UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Amanda Fagundes de Paula

Behavioral Analysis and Bot Detection in E-Commerce Navigation Data

Belo Horizonte
2025

Amanda Fagundes de Paula

Behavioral Analysis and Bot Detection in E-Commerce Navigation Data

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Fabricio Murai Ferreira

Belo Horizonte
2025

2025, Amanda Fagundes de Paula.
Todos os direitos reservados

Paula, Amanda Fagundes de.

P324b Behavioral analysis and bot detection in E-Commerce navigation data [recurso eletrônico] / Amanda Fagundes de Paula - 2025.
1 recurso online (84 f. il., color.) : pdf.

Orientador: Fabrício Murai Ferreira.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 81-84.

1. Computação – Teses. 2. Classificação (Computadores) – Teses. 3. Comercio eletrônico – Medidas de Segurança – Teses. 4. Usúarios da internet – Teses. 5. Markov, Processos de - Teses I. Ferreira, Fabrício Murai. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Behavioral Analysis and Bot Detection in E-Commerce Navigation Data

AMANDA FAGUNDES DE PAULA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. FABRÍCIO MURAI FERREIRA - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. ANA PAULA COUTO DA SILVA
Departamento de Ciência da Computação - UFMG

PROFA. GISELE LOBO PAPPA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 25 de fevereiro de 2025.

To my family and friends, who have always supported and endured me throughout this ~~extremely~~ long journey. To all Brazilian researchers who persevere despite all adversity.

Acknowledgments

There is no better way to begin than by thanking my parents, Rita and José Carlos, who, despite all challenges, always supported my studies and encouraged me to follow this path, even when I couldn't fully realize its true significance. Life may not have allowed them to walk this path themselves, but it never took away their wisdom, which I've always been fortunate to rely on. I also thank my younger brother, Rafael, for showing me that I can positively impact others and for always cheering me on.

I may not have many friends, but I am extremely grateful for those I have. Ana Rita, Aline, Camila Bicalho, Elisa, Luiz Fernando, and Gabriel were essential in helping me stay strong, and I truly feel blessed because you were exactly what I needed in the different "seasons" of my life.

Despite not following any specific belief, I have faith in my dreams because I know I am blessed. It took time to see it, but now that I do, I feel nothing but gratitude and a unfamiliar, comforting peace.

Since primary school, I have been fortunate to be guided by incredible teachers who sparked my curiosity and love for learning. During my undergraduate years, I was mentored by amazing professors. Thank you, Glauber, for that unforgettable first programming class; Alexandre Magno, for believing in me during what became known as the "worst semester ever"; and Leonardo Reis, my advisor, who guided my research beyond being a great professor.

In my Master's, I was taught by professors who truly impressed me. I recall feeling unexpectedly emotional in a Complex Networks class (surprisingly so for a class about disease and fake news spreading modeling) taught by Ana Couto, Fabrício Murai, and Jussara Almeida. That moment made me realize I wasn't fully taking advantage of the academic atmosphere, which led me to step away and return at a better time. I am deeply grateful to Fabrício for his continued support and patience, especially during his transition to another university.

I also want to thank my colleagues and managers at VTEX, especially Ericsson Lima, Thaynan Nunes, and Andre Silveira, who provided all the support I needed to complete this step, as well as enabling me to use VTEX data in this research.

This is the last piece of text I have written in this dissertation, and I do so with my heart full of gratitude for all that has brought me here. Thank you all!

"*It's the questions we can't answer that teach us the most.*"
(Patrick Rothfuss, The Wise Man's Fear)

Resumo

Compreender o comportamento dos usuários em plataformas de e-commerce é fundamental para aprimorar a experiência do cliente e identificar atividades anômalas, como interações realizadas por bots. Este estudo realizou uma análise exploratória de dados (EDA) abrangente, examinando informações de navegação e revelando diferenças significativas em atributos de sessão, padrões de transição e dinâmicas comportamentais. Esses insights oferecem uma base sólida para o estudo sistemático de comportamentos legítimos e anômalos em contextos de compras online.

Como etapa complementar à análise exploratória, utilizamos um modelo de One-Class Support Vector Machine (OC-SVM) para detectar anomalias nos dados de sessão. Um subconjunto de exemplos rotulados foi usado para ajustar os hiperparâmetros e selecionar as configurações ideais do modelo, reforçando a validade dos padrões comportamentais identificados na EDA.

Com o uso do Fβ-Score, onde $\beta = 0.5$, como métrica de desempenho, os experimentos mostraram que os kernels lineares e polinomiais de baixa ordem apresentaram bons resultados no equilíbrio entre precisão e recall. Os resultados indicaram que sessões anômalas tendem a exibir padrões de navegação dominados por transições "altamente prováveis", refletindo o poder discriminativo das features projetadas.

Posteriormente, o modelo otimizado foi aplicado para prever as classes de sessões anteriormente não rotuladas, ampliando o conjunto de dados com novas classificações. Essas previsões permitiram uma análise mais profunda das diferenças entre comportamentos anômalos e legítimos, revisitando análises-chave e explorando novos insights no conjunto de dados expandido. Essa etapa complementar destaca o valor prático dos achados da EDA no suporte à detecção de anomalias em e-commerce.

Este estudo reforça a importância de uma exploração robusta de dados na detecção de anomalias, demonstrando como uma EDA bem estruturada pode informar o design e a interpretação de modelos. Os resultados fornecem insights valiosos sobre o comportamento dos usuários e oferecem uma abordagem sistemática para a detecção de bots em ambientes de e-commerce.

Palavras-chave: cadeia de Markov; classificadores de uma classe; detecção de bots; e-commerce.

Abstract

Understanding user behavior on e-commerce platforms is crucial for enhancing customer experience and identifying anomalous activities, including bot interactions. This research conducts a comprehensive exploratory data analysis (EDA) to examine navigation data, revealing significant differences in session attributes, transition patterns, and behavioral dynamics. These insights create a foundation for systematically studying legitimate and anomalous user behaviors in online shopping contexts.

As a complementary step to the exploratory analysis, we utilized a One-Class Support Vector Machine (OC-SVM) to detect anomalies in session data. A subset of labeled examples guided the fine-tuning process, enabling the selection of optimal model configurations and reinforcing the validity of the behavioral patterns identified in the EDA.

Using the Fβ-Score with $\beta = 0.5$ as a performance metric, the experiments highlighted that linear and low-degree polynomial kernels performed well in balancing precision and recall. The results indicated that anomalous sessions tend to exhibit navigation patterns dominated by "highly likely" transitions, reflecting the engineered features' discriminative power.

Subsequently, the optimized model was used to predict the classes of previously unlabeled sessions, extending the dataset with new classifications. These predictions enabled a deeper examination of the distinctions between anomalous and legitimate behaviors by revisiting key analyses and exploring additional insights within the newly labeled dataset. This complementary step underscores the practical value of the EDA findings in supporting anomaly detection in e-commerce contexts.

This research underscores the importance of robust data exploration in anomaly detection, demonstrating how well-structured EDA can inform model design and interpretation. The findings provide valuable insights into user behavior and offer a systematic approach to bot detection in e-commerce environments.

Keywords: Markov chain; one class classifiers; one class SVM; bot detection; e-commerce.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

Over the past years, the Internet has experienced exponential growth, leading to an unprecedented increase in the number of individuals connected to the digital environment. However, as technology and internet access mechanisms have evolved, the volume of invalid traffic has significantly increased. This type of traffic, often driven by automated scripts and non-legitimate users, poses challenges for e-commerce platforms by distorting engagement metrics, impacting performance, and introducing potential security risks. According to a Wired research [21], around 40% of all online web traffic is invalid, making it crucial to develop mechanisms to detect their activities and prevent damage. This is in line with the "Dead Internet Theory", which suggests that bots and AI-generated content are increasingly dominating the online interactions [1]. Su et al. [34] considers bots to be one of the most severe threats to Internet security in recent years as when compared with other malware like viruses and worms; bot behavior can be very stealthy, making their detection extremely difficult.

Every day, web bots are responsible for generating a substantial amount of web traffic and are used for the most varied purposes. The most prevalent bots are search engine crawlers, which routinely access web pages to construct and update extensive search indexes [4, 15]. From a cybersecurity perspective, they can be classified into malicious and benign bots, depending on how they operate on the Internet.

Benign bots typically perform tasks like website indexing and monitoring. They are expected to inform the server of their identity by including the bot's name in the user agent of each HTTP request and to comply with the rules outlined in the *robots.txt* file—a text file stored in the root directory of a website that specifies which pages can be crawled. Usually, benign bots are operated by search engines and SEO (search engine optimization) agencies and should be allowed to crawl the data necessary to improve the website's performance.

Although benign bots often serve beneficial purposes, their presence can still pose challenges for the website's maintainers. Excessive crawling by search engine bots, for instance, can lead to increased server load, bandwidth consumption, and potential performance degradation. Moreover, the substantial bandwidth usage, particularly on content-rich websites, may impact loading speeds for legitimate users, adversely affecting the

overall user experience. Furthermore, the presence of benign bots can add noise to the website analytics data, making it difficult to accurately interpret user behavior and potentially leading to erroneous conclusions. In practice, many robots are camouflaged and may be detected on the server only by more sophisticated methods.

Otherwise, malicious bots pose a multifaceted threat, as they can be utilized for a diverse array of purposes, ranging from data scraping and credential stuffing to orchestrating large-scale cyberattacks. In [34], the authors investigate how malicious actors can exploit crowdsourcing platforms to manipulate search engine rankings by providing detailed instructions on how to perform queries that bypass the detection mechanisms.

According to the 2023 Imperva Bad Bot Report [13], bots were responsible for 47.2% of the internet traffic in 2022. This number highlights the alarming prevalence of malicious bot activities across the digital landscape, emphasizing the urgent need for proactive cybersecurity measures to safeguard online platforms.

E-commerce portals are highly susceptible to both malicious and benign bot activities. Storing sensitive personal and financial data makes these platforms attractive targets for unauthorized operations. Additionally, product catalog data is often exploited by crawler agents for purposes like price comparison or content scraping for use by competing businesses [25]. According to a Distil Network Report [14], almost two-thirds (61.8%) of attacks that targeted online retailers were classified as automated threats, highlighting the potential of these automated agents on major online platforms.

E-commerce platforms comprise multiple interdependent components, ranging from user authentication and payment systems to databases, web applications, and network infrastructure. Each of these components is vulnerable to numerous cyber attacks [26], that can be exploited by malicious actors, posing a significant threat to the confidentiality, integrity, and availability of sensitive data and transaction processes. As the e-commerce landscape continues to evolve, the sophistication and prevalence of cyber threats also increase. This makes it imperative to acknowledge and proactively address potential risks associated with malicious activities.

Even benign bots like search engine crawlers periodically scrape websites, increasing the system's workload. While these activities are not inherently harmful, they still consume cloud resources without providing direct benefits to the e-commerce platform. This can lead to increased workloads and operational costs. Moreover, if traffic data generated by these automated agents is not properly filtered, it can distort interpretations of user preferences and behaviors, leading to flawed analytics and misguided decision-making. Therefore, the first step in mitigating the impact of both benign and malicious bot activities is to distinguish their behavior from legitimate user interactions—without initially focusing on specific bot types. Effectively identifying and managing bot traffic, whether harmful or not, is crucial for maintaining accurate analytics and optimizing resource usage in e-commerce operations.

Recent studies have investigated different approaches to bot detection in e-commerce. For example, [30] analyzed session-based differences between bots and humans in web stores, employing unsupervised learning techniques to identify anomalous behaviors. Suchacka et al. [35] introduced real-time detection methods using sequential analysis and neural networks to enhance early identification of bots. Additionally, [21] highlighted the broader implications of bot activity in corrupting online ecosystems, emphasizing the need for robust detection mechanisms to mitigate these threats.

While these studies have advanced the field of bot detection, many rely heavily on session-level features or focus predominantly on static data from web server logs. This approach can miss temporal and behavioral patterns that emerge during navigation. To address this, the present study leverages navigation event data to extract valuable features such as session duration, sequence probabilities, transition frequencies, and access origins. By modeling the behavior of legitimate agents, informed by the insights from our analysis, our approach provides a robust framework for distinguishing between malicious and legitimate activities.

A common challenge reported in related studies is the lack of labeled data, which often limits the ability to validate solutions in realistic scenarios. Our study addresses this issue by thoroughly characterizing both invalid and legitimate activities through an extensive behavioral analysis, providing a solid foundation for modeling these behaviors. By extracting the pages accessed during navigation and ordering them chronologically, we reconstructed the agents' navigation paths within a session. This allowed us to model their behavior using a probabilistic approach—computing page transition probabilities and leveraging Continuous Time Markov Chains (CTMCs) to capture these patterns.

Beyond offering a structured representation of user and bot behaviors, our approach enables the creation of synthetic datasets that preserve key statistical properties of real-world navigation patterns. These datasets can be instrumental in training and evaluating models when labeled data is scarce. Additionally, by leveraging CTMCs, we provide a dynamic framework that can be adapted to different e-commerce environments, facilitating the detection of evolving bot strategies. We believe these findings contribute significantly to the field, supporting future research in bot detection, anomaly detection, and user behavior modeling.

1.1 Motivation

In today's digital landscape, bot activities are increasingly prevalent, mimicking human behavior in various domains. While automation offers certain advantages, we

believe it's crucial to preserve the authenticity and dynamic nature of human interactions. In many contexts, genuine human engagement is essential for fostering trust, driving innovation, and ensuring ethical and equitable outcomes.

E-commerce provides a prime example of this critical need. Understanding the nuanced behaviors and preferences of real shoppers is fundamental to successful online businesses. Events like search queries, click patterns, and purchase histories offer invaluable insights into consumer needs and desires. These insights inform crucial business decisions, from product development and marketing strategies to customer service and personalized recommendations.

The proliferation of bots can significantly distort valuable data. Malicious agents can manipulate search results, artificially inflate website traffic, and even engage in fraudulent transactions. This not only undermines the integrity of online businesses but also misleads businesses into catering to non-existent or even harmful demands.

Therefore, it is imperative to develop robust methods for detecting and mitigating the impact of bot activities in e-commerce. By accurately identifying and filtering out bot-generated data, businesses can gain a more accurate and reliable picture of their true customer base. This, in turn, enables them to make informed decisions, enhance customer experiences, and ultimately achieve greater success in the competitive online marketplace.

1.2 Specific objectives

Given the importance of modeling legitimate user behavior and understanding how malicious agents act in e-commerce environments, the purpose of this study is to explore features that can highlight the differences between bot and legitimate users. As a guide for the development of this study, we raise the following research questions:

Q1: Is there any significant difference between user and bot navigation patterns?

Q2: Can machine learning techniques designed for modeling "normal" data and trained on the user labeled data alone identify bots' activities as anomalous?

Q3: What are the most relevant features extracted from the navigation data that best discriminate between legitimate users and bots?

Q4: How do Continuous Time Markov Chains (CTMCs) contribute to the characterization and detection of anomalous navigation behaviors?

1.3 Overview of this work

We extract new features from raw data and use a Continuous Time Markov Chain models for modeling user behavior and then estimating the likelihood that a given sequence of events occurs. Analyzing the distribution of those features allowed us to understand better outliers in that context, which in turn led us to conduct additional experiments with a new, valuable feature: the frequency of most common transitions for each observed sequence.

Our study is performed on a new dataset, generously provided by VTEX [1]—a leading global e-commerce platform supporting a diverse range of merchants—, which offers a rich and representative sample of real e-commerce interactions. To safeguard the confidentiality of the participating merchants, all identifying information has been anonymized.

Our primary challenge in this study stems from the absence of fully-labeled data, which complicates the validation and evaluation processes. To label highly-likely bot activities, we relied on outputs from a basic Web Application Firewall (WAF) mechanism, capable of detecting only a limited subset of malicious behaviors. Conversely, we assumed that all users who completed a purchase at any point during the observed period are legitimate users.

After conducting the aforementioned labeling process, we extracted new attributes, particularly from the URLs and timestamp data available in the raw dataset in order to create new features. By introducing these features, we progressively enhanced the dataset, applying different aggregation techniques and adding new dimensions. Although the raw data holds significance, it is insufficient on its own to support an in-depth analysis of user and bot behaviors. The labeling process and feature engineering collectively guided our analysis, which is extensively detailed in Chapter 5. We believe these findings offer substantial value and hold potential to assist researchers in this field.

1.4 Outline

Next, we briefly summarize the structure of this study. Chapter 2 introduces prior studies focused on detecting anomalous behavior in the e-commerce context through various Machine Learning strategies, including Supervised, Semi-supervised, and Unsuper-

[1] https://vtex.com.br

vised Learning approaches (respectively in Sections 2.1, 2.2, and 2.3). Chapter 3 provides an overview of the core concepts and techniques underlying the methodology adopted in this research.

Chapter 4 details the Data and Experimental Setup. Section 4.1 describes the data collection process and the sampling strategies adopted, which were tailored to suit different analytical scopes. Section 4.2 presents the collected raw data and provides a summary of key metrics for individual merchants. Section 4.3 discusses the data pre-processing steps, including data validation, labeling strategies, and feature augmentation.

Chapter 5 explores the results of the Exploratory Data Analysis (EDA), delving into event data and the distinctions in behaviors between users and bots, derived from both raw and manually engineered features. Model Design implementation details are presented in Chapter 6. Study outcomes are presented in Chapter 7, which includes the evaluation of the proposed method and its effectiveness in detecting bot activities. Finally, Chapter 8 summarizes the key findings of this research, discusses the limitations of the proposed method, and outlines potential avenues for future work.

Chapter 2

Related Work

Bot detection in e-commerce and web environments is an ongoing challenge due to the increasing sophistication of automated agents that mimic legitimate user behavior. As bots become more advanced, their detection becomes more difficult, requiring new methods and technologies to keep abreast of threats. Bot detection techniques generally fall into three major categories: supervised learning, semi-supervised learning, and unsupervised learning. Each approach has its own set of advantages and limitations, and understanding these distinctions is crucial for selecting the appropriate method based on the available data and the specific requirements of the task at hand.

Supervised learning approaches have been widely used in bot detection due to their ability to leverage labeled data for model training. However, a major challenge lies in the availability of accurate labeled datasets, as acquiring labeled data often requires manual intervention and domain expertise, which is both time-consuming and costly [35, 11]. This limitation has prompted the exploration of semi-supervised learning techniques, which aim to reduce the dependency on labeled data by combining a small amount of labeled instances with a larger pool of unlabeled data [40, 15]. Such methods have shown promising results, particularly when the cost of obtaining labeled data is prohibitively high or when labeling is impractical.

Unsupervised learning, on the other hand, provides a powerful approach to bot detection, especially when labeled data is scarce or unavailable [3]. This approach focuses on identifying patterns, anomalies, or clusters within the data without requiring prior knowledge of the labels. Unsupervised methods, such as anomaly detection and clustering, have been effectively applied to detect bot activity based on behavioral irregularities or unusual patterns in user interactions [27, 42]. Moreover, the development of hybrid methods that combine unsupervised learning with supervised or semi-supervised components has further enhanced the detection accuracy, making unsupervised learning a valuable tool for large-scale detection systems.

In this chapter, we provide a comprehensive review of the literature on bot detection, organized according to these three learning paradigms. We discuss various methodologies, their applications, and the challenges they address. Specifically, the chapter is divided into three sections: Supervised Learning, Semi-supervised Learning, and Unsu-

pervised Learning. Each section will explore key works in the field, highlighting the advantages of each approach and the contexts in which they have been successfully applied.

The supervised learning approach remains dominant in bot detection research, with several studies focusing on the application of machine learning algorithms, such as decision trees, SVMs, and neural networks, to identify malicious activities based on labeled data [23, 34, 9]. However, the reliance on labeled data presents a significant challenge, which is where semi-supervised learning approaches have come into play, offering a middle ground by reducing the need for labeled data while still achieving high accuracy [7, 8]. In contrast, unsupervised learning methods are increasingly popular for their ability to detect novel bot behaviors without the need for labeled training sets, making them particularly useful for real-time bot detection in dynamic environments [38, 20].

By examining the latest advancements in each of these areas, this chapter aims to provide a detailed understanding of the state-of-the-art techniques and their applications in the field of bot detection.

2.1 Supervised Learning Bot Detection

Supervised learning approaches in bot detection require labeled data, where bot and legitimate activities are accurately identified and annotated. However, the acquisition of such labeled data is often resource-intensive, requiring manual intervention and domain expertise for validation.

