A MULTI-CLASS APPROACH TO NEXT SESSION AND IN-SESSION PURCHASE PREDICTION WITH REAL-TIME E-COMMERCE DATA USING MACHINE LEARNING TECHNIQUES

by

Gizem Sürhan

B.S., Management Information Systems, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computational Science and Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I would like to thank, first of all, my whole family who have supported me in every way throughout my education life.

To all my teachers, especially my thesis advisor, Prof. Bertan Badur, who raised me and played an inspirational role by advising me to reach where I am today.

To all my friends who made me who I am and who influenced my life, whether I noticed it or not.

Finally, for whom I share the same fate and whom I am grateful to have in my life every day. To my husband Onur Sürhan, for his support, courage, motivation, and everything.

ABSTRACT

A MULTI-CLASS APPROACH TO NEXT SESSION AND IN-SESSION PURCHASE PREDICTION WITH REAL-TIME E-COMMERCE DATA USING MACHINE LEARNING TECHNIQUES

Advances in machine learning yield implications for the rapidly growing e-commerce industry. Retailers are looking for ways to better understand and predict complex customer behavior. Two crucial problems that exist in the domain are predicting customers' platform engagement and purchase intent. Different techniques are employed in the literature addressing the problems separately, but the two tasks are highly dependent on each other since the ultimate goal for session engagement is purchase. Understanding if a next session will be made with a high purchase motive is critical to diversify the business actions taken. The main aim of the thesis is to develop a multi-class model that successfully distinguishes the next sessions with and without purchase intention. 38 million e-commerce sessions are collected for the specific task. Following the application of state-of-the-art LightGBM and LSTM algorithms, their results are compared, where LightGBM outperformed the latter. Additionally, a simple ensembling technique is used to increase the performance, leading to a 68% F1 score for the predictions of no session, 71% for the predictions of sessions without purchase and 59% for the predictions of sessions with purchase. Furthermore, an undersampling technique is employed to handle the imbalance differently than the technique used by LightGBM and LSTM, increasing F1 scores to 75%, 72% and 74% respectively.

ÖZET

MAKİNE ÖĞRENMESİ MODELLERİ İLE GERÇEK ZAMANLI E-TİCARET DATASINI KULLANARAK GELECEK OTURUM VE OTURUM İÇİ SATIN ALIM TAHMİNİ

Makine öğrenimindeki gelişmeler, hızla büyüyen e-ticaret endüstrisi için çalışma alanları doğurmaktadır. Perakendeciler, karmaşık müşteri davranışlarını daha iyi anlamanın ve tahmin etmenin yollarını aramaktadır. Karşılaşılan iki önemli problem, müşterilerin platforma tekrar gelip gelmeyeceğini ve satın alma niyetini tahmin etmektir. Literatürde sorunları ayrı ayrı ele alan farklı teknikler kullanılmaktadır, ancak platforma gelmenin nihai amacı satın alma olduğundan, iki görev birbirine oldukça bağlıdır. Bir sonraki seansın yüksek bir satın alma güdüsüyle olup olmayacağını anlamak, alınan iş aksiyonlarını çeşitlendirmek için kritik öneme sahiptir. Tezin ana amacı, satın alma niyeti olan ve olmayan sonraki oturumları başarılı bir şekilde ayıran çok sınıflı bir tahminleme modeli geliştirmektir. 38 milyon e-ticaret oturumu toplandı. LightGBM ve LSTM algoritmalarının uygulanmasının ardından, sonuçlar karşılaştırılarak LightGBM'in daha iyi performans gösterdiği görüldü. Performansı artırmak için bir birleştirme tekniği kullanılarak tekrar platforma gelmeyecek şeklinde yapılan tahminler için %68, satın almasız yapılacak seans tahminleri için %71 ve satın alma ile sonlanacak seans tahminleri için %59 F1 puanı elde edildi. Verisetinin dengesizliğini LightGBM ve LSTM tarafından kullanılan teknikten farklı bir şekilde ele almak için düşük örnekleme tekniği kullanıldı ve F1 puanlarının sırasıyla %75, %72 ve %74'e yükseldiği gözlemlendi.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES ix
LIST OF ACRONYMS/ABBREVIATIONS x
1. INTRODUCTION 1
1.1. Background Work 1
1.2. Dataset Types 3
1.3. Feature Extraction Techniques 4
1.4. Predictive Models 4
1.4.1. LightGBM 6
1.4.2. LSTM 8
1.5. Aim 9
2. METHODOLOGY 11
2.1. Data Collection 11
2.2. Target Variable Creation 12
2.3. Feature Extraction 13
2.3.1. Extraction of Bulk Features 17
2.3.2. Extraction of Sequence Features 18
2.3.3. Different Data Structure for Different Models 19
2.4. Dimensionality Reduction 19
2.5. Handling Outliers 21
2.6. Undersampling 22
2.7. Train Test Split 23
2.8. Hyperparameter Optimization 23
2.9. Cross Validation 26
2.10. Performance Evaluation Metrics 26

2.10.1. Error Metrics . . . 27
2.10.2. Performance Metrics . . . 29
2.11. Modelling . . . 30
2.11.1. LightGBM . . . 30
2.11.2. LSTM . . . 33
2.11.3. Ensemble . . . 34
3. RESULTS . . . 36
3.1. Multi-Log Loss . . . 36
3.1.1. LightGBM Results . . . 36
3.1.2. LSTM Results . . . 36
3.2. Performance Metrics . . . 37
3.2.1. LightGBM Results . . . 37
3.2.2. LSTM Results . . . 38
3.2.3. Ensemble Model Results . . . 40
4. DISCUSSION . . . 42
4.1. Comparing the Model Performances for the Non-Sampled Data . . . 42
4.2. Assessing The Effect of Undersampling on the Model Performances . . . 43
4.3. Choosing the Best Model . . . 44
5. CONCLUSION . . . 46
REFERENCES . . . 47
APPENDIX A: MONETARY FEATURES . . . 53
APPENDIX B: EVENT FEATURES . . . 55
APPENDIX C: RECENCY FEATURES . . . 57
APPENDIX D: DEMOGRAPHICS FEATURES . . . 58
APPENDIX E: SEQUENCE FEATURES . . . 59

LIST OF FIGURES

Figure 2.1. Illustration of current sessions and their preceding. 18

Figure 2.2. Number of leaves hyperparameter space. 25

LIST OF TABLES

Table 2.1. Target ratio. 13

Table 2.2. Illustration of the data structure. 13

Table 2.3. Feature groups and domains. 16

Table 3.1. LightGBM multi log loss. 36

Table 3.2. LSTM multi log loss. 36

Table 3.3. LightGBM precision. 37

Table 3.4. LightGBM recall. 38

Table 3.5. LightGBM F1. 38

Table 3.6. LSTM precision. 39

Table 3.7. LSTM recall. 39

Table 3.8. LSTM F1. 40

Table 3.9. Ensemble model precision. 40

Table 3.10. Ensemble model recall. 41

Table 3.11. Ensemble model F1. 41

LIST OF ACRONYMS/ABBREVIATIONS

DART	Dropouts meet Multiple Additive Regression Trees
EFB	Exclusive Feature Bundling
FN	False Negatives
FP	False Positives
GBDT	Gradient Boosted Decision Trees
GOSS	Gradient-based One-Side Sampling
GRU	Gated Recurrent Unit
IQR	Interquartile Range
LightGBM	Light Gradient-boosting Machine
LSTM	Long Short-term Memory
OSCF	Optimal Split for Categorical Features
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
SVD	Singular Value Decomposition
TN	True Negatives
TP	True Positives
TPE	Tree-structured Parzen Estimator
XGBoost	eXtreme Gradient Boosting

1. INTRODUCTION

Businesses have lots of options to analyze their customers' buying habits thanks to daily online actions on e-commerce platforms [1]. In fact, total spending online was $2.86 trillion in 2018, which is 18% more than that of 2017. To take targeted actions on this fast-growing world, more and more companies aim to predict the customers' needs to offer personalized services [2, 3].

One of the challenges of this task is that consumer behavior is so complex to predict [4]. To predict the likelihood of consumer actions, machine learning models are used utilizing customers' historical data [5, 6]. One of the most important actions is the purchase behavior of the customers.

There are mainly three tasks for purchase prediction in the e-commerce industry: Predicting customer intent, predicting buying session and predicting purchase decisions [7]. The former is for understanding the intention of consumer visits to the online platform. Examples are browsing, searching, purchasing, and bouncing. This type of task is essentially used to identify similar groups of customers for the purpose of creating customer segments. On the other hand, buying session prediction is to predict if a customer will make a purchase during a session or not. This task is of great importance to get an understanding of the consumer conversion, which is the situation where a website visit will end up in a sales. Finally, the latter is the prediction of the buying behavior in a more detailed matter. That is, not only if a customer will make a purchase is predicted, but also which product or category she/he will buy, when she/he will perform the transaction, what amount will be spent etc. are predicted.

1.1. Background Work

Purchase intent prediction is the forecast of the intention towards purchase using the subsequent actions of a customer [8]. Before making a purchase, customers spend

considerable amount of time looking for the best available products that can meet their needs, and they engage with several sessions from several devices in different e-commerce sites [9]. These sessions are nested and some of them takes much longer time than others do. The purchase intent may depend on more than one session, in such cases, a session cannot be analyzed without the findings from the previous sessions.

There is much research on the recurring visits of customers in the literature. E-commerce sites collect the visitors' online browsing data over time, among other significant information, to acquire knowledge of their behavior and purchase intent. Clickstream data is considered as one of the critical components in the literature [10–13]. In a study, different types of customers are identified using their clickstream data [14]. With the usage of k-means clustering algorithm, users are labeled as shallow, knowledge building, hedonic browsing, search/deliberation and directed buying, to indicate the likelihood of purchase. While the shallow customers represents the visitors not coming back to the website after 2 visits, directed buyers are the customers who come to the website for the purpose of purchase. In another study, the dynamics of the purchasing behavior is analyzed by modelling the conversion probabilities of different segments of customers [15]. In a separate inquiry, subsequent visits to the platform is modeled as a Markov chain and it is suggested that six prior sessions should be considered important for the purchase intention prediction task [16]. In a separate project, the tasks that are required for the purchase are identified and the probability of completion for each task is predicted using Markov chain Monte Carlo methods [17].

In a recent study, a clustering algorithm utilizing an integrated similarity function for the three indicators of a user's interest consisting of category visiting path, visiting frequency, and relative duration, is proposed [18]. While the category visiting path is the sequence of categories visited, visiting frequency is defined as the ratio of the number of visits to a category to the length of the user's browsing path. On the other hand, relative duration is the ratio of time spent on a category in a session. It is claimed that the newly proposed approach outperformed the traditional clustering algorithms. In an additional study, an approach described as a generalization of Hidden

Markov Models, which is employed to model a range of latent changes in customer behavior, is proposed [19]. While the transitions between states occur according to the Markov process in HMMs, in the proposed model, they depend on latent states of multiple periods where the order of dependence changes over time. In other words, the visit sequences are approached as clusters where close visits go into the same cluster. Furthermore, it is suggested that inside the clusters, the probability of a purchase taking place is higher for later visits. The potential of graph theory is also studied by representing sessions as navigation graphs [20]. For this purpose, graph metrics are calculated and it is claimed that these metrics substantially increased the performance of the Generalized Linear Model and Random Forest models. In a different scientific examination, the importance of the length of the first visit is found to be an important predictor of future sessions, leading to the use of navigation information and visit dynamics, in addition to temporal information like the time of the day [21].

1.2. Dataset Types

There are mainly five segments of data used for the purpose of purchase prediction in the literature, related to customer, product, time, channel and location data [7].

(i) Customer-related data:
- Demographics data such as age, gender, income.
- Clickstream data consisting of the clicks performed by users.
- Session-related data showing the starting and ending time, duration, activities during session.

(ii) Product-related data: Properties, value and status of availability.

(iii) Time-related data: The season, year, week, day, hour of the transactions and returns.

(iv) Channel-related data: The type and brand of the device, operating system used to connect to the website.

(v) Location-related data: The name and characteristics of the country, city and neighborhood.

