BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

CLASSIFICATION ON UNBALANCED DATA

KLASIFIKACE NA NEVYVÁŽENÝCH DATECH

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR **MARTIN HLOSTA**
AUTOR PRÁCE

SUPERVISOR **Doc. Ing. JAROSLAV ZENDULKA, CSc.**
ŠKOLITEL

BRNO 2017

Abstract

This thesis is focused on classification on unbalanced data. It is an important part of machine learning with the objective to address the issues when one class is significantly underrepresented compared to the other one. The minority class is usually more important, and the traditional algorithms favouring the majority class may ignore the importance of the minority class. Two application domains motivated the research and identification of two specific problems of the imbalanced data.

First, the presence of a constraint on the performance of a minority class in the computer security domain resulted in the formulation of the constrained classification problem. I proposed a solution that combines the cost-sensitive logistic regression and stochastic algorithms, which in the conducted experiments always improved the performance of the logistic regression.

The domain of Learning Analytics motivated me to define a general prediction problem, whether a goal is has been achieved within the deadline. I designed the Self-Learning framework, in which models are trained by analysing attributes of objects that achieved the goal early in the investigated period. Because only a few objects satisfy the goal at the beginning, the problem is by its nature imbalanced, with the imbalance decreasing in time. The evaluation, performed on the task of identification of at-risk students in the distance higher education, showed (1) the predictive power compared the specified baseline models and (2) that methods for tackling the class imbalance without domain information didn't lead to significant improvements. When the domain information is utilised in the extended version of Self-Learning, the evaluation showed the performance increase. Understanding and exploiting the source of imbalance can also lead to better results.

Keywords

Data mining, classification, imbalanced data, machine learning, data mining with constraints, time-variant imbalance ratio.

Abstrakt

Tématem této disertační práce je klasifikace daty s nevyváženými daty. Jedná se o oblast strojového, jejímž cílem je řešit problémy, které plynou z toho, že jedna ze tříd je v datech zastoupena výrazně méně než třída druhá. Minoritní třída má často větší význam a tradiční metody upřednostňující majoritní třídu nedosahují dobrých výsledků na třídě minoritní. Dvě aplikační domény motivovaly výzkum a vedly na identifikaci dvou specifických, dosud neřešených problémů.

V první z nich vedlo omezení kladené na minimální požadovanou přesnost na minoritní třídě v počítačové bezpečnosti na formulaci úlohy klasifikace s omezením. Navrhl jsem metodu, která kombinuje upravenou verzi logistické regrese a stochastické algoritmy, které vždy vylepšily výsledky logistické regrese.

Druhou je doména analýzy učení (Learning Analytics), která motivovala definici problému predikce splnění cíle, jenž má specifikovaný termín splnění. Byl představen koncept sebe-učení (Self-Learning), kdy trénování modelu probíhá díky jedincům, kteří tento cíl splní předčasně. Díky malému počtu jedinců splňujících úlohu na začátku je problém silně nevyvážený, ale nevyváženost klesá směrem k termínu splnění. Na problému identifikace rizikových studentů distanční univerzity bylo ukázáno, že (1) takový koncept dává lepší výsledky než specifikovaná základna (baseline), (2) a že metody pro vypořádání se s nevyvážeností, které neberou v potaz informaci o doméně, nevedly k velkým zlepšením. Evaluace ukázala, že metody založené na znalosti domény v rozšířené verzi pro Self-Learning vylepšily klasifikaci více než běžné metody pro vypořádání se s nevyvážeností a že znalost příčiny nevyváženosti může vést k lepším výsledkům.

Klíčová slova

Dolování z dat, klasifikace, nevyvážená data, strojové učení, dolování z dat s omezením, změna nevyváženosti v čase.

Reference

HLOSTA, Martin. *Classification on unbalanced data.* Brno, 2017. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Zendulka Jaroslav.

Classification on unbalanced data

Declaration

Hereby I declare that this thesis was prepared as an original author's work under the supervision of Doc. Ing. Jaroslav Zendulka, CSc. The supplementary information was provided by Doc. Zdeněk Zdráhal, CSc. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Martin Hlosta
August 31, 2017

Acknowledgements

I would like to thank my PhD. supervisor, doc. Ing. Jaroslav Zendulka, CSc. for motivating and encouraging me to start the PhD research in data mining, and for his rich and ongoing help for all the years while I was working on the thesis. I would also like to thank my external adviser at The Knowledge Media Institute at The Open University doc. Ing. Zdeněk Zdráhal, CSc. for his guidance and support with the research related to the second part of my thesis, especially for all the long lasting discussions in his free time during the weekends.

Moreover, my thanks go to all the other people that supported me on this journey, my parents, all the friends and my colleagues both from Faculty of Information Technology in Brno and from The Knowledge Media Institute in Milton Keynes.

Contents

Chapter 1

Introduction

Most of the computer applications that are used nowadays are data oriented. With the improvement in the information technology and the database systems field, the number of applications using it increased together with the amount of data stored per application. As of January 2017, there are around 46 billion indexed pages on Google[1]; in 2014 Facebook claimed that their data warehouse contained 300PB of data and growing by 600TB every day[2]; in 2015 around 205 billion emails were sent/received per day[3]. There is a famous quote by R.D. Rogers and J. Naisbitt that describes this growth [73]: "We are drowning in information, but starving for knowledge!". This quote perfectly captures the need for finding new previously unknown dependencies/patterns in the data, which would enable us to better understand the application domain captured by the data. Knowledge Discovery in Databases (or Data Mining) is a process that describes obtaining such patterns [39]. It combines methods from several computer science fields including database systems, statistics, machine learning and artificial intelligence.

The first methods developed in data mining were oriented for data typically stored in the relational database in tabular form. With the growth of technology, other sources have become used, i.e. data streams, multimedia data, graph structures etc. These new sources introduce specific problems, and it is very common in the research to propose new methods that are tailored to the particular issues. For instance, lots of new ideas for mining data streams have been presented lots of new ideas for mining the data streams (e.g. classification using Very Fast Decision Trees). These ideas might be applicable across several domains, given that the conditions for the task are appropriate, i.e. the data are arriving rapidly, and it might be impossible to process them more than once.

1.1 Motivation

> *"Necessity is the mother of invention."*
>
> English proverb

Researchers are not only motivated by specific data types, sometimes they are motivated by the domain itself. ID3, one of the fundamental algorithms in data mining, was motivated

[1] http://www.worldwidewebsize.com/

[2] https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/

[3] http://www.radicati.com/wp/wp-content/uploads/2013/04/Email-Statistics-Report-2013-2017-Executive-Summary.pdf

by the induction task challenge in the game of chess[4]. J.R.Quinlan responded to this challenge by ID3 [81, 82], motivating further development (e.g. C4.5 [83]) and used across almost all domains that were analysed using data mining techniques.

Similarly, PageRank by Page et al. [75] was invented purely for the specific task of objectively rating the relevance of the Web pages on the Internet. Ever since, the algorithm has been used not only in Web related research but also in many previously unexpected domains, such as ordering genes in bioinformatics [71], neuroscience [65] and many others [36].

Coming back to the amount of data that is collected nowadays, in data mining we come across problems with cases that occur in data rarely, yet still, they are very important. Frequently, these rare cases are even more important than the rest of the data, but they are difficult for learning algorithms to recognise. This *imbalanced data* [5] situation had been identified as an important problem in machine learning at around the year 2000 [19, 79]. Consequently, the whole myriad of algorithms for learning from imbalanced data has been published. In 2017, the problem is still considered to be important in machine learning, and several open issues and potential future directions have been recently presented in [58].

Imbalanced data sometimes pose new challenges and introduce new previously unseen problems. For example, in a medical environment such as the health care system in the USA, there is interest in identifying when staying in the hospital is not needed for the given patients' condition. This might save health-care system costs and improve patient comfort. Such a system is based on previous cases that are very rare and difficult to identify. Not addressing the imbalanced nature of the data will most likely lead to not understanding under-represented cases[6].

Two application domains motivated the research that I conducted in the field of imbalanced classification - 1) computer security data and 2) Data about students learning and performance in the distance based higher education.

1.1.1 Computer Security Data

In computer security, the number of malicious files present in the data is typically minor compared to the harmless ones. For example, the malicious files in the dataset used by Shabtai et al. in [88] represented 21% of the whole dataset. This is similar to the medical environment where there are a lot of healthy patients (in the computer security domain these would correspond to benign files) and only a few examples of the ill patients (corresponding to malicious files). Moreover, in order to allow the machine learning models to be used in the production environment, there might exist a demand for constraints limiting the maximum error that the classifier is allowed to make (i.e. false positives, false negatives). Such constraints can exist in classification problems in general, and they have been examined, e.g. by Grossi et al. in [37], but I examined the problem in the presence of imbalanced data, which has not been studied so far.

The motivation for this research came from the computer security domain. However, such examples exist in other fields. For example, it might be desirable to deploy the spam filter only when the misclassification of the minority class is lower than a prescribed maximum value. For web spam, the number of legitimate sites is expected to be significantly

[4]King and Rook vs King and Knight – is the game lost for the King and Knight in a fixed number of ply? [81]

[5]The term imbalanced and unbalanced data will be used with the same meaning.

[6]http://www.information-management.com/blogs/data-management/unbalanced-data-finding-rare-needles-in-the-haystack-10028083-1.html

higher than the web spam [30]. The problem of classification with constraints will be broadly explained in Sec. 3.2, leading also to the identified gap of constrained classification in imbalanced data.

1.1.2 Identification of at-risk Students

In *Learning Analytics* (LA) and *Educational Data Mining* (EDM), one of the most important tasks is the early identification of students who are at risk of failing in the courses, so that cost-effective support can be provided.Based on drop-out rates, course details, and other data, there might exist imbalance between the classes that need to be treated. For example, in the case of high schools in Mexico analysed in [70], only 13 percent of students dropped out in their courses, thus a class imbalance ratio 1:7. One of the proxies for identification of at-risk students is predicting the success and submission in the closest possible assessment in the course [105].

The standard way to train the predictive models in such a domain is to leverage the information about the previous runs of the course to make a prediction in the current one. For new courses, this information is not available, and for such a particular case, I proposed a solution that utilises the information from the students that submit the assessments in advance. These data are inherently imbalanced because, in the beginning, no student has submitted and the ratio decreases as the deadline[7] approaches.

In nature and human behaviour, there are situations where it is possible to observe a phenomenon of an emergence of a new concept that is growing in time. For example, a new virus spreads in a human population; a new product is released and then it is followed by purchases with eager customers; new research topics emerge in science, which also need to be identified [86]. The data about student's submission have similar properties as the phenomenon of a new concept emergence. Moreover, the presence of the deadline makes the problem more specific, and it is possible to generalise this as a prediction of satisfying a goal with the specified deadline.

The importance of predictions is naturally higher in the beginning as this gives the possibility for an intervention with the students that might change his/her behaviour to succeed. On the one hand, there is a decreasing imbalance, while on the other hand, there is the decreasing importance of the predictions being made.

1.2 Goals of the Thesis

The primary goal of this PhD thesis is to investigate solutions for the identified specific problems in the application domains of computer security and learning analytics in the classification of unbalanced data, and also how the domain influences the solution to the problem.

Based on the literature survey and the data that I have come across, there exist several specific problems when dealing with the class imbalance that, to the best of my knowledge, have not been addressed before. The first one is a problem of constraints posed on the performance measures in the presence of imbalanced data; the second one is the classification of goal achievement with the deadline and with the imbalance decreasing as the deadline approaches. Motivated by the examples from data mining above, not only might there be

[7] The term deadline, cut-off, cut-off date or due date describes the same concept throughout the whole thesis.

novel solutions to specific problems, but these solutions might also stimulate research in other domains.

This resulted in specifying the following research questions:

1.2.1 Research Questions

How to solve the specific problems of the selected application domains in the presence of unbalanced data?

1. How is it possible to train a classification model for a given data set of imbalanced classes with constraints imposed on classification performance measures?
2. How can a goal be predicted within a specific deadline from a population of objects, taking into account decreasing time of class imbalance?

1.3 Structure of the Thesis

The thesis is organised as follows. Chapter 2 introduces the necessary theoretical background starting with data mining and narrowing to classification, problems of imbalanced data and mining with constraints. Chapter 3 reviews state of the art in learning from imbalanced data, focusing on binary classification, classification with constraints and also the classification of the data with changing imbalance ratio in time with the focus on learning analytics data.

The subsequent chapters describe the research methods and results: chapter 4 explains the description of the problem of binary classification with the performance constraint on the minority class. It also presents two new solutions based on the combination of cost-sensitive logistic regression and stochastic algorithms. Chapter 5 presents the problem of predicting achievement of the goal with the specified deadline when the imbalance ratio is changing in time. The Self-Learning framework is presented as the way how to learn in such a case. The evaluation the Self-Learning approach in case study using data from the educational domain is described in chapter 6. Improvements of the Self-Learning and its impact on the original solutions are discussed in chapter 7.

Chapter 8 summarises the research results and suggests the possible future work.

Chapter 2

Theoretical Background

After the introduction and motivation of the research, this chapter introduces theoretical background of the thesis by first defining Knowledge Discovery in Databases, i.e. Data Mining. Then, the view is narrowed to Classification, performance measures for evaluating classifiers, constraints that might be posed to the data mining process, and to conclude the chapter, introducing problems of learning from imbalanced data.

2.1 Data Mining, Machine Learning

The Knowledge Discovery in Databases, or the popular misnomer Data Mining, can be defined as a process of discovering interesting patterns and knowledge from large amounts of data [39]. This process is typically described by the following iterative steps:

1. *Data pre-processing* that consists of:
   (a) *Data cleaning* – removal of noisy, faulty or inconsistent data,
   (b) *Data integration* – combining various sources into one integrated storage/table,
   (c) *Data selection* – selecting only data that are relevant for the performed analysis,
   (d) *Data transformation* – transforming the data to be easily consumed by the analysis, also some modification might be performed in order to improve the quality of analysis,
2. *Modelling - Data Mining* – analytical part, the main part of the process, application of data mining algorithm in order to obtain new patterns/models,
3. *Pattern evaluation* – quality assessment of the extracted models or measuring *interestingness* of the patterns,
4. *Knowledge presentation* – post-processing in a way that is easily understandable by users, usually using visualisations.

Data mining tasks (step (2) in the process) are divided into two basic categories: (1) *Descriptive*, also known as *Unsupervised learning* and (2) *Predictive* or *Supervised learning*. The goal of Descriptive tasks is to characterise and describe the data usually by some patterns such as association rules or clusters. The goal of Predictive tasks is to find a mapping function (also called a model) from input data to the outputs in order make predictions.

2.2 Classification

Together with *Regression*, *Classification* belongs to supervised learning tasks and it is one of the more utilised and examined tasks in data mining. Inspired by the notation from [72], the problem can be formally described as follows:

Given training data of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathcal{X}, \mathbf{x}_i = (X_i^1, X_i^2, \ldots, X_i^D)$ denote D-dimensional vectors of *attribute values* of input *instances* or *objects* that describe data, and $y_i \in \mathcal{Y}$ denote the output data; the goal of supervised learning is to find a mapping function between input and output $f : \mathcal{X} \to \mathcal{Y}$, i.e. $y = f(x)$. If y_i is categorical (i.e. $\mathcal{Y} = \{1, \ldots, C\}$), the task is called classification and C is the number of classes. When y_i is continuous (i.e. $\mathcal{Y} = \mathbb{R}$) it is referred as *regression* [72].

The data instances $\mathbf{x}_i$ are usually represented as D-dimensional vectors of *attribute values* (also called *features*), but in the real world, it can be any complex and structured object such as a Web document, a biological molecule, a video, voice recording and possibly many others. The transformation to a vector occurs in the preprocessing phase of the Data Mining process, i.e. the step (1).

Narrowing only to classification, when $C = 2$ and usually $y \in \{0, 1\}$, I refer to the problem as *binary classification*, when $C > 2$, it is *multiclass classification*.

The whole classification process usually happens in three subsequent phases:

1. *Training* – Based on a selected dataset (i.e. the *training data*), the mapping function/model is developed (*trained*). The training is usually guided by minimising some error function. This error varies based on the classification approach.

2. *Validation and Testing* – To ensure that the model is able to make predictions on data with unknown labels, its quality is first evaluated on testing data. The testing data must be different from the training data. Moreover, the models often have parameters (called hyper-parameters), which are not part of the training and they should be optimised for the specific problem. Hence, a part of the training data is usually taken aside, and the performance on such data is used to tune the parameters. Various measures are used for estimating the quality of the classifier. These are covered in Section 2.3.

3. *Application, Prediction* – When the analyst is satisfied with the model performance, it is being deployed on data with unknown classes, usually in the production. Later, true class labels might be obtained, which can be used for retraining the classifier.

Two basic approaches to classification exist [37]: (1) *Eager learners* first build a model during the training phase and once this is done, the prediction of new instances happens almost immediately; (2) *Lazy learners* on the other hand do not need to build a model, but all the work happens during the prediction, which might be very costly. An example of the second approach is *nearest-neighbour classifier*, which needs to compare the classified data with all the instances in the training set.

2.3 Performance Measures in Classification

In order to measure the quality of a machine learning model, it is important to select a proper performance measure. Depending on the number of classes, there exist several performance measures for (1) binary classification and (2) multiclass classification.

All classifiers are able to assign the input to one of the output classes. In addition, some classifiers can supply a continuous non-negative value (sometimes called score or confidence), which denotes the "strength" of the instance to the given class. Based on the notation from [29], I refer to the classifiers able to predict only a class as *discrete classifiers*, and those able to provide the confidence value as *scoring classifier*.

Moreover, classification algorithms can be compared not only according to their predictive quality but also by: (1) *speed* of learning the model (some algorithms need only one pass over the data), (2) *robustness* – ability to handle noise and various types of data, (3) *scalability* across more data instances and higher dimensionality, and (4) interpretability of the model (e.g. easily understandable rules from decision tree vs. weights of a complex multi-layer neural network) [39].

When more than two classes are predicted, some of the measures for binary classification cannot be used, but usually, they can be adapted for the multiclass problem using a *one-vs-all approach*, where one of the classes is treated as positive and all others as negative. This computation is performed for all the classes returning the mean of the partial results. The focus of this thesis is on binary classification, therefore multiclass classification will not be described.

2.3.1 Binary Classification

For binary classifiers, during the prediction, each instance is mapped by the model to either a positive or negative class. Most of the measures are based on the 2×2 *confusion matrix*, which makes use of four possible outcomes returned by the classifier. In Table 2.1, the positive instance correctly classified is denoted as *TP*, the positive instance classified as negative is *FN*, the correctly predicted negative instance as *TN*, and the negative instance predicted as positive is *FP*.

Table 2.1: Classification confusion matrix.

	Positive prediction	**Negative prediction**	**Total**
Positive	True Positive (TP)	False Negative (FN)	Total Positive (P)
Negative	False Positive (FP)	True Negative (TN)	Total Negative (N)
Total	Total Predicted P (P')	Total Predicted N (N')	P + N

Many performance measures are derived from the confusion matrix. According to [21, 90], the one that has been frequently used in the practice is *Accuracy* (2.1), measuring the ratio of correctly predicted instances to all the instances; its complementary measure is the *Error rate* (2.2), giving a number of incorrectly classified instances to all instances.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.1}$$

$$Errorrate = \frac{FP + FN}{TP + FN + FP + TN} \tag{2.2}$$

From Information Retrieval area, *Precision* (2.3) and *Recall* (2.4) have been adopted and widely used. They both emphasise the positive class. Recall indicates how many of all the positive instances were discovered, and Precision how many of the identified positive instances are truly positive. The harmonic mean of those measures, *F1-Score* (2.5) is used to provide one number characterising the quality of the predictions.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$TPR = Recall = Sensitivity = \frac{TP}{TP + FN} \tag{2.4}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.5}$$

Recall in the medical domain is denoted as *Sensitivity* and together with *Specificity* (2.6) provide another view on the classification performance. In contrast to the previous, thanks to Specificity it is possible to measure the error that is made on the negative class, for example in the American health care system detecting healthy people who were predicted as sick. Similarly, as F1-Score the *G-Mean* (2.7), being geometric mean of Sensitivity and Specificity is used to summarise the prediction by one number.

$$Specificity = \frac{TN}{FP + TN} \tag{2.6}$$

$$G - Mean = \sqrt{Sensitivity \cdot Specificity} = \sqrt{\frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}} \tag{2.7}$$

Receiver Operation Characteristics (ROC) curve was proposed in [80] to be used as another option to Accuracy. The ROC defines 2-dimensional space by True Positive Rate (TPR) on x-axis (2.4) and False Positive Rate (FPR) (2.8) on y-axis, and the point in the space represents the classifier performance. For scoring classifiers, using a different threshold for assigning the case to the positive class, we can obtain several points in the ROC space that can be visualised. This enables a visual comparison of several classification algorithms. Moreover, it enables us to see how the performance of TPR will change if we impose a constraint on maximal FPR. Very often, *Area Under Curve* of ROC *(ROC AUC* or *AUC)* is used as a single measure for comparing the classifiers. An example of ROC for two different models is depicted in Figure 2.1.

$$FPR = 1 - Specificity = \frac{FP}{FP + TN} \tag{2.8}$$

Apart from ROC curves, there is number of other curves usually working with the possibility of variable classification threshold, e.g. *Precision-Recall curves* [69], *Cost curves* [26].

2.4 Classifiers

This part describes several classification models that have been used in the thesis. They include Logistic Regression, Naive Bayes, Decision Trees and Random Forest. The descriptions of the classifiers are based on [72].

2.4.1 Logistic Regression

Logistic regression (LR) is a linear machine learning model for binary classification. The method is designed to handle numeric variables; categorical variables need to be converted using some encoding. The model is represented by a weight vector, usually of the same

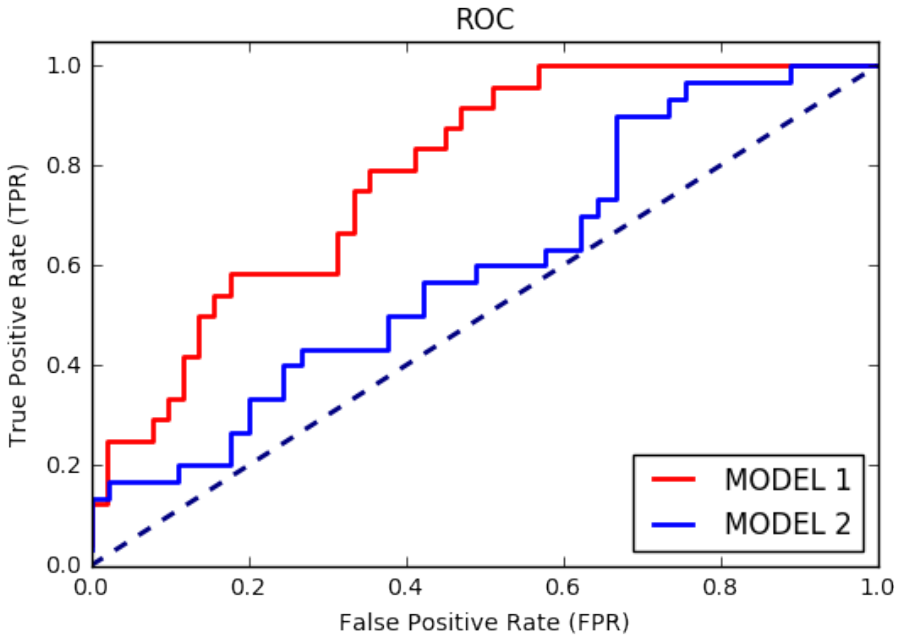

Figure 2.1: Example of ROC curve for two models.

length as the number of features. Then each weight corresponds to the importance of the feature. For a learnt model, the value of the output variable is computed by applying the logistic function to a linear combination of attribute values and the weight vector. The logistic function is defined as follows:

$$P(Y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{w}^T \cdot \boldsymbol{x}_i}} \tag{2.9}$$

, where $\boldsymbol{w}^T \cdot \boldsymbol{x}$ represents the inner or scalar product between the input vector $\boldsymbol{x}$ and the model's weight vector $\boldsymbol{w}$. The function converts the input value to interval $[0; 1]$. The result describes a confidence value for a given case being of class 1. Same as for other scoring classifiers, values higher than a specified threshold (usually $t = 0.5$) assigns the classified example to a positive class, values lower than the threshold to the negative class.

In the training phase, the algorithm tries to solve the unconstrained optimisation problem of the following objective function:

$$\min_{\boldsymbol{x}} C \sum_{i=1}^{n} \log(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}) + \lambda R \tag{2.10}$$

, where $C > 0$ is a penalty or cost parameter and λ stands for a regularisation parameter and R for the regularisation strategy. Setting smaller $C < 1$ limits the cost of misclassification, and it can be used together with regularisation to avoid over fitting of the model [28]. L1 and L2 are two common regularisations used in logistic regression. For L1 the R is

$$R = \|w\|_1 = \sum_{i=1}^{k} |w_i| \tag{2.11}$$

Similarly, for L2, the R is:

$$R = \|w\|_2 = \sum_{i=1}^{k} w_i^2 \tag{2.12}$$

2.4.2 Naive Bayes

Naive Bayes is a probabilistic classifier that is based on the Bayes Rule with the assumption that all the input attributes are independent. The Bayes theorem for two events A and B states:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{2.13}$$

, which means that conditional probability of observing event A given B, $P(A|B)$ can be computed from the conditional probability $P(B|A)$ and the prior probabilities $P(B)$ and $P(A)$. In the classification task, we are interested in computing probability that class y of an instance $\mathbf{x} = \{x^1, x^1, \ldots, x^D\}$ is c, leading to an equation:

$$P(y = c \mid \mathbf{x}) = \frac{P(y = c)p(\mathbf{x}|y = c)}{P(\mathbf{x})} = \frac{P(y = c)p(x^1, x^2, \ldots, x^D|y = c)}{P(x^1, x^2, \ldots, x^D)} \tag{2.14}$$

The most important part is the numerator of the fraction $p(\mathbf{x}|y = c)$, as it is difficult to compute the conditional joint probability of the features. That is, why in Naive Bayes there exists an assumption of the independence, which allows rewriting the equation as:

$$P(y = c \mid \mathbf{x}) = \frac{P(y = c) \prod_{i=1}^{D} p(x^i \mid y = c)}{P(x^1, x^2, \ldots, x^D)} \tag{2.15}$$

Then, the training of the models consists of computing the probabilities of each value of all the attributes x^i being of the given classes $c \in C$. During the prediction, the class with the highest probability is returned as the predicted class.

2.4.3 Decision Trees

Decision trees are models that can be used both for classification and regression. They are built by recursively partitioning of the input space to distinct non-overlapping regions. The whole model is a tree, and each region is represented by a node in the decision tree. The node can be described by a local model, i.e. the route from the root to the node. The leaves of the tree contain the final decision about the predicted class (in case of classification), as they represent the region in the state space with the same class.

Figure 2.2 shows an example of the classification tree for deciding whether playing outside given three features is suitable. Starting from the root node, the values of the attribute are evaluated and navigated towards the leaves to provide the predicted class. For example, in the Figure 2.2 for example with High humidity, Sunny outlook and Windy condition, first the high value of humidity would result in a comparison of the Outlook attribute and result in the prediction of playing being suitable, omitting evaluation of the Windy attribute entirely.

The principle of the training algorithm is to find the tree that best separates data into the given classes. Finding an optimal partition is an NP-complete problem, so greedy top-down algorithms based on a divide-and-conquer strategy are usually applied. They start from the root try to find locally the best split in the state space.

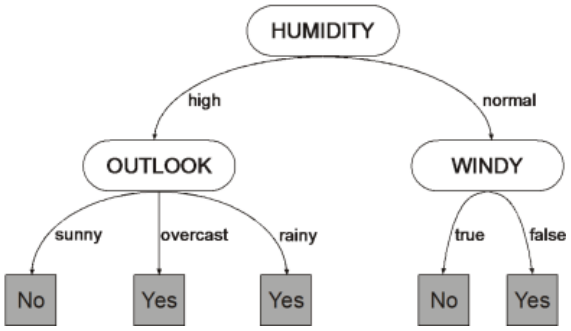

Figure 2.2: Example of a decision tree predicting if playing outside is suitable. Taken from [103]

The important parts are the definition of the splitting function and the definition of the stopping criteria. The heuristics for stopping the growth can include: all instances in the node are of the same class, the depth of the tree reached maximum allowed value, the gain of the splitting reduction is lower than some minimum threshold or the number of instances in the node is too small.

The most common algorithms are ID3 [82], C4.5 [83] and Classification and Regression Trees (CART) [14] and they differ by the way how they perform the splitting. ID3 is selecting the attribute with the highest *information gain*, which reduces the disorderliness of the dataset D with the classes $c \in \{1, 2, \ldots, C\}$ defined by the entropy as:

$$Entropy(D) = -\sum_{c=1}^{C} p(c|D) \log p(c|D) \tag{2.16}$$

The problem of information gain is that it favours attributes with many possible values. C4.5 solves this by introducing *gain ratio*, which normalises the information gain by the number of features values. Instead of entropy, CART algorithm proposed *Gini index*:

$$Gini(D) = 1 - \sum_{c=1}^{C} p(c|D)^2 \tag{2.17}$$

To prevent over-fitting of the decision trees, several strategies are implemented. Pre-pruning and post-pruning of the branches are performed using the validation set, maximum depth of the tree or a minimum number of instances in the leaves can be specified. Cross-validation is recommended to be used to obtain the best parameters for a particular dataset.

2.4.4 Random Forest

Random Forest [13] is a representative of ensemble classifiers based on bagging. The ensemble classifiers instead of relying only on learner construct the classifier by a set of *base learners* and combine their results. Each of the base learners focuses on modelling a part of the input space, and using their combination allows covering the whole state space. Boosting and Bagging are two basic ensemble strategies.

Bagging stands for *bootstrap aggregating*, i.e. combining bootstrap sampling and aggregating. Bootstrap sampling performs for each base learner sampling of data with replacement, resulting in the same number of instances as in the original training data. The base learners are trained on the sampled data, and the final class prediction is achieved by the

majority voting of the base learners. Moreover, the growing of the decision tree incorporates feature selection in each of the nodes. A predefined number of m features are selected by random (hence Random Forest). The splitting function of the trees remains deterministic. This strategy makes the model computationally more efficient than the standard bagging, which needs to process all the input features. Moreover, it also eliminates the problem of bagging connected to construct models that are correlated with the growing number of base learners.

Random Forests are typically trained on a high number of trees, usually hundreds, which allows even the local features to be selected in one of the trees as the main attribute. The algorithm showed also very good properties when dealing with high dimensionality of the features.

2.4.5 Other Classifiers

The number of existing classification models is large, each one usually suitable for a different type of task or used dataset.

Classification based on **k-Nearest Neighbours (k-NN)** is a lazy learning model, which predicts the class of the instances based on the class membership of the majority of its k nearest neighbours.

Neural Networks (NN) are an extension of linear classifiers, consisted of linear units (neurons) connected in a various way in the network. Nowadays, after the progress in computational power, different models of Deep Neural Networks are used in computer vision, image processing or speech recognition.

Support Vectors Machines (SVM) is very similar to the logistic regression, and in its base version, it only differs in the optimised function, which is based on maximising the margin separating the classes. The kernel trick, introduced later, allows the classification even in the spaces that are not linearly separable. This is performed by replacing the computation of the dot product by various kernel functions, with radial basis kernel being the most common one.

Boosting is along with bagging another type of ensemble classifier. In contrast to bagging, the classifiers are trained sequentially, with the models trained later to focus on the data that caused the most errors in the previous models. Similar to bagging, trees are popular base learning algorithms for boosting. Gradient Boosting Machines (GBM) [32] or recently introduced xgBoost (XGB) [20] are popular representatives.

2.5 Mining with Constraints

As mentioned in the Introduction, sometimes the application domain might provide additional background knowledge. It can be injected into the analysis to improve the relevance of the results, for example by introducing additional constraints. For example, specifying pairwise constraints for instances in classification, indicating that the instances have the same class, can improve classification on unlabeled data. The constraint may have a form: *I am interested only in transactions whose total value was at least $50.* This might result in: (1) selecting only the patterns that are interesting and (2) possible speed up of the extraction process by limiting the state space to be searched.

Recent surveys [37] have provided a systematic overview of the existing state-of-the-art about constraints in data mining. They classify the approaches in four dimensions:

1. **Mining task** – the main distinction is in the type of the task: (a) *classification*, (b) *cluster analysis*, and (c) *pattern mining*.

2. **Objects** in the mining process, to which the constraints apply, (a) *data* – constraints define which kind of data/items are allowed to enter modelling step, (b) *models* specify how the model should be constrained (e.g. maximum complexity), and (c) *measures* usually posit requirements on the quality of the models (e.g. quality of the clusters, or performance measures in classification).

3. **Type** of the constraint that can be either (a) *hard* or (b) *soft*. Soft constraints allow some solutions to be acceptable even if they do not satisfy all the constraints, hard constraints might cause the algorithm to return an empty result.

4. **Phase** of the process where the constraint is applied: (a) *pre-mining*, (b) *mining*, and (c) *post-mining*.

According to this taxonomy, Table 2.2 presents the dimensions classes for classification being selected as a mining task.

Table 2.2: Dimensions of constraints classes that can be imposed for classification

	Data	**Models**	**Measures**
Phase	pre-mining, mining	mining, post-mining	mining, post-mining
Type	hard	soft	hard, soft

2.6 The Problem of Imbalanced Data

In many real world supervised learning scenarios, a class exists that has significantly lower number of instances in the data than the other class. I refer to the first one as the *minority class* and the latter as the *majority class*. This has been identified as a research problem in machine learning. One of the first references dates back to 1993 and relates to properties of neural networks on imbalanced training data [3], and another to 1997 for the detection of oil spills using Neural Network and C4.5 classifiers [59]. When using a standard classification algorithm, the minority class is treated as being of the same importance as the majority one. For example, when using the Error rate as the driving factor for a classifier, the result will always return the majority class, which minimises the error. The problem is that the minority class has usually higher importance and such classifier, though formally optimal, is unacceptable [12]. The data can be imbalanced because of either low natural occurrence of the minority class in the data (e.g. ill patients in the medical diagnosis) or as a result of not collecting enough data (e.g. minority examples might be more expensive to obtain) [102].

It is not only the high ratio between the classes that causes the problem. Intuitively, if the concept that separates the data is not complex and there exists, for example, one attribute that discriminates between the two classes perfectly, the classifier would still be able to provide predictions with $Accuracy = 1$. However, as the complexity grows the higher imbalance ratio causes greater errors [49]. Naturally, the following question arises: *What is a significantly lower number of instances so that we can say that we face the imbalance problem?*. To be precise, imbalanced data are every data that do not have the same number of instances of the classes. According to the experiments done in the recent study [78], from

the practical view the losses start to be significant when the minority class represents 10% of the data or less.

[66] described 6 problems that occur together with the presence of imbalanced data, which are a decomposition of the complexity in the data together with several other problems:

1. Problem of overlapping between the classes – if the classes are perfectly separated, the problem of imbalanced data does not hinder the classification that much,
2. Identification of areas with small disjuncts – one compact concept (despite being small) is easier to be covered by a rule than many small disjuncts or clusters spread around the dataset,
3. Impact of noisy data – related to the previous problem, some small disjuncts can be treated as noise,
4. Lack of density and information in the training data,
5. Significance of the borderline instances,
6. Possible differences in the distribution of the training and testing data (dataset shift problem).

Similarly, as for classification methods, the domain might impose some constraints of the performance measures on some classes, which will become more difficult if it is the minority class. For example, it might be reasonable to ensure having a high discovery rate for ill patients, even at the cost of treating some of the patients with unnecessary treatment.

Chapter 3

State of the Art

This chapter reviews the current state of the art of Learning from Imbalanced Data, primarily focused on classification. With relation to the motivational examples, existing methods from classification with constraints are summarised.

