Survey and Analysis of Current End-user Data Analytics Tool Support

Hourieh Khalajzadeh[1], Mohamed Abdelrazek[2], John Grundy[1], John Hosking[3], Qiang He[4]
[1]Faculty of Information Technology, Monash University, Australia
[2]School of Information Technology, Deakin University, Australia
[3]Faculty of Science, University of Auckland, New Zealand
[4]School of Software and Electrical Engineering, Swinburne University of Technology, Australia
{hourieh.khalajzadeh, john.grundy}@monash.edu, mohamed.abdelrazek@deakin.edu.au,
j.hosking@auckland.ac.nz, qhe@swin.edu.au

Abstract— There has been a very large growth in interest in big data analytics to discover patterns and insights. A major challenge in this domain is the need to combine domain knowledge – what the data means (semantics) and what it is used for – with advanced data analytics and visualization techniques to mine and communicate important information from the huge volumes of raw data. Many data analytics tools have been developed for both research and practice to assist in specifying, integrating and deploying data analytics applications. However, delivering such big data analytics applications requires a capable team with different skillsets including data scientists, software engineers and domain experts. Such teams and skillsets usually take a long time to build and have high running costs. An alternative is to provide domain experts and data scientists – the end users – with tools they can use to create and deploy complex data analytics application solutions directly with less technical skills required. In this paper we present a survey and analysis of several current research and practice approaches to supporting data analytics for end-users, identifying key strengths, weaknesses and opportunities for future research.

Index Terms— big data, data analytics, data science, data visualization, domain specific visual languages, machine learning

1 Introduction

USING "big data" to improve decision-making has become a highly active research and practice area [2, 3]. In the domain of data analytics, data scientists construct models to predict outcomes or discover underlying patterns, aiming to gain insights. Organizations can then use these insights to take actions that ideally improve future outcomes [4]. Gartner's technical professional advice [5] recommended six stages for developing machine learning applications: classifying the problem, acquiring data, processing data, modeling the problem, validation and execution, and finally deploying. *Classifying the problem* describes how to categorize the problem or the research question to solve – objectives, success criteria, assess constraints, etc. *Acquiring data* identifies where to find the data to support the problem. *Processing data* involves how to prepare data for further analytics tasks. The steps in this stage include data transformation, normalization and cleansing, as well as selection of training and test sets for supervised learning. *Modeling the problem* determines the range of machine learning algorithms to be used given the category of the problem and the type and volume of the available datasets. *Validation and execution* involve validating the results, determining the platform on which to execute the models and algorithm, and tuning and refining results. Finally, *deploying* the output of the machine learning process provides business value. This phase entails determining where and how to deploy and use the results.

Traditionally one needs advanced machine learning knowledge and skills with complex data science toolsets to be able to do such work. Emerging analytics approaches seek to automate many of the steps in model building and its application, making machine learning technology more accessible to those who lack deep quantitative analysis and tool building skills [4]. To be effective, such big data analytics software systems need to support: diverse data ingestion, wrangling and cleansing; data integration and querying for very large data volumes; feature extraction and selection; tailoring and combination of diverse data analytics techniques, software and services; and communication of findings and integration with existing IT solutions. This becomes more complicated when addressing quality of service attributes including: scalability, privacy, security, reliability and adaptability to changes in the target environment.

Recently, a number of data analytics and machine learning tools have become popular, providing packaged data sourcing, integration, analysis and visualization toolkits

- *H. Khalajzadeh and J. Grundy are with the Faculty of Information Technology, Monash University, Clayton, VIC, Australia 3800. E-mail: {hourieh.khalajzadeh, john.grundy}@monash.edu.*
- *M. Abdelrazek is with the School of Information Technology, Deakin University, Burwood, VIC, Australia 3125. E-mail: mohamed.abdelrazek@deakin.edu.au.*
- *J. Hosking is with the Faculty of Science, University of Auckland, Auckland, New Zealand 1142. E-mail: j.hosking@auckland.ac.nz.*
- *Q. He is with the School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, VIC, Australia 3122. E-mail: qhe@swin.edu.au*

oriented towards end users such as Azure ML Studio [6], Amazon AWS ML [7], Google Cloud ML [8], and BigML [9]. Many of these tools do not require programming language knowledge and are based on using simple drag and drop interfaces. However, while end user configuration is desirable, these tools suffer from a lack of extensibility, scalability and ability to integrate diverse third-party solutions and a capacity to handle complex problems. Many focus on machine learning algorithm wiring (implementation) and sometimes one-click deployment, but lack domain knowledge and business problem capture, modeling, traceability to the solution and validation of the solution against the problem. They also lack explanation of the technical model; i.e., from an end-user perspective, whether this is the best model they can obtain given their dataset and more importantly why they are obtaining a certain answer when they feed in an example input. Tools filling these gaps would be very useful for end-users and data scientists. In the discovery and exploration phase, being able to model the problem, extract insights/patterns, and develop predictive and clustering models, would greatly enhance the data analytics process before they need to involve software engineers.

