FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 45 (2020) No. 4
ISSN 0867-6356
DOI: 10.2478/fcds-2020-0015 e-ISSN 2300-3405

sciendo

Return on Investment in Machine Learning: Crossing the Chasm between Academia and Business

Jan Mizgajski, Adrian Szymczak [*], Mikołaj Morzy [†],
Łukasz Augustyniak, Piotr Szymański [‡], Piotr Żelasko [§]

Abstract. Academia remains the central place of machine learning education. While academic culture is the predominant factor influencing the way we teach machine learning to students, many practitioners question this culture, claiming the lack of alignment between academic and business environments. Drawing on professional experiences from both sides of the chasm, we describe the main points of contention, in the hope that it will help better align academic syllabi with the expectations towards future machine learning practitioners. We also provide recommendations for teaching of the applied aspects of machine learning.

Keywords: applied machine learning, machine learning teaching, machine learning engineering

1. Introduction

Machine learning (ML) is quickly becoming one of the defining technologies of modern computer science (CS). It attracts attention from both academia and business, with some going as far as calling it the source of the growth of future economy [89]. However, the deployment of scalable and efficient ML solutions remains a very challenging task due to numerous practical obstacles. In the opinion of some practitioners, one of the consequential obstacles is the lack of alignment between the way ML is presented and taught in academia, and the industrial requirements [106, 107].

The main goal of this paper is to identify the divergences between academic and applied ML. Obviously, the paper is heavily opinionated as we draw from our experi-

[*] Avaya, {mizgajski.jan,adrian.dominik.szymczak}@gmail.com
[†] Poznan University of Technology, mikolaj.morzy@put.poznan.pl
[‡] Wroclaw University of Science and Technology, {lukasz.augustyniak, piotr.szymanski}@pwr.edu.pl
[§] The Johns Hopkins University, piotr.andrzej.zelasko@gmail.com

ences resulting from the involvement in scientific research, educational activities, and working both in startups and corporate environments. Even if some of our claims seem controversial, we hope that they provide ML educators with inspiration and business perspective. At this point we also want to stress that our paper is addressed to teachers and students of ML engineering specializations. We acknowledge the need to train ML researchers who are faced with a different set of challenges and requirements, so the curriculum of more research-oriented ML courses will be different.

There are consequential differences between the way ML is taught and the way ML is applied. We focus on enriching the academic curriculum with ML engineering best practices at the early stages of education. We address this paper primarily to students of ML, as well as to faculty members engaged in teaching ML engineering. We hope to enrich the academic perspective on applied ML, borrowing as the guiding principle the advice which opens the excellent Google's guide *Rules of Machine Learning* [119]:

> Do machine learning like the great engineer you are, not like the great machine learning expert you aren't.

We divide our discussion into four blocks covering the entire lifespan of an ML project. Each section concludes with recommendations of available tools and libraries that can be useful for teaching that particular aspect of ML deployment. In Section 2 we focus on the problem of business objectives. We discuss the importance of the proper definition of business objectives and their alignment with loss functions and evaluation metrics used during model design and training. Section 3 presents our perspective on the acquisition, transformation, and provenance of data used for training, testing, and validation. Section 4 is devoted to the problem of algorithm selection and training. Finally, in Section 5 we present specific recommendations regarding the orchestration of ML workflows, and best practices for the management of ML projects. The paper concludes in Section 6. The paper is an extended version of the publication submitted to the PP-RAI'2019 conference [83].

2. Business Objectives

The quality of an ML model is evaluated using a set of metrics. Examples include mean squared error for regression, accuracy for classification, mean average precision for ranking, silhouette score for clustering, and many others. Unfortunately, the reality is much more complex and requires a more holistic approach.

2.1. Machine learning metrics

A metric used for model evaluation is often not directly translatable into an actionable business metric, usually referred to as the key performance indicator (KPI). And the KPI is not synonymous with the business objective, which usually consists of multiple criteria and KPIs. Consider the problem of automatic speech recognition system (ASR) development. The most frequently used objective function for ASR training

is the word error rate (WER). From the business perspective, however, much more nuanced KPIs are important, such as verb confusion, keyword error rate, grammatical correctness (measured by the coherence of parse trees), intent recognition, etc. And these KPIs may not fully represent the major business objective, which may be defined as the overall text understanding or human impression of the quality of a transcript. Oftentimes, a business objective is to jointly optimize two metrics. In many natural language processing (NLP) tasks, the objective function is to optimize recall given a fixed minimum precision threshold, a goal which is relatively difficult to achieve. A typical example is the task of spam e-mail identification, where a model should maximize the recall (the number of identified spam messages), but only after securing a certain accuracy threshold (the probability that an identified spam message is indeed unsolicited). The definition of a good business objective that models the business case closely or evaluating the models using a Pareto frontier [52] in a multi-criteria setting (accuracy, latency, efficacy) should receive particular attention in the design of machine learning curricula.

2.2. Downstream tasks and constraints

One must not forget that there are essential upstream and downstream metrics which postpone the evaluation of an ML model [119]. Upstream metrics for model evaluation usually revolve around system infrastructure costs. For example, a simple lightweight model incurring minimum CPU usage and consuming less memory may be preferred to a more complex model consuming more memory and requiring more computing power, because it translates directly into monetary savings. A larger model which improves the WER of a speech recognition system by 1% adds almost nothing to product quality, but may significantly increase infrastructure costs. Downstream metrics postpone the evaluation of an ML model by measuring the quality of the model indirectly through the observation of signals produced by components which consume the results of the ML model. Consider the task of sentiment analysis applied to culinary recipe reviews. The metric used to evaluate the accuracy of the classifier (e.g., accuracy) is in no way connected to the metric used to train word embeddings for that particular domain.

In several applications, there is a strict response time limit imposed, depending on business requirements (e.g., a model must come up with a prediction within 30 ms). In various NLP tasks, using transformers such as `BERT` [48] or `Q8BERT` [116] might not be feasible due to long inference times and the size of the model itself. Especially in real-time systems using dictionary-based (static) embeddings, like `word2vec` [82], might be the only viable solution. In order to address certain issues, e.g., concept drift, models might have to be retrained periodically on recent data. Such a requirement makes the training speed and cost critical – together with the cost of building automated processes, data acquisition, human and finally compute resources.

