

Python – Learning Analytics

Published on

Work in progress version

Erasmus+ FITPED-AI

Future IT Professionals Education in Artificial Intelligence

Project 2021-1-SK01-KA220-HED-000032095



The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Licence (licence type: Attribution-Non-commercial-No Derivative Works) and may be used by third parties as long as licensing conditions are observed. Any materials published under the terms of a CC Licence are clearly identified as such.

All trademarks and brand names mentioned in this publication and all trademarks and brand names mentioned that may be the intellectual property of third parties are unconditionally subject to the provisions contained within the relevant law governing trademarks and other related signs. The mere mention of a trademark or brand name does not imply that such a trademark or brand name is not protected by the rights of third parties.

© 2023 Constantine the Philosopher University in Nitra

TABLE OF CONTENTS

1 Introduction.....	6
1.1 Definition	7
1.2 Objectives and related topics.....	9
1.3 Stakeholders	12
1.4 Challenges	13
2 Learning Analytics	17
2.1 LA basic elements.....	18
2.2 Learning analytics data	21
3 Data resources and types.....	23
3.1 Data	24
3.2 Big data in education	27
3.3 Data types.....	32
4 Methodological Frameworks for Learning Analytics Methods	37
4.1 Data science process	38
4.2 Methodologies	40
5 Exploratory Data Analysis Methods.....	45
5.1 EDA automation	46
5.2 Data analysis methods.....	49
6 Data Preprocessing Methods.....	76
6.1 Methods.....	77
6.2 Data transformation and reduction	80
7 Linear Regression	85
7.1 Problem understanding	86
7.2 Linear regression models	95
7.3 Find the best model.....	100
7.4 Prediction by Scikit-learn.....	105
8 Random Forest	108
8.1 Multiple Classifier Random Forest.....	109
8.2 Modeling using Random Forest	114
8.3 Hyperparameters Tuning.....	117
9 Clustering Using K-means	130
9.1 Cluster analysis.....	131
9.2 Optimal number of clusters	138
10 Association Rule Mining	147
10.1 Association Rule Mining.....	148

11 Learning Analytics Research Topics	159
11.1 Research topics	160
12 Development EdTech with AI/ML/DL Model.....	167
12.1 LA architecture	168
12.2 Open LA architecture.....	173
12.3 Software development.....	176
12.4 Project management.....	179

Introduction

Chapter **1**

1.1 Definition

1.1.1

The effort to effectively analyse the data associated with the learning process has been in the focus for many years, while the individual periods and disciplines that covered this issue have always been closely linked to the tools and approaches that maturity of the time brought. Currently, we are faced with the term Learning Analytics (LA), which represents a research discipline that deals with collecting, measuring and analysing available data on different types of users in order to understand and optimize not only the learning process, but also the entire environment in which the learning process takes place.

In the decision-making process, LA uses various techniques and methods based on artificial intelligence (AI) and machine learning (ML), statistics, data visualization, multimodal analysis, or mixed quantitative and qualitative methods.

1.1.2

Analytics generally involves the use of data and quantitative analysis in the decision-making process. Behind the increase in its popularity, and especially its importance, we must see the development of computational methods, the enormous increase in the volume of data as well as the availability of computational resources. It is therefore logical that interest in deploying analytical approaches in the educational context has increased significantly in the last decade.

We can list three factors that may have led to recent popularity of analytics. The first factor is the volumes of data collected in educational institutions, which have increased significantly, whether from learning management systems or from student and academic information systems. The second factor is the actual use of various forms of online and blended learning, which have brought with them several problems with learning itself, such as a lack of motivation of students, difficulties for educators to obtain direct feedback on the level of interest of students or to verify the degree of understanding of the submitted content by students. Finally, the third factor was the fear of loss of competitiveness, as several countries are increasingly drawing attention to and interested in improving the development of higher education, offering better learning opportunities leading to better employability in the labour market.

1.1.3

According to Bienkowski, LA emphasizes data collection and analysis, as well as the processes that companies engaged in various forms of education continue to implement, while LA also seeks to understand the system as a whole and support the decision-making process. Unlike related disciplines, it does not deal solely with the development of new methods and techniques of data analysis, but instead deals with the application of known methods and models in order to answer important

questions that affect the systems and organization of education in these societies (Bienkowski, Feng and Means 2012).

1.1.4

Bichsel (Bichsel 2012) subsequently defines LA as the use of data, statistical analysis, exploratory and predictive models to gain knowledge and research complex issues related to the stakeholders of the educational process, the educational environment, and the support of decision-making processes necessary for their management.

1.1.5

The authors (Pena-Ayala, Cardenas-Robledo, and Sossa 2018) state that LA focuses on creating systems capable of customizing content, support levels, and other personalized services through data collection, processing, and analysis in a way that minimizes delays between data collection and actual use. At the same time, the authors provide their own definition, in which LA represents a research line that seeks to study, understand, describe, explain and predict the learning process, both from an experimental and behavioural point of view, that happens with the help of information systems and applications in order to increase teaching effectiveness, the resulting skills and the satisfaction of user stakeholders.

1.1.6

However, there are other less formal definitions of LA. For example, Duval (Duval 2012) states that LA's goal is to collect the digital footprints students leave behind and effectively use those clues to improve the learning process, decision-making process, as well as the entire environment in which the process takes place.

1.1.7

Another definition sees LA as the process of identifying problems and applying statistical models and analysis to solve them using simulated or real data. Barneveld et al. (Barneveld, Arnold, and Campbell 2012) propose that LA should be considered as a more general concept that covers several more narrowly defined areas of research with its themes. They define LA as the use of analytical techniques that make targeted use of educational, curricular, and support resources to support decision-making and achieve specific educational and management goals.

1.1.8

Finally, the JISC initiative for the LA domain in the UK, defines LA as the process of identifying problems, describing them, and then applying statistical models and analyses not only to solve them, but also subsequently generalizing the findings to a whole set of similar problems in the future (Jordaan and Van der Merwe 2015).

The preceding definitions define LA in general. A closer look at the definition of LA from the perspective of identifying an appropriate approach to effective adaptation or implementation of LA benefits in education societies is characterized by several other definitions.

1.1.9

Adejo and Connolly (Adejo and Connolly 2017) narrowed down the definition of LA to its use in the context of universities and virtual learning environments. The authors understand LA as collecting, storing and analysing data from learning management systems and VLE in order to gain various useful insights into users, their behaviour and preferences, and thus support decision-making that will have a lasting impact on different types of users and the organization itself. In other words, LA's main intent is to collect data for the development of models, algorithms, and processes that can be further used and generalized to improve the performance of all parties involved in the learning process.

1.1.10

Similarly, Siemens and Baker (Siemens and Baker 2012) define LA as the discipline that deals with the measurement, collection, analysis, and reporting of data generated during the activities of different groups of people in the framework of their work in higher education institutions. According to the authors, the goal is to understand, optimize or improve the learning process and support decision-making at different levels of management. At the same time, LA uses the same approaches to optimize the entire learning environment. The authors add that LA focuses on evaluating user behaviour in the context of the learning process, analysing it, and then interpreting it to understand it more deeply with the help of new models of learning, learning, effective organizational management, and decision support.

1.1.11

Johnson et al. add that LA represents a research area that uses data analytics to support decision-making at all levels of educational organization (Johnson, Adams Becker, Cummins, Estrada, Freeman, and Ludgate 2013). It has been confirmed that LA can be beneficial for the management of the organization in terms of better allocation of organizational resources, higher number, and quality of knowledge of graduating graduates as well as more efficient spending of invested funds (Bichsel 2012).

1.2 Objectives and related topics

1.2.1

Learning analytics objectives are very diverse, as they are based on the requirements and expectations that individual stakeholders in the educational process must meet. Unlike previous approaches focused on education data, LA emphasizes achieving a

state that directly results in added value for any of the stakeholders. Therefore, it is most often monitoring and analysis of the course of the learning process, prediction of results and progress in study, effective feedback, and intervention, or even personalization and adaptation of the educational environment. These processes in any form are inherently very complex, fully unexplored and provide ample space for research.

More specifically, we can list the most often mentioned objectives of learning analytics (Papamitsiou & Economides, 2014):

- monitor learners' progress,
- model learners/learners' behaviour,
- detect affects/emotions of learners,
- predict learning performance/dropout/retention,
- generate feedback,
- provide recommendations,
- guide adaptation,
- increase self-reflection/ self-awareness,
- facilitate self-regulation.

Therefore, the main aim of this course is to introduce the student with this promising research discipline, determine learning analytics scope, show examples of methods, and point out its contribution to better understanding the stakeholders, who participate in the contemporary educational workspace.

1.2.2

At the core of LA are methods that use educational data to support the learning process, taking into account that there are many types of educational data with different characteristics such as their distribution, scope, size, and degree of privacy. In addition, LA is closely related to other research areas, such as Educational Data Mining (EDM), which focuses on data collection and the development of specific knowledge discovery methods for education, or Academic Analytics (AA), which focuses on applying business intelligence practices to analyse academic data to support institutional decision-making.

1.2.3

Educational Data Mining

Educational Data Mining (EDM) is a relatively new scientific field, which arose as a natural consequence of the spread of the Internet, modern statistical methods, and e-learning itself. Since different virtual learning environments store a lot of data about their users, their work and habits, the idea of using advanced statistical methods seems completely natural.

EDM is then characterized as a research area that deals with the development of new techniques and methods, testing new theories of learning, searching for patterns of user behaviour in unstructured and structured data created by the interaction of different types of users in a certain virtual learning environment, educational software, intelligent or adaptive learning system, or specialized testing software (Romero, Ventura & Pechenizkiy, 2010). A more recent definition considers EDM to be an example of a scientific discipline that deals with developing methods for exploring unique types of data that come from a learning environment (Bakhshinategh, Zaiane, ElAtia and Ipperciel 2018) and using these methods to better understand the behaviour of all user stakeholders, related processes, and features of the environment in which learning takes place (Ferguson 2016).

1.2.4

The main objective of EDM is to analyse data coming from virtual learning environments and find answers to problem questions about the learning process as well as all e-learning. EDM focuses on the development of methods that will allow a better understanding of student behaviour and the learning process itself using virtual learning environments or using any educational software.

EDM as a field has laid its foundation on research from other research areas, such as educational technology, pedagogy, psychology, e-learning and web log mining. At the same time, we must mention that the emergence and development of EDM methods is equally significantly influenced by dynamic developments in the field of the Internet, systems for learning management, intelligent education systems as well as in the field of adaptive educational hypermedia systems (Skalka et al., 2013).

1.2.5

Academic Analytics

Academic analytics focuses on improving organizational processes, allocating resources and tools for measurement, comparing institutions using data about students, teachers and institutions. Academic analytics can therefore be understood as the application of business intelligence methods in an academic environment that highlights analyses at institutional, regional, and international levels.

Academic analytics is an area that combines big datasets with statistical techniques and predictive modelling to improve decision-making, while helping institutions manage student success and accountability for their education. In the same work, the authors also develop the benefits of an analytical, data- and fact-based approach to support decision-making in a higher education institution instead of decision-making based purely on intuition or accumulated experience. In particular, data mining is presented as a promising alternative to extracting knowledge from large amounts of data. The potential of this area can be seen for all stakeholders. Thus, academic analytics mainly concerns higher levels of education and adult learning, while learning analytics, which can currently be considered the overarching and most general name for the field exploring educational data, focuses more on transferring

the benefits of advanced analysis to learning and its management in any learning environment.

1.3 Stakeholders

1.3.1

Learning analytics combines a wide range of techniques used to collect, store, visualize and report data used for administrative or pedagogical purposes. Whether it is through statistical techniques and predictive modelling, interactive visualizations or taxonomies and frameworks, the ultimate goal is to optimize the performance of both students and staff of an educational institution, improve pedagogical strategies, streamline institutional costs, improve student interaction with teaching materials, target potentially at-risk students, responsibly change pedagogical approaches to improve grading systems through real-time analysis and enable teachers to assess their own learning effectiveness. Stakeholders and related objectives are summarized as follows (Lang et al., 2021).

1.3.2

Student:

- recommendations for educational activities,
- different ways of teaching,
- adaptive advice and recommendations,
- discussion,
- guidance in the learning process.

1.3.3

Teacher:

- interest in course content,
- analysis of student behaviour,
- identification of at-risk students,
- performance prediction,
- grouping of students according to criteria,
- search for weak points,
- improving courses through more effective activities and customization.

1.3.4

Researcher:

- evaluation of the learning management system,
- evaluation of course content,

- comparison of available analytical models.

1.3.5

Education providers:

- assistance in decision-making,
- find effective ways to improve courses,
- reducing school drop-out rates,
- guidelines when selecting students.

1.3.6

Administrators:

- support for the allocation of resources,
- improvement of educational programs,
- effective online (distance) learning,
- evaluation of teachers and study programs.

1.4 Challenges

1.4.1

As in the whole field of artificial intelligence, learning analytics faces many challenges, the fulfilment of which is a prerequisite for fully exploiting the potential that learning analytics can bring to education. LA's main challenges relate to:

- technological aspects,
- ethical aspects,
- data integrity,
- privacy,
- managerial aspects.

The purpose of LA is to use the data collected to optimize teaching and the environment in which learning takes place. The interventions derived from this can serve as a basis for developing measures to support risk groups and provide them with better assistance during their studies. Based on this, recommendations are proposed to support students in order to encourage them. Personalized learning environments come to the fore, providing students with appropriate visual results in an appropriate visual form. This could have the effect of motivating the student in terms of improving his attitude towards academic achievement.

1.4.2

Choosing the right environment and the right visualization technique can be a big challenge for all parties involved. Because of the amount of data collected and the focus on quantitative metrics, interpreting this data can be incredibly difficult. Therefore, it may be more profitable not to provide the student with all the information related to his results. The educator can discuss the results with the student, however, researchers acting as educators need specialized training, focused on pedagogical and psychological skills, in order to correctly interpret data on student achievement.

1.4.3

Another challenge in the area is the data itself, as universities and colleges constantly analyse data from their students for a variety of reasons. LA can therefore be seen as an innovative continuation of this principle, applied to take advantage of modern technologies and various data sources available today. Data can be examined and analysed for their impact in the educational context to improve the quality of learning and teaching, as well as to increase students' chances of success.

Universities, of course, require an individual's permission to collect and evaluate sensitive data for these purposes. Students must be aware of the purpose of data collection and the process of data analysis and must always be able not to disclose their data, of course, within the framework of applicable legislation. Thanks to consent, educators are able to monitor and analyse student behaviour during interaction with the learning management system. Access to this data must be ensured and institutions must adopt policies that deal with data protection and access. In addition, to ensure the best support and data quality, students must keep their data up to date, while complying with GDPR and data protection laws.

1.4.4

Another challenge in LA is the IT infrastructure in which the data resides. The first approach may be that the data is stored and processed at the university. This method has the advantage that data access and ownership is located at the university, which makes it easier to work with the data. However, it also presents certain disadvantages, within which you also need to think about the cost-benefit ratio, since such a solution can be quite expensive.

The second approach concerns cooperation with external service providers. In this case, individual solutions can be used, as there are many providers available who can meet specific needs. The advantage is that the costs incurred should be somewhat easier to estimate and much lower than when providing a private, individual solution. The negative aspects of cooperation with an external service provider relate to issues of data access and ownership, as well as meeting the necessary security standards when working with sensitive data, such as student performance data.

Regardless of working with an internal or external service provider, it takes time to set up the right infrastructure. Therefore, considerable effort must be made from the outset to find possible solutions, saving time and resources when LA implementation becomes critically important (Leitner et al., 2017).

1.4.5

Another category of challenges connects the LA development and deployment process. It covers a wide range of development of various applications and modules, from the design of a simple questionnaire to the development of an enterprise software solution. In addition, activities cover research and development, modification, reuse, monitoring and maintenance of initiatives or projects. It is necessary to take into account the scalability of the implementation. The number of students can vary arbitrarily, which can lead to a completely new concept of existing infrastructure. In addition, processes that were first created manually must be redefined so that they can be executed at least semi-automatically or fully automatically.

1.4.6

Although student data is stored, this is usually done through several information systems – in different formats, on different servers, and with different data owners. The effort required to process all data can be challenging, posing an additional challenge in adapting the raw data into a format suitable for further analysis. This is a very complicated process that requires careful planning and rigorous final implementation.

All LA implementations must ensure the privacy of stakeholders. Students need to trust finite systems, which is why keeping information private is of utmost importance. It is necessary to start minimizing the data collected and/or take steps to anonymize or pseudonymise the data. Even so, the situation can become very complex, e.g., when different data sources are merged, new and surprising results can be visualised and therefore new insights can be provided that were never intended. Since universities are huge institutions, there is a high risk of unauthorized persons accessing these interpretations of data (Leitner et al., 2017).

Privacy is a fundamental right of every person and must be respected. This means that any LA implementation must take this into account from the very beginning. However, this is often difficult because merging data can result in complex situations. It is therefore recommended to work with the highest possible level of transparency.

1.4.7

Various ethical and practical concerns arise in the context of LA challenges, as there is potential to collect personalized data and intervene at the individual level (Prinsloo, Slade, 2015). Working with sensitive data is a particular challenge. If such information were made public, it could cause harm to a particular person. It is

therefore necessary to provide for restrictions on who has access to the information and for what purpose(s) it is used.

Transparency is key, as is understanding the different needs of stakeholders. All objectives, objectives and benefits for the collection and use of data shall be explained in a clear and comprehensible manner.

The above challenges and approaches that can be used to overcome them will help in the implementation of LA. The examples presented cover only a small range of problems and other potential problems may still arise.

Learning Analytics

Chapter 2

2.1 LA basic elements

2.1.1

Learning Analytics

Learning analytics is defined by SOLAR as the measurement, collection, analysis, and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs” (SOLAR, 2011). We can say that learning analytics is an ecosystem of methods and techniques that successively gather, process, report and act on machine-readable data on an ongoing basis in order to improve the learning environments and experience (learn2analyse, 2020).

We will summarise the main characteristics of Learning analytics in this chapter.

2.1.2

Essential Element of Learning Analytics

We can start with four essential elements involved in all learning analytics processes (learn2analyse, 2020):

- Data, which creates as the primary analytics asset, is the raw material that gets transformed into analytical insights. Data in education includes information that is gathered in the learning process. Data relates to the learners, the learning environment, the learning interactions, and the learning outcomes.
- Analysis is the process of transforming the collected data to obtain actionable information from them using a set of mathematical and statistical algorithms and techniques. During data is pre-processed, transformed, and modelled with the aim to discover hidden knowledge and support decision-making process and action.
- Report is the most frequent form used to summarize what the analysis of the collected data can say about learning and to present this information in a meaningful manner. It is a set of techniques for organizing and presenting the results of the analysis of learners’ and learning data into charts and tables. Reports provide insights about the learning process stakeholder’s states during learning. Interpreting those insights can guide data-driven decision making to action taken.
- Action is the goal of any learning analytics process. It can have a form of the informed decisions and the practical interventions that the educational stakeholders will undertake. The results of follow-up actions will determine the success or failure of the analytical efforts. Learning analytics is useful only if there is action as a result of its implementation.

2.1.3

Learning Analytics Cycle

Learning analytics research combines these four elements with the following steps to act in favour of all stakeholders, who participate in the learning process:

- capture,
- report,
- predict,
- act,
- refine.

While learning analytics starts with learners, capturing and gathering the raw data is the first step of the learning analytics cycle, following by introducing metrics for sharing a common understanding of the data in educationally meaningful ways. In other words, learner and contextual data are collected and transformed into metrics (analytics), according to the learning objective that needs to be addressed. Subsequently, the different types of metrics guide human decision-making and interventions. Data-driven decision-making requires integrity and quality to be ensured.

2.1.4

Subsequently, reports and analysis of the defined metrics can help predicting the future states of the learners and gaining insights into the learning processes. The cycle is not complete until these metrics are used to drive one or more interventions (actions) that have some effect on learners. As was already mentioned, delivering personalized learning action to everyone closes the cyclical process of learning analytics and generates new data about learners.



We see that the role of the learner is fundamental in this process. Since learning analytics actions are extracted from the learners' and learning data, we need to know what is the learner's data that will be used in learning analytics, and what types of

learning analytics can be formed from the learner's data. These questions will be discussed later.

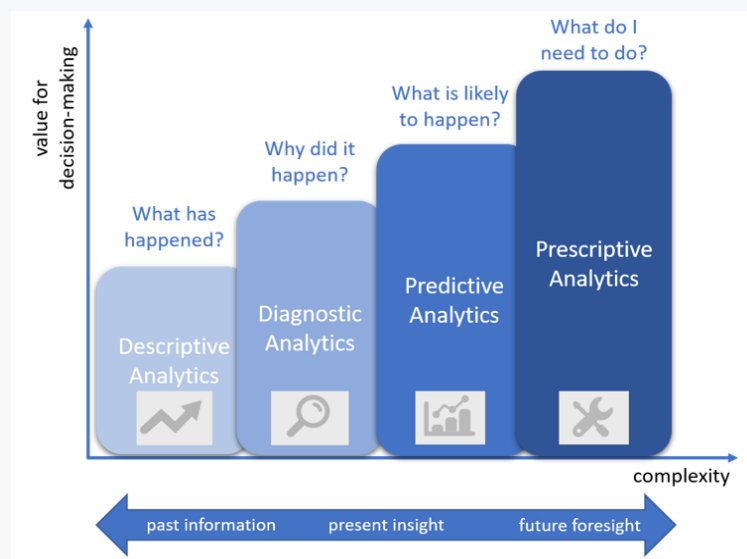
We will start with explanation what types of learning analytics we can use depending on data availability, quality, time for its analysis and expected type of intervention.

2.1.5

Learning Analytics Types

Depending on the learning analytics objectives, we can obtain different learning analytics outcomes from the same or different learner and context data. We can select the following levels of the metrics according to their sophistication, the complexity of the analysis method employed, and the value the metrics can add to human decision-making (Lang, Siemens, Wise & Gasevic, 2017):

1. Descriptive analytics - we use data aggregation and data mining to provide insight into the past and answer: "What has happened?" (e.g., reports and descriptions).
2. Diagnostic analytics – we dissect the data with methods like data discovery, data mining and correlations to answer the question "Why did it happen?" (e.g., interactive visualizations).
3. Predictive analytics – we utilize a variety of data to make the prediction and apply sophisticated analysis techniques (such as machine learning) to answer the question "What is likely to happen?" (e.g., trends and predictions).
4. Prescriptive analytics – we utilize an understanding of what has happened, why it has happened and a variety of "what-might-happen" analysis to help the user determine the best action to take and answer the question "What do I need to do?" (e.g., alerts, notifications, recommendations).



2.2 Learning analytics data

2.2.1

Data Quality

Data is generally considered high quality if it fit for [its] intended uses in operations, decision making and planning and data is deemed of high quality if correctly represents the real-world construct to which it refers. However, data often suffer from inaccuracies, biases or even manipulations and we should ensure to be as much reliable and valid as possible.

Data quality is critical for educational institutes, as well. There are many aspects to data quality, which can be evaluated. We summarised the most frequent data quality aspects as follows:

1. Completeness - There are no gaps in the data from what was expected to be collected and what was collected, i.e., there are no missing data. We can state the collected dataset is complete.
2. Consistency - The data types must align and be compatible with the expected versions of the data being collected, i.e., there are no contradictions in the data types and the data are usable.
3. Accuracy - Collected data are correct, relevant and accurately represent what they should.
4. Timeliness - The data should be received at the expected time for the information to be utilized efficiently.
5. Validity - A measurement is well-founded and likely corresponds accurately to the real world.
6. Uniqueness - There should be no data duplicates reported.

Among the 6 dimensions, completeness and validity usually are easy to assess, followed by timeliness and uniqueness. Accuracy and consistency are the most difficult to assess.

2.2.2

It is important to clarify that raw data quality strongly affects the analytics quality. Learning analytics transform of the raw learner and learning data collected, according to the objectives set. These metrics will next be treated as “data” themselves, and they will be subjected to further processing. Just like with any kind of data, quality also matters for learning analytics metrics: what the specific metrics can reveal is strongly dependent on their quality. In most cases, limited quality will have the direct result of lack of trust in the metrics, and consequently, poor decisions and gradual abandonment of the data-driven educational decision-support system. Poor quality data is troublesome (The data quality benchmark report, 2015). Educators cannot and will not trust insights that are acquired by processing corrupted, duplicate, inconsistent, missing, broken, or incomplete data. Learning

analytics metrics quality is expected to increase the value of the learner and learning data and the opportunities to use them properly (learn2analyze, 2018).

2.2.3

Data-driven Decision Making

Data analytics refers to methods and tools for analysing large sets of different types of data from diverse sources, which aim to support and improve decision-making. Data analytics are mature technologies that are currently applied in real-life financial, business and health systems. The provision of educational data by itself does not automatically lead to improved teaching and learning process. What is most important is not the amount of data that we have access to, but what we do with it. How will we identify actionable insights from the educational data?

We can follow the Data-Driven Decision-Making approach (DDDM), which is defined as the systematic collection, analysis, examination, and interpretation of data to inform practice and policy in educational settings. Data Driven Decision Making (DDDM) crosses all levels of the educational system and uses a variety of data from which decisions can be made. Therefore, it can be challenging to engage in DDDM due to data being siloed in different sources and at different levels. Considering the expected results of learning analytics application, we should decide, which fall into two categories (Marsh, Pane, & Hamilton, 2006):

- using data as a diagnostic tool to identify, inform, or clarify issues both at an individual (e.g., identifying goals or needs) and at a systematic level (e.g., informing the design of courses or curricula), and
- using data to act (e.g., assessing and acting upon differential outcomes among the learners' population, personalised interventions for at-risk students).

Data is not a static entity and therefore decisions based on data should not be static either. As a result, data usage and evaluation should be continuous and integrated into existing decision-making processes.

Data resources and types

Chapter **3**

3.1 Data

3.1.1

Educational Data

In the context of education, learners are leaving behind rich digital traces throughout the course of their study. Educational data comprises a wide range of datasets about learners, their learning and the environments in which they learn, stored in various sources.

Educational data and data analytics technologies can support us in developing a better understanding of our learners' activities, behaviour and preferences, by identifying patterns and trends in the data that, in turn, can help us predict possible future outcomes and take actions for improving the learners' experience in our courses.

For example, e-learning courses or virtual learning environments like Priscilla, allow:

- instructional designers to use data to (re)design their courses,
- tutors to use data to adjust their tutoring and learners' support strategies,
- school teachers to use data to better plan inside and outside classroom activities and assess students' learning.

On the other hand, data could potentially enable learners to take control of their own learning. When appropriately delivered, data can provide learners with better insights about their current academic performance in real-time, about their progress (also in comparison to their peers) and recommendations about what they need to do for meeting their learning goals and help them to make informed, data-driven choices about their studying.

3.1.2

Learning analytics can:

- enable schools to decrease drop-out rates,
- be a powerful tool only for assessing learners' performance,
- enable teachers to see what students learn but not how students learn.

3.1.3

Data Resources

The answer to the questions of where and how we obtain data can seem relatively simple, as data and technology currently accompany almost all areas of human activity, including education. The reality as usual not so easy. Let dive into this issue deeper.

Historically, learning analytics research, as well as related areas of research, originally focused only on data coming from direct interaction between lecturer and student. Typical examples of such data are attendance lists, partial or final grades, fulfilment of educational goals, etc. However, these administrative data do not bring the desired effect because they do not provide a comprehensive or sufficiently detailed picture of how the learning process takes place.

The situation began to change with the advent of computer-based education (CBE) and achieved significant development with the expansion and availability of various types of web-based educational systems (WBES), such as

- LMS (Learning Management System),
- ITS (Intelligent Tutoring System),
- AEHS (Adaptive Educational Hypermedia System) or
- MOOC (Massive Open Online Course).

3.1.4

LMS

LMS represents the most widespread type of WBES. Although its popularity has stagnated recently, it remains the focus of LA research, not only at the individual course level, but also at the higher organizational level of organizations. LMS collects large amounts of user data, track user activity, and provides simple analytics tools. Its problem is often the proprietary nature of logs.

Recently, we can encounter their successors in the literature called the new generation of digital learning environments (NGDLE), whose conceptual model of functionalities already directly envisages the use of LA modules for analysing user behaviour, assessing their progress and level of acquired knowledge, as well as personalizing the environment. In terms of storing user data, these systems can store logs in a standard format, e.g., xAPI, in both internal and external storage.

3.1.5

MOOCs

Another group of WBES, the so-called massive open online courses, MOOCs, can rightly be attributed to a considerable share in the development of the LA domain, because the massive expansion of the target group of potential users of MOOCs has gone beyond traditional educational institutions and brought interest to the commercial sector. Moreover, it is in this group of WBES that we can really start talking about the connection of the LA area with the area of big data, since thanks to their worldwide spread and availability, the numbers of active users and data on their activity in the course have acquired dimensions that are not easy to process and analyse by conventional means.

3.1.6

The systems known as **ITS** and **AEHS** have been developed in order to adapt to changing user demands and behaviour using different approaches to user modelling. Both systems, unlike the approach taken in LA, seek to largely automate the customization process, while LA seeks to support decision-making at the level of representatives of various stakeholders. For this reason, the recent deployment of ITS and AEHS is so far limited, rather experimental, and works mainly in narrowly oriented domains, such as teaching mathematics or geometry or languages. The main idea of these systems is interesting, but very difficult to implement in practice.

ITS creates an individual model of the behaviour of each user of the system and examines the way it interacts with the system. It creates a so-called user model. In this process, in addition to the enormous amount of data generated by users, it also uses a domain model together with a set of defined domain restrictions, a set of decision rules, etc. As a result, ITS provides direct instructions, recommendations or feedback to the user and thus helps him choose a more suitable way for him to acquire new knowledge. The weak point of these systems is precisely the need to describe the domain model in detail and define appropriate constraints.

A slightly different approach in terms of feedback to the user is provided by the so-called adaptive educational hypermedia systems (AEHS). AEHS represent one of the first and most popular applications of hypermedia systems. The aim is to adapt to user requirements through a combination of domain model, user model as well as advanced analysis of data resulting from user and system interaction. Unlike ITS, they mostly provide semantically richer data. Again, their wider spread in practice is hampered by the complexity of the models used. Based on the analysis of current review work, it is clear that the original perception of the concept of adaptivity and AEHS in general has changed over time and has settled in a broader sense than Adaptive Learning, which is part of LA and has also replaced the original designations.

3.1.7

We find other more detailed divisions of WBES in the LA area in addition to the above types of virtual learning environments that collect user data suitable for further analysis, for example:

- Computer-Based Information Systems (CBIS),
- Knowledge-Based Systems (KBS),
- Computer-Based Education Systems (CBES).

As we have already mentioned, most of these systems store data about users and their direct interaction in the form of log file records. These educational systems and their successors are also the most common source of data that are analysed from different points of view in professional publications using various methods knowledge discovery. Data from these systems will also be used in the practical examples given in the following chapters.

