UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

EIT DIGITAL
SOFTWARE AND SERVICE ARCHITECTURES

MASTER THESIS

Real-time persona classification of visitor event streams with Tensorflow graphs

Kristóf Morva
2018

MASTER THESIS

Título:	Real-time persona classification of visitor event streams with Tensorflow graphs
Autor:	Kristóf Morva
Tutor:	Dr. Carlos A. Iglesias
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

MASTER THESIS

Real-time persona classification of visitor event streams with Tensorflow graphs

June 2018

Abstract

With the steep emerging of machine learning and big data technologies, it is easier every day for companies to take advantage of all their data about their customers. One of the hottest topic is personalization to improve user experience, for example the "Did you forget this product?" suggestions a few supermarket implements, or recommendations based on the user's basket and purchase history.

One of the companies to utilize machine learning in their daily life is REWE Digital, one of the biggest supermarket-chains in Germany, which handles millions of customer events every day.

The broad goal of this thesis is to present a way to effectively predict the persona (for example "Family with two high school children", "Couple without children", "Working single", "Older family with a teenager", "Old couple" or "Student") of a web session in real-time based on their behavior, like events of visiting and buying certain products, basket size and price, et cetera.

First, it will be shown how web sessions can be clustered without hundreds of URL features expected by the paper of Olfa Nasraoui, Hichem Frigui, Raghu Krishnapuram and Anupam Joshi in case pages can be broken down to a fix amount of attributes. Then, the customer events shared by REWE will be prepared and transformed to feature vectors, with which several clustering methods will be tested and compared to find out which ones are useful for web sessions. After having the clusters, this thesis will introduce the possible ways to predict the corresponding cluster of a new session real-time. Lastly, possible methods to validate the effectiveness of the formed clusters will be discussed.

Keywords: Persona, classification, clustering, web mining, real-time, event streams

Contents

CHAPTER 1

Introduction

This chapter is going to introduce the context of the thesis, including a brief overview of all the different parts that will be discussed. It will also break down a series of objectives to be carried out during the realization, and it will introduce the structure of the document with an overview of each chapter.

1.1 Context

In the beginning of the century most companies were advertising their best products in order to attract more customers. You might also remember seeing ads like "How to lose 30kg in 2 weeks" even if you were thin as a stick, or "Get rid of your nail fungus in these 3 easy steps" even if you never had any. In the past years this trend started to disappear when companies realized that you can be much more effective with targeted ads. It's clearly indicated by the increasingly popular tracking IDs (and tracking blockers as a counter-reaction on the consumers' part). Amazon for example started putting much effort in targeting early - not without a result. According to Fortune [18], in 2012

> [...] the company reported a 29% sales increase to $12.83 billion during its second fiscal quarter, up from $9.9 billion during the same time last year. A lot of that growth arguably has to do with the way Amazon has integrated recommendations into nearly every part of the purchasing process [...]

To take advantage of the emerging technologies related to Big Data, many retail companies started to track visitor and customer data in order to be able to predict, which would be the most probable product or service each customer would be interested about, alongside with REWE Digital, the data of which will be used in this project. The company currently handles about 13 millions of events daily, and almost 1 billion in total.

1.2 Objectives

The objective is to answer the following questions:

- How to identify valuable clusters in web sessions?
- How to predict the cluster of a session real-time in event-based systems?
- How to validate the effectiveness of web session clusters?

1.3 Tasks

The project aims to answer the aforementioned questions, through the following tasks:

- Acquire a large enough dataset
- Identify the important features of the sessions
- Build the sessions by processing the dataset

- Explore the possible clustering techniques to group the sessions
- Build a model to predict the cluster of real-time sessions
- Validate the formed clusters

1.4 Structure

The rest of the thesis is categorized as the following:

- Chapter 2 introduces the most influential 3rd party technologies which are used by this paper
- Chapter 3 describes the method of processing the data to extract meaningful features
- Chapter 4 uses the formed input vector to build clusters, based on which it trains a classification model in order to be able to predict a cluster in real-time
- Chapter 5 discovers the ways of validating if the resulting clusters are meaningful or not

CHAPTER 2

Enabling Technologies

This chapter offers a brief review of the main technologies that have made the implementation possible.

2.1 Python

Python is a high-level programming language, one of the most used ones for scientific purposes and the fastest growing language currently (Figure 2.1).

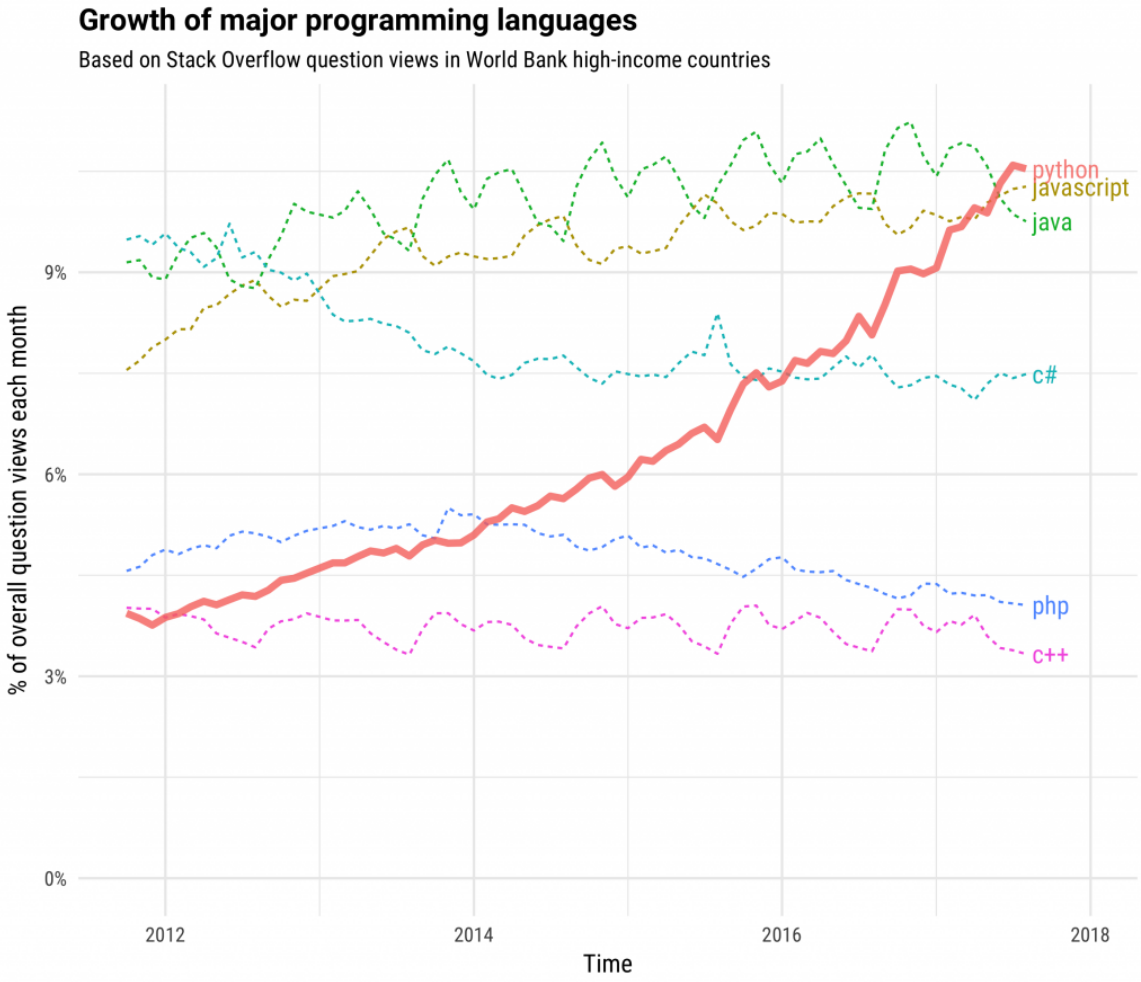

Figure 2.1: Growth of Python [24]

While the code execution efficiency can't compete with lower-level programming languages like C++ [3], it is a common practice to write the libraries in C++ or Go, and only create a Python wrapper around it. This way programmers can utilize the convenient and less error-prone syntax Python provides, while the speed won't be degraded drastically.

Having high level of syntax and libraries (especially regarding math and the handling of series) is probably the primary reason the language provides many choices in the topic of machine learning, like the ones described below.

This thesis will use Python for every included source code.

2.2 Scikit-learn

Sklearn [8] is a high-level machine learning package for Python, providing a very broad range of algorithms: clustering, classifying, data preparation, visualization, helpers, etc.

2.3 TensorFlow

TensorFlow [4] is an open-source machine-learning framework developed by Google. Since it provides both low-, and high-level functions with integrated CPU and GPU support, it can be used for almost any machine-learning problem. It will be used with the Keras wrapper in this thesis.

2.4 Keras

Keras [5] is an even higher-level abstraction over TensorFlow (with support for other machine-learning frameworks), specializing to build neural networks with the least code possible. It'll be used to build a classification model, with which web sessions can be classified.

2.5 BigQuery

The events of the company's system are stored in Google Cloud's BigQuery [9], which is a serverless database solution with an SQL-like query language and a focus on big data. Customers of the service include SAP, IBM, New York Times and many others.

One of the advantages of using BigQuery API is, that it handles paging automatically. In this context it means, there isn't a need to store all GBs or TBs of data in the memory as the result of a huge query. Instead, when the query is executed, BigQuery will build a temporary table with the results, from which data can be fetched in more separate batches.

In this thesis it will used to query the web session events from Python.

2.6 Kafka

Apache Kafka [2] is a distributed messaging and storage service built on Apache Zookeper (with other choices), providing first-class efficiency, numerous guarantees and a small footprint. This thesis is taking advantage of the concept, that every client can subscribe to specific type of events, and every client can also trigger its own events. As a result, services generating events do not need to care about where and who exactly is going to catch these -

if another program is interested in some groups of events, it can subscribe to these through Kafka.

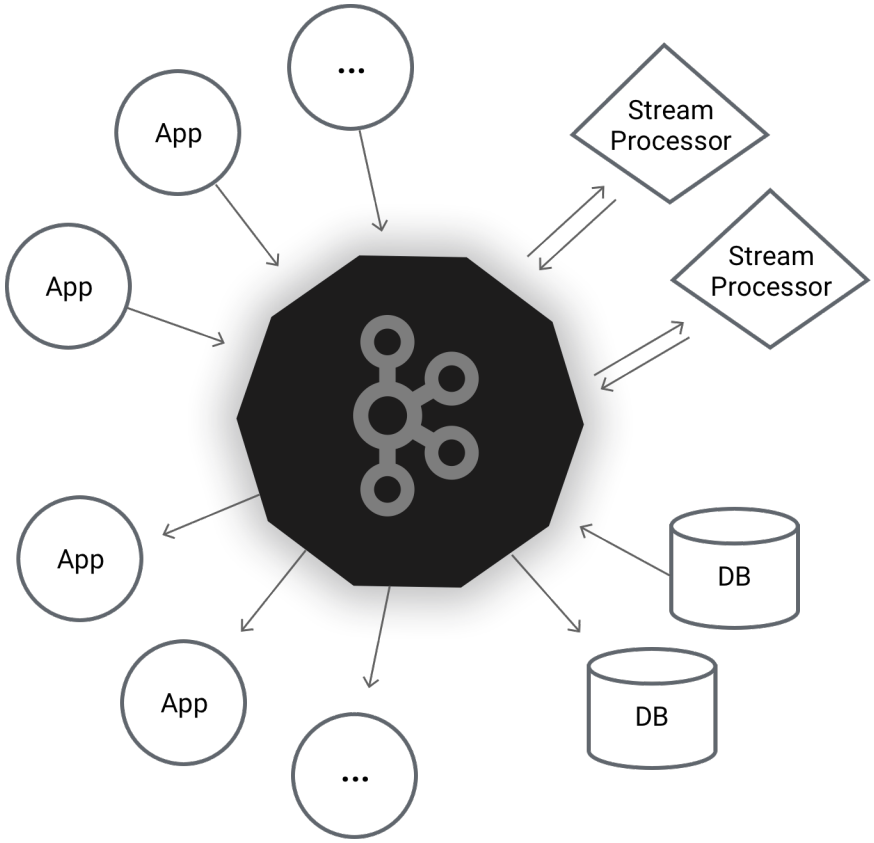

Figure 2.2: Architecture of Kafka [2]

In this thesis Kafka will be responsible to handle the events for the real-time persona classification.

CHAPTER 3

Data Preparation

This chapter describes a method to identify the possible features of web sessions, and the used methods to build the input vector for millions of semi-independent web events.

3.1 Methodology

In the paper *Extracting web user profiles using relational competitive fuzzy clustering* [22], Olfa Nasraoui, Hichem Frigui, Raghu Krishnapuram and Anupam Joshi states that

> [...] (a feature vector based clustering algorithm) is not suitable for clustering user sessions. This is because of the high dimensionality of the feature space (there are usually several hundred URLs in a typical Web site), and thelikely correlation between the features (URLs that often co-occur in the same session). The Web sessions are too complex to convert to simple numerical features [...]

