University of Padova

Department of Physics and Astronomy "Galileo Galilei"

Master Thesis in Physics of Data

Machine Learning Approaches to Enhance Revenue Capture in E-commerce Platforms

Supervisor
Prof. Marco Baiesi
University of Padova

Co-supervisor
Mattia Zoccarato

Master Candidate
Lorenzo Ausilio

Student ID
2046831

Academic Year
2023-2024

Abstract

This thesis explores techniques on how to increase revenue in e-commerce platforms through machine learning approaches, a critical endeavor in the face of rapidly evolving online retail landscapes.

An algorithm utilizing Markov Chains was implemented to assign scores to individual web pages, reflecting their importance in terms of potential revenue generation.

In a parallel analysis, the thesis delves into clustering techniques for mixed data types, utilizing k-prototype clustering and HDBSCAN coupled with UMAP for dimensionality reduction, to identify nuanced user clusters and compute the potential missing revenue from the malfunction of the e-commerce website. The selection of variables for this clustering was informed by a Random Forest analysis, ensuring an empirical approach to uncovering latent revenue opportunities. The results reveal significant insights into user behavior and platform efficiency, guiding e-commerce platforms in tailoring user experiences and maximizing revenue capture.

The discussion of these findings highlights the transformative potential of integrating machine learning methodologies into e-commerce strategies. By computing critical engagement metrics using Random Forest and unveiling hidden user clusters, this research offers a blueprint for e-commerce platforms to sustain growth and competitiveness in a digital-first economy.

Contents

Listing of figures

Listing of tables

Listing of acronyms

ML ..... Machine Learning

DL ..... Deep Learning

SVM ..... Support Vector Machine

PCA ..... Principal Component Analysis

KNN ..... K-Nearest Neighbours

ANN ..... Artificial Neural Network

CNN ..... Convolutional Neural Network

VGG ..... Visual Geometry Group Network

TCN ..... Temporal Convolutional Network

RNN ..... Recurrent Neural Network

LSTM ..... Long Short-Term Memory Network

GRU ..... Gated Recurrent Unit

NB ..... Naïve Bayes

URL ..... Uniform Resource Locator

MC ..... Markov Chains

UMAP ..... Uniform Manifold Approximation and Projection

HDBSCAN .... Hierarchical Density-Based Spatial Clustering of Applications with Noise

RF ..... Random Forest

t-SNE ..... t-Distributed Stochastic Neighbour Embedding

DBCV ..... Density Based Clustering Validation

1 Introduction

E-commerce, which stands for electronic commerce, fundamentally transforms the way goods and services are exchanged via electronic media and the internet[1]. This modern commerce paradigm leverages information technology, including Electronic Data Interchange (EDI), to optimize business operations. E-commerce platforms typically operate through a vendor's website, offering products or services directly to consumers via a digital interface. These platforms utilize digital shopping carts and facilitate transactions through various payment methods such as credit cards, debit cards, or electronic funds transfers. Beyond a mere transactional interface, e-commerce encompasses a broader spectrum of electronic communications and digital information management technologies. These technologies are essential in forming, transforming, and redefining relationships not only between businesses but also between companies and individuals, expanding into diverse models like Business-to-Business (B2B), Business-to-Consumer (B2C), Business-to-Government (B2G), Customer-to-Customer (C2C), and Mobile Commerce (M-commerce). This evolution marks a significant shift in the commerce landscape, driven by the advent of internet banking, marketing innovations, and self-service technologies.

The following sections are based on the recent comprehensive insights of the review 'A Brief Survey of Machine Learning and Deep Learning Techniques for E-Commerce Research'[2], in which the authors highlight prevalent ML and DL techniques in e-commerce, along with key research topics, challenges and trends which were obtained by analyzing 158 papers from the years 2018 until 2023.

1.1 Techniques & Research Topics

The most utilized machine learning techniques in the e-commerce space are Support vector machines (SVM)[3], Decision Trees[4], Random Forests (RF)[5, 6], Naïve Bayes[7], Logistic Regression[8], Principal component analysis (PCA)[9], Matrix factorization algorithms[10, 11] and K-nearest neighbors (KNN)[12].

Deep Learning techniques including Artificial Neural Networks (ANN)[13], Convolutional Neural Networks (CNN)[14], Visual Geometry Group networks (VGG)[15], Temporal Convolutional Network (TCN)[16], Recurrent Neural Networks (RNN)[17], Long Short-Term Memory (LSTM)[18], Gated Recurrent Units (GRU)[19] were found to be some of the most prevalent in this field.

The integration of ML and DL within the e-commerce sector focuses on eight important research areas :

- **Sentiment Analysis** is the process of determining the sentiment or emotional tone (e.g., positive, negative, or neutral) behind a piece of text, such as a review, comment, or social media post. ML and DL techniques have been widely applied in sentiment analysis to automate the process of sentiment classification[20]. In particular, DL has shown to be more efficient compared to traditional ML in this case, as can be seen in this study[21] where the task was to analyze product ratings and text reviews; RF was outperformed by a combination of RNN and LSTM.

- **Recommendation System** is a a technology that suggests personalized product recommendations to users based on their preferences, past behavior, and other relevant data. Recommendation systems have the aim of enhancing the user experience, increasing sales and improving customer satisfaction. In e-commerce, data regarding browsing history, purchase patterns and demographic information can be leveraged in order to create personalized recommendations. Yet again, DL models have shown to give the best results even in this case, particularly hybrid DL models have exhibited a lot of potential on personalized size and fit recommendations in fashion e-commerce[22].

- **Fake Review Detection** involves identifying and distinguishing fraudulent, or fake, reviews from genuine ones. These reviews are problematic because they can mislead consumers and impact their purchasing decisions. Several ML methods including SVM, KNN, Decision Trees, Logistic Regression and NB have shown prominent results[23, 24], each model having its own advantages and disadvantages in different environments. Researchers have shown that DL models such as CNN and LSTM generally outperform ML models like NB,SVM and KNN when it comes to detecting fake reviews[25, 26]. These findings support the notion that DL methods, in particular CNN and LSTM, outperform more traditional ML methods in detecting fake and spam reviews.

- **Fraud Detection** refers to the process of identifying and preventing fraudulent activities within online retail transactions. In this case however, DL methods did not consistently outperform ML methods, because of their need of large quantities of data to fully learn features. In a comparative analysis of Credit Card Fraud Detection[27], a RF model showed better accuracy than a deep neural network. However, in other instances LSTM and GRU models outperformed NB and other ML models[28], highlighting the fact that both ML and DL methods currently coexist when it comes to fraud detection.

- **Customer Churn Prediction** refers to the process of forecasting which customers are likely to stop engaging or making a purchase on a website. In this case, there is a great class imbalance since the majority of customers do not buy or keep engaging, thus increasing misclassification errors[29]. To address this problem, various techniques such as re-sampling, class weighing, data augmentation, and ensemble methods have been used to mitigate the effect of imbalanced class[30]. Feature engineering and choosing the right model architectures are especially important, as often is the case that multiple types of data, such as numerical or categorical, are present when solving this type of problem[31].

- **Customer Purchase Behavior Prediction** is used to anticipate the future purchasing decisions of individual customers or customer segments in order to design customer retention strategies to increase sales and increase customer satisfaction. The analyzed studies focused on predicting customer behavior to optimize marketing strategies with the use of RNNs[32] and also predicting time of next purchase using deep neural networks, outperforming conventional methods such as SVM and RF[33].

- **Prediction of Sales** in e-commerce empowers companies to stay competitive, anticipate market changes, and provide a good shopping experience to customers. In this scenario, DL models including CNNs have been shown to outperform traditional ML methods in terms of sales forecasting accuracy[34]. Hybrid DL models have also shown exceptional results in predicting sales with sentiment analysis[35], further solidifying the use of DL in this context compared to traditional ML methods.

- **Product Classification and Image Recognition** in the realm of e-commerce is a significant challenge due to the constant arisal of new products and the malleable nature of categories these products can be classified into. Several approaches have been proposed to solve this problem, including neural networks combining text and images to improve accuracy on large-scale product data sets[36], RF models utilizing product titles, which achieved superior accuracy compared to SVM and CNN[37], and also transfer learning, CNN-RNN and SVM-CNN to accurately categorize fashion products[38]. Advanced ML and DL is thus vital in enhancing product classification, image recognition, and categorization in the e- commerce industry.

1.2 Main Challenges & Trends

There are essentially six main challenges and trends that ML and DL are currently facing in the e-commerce business:

- **Imbalanced data** is prevalent in e-commerce fields such as fraud detection, fake review detection, customer churn prediction, and re-purchase behavior classification[39, 40, 41, 42] where one class heavily outweighs the others. This leads to biased models with poor performance on the minority class. As already mentioned before, we can try to solve this issue by using re-sampling techniques, weighted training and transfer learning in the case of Neural Networks.

- **Achieving robust generalization** and preventing over-fitting is another big challenge for ML and DL in e-commerce[43]. Ensembling techniques such as bagging and boosting combine multiple models to improve reduce over-fitting. More accurate and robust models can also be created with the help of regularization techniques, data augmentation, dropout, cross-validation, transfer learning and early stopping.

- **Multi-modal learning** is a type of DL which uses a combination of various modalities of data. An example of multi-modal data is data that combines text with imaging data. This type of data poses significant challenges for predictions in e-commerce[44] since integrating data from diverse sources makes the feature extraction a complex, resource-intensive and time-consuming process. Moreover the ways to fuse the different data together becomes challenging. Despite this, multi-modal learning has been utilized in e-commerce in tasks such as product classification[44] and sentiment analysis[45].

- **Model interpretability** is key in e-commerce, since we want to be able to understand why a model made specific recommendations or classifications for building trust with users. Currently, interpretability techniques such as feature visualization[46], attention mechanisms[47] and gradient-based methods[48] are being explored to better show the inner workings of ML and DL methods.

- **Personalization** is a well-known research area in e-commerce. It aims at enhancing the user experience using AI-powered customer services like chatbots and virtual assistants to provide support and also predict customer needs and give tailored assistance. For this task reinforcement learning[49] has been shown a good candidate, as well as transfer learning[50] to refine pre-trained models, enhancing their performance in specialized e-commerce settings.

- **Chatbots and virtual assistants** are emerging as a new trend in e-commerce[51, 52]. Conversational AI and Natural Language Processing (NLP) are at the base of these virtual communicators, enabling them to deliver personalized customer support.

1.3 Thesis Structure

The thesis is focused on two main chapters. The first chapter is about assigning scores to web pages based on their importance in terms of generating revenue. The score is calculated thanks to Markov chains. The second chapter goes into clustering methodologies for mixed-type data, using K-prototypes, an extension of K-means and K-median, and HDBSCAN combined with UMAP. Finally a comparison is made between the two clustering methods.

We could classify the work of this thesis under the category of website optimization and faulty behaviour detection, for what concerns the page value computed. Clustering of users would rather belong to the category of customer purchase behavior prediction. Some of the challenges and trends encountered involve imbalanced data, especially in the clustering analysis of chapter 3, and also model interpretability in chapter 2 with Markov chains.

All of the data, including user behavior on the website and characteristics such as the duration of visits, is taken from an Italian e-commerce website that will remain anonymous.

2

Markov Chains And Page Values

In the rapidly evolving digital marketplace, understanding the value of web pages within an e-commerce framework is more crucial than ever. The page value reflects the importance of a given web page in terms of how much revenue can be attributed to that web page alone, without considering all the others. The higher the page value, the more revenue that web page brought.

The value attributed to each web page on an e-commerce site is not just a reflection of direct sales but also an indicator of the page's role in the customer's journey towards making a purchase. High-value pages are often those that successfully capture user interest, provide valuable information, and guide visitors through the sales funnel effectively. By analyzing these metrics, businesses can pinpoint which aspects of their digital presence are driving success and which areas may require optimization.

To this end, an algorithm which gives a score to web pages based on their contribution to the revenue which was generated is needed.

This chapter uses an algorithm leveraging the power of Markov Chains (MC) to assign values to pages from 0 to 100, the former meaning the page is of no importance while the latter that the page is of utmost importance. This algorithm can be applied to specific pages, i.e. specific URLs, or more broadly to *page types*, which is what we will explore in this chapter. There are six different page types which we will be assigning scores to : *product, category, checkout success, checkout, home* and *other*. The page types are pretty self explanatory but a brief explanation of each can be formulated as such: The *home* page is the entrance page to any website; it is usually the first page users visit when they enter an e-commerce website. *Category* and *Product*

pages logically follow the home page and contain information regarding categories products fall into. As an example a *Music* category may contain products related to musical instruments or gadgets such as earphones. In the final stages of users navigating a website, they might find themselves in the *checkout* and *checkout success* page types, which means they are interested in buying something from that website. The *checkout success* means that a user bought from the e-commerce website and it is only included as a reference point, which ideally has a page value of 100. Meanwhile the *other* page type is any page type which does not belong to any of the previously mentioned page types.

