Low Level Design (LLD)

Stores Sales Prediction

Revision Number – 1.0

Last Date of Revision – 04/09/2024

Swapnil Shinde

Document Version Control

Date	Version	Description	Author
27-08-2024	1.0	Abstract, Introduction Architecture	Swapnil Shinde
27-08-2024	1.1	Data Preprocessing	Swapnil Shinde
04-09-2024	1.2	Deployment	Swapnil Shinde

Contents

Abstract

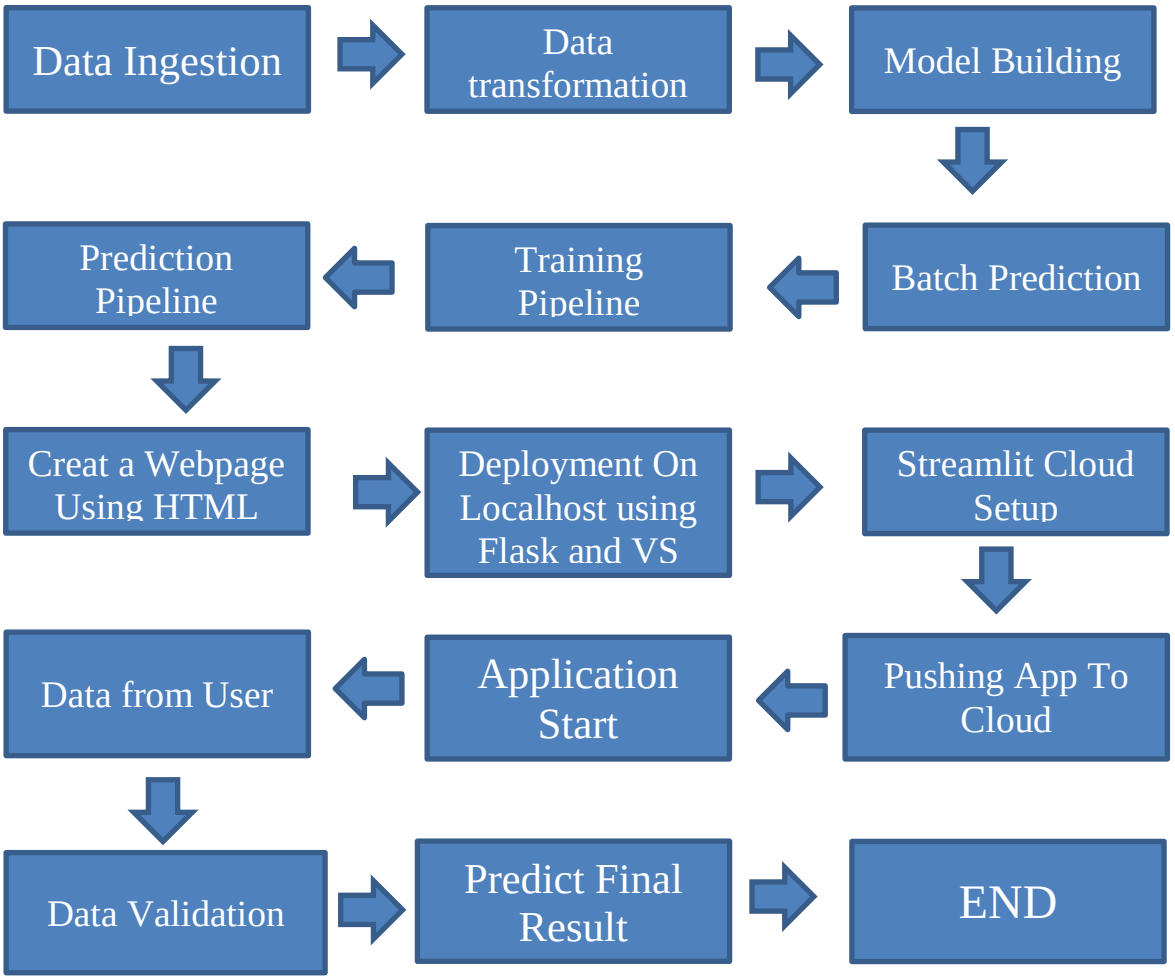
This Low-Level Design Document encapsulates the development of a comprehensive system poised to revolutionize inventory management and customer demand forecasting within the retail landscape. In the contemporary retail ecosystem, shopping malls and Big Marts meticulously collect and archive individual item sales data, yielding invaluable insights into consumer behavior and product specifics. These data repositories, securely stored within a data warehouse, serve as reservoirs of opportunity. Our system embarks on a transformative journey through the realms of data, expertly harnessing its power. The journey unfolds through meticulously orchestrated steps: data ingestion from Kaggle datasets, data transformation for cleanliness and relevance, model building to extract meaningful patterns, and the establishment of an efficient batch prediction pipeline. We don't stop there; we extend this journey to the end-users with a well-crafted, user-friendly interface, bridging the gap between data and actionable insights. This document, a testament to our technical prowess, delves deep into the modular architecture, interfaces, algorithms, and visualizations that underpin this transformative solution, setting the stage for a future where anomalies and common patterns emerge as strategic assets in the world of retail..