1.3. Feature Extraction Techniques

Two types of techniques adopted for purchase prediction task in the literature [7]. The first one is feature engineering whereas the second one is feature learning.

(i) Feature engineering:
- Rule-based: The method of creating binary features to show if a condition is satisfied or not. An example variable is if a transaction happened at the first visit of a customer or not.
- Aggregation-based: The variable creation technique by the usage of mathematical aggregations, such as average, sum and count. For example average time spent on the sessions is an aggregation-based derived feature.
- Personalized functions: The construction of novel approaches, such as entropy to determine the diversity of customer clicks under the same query.

(ii) Feature learning: This approach is used for non-linear machine learning models. It creates representations of the input data by utilizing complex functions, such as auto-encoders and recurrent neural networks.

1.4. Predictive Models

There are a lot of research with various kinds of predictive models for the purchase prediction task. Not only traditional data mining approaches are used, but also new machine learning techniques and advanced deep learning methodologies are employed.

Unsupervised clustering methods are widely used at the beginnings of the literature. Sessions and transactions are used to discover patterns in customers, resulting in the identification of purchase-inclined ones. K-means algorithm is used to segment the customers by exploiting clickstream data [22,23]. Association rules are also used to extract associations between features and create rules to predict purchase. In a study, customers are grouped into two as innovative customers and tradition customers according to their usage of specific product types, leading to the discovery of different

associations for both groups, by utilizing the A-priori algorithm [24]. Instance-based methods are other type of predictive algorithms, which classifies new data points based on similar characteristics. K-Nearest Neighbor classification is applied to identify sessions as buying or browsing [25]. Linear machine learning models are also one of the widely used predictive model groups. These algorithms have the naive assumption that positive and negative classes can be separated with a linear decision boundary. There are also examples where non-linearity is introduced with the usage of kernel trick [26] or creation of polynomial combinations of features [27]. Because of the complexity of the human behavior, non-linear models outperform linear models, one of which is the ensemble learning technique. An example practice of ensembling technique is performed by ensembling tree-based C4.5, LMT, RandomForest and REPTree algorithms, rule-based JRip and Ridor algorithms with the MultiLayer Perceptron algorithm to predict the purchase with high performance [28]. The proposed model consisting of heterogenous algorithms with different abilities achieved 97.20% F-score.

Probabilistic classifiers that use the probability theory to make predictions are also used for the purchase prediction task. Bayesian Classifier is an algorithm that uses the input data to estimate conditional probability distributions. This approach has been effectively applied to build models to make conversion predictions by calculating probabilities for customer purchase [29]. Hidden Markov Model is another extensively used probabilistic model. In this model, the environment is considered to be a Markov process that has hidden states. Research has been conducted using this model to examine examined customers' shopping cart choices to predict their hidden shopping intent states [30].

Additionally, the collaborative filtering method, which is used for the recommendation systems, is another model used for conversion estimation. This approach has been utilized to create a purchasing matrix consisting of users as columns and different items as rows, which showed if a person had bought a specific item or not in a binary format, where transfer learning is used to handle the sparsity in the data, resulting in a moderate performance [31].

Deep learning models are abundantly used as well, especially in the recent literature. They are considered as the representation of the human brain, and can accomplish manifold tasks with superior performance. Certain studies have leveraged algorithms such as Long Short-term Memory algorithm, which is a type of Recurrent Neural Networks, and compared it to the XGBoost algorithm of Gradient Boosted Machines [32]. Although both of the models resulted in a very similar high performance, LSTM came forward with requiring no feature engineering.

1.4.1. LightGBM

LightGBM, a cutting-edge machine learning library created by Microsoft, has earned a reputation for its ability to efficiently process large-scale datasets and high-dimensional features. As a gradient boosting framework, LightGBM iteratively builds an ensemble of decision trees to minimize a specified loss function. Its performance, when compared with other gradient boosting libraries such as XGBoost and CatBoost, is remarkable due to its efficiency and scalability.

One distinctive feature of LightGBM is its utilization of a leaf-wise tree growth strategy, which is different from the more conventional level-wise approach. This method involves choosing the leaf with the most significant delta loss to split, guaranteeing an optimal reduction in the loss function. Although this strategy might lead to deeper trees, the resulting models are frequently more accurate, especially when dealing with extensive datasets.

The capacity of LightGBM to handle categorical features is another noteworthy aspect. By using an exclusive feature bundling technique, it effectively merges categorical features with similar values. This approach considerably decreases memory usage and accelerates the training process.

Furthermore, LightGBM integrates two advanced techniques to enhance training speed: Gradient-based One-Side Sampling and Exclusive Feature Bundling. GOSS maintains instances with larger gradients while subsampling instances with smaller gradients, thus preserving data information while reducing computational complexity. EFB groups exclusive features (those that are never non-zero simultaneously), decreasing the number of features and expediting the training process.

In addition to these techniques, LightGBM also supports early stopping, which helps prevent overfitting by halting the training process if the model's performance on a validation set does not improve for a specified number of iterations. This feature allows users to strike a balance between model complexity and generalization performance.

LightGBM offers a variety of error metrics to evaluate a model's performance, including log loss, multi-log loss, mean squared error, and area under the ROC curve. These metrics provide essential insights into the model's accuracy, enabling users to fine-tune hyperparameters and make informed decisions about model selection.

The library also supports various boosting types, such as Gradient Boosted Decision Trees, Dropouts meet Multiple Additive Regression Trees, and Gradient-based One-Side Sampling. Each boosting type has its advantages, and users should select the most suitable method based on their problem's requirements and dataset characteristics [33, 34].

In summary, LightGBM stands as a powerful gradient boosting library with a wide range of applications, spanning from e-commerce and finance to healthcare. Its unique leaf-wise tree growth strategy, advanced sampling techniques, categorical feature handling capabilities, and support for early stopping make it a top choice among machine learning practitioners seeking to develop accurate and scalable models. The library's mathematical foundation and optimization techniques contribute to its success, as they enable the creation of models with remarkable accuracy and efficiency.

1.4.2. LSTM

Long Short-Term Memory networks, a type of Recurrent Neural Networks, have gained popularity due to their ability to address the limitations of traditional RNNs when dealing with sequential data. These networks are especially adept at capturing long-range dependencies, making them suitable for various applications, such as natural language processing, time series prediction, and speech recognition.

A pivotal innovation in LSTM networks is the memory cell, a component designed to store and regulate information over extended sequences. The memory cell incorporates three gates—input, output, and forget gates—working in harmony to control the information flow within the network. By using these gates, LSTM networks can learn essential information and discard irrelevant details, effectively mitigating the vanishing gradient problem that hinders conventional RNNs.

These gates' interaction contributes to the network's ability to manage long-range dependencies. In essence, the input gate determines which information from the current input enters the memory cell, while the forget gate selects the information to remove. Simultaneously, the output gate modulates the cell's contribution to the network's output. This intricate cooperation between gates enables LSTM networks to excel in processing time series or language data.

One crucial aspect of LSTM networks is their mathematical foundation. The network relies on non-linear activation functions, such as the sigmoid and hyperbolic tangent functions, to compute the state of the gates and the memory cell. The use of these activation functions allows the LSTM network to capture complex relationships within the input data and learn more effectively from long sequences.

LSTMs have been integrated into various deep learning architectures, including encoder-decoder models, to address tasks like machine translation, text summarization, and sentiment analysis. In these models, two LSTM networks work together: an

encoder processes the input sequence and generates a fixed-size representation, while a decoder produces the output sequence based on the encoded representation. This arrangement has demonstrated remarkable performance in sequence-to-sequence problems.

Bidirectional LSTMs represent another significant development in this area. By processing input sequences in both forward and backward directions, bidirectional LSTMs capture information from past and future contexts, leading to a more comprehensive understanding of the input sequence. This bidirectional processing has found applications in areas like named entity recognition, part-of-speech tagging, and text classification.

Despite the numerous advantages, LSTMs are not without their limitations. They can be computationally demanding, especially for large-scale problems, and often require significant training time. Additionally, input sequences of extreme length may negatively impact the network's performance. To address these challenges, alternative architectures like the Gated Recurrent Unit have emerged, simplifying the LSTM structure while maintaining the ability to capture long-range dependencies [35, 36].

In summary, LSTM networks have revolutionized the field of machine learning for their capacity to process and predict sequential data effectively. Their unique memory cell structure and gating mechanisms allow them to surpass traditional RNNs' limitations and excel in a broad range of applications. While challenges persist, LSTM networks and their variants continue to push the boundaries of natural language processing, time series analysis, and other domains that demand an understanding of intricate, sequential data.

1.5. Aim

There had been many studies which tried to predict whether the customers will purchase or not in their next sessions. As aforementioned, these studies seemed to be

more accurate as the years passed. However, the main purpose, which is predicting the likelihood of purchase of the customer in the next session, has not been transformed in accordance with the business needs. Likelihood of occurrence of the next session is also a critical information for e-commerce companies to be utilized in different marketing strategies. There is no study in the literature which combines both predicting the likelihood of the purchase in the next session and the likelihood of the session's occurrence at the same time. In this thesis, the purchase prediction task is not considered as a single task. Whether the purchase will be made or not in the next session and next session's occurrence are aimed to be predicted at the same time through developing machine learning models.

Data size and freshness of the data are critical to assure the current applicability of machine learning models. In many of the aforementioned studies data sizes are not enough to represent the all population. Also, a long time passed over the publication times of these studies to say they can be outdated. In this thesis, the data that is used to develop ML models is big, and is from a fairly recent time period. By utilizing a recent data with a huge size, a desirable performance is aimed to reach with the power of state-of-the-art machine learning techniques.

LSTM and LightGBM models have proven success in predicting the next session purchase likelihood. Although there are many studies ensembling different machine learning techniques, there is no study combining LSTM and LightGBM models. In this thesis, through utilizing original feature extraction methods, sequential session data tailored to tabular format and aimed to fed into the LightGBM algorithm. Ultimately, the ensembling of the LSTM and LightGBM is developed and its performance is compared those of the individual models.

2. METHODOLOGY

2.1. Data Collection

The organization whose data is being analyzed utilizes Google BigQuery as its primary data warehouse, facilitating the storage and management of vast quantities of data. A multitude of real-time event tables is available in BigQuery, ensuring comprehensive coverage of all relevant data. These real-time tables allow for the efficient and comprehensive analysis of various customer interactions and behaviors, contributing to the development of accurate and reliable forecasting models. The real-time tables utilized for data collection include:

- Order Table: Orders made by each customer, along with associated discount amounts and shipment payments.
- Add to Cart Table: Add to basket events for each customer.
- Add to List Table: Records of the products customers add to their lists, offering insights into product preferences and interests.
- Product View Table: Products viewed by customers, allowing for the monitoring of customer engagement with specific items.
- Comment Table: Customer comments, enabling the analysis of customer feedback.
- Banner View Table: Instances of customers viewing banners, providing information on customer exposure to marketing materials.
- Banner Click Table: Customer interactions with banners, offering insights into the effectiveness of marketing efforts.
- Screen Load Table: Mobile application page load data that collects session-specific information each time a customer opens the mobile application.
- Page Load Table: Website page load data that collects session-specific information each time a customer opens the website.

Screen load and page load tables contain the unique session identifier, session start time, and user id. The datapoints for the models are derived from these tables. Sessions occurring between 2021-10-01 and 2021-10-07 are collected. The resulting dataset has 38,597,345 sessions in total. These sessions that represent the datapoints are referred as 'current sessions', whereas the sessions consitituting the target are called as 'next sessions'. Because the sessions are organized using a sliding-window approach, they can serve multiple purposes within the dataset. In other words, a session can act as a datapoint for which to predict the next session while serving as the target session or previous session for another datapoint. The sliding-window technique basically is moving a window across the data, extracting a subset of the data inside that window, and using the data inside that window for prediction. Subsets of the data that overlap as the window moves across the data are formed, capturing the temporal links between sessions. The model is exposed to different combinations of sessions and their accompanying results, resulting in more data, which helps it to learn the dynamics of customer behavior better.

The unique session identifier and user id are also present in other event tables, providing an opportunity to extract features. By joining these tables using the common session identifier column, features are created, encompassing a range of customer behaviors and interactions.