2.1 Multi-touch attribution models

The algorithm we implement in this chapter is based on the paper *Mapping the Customer Journey: A Graph-Based Framework for Online Attribution Modeling*[53] and the available python library *ChannelAttribution*[54].

The python library implements what is know as *multi-touch attribution*, which is actually a method of marketing measurement that considers all touch points on the customer journey and allocates a certain amount of credit to each channel. In our specific case a channel corresponds to an individual page type, hence we will be using these terms interchangeably. This allows marketers to understand the value that each touch point contributes to driving a conversion.

For instance, suppose a consumer is contemplating the purchase of a new smartphone. After conducting some research, they come across ads. Initially, they encounter a display ad, which they overlook. Subsequently, they notice a native ad on a specific app capturing their attention and redirecting them to a website. Finally, they receive a promotional offer via email, complete with a discount code that motivates them to make the purchase.

Each of these ads represents a touch point in the buyer's journey. With multi-touch attribution, marketers can examine the impact of the native ad and the email campaign, attributing the sale to these specific efforts. Meanwhile, they can recognize that the display ad was ineffective and make adjustments accordingly.

There are various multi-touch attribution models available to marketers, analyzing user-level data such as clicks and impressions to understand the influence of these events on the ultimate goal. These models vary in their evaluation of ad effectiveness, offering marketers diverse perspectives to shape their strategies.

For our purpose of giving value to pages, the touch points involved in the customer journey

are not defined by ads that the customer went through, as the original paper and library suggest, but rather the web pages the user visited before making (or not making) a purchase on the website.

There are essentially two families of attribution models: heuristic models and probabilistic models.

2.1.1 Heuristic Attribution Models

Heuristic models use set rules to assign credit across various touch points. These rules typically stem from standard assumptions or industry norms. The main advantages of heuristic models include their simplicity and transparency. Being relatively straightforward, they allocate credit using clear-cut rules, which makes them particularly user-friendly for marketers who may lack deep statistical knowledge. Furthermore, the transparency of these models ensures that the rationale behind credit allocation is clear and understandable. Examples of heuristic models include:

- **First-touch Attribution** assigns all the credit to the initial touch point in a customer's journey, underscoring the importance of the first interaction in leading to a conversion.
- **Last-touch Attribution** gives all the credit to the customer journey's final touch point, underscoring the critical role of the last interaction in the conversion process.
- **Linear Attribution** allocates credit evenly across all touch points, ensuring each interaction is equally recognized for its contribution.

2.1.2 Probabilistic Attribution Models

Probabilistic models leverage statistical methods and algorithms to evaluate data and calculate the likelihood of each touch point contributing to a conversion. These models are characterized by their complexity and customization potential. Unlike simpler heuristic models, probabilistic models require extensive data analysis and account for a myriad of variables in assigning credit. This complexity allows for the models to be finely tuned according to specific business contexts, providing a nuanced and precise depiction of the customer journey. Key examples of probabilistic models include:

- **Markov Chain Models** scrutinize the sequence of touch points, assessing the probability of transitioning from one touch point to another and thereby determining their influence on the conversion path.
- **Algorithmic Models** employ machine learning algorithms to assess and predict the impact of individual touch points on the likelihood of conversion, offering a dynamic and data-driven approach to attribution.

2.1.3 Choosing The Right Model

In the realm of attribution modeling, selecting the appropriate model is pivotal to accurately understanding and leveraging the customer journey. This decision hinges on several factors, including the model's **Flexibility**, **Accuracy**, and **Resource Requirements**.

Flexibility: Probabilistic models exhibit greater flexibility, adeptly accommodating diverse business scenarios. In contrast, heuristic models, albeit simpler to deploy, may lack this level of adaptability.

Accuracy: Probabilistic models are typically regarded as more accurate, thanks to their reliance on data-driven insights that effectively capture shifts in consumer behavior over time.

Resource Requirements: The adoption and sustained operation of probabilistic models necessitates more resources, including both advanced data handling systems and specialized expertise, compared to heuristic models, which are generally simpler and less demanding in terms of resources.

In summary, the choice between heuristic and probabilistic models involves a trade-off between ease of implementation and the depth of insights offered. Businesses must weigh these considerations carefully to select a model that best fits their specific needs, data availability, and analytical capabilities.

In our case, the choice of a MC model was driven by the fact that a data-driven approach was necessary to rank and give score to the web pages, since none of the heuristic models are able to capture the nuances involved in the customer journey.

2.2 Markov Chain Model

In this section we will briefly talk about the details of the multi-touch attribution algorithm based on MC as well as the data preparation process and the results of the implementation of

this algorithm, i.e. the *page values*.

2.2.1 Deciphering The Algorithm

The algorithm implements a graph-based framework to analyze multi-channel online customer path data as first- and higher order Markov walks.

Markov chains are probabilistic models that can represent dependencies between sequences of observations of a random variable. It is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. The model represents customer journeys as chains in directed Markov graphs.

A **Markov graph** $M = \langle S, W \rangle$ is defined by a set of states

$$S = \{s_1, \ldots, s_n\} \tag{2.1}$$

and a transition matrix W with edge weights

$$w_{ij} = P(X_t = s_j | X_{t-1} = s_i), \quad 0 \leq w_{ij} \leq 1, \quad \sum_{j=1}^{n} w_{ij} = 1 \; \forall i. \tag{2.2}$$

In our case the states correspond to the specific page types and the transition probabilities indicate the probability of a user transitioning from, say, page type *home* to page type *product*.

Hence, for us the set of states can be written as:

$$S = \{\text{home, category, product, checkout, checkout success, other}\} \tag{2.3}$$

Using this graph-based approach allows us to represent and analyze customer journeys in an efficient way as the size of the final graph does not depend on the number of journeys in the data set but only on the number of states.

This chapter focuses on the implementation of the base model, which considers that the present depends only on the first lag, meaning that higher-order Markov models in which the present depends on the last k observations are not studied here.

Before talking about the how the algorithm assigns a score to web pages, it is necessary to understand how the data is prepared and then fed to the algorithm.

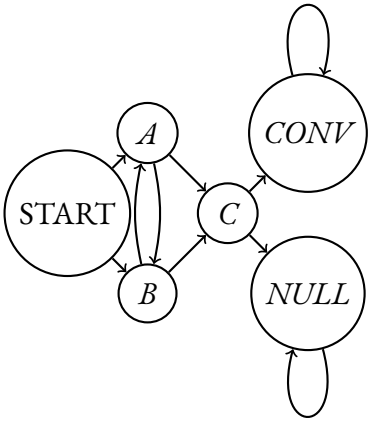
Imagine a simplified scenario where we have two user journeys to analyze and the states set

consists of arbitrary channels that we will name A, B and C. Let's say that User 1 did not buy while User 2 bought. Naturally these journeys can be represented via directed graphs. An example of such could be:

$$\underset{\text{User 1 path}}{\text{A} \longrightarrow \text{B} \longrightarrow \text{C}} \qquad \underset{\text{User 2 path}}{\text{B} \longrightarrow \text{A} \longrightarrow \text{C}}$$

Now, in order to create the transition matrix and define transition probabilities, we add three extra states to the state space : START, NULL and CONVERSION (CONV). Each user journey starts with the START state and ends with either NULL, if the user did not buy anything, or CONVERSION, if the user bought something. Transitions from a node to any other are determined based on the number of transitions from that specific node. Therefore, the probability of moving from state s_i to state s_j is determined by the number of transitions between these states, represented as $w_{ij} = N_{ij} / \sum_k N_{ik}$, where N_{ij} is the number of transitions from state s_i to state s_j and $\sum_k N_{ik}$ is the total number of outgoing transitions from state s_i. If state s_i does not directly lead to state s_j, the probability w_{ij} is 0.

For our example, the corresponding graph along with its transition matrix look like this



$$\begin{array}{c} \\ S \\ A \\ B \\ C \\ + \\ - \end{array} \begin{array}{c} \begin{array}{cccccc} S & A & B & C & + & - \end{array} \\ \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

where S stands for START, + for CONV and - for NULL.

Now that we have shown how to compute the transition matrix we can talk about how the algorithm aids us in giving the importance of each web page.

The channel score is computed using the ***Removal Effect***[55]. The removal effect $Re(s_i)$ of channel s_i helps us answer the question: *How much impact does removing a specific channel*

have on the conversion rate? A higher removal effect means that the channel is important for conversions on the site, and so removing it has a big impact on conversions. It is simply given as

$$Re(s_i) = 1 - \frac{P(CONV \text{ without } s_i)}{P(CONV)} \tag{2.4}$$

where $P(CONV)$ is the probability of a user journey to end with a conversion, which is essentially the sum of the probabilities of user paths that lead to a conversion. For paths p we can write:

$$P(\text{CONV}) = \sum_{\substack{\text{paths } p \\ \text{ending with } s_i = \text{CONV}}} \left(\prod_{(s_i, s_j) \in p} w_{ij} \right) \tag{2.5}$$

After having computed the removal effect for each node s_i, the final thing to do is normalize them. Once this is done, we then essentially have a ranking of which channel is most important. In order to have page values from 0 to 100, we simply divide each removal effect $Re(s_i)$ by the maximal value of the removal effects and multiply by 100:

$$\text{Score}(s_i) = \frac{Re(s_i)}{\max\{Re(s_i)\}_{i=1}^{n})} \times 100 \tag{2.6}$$

Note that computing $\text{Score}(s_i)$ by dividing by the maximal value of the removal effects was a choice, and we could indeed have chosen a different approach like min-max scaling.

2.2.2 Implementation & Results

In order to produce the user paths necessary for the algorithm, we first need to specify the transitions from channel to channel.

Furthermore we can visualize this data using a Sankey diagram, which is a visualization tool used to depict a flow from one set of values to another :

As we can see from 2.1 most transitions appear to come from the *product* channel, of which lots are self-loops. This means that once users visit product pages, they stay there for some time before leaving. Another interesting thing to note is that after users reach *checkout success*, some still continue navigating the website by going back to the *home* page for example. This suggests that after buying something from the website users might still be interested in navigating further (and possibly buying again, although unlikely).

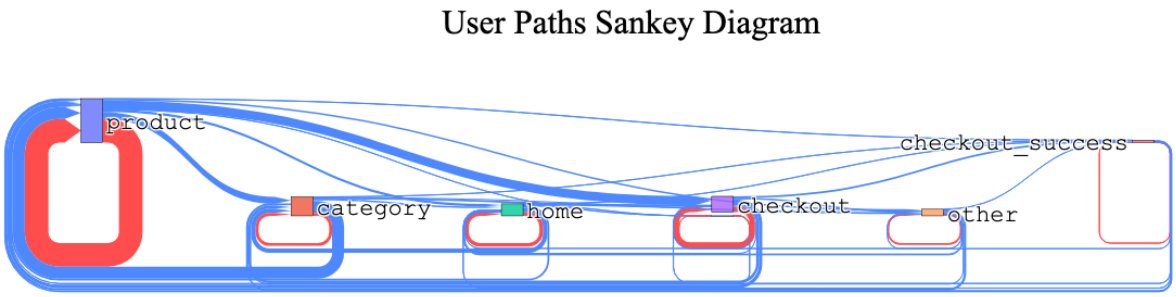


Figure 2.1: Sankey Diagram of User paths : Red transitions are self-loops and thicker lines indicate more transitions. Height of the rectangles is arbitrary

Once we have gathered the data from every user, we can build its path and send the corresponding transition matrix to the algorithm. The results of the algorithm, here computed for a specific time frame, are the following:

s_i (Page Type)	$Re(s_i)$	Score(s_i)
checkout success	74.10	100
checkout	67.12	90.53
product	62.68	84.40
category	51.07	68.94
home	35.20	47.49
other	18.82	25.11

Table 2.1: Results of Markov Chain Attribution Modelling

We can see that the algorithm successfully identified the *checkout success* page type as being the most important one. The *checkout* channel is a close second, which makes sense intuitively. Moreover we can also see that *product* was third most important, followed by *category*, *home* and *other*. The ordering of channels in this manner is more or less what we would have expected if we had approached the problem from a qualitative standpoint, by using domain knowledge to assign these scores.

The advantage of using a quantitative approach in this case is that it also allowed us to assign values to the more nuanced page types, like *other*; we know that this is not an important channel, but nevertheless the exact score to give to this page type becomes problematic if we do not use algorithmic approaches.

Another advantage of approaching the problem quantitatively is that in this manner we can

more easily track how these scores change throughout time. By looking at how the page type scores change in time we can understand if certain pages have become more or less important, indicating a possible problem in the e-commerce website or simply the fact that the path which users take to buy has changed.

This information can be leveraged to take business decisions according to the trend of these page type scores over time.

The plot below shows the variation of the page values over time, computed each time over a period of 30 days and by sliding each time by one day. For readability only five time frames are shown.