2.3. Real business impact of ML models

In the production environment, the quality of an ML model should be measured from the perspective of how it influences the production system as a whole, and the key business objectives in particular. Finding a link between these two metrics is a significant effort without the guarantee of success. The main reason is that such a correspondence can be elusive, indirect, and convoluted. How does the precision of an ML model influence overall customer satisfaction? Multiple factors need to be taken into consideration and all should be measured. Factors such as user interface changes or small latency reduction might be the reason for a change in measured customer satisfaction, not the model's predictive power. The first step towards addressing this issue is to invest in monitoring and to implement a large variety of measurements to get a better understanding of how different elements of a system inter-operate and influence each other. Due to the fact that it is hard to attribute business objectives to individual metrics, as they are interwoven by complex relationships, the key outcome of monitoring is the recognition of the role ML models play in a larger system. If measured metrics do not translate into business objectives, either the scope of monitoring should expand, or reconsidering ML model training objectives might be necessary. Rule-based and simple interpretable models are valuable during the first iteration – they not only help to launch product or release features faster, but also provide ample opportunity to better understand the system and offer more business insights.

2.4. Recommendations

In our view, the academia does not pay enough attention to the practical ramifications of model training and evaluation. ML models do not operate in a vacuum, they constitute a broader ecosystem, and we should evaluate them in the context of business objectives they serve. This principle may be difficult to apply in an academic environment, but we need to continuously remind the students of the necessity to think critically about model training and evaluation. For instance, a mismatch between the training criterion (i.e. the loss function optimized during training) and real objectives has led YouTube to recommend content full of racism and conspiracy theories because it maximized the number of views [104]. Students should be taught to check for potential information leaks between training and evaluation sets, biases in the training datasets, and model over-fitting (collectively known as the training-serving bias). It is especially important in cases where transfer-learning is involved and models are expected to perform on data substantially different from the ones used in training. We also suggest that students are made aware of:

- possible adversarial effects of ML objective functions [78],
- model bias caused by objective functions [55, 86],
- real environmental costs of ML training [76],
- real-world examples of over-fitting of ML models [77],

3. Data

Thoroughly knowing one's data and domain is the key requirement for the successful application of ML to solve real-world problems. Out of the box models and algorithms usually require sophisticated domain adaptation, which cannot be performed without a deep understanding of data and its characteristics. As an example, we can imagine a keyword extraction problem for scientific papers (a standard problem in academia) and the application of its related research to extract key phrases from transcribed calls in a call center (a very domain-specific problem). On the surface, these may seem like identical problems and one may hope to get good results with off-the-shelf algorithms. In practice, we could learn that transcribed calls have high word error rate, fundamentally different sentence structure (due to turn-taking by different participants, disfluencies, back-channels), and different parts of speech distribution, as spoken language differs from written language significantly. Of course, gaining this domain knowledge is impossible in the classroom setting, but students should be fully aware of the challenge of thorough domain exploration.

3.1. Teaching data processing

Most machine learning models are data greedy. Many courses and textbooks stress the importance of data and feature engineering, but it is challenging to comprehensively present all the intricacies of the process in an academic setting. What we often see is that students learn about the essential feature engineering techniques (normalization, binning, feature extraction, feature encoding) and then proceed to learn about various algorithms. On the other hand, every practitioner will recognize that data preparation is the most crucial step in the entire machine learning pipeline [93]. In a practical setting, they will spend the majority of time on data and feature engineering, and it will be time well-spent. According to many sources [5, 6], the so-called 80/20 rule is applied to most of data scientists' work. It means that 80% of the data scientist's valuable time is spent simply on finding, cleansing, and organizing data, leaving only 20% to perform actual analysis.

Academic curricula usually enclose data processing in a separate course or specialization, focusing on data warehousing [72], extraction-transform-load pipelines [108], and data lakes [71]. Of course, this is understandable as the subject of data processing is in itself too large and by no means limited to machine learning. Our recommendation to ML teachers is to make sure that ML teaching is synchronized and integrated with data processing teaching. It may be very tempting to point the students to a dataset posted on Kaggle [17] and to proceed directly to ML model training. In reality, getting high-quality data is much harder and time-consuming than it might seem.

3.2. Data annotation

Leaving the data processing aside, we want to concentrate on two distinct concepts that the students of ML should be exposed to. Domain adaptation combined with data annotation is a very useful tool to obtain high quality data for model training, and it can be effectively presented in an academic setting. When no high-quality data is available, another possibility is to turn to a family of techniques commonly referred to as weakly supervised learning.

Data annotation and acquisition strategies can be a key business competitive advantage. Investments in building or customizing software and processes for data annotation can be pivotal to business' success. Unfortunately, practical concerns surrounding data annotation can be detrimental. The need for additional human resources or questions regarding data security and confidentiality are examples of such concerns which limit the wide use of data annotation. On the other hand, the inclusion of the ML team in the data annotation process may have many positive side effects. ML engineers not only produce more high quality data for model training, but they gain invaluable understanding of the application domain and business ramifications at the same time.

It is worth mentioning that advancing the project from prototype to production could require additional investment in dedicated staff for data annotation. While crowd-sourcing services [2, 11] can be a viable way to obtain annotated data for some problems, it is not a one-size-fits-all solution. Obtaining and ensuring annotation quality may prove prohibitively expensive and unreliable if the annotation task is cognitively complex or requires some domain expertise.

3.3. Weakly supervised learning

Weak supervision is another approach for the curation of data for model training [36, 46]. Weakly supervised learning is an umbrella term for a family of techniques for the mitigation of data scarcity. In general, weakly supervised learning can be categorized into:

- incomplete supervision: only a part of the dataset is labeled, and more labels need to be induced from existing labels based on certain assumptions about data structure (this type of weak supervision is usually referred to as semi-supervised learning [105]),
- inexact supervision: labels are not as exact or detailed as desired, they may be expressed in terms of higher-order concepts or generalizations [59],
- inaccurate supervision: labels may contain errors [64].

