

Introduction-

• Customer Personality Analysis is a detailed analysis of a company's ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors, and concerns of different types of customers.



Content-

Brief overview of Customer

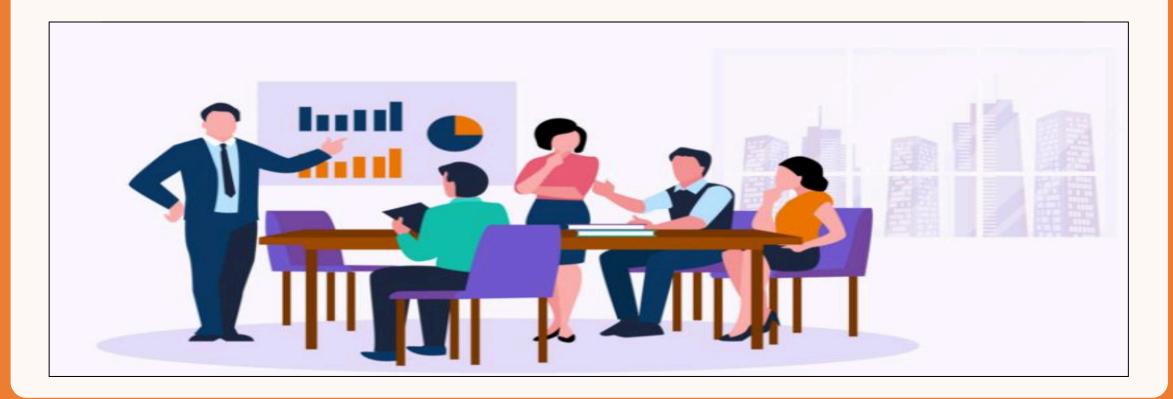
Personality Analysis.

- •Importance of understanding ideal customers.
- Mention the problem statement.

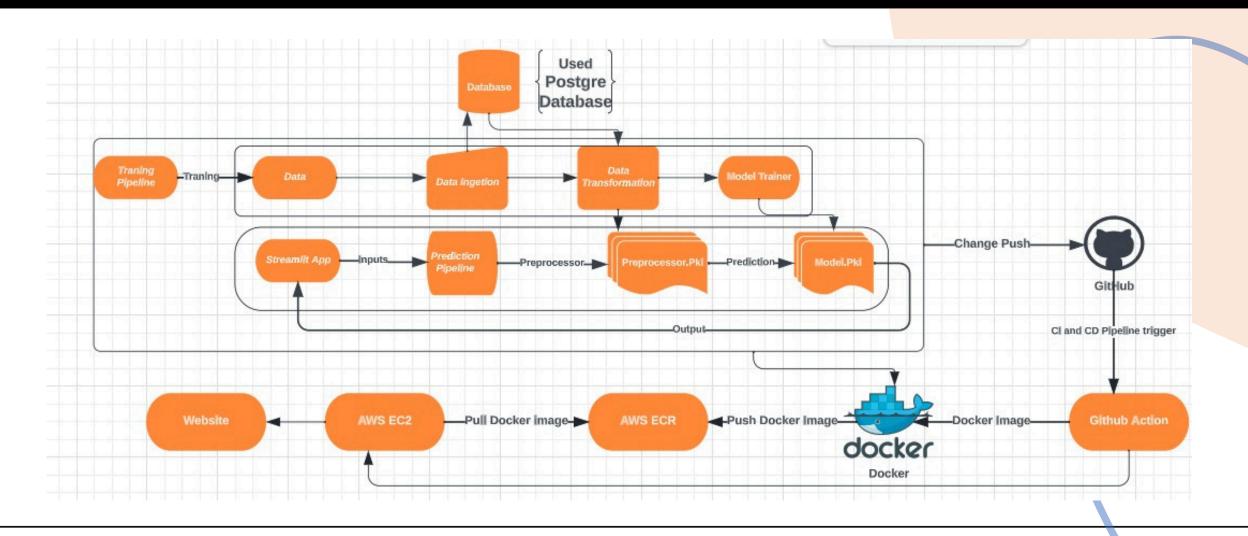


Agenda --

Customer Sentiment Analysis- Understand customers' attitudes towards the product. **Behavior Analysis**- Analyze what customers do rather than what they say about the product.



<u>Architecture</u> -



<u>Data</u> Ingestion-

DataFrame to Database: Load data into a DataFrame and use SQLAlchemy to seamlessly transfer it into a database.

