INSTACART SALES FORECASTING AND RECOMMENDATION SYSTEM

PROJECT BY: ADITI REDDY DOMA

INTRODUCTION

Instacart is a leading platform in the online grocery space. It handles millions of orders and user interactions daily, offering a lot of data. I was curious, so I explored a dataset from Kaggle and built a project using **Python** to uncover **data insights** on user behavior, product trends, and order frequency. The project includes interactive **visualizations** and a **machine learning powered product recommendation system** focused on reorder probability, collaborative filtering, customer loyalty and time sensitive suggestions for inactive users.

DATASETS

1. **Orders.csv**: About customer orders
2. **Order_products_prior.csv**: Details about the products by users whether the product was reordered and if yes, when.
3. **products.csv** : Product-level information like product ID, product name, aisle ID, and department ID.
4. **Departments.csv**: Department names like dairy, bakery, beverages.

Link to datasets:
https://www.kaggle.com/datasets/yasserh/instacart-online-grocery-basket-analysis-dataset

DATA PREPROCESSING

1. **Handling Missing Data**:
    - Checked for missing values using **.isna().sum()**.
    - I have found orders dataset had missing values in the days_since_prior_order column.
    - Missing values were **replaced with 0**, because missing values in this column typically indicate a new customer's first order. Treating them as 0 ensures consistency without affecting the analysis, as both NaN and 0 imply no prior order history.

2. **Merging Datasets**:

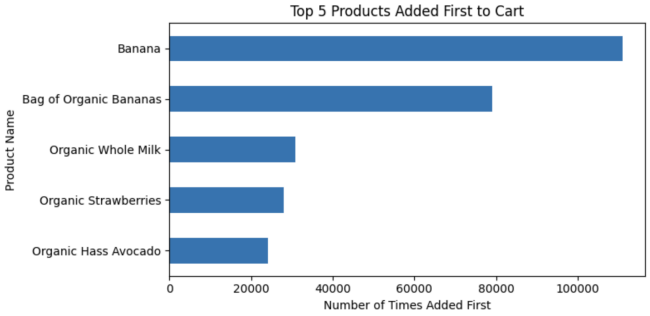
- `order_prior` and `products` datasets were merged using the `product_id`. This join helped in analysis and visualizations.

Exploratory Data Analysis (Python) and Visualizations (Matplot, Seaborn)

1. Top 5 Products Added First to Cart

- **Insight:** Products customers tend to pick first. This indicates high-priority items
- **Code Summary:** Using Python I filtered order_prior where add_to_cart_order == 1. Merged with products to get product names. Counted frequency of products. Visualized bar plot using Matplot.

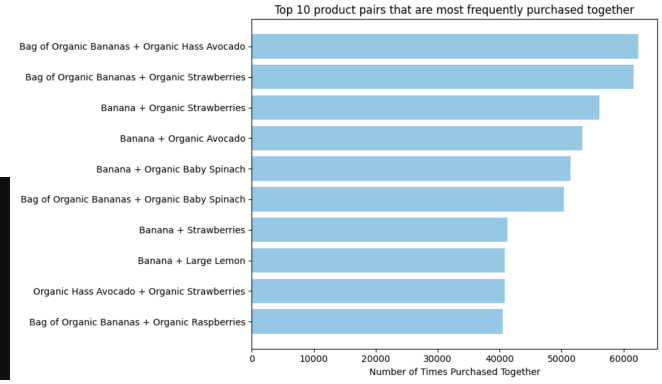
```
Top 5 First Products Added First to Cart:
product_name
Banana                   110916
Bag of Organic Bananas    78988
Organic Whole Milk        30927
Organic Strawberries      27975
Organic Hass Avocado      24116
```



2. Top 10 Product Pairs Frequently Purchased Together

- **Insight:** Product pairs that customers buy frequently. Used for collaborative filtering in recommendation system.
- **Code Summary:** Created product pairs from each order in order_prior. Counted how often each unique pair appears together. Visualized bar plot using Matplot.

```
Top 10 Product Pairs Frequently Purchased Together:

Bag of Organic Bananas + Organic Hass Avocado → 62341 times
Bag of Organic Bananas + Organic Strawberries → 61628 times
Banana + Organic Strawberries → 56156 times
Banana + Organic Avocado → 53395 times
Banana + Organic Baby Spinach → 51395 times
Bag of Organic Bananas + Organic Baby Spinach → 50372 times
Banana + Strawberries → 41232 times
Banana + Large Lemon → 40880 times
Organic Hass Avocado + Organic Strawberries → 40794 times
Bag of Organic Bananas + Organic Raspberries → 40503 times
```

