High Level Design (HLD)
Customer Segmentation

Revision Number - 1.2

Last Date of Revision - 29/1/2022

AKASH

Document Version Control

Date Issued	Version	Description	Author
19/1/2022	1.0	Abstract, Introduction, General Description	Akash Shetty
23/1/2022	1.1	Design Detail, KPI, Deployment	Akash Shetty
29/1/2022	1.2	Final Revision	Akash Shetty

Contents

Abstract

Not all customers are same. To know which group is your customer and their preferences is a big part for success in your business. Unsupervised machine learning can help marketers to know their audience globally and engage them with their products accordingly.

Here we can classify millions of people's interests through their activity and also through other surveys online & offline and cluster them in specific group of their interest .

1 Introduction

1. Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions before coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:

  -Security
  -Reliability
  -Maintainability
  -Portability
  -Reusability
  -Application compatibility
  -Resource utilization
  -Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2 General Description

2.1 Product Perspective & Problem Statement

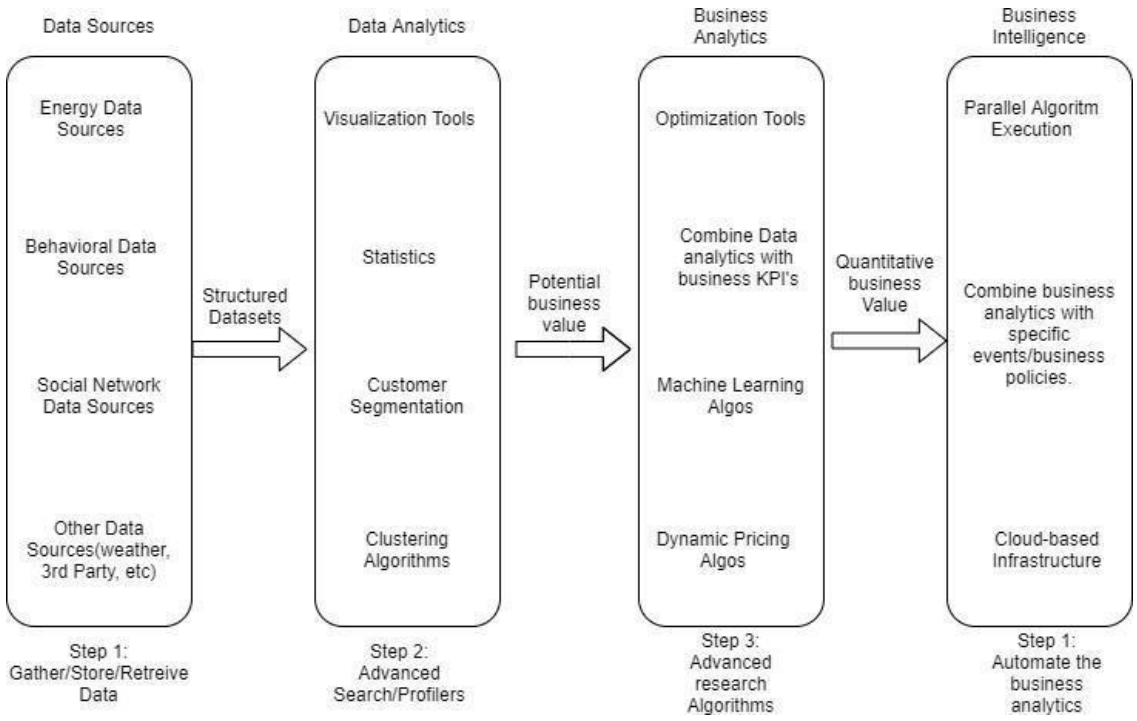
The goal of this project is to analyse to predict the group of cluster our customers belong, based on a combination of features that describes their spending and shopping habits. To achieve the goal, we used a data set that is formed by taking into consideration some of the information of 9000 individuals. The problem is based on the given information about each individual we have to calculate that whether that individual one of four different groups

2.2 Tools used

Business Intelligence tools and libraries works such as NumPy, Pandas, Seaborn, Matplotlib, MS-Excel, Jupyter Notebook and Python Programming Language are used to build the whole framework.