2.2. Target Variable Creation

The model targets to predict if a particular customer will have a next session and, if so, whether they will make a purchase during that session. Purchase is dependent in making a session: If there is a session there can be a purchase or not. If there is no session, purchase cannot happen. The data is highly imbalanced, which is showed in Table 2.1.

Table 2.1. Target ratio.

Target	Description	Count	Ratio
00	There is no next session	3,101,002	0.080
10	There is a next session and no purchase	34,654,545	0.898
11	There is a next session and purchase	841,798	0.022

An illustration of the sliding-window approach, applied to sessions and their corresponding targets, is provided in the Table 2.2. This visual representation offers a depiction of the technique, showcasing how sessions and targets are organized and connected within the dataset.

Table 2.2. Illustration of the data structure.

User id	Session id	Next Session id	Target
X	s1	s2	10
X	s2	s3	10
X	s3	s4	11
X	s4	s5	10
X	s5	null	00

2.3. Feature Extraction

The machine learning models have been given two basic kinds of features: bulk features and sequence features. These attributes are extremely important in boosting the model's understanding of consumer behavior and improving the effectiveness of the predictive capabilities of the model.

- Bulk Features: Bulk features are constructed for each data point by making use of data from the previous year. The majority of these characteristics are made up of aggregated customer events, which record different elements of consumer behavior and interactions over a prolonged period of time. The addition of these

data helps the model to discern larger patterns and trends in the behaviors of the consumer, hence offering a helpful framework for generating predictions regarding future sessions and purchases.

- Sequence Features: In contrast, sequence features are formed by using data from the most recent month for each data point. This feature group focuses on collecting the traits and patterns that have been present over the course of the previous seven sessions, bringing attention to current client behavior and involvement. By including these short-term patterns, the model is able to better react to fast-changing client preferences and trends, which enables more accurate predictions to be made in contexts that are dynamically driven by e-commerce.

The machine learning models are expected to successfully assess and forecast consumer behavior when both bulk Features and sequence Features are combined. This allows the models to take into account both long-term trends and short-term changes in customer behavior. Because of this all-encompassing approach, a more accurate and thorough depiction of client behavior is expected, which, in the end, leads to stronger predictions regarding the occurrence of future sessions and the likelihood of sales within those sessions.

The bulk features are further separated into four subcategories, each of which offers a different viewpoint on the traits and behaviors of customers. These subgroups help the machine learning models comprehend different facets of client interactions more thoroughly, ultimately enhancing their prediction ability.

(i) Monetary Features: The subgroup includes all metrics that relate to money, including total spending, average purchase value, shipment payments, discount amounts, total and average prices of items added to the cart and items added to lists. The group also accounts for the monetary implications of refunds, cancellations, and other important financial issues. These attributes offer a thorough understanding of the consumers' spending patterns, enabling the model to spot trends in purchasing behavior and other behaviors motivated by money.

(ii) Event Features: A wide range of consumer interactions are included in event features, including adding products to a basket, finishing transactions, taking advantage of discounts, canceling purchases, returning goods, and making comments. The models can find possible correlations between certain events and buying outcomes by averaging these events across time and detecting trends in consumer interaction.

(iii) Recency Features: The subgroup recency focuses on the amount of time that has passed since different client activities, such as the most recent purchase, product view, or cart addition. These parameters provide the models with the ability to evaluate the amount of recent activity of the consumers, which may be a good predictor of their likelihood of engagement and future purchases.

(iv) Demographic Features: Customer factors including age, gender, location, and other pertinent personal data are referred to as demographic features. With the use of these variables, the models may identify different customer categories and find possible connections between demographic traits and consumer behavior, which can be used to better target marketing campaigns and anticipate future consumer behavior.

On the other hand, sequence features consist of each customer's past seven sessions' characteristics, which give the machine learning models important data on recent consumer involvement and behavior patterns. The dataset has seven columns, each of which corresponds to one of the previous seven sessions, for each characteristic associated with a session. The preservation of the events' temporal sequence and the detection of potential patterns or trends over these sessions are both made possible by this structure.

For instance, each session's expenditure is denoted by seven distinct columns, indicating each customer's spending throughout the previous seven sessions. Another example is the number of unique products viewed by the consumer in each of their last seven sessions captured by the count of distinct products viewed, which is similarly divided into seven columns. These session-specific attributes provide a more detailed

perspective of the customer's most recent interactions and preferences, which can be crucial in comprehending and forecasting the future behavior of customers. Table 2.3 provides an overview of the feature groups, offering a general representation of some attributes considered within each category.

Table 2.3. Feature groups and domains.

Group 1	**Group 2**	**Feature Domains**	**Count**
Bulk Features	Monetary Features	Spending amount, discount amount, prices of liked products, refund amount, prices of the products added to cart etc.	70
Bulk Features	Event Features	Purchase count, discount usage count, add to cart count, add to list count, product view count, banner click count, banner view count etc.	79
Bulk Features	Recency Features	Days since; the first purchase, last purchase, add to cart, add to list, banner view, banner click, product view etc.	7
Bulk Features	Demographic Features	Gender, operating system, city, region, platform etc.	8
Sequence Features	Session Features	Spending amount, add to cart count, product view count, purchase category, view category etc.	189

2.3.1. Extraction of Bulk Features

(i) Monetary Features: These features consist of aggregations (sum, average, count) and one, six and twelve month versions of financially-related data. A comprehensive list of all the features created, along with detailed explanations, can be found in Appendix A. To account for inflation, two distinct methods are employed:

- Adjustment using Consumer Price Index: Historical monetary values are adjusted using the monthly consumer price index provided by Türkiye İstatistik Kurumu (TÜİK). This approach ensures that the monetary features accurately reflect changes in purchasing power over time.
- Representation as Quartiles: Instead of using currency values, features are expressed as quartiles. For each customer, their daily total payment amount is calculated on the days they made purchase within the last one month, six months, and year. Subsequently, the first, second, third, and fourth quartiles are determined for each day. If a person's daily expenditure falls within the first quartile, their first spend quartile count is incremented. Similarly, if a person's daily spending is below the interquartile range, their spend low outlier count is increased.

These methods of handling inflation and representing monetary features as quartiles provide a more robust and meaningful analysis of customer spending habits, ensuring that the machine learning models can make accurate predictions despite potential fluctuations in economic conditions.

(ii) Event Features: Consist of aggregations (sum, average, count) and one, six and twelve month versions of event related data. The number of add to list events in the last 1 month, 6 months and 12 months are three examples to this group. Appendix B has a complete list of all the event-related features developed, along with thorough explanations.

(iii) Recency Features: This group comprises features representing the number of days elapsed since a particular event occurred. They are created by subtracting the dates of past events from the current session date. The days passed since the last product view, claim request, and banner click are three examples of the features

included in this group. A comprehensive list of all the recency-related features created is provided in Appendix C, along with in-depth explanations.

(iv) Demographics Features: Consist of features representing the social characteristics of the customers. Gender, age and most used operating system are three examples to this group. Since these features are categorical, binary variables are created for each category of a variable. Appendix D has a thorough list of all the produced demographics-related features along with detailed explanations.

2.3.2. Extraction of Sequence Features

The datapoints consist of individual sessions, referred as the current sessions. To create sequence features, current sessions and their six previous sessions are used. For each session, 27 unique features are created. These features are created for the model to capture the recent patterns and trends in the data. By including the features of both the current session and the preceding sessions, the model is better equipped with the recent activity sequence of customers. Appendix E has a detailed explanation of these features. Figure 2.1 illustrates a visual representation of the session design. t=0 refers to the moment the model is executed, which coincides with the end of the current session. There are four example datapoints in the figure, each with a varying number of previous sessions. For instance, while session X has six previous sessions, session Y has two, session Z has four, and session W does not have any.

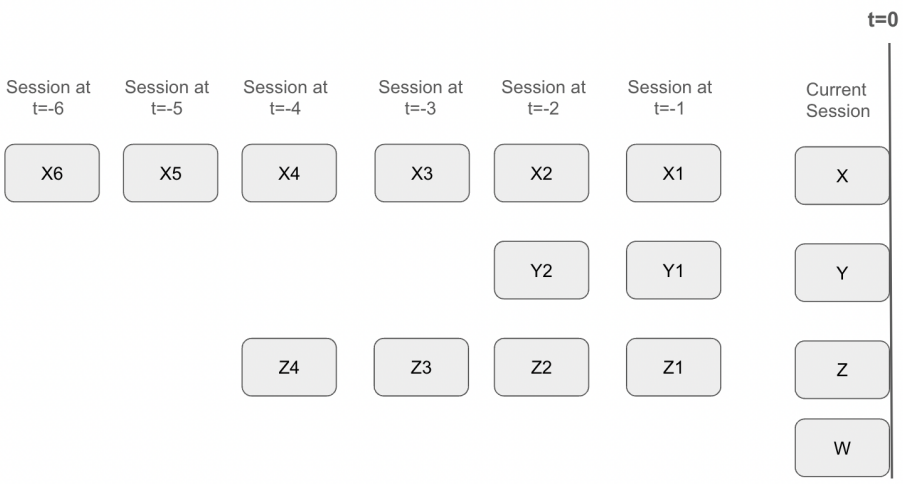

Figure 2.1. Illustration of current sessions and their preceding.

2.3.3. Different Data Structure for Different Models

The prediction task is implemented using both the LightGBM and LSTM models. These models need input data with different formats. Since LightGBM focuses on learning patterns from individual features and their interactions, data is arranged in a tabular format. On the other hand, since the LSTM model is designed to handle time series data to capture the dependencies in a sequence, only the sequential features are fed into the model.

The model is expected to learn and simulate the dynamic relationships between the current session, its previous sessions, and the targeted next session using this method. For this purpose, while the LightGBM model is supplied a pandas dataframe containing all the extracted features, the LSTM model is fed a 3-dimensional numpy array in the shape of (observation count, timestep count, feature count).

2.4. Dimensionality Reduction

Dimensionality reduction is a technique to reduce the number of features in a dataset to eliminate unnecessary features to enable models to be trained with relevant data and lower computational cost. There are various strategies to reduce the dimension of the data. However, the possibility of loosing valuable information exists, which creates a trade-off between cost and better model performance. Since the gradient boosted machines can capture the importance of the features and discover the little or no effect of irrelevant features on the prediction, they are also used for feature reduction. Considering the power of the machine used, computational cost is not a concern, which is why, only the Principal Component Analysis is used to lower the number of features.

PCA is a linear transformation technique that identifies the directions, called principal components, in which the variation in the data is maximized. These principal components are orthogonal. By projecting the original data onto these principal

components, PCA creates a new set of features which are the linear combinations of the original features while maintaining the majority of the data's variance.

For the implementation of PCA, sklearn's decomposition library's PCA function is used. The function utilizes full or randomized truncated Singular Value Decomposition method, depending on the size of the input data [37]. The full SVD is a matrix factorization method that decomposes a given matrix into three matrices: an orthogonal matrix representing the left singular vectors, a diagonal matrix containing the singular values, and another orthogonal matrix representing the right singular vectors. In the context of PCA, the full SVD method helps to determine the principal components and transform the input data into a lower-dimensional space. Since computing the full SVD can be computationally expensive for large datasets, the randomized truncated SVD is proposed [38]. The method approximates the top-k singular values and vectors of a matrix by using a randomized algorithm, which can significantly reduce the computational cost, especially when the number of components to extract is much smaller than the input data's dimension.

Different PCA models are applied to the one month, six months and one year versions of features. For this purpose, time aggregated the same 1m, 6m and 12m features are grouped, and for each group, PCA is performed after standardization. The principal components resulting from the PCA computations are sorted according to their contributions to the explained variance. The components providing the total variance of 70% or more are included in the feature space whereas the other components are not used. For instance, as a result of the PCA implemented for product view count in the last 1 month, 6 months and 1 year, the resulting number of features is 2, preserving the 73.6% of the variance. Finally, 43 PCAs are performed in total. The number of features decreased from 353 to 295.