Suchacka et al. [35] utilized publicly available databases, incorporating IP addresses, user agents, and browser information to generate labeled samples. A neural network with a logistic output was then trained to classify HTTP requests as either bot or human-generated. Similarly, Goseva-Popstojanova et al. [11] employed a honeypot system to collect and classify malicious activities using a combination of three classifiers: SVM, decision trees, and PART. Nguyen et al. [23] proposed Lino, a honeypot system simulating vulnerable web pages, to trap web crawlers and collect features for classification using machine learning methods such as SVM and decision trees.

Su et al. [34] manually annotated collected data and introduced an Activity Factor Graph Model (AFGM), which incorporates behavior, user, and item-related information to predict spamming activities, specifically “Add to favorites”. In contrast, Daya et al. [9] employed graph-based features like Betweenness Centrality and Local Clustering Coefficient to train classification models with labels generated by clustering methods.

Alahmadi et al. [2] explored the use of a Markov Chain model to represent bot net-

work behavior. By extracting high-level features and training a classifier, they successfully categorized bot activities and identified specific bot families. Kruegel et al. [17] also proposed an anomaly detection system that utilizes Bayesian analysis to examine web server logs, comparing HTTP query patterns to program-specific profiles for detecting web-based attacks.

In botnet detection, Narang et al. [22] combined clustering techniques with supervised machine learning algorithms, such as Random Forest and Decision Trees, to detect peer-to-peer (P2P) botnets.

Rasti et al. [28] discuss the application of deep learning methods, particularly recurrent neural networks (RNN), for enhancing bot detection in e-commerce platforms, where sequential behaviors are vital for accurate classification. Similarly, Liu et al. [18] applied ensemble learning, integrating decision trees, SVM, and neural networks, to improve the robustness and accuracy of bot detection systems, particularly in web applications where adversarial behaviors are often complex.

2.2 Semi-supervised Learning Bot Detection

Semi-supervised learning techniques are particularly useful when labeled data is scarce. These methods combine a small amount of labeled data with a larger set of unlabeled data to improve model performance. Xu et al. [40] proposed a semi-supervised approach using Expectation Maximization (EM) to identify suspicious Bot IP addresses (BIPs). By assuming that features follow a mixture of Gaussian distributions, they classified IPs based on their likelihood of being bot-related. In [15], semi-supervised learning methods were applied to distinguish between automated and genuine user activity in search engines.

Issues related to imbalanced datasets are addressed by Chawla et al. [7], who introduced techniques for generating synthetic minority class samples through interpolation between samples and their nearest neighbors in feature space. Robertson et al. [29] proposed an anomaly detection system that combines unsupervised learning to profile normal server behavior with supervised heuristics to classify anomalies as either bot-generated or legitimate.

Chwalinski et al. [8] combined clustering techniques to group similar HTML sequences, detecting attackers based on unlikely transitions within clusters. Similarly, Rovetta et al. [30] adopted clustering algorithms such as k-means and Graded Possibilistic c-Means for unsupervised grouping, followed by supervised labeling to classify legitimate and malicious activities in the e-commerce domain. Haq et al. [33] applied a hybrid ap-

proach combining k-means clustering with the J48 decision tree classifier, achieving high detection accuracy in a comparison to standalone methods.

Tran et al. [37] introduced deep semi-supervised learning methods, particularly focusing on autoencoders, to detect bots with minimal labeled data in e-commerce settings. This approach leverages unsupervised pre-training of neural networks to learn representations that are later fine-tuned with labeled data. Zhang et al. [41] explored a graph-based semi-supervised learning framework for bot detection, showing its effectiveness in large-scale network environments where conventional methods often struggle.

2.3 Unsupervised Learning Bot Detection

Unsupervised learning methods are increasingly popular for bot detection when labeled data is difficult or costly to obtain. These methods detect anomalies or clusters within data without predefined labels, identifying rare events or patterns that deviate significantly from the norm.

Anomaly detection, a form of unsupervised learning, focuses on identifying instances that differ significantly from normal data points [3]. While initially used for noise filtering, anomaly detection is now widely used for uncovering rare or malicious events. In e-commerce, unsupervised learning techniques can be employed for customer segmentation, grouping users based on behavior and characteristics. Rajput et al. [27] demonstrated the effectiveness of K-Means clustering in segmenting e-commerce customers by their communication preferences, improving marketing and customer retention efforts. Wang et al. [38] utilized clickstream similarity graphs to capture user behaviors, applying clustering techniques to identify distinct behavioral patterns.

In bot detection, unsupervised techniques can effectively identify anomalous user behavior. Rovetta et al. [30] explored a generative approach for characterizing bot sessions, using clustering methods such as K-means and Graded Possibilistic c-Means (GCPM). Their results showed the effectiveness of unsupervised learning in bot detection, outperforming some traditional supervised models. Similarly, Daya et al. [9] used clustering methods and graph-based features to distinguish between benign and malicious traffic, which were then used to train a classification model.

Zhao et al. [42] present a novel unsupervised learning framework that integrates clustering and anomaly detection for detecting bot behavior in financial transactions. Their approach leverages density-based spatial clustering methods to identify patterns in high-dimensional data, which is crucial for detecting fraud in financial networks. Lu et al. [20] also proposed a hybrid unsupervised approach combining density-based spa-

tial clustering and outlier detection to classify bot behavior in social media platforms, demonstrating its efficacy in handling large-scale datasets.

In this study, we adopt unsupervised learning methods, specifically a One-Class Support Vector Machine (OC-SVM), to detect anomalous behavior in e-commerce navigation data. This approach addresses the challenges of environments where accurately labeled data for bot detection is scarce or unreliable, as highlighted in studies such as [30, 35]. Unlike these works, which rely on labeled data or semi-supervised approaches to generate it, our methodology models legitimate user behavior directly. By doing so, it eliminates the dependency on extensive labeled datasets and instead leverages features derived from a comprehensive exploratory data analysis (EDA). This strategy enables the detection of subtle deviations from normal behavior, providing a scalable and robust solution for bot detection in e-commerce contexts.

Chapter 3

Technical Background

This chapter provides the theoretical background essential for understanding the methods employed in this study. By leveraging probabilistic models, such as Continuous-Time Markov Chains, we capture the temporal dynamics of user navigation and transitions between states. These insights, combined with the anomaly detection capabilities of One-Class Support Vector Machines, enable the identification of deviations from legitimate user behavior. Together, these approaches provide a robust framework for distinguishing bots from genuine users in e-commerce environments.

3.1 One-Class Classification

In scenarios involving severely imbalanced datasets, traditional binary or multi-class classification methods often result in a bias toward the majority class, making it challenging to accurately model and detect instances of the minority class. This imbalance is particularly problematic when the minority class is of greater interest, such as in fraud and anomaly detection, or rare event prediction. For instance, in the case of abnormal behavior detection in web navigation, normal user navigation patterns vastly outnumber anomalous behaviors such as bot activity or system faults. In such cases, anomaly detection becomes critical, and it is here that one-class classification (OCC) algorithms prove to be effective.

One-class classification methods are specifically designed for situations where the focus is on learning a model of the normal class, and the goal is to identify any deviations from this normal behavior as anomalies or outliers. These models do not rely on labeled instances of the abnormal class but instead learn the characteristics of the normal class, typically by modeling the distribution of normal data points [36]. Once the model is trained, any data point that falls outside the learned distribution is flagged as an anomaly. This approach is particularly advantageous in scenarios like bot detection, where labeled data for malicious agents may be sparse or unavailable. In such contexts, the model is

trained solely on data representing normal user behavior, and new instances are classified based on their similarity to this normal behavior. By identifying deviations from normal patterns, OCC methods can detect bot activity, even in the absence of explicit labels for malicious actions [31, 32].

Furthermore, the flexibility of OCC makes it applicable across a variety of domains beyond fraud and bot detection, such as intrusion detection, industrial system monitoring, and medical diagnosis. In either of these cases, the focus is on identifying rare or anomalous events, which could be indicative of significant problems, such as system failures or health risks. Recent advancements in OCC techniques have incorporated deep learning models, such as autoencoders, which have been shown to perform well in learning complex representations of normal behavior and detecting outliers based on reconstruction errors [31, 39]. Despite the challenges posed by the lack of labeled anomalous data, OCC techniques have demonstrated their effectiveness in detecting rare but important events in numerous high-stakes applications.

3.1.1 One-Class Support Vector Machine

The One-Class Support Vector Machine (OC-SVM) is a robust machine learning algorithm specifically designed for one-class classification tasks, such as anomaly detection and outlier identification. Based on the principles of Support Vector Machines (SVMs), the OC-SVM aims to learn a decision boundary that encapsulates the majority of data points from a single class representing "normal" behavior. Unlike traditional classifiers, One-Class Classifiers are trained exclusively on examples of this normal class.

During training, the algorithm attempts to construct the smallest hypersphere that encloses all – or most – of the data points. This hypersphere is defined by its radius r and center c, while the hyperparameter ν in the optimization function (3.1) controls the trade-off between the hypersphere's size and the allowance for outliers. The problem is formulated as the estimation of r, c, and a set of non-negative slack variables $\xi_1, \xi_2, \ldots, \xi_n$, with the data being mapped to a higher-dimensional space via a feature transformation $\phi(\cdot)$ to better capture complex patterns.

$$\min_{\mathbf{r},\mathbf{c},\xi} r^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \text{ subject to: } \|\phi(x_i) - c\|^2 + \xi_i \leq r^2 \textbf{ for all } i = 1, 2, ..., n. \tag{3.1}$$

Any data point that falls outside this region is considered an anomaly or outlier. The flexibility of OC-SVM comes from its ability to operate in both linear and non-linear spaces, allowing it to capture complex relationships within the data [32, 36].

OC-SVM has been widely applied in various fields, such as fraud detection, intrusion detection, and anomaly detection in sensor data. Its ability to detect deviations from normal patterns makes it particularly useful when there is a lack of labeled data for anomalous classes. For instance, in navigation anomaly detection, OC-SVM can be trained on normal user navigation behavior and subsequently identify outliers, such as bot activity or system errors. The kernel trick in OC-SVM enables it to map the data to higher-dimensional spaces, enhancing its capability to detect more subtle and complex anomalies that may not be linearly separable in the original feature space. Despite its strengths, OC-SVM is sensitive to the choice of parameters, such as the regularization parameter and the kernel function, which can significantly affect its performance [32, 31].

3.2 Markov Chain

A Markov Chain is a stochastic process $\{X_t\}_{t\geq 0}$ with a state space $S = \{s_1, s_2, \ldots, s_n\}$ and a transition matrix Q, where the Markov property holds: $P(X_{t+1} = s_j \mid X_t = s_i) = P_{ij}, \quad \text{with} \quad \sum_{j=1}^{n} P_{ij} = 1 \quad \forall i$, where P_{ij} is the probability of transitioning from state s_i to state s_j.

The Markov property states that the probability of transitioning to each of the possible next states in a Discrete-Time Markov Chain (DTMC) depends solely on the current state and not on the sequence of preceding states. This property defines a first-order DTMC, where future states are conditionally independent of past states given the present state [24]. Discrete-time Markov chains are characterized by transitions occurring at specific time steps and are particularly effective for modeling systems with a memoryless nature. They have been widely applied in fields such as bioinformatics, natural language processing, recommender systems, and web navigation and search patterns [16]. Mathematically, a DTMC is represented by a finite set of states and a transition probability matrix, where each element specifies the likelihood of moving from one state to another at a given discrete time step.

This model's simplicity and mathematical tractability make it a fundamental tool for studying probabilistic systems and for solving real-world problems that exhibit stochastic behavior [12].

Markov Chains have proven to be effective in anomaly detection and bot identification, particularly in systems where sequential patterns and transitions are critical. By modeling user behaviors as state transitions, Markov Chains can identify deviations from typical navigation patterns in applications such as website monitoring or fraud detection [6]. For example, the transition probabilities between states – such as webpage visits

or API requests – can help distinguish between legitimate users and automated bots that exhibit irregular navigation paths [10]. Additionally, Markov-based anomaly detection techniques are computationally efficient, making them suitable for real-time detection in high-traffic environments (Liu et al. [19]).

3.2.1 Continuous-time Markov Chain

A Continuous-Time Markov Chain (CTMC) is an extension of the discrete-time Markov Chain that models systems evolving over continuous time. Unlike discrete-time Markov Chains, where transitions occur at fixed time intervals, CTMCs allow transitions to occur at any point in time, governed by exponential waiting times. They are particularly suited for applications where the timing of transitions is critical, such as queuing systems, reliability analysis, and anomaly detection [24].

In the context of bot detection, CTMCs can be used for capturing the dynamics of user behavior by modeling the time spent in each state (e.g., webpage or API request) and the transitions between states. This capability makes CTMCs ideal for identifying irregular timing patterns, which are often indicative of automated or anomalous activity as pointed in [5]. Additionally, CTMCs can be integrated with machine learning models to enhance real-time detection systems by leveraging both temporal and probabilistic features [19].

A **Continuous-Time Markov Chain (CTMC)** is a stochastic process $\{X(t)\}_{t\geq 0}$ with state space $S = \{s_1, s_2, \ldots, s_n\}$ and a transition rate matrix $Q = (q_{ij})$, where q_{ij} is the rate of transition from state s_i to s_j, for $s_i \neq s_j$. The holding time in state s_i follows an exponential distribution with rate $-q_{ii}$, and the transition probability between states s_i and s_j is expressed as:

$$P_{ij}(t) = P(X(t) = s_j \mid X(0) = s_i).$$

The probability of a sequence of states $s_0, s_1, \ldots, s_k$ occurring at times $t_0 = 0, t_1, \ldots, t_k$ is given by:

$$P(X(t_1) = s_1, \ldots, X(t_k) = s_k \mid X(0) = s_0) = \prod_{i=0}^{k-1} P_{s_i,s_{i+1}}(t_{i+1} - t_i), \tag{3.2}$$

where $P_{s_i,s_{i+1}}(t)$ is the probability of transitioning from state s_i to state s_{i+1} in time t_{i+1}, given by:

$$P_{s_i,s_{i+1}}(t) = \left[e^{Qt}\right]_{i,i+1}.$$

Thus, the probability of observing a sequence of states over time is the product of the individual transition probabilities for each consecutive state pair, where each transition depends on the rate matrix Q and the time between transitions.

For example, for a sequence of transitions from state s_0 to s_1 at time t_1 and from s_1 to s_2 at time t_2, the probability is:

$$P(X(t_1) = s_1, X(t_2) = s_2 \mid X(0) = s_0) = \left[e^{Qt_1}\right]_{s_0,s_1} \left[e^{Q(t_2-t_1)}\right]_{s_1,s_2}.$$

Chapter 4

Data and Experimental Setup

This chapter outlines the decisions and methods employed in this study to analyze the differences between legitimate users and malicious agents in the e-commerce context. The methodology encompasses data collection, pre-processing, feature extraction, and analytical techniques to highlight relevant aspects of the bots behavior.

4.1 Data collection

The data for this study was collected directly from shoppers' browsers through a JavaScript plugin that captures all user interactions and sends them to the platform's data lake. This plugin passively records events such as page views, cart add and checkout events, providing a detailed log of user behavior. However, the API that receives these events lacks both client-side authentication and field validation before data is stored.

We retrieved the data directly from the data lake by executing SQL queries and subsequently exporting the query results into *CSV* files. Given the nature of the dataset, which comprises navigation events, the volume of data is substantial, estimated at approximately 7.30 Terabytes for the selected accounts.

Handling such a massive volume of information required substantial computational power, which exceeded the available resources at the time of research. Consequently, processing the entire dataset in its raw form was not feasible. To address these limitations, we employed two separate sampling strategies to ensure the data provided a representative sample of the population while allowing for efficient analysis: the **Store-Level Distribution Sampling**, to capture class distributions across stores, and **Actor Behavior Sampling** to provide a diverse set of interactions for in-depth behavioral analysis.

By employing these two sampling approaches, we aimed to balance our analysis' breadth and depth. The first sampling strategy, a stratified method, was designed to capture the distribution of bots and legitimate users across various stores, allowing us to accurately measure and characterize the presence of each class within individual stores and

across different segments. The second sampling strategy ensured that we captured enough behavioral data from each type of actor, providing a robust foundation for subsequent analysis and for training machine learning models to distinguish between legitimate users and bots.

This way, both samples were used on the Exploratory Data Analysis step, allowing us to understand the difference between bots and legitimate users from multiple perspectives. Additionally, this approach ensured that we had sufficient data to train the anomaly detection models effectively.

Both sampling strategies use the `request_type` and `labels` attributes available in the collected events. Details of these sampling processes are presented in the following sections and outlined in Algorithm 1 of the sampling processes.

The `request_type` feature refers to the type of page being accessed during a session. When a purchase is made, this feature assumes the value *"orderPlaced"*, which indicates a confirmed transaction. In this study, we operate under the assumption that actors who complete any purchase are legitimate users. This assumption is based on the fact that completing a purchase requires going through multiple steps, such as providing shipping addresses and payment information, which significantly reduces the likelihood of the action being automated.

The second feature, `labels`, is used to identify bots. These labels are assigned by a Web Application Firewall (WAF) service, which is capable of detecting and flagging basic bot behaviors. The combination of these two features — purchase activity and bot identification — enables us to initially classify users and bots, providing a foundation for identifying patterns in the data.

Algorithm 1 Data Extraction and Sampling Process

```
Initialize Variables
stores_list ← ["AA1", "AA2", ..., "SF3"]
users ← set()
bots ← set()
sampled_events ← []
Step 1: Identify Users with Orders and Labels
for each e in navigation_events_table do
    if store is in stores_list and event_date is within specified range then
        e.has_orders = False
        e.has_labels = False
        if request_type is "orderPlaced" then
            u ← {"store": e.store, "mac_id": e.mac_id, "label": "user"}
            bots.put(u)
        end if
        if event.labels is not empty then
            b ← {"store": e.store "mac_id": e.mac_id, "label": "bot"}
            bots.put(b)
        end if
    end if
end for
users_bots ← users.union(bots)
Step 2: Sampling Based on Labels
if stratified == True then
    sampled_users ← users_bots.sample(frac=0.0025, partitionBy=[store, label])
else
    sampled_users ← users_bots.sample(n=2500, partitionBy=[store, label])
end if
Step 3: Retrieve Full Session Data for Sampled Sessions
for each u in sampled_users do
    for each e navigation_events_table do
        if e.mac_id == u.mac_id then
            sampled_events.append(e)
        end if
    end for
end for
Output: An events sample according the specified stratification method
```

Store-Level Distribution Sampling. To ensure a representative and manageable dataset for analysis, we employed a stratified sampling method targeting 0.025% of agents from each analyzed store and pre-defined class – bots and users. This approach involved segmenting the population into strata based on shared attributes, specifically `account_name` and `label`. Random samples were then drawn from each stratum, maintaining proportional representation of the overall population. This methodology ensures a balanced perspective of user interactions across different classes while reducing data

volume for more detailed exploration. Additionally, all events associated with the selected agents during the observation period were extracted to facilitate a comprehensive behavioral analysis over time.

Actor Behavior Sampling. Since we are analyzing multiple stores, selecting 0.025% of shoppers may not provide a sufficient number of events for stores with lower engagement. This limitation could negatively impact the analysis, as the reduced volume of data may not be enough to capture significant patterns or trends in user and bot behavior. To address this, the second approach selects a fixed number of 2,500 actors per store and class, ensuring an adequate volume of events for the subsequent analysis. This number was determined by analyzing the total number of events, ensuring that each store contributes a sufficiently representative volume of interactions for the subsequent analysis.

4.2 Dataset

The dataset encompasses navigation events recorded from August 2023 to April 2024 via a JavaScript application that tracks shopper interactions on the platform stores. The 0.025% sampling rate mentioned earlier corresponds to a total of 89,408 distinct agents, 159,844 sessions and, 587,641 events, which are considered sufficient for our analyses.

The analyzed stores are categorized into five distinct segments: pharmacies, fashion, beauty, groceries, and collectible items. This segmentation is crucial for identifying potential differences in user behavior between legitimate users and bots based on the type of product they are interested in. Table 4.1 describes the samples size of each store.

4.2.1 Attributes

Collected data consists of records that describes a shopper event, enabling a comprehensive tracking of their interactions over specific sessions or even larger periods. Table 4.2 outlines the key attributes utilized in this study.

