jamk

Exploring Machine Learning in Higher Education Industry

Student Performance Prediction

Ari Sivula

Master's thesis
May 2021
Information and Communication Technologies
Master of Engineering, Artificial Intelligence and Data Analytics

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Sivula Ari

Exploring Machine Learning in Higher Education Industry
Student Performance Prediction

Jyväskylä: JAMK University of Applied Sciences, May 2021, 78 pages.

Information and Communication Technologies. Master's Degree Programme in Artificial Intelligence and Data Analytics. Master's thesis.

Permission for web publication: Yes

Language of publication: English

Abstract

Machine learning (ML) is utilized constantly in various industries because its possibility to provide innovative solutions to different stakeholders of an organization. ML is utilized in higher education industry to provide insights and supporting activities of an educational institution. The higher education industry organizations have commonly several data sources, which they can adapt in their activities. These systems provide the data, which can be utilized with machine learning algorithms.

The main objective of this study is to explore the ML in higher education industry. Moreover, the objective is to provide an example of ML-based project and its implementation utilizing ML project management approach. The CRISP-DM was selected an approach to implement the development task and answer the research questions. Several supervised and unsupervised ML algorithms were utilized during the research process. The research exists about ML utilization in higher education industries, but each research is conducting a different type of contribution, because of datasets and contexts. This study provides a ML-based solution related to the VIRTA data and systems, which are commonly utilized in a higher education industry organization’s in Finland.

The implementation of ML project was a success in overall and the deployment of the models were implemented in this study. The results of this study indicate that CRISP-DM approach can be adapted in higher education industry in several ways and, moreover, machine learning provides value in student performance prediction when appropriate algorithms are developed based on the requirements of an organization and its data. The results of this study can be adapted as well other higher education institutions, which have the VIRTA data. However, more research and data are required to make the student performance prediction more accurate and include more features as well. This additional data could be collected from various systems, for instance, student management, learning management, project management, and reporting systems.

Keywords/tags (subjects)

Educational Data Mining, Machine Learning, University of Applied Sciences, VIRTA data, CRISP-DM

Miscellaneous (Confidential information)

Contents

Figures

Tables

Code Blocks

1 Introduction

Machine learning (ML) is utilized constantly in various industries because its possibility to provide innovative solutions to various stakeholders of an organization. ML is utilized in higher education industry to provide insights and supporting activities of an educational institution. This scientific field is known as an educational data mining (EDM) and it is defined to be a field, which "*exploits statistical, machine-learning, and data-mining (DM) algorithms over the different types of educational data*" (Romero & Ventura, 2010). Thus, ML can be adapted in education industry in several ways and at the best, ML can be utilized to predict several matters related to students and provide valuable information concerning student performance during their study process from the very beginning.

The data is a key element in data analytics and in ML. The higher education industry organizations have commonly several systems, which they adapt in their activities including, for instance, student management, learning management, project management, reporting, and many other systems. These systems include the data, which can be utilized in the development of ML-based solutions. However, data quality can be an issue while developing ML models in various circumstances. The main objective of this study is to explore the ML in higher education industry. Moreover, the objective is to provide a practical example of ML-based project utilizing ML project management approach (CRISP-DM) to provide an answer to research questions. The case organization in this study is higher education institution, namely Seinäjoki University of Applied Sciences.

The research exists about ML utilization in higher education industries, but each research is conducting a different type of contribution, because of datasets and contexts. Thus, data is commonly an organization centric. This study provides a ML-based solution related to the VIRTA data and systems, which are commonly utilized in higher education industry organizations at Finland. The study has three research questions, which are answered based on practical implementation of ML-based system.

1. How CRISP-DM model can be adapted in the higher education industry?

2. Which unsupervised ML model performs best in student categorization?
3. Which supervised ML model performs best in predicting student performance?

The practical implementation is required to provide an answer to research questions. The CRISP-DM approach is adapted in practice and report the results of this study. Appropriate selection of data is crucial to provide the best possible predictions using ML models. The deployment of the ML-based solution to existing systems is part of this study. Thus, development of REST interface is implemented in this study, which can be utilized to provide predictions for existing systems in the case organization.

The rest of the study is constructed as follows. First, the relevant literature is presented in theoretical background in chapter 2. Second, methodological choices are highlighted in chapter 3. Third, the practical implementation is presented in results of the study in chapter 4. Finally, the research questions are answered, and study is concluded in chapter 5.

2 Theoretical Background

This chapter provides the review to state-of-the art literature related to ML and EDM in educational context. ML algorithms is an integral part of educational data mining and objective of this study is to explore the ML and its utilization in higher educational industry. Moreover, CRISP-DM approach is presented in theoretical level in this chapter.

2.1 Educational Data Mining

Educational data mining (EDM) or educational mining is the field of science, which explores, how educational data can be utilized to increase the performance of educational institutions (Vaidya & Saini, 2021). Moreover, objective of EDM is to find and predict hidden patterns and support managerial decisions in various domains from the huge amount of data (Singh & Pal, 2021). On the other hand, learning analytics (LA) is the field of science, which have been defined to be "*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*" (Siemens & Baker, 2012; Kollom et al., 2021).

EDM is highlighting several points of view to support the decision making. EDM as a theoretical aspect is combination of several computational fields of sciences, which are, for instance, computer science, statistics, ML, and data mining. EDM and its connection to different fields of sciences is presented in figure 1.

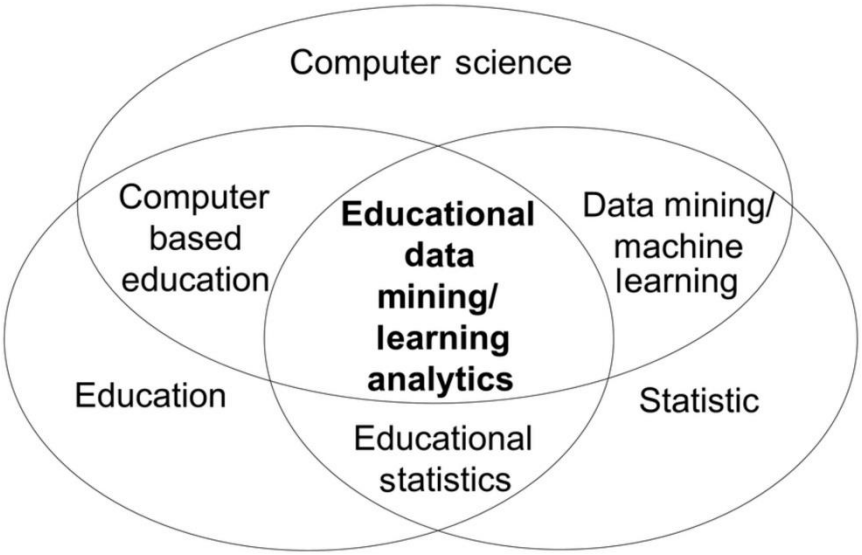


Figure 1. Educational data mining (Romero & Ventura, 2020)

EDM is a multidisciplinary field of science as figure 1 is illustrating. EDM highlights the possibilities to educational institutions to analyze the data with various tools and techniques to create better insights to an institution's activities. Data can be in several systems, which can be, for example, learning management, administrative and other systems. This data can be in several formats and can be, for example, structured, unstructured, semi-structured or binary data (Cauteruccio et al., 2020). Modern systems provide multiple possibilities to connect to different data sources using application programming interfaces (API) or provide simple processes to export the data in an appropriate format (e.g. CSV). This data can be analyzed in various ways with ML algorithms. Thus, educational data mining strives to utilize ML methodologies in educational context using the educational data.

2.2 Machine Learning Algorithm Development Process

ML and data modeling require a process and this process commonly follows the same pattern. Everything begins from a decent problem and ends to final version of the ML model, which can be adapted by different systems and software's. Figure 2 is illustrating a common ML algorithm development process.

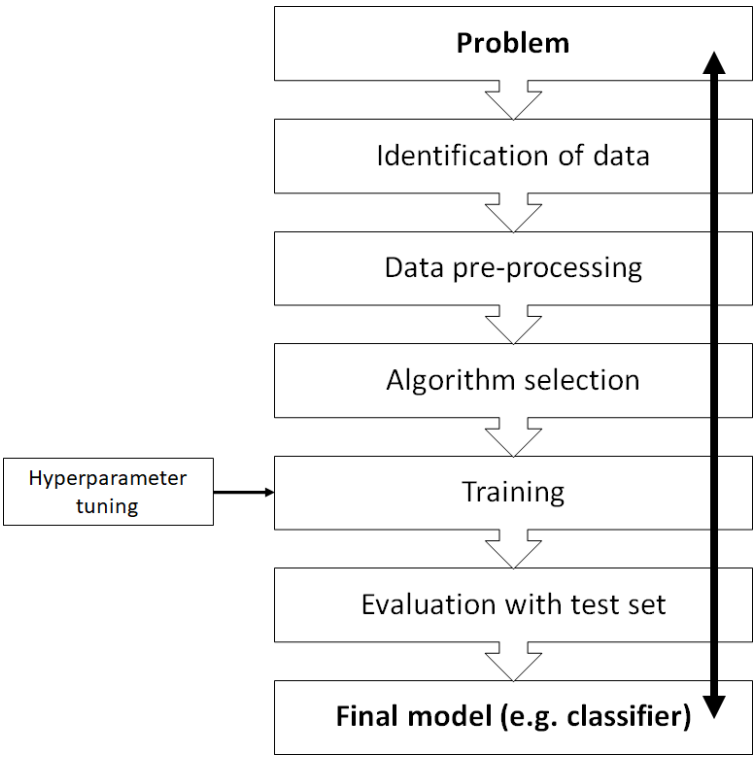


Figure 2. Machine learning algorithm development (Adapted from Ayodele, 2017)

After the problem has been identified and appropriately articulated, the next step is to identification of the data. The identification of the data requires an expert of the field, who understands what attributes are the most relevant to include in the ML model from the business point of view. The second option is to use "brute-force" if expert is not available. Here all available attributes are included but it is not the most effective method since it might contain noise and as well missing attributes and, moreover, might require a lot of pre-processing (Ayodele, 2017; Zhang, 2002).

The next step is the data pre-processing where the data will be processed in appropriate format for ML algorithms and new features are engineered. All irrelevant features should be removed, and dimensionality of the data should be decreased (Yu, 2004). Next the training set is defined. Quantity of the data in training set is defined by the field where ML is adapted but the rule of thumb is that 80%/70% of the data should be in the training set and 20%/30% should be in the test set (c.f. Akyol,

2017). Different training and test set distributions should be as well tested in ML modeling (Patgiri et al., 2019).

Final step is to select the appropriate ML algorithm and train the model. The ML developer should pay attention to the accuracy and precision of the model and evaluate how well the model performs prediction with different inputs. Here the parameters or hyperparameters can be tuned. Hyperparameters are the values which can be used to optimize the learning process (Li et al., 2018). At the end of the process the final model can be deployed in various systems and software's. The process is iterative, which means that developer moves between different phases in the process several times during the ML algorithm development.

2.3 Machine Learning Algorithms

The utilization of ML is getting easier and the programming is relatively simple with modern tools. Moreover, several ML cloud-based systems exist to enhance artificial intelligence development in various environments (e.g. Azure, AWS and Google Cloud). However, the utilization of the ML algorithms itself is relatively complex, because of several different algorithms, hyperparameters and different datasets. Figure 3 is illustrating categorization of different ML algorithms, which can be adapted in several ways in higher education industry.

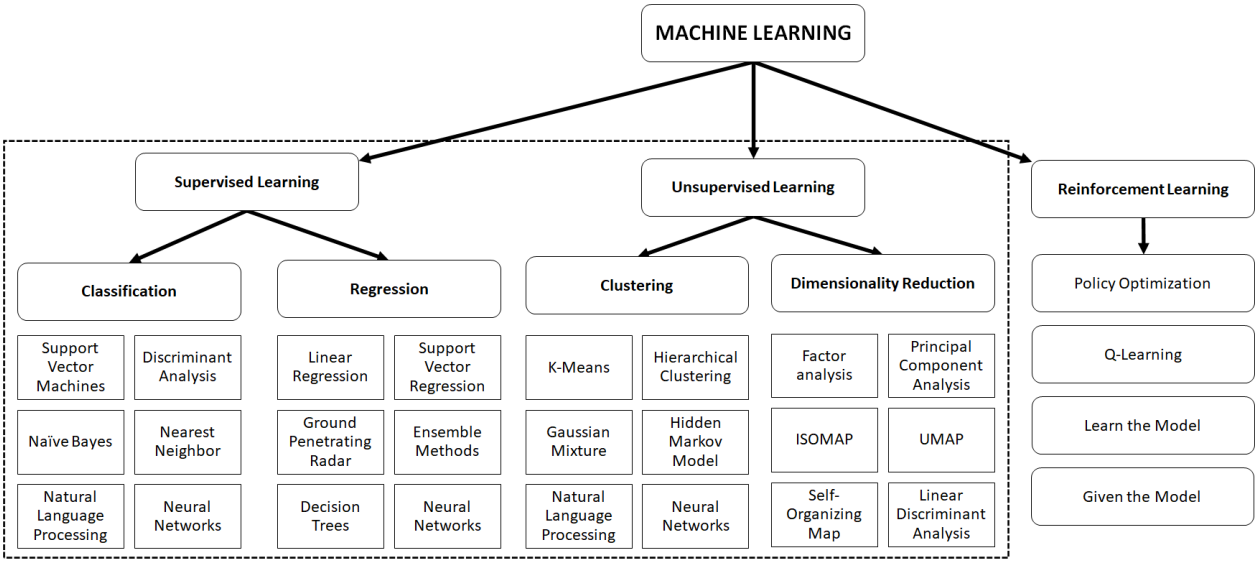


Figure 3. Summary of machine learning algorithms (Adapted from Durga & Rani, 2019)

Figure 3 is illustrating a common landscape of ML algorithms and, thus, is not holistic view of the entire landscape. Presented algorithms are examples of the presented ML category. Moreover, the ML categories, which are illustrated inside the dotted line are the most relevant at higher education industry. ML can be shared commonly to supervised, unsupervised and reinforcement learning, which can be adapted in an organization in several ways.

Supervised learning is based on mapping between a set of input variables and an output variable (Cunningham et al., 2008). This mapping is utilized to predict the output of the new data. Thus, in supervised learning the output is well known, and training of the model can be implemented using this data. Supervised learning algorithms are commonly distributed to classification and regression algorithms (Sethi & Mittal, 2019). Several ML models exist for this purpose. Some examples are support vector machine, naïve bayes and natural language processing.

Unsupervised learning is based on learning from the data, which have no specific output or labels (Ghahramani, 2003). Unsupervised learning is commonly utilized, for instance, to cluster the data

and in anomaly detection. Therefore, unsupervised ML algorithms can be utilized to locate the similarities from the data and expose hidden patterns. Unsupervised ML algorithms are generally distributed to clustering and dimensionality reduction algorithms (Dash et al., 1997). Several unsupervised learning algorithms exist and widely used examples are k-means, principal component analysis, and hierarchical clustering.

The ML algorithms, which are between supervised and unsupervised learning are called semi-supervised learning. Semi-supervised learning combines small amounts of labeled data samples to with large number of unlabeled data (Zhou & Belkin, 2014). Supervised and unsupervised algorithms can be utilized in semi-supervised ML purposes. This can be achieved, for instance, to combine clustering and classification algorithms.

Reinforcement learning is an area of ML where machine is taking suitable activities to maximize the reward (Sutton & Barto, 2018). The main goal of reinforcement learning is to make an agent to learn to give reward or penalty. The goal is to maximize the reward. Reinforcement learning is utilized, for example, by the robots in object picking and moving it to containers (Knight, 2016). Several algorithms exist for reinforcement learning, for instance, policy gradient, proximal policy optimization and soft actor critic.

2.4 Student Performance Prediction with Machine Learning

ML utilization to predict student performance has been implemented with various systems and datasets by researchers acting in different fields of science. The main objective is to find patterns from the data, which define when student performance is good enough to proceed with the studies and not to dropout while studying. Thakar et al. (2015) have implemented a literature review related to ML utilization in different educational industries, which results table 1 illustrate as appropriate.

Table 1. Machine learning in educational industries (Adapted from Thakar et al., 2015)

Methodology	Key Findings
Decision Tree	Success chances of curriculum estimation by implementing student profiling with storyboard system.
Classification and Clustering	The performance of the final year students.
K-means Clustering	Students' learning behavior analyzation to check the performance of students and predicted weak students.
Classification Tree Models	Enrolment attributes of pre-identify success of students.
Clustering, Association Rules and Decision Trees	Students' academic achievement, students drop out, and students' financial behavior.
Neural Network, Rough Set Theory	Dropout prediction of the course.
Decision Tree	Storyboard (e-learning system) success paths of students.
Decision Tree	High school student's evaluation and studying effectiveness
Decision Tree	Forecasting model for students' marks to identify negative learning habits or behaviors of students.
Association Rule	Student's achievement prediction systematically.
Rough Set Theory	Weak student prediction.
Association Rules, Apriori Algorithm	The success and failure factors of students.
Genetic Algorithm, Novel Spatial Mining Algorithm	Final grade prediction based on features extracted from log data in a web-based system.
J48 and farthest first algorithms	Managerial information on understanding, predicting, and preventing academic failure.
Statistical Approach, Association Rules,	Hidden patterns for students to avoid becoming low performer ones.
Grammar Trees, Regular Expressions	Concept Map Mining (CMM) done from essays written by students to judge their knowledge.
Decision Tree, Rough Set Theory, Naïve Bayes	The produced system generated rules to predict the final grade in a course under study.
Fuzzy Approaches	Fuzzy approaches to classify students' academic performance
Decision Tree	A roadmap for the application of data mining in higher education by pre-identifying weak students.
Genetic Algorithm	Final grade prediction of students based on features extracted from logged data in an education Web-based system.
Decision Tree, Naive Bayes Algorithm, NN	Student performance prediction before the final examination.
Decision Tree, Sota, Naïve Bayes	Comparison of three algorithms in terms of prediction of students result.
Cluster Analysis, NN, Logistic Regression and Decision Tree.	Three techniques comparison for understanding undergraduate's student Enrolment data.
Statistical Classifier, Decision Tree, Rule Induction, Fuzzy, NN	Classifier model for educational use in terms of accuracy and comprehensibility for decision making.
Co-relational Statistics, Multiple Regression	Relationships between the Participants' personality preferences and their employability attributes.
Bayesian Method and Decision Tree Method	Comparison of classification accuracy between two algorithms to evaluate employees' performance.
Decision Trees, NN, SVM, Logistic Regression	Student satisfaction determination for a particular course.
Classifiers, Random Tree, Random Forest	The faculty evaluation based on different parameters.
Decision Tree	Assisted in selecting students for enrollment in a course.
Decision Tree and Clustering	Learning disabilities prediction of school-age children.
Modified Apriori Algorithm	Evaluation index system and teaching index method based on data mining.
Association Rules	The patterns in matching organization and student interests, where they meet each other's requirements.
Data warehouse	Data warehouse to facilitate and provide a thorough analysis of department's data.
OLAP, Data warehouse	Curriculum's establishment analysis from several angles.
Bayesian Network, Decision Forest	Nbtree as the classifiers to predict student sequences for course registration planning
Conceptual	The data mining process in management education and focusing on academic aspects of admission and counseling process.
Decision Trees	Students' choice in continuing their education with post University studies (Master's degree or PhD)

Support Vector Machines and Rule-Based Approach	Predict the enrollment headcount by a predictive model built from new, continued and returned students.
Naive Bayes	The usage of Data Mining Techniques to support the evaluation of collaborative activities.

