14

Automating Data Science

Summary. It has been observed that, in data science, a great part of the effort usually goes into various preparatory steps that precede model-building. The aim of this chapter is to focus on some of these steps. A comprehensive description of a given task to be resolved is usually supplied by the domain expert. Techniques exist that can process natural language description to obtain task descriptors (e.g., keywords), determine the task type, the domain, and the goals. This in turn can be used to search for the required domain-specific knowledge appropriate for the given task. In some situations, the data required may not be available and a plan needs to be elaborated regarding how to get it. Although not much research has been done in this area so far, we expect that progress will be made in the future. In contrast to this, the area of preprocessing and transformation has been explored by various researchers. Methods exist for selection of instances and/or elimination of outliers, discretization and other kinds of transformations. This area is sometimes referred to as *data wrangling*. These transformations can be learned by exploiting existing machine learning techniques (e.g., learning by demonstration). The final part of this chapter discusses decisions regarding the appropriate level of detail (granularity) to be used in a given task. Although it is foreseeable that further progress could be made in this area, more work is needed to determine how to do this effectively.

14.1 Introduction

It is well known in data science that a greater part of the effort goes into various preparatory operations, while the model-building step typically requires less effort. This has motivated researchers to examine how to automate the steps that precede the model building. In this chapter our aim is to analyze these preparatory steps. They include:

1. Defining the current problem/task
2. Identifying the appropriate domain-specific knowledge
3. Obtaining the data
4. Data preprocessing and other transformations
5. Automating model generation and its deployment
6. Automating report generation

Some steps (e.g., step 5) were discussed in detail in some of the previous chapters. A brief overview is given in Section 14.6. Step 4 will be discussed further on.

P. Brazdil et al., *Metalearning*, Cognitive Technologies,
https://doi.org/10.1007/978-3-030-67024-5_14

As for steps 1, 2, and 3, these are normally carried out by data science specialists and many people believe that it is difficult to automate them. This may be the reason why these steps were not included among the topics of various recent workshops on the topic of automating data science.[1] Although automation is difficult at this stage, it is nevertheless interesting to consider how this could be done and as a result provide assistance to data scientists.

Step 1, the issue of defining problems (or tasks), is discussed in Section 14.2. It is foreseeable that, as more experience is gathered from many different domains, some common patterns could be identified which in turn could provide a basis for at least partial automation. Section 14.3 discusses step 2: the problem of identifying useful domain-specific knowledge that could be included in the training data or the learning process itself. Section 14.4 is dedicated to step 3, discussing various strategies used for obtaining the training data. As pointed out in that section, it may happen that the data may not already be available in some application areas, and so a process must be devised regarding how to overcome this.

Section 14.5 discusses data preprocessing. This includes many types of transformations that are already well covered by AutoML methods, such as feature selection, but also *data wrangling*, which is not so easily automated but which has attracted a great deal of attention recently. Evidence of this are various workshops (e.g., AutoDS 2019 (Bie et al., 2019)) and research projects (e.g., AIDA (The Alan Turing Institute, 2020)). This section discusses also *data aggregation operations*, which are essential in many data mining applications.

Our motivation was to advance the overall understanding of this area and to share this with others. We hope that, this way, we can contribute to the development of future decision support systems in this area or even partial automation of the processes involved.

14.2 Defining the Current Problem/Task

The formal definition of a given problem (task) involves a number of steps which can be summarized as follows:

- Problem understanding and description
- Generating task descriptors
- Determining the task type and goals
- Identifying the task domain

Each step is discussed in detail in the respective subsection.

14.2.1 Problem understanding and description

Here we will consider two main types of problems: business problems and scientific problems. Regarding business problems, the initial part of the KDD process is usually identified as *business problem formulation*. We assume that this is done by a human and is formulated in natural language. For instance, if the aim is to obtain an understanding of certain activities of some institution (e.g., company or hospital), we need to define which aspects are of interest. These may include for instance:

[1]See, e.g., ADS 2019 workshop (Bie et al., 2019).

- Financial aspects (e.g. net profits or production costs)
- Proportion (or frequency) of successful cases
- Proportion (or frequency) of problematic cases (e.g., unsuccessful car models/branches that had to be closed, mortality cases in a hospital, etc.), among other aspects

This information helps to obtain a more detailed understanding of the domain and the data, enabling to focus on the appropriate data mining task.