To identify these users, we use the *Mac ID* attribute, which is a long-term cookie present in the navigation events and retained in the shopper browsers for up to one year or until be cleared. Since the majority of users do not log into the stores while navigating,

Table 4.1: Store samples size

Segment	Store	Total users	Total sessions	Total events
Sports & Fitness	SF1	18,382	1,795	3,210
	SF2	56,865	3,631	8,107
	SF3	56,855	4,099	9,347
Apparel & Accessories	AA1	1,006	414	460
	AA2	14,906	1,722	2,860
	AA3	97,487	13,665	26,279
Department Stores	DS1	137,277	26,204	44,303
	DS2	23,554	1,926	3,466
	DS3	48,171	9,054	17,141
Beauty & Health	BH1	80,725	15,651	27,045
	BH2	47,813	9,490	15,462
	BH3	4,600	1,757	2,164

the *Mac ID* serves as an efficient proxy for this purpose. Additionally, it's important to note that a single user may have multiple *Mac IDs*, as they may navigate using different browsers and devices.

Table 4.2: Attributes summary

Attribute	Description
account_name	Unique alias used to identify the platform different stores
session_id	UUID stored as cookie in the shopper browser that is valid for 30 minutes and can be renewed with new interactions
mac_id	Long-life identifier stored in the browser cookies of each shopper that is valid for one year and is renewed as new events are sent. However, it expires if the user clears their cookies
clientdate	Timestamp from the moment the event is stored.
request_type	Page type that is being accessed by the shopper.
user_agent	User-Agent string extracted in the server-side from the request.
url	URL of the accessed page.
ref	URL of the page that referred the shopper to this page.
ip	Shopper IP address
labels	Web Application Firewall (WAF) generated label

The *Session ID* attribute serves as a critical identifier for distinguishing distinct user activities within a reduced time window. By associating events with their respective session identifiers, it becomes possible to delineate user interactions occurring within a specific timeframe. The initial step of this study involves analyzing the behaviors for a

variety of agents and sessions, identified by `mac_id` and `session_id` attribute, aiming to establish the optimal granularity for subsequent analyses.

The *requestType* attribute, found in the events data, indicates the type of page accessed by the user and is categorized into *productView*, *homeView*, *categoryView*, *cart*, *payment*, *login*, *profile*, *orderPlaced*, and *others*. Further details about each of these page types are described on Table 4.3.

Table 4.3: Request type details

Request type	**Description**
productView	Event sent when the user load or refr'esh a product details page
homeView	Event sent when the user load or refresh the store home page
categoryView	Event sent when the user load or refresh a category page
cart	Event sent when the user load or refresh the cart page
payment	Event sent when the user load or refresh the payment page
login	Event sent when the user load or refresh the login page
profile	Event sent when the user load or refresh its profile page
orderPlaced	Event sent when the user finishes an order and then is redirected or reload the order placed page
others	Refers to all the events that could not be mapped to the previous values. This category may include events that are similar to others, but due to specific store configurations, the script was unable to identify the corresponding page.

Other fields such as the *url*, *user_agent*, and the *clientdate* are present in the data, providing valuable insights for distinguishing between legitimate and abnormal behaviors. Additionally, the *ref* attribute indicates the preceding *url* for the current page, enabling tracking of the shopper's path while navigating on the platform stores.

4.2.2 Labels

One issue commonly faced in anomaly detection applications is the lack or limited availability of labeled data. These anomalies, by nature, occur as rare cases in the dataset, making them difficult to anticipate or capture in large numbers. In addition, manual labeling is often impractical, as it requires expert judgment and can be resource-intensive, especially when the anomalous patterns are complex or domain-specific.

In this study, we address this challenge by assuming that purchase events reliably indicate legitimate user behavior. Using a basic Web Application Firewall (WAF) mechanism, we labeled bot activity, while `orderPlaced` events (see Table 4.3) served as proxies for legitimate user actions. Although straightforward, this approach provides a practical and reliable foundation for the exploratory data analysis phase.

At the time of conducting this study, the WAF served as the sole resource capable of detecting highly suspicious activities. It analyzes the volume of requests alongside the associated user agent, using these parameters to evaluate the behavior of incoming traffic. Subsequently, the firewall system assigns a specific label to each activity based on the applied rule, effectively categorizing it within the system.

Although the implemented firewall relies on basic rules to identify suspicious activities, it plays a crucial role in guiding the initial analysis and providing valuable insights, despite the fact that only the most obvious bot activities are detected. Furthermore, establishing a baseline for the expected performance of the proposed solution is essential. The labeling process is discussed in more detail in Section 4.3.2.

4.3 Data Pre-processing

The attributes described previously in Section 4.2.1 went through a pre-processing phase to refine the raw data. This process involved generating additional features, eliminating noise, and removing irrelevant records. As a result, the data became more structured and consistent, allowing for more accurate and comprehensive analysis and enabling us to extract meaningful insights.

4.3.1 Validating Mac and Session IDs

The attributes `mac_id` and `session_id` play a crucial role in distinguishing shopper interactions within the e-commerce environment. Both attributes are Universally Unique Identifiers (UUIDs), generated by a JavaScript plugin when an actor accesses an e-commerce page.

The `session_id` is used to track user activity within a specific session. It remains valid for 30 minutes of inactivity, after which it expires. This short lifespan helps us track a user's session while they are actively engaging with the site, enabling real-time tracking

of interactions such as product view, add to cart, and checkout events. In contrast, the `mac_id` serves as a long-term identifier, stored as a cookie in the user's browser. It lasts for one year unless the user decides to clear their browser cookies before that. This extended duration allows for continuity in tracking user behavior across multiple sessions, facilitating a deeper understanding of long-term engagement patterns and customer preferences.

Data consistency and reliability are crucial for any data-related task. To ensure this, we implemented a validation method using the UUID Python Library [1]. Universally Unique Identifiers (UUIDs) are standardized 128-bit identifiers designed to ensure uniqueness across systems and contexts. In our approach, the validation method attempts to create a UUID object from the string value of the attribute – `mac_id and session_id, specifically`. If the string can be successfully converted into a valid UUID, it is considered correct; otherwise, an exception is raised, indicating that the string does not conform to the UUID format, thereby marking it as invalid. This ensures the integrity of attributes that are expected to follow the UUID standard.

This method classifies both `mac_id` and `session_id` fields as either valid or invalid, resulting in the creation of two new attributes: `is_valid_mac_id` and `is_valid_session_id`. These attributes enabled us to introduce an additional criterion for identifying malicious agents as attackers often exploit payload fields to inject malicious code, such as in SQL injection attacks. As result, we identified only 12 new suspicious agents, which suggests that such code injection attacks may either be rare or too sophisticated to be detected with basic validation.

4.3.2 Labeling

To label legitimate user activities, we identify them based on the occurrence of past purchase activity. Specifically, individuals who complete a purchase transaction are labeled as legitimate users. This approach assumes that purchasing activity serves as a reliable indicator of genuine user engagement, enabling us to differentiate between legitimate users and other types of interactions, such as bot activity or website indexing.

On the other hand, we assume that events flagged by the Web Application Firewall (WAF) mechanisms are likely generated by bots (such sessions receive a "bot" label). Details of the implementation are provided in Algorithm 2.

[1] UUID Python Library available at https://docs.python.org/3/library/uuid.html

Algorithm 2 Simple Labeler

```
1: user_mac_ids ← set()
2: potential_bot_mac_ids ← set()
3: Step 1: Identify Users and Bots
4: for each e in navigation_events_table do
5:     if request_type == "orderPlaced" then
6:         user_mac_ids.put(e.mac_id)
7:     else if "bot" in e.labels then
8:         potential_bot_mac_ids.put(e.mac_id)
9:     end if
10: end for
11: Step 2: Classify sessions according the mac_id label
12: for each e in navigation_events_table do
13:     if e.mac_id is in user_mac_ids then
14:         e.label ← "user"
15:     else if e.mac_id is in potential_bot_mac_ids then
16:         e.label ← "bot"
17:     else
18:         e.label ← "unknown"
19:     end if
20: end for
```

While these rules may effectively identify some types of user and bot behaviors, they are insufficient to cover the full spectrum of possible malicious activities. As a result, there may be instances where certain behaviors are misclassified or overlooked. Despite its limitations, this rudimentary labeling system serves as an initial step towards distinguishing between genuine user interactions and bot activity within the system.

4.3.3 Feature augmentation

Although the existing attributes provide a solid foundation for analysis, additional detail is required to identify the complex behaviors central to this research. To address this, feature augmentation plays a crucial role in enhancing the representational power of our dataset. This process involves generating new features derived from the existing data, facilitating deeper insights into shopper behavioral patterns.

Datetime Features

Measuring and tracking shopper activities over time is essential for achieving our research goals. To facilitate this analysis, we utilized the `clientdate` attribute to extract several new time-based features, including `month`, `week`, `day`, `week_day`, and `hour_of_day` (see Table 4.4). These additional attributes enhance our ability to analyze shopper interactions across different temporal dimensions. Temporal granularity enables us to explore whether any coordinated behaviors emerge among malicious agents, providing valuable insights into the dynamics of both legitimate and suspicious activities on the platform.

Table 4.4: Datetime features

Attribute	Format	Example
clientdate	`yyyy-MM-ddTHH:mm:ss.SSSZ`	`2024-10-30T13:57:21.836+0000`
month	`YYYY-MM`	`2024-10`
week	`YYYY-MM-DD`	`2024-10-27` (first day of week)
week_day	`EEEE`	`Wednesday`
day	`YYYY-MM-DD`	`2024-10-30`
hour_of_day	`HH`	`13`

In addition to the attributes described in Table 4.4, we extracted the `state_spent_time` feature to determine the amount of time an actor spent on a specific page. To capture this, we ordered all events generated within a session and calculated the time difference in seconds between consecutive events. Using this ordered sequence, we created the `session_start_date` and `session_end_date` attributes by identifying the timestamps of the first and last events in each session. This also enabled us to calculate `session_duration` as the time difference between the `session_start_date` and `session_end_date`, capturing the total duration of each session and providing a measure of the total time spent per session.

However, it's important to point that it isn't possible to precisely determine the time spent in the last state (and, consequently, the session duration). Since there is no subsequent event to signal the end of the last state, we lack a precise timestamp to measure the time spent on that page. To address this limitation, we assume that the session concludes when the shopper accesses the final page, which is excluded from the duration calculation. This approach provides consistency in session duration calculations while acknowledging that final state time may not be fully captured.

Query parameters features

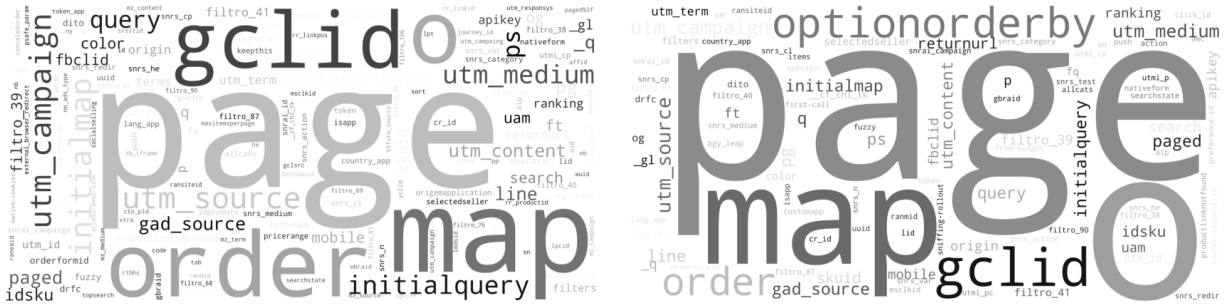
Query parameters are key-value pairs added to the end of the URL in web requests, usually after a question mark (?). They enable the client to send specific data to the server, helping to define or modify the request. Each query parameter consists of a key and a value, separated by an equal sign (=). This structure allows clients to include several key-value pairs in a single request, separating them using an ampersand (&). For example, in the URL `https://fakestore.com/search?query=books&category=fantasy`, the query parameters are `query=books` and `category=fantasy` with `query` and `category` as the keys and `books` and `fantasy` as their respective values. Using this extraction approach, we identified **411** distinct query parameters across various stores. Additional details on the parameter extraction process are provided in Algorithm 3.

Algorithm 3 Extracting query parameters

```
keys ← []
values ← []
for u in urls do
    url_path ← p.split('?')[-1]
    url_params ← url_path.split('&')
    for p in url_params do
        if '=' not in p then
            result.append('')
        else
            k ← p.split('=')[0]
            v ← p.split('=')[1]
            keys.append(k)
            values.append(v)
        end if
    end for
end forreturn keys, values
```

Although we have an established pattern for extracting query parameters from URLs, merchants retain the flexibility to customize these parameters based on their specific needs, leading to inconsistencies that complicate accurate extraction across different stores. To address this challenge, we selected the top 100 most frequent query parameters from both `url` and `ref` attributes to create the Word Clouds (see Figure 4.1a) highlighting their most commonly used parameters.

Next, we conducted a semantic analysis of the identified query parameters, examining their respective values to determine the type of information they referred to. For non-obvious parameters with unclear meanings, we conducted a manual analysis by navigating the stores' websites to determine the specific page types they referred to. This



(a) (b)

Figure 4.1: Query Parameters Word Cloud of `url` (a) and `ref` (b) attributes

process allowed us to identify parameters related to search operations—such as filtering, ordering, and search terms—as well as those associated with ad campaign pages and pagination details.

Request Type. Due the substantial volume of events labeled as "otherView" in the raw data, we decided to use the identified query parameters to further categorize label the `request_type` associated to these events. Some query parameters, such as `map`, may have multiple meanings depending on the context, making it necessary to examine the associated values to accurately determine the `request_type`. By adopting this strategy we successfully categorized 52% of the "otherView" events.

Table 4.5: Criteria for Determining Request Type

Request Type	Identified Query Parameters	Associated Values
searchView	`q`, `_q`, `query`, `_query`, `termo`, `initialquery`, `initialmap`, `fuzzy`, `filter`, `filters`, `filtro`, `busca`, `terms`	-
	`map`	`filter`, `filtro`, `terms`
	`o`	`order`, `pricerange`
categoryView	`map`	`c`, `category`, `categoria`
storePage	`returnurl`, `productlinknotfound`	-
productView	`idsku`	-

Table 4.5 summarizes the query parameters used to identify the type of page being accessed. Each request type is associated with a distinct set of query parameters and, in certain cases, it is necessary to analyze the associated parameter values to accurately assign the appropriate label, as some parameters may serve multiple purposes depending on their context.

Figure 4.2 illustrates the predominance of request types as percentages, before and after the labeling refinement process (see Table 4.6 for detailed numbers). The labeling

introduced two new request types, `categoryView` and `searchView`, accounting for 6.36% and 18.32% of the total events, respectively. This process significantly reduced the proportion of `otherView` events by 24.69%, enhancing our ability to distinguish between different types of events during the actors' navigation.

Figure 4.2: Request Type Percentage before and after labeling

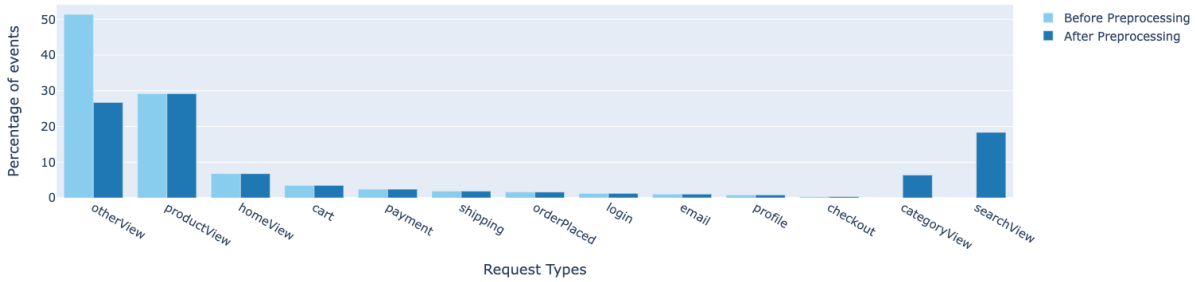


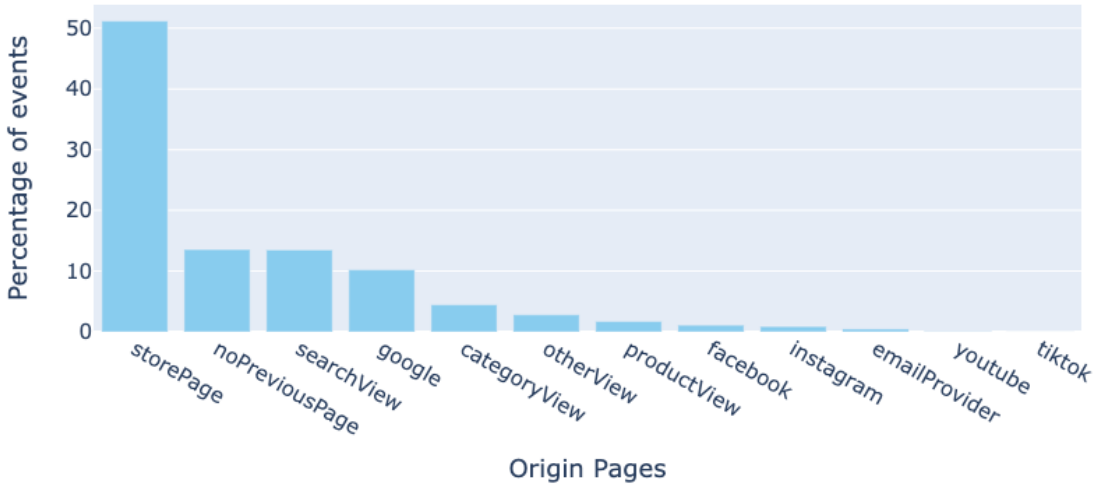
Table 4.6: Request Type Metrics before and after labeling

Request Type	Metric	Before	After
otherView	number of events	1,620,396	841,788
	percentage of events	51.37%	26.68%
productView	number of events	919,463	919,465
	percentage of events	29.15%	29.15%
categoryView	number of events	0	200,667
	percentage of events	0.00%	6.36%
searchView	number of events	0	577,939
	percentage of events	0.00%	18.32%

Origin Page. The `ref` attribute (see Table 4.2) contains the URL of the page accessed immediately before the current one, allowing us to trace session origins. By analyzing the `ref` attribute from the first event in each session, we can identify the source of user traffic. Since origin pages may refer to external websites, our methodology is not limited to query parameters alone. Initially, we analyze the domain of the `ref` URL to determine whether it belongs to a known source, such as Google, Facebook, TikTok, or the store itself. If the domain matches the one in the `url` attribute, we treat it as an internal page and apply the previously described methodology for inferring values to label the origin page. Unlike the `request_type`, where the `otherView` label is used when query parameters do not match any predefined rule, the `origin_type` assigns the label `storePage` in such cases.

The majority of the identified origin pages correspond to internal pages (see Figure 4.3), which explains the similarity between the query parameters extracted from both the `ref` and `url` attributes, as illustrated in Figures 4.1a and 4.1b.

Figure 4.3: Origin Page Percentage



Page Index. Similar to the `origin_type` and `request_type` attributes, the `page_index` is extracted from the query parameters present in the `url`. To achieve this, we first manually identified a set of query parameters commonly associated with page indices during shopper navigation. Algorithm 4 was then implemented to iterate through this set, checking for matches in the `param_keys` attribute and identifying the corresponding index. Once a match is found, the `page_value` is extracted from the `param_values`.

Figure 4.4: Number of pagination events for each request type

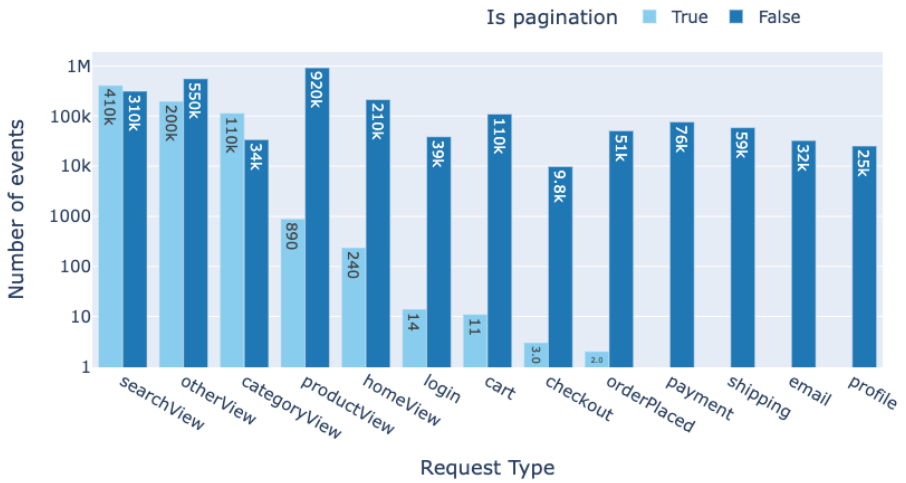


Figure 4.4 displays the distribution of identified pagination events across all request types, contrasted with events without pagination. It's interesting to observe that both `searchView` and `categoryView` events exhibit a higher number of pagination events compared to non-pagination events, highlighting the significant use of paginated navigation in these contexts.