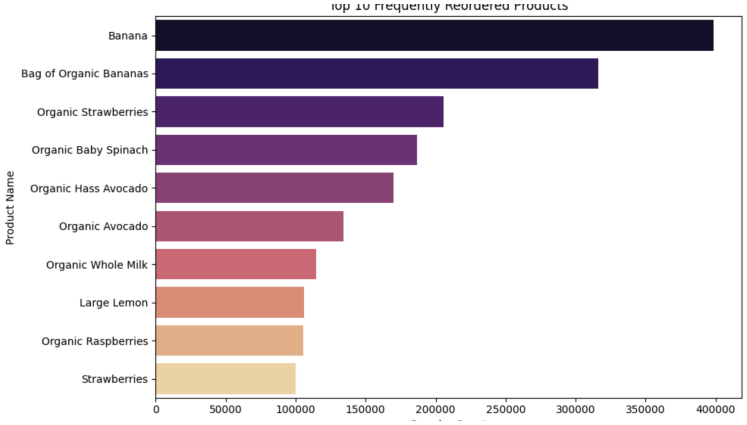


3. Top 10 Frequently Reordered Products

- **Insight:** Products with the highest reorder frequency, showing strong customer preferences. Used for high-retention items for customer loyalty.

- **Code Summary:** Filtered rows where `reordered == 1`, indicating the product was purchased again. Counted how often each product was reordered. Visualized a bar plot using seaborn.

	product_id	reorder_count	product_name
0	24852	398609	Banana
1	13176	315913	Bag of Organic Bananas
2	21137	205845	Organic Strawberries
3	21903	186884	Organic Baby Spinach
4	47209	170131	Organic Hass Avocado
5	47766	134044	Organic Avocado
6	27845	114510	Organic Whole Milk
7	47626	106255	Large Lemon
8	27966	105409	Organic Raspberries
9	16797	99802	Strawberries



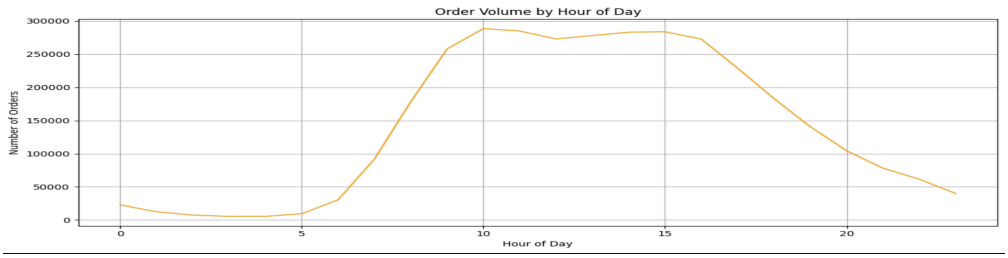
4. Orders Distribution by Day of the Week

- **Insight:** Identified peak shopping days.
- **Code Summary:** Grouped orders by order_dow (day of week). Counted orders per day. Visualized with a bar chart.

```
Number of Orders by Day of Week:
Sunday (0) : 600905 orders
Monday (1) : 587478 orders
Tuesday (2) : 467260 orders
Wednesday (3) : 436972 orders
Thursday (4) : 426339 orders
Friday (5) : 453368 orders
Saturday (6) : 448761 orders
```

5. Order Volume by Hour of Day

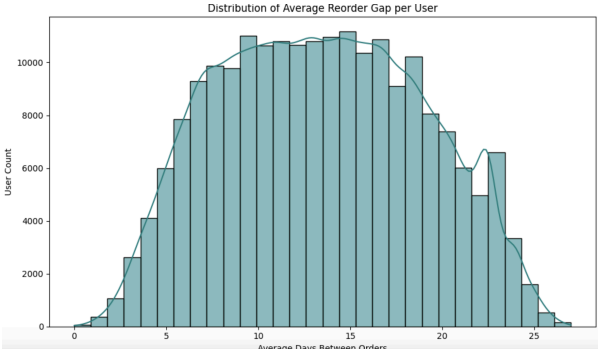
- **Insight:** User activity patterns throughout the day.Tells us about peak hours and low hours.
- **Code Summary**: A orange line plot with Seaborn tells how order volume changes over the 24 hours.