Custom Ingestion Function: Create a tailored function using SQLAlchemy to handle data ingestion, schema definition, and table creation.

Scalability and Efficiency: Optimize data ingestion for scalability and performance with techniques like bulk inserts and connection pooling.

<u>Data</u>

Transformation-

The data transformation phase is a critical step in preparing the dataset for analysis and modeling. It involves several key operations to ensure data quality, enhance features, and make it suitable for machine learning. The transformations applied include data type conversion, feature engineering, data cleaning, and encoding.

Date Conversion-

•The 'dt_customer' column, which represents the date when a customer became a client, was converted from a string format ('%d-%m-%Y') to a datetime format for date-based calculations.



Feature Engineering-

The 'Family_Size' column was

The 'customer_for' column was created to calculate the number of months a customer has been with the company, based on the 'dt_customer' column.

assigned a size of 1 and 'Partner' households assigned a size of 2.

The 'Is_Parent' column was computed to identify whether a customer is a parent based on the presence of children.

The 'Age' column was computed by subtracting the birth year from the current year.

The 'children' column was calculated by summing the 'kidhome' and 'teenhome' columns to determine the total number of children in a household.

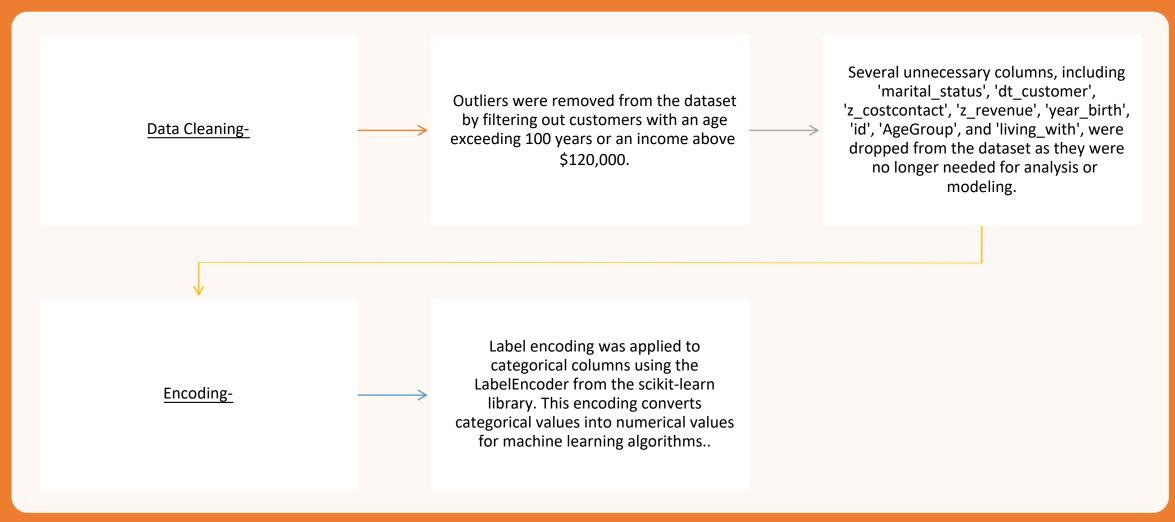
The 'education' column was standardized to categorize education levels as 'Undergraduate', 'Graduate', or 'Postgraduate' for consistency.

The 'Spent' column was created by summing the spending across various product categories.