Algorithm 4 Retrieve Page Number from Query Parameters

```
Input:
param_keys (array): parameter keys associated to the url/ref
param_values (array): parameter values associated to the url/ref
Initialize Variables
page_key_candidates ← ['page', 'pagenumber', 'paged', 'pg', 'p']
page_value ← NaN
Step 1: Check if any candidate key exists in the URL parameters.
for key in page_key_candidates do
    if key in param_keys then
        page_index ← IndexOf(param_keys, key)
        page_value ← param_values[page_index]
        break
    end if
end for
if page_value is digit then return page_value
end if
return NaN
```

Chapter 5

Exploratory Data Analysis

Despite the challenge posed by limited labeled samples, it is essential to build an initial foundation for guiding subsequent analysis and the development of bot detection models. In this section, we outline the general characteristics of the observed data, aiming to uncover distinctive behavioral patterns that differentiate genuine users from automated entities, as well as identify the features that most effectively distinguish the two groups.

Figure 5.1: Number of events

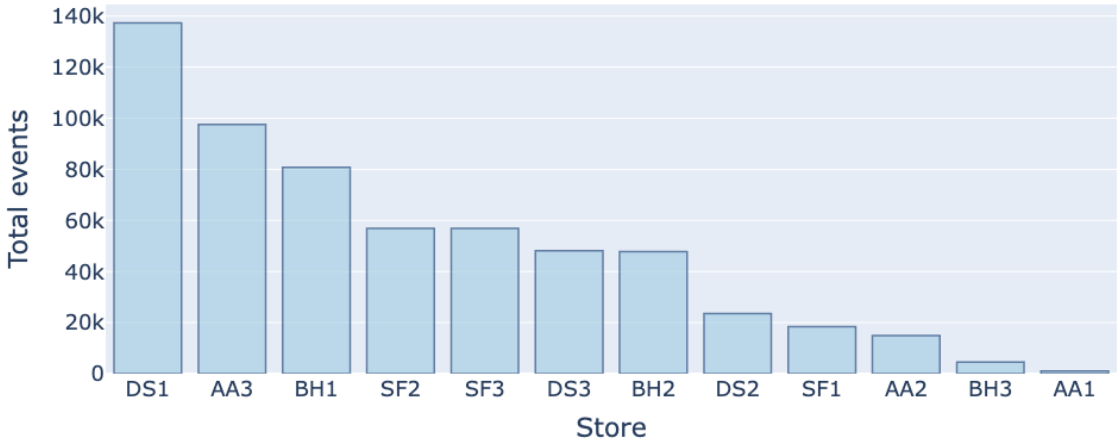


Figure 5.1 illustrates the total number of events collected for each store, highlighting their diverse customer bases. The dataset used to generate this figure was collected through the stratified sampling approach, ensuring that the distribution of events reflects the original proportions observed in each store. This method preserves the representativeness of overall customer interactions, enabling more accurate and meaningful insights into each store's performance and user engagement.

The *DS1* stands out as the store with the highest number of navigation events, totaling 132,277, while *AA1* has the fewest, with only 1,006. The selection of these stores was intentional, aiming to include a range of stores with different levels of audience engagement. This variety is crucial for understanding if malicious agents tend to target larger, more popular stores. By examining both high-traffic stores, like *DS1*, and smaller ones, like *AA1*, we aim to validate whether bots are more likely to concentrate their

activities on stores with a broader customer base, potentially exploiting higher visibility and more opportunities for unauthorized actions.

Figure 5.2: Predominance of session labels

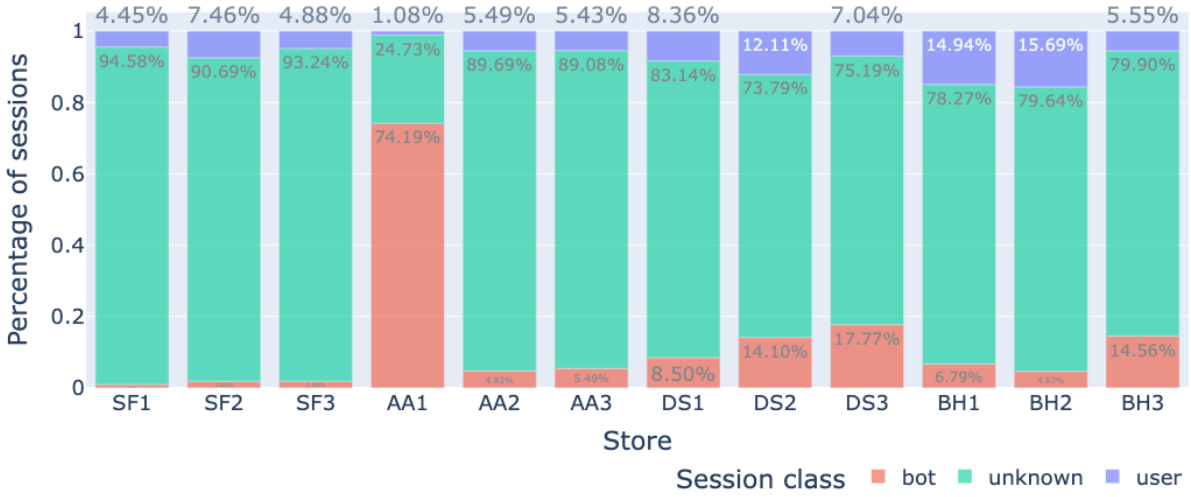


Figure 5.2 illustrates the distribution of user and bot sessions across the analyzed stores. Department Stores (*DS1*, *DS2*, *DS3*) exhibit a higher proportion of bot sessions compared to user sessions, suggesting that the diversity of their product catalogs may attract a broader range of automated activities, potentially aiming to exploit high-demand or popular items. This trend is also observed in the *AA3* and *BH3* stores, indicating that the phenomenon is not confined to a specific segment, although certain stores appear to be more heavily targeted.

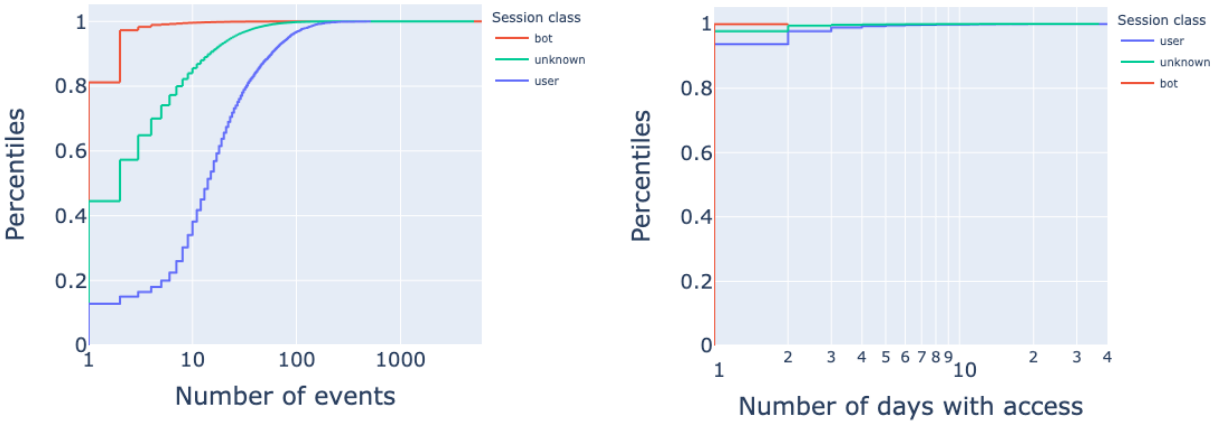
Notably, *AA1*, despite having the smallest audience, records the highest proportion of bot sessions at 74.19%, followed by *BH3* with 14.56% (Figure 5.2). The significant 60% gap between these two stores suggests that smaller stores, potentially perceived as less secure or more vulnerable, may attract a disproportionately large number of malicious bots compared to larger, more prominent stores. This disparity underscores the importance of considering store-specific characteristics when analyzing bot activity in e-commerce environments.

5.1 Mac IDs lifespan

Given that MAC IDs are stored in an actor's browser for up to one year or until be manually cleared, analyzing their lifespan offers valuable insights into shopper behavior, particularly in distinguishing between legitimate users and bots. Bots tend to clear their

browser caches more frequently to avoid detection, leading to shorter MAC ID lifespans. However, because the events collected for this study cover only a specific time window, it's impossible to precisely determine when a MAC ID expires or how long it was actively retained in the browser.

This limitation emphasizes the need to measure the number of accesses for each actor. A significant number of actors may only have a single session, resulting in a brief and potentially misleading MAC ID lifespan. Such short durations may simply represent a one-time shopper, rather than indicating bot behavior.



(a) (b)

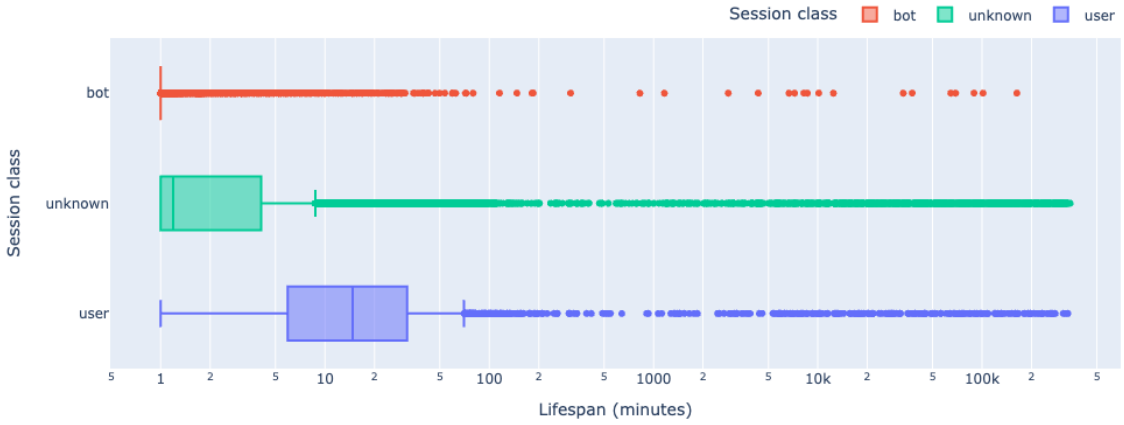
Figure 5.3: Frequency of (a) events and (b) access.

Figures 5.3(a-b) above illustrate a significant difference in the number of events generated by bots and users, with users showing a third quartile (Q3) of 23 events, compared to just 1 event for bots (Figure 5.3a). However, the majority of actors, both users and bots, only access the stores once (Figure 5.3b). This indicates that the "Lifespan" calculated later provides only a partial view of the overall behavior, as it fails to capture the behavior of recurrent shoppers, offering a limited representation of their long-term engagement.

Despite the presented limitations, the initial hypothesis relies on the assumption that bots' MAC IDs will exhibit a shorter lifespan, even within an established time window, a behavior that reflects a common practice among malicious agents of frequently clearing browser caches to avoid detection. This hypothesis is supported by the data shown in Figure 5.4, which illustrates the MAC ID lifespan across bots, users, and unknown actors. Figure 5.4 highlights that bot sessions tend to have significantly shorter lifespans compared to regular users, further indicating the deliberate attempts by bots to obscure their tracks by clearing cache and cookies more often.

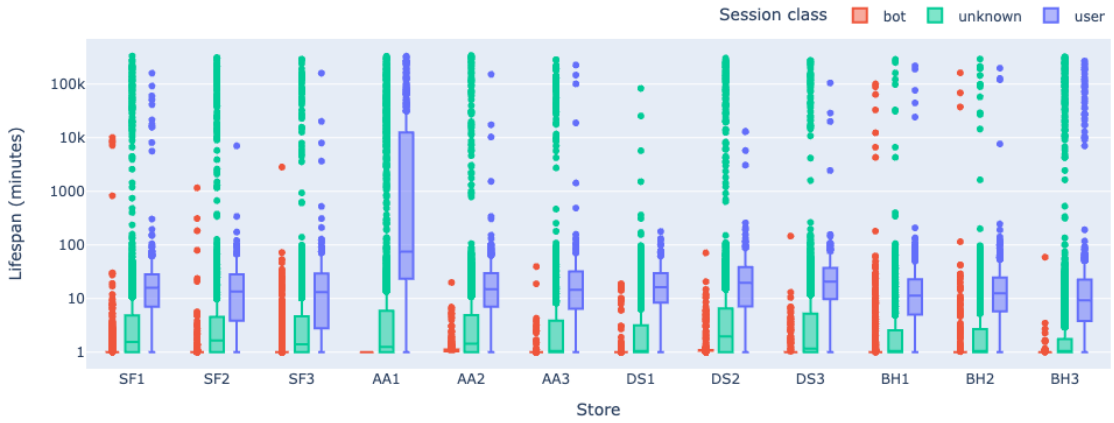
As illustrated in the Figure 5.4, all the actor classes presents a heavy-tailed distribution, with the majority of them having Mac IDs with relatively short lifespans. However,

Figure 5.4: Mac IDs lifespan - general perspective



there is a significant difference between the user and bot distributions, where the bots presents a 95th percentile (P95) of 1 minute, whereas users have a P95 of 346,138 minutes (around 240 days) for the users. The unknown actors distribution seems to be a mix of the other two and has a P95 of 56,091 minutes (around 39 days).

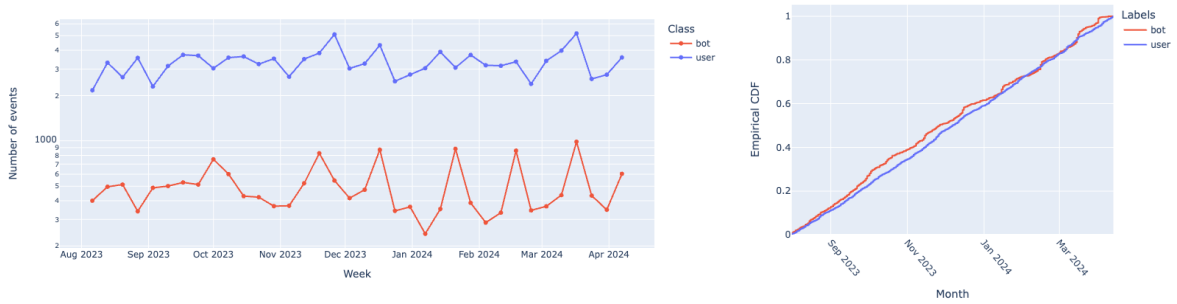
Figure 5.5: Mac IDs lifespan - individual perspective



The lifespan of Mac IDs was analyzed both in a generalized manner and individually for each store (Figure 5.5) to assess potential differences between stores or segments and evaluate the feasibility of a generalized perspective, which can help determine whether it is necessary to employ models tailored to specific segments or even individual stores based on variations in lifespan and other relevant attributes. Analyzing the stores distribution, it's reasonable to say that all of them present the same distribution pattern for all the actor classes.

5.2 Timeline behavior

An interesting behavior is illustrated in the Figure 5.6a, where seasonality in access patterns can be observed. For users, this seasonality is more consistent, while for bots, there are some more noticeable peaks of activity. This suggests that while users follow a relatively steady pattern of engagement, bots may be programmed to exploit certain time periods or events.



(a) Event Volume Trends Over Time (b) Event Volume Growth Over Time

Figure 5.6: Events over time

Figure 5.6b presents the Empirical Cumulative Distribution Function (ECDF) for the number of events generated by both users and bots during the analyzed period. Although both classes share some overlapping spike periods, resulting in the curves being relatively close to each other, the bot activity curve remains more unstable, with greater fluctuations in comparison to the user curve.

To determine whether the apparent coordinated behavior of bots is a general trend or more typical in certain stores, we illustrate the individual behavior for each analyzed store in Figures 5.7 and 5.8. These visualizations allow for a comparison of the bot activity patterns across stores, highlighting any consistent trends or store-specific variations.

Figure 5.7 illustrates the events timeline for each store separately, revealing two distinct behaviors. Although the previously identified pattern has been observed in the majority of the accounts, it is not the most prevalent behavior across all stores. For the stores *BH1*, *BH2*, *BH3*, *DS2*, and *DS3*, bot activity appears to be more stable and consistent over time. In contrast, the remaining stores exhibit behavior more aligned to the general trend, with noticeable fluctuations and spikes in bot activity at specific intervals.

The Empirical Cumulative Distribution Functions for each store are illustrated in Figure 5.8 offer an alternative perspective on the actors' behavior over time. It is evident that for some stores, the bots' ECDFs display noticeable "steps" that correspond to the

Figure 5.7: Number of events generated by each store over the time

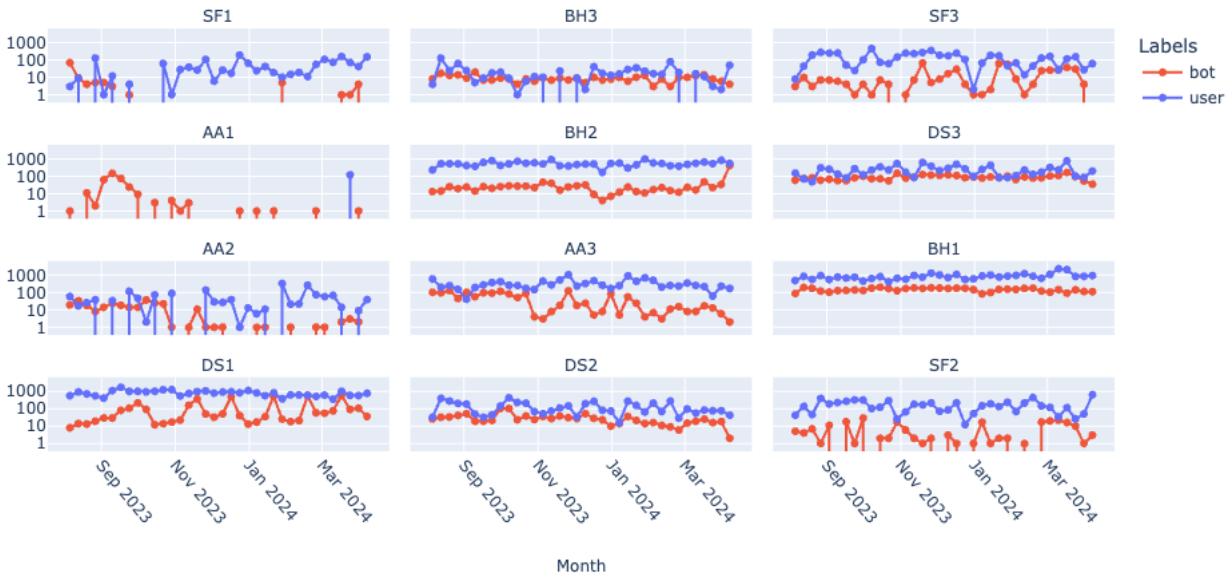
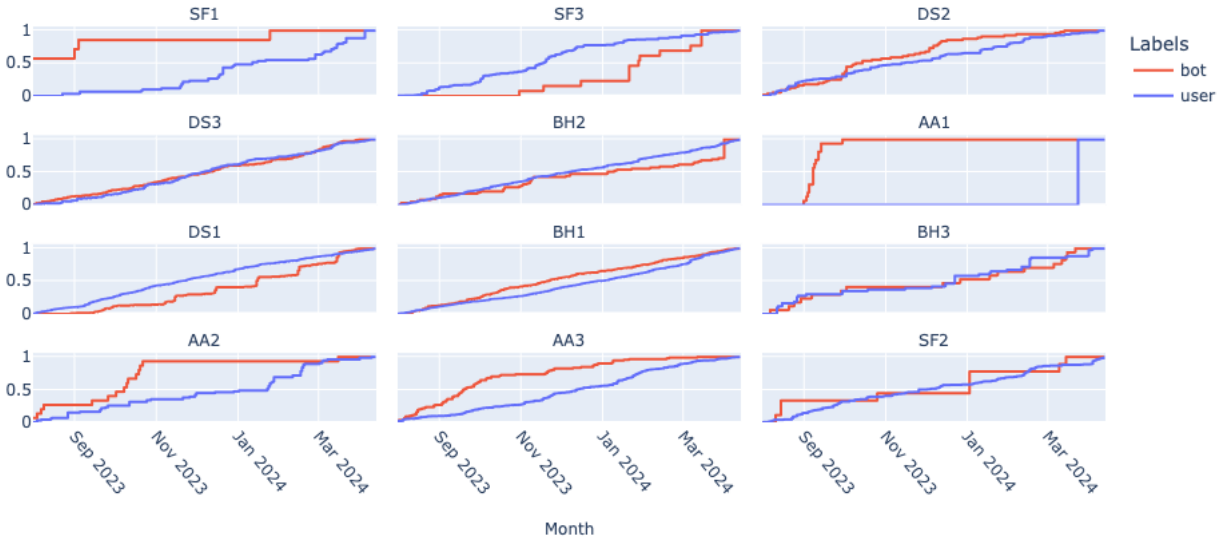


Figure 5.8: Cumulative number of events generated by each store over the time



peaks previously observed in Figure 5.7. This suggests a pattern of coordinated or periodic bot activity aligned with the spikes in event timelines.

