Machine Learning based analysis of users' online behaviour

Gábor Kőrösi

Supervisor: Dr. Richárd Farkas

A thesis submitted for the degree of doctor of philosophy
of the University of Szeged

University of Szeged
Doctoral School of Computer Science

March 2022

Preface

The various data logging systems transmit a huge amount of information about their users' online behaviour patterns, which is easily accessible, especially since the availability of open databases. The goal of behavioural analysis is to predict a particular reaction to different topics. Early approaches favoured manual data processing and traditional machine learning methods. Today, however, advances in machine learning have made it possible to solve these time-consuming pre-processing tasks automatically. This thesis presents a wide range of techniques and experiments on two databases that push the boundaries of state-of-the-art techniques on traditional Machine Learning and Deep Learning networks. Generally, behavioural analysis is performed on aggregated data from time series, but several studies have pointed out that a great amount of information is lost with cumulative data and that it is therefore worth working with raw data. This dissertation investigates the applicability and effectiveness of Deep Learning method based predictive models using aggregated and raw data, using continuous and discrete time series, and as a result, provides insights into the operation of predictive models for MOOC courses and webshops.

Acknowledgments

First of all, I would like to thank my supervisor, Dr. Richárd Farkas, for his guidance and for supporting my work with his useful comments. I am indebted to my colleague Dr. Vinkó Tamás, who helped me with the webshop behavior analysis. I am indebted to Stanford University and Havasi Ferenc, who produced the datasets which enabled me to work on the relevant problems discussed here. I would also like to thank the anonymous reviewers of my publications for their useful comments and suggestions. I would like to thank Dr. Livia Szedmina for scrutinizing and correcting this thesis from a linguistic point of view. I would like to thank my wife Anita for her endless love, support and inspiration. Last, but not least, I wish to thank my family and friends for their constant love and support. I would like to dedicate this thesis to them as a way of expressing my gratitude and appreciation. This dissertation was supported by the National Research, Development and Innovation Office of Hungary through the Artificial Intelligence National Excellence Program (grant no.: 2018-1.2.1-NKP-2018-00008) and by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program. I am grateful for this support, which definitely acted as an accelerator for the submission of this thesis. I am also thankful to CAROL (Center for Advanced Research through Online Learning), of Stanford University for providing me with the dataset necessary to conduct my research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Events and activities of daily life are increasingly often taking place in the online space, including, for example, the purchase of durable goods and education. Both of these areas, shopping and learning, which until a few years ago existed almost exclusively in the traditional offline format, have changed significantly. This change poses new challenges for professionals working in these fields, as most of the methods and methodologies used to date have become completely obsolete and unworkable in the online space. This is particularly true of the expertise of offline shop assistants or the role of teachers in brick-and-mortar educational facilities, roles which were once indispensable, but have now become outdated. The disappearance of these roles has not gone unnoticed, given that many online businesses are struggling with dwindling customer numbers and decreasing effectiveness of online learning systems (such as Massive Open Online Courses - MOOCs) with effectiveness at barely 25-30%. While it is undeniable that the online presence has created considerable challenges for business and education managers, it has also opened up new opportunities that can be exploited, notably by involving data science professionals. The various online platforms have a wealth of log data.
There are three levels to dive into:

- **High-level**: The simplest high-level of access to log data includes users' purchases, the provisional and final contents of the shopping cart, and in the case of educational platforms, interactions with videos, tutorials and quizzes.

- **Middle-level**: More in-depth than the previous category, the middle-level provides information on the time spent on the page and the order of the items involved within the page.

- **Low-level**: In addition, some log systems go deeper into hardware interactions, where mouse clicks and movements, keyboard press habits are stored, which is called the low-level information space.

With this data, one can create support systems and decision support systems that, in addition to aiding the operator, also improve the user experience. The topic of this dissertation is the development of different Machine Learning methods for webshop and MOOC applications based on log data analysis.

What all applications have in common is the creation of aggregated databases, so-called user profiles, using log data of different widths and depths, which are used for classification, regression or even clustering. For more than fifteen years now there has been active research on the analysis of user log data. Initially, research and development were carried out in isolation on small databases in research teams or on closed internal databases in companies. In recent years, as online business and online educational interfaces have become more common, the number of real business applications and the amount and depth of data generated by each application have increased. Therefore, the previously traditional feature extraction and Machine Learning methods have been replaced by Deep Learning methods, which can provide high- quality solutions for large amounts of data, even starting from low-level data.

Altogether this dissertation contains 7 chapters, composed of separate studies implementing the above-mentioned approaches. In the first two chapters, I will present the special challenges of log data collection and preparation on high-level log databases. I will describe forecasting results on a real-life Hungarian Webshop database, and the MOOC course 'Conscious and safe Internet use', which was developed and launched as a cooperation of two departments of the University of Szeged. Apart from the data collection and formulation of solutions, this dissertation also proposes application-specific feature sets. Through these training and evaluation databases, I will present several comparative Machine Learning experiments. Based on the experience of the feature space design work, I focused my efforts on the possibility of building end-to-end systems using Deep Learning algorithms directly from low-level log data. I subjected the data to minimal data processing and then successfully applied different neural network architectures. For the Deep Learning experiments, I used the log data from the Education-115-Spring-2014 MOOC course at Stanford University consisting of 39.5 million records. The experimental results achieved are primarily determined by the user profiles and data preprocessing techniques employed. The Deep Learning models outperform classical Machine Learning methods based on feature extraction in accuracy, but they are black-box in nature, which hinders their real-life application (for instance, an instructor who does not trust the prediction of a black box). In the last chapter, I propose three visualization methods for interpreting deep neural networks learned over sequential log data, which can contribute to human experts' better understanding of the patterns observed from the data.

The structure of the dissertation

In my dissertation, I am going to introduce online user behaviour modelling techniques and several empirical experiments. In Chapter 2, I summarize the application area and research challenges

of webshops and MOOC sites through a literature review. Special attention has been paid to summarizing the different classical Machine Learning and Deep Learning techniques in the field, as well as to investigating data processing and feature extraction solutions.

Using log data from a webshop which is based in Hungary and operating within the borders of the EU, I created an actual business solution to build a reaching model for targeted offers and promotional mailings. In order to achieve this, I implemented and analysed the shopping habits and behavioural patterns of users (high-level log data). This process consisted of feature extraction methods, where new aggregated preprocessed data was created from the log data. Using a combination of so-called combined regression and classification models on a hand-crafted feature space, I carried out various prediction experiments that were successfully applied to real business operations, which are presented in detail in Chapter 3.

Chapter 4 describes the next step of this research. Namely, in collaboration with the Software Engineering Department of the University of Szeged, I developed a MOOC course with the title 'Conscious and safe Internet use' along with the full stack user interface. This system records the task completion (middle-level log data), mouse movements and video viewing log data (low-level log data). I managed a dataset of 'course completed' among students aged 10-21 which yielded nearly 4.5 million log records from 510 students. On the resulting, considerably richer middle- and low-level datasets, I proposed preprocessing and feature extraction methods, and then investigated the effectiveness of traditional Machine Learning, as conducted for the previous, webshop research. The results highlighted the shortcomings of this dataset and the drawbacks of the implemented methods, but also proved the initial hypothesis that methods using richer and longer time series of middle and high-level log data provided higher efficiency.

In the final two parts of this dissertation, Chapters 5 and 6, I present student performance predictive models using 39.5 million log data of 142,395 students enrolled in a MOOC course at Stanford University. Instead of the difficult and time-consuming feature space extraction, I investigated raw, clickstream- level data and Deep Learning methods that could handle the raw sequences. In Chapter 5, I argue that given such a large training dataset, these methods are significantly more accurate than classical methods based on feature extraction. In Chapter 6, I compare convolutional and recurrent neural network architectures on the Stanford MOOC dataset. Further, I provide insights into numerical and discrete sequential data processing techniques, where I investigated embeddings per variable. Lastly, I also propose three visualization techniques for deep neural networks trained on sequential log data to help, to aid learning site owners in understanding the patterns learnt by the neural models.

Chapter 2

Background

Analyzing users online behavior using Machine Learning has a long-standing history, but analyzing user interaction log data from MOOCs and webshops directly on raw, low-level time series is a relatively new area of research. In terms of analyzing and predicting user behavior, there are many different methods available that will lead to the solution. Most traditional approaches rely mainly on classic Machine Learning, they typically work on the accumulated, aggregated dataset. In addition to those conventional methods, nowadays more and more studies tend to apply different Neural Network models. This chapter presents a review of the use cases on two application areas as well as various prediction approaches.

2.1 Online user behavior analysis

Big data is revolutionizing the digital world, influencing the way of making business and strategic decisions making. This area is evolving at an impressive speed. While more and more public data is being created, more and more expensive tools for data analysis are becoming low-cost solutions, available to users on a large scale. For example, several years ago, time series research was carried out within the framework of universities, whereas now countless small and medium-sized webshops have adopted this in their daily business. Another instance is the research on Neural Networks capable of interpreting models, which have once again come into focus. There are countless excellent studies on this topic in which user behavior patterns are examined and the expected attitudes predicted based on a sequenced dataset.

2.2 E-commerce user behaviour analysis

While in a traditional shop the tracking of customers may be cumbersome (e.g., loyalty card program), the back-end of a webshop offers numerous solutions for this task. Available methods include

cookies, tracing spending, sending newsletters and product tracking, among others (Ahmed et al., 2011; Banerjee and Ghosh, 2001; Grbovic et al., 2015). The main driving force behind this fast evolution is to understand and anticipate user behavior better, so that related questions can be answered in real-time. The main goal is to obtain the highest response from users by spending as little time and money on it as possible, and to create customer-oriented services (Aly et al., 2012). This is called personalization and targeting (Essex, 2009), where the objective is to find the best matching ads or form of sales promotion to be displayed for each user. This application field is not new, given that there were already similar solutions implemented in the first-generation webshops, but nowadays the amount of data is much greater than before. As data is increasing, more and more companies are demanding high-quality solutions from their data scientists.
In e-commerce content online behavior analysis is used for recommendation systems which is a quite general concept. It may be based on the collaborative filter solution, the content based method, the classification or regression, and their implementation in different depths and widths. Collaborative filtering (CF) is probably one of the most commonly used and most well-known technologies. The underlying concept in this solution is that, on the basis of the users' historical data, the users are put into an n^{th} dimensional space , make it thus possible to measure the distance between them. In light of this, recommendations were formulated based on the data of the users that portrayed similar behavior. Although this CF technique has proven its power, it does have disadvantages, including the huge amount of work it requires as well as the cold start problem, data sparsity, and scalability. Besides collaborative filtering, the second most popular solution is the content based method. It is a technique which operates with unique characteristics and behaviors of each customer, and in turn, delivers personalized content for each user, based on their content consumption history across channels. Another noteworthy option is the community based method. This approach works with the assumption that the content stemming from a given user's friends or authoritative users is more likely to be relevant for said user than other, non-relatable content. While collaborative filtering and content based models use only a static 'user snapshot', there are numerous papers which use uni- or multivariate user event history, i.e., sequences, time series to build a predictive model, so that the predictions can be utilized in recommendation systems (Lucas et al., 2013; Chen et al., 2014; Tian et al., 2019; Bozanta and Kutlu, 2018; Burke, 2002). Chapter 3 will introduce solutions for a specific e-commerce user-level prediction task.

2.3 Educational data mining

In today's world virtual online educational platforms emerge literally on a daily basis. Mushrooming as a scalable lifelong learning paradigm, online educational platforms have enjoyed significant high in recent years, both in industry and academia (Haggard, 2013). With the emergence of Coursera and eDX (collectively known as Massive Open Online Courses - MOOCs), educational platforms have

gained an additional impetus, a new aspect in their evolution process that has opened up a novel field of research in the context of data mining, thanks to the extraction of logging information. While thousands of students have been attracted to large online classes, keeping them motivated has become proven to be quite challenging (D. Liang, 2014). Learning outside the confines of an educational institution and without the supervision of a teacher can pose certain obstacles. As an example, the T. Sinha and Dillenbourg (2014) revealed that one of the most crucial aspects about MOOCs is that they require a higher-than-average level of self-regulation by students. Several studies reported that in online learning environments, students have poorer than average performance, lower motivation and measurably fewer experiences of success, which are specifically identified as the downside of MOOCs (Y. Bergner, 2012; C. G. Brinton and Ju, 2015; J. Guan, 2002; J. Cheng, 2013; D. Liang, 2014; T. Sinha and Dillenbourg, 2014; P. Esztelecki, 2016). In addition, one of the most controversial issues is the high dropout rate, analyzed in dozens of studies. It would therefore be paramount to gain an understanding of the students' motivation, or the reasons behind the loss thereof.

Table 2.1: The primary categories of educational data mining

Category of Method	Goal of Method	Key applications
Prediction	Develop a model which can infer a single aspect of the data (predicted variable) from some combination of other aspects of the data (predictor variables)	Detecting student behaviors (e.g.gaming the system, offtask behavior, slipping); Developing domain models; Predicting and understanding student educational outcomes
Clustering	Find data points that naturally group together, splitting the full dataset into a set of categories	Discovery of new student behavior patterns; Investigating similarities and differences between schools
Relationship Mining	Discover relationships between variables	Discovery of curricular associations in course sequences; Discovering which pedagogical strategies lead to more effective/robust learning
Discovery with Models	A model of a phenomenon developed with prediction, clustering, or knowledge engineering, is used as a component in further prediction or relationship mining.	Discovery of relationships between student behaviors, and student characteristics or contextual variables; Analysis of research question across wide variety of contexts
Distillation of Data for Human Judgment	Data is distilled to enable a human to quickly identify or classify features of the data.	Human identification of patterns in student learning, behavior, or collaboration; Labeling data for use in later development of prediction model

Ironically, the dropout and loss of motivation problems can be solved through the online platform itself because its structure allows all around logging of student activities, providing some previously

unknown tools of pedagogical research. This idea was based on a study by Romero and S.Ventura (2010), who argued that if learning management systems already accumulated a lot of log data on students' activities, those could be used for research purposes. An educational system can automatically record whatever student activities are involved, such as reading, writing, taking tests, performing various tasks, and even communicating with peers (J. Mostow and Heiner, 2005). That data can be aggregated over large numbers of students and can contain many variables that data mining algorithms and techniques can explore for model building (Prabha and Shanavas, 2014). Working from student data can help educators both track academic progress and understand which instructional practices are effective (L. Cao, 2009).
Following Romero and S.Ventura (2010) line of thought, over the last decade, a number of studies were carried out in various institutions, most of which were promising and made significant breakthroughs in the field nownamed Educational Data Mining (EDM). Its origin dates back to an Educational Data Mining conference in 2008, where the idea of educational data mining of MOOC courses first emerged. Educational Data Mining is a multidisciplinary area in which some of the most useful data mining tasks and methods are: statistics, visualization, clustering, classification, association rule mining, sequential pattern mining, text mining, etc. The goal is the discovery of non-obvious valuable patterns from a large collection of data (W. Klosgen, 2002).
Several studies (Luan, 2002; Baker, 2010) showed that data mining can be used to detect at-risk students and help institutions become more proactive in identifying them and responding to their issues. Four main axes can be identified along which EDM methods may be helpful for constructionist research:

- EDM methods do not require constructionists to abandon deep qualitative analysis for sim plistic summative or confirmatory quantitative analysis;
- EDM methods can generate different and complementary new analyses to support qualitative research;
- By enabling precise formative assessments of complex constructs, EDM methods can support an increase in methodological rigor and replicability;
- EDM can be used to present comprehensible and actionable data to learners and teachers in situ;
- In order to investigate those axes, the first step is to describe one's perspective on compatibilities and incompatibilities between constructionism and EDM (M. Berland and Blikstein, 2014).

The strengths of EDM systems can be traced back to their tools, in particular the logging methods, which provide information to researchers, who could then discover previously unknown pedagogical implications. Baker summarizes in Table 2.1 what is known about these tools and the

results in Baker (2010).
As stated previously, educational data mining covers areas that directly affect students and their learning methods (Huebner, 2013). Although EDM has been around for more than a decade, it is still an emerging field that from time to time reveals new and unique ways of exploring problems related to education (Huebner, 2013). All research results from the 2008 conference were published, confirming that it is worth digging deeper. Numerous successful methods for improving education in the online space were presented. For example, in a review of relevant studies Huebner (2013) revealed works that suggested ways to keep students in a learning environment, to find more effective teaching techniques and create better curricula, or help reduce dropout rates in a predictive way (Huebner, 2013). Along this line of thinking, in a successful case study, researchers at Bowie State University assigned risk factor scores to each student that indicated who would have difficulty (F. Chacon, 2012). This research piqued my interest, and based on it, using log data from Stanford University's online course, I attempted to predict the expected behavior of users at a particular point in the course, as detailed in Chapters 5 and 6.

2.4 Type of data

When discussing sequential data or time series processing, one must not forget that this process actually starts with the collection and processing of data. In terms of time series, there are univariate or multivariate time series, continuous or discrete sets of variables, each requiring a specific preprocessing or transformation. The depth of the users' online activity log data can be divided this into three levels:

- **High-Level**: The simplest type of data-set is the high-Level of log data, which includes, for example, users' shopping carts and bills, and for educational platforms, quiz scores, basic interaction with course content. This dataset typically stores only the most important actions. For example, in a webshop, what is known are the contents of purchased baskets (purchase history), registration date, and login date. However, there is no information about what the user did while on the site.

- **Middle-level**: middle-level data contains more information on user operations. This level, for example, reveals some information in chronological order with a timestamp, which is not available in high-level data. These could be, e.g., in the case of a webshop, the details of logins and logouts, the times and exact dates of the pages opened, or the eventual process of loading the shopping cart. midle-level data could be the clicks on products and advertisements or a user interface action , like hitting a button.

- **Low-level**: The deepest data container is the low-level data. Here, in addition to the data types already listed, one can access the number of characters typed, the pattern of mouse clicks,

the cursor path, and all interactions between the user and the browser page. Other accessible data includes the number of mouse scrolls performed while reading course material, the path of mouse movements, or the path of answer changes performed while completing quizzes.

2.5 Prediction methods and algorithms

Several approaches have been used in the literature to process sequences and build predictive models. The simplest methods work with uni-variate data and try to predict its value. In another approach, multivariate data is cumulated over days, weeks, or any given time period, and then a classical Machine Learning (classification or regression) approach is applied. A somewhat more complex yet much more efficient method is the use of Neural Networks, which makes work easier by allowing some or all of the tedious feature engineering to be omitted.

2.5.1 Evaluation methodology

To develop a Machine Learning model, it is necessary to, often randomly, split the dataset into training data and test data. The training data is used to train the Machine Learning model and the same model is tested on independent test data to evaluate the model performance. Test data should be kept independent of training data to avoid data leakage. The ML model development should use the test data to evaluate the model performance. Cross-validation is a resampling procedure used to evaluate Machine Learning models and how the model will perform on an independent test data set. It has eight different types, though in this thesis work I only implemented two, namely Leave One Out Cross-Validation and K-fold Cross-Validation. These techniques are summarized below.

- **Leave-One-Out Cross-Validation (LOOCV)** is an exhaustive cross-validation technique. For a dataset of n rows, row1 is selected for validation and the remaining (n-1) rows are used for model training. For the next iteration, row2 is selected for validation and the others are selected for model training. The process is repeated in a similar manner for n steps or the desired number of operations. This cross-validation method learns and tests in all possible ways.

- **K-fold Cross-Validation** is a technique where the original dataset is equally divided into k parts or folds. From the k folds or groups, one group is selected as validation data at each iteration and the remaining (k-1) groups are selected as training data. The final accuracy of the model is calculated based on the average accuracy of the validation data of the k-model.

2.5.2 Classic Machine Learning methods

When considering the bulk of the work on EDM, it must be pointed out that a significant percentage of the studies used the easiest-to-apply method first. Examples include Decision Tree based models, or simpler regression approaches which proved to be highly popular in EDM related research (Y. Bergner, 2012; Baker, 2010; Aldowah et al., 2019; Pigeau et al., 2019; Baker and Inventado, 2014; X. Wang and Rose, 2015). Following this line of investigation, I used similar approaches in some of the theses of my dissertation. In the next few pages, I briefly describe the supervised machine learning methods used in this work, drawing on books by Fielding (1999) and Sarker (2021), among others.

Support Vector Machine

The Support Vector Machine (SVM) algorithm can be formulated in the following simple manner: in the Support Vector Machine (SVM) algorithm, each data item is represented as a point in n-dimensional space (where n is the number of features), where the value of each feature is a coordinate value. Classification is then performed by finding the hyper-plane that distinguishes the two classes very well. The support vectors are simply the coordinates of each observation. The SVM classifier is a boundary line that best separates the two classes (hyper-plane/line).

K-Nearest Neighbor Classifier

The K-Nearest Neighbor Classifier (k-NN) algorithm is a straightforward, easy-to-implement supervised machine learning approach. The algorithm assumes the similarity between the new case/data and the available cases, and classifies the new case into the category that is most similar to the available categories. The trained model stores all the available data and classifies the new data point based on the similarity. This means that when new data appears, it can be easily classified into a well-matched category using the k-NN algorithm. The k-NN algorithm only stores the dataset in the training phase and when a new piece of data is received, it classifies that given piece of data into a category that is very similar to the new data.

Decision Tree

Tree based algorithms are one of the best and most commonly used supervised learning methods over limited sized feature spaces. Tree based algorithms provide highly accurate, stable and easy-to-interpret predictive models. Unlike linear models, they represent non-linear relationships quite well and they can be adapted to solve any type of problem, as well (classification or regression). Methods such as decision trees, random forests, gradient boosting are popularly used in all kinds of data science problems.

- **Decision tree** is a type of supervised learning algorithm with predefined target variables,mostly used for classification or regression problems. It works for both categorical and continuous input and output variables which could be highly useful for high-level data which contain discreet valued variables. In this technique, a population or sample is divided into two or more homogeneous sub-populations based on the most significant divisor/discriminator of the input variables.

- **Conditional inference trees (Ctree)** are another type of decision trees that use recursive partitioning of dependent variables based on the value of correlations. Like other algorithms for classification and regression in Machine Learning, it avoids bias, therefore, it also avoids vulnerability to errors, making it more flexible to problems in the data. Conditional inference trees use a significance test, which is a permutation test that selects a covariate for variable partitioning and recursivity. The p-value is calculated in this test. The significance test is performed at each run of the algorithm. This algorithm is not suitable for learning data with missing values.

Ensemble Tree based models

Like any other model, the tree based algorithm suffers from bias and variance, which can be countered by the ensemble methods when trying to remedy those. Ensemble methods involve a group of predictive models to achieve better accuracy and model stability. Ensemble methods are known to give a top boost to tree based models. In the ensemble models the trees are added to the ensemble one by one and are fitted to correct for prediction errors made by previous models. This is a type of ensemble Machine Learning model called gradients boosting. Some commonly used ensemble methods include Bagging, Boosting and Extreme Gradient Boosting.

- **Random Forest** approach works in a simple but effective way, building several decision trees and combining them to obtain a more accurate and stable forecast. It is a type of ensemble learning method where a set of weak models are combined to form a strong model. The random forest has almost the same hyper-parameters as the decision tree, but uses the classifier-class of the random forest for a better solution. The other difference compared with the Decision tree is that this approach adds additional randomness to the model while growing the trees. The Decision tree searches for the most important feature when partitioning a node, whereas this method searches for the best feature among a random subset of features. So, in the random forest, the algorithm considers only a random subset of features when partitioning nodes. The model can make the trees even more random by also using random thresholds for each feature, rather than looking for the best possible threshold. This leads to a wide range of variation, which generally results in a better model.

- **Bagging** decision trees are sensitive to the specific data on which they have been trained.

If the training data is changed, the resulting decision tree may be quite different, and consequently the predictions may also be quite different. This problem is known as the problem of algorithms with high variance, which the bagging algorithm tries to solve by reducing the variance. Bagging is an ensemble technique that often takes homogeneous weak learners, trains them in parallel, independently of each other, and combines them following some deterministic averaging process. The bagging algorithm consists of three basic steps:

  - In the first step, bagging uses a bootstrapping sampling technique to generate diverse samples. This resampling method creates different subsets of the training dataset by randomly and surrogately selecting data points. This means that each time one selects a data point from the training dataset, one can select the same instance multiple times. As a result, a value may be repeated twice or even more in a sample.
  - In the second step, parallel learning is initiated, where bootstrap patterns are trained independently and in parallel with each other using weak or base learners.
  - Finally, the average or majority of predictions is taken to calculate a more accurate estimate. In the case of regression, the average of all the outcomes predicted by each classifier is taken (soft voting). For classification problems, the class with the largest majority of votes is adopted (hard voting or majority voting).

- **Boosting** The way boosting works is rather similar to bagging, the main difference lies in the way it is trained. In bagging, weak learners are trained in parallel, whereas in boosting, they are trained sequentially. This means that a sequence of models is built and with each new model iteration, the weights of the data misclassified in the previous model are increased. This redistribution of weights helps the algorithm to identify parameters which to focus on in order to improve performance. Boosting methods are typically used when low variance and high bias are observed. The AdaBoost (adaptive boosting algorithm) is one of the most popular boosting algorithms, as it was one of the first such approaches.
  Tree boosting is a highly effective and widely used Machine Learning method. Chen and Guestrin (2016) described a scalable end-to-end tree boosting system called XGBoost(Extreme Gradient Boost), which is used widely by data scientists to achieve top results on many Machine Learning challenges. XGBoost provides parallel tree boosting that solves many data science problems quickly and accurately. I opted for using XGBoost in the studies presented in Chapters 3 and 4.

Ridge regression

Ridge regression is a way of creating a parsimonious model when the number of predictor variables exceeds the number of observations, or when there are correlations between predictor variables in the dataset. Ridge regression works by shrinking the coefficients or weights of the regression

model towards zero. This is achieved by applying a quadratic penalty to their size. This technique analyses multiple regression data that suffers from correlations between predictor variables. If there are correlations between one or more predictor variables, the least-squares estimates are unbiased,but their standard deviations are large and so they may be far from the true value. By adding a degree of bias to the regression estimates, spine regression reduces standard errors.

Multilayer Perceptron

The Multilayer Perceptron (MLP) is a feed-forward neural network often used for simpler classification tasks. In my work I used this method exclusively on cumulative data. The Multilayer Perceptron consists of three layers: input layer, output layer and hidden layer. The input layer receives the input signal to be processed. The output layer performs the purpose, such as prediction and classification. Between the input layer and the output layer there are any number of hidden layers in the MLP. In the MLP, data flows forward from the input layer to the output layer in a similar way to a feed forward network. The neurons in MLP are trained using the back propagation learn ing algorithm. MLP can approximate any continuous function and can solve problems that are not linearly separable.

Feature selection methods

In classical Machine Learning, it is advisable to select the most important features. Several solutions can be used to do this, for my work described in Chapter 4, I chose information gain for its ease of implementation and robustness. The information gain calculation starts by determining the information of the training data. The information in a response value, r, is calculated in the following expression:

$$-\,log\left(\frac{freq\,(r,T)}{|T|}\right) \tag{2.1}$$

T represents the training data and $|T|$ is the number of observations. To determine the expected information of the training data, sum this expression for every possible response value:

$$I(T) = -\sum_{i=1}^{n} \frac{freq(r_i mT)}{|T|} * log_2(\frac{freq(r_i), T}{|T\|}) \tag{2.2}$$

Here, n is the total number of response values. This value is also referred to as the entropy of the training data. Next, consider a split S on a variable X with m possible attributes. The expected information provided by that split is calculated by the following equation:

$$I_s(T) = \sum_{j=1}^{m} \frac{|T_j|}{|T|} * I(T_j) \tag{2.3}$$

In this equation, T_{j} represents the observations that contain the j^{th} attribute. The information gain of split S is calculated by the following equation:

$$G(s) = I(S) - I_s(T) \tag{2.4}$$

Information gain ratio attempts to correct the information gain calculation by introducing a split information value. The split information is calculated by the following equation:

$$SI(S) = -\sum_{j=1}^{m} \frac{|T_j|}{|T|} * log_2 \left(\frac{|T_j|}{|T|} \right) \tag{2.5}$$

As its name suggests, the information gain ratio is the ratio of the information gain to the split information:

$$GR(S) = \frac{G(S)}{SI(S)} \tag{2.6}$$

2.5.3 Neural Network based predictors for sequential data

The Multilayer Perceptron (MLP) is so popular because of its simple design and efficiency, but this simple approach is worth digging much deeper into neural networks. In addition to the classical approach, there is research available on temporal prediction using recurrent or convolutional Neural Net works to process raw or partially preprocessed sequences. In several of the studies of this dissertation, I focused on similar solutions which were based on Neural Networks. These methods are described in more detail in the following sections.