However, this thesis will introduce a solution to overcome this limitation in case the website consists of pages with the same structure (like products, books, movies, etc).

The proposal is an aggregated score system for the selected attributes of the pages. More precisely, some attributes will be defined which apply to all pages, then the pages will be scored for each attribute, and lastly, the scores of the visited pages will be aggregated in each session.

A book store can be considered as an example. Let each book detail page have a release year, a category (action, sci-fi, drama, horror, etc), and any further book-specific qualities. First the feature set has to be decided, for example Antique, 1950s, 1960s, 1970s, 1980s, 1990s, 2000s, 2010s, Action, SciFi, Drama, Horror. In most cases the values should be comparable normalized numbers (detailed in Section 3.4.3), for example projected to the [0, 1] range. Then, each book will be scored for every feature. For example Stephen King's It might have a 1 score for 1980s, 0.9 for Horror, 0.1 for Drama, and 0 for the others. Then, for each session the values of the visited (or purchased or rated) books will be aggregated, and this aggregated vector will be considered as the feature vector for this session. The computation of the aggregated values is arbitrary - in these examples the average of the individual values will be used, like presented in table 3.1.

Book	Old	1970s	1980s	Modern	Horror	Drama
Dracula	0	0	1	0	0.5	0.5
Hamlet	1	0	0	0	0	1
Othello	1	0	0	0	0	1
Avg	**0.67**	**0**	**0.33**	**0**	**0.17**	**0.83**

Table 3.1: Calculate the average of book features

The last line is now a normalized, numerical array, which is fully compatible with feature-vector based clustering algorithms. By clustering a big amount of session vectors, book shoppers with similar attributes could be distinguished. It is very useful if besides books, events or movies are planned to be suggested too, as probably people in the same cluster will behave similarly and like similar topics.

3.2 The available data

The data to be used in this thesis are all REWE website events of the past years, stored in Google BigQuery. Altogether it has 1 billion records, taking up more than 320GB storage space. The schema is the following:

- reference: Browser cookie, lives until it is cleared manually or by the browser.
- customer_uuid: If the user is logged in, it is their ID
- activity: The event that has been logged. Can be:
  - shop_visited: The visitor has opened the shop.
  - product_visited: The visitor has visited a specific product. It does not log when the product is displayed in a list.
  - user_left: Visitor session timed out.
  - basket_created: A basket has been created for the visitor.
  - added_to_basket: The specific product has been added to the basket.
  - removed_from_basket: The product has been removed from the basket.
  - updated_in_basket: The amount of the product in the basket has been changed.
  - search_executed: The visitor was searching for the specific keyword(s).
  - campaign_hit: The user has clicked on an ad.
  - market_selected: The user has changed the shop.
  - checkout_started: The user has started the checkout.
  - checkout_finished: The user has finished the checkout.
  - order_created: The order has been placed.
  - order_cancelled: The order has been cancelled.
  - order_completed: The order was completed.
  - user_registered: The visitor has created a new user account.

  - user_logged_in: The visitor has logged in.

- mandator: The company producing the event. The focus is on REWE for now.
- received_at: Time of the event.
- payload: Details of the event (product ID, basket ID, etc). It is stored as a JSON object, so while it can be parsed, filtering it directly in the database engine might be too complex or expensive.
- is_assumption: Is it an actual happening or an assumption? For example if the session is not used for an hour, it is assumed that the user has left.
- id: Event ID.
- producer: Subsystem which created the event.
- cause, threat_level: Internal use.

3.3 Session identification

In order to prepare the available data, it is important to know what is exactly needed from it and in what type of format. As the goal is to identify sessions on the website, the model needs to be trained with sessions and their properties. A session lasts from entering the website until the user places an order. Sessions without orders are currently out of scope. Multiple attempts have been done before reaching the final solution.

3.3.1 Simple linear

The first attempt to find sessions was simple: all order and product visit events have been ordered by reference in reverse chronological order. The program iterated through it, and if it found an order, it collected the product visit events until it found another order or another reference. At the end, it had the orders with associated visited products, and from the difference of the order time and the first product visit time it also knew the session length. However, it has been discovered that some order events (which are done from the phone application) have no reference ID as they are fed from the order service, which has no knowledge about the session ID (browser cookie).

3.3.2 Separate orders and linear

In order to solve the issue in the previous attempt, the orders and the product visits have been separated. The program still had to know which order belongs to which session, so

the following method was used:

- All orders were fetched and put in a map with the basket ID as the key (which was read out from the `payload`) - it is easily doable as there are < 5 million orders in total.
- The program went through the dataset linearly, but now looking for a checkout event instead of an order event. It is useful, because checkouts have both session ID and basket ID, and they probably lead to an order.
- If it found the given checkout's basket ID in the order array, it knew that this checkout has lead to an order.
- It read all product visits until it found a different checkout or a different session ID.

Later the algorithm has been improved to also check for login events, as product visits not always happen if someone is only putting products to the basket from the search list. However, it made the processable events from 1 to 4 million per day, which made it significantly slower - but the impression was, that it won't cause problems in the production environment. The issue came when 2 days of dataset were fetched instead of 1, as BigQuery threw an "Resources exceeded during query execution: The query could not be executed in the allotted memory" error, which was most likely caused by the ordering, as ordered datasets cannot be parallelized. The ordering by 2 columns had to be prevented.

3.3.3 Separate everything

A possible strategy might have been to fetch and order only one day in each iteration. However, it would have led to the corruption of all events which were not ended before midnight, as the program would not have seen the remaining product and shop visit events on the next day which belonged for an order this day.

At last, a different solution was used to prevent ordering altogether (Listing 3.1):

- First, all orders are fetched.
- Then, for each order, the program found the corresponding checkout through the basket ID, so now it also knew the reference (session ID) of the order.
- For each order, it found the latest shop visit event (which is older than the order, but from the older ones the newest), so now it knew the beginning of each session too.
- Finally, for each reference it collected all product visits, which happened between the known visit and order events.

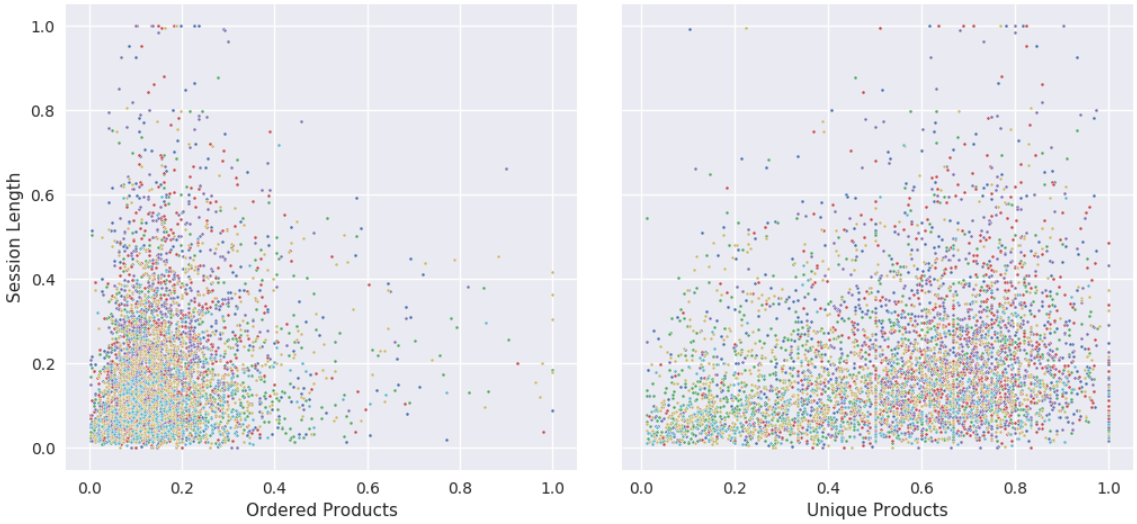

Figure 3.1: Plot of the second session identifier method

Listing 3.1: High-level code of collecting sessions

```
orders = loadObject('orders', getOrders)
sessions = getReferences(orders)
getSessionStartTimes(sessions)
getVisitedProducts(sessions)
return sessions
```

Even though it requires more queries in total, all these steps can be batched (queried without sorting or fetching the whole dataset), so it'll not run into resource problems anymore. It was expected to see some differences between this and the previous attempt for the same day, as some events might be missing or might be disordered due to software bugs or network latencies/errors, and the different logic in finding the sessions might be more or less sensitive to some events. Surprisingly, the found points are very similar (compared with Figure 3.1 and figure 3.2), leading to the assumption that the events are quite reliable.

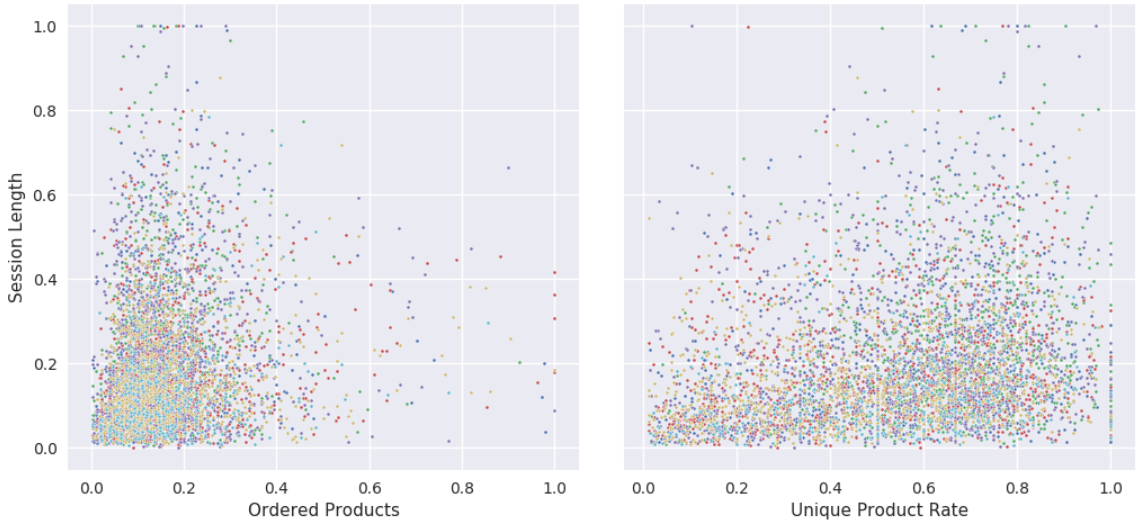

Figure 3.2: Plot of the third session identifier method

3.4 Feature extraction

3.4.1 Used features

After collecting all the sessions, extracting the needed features is a relatively easy task. The features used by this thesis:

- Session length
- Basket size
- Basket unique item ratio
- If the session began on a weekend
- If a coupon was used
- Which day quarter did the session begin
- Aggregated product property scores (for example bio=1 means that mostly bio products have been bought, sweets=0.003 means a very little amount of sweets were ordered):
  - Price scale compared to other products in the same category
  - Organic
  - Vegetable
  - Fruit

- Convenience
- Foreign
- Soft drink
- Dairy drink
- Coffee
- Alcohol

A few generic rules have to be ensured when the feature vector is built.

3.4.2 One-hot encoding

Categorical columns must be separated into multiple boolean (yes/no, 1/0) features. For example the hour (or the day quarter) feature cannot be present as a single column, as it would mean, that 22:00 o'clock is a little smaller than 23:59, but it is much bigger than 00:01. However, in the sense of purchase time, it only means that someone ordered at 23:59, while someone else at 00:01, not that the latter is smaller than the former. As a result, the time has to be first converted to a fix amount of values (categories), like one category for every hour or day quarter (00:00, 01:00, 02:00), then each category will represent a feature (column). These types of columns are called categorical columns (or, after separation, boolean columns), in counter to continuous or numerical columns, like price.

3.4.3 Normalization

Columns need to be normalized. It means, that features should have a range of values, usually between 0.0 and 1.0, not their original value. For example, if the model would be trained with the original values of "Basket size" (which is for example between 1 and 500) and "Viewed recipes" (which is in this example between 0 and 10), it would mean that the basket size is 50 times more important. However, in the real life they are probably wanted to be similarly important, so a range of available values for all columns needs to be defined, for example let's say 500 basket size means 1.0 and 10 viewed recipes also mean 1.0, then these can be compared relatively (50 products in the basket and 1 viewed products are both 0.1).

While the task of a classification (Section 4.3.1) is to optimize the weights for the input features, clustering (Section 4.2) groups data by **using** the defined weights of the features. It is a very important distinction, as for classification it is completely natural to have square meters and list price as features without any transformation, but in many clustering

algorithms (like K-Means) it would mean that $\$100 > 60m^2$ as they are compared numerically by Euclidean distance, resulting in a totally unbalanced clustering, demonstrated in Figure 3.3.

