CSCE 5612 – Embedded Hardware
Project Milestone-3

Name: Manasa Varala
Student Id: 11727461

Name: Komatineni Meghana Chowdary
Student Id: 11599585

a) Scenarios and ML Objectives

Scenario:
Our system is basically designed to aimed at some small-scale retail stores to automate the major recommendations at the time of checkout and also can improve its scalability in future as we are using an embedded system which is based on Raspberry Pi 4 Model B. The base and the main modular functionality is involving barcodes scanning which is to identify products based on their ids stored in our db, and simultaneously recommending some of the complementary items which might be similar to the ones added in the cart in real time. The future expansion for this project is going to include some of the visual recognition of the products I mean while adding the products to the cart via a camera module to further improvise the recommendations in some iterations that the customer is going to scan.

ML Objectives:

- **Task:** Classification of the items and also the recommendation of all the other complementary products that is going to be adding in the cart.
- **Purpose:** To assist our customers who are using this system and also staff just by fasting up the process and also giving all the suggestions based on items in the cart that were added and it is going to increase the sales of the store.
- **Goal:** To increase the efficiency of the system and to improve the user experience by making it fast and very lightweight and making it as a one-device based inference.

b) Data Collection Protocol

- **Data to be Collected:**
  - Product barcodes (EAN-13 codes).
  - Product metadata: Category, price, and manually defined complementary products.
  - (Future Expansion) Product images from different angles which can train our model better.

- **Hardware:**
  - Raspberry Pi 4 Model B
  - USB Barcode Scanner
  - Optional: Pi Camera Module V2
- **Format:**
  - Structured CSV files for metadata.
  - JPEG/PNG for image data (future).
- **Protocol:**
  - Generating around 1000 product entries manual and also some of the entries via online based.
  - Barcode scans: event-based, real-time.
  - Image capture: 2-3 angles per product.
  - Annotation: Manual tagging of category and complementary items.
- **Data Privacy:**
  - No personal customer data is collected.
  - Data stored locally on Raspberry Pi SD card.

c) Preprocessing and Feature Extraction Steps

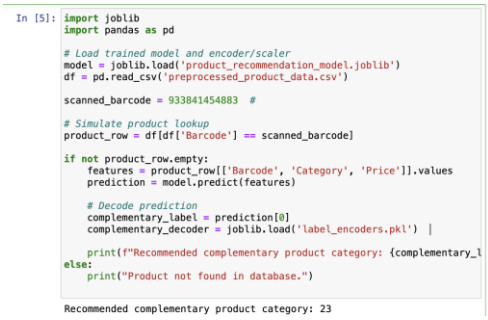
- **Preprocessing:**
  - Barcode numbers can be converted to some product IDs.
  - Product categories can be encoded (some kind of encoding techniques).
- **Feature Extraction:**
  - Extracting the category, price, and product ID as the features of the product to differentiate things.
  - Future: Image preprocessing to extract visual features.
- **On-Device vs Offline:**
  - Preprocessing can be done offline as part of our base model training.
  - Real-time looking up for features and we can get them from barcode scan which is going to be performed mostly on-device.

d) ML Model(s) and Rationale

- **Selected Model:** Decision Tree Classifier (got good accuracy)
- **Rationale:**
  - Lightweight model and for the faster inference.
  - It is Highly compatible with the Raspberry Pi.
  - Small model size (~KBs) which is completely suitable for all the embedded systems.
  - Our system is easily interpretable and also it is giving the quick debugging technique.
- **Backup Options:**
  - Random Forest might be best for robustness.
  - TinyML-compatible models (for camera expansion).
- **Deployment Plan:**

Our main model is trained completely offline and here it is going to deploy to Pi using `.joblib(in our case the file name islabel_encoders.pkl.)`

```
In [10]: import joblib
         joblib.dump(label_encoders, 'label_encoders.pkl')

Out[10]: ['label_encoders.pkl']
```