Recurrent Neural Networks

When working with sequential data, classic feed-forward networks may not be the best for learning and forecasting. In such cases, a mechanism is needed that can preserve past information to predict future values. Recurrent Neural Networks, or RNNs for short, are a variant of traditional feed-forward artificial Neural Networks that can handle sequential data and can be trained to retain past knowledge. RNNs are a group of Neural Networks that allow the use of previous outputs as inputs while having hidden states. These networks represent a special type of artificial Neural Networks that can process data in time series or sequences. Simple feed-forward Neural Networks are only suitable for independent data points. However, if the data are in a sequence so that one data point is dependent on a previous data point, then the Neural Network must be modified to recognize and use the dependencies between these data points. Recurrent Networks have a "memory unit" that helps them store the state of previous inputs, which is later used in the next output of the sequence. There are different architectural types of recurrent Neural Networks (Figure 2.1), such as:

- One To One - Traditional Neural Network (1:1)

- One To Many - Music, text generation (1:N)
- Many To One - Sentiment or sequence classification, regression (N:1)
- Many To Many - Name entity recognition (N:N)
- Many To Many - Machine translation (N:M)

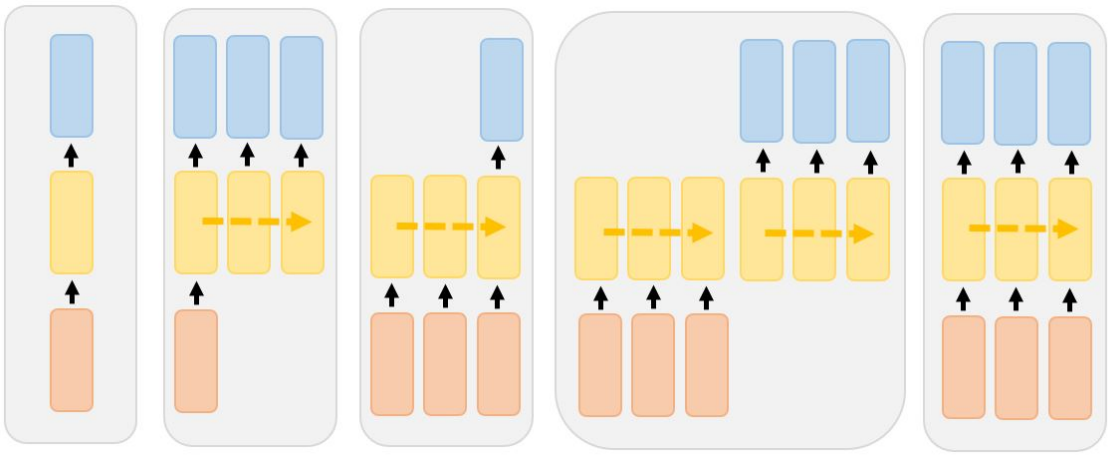


Figure 2.1: Illustration of different architectural types of Recurrent Neural Networks . From left to right: One to One, One to Many, Many to One, Many to Many, Many to Many

In natural language processing, time series prediction tasks, recurrent Neural Networks have been more widely used due to their ability to memorize long-range, sequence dependencies of data. However, simple RNNs are prone to problems associated with gradient diminishing or explosion. RNNs can be considered deep Neural Networks over many time instances, the gradients at the end of a sequence may not be able to back-propagate to the beginning of the sentence because of the many layers of nonlinear transformations (Yao et al., 2015). To solve this problem, there are different variations of RNNs that are being applied:

- Bidirectional recurrent Neural Networks (BRNN) are a variation of RNNs where the inputs of future time steps are used to improve the accuracy of the network. It can be compared to predicting the middle words based on the knowledge of the first and last words of a sentence.
- Long Short Term Memory (LSTM) are also designed to solve the vanishing gradient problem of RNNs. The LSTM uses three gates called input, output and forget gates. Like the GRU, these gates determine which information is to be retained.
- Gated Recurrent Units (GRU) is designed to deal with the disappearing gradient problem. They have a reset and a refresh gate. These gates determine which information is to be kept for future predictions.

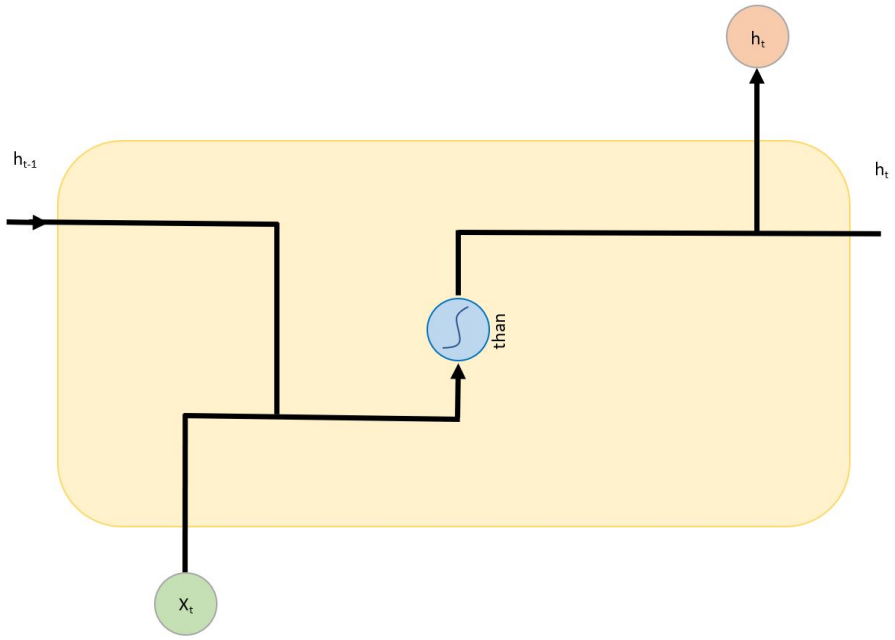


Figure 2.2: Illustration of the RNN layer, where x_t is the input and h_{t-1} are the output and the previous hidden state. The h_t update of the recurrent hidden state in the vanilla RNN is as follows $h_j^{(t)} = g([Wx_t]_j + [Uh_{t-1}]_j)$ where g is a smooth, bounded function such as a logistic sigmoid function or a hyperbolic tangent activation function. While x and h_{t-1} are the input and the previous hidden state, respectively.

Long Short-Term Memory (LSTM) Recurrent Neural Network

In a conventional recurrent Neural Network, the error signals "flowing" backwards in time either ex- plode or disappear exponentially, depending on the size of the weights, as the backward propagating error evolves over time. This can lead to oscillating weights, moreover, learning to bypass long time delays takes excessively long times, or does not work at all. As a solution Hochreiter and Schmidhuber (1997) proposed their new recurrent network architecture which would solve this problem, and named it Long short-term memory (LSTM). LSTM is designed to overcome these error feedback problems. It is able to learn to traverse time intervals greater than 1000 steps even for noisy, incompressible input sequences, without loss of short time-shifting capabilities. This is achieved by an efficient gradient based algorithm for an architecture that imposes a constant error flow through the internal states of specialized units. The LSTM network is based on the forget gate, input gate, cell state, output gate. To make the equations uncluttered, I omitted biases. Based on (Hochreiter and Schmidhuber, 1997) approach, I described how the activation of the j^{th} hidden unit is computed. The extent to which the existing memory is forgotten, is modulated by a forget gate f_j , which is computed by

$$f_t = \sigma([W_f X_t]_j + [U_f h_{t-1}]_j) \tag{2.7}$$

where σ is a logistic sigmoid function. $U_f$and W_f are the weights to be learnt.The forget gate is controlled based on the input x_t and the previous hidden state h_{t-1}.
The extent to which new memory content is transferred to the memory cell is modulated by an input gate:

$$i_t = \sigma([W_i X_t]_j + [U_i h_{t-1}]_j) \tag{2.8}$$

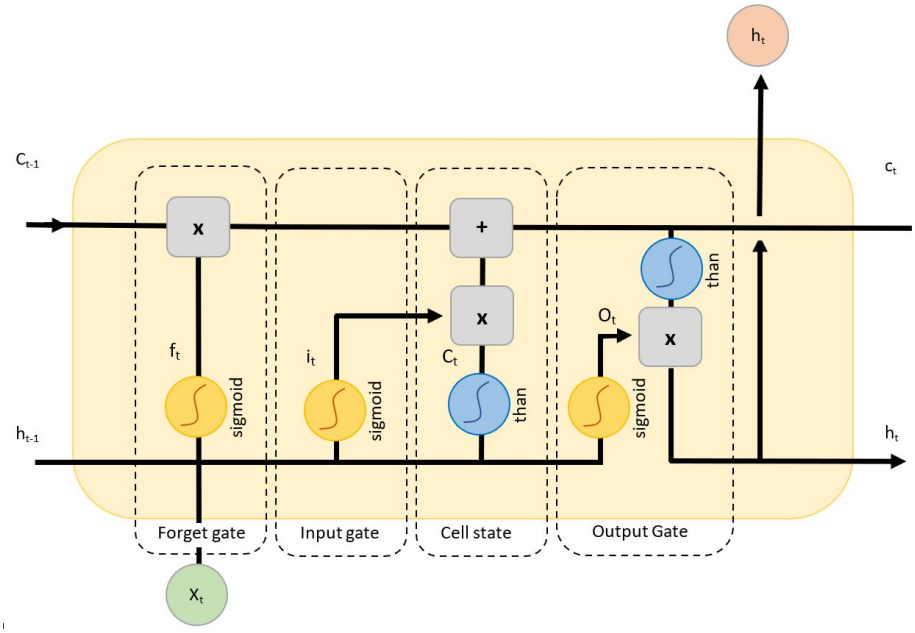


Figure 2.3: Illustration of the LSTM layer, where x_t is the input and h_{t-1} are the output and the previous hidden state, *sigmoid* and *than* are activation functions

The input gate is also controlled by the x_t input and the previous hidden state h_{t-1}. However, the weights of the input gate are independent of the weights of the forget gate.
The memory cell c_t is updated by partially forgetting the existing memory and adding a new memory content $c_t^{'}$:

$$c_t = f_t \odot c_{t-1} + i_t \odot c_t^{'} \tag{2.9}$$

where the new memory content $c_t^{'}$ is:

$$c_t^{'} = \sigma([W_c X_t]_j + [U_c h_{t-1}]_j) \tag{2.10}$$

The o_t is the output gate that modulates the amount of exposure of memory content:

$$o_t = than([W_o X_t]_j + [U_o h_{t-1}]_j) \tag{2.11}$$

The LSTM unit output h_t, or activation is :

$$h_t = o_t \odot than(c_t) \tag{2.12}$$

Gated Recurrent Unit (GRU) network

Sequence based Neural Networks are recurrent Neural Networks, with feedback connections enclosing several layers of the network. Gated Recurrent Unit (GRU) networks is an example of recurrent Neural Networks. GRU is good at solving problems that require learning long-term temporal dependencies. This Neural Network has proven its success in many applications involving sequential or temporal data. For example, they have been widely used in speech recognition and machine translation. This model's success is mainly due to the gated network signals that control how the current input and the previous memory are used to update the current activation and generate the current state (Dey and Salem, 2017). These gates have their own weight sets, which are updated adaptively during the learning. Although these models enable successful learning in the RNN, they increase the parameterization through their gate networks. Consequently, they have an additional computational cost compared to a simple RNN model. Note that the LSTM RNN uses three different gate networks, while the GRU RNN reduces the number of gate networks. The number of networks is reduced to two. It is recommended to reduce the number of external gates to the lowest number possible, one, by preliminary evaluation of sustainable performance.

The GRU network is based on update, reset gate, current memory content and final memory. To make the equations uncluttered, I omitted biases. How the activation of the j^{th} hidden unit is computed is described on the basis of the idea by (Cho et al., 2014). First, the reset gate r_t

$$r_t = \sigma([W_r X_t]_j + [U_r h_{t-1}]_j) \tag{2.13}$$

where σ is the logistic sigmoid function, and $[\odot]_j$ denotes the j^{th} element of a vector. x and h_{t-1} are the input and the previous hidden state, respectively. W_r and U_r are weight matrices which are learned. Similarly, the update gate z_t is computed by

$$z_t = \sigma([W_z X_t]_j + [U_z h_{t-1}]_j) \tag{2.14}$$

The actual activation of the proposed unit h_t is then computed by

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'} \tag{2.15}$$

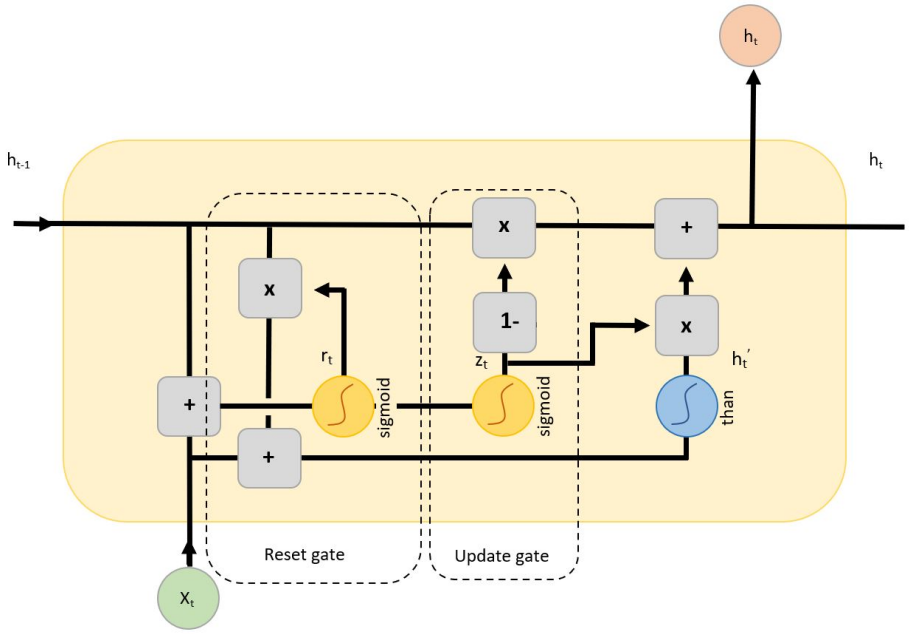


Figure 2.4: Illustration of the GRU layer, where x_t is the input and h_{t-1} are the output and the previous hidden state, *sigmoid* and *than* are activation functions

where

$$h_t^{'} = than([Wx_t]_j + [r \odot Uh_{t-1}]_j) \tag{2.16}$$

In this formulation, when the reset gate is close to 0, the hidden state is forced to ignore the previous hidden state and reset with the current input only. This effectively allows the hidden state to drop any information that is found to be irrelevant later on, thus, allowing a more compact representation. However, the update gate controls how much information from the previous hidden state will carry over to the current hidden state. This acts similarly to the memory cell in the LSTM network and helps the RNN to remember long-term information.(Cho et al., 2014)

Convolutions neural networks

Recent advances in neural architectures and their application to time series offer an end-to-end learning framework that is often more flexible than standard time series methods. Although there has been extensive work on the former, recent temporal models have been limited to sliding window action detectors, segmental and recurrent models. For many of these models, such as RNNs with LSTM or GRUs, the latent state at each time step, t, is only a function of the data at t and the hidden state and memory at $t-1$. Conversely, some new approaches, capable of considering the temporal dimension, have recently been tested, such as the Convolutional Neural Networks (CNNs).

Temporal 1D-CNNs (TempCNNs) where convolutions are applied in the temporal domain proved to be effective for handling the temporal dimension for time series classification. Pelletier et al. (2017) provided an exhaustive study of new deep learning approaches, namely Temporal Convolutional Neural Networks (TempCNNs) where convolutions were applied in the temporal dimension (see in 2.5).

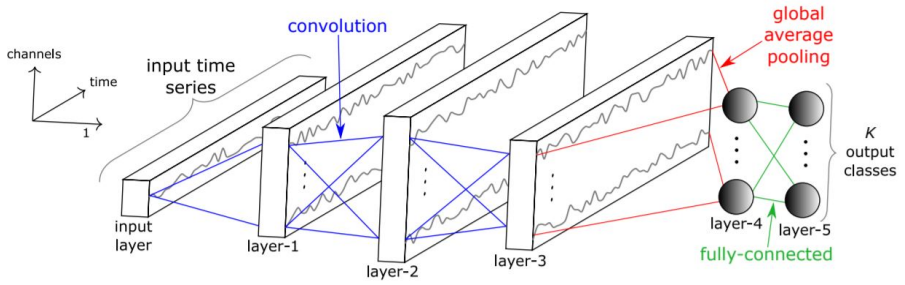


Figure 2.5: A unified deep learning framework for time series classification.(Fawaz et al., 2019)

Activation Function

It has become clear that the most important unit in Neural Network structure is their net inputs by using a scalar-to-scalar function called "the activation function or threshold function or transfer function", output a result value called the "unit's activation". An activation function is used for limiting the amplitude of the output of a neuron. Enabling in a limited range of functions is usually called squashing functions. It squashes the permissible amplitude range of the output signal to some finite value.(Karlik and Olgac, 2011) In the RNN, LST and GRU network, I had to use Uni-polar Sigmoid (σ), $Softmax$, Hyperbolic Tangent Function ($than$) and Rectified Linear Unit ($ReLU$) activation functions (see in 2.6).
The activation function of the Uni-polar sigmoid function is given as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.17}$$

The term sigmoid means 'S-shaped', so the logistic form of the sigmoid maps the interval $(-\infty, \infty)$ onto $(0, 1)$.
Hyperbolic Tangent Function is easily defined as the ratio between the hyperbolic sine and the cosine functions or expanded as the ratio of the half-difference and half-sum of two exponential functions in the points x and $-x$ as seen below (Karlik and Olgac, 2011). The Hyperbolic Tangent Function is similar to the sigmoid function, with its range outputs between -1 and 1.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.18}$$

ReLU stands for Rectified Linear Unit. Although bearing similarities to a linear function, *ReLU* has a derivative function and allows backpropagation, while also providing computational efficiency. The main idea behind prefunctions is that the ReLU function does not activate all neurons at once. Neurons are only deactivated when the output of the linear transformation is less than 0.

$$ReLU(x) = max(0, x) \tag{2.19}$$

The output of the sigmoid function falls within the range of 0 to 1, which can be regarded as the classification probability. However, the function faces problems when there are multiple output layers (e.g., multiclass classification). This problem is solved by the $Softmax$ function. The $Softmax$ function is defined as a combination of several sigmoids and computes the relative probabilities. Like the sigmoid/logistic activation function, the $Softmax$ function returns the probability of each class. It is most often used as an activation function for multiclass classification. It can be mathematically represented as follows:

$$Softmax(x) = \frac{\exp(x)}{\sum \exp(x)} \tag{2.20}$$

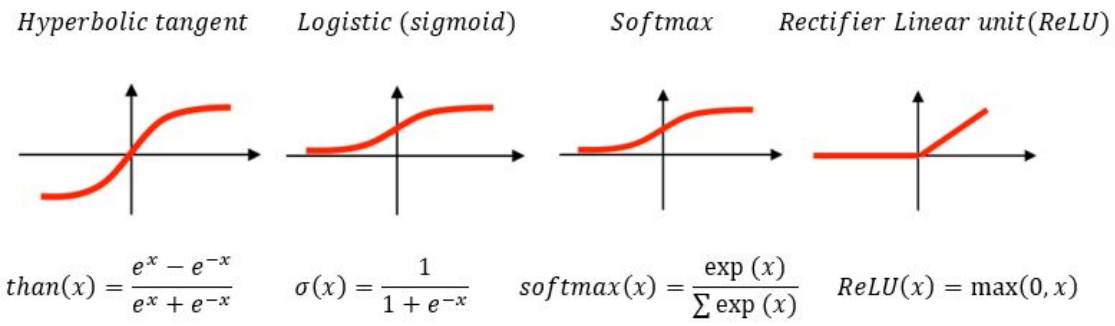


Figure 2.6: Illustration of activation functions, from left to right $than, \sigma, Softmax$ and $ReLU$

One-Hot-Encoding

Some Machine Learning algorithms can work directly with categorical data, such as a decision tree, but in most cases the input or output variables must be a number or numeric value. Since user log data is often composed of more than just numeric data, the need arises for some way to handle discrete variables in such cases. One commonly used method is the process of One-Hot-Encoding, which can map all categorical data into integers. With One-Hot-Encoding, each categorical value is converted into a new categorical column and the columns are assigned a binary value of 1 or 0. Each integer value is represented as a binary vector. This approach is a simple and efficient solution, but for many categories it produces sparse matrices which offer poor results for neural-relational

analysis.

Embedding Layer

In Deep Learning based approaches, the conventional way to extract information from past history is to feed raw or prepared sequences into the RNN or CNN. In the case of discrete valued time series, the categorical values must be transformed into the numeric space. Using a common encoding approach such as One-Hot-Encoding might not be very useful, as it explodes the dimensionality of the input feature vector and dramatically increases its sparsity.
There is another way to handle this problem, as in natural language processing, the categorical data could be transformed into a continual space with embedding. The embedding layer is mainly used in natural language processing applications such as language modelling, but can also be applied to other tasks involving Neural Networks. Thus one can train their own embeddings using the embedding layer. It is well-know that when dealing with textual data, it is necessary to transform it into numeric space before feeding it into any Machine Learning model, including Neural Networks. The same approach must be taken with discrete-valued sequences. The embedding layer maps the value of the discrete variable into a fixed-sized vector. The resulting vector is dense and contains real values instead of 0 and 1 as in One-hot-encoding model. The fixed length of the vectors of discrete sequences helps to better represent discrete labels with reduced dimensions. Thus, the embedding layer acts like a lookup table. The discrete labels are the keys in this table, while the dense discrete label vectors are the values.

2.6 Related work in the analysis of users' online behavior

Koehn et al. (2020) summarized the methods of event sequence data preprocessing, highlighting their advantages and disadvantages. One of the most often implemented methods is to create aggregated, cumulated data, which, however, results in data loss and requires manual feature engineering by the domain experts. Another conventional method is to create sequence segments or sliding a window, where only a chunk/fixed-length part of the data is used. Lastly, there are Neural Networks and embedding layers, where one can work with partially or completely raw data. Although classical Machine Learning methods have proven their value in many case studies, they are difficult to apply. One of the main obstacles is that they require a lot of data preprocessing and there is no general description of what counts as "good enough" cumulative data, so it is often a matter of luck whether a good feature is detected or not. Although Deep Learning based solutions can overcome these problems, but their use requires a higher level of knowledge. For example, even a classical Machine Learning method works with 2D data, while an RNN requires mapping the same in 3D or more dimensional space.
The usual time series prediction model works with numerical data such as stock prices, meteorology,

or sensor data. In essence, many publications use some implementation of recurrent (LSTM, GRU) or convolutional networks (CNN) to find patterns in processed or raw time series. In the last decades vanilla and combined solutions have proven successful in solving a given numerical sequence based problem. As an example, the work of (Yu et al., 2016) used Belief Networks and Bidirectional Long Short Term Memory to analyze sleep disorder data, while others combined Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN to solve multivariate time series classification problems. The list of research on numerical sequence processing may be quite lengthy, but when focusing on studies about discrete, categorical time series, it narrows down this list considerably. In my literature research I found only few studies that dealt with the analysis of categorical time series or discrete sequences. So far, most of field's research work has been carried out by biologists (in DNA research) and researchers in natural language processing.

Sequence predictors

The preprocessing of the time series data preprocess is rather challenging task. There are numerous gaps to handle, e.g., cleaning the dataset, creating and handle features with different types (time, numeric, categorical, etc.) or scale. In the case of data from a webshop or web service like education platforms, the sequences often contain discrete, categorical datasets. For instance, the log data of a video based educational site (MOOC) could often contain only video commands, i.e., *play, stop, pause* and *rewind* and the contents of the webshop basket could also be taken as a categorical data sequence. Thus it can be seen, the handling of the numerical order must be done in the same as the handling of categorical data. Dealing with the problem of categorical sequences does not seem difficult at first glance, as there are many methods for preprocessing the categorical data. For example, several studies successfully used the One-Hot-Encoding, Label encoding, and CountVectorizer techniques. Each method has its advantages and disadvantages: although these methods are easy to use, they require an immense amount of memory for long sequences with thousands of categories. Another disadvantage of these techniques is that the relationships between features and their temporality are lost. To solve these problems, many research works used the embedding layer in their model, since the embedding technique is able to deal with the problem of temporality and categorical discrete-valued variables. For instance, Ng (2017) used the popular SkipGramm embedding model to process DNA sequences and create the dna2vec approach. Asgari and Mofrad (2015) work involved embedding layers and introducing a new feature extraction method for protein-vectors (BioVec). Kimothi et al. (2016) also applied the same technique to create seq2vec for biological sequences. Koehn et al. (2020) proposed their impressive clickstream classification results where they applied RNN architectures and embedding layers.

Chapter 3

User behaviour analysis from high-level log data

One of the most commonly used areas of online user behaviour analysis is the webshop. In most cases, such a predictive model tries to predict some parameter (purchase size, next purchase time) on high-level data. That approach is based on some statistical or classical machine learning method. In this thesis, I will present a case study on sales promotion prediction which is based on the most common forecasting methods and their advantages and disadvantages.

3.1 Introduction

As stated in the previous chapter, understanding user behavior for log data is crucial for e-commerce and related services. High-level log data usually contains a great amount of information on the user's history with the webshop, e.g., purchase history, page visit dates, or wish list, which is useful data for purchase prediction, user clustering, or recommendation system. This chapter describes the task of high-level log data behavior analysis and presents a practical framework for real-life webshop sales promotion targeting. The use of recommendation systems has become a daily concept in product suggestion, product group selection, and promotional message content generation which is supported by Machine Learning techniques. Common examples of applications include the recommendation of movies (e.g., Netflix, Amazon Prime Video), music (e.g., Pandora), videos (e.g., YouTube), news content (e.g., Outbrain) or advertisement (e.g., (Sidana, 2018)). Section 3.2 highlights that recommendation systems are not directly applicable to users described by event history, like in the case of marketing letters and sales offer promotion prediction.
As the main contributions of this chapter, I will present techniques for the development of an ML based recommendation system. This type of system classifies users based on their event history

and makes sales promotion prediction on two levels. I solved the classification task of first level and the set of regression tasks of second level. I also exploited the clickstream high-level data (as described in 3.3) which provided an individual-level approach for sales promotion types. In this study, I worked with both high-level log data and static user profile attributes and therefore, proposed an efficient method to handle the problems of cold start, data sparsity and scalability. This chapter also introduces high-level data preprocessing methods and proposes cumulative features. I empirically compared several traditional Machine Learning models on these datasets. The last part of this chapter outlines their advantages and disadvantages.

3.2 Related works

3.2.1 Behavior prediction in e-commerce

Koehn et al. (2020) summarized the filed of user behavior prediction from log data in e-commerce, and divided the task into four groups, namely the 'predict the product group', 'classify a user log history', 'predict the outcome of an incomplete session', and 'click-through rate prediction' groups. In this chapter I focus on predicting the user's interest, based on observations of the user's purchase behavior during the shopping process. Hence this task belongs to the 'predict the product group' task of e-commerce user behavior prediction.