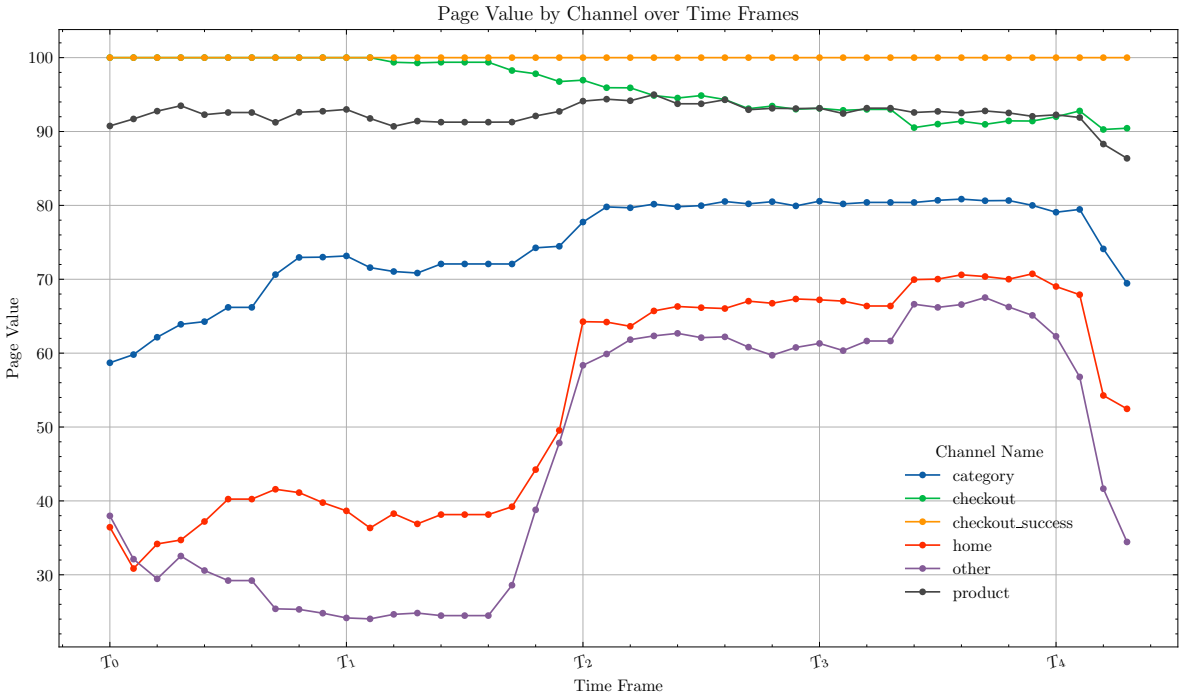


Figure 2.2: Page value by channel over time frames T_i

We can see that the general order of page value scores is mostly maintained throughout time between the different page types. Right before T_2 *home* and *other* have an upwards trend; they are becoming more important with respect to *checkout success*. *Product* and *category* also seem to be increasing in importance, meaning their removal effect score is closer to the one of *checkout success*. It seems that only *checkout* is losing in importance during this time.

In periods from T_2 to T_4 a constant behaviour can be observed for each score in general, while also the values of *product* and *checkout* become closer to each other.

At the start of T_4 the scores decrease sharply. It is difficult to say why there was a rise at the first time frames and a decrease at the later ones, although what we can say is that the relative importance of these scores with respect to the page type *checkout success* changed. This decrease in importance should be further investigated to understand the causes and potential implications of it.

3

Clustering Users

Clustering or cluster analysis is a task of unsupervised learning concerned with finding groups in a set of objects in such a way that objects in the same group are more similar than objects in other groups. The groups may form a partition of the object set, but they may also be overlapping or non-exhaustive. Group memberships may be crisp or fuzzy, meaning that a data point may strictly belong to one cluster or probabilistically, depending on the algorithm used.

It is widely acknowledged that there is no agreed definition of what a cluster is, and a formal definition of what the "true clusters" are for a dataset is also hard if not impossible to give. This is because there is never truly a "right" answer when labels are unknown. Even if one algorithm might fit a certain dataset well, there are no guarantees that it will work on a different dataset in the exact same way.

Moreover, clustering is noted as being "strongly dependent on contexts, aims and decisions of the researcher," which adds fire to the argument that there is no such thing as a "universally optimal method that will just produce natural clusters"[56].

In this chapter we attempt to cluster users of an e-commerce website in an attempt to have clusters in which users have a similar propensity of making a purchase. These clusters can then aid in computing the *missed revenue*, which is the total amount of money lost by faulty behavior of the website. We will use RF to extract the relevant features for our clustering problem, apply two algorithms capable of dealing with mixed data types, that is numerical and categorical features, which are K-prototypes[57] and Uniform Manifold Approximation and Projection[58] (UMAP) coupled with Hierarchical Density-Based Spatial Clustering of Applications with

Noise[59] (HDBSCAN). Later in the chapter we will also briefly talk about the assumption of normality and how the size of a dataset may trick statistical tests testing for normality. After having applied the clustering algorithms we will also visualize the clusters to try and interpret the results.

3.1 Exploratory Data Analysis

Before applying any algorithm it is always recommended to explore the data by doing some basic plots like scatter plots or histograms, cleaning the data and removing outliers, if necessary. The data at hand extends over a period of three months and contains user information split into ten variables (six categorical and four numeric) that are potential candidates for our cluster analysis. A brief explanation of what each variable means is given below:

1. **Total page viewed:** the total number of pages a user viewed during a single session. This can indicate how engaged the user was with the content.

2. **Total session time:** the total time spent by the user in a session, measured in seconds. Longer sessions indicate higher engagement.

3. **Total interactions:** the total number of interactions (clicks, form submissions, etc.) a user had with the website during their session. This provides a measure of user activity and engagement.

4. **Referrer type:** categorical variable which can either be *organic*, representing traffic from unpaid search engine results, *paid*, representing traffic from online ads related to paid marketing campaigns and finally *direct* which represents the traffic from users entering the site's URL directly or through untracked links, reflecting direct user interest or brand recognition.

5. **Referrer:** categorical variable corresponding to the URL or identifier of the web page or source from which the user came to the e-commerce site. It helps in understanding where traffic is coming from, e.g. *Facebook,google,...*

6. **User agent device:** categorical variable holding information about the device used by the visitor, such as mobile, tablet, or desktop computer. This helps in understanding the device preferences of the user base.

7. **Country code:** categorical two-letter variable known as International Organization for Standardization (ISO) code, constituting the user's location. It's valuable for geo-targeting and understanding the geographic distribution of users.

8. **User agent browser info:** categorical variable containing details about the browser the user employed to access the website, e.g. *Chrome, Firefox, Safari,..*
9. **Total session click:** the total number of clicks made by the user during their session. Similar to total interactions but specifically focused on click actions.
10. **is campaign:** a binary indicator showing whether the user's session was a result of a marketing campaign. This helps in measuring the effectiveness of marketing efforts

Now that we have defined what each variable means we can examine the data by showing some basic histograms and bar plots:

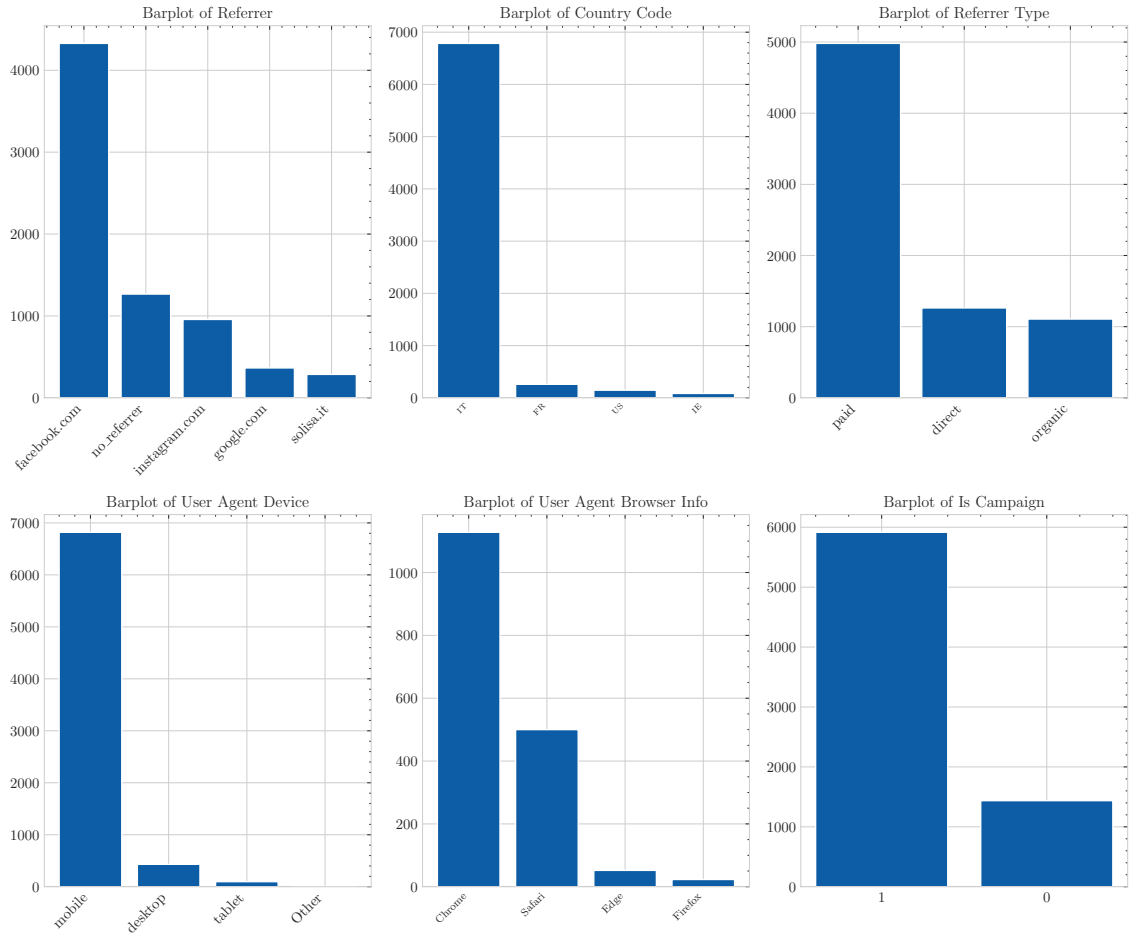


Figure 3.1: Bar plots of categorical variables: referrer, country code, referrer type, user agent device, user agent browser info and is campaign

It is worth mentioning that the data we have also contains two additional variables **session id** and **success**. Session id is a unique identifier for each user session on the website which helps

tracking individual visits. There is not a one-to-one correspondence between session ids and users, since a user (physical person) may visit an e-commerce website multiple times during an extended period of time, and each time he may have a different session id. For the cluster analysis task we will make the approximation though that one session id corresponds to one user. Success on the other hand indicates whether a user has made a purchase or not.

Few Comments about the bar plots in 3.1: It seems like most users enter the website through paid referrer types, like marketing campaigns from *Facebook* for example. A large proportion of users also entered the website in a direct way, without a referrer needed. Moreover, since this is an Italian website, most users come from Italy as can be seen by the country code variable being *IT*. Also, most users access the website via mobile. There has been a rise in recent years of e-commerce solutions optimizing for mobile users, so much so that the term *m-commerce*[60] was coined to indicate e-commerce companies specialized in mobile applications. Indeed our data reflects the need for such specialisations, as targeted solutions are now necessary to meet user needs.

Finally we observe that most users come from marketing campaigns, which is expected because the most frequent referrer type is *paid*.

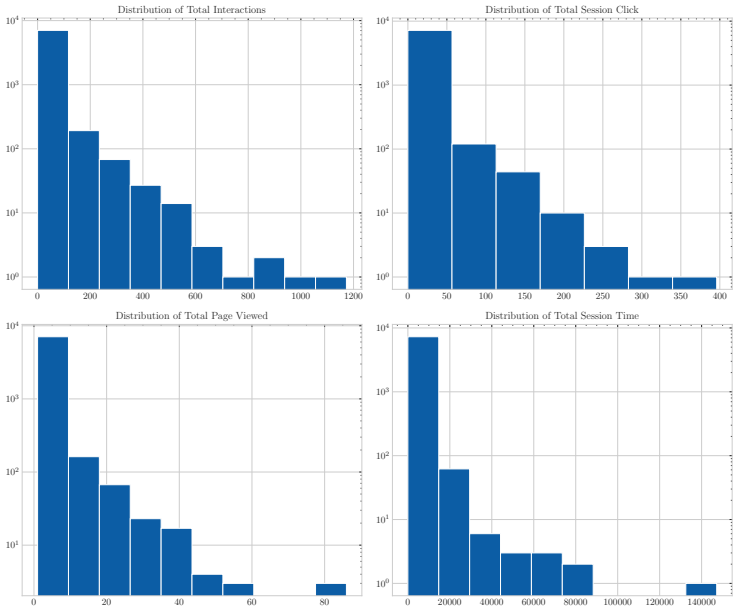


Figure 3.2: Histogram of numerical variables in log scale: total interactions, total session click, total page viewed abd total session time

The log scale was needed for better visualization. We observe in 3.2 that all numerical variables exhibit an exponential decay. The most frequent value is therefore close to 0 and this is expected because a lot of sessions will be what's called *bounce* sessions. These are users who enter the website and quickly leave after spending little time on it and don't really engage much.