3.1 Learning from Imbalanced Data

Now that the problem of imbalanced data has been explained in Section 2.6, it is possible to introduce the existing state of the research. Hundreds of papers addressing the problem have been published since its first occurrence. Therefore, it is worth systematically dividing the current work as follows: (1) investigating evaluation metrics, (2) finding ways to deal with the imbalance in order to improve prediction quality, and (3) investigating the source of the imbalance problems in depth, i.e. what are the common problems of imbalanced data or what kind of problems typically accompany them.

3.1.1 Evaluation Measures

Accuracy and error rate are considered as typical evaluation metrics for classifiers. But the usage of these metrics is not suitable when the data are not balanced, and also when false positives and false negatives have different importance. The problem of accuracy is that it takes positive and negative classes with the same importance. If it is used as a selection measure, with imbalanced data, it will prefer trivial models that return majority class no matter what the data are. For example, if the class ratio is 1:99, then selecting the majority class will give Accuracy 99%, regardless of the attribute vector.

For this reason, measures that take the distribution of the classes are favoured. Most of the papers use ROC AUC for ROC if the focus is to provide the view for various values of the prediction threshold. In cases of focusing on the fixed version of the prediction threshold, G-Mean or F1-Score is used, depending on the specific domain, either preferring Precision (in F1-Score) or Specificity (in G-Mean).

Several other measures designed specifically for imbalanced data have been proposed. When the skew in the data is high, Precision-Recall curves *(PR-curves)* are the preferred variation to ROC curves, as the ROC tends to give a too optimistic view of the performance [22, 51]. Similarly to the area under the ROC, the performance of the PR-curves can be measured using one value as an area under the PR-curve *(PR AUC)*. The difference between these two is that ROC eliminates the influence of the class imbalance in the metric, working only with the ratio with respect to the number of instances per each class. On the other

hand, PR-curves replace Specificity (or more precisely FPR) with Precision. This removes correctly identified instances of the negative class and it makes the metric more sensitive towards the positive class.

In [35], *Dominance* (3.1) was proposed to show which class accuracy prevails in the data; *Index of balanced accuracy* (3.2) presented in [34] weighs any performance measure M by the dominance, which highlights accuracy on the minority class, i.e. Sensitivity. This is passed into the equation by the aforementioned Dominance.

The purpose of this review is not to provide an exhaustive list of all the existing measures, but to show that there are many of them, which might be driven by the special purposes in the data. Other measures can be found in [42, 12, 66].

$$Dominance = Sensitivity - Specificity \tag{3.1}$$

$$IBA_{\alpha}(M) = (1 + \alpha \cdot Dominance) \cdot M \tag{3.2}$$

3.1.2 Approaches to Tackle the Imbalanced Data

In recent years, there have been several review studies to address the imbalance problem. All of them classify the approaches into data preparation and algorithmic phases, some of them in a more general way [58, 66, 33], others directly subdividing the algorithmic ways mentioning the Ensemble approach [19], active and kernel learning [42]. One of the most recent and most systematic approaches was proposed in [12], classifying the methods into 4 categories: (1) Data Pre-Processing, (2) Special-purpose Learning methods, (3) Prediction Post-Processing, and (4) Hybrid methods, combining the previous approaches. (3) and (4) can be viewed as two categories of algorithmic approach, but some of the approaches that would be classified in other reviews as algorithmic, are treated as data pre-processing, such as Active Learning and Weighting the Data Space. The classification is also depicted in Figure 3.1.

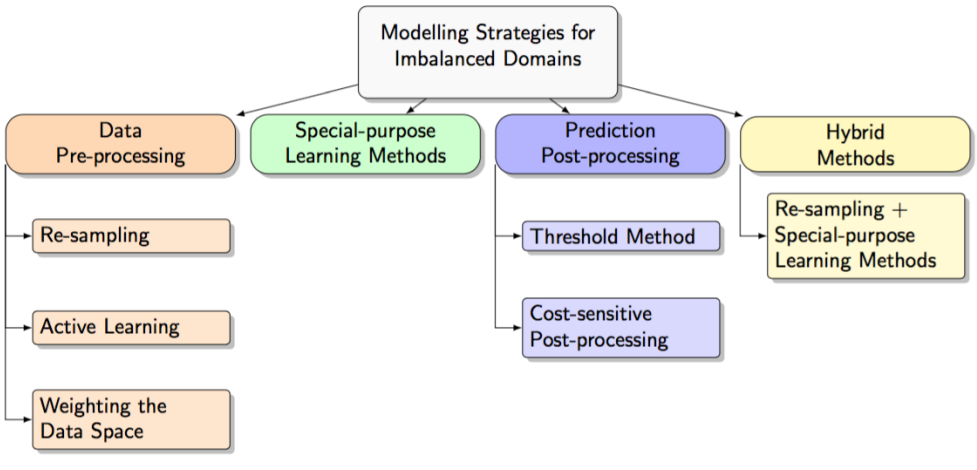

Figure 3.1: Taxonomy of ways how to tackle the imbalance problem [12].

3.1.3 Data Pre-Processing – Sampling

The idea behind sampling methods is to modify the class distribution so that the proportion of instances of each class is balanced. It is possible to distinguish between random and informed methods and between over-sampling and under-sampling. Random oversampling randomly selects a group of minority class instances and duplicates them. In contrast, random under-sampling randomly selects a group of majority class instances and removes them from the training set. The over-sampling approach has an advantage in that no information is lost, on the other hand, it might lead to overfitting; under-sampling leads to smaller datasets being generated, thus the learning algorithms are more time and memory efficient, at the cost of losing information, due to the removal happens by random.

In order to improve the quality of sampling, more complex informed sampling methods were presented. The best known is the **Synthetic Minority Oversampling Technique (SMOTE)** [18]. Instead of duplicating the points of the minority class, new points are synthetically generated along the lines that connect the nearest neighbours of minority class points. The problem of SMOTE is not considering, whether the points are generated for borderline examples between the two classes and thus creating overlapping between them. This led to a development of several other methods, such as **Borderline-SMOTE** [38]. In contrast to SMOTE, Borderline-SMOTE selects only minority class points surrounded mostly by the majority points as candidates for generating synthetic data.

To eliminate the issues of random under-sampling, informed techniques include some heuristic that should avoid removing the important information from the majority class. The methods often work with distance and the neighbourhood of the data.

A group of three **NearMiss** methods [68] selects to retain only the majority points with (1) the smallest distance to three closest points; or (2) farthest minority points; or (3) to a given number of closest majority points to minority points. However, only the second method was able to perform in their study better than the random under-sampling.

Edited Nearest Neighbourhood (ENN) again uses three nearest neighbours of the data by applying ENN rule. For a majority class example x, if the classification of at least two of its three nearest neighbours is a minority class, then the example x is removed from the training dataset.

Neighbourhood Cleaning Rule (NCR) [63] is an update of ENN [104]. In addition to applying the ENN rule for majority class examples, for each minority class applies the following: If its three neighbours misclassify the example as a majority example, these neighbours that belong to the majority class are removed. This means that more examples should be eliminated.

Condensed Nearest Neighbours (CNN) method focuses on removal of majority class examples distant from the decision border of the classes. The CNN rule is applied [41] to find the subset consistent with the training data, i.e. a subset that can classify the training data correctly using the 1-NN classifier, i.e. a k-NN classifier with $k = 1$. An algorithm proposed in [59], after a random pick of one majority class example and all the minority classes examples, it adds the misclassified examples recursively to the consistent subset.

One-sided selection (OSS) [59] applies the CNN rule after Tomek-Links, i.e. it is a cascade of two under-sampling algorithms. One is focusing on removal of the borderline examples and the other one on the examples far from the borderline.

Another approach is an elimination of majority class examples by identification of *Tomek Links*. Two data points E_i and E_j with a distance $d(E_i, E_j)$ are considered as Tomek Links

if they are from the opposite class and there is no point E_l such that $d(E_i, E_l) < d(E_i, E_j)$ or $d(E_j, E_l) < d(E_i, E_j)$. In other words, they are the links between the points from opposite classes with no other point in between. Such cases are either borderline or one of them is a noise.

SMOTE-Tomek is a method that applies Tomek-Links removal after over-sampling by SMOTE to eliminate generated borderline points [8]. In comparison with Tomek-Links, both majority and minority examples can be removed from the data. This is because the newly generated clusters of examples from a minority class can reach it too deep in the majority class space.

Similarly to SMOTE-Tomek, **SMOTE-ENN** [9] applies a different cleaning method after SMOTE and likewise might remove both majority and minority class data. On the other hand, this method usually selects more data for removal.

There are many other algorithms focused on both under-sampling, over-sampling, combining the two together, or using an ensemble of classifiers on the sampled data such as [64]. Also, several clustering based approaches have been proposed [42, 12, 66]. To conclude, the research field in re-sampling is well examined, now usually consisting of tweaking up the existing approaches by combining with others, such as the recent combination of SMOTE and DBSCAN clustering algorithm *DBSM* [87].

3.1.4 Other Data Pre-Processing techniques

The Active Learning

Traditionally, the active learning approach is used to select the unlabelled instances for an expert to annotate it. In imbalanced learning, this means selecting instances that are difficult to classify, those that are close to the boundaries to construct a classifier there. Afterwards, it is usually combined with another classification method, such as SVM [27]. The problem with these methods is in most cases a computational cost related to identifying the most informative instances [27] and these methods are usually not included in the existing algorithm comparison and reviews.

Weighing the Data Space

This approach is related to cost-sensitive learning, described more in the Section 3.1.5. In [12], they argue that it is one of the ways how to achieve it. The study mentions [111], where the weights are assigned to data to reflect their importance by either providing it to the algorithm or by subsampling. In fact, the first approach can be considered as a specific algorithmic approach and the second one can also be categorised as a sampling method.

3.1.5 Special-Purpose Learning Methods

Instead of modifying the underlying data, many research efforts have focused on algorithm modifications. The goal is to put the bias towards the minority class into the learning process so that it penalises the errors on the minority class more. Many of the existing work is based on the foundation of cost-sensitive learning, building on the concept of cost matrix, sometimes utilising ensemble techniques; some other techniques are based on editing kernel functions.

Cost-Sensitive Learning

Cost-sensitive learning methods penalise the misclassification of minority class instances more than the instances from the majority class. To achieve this, they utilise the cost matrix, which contains penalties/costs for all correct and wrong classifications. The concept is applicable not only for imbalanced data but in any case where one wants to express that one class is more important than another. Also, it is not only limited to binary classification. Let's denote a cost of misclassifying the class c_j as c_i as follows:

$$cost(c_i, c_j) = N \tag{3.3}$$

Then let's denote Min as the minority class and Maj as the majority class. For imbalanced data, the typical case is that correct classifications are not penalised, and $cost(Maj, Min) > cost(Min, Maj)$, i.e. there is a higher cost of classifying minority example as the majority. The example of a cost matrix for binary imbalanced classification is shown in Table 3.1. Notice that in the case of binary classification, it is worth setting only one cost. Cost-sensitive learning methods suppose that these costs are provided by a user in advance. But very often, the costs are not known and it is up to the algorithm to infer them. Opposed to *cost-insensitive* algorithms (i.e. usual ones), the cost needs to be considered in the optimised criterion during training.

Table 3.1: Cost matrix for majority and minority class.

	Predicted minor	Predicted major
Actual minor	0	> 1
Actual major	1	0

Neural Networks have been investigated in [60] using three methods, how cost-sensitive learning can be incorporated in the training. The first method changes the output of the output layer in the neurons, giving classes with a higher cost a higher impact in the error backpropagation. The second method applies different a *learning rate* in the update of the weights of the neurons as follows:

$$\eta(x) = \frac{\eta \cdot CostVector(y(x))}{max(CostVector(i)_i^C} \tag{3.4}$$

, where η is the learning rate of the data example x where its class is $y(x)$, and it is normalised by the maximum available misclassification cost. $CostVector(i)$ represents the cost that can be made on the class i. The last option is changing the objective function that is being minimised from square error to misclassification cost as follows:

$$E = \frac{1}{2n} \sum_x \|(y(x) - y'(x)) \cdot K[y(x), y'(x)]\|^2 \tag{3.5}$$

$$K[i, j] = \begin{cases} CostVector(i), & \text{if } i = j \\ cost(i, j), & \text{if } i \neq j \end{cases} \tag{3.6}$$

,where $y(x)$ is the real class of x, $y'(x)$ the predicted class, and K represents the cost factor that is computed based on the correctness of the predicted class. Based on their experiments, the updated error function provided the best results in terms of the minimum cost on the testing data.

Some studies such as [12, 42] claim that Decision Trees have been investigated for cost sensitive learning by a special purpose algorithm update. However, the only mention relates to using a proper splitting criterion that is class distribution independent, such as Entropy or Gini index. But these measures have been designed for the decision trees from the very first beginning [82, 83].

Ensembles Methods

Many ensemble methods have been modified for the cost-sensitive scenario, most of them boosting techniques. AdaC1, AdaC2 and AdaC3 in [92] are changed versions of the AdaBoost algorithm [31] in how it updates the weight in every step, again giving higher importance to the examples with higher misclassification cost.

Other Techniques

Moreover, among other paradigms that have been explored is the modification of $k - NN$, where the distance for the majority class examples is extended, causing the bias towards the minority class. Several kernel techniques (e.g. SVM), with the goal of moving the maximising hyperplane towards the minority class [12, 42].

3.1.6 Prediction Post-Processing

Two approaches can be applied in the post-processing phase:

Thresholding

Thresholding is an easy method to deal with the imbalance after the training of the classifier. Having a scoring classifier, it is possible to apply different threshold values and thus move on the ROC curve. Then, it is important to select the threshold where the error (e.g. cost error defined by the cost matrix) is minimal. This has been mentioned in [67, 101].

Cost-Sensitive Post Processing

To perform cost-sensitive learning using existing algorithms, several meta-learning methods have been proposed, such as *MetaCost* in [24]. One base learner is used to train the classifier. MetaCost only operates with the predicted probabilities of the base learner, i.e. it post-processes the probabilities. It then chooses the class with the minimum expected cost of the wrong prediction. Partially, this method can be considered as an ensemble method as it internally uses the aggregate of predictions across more samples of the data. Despite their easiness of use for the imbalanced data, according to the study in [12], the cost-sensitive methods have never been applied for it.

3.1.7 Hybrid Methods

Hybrid methods combine the aforementioned approaches, usually using ensemble techniques.

3.2 Classification With Constraints

Referring back to the taxonomy of constraints in data mining in 2.5, constraints can be defined on data, models and measures [37].

3.2.1 Data Constraints

Among the data oriented method, the simplest approach is value restriction of some attributes, for example in order to keep some information secret/protected in the model. Motivated by a possibility of some organisation to pass some data to other organisations, this was discussed in [23]. Ever since this was used in privacy-related research. In contrast, to improve the classification for unlabelled data, *pairwise constraints* between two data instances, which indicates that they belong to the same class, can be used for several algorithms such as $k - NN$, SVM or *Logistic Regression* [62]. Instead of using the constraints on the instances, a background knowledge about features can be utilised. Consider an example with unavailable classes for news from the Internet for classifying whether articles come from either a hockey or baseball area. Even with no data, we can expect that the word *puck* will be more probable to appear in hockey articles. A generalised expectation probabilistic model was proposed in [25], which is shown to be more efficient than using labelled instances and suitable to be used with existing ontology or task-specific constraints. Further, many examples come from multilabel classification when there might exist information about ordering to which the class belong, or there might be some hierarchy defined on the classes. The data constraints are usually hard, thus not allowed to be relaxed.

3.2.2 Performance Measure Constraints

Imposing a constraint on the classification performance measure can be a typical requirement of a user/customer of the models. For example, let's consider the classification of programs, whether they are Benign or Malicious and a fictional company that will release a software product to do this. They might decide that they cannot afford to release a software product that has Accuracy below some value, let's say 80%, and thus if the value is not high enough, they will just throw the model away. Going back to the classification problem and confusion matrix related measures for binary classification 2.3.1, the scoring models can be adjusted by changing the threshold when classifying the predicted example. Increasing this measure can increase the performance of one measure but decrease the value of another. For example, we can increase Recall up to 100%, leaving usually $Precision = 0$. Similarly, Sensitivity and Specificity can be balanced. Going back to the file classification problem, instead of putting the constraint on Accuracy, it might be reasonable to require that the Sensitivity (i.e. accuracy on the malicious files) cannot drop below 95%. ROC graph is an easy way to visualise the trade-off between those measures and applying different threshold is a simple method how to include the performance measure constraint in the post-mining phase.

ROC Isometrics

Vanderlooy et al. in [96] posed a question, whether it is possible to construct a *reliable classifier*, which is the one guaranteeing classification performance on each class. Such a classifier can be achieved by classifying only instances where it is confident and leaving the other ones unclassified. This "I do not know" decision might be then passed to another

classifier, human expert or completely thrown away. They call this as a *abstaining classifier*, which classifies the result using two specified thresholds $a, b; a > b$ according to the Algorithm 1.

Algorithm 1: Classification process using an abstaining classifier [96].

Input : Instance x to be classified, thresholds a, b, $a > b$
Output: Classification for x, positive, negative or unknown

$l(x) = score_{positive}(x)$
if $l(x) \geq a$ **then**
  return positive;
else if $l(x) \leq b$ **then**
  return negative;
else
  return unknown;

In the algorithm, $l(x)$ denotes the confidence of the scoring classifier that the instance is of a positive class, if $a = b$, then the behaviour is the same as for a usual discrete classifier, i.e. without the thresholds. Their goal is then to train a classifier that can satisfy the performance constraints having the number of unclassified instances as low as possible. The performance measures taken into account are: (a) Precision, (b) F-Measure, and (c) $m - estimate$, which is edited Precision counting with some instances being classified *a priori*. The final constraint consists of the constraints on both positive and negative classes. These constraints can be visually described using the ROC isometrics, i.e. lines in the ROC space constructed of points with the same value of the performance measure. An example of such a measure is depicted in Figure 3.2.

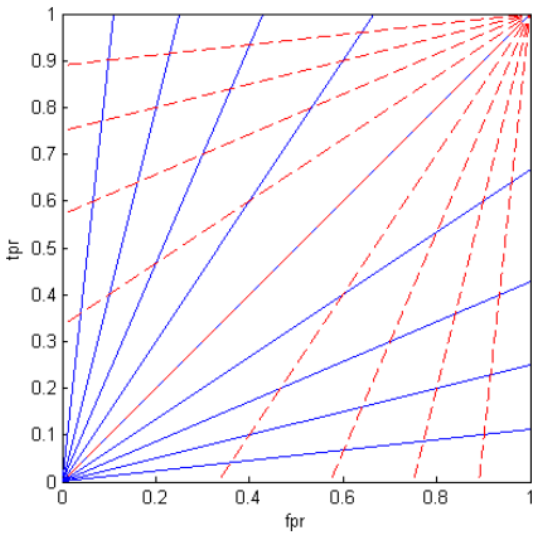

Figure 3.2: ROC Isometric lines for the Precision of the positive (blue) and negative (red) class [96].

The optimisation then consists of finding an intersection of the isometric lines for positive and negative class and examining the position of the intersection with the ROC curve of the classifier. Three cases can happen: (1) If the intersection of the isometric lines lies on the

ROC curve or (2) their intersection lies below the ROC, then the classifier already satisfies the constraints. In the second case, it also performs better than desired. (3) Otherwise, an abstaining version of the classifier is constructed by omitting some cases for classification. These cases are depicted in the Figure 3.3.

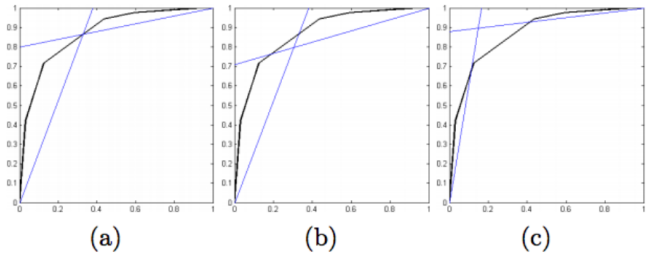

Figure 3.3: Three options of position of the isometric lines intersection and the ROC curve (a) Case 1, (b) Case 2, (c) Case 3

The method was evaluated using two classification models(k-NN and Naive Bayes, called ROC-kNN and ROC-NB respectively) on ten benchmark datasets, both for positive and negative class. The Precision was selected as a performance constraint. Both approaches gave a similar average percentage of correct classifications; ROC-kNN resulted in higher *effectiveness*, which means that fewer data instances were needed to be removed from ROC-kNN than from ROC-NB.

Ensembles for Constraint Performance Requirement

Another problem was proposed by Yan and Goebel in [106] by the possibility to specify maximum FPR or FNR that a classifier is allowed to make, i.e. maximum allowed error on positive or negative class, and minimising the other. They refer to this as *constraint performance requirement*. For some given maximum allowed FNR, denoted as a, it can be written as follows:

$$\begin{aligned} \min \quad & FPR \\ \text{subject to} \quad & FNR \leq a \end{aligned} \tag{3.7}$$

The same can be expressed as the minimisation problem in a dual form as:

$$\xi = FPR + \lambda \cdot (1 - FNR - a) \tag{3.8}$$

, where λ is a Lagrange multiplier.

The proposed solution consists of building an ensemble of single classifiers in the training phase and then averaging their result. Here Neural Networks were proposed to be the inner classifier used for the ensemble. Each of the single Neural Network is usually constructed on the different subset of the data and build with a different number of hidden neurons, ensuring that the resulting classifiers are as diverse as possible. Since there is a performance constraint put on the result, this might not be enough; so the authors proposed to edit the feature selection for each of the Neural Networks by one run of a Genetic Algorithm, with the constraint encoded as a fitness function. The exact formula of the function is, however not presented in the paper. The resulting ensemble is then constructed of 10 Neural Networks as base classifiers.

Their experiments were conducted on their real-world dataset for non-destructive inspection. The dataset contains 5600 examples described by 370 features, with 2600 positive and 3000 negative cases, i.e. relatively balanced classes. The goal was to achieve $FPR \leq 50\%$ and $TPR \geq 98\%$, with the problem specified as reaching minimal FPR given that TNR has to be at least 98%. The comparison was made against the same solution, but using Accuracy as the fitness measure for the genetic algorithm. The method was able to achieve $FPR = 44.2\%$ compared to $FPR = 49.7\%$, when Accuracy was used in the fitness function.

Other Approaches

In addition to the two approaches, several specific problems related to performance measures were also proposed. For online learning [10] presented maximisation of average TPR under average FPR constraints. Given this scenario, they claimed that this goal is not attainable and proposed an online learning meta-algorithm with a relaxed goal. Compared to the previous solutions, which were working with hard constraints, soft constraint was specified.

3.2.3 Classification With Constraints On Unbalanced Data

As with constraints for the general classification, there may be constraints set for learning in imbalanced data, especially for the minority class which usually has a higher importance. Given the performance constraint on the minority class, the question remains of what performance can be achieved for the minority class. To the best of my knowledge, there has not been any publication presenting a solution to this problem.

3.3 Classification in Temporally Decreasing Imbalance Ratio

If a new concept appears in nature or in the society, it takes some time until it is spread out. In the beginning, there are usually only a few instances of this new concept. Let's mention some examples:

- **Example 1:** a new virus affects only a few people/patients in the beginning and then it spreads across the population.
- **Example 2:** When a new product is released into the market, possibly accompanied by a marketing campaign, it is followed by purchases by eager customers.
- **Example 3:** In the research environment, new research topics and communities emerge, mostly as divisions from existing ones. Again, in the beginning, only a few published papers exist before the community grows.

For all the examples, there is also research focused on analysing this phenomenon. In example 1, there are mathematical models trying to explain the spread of viruses [5]; similarly in Example 2 there are many studies focusing on predicting the behaviour of new products, units sold, and estimating revenues [40]. Likewise in the research field, Example 3, there has been published work focusing on detecting new, embryonic, research topics [86].

Temporal changes of the class imbalanced have generated considerable research interest. The survey, released in 2016, which proposes open challenges in learning from imbalanced

data, [58] mentions it among problems related to data streams[1]. This is not the case with the problem of the current data. However, several other issues are often related to data streams. One of them is *new class emergence*, which starts with a highly under-sampled number of instances in the beginning and then grows over time.

3.3.1 Classification with the Event Deadline

This phenomenon can also be specialised to cases, where there exists some final date or deadline when the event should end. For example, people are expected to fill in their tax report and make a claim; similarly, customers are expected to make a purchase before the expiration date of the voucher or before the Christmas day etc.

The specific phenomenon with the fixed deadline also exists in educational data, when students are supposed to submit their assessments before a specified deadline. Here, the courses that are new and do not have any history are considered for analysis. The first assessment is always an important milestone for identifying students at risk of failing the course; there are students that submit this assessment in advance, and these submissions are used for training the model whether students will submit within the deadline. The importance is in identifying at-risk students as early as possible. On the other hand, in the beginning, only a few students submit the assessment, so the imbalance ratio is higher than closer to the deadline.

The specificity of the problem with students lies in the specific deadline, where the students are requested to submit the assessment. For example, [99] deals with the changing imbalance ratio as well, having both abrupt and gradual change, however, there is no assurance whether the class ratio will be increasing or decreasing; moreover, new data arrives instead of new information about the current objects.

A similar task has been studied in [93], where they focused on predicting defects in the source code from the versioning system of the open source projects. The goal was to detect changing the line of the source code by another that fixed the bug introduced by the previous code. Although this data generates imbalance by its nature as well, the absence of the deadline differentiates the problem. Also, in this particular domain, one of ways imbalanced data was handled was by extending the gap that defining the line has indeed changed in the source code.

To the best of my knowledge, a solution that specifically addresses the problem taking into account a deadline has not been published. One of the reasons for this might be not having data available data for such an analysis.

[1] Data Streams – Research in this area focuses on the methods, where the data arrive into the system fast and in such amount that is almost impossible to perform iterative approach and they need to be processed only once.

Chapter 4

Constrained classification

The first identified gap includes handling binary classification on imbalanced data with performance constraints. Motivated by the appearance of the problem in the computer security domain, no existing method has been found in the current state-of-the-art. First of all, the problem has to be described and defined, together with the selection of proper evaluation measure and evaluation strategy. The current approaches to tackling imbalanced data need to be considered as the starting point together with the existing solutions for classification using constraints.

As the previous research suggested, there is no easy way how to train the classifiers given the constraints. Moreover, the bias that puts minority class into more importance has to be taken into account because of the class imbalance. For this reason, cost-sensitive learning techniques were considered as a starting point. They allow putting different costs in the training process, which can be utilised both for tackling the imbalance ratio and for incorporating the constraint function. Moreover, as the constrained solution is difficult to achieve by standard optimisation methods, stochastic methods such as Genetic Algorithm were used to improve the performance of the final model.

Two papers, which describe the original solution of the problem of classification under performance measures, have been published. The first one [44] describes the problem and propose a method based on cost-sensitive logistic regression and genetic algorithm; the second one [45] presents novel constrained strategies using Particle Swarm Optimisation and compares the usage both with and without logistic regression as a pre-processing step.

4.1 Problem Description

For the binary classifcation in imbalanced data, the class of interest (or the positive class, class 1) is the minority class; and the negative class is the majority class. I will refer to the Sensitivity also as accuracy on the minority class and to the Specificity as the accuracy on the majority class. The primary goal is to maximise the Specificity given the minimum Sensitivity constraint. This can be rewritten as the constrained optimisation problem:

$$\begin{aligned} \max \quad & Specificity \\ \text{subject to} \quad & Sensitivity \geq minSensitivity \end{aligned} \tag{4.1}$$

where $Specificity(M, D)$ refers to the specificity for the machine learning model M on the given dataset D, similarly $Sensitivity(M, D)$. Sensitivity and Specificity are the

performance measures from the confusion matrix (Section 2.3.1). This notation can be expressed in TNR and TPR as follows:

$$\begin{aligned} \max \quad & TNR \\ \text{subject to} \quad & TPR \geq minTPR \end{aligned} \tag{4.2}$$

, and then:

$$\begin{aligned} \max \quad & 1 - FPR \\ \text{subject to} \quad & (1 - FNR) \geq minTPR \end{aligned} \tag{4.3}$$

, which can written as a minimisation problem:

$$\begin{aligned} \min \quad & FPR \\ \text{subject to} \quad & FNR \leq maxFPR \end{aligned} \tag{4.4}$$

This notation is almost the same used in the constrained optimisation by Yan and Goebel in [106]. I prefer to use maximisation of the measure instead of minimisation of the error and the terms Sensitivity and Specificity, as my personal view is they are easier to explain and follow while reading.

The minimum sensitivity constraint for our case was selected as $minSensitivity = 99.0\%$. The data that were under analysis come from the computer security, and the value of the constraint was selected after the consultation with the domain experts from the computer security field. According to them, this is the value that is necessary to fulfil in order to use the machine learning model in the production environment.

4.2 CC-LRGA:Constrained Classification Based on Logistic Regression and Genetic Algorithm

4.2.1 Method Summary

Motivated by the problem of constraints in the classification of imbalanced data, I designed a method based on Cost-Sensitive Logistic Regression (CS-LR) and optimisation using Genetic Algorithm (shortened as CC-LRGA). First, a meta-learning procedure using CS-LR is run to find initial models for the subsequent optimisation. It uses various cost settings for CS-LR and threshold moving with heuristic optimisations to find these models faster. After initial models are found, they are given as the input solution for the Genetic Algorithm optimisation, which is used to improve the overall quality of the proposed solution.

4.2.2 Theoretical Background

Cost-Sensitive Logistic Regression

Logistic regression was introduced in the Section 2.4. The model can handle imbalanced data in two ways: (1) The threshold moving technique moves the threshold closer to 0 or 1 resulting in the accuracy increase on the class from which the threshold is further away from, which is typically the minority class.

The second technique wraps LR with a cost-sensitive framework. Setting the higher value of C parameter in Equation (2.10) can be used to penalise the misclassifications more [28]. But in this case, we want to penalise different errors with different costs. The error made by the prediction on one example x_i of the loss function in the Equation (2.10) can

be rewritten as:

$$\begin{aligned} err(x_i) &= \log(1 + e^{-y_i \boldsymbol{w}^T x_i}) \\ &= -y_i \log(\tfrac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}_i}}) - (1 - y_i) \log(1 - \tfrac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}_i}}) \end{aligned} \tag{4.5}$$

If we denote the prediction of one example using the logistic function as:

$$h(x_i) = \frac{1}{1 + e^{-\boldsymbol{w}^T x_i}} \tag{4.6}$$

then the equation can be rewritten as:

$$err(x_i) = -(y_i \log(h(\boldsymbol{x})) + (1 - y_i) \log(1 - h(\boldsymbol{x})) \tag{4.7}$$

The first member of the equation with y_i expresses the error for a positive example, and the second part for negative one. The cost of classifying the positive example as negative one, i.e. false negative, is C_{FN}, and the cost of classifying the negative one as positive, i.e. false positive, is C_{FP}. Then we can incorporate these costs into computing the cost function for one example as:

$$err(x_i) = -(y_i C_{FN} \log(h(\boldsymbol{x})) + C_{FP}(1 - y_i) \log(1 - h(\boldsymbol{x})) \tag{4.8}$$

This notation is similar to the one in [4].The overall function that is minimised then counts with these different costs for each error. For binary classification in imbalanced data, the C_{FP} is usually 1, and we need to set only C_{FN} as a cost of missing the minority class.

This cost C_{FN} is in some machine learning libraries also referred as weight w [77]. The reason is that the costs can be assigned in terms of weight vector for each example. For ambiguity with the weight vector of logistic regression, I will denote this vector as we as a weight of the examples. Then, for instance, x_i the value of we_i is the same as the cost of misclassifying it as the wrong class, i.e. C_{FN} for minority class example or 1 for the majority class. Then the equation for cost of one example i can be rewritten according to [55] as:

$$\begin{aligned} err(x_i) &= -(y_i we_i \log(h(x)) + we_i(1 - y_i) \log(1 - h(x))) \\ &= we_i \log(1 + e^{-y_i \boldsymbol{w}^T x_i}) \end{aligned} \tag{4.9}$$

As a consequence, throughout this chapter, I will refer to the cost of misclassifying minority class as a majority one C_{FN} as C. The weight vector of logistic regression will be denoted as $\boldsymbol{w}$.

Genetic Algorithm

Genetic algorithm (GA) is a random optimisation method based on a principle of natural selection and biological evolution. The examined problem is encoded into a set of special genome-like structures. These structures are data strings representing our original binary encoded data models. The core part of any genetic algorithm is a fitness function. A fitness function is able to rank data genomes, ideally, with regards to the desired result of the algorithm. The set of genomes is called a population. The algorithm works iteratively, creating a new altered population in each step, towards the best (highest ranked by the fitness function) possible population. However, as the algorithm is non-deterministic, some stopping criterion should be implemented.

Genetic algorithm uses selection, crossing and mutation techniques to produce a new population. Pure selection enables the algorithm to simply copy excellent individuals from the old population to the new one. Crossing usually combines two individuals by splitting those into (mostly two) parts and joining them to create different individuals. Mutation produces a new genome by randomly altering parts of the old one with regards to alternation constraints (e.g., random new value has to be valid in the changed attribute's domain). Crossing and mutation are applied randomly, and the probability of their occurrence is often adjustable by initial algorithm parameters.

4.2.3 Method Description

The method consists of several consequential blocks. The whole concept can be best understood from the method visualised in Figure 4.1. The method processes labelled input data and returns an optimised model satisfying user accuracy constraints. These constraints can be specified for the accuracy of one, both, or neither of the classes. If such model, satisfying all given constraints, does not exist, the method returns an empty result.

The primary task of the LR block is to find a set of logistic regression models, i.e., attribute weights, bias and a threshold. These models are found with respect to a given accuracy constraint and class imbalance in analysed data.

The GA block utilises these models and uses them as the initial candidate solution for the genetic algorithm optimisation task. Using a particular fitness function and the aforementioned initial candidates, the algorithm begins finding the global optimum, again with respect to given constraints. The main advantage of this approach compared to other methods is speed. There is a need to analyse millions of records in short time periods with a very high minimum Sensitivity constraint; moreover, the final model application to unseen data has to be swift and as simple as possible. The required speed and simplicity of the model application are some of the reasons why classifier ensembles were omitted from the choice. Another significant advantage of the proposed method is a comprehensible result model. In comparison with neural networks or SVMs, the method presents an easy-to-read and understandable model that can be easily applied to unseen data or implemented as a part of another application or system.

Logistic Regression block

The LR block uses the combination of cost-sensitive LR learning and threshold moving to find an initial solution for the GA block. To perform cost-sensitive learning, the block expects a set of costs for the minority class to be delivered with the input data. This set defines interval borders. Inside each of these intervals, the local maximum of the G-Mean is found with respect to specified constraints. This optimisation technique is simple, and it expects that the objective function is either convex or concave in the given interval. The goal is not to find a real maximum but only its rough estimate. Each interval is a binary split until the maximum value increases by a given difference. The usage of more initial costs and thus more intervals where the maximum is searched for allows dealing with local maxima. Also, optimisation is increased with wide variation of the learnt models for GA.

The value of the G-Mean for a given cost is computed as follows: first, the LR model is learnt with C parameter set to the selected cost value. Second, the model is applied to testing data to obtain the confidence vector. Third, the threshold moving technique is used to find the maximum G-Mean value. For each evaluated threshold, the confusion matrix and G-Mean value are computed. The accuracy constraints can be used to find this solution

more quickly, and the interval is divided into bins with an equal size. The final models are then sorted by the G-Mean value, and they are ready for use in the GA block.

Genetic Algorithm Block

On input, the GA block is given models from the LR block, and its main purpose is to make these models more accurate with regards to given class restrictions. It iteratively creates populations of genomes that represent groups of differently weighted models. The core part of the GA block is the fitness function, which ranks all generated models. It is possible to alter the fitness function in such a way that it reflects all the preferences and restrictions.

Although GA is based on random data modification, there are many ways how to alter the optimisation process. Custom heuristics can be easily applied to the mutation and crossing methods; moreover, it is possible to experiment with sizes of initial population, crossing and mutation probabilities, or the number of population cycles.

