Name – **Shivam Ganesh Dhumal**

Email – [shivdhumal2003@gmail.com](mailto:shivdhumal2003@gmail.com)

Phone – 7822053051(India)

Linkdn - www.linkedin.com/in/shivam-dhumal

Google drive link of all files = <https://drive.google.com/drive/folders/1UH_szqIl6Z6uqWX1T1lExmAL_3OVmaZi?usp=drive_link>

THIS IS THE **OVERVIEW AND MUST READ** PDF FOR ACCESSING FOLDERS , FOR VISUALISATION AND UNDERSTANDING , LOOK FOR THE **PRESENTAITON**

Link=

**Marketing (First model) – Description and links**

In this Model , we performed RFM (Recency, Frequency, Monetary) analysis on customer transaction data for the year 2010. Here's a breakdown of what we did:

**Data Loading and Preprocessing**: We loaded customer transaction data from an Excel file and converted the 'InvoiceDate' column to datetime format. Then, we extracted the year from the invoice dates and focused only on data for the year 2010.

**RFM Calculation**: We calculated three key metrics for each customer:

**Recency**: How recently a customer made a purchase (measured in days since their last purchase in 2010).

**Frequency**: The number of transactions a customer made in 2010.

**Monetary**: Total spending by each customer in 2010.

**K-Means Clustering**: We applied K-Means clustering to segment customers based on their RFM metrics. We chose 4 clusters.

**Customer Type Assignment**: We defined customer types based on the clusters obtained from K-Means clustering. These types are:

**High Value Customers**: Customers who are recent, frequent, and spend a lot.

**Low Value Customers**: Customers who are less recent, less frequent, and spend less.

**Medium Value Customers**: Customers falling between high and low value customers in terms of RFM metrics.

**New Customers**: Customers who are recently acquired and may need special attention to retain.

**Visualization**: We visualized the distribution of customer types using a bar graph showing the count of each customer type based on their recency. Additionally, we created a scatter plot to visualize RFM segmentation, where each point represents a customer, with Recency on the x-axis, Monetary on the y-axis, and different customer types represented by colors.

**Code links**

2010\_RFM model code = <https://drive.google.com/file/d/1-IXwsx5Eg7QBSB-vEaNQnOyoe6wV2mxe/view?usp=drive_link>

2011\_RFM model code = <https://drive.google.com/file/d/1mLhOOIbgC5Z7wM58piE5u4WMyfMzoX4F/view?usp=drive_link>

Images RFM\_2010 image =

<https://drive.google.com/file/d/1HyNA1-Y2aFlZKhBFQlpicOyhLrm7ud7a/view?usp=drive_link>

images RFM\_2011 image = <https://drive.google.com/file/d/1fdBCn9YLchDqKGXeyuxgHZKsxadxlijy/view?usp=sharing>

**(second model) – only using the Recency data model (description)**

Data is loaded using the Pandas library from an Excel file containing customer transaction records.

Specifically targeting the year 2010, the code filters the dataset to include transactions only from that year.

Recency, indicating how recently a customer made a purchase in 2010, is computed for each customer. This is achieved by finding the difference between the maximum date of transactions for each customer and the latest transaction date in the dataset

.

Customer types are determined based on their recency values. These types serve as indicators of customer engagement and potential churn risk.

The function **get\_customer\_type** assigns each customer to one of five categories:

**"Active (loyal) Customers"** for those who made a purchase within the last 30 days,

**"Less Active Customers"** for those with a purchase between 31 to 90 days ago,

**"Potential Churn Customers"** for customers with a purchase between 91 to 180 days ago,

**"Churn Risk Customers"** for those with a purchase between 181 to 365 days ago, and

**"Inactive Customers"** for customers who haven't made a purchase in over a year.

Using the Matplotlib library, a bar graph is generated to visualize the distribution of customer types. Each bar represents the count of customers falling into a particular category.

Additional features like labeling each bar with its count, rotating x-axis labels for readability, adding grid lines for better visualization, and adjusting layout spacing for clarity are implemented to enhance the graph's effectiveness in conveying information.

