

Stores Sales Prediction

Low Level Design

Domain: Machine Learning

Creator: Harshal S. Patil

Date: 02.06.2023

**Document Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| Jun 2, 2023 | 1.1 | First Draft | Harshal Patil |

Contents

Introduction ………………………………………………..3

What is Low-Level Design Document?……………………3

Scope……………………………………………………….4

General Description………………………………………..4

Definitions………………………………………………….4

Product Description…………………………………………4

Problem Statement…………………………………………4

Proposed solution …………………………………………..5

Further improvements………………………………………5

Data requirements…………………………………………..5

Tools used…………………………………………………..6

Hardware Requirements…………………………………….7

Constraints…………………………………………………..7

Assumptions…………………………………………………7

Design Details……………………………………………….7

Process Flow………………………………………………...7

Model deployment…………………………………………..8

Event Log……………………………………………………8

Error Handling……………………………………………….9

Performance………………………………………………….9

Re usability……………………………………………..……9

Application Compatibility……………………………………9

Resource Utilization………………………………………….9

Dashboards………………………………………………….10

KPIs (Key Performance Indicators)………………………...10

Conclusion…………………………………………………..10

# Introduction

## What is Low-Level Design Document?

The goal of LLD or a low-level design document is to give the internal logical of the actual program code for Metro Interstate Traffic Volume Prediction. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

The main objective of the project is to predict if traffic volume is in high or low on particular date. Weather circumstance, special days like holidays, daytime (morning, afternoon, night and etc.), a temperature, a weekday, a numeric percentage of cloud cover are vital attributes for predicting traffic volume.

## Scope

The HLD documentation presents the structure of the system, such as 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 administrators of the system.

## 

# Design Details

## Process Flow

For accomplishment of the task, we will use a trained Machine Learning model. The process flow diagram is shown below:

**Model Development**

Design UI On

Flask

Model

Evaluation

Cloud

Deployment on

Cloud

Hyper-

Parameter

Tuning

Desinging a

Server

Model

Implementation

Feature

Engineering

Exploratory Data

Analysis

Data

Preprocessing

**Deployment**

Render App to keep Server Running

Loading Pipeline on Github

Integrating Github pipeline code with Flask

Deploying The code on

Render

# Architecture Description

## Data Preparation

Data Preprocessing

In data preprocessing step, we check if there missing data, duplicate values, and data types of each feature. Some column names are renamed, Non required column are drop.

Exploratory Data Analysis

This step includes bivariate and univariate analysis of features. Checking outliers using boxplots, and outlier treatment is carried out as well. Distribution of numerical values is plotted to see to what extent our data is skewed.

Feature Engineering

For Feature I have use this columns as feature 'Item\_Weight','Item\_Fat\_Content','Item\_Visibility','Item\_Type','Item\_MRP','Outlet\_Size','Outlet\_Location\_Type','Outlet\_Type','Years\_Established' And Item\_Outlet\_Sales this is target columns.

**Model Development**

Model implementation

After train and test splitting, pipeline containing Standard Scaler and Ordinal Encoder was fitted to several models such as AdaBoost Regressor, Gradient Boosting Regressor, RandomForest Regressor, XGB Regressor. Their R2 score were obtained. The highest score is acquired from the XGBoost Model.

Model Evaluation

Test dataset is used to evaluate the model. 20% of dataset was separated for testing. Predicted results of the model are compared with the actual data to check the amount of error.

**Deployment**

Designing UI with Flask

For this project, a user interface is built on Flask. It is a web application that helps us to create applications for projects. It is a free Python-based drag-and-drop web app builder.

Code deployment on cloud

The codes for this machine learning model should be deployed to the cloud, so that when data is entered into the application, our code runs and a user gets the result online.

**Deployment Process**

In this stage, we establish a server using Flask that runs the uplink code (server code) in parallel, where our model is executing, via an uplink. We will post the hole after execution or asynchronous execution. Git and Github are used to code in the Render cloud. Then, we'll configure a cron job to maintain the server and server code in operation indefinitely.

**Test Case Description**

Verify whether the Application URL is

accessible to the user

Verify whether the Application loads completely for the user when the URL is accessed

Verify whether user is able to see input fields.

Verify whether user is able to edit all input fields

Verify whether user gets Submit button to submit the inputs

Verify whether user is presented with results on clicking submit

Verify whether the results are in accordance to the selections user made

**Pre-Requisite**

Application URL should be defined

1. Application URL is accessible 2. Application is deployed

**Expected Result**

Application URL should be accessible to the user

The Application should load completely for the user when the URL is accessed

User should be able to see input fields

User should be able to edit all input fields

User should get Submit button to submit the inputs

User should be presented with results on clicking submit

The results should be in accordance to the selections user made