3.1.8

Even more complex requirements for data, their sources and technical problems associated with their collection, processing and analysis are the subject of research in Multimodal Learning Analytics (MMLA), which seeks to understand and optimize learning in a real environment that does not necessarily use computers but other devices in the learning process. MMLA uses a number of sensors to attempt to capture, process and integrate and analyse natural communication patterns such as speech, writing and nonverbal interaction (such as movements, gestures, facial expressions, gaze, biometrics, etc.) taking place during real learning activities. We are increasingly aware that learning is a multimodal process that involves voice, gestures, visual attention, and other biological and mental processes that happen simultaneously and cannot simply be captured by one device or technology. An open question of current research in the field of MMLA remains how to collect this data from various systems and devices without affecting the learning process, how to transmit, integrate and standardize data from various sensors, as well as further analyse.

We see that in the field of education there is many data sources that we can use in the analysis process, provided, of course, compliance with the relevant ethical and legislative standards. However, the data rarely meet our requirements and are not suitable directly for research. As in other areas of data science, it is also necessary to first understand the data and prepare it in the required form.

Individual activities related to data extraction, transformation and storage in the field of education do not differ significantly from established procedures and are described in data engineering and to it closely related topics like database systems, data warehouses, data lakes and big data.

3.2 Big data in education

3.2.1

Data is identified as one of the key factors driving change in the 21st century. We use the term the “data revolution”, the “era of big data”, or more simply “big data” to describe the tremendous increase in the amounts of data we generate in all aspects of our lives, including education.

We often use the 4 V characteristics of Big Data to emphasize, that this data can not be processed using conventional tools and approaches. However, big data allow us to achieve superior value from analytics on data, which has the following characteristics:

- Volume - The size of available data has been growing at an exponential rate. "With higher data volumes, you can take a more holistic view of your subject's past, present and likely future".

- Velocity - Data streams are created at an unprecedented speed. "At higher data velocities, you can ground your decisions in continuously updated, real-time data".
- Variety - Data comes in all types of formats. "With broader varieties of data, you can have a more nuanced view of the matter at hand".
- Veracity - Data veracity is not only how accurate or truthful a data set may be, but also how trustworthy the data source, type, and processing of it is. "As data veracity improves, you can be confident that you're working with the truest, cleanest, most consistent data".

3.2.2

We need to use huge computers to analyse the million pieces of data we generate on a daily basis to process big data and elaborate the endless possibilities it offered. Please select the right answer.

- False
- True

3.2.3

Which are the main V characteristics of Big data? Please select the right answers.

- Velocity
- Variety
- Variability
- Value
- Viscosity
- Vagueness

3.2.4

Metadata

If we talk about data and systems, which generate or store data, we need to also mention metadata. In the context of education, metadata can more aptly be defined as tags used to describe educational assets.

Metadata is usually defined as data about data. Johnson, L.R., et al. (2018) introduced more precise definition, according to which, metadata is information about a data set that is structured (often in machine-readable format) for purposes of search and retrieval. Metadata elements may include basic information (e.g., title, author, date created) and/or specific elements inherent to data sets (e.g., spatial coverage, time periods). Therefore, metadata helps:

- to organize,
- find,

- understand data.

Simultaneously, metadata answers the following questions about data:

- *Who created it?*
- *What is it?*
- *When was it created?*
- *How was it generated?*
- *Where was it created?*
- *How may it be used?*
- *Are there restrictions on it?*

3.2.5

We can divide metadata to the following types:

- Descriptive metadata, which can describe a learning asset or resource related to education – including learning standards, lessons, assessment items, books, etc. – for purposes such as identification, search, and discovery. Descriptive metadata can be thought of as a keyword or tag on an asset that makes it easier to find. Examples include subject, grade level, and related skills and concepts.
- Administrative metadata is used to manage a learning asset. Examples of this type of metadata include status, disposition, rights, and licensing.
- Structural metadata describes how data is organized or formatted and is often governed by a widely adopted standard that ensures the data is accurately represented when exchanged and presented. Structural metadata enables content to be machine readable.

3.2.6

Educational Data Literacy

We see that all stakeholders, who are involved in learning analytics process should have a set of special competences, which are covered by the term educational data literacy. Educational data literacy can be defined in several ways as:

- the ability to collect, manage, evaluate, and apply data, in a critical manner (Ridsdale et al., 2015),
- the ability to accurately observe, analyse and respond to a variety of different kinds of data for the purpose of continuously improving teaching and learning in the classroom and school (Love, 2012),
- the ability to understand and use data effectively to inform decisions ... composed of a specific skill set and knowledge base that enables educators to transform data into information and ultimately into actionable knowledge (Mandinach & Gummer, 2013),

- the ability to continuously, effectively, and ethically access, interpret, act on, and communicate multiple types of data from state, local, classroom, and other sources in order to improve outcomes for students in a manner appropriate to their professional roles and responsibilities” (Data Quality Campaign, 2014).

3.2.7

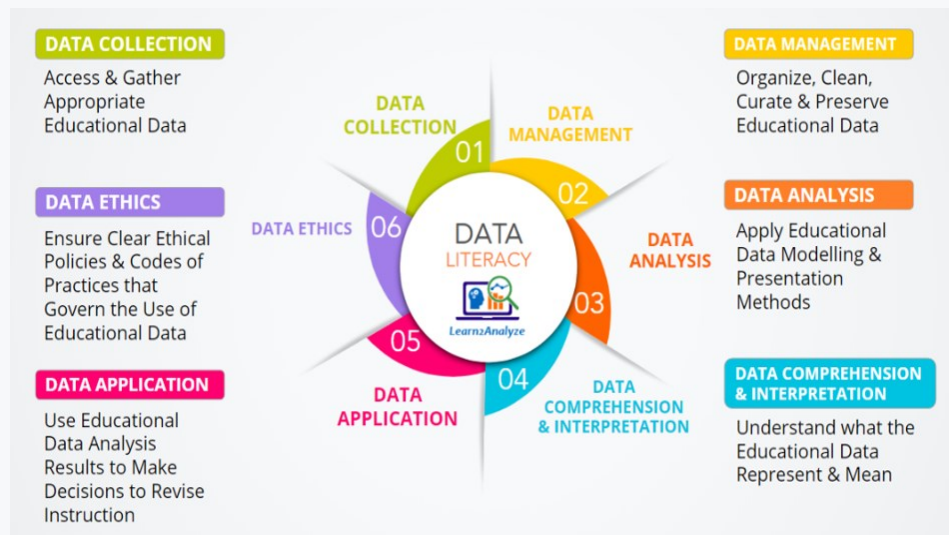
As a result, the Data Literacy Campaign recommends the following set of Data Literacy Competences for stakeholders of the learning process:

1. Locate and Collect Relevant Educational Data.
2. Synthesise and Analyse Educational Data from Diverse Sources.
3. Know about Educational Data beyond Grades.
4. Understand How to Use Educational Data beyond Grades.
5. Engage in a Data-Driven Continuing Inquiry Process.
6. Use Data Analysis to Customise Teaching Plans to Diverse Groups.
7. Use Own Data to Reflect on Practice.
8. Facilitate Students to Understand their Data.
9. Communicate Insights from Data Analysis to Diverse Internal and External Stakeholders.
10. Monitor this process in a continuous manner.

However, emerging advancements related to the use of data-driven design and delivery of technology supported learning, exploiting Educational Data Analytics are not yet thoroughly addressed by existing competence frameworks for education professionals (instructional designers, trainers, educators, teachers). Existing professional competence frameworks for instructional designers and trainers almost ignore the dimension of Educational Data Literacy.

3.2.8

This gap tries to fulfil the Learn2Analyze project, which developed a comprehensive proposal for an Educational Data Literacy Competence Framework to enhance existing competence frameworks for instructional designers and e-trainers of online courses with new Educational Data Literacy competences. The Learn2Analyze Educational Data Literacy Competence Framework comprises of six competence dimensions and 17 competence statements, as captured in the above graphic.



The competences summarised in the graphics is described in the following:

1. Data Collection

- 1.1. Know - understand - be able to obtain, access and gather the appropriate data and/or data sources.
- 1.2. Know - understand - be able to apply data limitations and quality measures (e.g., validity, reliability, biases in the data, difficulty in collection, accuracy, completeness).

2. Data Management

- 2.1. Know - understand - be able to apply data processing and handling methods (i.e., methods for cleaning and changing data to make it more organized – e.g., duplication, data structuring).
- 2.2. Know - understand - be able to apply data description (i.e., metadata).
- 2.3. Know - understand - be able to apply data curation processes (i.e., to ensure that data is reliably retrievable for future reuse, and to determine what data is worth saving and for how long).
- 2.4. Know - understand - be able to apply the technologies to preserve data (i.e., store, persist, maintain, backup data), e.g., storage mediums/services, tools, mechanisms.

3. Data Analysis

- 3.1. Know - understand - be able to apply data analysis and modelling methods (e.g., application of descriptive statistics, exploratory data analysis, data mining).
- 3.2. Know - understand - be able to apply data presentation methods (e.g., pictorial visualisation of the data by using graphs, charts, maps, and other data forms like textual or tabular representations).

4. Data Comprehension & Interpretation

- 4.1. Know - understand - be able to interpret data properties (e.g., measurement error, outliers, discrepancies within data, key take-away points, data dependencies).

- 4.2. Know - understand - be able to interpret statistics commonly used with educational data (e.g., randomness, central tendencies, mean, standard deviation, significance).
- 4.3. Know - understand - be able to interpret insights from data analysis (e.g., explanations of patterns, identification of hypotheses, connection of multiple observations, underlying trends)
- 4.4. Be able to elicit potential implications/links of the data analysis insights to instruction.
5. Data Application
 - 5.1. Know - understand - be able to use data analysis results to make decisions to revise instruction.
 - 5.2. Be able to evaluate the data-driven revision of instruction.
6. Data Ethics
 - 6.1. Know - understand - be able to use the informed consent.
 - 6.2. Know - understand - be able to protect individuals' data privacy, confidentiality, integrity, and security.
 - 6.3. Know - understand - be able to apply authorship, ownership, data access (governance), re-negotiation and data-sharing.

3.3 Data types

3.3.1

Obtaining a suitable dataset on which individual knowledge discovery tasks or specific algorithms could be easily trained is quite complicated. The simplest, and at the same time the most certain way, is to create a data set from our own available data sources. Simultaneously, it is important to emphasize that we have all the necessary permissions, and we can guarantee data security throughout the research. In this case, we can apply some approaches typical for the data pre-processing phase and prepare a set of variables that we want to analyse already in the process of data acquisition.

If we consider a set of data freely available on the Internet, we should pay attention to the level of granularity of the data. In other words, this data should meet our requirements, the goal of the research as well as the input assumptions of the methods we want to apply. For example, we often encounter the following division of data levels:

- national and international level,
- regional level,
- institutional level,
- workplace level,
- study program level,
- class,
- course,
- student.

3.3.2

The field of education represents a broad application domain in which the benefits and possible links with other disciplines such as data science, computer science, artificial intelligence, as well as behavioural, economic, social and ethical aspects are intensively explored. As in other application areas of knowledge acquisition, also in the field of education, the growing volume of data has an impact on the overall development of the area, research, on the infrastructure of the organization and its management, return on investment, sustainability, or ultimately competitiveness in the given domain.

The scope of this domain, whether in terms of data quantity, number of companies involved, research tasks solved, turnover, diversity of technologies, or social impact, are comparable with other application domains of knowledge acquisition. If we focus on the sources and process of data collection, we can conclude that LA is moving from the original field of education to the field of log mining or even big data analysis, although not all characteristics of big data referred to as "4Vs" are currently considered to be met in the field of education. LA works with more complex data formats to capture user behaviour in the heterogeneous environment of information systems and ubiquitous wearable technologies.

3.3.3

Data that come from different systems can be classified into groups based on the selected characteristic:

- relational data
- transaction data,
- time series, sequential data,
- text data,
- multimedia data,
- web data.

3.3.4

Another important view emphasizes the sensitivity of data, the importance of data origin (provided, observed, derived data) and especially the primary purpose of data use, why the type of data is collected in the organization:

- demographic data – This is a sensitive type of data, but sometimes it can bring new insights into different groups of users and the specifics of their learning,
- academic data – presents partial achievements, scores and grades,
- data about the student's past performance,
- student-generated data – often in the form of unstructured data, texts (discussion papers, papers, projects, presentations),

- data directly related to the learning process – activities, digital footprint of the student, stored most often in the form of logs in a known structure directly in the systems,
- metadata illustrating information about the environment in which learning takes place, reasons, student preferences.

3.3.5

For effective data-driven decision-making, it is essential that we understand the domain being studied, the data and the relationships between them. If we are looking for data suitable for analysis using LA methods, we should be able to answer the following questions:

- *Why do we need this data?*
- *What data do we need?*
- *When is a good time to collect data?*
- *Where is the necessary data located?*
- *How will we collect the necessary data?*
- *Who will collect the data and be authorized to work with it?*

Without answering these questions, we cannot fully understand the data and correctly estimate the possibilities of their use for the goals pursued, which may be, for example, the choice of a suitable strategy for various forms of study, finding out the causes and proportion of too early abandonment of studies by students, or predicting their final evaluation from the subject or the entire study.

3.3.6

In general, we can further divide the data into:

1. Static, which remains unchanged for a long time, for example, student, teacher, budget, stored at school, university or government registry informing about the current state of the sector.
2. Dynamic, generated in a shorter period, created by the activities of all stakeholders in the learning process, for example in the form of logs.



3.3.7

At the same time, according to the time at which the data was created or stored, we can divide the data into the following categories:

- input,
- context,
- procedural, and
- output.

3.3.8

Moreover, according to the definition, we can distinguish two major categories of data, the qualitative and quantitative data. The following figure illustrates differences between both categories.

 qualitative data	vs  quantitative data
qualitative data is data that is not easily reduced to numbers. In a school setting, qualitative data may come from observations, work samples, conversations, written documents and more.	quantitative data is any information that can be reduced to a set of numbers. Information from which you can create averages, differences or totals is quantitative data.
is particularly helpful for summarising large amounts of information in a snapshot, tracking trends over time and understanding patterns and differences from one group of participants to another.	is particularly helpful for exploring emerging issues, providing rich description and context about a complicated issue and building theories about what might explain trends and patterns in quantitative data.
help us answer questions about <ul style="list-style-type: none"> • 'what', • 'how' and • 'why' of a phenomenon 	help us answer questions about <ul style="list-style-type: none"> • 'how many' or • 'how much'

A combination of different types of data is most effective in generating powerful evidence to assess learning performance and improve teaching practice. Both quantitative and qualitative data is equally important in these processes.

3.3.9

Educational Data is everywhere. We need to collect the necessary data to inform our decisions and benefit from them. To do this, we need to answer to the following Four Ws and One How:

- *Why is data needed?*
- *What data is needed?*
- *When will the data be collected?*
- *Where is the data located?*
- *How will the data be collected?*

We can add another one question: Who will collect or grant access to the needed data. This question should be answered, since, obviously, we can only collect data to which we have access and which we have been granted permission to use.

The collected data must meet three basic characteristics:

- **Relevancy:** The data must directly relate to the research questions being answered.

- Reliability: The data must be measured, trustworthy, and consistent.
- Validity: The data must measure what we intend to measure.

3.3.10

Collecting and analysing educational data are joined with several types of barriers like:

- access to educational data,
- timely collection and analysis of educational data,
- quality of educational data,
- lack of time and support.

Moreover, we must not to forget the educational data ethics. Open Data Institute (ODI) defines Data Ethics as a branch of ethics that evaluates data practices with the potential to adversely impact on people and society – in data collection, sharing and use.

Meantime, several frameworks, policies, and guidelines have been developed to address data ethics issues, including JISC's code of practice in 2015 (updated in 2018), the LACE (Learning Analytics Community Exchange) framework in 2016 and the ICDE (International Council for Open and Distance Education) Global guidelines in 2019. To help identify potential ethical issues associated with a data project or activity and the steps needed to act ethically, Open Data Institute has also designed the Data Ethics Canvas in 2018.

Methodological Frameworks for Learning Analytics Methods

Chapter **4**

4.1 Data science process

4.1.1

In the previous chapter, we dealt with the characteristics of data and got acquainted with the sources of educational data. Like a software project, a data science project follows certain recommendations that can be summarized in several steps. If they are generally accepted and proven in practice, we refer to them collectively as methodologies or process frameworks.

The methodology will guide us to perform all steps of data analysis leading to useful results, their correct interpretation, or to making the results available in the form of an application or service.

In this chapter, we will briefly introduce some methodologies, indicate current developments in the field of data science, machine learning, respectively LA. We will then try their use on practical examples.

4.1.2

Data Science Process

The Data Science Process framework provides a simplified view of the entire knowledge discovery process in projects where machine learning algorithms are currently used (Blitzstein, 2013; Blitzstein and Pfister, 2015). It lists the following phases:

- Phase 1: Ask the right questions,
- Phase 2: Get data,
- Phase 3: Explore the data,
- Phase 4: Create a model,
- Phase 5: Interpret the result.

Although phases are presented sequentially, their actual use takes place in multiple iterations.

4.1.3

Enhanced Data Science Process

This rather intuitive sequence of phases of the Data Science Process framework is very brief to know without prior practical experience what activities or tasks we should carry out in each phase. Because of this, an extended version of it was created. It contains the following 8 steps, which contain tasks leading to solving the project (Mayo, 2018):

- Phase 1: We formulate the studied problem and look at it from a distance - we define goals from the point of view of the issue or domain being studied, consider the need to deploy machine learning, compare existing scenarios on how the problem can be solved, meet the prerequisites for their deployment and the level of knowledge needed. At the same time, we will determine the type of knowledge discovery task we want to solve.
- Phase 2: We obtain data – we determine the amount of data we need, its type and source, examine the conditions for obtaining this data, remember to anonymize the data and partially transform it into the desired form at the time of acquisition, as well as create a training, testing and validation set.
- Phase 3: Review the data – we take all steps necessary to understand the data we will model using exploratory analysis methods, examine whether the data meets the necessary prerequisites for the use of algorithms of the selected knowledge discovery task, communicate with a domain expert, study input variables, create model documentation and record metadata.
- Phase 4: Prepare the data – we perform all necessary data transformations based on the findings of the previous step, clean the data, select the appropriate combination of input variables, standardize, and normalize the data as needed.
- Phase 5: Create a list of promising models – we create a set of models that we compare with each other with appropriately selected metrics and select the most promising ones, which we will tune in the next step by searching for suitable input parameter values for each model.
- Phase 6: Tune the model – we investigate the impact of different model parameter settings (hyperparameter tuning). Use not only a training but also a test dataset.
- Phase 7: We present the solution – we visualize the modelling results, focus on assessing to what extent the original objectives have been met, whether all assumptions and limitations have been met, making sure that the client understands the added value that the created model brings.
- Phase 8: Deploy the solution – we prepare the created model using machine learning into production, implement the solution as part of the software solution, i.e., in the form of an application or service, retrain the model on new or updated data.

We apply usually these steps to learning analytics tasks. It is important to emphasize that these steps serve mainly to understand the complexity of all research based on data analysis, to familiarize yourself with procedures. In real analytical projects, data is always unique, so each of these steps requires theoretical knowledge and practical experience.

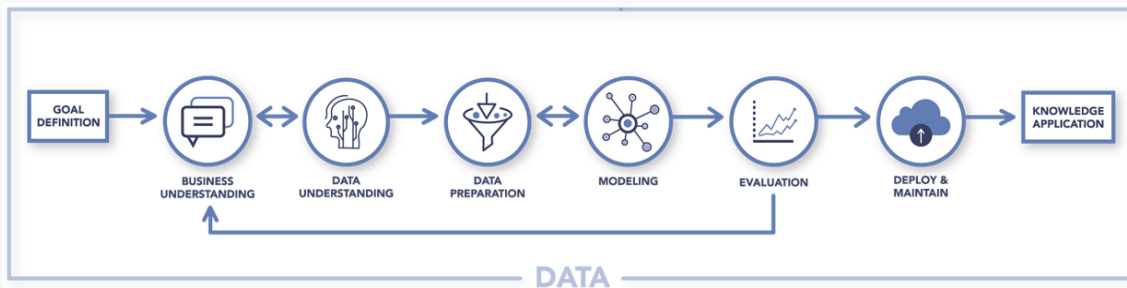
4.2 Methodologies

4.2.1

CRISP-DM Methodology

We use the proven CRISP-DM methodology as the basis for the sequence of steps in most examples focused on research in the LA domain and knowledge discovery in general. The CRISP-DM methodology is a software-independent methodology that provides a unified framework for solving various knowledge discovery tasks (Munk, 2013).

This methodology is based on the more general KDD process model (KDD process), which applies specific mathematical and statistical methods to data in order to discover and extract hidden patterns. This process defines five steps (selection, pre-processing, transformation, data mining, interpretation).

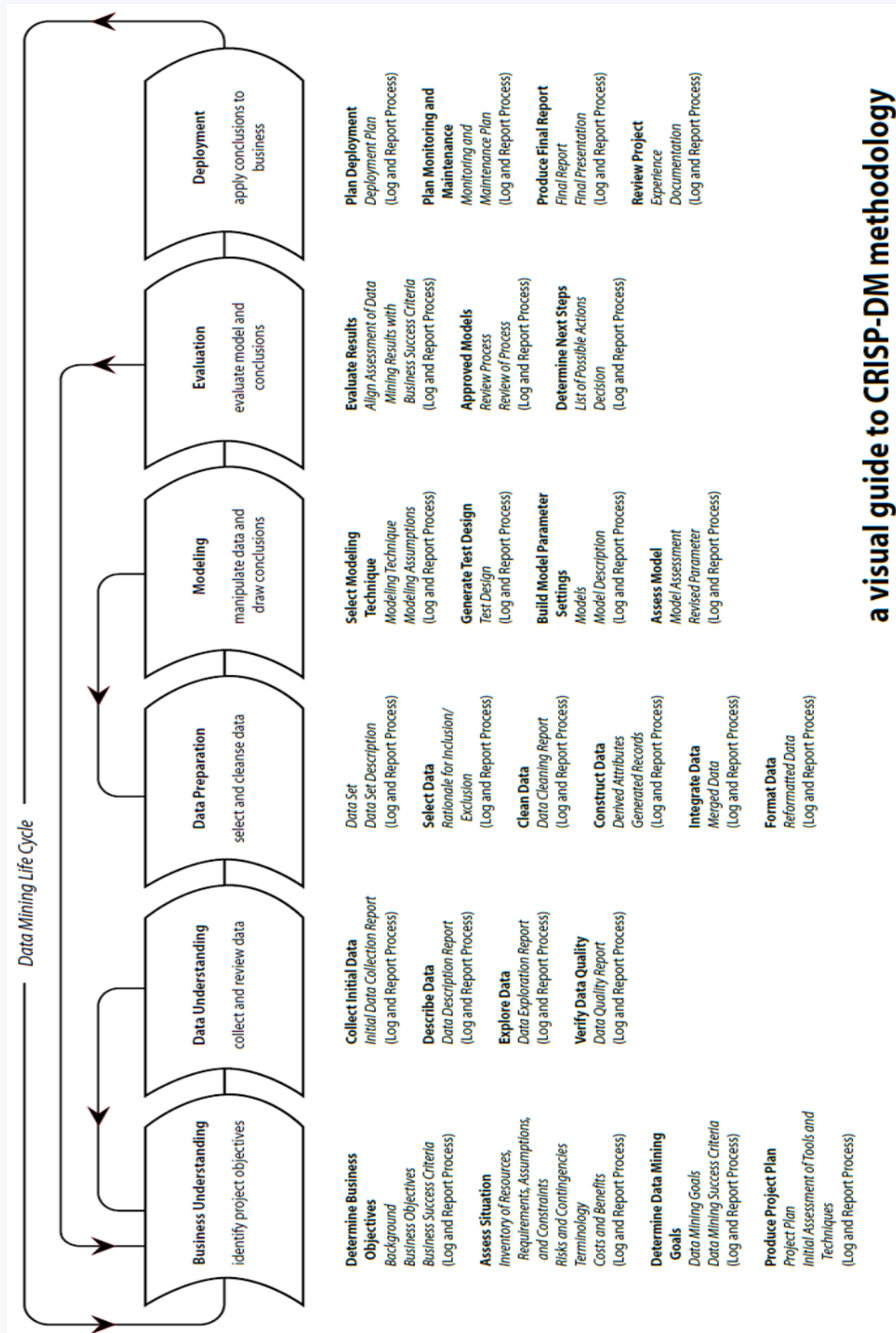


The CRISP-DM methodology assumes that the process of knowledge acquisition and data mining consists of six, interrelated, phases (Berka, 2003):

1. business understanding phase,
2. data understanding phase,
3. data preparation phase,
4. modelling phase,
5. evaluation phase,
6. deployment and maintenance phase.

4.2.2

These phases are applicable with minor variations in various application domains, including the analysis of educational data. We shortly describe each phase in the next chapters.



Generic Tasks
Specialized Tasks
 (Process Instances)

SOURCE CRISP-DM 1.0
<http://www.crisp-dm.org/download.htm>
 DESIGN Nicole Leaper
<http://www.nicoleleaper.com>



The initial phase focuses on business understanding and refining the objectives as well as research requirements in the understanding phase. The next stage consists of understanding the data, which is fundamental to familiarize ourselves with the collected or obtained data, identifying data quality problems, and also can reveal interesting subsets of variables to form hypotheses regarding hidden knowledge.

The methodology emphasizes data preparation phase, which often represents the most time-consuming part of the whole process. This phase applies the data pre-processing techniques needed to compile the final dataset. Part of this phase is also the use of appropriate methods for extracting properties and defining their abstraction, which will enrich the original data with semantics and thus allow a more accurate interpretation of the acquired knowledge. Such data can be used to input algorithms in the modelling phase.

In the modelling phase, various modelling techniques and algorithms are selected and applied, while optimal values of the model parameters are also sought. Since there are many techniques and algorithms for the same type of knowledge discovery problem/task, we decided to apply mainly machine learning techniques in the presented research.

The proposed models from the previous phase should be evaluated and evaluated for satisfactory implementation in the penultimate phase of the CRISP-DM methodology. After finding a suitable model based on selected performance metrics, we qualitatively evaluate the benefits of the model with the help of an expert in the field of knowledge discovery and education.

4.2.3

Agile Project Methodology

The CRISP-DM methodology and its variations represent a classic systematic approach to solving the problem. Given the success of agile approaches in software projects, it is natural to examine the potential benefits of basic ideas of agile process frameworks in data science project management. Available sources show that the basic principles of agile development and project management have intervened in the field of data science. Therefore, we will now summarize their basic characteristics.

As mentioned above, the CRISP-DM methodology can be compared to the classic waterfall model of the software development lifecycle in analogy with software development. The basic characteristic of the waterfall model is the division of project activities into several consecutive phases, while the start of the next phase is conditioned by the completion of the previous one. This sequential approach is not entirely suitable for managing a software project due to its little flexibility and absence of iterations.

From this point of view, the implementation of an iterative and incremental approach to CRISP-DM, typical of agile methodologies, is clearly beneficial. The addition of elements of agile process approaches then leads to methodologies, sometimes referred to as Agile + CRISP-DM.

If we compare software and analytics project management, we may encounter an effort to adapt the well-known core values and principles of the Agile Manifesto we

know from software development to agile analytics projects. This results in the following four customized values (Thurber, 2020):

- We prioritize regular collaboration with stakeholders over processes and tools to quickly integrate their feedback into the proposal.
- We take into account changing project requirements, even later in the knowledge discovery process.
- We try to increase customer satisfaction through customer involvement in the project as well as timely and continuous provision of valuable overview of project events.
- We evaluate changes based on feedback and interim results derived from ongoing processes and implement them beyond the baseline.

4.2.4

The following basic values can be considered as confirmation of the current pragmatic trend:

- Solving complex problems requires flexibility, responding to change that is bound to occur, short iterations, and communicating in a team whose members have overlapping knowledge but are aware that the client is a source of valuable information about the modelled phenomenon in the domain being studied.
- This creates team-specific process frameworks whose actual deployment depends on the task being solved, its complexity and the experience of the project team.

The project managed in this way is then built on trust, flexibility, cooperation, and joint responsibility of the team for the result. At the same time, the project is divided into several sprints, in which individual phases of the CRISP-DM methodology can be represented differently (Thurber, 2020).

For example, the first sprint can be focused on understanding the issue, data availability and consolidation, current practices, and current performance. The next sprint may focus on creating a base model. This is followed by one or two sprints aimed at finding the best specification of the model by tuning its parameters. The last sprint can ensure appropriate deployment of the solution. Each sprint has scheduled meetings to explore and collaborate with relevant stakeholders.

4.2.5

Team Data Science Process

The Team Data Science Process (TDSP) is an example of such an agile iterative methodology created for the effective delivery of predictive analytics solutions. Its advantage is the sophistication of procedures and documentation, as well as the connection with the possibilities of current technologies. At the same time, we must

mention that despite its agility, it is quite complex in terms of documentation of individual phases.

4.2.6

Guerrilla Analytics

Guerrilla Analytics is another example of an agile methodology that defines the following seven principles to eliminate chaos by introducing dynamics, complexity, and constraints into data project management (Ridge, 2014):

1. Principle 1: Space is cheap, confusion is expensive – saving space costs time if we cannot go back to previous versions of data and their origin (data provenance) or reproduce previous research steps.
2. Principle 2: We prefer simple visual structures and conventions – we maintain a clear structure of source codes, data throughout the project, so we will orient ourselves in the project with the passage of time, just like new team members.
3. Principle 3: We prefer automation using programming languages and graphics tools, avoiding modifications for which there is no record.
4. Principle 4: We maintain a link between raw data, analytical environment and resulting product data – we try to maintain traceability of data from its source, storage, through individual stages of analysis to the final product used by the client, so that we can trace an error if necessary.
5. Principle 5: We manage versions of data and analytical code – we can easily eliminate problems that arise when changing input data, algorithm parameters.
6. Principle 6: We consolidate the knowledge of the team and support project management by releasing versions - data is developed during the project based on client requirements, individual stable versions allow us to return to a stable version covered by tests, from which we can bounce back in search of a new solution.
7. Principle 7: We prefer to create analytical code that runs completely – searching for a suitable model requires constant changes in the code, so it is more certain to constantly verify that it works completely, so we prefer code testing in the project, which is analogous to integration tests.

Exploratory Data Analysis Methods

Chapter **5**

5.1 EDA automation

5.1.1

We will summarize in this chapter the techniques we will often use in the data understanding phase. By applying them, we can identify potential problems that could prevent us from arriving at meaningful and useful findings during the modelling phase. We will deal with the elimination of duplicate values, incorrect data types, incorrect and missing data and introduce techniques that eliminate these problems.

However, we should emphasize right away that we cannot apply these techniques mechanically, without deeper consideration, because in some cases we are interested in these special cases and eliminating them from the data set would deprive us of the possibility of their research. In addition, we often cooperate at this stage with the client, the originator of the data, who is able to explain to us in detail what the individual measured characteristics represent and what the individual values, at first glance perhaps erroneous, express.

Exploratory data analysis (EDA) represents a sequence of methods for analyzing the examined data set and summarizing its basic characteristics. Its purpose is to:

- discover patterns in a data file,
- understand the relationship between independent and dependent variables,
- find out how important each independent variable is,
- identify anomalies,
- formulate hypotheses,
- verify assumptions.

