

**SALES PRICE PREDICTION**

Architectural Design

Domain: Machine Learning

Creator: Aravind S

Date: 23.07.2023

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| July 23, 2023 | 1.1 | First Draft | Aravind S |

**Document Version Control**

Contents

[Introduction 3](#_Toc141031866)

[What is Architectural Design? 3](#_Toc141031867)

[Scope 3](#_Toc141031868)

[Architecture 4](#_Toc141031869)

[Process Flow 4](#_Toc141031870)

[Deployment Process 4](#_Toc141031871)

[Data Description 5](#_Toc141031872)

[Dataset 5](#_Toc141031873)

[Exception and logging file 6](#_Toc141031874)

[Technology stack 6](#_Toc141031875)

[Data preprocessing 6](#_Toc141031876)

[Feature Engineering 7](#_Toc141031877)

[Hyperparameter Tuning 7](#_Toc141031878)

[Model Building 7](#_Toc141031879)

[Deployment using Streamlit 8](#_Toc141031880)

[GitHub repository 8](#_Toc141031881)

[Deployment 8](#_Toc141031882)

# Introduction

## What is Architectural Design?

The objective of the Low-Level Design Document (LLDD) is to provide an internal logical design of the 'Stores Sales Prediction' program code. The architectural design of the Sales Price Prediction system encompasses the system's overall structure and components. It outlines how data flows within the system and defines the interactions between various modules. The design incorporates machine learning algorithms, data preprocessing, and model evaluation techniques. It includes class diagrams illustrating the relationships between classes and methods for model training and prediction. The system's scalability, flexibility, and performance are considered in the design. Additionally, it outlines the integration of web UI for user interaction and deployment on cloud platforms for efficient utilization of resources. The architectural design ensures a robust and effective sales price prediction solution.

## Scope

The scope of architectural design involves defining the high-level structure and components of a system, including data flow, module interactions, and system functionalities. It considers scalability, performance, and integration of technologies for efficient development and successful system implementation.

# Architecture

## Process Flow

## Deployment Process

Architecture Description

## Data Description

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Measurement |
| Item\_Identifier | String | Unique product ID |
| Item\_Weight | Float | Weight of product |
| Item\_Fat\_Content | String | Whether the product is low fat or not |
| Item\_Visibility | Float | The % of a total display area of all products in a store allocated to the particular product |
| Item\_Type | String | The category to which the product belongs |
| Item\_MRP | Float | Maximum Retail Price (list price) of the product |
| Outlet\_Identifier | String | Unique store ID |
| Outlet\_Establishment\_Year | Integer | The year in which the store was established |
| Outlet\_Size | String | The size of the store in terms of ground area covered |
| Outlet\_Location\_Type | String | The type of city in which the store is located |
| Outlet\_Type | String | Whether the outlet is just a grocery store or some sort of supermarket |
| Item\_Outlet\_Sales | Float | Sales of the product in the particular store. This is the outcome variable to be predicted. |

## Dataset

Data source: [**https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data**](https://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data)

Train and Test data are stored in .csv format.

## Exception and logging file

Logging is a crucial aspect of the system, capturing user activities and system flow. The system identifies the necessary steps for logging, providing comprehensive insights. Developers can opt for logging methods, including database logging, to track system behavior effectively. Despite extensive logging, the system should maintain optimal performance, ensuring smooth operation. Logging is essential for efficient debugging and issue resolution, making it a mandatory aspect of the system's development process.

## Technology stack

|  |  |
| --- | --- |
| Front End | Streamlit |
| Back End | Python |
| Deployment | Streamlit |

## Data preprocessing

Data preprocessing involves essential tasks performed before using the data for model building. In this context, the 'Item Visibility' attribute had some values equal to 0, which is inappropriate since an item in the market should not have zero visibility. To address this, these zero values were replaced with the average visibility of items within the respective 'Item Identifier' category.

Additionally, two new attributes were introduced: 'Outlet years', which calculates the number of years since the establishment year by subtracting it from the current year; and 'Item Type', which extracts the first two characters from the 'Item Identifier', indicating the type of items.

Moreover, the 'Fat content' attribute was mapped to categorical values ('Low', 'Regular') for better representation and analysis.

By performing these preprocessing steps, the data is now ready for effective model building and analysis.

## Feature Engineering

After preprocessing the data, it was observed that certain attributes do not significantly impact the item sales for the particular outlet. Consequently, these non-important attributes were removed from the dataset to streamline and optimize the modeling process.

Furthermore, to handle categorical features, the technique of one-hot encoding was applied. This conversion process transformed categorical attributes into numerical features, making them compatible with machine learning algorithms that typically work with numerical data.

By removing irrelevant attributes and converting categorical features into numerical representations, the dataset is now better prepared for model building and analysis, focusing on the most relevant information to predict item sales for each outlet accurately.

## Hyperparameter Tuning

Hyperparameter tuning with GridSearchCV involves systematically searching through a predefined hyperparameter grid, evaluating model performance via cross-validation, and selecting the best combination of hyperparameters to optimize model accuracy or performance. It automates the process and ensures the most effective hyperparameters are chosen.

## Model Building

After conducting all the preprocessing steps, including scaling and hyperparameter tuning, the dataset is split and fed into various models such as Linear Regression, Random Forest Regressor, XGBoost, Catboost, Gradient Boosting, Ada boosting. The evaluation results reveal that the Catboost regressor performs the best among all, achieving the highest R-squared (R2) score of 0.62. Thus, the Catboost model is the preferred model for this specific problem, demonstrating superior performance.

## Deployment using Streamlit

After saving the trained model, the process of building a Web UI using streamlit begins. This involves creating a web application where users can input data. The web application extracts the user-entered data, which is then passed to the saved model for sales prediction. The model processes the data and returns the sales prediction to the user through the web application. This stage enables real-time sales predictions based on user input.

## GitHub repository

Pushing a project to a GitHub repository allows for version control, collaboration, and easy access to project files. It enables teams to work together, track changes, and maintain a history of project updates. Additionally, it serves as a secure backup and facilitates seamless deployment to production environments.

## Deployment

The cloud environment was set up and the project was deployed from GitHub into the Streamlit cloud platform.

App link-