Low-Level Design

Travel Package Purchase Prediction

|  |  |
| --- | --- |
| Written By | Ubaid Ahmad Khanday |
| Version | 1.0 |
| Date | 08-02-2024 |

**Document Change Control Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Reviews:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

### Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review**  **Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

**Index**

|  |  |
| --- | --- |
| **Content** | **Page No** |
| 1. Introduction | 2 |
| 1.1 What is Low-Level Design Document | 2 |
| 1.2 Scope | 2 |
| Architecture | 3 |
| 2. Architecture Description | 3 |
| 2.1 Data Description | 3 |
| 2.2 Data Gathering | 4 |
| 2.3 Raw Data Validation | 4 |
| 2.4 Data Transformation | 4 |
| 2.5 Data Preprocessing | 4 |
| 2.6 Feature Engineering | 5 |
| 2.7 Pipelining |  |
| 2.8 Parameter Tuning | 5 |
| 2.9 Model Building | 5 |
| 2.10 Model Saving | 5 |
| 2.11 GitHub | 6 |
| 2.12 Deployment | 6 |
| 3. Unit Test Cases | 6 |

**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 **‘Travel Package Purchase Prediction’**. 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](https://en.wikipedia.org/wiki/Refinement_(computing)) 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.

**Architecture :**

**2. Architecture Description**

**2.1 Data Description**

Given is the variable name, variable type, the measurement unit, and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Measurement |
| Customer\_id | int | Unique customer ID |
| Prod\_Taken | int | Whether the customer has purchased a package or not (0: No, 1: Yes) |
| Age | float | Age of Customer |
| TypeofContact | object | How customer was contacted (Company Invited or Self Inquiry) |
| CityTier | int | City tier depends on the development of a city, population, facilities, and living standards. The categories are ordered i.e. Tier 1 > Tier 2 > Tier 3 |
| Occupation | object | Occupation of customer |
| Gender | object | Gender of customer |
| NumberOfPersonVisiting | Int | Total number of persons planning to take the trip with the customer |
| PreferredPropertyStar | float | Preferred hotel property rating by customer |
| MaritalStatus | object | Marital status of customer |
| NumberOfTrips | float | Average number of trips in a year by customer |
| Passport | int | The customer has a passport or not (0: No, 1: Yes) |
| OwnCar | int | Whether the customers own a car or not (0: No, 1: Yes) |
| NumberOfChildrenVisiting | float | Total number of children with age less than 5 planning to take the trip with the customer |
| MonthlyIncome | float | Gross monthly income of the customer |
| Designation | object | Designation of the customer in the current organization |

**2.2 Data Gathering**

Data source: <https://question.transtutors.com/6129343_1_tourism-data.xlsx>

Train and Test data are stored in .csv format.

**2.3 Raw Data Validation**

After data is loaded, various types of validation are required before we proceed further with any operation. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because The attributes which contain these are of no use. It will not play role in contributing to the sales of an item from respective outlets.

Like if any attribute is having zero standard deviation, it means that’s all the values are the same, its mean is zero. This indicates that either the sale is increasing or decrease that attribute will remain the same. Similarly, if any attribute is having full missing values, then there is no use in taking that attribute into an account for operation. It’s unnecessary increasing the chances of dimensionality curse.

**2.4 Data Transformation**

Before sending the data into the database, data transformation is required so that data are converted into such form with which it can easily insert into the database. Here, the ‘Age’,’Duration of Pitch’and “Monthly Income’ attributes contain the missing values. So they are filled in both the train set as well as the test set with supported appropriate data types.

**2.5 Data Preprocessing**

In preparation for model building, the customer data underwent thorough preprocessing. Missing values were handled based on data type and distribution (e.g., imputed with mean for numerical features, filled with mode for categorical features, or potentially removed if significant). Invalid values were corrected or removed depending on severity and impact. Outliers were identified and addressed and removed based on their influence on analysis.

Furthermore, feature scaling and normalization were applied to ensure all features operated on a similar scale, improving the effectiveness of the model building process.

**2.6 Feature Engineering**

After preprocessing it was found that some of the attributes are not important to the item sales for the particular outlet. So those attributes are removed. Even one hot encoding is also performed to convert the categorical features into numerical features.

**2.7 Pipelining**

In my project's preprocessing phase, we established separate pipelines for numerical and categorical features. The numerical pipeline handles tasks like imputation and scaling, while the categorical pipeline employs techniques like one-hot encoding. This tailored approach ensures that each feature type is appropriately processed, optimizing model performance and interpretability. By streamlining the preprocessing process, our pipelines contribute to the efficient transformation of the dataset and enhance the accuracy of our predictive models.

**2.8 Parameter Tuning**

Parameters are tuned using Grid searchCV. Many algorithms were used in this problem, logistic Regression ,Decision tree, XGBoost, SVM, AdaBoost,Random Forest etc. The parameters of these algorithms were tunned and passed into the model.We got the best accuracy from XGBoost Classifier with training accuracy of 99% and the testing accuracy of 92.5%.

**2.9 Model Building**

After doing all kinds of preprocessing operations mention above and performing scaling and hyperparameter tuning, the data set is passed into these models,

**2.10 Model Saving**

Model is saved using pickle library in `.pkl` format.

**2.11 Django Setup for Data Extraction**

After saving the model, the API building process started using Flask. Web application creation was created here. Whatever the data user will enter and then that data will be extracted by the model to predict the prediction of sales, this is performed in this stage.

**2.13 GitHub**

The whole project directory will be pushed into the GitHub repository.

**2.14 Deployment**

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

App link- https://x9rs73j6e7.us-east-1.awsapprunner.com/

**3. 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 a user is able to see input fields while opening the application | 1. Application is accessible 2. The user is able to see the input fields | Users should be able to see input fields on logging in |
| Verify whether a user is able to enter the input values. | 1. Application is accessible 2. The user is able to see the input fields | The user should be able to fill the input field |
| Verify whether a user gets predict button to submit the inputs | 1. Application is accessible 2. The user is able to see the input fields | Users should get Submit button to submit the inputs |
| Verify whether a user is presented with recommended results on clicking submit | 1. Application is   accessible   1. The user is able to see the input fields. 2. The user is able to see the submit button | Users should be presented with recommended results on clicking submit |
| Verify whether a result is in accordance with the input that the user has entered | 1. Application is accessible 2. The user is able to see the input fields. 3. The user is able to see the submit button | The result should be in accordance with the input that the user has entered |