3.2.2 Behavior prediction methods

There are many classic data mining and Machine Learning methods published which deal with the problem of user behavior prediction in e-commerce (e.g., Bozanta and Kutlu, 2018; Cheng et al., 2016; Burke, 2002; Grbovic et al., 2015; Sidana, 2018; Çano and Morisio, 2017; Velingker and Alphonso, 2016). For instance, classification can predict the occurrence of an event (Adede, 2012), or regression techniques can aid in predicting the time or amount of money the user will spend on the website (Groves and Gini, 2011). The solutions that were more sophisticated and also more popular were, in fact, offered by collaborative filtering (Goldberg et al., 1992) or content based approaches (Van Meteren and Van Someren, 2000), as discussed in more detail in Chapter 2. Apart from the classical approach, many recent publications introduced combined solutions for this complex problem (Lucas et al., 2013; Chen et al., 2014; Tian et al., 2019), in which simple methods had to be combined and embedded to find a proper model. Bozanta and Kutlu (2018), for example, published a hybrid recommendation model that integrated user based and item based collaborative filtering, content based filtering together with contextual information to avoid the disadvantages of each approach. These combined methods proved their robustness and ability to solve many recommendation, or user behavior prediction problems. However, the majority of them used only a static snapshot of users and could not handle dynamic single- or multivariate user event

sequences, time series based data-sets. The event timeline in user history is vital in this database, thus applying a classic recommendation system was not an option.
The methods of event sequence data preprocessing were summarized in Chapter 2. One of the most often implemented methods use static user states or product information, which is a type of snapshot, though, working with only this small piece of data results in considerable information loss. Another common method is to create sequence segments or sliding a window, using only a fixed-length chunk of the data, but such action could also lose some important part from user history if the aim is to handle variable length user log-sequence. In this study I used high-level webshop log data, which consists of variable-length sequences. Since the goal was to avoid the loss of information, I used full sequences. For high-Level data, the most obvious solution was to create a cumulative feature-set capable of exploiting the whole time series.
Another approach is the Deep Learning based solution, but as several papers have shown (e.g., Chen and Guestrin, 2016; Osman et al., 2021), when the dataset size is limited and based on only short sequences (as in this high-Level log data-set), a traditional ML model can outperform a DLL based model. Moreover, because of the client's needs, I opted for an interpretable model, where the results were easily explained and visualized. Given these two requirements, I chose to work with classical Machine Learning methods and decided to build a combined model for this system that used both regression and classification methods.

3.2.3 Sales promotion prediction

The goal was to solve the problem of predicting the sales promotion from high-level log-lines. Martínez et al. (2020) and Liu et al. (2016) published the results that were probably most similar to those obtained in my work. They developed models that could predict future customer behavior which was based on the set of customer-relevant features that were derived from the times and values of previous purchases. Similar to the solution presented in this chapter, they applied Machine Learning algorithms including Logistic Lasso Regression, the Extreme Learning Machine and Gradient Tree Boosting for predicting whether the customer would make a purchase in the upcoming month. Although these two cited papers were quite similar to the solution implemented here, there were also differences: Unlike the afore-mentioned works, the here-presented prediction algorithm used a combination of methods instead of a single one. Since each unique problem required a unique feature set, much effort was invested in building a new cumulative dataset to meet our needs, in addition to building the model.

3.3 Problem Statement

This chapter focuses on ways to solve the problem of predicting the purchase behaviors of users who have a known history on an e-commerce website. More specifically, the aim was to forecast which ads

Table 3.1: Illustration: problem statement as a binary classification.

1^{st} purchase 2^{nd} purchase 3^{rt} purchase ..... n^{th} purchase		
time of prediction	->	Likely to buy with sales promotion (user who use more than 50% promotion for buying something)

Table 3.2: Illustration: problem statement as a recursion; the distribution of sales promotion types

1^{st} purchase	2^{nd} purchase	3^{rd} purchase	.....	n^{th} purchase	
		Time of prediction	->	SPType1 SPType2 ..... SPType n	35% 25% 50%

group or form of sales promotion the user will most likely use based on his or her purchase history and profile information. This form of sales promotion could include buy two, get one free; price deal; sampling, etc. While I did not directly use the work of others to design this system, the solution I arrived at did, in fact, show considerable similarity to the description of (Zhang and Pennacchiotti, 2013). In other words, the predictive system would help in several practical scenarios, including:

- building a cold start recommender system, by providing high-level recommendations to users who visit an e-commerce website for the first time;
- improve existing product recommendation engines by providing category-level priors that can guide the recommender system to, and domains of interest for the user;
- provide e-commerce companies with tools for targeted email/social media campaigns.

This implementation of the algorithm had two main goals. The first one was to explore which piece of information was correlated with the form of sales promotion which the users were most likely to opt for (see 3.1 for an illustrative example.) Based on this a combined model was built and tested, which optimized a user-level table, in order to propose the form of sales promotion to users that best fit their interests and preferences, (as presented in 3.2). The second goal was to back-test and thoroughly document each critical point of combined Machine Learning algorithms that could be used as a base structure for those who aimed to replicate this model or build a similar system.

3.4 Dataset

The data used in this work was recorded from a health and beauty webshop. The data contained more than one million users, from different markets (countries), however, in order to obtain the

richest data possible, it was filtered by the oldest market which included 230.000 user-profiles and their purchase history. The data consisted of seven years' of user interaction logs with the webshop. Each event had a user identifier, timestamp, and an event type. The purchase data contained five categories of events: pageview of a product, basket view, buy, ordered timestamp, and delivered timestamp. There were approximately 240 different types of products. In the case of a buy or a basket view, there was information about the price and extra details. An average customer visited the shop two or three times a year, which led to a very sparse and high dimensional dataset. This was not surprising since this is a common occurrence in recommender systems (Sidana, 2018). As a solution, there were two obvious ways to reduce the dimensionality of the data: either by marginalizing the time (aggregate pageviews per user over the period) or by marginalizing the product pageviews (aggregate products viewed per time frame) (Vieira, 2015). I explored both approaches in this work. As a first step, the solution presented in this chapter connected unique events with sessions. I used homogeneous data such as purchase history only and heterogeneous example clicks, profile data in nature. These events were then cleaned and ordered by their timestamps to form the action chain. As a next step, the unique events were transformed into a feature list (e.g., number of purchases, the distance between two logins, etc.). Besides the evident data (number of, sum of, mean of purchases), the script accumulated other data such as:

- distance (in time) between first and second, third,etc. actions;
- number of purchases in first, second, etc. months;
- increase or decrease in purchases compared to the previous month by month;
- the reaction times between advertising letters and a purchase.

Feature engineering

One of the crucial steps for better performance of a classifier was to preprocess the data correctly. Apart from the regular data cleaning process, the features were transformed by scaling each feature to a given range with min-max scaling. As a last preprocessing step, feature importance was calculated with a tree based ensemble method, namely the ExtraTreesClassifier method (Geurts et al., 2006). Based on the obtained results, the model used only the top 20 features, which significantly increased the accuracy of the results.

3.5 Methodology

In order to handle the popularity-bias, the problem was divided into two subtasks:

- predict if a user is sensitive for the sales promotion or not, and

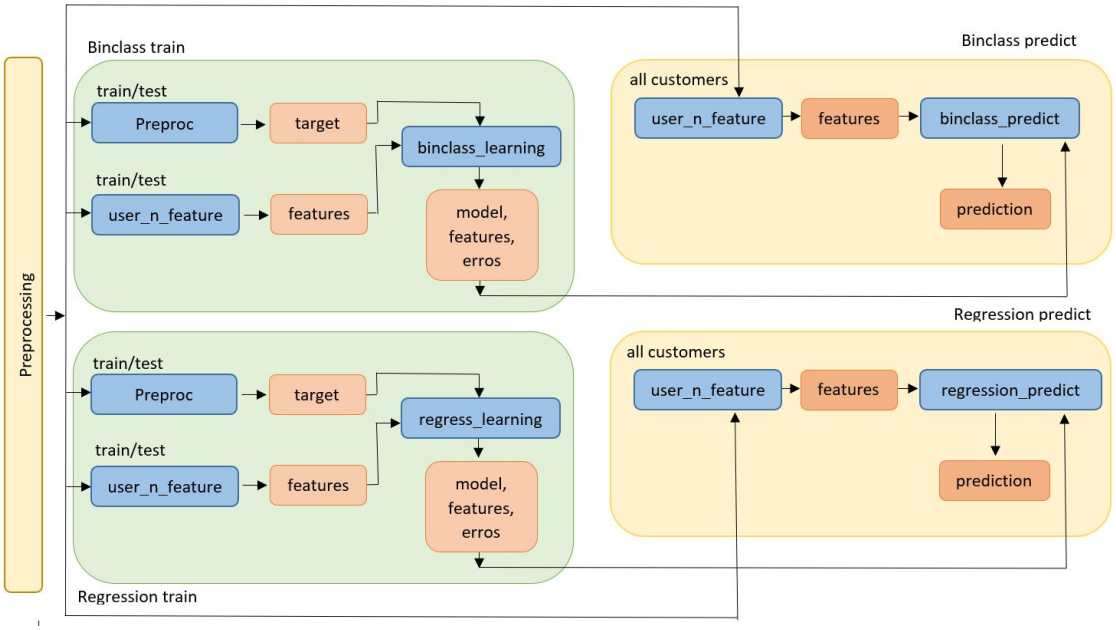


Figure 3.1: State diagram of our combined solution

- predict which kind of form of sales promotion is more interested in it.

As a solution, a combined model was created which used both the regression and classification methods, see Figure 3.1. The recommendation model returned two lists. The first list gave information about the users, whether or not they were likely to use any of the sales forms (the sensitivity for sales promotion). The second list provided the data to calculate the probability for every sale (which form of sales the user was likely to opt for). For the results, I proposed a novel combined recommendation algorithm where similarity measurement is performed between a user the form of sales based on features derived from the user's profile and history information. Table 3.3 presents a table where every single user is designated their specific predicted value.

3.6 Experimental setup

When using raw log data to make a prediction for recommendation, one must handle the data sparsity problem. As already mentioned, this dataset contained 230.000 pieces of data. However, only 33.000 of the entries had data of sufficient quality. Thus, in the research, only this reduced and filtered dataset was used, where the entire dataset was split into test sets (20%) and training sets (80%). The first step was to train various classification models, including Logistic Regression (Darroch and Ratcliff, 1972), Random Forest (Breiman, 1999), LightGBM (Ke et al., 2017), and XGBoost (Chen and Guestrin, 2016), where grid search was used to select the optimal parameters. As proven by

Table 3.3: Example of model outcome

user ID	likely to use sales promotions	likely to use sales promotion type				
		type1	type2	type3	type4	type5
1000	YES	35%	50%	5%	6%	7%
1001	NO	0%	0%	0%	0%	0%

the final results, the XGBoost classifier and XGBRegressor performed the best. Additionally, the majority of classifier (MC) (James, 1998) was used as a baseline for comparison with the above learning algorithms. For the regression problem, the central tendency measure was used as the baseline for all predictions. Based on these, I examined the combined models using the training set and adjusted the parameters of the predictive algorithms' achieving the best performance on the validation set. A prediction was made for each instance in the test set and the forecast results were compared with the true values by computing corresponding performance metrics. To obtain the best evaluations I applied the K-fold validation, where both training and validation sets were used for prediction.

Handling the problem with an ensemble classification and regression tree

The first goal was to predict if a user was likely or not likely to use a sales promotion, which was a binary classification problem. In order to find the best solution, I trained and tested classification models a great many times. Following extensive research, it was established that the XGBoost ensemble classifier (Chen and Guestrin, 2016) provided the best results. This was not entirely surprising, because tree boosting is a highly effective and widely used Machine Learning method. Another important feature was that the algorithm performed well, as it included an efficient linear model solver and could also exploit parallel computing capabilities (Chen and Guestrin, 2016). Ensemble learning provided a systematic solution to merge the power of multiple learners. The prediction value of XGBoost can have different interpretations depending on the task, i.e., regression or classification. XGBoost is a tree ensemble model set of classification and regression trees. It was able classify the data used in this research into one of a finite number of values, that was why it was called a regression model (nonlinear model). Apart from XGBoost, the results were also compared with Linear regression (Cook, 1977), Lasso regression (Park and Casella, 2008) and Ridge regression (Hoerl and Kennard, 1970).

3.7 Results

3.7.1 Classification

The greatest challenge of the recommendation system is usually the cold start problem. It may appear when the user starts the initial steps, or as in the present case, when the shop owner starts a

Table 3.4: Results of classifications

Model	ACC	F1	precision	recall
Baseline	0.587	0.342	0.351	0.337
Logreg all	0.676	0.409	0.620	0.306
Logreg top10	0.685	0.404	0.661	0.291
XGBoost all	0.706	0.527	0.652	0.436
XGBoost top10	0.703	0.518	0.657	0.419
XGBoost all HPT	0.768	0.519	0.666	0.423
XGBoost top10 HPT	0.771	0.509	0.658	0.417
XGBoost top10 HPT(4)	0.790	0.624	0.713	0.554

new sales promotion type, which makes very sparse data. To solve this problem, I filtered (dropped) those particular users and promotions from the training dataset which had too sparse data or no data at all. Based on my model, I created a binary classification with XGBoost to predict whether or not a user was likely to use a given sales promotion. The parameters of the estimator were used to apply optimization by cross-validated grid-search over a parameter grid. Several models and settings were tested before the most accurate model was identified.

These results are displayed in Table 4, where the window size (number of purchases) was three for all the methods, except in the last configuration. During the first phase, XGboost was used with all features, more specifically, with only the top-10 features, which achieved a 70% rate of accuracy. To improve this, I applied hyperparameter tuning (HPT), namely cross-validated grid search over a parameter grid which would provide greater accuracy. The aim was to make further improvements, but the sparsity of the data did not allow for it. The challenge was to predict user feature habits in as short a time as possible. For that reason, I used the user's first three instances of purchase history to train the model, but this failed to improve the results, which was, however, to be expected. Simply put, it would take more data to gain better results. This could only be achieved by waiting for more information, or by prompting clients to fill in the profile table. To prove this concept, I trained the model with the user's first four purchases, which achieved 0.79 accuracy (as highlighted in the last row of Table 3.4). Surmounting this obstacle called for a different approach, which had actually been suggested by many researchers: if the classification model is not accurate enough, one should resort to changing the point of view. In line with this suggestions, I retested this solution as a regression with XGBRegression (as a regression problem). This provided the result RMSE = 16.77, which was, indeed, not a better outcome, because if this result were transformed into a classification result, it would still be accuracy 0.686, precision: 0.578, recall: 0.546, and F1: 0.562.

3.7.2 Regression

In the second phase, the goal was to determine which type of sales promotion (SP) most users would prefer (see Figure 3.1). This was a regression problem, in which every SP type had to be predicted

Table 3.5: This table presents ways to determine which type of sales promotion most of the users would prefer. The table summarizes the results of the sales promotion regression problem. The different settings demonstrate the LGBMReg CV TOP offer the best results.

model	Sales promotion Type1			Sales promotion Type2			Sales promotion Type1		
	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE
Baseline CV	5.840	53.568	7.313	9.970	161.948	12.723	12.679	256.261	16.001
DNN	5.906	53.275	7.298	9.870	158.492	12.589	12.600	259.132	16.097
LR all CV	5.039	46.029	6.779	8.927	131.045	11.442	11.368	206.408	14.3651
LGBMReg CV	4.7152	42.469	6.551	8.446	118.202	10.869	10.946	191.677	13.843
StackReg CV TOP	4.778	43.153	6.564	8.720	125.506	11.200	11.092	196.392	14.013
LR CV TOP	4.986	44.844	6.691	8.829	127.842	11.301	11.234	203.471	14.284
LGBMReg CV TOP	4.700	42.349	6.501	8.602	112.589	11.067	10.895	191.12	13.824

for every user. Instead of testing all the types of SP for a measurable result, I only selected three types of promotions:

- Type1 was an SP type which had a long history in this webshop;
- Type2 only had a one-year background, and
- Type3 was the most recent SP type, with less than six months of use.

Based on the above-described, the results were obtained as reported in Table 3.5. To ensure the best outcome, I ran more models with different settings, including linear regression (LR), LightGBM (LGBMReg), and a simple Deep Neural Network (DNN). In the initial step, the model used all (n = 129) normalized, scaled, and skewed feature sets. Based on this method, the LGMBReg made the most accurate solution. The second step was to increase the model's accuracy, which required finding the most important features. For this purpose, I implemented the wrapper method, specifically, backward elimination. As the name suggests, all the possible data were fed into the model at first. Then the performance of the model was tracked and the worst performing features were repetitively removed one by one, until the overall performance of the model came in a suitable range. To calculate feature importance, I used the ordinary least squares (OLS) model (Weiss, 1988). After numerous attempts and settings, it was found that the best solution was provided by LGBMReg, a tree based regression model, which created a much more accurate model than the random choice.

3.7.3 Discussion

The problem and solution of predicting the acceptance of the sales promotion were highly specific since the task was not to predict the next purchase but to gauge a user reaction to advertising letters. Regardless, I had to identify a way to compare the performance of this model with other ones. The methodology and results of this experiment were similar to the results obtained by Martínez et al. (2020), so I selected their results as a basis for comparison. The problem of predicting whether a

user was likely or not to use a sales promotion was essentially identical to their binary classification problem. While my model achieved quiet a good level of accuracy, their solution gave an almost perfect solution. The difference between these two models was not surprising, given the fact that I used only the first four purchases and high-level sequences, whereas they used 24 months' worth of data with a middle and low-level time series for the same task. The comparison results confirmed Martínez et al.'s observation that it was difficult to make an accurate prediction model from short data and few purchases, however, over time, as data was collected, more accurate results could be gleaned from the data.

3.8 Contributions

This thesis has presented the following contributions. The construction of the model and its results were presented in the journal paper: (Kőrösi and Vinkó, 2021).

- I used the log data of an existing webshop in Hungary to develop a solution that can reliably predict the sales promotion acceptance probability from the high-level user log data. I proposed a specific feature representation that allowed me to generate cumulative data from the obtained user sequences that effectively supported the operation of the model, which was designed in the form of a combined classification and regression solution.

- In the combined model the classification method was used to determine whether or not a user would accept the sales promotion. Using the output of this classification model with a regression task, I separately predicted the probability of promotional package acceptance. The output of this combined model was not only able to predict user behavior with relatively efficiently, but also provided a solution that was easy for the client to interpret.

- I made empirical measurements with almost a dozen different Machine Learning methods, and run hyper-parameter tuning to find the optimal solution. I demonstrated that when using high-level log data, the cumulative feature extraction method with a combined classification and regression solution was able to provide fast and efficient results, which was confirmed by the customer's satisfaction.

Chapter 4

Educational performance prediction from middle-level click-stream data

Researchers in the field of online education have been working on predicting student behavior for almost a decade, but so far they have only been able to do so on poor data sets. Today, we live in the age of online education platforms, which have given us access to greater amounts of data. The availability of new middle- and low-level data has opened up new opportunities for researchers, allowing the use classical Machine Learning methods in more accurate ways. In this thesis, I present the middle-level data collection techniques and prediction methods of an online reasoning platform.

4.1 Introduction

Mushrooming as a scalable lifelong learning paradigm, Massive Open Online Courses (MOOCs) have enjoyed considerable limelight in recent years, both in industry and academia (Haggard, 2013). With the appearance of MOOCs, educational platforms gained additional boosting, a new aspect in their evolutionary process, which opened a new field of research thanks to the extraction of user log behaviour information within the frames of data mining. Despite their early promise, however, MOOCs are still relatively unexplored and poorly understood (A. Anderson and Leskovec, 2014). Meanwhile, MOOCs often attract an enormous number of registrants, but only a fraction of them are able to successfully complete their courses. High drop-off rates are often attributed to factors such as low teacher-to-student ratios, the asynchronous nature of interaction, and heterogeneous educational backgrounds and motivations, which make it difficult to scale the efficacy of traditional teaching methods with the size of the student body (X. Wang and Rose, 2015; Yang et al., 2017).
The time series structure and analysis of e-commerce and MOOC platforms shows great similarity. However, during the analysis of the high-level webshop log data used in the previous step of this

research, detailed in Chapter 3, it transpired that the short and high-level log data limit the accuracy of the model. Thus, in order to build models with higher accuracy, it was first necessary to change the depth of the data. For the purpose of obtaining better and deeper data, I designed an e-learning course on a Moodle site, called Conscious and Safe Internet Usage, or in Hungarian, Tudatos és biztonságos internethasználat alapjai TÉBIA. The course was attended by primary and secondary school pupils, as well as university students. The term 'learners' is used to denote both pupils and students, I also added a middle-level user behaviour logging component in order to collect the users' online activities. This chapter describes the analysis of the log- data of learners who were motivated by their teacher and their school to attend and complete the short MOOC course, lasting only a few days. The logged data is similar to the data from edX and Coursera, even though it was created in a much shorter time period than those, and it is of a school-class nature. This log file ensured an opportunity to study clickstream data and user attitudes in short MOOCs. This chapter presents the following:

- the structures of the self developed middle-level user behaviour logging component;
- the data collection and preparation methodologies: The logging system during the two courses registered 4.663.120 logs, out of which 26 variables were generated and assigned to the user;
- the structure of the curriculum;
- feature design and selection methods: a feature space of 263 attributes was proposed to describe learners' clickstream data, while various feature selection and various classification approaches are applied;
- comparison of model results.

The main contributions of this investigation are that deeper middle-level data is able to support more accurate model even if using short MOOC courses. Those features which influence the classifier results the most are highlighted, thus providing useful insights for MOOC developers.

4.2 Related works

There is significantly less research specifically using Machine Learning models to support e-learning systems than works focusing implementing those in the area of e-commerce. Brinton and his research group published various studies (C. G. Brinton and Ju, 2015; C. G. Brinton and Poor, 2015; Brinton and Chiang, 2015; C. G. Brinton, 2016) aimed at predicting student behavioural classification from log data of MOOC systems. They stated that the most common way for time series prediction was to use the recurrent Neural Networks. However, they specified that if the sequence was short, discrete-valued and variable- length, the RNN based solution was not the best option. To solve

this problem, they opted for preprocessing the cumulative middle-level log data based features and feed them into a Multi-layer Perceptron instead of the RNN based method. Given that the present research worked with a similar, middle-level dataset, my solution also used cumulative data, which I did using various classical Machine Learning methods.
The dataset captured various user events, but the most data instances on interactions were captured during video viewing. The work of (C. G. Brinton, 2016) underlined the value of video interactions in student performance prediction, thus I also chose to focus on the video-interaction logs. In this reference work Brinton studied the behaviour during video viewing and the success rate of quizzes embedded in video.
For this purpose, he searched for so-called motifs in preprocessed and denoised cumulative data. The resulting motifs and a Support Vector Machine were used to predict the outcome of video-in-quizzes. In another study, M. Speiser (2012) also tried to build a predictive model on video viewing data, which served to support the aim of my research. Brinton explained why it is not worth using raw log data for short time series with few data points, which confirmed my notion that for a middle- and low-level data set with variable and short time series, it is preferable to carry out thorough preprocessing and define cumulative features and only then use classical Machine Learning methods to predict the success or failure of the course.

4.3 E-learning logger module layer

As part of a team, I participated in the development of a middle-level user behaviour logging component, presented in Kőrösi and Havasi (2017). It was built on the basis of Moodle which is an open-source, free, well-supported, popular e-learning platform. The Moodle platform is known for its robustness, though its user interface is less modern than it would be expected these days. This was the underlying reason for completely replacing its font-end and developing a new one, which is called Moodle's back-end. One module of its front-end is responsible for logging and this logger front-end collects and processes events, and calls its back-end part via HTTP call to store them. The back-end part – which is completely independent from the back-end of the Moodle - is developed in NodeJS and uses MongoDB to store events (Figure 4.1). Every log entry is an event. Each of the events classified the log data into different types, and depending on the type, they store different parameters for it. For example, a "textinput" event has a parameter, which stores the typed text, called text:

```
{
    type: "textinput",
    data: {
        target: "search-target",
        text: "database"
```

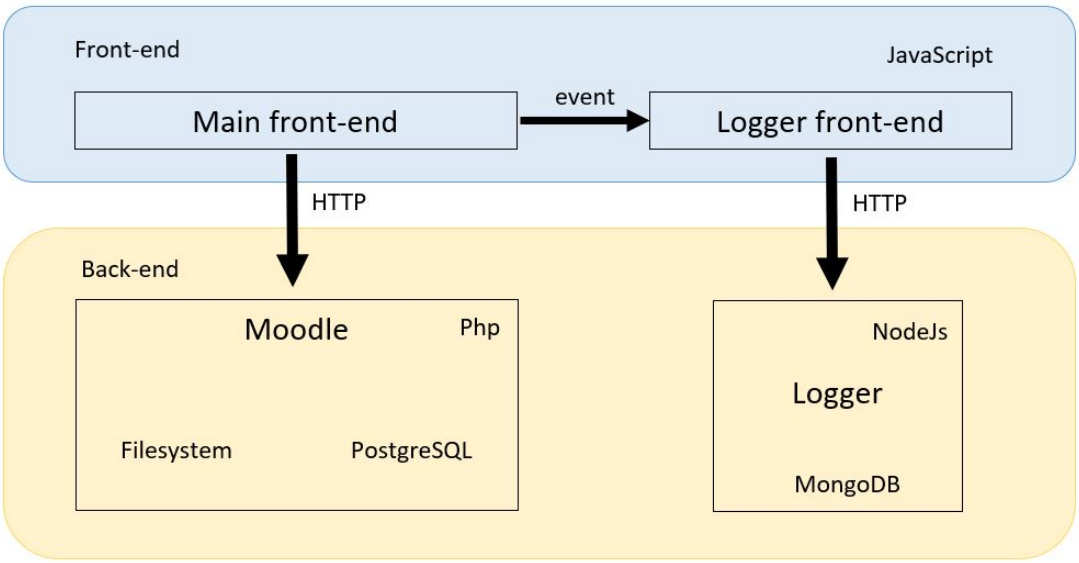


Figure 4.1: Front-end logging system

```
        },
    time: "2017.01.23. 16:01:28.242",
    page: "https://...",
    userid: 1876
}
```

There are some parameters, which are stored for all events:

- userid: the ID of the user, who executed the operation, or 0, if there was an anonymous user
- time: the date of the event
- page: the URL of the page, where the event happened
- type: the type of the event (see below)

The type of the events, and their parameters can be as follows:

- load, unload, focus, blur: generated in the case of loading or unloading of the page and achieving and losing the focus.
- resize: meaning to resize the browser window. It has two parameters: x, y (the new size of the windows).
- click: it represents a mouse click. Its parameters are x and y.

- testClick: it signifies a mouse click to an answer of a quiz. It is a preprocessed event: this javascript event handler automatically recognizes whether the mouse click happened over an answer and generates a testClick event, not a simple click event. Its parameters are: question, answer, correct, choiceCleared.

- download: generated in the case of downloading a file. Parameter: filename.

- textinput: this event represents a change of a text input field. Parameters: target (ID of the text element), text (actual value of the text element).

- textinput_focus, textinput_blur: they are generated when a text input gains or loses the focus. Parameters: target (ID of the text element), text (actual value of the text element).

- passwordinput, passwordinput_focus, passwordinput_blur: similar to the previous ones, but because of security consideration, the value of the password text input is not stored. Parameters: target (ID of the text element), and length (of the text element).

- mouse move: mouse moving event. Parameters: x, y, xDistance, yDistance, realDistance. The system stores only two mouse events in a second.

- scroll: means scrolling the page. Parameter: top. The system stores only two mouse events in a second.

There are video events, as well. The system supports two kinds of video: html5 video element and embedded YouTube video. Events:

- videoSeek: means seeking in the html5 video element. Parameters: seekTime, videoId, totalTime, src.

- videoPlay, videoPause, fullscreenOn, fullscreenOff: html5 video playing events. Parameters: actualTime, videoId, totalTime, src.

- volumeChange: html5 video element volume change. Parameters: actualTime, videoId,totalTime, src, newVolume.

- youtubePlay, youtubeEnd, youtubePause, youtubeBuffering: youtube video playing events. Parameters: actualTime, videoId, totalTime, src.

- youtubeQuality. changing youtube video quality settings. Parameters actualTime, videoId, totalTime, src, quality

- youtubeRate. parameters: actualTime, videoId, totalTime, src, rate.

4.4 Collected dataset

The courses were created to test the logging platform and collect user behaviour data. In the first part of this research, a pilot study was conducted between the dates of March 1 and May 30, 2016, which was followed up by the second study, carried out between October 1 and December 10, 2016. Altogether 163 learners participated in the initial study and 1370 student signed up for the second course in the autumn semester. The course details are summarized in Table 4.1 below. The learning material for the course comprised a six-week study period. The course, in fact, courses, as the same course was offered several times, ran under the name 'TÉBIA,' and included 4 + 1 (embedded) videos with attached embedded texts, or external links. The primary point of interest lay not only in the dropout rate, but in discerning how the platform functioned and how the learners would behave. Eventually, 99.8% of the learners who had signed up for the course, had also completed it. The logging system during the pilot and the second course registered 4.663.120 logs, out of which 26 variables were generated and assigned to the users. These were the following: Data, Page, Pid, Time, Type, User, Data.realDistance, Data.x, Data.xDistance, Data.y, Data.yDistance, Data.Text, Data.Top, Data.Target, Data.Filename, Data.Length, Data.ActualTime, Data.Scr, Data.TotalTime, Data.VideoId, Data.SeekTime, Data.NewVolume, Data.Ip Adress, Data.Quality, IP.
In the course of recording, middle-level time series data was obtained through the developed a middle-level user behaviour logging component in the form of student IDs, time stamps, and activities. The samples of data are constructed from the TÉBIA course which involved upper grade learners form 20 elementary schools. Components of a previously used and tested curriculum were taken as the basis of the course content, which included an initial test with a video lesson and three further units. Every unit consisted of an obligatory video task and further optional textual learning materials. To complete a unit, the learners had to solve three tests with a minimum score of 5 points out of 10. Every unit culminated in a test with a maximum score of 10 points, except for the initial test, which carried a max of 15 points. The distribution of the learners' final scores showed a Gaussian distribution (Figure 4.2) which confirmed the validity of the outcome test.

