

**Clustering For Customer Segmentation & Understanding**

Low Level Design Document

Domain: Machine Learning

Creator Rumanshu Chandekar

Shreyas Dongre

Date: 30.04.2022

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| April 30, 2024 | 1.1 | First Draft | Shreyas Dongre  Rumanshu Chandekar |
| May 20, 2024 | 1.2 | Added unit test cases | Shreyas Dongre  Rumanshu Chandekar |

**Document Version Control**

Contents

[Introduction 3](#_Toc110433758)

[What is Low-Level Design Document? 3](#_Toc110433759)

[Scope 3](#_Toc110433760)

[Architecture 4](#_Toc110433761)

[Architecture Description 5](#_Toc110433762)

[Data Preparation 5](#_Toc110433763)

[Data Description 5](#_Toc110433764)

[Data Preprocessing 5](#_Toc110433765)

[Exploratory Data Analysis 5](#_Toc110433766)

[Feature Engineering 5](#_Toc110433767)

[Model Development 6](#_Toc110433768)

[Model implementation 6](#_Toc110433769)

[Hyper-parameter Tuning 6](#_Toc110433770)

[Model Evaluation 6](#_Toc110433771)

[Deployment 6](#_Toc110433772)

[Designing UI with Streamlit 6](#_Toc110433773)

[Designing a server 6](#_Toc110433774)

[Code deployment on cloud 7](#_Toc110433775)

[Deployment Process 7](#_Toc110433776)

[Unit cases 8](#_Toc110433777)

# Introduction

Project Overview: This project aims to segment customers using clustering techniques to enhance business strategies and customer service. The PyCaret library is utilized for its efficient and effective clustering capabilities.

Objectives: The main objectives include identifying distinct groups within the customer base to tailor marketing strategies, improve customer service, and enhance product recommendations.

## What is Low Level Design Document?

The goal of LLD or a low-level design document is to give the internal logical of the actual program code for Metro Interstate Traffic Volume Prediction. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

The main objective of the project is to predict if the traffic volume is high or low on the date. Weather circumstances, special days like holidays, daytime (morning, afternoon, night etc.), a temperature, a weekday, a numeric percentage of cloud cover are vital attributes for predicting traffic volume.

## 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.

# Architecture

**Data Preparation**

**Model**

**Development**

**Deployment**

**Deployment**

# Architecture Description

## Data Preparation

### Data Description

### In our project's architecture, we efficiently gather and process customer data, ensuring optimal use of computing resources. This involves understanding various customer attributes, such as purchase history and demographics. By optimizing data processing techniques, we ensure accurate insights for customer segmentation.

### Data Preprocessing

In the data preprocessing step, we check if there is missing data, duplicate values, and datatypes of each feature. By ensuring efficient resource allocation, we streamline data preprocessing tasks such as cleaning, transformation, and normalization, leading to accurate customer segmentation results.

### Exploratory Data Analysis

During the exploratory data analysis phase of our project, we ensure efficient resource utilization by optimizing computing resources. This involves effectively managing memory and processing power to analyze and visualize customer data. By leveraging computing resources efficiently, we gain valuable insights into customer behavior patterns, which inform our clustering process for segmentation accurately.

### Feature Engineering

During feature engineering, we optimize resource utilization by efficiently managing computing resources. This involves leveraging memory and processing power to extract relevant features from customer data. By maximizing resource efficiency, we enhance the accuracy of customer segmentation, leading to meaningful insights for businesses.

### Data Transformation

In data transformation, we prioritize efficient resource utilization by optimizing computing resources. This involves managing memory and processing power effectively to perform data transformations such as normalization or scaling. By maximizing resource efficiency, we ensure accurate data representation, leading to reliable customer segmentation results.

### DataInsertion into Database

During data insertion into the database, we focus on efficient resource utilization by optimizing computing resources. This involves effectively managing memory and processing power to ensure seamless insertion of customer data into the database. By maximizing resource efficiency, we maintain data integrity and accessibility for subsequent analysis and segmentation tasks.

### Export Datafrom Database

During the export of data from the database, our focus lies in efficient resource utilization. This ensures smooth and timely extraction of customer data from the database, facilitating further analysis and segmentation tasks. By maximizing resource efficiency, we maintain data integrity and accessibility for insightful decision-making.

### Data Clustering

In the data clustering phase, our focus is on making the most of our computing resources. We ensure that memory and processing power are used efficiently to run clustering algorithms smoothly. By optimizing resource utilization, we guarantee timely and precise segmentation of customer data. This enables us to extract valuable insights that can guide strategic decisions for businesses effectively.

