Low Level Design

Amazon Sales Data Analysis

Revision Number: 1.0 Last
Date of revision: 10/02/2024

Bodapatla Vinay Kumar Reddy

Document Version Control

Date Issued	Version	Description	Author
10th Feb 2024	1.0	First Version of Complete LLD	Vinay Kumar Reddy Bodapatla

Contents

1. Introduction

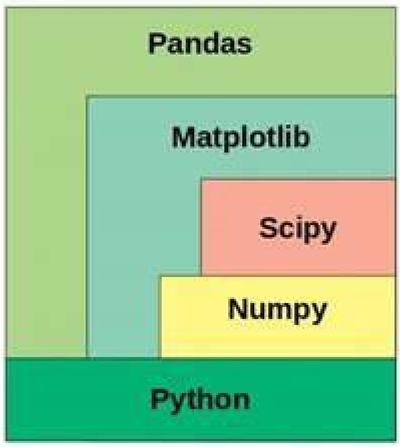
1.1 What is a Low-Level design document?

The goal of the LDD or Low-level design document (LLDD) is to give the internal logic design of the actual program code for the House Price Prediction dashboard. LDD describes the class diagrams with the methods and relations between classes and programs specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2. Architecture



NumPy Architecture Design Document:

Introduction:
Overview of NumPy as a numerical computing library for Python.

Architectural Goals and Objectives:
Provide efficient and flexible array operations for numerical computing. Facilitate vectorized operations and broadcasting.

Architectural Overview:
Description of N-dimensional array as the fundamental data structure. Explanation of universal functions (ufunc) for element-wise operations.

Key Design Decisions:
Adoption of N-dimensional array for efficient array manipulation. Implementation of universal functions for vectorized operations.

System Components:
N-dimensional array as the core component for storing and manipulating arrays. Universal functions for element-wise operations.

Data Design:
Discussion on how NumPy efficiently manages memory layout. Support for homogeneous, fixed-size arrays.

Performance Considerations:
Emphasis on vectorized operations for improved performance. Memory layout optimizations.

Scalability and Extensibility:
Discuss how NumPy supports scalability through efficient array operations. Extensibility through integration with other scientific libraries.

Matplotlib Architecture Design Document:

Introduction:
Overview of Matplotlib as a 2D plotting library for Python.

Architectural Goals and Objectives:
Provide a versatile and customizable plotting framework. Support diverse plot types and visualizations.

Architectural Overview:
Description of Figure and Axes structure. Explanation of backend rendering.

Key Design Decisions:
Adoption of a layered design for customization. Support for multiple backends for rendering.

System Components:
Figure as the top-level container. Axes representing individual plots within a Figure.

Data Design:
Handling and plotting of data using high-level functions. Data visualization through a variety of plot types.

Performance Considerations:
Backend rendering optimizations. Layered design for efficient customization.

Scalability and Extensibility:
Support for creating multiple plots within a single Figure. Customization and extensibility through high-level and low-level components.

Seaborn Architecture Design Document:

Introduction:
Overview of Seaborn as a statistical data visualization library for Python.

Architectural Goals and Objectives:
Simplify the creation of informative statistical plots. Enhance aesthetics and styling compared to Matplotlib.

Architectural Overview:
Built on top of Matplotlib. Integration with Pandas DataFrames.

Key Design Decisions:
Focus on statistical plotting functions. Aesthetic enhancements and built-in themes.
System Components:
Utilization of Matplotlib components for plotting. Statistical plotting functions specific to Seaborn.
Data Design:
Seamless integration with Pandas DataFrames. Statistical plotting for deeper insights.

Performance Considerations:
Leveraging Matplotlib backend rendering optimizations. Aesthetic enhancements without sacrificing performance.

Scalability and Extensibility:
Integration with Pandas for handling tabular data. Customization and extensibility through Matplotlib's capabilities.

3.Architecture Description

3.1 Data Loading

Step 1: - Open Jupyter and import python libraries as

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt # visualizing data

%matplotlib inline

import seaborn as sns
```

Step 2: -import csv file.

```
df = pd.read_csv('/Users/vinayreddy/Downloads/Python_Diwali_Sales_Analysis/Diwali Sales Data.csv')
```

3.2 Data Transformation

In the Transformation Process, we will convert our original datasets with other necessary attributes format and change the features according to the problem statement.

3.3 Data Modelling

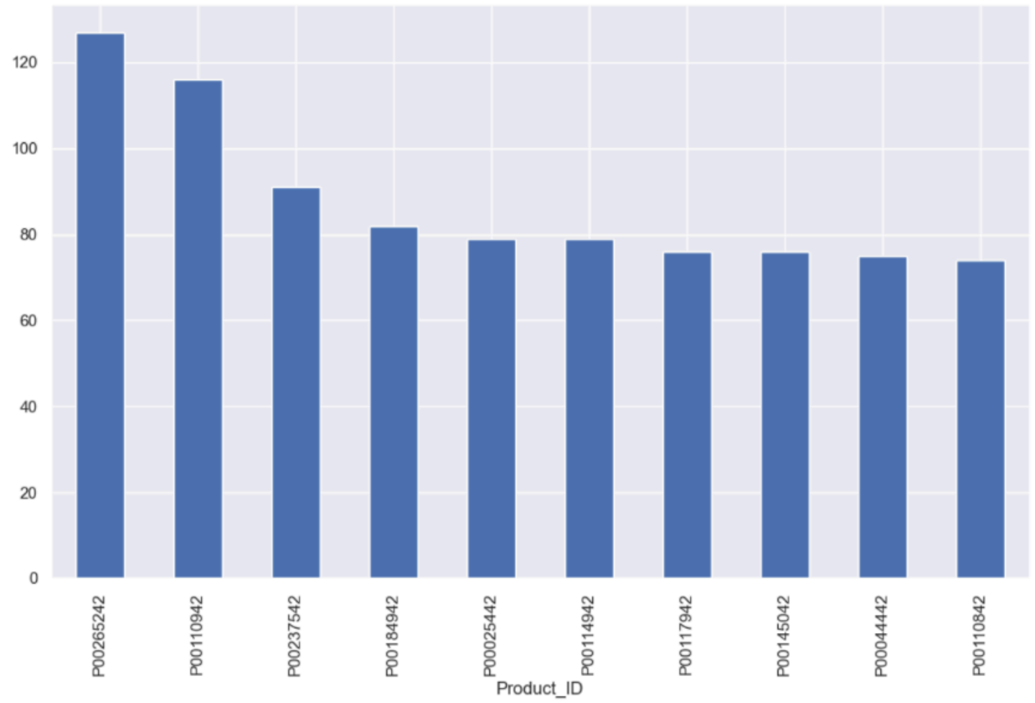
After the data is transformed the data is modeled for visualizing and analysis.

3.4 Data analysis and visualization

Analyzed and Visualized the Data.

3.5 Deployment

- At last conclusion was a Married women age group 26-35 yrs from UP, Maharashtra and Karnataka working in IT, Healthcare and Aviation are more likely to buy products from Food, Clothing and Electronics category


P00265242
P00110942
P00237542
P00184942
P00025442
P00114942
P00117942
P00145042
P00044442
P00110842
Product_ID