3.2 Random Forest Feature Extraction

In order to successfully perform clustering we need to select a subset of features which are deemed relevant for our task. The goal is to reduce the initial ten features and only keep the ones that are most strongly correlated to our response variable *success*.

Keeping only the features most strongly correlated to our response helps the clustering algorithms converge to solutions that partition the data in the "correct" way, although as we have already mentioned[56] the truly correct way to partition the data doesn't exist.

Anyway, in order to find this subset of features we use RF. RF[6] is an ensemble method for classification and regression, which works by constructing many decision trees at training time. For classification tasks, the output is the class selected by most trees while for regression, the mean or average of the individual trees is returned.

3.2.1 Gini Importance

RF can also be used to rank the importance of variables. We use the the **Mean Decrease in Impurity** technique to compute the feature importance, which is the default implementation in *scikit-learn*, and a way to measure how much each feature helps in organizing the data. Variables decreasing the impurity during splits are considered important and get a higher mean decrease impurity score.

Breiman[6] proposed to evaluate the importance of a variable X_m for predicting Y by adding up the weighted impurity decreases $p(t)\Delta i(s_t, t)$ for all nodes t where X_m is used, averaged over all N_T trees in the forest:

$$Imp(X_m) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = X_m} p(t)\Delta i(s_t, t) \tag{3.1}$$

and where $p(t)$ is the proportion N_T/N of samples reaching node t, $v(s_t)$ is the variable used in split s_t and $i(t)$ the impurity measure. We use the Gini index as impurity function, which makes this importance score known as *Gini importance* or *Mean Decrease Gini*.

The Gini index is defined as

$$i(t) = 1 - \sum_{j=1}^{C} p(j|t)^2 \tag{3.2}$$

where $p(j|t)$ is the proportion of samples in node t that belong to class j, and C is the number of classes. It represents the probability of incorrectly classifying a randomly chosen element in the dataset if it were randomly labeled according to the distribution of labels in the node.

Note that different ways to compute feature importances in RF exist, one such is known as **Permutation Feature importance**[6] and it works by randomly shuffling the values of a single feature and observing the resulting degradation of the model's score. By breaking the relationship between the feature and target, we can see how much the model relies on that particular feature.

Impurity-based feature importance scores are generally biased towards high cardinality features[61], and if computed on a training set they might not reflect the ability of a feature to be useful to make predictions on test data. This is because the RF model can still use them to overfit.

In our case, the feature importances computed from both methods yield the same results for the top features, so we will proceed with the impurity-based method and by splitting the data in a *80/20* training/test fashion.

3.2.2 Results

We use *scikit-learn*'s *RandomForestClassifier* for this, where the number of trees in the forest can be controlled by the parameter *n_estimators* and we will use the default value of 100. By default the importance scores are normalized so that they can more easily be interpreted. The categorical variables are one-hot encoded and then the importance results aggregated.

It is also worth mentioning that the dataset is highly imbalanced, in fact only around 2.5% of users end up buying. This affected the performance of RF in terms of correctly identifying samples belonging to the minority class, compared to the majority class; in fact the *Recall* score was 0.88 while *Specificity* was 0.99. Defined as:

$$Recall\ (Sensitivity) = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.3}$$

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} \tag{3.4}$$

Regardless,the importance results can be observed below:

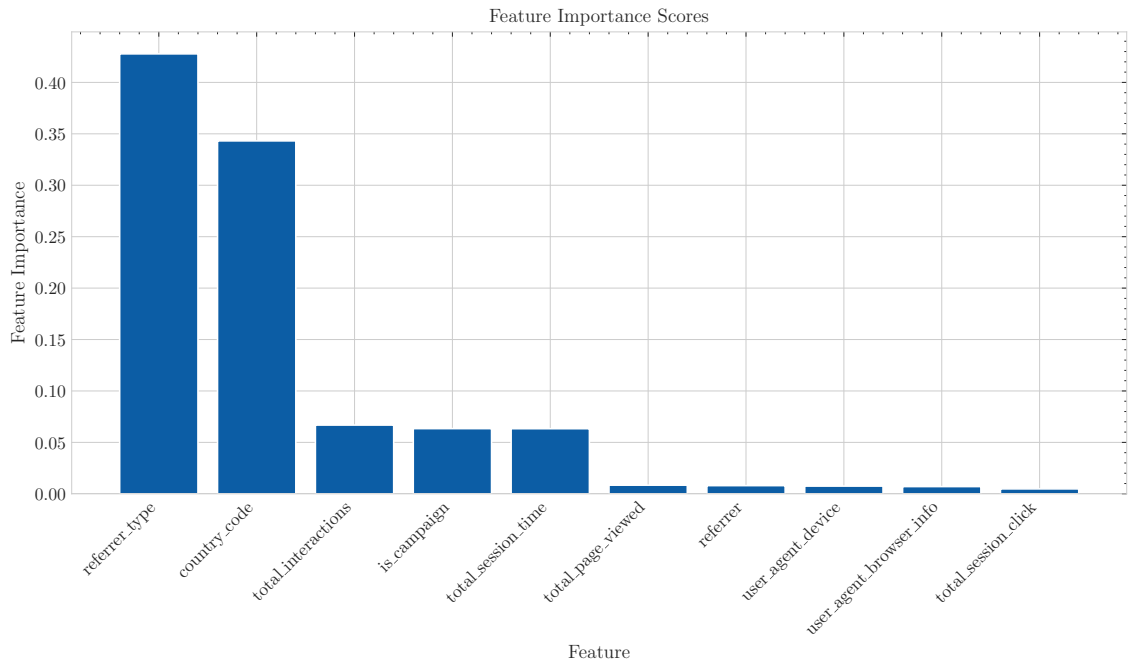


Figure 3.3: Bar plot of Feature Importance Scores

For clustering we will pick the top five most important variables (visible from 3.3), since the bottom five seem to have a very low impact on predicting *success*. The correlation between certain numeric variables is high, and it looks like RF was able to recognize this and remove redundant information. For example the total amount of time spent on the website is highly correlated with the number of total clicks the user made and also the number of pages he viewed.

3.3 K-prototypes Clustering

K-prototypes[57] is a clustering algorithm specifically designed to cluster mixed data. It combines the well-known K-means[62] and K-modes[63] algorithms which are used to cluster nu-

merical and categorical data respectively. We can't use K-means because K-means relies on calculating means for cluster centers, and means don't make sense for categorical data, even if one-hot encoding is applied. In the following subsections we will discuss how the algorithm works and its implementation for our data.

3.3.1 Deciphering The Algorithm

Consider a data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_j, 1 \leq j \leq n$, is the jth data point with an array of m attributes, i.e., $\mathbf{x}_j = \{A^c_{1,j}, A^c_{2,j}, \ldots, A^c_{p,j}, A^n_{p+1,j}, \ldots, A^n_{m,j}\}$; here $A^c_{s,j}, 1 \leq s \leq p$, is the sth categorical attribute of the jth data point, $p+1 \leq s \leq m$, is the sth numerical attribute of the jth data point. We assume that every data point in the data set is described by exactly m attributes. The objective of the clustering algorithm is to divide the data set into k clusters by minimizing the cost function as given in the following.

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two data points in their mixed attribute space. The dissimilarity between these two data points is defined as follows:

$$\text{dist}(\mathbf{x}_1, \mathbf{x}_2) = \sum_{s=p+1}^{m} (\mathbf{A}^n_s(\mathbf{x}_1) - \mathbf{A}^n_s(\mathbf{x}_2))^2 + \gamma \sum_{s=1}^{p} d_c(\mathbf{A}^c_s(\mathbf{x}_1), \mathbf{A}^c_s(\mathbf{x}_2)) \tag{3.5}$$

where the first part of the equation is the squared Euclidean distance on numerical attributes and the second part is simple matching distance on categorical attributes. The variable γ is a user-specified parameter to balance the proportions of two distance functions. The simple matching distance between two categorical attributes is defined as follows:

$$d_c(\mathbf{A}^c_s(\mathbf{x}_1), \mathbf{A}^c_s(\mathbf{x}_2)) = \begin{cases} 0 & \text{if } \mathbf{A}^c_s(\mathbf{x}_1) = \mathbf{A}^c_s(\mathbf{x}_2) \\ 1 & \text{if } \mathbf{A}^c_s(\mathbf{x}_1) \neq \mathbf{A}^c_s(\mathbf{x}_2) \end{cases} \tag{3.6}$$

The objective of the k-prototypes algorithm is to divide the data set into k clusters by minimizing the cost function as given in the following equation:

$$C(W, Q) = \sum_{i=1}^{k} \sum_{j=1}^{n} w_{ji} \text{dist}(\mathbf{x}_j, q_i) \tag{3.7}$$

where $\mathbf{x}_j$ is the jth data point from D and q_i is prototype of ith cluster, i.e. the point considered most representative of that cluster; $w_{ij} \in (0, 1)$ is an element of partition matrix $W_{(n \times k)}$.

The pseudo-code for the algorithm can be written as such:

Algorithm 3.1 K-Prototypes Algorithm

1: Choose an initial prototype set $Q^{(1)}$ from D. Find $W^{(1)}$ such that $C(W, Q^{(1)})$ is minimized. Set $t = 1$.
2: Find $Q^{(t+1)}$ such that $C(W^{(t)}, Q^{(t+1)})$ is minimized.
3: **if** $C(W^{(t)}, Q^{(t+1)}) = C(W^{(t)}, Q^{(t)})$
4: **stop**
5: **else**
6: go to step 3
7: **end if**
8: Find $W^{(t+1)}$ such that $C(W^{(t+1)}, Q^{(t+1)})$ is minimized.
9: **if** $C(W^{(t+1)}, Q^{(t+1)}) = C(W^{(t)}, Q^{(t+1)})$
10: **stop**
11: **else**
12: set $t = t + 1$ and go to step 2
13: **end if**

3.3.2 Implementation & Results

The algorithm is implemented thanks to the *kmodes* library in python. It is possible to choose different dissimilarity measures for both numerical and categorical variables. We use the default euclidean distance for the numerical variables, and the matching similarity for the categorical ones. The weighing factor γ is automatically calculated from the data and the initialization of the algorithm is set to random.

The number of clusters is chosen with the help of the **Elbow method**. The elbow method is a heuristic method which consists of plotting the value of the cost function for different number of clusters and picking the *elbow* of the curve, which is the point where the curve visibly bends, as the number of clusters to use. It is a common method used in mathematical optimization to choose a point where diminishing returns are no longer worth the additional cost. It is a subjective way to choose the number of clusters and in many practical applications, like is the case for us, the choice of an "elbow" is ambiguous because the plot may not contain a sharp elbow.

The plot of the cost function value for different values of k is the following:

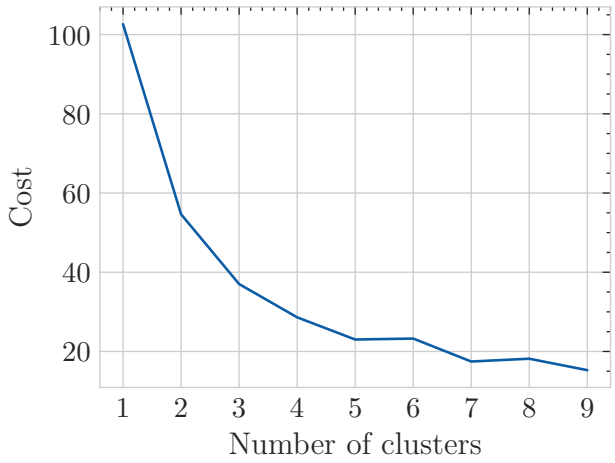


Figure 3.4: Elbow method plot; value of cost function for different number of clusters

As we can see in 3.4 there is no sharp elbow in the plot and the values $k = 2, 3, 4$ or 5 are all viable options. Nevertheless, we can look at the feature importance to aid us in selecting the "optimal" number of clusters. As we have seen the *referrer type* is the most important variable according to the importance scores obtained. Since this value is categorical and can take on three distinct values it is wise to choose the number of clusters to be $k = 3$ (and actually after performing numerous tests with various values of k it indeed seems like the clusters obtained with $k = 3$ are most interpretable and intuitive).

