Problem statement

Context

We present a set of products in which we are trying to determine actions to perform on products that present a correlation with each other, in particular negative correlation. For example, sales promotion or order amounts readjustment; further on, possibly which products should continue to be sold, which products to remove from the inventory. The data files contain historical sales data and active inventory. More details below.

- **Goal:**
  1. We want to determine which products from the store inventory are (negatively) correlated, so that we can splitt them into the ones that should be retained for sell (in particular, applying promotional actions) and the ones to discard (or to decrease the amount of order requests).
  2. Among possible ways forward, we intend to build a binary classifier with a list of products ID which could be retained in the inventory or list of products for which further actions need to be done.
- **Dataset**
  In addition to all publicly available data, we have daily sales and associated waste for each product reaching back 1.5 year. We also have the physical location of a subset around 15% of products.

Dataset looks like this:

EAN	BEN	VGR	DATUM	Ord_Akt	FSG	ANTAL	ANTAL_KG
64	3123	Lösviktsgodis	159	2019-11-05	11	3913.86	106 48.760
65	3123	Lösviktsgodis	159	2019-11-05	11	3913.86	98 44.215

We also have two months worth of receipts, e.g. for each transaction what products sold together along with timestamps (date) in a DataFrame with the following columns:

transaction_id	date	subtotal	number_of_items	ean	product	quantity	value_is_discount	incomplete_shopping

A few comments about the attributes included, as we realize we may have some attributes that are unnecessary or may need to be explained.

```
EAN : standardized barcode and marked on most commercialized products currently available at the stores. EAN is a universal code throughout the world. (EAN=European Article Number) Type int
Ord_Akt : indicator of promotions (type date)
date: timestamp variable of date type
value    is_discount : Type boolean
```

Note.
Ultimately we will merge both dataset into one ('EAN', 'quantity' are present in both dataset; value_is_discount relates to Ord_Akt), removing uneeded colums and perform feature engineering as/if needed, and perform analysis on variables interdependency (if any) for predictive analytics.

Methods

To meet Goal 1., we can either ***determine if there is correlation*** within products data and, if negative value (eg. Pearson correlation coefficient PCC = -1), then identify specific products that meet this condition
OR, preferably, we proceed in a sequential manner with the ***Exploratory Data Analysis***: the first order of operations needed to get a grasp of the what, why, and how of the problem statement, i.e. analyzing the data set by summarizing its main characteristics then visualize them.

1. Finding Correlations within products data

A bit of theory

Correlation Between two variables, this concept generally refers to to their ‘relatedness’. It allows for predictions about one variable based upon another. Nevertheless, beware that “Correlation does not imply causation”. Spurious statistical associations can be found in a multitude of quantities, simply due to chance. Often, a relationship may appear to be causal through high correlation due to some unobserved variables.

Assuming we deal with linear data,
Pearson Correlation coefficient
(also known as Pearson's r, most common measure of correlation) helps quantifying the degree to which a relationship between two variables can be described by a line. Mathematically, it is defined as ***the covariance between two vectors, normalized by the product of their standard deviations***. Alternatively, we can consider the correlation between X and Y as mathematically equivalent to the slope of the regression line of Y and X to, standardized by the ratio of their standard deviations

Let's briefly introduce the concept of covariance, that is of a statistical measure of association between two variables X and Y. Thus related to Pearson's r.

When we have a variables sample of size N:

The use of the mean in the calculation suggests the need for each data sample to have a probability distribution following a Gaussian or Gaussian-like.
The sign of the covariance can be interpreted as follows: if positive, the two variables change in the same direction; change in different directions if negative.
A covariance value null indicates that variables are independent.

To know more: Each variable is centered by subtracting its mean; centered scores are multiplied together to measure whether the increase in one variable associates relates to the increase in another variable. Finally, the expected value (E) of the product of these centered scores is calculated as a sum of the association.
Note that expected value (or "expectation") is also known as average of a random variable, or mean μ.