14.2.2 Generating task descriptors

The task description is often done in natural language. It needs to be processed in order to extract a set of task descriptors (keywords). Let us see a few examples.

- Suppose the aim was to develop a system for credit rating and we were given its description. The description would be processed to retrieve a set of descriptors that would characterize the entity that has requested the credit (e.g., medium-sized company), in how many years the loan would be paid off, existing guarantees, purpose (e.g., amplifying installations or buying machinery), history of payments, possible spread, etc.
- If the aim were to develop a method for navigating a robot from one position to another, rather different descriptors would be required. Typically they would need to characterize thr robot's initial position, direction (angle), speed, acceleration, and similar descriptors for the final set of attributes. Similarly, the descriptors could characterize the method for planning the trajectory and the avoidance strategy.
- If the aim were to provide control of a patient in an intensive unit, rather different descriptors would be required. These could include, for instance, the patient's heart rate, blood pressure, level of potassium, etc. and the need to maintain these within safe limits, which could also be specified.

This strategy was exploited, for instance, by Contreras-Ochando et al. (2019), who used certain domain-specific metafeatures to determine the type of task to carry out (e.g., certain metafeatures could indicate that the task involves transforming a date into a normalized format).

Whenever possible, the descriptors used should belong to a given set of common vocabulary/ontology terms, as this facilitates further processing.

14.2.3 Determining the task type and goals

Particular business or scientific objectives captured by appropriate descriptors determine, to a large extent, what kind of top-level task should be considered. If the aim was to simply obtain understanding of a particular domain, an unsupervised task (e.g., clustering) could be a good choice. If the aim is to predict values of certain variables (objective variables), the most appropriate top-level task could be classification or regression.

Determining the task type from a set of descriptors (keywords) can be formulated as a meta-level classification problem.

We note that a complex task may include various subtasks. The top-level task of a certain type (e.g., classification) may involve subtasks of a different type (e.g., regression).

Role of learning goals

Determining which concepts (descriptors) should be brought into play is influenced by the learning goal. This issue was noted by the Russian psychologist Wygotski (1962). He drew attention to the fact that concepts arise and get developed if there is a specific need for them. Acquisition of concepts is thus a purposeful activity directed towards reaching a specific goal or a solution of a specific task.

This problem has been noted also by people in AI and ML. Various researchers, including Hunter and Ram (1992b,a), Michalski (1994), and Ram and Leake (2005), have argued that it is important to define explicit goals that guide learning. Learning is seen as a search through a knowledge space guided by the learning goal. Learning goals determine which part(s) of prior knowledge are relevant, what knowledge is to be acquired and in what form, how it is to be evaluated, and when to stop learning. The importance of planning in this process has also been identified by Hunter and Ram (1995).

Providing a schedule for learning goals

As some more complex tasks are formulated in terms of multiple goals, it may be desirable to define also an ordering in which (some of) the goals should be achieved. This problem can be seen as the problem of *learning multiple interdependent concepts*. In effect, defining the ordering can be regarded as defining the appropriate procedural bias, as the ordering determines how the hypothesis space should be searched.

14.3 Identifying the Task Domain and Knowledge

The description of a given task using descriptors can be used to determine the domain to which that task belongs. Determining the domain is important, as it facilitates determining what type of data is required to solve the task.

One way to address this is to activate the appropriate domain-specific descriptors (ontology).

This process can be seen as the problem of activating the appropriate domain together with the corresponding set of domain-specific descriptors (ontology). This process can be compared to the process of activating Minsky's frames (Minsky, 1975). However, when the proposal on frames was written, the area of machine learning was not yet very advanced and so the mechanism used that would invoke frames was not well clarified at the time.

Basically, we see two different mechanisms that can be used for this aim. One involves keyword matching, and the other one classification. Both are described in the following subsections.

Identifying the domain by matching descriptors/metafeatures

Both the task and the domain can be described using descriptors (keywords possibly organized in ontologies) or metafeatures. The domain can then be identified by matching the task and domain descriptors. The aim is to identify the domain with the closest match. It is necessary to define the appropriate measure of similarity between the task descriptors and particular domain descriptors. Naturally, one would seek the domain which is most similar to the given task.