I defined the fitness function using the following pattern:

$$\begin{aligned} fv &= (Sensitivity * C_1 + Specificity * C_2) * Constr \\ &\quad +(Sensitivity * C_3 + Specificity * C_4), \end{aligned} \tag{4.10}$$

where $Sensitivity$, $Specificity$ denotes minority and majority class accuracy respectively; $Constr \in \{0, 1\}$; $Constr = 1$ when the given constraint is met, otherwise $Constr = 0$; C_1, C_2, C_3, C_4 are user-defined constants that determine the importance ratio between Sensitivity and Specificity, in some cases these constraints can be 0, which depreciates the related class.

The fitness function requires the model to fulfil all required restriction rules. Subsequently, the models are ranked according to their overall classifying performance, favouring the minority class by applying cost-sensitive accuracy computation.

4.2.4 Experimental Evaluation

Experiments were performed over a large dataset from the Internet security research. Examined data are labelled with two highly imbalanced classes. Characteristics of analysed data are summarised in Table 4.1.

Table 4.1: Source data characteristics.

Number of Data Cases	5 000 000
Number of Attributes	120
Attribute type	Binary
Class ratio	1 : 99

As mentioned, the main goal was to achieve at least 99.0% Sensitivity when classifying minority class examples. There were no further restrictions regarding the majority class. The final solution was built to maximise the Specificity while still satisfying the Sensitivity restriction.

The experiments show the behaviour of the proposed method under several parameter settings and their influence on its accuracy and run time. Performed experiments were divided into two parts, examining the properties of the LR block and GA block separately.

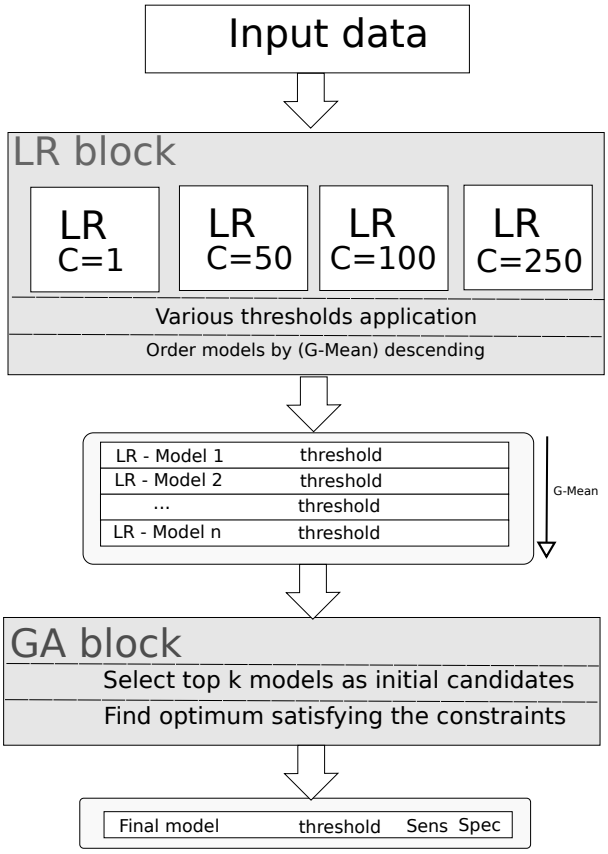

Figure 4.1: Block schema of the CC-LRGA method.

The Logistic Regression block

The main task of the LR block is to provide a set of initial solutions that are good enough as starting models for the subsequent GA block.

In order to provide best candidates for the GA block, both ROC AUC and G-Mean were examined. For approximate ROC AUC computation, it was necessary to evaluate the model using several thresholds. On the other hand, the G-Mean is computed only for one threshold. Both ROC AUC and the highest G-Mean across all the thresholds for 30 different costs have been compared. The results show that there is a high correlation (above 99%) between the two, which can be seen in Figure 4.2. Our experiments also showed that the run time for the ROC AUC computation for given thresholds is considerably longer than for finding the threshold with maximal G-Mean.

We examined two candidate model learning strategies for the LR block. The first one uses constraints, i.e., 99.0% Sensitivity. The second one only finds the unconstrained maximum of the G-Mean, thus making models that don't have to satisfy given constraints. For both strategies, we used the same initial costs for the LR learning phase (ranging from 1 to 1000). The final comparison of both strategies is described by Figure 4.3. We can see that G-Mean values are higher for unconstrained solutions; this is because G-Mean penalises solutions where one class achieves very high accuracy at the expense of the other.

Additionally, the best three candidate models are listed in Table 4.2 for the constrained strategy and Table 4.3 for the unconstrained strategy. These tables reveal that neither of

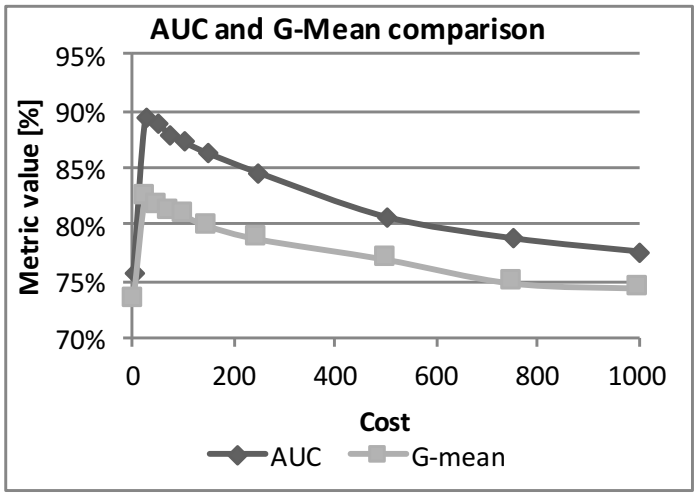

Figure 4.2: G-Mean and AUC values for different costs.

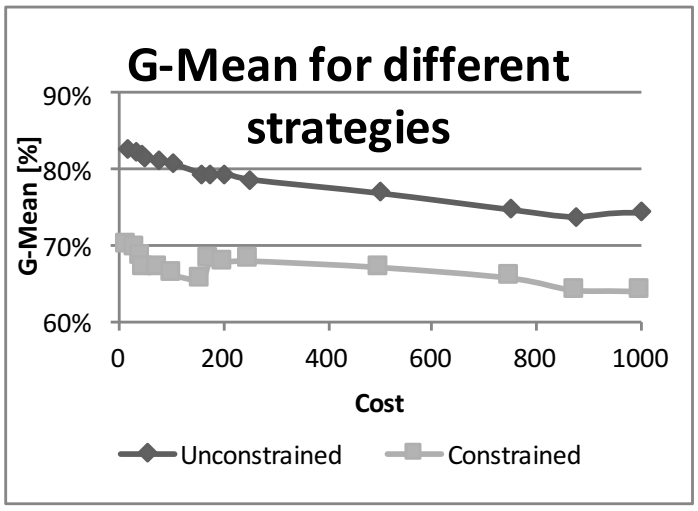

Figure 4.3: G-Mean comparison for constrained and unconstrained learning strategy.

these models satisfies given constraints; however, all of them were successfully provided in our GA block.

In all experiments, we used L2 regularisation for learning the model. We also examined L1 regularisation with very similar results but longer learning time.

It is worth mentioning that besides LR we tried to deploy similar processes using SVMs (Support Vector Machines) and neural networks, however, on our examined dataset, both learning algorithms were unable to finish in a reasonable time frame. Also, Random Forest with under-sampling and naive Bayes algorithms were compared to our solution, with all of them being surpassed by cost-sensitive logistic regression.

Genetic Algorithm block

As previously mentioned, our GA block processes input models generated by the LR block and aims to improve on them with regards to specified constraints. In all performed experiments ($n = 1000$), the use of genetic algorithm improved upon received initial models. In

Table 4.2: Top 3 Models for constrained strategy

Cost	Threshold	Specificity	Sensitivity	G-Mean
32	0.9	0.482	0.991	0.696
225	0.7	0.48	0.99	0.689
175	0.75	0.466	0.466	0.68

Table 4.3: Top 3 Models for unconstrained strategy

Cost	Threshold	Specificity	Sensitivity	G-Mean
13	0.7	0.777	0.881	0.828
25	0.55	0.779	0.872	0.824
33	0.5	0.777	0.874	0.824

Table 4.4 we can see a few real examples of models whose accuracy has been improved by the subsequent use of the GA block.

Table 4.4: Models on the input trained by the logistic regression (Orig.) compared with results optimised by genetic algorithm (GA Opt.).

ID	Orig. Specificity	Orig. Sensitivity	GA Opt. Specificity	GA Opt. Sensitivity
1	0.479	0.989	0.592	0.99
2	0.783	0.869	0.597	0.99
3	0.48	0.99	0.579	0.99

All optimised models satisfy given constraint, i.e., at least 99% Sensitivity. Our experiments also confirm the hypothesis from the previous section that models generated by logistic regression with high accuracy constraints may lead to worse results than models produced without those restrictions, or with lower constraints.

Regarding the course of the GA itself, introduced to models from the LR block, the population improvements are most significant at the beginning of the optimisation phase, which is to be expected. With further iterations, the resulting models converge on their maximum value. An example of such evolution can be seen in Table 4.5.

Table 4.5: Source data characteristics.

Population no.	Specificity	Sensitivity
50	0.562	0.990
100	0.573	0.990
200	0.584	0.990
400	0.590	0.990
1000	0.592	0.990

As can be seen, Specificity rises dramatically at the beginning of the optimisation process. Subsequently, the rise becomes less and less significant, and after extensive iterations of the algorithm we should find a model with the highest possible Specificity. We can experimentally determine an approximate number of re-population cycles in which there is the potential for our optimisation to be maxed out.

4.3 Particle Swarm Optimisation for Constrained Imbalanced Classification

4.3.1 Method summary

Motivated by the presented results of the CC-LRGA method, further optimisations were investigated, for example with the goal of avoiding the randomness caused by the genetic algorithm. Further, in the previous solution, the difference between using the stochastic algorithm without the candidate generating step was not examined. This resulted in a solution solving the problem by using a different optimisation algorithm – Particle Swarm Optimisation (PSO), already employed in various applications in the field of data mining [1].

We proposed two strategies for constrained optimisation using PSO: (1) the penalty function approach, which penalises solutions that do not satisfy the constraints (denoted as PSO1) and (2) using strategy with modified updating, when new solution replaces the current best solution only if it satisfies the constraints (denoted as PSO2). These strategies together with Genetic Algorithm were compared in experiments performed on the same dataset as in the CC-LRGA method.

4.3.2 Theoretical Background - Particle Swarm Optimisation

In contrast to GA, PSO is a typical candidate from the swarm intelligence algorithm family. Swarm intelligence studies the collective behaviour of unsophisticated agents that interact locally with their environment [11]. The research takes inspiration from a social behaviour of insects such as ants or bees, or a flock of birds. PSO was first used to simulate birds searching for food. Every bird in a population is denoted as a particle and the whole population of particles as a swarm. The basic idea of PSO is that every particle has its velocity, which is continuously updated based on two factors – 1) towards own personal best solution and 2) towards the best solution of particles in its neighbourhood (the neighbourhood can be defined in various ways). In the global variant, the particle moves toward the best solution of all the particles in the swarm. The initial particles are initialised by random, and then, continuous update of their velocities causes the swarm to move towards the desired optimum.

More formally, a new updated position x_i of the particle i is computed as follows:

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{4.11}$$

where $i = 1, 2, \ldots, N$ and N is the size of the population; x_i^t is the old position and v_i^{t+1} is the current velocity, which is calculated as:

$$v_i^{t+1} = \chi(\omega v_i^t + c_1 r_{i1}^t (pBest_i^t - x_i^t) + c_2 r_{i2}^t (gBest^t - x_i^t)), \tag{4.12}$$

where $pBest$ and $gBest$ are personal and global best solution; χ is a constriction factor used to control velocities; ω is the inertia factor; c_1 and c_2 are two learning factors, called *cognition* and *social* respectively; r_{it}^1 and r_{it}^2 are uniformly distributed random numbers between 0 and 1.

4.3.3 Modifications for Constrained Imbalanced Learning

In a general optimisation problem, there are usually mechanisms for dealing with constraints. In [48] , feasible initial solutions for PSO were generated by random. Unfortunately, in some cases, this approach has a high computation cost. This method is slightly similar to our solution in a way that we also feed the initial swarm population with some feasible or almost feasible solutions. In contrast to their method, we do not generate it randomly but with the use of Cost Sensitive Logistic Regression. Other researchers in [15] tried to handle the problem with small population size. We dealt with the constraints using two strategies – 1) the penalty function approach, which is based on the work of [76] and 2) using strategy with modified updating.

Penalty Function

The idea of this approach is penalising the solutions violating given constraints. In a maximisation task, the penalty value is subtracted from the original objective function, and the resulting penalty function becomes a new objective. According to [76], the penalty function is defined as:

$$F(x) = f(x) - h(k)H(x), \qquad x \in S \subset \mathbb{R}^n, \tag{4.13}$$

where $f(x)$ is the original objective function, $h(k)$ is dynamically modified with the current algorithm iteration k and $H(x)$ is the penalty factor, defined as:

$$H(x) = \sum_{i=0}^{m} \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))}, \tag{4.14}$$

where $q_i(x) = \max\{0, g_i(x)\}$, $i = 1, \ldots, m$ and is denoted as relative violated function; $\theta(q_i(x))$ is multi-stage assignment function; $\gamma(q_i(x))$ the power of the penalty function and $g_i(x)$ are the constraints in the form:

$$g_i(x) \leq 0, \quad x \in S \subset \mathbb{R}^n. \tag{4.15}$$

For our purpose, we have only one constraint: $sensitivity > minSensitivity$ and following the given form $minSensitivity - sensitivity \leq 0$. The functions $h(.)$, $\theta(.)$ and $\gamma(.)$ are problem dependent. In experiments we used $h(k) = 1$, resulting in the same penalty factor. For the two other functions we adapted and modified the strategy that was recommended in [108] and [76]. The $\gamma(q(x))$ is computed as:

$$\gamma((q(x)) = \begin{cases} 1 & \text{if} \quad 0 < q(x) < 1 \\ 2 & \text{if} \quad q(x) \geq 1 \end{cases} \tag{4.16}$$

The $\theta((q(x))$ function is defined as follows:

$$\theta((q(x)) = \begin{cases} 5 & \text{if} \quad 0 < q(x) \leq 0.001 \\ 10 & \text{if} \quad 0.001 < q(x) \leq 0.1 \\ 50 & \text{if} \quad 0.1 < q(x) \leq 1 \\ 100 & \text{if} \quad q(x) \geq 1 \end{cases} \tag{4.17}$$

Recall that $q(x) \geq 0$ and, as expected, for $q(x) = 0$ the penalty $H(x)$ is always 0. To sum up, the optimisation process is the same as for PSO described in the previous section;

the only difference is the modified objective function with a penalty. Therefore, in each step of the algorithm, the personal best and the global best of each particle is computed according to the function with penalisation in Equation 4.13. Then the velocity of the particle is updated according to the Equation 4.12.

Strategy with Modified Updating

As opposed to the previous unchanged optimisation process, this approach slightly modifies updating strategy of the personal best solution of a particle. Typically, the personal best solution is updated, if the new particle has a better value of an objective function. In our case, this means greater specificity. However, the update does not account for constraints, so the personal best is replaced by the new one if at least one of the following condition is true:

1. The global best does not satisfy the sensitivity constraint, and the new particle has greater sensitivity than its personal best,

$$\begin{aligned} &gBest.Sensitivity < minSensitivity \wedge \\ &x_{i+1}.Sensitivity > pBest.Sensitivity. \end{aligned} \tag{4.18}$$

2. The particle has greater specificity than its best, and either the particle or the best neighbour satisfies the constraint.

$$\begin{aligned} &x_{i+1}.Specificity > pBest.Specificity \wedge \\ &(x_{i+1}.Sensitivity \geq minSensitivity \\ &\vee\, gBest.Sensitivity \geq minSensitivity) \end{aligned} \tag{4.19}$$

Here, the dot notation is used for the measure of a particle, thus $gBest.Sensitivity$ is a sensitivity of the global best particle. At first, the algorithm tries to reach a solution that satisfies the conditions. After such a solution is found, particles follow other particles with greater specificity.

4.3.4 Evaluation

Experiments were conducted over the same dataset as for the CC-LRGA method, that is the dataset from the Internet security research. We were able to obtain additional approximately 70 000 positive class examples, making the imbalance ratio lowered from 1:99 to 1:34. The attributes used for training the models remained the same.

In the experiments, we focused on comparing the behaviour of the two strategies used for constrained optimisation and comparing with the Genetic Algorithm. Also, I provide the comparison of learning from random initial weights to initial models from Cost-Sensitive Logistic Regression (CS-LR).

I conducted two types of evaluations. The first one was performed with a 5-fold cross-validation and the second one shows where the algorithms converge after the reasonable number of iterations on the training data. In both types of evaluation, three different types of initial candidates were given as the input of the optimisation algorithms:

1. Solutions with highest G-Mean from CS-LR denoted as Uncons. (unconstrained) in tables,

2. solutions with highest G-Mean satisfying the *minSensitivity* constraint, denoted as Cons., and

3. uniformly distributed random weights denoted as Rand.

In all experiments, the size of the swarm in PSO was set to $n = 70$ and population $n = 5000$ for GA. Moreover, I present a behaviour of the strategies in graphic form to demonstrate the growth of objective functions in the first 300 iterations. In the following tables 4.6 and 4.7, GA stands for Genetic Algorithm, PSO1 is PSO with the penalty strategy and PSO2 the strategy with modified updating.

Cross-Validation

The performance of a classifier is typically evaluated with the cross-validation (CV). Here we performed the 5-fold stratified CV, where each fold has the same class ratio, which is necessary for imbalanced data. First, 1000 iterations of the optimisation algorithm were performed and then the resulting model was evaluated using testing data. Because of the stochastic character of the algorithms, we also ran the entire CV procedure 5 times with the same folds. We took the mean and the standard deviation of all the objective function results and also mean and standard deviation of the maximum solutions for each fold. The results are in Table 4.6.

We can see that best results were achieved by penalty strategy for PSO given initial models satisfying the *minSensitivity* constraint. This stands for the mean and the maximum. The Genetic algorithm performed slightly worse. Updating penalty had the worst results, especially when given models satisfying the *minSensitivity* constraint. Only initial models with the greatest G-Mean achieved a mean value above 0.55.

Comparing strategies, it is possible to conclude that PSO1 had better results with Cons. initial models, PSO2 with the Cons. GA has very similar results for both initial models. Starting from randomly generated initial solutions, all the approaches achieved lower objective function values than with initial models from CS-LR. The greater standard deviation value for starting from random solution shows that the solutions are much less stable. It is worth mentioning that the sensitivity constraint was narrowly violated in some measurements but did not drop under 0.9899.

	Mean		
	GA	PSO1	PSO2
Uncons.	0.5661 ± 0.0049	0.5641 ± 0.0081	0.5596 ± 0.0029
Cons.	0.5536 ± 0.0055	$\mathbf{0.5834 \pm 0.0037}$	0.4988 ± 0.0001
Rand.	0.5452 ± 0.0127	0.4849 ± 0.0388	0.4970 ± 0.0276
	Maximum		
	GA	PSO1	PSO2
Uncons.	0.5682 ± 0.0039	0.5679 ± 0.0058	0.5618 ± 0.0024
Cons.	0.5595 ± 0.0042	$\mathbf{0.5855 \pm 0.0018}$	0.4989 ± 0.0000
Rand.	0.5618 ± 0.0091	0.5353 ± 0.0165	0.5271 ± 0.0137

Table 4.6: The results from the cross-validation for optimised Specificity.

Training and Testing on the Same Dataset

In this section, I show how the algorithms were able to produce good solutions during the optimisation. In terms of classification evaluation, this meant evaluating on the training dataset. For each method, the mean, the best and also the worst solution achieved in 10 runs were obtained and reported in Table 4.7. As expected, the results show that the values are higher than in cross-validation. For almost all cases the values slightly correlate with the previous evaluation. The difference is the PSO2 strategy, which performed significantly better than in cross-validation. It is also evident that in all cases, starting from initial weights from the CS-LR leads to better results than from random weights.

	Mean		
	GA	PSO1	PSO2
Uncons.	0.5672 ± 0.0029	0.5469 ± 0.0130	0.5804 ± 0.0091
Cons.	0.5548 ± 0.0003	$\mathbf{0.5817 \pm 0.0074}$	0.5373 ± 0.0046
Rand.	0.5469 ± 0.0179	0.5506 ± 0.0190	0.5290 ± 0.0235
	Maximum		
	GA	PSO1	PSO2
Uncons.	0.5700	0.5603	**0.5930**
Cons.	0.5601	0.5871	0.5426
Rand.	0.5680	0.5748	0.5566
	Minimum		
	GA	PSO1	PSO2
Uncons.	0.5620	0.5286	0.5679
Cons.	0.5412	0.5666	0.4916
Rand.	0.5200	0.5221	0.4923

Table 4.7: The results from the evaluation with training and testing on the same dataset.

Behaviour of the Algorithms

To demonstrate the different behaviour of the approaches, we provide the following graphs with objective function value in the first 300 iterations. For each approach, i.e. GA, PSO1, PSO, there is separate figure each containing the three types of initial solutions. The first graph in Fig. 4.4 shows the GA, Fig. 4.5 the PSO1 and Fig. 4.6 the PSO2. The GA and PSO1 have similar behaviour for all of the initial solutions. For Cons., the initial solution already satisfies the constraint and thus the performance only grows, with steeper growth in PSO1. For Uncons. the objective function falls rapidly in the beginning because the initial solutions do not satisfy the constraint. PSO1 begins to rise earlier than GA. For Rand. we can observe faster growth in the objective function for GA.

Starting from random solution, the modified version of updating strategy (PSO2) has similar behaviour as the other two, only the growth is much slower. For Cons. it very quickly gains local maximum but it is unable to find another solution. On the other hand, for Uncons. it grows slower but outperforms the Cons. initial models.

We also experimented with the neighbourhood of particles. The local setting with 10 closest neighbours worked better compared to taking the whole population.

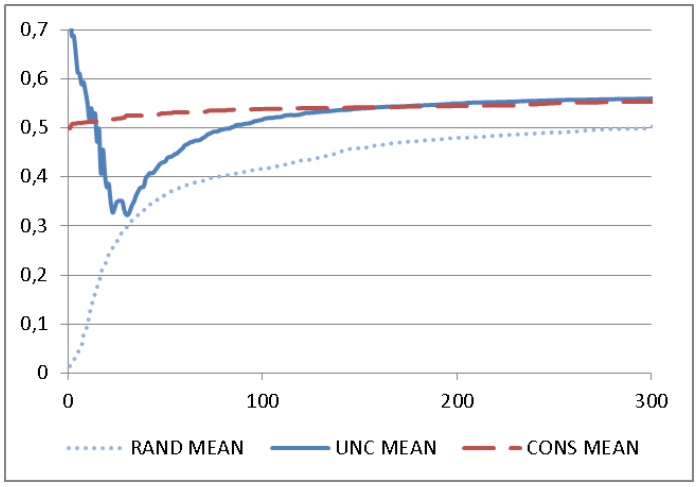

Figure 4.4: First 300 iterations of GA.

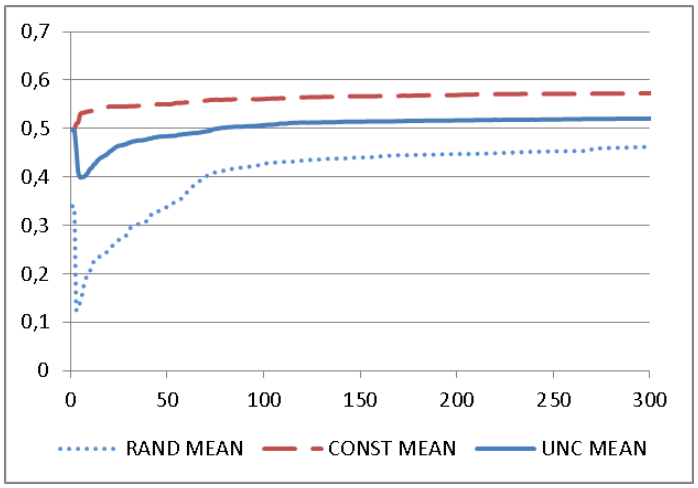

Figure 4.5: First 300 iterations of PSO with the penalised function – PSO1.

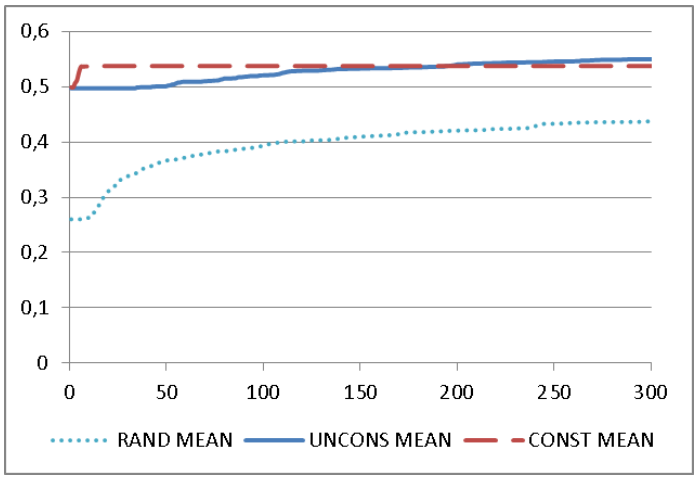

Figure 4.6: First 300 iterations of PSO with modified update strategy – PSO2.

4.3.5 Evaluation Summary

Experiments were conducted using 5-fold stratified Cross-Validation. Because of the stochastic character of the algorithms, the whole CV procedure was run 5 times with the same folds. The mean and the standard deviation of all the objective function results were taken as well as the mean and standard deviation of the maximum solutions for each fold. The results are summarised in Table 4.6. The experiments showed that the penalty function approach for PSO outperformed the Genetic Algorithm and together with the initial weights providing by CS-LR, it can converge quickly to a very good solution with specificity around 0.58. Cross-validation also showed that the constraint is violated only narrowly while provided unknown data. It is worth mentioning that the sensitivity constraint was narrowly violated in some measurements but did not drop under 0.9899.

4.4 Related Work

Three similar problems were published in [16] using PSO, [106] utilising GA and [96] using ROC isometrics. They have been presented and discussed in the *State of the Art* chapter in Section 3.2. In the further subsections, these are compared with my solution of constrained classification. Moreover, several other approaches of how GA and PSO have been utilised in data mining are introduced.

4.4.1 Genetic Algorithms

In, [106], the problem was almost similar, but the data were not imbalanced and were much smaller. Opposed to my approach, they incorporated the constraints in the data preparation phase using the genetic algorithm, and then they used an ensemble of Neural Networks to train the classifier.

The use of Genetic Algorithms for data mining has been proposed in [53]. The potential of this approach lies in the search performance of the algorithm, which is very easy to parallelise. There is also no need for dividing data into training and testing sets since the algorithm trains and tests itself on the same dataset. Presently, a genetic algorithm is often used as an optimisation technique for other data mining methods, mostly for feature subset selection [107, 106], in hybrid decision structures [17], or for rule induction [74].

Some approaches already combine genetic algorithm with logistical regression; however, previous methods use the genetic algorithm mostly just as an optimisation technique for adjusting costs in cost matrix, or reduction of feature space [107, 97]. I proposed a different approach where both methods are used subsequently to achieve better results than other techniques.

4.4.2 Particle Swarm Optimisation

At about the same time as my research, Cao et al. published [16], where they constructed a Neural Network for imbalanced data classification using Particle Swarm Optimisation, optimising the misclassification cost. The outer and inner phase of PSO was introduced, the outer one is used for the parameters or architecture of the neural network and the inner one for the weights of the neural network. They compared the approach to the existing sampling methods, and their solution outperforms them in terms of G-Mean for binary classification and G-Mean and ROC AUC for the multiclass problem. In comparison to my research, their solution did not count with any constraints. Moreover, they used PSO not

only for constructing the classifier but also for feature selection, and their approach counted both using the binary and the multiclass problem.

The PSO has been used in data mining for various purposes. For classification, in [100] an algorithm for mining classification rules based on PSO was proposed. More usages of PSO and other swarm intelligence methods can be found in [1]. Besides PSO, there are also two known algorithms from the swarm intelligence family that are used for data mining – Ant Colony Optimisation (ACO) algorithm and Artificial Bee Colony.

4.4.3 ROC Isometrics

In contrast to previous methods, in [96], Vanderlooy et al. specified the constraints differently. Instead of maximising the performance on one class, given a constraint on the other one, they put constraints on both of the classes and excluded the data, where the classifier is not confident about the result. The goal was then to satisfy the constraints and minimise the amount of data that is removed in order to satisfy the constraints. The details of the approach can be found in Section 3.2.

Chapter 5

Goal achieving problem with the deadline

This chapter introduces the problem of predicting which objects out of a finite set achieve a goal within a predefined time interval. For example, taking the objects as students who registered for a course whose goal is to submit their assessment. The time interval is defined as the start time and deadline date for homework submission. Start time begins when the students are given the instructions for the assessment task, and ends at the given deadline date and time.

At the start, no object has achieved the goal, but by the deadline a certain number of objects between zero and all participating objects will have achieved the goal. Objects are described by a set of attributes and their values. Some attributes change their value during the start-to-deadline interval, other attributes might remain static. The objective is to use the available values of attributes and to predict as early as possible whether each object will achieve the goal by the deadline. Depending on the domain, this early identification can allow influencing the subjects to encourage them towards achieving the goal, e.g. customers buying a product or students submitting the assessment. The predictive engine is expected to apply machine learning methods to the available data.

The problem is a dichotomy, which is inherently imbalanced. At the start, no object has achieved the goal, and all the data are examples of a single, unachieved class. Obviously, examples of the achieved class are missing. With the increasing time, the number of achievers (examples of objects which have satisfied the goal) increases. The imbalance ratio decreases, the models will have more information for training, and an increase in model performance can be expected.

Because the set is finite and not changing, once half of the population reaches the goal, the ratio can start increasing as fewer non-achievers than achievers will remain. However, as it will turn out, the main problem connected with the imbalance is the lack of information in the beginning. Also, the case study in the following section will show that in human behaviour, the presence of a deadline usually leads to an increase in the activity before the deadline, with most of the objects achieving the goal just before the end time [57].

This chapter is structured as follows: first, the problem of predicting achievement of the goal is described, then the Self-Learning approach is presented together with the evaluation strategy and with the issues that result from the imbalanced nature of the problem.

5.1 Problem Description

Let D be a set of objects x_i, $D = \{x_1, x_2, \ldots, x_N\}$, where x_i is an object represented by an m-dimensional feature vector $x_i = (x^1, x^2, \ldots, x^m)$, i.e. an object described by m features (or attributes) $A^1, A^2, \ldots, A^m$. These attributes can be either a numerical or categorical type. Let G be a set of goals g_k, $G = \{g_1, g_2, \ldots, g_K\}$ and time be discrete starting at point t_0. The goals of G can by achieved by the objects in D in time $t \in [t_0; t_d]$, where t_d is called the deadline. Let's denote achieving the goal g_k by the object x_i in time t by a predicate

$$Achieved(x_i, g_k, t). \tag{5.1}$$

For example, a customer Mark who made a purchase on 24th December 2010 would be denoted as $Achieved(Mark, Purchase, 24Dec2010)$; a student John, who submitted the first assessment on the 10th day of the course as $Achieved(John, SubmitA1, 10)$. [1]

To specify that the goal g was achieved by the object x before or at time t, let's define the predicate $AchievedBy$ as:

$$AchievedBy(x, g, t) = \begin{cases} True & \text{if} \quad \exists t_i : Achieved(x, g, t_i), t_0 \leq t_i \leq t \\ False & \text{otherwise} \end{cases} \tag{5.2}$$

The set of objects that have achieved the goal before or at time t is defined as:

$$DA(D, g, t) = \{x | x \in D, AchievedBy(x, g, t) = True\}. \tag{5.3}$$

Analogously, the set of objects from D that have not achieved (unachieved) the goal at the time t is defined as:

$$\begin{array}{rcl} DU(D, g, t) & = & \{x | x \in D, AchievedBy(x, g, t) = False\} \\ & = & D \setminus DA(D, g, t) \end{array} \tag{5.4}$$

Next, the number of objects that achieved or unachieved the goal g up to time t is:

$$\begin{array}{c} NumAchievedBy(D, g, t) = |DA(D, g, t)| \\ NumUnachievedBy(D, g, t) = |DU(D, g, t)|. \end{array} \tag{5.5}$$

Let us assume, that in the beginning, $t = t_0$, none of the objects has achieved the goal, i.e.

$$\begin{array}{rcl} NumAchievedBy(D, g, t_0) & = & 0 \\ NumUnachievedBy(D, g, t_0) & = & |D| \end{array} \tag{5.6}$$

and the time of the first achievement t_{first} for set D and goal g with the deadline t_d is defined as:

$$t_{first} = \min\{t | t \in [t_0, t_d] \wedge NumAchievedBy(D, g, t) > 0\} \tag{5.7}$$

Example 5.1 From this moment on, the running example of students submitting their assessment in a course will be used to support the description of the problem definition.

Let us have a set of seven students $D = \{s_1, s_2, \ldots, s_7\}$ with the goal of submitting the assessment denoted as $g = g_1$ having the deadline in $t_d = 10$. The time is measured since time $t_0 = 0$. The student s_1 submits the assessment in $t = 3$, s_2 and s_3 in $t = 7$,

[1] This notation was used for simplifying the explanation. Formally, it would be an object x_i, where one of the attributes in x_i is the name.

the students s_4, s_5, s_6 in the deadline $t = 10$. The student s_7 does not submit at all. Then $t_{first} = 3$, and for $t \in [7, 8]$

$$\begin{array}{rcrcl} DA(D, g_1, 7) & = & DA(D, g_1, 8) & = & \{s_1, s_2, s_3\}, \\ DU(D, g_1, 7) & = & DU(D, g_1, 8) & = & \{s_4, s_5, s_6, s_7\} \\ NumAchievedBy(D, g_1, 7) & = & NumAchievedBy(D, g_1, 8) & = & 3 \\ NumUnachievedBy(D, g_1, 7) & = & NumUnachievedBy(D, g_1, 8) & = & 4 \end{array}$$

Let's suppose that the objects achieve the goals independently on each other. The number of objects achieved the goal before or on time (i.e. $NumAchievedBy(g, t, D)$ is a non-decreasing function with the maximum reaching in the deadline t_d:

$$\begin{aligned} \forall t_i, t_j \in [t_0; t_d], t_i \leq t_j :: \quad & NumAchievedBy(D, g, t_0) \\ & \leq NumAchievedBy(D, g, t_i) \leq NumAchievedBy(D, g, t_j) \\ & \leq NumAchievedBy(D, g, t_d) \end{aligned} \tag{5.8}$$

Analogously, the $NumUnachievedBy$ is a non-increasing function, as each object, after achieving the goal, is moved from DU to DA. The imbalance ratio IR is usually defined as a ratio between the majority and the minority set, i.e. the maximum and minimum of the set sizes in our case as:

$$IR(D, g, t) = \frac{\max[NumAchievedBy(D, g, t), (NumUnachievedBy(D, g, t)]}{\min[NumAchievedBy(D, g, t), (NumUnachievedBy(D, g, t)]} \tag{5.9}$$

In the beginning, the majority set is the DU until the moment where the number of achieved objects reaches 50% of the objects in D. Let's denote this time as t_{eq}. Let's also expect that not all objects will achieve the goal by the deadline. Then the IR function is defined in $[t_{first}; t_d]$[2], where t_{first} denotes the first achievement of the goal. For $t < t_{first}$, the function is undefined. The function is non-increasing in $[t_{first}, t_{eq}]$ and non-decreasing in $[t_{eq}, t_d]$. Hence, the majority and the minority set can exchange their roles in time, i.e. majority set will become a minority and vice versa. However, depending on the domain, such a case might not happen, especially if the majority of the objects achieve the goal at the last minute before the deadline. The problem becomes more interesting when the number of achievers and non-achievers approaches equal, as less information about the reasons for achievement is available.

