High Level Design (HLD)

Backorder Prediction

Document Version Control

Date Issued	Version	Description	Author
7 Feb 2025	1	Initial HLD – V1.0	Amar Hulamani

Table of Contents

Abstract

Backorders occur when a product cannot be fulfilled at the time of purchase due to temporary stock unavailability, yet a future delivery date is guaranteed. Unlike an out-of-stock scenario, where no clear replenishment timeline exists, backorders allow businesses to continue accepting customer orders while inventory is being replenished. Although effective demand forecasting can mitigate backorders, unpredictable factors such as supply chain disruptions and sudden demand surges can still make them unavoidable.

Frequent backorders, however, indicate inefficiencies in inventory management and supply chain operations, potentially leading to revenue loss and customer dissatisfaction. To optimize inventory and minimize backorder risks, businesses must implement accurate demand forecasting, efficient supply chain strategies, and proactive customer communication. By balancing inventory availability and operational agility, companies can enhance customer satisfaction while maintaining a resilient and efficient supply chain.

1. Introduction

This document will be used for documenting High-level designs of project.

1.1 Purpose of the Document

The purpose of this plan is to :-

- Identify different design approaches.
- Identify core modules/sub-systems of the system and sub-system boundary.
- Identify the best suitable technology for various sub-systems.
- Identify areas that need R&D.
- Identify third party components required in the system. Identify components, state, life cycle, and communication mechanisms between different sub-systems, and also identify the external interface.
- Identify various usage scenarios.

1.2 Objective of HLD

- To provide an overview of the entire system.
- To provide a module-wise breakup of the entire system.
- To provide introduction and high level working of every module involved.

1.3 Scope of HLD

This HLD covers all areas of system.

2. System Overview

2.1 Product Prospective

Backorder prediction problem using classification-based Machine Learning algorithms.

2.2 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 contain a lot of historical data. A predictive model to forecast backorders is to be made and inventory planning accordingly can be constructed. Based on past data from inventories, supply chain, and sales, classify the products as going into backorder (Yes or No).

2.3 Proposed Solution

The solution here is a Classification based Machine Learning model. It can be implemented by different classification algorithms (like Logistic Regression, Random forest, Decision tree , XGBoost, Balanced Random Forest and so on). Here First we are performing Data preprocessing step, in which Data cleaning, Data Transformation, feature engineering, feature selection steps are performed and then we are going to build model.

2.4 Technical Requirements

In this Project the requirements to check to predict the backorder sales for a particular product according to the provided dataset. For that, in this project we are going to use different technologies. Here is some requirements for this project.

- Model should be exposed through API or User Interface, so that anyone can test model.
- Model should be deployed on cloud (Azure, AWS, GCP).

2.5 Data Requirements

Data Requirement completely depend on our problem.

For training and testing the model, we are using Backorder prediction dataset that is provided by Ineuron.ai. From user we are taking following input:

- Current National Inventory of the product
- Lead Time of the product
- Total quantity of products in transit
- Forecasted Sales of the product for next 3 months
- Previous month Sales of the product
- Minimum Quantity of Products required to keep in stock
- Is there a potential issue with the product?
- No of pieces/products overdued
- Average Source Performance over the last 6 month
- Current Unmet demand
- Is there a risk of mismatch between forecasted and actual demand of the product?
- Did the product experience operational constraints?
- Is there a Production Part Approval Process risk associated with the product?
- Is automatic purchasing for this product halted?
- Is review process for this product paused?