The distribution of students' final scores shows a Gaussian distribution (Figure 4.2) which supports the validity of the outcome test.

4.4.1 Data cleaning

The types of activities recorded are those which correspond to broad categories of student behaviour, such as previewing lectures, mouse behaviours (move, scroll, click), video watching attitudes and text inputs. As mentioned before, the course analysed in this chapter had 1370 registered learners who generated 4.663.120 click events over a 6-week period. The portal recorded 1370 learners and lecturers, out of which only 1077 filled in and completed the initial test (Q0). As Chapter 4.4 shows, the noisy and complex nature of this set of data made it impossible to use simple statistical

Table 4.1: Course contents

Course name	TÉBIA
Content	Basics of Conscious and Safe Internet Usage
Time frame	6 weeks
Parts of the Learning Material	Introduction:Video (3.37 min., Embed);
	Digital footprint: Video (14.04 min, Embed); HTML embedded text;
	Conscious and Safe Internet Usage: Video (13.07 min, Embed); HTML embedded text; External link;
	Online bullying: Video(13.31 min, Embed); HTML embedded text; Extra video (11.55 min, Embed);

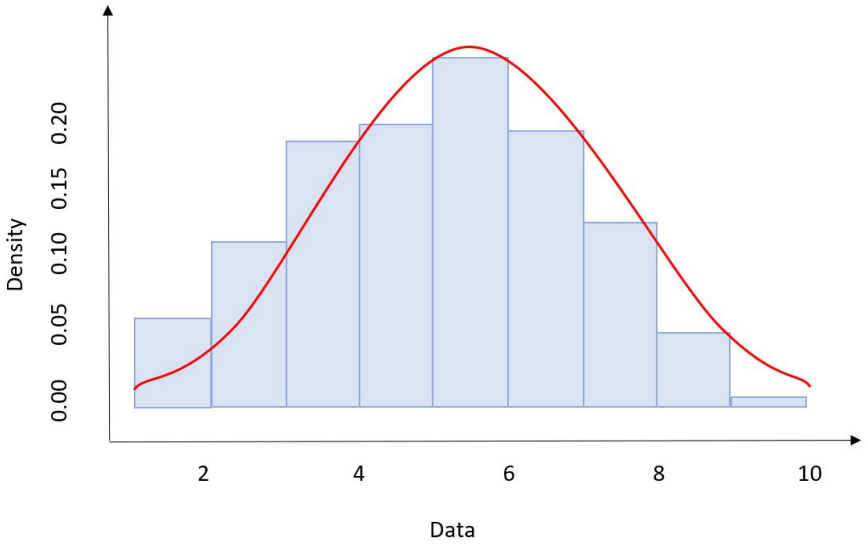


Figure 4.2: Number of clicks by user

or clustering methods to create a predictive model. Those learners, who had an output test but had an insufficient number of activities, were eliminated from the measurement. The number of obtained results amounted to 603. Based on the conditions set to complete the course, the group was split into two parts (Q1>=5 and Q2>=5 and, Q3>=5), which were labelled as 0 ("Failed") and 1 ("Completed").

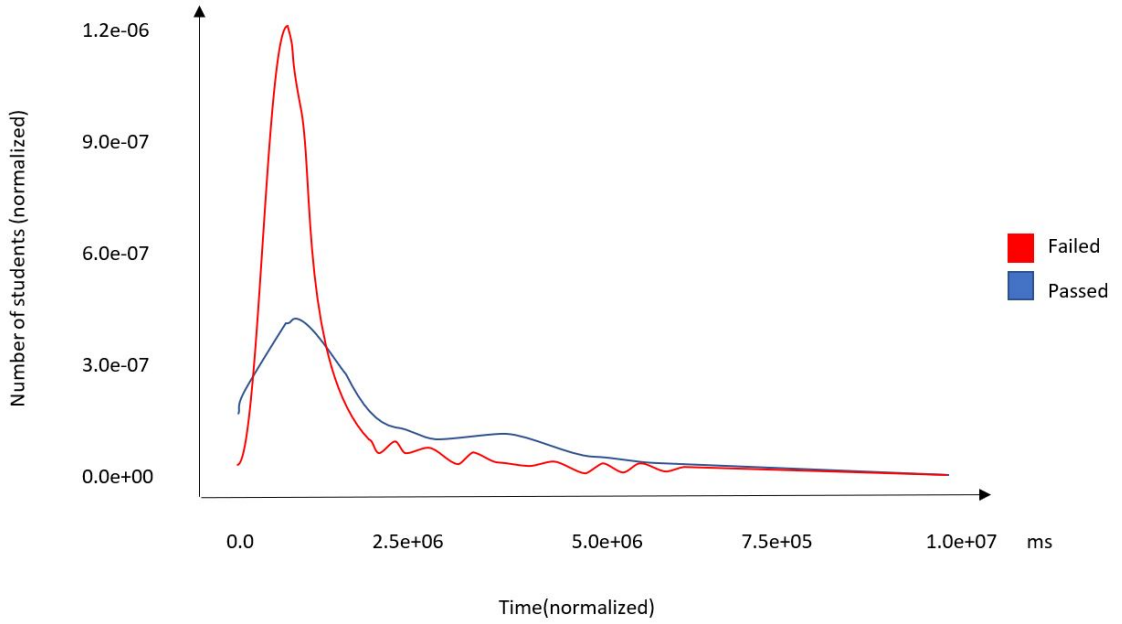


Figure 4.3: Time spent in course

Table 4.2: Time spent in course

Failed

Min.	1^{st} Qu.	Median	Mean	3^{rd} Qu.	Max.	NA's
51000	684300	861900	1294000	1291000	9725000	3

Completed

Min.	1^{st} Qu.	Median	Mean	3^{rd} Qu.	Max.	NA's
123500	123500	1098000	1987000	2970000	9462000	3

4.4.2 Preliminary investigation

The class- label-wise distributions of several log properties were visualized so that the structure of the data can be investigated, and user behaviour better understood. Due to the unbalanced nature of the data ($n_{failed} = 419$, $n_{completed} = 184$), density distribution is presented. The following density figures show the differences between the main attributes of the learners who failed and completed the course. The final diagrams (Figure 4.5) show a significant overlap between the two groups, which makes it harder to adjust the weighting settings. While the list of differences between these two groups is far from complete, the following illustrations sum up the most significant distinctions, as given in Figure 4.3, Figure 4.4, and Figure 4.5 as well as Tables 4.2, 4.3, and 4.4. All the diagrams below were constructed using the Ggpolt R package of Wickham, et al. (Wickham et al., 2016)

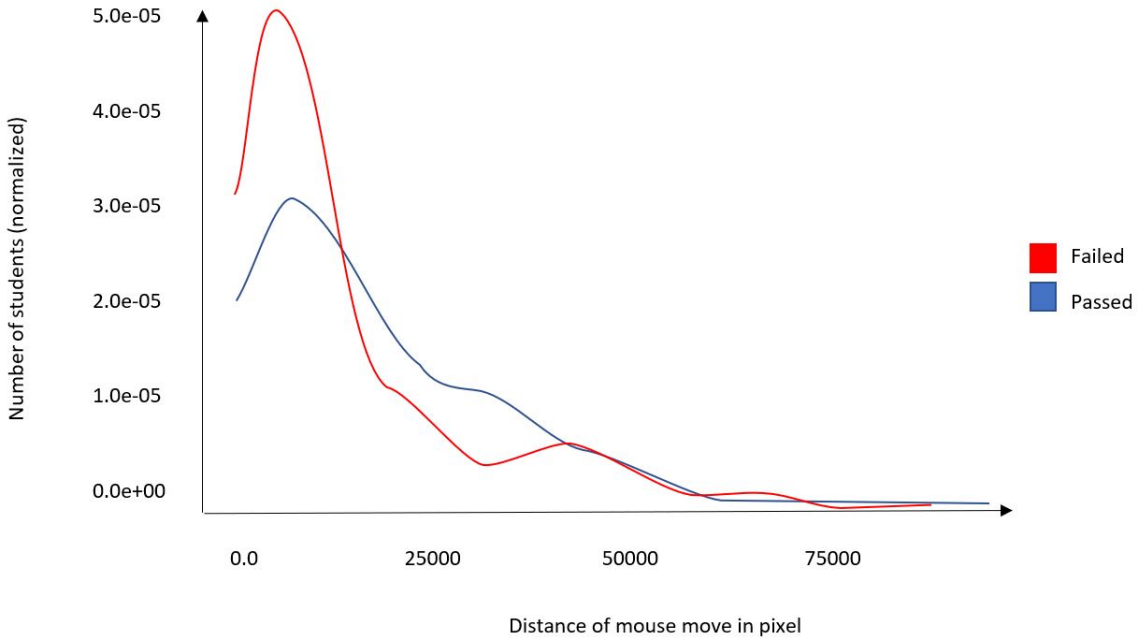


Figure 4.4: Average distance of mouse in course contents

Table 4.3: Average distance of mouse in course contents

Failed						
Min.	1^{st} Qu.	Median	Mean	3^{rd} Qu.	Max.	NA's
226	4325	8190	14800	20570	82210	262
Completed						
Min.	1^{st} Qu.	Median	Mean	3^{rd} Qu.	Max.	NA's
541	5632	12990	19040	28520	95030	76

4.5 Machine learning experiments

I conducted Machine Learning experiments using clickstream log data to predict whether a particular student would pass or fail the final exam of the MOOC. I employed the Rminer (Cortez et al., 2009) package of R.

4.5.1 Feature space

As stated above (section 4.3), a total of 263 features were defined to describe the clickstream data. There were two types of data. In the first group was the data which was collected during the filling process in the incoming test. The second type of data was the clickstream which was collected during the learning process in the three parts of the curriculum (see Table 4.1). This collection was divided

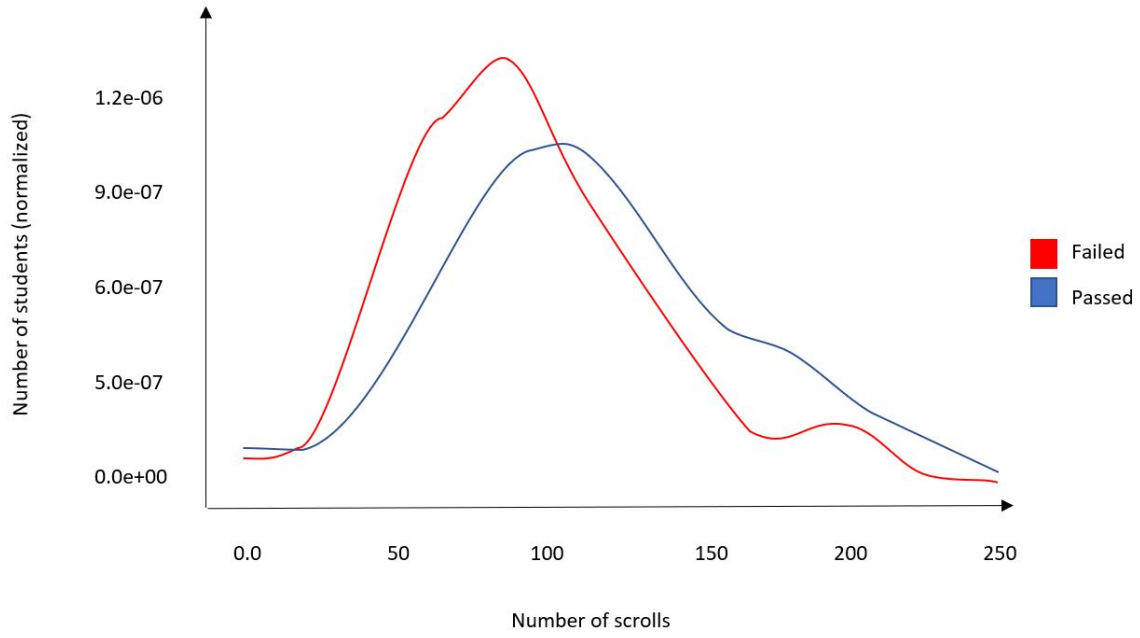


Figure 4.5: Average number of scrolls in course contents

Table 4.4: Average number of scrolls in course contents

Failed

Min.	1^{st} Qu.	Median	Mean	3^{rd} Qu.	Max.	NA's
0	64	89.50	95.58	118.80	244.00	9

Completed

Min.	1^{st} Qu.	Median	Mean	3^{rd} Qu.	Max.	NA's
0	81	109.00	115.00	144.00	249.00	9

into 18 major categories:

- binary and numeral answers provided to input and output tests (28+60).
- time spent on the quizzes (6).
- and the sites of the curriculum (7).
- the number of visits to the site of quizzes (6) and site of the curriculum (7).
- the mouse movement distance in pixels (13).
- the average mouse speed (13).
- the cumulated data (6).

- the number of mouse movements on a page (13).
- the number of clicks on a page (13).
- the use of test buttons during the input/output testing (4).
- the number of scrolls on a page (13).
- the last login date to a page compared to the first login to the site (13).
- the first login date compared to the first login to the site (13).
- the days spent on the sites (13).
- the mean behaviour on the sites (19).
- the number of calendar days between the output tests (7).
- binary output results (1).
- output results (2).
- user-related data (6).

4.5.2 Feature selection

Various feature selection methods were investigated and the gain-ratio function in the FSelector package (P. Romanski, 2017) proved to be the most effective. Gain-ratio examines all the parameters one-by-one and creates a hierarchy, which distinguishes weak and strong correlational connections. The FSelector package was designed to handle such problems and the most useful functions of all were the chi.square and gain.ratio filtering algorithms. Out of these two, the latter provided accurate calculations so the choice highlighted the underlying theory. The information gain method selected a split based on which attribute provides the greatest information gain. The gain was measured in bits. Although this method provided satisfactory results, it favoured splitting on variables that have many attributes. The information gain ratio method incorporated the value of a split to determine what proportion of the information gain was valuable for that split. The split with the greatest information gain ratio was chosen. (Brinton and Chiang, 2015)

4.5.3 Prediction Models

The primary goal of this study was classifying whether the pupil or student failed or completed the course, which was achieved by training various Machine Learning models for prediction. Due to the limited size of my dataset, I opted for using the LEAVE-ONE-OUT cross validation method. The following classifiers were also explored and compared: "LR"- Logistic Regression, "XGboost"

Table 4.5: Performance of certificate earner prediction with different methods (%), the most weighted 60 feature

	Bagging	Boosting	Ctree	kknn	Ksvm	Lr	Mlpe	Random Forest
ACC	80.1	78.11	64.34	71.14	77.61	78.44	73.47	79.44
RECALL	91.14	87.88	79.25	76.92	94.87	87.65	82.05	92.07
PRECISION	82.66	82.49	72.96	81.48	78.27	83	80.92	81.44
F1	86.7	85.1	75.98	79.14	85.77	85.26	81.48	86.43

- Extreme Gradient Boosting ,"MLPE" - Multilayer Perceptron Ensemble, "MLP" - Multilayer Perceptron with one hidden layer ,"KSVM" - Support Vector Machine, ,"kknn"- k-Nearest Neighbor,"NaiveBayes" - Naive Bayes,"naive","Ctree" - Conditional Inference Tree, "Rpart" – Decision Tree,"RandomForest" - Random Forest algorithm,"Boosting"- Boosting,"Bagging"- Bagging.

4.6 Experimental results

During the data cleaning process, the number of learners were reduced from 1370 to 603. The preliminary results indicated that every student-user had a unique clickstream pattern, which was highly similar and independent of user achievement and final scores. Such a finding underpinned the fact that data saved by the MOOC system was suitable for building prediction models. It will be possible to help educational institutions in battling the high dropout rate. They could also take steps to help users whose achievement results fall below the average and thereby prevent negative outcomes. Binary classification experiments were conducted to predict whether a student would successfully complete the MOOC and obtain the certificate. A total of 429 out of 603 learners managed to complete the course, i.e., the most frequent class baseline was 71%. The gain-ratio feature selection ranked featured in an ascending order and the experiment demonstrated that approximately the top 60 features are useful. Results were thoroughly tested in 60 cases and by halving those, a further 30, and then 15 cases. The following table summarizes accuracies achieved by the 12 classifiers using the top 60 features. Apart from accuracy (ACC), the recall, precision, and F-score values of the Completed class are also displayed. Figure 4.6 provides an overview of the classifiers using only the top 15, 30 and 60 features. It can be concluded that among the 30 properties, the most accurate results were achieved by the supported vector machine and the random forest function, while in the case of 60 features, the greatest accuracy was provided by the bagging function. Overall, the most successful models proved to be the bagging (ACC 80.10%) and the random forest (ACC 79.44%) methods (Table 4.5, Figure 4.6).

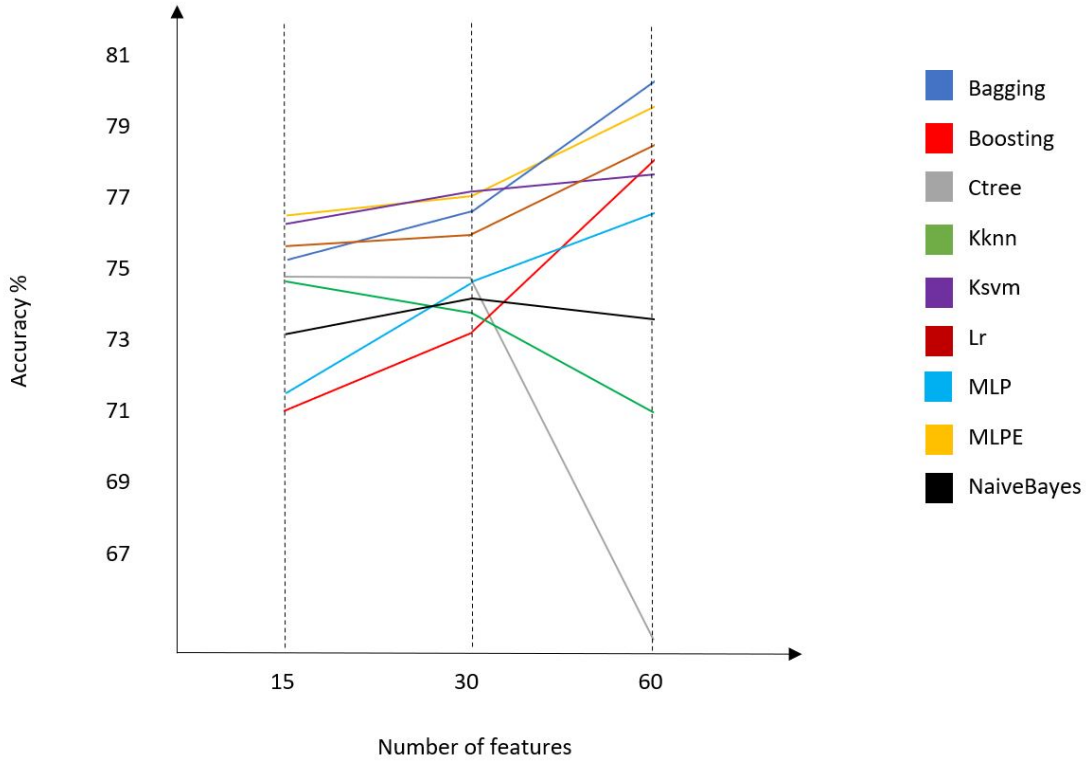


Figure 4.6: Average Prediction performance in the function of the number of features for training

4.7 Discussion

Based on the implemented methods the most notable features in the prediction models can be defined. As can be seen in Figure 4.7, the features that carry the most weight are those that make up the largest proportion of the log data. This particularly refers to the number of mouse clicks or scrolls, whose weight in the prediction accuracy scale was much larger than the result obtained on the input quiz. As expected, the average time spent on the course received the highest weight, followed by the mouse speed and mouse distance spent in the whole course. Other vital issues included the number of clicks, and scrolls, and the number of mouse moves on the page of the curriculum. The initial expectation was that the amount of time would influence the outcome to the greatest degree because those who spend more time on the system, were also likely to learn more. Eventually, I realized that spending more time in the course did not have a considerable effect on the outcome of grades. Still, just like in ordinary schools, the number of days spent learning and testing proved to have an effect during the evaluation process.

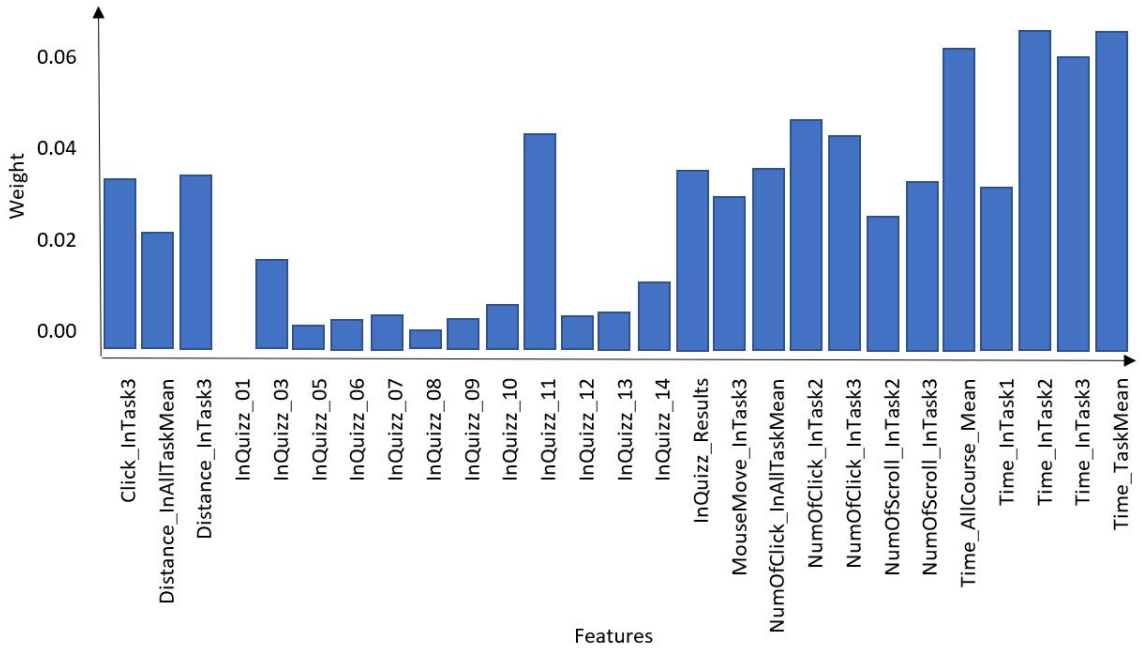


Figure 4.7: The highest weighted features

4.8 Contributions

This thesis focused on the statistical methodology for predicting student performance based on log data which was created in a short MOOC and driven by the teacher. Based on these data sets, it was found that classic Machine Learning models were successful and they were also influenced by several strong features. The best results were achieved by those features which were connected to the learning material, videos or the average value of cursor distance while interacting with the curriculum and videos.
The contributions of this thesis are:

- I was able to add a working analysis system to the weak analysis tools of the Moodle platform, that would serve well for similar portals to meet current measurement requirements. The system was published in Kőrösi and Havasi (2017).

- I designed a data engineering solution that automatically processes input data without human intervention, and which could intervene if extreme values emerge. I proposed 263 features to describe middle-level clickstream sequence of short video MOOCs.

- Despite the relatively low sample size, I still managed to render clickstream based predictive algorithms. I introduced a Machine Learning methodology for feature selection and

binary classification techniques with leave-one-out cross validation for short video MOOCs based on middle- level sequence data. My principal goal was to perform binary prediction of course completion. My models were able to predict who was likely to "Fail" or "Complete" a given online course, which would be an immense help for the faculties that provide e-learning courses.(Kőrösi et al., 2018)

- I implemented and tested nearly a dozen Machine Learning approaches. The most effective tools for my models were the Random Forest and Bagging methods, which achieved about 80% accuracy.

Chapter 5

MOOC performance prediction by Deep Learning from raw clickstream data

As indicated in the previous chapter, the more data one has access to, the wider the range of methods that can be used. The more data there is and the longer the sequence, the better and more accurate predictive models can be built. In this chapter, I use Stanford Lagunita's datasets (Center for Advanced Research On Learning) on low-level data to demonstrate the possibilities of sequence processing and forecasting with Neural Network.

5.1 Introduction

As the Massive Open Online Courses (MOOC) became ever more popular among students and institutions, they sparked a great deal of research interest in MOOC data analytics. (Fei and Yeung, 2015) Despite all their benefits, the quality of MOOCs has been the target of criticism (Whitehill et al., 2015; Xing and Du, 2019; Kloft et al., 2014; Whitehill et al., 2017b). Almost all studies pointed out the MOOCs' low completion rates, not surpassing 7-10% on average, as a property preventing the more widespread adoption of these courses (Yang et al., 2017; Xing and Du, 2019). Stakeholders would benefit from knowing whether a given student was expected to complete the course, especially in light of the low completion rates.

In terms of low-level log data collection in the form of clickstream or social network measures, the MOOC systems offer a treasure trove of data. They enable the design of efficient online user (student) models which would, in turn, serve as a forecasting tool for estimating how many students were likely to drop out, or preferably complete the course. This was made possible by extensive research

into comprehending and, hopefully, increasing the registration and completion rate, ultimately contributing to a better all- round learning experience in MOOCs (Gardner and Brooks, 2018). These issues involved the application of different supervised Machine Learning approaches so as to obtain an estimation for future learning results in MOOCs (Al-Shabandar et al., 2017; Pigeau et al., 2019). Several studies confirmed that low-level data can be used to create more successful prediction models. The purpose of this chapter is to demonstrate this, namely, I present experiments using data from Stanford Lagunita's datasets (Center for Advanced Research On Learning) to predict learner behavior using Deep Learning models on low-level data. I also comparatively evaluate traditional and Deep Learning models. The main contributions in this chapter are:

- I preprocessed and cleaned Stanford Lagunita's data-sets and defied the Machine Learning task and evaluation framework. In the regression task, the goal was to predict student performance over a range of 0-100%, while in the multiclass classification problem I targeted seven classes of student performance. To measure the accuracy and error of the prediction approaches, I created time-varying performance prediction models on a weekly basis.

- I used low-level log data and Recurrent Neural Networks (RNNs) namely Gated Recurrent unit (GRU) to predict student performance at the end of the MOOC course as both a multi-classification and a regression task. The Gated Recurrent unit based models use each element of an activity log sequence (line by line), from the beginning of the given student's activity until a certain point in time, and predicts the student's final performance at the very end of the MOOC.

- I used two different feature extraction methods and tested them on seven prediction models. For classical Machine Learning approaches, I extracted commutative features from the collected data, while for Deep Learning methods, I worked on raw low-level data. I compared classic Machine Learning and Deep Learning solutions for both a regression and classification tasks.