5.3 Sequences Analysis

The sequences refer to the chronologically ordered pages accessed within a session, where each page corresponds to a specific `request_type` (see Table 4.2). Subsequent

analysis aims to determine whether there are noticeable differences in the navigation patterns of bots and users, potentially revealing distinct behavioral trends between the two groups.

5.3.1 Pagination

During navigation, an user may need to paginate through a Product Listing Page (PLP) to locate the desired product(s). These PLPs typically consist of search results or category pages, allowing shoppers to refine their search by applying filters and selecting an ordering criterion. The raw data doesn't include the page index, making it necessary to extract it during the pre-processing phase described in Section 4.3.

Page index extraction primarily involves manually identifying the query parameters that indicate the pagination index for each store. The identified keys were *"page"*, *"pagenumber"*, *"paged"*, *"pg"*, *"p"*. However, for stores *AA1*, *BH1* and *BH3*, we were unable to identify a specific query parameter associated with the page index.

We incorporated pagination events into the `request_type` as "nextPageView" events and introduced a new attribute, `page_index`, into the dataset to indicate which page is being accessed. This addition enables us to track page navigation behavior more precisely, capturing instances where users or bots progress across multiple pages within a session.

Figure 5.9: Number of pages accessed in each session

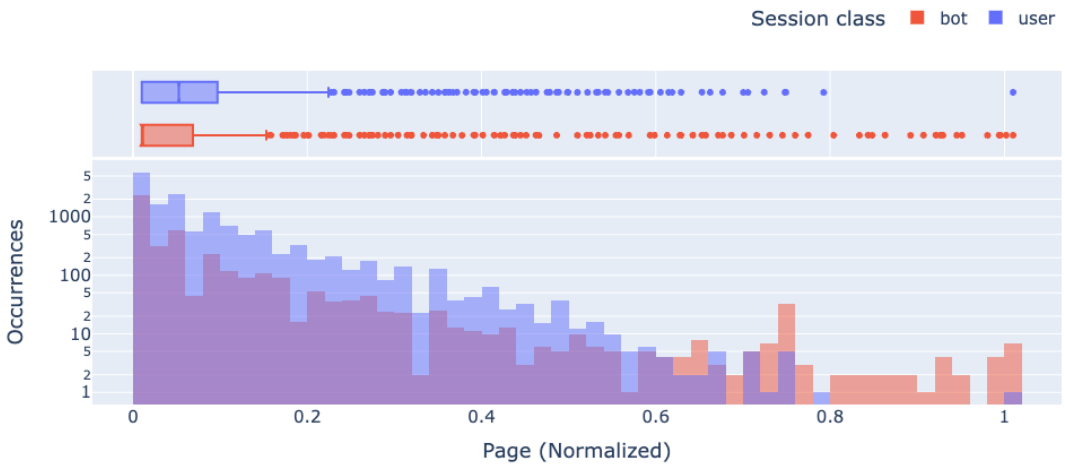
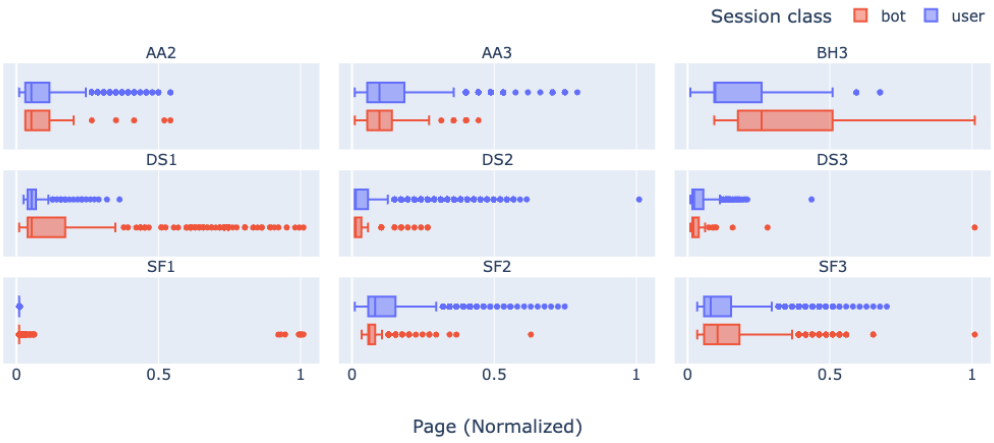


Figure 5.9 illustrates the distribution of pagination activity, where the `page_index` is normalized by the 90th percentile for each store. This normalization adjusts for vari-

ations in catalog sizes across stores, allowing for a consistent comparison of pagination behavior between users and bots, regardless of differences in product catalog lengths.

Two interesting behaviors can be observed in Figure 5.9. First, the second quartile (Q2, or median) normalized value for legitimate users is 0.0525, while for bots, it is 0.0103. Additionally, actors who navigate deeply are predominantly identified as bots. This suggests that both superficial and deeper navigation is more characteristic of bot behavior compared to legitimate users.

Figure 5.10: Number of pages accessed in each session by each account



We also analyzed the pagination distribution separately for each store, revealing both high and low levels of bot activity across pages. In stores such as *SF2*, *AA3*, *DS2*, and *DS3*, bots display a tendency to navigate deeply. In contrast, in the remaining stores, we observe the opposite trend, where bots paginate less frequently than legitimate users. This variation in pagination behavior suggests different browsing patterns across stores and underscores the importance of tailoring detection strategies to account for these differences.

5.3.2 Page types

Sequence states are composed of `request_type` events. However, the raw events, as outlined in Table 4.3, lack sufficient granularity to capture the nuances of agent behavior, with many events labeled simply as "otherView". Applying the method outlined in the Section 4.3.3, we successfully reclassified 23% of the "otherView" events, allowing for a clearer distinction of behaviors within the dataset.

Among the available attributes is the `ref` attribute, which specifies the page that a user accessed immediately before the current page. This attribute is available for the first

session event, enabling us to trace the origin of each access. However, the origin pages themselves are not labeled as the events pages. To overcome this limitation, we applied the method described in Section 4.3 to create the `origin_page` attribute.

Figure 5.11: Sessions Origin Page Breakdown

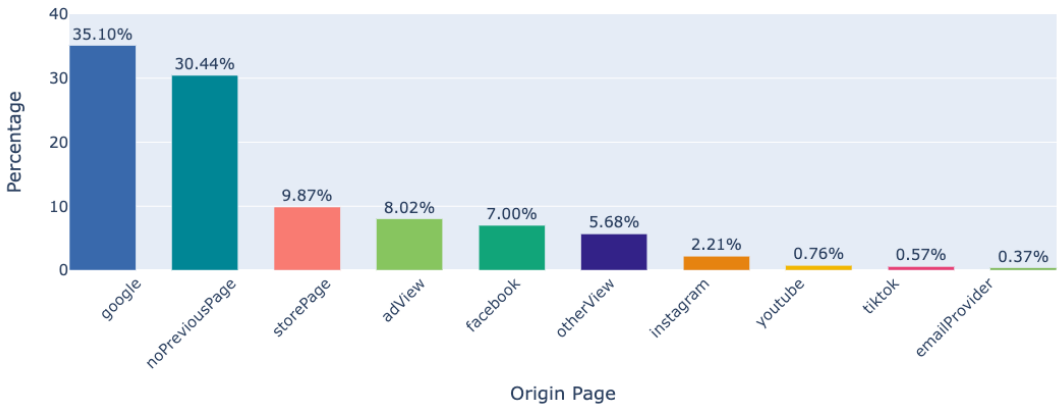


Figure 5.11 displays the percentage of sessions initiated from each of the origin pages and provide an overview of how different pages serve as the starting point for the actor sessions. The majority of sessions (35.10%) originate from Google, while a significant portion (30.44%) has no identified origin page. Sessions without an identified origin may occur due to failures in the data collection process, or they could refer to situations where the actor directly enters the store URL into the browser, bypassing any referral source.

The origin "storePage" ranks third, with 9.87% of sessions beginning directly within the store. This may occur if the actor remains inactive for more than 30 minutes, prompting the plugin to generate a new `session_id` for the session.

Figure 5.12: Sessions Origin Page Breakdown by Class

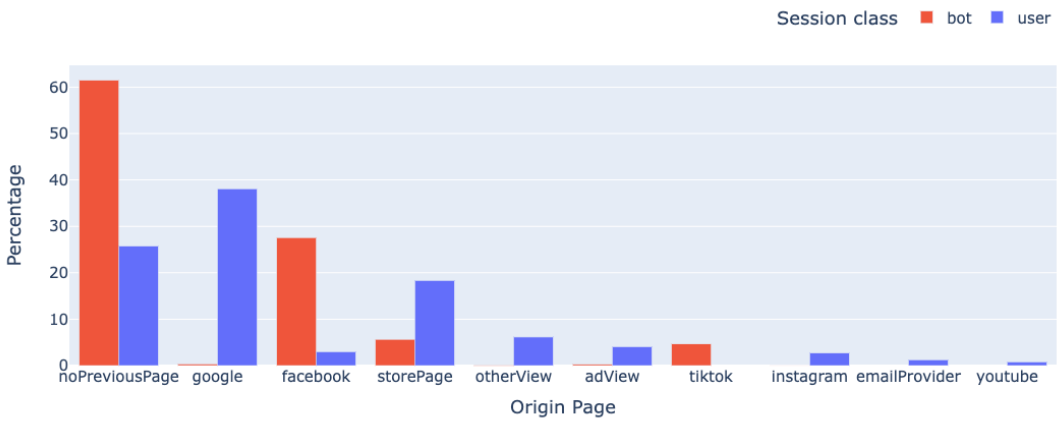


Figure 5.12 illustrates the percentage of bots and users who initiated their sessions across the different origin pages. The results reveal that the most frequently used origins exhibit a predominance of one class over the other. Google, for instance, is a primary

entry point for sessions initiated by legitimate users, indicating a strong association with authentic traffic. In contrast, sessions without an origin page show a predominance of bot activity, suggesting that this absence of an origin page may be an indicator of bot-driven interactions. More details are presented in Table 5.1.

Table 5.1: Sessions Origin Page Breakdown by Class

Previous Page	**Bot (%)**	**User (%)**
noPreviousPage	61.46%	25.73%
google	0.32%	38.08%
facebook	27.56%	2.97%
storePage	0.83%	14.18%
searchView	4.81%	3.12%
otherView	0.10%	6.16%
tiktok	4.64%	0.00%
adView	0.26%	4.05%
instagram	0.01%	2.69%
emailProvider	0.00%	1.23%
productView	0.01%	0.81%
youtube	0.77%	0.77%
categoryView	0.20%	0.20%

The origins “storePage”, “adView”, “instagram”, “emailProvider” and “youtube” also show a higher prevalence of legitimate user sessions, while “facebook” and “tiktok” appear to be common origins for bot traffic, suggesting these sources are frequently exploited by automated agents. It’s interesting to observe that the predominant class varies among social networks, with certain platforms showing a higher prevalence of legitimate users while others are more commonly associated with bot traffic.

The predominance of legitimate users across different origins offers valuable insights that can serve as an important feature in training anomaly detection models. By analyzing the origin of a session, models can leverage the correlation between entry points and legitimate engagement, helping to accurately identify potential bot traffic.

This chapter demonstrates that bot and user behaviors follow distinct patterns, allowing for effective differentiation. Through extensive analysis, we identified key navigation characteristics that distinguish legitimate users from automated entities, enabling us to determine the most relevant features for classification. These findings establish a strong foundation for the next stages of our study, where we leverage these insights to design and evaluate machine learning approaches for bot detection.

Chapter 6

Model Design, Implementation and Evaluation

In this chapter, we explore the design and implementation of the models developed for bot detection within the e-commerce context. At the time of this study, the Web Application Firewall (WAF) served as the only detection mechanism in place. Consequently, we adopted it as a baseline for our experiments, focusing on developing a model capable of replicating the WAF's bot detection labels. This strategy validates the reliability of the proposed approach in identifying less sophisticated malicious agents while establishing a solid foundation for extending its capabilities to detect more advanced and evasive threats.

Our hypothesis is based on the assumption that bots and legitimate users exhibit distinct navigation patterns, particularly in terms of the types of pages accessed and the time spent on each. To capture this behavior accurately, we modeled agent actions using the Continuous Time Markov Chain (CTMC) framework. This approach maintains a transition rate matrix for every possible state transition, which serves as a reference to compute the transition probabilities based on the pair of pages and the time spent on the source page. This chapter aims to assess the CTMC's capability in characterizing agent behavior, particularly its effectiveness in distinguishing between legitimate users and bots by analyzing their navigation sequences.

6.1 Features modeling

6.1.1 Continuous-Time Markov Chain Modeling

The attributes presented in previous approaches focused primarily on session-level metrics and isolated session events, demonstrating their utility in establishing a baseline

model for our study. However, our hypothesis posits that bots and legitimate users exhibit fundamentally distinct navigation patterns. By effectively capturing these patterns, we aim to enhance the model's ability to differentiate between these two groups more accurately.

Transition rate matrix

By modeling the sequence of session events as a Markov Chain (Section 3.2), we analyzed the navigation patterns of users and bots by calculating the transition rates between all possible states. To better understand the user and bot behavior, we created heatmaps containing these rates for all possible transitions for three distinct groups: users, bots, and unlabeled sessions. The heatmaps are structured such that each row represents a source state, and each column corresponds to a target state, with the values indicating the transition rates.

Figure 6.1: Transition Rates for legitimate sessions (in log scale)

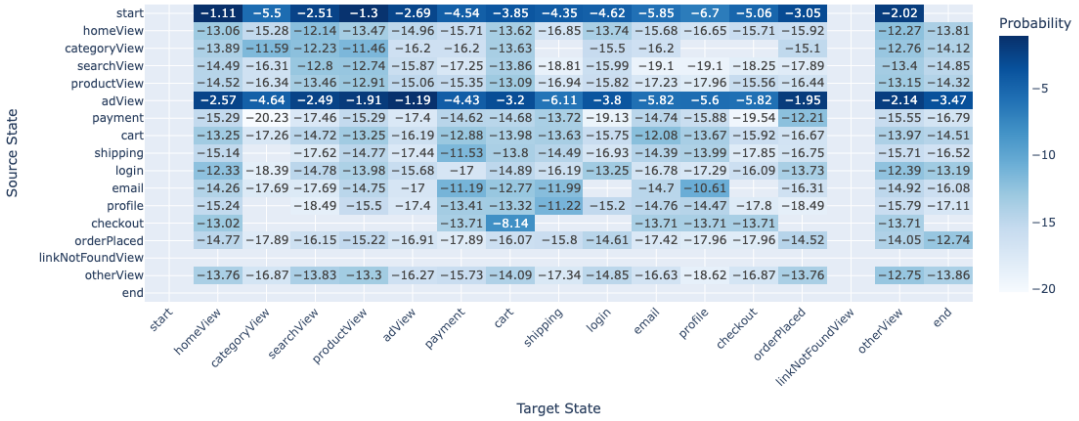
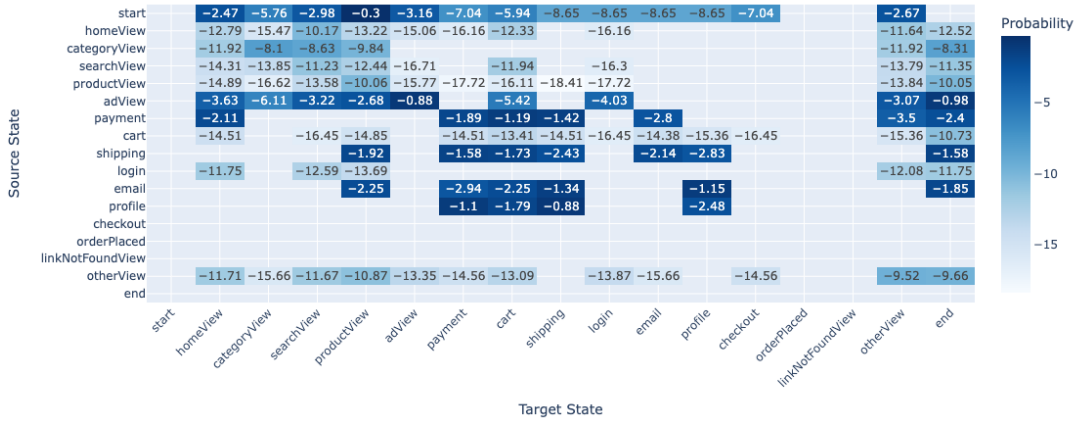


Figure 6.1 visualizes the transition rates between web pages visited by legitimate users. Each cell in the heatmap represents the rate of transition from one page (row) to another page (column). The transition rate is calculated as the inverse of the average time (in milliseconds) users spend on each page before navigating to the next one – regardless the destination. These rates are displayed on a logarithmic scale to better visualize the wide range of values. Notably, the heatmap reveals that nearly all state-to-state transitions are possible within this group, reflecting the diversity of navigation behaviors exhibited by legitimate users. A particularly notable pattern emerges from transitions originating from the `adView` event: it serves as one of the most frequent sources for transitions to other pages. Additionally, it is important to emphasize that, since purchase

occurrences are the sole criterion for labeling sessions as legitimate, transitions involving end-funnel steps (e.g., `checkout` and `payment`) tend to exhibit elevated rates.

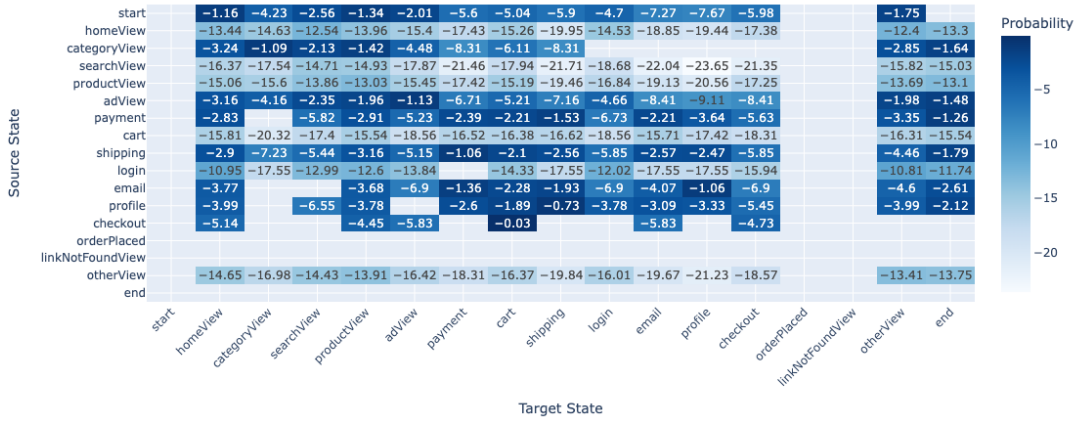
Similarly, Figure 6.2 illustrates the transition rates for bot sessions, revealing a stark contrast compared to the previous heatmap. This difference lies in the significantly reduced number of possible transitions, as evidenced by numerous gaps in the transition matrix. This sparsity highlights the more constrained and repetitive navigation patterns characteristic of bot behavior.

Figure 6.2: Transition Rates for suspicious sessions (in log scale)



It's interesting to observe that suspicious activities are concentrated on `productView` events, followed by checkout-related events such as `shipping` and `payment`, highlighting objectives such as scraping product catalogs, reserving items by adding them to the cart, and potentially attempting profile invasions during the checkout process. However, despite the elevated rate, it is essential to note that this does not imply a higher likelihood of occurrence for the transition itself. Instead, it means that, given the source state, the most frequent choice is the observed destination state.

Figure 6.3: Transition Rates for unlabeled sessions (in log scale)



Finally, Figure 6.3 displays the transition rates for unlabeled sessions—i.e., those generated by agents without an assigned label. Since these sessions are a mix of both

legitimate and malicious agents, the heatmap is expected to exhibit a combination of patterns observed in the user and bot transition matrices.