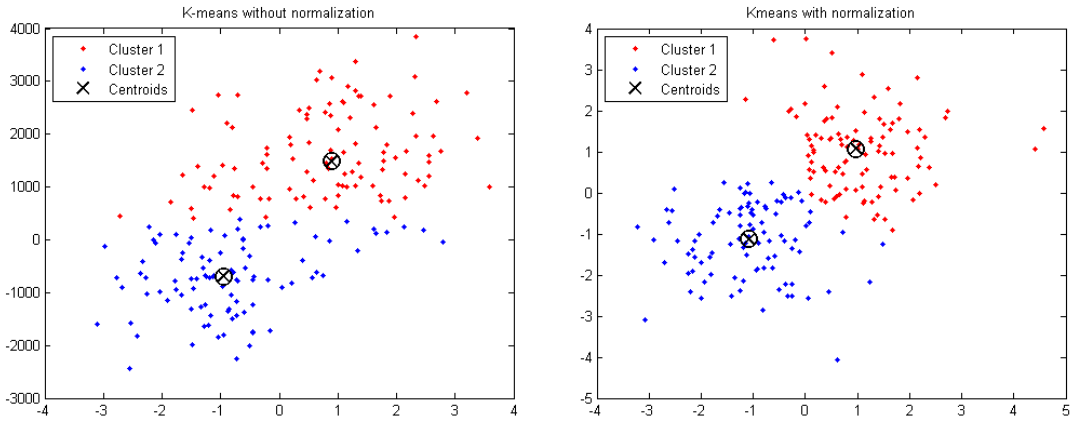

Figure 3.3: K-Means before and after normalization [7]

As a result, it is necessary to normalize the input data **before** clustering, which is usually done by compressing all features to the [0, 1] range, for example having both \$1.000.000 and $60m^2$ equal to 1, \$0 and $0m^2$ equal to 0, and the in-between values linearly (or non-linearly) spread between these 2 endpoints.

3.4.4 Scaling

Outliers have to be capped. The previous point still breaks if there is only one listing out of thousands with \$1.000.000 listing price, while all the others are under \$200.000, because as a result, listing price will still be 5 times less important than the square meters just because of one listing.

Figure 3.4 shows an example with continuous thickening on the Y axis and one single huge outlier on the X axis (yellow dot on the bottom right), which drives all other points to the left edge of the plot.

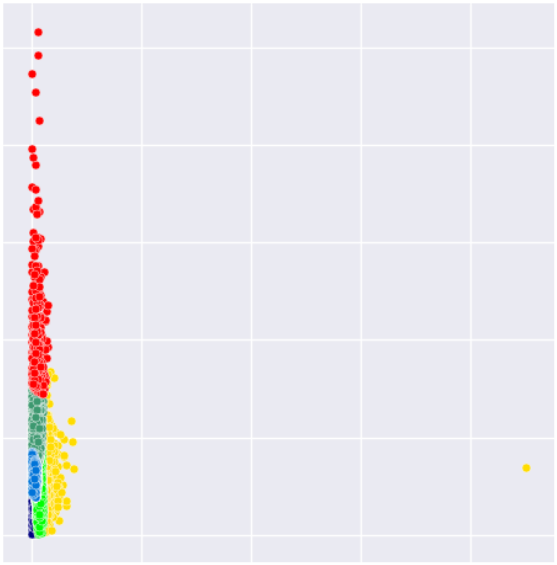

Figure 3.4: Unscaled data points

In these situations, one would rather say "Let's consider that this one costs only $200.000, as it is not so important if a listing is very expensive, or ultra expensive". In this case we'll lose some accuracy, but it helps avoiding a few features to dominate throughout the whole dataset. Figure 3.5 shows the result, if we cap 10% of the values from all directions, and lay out the rest linearly. As it is visible, too large and too small values are now stuck to the edges of the plot.

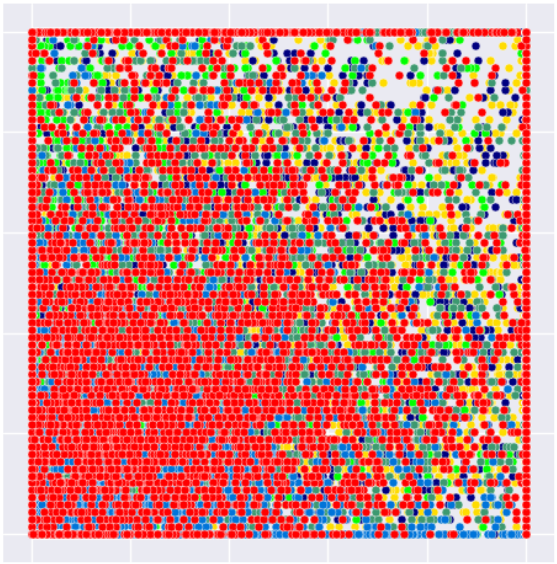

Figure 3.5: Scaled data points

In some scenarios however, clamping to a range might not be enough. If the data points

are growing thinner continuously by distance, it is hard to find a point to cut at, as either most of the pair-wise difference in the data will be lost if the cut is deep, or a big part of the data might remain unbalanced in the cut is small. Figure 3.6 shows a case, when a part of the data on the x axis is already cut down, but rest of the samples are still thickening nearly linearly towards the zero point.

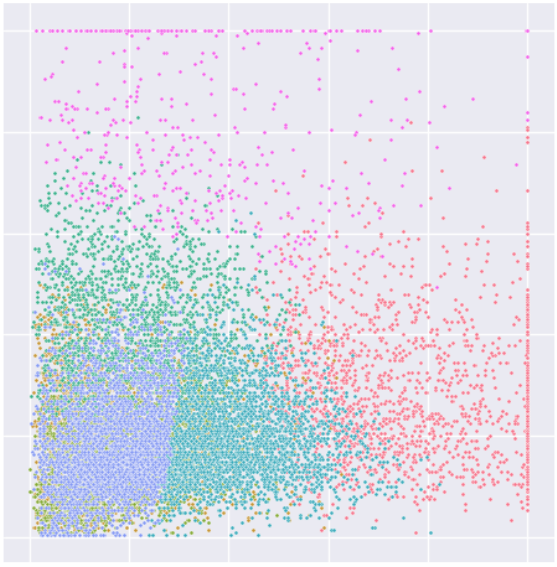

Figure 3.6: Linear scaling

It'll yield a quite unbalanced distribution, as on Figure 3.7. At the cut points (beginning and end) the samples are gathering, while usually growing / decreasing in-between.

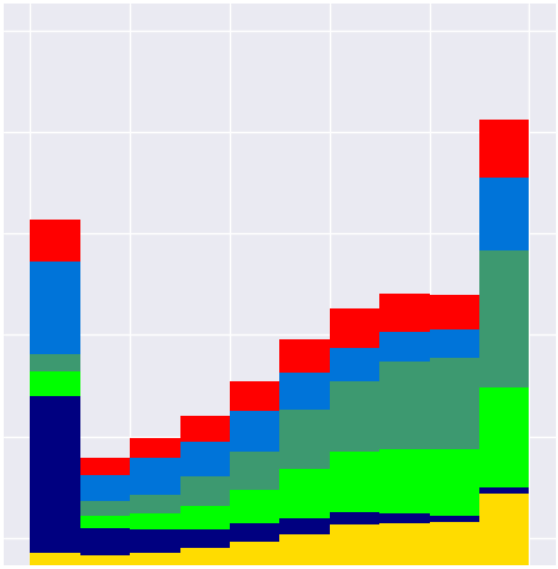

Figure 3.7: Distribution of the linear scaling

In these cases introducing a non-linear scaling might be considered. It results in a vector, in which most of the smaller-bigger relations between samples are kept, but the distance between them is better distributed. While some accuracy in pairwise distance calculations is lost, the resulting vector will compare must easier with other features, helping the avoidance of some features dominating the dataset (Figure 3.8 and 3.9).

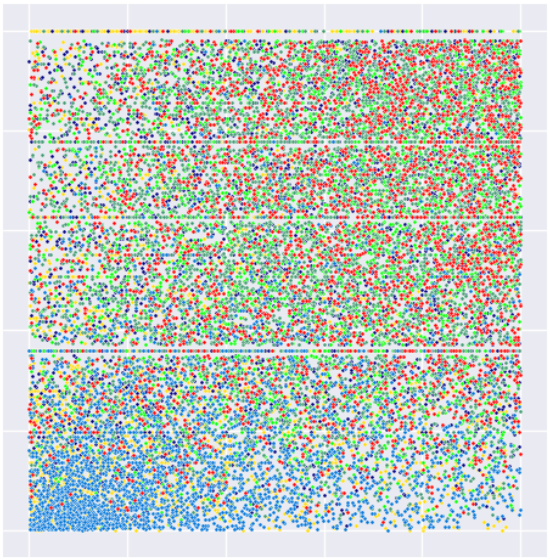

Figure 3.8: Non-linear scaling

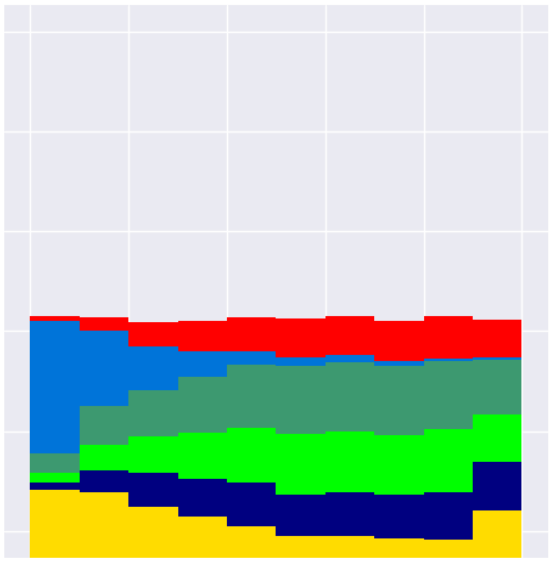

Figure 3.9: Distribution of the non-linear scaling

3.4.5 Implementation

A file has been created for each feature, which extracts the specific feature data from a session. When building the input vector, the program just iterates through these, and combines them into a list. Most of them are quite straightforward, like listing 3.2.

Listing 3.2: Coupon Used feature

```
from datetime import datetime

title = 'Coupon Used'
isCategorical = True
columnCount = 1

def process(session):
  return 1 if session['coupon_used'] else 0
```

The `isCategorical` indicates if the column should be handled as categorical.
Some features (especially categorical ones), like the "Day Quarters" (listing 3.3) might return more columns.

Listing 3.3: Day Quarters feature

```
import numpy as np
from datetime import datetime

title = 'Day Quarters'
isCategorical = True
valueTitles = [
  'Q1AM', 'Q2AM', 'Q3AM', 'Q4AM',
  'Q1PM', 'Q2PM', 'Q3PM', 'Q4PM',
]
columnCount = 8

def process(session):
  dayQuarters = np.zeros(8, dtype=np.int8).tolist()
  dayQuarters[datetime.fromtimestamp(session['time_end']).hour // 3] = 1

  return dayQuarters
```

The clamping and normalization process has been defined in a simple function (listing 3.4). It caps the top and bottom 10% of outliers, and spreads the inner values linearly.

Listing 3.4: Clamping and normalization

```
def clampAndNormalize(column, ratio = 0.1):
  minIndex = math.floor(ratio * len(column))
  minValue = np.partition(column, minIndex)[minIndex]
  maxValue = np.partition(column, -minIndex)[-minIndex]

  # We're only working with positive numbers for now
  if maxValue != 0: # Totally empty feature
    # Only a few occurence of differences (like 0 0 0 0 0 1 0 0 0 0 0 0 0)
    if minValue == maxValue:
      minValue = np.min(column)
      maxValue = np.max(column)
    else:
      np.clip(column, minValue, maxValue, out=column)

    column -= minValue
    column /= (maxValue - minValue)
```

CHAPTER 4

Model building

This chapter describes the possibilities to cluster and classify web session data, and the workflow to get to real-time predictions.

4.1 Introduction

The work of getting real-time predictions from the input vector will be done in multiple steps:

- Section 4.2: Choosing a clustering algorithm and cluster the data
- Section 4.3: Build a classifier with the cluster identifiers as the labels
- Section 4.4: Predict real-time with the built classifier

The data flow is visualized in figure 4.1.

