**Documentation for MMM (Market Mix Modeling) Proof of Concept**

**Introduction:**

The MMM (Market Mix Modeling) Proof of Concept aims to develop a flexible and efficient solution for analyzing marketing campaigns and optimizing budgets based on real-time data. This documentation provides an overview of the approach taken and the user interface developed for the MMM PoC.

**Approach:**

1. Understanding MMM:

The project began by exploring the concept of Market Mix Modeling (MMM) to gain insights into its principles and methodologies. MMM involves analyzing the impact of various marketing channels on sales and optimizing budgets accordingly.

1. Exploring Python Modules:

Several Python modules designed for Market Mix Modeling, such as *hommoer*, *mamimo*, and *lightweight*-*mmm*, were investigated. However, none of these packages offered the desired flexibility to handle diverse real-time data.

1. Paper Implementation:

The exploration led to the identification of a repository that implemented Google's paper on "Bayesian Methods for Media Mix Modeling with Carryover and Shape Effects." This existing implementation from a previous PoC formed the basis for the MMM PoC, with an emphasis on enhancing code flexibility to handle diverse data formats. Leveraging the Stan language used in the paper, the Pystan library was employed to seamlessly integrate the Stan code into Python, facilitating efficient MMM calculations.

**User Interface:**

A user-friendly interface was developed using Streamlit to facilitate the interaction with the MMM PoC. The interface offers the following functionalities:

1. Data Upload:

Users can upload weekly data files containing information on impressions and spending for various media channels. Additionally, users can specify the budget, the number of weeks for calculations, and constraints for each channel.

1. Sales Graph Visualization:

The interface provides a visual representation of sales data over the years. This graph allows users to observe trends and patterns in sales, helping them make informed decisions.

1. Model Training:

Users can train the MMM model using the uploaded data. Upon clicking the "Train Model" button, the model will calculate the Return on Advertising Spend (ROAS) and Marketing Return on Advertising Spend (MROAS) metrics.

1. Budget Optimization:

The interface includes an "Optimize Budget" button that, when clicked, suggests an optimized budget based on the MMM model's calculations. This helps users allocate their budgets effectively across different marketing channels.

**Dockerization:**

* **Dockerfile**

1. Define the Base Image

Start by specifying the base image as "python:3.9-slim-buster" in the Dockerfile.

**FROM python:3.9-slim-buster**

1. Install Dependencies

Install the necessary dependencies using the following commands:

**RUN apt-get update**

1. Upgrade pip

Upgrade pip to the latest version using the following command:

**RUN python -m pip install --upgrade pip**

1. Set Working Directory

Set the working directory to "/app" using the following command:

**WORKDIR /app**

1. Copy Requirements File

Copy the "requirements.txt" file from the host to the working directory in the container using the following command:

**COPY requirements.txt .**

1. Install Python Packages

Install Python packages specified in the "requirements.txt" file using the following command:

**RUN pip install -r requirements.txt**

1. Expose port 8502 for external access in Dockerfile.

**EXPOSE 8502**

1. Copy Application Files

Copy all the files from the host to the working directory in the container using the following command:

**COPY . .**

1. Define the Default Command

Set the default command to run the application using Streamlit by executing the following command:

**CMD ["streamlit", "run","app.py","--server.port=8502"]**

* **Docker Compose**

1. Create a Docker Compose File

Create a new file named "docker-compose.yml" and open it for editing.

1. Step 2: Specify Version

In the first line of the file, specify the version of Docker Compose being used. For example, use version '3'.

**version: '3'**

1. Define Services

Under the 'services' section, specify the services you want to build and run. In this case, we will define a service named 'app'.

**services:**

**App:**

1. Specify Build Configuration

Under the 'app' service, define the build configuration by specifying the build context. Use a dot ('.') to represent the current directory where the Dockerfile is located.

**build: .**

1. Specify Port Mapping

Specify the port mapping to expose the application's port. In this case, we wil map port 8502 of the host to port 8502 of the container.

**ports:**

**- "8502:8502"**

**Deployment**

To deploy the Camera Detection PoC on AWS, a ***t2 large*** instance with ***12GiB*** storage was created. This documentation outlines the steps required to set up a local instance of the "MMM PoC" project using Docker and Docker Compose.

1. Update Package Repository

Before installing any new packages, it's always a good idea to update the package repository to ensure that you have the latest versions of all packages.

Run the following command to update the package repository:

***sudo apt-get update***

1. Install Python 3 pip Package

Python 3 pip is a package manager for Python. It is used to install and manage packages and dependencies for Python projects. Run the following command to install the Python 3 pip package:

***sudo apt-get install python3-pip***

1. Install Docker Compose

Docker Compose is a tool that allows you to define and run multi-container Docker applications. Run the following command to install Docker Compose:

***sudo apt install docker-compose***

1. Install Docker.io

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Run the following command to install Docker:

***sudo apt install docker.io***

1. Clone the Project Repository

The "MMM\_poc" project code is hosted on GitHub. Clone the project repository using the following command:

***git clone*** [***https://github.com/ayushpatel-srijan/MMM\_poc.git***](https://github.com/ayushpatel-srijan/MMM_poc.git)

1. Change Directory to the Project Directory

Change the current working directory to the "MMM\_poc" project directory using the following command:

***cd MMM\_poc***

1. Start the Docker Compose Build Process

Run the following command to start the Docker Compose build process:

***sudo docker-compose up --build***

This will download all the necessary dependencies and build the project. Once the build process is complete, you can access the project by navigating to exposed ports in your web browser.