**SMARTSPACE : TRANSFORMING RETAIL SHELVES WITH MARKET BASKET ANALYSIS**

**PROJECT CODE :**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# File path for the Excel file

file\_path = '/content/Book1.xlsx'

try:

    # Step 1: Load data from the Excel file

    df = pd.read\_excel(file\_path)

    # Step 2: Calculate Total Transactions and Total Price per Product

    product\_analysis = df.groupby('Product Name').agg(

        Total\_Transactions=('Product Name', 'size'),  # Count of transactions

        Total\_Price=('Price', 'sum')  # Sum of price for each product

    ).reset\_index()

    # Display the product analysis

    print("Product Analysis (Total Transactions and Total Price):")

    print(product\_analysis)

    # Step 3: Calculate Support for Each Product

    total\_transactions = df.shape[0]

    product\_analysis['Support'] = product\_analysis['Total\_Transactions'] / total\_transactions

    # Define suitability rules for recommendations

    suitability\_rules = {

        'Milk': ['Tea', 'Bread'],

        'Beer': ['Cola'],

        'Cola': ['Beer'],

        'Diaper': ['Milk', 'Bread', 'Beer', 'Cola'],

        'Bread': ['Milk', 'Diaper'],

        'Eggs': ['Milk', 'Bread', 'Cola', 'Beer']

    }

    # Function to calculate metrics

    def calculate\_metrics(df, itemA, itemB):

        support\_A = df[df['Product Name'] == itemA]['Price'].count() / total\_transactions

        support\_B = df[df['Product Name'] == itemB]['Price'].count() / total\_transactions

        support\_AB = df[(df['Product Name'] == itemA) | (df['Product Name'] == itemB)]['Price'].count() / total\_transactions

        confidence = support\_AB / support\_A if support\_A != 0 else 0

        lift = support\_AB / (support\_A \* support\_B) if support\_A \* support\_B != 0 else 0

        return support\_AB, confidence, lift

    # Prepare a list to hold recommendations with metrics

    recommendations = []

    # Calculate support, confidence, and lift for each product's recommendations

    for product, suitable\_items in suitability\_rules.items():

        product\_recommendations = []

        for recommendation in suitable\_items:

            if recommendation in product\_analysis['Product Name'].values:

                support\_AB, confidence, lift = calculate\_metrics(df, product, recommendation)

                product\_recommendations.append([product, recommendation, support\_AB, confidence, lift])

        # Sort by lift and select top two recommendations

        product\_recommendations = sorted(product\_recommendations, key=lambda x: x[4], reverse=True)[:2]

        recommendations.extend(product\_recommendations)

    # Convert recommendations to a DataFrame for display

    recommendations\_df = pd.DataFrame(recommendations, columns=['Product', 'Recommended', 'Support', 'Confidence', 'Lift'])

    # Display Support, Confidence, and Lift for Each Product

    print("\nSupport, Confidence, and Lift values for each Product:")

    print(product\_analysis[['Product Name', 'Support']])

    # Display Top 2 Recommendations for Each Product Based on Suitability Rules

    print("\nTop 2 Recommendations for Each Product with Support, Confidence, and Lift:")

    print(recommendations\_df)

    # Plotting the line graph for Support, Confidence, and Lift

    plt.figure(figsize=(12, 6))

    plt.plot(recommendations\_df.index, recommendations\_df['Support'], label='Support', marker='o', color='b')

    plt.plot(recommendations\_df.index, recommendations\_df['Confidence'], label='Confidence', marker='o', color='g')

    plt.plot(recommendations\_df.index, recommendations\_df['Lift'], label='Lift', marker='o', color='r')

    plt.xticks(recommendations\_df.index, recommendations\_df['Product'] + " -> " + recommendations\_df['Recommended'], rotation=45)

    plt.xlabel("Product Pair")

    plt.ylabel("Value")

    plt.title("Support, Confidence, and Lift for Top Recommended Product Pairs")

    plt.legend()

    plt.tight\_layout()

    plt.show()

    # Create Pivot Table for Heatmap of Lift Values

    heatmap\_data = recommendations\_df.pivot(index="Product", columns="Recommended", values="Lift")