Many techniques have been proposed over the years to implement the paradigm of weakly supervised learning. Active learning [57, 97] uses a model-in-the-loop approach to human annotation by carefully selecting data items for human annotation.

Transfer learning [94] uses models trained on different tasks and datasets to solve other problems. This transfer of knowledge encoded in a model can be accompanied by either unsupervised domain adaptation [50], or by supervised fine-tuning [70]. Recently, a new paradigm of self-supervised learning has been quickly gaining momentum [51, 65]. The main premise of self-supervised learning is to extract a useful supervision signal from unlabeled data using an auxiliary task (also known as the pretext task). The design of the auxiliary task requires domain knowledge and a significant dose of engineering ingenuity, and can be a valuable teaching exercise.

3.4. Recommendations

We assume that the students are exposed to data gathering, cleaning, imputation, transformation, and integration techniques within the scope of relevant subjects in their curriculum. In this section we focus on data annotation and we provide recommendations on how to make working with data more efficient and attractive to students.

- **Annotation protocol**: Students should be familiar with the concept of the annotation protocol and how such protocol can be designed and implemented. Probably the best place to start is to work with text data. An excellent introduction to the annotation of text corpora can be found in [90]

- **Manual data annotation**: Affordable and effective tools for annotation are readily available and should be presented as the part of the ML course curriculum. Data annotation can be done directly from Jupyter Notebooks [16, 24], or tools like `Prodigy` [84] and `doccano` [85]. For crowd-sourcing platforms, such as Amazon Mechanical Turk [3] or Amazon Sagemaker Ground-Truth [2], a thorough comparison of platforms is available [34].

- **Programmatic data annotation**: Manual data annotation can be time-consuming and expensive, even if ML models are included in the annotation loop. Programmatic data annotation is an interesting alternative. Snorkel [91] is the tool which allows to encode basic labeling intuitions into a rule-based system that automatically annotates data. This is especially useful if one already has a small set of manually labeled data or a reasonable understanding of labeling rules from data exploration.

- **Pre-trained models**: Many pre-trained models have been published over the recent years `BERT` [48], `Glove` [88], `GPT-3` [43], `MobilesNet` [69], `VGG` [100], and many more. A promising direction is to use these models in new application domains via transfer learning [75, 112], especially when training of models from scratch is prohibitively expensive [26]. This practice has become *de facto* the baseline approach in computer vision and NLP tasks. Using pre-trained models is trivial since many of those models are available via standard APIs in libraries such as `SpaCy` [67], `Flair` [12], `HuggingFace` [15], or `PyTorch Vision` [87].

- **Augment data if possible**: Data augmentation can be used to enrich and diversify data available for model training. Data augmentation techniques, such as cropping, padding, and horizontal flipping, are commonly used to train large neural networks [66] in image recognition tasks, but data augmentation can apply to text and audio data as well. Words and sentences can be shuffled, words can be replaced by synonyms, tokens can be randomly inserted, swapped, or deleted, back translation can be used to circle a sentence between languages, sentences can be randomly modified by manipulating the syntax tree directly. For audio data most typical augmentations include noise injection, frequency shifting, and speed manipulation. Several augmentations are readily available in popular libraries (`TensorFlow.image`, `Keras`, `TorchVision.transforms`), and libraries dedicated to image augmentation, such as `Augmentator` [38] or `Albumentations` [44] are available. For text and audio data augmentation can be easily performed using `NLPAug` [80] library.

- **Be aware of ways to find open datasets**: There are numerous repositories with openly available datasets such as Kaggle Dataset [17], UCI Machine Learning Repository [30], Amazon Open Data [4], Google Dataset Search [14], Network Repository [22], Data Gov [8], and Data USA [9]. Open datasets may be especially valuable in tandem with proprietary licensed or internal datasets, as they could provide ample data for pre-training a model that can later be fine-tuned [70] with proprietary data. Furthermore, students should be aware of different license types and ways to understand their applicability for commercial use [29].

- **Use data versioning tools**: In an ideal world data do not evolve with time. However, the real-life projects have to face data streaming, concept drifts, degradation of data quality over time, etc. Students should know how to create data snapshots, or how to rollback data and models to a certain time point. Data Version Control (`dvc`) [10] and `kedro` [18] are examples of tools which allow for advanced data versioning.

- **Bad examples of ML models**: Discuss examples of biased models [39, 68, 103]. Students' understanding of concepts such as information leakage, concept drift, sampling error, survival bias, etc., may be significantly increased by showing real world examples of ML models gone bad [98].

4. Algorithms

A typical ML curriculum puts significant focus on detailed presentations of algorithms for classification, regression, and clustering. It is, of course, fundamental to understand the inner workings of an algorithm, to be able to adjust its hyper-parameters, and to interpret the results. However, focusing solely on these aspects may result in over-stressing the role of individual algorithms, which becomes detrimental to the application of ML to solve real-world problems.

4.1. ML baselines

In practice, starting every experiment with a simple baseline is not only a good strategy, but a necessity as well. General-purpose classification algorithms, such as Random Forests, Gradient Boosting, or Naive Bayes, can provide strong baselines in many classification tasks, whereas Agglomerative Hierarchical Clustering, k-means, or DBSCAN tend to perform well for unsupervised tasks. In the case of NLP tasks, pre-computed word embeddings, combined with simple tools (such as cosine distance for clustering, or Support Vector Machines for classification), tend to work remarkably well across a wide range of problems. Similarly for image recognition, stripping layers from large pre-computed models, such as `VGG19`, `ResNet`, or `InceptionV3`, is a very reasonable starting point.

4.2. Recommendations

Teaching and presenting ML algorithms is probably the most gratifying part of the ML curriculum. While not questioning the central point which ML algorithms occupy in the curriculum, below we present some recommendations for enriching this curriculum with applied ML concepts.