Implementation of covariance in python

``` {code sample illustrated for two lists with objects, here of int type, represented here by products A and B.} def mean(X): return sum(X)/len(X)

def covariance(X,Y): calc = [] for i in range(len(X)): Xi = X[i] - mean(X) Yi = Y[i] - mean(Y) calc.append(Xi * Yi) return sum(calc)/(len(X) - 1)

A = [1,2,3,4,5] ; B = [5,4,3,2,1] print(covariance(A,B))

~~~~
<b>Limitation</b><br>
Covariance is scale-dependent, i.e. it keeps the scale of the variables X and Y; therefore can take on any value. This makes interpretation
difficult and comparing covariances to each other impossible.<br>
To obtain a more meaningful illustration of the association between variables (or between vectors -- see more details below), normalizing th
e covariance is needed.</br>
![Pearson's r](https://raw.githubusercontent.com/dnzengou/products-correlation/master/correlation/img/pearson-correlation.png)<br>
![Pearson's r](https://raw.githubusercontent.com/dnzengou/products-correlation/master/correlation/img/pearson-correlation0.png)<br>
where <i>E[(X - μX)(Y - μY)] = E[(X - E[X])(Y - E[Y])] = Cov(X,Y) </i>
ρ represents the Pearson correlation (or r), σX and σY the standard deviations of each of the vectors.


***Implementation of Pearson correlation in python***<br>

``` {code sample illustrated for two lists with objects, here of int type, represented here by products A and B}

import math

def stDev(X):
    variance = 0
    for i in X:
        variance += (i - mean(X) ** 2) / len(X)
    return math.sqrt(variance)

def Pearsons(X,Y):
    cov = covariance(X,Y)
    return cov / (stDev(X) * stDev(Y))

A = [1,2,3,4,5] ; B = [5,4,3,2,1]
print(Pearsons(A,B))
~~~~

***See more here***

The correlation coefficient rests between -1 and +1, thus characterising it : negative, neutral or negative correlation.

A correlation could be positive, meaning both variables move in the same direction, or negative, meaning that when one variable's value increases, the other variables' values decrease. Correlation can also be neural or zero, meaning that the variables are unrelated.

By considering the data as arrow vectors in a high-dimensional space, we can use the angle ϴ between both vector (here a and b, representing in code samples above lists or vectors of objects of type int) as a measure of similarity.

In this exploration, we will limit ourselves to linear data. Nevertheless, we are presenting below just for information a couple of other expressions of the measure of correlation between variables: first, a special case of Pearson ρ applied to ranked (sorted) variables:

**Spearman Correlation**
But unlike Pearson, Spearman's correlation is not restricted to linear relationships. Rather than comparing means and variances, it measures the monotonic association (only strictly increasing or decreasing, but not mixed) between two variables X and Y and looks at the relative rank (or order of values) for each variable.

**Kendall's tau**
Also based on ranks between variables,but unlike Spearman Kendalls' τ does not take into account the difference between ranks — only directional agreement. Therefore, this coefficient is more appropriate for discrete data.

## 2. Exploratory Data Analysis on products data

***See more here***

---

### Reference AlibabaCloudDocs/kvstore (https://github.com/AlibabaCloudDocs/kvstore/blob/master/intl.en-US/Best%20Practices/Build%20an%20e-commerce%20short-term%20sales%20promotion%20system%20by%20using%20ApsaraDB%20for%20Redis.md)
Solving Case study : Optimize the Products Price for an Online Vendor (https://www.analyticsvidhya.com/blog/2016/07/solving-case-study-optimize-products-price-online-vendor-level-hard/)
Finding Correlations in Non-Linear Data (https://www.freecodecamp.org/news/how-machines-make-predictions-finding-correlations-in-complex-data-dfd9f0d87889/)
~~~~

Exploratory Data Analysis (EDA) and Data Visualization with Python (https://kite.com/blog/python/data-analysis-visualization-python/)
Introduction to Correlation with pandas (https://blogs.oracle.com/datascience/introduction-to-correlation) Correlation Between Variables in Python (https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/)

Use the data control module to cache the promotion commodity data to the read/write splitting instance in advance, and specify the flag for starting the promotion as follows:

%22goodsId_count%22%3A%20100%20//The%20total%20number%20of%20commodities.%0A%22goodsId_start%22%3A%200%20%20%20//The%20flag%20to%2

1. Before the promotion starts, the server cluster for the short-term sales promotion reads goodsId_start as 0, and indicates in the response that the promotion has not started.

2. When the data control module changes goodsId_start to 1, the promotion starts.

3. The server cluster for the short-term sales promotion caches the promotion start flag and accepts order requests. The cluster records the requests to goodsId_access. The number of remaining commodities is the result of the value of goodsId_count minus the value of goodsId_access.

4. After the number of orders that the short-term sales promotion system accepts reaches the value of goodsId_count, the short-term sales promotion system intercepts all requests. The number of remaining commodities is 0.