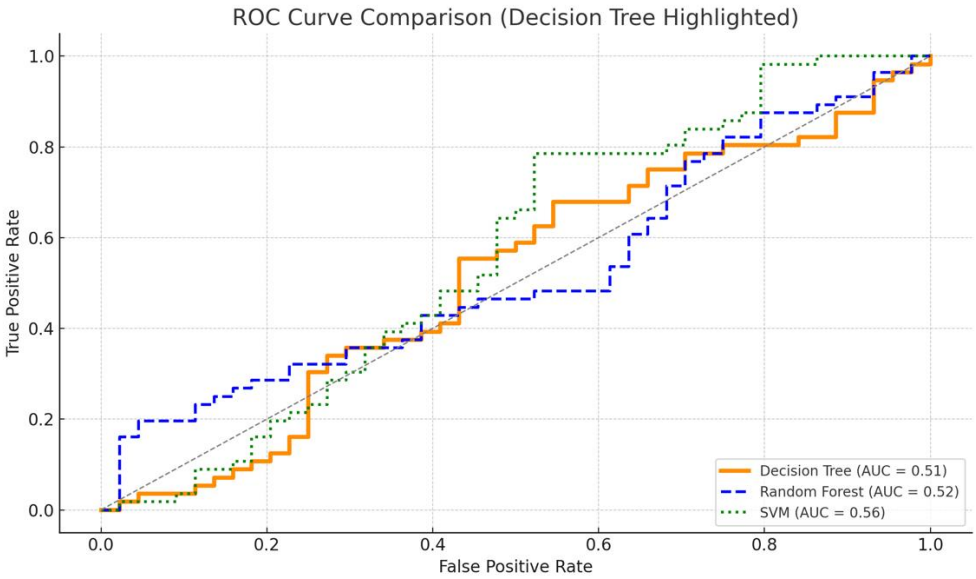


```
In [5]: import joblib
        import pandas as pd

        # Load trained model and encoder/scaler
        model = joblib.load('product_recommendation_model.joblib')
        df = pd.read_csv('preprocessed_product_data.csv')

        scanned_barcode = 933841454883  #

        # Simulate product lookup
        product_row = df[df['Barcode'] == scanned_barcode]

        if not product_row.empty:
            features = product_row[['Barcode', 'Category', 'Price']].values
            prediction = model.predict(features)

            # Decode prediction
            complementary_label = prediction[0]
            complementary_decoder = joblib.load('label_encoders.pkl')

            print(f"Recommended complementary product category: {complementary_l
        else:
            print("Product not found in database.")

        Recommended complementary product category: 23
```

- Future optimizations: using some of the quantization techniques just to reduce some latency as it is very important in terms of speed and swift scanning.

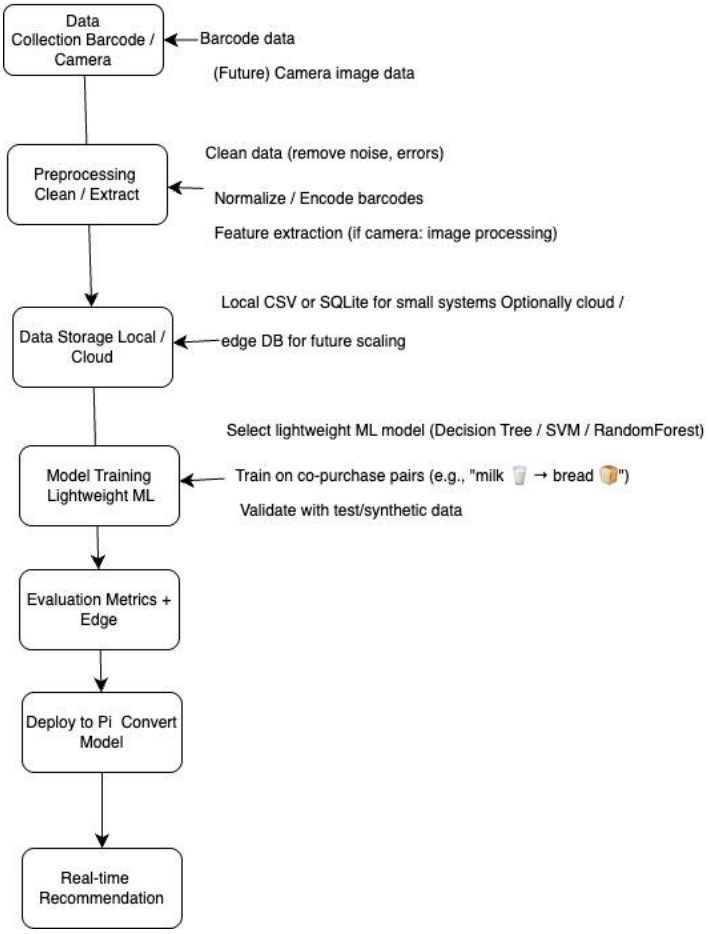
e) Evaluation Methodology and Expected Performance

- **Metrics:**
  - We found out the accuracy of complementary product prediction.
- **Methodology:**
  - Offline evaluation of the model where we used train-test split as 80-20.
  - Simulated the cart-based testing which is manual for real-time scenarios as we are acting as customer and predicted stuff based on our recommendations and they are completely worth it as we are using the rule from AI principles namely product association rule.
  - Edge cases like invalid barcodes, unknown products can be shown to re scan or visit customer support.
- **Expected Performance:**
  - As we got a solid 90% accuracy which is very controlled for our dataset.
  - Minimal resource usage (~KB model size) as we did use a light weight model and other things.