This paper surveys and analyses the pros and cons of many existing tools in this domain and identifies gaps and requirements for better end user usage of such tools. This work is an extension of our previous survey [10] with new tools, tool evaluation and analysis. Our key findings include:

- Many existing tools focus on the machine learning modeling and implementation aspects of the software development lifecycle of AI-powered systems.
- Many are complicated for a domain expert with no data science and programming background to conduct initial exploration or discovery of the problem.
- Few are designed to enable collaboration between software engineers, data scientists and domain experts involved in the development of the systems.

In Section 2, motivation and background of this research are described using two examples. Current end user tools are analyzed and compared in Section 3 while their strengths and weaknesses are discussed in Section 4. Future research directions are presented in Section 5. Finally, conclusions are made in Section 6.

2 Motivation and Background

In this Section, we present two different representative use-cases of big data analytics applicationsm, one applying data analytics for property price prediction, and the other analyzing medical data for stroke prediction. These illustrate common big data analytics application issues, and different datasets and target end users we want to better support in creating such solutions.

2.1 Property Price Prediction Problem

Consider a property price prediction problem based on the solution developed in Azure ML Studio, as shown in Figure 1. In this scenario, a real estate agency wants to improve its agents' focus and outcomes by looking for patterns in a large amount of real estate, government and financial data. The company employed a technical team of software engineers and data scientists to work on the problem. In this figure, (a) the user drags the "Real Estate Sales Price" module to upload input data, (b) "Select Columns in Dataset", "Edit Metadata", "Clean Missing Data", and "Filter Based Feature Selection" modules to prepare and clean data, (c) "Linear Regression" and "Train Model" modules to apply linear regression and train the model, and finally (d) "Cross Validate Model" module to validate the model. To choose the modules and change the properties, the user needs to have data science knowledge. If the user now wants to use a preprocessing method or apply a model not in the list of modules, knowledge of programming languages such as Python and R is required to embed code that adds features.

However, the company realized that the development team lacked understanding of very important domain knowledge. The company appointed a senior real estate agent, highly experienced with the nuances in the domain, to help the team build the analytics solution. The team struggled with a lack of a common dialect between engineers, scientists and domain experts. Eventually, the team realized that the solution was not ready due to issues in the available dataset that needed to be rectified. It took a long time for the team to design and build a working analytics solution. A few months later, after the team had been disbanded, the company wanted to add new features e.g. to build a new model to predict who would be willing to sell their property in a given suburb. In addition, the company started to notice degradation in the performance of the existing property price prediction model, which needed to be updated. The company recruited a new team that struggled to develop models for the new capability, due to lack of business knowledge, and to update the existing model to integrate both.

In order to solve this problem using conventional approaches, such end users would need to have a basic knowledge of a data analytics programming languages such as Python and R and also basic knowledge of data science. The end user would choose the features based on the quality of the features, apply data type casting and data cleansing, and finally filter the features to build the dataset as an input for the prediction model. The end user would then need to choose the best model, algorithm, and validation method and adapt the characteristics based on the problem requirements.

2.2 Stroke Prediction Problem

Early prediction of strokes can have a huge impact on decreasing the number of mortalities each year. Consider a research group that decides to develop such a stroke prediction model. Since they do not have the funds to appoint professional data scientists as above, they plan to use the AWS SageMaker platform to create a model. The research group consists of PhD students, research associates, and academics with a basic knowledge of data analytics and data science. However, they lack sufficient Python or R