Now that we have chosen the number of clusters we can apply the algorithm and interpret the results.

In terms of data pre-processing, for the categorical variables we use **label encoding**, which converts each unique categorical value within a column to an integer value in the range 0 to $n - 1$ unique values in the column. Instead for the numerical variables we use **normalization**, i.e. Min-Max scaling. There isn't really a rule to tell which data pre-processing technique to use for numerical features when it comes to clustering, as both normalization and standardization are frequently used[64] in clustering. In our case normalization gave more meaningful clusters so we stick to this one.

We can now comment on the results obtained from K-prototypes :

Cluster	Users	Users Who Bought	% of Purchases in Cluster	CR (%)	Interactions		Session Time (s)	
					Mean	Median	Mean	Median
0	4980	32	18.60	0.64	16.39	7	206.18	21
1	1205	126	73.26	10.46	63.21	24	2030.97	90
2	1166	14	8.14	1.20	23.93	10	333.07	28

Table 3.1: K-prototypes Cluster Statistics Including User Purchases, Conversion Rates (CR), and Interaction/Session Time

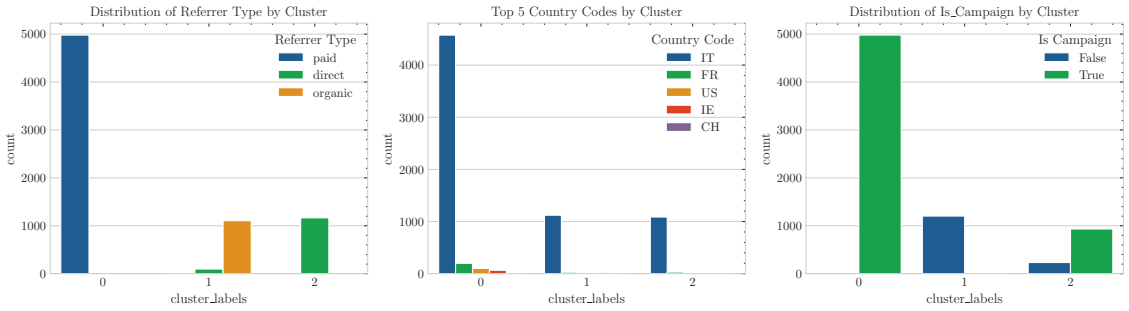


Figure 3.5: K-prototypes Cluster statistics Including Referrer Type, Country Code And Campaigns

According to table 3.1, we can see that Cluster0 (C0) has the most amount of data points while Cluster1 (C1) and Cluster2 (C2) have around the same amount of points. Furthermore, C1 seems to be clearly the cluster containing the most amount of users who ended up buying, since it has the highest conversion rate (10.46) and also highest percentage of total users who bought (73.26). It is also visible that the mean and median for interaction and session time are considerably higher for C1 compared to C0 and C2, which intuitively makes sense; it is expected that users who buy have a larger number of interactions and a greater total session time. It also makes sense that the median is much lower than the mean for both interactions and session times, since we know from the histograms 3.2 that we are dealing with right-skewed distributions.

When it comes to interpreting C0 and C2 we can say the following: since the conversion rate, mean and median interactions/total session time are higher for C2 compared to C0, then we can conclude that C2 is likely the cluster with the second highest propensity of buying.

Thus our clustering results have given the following classification in terms of buying propensity: $\boxed{\text{Cluster1} \succ \text{Cluster2} \succ \text{Cluster0}}$.

Now, figure 3.5 tells us the characteristics of the users belonging to the different clusters:

- For **C1** it tells us that users in that cluster mostly come from unpaid search engine results, i.e. *organic* referrer type. Also they don't come from marketing campaigns and mostly come from Italy.

- For **C**2 it shows that these users have the referrer type as *direct*, their country code is IT and it has a mix of users coming from campaigns and not, most of them coming from campaigns. Likely the portion of users not coming from campaigns are those who end up buying.

- For **C**0 these users come from campaigns and with a referrer type *paid*. We also observe that this cluster contains users from nationalities other than Italy. It contains users who are not interested in buying as we've seen, which makes sense since users from countries other than Italy are not as interested in buying (because the website is Italian).

To conclude this section we can confidently say that K-prototypes successfully clustered the data in a way that is meaningful, with clusters of different buying propensities which are fairly interpretable.

3.4 Density-based Clustering

In this section we will explore another technique to cluster users of an e-commerce that falls under the category of density-based clustering[65].

From a statistical point of view, non-density based methods correspond to a parametric approach where the unknown density $p(x)$ of the data is assumed to be a mixture of k densities $p_i(x)$, each corresponding to one of the k groups in the data. The $p_i(x)$ are assumed to belong to some parametric family (e.g., Gaussian distributions) with unknown parameters. The parameters are then to be estimated based on the given sample (the data set) as, e.g., in k -means, where the means of k Gaussians are estimated, assuming unit variances and zero co-variances for each of the k models.

In contrast, density-based clustering is a non-parametric approach where the clusters are considered to be high density areas of the density $p(x)$. Density-based clustering methods do not require the number of clusters as input parameters, nor do they make assumptions concerning the underlying density $p(x)$ or the variance within the clusters that may exist in the data set. As a consequence, density-based clusters are not necessarily groups of points that have a convex shape but can be arbitrarily shaped in the data space. Intuitively, a density-based cluster is a set of data objects spread in the data space over a contiguous region of high density of objects, separated from other density-based clusters by contiguous regions of low density of objects.

In this section we use the combination of UMAP and HDBSCAN to produce high-quality clusters of mixed data. We will also briefly mention the normality assumption in machine and

deep learning with different sample sizes, while finally talking about the results obtained from this clustering approach.

3.4.1 UMAP & HDBSCAN

The combination of UMAP and HDBSCAN for clustering was influenced by the *Amazon DenseClus*[66] package which was specifically designed to cluster mixed-type data. UMAP is used to to map mixed-type data into a dense, lower dimensional space. From this dense space, HDBSCAN then builds groups hierarchically into clusters based on the density of points.

UMAP is an algorithm for dimension reduction based on manifold learning techniques and ideas from topological data analysis, Riemannian geometry, and algebraic topology. Because the mathematics involved in fully understanding the algorithm are rather complex, we will limit ourselves with a high-level explanation.

Most dimensionality reduction techniques like PCA[9] or t-distributed stochastic neighbour embedding (t-SNE)[67] fit into either of two broad categories: Matrix factorization algorithms (e.g PCA) or Graph layout algorithms (e.g. t-SNE). UMAP is a graph layout algorithm, similar to t-SNE but with a more solid theoretical base.

In its simplest sense, the UMAP algorithm works in two steps:

1. Construction of a weighted graph from the high-dimensional data, with edges representing how ”close” a point is to another
2. Optimization step to find the most similar graph in lower dimensions, via a graph layout algorithm

In order to achieve this goal, the algorithm relies on algebraic topology and Riemannian geometry.

UMAP has four parameters that need to be tuned:

- **n_neighbors** parameter controls how UMAP balances local vs global structure in the data. Low values will force UMAP to concentrate on very local structure, while large values will push UMAP to look at larger neighborhoods of each point when estimating the manifold structure of the data
- **min_dist** parameter controls how tightly UMAP is allowed to pack points together. It provides the minimum distance apart that points are allowed to be in the low dimensional representation. Low values will result in clumpier embeddings, while larger values will prevent UMAP from packing points together and will focus on preserving the broader structure instead

- **n_components** allows the user to determine the dimensionality of the reduced dimension space we will be embedding the data into.
- **metric** parameter controls how distance is computed in the ambient space of the input data.

HDBSCAN on the other hand extends DBSCAN by converting it into a hierarchical clustering algorithm. It is a density-based clustering algorithm that segregates data points into high-density regions separated by regions of low-density. It does not require that every data point is assigned to a cluster, since it identifies dense clusters, and thus points not identified to clusters are considered as outliers or noise. The "hierarchical" part of the algorithm allows it to find clusters of any densities, which is not the case for the classic DBSCAN algorithm. A great advantage of hierarchical clustering algorithms is that they can work without the need to specify the number of clusters, and that can provide high interpretability since the results can be presented in a dendrogram.

The algorithm is described as follows[68]:

Algorithm 3.2 HDBSCAN main steps

1: Compute the core distance for the k nearest neighbors for all points in the dataset;
2: Compute the extended minimum spanning tree from a weighted graph, where the mutual reachability distances are the edges;
3: Build the HDBSCAN hierarchy from the extended minimum spanning tree;
4: Find the prominent clusters from the hierarchy;
5: Calculate the outlier scores;

The first step in the algorithm is to compute the core distances for each point in the dataset. The core distance of a point is the distance to the kth nearest neighbor. The value of k includes the point itself. This is a technique to obtain an approximate density for each point. Using the core distances, a new distance metric is computed which is called the mutual reachability distance. The mutual reachability distance between two points x and y, is the maximum value of: the core distance of x, the core distance of y, or the distance between x and y. Next, the mutual reachability distances between the points can be used to establish a weighted graph, where the data points are vertices and an edge between any two points has the weight of the mutual reachability distance of those points. Now we need to compute the graph's minimum spanning tree. A minimum spanning tree is a spanning tree whose sum of edge weights is as small as possible. That is, it has all vertices connected with a subset of the edges from the complete

graph, without any cycles and with the minimum possible total edge weight. The resulting minimum spanning tree is modified by adding a self-edge to each vertex with the point's core distance as the weight. This gives a graph called the extended minimum spanning tree. Now, the graph can be used to build the HDBSCAN hierarchy. To begin with, there is one cluster label, and all the points are assigned to this cluster. This cluster is added to a cluster list. The graph is then sorted by edge weight in ascending order. Starting from the bottom of the graph, edges are iteratively removed from the extended minimum spanning tree. Edges with equal weights must be removed simultaneously. The weight value of the edge(s) being removed is used to denote the current hierarchical level. As an edge is removed, the cluster containing the removed edge is explored. Clusters that have become disconnected and contain fewer points than the minimum cluster size are all assigned with the noise label. A cluster that has become disconnected, but has more points than the minimum cluster size, is assigned a new cluster label. This is called a cluster split. Additionally, the new cluster is added to the list of clusters. A new hierarchy level is created when an edge removal has resulted in new clusters due to cluster splits. By the end of the process, in the last level of the hierarchy, all the points in the dataset will have been assigned to noise. During the construction of the HDBSCAN hierarchy, a list of the clusters is also maintained, where each cluster holds a reference to its parent.

After constructing the hierarchy, the next step is to identify the most prominent clusters. This involves calculating the stability of a cluster, defined as:

$$\text{stability} = \sum_{p \in \text{Cluster}} (\lambda_p - \lambda_{\text{birth}}) \tag{3.8}$$

where $\lambda = 1/\text{edge weight}$, λ_{birth} is when the cluster was created, and λ_p is the value at which a point leaves the cluster.

The algorithm propagates stabilities up the hierarchy, comparing the stability of a cluster against the cumulative stability of its descendants to determine prominence. The root cluster ultimately references the clusters with the highest stabilities.

In addition to identifying clusters, HDBSCAN computes the outlier scores for each data point, known as GLOSH (Global–Local Outlier Score from Hierarchies). This computation takes into account additional information recorded during hierarchy construction. For a data point x, the values of $e(x)$, the edge weight prior to being marked as noise, and $e_{\max}(x)$, the

lowest death level of its last cluster, are used in the GLOSH formula:

$$\text{GLOSH}(x) = 1 - \frac{e_{\max}(x)}{e(x)} \tag{3.9}$$

This score provides an assessment of how likely it is that each point is an outlier, based on its participation in the clusters throughout the hierarchy.