1 Introduction

1.1 Why this Low-Level Design Document?

The main purpose of this LLD documentation is to feature the required details of the project and supply the outline of the machine learning model and also the written code. This additionally provides the careful description on however the complete project has been designed end-to-end.

1.2 Architecture



2.Architecture Description

2.1. Data Gathering

The data for the current project is being gathered from Kaggle dataset, the link to the data is: BigMart Sales Data | Kaggle

2.2. Tool Used

- Python 3.9 is employed because the programming language and frame works like numpy, pandas, sklearn, Flask, Streamlit and alternative modules for building the model.
- Visual Studio Code is employed as IDE.
- For code versioning GitHub is used
- Localhost and Streamlit Cloud is employed for deployment

2.3 Data Description

We have train (8523) and test (5681) data set, train data set has both input and output

Columns Are :
variable(s). We need to predict the sales for test data set.
Item_Identifier: Unique product ID
Item_Weight: Weight of product
Item_Fat_Content: Whether the product is low fat or not
Item_Visibility: The % of total display area of all products in a store allocated to the
particular product
Item_Type: The category to which the product belongs
Item_MRP: Maximum Retail Price (list price) of the product
Outlet_Identifier: Unique store ID
Outlet_Establishment_Year: The year in which store was established
Outlet_Size: The size of the store in terms of ground area covered
Outlet_Location_Type: The type of city in which the store is located
Outlet_Type: Whether the outlet is just a grocery store or some sort of supermarket
Item_Outlet_Sales: Sales of the product in the particulat store. This is the outcome

variable to be predicted.

	Item_Ider	Item_Wei	Item_Fat_	Item_Visil	Item_Type	Item_MRF	Outlet_Id	Outlet_Es	Outlet_Si	Outlet_Lo	Outlet_Ty	Item_Outlet_Sales
2	FDA15	9.3	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermar	3735.138
3	DRC01	5.92	Regular	0.019278	Soft Drink	48.2692	OUT018	2009	Medium	Tier 3	Supermar	443.4228
4	FDN15	17.5	Low Fat	0.01676	Meat	141.618	OUT049	1999	Medium	Tier 1	Supermar	2097.27
5	FDX07	19.2	Regular	0	Fruits and	182.095	OUT010	1998		Tier 3	Grocery St	732.38
6	NCD19	8.93	Low Fat	0	Household	53.8614	OUT013	1987	High	Tier 3	Supermar	994.7052
7	FDP36	10.395	Regular	0	Baking Go	51.4008	OUT018	2009	Medium	Tier 3	Supermar	556.6088
8	FDO10	13.65	Regular	0.012741	Snack Foo	57.6588	OUT013	1987	High	Tier 3	Supermar	343.5528
9	FDP10		Low Fat	0.12747	Snack Foo	107.7622	OUT027	1985	Medium	Tier 3	Supermar	4022.764
10	FDH17	16.2	Regular	0.016687	Frozen Fo	96.9726	OUT045	2002		Tier 2	Supermar	1076.599
11	FDU28	19.2	Regular	0.09445	Frozen Fo	187.8214	OUT017	2007		Tier 2	Supermar	4710.535
12	FDY07	11.8	Low Fat	0	Fruits and	45.5402	OUT049	1999	Medium	Tier 1	Supermar	1516.027
13	FDA03	18.5	Regular	0.045464	Dairy	144.1102	OUT046	1997	Small	Tier 1	Supermar	2187.153
14	FDX32	15.1	Regular	0.100014	Fruits and	145.4786	OUT049	1999	Medium	Tier 1	Supermar	1589.265
15	FDS46	17.6	Regular	0.047257	Snack Foo	119.6782	OUT046	1997	Small	Tier 1	Supermar	2145.208
16	FDF32	16.35	Low Fat	0.068024	Fruits and	196.4426	OUT013	1987	High	Tier 3	Supermar	1977.426
17	FDP49	9	Regular	0.069089	Breakfast	56.3614	OUT046	1997	Small	Tier 1	Supermar	1547.319
18	NCB42	11.8	Low Fat	0.008596	Health an	115.3492	OUT018	2009	Medium	Tier 3	Supermar	1621.889
19	FDP49	9	Regular	0.069196	Breakfast	54.3614	OUT049	1999	Medium	Tier 1	Supermar	718.3982
20	DRI11		Low Fat	0.034238	Hard Drink	113.2834	OUT027	1985	Medium	Tier 3	Supermar	2303.668
21	FDU02	13.35	Low Fat	0.102492	Dairy	230.5352	OUT035	2004	Small	Tier 2	Supermar	2748.422
22	FDN22	18.85	Regular	0.13819	Snack Foo	250.8724	OUT013	1987	High	Tier 3	Supermar	3775.086
23	FDW12		Regular	0.0354	Baking Go	144.5444	OUT027	1985	Medium	Tier 3	Supermar	4064.043
24	NCB30	14.6	Low Fat	0.025698	Household	196.5084	OUT035	2004	Small	Tier 2	Supermar	1587.267
25	FDC37		Low Fat	0.057557	Baking Go	107.6938	OUT019	1985	Small	Tier 1	Grocery St	214.3876