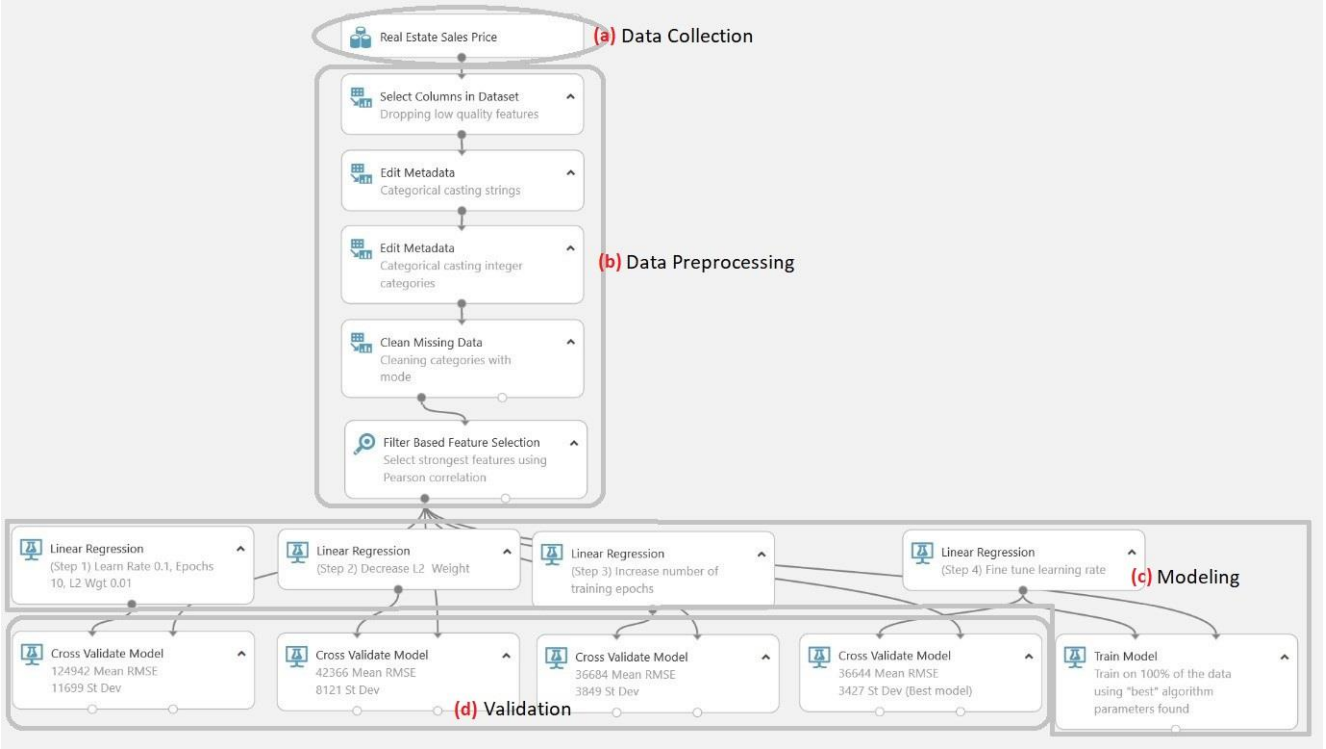

Figure 1. Real estate sales price prediction project in Azure ML Studio

programming skills to be able to directly develop their analysis and modelings. They decide to use the Mckinsey healthcare dataset [11] and create an account in AWS to use SageMaker and pay for the services they use.

They initially create a Jupyter notebook through the SageMaker dashboard, as shown in Figure 2 (a), and set up the SageMaker Amazon Resource Name (ARN) role. Then, through the (b) Jupyter notebook, they need to (c) set up the S3 bucket to use for training and saving the models. Required Python libraries need to be imported (d). They could then use either the ready to use templates provided by AWS SageMaker for the initial settings or do some training to learn Python. They then save the dataset in the S3 bucket, load the dataset and do some analysis and visualization, again using Python code, which needs some Python knowledge. They can create additional features by considering cross-product/interaction of multiple features, squaring or raising higher powers of the features to induce non-linear effects, etc. All these need to be programmed as shown in Figure 2 (e).

After finishing the analysis and visualization steps, they need to split training, testing and validation sets, convert the datasets to the recordIO-wrapped protobuf format used by the Amazon SageMaker algorithms, and then upload this data to S3. They use Amazon SageMaker's Linear Learner, which fits many models in parallel, each with slightly different hyperparameters, returning the one with the best fit. Finally, they can set up a model which can later be hosted. Once they have set up a model, they can configure and create the hosting endpoints, finally invoke the endpoint to obtain predictions. The endpoints need to be deleted when they are done. The data analytics model is now finalized, and they can re-run the model with different values of the hyper-parameters, loss functions etc, and also by using non-linear models available in SageMaker such as XGBoost, MXNet, etc to see if they can obtain improved prediction. Once they are done with tuning the paramaters and finding the best models which fit their problem, they can deploy their models.

All of these steps may be easy for a user with programming and a data science background. However, it is very problematic and timeconsuming for these health research end users. They find it very complicated to do all the data format changes and tuning settings.

2.3 Background

As illustrated with these two representative examples, there are many different stages in a data analytics application development life cycle. We refer to three comprehensive references to describe various perspectives on data science project life cycles.

In [4], the Data science methodology comprises 10 stages to form an iterative process for using data to uncover insights. These stages comprise business understanding, analytic approach, data requirements, data collection, data understanding, data preparation, modeling, evaluation, deployment, and feedback. As mentioned in this report, models are not created once, deployed and left in place as is; instead, through feedback, refinement and redeployment, models are continually improved and adapted to evolving conditions. In this way, both the model and the work behind it can provide continuous value to the organization for as long as the solution is needed.