5.1.1 Goal Achievement Prediction Problem

For the goal g, the set of objects D, start time t_0, the deadline time t_d and the prediction time $t_p \in [t_0, t_d)$ I define the task as a binary classification problem of achieving the goal before or at the deadline, *Goal Achievement Prediction Problem (GP)* as:

$$GP = (D, g, t_d, t_0, t_p, cpm), \tag{5.10}$$

, where cpm is a classification performance measure that we want to optimise, defined as a function:

$$cpm(y_{pred}, y_{true}) \tag{5.11}$$

[2]If we expect all objects to achieve the goal before the deadline, the function would be defined until $\min(t_d, t_{last})$ with t_{last} being the last achievement time.

y_{pred} denotes the the vector of predictions for the objects in the testing data and y_{true} their true class labels. The examples of *cpm* are Accuracy, PR AUC and other measures presented in the Section 2.3. The testing data consists of the objects that have not achieved the goal before or at time t_p, defined by the function $DU(D, g, t_p)$, see Equation 5.4. For such objects $x \in DU(D, g, t_p)$ at time t_p, the target classes are defined as:

$$class(x, g, t_p, t_d) = \begin{cases} Achieve & \text{if} \quad AchievedBy(x, g, t_d) \\ NotAchieve & \text{if} \quad \neg AchievedBy(x, g, t_d) \end{cases} \tag{5.12}$$

In other words, the goal is to find the model approximating the *class* function, i.e. predicting goal achievement within the deadline for the objects that have not achieved the goal by the prediction time. Moreover, we are interested in finding a scoring classification model, i.e. the one providing confidence of the object being a member of the class (see Section 2.2). Notice that for t_p, the available data are known and the first unknown time that we are predicting in t_p is $t_p + 1$. The true values of the classes are known just after the deadline passes, they might not be necessary to train the models, but they are important in order to evaluate the problem.

Example 5.2 Following the running example 5.1 with the start t_0 and the deadline $t_d = 10$, the performance measure we are interested in is *ROC AUC* (now shortened as *AUC*). The problem for $t_p = 7$ is depicted in Figure 5.1, i.e. $GP = (D, g_1, 10, 0, 7, AUC)$. The predictions are computed for days 0 - 7, i.e. the last prediction day is $t_p = 7$. The number of days to the deadline for which the predictions are made is $t_d - t_p = 10 - 7 = 3$.

Moreover, the last day for prediction can be $t = t_d - 1 = 9$, the prediction problem is defined as $GP = (D, g_1, t_d, 0, t_d - 1, AUC) = (D, g_1, 10, 0, 9, AUC)$ and the predictions are computed for only one day, i.e. one day before the deadline t_d.

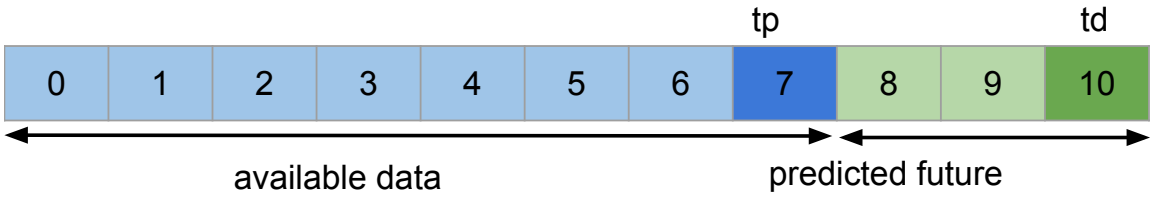

Figure 5.1: Time line for the problem in the 7th day, with the deadline denoted as t_d (dark green), the day of prediction as t_p (dark blue). Known past (training) data are highlighted in blue, predicted future is in green.

5.1.2 Backward Aligned Problem

Without the loss of generalisation, instead of counting the time of prediction from the beginning, let us define it relative to the deadline, i.e. $t_d - t$. In this way, the *Goal Achievement Prediction Problem Relative* (GP^R) can be defined as

$$GP^R = (D, g, t_d, t_0, t_p^R, cpm) \tag{5.13}$$

, where t_p^R is the prediction time relative to the deadline such as $0 < t_p^R \leq |t_d - t_0|$. The superscript R will be used from here on to emphasise that the time is counted relatively

from the deadline. The other parts are the same as in the previous definition. Also, $t^R_{first} = t_d - t_{first}$ denotes the first day with the goal achievement in a relative manner. Similarly, $t^R_d = 0$ will denote the deadline and $t^R_0 = t_d - t_0$ the time t_0 of the problem.

Example 5.3 Analogously to the previous Example 5.2 with absolute ordering, Figure 5.2 depicts the same problem but with relatively defined prediction time $t^R_p = 3$. The relative definition of the prediction day will be more intuitive for defining how to compute the predictions. Such problem is defined as $GP^R = (D, g_1, 10, 0, 3, AUC)$.

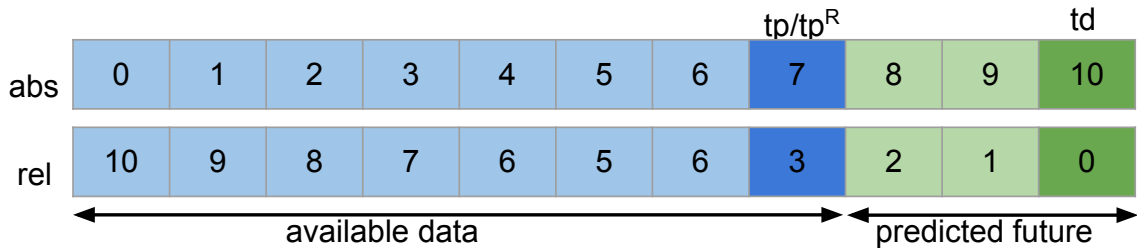

Figure 5.2: Time line for the problem 3 days before the deadline, depicted in both absolute (top) and relative (bottom) time counting. The deadline t_d is in day 10 (dark green), prediction time is t_p for absolute counting, t^R for relative counting.

Composite Problem

The composite problem denotes a composition of problems for all of the available prediction times, which for relative counting is $0 < t^R_p \leq |t_d - t_0|$. Let us define the problem using the backwards alignment relative to the deadline as *Composite problem* CGP^R as:

$$CGP^R \quad = \quad (D, g, t_d, t_0, cpm) \tag{5.14}$$

Example 5.4 Following the running example, the composite problem is defined as $CGP^R = (D, g_1, 10, 0, AUC)$. The problem is defined in times $t^R_p \in [1, |t_d - t_0|]$, i.e. in $[1, 10]$. The first prediction will be 10 days before the deadline and the last one the day before.

5.1.3 Problem Formulation

The goal is to find a method being able to construct a predictive model in each time after the first goal achievement, i.e. for the composite prediction model. In an ideal situation, such a method will perform better in all the given prediction times. In case different methods provide best results in different parts of the predicted time interval, the reason for that should be investigated.

One key issue is the selection of a performance measure for the prediction problem. This issue is highly related to the presence of imbalanced data and will be discussed later in Section 5.3. For a selected classification performance measure *cpm* and a trained predictive model m, the problem performance measure *ppm* for GP^R is computed by: (1) retrieving the predictions by applying the model to the testing data and (2) calculating the classification performance measure *cpm* using the predictions and true labels. This can be denoted as

$$ppm = f(GP^R, m) = f(D, g, t_d, t_0, t^R_p, cpm, m)$$

Theoretically, it is possible to provide predictions in all the times, but the meaningful models can be computed only after the first object achieves the goal in t^R_{first}. In the running example, the first student submitted on day $t = 3$, i.e. in $t^R_{first} = 10 - 3 = 7$, so it is meaningful to evaluate the composite problem only in relative days $[1, t^R_{first}] = [1, 7]$.

Thus, *ppm* for a composite problem CGP^R is defined as the mean over all the prediction times $t \in [1, t^R_{first}]$ as:

$$\begin{aligned} ppm(CGP^R, \boldsymbol{m}) &= ppm(D, g, t_d, t_0, cpm, \boldsymbol{m}) = \\ &= \frac{1}{t^R_{first}} \sum_{t=1}^{t^R_{first}} ppm(D, g, t_d, t_0, t, cpm, m_t) \end{aligned} \tag{5.15}$$

, where $\boldsymbol{m}, |\boldsymbol{m}| = t^R_{first}$ denotes the vector of trained models for each prediction time $t \in [1, t^R_{first}]$, i.e. denoted as m_t. Usually, we will be interested in the performance of one learning algorithm, e.g. logistic regression, which will produce in each time t a different predictive model m_t.

Multiple Composite Problems

Let's consider a set of n composite problems $C = \{CGP^R_1, CGP^R_2, \ldots, CGP^R_n\}$ with the same achievement goal g, and we want to evaluate the performance of a model on all of the problems. This is possible by extending the Equation 5.15 by summation over all the composite problems. Since these problems differentiate only by the dataset and the deadline, two modifications are performed. First, the sum is defined over the set of datasets $R = \{D_1, D_2, \ldots, D_n\}$, where D_i corresponds to the dataset in the composite problem CGP_i in C. Analogously t_{di} and t_{0i} refers to the deadline and the start time in the problem CGP_i. Second, the minimum of the first times of achievement t^R_{first} over all the problems is selected so that the evaluation is aligned with the same number of days. Let's denote this as t^R_{min0}. Then the performance measure is defined as:

$$ppm(C, \boldsymbol{MO}) = \frac{1}{n \cdot t^R_{min0}} \sum_{i=1}^{n} \sum_{t=1}^{t^R_{min0}} ppm(D_i, g, t_{di}, t_{0i}, t, m_{i,t}) \tag{5.16}$$

, where $\boldsymbol{MO}$ is a $n \times t^R_{min0}$ matrix of trained models, defined analogously as in for CGP_R, but extended across all the courses. Hence, a predictive model $M_{i,t}$ refers to the model trained on the dataset D_i in the relative prediction time t. Recall that C denotes the set of n composite problems.

Example 5.5 In case of students submitting assessments, we might be interested in the average performance measure for the first assessment $g = g_1$ of three courses. Each course is described by a dataset, i.e. $= R = \{D_1, D_2, D_3\}$. If the first submission occurs in times $t^R = 7$ for the course D_1, $t^R = 5$ for D_2 and $t^R = 6$ for D_3, then $t^R_{fmin} = 5$. Hence, the performance measure would be mean of $3 \times 5 = 15$ values, i.e. mean over 15 models. As mentioned, these models will usually be trained using one type of learning algorithm, such as logistic regression.

Multiple Composite Problem vs. Temporal View

In addition to a global performance calculated across all time points and all the datasets, we define the average view across all datasets in the given time point t^R. Then, the performance measure 5.16 changes to:

$$ppm(C, \boldsymbol{m}, t_p^R) = \frac{1}{n} \sum_{i=1}^{n} ppm(D_i, g, t_{di}, t_{0i}, t_p^R, m_i) \tag{5.17}$$

, where the size of the model vector is $|\boldsymbol{m}| = n$, m_i refers to the the model of the dataset D_i in the given prediction time t_p^R.

In chapter 6 we investigate the performance according to 5.17, while in chapter 7 we use the concept introduced in the equation 5.16 to measure the proposed improvements.

5.2 Self-Learning Framework

To utilise the data that are already available at the time of computing the predictions, I propose a Self-Learning framework to create a discriminative model to predict achievement of the goal within the specified deadline.

Let's assume that the behaviour of objects which achieve the goal closer to the deadline follows a similar pattern as those who have already achieved the goal in advance; and also differs from the objects that will not achieve the goal within the deadline. Then, it is possible to use the patterns of the objects available in the time of the prediction, especially those that achieved the goal. Although the distribution of the achievement of the future is not known, the number of objects will be nondecreasing.

5.2.1 Extending Labelling Window

Given the problem $GP^R = (D, g, t_d, t^R)$, $t_p^R = 0$ denotes the deadline in a relative manner. To be able to create the prediction model for interval $[t_p^R + 1; 0]$ it is essential to have labelled examples to be used as the training data. As the classifier can be sure only about the objects that achieved the goal, an artificial labelling interval will be created, which will be used to measure if the goal was achieved or not, given the features from the data before the interval starts.

To simulate the problem as occuring in the prediction time t_p^R, the suitable choice of the interval size appears to be the same size as the time remaining to the deadline, i.e. the labelling window will be defined as:

$$w_{label} = [virt\_t_d^R; virt\_t_p^R - 1] = [t_p^R; 2t_p^R - 1] \tag{5.18}$$

The $virt\_t_p^R$ and $virt\_t_d^R$ can be considered to be a virtual prediction time and a virtual deadline. The features measured in this interval will not be used for training data. Instead, the features will be taken from the beginning of gathering the data until the last time before the interval start, the most recent day for the features is $virt\_t_p^R$.

Notice that the creation of the virtual deadline allows using the existing definition, how to assign the class labels in the Equation 5.12. It's only necessary to substitute the

real deadline t_d with the virtual one and map the relative time to the absolute one, i.e $t_d - virt\_t_p^R$.

Example 5.6 The example of the problem is depicted in Figure 5.3 in the top part a). Here, the deadline is within three days from the prediction time ($t_p^R = 3$), and we want to predict if the goal will be achieved either on day $t^R = 2$ or in the next days (the orange area). The labelling window for measuring the achievement of the goal is three days long, i.e. in days $[t_p^R; 2t_p^R - 1] = [3; 5]$ (the green area). Therefore, the values of the features used for training will come from days $[virt\_t_p^R; t_0^R] = [6; 10]$ (the blue area), i.e. the most recent day to be considered for gathering features in the training data will be day 6.

The bottom part of Figure 5.3 b) shows the relative view of the days for training and testing data. The $day = 3$ denotes the current prediction day, the green area depicts the predicted interval until the deadline, and the blue area the days from which are extracted values of the features. This view shows the shift in the training and testing data. For example, values of the features for $day = 3$ in the training data relate to the $day = 6$ in the testing data, because the data for training are aligned towards the virtual deadline $virt\_t_d^R = 3$.

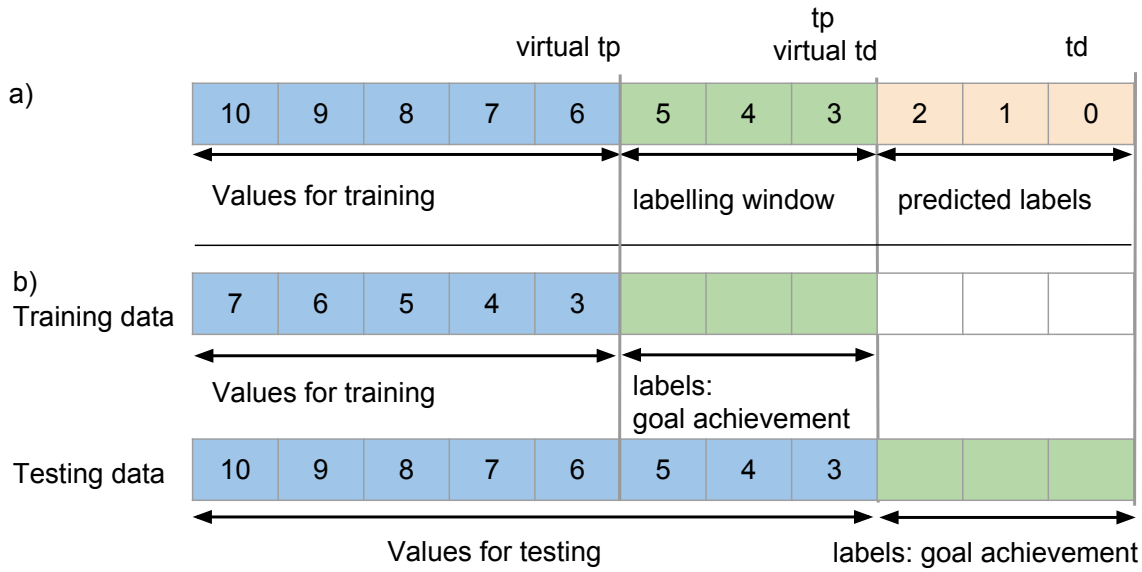

Figure 5.3: Classification framework for Self-Learning and testing predictions of at-risk students. The available features denote from which days the features can be used for training or testing data.

Based on the described concept of the virtual deadline and labelling window of the same size as the time remaining to the deadline, going back in history means the window for labels is growing. The more days before the deadline, the more days for training labels are needed. The situation for the prediction day being 1 to 4 days before the deadline is depicted in Figure 5.4. For n days before the deadline, the size of the window both for training and testing labels is n (the green area for each day).

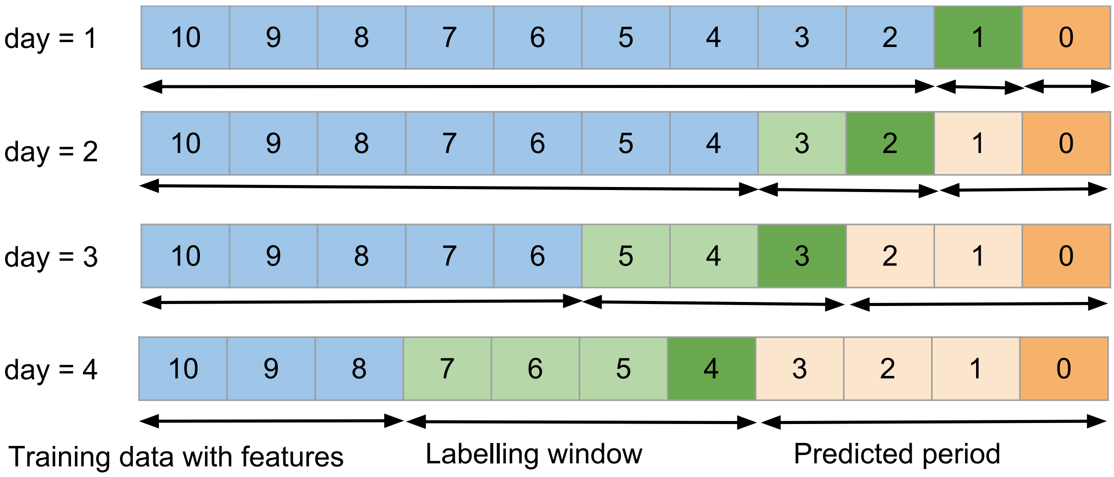

Figure 5.4: Extending window for training and testing labelled data. $Day = n$ denotes that the day of predictions is n days from the deadline, (i.e. $day = 1$ the deadline is due tomorrow, $day = 2$ in two days, etc.)

5.2.2 Evaluation Strategy

The common way of comparing the classifiers is K-fold cross-validation, by dividing the data into k folds, and repeatedly using the data in each of the fold as a testing data and the remaining data for training. In the case of imbalanced data, it is essential to use the stratified version of cross-validation, ensuring that the class ratio in all the folds is the same or almost the same.

In the case of Self-Learning, a different strategy was selected to evaluate the models. In each point of time, some objects have not yet achieved the goal. It is not certain whether they will reach the target in the end or not. Therefore, instead of creating the folds from these data, they are always used as testing data with the labels that are unknown in the moment of the predictions but available after the deadline passes. At the same time, these objects are used in the training data to represent the cases that will not achieve the goal. So, at the point of making the predictive models, all the objects in the testing set are known, however neither their labels nor their behaviour after the prediction date. Thus, there is no leakage of the testing data into the training data.

Given the prediction problem $GP^R = (D, g, t_d, t_0, t_p^R)$, the training and testing set can be defined using the set of unachieved objects DU from Equation 5.4 as follows:

$$\begin{aligned} D_{test}(D, g, t_d, t_p^R) &= DU(D, g, t_d - t_p^R) \\ D_{train}(D, g, t_d, t_p^R) &= DU(D, g, t_d - 2t_p^R) \end{aligned} \tag{5.19}$$

The time mapping $t_d - t_p^R$ in the DU function for testing data only ensures that this function receives absolute time, as defined in Equatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiu-atiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuatiuat 5.4, and analogously for the training data, where $2t_p^R$ denotes the virtual deadline from the Equation 5.18. So the testing data consists of objects not achieving the goal before or at

the prediction time, plus the training data from the objects not yet achieved the goal before or at the virtual prediction time.

Example 5.7 Figure 5.5 illustrates this for the same running example 3 days before the deadline, i.e. $t_p^R = 3$. It shows the time line of all the students at all the times, highlighting the change when a student submits the g_1. In the beginning, there are seven students, none of them having submitted the assessment, thus highlighted with the grey background. For the sake of brevity, the students in the figure are denoted only by a number, e.g. x_1 as 1. Seven days before the deadline, x_1 submits the assessment, which is highlighted by the dark green background for the student at this time. After the submission, the student is in the set of achievers and highlighted by the light green colour.

The testing data for this situation, $D_{test}(D, g_1, 10, 3) = DU(D, g_1, 10 - 3 = 7)$, contain all the students that have not achieved the goal until three days before the deadline, i.e. $D_{test} = \{x_4, x_5, x_6, x_7\}$. Three of these students submit at the end and the other one does not. The students x_1, x_2, x_3 are not included in the predictions because we know their class already.

The training data, $D_{train}(D, g_1, 10, 3) = DU(D, g_1, 10 - 2 \cdot 3 = 4$, is constructed from the students that have not submitted before the start of the labelling window, i.e. in the virtual deadline $t\_virt_d^R = 6$ (see Section 5.2.1), and D_{test} and the students x_2 and x_3. The achievement state of these students on the prediction day determines the class used for model training. Student x_1 submitted before the start of the labelling window, so it is not included in the training data.

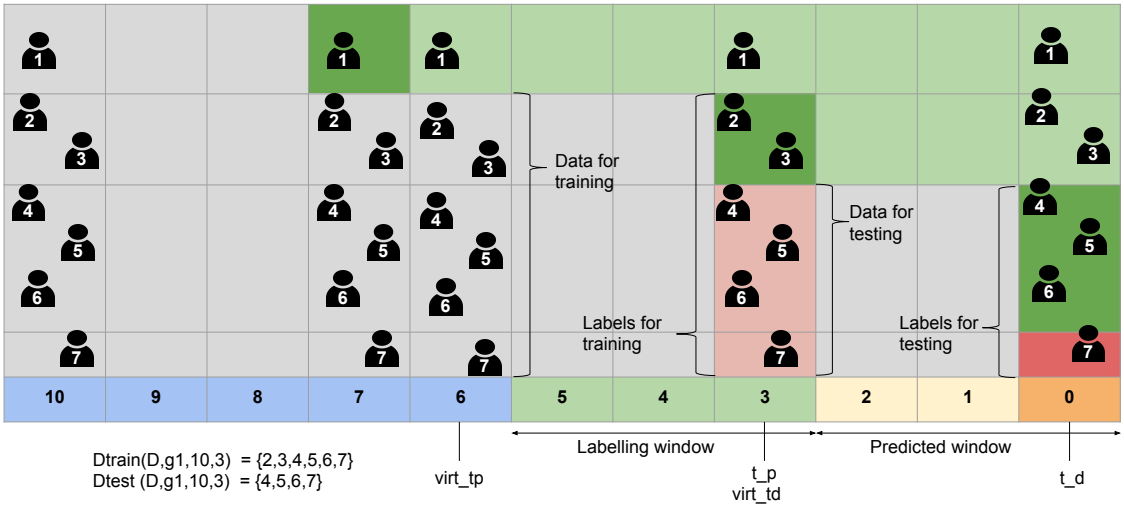

Figure 5.5: Training and testing data for predictions three days before the deadline, $t_p^R = 3$ for 7 students. Grey colour denotes that the underlying students have not yet submitted, the dark green in time t denotes the achievement, light green the presence in the DA set. Dark red colour denotes the objects pertaining to class $NotAchieved$ in the testing data, and light red the same class for the training data.

5.3 Tackling Imbalanced Data

There are two important questions to ask when dealing with imbalanced data: (1) Which evaluation measure should be selected for comparing the best solution? (2) How to improve

the performance of the classifier with respect to the imbalanced data? The specificity of this problem lies in the change of the information in the training data when getting closer to the deadline. The more we move backwards from the deadline, the higher is the imbalance ratio in the training data, due to fewer objects having already achieved the goal. Also, in the beginning, there is less information available about the objects, e.g. their behaviour in time. Moreover, depending on the domain, the majority class in the training data can become a minority class in the testing data. For example, it might be expected that most of the objects will, in the end, achieve the goal, e.g. most of the students will submit their assessment.

5.3.1 Evaluation Measure Selection

The most used measures are the area under the ROC curve (ROC AUC); the area under the Precision-Recall curve (PR AUC); F1-Score and G-Mean. Out of them ROC AUC and PR AUC are the most suitable ones as we want to capture various thresholds that the classifiers can use for predicting the data to the positive or negative class, in other words Achieve or Not Achieve the goal in our case. Such a prediction threshold might not be specified directly, so these measures provide the overall view across all of them.

Most published work focuses on using the ROC curve. The final decision is usually dependent on the domain and the problem being solved, as both of the measures have their specifics.

Both PR AUC and ROC AUC include Recall, i.e. proportion of target class that was identified by a classifier. However, there is a distinction in treating different types of errors made by the classifiers. Let's denote the class of our interest as 1 and the less important class as 0. ROC AUC combines Recall with the False Positive Rate, usually eliminating the importance of the minority class in the class imbalance. On the other hand, for PR AUC, Precision does not take into account the number of correctly identified zero-class objects and specifies the ratio of correctly identified students to all the students that were flagged as at risk. In the Self-Learning scenario, when an object achieves the goal in the time t, it is omitted from the testing data from the following time $t+1$ until the deadline. Consequently, the number of objects with already achieved goals in the testing data is changing, while the number of the unachieved class remains constant across the time. If the object will not achieve the goal, it will be available and marked as $NotAchieve$ in the testing dataset in all the days. On the other hand, the objects in the $NotAchieve$ class in the training data is changing, as the objects that achieve the goal will be moved to the *Achieve* class.

The PR AUC provides more information about the algorithm performance on the target (positive) class, especially when the imbalance ratio is large and the target class is more important [22]. Based on the domain of the problem, it might be important to select one of the measures or provide results for both of them.

5.3.2 Tackling the Imbalance

In the first phase, the goal was to show that it is possible to use the Self-Learning framework for training the predictive models before the deadline, even when given few objects that have already achieved the goal. Given the imbalanced nature of the data, especially in the early phases, existing approaches from the state-of-the-art should be investigated to see how they contribute to the performance measures.

A similar concept to the Self-Learning framework being used in the semi-supervised learning is *Self-Training* [112]. This technique utilises both the set of labelled and unlabeled

data to improve the performance of the classification. First, the model is trained solely on the labelled examples, and the unlabelled ones are then iteratively added until the performance of the classifier stops improving. Such a method has also been used in the imbalanced data. In [91], Stanescu and Caragea used the original method and several modifications tailored to the imbalanced data, achieving the best results when the training set was extended only with the examples predicted as a minority class.

The difference between the proposed Self-Learning framework and semi-supervised method in [91] is, there are no annotated objects of the negative class $NotAchieve$. In contrast, I use the temporal character of the data to construct them from the pool of available objects. Therefore, there is more information available about the object when approaching the deadline and the existence of the deadline itself.

5.4 Summary

This chapter introduced a process in which objects are supposed to achieve a goal within the specified deadline. The problem of predicting of the achievement of the goal by the objects that have not achieved the goal in the prediction was then defined. I presented a framework that can construct a predictive model based on the objects that have already achieved the goal in advance. Such a problem and approach by nature generate imbalanced data, because of only a few of the objects achieving the goal in the beginning. For this reason, it is important to properly select the evaluation metric and eliminate the influence of the imbalance in the training of the model. In addition to the above, the approach of testing the method in the prediction date was introduced. An example of such a problem from the domain of education was selected and examined as a case-study to evaluate the framework.

Chapter 6

Case Study: Predicting at-risk students without the legacy data

"Work expands so as to fill the time available for its completion."

C. Northcote Parkinson

The data that have been selected to evaluate the method come from the educational domain, from a distance based higher educational institution. They contain information about student demographics, assessments and their behaviour in the Virtual Learning Environment (VLE) in several courses. The Self-Learning approach has been applied to identify students at risk of failing the course by focusing on those that are unlikely to submit the first assessment. This means that only data from the already running course have been used, i.e. without any legacy data from the previous presentation of the course[1], which is a common practice for training the models.

In recent years, Learning Analytics (LA) and Educational Data Mining (EDM) have rapidly grown in interest. This growth is visible especially in distance based education, which is nowadays often available for free by various online platforms (edX, Coursera, Udacity[2]). Both LA and EDM analyse the data about student learning, EDM being focused on the algorithmic part and LA on the connection with learning theories. One of the key issues, which the recent research investigates, is the identification of students at risk of failing the courses. According to [84, 98] the number of students not completing university degrees in Europe is between 20 and 50%. For distance education, these figures are even more pessimistic with 78% of students not finishing their degree [89].

The highest level of dropout typically occurs in the first years' courses, and many students drop out even during the first few weeks of the course presentation. This finding has been confirmed by the analysis of both distance Higher Education (HE) courses [105] and Massive Open Online Courses (MOOCs)[3] [94]. Therefore, the goal of the analysis is usually to identify at-risk students as early as possible. It is worth noting that the same pattern might not necessarily hold for all educational institutions - depending on the course design, significant student drop-out may sometimes occur later in the course [56].

[1] the term presentation and a run of a course will be used interchangeably with the same meaning

[2] edX – http://www.edx.org
Coursera – http://www.coursera.org
Udacity – http://www.udacity.com

[3] MOOCs – Massive Open Online Courses – freely available courses on the Internet focused on providing distance education to as many students as possible.

The results of the state-of-the-art solutions are highly determined by the data available for analysis, which is dependent on the type of educational institution (i.e. whether it is a high school, university, distance learning university or MOOC). Nevertheless, the main idea is usually the same, i.e. to use legacy data to train predictive models. Then, the models are used to provide predictions for the current students - current presentation of the course, current cohort of high-school, etc.

6.1 Data Collection

The data, which have been used to evaluate the methods proposed in the thesis, come from the combined sources of The Open University (OU), one of the largest distance learning institutions in Europe. The available records contain both demographic and behavioural data from the Virtual Learning Environment (VLE).

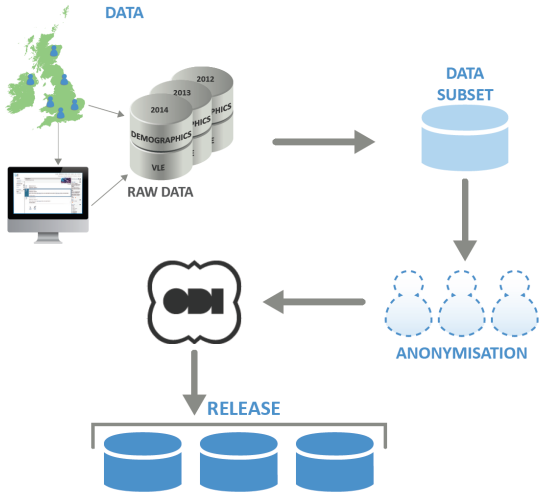

Figure 6.1: Anonymisation process of the OULAD dataset

In order to make this research possible, and also to support the research communities in LA and EDM, we have published the anonymised version of selected students data in The Open University Learning Analytics Dataset (OULAD) [61]. First, based on consultation with the educational staff, we selected seven courses across various fields and then anonymised them. In the anonymisation process, we followed the recommended principles of k-anonymity. This means that no student can be distinguished from at least $k-1$ other students in the dataset. Such students were either removed from the dataset, or their attributes were grouped to a higher level of a defined hierarchy. For example, having only one student from a small village would result in grouping the student into a regional or even a country level. The dataset has been certified by the Open Data Institute[4] and it is available for download at the OUAnalyse website[5] and from UCI Machine Learning Repository[6]. This process is depicted on Figure 6.1. The result is a unique dataset containing both demographic and detailed activity data in the VLE. Closest to our dataset is the MIT

[4]https://certificates.theodi.org/en/datasets/26648/certificate

[5]OULAD website: https://analyse.kmi.open.ac.uk/open_dataset

[6]OULAD at UCI Machine Learning repository: https://archive.ics.uci.edu/ml/datasets/Open+University+Learning+Analytics+dataset

and Harvard MOOC dataset released in 2014 [47]. However, in contrast to our dataset, the VLE activities here are only summarised across the whole course and grouped by the activity type.

The dataset contains 7 courses denoted as AAA to GGG with 4 presentations of the courses in years 2013 and 2014. A presentation starting in February is denoted as B and a presentation in October as J. This means that the dataset contains the presentations 2013B, 2013J, 2014B and 2014J. Not all of the courses have all the presentations available. The courses cover a broad range of fields such as science, technology, engineering, maths (STEM) and social sciences. AAA is a level three course, GGG is a preparatory course, and the rest are level one courses. The summary of information about the course with the number of students in each of them across all the presentations is described in Table 6.1. The entire schema of the dataset and its detailed description is amended in the attachment B.1.

Table 6.1: **Course summary and domain information.**

Course	Domain	Presentations	Students
AAA	Social Sciences	2	748
BBB	Social Sciences	4	7909
CCC	STEM	2	4434
DDD	STEM	4	6272
EEE	STEM	3	2934
FFF	STEM	4	7762
GGG	Social Sciences	3	2534

6.1.1 OULAD Course Structure

The courses in the OULAD take between 20-30 weeks. During this period, students are assigned 4-6 assessments A1, A2, ... with fixed cut-off days so that the tutors can verify whether the student understands what has been presented. The course could be completed by the final exam or a project. The overall success or failure of the student depends on the results of the assessments and the final exam. The simplified model of a course is shown in 6.2.

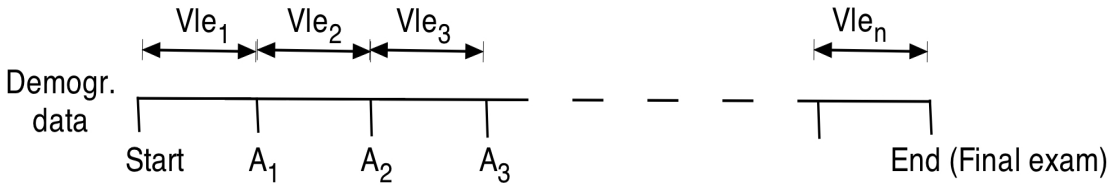

Figure 6.2: Course structure at the Open University in the OULAD dataset

The courses in the dataset come from distance education, and they are delivered through a VLE. The study plan of a course is organised as a sequence of blocks that are terminated with assessments. The students' interactions with the VLE system and their assessment submissions are recorded and stored. This is demonstrated in the Figure 6.2. $A1, A2 \ldots, A_n$

indicate the assessments, $VLE1, VLE2, \ldots$ denote the VLE blocks in the periods between either the start of the course and the first assessment or else in between assessments. Moreover, the data include demographic information (age, gender, region of residence, etc.), which are available from the start of the course.

6.2 Early Identification of At-Risk Students

Most of the recent work in Higher Education makes use of demographic, performance and behavioural data extracted from the VLE. The key issue addressed by different educational institutions and researchers is how the concept of an at-risk student is defined. It can be a student with a grade lower than C [50] or even B- [6], less than 60% [7], not submitting the following assessment [105], or dropping out in the days following enrollment [56, 94].

When all data are available, the best predictor makes use of actual performance either by (a) student study history (e.g. measured by GPA) or (b) by evaluating the progress in the current course from assessment results. However, student study history is not available for entry-level courses, i.e. courses that are taken at the beginning of the study programme. These courses require increased attention because student dropout is typically high [105].

The progress in the current course is unavailable before the first assessment is marked, though A1 is often important for early predictions. Many students drop out even at the beginning of the course. This issue has been addressed in [105] by predicting submissions of the A1 from demographic and pre-A1 VLE activities with models trained on the previous presentation of the same course. To early identify at-risk students, [52] used data on behaviour from the first week, by evaluating quiz results as the most important attribute. Similarly, [109] found quizzes to be most informative for predictions.

6.2.1 Importance of the First Assessment

The importance and suitability for predictions of the first assessment have been investigated in [105, 46]. For OULAD, the courses across all their runs are depicted in Table 6.2 in 4 columns: (a) probability of failing the course given student failing A1 (scoring less than 50%), (b) probability of failing in the course given that the student did not submit A1, (c) number of students who failed the course and submitted but failed A1, and (d) number of students failing the course and not submitting A1. The numbers in the table are means calculated across all presentations in the OULAD dataset.