Sequence Probabilities

Sequence probabilities are calculated by integrating both the structural and temporal dynamics of agent behavior. The structural aspect is captured through a transition rate matrix, which encodes the rates at which agents move between different states. These rates, stored in logarithmic form, represent the intensity or speed of transitions, rather than direct probabilities. The logarithmic scale is employed because, due to the large number of possible transitions, the resulting likelihood values can be extremely small, which can hinder the analysis and interpretation of the data.

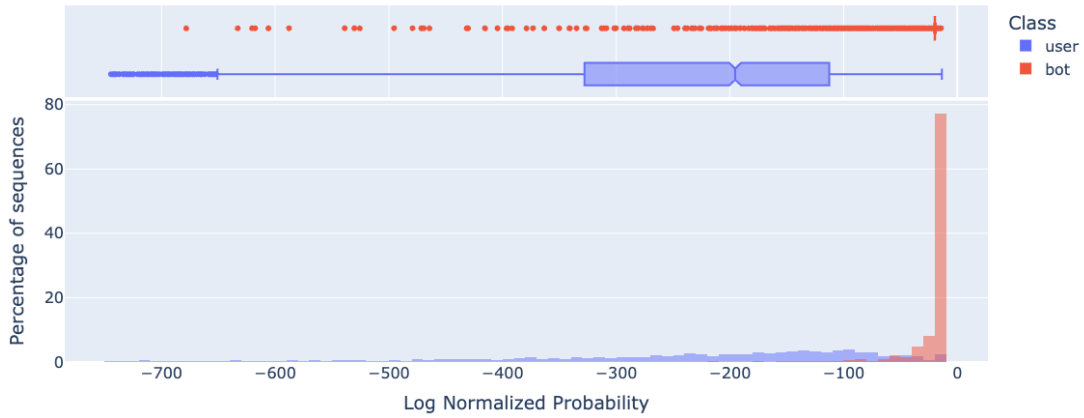
To compute the actual transition probabilities, the log-transformed rates are adjusted using a Continuous Time Markov Chain (CTMC) model. This model encodes the time spent in the current state and transition rates to other states, assuming a "memoryless" property. Specifically, the likelihood of leaving a state does not change over time, with longer durations in a state corresponding to a lower likelihood of transitioning. By multiplying the transition probabilities for each step across the entire sequence, the overall sequence probability is computed, which captures both the structural transition dynamics and the temporal patterns of agent behavior.

In our study, this framework was employed to compute sequence probabilities for both user- and bot-exclusive sequences. As outlined in Equation 3.2, the likelihood of a given sequence is determined as the product of the transition rates between consecutive states and the probabilities of remaining in a state for a specific duration. Given the substantial number of distinct sequences, the probabilities were log-transformed to ensure numerical stability and streamline the analysis. This transformation also accounted for the influence of sequence length on probability, mitigating the bias that could otherwise favor longer sequences.

With the achieved results, we could investigate the probability distributions for both groups in a scenario where the transition matrix is modeled based on user behavior, allowing us to examine how each group aligns with or diverges from typical user behavioral patterns.

Figure 6.4 displays the log-normalized probability distribution for both bot and user sequences, revealing an intriguing pattern. The likelihood of observing a bot sequence, given the user transition matrix, is concentrated on the right side of the chart, whereas the user probabilities are more evenly distributed across the range. This be-

Figure 6.4: Distribution for Sequences Normalized Probability



havior can be attributed to overlapping transitions between bot and user sequences. It means bots may exploit common transitions that are frequently observed in user behavior, resulting in a higher concentration of probability in specific regions of the distribution.

This overlap indicates that, while the sequences themselves are expected to differ—due to the unique goals or strategies of bots—the transitions within the sequences are not entirely different from those of legitimate users. Bots may follow the same common transition paths as users, but they probably tend to do so in a more deterministic way, resulting in fewer variations across their sequences compared to the broader range of user behaviors.

Sequence probabilities were then added to our dataset as a new feature, representing the likelihood of the observed sequence of events occurring within a session. In Chapter 7, we discuss the impact of this new feature on model's performance.

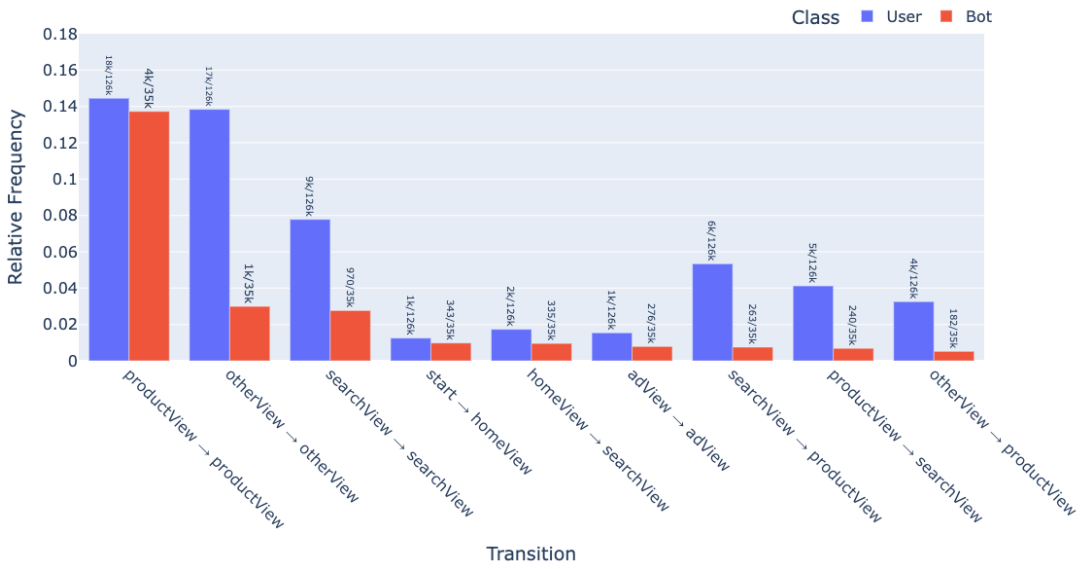
State transitions

For a better understanding of the behavior presented previously, we computed the top 20 (most frequent) transitions present in bot and user-exclusive sequences and analyzed the intersection between these transitions (see Figure 6.5).

The top 20 transitions in both bot and user-exclusive sequences include 9 shared transitions, accounting for 45% of the total, highlighting that nearly half of the most frequent transition patterns are common across both groups. This overlap suggests that bots are not entirely different in terms of the structural dynamics of their behavior. Rather, they tend to mimic certain user transition patterns, which contributes to the concentration of bot sequence probabilities in specific regions of the distribution.

Despite the observed similarity, an deeper analysis reveals that `productView →`

Figure 6.5: Shared Transitions Ranking Between Bot and Users



`productView` and `otherView → otherView` are the most frequent in both user and bot groups. The predominance of the `productView` transition is expected, as viewing products is a natural behavior in both legitimate shopping activities and bot-driven navigation. This suggests that relying solely on transition states may not be sufficient to distinguish between bots and users, as both groups share a significant overlap in their structural navigation patterns.

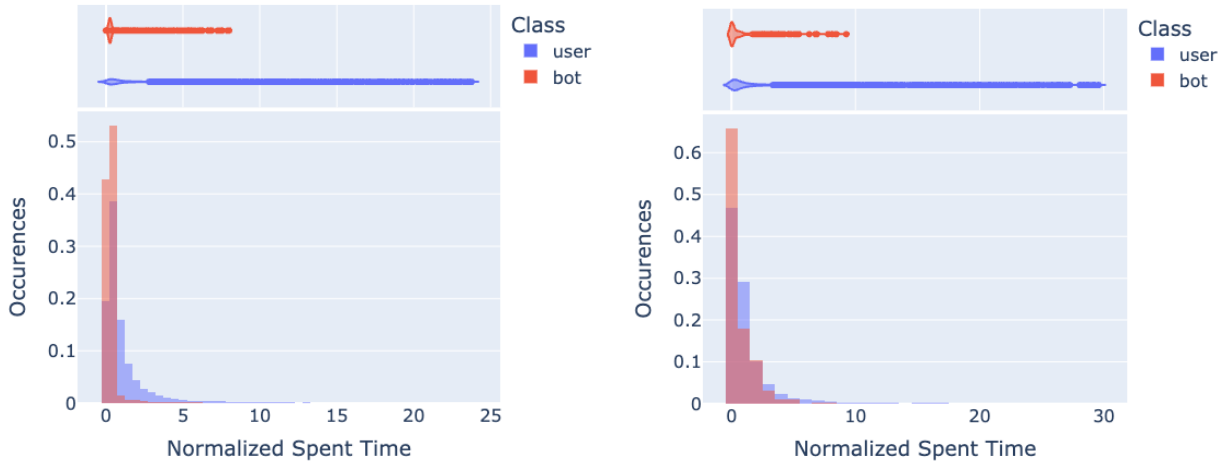
On the other hand, the `otherView → otherView` transition also appears prominently – especially for legitimate users – but offers limited distinguishing power. The vague nature of the "other" state likely contributes to its inability to differentiate between user and bot behaviors, underscoring the need to refine and improve the current page labeling process. Ambiguous or poorly defined states may dilute the effectiveness of behavioral analysis, making it harder to extract actionable insights from the navigation sequences.

To further investigate these transitions, we analyzed the distribution of time until transition for both `productView → productView` and `otherView → otherView` transitions. Figures 6.6a and 6.6b illustrate the probability distributions of time spent on state until these transitions occur for both bots and users.

The results reveal a significant overlap in short time-spent durations, which is consistent across both groups. This overlap highlights a challenge in detecting bots solely based on these transitions, as the structural and temporal patterns for shorter durations are nearly indistinguishable between bots and legitimate users. Such shared behavior could reflect common, fast browsing actions that occur naturally in both automated and human navigation.

However, the distributions also reveal distinctive patterns for longer transition

times. A notable number of transitions with extended time durations are more heavily associated with user activities, suggesting that legitimate actors occasionally exhibit irregular or prolonged pauses in navigation. These longer durations may result from nature of shopping online, where the shopper analyses the product details.



(a) (b)

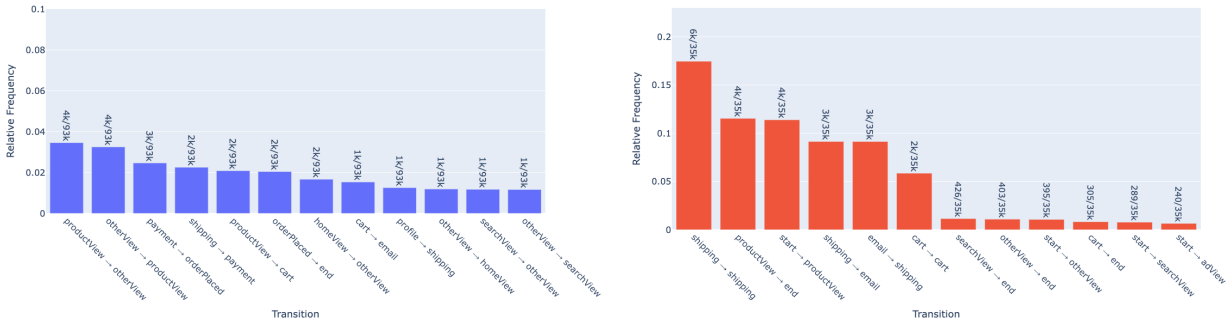
Figure 6.6: Transition Time Distributions for `productView → productView` (a) and `otherView → otherView` (b)

These findings help explain the high probability observed for bot sequences when user behavior is used as a reference. The overlap in shorter transition times suggests that bots mimic legitimate user behavior to some extent, particularly in common and quick transitions such as `productView → productView`.

At the same time, the distinctive presence of longer-duration transitions among bots introduces variability that contributes to the elevated probabilities. The user-based transition matrix, designed to model legitimate behavior, may not penalize these irregular patterns strongly enough, further inflating the likelihood of bot sequences. Figures 6.7a and 6.7b illustrate the frequency of exclusive transitions within user and bot sessions. Checkout events were excluded since purchase occurrences serve to label legitimate agents. The results reveal that bots tend to focus their activities on the first six transitions, whereas regular user behavior is more evenly distributed.

To better understand the differences in transitions between bots and users, we analyzed the relative frequency distributions of the most common transitions for both groups. Figures 6.8 and 6.9 illustrate these distributions, providing valuable insights.

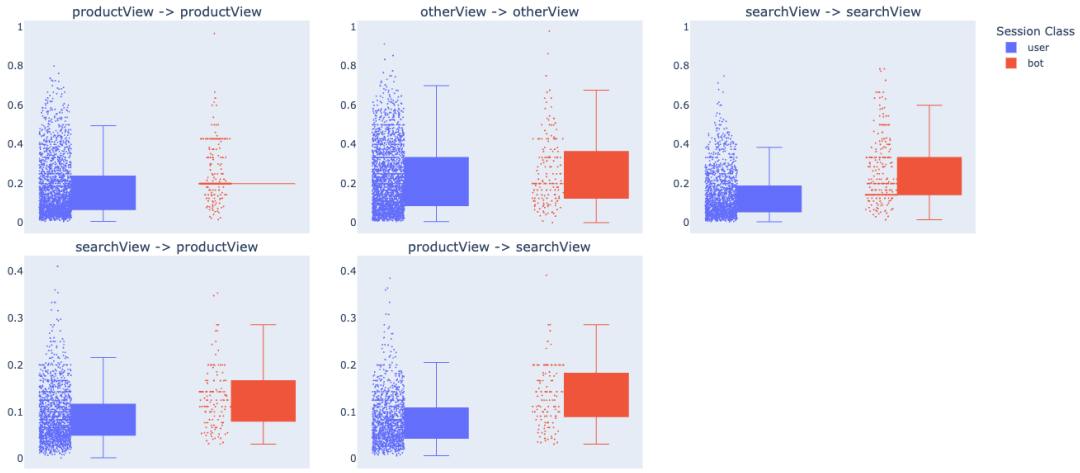
In Figure 6.8, the frequency distributions for the most common user transitions encompass or overlap with those of bots. Conversely, for bots, the frequency distributions often fall outside the user distribution range, as shown in Figure 6.9. The `productView → productView` transition is the only exception, as it occurs frequently in both groups, making it a shared behavior.



(a) (b)

Figure 6.7: Exclusive transitions for `users` (a) and `bots` (b)

Figure 6.8: Relative Frequency of Users' Most Common Transitions

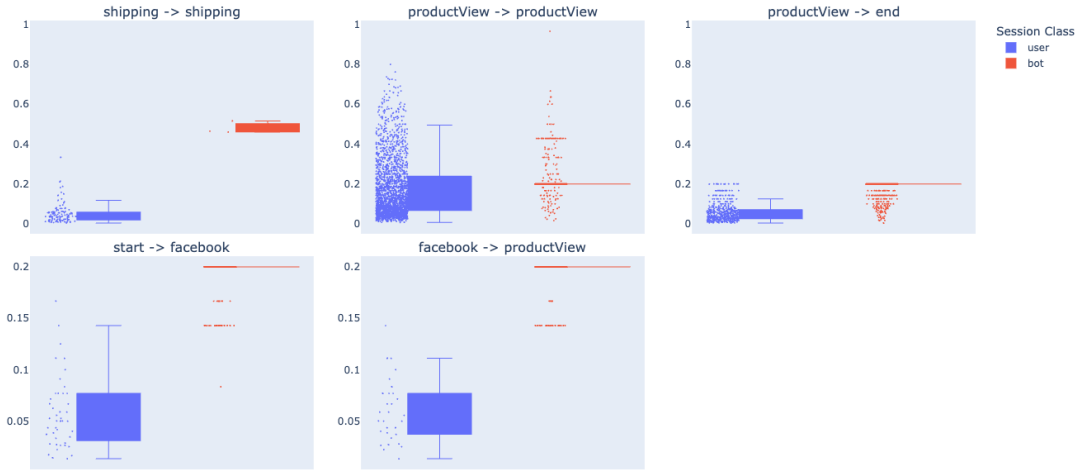


Another significant finding is the variance within the distributions. For certain transitions among users (`productView → searchView`, `searchView → searchView`, and `searchView → productView`), variances are comparable, suggesting a consistent pattern. In contrast, bot transitions exhibit highly concentrated distributions, with values tightly clustered depending on the transition type, whereas user distributions maintain reasonable variability.

Based on these analyses, we introduced the relative frequency attribute $R(T_{ij})$, a new feature that quantifies the relative frequency of the transition T_{ij} from state s_i to state s_j, normalized by the sequence size. This relative frequency is computed for the transitions presented in Figures 6.8 and 6.9. The purpose of this feature is to describe the sequence in terms of the most frequent behaviors observed in both groups.

In the end, these observations highlight the limitations of relying solely on transition states and their durations to distinguish between bots and legitimate users. While bots can mimic user behavior to some extent, particularly in common transitions, they also exhibit temporal anomalies that set them apart.

Figure 6.9: Relative Frequency of Bots' Most Common Transitions



6.2 One-Class Support Vector Machine

In the context of bot detection, the One-Class Support Vector Machine (OC-SVM) is a powerful tool for addressing the imbalance between legitimate user and bot activities. Unlike traditional classifiers, which distinguish between multiple classes, the OC-SVM focuses solely on modeling the characteristics of one class — legitimate user behavior — while treating deviations from this norm as potential anomalies.

Compared to other one-class classifiers, OC-SVM provides a well-defined decision boundary, making it more interpretable than deep learning-based approaches like Autoencoders, which require extensive hyperparameter tuning and large training datasets. Unlike methods such as Isolation Forest, which assumes anomalies are sparse and well-separated, OC-SVM effectively captures nuanced bot behaviors that closely resemble legitimate users. Additionally, it leverages kernel functions to handle high-dimensional data, enabling it to model subtle deviations in navigation patterns. This capability is particularly valuable in bot detection, where fraudulent behaviors may not always exhibit clear, easily separable characteristics. By focusing on deviations from learned user behavior rather than requiring explicit knowledge of bot activities, OC-SVM remains effective even in scenarios where little prior information about bot strategies is available.

For the context presented in this study, we used the OC-SVM to identify anomalous sequences in agents' navigation. A sequence is formally defined as an ordered list of state transitions within a single session, where each transition is represented by a pair (s_i, s_j), with s_i denoting the origin state and s_j the destination state in the sequence. For each pair, the transition time t_{ij} is calculated as the difference between the timestamps of the two states. The sequence is constructed chronologically based on t_i, encapsulating the temporal and navigational behavior of an agent during a session.

This section outlines the application of OC-SVM for bot detection, detailing its configuration, the rationale for choosing this method, and the evaluation of its performance. Although OC-SVM is an unsupervised approach, we validate its predictions by checking whether the identified anomalies correspond to known bot activities, as labeled by the Web Application Firewall (WAF), and whether the non-anomalous cases align with legitimate user behavior. This dual validation process ensures the model's reliability in distinguishing bots from genuine users.

To evaluate the model's effectiveness, we apply the metrics described in Section 6.3, enabling a comprehensive assessment of its performance. By leveraging the unsupervised nature of OC-SVM, our goal is to establish a baseline model capable of detecting anomalies in user navigation behavior, specifically differentiating between bot and legitimate user interactions. If the OC-SVM model proves successful in detecting these anomalies, it can be trusted to identify unknown malicious agents. This would enable the extension of its application to label unkown sessions or users, ensuring continuous and adaptive bot detection as new threats evolve over time.

6.2.1 Model Training

During the model training phase, we systematically experimented different input configurations, progressively enhancing them by adding new and more sophisticated features. This iterative approach allowed us to explore the impact of increasingly complex representations on model performance. Table 6.1 summarizes the features explored.

To enhance the understanding of the experiments conducted, the features have been grouped into distinct scopes, with each scope reflecting a specific dimension of the observed data. This categorization not only highlights the varied aspects of user behavior and session characteristics but also facilitates a systematic exploration of how different feature sets contribute to the model's performance. The experiments, along with the respective data configurations, are systematically presented in Table 7.1.