ML can be widely adapted in higher education industry as table 1 is presenting. The objective of ML utilization in higher education industry is to research, for example, how to minimize the dropouts in courses and study programmes (Thakar et al., 2015). This is challenging task since not all the data is available from the students even though a lot of data can be collected, analyzed, and modeled with different educational IT systems. Thus, understanding of dropouts and weak performance is a complex task since systems and datasets varies between different educational institutions. Therefore, developed ML models are always tailormade in educational institutions.

This study is utilizing CRISP-DM methdology as the main approach in development of ML-based system. Moreover, a survey was implemented for key stakeholders to provide an understanding of the requirements of the organization, which supports ML-based system development. The survey form can be found in appendix 1 and results are extracted mainly in business understanding section of the study.

3 Research Design

The objective of this chapter is to provide an understanding of general methodological choises of the study and an outlook to the case organization and the business domain where the case organization is acting on. This research can be concluded as a case study. Based on Yin (2009), a case study is "*an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident*". This study is a single case study, because a single organization selected and it is acting as well the unit of analysis in this study. The study is utilizing CRISP-DM methdology as the main approach of implementation. Moreover, a survey was implemented for key stakeholders to provide an understanding of the requirements of the organization. The survey form can be found in appendix 1.

3.1 The Case Organization

Seinäjoki University of Applied Sciences (SeAMK) is a mid-sized University of Applied Sciences located in Seinäjoki, Finland. SeAMK has about 5000 students (about 10% international students) in total, 800 graduating students annually, 250 international mobility students, 200 international degree students, 180 academic staff and 160 other staff in total (SeAMK, 2020a). SeAMK has four different faculties, which are Business and Culture, Technology, Health Care and Social Work and Food and Agriculture. Vision is "International, entrepreneurial SeAMK – Best for our students", mission is "The mission of SeAMK is to increase know-how, competitiveness and welfare" and values are (1) Entrepreneurial mindset, (2) Internationalisation, (3) Responsibility and (4) SeAMK spirit. SeAMK's organization's structure is illustrated in figure 4.

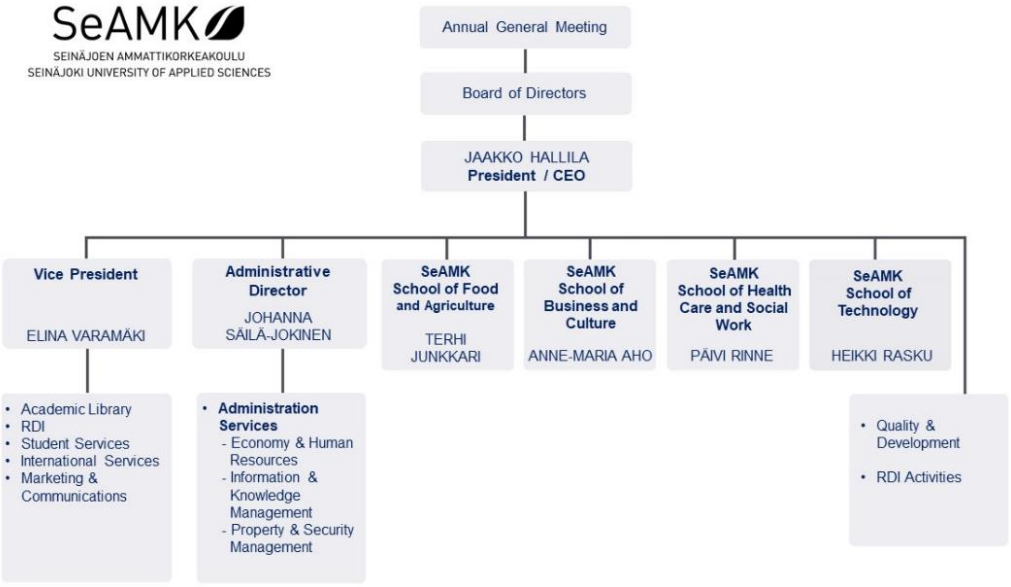


Figure 4. Case organization's chart (SeAMK, 2021a)

SeAMK is organized into different schools as figure 4 is illustrating. Schools provide different study programmes and implement research, development, and innovation activities. Study programmes are different (e.g. length and level of the programme), which have an affection to the data in different systems. SeAMK provides 20 different Bachelor and 13 Master's degree programs in total. Moreover, SeAMK provides double degree programmes in the field of technology and business. SeAMK has four profile areas, which are food safety, digital manufacturing and industrial internet, welfare technology and growth entrepreneurship and business transfers (SeAMK, 2021a).

3.2 CRISP-DM as an Approach

The implementation of this research was carried out with CRISP-DM approach (CRoss Industry Standard Process for Data Mining). Several other ML and data mining process models (e.g. KDD and SEMMA) exist but CRISP-DM is the mostly accepted model in several industries (Martínez-Plumed et al., 2019). The CRISP-DM model is a holistic process model, which supports carrying out data mining and ML projects (Wirth & Hipp, 2000; Jaggia et al., 2020). The model has six different iterative phases: (1) Business understanding, (2) Data Understanding, (3) Data preparation, (4) Modeling, (5) Evaluation and (6) Deployment (Wirth & Hipp, 2000). CRISP-DM model is illustrated in figure 5.

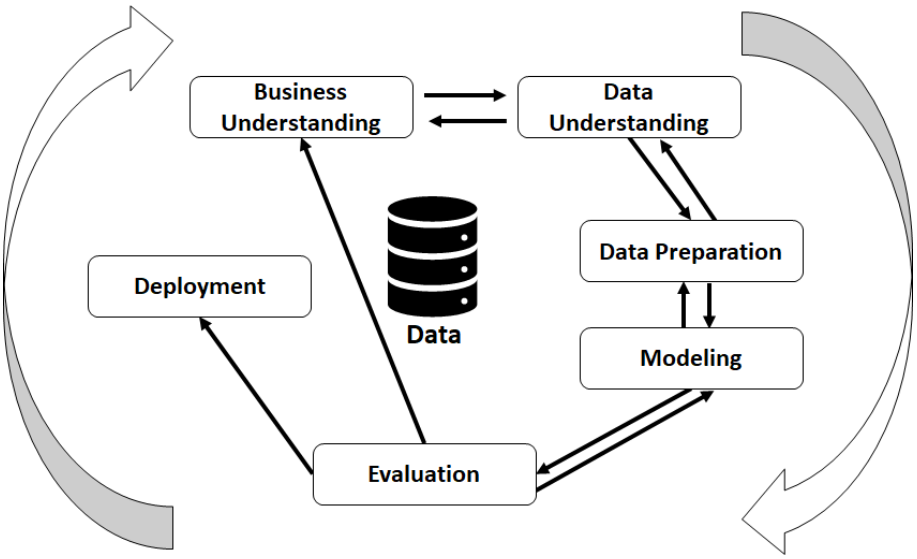


Figure 5. CRISP-DM model (Adapted from Wirth & Hipp, 2000)

The CRISP-DM model is based on six different iterative phases as figure 5 is illustrating. The process is highlighting the continuous and developing nature of data and ML. The solution is not finished even though the ML project or process might be ready. The solution requires constant renewal and iteration for updates. Moreover, data is not commonly static and, thus, requires constant monitoring for quality. Each phase has its own important role in the CRISP-DM process which table 2 is presenting.

Table 2. CRISP-DM model phases (Wirth & Hipp, 2000)

CRISP-DM model phase	Description of the phase
1. Business understanding	**Determine Business Objectives** • Background • Business objectives • Business success • Criteria **Assess Situation** • Inventory of resources • Requirements, assumptions, and constraints • Risks and contingencies • Terminology

	• Costs and benefits **Determine Data Mining Goals** • Data mining goals • Data mining success • Criteria **Produce Project Plan** • Project plan • Initial assessment of tools and techniques
2. Data understanding	**Collect Initial Data** • Initial data collection report **Describe Data** • Data description report **Explore Data** • Data exploration report **Verify Data Quality** • Data quality report
3. Data preparation	**Data Set** • Dataset description **Select Data** • Rationale for inclusion or exclusion **Clean Data** • Data cleaning report **Construct Data** • Derived attributes • Generated records **Integrate Data** • Merged data **Format Data** • Reformatted data
4. Modeling	**Select Modeling Technique** • Modeling technique • Modeling assumptions **Generate Test Design** • Test design **Build Model** • Parameter settings • Models • Model description

	Assess Model • Model assessment • Revised parameter settings
5. Evaluation	**Evaluate Results** • Assessment of data mining results (c.f. Business success criteria) • Approved models **Review Process** • Review of process **Determine Next Steps** • List of possible actions • Decision
6. Deployment	**Plan Deployment** • Deployment plan **Plan Monitoring and Maintenance** • Monitoring and maintenance plan **Produce Final Report** • Final report • Final presentation **Review Project** • Experience documentation

The ML development project or process begins from creating business understanding and ends to the solution deployment as table 2 is presenting. The CRISP-DM model phases are process based but the approach is iterative and can be adapted in several ways in ML and data mining projects. This study adapts CRISP-DM model in higher education industry as appropriate and as well report the results based on the model structure.

3.3 Data Sources and Primary Data Description

Several data sources and tools are utilized in higher education industry organizations. Each function of an organization has commonly its own data source, which is a challenge when developing different ML-based solutions. Thus, data aggregation is required to implement different analyses and ML models. However, some data sources already aggregate the data, which can be used in several ways

in ML projects (e.g. VIRTA data). These datasets are not complete, which requires computed features during pre-processing when developing the ML models. The different data sources are illustrated in figure 6.

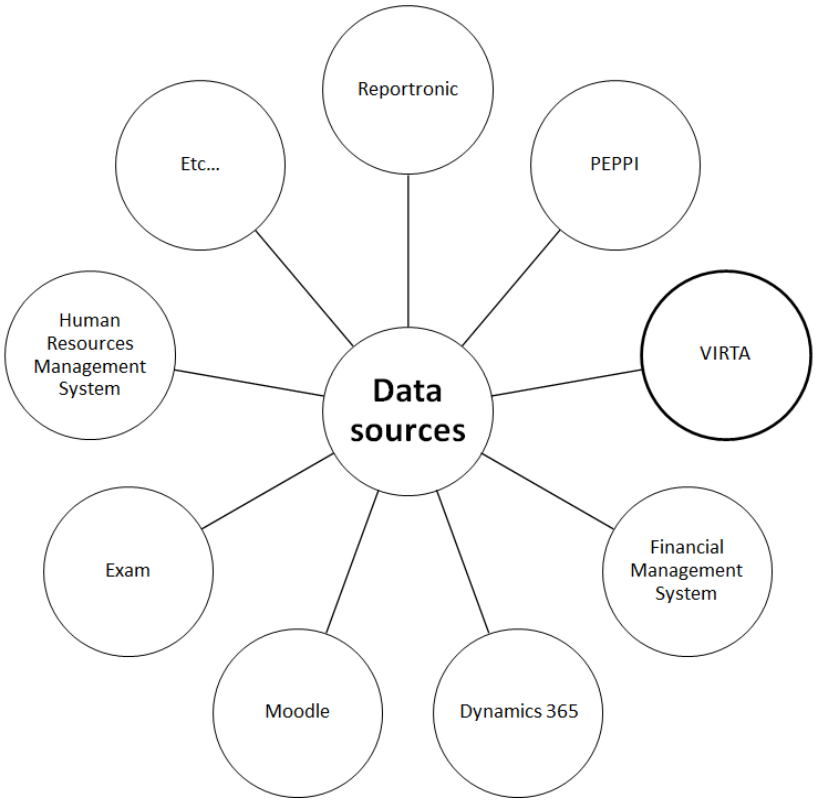


Figure 6. Data sources

Figure 6 illustrates several different data sources in the case organization. These data sources are relatively common at different universities of applied sciences organizations in Finland. The VIRTA data have an important role in the prediction modeling in ML in the case organization. However, VIRTA data is not complete, but VIRTA data was utilized in this study in several ways to create the ML models. VIRTA data was selected as the main data source of data in this study for ML algorithms since it includes enough relevant information to predict, for instance, student performance. Moreover, VIRTA data is relatively easy to aggregate with other datasets in the future if required.

VIRTA data is a dataset collection, which provides all relevant information related to the students. The data include, for instance, student background information (e.g. study programme and study periods), courses, term-specific enrolments, and other relevant information related to studies. VIRTA dataset is XML formatted data and it is sent to CSC, which aggregates all data from different universities and universities of applied sciences (CSC, 2021a). Various analyses and ML models can be implemented using the VIRTA data to support activities of educational institution.

3.4 Development Process

This study was carried out as a development process using CRISP-DM approach. The study process included all relevant steps and the process consist of two different main phases: pre-processing phase and ML development phase. The pre-processing phase is mainly for formatting the data for ML algorithms and ML development phase are mainly for modeling, evaluation, and deployment. Figure 7 is illustrating the pre-processing phase.

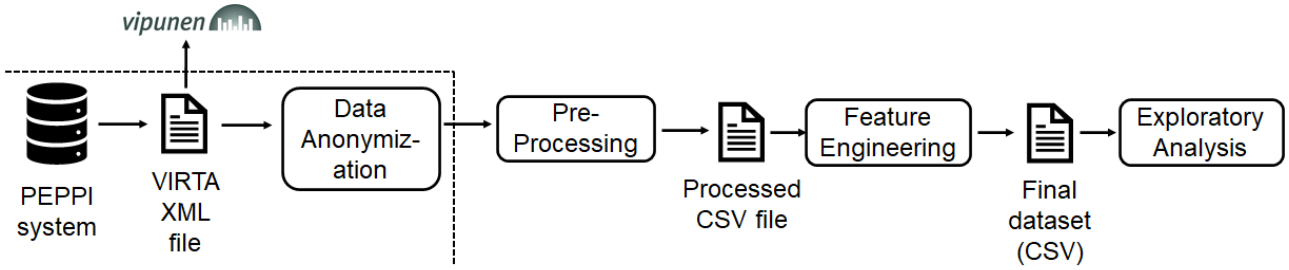


Figure 7. Pre-processing phase of the study (phase 1)

The VIRTA data is constructed from the data of PEPPI student administration system. The script, which constructs the VIRTA XML dataset is provided by PEPPI consortium and all PEPPI student administration system organizations are utilizing the same script since the data should be aggregable with other organizations XML files in CSC systems. This same file can be utilized in data analytics and ML. The data is anonymized before pre-processing phase and anonymization is implemented by IT department of the case organization.

VIRTA XML file requires pre-processing and feature engineering to provide the data in an appropriate format for ML algorithms. The pre-processing was implemented as its own Python algorithm and feature engineering as a separate algorithm. The exploratory data analysis was implemented after the final feature engineering calculations were completed. There are two main data saving points (CSV files) in the process as figure 7 is illustrating. The next step was to proceed ML development phase after the final dataset was created and exploratory analysis was implemented. Figure 8 is illustrating ML development phase.