5.2 Related works

5.2.1 Predicting the MOOC dropout rates

The fact that the online educational system is automatic, basing its work on learning analytics, makes it possible to both monitor and recognize those particular students who are at risk of leaving the course. At the same time, it will also be able to support early intervention design (Xing and Du, 2019). Hence, there is a considerable amount of research available about predictive modeling in MOOCs, especially focusing on modelling the likelihood of a given student's dropout, stop out, or overall failure of completing a MOOC. There were earlier studies on student outcome forecast, based on a wide range of characteristics obtained from clickstream data and the natural language used

during postings in discussion forums, social networks, and assignment grades and activity (Gardner and Brooks, 2018). Those works relied on trace data from the introductory week or other specific time intervals which had then served as the basis for predicting students' outcome by way of the created prediction models (Xing and Du, 2019). Such a prediction method enables the efficient detection of whether or not specific students are likely to drop out in the initial education phase, which, however, requires considerable time for feature extraction.
Viewing the problem from a different perspective, scientists compared classification against regression approaches (Gavai et al., 2015). For instance, He et al. (2015) dealt with the Support Vector Machine (SVM) and Least Mean Square (LMS) algorithms in order to identify what the learners' dropout rates were or how well they were performing in the MOOCs during the course period. This work was also conducted on clustering techniques, in which students were put into groups, clustering them on the basis of their student behavioral patterns. (Kizilcec et al., 2013)

5.2.2 Cumulative feature representation in MOOCs

The most popular feature representations include the measures of the distances among log events (time, points, etc.), aggregating the clickstream logs on a weekly basis and/or applying Natural Language Processing (NLP) on discussion forum content. However, the promise of Machine Learning and Deep Neural Networks is to learn the temporal context of raw low-level input sequences in order to make better predictions, not many works have explored this opportunity to date. My extensive literature review indicated that all studies in Educational Data Mining following the classic Machine Learning or Deep Learning approach used cumulated data (daily, weekly, etc.) of feature engineering (Fei and Yeung, 2015; Xing and Du, 2019; Xiong et al., 2019; Yang et al., 2017; Whitehill et al., 2017b; Pigeau et al., 2019). No study was found on educational behavior analysis where the input of the Deep Learning model was the pure raw log-line-level activity data in MOOC courses. In this chapter, cumulative features and classic Machine Learning methods are taken as the baseline solution and the results compared against the Deep Learning solution operating on raw data. These methods were successful, but the key to success lies not in the choice of models but in the preprocessing of the data.
In order to answer the question of whether similar results could be achieved without the labor-intensive preprocessing, I tried to predict student behavior on a weekly basis, similar to the research mentioned above. For the baseline models, I also worked with weekly cumulative data (e.g., distances between actions, number of clicks, etc.).
In terms of Machine Learning architecture, most previous approaches used generalized linear models (including linear SVMs), survival analysis (e.g., Cox proportional hazard model), and Logistic Regression (Xiong et al., 2019; Whitehill et al., 2017a).

5.2.3 Deep Learning methods in MOOC performance prediction

In the past few years several papers were published which started using classic Machine Learning or Deep Learning to predict MOOC outcomes, yet the main concept remained unchanged. Kim et al. (2018) surveyed these studies, so far these have shown low accuracies. The fact that accuracy tended to be low may have been due to the model's continued reliance on feature engineering to decrease input dimensions which seemed to hamper the development of greater and improved Neural Network models (Kim et al., 2018). A good example is the paper Al-Shabandar et al. (2017), in which numerous characteristics were obtained from the learners' historical data, including how many sessions they took, how often they watched the videos, how many courses they participated in, and this was then all fed into a Feedforward Neural Network. Because of the information loss of feature extraction, the accuracy of the current dropout or student performance prediction model is limited, and Deep Learning methods could not give much better performance than classic Machine Learning methods.

Although there is not much research using neural network based models operating on raw MOOC data, in other areas too, such as anomaly research, the DNNs and RNNs have proven to be robust enough for log-linear data, and in most of these research articles unprocessed data is used. Zhang et al. (2016), for instance, opted for using clustering techniques for the raw text from numerous log sources so as to create feature sequences fed to an LSTM for hardware and software failure predictions. Du et al. (2017) implemented special parsing methods on the unprocessed text of system logs to create sequences for LSTM Denial of Service attack identification (Brown et al., 2018). I address this challenge by using RNNs directly on raw low level log data, rather than hand-designed feature extracted statistics.

5.3 Methodology

From an algorithmic perspective, this chapter explores the power of Deep Learning in MOOC context. My prediction algorithm uses raw clickstream data to forecast students' course performance.

5.3.1 Data preprocessing

Before introducing the Neural Network models created as part of my doctoral studies, I am going to discuss the method used for addressing data sparsity, as dealing with sparse data is one of the challenges of doing predictions in MOOCs (Yang et al., 2017). In Stanford Lagunita's datasets (Center for Advanced Research On Learning), most students did not answer all of the quiz questions in a given MOOC, leading to a sparse set of quiz responses for any individual student. To handle this problem, I only used those users who took the second quiz after Week1, or filled in the third quiz after Week2, etc. To avoid under or over learning problem, I removed extreme outliers (Tukey

et al., 1977) from datasets. This logic is reflected in the definition of the dataset.
My prediction algorithm was based on raw clickstream data in order to provide a prediction of how well students would perform in an online course. I used the students' final assessment quiz responses to define the course performance measure.

5.3.2 Evaluation methodology

To measure the accuracy of the prediction approaches, I build temporal performance prediction models on a weekly basis. The log data observed in Week2 was directly added to the dataset of Week1, and similarly for the other weeks as well. I used the data collected until the current week to predict the student's outcome in the very end of the course as shown in Figure 5.2. I extracted the cumulated features from the collected data for the Baseline solutions and trained a GRU model on the raw data of the given period.

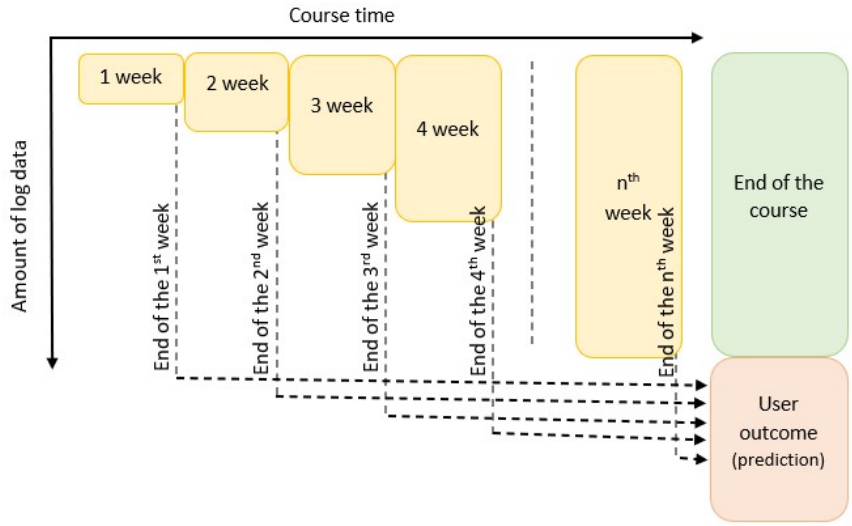


Figure 5.1: Formulation of performance prediction problem

After the training step, I evaluated the accuracy of the prediction models on the holdout set of students. The predictions on this hold out set were compared against the true value (either the actual score or the performance class) of the students' final course performance. Following Willmott and Matsuura (Willmott and Matsuura, 2005), for measuring the accuracy of the solutions, I used accuracy scores (ACC) for classification and root mean squared error (RMSE) for regression. I split the students into four groups randomly and employed a 4-fold cross-validation at each week. The user performance prediction was measured in two ways:

- as a simple regression problem (0-100%) with recurrent network and logistic regression, and

Table 5.1: Feature set of Stanford's course

Feature	Explanation (feature aggregated on a weekly basis)
Lecture view	Number of lecture videos viewed by a student
	Number of times a student visits a lecture site
	Number of plays, stop, pause, forward, backward
	Number of plays, stop, pause, forward, backward
	The distance of time between two log events
Quiz attempt	Number of quizzes attempted by a student

- as a multiclass problem (0-6 point) with recurrent network and XGboost. To obtain labels for the multiclass, I used uniform democratization of continuous data (0-10% = 0, 11-20% = 2, etc.)

5.3.3 Baseline solutions

In order to compare my study with other solutions, I implemented baseline solutions, i.e., feature extractors on raw clickstream and used traditional classifiers and regressors on these feature set. The extracted cumulated features from the raw data (see Table 5.1) were created in the same method as described in Chapter 4. All of these features were normalized by min-max scaling with the maximum and minimum values of each feature in training data-set. No hyper-parameter tuning was performed, since the main goal was not to find the most accurate model. In order to compare my research with other solutions, I compared the classic Machine Learning namely XGboost, XGBregression and Ridge Regression to a GRU based model on raw log-line level data (GRU).

5.3.4 Deep Learning architecture

GRU is a good choice for solving problems that require learning long-term temporal dependencies. Figure 5.3 depicts the architecture of the proposed Deep Learning model consisting of nine layers. The input layer of the deep network uses a flat feature structure (one hot encoded 3-dimensional data), as seen in Figure 5.2. The rows in the 3D dataset contain user-generated log data. These rows represent a user-generated 2D sequence of actions, set up in chronological order. In this 2D sequence, a line is a 1714-long vector, which represents one of the 1713 possible actions, illustrated with a One Hot Encoded vector, along with a single feature denoting the time elapsed since the last action. An action encodes either the type of action (e.g., "video stop") or the item that was accessed. For example, when the student opened "Lecture1 Part3," which is a web-page containing a lecture video, this event was logged. Next, when the student played the video, a new action was added to the sequence, but only its action type "video play" was stored. While this condensation of data was necessary for keeping the input space at tractable size, I expected from the RNN, that it would be able to learn the representation of items in its hidden states.

RNN tends to quickly overfit a training data-set. To reduce the chance of overfitting, I used dropout layers which offered a computationally very cheap and remarkably effective regularization method to reduce overfitting and improve generalization error in my model. Following the GRU and dropout layers, I applied fully connected (dense) hidden layers with a different number of neurons. I use the same network architecture for the regression and classification tasks besides the output layer. At the regression task, I used the result from the single output neuron without any transformation. At the multiclass classification task, the output layer consisted of a vector that contained values for each class along with a SoftMax activation function.

My primary goal was not to find the most accurate model, however, the results for the selected parameters were sufficient to demonstrate the feasibility of using recurrent neural networks to predict MOOC students' performance.

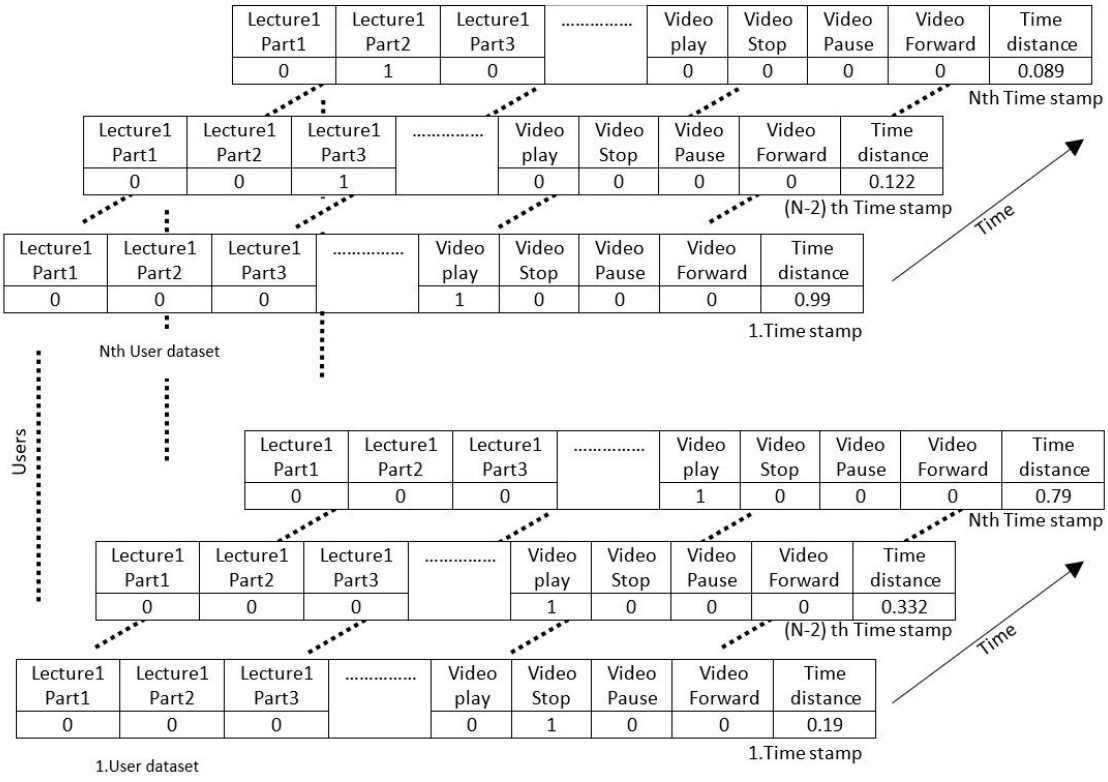


Figure 5.2: Formulation of 3-dimensional data

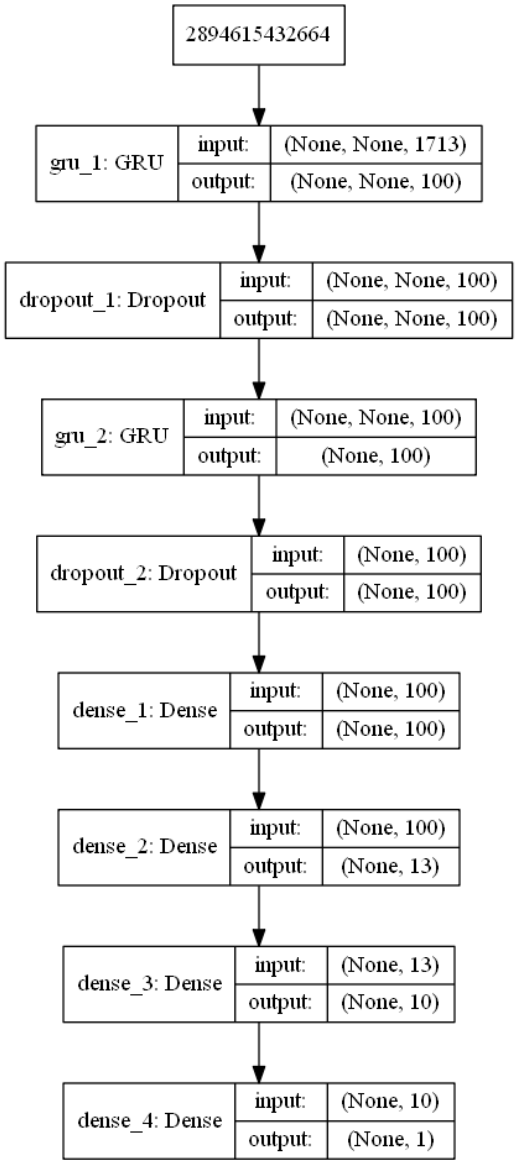


Figure 5.3: Architecture overview of the proposed RNN model

5.4 Dataset

The dataset used in this work originated from Stanford Lagunita's MOOC Computer Science 101 from Summer of 2014. The MOOC ran for six weeks, with video lectures, optional homework assignments, discussion forums, and quizzes. The original main dataset contained 39.6 million actions from around 142,395 students, where each action represented accessing a particular event in the course (video view, assignment view, problem view, etc.). Of the 142395 students, 28,368

Table 5.2: The Stanford Lagunita Science 101 dataset

Feature	Examples	No. unique value
Links	'courseware/z187/z172/', 'courseware/z187/z184/'	243
Events	'load_video', 'login', 'problem_check'	34
Resource	'Q1', 'Week 2 Course Survey'	35
Success	0,1,-1* (* missing value)	2

Table 5.3: Number of logged events in the different progress sections of the course

Event type/progress		20%	40%	60%	80%
Video	Load	34999	67411	97070	127552
	Play	61003	123338	182821	238408
	Seek	18862	41574	61490	80236
	Speed change	3442	5668	7600	9516
Quiz	Quiz 1	20283	42655	73102	110348
	Quiz 2	14581	33294	61091	96684
	Quiz 3	8281	21760	48636	81614
	Quiz 4	46	648	5365	9261

were active and 15,673 completed enough assignments and scored high enough on the exams to be considered "certified" by the instructors of the course. The certified students accounted for 17.79 million of the original 39.6 million actions, with an average of 1,135 actions per certified student. In my research, the set of 12,015 students was used.

On the filtered data each logline was made up of five attributes describing a clickstream level event: event type (categorical variable), visited URL (categorical variable), re-source name (categorical variable), and quiz success (binary variable). Table 5.3 lists some of these examples.

5.5 Results

I ran my models at the end of each week, i.e., after a quiz, and based on that information, I predicted the final completion of the course. For example, Week1 represented all collected log data from the start of the course until the end of Week1, and Week2 represented the collected data until the end of Week2. The results of the proposed GRU method and the baseline methods on Computer Science 101 dataset are shown in Table 5.4. The results of the two experiments, as summarized in Table 5.4and Figures 5.4 and 5.5 , show that the GRU model is generally better than the XGBoost-regression and XGBoost, as both RNN based models increase their prediction quality week by week.

This indicates that RNNs with raw datasets are "more sensitive" and are better able to identify patterns than XGBoost or XGBoost-regression. In addition to basic transformations (sum, avg, normalization), I also implemented other methods to increase the performance of the XGBoost model, as discussed previously in Chapter 3, but apparently this solution was not efficient enough.

Table 5.4: Tabulated statistics for results which corresponding to GRU for baseline methods. The two columns on the left compare GRU and logistic regression with RMSE (root mean squared error) while the two columns on the right show the results for multiclass classification problem.

	RMSE			MAE			ACC	
	XGBreg	Ridge	GRU-reg	XGBreg	Ridge	GRU-reg	XGBoost	GRU-class
week1	16.010	18.451	10.001	11.706	12.854	7.743	0.361	0.496
week2	16.217	19.173	9.831	11.784	13.564	7.451	0.378	0.486
week3	15.730	20.779	9.378	11.251	13.101	7.117	0.39	0.545
week4	15.346	29.207	8.746	10.202	15.378	6.096	0.405	0.559
week5	15.265	25.355	8.653	10.585	15.280	6.568	0.405	0.551

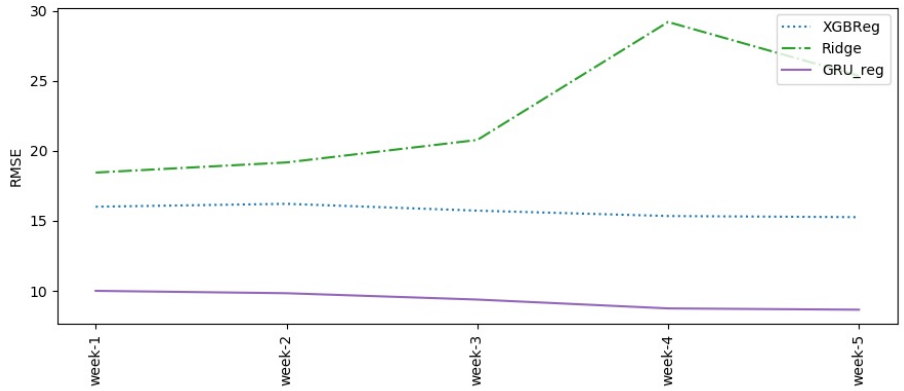


Figure 5.4: The results of the proposed GRU method, XGboost regression and the RIDGE regression on Computer Science 101 dataset

Remarkably, for all the weeks, the proposed method outperformed the best baseline by 30% of ACC, or 15 of RMSE.

As expected, the performance started out poor (low ACC or high RMSE) but steadily improved as more data were fed into the model. Predictions continued to improve until approximately week3, after which the performance leveled out. After 3 weeks of observations, my model achieved an accuracy of 54%, significantly better than the 39% baseline of predicting the class role. My empirical results demonstrated the feasibility of using Recurrent Neural Networks on raw log-line level dataset to predict the performance of MOOC students. I do expect, however, that a more extensive search for the optimal choices of the number of units and hidden layers (through e.g., hyper- parameter tuning, embedding layer) will improve my prediction quality even further. The experiments were implemented in Python by using Keras[1] Google's Tensor-Flow (Abadi et al., 2016), Sckikit learn (Pedregosa et al., 2011) and XGBoost (Chen and Guestrin, 2016) package.

[1] `https://github.com/fchollet/keras`

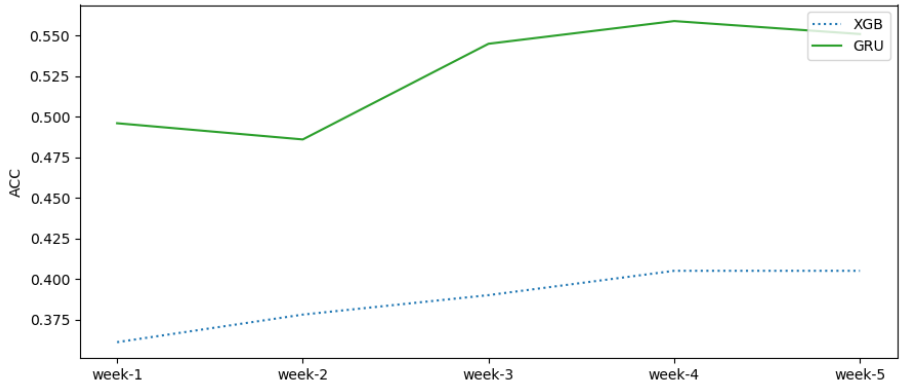


Figure 5.5: The results of the proposed GRU method and the XGBoost on Computer Science 101 dataset

5.6 Discussion

To gain a better understanding of the obtained results, I performed another examination (see Figure 5.7a), where the relation between the number of log data and the absolute error (AE) of the model is plotted. In this process I calculated AE for every user falling into a particular bin of log data sequence length and made a boxplot form this data to compare the outputs of the two models.

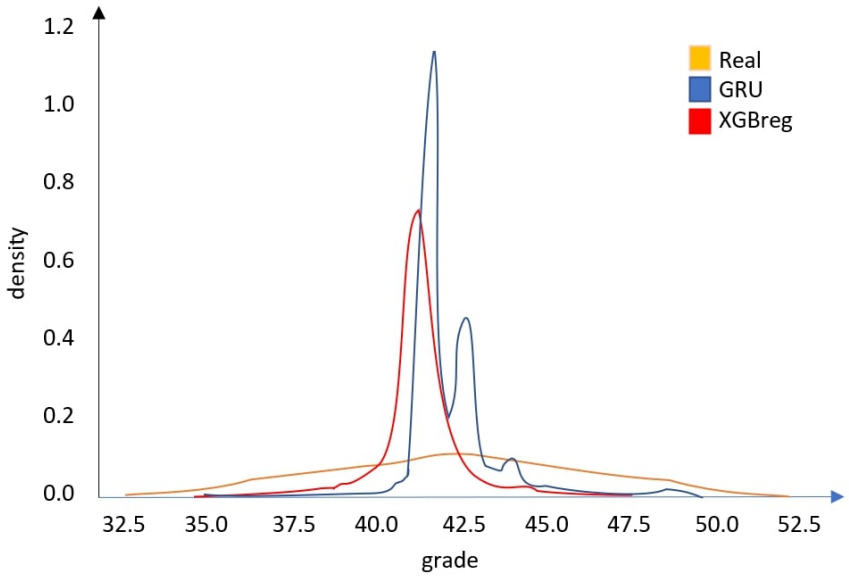
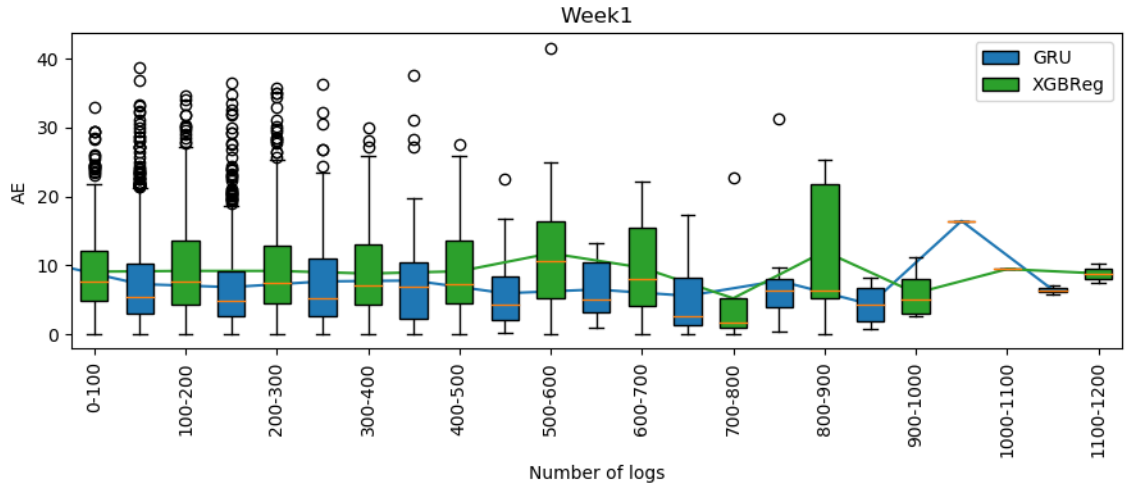
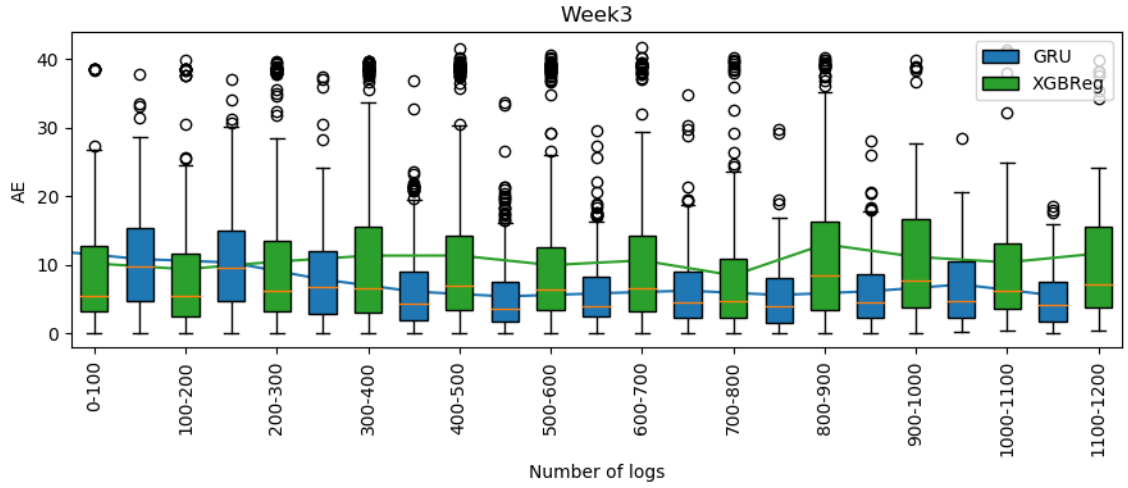


Figure 5.6: The distribution of Students' final outcomes - The test results of the proposed GRU method and the baseline (XGBRegression) methods on Computer Science 101 dataset in Week4 (n = 2159)

In the first two weeks, the GRU network provided better results than the traditional approach on any log size. In Week3, Week4, and Week5, the traditional approaches outperformed RNNs at short log sequences, so that GRU was only superior when it had more data of longer sequences that provided extra information.