**Link (whole folder)=** <https://drive.google.com/drive/folders/1qIVIx8nU3zdTUkb4HOcGKdIaCdmfu8jD?usp=drive_link>

code link (2010)= <https://drive.google.com/file/d/1BG6CIycX7X3rh2AEPQxrVNJqlxxNHO9r/view?usp=drive_link>

code link (2011)=

<https://drive.google.com/file/d/1D5rg6M7QNbnPI9-2TLp2kmHtS7sZGCIc/view?usp=drive_link>

(Images are in the same folder )

**THE GRAPHS AND MARKETING STRATEGIES ARE IN PRESENTATION ITSELF**

**Retail products team:**

**First data model:**

**Link=** <https://drive.google.com/drive/folders/1VTGEpUOFfIMrZ9GfxKTQT52QDKS8TEfb?usp=drive_link>

The dataset link ( extracted) = <https://docs.google.com/spreadsheets/d/1R2TXnsmEVGBLVsiRWly5VbX-AJ4unl71/edit?usp=drive_link&ouid=114090571837239378236&rtpof=true&sd=true>

The script performs a market basket analysis on sales data to identify the next best product to recommend for the top 10 selling products based on the Average Basket Index (ABI). Here's a brief explanation of each step:

**Calculate ABI Index**:

Group sales data by invoices to determine which items were bought together.

Create pairs of items bought together and count their occurrences.

Divide the count of each pair by the total number of transactions to get the ABI index for each pair.

**Identify Top 10 Selling Stock Codes**:

Sum the quantities sold for each stock code.

Sort the stock codes by total quantity sold and select the top 10.

**Find Next Best Product for Each Top-Selling Product**:

For each top-selling product, filter the ABI index to find pairs involving the product.

Select the product with the highest ABI index as the next best product, excluding pairs of the product with itself.

Finally, the results are displayed in a table showing each top-selling product and its next best product recommendation.

**Second data model using ABI index and matplotlib**

This code analyzes customer transaction data for the year 2010 to calculate the Average Basket Item (ABI) index for each product pair. The ABI index indicates the likelihood of two products being purchased together in the same transaction. Here's a breakdown:

**Data Loading**: Transaction data is loaded from an Excel file into a Pandas DataFrame.

**Date Formatting**: The 'InvoiceDate' column is converted into datetime format for ease of manipulation.

**Year Filtering**: Data is filtered to include transactions only from the year 2010.

**ABI Index Calculation**: A function **calculate\_abi\_index** is defined to compute the ABI index for each product pair. It iterates through unique stock codes, calculates the co-occurrence of each pair in transactions, and computes the ABI index based on the proportion of occurrences of one product alongside another.

**ABI Index Visualization**: The ABI index matrix for 2010 is visualized using a heatmap. Each cell in the heatmap represents the ABI index between two products, with higher values indicating a stronger association between them. Seaborn's heatmap function is utilized for visualization, annotating each cell with the ABI index value.

**Plot Customization**: The heatmap is customized with titles, labels, and rotation of x and

**Customer relationship management: (description)**

Link = <https://drive.google.com/drive/folders/11-WxdO8qGWDLkuudhkTM646FFJ-yWsgD?usp=drive_link>

**(Description of the model )**

**Loading the Dataset**: The dataset containing customer transactions is loaded from an Excel file.

**Converting InvoiceDate to Datetime Format**: The **InvoiceDate** column is converted to a datetime format for easier manipulation and calculation.

Calculate Recency: Recency, defined as the number of days since the last purchase, is calculated for each customer. This involves:

Finding the most recent purchase date for each customer.

Calculating the number of days between the current date and the most recent purchase date.

**Calculating the Monetary Value**: The total spending for each customer is calculated by multiplying the quantity of items purchased by their price.

**Merge Recency and Monetary Data**: The Recency and Monetary values are merged into a single DataFrame.

**Set Thresholds and Filter Churn Candidates**: Customers are filtered based on specified thresholds for Recency (90 days) and Monetary value (1000 currency units).

**Add Customer Information**: Customer names and countries are added to the filtered churn candidates list

**Display and Export Churn Candidate Information**: The churn candidate information is displayed and then exported to a new Excel file.

Full folder Link = <https://drive.google.com/drive/folders/11-WxdO8qGWDLkuudhkTM646FFJ-yWsgD?usp=drive_link>

**( consists of graphs , data models and Extracted data set of year 2010 – 2011)**