Label-> went_on_backorder

□

2.6 Tools Used

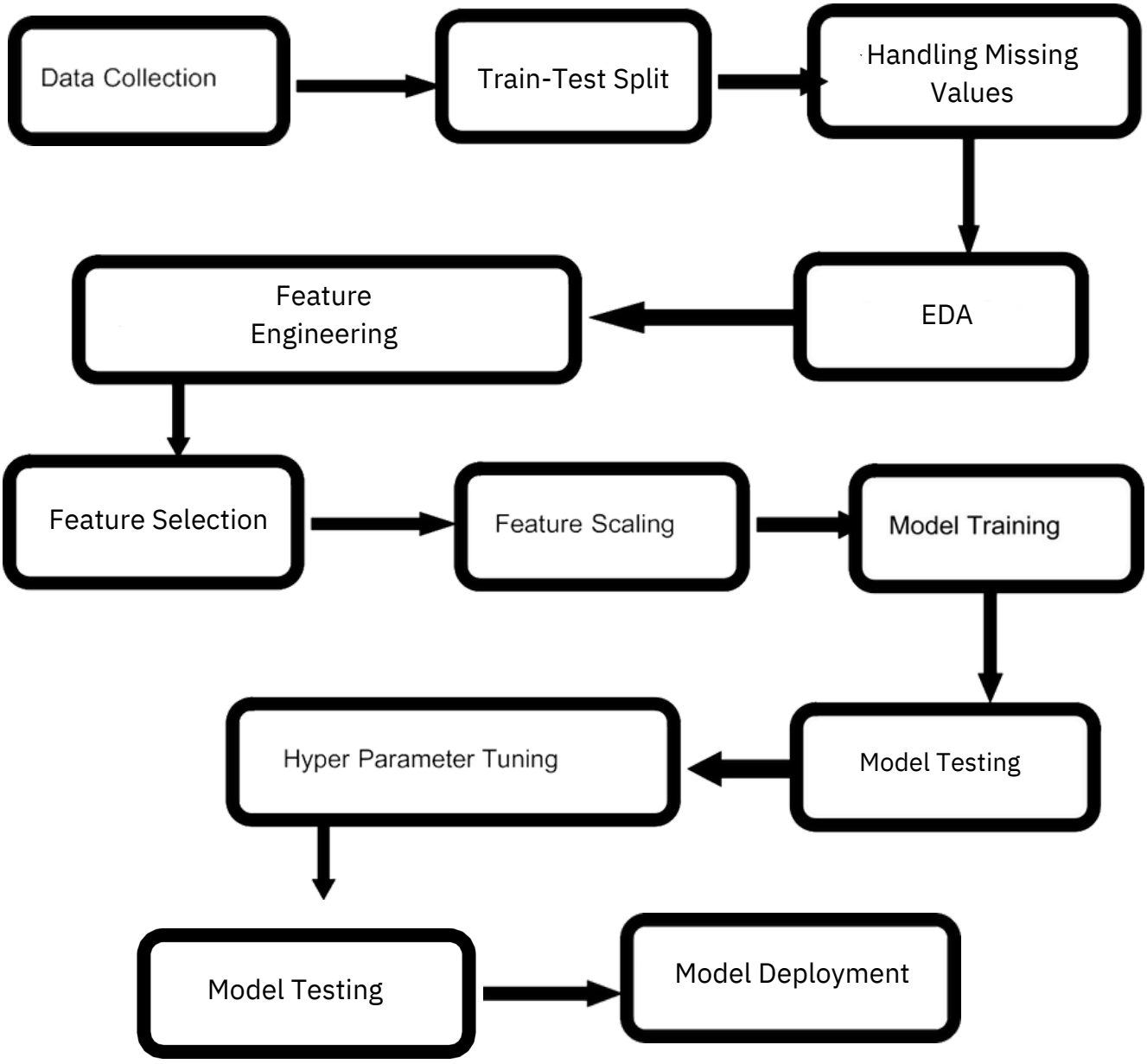
- Visual Studio Code is used as IDE. Python is used for modular programming.
- For visualization of the plots, Matplotlib, Seaborn are used.
- Scikit-Learn is used for model training along with imblearn techniques.
- Azure Virtual Machine is used for deployment of the model.
- Front end development is done using Streamlit.
- FastAPI is used as Backend API for model predictions.
- Github is used as version control system.