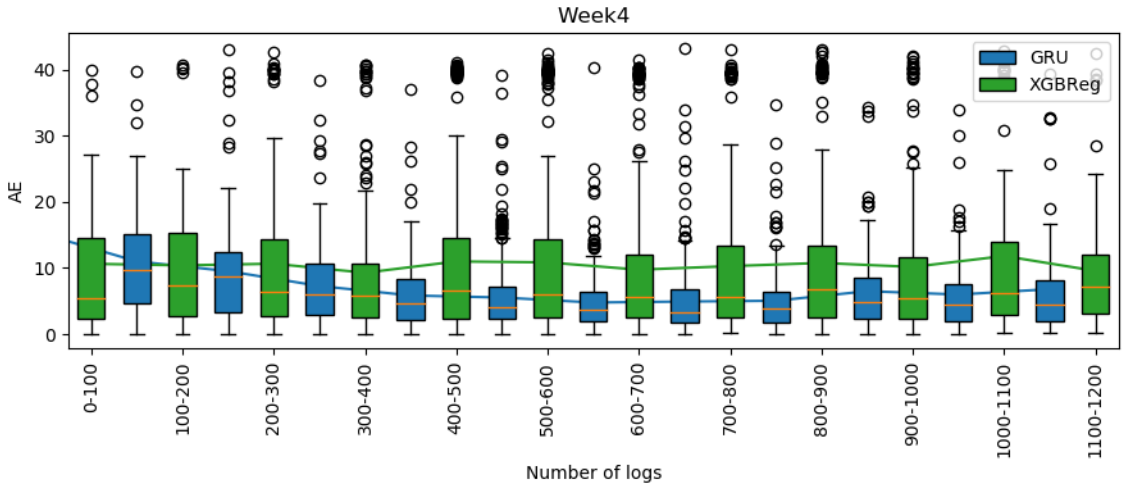


(a) Boxplots of average error (AE) achieved by GRU-reg and XGBReg in the function of students' log sequence length at 1^{th} week

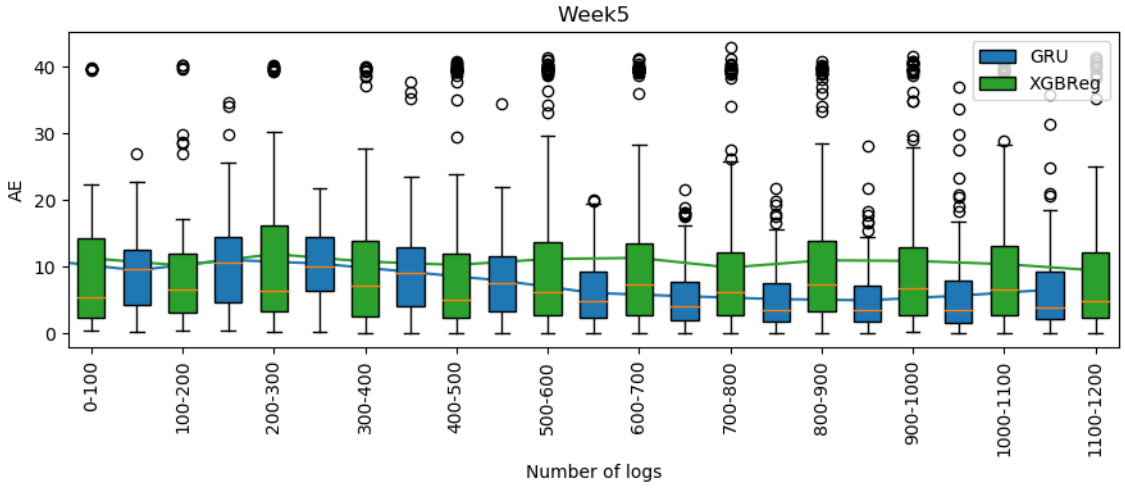


(b) Boxplots of average error (AE) achieved by GRU-reg and XGBReg in the function of students' log sequence length at 2^{nd} week

Another challenge was the variable sequence length because that was not identical in length for all users, actually, it had an impressive impact on the performance of the model. For instance, some users finished the six-week course very quickly and in a straightforward manner (which means that



(c) Boxplots of average error (AE) achieved by GRU-reg and XGBReg in the function of students' log sequence length at 3^{rd} week



(d) Boxplots of average error (AE) achieved by GRU-reg and XGBReg in the function of students' log sequence length at 4^{th} week

Figure 5.7: Boxplots of average error (AE) achieved by GRU-reg and XGBReg in the function of students' log sequence length

they fell into the category "Does not check additional information"), whereas other users would look into everything and spend much more time checking the details of the learning material. These results also highlighted that there was a significant correlation between the length of the sequences and the prediction quality of the RNN.
I experimented with various regression models, but they provided almost the same or even worse results (Ridge regressor was among the best ones, see its results in Table 5.4). This fact was also supported by my prediction density distribution graph of real and predicted course outcomes. (Figure 5.6). The diagram shows that XGBReg was not able to find any useful relationship among its features. On the other hand, GRUreg was capable of identifying different class-like regions in its prediction space. The results are not exhaustive, but they are encouraging in terms of further exploration.

5.7 Contributions

In this thesis, I presented a recurrent Neural Network for solving outcome performance prediction problem in online learning platforms. The main contribution is the created prediction model itself, which could use raw low-level datasets, and obtain equal or even better results than the regular prediction models. The key advantage of this model is that no manual feature engineering is needed, because it could be automatically extracted from the raw log-line level records. In this way, this approach can save a lot of time and human effort, and ignore the possible inconsistency introduced by the hand-made process. Experimental results on Stanford Lagunita's dataset (Center for Advanced Research On Learning), containing data of 12015 students highlighted that the expected model could achieve significantly better outcomes than the baseline models. The results for the model are sufficient to demonstrate the feasibility of using recurrent Neural Networks if a large dataset is available.
The main contributions of this thesis are as follows (Kőrösi and Farkas, 2020):

- I proposed an evaluation methodology for student performance pereditction along with data preparation step for low-level clickstream event data forming a 3D tensor for RNNs.
- I built a feature extraction and classic Machine Learning-based baseline pipeline and Deep Learning model to predict student outcome as a regression and multiclass classification problem.
- My experiments confirmed that Deep Learning based prediction pipeline without manual feature engineering is able to outperform the classic Machine Learning based solution when a huge amount of data is available.

Chapter 6

Deep learning models and interpretations for MOOC performance prediction

Low-level sequences with longer and more extensive attributes allow for better model building, making it possible to work with extremely rich and long time series. In this chapter, I aim to demonstrate how such long multivariate time series with discrete values can be efficiently processed using recurrent and convolution Neural Networks and embedding layers. I also present the efficiency of the methods used, as well as provide an insight into the black box of my Neural Network, explaining the behavior of the individual activation layers.

6.1 Introduction

Recently, Neural Networks have been widely used as low-level sequence predictors and time series forecasters, as they can capture complex nonlinear patterns. (Ke et al., 2017) The most commonly used model is the Recurrent Neural Network (RNN) which has outperformed statistical models, e.g., autoregressive and moving-average models (Priestley, 1981). Apart from the dominant RNN models, there are Convolutional Networks (CNN) proposed for time series forecasting and sequence classification, namely Temporal CNNs (TCNN). Whereas the majority of the time series Deep Learning models have been applied to numerical data, event logs, such as clickstream-level MOOC data used, consist of multivariate discrete-valued sequences. Hence, time series Deep Learning techniques could not be directly applied. Still, most of the discrete-valued sequence prediction solutions were published for Natural Language Processing. The raw event logs were significantly longer than natural language sentences, with their varying lengths, thus NLP techniques could not be applied directly.

To handle these special characteristics of the given clickstream-level MOOC dataset, I created an embedding based Deep Learning model architecture. This chapter outlines how I trained state-of-art RNN and CNN models to predict the outcome score of the students at the MOOC. I conducted analysis using various embedding layers to represent the multivariate discrete-valued data. Recurrent and Temporal Convolutional Neural Networks provide accurate forecasts without having any access to explicit knowledge about the investigated system.
Deep Learning methods are usually considered 'black boxes,' where it is almost impossible to fully understand what, why, and how RNN and CNN make their forecasting decisions. My research aimed to reveal the operational black box of the RNNs and CNNs trained for time series regression. I proposed three visualization techniques which would help domain-expert users in interpreting discrete-valued multivariate time series regression neural models.
In this particular study I examined the online behavior of Massive Online Open Course (MOOC) students. I introduced a DL architecture to predict the outcome score of the students in a MOOC. Chapter 6 is centered around two specific contributions:

- I presented experimental results on various deep learning architectures and embedding strategies, evaluated on a MOOC clickstream event, discrete-valued time series regression task.

- I proposed application-oriented, user-friendly visualizations for explaining the behavior of the Machine-Learnt RNN and CNN, regression models.

6.2 Related Work

6.2.1 MOOC clickstream data analysis

Analyzing student behavior in MOOCs directly on the clickstream-level is quite a novel field of study. Li et al. (2020) and Baker et al. (2020) sought to understand student behavior using log sequence from different MOOC courses. They investigated and visualized behavioral patterns of student groups by employing statistics and classic Machine Learning methods over hand-crafted features. To the best of my knowledge, the results presented in Chapter 5 and published in Kőrösi and Farkas (2020) are unique, as it is the first study to date utilizing DL techniques to exploit raw clickstream data recorded during MOOC courses. As demonstrated earlier, my model managed to outperform hand-crafted feature based classic Machine Learning approaches.
I implemented Deep Learning techniques to solve the same goal as Li et al. (2020) and Baker et al. (2020), i.e., to analyze student behavior. I drew educational conclusions similar to those presented in Li et al. (2020) and Baker et al. (2020) , but since I used raw sequences directly, my approach did not require any feature engineering of pedagogical expertise.

6.2.2 Neural models

Recent advances in neural architectures and their application to raw time series and sequences offer an end-to-end learning framework that is often more flexible than classic feature engineering based approaches (Guo et al., 2017). Among neural architectures, recurrent solutions are very popular. For example, Koehn et al. (2020) demonstrated that an RNN based method was able to outperform common Machine Learning while using mixed continuous and discrete-valued time series to predict the order value. Guo et al. (2017) proposed the feed forward NN and embedding layer based DeepFM for multivariate partially raw discrete-valued clickstream data.

Apart from the recurrent approaches, convolutional models capable of considering the temporal dimension have recently been proposed. Sadouk (2019) used Temporal 1D-CNNs (TempCNNs) to identify stereotypical motor movements from sequenced data. They applied convolution layers in the temporal domain and proved their effectiveness for handling the temporal dimension for time series classification. Sadouk et al. (2018) conducted an exhaustive study of Convolutional Neural Networks where convolutions were applied in the sequence recognition tasks. They suggested that the reason why so few research papers used CNNs was that most papers were focused on continuous time series problems, such as EEG anomaly detection or human activity recognition, and not on discrete-valued sequence classifications. My work was motivated by these studies, thus I experimentally compared CNN and RNN models on discrete-valued sequences.

6.2.3 Embedding

In Deep Learning based approaches, the discrete-valued sequences must be transformed into the numeric space. Using one-hot encoding might not prove to be overly useful, as it explores the dimensionality of the input feature vector and dramatically increases its sparsity. Inspired by Natural Language Processing, I transformed my categorical data into a dense space utilizing embeddings. The methods in the previously mentioned papers used encode words as vectors based on contextual similarities and then fed them into the recurrent or convolutional Neural Network. The embedded vectors were generally trained together with the time series/sequence model training process (Li et al., 2018). The embedding of discrete-valued sequences was successfully applied in user behavior analysis. (An and Kim, 2020), for instance, presented their neural user embedding (NEU) approach which was capable of learning informative user embeddings by using the unlabeled browsing-behavior. (Koehn et al., 2020) reported impressive clickstream classification results where they applied LSTM and GRU architectures and embedding layers. Cheng et al. (2016) introduced the Wide and Deep feature representation method. In their terminology, Wide representations were one-hot encoded features which could memorize sparse feature coincidences, while Deep representations consisted of dense embeddings which gave generalization power to Deep Learning systems. In my own research work, I embedded discrete-valued attributes for enhancing the generalization capability of my Neural Networks.

6.3 Dataset

In this chapter the same Stanford Lagunita's dataset (Center for Advanced Research On Learning) was utilized and investigated as already used in Chapter 5. Whereas previous parts of this dissertation detailed the handling of the discrete variables with one hot encoding, this chapter explains how the same problem was solved using embedding layers. Apart from processing the data, I also changed the target parameter, namely I sought to predict the performance of the users in a different way. The aim of this study was to predict the student's final scores (from 0 to 100) achieved in the four quizzes based on the raw log sequence. The user could take the quizzes multiple times, but the final score was the sum of the first attempts. The goal was to predict the final score at a certain point of the course content (20%,40%,60%,80% of the progress of the course content), as opposed to the goal of Chapter 5, which was to predict the final scores at the end of the course weeks. To gain a better understanding of the users' learning behavior and the predictive power of raw log data, I split the time series into progress sections, namely 20%, 40%, 60%, 80% of the course progress. Table 5.3 displays the number of some event types.

6.4 Embedding-based Multivariate Sequence Regression

The focus of this study was on multivariate discrete-valued sequence neural regression. As mentioned above, I proposed a Deep Learning architecture in my MOOC scenario, depicted in Figures 6.1 and 6.2.

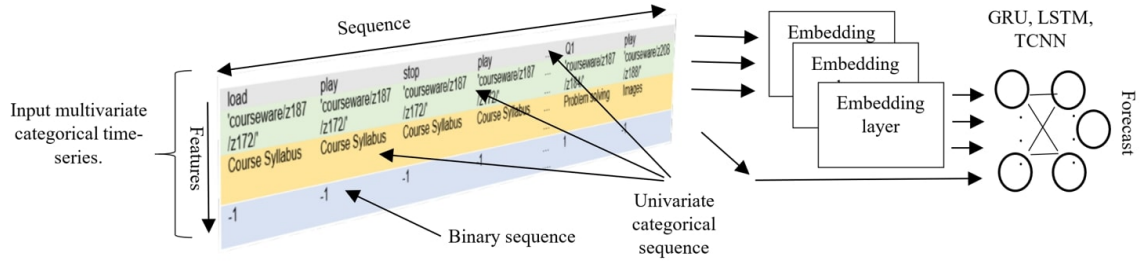


Figure 6.1: A unified Deep Learning framework for discrete sequence forecasting. A DL architecture where the Embedding layers are designed to encode each categorical attribute separately. Following this, the TCNN and RNN networks learn the hierarchical representations of the sequenced data.

Embedding layers are designed to encode each categorical attribute separately. Following this, the TCNN and RNN networks learn the hierarchical representations of the sequenced data. Recurrent and Temporal Convolutional Neural Networks proved their ability to discover patterns in multivariate time series, making forecasts without explicit knowledge of the inspected system (Lee

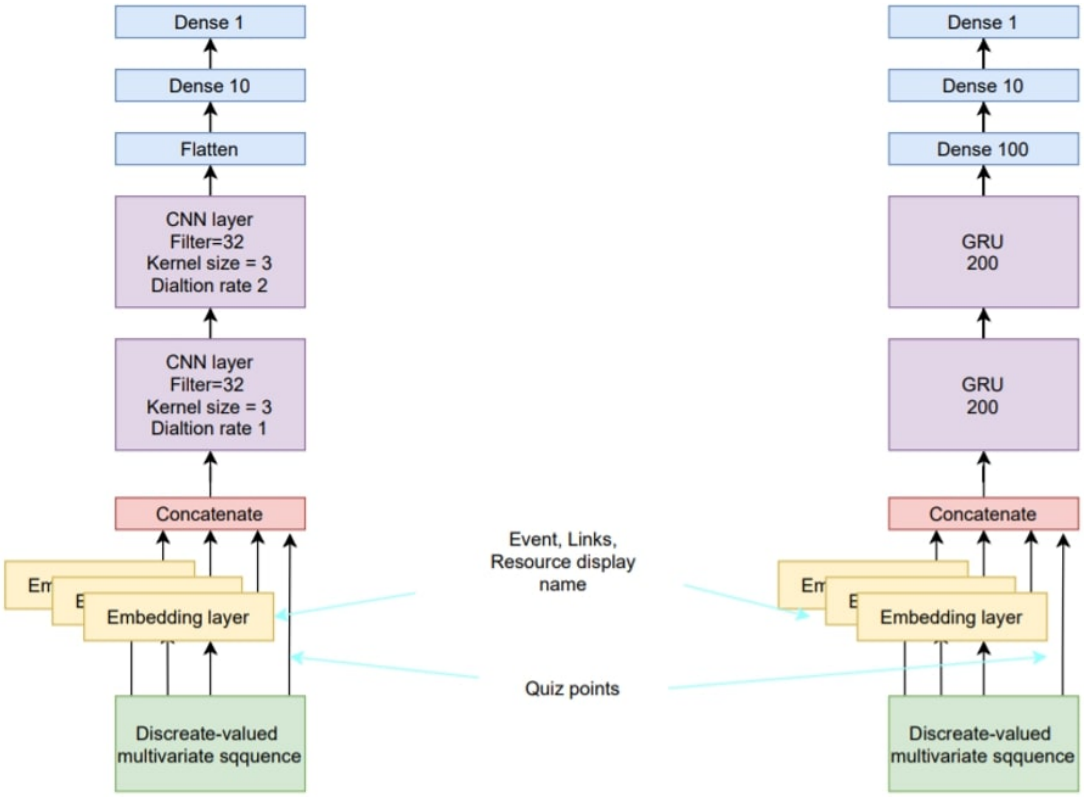


Figure 6.2: Overview of the configurations for multivariate sequence prediction. TCNN architecture is displayed on the left, RNN (GRU and LSTM) on the right. The numbers in boxes refer to layer sizes, i.e., the number of hidden units.

and Hauskrecht, 2019). My research aimed to create an accurate way of using the same methodology on discrete-valued sequences in RNN and TCNN. The framework for discrete-valued sequence prediction is depicted in Figure 6.2: the proposed representation of discrete sequence in the form of a vector embedding. Instead of any data preparations, the label encoded uni-variate sequences themselves were inserted into the embedding layer which could autonomously transform the categorical labels into a continuous space. This entails the following advantages: it does not contain any artificial "human based" parameters which could affect the behavior of the model. While the embedding layer learns without human intervention, it does not strongly depend on how many data points are available; it is suitable for the time based discrete sequence from high-dimensional attractors.

6.5 Regression results

The student dataset was randomly split into training dataset (9502 sequences) and evaluation dataset (4072 sequences). The mean absolute error (MAE) of final student scores was taken as an evaluation

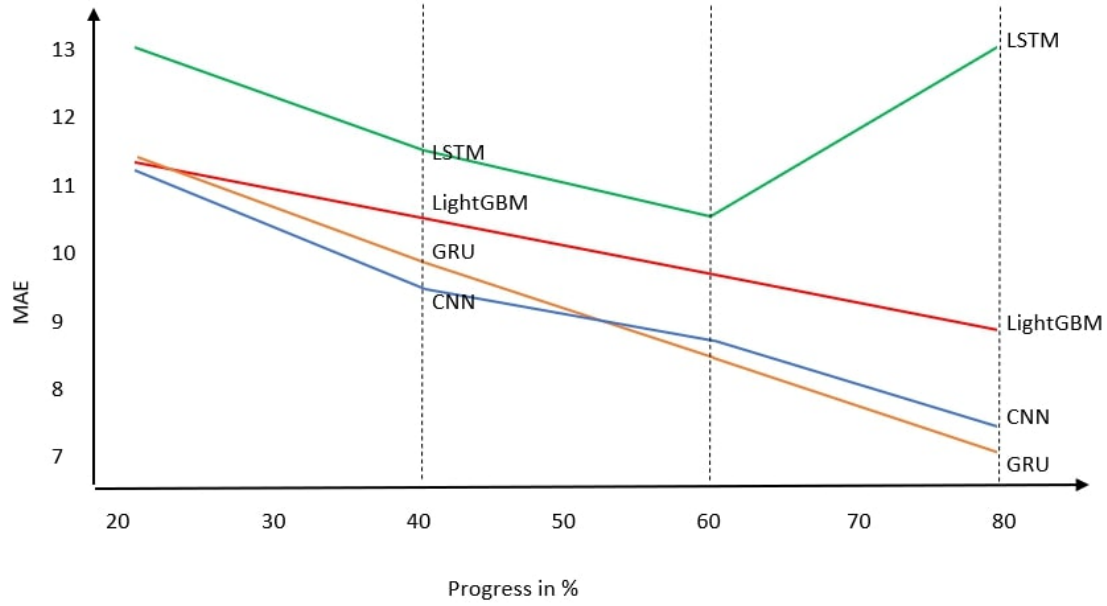


Figure 6.3: Mean Absolute Errors achieved by various models at different progress state of the course

metric. The size of the neural networks was calibrated on a development set (random subset of the training dataset).

Embedding layers were used of length 30 (the sizes of other layers are shown in Figure 6.2). Tangent activation function was implemented in the GRU and LSTM experiments, while ReLU was used in the CNN ones. As the optimizer, ADAM was used with the default 0.0001 learning rate and early stopping criteria. The sequences were post padded to lengths which varied in function of student progress datasets (lengths: 20%: 220, 40%: 520, 60%: 720, 80%: 920). In the baseline model, the user's behavior in the course was encoded as a 28-dimensional feature vector. These cumulated features consisted of the number of video interactions (play, stop, pause), quiz success (quiz 1, 2, 3, 4), etc. LightGMB regression (Ke et al., 2017) was conducted on the cumulated features as a baseline. Figure 6.3 shows that there was no significant difference among the models at 20% progress. The CNN architecture yielded either the best, or the second-best performance in most of the datasets. Table 6.4 displays that GRU and CNN with embedding 'captured' the patterns in data better and provided a much better forecast than other implementations. The LSTM was also tested with embedding, but it generated unmeasurable results. This could be explained by the amount of data, given that LSTM is sensitive to long sequences, and in this case there were on average 720 time-steps.

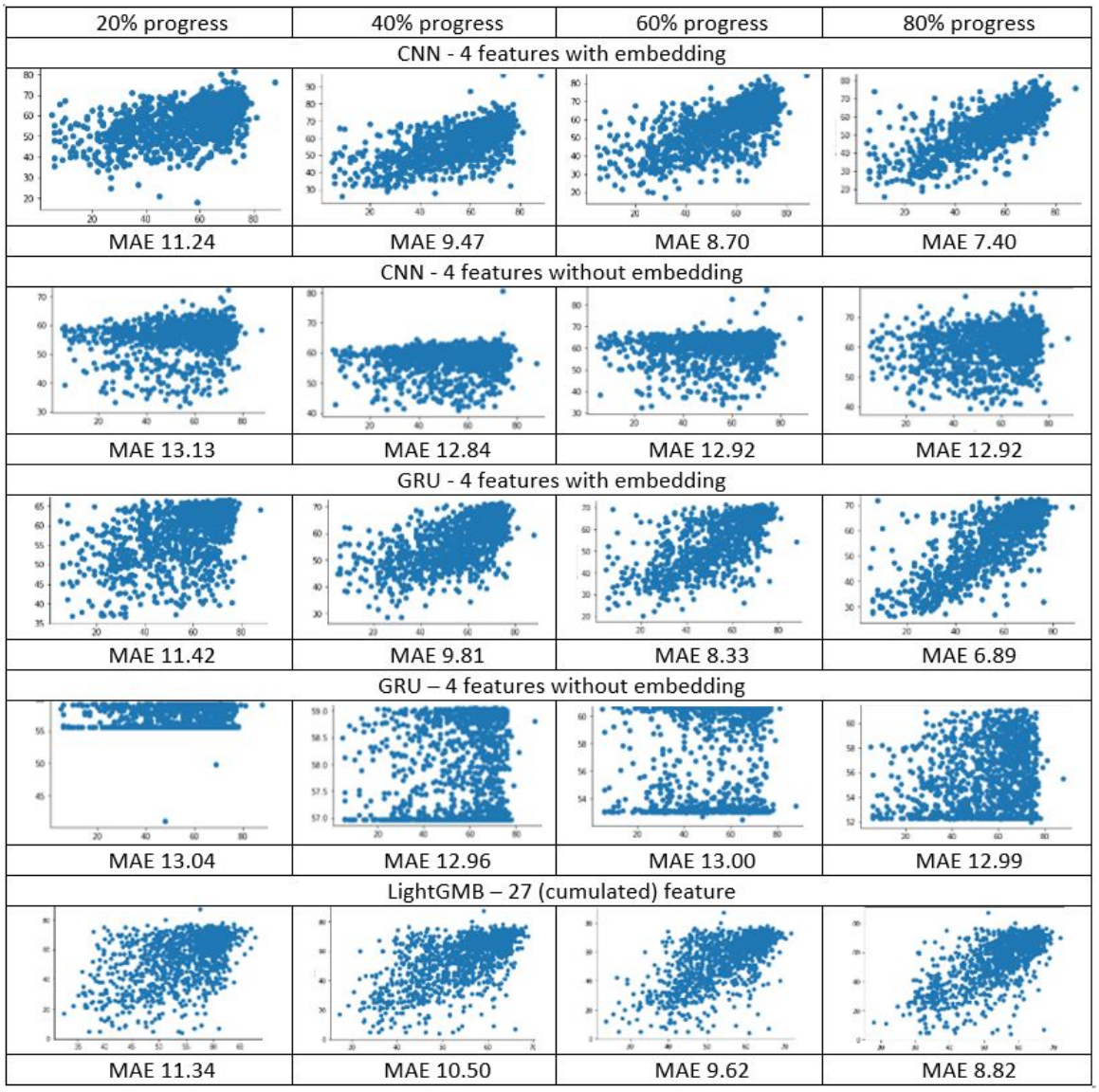


Figure 6.4: Real (x axis) vs predicted (y axis) final student scores results from LightGMB, CNN, GRU, and LSTM models in different progress points of the course.

6.6 Interpretations

Recurrent and Convolutional Neural Network models have recently achieved state-of-the-art sequence prediction accuracy. However, in terms of data analysis, it remains unclear what the models learned, how these approaches identified patterns and meaningful segments from time series. This section explores these issues in order to gain better understanding of the behavior of categorical time series prediction DNN models.

6.6.1 Embedding spaces

The MOOC dataset contains four attributes, including three discrete-valued variables. I transferred those three attributes to three parallel embedding layers (see Figure 6.2) so as to learn and transform discrete-valued variables into an nth dimension continuous space.

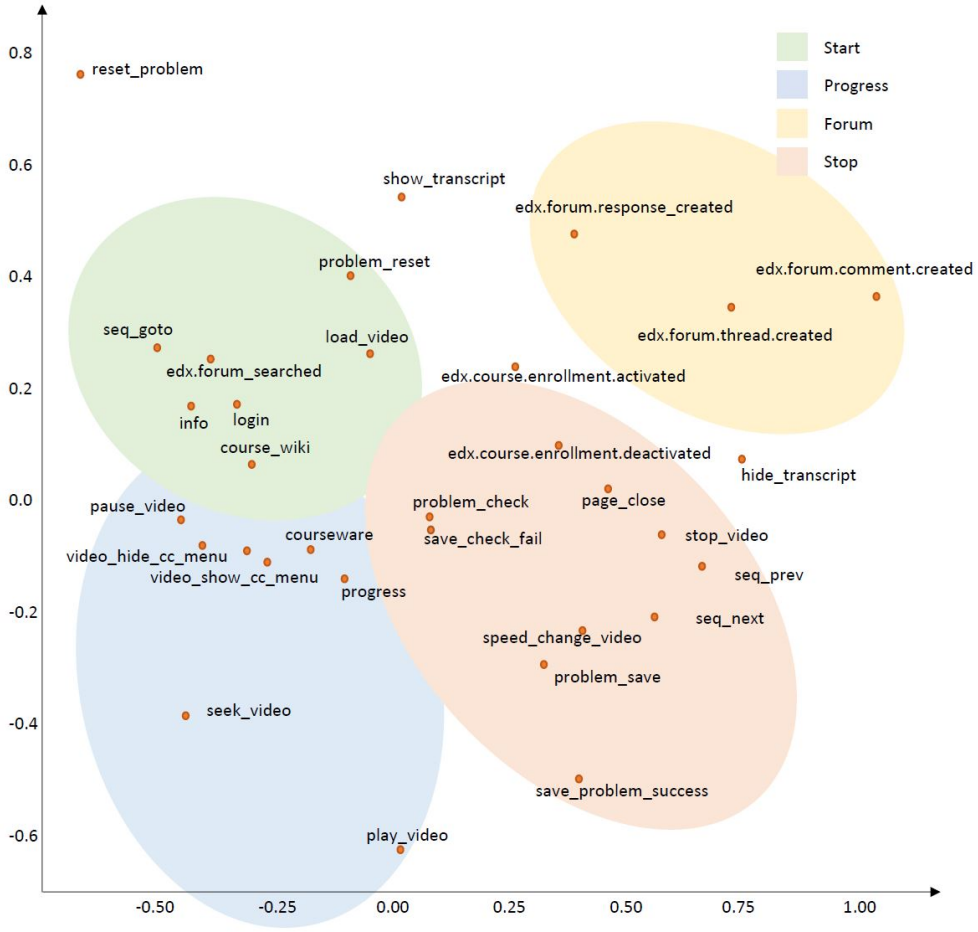


Figure 6.5: T-Distributed Stochastic Neighbor Embedding (t-SNE) results for EVENT feature embedding layer. The figure illustrates the learning capability of the Embedding layer, as it was able to group events from a raw dataset into similar groups according to their role in the course. The figure shows the four main groups based on the start, processing, and stopping of the course content and forum posts.