In order to examine the dataset properly, we can gradually re-establish a checklist of activities or a process framework that we will adhere to. In this framework, we will use techniques to prepare data at the level of rows (add and remove), columns (add and remove) as well as individual values (value change). At the same time, we can apply techniques leading to a reduction in the number of input independent variables, the so-called dimensionality reduction.

5.1.2

EDA Automation

If we are using Python for exploratory data analysis, we often notice, that Exploratory data analysis can be boring and time-consuming. Moreover, we are not able to easily prepare all the tables, charts, and diagrams, which provide potentially interesting view on the dataset. Therefore, we always appreciate an option to automatize some repetitive tasks and focus directly on data understanding and pre-processing.

There are many examples of Python libraries for EDA automation, like Sweetviz, Pandas-profiling, D-tale, Autoviz and many others. We can try to shorten the whole

process using these libraries and examine to what extent the displayed analysis will be sufficient for us and which adjustments to the file we will still have to make.

The steps required for their use in our project, are almost identical. First, we install the mentioned library with the already known command.

```
pip install sweetviz
```

or directly in the Python Notebook environment.

```
!pip install sweetviz
```

Then we import the Pandas library and load the source data file.

```
import pandas as pd
df = pd.read_csv('grades.csv')
```

The most important method of the Sweetviz library is the Analyze() method, which helps us to analyze the input data frame and display the results in the form of an html file.

```
import sweetviz as sv
znamky_report = sv.analyze(df)
znamky_report.show_html('grades.html')
```

The result of the rapid analysis can be as follows (picture).

We can also use this library to compare multiple data frames using the Compare() method. In the case of comparing training and test data, we will use the method Comapre_intra(), but it is not yet possible to use the usual method of random division of the studied data set. Therefore, we must divide them before comparing them with this library.

DataFrame

NO COMPARISON TARGET

ROWS	62
DUPPLICATES	0
FEATURES	7
CATEGORICAL	1
NUMERICAL	6
TEXT	0

Associations

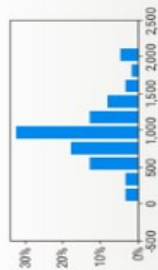
DataFrame

pristupy

VALUES: 62 (100%)
MISSING: ---
DISTINCT: 60 (97%)

MAX 2,135
Q3 1,752
AVG 1,178
MEDIAN 585
Q1 692
5% 366
MIN 28

RANGE 2,107
ICR 487
STD 415
VAR 172k
KURT 0.985
SKEW 0.561
SUM 61,073

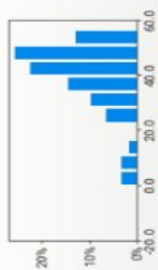


testy

VALUES: 62 (100%)
MISSING: ---
DISTINCT: 39 (63%)

MAX 57.0
Q3 54.5
AVG 49.0
MEDIAN 43.5
Q1 40.1
5% 35.2
MIN 7.5

RANGE 57.0
ICR 13.8
STD 13.0
VAR 169
KURT 2.24
SKEW -1.50
SUM 2,487

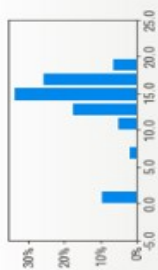


skuska

VALUES: 62 (100%)
MISSING: ---
DISTINCT: 13 (21%)

MAX 20.0
Q3 17.9
AVG 16.0
MEDIAN 14.0
Q1 13.2
5% 0.0
MIN 0.0

RANGE 20.0
ICR 3.00
STD 4.88
VAR 23.8
KURT 2.92
SKEW -1.62
SUM 817

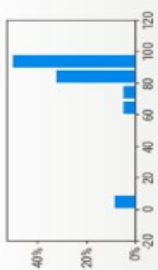


projekt

VALUES: 62 (100%)
MISSING: ---
DISTINCT: 58 (94%)

MAX 99.5
Q3 93.7
AVG 89.6
MEDIAN 81.2
Q1 62.1
5% 0.0
MIN 0.0

RANGE 99.5
ICR 11.6
STD 25.5
VAR 651
KURT 6.20
SKEW 5.036

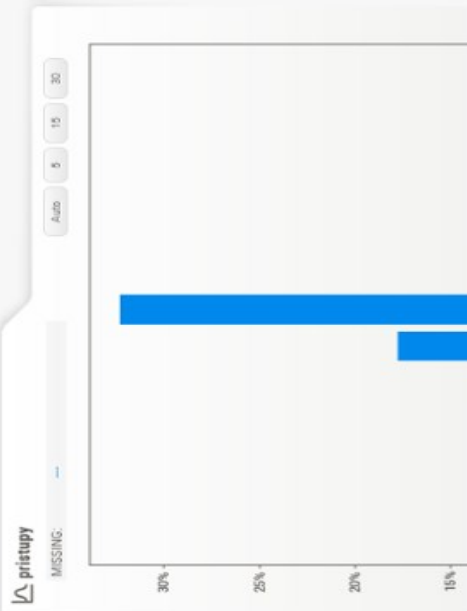
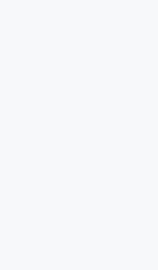


zadania

VALUES: 62 (100%)
MISSING: ---
DISTINCT: 58 (94%)

MAX 99.5
Q3 93.7
AVG 89.6
MEDIAN 81.2
Q1 62.1
5% 0.0
MIN 0.0

RANGE 99.5
ICR 11.6
STD 25.5
VAR 651
KURT 6.20
SKEW 5.036



NUMERICAL ASSOCIATIONS

(PEARSON, -1 to 1)

zadania	0.63
vysledne_body	0.54
testy	0.45
projekt	0.43
skuska	0.42
pristupy	0.00

CATEGORICAL ASSOCIATIONS

(CORRELATION RATIO, 0 to 1)

vysledna_znamka	0.57
-----------------	------

MOST FREQUENT VALUES

1162	2	3.2%
996	2	3.2%
1021	1	1.6%
799	1	1.6%
942	1	1.6%
695	1	1.6%
173	1	1.6%
1502	1	1.6%
680	1	1.6%
935	1	1.6%
1056	1	1.6%
804	1	1.6%
605	1	1.6%
1256	1	1.6%
673	1	1.6%

SMALLEST VALUES

28	1	1.6%
173	1	1.6%
265	1	1.6%
300	1	1.6%
478	1	1.6%
548	1	1.6%
603	1	1.6%
605	1	1.6%
618	1	1.6%
630	1	1.6%
645	1	1.6%
673	1	1.6%
680	1	1.6%
682	1	1.6%

LARGEST VALUES

2135	1	1.6%
2067	1	1.6%
1941	1	1.6%
1757	1	1.6%
1664	1	1.6%
1537	1	1.6%
1502	1	1.6%
1455	1	1.6%
1354	1	1.6%
1347	1	1.6%
1311	1	1.6%
1290	1	1.6%
1259	1	1.6%
1256	1	1.6%
1216	1	1.6%

5.2 Data analysis methods

5.2.1

Understanding Data

We will start by importing standard libraries for visualization and work with data. We will use a set with partial results of students obtained in a subject that has been used for several years.

```
#visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
file_url = 'https://priscilla.fitped.eu/data/1a/results-16-19.xlsx'
df = pd.read_excel(file_url)
```

The imported file did not require any special modifications. Let's look at the basic characteristics of the imported file.

```
print(df.columns)
```

Program output:

```
Index(['pristupy', 'testy', 'testy_znamka', 'skuska',
      'projekt',
      'projekt_znamka', 'zadania', 'vysledne_body',
      'vysledna_znamka',
      'absolvoval', 'rok', 'prieb_test_sql', 'prieb_test_db',
      'akad_rok'],
      dtype='object')
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
  and should_run_async(code)
df.info()
```

Program output:

```
RangeIndex: 286 entries, 0 to 285
Data columns (total 14 columns):
```

#	Column	Non-Null Count	Dtype
0	pristupy	285 non-null	float64
1	testy	284 non-null	float64
2	testy_znamka	284 non-null	object
3	skuska	285 non-null	float64
4	projekt	285 non-null	float64
5	projekt_znamka	284 non-null	object
6	zadania	286 non-null	float64
7	vysledne_body	286 non-null	float64
8	vysledna_znamka	286 non-null	object
9	absolvoval	286 non-null	int64
10	rok	286 non-null	int64
11	prieb_test_sql	220 non-null	float64
12	prieb_test_db	220 non-null	float64
13	akad_rok	286 non-null	object

dtypes: float64(8), int64(2), object(4)
memory usage: 31.4+ KB
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
and should_run_async(code)
print(df.head())

Program output:

	pristupy	testy	testy_znamka	skuska	projekt
projekt_znamka					
0	79.0	0.00	FX	0.0	0.00
FX	12.00				
1	79.0	0.00	FX	0.0	0.00
FX	12.00				
2	792.0	40.60	E	7.0	76.89
A	13.45				
3	1111.0	42.76	D	10.0	79.29
A	11.77				
4	487.0	47.20	C	0.0	0.00
FX	12.58				
	vysledne_body	vysledna_znamka	absolvoval	rok	
prieb_test_sql					

```

0          20.52          FX          0  2016
NaN
1          20.52          FX          0  2016
NaN
2         140.73          D          1  2016
NaN
3         150.36          C          1  2016
NaN
4          68.68          FX          0  2016
NaN

    prieb_test_db    akad_rok
0          NaN  2016/2017
1          NaN  2016/2017
2          NaN  2016/2017
3          NaN  2016/2017
4          NaN  2016/2017
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)

```

We can see that there are numeric and text (categorical) variables in the file. Each line of the file corresponds to the partial results of the student, which are presented mainly in numerical form, supplemented by a grade. In addition, the file contains information about the academic year, the number of accesses to the e-learning course, which formed the support for teaching. Let's examine columns with numeric values.

```
print(df.describe())
```

Program output:

```

          pristupy          testy          skuska          projekt
zadania \
count  285.000000  284.000000  285.000000  285.000000
286.000000
mean    664.985965    35.277782    11.150877    58.954982
18.001573
std     365.434942    15.844801     6.700986    32.654879
10.796514

```

```

min      13.000000      0.000000      0.000000      0.000000
0.000000
25%      440.000000     31.542500      8.000000     57.680000
11.000000
50%      598.000000     39.200000     14.000000     73.330000
14.500000
75%      837.000000     45.500000     16.000000     81.090000
24.000000
max      2392.000000     57.000000     20.000000     90.000000
40.000000

      vysledne_body  absolvoval      rok  prieb_test_sql
prieb_test_db
count      286.000000   286.000000   286.000000     220.000000
220.000000
mean      126.108392    0.744755  2017.678322     13.277273
11.859091
std        55.605239    0.436763    1.152276     7.073355
6.500661
min         0.000000    0.000000  2016.000000     0.000000
0.000000
25%      111.517500    0.000000  2017.000000     10.000000
8.000000
50%      146.990000    1.000000  2018.000000     15.000000
14.000000
75%      165.125000    1.000000  2019.000000     18.000000
17.000000
max      201.920000    1.000000  2019.000000     26.000000
20.000000
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)

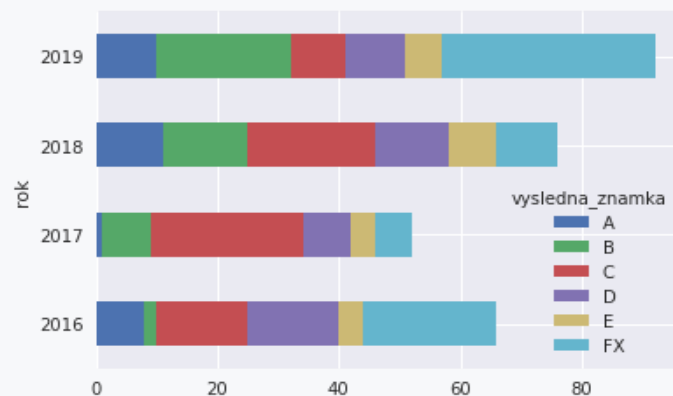
```

The most interesting is the distribution of the resulting grades by year. To view them, we can directly use the pivot chart available in the Pandas library.

```
pd.crosstab(df.rok, df.vysledna_znamka).plot.barh(stacked =
True)
```

Program output:

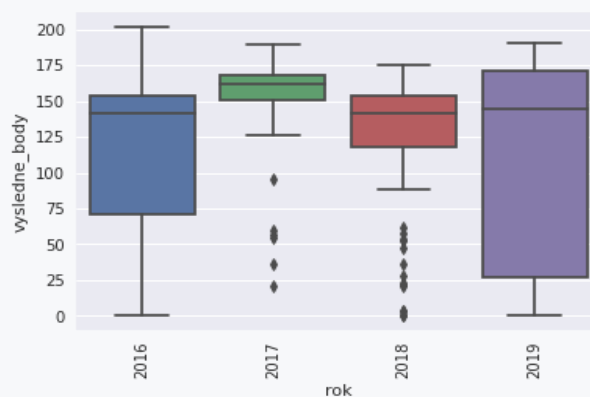
```
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```



```
sns.boxplot(x = 'rok', y = 'vysledne_body', data=df)
plt.xticks(rotation=90)
```

Program output:

```
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```



More accurate information about the results can be obtained from a numerical representation of the resulting points from the subject using a graph of type *boxplot()*.

5.2.2

Duplicate values

Detecting the proportion, or elimination, of duplicate values is a frequent step in data preprocessing. Once again, we will use the pre-made methods of the Pandas package.

```
# prepared code
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
file_url = 'https://priscilla.fitped.eu/data/1a/results-16-19.csv'
df = pd.read_csv(file_url)

print(df.duplicated().head())
```

Program output:

```
0    False
1     True
2    False
3    False
4    False
dtype: bool
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

We can see that the following method only indicates whether a record is a duplicate or not, treating the first occurrence of the record as the original, the next as its copies. Let's get to a more useful look at duplicate records. First, let's count them. There are 5 duplicate records in the examined file.

```
print(df.duplicated().sum())
```

Program output:

```
5
```

```
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

We display duplicate records using the fact that we can use duplicate validation as a condition. In the statement that appears, we see records that have duplicates in the file. We don't see all their occurrences. Note that records 214 and 224 are exactly the same, and thus there is also a third identical record in the file.

```
print(df[df.duplicated()].head())
```

Program output:

	pristupy	testy	testy_znamka	skuska	projekt
projekt_znamka	zadania				
1	79.0	0.0	FX	0.0	0.00
FX	12.0				
151	89.0	0.0	A	0.0	0.00
FX	2.0				
186	790.0	40.5	E	10.0	84.34
A	14.0				
214	635.0	28.0	FX	13.0	83.65
A	12.0				
224	635.0	28.0	FX	13.0	83.65
A	12.0				

	vysledne_body	vysledna_znamka	absolvoval	rok
prieb_test_sql				
1	20.52	FX	0	2016
NaN				
151	3.60	A	1	2018
0.0				
186	134.02	D	1	2018
17.5				
214	114.06	FX	0	2019
10.0				
224	114.06	FX	0	2019
10.0				

	prieb_test_db	akad_rok	Unnamed: 14	Unnamed: 15
1	NaN	2016/2017	NaN	NaN

```

151          0.0  2018/2019          NaN          NaN
186         13.0  2018/2019          NaN          NaN
214          5.0  2019/2020          NaN          NaN
224          5.0  2019/2020          NaN          NaN
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)

```

If we are only interested in a certain subset of rows or columns and the occurrence of duplicates in them, we can edit our expression further. To do this, we will use the already known *methods of loc* and *iloc*. If we use them without a list of columns, the result will be the same.

```
print(df.loc[df.duplicated()].head())
```

Program output:

```

      pristupy  testy testy_znamka  skuska  projekt
projekt_znamka  zadania  \
1          79.0    0.0          FX    0.0    0.00
FX          12.0
151         89.0    0.0          A    0.0    0.00
FX          2.0
186        790.0   40.5          E   10.0   84.34
A          14.0
214        635.0   28.0          FX   13.0   83.65
A          12.0
224        635.0   28.0          FX   13.0   83.65
A          12.0

      vysledne_body vysledna_znamka  absolvoval  rok
prieb_test_sql  \
1          20.52          FX          0  2016
NaN
151          3.60          A          1  2018
0.0
186        134.02          D          1  2018
17.5
214        114.06          FX          0  2019
10.0

```



```

224          114.06          FX          0  2019
10.0

      prieb_test_db   akad_rok  Unnamed: 14  Unnamed: 15
1          NaN  2016/2017          NaN          NaN
151         0.0  2018/2019          NaN          NaN
186        13.0  2018/2019          NaN          NaN
214         5.0  2019/2020          NaN          NaN
224         5.0  2019/2020          NaN          NaN
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)

```

We are interested in whether there are duplicates based on the equality of the data in the following four columns.

```

print(df.loc[df.duplicated(), ['pristupy', 'testy',
'testy_znamka', 'rok']])

```

Program output:

```

      pristupy  testy  testy_znamka   rok
1         79.0    0.0          FX  2016
151        89.0    0.0           A  2018
186       790.0   40.5           E  2018
214       635.0   28.0          FX  2019
224       635.0   28.0          FX  2019
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)

```

If we need to mark as duplicate other than the second and next occurrences of a record, we can use the *keep* parameter. To keep the last record, we use the *last* value, to indicate all duplicate records, we use the value *False*. Although the records in the following list look the same, note a different index for the record.

```
print(df.loc[df.duplicated(keep='last'), ['pristupy', 'testy',
'testy_znamka', 'rok']].head())
```

Program output:

```
      pristupy  testy  testy_znamka   rok
0         79.0    0.0             FX  2016
150        89.0    0.0             A  2018
185       790.0   40.5             E  2018
213       635.0   28.0            FX  2019
214       635.0   28.0            FX  2019
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

Using the *subset* parameter of the *duplicated()* method, we can examine the duplication of records only in selected attributes (columns) directly. For example, we can find out how many duplicates a data set contains if we examine only the following columns.

```
print(df.duplicated(subset=['pristupy', 'testy',
'testy_znamka', 'rok'], keep='first').sum())
```

Program output:

```
6
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

At this point, we should find out the reason for the duplication, whether these records were created in real conditions or it is an import or manual preprocessing error. In this activity, important information is often given to us by a client, programmer or collaborator who was directly involved in the preparation of the file. Finally, after sufficient consideration of what the found duplicates really mean, we can use the *drop_duplicates* method to remove them from the examined dataset.

```
df2 = df.drop_duplicates(keep='first').sample(10)
```

The resulting *df* data frame contains only unique records. If this frame is to replace the original, we must use the *inplace=True* parameter. The following commands will verify that this is indeed the case.

```
print(df2.duplicated().sum())
```

Program output:

```
0
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

```
print(df2[df2.duplicated()].head())
```

Program output:

```
Empty DataFrame
Columns: [pristupy, testy, testy_znamka, skuska, projekt,
projekt_znamka, zadania, vysledne_body, vysledna_znamka,
absolvoval, rok, prieb_test_sql, prieb_test_db, akad_rok,
Unnamed: 14, Unnamed: 15]
Index: []
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

We can see that the table of duplicates is empty.

5.2.3

Change data types

Another relatively common problem encountered in the data preprocessing phase is the problem of incorrect data types of individual attributes of the original data set. In

the Pandas package, we have several methods available to help us solve this problem. First, let's display the data types for each attribute using the *dtypes* data frame attribute.

```
# prepared code
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
file_url = 'https://priscilla.fitped.eu/data/1a/results-16-19.csv'
df = pd.read_csv(file_url)

print(df.dtypes)
```

Program output:

```
pristupy          float64
testy             float64
testy_znamka      object
skuska           float64
projekt          float64
projekt_znamka    object
zadania          float64
vysledne_body     float64
vysledna_znamka   object
absolvoval        int64
rok              int64
prieb_test_sql    float64
prieb_test_db     float64
akad_rok          object
Unnamed: 14       float64
Unnamed: 15       float64
dtype: object
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

We see that most columns have numeric data types, which is beneficial for most machine learning algorithms. If we need to change the data type of an attribute, we

can define its new type already at the data retrieval stage or retype the attribute later. Let's consider both procedures in more detail.

In the first case, we specify the data type using the *dtype* parameter, which expects the *dictionary* data type, where the column names represent the key and their data type represents the value, for example, {'col1':np.float64, 'col2':np.int32}.

```
df = pd.read_csv(file_url, dtype={'vysledna_znamka':  
'category'})  
print(df.dtypes)
```

Program output:

```
pristupy          float64  
testy             float64  
testy_znamka      object  
skuska           float64  
projekt          float64  
projekt_znamka    object  
zadania          float64  
vysledne_body     float64  
vysledna_znamka   category  
absolvoval        int64  
rok              int64  
prieb_test_sql   float64  
prieb_test_db    float64  
akad_rok         object  
Unnamed: 14       float64  
Unnamed: 15       float64  
dtype: object  
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:  
DeprecationWarning: `should_run_async` will not call  
`transform_cell` automatically in the future. Please pass the  
result to `transformed_cell` argument and any exception that  
happen during the transform in `preprocessing_exc_tuple` in  
IPython 7.17 and above.  
and should_run_async(code)
```

In the latter case, we use the equally intuitive *astype()* method, which returns a new column (of type *series*). We need to reassign it to the original attribute. The attribute *vysledna_znamka* thus acquires a data type that clearly defines that it is a categorical variable.

```
df['vysledna_znamka'] =  
df['vysledna_znamka'].astype('category')  
print(df.dtypes)
```

Program output:

```
pristupy          float64
testy             float64
testy_znamka      object
skuska           float64
projekt          float64
projekt_znamka    object
zadania          float64
vysledne_body     float64
vysledna_znamka   category
absolvoval        int64
rok              int64
prieb_test_sql    float64
prieb_test_db     float64
akad_rok          object
Unnamed: 14       float64
Unnamed: 15       float64
dtype: object
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

Data type *category* represents effective work with categorical variables, contains a finite number of values unlike data type *object*. If we want to see them, we will use the *cat.categories* attribute. In the case of the examined attribute *vysledna_znamka*, we can see that this column contains correct values of marks, always written in the same form. If there were typos or another form of grades in this column, we would identify the problem in this view.

```
print(df['vysledna_znamka'].cat.categories)
```

Program output:

```
Index(['A', 'B', 'C', 'D', 'E', 'FX'], dtype='object')
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

Similarly, we can retype all *object* attributes to data *category* if they contain a finite number of values that we plan to work with further. We select all attributes of this data type and display unique values in the cycle.

```
obj_df = df.select_dtypes(include='object')
obj_cols = obj_df.columns
for col_name in obj_cols:
    print(col_name)
    print(df[col_name].unique())
```

Program output:

```
testy_znamka
['FX' 'E' 'D' 'C' 'B' 'A' nan]
projekt_znamka
['FX' 'A' 'C' 'B' 'D' 'E' nan]
akad_rok
['2016/2017' '2017/2018' '2018/2019' '2018/19' '2019/2020'
'2019/20020']
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

There are three columns in the examined dataset with the potential to make them a categorical data type. So now we are going to retype them en masse using a simple cycle.

```
for col_name in obj_cols:
    df[col_name] = df[col_name].astype('category')
print(df.dtypes)
```

Program output:

pristupy	float64
testy	float64
testy_znamka	category
skuska	float64
projekt	float64
projekt_znamka	category
zadania	float64
vysledne_body	float64
vysledna_znamka	category

```
absolvoval          int64
rok                 int64
prieb_test_sql      float64
prieb_test_db       float64
akad_rok            category
Unnamed: 14          float64
Unnamed: 15          float64
dtype: object
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

5.2.4

Edit incorrect values

Modifying incorrect values, or unifying them into a suitable form, is another common problem that we have to solve in the pre-processing phase. Otherwise, we may have, for example, multiple categories of values in the input data set that actually express the same thing, i.e., one category. For example, if we need to correct an incorrectly written value, we proceed as follows:

- We create a new helper column, which will be a copy of the column where the incorrect value is located.
- We use *loc* to identify records where the problem occurred.
- We assign it a new value.
- To replace a value that is part of a more complex text string, use *str.contains()* to find all occurrences.
- Then we use the *str.replace()* method to replace it with the desired value.

We have already examined the grade columns in the previous chapter. Let's now examine column *akad_rok*, where there is the greatest assumption that there may be an error in the designation of the academic year.

```
# prepared code
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
file_url = 'https://priscilla.fitped.eu/data/la/results-16-19.csv'
df = pd.read_csv(file_url)
```



```
print(df['akad_rok'].unique())
```

Program output:

```
['2016/2017' '2017/2018' '2018/2019' '2018/19' '2019/2020'
'2019/20020']
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

In the list of unique values, we discovered two different designations for the academic year 2018/2019. At the same time, we found an error in one of the lines from the academic year 2019/2020. At first glance, it is obvious that this is a flawed year 2019/20020.

Let's look for records with an abbreviated version of the 2018/2019 academic year. In the case of the examined file, we find that it is only one record.

```
print(df.loc[df['akad_rok'].str.contains('2018/19',
na=False),])
```

Program output:

```
pristupy  testy testy_znamka  skuska  projekt
projekt_znamka  zadania  \
166      828.0    38.5          E    10.0    57.68
E      24.0

vysledne_body  vysledna_znamka  absolvoval  rok
prieb_test_sql  \
166      129.57          D          1  2018
10.5

prieb_test_db  akad_rok  Unnamed: 14  Unnamed: 15
166      18.0  2018/19          NaN          NaN
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
```

```
and should_run_async(code)
```

We can simply correct this value with `str.replace` and check the result.

```
df['akad_rok'] = df['akad_rok'].str.replace('2018/19',  
'2018/2019')  
print(df['akad_rok'].unique())
```

Program output:

```
['2016/2017' '2017/2018' '2018/2019' '2019/2020' '2019/20020']  
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:  
DeprecationWarning: `should_run_async` will not call  
`transform_cell` automatically in the future. Please pass the  
result to `transformed_cell` argument and any exception that  
happen during the transform in `preprocessing_exc_tuple` in  
IPython 7.17 and above.  
and should_run_async(code)
```

We successfully solved the first problem. We can do the same in the latter case.

```
df.loc[df['akad_rok'].str.contains('2019/20020',  
na=False), ['akad_rok']]  
df['akad_rok'] = df['akad_rok'].str.replace('2019/20020',  
'2019/2020')  
print(df['akad_rok'])
```

Program output:

```
0      2016/2017  
1      2016/2017  
2      2016/2017  
3      2016/2017  
4      2016/2017  
...  
281    2019/2020  
282    2019/2020  
283    2019/2020  
284    2019/2020  
285    2019/2020  
Name: akad_rok, Length: 286, dtype: object  
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:  
DeprecationWarning: `should_run_async` will not call  
`transform_cell` automatically in the future. Please pass the  
result to `transformed_cell` argument and any exception that
```

```
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

5.2.5

Missing values

A frequent problem of input data is missing values. In the pre-processing phase, we should assess whether their absence means that we delete the entire recording/observation or try to preserve the recording and replace the missing value with a suitable alternative. Let's look at our practice file to see if it contains missing values.

```
# prepared code
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
file_url = 'https://priscilla.fitped.eu/data/1a/results-16-19.csv'
df = pd.read_csv(file_url)

print(df.isna().sample(10))
```

Program output:

	pristupy	testy	testy_znamka	skuska	projekt
projekt_znamka	zadania	\			
53	False	False	False	False	False
False	False				
60	False	False	False	False	False
False	False				
67	False	False	False	False	False
False	False				
270	False	False	False	False	False
False	False				
138	False	False	False	False	False
False	False				
55	False	False	False	False	False
False	False				
116	False	False	False	False	False
False	False				
32	False	False	False	False	False
False	False				

```

232      False  False      False  False  False
False      False
41      False  False      False  False  False
False      False

```

```

      vysledne_body  vysledna_znamka  absolvoval  rok
prieb_test_sql  \
53      False      False      False  False
True
60      False      False      False  False
True
67      False      False      False  False
False
270      False      False      False  False
False
138      False      False      False  False
False
55      False      False      False  False
True
116      False      False      False  False
False
32      False      False      False  False
True
232      False      False      False  False
False
41      False      False      False  False
True

```

```

      prieb_test_db  akad_rok  Unnamed: 14  Unnamed: 15
53      True      False      True      True
60      True      False      True      True
67      False      False      True      True
270      False      False      True      True
138      False      False      True      True
55      True      False      True      True
116      False      False      True      True
32      True      False      True      True
232      False      False      True      True
41      True      False      True      True

```

```

/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that

```

```
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

To get a better idea, simply count the missing values by column.

```
print(df.isna().sum())
```

Program output:

```
pristupy          1
testy             2
testy_znamka      2
skuska            1
projekt           1
projekt_znamka    2
zadania           0
vysledne_body     0
vysledna_znamka   0
absolvoval        0
rok               0
prieb_test_sql    66
prieb_test_db     66
akad_rok          0
Unnamed: 14       286
Unnamed: 15       286
dtype: int64
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

```
print(df.shape)
```

Program output:

```
(286, 16)
```

Most columns contain a small number of duplicates. The problem is columns *prieb_test_sql* and *prieb_test_db*. The absence of 66 values in this case means a significant interference with the data set, which contains a total of 286 records. We

therefore need to find out what eliminating them means, or whether it makes sense to replace them with some appropriate value.

So, let's first look at these two problematic columns. We'll view specific records using a condition.

```
print(df[df['prieb_test_sql'].isna()].head())
```

Program output:

```

    pristupy  testy testy_znamka  skuska  projekt
projekt_znamka  zadania \
0      79.0    0.00          FX    0.0    0.00
FX      12.00
1      79.0    0.00          FX    0.0    0.00
FX      12.00
2     792.0   40.60          E    7.0   76.89
A      13.45
3    1111.0   42.76          D   10.0   79.29
A      11.77
4     487.0   47.20          C    0.0    0.00
FX      12.58

    vysledne_body vysledna_znamka  absolvoval  rok
prieb_test_sql \
0          20.52          FX          0  2016
NaN
1          20.52          FX          0  2016
NaN
2         140.73          D          1  2016
NaN
3         150.36          C          1  2016
NaN
4          68.68          FX          0  2016
NaN

    prieb_test_db  akad_rok  Unnamed: 14  Unnamed: 15
0          NaN  2016/2017          NaN          NaN
1          NaN  2016/2017          NaN          NaN
2          NaN  2016/2017          NaN          NaN
3          NaN  2016/2017          NaN          NaN
4          NaN  2016/2017          NaN          NaN
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the

```

```
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

The values in columns *prieb_test_sql* and *prieb_test_db* cannot be easily added as these are test results. Presumably, the tests were not conducted this year or were conducted in another form. To delete records with missing values, we can use the *dropna()* method, which returns a new dataframe without rows that contained the missing value somewhere. However, this can have a negative impact on the size of the input data file, so we should carefully consider this simple option.

If we are only interested in a specific attribute, we define it in the *subset parameter*. In addition, if we want to definitively remove these values from the original data frame, we add the *inplace=True* parameter.

```
df.dropna(subset=['prieb_test_sql'], inplace=True)
print(df.isna().sum())
```

Program output:

```
pristupy          1
testy             2
testy_znamka      2
skuska            1
projekt           1
projekt_znamka    2
zadania           0
vysledne_body     0
vysledna_znamka   0
absolvoval        0
rok               0
prieb_test_sql    0
prieb_test_db     0
akad_rok          0
Unnamed: 14       220
Unnamed: 15       220
dtype: int64
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

We see a different situation with records of grades. We don't want to remove these rows with a missing value, but we want to add a value that better describes why the value wasn't entered or their purpose. We will use the *fillna()* method.