- **Always start with a baseline**: Whenever presenting a new algorithm, strive to compare its performance with a baseline. We advocate the use of Random Forests [42] as a universal baseline, but any simple ML algorithm will do.
- **Use benchmarks**: Many ML tasks have accompanying benchmarks that allow to extensively compare many algorithms. Examples include general benchmarks, such as the suite of clustering benchmarks [54], or task-dependent specialized benchmarks, like Chinese-to-English neural translation benchmark [61].
- **Visualize algorithms**: In our experience, visual tools which help to interpret and explain trained models are invaluable. Visualization can be easily introduced into the ML workflow using tools such as `Yellowbrick` [33] or `Visdom` [31]. Text datasets can be visualized using `Scattertext` [73]. Deep neural nets can be inspected using `TensorBoard` [27], `GanLab` [13], saliency maps [117], or Deep Visualization [115]. Even recurrent neural networks can be easily visualized to reveal activations and attention [101, 111].

5. Deployment

5.1. Methodologies for ML processes

Finally, we arrive at the most contentious point: limited teaching of ML productization in the academia. For obvious reasons, it is difficult to comprehensively present

this subject in the classroom setting, so the focus of the academic community was on the presentation of methodologies rather than practical tools. Over the years, several methodologies for knowledge discovery and machine learning processes have been proposed. Cross-Industry Standard Process for Data Mining (CRISP-DM) [113], Knowledge Discovery from Databases [53], Sampling, Exploring, Modifying, Modeling, and Assessing data (SEMMA) [81], Ontology Driven Knowledge Discovery Process [60], or Knowledge Discovery Life Cycle (KDLC) [79] come to mind. Unfortunately, these attempts to introduce rigid methodologies to ML workflows have never gained momentum and commercial recognition. Even the most popular among them, CRISP-DM, is no longer maintained, and its special interest group seems to have dissolved.

The widespread adoption of ML in practical tools and services resulted in the rekindling of research interests around ML processes and ML management. The complexity of ML-based systems makes their validation and verification a challenging task [63]. In recent years we see much more work, both practical and theoretical, allocated to designing robust and secure ML workflows, in particular, by adapting well-known practices and patterns of software engineering to ML systems. A good place to start researching software engineering methodologies for ML is a recent case study [35], where the authors review ML-based initiatives at Microsoft from the perspective of agile software engineering processes. The resulting nine-stage workflow presents best practices that address the challenges of deploying ML solutions. A similar best practices guide from Google has already been mentioned [119]. Another excellent resource of this type is a detailed description of best practices resulting from implementing over 150 machine learning models within `Booking.com` infrastructure [37]. The authors emphasize the importance of a good metric design which takes into consideration the business impact. The authors also note that there is practically no correlation between the offline performance gain of a model and the business value gain.

Interestingly, machine learning is not only benefiting from software engineering methods, but it changes software engineering at the same time. A recent study [110] presents the results of a large survey of software engineers on their experiences when developing ML-based products. The ability of a system to learn adds a layer of complexity and instability to the overall system design. This uncertainty is reflected in changes to the software development process. A survey [74] of almost 800 data scientists working on ML products in Microsoft reveals that the term *data scientist* encompasses at least nine distinct roles with disjoint characteristics and responsibilities within an ML project. This is just one out of many challenges of engineering ML solutions [40] and of teaching ML engineering.

5.2. Deployment of ML models

ML deployment is a very complex and iterative process, involving data acquisition, data labeling, model training and evaluation, KPI monitoring, model re-training, and more. This process is driven by engineering principles, rather than research guidelines. At the same time, it is not simply the adaptation of software engineering

best practices to the domain of ML. Practitioners of ML are fully aware how much more challenging and difficult is the deployment of ML-backed solutions as compared to more traditional information systems [96] due to:

- Entanglement: Input features are all interconnected, adding, removing or changing the distribution of a single feature can impact all other features. ML workflows are often cited as good examples of the CACE principle: Changing Anything Changes Everything.
- Unstable data dependencies: The relationship between input and output of an ML model can be unstable, a small change in the input signal can have a detrimental effect on the model which consumes the signal.
- Underutilized data dependencies: For various reasons ML workflows may keep input features which provide almost no value to the model. A legacy feature may have become obsolete during model training, but is still served as input, a feature may be bundled together with truly useful features, two features may be correlated but the model makes a mistake and prioritizes non-causal feature, etc.
- Direct and hidden feedback loops: A model can directly or indirectly influence the selection of input features, creating a feedback loop which is very difficult to analyze. Also, a model can implicitly select input features for another model, creating a hidden dependency.
- Dead experimental codepaths: Due to the way ML models are designed and developed, the codebase can contain hundreds of abandoned branches with data, models, and configurations. Such branches may be the source of integration errors.
- Process management debt: When the product becomes large and mature, the number of ML models contained in the product grows rapidly. Coordination of models, synchronized updating of configurations, management of resource allocation between models, monitoring of data flows in the production pipeline, all these tasks become increasingly complex and error-prone.

5.3. Testing of ML code

One of the most neglected areas of ML productization is testing. Despite years of developing best practices in software engineering testing, ML engineers exhibit a disquieting restraint when it comes to testing. In general, testing of machine learning pipelines is difficult as it involves multiple levels: data pipelines, feature extraction pipelines, model training and validation, model serving. Time dimension plays an important role – many regressions may be delayed as they accumulate over time in the training data. In order to fully comprehend the scope of this challenge, readers are invited to take a simple test [41]. The authors propose a set of tests for data.

They discuss testing of the data itself, but also testing ML infrastructure, model development, and methods of ML monitoring. There are also methodologies developed for particular branches of ML, such as methodologies dedicated to deep learning testing [47, 56, 114], testing for machine translation [102], or computing bounds on ML assurances from manifold learning [45]. A comprehensive survey of over 140 scientific papers on ML testing is available [118].

Teaching ML deployment in an academic setting is difficult. Nevertheless, students should be constantly reminded that ML projects are just like any other IT projects, only more so. Solid software engineering workflows and best practices are as relevant to ML projects as they are to regular projects.