2.7 Constraints

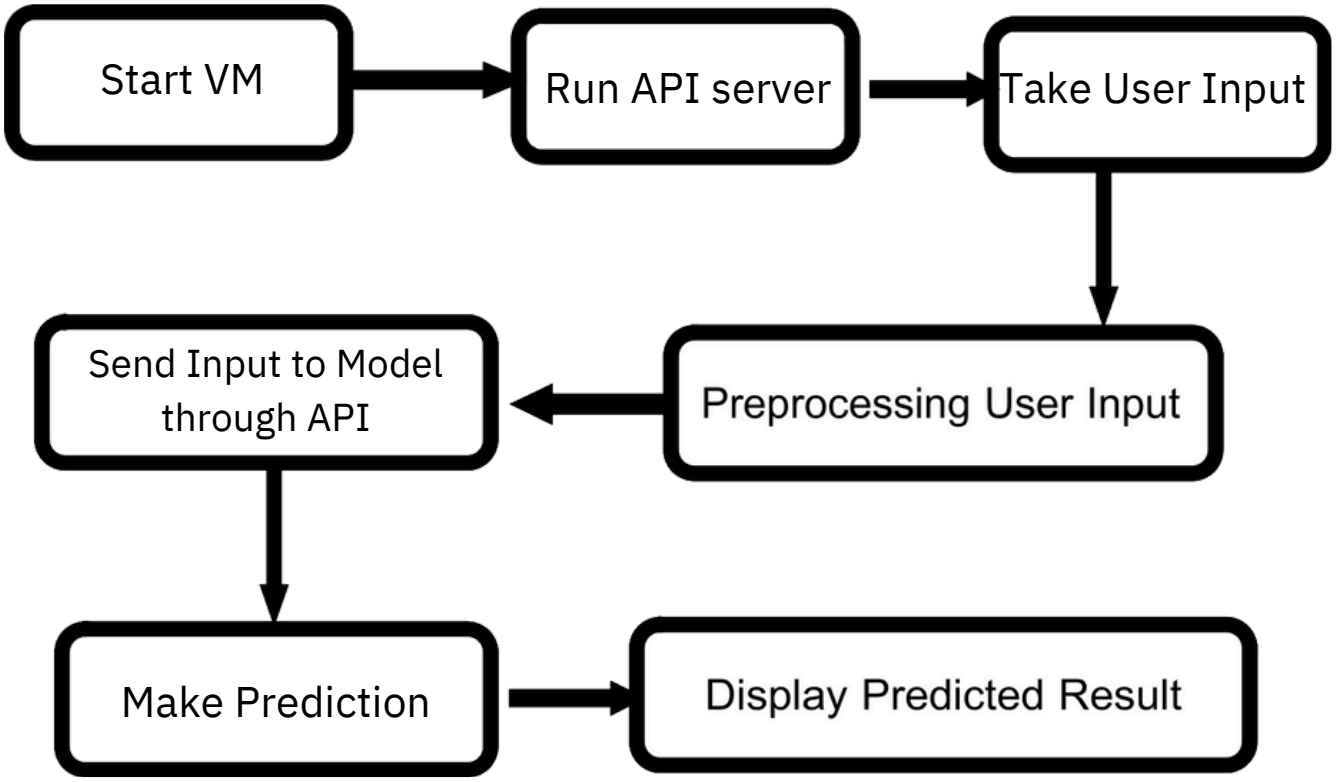
The Backorder prediction system must be user friendly, errors free and users should not be required to know any of the back end working.

3. Design Flow

3.1 Process Flow



3.2 Prediction Process



3.3 Event Log

In this Project we are logging every process to know what process is running internally.
Step-By-Step Description:

- In this Project we defined logging for every function, class.
- By logging we can monitor every insertion, every flow of data in database.
- By logging we can monitor every step which may create problem or every step which is important in file system.

3.4 Error Handling / Exception Handling

Our project is designed to handle errors smoothly without crashing. If an error occurs at any step, the system will catch it, explain what went wrong, and continue running instead of stopping abruptly.
This ensures:

- Error Detection – Identifying issues at any stage.
- Clear Messages – Providing easy-to-understand explanations.
- Smooth Execution – Preventing system failure.