Identifying the domain by classification

Determining the domain can be regarded as a meta-level classification task. The input is a particular set of task descriptors of the given task, and the output is a particular domain characterized by a set of domain descriptors. As domains could be organized in a hierarchy, classification can be done at various levels of this hierarchy. For instance, the task descriptors could suggest that the given task pertains to *medicine* at a higher level and to *obstetrics* at a lower level.

Representation of data and goals

It is important to have an appropriate representation for both data and the goals. Various schemes have been proposed in the past. For instance, Stepp and Michalski (1983) proposed *goal dependency networks* (GDN). In this chapter we have discussed the role of descriptors that help to determine the task type and the appropriate data. Some of these can be identified as descriptors of learning goals. So, for instance, *company profit* can be seen as a descriptor of data, if we are talking about a particular data item stored in a database. If, on the other hand, the aim is to predict its value from other information, then it becomes a learning goal.

14.4 Obtaining the Data

After the domain descriptors have been identified, it is necessary to identify and obtain the actual data. Typically, the aim is to identify a part of the data which is relevant for the task at hand, as this facilitates further processing. If the data were held in a relational database, then it would be necessary to identify the relevant part, i.e., a subset of possibly interconnected tables. Each table has a heading, and the terms used in each column often suggest the type of information contained there. However, sometimes the terms may be abbreviations or codes attributed by the designer. Hence, the problem of matching task descriptors to data descriptors would be facilitated if each name in the table heading were accompanied by the correct ontological descriptor(s).

14.4.1 Select existing data or plan how to obtain data?

The next important question is whether the data is already available or not. If it is available, it is possible to advance. If the data is not available or insufficient for the current task, new data needs to be gathered (e.g., by interacting with the "outside world" or by running experiments). The latter strategy was used in a robot scientist called *Adam* (King et al., 2009). This system autonomously generates hypotheses concerning functional genomics of a particular type of yeast (*Saccharomyces cerevisiae*) and then conducts tests to experimentally verify them.

14.4.2 Identifying the domain-specific data and background knowledge

Identifying the correct part of the data is not a trivial matter. Consider, for instance, the task of predicting whether to concede credit to a given customer. If the data did not include the relevant part (e.g., tables and the corresponding features), the learning system would be unable to generate the correct inductive hypothesis. In this case, the search space does not include the inductive hypotheses that we would like the system to come up with.

If, on the other hand, the data unnecessarily included too many items, some of which might be irrelevant, the system might not arrive at the right hypothesis within a given time budget. Although in this case the search space would include the right inductive hypotheses, the system may have difficulty discovering them.

In inductive logic programming (ILP) the data is represented in the form of facts. As Srinivasan et al. (1996) have shown, the learning problem can often be defined using a set of constraints. One important one is $H \wedge Bk \models E+$, stating that the inductive hypothesis H together with the background knowledge Bk should logically imply the set of positive examples $E+$.[2] Srinivasan et al. (1996) observed that the performance of the learning system is sensitive to the type and amount of background knowledge. In particular, background knowledge that contains information irrelevant to the task considered can make the search for the correct hypothesis more difficult.

So, in general, if our aim was to automate this process, we would need to identify not only the relevant domain-specific data but also the relevant part of background knowledge. More details on some related issues are given in the following subsections.

14.4.3 Obtaining the data and background knowledge from different sources

As Contreras-Ochando et al. (2019) have pointed out, data science must integrate data from very different data sources. These may include databases, repositories, the web, spreadsheets, text documents, and images, among others. The integration may involve homogeneous sources (e.g., two different databases of similar type, but with different content), or heterogeneous sources (e.g. text and image).

Obtain data by accessing the OLAP data cube

One useful concept in this area is the so-called *OLAP data cube*, which is simply a multidimensional array of data (Gray et al., 2002). The operation *slice* permits to select a rectangular subset of a cube by choosing a single value for one of its dimensions. The operation *dice* is similar, but allows us to set specific values of multiple dimensions.

Several other operations are discussed in Section 14.5.4, which focuses on rather specific transformations whose aim is to alter the granularity of data.