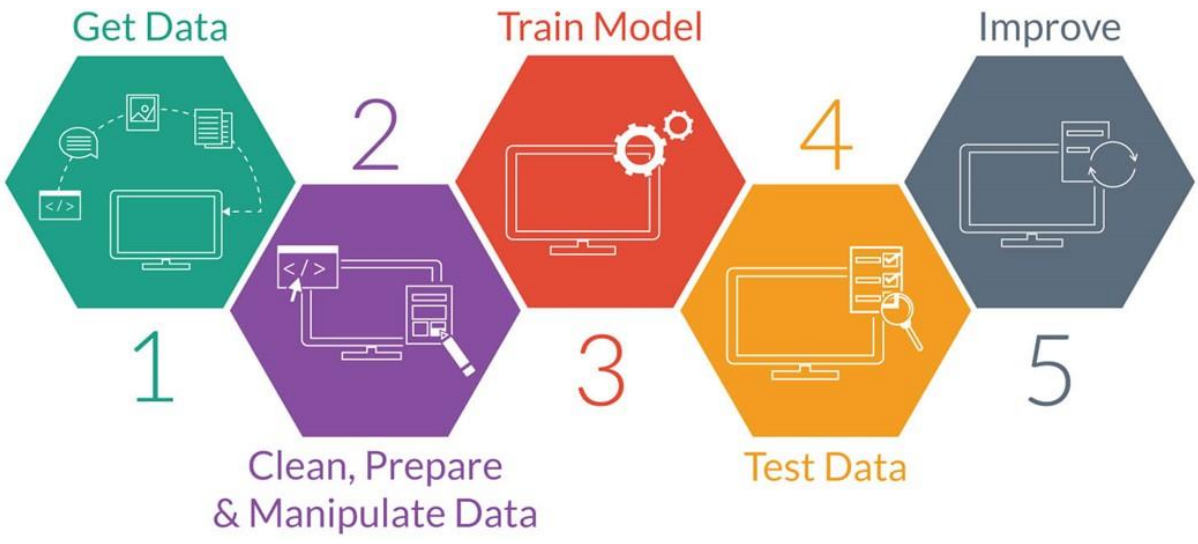
3 Design Details

1. Functional Architecture



Model Building

-



2. Optimization

1. Your data strategy drives performance

- Minimize the number of fields
- Minimize the number of records
- Optimize extracts to speed up future queries by materializing calculations, removing columns and the use of accelerated views

2. Reduce the marks (data points) in your view

- Practice guided analytics. There's no need to fit everything you plan to show in a single view. Compile related views and connect them with action filters to travel from overview to highly-granular views at the speed of thought.
- Remove unneeded dimensions from the detail shelf.
- Explore. Try displaying your data in different types of views.

3. Limit your filters by number and type

- Reduce the number of filters in use. Excessive filters on a view will create a more complex query, which takes longer to return results. Double-check your filters and remove any that aren't necessary.
- Use an include filter. Exclude filters load the entire domain of a dimension while including filters do not. An include filter runs much faster than an exclude filter, especially for dimensions with many members.
- Use a continuous date filter. Continuous date filters (relative and range-of-date filters) can take advantage of the indexing properties in your database and are faster than discrete data filters.
- Use Boolean or numeric filters. Computers process integers and Booleans (t/f) much faster than strings.
- Use parameters and action filters. These reduce the query load (and work across data sources).

4. Optimize and materialize your calculations

- Perform PCA
- Reduce the number of nested calculations.
- Reduce the granularity of LOD or table calculations in the view. The more granular the calculation, the longer it takes.

  ✓ LODs - Look at the number of unique dimension members in the calculation.

  ✓ Table Calculations - the more marks in the view, the longer it will take to calculate.

- Used elbow method to determine the best possible value of k in kmeans clustering.

- Make groups with calculations. Like include filters, calculated groups load only named members of the domain, whereas Tableau's group function loads the entire domain.

- Use Booleans or numeric calculations instead of string calculations. Computers can process integers and Booleans (t/f) much faster than strings. Boolean>Int>Float>Date>DateTime>String.

4 KPIs:

1. KPIs (Key Performance Indicators)

Key indicators displaying a summary of the customer segmentation and its relationship with different metrics

1. Percentage of People in different clusters.
2. Age Distribution including Gender and spending capacity.
3. F1 score (0.91)
4. Accuracy (0.94)
5. Confusion matrics

5 Deployment

With the help of Streamlit we created apps for our machine learning project using simple python scripts. It also supports hot-reloading, so that your app can update live as you edit and save your file. An app can be built in a few lines of code only(as we will see below) using the Streamlit API. Adding a widget is the same as declaring a variable , with defined different routes or handle HTTP requests. It is easy to deploy and manage.