5.4. Keep it simple

The phrase "better done than perfect" has profound implications in business, especially in the early stages of a project. Students should be encouraged to develop their solutions in an iterative way and not shy away from rules and heuristics [119]. A combination of rules, heuristics, and a simple predictive model often proves to be a formidable baseline. Such solutions are usually computationally efficient and may prove hard to beat by subsequent effort if contenders are evaluated holistically, taking into account their performance, efficacy, hidden costs, and maintainability.

Consider the use case of moderating online classified ads market to exclude ads which violate the terms of service. Initially, it may be tempting to assume that this binary classification task can be fully automated with ML. In practice, due to the concept drift introduced by the changing market, the ingenuity of spammers and scammers, or even changes in the terms of service itself, the problem is much better framed as an attempt to augment a rule-based system and a team of human moderators. By ranking ads according to the probability of acceptance (and automatically approving those with probability below the threshold that satisfies the false positive rate), prioritizing the moderation of less questionable ads, and explaining and highlighting the reason for a low score, moderators can make their decisions faster. With such definition, even a weak classifier could be successful in providing a true business impact. Careful analysis of such use cases with students, even if only theoretical, can dramatically improve students' understanding of the entire ML workflow and its business ramifications. While designing projects for students, we encourage seeking problems that enable or require problem re-framing and simplification, or simulate evolving and flexible requirements.

The main incentive in academic research is being published in a prestigious journal, which enforces certain constraints on the minimum scope of the experiment and introduces a strong bias towards positive results. In business, however, we are interested in the most cost-effective way of validating or invalidating a hypothesis. The main focus in applied ML is on fast pruning of unpromising paths. Students should be encouraged to design quick validation experiments for guidance. In addition, students should quickly learn not to get attached to their ideas, but to embrace failure and learn. Many ideas and approaches do not work or do not yield sufficient improvements

in practice, so being able to let go of a failed idea and move to the next experiment is essential in applied research. Failures are not synonymous with wasted resources, but trying to squeeze microscopic improvements over baseline solutions erratically is.

5.5. Market perspective

Last but not least, academia rarely teaches students to think of ML models through the prism of the final product. This perspective requires a careful consideration of several factors that may or may not fall outside of the area of expertise of machine learning researchers and engineers. Apart from customer research, user experience design, profitability, marketing, and the expertise of ML practitioners can be beneficial for the development of the final product. There are several possible strategies that a product can employ to gain popularity: being first to the market, being cheaper than the competition, being better than the competition (claimed with data), being a fast follower with good marketing strategy, or maybe just being good enough to carve up a wide niche. Students should be aware of the strategy the product is employing and should evaluate their progress and objectives in the context of that strategy. The final goal is not to be perfect, but to adequately solve a business need.

5.6. Recommendations

We understand that the presentation of the full deployment of an ML model may be difficult in the classroom setting. Also, with the rapid update of the ML toolbox the curriculum focused on a particular set of tools and libraries faces the danger of quickly becoming obsolete. Thus, in this section we have gathered general recommendations regarding the tools that may be presented in the context of ML deployment.

1. **Use agile processes all the way**: Run ML projects using a flexible methodology (e.g. Kanban or Scrum), with short daily re-caps, planning, definitions of done, retrospectives, and frequent syncing between team members to brainstorm and validate ideas. Pair programming can help with idea generation, training, and reducing errors and failed assumptions. Machine learning tasks are complex and time-consuming, so sync often to help each other avoid dead ends and flaws, especially in the early phase. We recommend to structure ML assignments in a way that they require a group effort and a prolonged collaboration between students, possibly spanning over different courses.

2. **Do code reviews**: This is one of the best and quickest ways to teach high-quality coding to students. It promotes the best programming practices and supports horizontal knowledge transfer between students. Try to define a set of practices that students will enforce during the review, e.g., verifying if all tests have passed in the automatic build, keeping folder structure and naming conventions, and checking if all artefacts needed to reproduce a model are available and properly described.

3. **Test (almost) everything**: Seldom do we see ML code presented to the students accompanied by unit tests, integration tests, and automatic (or at least reproducible) builds. At the same time, releasing an untested code to a production environment is a recipe for disaster. ML code should be tested at least as thoroughly as the rest of the codebase. Testing of ML code is not simple, it depends on train/test/validation datasets, which may not be time-invariant. But the most difficult obstacle is that ML tests should verify not only the structure of the code, but the logic learned during training [92].

4. **Build a "second brain"**: Often, it is more important to know what did not work and what was considered, but rejected, and why. Everything should be documented: rationales, tools, assumptions, solutions, outcomes, requirements. The term "documentation" is used here loosely as a set of semi-structured notes available for the ML team for reference. Notion [23] is a perfect tool to build such "second brain" for the project.

5. **Outsource model serving complexity**: There are many services that may provide stable and robust execution and monitoring environments for ML deployment. Consider including tutorials on how to deploy models to TensorFlow Serving [28], Weights & Biases [32], Amazon SageMaker [25] or Neptune [21]. Most of these environments offer free academic licences and sandbox versions suitable for classroom use.

6. **Avoid over-engineering**: Discourage students from premature optimization and imprint in them the key rule of extreme programming: *you ain't gonna need it* (YAGNI). ML workflows tend to become convoluted quite quickly, so do not add any unnecessary complexity. Highlight to students that complex pipelines with many features, even if they deliver better predictive results, may be less preferable in real-world scenarios due to maintainability and availability considerations [119].

7. **Structuring data and model repositories**: Organizing projects around a set of commonly followed conventions is a frequent pattern in many areas of computer science. ML codebase can quickly deteriorate into a labyrinth of directories filled with exotically named files. Students will benefit greatly from an early exposure to ML engineering conventions related to project's structure. QuantumBlack's `kedro` [18] and `cookiecutter` [7] are two excellent tools to organize the structure of the codebase in a logical and interpretable way.