To prevent data leakage, we excluded end-funnel events from the totalizer features, specifically those ranging from `email` to `orderPlaced`, retaining only events shared between bots and users. The "Session Attributes" encapsulates general characteristics of each session, providing an overarching view of its structure and activity. "Sequence Probability", derived from transition matrix rates, represents the likelihood of an agent's sequence of events occurring based on legitimate user behavior. Finally, the "Transition Attributes" capture the relative frequency of the most common transitions for bots and users within each sequence, offering insights into distinct navigational patterns. Based on

Scope	Attributes	Description
Totalizers	`totalAdEvents`	Count of ad-related events in a session.
	`totalCartEvents`	Count of cart-related events in a session.
	`totalCategoryEvents`	Count of category events in a session.
	`totalHomeEvents`	Count of home page events in a session.
	`totalProductEvents`	Count of product-related events in a session.
	`totalSearchEvents`	Count of search-related events in a session.
	`totalShippingEvents`	Count of shipping events in a session.
	`totalOtherEvents`	Count of uncategorized events in a session.
Session Attrs	`uniquePages`	Count of distinct pages visited in a session.
	`originPage`	Source page of the session.
	`sessionDuration`	Total duration of the session.
	`totalEvents`	Total count of events in a session.
	`avgTransitionTime`	Average time between events.
Sequence Prob.	`sequenceProbability`	Probability of a sequence occurring given the modeled CTMC.
Transition Attrs	`startToProduct`	Fraction of `start → product` in a session.
	`facebookToProduct`	Fraction of `facebook → product` in a session
	`searchToSearch`	Fraction of `search → search` in a session
	`searchToProduct`	Fraction of `search → product` in a session
	`productToSearch`	Fraction of `product → search` in a session
	`productToProduct`	Fraction of `product → product` in a session
	`productToEnd`	Fraction of `product → end` in a session
	`shippingToShipping`	Fraction of `shipping → shipping` in a session

Table 6.1: Summary of training input features

experiments summarized in 7.1, we trained different instances of OC-SVM model varying the input and documenting the results, described in Section 7.

For model training, the dataset of user events was partitioned into training (60%), validation (20%), and testing (20%). Since the OC-SVM is trained exclusively on legitimate user data, bot events were excluded from the training phase and instead evenly distributed between the validation and testing sets, with each receiving 50% of the bot events. This approach ensures that the model is evaluated on unseen anomalies while preserving a balanced distribution for performance assessment.

In this study, we deliberately chose not to implement outlier removal techniques during data preprocessing. This decision was driven by the understanding that outliers may represent anomalous or unexpected behaviors that are essential to the identification of bots. Removing these data points could potentially obscure patterns indicative of malicious activity, thereby reducing the effectiveness of the model in detecting such behaviors. By retaining all data, including outliers, we ensured that the full spectrum of naviga-

tional behaviors, both typical and atypical, was represented in the analysis, enabling a more robust and comprehensive detection process.

Finally, during the training phase, we conducted hyperparameter tuning to optimize the model's performance by systematically varying the values of key parameters. Table 6.2 outlines the parameters explored, their descriptions, and the range of values tested.

Table 6.2: SVM-OC Hyperparameter tuning configurations.

Parameter	**Description**	**Values**
`kernel`	Function that transforms input data into a higher-dimensional space for better separability.	`rbf, poly, sigmoid`
`gamma`	Coefficient for `rbf`, `poly`, and `sigmoid` kernels, controlling the influence of individual data points.	`0.001, 0.01, 0.1, 0.5, 1`
`nu`	Upper bound on the fraction of training errors and lower bound on the fraction of support vectors.	`0.001, 0.01, 0.1`
`shrinking`	Boolean flag indicating whether to use the shrinking heuristic to improve training efficiency.	`True, False`

6.3 Evaluation Metrics

Evaluating unsupervised learning models like OC-SVM poses a unique challenge due to the absence of predefined labels for direct comparison. However, in scenarios where labeled data is partially available, metrics such as Precision, Recall, and F1-Score can be utilized to assess performance. In this study, these metrics are derived by applying the trained model to a labeled subset of the data, comparing the predicted anomalies with known bot instances, and verifying non-anomalous predictions against legitimate user data. This methodology provides a quantitative assessment of the model's capability to effectively distinguish anomalous behaviors, even within an unsupervised learning framework. Next, we summarize the metrics used to evaluate the proposed method.

Since the OC-SVM is designed to learn the normal behavior of legitimate users, it constructs a boundary around this class in the feature space. Any instance that significantly deviates from this learned representation is flagged as an anomaly. In this context, bots are assigned as the positive class, while legitimate users are considered the nega-

tive class. This choice aligns with the typical anomaly detection framework, where rare or unexpected behaviors (e.g., fraudulent transactions, network intrusions) are treated as positive instances requiring identification. By structuring the classification in this manner, the evaluation metrics effectively capture the model's ability to minimize false positives (misclassifying legitimate users as bots) and maximize true positives (correctly identifying bots), ensuring a robust assessment of detection performance.

True Positive (TP). Refer to the instances where the model correctly identifies positive cases, such as bots accurately labeled as bots. This metric is crucial for understanding the model's success in detecting intended targets and serves as a foundation for calculating other performance metrics like Recall and F1-score. High TP values indicate effective detection of the targeted class.

False Positive (FP). Occur when the model incorrectly classifies false cases as positives – in our context, it means label legitimate users as bots. This error type is particularly critical in e-commerce contexts, as it can severely impact the user experience, depending on the platform's mitigation strategy, and undermine trust in the platform. Additionally, is a key factor in the calculation of other metrics, such as Precision.

True Negative (TN). Represent the scenario where the model correctly identifies a negative occurrence precisely—in other words, when legitimate users are precisely classified as legitimate. This metric reflects the model's ability to avoid classifying benign traffic as malicious, contributing to overall model reliability and specificity. High TN rates indicate the model is successfully distinguishing non-bot activity.

False Negative (FN). Occur when the model fails to identify positive occurrences, labeling them as negative. In our context, high FN rates indicates the model is failing to identify bot activities, which can have significant consequences, such as allowing malicious actors to bypass detection. Minimizing FNs is critical for ensuring robust Recall, as failing to detect bots compromises the effectiveness of the detection system.

Precision. Measures the proportion of correctly identified positive instances out of all instances classified as positive—i.e., the fraction of correctly identified bots among all the bot instances. It indicates the model's ability to avoid False Positive (FP), a critical aspect in contexts such as e-commerce bot detection, where misclassifying legitimate users as bots can harm the user experience. The Precision is formally defined in Equation 6.1.

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

Recall. Also known as **sensitivity**, measures the proportion of actual positive instances correctly identified by the model. Unlike Precision, which focuses on minimizing false positives, Recall emphasizes minimizing false negatives, ensuring the model correctly identifies the maximum possible number of bots. This metric, defined formally in Equation 6.2, is particularly valuable in scenarios where identifying all positive cases is critical.

$$Recall = \frac{TP}{TP + FN} \tag{6.2}$$

F1-Score. Is a statistical measure used to evaluate the performance of a classification model, particularly in scenarios with imbalanced datasets as it ensures neither precision nor recall dominates the evaluation. It represents the harmonic mean of precision and recall, combining these two metrics into a single value to balance the trade-off between false positives and false negatives. Therefore, to better address the real-world implications of our model, we prioritize an evaluation metric that allows adjusting the balance between precision and recall according to the practical requirements of the application.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

Fβ-Score. In e-commerce applications, false positives (i.e., classifying a user as a bot) can be more harmful than false negatives if, for example, misclassified users are blocked in the platform. Therefore, to better address the real-world implications of our model, we also consider an evaluation metric that allows a custom balance between precision and recall, chosen according to the practical requirements of the application. The Fβ-Score (Equation 6.4) generalizes the F1-Score by introducing a positive real factor β, which determines the relative importance of recall compared to precision, allowing the metric to adapt to different contexts where either minimizing false positives or false negatives is more critical. To minimize false positives, we set β `= 0.5` in our experiments, prioritizing precision over recall.

$$F\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{6.4}$$

Specificity Also known as the true negative rate, measures a model's ability to correctly identify negative instances. In the context of e-commerce bot detection, this means accurately classifying legitimate users as non-bots. High specificity indicates that the model effectively avoids false positives, which is crucial for minimizing disruption to genuine users.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{6.5}$$

Chapter 7

Experimental Results and Analysis

In this chapter we present the experimental results and analysis conducted to evaluate the proposed method for detecting anomalous behaviors in e-commerce based on navigation data. By applying the OC-SVM model to labeled and unlabeled data subsets, we aim to assess its effectiveness in distinguishing legitimate user sessions from bot activity.

The experiments systematically explore different input configurations, combining distinct feature scopes with diverse hyperparameter settings to ensure a thorough evaluation of the model's performance. Furthermore, we examine how feature engineering and data representation influence detection capabilities, shedding light on behavioral patterns within both user and bot groups. The adopted criteria for determining the most relevant results is the Fβ-Score, chosen for its ability to balance precision and recall based on their relative importance in this context. Table 7.1 provides a summary of the data scope utilized across the conducted experiments.

Experiment	**Attributes Scope**
Experiment 1	Totalizers
Experiment 2	Totalizers and Session Attributes
Experiment 3	Transition Attributes
Experiment 4	Transition and Session Attributes and Sequence probability

Table 7.1: Experiments Summary

7.1 Experiment Results

Experiment 1. In the first experiment, we focused exclusively on the totalizer attributes, a straightforward set of features derived from the navigation data. These attributes encapsulate all navigation events, excluding those associated with end-funnel steps such as `login` and `payment`. This exclusion was necessary to prevent potential data leakage, given that our criteria for defining legitimate behavior relies on purchase-related

events. Table 7.2 summarizes the results and the hyperparameter configurations selected for the best-performing models in this experiment.

Parameters	**TN**	**TP**	**Precision**	**Accuracy**	**Fβ-Score**

`kernel: linear`
`nu: 0.001`
`shrinking: True`
	94.97%	91.27%	99.2%	91.74%	98.49%

`kernel: linear`
`nu: 0.1`
`shrinking: True`
	91.16%	94.38%	98.65%	93.97%	98.28%

`gamma: scale`
`kernel: poly`
`degree: 3`
`nu: 0.1`
`shrinking: True`
	90.85%	92.38%	98.58%	92.19%	98.04%

`gamma: scale`
`kernel: poly`
`degree: 2`
`nu: 0.1`
`shrinking: True`
	90.55%	92.45%	98.53%	92.21%	98.00%

`gamma: scale`
`kernel: poly`
`degree: 2`
`nu: 0.01`
`shrinking: True`
	89.33%	92.4%	98.35%	92.01%	97.83%

Table 7.2: Experiment 1 Results: Summary of model performance and chosen hyperparameters ($\beta = 0.5$).

By analyzing the results using the Fβ-Score as the performance criterion, with $\beta = 0.5$, the dominance of linear kernels, followed by low-degree (2 and 3) polynomial kernels, indicates that totalizer features exhibit a strong degree of linear separability.

The use of polynomial kernels, compared to the top-performing linear kernel, resulted in a 4.1% decrease in the True Negative Rate, accompanied by a slight 1% increase in the True Positive Rate. This trade-off caused a 0.6% reduction in Precision, underscoring the inherent challenge of optimizing bot detection accuracy while minimizing the misclassification of legitimate users.

When comparing linear kernels, an increase in the regularization parameter from 0.001 to 0.1 had a notable impact on model performance. Both True Negative and True Positive rates were affected, with reductions and increases of approximately 3%, respectively. This outcome aligns with expectations, as a higher regularization parameter establishes a softer decision boundary, thereby reducing False Positive occurrences and favoring

a balanced classification. Regarding the polynomial kernels, we observe that lower degrees, combined with a less strict decision boundary (`nu = 0.1`), yield improved results.

Experiment 2. In the second Experiment we added the Session Attributes (Table 6.1) to the totalizers, which includes temporal information through `sessionDuration` and `avgTransitionTime` and other contextual features such as `originPage` and `totalEvents`.

Parameters	TN	TP	Precision	Accuracy	Fβ-Score

`gamma: 0.01`
`kernel: poly`
`degree: 2`
`nu: 0.01`
`shrinking: True`
	99.09%	92.58%	99.86%	93.41%	99.22%

`gamma: 0.001`
`kernel: poly`
`degree: 3`
`nu: 0.01`
`shrinking: True`
	99.09%	92.51%	99.86%	93.35%	99.21%

`gamma: scale`
`kernel: poly`
`degree: 2`
`nu: 0.01`
`shrinking: True`
	99.09%	92.49%	99.86%	93.33%	99.21%

`gamma: 0.001`
`kernel: poly`
`degree: 2`
`nu: 0.01`
`shrinking: True`
	98.93%	92.51%	99.83%	93.33%	99.18%

`gamma: 0.1`
`kernel: poly`
`degree: 3`
`nu: 0.01`
`shrinking: True`
	98.93%	92.54%	99.83%	93.35%	99.18%

Table 7.3: Experiment 2 Results: Summary of model performance and chosen hyperparameters ($\beta = 0.5$).

The results presented in Table 7.3 highlight a significant shift in performance compared to the findings discussed in Experiment 1. Precision improved from 99.2% to 99.86% for the top 1 result in both experiments. This increase can be attributed to a marked improvement in the True Negative Rate—from 94.97% in Experiment 1 to 99.09% in the second experiment.

The incorporation of new features enhanced performance with polynomial kernels, suggesting that the added complexity of the data necessitated a more flexible decision

boundary, as a linear boundary proved inadequate for effective separation. While the precision score remained stable across the top three results, minor reductions were observed in the True Positive Rate. Additionally, both Accuracy and the Fβ-score saw slight decreases, with percentage reduction of -0.08% and -0.01%, respectively.

Experiment 3. Similar to the approach used in Experiment 1, this experiment focused exclusively on the Transition Attributes, which capture the relative frequency of the most relevant transitions occurring in each session. By isolating these features, we aimed to assess their individual contribution to the model's performance, free from the influence of other attribute subsets.

Parameters	TN	TP	Precision	Accuracy	Fβ-Score

`gamma: scale`
`kernel: rbf`
`nu: 0.01`
`shrinking: True`
	97.41%	91.49%	99.59%	92.25%	98.87%

`gamma: scale`
`kernel: rbf`
`nu: 0.001`
`shrinking: True`
	96.95%	91.76%	99.52%	92.42%	98.83%

`gamma: 1`
`kernel: rbf`
`nu: 0.1`
`shrinking: True`
	89.48%	93.31%	98.38%	92.83%	97.94%

`gamma: 0.001`
`kernel: rbf`
`nu: 0.1`
`shrinking: True`
	89.18%	93.03%	98.33%	92.54%	97.87%

`gamma: scale`
`kernel: rbf`
`nu: 0.1`
`shrinking: True`
	89.02%	92.87%	98.31%	92.38%	97.84%

Table 7.4: Experiment 3 Results: Summary of model performance and chosen hyperparameters ($\beta = 0.5$).

By using only Transition Attributes yielded a notable improvement of approximately 3% in the True Negative Rate, accompanied by a slight increase in precision of approximately 0.4%. Contrary to previous experiments where linear and polynomial kernels demonstrated superior performance, the top five performing models in this experiment exclusively employed the RBF kernel. This result strongly suggests that the RBF kernel is particularly effective at capturing the inherent clustering of users and bots within the feature space defined by the Transition Attributes. This implies that the patterns

captured by these attributes are not linearly separable and exhibit complex, non-linear relationships that the RBF kernel is better equipped to model.

Experiment 4. In this experiment, we extended the attribute set used in Experiment 3 by incorporating the "Session Attributes," which encapsulate general characteristics of each session.

Parameters	TN	TP	Accuracy	F1-Score	Fβ-Score

`gamma: 0.001`
`kernel: poly`
`degree: 3`
`nu: 0.01`
`shrinking: True`
	99.09%	92.51%	99.86%	93.35%	99.21%

`gamma: 0.01`
`kernel: poly`
`degree: 2`
`nu: 0.01`
`shrinking: True`
	99.09%	92.49%	99.86%	93.33%	99.21%

`kernel: linear`
`nu: 0.01`
`shrinking: True`
	99.09%	92.38%	99.86%	93.23%	99.20%

`gamma: 0.001`
`kernel: poly`
`degree: 4`
`nu: 0.01`
`shrinking: True`
	98.93%	92.56%	99.83%	93.37%	99.19%

`gamma: 0.001`
`kernel: poly`
`degree: 2`
`nu: 0.01`
`shrinking: True`
	98.93%	92.51%	99.83%	93.33%	99.18%

Table 7.5: Experiment 4 Results: Summary of model performance and chosen hyperparameters ($\beta = 0.5$).

The results are intriguing, as they closely align with those obtained in Experiment 2, exhibiting only marginal differences in metrics such as `TN`, `TP`, and `F`β`-Score`. Given that this experiment leverages the relative frequencies of the most common transitions identified in State transitions while Experiment 1 computes the frequency of distinct events, it is possible that these different subsets of features act as proxies for each other, capturing overlapping behavioral patterns.

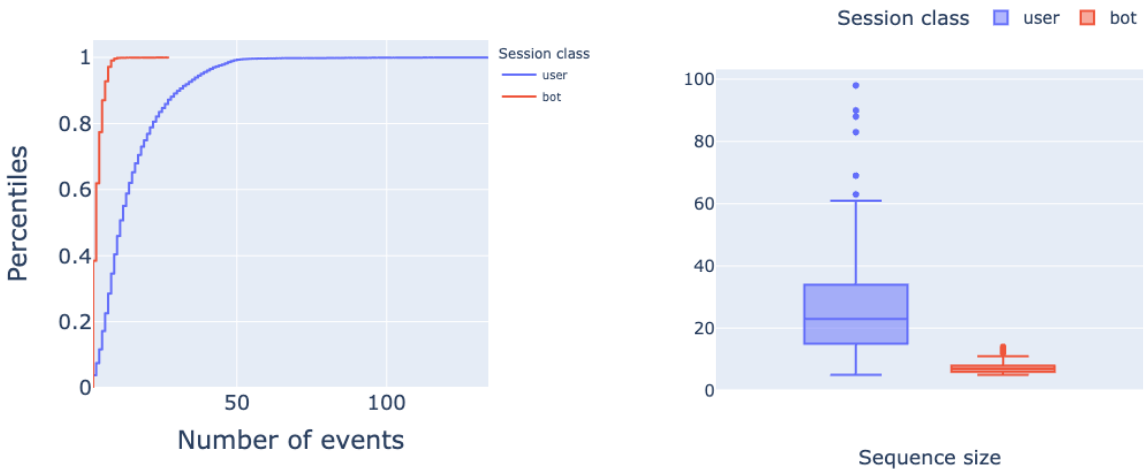
7.2 Results Analysis

In this section, we utilize the models with best result in Experiment 4, as it presented the better performance when compared to the others. Subsequently, we re-run some analyses from Chapter 5 to examine whether any notable differences emerge between instances predicted as anomalous and those classified as non-anomalous. This analysis serves as a measure of our solution's effectiveness in detecting bots within the e-commerce context, providing valuable insights into its practical applicability.

gamma	**kernel**	**degree**	**nu**	**shrinking**
0.001	poly	3	0.01	True

Table 7.6: Selected parameters

The One-Class SVM (OC-SVM) model was trained using the parameters detailed in Table 7.6. Unlike the experimental phase, where only 60% of user events were used for training, this final model was trained on the entire dataset of legitimate user events, as the optimal parameters had already been determined. This decision allowed us to maximize the number of training examples, improving the model's ability to characterize normal user behavior and, consequently, enhancing its anomaly detection capabilities. By leveraging all available normal data, the model is expected to learn a more robust and representative decision boundary, increasing its sensitivity to deviations indicative of bot activity.



(a) (b)

Figure 7.1: Total of events per (a) agent and (b) session.

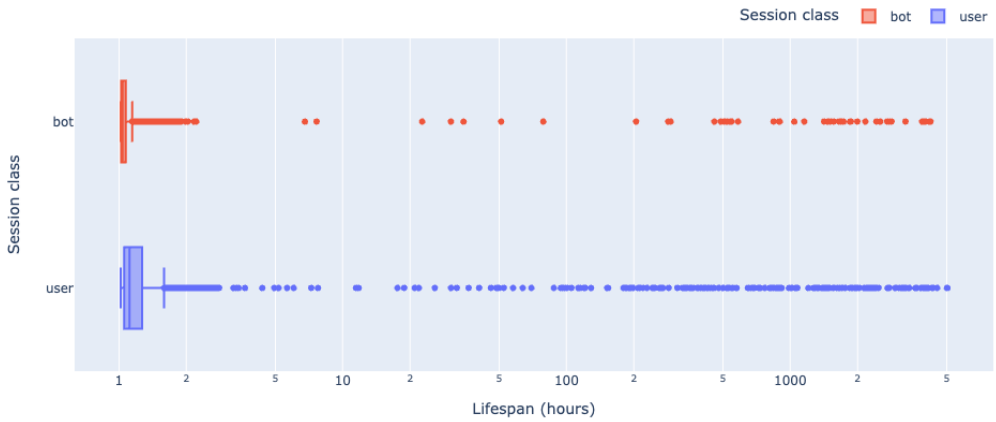
Our initial analysis focused on the frequency of events generated by each predicted class, mirroring the approach employed in the exploratory data analysis (EDA) presented

in Figure 5.3. A comparison of these results reveals a striking similarity in event frequency patterns between the predicted classes (see Figure 7.1a) and those observed during the EDA phase.