The probability of failing the course if a student failed or did not submit A1 is almost 90%, making A1 a strong predictor of future failure. If no data from the previous presentation of the course is available, it is impossible to use the results from the assessment before they are marked. However, even the assessment submission is a good predictor for identifying at-risk students. On average, there is a 95% probability that a student will not finish the course if he/she has not submitted A1. When limiting the approach only to predict submissions rather than predicting failures, it is not possible to identify some of the at-risk students. According to the 3rd and 4th column in Table 6.2, the predictions are missing 1392 students (i.e. 15% of those that fail), but at the same time, the probability of failing the course is 5% higher, making the predictions more accurate.

Table 6.2: Probability of failing the course (F) if failed (F_{A1}) or not submitted (NS_{A1}) A1 for courses averaged for all available presentations and count of students.(denoted as #)

Course	$P(F\|F_{A1} \vee NS_{A1})$	$P(F\|NS_{A1})$	$\#(F \wedge F_{A1} \wedge S)$	$\#(F \wedge NS_{A1})$
AAA	0.8004	0.9421	23	50
BBB	0.8809	0.9905	434	1888
CCC	0.8381	0.8252	255	1710
DDD	0.9651	0.9876	431	1436
EEE	0.9808	0.9932	49	643
FFF	0.9805	0.9930	122	1527
GGG	0.8300	0.9157	78	453
AVG / SUM	**0.8966**	**0.9496**	**1392**	**7707**

6.3 Predicting Assessment Submission

In the educational context, the submission of an assessment can be viewed as a goal achievement problem with a deadline.

Given the day of the prediction, which is n days before the deadline, the goal is to construct a binary classifier to predict if the student (1) will submit or (2) will not submit the next assessment in time. If $n = 1$, all the data except the ones from the deadline day are available for the predictions.

Formally, the Assessment Prediction Problem (APP) is defined as a Goal Achievement Prediction Problem with relative counting, see Equation 5.13. According to this equation, D is a set of all students enrolled in the course, the goal is defined as the submission of the assessment, which has a specified due date, and t_p is the prediction time, in this context the prediction day. I will refer to the set of students who submitted until time t as $Submit(t)$ and those who have not as $NotSubmit(t)$. Further, $Submit$ and $NotSubmit$ without specification of the time refers to $t = t_d$, representing the classes that the model is trying to predict. Because the main target in Learning Analytics is identification of at-risk students, the class of interest for the evaluation measures is $NotSubmit$. Then, for example, if the model has a high recall, it means that it can identify the students that will not submit the assessment. In case of the highest possible precision, all the students that the model predicts as not submitting belong to the class $NotSubmit$.

The remainder of this thesis will focus on predicting submission of $A1$, the importance of which was explained in 6.2.1. Prediction of the following assessments $A2, A3, \ldots$ can be defined in a similar fashion.

6.4 Features for Learning

The available data include information about student demographics and activities in the VLE. The VLE data contain student activities summarised by day. For example, student John viewed ten times the specific PDF *study_material.pdf* in day 8. All of the activities are grouped into *activity types*, therefore all of the PDF materials are grouped as a resource. There are twenty different activity types such as *forum, video, resource, glossary, wiki*, etc.

Following the backwards problem definition in Equation 5.13, all the clicks have been aligned backwards with respect to the deadline. In addition to the VLE daily counts, various statistical information that summarises student behaviour has been extracted. For instance, the number of days that the student was active in the VLE (when he/she logged in). These statistics and all the other features are described in Table 6.3.

Table 6.3: Static and Dynamic features used for training the predictive model.

No.	Type	Dim.	Description	Examples
1	Demographic (static)	8	Socio-demographic data	Age, IMD[7], Qualification, Region, Gender, Declared Disability, No. of previous attempts, No. of currently studied credits
2	Registration info (static)	1	The registration day relative to start of the course.	-
3	VLE statistics (dynamic)	28	Various statistical measures about student behaviour in VLE	1) No. of consecutive days that the student is currently active, 2) first and last day he/she was active or indication of never having logged in, 3) average/median no. of clicks and no. of materials visited per day normalised either by all days or only by active days in the VLE, 4) total no. of active days in VLE
4	VLE statistics before presentation start (static)	19	Same as 3), measured before only the start of the presentation	Same type of statistics as for previous feature type (3) but only limited to 3) and 4) features mentioned in the examples.
6	VLE daily counts per activity type (dynamic)	50-560	No. of clicks in the VLE by activity type per day	No. of clicks in resources/forum in day 0, 1, ...

6.5 Imbalanced Data

During a course, only some students submit the assessment at the beginning, and the number will grow as the deadline approaches. Part (a) of Figure 6.3 shows the daily submission counts for the courses in the OULAD dataset and confirms what was expected. Very few students submit the assessments early in the course, and the number grows exponentially when the deadline approaches. Consequently, the data are highly imbalanced in the begin-

[7]Index of multiple deprivation – qualitative indicator of United Kingdom deprived areas

ning with the ratio decreasing closer to the deadline. This is shown in the right section of Figure 6.3 in (b).

The figure indicates that not only the ratio of submitted students in the training data grows, but the situation is opposite to that in the testing data, where the majority is the Submit class, and gets lower as the deadline approaches. This is because in this case most of the students in the courses will submit the assessment at the end and they remain in the testing data until they submit. After submission of the assessments, the students are removed from the testing data and thus lowering the imbalance.

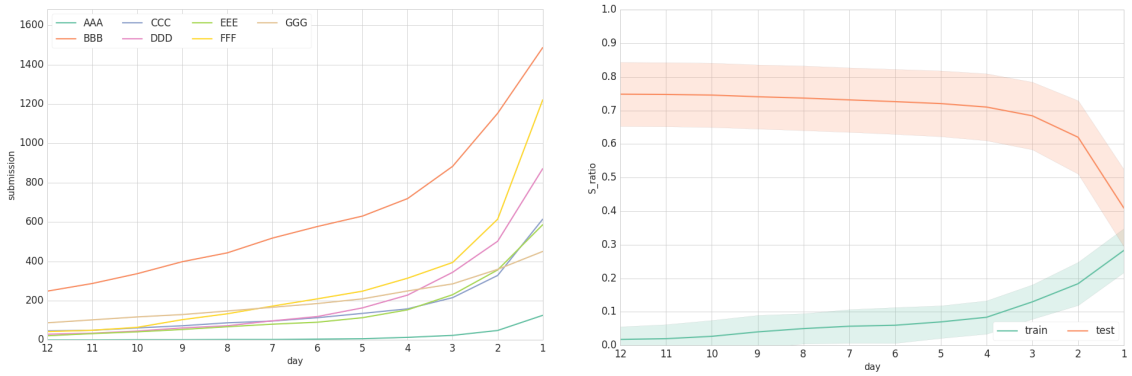

Figure 6.3: (a) Students submissions number per course counted as a number of days to the deadline and (b) ratio of Submitted students in the training and testing data.

6.5.1 Tackling Imbalanced Data

To construct the predictive model, I use several machine learning algorithms, all of them scoring classifiers (see Section 2.2. This enables ordering of the predicted data according to their confidence. The selected algorithms include[8]:

- Logistic Regression (LR),
- Support Vector Machines (SVM),
- Random Forest (RF) [13],
- Naive Bayes (NB) and
- eXtreme Gradient Boosting (XGB) [20]

For the algorithms such as SVM and Logistic Regression, it is possible to use cost-sensitive learning by specifying the weights of the classes during the training. Moreover, RF and XGB can be considered as ensemble-based algorithms, which can cope with the imbalance data by bagging or boosting.

I utilised several sampling methods for modifying the class distribution. I selected the algorithms both from random and informed methods and both for under-sampling and over-sampling. All of them were described in the data prepossessing methods in Sec 3.1.3. The goal was to investigate if sampling improves the quality of the predictions and which kind of sampling works best.

[8]For training the models, and for the whole evaluation framework, the implementation of the algorithms from Python Scikit-learn library [77] was used

- Under-sampling
  - Random under-sampling (Rand-Under)
  - Tomek-Links
  - Extended Nearest Neighbours (ENN)
  - Neighbour Cleaning Rule (NCR)
- Over-sampling
  - Random over-sampling (Rand-Over)
  - SMOTE [18]
- Both under and over sampling
  - SMOTE-ENN – SMOTE over-sampling followed by elimination of the Extended nearest neighbours [9]
  - SMOTE-Tomek – SMOTE over-sampling with subsequent removal of Tomek-Links [8]

6.6 Evaluation

The experiments were focused on investigating the predictive power of the machine learning models under the Self-Learning framework, and comparing them with two defined baseline models. Furthermore, the focus was to investigate which strategy for tackling the imbalanced data led to better results. The goal was to examine the difference between the Self-Learning approach and a typical way of training the models, using the legacy data from the previous presentation.

6.6.1 Experimental Setup

The experiments were conducted on the selected courses from the OULAD dataset, described in more depth in 6.1.1. I excluded from the analysis the course AAA because it is a level-3 course and these courses are usually not analysed with regards to at-risk students, but are instead analysed in order to improve the performance of remaining third year students. To compare the Self-Learning approach with training on the previous presentation, I selected only those courses from the 2014J and 2014B presentations, for which 2013J or 2014J presentation exists. For this reason, the course CCC is missing in the experiments.

The courses have between 750 and 2500 students with the pass-rate ranging from 37 to 60%. The goal was to predict the submission of the first assessment by students registered in the course within the deadline. The submission rate of the first assessment for the selected courses was between 73 to 79 percents. The performance was measured using ROC AUC and PR AUC from 1 to 19 days before the deadline. The assessment deadline ranged from day 19 to 33 for the level-1 courses and for the preparatory course GGG it was exceptionally later, on day 61. This information is available for each course for the 2014 presentations in Table 6.4 and in Table 6.5 for 2013 presentations.

Because there are more courses available in the dataset, for each day before the deadline and each model, the mean value of the performance measure across all the courses was computed and used for comparison, following the definition of the performance measure in Equation 5.17.

Table 6.4: Information about the courses under analysis - 2014 presentation

Course	Pres	No. of students	Pass Rate [%]	A1 S/NS [%]	Deadline [Day]
BBB	2014B	1613	54.93	73.65	12
BBB	2014J	2292	49.74	77.31	19
DDD	2014B	1228	60.99	75.65	25
DDD	2014J	1803	56.07	78.48	20
EEE	2014J	1188	42.42	78.20	33
FFF	2014B	1500	56.40	79.40	24
FFF	2014J	2365	52.77	77.12	24
GGG	2014J	749	40.72	77.97	61

6.6.2 Difference in Setup with Published Results

The experimental setup here slightly differs from the published results in the paper in [46]. In the article, only one presentation (the most recent one) was used, while here the focus was extended to both of 2014 presentations. Further, I decided to include a previously discarded preparatory course GGG, since there is interest at The Open University to widen analysis of at-risk students at an early level.

The ROC AUC was added as a supplemental metric for the evaluation, as it shows a different view on the performance, counting also the correctly identified submitted students. Most importantly, the values for the PR AUC slightly differs from the article. I discovered that the area under PR curve in the used Sci-Kit library [77] was computed by linear interpolation. In this case, it might give overly optimistic results for poorly performing models, particularly the baseline models.[9]

Table 6.5: Information about the courses under analysis - 2013 presentation

Course	Pres	No. of students	Pass Rate [%]	A1 S/NS [%]	Deadline [Day]
BBB	2013B	1767	54.56	76.23	19
BBB	2013J	2237	52.08	75.77	19
DDD	2013B	1303	60.86	75.59	25
DDD	2013J	1938	57.22	76.83	25
EEE	2013J	1052	42.11	78.42	33
FFF	2013B	1614	51.55	83.95	19
FFF	2013J	2283	52.04	81.30	19
GGG	2013J	952	37.82	82.25	61

6.6.3 Baseline Models

To show the influence of using the machine learning modelling, I defined two baseline models that have been selected with respect to the Learning Analytics domain.

1. **Base[NotSubmit]** or **B[NS]** – this assigns all the students to NotSubmit class, meaning that the possible interventions would be done with all the students. The

[9] This issue has been fixed in the new release in August 2017 in https://github.com/scikit-learn/scikit-learn/issues/5379.

model will have maximum $Recall = 1$, but it is expected to have low Precision and $Specificity = 0$. Then FPR will always be 1. The results will reflect the class imbalance in the data in case of PR AUC. $AUCROC$ will always be 0.5 because $[1, 1]$ will be the only point defining the curve, making the size of the area 0.5.

2. **Base[NotAccessed]** or **B[NA]** – it reflects the simple belief that students that have not logged into the system since its opening are not showing effort to submit the assessment. The model classifies all these students as NotSubmit and the others as Submit. The model should be able to capture the most critical students, but it is not expected to identify all of them.

6.6.4 Self-Learning Results

Here the performance of the algorithms for various days relative to the deadline was compared with each other and with the baseline models. Notice, that time is measured backwards, with the deadline t_d being the right most position of the figures. LR stands for Logistic Regression, with W indicating weighted (or cost-sensitive) LR, used for imbalanced data, the same for SVM. SVM and SVM-W = SVM with RBF kernel, RF = Random Forest, XGB=XGBoost, NB=NaiveBayes, B[NS] and B[NA] are the baseline models.

PR AUC of the classification methods is depicted in Figure 6.4 and Table 6.6. Not surprisingly, the performance reaches its best value on the last day of the prediction, i.e. one day before the deadline. Then, for PR AUC the performance drops down when going back in time, especially from day 1 and 2. This drop is steeper especially for the models that do not cope with the class imbalance, such as SVM and LR without class weighting. The performance of their weighted version is not decreasing as much when going back in time. However, the best results were achieved by the Random Forest in all days.

The Base[NotSubmit] baseline model was outperformed by all the other models from day 9 until the deadline. Before day 9, the results of the baseline were similar to NB, both being better than LR. The other baseline model, Base[NotAccessed], achieved much better results and especially at the start it can be considered as a good proxy because from day 19 to 15 it was outperformed only by Random Forest, and until day 9 only by weighted versions of LR and SVM.

Figure 6.5 depicts the ROC AUC and for days 1-9, Random Forest and weighed SVM perform similarly, starting from day 10, Random Forest starts outperforming the SVM-W. These two models are followed by weighted LR and XGB. In comparison to PR AUC, there is no steep drop in performance from day 1 to 2 for these four models. This is due to the metric also counting with correctly identified submissions, which are easier for the classifier to identify earlier. In other words, it is more difficult to reach better Precision than better Specificity when having less information about submitted students.

Imbalanced Data and Sampling Methods

Both Figure 6.4 and Figure 6.5 reveal how important it is to set the class weights to handle the imbalanced data. The difference in performance between weighted and unweighted versions of LR and SVM becomes visible when moving further from the deadline and having a higher imbalance ratio both for PR AUC and ROC AUC. The performance of the weighted version is not affected by the change that much because the error made on the minority class influences the model. In case of a high imbalance ratio, the model might not underpin the minority class and classify all the data to the majority class.

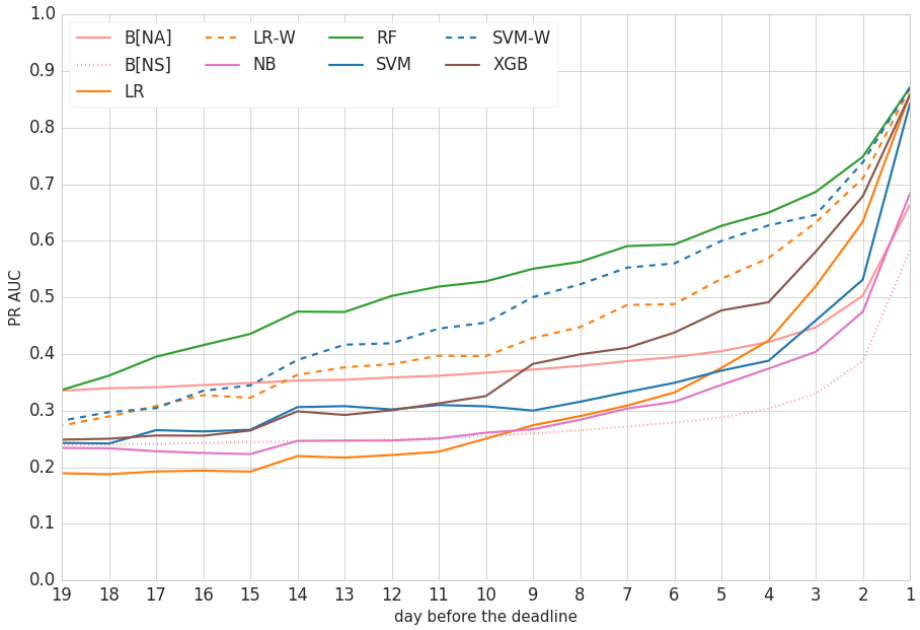

Figure 6.4: PR AUC for 1 to 19 days before the deadline for different machine learning models using Self-Learning.

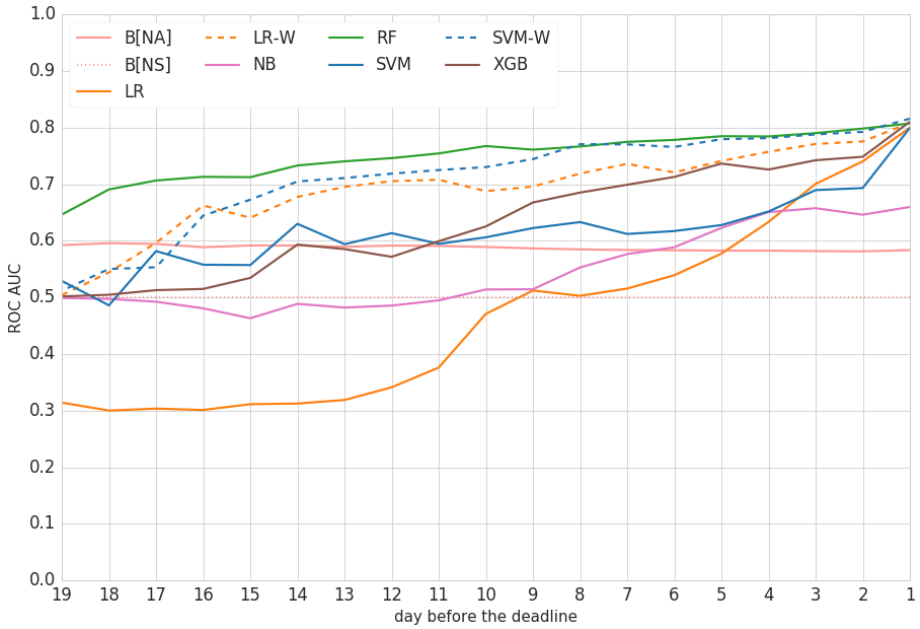

Figure 6.5: ROC AUC for 1 to 19 days before the deadline for different machine learning models using Self-Learning.

Being the best classifier, Random Forest was chosen and used with selected sampling methods to balance the data before training the model. Table 6.7 shows that the differences in PR AUC performances are not as significant as differences between the models. Approaching to the deadline, differences are negligible, and they start to be visible from day 8 on backwards to the past. The only conclusions that we can draw are (1) random under-

Table 6.6: PR AUC values for days 1 to 19 trained on the same presentation using Self-Learning.

Day	B[NA]	B[NS]	LR	LR-W	NB	RF	SVM	SVM-W	XGB
1	0.6322	0.5363	0.8658	0.8810	0.6662	0.8815	0.8526	**0.8832**	0.8626
2	0.4652	0.3587	0.6124	0.7217	0.4869	**0.7653**	0.5436	0.7483	0.7185
3	0.4165	0.3139	0.5259	0.6465	0.4133	**0.6846**	0.4656	0.6643	0.6073
4	0.3980	0.2872	0.3990	0.5888	0.3914	**0.6612**	0.4114	0.6395	0.5078
5	0.3803	0.2732	0.3120	0.5576	0.3474	**0.6337**	0.3964	0.5947	0.4827
6	0.3738	0.2652	0.2905	0.4849	0.3186	**0.5880**	0.3728	0.5701	0.4211
7	0.3695	0.2599	0.2569	0.5127	0.3185	**0.5821**	0.3437	0.5667	0.3945
8	0.3633	0.2547	0.2498	0.4265	0.2881	**0.5801**	0.3285	0.5162	0.4062
9	0.3623	0.2517	0.2558	0.4479	0.2689	**0.5618**	0.3007	0.4930	0.3708
10	0.3642	0.2486	0.2218	0.4282	0.2530	**0.5483**	0.3283	0.4612	0.3172
11	0.3546	0.2462	0.1908	0.4060	0.2527	**0.5316**	0.2969	0.4438	0.3185
12	0.3517	0.2454	0.1791	0.3911	0.2432	**0.5235**	0.3142	0.4382	0.2966
13	0.3513	0.2448	0.1807	0.4060	0.2398	**0.4798**	0.2776	0.4391	0.2994
14	0.3502	0.2441	0.1803	0.3523	0.2428	**0.5256**	0.3125	0.3909	0.2864
15	0.3431	0.2433	0.1805	0.3275	0.2200	**0.4840**	0.2756	0.3434	0.2665
16	0.3395	0.2427	0.1771	0.3323	0.2238	**0.4665**	0.2504	0.3181	0.2512
17	0.3340	0.2420	0.1763	0.2867	0.2263	**0.4177**	0.2830	0.2576	0.2462
18	0.3344	0.2416	0.1750	0.2465	0.2269	**0.3912**	0.2243	0.2579	0.2496
19	0.3294	0.2412	0.1777	0.2166	0.2267	**0.3311**	0.2274	0.2353	0.2435

sampling performs better than random over-sampling and (2) only *random under-sampling* and SMOTE-ENN in several days perform better than the solution without sampling. Both of them being the best performing solution in seven different and overlapping days. The better performance of under-sampling to over-sampling can be explained by the domain and data available. There are students in the created class for non-submission in the training data, who are similar to the submitted students, but they submit little bit later making them being labelled as $NotSubmit$. This can introduce error in the training. The informed sampling methods do not have this domain information, and we can argue that under-sampling based on randomness leads to better results than sampling based on the informed methods that are usually based on the distances between classes.

SMOTE-ENN, combining both under-sampling and over-sampling, achieves usually good results in the published results such as in the large comparison study in [66, 9]. The good results of this algorithm in the performed experiments were surprising because the algorithm performs over-sampling by SMOTE first, and only then does it clean the data by ENN. The elimination by ENN might explain the good results despite the fact that the results of ENN itself were not as good as expected. Expectations were high because the method should remove the borderline data from the majority class. These should be the ones that did not make it to the submit group while being similar to them.

To better see the performance of the algorithms in time, random under-sampling and SMOTE-ENN were chosen and compared with the unsampled solution by their PR AUC. The Figure 6.6 shows that the performance is nearly equal between days 12 and 1. For days 14 to 12, random under-sampling being better; in days 19 to 14 SMOTE-ENN performs best, with random under-sampling performing worse than the original solution.

However, if we take ROC AUC as a performance measure, it turns out that the differences are even lower. The Figure 6.7 reveals that ROC AUC for the original solution and

Table 6.7: PR AUC values for days 1 to 19 trained on the same presentation using Self-Learning with different sampling methods.

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
1	0.8777	**0.8833**	0.8815	0.8797	0.8804	0.8734	0.8748	0.8746	0.8820
2	0.7617	0.7594	0.7621	0.7722	**0.7749**	0.7578	0.7692	0.7607	0.7609
3	0.6927	0.6922	0.6914	0.6848	0.6763	0.6839	0.6797	**0.6985**	0.6964
4	0.6496	0.6398	0.6520	0.6756	0.6720	0.6609	0.6608	**0.6831**	0.6569
5	0.6250	0.6385	0.6324	0.6374	**0.6461**	0.6316	0.6246	0.6408	0.6414
6	0.6022	0.5958	0.6093	0.6087	0.5984	0.5715	0.5664	**0.6127**	0.6077
7	0.5907	0.5895	0.5967	**0.6118**	0.5864	0.5589	0.5629	0.5834	0.5852
8	0.5751	0.5844	0.5841	0.6007	**0.6067**	0.5672	0.5744	0.5924	0.5833
9	0.5611	0.5632	0.5595	0.5691	**0.5960**	0.5506	0.5406	0.5614	0.5518
10	0.5377	0.5443	0.5435	**0.5547**	0.5458	0.5355	0.5150	0.5432	0.5356
11	0.5257	0.5244	0.5300	0.5387	**0.5535**	0.5235	0.5254	0.5482	0.5336
12	0.5366	0.5148	0.5323	0.5118	0.5339	0.5173	0.5204	**0.5518**	0.5500
13	0.4830	0.4641	0.4707	0.4685	**0.5308**	**0.5308**	0.5101	0.5191	0.5062
14	0.5055	**0.5219**	0.5143	0.4760	0.5143	0.5024	0.5033	0.5164	0.4867
15	0.4737	0.4806	0.4892	0.4585	0.4748	0.4935	0.5039	**0.5224**	0.4995
16	0.4475	0.4791	0.4490	0.3998	0.4115	0.4671	0.4722	**0.4959**	0.4447
17	0.4138	0.4236	0.4165	0.3999	**0.4594**	0.4209	0.4022	0.4234	0.4292
18	0.3907	0.3815	0.3775	0.3417	0.2731	**0.4142**	0.3994	0.3822	0.3849
19	0.3543	0.3596	0.3615	0.3117	0.3251	0.3436	0.3357	**0.3620**	0.3619

SMOTE-ENN is almost similar. Random under-sampling again achieves better results in days 14 to 12 and worse than the other two methods in days 19 to 17. Thus, SMOTE-ENN can be considered the best algorithm to use with Random Forest, but only for the PR AUC measure, due to its improvement of this measure in the early days of predictions. However, we cannot claim that sampling the data significantly improves performance of the classifiers.

6.6.5 Comparing with Learning from Legacy Data and Training Using Testing Data

The aim of this experiment was to compare the Self-Learning approach with training on the legacy data, and with training on the same data used for testing. The latter meaning that during the training phase, the algorithms have the same information that for testing. During production this occurrence of matching information is impossible. Regardless, the approach enables identifying to which extent the model can explain the variability in the student data, when given all the information.

For training on the same course as predicted, the data provided to the learning algorithms were the same as in the previous experiments. The only difference was that all the students were correctly labelled with respect to the submission within the deadline. For training on the previous presentation, the same type of data for the given number of days before the deadline were used, but only from the different presentation, and again with all the data correctly labelled in the training data. The testing data were identical to the previous experiment.

Figure 6.8 shows that for PR AUC, the best performing model is once again Random Forest followed by weighted SVM and LR. In contrast to the Self-Learning, using training

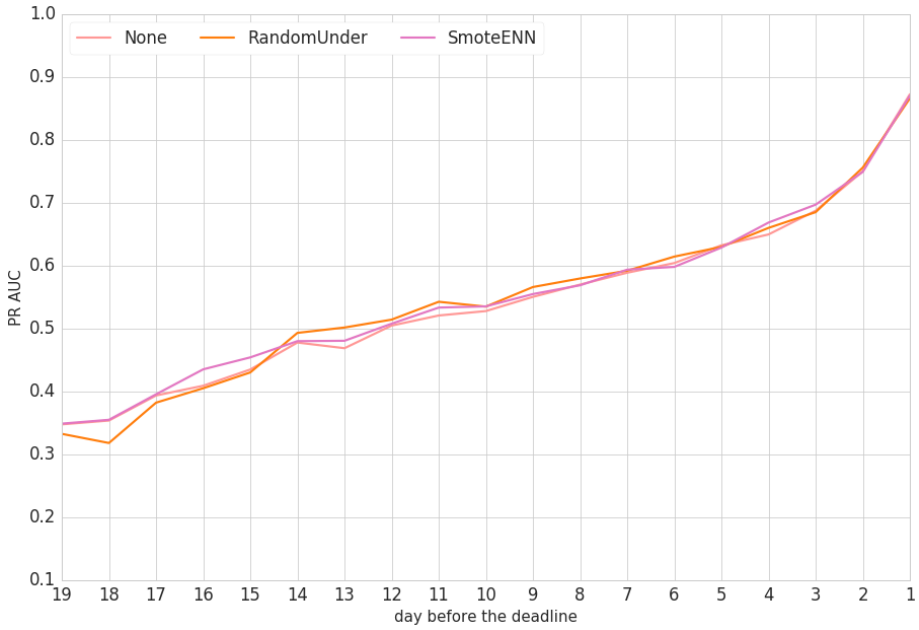

Figure 6.6: PR AUC for 1 to 19 days before the deadline using Self-Learning with Random Forest preceded by the sampling methods.

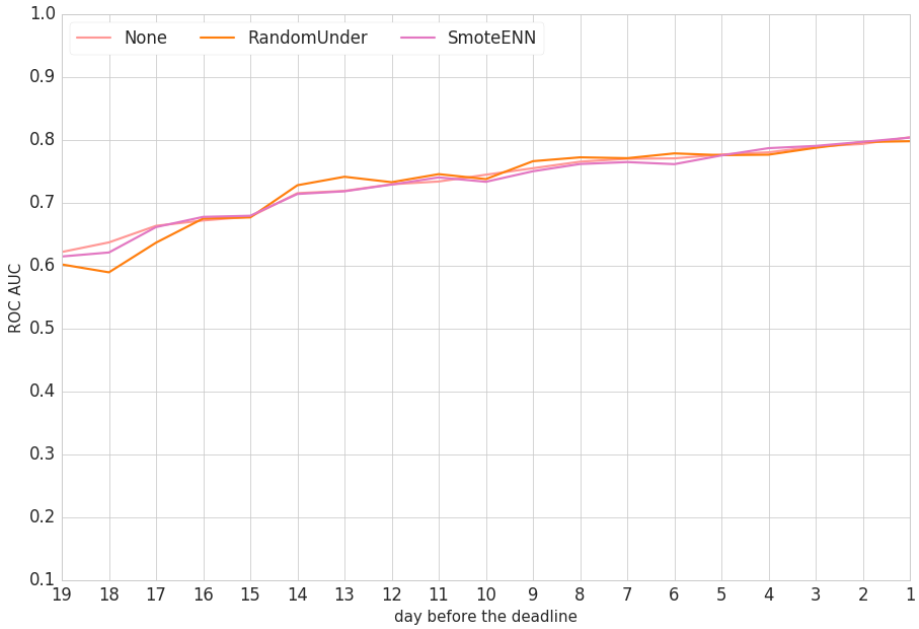

Figure 6.7: ROC AUC for 1 to 19 days before the deadline using Self-Learning with Random Forest preceded by the sampling methods.

on the previous presentation enabled more stable results from the algorithms. Only Naive Bayes performed poorly, even worse than the baseline model B[NA].

More stable results for PR AUC were achieved when using the same data for training and testing. As in the previous experiment, Random Forest performed best when followed

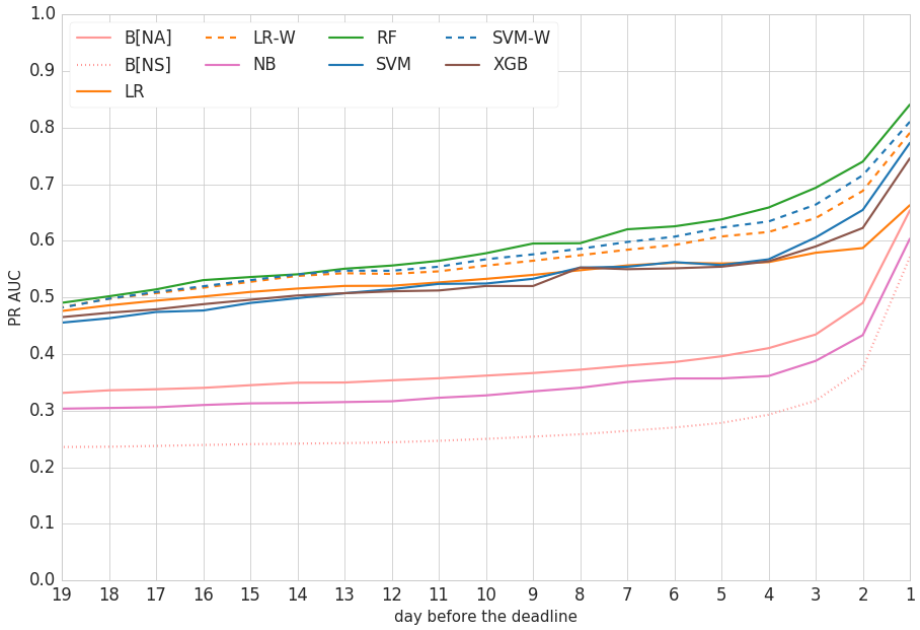

Figure 6.8: PR AUC for 1 to 19 days before the deadline for different machine learning models using training on the previous presentation.

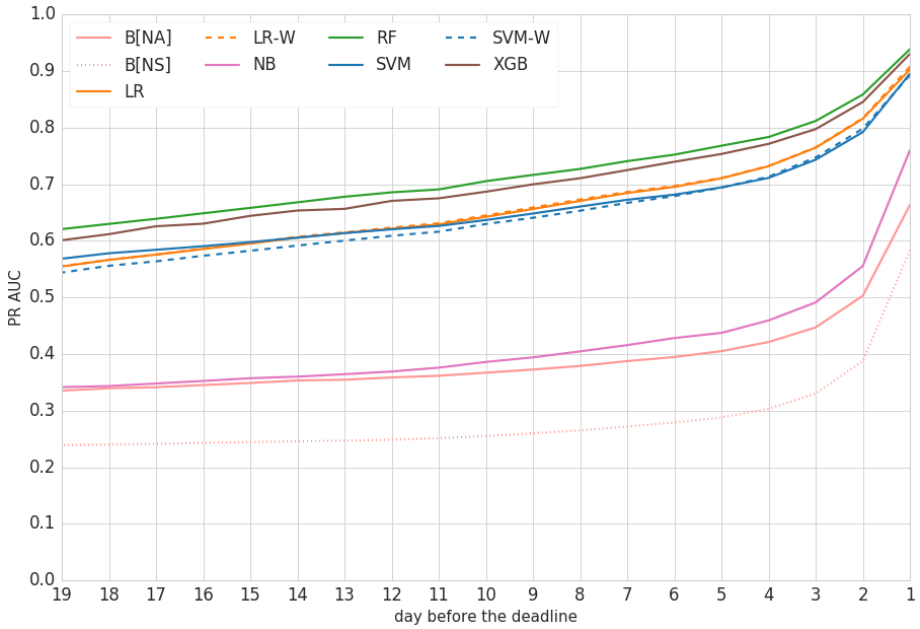

Figure 6.9: PR AUC for 1 to 19 days before the deadline for different machine learning models using training on same presentation as the testing data.

by XGB. The Naive Bayes model was again the only consistently poor performer for all days. The results are depicted in Figure 6.9.

Comparing the Best Solutions

For all three approaches (Self-Learning, Prev-Pres and Self-Test) the best performing algorithm was Random Forest. Therefore, Random Forest and the B[NA] baseline model was selected for comparing these three training methods. Now in Figure 6.10 the Prev-Pres denotes training on the previous presentation and Self-Test using training on the testing data. We can see that the best performance is achieved by the theoretical Self-Test approach. Together with the baseline model, it marks the upper and lower boundaries where we expect the performance of the machine learning models. No model is expected to have the performance measure either higher than Self-Test or lower than the baseline.

When comparing Prev-Pres with Self-Learning, one can notice that in general Prev-Pres achieves better performance, particularly at the beginning, when the Self-Learning makes use of very imbalanced data while lacking the information about the students. As the model approaches to the deadline, the difference Self-Learning and Prev-Pres diminishes, the former outperforming the latter on days 2 and 1. This can be explained by possible changes in course study plan in comparison to the previous presentation, which does not affect the Self-Learning approach. The results for ROC AUC suggest the same conclusions, differing slightly with Self-Learning performing better only in day 1.

Similar results were also achieved for ROC AUC depicted in Figure 6.11, which shows the measure together with PR AUC.

To summarise, Figure 6.10 demonstrates that: (1) Self-Learning has better predictive power than the baseline model and that (2) its performance approaches to Prev-Pres as it approaches to the deadline.