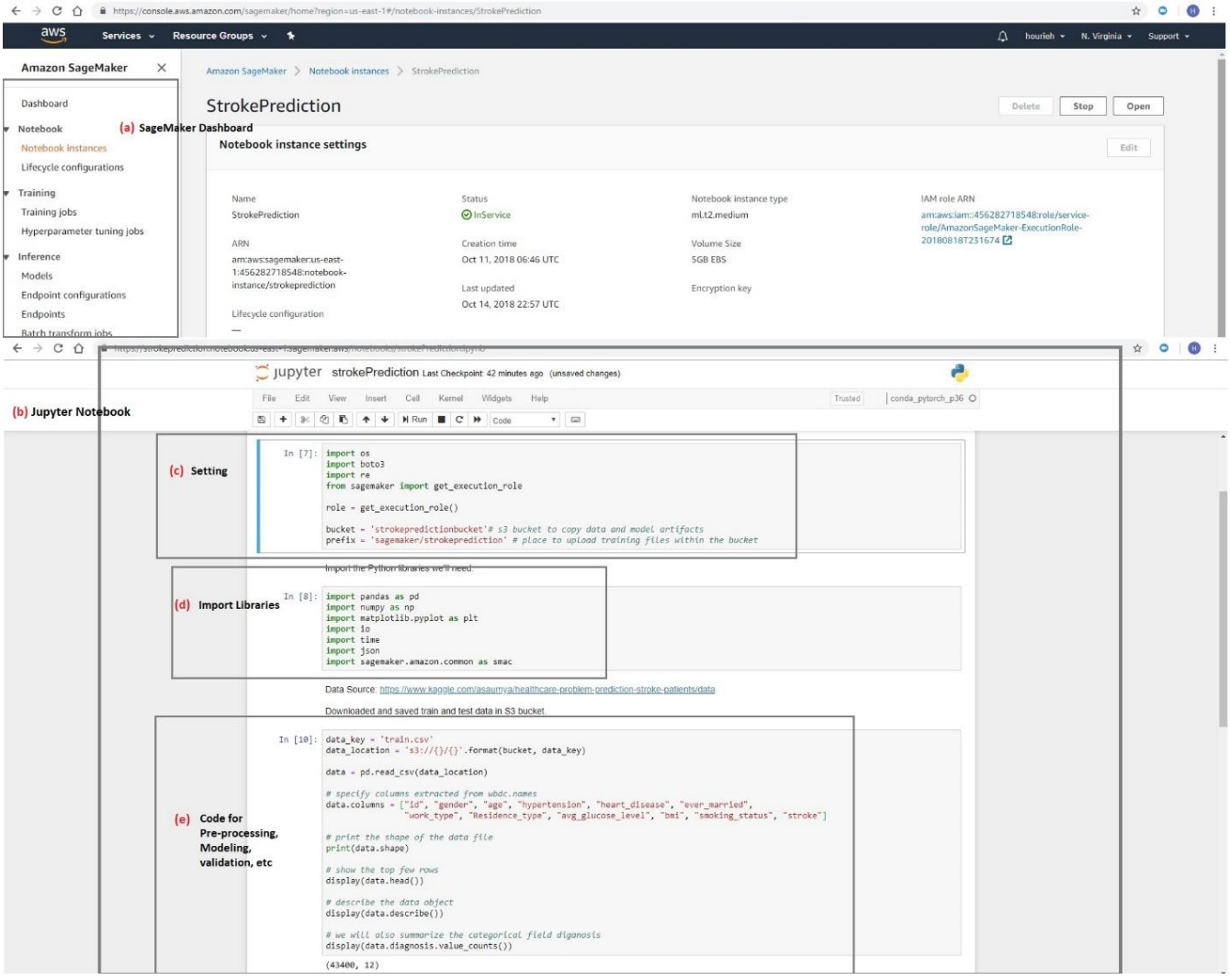

Figure 2. Stroke prediction project in AWSagemaker

In [12], a hierarchical process model, is described consisting of sets of tasks described at four levels of abstraction (from general to specific): phases, generic tasks, specialized tasks, and process instances. The CRISP-DM methodology, described in this paper, distinguishes four different dimensions of data mining contexts: the application domain; the data mining problem type; the technical aspect; and finally the tool and technique dimension. The reference model in this paper defines the life cycle of a data mining project as six phases of business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The sequence of the phases is not rigid and moving back and forth between different phases is always required. The user guide then describes the tasks, activities, techniques, outputs, and objectives of each step in the reference model in more detail. It finally focuses on the reports to be produced during and after a project for each step and suggests outlines for these reports.