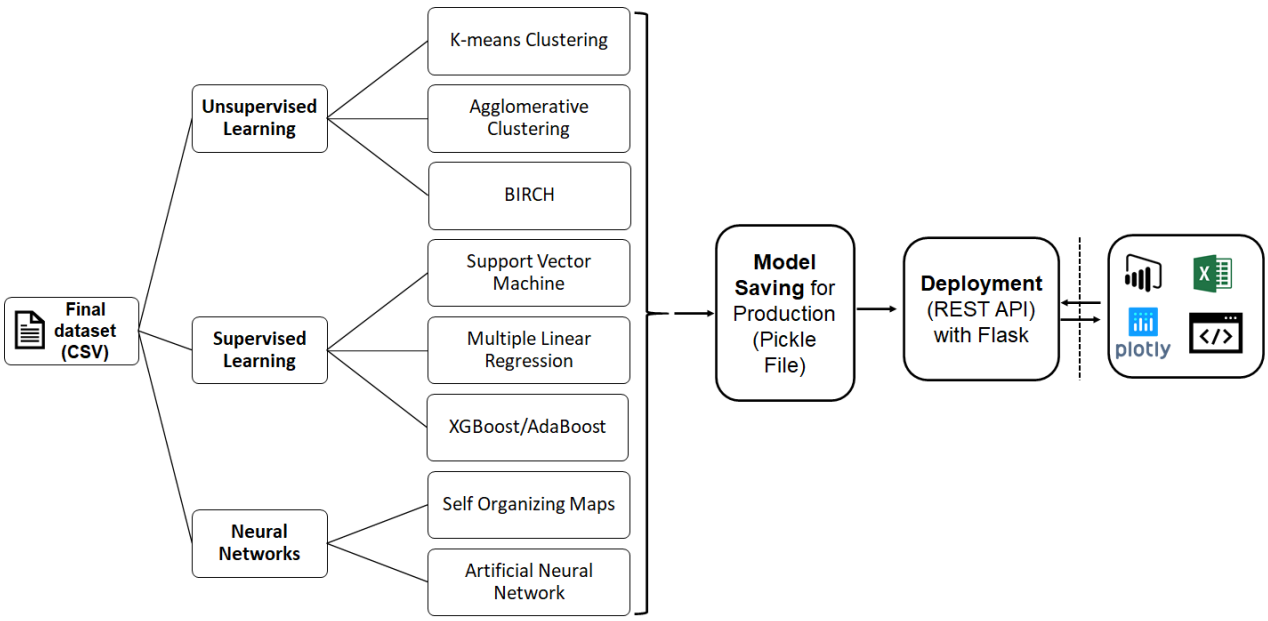


Figure 8. Machine learning development phase (phase 2)

The final dataset, which was created in phase 1, is utilized in ML development phase. It is possible to adapt three different ML approaches in this study: unsupervised learning, supervised learning, and neural networks. Selected algorithms in this study are based on existing EDM and ML literature. The main objective of unsupervised is to identify different types of students based on features with clustering. The main objective of supervised learning and neural networks is to predict students,

who will graduate on time (student performance). On the other hand, the same approach can be utilized identify students with lower performance.

Both are relevant ML problems in higher education industry. Suitable models are selected after the testing and evaluation. Finally, the models are saved to Pickle files, which can be read by the REST API, which will be utilized as a deployment solution for the ML models. Different systems and software's (e.g. Power BI and Excel) can utilize the models in practice with certain parameters using the REST API.

4 Results of the Study

The implementation of ML-based prediction system is following CRISP-DM approach and this chapter is formatted on the basis of CRISP-DM. Moreover, this chapter is providing insights from the results of the survey, which was sent to key stakeholders at SeAMK. The selected programming language in this study is Python (version 3.8.5) and its main data analytics and ML libraries (e.g. *sklearn* and *pandas*).

4.1 Business Understanding

This section provides an outlook to the business domain of the case organization and it include the results of the survey including other relevant information related to this study. Higher education industry in Finland is distributed to universities and university of applied sciences, which are both higher education institutions (HEIs). Universities focus on scientific research and university of applied sciences to more practical education, which is closer to working life in various industries. Currently, there are 13 university level institutions and 23 university of applied sciences institutions at Finland (Fulbright, 2021). This study is focusing on mainly needs of university of applied sciences.

Ministry of Education and Culture (OKM) is coordinating and funding the activities of universities of applied sciences and universities in Finland. Universities and universities of applied sciences have their own individual autonomy and freedom of science even though the coordination is implemented by the ministry. The objective of ML and data analytics in higher education industries is to support the activities of institutions to increase the performance and enhance the competitive edge of an organization. Therefore, the main objective is to support students (customers) success in their studies and at the best create personalized and adaptive education (Ciolacu et al., 2018). The ministry's funding model has an affection for the activities of university of applied sciences. The core funding model of university of applied sciences is presented in figure 9.

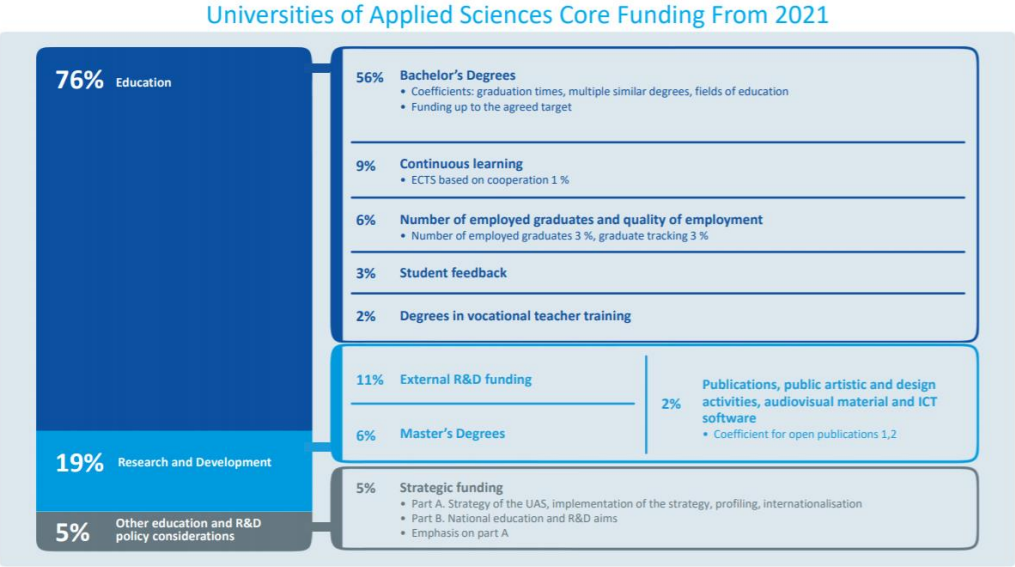


Figure 9. The core funding model (2021) of University of Applied Sciences (Minedu, 2021)

Thus, the core funding model is defining the activities of educational institutions in Finland. Several areas are measured, and funding is based on these areas as figure 9 is illustrating. In general, several possibilities and requirements arise for ML utilization in higher education industry in Finland. These are, for instance,

- Prediction of several matters related to students (e.g. graduation times in different fields of education) ➔ *Which students can be dropouts, and who are in-risk? Why are students dropping out from different programs?*
- Supporting employees in their daily activities (e.g. teachers, student counsellors and management) ➔ *Which students require more support? Which are the high-quality courses? What is the performance of teachers? What should be intake in a specific year that enough will graduate at the end in a specific study programme?*
- Student behavior analysis (e.g. feedback, grades, and activity) ➔ *How feedback and grades are correlating with different courses? What is the activity level in different learning areas? Which one of these areas predict the performance at best?*
- Real-time data visibility ➔ *Which data should be viewed in the real-time, for instance, in dashboards? How data is collected and viewed to different stakeholders?*
- Administrative support and prediction (e.g. accounting, budgeting, and human resources) ➔ *What will be the financial performance for the next five years? How to implement budgeting, posting, and routing as automatic as possible?*

This study is focusing on mainly to student performance prediction and, therefore, is not answering all of these questions. However, these questions are relevant and are connected directly to the

funding model of university of applied sciences and as well to universities in Finland. ML models can provide an answer to some of these questions.

4.1.1 Strengths and Weaknesses

The case organization has a good background of data and its processing in overall based on the results of the survey. Data systems are managed by IT department of the organization. This provides a good base structure for adapting the ML approaches. However, most of the systems are in their own silos and organizational units are not working in the same manner, which is relatively common in different organizations. This requires combining different data sources and as well processes which can gather data from the different systems where access is impossible from one reason or another (e.g. utilization of RPA).

Resourcing to ML and data-analytics development is required in the near future in the case organization. Resources are required to these activities now and even more in the future when the field of data analytics and ML is developing in the case organization's processes. Data gathering and preparation processes requires development, which are related to the customers (e.g. students) of the case organization (e.g. student and company feedback). This development should be mainly appointed to data preparation, pre-processing, feature engineering, and collecting of appropriate data.

4.1.2 Data Suppliers and Consumers

The case organization has several data suppliers, which are the organization's stakeholders. The data suppliers are employee groups from different functions (e.g. financial, international, information, marketing, and student services) and students based on the results of the survey. Moreover, for instance teachers and other personnel (e.g. RDI employees) provides data to different data

sources. External data suppliers are, for instance, different companies and partners (e.g. universities, ministry, Arene, and student unions), which are participating to activities in several ways. All data suppliers are important for data supplying and their needs should be highlighted during the development processes. All systems need to be easy to use, learn, and adapt, which have a direct impact on data quality.

Several consumer groups exist for data consuming. The management of the case organization has an interest towards of data and its analysis. The systematic utilization of data will be enhanced in managerial level in the case organization. The analysis reports and ML models would create value for stakeholder groups in different activities (e.g. new project proposal support). The amount of the data is not the problem on the case organization, but its utilization rate could be higher. ML algorithms can support activities of case organization in several ways.

Internal and external stakeholders consume the data on the case organization but different ways. The students provide inputs to different systems (e.g. Moodle and EXAM) as they utilize the systems during the studies. On the other hand, teachers and management provide data to the student administration system (e.g. grades and resource allocation). The consumers of the data are currently the same groups that are as well supplying data to the systems of SeAMK. The added value should be provided for all stakeholder groups in the future using ML.

4.1.3 Ethical and Data Privacy Aspects

Ethics, privacy, and data protection are important organizational matters and these topics should be covered in all organizations. European regulation (GDPR) supposes organizations to pay attention to it. Fundamentally, these are as well human rights (European Commission, 2018). Moreover, many researchers nowadays have paid attention on these questions because people are more aware of

their own human digity especially in their own personal identity and data (Fabiano, 2019). These matters should be as well tightly integrated part of the ML implementation in an organization. At least the following things need to consider (e.g. Aiken and Epstein, 2000):

- Anonymity
- Human rights
- Results interpretation
- Risk management
- Data utilization and authorization

The ML and its implementation on the case organization include several aspects related to ethics, privacy, and data protection, which need to be considered. The data, which will be utilized in the analysis and new applications are sensitive. The case organization has data privacy policy defined from several aspects, which have positive impact to risk management of the case organization (SeAMK, 2021b). The data privacy policy of the case organization is discussing how the data are utilized at the organization from several point of views. However, the utilization of ML might require revising the privacy policy.

The data, which will be utilized by the ML models, are related to the students of the case organization. This data is anonymized, and no single person can be identified from the dataset. Moreover, all data and its transfer are secure and is implemented only inside the network of the case organization. This way the risk of data capturing is minimized. Moreover, case organization's privacy policy statement provides excellent tools for different person groups to decline utilization of the data in case organization's systems. This is the fundamental right of a person, which are part of the case organization's student and teaching communities.

Appropriate interpretation of results is important. Even though different ML-based applications can be implemented with ML algorithms and the data, but at the end, the responsibility of result utilization and its interpretation is in the hands of the end user. This means, for instance, if teacher can

identify a dropout student using the ML-based application, it does not necessarily mean that a student is a dropout. This is an important aspect when ML is adapted in the case organization. Teachers should be aware of the "real-life situation" even though ML models predict something else. This requires appropriate training of end users before ML systems are launched within the case organization.

Data utilization and authorization are an important part of the data privacy. Data should be utilized only authorized persons. This means that persons, who do not have access or privileges to see the data, should not be able to access it. The dashboards, applications, and ML models should be covered, and roles should be defined on the case organization. Without roles, it is impossible to share the users and their privileges in different categories.

4.2 Data Understanding

The next step is data understanding after business understanding has been gained (Shafique & Qaiser, 2014). This study is utilizing VIRTA XML-based dataset. This data is extracted originally from student administration system PEPPI. VIRTA data and its structure is well documented by CSC. The documentation was utilized in feature engineering and development of the final dataset (CSV). The VIRTA data is anonymized and, thus, no single person cannot be identified. Table 3 is presenting the selected data features including the calculated features from VIRTA data.

Table 3. Dataset structure including calculated features

Field	**Data type**	**Description**
studentID	int64	Student ID. *Calculated feature.*
KirjoihintuloPvm	datetime64	Date when a student has entered the studies.
AlkuPvm	datetime64	Date when a student has begun the studies.

LoppuPvm	datetime64	Date when a student has ended the studies.
PvmTotal	int64	Total mount of dates in the studies. *Calculated feature.*
Tyyppi	int64	Type of the studies.
Koulutusala	int64	Field of study.
Koulutusmoduulitunniste	object	Name of the study programme.
TilaKoodi	int64	Status code.
Koulutuskoodi	float64	Code of the study programme.
Koulutuskieli	object	Language of the study programme.
Luokittelu	float64	Classification.
Valmistunut	int64	Graduated. *Calculated feature.*
Opintosuoritukset	object / pandas dataframe	All courses including the grades.
LukukausiCount	float64	Number of semesters.
semester-ECTS	float64	Semester average ECTS. *Calculated feature.*
semester-GPA	float64	Semester average GPA. *Calculated feature.*
total-ECTS	float64	ECTS in total. *Calculated feature.*
total-GPA	float64	GPA in total. *Calculated feature.*
Opt_ECTSCount	float64	Optimal amount of ECTS. *Calculated feature.*
Opt_SemesterCount	float64	Optimal number of semesters. *Calculated feature.*
studentGOT	int64	Student graduated on time. *Calculated feature.*

Table 3 is presenting data and features, which are utilized by the ML algorithms. The data does not include any data, which would bias the ML model (e.g. gender and place of residence). Moreover, not all the data is utilized by ML algorithms. Only the most valuable data is selected for final implementations of ML models. More information related to the VIRTA data is available directly from several CSC services (CSC, 2020; CSC, 2021b; CSC, 2021c).

Understanding the data quality is an important aspect in data understanding phase. Data can be considered high quality when it fits to its intended uses in an organization's activities (e.g. operations and decision making) (Fadahunsi et al., 2019). Moreover, data quality has several dimensions, which are, for instance, accuracy, reliability, timeliness, relevance, completeness, currency, consistency, flexibility, precision, format, content, usefulness, clarity, scope, understandability, and freedom from bias (Wand & Wang, 1996).

VIRTA data is clear and understandable in overall from several point of views, but it requires pre-processing and preparation before ML can be adapted in the model development. VIRTA data is well documented, which increases the data understandability and as well usefulness in various scenarios. VIRTA data is in XML format, which is relatively heavy data format and, thus, should be processed to the format, which increases usability of the data in different ML scenarios. VIRTA data can be concluded to be reliable since data is collected from student administration system, which is the main source for several student data (e.g. records). VIRTA data will be free from bias since the several features are removed in pre-processing phase, which could create bias for ML models (e.g. gender).

4.3 Data Preparation

Data is not normally ready for ML modeling and pre-processing is important because of this matter (Kotsiantis et al. 2006). There is no "out-of-the-box" solution available. This is as well the case of the VIRTA data. The VIRTA data is XML based data and it requires pre-processing and feature engineering. First, the data pre-processing was implemented. Second, feature engineering was implemented based on pre-processed data.

4.3.1 Data Pre-Processing

The pre-processing of the data was implemented with Python utilizing *xml*, *numpy*, and *pandas* libraries. Each XML node was processed individually and saved to pandas dataframe. All student grades will be saved as well to pandas dataframe and nested to the master dataframe. Some of the features are calculated already in pre-processing phase since it is more practical in this phase. Required Python libraries in this phase are *pandas*, *numpy*, *xml*, and *datetime*. Code block 1 is illustrating the basic structure of the pre-processing script.

```
# FIRST, extract data from the VIRTA dataset
xtree = et.parse("virta-out.xml")
xroot = xtree.getroot()

df_studentCols = ["studentID", "KirjoihintuloPvm", "AlkuPvm", "LoppuPvm",
"PvmTotal", "Tyyppi", "Koulutusala", "Koulutusmoduulitunniste", "TilaKoodi",
"Koulutuskoodi", "LukukausiCount", "Koulutuskieli", "Luokittelu", "total-GPA",
"Valmistunut", "Opintosuoritukset"]
allRows = []
s_ID = 1000 # Unique ID (beginning of 1001)

# Iterate all nodes
for node in xroot.iter("{urn:mace:funet.fi:virta/2015/09/01}Opiskelija"):
    s_ID = s_ID + 1
    s_kirjoihintulopvm =
node.find("{urn:mace:funet.fi:virta/2015/09/01}KirjoihintuloPvm")

    if s_kirjoihintulopvm != None:
        s_kirjoihintulopvm =
node.find("{urn:mace:funet.fi:virta/2015/09/01}KirjoihintuloPvm").text if node
is not None else None
    else:
        s_kirjoihintulopvm = np.NaN

    s_LukukausiCount = None
    s_LukukausiCount =
node.find("{urn:mace:funet.fi:virta/2015/09/01}LukukausiIlmoittautumiset")

    if s_LukukausiCount != None:
        s_LukukausiCount = len(node.find("{urn:mace:fu-
net.fi:virta/2015/09/01}LukukausiIlmoittautumiset"))
    else:
        s_LukukausiCount = np.NaN

    grade_rows = [] # All data
    s_WMeanCalc = 0
    s_WMeanEctsCalc = 0
    s_Valmistunut = 0

    for opintosuoritukset in node.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Opintosuoritukset"):
```

```
        for opintosuoritus in opintosuoritukset.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Opintosuoritus"):
            s_SuoritusPvm = opintosuoritus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}SuoritusPvm")
            s_Nimi = opintosuoritus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Nimi")
            s_Kieli = opintosuoritus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Kieli")

            if s_SuoritusPvm != None:
                s_SuoritusPvm = opintosuori-
tus.find("{urn:mace:funet.fi:virta/2015/09/01}SuoritusPvm").text if node is
not None else None
            else:
                s_SuoritusPvm = 'nan' # For string formatting (csv)

            for laajuus in opintosuori-
tus.iter("{urn:mace:funet.fi:virta/2015/09/01}Laajuus"):
                s_Opintopiste =
laajuus.find("{urn:mace:funet.fi:virta/2015/09/01}Opintopiste")

                if s_Opintopiste != None:
                    s_Opintopiste =
laajuus.find("{urn:mace:funet.fi:virta/2015/09/01}Opintopiste").text if node
is not None else None
                else:
                    s_Opintopiste = 'nan' # For string formatting (csv)

            for arvosana in opintosuoritus.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Arvosana"):
                s_Arvosana = arvosana.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Viisiportainen")

                if s_Arvosana != None:
                    s_Arvosana = arvo-
sana.find("{urn:mace:funet.fi:virta/2015/09/01}Viisiportainen").text if node
is not None else None

                    if s_Arvosana.isnumeric():
                        s_WMeanCalc = s_WMeanCalc + (float(s_Arvo-
sana)*float(s_Opintopiste))
                        s_WMeanEctsCalc = s_WMeanEctsCalc + float(s_Opin-
topiste)

                else:
                    s_Arvosana = 'nan' # For string formatting (csv)
                    s_Valmistunut = 1


            # Finally add row to the list (only one row)
            grade_rows.append({"SuoritusPvm": s_SuoritusPvm, "Opintopiste":
s_Opintopiste,
                               "Arvosana": s_Arvosana, "Nimi": s_Nimi,
                               "Kieli": s_Kieli, "Ohjausala": s_Ohjausala})

    # Finally, calculate weighted arithmetic mean and append
    if s_WMeanCalc != 0:
        s_GPA = s_WMeanCalc/s_WMeanEctsCalc
```

```
    else:
        s_GPA = np.NaN

    allRows.append({"studentID": s_ID, "KirjoihintuloPvm": s_kirjoihintulopvm,
"AlkuPvm": s_AlkuPvm,
                    "LoppuPvm": s_LoppuPvm, "PvmTotal": s_PvmTotal, "Tyyppi":
s_Tyyppi,
                    "Koulutusala": s_Koulutusala, "Koulutusmoduulitunniste":
s_koulutusmoduulitunniste,
                    "TilaKoodi": s_TilaKoodi, "Koulutuskoodi": s_Koulutuskoodi,
"LukukausiCount": s_LukukausiCount,
                    "Koulutuskieli": s_Koulutuskieli, "Luokittelu": s_Luokittelu,
                    "total-GPA": s_GPA, "Valmistunut": s_Valmistunut, "Opintosuo-
ritukset": grade_rows})

# THIRD, save data to CSV
df_students.to_csv('students.csv', encoding='utf-8', index=False)
```

Code Block 1. Basic structure of the pre-processing script

Code block 1 presents the pre-processing solution party. Full Python script (Preprocessing.py) is available in appendix 1. Pre-processing begins with parsing the XML file with *xml* library. Each XML node is processed separately. The tag "*{urn:mace:funet.fi:virta/2015/09/01}*" is presenting the target namespace, which is required while reading XML-based data in Python. Each value is read to different variables, which are combined at the end to the final dataframe. The following features are calculated in the pre-processing phase

- Student ID
- Weighted arithmetic mean based on the courses
- Amount of studying days in total
- Student graduation.