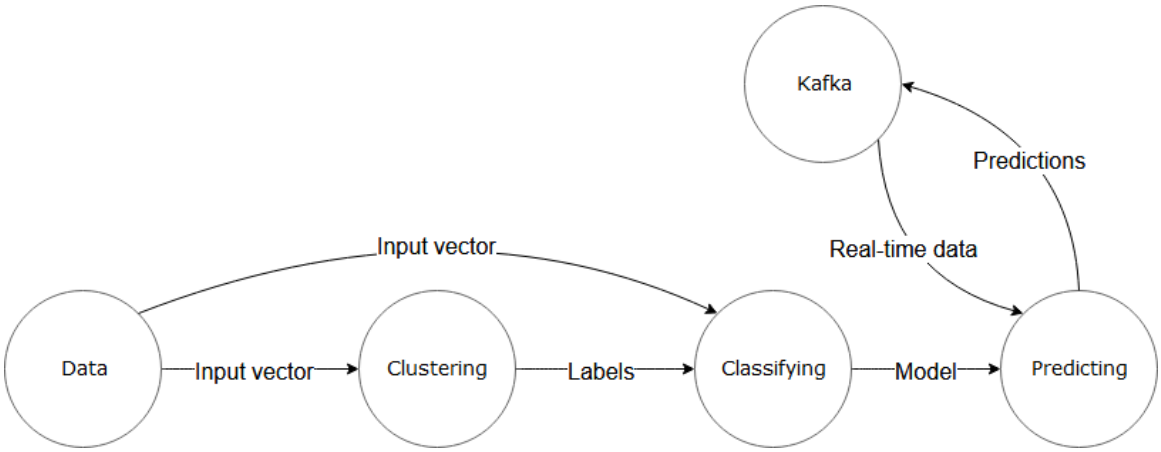

Figure 4.1: Overall architecture of the project

4.2 Clustering

4.2.1 Principles

After constructing the input vector, all sessions (called samples or observations) will represent a single point in an n-dimensional virtual space, where `n` is the number of features (so every feature will define a dimension). Figure 4.2 represents it in a 2-dimensional space.

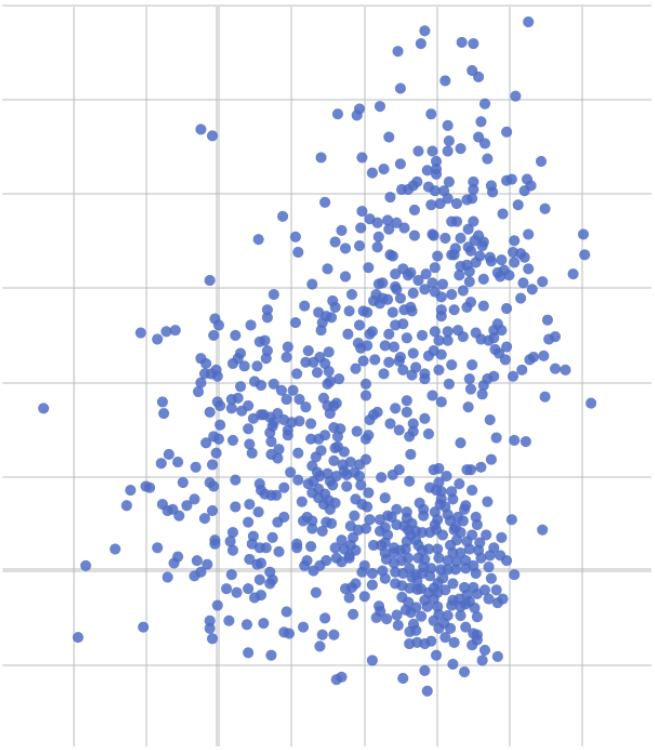

Figure 4.2: Input points in a 2-dimensional dataset [14]

As there are no labels (pre-defined groups for the sessions), an unsupervised method has to be used to discover the potential customer groups, out of which the most common choice is clustering. A clustering algorithm tries to find sets of points in a dataset which are more alike with each other regarding some attributes than with other nodes, like in figure 4.3.

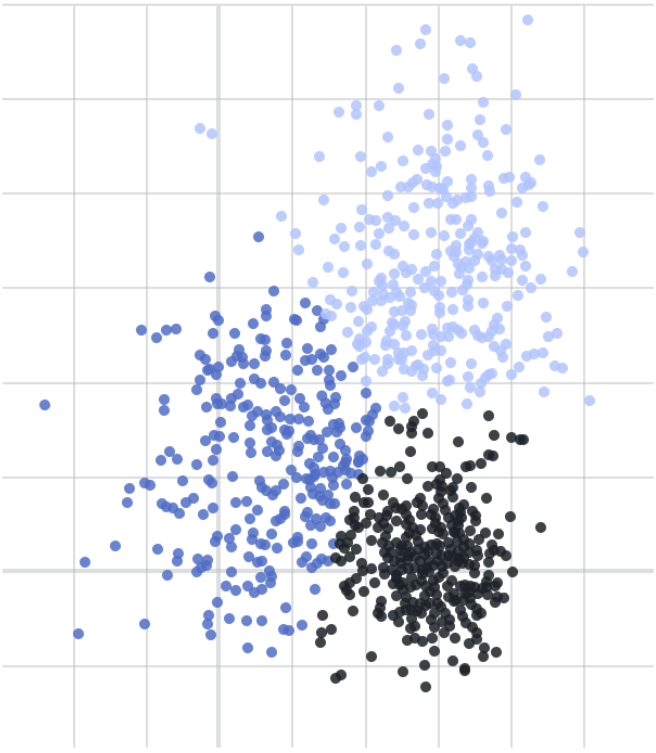

Figure 4.3: Clusters in a 2-dimensional dataset [14]

Here, the edge of the clusters are not well-distinguishable. It is the case in almost every scenario involving big data - we'll assign some nodes to a cluster because they might be nearer to its center by 0.00001%. In this case, it is clear that sometimes it'll lead to some errors, for example when detecting personas, an Old Couple might be recognized as a Young Couple - however, aiming for 100% predictions is not the goal (as in almost every case it is impossible), but to do a best guess based on the available data.

4.2.2 Determine the number of clusters

One of the challenges in clustering is to know how many clusters we'd like to have in case the chosen algorithm can't find the optimum itself - in some cases it is trivial, for example if one would like to recognize numbers between 0 and 9, they'll need 10 clusters. However, for the problem at hand the number of personas is not trivial. In a broader perspective, every person is different, in another view they all act the same - they come to the shop and buy products. The goal is to find a golden path in order to create a usable model - group people who act similarly enough to be reached effectively the same way.

4.2.2.1 Manual analysis

It is usually viable to explore the clusters manually. For example, REWE Digital has identified the 6 most common personas purchasing in its shops by talking to and surveying hundreds of actual customers (it is the method to be used in the rest of this thesis). If the research is deep enough, it is expected to recognize some of the discovered groups of people in the formed clusters if all sessions are analyzed on a website and then classified based on similar attributes, which were also used in the research (time spent on the website, amount of bought products, product properties, etc). However, it might also be interesting to see what results a larger number of clusters would return to see, what other bigger groups can be identified, which might have been missed by the research.

4.2.2.2 Silhouette analysis

The Silhouette analysis defined by Peter J. Rousseeouw [25] gives a score between -1 and 1, which represents, how "far" (which might not be an euclidean distance) are the points of a cluster from the nodes of other clusters. 1 means the distance between clusters is high (it is the best possibility), 0 means they touch each other (error-prone clustering) and -1 means items are probably assigned to the wrong cluster. For a week of data the result is represented by listing 4.1.

Listing 4.1: Silhouette analysis results

```
Score for 2 clusters: 0.077969
Score for 3 clusters: 0.125521
Score for 4 clusters: 0.155788
Score for 5 clusters: 0.180097
Score for 6 clusters: 0.193924
Score for 7 clusters: 0.167241
Score for 8 clusters: 0.152105
Score for 9 clusters: 0.149885
Score for 10 clusters: 0.138457
```

As it can be seen, the best number of clusters according to scoring is 6. However, it must be noted, that based on multiple tests the score highly depends on the data and the selected features, so automatizing this method might not be trivial, but it can provide insights.

4.2.3 Algorithms

4.2.3.1 K-Means

K-Means is a well-known and wide-spread clustering algorithm with a huge amount of variations and applicable heuristics [27]. First, it selects (randomly or with a special initialization step) `k` centrals (where `k` is the amount of clusters it is expected to have), and assigns every point to their nearest centroid (where nearest is defined as the squared Euclidean distance for the original K-Means). Then, it iteratively moves these centroids and assigns the points until the mean distance of all points and their assigned centroid is minimal.

It must be noted, that the algorithm is not consistent, but depends on the initialization step of placing the centroids, as shown in Figure 4.4 (some items are misplaced, because these plots are 2-dimensional projections of a multi-dimensional set).

Figure 4.4: Different random seed for the same dataset using K-Means

A solution is to set a fix randomization state, so the same centroids will be always selected. However, it still indicates, that the verification of the clusters will be hard, since changing the random state might change the end result. It is probably worth to try out many different states, and record the significantly different versions.

Another issue with K-Means is, that it only handles continuous data. After generating the clusters, the results (Appendix A) show, that some of those features are dominating, which are categorical (therefore represented as 0/1 columns), for example cluster 4 has a Q4AM of 1, cluster 3 has a Q1PM of 1, cluster 6 has a Q2PM of 1 and cluster 2 has a Q3PM of 1. It means, that these clusters have been separated mostly by the time of the purchase - while in the real life, the time of the day is certainly not enough to identify

personas.

4.2.3.2 Mini-Batch K-Means

One of the K-means variations is Mini-Batch. It fetches a random batch from the whole dataset with `k` elements, then updates the centroid for each item in the current batch based on the average of this item, and the currently assigned items.

According to D. Sculley's paper [26], in many cases it is much more effective than K-Means, while only a small amount of items are assigned to the wrong cluster.

However, in a tested set of web sessions, while for some random seeds the results were almost exactly the same, in some other cases the difference was drastic, as represented in Figure 4.5. It is advised to try out multiple different combinations, or tweak the features to get more consistent results.

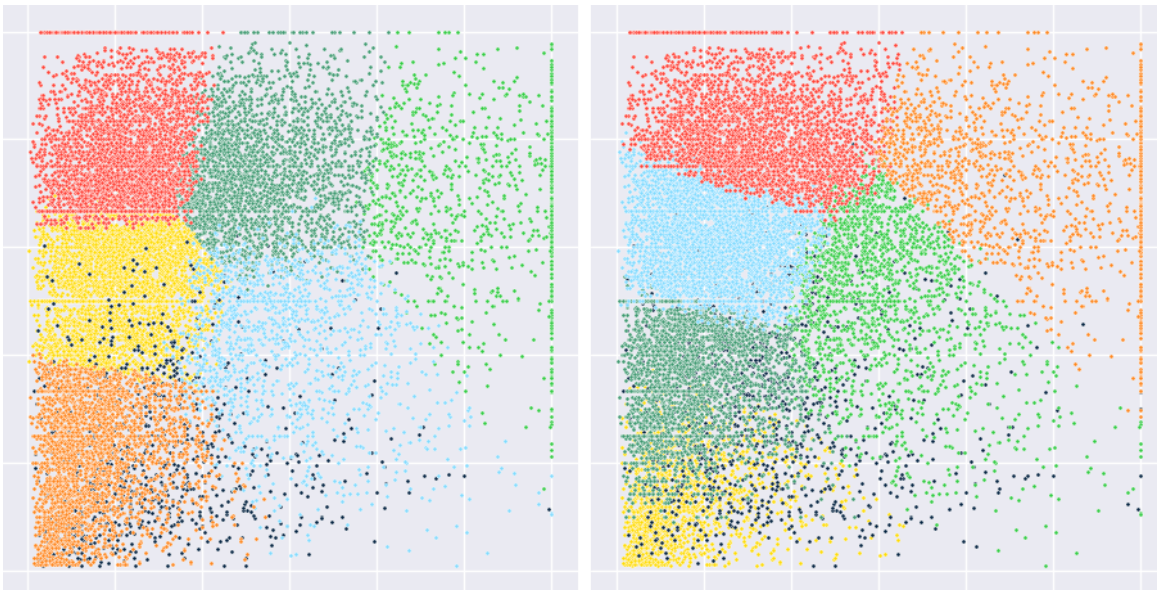

Figure 4.5: K-Means and Mini-Batch K-means for the same seed and dataset

In addition, it also suffers from the unbalanced clustering because of the dominating categorical columns.

4.2.3.3 DBSCAN

DBSCAN is a density-based algorithm. It constructs clusters from items, which have a maximum of `eps` distance between each other, and at least `min_samples` of them are in a group. It is much more suitable for arbitrary-shaped clusters, but will be probably ineffective for any type of web session, as some part of these will be too dense (very typical shopping),

while other parts too sparse (unique needs). Tweaking the parameters to get 5-10 clusters, something like in Figure 4.6 is expected to be seen.