6. Average Day Gap Between Orders

- **Insight:** Helps to understand how frequently customers reorder.A left-skewed distribution with a peak near low day values indicates many frequent buyers. A long tail reflects users who take longer to reorder, representing occasional or inactive customers.
- **Code Summary**: The dataset is grouped by user_id, and the mean of days_since_prior_order is calculated for each user. A histogram is plotted using Seaborn's histplot, with variability of 30 and a Kernel Density Estimation (KDE) curve for smooth distribution analysis.

```
Average reorder gap across users: 13.43
```

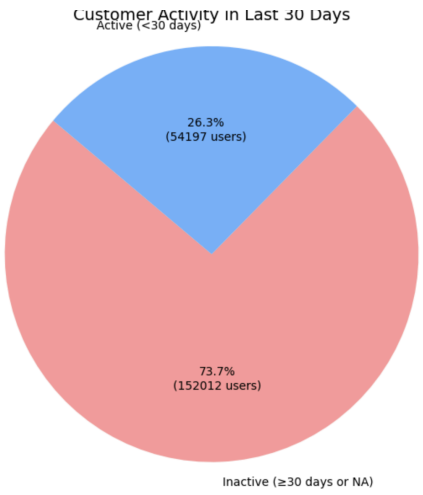


7. Identifying Inactive Customers

- **Code Summary:** calculated the maximum value of days_since_prior_order. If this value is ≥ 30 days or missing, the customer is labeled inactive. Visualised using Pie Chart Inactive: No order in the last 30 days or data missing and Active: Ordered within 30 days.
- **Insight**:This analysis helps in identifying inactive customers so that we can send them messages saying this product is popular at that time.

```
Number of inactive customers in the last 30 days: 152012
```

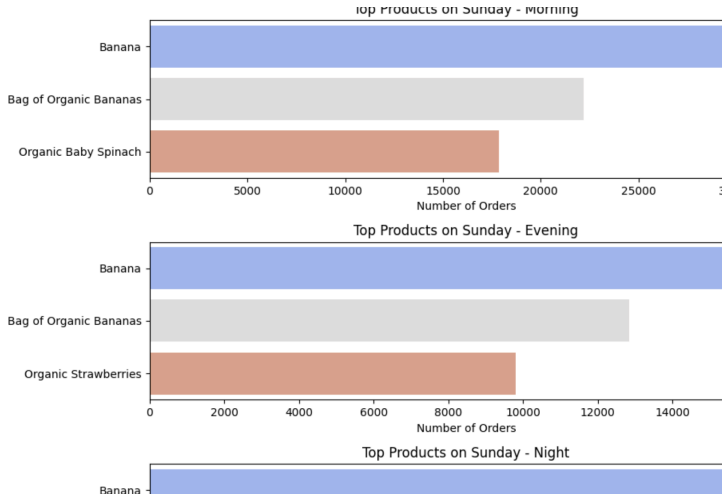
	user_id	max_days_since_prior
0	1	30.0
1	2	30.0
3	4	30.0
6	7	30.0
7	8	30.0



8. Top 3 Products by Day and Time Slot

- **Insight:**Tells what users buy most frequently on that day and at that time the user used to give messages for inactive customers.
- **Code Summary:** Used time_slot(hour) to divide 24 hours into Morning (6–12 AM) , Afternoon (12–6 PM), Evening (6–11 PM), Night (11 PM–5 AM). Applied this to each order using .apply() to create a time_slot column. Grouped data by order_dow, time_slot, and product_name, then counted the occurrences. Sorted and selected the top 3 products in each (day, slot) pair using .groupby().head(3). Used Seaborn barplots for visualization.