The pre-processed data is exported to CSV format after the final dataframe is created. This makes easier to read the data directly to pandas dataframe, for example, in data modeling phase. The VIRTA data is versatile and can be adapted several ways in modeling phase. The VIRTA data is not complete or enough for ML modeling. Several features need to be calculated based on the data, which is implemented mainly in feature engineering phase.

4.3.2 Feature Engineering

A feature is an individual variable of an individual object (Dong & Liu, 2018). This can be, for instance, height or weight of a person. Thus, feature engineering is the development of different features based on the different variables of the data. This study adapts the VIRTA data in feature engineering and part of the features are developed for practical reasons (e.g. processing time reduction) in pre-processing phase. Required Python libraries in this phase are *pandas*, *numpy*, *ast*, and *sklearn*. Code block 2 is presenting the basic structure of the feature engineering script.

```
# Remove degree from graduated students
df_Opintosuoritukset = df_Opintosuoritukset[df_Opintosuoritukset['Arvosana']
!= 'nan']

# Set the dtypes
df_Opintosuoritukset['studentID'] = pd.to_numeric(df_Opintosuoritukset.studen-
tID, errors='coerce')
df_Opintosuoritukset['SuoritusPvm'] = pd.to_datetime(df_Opintosuoritukset.Suo-
ritusPvm, errors='coerce')
df_Opintosuoritukset['Opintopiste'] = pd.to_numeric(df_Opintosuoritukset.Opin-
topiste, errors='coerce')
df_Opintosuoritukset['Arvosana'] = pd.to_numeric(df_Opintosuoritukset.Arvo-
sana, errors='coerce')
df_Opintosuoritukset['Nimi'] = df_Opintosuoritukset['Nimi'].astype(str)
df_Opintosuoritukset['Kieli'] = df_Opintosuoritukset['Kieli'].astype(str)
df_Opintosuoritukset['Ohjausala'] = pd.to_numeric(df_Opintosuoritukset.Ohjaus-
ala, errors='coerce')

# Impute missing grades (HYV grades with mean)
imputerG = SimpleImputer(missing_values=np.NaN, strategy='mean')
df_Opintosuoritukset.Arvosana = imputerG.fit_transform(df_Opintosuorituk-
set['Arvosana'].values.reshape(-1,1))[:,0]

# Reorder the dataframe and remove unnecessary columns
df_Opintosuoritukset = df_Opintosuoritukset.reset_index()
df_Opintosuoritukset = df_Opintosuoritukset.drop(columns=['index', 'Nimi',
'Kieli', 'Ohjausala'])

# Calculate total amount of ECTS per student
df_OpintosTotal = df_Opintosuoritukset.groupby(['studentID'])['Opin-
topiste'].sum().reset_index()
df_OpintosTotal.rename(columns={'Opintopiste': 'total-ECTS'}, inplace=True)

# Sum the number of grades per a semester, per a student and create new pandas
dataframe
df_OpintosPerSemester = df_Opintosuoritukset.set_index('Suori-
tusPvm').groupby('studentID')["Opintopiste"].resample("2QS-JUN").sum()
df_OpintosPerSemester = pd.DataFrame(df_OpintosPerSemester).reset_index()
df_OpintosPerSemester = df_OpintosPerSemester.drop(columns=['SuoritusPvm'])
```

```
df_OpintosPerSemester = df_OpintosPerSemester[df_OpintosPerSemester['Opin-
topiste'] != 0]
df_OpintosPerSemester = df_OpintosPerSemester.groupby(['studentID'])['Opin-
topiste'].mean().reset_index()

# Calculate GPA per semester, per a student and create new pandas dataframe
df_GPAPerSemester = df_Opintosuoritukset.set_index('Suori-
tusPvm').groupby('studentID')["Arvosana"].resample("2QS-JUN").mean()
df_GPAPerSemester = pd.DataFrame(df_GPAPerSemester).reset_index()
df_GPAPerSemester = df_GPAPerSemester.drop(columns=['SuoritusPvm'])
df_GPAPerSemester = df_GPAPerSemester[df_GPAPerSemester['Arvosana'] != 0]
df_GPAPerSemester = df_GPAPerSemester.groupby(['studentID'])['Arvo-
sana'].mean().reset_index()

# Aggregate the datasets (dataset + df_OpintosPerSemester + df_GPAPerSemester)
dataset = dataset.join(df_OpintosPerSemester.set_index('studentID')[['Opin-
topiste']], on='studentID')
dataset = dataset.join(df_GPAPerSemester.set_index('studentID')[['Arvosana']],
on='studentID')
dataset = dataset.join(df_OpintosTotal.set_index('studentID')[['total-ECTS']],
on='studentID')
dataset.rename(columns={'Arvosana': 'semester-GPA', 'Opintopiste': 'semester-
ECTS'}, inplace=True)

# A binary feature for a student, which defines a student performance (Gradu-
ate-On-Time, GOT)
OptLukukaudet = pd.read_csv('OptLukukaudet.csv', sep=';')

dataset['Opt_ECTSCount'] = np.NaN
dataset['Opt_SemesterCount'] = np.NaN

# Find optimal amount of ECTS and semesters and aggregate
dataset.set_index('Koulutusmoduulitunniste', inplace=True)
dataset.update(OptLukukaudet.set_index('Koulutusmoduulitunniste'))
dataset = dataset.reset_index()
dataset = dataset.reset_index()
dataset = dataset.drop(columns=['index'])

# Set the dataset to appropriate order (features last)
dataset = dataset[['studentID', 'KirjoihintuloPvm', 'AlkuPvm', 'LoppuPvm',
'PvmTotal', 'Tyyppi', 'Koulutusala', 'Koulutusmoduulitunniste', 'TilaKoodi',
'Koulutuskoodi', 'Koulutuskieli', 'Luokittelu', 'Valmistunut', 'Opintosuori-
tukset', 'LukukausiCount', 'semester-ECTS', 'semester-GPA', 'total-ECTS', 'to-
tal-GPA', 'Opt_ECTSCount', 'Opt_SemesterCount']]

# Set the binary variable (student has graduated AND number of semesters is
smaller than optimal (target duration of the degree))
dataset['studentGOT'] = np.where((dataset['Valmistunut'] == 1) & (dataset['Lu-
kukausiCount'] <= dataset['Opt_SemesterCount']), 1, 0)
```

Code Block 2. Basic structure of the feature engineering script

Code block 2 presents the feature engineering solution party and full solution is available in appendix 1 (FeatureEngineering.py). Dataset includes as well graduated students, which can be noticed

from code block 2. Graduation is marked as one course, which is removed in feature engineering phase. Missing grades (courses without grading, accepted/failed) of the students are imputed with mean values. Following features are calculated during the feature engineering phase

- Total amount of ECTS
- Mean of ECTS per a semester
- Mean GPA per a semester
- Optimal number of semesters in a degree
- Optimal number of ECTS in a degree
- Student on-time graduation.

Optimal number of ECTS's and semesters are read from separate file, which includes ECTS and semester per a degree. This data is combined with a student data, which is processed by the ML algorithms at the end.

4.4 Modeling and Evaluation

The modeling phase is important phase in CRISP-DM process since the ML models are selected, developed, and tested in this phase (Schröer et al., 2021). Exploratory analysis is as well implemented in this phase in this study, even though normally it is carried out in data understanding phase. Moreover, the dataset requires new features and, thus, it is more practical to implement exploratory analysis in this phase. Evaluation of models and solutions are as well implemented during this phase.

4.4.1 Exploratory Data Analysis

The exploratory data analysis (EDA) provides a lot of useful information about the data. EDA is an important step in data science and ML since it provides a good way to explore unfamiliar datasets with several analysis techniques (Milo & Somech, 2020). The dataset is analyzed in this study from several point of views, which are, for instance,

- Overall description of the dataset

- Descriptive statistics
- Missing values
- Correlations between different features.

The full solution of the EDA analysis is available in appendix 3 (EDA.py). The pre-processed dataset includes 36704 rows and 16 columns in total (includes one categorial variable). Each row presents data of a one student and includes all course data as well. Moreover, descriptive statistics includes several calculated features, which provides more detailed information about the data. Descriptive statistics of the dataset is illustrated in table 4.

Table 4. Descriptive statistics of the dataset

	PvmTotal	Typpi	Koulutusala	TilaKoodi	Koulutuskoodi	Luokittelu	Valmistunut	LukukausiCount	semester-ECTS	semester-GPA	total-ECTS	total-GPA	Opt_ECTSCount	Opt_Semester-Count	student-GOT	Koulutuskieli_cat
count	36703	3670 3	36703	367 03	25682	8220	36703	27267	32942	32942	3294 2	3009 7	26396	26396	36703	36703
mean	*	3,79	7,64	3,01	654571 ,21	2,71	0,44	*	*	*	*	*	214,86	7,23	*	1,87
std	*	4,35	2,85	1,00	25848, 92	0,92	0,50	*	*	*	*	*	34,39	0,88	*	0,34
min	*	1	2	1	621101	2	0	*	*	*	*	*	60	4	*	0
25 %	*	1	5	3	631101	2	0	*	*	*	*	*	210	7	*	2
50 %	*	1	8	3	651409	3	0	*	*	*	*	*	210	7	*	2
75 %	*	8	11	3	671101	3	1	*	*	*	*	*	240	8	*	2
max	*	19	12	5	781103	7	1	*	*	*	*	*	240	8	*	2

Table 4 does include all relevant features of the dataset and analysis of values was implemented carefully per a feature. Moreover, part of data is censored (*) for data privacy reasons. Some values are relatively low because the dataset includes both graduated and undergraduate students. This should be noted in ML-algorithm development phase. Dataset includes some missing values in-total. These are in the different features as follows

- Koulutuskoodi ➔ 11021 in total
- Luokittelu ➔ 28483 in total
- LukukausiCount ➔ 9436 in total
- Semester-ECTS ➔ 3761 in total
- Semester-GPA ➔ 3761 in total
- Total-ECTS ➔ 3761 in total
- Total-GPA ➔ 6606 in total

- Opt_ECTSCount ➔ 10307 in total
- Opt_SemesterCount ➔ 10307 in total.

Missing values are imputed with median for further analysis. Data imputation is a process where missing values are replaced with some probable values and it is implemented because missing data commonly creates problems in analysis and ML modeling (Qin et al., 2007). Next step is to visualize the data in different ways. Visualizations were implemented with *seaborn* and *matplotlib* libraries. First, the dataset is visualized in overall with histograms as appropriate, which is presented in figure 10.

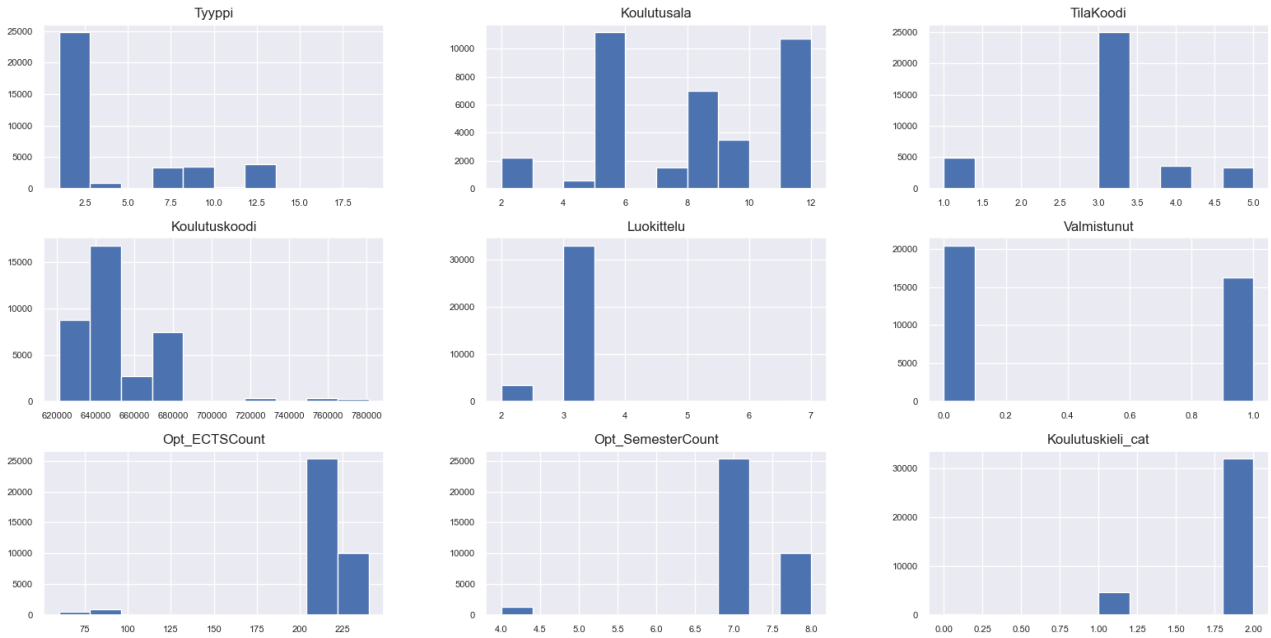


Figure 10. Histograms of features

Figure 10 is presenting most of the features of the dataset. Dataset includes both graduated and undergraduate students, which should be noted in development of ML algorithms. Next, the correlations between the features are analyzed with correlation heatmap, which is illustrated in figure 11.

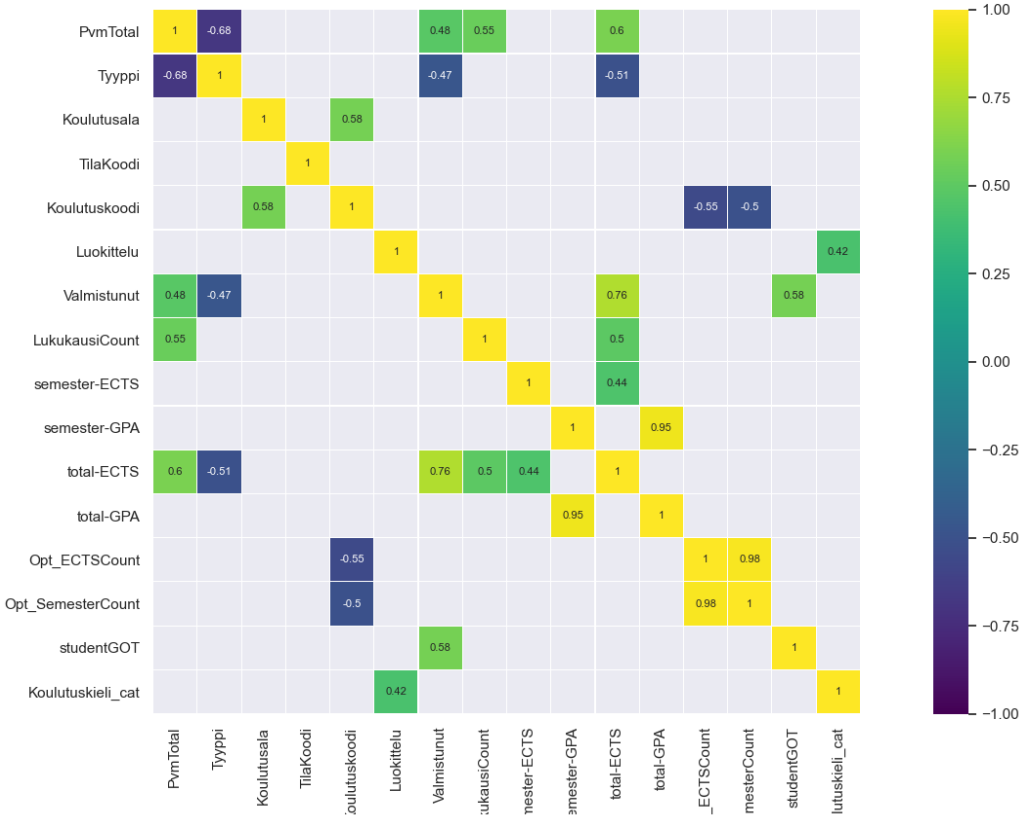


Figure 11. Correlation heatmap

Correlation heatmap provides an outlook of the relationship of the different variables. Positive correlations are marked with warmer color and negative correlations with colder color. Moreover, the figure 11 does not include the weak correlations. Correlation should be more than 0,4 or less than -0,4 to be visible in the correlation heatmap (c.f. appendix 3). Moreover, some additional analyses were implemented during EDA process, which are for instance, ANOVA and density plots. Python scripts of these analyses are available in appendix 3.