6.7 Related Work Dealing With Imbalanced Data in Education

The main difference in the EDM and LA domain is the usage of the data from the running course to train the predictive models. To the best of my knowledge, no existing work has utilised this information.

Some of the research for predicting at-risk students already tackled the problem of imbalanced data. In [2], motivated by the class imbalance caused by the minority of students who dropped out, they tried to perform over-sampling to improve the results of SVM. However, the results were negative, possibly influencing the method they used. Even though the specific method was not mentioned, if a random over-sampling was used, the result would not be surprising. Recall that in my experiments this sampling method was among the worst performers.

In [50] a data level approach was used, combining both random over-sampling and under-sampling. They evaluated the data from one semester trained by the two previous semesters. The only conclusion they drew was that using these approaches improves the Recall. However, at the same time, Precision dropped severely, which is understandable from the definition of these measures. Even though the authors did not include the F1-Score results explicitly, after computation, the best results for F1-Score was achieved by SVM without balancing. Similar as in [2], this might suggest that using only sampling without comparing informed methods, or without using domain knowledge might not necessarily lead to better results.

In [95], Nguyen et al. improved the ROC AUC and F1-Score of the predictive models by over-sampling the dataset using informed SMOTE algorithm. They examined, which

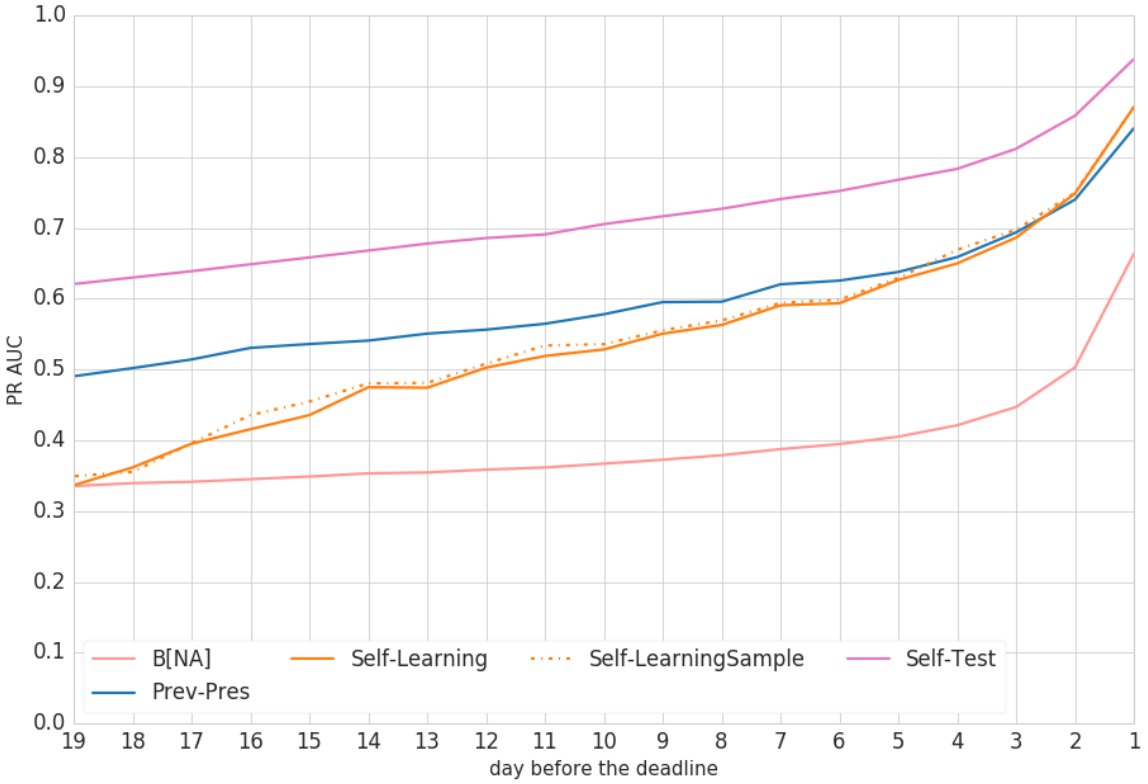

Figure 6.10: PR AUC for 1 to 19 days before the deadline for different machine learning models using training on the previous presentation.

algorithm best copes with different cost ratios specified to False Negatives and False Positive errors. In contrast, in [43] the problem of imbalance was simplified by focusing only on 'active' students and completely omitting those who have not shown interest in submitting the assessments. This strategy eliminated many of the majority class examples, which removed the imbalance between the classes, however, comparison against not using this solution was not provided.

The reason for the imbalance in the data of [50],[95], [43] and [2] is caused by the low number of at-risk students in the data. On the other hand, the source of the imbalance in the Self-Learning approach is caused by not having enough data on student submissions at the start.

There is much research in the educational field that investigates how to predict student performance and identify at-risk students. There is less research discussing the issues of imbalanced data, despite this being a possible cause of poor performance. Some of the published work uses imbalance prevention techniques to tackle the issues without comparing more methods and investigating sources of imbalance. Most importantly, to the best of my knowledge, no other research has been published with regards to learning using the data from the same course and accounting for a decreasing imbalance ratio as a deadline approaches, as occurs in the experiment above.

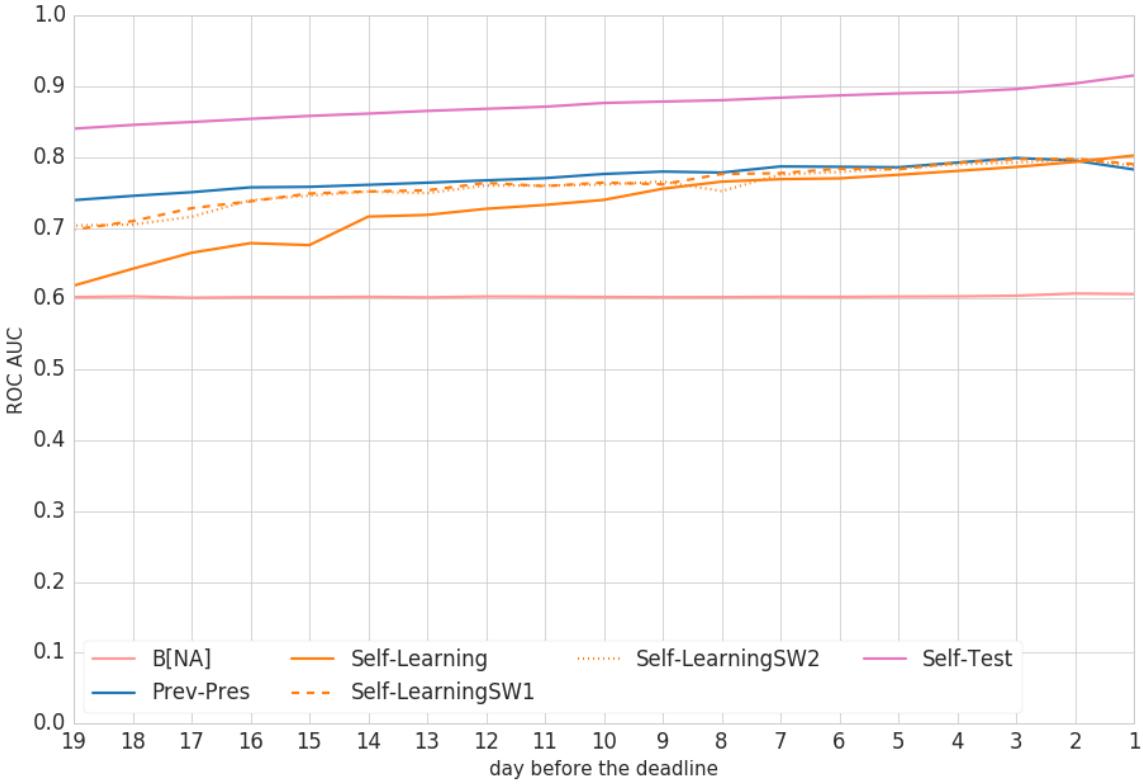

Figure 6.11: ROC AUC for 1 to 19 days before the deadline for different machine learning models using training on the previous presentation.

6.8 Discussion

For training the models, the Self-Learning approach uses data from the same population that is considered for the predictions. At the beginning, the lack of information causes the performance issues. As the window for labelling gets wider, it might be possible to use the information for training in a better way. Several sampling techniques were evaluated in order to improve the performance. These samples were taken from random and informed over-sampling and under-sampling methods. SMOTE-ENN was the only method that improved the performance of the original solution, albeit by only a small amount and only for the PR AUC measure. Random under-sampling performed better for both ROC AUC and PR AUC on two days (13 and 14). The low improvement opens the potential for improvements using knowledge of the domain, for instance, an understanding of the nature of how objects achieve a goal closer to the deadline.

The proposed method is not limited only to prediction of student assessment in distance based Higher Education. Given several conditions, it can be used without adaptation for further assessments and in other educational contexts, for instance MOOCs. It can also be used for different kinds of tasks such as accounting for behavioural data to predict whether students will register for a course. It is important to satisfy the conditions: (1) there needs to be a goal with a specified deadline and (2) there needs to be data (i.e. students) that achieve this goal in advance.

6.8.1 Limitations

When there is no deadline specified, the method would need to be adapted to treat the window for training the model differently. It would be possible to use the same approach for predicting for example student dropout or registration for paid certificate in MOOCs. Similarly, for the second condition, this approach would not be suitable for a high school scenario. Predicting whether the students will finish their studies in time is not feasible because the students are not expected to end in advance.

Different Domain Applications

Although the approach has been evaluated on a case study from the Learning Analytics domain, I expect that the same approach can be used for a broad class of problems, given that they satisfy the conditions mentioned above.

6.9 Summary

This chapter presented the evaluation of the Self-Learning approach in the educational domain, focused on identification of at-risk students in distance learning courses. The generic problem formulation for the goal achievement with a deadline was formulated as a submission of the assessment within the specified day. The key idea is that learning patterns can be extracted purely from the behaviour of students who have already submitted their assessment earlier; thus the approach is applicable to the courses without any historical data.

The data used for the experiments come from The Open University Learning Analytics Dataset, which were anonymised by myself and my colleagues using k-anonymity and l-diversity principles. Also, the dataset is publicly available for other researchers on the OUAnalyse project website[10] and from UCI Machine Learning Repository and on the UCI Machine Learning repository[11].

The experiments showed that the Self-Learning method can successfully predict at-risk students, giving better performance than for the two specified baseline models. As it approaches the deadline, the performance is getting closer to the approach utilising training on the legacy data (the Prev-Pres method). It is even outperforming this method in the last day both for PR AUC and ROC AUC, and for PR AUC even in day 2. A potential point of improvement was identified in the performance gap caused by a lack of data in backwards iteration and class imbalance.

[10] OULAD website: https://analyse.kmi.open.ac.uk/open_dataset

[11] OULAD at UCI Machine Learning repository: https://archive.ics.uci.edu/ml/datasets/Open+University+Learning+Analytics+dataset

Chapter 7

Extending the Self-Learning framework

This chapter builds on the two previous chapters 5 and 6. The experiments provided insight into the properties of the submission prediction process and indicated that exploitation of domain knowledge might provide information useful for further improvement. The first extension deals with the loss of information due to the extending window for labelling the data. The second improvement introduces informed, domain-driven sampling strategies specific to the problem. We can observe that (1) some labelled data are better candidates for removal from the training data and (2) the imbalance decreases towards the deadline.

To evaluate the impact of the improvements, a modified evaluation approach has been proposed. Similarly as in chapter 6, PR AUC and ROC AUC are used to measure the quality of predictions. Again, the testing data were constructed from the examples that have not achieved the expected goal (submission) until the prediction time t_p. Unlike in the previous experiments, the best solution is defined in terms of the minimum performance loss in comparison with the model learnt from the same course with all the labels available after the deadline, described in 6.6.5.

The improvements are evaluated on the same problem setup using the OULAD dataset.

7.1 Issues of Self-Learning

The previous experiments revealed several areas, where the Self-Learning approach might be improved:

7.1.1 Information Loss in Self-Learning

In order to create the classification model based on Self-Learning, the Self-Learning labelling window technique results in two types of information loss. They can be described as follows:

1. **Ignoring objects' behaviour in the labelling window** – the labelling window enables creating a proxy for distinguishing which objects will or will not achieve the goal within the deadline. To simulate the problem, the size, the size of the window is set to the same length as the number of days remaining up to the deadline. However, the data in the labelling window are not used for training the model but only for labelling objects as *Achieve* or *NotAchieve*.

2. **Ignoring early goal achievers** – some objects are not part of the training data because they achieved the goal before the start of the labelling window. More data are excluded since we are closer to the deadline and the window is getting narrower. On the other hand, more objects achieve the goal closer to the deadline, potentially mitigating impact.

7.1.2 Imbalanced Data and Noise

Another issue arises from the imbalance present in the training data. The earlier the predictions are made, the higher the imbalance ratio. This is due to the majority of objects not yet achieving the goal. Some of the data are labelled as $NotAchieve$ for the training purpose, though they will achieve the goal in the end. Consequently, in the prediction time, the data of these objects contribute to the construction of the $NotAchieve$ class though their patterns already indicate that they will eventually belong to the $Achieve$ class. The behaviour of the $NotAchieve$ students can be perceived as a kind of noise in the data, which is one of the problems that accompanies imbalanced data and it, hindering the performance of the classifiers seen in Chapter 2.6. This domain knowledge may be useful in contributing to an under-sampling method.

7.1.3 Different Imbalanced Ratio in Training and Testing Data

In Figure 6.3 above, we can observe the opposite ratio to the class ratio in the training and testing data. According to the classification of problems in imbalanced data in 2.6 item 6, we can identify this phenomenon as an example of a *Dataset shift problem*. Although it might not hold for every domain, in the analysed case study the number of student submissions grows exponentially with the approaching deadline. This property can be used to estimate the number of objects that will achieve the goal and provide a stopping condition for the under-sampling method.

7.2 Improvements

7.2.1 Modifying Labelling Window size

In the Self-Learning strategy introduced in 5.2.1 the size of the labelling window is the same as time to the deadline. It will be denoted as $wSame$. I will relax this condition and introduce an additional parameter specifying the size of the window. This parameter will be denoted as $SizeOfLabellingWindow$. Therefore, in the original Self-Learning strategy, $SizeOfLabellingWindow = |t_d - t_p|$. Shrinking the labelling window allows the algorithms to use more information about each object, as the behaviour of the objects previously used only for labelling is now available for training. However, the number of objects in the labelling window decreases. This will increase the imbalance ratio as fewer objects are used for training, negatively impacting performance (as mentioned in 7.1.1).

The parameter is explained in Figure 7.1 and the date for prediction remains fixed, in that we are predicting on day 4 whether the object will achieve the goal after that day up until the deadline. The first row, length=4, refers to the original solution and the following three rows show the shrinking of the labelling window.

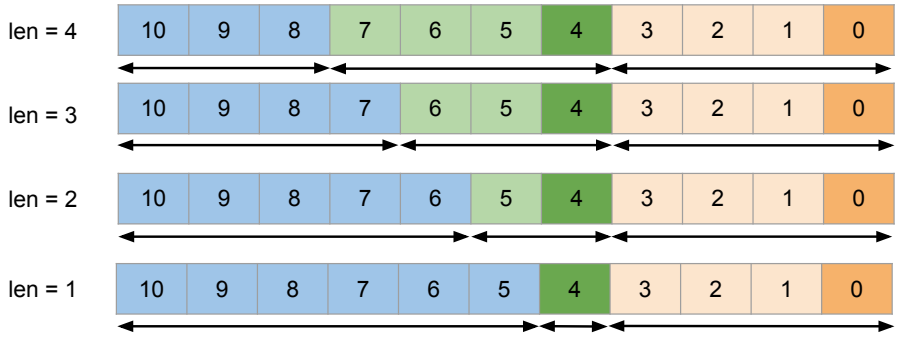

Figure 7.1: Shrinking size of the labelling window for day 4, using the original $SizeOfLabellingWindow = wSame = 4$ and shrinking the window until the $SizeOfLabellingWindow = 1$.

7.2.2 Including Early Goal Achievers

Instead of ignoring the objects that achieved the goal before the start of the labelling window, these objects will extend the training dataset. The time of their goal achievement will be set as a *virtual deadline* and the behavioural features will be aligned with respect to this time. INC refers to the solution with including students, and NOTINC refers to the original solution.

Removal of Objects that Have Achieved the Goal Too Early

In contrast to the previous strategy, I investigate whether or not goals achieved too early negatively influence performance. In the educational context, one can argue for students who were very active and submitted very early being outliers because they are likely to have behaved differently. Thus, we can define the parameter $IncludeBackWindow$, which specifies the maximum number of days from the start of the labelling window that can be used to add the students back to the training data. The days are counted backwards in time. The students that submitted in the interval

$$[virt\_t_d^R + IncludeBackWindow; t_p^R]$$

will be included in the training data. Recall, that $virt\_t_d^R$ denotes the virtual deadline or the start of the labelling window. The minimum value of the parameter is $IncludeBackWindow = 0$, when no additional achievers outside of the labelling window will be added.

7.2.3 Domain Driven Sampling Methods

To decrease the imbalance ratio and eliminate the possible noise in the data, I designed an informed under-sampling method with three different strategies. First, the machine learning algorithm is trained making use of all data, and the model computes probabilities of achieving the goal for all objects, i.e. the confidence of a classifier of being a member of the minority class. Next, I denote the minority class as $c^{min} = 1$ and the majority class $c^{maj} = 0$. Finally, a function $remBottomMajData$ is used to obtain a sample consisting of removed objects from the majority class, which are on the borderline with the minority class. The schema of the approach is described in the Algorithm 2.

I propose three methods of the function $remBottomMajData$ for removing the bottom majority class data:

Algorithm 2: Algorithm for informed under-sampling on higher level.

Input : Training data D, vector of true labels y_true, $|D| = |y|$, Classifier C
Output: Sampled data $D_{sampled}$, $|D_{sampled}| \leq |D|$

Train classifier C on D and y_true
$y\_pred =$ compute probability score for all $x \in D$ using C
Sort D in an ascending way by the probabilities in y_pred
$D_{sampled} = remBottomMajData(D, y\_true, y\_pred, \dots)$
return $D_{sampled}$

Method 1: EqualClassNumber

The usual goal of the sampling algorithms for imbalanced data achieves an equal number of objects in the minority and majority classes. This method achieves this by using the function $removeTopMajority$ in Algorithm 3. The function creates a sample with removed n objects from the majority class with the highest probability of being in the minority class. Let us denote the number of majority class objects n_{maj} and the number of minority class objects n_{min}. The sampling is performed using the function $removeTopMajority(D, y\_true, y\_pred, n_{maj} - n_{min})$.

Algorithm 3: Function removeTopMajority(D,y_true,y_pred,n)

Input : Training data D sorted by the predicted probabilities, vector of true labels y_true, vector of predicted probabilities y_pred, number of objects to remove n
Output: Union of Minority and sampled majority class objects

$D^{min} = \{x_i | y\_true_i = c_{min}\}$
$D^{maj} = \{x_i | y\_true_i = c_{maj}\}$
$n_{keep} = |D^{maj}| - n$
$D^{maj}_{sampled} = n$ objects from D^{maj} with the lowest value of predicted probability of being c^{min}
return $D^{min} \cup D^{maj}_{sampled}$

Method 2: EstimateGoalAchievmentNumber

This method also utilises the $removeTopMajority(D, y\_true, y\_pred, n)$ function in Algorithm 3. Instead of balancing the classes equally, it tries to estimate this number based on the domain information. In our case, we have the data available from the educational domain from the OULAD dataset. If we plot the relative number of students that had this assessment submitted in different days before the deadline, we obtain the graph in 7.2. This suggests that in this case, the number of goal-achievers follows the exponential function, that is, the number of submitted assessments grows exponentially as the deadline approaches.

Using this data it is possible to estimate the parameters of the exponential function, that would be created by the average of all the functions. The function $y(t)$ is defined for time $t \in [0; t_0]$, where 0 denotes the deadline and t_0 the first goal achievement. Following this,

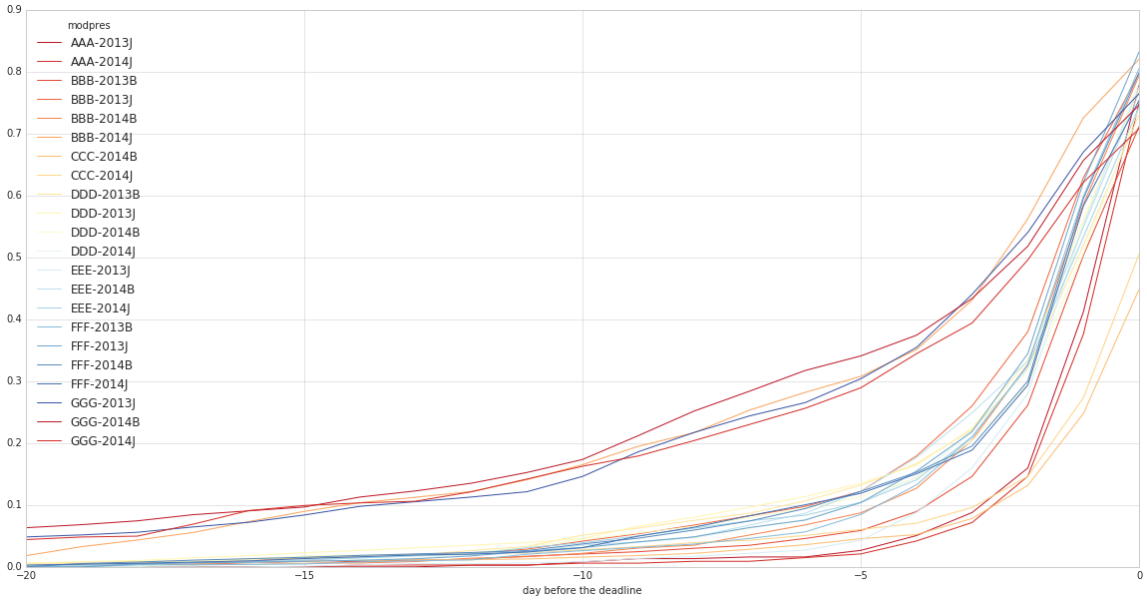

Figure 7.2: Ratio of submitted students in the data in all the courses of OULAD.

$$y(t) = \lambda e^{\beta t} \tag{7.1}$$

, where λ is the estimated number (ratio) of objects achieved the goal in $t = 0$, i.e. in the deadline and β is the coefficient for time t. I took the daily submission data from all the courses in the 2013 presentations and applied the nonlinear regression using least-squares to approximate the parameters of the exponential function. Taking the average for the courses, we get the function with the following parameters:

$$y(t) = 0.7818e^{-0.4167t} \tag{7.2}$$

To perform the under-sampling, we need only an estimate of the function for $t = 0$, i.e. $y(0) = 0.7818e^{-0.4167 \cdot 0} = 0.7818$. If n denotes the number of all the predicted students, the estimated number for removal is $n - 0.7818 \cdot n_{min}$). Consequently, the sampled training data are obtained using the function:

$$removeTopMajority(D, y\_true, y\_pred, n - 0.7818 \cdot n_{min})).$$

Method 3: ClassOverlapRemoval

Instead of removing the fixed number of majority class objects, this method focuses on removing the majority objects that are overlapping with the minority class. First, the lowest prediction probability of the minority class is taken, and then it is used with the procedure $removeMajorityByThr$ in Algorithm 4. The function removes all the data from majority class having the predicted probability to the class c^{min} higher than the threshold.

7.3 Evaluation

Similar to the case study in chapter 6, the enhancements were evaluated on the selected courses on the OULAD dataset from the educational domain. The value to compare was

Algorithm 4: Function removeMajorityByThr(D,y_true,y_pred,threshold)

Input : Training data D sorted by the predicted probabilities, vector of true labels y_true, vector of predicted probabilities, y_pred, $threshold$ defining maximum allowed value of prediction for the majority class

Output: Union of Minority and sampled majority class objects

$D^{min} = \{x_i | y\_true_i = c^{min}\}$

$D^{maj} = \{x_i | y\_true_i = c^{maj}\}$

$D^{maj}_{sampled} = \{x_i | x_i \in D^{maj} \wedge y\_pred_i \leq threshold\}$

return $D^{min} \cup D^{maj}_{sampled}$

the average of all courses per day before the assessment deadline. Following this, I propose a new strategy of how to compare the solutions more easily.

7.3.1 Evaluation Strategy

The evaluation of the models from the case study in chapter 6 was based on comparing the performance measures in each day separately. This is suitable if the number of evaluated models is relatively small and having a small number of days for analysis. If the number increases, it might be difficult to easily recognise the best solution. For example, having 100 different models across 100 days will pose difficulties for using both a table and a graph.

This has led me to propose the evaluation strategy, which would allow quantifying the performance of methods for the whole composite problem in 5.14. The straightforward solution would be to use the average across all prediction days as in equation 5.15. If we use more datasets, as is the case here, we can use the extension in 5.16.

The average value, however, might be biased if one course has significantly different performance than the others. The resulting measure would correctly order the models according to the performance, but the value might be difficult to interpret. Thus, I define the performance in terms of loss of performance to a specified gold standard. This standard is defined as the best model out of those trained on the testing data. Based on the previous experiments, the Random Forest model proved to be the best. So for each course and each prediction day, the problem performance measure ppm will be measured relatively to the performance of this RF model. This approach captures the variability and prediction power of features with respect to the predicted target. Let us define the loss of the model m to the best model m_{best} for a goal achieving problem GP^R as:

$$ppmLoss(D, g, t_d, t_0, t_p^R, cpm, m) = \begin{array}{l} ppm(D, g, t_d, t_0, t_p^R, cpm, m_{best}) - \\ ppm(D, g, t_d, t_0, t_p^R, cpm, m) \end{array} \tag{7.3}$$

The performance loss for the composite problem is defined as the average across the partial problems as:

$$ppmLoss(GP^R, cpm, \boldsymbol{m}) = \frac{1}{t^R_{first}} \sum_{t=1}^{t^R_{first}} ppmLoss(D, g, t_d, t_0, t, cpm, m_t) \tag{7.4}$$

Analogously, instead of comparing the loss against the best solution, it might be useful to express the performance in terms of the gain with respect to the baseline model $m_{baseline}$. Then the equation 7.3 can be rewritten as:

$$ppmGain(D, g, t_d, t_0, t_p^R, cpm, m) = \begin{array}{l} ppm(D, g, t_d, t_0, t_p^R, cpm, m) - \\ ppm(D, g, t_d, t_0, t_p^R, cpm, m_{baseline}) \end{array} \quad (7.5)$$

For given n composite problems in C, it is necessary to compute the sum of their performances. Analogous to Equation 5.16, let us denote the dataset related to the problem $CGP_i \in C$ as D_i. The loss is defined as:

$$ppmLoss(C, cpm, \boldsymbol{MO}) = \frac{1}{n \cdot t_{min0}^R} \sum_{i=1}^{n} \sum_{t=1}^{t_{min0}^R} ppmLoss(D_i, g, t_{di}, t_{0i}, cpm, t, m_{i,t}) \quad (7.6)$$

The performance gain would be defined in the same way.

7.4 New Evaluation of the Original Approach

The results of the original solution are presented in terms of the new evaluation strategy, both with and without using the sampling methods. The losses of both PR AUC and ROC AUC are shown. Table 7.1 lists the results of the Self-Learning approach. Both for PR AUC an ROC AUC, the lowest loss of performance was achieved by Random Forest, which confirms the results from the previous experiments. The loss is lower for ROC AUC, which holds for all the models. The loss of Random Forest is averaged as almost half of the loss of the baseline model $B[NA]$.

The results for training the model using the previous presentation are similar and depicted in Table 7.2. Again, the results confirm the lowest loss using Random Forest. The difference against Self-Learning is 0.0701 for PR AUC and 0.0456 for ROC AUC.

Table 7.1: PR AUC loss ($Loss_{PR}$) and ROC AUC loss ($Loss_{ROC}$) on the selected courses using all the machine learning models and the sampling methods, trained using Self-Learning

	B[NA]	B[NS]	LR	LR-W	NB	RF	SVM	SVM-W	XGB
$Loss_{PR}$	0.3285	0.4366	0.4006	0.2694	0.4143	**0.1788**	0.3693	0.2322	0.3262
$Loss_{ROC}$	0.2715	0.3746	0.3811	0.1888	0.3319	**0.1425**	0.2662	0.1710	0.2377

Table 7.2: PR AUC loss ($Loss_{PR}$) and ROC AUC loss ($Loss_{ROC}$) on the selected courses using all the machine learning models and the sampling methods, trained on the previous presentation of the course.

	B[NA]	B[NS]	LR	LR-W	NB	RF	SVM	SVM-W	XGB
$Loss_{PR}$	0.3203	0.4284	0.1682	0.1327	0.3568	**0.1087**	0.1656	0.1224	0.1700
$Loss_{ROC}$	0.2657	0.3693	0.1521	0.1099	0.2449	**0.0969**	0.1374	0.1034	0.1291

The new evaluation strategy allows the easier comparison of the sampling methods for all the models. Tables 7.3 and 7.4 show this for PR AUC and for ROC AUC respectively. These tables indicate that the lowest overall loss was achieved by Random Forest. For PR AUC, SMOTE-ENN performed best, followed by random over-sampling. The solution without any sampling was worse by 0.0074. For ROC AUC the lowest loss was achieved

by NCR, but again, with only small improvement 0.0012 over Random Forest without any sampling method.

For, PR AUC, SMOTE-ENN was the technique that improved the performance best for four of the models and random under-sampling for the other three. The results are the same for ROC AUC, with the only exception being Random Forest with NCR. The highest impact of sampling methods was achieved for LR decreasing the loss of PR AUC by 0.1958 and for SVM by 0.1351. A similar result had been achieved for ROC AUC, but the gap between LR and SVM has widened.

Table 7.3: PR AUC loss on the selected courses using all the machine learning models and the sampling methods.

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
B[NA]	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285	0.3285
RF	0.1800	0.1786	0.1788	0.1879	0.1741	0.1826	0.1839	**0.1714**	0.1783
XGB	0.3239	0.3238	0.3262	0.2481	0.2441	0.2549	0.2554	0.2360	0.3263
B[NS]	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366	0.4366
LR-W	0.2570	0.2605	0.2694	0.2911	0.2048	0.3337	0.3340	0.2290	0.2688
LR	0.3788	0.3849	0.4006	0.2911	0.2048	0.3370	0.3373	0.2266	0.4005
NB	0.4055	0.4081	0.4143	0.4154	0.3594	0.4405	0.4401	0.3780	0.4139
SVM	0.3430	0.3506	0.3693	0.2666	0.2654	0.3111	0.3106	0.2342	0.3686
SVM-W	0.2257	0.2263	0.2322	0.2666	0.2654	0.3107	0.3095	0.2253	0.2306

Table 7.4: ROC AUC loss on the selected courses using all the machine learning models and the sampling methods.

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
B[NA]	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715	0.2715
RF	0.1434	**0.1413**	0.1425	0.1543	0.1432	0.1606	0.1616	0.1436	0.1428
XGB	0.2330	0.2342	0.2377	0.1849	0.1796	0.1972	0.1971	0.1725	0.2379
B[NS]	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746	0.3746
LR-W	0.1817	0.1837	0.1888	0.2055	0.1490	0.2408	0.2408	0.1698	0.1885
LR	0.3557	0.3629	0.3811	0.2055	0.1490	0.2522	0.2521	0.1784	0.3812
NB	0.3201	0.3232	0.3319	0.3334	0.2552	0.3833	0.3826	0.2881	0.3314
SVM	0.2461	0.2508	0.2662	0.1902	0.2780	0.2249	0.2246	0.1762	0.2656
SVM-W	0.1679	0.1676	0.1710	0.1902	0.2780	0.2243	0.2226	0.1673	0.1702

7.5 Improvements Results

Random Forest performed best in all experiments and all training strategies, i.e. Self-Learning, training using legacy data and training on the testing data. For this reason, I present only the analysis of improvements performed on Random Forest.

7.5.1 Modifying Labelling Window Size

Changing the size of the labelling window enables us to compare whether it is more important to have additional information about the individual student or more students who submitted (i.e. the minority class) in the training data. For each day of the prediction, the size of the labelling window has been changed from 1 to 19.

The results for various sizes of the window were compared to the original solution *wSame* when the size changed with respect to the number of days remaining up to the deadline. Also, for comparison, I built the models both with and without including the submitted students before the start of the labelling window. Figure 7.3 and Table 7.5 demonstrate the results for both ROC and PR AUC losses. INC denotes the solution with including students, NOTINC is the original solution, i.e. without these students.

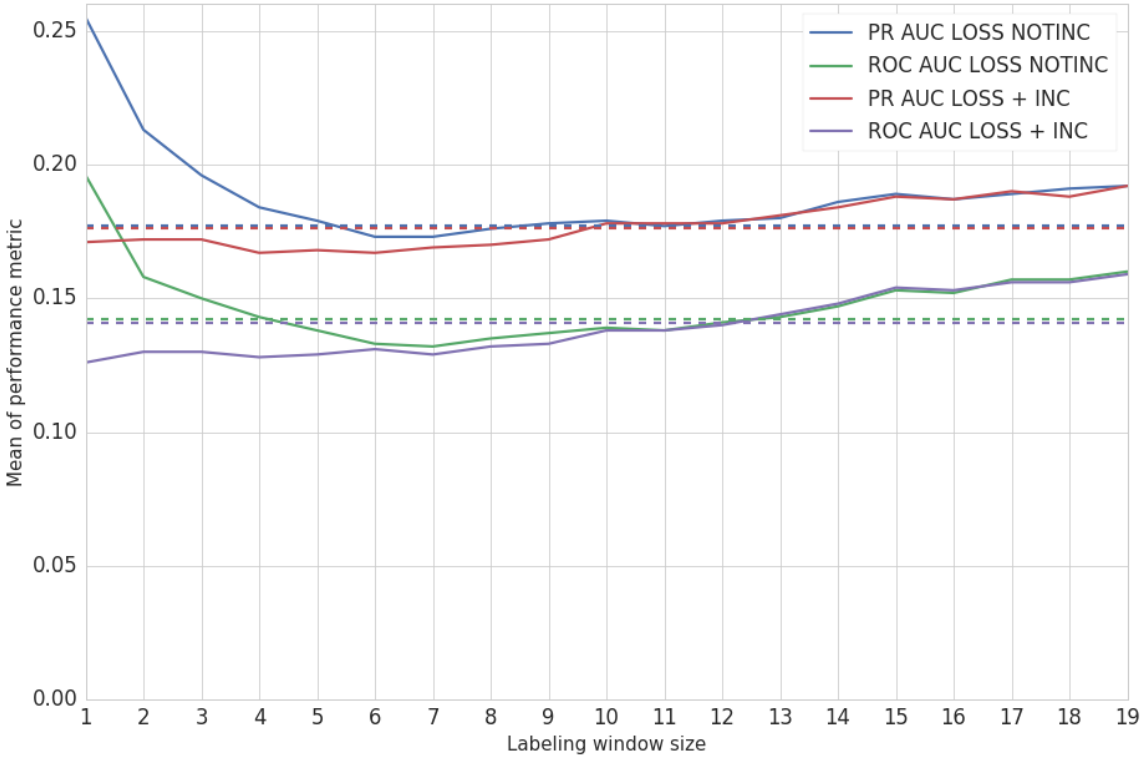

Figure 7.3: Loss of PR AUC and ROC AUC for different sizes of the labelling window and the influence of including the early achievers in the training data. INC denotes including students that submitted the assessment before the start of the labelling window. The dotted lines correspond to the *wSame* approach, being comparable to the strategy denoted with the same colour with full line.

The predictions have been computed for days 1 to 19, and the performance losses have been averaged across all prediction days and across all the courses. With the *wSame* strategy, the performance measure was also computed across days and courses. This value is independent of the parameter for the fixed window size $SizeOfLabellingWindow$; therefore the value is constant, and it is represented as a horizontal line. Therefore, two different window sizes, both with INC and NOTINC strategies, will result in four possible strategies, resulting in as many different models. Afterwards, we compute their performance loss

according to the Equation 7.6. Two window sizes 1 and 2 would result in strategies created by $(1, INC)$, $(1, NOTINC)$,$(2, INC)$,$(2, NOTINC)$.