Model Development

### Model Implementation.

### During model development and implementation using PyCaret, we focus on efficient resource utilization. PyCaret optimizes memory and processing power effectively to train and deploy clustering models. For each cluster, different predictive models are tested using PyCaret’s compare models function to find the best performer based on AUC scores.

### Hyper-parameter Tuning

The best parameters for each model are determined using PyCaret’s tune\_model function, which uses a grid search approach to optimize performance. The best model is chosen, and Grid Search with Cross Validation is applied on that model to get the best parameters. Those parameters are then used on the model to get better results.

### Model Evaluation

Test dataset is used to evaluate the model. 20% of dataset was separated for testing. Predicted results of the model are compared with the actual data to check the number of errors. As there was no considerable change after hyperparameter tuning, it helped us to overcome overfitting and perform better on new data.

### Model Finalization

The best model for each cluster is finalized and saved for deployment, ensuring that each segment has a tailored predictive model. PyCaret optimizes the finalization process to ensure the selected clustering model is trained on the entire dataset with the best parameters. By maximizing resource efficiency, we guarantee the production-ready model is accurately tuned and ready for deployment.

Data Validation

PyCaret optimizes the validation process by employing various techniques such as cross-validation, which helps in assessing the performance of clustering algorithms using multiple subsets of data. This approach ensures that the clustering model generalizes well to unseen data, enhancing its reliability. Additionally, PyCaret offers automated feature engineering and selection capabilities, which further improve the quality of data used for clustering. By maximizing resource efficiency, PyCaret streamlines the validation process, allowing for thorough examination of data quality and model performance. This meticulous validation process ultimately leads to more accurate insights derived from customer segmentation, enabling businesses to make informed decisions based on reliable data analysis.

Model Call for Specific Cluster

PyCaret efficiently manages computational resources during model call for specific clusters. It optimizes memory and processing power to swiftly retrieve insights for targeted customer segments. By streamlining resource utilization, PyCaret ensures timely and accurate analysis, empowering businesses to tailor strategies effectively. Moreover, PyCaret's advanced capabilities, like model caching, enhance efficiency by storing previously computed results, reducing redundant computations. This meticulous resource management enables seamless navigation through customer clusters, facilitating precise decision-making based on cluster-specific insights.

## Recipe Recommendation & Saving Output in Database

Recommendation Logic

PyCaret optimizes the recommendation process to generate personalized recipe recommendations based on clustering results. By maximizing resource efficiency, we ensure timely and accurate recommendation generation, enhancing customer segmentation insights and facilitating database output storage for future reference. Based on the output of the predictive model, personalized recommendations are generated for each customer. This could include product recommendations, promotional offers, or tailored content.

Data Integration

PyCaret optimizes the integration process to combine multiple data sources seamlessly. By maximizing resource efficiency, we ensure smooth data integration, enabling comprehensive analysis and accurate customer segmentation for businesses. All recommendations are stored in an SQL database, allowing for easy retrieval and analysis. This database also tracks customer responses to recommendations, which helps with refining future models.

## **Deployment**

### Designing UI with Flask:

For this project, a user interface is built using Flask. Flask is a free, framework which can be used using python that allows you to create interactive web applications for your projects quickly and easily.

### Designing a server:

### Server A server should be created to run the UI application continuously. An AWS EC2 instance is set up to host the Stream lit server, ensuring it runs continuously and handles user requests efficiently. Code Deployment on Cloud The code for this machine learning model should be deployed to the cloud.

### Code deployment on cloud:

## The code for this machine learning model should be deployed to the cloud. When data is entered into the application, the code runs, and the user gets the result online. AWS EC2 is used for this purpose.

## Deployment Process:

In this stage, we establish a server using AWS EC2 that runs the Stream lit application. The EC2 instance is configured to handle the necessary workload and ensure proper security settings are in place. The machine learning model and application code are pushed to a GitHub repository. GitHub Actions are used for continuous integration and continuous deployment (CI/CD) to automate the deployment process. The Stream lit application is deployed on the AWS EC2 instance, and the server is configured to run the application continuously. A cron job or AWS Lambda function is set up to maintain the server and ensure the application runs smoothly.

# Unit cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application is accessible to the user | Application should be defined | Application should be accessible to the user |
| Verify whether the Application loads completely for the user when the application is accessed | 1. Application is accessible  2. Application is deployed | The Application should load completely for the user when the application is accessed |
| Verify whether user can see input fields. | Application is accessible | Users should be able to see input fields |
| Verify whether user can edit all input fields | Application is accessible | Users should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | Application is accessible | Users should get Submit button to submit the inputs |
| Verify whether user is presented with results on clicking submit | Application is accessible | Users should be presented with results on clicking submit |
| Verify whether the results are in accordance with the selections user made | Application is accessible | The results should be in accordance with the selections user made |