4.4.2 Unsupervised Machine Learning Models

Unsupervised ML models provides a way to find patterns from untagged data. Unsupervised learning models are utilized in this study in clustering purposes. K-means and hierarchical clustering (agglomerative) was selected for testing and evaluation in this study. Several Python libraries are utilized to implement unsupervised ML models (e.g. *sklearn* and *scipy*). Moreover, selected model is saved to the Pickle file, which can be utilized during ML model deployment. Python scripts of unsupervised ML are available in appendix 4.

K-means clustering is implemented to create an understanding of different clusters in the data. The k-means clustering receives dataset with all variables and optimizes over the position of the means using, for instance, Euclidean distance (Looney et al., 2021). The dataset is at first read with *pandas* library and after that unnecessary columns are dropped and missing values are imputed with mean. The outliers are removed after data imputation using z score (3 sigma's). After that data is normalized using min-max scaler. Code block 3 is presenting outlier removal and normalization process.

```
# Remove outliers using z score (3 sigma's)
z_scores = stats.zscore(f_dataset)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
f_dataset = f_dataset[filtered_entries]

# Data normalization
np_x = f_dataset.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(np_x)
normdata = pd.DataFrame(x_scaled)
normdata.columns = f_dataset.columns
f_dataset = normdata
del normdata
```

Code Block 3. Outlier removal and data normalization script

Next step is to find an optimal number of clusters. This is implemented with Elbow method and Silhouette score. The figure 12 is presenting an optimal number of clusters for undergraduate and graduated students.

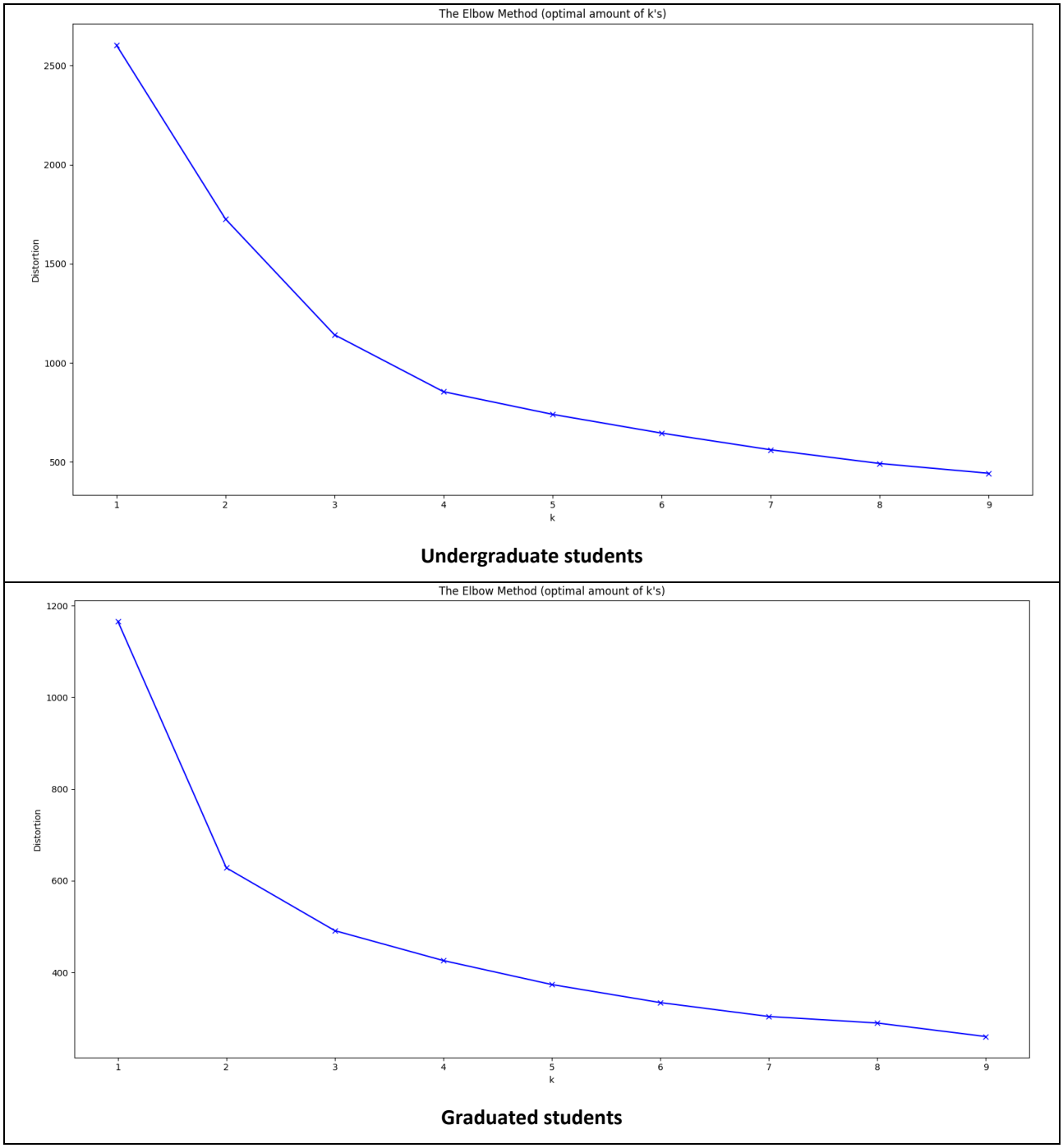
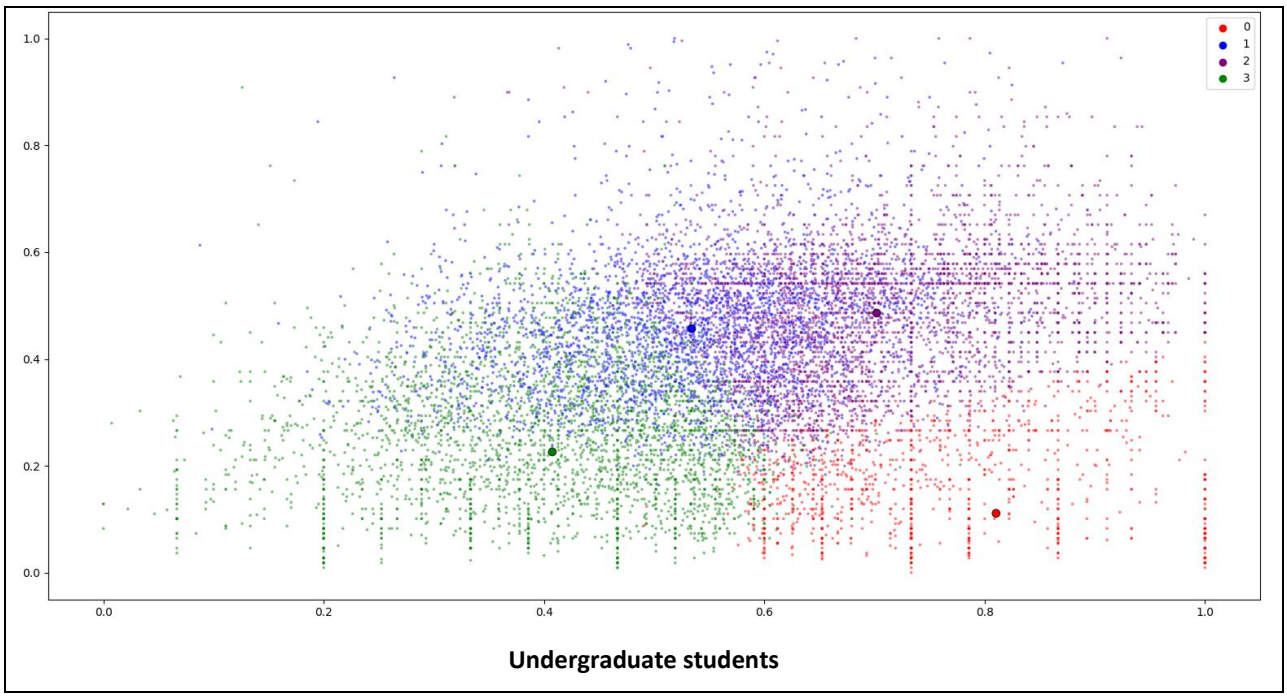


Figure 12. Elbow method

Silhouette score is another way to find an optimal number of clusters and it represents a value of the distance between the clusters (Ogbuabor & Ugwoke, 2018). Silhouette scores (number of clusters) for undergraduate students are (2) 0.3066043438173687, (3) 0.3603215411892813, (4) *0.37230502862201803*, (5) 0.33040590027159966, (6) 0.3387236766265345, (7) 0.34144483357059846, (8) 0.3443545114152751 and (9) 0.34425046111387914. Silhouette scores for graduated students (number of clusters) are (2) *0.3968732975656796*, (3)

0.30584444035961056, (4) 0.3196159353243365, (5) 0.26455360953597873, (6) 0.278223566509381, (7) 0.2368155924273732, (8) 0.24514454671144176 and (9) 0.2517090725642439. Thus, optimal number of clusters for undergraduate students is 4 and graduated students it is 2 based on Silhouette score. This study adapts cluster amount of 4 to both student groups.

The next step is to implement the actual clustering with different algorithms starting from k-means. K-means clustering is implemented with k-means algorithm and visualized with seaborn (c.f. appendix 3). Figure 13 is illustrating the results of undergraduate and graduated students k-means clusters with two different features (x and y).



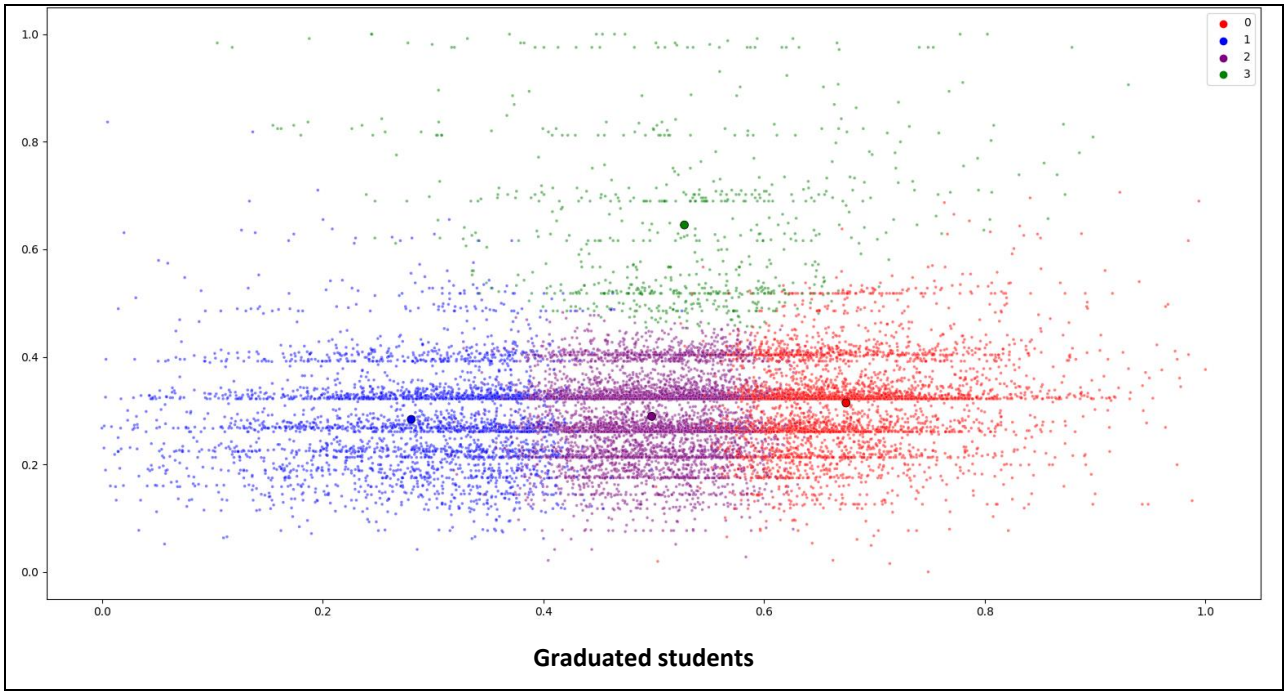


Figure 13. K-means clustering visualization

Four different student performance clusters (0-3) were identified using k-means as figure 13 is illustrating. These performance clusters can be utilized while predicting the student performance in a study programme. The model is a different when comparing the graduated students to undergraduate students. Model can be saved as a Pickle file including the min-max scaler. Scaler is required in the deployment since the values are received in general format, which are required to normalize with the same parameters.

The process for hierarchical clustering algorithm (agglomerative) is the same (c.f. appendix 4) but the final modeling is implemented with different algorithm. Agglomerative hierarchical clustering is bottom-up approach where all data points have their own clusters, which are at the end paired with other cluster (Pasupathi et al., 2021). This process continues until data points have reached to the top. The dendrogram is the basic format to present the linkages. Figure 14 is illustrating dendrogram of the dataset.

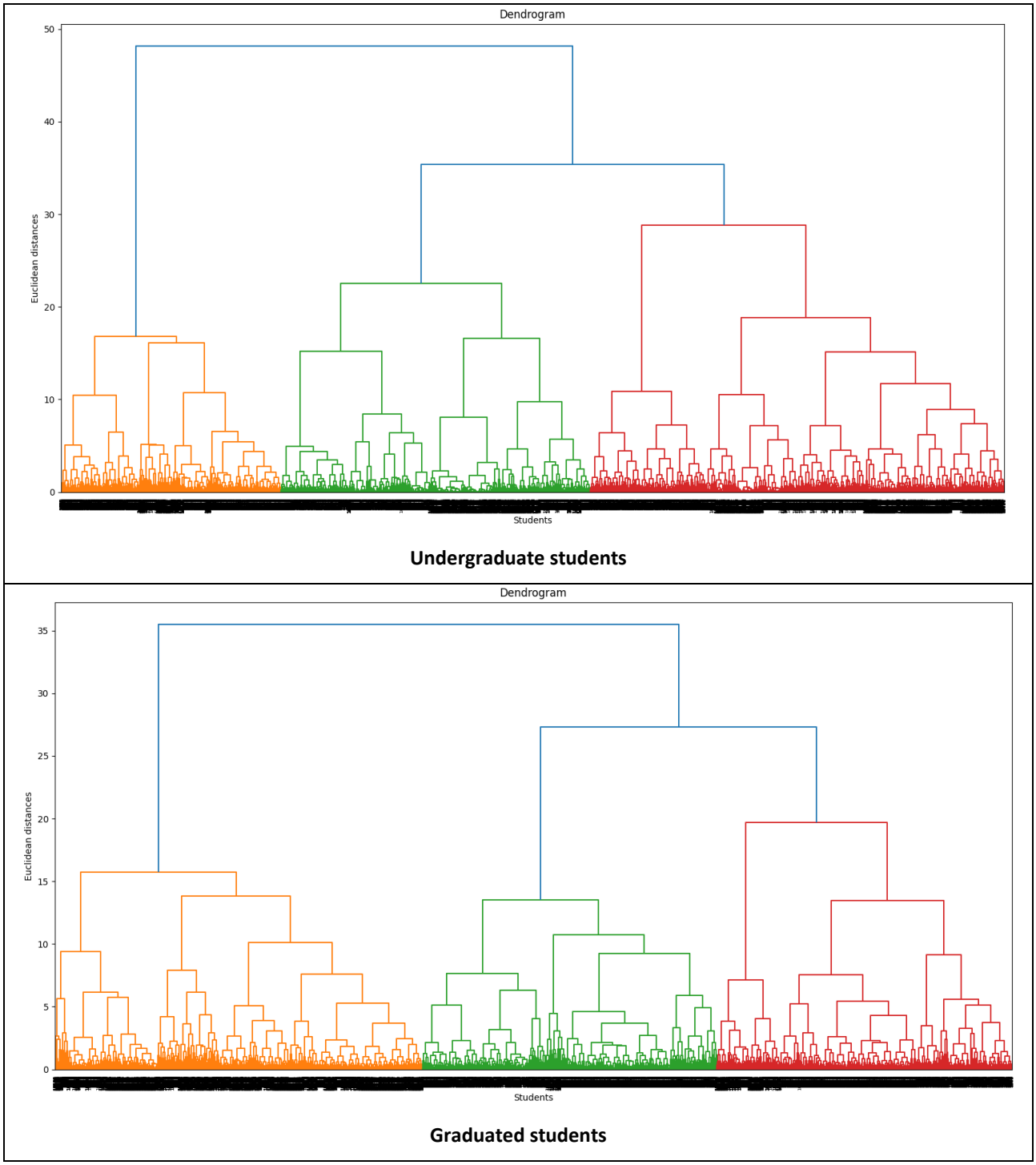


Figure 14. Dendrogram of the dataset

Dendrogram is illustrating hierarchical relationships in the data and optimal amount of the clusters in the dataset is three (orange, green and red) in both cases based on dendrograms which figure 14 is illustrating. Three was given a cluster amount based on dendrograms. Next step was to implement the hierarchical clustering. Figure 15 is illustrating clusters of the dataset in undergraduate and graduated students.