Finally, [13] presents a technology independent reference architecture for big data systems, which is based on analysis of published implementation architectures of big data use cases. High-level design of the reference architecture in this paper consists of data sources, data extraction, data loading and preprocessing, data processing, data analysis, data loading and transformation, interfacing and visualization, data storage, and job and model specification. Different use cases such as Facebook, Linkedin, Twitter, and Netflix are then mapped into the presented reference architecture.

2.4 Requirements of End User Data Analytics Tools

Based on the key data analytics steps discussed in Section 2.3, we see that data processing and machine learning tasks are only a small component in the building blocks necessary to build real-world deployable data analytics systems, as shown by the small black box in the middle of Figure 3, covering the Machine Learning (ML) code, and the box covering feature extraction. Other major activities that need to be supported include designing the analytics pipeline, collecting and wrangling data, determining features from the dataset needing extraction, managing machines and processes for large scale analytics, presenting and using the resultant information, and monitoring the processes.

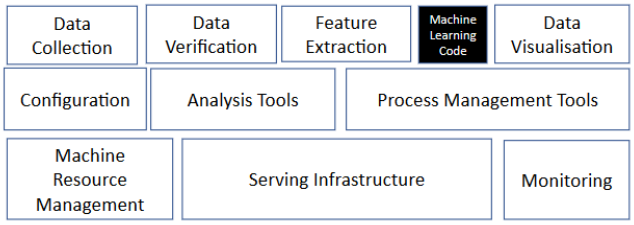

Figure 3. Artificial intelligence systems development building blocks (adapted from [14])

Key requirements that we have identified for such end user data analytics tools include the need for:

- Covering all data preprocessing operations such as cleaning, wrangling, anomaly detection and so on.
- Easy to understand tools for use by all groups of users including domain experts, data scientist and so on. Users with no data science and programming knowledge should be able to use them productively.
- Tools that provide a variety of the algorithms for each stage of data processing, modeling and evaluation processes e.g. different data integration, analysis and visualization algorithms and approaches, as well as guidance as to which is best to choose.
- Experienced users such as data scientists to have the ability to add their own features to the tool and incorporate their own specialized functions and algorithms. Therefore, the tool needs to offer flexible options for the users in case they want to expand the features based on their requirements.
- Covering all AI-SDLC stages including business problem description, requirements, design, implementation, testing and deployment.
- Industry ready tools that can be used for large scale industry-based projects, including scaling to very large datasets and leveraging large compute platforms.
- Cost effectiveness in terms of deployment, where users need to have the ability to deploy their solutions on the cloud or on premises or both.

We use these requirements in the next sections to analyze several current research and commercial toolsets, to compare and contrast them, and identify key strengths, weaknesses and gaps.

3 Overview of Current End User Data Analytics Tools

The traditional software development lifecycle (SDLC) includes: analysis of the requirements, design, implementation, testing, and maintenance. The current practices and tools for big data analytics applications lack coverage of most of the activities in the analysis and design phase. Most existing tools do not cater for business requirements and focus mainly on the implementation phases.

According to Google's analysis of the technical debt of AI systems in [14], only a small component of real-world ML systems is the ML model. The required surrounding infrastructure is vast and complex. There are a variety of tools developed to automate the ML code as well as the data verification and feature extraction phases.

We group these components (building blocks of an AI-powered system) into three groups: data-related activities (***DataOps***); artificial intelligence and machine learning -related activities (***AIOps***); and development and deployment activities (***DevOps***). The DataOps activities include data collection/ingestion, data validation, cleansing, wrangling, filtering, unioning, merging, etc. AIOps covers feature engineering and model selection, model training and tuning. Finally, DevOps covers model integration and deployment, monitoring and serving infrastructure.

From the perspective of DataOps, there are many widely used tools such as Tableau [15], Plotly [16], and Trifacta [17] that focus on the data operations such as visualization, data cleaning, and data wrangling.

As a well-known and popular example, Tableau provides end user support for visualizing, analyzing, and understanding the data. End users are typically any business person wanting to explore data insights. It offers an interactive visual analysis toolbox which allows its users to obtain insights that support their business. An example of Tableau in use is shown in Figure 4. In this example the user wants to gain some insights on housing sale and profit within different cities. Users need to choose and connect to their data stored in a file or a server. By loading the dataset, the features are accessible for the users and they can drag and drop different features and generate different charts and add more fields to obtain the right level of detail based on their need. Then, the users are ready to begin focusing on their results and use filters and colors to help them explore their data visually. For example, the users might want to know sales and profit in places which have some problems and therefore build a map view to leverage this issue and find a solution for it. In this example, Tableau has automatically assigned the proper geographic roles to different countries and the states, cities, and postal code fields within different countries. Finally, once all the visualizations include all required details, users can build a dashboard to keep and use their insights and finally to create a story in Tableau to present their insights and walk the viewers through their data discovery process.