2.5. Handling Outliers

Outlier handling is an essential step in the data pre-processing phase of machine learning. Outliers are data points that significantly deviate from the majority of the data, potentially affecting the performance of machine learning models. Outlier detection is especially critical in the e-commerce domain. Very active or high-value customers engage with the platform regularly, and contribute significantly to the company's revenue. These customers may exhibit behaviors that deviate from the norm, such as higher spending, more frequent visits, or more significant engagement with the platform compared to the average customer. These behaviors, while valuable, can be considered as outliers in the data. However, treating these customers as anomalies and excluding them from analysis or marketing strategies could be detrimental to the business. Therefore, it is extremely important to carefully handle outliers in e-commerce. Rather than blindly eliminating outliers, it is essential to understand the context and reasons behind these deviations.

There are different customer groups in the online retail sector, and the marketing campaigns are differently structured considering the needs of the groups. Customers who own a firm and purchase high-priced technological tools, fashion enthusiasts who shops premium cosmetics, new customers who have not bought anything, one-timers who have only one purchase are some examples to diverse range of customer groups. Any of these customers are not considered as outliers. The accounts who have unreasonable amount of product view count and no transaction are considered outliers since they are possible scrappers developed by competitors to get product information from the website. Additionally, the accounts who have critically deviated amounts of spending can be frauds who make lots of purchase to sell the goods later with higher price. Outlier detection is implemented to catch these two kinds of members and the input data corresponding to these customers are removed.

There are several strategies in the literature to identify the outliers. In a study, the usage of a range calculated by subtracting and adding 2 or 3 standard deviations to

the mean is suggested [39]. In this approach, the datapoints fall outside of the range are considered outliers, under the assumption of normal distribution. In another inquiry, approach called Tukey's Fences is proposed, which is widely taught in machine learning classes [40]. With this method, outliers are identified based on the interquartile range. Data points below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR are considered outliers. This method is more robust to non-normal distributions compared to the standard deviation method.

Tukey's Fences method is applied to determine the outlier customers by leveraging their spending amount, transaction and product view counts since these features are considered as the good identifiers of customer characteristics in the e-commerce domain. However this method identified the most premium customers and the ones with little to no purchase as outliers. To combat this problem, percentile range approach is applied. The customers whose spending amount, transaction and product view counts are above the 99.5 percentile are considered as outliers to detect frauds and scrapers. As a result, data of 21 members is removed, reducing the whole data from 38,597,345 to 38,589,864.

2.6. Undersampling

In e-commerce, it is common for models to encounter imbalanced data. In this particular case, the model has three classes, and the distribution of the classes is highly skewed. While almost 90% of the sessions in the input data end up with a next session and no purchase, 8% do not have a next session and only 8% have a purchase in the next session. To address this issue, undersampling was employed to balance the dataset, and LightGBM and LSTM models are applied to both the whole data and undersampled version. The performance of different models with the two versions of the data are compared afterwards.

Undersampling was performed by retaining all instances belonging to the least represented class, which is encoded as '11', and selecting a proportionate number of instances from the other classes, according to their target ratios. This ensures that

the distribution of the classes in the modified dataset maintains a similar ratio to the original data. Additionally, to preserve the timely patterns in the data, the session day and hour distribution are took into account for undersampling process. That is, the target ratio in each hour for each day is preserved. By doing so, the modified dataset is expected to retain a highly representative sample of customer sessions. This approach is applied for the data to maintain its real nature, while reducing the high proportion of the available data, allowing the model to capture meaningful insights and generate accurate predictions even with low number of datapoints. As a result, the input data size is decreased from 38,597,345 to 2,525,216.

2.7. Train Test Split

In machine learning, it is necessary to split the data into training and test sets to evaluate the performance of the developed model. This process have several objectives. First of all, it mitigates the risk of overfitting, that is, it prevents for the model to become too specialized to the training data and fails to perform adequately on unseen data [41]. Secondly, the test set allows for an unbiased evaluation of the model's performance. This is crucial for choosing the best model from several alternatives or for tuning the model's hyperparameters [42].

To perform the split, sklearn's model_selection module's train_test_split function is used. 80% of the input data is included in the training set whereas 20% of the data is included in the test set. By using the optional stratify parameter of the model, the target ratio is preserved in the test set.

2.8. Hyperparameter Optimization

Model performance is of vital importance in the field of machine learning. Hyperparameter tuning is a crucial factor that affects performance. A model's behavior is determined by hyperparameters such as learning rate, regularization factor, and number of iterations. These parameters are not taught during training but must be

established beforehand. Tuning hyperparameters is critical for attaining optimal model performance.

Bayesian optimization is a popular method for hyperparameter tuning, especially when dealing with costly or time-intensive objective functions. This optimization technique is founded on Bayesian modeling, which approximates the objective function using probabilistic models. Bayesian optimization relies on prior knowledge and updates its beliefs as new data points are added. This process of updating helps the approach to make better selections when looking for ideal hyperparameters, resulting in a more effective search space exploration.

When it comes to hyperparameter tuning, practitioners frequently employ grid search and randomized search as alternative optimization techniques. Grid search is a brute-force technique that thoroughly investigates all potential hyperparameter value combinations inside a predetermined search space. Despite the fact that this method assures the optimal configuration, it might be computationally intensive, particularly for high-dimensional spaces. Randomized search, on the other hand, selects a random selection of hyperparameter configurations to assess, allowing for a more effective exploration of the search space. This strategy does not ensure the ideal setting.

Bayesian optimization differs from grid search and randomized search in that it explores the hyperparameter search space with a more strategic approach. It approximates the objective function with a surrogate model. This surrogate model, typically a Gaussian Process, assesses the performance of various hyperparameter values based on prior information and available data points. By exploiting this knowledge, Bayesian optimization strikes a balance between exploration and exploitation, enabling a more efficient search and potentially accelerating convergence to the ideal design [43].

Bayesian optimization procedure operates iteratively. Initially, a surrogate model is developed, and an acquisition function is selected to direct the search for the subsequent set of hyperparameter values to assess. The acquisition function determines

the equilibrium between investigating new sections of the search space and exploiting promising regions where the proxy model predicts superior performance. Once the acquisition function has been tuned, the subsequent set of hyperparameters are evaluated through model training and validation. The derived performance metric is then utilized to update the proxy model, and the procedure is repeated until a termination requirement is fulfilled [44].

To optimize the hyperparameters in the models, Bayesian optimization is used by utilizing the Hyperopt library [45]. The library employs the Tree-structured Parzen Estimator algorithm which is a sequential model-based optimization method. TPE algorithm of the Hyperopt library is given the objective function, which is the model training, parameter space that includes the distribution of parameters to tune, and the number of iterations. An example distribution for one of the hyperparameters that is tuned in LightGBM can be seen in Figure 2.2. The loss calculated after each training with different hyperparameter combinations is stored in a comma-separated file. The hyperparameters resulting in the lowest loss are selected as the final values.

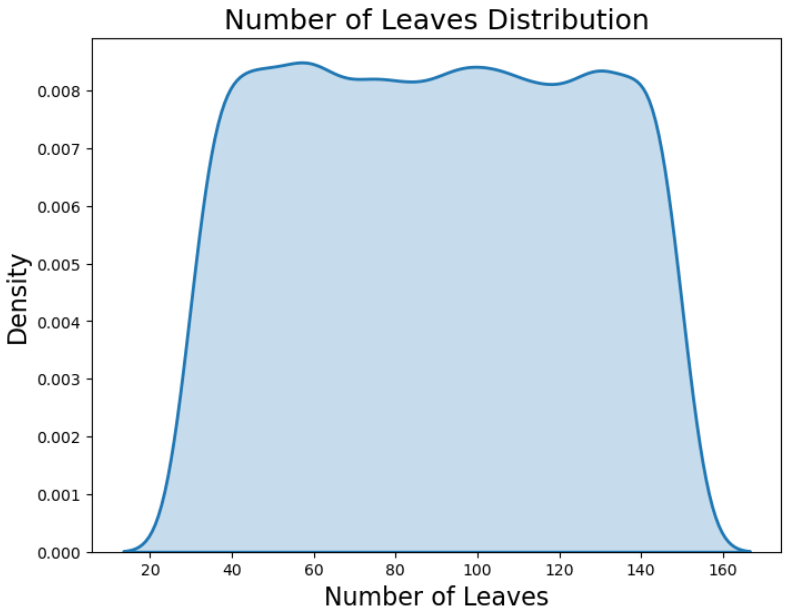

Figure 2.2. Number of leaves hyperparameter space.

2.9. Cross Validation

Cross-validation is a technique to help gauge the performance of models, as well as their ability to generalize to new, unseen data [42]. By employing this technique, a more comprehensive understanding of a model's capacity to adapt to novel data is achieved, thereby minimizing the risk of overfitting. The overarching aim of cross-validation is to alleviate biases that may arise from the specific distribution of data, resulting in a more robust assessment of the model's ability to accommodate new data [46, 47].

There are various cross-validation techniques. Leave-one-out cross-validation involves training the model on all the data points except one and testing its performance on the excluded instance. This process is repeated for every data point, with the performance metrics subsequently averaged. Since leave-one-out cross-validation imposes a high computational burden, particularly for larger datasets, k-fold cross-validation is utilized. In this approach, the dataset is partitioned into k distinct folds, and the model undergoes training and validation on different combinations of these subsets [42]. Then, the performance metrics are aggregated.

Since the computational cost rises as the number of folds increases, 3-fold cross-validation is implemented on the training data for the LightGBM model. The folds are created in a stratified manner to ensure the target ratio stays the same in different folds.

2.10. Performance Evaluation Metrics

Error metrics and performance metrics are two ways to evaluate machine learning models. Error metrics quantify the gap between predictions and actual outcomes, with lower values indicating better performance (e.g., mean squared error or log loss). In contrast, performance metrics focus on the real-world impact of the model and how effectively it meets specific objectives (e.g., accuracy, precision, recall, or F1-score). The key difference is that error metrics assess discrepancies between predictions and

true values, while performance metrics consider the model's effectiveness in addressing a problem or goal. Error metrics guide model optimization during training, while performance metrics are used for model comparison and evaluation.

2.10.1. Error Metrics

Error metrics in machine learning are essential for assessing the performance of a model during training and validating its generalization to unseen data. These metrics provide quantitative feedback on the model's ability to make accurate predictions, enabling making informed decisions about model selection, hyperparameter tuning, and the overall effectiveness of the chosen algorithms. By monitoring the error metrics throughout the training process, potential overfitting or underfitting can be identified and corrective actions can be taken to combat these issues, such as early stopping or adjusting the learning rate. There are various boosting types the LightGBM library supports. Gradient Boosted Decision Trees and Dropouts meet Multiple Additive Regression Trees are utilized during the hyperparameter tuning. While the GBDT boosting type follows the standard gradient boosting procedure where a new tree is added at each iteration to minimize the loss function, the DART boosting type introduces an additional regularization step by randomly dropping a subset of trees from the model before updating the predictions and adding a new tree.

The prediction update process as a result of the error metric calculation is slightly different between GBDT and DART. GBDT model computes the gradients of the loss function with respect to the current predictions at each iteration. Then it creates a new tree to fit the computed gradients. After the model updates its predictions by adding the output of the new tree multiplied by a learning rate, the error metric is calculated and the process is repeated for a specified number of iterations or until a stopping criterion is met.

On the other hand, DART model randomly selects a subset of trees to drop from the current model at each iteration. Then, similar to GBDT, the model computes

the gradients of the loss function with respect to the updated predictions and adds a new tree to fit the computed gradients. After the dropped trees are added back to the model and the model updates its predictions by adding the output of the new tree multiplied by a learning rate, the error metric is calculated and the process is repeated for a specified number of iterations or until a stopping criterion is met like in the GBDT model [48].

For LSTM, the loss function is calculated after each forward pass of the data through the training. In each forward pass, the model takes input data and generates predictions based on its current weights and biases. The loss function is then computed using the model's predictions and the actual target values. To reduce the loss, weights and biases need to be updated. This is done through backpropagation, which is performed after the loss is calculated. It is a method of computing gradients of the loss function with respect to each parameter in the model. Then an optimization algorithm is used to update the model's parameters in a direction that minimizes the loss function [35].