```
df['testy_znamka'].fillna('FX', inplace=True)
df['projekt_znamka'].fillna('FX', inplace=True)
print(df.isna().sum())
```

Program output:

```
pristupy          1
testy             2
testy_znamka      0
skuska            1
projekt           1
projekt_znamka    0
zadania           0
vysledne_body     0
vysledna_znamka   0
absolvoval        0
rok               0
prieb_test_sql    0
prieb_test_db     0
akad_rok          0
Unnamed: 14        220
Unnamed: 15        220
dtype: int64
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

If we know that the input file contains a different label for missing values than NaN or an empty string that is replaced with NaN, we can assign this label during data import using the *na_values* parameter in the *read_csv* or *read_excel* method.

```
df = pd.read_csv(file_url, na_values='?')
```

In justified cases, we need to keep as many rows in the data set as possible. Therefore, we need to replace the missing numerical values with an appropriate value that does not affect further data modeling, such as the median. In this case, we identify all the necessary attributes of the data frame with numerical values and use a combination of methods already known to us.


```
num_df = df.select_dtypes(include='number')
num_cols = num_df.columns
print(num_cols)
```

Program output:

```
Index(['pristupy', 'testy', 'skuska', 'projekt', 'zadania',
      'vysledne_body',
      'absolvoval', 'rok', 'prieb_test_sql', 'prieb_test_db',
      'Unnamed: 14',
      'Unnamed: 15'],
      dtype='object')
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

To inspire how a routinely repetitive process can be speeded up, we adjust individual numeric columns in one cycle.

```
for col_name in num_cols:
    col_median = df[col_name].median()
    df[col_name].fillna(col_median, inplace=True)
    print(col_name)
    print(col_median)
```

Program output:

```
pristupy
598.0
testy
39.2
skuska
14.0
projekt
73.33
zadania
14.5
vysledne_body
146.99
absolvoval
1.0
```

```

rok
2018.0
prieb_test_sql
15.0
prieb_test_db
14.0
Unnamed: 14
nan
Unnamed: 15
nan
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
/home/johny/.local/lib/python3.9/site-
packages/numpy/lib/nanfunctions.py:1216: RuntimeWarning: Mean
of empty slice
    return np.nanmean(a, axis, out=out, keepdims=keepdims)
/home/johny/.local/lib/python3.9/site-
packages/numpy/lib/nanfunctions.py:1216: RuntimeWarning: Mean
of empty slice
    return np.nanmean(a, axis, out=out, keepdims=keepdims)

```

The missing values in each column have been replaced with the following median values.

Finally, we check if we have successfully resolved all missing values.

```
print(df.isna().sum())
```

Program output:

```

pristupy      0
testy         0
testy_znamka   2
skuska        0
projekt       0
projekt_znamka 2
zadania       0
vysledne_body 0
vysledna_znamka 0
absolvoval    0

```

```
rok          0
prieb_test_sql  0
prieb_test_db  0
akad_rok      0
Unnamed: 14    286
Unnamed: 15    286
dtype: int64
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

Data Preprocessing Methods

Chapter 6

6.1 Methods

6.1.1

Data preparation is an important step in any knowledge mining process from data. In the educational domain, it is particularly important to obtain suitable, sufficiently large, and representative datasets, which often requires extra effort.

Data preparation is an iterative process involving data cleansing, merging, transforming, selecting or reducing variables in order to uncover hidden patterns of behavior and structure. This process must exclude unnecessary data without impoverishing the data set or introducing an unwanted bias. The output from this phase is a pre-processed dataset, which is considered a suitable input to the modelling phase (Mubarak et al., 2020).

6.1.2

Data Inconsistency and Data Type Conversions

We can start with relatively common problem encountered in the data preprocessing phase. This problem relates to data inconsistency and incorrect data types of individual attributes of the original data set.

Inconsistent data appears when a dataset or data group differs dramatically from a similar dataset (a conflicting dataset) for no apparent reason. Data in a single column must have consistent formats. For example, when datasets from multiple sources are merged, the same data may have different formats. Dates can often be problematic. A column of data cannot have a date format such as mm/dd/yy and mm/dd/y. Data must be corrected to have consistent formats.

In fact, some seemingly incorrect data may also result from inconsistencies in the naming conventions used or established data codes, or from the use of inconsistent input field formats. For example, duplicate tuples may require data cleaning, e.g., age = "42" and birthday = "03/03/1997" indicating discrepancies between duplicate entries (Siegel, 2018).

Many inconsistencies come to the place when we must combine different data resources. In this case, data engineering methods and techniques can help us to prepare the dataset to the more suitable form required by the next steps of data-preprocessing.

6.1.3

Missing Values

When we analyze data, it is important to first determine the pattern of missing data. There are three types of patterns:

- missing completely randomly (MCAR),
- randomly missing (MAR),
- missing not randomly (MNAR).

Data is missing completely randomly when there is no pattern in the missing data in any variable. Random missing occurs when there is a pattern in the missing data that can affect the primary dependent variables. An example of MAR would be women's propensity not to tell their age. For example, if the study was about weight loss. If heavier individuals responded less, it would affect the results. Missing values can lead to analysis problems with predictable modeling depending on the type of model used (Luna et al., 2017).

There are two strategies for resolving missing values, namely deleting records or deleting a column and deriving the missing values. In the first case, deletion involves deleting a row (record) from the dataset. If there are only a few missing values, this may be an appropriate approach.

6.1.4

On the other hand, a smaller dataset can weaken the predictive power of a model. Deleting a column removes any variables that contain the missing values. Deleting a variable that contains only a few missing values is not recommended. The second and more advantageous strategy is to derive the missing value from the other existing values of the variables. This adjustment changes the missing data value to one that represents a logically reasoned value (Siegel, 2018).

The most used method is to replace the missing value with another constant value. Typically, values expressed as Null are replaced by a constant value. However, this can be problematic, for example, replacing age with 0 does not make sense. Likewise, for categorical variables such as gender, replacing the missing value with a constant such as F leads to fundamental changes. This method works well when the missing value is completely random (MCAR).

It is also possible to replace the missing numeric values with a mean or median. Replacing missing values with the mean of a variable is a common and simple method. Moreover, is unlikely it impairs the predictability of the model since the average values should be close to the average. However, if many values are missing for a particular variable, replacing it with mean can cause problems because a higher average value will cause changes in the variable's distribution and a smaller standard deviation. If this is the case, replacing with a median may be a better solution.

We can replace categorical values with modus (the most common value) because there is no mean or median. We could also replace numerical missing values with the most common values.

The last method of replacing missing values is to replace the missing value by randomly selecting a value from the missing values, the so-called custom distribution. This solution is preferred to using the average, but it is not so simple.

For the sake of completeness, let us add that in the case of large data sets, we can replace the missing values with values obtained by the application of some machine learning method or neural networks.

6.1.5

Outliers

An extreme value, outlier, is a data value that is scattered from other data values in the dataset. The degree of variability tells us how close or distant the values of the variables of a set are from each other.

An example of measures of variability is the interquartile range. The interquartile range, also referred to as the Interquartile Range (IQR), represents the difference between the third and first quartiles (i.e., between 75. and the 25th percentile). Thus, it represents an area of values that has a mean 50% of the values of the variable. This degree of variability is not affected by extreme values and can be expressed as follows:

$$IQR = q_3 - q_1$$

We can also use the interquartile range to identify outliers, to create a boundary outside q_1 and q_3 . Any values less than 1.5 IQR below q_1 or greater than 1.5 IQR above q_3 are considered outliers.

Outliers can be visually identified by creating histograms, a box chart, and looking for values that are too high or too low.

There are five common ways to process outliers (Edwards, 2018):

1. Remove outliers from modelling data.
2. Separate outliers and create separate models.
3. transform outliers so that they are no longer outliers.
4. Group data.
5. Leave outliers in the data.

Many knowledge discovery algorithms attempt to minimize the impact of outliers on the final model or eliminate them during the preprocessing stages. For example, in the case of a one-dimensional value, we can prevent outliers from distorting the results by using the median. It is also important to emphasize that although outliers can usually be caused by noise, they can often be true observations in educational data.

6.1.6

For example, we often meet exceptional students who succeed with little effort or, conversely, fail despite all expectations. However, distinguishing between outliers that should be taken into account and those that should be eliminated is not always

easy. The right decision requires knowledge of the area in which the data was collected, and at the same time depends on the objectives of the analysis. In this case, a relatively simple technique, based on the visualization of data quartiles, where values that do not fall within the identified quartiles can be considered outliers.

Omitting outliers changes data variability. Before automatically removing outliers, it is advisable to investigate the possible causes of outliers and the impact that outliers may have on the analysis. If the outlier value is caused by incorrectly entered or measured data values, or if the outlier value is outside the interest file, it should be removed. If extreme value is unlikely to distort the model, in other words, cause more harm than good, the outlier value should equally be eliminated (Siegel, 2018).

Some analysts remove outliers and create a separate model. Decision trees in this case can help determine whether extreme values are good predictors of the target variable (Kusner et al., 2017).

On the other hand, there may be cases where outliers represent the group of data to be studied and should remain as they are in the dataset. Thus, there is no universal method for manipulating outlier values, we should always consider them in the context of the problem under study (Peña-Ayala, 2017).

6.2 Data transformation and reduction

6.2.1

Data transformation

Data transformation can facilitate a better interpretation of discovered knowledge by allowing variables or data from different sources to be compared. We will now give some examples of transformation, such as normalization, standardization, discretization, and class inequality.

Machine learning algorithms are effectively trained when datasets are normalized or standardized, resulting in faster processing or training. Normalization is a data transformation technique where variable values are scaled within a specified range, usually from -1.0 to 1.0 or between 0.0 and 1.0. Within a variable there is often a large difference between the maximum and minimum values, e.g., 0.01 and 1,000. For example, this difference could significantly affect the performance of some machine learning algorithms. Therefore, it is necessary to perform normalization and rescale the original values to values from the same defined interval. In this way, normalization can improve the accuracy and efficiency of the knowledge discovery algorithms used, including distance measurements.

Normalization also helps prevent variables with initially large ranges from prevailing over variables with originally smaller ranges. For example, one of the most important steps in data preprocessing for clustering is to standardize or normalize data to avoid clusters dominated by variables of great variability (Charitopoulos et al., 2020).

6.2.2

There are many other ways to normalize data. In education, however, the most used method is Min-max normalization, which performs a linear transformation of the original data.

$$X_{norm} = (X - X_{min}) / (X_{max} - X_{min})$$

Standardization is another data transformation technique where values are centered around an average with a unit standard deviation.

$$X_{stand} = (X - \text{priemer}) / \text{Std}$$

It means, the average of the variable becomes zero, and the resulting distribution has a unit standard deviation. The most widely used method of standardization in education is z-score. When choosing scaling methods, it's a good idea to follow the recommendations associated with using a particular machine learning method in the modeling process.

Normalization

- For scaling, the minimum and maximum values of variables are used.
- It is used when variables are of different ranges.
- Variable values are usually adjusted between [0.1] or [-1.1].
- It is affected by outlier values.
- Python provides the Scikit-Learn library with the MinMaxScaler function.
- The distribution of elements is unknown.

Standardization

- Average and standard deviation are used for scaling.
- It is used when it is necessary to ensure zero mean and unit standard deviation.
- It is not bounded by a specific range.
- There is much less influence in outlier values.
- Python provides a Scikit-Learn library with StandarScaler.
- The distribution of elements has a normal or Gauss distribution.

Discretization divides numerical data into categorical classes that are more user-friendly than exact quantities and ranges. This reduces the number of possible continuous character values and provides a much clearer view of the data. Discretization generally smooths out the impact of noise and allows for simpler models that are less prone to overlearning. For example, algorithms for obtaining association rules usually work only with categorical data. A special type of discretization is the transformation of ordinal to binary representation, that is, from numbers indicating position in the sequence to a value of 0 or 1. This type of codification is used, for example, in the discovery of patterns.

6.2.3

A different approach is to use fuzzy intervals, in which fuzzy sets are used instead of sharp intervals. Finally, the most extreme but simplest case of discrimination is when we use a binary variable (0 or 1). In this case, even if some information is lost, the resulting model can yield, for example, a more accurate classification.

Some other authors suggest making log file files more expressive. This change involves changing the representation of events in the log file and enriching the recorded expressions with semantics so that further conclusions useful for the field of education can be drawn more easily. Another approach is to fully represent each event using English words, English grammar, and standard log file forms, such as xAPI (Torrance, Houck, 2017).

An unbalanced dataset is also a common problem with educational data. For example, if you create a model that helps identify at-risk students where there are 1,000 students available, 900 of whom are successful (higher-represented class) and another 100 fail (underrepresented class), machine learning algorithms will tend to gravitate toward a higher-represented class. The accuracy of the models will reach 90% in such a case, since 900 successful students were available. However, it will not be possible to create a model that would be able to reliably predict at-risk students, since they make up only 10% of the input set. In this case, subsampling or resampling methods shall be used.

The purpose of under-sampling is to reduce a class with higher abundance. There are two ways. The first method randomly deletes some records of a class with a higher representation, which is known as random under-sampling. The second method uses statistical methods to remove a class with a higher-class representation, known as informed under-sampling. Under-sampling methods are generally not preferred because there is a possibility of losing valuable information.

Oversampling is the opposite of under-sampling. The aim is to increase the number of samples of a class with a lower representation so that the classes become equivalent. Unlike subsampling, which deletes datasets when you reload, new data is added to the dataset. Again, two methods are available: random or synthetic up-sampling. In case of random up-sampling, data from the under-represented class shall be replicated. The synthetic up-sampling technique generates artificial samples. New samples are created by adding relevant information to a less represented class, while avoiding misclassification. The most widely used resampling method is the Synthetic Minority Oversampling Technique (SMOTE), which generates synthetic data points for a lower-represented class that tries to track as closely as possible the distribution of variables in a dataset (Fernandes et al., 2019).

6.2.4

Data Reduction

The data reduction step consists of aggregation, selection, extraction of attributes, is an activity where a subset of relevant variables is selected from all available variables. The main goal of data aggregation is to group it. It is the process of applying statistical metrics such as mean, median, and variance that are necessary to summarize the data and replace several elementary values with one placeholder (Charitopoulos et al., 2020).

Selecting the right variables is one of the main tasks to perform before applying knowledge discovery techniques, since variables can correlate with each other or be redundant. As a result, data must be preprocessed to represent an appropriate subset of variables and ignore irrelevant and redundant. It is a redundant variable if it can be derived from another or more variables.

The choice of input variables is very important in education, since in many practical situations there can be a large number of variables that can lead to a decrease in the accuracy of the learning model or a more difficult to meaningfully interpret the results. Therefore, when selecting variables, it is necessary to analyze the dependencies between the output variable (target class) and the explanatory (input variables) through correlation analysis or association rules. Another possible approach is to use a machine learning algorithm and experiment with selecting appropriate variables in the resulting classifier. One solution to this problem is to select only the most important variables using specialized algorithms. For example, a decision tree can be used to determine appropriate characteristics. This method is used to obtain the most consistent variables by presenting them at different levels of the tree (Fernandes et al., 2019).

6.2.5

Learning management systems, such as LMS Moodle, store a huge number of variables about courses, students, and activities. Therefore, it is really important to select only a useful group of variables to reduce the dimensionality of the data. Then these selected variables can all be stored together in a new table containing all relevant information related to the students enrolled in the course. For example, in the case of predicting the final performance of students in a course based on information about the use of the LMS Moodle course, there are many variables characterizing the interaction between students and the Moodle LMS. Therefore, it is only necessary to select variables closely related to student performance.

Some data preparation techniques perform extraction and selection of variables at the same time. For example, in major component analysis (PCA) and independent component analysis (ICA), new variables are created as linear combinations of the original ones. At the same time, these algorithms suggest which of the new variables best captures the original data. The problem is that the appropriateness of extraction and selection of variables cannot be evaluated until the classifier has passed the

learning and evaluation stage. Therefore, in the case of a larger number of variables, not all options can be verified. For this reason, all methods of extraction and selection of functions are heuristic (Lang et al., 2017).

6.2.6

Other preprocessing techniques

User identification, session identification or path reconstruction are examples of data pre-processing techniques required by special learning analytics methods, which often take into account also time.

The user identification is a less demanding problem in the educational domain compared with other data mining application domains because the stakeholders can be identified by their unique ID. Anonymous logins to the VLE are often forbidden.

A user session is defined as a sequence of requests made by a particular user over a certain navigation period. A user may have one or multiple sessions during this period. The session identification is a process of segmenting the logs of each stakeholder into disjoint sequences of individual sessions (Drlik, Munk, 2018). A more detailed discussion about the different session identification techniques is outside the scope of this course.

Finally, path reconstruction is a technique, which allows adding missing pieces of puzzle to the sequence of stakeholder's activity records in VLE with the aim to better understand his or her behaviour.

Linear Regression

Chapter **7**

7.1 Problem understanding

7.1.1

Regression analysis

The goal of predictive modeling is generally to create a model based on historical data with which we can predict a certain phenomenon for new data. In predictive modeling, we look for the most suitable approximation of a function that maps the input variables X to the output variables for y .

All approximation functions can be divided into classification and regression tasks. While the goal of classification is to assign the correct classification class for a given input data, the goal of a regression task is to assign a specific quantitative (continuous) variable to the input data.

The regression task is to find an approximation of the function f , which maps the input variables X to a continuous output variable y that has a numeric data type.

$$y = \beta_0 + \beta_1 * x$$

The success rate of the regression model is the root mean square error (RMSE) value, which is expressed in the same units as the output predicted variable.

7.1.2

Problem Understanding

The use of regression analysis in the LA domain is very common, as it is a very simple method. Therefore, we cannot expect groundbreaking findings that we would not know about before, rather we can use it in cases where we are interested in the influence of individual factors on the result. Still, LA areas are often used, for example, to predict student performance in a course, the correlation between different student characteristics, their activity, and academic achievement.

The dataset used in this example contains anonymized results of multiple evaluated student activities in a particular course. Since all activities were evaluated by points in the range, we will consider them as continuous variables. Although the final evaluation of the subject is a mark, in our example we will be interested in the achieved number of points.

The discussion of the legitimacy of classifying it as a machine learning algorithm is beyond the scope of this publication. From a practical point of view, it is important to know that the use of the linear regression method requires the fulfillment of several prerequisites. If these assumptions are not met, the results of the regression analysis may be misleading or completely incorrect. Therefore, it is always necessary to check that the following prerequisites are not violated:

- The relationship between the dependent and independent variables must be linear. In the case of multiple linear regression, the weighted sum of independent variables shall be able to explain the variability of the dependent variable.
- The residues must have a normal distribution.
- The residues must have a constant deviation (homoskedasticity).
- Residues must not be correlated (autocorrelation).
- There must be no correlation between independent variables. If they correlate, we speak of multicollinearity. In such a model, the coefficients depend on the existence of another independent variable.

Since this is the first example, suppose these assumptions are met in our example. If they were not, we can transform them to a great extent using Pandas methods. This example is more of a demonstration of the procedure than a benefit to LA given the quality of the available data.

7.1.3

Import of necessary modules and functions

As usual, we will start by importing the necessary modules. Most modules are encountered repeatedly, so there is no need to introduce them further. First, we import data visualization functions.

```
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')
```

And now the functions for their analysis. The *patsy* library simplifies model writing. We imported *plot_corr* and *train_test_split* functions from *statsmodels.graphics.correlation* and *sklearn.model_selection*.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split
```

As we are used to, we upload the data file to the *Pandas* dataset.

```
dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)
```

7.1.4

Data Understanding

At the beginning, show an example of the dataset.

```
# prepared code
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split

dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)
print(dataset.head())
```

Program output:

	pristupy	testy	skuska	projekt	zadania	vysledne_body
vysledna_znamka						
0	1354	41.5	14.0	88.40	40.0	162.21
C						
1	360	24.0	0.0	82.33	7.0	96.40
FX						
2	849	49.0	16.0	79.79	40.0	169.54
B						
3	173	41.0	18.0	83.16	0.0	118.23
FX						
4	478	7.0	0.0	0.00	10.0	21.67
FX						

7.1.5

Data pre-processing

We will remove any empty and duplicate values from the dataset.

```
# prepared code
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split

dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)
dataset2 = dataset.dropna()
dataset2 = dataset2.drop_duplicates()
```

Now we get acquainted in more detail with each column of the dataset.

```
print(list(dataset2.columns))
```

Program output:

```
['pristupy', 'testy', 'skuska', 'projekt', 'zadania',
'vysledne_body', 'vysledna_znamka']
```

We rename the columns if necessary to make it easier to interpret the results. We will create the second dataset to renamed fields.

Knowing the data types of individual columns of the dataset is a prerequisite for the successful application of any model. Using *the info()* function, we display the number of rows of the dataset. In addition, we can see if any of the columns contain blank values (*non-null*) as well as the data types of each column.

```
dataset2.info()
```

Program output:

```
Int64Index: 62 entries, 0 to 61
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   pristupy              62 non-null    int64
1   testy                 62 non-null    float64
2   skuska                62 non-null    float64
3   projekt               62 non-null    float64
4   zadania               62 non-null    float64
5   vysledne_body        62 non-null    float64
6   vysledna_znamka      62 non-null    object
dtypes: float64(5), int64(1), object(1)
memory usage: 3.9+ KB
```

We calculate basic descriptive statistics of a data set to understand the data, using the *function describe()*. The *include* parameter specifies the data types for which descriptive statistics should be displayed. It usually makes sense to use only numerical data types. Finally, we transposed the table for a clearer display of data using the *function .T*.

```
print(dataset2.describe(include=[np.number]).T)
```

Program output:

	count	mean	std	min	25%
50% \					
pristupy	62.0	985.048387	414.657252	28.0	691.7500
961.00					
testy	62.0	40.112903	13.015885	0.0	35.2500
43.50					
skuska	62.0	13.177419	4.880621	0.0	13.0000
14.00					
projekt	62.0	81.230000	25.512808	0.0	82.1275
89.58					
zadania	62.0	31.693548	11.148166	0.0	28.5000
36.00					
vysledne_body	62.0	147.286613	43.066892	0.0	135.9725
162.07					
		75%	max		
pristupy	1178.5000	2135.00			
testy	49.0000	57.00			
skuska	16.0000	20.00			

projekt	93.7475	99.48
zadania	40.0000	40.00
vysledne_body	175.8125	189.92

We divide the data from the dataset into training and test sets using the *train_test_split* function. For example, we can allocate 30% of data to *test_data_size* parameters for testing. In order to reproduce the calculation, we set the *seed* for the random number generator, e.g., to 10.

```
X =
dataset2[['projekt', 'zadania', 'pristupy', 'skuska', 'testy']]
y = dataset2[['vysledne_body']]
seed = 10
test_data_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = test_data_size, random_state = seed)
train_data = pd.concat([X_train, y_train], axis = 1)
test_data = pd.concat([X_test, y_test], axis = 1)
```

As part of this phase, we should examine the assumptions for the use of linear regression. To do this, we can use visualization using charts from the Seaborn library.

Multicollinearity can be investigated using the *corr()* method. More precisely, we calculate a matrix of correlation coefficients for the numeric data columns of the training set. The matrix is symmetrical according to the left diagonal. The diagonal shows the correlation of the variable with itself, so it is natural that it is equal to 1 and is not suitable for further analysis.

```
corrMatrix = train_data.corr(method = 'pearson')
print(corrMatrix)
```

Program output:

	projekt	zadania	pristupy	skuska
testy	vysledne_body			
projekt	1.000000	0.801389	0.467812	0.557367
0.666989	0.899312			
zadania	0.801389	1.000000	0.648714	0.629827
0.667937	0.876565			
pristupy	0.467812	0.648714	1.000000	0.528425
0.492659	0.578795			
skuska	0.557367	0.629827	0.528425	1.000000
0.863221	0.790099			
testy	0.666989	0.667937	0.492659	0.863221
1.000000	0.907615			

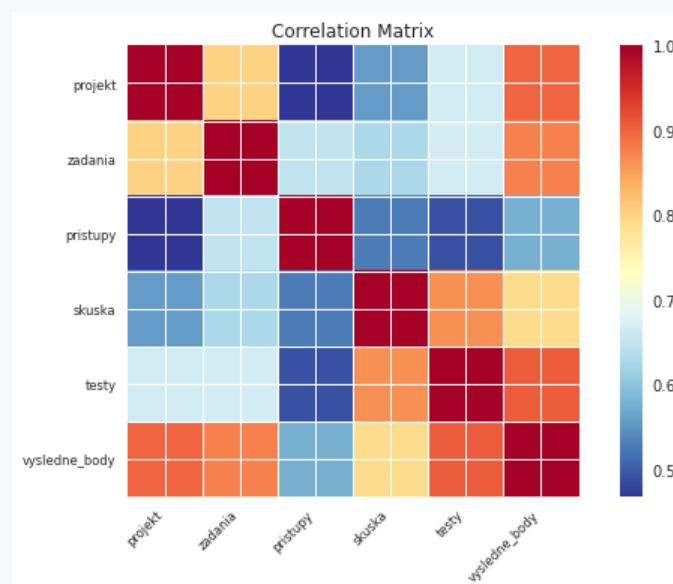
```
vysledne_body  0.899312  0.876565  0.578795  0.790099
0.907615      1.000000
```

The most common method of detecting a correlation between two numerical values is the Pearson correlation coefficient. We also used it as a parameter of the `corr()` function. Let's repeat what the individual values of the correlation coefficient (r) mean. It takes values from the interval $[-1, 1]$. If $r = 1$, both variables x and y , ideally grow and fall together in the same direction. In the case if $r = -1$, this direction is the opposite, if x grows, y decreases. If $r = 0$, there is no linear dependence between the variables x and y . Under normal conditions, a coefficient r equal to 1 or -1 is not common. It is often sufficient if the correlation coefficient is greater than 0.6 to consider the relationship between the variables as linear. If there is no linear dependence between these variables, this does not mean that there are no other types of dependencies between them, but we cannot express them with a simple line.

We display the result using a heat map of the correlation coefficients (Figure 11). Using the color scale, we can immediately see how strong the positive or negative relationship between the individual pairs of variables is.

```
xnames=list(train_data.columns)
ynames=list(train_data.columns)
plot_corr(corrMatrix, xnames=xnames, ynames=ynames,\
          title=None, normcolor=False, cmap='RdYlBu_r')
```

Program output:



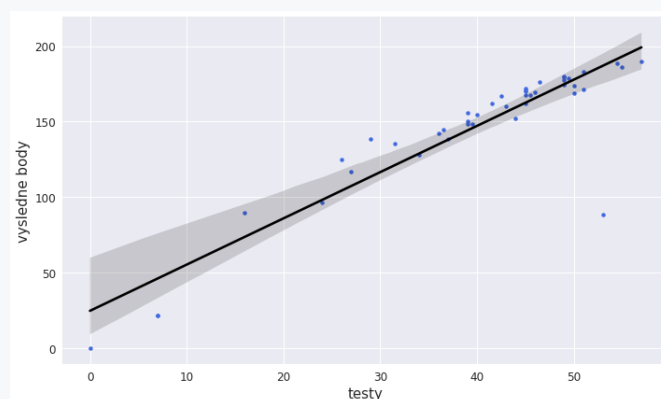
We can see in the displayed map the degree of correlation between individual variables. Based on it, we can say between which variables there is a linear relationship, and therefore which variables are a suitable adept for further regression analysis.

In addition to the map of correlation coefficients, we can use a scattering graph in which we also display a linear regression line. To do this, we will use the *subplots* functions of the *matplotlib* module and the *regplot* function from the *seaborn* module. The function expects the following arguments:

- independent variable *x*,
- dependent variable *y*,
- confidence interval of the regression parameter *CI* from the interval $<0.100>$,
- a dataset that contains a training set,
- *Matplotlib* chart object, labeled *AX*,
- description of the *x* and *y* axes,
- the size of the display of the canvas on which the chart will be rendered,
- other appearance settings.

```
fig, ax = plt.subplots(figsize=(10, 6))
sns.regplot(x='testy', y='vysledne_body', ci=95,
data=train_data, ax=ax, color='k', scatter_kws={"s": 20,
"color": "royalblue", "alpha": 1})
ax.set_ylabel('vysledne body', fontsize=15, fontname='DejaVu
Sans')
ax.set_xlabel("testy", fontsize=15, fontname='DejaVu Sans')
ax.set_xlim(left=None, right=None)
ax.set_ylim(bottom=None, top=None)
ax.tick_params(axis='both', which='major', labelsize=12)
fig.tight_layout()
```

Program output:

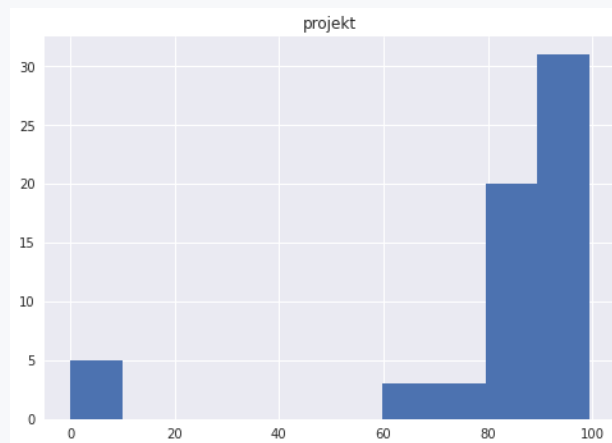


On the graph we see a line that characterizes the relationship between independent and dependent variables.

At the same time, as part of exploratory analysis, we can assess the distribution of values of the selected attribute using the *hist()* method.

```
X[['projekt']].hist()
```

Program output:



At the same time, we should examine the skewness and pointiness of the distribution of independent variables and make the necessary transformations.

```
print("Kurtosis",X_train.kurtosis(axis=0))
print("Skew",X_train.skew(axis=0))
```

Program output:

```
Kurtosis projekt      5.111897
zadania      2.191972
pristupy     0.507143
skuska       2.325862
testy        1.897767
dtype: float64
Skew projekt      -2.479007
zadania      -1.723516
pristupy     0.473184
skuska       -1.681946
testy        -1.449243
dtype: float64
```

As well as the dependent *variable* `vysledne_body`.

```
print("Target Kurtosis",y_train.kurtosis(axis=0))
print("Target Skew",y_train.skew(axis=0))
```

Program output:

```
Target Kurtosis vysledne_body      3.811987
dtype: float64
Target Skew vysledne_body      -1.989368
dtype: float64
```

We will learn more about exploratory analysis methods, for example, in a professional publication (Munk, 2011).

7.2 Linear regression models

7.2.1

Simple linear regression model

Now we wonder what the equation that would characterize the straight line in previous figure looks like. To do this, we will use functions from the *Statsmodels* library. This library, together with the *patsy* library, will allow us to easily define multiple regression models and experiment with them in search of the most appropriate model.

We define a linear regression model and assign it to the variable *linearModel*.

