1. Data Evaluation and preparation
   1. The dataset

An E-Commerce store provided their E-Commerce Data in this dataset during the period of 12. January 2010 and 12. September 2011. This dataset consists of 541909 entries with purchases of customers in their online shop. It also consists of 8 features (5 categorical and 3 numerical) from Invoice number to Country were the customer initiated the purchase.

* 1. Features

**InvoiceNo** – *object*   
 It provides a 6-digit number for the unique identification of a purchase of one or more products. An additional letter indicates several special cases like cancellation (z.B. C123456)

**StockCode** – *object*  
 A unique label consisting of a 5-digit number with a possible letter to identify the product which was purchased. Every product has its own StockCode number.

**Description** – *object*  
 Every product has a unique description besides their Stockcode.

**Quantity** – *integer*  
 This feature provides the maximum number of purchases of the same product per Invoice.

**InvoiceDate** – *object*  
 It shows the timestamp when the purchase was initiated.

**UnitPrice** – *float*  
 Every product has its own price per unit.

**CustomerID** – *float*  
 A 5-digit number to identify the same customer per purchase.

**Country** – *object*  
 The name of the country the purchase was initiated from.

* 1. Description of the dataset

Numerical features:

The numerical features Quantity and UnitPrice show no missing values but have some outliers and extreme values. Due to Cancellations or Returns some of the values were negative.  
 The feature CustomerID has nearly 150.000 missing values, but it seems there are no possible outliers or extreme values.

A screenshot of a black and white table

Description automatically generated

Categorical features:

All categorical features besides Description show no missing values. Description itself has about 1.000 missing values.  
 The strong variety of unique values in each feature depends on its use case. In example the number of different InvoiceNo is typically higher than the StockCode because several purchases have the same product and therefore the same StockCode per InvoiceNo.

A black rectangular object with white text

Description automatically generated

* 1. Preprocessing

NaN:

InvoiceNo 0.000000  
 StockCode 0.000000  
 Description 0.268311  
 Quantity 0.000000  
 InvoiceDate 0.000000  
 UnitPrice 0.000000  
 CustomerID 24.926694  
 Country 0.000000

One can see that the missing values in the feature CustomerID nearly reach a quarter of all entries. I therefore assume that most of these entries existed for other administrative transactions. If I drop all these NaNs it is possible the small number of NaNs in Description vanishes too.

Duplicates:

There are several duplicates which I will identify and delete.

Change type:

The feature ‘InvoiceDate’ is of type object, and I will change it to type datetime. This change helps for a possible use of this feature as index.

There is an additional possibility to change the features ‘InvoiceNo’ from object to integer.

Outliers:

The handling of outliers in the numerical features ‘Quantity’ and ‘UnitPrice’ is necessary to keep the data consistent for the task ahead. The approach I use is to cut the negative values off and limit the points after 1.5 times IQR of the maximum value. The minimum and logical value for both features limits the lower end of the points. These values are 1 product for ‘Quantity’ and 0.01 pence for ‘UnitPrice’.  
 The extreme values also get handle with the approach.

after cut-off:

A comparison of a line graph

Description automatically generatedA comparison of a graph

Description automatically generated with medium confidence

after limitation:

A line drawing of a rectangular object with a line in the middle

Description automatically generatedA line graph with a square and a line

Description automatically generated

unnecessary values in StockCode:

Several values in StockCode have a letter as indicator. These often describe a process like ‘Discount’ and aren’t necessary for further analysis. I therefore filtered these codes and removed them from the dataset.

1. Tasks
   1. Implement a solution of your choice for recommending product bundles e.g. rule-based, statistics, or ML-based. Please describe any reasoning behind your solution.

Statistical attempt:

I used my cleaned dataset to create a word cloud of the feature Description

A close up of words

Description automatically generated

Here we can see the most common articles and their relative quantities indicated through their word size. The most purchased product is “Pack of 72 Retrospot Cake Cases”.

A list of absolute values od the Top20 can be a guide to create different bundles.

A black screen with white text

Description automatically generated

Rule-based attempt:

Through an iteration process I created a Top1000 2-part bundle list. It considers which combination of 2 products is the most common per InvoiceNo. I used the StockCode per product for iteration. Afterwards I translated the bundle combination in StockCode into their Description. Every bundle combination was counted to find the most purchased combination.

The list was used to find the recommended product to every purchased one and was added to the dataset.  
Combined with an ML-based solution I used the recommended item as a target and prepared the dataset for splitting. I used a KNN and SVC model to predict a recommended item. Both models have an accuracy of 70 % (KNN ca. 71 % and SVC ca. 69 %) to recommend the right item based on a purchased item.

* 1. Provide a splitting to train and test datasets. Discuss possible different splitting criteria. What other splitting criteria would you choose if you could gather more features/data?

I used the most common splitting criteria “randomized” with a ratio 4:1 (20 %) and the cross validation to find the best parameters for my ML-Algorithm.

Other criteria are the stratified and Time-based Splitting. The first one is used for data which is unevenly distributed across the classes. It is therefore necessary to represent every class properly in the training and test set. The Time-based is often used in Timeseries to train i.e. Sarimax models. The data can be split into several years to train the model, to predict the same period of the test set and then compare it with the test set.

We also can split the data in feature-, domain- or cluster-based criteria if one can gather more data/features. These criteria try to represent every feature (weighted) and domain (different sources) adequately in both sets. The cluster-based is used if the data can be group (i.e. different part of clothes) and therefore every set gets parts of each group.

* 1. Discuss the size of the output list and how it can be decided per product.

My word cloud shows that every product is unevenly purchased in their quantity and a logical approach reveals that some products are often combined in a purchase like T-Lights and T-Light holders. A model would recommend those combination and frequencies more and therefore give a certain product a higher output list, then other.  
Besides this there can also be customer-based and time-based factors which influence the quantity of a certain product. But with the right analysis of these factors, experimentation, and iteration one can determine the output lists.

* 1. Discuss/implement any price computation per bundle e.g. the sum of products prices.

I used my Top100 per product list to generate a total price based on the highest amount of quantity purchased and combined it with the products of my bundles. I therefore used the UnitPrice to generate the TotalPrice for each bundle.

* 1. How would you evaluate the business impact of the solution and share the outcome with the internal stakeholders?

1. Optional tasks:
   1. Implement a regression model for the products’ prices (UnitPrice) prediction. Is the provided data sufficient to predict the price? What other data would you like to gather to improve your solution?
   2. Your bundle's code is a great success and the Frontend team wants to use it in production. Implement a simple Rest API to serve the bundles with an endpoint getting as a parameter a product ID and returning a list of products and the price for the whole bundle. Ideally, provide a Dockerized version of the implemented API.