Course:ARTIFICIALINTELLIGENCE TITLE: MARKET BASKET INSIGHTS

PHASE 4 SUBMISSION

# TEAM MEMBERS:

# MOHANA PRIYA.M

# (au311521106061)

# MOUNICKA G S

# (au31152110602)

# PADMA PRIYA .M

# (au311521106069)

# YAMINI.H

( au 311521106114)

## INTRODUCTION:

## In today's data-driven world, extracting valuable insights from vast datasets is a fundamental aspect of making informed decisions in various domains, such as retail, marketing, healthcare, and more. Advanced association analysis techniques and visualization tools play a pivotal role in uncovering hidden patterns, relationships, and trends within data, ultimately leading to enhanced insights and more effective strategies.

**Data Collection**: Obtain the dataset that contains transaction data. This data typically includes information about items purchased together in different transactions.

1. **Data Cleaning**:
   * Remove any duplicates or irrelevant records.
   * Handle missing data, if any, by either removing rows with missing values or imputing them.
   * Ensure data consistency and integrity.
2. **Data Transformation**:
   * Convert the data into a suitable format, often a transaction list or a binary matrix where rows represent transactions, and columns represent items. Each cell is marked as 1 if the item is in the transaction and 0 otherwise.
3. **Support Threshold**: Determine a minimum support threshold, which is the minimum percentage of transactions an itemset (combination of items) must appear in to be considered frequent. This threshold helps in reducing the number of itemsets to be analyzed.
4. **Apriori Algorithm** (or other association rule mining algorithms):
   * Apply the Apriori algorithm or a similar algorithm to find frequent itemsets. The algorithm iteratively discovers itemsets that meet the minimum support threshold.
   * Generate association rules from frequent itemsets.
5. **Rule Pruning**: Optionally, you can prune rules based on additional criteria like confidence, lift, or interest to focus on the most meaningful associations.
6. **Loading the Processed Data**: Depending on your analysis tool, load the preprocessed data, frequent itemsets, and association rules into a data structure that can be used for insights and further analysis.
7. **Analysis and Insights**: Analyze the generated association rules to gain insights into item relationships. This can include identifying frequently co-purchased items, cross-selling opportunities, and other market basket insights.

The exact tools and libraries you use for these tasks may vary depending on your preference and the size of your dataset. Common choices include Python with libraries like pandas, scikit-learn, and mlxtend for data preprocessing and association rule mining. For larger datasets, you might want to use distributed data processing frameworks like Apache Spark.

Remember that market basket analysis is a valuable technique for understanding customer behavior and making data-driven business decisions, such as optimizing product placements, running targeted promotions, and improving inventory management.

As a first step, I load all the modules that will be used in this notebook

import pandas as pd

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

import seaborn as sns

import datetime, nltk, warnings

import matplotlib.cm as cm

import itertools

from pathlib import Path

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_samples, silhouette\_score

from sklearn import preprocessing, model\_selection, metrics, feature\_selection

from sklearn.model\_selection import GridSearchCV, learning\_curve

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn import neighbors, linear\_model, svm, tree, ensemble

from wordcloud import WordCloud, STOPWORDS

from sklearn.ensemble import AdaBoostClassifier

from sklearn.decomposition import PCA

from IPython.display import display, HTML

import plotly.graph\_objs as go

from plotly.offline import init\_notebook\_mode,iplot

init\_notebook\_mode(connected=True)

warnings.filterwarnings("ignore")

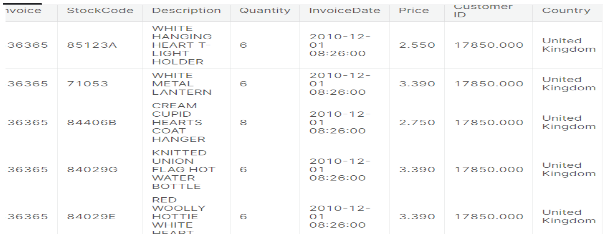
plt.rcParams["patch.force\_edgecolor"] = True

plt.style.use('fivethirtyeight')

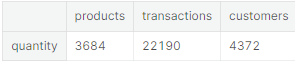
mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)

%matplotlib inline

Then, I load the data



### Customers and products



It can be seen that the data concern 4372 users and that they bought 3684 different products. The total number of transactions carried out is of the order of ∼∼22'000.

Now I will determine the number of products purchased in every transaction:

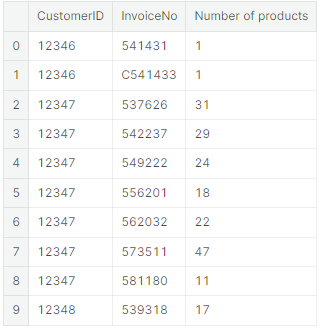
INPUT:

temp = df\_initial.groupby(by=['CustomerID', 'InvoiceNo'], as\_index=False)['InvoiceDate'].count()

nb\_products\_per\_basket = temp.rename(columns = {'InvoiceDate':'Number of products'})

nb\_products\_per\_basket[:10].sort\_values('CustomerID')

OUTPUT:



#### Data encoding

The dataframe transactions\_per\_user contains a summary of all the commands that were made. Each entry in this dataframe corresponds to a particular client. I use this information to characterize the different types of customers and only keep a subset of variables:

INPUT :

list\_cols = ['count','min','max','mean','categ\_0','categ\_1','categ\_2','categ\_3','categ\_4']

*#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

selected\_customers = transactions\_per\_user.copy(deep = True)

matrix = selected\_customers[list\_cols].as\_matrix()

scaler = StandardScaler()

scaler.fit(matrix)

print('variables mean values: **\n**' + 90\*'-' + '**\n**' , scaler.mean\_)

scaled\_matrix = scaler.transform(matrix)

pca = PCA()

pca.fit(scaled\_matrix)

pca\_samples = pca.transform(scaled\_matrix)

fig, ax = plt.subplots(figsize=(14, 5))

sns.set(font\_scale=1)

plt.step(range(matrix.shape[1]), pca.explained\_variance\_ratio\_.cumsum(), where='mid',

label='cumulative explained variance')

sns.barplot(np.arange(1,matrix.shape[1]+1), pca.explained\_variance\_ratio\_, alpha=0.5, color = 'g',

label='individual explained variance')

plt.xlim(0, 10)

ax.set\_xticklabels([s if int(s.get\_text())%2 == 0 else '' for s in ax.get\_xticklabels()])

plt.ylabel('Explained variance', fontsize = 14)

plt.xlabel('Principal components', fontsize = 14)

plt.legend(loc='best', fontsize = 13);

fig, ax = plt.subplots(figsize=(14, 5))

sns.set(font\_scale=1)

