# Memory code Professor's week 3

June 23, 2025

[2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import random
from sklearn.metrics.pairwise import cosine_similarity

print('\n\n#### Load Data ####')
data = pd.read_excel('FruitSales.xlsx')
print(data.shape)
print(data.isnull().sum().sort_values(ascending=False))
data.dayCode = data.astype(str)
print(data.head())

print('\n\n##### Purchase history for each user - user-item matrix ####')
print("Each row is a unique CustomerID")
print("Each column is a unique StockCode (i.e., a product)")
print("Each cell contains the total quantity of that product purchased by that␣
 ↪customer")
print("Missing values (i.e., products the customer never bought) are filled␣
 ↪with 0")
purchase = (data.groupby(['CustomerID', 'StockCode'])['Quantity'].sum().
 ↪unstack().reset_index().fillna(0).set_index('CustomerID'))
def encode_units(x):
    if x < 1:
        return 0
    else:
        return 1
print("Convert to only 0 and 1's for not purchased or purchased")
purchase = purchase.applymap(encode_units)
print(purchase.head())

print('\n\n##### User Similarities ####')
user_similarities = cosine_similarity(purchase)
user_similarities_data = pd.DataFrame(user_similarities,index=purchase.
 ↪index,columns=purchase.index)
print(user_similarities_data.head())
```

```
userid = 12347
print(f'\n\n##### Similar Users to user {userid} ####')
def fetch_similar_users(user_id,k=5):
    # separating df rows for the entered user id
    user = user_similarities_data[user_similarities_data.index == user_id]
    # a df of all other users
    other_users = user_similarities_data[user_similarities_data.index !=␣
 ↪user_id]
    # calc cosine similarity between user and each other user
    similarities = cosine_similarity(user,other_users)[0].tolist()
    # create list of indices of these users
    indices = other_users.index.tolist()
    # create key/values pairs of user index and their similarity
    index_similarity = dict(zip(indices, similarities))
    # sort by similarity
    index_similarity_sorted = sorted(index_similarity.items(),reverse=True)
    # grab k users off the top
    top_users_similarities = index_similarity_sorted[:k]
    users = [u[0] for u in top_users_similarities]
    return users
similar_users = fetch_similar_users(userid)
print(similar_users)

print(f'\n\n##### 10 recommendations for user {userid} from similar users ␣
 ↪####')
def similar_users_recommendation(userid):
    simu = fetch_similar_users(userid)
    #obtaining all the items bought by similar users
    simu_rec = []
    for j in simu:
        desc = data[data["CustomerID"]==j]['StockCode'].to_list()
        simu_rec.append(desc)
    #this gives us multi-dimensional list
    # we need to flatten it
    flat_list = []
    for sublist in simu_rec:
        for item in sublist:
            flat_list.append(item)
    final_list = list(dict.fromkeys(flat_list))
    # storing 10 random recommendations in a list
    ten_recs = random.sample(final_list, 10)
    print('Items bought by Similar users based on Cosine Similarity')
    #returning 10 random recommendations
    return ten_recs
print(similar_users_recommendation(userid))
```

```
#### Load Data ####
(8, 6)
Unnamed: 0    1
Unnamed: 1    0
Unnamed: 2    0
Unnamed: 3    0
Unnamed: 4    0
Unnamed: 5    0
dtype: int64
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_158/450313582.py in ?()
      8 print('\n\n#### Load Data ####')
      9 data = pd.read_excel('FruitSales.xlsx')
     10 print(data.shape)
     11 print(data.isnull().sum().sort_values(ascending=False))
---> 12 data.StockCode = data.StockCode.astype(str)
     13 print(data.head())
     14 
     15 print('\n\n##### Purchase history for each user - user-item matrix #### )

/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages/pandas/core
  ↪generic.py in ?(self, name)
   6200             and name not in self._accessors
   6201             and self._info_axis.
  ↪_can_hold_identifiers_and_holds_name(name)
   6202         ):
   6203             return self[name]
-> 6204         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'StockCode'
```

```
[ ]:
```