Log loss is a widely used loss function for classification tasks. It measures the performance of a classification model by quantifying the difference between the predicted probabilities and the true labels. Log loss is a continuous and differentiable function, making it suitable for gradient-based optimization algorithms. The log loss is expressed as

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log(p) + (1 - y_i) \cdot \log(1 - p)\right). \tag{2.1}$$

In this expression, N is the number of samples, y is the true label (either 0 or 1), and p is the predicted probability of the class with label 1.

Multi-class log loss metric is used, which is a version of log loss extended to multiple classes. The multi-class log loss is expressed as

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij}). \tag{2.2}$$

In this expression, N is the number of samples, y_{ij} is the true label for sample i and

class j, and p_{ij} is the predicted probability of sample i belonging to class j. The inner summation runs over all classes, while the outer summation runs over all samples. A lower log loss value indicates a better model performance. Log loss penalizes models with incorrect and overconfident predictions, making it a useful metric for probabilistic classification tasks [49].

2.10.2. Performance Metrics

Performance metrics are used to evaluate the effectiveness of a machine learning model in meeting specific objectives, particularly in classification problems. These metrics help to understand the model's ability to make correct predictions and the balance between false positives and false negatives.

- Accuracy: Out of all the samples predicted, shows how many have been predicted correctly. Accuracy is expressed as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{2.3}$$

  Accuracy is useful when the classes are balanced and the cost of misclassification is equal for all classes.

- Precision: Out of all the samples predicted as positive, shows how many have been predicted correctly. Precision is calculated as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.4}$$

  Precision is important when the cost of false positives is high, such as in spam detection, where mislabeling non-spam emails as spam can lead to user dissatisfaction.

- Recall: Out of all the actual positives, shows how many have been predicted correctly. Recall is expressed as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.5}$$

  Recall is crucial when the cost of false negatives is high, such as in medical diagnosis, where failing to identify a disease can have severe consequences.

- F1 Score: Harmonic mean of the precision and recall. It is calculated as

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.6}$$

  F1-score is useful when both false positives and false negatives are important, or when there is a significant class imbalance, as it balances the trade-off between precision and recall.

Precision, recall and F1 score are used calculated as the performance metrics. Since the problem is multi-class, they are calculated by assigning the class with the highest probability as the predicted label [50].

2.11. Modelling

2.11.1. LightGBM

The LightGBM library is one of the implementations utilized for the prediction task. It is a powerful tool available in Python, offers an efficient implementation of gradient boosted decision trees. This library is especially esteemed for its ability to handle large-scale datasets and high-dimensional features with ease. Its unique approach to tree growth, which favors leaf-wise expansion over the conventional depth-wise strategy, contributes to its notable performance. In addition to supporting a variety of loss functions, LightGBM provides advanced features like categorical feature handling and GPU acceleration [33, 34].

LightGBM adeptly manages categorical variables by employing a technique called Optimal Split for Categorical Features. This method identifies the optimal split point for categorical features by considering the target variable's relationship with the categories. The OSCF algorithm's core concept is to find the best split by calculating the optimal partitioning of category combinations. It evaluates the gain obtained from each potential split and chooses the one that maximizes the gain. By directly handling categorical variables in this way, LightGBM avoids the need for additional preprocessing steps and reduces the risk of information loss, ultimately leading to faster training and

better model performance. Instead of one-hot encoding, which is a common preprocessing step for other algorithms, LightGBM is utilized to feed the created categorical features to the model. For this purpose, categorical features are converted to the type 'category' and fed into the model like the other variables.

LightGBM also effectively copes with missing values, eliminating the need for manual data imputation. As the decision tree is being built, the algorithm determines the most suitable direction for data points with absent values. For features with missing entries, either categorical or numerical, LightGBM assesses possible split points, calculating the gain when directing the missing value to the left or right child node. The direction yielding the highest gain becomes the preferred choice for the missing value. This intrinsic handling of missing values in LightGBM has the benefit of reducing potential noise or bias that could arise from manual imputation methods, hence null values are not manually imputed but left to the model to be handled.

There are various hyperparameters to tune that are available in the LightGBM library. The used hyperparameters and their explanation are as follows:

- objective: The objective function determines the type of problem that LightGBM aims to solve. 'multiclass' is used.
- is_unbalance: This is a boolean parameter that indicates if the dataset is imbalanced. When set to True, LightGBM will automatically adjust the weights of the classes to handle the imbalance. By assigning larger weights to underrepresented classes, the model effectively increases their importance during training. As a result, the model strives to minimize errors associated with minority classes, improving its overall performance on imbalanced datasets. This parameter is set to True.
- num_class: Specifies the number of classes in a multi-class classification problem. This is set to the number of classes in the target variable, which is 3.
- boosting_type: Determines the boosting algorithm to use. GBDT, DART, and GOSS are tried during hyperparameter tuning.

- num_leaves: Controls the maximum number of leaves in a tree. Larger values result in a more complex model but may lead to overfitting. The bayesian optimization algorithm is given a quantized continuous distribution between 30 and 150.
- learning_rate: The learning rate influences how quickly the model learns from the data. Smaller values require more iterations to achieve similar performance but can result in a more accurate model. The bayesian optimization algorithm is given values between 0.0001 and 0.2 by utilizing loguniform function of hyperopt.
- subsample_for_bin: Specifies the number of samples to use when constructing the histogram bins. A larger value may lead to more accurate splits, but it may also increase the training time. The bayesian optimization algorithm is given a quantized continuous distribution between 20000 and 300000.
- min_child_samples: Determines the minimum number of samples required in a leaf node. Larger values can help prevent overfitting but may result in a less complex model. The bayesian optimization algorithm is given a quantized continuous distribution between 20 and 500.
- reg_lambda: The L2 regularization term on the weights, also known as the ridge penalty. Increasing this value can help reduce overfitting by shrinking the coefficients toward zero. The bayesian optimization algorithm is given uniform distribution between 0 and 1.
- colsample_bytree: Specifies the fraction of features to select for each tree. Setting this value to less than 1 can help prevent overfitting and improve generalization by introducing randomness into the model.The bayesian optimization algorithm is given uniform distribution between 0.6 and 1.

LightGBM multiclass classification model gives class probabilities for each data point. To create easy-to-use predictions, the class with the highest probability is selected as the predicted class. Then, the performance metrics are calculated.

2.11.2. LSTM

LSTM, which is a widely preferred machine learning algorithm for sequential forecasting, is also used for the prediction task. Since the model combats the vanishing gradient problem of the traditional RNN, it captures the long-term dependencies by employing a unique gating mechanism. As a result, LSTMs can successfully learn from long sequences without losing essential information. TensorFlow's Keras API is used for the implementation of LSTM. It is a user-friendly, high-level library designed to facilitate the development of deep learning models. One of the key features of Keras is its modular approach to constructing neural networks through the concept of layers. Layers represent the fundamental building blocks of deep learning models, enabling the definition of various types of neural network architectures by simply stacking these layers sequentially [51].

When dealing with sequence data, padding is a technique used to ensure that all input sequences have the same length. This uniformity is crucial since neural networks require fixed-size input tensors. Padding is performed by adding a specific value to the beginning or the end of the sequences to achieve the desired length. Keras provides a masking layer to handle padded and missing values in sequence data, allowing the model to ignore certain time steps during processing. The Masking layer works by specifying a mask value, which represents the missing or padded values in the input sequence. When the model encounters this mask value, it skips the corresponding time step, ignoring it during both the forward and backward pass of training. -1 is added to the beginning of the sequences for the data points not having 7 sessions. Additionally, null values are also imputed with the same value, and the masking value of -1 is given to the masking layer.

There are a few hyperparameters to tune that are available in the keras library. The used hyperparameters and their explanation are as follows:

- batch_size: The number of samples used to update the model's weights during

each iteration of the training process. The Bayesian optimization is given a uniform distribution between 64 and 2048.

- class_weight: Used to assign different weights to each class during the computation of the loss function. This ensures that the model pays more attention to the minority class. Class weight is calculated by dividing 1 by the corresponding class size.
- Number of Hidden Units: The number of LSTM neurons in each LSTM layer. The Bayesian optimization is given a uniform distribution between 64 and 256.
- Number of Layers: The number of LSTM layers stacked together to form the model. Since adding layers results in a high computational cost, only 1 LSTM layer is used.
- Dropout: Dropout is a regularization technique that helps prevent overfitting by randomly dropping out a fraction of the input units during training. There are two parameters for this task. The dropout parameter controls the dropout rate applied to the input units, while recurrent_dropout controls the dropout rate applied to the recurrent units. The Bayesian optimization algorithm is given a uniform distribution between 0 and 1 for both parameters.
- Optimizer: An optimizer is an algorithm used to update the weights of the LSTM model during training. The optimizer's primary goal is to minimize the loss function. Adadelta and RMSprop are applied to the model.

Softmax is used as the activation function in the output layer to transform the prediction into probability for each class. Similarly to LightGBM, the class with the highest probability is selected as the predicted class also for the LSTM model. Performance metrics are calculated with the resulting class assignments.

2.11.3. Ensemble

LightGBM and LSTM models have different underlying architectures, which allows them to capture different patterns in the data. By combining their results, a performance increase is expected by benefitting from the learning capabilities of both

models. There are various possible ensembling techniques that can be used for the specific prediction task. Due to the increased complexity and computational cost resulting from the combination of multiple models, a simple technique is employed to make more reliable predictions.

For this purpose, addition is performed on the probability matrices of the LightGBM and LSTM models, outputting a new matrix where each entry represents the sum of the predicted probabilities for each class. The final predicted class for each sample is the one with the highest summed probability. By utilizing this technique, errors made by one model may be averaged out by the other model, which can result in an increase in overall performance. Additionally, if one model performs poorly in a specific region of the input space, the other model in the ensemble might compensate for this weakness, leading to more accurate predictions. Furthermore, summing the probability matrices can provide an estimation of the ensemble's confidence in its predictions. Higher summed probabilities for a particular class indicate that multiple models agree on the prediction, which can be useful for decision-making.

3. RESULTS

3.1. Multi-Log Loss

3.1.1. LightGBM Results

Table 3.1 shows the resulting LightGBM multi-logarithmic loss for both the test and train data, dimensioned by the sampling implementation. The model has less loss for the sampled data. Since train and test metrics are very close to each other, it can be said that there is no overfitting to the train data.

Table 3.1. LightGBM multi log loss.

Dataset	**Non-Sampled LGB**	**Sampled LGB**
Test	0.88	0.76
Train	0.87	0.76

3.1.2. LSTM Results

Table 3.2 shows the resulting LSTM multi-logarithmic loss for both the test and train data, dimensioned by the sampling implementation. The model has less loss for the sampled data. Since train and test metrics are very close to each other, it can be said that there is no overfitting to the train data.

Table 3.2. LSTM multi log loss.

Dataset	**Non-Sampled LSTM**	**Sampled LSTM**
Test	0.91	0.8
Train	0.9	0.79

3.2. Performance Metrics

3.2.1. LightGBM Results

Tables 3.3, 3.4, and 3.5 depict the resulting precision, recall, and F1 score of the LightGBM model implementation on sampled and non-sampled data, respectively.

Table 3.3 shows that the model has higher precision for Class 10 compared to other classes in the case of both non-sampled and undersampled data. On the other hand, Class 11, which has the least relative number of data points, is predicted with the lowest precision for the two datasets. However, with the application of undersampling, not only does the precision belonging to each class rise, but also the performance difference between classes gets considerably low.

Table 3.3. LightGBM precision.

Target Label	**Non-Sampled LGB**	**Sampled LGB**
Class 00	0.64	0.69
Class 10	0.68	0.7
Class 11	0.54	0.67

Table 3.4 shows that the model has a higher recall for Class 10 compared to other classes in the case of non-sampled data. On the other hand, with undersampling, the recall is the highest for Class 00. Additionally, while Class 11 is predicted with the lowest recall for the non-sampled data, its prediction results in a higher recall with undersampling. It can be seen that sampling has dramatically changed the recall performance of the model for all classes.

Table 3.4. LightGBM recall.