14.5 Automating Data Preprocessing and Transformation

Data preprocessing and transformation can be regarded as one of many steps in the target workflow (pipeline). We will be discussing various preprocessing operations that are useful in classification pipelines. Usually, the selection of a particular preprocessing operation cannot be done in isolation from other elements in the pipeline. For instance, discretization needs to be carried out only if the particular classifier (e.g., naive Bayes) requires this.

[2]Srinivasan et al. (1996) refer to this constrain as *strong sufficiency*.

Different methods that can be used to design such a pipeline in a systematic fashion were discussed in detail in Chapter 7 and hence will not be reviewed here again. Our aim here is to complement this discussion by describing studies whose aim is to further automate the process, including tasks such as data wrangling.

As Chu et al. (2016) pointed out, data preprocessing serves either to repair data (e.g., based on some quality rules) or to transform data such that it leads to better results in further analysis (e.g., when applying a machine learning algorithm). Here, we consider the following operations:

- Data transformation/data wrangling
- Instance selection/cleaning/outlier elimination
- Data preprocessing

Data transformation/data wrangling is discussed in Subsection 14.5.1. Instance selection is discussed in Subsection 14.5.2. Data preprocessing includes various operations, such as:

- Feature selection
- Discretization
- Nominal to binary transformation
- Normalization/standardization
- Missing value imputation
- Feature generation (e.g., PCA or embeddings)

Detailed descriptions of these operatiopn are discussed in various textbooks (see, e.g., Dasu and Johnson (2003)). Subsection 14.5.3 discusses ways of automatizing the selection of the appropriate preprocessing operation.

Although full automation of data science has been an ultimate goal of many research studies, aiming for this has shown to be computationally expensive, mainly due to the search space involved. In addition, some decisions may require domain knowledge (e.g., is a certain outlier safe to remove or not?), or there may not exist a ground truth to learn from. In Chapter 7, we discussed systems that go a long way towards automating pipeline design, such as Auto-sklearn (Feurer et al., 2015, 2019). This system includes many preprocessing operations (e.g., normalization, missing value imputation, and feature selection). However, even this state-of-the-art system is currently not capable of automating the full data science pipeline. Still, such systems can serve as a useful starting point for further developments.

The aim of this section is to cover other, quite different approaches that typically aim to automate or semi-automate a specific preprocessing technique (e.g., outlier detection). Even though these approaches are not always easy to include in AutoML systems, they do show how to solve sub-problems in isolation, and still help in building larger, efficient systems in the future.

14.5.1 Data transformation/data wrangling

Data wrangling is the process of transforming and mapping data from one data representation format (e.g., “raw” data) to another. The objective of this transformation is to make it appropriate for subsequent processing. So, for instance, the transformation may convert information in a spreadsheet into a tabular dataset.

Inferring data types

This process often requires inferring the *data types* (e.g., date, float, integer, and string) of certain dataset entities (e.g., features represented in columns). Valera and Ghahramani (2017) proposed a Bayesian method that is able to detect such types based on the distribution of the data in a certain region. However, this does not work optimally if there are missing data and anomalies in the data. System *ptype* by Ceritli et al. (2020) is a more recent method that employs a simpler probabilistic model that detects such anomalies and robustly infers data types.

Some wrangling approaches

Some systems automatically infer relevant transformations, learned either from a demonstration provided by the user or from a given set of examples. However, as providing examples by the user incurs costs, ideally this is done on the basis of a limited number of examples. This is normally achieved by imposing a strong bias on the actual task (e.g., assumptions about the output format in a particular domain), enabling the system to prioritize some transformations over others.

This was exploited in one early system in this area – *Trifacta Wrangler* (Kandel et al., 2011) – which generates a ranked list of possible transformations. These transformations can be seen as simple programs, which was explored by various researchers already in the 1970s and 1980s (e.g., Mitchell (1977); Mitchell et al. (1983); Brazdil (1981), among others).

System FOOFAH

Jin et al. (2017) developed a technique to synthesize data transformation programs from example transformations, described using input–output example pairs. Their system, FOOFAH, searches the space of possible data transformation operations to generate a program that will perform the desired transformation. In general, the aim is to transform a poorly structured grid of values (e.g., a spreadsheet table) into a relational table that can be used by ML programs. Such a transformation can be a combination of operators. Some operators are oriented towards columns, and others to rows or even to the whole table (e.g. transpose). Some column operators are shown below:

- Drop: deletes a column in the table
- Move: relocates a column from one position to another
- Merge: concatenates two columns and appends the merged column to the end
- Split: separates a column into two or more parts at the occurrences of the delimiter
- Divide: divides one column into two columns based on some predicate
- Extract: extracts the first match of a given regular expression in each cell of a designated column

The authors show that their system obtains the required programs faster than Wrangler discussed earlier.