8. **Use orchestration tools**: A comprehensive ML teaching curriculum contains all steps of the ML life-cycle. Exposing students to platforms for ML life-cycle management completes the curriculum and allows the students to see the "big picture": how all the steps taught during the course come together. We strongly suggest to include in the curriculum platform such as `MLflow` [20], `MetaFlow` [19], or `Algorithmia` [1] to demonstrate experimentation, reproducibility, and deployment of ML models.

6. Conclusions

Teaching ML engineering in a comprehensive way is a very challenging task, as ML life-cycle consists of multiple interwoven steps requiring different techniques, tools, and libraries. The basic recipe may sound simple: create a bulletproof processing pipeline, design a reasonable objective, and keep adding features, making sure at each step that the processing pipeline is unbroken. In practice, however, the landscape of available tools, platforms, services, and libraries is vast and quickly changing. A solid ML engineering curriculum should not be detached from the currently available toolset, but it should emphasize the role of engineering methodologies beyond novelty tool fads.

6.1. Bigger picture

And there, of course, remains the most fundamental question: why are we doing machine learning? Aren't we engaging in ML for ML's sake only? We all strive to accomplish something impactful and meaningful with our work, so it would be wise to focus attention on machine learning that matters [109] and measuring the true impact of ML solutions within the broader societal context. Students should be encouraged to focus on meaningful evaluation methods (i.e., instead of measuring accuracy, precision, AUROC, etc., measure the change in the real world), to engage in the outside world (define real problems, collect real data, design real evaluations), and to keep their eyes on the price, which is choosing research problems based on their potential impact. A comprehensive list of such research challenges related to the imminent climate crisis can be found in [95]. Machine learning provides ample opportunities for researchers to have a real impact on the world and to advance social good [62]. For a copious compendium of possible ML applications advancing social good, please refer to [99]. We should strive to design ML solutions in such a way that they promote and foster human development, economical growth, ethical justice, and ecological sustainability. We have to take responsibility for the societal impacts of ML [49]. Academia has a unique opportunity to fulfill this goal by shaping not only the professional profiles of the alumni, but by planting the seeds of social responsibility in them.

6.2. Business pressure on academia

In this paper, we have outlined issues which hinder the quick adaptation of students of machine learning who leave academia for business environments and we have provided some recommendations that could help address this problem. We are not discussing the question of the relevance of basic research in machine learning to the actual needs of the market, nor are we criticizing the way ML research is conducted by academia. Far from it, we focus exclusively on the problem of teaching ML in a way which enables students to quickly become proficient in translating their skills into

functioning products. Academia remains the main source of human talent for applied ML, and the constant drain of top ML talent by large companies exerts constant pressure on ML teaching resources. A recent study [58] reveals an exponential growth of the number of tenure-track and tenured faculty leaving universities for the industry. This "corporate poaching" negatively impacts the overall economy by stifling the ML entrepreneurship among the alumni, weakening the ML brainpower available to government and universities, and lowering the rate of innovation and disruptive creativity. We hope that our recommendations can be useful for those who chose to remain in academia thus reducing the negative impact of this unmet demand for machine learning talent. We also hope that our remarks regarding the social responsibility aspect of the ML education will inspire educators to take an active role in shaping the societal awareness of their students. Most students of ML will leave the university to find a job in the industry, thus a practical alignment of their skills and mindset with the challenges they will encounter should be beneficial for both students and the economy. Those who stay in academia will hopefully have a stronger background to participate in product-oriented grants, industry collaborations, and open source projects. We believe that a better alignment of academia and business practices in ML is possible and we hope to strengthen the feedback loop between both environments.

Acknowledgements

The authors want to express their gratitude to Avaya for financial support and providing an environment for applied machine learning research. The work has been also supported by the statutory funds of the Faculty of Computing and Telecommunications of the Poznan University of Technology and the statutory funds of the Faculty of Computer Science and Management of the Wrocław University of Science and Technology.

References

[1] Algorithmia. `algorithmia.com/product`. Accessed: 2020-11-30.

[2] Amazon ground truth. `https://amzn.to/3g0AGqf`. Accessed: 2020-11-30.

[3] Amazon mechanical turk. `www.mturk.com`. Accessed: 2020-11-30.

[4] Amazon open data. `registry.opendata.aws`. Accessed: 2020-11-30.

[5] Breaking the 80/20 rule: How data catalogs transform data scientists' productivity. `ibm.co/2QhuGxK`. Accessed: 2020-11-30.

[6] Cleaning big data: Most time-consuming, least enjoyable data science task, survey says. `bit.ly/3d3M0zQ`. Accessed: 2020-11-30.

[7] Cookiecutter data science. https://bit.ly/3msmp8j. Accessed: 2020-11-30.

[8] Data gov. www.data.gov. Accessed: 2020-11-30.

[9] Data usa. datausa.io. Accessed: 2020-11-30.

[10] Data version control. dvc.org. Accessed: 2020-11-30.

[11] Figure eight. www.figure-eight.com. Accessed: 2020-11-30.

[12] Flair. github.com/flairNLP/flair. Accessed: 2020-11-30.

[13] Ganlab. poloclub.github.io/ganlab. Accessed: 2020-11-30.

[14] Google dataset search. https://bit.ly/2JucsbU. Accessed: 2020-11-30.

[15] Huggingface. https://bit.ly/39vUDE4. Accessed: 2020-11-30.

[16] Jupyanno. github.com/chestrays/jupyanno. Accessed: 2020-11-30.

[17] Kaggle datasets. www.kaggle.com/datasets. Accessed: 2020-11-30.

[18] Kedro. github.com/quantumblacklabs/kedro. Accessed: 2020-11-30.

[19] Metaflow. metaflow.org. Accessed: 2020-11-30.

[20] Mlflow. mlflow.org. Accessed: 2020-11-30.

[21] Neptune. neptune.ml. Accessed: 2020-11-30.

[22] Network repository. networkrepository.com. Accessed: 2020-11-30.

[23] Notion: All-in-one workplace. ww.notion.so. Accessed: 2020-11-30.

[24] Pigeon. github.com/agermanidis/pigeon. Accessed: 2020-11-30.