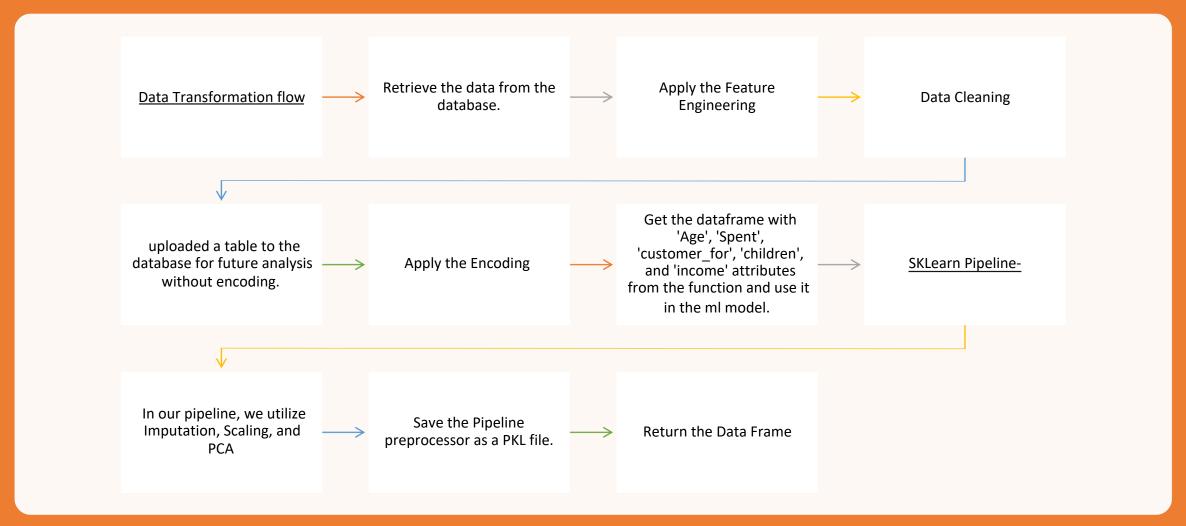
The 'living_with' column was created to simplify marital status categories, mapping 'Married' and 'Together' to 'Partner' and other categories to 'Single'.

Age groups were defined in the 'AgeGroup' column, categorizing customers as 'Teen', 'Adult', 'Middle Age Adult', or 'Senior Adult' based on their age.

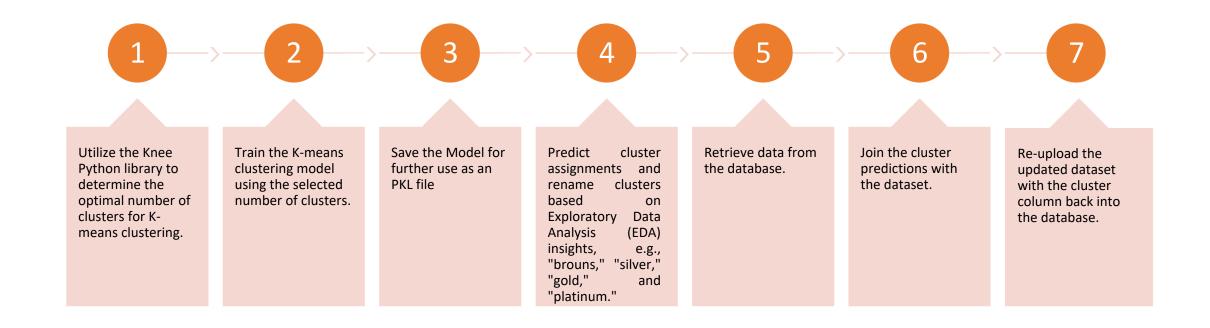
DataCleaning and Encoding



Data Transformation Flow

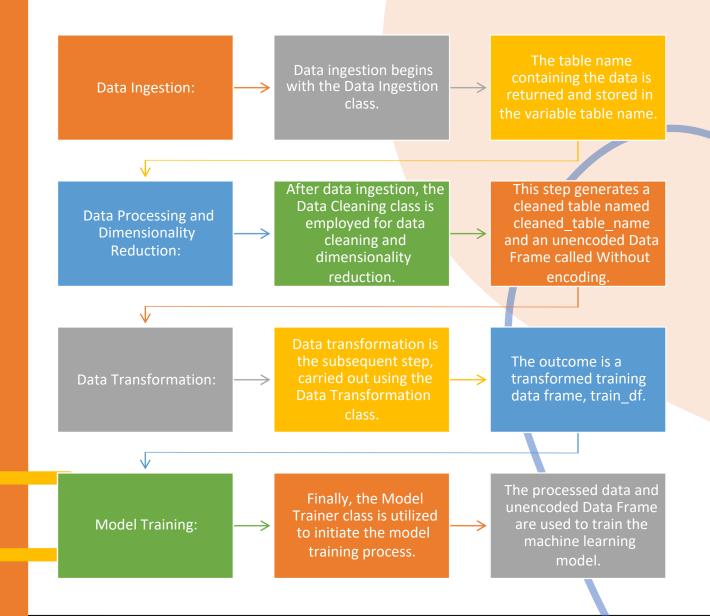


Model Training-



20XX presentation title 11

Training Pipeline-



Prediction Pipeline-

Prediction Function:

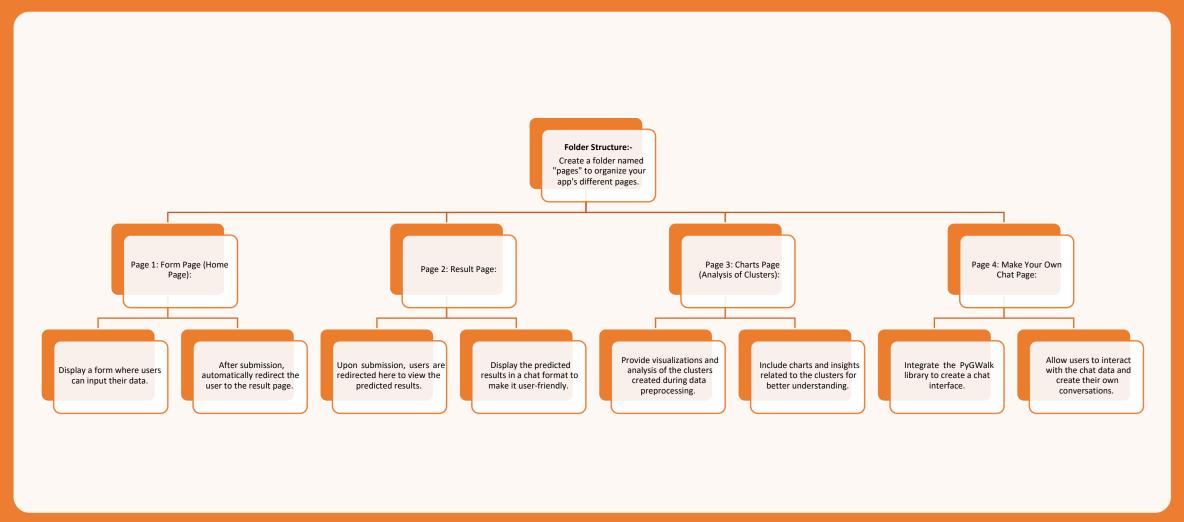
The Prediction Function is responsible for making predictions using a trained machine learning model.

It loads both the preprocessor (.pkl) file and the trained model (.pkl) file.

Input data is processed through the preprocessor to prepare it for prediction.

The model is applied to generate predictions.

Stream Lit App-



Integration of

Prediction

Pipeline with

- In the Form Page, connect the input data to the prediction pipeline.
- Upon submission, pass the input data through the prediction pipeline to generate predictions.
- Display the predicted results on the Result Page.



Streamlit App Workflow

The user visits the Home Page (Form Page) and fills in the required data.

Upon hitting the submit button, the app triggers the prediction process.

The user is redirected to the Result Page, where the predicted results are displayed in a chat format.

The user can navigate to the Charts Page to see cluster analysis and insights.

The Make Your Own Chat Page allows users to interact with existing chat data and create their own conversations.

CI/CD Pipeline-

GitHub Code
Repository Update:
Push your code
changes to your GitHub
repository, ensuring
that your project is upto-date.

GitHub Action
Automation: Set up
GitHub Actions to
automate your CI/CD
pipeline. Configure
triggers to initiate
workflows upon code
commits and other
events.

Docker Image
Generation: Utilize a
Dockerfile to create a
Docker image
encapsulating your
application or service,
ensuring
reproducibility and
portability.

AWS ECR Image
Repository: Push your
Docker image to
Amazon Elastic
Container Registry
(ECR), a fully managed
container registry
service, ensuring
secure storage and
distribution.

AWS EC2 Self-hosted
GitHub Runner:
Activate a self-hosted
GitHub runner on an
Amazon EC2 instance
to ensure reliable and
scalable CI/CD
execution. Configure it
to periodically check
for updates.

Pull Latest Docker Image: Enable the EC2 runner to fetch the latest Docker image from ECR, ensuring that the most current version of your application is deployed.

Execute Docker
Container: Run the
new Docker image on
your EC2 instance,
ensuring your
application is up and
running with the latest
changes.

App Deployment-

In the deployment phase, we have established our website's hosting infrastructure using an Amazon Elastic Compute Cloud (EC2) instance. This EC2 instance functions as the platform where our website is hosted, enabling accessibility to users via the internet. This configuration provides us with the capability to deploy and operate our web application in a highly scalable and dependable manner, leveraging the robust AWS infrastructure.

thank you

Katta Sai Pranav

kattapranavreddy@gmail.com