Usage of ILP in wrangling approaches

Some researchers followed the approach of inductive programming (IP) or inductive logic programming (ILP) (Gulwani et al., 2015), which led to a wider spread. Microsoft, for instance, has decided to include a data wrangling tool *Flashfill* (Gulwani et al., 2012) in Excel.

The ILP approaches seem particularly useful to the task of transformation, as it is possible to constrain the search by specifying the appropriate domain-specific background knowledge (BK). This was exploited for instance by Contreras-Ochando et al. (2019), who used certain domain-specific metafeatures to determine the type of task to carry out, which in turn brought into play the appropriate background knowledge. The problem tackled by these authors involved recognizing the type of a given *named entity*, which may be a date, email, name, phone number, or some other entity, together with the coding convention that depends on the respective country (e.g. France, UK, etc.). This determines which part of the BK should be activated. In one task, for instance, the goal was to extract the day of the month from inputs in different formats. For instance, “25-03-74” should give 25 and “03/29/86” should return 29.

Systems TDE and SYNTH

TDE (Transform Data by Example) (He et al., 2018) works in a similar way, but it uses a library of program snippets that represent different transformations. The given examples are used to select the ones that generate the correct output.

SYNTH (De Raedt et al., 2018) can learn to carry out automatic completion of data in a set of worksheets. As the authors have shown, solving this problem requires a number of different steps. First, it is necessary to discover equations and/or obtain one or more predictive models that permit to calculate the values in one cell using the values in other cells. Secondly, it is necessary to apply the equations and/or induced models to fill in empty cells, parts of rows or columns. To address these different tasks, the system uses several different components:

- An automatic data wrangling system (Synth-a-Sizer) that transforms a dataset into a traditional attribute-value format so that standard machine learning systems could be applied
- System *Mercs* that induces predictive models (Van Wolputte et al., 2018)
- System *TacLe* that induces constraints and formulas in spreadsheets
- A component that ties learning and inference together

14.5.2 Instance selection and model compression

Some systems select a subset of data to be cleaned prior to passing them to further processing. *ActiveClean* (Krishnan et al., 2016), for instance, identifies the instances (records) that are more likely to affect the results of subsequent modeling.

The area of outlier detection/elimination is related to this. Many well-known methods exist, such as *isolation forests* (Liu et al., 2012), *one-class SVMs*, or *local outlier factors* (Breunig et al., 2000), among others. Typically these methods do not work well in unsupervised mixed-type scenarios. Eduardo et al. (2020) uses *variational autoencoders* (VAEs) to detect and repair outliers even in this setting.

One application of this is *model compression* (or *model distillation*), where a simpler and hopefully better model is obtained by selecting the training data. One pioneering work in this area was the work of John (1995) on so-called *robust decision trees*, who studied the effects of label noise. After learning a tree, all misclassified instances were removed from the learning set and a new tree was learned. Although this process did not result in an accuracy increase, the resulting tree was much smaller. Similar approaches for k-NN were presented by Tomek (1976) and Wilson and Martinez (2000). Further work on filtering misclassified instances was reported by Smith and Martinez (2018).

14.5.3 Automating the selection of the preprocessing method

Bilalli et al. (2018, 2019) predicted which preprocessing techniques are appropriate for a given classification algorithm. The preprocessing methods include, for instance, discretization, transformation from nominal to binary, normalization and standardization of values, replacement of missing values, and introduction of principal components. For some of these operations, both supervised and unsupervised variants have been considered. Besides, some of the operations can be applied to just one attribute, while others are global and can thus be applied to all attributes. They build a meta-model in the form of random forest for each target classification algorithm and use it together with the metafeatures of the target dataset to predict which preprocessing technique should be included in the pipeline. The associated probability of classification can be used to order these operations in terms of their expected utility. The resulting system, called PERSISTANT, thus recommends which preprocessors to consider given a certain classification algorithm, but not which classifier should be used. Similarly, Schoenfeld et al. (2018) build meta-models predicting when a preprocessing algorithm is likely to improve the accuracy or runtime of a particular classifier.