[25] Sagemaker. aws.amazon.com/sagemaker. Accessed: 2020-11-30.

[26] The staggering cost of training sota ai models. bit.ly/39O20nL. Accessed: 2020-11-30.

[27] Tensorboard. www.tensorflow.org/tensorboard. Accessed: 2020-11-30.

[28] Tf serving. github.com/tensorflow/serving. Accessed: 2020-11-30.

[29] Tldrlegal - software licenses explained in plain english. tldrlegal.com/. Accessed: 2020-11-30.

[30] Uci ml repository. archive.ics.uci.edu/ml. Accessed: 2020-11-30.

[31] Visdom. github.com/facebookresearch/visdom. Accessed: 2020-11-30.

[32] Weights & biases. www.wandb.com. Accessed: 2020-11-30.

[33] Yellowbrick. www.scikit-yb.org. Accessed: 2020-11-30.

[34] Fair crowd work: Shedding light on the real work of crowd-, platform-, and app-based work. `http://faircrowd.work/platform-reviews`, 2018. Accessed: 2020-11-30.

[35] Amershi S., Begel A., Bird C., DeLine R., Gall H., Kamar E., Nagappan N., Nushi B., and Zimmermann T. Software engineering for machine learning: A case study. In *41st IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice*, pages 291–300. IEEE, 2019.

[36] Badene S., Thompson K., Lorré J.-P., and Asher N. Weak supervision for learning discourse structure. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2296–2305, 2019.

[37] Bernardi L., Mavridis T., and Estevez P. 150 successful machine learning models: 6 lessons learned at booking.com. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1743–1751, 2019.

[38] Bloice M. D., Roth P. M., and Holzinger A. Biomedical image augmentation using augmentor. *Bioinformatics*, 35(21):4522–4524, 2019.

[39] Bolukbasi T., Chang K.-W., Zou J., Saligrama V., and Kalai A. Man is to computer programmer as woman is to homemaker? debiasing word embeddings, 2016.

[40] Bosch J., Crnkovic I., and Olsson H. H. Engineering ai systems: A research agenda, 2020.

[41] Breck E., Cai S., Nielsen E., Salib M., and Sculley D. What's your ml test score? a rubric for ml production systems. In *Reliable Machine Learning in the Wild - NIPS 2016 Workshop*, 2016.

[42] Breiman L. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[43] Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., and Amodei D. Language models are few-shot learners, 2020.

[44] Buslaev A., Iglovikov V. I., Khvedchenya E., Parinov A., Druzhinin M., and Kalinin A. A. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[45] Byun T. and Rayadurgam S. Manifold for machine learning assurance. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 06 2020.

[46] Dehghani M., Severyn A., Rothe S., and Kamps J. Learning to learn from weak supervision by full supervision, 2017.

[47] Demir S., Eniser H. F., and Sen A. Deepsmartfuzzer: Reward guided test generation for deep learning, 2019.

[48] Devlin J., Chang M.-W., Lee K., and Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[49] Dignum V. Responsible artificial intelligence: designing ai for human values. *The ITU Journal on Future and Evolving Technologies*, 2017.

[50] Dingwall N. and Potts C. Mittens: An extension of glove for learning domain-specialized representations, 2018.

[51] Doersch C. and Zisserman A. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017.

[52] Drozdowski M., Kowalski D., Mizgajski J., Mokwa D., and Pawlak G. Mind the gap: a heuristic study of subway tours. *Journal of Heuristics*, 20(5):561–587, 2014.

[53] Fayyad U., Piatetsky-Shapiro G., and Smyth P. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996.

[54] Gagolewski M. et al. Benchmark suite for clustering algorithms – version 1, 2020.

[55] Garcia M. Racist in the machine: The disturbing implications of algorithmic bias. *World Policy Journal*, 33(4):111–117, 2016.

[56] Gerasimou S., Eniser H. F., Sen A., and Cakan A. Importance-driven deep learning system testing, 2020.

[57] Gilyazev R. and Turdakov D. Y. Active learning and crowdsourcing: A survey of optimization methods for data labeling. *Programming and Computer Software*, 44(6):476–491, 2018.

[58] Gofman M. and Jin Z. Artificial intelligence, human capital, and innovation. *Human Capital, and Innovation (August 20, 2019)*, 2019.

[59] Gong C., Zhang H., Yang J., and Tao D. Learning with inadequate and incorrect supervision. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 889–894. IEEE, 2017.

[60] Gottgtroy P. Ontology driven knowledge discovery process: a proposal to integrate ontology engineering and kdd. *PACIS 2007 Proceedings*, page 88, 2007.

[61] Hadiwinoto C. and Ng H. T. Upping the ante: Towards a better benchmark for chinese-to-english machine translation, 2018.

[62] Hager G. D., Drobnis A., Fang F., Ghani R., Greenwald A., Lyons T., Parkes D. C., Schultz J., Saria S., Smith S. F., and Tambe M. Artificial intelligence for social good, 2019.

[63] Hand D. J. and Khan S. Validating and verifying ai systems. *Patterns*, 1(3):100037, 2020.

[64] Hao D., Zhang L., Sumkin J., Mohamed A., and Wu S. Inaccurate labels in weakly supervised deep learning: Automatic identification and correction and their impact on classification performance. *IEEE Journal of Biomedical and Health Informatics*, 2020.

[65] Hendrycks D., Mazeika M., Kadavath S., and Song D. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, pages 15663–15674, 2019.

[66] Ho D., Liang E., Chen X., Stoica I., and Abbeel P. Population based augmentation: Efficient learning of augmentation policy schedules. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2731–2741, Long Beach, California, USA, 06 2019. PMLR.

[67] Honnibal M. and Montani I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[68] Howard A. and Borenstein J. The ugly truth about ourselves and our robot creations: The problem of bias and social inequity. *Science and engineering ethics*, 24(5):1521–1536, 2018.

[69] Howard A. G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., and Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[70] Howard J. and Ruder S. Universal language model fine-tuning for text classification, 2018.