```
# prepared code
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split

dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)

linearModel = smf.ols(formula='vysledne_body ~ projekt',
data=train_data)
```

We will use the *OLS* function, where as an argument we will use a formula created using a special *patsy* library, which simplifies the writing of relationships between independent and dependent variables using the \sim sign. In the *data* argument, we will indicate where the function finds these variables, in our case it will again be a training dataset *train_data*.

Now we can call *the fit()* method and assign the result to the variable *linearModelResult*. This method adapts the linear regression model to the data, i.e., it tries to estimate the regression coefficients using the method of least squares.

```
linearModelResult = linearModel.fit()
```

Finally, we can visualize all the important characteristics of the model using the *summary()* method.

```
print(linearModelResult.summary())
```

Program output:

```

                                OLS Regression Results
=====
Dep. Variable:                  vysledne_body    R-squared:
0.809
Model:                          OLS            Adj. R-squared:
0.804
Method:                        Least Squares    F-statistic:
173.4
Date:                          Thu, 30 Mar 2023    Prob (F-statistic):
2.56e-16
Time:                          12:03:02          Log-Likelihood:
-188.00
No. Observations:              43                AIC:
380.0
Df Residuals:                  41                BIC:
383.5
Df Model:                      1
Covariance Type:              nonrobust
=====
=====
                                coef    std err          t      P>|t|
-----
[0.025    0.975]
-----
Intercept    28.8132    9.413    3.061    0.004
9.803    47.823
projekt      1.4650    0.111   13.168    0.000
1.240    1.690
=====
=====

```



```

Omnibus:                    5.880    Durbin-Watson:
1.655
Prob(Omnibus) :             0.053    Jarque-Bera (JB) :
7.323
Skew:                      -0.242    Prob(JB) :
0.0257
Kurtosis:                  4.963    Cond. No.
266.
=====
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.

```

The *fit()* method provides a number of functions by which we can examine its outputs. The most useful function is *summary()*, which allows you to display model parameters, confidence intervals, p-values, and t-values.

We find information about the dependent variable of the model in the upper left corner. On the right there are other characteristics of the model that speak of its quality, for example, *R-squared* - the coefficient of determinance. This statistic represents a measure of variability of the dependent variable that the model created by us can explain.

We see on the left that the dependent variable of the *Dep. Variable* model is *vysledne_body*. The R-squared statistic takes on a value of 0.809, i.e. 81%. This characteristic tells us to what extent we can describe the variability of the dependent variable in the proposed model. Thus, our model describes 80% of all values of the dependent variable well enough.

Simultaneously, we see in the *coef* section just the coefficients that are sought, which characterize the linear regression line. The *Intercept* coefficient represents the value at which the line intersects the y-axis. The second coefficient, together with the sign of the independent variable, characterizes the slope of the line.

The resulting straight line that characterizes the relationship between the independent variable *projekt* and the dependent variable *vysledne_body* has the following form:

$$y = 1,465 * projekt + 28.8132.$$

If we look at whether our proposed model models the relationship between variables, we examine its statistical significance. We are interested in the result of the F-test, which is expressed in the summary table by *the value of F-statistic*. We consider this value good if it is greater than 1. In our case, it is equal to 173.4.

At the same time, the probability value *Prob (F-statistic)* = 2.56e-16 is less than the selected significance level value of 0.05. Therefore, we can conclude that the risk that we have mistakenly rejected the null hypothesis and the model is not statistically significant is less than the specified 5%. In other words, our model is statistically significant at the selected 95% confidence level.

Once we have verified that our model is statistically significant in general, we still need to verify the statistical significance of the individual independent variables of the model. We will look at *the value of p-values* in the summary table labeled *p>|t|*. Again, we will compare this value for each independent variable with the chosen significance level $\alpha = 0.05$. If the *p-value* value for a given variable is less than 0.05, the variable chosen is statistically significant for our model and contributes to the expression of variability of the independent variable. Conversely, if it is larger, the variable is not statistically significant and should be removed from the model. In our model, both independent variables chosen are statistically significant.

7.2.2

Multiple linear regression model

In the previous example of linear regression models, we modeled the relationship between the selected independent variable and the dependent variable. In real conditions, however, we often encounter that the dependent variable is affected by several independent variables, each of which can affect the dependent variable with different weights. Therefore, it is necessary to investigate all independent variables that may explain the variability of the dependent variable.

The procedure will be very similar to the previous models, with the difference that in the formula used to create the model indicate all independent variables that could affect the dependent variable. Using the *patsy* library, these variables are given as an argument to the *OLS* method, and it is a symbolic notation, the + sign does not mean that the variables are arithmetically summed.

Let's add a second independent variable to the model - *tests*. Again, we can see the result in a clear table.

```
# prepared code
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
```

```

import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split

dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)
multiLinearModel2 = smf.ols(formula= 'vysledne_body ~ testy +
projekt', data=train_data)

```

The other steps are identical to those we have already described and implemented.

```

multiLinearModResult = multiLinearModel2.fit()
print(multiLinearModResult.summary())

```

Program output:

```

                                OLS Regression Results
=====
Dep. Variable:                  vysledne_body    R-squared:
0.979
Model:                            OLS          Adj. R-squared:
0.978
Method:                        Least Squares    F-statistic:
951.3
Date:                          Thu, 30 Mar 2023  Prob (F-statistic):
1.88e-34
Time:                          12:05:33        Log-Likelihood:
-140.08
No. Observations:                43            AIC:
286.2
Df Residuals:                    40            BIC:
291.5
Df Model:                        2
Covariance Type:                nonrobust
=====
=====

```

	coef	std err	t	P> t	
[0.025	0.975]				
Intercept	2.8192	3.438	0.820	0.417	-
4.128	9.767				

```

-----

```

```

testy          1.8704      0.103      18.207      0.000
1.663          2.078
projekt        0.8626      0.050      17.389      0.000
0.762          0.963
=====
=====
Omnibus:                  17.223      Durbin-Watson:
1.681
Prob(Omnibus) :           0.000      Jarque-Bera (JB) :
21.503
Skew:                    -1.332      Prob(JB) :
2.14e-05
Kurtosis:                5.214      Cond. No.
325.
=====
=====
Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.

```

The summary of the multiple regression model is above.

We can see that by adding another independent *variable test*, the *R-squared* coefficient of determinance increased up to 0.979. But since this coefficient naturally grows with the number of independent variables, even if this newly added variable does not have a strong correlation with the independent variable, it is necessary to observe the characteristic of *Adj. R-squared*. This characteristic only grows if the added independent variable actually contributes to explaining the variability of the model. Since its value is also high in our model, 0.978, we can say that both independent variables explain the variability of the independent variable up to 98%. The resulting relationship takes the form:

$$y = 0.8626 * \text{project} + 1.8704 * \text{tests} + 2.8192.$$

7.3 Find the best model

7.3.1

Find the most appropriate linear regression model

We can obtain the model that best explains the variability of the dependent variable by combining multiple linear regression and other mathematical adjustments. The result may look like the following example. However, we must be aware of the increased risk of overfitting. The model will look like this taking into account all available independent variables.

```
# prepared code
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split

dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)

```

Again, we use the summary table to better understand the model metrics.

```
multiLinearModel3 = smf.ols(formula= 'vysledne_body ~ testy +
zadania + projekt + skuska + pristupy', data=train_data)
multiLogLinModResult = multiLinearModel3.fit()
print(multiLogLinModResult.summary())

```

Program output:

OLS Regression Results		
=====		
=====		
Dep. Variable:	vysledne_body	R-squared:
1.000		
Model:	OLS	Adj. R-squared:
1.000		
Method:	Least Squares	F-statistic:
1.794e+09		
Date:	Thu, 30 Mar 2023	Prob (F-statistic):
5.07e-154		
Time:	12:08:09	Log-Likelihood:
191.52		
No. Observations:	43	AIC:
-371.0		
Df Residuals:	37	BIC:
-360.5		

```

Df Model:                    5
Covariance Type:            nonrobust
=====
=====
              coef      std err          t      P>|t|
[0.025      0.975]
-----
-----
Intercept      0.0021      0.002      1.283      0.207      -
0.001      0.005
testy          1.6664      7.88e-05      2.12e+04      0.000
1.666      1.667
zadania        1.0000      8.43e-05      1.19e+04      0.000
1.000      1.000
projekt        0.6000      3.09e-05      1.94e+04      0.000
0.600      0.600
skuska         0.0004      0.000      2.270      0.029
4.54e-05      0.001
pristupy       4.38e-07      1.46e-06      0.299      0.766      -
2.53e-06      3.4e-06
=====
=====
Omnibus:                    1.443      Durbin-Watson:
1.823
Prob(Omnibus) :              0.486      Jarque-Bera (JB) :
1.205
Skew:                       -0.402      Prob(JB) :
0.548
Kurtosis:                   2.840      Cond. No.
3.85e+03
=====
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 3.85e+03. This might indicate that there are strong multicollinearity or other numerical problems.

We can see that the proposed model is statistically significant (*Adj. R-squared* = 1.00, *F-statistic* = 1.794e+09 takes on large values, while *Prob (F-statistic)* approaches zero).

On the other hand, we see that the independent variable *approaches* is not statistically significant ($p > |t| = 0.766$) and should be removed from the model. At the same time, it should be stressed again that not all preconditions have been sufficiently addressed, as evidenced by the warning in note [2] on multicollinearity.

7.3.2

Explanation of the results of the regression analysis

As we have already highlighted, the objective of the regression analysis is to find the model that will best explain the variability of the observed dependent variable. We can use the *R-squared* (R^2) statistic to express it quantitatively. What does R^2 mean? R^2 is the quotient of two characteristics:

- Total Sum of Squares (TSS), which represents the total variance of the dependent variable on its average value
- Regression Sum of Squares (RSS) represents the total variability of the dependent variable that our model can explain.

We will base our explanation on a publication (So et al., 2020). In the case of an ideal model without prediction errors, both characteristics will be equal, i.e., the model will cover all variability in the data of the dependent variable relative to its average value.

In fact, RSS is always smaller than TSS. We refer to this difference as Error Sum of Square (ESS). ESS represents a certain degree of variability of the dependent variable that we cannot explain to our models. If we use simple linear regression with one independent variable, R^2 expresses with sufficient precision how well the model models the data obtained. The situation is different for multiple linear regression because R^2 reacts sensitively to the addition of another independent variable to the model, even if that variable correlate only slightly with the dependent variable. Adding an independent variable causes R^2 to naturally increase, which may lead to an attempt to automatically add more independent variables to the model. In doing so, however, we risk overfitting.

The adjusted *Adjusted R^2* statistic is a solution to this problem. Its value will only increase if the added independent variable contributes to a better explanation of the variability of the values of the dependent variable. In our example, naturally, if we consider the influence of all independent variables on the final score, we get closer to 100%.

7.3.3

Model validation

Thus, we see that the application of regression analysis represents an iterative process of searching for a suitable model, also using the characteristics *R-squared* and *Adj. R-squared*. Both characteristics provide an estimate of how strong the relationship between the model and the dependent variable is.

However, this is not all, because after we find a suitable model, we must formally verify the model statistically. In other words, we want to see if the model only models the data in our dataset or actually provides an explanation of the problem being studied in general. We can use one of the proven statistical hypothesis tests for this. Thus, if any independent variable has a relationship with the dependent variable in our model, it must have a non-zero coefficient β (it can be positive or negative).

At that point, we should verify whether this coefficient will exist even if we use data from the same domain but from a different period, or it is valid only for the currently used data. Next, we need to find out if we did not find this coefficient only by chance. These are the questions that we get answered with the help of hypothesis tests. Although hypotheses do not give us one hundred percent guarantee that we did not determine the coefficient by chance, thanks to them we can tell at what level of significance the coefficient will be true, that the coefficient β is not random.

So, let's start by agreeing on the amount of risk that we found the coefficient by chance. Usually, this amount of risk is referred to as α and has a value of 0.05 and 5% respectively. If we calculate $1 - \alpha$, we get a measure of the confidence level that we did not find a non-zero coefficient in our analysis by chance. Our confidence level is 95% to 5% level.

Now we can calculate the probability (p-value) corresponding to the coefficient β found by us in our model with a specified confidence value of $1 - \alpha$. If the probability is less than the specified value, we reject the hypothesis that the coefficient β was determined by chance. In other words, the coefficient β is statistically significant, and we reject the null hypothesis.

F-test

We will use the F-test to verify the statistical significance between the model and the dependent variable. If the p-value in the F-test is less than the selected significance level α , we reject the null hypothesis and thus the model is statistically significant.

In the summary table, we notice the value of F-value, which increases together with R^2 . We are looking for such a model that the F-value is greater than 1. At the same time, we see that the characteristic of Prob (F-statistic) is close to zero, which means that the risk that we rejected the hypothesis by mistake is less than 5%. In other words, in this case, our model is statistically significant at a significance level of 95%.

T-test

Once we know that our model is statistically significant in general, we can investigate the statistical significance of independent variables in it. We will look at the value of the p-values characteristic of each independent variable from the summary table. In it we note the column $p > |t|$. Again, we compare the current p-value with the significance level $\alpha = 0.05$. If the independent variable has a p-value value less than 0.05, this variable is statistically significant and contributes to explaining the

variability of the dependent variable. Conversely, if it is equal to or greater, the variable is not statistically significant and can be removed from the model.

7.4 Prediction by Scikit-learn

7.4.1

Prediction of the value of a continuous dependent variable using Scikit-learn

We can also analyze the linear regression model without using *statmodels* and *patsy* libraries. In the following example, let's briefly introduce the whole procedure. First, we initialize the linear regression module, divide the data into training and test parts.

```
# prepared code
%matplotlib inline
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('seaborn')

import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import statsmodels.graphics.api as smg
import patsy
from statsmodels.graphics.correlation import plot_corr
from sklearn.model_selection import train_test_split

dataset =
pd.read_csv('https://priscilla.fitped.eu/data/1a/course02.csv'
)
dataset2 = dataset.dropna()
dataset2 = dataset2.drop_duplicates()

from sklearn.linear_model import LinearRegression
from sklearn import metrics
X2 = dataset2[['projekt', 'zadania', 'skuska', 'testy']]
y = dataset2[['vysledne_body']]
seed = 10
test_data_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X2, y,
test_size = test_data_size, random_state = seed)
```

This gives us two data frames.

```
train_data = pd.concat([X_train, y_train], axis = 1)
test_data = pd.concat([X_test, y_test], axis = 1)
```

Now we create a regression model and train it on training data.

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
print(LinearRegression(copy_X=True, fit_intercept=True,
n_jobs=1, normalize=False))
```

Program output:

```
LinearRegression(n_jobs=1, normalize=False)
```

The intercept parameter, that is, the point at which the straight line passes through the y-axis, is obtained by the following command.

```
print(regressor.intercept_)
```

Program output:

```
[0.00218585]
```

The values of individual coefficients are obtained using the variable *slope*.

```
print(regressor.coef_)
```

Program output:

```
[[6.00014257e-01 1.00004975e+00 4.32229007e-04
1.66641881e+00]]
```

Regression accuracy on test data is high.

```
print(regressor.score(X_train, y_train))
```

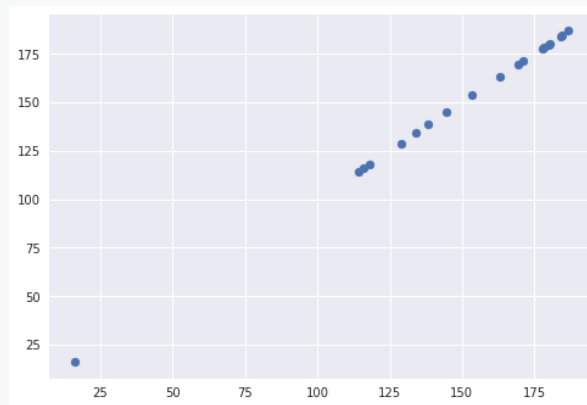
Program output:

```
0.9999999958651864
```

However, if we test our model on test data, we see that some values differ significantly from the predicted ones.

```
predictions = regressor.predict(X_test)
plt.scatter(y_test, predictions)
```

Program output:



This problem is caused by outlier values. Their origin requires a more detailed discussion that goes beyond the scope of this publication. Finally, we will show the basic characteristics of regression. Their values are not too far from 0, but the influence of outliers is also captured in them.

```
print('MAE:', metrics.mean_absolute_error(y_test,
predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,
predictions)))
```

Program output:

```
MAE: 0.0031487155172219387
MSE: 1.4846435055662263e-05
RMSE: 0.0038531071949353115
```

However, this does not mean that these are erroneous values, but the phenomenon that we are trying to model using linear regression has its own specifics. Simply, for example, the teacher in some cases adjusted the assigned points with bonus tasks. A more detailed discussion of each metric is available in many publications.

Random Forest

Chapter 8

8.1 Multiple Classifier Random Forest

8.1.1

The task of the classification algorithm is to approximate the mapping function f , which maps input variables (X) to discrete output variables(s), which we call categories or classification classes. Input variables must have a numeric data type. The condition for using the classification algorithm is the existence of two (binary classification) and more classification classes (multi-class classification).

Classification models seek to predict a continuous value as the probability with which a given observation falls into each of the defined classification classes. Probability can be interpreted as a measure of the confidence with which a given value will belong to a given classification class. The resulting classification class then represents the class for which we found the highest probability.

The measure of classification success is accuracy, which represents the ratio between correctly classified and all observations. In addition to accuracy, there are other important metrics, some of which we will imagine.

A binary classifier, such as logistic regression, allows you to classify occurrences into only two classes - 0 (no) and 1 (yes). The multiple classifier is an extension of the binary classifier. We will introduce its use in the context of educational data in this chapter.

Random Forest is a very common classification algorithm. Its theoretical foundations are outside the scope of this publication. Details can be found, for example, in James et al. (James et al., 2017) or other scholarly publications.

8.1.2

Data Import

We will again examine partial student achievement and the ability of the Random Forest classifier to predict the final grade. However, unlike the previous examples, we must modify the original file to a form with which the algorithm can work effectively.

Since the original partial compulsory activities were evaluated by a continuous variable – points, we must transform them into a discrete form. Therefore, we replaced them with four levels of quality (poor, average, above average, excellent). Thus, we encoded the original points from the activity as follows. We assigned it a value of 1 in this interval and entered values of 0 in all the others depending on which interval the achieved value fell into.

These operations could also be performed in the Jupyter Python Notebook environment using selected scikit-learn library methods. However, we used MS Excel.

pristupy	pristup1	pristup2	pristup3	pristup4
405	1	0	0	0
177	1	0	0	0
334	1	0	0	0
682	0	1	0	0
383	1	0	0	0
472	0	1	0	0
204	1	0	0	0
696	0	1	0	0

We start by importing the usual libraries for working with data.

```
import pandas as pd
import numpy as np
```

We import a data file with pre-prepared data.

```
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
```

8.1.3

EDA and Data Understanding

We will examine the basic characteristics of the ensemble.

```
import pandas as pd
import numpy as np

file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)

print(df.head())
```

Program output:

	pristup1	pristup2	pristup3	pristup4	testy1	testy2
testy3	testy4	\				
0	1	0	0	0	0	1
0	0					
1	1	0	0	0	1	0
0	0					

```

2      1      0      0      0      1      0
0      0
3      0      1      0      0      0      1
0      0
4      1      0      0      0      0      0
1      0

      skuska1  skuska2  ...  ptsql1  ptsql2  ptsql3  ptsql4  ptd1
ptd2  ptd3  \
0      0      0  ...      1      0      0      0      0
0      0
1      1      0  ...      1      0      0      0      1
0      0
2      1      0  ...      0      0      1      0      1
0      0
3      0      0  ...      0      0      0      1      0
0      0
4      0      0  ...      0      0      1      0      0
0      0

      ptd4  kontrolny_sucet  vysledna_znamka
0      1      7      D
1      0      7      FX
2      0      7      FX
3      1      7      C
4      1      7      C

[5 rows x 30 columns]

```

```
print(df.shape)
```

Program output:

```
(220, 30)
```

We can see that we have processed the original file into a form that contains 29 columns, mostly numeric data type.

```
df.info()
```

Program output:

```

RangeIndex: 220 entries, 0 to 219
Data columns (total 30 columns):
#   Column                Non-Null Count  Dtype

```

```

---      -----      -----
0      pristup1      220 non-null      int64
1      pristup2      220 non-null      int64
2      pristup3      220 non-null      int64
3      pristup4      220 non-null      int64
4      testy1        220 non-null      int64
5      testy2        220 non-null      int64
6      testy3        220 non-null      int64
7      testy4        220 non-null      int64
8      skuska1       220 non-null      int64
9      skuska2       220 non-null      int64
10     skuska3       220 non-null      int64
11     skuska4       220 non-null      int64
12     projekt1      220 non-null      int64
13     projekt2      220 non-null      int64
14     projekt3      220 non-null      int64
15     projekt4      220 non-null      int64
16     zadania1      220 non-null      int64
17     zadania2      220 non-null      int64
18     zadania3      220 non-null      int64
19     zadania4      220 non-null      int64
20     ptsql1        220 non-null      int64
21     ptsql2        220 non-null      int64
22     ptsql3        220 non-null      int64
23     ptsql4        220 non-null      int64
24     ptd1          220 non-null      int64
25     ptd2          220 non-null      int64
26     ptd3          220 non-null      int64
27     ptd4          220 non-null      int64
28     kontrolny_sucet 220 non-null      int64
29     vysledna_znamka 220 non-null      object
dtypes: int64(29), object(1)
memory usage: 51.7+ KB

```

We need to remove text values from the dataset, as well as unnecessary columns. In our case, it is column *kontrolny_sucet* that served to verify that the transformation to discrete values was carried out correctly for all observations.

```
df2 = df.drop(['kontrolny_sucet'], axis=1)
```

The **sklearn** library requires that the target variable be separated from the variables on the basis of which we will determine the classification class. To remove the target variable, we'll use the *pop()* method and assign it to a new Pandas DataFrame variable.


```
y = df2.pop('vysledna_znamka')
print(df2.head())
```

Program output:

```

    pristup1 pristup2 pristup3 pristup4 testy1 testy2
testy3 testy4 \
0         1         0         0         0         0         1
0         0
1         1         0         0         0         1         0
0         0
2         1         0         0         0         1         0
0         0
3         0         1         0         0         0         1
0         0
4         1         0         0         0         0         0
1         0

    skuska1 skuska2 ... zadania3 zadania4 ptsql1 ptsql2
ptsql3 ptsql4 \
0         0         0 ...         0         0         1         0
0         0
1         1         0 ...         0         0         1         0
0         0
2         1         0 ...         0         0         0         0
1         0
3         0         0 ...         0         0         0         0
0         1
4         0         0 ...         0         0         0         0
1         0

    ptd1 ptd2 ptd3 ptd4
0     0     0     0     1
1     1     0     0     0
2     1     0     0     0
3     0     0     0     1
4     0     0     0     1

[5 rows x 28 columns]
```

8.1.4

Create a training and test set

We proceed similarly to other models, dividing the data set into a training and test set. The model uses a training dataset to learn the relationships between variables, based on which it then predicts the resulting classification class. The test dataset is then used to verify how exactly we can predict the classification class using the model on previously unknown data. If the model appropriately models only data from the training set, we speak of overfitting.

The model is not capable of predicting on test data. Therefore, we are looking for such a model setting in terms of its hyperparameters, which will achieve a very similar level of model accuracy on both the training and test set.

Everything essential can be found again in the *scikit-learn* library. We will use the *train_test_split()* function to divide the data set into a training and test part.

```
import pandas as pd
import numpy as np
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)
```

This function returns four variables containing a given fraction of data:

- X_train
- X_test
- y_train
- y_test

8.2 Modeling using Random Forest

8.2.1

Modeling using Random Forest

We import the Random Forest classifier.

```
import pandas as pd
import numpy as np
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
```

```
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)

from sklearn.ensemble import RandomForestClassifier
```

This classifier, like other machine learning methods, needs to select the appropriate values of hyperparameters. A hyperparameter is a type of parameter that enters the model, and the model cannot learn this parameter itself and must therefore be explicitly selected based on user experience.

In our example, we will use predefined parameters for now. We will ensure the interpretability of the model by the *random_state* parameter.

```
rf_model = RandomForestClassifier(random_state=42)
```

Now we can train the model on training data. The model will attempt to learn the relationships between independent and modeled variables.

```
print(rf_model.fit(X_train, y_train))
```

Program output:

```
RandomForestClassifier(random_state=42)
```

Then we have the model ready, and we can predict the values of the classification class, so far on the training dataset.

```
train_preds = rf_model.predict(X_train)
print(train_preds)
```

Program output:

```
['D' 'B' 'A' 'D' 'FX' 'C' 'D' 'A' 'C' 'A' 'A' 'FX' 'C' 'C' 'B'
'D' 'C' 'A'
'B' 'B' 'C' 'B' 'B' 'FX' 'D' 'B' 'FX' 'B' 'C' 'C' 'D' 'B' 'A'
'FX' 'C'
'E' 'C' 'E' 'FX' 'E' 'C' 'FX' 'A' 'FX' 'C' 'D' 'B' 'C' 'C'
'D' 'FX' 'B'
'E' 'B' 'D' 'FX' 'C' 'A' 'C' 'D' 'A' 'FX' 'C' 'D' 'E' 'FX'
'FX' 'C' 'FX'
'B' 'FX' 'C' 'C' 'E' 'D' 'C' 'D' 'A' 'A' 'C' 'B' 'D' 'D' 'FX'
'C' 'FX'
'B' 'D' 'B' 'A' 'A' 'B' 'C' 'E' 'C' 'FX' 'FX' 'FX' 'E' 'D'
'A' 'C' 'B']
```

```
'E' 'FX' 'B' 'FX' 'FX' 'FX' 'C' 'FX' 'FX' 'E' 'D' 'B' 'A' 'C'
'B' 'C' 'B'
'A' 'C' 'C' 'FX' 'C' 'C' 'C' 'E' 'C' 'C' 'E' 'B']
```

8.2.2

Model Evaluation

We also assess the accuracy of the model in this case by metrics. Metrics such as *F1 score*, *precision*, *recall*, *ROC AUC* are used for classification tasks. We use *accuracy score*, which represents the ratio between correctly predicted classes and the total number of predictions made in the model. Accuracy ranges from 0 to 1.

```
# prepared code
import pandas as pd
import numpy as np
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)

from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(random_state=42)
print(rf_model.fit(X_train, y_train))
train_preds = rf_model.predict(X_train)
```

Program output:

```
RandomForestClassifier(random_state=42)
from sklearn.metrics import accuracy_score
```

First, we compare the accuracy of the model for the training set. Actual values are represented by *the variable y_train* and predicted by *preds*. Although the accuracy value may be high in this case, we do not win because we have not yet verified the model on the test data. Our goal is to achieve comparable model accuracy on both sets of data.

```
train_acc = accuracy_score(y_train, train_preds)
print(train_acc)
```

Program output:

```
0.9545454545454546
```

We can see that we achieve high AUC values on training data. Let's see what it looks like on the test data.

```
test_preds = rf_model.predict(X_test)
test_acc = accuracy_score(y_test, test_preds)
print(test_acc)
```

Program output:

```
0.6363636363636364
```

The difference is significant. If the model achieves much higher accuracy on training data than on test data, it is overfitted, and we need to focus on the appropriate selection of hyperparameters of the model.

8.3 Hyperparameters Tuning

8.3.1

Hyperparameters Tuning

As we noted before, the reliability of the model depends very much on the correct setting of hyperparameters, that is, the parameters that are set by the user. The search for suitable parameters is an iterative process that we can implement manually, choosing values based on experience or randomly. It is more effective to use again integrated methods that automate the process of finding parameters.

But as we repeatedly create and train a model, we can shorten writing repetitive code with a few custom features. We will prepare a method for training a model with several hyperparameters.

```
def train_rf(X_train, y_train, random_state=42,
n_estimators=10, max_depth=None, min_samples_leaf=1,
max_features='sqrt'):
    rf_model = RandomForestClassifier(random_state=random_state,
n_estimators=n_estimators, max_depth=max_depth,
min_samples_leaf=min_samples_leaf, max_features=max_features)
    rf_model.fit(X_train, y_train)
    return rf_model
```

We will also prepare a method for prediction.

```
def get_preds(rf_model, X_train, X_test):
    train_preds = rf_model.predict(X_train)
    test_preds = rf_model.predict(X_test)
```

```
return train_preds, test_preds
```

Another method is designed to calculate the accuracy of the model. Specifically, returns *accuracy* for both the training and test datasets.

```
def print_accuracy(y_train, y_test, train_preds, test_preds):
    train_acc = accuracy_score(y_train, train_preds)
    test_acc = accuracy_score(y_test, test_preds)
    print(train_acc)
    print(test_acc)
    return train_acc, test_acc
```

The resulting method combines the above partial methods and returns us the complete characteristic of the model.

```
def fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=10, max_depth=None,
min_samples_leaf=1, max_features='auto'):
    rf_model = train_rf(X_train, y_train,
random_state=random_state, n_estimators=n_estimators,
max_depth=max_depth, min_samples_leaf=min_samples_leaf,
max_features=max_features)
    train_preds, test_preds = get_preds(rf_model, X_train,
X_test)
    train_acc, test_acc = print_accuracy(y_train, y_test,
train_preds, test_preds)
    return rf_model, train_preds, test_preds, train_acc,
test_acc
```

Once we have the methods ready, we will use them just like in this example.

```
# prepared code
import pandas as pd
import numpy as np
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
rf_model_1, trn_preds_1, tst_preds_1, trn_acc_1, tst_acc_1 =
fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=30, max_depth=2,
min_samples_leaf=7, max_features=0.2)
```

Program output:

```
0.5378787878787878
0.5454545454545454
```

We can see that the random determination of hyperparameters led to a decrease in the *accuracy* value, and the resulting model is very inaccurate. Therefore, we introduce some hyperparameters and their meaning in more detail.

8.3.2

Number of Trees Estimator

A decision tree is an oriented graph that contains nodes and leaves. The node contains a condition that can be true (yes, 1) or not true (no, 0). Accordingly, decision-making is shifted to the next node or letter. The sheet is a special type of node in which prediction is realized in the model.

The Random Forest algorithm runs through an input dataset and creates multiple decision trees because it searches for optimal values for dividing data into two groups with similar classes.

The *n_estimator* parameter defines the number of decision trees that will be trained by the Random Forest algorithm. Its starting value is 10, so it creates 10 decision trees and assesses their ability to predict the target class. It then calculates the average of the above predictions, which we consider to be the resulting prediction. Naturally, as the number of trees increases, so does the accuracy of the model.