Target Label	**Non-Sampled LGB**	**Sampled LGB**
Class 00	0.69	0.8
Class 10	0.78	0.69
Class 11	0.61	0.76

Table 3.5 shows the calculated F1 scores from the resulting recall and precision values. Although the F1 score is higher for Class 10 compared to other classes in the case of non-sampled data, it decreases with undersampling and becomes the lowest score. On the other hand Class 11's score increases with undersampling. It can be seen that F1 score belonging to each class for the undersampled data is close to each other, which makes the result of sampled data better than that of non-sampled data overall.

Table 3.5. LightGBM F1.

Target Label	**Non-Sampled LGB**	**Sampled LGB**
Class 00	0.66	0.74
Class 10	0.73	0.69
Class 11	0.57	0.71

3.2.2. LSTM Results

Tables 3.6, 3.7, and 3.8 depict the resulting precision, recall, and F1 score of the LSTM model implementation on sampled and non-sampled data, respectively.

Table 3.6 shows that the model has higher precision for Class 10 compared to other classes in the case of both non-sampled and undersampled data. On the other hand, Class 11 is predicted with the lowest precision for the non-sampled data, and Class 00 hast eh lowest score for the undersampled data. However, with the application of undersampling, the precision of Class 11 increases, while the precision for Class 10 decreases, and that of Class 00 remains the same.

Table 3.6. LSTM precision.

Target Label	**Non-Sampled LSTM**	**Sampled LSTM**
Class 00	0.59	0.59
Class 10	0.69	0.63
Class 11	0.51	0.60

Table 3.7 shows that the model has a higher recall for Class 10 compared to other classes in the case of non-sampled data. On the other hand, with undersampling, the recall is the highest for Class 00. Additionally, while Class 11 is predicted with the lowest recall for the non-sampled data, its prediction results in a higher recall with undersampling.

Table 3.7. LSTM recall.

Target Label	**Non-Sampled LSTM**	**Sampled LSTM**
Class 00	0.60	0.76
Class 10	0.72	0.59
Class 11	0.57	0.71

Table 3.8 shows the calculated F1 scores for the LSTM model from the resulting recall and precision values. Although the F1 score is higher for Class 10 compared to other classes in the case of non-sampled data, it decreases with undersampling and becomes the lowest score. On the other hand Class 11's score increases with undersampling. It can be seen that the F1 score belonging to each class for the undersampled data is close to each other, which makes the result of sampled data better than that of non-sampled data overall.

Table 3.8. LSTM F1.

Target Label	**Non-Sampled LSTM**	**Sampled LSTM**
Class 00	0.59	0.66
Class 10	0.70	0.61
Class 11	0.54	0.65

3.2.3. Ensemble Model Results

Tables 3.9, 3.10, and 3.11 depict the resulting precision, recall, and F1 score of the ensemble model on sampled and non-sampled data, respectively. Table 3.9 shows that the ensembled model has higher precision for Class 10 compared to other classes in the case of both non-sampled and undersampled data. On the other hand, Class 11 is predicted with the lowest precision for both the non-sampled and undersampled data. The application of undersampling resulted in closer precision scores between classes.

Table 3.9. Ensemble model precision.

Target Label	**Non-Sampled Ensemble Model**	**Sampled Ensemble Model**
Class 00	0.66	0.71
Class 10	0.69	0.73
Class 11	0.54	0.69

Table 3.10 shows that the model has a higher recall for Class 10 compared to other classes in the case of non-sampled data. On the other hand, with undersampling, the recall is the highest for Class 00 and Class 11. Additionally, while Class 11 is predicted with the lowest recall for the non-sampled data, its prediction results in a higher recall with undersampling.

Table 3.10. Ensemble model recall.

Target Label	**Non-Sampled Ensemble Model**	**Sampled Ensemble Model**
Class 00	0.71	0.80
Class 10	0.73	0.72
Class 11	0.65	0.80

Table 3.11 shows the calculated F1 scores for the ensembled model from the resulting recall and precision values. Although the F1 score is higher for Class 10 compared to other classes in the case of non-sampled data, that of other classes gets higher and closer with the implementation of undersampling.

Table 3.11. Ensemble model F1.

Target Label	**Non-Sampled Ensemble Model**	**Sampled Ensemble Model**
Class 00	0.68	0.75
Class 10	0.71	0.72
Class 11	0.59	0.74

4. DISCUSSION

The preferred preprocessing techniques, sampling strategy, and modeling algorithms have all an effect on the performance of the classification task. When deciding on which technique to choose, the resulting performance metrics are of significant importance, but not enough to make an informed decision. To pick the best model, it is important to consider all the advantages and disadvantages of the steps employed during the model development process in addition to the consideration of the business requirements. Hence arises the need to contemplate a detailed evaluation of the different approaches.

4.1. Comparing the Model Performances for the Non-Sampled Data

Among the two modeling techniques implemented, the best multi-log loss for the non-sampled data is 0.88, which is the result of the LightGBM model, whereas the LSTM model results in a slightly lower loss of 0.91. On the other hand, with the usage of undersampling, the errors for both models have a significant fall while the LightGBM model preserves its superior performance over the LSTM.

On the other hand, for the evaluation of the resulting precision, recall, and F1 score, business needs are highly important. The practitioners need to decide which performance metric is the most suitable by taking the cost of the usage of each metric into account. Considering precision, when the LightGBM model predicts 100 data points as Class 00, the prediction is correct for 64 samples, when it predicts them as Class 10, 68 are correctly predicted and in the case of Class 11, correct predictions are made for 54 out of 100. Alternatively, the LSTM model's precision is 59% for Class 00, 59% for Class 10, and 51% for Class 11. Upon comparison, it can be said that the LightGBM model is more confident about its predictions overall, even though the LSTM model's precision score for Class 10 outperforms that of the LightGBM by 1% points. Additionally, the Ensemble model is 2% more confident for the Class 00

predictions than that of LightGBM and maintains the Class 10 performance of the LSTM and Class 11 performance of the LightGBM, which makes it the best choice considering only the precision.

The recall metrics of the LightGBM model are also better than those of the LSTM model. LightGBM is able to identify the 69% of the actual data points of Class 00, 78% of Class 10, and 61% of Class 11, significantly outperforming the LSTM model for all three targets. Although LSTM can recognize only 60% of the actual Class 00, it contributes to the LightGBM performance, increasing its recall by 2% as a result of the ensembling implementation. Similarly, the ensemble of the LSTM model results with the LightGBM model results increases the LightGBM recall by 4% to 65% for Class 11. In contrast, the ensemble recall for Class 10 is lower than that of LightGBM by 5%. Considering the recall metrics, the ensemble model seems to be able to identify the classes with fewer samples, which makes it the best possible choice as a result of comparing the models' overall ability to detect the true classes.

When both the model's capability of detecting the true classes and lowering the number of false detections are considered equally important, the LightGBM model gives much better results for all three classes. While the resulting F1-score is 66% for Class 00, it is 73% for Class 10 and 57% for Class 11. In comparison, the LSTM model leads to F1-scores of 59% for Class 00, 70% for Class 10, and 54% for Class 11. The ensembling technique increases the LightGBM F1-scores for the underrepresented classes, 2% for both Class 00 and Class 11 while decreasing that of Class 10 also by 2%. Hence, the ensemble model seems to be better at the prediction task when precision and recall are equally valuable.

4.2. Assessing The Effect of Undersampling on the Model Performances

The employment of undersampling boosts the performances of underrepresented classes in most cases. Although the performance of Class 10 worsens in some cases, it can be acceptable since undersampling makes the class performances approach each

other while maintaining high performance for all the classes. Considering this, the ensembled model built with the undersampled data gives the best performance in theory. However, the performance metrics measured may not be enough to make an accurate interpretation of the effect of the undersampling for several reasons.

First of all, millions of data are lost as a result of the sampling strategy. Although the timely target distribution is maintained, there can still be a case where critical information leading to overrepresented classes may be disappeared. Secondly, since the sample may not be a true representation of the actual population, the models may have a decreased performance in production. Additionally, undersampling comes with a risk of overfitting the training data. Since the size of the dataset is reduced significantly, the models may learn the training data very well but lose the complex patterns and therefore not generalize to new, unseen data.

To make sure there is a real boost in performance when the training is made with undersampling, more data can be collected if possible and the prediction results of the models can be compared using this validation data. If it is not possible, A/B testing can be performed in production by randomly selecting two groups and applying the non-sampled and undersampled models separately. After taking the same set of business actions according to the predictions for each group, the performance can be measured with a specific metric and the difference between groups can be statistically tested to see which one performs better and if the difference is significant.

4.3. Choosing the Best Model

The evaluation of the metrics shows that the ensembled model is the best-performing one. In addition to the need for further analysis to see the real effect of undersampling, model performance should also be analyzed in more depth to see if there is an actual additional business value resulting from the usage of the models. For this purpose, A/B testing should be performed on a determined sample of test users that represent the population. This sample should be divided into identical groups.

While random sampling can be chosen as the sampling technique, different methods can be used according to the business needs. Then different models are applied to each group, and planned business actions are taken according to the model results. Additionally, if an existing model is available, it is applied to one of the groups. If not, no model is applied to that group to see if the models contribute to the business metrics. The difference between the calculated metrics is statistically tested to see a significant increase.

Moreover, the performance of the models should not be the only consideration. There is always a trade-off between the high model performance and computational cost since using lots of features and employing complex modeling techniques may be costly, which is something to contemplate before making a decision. Also, the planned utilization of online prediction or batch prediction is of great importance since latency can be a problem for real-time prediction, depending on the computational power of the machine.

Furthermore, if the models result in undesirable performance, supplementary actions can be taken. In the case of LightGBM, hyperparameter optimization can be performed more extensively with the usage of a machine with a GPU and higher RAM. For LSTM, the number of sequence features may not be enough for the particularly complex prediction task, extracting new features may result in an increase in performance. Additionally, the LSTM model is developed with only one hidden layer, which also may not be enough to capture complex customer behavior. The model can be developed with more hidden layers to boost the performance while contemplating the usage of regularization methods to combat overfitting. Additionally, although the ensembling technique results in an increase in performance, it may not be considered significant. For a better performance boost, more complex ensembling techniques can be employed.

5. CONCLUSION

The aim of this thesis is to provide a combined approach to the two common prediction tasks in e-commerce; if a customer will engage in a session and if she will make a purchase. In the literature, these crucial tasks are treated differently, arising the need to develop and deploy two separate models for the practitioners. However, session engagement and purchase behavior cannot be considered independently since the main driving factor behind the act of logging in to the website or mobile app is to make a purchase. Addressing only the next session prediction may be enough to increase customer traffic, but, knowing the customers' tendency to make a purchase during the session is important to take more targeted actions in a short period of time. Hence, the prediction problem is treated with a multiclass approach to capture both the session engagement likelihood and purchase tendency.

Nearly 38 million data points representing the sessions are collected. State-of-the-art LightGBM and LSTM models are deployed for the task. Feature extraction and preprocessing are employed differently considering the different model requirements. Different strategies are employed to handle high imbalance. Results are ensembled with a simple approach to see if there will be a performance upgrade to the individual models.

The results suggest that LightGBM performed better than LSTM, while the ensembled model has the best performance overall. On the other hand, the employed undersampling technique boosted the performance. Although this study has made significant strides in addressing the research problem, there are certain limitations as discussed in the discussion section that warrants further exploration. In light of these findings, it is clear that session and purchase prediction remains a critical area of study with promising opportunities for future research and practical applications.

REFERENCES

1. Agnihotri, R., R. Dingus, M. Y. Hu and M. T. Krush, "Social Media: Influencing Customer Satisfaction in B2B Sales", *Industrial Marketing Management*, Vol. 53, pp. 172–180, 2016.

2. Bradlow, E. T., M. Gangwar, P. Kopalle and S. Voleti, "The Role of Big Data and Predictive Analytics in Retailing", *Journal of Retailing*, Vol. 93, No. 1, pp. 79–95, 2017.

3. Le, D. T., Y. Fang and H. W. Lauw, "Modeling Sequential Preferences with Dynamic User and Context Factors", *Machine Learning and Knowledge Discovery in Databases*, Vol. 9852, pp. 145–161, 2016.

