Hyper-parameter tuning to make the best possible model that can be made for accurate and correct prediction.

Model building is an iterative process, and the specific steps and techniques used will depend on the nature of the data and the goals of the analysis.

### 3.7 Model Evaluation

We evaluate our regression model based on performance metrics such as R- square, Adjusted R-square, Root mean square values. We can also compare the model's predictions to the actual values using a scatter plot.




### 3.8 Deployment Process

To deploy a model, we used the following steps:

1. Save the trained model as a pickle file using Python's pickle library.
2. Create a Flask app in Python, which will act as the server for your model.
3. Define the routes for the Flask app, which will determine the behavior of the server when it receives different HTTP requests.
4. In the routes, you can load the pickle file and use it to make predictions based on the input received in the request.
5. We created HTML templates to display the results of the predictions on a website.
6. Test the Flask app using Postman or a similar API testing tool to ensure it is working correctly.

# 4 Unit Test Cases

| Test Case Description | Pre-Requisite | Expected Result |
|---|---|---|
| Verify whether the Application URL is accessible to the user | 1. Application URL should be defined. | 1. Application URL should be accessible to the user. |
| Verify whether the application loads completely for the user when the URL is accessed | 1. Application URL is accessible<br>2. Application URL is deployed. | Application URL should load completely for the user when URL is accessed. |
| Verify whether user can see input field after opening URL | 1. Application is accessible | User should be able to see input fields after opening URL |
| Verify whether user can edit all the input fields | 1. Application is accessible | User should be able to edit all the input fields |
| Verify whether user has options to filter the inputs fields | 1. Application is accessible | User should filter the options of input fields |
| Verify whether user gets submit button to submit the inputs | 1. Application is accessible | User should get submit button to submit the inputs |
| Verify whether user can see the output after submitting the inputs | 1. Application is accessible | User should get outputs after submitting the inputs |