Other systems extend this idea to *sequences* of preprocessing steps (sub-pipelines). Learn2Clean (Berti-Equille, 2019) attempts to select the optimal sequence of tasks for preprocessing and takes into account the given dataset with the aim of maximizing the performance of the complete workflow. Similarly, in DPD (Quemy, 2019), the sequential model-based optimization (SMBO) technique is used to automatically select and tune preprocessing operators to improve the performance score within a restricted time budget. The authors have found that, for certain NLP preprocessing operators, specific configurations are optimal for several different algorithms.

Finally, some systems produce rankings of promising workflows (pipelines) that can be used to speed up the search. Cachada et al. (2017) and Abdulrahman et al. (2018) describe a method of recommending the most suitable workflow for a target dataset. Unlike PERSISTANT, it is capable of recommending a workflow, which may include preprocessing (CFS feature selection) followed by a classifier with a certain configuration of hyperparameter settings. The recommendations are in the form of a ranking of the most useful workflows. The system can then use this ranking to conduct experiments to identify the workflow with the best performance.

14.5.4 Changing the granularity of representation

Finally, one of the most low-level tasks that could potentially be automated is how to extract data from a database so that it can be used for learning.

Generating aggregate data from an OLAP cube

As we have mentioned earlier, an *OLAP data cube* is a multi-dimensional array of data (Gray et al., 2002). The operation *drill down/up* allows us to navigate in the data cube, going from the most summarized (up) to the most detailed (down). A *roll-up* involves summarizing the data along a particular dimension. The summarization rule might be an aggregate function, such as computing totals, for one or more of the dimensions, depending on the required level of detail. Common aggregate functions are SUM, AVG, COUNT, MAX, and MIN, among others. The intermediate data structures, sometimes referred to as *cuboids*, can be organized in a lattice. So, the roll-up operation can be seen as moving up in the lattice of cuboids.

Various studies have been carried out in the past. One integrates features obtained as a result of aggregation into the data mining process. For instance, Charnay (2016) used such features in relational decision trees and random forests.

Features that result from aggregation operations are commonly used in many different systems. Let us consider one example here from robotic soccer. As Stone (2000) pointed out, we may want to identify a receiver that can safely receive the ball. One way of finding this is by establishing the distances to surrounding opponent players and identifying the nearest one. The nearest distance represents an aggregate feature.

Some data mining systems, including for instance RapidMiner, provide an aggregate operator, which is well integrated with the rest of the data mining operations (Hofmann and Klinkenberg, 2013).

All legal operations define a search space, which may be too large to search through manually. There are many possibilities regarding which tables may be selected and/or joined. Further choice arises with respect to the selection of columns or aggregation operations that can be applied to them. Hence, there is a need for a kind of automatic data selection that would be able to support this process by exploring this space and providing good recommendation(s) to the user.

The topic of changing the granularity of representation is continued in Chapter 15. However, the focus there is different from this section, as it discusses the problem of changing granularity by introducing new concepts.

14.6 Automating Model and Report Generation

14.6.1 Automating model generation and deployment

The aim of this step is to a search for the best workflow (pipeline) for the given task. This can be done with a suitable AutoML/meta-level system.

AutoML systems conduct a search for the best workflow. The search may involve experiments on the target dataset with the objective of determining how different candidate workflows (and differently configured variants) perform. Chapter 6 provides more details about the techniques involved and about some systems that exist in this area.

Metalearning systems can be used, provided similar problems have been dealt with in the past and hence a corresponding meta-database exists. Chapters 2, 5, and 7 provide more details. Another possible strategy is to adapt an existing model using knowledge transfer techniques. More details on this can be found in Chapter 12.

14.6.2 Automating report generation

One good example in this area is the Automated Statistician (Steinruecken et al., 2019). This project aims to automate various phases of data science, including automated construction of models from data, comparison of different models, and automatic elaboration of reports with minimal human intervention. The reports include, besides basic graphs and statistics, human-readable descriptions in natural language.