2.4 Data Ingestion

The cornerstone of our data-driven project was established through a systematic process of data acquisition and ingestion. Utilizing Kaggle, a reputable platform renowned for its high-quality datasets, we identified and acquired the crucial data required for our price prediction project. This dataset, integral to our goal of accurate price forecasting, was meticulously downloaded and securely stored within our local system infrastructure. Subsequently, we initiated the data ingestion phase, where the dataset seamlessly integrated into our project's data pipeline. This meticulous approach ensures that our project is built upon a solid foundation, setting the stage for robust and precise price prediction models and analysis.

2.5 Data Transformation

Steps performed in pre-processing are:

- First read data from Artifact folder
- Checking unnecessary columns
- One column has product id which is unique for every product so I deleted that column.
- Checked for null values
- there are too many null values are present in two columns that’s why I deleted them
- Performed one-hot encoder on categorical columns.
- Perform Ordinal Encoder on Ordinal Columns.
- Scaling is performed for needed information.
- And, the info is prepared for passing to the machine learning formula

2.6 Modelling

The pre-processed information is then envisioned and every one the specified insights are being drawn. though from the drawn insights, the info is at random unfold however still modelling is performed with completely different machine learning algorithms to form positive we tend to cowl all the chances. and eventually, Gradient Boosting performed well .

2.7 Batch Prediction

In the pursuit of creating a comprehensive and efficient system, we have successfully executed batch prediction as a pivotal component of our project. Leveraging a meticulously designed data transformation pipeline, we have harnessed the power of our predictive model to generate accurate and timely batch predictions. This milestone signifies the culmination of our efforts in seamlessly processing and analyzing data, resulting in actionable insights that drive informed decision-making. As we prepare our Low-Level Design Document, this achievement underscores the significance of our data transformation pipeline and predictive model, which will be elaborately detailed to ensure clarity and scalability in our system architecture.

2.8 Training And Prediction Pipeline

In our endeavor to create a robust and end-to-end data-driven solution, we have meticulously crafted both a training pipeline and a prediction pipeline. The training pipeline serves as the backbone for developing our predictive models, allowing us to iteratively train and fine-tune them with the highest precision possible. Meanwhile, the prediction pipeline enables us to seamlessly apply these trained models to new data, ensuring that our insights and forecasts remain consistently accurate and adaptable to real-

world scenarios. This dual pipeline approach embodies our commitment to providing a comprehensive, data-driven solution that empowers decision-makers with the most reliable and up-to-date information. As we delve into the creation of our Low-Level Design Document, we will intricately detail these pipelines, showcasing their sophistication and efficiency in our system architecture.

2.9 UI Integration

Both CSS and HTML files are being created and are being integrated with the created machine learning model. All the required files are then integrated to the app.py file and tested locally

2.3 Data from User

The data from the user is retrieved from the created HTML web page and Streamlit application .

2.4 Data Validation

The data provided by the user is then being processed by app.py and application.py(streamlit application) file andvalidated. The validated data is then sent for the prediction.

2.11 Rendering Result

The data sent for the prediction is then rendered to the web page.

3. Deployment

The tested model is then deployed on local machine and Streamlit Cloud. So, users can access the project from any internet devices.

3.1 Unit Test

Test Case	Description Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. 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	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets Submit button to submit the inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should get Submit button to submit the inputs
Verify whether user is presented with Predicted results on clicking submit	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be presented with Predicted results on clicking submit

Low Level Design (LLD)

11