Low Level Design

Customer Segmentation : Purchasing capabilities of a customer

Written By	
Document Version	0.1
Last Revised Date	10th September, 2021

Document Control

Change Record:

Version	Date	Author	Comments
0.1	10th Sept 2021	Prasad Aher	Introduction & Architecture defined

Reviews:

Version	Date	Reviewer	Comments
0.1			

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

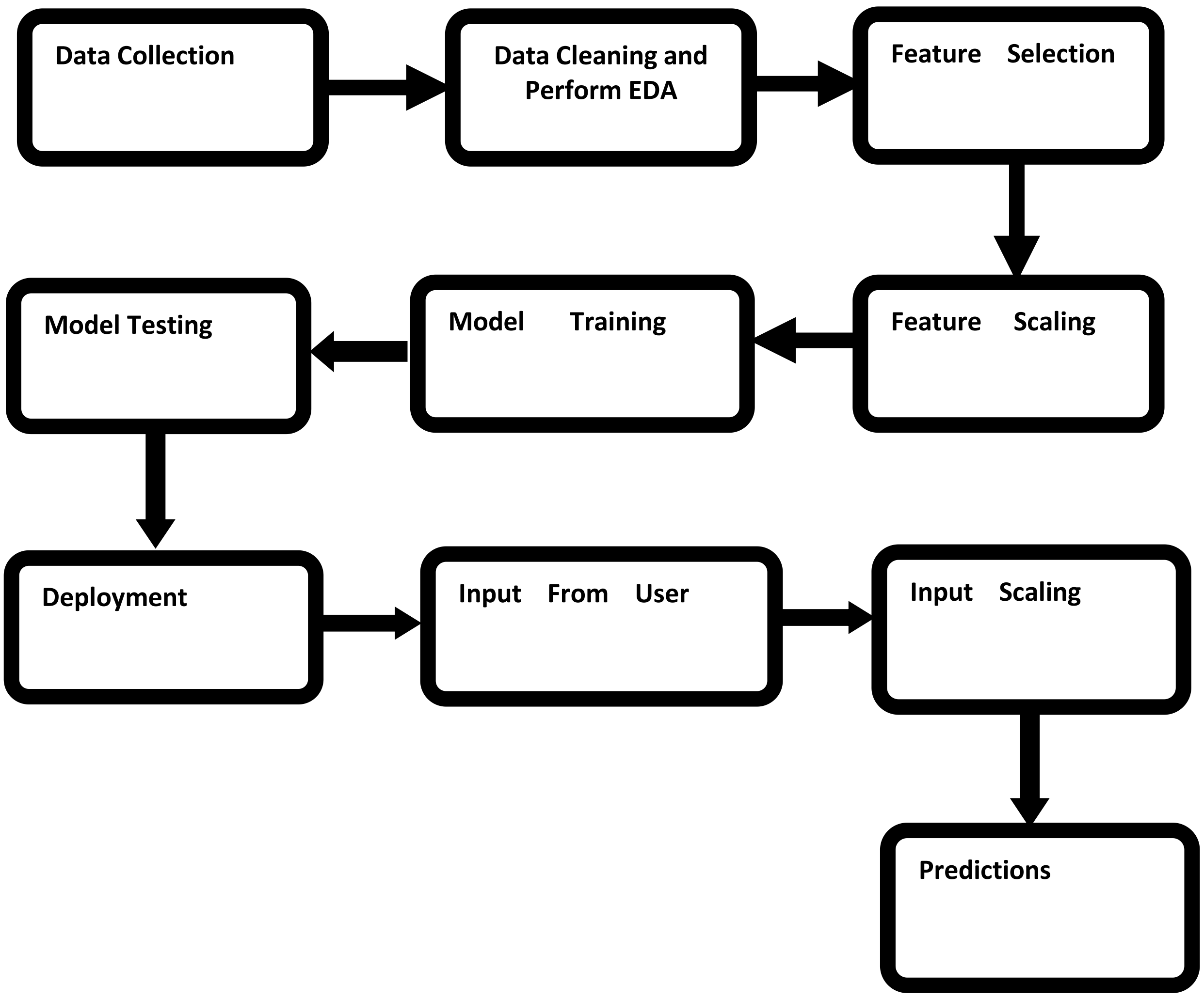
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Purchasing Capability of a Customer. LLD describes the class diagrams with the methods and relations between classes and program 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. This 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