    # Plot Heatmap

    plt.figure(figsize=(10, 8))

    sns.heatmap(heatmap\_data, annot=True, fmt=".2f", cmap="YlGnBu", cbar\_kws={'label': 'Lift Value'})

    plt.title('Lift Values for Recommended Product Pairs (Heatmap)')

    plt.show()

except FileNotFoundError:

    print(f"Error: '{file\_path}' not found. Please ensure the file exists in the current directory or provide the correct path.")

except KeyError as e:

    print(f"Error: Column '{e}' not found in the Excel file. Please provide the correct column names.")

except Exception as e:

    print(f"An unexpected error occurred: {e}")

**WEBSITE MODEL CODE :**

**APP.PY**

from flask import Flask, request, jsonify

from mlxtend.frequent\_patterns import apriori, association\_rules

import pandas as pd

app = Flask(\_\_name\_\_)

@app.route('/run-apriori', methods=['POST'])

def run\_apriori():

data = request.json.get("transactions")

df = pd.DataFrame(data)

# Run Apriori algorithm

frequent\_itemsets = apriori(df, min\_support=0.05, use\_colnames=True)

# Generate association rules

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

# Prepare response data

recommendations = rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].to\_dict('records')

for rule in recommendations:

rule['antecedents'] = list(rule['antecedents'])

rule['consequents'] = list(rule['consequents'])

return jsonify(recommendations)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**INDEX.HTML :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SmartSpace</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<!-- Module Selection Screen -->

<section id="module-selection" class="screen" style="display: block;">

<h1 id="title">SmartSpace</h1>

<h2>Select Module</h2>

<button onclick="showScreen('user-login')">User</button>

<button onclick="showScreen('admin-login')">Admin</button>

</section>

<!-- User Login Screen -->

<section id="user-login" class="screen">

<h1 id="title">SmartSpace</h1>

<h2>User Login</h2>

<label for="customer-name">Customer Name:</label>

<input type="text" id="customer-name" placeholder="Enter your name" required>

<label for="customer-address">Customer Address:</label>

<input type="text" id="customer-address" placeholder="Enter your address" required>

<button onclick="showScreen('product-selection')">Proceed</button>

<button onclick="showScreen('module-selection')">Go Back</button>

</section>

<!-- Product Selection Screen -->

<section id="product-selection" class="screen">

<h2>Select Products</h2>

<div id="product-list">

<label>Tea: <input type="number" id="tea-qty" min="0" value="0"> Boxes</label>

<label>Coffee: <input type="number" id="coffee-qty" min="0" value="0"> Bags</label>

<label>Eggs: <input type="number" id="eggs-qty" min="0" value="0"> Dozens</label>

<label>Bread: <input type="number" id="bread-qty" min="0" value="0"> Loaves</label>

<label>Beer: <input type="number" id="beer-qty" min="0" value="0"> Bottles</label>

<label>Diaper: <input type="number" id="diaper-qty" min="0" value="0"> Packs</label>

</div>

<button onclick="goToRecommendations()">Proceed</button>

<button onclick="showScreen('user-login')">Go Back</button>

</section>

<!-- Recommendation Screen -->

<section id="recommendation-screen" class="screen">

<h2>Recommended Products</h2>

<p>Customers usually buy these products with your selected items:</p>

<div id="recommendation-list">

<!-- Recommendations will appear here -->

</div>

<button onclick="goToCheckout()">Proceed to Checkout</button>

<button onclick="showScreen('product-selection')">Go Back</button>

</section>

<!-- Checkout Screen -->

<section id="checkout" class="screen">

<h2>Checkout</h2>

<div id="bill-summary">

<!-- Bill details will appear here -->

</div>

<button onclick="savePurchase()">Finish</button>

</section>

<!-- Admin Login Screen -->

<section id="admin-login" class="screen">

<h1 id="title">SmartSpace</h1>

<h2>Admin Login</h2>

<label for="admin-username">Username:</label>

<input type="text" id="admin-username" placeholder="Enter username" required>

<label for="admin-password">Password:</label>

<input type="password" id="admin-password" placeholder="Enter password" required>

<button onclick="adminLogin()">Login</button>

<button onclick="showScreen('module-selection')">Go Back</button>

</section>

<!-- Admin Screen -->

<section id="admin-screen" class="screen">

<h2>Customer Purchases</h2>

<div id="customer-data">

<!-- Customer data will be populated here -->

</div>

<button onclick="showScreen('module-selection')">Log Out</button>

</section>

<script src="script.js"></script>

</body>

</html>

**STYLES.CSS :**

/\* Link to custom Google Font \*/

@import url('https://fonts.googleapis.com/css2?family=Pacifico&display=swap');