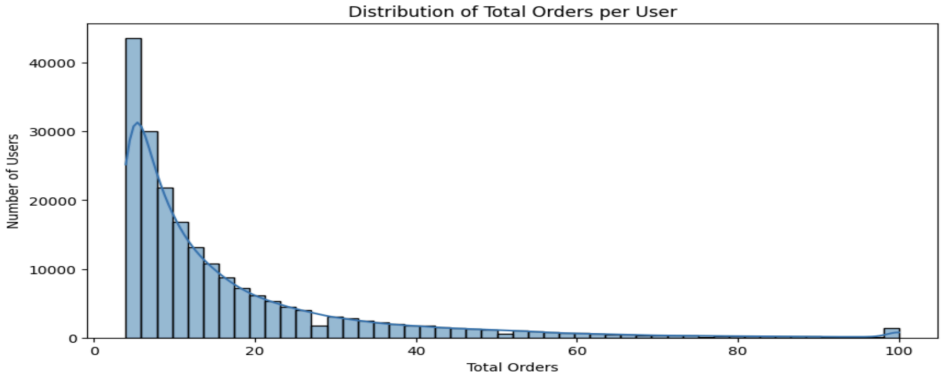
```
Sunday

Morning
    Banana (ordered 30824 times)
    Bag of Organic Bananas (ordered 22185 times)
    Organic Baby Spinach (ordered 17849 times)

Afternoon
    Banana (ordered 48389 times)
    Bag of Organic Bananas (ordered 35321 times)
    Organic Baby Spinach (ordered 27339 times)

Evening
    Banana (ordered 16140 times)
    Bag of Organic Bananas (ordered 12838 times)
    Organic Strawberries (ordered 9800 times)

Night
    Banana (ordered 1416 times)
    Bag of Organic Bananas (ordered 1149 times)
    Organic Baby Spinach (ordered 870 times)

Monday

Morning
    Banana (ordered 35353 times)
    Bag of Organic Bananas (ordered 27497 times)
    Organic Strawberries (ordered 17289 times)

Afternoon
    Banana (ordered 37890 times)
    Bag of Organic Bananas (ordered 30564 times)
    Organic Strawberries (ordered 20896 times)

Evening
    Banana (ordered 13227 times)
    Bag of Organic Bananas (ordered 10423 times)
    Organic Strawberries (ordered 7384 times)
```



9. Total Orders per User

- **Insight:** Tells how often users place orders. Peaks tell typical order frequency; long tails indicate power users or one-time users.
- **Code Summary:** Grouped the orders dataset by user_id and counted the number of orders each user has placed. Plotted a histogram (with KDE curve) showing how many users fall into each

order count range.



10. Spending-Based Customer Segmentation

- **Insight:** Categorizes users by purchasing frequency. Helped in identifying loyal, regular, and occasional customers.
- **Code Summary:** Calculated 75th and 90th percentiles of total orders.Classified users into: High Spender (≥ 90th percentile),Medium Spender (≥ 75th percentile), Low Spender (below 75th percentile).

```
spending_category
Low Spender       152278
Medium Spender     33287
High Spender       20644
Name: count, dtype: int64
```

RECOMMENDATION SYSTEM

FEATURES INCLUDED:

- order_dow – day of the week
- order_hour_of_day – hour of the order
- days_since_prior_order – time gap since last order
- add_to_cart_order – cart position of product in that order
- avg_cart_position – average cart position over all orders

TRAIN-TEST SPLIT AND SCALING:

- The data was split into **training (80%)** and **testing (20%)** sets.
- Features were scaled using **StandardScaler to normalize inputs** before model training.
- A reference table of (user_id, product_id) pairs was retained for mapping predictions back to users.

MODEL BUILDING:

Recommendations include:Model-based reorder predictions, KNN collaborative filtering, Spending-tier-based promotions, Time-sensitive inactive user prompts.

1. **Model-Based Recommendations:Reorder probabilities predicted by an XGBoost classifier**, trained on features like cart position, order timing, and reorder history. If a product has a high probability of being reordered, the model recommends. **Evaluation Metric: 'logloss'.**
    - If **reorder_prob > 0.98 and days_since_prior_order ≥ 5**, the item is recommended.