To understand the trained em- bedding layers behavior, I used the output of trained embedding layers which was trained on the event attributes, further, I employed t-Distributed Stochastic Neighbor Embedding (t-SNE) to map the 30-dimensional embedding space to 2D. Embedding with the t-SNE method is useful because embeddings are learned, thus events, links, or resources that are more similar in the context of my problem are closer to one another in the embedding. The general idea was to group each event type according to its "location" of the curriculum. For example, `play, stop, pause` would be in the group of video interactions, while `problem_check, problem_reset, save_problem_succes` would be in the quiz group. However, the embedding layer processes this differently. Figure 6.4 highlights that both video and forum based events are coming closer to each other, yet, what is even more peculiar is the fact that the save and play video events seem to be similar. The trained embedding layer was able to significantly improve my forecasting results (GRU based model average increase 8%, CNN based model average increase 10%), proving to be an effective aid in preprocessing the discrete sequence.

6.6.2 Temporal saliency

The temporal activity of students during a MOOC is a fascinating pedagogical area to explore. The visualization below indicates how strongly the different temporal segments relate to the Deep Learning prediction. I also aimed to investigate whether students with various outcome scores display different temporal behavior. The RNN and CNN methodology uses the output of embedding layers and one binary attribute to train the models (see Figure 6.2)). As a result of the training process, I used the output of CNN and RNN layers with the absolute value of the derivative of the loss function with respect to each dimension of all sequence inputs. Each row in Figure 6.6 corresponds to the predicted student outcome group. Since the first group comprised very few users (0-10 final student scores) and so did the last group (80-90 final student scores), this interpretation was omitted. The columns in the figure represent the output of the CNN and GRU layers as the mean of the loss values.

During visual inspection of the mean of the loss function values, it becomes clear from the heat map (Figure 6.6) that CNNs tend to focus on short contiguous subsequences ("windows/boxes") when predicting the outcomes, whereas GRU uses the whole sequence for the same task. In other words, the CNNs model finds "motifs" that are important for prediction, by comparison, GRU seems to give a different gradient for each time step. The results were almost identical to those seen in Lanchantin et al. (Lanchantin et al., 2017), in their research on the use of CNN and RNN to understand DNA sequence. They found that the recurrent Neural Network tended to be spread out more across the entire sequence, indicating that they focused on all sequences together and inferred relationships among them. They also mentioned that, when using convolutional and recurrent networks for sequence forecast, those tended to have strong heat points around motifs, where one could see that there were other steps further away from the motifs that were significant for the model results. Both

CNN and GRU had a considerably wide range of steps, moreover, for the low outcome final student scores (0-40) the RNN model used the entire sequence, while for high final student scores it used only the first part of the dataset. CNN used windows of almost the same size as all outcome classes, and although the distribution of weights was different, it learnt from the middle of sequences which was completely different from RNN.

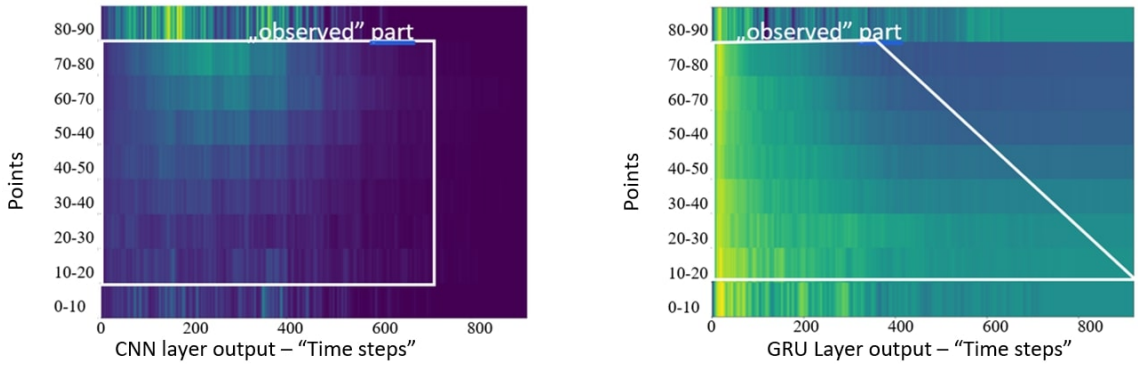


Figure 6.6: Representations over time from CNNs and GRUs layers. Each row corresponds to the predicted student result group from CNN and GRU at each timestep. Each grid from the column corresponds to each dimension of the current sequence step representation. I examined only that part of the heatmap, where the data was not constant, or not too uniformly distributed. The brighter color indicates high activation at the output of the layer of my neural network, while the dark means weak activation.

6.6.3 User behavior clustering

Further studies were conducted in order to identify the different learning strategies and examine whether they appeared in the data sequence. I investigated the best and worst 20% of student groups. I performed a cluster analysis (Kmeans, n_clusters = 2, algorithm=Elkan), utilizing the hidden vector representation learnt by my CNN and GRU models. The clustering was based on the cosine similarity of the output 50-dimensional vectors of the CNN and the GRU layers. As an interpretation of the clusters, the features introduced in Section 6.3 were accumulated from the cluster members. Figures 6.7 and **??** shows the boxplots of the key features by clusters.

The results of the best and worst 20% clustering show that there are two different clusters among both the top and the worst-performing students. The first group (marked in blue) watched significantly fewer videos (`rdnVideo`) than the others while achieving the same result. The feature values describing the interaction between the users and videos (`numoplay_video`, `numostop_video`, `numopause_video`, `numoseek_video`) also underpinned this observation. My clickstream level raw data-driven results were in line with educational/pedagogical results. For example, Mozhaeva et. al.

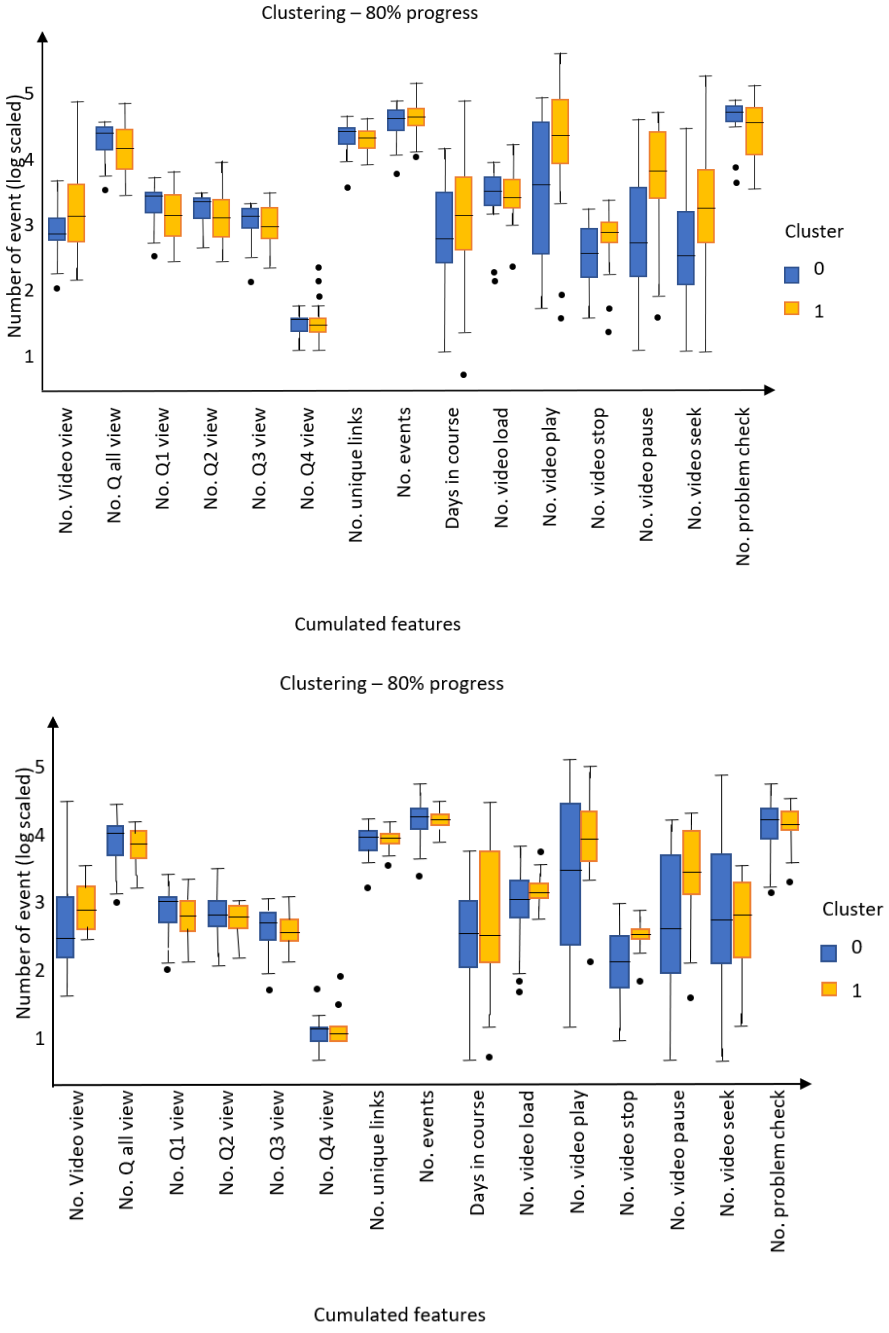


Figure 6.7: Cluster analysis of the group of 20% - 20% students who achieved best (top) and worst (bottom) final student scores during the course. The blue and brown colors show the different clusters in the observed group.

(Mozhaeva et al., 2020) sought to understand the behavior patterns of learners in MOOC courses and they found that at the very base level, there were "All-rounders" and "Viewers", the terminology being similar to the results of my unsupervised clustering analysis: users marked blue appeared to be "All-rounders".
The blue cluster members completed most assignments, watched all video lectures, and had numerous interactions with the videos, unlike "Viewers" (brown cluster), who watched almost all video lectures, yet hardly ever made any greater effort than absolutely necessary to complete the course. This data-driven interpretation of MOOC log data was a promising direction for educational data mining, as I was able to show sociological-pedagogical results using only raw logline data, which has not been seen before.

6.7 Contributions

Existing Deep Learning based time series prediction models can handle both continuous and discrete-valued sequences. While the prediction on continuous time series with univariate and multivariate sequences appeared to be a solved problem, the multivariate discrete-valued time series tasks have so far been studied by only few researchers. On the other hand, there are many successful solutions for fixed-size univariate discrete-valued problems, e.g., user search or behavior events classification, but to the best of my knowledge there are no Deep Learning interpretation papers about multivariate variable-size sequences prediction methods operating on low-level MOOC datasets.
The contributions of this thesis include the following (Kőrösi and Farkas, 2021):

- Empirical results showed that the embedding method was able to significantly outperform one-hot-encoding and provide an effective aid in the unsupervised preprocessing of discrete-valued sequence.
- In order to better understand the learned models, I performed three visual checks to illustrate the averaged values of the loss functions for each layer. My investigation clearly showed the different learning methods between RNN and CNN models, and offered useful visualization for pedagogical analysis.
- Further, I comparatively evaluated GRU, LSTM and TCNN architectures along with classic Machine Learning on cumulative features.

Chapter 7

Summary

7.1 Summary in English

Many aspects of daily life, including the purchase of consumer durables or even education, have been moving into online space. When compared with the situation of a decade ago, when shopping and learning were still almost exclusively offline activities, not only the format, but also the jobs related to retail and education have gone through considerable changes, requiring quite a different skill set. Many of those methods and methodologies have become unusable in the online space. One the one hand, online presence mounted considerable challenges for business and education leaders alike, on the other hand, the use of user log data has also created new opportunities for data science professionals. Logged user online behavior data can aid in creating decision support systems that not only help the operator, but also improve the user experience. However, the effort it takes to preprocess or predict time series data is far from negligible. They can be applied in creating aggregated databases of log data of varying breadth and depth, called user profiles. Moreover, raw data can also be used in whole or in part for classification, regression or even clustering. Research on the analysis of user log data started approximately 15 years ago. Traditional feature extraction and Machine Learning methods have been replaced in recent years by Deep Learning methods, which can provide high quality solutions for large amounts of data, even from low-level data. An as yet unexplored area within this novel approach is the construction and analysis of predictive models from discrete-valued sequences. In this dissertation, the author explored the different aspects of the topic of user online behavior through the analysis of various web-shop and MOOC log datasets.

7.1.1 User behavior analysis from high-level log data

In the first thesis point, the author presented the special challenges of log data collection and preparation on high-level log databases. He dealt with forecasting results on a real-life Hungarian

webshop database, which was developed and launched as a collaboration of the University of Szeged and a Hungarian company (Kőrösi and Vinkó, 2021). Apart from data collection and preparation solutions, he proposed application-specific feature sets. Further, He presented several comparative Machine Learning experiments. The proposed problem and its solution to predict acceptance of the sales promotion were significant, since He did not predict the next purchase but, in fact, the buyer's reaction to advertising letters. His goal was to predict whether or not a given user was likely to act upon a received sales promotion, which was a binary classification problem. The model achieved a considerable level of accuracy based on the first four purchases and high-level sequences data.

7.1.2 Educational performance prediction from mid-level click-stream data

The time series structure and analysis of e-commerce and MOOC platforms are very similar. However, during the analysis of the high-level webshop log data used in Chapter 3, the author recognized that the short and high-level log data limited the accuracy of his model. This meant that, in order to build models with higher accuracy, He first needed to change the depth of the data. To obtain better and deeper data, He designed an e-learning course on a Moodle site (Conscious and Safe Internet Usage - Tudatos és biztonságos internethasználat alapjai TÉBIA) and collaboration of two departments of the University of Szeged developed a middle-level user behavior logging component to collect the users' online activities (Kőrösi and Havasi, 2017; Kőrösi et al., 2018). In that study He analyzed the log data of pupils and students who were motivated by their teachers and schools to attend and complete the short (few-day-long) MOOC course. The main contribution of his investigation was that He managed to confirm that deeper middle-level data can support a more accurate model even when using short MOOC courses. The author also highlighted the features which influenced the classifier results the most, hence providing useful insights for MOOC developers. Based on the implemented methods, He described which the most notable features in his prediction models were. The chapter offers a detailed statistical methodology for predicting student performance based on log data which was created in short MOOCs and led by the teacher. Based on these datasets, the author determined that classic Machine Learning models were successful and they were influenced by several strong features. The accuracy of the models achieved a satisfactory accuracy of more than 80%.

7.1.3 MOOC performance prediction by Deep Learning from raw click-stream data

In terms of low-level log data collection in the form of clickstream or social network measures, the MOOC systems offer a treasure trove of data. We can design efficient online user (student) models which would then serve as a forecasting tool for estimating how many students were likely to drop

out, or preferably, complete the course. This was made possible by extensive research into comprehending and, hopefully, increasing the registration and completion rate, ultimately contributing to a better all-round learning experience in MOOCs.
Several studies showed that low-level data can be used to create more successful prediction models. To demonstrate this, conducted experiments using data from Stanford Lagunita's datasets to predict learner behavior using Deep Learning models on low-level data, and the author comparatively evaluated traditional and Deep Learning models.
To better understand the obtained results, the author performed a recurrent Neural Network for solving the outcome performance prediction problem in an online learning platform. The main contribution of that research part was the building of a prediction model which could use raw low-level datasets, and obtain the same or better results than regular prediction models. The key advantage of the model was that there was no need for manual feature engineering, because it could be automatically extracted from the raw log-line level records. Therefore, this approach could save a lot of time and human effort, and ignore the possible inconsistencies introduced by the hand-made process. Experimental results on Stanford Lagunita's dataset consisting of data by 12015 students showed that the expected model achieved significantly better than the baseline models. The results for the model were sufficient to demonstrate the feasibility of using recurrent Neural Networks when large datasets were available.

7.1.4 Deep learning models and interpretations for MOOC performance prediction

The majority of the time series Deep Learning models were applied to numerical data, event logs, such as the clickstream-level MOOC data used, consisting of multivariate discrete-valued sequences. Hence, time series Deep Learning techniques could not be directly applied. However, most of the discrete-valued sequence prediction solutions have been published for Natural Language Processing. The raw event logs were significantly longer than natural language sentences, with their varying lengths, thus NLP techniques could not be applied directly. To handle these special characteristics of the given clickstream-level MOOC dataset, the author proposed an embedding based Deep Learning model architecture (Kőrösi and Farkas, 2021). In this part of the research He trained state-of-art RNN and CNN models to predict the outcome scores of the students at the MOOC. He conducted experiments using various embedding layers to represent the multivariate discrete-valued data. Recurrent and Temporal Convolutional Neural Networks provided accurate forecasts without having any access to explicit knowledge about the investigated system. Yet, Deep Learning methods are typically considered 'black boxes', where it is almost impossible to fully understand based on what, why, and how RNN and CNN make forecasting decisions. His research aimed to open the black boxes of RNNs and CNNs trained for time series regression. The offered three visualization techniques which could support domain-expert users in interpreting discrete-valued multivariate time

series regression neural models. Moreover, He analyzed the online behavior of Massive Online Open Course (MOOC) students and introduced a Deep Learning architecture to predict the outcome score of the students at a MOOC.

7.1.5 Contributions of the thesis

The overall research work for this doctoral dissertation was made up of four separate studies and numerous subtasks, described in detail in Chapters 3-6.

The contributions of the **first group of studies** were published in Kőrösi and Vinkó (2021). Detailed discussion can be found in Chapter 3.

I/1. The author used the log data of an existing webshop in Hungary to develop a solution that could reliably predict the sales promotion acceptance probability from the high-level user log data. He proposed a specific feature representation that contained cumulative data from the obtained user sequences that effectively supported the operation of the model, which was designed as a combined classification and regression solution.

I/2. In the combined model the classification method aimed to determine whether a user would accept the sales promotion or not. Using the output of this classification model with a regression task, the author separately predicted the probability of the promotional package to be accepted. The output of this combined model was not only able to predict user behavior with considerable efficiency, but also to provide a solution that was easy for the client to interpret.

I/3. He performed empirical measurements with almost a dozen different Machine Learning methods, and ran hyper-parameter tuning to find the optimal solution. He demonstrated that when using high-level log data, the cumulative feature extraction method with a combined classification and regression solution could provide fast and effective results, which was confirmed by the customer's satisfaction.

The contributions of the **second group of studies** are related to the publications Kőrösi and Havasi (2017); Kőrösi et al. (2018). Detailed discussion can be found in Chapter 4.

II/1. The author was able to add a functioning logging system to a Moodle platform with weak tools of analyses, which would be useful for similar portals to live up to current measuring requirements.

II/2. He designed a data engineering solution that would automatically process input data without human intervention and which could intervene if extreme values emerged. He defined 263 features to describe the mid-level clickstream sequence of short video MOOCs.

II/3. Despite a relatively low sample size, He was able to render clickstream based predictive algorithms. He introduced a Machine Learning methodology for feature selection and binary

classification techniques with leave-one-out cross validation for short video MOOCs based on mid-level sequence data. The primary goal was to make binary prediction of course completion. The created models were capable of predicting who would "Fail" or "Complete" an online course, which would be an immense help for the faculties that provide e-learning courses.

II/4. He implemented and tested more than ten Machine Learning approaches, the most efficient tools were the Random Forest and Bagging achieving approximately 80% accuracy

The contributions of the **third group of studies** are related to the work of Kőrösi and Farkas (2020). Detailed discussion can be found in Chapter 5.

III/1. The author proposed a data preparation step for low-level clickstream event data. On the 3D tensor, He was able to effectively run RNN experiments.

III/2. . he built a baseline pipeline and Deep Learning model to predict student outcome as a regression and multi class classification problem. He evaluated the Deep Learning based prediction pipeline which outperformed the classic Machine Learning based solution.

Last, but not least, the contributions of the **forth group of studies** are related to the paper Kőrösi and Farkas (2021). Detailed discussion can be found in Chapter 6.

IV/1. Empirical results showed that the embedding method was able to significantly improve his forecasting results and provide an effective aid in the preprocessing of discrete-valued sequence.

IV/2. The author also comparatively evaluated GRU, LSTM and TCNN architectures along with classic Machine Learning on cumulative features.

IV/3. To better understand the results, He performed three visual inspections of the deep learnt models. His investigation clearly showed the different learning methods between RNN and CNN models, and offered useful visualization for pedagogical analysis.

7.2 Magyar nyelvű összefoglaló

7.2.1 Bevezető

Mind a vásárlásról és a tanulásról is elmondható, hogy néhány évvel ezelőtt, még szinte teljesen kizárólag a hagyományos offline formátumban zajlottak, addig ez mára jelentősen megváltozott. Ez a változás új kihívások elé állítja a területen dolgozó szakembereket, hiszen a legtöbb hagyományos módszer és módszertanok nagy része teljesen elavulttá és működésképtelenné vált az online térben. Ezek a szerepekörök eltűnése azonban nem maradt észrevétlen, mivel számos online vállalkozás küzd a csökkenő ügyfélszámmal, és az online tanulási rendszerek is csupán gyenge hatékonysággal képesek működni.

Bár tagadhatatlan, hogy az online jelenlét jelentős kihívásokat teremtett az üzleti és oktatási vezetők számára, ugyanakkor új lehetőségeket is megnyitott. A különböző online platformok rengeteg naplóadattal rendelkeznek, mely teret biztosít az adattudományi szakemberek bevonására, akik képesek gépi tanuláson alapuló online felhasználói viselkedés elemzése. Az elemzések egyik legfontosabb eszköze maguk log adatokat, melyek többfélék lehetnek. Mélységük szerint három szinten lehet őket elkülöníteni:

- **Magas szintű adat**: A legegyszerűbb magasszintű hozzáférésű naplóadatok a felhasználók vásárlásait, a kosár ideiglenes- és végleges tartalmát foglalják magukba, vagy oktatási platformok esetében a videókkal, oktatóanyagokkal és kvízekkel való interakciókat tartalmazzák.
- **Középszintű adat**: Az előző besorolásnál mélyebb, mely az többek között az oldalon eltöltött idő és az elemek sorrendjére vonatkozó információkat is tartalmaznak.
- **Alacsony szintű adat**: Az előbbi adatoknál is mélyebbre hatolnak és képesek további interakciókat is eltárolni, ilyenek lehetnek az egérkattintások és mozgások, billentyűzetnyomási szokások.

A különböző mélységű naplóadatok és gépi tanuló algoritmusok felhasználásával olyan döntéstámogató rendszereket lehet létrehozni, amelyek az üzemeltető segítése mellett a felhasználói élményt is javíthatják. A szerző munkájában e három adattípus segítségével próbálja bemutatni gépi tanuláson alapuló online felhasználói viselkedés elemzésének módszertanát.

Ez a kutatási terület nem újkeletű, hiszen a felhasználói naplóadatok elemzésével kapcsolatban már több, mint tizenöt éve folyik aktív kutatás. A szakirodalmi áttekintése alapján megállapítható, hogy az egyik leggyakoribb megoldás az, amelyben magas-középszintű naplóadatok felhasználásával aggregált adatbázisokat, úgynevezett felhasználói profilokat hoznak létre. Ezeket később osztályozásra, regresszióra vagy akár klaszterezésre használják fel. S bár a kutatások régóta folynak, ám az utóbbi időben a felhasznált módszerek valamelyest megváltoztak. Még kezdetben a kutatás legnagyobb hányada kumulált, előfeldolgozott adatokkal dolgozott, mára új technológiai megoldások

kerültek előtérbe. E fejlődésnek köszönhetően a korábban hagyományos jellemzők kinyeréseit és a gépi tanulási módszereket mára felváltották a Deep Learning-en alapuló megoldások. Ezek egyik nagy ígérete, hogy a nagy mennyiségű adatra is képesek magas minőségű megoldásokat nyújtani, és kepések akár nyers alacsony szintű adatokból kiindulva is dolgozni.

A disszertáció összesen 7 fejezetet tartalmaz, amelyek a fent említett megközelítéseket felhasználó különálló tanulmányokból állnak. Az első két fejezetben a szerző bemutatja a naplóadatok gyűjtésének és előkészítésének speciális kihívásait a magas szintű naplóadatbázisokon. Továbbá bemutatja az előjelzési megoldásainak eredményeit egy valós magyar webshop adatbázisán. Emellett betekintést enged a Szegedi Tudományegyetem két tanszékének együttműködéseként kifejlesztett és elindított "Tudatos és biztonságos internethasználat" (TÉBIA) című MOOC kurzusán végzett kísérleteibe.

A disszertáció az első két tézispontban alkalmazott adatgyűjtésen és a megoldások megfogalmazásán túl, a következő két fejezetben javaslatot tesz az alkalmazásspecifikus jellemzőkészletekre is. A szerző kiértékelési adatbázisokon keresztül több összehasonlító gépi tanulási kísérletet mutat be, melyben a jellemzőtér tervezési feladatok alapján arra összpontosította erőfeszítéseit, hogy Deep Learning algoritmusok segítségével közvetlenül alacsony szintű naplóadatokból úgynevezett end-to-end rendszereket lehessen építeni.Ezekben a megoldási javaslatokban a szerző a szekvenciális adatokat csupán minimális adatfeldolgozásnak vetette alá, majd a nyers adatokon sikeresen alkalmazott különböző neurális hálózati architektúrákat. Ehhez a Stanford Egyetem MOOC kurzusában résztvevő 142 395 diák naplóadatait használta fel. A kutatás eredményei megmutatták, hogy a Deep Learning modellek nyers adatokon történő alkalmazása pontosságban felülmúlták a klasszikus, jellemző-kivonáson alapuló Machine Learning módszereket. S bár a Deep Learning megoldások sikeressége kétségtelen, azonban fekete doboz jellegük akadályozzák azok valós alkalmazását. Ennek a problémának a feloldására a szerző az utolsó fejezetben három vizualizációs módszert javasolt, amelyek hozzájárulhatnak ahhoz, hogy a szakértők jobban megértsék a Deep Learning modellek működési mechanizmusait.

7.2.2 A disszertáció felépítése

A disszertációban a szerző az online felhasználói viselkedés modellezési technikákat és több empirikus kísérletet kívánt bemutatni. Az 1. fejezetben irodalmi áttekintésen keresztül összefoglalja a webshopok és MOOC oldalak alkalmazási területét és kutatási kihívásait. Különös figyelmet fordított a területen alkalmazott különböző klasszikus gépi tanulási és mélytanulási technikák összefoglalására, valamint az adatfeldolgozási és jellemzőkinyerési megoldások vizsgálatára. Egy magyarországi székhelyű, az EU határain belül működő webshop naplóadatainak felhasználásával egy tényleges üzleti megoldást hozott létre, melyben célzott ajánlatok és promóciós küldemények készítésihez épített egy modellt.

A kutatás további lépéseit a 2. fejezetben mutatja be, melyben a Szegedi Tudományegyetem

Szoftvertechnika Tanszékével együttműködve létrehozott egy MOOC kurzust "Tudatos és biztonságos internethasználat" címmel. Az így létrehozott rendszer rögzítette a felhasználók viselkedését a feladatok elvégzése közbe (középszintű naplóadatok), illetve rögzítette az egérmozgásokat és a videónézési napló adatokat is (alacsony szintű naplóadatok). Az így kapott, lényegesen gazdagabb közép- és alacsony szintű adathalmazokon előfeldolgozási és jellemző-kivonási módszereket javasolt, majd megvizsgálta a hagyományos gépi tanulási modellek hatékonyságát. Az eredmények rávilágítottak az új adathalmaz a hiányosságaira és a bevezetett módszerek hátrányaira, ugyanakkor igazolták a kezdeti hipotézist, miszerint a közép- és magas szintű naplóadatok, valamint gazdagabb és hosszabb idősort használó módszerek nagyobb hatékonyságot biztosítanak.

A disszertáció utolsó két fejezetében a szerző a Stanford Egyetem egyik MOOC kurzusára beiratkozott 142 395 hallgató 39,5 millió naplóadatának felhasználásával mutatja be a hallgatói teljesítményt előjelző modelleket. A nehéz és időigényes jellemzőtér-kinyerés helyett a nyers, kattintásfolyamszintű, diszkrét értékű, változó hosszúságú adatokat kezelni tudó Deep Learning módszereket vizsgálta meg. Ebben a fejezetben a szerző arra a kérdésre keresi a választ, hogy vajon az ilyen nagyméretű képzési adathalmaz esetén ezek a módszerek lényegesen pontosabbak-e, mint a klasszikus, jellemzőkinyerésen alapuló módszerek. Az utolsó fejezetben a szerző összehasonlítja a konvolúciós- és a rekurrens neurális hálózati architektúrákat a Stanford MOOC-adatkészleten. Továbbá betekintést nyújt a numerikus- és a diszkrét szekvenciális adatfeldolgozási technikákba, ahol változókénti beágyazásokat vizsgál. Végül három vizualizációs technikát is javasolt a szekvenciális naplóadatokon képzett mély neurális hálózatokhoz, melynek célja, hogy segítséget nyújtson a neurális modellek által tanult minták megértésében.