/\* General Styling \*/

body {

font-family: Arial, sans-serif;

background: linear-gradient(to bottom right, #FF6347, #9370DB);

color: #ffffff;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

h1#title {

font-size: 3em;

color: #ffffff;

font-family: 'Pacifico', cursive;

text-align: center;

letter-spacing: 2px;

margin-bottom: 20px; /\* Adjusted margin \*/

}

/\* Screen Containers \*/

.screen {

display: none;

width: 350px;

max-width: 90%;

background-color: rgba(255, 255, 255, 0.1);

border-radius: 15px;

padding: 20px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

text-align: center;

}

h2 {

font-size: 1.5em;

margin-bottom: 15px;

}

/\* Form Inputs \*/

label {

display: block;

font-size: 1em;

margin-bottom: 5px;

text-align: left;

}

input[type="text"],

input[type="password"],

input[type="number"] {

width: 100%;

padding: 8px;

margin: 10px 0 15px 0;

border: none;

border-radius: 8px;

background-color: #f8f8f8;

color: #333;

font-size: 1em;

}

input[type="number"] {

width: 60px;

}

/\* Buttons \*/

button {

width: 100%;

padding: 12px;

margin: 10px 0;

font-size: 1em;

font-weight: bold;

color: #ffffff;

background-color: #ff4500;

border: none;

border-radius: 8px;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #ff6347;

}

/\* Admin and User Buttons \*/

#module-selection button {

width: 48%;

margin: 5px 1%;

background-color: #4CAF50;

}

#module-selection button:hover {

background-color: #66BB6A;

}

/\* Custom Styling for Product List \*/

#product-list label {

font-size: 1em;

display: flex;

justify-content: space-between;

padding: 8px 0;

}

input[type="checkbox"] {

margin-left: 10px;

}

/\* Checkout Summary Styling \*/

#bill-summary h3 {

font-size: 1.2em;

color: #ffd700;

margin-top: 15px;

}

/\* Admin Customer Data \*/

#customer-data {

max-height: 500px;

overflow-y: scroll;

background-color: rgba(255, 255, 255, 0.2);

padding: 10px;

border-radius: 10px;

margin-top: 15px;

font-size: 0.9em;

}

#customer-data div {

margin-bottom: 10px;

}

**SCRIPT.JS :**

// Sample prices for products

const prices = {

tea: 3.0,

coffee: 2.5,

eggs: 1.2,

bread: 2.5,

beer: 1.5,

diaper: 2.0,

cola: 1.0

};

// Sample recommendation rules

const recommendations = {

tea: [

{ name: 'Bread', price: 2.5 },

{ name: 'Coffee', price: 2.5 }

],

coffee: [

{ name: 'Tea', price: 3.0 },

{ name: 'Bread', price: 2.5 }

],

eggs: [

{ name: 'Bread', price: 2.5 },

{ name: 'Diaper', price: 2.0 }

],

bread: [

{ name: 'Eggs', price: 1.2 },

{ name: 'Milk', price: 1.5 }

],

beer: [

{ name: 'Diaper', price: 2.0 },

{ name: 'Cola', price: 1.0 }

],

diaper: [

{ name: 'Beer', price: 1.5 },

{ name: 'Bread', price: 2.5 }

]

};

// Store data

let userCart = {

customerName: "",

customerAddress: "",

products: {},

total: 0

};

let customerHistory = [];

// Show screen function

function showScreen(screenId) {

const screens = document.querySelectorAll('.screen');

screens.forEach(screen => screen.style.display = 'none');

document.getElementById(screenId).style.display = 'block';

}

// Collect products and go to recommendations

function goToRecommendations() {

const teaQty = parseInt(document.getElementById('tea-qty').value) || 0;

const coffeeQty = parseInt(document.getElementById('coffee-qty').value) || 0;

const eggsQty = parseInt(document.getElementById('eggs-qty').value) || 0;

const breadQty = parseInt(document.getElementById('bread-qty').value) || 0;

const beerQty = parseInt(document.getElementById('beer-qty').value) || 0;

const diaperQty = parseInt(document.getElementById('diaper-qty').value) || 0;

userCart = {

customerName: document.getElementById('customer-name').value,

customerAddress: document.getElementById('customer-address').value,

products: {

tea: teaQty,

coffee: coffeeQty,

eggs: eggsQty,

bread: breadQty,

beer: beerQty,

diaper: diaperQty

},

total: 0

};