References

Abdulrahman, S. M., Cachada, M. V., and Brazdil, P. (2018). Impact of feature selection on average ranking method via metalearning. In *European Congress on Computational Methods in Applied Sciences and Engineering, 6th ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing (VipIMAGE 2017)*, pages 1091–1101. Springer.

Berti-Equille, L. (2019). Learn2clean: Optimizing the sequence of tasks for web data preparation. In *The World Wide Web Conference*, page 2580–2586. ACM, NY, USA.

Bie, D., De Raedt, L., and Hernandez-Orallo, J., editors (2019). *ECMLP-KDD Workshop on Automating Data Science (ADS), Würzburg, Germany*. https://sites.google.com/view/autods.

Bilalli, B., Abelló, A., Aluja-Banet, T., and Wrembel, R. (2018). Intelligent assistance for data pre-processing. *Computer Standards & Interf.*, 57:101–109.

Bilalli, B., Abelló, A., Aluja-Banet, T., and Wrembel, R. (2019). PRESISTANT: Learning based assistant for data pre-processing. *Data & Knowledge Engineering*, 123.

Brazdil, P. (1981). *Model of Error Detection and Correction*. PhD thesis, University of Edinburgh.

Breunig, M., Kriegel, H.-P., Ng, R., and Sander, J. (2000). LOF: Identifying density-based local outliers. In *Proceedings of the MOD 2000*. ACM.

Cachada, M., Abdulrahman, S., and Brazdil, P. (2017). Combining feature and algorithm hyperparameter selection using some metalearning methods. In *Proc. of Workshop AutoML 2017, CEUR Proceedings Vol-1998*, pages 75–87.

Ceritli, T., Williams, C. K., and Geddes, J. (2020). ptype: probabilistic type inference. *Data Mining and Knowledge Discovery*, pages 1–35.

Charnay, C. (2016). *Enhancing supervised learning with complex aggregate features and context sensitivity*. PhD thesis, Université de Strasbourg, Artificial Intelligence.

Chu, X., Ilyas, I., Krishnan, S., and Wang, J. (2016). Data cleaning: Overview and emerging challenges. In *Proceedings of the International Conference on Management of Data, SIGMOD '16*, page 2201–2206.

Contreras-Ochando, L., Ferri, C., Hernández-Orallo, J., Martínez-Plumed, F., Ramírez-Quintana, M. J., and Katayama, S. (2019). Automated data transformation with inductive programming and dynamic background knowledge. In *Proceedings of ECML PKDD 2019 Conference*.

Dasu, T. and Johnson, T. (2003). *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons, Inc., NY, USA.

De Raedt, L., Blockeel, H., Kolb, S., Kolb, S., and Verbruggen, G. (2018). Elements of an automatic data scientist. In *Proc. of the Advances in Intelligent Data Analysis XVII, IDA 2018*, volume 11191 of *LNCS*. Springer.

Eduardo, S., Nazábal, A., Williams, C. K., and Sutton, C. (2020). Robust variational autoencoders for outlier detection and repair of mixed-type data. In *International Conference on Artificial Intelligence and Statistics*, pages 4056–4066.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, NIPS'15, pages 2962–2970. Curran Associates, Inc.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. (2019). Auto-sklearn: Efficient and robust automated machine learning. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *Automated Machine Learning: Methods, Systems, Challenges*, pages 113–134. Springer.

Gray, J., Bosworth, A., Layman, A., and Pirahesh, H. (2002). Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In *Proceedings of the International Conference on Data Engineering (ICDE)*, page 152–159.

Gulwani, S., Harris, W., and Singh, R. (2012). Spreadsheet data manipulation using examples. *Communications of the ACM*, 55(8):97–105.

Gulwani, S., Hernandez-Orallo, J., Kitzelmann, E., Muggleton, S., Schmid, U., and Zorn, B. (2015). Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99.

He, Y., Chu, X., Ganjam, K., Zheng, Y., Narasayya, V., and Chaudhuri, S. (2018). Transform-data-by-example (TDE): an extensible search engine for data transformations. In *Proceedings of the VLDB Endowment*, pages 1165–1177.

Hofmann, M. and Klinkenberg, R. (2013). *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Data Mining and Knowledge Discovery. Chapman & Hall/CRC.

Hunter, L. and Ram, A. (1992a). Goals for learning and understanding. *Applied Intelligence*, 2(1):47–73.

