4th food reccomendation

September 19, 2025

[1]:
```
# Food Recommendation Project - v3
# Includes: Item-CF with lower thresholds, time/customer features, RandomForest␣
 ↪& XGBoost tuning

import pandas as pd
import numpy as np
from faker import Faker
import random
from collections import defaultdict

import warnings
warnings.filterwarnings('ignore')

fake = Faker()
num_orders = 100000

categories = {
    'Pizza': ['Margherita', 'Pepperoni', 'Hawaiian', 'BBQ Chicken', 'Veggie'],
    'Burger': ['Cheeseburger', 'Chicken Burger', 'Veggie Burger', 'Bacon␣
 ↪Burger'],
    'Pasta': ['Spaghetti Bolognese', 'Penne Arrabiata', 'Fettuccine Alfredo',␣
 ↪'Lasagna'],
    'Fish and chips': ['Classic Fish & Chips', 'Spicy Fish & Chips', 'Vegan␣
 ↪Fish & Chips'],
    'Kebabs': ['Chicken Kebab', 'Lamb Kebab', 'Beef Kebab', 'Veg Kebab'],
    'Wrap': ['Chicken Wrap', 'Veggie Wrap', 'Falafel Wrap'],
    'Calzone': ['Ham & Cheese Calzone', 'Veggie Calzone', 'Pepperoni Calzone']
}

drinks = ['Coke', 'Pepsi', 'Fanta', 'Water', 'Orange Juice', 'Lemonade']
sides = ['Fries', 'Onion Rings', 'Salad', 'Garlic Bread', 'Mashed Potatoes']
starters = ['Soup', 'Spring Rolls', 'Chicken Wings', 'Bruschetta', 'Garlic␣
 ↪Mushrooms']
desserts = ['Ice Cream', 'Brownie', 'Cake', 'Fruit Salad', 'Pudding']

prices = {
    'Pizza': (8, 20),
```

```
    'Burger': (5, 12),
    'Pasta': (7, 15),
    'Fish and chips': (6, 14),
    'Kebabs': (6, 14),
    'Wrap': (5, 12),
    'Calzone': (8, 16),
    'Drink': (1, 4),
    'Side': (2, 6),
    'Starter': (3, 8),
    'Dessert': (3, 7)
}
```

[2]:
```
# Generate orders
order_data = []
for i in range(1, num_orders+1):
    customer_id = fake.uuid4()
    order_date = fake.date_time_this_year()
    main_category = random.choice(list(categories.keys()))
    main_item = random.choice(categories[main_category])
    quantity = random.randint(1, 5)

    total_price = random.uniform(*prices[main_category])
    chosen_drink = None
    chosen_side = None
    chosen_starter = None
    chosen_dessert = None

    if random.random() < 0.6:
        chosen_drink = random.choice(drinks)
        total_price += random.uniform(*prices['Drink'])
    if random.random() < 0.5:
        chosen_side = random.choice(sides)
        total_price += random.uniform(*prices['Side'])
    if random.random() < 0.4:
        chosen_starter = random.choice(starters)
        total_price += random.uniform(*prices['Starter'])
    if random.random() < 0.3:
        chosen_dessert = random.choice(desserts)
        total_price += random.uniform(*prices['Dessert'])

    total_price *= quantity
    total_price = round(total_price, 2)
    order_data.append([
        i, customer_id, order_date, main_category, main_item,
        chosen_drink, chosen_side, chosen_starter, chosen_dessert,
        quantity, total_price
    ])
```

```
df = pd.DataFrame(order_data, columns=[
    'OrderID', 'CustomerID', 'OrderDate', 'MainCategory', 'MainItem',
    'Drink', 'Side', 'Starter', 'Dessert', 'Quantity', 'TotalPrice'
])
df['ComplementaryItems'] = df[['Drink', 'Side', 'Starter', 'Dessert']].
 ↪apply(lambda row: [x for x in row if pd.notna(x)], axis=1)
```

[3]:
```
# Add time features
s = pd.to_datetime(df['OrderDate'])
df['OrderTimestamp'] = s
df['Hour'] = s.dt.hour
df['DayOfWeek'] = s.dt.dayofweek
df['IsWeekend'] = (df['DayOfWeek'] >= 5).astype(int)

# Customer-level features
cust_stats = df.groupby('CustomerID').agg(PastOrders=('OrderID','count'),␣
 ↪AvgSpend=('TotalPrice','mean')).reset_index()
df = df.merge(cust_stats, on='CustomerID', how='left')
```