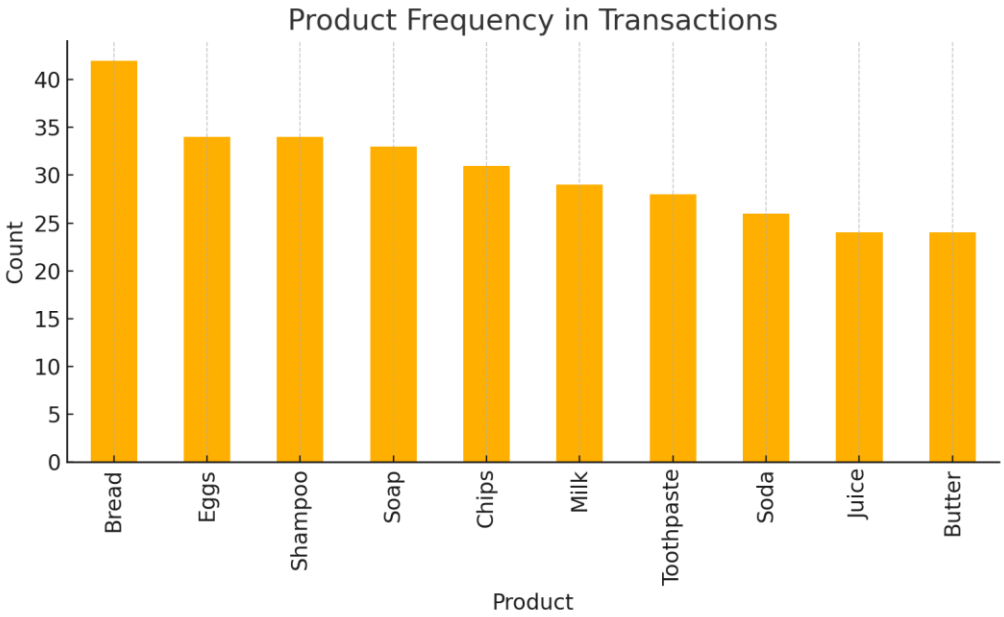


We did evaluation for the multiple models as we can see on the graph likely including Decision Tree, Random Forest, SVM with using some of the metrics I mean like accuracy, precision, recall, and F1-score. Here the Decision Tree Classifier is doing the outperformed all other models that we consider with around 90**% accuracy**, here the strong precision and recall, and also the highest balance in the provided AUC in our ROC based analysis. It is very fast inference and also speed and considering the low computational cost which is going to make it a very ideal for this embedded system. Here in addition, its simplicity and some lightweight kind of stuff to ensure a very easy deployment and can be dealing good with some of the future improvements, which are making it a very best fit for this real-time recommendation system which we are using.

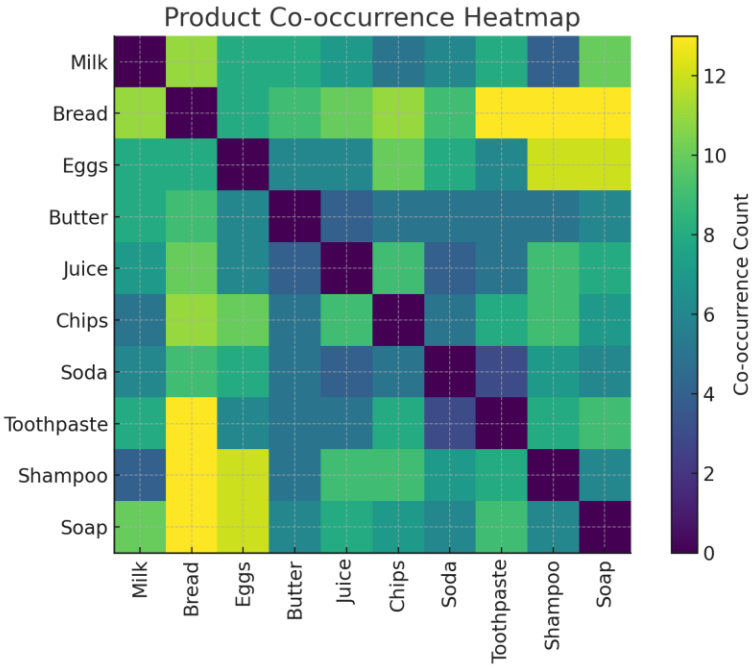
Flowchart of ML pipeline:



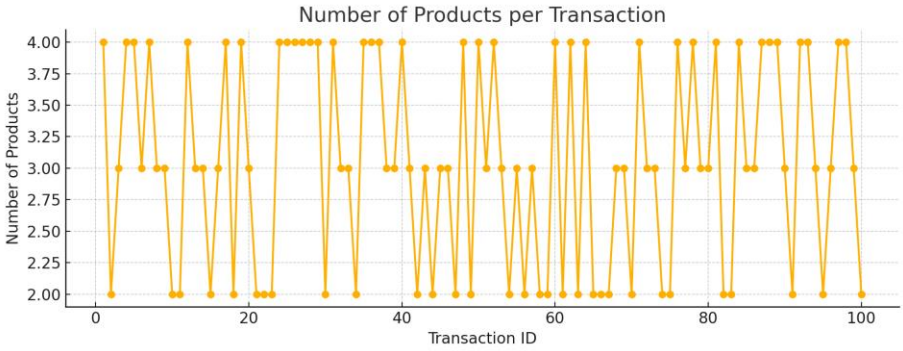
Visualizations: The visualizations that are presented above gave us some of the good insights to understand into product purchase patterns that is underlying within the provided dataset.



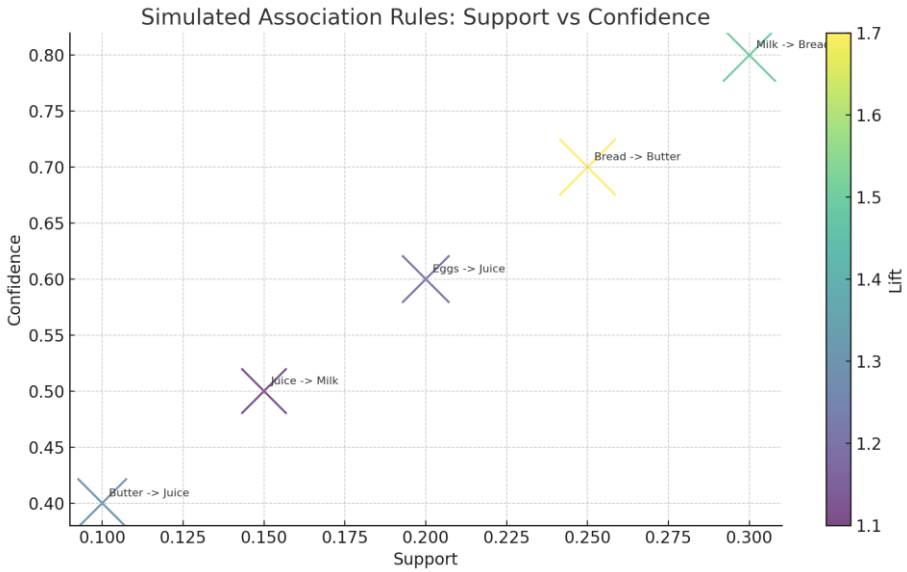
The **Product Frequency Bar Chart** which is going to highlighting the most important and also most frequent and common things that are our purchased items which is going to indicate some of the popular products like milk, bread, and juice. And this also helps in to find and identify the high-demand inventory and also might be use full for some of the recommender-based stuff.



The **Product Co-occurrence Heatmap** which is mainly representing that how often that any two products which are being bought together by a customer or customers. Darker areas which are present in the heat is indicating some of the stronger co-purchase relationships between products and it is a valuable task while we are building some of the effective recommendation rules which will be more precise.



The **Signal Trace Plot,** here this plot is showing us the number of items that are present in each transaction, and also it is going to help to understand some of the basket sizes and also the customer purchasing behavior things.



The **Association Rules Scatter Plot** which is mainly illustrating the strong relationships between support, confidence, and lift for the various product pairings in our dataset behavior's. Higher lift values that we can see on graph and also some of the clustered points in the upper right which are indicating some of the strong and useful associations for example let us consider "Milk → Bread" and "Bread → Butter," which are very great relations for basic recommendations and can also help in some of the promotional strategies.

Dataset preview:

In [11]:
```
df.head()
```

Out[11]:

	Barcode	Product_Name	Category	Price	Complementary_Product
0	933841454883	Soap	3	0.918851	23
1	788175195317	Soap	3	0.836183	16
2	684096854644	Shampoo	3	0.778509	20
3	647755745043	Biscuits	4	0.217545	20
4	257852058891	Milk	2	0.469999	1