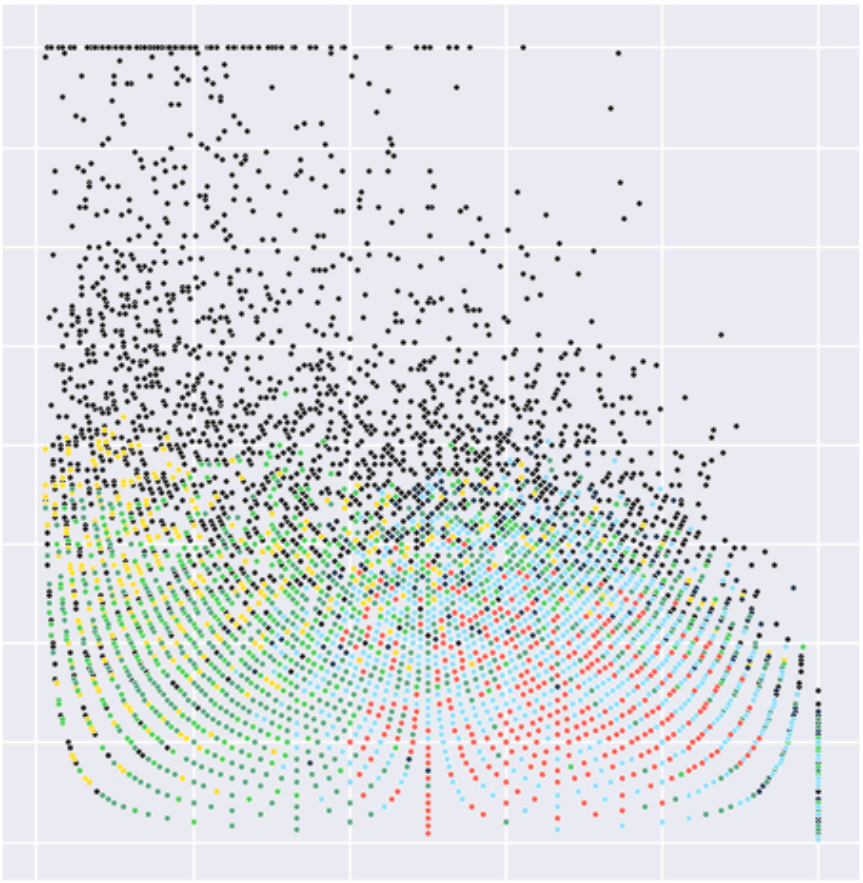

Figure 4.6: Applying DBSCAN to web sessions

The black points are ignored, as there aren't enough of them near each other. It means, almost half of the sessions are immediately dropped, which is probably unacceptable in any context. If the amount of nodes required to form a cluster is decreased, it'll result in hundreds of clusters, if the maximum distance between each other is increased, a few huge clusters will be formed, both of which are unhelpful when clustering sessions.

4.2.3.4 Agglomerative Clustering

The idea behind Agglomerative Clustering is to build a tree of the items from bottom-up. In the first step, each item has its own cluster, then, based on a distance metric, the nearest clusters are merged, and placed one level above in the tree. If the tree reaches the expected amount of clusters, the algorithm stops.

Different distance calculations can be used for different needs. If the goal is to have identical size clusters, the `ward` linkage can be used. With `average`, those clusters will be

merged, which have the smallest average of distance between all the points, while `complete` joins clusters, which have the smallest distance between the two farthest points of these.

Its advantage over K-Means is, that it can also recognize unique shaped clusters. However, it is much slower and requires multiple times more memory (it required 5GB of RAM under more than 30 seconds, for the same dataset Mini-Batch K-means used 400MB under 0.9 seconds), while it cannot even be partially fitted, like K-Means.

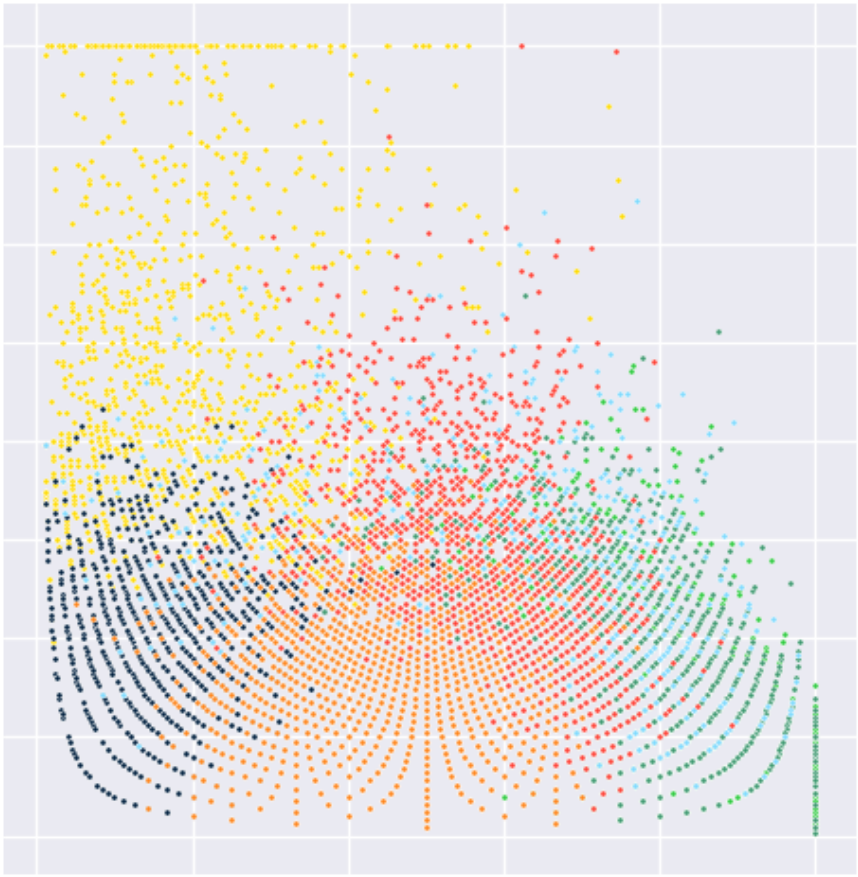

Figure 4.7: Applying Agglomerative Clustering to web sessions

However, a more important advantage is the ability to use a custom distance function, therefore it is possible to handle mixed data (categorical and numerical) with the Gower distance [10]. It can handle mixed numerical, logical and categorical data also. However, it is not suitable for big data, as it manages an n^2 matrix - for just one week of data it consumed 13GB of RAM.

4.2.3.5 K-Prototypes

Zhexue Huang has proposed an extension of K-Means which handles categorical data (K-Modes), and one, which combines the two: K-Prototypes [13]. The logic behind it is simple:

it uses Euclidean distance for numerical columns, while Manhattan distance for categorical data (number of whole steps to take). Table 4.1 shows the distance of point X from the $(0, 0)$ point calculated with both the Euclidean and Manhattan distance.

X_1	X_2	**Euclidean**	**Manhattan**
0	0	0	0
1	0	1	1
1	1	1.4142	2

Table 4.1: Difference between Euclidean and Manhattan distance

Going further, table 4.2 shows the method of calculating the difference between two one-hot vectors with a distance of 6.

0	1	1	0	1	0	1	0	0
0	0	0	1	1	1	0	0	1

Table 4.2: Difference between two one-hot vectors

A current drawback is the lack of an official implementation for any well-known Python package. Nico de Vos has implemented one in Python [6], however, while K-Means takes < 1 second, it takes 10-30 minutes due to the high-level implementation. Robin Hes has taken the time to transform Nico's version to Cython (a compiler for writing performant modules for Python in C) [12], which completes processing the same dataset under a minute. After talking with Robin, it became clear that this implementation is also not ready or optimized well yet, therefore it is also far from being production ready (during the writing of the paper, the author has also contributed to Robin's implementation [20] [21] to make it work for the requirements of this thesis), however, this project will rely on this one, with the hope of more future work being done to these (or new K-Prototypes) packages, like the integration of the newest achievements on this field, one of which is an optimization to distance calculations proposed by Byoungwook Kim [16].

4.2.4 Clusters

Based on the research of the different clustering algorithms, this thesis will rely on K-Prototypes with non-linear scaling in the future, including clustering, classification, predic-

tion and validation. The pair plot of clustering one week of data is represented on figure 4.8 (it only shows the continuous features, but the clustering has been run on all features).

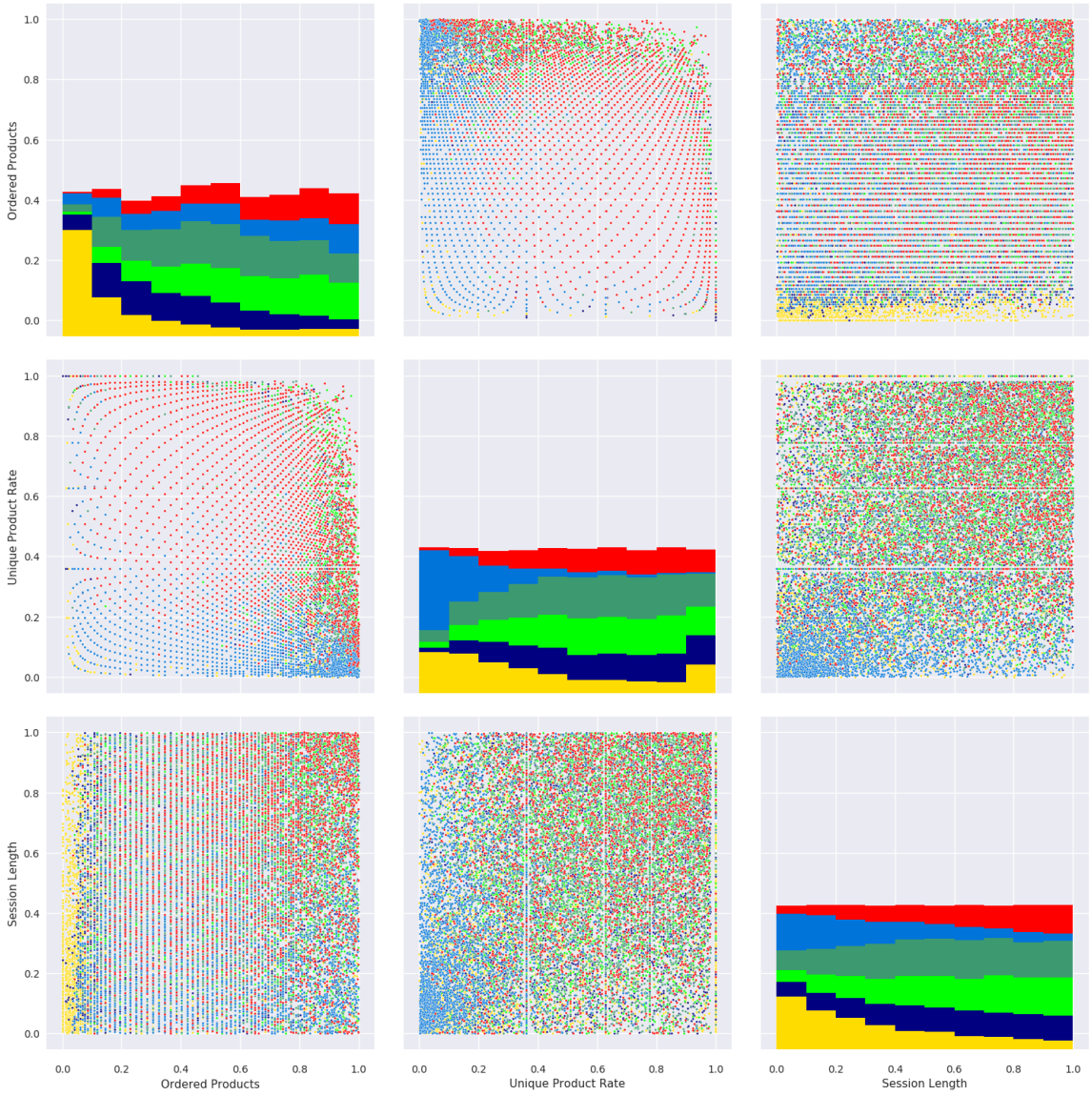

Figure 4.8: Non-linear K-Prototypes clustering

Because of the high dimensionality, most of the clusters are mixed up in the graph. However, some quality of a few clusters can still be recognized:

- The yellow cluster mostly contains samples with a low amount of total ordered products, while red and purple are usually buying more
- Purple has low rate of unique products (so they buy many of the same), and they are usually shopping fast

- Reds are less likely to buy in bulks, and they are also shopping slower

The deeper analytics of the formed clusters will be detailed in Validation (section 5.2).

4.3 Classifier

We need a classifier in order to be able to predict the cluster of a never-seen observation, where prediction means the assigning of samples to clusters or classes, which were unknown in the time of the training.

Generally, it is usually not possible to predict with clustering. An exception is K-Means, which is able to assign new points to existing cluster centroids, simply by putting the point to the cluster, which is the nearest one to it. However, it is not the same as if the algorithm is run again, as predicting this way won't update the centroids - which means, if the algorithm would be re-run again with the new point added, clusters might be formed in totally different ways.

For other algorithms, such assignments are not possible however. For example, agglomerative clustering needs to build the leaves of the tree in the first step, meaning, it needs to know every sample in the time the algorithm runs. There is no way to know which cluster a new observation would have been merged into without running the clustering again, which would lead to different merges of the intermediate clusters, and possibly entirely different end results.