```
# prepared code
import pandas as pd
import numpy as np
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```

# fit_predict_rf
#####
def fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=10, max_depth=None,
min_samples_leaf=1, max_features='auto'):
    rf_model = train_rf(X_train, y_train,
random_state=random_state, n_estimators=n_estimators,
max_depth=max_depth, min_samples_leaf=min_samples_leaf,
max_features=max_features)
    train_preds, test_preds = get_preds(rf_model, X_train,
X_test)
    train_acc, test_acc = print_accuracy(y_train, y_test,
train_preds, test_preds)
    return rf_model, train_preds, test_preds, train_acc,
test_acc

# train_rf #####
def train_rf(X_train, y_train, random_state=42,
n_estimators=10, max_depth=None, min_samples_leaf=1,
max_features='sqrt'):
    rf_model = RandomForestClassifier(random_state=random_state,
n_estimators=n_estimators, max_depth=max_depth,
min_samples_leaf=min_samples_leaf, max_features=max_features)
    rf_model.fit(X_train, y_train)
    return rf_model

# get_preds
#####
def get_preds(rf_model, X_train, X_test):
    train_preds = rf_model.predict(X_train)
    test_preds = rf_model.predict(X_test)
    return train_preds, test_preds

# print_accuracy
#####
def print_accuracy(y_train, y_test, train_preds, test_preds):
    train_acc = accuracy_score(y_train, train_preds)
    test_acc = accuracy_score(y_test, test_preds)
    print(train_acc)
    print(test_acc)
    return train_acc, test_acc

rf_model_1, trn_preds_1, tst_preds_1, trn_acc_1, tst_acc_1 =
fit_predict_rf(X_train, X_test, y_train, y_test,

```



```
random_state=42, n_estimators=30, max_depth=2,  
min_samples_leaf=7, max_features=0.2)
```

Program output:

```
0.5378787878787878  
0.5454545454545454
```

```
rf_model_1, trn_preds_1, tst_preds_1, trn_acc_1, tst_acc_1 =  
fit_predict_rf(X_train, X_test, y_train, y_test,  
random_state=42, n_estimators=1, max_depth=None,  
min_samples_leaf=1, max_features='auto')
```

Program output:

```
0.8333333333333334  
0.5795454545454546  
/home/johny/.local/lib/python3.9/site-  
packages/sklearn/ensemble/_forest.py:427: FutureWarning:  
`max_features='auto'` has been deprecated in 1.1 and will be  
removed in 1.3. To keep the past behaviour, explicitly set  
`max_features='sqrt'` or remove this parameter as it is also  
the default value for RandomForestClassifiers and  
ExtraTreesClassifiers.  
    warn(
```

The accuracy values for the training and test set have approached in the case of a value of 5, so we will further consider *the parameters* `n_extimator = 30` and `max_depth = 5`.

8.3.3

Minimum Sample per Leaf

The *parameter* `min_samples_leaf` determines the minimum number of observations (predictions) that must fall into one of the leaves of the decision tree. For example, if it has a value of 5, then the algorithm will create a condition only if at least five observations fall into each of the formed leaf nodes of the tree. The default value of the parameter is 1.

We try to change it to 3.

```
# prepared code  
import pandas as pd  
import numpy as np
```

```

file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# fit_predict_rf
#####
def fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=10, max_depth=None,
min_samples_leaf=1, max_features='auto'):
    rf_model = train_rf(X_train, y_train,
random_state=random_state, n_estimators=n_estimators,
max_depth=max_depth, min_samples_leaf=min_samples_leaf,
max_features=max_features)
    train_preds, test_preds = get_preds(rf_model, X_train,
X_test)
    train_acc, test_acc = print_accuracy(y_train, y_test,
train_preds, test_preds)
    return rf_model, train_preds, test_preds, train_acc,
test_acc

# train_rf #####
def train_rf(X_train, y_train, random_state=42,
n_estimators=10, max_depth=None, min_samples_leaf=1,
max_features='sqrt'):
    rf_model = RandomForestClassifier(random_state=random_state,
n_estimators=n_estimators, max_depth=max_depth,
min_samples_leaf=min_samples_leaf, max_features=max_features)
    rf_model.fit(X_train, y_train)
    return rf_model

# get_preds
#####
def get_preds(rf_model, X_train, X_test):
    train_preds = rf_model.predict(X_train)
    test_preds = rf_model.predict(X_test)
    return train_preds, test_preds

```

```
# print_accuracy
#####
def print_accuracy(y_train, y_test, train_preds, test_preds):
    train_acc = accuracy_score(y_train, train_preds)
    test_acc = accuracy_score(y_test, test_preds)
    print(train_acc)
    print(test_acc)
    return train_acc, test_acc

rf_model_6, trn_preds_6, tst_preds_6, trn_acc_6, tst_acc_6 =
fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=30, max_depth=2,
min_samples_leaf=3, max_features='auto')
```

Program output:

```
0.5303030303030303
0.5227272727272727
/home/johny/.local/lib/python3.9/site-
packages/sklearn/ensemble/_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set
`max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and
ExtraTreesClassifiers.
    warn(
```

In the same way, if we change it to 7, we get a very low prediction accuracy:

```
0.553030303030303
0.4659090909090909
```

Therefore, we will not further tune this hyperparameter.

8.3.4

Maximum Input Features

The last of the hyperparameters whose influence on the accuracy of the model we examine is the maximum number of input variables *max_features*. This parameter is responsible for the randomness of selection, hence Random Forest.

The algorithm creates several independent trees using a randomly selected data sample (changes the proportion of test and training data). In addition, it randomly selects independent variables, i.e., columns whose values it will consider in the decision tree. Thus, each of the created trees contains only a subset of all

independent variables. A parameter can be defined by a number or percentage of variables involved. The square root or natural logarithm, or all variables (None) can also be used.

```
# prepared code
import pandas as pd
import numpy as np
file_url =
'https://priscilla.fitped.eu/data/la/rf_grades.xlsx'
df = pd.read_excel(file_url)
df2 = df.drop(['kontrolny_sucet'], axis=1)
y = df2.pop('vysledna_znamka')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df2, y,
test_size=0.4, random_state=42)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# fit_predict_rf
#####
def fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=10, max_depth=None,
min_samples_leaf=1, max_features='auto'):
    rf_model = train_rf(X_train, y_train,
random_state=random_state, n_estimators=n_estimators,
max_depth=max_depth, min_samples_leaf=min_samples_leaf,
max_features=max_features)
    train_preds, test_preds = get_preds(rf_model, X_train,
X_test)
    train_acc, test_acc = print_accuracy(y_train, y_test,
train_preds, test_preds)
    return rf_model, train_preds, test_preds, train_acc,
test_acc

# train_rf #####
def train_rf(X_train, y_train, random_state=42,
n_estimators=10, max_depth=None, min_samples_leaf=1,
max_features='sqrt'):
    rf_model = RandomForestClassifier(random_state=random_state,
n_estimators=n_estimators, max_depth=max_depth,
min_samples_leaf=min_samples_leaf, max_features=max_features)
    rf_model.fit(X_train, y_train)
    return rf_model
```

```

# get_preds
#####
def get_preds(rf_model, X_train, X_test):
    train_preds = rf_model.predict(X_train)
    test_preds = rf_model.predict(X_test)
    return train_preds, test_preds

# print_accuracy
#####
def print_accuracy(y_train, y_test, train_preds, test_preds):
    train_acc = accuracy_score(y_train, train_preds)
    test_acc = accuracy_score(y_test, test_preds)
    print(train_acc)
    print(test_acc)
    return train_acc, test_acc

rf_model_7, trn_preds_7, tst_preds_7, trn_acc_7, tst_acc_7 =
fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=30, max_depth=2,
min_samples_leaf=7, max_features=10)

```

Program output:

```

0.5303030303030303
0.5340909090909091

```

We were not able to improve the prediction ability of the model with different values.

As a next step, it would be advisable to use the cycle to start calculating the optimal characteristics of the model and find the best *accuracy* values for both the training and test dataset. Individual hyperparameters should acquire values that the researcher can interpret or that are related to the phenomenon under study.

However, we can help ourselves with integrated methods that automate manual testing of various hyperparameters. These methods either sequentially pass individual hyperparameter options (GridSearchCV), which is quite time-consuming, or choose a reasonable strategy of random selection of the value of individual hyperparameters (RandomizedSearchCV).

Although their use is again slightly beyond the capabilities of this course, we will give an example of their application to our problem without a detailed explanation of each step. We initialize the classifier.

```

#Initialize the classifier.
from sklearn import ensemble

```

```
rfc = ensemble.RandomForestClassifier(n_estimators=100,
random_state=42)
```

We use the method `GridSearchCV()`, defining the parameters we want to investigate in the grid object.

```
from sklearn import model_selection
grid = {
    'criterion': ['gini', 'entropy'],
    'max_features': [2, 4, 6, 8, 10, 12, 14]
}
gscv = model_selection.GridSearchCV(estimator=rfc,
param_grid=grid, cv=5, scoring='accuracy')
```

We will train the model and store the results in the results variable, sorting them by *rank_test_score* value.

```
gscv.fit(X_train,y_train)
results = pd.DataFrame(gscv.cv_results_)
results.sort_values('rank_test_score',
ascending=True).head(10)
```

We do the same for the `RandomizedSearchCV()` method. We have stored hyperparameters in the *variable param_dist* that the algorithm of the method is supposed to test. We assign the results to the results variable.

```
np.random.seed(100)
from scipy import stats
max_features = X_train.shape[1]
param_dist = {
    'criterion': ['gini', 'entropy'],
    'max_features': stats.randint(low=1, high=max_features)
}
rscv = model_selection.RandomizedSearchCV(estimator=rfc,
param_distributions=param_dist, n_iter=50, cv=5,
scoring='accuracy', random_state=42)
rscv.fit(X_train,y_train)
results = pd.DataFrame(rscv.cv_results_)
results.sort_values('rank_test_score',
ascending=True).head(10)
print(results.head(10))
```

Program output:

```
mean_fit_time  std_fit_time  mean_score_time
std_score_time  \
```

0	0.183534	0.001121	0.015421
0.000073			
1	0.171203	0.012142	0.014418
0.001439			
2	0.109450	0.013744	0.009339
0.001045			
3	0.096634	0.000112	0.008122
0.000034			
4	0.098379	0.000187	0.008094
0.000030			
5	0.097435	0.000113	0.008126
0.000035			
6	0.093040	0.000184	0.008140
0.000024			
7	0.099423	0.001196	0.008189
0.000204			
8	0.093051	0.000134	0.008091
0.000035			
9	0.091166	0.001120	0.008108
0.000023			

	param_criterion	param_max_features	\
0	gini	20	
1	gini	15	
2	gini	8	
3	gini	21	
4	gini	26	
5	gini	23	
6	gini	11	
7	entropy	21	
8	entropy	8	
9	entropy	3	

	split0_test_score	\	params
0	0.666667	{'criterion': 'gini', 'max_features': 20}	
1	0.629630	{'criterion': 'gini', 'max_features': 15}	
2	0.666667	{'criterion': 'gini', 'max_features': 8}	
3	0.629630	{'criterion': 'gini', 'max_features': 21}	

```

4      {'criterion': 'gini', 'max_features': 26}
0.629630
5      {'criterion': 'gini', 'max_features': 23}
0.629630
6      {'criterion': 'gini', 'max_features': 11}
0.592593
7      {'criterion': 'entropy', 'max_features': 21}
0.666667
8      {'criterion': 'entropy', 'max_features': 8}
0.592593
9      {'criterion': 'entropy', 'max_features': 3}
0.592593

      split1_test_score  split2_test_score  split3_test_score
split4_test_score \
0          0.703704          0.615385          0.538462
0.538462
1          0.592593          0.576923          0.538462
0.538462
2          0.592593          0.576923          0.500000
0.461538
3          0.703704          0.615385          0.500000
0.500000
4          0.703704          0.615385          0.500000
0.538462
5          0.666667          0.615385          0.500000
0.576923
6          0.629630          0.615385          0.538462
0.538462
7          0.703704          0.615385          0.576923
0.538462
8          0.592593          0.615385          0.500000
0.538462
9          0.555556          0.576923          0.653846
0.576923

      mean_test_score  std_test_score  rank_test_score
0          0.612536          0.066669           4
1          0.575214          0.034547          38
2          0.559544          0.072139          48
3          0.589744          0.079175          24
4          0.597436          0.071587          14
5          0.597721          0.056684          13
6          0.582906          0.038164          30

```


7	0.620228	0.059513	1
8	0.567806	0.042311	44
9	0.591168	0.033480	17

The data in the table show that for the selected hyperparameters, the best results are achieved if `n_estimator = 9` and criterion is gini.

```
rf_model_8, trn_preds_8, tst_preds_8, trn_acc_8, tst_acc_8 =
fit_predict_rf(X_train, X_test, y_train, y_test,
random_state=42, n_estimators=9, max_depth=None,
min_samples_leaf=1, max_features='auto')
```

Program output:

```
0.9166666666666666
0.6363636363636364
/home/johny/.local/lib/python3.9/site-
packages/sklearn/ensemble/_forest.py:427: FutureWarning:
`max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set
`max_features='sqrt'` or remove this parameter as it is also
the default value for RandomForestClassifiers and
ExtraTreesClassifiers.
  warn(
```

The resulting prediction accuracy for the training or test dataset is as above.

As already mentioned, the observed characteristic of model accuracy is not exceptional, but the example described describes the procedure and importance of the ECA phase, as well as the search for suitable hyperparameters of the model.

Clustering Using K-means

Chapter 9

9.1 Cluster analysis

9.1.1

Cluster analysis using K-means

K-means represents one of the most popular and simple algorithms of unsupervised machine learning. The aim of the algorithm is to group data (observations) into clusters based on a certain pattern or their distance from each other. Theoretical background on this topic and examples of algorithms can be found, for example, in the publication (Berka, 2003; So et al., 2020).

Thus, the algorithm tries to identify patterns/clusters based on calculating the distance between individual points/data. It uses different approaches to calculate the distance. We can get acquainted with it in more detail in the documentation in scikit-learn.

Basic characteristics:

- Divides a dataset into disjoint clusters,
- each cluster is represented by the average of the distance between each point of the cluster, which is called a centroid,
- in general, centroids are not the actual values of observations, the data of the original data set, but calculated values,
- the goal of the K-means algorithm is to find centroids that minimize dispersion (referred to as inertia), the sum of squares of distance.

Thus, we must take into account that:

- we may not find the most optimal global solution,
- we determine the number of clusters before using K-means,
- K-means is limited by the linear boundaries of clusters,
- K-means is computationally intensive in the case of larger data sets.

9.1.2

Import libraries

We import well-known libraries.

```
import pandas as pd
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans
```

Data import

In this example, we will use a dataset that contains the accesses of different types of Moodle LMS users to each course. These approaches were created by aggregating logs that the system stores in its database as a detailed record of user activities. Using the SQL aggregate function, we produced the resulting data set with three columns, *CourseID*, *Teachers_logs* and *Students_logs*.

Using the K-means algorithm, we will investigate whether *Teachers_logs* and *Students_logs* variables form similar clusters. We will measure the distance between them.

```
file_url = 'https://priscilla.fitped.eu/data/1a/logs_lms.csv'
```

We load the file in a standard way. The *usecols* parameter specifies a subset of the columns that we want to import. In our case, we will examine potential clusters between two variables.

```
df = pd.read_csv(file_url,
usecols=['CourseID', 'Teachers_logs', 'Students_logs'])
```

EDA and Data Understanding

We will explore the basic characteristics of variables.

```
print(df.head())
```

Program output:

	CourseID	Teachers_logs	Students_logs
0	1	2171	27655
1	2	3904	28242
2	3	1840	15749
3	4	4017	54076
4	5	3533	32530

```
print(df.tail())
```

Program output:

	CourseID	Teachers_logs	Students_logs
2468	2469	2477	14888
2469	2470	1318	9117
2470	2471	2409	16674
2471	2472	3139	28895
2472	2473	2291	18241

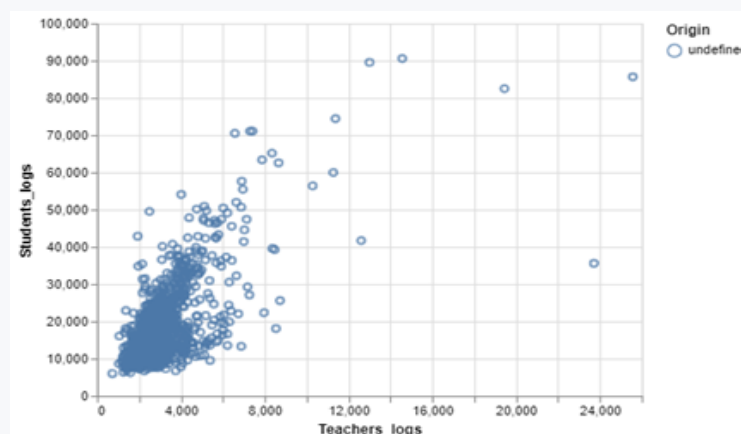
```
print(df.describe())
```

Program output:

	CourseID	Teachers_logs	Students_logs
count	2473.000000	2473.000000	2473.000000
mean	1237.000000	2782.669632	15957.382936
std	714.037931	1236.193614	7915.109176
min	1.000000	729.000000	5979.000000
25%	619.000000	2220.000000	11415.000000
50%	1237.000000	2540.000000	13718.000000
75%	1855.000000	2977.000000	17945.000000
max	2473.000000	25572.000000	90585.000000

This statistic is of little importance in the case of the nominal *CourseID*. If we visualize the other two variables, we see the relative position of the individual points.

```
alt.Chart(df).mark_point().encode(
    x='Teachers_logs:Q',
    y='Students_logs:Q',
    color='Origin:N',
)
```



9.1.3

Data Standardization

In calculating the distances between individual cases as well as clusters, differences in the size of individual variables that enter the algorithm play an important role. Different approaches are used to calculate distance, but whenever variables are not on the same scale, a variable with large values tends to significantly affect the resulting cluster layout. For this reason, it is often necessary to normalize or standardize the data before entering the algorithm.

There are several ways we can standardize data. In this section, we will introduce two basic approaches:

1. *min-max* - we subtract the minimum from each value and divide the result by the difference between the maximum and minimum values. Given the great influence of extreme values, we should have an idea of how the data in the normalized variable is distributed.
2. *z-score* - we subtract the average value from each value and divide the result by the standard deviation. The adjusted data shall have an average equal to zero and a standard deviation of 1.

We initialize *min_max* in the following way.

```
# prepared code
import pandas as pd
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans

file_url = 'https://priscilla.fitped.eu/data/1a/logs_lms.csv'
df = pd.read_csv(file_url,
usecols=['CourseID', 'Teachers_logs', 'Students_logs'])

from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
```

Similarly, we can initialize a method for data standardization.

```
from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()
```

In the next step, we will use them to edit data before entering the *k-means* algorithm.

Data modeling

The *k-means* algorithm is part of the *scikit-learn* library. As in the previous chapters, in the case of this algorithm, we first create an instance of the algorithm with defined hyperparameters, train the model on the training dataset, and then predict the results for the test dataset.

```
kmeans = KMeans(random_state=42)
```

We will only look at random variables with continuous numerical values. There is reason in using the *CourseID* variable as it is a nominal variable.

```
X = df[['Teachers_logs', 'Students_logs']]
```

We need to normalize or standardize the data before being used in an algorithm. We opted for the second option. Note that the listed methods of normalization differ in the sequence of calling the methods `fit()` and `transform()`, respectively `fit_transform()`.

```
# min_max_scaler.fit(X)
# X_min_max = min_max_scaler.transform(X)
# X_min_max
X_scaled = standard_scaler.fit_transform(X)
print(X_scaled)
```

Program output:

```
[[-0.49490091  1.47818338]
 [ 0.90726658  1.55236034]
 [-0.76271247 -0.02633256]
 ...
 [-0.30233549  0.09055617]
 [ 0.28830632  1.63487747]
 [-0.3978091   0.288572   ]]
```

The original values no longer reach the order of thousands, as it was in the original file, but are distributed around zero.

We can proceed to train the model with the default values of hyperparameters.

```
kmeans.fit(X_scaled)
```

After training the model, we can try to predict the resulting data, still on the training set.

```
y_preds = kmeans.predict(X_scaled)
print(y_preds)
```

Program output:

```
[1 7 0 ... 4 7 1]
```

As a result, we see the number of the cluster to which the given record will fall according to the algorithm. In accordance with the predefined values of the `n_cluster` parameter, 8 clusters were created.

```
df['cluster'] = y_preds
print(df.head())
```

Program output:

	CourseID	Teachers_logs	Students_logs	cluster
0	1	2171	27655	1
1	2	3904	28242	7
2	3	1840	15749	0
3	4	4017	54076	6
4	5	3533	32530	7

9.1.4

Results

Now we try to interpret the results in more detail. We should be able to justify the causes of the formation of individual clusters and their resulting number. We use a pivot table to see cluster centers. In this method, we must specify as parameters:

- *values* - expects numerical variables from which aggregation is to be calculated,
- *index* - specifies the columns for which we want to make individual aggregations,
- *aggfunc* - specifies the aggregate functions we want to use.

```
# prepared code
import pandas as pd
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()

file_url = 'https://priscilla.fitped.eu/data/1a/logs_lms.csv'
df = pd.read_csv(file_url,
usecols=['CourseID', 'Teachers_logs', 'Students_logs'])
kmeans = KMeans(random_state=42)
X = df[['Teachers_logs', 'Students_logs']]
X_scaled = standard_scaler.fit_transform(X)
kmeans.fit(X_scaled)
y_preds = kmeans.predict(X_scaled)
df['cluster'] = y_preds

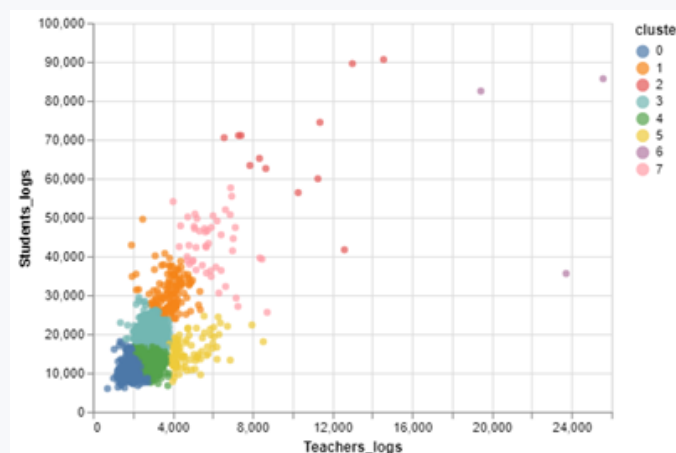
print(df.pivot_table(values=['Teachers_logs',
'Students_logs'], index='cluster', aggfunc=np.mean))
```


Program output:

	Students_logs	Teachers_logs
cluster		
0	11081.360939	2067.358467
1	20340.758475	2856.269068
2	15618.445652	4833.369565
3	68020.583333	9940.500000
4	13692.201099	2697.378022
5	67908.000000	22911.000000
6	42628.936170	5925.808511
7	31067.328125	3867.765625

The k-means algorithm created eight clusters. We will be better able to analyze them if we visualize individual clusters. Therefore, we assign a different color to each value of the *cluster* variable for better clarity, where :N means that it is a categorical variable.

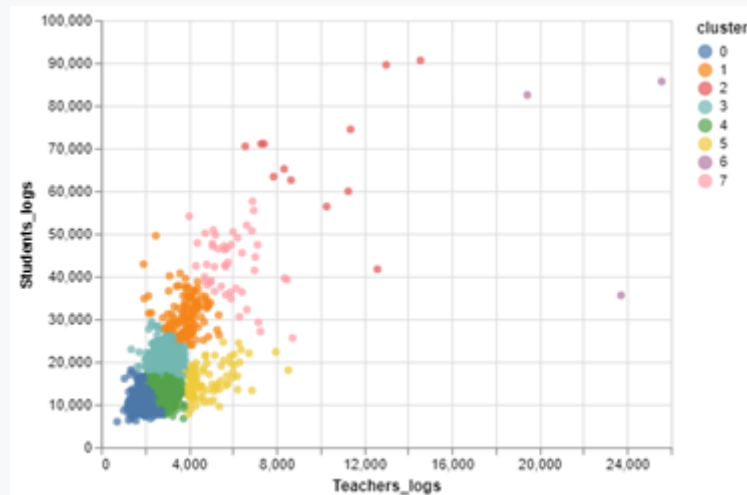
```
chart = alt.Chart(df)
scatter_plot = chart.mark_circle()
scatter_plot.encode(x='Teachers_logs', y='Students_logs',
color='cluster:N')
```



At first glance, we can see that the division of data into clusters happens mainly due to a variable on the x-axis and the boundaries between clusters are almost vertical. Thus, we suspect that the result will not be very useful, and we should take a closer look at the role of individual hyperparameters of the model.

If we want to see individual values in the chart in the form of a floating label, we modify the command as follows.

```
scatter_plot.encode(x='Teachers_logs',
y='Students_logs', color='cluster:N', tooltip=['cluster',
'Teachers_logs', 'Students_logs']).interactive()
```



9.2 Optimal number of clusters

📖 9.2.1

Find the optimal number of clusters

The unanswered question is why the algorithm created eight clusters. This brings us back to the issue of hyperparameters, i.e., parameters that should be set by a researcher based on his experience (So et al., 2020).

In the case of the *K-means* algorithm, the predefined number of clusters into which the algorithm should try to split the data set is set at eight. Specifically, we set this value in the hyperparameter `n_cluster`. Its final value must be chosen judiciously, since too few clusters group many unrelated observations together, and vice versa, too many very similar clusters.

Fortunately, we cannot find a suitable number of clusters only by gradual iteration. We can use the following methods that estimate the optimal number of clusters:

- the Elbow method,
- the Silhouette method,
- the MeanShift method,
- [fcluster](#) function from the *scipy* library, taking as a criterion the distance between clusters or simply their number.

Nevertheless, the final decision on how many clusters makes sense to us remains with us.

9.2.2

Elbow method

The *Elbow method* estimates the compactness of clusters by looking at the Sums of Squared Error (SEE) value as an *inertia* parameter for different cluster counts. Currently, this parameter has the following value.

```
# prepared code
import pandas as pd
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans

print(kmeans.inertia_)
```

Program output:

```
639.0214149347279
```

Let's examine how the value would change based on the different number of clusters defined in *n_cluster*. Assume values 1 through 9.

```
clusters = pd.DataFrame()
clusters['cluster_range'] = range(1, 10)
inertia = []
```

We use a cycle for the calculation, always storing the value of the *variable inertia*.

```
for k in clusters['cluster_range']:
    kmeans = KMeans(n_clusters=k, random_state=42).fit(X)
    inertia.append(kmeans.inertia_)
```

We assign the resulting values to other data that characterize clusters and visualize the relationship.

```
clusters['inertia'] = inertia
print(clusters)
```

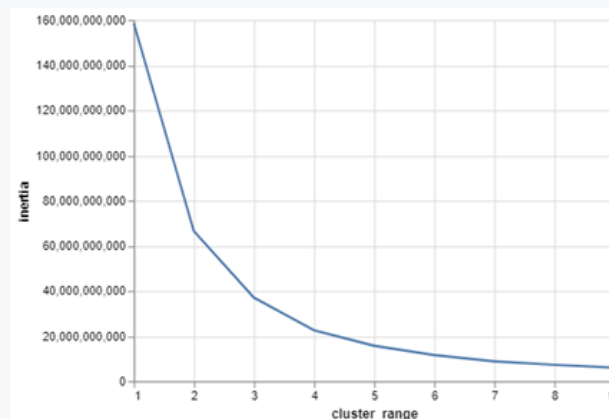
Program output:

	cluster_range	inertia
0	1	1.586459e+11
1	2	6.655286e+10

2	3	3.708316e+10
3	4	2.253880e+10
4	5	1.577041e+10
5	6	1.151998e+10
6	7	8.774815e+09
7	8	7.284518e+09
8	9	5.999906e+09

We can notice in the graph below, where the curve begins to descend at an angle of about 45°. We can see that this break corresponds to a value of 3 to 4. Therefore, we can expect, that the most suitable number of clusters in our case will be three or four.

```
alt.Chart(clusters).mark_line().encode(x='cluster_range',
y='inertia')
```



9.2.3

Silhouette Analysis

A more accurate estimate can be obtained by using Silhouette analysis, which works with distances between clusters and uses the following formula:

$$\text{silhouette score} = \frac{p - q}{\max(p, q)}$$

where, p is the average distance to points of the nearest cluster whose data are not part of the cluster being studied, q is the average distance between all points within the cluster.

This parameter takes values from the range -1 to 1.

- A value close to 1 indicates that the cluster data is very similar.
- A value close to -1 indicates that the data is not like each other in the cluster.

Like many others, this method is available in the Scikit-learn library.

```
# prepared code
import pandas as pd
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score
```

We calculate *silhouette_score* for different k-means values and plot the result in a graph.

```
sse_ = []
sse_.append([0,0])
for k in range(2,20):
    if k>1:
        kmeans = KMeans(n_clusters=k, random_state=42).fit(X)
        sse_.append([k-1,silhouette_score(X, kmeans.labels_)])
print(sse_)
```

Program output:

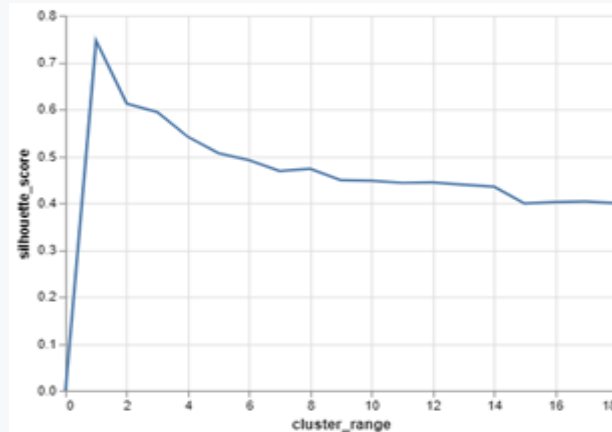
```
[[0, 0], [1, 0.7463511627179947], [2, 0.6119882102963004], [3,
0.5938317978467145], [4, 0.526336947741042], [5,
0.5052498105822446], [6, 0.5011951259746238], [7,
0.4649287296640222], [8, 0.47302950953842215], [9,
0.4495072599541057], [10, 0.44677786842469064], [11,
0.44155764870351577], [12, 0.44360631004928125], [13,
0.4412882648805847], [14, 0.43620287323903556], [15,
0.3909643947116154], [16, 0.3929404282349496], [17,
0.3899873583509344], [18, 0.383745190641355]]
```

```
df2 = pd.DataFrame(sse_,
columns=['cluster_range','silhouette_score'])
print(df2.head())
```

Program output:

	cluster_range	silhouette_score
0	0	0.000000
1	1	0.746351
2	2	0.611988
3	3	0.593832
4	4	0.526337

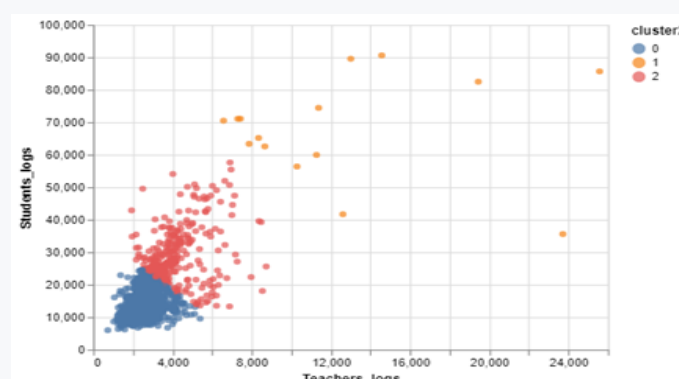
```
alt.Chart(df2).mark_line().encode(x='cluster_range',
y='silhouette_score')
```



Again, we see that the curve begins to converge somewhere around 3-4. Therefore, we will try to set the value of the hyperparameter `n_clusters` to 3

If we now introduce this hyperparameter into the model, we see three clusters.