Next, this pipeline puts together all elements of our clustering task:

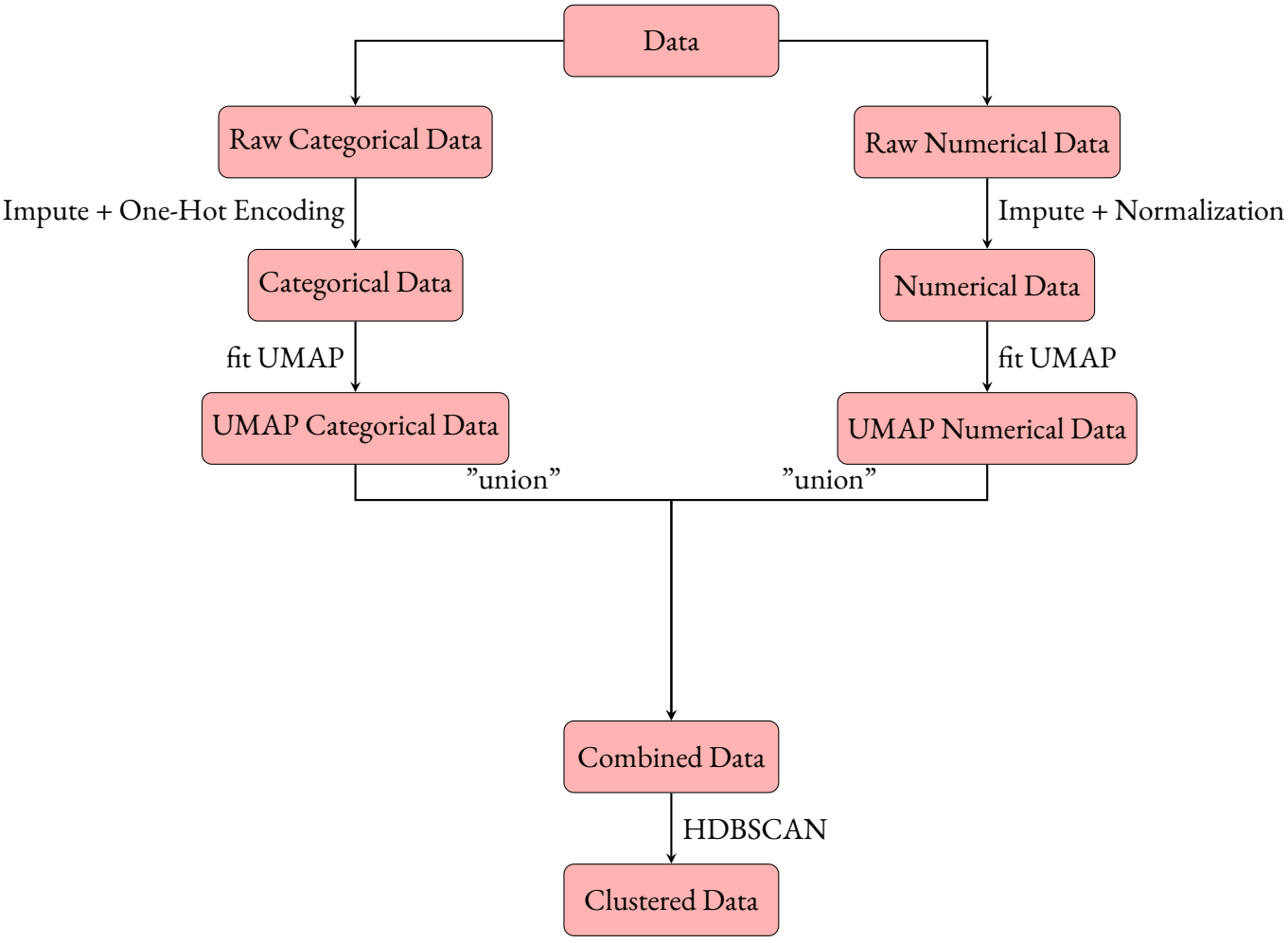


Figure 3.6: HDBSCAN clustering pipeline with data pre-processing and model fitting

The data first gets split into categorical and numerical data. Then some data pre-processing is done to impute missing values on the categorical data using the mode, while for numerical

data we use the median value. We normalize the numerical data using a non-linear power transformation technique called Box-Cox[69], given as:

$$x(\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases} \tag{3.10}$$

while for categorical data we use one-hot encoding. Once this is done we fit the UMAP algorithm separately on numerical and categorical data to obtain the embeddings in the lower dimensional, dense space. Finally we combine together the embeddings using the "union" method[66], which consists in simply adding the embeddings together. We do this because this method of combination supposedly preserves the categorical embeddings more[66], which is what we want because categorical variables are more important, as we've seen in 3.2. Finally the combined data is fed to HDBSCAN to obtain the results of the clustering.

HDBSCAN also has a few parameters to tune:

- **min_samples** parameter controls how conservative you want the clustering to be. The larger the value, the more conservative the clustering - more points will be declared as noise. It is the minimum number of neighbours to a core point.
- **min_cluster_size** parameter controls the minimum size a final cluster can be
- **cluster_selection_method** used to select clusters from the condensed trees.
- **metric** to use when calculating the distance between instances

3.4.2 Normality Assumption & Sample Size

Before talking about the results obtained using density-based clustering, it is important to talk about data normalization and sample size. The clustering pipeline 3.6 shows that the numerical data pre-processing step involves the normalization of the data.

Data normalization techniques like Box-Cox[69] have as a goal to make the data more Gaussian-like, and thus it comes as a natural question whether transformations like these are effectively able to transform the data distribution successfully

In order to gauge how well these transformations do we can use two tools in our arsenal: **visual inspection** and **statistical tests**. Visual inspection consists in visualizing a histogram representation of the data, or we can use a Quantile-Quantile plot which plots the quantiles of

the distribution at hand vs those of a normal distribution. Statistical tests on the other hand offer a more objective assessment, providing numerical values that can indicate the likelihood that the data was drawn from a given distribution.

We will focus on the Anderson-Darling test[70].

The Anderson-Darling test is a statistical test used to check if a sample comes from a population with a specific distribution.

The test works by calculating a measure of discrepancy between the empirical distribution function of the sample and the cumulative distribution function of the specified theoretical distribution.

The hypothesis for the Anderson-Darling test are defined as follows:

- H_0: The data follow a specified distribution.
- H_a: The data do not follow the specified distribution.

The test statistic for the Anderson-Darling test is given by the equation:

$$A^2 = -N - S \tag{3.11}$$

where

$$S = \sum_{i=1}^{N} \frac{(2i-1)}{N} \left[\ln F(Y_i) + \ln\left(1 - F(Y_{N+1-i})\right)\right] \tag{3.12}$$

and F is the cumulative distribution function of the specified distribution, and Y_i are the ordered data values.

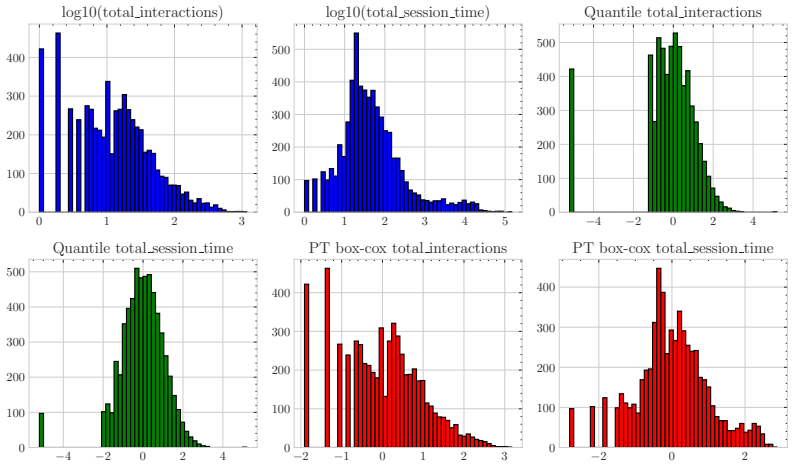
The test is inherently a one-sided test, and the null hypothesis that the distribution fits is rejected if the test statistic, A^2, is larger than the critical value.

With this being said, we will now investigate the role of sample size in statistical normality tests.

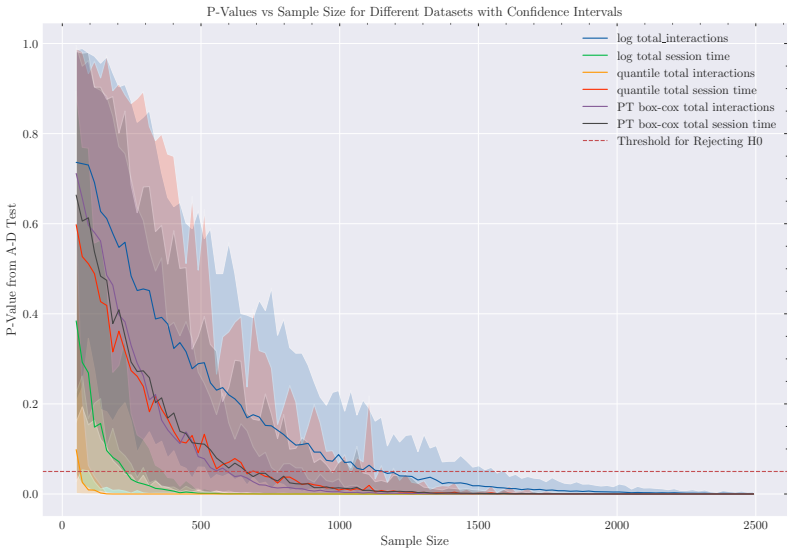
The data we are transforming is the *total session time* and the *total interactions*. As we've seen previously, this data has a right-skew, so a transformation is necessary.

We use three transformations to make the data Gaussian-like, which are the $\log_{10}$, Power transform Box-Cox and the Quantile transformation. The last two are implemented using *sklearn*'s *PowerTransformer* and *QuantileTransformer*.

The full data contains around 6000 data points. We will now proceed by plotting the p-values obtained by performing the Anderson-Darling test for a single run, for different sample sizes :



(a) Histograms of the entire data of numerical variables for different transformations



(b) P-values from Anderson-Darling test for different sample sizes and datasets with a 95% confidence intervals computed

Figure 3.7: Histograms of transformed data and P-values from Anderson Darling test

As we can see from 3.7a visual inspection tells us that the transformations worked in making the full data more Gaussian-like. In particular the log and Power transform seem to have been most successful, while the Quantile transformation a little less. Instead when we look at 3.7b, we see a pattern; the bigger the sample size (the plot goes up to 2500), the more likely it is for the p-value to be smaller than the significance level (here set at 0.05) suggesting that the samples do not come from a normal distribution. For this plot, 100 runs of the Anderson-Darling tests are made for each sample size, each point in the sample being drawn at random and without

replacement from the original dataset.

So the results from 3.7a and 3.7b are clearly contradictory. What should we believe then?

The recent paper "Comparison of Normality Tests in Terms of Sample Sizes under Different Skewness and Kurtosis Coefficients"[71] compares normality tests in different sample sizes in data with normal distribution under different kurtosis and skewness coefficients obtained simulatively. The authors conclude that "normality tests were not affected by the sample size when the skewness and kurtosis coefficients were equal to or close to zero; however, in cases where the skewness and kurtosis coefficients moved away from zero, it was found that normality tests are affected by the sample size, and such tests tend to give significant results. Therefore, in large samples, it may be suggested that critical values for skewness and kurtosis coefficients' z-scores or the histogram graphs be used".

The sample sizes they tested went up to a maximum of 900 and skewness/kurtosis coefficients ranged from -0.5 t 0.5. As a reference, these are the skewness and Kurtosis coefficients we are dealing with in our dataset (computed with *scipy.stats* and using *Fisher's* definition that a normal distribution has kurtosis of zero):

Dataset	Skewness	Kurtosis
log total_interactions	0.3106	-0.1894
log total session time	1.0438	1.6660
quantile total interactions	-1.8159	3.8848
quantile total session time	-1.1722	4.7594
PT box-cox total interactions	0.0048	-0.3279
PT box-cox total session time	-0.0306	0.4855

Table 3.2: Skewness and Kurtosis of Datasets

To conclude this part we can say that indeed it's not that our data is not normal after transforming it, but instead the statistical tests are more sensitive in detecting deviations from normality in cases where there are a lot of samples. It is better to rely on histograms, as the authors point out.

3.4.3 Implementation & Results

Before talking about the results of density-based clustering we will talk about the selection of parameters for UMAP and HDBSCAN. For both algorithms we select the parameters through a grid search and pick the best ones according to specific scores.

For UMAP we look at the **Trustworthiness score**[72] to evaluate the dimensionality reduction. The distance of point i in high-dimensional space is measured against its k closest neighbours using rank order, and the extent to which each rank changes in low-dimensional space is measured. For n samples, let $r(i,j)$ represent the rank of sample j based on its distance from sample i, in comparison to the distances from i to all other samples in the high-dimensional space $U_i^{(k)}$. Similarly, let $\hat{r}(i,j)$ be the rank of the distance between sample i and sample j in low-dimensional space $V_i^{(k)}$. Using the nearest neighbours, the map is considered trustworthy if these k neighbours are also placed close to point i in the low-dimensional space. The corresponding formula is [73]:

$$T(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in U_i^{(k)}} (r(i,j) - k) \tag{3.13}$$

It's a score between 0 and 1, where 1 indicates a more trustworthy embedding for that particular value of k, while 0 that the embedding does not faithfully represent the original data in terms of rank order.

For each combination of parameters we calculate the trustworthiness for k in the range of 1 to 2000, to see how the structure of the data is preserved across a wide range of neighborhood sizes. The choice of evaluating the embedding until a value of 2000 was made since the dataset consists of around 6000 points, and a value of 2000 means that also the long-range rank orders are taken into account. The best parameter combination is chosen as the one with the highest mean trustworthiness. For the *n_components* parameter it was chosen to be 1 for numerical and 2 for categorical variables. So effectively we reduce the dimensionality from 2 to 1 for numerical variables, and from 3 to 2 for categorical ones.