4. Erevelles, S., N. Fukawa and L. Swayne, "Big Data Consumer Analytics and the Transformation of Marketing", *Journal of Business Research*, Vol. 69, No. 2, pp. 897–904, 2016.

5. Shmueli, G., "To Explain or to Predict?", *Statistical Science*, Vol. 25, No. 3, pp. 289–310, 2010.

6. Martens, D., F. Provost, J. Clark and E. Junqué de Fortuny, "Mining Massive Fine-Grained Behavior Data to Improve Predictive Analytics", *MIS Quarterly*, Vol. 40, No. 4, pp. 869–888, 2016.

7. Cirqueira, D., M. Hofer, D. Nedbal, M. Helfert and M. Bezbradica, "Customer Purchase Behavior Prediction in E-commerce: A Conceptual Framework and Research Agenda", *New Frontiers in Mining Complex Patterns*, Vol. 11948, pp. 119–136, 2020.

8. Morwitz, V. G. and D. Schmittlein, "Using Segmentation to Improve Sales Forecasts Based on Purchase Intent: Which "Intenders" Actually Buy?", *Journal of*

Marketing Research, Vol. 29, No. 4, pp. 391–405, 1992.

9. Hendriksen, M., E. Kuiper, P. Nauts, S. Schelter and M. de Rijke, "Analyzing and Predicting Purchase Intent in E-commerce: Anonymous vs. Identified Customers", ArXiv:2012.08777 [cs.IR], 2020.

10. Bucklin, R. E., J. M. Lattin, A. Ansari, S. Gupta, D. Bell, E. Coupey, J. D. C. Little, C. Mela, A. Montgomery and J. Steckel, "Choice and the Internet: From Clickstream to Research Stream", *Marketing Letters*, Vol. 13, No. 3, pp. 245–258, 2002.

11. Bucklin, R. E. and C. Sismeiro, "Click Here for Internet Insight: Advances in Clickstream Data Analysis in Marketing", *Journal of Interactive Marketing*, Vol. 23, No. 1, pp. 35–48, 2009.

12. Olbrich, R. and C. Holsing, "Modeling Consumer Purchasing Behavior in Social Shopping Communities with Clickstream Data", *International Journal of Electronic Commerce*, Vol. 16, No. 2, pp. 15–40, 2011.

13. Lo, C., D. Frankowski and J. Leskovec, "Understanding Behaviors that Lead to Purchasing", *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 531–540, New York, USA, 2016.

14. Moe, W. W., "Buying, Searching, or Browsing: Differentiating Between Online Shoppers Using In-Store Navigational Clickstream", *Journal of Consumer Psychology*, Vol. 13, No. 1, pp. 29–39, 2003.

15. Moe, W. W. and P. S. Fader, "Dynamic Conversion Behavior at E-Commerce Sites", *Management Science*, Vol. 50, No. 3, pp. 326–335, 2004.

16. Montgomery, A. L., S. Li, K. Srinivasan and J. C. Liechty, "Modeling Online Browsing and Path Analysis Using Clickstream Data", *Marketing Science*, Vol. 23, No. 4, pp. 579–595, 2004.

17. Sismeiro, C. and R. E. Bucklin, "Modeling Purchase Behavior at an E-Commerce Web Site: A Task-Completion Approach", *Journal of Marketing Research*, Vol. 41, No. 3, pp. 306–323, 2004.

18. Su, Q. and L. Chen, "A Method for Discovering Clusters of E-commerce Interest Patterns Using Click-stream Data", *Electronic Commerce Research and Applications*, Vol. 14, No. 1, pp. 1–13, 2015.

19. Park, C. H. and Y. H. Park, "Investigating Purchase Conversion by Uncovering Online Visit Patterns", *Marketing Science*, Vol. 35, No. 6, pp. 894–914, 2016.

20. Baumann, A., J. Haupt, F. Gebert and S. Lessmann, "Changing Perspectives: Using Graph Metrics to Predict Purchase Probabilities", *Expert Systems with Applications*, Vol. 94, pp. 137–148, 2018.

21. Bhatnagar, A., A. Sen and A. P. Sinha, "Providing a Window of Opportunity for Converting eStore Visitors", *Information Systems Research*, Vol. 28, No. 1, pp. 22–32, 2017.

22. Schellong, D., J. Kemper and M. Brettel, "Generating Consumer Insights from Big Data Clickstream Information and the Link with Transaction-Related Shopping Behavior", *European Conference on Information Systems*, Guimarães, Portugal, 2017.

23. Schellong, D., J. Kemper and M. Brettel, "Clickstream Data as a Source to Uncover con-Sumer Shopping Types in a Large-Scale Online Setting", *European Conference on Information Systems*, Istanbul, Turkey, 2016.

24. Suchacka, G. and G. Chodak, "Using Association Rules to Assess Purchase Probability in Online Stores", *Information Systems and E-Business Management*, Vol. 15, No. 3, pp. 751–780, 2016.

25. Suchacka, G., M. Skolimowska-Kulig and A. Potempa, "A k-Nearest Neigh-

bors Method for Classifying User Sessions in E-Commerce Scenario", *Journal of Telecommunications and Information Technology*, Vol. 3, pp. 64–69, 2015.

26. Schölkopf, B., "The Kernel Trick for Distances", *Advances in Neural Information Processing Systems*, Vol. 13, pp. 301–307, Denver, USA, 2000.

27. Li, Q., M. Gu, K. Zhou and X. Sun, "Multi-Classes Feature Engineering with Sliding Window for Purchase Prediction in Mobile Commerce", *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pp. 1048–1054, Atlantic City, USA, 2015.

28. Gohari Boroujerdi, E., S. Mehri, S. Sadeghi Garmaroudi, M. Pezeshki, F. Rashidi Mehrabadi, S. Malakouti and S. Khadivi, "A Study on Prediction of User's Tendency Toward Purchases in Websites Based on Behavior Models", *2014 6th Conference on Information and Knowledge Technology (IKT)*, pp. 61–66, Shahrood, Iran, 2014.

29. Yeo, J., S. Kim, E. Koh, S. W. Hwang and N. Lipka, "Predicting Online Purchase Conversion for Retargeting", *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 591–600, Cambridge, United Kingdom, 2017.

30. Ding, A. W., S. Li and P. Chatterjee, "Learning User Real-Time Intent for Optimal Dynamic Web Page Transformation", *Information Systems Research*, Vol. 26, No. 2, pp. 339–359, 2015.

31. Zeng, M., H. Cao, M. Chen and Y. Li, "User Behaviour Modeling, Recommendations, and Purchase Prediction During Shopping Festivals", *Electronic Markets*, Vol. 29, No. 2, pp. 263–274, 2018.

32. Sheil, R. O. R. R., H., "Predicting Purchasing Intent: Automatic Feature Learning Using Recurrent Neural Networks", ArXiv:1807.08207 [cs.LG], 2018.

33. Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", *Advances in Neural Information Processing Systems*, Vol. 30, pp. 3146–3154, Long Beach, USA, 2000.

34. "LightGBM", https://lightgbm.readthedocs.io/en/latest/index.html, accessed on 08.04.2023.

35. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, Vol. 9, pp. 1735–1780, 1997.

36. Chung, J., Çaglar Gülçehre, K. Cho and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", ArXiv:1412.3555 [cs.NE], 2014.

37. "sklearn.decomposition.PCA", https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html, accessed on 08.04.2023.

38. Halko, N., P. G. Martinsson and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions", *SIAM Review*, Vol. 53, No. 2, pp. 217–288, 2011.

39. Grubbs, F. E., "Procedures for Detecting Outlying Observations in Samples", *Technometrics*, Vol. 11, No. 1, pp. 1–21, 1969.

40. Tukey, J. W., *Exploratory Data Analysis*, Addison-Wesley, Boston, 1977.

41. James, G., D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, Springer, New York, 2013.

42. Kohavi, R., "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", *International Joint Conference on Artificial Intelligence*, Vol. 2, pp. 1137–1143, Montreal, Canada, 1995.

43. Bergstra, J., R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for Hyper-Parameter Optimization", *Advances in Neural Information Processing Systems*, Vol. 24, pp. 2546–2554, Granada, Spain, 2011.

44. Shahriari, B., K. Swersky, Z. Wang, R. P. Adams and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization", *Proceedings of the IEEE*, Vol. 104, pp. 148–175, 2016.

45. Bergstra, J., D. Yamins and D. D. Cox, "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures", *International Conference on Machine Learning*, Vol. 28, pp. 115–123, Atlanta, USA, 2013.

46. Arlot, S. and A. Celisse, "A Survey of Cross-validation Procedures for Model Selection", *Statistics Surveys*, Vol. 4, pp. 40 – 79, 2010.

47. Cawley, G. C. and N. L. C. Talbot, "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation", *Journal of Machine Learning Research*, Vol. 11, pp. 2079–2107, 2010.

48. Friedman, J. H., "Greedy Function Approximation: A Gradient Boosting Machine.", *Annals of Statistics*, Vol. 29, pp. 1189–1232, 2001.

49. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, The MIT Press, Cambridge, USA, 2016.

50. Powers, D. M. W., "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation", ArXiv:2010.16061 [cs.LG], 2020.

51. "keras", https://keras.io/about/, accessed on 08.04.2023.

APPENDIX A: MONETARY FEATURES

Name	Description	Type	Discrete/Continuous
first_spend_quartile_count_lm	1. Each person's daily total payment amount is calculated for the days they made purchase in the last month/6 months/year. 2. first, second, third and fourth quartiles are calculated for each day. 3.If a person's daily spending is in the first quartile, their first_spend_quartile_count is increased. 4. If a person's daily spending is lower than first_quartile-(1.5*(third_quartile-first_quartile)) then their spend_low_outlier_count is increased.	INT64	discrete
second_spend_quartile_count_lm		INT64	discrete
third_spend_quartile_count_lm		INT64	discrete
fourth_spend_quartile_count_lm		INT64	discrete
spend_low_outlier_count_lm		INT64	discrete
spend_high_outlier_count_lm		INT64	discrete
first_spend_quartile_count_l6m		INT64	discrete
second_spend_quartile_count_l6m		INT64	discrete
third_spend_quartile_count_l6m		INT64	discrete
fourth_spend_quartile_count_l6m		INT64	discrete
spend_low_outlier_count_l6m		INT64	discrete
spend_high_outlier_count_l6m		INT64	discrete
first_spend_quartile_count_l12m		INT64	discrete
second_spend_quartile_count_l12m		INT64	discrete
third_spend_quartile_count_l12m		INT64	discrete
fourth_spend_quartile_count_l12m		INT64	discrete
spend_low_outlier_count_l12m		INT64	discrete
spend_high_outlier_count_l12m		INT64	discrete
first_disc_quartile_count_lm	1. Each person's daily total discount usage amount is calculated for the days they made purchase in the last month/6 months/year. 2. first, second, third and fourth quartiles are calculated for each day. 3.If a person's daily discount usage is in the first quartile, their first_disc_quartile_count is increased. 4. If a person's daily spending is lower than first_quartile-(1.5*(third_quartile-first_quartile)) then their disc_low_outlier_count is increased.	INT64	discrete
second_disc_quartile_count_lm		INT64	discrete
third_disc_quartile_count_lm		INT64	discrete
fourth_disc_quartile_count_lm		INT64	discrete
disc_low_outlier_count_lm		INT64	discrete
disc_high_outlier_count_lm		INT64	discrete
first_disc_quartile_count_l6m		INT64	discrete
second_disc_quartile_count_l6m		INT64	discrete
third_disc_quartile_count_l6m		INT64	discrete
fourth_disc_quartile_count_l6m		INT64	discrete
disc_low_outlier_count_l6m		INT64	discrete
disc_high_outlier_count_l6m		INT64	discrete
first_disc_quartile_count_l12m		INT64	discrete
second_disc_quartile_count_l12m		INT64	discrete
third_disc_quartile_count_l12m		INT64	discrete
fourth_disc_quartile_count_l12m		INT64	discrete
disc_low_outlier_count_l12m		INT64	discrete
disc_high_outlier_count_l12m		INT64	discrete