// Calculate total price

userCart.total = (teaQty \* prices.tea) + (coffeeQty \* prices.coffee) +

(eggsQty \* prices.eggs) + (breadQty \* prices.bread) +

(beerQty \* prices.beer) + (diaperQty \* prices.diaper);

// Show recommendations

showRecommendations();

}

// Display recommendations based on products in the cart

function showRecommendations() {

const recommendationList = document.getElementById('recommendation-list');

recommendationList.innerHTML = ''; // Clear the list

// Find products selected

const selectedProducts = Object.keys(userCart.products).filter(product => userCart.products[product] > 0);

selectedProducts.forEach(product => {

const recommendedItems = recommendations[product] || [];

recommendedItems.forEach(item => {

const itemElement = document.createElement('div');

itemElement.classList.add('recommended-item');

itemElement.innerHTML = `<label><input type="checkbox" class="recommend-checkbox" data-name="${item.name}" data-price="${item.price}"> ${item.name} - $${item.price.toFixed(2)}</label>`;

recommendationList.appendChild(itemElement);

});

});

showScreen('recommendation-screen');

}

// Proceed to checkout with selected recommendations

function goToCheckout() {

const checkboxes = document.querySelectorAll('.recommend-checkbox');

checkboxes.forEach(checkbox => {

if (checkbox.checked) {

const productName = checkbox.getAttribute('data-name').toLowerCase();

const productPrice = parseFloat(checkbox.getAttribute('data-price'));

if (!userCart.products[productName]) {

userCart.products[productName] = 1; // Add 1 unit of the selected recommendation

} else {

userCart.products[productName] += 1; // Increment quantity if product already exists

}

// Recalculate total

userCart.total += productPrice;

}

});

updateCheckout();

}

// Update checkout page with the selected products and total

function updateCheckout() {

const billSummary = document.getElementById('bill-summary');

billSummary.innerHTML = `<h3>Customer: ${userCart.customerName}</h3>

<h3>Address: ${userCart.customerAddress}</h3>

<h3>Products:</h3>

<ul>

${Object.keys(userCart.products).map(product => {

return userCart.products[product] > 0 ?

`<li>${product.charAt(0).toUpperCase() + product.slice(1)}: ${userCart.products[product]}</li>` : '';

}).join('')}

</ul>

<h3>Total: $${userCart.total.toFixed(2)}</h3>`;

showScreen('checkout');

}

// Save purchase and add to admin history

function savePurchase() {

customerHistory.push(userCart);

alert('Purchase completed!');

showScreen('module-selection');

}

// Admin login

function adminLogin() {

const username = document.getElementById('admin-username').value;

const password = document.getElementById('admin-password').value;

// Simple admin authentication

if (username === "sai" && password === "sairecaids42") {

showAdminScreen();

} else {

alert('Invalid credentials!');

}

}

// Show admin screen with customer purchase data

function showAdminScreen() {

const customerDataDiv = document.getElementById('customer-data');

customerDataDiv.innerHTML = '';

if (customerHistory.length === 0) {

customerDataDiv.innerHTML = '<p>No purchases yet.</p>';

} else {

const table = document.createElement('table');

table.innerHTML = `

<tr>

<th>Customer Name</th>

<th>Address</th>

<th>Time</th>

<th>Products</th>

<th>Total</th>

</tr>

`;

customerHistory.forEach(purchase => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${purchase.customerName}</td>

<td>${purchase.customerAddress}</td>

<td>${new Date().toLocaleString()}</td>

<td>${Object.keys(purchase.products).filter(product => purchase.products[product] > 0).map(product => {

return `${product.charAt(0).toUpperCase() + product.slice(1)}: ${purchase.products[product]}`;

}).join(', ')}</td>

<td>$${purchase.total.toFixed(2)}</td>

`;

table.appendChild(row);

});

customerDataDiv.appendChild(table);

showScreen('admin-screen') }}