To complement this analysis, we examined the sequence size for each class—i.e., the number of events per session in each group. The results, displayed in Figure 7.1b, highlight a clear distinction: users tend to interact more extensively with the stores during their sessions, while bots exhibit significantly shorter sequences of events. This observation reinforces the behavioral differences between the two groups, with legitimate users demonstrating more engaged and prolonged browsing sessions compared to bots.

The similarity between the results observed during the EDA and the model predictions is particularly remarkable, given that event frequencies were not directly utilized as features in the model. However, as illustrated in Figures 7.1a and 7.1b, actors who interact more within their sessions naturally generate a higher number of events. Thus, even though the model was trained and made predictions at the session level, the sequence size—implicitly captured through totalizer attributes—appears to reflect this behavior. This suggests that the model may have indirectly learned to associate sequence length with session activity patterns, aligning its predictions with the event frequency trends observed in the exploratory phase.

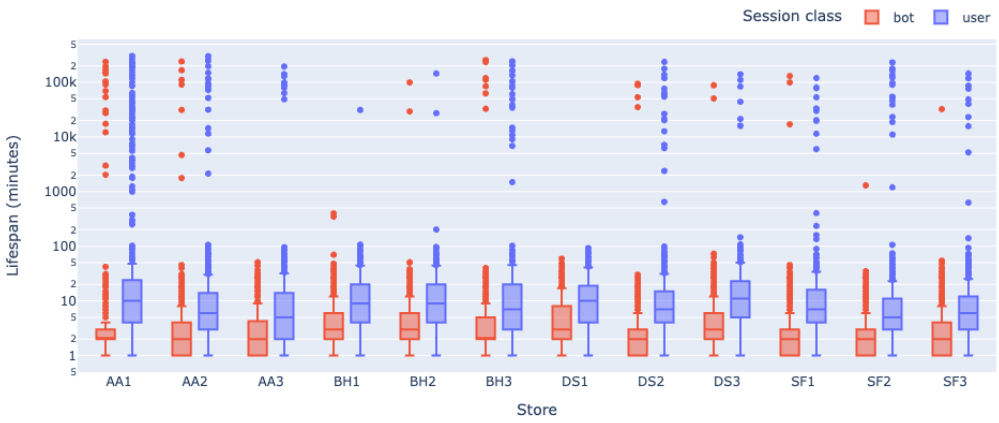
Figure 7.2: Lifespan for Predicted Actors



Next, we analyze the actors lifespan, defined as the interval between their first and last interactions. As discussed in Chapter 5, legitimate users tend to exhibit a longer lifespan compared to malicious actors. In our results, although the difference is more subtle, a similar pattern is observed: users access the stores over a more extended period than bots. This finding further supports the hypothesis that legitimate users engage in more sustained activity, while bots tend to interact in a more transient manner.

This behavior can be also noticed when analyzing stores individually as illustrated in Figure 7.3. For all analyzed stores we found the same behavior when analyzing it from a general perspective.

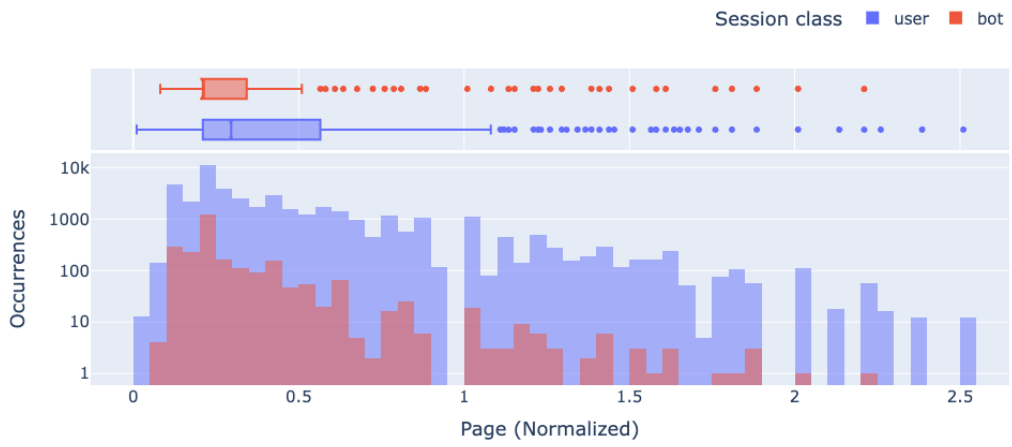
Figure 7.3: Lifespan for Predicted Actors - individual perspective



While the exploratory data analysis (Figure 5.9) revealed that certain bots tend to navigate more deeply through paginated content, our model predictions did not replicate this specific behavior. However, the predominant trend, where bots navigate less than users, was successfully captured, as shown in Figure 7.4. Bot activity is primarily concentrated on the initial pages, with a median normalized value of 0.21 compared to 0.30 for users.

The higher normalized values observed in the model's predictions compared to those in the EDA can be attributed to differences in the datasets analyzed. The EDA focused on users who completed purchases, likely reflecting a more deliberate navigation pattern aimed at finalizing an order. In contrast, bots detected by the Web Application Firewall (WAF) are restricted in scope due to the WAF's inherent limitations.

Figure 7.4: Pagination for predicted actors

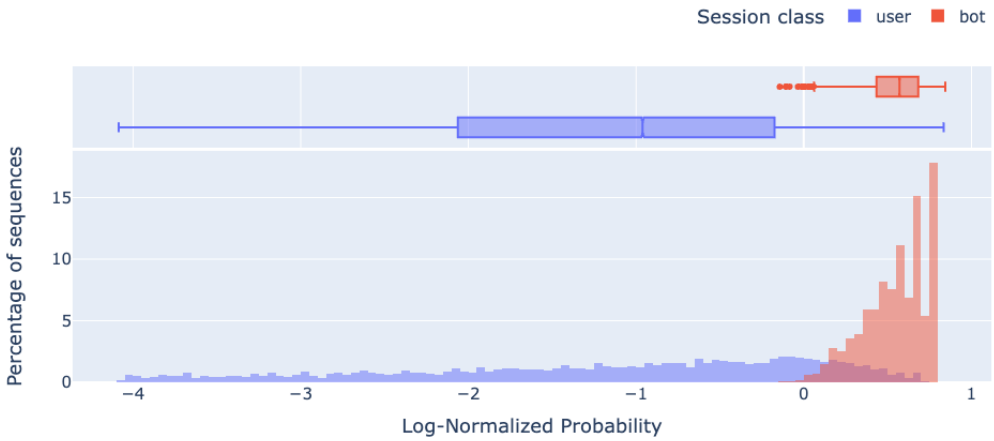


In the results shown in Figure 7.4, both users and bots exhibit normalized page values exceeding 1. Since the normalization is based on the 90th percentile, the agents on the right hand side of the plot represent the top 10% who navigate more deeply—i.e.,

beyond 90% of other agents. This behavior aligns with the previously stated hypothesis: the unlabeled sample encompasses a more diverse range of behaviors compared to the examples used during the training phase.

Finally, when analyzing the Sequence Probabilities (as we did in Chapter 6), we found the same behavior. Sequence sessions identified as anomalous present higher probability of occurring when modeling the expected behavior of legitimate sequences as illustrated in Figure 7.5. As mentioned previously, a possible explanation for that behavior is that bots tend to frequently perform "highly likely transitions".

Figure 7.5: Pagination for predicted actors



As detailed in Chapter 6, to address this limitation, we analyzed the most relevant transitions for each class and incorporated them as attributes for each session. This process involved calculating the relative frequency of these transitions within each session sequence. This approach led to the results observed in Experiment 4, which closely mirrored those obtained in Experiment 2.

The observed similarity in results between Experiment 2 and Experiment 4 is attributed to the shared characteristic of their feature representations. In both experiments, the training features—state frequencies in Experiment 2 and transition frequencies in Experiment 4—focused solely on counts, thereby omitting information about the sequential order of events.

Despite the limitations of frequency-based representation in capturing the full sequential structure of user sessions, the results obtained in Experiment 2 (leveraging totalizers and session attributes) and Experiment 4 (combining transitions and session attributes) demonstrate the model's ability to effectively identify key behavioral patterns distinguishing legitimate users from bots. These findings suggest that the types of pages frequently accessed by each group offer valuable discriminatory insights, even without incorporating the sequential context.

Chapter 8

Conclusions and Future Work

The increasing prevalence of automated and malicious bot activity necessitates a deep understanding of their behavioral patterns. This study offers valuable insights into the distinct characteristics of legitimate users and malicious bots in the e-commerce context. Through an extensive analysis of navigation data, encompassing interaction patterns and temporal aspects such as session duration and average transition times, we identified key behavioral distinctions between these groups.

Our methodology, rooted in exploratory data analysis, enabled the design of novel, informative features for machine learning models. This approach proved effective in uncovering patterns that differentiate bots from users, providing a robust foundation for understanding agent behavior and identifying relevant features for subsequent model training.

By employing a transition matrix derived from user behavior and modeled as a Continuous Time Markov Chain (CTMC), we computed the sequence probability for each session. However, as shown in Figure 6.4, sequence probabilities did not significantly enhance model performance. Unexpectedly, bot session probabilities were higher than anticipated, despite prior assumptions that bot sequences would exhibit lower probabilities when compared to the reference model of user behavior.

This unexpected result prompted a deeper investigation, shifting the focus from entire session sequences to individual state transitions. The analysis uncovered intriguing patterns, including exclusive transitions unique to both users and bots, as depicted in Figures 6.7a and 6.7b. These findings informed the creation of "Transition Attributes" (see Table 6.1) and a reevaluation of our approach.

Despite notable differences in exclusive transitions, common transitions posed a significant challenge (Figure 6.5). For instance, the `productView → productView` transition emerged as the most frequent in both groups. Analyzing the time spent on this transition revealed considerable overlap between users and bots, highlighting the limitations of relying solely on transition states to distinguish between the two groups.

To address this challenge, we expanded our analysis beyond sequence probabilities and transition attributes by introducing additional features such as the number of accessed pages, session duration, and origin page—referred to as "Session Attributes." During the exploratory data analysis, these attributes revealed significant behavioral differences,

enabling a more comprehensive framework for distinguishing users from bots.

In our experiments (see Chapter 6), combining Session Attributes with Totalizers and Transition Attributes yielded the most effective results. These findings confirmed our hypothesis that relying solely on transition and state-based aspects is insufficient for producing meaningful distinctions. Consequently, the developed One-Class SVM model emerged as a powerful tool for identifying anomalous behaviors in agent navigation, effectively detecting deviations from established patterns of legitimate user activity.

Revisiting the questions raised in Section 1.2, we conclude that the extensive Exploratory Data Analysis revealed significant differences between users and bots in the navigation context. These differences are particularly notable in state transitions, page counts, and broader aspects such as session duration and the number of accessed pages. On the other hand, the transition probabilities derived from CTMCs proved not to be a relevant feature for discriminate between legitimate and malicious agents. Finally, the OC-SVM demonstrated to be a highly effective mechanism for detecting anomalous behavior in the e-commerce context, achieving an accuracy of up to 99%. The predicted classes for the unlabeled data also exhibited distinct behavioral patterns for both groups. This evidence supports the assertion that it is possible to attain good performance in detecting anomalous behavior using solely navigation data.

Despite the significant results, the findings lack validation by experts – an essential step to more accurately assess the model's performance and establish confidence in its reliability. Furthermore, our approach is limited by the reliance on WAF-generated labels and the "made a purchas" criterion for bot and user labeling. The WAF mechanism only detects a narrow subset of malicious behaviors, and assuming that all purchasing users are legitimate may fail to account for sophisticated bots capable of mimicking legitimate user actions. This reliance potentially limits our ability to capture the full diversity of user behaviors, introducing biases into the analysis.

Additionally, our study does not explicitly utilize agents labeled as bots, either in the Continuous-Time Markov Chain (CTMC) modeling or during the training of the One-Class Support Vector Machine (OC-SVM). Instead, we focused on modeling legitimate user behavior and detecting anomalies based on deviations from these patterns. While this approach reduces the risk of overfitting to the specific bot behaviors captured by the Web Application Firewall (WAF), it may limit the model's ability to detect more nuanced or unconventional bot activities that fall outside the scope of legitimate behavior deviations. This trade-off reflects a deliberate choice to prioritize generalization and robustness over reliance on potentially biased or incomplete bot labels. A further constraint of this study is the significant volume of "otherView" events present in the navigation data, as they offer minimal actionable insights, making it difficult to extract meaningful conclusions from their analysis.

Lastly, while we focused on the One-Class Support Vector Machine (OC-SVM)

for anomaly detection, experimenting with other one-class classifiers, such as Isolation Forests or Deep Autoencoders, could have offered additional perspectives and potentially improved model performance. Future research could explore these alternative approaches to refine the detection of anomalous behaviors in e-commerce environments.

Concurrent with this study, new data sources have become available, including mouse movement data and search queries. Future research will therefore investigate the potential of these attributes to further enhance bot detection accuracy. Specifically, the incorporation of mouse movement data could provide valuable insights into user interaction styles, as bots often exhibit less natural mouse trajectories and movements compared to human users. Similarly, analyzing search queries could reveal patterns indicative of automated behavior, such as repetitive searches or queries related to specific exploits.

In addition to these newly introduced features, future research will focus on exploring more granular time-based attributes, such as the time spent on specific page elements and the distribution of inter-click intervals. A finer-grained temporal analysis could provide deeper insights into distinguishing human users, who naturally exhibit pauses and reflective browsing patterns, from bots, which often perform actions at consistent and rapid intervals. Furthermore, we plan to investigate second and third-order Markov Chains to enhance our capability to model more complex dependencies in navigation patterns, potentially improving the detection of anomalous behaviors and uncovering subtler distinctions between bots and legitimate users.

References

[1] Amna Ahmed, Rohail Qamar, Raheela Asif, Muhammad Imran, Muhammad Khurram, and Saad Ahmed. Dead internet theory. *Pakistan Journal of Engineering, Technology & Science*, 12:37–48, 6 2024.

[2] Bushra A. Alahmadi, Enrico Mariconti, Riccardo Spolaor, Gianluca Stringhini, and Ivan Martinovic. Botection: Bot detection by building markov chain models of bots network behavior. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2020*, pages 652–664. Association for Computing Machinery, Inc, 10 2020.

[3] Maroua Bahri, Flavia Salutari, Andrian Putina, Mauro Sozio, and Mauro Sozio AutoML. Automl: state of the art with a focus on anomaly detection, challenges, and research directions. *International Journal of Data Science and Analytics*, 2022.

[4] Andoena Balla, Andoena Balla, Athena Stassopoulou, Athena Stassopoulou, Marios D. Dikaiakos, and Marios D. Dikaiakos. Real-time web crawler detection. *International Conference on Telecommunications*, 2011.

[5] Lothar Breuer and Dieter Baum. *An Introduction to Queueing Theory and Matrix-Analytic Methods*. Springer, 2005.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique, 2002. semi supervised.

[8] Pawel Chwalinski. Detection of unsolicited web browsing with clustering and statistical analysis, 2013. semi supervised.

[9] Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam, and Raouf Boutaba. Botchase: Graph-based bot detection using machine learning. *IEEE Transactions on Network and Service Management*, 17:15–29, 3 2020.

[10] Manuel Gómez-Rodríguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):1–37, 2011.

[11] Katerina Goseva-Popstojanova, Goce Anastasovski, Ana Dimitrijevikj, Risto Pantev, and Brandon Miller. Characterization and classification of malicious web traffic. *Computers and Security*, 42:92–115, 5 2014.

[12] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, 1997.

[13] Imperva. Bad bot report. Technical report, Imperva, 2022. Accessed: 2024-12-22.

[14] Imperva. The state of security within ecommerce in 2022. Technical report, Imperva, 2022. Accessed: 2024-12-22.

[15] Hongwen Kang, Kuansan Wang, David Soukal, Fritz Behr, and Zijian Zheng. Large-scale bot detection for search engines. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pages 501–510, 2010.

[16] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

[17] Christopher Kruegel, Christopher Kruegel, Giovanni Vigna, Giovanni Vigna, William Robertson, and William Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 2005.

[18] H. Liu et al. Ensemble learning for web bot detection. *Journal of Web Engineering*, 2020.

[19] Yang Liu, Xiangning Ning, Hao Xu, and Yixin Chen. Real-time anomaly detection for streaming data using probabilistic graphical models. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):909–922, 2019.

[20] C. Lu et al. Hybrid unsupervised approach for bot detection in social media. *Social Media Studies*, 2020.

[21] Morgan Meaker. How bots corrupted advertising. https://www.wired.com/story/bots-online-advertising/. Acesso em 09/10/2023.

[22] Pratik Narang, Pratik Narang, Chittaranjan Hota, Chittaranjan Hota, V. Venkatakrishnan, and V. N. Venkatakrishnan. Peershark: flow-clustering and conversation-generation for malicious peer-to-peer traffic identification. *Eurasip Journal on Information Security*, 2014.

[23] Ngoc Thanh Nguyen, Bogdan Trawiński, and Raymond Kosala. Intelligent information and database systems 7th asian conference, aciids 2015 bali, indonesia, march 23–25, 2015 roceedings, part ii, 2015. supervised.

[24] J. R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[25] Nicolás Poggi, J. L. Berral, Toni Moreno, Ricard Gavaldà, and J. Torres. Automatic detection and banning of content stealing bots for e-commerce. *null*, 2007.

[26] Rizwan Rahman and Deepak Tomar. Threats of price scraping on e-commerce websites: attack model and its detection using neural network. *Journal of Computer Virology and Hacking Techniques*, 17, 03 2021.

[27] Lucky Rajput and Shailendra Narayan Singh. Customer segmentation of e-commerce data using k-means clustering algorithm. In *2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 658–664, 2023.

[28] A. Rasti et al. Deep learning-based bot detection in e-commerce. *Computational Intelligence and Neuroscience*, 2021.

[29] William Robertson, William Robertson, Giovanni Vigna, Giovanni Vigna, Christopher Krügel, Christopher Krügel, Richard A. Kemmerer, and Richard A. Kemmerer. Using generalization and characterization techniques in the anomaly-based detection of web attacks. *Network and Distributed System Security Symposium*, 2006.

[30] Stefano Rovetta, Grażyna Suchacka, and Francesco Masulli. Bot recognition in a web store: An approach based on unsupervised learning. *Journal of Network and Computer Applications*, 157, 5 2020.

[31] Lukas Ruff, Jörn Kauffmann, Roland Vandermeulen, Nils Görnitz, Eskil Müller, and Marius Kloft. Deep one-class classification. *Proceedings of the 35th International Conference on Machine Learning*, 80:4393–4402, 2018.

[32] Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alexander J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[33] Yashwant Singh Shamsul Haq. *Botnet Detection using Machine Learning*. IEEE, 2019.

[34] Ning Su, Yiqun Liu, Zhao Li, Yuli Liu, Min Zhang, and Shaoping Ma. Detecting crowdturfing "add to favorites" activities in online shopping. In *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*, pages 1673–1682. Association for Computing Machinery, Inc, 4 2018.

[35] Grażyna Suchacka, Alberto Cabri, Stefano Rovetta, and Francesco Masulli. Efficient on-the-fly web bot detection. *Knowledge-Based Systems*, 223, 7 2021.

[36] David M. J. Tax and Robert P. W. Duin. One-class classification. *International Conference on Artificial Neural Networks*, pages 131–136, 2001.

[37] D. Tran et al. Deep semi-supervised learning for bot detection using autoencoders. *Computational Intelligence*, 2021.

[38] Gang Wang, Xinyi Zhang, Shiliang Tang, Christo Wilson, Haitao Zheng, and Ben Y. Zhao. Clickstream user behavior models. *ACM Transactions on the Web*, 11, 7 2017.

[39] Chang Xu, Zhiqiang Wang, Fang Xu, and Fei Liu. A survey of one-class classification algorithms. *Pattern Recognition*, 98:107045, 2020.

[40] Haitao Xu, Zhao Li, Chen Chu, Yuanmi Chen, Yifan Yang, Haifeng Lu, Haining Wang, and Angelos Stavrou. Detecting and characterizing web bot traffic in a large e-commerce marketplace. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11099 LNCS, pages 143–163. Springer Verlag, 2018.

[41] X. Zhang et al. Graph-based semi-supervised learning for bot detection in large-scale networks. *Neural Networks*, 2020.

[42] L. Zhao et al. An unsupervised framework for bot detection in financial transactions. *Journal of Financial Technology*, 2022.