With proper logging and exception handling, our application remains stable, user-friendly, and easy to debug even when unexpected errors occur.

4. Performance

The Backorder Prediction solution is designed to predict whether a product will go on backorder in advance. Since accurate predictions are crucial for effective supply chain management, our model is built with a strong emphasis on precision and reliability. To achieve this, we followed a comprehensive Machine Learning (ML) workflow, ensuring a structured approach to data preparation, model development, and deployment. Below is a summary of the complete process:

1. Problem Understanding & Data Exploration

- Defined the backorder problem and its business impact.
- Explored the dataset, which includes inventory levels, transit time, sales forecasts, overdue stock, supplier performance, and risk flags.
- Identified missing values, outliers, and potential correlations between features.

2. Data Preprocessing & Feature Engineering

- Handled missing values using Imputer class techniques.
- Converted categorical variables into numerical format.
- Scaled and normalized numerical features to improve model performance.
- Handled multicollinearity and considered only those features that might add value to the performance of the model.

3. Model Training & Selection

- Implemented various machine learning algorithms, including Logistic Regression, Random Forest, XGBoost, along with sampling techniques like RandomOverSampling, RandomUnderSampling, SMOTE to handle imbalance nature of the data.
- Also used imblearn ensemble techniques like Balanced Random Forest and Easy Ensemble.
- Used Stratified K-Fold Cross-Validation to ensure balanced training and testing.
- Tuned hyperparameters to optimize model accuracy.

4. Model Evaluation & Optimization

- Assessed models using metrics like Precision, Recall, PR-AUC, and AUC-ROC, prioritizing recall due to the imbalanced dataset.
- Selected the best-performing model based on evaluation metrics.

5. Deployment & Integration

- Developed a FastAPI backend to expose the model as an API.
- Created a Streamlit web application to provide an interactive UI for predictions, thus allowing users to input the data and get appropriate predictions about product's backorder situation.
- Implemented a model monitoring script to track prediction latency.
- Deployed both FastAPI and Streamlit on an Azure Linux VM, ensuring scalability and accessibility.

4.1 Re-usability

The programming of this project is done in such a way that it should be reusable, so that anyone can add and contribute without facing any problems.

4.2 Application Compatibility

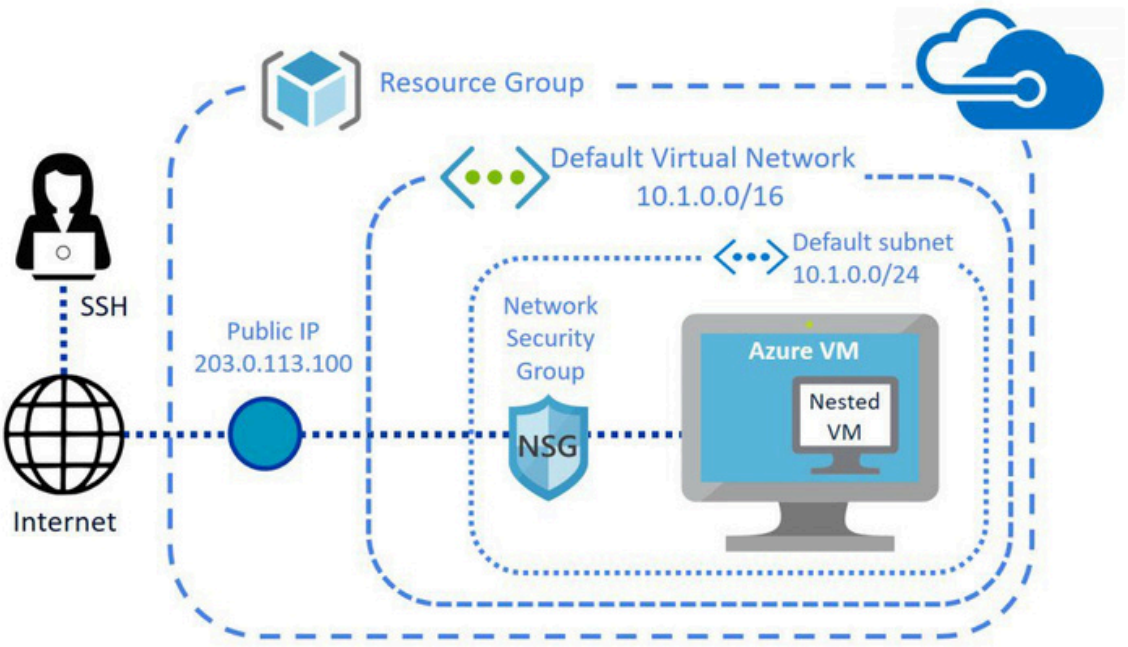
The different module of this project is using Python as an interface between them. Each modules have its own job to perform and it is the job of the Python to ensure the proper transfer of information.