Similarly, a new point added to a density based clustering might merge two clusters, or could allocate a new cluster from invalid observations if them were too far away from each other before but the new one connects them, or if there wasn't enough of them near to each other.

A solution for this problem is to combine clustering with classification, in the way Yaswanth Kumar Alapati [1] describes. The logic is to first cluster the samples, use the cluster identifiers as labels, train a classification model with these data and then predict with this model instead.

4.3.1 Classification

Classification, unlike clustering, is a supervised machine learning method, which means, besides having features, each sample has a label too. This label indicates, which cluster or class the sample belongs to, like the numbers between 0 and 9 for classifying digits, or spam / not-spam if it is used for a spam filter. When predicting the persona of a web session, the label would indicate which persona the sample belongs to. When combined with clustering, the label stores the ID of the assigned cluster.

After having the clusters, any classification method can be applied, which can adapt to the shape of the clusters. For example, since K-Means cannot create clusters of arbitrary shapes, a linear SVC can be effective (see Appendix B). However, it might not work for agglomerative clustering, as it can form clusters in any shape. To be generic enough, in this master thesis a neural network will be built with Keras, which can easily be tweaked to work with any clusters.

4.3.2 Neural networks

Neural networks are based on the assumption of how the human brain works. The sensed data flows through multiple layers of neurons, resulting in one or more outputs. Each network consists of at least 2 layers, an input layer, which represents the perceived data (the feature vector), some hidden layers, and an output layer, as in Figure 4.9.

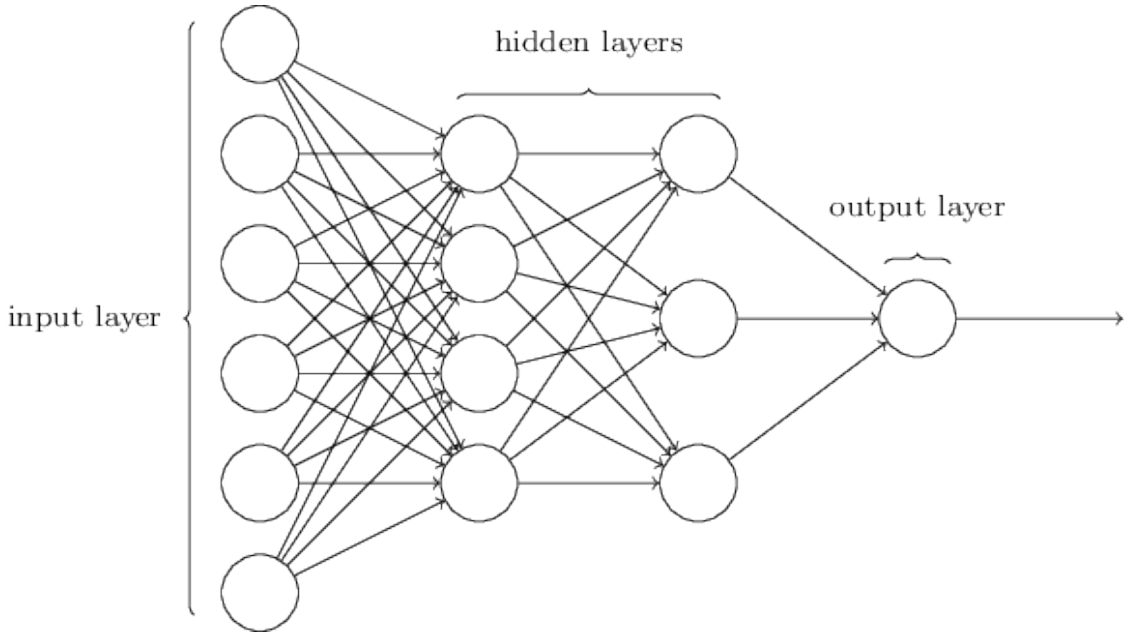

Figure 4.9: Structure of a neural network [23]

Every link has a weight, and each neutron receives the output from all nodes of the previous layer, multiplied by the weight of the link and summed. Then, it calculates a single output with a bias and an activation function, which it passes to the nodes of the next layer.

$$output = activation(\sum(x_i w_i) + bias)$$

Activation functions are responsible for making the network non-linear [28]. If the value is just passed over as-is, it'll mean a linear function between $[-\infty, +\infty]$. As Avinash Sharma V explains,

> [...] No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer [...] That means these two layers (or N layers) can be replaced by a single layer. [...] We just lost the ability of stacking layers this way. No matter how we stack, the whole network is still equivalent to a single layer with linear activation (a combination of linear functions in a linear manner is still another linear function). [...] [28]

However, machine learning usually aims to solve more complex problems, so in most cases an activation function is needed. There are numerous choices, for example `softmax`, which normalizes the input values to the $[0, 1]$ range, adding up to exactly 1 [29]:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (j \in [1, K])$$

Another common function is ReLU, which is defined as follows:

$$f(x) = max(0, x)$$

It can be visualized as easy as the definition is, see Figure 4.10.

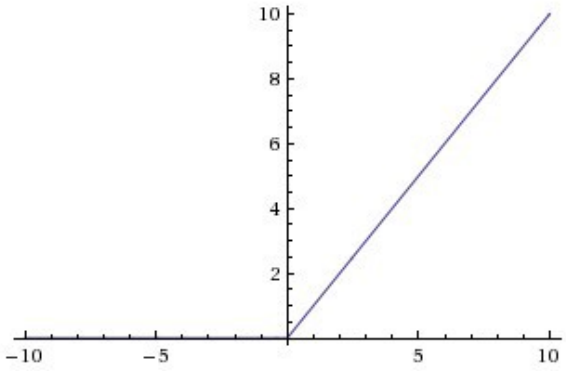

Figure 4.10: ReLU activation function [28]

The task of the training in this scenario is to **optimize the weights** of the network. It does it by feeding the network with each sample one-by-one, calculates the loss with a **loss (cost) function** (like mean squared error, logarithmic error, ...), then updates the weights with an **optimizer function** (like stochastic gradient descent).

4.3.3 Building the model

According to Jeff Heaton's research [11], in most cases a second hidden layer does not lead to more effective predictions, and to have an optimal starting point:

- *The number of hidden neurons should be between the size of the input layer and the size of the output layer.*
- *The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.*
- *The number of hidden neurons should be less than twice the size of the input layer.*

There are uncountable other rules of thumb for selecting an initial number of neurons. In almost every case, it still has to be tweaked to the specific case with the try-and-test methodology.

In this thesis, to comply with the first point of the list above and keep the size of the network low, the mean of the input and output layer size will be used. As a result the Keras model will have an architecture as shown in Figure 4.11.

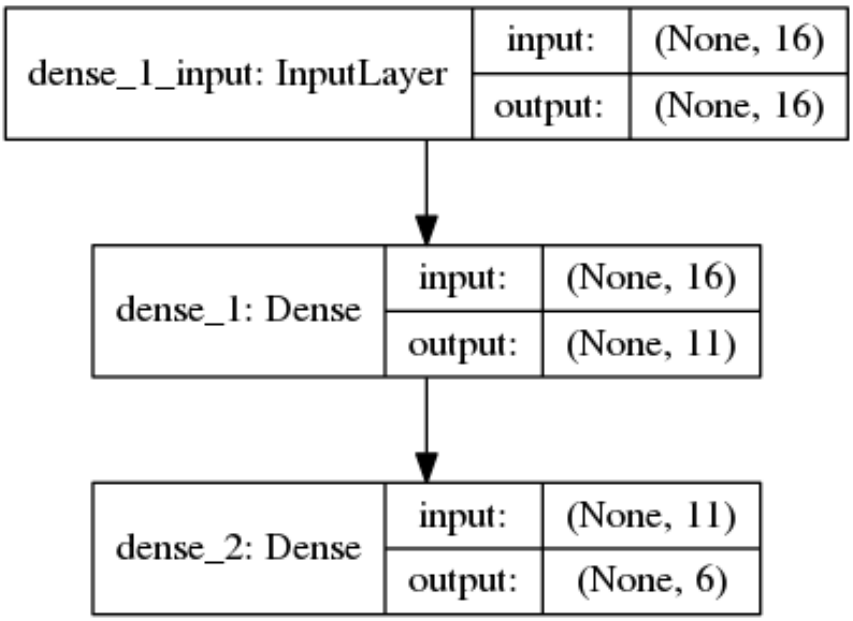

Figure 4.11: Architecture of the used Keras model

It is going to use the Adam optimization function, as suggested by Diederik P. Kingma and Jimmy Lei Ba for sparse data [17].

It yields a 95%+ accuracy for one week of data with just a few epochs, which, taking in consideration that the clustering already has some error, is acceptable.

4.4 Real-time predictions

In this context Kafka will be used as a central message-delivery service. It has been chosen by the company, because it eliminates the need of product-specific API integrations. Therefore, it becomes easy to listen to the events happening on the website, and it is also straightforward to produce a prediction event.

The classifier algorithm subscribes to the *activity* topic, which is fed by services handling customer events, like log-in, order, product visit, etc (as listed in Section 3.2). Afterwards, it will produce *prediction* events for any other service to consume.

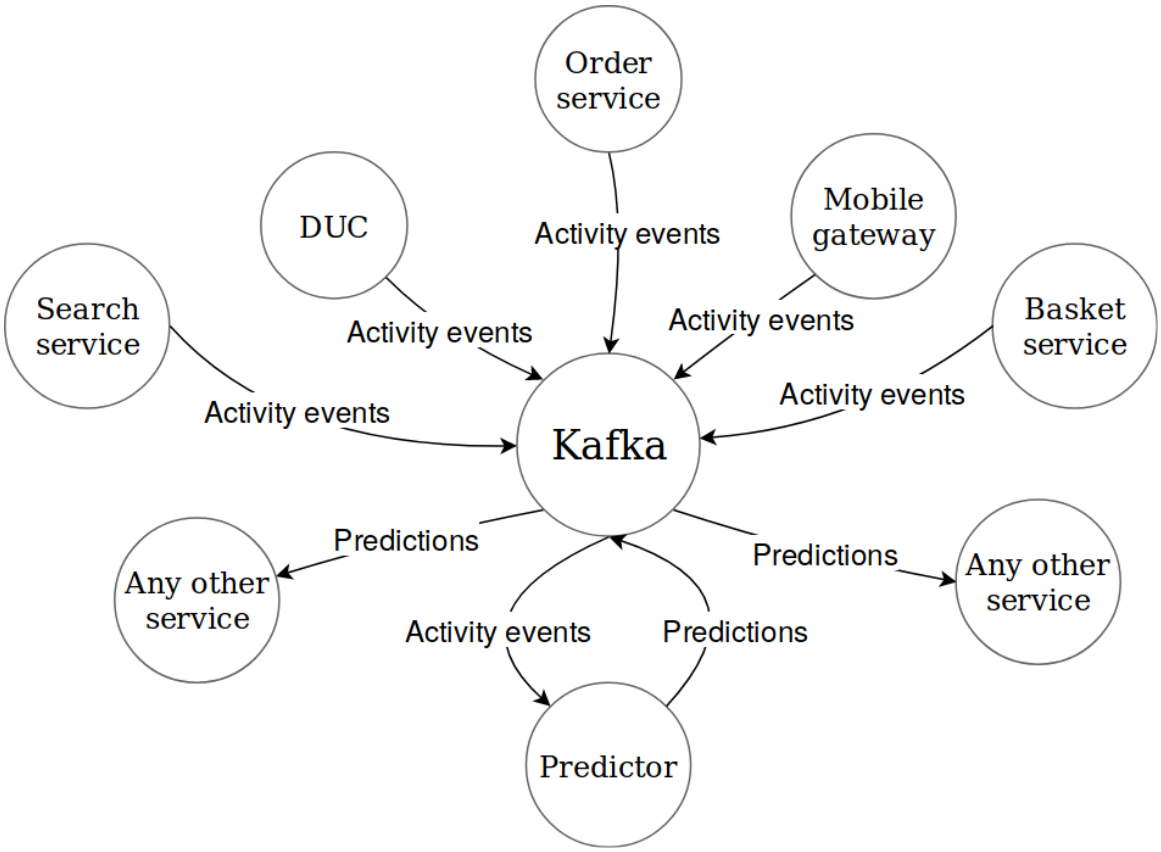

Figure 4.12: Relevant event flow through Kafka

After training, the scaler and the model is dumped to the disk with the intention of loading it later for classification (listing 4.2).

Listing 4.2: Dumping the Keras model

```
joblib.dump(scaler, 'cache/scaler.pkl')
model.save('cache/model.h5')
```

The classifier service will load both, and wait for activity events via Kafka. The construction of the sessions is generally the same, but there are some differences, as here it can't rely on the data of the orders, as these are running sessions.