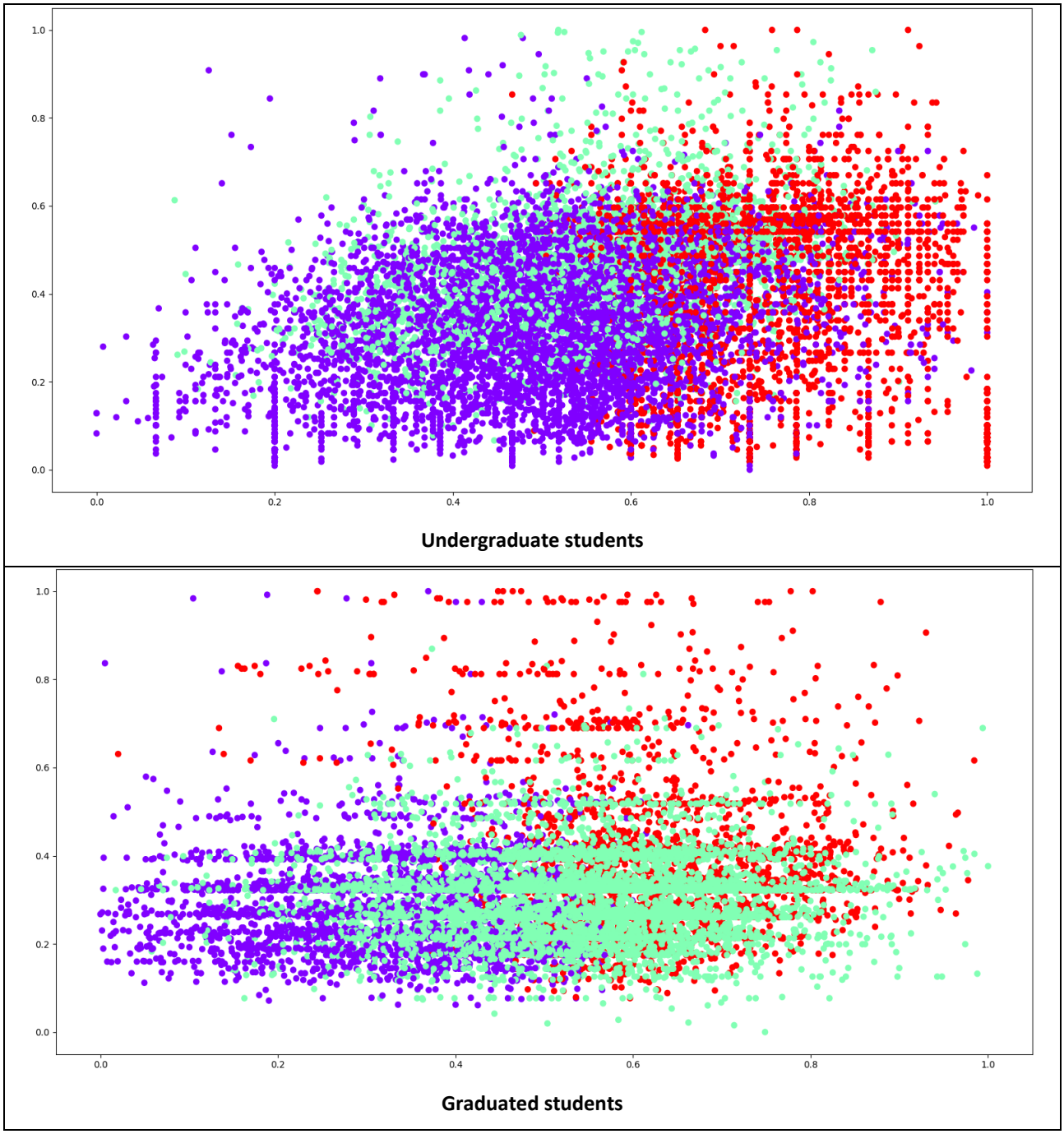


Figure 15. Clusters in hierarchical clustering

There are differences between undergraduate and graduated students as hierarchical clustering algorithm in figure 15 is illustrating. Hierarchical clustering is not that accurate when comparing the

results to the k-means clustering. Thus, the k-means was selected to clustering algorithm for students based on the results of k-means and hierarchical clustering algorithms.

4.4.3 Supervised Machine Learning Models

Supervised ML algorithms maps the input variables to output variable and strives to find a relationship between these two (Osisanwo et al., 2017). Thus, in supervised ML models the output is well known, and predictions are based on this output variable. This study is utilizing two different supervised learning algorithms, which are Support Vector Machine (SVM) and eXtreme Gradient Boosting (XGBoost). Full code of supervised ML models developed in this study are available in appendix 5.

The main objective of supervised ML algorithms in this study is to predict if student can finalize the studies on-time. Thus, the prediction feature (y) in this study is studentGOT, which was engineered in feature engineering phase. The process of development of supervised ML algorithm does not differ of development of unsupervised ML algorithms. Data is read to pandas dataframe at first, outliers are removed, and data is normalized. After that the dataset is split to training and test set and prediction variable (y) is defined. SVM is calculated after data has been split to test and training set. The code block 4 is presenting splitting the data to training and test set and SVM calculation.

```
# Split the dataset (train & test set)
X = f_dataset.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 10]].values
y = f_dataset.iloc[:, 9].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state = 7)

# Compute SVM and predict
classifier = SVC(kernel = 'linear', random_state = 7)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Code Block 4. Dataset split to training and test set and SVM computation

Full solution of SVM algorithm is available in appendix 5 (SVM.py). The dataframe X includes all relevant features which are required to make predictions of variable y. The SVM is calculated with linear kernel and after that predictions are implemented. The confusion matrix provides an outlook how well the model performs of predictions and it is the most classical approach to measure accuracy of binary classification. Table 5 is presenting confusion matrix of the SVM model.

Table 5. The confusion matrix

		Predicted	
		Negative	Positive
Actual	Negative	2721	268
	Positive	15	1258

The SVM model performs well based on the confusion matrix presented in table 5. The model has more true and false positives (green) where predicted value matches the actual value. Accuracy of the model is 93.36% based on sklearn accuracy score. The XGBoost is an algorithm, which is based on decision trees and it can be utilized as well to unstructured data (Chen & Guestrin, 2016). The process of adapting the XGBoost proceed the same way than SVM algorithm, but classifier is implemented a using XGBClassifier and the data is not normalized. The code block 5 is illustrating implementation XGBoost algorithm.

```
# XGBoost
model = XGBClassifier(booster='gblinear')
model.fit(X_train, y_train)

# Predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

Code Block 5. XGBoost algorithm

Full solution of XGBoost algorithm is available in appendix 5 (XGBoost.py). Accuracy of the XGBoost model is 88.36% based on sklearn accuracy score. SVM algorithm accuracy score is higher and, thus, it was selected as an algorithm for predicting the student performance.

4.5 Deployment

The final step of CRISP-DM process is deployment. The aim of the deployment phase is to deploy the ML models in existing or new systems of an organization (Clark, 2018). This study utilizes Flask to deploy ML models. Flask is lightweight web application framework for Python. Flask support different types of web application development. The objective of deployment is to provide the data in JSON format to applications, which request the information from ML models. Developed REST API provides k-means clustering based on user inputs. Code block 6 is presenting the implementation of REST API with Flask.

```
# -*- coding: utf-8 -*-
"""
JSON Deployment of Machine learning models

@author: Ari Sivula

Developed with Python version 3.8.5

"""

# Import required libraries
import pandas as pd
import pickle
from flask import Flask, request, jsonify

# Define the app
app = Flask(__name__)

# Read the required models and scalers
kmeans = pickle.load(open("./models/kmeans.pkl", "rb"))
kmeansScaler = pickle.load(open("./models/kmeans-scaler.pkl", "rb"))

@app.route('/')
def nothing():
    return "Absolutely Nothing"
```

```
@app.route('/predict', methods=['POST'])
def predict():
    student = request.get_json()
    print(student)

    # KMeans prediction (Normalize input and Predict)
    k_pd_dataframe = pd.DataFrame({'feature1': [student['feature1']], 'fea-
ture2': [student['feature2']], 'feature3': [student['feature3']], 'feature4':
[student['feature4']]})
    k_normData = kmeansScaler.transform(k_pd_dataframe)
    kmeansPred = kmeans.predict(k_normData)

    results = {
        'studentCategory': int(kmeansPred[0])
    }

    # Return the results
    return jsonify(results)

if __name__ == '__main__':
    app.run(debug=True, host='127.0.0.1', port=9696)
```

Code Block 6. Deployment Python script

The ML model is read from Pickle file including the scaler. These files were saved in modeling phase in the ML model development. Next, the user JSON input is read and after that the prediction is implemented. Finally, the value is returned to the application, which requested the prediction. Application can be, for example, Power BI or Excel. Code block 6 acts as a simple example, which can be extended in several ways to provide multiple ML models to different applications within an organization. Moreover, the REST API itself should be more secure in production environment.

5 Conclusions

The objective is this study was to explore ML in the higher education industry and answer three research questions. The objective of artificial intelligence in higher education industry is to support activities of an organization. Research questions of this study were answered based on practical implementation of ML-based system. The case organization in this study was Seinäjoki University of Applied Sciences and dataset utilized was VIRTA data. Moreover, Python was selected as the programming language for implementing the ML-based system. Selection was natural since several libraries exists to implement, for instance, data pre-processing and ML modeling, and Python tends to be an industry standard in development of ML-based solutions.

The first research question was “How CRISP-DM model can be adapted in the higher education industry?”. The CRISP-DM model is a good fit for developing ML-based solutions as well in higher education industry. Thus, the process of CRISP-DM provides all relevant steps required to implement the ML-based solution based on this study. The data preparation phase requires most of the computation power and time with the VIRTA dataset utilized in this study. On the other hand, multidimensional data always requires computation power and time in all environments. Moreover, this phase includes feature engineering, which requires the development of scripts as well to provide target feature based on the background features.

The second question was “Which unsupervised ML model performs best in student categorization?”. The students can be categorized in several ways and two different unsupervised ML algorithms were utilized in this study. This study adapted k-means and agglomerative hierarchical clustering based on earlier studies implemented in the higher education industry. The clustering was implemented separately for undergraduate and graduated students. The performance of k-means clustering was better based on this study and four different student clusters was found by the algorithm. The k-means clustering was selected to development phase. K-means clustering algorithm can be utilized to find the different student types in the case organization.

The third question was “Which supervised ML model performs best in predicting student performance?”. In this case the performance defines student’s ability to receive enough ECTS points during a semester and, thus, graduate on time. The prediction can be implemented with several algorithms. This study was adapting SVM and XGBoost algorithms to predict student performance based on the feature of graduate-on-time. The SVM algorithm performed better based on accuracy score. Thus, SVM was selected to deployment of performance prediction of a student.

This study was exploring ML in higher education industry in the Finnish case organization. The research was a single case study and, thus, it should be acknowledged that more research should be carried out to raise the validity of the research. Moreover, this study was adapting a single dataset (VIRTA), which does not necessarily include all relevant data to implement the prediction as precise as it should. This data comes from a single source and should aggregated with other useful data, for instance, from learning management systems. Moreover, collecting data directly from the students using, for example, crowdsourcing could be beneficial (Sivula & Kantola, 2014; Sivula, 2016). This type of a system could provide valuable insights directly from the students, which can be combined with existing data from several data sources in higher education industry.

References

Aiken, R.M., & Epstein, R.G. (2000). Ethical Guidelines for AI in Education: Starting a Conversation. *International Journal of Artificial Intelligence in Education*, 11, pp. 163-176

Akyol, K. (2017). A Study on the Diagnosis of Parkinson's Disease using Digitized Wacom Graphics Tablet Dataset. *Information Technology and Computer Science*, 12, pp. 45-51

Ayodele, T.O. (2021). Types of Machine Learning Algorithms. In Zhang, Y. *New Advances in Machine Learning*. InTech: Croatia.

Cauteruccio, F., Giudice, P.L., Musarella, L., Terracina, G., Ursino, D., & Virgili, L. (2020). A Lightweight Approach to Extract Interschema Properties from Structured, Semi-Structured and Unstructured Sources in a Big Data Scenario. *International Journal of Information Technology & Decision Making*, 19(3), pp. 849-889

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785-794.

Ciolacu, M., Tehrani, A. F., Binder, L., & Svasta, P. M. (2018). Education 4.0-artificial intelligence assisted higher education: Early recognition system with machine learning to support students' success. *IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 23-30.

Clark, A. (2018). The machine learning audit—crisp-dm framework. *ISACA J*, 1, pp. 1-6.

CSC. (2020). CSV-Aineistojen ohjeet. Retrieved from https://wiki.eduuni.fi/display/CSCVIRTA/CSV-Aineistojen+ohjeet

CSC. (2021a). VIRTA - a single location, the data of over one million students. Retrieved from https://www.csc.fi/en/home/-/asset_publisher/KbGpWCsF7Wos/content/virta?_101_INSTANCE_KbGpWCsF7Wos_redirect=%2F

CSC. (2021b). Tietovarannon koodistot. Retrieved from https://wiki.eduuni.fi/display/CSCVIRTA/Tietovarannon+koodistot

CSC. (2021c). VIRTA-Opintotietopalvelu [GitHub code repository]. Retrieved from https://github.com/CSCfi/VIRTA-Opintotietopalvelu

Cunningham, P., Cord, M., & Delany, S. J. (2008). Supervised learning. *Machine learning techniques for multimedia*. Springer: Berlin.

Dash, M., Liu, H., & Yao, J. (1997). Dimensionality reduction of unsupervised data. *Proceedings ninth ieee international conference on tools with artificial intelligence*, pp. 532-539.

Dong, G., & Liu, H. (2018). Feature engineering for machine learning and data analytics. CRC Press: Boca Raton.

Durga, S.N., & Rani, K.U. (2019). A perspective overview on machine learning algorithms. *International Conference On Computational And Bio Engineering*. Cham: Springer, pp. 353-364.

European Commission. (2018). Ethics and data protection. Retrieved from https://ec.europa.eu/info/sites/info/files/5._h2020_ethics_and_data_protection_0.pdf

Fabiano, N. (2019). *Ethics and the Protection of Personal Data. Systemics, Cybernetics and Informatics*, 17(2), pp. 58-64.

Fadahunsi, K. P., Akinlua, J. T., O'Connor, S., Wark, P. A., Gallagher, J., Carroll, C., Majeed, A., & O'Donoghue, J. (2019). Protocol for a systematic review and qualitative synthesis of information quality frameworks in eHealth. *BMJ open*, 9(3).

Fulbright. (2021). Higher Education in Finland. Retrieved from https://www.fulbright.fi/studies-and-research-finland/higher-education-finland

Ghahramani, Z. (2003). Unsupervised learning. *Summer School on Machine Learning*. Springer: Berlin.

Jaggia, S., Kelly, A., Lertwachara, K. & Chen, L. (2020). Applying the CRISP-DM Framework for Teaching Business Analytics. *Decision Sciences Journal of Innovative Education*, 18, 612-634.

Knight, W. (2016). This Factory Robot Learns a New Job Overnight. *MIT Technology Review*. Retrieved from https://www.technologyreview.com/2016/03/18/161519/this-factory-robot-learns-a-new-job-overnight/

Kollom, K., Tammets, K., Scheffel, M., Tsai, Y-S., Jivet, I., Muñoz-Merino, P.J., Moreno-Marcos, P.M., Whitelock-Wainwright, A., Calleja, A.R., Gasevic, D., Kloos, C.D., Drachsler, H., & Ley, T. (2021). A four-country cross-case analysis of academic staff expectations about learning analytics in higher education. *The Internet and Higher Education*, 49.

Kotsiantis, S. B., Kanellopoulos, D. and Pintelas, P. E. (2006). Data Preprocessing for Supervised Leaning. *International Journal of Computer Science*, 1, pp. 111-117.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1), pp. 6765-6816.

Looney, E.E., Liu, Z., Classen, A., Liu, H., Riedel, N., Braga, M., Pradeep, B., Augusto, A., Buonassisi, T. & Peters, I.M. (2021). Representative identification of spectra and environments (RISE) using k-means. *Progress in Photovoltaics: Research and Applications*, 29(2), pp. 200-211.

Martínez-Plumed, F., Contreras-Ochando, L., Ferri, C., Orallo, J. H., Kull, M., Lachiche, N., Ramírez-Quintana, M.J., & Flach, P. A. (2019). CRISP-DM twenty years later: From data mining processes to data science trajectories. *IEEE Transactions on Knowledge and Data Engineering*.

Milo, T., & Somech, A. (2020). Automating exploratory data analysis via machine learning: An overview. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 2617-2622.

Minedu. (2020). Steering, financing and agreements of higher education institutions, science agencies and research institutes. Retrieved from https://minedu.fi/en/steering-financing-and-agreements

Ogbuabor, G., & Ugwoke, F. N. (2018). Clustering algorithm for a healthcare dataset using silhouette score value. *International Journal of Computer Science & Information Technology*, 10(2), pp. 27-37.

Osisanwo, F. Y., Akinsola, J. E. T., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017). Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology*, 48(3), pp. 128-138.

Pasupathi, S., Shanmuganathan, V., Madasamy, K., Yesudhas, H. R., & Kim, M. (2021). Trend analysis using agglomerative hierarchical clustering approach for time series big data. *The Journal of Supercomputing*, pp. 1-20.

Patgiri, R., Katari, H., Kumar, R., & Sharma, D. (2019). Empirical study on malicious URL detection using machine learning. *International Conference on Distributed Computing and Internet Technology*. Cham: Springer, pp. 380-388.

Qin, Y., Zhang, S., Zhu, X., Zhang, J., & Zhang, C. (2007). Semi-parametric optimization for missing data imputation. *Applied Intelligence*, 27(1), pp. 79-88.

Romero, C., & Ventura, S. (2010). Educational Data Mining: A Review of the State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(6), pp. 601-618.

Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. *WIREs – Data Mining and Knowledge Discorery*, 10(3).

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science*, 181, pp. 526-534.

SeAMK. (2021a). SeAMK Introduction. Retrieved from https://www.seamk.fi/en/aboutus/seamk-introduction/

SeAMK. (2021b). Privacy Policy. Retrieved from https://www.seamk.fi/en/privacy-policy/

Sethi, J., & Mittal, M. (2019). Ambient air quality estimation using supervised learning techniques. *EAI Endorsed Transactions on Scalable Information Systems*, 6(22).

Shafique, U. & Qaiser, H. (2014). A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA). *International Journal of Innovation and Scientific Research*, 12(1), pp. 217-222.

Siemens, G., & Baker, R.S.J.d. (2012). Learning analytics and educational data mining: towards communication and collaboration. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge* (LAK '12), pp. 252–254.

Singh R., & Pal S. (2021). A Critical Review on Educational Data Mining Segment: A New Perspective. In Jeena Jacob I., Kolandapalayam Shanmugam, S., Piramuthu, S., & Falkowski-Gilski, P. (eds). *Data Intelligence and Cognitive Informatics. Algorithms for Intelligent Systems*. Springer: Singapore.

Sivula, A. (2016). Generic Crowdsourcing Model for Holistic Innovation Management. *Acta Wasaensia*, 355.

Sivula, A., & Kantola, J. (2014). Crowdsourcing in a project lifecycle. *International Conference on Knowledge Management in Organizations*. Cham: Springer.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press: Cambridge.

Thakar, P., Mehta, A., & Manisha. (2015). Performance analysis and prediction in educational data mining: A research travelogue. *International Journal of Computer Applications*, 110(15), pp. 60-68.

Vaidya A., & Saini J.R. (2021) A Framework for Implementation of Learning Analytics and Educational Data Mining in Traditional Learning Environment. In Fong S., Dey N., & Joshi A. (eds). *ICT Analysis and Applications. Lecture Notes in Networks and Systems*, 154. Springer: Singapore.

Wand, Y., & Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11), pp. 86-95.

Wirth, R. & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*.

Yin, R. (2009). Case study research: design and methods (4th edition). Sage Publications Inc: California.

Yu, L.L. (2004). Efficient feature Selection via Analysis of Relevance and Redundancy. *Journal of Machine Learning Research*, 5, pp. 1205-1224.

Zhang, S.Z. (2002). Data Preparation for Data Mining. *Applied Artificial Intelligence*, 17, pp. 375 - 381.

Zhou, X., & Belkin, M. (2014). Semi-supervised learning. *Academic Press Library in Signal Processing*, 1, pp. 1239-1269.

Appendices

Appendix 1. Survey to Key Stakeholders

Obstacles of your work

- What data would you need in your work, what is not currently available or is difficult to obtain? What data would improve and streamline your operations as well as the operations of the entire organization?

Metrics and KPI's

- What kind of real-time indicators and visualizations would support and enhance teaching and RDI activities? What kind of estimation information should be used to support operations based on indicators? What is required to estimate?

Objective

- How do we support and enhance SeAMK's teaching and RDI activities with data?

Challenges

- What challenges or problems are we trying to solve with the data?

Data producers

- Who are SeAMK's internal and external stakeholders who provide us a (useful) data?

Data consumers

- Which internal and external stakeholders use data, analytics, or artificial intelligence? How do they benefit from data and analytics?

Existing data sources

- What data do we already have? From which sources does this data come? Are there any restrictions on obtaining and utilizing data from these data sources? Is there access to these data sources?

New data sources

- What new data is required? From which sources does the data come? Are there any restrictions on obtaining and utilizing data from these data sources? Is there access to these data sources?

Tools and technology

- What technology or IT systems are currently being used? What new technology or systems are required?

Artificial intelligence

- What kind of artificial intelligence applications could we utilize as part of our operations? Do we already have these in place? What kind of machine learning models are required? Is artificial intelligence the appropriate solution?

Competences

- What knowledge and skills does SeAMK already have? What new skills should be acquired?

Channels

- What communication channels do we want to use? How do we communicate? Which channel should we activate?

Strengths

- Where are we good and what is already working well for us?

Weaknesses

- What should we develop and what risks can be identified?

Other comments

- What else came to your mind? What else should be considered?

Your name and/or school

- If you wish, you can enter your name and/or school here

Appendix 2. Pre-Processing and Feature Engineering Python Scripts

Pre-processing script (Preprocessing.py)

```
# -*- coding: utf-8 -*-
"""
Preprocessing script for VIRTA XML data:
    - Collects the most crucial variables for feature engineering and machine
learning algorithms
    - Extracts some features from the VIRTA data, which is necessary in this
stage
    - Records the pre-processed data to CSV format for later use
    - Removes unethical data from original dataset

@author: Ari Sivula

Developed with Python version 3.8.5

"""

import pandas as pd
import numpy as np
import xml.etree.ElementTree as et
import datetime

# FIRST, extract data from the VIRTA dataset
xtree = et.parse("virta-out.xml")
xroot = xtree.getroot()

df_studentCols = ["studentID", "KirjoihintuloPvm", "AlkuPvm", "LoppuPvm",
"PvmTotal", "Tyyppi", "Koulutusala", "Koulutusmoduulitunniste", "TilaKoodi",
"Koulutuskoodi", "LukukausiCount", "Koulutuskieli", "Luokittelu", "total-GPA",
"Valmistunut", "Opintosuoritukset"]
allRows = []
s_ID = 1000 # Unique ID (beginning of 1001)

# Iterate all nodes
for node in xroot.iter("{urn:mace:funet.fi:virta/2015/09/01}Opiskelija"):
    s_ID = s_ID + 1
    s_kirjoihintulopvm =
node.find("{urn:mace:funet.fi:virta/2015/09/01}KirjoihintuloPvm")

    if s_kirjoihintulopvm != None:
        s_kirjoihintulopvm =
node.find("{urn:mace:funet.fi:virta/2015/09/01}KirjoihintuloPvm").text if node
is not None else None
    else:
        s_kirjoihintulopvm = np.NaN

    for opiskeluoikeudet in
node.iter("{urn:mace:funet.fi:virta/2015/09/01}Opiskeluoikeudet"):
        for opiskeluoikeus in opiskeluoikeudet.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Opiskeluoikeus"):
            s_AlkuPvm = opiskeluoikeus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}AlkuPvm")
            s_LoppuPvm = opiskeluoikeus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}LoppuPvm")
```

```
            s_Tyyppi = opiskeluoikeus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Tyyppi")
            s_Koulutusala = opiskeluoikeus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutusala")

            if s_AlkuPvm != None:
                s_AlkuPvm =
opiskeluoikeus.find("{urn:mace:funet.fi:virta/2015/09/01}AlkuPvm").text if
node is not None else None
            else:
                s_AlkuPvm = np.NaN

            if s_LoppuPvm  != None:
                s_LoppuPvm =
opiskeluoikeus.find("{urn:mace:funet.fi:virta/2015/09/01}LoppuPvm").text if
node is not None else None
            else:
                s_LoppuPvm = np.NaN

            if (s_AlkuPvm != np.NaN) and (s_LoppuPvm != np.NaN):
                tempPvm = datetime.datetime.strptime(s_LoppuPvm, '%Y-%m-%d') -
datetime.datetime.strptime(s_AlkuPvm, '%Y-%m-%d')
                s_PvmTotal = tempPvm.days
            else:
                s_PvmTotal = np.NaN

            if s_Tyyppi != None:
                s_Tyyppi =
opiskeluoikeus.find("{urn:mace:funet.fi:virta/2015/09/01}Tyyppi").text if node
is not None else None
            else:
                s_Tyyppi = np.NaN

            if s_Koulutusala != None:
                s_Koulutusala = opiskeluoikeus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutusala").text if node is not None else None
            else:
                s_Koulutusala = np.NaN

            for tila in opiskeluoikeus.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Tila"):
                s_TilaKoodi = tila.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koodi")

                if s_TilaKoodi != None:
                    s_TilaKoodi =
tila.find("{urn:mace:funet.fi:virta/2015/09/01}Koodi").text if node is not
None else None
                else:
                    s_TilaKoodi = np.NaN

            for jakso in
opiskeluoikeus.iter("{urn:mace:funet.fi:virta/2015/09/01}Jakso"):
                s_koulutusmoduulitunniste = jakso.attrib.get("koulutusmoduuli-
tunniste")

                s_Koulutuskoodi = jakso.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutuskoodi")
```

```
                s_Koulutuskieli = jakso.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutuskieli")
                s_Luokittelu = jakso.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Luokittelu")

                if s_Koulutuskoodi != None:
                    s_Koulutuskoodi = jakso.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutuskoodi").text if node is not None else None
                else:
                    s_Koulutuskoodi = np.NaN

                if s_Koulutuskieli != None:
                    s_Koulutuskieli = jakso.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutuskieli").text if node is not None else None
                else:
                    s_Koulutuskieli = np.NaN

                if s_Luokittelu != None:
                    s_Luokittelu =
jakso.find("{urn:mace:funet.fi:virta/2015/09/01}Luokittelu").text if node is
not None else None
                else:
                    s_Luokittelu = np.NaN

    # LukukausiIlmoittautumiset
    s_LukukausiCount = None
    s_LukukausiCount = node.find("{urn:mace:funet.fi:virta/2015/09/01}Lukukau-
siIlmoittautumiset")

    if s_LukukausiCount != None:
        s_LukukausiCount = len(node.find("{urn:mace:fu-
net.fi:virta/2015/09/01}LukukausiIlmoittautumiset"))
    else:
        s_LukukausiCount = np.NaN

    grade_rows = [] # All data
    s_WMeanCalc = 0
    s_WMeanEctsCalc = 0
    s_Valmistunut = 0

    for opintosuoritukset in node.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Opintosuoritukset"):
        for opintosuoritus in opintosuoritukset.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Opintosuoritus"):
            s_SuoritusPvm = opintosuoritus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}SuoritusPvm")
            s_Nimi = opintosuoritus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Nimi")
            s_Kieli = opintosuoritus.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Kieli")

            if s_SuoritusPvm != None:
                s_SuoritusPvm = opintosuori-
tus.find("{urn:mace:funet.fi:virta/2015/09/01}SuoritusPvm").text if node is
not None else None
            else:
                s_SuoritusPvm = 'nan' # For string formatting (csv)
```

```
            for laajuus in opintosuori-
tus.iter("{urn:mace:funet.fi:virta/2015/09/01}Laajuus"):
                s_Opintopiste =
laajuus.find("{urn:mace:funet.fi:virta/2015/09/01}Opintopiste")

                if s_Opintopiste != None:
                   s_Opintopiste =
laajuus.find("{urn:mace:funet.fi:virta/2015/09/01}Opintopiste").text if node
is not None else None
                else:
                   s_Opintopiste = 'nan' # For string formatting (csv)

            for arvosana in opintosuoritus.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Arvosana"):
                s_Arvosana = arvosana.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Viisiportainen")

                if s_Arvosana != None:
                   s_Arvosana = arvo-
sana.find("{urn:mace:funet.fi:virta/2015/09/01}Viisiportainen").text if node
is not None else None

                   if s_Arvosana.isnumeric():
                       s_WMeanCalc = s_WMeanCalc + (float(s_Arvo-
sana)*float(s_Opintopiste))
                       s_WMeanEctsCalc = s_WMeanEctsCalc + float(s_Opin-
topiste)

                else:
                   s_Arvosana = 'nan' # For string formatting (csv)
                   s_Valmistunut = 1

            if s_Nimi != None:
                s_Nimi = opintosuori-
tus.find("{urn:mace:funet.fi:virta/2015/09/01}Nimi").text if node is not None
else None
            else:
                s_Nimi = 'nan' # For string formatting (csv)

            if s_Kieli != None:
                s_Kieli = opintosuori-
tus.find("{urn:mace:funet.fi:virta/2015/09/01}Kieli").text if node is not None
else None
            else:
                s_Kieli = 'nan' # For string formatting (csv)

            for koulutusala in opintosuoritus.iter("{urn:mace:fu-
net.fi:virta/2015/09/01}Koulutusala"):
                s_Ohjausala = koulutusala.find("{urn:mace:fu-
net.fi:virta/2015/09/01}Koodi")

                if s_Ohjausala != None:
                   s_Ohjausala = koulu-
tusala.find("{urn:mace:funet.fi:virta/2015/09/01}Koodi").text if node is not
None else None
                else:
                   s_Ohjausala = np.NaN
```

```
            # Finally add row to the list (only one row)
            grade_rows.append({"SuoritusPvm": s_SuoritusPvm, "Opintopiste":
s_Opintopiste,
                               "Arvosana": s_Arvosana, "Nimi": s_Nimi,
                               "Kieli": s_Kieli, "Ohjausala": s_Ohjausala})

    # Finally, calculate weighted arithmetic mean and append
    if s_WMeanCalc != 0:
        s_GPA = s_WMeanCalc/s_WMeanEctsCalc

    else:
        s_GPA = np.NaN

    allRows.append({"studentID": s_ID, "KirjoihintuloPvm": s_kirjoihintulopvm,
"AlkuPvm": s_AlkuPvm,
                "LoppuPvm": s_LoppuPvm, "PvmTotal": s_PvmTotal, "Tyyppi":
s_Tyyppi,
                "Koulutusala": s_Koulutusala, "Koulutusmoduulitunniste":
s_koulutusmoduulitunniste,
                "TilaKoodi": s_TilaKoodi, "Koulutuskoodi": s_Koulutuskoodi,
"LukukausiCount": s_LukukausiCount,
                "Koulutuskieli": s_Koulutuskieli, "Luokittelu": s_Luokittelu,
                "total-GPA": s_GPA, "Valmistunut": s_Valmistunut, "Opintosuo-
ritukset": grade_rows})

# SECOND, create the Pandas dataframe from the results and save to CSV
df_students = pd.DataFrame(allRows, columns = df_studentCols)

# THIRD, save data to CSV
df_students.to_csv('students.csv', encoding='utf-8', index=False)
```

Feature engineering script (FeatureEngineering.py)

```
# -*- coding: utf-8 -*-
"""
Feature engineering script from VIRTA data

@author: Ari Sivula

Developed with Python version 3.8.5

"""

import numpy
import pandas as pd
import ast
import numpy as np
from sklearn.impute import SimpleImputer

# Read the dataset
dataset = pd.read_csv('students.csv')

OpintosRows = []

df_OpintosColums = ["studentID", "SuoritusPvm", "Opintopiste", "Arvosana",
"Nimi", "Kieli", "Ohjausala"]
```

```
# Copy all courses to Pandas dataframe
for index in dataset.index:
    if not dataset["Opintosuoritukset"].iloc[index] == '[]':
        OpintosRows.append(str(dataset["studentID"].iloc[index]) + "^^"+ da-
taset["Opintosuoritukset"].iloc[index])

# Define DataFrame columns
df_Opintosuoritukset = pd.DataFrame(columns = df_OpintosColums)

# Go through the list and add content to Pandas dataframe
_studentid = ''
_courses = ''
for i in range(len(OpintosRows)):
    _studentid = OpintosRows[i].split('^^')[0]
    _courses = OpintosRows[i].split('^^')[1]
    # First, read studies of a single student
    df_Opintosuoritukset = df_Opintosuoritukset.append(ast.lit-
eral_eval(_courses), ignore_index = False)
    # Second, update the row with a student ID
    df_Opintosuoritukset['studentID'] = df_Opintosuoritukset.studen-
tID.fillna(_studentid)

# Remove degree from graduated students
df_Opintosuoritukset = df_Opintosuoritukset[df_Opintosuoritukset['Arvosana']
!= 'nan']

# Set the dtypes
df_Opintosuoritukset['studentID'] = pd.to_numeric(df_Opintosuoritukset.studen-
tID, errors='coerce')
df_Opintosuoritukset['SuoritusPvm'] = pd.to_datetime(df_Opintosuoritukset.Suo-
ritusPvm, errors='coerce')
df_Opintosuoritukset['Opintopiste'] = pd.to_numeric(df_Opintosuoritukset.Opin-
topiste, errors='coerce')
df_Opintosuoritukset['Arvosana'] = pd.to_numeric(df_Opintosuoritukset.Arvo-
sana, errors='coerce')
df_Opintosuoritukset['Nimi'] = df_Opintosuoritukset['Nimi'].astype(str)
df_Opintosuoritukset['Kieli'] = df_Opintosuoritukset['Kieli'].astype(str)
df_Opintosuoritukset['Ohjausala'] = pd.to_numeric(df_Opintosuoritukset.Ohjaus-
ala, errors='coerce')

# Impute missing grades (HYV grades with mean)
imputerG = SimpleImputer(missing_values=np.NaN, strategy='mean')
df_Opintosuoritukset.Arvosana = imputerG.fit_transform(df_Opintosuorituk-
set['Arvosana'].values.reshape(-1,1))[:,0]

# Reorder the dataframe and remove unnecessary columns
df_Opintosuoritukset = df_Opintosuoritukset.reset_index()
df_Opintosuoritukset = df_Opintosuoritukset.drop(columns=['index', 'Nimi',
'Kieli', 'Ohjausala'])

# Calculate total amount of ECTS per student
df_OpintosTotal = df_Opintosuoritukset.groupby(['studentID'])['Opin-
topiste'].sum().reset_index()
df_OpintosTotal.rename(columns={'Opintopiste': 'total-ECTS'}, inplace=True)

# Sum the amount of grades per a semester, per a student and create new pandas
dataframe
```

```
df_OpintosPerSemester = df_Opintosuoritukset.set_index('Suori-
tusPvm').groupby('studentID')["Opintopiste"].resample("2QS-JUN").sum()
df_OpintosPerSemester = pd.DataFrame(df_OpintosPerSemester).reset_index()
df_OpintosPerSemester = df_OpintosPerSemester.drop(columns=['SuoritusPvm'])
df_OpintosPerSemester = df_OpintosPerSemester[df_OpintosPerSemester['Opin-
topiste'] != 0]
df_OpintosPerSemester = df_OpintosPerSemester.groupby(['studentID'])['Opin-
topiste'].mean().reset_index()

# Calculate GPA per semester, per a student and create new pandas dataframe
df_GPAPerSemester = df_Opintosuoritukset.set_index('Suori-
tusPvm').groupby('studentID')["Arvosana"].resample("2QS-JUN").mean()
df_GPAPerSemester = pd.DataFrame(df_GPAPerSemester).reset_index()
df_GPAPerSemester = df_GPAPerSemester.drop(columns=['SuoritusPvm'])
df_GPAPerSemester = df_GPAPerSemester[df_GPAPerSemester['Arvosana'] != 0]
df_GPAPerSemester = df_GPAPerSemester.groupby(['studentID'])['Arvo-
sana'].mean().reset_index()

# Aggregate the datasets (dataset + df_OpintosPerSemester + df_GPAPerSemester)
dataset = dataset.join(df_OpintosPerSemester.set_index('studentID')[['Opin-
topiste']], on='studentID')
dataset = dataset.join(df_GPAPerSemester.set_index('studentID')[['Arvosana']],
on='studentID')
dataset = dataset.join(df_OpintosTotal.set_index('studentID')[['total-ECTS']],
on='studentID')
dataset.rename(columns={'Arvosana': 'semester-GPA', 'Opintopiste': 'semester-
ECTS'}, inplace=True)

# A binary feature for a student, which defines a student performance (Gradu-
ate-On-Time, GOT)
OptLukukaudet = pd.read_csv('OptLukukaudet.csv', sep=';')

dataset['Opt_ECTSCount'] = np.NaN
dataset['Opt_SemesterCount'] = np.NaN

# Find optimal amount of ECTS and semesters and aggregate
dataset.set_index('Koulutusmoduulitunniste', inplace=True)
dataset.update(OptLukukaudet.set_index('Koulutusmoduulitunniste'))
dataset = dataset.reset_index()
dataset = dataset.reset_index()
dataset = dataset.drop(columns=['index'])

# Set the dataset to appropriate order (features last)
dataset = dataset[['studentID', 'KirjoihintuloPvm', 'AlkuPvm', 'LoppuPvm',
'PvmTotal', 'Tyyppi', 'Koulutusala', 'Koulutusmoduulitunniste', 'TilaKoodi',
'Koulutuskoodi', 'Koulutuskieli', 'Luokittelu', 'Valmistunut', 'Opintosuori-
tukset', 'LukukausiCount', 'semester-ECTS', 'semester-GPA', 'total-ECTS', 'to-
tal-GPA', 'Opt_ECTSCount', 'Opt_SemesterCount']]

# Set the binary variable (student has graduated AND number of semesters is
smaller than optimal (target duration of the degree))
dataset['studentGOT'] = np.where((dataset['Valmistunut'] == 1) & (dataset['Lu-
kukausiCount'] <= dataset['Opt_SemesterCount']), 1, 0)

# Save
dataset.to_csv('students-Features.csv', encoding='utf-8', index=False)
```