In total 90 different combinations were tested for numerical and also 90 for categorical variables.

Figure 3.8: Heat map of Mean Trustworthiness for different parameter combinations; euclidean and manhattan distance for numerical variables, hamming and jaccard for categorical variables. Brighter colours indicate a higher mean trustworthiness

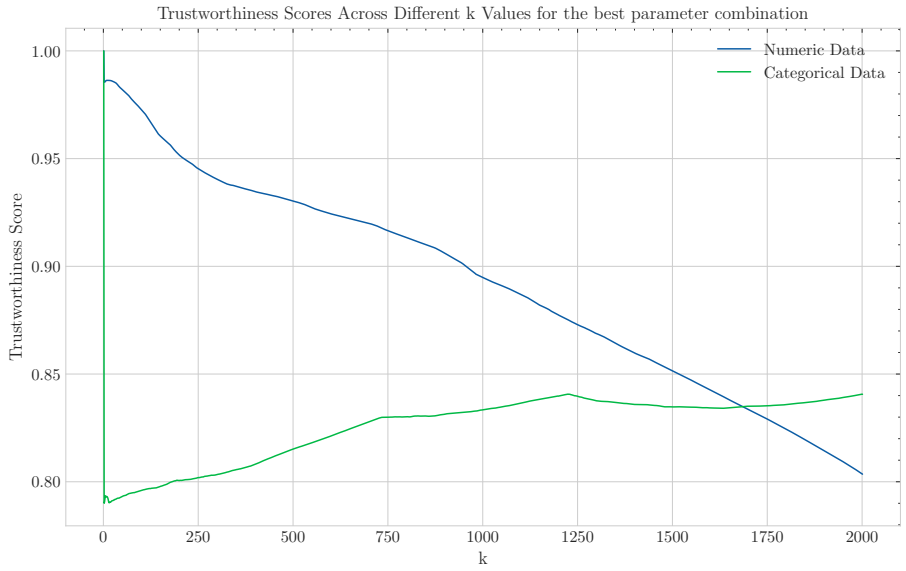


Figure 3.9: Trustworthiness plot for categorical and numerical variables for different values of k for the best parameters

Figure 3.8 tells us that the best parameter combination for numerical features was n_neighbors of 300, min_dist of 0.1 and the euclidean metric, for a mean trustworthiness of 0.89. For categorical variables it was n_neighbors of 15, min_dist of 0.1 and the jaccard metric for a mean trustworthiness of 0.83. Generally speaking a score above 0.80 is considered to be a "good" result[66].

Figure 3.9 shows that the trustworthiness decreases for numerical data while it increases for categorical data, as the size of the neighborhood increases. This suggests that categorical data may require a larger k to adequately reveal the underlying relationships, while for numerical data short-range interactions are better represented.

When it comes to HDBSCAN instead we look at **Density Based Clustering Validation** (DBCV)[74]. Metrics such as Silhouette Score[75] measure cluster cohesiveness and separation with an index between -1 and 1, but do not take into account noise in the index calculation and make use of distances. We know that distance is not applicable for a density based technique[66] and not including noise in the objective metric calculation violates an inherent assumption in density-based clustering. This means that Silhouette score and similar indexes are inappropriate for density-based techniques. DBCV instead takes noise into account and also uses densities instead of distances. The final result of DBCV is a weighted sum of "validity index" values of clusters, which produces a score between -1 and 1.

In practice a score above 0.45 shows that the clusters are well-separated. While we tried a wide range of parameters, the best parameters are a value of min_samples equal 300, min_cluster_size equal 100, cluster_selection_method is excess of mass and the metric euclidean. The DBCV score for this combination was 0.65. Excess of mass identifies clusters by looking at areas where data points are heavily concentrated and marks these as clusters.

We can now look at the results obtained from the clustering combining UMAP and HDBSCAN. A condensed tree plot lets you see what the cluster hierarchy looks like, i.e. which clusters are near each other or could perhaps be merged, and which are far apart.

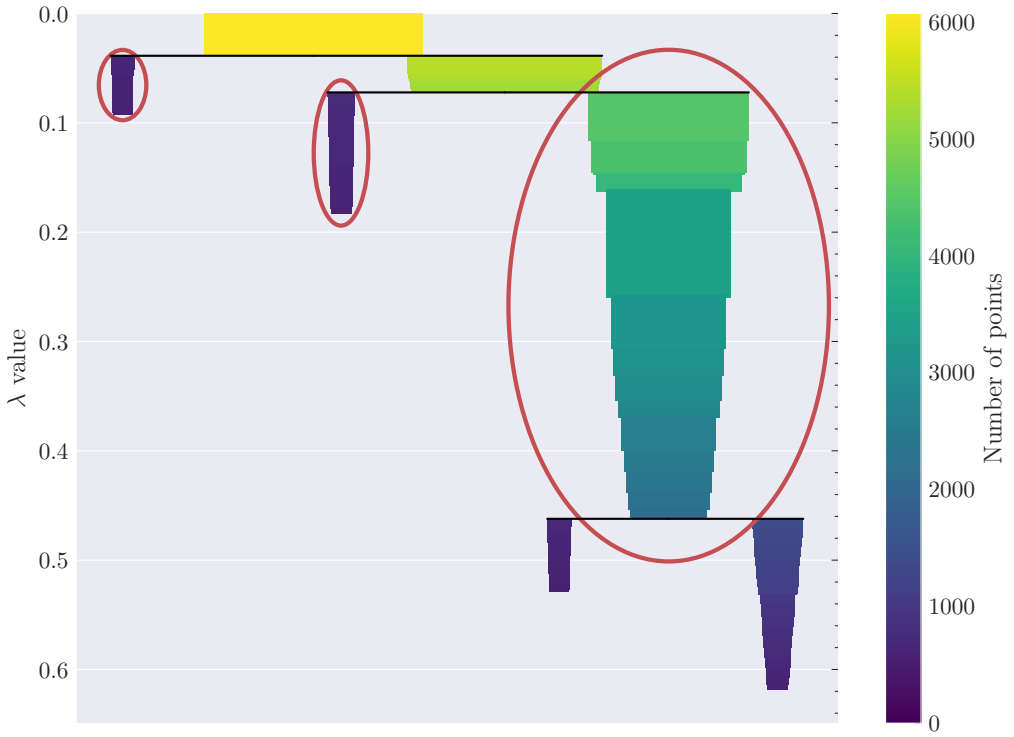


Figure 3.10: Condensed Tree plot of the clustering results; we can see the cluster hierarchy as a dendrogram. The width and colour of each branch represents the points in the cluster at that level. The encircled nodes correspond to the chosen clusters

The λ value in 3.10 represents a measure of density at which points are considered to merge together into a single cluster. A higher value for an edge means that the merge happens at a higher density level. The range of λ values over which a cluster exists in the condensed tree can give insights into the cluster's stability. A cluster that exists over a wide range of λ values is typically more significant and stable, indicating a robust cluster that persists over varying

densities. The dendrogram in 3.10 shows that HDBSCAN recognized three clusters of varying size.

Cluster	Users	Users Who Bought	% of Purchases in Cluster	CR (%)	Interactions		Session Time (s)	
					Mean	Median	Mean	Median
0	660	16	9.94	2.42	1.78	2	9.89	6
1	758	28	17.39	3.69	56.62	25	1129.58	101.5
2	4452	117	72.67	2.63	31.5	14	733.54	41

Table 3.3: HDBSCAN Cluster Statistics Including User Purchases, Conversion Rates (CR), and Interaction/Session Time

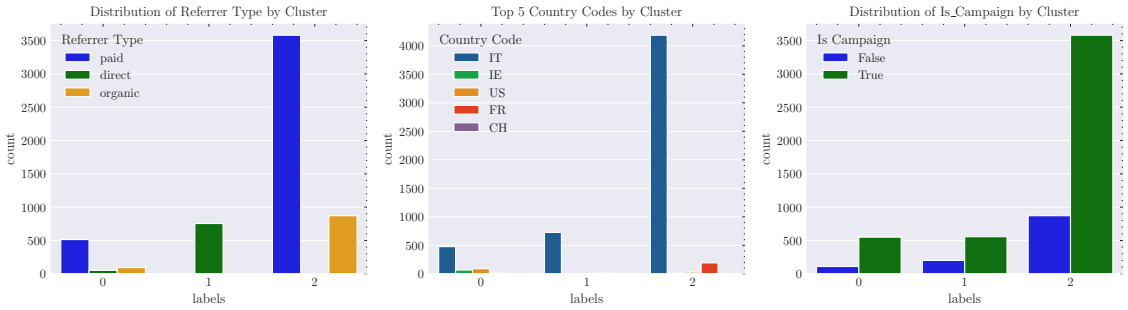


Figure 3.11: HDBSCAN Cluster statistics Including Referrer Type, Country Code And Campaigns

Looking at table3.3 we can see that HDBSCAN recognized one large cluster C2 which has the most amount of users and a mean/median interactions and session time which are values between the ones of the other two clusters. The conversion rate for this cluster is also a value between the two other. Meanwhile cluster C1 has the highest conversion rate, mean/median interactions and session time while having less data points than cluster C2. We can conclude that this is the cluster with the highest buying propensity. Finally cluster C0 clearly has the lowest buying propensity, reflected by having the lowest conversion rate and mean/median interaction and session time.

Thus, density-based clustering has given the following classification in terms of buying propensity: $\boxed{\text{Cluster1} \succ \text{Cluster2} \succ \text{Cluster0}}$.

Meanwhile, from 3.11 we can see the user characteristics in the individual clusters: the most successful cluster C1 contains mostly users from Italy coming from campaigns and have a *referrer type* of direct. The second most successful cluster C2 contains a wide range of users coming both from campaigns and not, and also from Italy and other countries like France. It also is the only cluster having a mix of users coming from *referrer type* organic and paid. Lastly, C0 is very similar to C1 in terms of countries where users come from and campaign provenance. They seem to only differ in the referrer type.

We conclude by saying that HDBSCAN successfully clustered users to create meaningful clusters, each with users having different characteristics in terms of buying propensity, reflected not only by the conversion rates but also by the varying levels of total interactions and session times.

3.5 Comparison Of The Methods

It is clear by now that both clustering methods, K-prototypes and HDBSCAN, produced meaningful clusters. These clusters indeed have users with different buying propensities. While both methods were successful, the results from K-prototypes appear to have created clusters which are better separated in terms of buying propensity. This is because the cluster with the highest conversion rate (10.46%) for K-prototypes is also the one containing the most amount of users who bought, which is not the case for the one of HDBSCAN. It seems like HDBSCAN focused more on producing clusters by looking at the numerical features, since these display larger intercluster discrepancies compared to the clusters of K-prototypes; we have that the lowest median inter-cluster session time is 21 while the highest is 90 for K-prototypes, while for HDBSCAN we have 6 and 101.5 respectively. Therefore we can also conclude that K-prototypes focused more on producing clusters better separated in terms of the categorical features; this can be observed by the *referrer type* variable being clearly separated between clusters, each cluster mostly containing only one type, but also from the campaign variable (although not as obvious compared to the referrer type). Additionally, HDBSCAN doesn't cluster all data points in clusters, since it can also assign points as being "noise" points, with a label of -1, which is why there are less points clustered in the output of HDBSCAN compared to the one of K-prototypes. In the case of HDBSCAN, each point assigned to a cluster also has a probability of belonging to that cluster. This information is not present when it comes to K-prototypes.

To conclude we can say that both methods have their advantages. K-prototypes was built to incorporate categorical features while the combination of UMAP and HDBSCAN seemed to have struggled more with those and gave meaningful clusters based on mostly the numerical features. On the other hand, HDBSCAN can provide more information about a clustering, such as points being labeled as "noise" and probabilities of points belonging to clusters.

4
Conclusion

This thesis embarked in the world of machine learning applied to e-commerce, tackling the challenges of website optimization and faulty behaviour detection through the calculation of page values using Markov Chains. Furthermore, clustering was applied to classify users into clusters having different buying propensities. The results obtained can now be leveraged to monitor the value of web pages in e-commerce websites, enabling potential issues to be discovered related to website performance, which could previously not be found. Moreover, the obtained clusters can now be used to produce the missing revenue and personalized recommendations to users.

Machine learning offers a powerful boost to e-commerce, optimizing websites and classifying user behaviors effectively. Businesses should leverage these insights for enhanced customer experiences and revenue growth, continually exploring new technologies to stay competitive in the digital marketplace.

References

[1] S. Amin, "A review paper on e-commerce," vol. 6, pp. 16–21, 06 2016.

[2] X. Zhang, F. Guo, T. Chen, L. Pan, G. Beliakov, and J. Wu, "A brief survey of machine learning and deep learning techniques for e-commerce research," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 18, pp. 2188–2216, 12 2023.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth Publishing Group, 1984.