- If the service receives a *shop_visited* event, it registers a new session locally.
- If it consumes a product visit, basket modifying or coupon redeem event, it alters the session.
- After every 5th visited or basketed product it's going to produce a new prediction. It'll be done by constructing a single feature array from the session, scaling it, predicting with Keras, and sending an event with the session ID and the predicted class to Kafka.
- If a $user\_left$ event is encountered, the session is destroyed.
- Every other event is ignored.

CHAPTER 5

Validation

This chapter describes methods to validate the resulting clusters.

5.1 Validation methods

Two differentiations will be considered in this thesis. In one case, it doesn't matter what type of people are behind the sessions. For example, if it is used to improve UI, it can be tested out that people in cluster 1 prefer the menu bar to be on the top of the page, while people in cluster 2 like it better on the left side.

However, in some situations it might also be important, people with what personalities or distinguishing marks belong to cluster 1 or 2. For example, it might be interesting to know which cluster is the middle-aged people, and which is the young students in case the goal is to suggest them concerts.

5.1.1 A/B testing

If the persona of the sessions does not matter, a simple A/B testing can be used to validate the effectiveness of the clusters and to improve them.

During A/B testing, random 50% of users get the `"A"` version of the page, while others get the `"B"` version. The website tracks the KPIs (Key Performance Indicators), like orders placed, time spent on the website or money spent, and if enough data is collected, the results of version `"A"` and `"B"` are compared, and the one with better KPIs will be selected as the more effective one.

In case the persona or some attributes of the person behind the session matters, validating and recognizing these is a more complex task to do, these are explained below.

5.1.2 Researching customer behavior

Interviewing and contacting real customers can be an expensive and time-consuming task. However, many companies have already done their own research for UI/UX polishing, training their employees or targeting purposes. The explored customer personas can be good starting points for cluster identification.

As it was detailed in Section 4.2.2.1, REWE has already conducted a market research to identify its 6 most common personas. There are numerous qualities for each of these, as detailed in Appendix C.

As it can be seen, the groups are not really distinguishable just by their top product categories. They might help differentiate a few cases, but they are definitely not enough for identifying clusters. Combining with the other insights, one can see a clearer overall profile, but people still behave very similarly, so it'd be quite hard to differentiate these sessions manually.

5.1.3 User profiles

The described method of clustering sessions does not depend on user data at all - even so, it does not even require the visitor to be registered. On the other hand, in case they are, the profile data can be used to validate and improve the clusters. However, since these data are not reliable as discussed in the paper *Automatic Personalization Based on Web Usage Mining* [19] (as the time passes, it becomes more and more irrelevant, and the user might have provided untrue data accidentally or intentionally), it is important not to rely on these data too heavily.

Staying at the bookstore example introduced in Section 3.1, the user could be asked to input their age and gender. If there are enough responses, the formed clusters can be compared with these data. For example, it might be visible that old men prefer classics and check 3 books per hour, while young women prefer dramas and check 20 books per hour. While some of these could be deducted, it might be shown that 30-35 old men read a surprising amount of some books, which are usually perceived as written for women. As a result, when analyzing the session of an unregistered costumer and recognizing him as a 30 year old man, some of these specific assumed-feminine books could also be shown to him as suggestions.

As a side note, on the other way around, the user profile might be improved with these predictions too. It might be useful if other services depend on user data, if the profiles are public, or to improve the predictions as a feedback. All session prediction for each user could be stored, so at any given point in time the past predictions for a profile can be queried. If in the past arbitrary number of sessions the aggregated values and the profile data differ significantly, the user can be notified to update their profile. This way, there is no need to periodically ask the users to update their profile as for example Google does with the phone numbers, but they can be targeted individually when it is probable that a profile update is relevant.

5.1.4 Deductive assumptions

Probably the least precise, but still somewhat meaningful results might be achieved by common sense. For example, slow browsing is probably not a young person, 50+ products is not usually bought by singles, and buying condoms is not very likely to be a retired person. Exploring these possibilities however will only be based on assumptions, and depends very much on the website and its user base, so must be handled with care.

5.2 Matching actual clusters

The results of the clustering is detailed in Appendix D. Since non-linear scaling has been applied, the exact difference between the numbers is meaningless - far away points might have been put next to each other, and points next to each other might have been expanded. Therefore, only the relative order will be examined, as represented on table 5.1. The rows show the clusters, while the folders are the features, respectively: Ordered Products, Unique Product Rate, Session Length, Alcohol, Coffee, Dairy drink, Soft drink, Foreign, Convenience, Fruit, Vegetable, Organic, Q1AM, Q2AM, Q3AM, Q4AM, Q1PM, Q2PM, Q3PM, Q4PM, Is Weekend and Coupon Used.

C1	2	4	2	1	2	1	6	3	3	3	2	3	5	5	4	1	6	4	3	4	3	2
C2	6	6	6	5	4	5	5	6	6	4	5	5	6	6	2	3	2	2	5	5	5	6
C3	5	5	5	4	6	3	4	1	5	5	4	4	3	4	3	4	3	5	4	2	4	3
C4	4	2	4	3	1	6	2	2	4	6	6	6	2	2	5	5	1	3	2	6	2	5
C5	3	1	1	2	5	4	1	5	1	1	1	1	1	1	6	6	5	1	1	1	1	1
C6	1	3	3	6	3	2	3	4	2	2	3	2	4	3	1	2	4	6	6	3	6	4

Table 5.1: Cluster orders for each feature

If we compare it with the company's researched customer behavior (Section 5.1.2), it can be observed, the resulting clusters do not match the surveyed personas. It can have many reasons, one of which is the unpredictability of clustering. For example, just by running the same program with different random seeds for the clustering algorithm and the scaler, the clusters are already different, as it can be seen on table 5.2. They are not entirely different, but it is still highly recognizable.

C1	2	4	3	1	2	1	6	3	3	3	3	3	5	6	4	3	3	4	3	4	3	2
C2	6	6	6	5	4	5	5	6	6	4	5	5	6	4	2	2	5	3	6	6	5	6
C3	5	5	5	4	6	3	4	2	5	5	4	4	3	3	5	5	2	5	2	2	4	4
C4	4	3	4	3	1	6	2	1	4	6	6	6	2	1	3	4	4	2	5	5	2	5
C5	3	1	1	2	5	4	1	5	1	1	1	1	1	2	6	6	6	1	1	1	1	1
C6	1	2	2	6	3	2	3	4	2	2	2	2	4	5	1	1	1	6	4	3	6	3

Table 5.2: Cluster orders with different random initialization

However, by changing the algorithm of any stage (clustering, metric, scaling, etc), the formed clusters can differ drastically. Table 5.3 shows the results of running the same clustering with linear scaling. Some similarity can still be detected, but now the clusters show totally different personas.

C1	6	2	2	1	1	6	1	3	4	5	5	5	3	2	6	4	4	1	3	3	1	2
C2	5	6	6	4	5	3	6	6	6	2	3	2	6	5	1	1	1	5	6	6	6	6
C3	3	5	5	5	4	4	4	4	2	6	6	6	2	1	3	2	6	3	4	5	4	4
C4	4	3	3	3	6	5	2	5	3	4	2	3	4	3	4	5	5	4	2	1	2	3
C5	2	1	1	2	3	2	5	1	1	1	1	1	1	4	5	6	3	2	1	2	3	1
C6	1	4	4	6	2	1	3	2	5	3	4	4	5	6	2	3	2	6	5	4	5	5

Table 5.3: Cluster orders with linear scaling

Based on the results it is clear that classifying personas unambiguously is not a possibility. However, it doesn't mean that the shaped clusters are useless - they still define users, who act differently, these are just grouped in different ways.

Rewe Digital will be able to decide if they would like to attempt to use the shaped clusters, or if they'd like to match the clusters with the researched personas, in which case section 5.3 will introduce methods to improve the clustering for specific needs.

5.3 Improve the clustering

If in the validation process the clusters do not match, the normalized features might be tweaked. As *Kaufman et al.* describe in their paper:

" [...] it may well be that some variables are intrinsically more important than others in a particular application [...] " [15]

For example, it might have been assumed, that the book category is as important as the year of release, therefore both of these were indicated with 0 or 1. However, maybe this assumption was wrong, and in reality, the category is more important than the year. In this case the year should have a lower value than the category, for example 0.5.

Testing it out is not an easy task, but as many elements in machine-learning, it also works in a trial-and-error basis - the parameters are tweaked until the results get a meaning. In addition, it is always a possibility that there is actually no correlation at all between the features and the tested attributes - in the example above, there is a chance that age or gender does not affect the visited books at all - maybe old ladies read just as much Harry Potter as teenage boys. In this case new features can to be found, different scaling might be applied, or other metrics can be tried out, which would cluster by different distance calculations.

CHAPTER 6

Conclusions

This chapter will describe the achieved goals done by the master thesis following some of the key points developed in the project.

6.1 Achieved Goals

The achieved goals for this project are the following ones:

- A method has been demonstrated to build feature vectors for web sessions.
- The possible web session clustering and predicting methods have been introduced and compared.
- Multiple ways have been shown to decide if the formed clusters fit the problem at hand.
- A real-time classification method has been introduced.

6.2 Future works

While this master thesis provides a detailed concept, there are numerous possibilities for improvement:

- Researching and implementing a highly-efficient K-Prototypes Python package.
- Analyzing further metrics to handle this type of data better.
- Code optimizations. Currently in most part of the source, transparency was preferred over efficiency. However, to make it function well as a service, array and dictionary handling must be reviewed and improved.
- Instead of storing sessions in-memory for real-time classification, Redis (or similar) storage should be used to make the service scalable.
- Improve the clustering to maximize the cover of the researched personas.
- Gain more product features from the company.
- Run cluster tests with more data.

6.3 Conclusion

As the results show, clustering and predicting personas from web sessions is not an easy task. Not only the choice of estimators, scalers, metrics and the proper preparing of data introduce challenges, but also the validation of the formed clusters, especially to assign some meaning (personas) to them.

Since clustering has unpredictable factors, the results can be entirely different just by changing a random seed or adding / removing a single feature. It leads to much trial-and-error work to do to reach an expected outcome.

However, this thesis has introduced some methods to deal with most of the challenges emerging from classifying web sessions, therefore it is definitely a possibility to reliably figure out the persona of a web session, but it is also clear, that more research on this topic has yet to be done in order to shorten the workflow and maximize the efficiency.

APPENDIX A

K-Means clustering results

Ordered Products

1. 0.41220169768893
2. 0.377339849830232
3. 0.422120199701508
4. 0.424762871247506
5. 0.400445043985565
6. 0.402923690315524

Unique Product Rate

1. 0.331687298280826
2. 0.661633086084209
3. 0.531036434260861
4. 0.497780610299476
5. 0.687643897991953
6. 0.640476287331456

Session Length

1. 0.22394512999712
2. 0.423070932447671
3. 0.375574750315994
4. 0.352021079118089
5. 0.460282649964432
6. 0.439923581524312

Q1AM

1. 0.060113728675873
2. 0.000695603784085
3. 0.000313643491898
4. 0.000416796915703
5. 0.10230708035004
6. 0.001217285453439

Q2AM

1. 0.034389385323585
2. 0.000417362270451
3. 0
4. 0.000104199228926
5. 0.039140811455847
6. 0.000608642726719

Q3AM

1. 0.539940427836448
2. 0.001112966054535
3. 0.00083638264506
4. 0.001354589976034
5. 0.106284805091488
6. 0.000760803408399

Q4AM

1. 0.015434606011373
2. 0.000417362270451
3. 0.000209095661265
4. 0.996874023132229
5. 0.001750198886237
6. 0.000608642726719

Q1PM

1. 0.005957216355267
2. 0
3. 0.997281756403555
4. 0.000312597686777
5. 0.000477326968974
6. 0.00015216068168

Q2PM

1. 0.329813160032494
2. 0.000278241513634
3. 0.000209095661265
4. 0.000312597686777
5. 0.002068416865553
6. 0.994065733414486

Q3PM

1. 0.00622799891687
2. 0.996939343350028
3. 0.000627286983795
4. 0.000312597686777
5. 0.000477326968974
6. 0.001521606816799

Q4PM

1. 0.008123476848091
2. 0.000139120756817
3. 0.000522739153163
4. 0.000312597686777
5. 0.747494033412887
6. 0.001065124771759

Is Weekend

1. 0.074194421879231
2. 0.292153589315526
3. 0.225718766335598
4. 0.182765447535688
5. 0.279554494828958
6. 0.334753499695679

Coupon Used

1. 0.106959111833198
2. 0.254451864218142
3. 0.228959749085206
4. 0.197665937272064
5. 0.286077963404932
6. 0.232501521606817

Soft drink

1. 0.292655344237945
2. 0.460853845885407
3. 0.418483742286606
4. 0.422036339306108
5. 0.495650011750208
6. 0.486297201848966

Vegetable

1. 0.156891267609931
2. 0.381268042753606
3. 0.316718056745255
4. 0.301651558442409
5. 0.419311952856889
6. 0.377712155295464

Alcohol

1. 0.502117861933856
2. 0.38763975371665
3. 0.387519137043203
4. 0.376525642567984
5. 0.342966866193135
6. 0.342047239005439

Foreign

1. 0.08473142676222
2. 0.147285459805997
3. 0.133087538077022
4. 0.125148295831579
5. 0.157112695246951
6. 0.141561986835435

Organic

1. 0.169805274666195
2. 0.30737705726242
3. 0.284797938507316
4. 0.273265839051547
5. 0.350632594117178
6. 0.296849091664205

Coffee

1. 0.280127009394224
2. 0.223998613807959
3. 0.26422694882608
4. 0.271938202177727
5. 0.211352978256062
6. 0.240639872118478

Convenience

1. 0.179022918840598
2. 0.41872808383611
3. 0.340850055238487
4. 0.321544357616767
5. 0.445751979121358
6. 0.417396007008222

Price

1. 0.507103985581545
2. 0.477963352623955
3. 0.472820745184027
4. 0.483695348047153
5. 0.463650529159802
6. 0.458168537041377

Dairy drink

1. 0.469442591932425
2. 0.324298609912423
3. 0.36182130243553
4. 0.371134589238075
5. 0.324340836842278
6. 0.298896724241041

Fruit

1. 0.211448834806247
2. 0.300870144421624
3. 0.296922921155452
4. 0.306198568629132
5. 0.331723115702945
6. 0.322847803000286

APPENDIX B

SVC classification on K-Means clusters

Clustering 2 weeks of data, then classifying it with Scikit-Learn's Linear SVC implementation (without tweaking any parameters), a $> 98\%$ accuracy has been achieved, which shows, Support Vector Machines can be applied to K-Means clusters successfully. Figure B.1 shows the differences between the clusters and the classified samples.