4.3 Resource Utilization

In this project, when any task is performed, it will likely that the task will use all the processing power available in that particular system until it's job finished. By keeping this in mind, In this project we have used the concept of multithreading.

4.4 Deployment

The deployment of the Streamlit Web app and the FastAPI backend has been done on cloud using Azure Virtual Machine.



4.5 User Interface

The user interface has been created using Streamlit that sends a request to the API and in return displays the predicted outcome using model's prediction.

Backorder Prediction

Enter the required details to predict backorder situation

Current National Inventory of the product

0.00 − +

Lead Time of the product

0.00 − +

Total quantity of products in transit

0.00 − +

Forecasted Sales of the product for next 3 months

0.00 − +

Previous month Sales of the product

0.00 − +

Minimum Quantity of Products required to keep in stock

0.00 − +

Is there a potential issue with the product?

- (●) Yes
- ( ) No

No of pieces/products overdued

0.00 − +

Average Performance over the last 6 month

0.00 − +

Current Unmet demand

0.00 − +

Is there a risk of mismatch between forecasted and actual demand of the product?

- (•) Yes
- ( ) No

Did the product experience operational constraints?

- (•) Yes
- ( ) No

Is there a Production Part Approval Process risk associated with the product?

- (•) Yes
- ( ) No

Is automatic purchasing for this product halted?

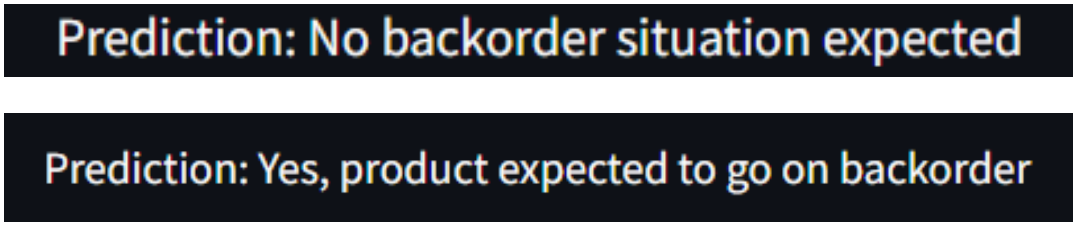
- (•) Yes
- ( ) No

Is review process for this product paused?

- (•) Yes
- ( ) No

Predict

OUTPUT:



5. Conclusion

This project successfully identifies products that are likely to be backordered based on key features derived from historical inventory, sales, and supply chain data. The results demonstrate that a predictive machine learning classification model can effectively enhance inventory management, reducing supply chain pressure and optimizing stock levels.

By accurately forecasting backorders, businesses gain:

- Greater flexibility in inventory control
- Improved customer satisfaction through timely product availability
- Cost savings by minimizing excess stock and stockouts

Among the models tested, the Balanced Random Forest achieved the best performance, making it the most effective approach for accurate and reliable backorder prediction.