```
kmeans = KMeans(random_state=42, n_clusters=3)
kmeans.fit(X_scaled)
df['cluster2'] = kmeans.predict(X_scaled)
scatter_plot.encode(x='Teachers_logs',
y='Students_logs', color='cluster2:N',
tooltip=['cluster', 'Teachers_logs', 'Students_logs']
).interactive()
```



9.2.4

Initialize clusters

Another interesting hyperparameter that can greatly affect the number and distribution of clusters is called *init*. Its baseline value is *k-means++* (So et al., 2020).

This parameter determines how the starting center of the cluster, called a centroid, is selected. It is obvious that if we selected the center of the cluster incorrectly, the resulting distribution of data into clusters would not be optimal. If we keep the default value of the *init* parameter, the algorithm first selects the first cluster and calculates the others using the probability spread. In some cases, it is justified to change the value of the *init* parameter to *k-means* or *random*.

Closely related to cluster initialization is another hyperparameter *n_init*, which defines how many times the algorithm tries to initialize the cluster.

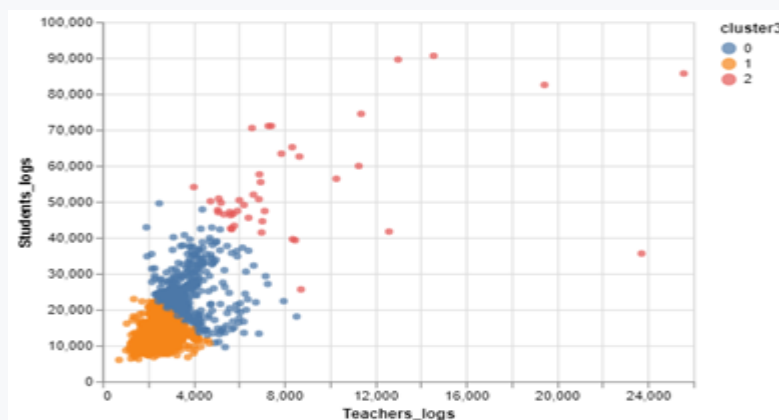
```
# prepared code
import pandas as pd
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans

kmeans = KMeans(random_state=42, n_clusters=3, init='random',
n_init=1)
print(kmeans.fit(X_scaled))
```

Program output:

```
KMeans(init='random', n_clusters=3, n_init=1, random_state=42)
```

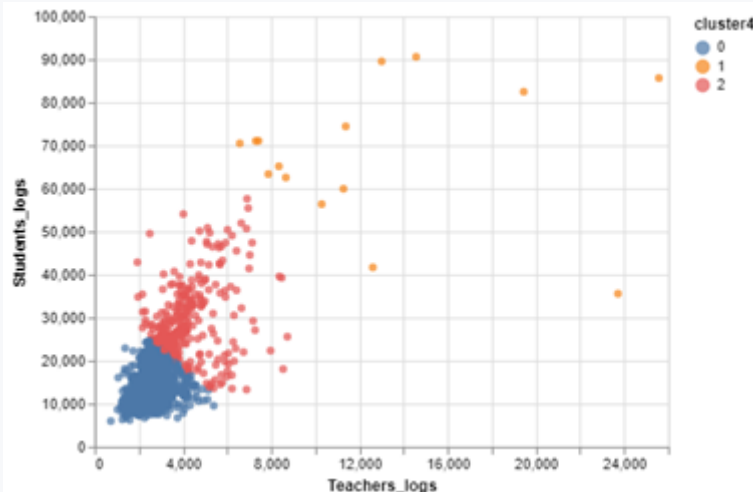
```
df['cluster3'] = kmeans.predict(X_scaled)
alt.Chart(df).mark_circle().encode(x='Teachers_logs',
y='Students_logs',color='cluster3:N',tooltip=['cluster',
'Teachers_logs', 'Students_logs']
).interactive()
```



The resulting shape of clusters seems like above.

```
kmeans = KMeans(random_state=42, n_clusters=3, init='k-means++', n_init=5)
kmeans.fit(X_scaled)
df['cluster4'] = kmeans.predict(X_scaled)
chart1 = alt.Chart(df).mark_circle().encode(x='Teachers_logs',
y='Students_logs', color='cluster4:N', tooltip=['cluster',
'Teachers_logs', 'Students_logs'])
```

chart1



9.2.5

Calculation of the distance between centroids

K-means groups data based on their similarity. This is closely related to the distance between them. Most often, this distance is determined based on calculating the distance of two points. More precisely, it represents the sum of the square of the difference of the x and y coordinates of the point in individual dimensions. In the case of two variables, we calculate the difference in the distance of two points in the plane.

The algorithm proceeds in the following steps:

- Randomly selects the center of clusters (centroids).
- Assigns each data point to the nearest centroid by calculating the Euclidean distance.
- Updates the coordinates of all centroids to a newly calculated centroid.
- Repeats steps 2 and 3 until the clusters converge or the maximum iteration count is reached.

We can find out the position of centroids directly from a trained model.

```
# prepared code
import pandas as pd
```



```
import numpy as np
import altair as alt
# alt.renderers.enable('notebook')
alt.renderers.enable('default')
from sklearn.cluster import KMeans

kmeans = KMeans(random_state=42, n_clusters=3, init='k-
means++', n_init=5)
kmeans.fit(X)
df['cluster5'] = kmeans.predict(X)
```

Now we will create a chart1, which will show the identified clusters.

```
chart1 = alt.Chart(df).mark_circle().encode(x='Teachers_logs',
y='Students_logs', color='cluster5:N',
      tooltip=['cluster', 'Teachers_logs', 'Students_logs'])
```

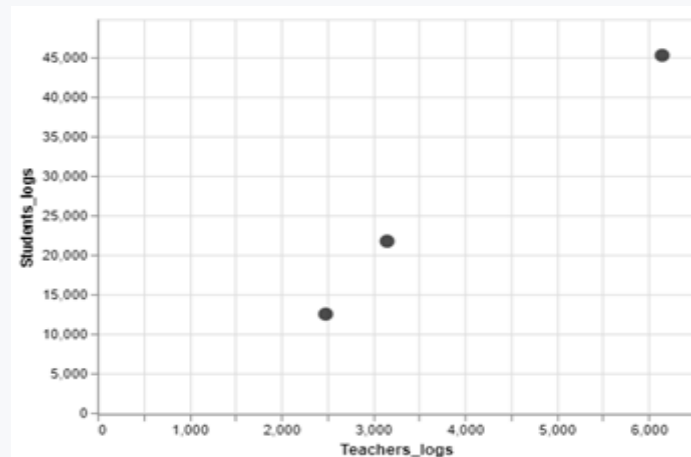
In the second *chart2*, we show the centroids that we first stored in the variable *centroids*.

```
centroids = kmeans.cluster_centers_
centroids = pd.DataFrame(centroids, columns=['Teachers_logs',
'Students_logs'])
print(centroids)
```

Program output:

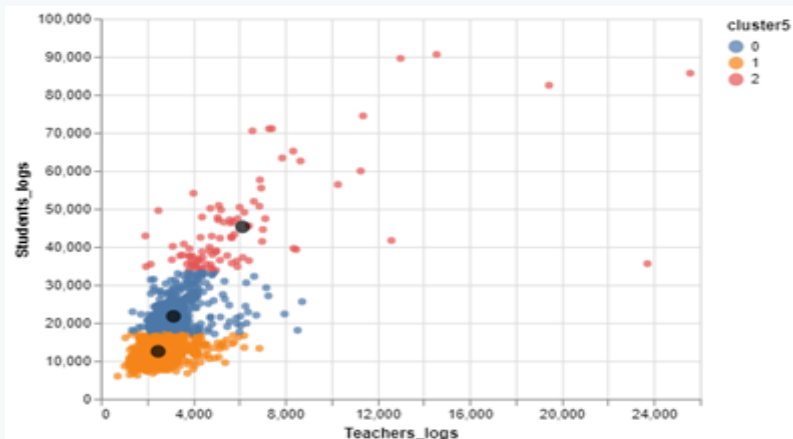
	Teachers_logs	Students_logs
0	2483.760674	12495.607303
1	3154.901993	21745.622924
2	6167.000000	45379.802198

```
chart2 =
alt.Chart(centroids).mark_circle(size=100).encode(x='Teachers_
logs', y='Students_logs',
color=alt.value('black'), tooltip=['Teachers_logs',
'Students_logs']).interactive()
```



Finally, let's combine both charts into one chart.

chart1 + chart2



We identified three clusters in this example. We must explain their importance and usefulness ourselves or in cooperation with a person who knows the background of the data we have processed. If we fail to logically explain the identified clusters, we can try a different expected number of clusters, the optimal number of which we estimated using the above methods.

At the same time, we compared two variables in our example. However, K-means can work with multiple variables, with clusters then arising in n-dimensional space. They are more difficult to interpret, which is why approaches are used that try to reduce the number of dimensions.

Association Rule Mining

Chapter **10**

10.1 Association Rule Mining

10.1.1

Association Rule Mining

In the next example, we will get acquainted with the application of selected methods of association rule analysis. We can use the results of this analysis in the field of education as follows:

- We can create an agent to recommend activities in the course,
- automatically guide the student through activities.
- intelligently generate and recommend study materials,
- identify attributes that characterize differences between groups of students,
- discover interesting relationships as the student uses available information resources,
- find relationships between found patterns and student behaviour,
- find frequently recurring mistakes that occur together,
- optimize the content of the e-learning portal by determining the most suitable content for the student,
- extract useful patterns of user behaviour to evaluate and interpret online activities in the course,
- personalize e-learning based on aggregation of usage profiles and domain ontology.

10.1.2

Problem Understanding

The association rules are among the best-researched methods of learning analytics and data mining research in general, helping teachers and course developers get detailed feedback:

- about the course of educational processes,
- how the student learns,
- how to evaluate a student based on navigation patterns,
- how to classify students into groups with similar behaviour or preferences,
- or how to personalise course content.

At the same time, they allow students to better interact with LMS, adapt the course based on their individual learning progress, recommend a personalized learning path to the student based on previous experience with similar students.

10.1.3

Metrics

The importance of rules is assessed using several characteristics, metrics:

Support is a ratio between the records that contain the studied set of items and all transactions in the data set. In other words, it represents the probability of the item occurring in individual transactions, with A standing for Antecedent and C for Consequent. The metric *support* is used to express the frequency of an item in the examined data set. If this frequency is greater than the minimum defined, we speak of a frequent item set, while it is true that all subsets of frequent item sets are equally frequent.

$$\text{support}(A \rightarrow C) = \text{support}(A \cup C), \quad \text{range: } [0, 1]$$

Confidence represents the conditional probability of a given combination of occurrences of items A and C in identified transactions, as opposed to *support* of a rule is oriented, *confidence* is equal to 1 if rule $A \rightarrow C$ always occur together.

$$\text{confidence}(A \rightarrow C) = \frac{\text{support}(A \rightarrow C)}{\text{support}(A)}, \quad \text{range: } [0, 1]$$

Lift can be defined as a correlation, which in other words means how many times a combination of items occurs more often together than if the items were statistically independent. If the items are independent, the *lift* will be equal to 1.

$$\text{lift}(A \rightarrow C) = \frac{\text{confidence}(A \rightarrow C)}{\text{support}(C)}, \quad \text{range: } [0, \infty]$$

Leverage is a less frequently used metric, representing the difference between observed frequent items A and C that occur together and the frequent sets that would be expected if items A and C were independent. Again, a value of 0 represents item independence in this metric.

$$\text{leverage}(A \rightarrow C) = \text{support}(A \rightarrow C) - \text{support}(A) \times \text{support}(C), \quad \text{range: } [-1, 1]$$

Conviction, its high values mean that the Consequent is heavily dependent on its Antecedent. We consider items to be independent if *the conviction* value is equal to 1.

$$\text{conviction}(A \rightarrow C) = \frac{1 - \text{support}(C)}{1 - \text{confidence}(A \rightarrow C)}, \quad \text{range: } [0, \infty]$$

10.1.4

Frequent Itemsets

Let's show you how easily we can calculate frequent itemsets and from them association rules for data that characterize individual user sessions in an e-learning course. We will not consider the time characteristics of the session. We are only interested in what activities and resources the students completed within each session in the course.

Association analysis is not directly integrated into the *scikit-learn* library. Therefore, we will use another interesting *mlxtend* library. First, let's import important libraries. We assume that we previously installed *the mlxtend* library in a standard way.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
```

Let's have the following transactions. Each line corresponds to one session of the student, i.e. a list of activities he visited during it.

```
dataset = [['zadanie1', 'test1', 'kniha1', 'zadanie2',
            'kniha2', 'test2'],
            ['test4', 'test1', 'kniha1', 'zadanie2', 'kniha2',
            'test2'],
            ['zadanie1', 'kniha3', 'zadanie2', 'kniha2'],
            ['zadanie1', 'zadanie3', 'test3', 'zadanie2',
            'test2'],
            ['zadanie3', 'test1', 'test2', 'zadanie2',
            'kniha3', 'kniha2']]
```

Using the *TransactionEncoder()* method, we transform them into a format suitable for further association analysis. This transformation creates a column from each unique item and transforms each transaction into a vector whose individual items take values 0 and 1, depending on whether or not the transaction contains the item that became the column name.

```
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
print(df)
```

Program output:

```
   kniha1  kniha2  kniha3  test1  test2  test3  test4
zadanie1  zadanie2  \
```

```

0    True    True    False    True    True    False    False
True          True
1    True    True    False    True    True    False    True
False          True
2    False    True    True    False    False    False    False
True          True
3    False    False    False    False    True    True    False
True          True
4    False    True    True    True    True    False    False
False          True

```

```

zadanie3
0    False
1    False
2    False
3     True
4     True

```

We used a combination of *fit()* and *transform()* methods and stored the result in a *df* variable of type Dataframe. The following table describes this transformation in more detail.

Now we will use the most common algorithm for finding frequent item sets, known as Apriori. As a parameter, expects the rule minimum support value that is characterized by the *min_support* parameter. For example, if we want to see frequent sets with support greater than 60%, we will use this limit as a parameter *min_support* method *a priori()*.

```

from mlxtend.frequent_patterns import apriori
print(apriori(df, min_support=0.6))

```

Program output:

```

      support  itemsets
0         0.8        (1)
1         0.6        (3)
2         0.8        (4)
3         0.6        (7)
4         1.0        (8)
5         0.6      (1, 3)
6         0.6      (1, 4)
7         0.8      (8, 1)
8         0.6      (3, 4)
9         0.6      (8, 3)
10        0.8      (8, 4)
11        0.6      (8, 7)

```

```

12      0.6      (1, 3, 4)
13      0.6      (8, 1, 3)
14      0.6      (8, 1, 4)
15      0.6      (8, 3, 4)
16      0.6      (8, 1, 3, 4)
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)

```

If we want to display column names instead of an index, we'll list this as an additional attribute. This view is certainly clearer and tells us how much support each frequent set has. We see high support for multiple multi-item sets. *Support* represents the probability of the item occurring in each transaction. Since this is a practice example with a small number of short transactions, it is not our job to interpret their contribution exactly.

```
print(apriori(df, min_support=0.6, use_colnames=True))
```

Program output:

	support	itemsets
0	0.8	(kniha2)
1	0.6	(test1)
2	0.8	(test2)
3	0.6	(zadanie1)
4	1.0	(zadanie2)
5	0.6	(test1, kniha2)
6	0.6	(test2, kniha2)
7	0.8	(zadanie2, kniha2)
8	0.6	(test2, test1)
9	0.6	(zadanie2, test1)
10	0.8	(test2, zadanie2)
11	0.6	(zadanie2, zadanie1)
12	0.6	(test2, test1, kniha2)
13	0.6	(zadanie2, test1, kniha2)
14	0.6	(test2, zadanie2, kniha2)
15	0.6	(test2, zadanie2, test1)
16	0.6	(test2, zadanie2, test1, kniha2)

```

/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the

```



```
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

Using filters that are part of the Pandas library, we can adjust the displayed result of identifying frequent sets to a more acceptable form. For example, if we want to display only two-item sets with support greater than 60%, we first create frequent sets using the Apriori algorithm and add a column with information about the number of items in the multicomponent set. To do this, we will use the *apply()* method and the *lambda* function. Note that the order of items in each frequent set does not matter.

```
frequent_itemsets = apriori(df, min_support=0.6,
use_colnames=True)
frequent_itemsets['length'] =
frequent_itemsets['itemsets'].apply(lambda x: len(x))
print(frequent_itemsets)
```

Program output:

	support	itemsets	length
0	0.8	(kniha2)	1
1	0.6	(test1)	1
2	0.8	(test2)	1
3	0.6	(zadanie1)	1
4	1.0	(zadanie2)	1
5	0.6	(test1, kniha2)	2
6	0.6	(test2, kniha2)	2
7	0.8	(zadanie2, kniha2)	2
8	0.6	(test2, test1)	2
9	0.6	(zadanie2, test1)	2
10	0.8	(test2, zadanie2)	2
11	0.6	(zadanie2, zadanie1)	2
12	0.6	(test2, test1, kniha2)	3
13	0.6	(zadanie2, test1, kniha2)	3
14	0.6	(test2, zadanie2, kniha2)	3
15	0.6	(test2, zadanie2, test1)	3
16	0.6	(test2, zadanie2, test1, kniha2)	4

```
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
```

```
and should_run_async(code)
```

In the table we see a list of one-, two-, three- and four-item sets that meet the minimum support rule (*min_support*). These multi-item sets can be found in the studied data set.

Now we can only select those items that meet our requirements, for example, two-item frequent flyers with a support value of *support* ≥ 0.75 .

```
print(frequent_itemsets[ (frequent_itemsets['length'] == 2) &
(frequent_itemsets['support'] >= 0.75)])
```

Program output:

```
support      itemsets  length
7          0.8  (zadanie2, kniha2)      2
10         0.8  (test2, zadanie2)      2
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
and should_run_async(code)
```

We can also do the opposite if we want to find all the frequent sets that contain specific items. We simply define the condition that each item stored in the data frame must meet.

```
print(frequent_itemsets[ frequent_itemsets['itemsets'] ==
{'kniha2', 'test1'} ])
```

Program output:

```
support      itemsets  length
5          0.6  (test1, kniha2)      2
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during thetransform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
and should_run_async(code)
```

For the sake of completeness, let us add that in order to save computational resources in cases where the examined data set contains a large number of different

items and especially short transactions, we can adjust the algorithm's progress using the parameter `sparse = True`, which we will use at the stage of preparing a data frame containing information about the occurrence of the item in the dataset.

```
oht_ary = te.fit(dataset).transform(dataset, sparse=True)
sparse_df = pd.DataFrame.sparse.from_spmatrix(oht_ary,
columns=te.columns_)
print(sparse_df)
```

Program output:

```
   kniha1 kniha2 kniha3 test1 test2 test3 test4
zadanie1 zadanie2 \
0         1         1         0         1         1         0         0
1         True
1         1         1         0         1         1         0         1
0         True
2         0         1         1         0         0         0         0
1         True
3         0         0         0         0         1         1         0
1         True
4         0         1         1         1         1         0         0
0         True

   zadanie3
0          0
1          0
2          0
3          1
4          1
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

Subsequently, we can again display frequent sets that meet a condition defined by us, such as minimum *min_support* support. The *verbose* parameter specifies whether to display the number of iterations. Its actual effect depends on the parameter *low_memory*. We will read in more detail about the advantages of additional parameters in the documentation (<http://rasbt.github.io/mlxtend/>).

```
print(apriori(sparse_df, min_support=0.6, use_colnames=True,
verbose=1))
```

Program output:

```
Processing 20 combinations | Sampling itemset size 2
Processing 21 combinations | Sampling itemset size 3
Processing 4 combinations | Sampling itemset size 4
    support                                itemsets
0         0.8                             (kniha2)
1         0.6                             (test1)
2         0.8                             (test2)
3         0.6                             (zadanie1)
4         1.0                             (zadanie2)
5         0.6                         (test1, kniha2)
6         0.6                         (test2, kniha2)
7         0.8                   (zadanie2, kniha2)
8         0.6                   (test2, test1)
9         0.6                   (zadanie2, test1)
10        0.8                   (test2, zadanie2)
11        0.6                   (zadanie2, zadanie1)
12        0.6                   (test2, test1, kniha2)
13        0.6                   (zadanie2, test1, kniha2)
14        0.6                   (test2, zadanie2, kniha2)
15        0.6                   (test2, zadanie2, test1)
16        0.6 (test2, zadanie2, test1, kniha2)
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call
`transform_cell` automatically in the future. Please pass the
result to `transformed_cell` argument and any exception that
happen during the transform in `preprocessing_exc_tuple` in
IPython 7.17 and above.
    and should_run_async(code)
```

10.1.5

Association Rules Mining

We can easily generate association rules from frequent sets. We use *support*, *confidence* or *lift* metrics to evaluate them.

If we want to generate association rules, we take as input the frequented item sets that we obtained by applying the Apriori algorithm.

```
# prepared code
dataset = [['zadanie1', 'test1', 'kniha1', 'zadanie2',
'kniha2', 'test2'],
```

```

        ['test4', 'test1', 'kniha1', 'zadanie2', 'kniha2',
'test2'],
        ['zadanie1', 'kniha3', 'zadanie2', 'kniha2'],
        ['zadanie1', 'zadanie3', 'test3', 'zadanie2',
'test2'],
        ['zadanie3', 'test1', 'test2', 'zadanie2',
'kniha3', 'kniha2']]

from mlxtend.frequent_patterns import association_rules

```

The `generate_rules()` method will then allow us to specify the metric we want to use and define its threshold minimum value that the rules must meet. Support, *confidence* and lift metrics are currently *supported*. For example, we may require rules to meet a *confidence* > 0.7.

```

association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.7)

```

By adding another restriction, we get rules that also meet the lift requirement. In the next row, before displaying the rules, we sorted so that the rules with the highest value are both *confidence* and *lift*.

```

rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=1.5)
rules = rules.sort_values(['confidence', 'lift'], ascending
=[False, False])
print(rules)

```

Program output:

	antecedents	consequents
antecedent support \		
0	(test2, kniha2)	(test1)
0.6		
1	(test1)	(test2, kniha2)
0.6		
2	(test2, zadanie2, kniha2)	(test1)
0.6		
3	(test2, kniha2)	(zadanie2, test1)
0.6		
4	(zadanie2, test1)	(test2, kniha2)
0.6		
5	(test1) (test2, zadanie2, kniha2)	
0.6		

	consequent	support	support	confidence	lift	leverage
conviction						
0		0.6	0.6	1.0	1.666667	0.24
inf						
1		0.6	0.6	1.0	1.666667	0.24
inf						
2		0.6	0.6	1.0	1.666667	0.24
inf						
3		0.6	0.6	1.0	1.666667	0.24
inf						
4		0.6	0.6	1.0	1.666667	0.24
inf						
5		0.6	0.6	1.0	1.666667	0.24
inf						
/usr/lib/python3/dist-packages/ipykernel/ipkernel.py:283:						
DeprecationWarning: `should_run_async` will not call						
`transform_cell` automatically in the future. Please pass the						
result to `transformed_cell` argument and any exception that						
happen during the transform in `preprocessing_exc_tuple` in						
IPython 7.17 and above.						
and should_run_async(code)						

The lack of use of association rules in the field of LA is the relatively high computational complexity of algorithms, problematic interpretation of acquired rules in terms of their division into inexplicable, trivial and useful, the absence of an understandable way to visualize them, as well as accessibility for different types of users.

Learning Analytics Research Topics

Chapter **11**

11.1 Research topics

11.1.1

Among the most common research topics in learning analytics, we can currently include the following topics:

- Performance prediction - the goal is to predict the student's grade and find out what variables have the greatest impact on the grade.
- Predicting early course dropout - the aim is to predict whether a student will pass the course and identify students who need help as soon as possible.
- Course design improvement - creating and developing a course that better meets the needs of students and teachers using data.
- Behavioral analysis - analysis of how students interact with the course.
- Ethical issues - problems that come with using the data generated by students.

We will summarise some interesting findings in the following parts.

11.1.2

Student's Performance Prediction

The researchers conducted various studies to build an appropriate model of student academic performance prediction for specific courses or subjects. These studies use different types of student data with different parameters to predict grade outcomes.

In general, among machine learning algorithms, logistic regression algorithms are the most popular approaches for predicting outcome grades (Baneres et al., 2019), followed by decision tree algorithms (Conjin et al., 2017).

Several studies have compared the performance of regression algorithms with many other classification algorithms. The objective of these comparisons was to identify the most effective approach for creating grading models of students' academic achievements. For example, in a study conducted (Jayaprakash et al., 2014), student variables extracted from different data sources were embedded in classification and regression algorithms to find students who were exposed to risks associated with studying. The results showed 50-75% accuracy of the models.

In the paper (Baneres et al., 2019), they applied various regression-based algorithms using data captured from log files of virtual learning environments. The results showed that the use of logistic regression leads to better predictions.

In addition, the authors (Pardo et al., 2016) used decision tree algorithms to construct their regression models. They used models to predict intermediate and final exam scores. Each model corresponded to a week of lessons during a thirteen-week

course. The models trained based on first-year engineering students' online interaction data and internal assessment scores were derived from online learning sources. The authors created models with predictions for each week. The paper was considered as a case study aimed at processing many numerical variables derived from student interactions. The overall classification error of the model varied in the range of 15-20%.

11.1.3

The authors (Queiroga et al., 2020) used decision tree, polynomial and logistic regression, which they designed to predict students' grades based on their CGPA, partial assessments, and attendance records. The results of the model evaluation showed that polynomial regression and logistic regression outperform decision tree models, where both models provide similar levels of accuracy.

In the paper (Al-Shehri et al., 2017), researchers sought to predict the final grade in a math subject to be able to select the right student for certain assignments using two methods, the k-nearest neighbor (KNN) model and the support vector algorithm (SVM) prediction model. The process of property selection was carried out by finding a correlation between the input variables and the grade. The results showed that both classifiers worked very well for this type of problem, but SVM slightly outperformed KNN with correlation coefficients of 0.96 and 0.95, respectively. The performance of the proposed techniques should also be evaluated regarding classification aspects not only regression. The SVM algorithm would not work very well with a larger dataset as it requires a long time to train the model. In this case, its good performance was due to a small dataset, but in the case of a larger one, it may work poorly.

The authors examined student achievement and found that the most influential factors were achieved points in university admissions and the number of failed exams during the first year. Research (Tempelaar et al., 2015) focused on courses in a combined form of study, and the results showed that the most significant variables in performance prediction are: forum usage, student creation of course content, tests, and number of materials viewed. The author (You, 2016) identified that regular attendance, late assignment submissions, number of course logins, and proof of viewing of learning material are not relevant for predicting student performance in online learning.

Likewise, the authors (Asif et al., 2017) found that students' use of forums or time intervals for accessing a course did not have much correlation with students' final grades. However, it is important to note that the scope of research and the data used have a big impact on which input variables are useful. In summary, active participation often leads to better performance, and especially in the online environment, separate tasks that demonstrate the student's interest are important.

11.1.4

Predicting early course dropout

High failure rates of students in different courses is considered a significant problem today. Therefore, this phenomenon is often researched, if it is possible to create a prediction model to identify students who are at risk of failure.

Kalaivani et al. (2019) conducted a comparative analysis between three selected classification algorithms: decision tree and naive bayes. The input dataset consisted of 497 records over 8 academic years. The data included various aspects of the students' records, including family background, previous academic records, and other demographic elements. The naive bayes algorithm showed a peak accuracy value of 71.3%. The model was intended to enable teachers to take early steps to help poor and average students improve their performance. The dataset was relatively small due to incomplete and missing values. The study could be extended by adding more data to improve overall accuracy.

In a study (Ma et al., 2019), they proposed a Multi-Instance Multi-Label (MIML) algorithm using a KNN technique. The research was conducted on at-risk students in a course with a strong correlation with previous courses in which they failed. In the research, they used online learning activities, failing to predict student performance. This was due to a small input dataset.

Anusha et al. (2019) used an input dataset containing information about computer science students for years 2015-2019, taking into account their academic performance. Instead of a classification model, they used a regression model. The proposed system predicted the outcome in a numerical way using KNN, decision tree, SVM, random forest, linear regression, and multilinear regression. In this case, multiple linear regression proved to be the optimal solution.

11.1.5

In the study (Kalaivani et al., 2017), they focused on predicting final grades from a semester project. A dataset with 1938 records was used. To increase accuracy, they introduced an improved boost algorithm. The results were compared with existing algorithms Adaboost (decision stump) and Adaboost (J48). The best accuracy achieved was 69% by the Adaboost algorithm(J48), with an unbalanced dataset posing the problem.

Amrieh et al. (2016) used a prediction model on a dataset collected from LMS. In addition to academic performance, demographic data were also taken into account. The result showed a strong impact of academic performance compared to demographic characteristics. Traditional classification algorithms were used, namely decision trees, naïve bayes, and a neural network. The algorithm of decision trees gave the best results. The model performed very well with 480 records and showed 80% accuracy but should be verified with a larger dataset.

Almasri et al. (2019) analyzed student performance in a 4-year bachelor's degree. The study only considered grades as an input, without considering any other characteristics. The dataset used consisted of a sample of 210 students. Naive bayes achieved excellent results with an accuracy of 83.63%, followed by neural network and random forest. Due to the difficulty of interpreting the results of the neural network and random forest, it was not possible to establish the cause of the poor prediction.

In a study (Daud et al., 2017), they examined the impact of input variables on predicting the performance of students who have scholarships. The dataset consisted of 23 selected variables and 776 records. In addition to academic performance, variables were mostly related to personal information, specifically family background. However, they did not have a significant influence on predicting their performance. But to build the model, the classifiers naive bayes, SVM and CART were used. SVM emerged as the best classifier with an f1 score of 0.867 compared to others. To improve accuracy, it would be necessary to supplement the input variables with others that would have a greater impact on the predicted variable.

In a study (Marbouti et al., 2018), they tried to identify at-risk students in a selected course during the semester. The authors used logistic regression, support vector machines (SVM), decision trees (DT), ANN, and the naive bayes classifier. The aim of the study was, among other things, to focus on false negatives and false positives. This study used input variables such as grades, projects, attendance, tests, weekly homework, and exams. The best model for predicting performance was the decision tree with 96.1% accuracy. All models studied had low accuracy when it came to identifying failed students. Since the research focused on at-risk students, the best models were naive bayes with 86.2%, SVM with 72.4 and logistic regression with 58.6%. The reason for the poor prediction was that the set contained only 10% of failed students.

11.1.6

In terms of the most used predictors of success, several basic categories can be created. In some studies (Estacio, Raga, 2017; Gašević et al., 2016) used student data coming from student information systems (IS) or other systems of an administrative rather than educational nature. These were demographic and descriptive data (gender, citizenship, residence, semester of study, etc.) or data on previous study success (results at secondary school, results of admission procedure, results in previous courses at university, etc.). Based on these predictors, it was also possible to draw other variables that could be used in the modelling process (e.g., attendance for full-time teaching, activity in full-time teaching, frequency of personal consultations with the teacher, etc.).