[4]:
```
df
```

[4]:

	OrderID	CustomerID	OrderDate	\
0	1	550d86a6-724f-4251-a19d-4004f8e1c7fa	2025-01-10 11:55:34	
1	2	92e16d36-3c97-431e-8881-aa9683f8c0ab	2025-08-11 05:02:22	
2	3	f8617db0-974b-4035-ab9e-56fd9d83eff8	2025-08-25 18:30:35	
3	4	8b096810-11ad-46b9-84fd-ee648aa55872	2025-09-05 13:46:42	
4	5	1e671c17-1c93-4e6f-a0bb-24bb348e7a49	2025-04-27 21:42:10	
…	…	…	…	
99995	99996	2f7a5a10-f51c-4072-82a5-3b529bda322e	2025-04-01 20:24:14	
99996	99997	6a7489a3-6d2d-463e-bab2-2b07978fa4d6	2025-07-15 11:50:10	
99997	99998	dcb6d0cb-44e3-43bf-9057-c3879a96e826	2025-01-09 08:18:18	
99998	99999	6efa9c19-25fb-4d86-b325-3b21aa161063	2025-02-13 20:33:46	
99999	100000	11749506-81dc-48b2-b973-07aae4ed5bdb	2025-01-16 09:47:36	

	MainCategory	MainItem	Drink	Side	\
0	Kebabs	Beef Kebab	Coke	None	
1	Calzone	Ham & Cheese Calzone	None	Garlic Bread	
2	Burger	Chicken Burger	Coke	None	
3	Fish and chips	Classic Fish & Chips	Pepsi	None	
4	Wrap	Falafel Wrap	Pepsi	Salad	
…	…	…	…	…	
99995	Calzone	Ham & Cheese Calzone	Lemonade	Onion Rings	
99996	Burger	Cheeseburger	Pepsi	Garlic Bread	
99997	Fish and chips	Spicy Fish & Chips	None	None	
99998	Pizza	Pepperoni	None	None	
99999	Pasta	Penne Arrabiata	Fanta	None	

```
             Starter      Dessert  Quantity  TotalPrice  \
0      Chicken Wings         None         1       14.79
1      Chicken Wings         None         5       82.59
2               None         None         4       48.96
3               None         None         2       30.99
4      Chicken Wings         None         3       49.85
…                …            …       …          …
99995     Bruschetta         None         2       48.49
99996           None         None         4       47.18
99997           None         None         2       23.31
99998   Spring Rolls  Fruit Salad         2       33.38
99999           Soup      Pudding         3       77.66

                       ComplementaryItems      OrderTimestamp  Hour  \
0                     [Coke, Chicken Wings] 2025-01-10 11:55:34    11
1             [Garlic Bread, Chicken Wings] 2025-08-11 05:02:22     5
2                                   [Coke] 2025-08-25 18:30:35    18
3                                 [Pepsi] 2025-09-05 13:46:42    13
4             [Pepsi, Salad, Chicken Wings] 2025-04-27 21:42:10    21
…                                      …                  …   …
99995  [Lemonade, Onion Rings, Bruschetta] 2025-04-01 20:24:14    20
99996                 [Pepsi, Garlic Bread] 2025-07-15 11:50:10    11
99997                                   [] 2025-01-09 08:18:18     8
99998           [Spring Rolls, Fruit Salad] 2025-02-13 20:33:46    20
99999                [Fanta, Soup, Pudding] 2025-01-16 09:47:36     9

       DayOfWeek  IsWeekend  PastOrders  AvgSpend
0              4          0           1     14.79
1              0          0           1     82.59
2              0          0           1     48.96
3              4          0           1     30.99
4              6          1           1     49.85
…          …        …         …       …
99995          1          0           1     48.49
99996          1          0           1     47.18
99997          3          0           1     23.31
99998          3          0           1     33.38
99999          3          0           1     77.66

[100000 rows x 18 columns]
```