7.2.3 Felhasználói viselkedéselemzés magas szintű naplóadatokból

Az első tézispontban a szerző bemutatja a magas szintű naplóadatok gyűjtésének és előkészítésének speciális kihívásait. Ehhez, a kutatásban, egy valós magyar webshop adatbázisán végzett előjelzési feladattal foglalkozott (Kőrösi and Vinkó, 2021). Az adatgyűjtési és adatelőkészítési megoldások mellett alkalmazásspecifikus jellemzőkészleteket javasolt. Továbbá bemutatott több összehasonlító Machine Lear- ning kísérletet. Javaslatot tett egy probléma megoldására, melyben a célzott ajánlatok és promóciós küldemények elfogadásának előrejelzésé volt a cél. A kutatásban nem a következő vásárlást, hanem valójában a vevő reklámlevelekre adott reakcióját jelezte előre.

7.2.4 Oktatási teljesítmény előrejelzése középszintű kattintásfolyam adataiból

A 1. fejezetben használt magas szintű webshop naplóadatok elemzése során a sikerült egy működő modellt alkotni. Ám ugyanakkor az is világossá vált, hogy a rövid- és magasszintű naplóadatok korlátozzák a modell pontosságát. Ez egyben azt is jelenti, hogy a nagyobb pontosságú modellek építéséhez először meg kellett változtatni az adatok mélységét. Mivel az e-kereskedelmi és a MOOC-platformok idősorszerkezete és annak elemzése nagy hasonlóságot mutat, így kutatás a

MOOC naplóadatok irányába fordult. A szerző jobb és mélyebb adatok megszerzése érdekében egy e-learning tanfolyamot tervezett, melynek alapja egy Moodle oldal volt (Tudatos és biztonságos internethasználat alapjai TÉBIA), és ehhez a Szegedi Tudományegyetem két tanszékének együttműködésével kifejlesztettek egy új, viselkedésnaplózási komponenst (Kőrösi és Havasi, 2017; Kőrösi et al., 2018). Ez a tanulmány a komponens által rögzített adatokkal, annak megértésével és feldolgozásával foglalkozott. A kutatás egyik jelentős hozzájárulása az volt, hogy sikerült megerősíteni azt a feltevést, hogy a mélyebb, gazdagabb adatsorok még rövid MOOC-tanfolyamok esetén is pontosabb modellt hoznak létre, mint a rövid és magas szintű naplóadatokra épülő megoldások. A szerző emellett kiemelte azokat a jellemzőket, amelyek a leginkább befolyásolták az osztályozó modell eredményeit, amely hasznos meglátásokkal szolgál a MOOC-fejlesztők számára. Továbbá feltárta, hogy az adatfolyam feletti jellemzőválasztás során mely jellemzők voltak a legnagyobb súlyúak.

7.2.5 MOOC teljesítmény előrejelzés Deep Learning segítségével a nyers kattintásfolyam adatokból

Számos tanulmány kimutatta, hogy az alacsonyszintű adatok felhasználásával a sikeresebb előjelzési modelleket hozhatunk létre, mint a magas- vagy középszintű adatokkal. Ennek bizonyítására a szerző a Stanford Lagunita adathalmazainak felhasználásával végzett kísérleteket, melynek célja a tanulói viselkedés előrejelzésére volt. Ehhez nyers adatsorozatokat és Deep Learning modelleket használt fel, melyek eredményét összehasonlította a hagyományos módszerekkel. E kutatási rész fő hozzájárulása egy olyan előjelző modell felépítése volt, amely képes volt nyers, alacsony szintű, változó hosszúságú, diszkrét értékű adatok felhasználásával a hagyományos előjelző modellekkel azonos vagy jobb eredményeket elérni. A modell legfőbb előnye, hogy ennél a megközelítésnél nem volt szükség manuális jellemzőtervezésre, hiszen a nyers naplóadatokból a modell ezt automatikusan ki tudta nyerni. A Stanford Lagunita 12 015 hallgató adataiból álló adatállományán végzett kísérleti eredmények azt mutatták, hogy a várt modell jelentősen jobb eredményt ért el, mint az alapmodellek. A modell eredményei elegendőek voltak ahhoz, hogy bizonyítsák a rekurrens neurális hálózatok használatának létjogosultságát, a nyers, alacsony szintű, változó hosszúságú, diszkrét értékű adatsorozatokon.

7.2.6 Mélytanulási modellek és azok értelmezése a MOOC teljesítmény előrejelzéséhez

A Deep Learning modellek többségét numerikus adatokra alkalmazzuk, azonban a MOOC eseménynaplók gyakran többváltozós, változó hosszúságú, diszkrét értékű szekvenciákból állnak. Lévén, hogy a Deep Learning technikákat nem lehetett közvetlenül ilyen adatokra alkalmazni, egy új megközelítéssel kellett előállni. Ennek megoldására már ismert módszertannal rendelkezünk, melyet a természetes nyelvfeldolgozáshoz (NLP) kapcsolódóan publikáltak. S bár a módszer hatékonyan működik az NLP

esetén, a MOOC log adatsorozatok jelentősen hosszabbak voltak, mint a természetes nyelvi mondatok, és gyakran több változóból állnak és változó hosszúságúak is, így az az eddig bevált technikákat nem lehetett közvetlenül alkalmazni. Ennek megoldására a szerző a nyers logadatokon alapuló, azok speciális jellemzőinek kezelésére egy beágyazás alapú Deep Learning modellarchitektúrát javasolt (Kőrösi és Farkas, 2021), melyhez korszerű rekurrens és konvolúciós modelleket képzett. A kísérletekben sikerült különböző beágyazási rétegek használatával a többváltozós diszkrét értékű adatokat megfelelően reprezentálni. Ennek az egyik bizonyitéka, hogy rekurrens és temporális konvolúciós neurális hálózatok pontos előrejelzéseket biztosítottak anélkül, hogy a vizsgált rendszerre vonatkozó explicit tudáshoz hozzáfértünk volna. A szerző a sikeres modell építés mellett többek által "fekete dobozoknak" tekintett, Deep Learning modellek dobozainak felnyitására is javaslatot tett. Munkájában javasol három olyan vizualizációs technikát, amelyek hathatós segítséget nyújthatnak a diszkrét értékű, többváltozós, idősoros regressziós neurális modellek értelmezésében. Kőrösi and Farkas (2021) publikációjához kapcsolódóan a szerző a következő eredményeket tekinti a területhez való jelentős hozzájárulásának:

A disszertáció tézisei

A doktori disszertáció teljes kutatási munkája négy különálló tanulmányból és számos részfeladatból állt, amelyeket a szerző a 3-6. fejezetben részletesen ismertetett.
Az **első téziscsoport** a hozzájárulásai a Kőrösi and Vinkó (2021) publikációhoz kapcsolódnak. A részletes bemutatás a 3. fejezetben található.

I/1. A szerző egy létező magyarországi webáruház naplóadatait használta fel egy olyan megoldás kidolgozására, amely a magasszintű felhasználói naplóadatokból megbízhatóan képes volt előjelezni az eladási promóció elfogadásának valószínűségét. Javasolt egy speciális jellemzőreprezentációt, amely hatékonyan támogatta a modell működését, mely modellt egy kombinált osztályozási és regressziós megoldásként mutatott be.

I/2. A kombinált modellben az osztályozási feladat célja annak meghatározása volt, hogy egy felhasználó elfogadja-e az értékesítési promóciót vagy sem. Ennek az osztályozási modellnek a kimenetét felhasználva egy további regressziós megoldást használva, külön-külön megjósolta egy-egy promóciós csomag elfogadásának valószínűségét. Az így kapott kombinált modell nemcsak a felhasználói viselkedést tudta jelentős hatékonysággal megjósolni, hanem az ügyfél számára könnyen értelmezhető megoldást is nyújtott.

I/3. A szerző közel egy tucat különböző Machine Learning módszerrel végzett empirikus méréseket, valamint hiperparaméter-hangolást futtatott az optimális paraméterek megtalálásához. Bemutatta, hogy a magas szintű naplóadatok használata esetén a kumulatív jellemző-kivonási módszer egy kombinált osztályozási és regressziós megoldással gyors és hatékony eredményeket

tudott nyújtani, amit az ügyfél elégedettsége is megerősített.

A **második téziscsoport** a hozzájárulásai a Kőrösi and Havasi (2017);Kőrösi et al. (2018) publikációkhoz kapcsolódnak. A részletes bemutatás a 4. fejezetben található.

II/1. A szerző egy gyenge elemzési eszközökkel működő Moodle platformot, egy hatékony naplózási rendszerrel tudott kiegészíteni, amely akár a hasonló portálok számára is hasznos lehet, hiszen ezzel megfelelnének a jelenkor adattudományi adat rögzítési követelményeinek.

II/2. Olyan adatmérnöki megoldást tervezett, amely emberi beavatkozás nélkül dolgozza fel a bemeneti adatokat. Képes volt feltárni a szélsőséges felhasználói viselkedési értékeket, mely a hathatós segítséget nyújthat a veszélyeztetett diákcsoportok felkutatásában. A kutatásban összesen 263 egyedi jellemzőt határozott meg, mely a rövid videós MOOC-ok középszintű kattintássorozatának leírására alkalmazható

II/3. A viszonylag alacsony mintanagyság ellenére képes volt hatékony clickstream alapú előjelző modellt készíteni. Bevezetett egy gépi tanulási módszertant, amely a jellemzők kiválasztásában és a bináris osztályozási technikákat alkalmazásában segít az olyan rövid videó MOOC-oknál melyek középszintű szekvencia-adattal rendelkeznek. A létrehozott modellek képesek voltak megjósolni, hogy ki fog megbukni vagy ki fogja befejezni az online kurzust, ami hatalmas segítséget jelent az e-learning kurzusokat működtető tanszékek számára

II/4. Több mint tíz gépi tanulási megközelítést alkalmazott és tesztelt, melyek eredményről részletes statisztikát mutatott be.

A **harmadik téziscsoport** hozzájárulásai a Kőrösi and Farkas (2020) publikációhoz kapcsolódnak. Részletes bemutatás a 5. fejezetben található.

III/1. A szerző javaslatot tett egy adatelőkészítési módszertanra az alacsonyszintű clickstream eseményadatok kezeléséhez. Ennek eredményeként 3D adatokkal hatékonyan tudott Rekurrens Hallózótokat felhasználó kísérleteket futtatni. .

III/2. Felépített egy Deep Learning modellen alapuló módszertant melyben klasszifikáció és regresszió segítségével jelzi előre a diákok tanulmányi eredményét. Kutatásában kiértékelte a javasolt Deep Learning alapú előrejelző módszertant, amely az eredmények alapján felülmúlta a klasszikus Machine Learning alapú megközelítést.

A **negyedik téziscsoport** hozzájárulásai a Kőrösi and Farkas (2021) publikációhoz kapcsolódnak. Részletes bemutatás a 6. fejezetben található.

VI/1. Az empirikus eredmények azt mutatták, hogy a beágyazási módszere képes volt jelentősen javítani az előjelzési eredményeit, és hatékony segítséget nyújtott a többváltozós, diszkrét értékű, változó hosszúságú szekvenciális adatok előfeldolgozásában.

VI/2. Emellett összehasonlítóan értékelte a GRU, LSTM és TCNN architektúrákat, valamint a klasszikus gépi tanulási megoldásokat a kumulatív jellemzőkkel.

VI/3. Az eredmények jobb megértése érdekében a szerző három vizuális megoldást mutatott be, mellyel betekintést nyerhettünk a mély tanuló modellekbe. Vizsgálata egyértelműen megmutatta az RNN és CNN modellek közötti eltérő tanulási módszereket.

Bibliography

J. Kleinberg A. Anderson, D. Huttenlocher and J. Leskovec. 2014. Engaging with Massive Online Courses. *Proceedings of the 23rd international conference on World wide web*, pages 687–698.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Chrisgone O Adede. 2012. *Model for predicting the probability of event occurrence using logistic regression: case for a Kenyan commercial bank*. Ph.D. thesis, University of Nairobi.

Amr Ahmed, Yucheng Low, Mohamed Aly, Vanja Josifovski and Alexander J Smola. 2011. Scalable distributed inference of dynamic user interests for behavioral targeting. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 114–122.

Raghad Al-Shabandar, Abir Hussain, Andy Laws, Robert Keight, Janet Lunn and Naeem Radi. 2017. Machine learning approaches to predict learning outcomes in massive open online courses. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 713–720. IEEE.

Hanan Aldowah, Hosam Al-Samarraie and Wan Mohamad Fauzy. 2019. Educational data mining and learning analytics for 21st century higher education: A review and synthesis. *Telematics and Informatics*, 37:13–49.

Mohamed Aly, Andrew Hatch, Vanja Josifovski and Vijay K Narayanan. 2012. Web-scale user modeling for targeting. In *Proceedings of the 21st international conference on world wide web*, pages 3–12.

Mingxiao An and Sundong Kim. 2020. Neural user embedding from browsing events. In *ECML/PKDD (4)*, pages 175–191.

Ehsaneddin Asgari and Mohammad RK Mofrad. 2015. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

R. Baker. 2010. Data Mining for Educatio. *International Encyclopedia of Education*, 7:112–118.

Rachel Baker, Di Xu, Jihyun Park, Renzhe Yu, Qiujie Li, Bianca Cung, Christian Fischer, Fernando Rodriguez, Mark Warschauer and Padhraic Smyth. 2020. The benefits and caveats of using clickstream data to understand student self-regulatory behaviors: opening the black box of learning processes. *International Journal of Educational Technology in Higher Education*, 17(1):1–24.

Ryan Shaun Baker and Paul Salvador Inventado. 2014. Educational data mining and learning analytics. In *Learning analytics*, pages 61–75. Springer.

Arindam Banerjee and Joydeep Ghosh. 2001. Clickstream clustering using weighted longest common subsequences. In *Proceedings of the web mining workshop at the 1st SIAM conference on data mining*, volume 143, page 144. Citeseer.

Aysun Bozanta and Birgul Kutlu. 2018. Developing a contextually personalized hybrid recommender system. *Mobile information systems*, 2018.

L Breiman. 1999. Random forests-random features technical report 576. *Statistical Department, UC Berkeley, USA*.

Christopher G Brinton and Mung Chiang. 2015. Mooc performance prediction via clickstream data and social learning networks. In *2015 IEEE conference on computer communications (INFOCOM)*, pages 2299–2307. IEEE.

Andy Brown, Aaron Tuor, Brian Hutchinson and Nicole Nichols. 2018. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pages 1–8.

Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370.

M. Chiang C. G. Brinton, S. Buccapatnam and H. V. Poor. 2015. Mining MOOC Clickstreams: On the Relationship Between Learner Behavior and Performance. pages 10–22.

M. Chiang H. V. Poor C. G. Brinton, S. Buccapatnam. 2016. Mining MOOC Clickstreams: Video-Watching Behavior vs. In-Video Quiz Performance. *Signal Processing IEEE Transactions on*, 64:3677–3692.

S. Ha M. Chiang R. Smith C. G. Brinton, R. Rill and W. Ju. 2015. Individualization for Education at Scale: MIIC Design and Preliminary Evaluation. *Transactions on Learning Technologies*, 8(1):136–148.

Erion Çano and Maurizio Morisio. 2017. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Wei Chen, Zhendong Niu, Xiangyu Zhao and Yi Li. 2014. A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web*, 17(2):271–284.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

R Dennis Cook. 1977. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos and José Reis. 2009. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553.

X. Wu J. Miao J. Wang D. Liang, J. Jia. 2014. Analysis of learners' behaviors and learning outcomes in a massive open online course. *Knowledge Management & E-Learning*, 6(3):281–298.

John N Darroch and Douglas Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480.

Rahul Dey and Fathi M Salem. 2017. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE.

Min Du, Feifei Li, Guineng Zheng and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298.

David Essex. 2009. Matchmaker, matchmaker. *Communications of the ACM*, 52(5):16–17.

A. Valbuena F. Chacon, D. Spicer. 2012. Are Domain Specific Models Easier to Maintain Than UML Models? *Analytics in Support of Student Retention and Success (Research Bulletin 3, 2012 ed.)*, pages 1–9.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963.

Mi Fei and Dit-Yan Yeung. 2015. Temporal models for predicting student dropout in massive open online courses. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 256–263. IEEE.

Alan H Fielding. 1999. An introduction to machine learning methods. In *Machine learning methods for ecological applications*, pages 1–35. Springer.

Josh Gardner and Christopher Brooks. 2018. Student success prediction in moocs. *User Modeling and User-Adapted Interaction*, 28(2):127–203.

Gaurang Gavai, Kumar Sricharan, Dave Gunning, Rob Rolleston, John Hanley and Mudita Singhal. 2015. Detecting insider threat from enterprise social and online activity data. In *Proceedings of the 7th ACM CCS international workshop on managing insider security threats*, pages 13–20.

Pierre Geurts, Damien Ernst and Louis Wehenkel. 2006. Extremely randomized trees. *Machine learning*, 63(1):3–42.

David Goldberg, David Nichols, Brian M Oki and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1809–1818.

William Groves and Maria Gini. 2011. A regression model for predicting optimal purchase timing for airline tickets. Technical Report 11-025, University of Minnesota Digital Conservancy.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*.

Stephen Haggard. 2013. The maturing of the mooc: literature review of massive open online courses and other forms of online distance learning. *BIS research paper*, (130).

Jiazhen He, James Bailey, Benjamin Rubinstein and Rui Zhang. 2015. Identifying at-risk students in massive open online courses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1):69–82.

R. A. Huebner. 2013. A Survey of Educational Data-Mining Research. *Research in Higher Education Journal*, 19:1–19.

S. Klemmer J. Cheng, C. Kulkarni. 2013. Tools for predicting drop-off in large online classes. *Proceedings of the 2013 conference on Computer supported cooperative work companion*, pages 121–124.

J. Welsh J. Guan, W. Nunez. 2002. Institutional strategy and information support: the role of data warehousing in higher education. *Campus-Wide Information Systems*, 19(5):168–174.

H. Cen A. Cuneo E. Gouvea J. Mostow, J. Beck and C. Heiner. 2005. An educational data mining tool to browse tutor–student interactions: Time will tell! . *In Proceedings of the workshop on educational data mining*, pages 15–22.

Gareth Michael James. 1998. *Majority vote classifiers: theory and applications*. Stanford University.

Bekir Karlik and A Vehbi Olgac. 2011. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

Byung-Hak Kim, Ethan Vizitei and Varun Ganapathi. 2018. Gritnet: Student performance prediction with deep learning. *arXiv preprint arXiv:1804.07405*.

Dhananjay Kimothi, Akshay Soni, Pravesh Biyani and James M Hogan. 2016. Distributed representations for biological sequence analysis. *arXiv preprint arXiv:1608.05949*.

René F Kizilcec, Chris Piech and Emily Schneider. 2013. Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In *Proceedings of the third international conference on learning analytics and knowledge*, pages 170–179.

Marius Kloft, Felix Stiehler, Zhilin Zheng and Niels Pinkwart. 2014. Predicting mooc dropout over weeks using machine learning methods. In *Proceedings of the EMNLP 2014 workshop on analysis of large scale social interaction in MOOCs*, pages 60–65.

Dennis Koehn, Stefan Lessmann and Markus Schaal. 2020. Predicting online shopping behaviour from clickstream data using deep learning. *Expert Systems with Applications*, 150:113342.

Gábor Kőrösi, Péter Esztelecki, Richard Farkas and Krisztina Tóth. 2018. Clickstream-based outcome prediction in short video moocs. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5. IEEE.

Gábor Kőrösi and Richard Farkas. 2020. Mooc performance prediction by deep learning from raw clickstream data. In *International Conference on Advances in Computing and Data Sciences*, pages 474–485. Springer.

Gábor Kőrösi and Richárd Farkas. 2021. Deep learning models and interpretations for multivariate discrete-valued event sequence prediction. In *International Conference on Artificial Neural Networks*, pages 396–406. Springer.

Gábor Kőrösi and Ferenc Havasi. 2017. Moodle-based data mining potentials of mooc systems at the university of szeged. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 755–760. IEEE.

Gábor Kőrösi and Tamás Vinkó. 2021. A practical framework for real life webshop sales promotion targeting. *Informatica*, 45(4).

M. ROSSI L. Cao, B. RAMESH. 2009. Are Domain Specific Models Easier to Maintain Than UML Models? *IEEE Software*, 26:19–21.

Jack Lanchantin, Ritambhara Singh, Beilun Wang and Yanjun Qi. 2017. Deep motif dashboard: Visualizing and understanding genomic sequences using deep neural networks. In *Pacific Symposium on Biocomputing 2017*, pages 254–265. World Scientific.

Center for Advanced Research On Learning. CAROL learner mooc data set. `http://datastage.stanford.edu`.

Jeong Min Lee and Milos Hauskrecht. 2019. Recent context-aware lstm for clinical event time-series prediction. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 13–23. Springer.

Qiujie Li, Rachel Baker and Mark Warschauer. 2020. Using clickstream data to measure, understand, and support self-regulated learning in online courses. *The Internet and Higher Education*, 45:100727.

Zhongliang Li, Raymond Kulhanek, Shaojun Wang, Yunxin Zhao and Shuang Wu. 2018. Slim embedding layers for recurrent neural language models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Guimei Liu, Tam T Nguyen, Gang Zhao, Wei Zha, Jianbo Yang, Jianneng Cao, Min Wu, Peilin Zhao and Wei Chen. 2016. Repeat buyer prediction for e-commerce. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164.

J. Luan. 2002. Data Mining and Knowledge Management in Higher Education – Potential Applications. *Paper presented at the Annual Forum for the Association for Institutional Research*, pages 1–13.

Joel P Lucas, Nuno Luz, María N Moreno, Ricardo Anacleto, Ana Almeida Figueiredo and Constantino Martins. 2013. A hybrid recommendation approach for a tourism system. *Expert systems with applications*, 40(9):3532–3550.

R. S. Baker M. Berland and P. Blikstein. 2014. Educational Data Mining and Learning Analytics: Applications to Constructionist Research. *Technology, Knowledge and Learning*, 19(1):205–220.

A. Labbi J. Sutanto M. Speiser, G. Antonini. 2012. On nested palindromes in clickstream data. *KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 18:1460–1468.

Andrés Martínez, Claudia Schmuck, Sergiy Pereverzyev Jr, Clemens Pirker and Markus Haltmeier. 2020. A machine learning framework for customer purchase prediction in the non-contractual setting. *European Journal of Operational Research*, 281(3):588–596.

Patrick Ng. 2017. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279*.

Ahmedbahaaaldin Ibrahem Ahmed Osman, Ali Najah Ahmed, Ming Fai Chow, Yuk Feng Huang and Ahmed El-Shafie. 2021. Extreme gradient boosting (xgboost) model to predict the groundwater levels in selangor malaysia. *Ain Shams Engineering Journal*, 12(2):1545–1556.

N. Namesztovszki L. Major P. Esztelecki, G. Korosi. 2016. The comparison of impact offline and online presentation on student achievements: A case study. *39th International Convention on Information and Communication Technology Electronics and Microelectronics (MIPRO)*, 39:802–806.

L. Kotthoff P. Romanski. 2017. Package 'fselector'. *IEEE Journal of Selected Topics in Signal Processing*, 11(5):716–728.

Trevor Park and George Casella. 2008. The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Charlotte Pelletier, Silvia Valero, Jordi Inglada, Nicolas Champion, Claire Marais Sicre and Gérard Dedieu. 2017. Effect of training class label noise on classification performances for land cover mapping with satellite image time series. *Remote Sensing*, 9(2):173.

Antoine Pigeau, Olivier Aubert and Yannick Prié. 2019. Success prediction in moocs: A case study. *International Educational Data Mining Society*.

S. Lakshmi Prabha and A.R.Mohamed Shanavas. 2014. Educational data mining applications. *Operations Research and Applications: An International Journal (ORAJ)*, 1(1):1–6.

C. Romero and S.Ventura. 2010. Educational Data Mining: A Review of the State of the Art. *Systems, Man, and Cybernetics Part C: Applications and Reviews*, 40(6):601–618.

Lamyaa Sadouk. 2019. Cnn approaches for time series classification. In *Time Series Analysis-Data, Methods, and Applications*, pages 1–23. IntechOpen.

Lamyaa Sadouk, Taoufiq Gadi and El Hassan Essoufi. 2018. A novel deep learning approach for recognizing stereotypical motor movements within and across subjects on the autism spectrum disorder. *Computational intelligence and neuroscience*, 2018.

Iqbal H Sarker. 2021. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21.

Sumit Sidana. 2018. *Recommendation systems for online advertising*. Ph.D. thesis, Université Grenoble Alpes.

N. Li T. Sinha, P. Jermann and P. Dillenbourg. 2014. Your click decides your fate: Inferring Information Processing and Attrition Behavior from MOOC Video Clickstream interactions. *EMNLP Workshop on Modelling Large Scale Social Interaction in Massive Open Online Courses*, pages 3–14.

Yonghong Tian, Bing Zheng, Yanfang Wang, Yue Zhang and Qi Wu. 2019. College library personalized recommendation system based on hybrid recommendation algorithm. *Procedia CIRP*, 83:490–494.

John W Tukey et al. 1977. *Exploratory data analysis*, volume 2. Reading, Mass.

Robin Van Meteren and Maarten Van Someren. 2000. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pages 47–56.

Vedita Velingker and Malony Alphonso. 2016. We recommend: Recommender system based on product reviews. *IJSTE International Journal of Science Technology & Engineering*, 2(12):333–337.

Armando Vieira. 2015. Predicting online user behaviour using deep learning algorithms. *arXiv preprint arXiv:1511.06247*.

M. Jan W. Klosgen, J. Zytkow. 2002. Knowledge discovery in databases: the purpose, necessity, and challenges. *Handbook of data mining and knowledge discovery.*, pages 1–9.

Andrew A Weiss. 1988. A comparison of ordinary least squares and least absolute error estimation. *Econometric Theory*, 4(3):517–527.

Jacob Whitehill, Kiran Mohan, Daniel Seaton, Yigal Rosen and Dustin Tingley. 2017a. Delving deeper into mooc student dropout prediction. *arXiv preprint arXiv:1702.06404*.

Jacob Whitehill, Kiran Mohan, Daniel Seaton, Yigal Rosen and Dustin Tingley. 2017b. Mooc dropout prediction: How to measure accuracy? In *Proceedings of the fourth (2017) acm conference on learning@ scale*, pages 161–164.

Jacob Whitehill, Joseph Williams, Glenn Lopez, Cody Coleman and Justin Reich. 2015. Beyond prediction: First steps toward automatic intervention in mooc student stopout. *Available at SSRN 2611750*.

Hadley Wickham, Winston Chang and Maintainer Hadley Wickham. 2016. Package 'ggplot2'. *Create Elegant Data Visualisations Using the Grammar of Graphics. Version*, 2(1):1–189.

Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82.

M. Wen K. R. Koedinger X. Wang, D. Yang and C. P. Rose. 2015. Investigating how student's cognitive behavior in MOOC discussion forum a ect learning gains. *In Proceedings of the EDM Conference,International Educational Data Mining Society (IEDMS)*, pages 226–233.

Wanli Xing and Dongping Du. 2019. Dropout prediction in moocs: Using deep learning for personalized intervention. *Journal of Educational Computing Research*, 57(3):547–570.

Feng Xiong, Kaifa Zou, Zhemin Liu and Hongzhi Wang. 2019. Predicting learning status in moocs using lstm. In *Proceedings of the ACM Turing Celebration Conference-China*, pages 1–5.

G. Kortemeyer S. Rayyan D. Seaton D. Pritchard Y. Bergner, S. Droschler. 2012. Model-Based Collaborative Filtering Analysis of Student Response Data: Machine-Learning Item Response Theory. *the International Conference on Educational Data Mining (EDM)*, 5(e):95–102.

Tsung-Yen Yang, Christopher G Brinton, Carlee Joe-Wong and Mung Chiang. 2017. Behavior-based grade prediction for moocs via time series neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(5):716–728.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh and Chris Dyer. 2015. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*, 9.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 729–732.

Ke Zhang, Jianwu Xu, Martin Renqiang Min, Guofei Jiang, Konstantinos Pelechrinis and Hui Zhang. 2016. Automated it system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1291–1300. IEEE.

Yongzheng Zhang and Marco Pennacchiotti. 2013. Predicting purchase behaviors from social media. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1521–1532.