Architecture

Back Order Prediction

Document Version:1.0

Control Version Date: 22-Feb-24

Author : Akshay Kumar BM

Comments: Introduction & Architecture Defined

Context

Architecture Document

1. Introduction

1.1. What is Low-Level Design Document?

A low-level design document (LLD) is a detailed blueprint of a software product, which outlines the specific components of a system and their interactions.

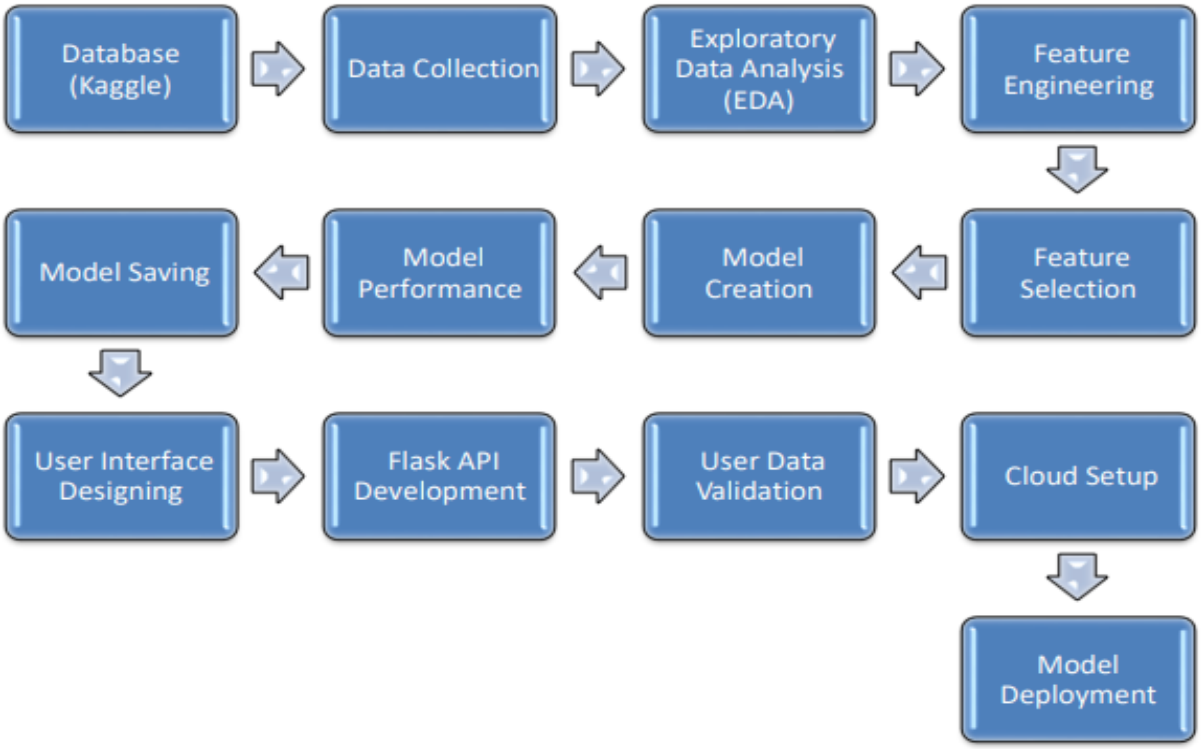
1.2. Scope

This document covers the architecture of a machine learning model for predicting backorders in inventory management.

2. Architecture

The architecture consists of several components, each responsible for a specific aspect of the system, such as data ingestion, data transformation, model training, and

prediction.



3. Architecture Description

3.1. Database (Kaggle)

The data used for this project is sourced from Kaggle. It contains various features related to product orders.

3.2. Data Collection

The `DataIngestion` class handles data collection. It reads the data from CSV files and returns the paths of the train and test data.

3.3. Exploratory Data Analysis (EDA)

EDA is performed to understand the underlying patterns of the data, identify outliers, and understand the relationship between different variables.

3.4. Feature Engineering

The `DataTransformation` class handles feature engineering. It includes handling missing values, scaling numerical features, and encoding categorical features.

3.5. Feature Selection

Relevant features are selected for the model. This includes ‘national_inv’, ‘lead_time’, ‘in_transit_qty’, ‘forecast_3_month’, ‘sales_1_month’, ‘min_bank’, and ‘perf_6_month_avg’.

3.6. Model Creation

The `ModelTrainer` class handles model creation. It includes the training of different models and evaluation based on the F1 score.

3.7. Model Performance

The performance of the model is evaluated using the F1 score. The model with the highest F1 score is selected as the best model.

3.8. Model Saving

The best model is saved using the `pickle` module for future use.

3.9. User Interface Designing

A user interface is designed to allow users to input the required features and get the prediction results.

3.10. Flask API Development

A Flask API is developed to serve the model predictions. It takes the user input from the user interface, passes it to the model, and returns the prediction.

3.11. User Data Validation

The `CustomData` class handles user data validation. It ensures that the user input is in the correct format before it is passed to the model.

3.12. Cloud Setup

The application is set up on a cloud server to allow users to access the model predictions remotely.

3.13. Model Deployment

The model is deployed on the cloud server. The Flask API serving the model predictions is also hosted on the server.

4. Unit Test Cases

Unit tests are written to ensure that all components of the system are working as expected. This includes tests for data ingestion, data transformation, model training, and prediction.

Architecture

Test Case Description	Pre-requisite	Expected Result
Verify whether the Application URL is accessible to the user.	Application URL should be defined.	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	Application URL is accessible. Application is deployed.	The Application should load completely for the user when the URL is accessed.
Verify whether user is giving standard input.	Handled test cases at backends.	User should be able to see successfully valid results.
Verify whether user is able to edit all input fields	Application is accessible.	User should be able to edit all input fields.
Verify whether user gets Predict button to submit the inputs	Application is accessible.	User should get Submit button to submit the inputs.
Verify Predicted result is displayed	Application is accessible.	User should be able to see the predicted result

Please note that this is a high-level overview of the architecture. Each component may have additional details not covered in this document. For more specific information, please refer to the code comments or additional documentation.