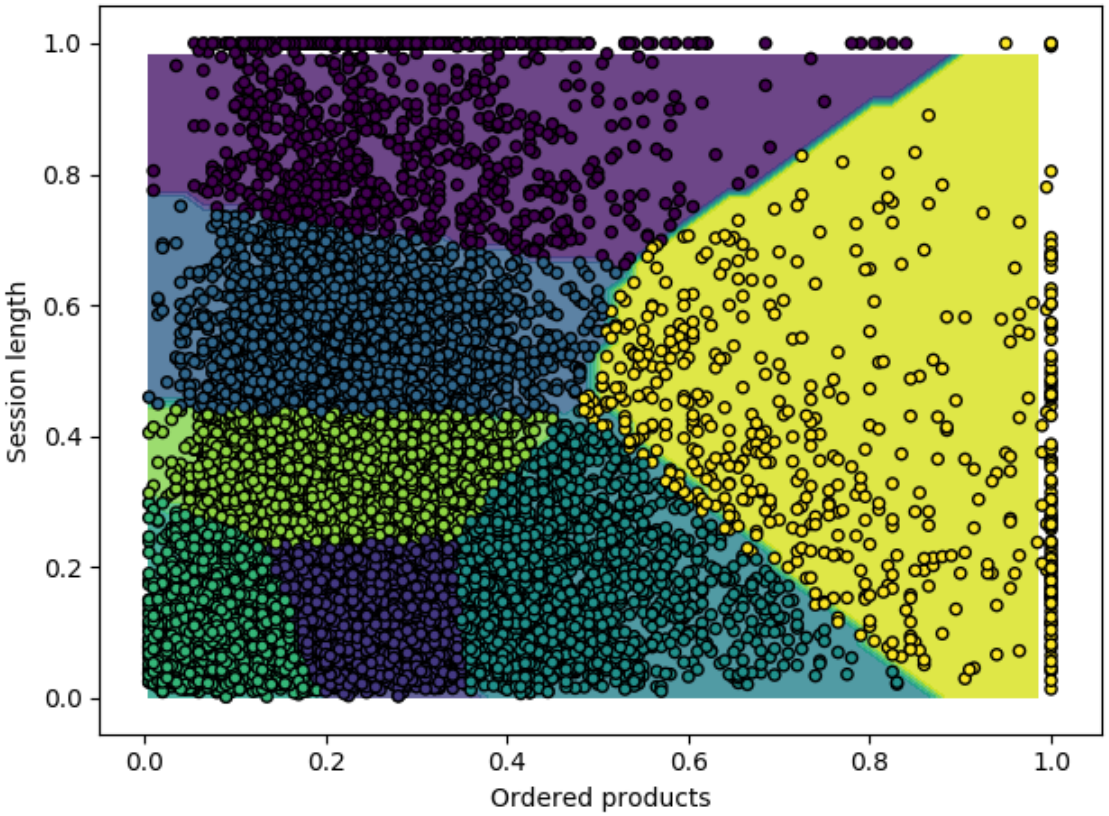

Figure B.1: Applying SVC to K-Means clusters

APPENDIX C

REWE Personas

C.1 Couple without children

C.1.1 Top products

- Non-alcoholic drinks
- Vegetables
- Dairy-products
- Fruits
- Cheese

C.1.2 Eating and shopping

- Eating healthy and consciously
- Cooking fresh, regional meals
- One of them is vegetarian
- They are willing to try out new recipes and flavors

Time-saving > Light bag > Comfort

C.2 Working single

C.2.1 Top products

- Non-alcoholic drinks
- Beer
- Dairy-products
- Precooked-meals
- Fruits

C.2.2 Eating and shopping

- Rarely cooks
- Eating outside

- Heating precooked-meals if they have to
- Healthy is not important
- Mobile-checkout
- Small orders
- Fast shopping

Light bag > Time-saving > Health

C.3 Older family with a teenager

C.3.1 Top products

- Dairy-products
- Non-alcoholic drinks
- Vegetables
- Cheese
- Fruits

C.3.2 Eating and shopping

- Cooking usually, eating with family
- Healthy and varying meals
- Prefers fresh over convenient food
- Browsing nutrition advices
- Offers, coupons and pay-back possibility
- Prefers pick-up over delivery

Time-saving > Light bag > No driving

C.4 Old couple

C.4.1 Top products

- Dairy-products
- Non-alcoholic drinks
- Vegetables
- Precooked-meals
- Fruits

C.4.2 Eating and shopping

- If the husband is alone, he prefers precooked-meals instead of cooking

Health > Light bag > Curiosity

C.5 Student

C.5.1 Top products

- Non-alcoholic drinks
- Dairy-products
- Precooked-meals
- Fruits
- Vegetables

C.5.2 Eating and shopping

- Likes to cook
- Sometimes tries exotic and extraordinary meals
- Eats pre-cooked meals when busy
- Price and coupons are important
- Free shipping without minimal order

Deals > No driving

C.6 Family with two high school children

C.6.1 Top products

- Non-alcoholic drinks
- Dairy-products
- Vegetables
- Fruits
- Cheese

C.6.2 Eating and shopping

- Structured shopping
- Shops more times each week
- Offers are important

Time-saving > Light bag > Comfort

APPENDIX D

Non-linear K-Prototypes clustering results

Ordered Products	Coffee	Convenience
1. 0.381834413014405	1. 0.01627544744965	1. 0.48471541701338
2. 0.640705861870621	2. 0.261041485060476	2. 0.580428164965294
3. 0.598431435250295	3. 0.867779364913233	3. 0.535449012868812
4. 0.583637332221487	4. 0	4. 0.513096071791226
5. 0.456298835412774	5. 0.36388581529682	5. 0.085106448296865
6. 0.339158690171562	6. 0.082529180246358	6. 0.402579513529044
Unique Product Rate	**Dairy drink**	**Fruit**
1. 0.545101747614599	1. 0.152724791192593	1. 0.2550967349316
2. 0.596430549593874	2. 0.491827887647786	2. 0.499508516883031
3. 0.558504652793942	3. 0.426671067125457	3. 0.547387443196724
4. 0.542711144445694	4. 0.65244221100045	4. 0.662467736592715
5. 0.252203147583715	5. 0.479087185430346	5. 0.095675515063917
6. 0.543084618707222	6. 0.199823574733205	6. 0.227125141774308
Session Length	**Soft drink**	**Vegetable**
1. 0.48567020232528	1. 0.862872217195376	1. 0.286598847177352
2. 0.599724228243364	2. 0.484038463268144	2. 0.570892342125361
3. 0.583113726513274	3. 0.390122941207425	3. 0.561138006621575
4. 0.546364244410179	4. 0.368229361948861	4. 0.684034237225448
5. 0.327738917517087	5. 0.236706805304899	5. 0.02247999616413
6. 0.487092550719621	6. 0.376164947555665	6. 0.296404318891402
Alcohol	**Foreign**	**Organic**
1. 0	1. 0.02988208873496	1. 0.299020316551284
2. 0.200775594835896	2. 0.919299242918542	2. 0.537478893145571
3. 0.19248920136323	3. 0	3. 0.507870325929648
4. 0.095966001069076	4. 0.000145026601668	4. 0.628968767251711
5. 0.017087970093163	5. 0.065317802316964	5. 0.139366811574485
6. 0.905532710634994	6. 0.036061065701525	6. 0.294295981626977

Q1AM

1. 0.02398022249691
2. 0.030531845042679
3. 0.019902020820576
4. 0.01980198019802
5. 0.009810631987223
6. 0.02345645726611

Q1PM

1. 0.259085290482077
2. 0.220288903479974
3. 0.224433557868953
4. 0.197045934101607
5. 0.24663472507415
6. 0.229981126988407

Is Weekend

1. 0.266254635352287
2. 0.270518713066316
3. 0.269442743417024
4. 0.245739328031164
5. 0.140086698608259
6. 0.282825559449986

Q2AM

1. 0.009394313967862
2. 0.010833880499015
3. 0.009185548071035
4. 0.006492452523941
5. 0.006160164271047
6. 0.008897276894042

Q2PM

1. 0.186155747836836
2. 0.170387393302692
3. 0.187078995713411
4. 0.177243953903587
5. 0.16974674880219
6. 0.193583176058237

Coupon Used

1. 0.210630407911001
2. 0.27478660538411
3. 0.24464176362523
4. 0.250284044797922
5. 0.123431439653206
6. 0.249932596387166

Q3AM

1. 0.059332509270705
2. 0.052856204858831
3. 0.056031843233313
4. 0.060055185846454
5. 0.08373260323979
6. 0.0498786734969

Q3PM

1. 0.176266996291718
2. 0.196651346027577
3. 0.183404776484997
4. 0.172212303197533
5. 0.086470454026922
6. 0.199245079536263

Sample Count

1. 4045
2. 3046
3. 3266
4. 6161
5. 4383
6. 3709

Q4AM

1. 0.164153275648949
2. 0.185489166119501
3. 0.199326393141457
4. 0.233079045609479
5. 0.34839151266256
6. 0.174170935562146

Q4PM

1. 0.121631644004944
2. 0.132961260669731
3. 0.120636864666258
4. 0.13406914461938
5. 0.049053159936117
6. 0.120787274197897

Bibliography

[1] Yaswanth Kumar Alapati. Combining clustering with classification: A technique to improve classification accuracy.

[2] Apache. Kafka.

[3] The computer language benchmarks game.

[4] Google Brain. Tensorflow: A system for large-scale machine learning, November 2016.

[5] François Chollet. Keras: The python deep learning library, 2018.

[6] Nico de Vos. K-modes implementation in Python, 2017.

[7] Franck Dernoncourt, August 2015.

[8] Fabian Pedregosa et al. Scikit-learn: Machine learning in python, October 2011.

[9] Google. Bigquery, 2018.

[10] J. C. Gower. A general coefficient of similarity and some of its properties, April 2007.

[11] Jeff Heaton. The number of hidden layers.

[12] Robin Hes. K-modes implementation in Python with Cython, 2017.

[13] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values, 1999.

[14] Inna Kaler. So you have some clusters, now what?, November 2017.

[15] Leonard Kaufman and Peter J. Rousseeuw. Finding groups in data: An introduction to cluster analysis, March 1990.

[16] Byoungwook Kim. A fast k-prototypes algorithm using partial distance computation, April 2017.

[17] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization, January 2017.

[18] JP Mangalindan. Amazon's recommendation secret, July 2012.

[19] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining, August 2000.

[20] Kristóf Morva. Handle any distance in Robin Hes' k-prototypes implementation, May 2018.

[21] Kristóf Morva. Handle tuples correctly in Robin Hes' k-prototypes implementation, May 2018.

[22] Olfa Nasraoui, Hichem Frigui, Raghu Krishnapuram, and Anupam Joshi. Extracting web user profiles using relational competitive fuzzy clustering, 2000.

[23] Michael Nielsen. Neural networks and deep learning, December 2017.

[24] David Robinson. The incredible growth of python, September 2017.

[25] Peter J. Rousseeouw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, November 1986.

[26] D. Sculley. Web-scale k-means clustering.

[27] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. Introduction to data mining.

[28] Avinash Sharma V. Understanding activation functions in neural networks, March 2017.

[29] Wikipedia. Softmax function.