Name	Description	Type	Discrete/Continuous
hb_hx_avg_adj_basket_value_lm	the ratio of the total amount spend to the number of orders without considering returns (adjusted with inflation)	FLOAT64	continuous
hb_hx_avg_adj_basket_value_l6m		FLOAT64	continuous
hb_hx_avg_adj_basket_value_l12m		FLOAT64	continuous
hb_hx_net_avg_adj_basket_value	the ratio of the total amount spend to the number of orders with considering returns (adjusted with inflation)	FLOAT64	continuous
hb_hx_discount_usage_lm	the number of times a discount is used	INT64	discrete
hb_hx_discount_usage_l6m		INT64	discrete
hb_hx_discount_usage_l12m		INT64	discrete
hb_hx_total_discount_used_lm	the amount of discount used (adjusted with inflation)	FLOAT64	continuous
hb_hx_total_discount_used_l6m		FLOAT64	continuous
hb_hx_total_discount_used_l12m		FLOAT64	continuous
hb_hx_monetary_adj_lm	the spending amount (adjusted with inflation)	FLOAT64	continuous
hb_hx_monetary_adj_l6m		FLOAT64	continuous
hb_hx_monetary_adj_l12m		FLOAT64	continuous
hb_hx_net_monetary_lm	the spending amount considering returns and cancels (adjusted with inflation)	FLOAT64	continuous
hb_hx_net_monetary_l6m		FLOAT64	continuous
hb_hx_net_monetary_l12m		FLOAT64	continuous
hb_hx_cargo_amt_adj_lm	the amount spent for shipment (adjusted with inflation)	FLOAT64	continuous
hb_hx_cargo_amt_adj_l6m		FLOAT64	continuous
hb_hx_cargo_amt_adj_l12m		FLOAT64	continuous
hb_hx_disc_avg_amt_per_trx_lm	the ratio of the discount amount used to the the number of transactions (adjusted with inflation)	FLOAT64	continuous
hb_hx_disc_avg_amt_per_trx_l6m		FLOAT65	continuous
hb_hx_disc_avg_amt_per_trx_l12m		FLOAT66	continuous
atc_price_avg_l12m	the average price of the products added to basket (adjusted with inflation)	FLOAT64	continuous
atc_price_avg_l6m		FLOAT64	continuous
atc_price_sum_l12m	the sum price of the products added to basket (adjusted with inflation)	FLOAT64	continuous
atc_price_sum_l6m		FLOAT64	continuous
atc_price_sum_lm		FLOAT65	continuous
avg_atl_price_l12m	the average price of the products added to a list (adjusted with inflation)	FLOAT64	continuous
avg_atl_price_l6m		FLOAT64	continuous
avg_atl_price_lm		FLOAT64	continuous
sum_atl_price_l12m	the total price of the products added to a list (adjusted with inflation)	FLOAT64	continuous
sum_atl_price_l6m		FLOAT64	continuous
sum_atl_price_lm		FLOAT64	continuous

APPENDIX B: EVENT FEATURES

Name	Description	Type	Discrete/Continuous
hb_hx_frequency_l2m	the number of purchase	INT64	discrete
hb_hx_frequency_l3m		INT64	discrete
hb_hx_frequency_l4m		INT64	discrete
hb_hx_frequency_l6m		INT64	discrete
hb_hx_frequency_l12m		INT64	discrete
hb_hx_discount_usage_lm	the number of times a discount is used	INT64	discrete
hb_hx_discount_usage_l6m		INT64	discrete
hb_hx_discount_usage_l12m		INT64	discrete
hb_hx_cancel_lm	the number of cancelled orders	INT64	discrete
hb_hx_cancel_l6m		INT64	discrete
hb_hx_cancel_l12m		INT64	discrete
hb_hx_return_lm	the number of returned orders	INT64	discrete
hb_hx_return_l6m		INT64	discrete
hb_hx_return_l12m		INT64	discrete
addtocart_l12m	the count of add to cart	INT64	discrete
addtocart_l6m		INT64	discrete
addtocart_lm		INT64	discrete
atl_count_l12m	the number of add to list event	INT64	discrete
atl_count_l6m		INT64	discrete
atl_count_lm		INT64	discrete
productview_l12m	the number of product views	INT64	discrete
productview_l6m		INT64	discrete
productview_lm		INT64	discrete
total_commments_lm	the number of comments	INT64	discrete
total_commments_l6m		INT64	discrete
total_commments_l12m		INT64	discrete
campaign_banner_click_count_l12m	the number of campaign banner clicks	INT64	discrete
campaign_banner_click_count_l6m		INT64	discrete
campaign_banner_click_count_lm		INT64	discrete
campaign_banner_view_count_l12m	the number of campaign banner views	INT64	discrete
campaign_banner_view_count_l6m		INT64	discrete
campaign_banner_view_count_lm		INT64	discrete
campaign_banner_click_ratio_l12m	the ratio of the campaign banner click counts to the campaign banner view count	FLOAT64	continuous
campaign_banner_click_ratio_l6m		FLOAT64	continuous
campaign_banner_click_ratio_lm		FLOAT64	continuous

Name	Description	Type	Discrete/Continuous
claim_count_lm	opened request count	INT64	discrete
claim_count_l6m		INT64	discrete
claim_count_l12m		INT64	discrete
claim_rejected_count_lm	rejected request count	INT64	discrete
claim_rejected_count_l6m		INT64	discrete
claim_rejected_count_l12m		INT64	discrete
claim_rejected_percentage_lm	the ratio of the number of rejected requests to opened requests	FLOAT64	continuous
claim_rejected_percentage_l6m		FLOAT64	continuous
claim_rejected_percentage_l12m		FLOAT64	continuous
nps_count_lm	the number of times a score is given	INT64	discrete
nps_count_l6m		INT64	discrete
nps_count_l12m		INT64	discrete
bannerclick_frequency_lm	the number of banner clicks	INT64	discrete
bannerclick_frequency_l6m		INT64	discrete
bannerclick_frequency_l12m		INT64	discrete
atl_count_l12m		INT64	discrete
atl_count_l6m	the number of add to list event	INT64	discrete
atl_count_lm		INT64	discrete
hb_hx_atc_click_over_order_count_l12m	the ratio of the number of purchase to the number of add to cart	FLOAT64	continuous
hb_hx_atc_click_over_order_count_l6m		FLOAT64	continuous
hb_hx_atc_click_over_order_count_1m		FLOAT64	continuous
hb_hx_avg_basket_size	the ratio of the number of skus purchased to the number of orders	FLOAT64	continuous
hb_hx_avg_net_basket_size	the ratio of the number of skus purchased minus returns and cancels to the number of orders	FLOAT64	continuous
hb_hx_one_timer_flg	if a customer made only one purchase in the last year	INT64	binary
hb_hx_frequency_per_recency_l12m	the ratio of the number of purchase in the last year to the recency	FLOAT64	continuous
hb_hx_frequency_per_time_l12m	the ratio of the number of purchase in the last year to the number of days passed since the first purchase in the last year	FLOAT64	continuous
hb_hx_avg_atc_order_ratio_l12m	the ratio of average revenue from the customer to the average price of the products added to cart (adjusted with inflation)	FLOAT64	continuous
hb_hx_avg_atc_order_ratio_l6m		FLOAT64	continuous
hb_hx_avg_atc_order_ratio_1m		FLOAT64	continuous
hb_hx_sum_atc_order_ratio_l12m	the ratio of total revenue from the customer to the total price of the products added to cart (adjusted with inflation)	FLOAT64	continuous
hb_hx_sum_atc_order_ratio_l6m		FLOAT64	continuous
hb_hx_sum_atc_order_ratio_1m		FLOAT64	continuous
avg_star_lm	average score given to products	FLOAT64	continuous
avg_star_l6m		FLOAT64	continuous
avg_star_l12m		FLOAT64	continuous
last_star	the last score given	INT64	discrete

Name	Description	Type	Discrete/Continuous
nps_avg_score_lm	the average score given	INT64	continuous
nps_avg_score_l6m		INT64	continuous
nps_avg_score_l12m		INT64	continuous
last_transaction_success_flg	if the last purchase is success	INT64	binary
last_transaction_returned_flg	if the last purchase is returned	INT64	binary
last_transaction_cancelled_flg	if the last purchase is cancelled	INT64	binary
isinactive_l2m	if the customer is inactive in the last 2 months	INT64	binary
isinactive_lm	if the customer is inactive in the last 1 month	INT64	binary
hb_hx_avg_inactive_l4m	average inactive score in the last 4 months	FLOAT64	continuous

APPENDIX C: RECENCY FEATURES

Name	Description	Type	Discrete/Continuous
hb_hx_T	the number of days passed since the first purchase in the last year	INT64	discrete
hb_hx_recency	the number of days passed since the last purchase in the last year	INT64	discrete
day_since_last_addtocart	the number of days passed since last add to cart	INT64	discrete
day_since_last_addtolist	the number of days passed since the last add to list	INT64	discrete
days_since_last_pv	the number of days passed since the last product view	INT64	discrete
claim_days_since_last_claim	the number of days passed since the last request	INTERVAL	discrete
day_since_last_bannerclick_l3m	the number of days passed since the last banner click	INT64	discrete

APPENDIX D: DEMOGRAPHICS FEATURES

Name	Description	Type
gender	man, female	CATEGORY
most_used_os	most used operating system for purchase. android, ios or windows	CATEGORY
last_used_os	operating system of the last purchase. android, ios, windows	CATEGORY
most_used_city_region	most used city region. ege, akdeniz etc.	CATEGORY
most_used_city_size	most used city size according to the population. big, medium, small	CATEGORY
last_state_region	last used city region. ege, akdeniz etc.	CATEGORY
last_state_size	last used city size according to the population. big, medium, small	CATEGORY
last_used_platform	the platform of the last purchase. desktop, mobile etc.	CATEGORY

APPENDIX E: SEQUENCE FEATURES

Name	Description	Type	Discrete/Continuous
trx_count	transaction count in a session	INT64	discrete
spent_amt	amount spent in a session	FLOAT64	continuous
trx_distinct_sku_count	number of distinct products bought in a session	INT64	discrete
trx_distinct_domain_count	number of distinct domains bought in a session	INT64	discrete
trx_distinct_h1_count	number of distinct h1 category bought in a session	INT64	discrete
trx_distinct_h2_count	number of distinct h2 category bought in a session	INT64	discrete
trx_distinct_h3_count	number of distinct h3 category bought in a session	INT64	discrete
pv_distinct_sku_count	number of distinct products viewed in a session	INT64	discrete
pv_distinct_domain_count	number of distinct domains viewed in a session	INT64	discrete
pv_distinct_h1_count	number of distinct h1 category viewed in a session	INT64	discrete
pv_distinct_h2_count	number of distinct h2 category viewed in a session	INT64	discrete
pv_distinct_h3_count	number of distinct h3 category viewed in a session	INT64	discrete
atc_distinct_sku_count	number of distinct products added to basket in a session	INT64	discrete
atc_distinct_domain_count	number of distinct domains added to basket in a session	INT64	discrete
atc_distinct_h1_count	number of distinct h1 category added to basket in a session	INT64	discrete
atc_distinct_h2_count	number of distinct h2 category added to basket in a session	INT64	discrete
atc_distinct_h3_count	number of distinct h3 category added to basket in a session	INT64	discrete
most_viewed_sku_pv_count	view count of the most viewed product	INT64	discrete
pv_frequency_by_sku	the most viewed product's view ratio over all views in a session	FLOAT64	continuous
most_viewed_domain_pv_count	view count of the most viewed domain	INT64	discrete
pv_frequency_by_domain	the most viewed domain's view ratio over all views in a session	FLOAT64	continuous
most_viewed_h1_pv_count	view count of the most viewed h1	INT64	discrete
pv_frequency_by_h1	the most viewed h1's view ratio over all views in a session	FLOAT64	continuous
most_viewed_h2_pv_count	view count of the most viewed h2	INT64	discrete
pv_frequency_by_h2	the most viewed h2's view ratio over all views in a session	FLOAT64	continuous
weekend flg	if the session is performed on weekend	INT64	discrete
daytime flg	if the session is performed between 10 am and 11 pm	INT64	discrete