[5] T. Ho, "Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, vol. 1, Montreal, QC, Canada, 14–16 August 1995, pp. 278–282.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[7] D. Hand and K. Yu, "Idiot's bayes—not so stupid after all?" *Int. Stat. Rev.*, vol. 69, pp. 385–398, 2001.

[8] J. Cramer, "The origins of logistic regression," University of Amsterdam and Tinbergen Institute, Amsterdam, The Netherlands, Tech. Rep., 2002.

[9] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 24, p. 4171, 1933.

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, pp. 30–37, 2009.

[11] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 13, 2000, paper available online.

[12] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, pp. 21–27, 1967.

[13] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[14] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193–202, 1980.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, vol. arXiv:1409.1556, 2014.

[16] C. Lea, R. Vidal, A. Reiter, and G. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Proceedings of the 2016 Computer Vision Workshops*, Springer, Ed. Amsterdam, The Netherlands: Springer, Oct. 2016, pp. 47–54.

[17] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, 1997.

[19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv*, vol. arXiv:1406.1078, 2014.

[20] B. Agarwal, R. Nayak, N. Mittal, and S. Patnaik, *Deep Learning-Based Approaches for Sentiment Analysis*. Berlin/Heidelberg, Germany: Springer, 2020.

[21] J. Gope, T. Tabassum, M. Mabrur, K. Yu, and M. Arifuzzaman, "Sentiment analysis of amazon product reviews using machine learning and deep learning models," in *Proceedings of the 2022 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*. Gazipur, Bangladesh: IEEE, 24–26 February 2022, pp. 1–6.

[22] A. Sheikh, R. Guigourès, E. Koriagin, Y. Ho, R. Shirvany, R. Vollgraf, and U. Bergmann, "A deep learning system for predicting size and fit in fashion e-commerce," in *Proceedings*

of the 13th ACM Conference on Recommender Systems. New York, NY, USA: ACM, Sep. 2019, pp. 110–118.

[23] E. Elmurngi and A. Gherbi, "An empirical study on detecting fake reviews using machine learning techniques," in *Proceedings of the 2017 Seventh International Conference on Innovative Computing Technology (INTECH)*. Luton, UK: IEEE, 16–18 August 2017, pp. 107–114.

[24] X. Wang, X. Zhang, C. Jiang, and H. Liu, "Identification of fake reviews using semantic and behavioral features," in *Proceedings of the 2018 4th International Conference on Information Management (ICIM)*. Oxford, UK: IEEE, 25–27 May 2018, pp. 92–97.

[25] D. Baishya, J. Deka, G. Dey, and P. Singh, "Safer: Sentiment analysis-based fake review detection in e-commerce using deep learning," *SN Comput. Sci.*, vol. 2, pp. 1–12, 2021.

[26] T. Aono, "Fake review detection focusing on emotional expressions and extreme rating," *Assoc. Nat. Lang. Process.*, vol. 13, pp. 422–425, 2019.

[27] M. Ashraf, M. Abourezka, and F. Maghraby, "A comparative analysis of credit card fraud detection using machine learning and deep learning techniques," in *Digital Transformation Technology*. Berlin/Heidelberg, Germany: Springer, 2022, pp. 267–282.

[28] H. Najadat, O. Altiti, A. Aqouleh, and M. Younes, "Credit card fraud detection based on machine and deep learning," in *Proceedings of the 2020 11th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 7–9 April 2020, pp. 204–208.

[29] A. Dingli, V. Marmara, and N. Fournier, "Comparison of deep learning algorithms to predict customer churn within a local retail industry," *Int. J. Mach. Learn. Comput.*, vol. 7, pp. 128–132, 2017.

[30] S. Dhote, C. Vichoray, R. Pais, S. Baskar, and P. Mohamed Shakeel, "Hybrid geometric sampling and adaboost based deep learning approach for data imbalance in e-commerce," *Electron. Commer. Res.*, vol. 20, pp. 259–274, 2020.

[31] S. Agrawal, A. Das, A. Gaikwad, and S. Dhage, "Customer churn prediction modelling based on behavioural patterns analysis using deep learning," in *Proceedings of the 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam, Malaysia, 11–14 July 2018, pp. 1–6.

[32] D. Koehn, S. Lessmann, and M. Schaal, "Predicting online shopping behaviour from clickstream data using deep learning," *Expert Syst. Appl.*, vol. 150, p. 113342, 2020.

[33] A. Utku and M. Akcayol, "Deep learning based prediction model for the next purchase," *Adv. Electr. Comput. Eng.*, vol. 20, pp. 35–44, 2020.

[34] M. Dharshini and S. Vijila, "Survey of machine learning and deep learning approaches on sales forecasting," in *Proceedings of the 2021 4th International Conference on Computing and Communications Technologies (ICCCT)*, Chennai, India, 16–17 December 2021, pp. 59–64.

[35] H. Zhu, "A deep learning based hybrid model for sales prediction of e-commerce with sentiment analysis," in *Proceedings of the 2021 2nd International Conference on Computing and Data Science (CDS)*, Stanford, CA, USA, 19–20 January 2021, pp. 493–497.

[36] T. Zahavy, A. Krishnan, A. Magnani, and S. Mannor, "Is a picture worth a thousand words? a deep multi-modal architecture for product classification in e-commerce," in *Proceedings of the 2018 AAAI Conference on Artificial Intelligence*, vol. 32, New Orleans, LA, USA, 2–7 February 2018.

[37] J. Dai, T. Wang, and S. Wang, "A deep forest method for classifying e-commerce products by using title information," in *Proceedings of the 2020 International Conference on Computing, Networking and Communications (ICNC)*, Big Island, HI, USA, 17–20 February 2020, pp. 1–5.

[38] B. Samia, Z. Soraya, and M. Malika, "Fashion images classification using machine learning, deep learning and transfer learning models," in *Proceedings of the 2022 7th International Conference on Image and Signal Processing and Their Applications (ISPA)*, Mostaganem, Algeria, 8–9 May 2022, pp. 1–5.

[39] A. Dingli, V. Marmara, and N. Fournier, "Comparison of deep learning algorithms to predict customer churn within a local retail industry," *Int. J. Mach. Learn. Comput.*, vol. 7, pp. 128–132, 2017, [CrossRef].

[40] S. Dhote, C. Vichoray, R. Pais, S. Baskar, and P. Mohamed Shakeel, "Hybrid geometric sampling and adaboost based deep learning approach for data imbalance in e-commerce," *Electron. Commer. Res.*, vol. 20, pp. 259–274, 2020, [CrossRef].

[41] M. Pondel, M. Wuczyński, W. Gryncewicz, M. Łysik, M. Hernes, A. Rot, and A. Kozina, "Deep learning for customer churn prediction in e-commerce decision support," in *Proceedings of the 24th International Conference on Business Information Systems*, Hannover, Germany, 14–17 June 2021, pp. 3–12.

[42] A. Vieira, "Predicting online user behaviour using deep learning algorithms," *arXiv*, vol. arXiv:1511.06247, 2015.

[43] D. Petrosanu, A. Pîrjan, G. Căruţaşu, A. Tăbuşcă, D. Zirra, and A. Perju-Mitran, "E-commerce sales revenues forecasting by means of dynamically designing, developing and validating a directed acyclic graph (dag) network for deep learning," *Electronics*, vol. 11, p. 2940, 2022.

[44] T. Zahavy, A. Krishnan, A. Magnani, and S. Mannor, "Is a picture worth a thousand words? a deep multi-modal architecture for product classification in e-commerce," in *Proceedings of the 2018 AAAI Conference on Artificial Intelligence*, vol. 32, New Orleans, LA, USA, 2–7 February 2018, p. unknown.

[45] L. Jiming, Z. Peixiang, L. Ying, Z. Weidong, and F. Jie, "Summary of multi-modal sentiment analysis technology," *J. Front. Comput. Sci. Technol.*, vol. 15, p. 1165, 2021.

[46] Q. Zhang and S. Zhu, "Visual interpretability for deep learning: A survey," *Front. Inf. Technol. Electron. Eng.*, vol. 19, pp. 27–39, 2018.

[47] Z. Yang, J. Zhang, Z. Zhao, Z. Zhai, and X. Chen, "Interpreting network knowledge with attention mechanism for bearing fault diagnosis," *Appl. Soft Comput.*, vol. 97, p. 106829, 2020.

[48] I. Nielsen, D. Dera, G. Rasool, R. Ramachandran, and N. Bouaynaya, "Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks," *IEEE Signal Process. Mag.*, vol. 39, pp. 73–84, 2022.

[49] S. Degirmenci and C. Jones, "Benchmarking offline reinforcement learning algorithms for e-commerce order fraud evaluation," *arXiv*, vol. arXiv:2212.02620, 2022.

[50] S. Bhoir and S. Patil, "Transfer learning with deep neural networks for image classification in the e-commerce industry," in *Proceedings of the 2022 IEEE 7th International conference for Convergence in Technology (I2CT)*, Pune, India, 7–9 April 2022, pp. 1–8.

[51] M. Rakhra, G. Gopinadh, N. Addepalli, G. Singh, S. Aliraja, V. Reddy, and M. Reddy, "E-commerce assistance with a smart chatbot using artificial intelligence," in *Proceedings of the 2021 2nd International Conference on Intelligent Engineering and Management*, London, UK, 28–30 April 2021, pp. 144–148.

[52] A. Landim, A. Pereira, T. Vieira, E. de B. Costa, J. Moura, V. Wanick, and E. Bazaki, "Chatbot design approaches for fashion e-commerce: An interdisciplinary review," *Int. J. Fash. Des. Technol. Educ.*, vol. 15, pp. 200–210, 2022.

[53] E. Anderl, I. Becker, F. V. Wangenheim, and J. H. Schumann, "Mapping the customer journey: A graph-based framework for online attribution modeling," SSRN, October 2014, available at SSRN: https://ssrn.com/abstract=2343077 or http://dx.doi.org/10.2139/ssrn.2343077.

[54] PyPI, "Channelattribution: Markov model for online multi-channel attribution," 2021, uRL: https://pypi.org/project/ChannelAttribution/ (accessed on September 30, 2021).

[55] F. Author and S. OtherAuthor, "Mining advertiser-specific user behavior using adfactors," in *Proceedings of the 19th International Conference on World Wide Web, WWW 2010*. Raleigh, North Carolina, USA: ACM, 4 2010.

[56] C. Hennig, "What are the true clusters?" 2015.

[57] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery*, vol. 2, pp. 283–304, 1998. [Online]. Available: https://doi.org/10.1023/A:1009769707641

[58] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020.

[59] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. [Online]. Available: https://doi.org/10.21105/joss.00205

[60] F. Safieddine, *M-Commerce*, 11 2016.

[61] scikit-learn developers, "Permutation importance," https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html, 2023, accessed: 2024-03-28.

[62] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1. University of California Press, 1967, pp. 281–297.

[63] M. Carreira-Perpiñán and W. Wang, "The k-modes algorithm for clustering," 2013.

[64] P. Trebuňa, J. Halčinová, M. Fil'o, and J. Markovič, "The importance of normalization and standardization in the process of clustering," in *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2014, pp. 381–385.

[65] R. J. G. B. Campello, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *WIREs Data Mining and Knowledge Discovery*, vol. 10, no. 2, p. e1343, 2020. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1343

[66] AWS Labs, "Amazon DenseClus: Density-Based Clustering for Machine Learning," https://github.com/awslabs/amazon-denseclus, 2023, accessed: 2024-03-30.

[67] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[68] G. Stewart and M. Al-Khassaweneh, "An implementation of the hdbscan* clustering algorithm," *Applied Sciences*, vol. 12, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/5/2405

[69] G. E. P. Box and D. R. Cox, "An analysis of transformations," *Journal of the royal statistical society series b-methodological*, vol. 26, pp. 211–243, 1964. [Online]. Available: https://api.semanticscholar.org/CorpusID:15192218

[70] M. A. Stephens, "EDF statistics for goodness of fit and some comparisons," *Journal of the American Statistical Association*, vol. 69, pp. 730–737, 1974. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1974.10480196

[71] S. Demir, "Comparison of normality tests in terms of sample sizes under different skewness and kurtosis coefficients," *International Journal of Assessment Tools in Education*, vol. 9, p. 397–409, 2022.

[72] J. Venna and S. Kaski, "Local multidimensional scaling," *Neural Netw.*, vol. 19, no. 6, p. 889–899, jul 2006. [Online]. Available: https://doi.org/10.1016/j.neunet.2006.05.014

[73] S. Stasis, R. Stables, and J. Hockman, "Semantically controlled adaptive equalisation in reduced dimensionality parameter space," *Applied Sciences*, vol. 6, p. 116, 04 2016.

[74] D. Moulavi, P. Andretta Jaskowiak, R. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," 04 2014.

[75] P. Rousseeuw, "Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 11 1987.