2. **Collaborative Filtering:** Using the **k-Nearest Neighbors algorithm, the system identifies users with similar purchasing behavior**.
    - A **sparse matrix** was created with users and products using binary interactions.
    - **Model Used: k-Nearest Neighbors** with **cosine similarity**
    - **Logic:** For each user, the top 3 most similar users were identified, and the most popular products among them were recommended.

3. **Customer Segmentation-Based Promotions:** Customers are segmented based on their shopping frequency and given message for user engagement:
    - **High-frequency users** receive **Loyalty Bonus messages** .
    - **Medium-frequency users** are offered **Special Deals**.
    - **Low-frequency users** get **Essential Discounts**.

4. **Inactive User Targeting:** Based on exploratory data analysis, the system identifies inactive users (those with no orders in 30+ days). It s**uggests the top 3 popular products for the current day and time slot** , accompanied by message like *"*Order now — it's popular at this time!"

OUTPUT:

Output includes: user_id, product_name, message.

```
Final Recommendations (Top 25 + Inactive Users):
           user_id                                     product_name  \
0            83407                              Organic Strawberries
1           148940                            Organic Fat Free Milk
2           105748                           Bag of Organic Bananas
3           147761                             Organic Baby Spinach
4           205943                                 Honeycrisp Apple
5            82085                                           Banana
6           199305          Total 2% All Natural Plain Greek Yogurt
7           108199                                          Banana
8           110243                                          Banana
9            74014  YoKids Squeeze! Organic Strawberry Flavor Yogurt
10          198727                                     Half & Half
11          181111                           Bag of Organic Bananas
12          112557                             Organic Baby Spinach
13           78314                                          Banana
14          185652                                  Hass Avocados
15           53004                                          Banana
16           43058                          Giant Roll Paper Towels
17          183613                  Organic Unsweetened Almond Milk
18           78705          Super Chunk Extra Crunchy Peanut Butter
19          128558               Brown Fertile Large Grade AA Eggs
20  InactiveUser_1                                          Banana
21  InactiveUser_2                           Bag of Organic Bananas
22  InactiveUser_3                             Organic Strawberries

                                    message
0      Try reordering essentials at a discount!
1   Loyalty Bonus: Save more on your next order!
2     Special Deal: Limited-time bundle for you!
3   Loyalty Bonus: Save more on your next order!
4      Try reordering essentials at a discount!
5   Loyalty Bonus: Save more on your next order!
6   Loyalty Bonus: Save more on your next order!
7   Loyalty Bonus: Save more on your next order!
8      Try reordering essentials at a discount!
9     Special Deal: Limited-time bundle for you!
10     Try reordering essentials at a discount!
11     Try reordering essentials at a discount!
12    Special Deal: Limited-time bundle for you!
13     Try reordering essentials at a discount!
14     Try reordering essentials at a discount!
15    Special Deal: Limited-time bundle for you!
16     Try reordering essentials at a discount!
17  Loyalty Bonus: Save more on your next order!
18  Loyalty Bonus: Save more on your next order!
19     Try reordering essentials at a discount!
```

```
20        Order now – it's popular at this time!
21        Order now – it's popular at this time!
22        Order now – it's popular at this time!

Inactive User Recommendations:
          user_id            product_name  \
93  InactiveUser_1                  Banana
94  InactiveUser_2  Bag of Organic Bananas
95  InactiveUser_3    Organic Strawberries

                                   message
93  Order now – it's popular at this time!
94  Order now – it's popular at this time!
95  Order now – it's popular at this time!
```

EVALUATION:

- To evaluate the performance of the recommendation model, standard classification metrics were calculated using scikit-learn.

```
[40]: # Evaluation
      from sklearn.metrics import precision_score, recall_score, f1_score
      print("\nEvaluation:")
      print(f"Precision: {precision_score(recommendation_final['actual'], recommendation_final['predicted'], pos_label=1):.4f}")
      print(f"Recall: {recall_score(recommendation_final['actual'], recommendation_final['predicted'], pos_label=1):.4f}")
      print(f"F1 Score: {f1_score(recommendation_final['actual'], recommendation_final['predicted'], pos_label=1):.4f}")


      Evaluation:
      Precision: 0.9783
      Recall: 1.0000
      F1 Score: 0.9890
```