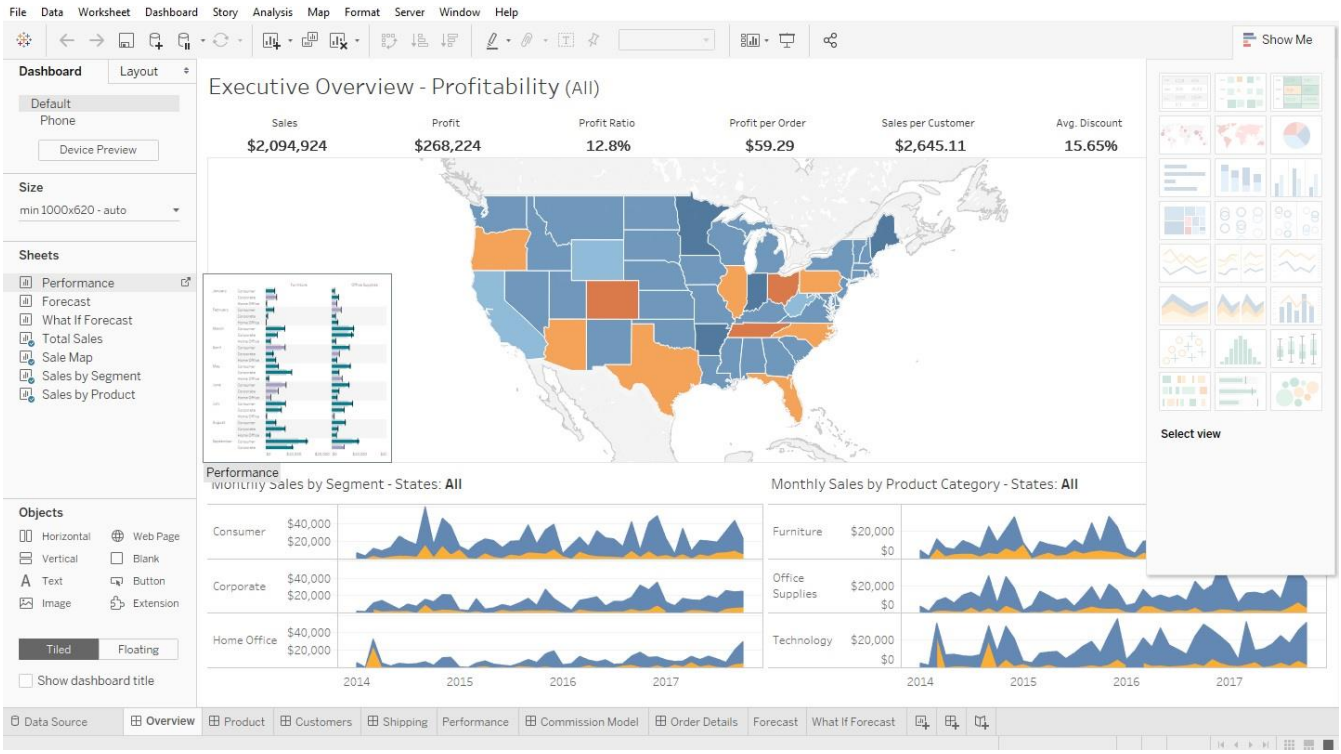

Figure 4. An example of Tableau in use

From the viewpoint of AIOps, a large number of tools focus on the artificial intelligence and machine learning operations. Most tools in the market belong to this group. Examples are Azure ML Studio [6], Amazon AWS ML [7], Google Cloud ML [8], BigMl [9], Weka [18], TensorBoard [19], MLJar [20], and DeepCognition [21]. The tools in this group also often cover DataOps to some extent.

A representative example of such tools is Azure ML Studio as shown in Section 2, Figure 1. This provides different data visualization and preprocessing, and machine leaning algorithms with an interactive drag and drop method. Target end users include business analysts but also experienced data scientists. An example of the tool in use is shown in Figure 5. In this example, price estimation for vehicles, users create data analysis experiments by dragging different modules to the canvas to do the analysis. To start, users upload their own datasets or find a related dataset to their problem from a list of available datasets and drag the chosen dataset to the experiment canvas. Then, they visualize and prepare their data. To preprocess the data some modules are available such as “Select Columns in Dataset”, “Clean Missing Data”, “Apply Filters”, “Add Columns”, “Add rows”, and many other modules with different editable properties. Different modules can be connected, and the experiment can be run at any stage.

After reading, visualizing, and preprocessing the data, data needs to be split to training and testing parts by dragging the “Split Data” module to the canvas and modifying the properties. Then, a variety of learning algorithms such as logistic regression, decision forest, neural network, decision tree, and support vector machine, can be applied to the dataset by choosing the appropriate modules. Then, the dataset can be trained, tested, and evaluated and the results can be visualized simultaneously. Finally, users can iteratively change and add or delete new modules to improve the model. For expert users, R and Python scripts can also be embedded into the model.