The Figure 7.3 shows that the loss of PR AUC is higher than that for ROC AUC. This is consistent with the previous results. For both measures, the loss slowly decreases for INC and NOTINC from size 19 to size 7. For the sizes 19 to 10, the differences the difference between the INC and NOTINC are only 0.001. Moving from the window 9 to 1, the differences start increasing, mainly because the loss of NOTINC starts increasing exponentially until the window size 1. This is caused by increasing the imbalance in the training data due to narrowing the labelling window and consequently the number of students submitting in this interval. However, close to the deadline, a problem may not emerge, because enough students submit during the interval. This is confirmed by the results from the original solution, described in Chapter 6, where the highest performance is achieved in the last days despite a small window size. However, the performance decreases for days further from the deadline t_d.

Adding the early submitting students also helps to mitigate the impact of the narrow window. For PR AUC the loss is the lowest for size 4. For ROC AUC, the loss is decreasing until the end of window size 1.

For comparison, the dotted lines in Figure 7.3 denote the $wSame$ solutions. For both metrics, the values for NOTINC and NOTINC of $wSame$ are almost equal, note the last row in Table 7.5. This means, that including the students for $wSame$ itself does not significantly improve the performance.

For NOTINC, there is an interval with loss lower than for the $wSame$. For PR AUC it is $[5; 9]$ and for ROC AUC $[5; 11]$. Due to narrowing window, the performance degrades from the window size 7 to 1.

The results showed the best performance improvement for $SizeOfLabellingWindow \leq 7$ and trained with including the early submitting students. Window sizes $[1; 7]$ were selected for evaluation and analysis for improvement of the model. Window size 1 is the global minimum of the ROC AUC loss and 4 and 6 are the global minimums for the PR AUC loss. ROC AUC has a local minimum for the size 7.

7.5.2 Removal of Very Early Achievers

As shown, including students that submitted before the start of the labelling window increases the performance. The question is, whether the students who submitted a long time before the start of the labelling do not hinder the performance. Especially, as getting closer to the deadline, one might expect that students who submit very early behave differently than those who submit at the last moment.

Having $SizeOfLabellingWindow = 1$, I varied the maximum number of days (IncludeBackWindow) before the labelling window that is allowed for a student to be added to the training data. The value of the parameter was set from 0 to 40, which is as far as considering infinity, given that the maximum deadline in the dataset is 61, seen in course GGG. If the data pattern of the early achievers was different, we would notice the decrease of performance measures for increasing value of the parameter $IncludeBackWindow$. The results in Figure 7.4 show this neither for PR AUC loss nor for the ROC AUC loss. The only visible trend is the exponential increase of loss when lowering the maximum window size. The further analysis showed that the main loss does not come from the days close to the deadline, but those that are far away.

Table 7.5: Loss of PR AUC and ROC AUC for different sizes of the labelling window and the influence of including the early achievers in the training data. INC denotes including students that submitted the assessment before the start of the labelling window.

WinSize	PR AUC LOSS NOTINC	ROC AUC LOSS NOTINC	PR AUC LOSS INC	ROC AUC LOSS INC
1	0.255	0.196	0.171	**0.126**
2	0.213	0.158	0.172	0.130
3	0.196	0.150	0.172	0.130
4	0.184	0.143	**0.167**	0.128
5	0.179	0.138	0.168	0.129
6	**0.173**	0.133	**0.167**	0.131
7	**0.173**	**0.132**	0.169	0.129
8	0.176	0.135	0.170	0.132
9	0.178	0.137	0.172	0.133
10	0.179	0.139	0.178	0.138
11	0.177	0.138	0.178	0.138
12	0.179	0.141	0.178	0.140
13	0.180	0.143	0.181	0.144
14	0.186	0.147	0.184	0.148
15	0.189	0.153	0.188	0.154
16	0.187	0.152	0.187	0.153
17	0.189	0.157	0.190	0.156
18	0.191	0.157	0.188	0.156
19	0.192	0.160	0.192	0.159
same	0.177	0.142	0.176	0.141

In conclusion, including all the students back into the analysis, even with the size of the labelling window 1, does not negatively influence performance.

7.5.3 Domain Driven Sampling Methods

Taking the best results from the previous experiments, labelling window of sizes $1-7$ were taken for evaluation together with the original window, (i.e. $wSame$). The three presented sampling methods were compared with each other and to the results without any sampling. Again, the loss of PR AUC and ROC AUC were the measures of interest.

Table 7.6 shows the loss of PR AUC. EQ_CLS denotes the sampling with the equal number of data in both classes, EST_RAT is the estimation of the submission ratio, and RM_OVLAP stands for the removal of the overlap between the classes. For RM_OVLAP, 100 denotes the removal of all the majority data that overlap with the minority class and 25 for the 25% of the borderline minority data. Table 7.7 shows the same analysis using ROC AUC loss as the measure.

Results indicate that the best performance for both measures is achieved for the EST_RAT. With the best PR AUC achieved for window 2, loss was decreased from 0.1716 to 0.1417, i.e.by 0.0299. Window 4 has loss of only 0.0010 higher, i.e. 0.1427. For window 1 the loss is 0.1432. The results for ROC AUC are similar, EST_RAT for window size 1 achieving the best results with the loss 0.1133 followed by EST_RAT with window size 2 with a loss of 1162.

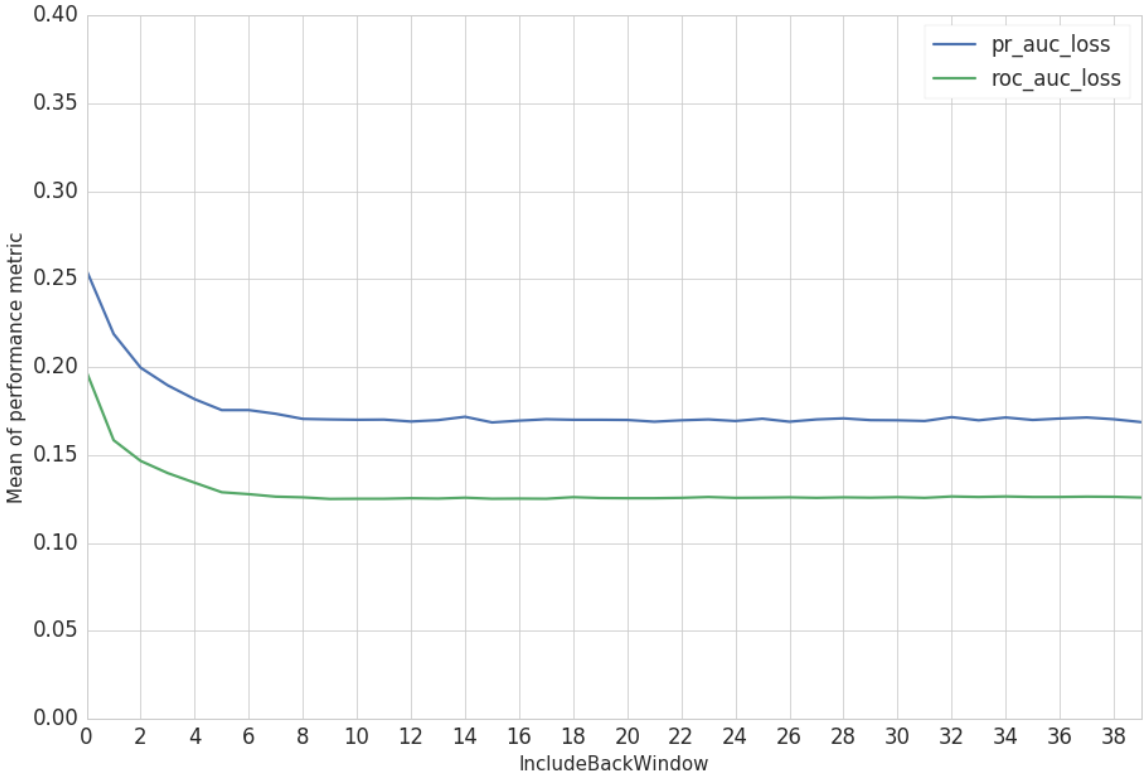

Figure 7.4: Loss of PR AUC and ROC AUC for varying maximum days from the start of the labelling window for students to be included back in the training data, for $SizeOfLabellingWindow = 1$

From the other methods, EQ_CLS improved performance but only for PR AUC. The problem with EQ_CLS is probably the removal of too many students, retaining mainly the most obvious submitters. For the PR AUC, it still performs well.

Table 7.6: PR AUC loss for domain-driven sampling techniques in various window sizes, winSize denotes the SizeOfLabellingWindow parameter

winSize	EQ_CLS	EST_RAT	None	RM_OVLAP_100	RM_OVLAP_75
1	0.1559	0.1432	0.1707	0.1772	0.1728
2	0.1554	**0.1417**	0.1716	0.1781	0.1730
3	0.1594	0.1447	0.1716	0.1750	0.1728
4	0.1596	0.1427	0.1672	0.1746	0.1684
5	0.1616	0.1450	0.1680	0.1737	0.1691
6	0.1625	0.1458	0.1674	0.1723	0.1692
7	0.1644	0.1484	0.1690	0.1733	0.1682
wSame	0.1833	0.1603	0.1765	0.1814	0.1783

Table 7.7: ROC AUC loss for domain-driven sampling techniques in various window sizes, winSize denotes the SizeOfLabellingWindow parameter

winSize	EQ_CLS	EST_RAT	None	RM_OVLAP_100	RM_OVLAP_75
1	0.1317	**0.1130**	0.1265	0.1321	0.1279
2	0.1330	0.1160	0.1297	0.1352	0.1304
3	0.1377	0.1174	0.1304	0.1347	0.1308
4	0.1368	0.1170	0.1279	0.1361	0.1289
5	0.1374	0.1199	0.1288	0.1360	0.1299
6	0.1395	0.1220	0.1307	0.1352	0.1327
7	0.1395	0.1233	0.1289	0.1364	0.1300
wSame	0.1597	0.1345	0.1408	0.1470	0.1430

Sampling With the Original Solution

The sampling methods were used to improve the performance of the original version of Self-Learning, adjusting the labelling window and not including students submitting before the start of the window. Table 7.8 shows the results for both PR AUC and ROC AUC loss. Two important findings are that (1) the EST_RAT performs again best for both measures and (2) the results for sampling confirm the previous finding that using a smaller labelling window leads to a better performance.

Table 7.8: PR AUC and ROC AUC loss for domain-driven sampling techniques for the original version of Self-Learning

sampler	PR AUC LOSS	ROC AUC LOSS
EQ_CLS	0.1824	0.1570
EST_RAT	**0.1603**	**0.1349**
None	0.1772	0.1415
RM_OVLAP_100	0.1781	0.1443
RM_OVLAP_75	0.1779	0.1420

7.5.4 Comparison With Existing Sampling Methods

Existing sampling methods were applied to data with window sizes $1 - 7$ and compared with the domain-driven methods. The results for PR AUC loss in Table 7.9 and for ROC AUC in Table 7.10 indicate that the best-performing method, in general, is random under-sampling. For PR AUC it reaches the minimum loss for the window 2. For ROC AUC the global minimum is achieved by the SMOTE-ENN for window size 1, but in the other windows, random under-sampling performs better. Decreasing the window size helps the performance, but not as much as for the domain-driven sampling.

Daily Performance Analysis

Both Random Under-Sampling and SMOTE-ENN were selected together with the two best performing domain-driven methods, i.e. EST_RAT and EQ_CLS, all of them with the parameter $SizeOfLabellingWindow = 1$. Their results in terms of average absolute

Table 7.9: PR AUC loss for sampling techniques in various window sizes

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
1	0.1712	0.1714	0.1707	0.1714	0.1573	0.1720	0.1699	0.1629	0.1709
2	0.1729	0.1710	0.1716	0.1723	**0.1554**	0.1714	0.1704	0.1632	0.1719
3	0.1712	0.1709	0.1716	0.1720	0.1594	0.1725	0.1717	0.1629	0.1691
4	0.1692	0.1692	0.1672	0.1713	0.1595	0.1683	0.1672	0.1609	0.1666
5	0.1693	0.1670	0.1680	0.1690	0.1578	0.1662	0.1673	0.1593	0.1677
6	0.1716	0.1687	0.1674	0.1733	0.1571	0.1692	0.1689	0.1593	0.1668
7	0.1710	0.1706	0.1690	0.1749	0.1589	0.1690	0.1686	0.1591	0.1675
wSame	0.1777	0.1774	0.1765	0.1867	0.1694	0.1782	0.1771	0.1693	0.1773

Table 7.10: ROC AUC loss for sampling techniques in various window sizes

	ENN	NCR	None	Rand Over	Rand Under	SMOTE Tomek	SMOTE	SMOTE ENN	Tomek Links
1	0.1260	0.1256	0.1265	0.1310	0.1260	0.1359	0.1349	**0.1235**	0.1256
2	0.1308	0.1302	0.1297	0.1338	0.1256	0.1383	0.1382	0.1277	0.1294
3	0.1306	0.1303	0.1304	0.1355	0.1272	0.1414	0.1409	0.1277	0.1297
4	0.1298	0.1288	0.1279	0.1347	0.1275	0.1382	0.1385	0.1282	0.1272
5	0.1294	0.1298	0.1288	0.1341	0.1278	0.1378	0.1388	0.1286	0.1285
6	0.1325	0.1311	0.1307	0.1374	0.1278	0.1412	0.1409	0.1297	0.1305
7	0.1311	0.1313	0.1289	0.1361	0.1307	0.1404	0.1409	0.1298	0.1301
wSame	0.1415	0.1414	0.1408	0.1534	0.1409	0.1549	0.1539	0.1421	0.1417

performance are plotted in Figure 7.5 and 7.6. For both measures, the best performer is EST_RAT. For PR AUC, however, in days $14-11$ the EQ_CLS performs slightly better. Furthermore, the main increase of performance of the sampling methods occurs between days $19-10$. From day 10 to the deadline, the differences are negligible, apart from day 1. In day 1, the EQ_CLS method performs worse than the other methods. The plotted results for window size 2, which achieves slightly higher results for ROC AUC, were consistent with Figure 7.5 and 7.6, thus they were omitted in the figures for brevity.

7.5.5 Impact of the Improvements

To make the impact of the single improvements and their combination clear, I selected the best results for the window analysis (i.e. window sizes 1 and 2) and the best sampling method: EST_RAT. I compute their losses both separately and combined.

Table 7.11 shows the losses of PR AUC and ROC AUC and the difference between the original solution *original* and the loss of the improvement. The differences are denoted as $d\_pr\_auc\_loss$ and $d\_roc\_auc\_loss$. This reveals that the highest individual contribution is achieved by EST_RAT sampling for PR AUC and by the window size 1 for ROC AUC. The combination of improvements substantially contributes to the results substantially. For example, while using only $w2$ leads to a difference of 0.0072 and using EST_RAT to 0.0184 for PR AUC, their combination makes the difference $d\_pr\_auc\_loss = 0.0371$.

Finally, Figures 7.7 and 7.8 show the impact of improvements per day in the context of the baseline model, the model trained on the previous presentation and the model trained on the testing data. EST_RAT with both window sizes 1 and 2 are presented, denoted

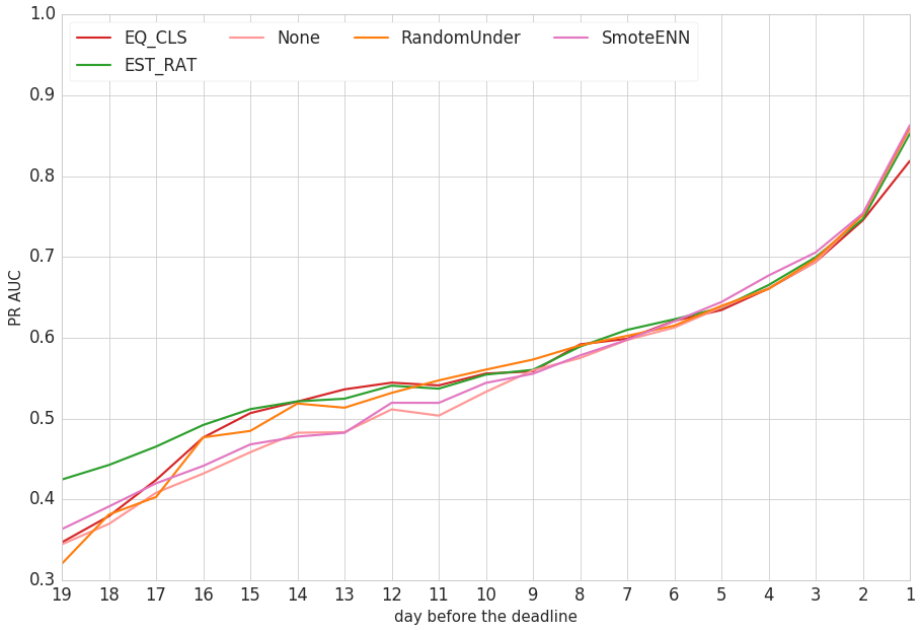

Figure 7.5: Daily comparison of PR AUC for sampling methods

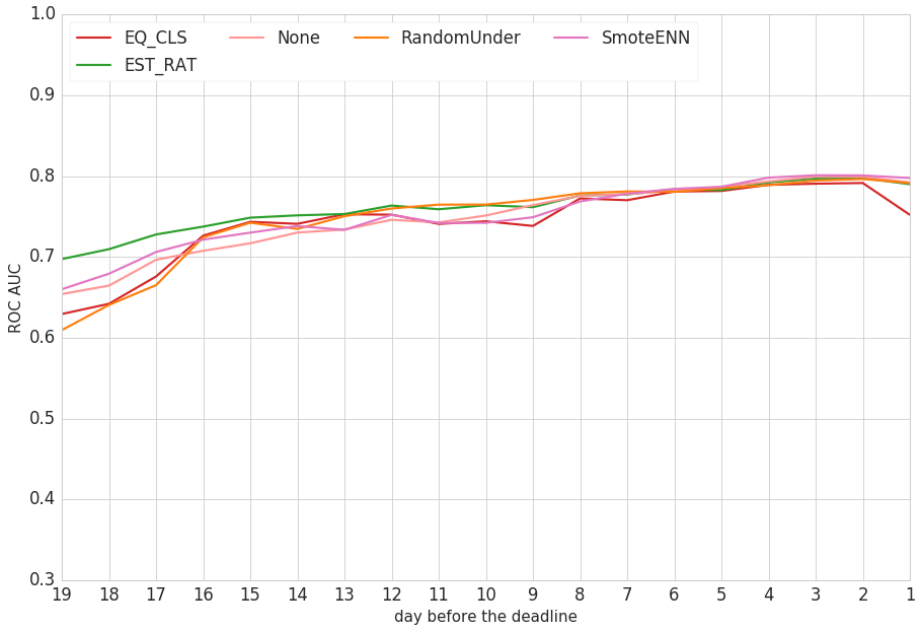

Figure 7.6: Daily comparison of ROC AUC for sampling methods

in the figure as *Self-LearningSW1* and *Self-LearningSW2*. Both of them improved both *PRROC* and ROC AUC especially in the early phases of predictions. They narrowed the performance gap, especially to the *PrevPres* strategy. For example, the difference for ROC AUC instead of being visible from day 10 back to the past, is now visible around days 16-17.

Table 7.11: Performance loss of individual best improvements and their combination

	pr_auc_loss	auc_loss	d_pr_auc_loss	d_roc_auc_loss
original	0.5434	0.7322	0.0000	0.0000
w1	0.5515	0.7482	0.0081	0.0160
w2	0.5506	0.7450	0.0072	0.0128
EST_RAT	0.5618	0.7397	0.0184	0.0075
w1 + EST_RAT	0.5790	**0.7617**	0.0356	**0.0295**
w2 + EST_RAT	**0.5805**	0.7586	**0.0371**	0.0264

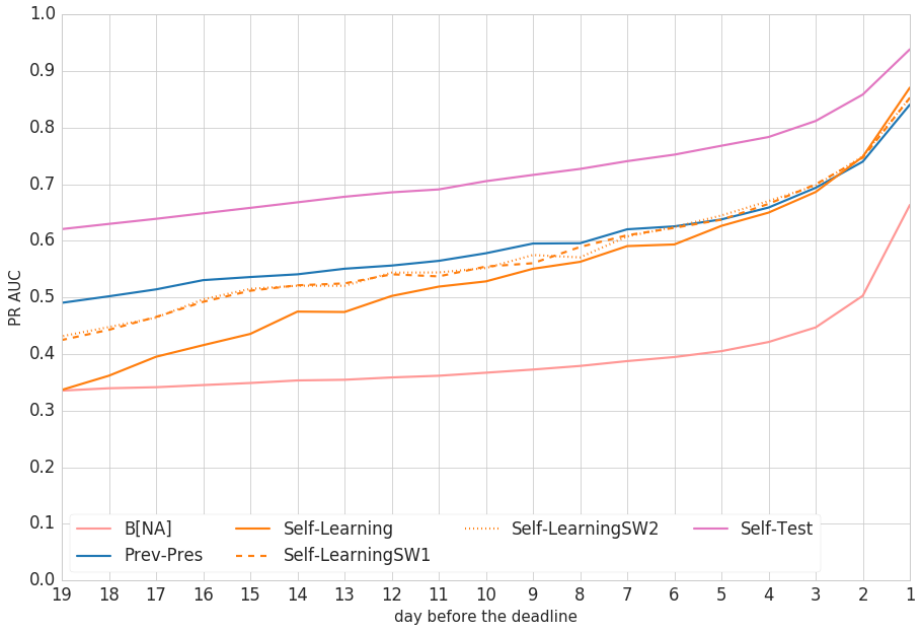

Figure 7.7: Comparison of PR AUC of the best Self-Learning improvement with the best solutions from the original approach

7.5.6 Summary Results

This chapter discussed issues pertaining to the original Self-Learning approach. Based on these issues, two types of improvements were suggested: (1) modifying labelling window size, including students submitting before the start of the window and (2) using domain-driven sampling methods.

To better evaluate the impact of the improvements, a new evaluation strategy was defined. This strategy is based on computing the loss against the model that was trained on the same data as tested, representing the limits of what the model can explain based on the given features.

Best parameters were found for the size of the labelling window (window size = 1 and 2) and also the best performing sampling-method (EST_RAT). The results showed that both of the improvements decrease the loss made against the selected model in comparison to the original version of Self-Learning. It was shown that loss reaches the minimum when both improvements are used in combination.

It is worth noting that such improvements were achieved primarily due to knowledge of the selected domain where the approach was evaluated. In the case of another domain,

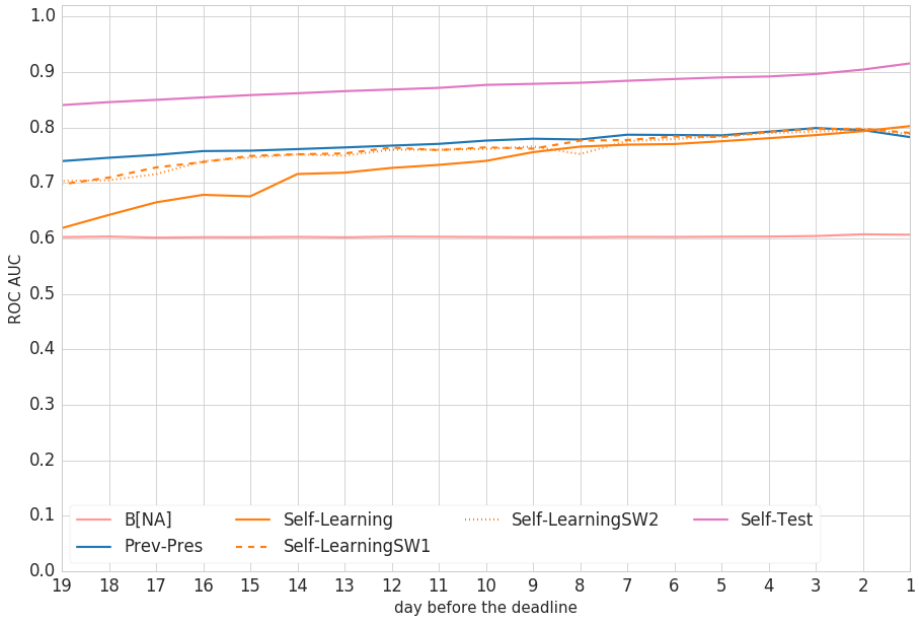

Figure 7.8: Comparison of ROC AUC of the best improvement with the best solutions from the original approach

new evaluation needs to be performed to confirm or refute the findings. Nevertheless, the notion behind both of the improvements should be transferable to other domains.

The widening of the labelling window when going back in time will probably negatively influence any domain, and I expect that shrinking it to a smaller size should always lead to better results. The expectation of the sampling method is that some of the data that are labelled as negative will, in the end, become positive. If this is true for the problem, then knowledge of the function that characterises the number of achieved goals in time is expected to improve performance.

Chapter 8

Conclusions

The focus of this Phd thesis is on the classification of imbalanced data. This is an important aspect of machine learning where the objective is to address the issues when one class is significantly underrepresented with respect to another. The minority class usually has higher importance, but the traditional techniques have led to models that favour the performance of the majority class.

Various application domains often bring new kinds of problems that, if generalised, might also contribute to other areas with the similar problem. My research was motivated by two application domains, which by nature contain imbalanced data: classification of malicious/benign files in Computer Security, and early identification of at-risk students in Learning Analytics. The primary research goal was to investigate solutions for the identified specific problems in these application domains in the classification of unbalanced data, and also how the domain can influence a solution to the problem..

The challenge of the computer security domain was in having a large dataset with class imbalance 1:99 with an additional constraint on the quality of the predictions for the minority class. This constraint gains its motivation from usage in the production of a computer security-oriented company. The goal was to maximise the performance of the majority class (Specificity) while ensuring the minimum specified classification performance in the minority class (Sensitivity). The literature review revealed that only a few methods for classification with constraints exist, none of them focused on the unbalanced data. I presented a technique combining the Genetic Algorithm that utilised candidates generated in the previous step by an edited version of Cost-Sensitive Logistic Regression. This was further modified by replacing the Genetic Algorithm by Particle Swarm Optimisation. The results showed that such an approach is always able to improve the results of Logistic Regression without significant violation of the key constraints. The choice of linear models, which were optimised, was also supported by easily explainable resulting models, explained to security experts.

Further, the prevalence of binary features in the evaluated data influenced the performance, and was the likely reason for why sampling methods were not suitable candidates. For example, SMOTE usually achieves good results, but will not likely create new data examples that are near the minority class. In such cases, the value of the attribute would have to be flipped, i.e. from 1 to 0 or vice-versa.

If this occurs, the domain played a role in defining the problem while also suggesting the source of the imbalance. This is caused by not having enough data from the minority class, and we can not estimate how the distribution in the classes will evolve in the future. Also,

feature types and knowledge of the target users directed the usage of explainable models by using weights of the same features.

The Learning Analytics domain, and identification of at-risk students, in particular, motivated the need to articulate the general problem of achieving goals within a deadline. In this problem, there are no legacy data for training the model. As such, the problem faces a large imbalance especially in the beginning as only a few objects satisfy the goal very early. I proposed the Self-Learning framework and evaluated it in a case study of predicting at-risk students. In this domain, the lack of legacy data means that the course is presented for the first time and there is no other course that is structurally similar in order to provide data for building the predictive model by machine learning algorithms. The experiments showed that Random Forest is the best performing base classifier, with a slight performance improvement when combined with SMOTE-ENN sampling. The approach showed the predictive power, but there was also a performance gap with respect to training the model using legacy data from the previous presentation of the same course. Based on knowledge about the problem, I designed two modifications that tackle the loss of information and the imbalance in the data caused by the noise. This is crucial especially in the beginning of the predictions. Both of these modifications improved the performance of the original solution, with the best results achieved when combined together.

To evaluate the quality of the suggested solutions and modifications, I designed new evaluation strategies that measure the performance summarised both across all prediction times and all datasets (here the available courses) with the classification measure by a single value. Instead of counting the absolute value, it calculates the loss against the best achievable model, which is the one trained on the testing dataset. Relating performance measure of the method by comparing with the theoretical baseline makes it possible to evaluate the methods while eliminating the impact of other factors, such as using different data and features.

Here, the domain helped to define the problem of achieving a goal within the deadline, revealing that the problem naturally generates imbalanced data. Moreover, the information about the process helped to realise the loss of information in the original solution and guide the design of the sampling method. The underlying process of student submission generates high activity and more submissions close to the deadline. The presence of a deadline is something, what makes this problem unique and influences the behaviour of the participating subjects, i.e. students. Most probably it is a problem that is specific to a human behaviour. One of the possible explanation is a procrastination, a phenomenon of preferring short term goal over the long term goals and then postponing the activity until the very end [57, 54].

8.1 Summary of contribution

The whole thesis can be summarised in the following points, divided by the two research questions:

1. Constrained classification problem
   (a) Definition of the problem for classification of imbalanced data with the performance constraint specified on the minority class and maximising the performance of the majority class (Section 4.1).
   (b) Development of a method combining the cost-sensitive logistic regression with stochastic algorithms that solves the constraint classification problem; stochastic

algorithm always improved the performance of the previous solution and violated the constraint only marginally, (The version with Genetic Algorithm in Section 4.2, the method with PSO in Section 4.3).

2. Goal Achieving problem with the deadline

   (a) The Self-Learning concept for training models for prediction of goal achievement by objects within a specified deadline, with a natural presence of imbalanced data especially in the beginning of the training (Chapter 5).

   (b) Evaluation of the approach on a case-study from the distant Higher Education by identification of students at-risk of not submitting the assessment from the data on the running course, i.e. without the need of any legacy data from the previous run of the course (Chapter 6).

   (c) Using the information about the problem to extend the framework and improve the performance by (1) parametrised labelling windows size and including objects that were not included in the labelling window; (2) designing a domain-driven under-sampling strategy with estimating the number of expected objects that will achieve the goal. Both strategies tested on the lead separately to performance improvement reaching best results when combined together. In this way, the performance narrows the gap between the Self-Learning and the theoretical possibilities of the machine learning defined by training on the testing data (Chapter 7).

To conclude, I identified two types of problems for imbalanced data classification that were a result of the used application domain, the goals defined in Chapter 1. Both of them have specifics that to the best of my knowledge have not been published. In Chapter 4, I showed how having a performance constraint on the minority class was a result of a computer security domain. I presented a method combining logistic regression and stochastic algorithms for solving the problem, with violating the constraint only marginally. The Learning Analytics motivated to define the general problem of predicting achieving the goal with the deadline without legacy data, described in Chapter 5. The designed Self-Learning approach evaluated on the Learning Analytics domain in Chapter 6 showed its predictive power against baseline models and that methods for tackling the class imbalance without domain information did not lead to significant improvements. This information has been implemented, and an extended version has been evaluated in Chapter 7, confirming that knowing the source of imbalance can lead to better results results if used properly.

8.2 Future research

Future research for the **constrained classification** problem can lead to the investigation of other than linear models, such as rule-based models. In these cases, other algorithms besides Logistic Regression should be used for preparing the candidate solutions for the stochastic algorithms. If other features besides binary ones are available, sampling techniques, which are usually based on nearest neighbours, can be explored instead of using cost-sensitive learning. In other research, the recent growth of Deep Learning and Explainable Artificial Intelligence (for example LIME[1] [85]) might overcome both of the performance issues we

[1]LIME - Local Interpretable Model-Agnostic Explanations

tackled, and lead to the models that are interpretable and understandable to the target users, for instance security experts.

Also, for the computer security domain or similar type of problem, there might exist many data that are unlabelled, as the labelling can be costly or even impossible. Utilisation of semi-supervised techniques could improve classification performance. Also, recent improvements in computation power along with Deep Learning can be used not only for supervised classification but also to use unlabelled data to increase their performance. For example, [110] used the Deep Belief Network to pre-train the generative classifier purely from unlabelled data. For imbalanced data, possibly only highly different cases from the normal distribution or the ones that are close to the malicious file might turn up to improve the classification. Another way is using such an approach to identify the data that should next be manually examined by the domain experts, i.e. security researchers. Consequently, I believe that providing unlabelled data should enable enhancement of the methods and improve the results for constrained classification in imbalanced data.

In the case of the **goal achievement problem**, several possible avenues for further research are possible. First, the suitability of the method across different domains can be investigated and possibly discover whether the domain specific improvements are generalisable in different contexts. Second, from the perspective of Learning Analytics, the Self-Learning approach can be combined with the traditional way of learning from the legacy data to improve the prediction power.

Moreover, the theoretical properties of the underlying process can be studied with more focus on parameters influencing the distribution of achievement times. Investigating which parameters affect the submission of the assessment, or achieving goals in general, can lead not only to an additional classification improvement but also to better understanding of this process. Parameters that would allow controlling the process may be discovered, and the process can be optimised so that more objects achieve the goal.

Bibliography

[1] Abraham, A.; Grosan, C.; Ramos, V.: *Swarm intelligence in data mining*. Berlin; New York: Springer. 2006. ISBN 9783540349563 3540349561 9786610744220 661074422X 3540349553 9783540349556.

[2] Amnueypornsakul, B.; Bhat, S.; Chinprutthiwong, P.: Predicting Attrition Along the Way: The UIUC Model. In *Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs*. Doha, Qatar: Association for Computational Linguistics. October 2014. pp. 55–59.
Retrieved from: http://www.aclweb.org/anthology/W14-4110

[3] Anand, R.; Mehrotra, K. G.; Mohan, C. K.; et al.: An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*. vol. 4, no. 6. 1993: pp. 962–969.

[4] Bahnsen, A. C.; Aouada, D.; Ottersten, B.: Example-dependent cost-sensitive logistic regression for credit scoring. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*. IEEE. 2014. pp. 263–269.

[5] Bailey, N. T.; et al.: *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE.. 1975.

[6] Bainbridge, J.; Melitski, J.; Zahradnik, A.; et al.: Using Learning Analytics to Predict At-Risk Students in Online Graduate Public Affairs and Administration Education. *The JPAE Messenger*. vol. 21, no. 2. 2015: pp. 247–262. ISSN 15236803.

[7] Baker, R. S.; Lindrum, D.; Lindrum, M. J.; et al.: Analyzing Early At-Risk Factors in Higher Education e-Learning Courses. *Students at Risk: Detection and Remediation*. 2015.

[8] Batista, G. E.; Bazzan, A. L.; Monard, M. C.: Balancing Training Data for Automated Annotation of Keywords: a Case Study. In *WOB*. 2003. pp. 10–18.

[9] Batista, G. E. A. P. A.; Prati, R. C.; Monard, M. C.: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.*. vol. 6, no. 1. June 2004: pp. 20–29. ISSN 1931-0145.
doi:10.1145/1007730.1007735.
Retrieved from: http://doi.acm.org/10.1145/1007730.1007735

[10] Bernstein, A.; Mannor, S.; Shimkin, N.: Online Classification with Specificity Constraints. In *Advances in Neural Information Processing Systems 23*, edited by

J. D. Lafferty; C. K. I. Williams; J. Shawe-Taylor; R. S. Zemel; A. Culotta. Curran Associates, Inc.. 2010. pp. 190–198.
Retrieved from: http://papers.nips.cc/paper/3896-online-classification-with-specificity-constraints.pdf

[11] Bonabeau, E.; Dorigo, M.; Theraulaz, G.: *Swarm intelligence: from natural to artificial systems*. New York, NY, USA: Oxford University Press, Inc.. 1999. ISBN 0-19-513159-2.

[12] Branco, P.; Torgo, L.; Ribeiro, R. P.: A Survey of Predictive Modeling on Imbalanced Domains. *ACM Comput. Surv.*. vol. 49, no. 2. August 2016: pp. 31:1–31:50. ISSN 0360-0300.

[13] Breiman, L.: Random Forests. *Machine Learning*. vol. 45, no. 1. October 2001: pp. 5–32. ISSN 0885-6125.

[14] Breiman, L.; Friedman, J. H.; Olshen, R. A.; et al.: Classification and Regression Trees. 1999.