Hunter, L. and Ram, A. (1992b). The use of explicit goals for knowledge to guide inference and learning. In *Proceedings of the Eighth International Workshop on Machine Learning (ML'91)*, pages 265–269, San Mateo, CA, USA. Morgan Kaufmann.

Hunter, L. and Ram, A. (1995). Planning to learn. In Ram, A. and Leake, D. B., editors, *Goal-Driven Learning*. MIT Press.

Jin, Z., Anderson, M. R., Cafarella, M., and Jagadish, H. V. (2017). Foofah: A programming-by-example system for synthesizing data transformation programs. In *Proc. of the International Conference on Management of Data, SIGMOD '17*, page 1607–1610.

John, G. H. (1995). Robust decision trees: Removing outliers from databases. In *Knowledge Discovery and Data Mining*, pages 174–179. AAAI Press.

Kandel, S., Paepcke, A., Hellerstein, J., and Heer, J. (2011). Wrangler: Interactive visual specification of data transformation scripts. In *CHI '11, Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, page 3363–3372.

King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., Sparkes, A., Whelan, K. E., and Clare, A. (2009). The automation of science. *Science*, 324(5923):85–89.

Krishnan, S., Wang, J., Wu, E., Franklin, M., and Goldberg, K. (2016). ActiveClean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12):948–959.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1):3:1–3:39.

Michalski, R. (1994). Inferential theory of learning: Developing foundations for multistrategy learning. In Michalski, R. and Tecuci, G., editors, *Machine Learning: A Multistrategy Approach, Volume IV*, chapter 1, pages 3–62. Morgan Kaufmann.

Minsky, M. (1975). A framework for representing knowledge. In Winston, P. H., editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill.

Mitchell, T. (1977). *Version spaces: A candidate elimination approach to rule learning*. PhD thesis, Electrical Engineering Department, Stanford University.

Mitchell, T., Utgoff, P., and Banerji, R. (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning. Symbolic Computation*, pages 163–190. Tioga.

Quemy, A. (2019). Data pipeline selection and optimization. In *Proc. of the Int. Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, DOLAP '19*.

Ram, A. and Leake, D. B., editors (2005). *Goal Driven Learning*. MIT Press.

Schoenfeld, B., Giraud-Carrier, C., M. Poggeman, J. C., and Seppi, K. (2018). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Workshop AutoML 2018 @ ICML/IJCAI-ECAI*. Available at site https://sites.google.com/site/automl2018icml/accepted-papers.

Smith, M. R. and Martinez, T. R. (2018). The robustness of majority voting compared to filtering misclassified instances in supervised classification tasks. *Artif. Intell. Rev.*, 49(1):105–130.

Srinivasan, A., King, R. D., and Muggleton, S. H. (1996). The role of background knowledge: using a problem from chemistry to examine the performance of an ILP program.

Steinruecken, C., Smith, E., Janz, D., Lloyd, J., and Ghahramani, Z. (2019). The Automatic Statistician. In Hutter, F., Kotthoff, L., and Vanschoren, J., editors, *Automated Machine Learning*, Series on Challenges in Machine Learning. Springer.

Stepp, R. S. and Michalski, R. S. (1983). How to structure structured objects. In *Proceedings of the International Workshop on Machine Learning*, Urbana, IL, USA.

Stone, P. (2000). *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press.

The Alan Turing Institute (2020). *Artificial intelligence for data analytics (AIDA)*. https://www.turing.ac.uk/research/research-projects/artificial-intelligence-data-analytics-aida.

Tomek, I. (1976). An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:448–452.

Valera, I. and Ghahramani, Z. (2017). Automatic discovery of the statistical types of variables in a dataset. volume 70 of *Proceedings of Machine Learning Research*, pages 3521–3529, International Convention Centre, Sydney, Australia. PMLR.

Van Wolputte, E., Korneva, E., and Blockeel, H. (2018). Mercs: multi-directional ensembles of regression and classification trees. *https://www. aaai. org/ocs/index. php/AAAI/AAAI18/paper/view/16875/16735*, pages 4276–4283.

Wilson, D. and Martinez, T. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286.

Wygotski, L. S. (1962). *Thought and Language*. MIT Press.