Appendix 3. Explanatory Data Analysis Python Script

EDA.py

```
# -*- coding: utf-8 -*-
"""
Exploratory Data Analysis (EDA) for the VIRTA dataset

@author: Ari Sivula

Developed with Python version 3.8.5

"""

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import pingouin as pg
from sklearn.impute import SimpleImputer

sns.set(color_codes=True)

# Read the dataset
dataset = pd.read_csv('students-Features.csv')

# Drop irrelevant columns and encode the Koulutuskieli column
dataset['Koulutuskieli'] = dataset["Koulutuskieli"].astype('category')
dataset['Koulutuskieli_cat'] = dataset['Koulutuskieli'].cat.codes
dataset = dataset.drop(columns=['studentID', 'KirjoihintuloPvm', 'AlkuPvm',
'LoppuPvm', 'Koulutusmoduulitunniste', 'Koulutuskieli', 'Opintosuoritukset'])

# Begin EDA analysis
dataset.head()

# Number of observations and variables
dataset.shape

# Datatypes
dataset.info()

# Spread for numerical values
dataset.describe()

# Amount of null values with missingno visualization
dataset.isnull().sum()
msno.bar(dataset)

# Missing data imputation with mean
values = dataset.values
imputer = SimpleImputer(missing_values=np.NaN, strategy='median')
f_dataset = pd.DataFrame(imputer.fit_transform(values))
f_dataset.columns = dataset.columns
del dataset
dataset = f_dataset
del f_dataset
```

```
# Visualize the dataset
dataset[['Tyyppi', 'Koulutusala', 'TilaKoodi', 'Koulutuskoodi', 'Luokittelu',
'Valmistunut', 'Opt_ECTSCount', 'Opt_SemesterCount', 'Koulutuski-
eli_cat']].hist(xlabelsize=8, ylabelsize=8)

# Correlation heatmap
corr = dataset.corr()
plt.figure(figsize=(12, 10))

sns.heatmap(corr[(corr >= 0.4) | (corr <= -0.4)],
            cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot=True, annot_kws={"size": 8}, square=True)


# Feature centric visualizations
fig, ax = plt.subplots(2,2)
sns.distplot(dataset['semester-ECTS'], color='b', hist_kws={'alpha': 0.4},
ax=ax[0,0])
sns.distplot(dataset['total-ECTS'], color='b', hist_kws={'alpha': 0.4},
ax=ax[0,1])
sns.distplot(dataset['semester-GPA'], color='g', hist_kws={'alpha': 0.4},
ax=ax[1,0])
sns.distplot(dataset['total-GPA'], color='g', hist_kws={'alpha': 0.4},
ax=ax[1,1])
plt.show()

# Compute the ANOVA
dataset = dataset.rename(columns={'semester-ECTS': 'semesterECTS'})
anova = pg.anova(dv='semesterECTS', between=['Koulutusala', 'Valmistunut'],
data=dataset)
```

Appendix 4. Unsupervised Machine Learning Python Scripts

K-means clustering (Kmeans.py)

```
# -*- coding: utf-8 -*-
"""
Machine learning algorithm for VIRTA data

@author: Ari Sivula

K-means algorithm for clustering students with their performance

Developed with Python version 3.8.5

"""

import numpy
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import pickle
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
from sklearn.metrics import silhouette_score
from scipy import stats

""" 1. PRE-PROCESSING OF THE DATA FOR K-MEANS ALGORITHM """

# Read the dataset
dataset = pd.read_csv('students-Features.csv')

# Drop the students which are not graduated
dataset = dataset[dataset['Valmistunut'] == 1]
dataset = dataset.reset_index()

# Drop not required columns and rows
dataset = dataset[dataset['semester-ECTS'].notna()]
dataset = dataset[dataset['total-GPA'].notna()]
dataset = dataset.drop(columns=['Opintosuoritukset', 'index',
'KirjoihintuloPvm', 'AlkuPvm', 'LoppuPvm', 'Koulutusmoduulitunniste', 'Koulu-
tuskieli', 'Valmistunut', 'Luokittelu', 'Opt_ECTSCount', 'Opt_SemesterCount'])
dataset = dataset.reset_index()
dataset = dataset.drop(columns=['index', 'studentID'])

# Missing data imputation
values = dataset.values
imputer = SimpleImputer(missing_values=np.NaN, strategy='mean')
f_dataset = pd.DataFrame(imputer.fit_transform(values))
f_dataset.columns = dataset.columns
del dataset

""" 2. KNN ALGORITHM """
f_dataset = f_dataset.drop(columns=['Tyyppi', 'TilaKoodi', 'Koulutuskoodi',
'Koulutusala', 'PvmTotal', 'LukukausiCount', 'studentGOT'])

# Remove outliers using z score (3 sigmas)
```

```
z_scores = stats.zscore(f_dataset)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
f_dataset = f_dataset[filtered_entries]

# Data normalization
np_x = f_dataset.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(np_x)
normdata = pd.DataFrame(x_scaled)
normdata.columns = f_dataset.columns
f_dataset = normdata
del normdata

# FIRST, find optimal amount of k's (model validation)
# The Elbow Diagram
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(f_dataset)
    distortions.append(kmeanModel.inertia_)

plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title("The Elbow Method (optimal amount of k's)")
plt.show()

# The Silhouette Score
range_n_clusters = list (range(2,10))
print ("Number of clusters from 2 to 9: \n", range_n_clusters)

for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict(f_dataset)
    centers = clusterer.cluster_centers_

    score = silhouette_score(f_dataset, preds)
    print("For n_clusters = {}, silhouette score is {})".format(n_clusters,
score))

# SECOND, compute KNN with optimal amount of k's
clusterAmount = 4
kmeans = KMeans(n_clusters = clusterAmount, init = 'k-means++', random_state =
42)
y_kmeans = kmeans.fit_predict(f_dataset)

""" 3. VISUALIZATION """
colors = ['red', 'blue', 'purple', 'green']
ax = sns.scatterplot(x=f_dataset.iloc[:, 1], y=f_dataset.iloc[:, 0],
hue=kmeans.labels_, palette=colors, alpha=0.5, s=7)
ax = sns.scatterplot(x=kmeans.cluster_centers_[:, 1], y=kmeans.cluster_cen-
ters_[:, 0], hue=range(clusterAmount), palette=colors, s=50, ec='black', leg-
end=False, ax=ax)
plt.show()
```

```
""" 4. TEST PREDICTION """
# Test with normalized data
testPrediction = kmeans.predict([[0.91, 0.92, 0.93, 0.95]])

""" 5. SAVE THE MODEL AND SCALER FOR POSSIBLE DEPLOYMENT """
pickle.dump(kmeans, open("student-KMeans.pkl", "wb"))
pickle.dump(min_max_scaler, open("student-KMeans-Scaler.pkl", "wb"))
```

Hierarchical clustering algorithm (Agglomerative) (Hc.py)

```
# -*- coding: utf-8 -*-
"""
Machine learning algorithm for VIRTA data

@author: Ari Sivula

Hierarchical clustering algorithm (Agglomerative) for clustering students with
their performance

Developed with Python version 3.8.5

"""

import numpy
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pickle
import scipy.cluster.hierarchy as sch
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
from scipy import stats
from sklearn.cluster import AgglomerativeClustering

""" 1. READ THE DATASET AND PREPROSESS THE DATA """
dataset = pd.read_csv('students-Features.csv')

# Drop the students which are not graduated
dataset = dataset[dataset['Valmistunut'] == 1]
dataset = dataset.reset_index()

# Drop not required columns and rows
dataset = dataset[dataset['semester-ECTS'].notna()]
dataset = dataset[dataset['total-GPA'].notna()]
dataset = dataset.drop(columns=['Opintosuoritukset', 'KirjoihintuloPvm',
'AlkuPvm', 'LoppuPvm', 'Koulutusmoduulitunniste', 'Koulutuskieli', 'Valm-
istunut', 'Luokittelu', 'Opt_ECTSCount', 'Opt_SemesterCount'])
dataset = dataset.reset_index()
dataset = dataset.drop(columns=['index', 'studentID'])

# Missing data imputation
values = dataset.values
imputer = SimpleImputer(missing_values=np.NaN, strategy='mean')
f_dataset = pd.DataFrame(imputer.fit_transform(values))
f_dataset.columns = dataset.columns
del dataset

""" 2. HC ALGORITHM """
```

```
f_dataset = f_dataset.drop(columns=['Tyyppi', 'TilaKoodi', 'Koulutuskoodi',
'Koulutusala', 'studentGOT'])

# Remove outliers using z score (3 sigmas)
z_scores = stats.zscore(f_dataset)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
f_dataset = f_dataset[filtered_entries]

# Data normalization
np_x = f_dataset.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(np_x)
normdata = pd.DataFrame(x_scaled)
normdata.columns = f_dataset.columns
f_dataset = normdata
del normdata

# Compute HC
# Utilize dendrogram to find the optimal number of clusters
dendrogram = sch.dendrogram(sch.linkage(f_dataset, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Students')
plt.ylabel('Euclidean distances')
plt.show()

# Fit HC to the dataset
hc = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean', linkage =
'ward')
y_hc = hc.fit_predict(f_dataset)

""" 3. FINAL VISUALIZATION """
plt.figure(figsize=(10, 7))
plt.scatter(f_dataset.iloc[:, 4], f_dataset.iloc[:, 3], c=hc.labels_,
cmap='rainbow')

""" 4. SAVE THE MODEL FOR POSSIBLE DEPLOYMENT """
pickle.dump(hc, open("student-HC.pkl", "wb"))
pickle.dump(min_max_scaler, open("student-hc-Scaler.pkl", "wb"))
```

Appendix 5. Supervised Machine Learning Python Scripts

Support vector machine algorithm (SVM.py)

```
# -*- coding: utf-8 -*-
"""
Machine learning algorithm for VIRTA data

@author: Ari Sivula

Support Vector Machine algorithm for predicting student performance.

Developed with Python version 3.8.5

"""

import numpy
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from scipy import stats

""" 1. READ THE DATASET AND PREPROSESS THE DATA """
dataset = pd.read_csv('students-Features.csv')

# Categories
dataset['Koulutusmoduulitunniste'] = dataset["Koulutusmoduulitunniste"].as-
type('category')
dataset['Koulutusmoduulitunniste_cat'] = dataset['Koulutusmoduulitunnis-
te'].cat.codes

# Drop not required columns and rows
dataset = dataset[dataset['semester-ECTS'].notna()]
dataset = dataset[dataset['total-GPA'].notna()]
dataset = dataset[dataset['LukukausiCount'].notna()]
dataset = dataset[dataset['Opt_ECTSCount'].notna()]
dataset = dataset.drop(columns=['Opintosuoritukset', 'KirjoihintuloPvm',
'AlkuPvm', 'LoppuPvm', 'Koulutusmoduulitunniste', 'Koulutuskieli', 'Luokit-
telu'])
dataset = dataset.reset_index()
dataset = dataset.drop(columns=['index', 'studentID'])

# Missing data imputation
values = dataset.values
imputer = SimpleImputer(missing_values=np.NaN, strategy='mean')
f_dataset = pd.DataFrame(imputer.fit_transform(values))
f_dataset.columns = dataset.columns
del dataset

""" 2. SVM ALGORITHM """
```

```
f_dataset = f_dataset.drop(columns=['Tyyppi', 'TilaKoodi', 'Valmistunut',
'Koulutuskoodi'])

# Remove outliers using z score (3 sigma's)
z_scores = stats.zscore(f_dataset)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
f_dataset = f_dataset[filtered_entries]

# Data normalization
np_x = f_dataset.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(np_x)
normdata = pd.DataFrame(x_scaled)
normdata.columns = f_dataset.columns
f_dataset = normdata
del normdata

# Split the dataset (train & test set)
X = f_dataset.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 10]].values
y = f_dataset.iloc[:, 9].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state = 7)

# Compute SVM and predict
classifier = SVC(kernel = 'linear', random_state = 7)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Model accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f%%" % (accuracy * 100.0))

""" 3. VISUALIZATION """
# Get support vectors themselves
support_vectors = classifier.support_vectors_

# Visualize support vectors
plt.scatter(X_train[:,4], X_train[:,3])
plt.scatter(support_vectors[:,0], support_vectors[:,1], color='red')
plt.title('Linearly separable data with support vectors')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()

""" 4. SAVE THE MODEL FOR DEPLOYMENT """
pickle.dump(classifier, open("student-SVM.pkl", "wb"))
pickle.dump(min_max_scaler, open("student-SVM-Scaler.pkl", "wb"))
```

XGBoost algorithm (XGBoost.py)

```
# -*- coding: utf-8 -*-
"""
Machine learning algorithm for VIRTA data
```

```
@author: Ari Sivula

XGBoost algorithm for predicting student performance.

Developed with Python version 3.8.5

"""

import numpy
import pandas as pd
import numpy as np
import pickle
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from scipy import stats
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

""" 1. READ THE DATASET AND PRE-PROCESSING THE DATA """
dataset = pd.read_csv('students-Features.csv')

# Categories
dataset['Koulutusmoduulitunniste'] = dataset["Koulutusmoduulitunniste"].as-
type('category')
dataset['Koulutusmoduulitunniste_cat'] = dataset['Koulutusmoduulitunnis-
te'].cat.codes

# Drop not required columns and rows
dataset = dataset[dataset['semester-ECTS'].notna()]
dataset = dataset[dataset['total-GPA'].notna()]
dataset = dataset[dataset['LukukausiCount'].notna()]
dataset = dataset[dataset['Opt_ECTSCount'].notna()]
dataset = dataset.drop(columns=['Opintosuoritukset', 'KirjoihintuloPvm',
'AlkuPvm', 'LoppuPvm', 'Koulutusmoduulitunniste', 'Koulutuskieli', 'Luokit-
telu'])
dataset = dataset.reset_index()
dataset = dataset.drop(columns=['index', 'studentID'])

# Missing data imputation
values = dataset.values
imputer = SimpleImputer(missing_values=np.NaN, strategy='mean')
f_dataset = pd.DataFrame(imputer.fit_transform(values))
f_dataset.columns = dataset.columns
del dataset

""" 2. XGBOOST ALGORITHM """
f_dataset = f_dataset.drop(columns=['Tyyppi', 'TilaKoodi', 'Valmistunut',
'Koulutuskoodi'])

# Remove outliers using z score (3 sigma's)
z_scores = stats.zscore(f_dataset)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
f_dataset = f_dataset[filtered_entries]

# Split the dataset (train & test set)
X = f_dataset.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 10]].values
y = f_dataset.iloc[:, 9].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state = 7)

# XGBoost
model = XGBClassifier(booster='gblinear')
model.fit(X_train, y_train)

# Predictions for test data
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]

# Evaluate predictions
accuracy = accuracy_score(np.array(y_test, dtype=int), np.array(predictions,
dtype=int))
print("Accuracy: %.2f%%" % (accuracy * 100.0))

""" 3. SAVE THE MODEL FOR DEPLOYMENT """
pickle.dump(model, open("student-XGBoost.pkl", "wb")) # Save
```