[15] Cabrera, J. C. F.; Coello, C. A. C.: Handling constraints in particle swarm optimization using a small population size. In *Proc. Mexican Int'l. Conf. on Advances in artificial intelligence*. MICAI'07. Berlin, Heidelberg: Springer-Verlag. 2007. ISBN 3-540-76630-8, 978-3-540-76630-8. pp. 41–51.

[16] Cao, P.; Zhao, D.; Zaïane, O. R.: A PSO-Based Cost-Sensitive Neural Network for Imbalanced Data Classification. In *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2013 International Workshops*. Springer. Berlin, Heidelberg: Springer Berlin Heidelberg. 2013. ISBN 978-3-642-40319-4. pp. 452–463. doi:10.1007/978-3-642-40319-4_39.
Retrieved from: http://dx.doi.org/10.1007/978-3-642-40319-4_39

[17] Carvalho, D. R.; Freitas, A. A.: A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences*. vol. 163, no. 1. 2004: pp. 13–35.

[18] Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; et al.: SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Int. Res.*. vol. 16, no. 1. June 2002: pp. 321–357. ISSN 1076-9757.

[19] Chawla, N. V.; Japkowicz, N.; Kotcz, A.: Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*. vol. 6, no. 1. June 2004: pp. 1–6. ISSN 1931-0145.

[20] Chen, T.; Guestrin, C.: XGBoost: A Scalable Tree Boosting System. *CoRR*. vol. abs/1603.02754. 2016.
Retrieved from: http://arxiv.org/abs/1603.02754

[21] Costa, E.; Lorena, A.; Carvalho, A.; et al.: A review of performance evaluation measures for hierarchical classifiers. In *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop*. 2007. pp. 1–6.

[22] Davis, J.; Goadrich, M.: The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006. pp. 233–240.

[23] Dawson, S.; di Vimercati, S. D. C.; Samarati, P.: Specification and enforcement of classification and inference constraints. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE. 1999. pp. 181–195.

[24] Domingos, P.: MetaCost: a general method for making classifiers cost-sensitive. In *Proc. of the 5th ACM SIGKDD*. KDD '99. New York, NY, USA: ACM. 1999. ISBN 1-58113-143-7. pp. 155–164.

[25] Druck, G.; Mann, G.; McCallum, A.: Learning from Labeled Features Using Generalized Expectation Criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08. New York, NY, USA: ACM. 2008. ISBN 978-1-60558-164-4. pp. 595–602. doi:10.1145/1390334.1390436.
Retrieved from: http://doi.acm.org/10.1145/1390334.1390436

[26] Drummond, C.; Holte, R. C.: Cost curves: An improved method for visualizing classifier performance. *Machine Learning*. vol. 65, no. 1. Oct 2006: pp. 95–130. ISSN 1573-0565. doi:10.1007/s10994-006-8199-5.

[27] Ertekin, S.; Huang, J.; Bottou, L.; et al.: Learning on the Border: Active Learning in Imbalanced Data Classification. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. CIKM '07. New York, NY, USA: ACM. 2007. ISBN 978-1-59593-803-9. pp. 127–136. doi:10.1145/1321440.1321461.

[28] Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; et al.: LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*. vol. 9. June 2008: pp. 1871–1874. ISSN 1532-4435.
Retrieved from: http://dl.acm.org/citation.cfm?id=1390681.1442794

[29] Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers. Technical report. HP Laboratories. 2004.

[30] Fdez-Glez, J.; Ruano-Ordás, D.; Fdez-Riverola, F.; et al.: *Analyzing the Impact of Unbalanced Data on Web Spam Classification*. Cham: Springer International Publishing. 2015. pp. 243–250.

[31] Freund, Y.; Schapire, R. E.: Experiments with a New Boosting Algorithm. 1996.

[32] Friedman, J. H.: Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*. vol. 29. 2000: pp. 1189–1232.

[33] Ganganwar, V.: An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*. vol. 2, no. 4. 2012: pp. 42–47.

[34] Garcıa, V.; Mollineda, R.; Sánchez, J.: Theoretical analysis of a performance measure for imbalanced data. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR'10)*. 2010. pp. 617–620.

[35] García, V.; Mollineda, R. A.; Sánchez, J. S.: A New Performance Evaluation Method for Two-Class Imbalanced Problems. In *Structural, Syntactic, and*

Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg. 2008. ISBN 978-3-540-89689-0. pp. 917–925. doi:10.1007/978-3-540-89689-0_95.

[36] Gleich, D. F.: PageRank beyond the Web. *CoRR*. vol. abs/1407.5107. 2014.

[37] Grossi, V.; Romei, A.; Turini, F.: Survey on using constraints in data mining. *Data Mining and Knowledge Discovery*. 2016: pp. 1–41.

[38] Han, H.; Wang, W.-Y.; Mao, B.-H.: Borderline-SMOTE: A New Over-sampling Method in Imbalanced Data Sets Learning. In *Proceedings of the 2005 International Conference on Advances in Intelligent Computing - Volume Part I*. ICIC'05. Berlin, Heidelberg: Springer-Verlag. 2005. ISBN 3-540-28226-2, 978-3-540-28226-6. pp. 878–887. doi:10.1007/11538059_91.

[39] Han, J.; Kamber, M.; Pei, J.: *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science. 2011. ISBN 9780123814791.

[40] Hardie, B. G.; Fader, P. S.; Wisniewski, M.: An empirical comparison of new product trial forecasting models. *Journal of Forecasting*. vol. 17, no. 34. 1998: pp. 209–229.

[41] Hart, P.: The condensed nearest neighbor rule. *IEEE transactions on information theory*. vol. 14, no. 3. 1968: pp. 515–516.

[42] He, H.; Garcia, E. A.: Learning from Imbalanced Data. *IEEE Trans. on Knowl. and Data Eng.*. vol. 21, no. 9. September 2009: pp. 1263–1284. ISSN 1041-4347.

[43] He, J.; Bailey, J.; Rubinstein, B. I.; et al.: Identifying At-Risk Students in Massive Open Online Courses. In *AAAI*. 2015. pp. 1749–1755.

[44] Hlosta, M.; Stríž, R.; Kupčík, J.; et al.: Constrained Classification of Large Imbalanced Data by Logistic Regression and Genetic Algorithm. *International Journal of Machine Learning and Computing*. vol. 2013, no. 3. 2013: pp. 214–218. ISSN 2010-3700.

[45] Hlosta, M.; Stríž, R.; Zendulka, J.; et al.: PSO-based Constrained Imbalanced Data Classification. In *Proceedings of the Twelth International Conference on Informatics INFORMATICS'2013*. The University of Technology Košice. 2013. ISBN 978-80-8143-127-2. pp. 234–239.

[46] Hlosta, M.; Zdrahal, Z.; Zendulka, J.: Ouroboros: Early Identification of At-risk Students Without Models Based on Legacy Data. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*. LAK '17. New York, NY, USA: ACM. 2017. ISBN 978-1-4503-4870-6. pp. 6–15. doi:10.1145/3027385.3027449. Retrieved from: http://doi.acm.org/10.1145/3027385.3027449

[47] Ho, A. D.; Reich, J.; Nesterko, S. O.; et al.: HarvardX and MITx: the first year of open online courses, fall 2012-summer 2013. *Ho, AD, Reich, J., Nesterko, S., Seaton, DT, Mullaney, T., Waldo, J., & Chuang, I.(2014). HarvardX and MITx:*

The first year of open online courses (HarvardX and MITx Working Paper No. 1). 2014.

[48] Hu, X.; Eberhart, R.: Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *World Multiconference on Systemics, Cybernetics and Informatics SCI'02*. 2002. pp. 203–206.

[49] Japkowicz, N.; Stephen, S.: The Class Imbalance Problem: A Systematic Study. *Intell. Data Anal.*. vol. 6, no. 5. October 2002: pp. 429–449. ISSN 1088-467X.

[50] Jayaprakash, S. M.; Moody, E. W.; Lauria, E. J. M.; et al.: Early Alert of Academically At-Risk Students: An Open Source Analytics Initiative. *Journal of Learning Analytics*. vol. 1, no. 1. 2014: pp. 6–47. ISSN 1929-7750.

[51] Jeni, L. A.; Cohn, J. F.; De La Torre, F.: Facing Imbalanced Data–Recommendations for the Use of Performance Metrics. In *Proceedings of the 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*. ACII '13. 2013. ISBN 978-0-7695-5048-0. pp. 245–251. doi:10.1109/ACII.2013.47.
Retrieved from: http://dx.doi.org/10.1109/ACII.2013.47

[52] Jiang, S.; Warschauer, M.; Williams, A. E.; et al.: Predicting MOOC performance with week 1 behavior. In *Proceedings of the 7th International Conference on Educational Data Mining*. 2014. pp. 273–275.

[53] Jun-shan, T.; Wei, H.; Yan, Q.: Application of Genetic Algorithm in Data Mining. In *ETCS'09*, vol. 2. IEEE. 2009. pp. 353–356.

[54] Kazerouni, A. M.; Edwards, S. H.; Shaffer, C. A.: Quantifying Incremental Development Practices and Their Relationship to Procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. New York, NY, USA: ACM. 2017. ISBN 978-1-4503-4968-0. pp. 191–199. doi:10.1145/3105726.3106180.
Retrieved from: http://doi.acm.org/10.1145/3105726.3106180

[55] King, G.; Zeng, L.: Logistic regression in rare events data. *Political analysis*. vol. 9, no. 2. 2001: pp. 137–163.

[56] Kloft, M.; Stiehler, F.; Zheng, Z.; et al.: 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*. 2014. pp. 60–65.

[57] Konig, C. J.; Kleinmann, M.: Deadline Rush: A Time Management Phenomenon and Its Mathematical Description Relationships Between Critical Thinking and Attitudes Toward Women's Roles in Society. *The Journal of psychology*. vol. 139, no. 1. 2005: pp. 33–45.

[58] Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*. vol. 5, no. 4. 2016: pp. 221–232. doi:10.1007/s13748-016-0094-0.

[59] Kubat, M.; Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann. 1997. pp. 179–186.

[60] Kukar, M.; Kononenko, I.; et al.: Cost-Sensitive Learning with Neural Networks. In *ECAI*. 1998. pp. 445–449.

[61] Kuzilek, J.; Hlosta, M.; Zdrahal, Z.: Open University Learning Analytics Dataset. In *Data literacy for Learning Analytics workshop at LAK16*. 2016.

[62] Lange, T.; Law, M. H. C.; Jain, A. K.; et al.: Learning with Constrained and Unlabelled Data. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. Washington, DC, USA: IEEE Computer Society. 2005. ISBN 0-7695-2372-2. pp. 731–738. doi:10.1109/CVPR.2005.210.
Retrieved from: http://dx.doi.org/10.1109/CVPR.2005.210

[63] Laurikkala, J.: *Improving Identification of Difficult Small Classes by Balancing Class Distribution*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2001. ISBN 978-3-540-48229-1. pp. 63–66. doi:10.1007/3-540-48229-6_9.

[64] Liu, X.-Y.; Wu, J.; Zhou, Z.-H.: Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. vol. 39, no. 2. 2009: pp. 539–550.

[65] Lohmann, G.; Margulies, D. S.; Horstmann, A.; et al.: Eigenvector centrality mapping for analyzing connectivity patterns in fMRI data of the human brain. *PloS one*. vol. 5, no. 4. 2010: page e10232.

[66] López, V.; Fernández, A.; García, S.; et al.: An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*. vol. 250. 2013: pp. 113–141.

[67] Maloof, M. A.: Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*. 2003. page 8.

[68] Mani, I.; Zhang, I.: kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*. 2003. page 7.

[69] Manning, C. D.; Schütze, H.: *Foundations of statistical natural language processing*. vol. 999. MIT Press. 1999.

[70] Márquez-Vera, C.; Cano, A.; Romero, C.; et al.: Early dropout prediction using data mining: a case study with high school students. *Expert Systems*. vol. 33, no. 1. 2016: pp. 107–124. doi:10.1111/exsy.12135.

[71] Morrison, J. L.; Breitling, R.; Higham, D. J.; et al.: GeneRank: Using search engine technology for the analysis of microarray experiments. *BMC Bioinformatics*. vol. 6. 2005: page 233. doi:10.1186/1471-2105-6-233.

[72] Murphy, K. P.: *Machine Learning: A Probabilistic Perspective*. The MIT Press. 2012. ISBN 0262018020, 9780262018029.

[73] Naisbitt, J.: *Megatrends : ten new directions transforming our lives*. Warner Books, New York :. 1982. ISBN 0446512516. 290 p. ; pp.

[74] Noda, E.; Freitas, A. A.; Lopes, H. S.: Discovering interesting prediction rules with a genetic algorithm. In *Evolutionary Computation - CEC 99*, vol. 2. IEEE. 1999. pp. 1322–1329.

[75] Page, L.; Brin, S.; Motwani, R.; et al.: The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66. Stanford InfoLab. November 1999.

[76] Parsopoulos, K. E.; Vrahatis, M. N.: Particle Swarm Optimization Method for Constrained Optimization Problems. In *In Proc. of the Euro-Int'l. Symposium on Computational Intelligence*. IOS Press. 2002. pp. 214–220.

[77] Pedregosa, F.; Varoquaux, G.; Gramfort, e.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. vol. 12. 2011: pp. 2825–2830.

[78] Prati, R. C.; Batista, G. E. A. P. A.; Silva, D. F.: Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems*. vol. 45, no. 1. 2015: pp. 247–270. ISSN 0219-3116. doi:10.1007/s10115-014-0794-3.

[79] Provost, F.: Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 workshop on imbalanced data sets*. 2000. pp. 1–3.

[80] Provost, F.; Fawcett, T.; Kohavi, R.: The Case Against Accuracy Estimation for Comparing Induction Algorithms. In *In Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann. 1997. pp. 445–453.

[81] Quinlan, J. R.: Discovering Rules by Induction from Large Collections of Examples. In *Expert Systems in the Micro-Electronic Age*, edited by D. Michie. Edinburgh: Edinburgh University Press. 1979. pp. 168–201.

[82] Quinlan, J. R.: Induction of Decision Trees. *Mach. Learn.*. vol. 1, no. 1. March 1986: pp. 81–106. ISSN 0885-6125. doi:10.1023/A:1022643204877.
Retrieved from: http://dx.doi.org/10.1023/A:1022643204877

[83] Quinlan, J. R.: *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.. 1993. ISBN 1-55860-238-0.

[84] Quinn, J.: Drop-out and Completion in Higher Education in Europe among students from under-represented groups.

[85] Ribeiro, M. T.; Singh, S.; Guestrin, C.: „Why Should I Trust You?“: Explaining the Predictions of Any Classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: ACM. 2016. ISBN 978-1-4503-4232-2. pp. 1135–1144. doi:10.1145/2939672.2939778.
Retrieved from: http://doi.acm.org/10.1145/2939672.2939778

[86] Salatino, A. A.; Motta, E.: Detection of Embryonic Research Topics by Analysing Semantic Topic Networks. In *SAVE-SD 2016*. April 2016. page 15.

[87] Sanguanmak, Y.; Hanskunatai, A.: DBSM: The combination of DBSCAN and SMOTE for imbalanced data classification. In *Computer Science and Software Engineering (JCSSE), 2016 13th International Joint Conference on*. IEEE. 2016. pp. 1–5.

[88] Shabtai, A.; Moskovitch, R.; Feher, C.; et al.: Detecting unknown malicious code by applying classification techniques on OpCode patterns. *Security Informatics*. vol. 1, no. 1. 2012: page 1. ISSN 2190-8532. doi:10.1186/2190-8532-1-1.

[89] Simpson, O.: 22% - can we do better? In *The CWP Retention Literature Review*. 2010. page 47.

[90] Sokolova, M.; Lapalme, G.: A systematic analysis of performance measures for classification tasks. *Information Processing & Management*. vol. 45, no. 4. 2009: pp. 427 – 437. ISSN 0306-4573. doi:http://dx.doi.org/10.1016/j.ipm.2009.03.002.

[91] Stanescu, A.; Caragea, D.: Semi-supervised self-training approaches for imbalanced splice site datasets. In *Proceedings of The Sixth International Conference on Bioinformatics and Computational Biology, BICoB 2014*. 2014. pp. 131–136.

[92] Sun, Y.; Kamel, M. S.; Wong, A. K. C.; et al.: Cost-sensitive boosting for classification of imbalanced data. *Pattern Recogn.*. vol. 40, no. 12. December 2007: pp. 3358–3378. ISSN 0031-3203.

[93] Tan, M.; Tan, L.; Dara, S.; et al.: Online Defect Prediction for Imbalanced Data. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2*. ICSE '15. Piscataway, NJ, USA: IEEE Press. 2015. pp. 99–108.
Retrieved from: http://dl.acm.org/citation.cfm?id=2819009.2819026

[94] Taylor, C.; Veeramachaneni, K.; O'Reilly, U.: Likely to stop? Predicting Stopout in Massive Open Online Courses. *CoRR*. vol. abs/1408.3382. 2014.

[95] Thai-Nghe, N.; Busche, A.; Schmidt-Thieme, L.: Improving Academic Performance Prediction by Dealing with Class Imbalance. In *Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, Pisa, Italy , November 30-December 2, 2009*. 2009. pp. 878–883. doi:10.1109/ISDA.2009.15.

[96] Vanderlooy, S.; Sprinkhuizen-Kuyper, I. G.; Smirnov, E. N.; et al.: The ROC Isometrics Approach to Construct Reliable Classifiers. *Intell. Data Anal.*. vol. 13, no. 1. January 2009: pp. 3–37. ISSN 1088-467X.

[97] Vinterbo, S.; Ohno-Machado, L.: A genetic algorithm to select variables in logistic regression: example in the domain of myocardial infarction. In *Proceedings of the AMIA Symposium*. American Medical Informatics Association. 1999. page 984.

[98] Vossensteyn, H.; Kottmann, A.; Jongbloed, B.; et al.: Drop-out and Completion in Higher Education in Europe.

[99] Wang, S.; Minku, L. L.; Yao, X.: Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*. vol. 27, no. 5. 2015: pp. 1356–1368.

[100] Wang, Z.; Sun, X.; Zhang, D.: A PSO-Based Classification Rule Mining Algorithm. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Third Int'l Conf. on Intelligent Computing, ICIC 2007.*, *Lecture Notes in Computer Science*, vol. 4682. Springer. 2007. ISBN 978-3-540-74201-2. pp. 377–384.

[101] Weiss, G. M.: Mining with rarity: a unifying framework. *SIGKDD Explorations*. vol. 6, no. 1. 2004: pp. 7–19.

[102] Weiss, G. M.; Tian, Y.: Maximizing classifier utility when there are data acquisition and modeling costs. *Data Mining and Knowledge Discovery*. vol. 17, no. 2. 2008: pp. 253–282. ISSN 1573-756X.

[103] Wikipedia: Gene expression programming — Wikipedia, The Free Encyclopedia. 2017. [Online; accessed 13-August-2017].
Retrieved from: https://en.wikipedia.org/w/index.php?title=Gene_expression_programming&oldid=776311480

[104] Wilson, D. L.: Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*. vol. 2, no. 3. 1972: pp. 408–421.

[105] Wolff, A.; Zdrahal, Z.; Herrmannova, D.; et al.: Developing predictive models for early detection of at-risk students on distance learning modules. In *Machine Learning and Learning Analytics workshop at LAK14, 24-28 March 2014, Indianapolis, Indiana, USA*. 2014. page 4.

[106] Yan, W.; Goebel, K. F.: Designing classifier ensembles with constrained performance requirements. In *Defense and Security*. International Society for Optics and Photonics. 2004. pp. 59–68.

[107] Yang, J.; Honavar, V.: Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*. Springer. 1998. pp. 117–136.

[108] Yang, J.-M.; Chen, Y.-P.; Horng, J.-T.; et al.: Applying Family Competition to Evolution Strategies for Constrained Optimization. In *Proc. of the 6th Int'l. Conf. on Evolutionary Programming VI*. EP '97. London, UK, UK: Springer-Verlag. 1997. ISBN 3-540-62788-X. pp. 201–211.

[109] Ye, C.; Biswas, G.: Early Prediction of Student Dropout and Performance in MOOCs using Higher Granularity Temporal Information. *Journal of Learning Analytics*. vol. 1, no. 3. 2014: pp. 169–172.

[110] Yuxin, D.; Siyi, Z.: Malware detection based on deep learning algorithm. *Neural Computing and Applications*. Jul 2017. ISSN 1433-3058. doi:10.1007/s00521-017-3077-6.
Retrieved from: https://doi.org/10.1007/s00521-017-3077-6

[111] Zadrozny, B.; Langford, J.; Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE. 2003. pp. 435–442.

[112] Zhu, X.: Semi-Supervised Learning Literature Survey. Technical Report 1530. Computer Sciences, University of Wisconsin-Madison. 2005.
Retrieved from: http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

Appendices

Appendix A

List of Publications and Products

The existing work is divided by the type of publication and its relationship to the PhD thesis.

A.1 Publications important for the PhD thesis

A.1.1 Journals

- *HLOSTA Martin*, STRÍŽ Rostislav, KUPČÍK Jan, ZENDULKA Jaroslav and HRUŠKA Tomáš. Constrained Classification of Large Imbalanced Data by Logistic Regression and Genetic Algorithm. International Journal of Machine Learning and Computing. Singapore: of Computer Science and Information Technology Press, 2013, vol. 2013, no. 3, pp. 214-218. ISSN 2010-3700.

A.1.2 Conference proceedings

- *HLOSTA Martin*, STRÍŽ Rostislav, ZENDULKA Jaroslav and HRUŠKA Tomáš. PSO-based Constrained Imbalanced Data Classification. In: Proceedings of the Twelth International Conference on Informatics INFORMATICS'2013. Spišská Nová Ves: The University of Technology Košice, 2013, pp. 234-239. ISBN 978-80-8143-127-2.

- *HLOSTA Martin*, ZDRÁHAL Zdeněk and ZENDULKA Jaroslav. Ouroboros: Early identification of at-risk students without models based on legacy data. In: LAK '17 Proceedings of the Seventh International Learning Analytics & Knowledge Conference. Vancouver: Association for Computing Machinery, 2017, pp. 6-15. ISBN 978-1-4503-4870-6.

A.1.3 Conference workshops

- KUŽÍLEK Jakub, *HLOSTA Martin*, HERRMANNOVÁ Drahomíra, ZDRÁHAL Zdeněk. Open University Learning Analytics Dataset. In Data literacy for Learning Analytics workshop at LAK16, 25–29th April 2016, Edinburgh, UK. page 9.

A.2 Other Publications

A.2.1 Journals

- ŠEBEK Michal, *HLOSTA Martin*, KUPČÍK Jan, ZENDULKA Jaroslav and HRUŠKA Tomáš. Multi-level Sequence Mining Based on GSP. Acta Electrotechnica et Informatica. 2012, vol. 2012, no. 2, pp. 31-38. ISSN 1335-8243.

- KUŽÍLEK Jakub, *HLOSTA Martin*, HERRMANNOVÁ Drahomíra, ZDRÁHAL Zdeněk. OU Analyse: Analysing at-risk students at The Open University. Learning Analytics Review. vol. LAK15-1. March 2015: pp. 1–16.

A.2.2 Conference proceedings

- HERODOTOU Christothea, RIENTIES Bart, ZDRÁHAL Zdeněk, *HLOSTA Martin*, BOROOWA Avinash and NAYDENOVA Galina: Implementing Predictive Learning Analytics on a Large Scale: The Teacher's Perspective. In: LAK '17 Proceedings of the Seventh International Learning Analytics & Knowledge Conference. Vancouver: Association for Computing Machinery, 2017, ISBN 978-1-4503-4870-6.

- HUPTYCH Michal, BOHUSLÁVEK Michal, *HLOSTA Martin*, ZDRÁHAL Zdeněk: Measures for recommendations based on past students' activity.In: LAK '17 Proceedings of the Seventh International Learning Analytics & Knowledge Conference. Vancouver: Association for Computing Machinery, 2017, ISBN 978-1-4503-4870-6.

- *HLOSTA Martin*, ŠEBEK Michal and ZENDULKA Jaroslav. Approach to Visualisation of Evolving Association Rule Models. In: Proceedings of The Second International Conference on Informatics & Applications (ICIA 2013). Łódź: The Society of Digital Information and Wireless Communications, 2013, pp. 47-52. ISBN 978-1-4673-5255-0.

- ŠEBEK Michal, *HLOSTA Martin*, ZENDULKA Jaroslav and HRUŠKA Tomáš. MLSP: Mining Hierarchically-Closed Multi-Level Sequential Patterns. In: 9th International Conference, ADMA 2013. Hangzhou: Springer Verlag, 2013, pp. 157-168. ISBN 978-3-642-53913-8.

- ŠEBEK Michal, *HLOSTA Martin* and ZENDULKA Jaroslav. Knowledge Discovery in Data with FIT-Miner. In: Znalosti 2011: Sborník příspěvků 10. ročníku konference. Stará Lesná: VŠB-Technical University of Ostrava, 2011, pp. 182-193. ISBN 978-80-248-2369-0.

- ŠEBEK Michal, *HLOSTA Martin*, KUPČÍK Jan, ZENDULKA Jaroslav and HRUŠKA Tomáš. Multi-level Sequence Mining Based on GSP. In: Proceedings of the Eleventh International Conference on Informatics INFORMATICS'2011. Košice: Faculty of Electrical Engineering and Informatics, University of Technology Košice, 2011, pp. 185-190. ISBN 978-80-89284-94-8.

- HERRMANNOVÁ Drahomíra, *HLOSTA Martin*, KUŽÍLEK Jakub, ZDRÁHAL Zdeněk. Evaluating Weekly Predictions of At-Risk Students at The Open University: Results and Issues. In EDEN 2015 Annual Conference Expanding Learning Scenarios: Opening Out the Educational Landscape. June 2015.

- FOURNIER-VIGER Philippe, GOMARIZ Antonio, ŠEBEK Michal and *HLOSTA Martin*. VGEN: Fast Vertical Mining of Sequential Generator Patterns. In: Data Warehousing and Knowledge Discovery. Munich: Springer Verlag, 2014, pp. 476-488. ISBN 978-3-319-10159-0.

A.2.3 Conference workshops

- ZDRÁHAL Zdeněk, *HLOSTA Martin*, KUŽÍLEK Jakub. Analysing performace of first year engineering students. In Data literacy for Learning Analytics workshop at LAK16, 25–29th April 2016, Edinburgh, UK.
- WOLFF Annika, ZDRÁHAL Zdeněk, HERRMANNOVÁ Drahomíra, KUŽÍLEK Jakub, *HLOSTA Martin*. Developing predictive models for early detection of at-risk students on distance learning modules. In Machine Learning and Learning Analytics workshop at LAK14, 24-28 March 2014, Indianapolis, Indiana, USA. 2014.
- *HLOSTA Martin*, HERRMANNOVÁ Drahomíra, VACHOVA Lucie, KUŽÍLEK Jakub, ZDRÁHAL Zdeněk, WOLFF Annika. Modelling student online behaviour in a virtual learning environment. In Machine Learning and Learning Analytics workshop at LAK14, 24-28 March 2014, Indianapolis, Indiana, USA. 2014.

A.2.4 In review

- HERODOTOU Christothea, RIENTIES Bart, BOROOWA Avinash, ZDRÁHAL Zdeněk, *HLOSTA Martin*. Using Predictive Learning Analytics to Support Just-in-time Interventions: The Teachers' Perspectives across a large-scale implementation. Submitted in Internet and Higher Education.

A.2.5 Other

- *HLOSTA Martin.*: Cluster Analysis Module of Data Mining System, In Proceedings of the 16th Conference Student EEICT 2010: Volume 2, Brno, FIT VUT,Brno, 2010

A.3 Products

- Malware Analysis System, software 2013 – the result of the project for TACR, more information available here:
  URL: http://www.fit.vutbr.cz/research/grants/AVGMAS/index.html
- Ouroboros classification, 2017 – Python package and experiments, with the results supporting the paper accepted for LAK 2017 conference, available on GitHub:
  URL: https://github.com/hlostam/ouroboros_paper

A.4 Project participation

- Innovation of practical teaching in course Knowledge Discovery in Databases, FRVŠ MŠMT, FR882/2013/G1, 2013, completed
- Advanced recognition and presentation of multimedia data, BUT, FIT-S-11-2, 2011-2013, completed

- Improving Security of the Internet by Using System for Analyzing of Malicious Code Spreading, TACR, TA01010858, 2011-2013, completed
- OUAnalyse & Early Alert Indicators: Early identification of students at risk of failing, external project at The Open University, Knowledge Media institute. 2014 -, running

Appendix B

Supporting material

This appendix contains the supporting material for the Self-Learning approach.

B.1 OULAD Schema

Table studentInfo

This table contains student demographic information and also their results in each module they studied. The module denotes in the naming convention for a course. It consists of 32,593 rows with the following columns:

- *code_module* - module identification code on which the student is registered.
- *code_presentation* - presentation identification code during which the student is registered on the module.
- *id_student* - the unique student identification number.
- gender - student's gender.
- region - the geographic region, where the student lived while taking the module-presentation.
- highest_education - the highest student education level upon entry to the module presentation.
- imd_band - the IMD band of the place where the student lived during the module-presentation.
- age_band - a band of student's age.
- num_of_prev_attempts - the number of how many times the student has attempted this module.
- studied_credits - the total number of credits for the modules the student is currently studying.
- disability - indicates whether the student has declared a disability.
- final_result - student's final result in the module-presentation.

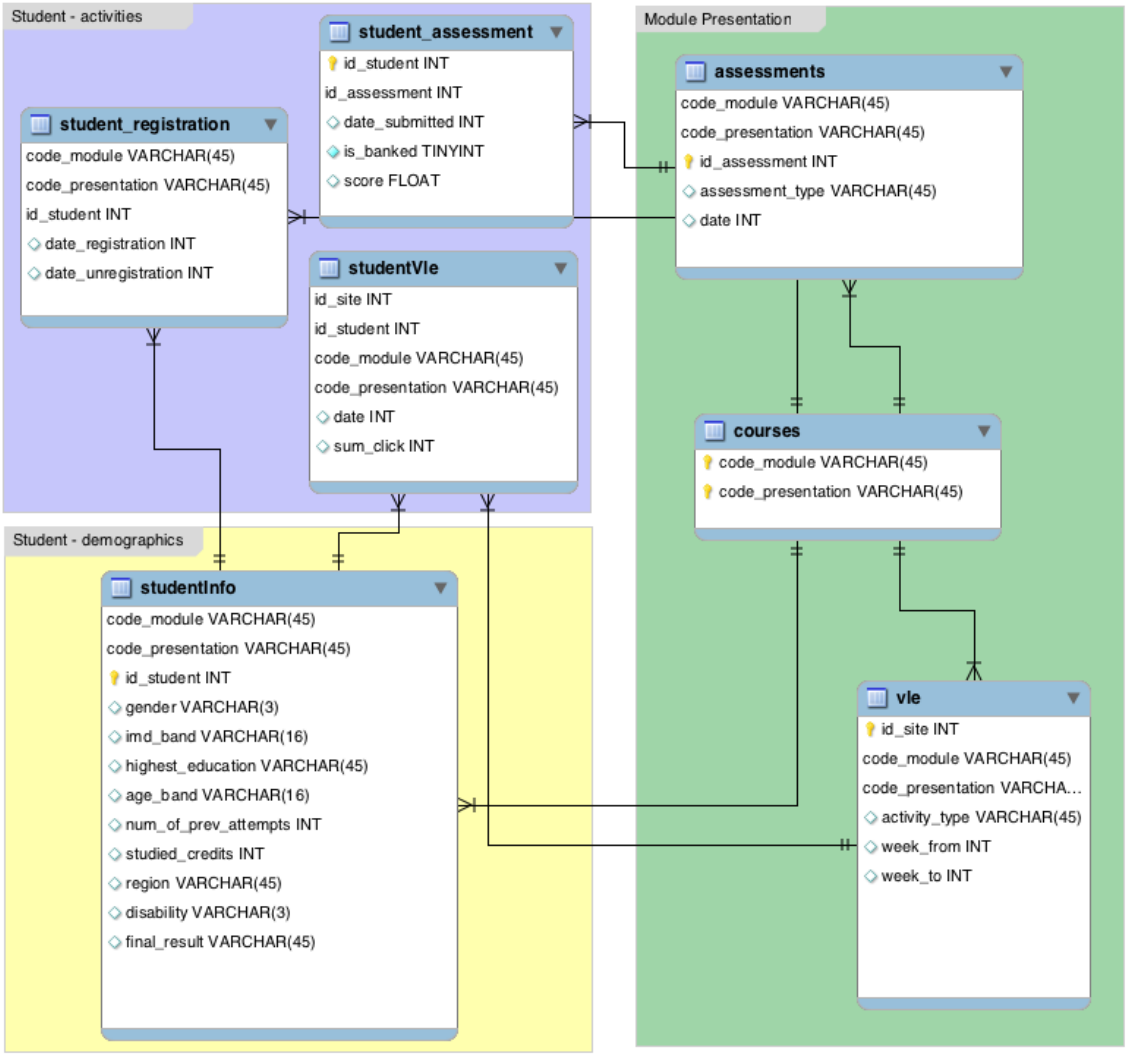

Figure B.1: Schema of the OULAD dataset

Table courses

The table contains the list of all available modules and their presentations. It consists of 22 rows with the following columns:

- *code_module* - code name of the module, which serves as the identifier.
- *code_presentation* - code name of the presentation.
- length - the length of the module-presentation in days from module start date to module end date.

The structure of B and J presentations may differ and therefore it is recommended to analyse the B and J presentations separately. .

Table studentRegistration

Contains information about the time when the student registered for the module presentation. For students who unregistered, the date of un-registration is also recorded. It consists of 32,593 rows with the following columns:

- *code_module* - the module identification code.
- *code_presentation* - the presentation identification code.
- *id_student* - the unique student identification number.
- date_registration - the day of student's registration for the module presentation.
- date_unregistration - the day of student unregistration from the module presentation. Students, who completed the course have this field empty. Students who unregistered have Withdrawal as the value of the final_result in the studentInfo table.

Table assessments

This table contains information about assessments in module-presentations. Usually, every presentation has a number of assessments followed by the final exam. The table consists of 206 rows with the following columns:

- *code_module* - module identification code, to which the assessment belongs.
- *code_presentation* - presentation identification code, to which the assessment belongs.
- *id_assessment* - assessment identification number.
- assessment_type - a type of assessment. Three types of assessments exist - Tutor Marked Assessment (TMA), Computer Marked Assessment (CMA) and Final Exam (Exam).
- date - information about the cut-off day of the assessment.
- weight - the weight of the assessment. Typically, Exams are treated separately and have the weight equal to 100%; the sum of all other assessments is also 100%.

If the information about the final exam cut-off day is missing, it takes place during the last week of the module-presentation.

Table studentAssessment

The table contains the results of students' assessments. If the student does not submit the assessment, no result is recorded. Results of the final exam are usually missing (since they are scored and used for the final marking immediately at the end of the module). It consists of 173,912 rows with the following columns:

- *id_assessment* - the assessment identification number.
- *id_student* - the unique student identification number.
- date_submitted - the day of assessment submission.

- is_banked - the status flag indicating that the assessment result has been transferred from a previous presentation.
- score - the student's score in this assessment. The range is from 0 to 100. A score lower than 40 is interpreted as Fail. The marks are in the range from 0 to 100.

Table studentVle

The studentVle table contains information about student's interactions with the VLE. It consists of 10,655,280 rows with the following columns:

- *code_module* - the module identification code.
- *code_presentation* - the presentation identification code.
- *id_student* - the unique student identification number.
- *id_site* - the VLE material identification number.
- date - the day of student's interaction with the material.
- sum_click - the number of times the student interacted with the material.

Table vle

The vle table contains information about the materials available in the VLE. Typically these are HTML pages, pdf files, etc. It consists of 6,364 rows with the following columns:

- *id_site* - the identification number of the material.
- *code_module* - the identification code for the module.
- *code_presentation* - the identification code of the presentation.
- activity_type - the role associated with the module material.
- week_from - the week from which the material is planned to be used.
- week_to - the week until which the material is planned to be used.