[5]:
```
# Item-based Collaborative Filtering (lower thresholds, time weighted)
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
transactions = df.apply(lambda r: [r['MainItem']] + r['ComplementaryItems'],␣
 ↪axis=1).tolist()
mlb_items = MultiLabelBinarizer()
trans_mat = mlb_items.fit_transform(transactions)
items = list(mlb_items.classes_)
item_vectors = trans_mat.T
sim_matrix = cosine_similarity(item_vectors)
index_of = {item: idx for idx, item in enumerate(items)}
categories_split = {'Drink': drinks, 'Side': sides, 'Starter': starters,␣
 ↪'Dessert': desserts}

def recommend_itemcf(main_item, top_k=6, by_category=True):
    if main_item not in index_of: return []
    idx = index_of[main_item]
    scores = sim_matrix[idx].copy()
    scores[idx] = -1
    recs = []
    if by_category:
        for cat_name, cat_items in categories_split.items():
            candidates = [(item, scores[index_of[item]]) for item in cat_items␣
 ↪if item in index_of]
            candidates.sort(key=lambda x: x[1], reverse=True)
            if candidates and candidates[0][1] > 0:
                recs.append(candidates[0][0])
    if len(recs) < top_k:
        ranked = sorted([(items[i], scores[i]) for i in range(len(items))],␣
 ↪key=lambda x: x[1], reverse=True)
        for it, sc in ranked:
            if it not in recs and sc > 0: recs.append(it)
            if len(recs) >= top_k: break
    if not recs:
        from collections import Counter
        counter = Counter()
        for t in transactions:
            if main_item in t:
                for it in t:
                    if it != main_item: counter[it] += 1
        recs = [it for it,_ in counter.most_common(top_k)]
    return recs[:top_k]

print('Item-CF recommendations for Margherita:', recommend_itemcf('Margherita',␣
 ↪top_k=6))
```

```
Item-CF recommendations for Margherita: ['Orange Juice', 'Garlic Bread', 'Spring
Rolls', 'Brownie', 'Fries', 'Coke']
```

[6]:
```
# Random Forest with features
df_feat = pd.get_dummies(df[['MainItem','Hour','DayOfWeek','IsWeekend']])
X = df_feat
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.multioutput import MultiOutputClassifier
from sklearn.preprocessing import MultiLabelBinarizer
mlb = MultiLabelBinarizer()
Y = mlb.fit_transform(df['ComplementaryItems'])
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.
 ↪2,random_state=42)
rf =␣
 ↪RandomForestClassifier(n_estimators=200,max_depth=10,random_state=42,n_jobs=-1)
multi_rf = MultiOutputClassifier(rf)
multi_rf.fit(X_train,Y_train)
Y_pred = multi_rf.predict(X_test)
print('RandomForest done')

def recommend_rf(main_item, top_k=6):
    vec = pd.DataFrame([1 if c==main_item else 0 for c in X.columns]).T
    vec.columns = X.columns
    proba_list = []
    for est in multi_rf.estimators_:
        try: p = est.predict_proba(vec.values); proba_list.append(p[:,1])
        except: p = est.predict(vec.values); proba_list.append(p)
    proba = np.array(proba_list).flatten()
    items = mlb.classes_
    ranked = sorted(zip(items,proba), key=lambda x:x[1], reverse=True)
    return [it for it,_ in ranked[:top_k]]

print('RandomForest recommendations for Margherita:',␣
 ↪recommend_rf('Margherita', top_k=6))
```

```
RandomForest done
RandomForest recommendations for Margherita: ['Lemonade', 'Fries', 'Coke',
'Pepsi', 'Garlic Bread', 'Water']
```

[7]:
```
# XGBoost (optional)
try:
    import xgboost as xgb
    XGBOOST_AVAILABLE = True
except:
    XGBOOST_AVAILABLE = False

if XGBOOST_AVAILABLE:
    xgb_clf = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss',␣
 ↪n_jobs=-1, random_state=42)
```

```
    multi_xgb = MultiOutputClassifier(xgb_clf)
    multi_xgb.fit(X_train, Y_train)
    print('XGBoost done')

    def recommend_xgb(main_item, top_k=6):
        vec = pd.DataFrame([1 if c==main_item else 0 for c in X.columns]).T
        vec.columns = X.columns
        proba_list = []
        for est in multi_xgb.estimators_:
            try: p = est.predict_proba(vec.values); proba_list.append(p[:,1])
            except: p = est.predict(vec.values); proba_list.append(p)
        proba = np.array(proba_list).flatten()
        items = mlb.classes_
        ranked = sorted(zip(items,proba), key=lambda x:x[1], reverse=True)
        return [it for it,_ in ranked[:top_k]]
    print('XGBoost recommendations for Margherita:',␣
↪recommend_xgb('Margherita', top_k=6))
```

```
XGBoost done
XGBoost recommendations for Margherita: ['Water', 'Coke', 'Pepsi', 'Onion
Rings', 'Garlic Bread', 'Orange Juice']
```