[71] Inmon B. *Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump*. Technics Publications, 2016.

[72] Inmon W. H. *Building the data warehouse*. John Wiley & Sons, 2005.

[73] Kessler J. S. Scattertext: a browser-based tool for visualizing how corpora differ, 2017.

[74] Kim M., Zimmermann T., DeLine R., and Begel A. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, 44(11):1024–1038, 2017.

[75] Krizhevsky A., Sutskever I., and Hinton G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[76] Lacoste A., Luccioni A., Schmidt V., and Dandres T. Quantifying the carbon emissions of machine learning, 2019.

[77] Lazer D., Kennedy R., King G., and Vespignani A. The parable of google flu: traps in big data analysis. *Science*, 343(6176):1203–1205, 2014.

[78] Ledwich M. and Zaitsev A. Algorithmic extremism: Examining youtube's rabbit hole of radicalization, 2019.

[79] Lee S. W. and Kerschberg L. A methodology and life cycle model for data mining and knowledge discovery in precision agriculture. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2882–2887, 1998.

[80] Ma E. NLP augmentation. `github.com/makcedward/nlpaug`, 2019.

[81] Matignon R. *Data mining using SAS enterprise miner*, volume 638. John Wiley & Sons, 2007.

[82] Mikolov T., Chen K., Corrado G., and Dean J. Efficient estimation of word representations in vector space, 2013.

[83] Mizgajski J., Szymczak A., Żelasko P., Morzy M., Augustyniak Ł., and Szymański P. Return of investment in machine learning: Crossing the chasm between academia and business. In *Proceedings of the PP-RAI'2019 Conference: Polskie Porozumienie na Rzecz Rozwoju Sztucznej Inteligencji*, pages 285–291. Wroclaw University of Science and Technology, 2019.

[84] Montani I. and Honnibal M. Prodigy: A new annotation tool for radically efficient machine teaching. `prodi.gy/`, 2018. Accessed: 2020-11-30.

[85] Nakayama H., Kubo T., Kamura J., Taniguchi Y., and Liang X. Doccano: Text annotation tool for human. `github.com/doccano/doccano`, 2018. Accessed: 2020-11-30.

[86] Obermeyer Z. and Mullainathan S. Dissecting racial bias in an algorithm that guides health decisions for 70 million people. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 89–89, 2019.

[87] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[88] Pennington J., Socher R., and Manning C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[89] Purdy M. and Daugherty P. Why artificial intelligence is the future of growth, 2016.

[90] Pustejovsky J. and Stubbs A. *Natural Language Annotation for Machine Learning: A Guide to Corpus-Building for Applications*. O'Reilly Media, 2012.

[91] Ratner A., Bach S. H., Ehrenberg H., Fries J., Wu S., and Ré C. Snorkel: rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282, 11 2017.

[92] Ribeiro M. T., Wu T., Guestrin C., and Singh S. Beyond accuracy: Behavioral testing of nlp models with checklist, 2020.

[93] Roh Y., Heo G., and Whang S. E. A survey on data collection for machine learning: a big data – ai integration perspective, 2019.

[94] Ruder S., Peters M. E., Swayamdipta S., and Wolf T. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, 2019.

[95] Schwartz R., Dodge J., Smith N. A., and Etzioni O. Green ai, 2019.

[96] Sculley D., Holt G., Golovin D., Davydov E., Phillips T., Ebner D., Chaudhary V., Young M., Crespo J.-F., and Dennison D. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, pages 2503–2511, 2015.

[97] Settles B. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[98] Shane J. *You Look Like a Thing and I Love You: How Artificial Intelligence Works and why It's Making the World a Weirder Place*. Voracious, 2019.

[99] Shi Z. R., Wang C., and Fang F. Artificial intelligence for social good: A survey, 2020.

[100] Simonyan K. and Zisserman A. Very deep convolutional networks for large-scale image recognition, 2015.

[101] Strobelt H., Gehrmann S., Behrisch M., Perer A., Pfister H., and Rush A. M. Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):353–363, 2018.

[102] Sun Z., Zhang J. M., Harman M., Papadakis M., and Zhang L. Automatic testing and improvement of machine translation, 2019.

[103] Tatman R. Gender and dialect bias in youtube's automatic captions. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 53–59, 2017.

[104] Tufekci Z. Youtube's recommendation algorithm has a dark side. `bit.ly/2m09tvZ`. Accessed: 2020-11-30.

[105] Van Engelen J. E. and Hoos H. H. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[106] VanderPlas J. The big data brain drain: Why science is in trouble. Accessed: 2020-11-30.

[107] VanderPlas J. Hacking academia: Data science and the university. Accessed: 2020-11-30.

[108] Vassiliadis P. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining*, 5(3):1–27, 2009.

[109] Wagstaff K. Machine learning that matters, 2012.

[110] Wan Z., Xia X., Lo D., and Murphy G. C. How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, pages 1–14, 2019.

[111] Wang J., Gou L., Shen H.-W., and Yang H. Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):288–298, 2018.

[112] Weiss K., Khoshgoftaar T. M., and Wang D. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[113] Wirth R. and Hipp J. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39. Springer-Verlag, 2000.

[114] Yan M., Wang L., and Fei A. Artdl: Adaptive random testing for deep learning systems. *IEEE Access*, 2019.

[115] Yosinski J., Clune J., Nguyen A., Fuchs T., and Lipson H. Understanding neural networks through deep visualization, 2015.

[116] Zafrir O., Boudoukh G., Izsak P., and Wasserblat M. Q8bert: Quantized 8bit bert, 2019.

[117] Zeiler M. D. and Fergus R. Visualizing and understanding convolutional networks. In *13th European Conference on Computer Vision, ECCV 2014*, pages 818–833. Springer Verlag, 2014.

[118] Zhang J. M., Harman M., Ma L., and Liu Y. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[119] Zinkevich M. Rules of machine learning: Best practices for ml engineering. `developers.google.com/machine-learning/guides/rules-of-ml`. Accessed: 2020-11-30.

Received 16.03.2020, Accepted: 30.11.2020