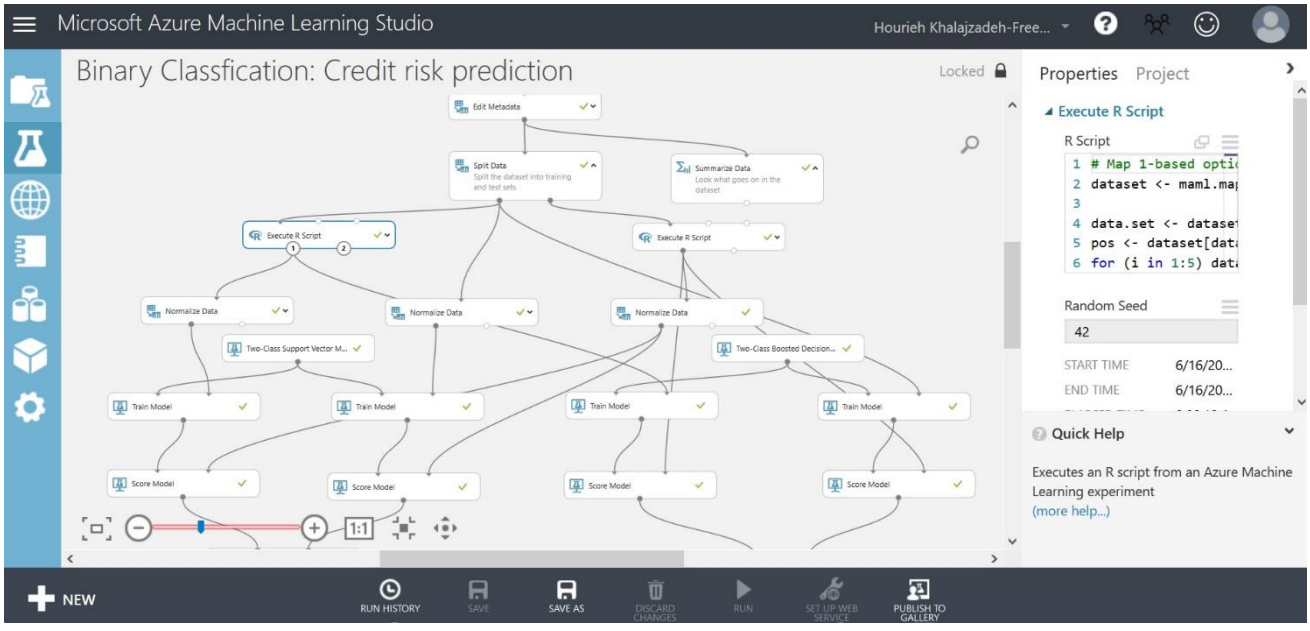

Figure 5. An example of Azure ML Studio in use

Finally, ***from the DevOps point of view***, some of the available tools focus on the deployment of the solutions on the cloud or on premises as well as building industry ready solutions. This group includes some of the tools within the second group. Example Devops focused tools are Rapidminer [22], IBM Watson ML [23], SAS [24], KNIME [25], DataRobot [26] and AWS SageMaker [27].

One such tool is KNIME. This provides an industry ready open source data analytics solution covering all the data, AI, and deployment operations. Target end users are data scientists and experienced end users wanting to build and deploy comprehensive AI-based solutions. Nodes can be added in KNIME by expanding the "Node Repository" and dragging and dropping different nodes to the workflow editor window. As shown in Figure 6, the "File Reader" icon is used for uploading datasets by expanding the "IO" and the contained "Read" category. A variety of nodes such as classification and clustering algorithms, manager node as well as interactive table and scatter can be used to visualize data and build different models. Then, nodes can be connected together in order to construct the appropriate dataflows. Finally, nodes can be iteratively configured and executed.