If we focus our attention only on data coming from LMS, two basic groups of the most frequently used predictors can be distinguished in the studies so far.

The first group consisted of numerous aspects of student activity in specific areas of the course. First, it is the sum of records of student activity (logs) within a specific

activity in LMS, such as tests, sources of study materials, assignments, etc. (Lu et al., 2018). However, a certain disadvantage of predictors of this type is the direct link to a specific LMS (e.g. Moodle) and, accordingly, the worse possibilities of generalizing such solutions between different LMS.

The second group consisted of general temporary characteristics of course attendance. These are different measures of how much a course is attended, and generality in this context means that it is the attendance of the course as a whole and does not distinguish, for example, the activity of students in different parts of the course in terms of content. Temporality then refers to the fact that these are traffic metrics that are in some way tied to the time, e.g. when visits occur, how regularly, etc. (AlJarrah et al., 2018).

There are also other studies that use specific types of predictors, while they do not fall into any of the categories defined above. However, they are significantly less represented in terms of numbers. Examples include authors Nakayama, Mutsuura, and Yamamoto (2017), who used specific measures of how well students write continuous study notes throughout the course. Another example might be simple numerical characteristics such as the number of discussion posts created, the number of replies to other students' posts, and others. (Sclater et al., 2017).

11.1.7

Course design improvement

Learning analytics can also provide teachers with tools to improve and streamline their courses. However, for this to be possible, a contextual framework is needed to help teachers better understand the outcomes provided by learning analytics.

A study (Costa et al., 2017) suggested learning design as a form of documentation. Learning design includes resources that students can access, tasks for students to complete, support mechanisms for educators to use, and checkpoints where analytical methods can be used.

In the paper (Tomasevic et al., 2020), researchers analyzed the way courses are delivered to meet standards and desired outcomes. This study concluded that student engagement in technology improved academic performance. The teaching material of the online course has played a significant role in student achievement. Access to online learning materials was an indicator of good student performance. Full-time students were compared with part-time students for two semesters. The result showed that students regularly attending online courses achieve better grades and grades. It was also found that there is a significant relationship between expertise and level of participation. The authors illustrated the results of the comparison through a graph, but the overall results are not processed precisely enough. There was a group that showed a high level of attitude and involvement in the course, but at the same time their achieved results on the exam were the worst. It could be possible to focus on the role played by time spent with online materials to improve academic performance.

In their study, Rientes & Toetenel (2016) looked at course design combined with prediction of early graduation and found that the primary predictor is many diverse activities.

In the study (Kizilcec et al., 2017), they developed a strategy for course design. Setting clear goals and strategically scheduling content has been found to work best. Furthermore, it has been proven that it is appropriate to create activities in which students have to evaluate their elaborated assignments themselves. The authors (Tabuenca et al., 2015) found that alerting students to track their time spent in a course and over individual activities positively affects their ability to manage time and leads to students using their time in courses much more effectively.

11.1.8

Behavioral analysis

Learning analytics can be used to identify learning strategies in online and blended forms of teaching. Currently, researchers in these areas are very interested in the problem of analyzing specific patterns of stakeholder interaction in LMS. This means that research focused on clustering attempts to characterize, for example, the current level of behavior of a group of students within a given content being studied.

Aljohani et al. (2019) used clustering techniques to determine which students in online courses had the highest and lowest levels of success. The k-means aggregation technique was used to divide students into classes based on variables such as their enrollment history and the number of attempts to complete the course. However, the aggregated data used in this study were processed incorrectly. During the training phase of the model, they consisted of a large number of blanks, and a relatively small number of variables were used to create the model.

Hooshyar et al. (2020) provided a detailed overview of learning context clustering algorithms. They designed an evaluation approach that automatically compared several clustering methods using multiple internal and external performance measurements on nine educational datasets of different sizes.

Cao et al. (2018) analyzed the records of 98 college students to gain insight into their activities when they were engaged in online learning. Results from the k-means clustering algorithm pointed to three clusters. Students in clusters 1 and 2 had above-average performance and high interaction, while students in cluster 3 showed poor interaction and poor performance patterns.

11.1.9

Another study (Bharara et al., 2018) used clustering methods to understand student behavior in courses. This research examined input variables that affect the performance of most students. An input file from the Moodle LMS was used, which contained 489 records with 16 variables. The k-means algorithm in this model was used for student interactions along with the involvement of students' parents in

tracking students' academic performance. The result showed that clustering works well for heterogeneous data type. However, along with other clustering techniques, other variables would need to be addressed.

Students' grades in courses can be predicted from forum data, as suggested by the authors (Hussain et al., 2018), who proposed a classification using various segmentation techniques combined with association rules. The results were very similar compared to standard machine learning algorithms.

In their study, the authors (Navarro et al., 2018) used a large dataset that contained no outliers to see which clustering technique was more effective in finding students with low levels of proficiency in LMS. This study used seven different clustering models and compared different evaluation metrics to compare their performance, such as: Dunn index, Silhouette Score and Davies-Bouldin score. The goal of the metrics was to determine which algorithm worked more efficiently. However, a particular disadvantage of this research was that missing data in individual variables was eliminated. Some of them may have contained useful information to provide more information. In this way, one of the potential benefits of the research, as well as one of its limitations, has been eliminated. The original data was cleaned up to 44% of the total (Navarro et al., 2018).

11.1.10

Alternative research has provided a method of clustering students (Gupta et al., 2020). Three courses with a small number of students (15, 30 and 56 students) were used to evaluate the performance of four different clustering algorithms (x-means, k-means, hierarchical and expectation maximisation). The algorithms identified a minimum number of clusters (2 or 3 depending on the algorithm used), and the authors did not observe groups with obvious behavioral differences. The main reason for this was the relatively small number of students involved. In addition, these students tended to group themselves into similar groups based on characteristics that did not fit into the model.

When evaluating student activity data from Moodle LMS log files, the authors of Palani et al. (2021) compared three different clustering methods (k-means, hierarchical, and louvain) to determine the most effective segmentation method. Based on their findings, the louvain algorithm overcame k-means and hierarchical clustering.

Luna et al. (2017) designed an MDM tool integrated into the Moodle LMS that supported the entire knowledge discovery process. They did not describe details about student clustering methods.

The authors of the paper (Saiz-Manzanares et al., 2021) developed a desktop application that used visualization techniques and cluster analysis modules (k-means++, fuzzy k-means, and DBSCAN) to monitor students and predict dropout rates. They did not use LMS log files as input.

Development EdTech with AI/ML/DL Model

Chapter **12**

12.1 LA architecture

12.1.1

The combination of software, hardware and a specific form of education or educational approach has recently been referred to as educational technology. Software applications, which are its most tangible part, are referred to as EdTech applications. The development of Edtech applications has recently experienced a rapid boom, and we place similar requirements and expectations on them as in other areas of software deployment:

- Achieving an acceptable level of quality,
- support for standardization,
- implementation of artificial intelligence and machine learning elements,
- appropriate deployment of augmented and virtual reality,
- focus on STEM, programming, and gamification.

From the perspective of the developer of Edtech applications, events in this area focus mainly on the development of the following types of applications:

- virtual learning environments, including new generation learning environment (NGLE),
- digital publishing platforms for open educational resources (OER),
- implementation of gamification into e-learning solutions,
- online communication environments,
- an environment for automated evaluation and evaluation of learning outcomes,
- management and administration of the educational process and support of the management of the educational organisation.

Therefore, in the development of new educational applications we should know the theoretical basis of these trends and be able to incorporate them in an appropriate way. In this chapter, we will learn about the basic architecture of LA, which allows you to collect, store and further provide data related to students and their activities in the learning process, as well as exchange this data in a standardized format. Subsequently, we will summarize the basic rules that we should follow if we are thinking about developing software in the domain of education that will implement selected machine learning algorithms.

12.1.2

Learning Analytics Architecture

Learning analytics is a complex area that connects many different data ecosystems, standards, data-intensive applications, processes as well as different types of stakeholders. Sclater (Sclater 2017) stated that given the diversity of LA applications, the complexity of the issue, and the interests of stakeholders, we cannot expect a

simple consensus on what components and processes should be included in the conceptual model of LA architecture to provide a framework for its universal deployment.

On the other hand, Siemens et al. (Siemens, Gasevic, Haythornthwaite, Dawson, Shum, Ferguson, Duval, Verbert, and Baker 2011a) emphasize that for the development and wider adaptation of LA, it is essential to strive to create just such an architecture. This can take the form of an integrated toolkit, available as an open platform, bringing together the interests of all key user groups:

- students,
- educators,
- administrators,
- researchers,
- developers,
- as well as managers and analysts.

Recently, several initiatives have emerged that agree on what this LA architecture and its core software components might look like and which of the data science processes they should cover.

12.1.3

Experience API

The new lightweight Experience API (xAPI) standard plays an important role in standardizing the entire process. It is a communication protocol that connects all components of architecture (Martin 2018). The basic concept of xAPI assumes that people learn from interactions with other people and with study content. These interactions can occur anywhere. Tracking of learning-related events is no longer tied to WBES but can occur wherever a student is currently and works with a device that he or she decides to use for learning. If this device can store and exchange data using xAPI, it becomes part of the ecosystem. In order to combine event data with each other, the same event is described in the same way, regardless of what system or device the student uses. The interaction is recorded using xAPI in the format "subject, predicate, object" to LRS (Silver 2016).

Experience API (xAPI) is an extended and flexible standard suitable not only for educational technologies. It defines the way education systems describe user activities and how these systems communicate with each other. Thanks to xAPI, we can standardize most of the tasks we perform in an educational organization to objectively evaluate the effectiveness of education, for example, we can:

- better understand the steps leading to experience,
- learn more about the context in which learning takes place,
- Improve performance,
- find a correlation between learning and performance or learning and learning outcomes,

- personalize training,
- learn from each other,
- share experiences with other,
- compare performance and outcomes across organisations and learners,
- share learners' activity between multiple platforms and tools supporting the learning process in a standard form.

12.1.4

Software clients that implement xAPI can read and write data in the form of JSON objects. Activity information is stored in standardized forms. The basic form contains information about the actor, verb, and object, but it can also contain other metadata such as results, score, language, platform, or GPS coordinates. An example of a basic form of writing can be "John read a manual for safe work with chemicals" (Delano, Shahrazdat, 2013).

An actor is a person who performs the activity and is identified within the system. A verb defines a type of activity such as reading, accomplished, taught. An object represents what the record refers to, such as study material or a test.

The xAPI itself consists of two parts. The first part consists of an API that can be implemented in educational applications, including mobile ones. The second part consists of LRS (Learning Record Store) or LRW (Learning Record Warehouse) repository. The aim of LRS and LRW is to record user results and activities. If someone in an application that uses xAPI completes the task of reaching a higher level of play, the interface sends the information to the appropriate LRS in a standardized form. In LRS, all information is collected and can be evaluated, for example, by an independent analytical or reporting tool (Brdička, 2014).

12.1.5

Let's consider the above process in more detail. As mentioned above, xAPI is a JSON-encoded expression. This means that if we want to write xAPI commands, we should know the basics and rules of writing JSON. The xAPI command is a JSON object that contains at least three parts:

- actor,
- verb,
- object.

All three are again JSON objects, so we nest objects into each other.

```
{
  "actor": {
  },
  "verb": {
  },
  "object": {
```

```
}  
}
```

The first part of the expression is *actor*. The value of this object tells us who performed the action. It can contain several properties, in a minimal configuration it should be the name of the actor and the email (*mbox*), which performs the role of a unique identifier.

```
{  
  "actor": {  
    "name": "Jožko Mrkvička",  
    "mbox": "mailto:jozko.mrkvicka@ukf.sk"  
  },  
  "verb": {  
  },  
  "object": {  
  }  
}
```

The *object* verb (*verb*) defines what kind of action the actor performed. It contains *id* and *display* properties. The *id* key is a unique resource identifier (URI), most commonly a URL where a dictionary that defines each verb is available. Using a common dictionary of verbs that we can use to describe activities will ensure that we can combine entries stored as xAPI from different sources if necessary. Thanks to this, we can get a better idea of how the student learns using various applications and systems supporting the learning process.

The second key is labeled *display*. The latter provides a description of the verb in a readable form. Often, therefore, it takes the form of another pair in the shape of *key: value* so that we are able to provide multilingual content.

```
{  
  "actor": {  
    "name": "Joe Doe",  
    "mbox": "mailto:joe.doe@ukf.sk"  
  },  
  "verb": {  
    "id": "https://w3id.org/xapi/dod-isd/verbs/submitted",  
    "display": { "en-EN": "submitted" }  
  },  
  "object": {  
  }  
}
```

The last of the three basic objects of the xAPI expression is the *object*, which most often contains information about the result of the activity that the user performed. At

the minimum, it contains *the id* and *definition* properties. Again, the first id key is a URI meant to point to a unique address with an explanation. We can use our own expository dictionary, but we recommend using a standardized dictionary, such as Activity Types from the xAPI Vocab Server.

```
{
  "actor": {
    "name": "Jožko Mrkvička",
    "mbox": "mailto:jozko.mrkvicka@ukf.sk"
  },
  "verb": {
    "id": "https://w3id.org/xapi/dod-isd/verbs/submitted",
    "display": { "sk-SK": "zaslaný" }
  },
  "object": {
    "id": " http://id.tincanapi.com/activitytype/school-assignment",
    "definition": {
      "name": { "sk-SK": "Odovzdanie programu Tic-Tac-Toe" }
    }
  }
}
```

The *definition* key then contains the *name* property, which performs a similar function to the display property, providing a clear description of the subject of the activity. The final JSON shape of the xAPI command then takes the following form.

12.1.6

In addition to the mandatory parts of the xAPI command mentioned above, we have the option to add other nested JSON objects, for example:

- result of action,
- the context in which the action took place (context),
- existing attachments,
- statements.

Each of them contains distinctive features that illustrate the overall semantics of the user's activity associated with learning. In order to comply with the currently valid rules for creating more advanced xAPI expressions, we can use one of the available xAPI editors.

The resulting xAPI expression can be stored directly in the LMS or, in accordance with current trends, written into a common LRS or LRW repository for further analysis or needs of other applications.

In case the application or system stores records in its own format, we can consider the xAPI example as a suggestion how to export these logs for further analysis in a standardized form. As an example, we can explore a specific implementation of xAPI in the Moodle LMS.

12.2 Open LA architecture

12.2.1

Open Learning Analytics Architecture

The Society for Research for Learning Analytics (SoLAR) initiative introduced the Open Learning Analytics (OLA) Project, which can be considered the first attempt to design the LA architecture. The result of their systematic efforts is the conceptual design of an integrated LA system that includes the following modules (Siemens, Gasevic, Haythornthwaite, Dawson, Shum, Ferguson, Duval, Verbert and Baker 2011):

- The Learning Adaptation and Personalization Engine is the basis for collecting, identifying, and subsequently processing data by suitable available analytical modules. The adaptive part of the module covers the learning process, instructional design and educational content.
- The Analytics Module provides a set of predictive models in PMML format, allowing additional modules developed by third parties to be easily added.
- The Intervention Module monitors the progress of the learning process and provides various forms of automated and instructor-driven intervention using various predictive models created in the analytical module.
- Dashboard, reporting tools and visualization tools provide a comprehensible interface of the entire system, helping individual stakeholders to make better decisions. It provides the perspective of the learner, learner, researcher, and institution.

12.2.2

Although it is only a conceptual design of architecture, it provided ample space for further research and development of technologies and standards (Griffiths, Hoel and Cooper 2016). It also established the idea of a three-layer conceptual model of the LA platform, which includes:

- Activity Providers (APs) - systems and applications in which learning activities and events take place create data in xAPI format and send it to LRS.
- Learning Record Store (LRS) - a robust scalable database system designed specifically for storing educational data, which can exist standalone or as part of WBES. If a system or application generates data with the Experience API specification (xAPI or Tin Can), which we will introduce later, LRS can log almost any activity. LRS verifies that the input format conforms to the xAPI specification, stores all valid data in that format, and provides it for further

processing to activity consumers. In addition, it may share this data with other LRS.

- Activity Consumers (AC) - similar systems to activity providers, usually systems and applications that present data in a comprehensible form and ensure an appropriate form of intervention. They also work with data in xAPI format (Betts and Smith 2018).

12.2.3

The Learning Analytics Initiative (LAI), hosted by the Apereo Foundation, is intensively developing an open platform for LA with the following elements (Sclater 2014):

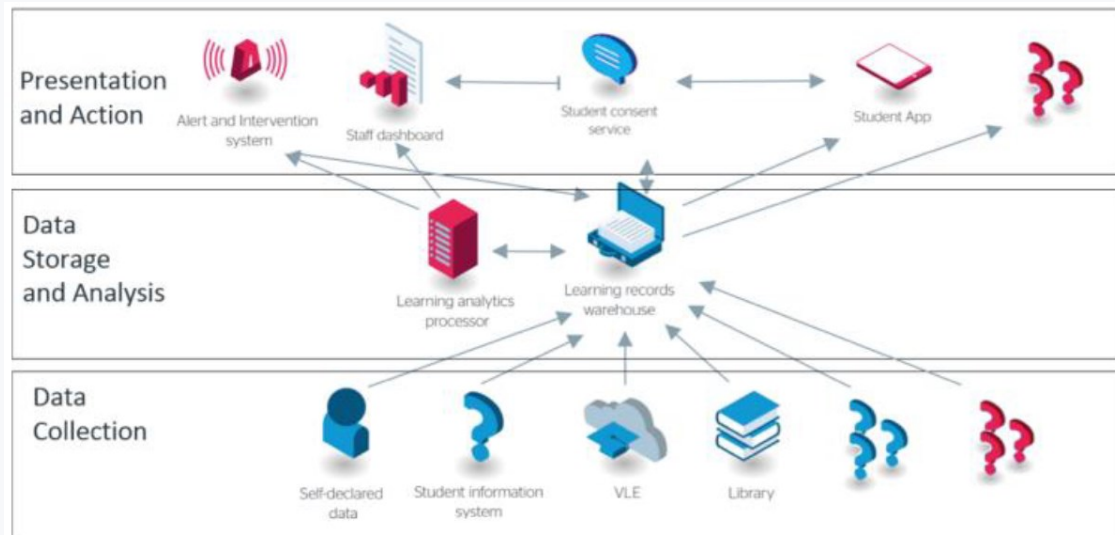
- collection - a collection of data built on xAPI, IMS Calipher/Sensor API standards,
- LRS-based repository,
- analysis - analytical processor enabling creation of reports and deployment of advanced analytical methods,
- action module providing various forms of alerts, interventions,
- communication – visualization of analytical processor and action module outputs.

12.2.4

These examples of good practice have been significantly developed by JISC, which represents a positive example of systematic development of national architecture for the LA domain. Architecture created primarily on open standards is intended for educational institutions in the UK. The JISC maintains a set of standards, models, and protocols by which LA's national open architecture is developed (Sclater, Berg, and Webb 2015). The architecture consists of three layers and distinguishes the following basic and optional elements:

- The data collection layer collects data about a student and their activity, for example from VLE, mobile apps, wearables, etc. Student data is typically stored in the Student Record System (SRS) in a standard data format called Universal Data Definition (UDD). Subsequently, the data is transferred via a standard ETL process to the LRW.
- The data storage and analytics layer consists of a central data warehouse called the Learning Records Warehouse (LRW), which, unlike LRS, allows you to store data in both structured and unstructured formats. In addition, students can make decisions and enter additional data, e.g. from wearable technologies (Betts and Smith 2018).
- The Learning Analytics processor pulls data from the LRW and runs predictive models. On their basis, it coordinates the actions of the warning and intervention system of the next layer. The Learning Analytics processor includes a library of open predictive models that can be shared across higher education institutions, allowing institutions to work together to refine those models over time.

- The presentation and action layer provides visualization of analyses in the form of bulletin boards for all stakeholders (Adams Becker, Cummins, Davis, Freeman, Hall Giesinger and Ananthanarayanan 2017).
- The student app allows students to view their own data and compare it with others. A key student consent service helps ensure privacy by allowing students to manage access and grant permissions to collect and use data (Sclater 2014).



12.2.5

The Learning Record Store (LRS) is an important component intensively used in monitoring students in the learning process, using the aforementioned xAPI standard, allowing systems, including LMS Moodle, to combine and exchange records of the activities of all stakeholders. LRS can be implemented in a particular LMS, but more often we will encounter it in the form of a separate system. For example, a list of available LRS supporting current standards in the LA domain can be found on the ADL website. It is the current development that is moving towards greater decentralization and personalization, as a reaction to the fact that only part of the learning process takes place in LMS. The xAPI-powered ecosystem then introduces a NoSQL database that can store and accessing data in xAPI format. If we are deciding how to integrate xAPI correctly, we can select any of the following variants:

- LRS is part of the LMS,
- LMS is an activity provider,
- LMS a LRS coexist together,
- sole use LRS,
- LRS used in combination with a selected data analysis or business intelligence tool.

12.3 Software development

12.3.1

Software Development Lifecycle EdTech

However, adding algorithms of artificial intelligence or machine learning significantly increases the complexity of the whole process of developing an educational software project.

The classic software development life cycle (SDLC) is a generalization of software development process phases, which takes into account examples of good practice. In general, SDLC is iterative and incremental in nature, comprising the following phases:

- analysis of requirements,
- planning
- design of architecture and subsystems,
- implementation
- testing
- deployment
- maintenance.

In the case of agile approaches, those phases are equally present at the scale required by the specific state of software development in each iteration or sprint.

On the other hand, the life cycle of a regular ML project has the following phases:

- business or problem understanding,
- data acquisition and preparation,
- exploratory analysis,
- model training,
- selection of suitable models and their evaluation and verification,
- deployment,
- maintenance.

12.3.2

If we want to successfully combine both areas and create an application that supports machine learning or artificial intelligence, we should thoroughly understand the differences and identify the appropriate way to connect both life cycles.

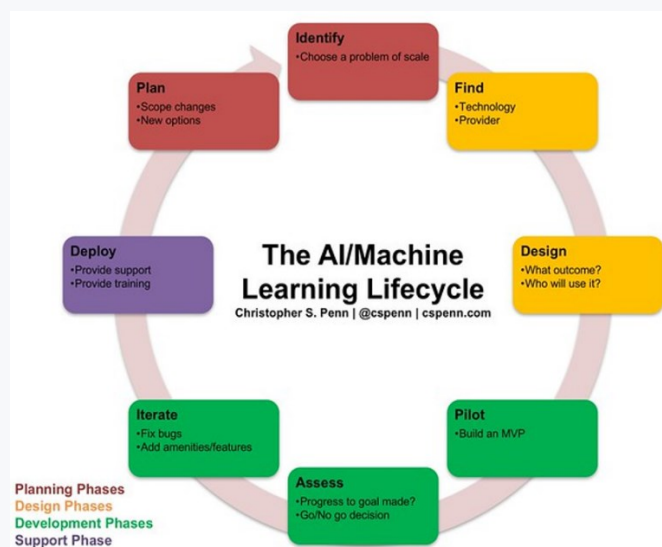
First of all, we must take into account that at the end of development there should be software that meets the expected requirements. Therefore, the software should work reliably until they change radically. In an analytically oriented project, we create a model based on available data. If the characteristics of this data on which the model is trained change, it may cease to provide accurate results. This phenomenon

is referred to as drift and can, for example, lead to incorrect classification of objects, photographs, etc.

Therefore, in terms of maintenance and monitoring, we regularly monitor the availability and reliability of software, eliminate found errors. For a deployed machine learning model, we must regularly monitor data integrity, data distribution, updating all libraries and their dependencies, metrics describing model performance, and the infrastructure that enabled the model to be made available as a service or application.

During the life cycle of the development of the ML model, we need to keep in mind the origin of data and their natural changes over time, track and adapt the source code, debug hyperparameters and performance metrics. At the same time, we should be able to explain, both internally and externally, how the model currently deployed in production was created. The last important requirement is that we should be open to collaboration and be able to reproduce individual iterations of development, even when changing the composition of the project team.

The task of software engineering is to automate the task using a computer by writing the rules that the computer should follow to solve the task reliably. On the other hand, the goal of machine learning is to automate the task of writing these rules itself, that is, the idea that the computer will find a program that fits the data obtained. How can we combine both approaches in one application?



12.3.3

Again, we can use a process framework that considers examples of good practice and recommendations from developers who have summarized their experiences in the following stages (Penn, 2017):

At the beginning, we will define why we should use artificial intelligence in the application at all. Most of the time, artificial intelligence is only one of the possible solutions and carries several risks. In addition, it places increased demands on the

development team and experience in knowledge transfer between multiple areas of expertise.

We choose which decisions we want to automate in the application in the sense that we rely on the solution we arrive at with the help of artificial intelligence algorithms. These algorithms are not just a simple set of rules, but they have the ability to learn from their own experience. At the same time, we will consider which part of the decision to leave to users and what to AI.

We carefully select data suitable as input to artificial intelligence algorithms. There is no simple rule, the more data – the better and more accurate, as we have seen in previous chapters, it is rather the opposite. Therefore, we first obtain data from primary sources that we can safely store, transform, update and optimize. We will provide space for communication between the following members of the development team:

- Data Engineer responsible for establishing the data channel.
- AI/ML Solution Architect who trains created models to perform a given task.
- Software Engineer, who helps bridge ML processes with other software processes.
- project manager responsible for the continuity of development and communication with the client.
- We identify which specific AI capabilities we need. The result should be a rough estimate of the size of the solution. This is because artificial intelligence provides a variety of approaches, among which we should decide:
 - machine learning,
 - natural language processing,
 - expert systems,
 - computer vision,
 - speech processing,
 - robotics,
 - autonomous decision-making.
- We will agree on a suitable SDLC model with regard to the most comprehensive list of requirements. The classic waterfall model in this case seems very suitable, at least for the initial stages of development since it requires precise outputs of individual phases. Of course, it can be appropriately combined with agile project management best practices.
- When defining a requirement, we rely on the experience of analysts who can assess future customer behavior.
- The design of the architecture and platform of the application means another crucial decision that requires experienced developers who transform the defined requirements into specific specifications and consider user behavior with respect to the specifics of the selected AI development platform.
- During the creation of the solution, we need a complete development team that knows the chosen AI development platform.
- Also, during the testing phase, we need experienced testers and DevOps in the team, because in addition to standard testing approaches, we should take into account the volume and complexity of available test data, the incorrect use of

which can cause problems with the accuracy of the created model. In addition, this team should be able to test AI and ML algorithms so that their desired characteristics are maintained.

- The deployment of the solution assumes that we know the specifics that the implementation of the AI/ML solution brings, and we can transfer them to the team that will take care of the operation of the created software solution.
- Finally, the management and further development of the resulting solution requires that, if necessary, we have sufficient capacities that we would be able to deploy operationally to eliminate the problem, whether at the application or AI/ML level.

12.4 Project management

12.4.1

EdTech Software Development Project Management

As mentioned above, as in the case of common software product development, in the case of software development for the education domain or LA directly, project management is iterative and incremental in nature, in which we should take into account best practices, often covered by an appropriate process framework. The nature and speed of each iteration will depend on the complexity of the problem and the "quality" of the data.

We can use the general CRISP-DM methodology for project management, possibly suitably supplemented with elements taken from agile approaches. We described both above.

If we now focus on adapting these software development methodologies for the domain of education or LA, we can look for inspiration in approaches aimed at developing applications using AI/ML algorithms (Geron, 2017; Tyagi, 2020). These approaches are more specific than the CRISP-DM methodology because they define the tasks that the development team should address at each stage of development. In addition, in some cases, they also contain instructions on which tools to use.

Therefore, we will first introduce the individual phases based on CRISP-DM in this chapter. In the next chapter, we will then concretize them into the form of individual tasks.

12.4.2

Business Understanding

Defining and understanding the problem from a higher, more general perspective will allow us to clarify the nature of the problem being studied (supervised or unsupervised machine learning), identify the type of task (classification, regression),

type of solution, choose a suitable metrics, consider whether machine learning is the right approach to solving a given problem, examine assumptions.

12.4.3

Identification of Data Resources and Data Pipeline

This step can take place in parallel or even precede the stage of understanding the problem. In any case, it is worth thinking through the whole procedure of obtaining and combining data from different sources. It is often a lengthy process in which if we make a mistake or forget something, we can get inconsistent or incomplete data and we will have to repeat the whole process.

If we plan to implement the results of data analysis into a software application, we should consider creating a data pipeline (ETL) through which we will add data continuously or in batches.

During this process:

- we create a list of available data sources,
- we provide enough space for their storage and processing,
- we verify whether we have the necessary permissions to the data,
- we obtain data in a suitable format,
- we check data types,
- we set aside some data for model verification.

12.4.4

Data Understanding

In this phase, we will focus on all data characteristics that may affect the result. Using the selected tool, we examine individual variables, find out their type (categorical, continuous, ordinal, ...), choose a suitable target variable, verify the correlation between independent variables, examine metadata. Often, we do not work with the entire data set, but with a representative sample of it.

12.4.5

Exploratory Data Analysis

During this phase, we will prepare methods with which we will perform individual steps of data pre-processing, such as data transformation, removal or replacement of missing values, selection of a subset of suitable variables, removal of extreme values, scaling and normalization.

12.4.6

Base Model

By creating a simple model, we verify the feasibility of the entire project. At this stage, our goal is to:

- Train several common machine learning models such as linear regression, SVM, Bayes classifier using predefined hyperparameters.
- Compare their performance against the chosen basic simple model and with each other.
- Perform k-fold cross-validation for each of the models created and calculate the mean and standard deviation.
- Examine individual independent variables (features) and see what impact they have on the target variable.
- Analyse the types of model errors.
- Process, or reduce, the number of independent variables entering the model using Principal Component Analysis or Independent Component Analysis.
- Repeat the above steps to create a final list of models that look promising.

12.4.7

Hyperparameter Tuning

In this very important step, we will focus on selecting suitable hyperparameters using cross-validation, using automated *grid search* and *random search methods* of the Scikit-learn library. At the same time, we can try to connect several methods as in the ensemble learning approach. We should use as much test data as possible at this stage. Finally, we will use a test sample of data from another dataset to validate and examine whether our proposed model is overfitted or underfitted.

12.4.8

Evaluation

This stage is often underestimated and not finished, although it brings multiple benefits to all stakeholders. We make it easier to return to the model in the future by creating documentation for the created code. In addition, it is recommended processing the created model into the form of a blog or tutorial, of course, if the nature of the data or research allows it. In addition, file sharing in a Github repository is a matter of course today.

12.4.9

Model Deployment, Maintenance, Bias, Further Development

If our project requires deployment and work with new data, we can create a web application or web service using the REST API, ensuring that the model is available to other users or clients. In this case, let's not forget:

- save the latest version of the code as binaries for example using Pickle,
- provide a model using web services, for example using Flask,
- connect to the data source by setting up an ETL data channel,
- manage dependencies using docker/Kubernetes,
- alternatively, we can use the services of a selected cloud (AWS, Azure),
- continuously monitor data availability and use of the created solution.



PRISCILLA



priscilla.fitped.eu