3. Architecture Description

3.1. Data Description

This case requires developing a customer segmentation to define Purchasing Capability of Customer. The sample Dataset summarizes the usage behaviour of about 9000 active credit card holders over the period of 6 months. The file is at a customer level with 18 behavioural variables.

<u>Following is the Data Dictionary for Credit Card dataset: -</u>

CUSTID	**: Identification of Credit Card holder (Categorical)**
BALANCE	**: Balance amount left in their account to make purchases (numerical)**
BALANCEFREQUENCY	**: How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated)**
PURCHASES	**: Amount of purchases made from account**
ONEOFFPURCHASES	**: Maximum purchase amount done in one-go**
INSTALLMENTSPURCHASES	**: Amount of purchase done in instalment**
CASHADVANCE	**: Cash in advance given by the user**
PURCHASESFREQUENCY	**: How frequently the Purchases are being made, score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased)**
ONEOFFPURCHASESFREQUENCY	**: How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased)**
PURCHASESINSTALLMENTSFREQUENCY	**: How frequently purchases in instalments are being done (1 = frequently done, 0 = not frequently done)**
CASHADVANCEFREQUENCY	**: How frequently the cash in advance being paid**
CASHADVANCETRX	**: Number of Transactions made with "Cash in Advanced"**
PURCHASESTRX	**: Number of purchase transactions made**
CREDITLIMIT	**: Limit of Credit Card for user**
PAYMENTS	**: Amount of Payment done by user**
MINIMUM_PAYMENTS	**: Minimum amount of payments made by user**
PRCFULLPAYMENT	**: Percent of full payment paid by user**
TENURE	**: Tenure of credit card service for user**

3.2 Data Pre-processing

Unimportant variables were dropped and missing values were replaced with mean. Since we are building a clustering model, data cleaning is not required as the main goal of clustering is to capture the outliers and make them separate.

3.3 Scaling

The independent variables have been scaled using the Min Max scaling technique.

3.5 Model Building

Clustering Model:

Here we have used K-means clustering for the segmentation of customers possessing similar characteristics with respect to spending behavior. Three clusters of customers have been created.

3.6 Deployent

We will be deploying the model to Heroku.

3.7 Input from User

Here we will collect data from user such as per description mentioned in Data Set Description. That data will be scaled down by Min Max scaler model, then this scaled data will be used as an input to our model.

3.8 Prediction

The prediction model will predict the cluster number and purchase amount of the customer based on the input to our model.

4. Unit Test Cases

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 the User is able to sign up in the application	1. Application is accessible	The User should be able to sign up in the application
Verify whether user is able to successfully login to the application	1. Application is accessible 2. User is signed up to the application	User should be able to successfully login to the application
Verify whether user is able to see input fields on logging in	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 see input fields on logging in
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 recommended 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 recommended results on clicking submit

Verify whether the recommended results are in accordance to the selections user made	1. **Application is accessible** 2. **User is signed up to the application** 3. **User is logged in to the application**	**The recommended results should be in accordance to the selections user made**
Verify whether user has options to filter the recommended results as well	1. **Application is accessible** 2. **User is signed up**	**User should have options to filter the recommended results as well**

	to the application 3. User is logged in to the application	
Verify whether KPIs modify as per the user inputs for the user's health	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	KPIs should modify as per the user inputs for the user's health
Verify whether the KPIs indicate details of the suggested recipe	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The KPIs should indicate details of the suggested recipe