HIGH LEVEL DESIGN (HLD)

Product Backorder Prediction

REVISION NUMBER – 1.0
LAST DATE OF REVISION – 01/08/2022

SHIVANSH KAUSHAL, CHIRAG SHARMA

Data Version Control

Date Issued	Version	Description	Author
01/08/2022	1	Initial HLD-V1.0	Shivansh Kaushal & Chirag Sharma

Contents

Abstract

Backorders are unavoidable, but by anticipating which things will be backordered planning can be streamlined at several levels, preventing unexpected strain on production, logistics, and transportation. ERP systems generate a lot of data (mainly structured) and also contain a lot of historical data; if this data can be properly utilized, a predictive model to forecast backorders and plan accordingly can be constructed. Based on past data from investors, supply chain, and sales, classify the products as going into backorder (Yes or No).

The Goal here is to build a solution that should be able to predict the backorder sales for a particular product according to the provided dataset.

Introduction

1.1. Why this High-Level Design (HLD)?

The purpose of this High-Level Design (HLD) Document is to add the necessary details to the current project description to represent a suitable model for coding. This document is also intended to help detect contradiction prior to 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 being implemented.
- Includes design features and the architecture of the project
- List and describe the non-functional attributes like :
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - 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 architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrator of the system.

1.3. Definitions

Term	Definition
Database	Collection of all the information uploaded by the system or used by the system.
IDE	Integrated Development Environment
AWS	Amazon Web Services

General Description

2.1 Problem Statement

Backorders are unavoidable, but by anticipating which things will be backordered planning can be streamlined at several levels, preventing unexpected strain on production, logistics, and transportation. ERP systems generate a lot of data (mainly structured) and also contain a lot of historical data; if this data can be properly utilized, a predictive model to forecast backorders and plan accordingly can be constructed. Based on past data from investors, supply chain, and sales, classify the products as going into backorder (Yes or No).

The Goal here is to build a solution that should be able to predict the backorder sales for a particular product according to the provided dataset.

2.2 Dataset Description

We have the collected the dataset from the following Github repository: https://github.com/rodrigosantis1/backorder_prediction/blob/master/dataset.rar .

2.3 Attributes Information

1. Went_on_backorder: If the product went on backorder or not.
2. SKU: A stock keeping unit (SKU) is a number that is assigned to a product for the purpose of inventory management and ease tracking.
3. National_inv: Current inventory levels of different products.
4. Lead_time: In general, lead time in inventory management is the amount of time between when a purchase order is replenish products and when the order is received in the warehouse.
5. In_transit_quantity: Transit quantity generally refers to the goods that have not yet made it from one company to another.
6. Forecast_x_month: The demand estimated for **x** months where **x** can be 3, 6 and 9.

7. Sales_x_month: Actual sales for the prior **x** months where **x** can be 1, 3, 6 and 9.

8. Min_bank: A minimum stock level is a threshold value that indicates the level below which actual material stock items should not normally be allowed to fall. In other words, a minimum stock level is a minimum quantity of a particular item of material that must be kept at all times. The fixing of this level acts as a safety measure. For this reason, the minimum stock level is commonly known as safety stock or buffer stock.

9. Potential_issue: Indicator variable noting potential issue with them.

10. Pieces_past_due: Products overdue from source (sourcing is identifying vendors who will produce and services to meet planned/actual demand in the most economical and efficient way).

11. Perf_x_month: Source average performance in last **x** months where can take values 6 or 12.

12. Local_bo_quantity: Amount of overdue stock orders.

13. Deck_risk: The products that might remain in the deck/ship/stock.

14. Oe_constraint: Products that are facing operational limiting factors such as bottleneck.

15. Ppap_risk: Risks associated with packaging and production.

16. Stop_auto_buy: Whether automatic selling process has been stopped or not.

17. Rev_stop: Revenue status for product.

2.4 Tools used

- Python is used as a programming language.
- Flask is used for the backend development.
- PyCharm is used as an IDE.
- Frontend is done by HTML & CSS.
- MongoDB is used as a database to store the data & logging.
- Scikit-Learn is used to create models.
- For Visualization of Plots Matplotlib & Seaborn are used.

- Github is used as Version Control.
- Heroku is used to deploy the application.

2.5 Constraints

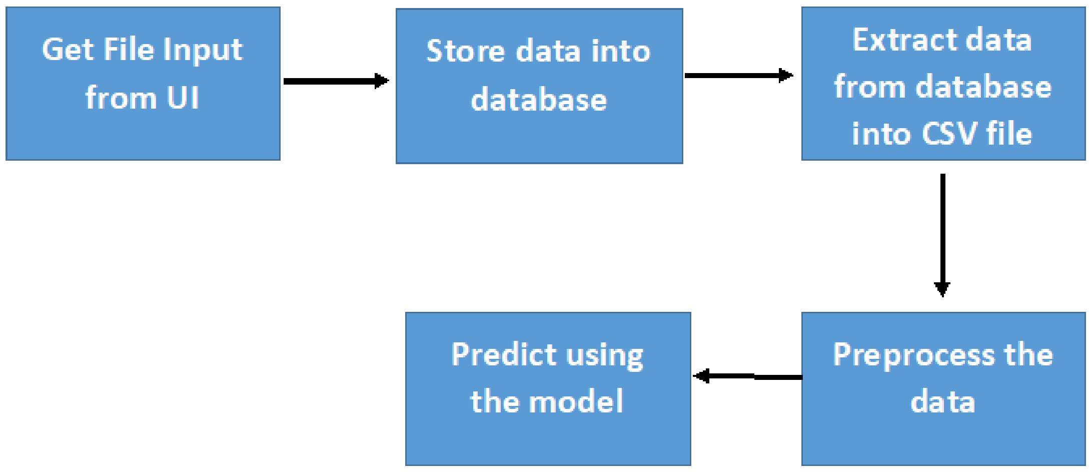
The Backorders prediction application must be user-friendly, as automated as possible, and the user should not be required to know any other work.

2.6 Assumptions

The main objective of this project is to implement the use cases as previously mentioned (2.2 Problem statement) as fast as possible, so that the organizations can test the cases. It is also assumed that all the aspects of the project have the ability to work together in the way organization is expecting.

Design Details

3.1 Process Flow –



3.2 Event Log

The system should log every event so that the user will know what process is running internally.

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. Developers can choose a logging method. You can choose database logging or file logging as well.
4. System should not hang even after so many loggings. Logging is used just because we can debug issues, so logging is mandatory.

3.3 Error 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.

Performance

This system will help to predict the backorder sales for a particular product.

4.1 Reusability

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

4.2 Application Compatibility

The different components for this project will be using Python as an Interface between them. Each component will have its own task to perform, and it is the job of the python to ensure the proper transformation of Information.

4.3 Resource Allocation

When any task is performed, it will likely use all the processing power available until that function is finished.

4.4 Deployment –

aws Azure Google Cloud Platform

KPI'S (Key Performing Indicators)

Organization can predict the backorder sales for a particular product according to the given dataset.

Conclusion

Backorders are unavoidable, but by anticipating which things will be backordered planning can be streamlined at several levels, preventing unexpected strain on production, logistics, and transportation. ERP systems generate a lot of data (mainly structured) and also contain a lot of historical data; if this data can be properly utilized, a predictive model to forecast backorders and plan accordingly can be constructed. Based on past data from investors, supply chain, and sales, classify the products as going into backorder (Yes or No).