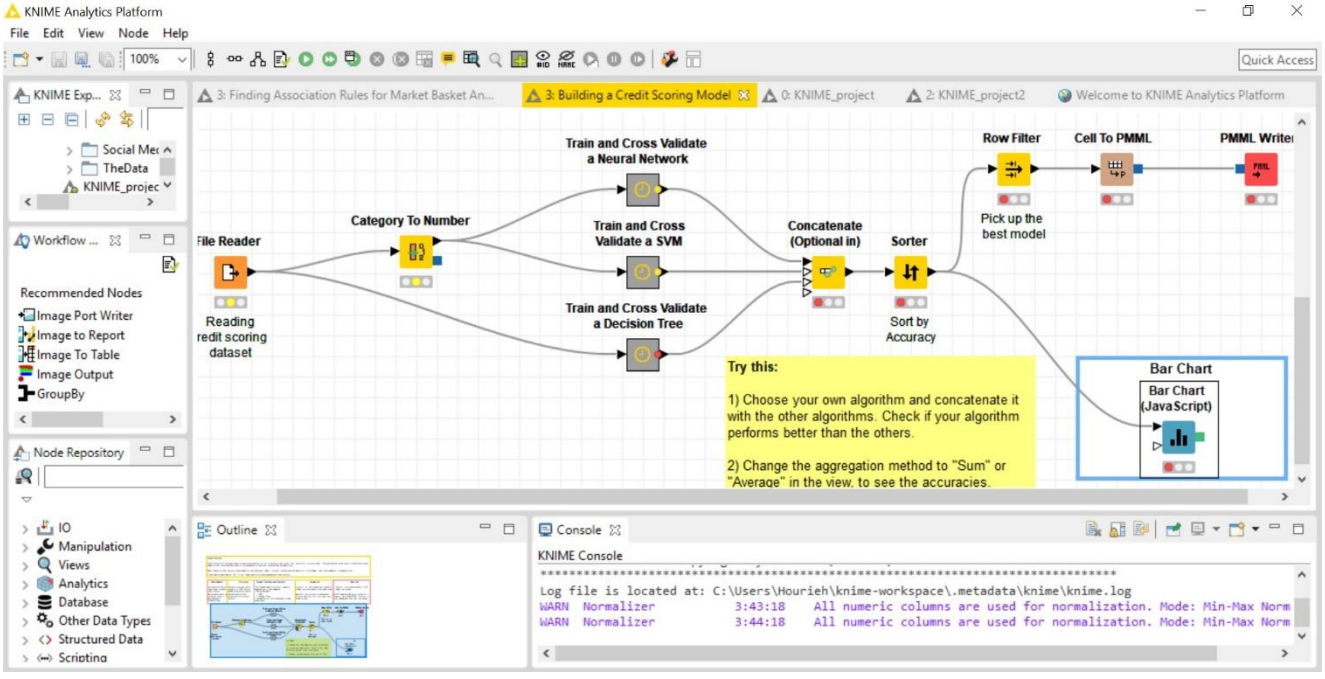

Figure 6. An example of KNIME in use

4 Landscape analysis

We have selected a few most popular representative tools within these three groups of DataOps, AIOps, and DevOps and their key strengths and weaknesses are analysed and compared. These tools are commonly used by students, data scientists and software engineers, and are nor purely programming based, i.e. R and Python neither completely pre-trained services, i.e. Google Cloud AI & Machine Learning Products. It was not possible to compare all the existing tools and therefore we limited our tool selection to the most popular and comprehensive products that have been in the market for quite a long time. Some of these were selected due to their being reported as the leaders within industry-based tools according to [1], based on completeness of vision and ability to execute, as shown in Figure 7.

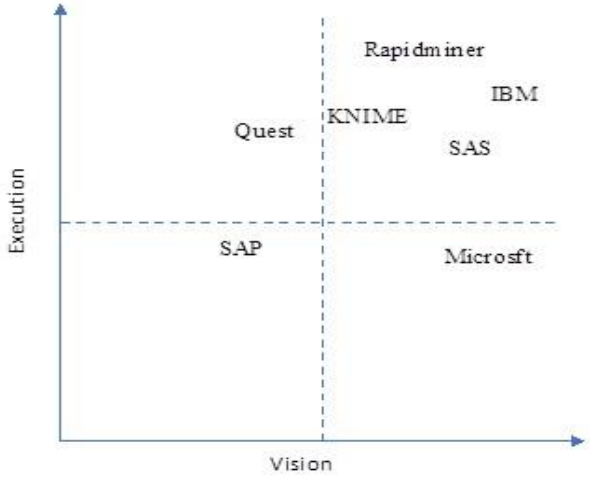

Figure 7. Based on Gartner 2017 Magic Quadrant for Data Science Platforms [1]

The tools are briefly introduced in Section 4.1 and their strengths and weaknesses are discussed. A comparison of selected tools is shown in tables 1, 2 and 3 from the perspectives of operations, usability and cognitive effectiveness, and discussed in detail in Section 4.2.

4.1 Broad Landscape

Tools, grouped in the three categories of DataOps, AIOps, and DevOps introduced in Section 3, are discussed in more detail in this section. There are conflicts in grouping some of these tools into just one of these categories, but we have chosen the best fit for each tool in case they cover features of more than one group.

4.1.1 DataOps Tools

Within the first group of DataOps, Tableau, Plotly, and Trifacta Wrangler are chosen and compared in Tables 1-3. Tableau is a very comprehensive and easy to use tool used by many organizations to visualize and understand data. Virtually no knowledge of data analytics and programming is required to use this tool. At the same time, users can embed their own code to extend the usability of this tool, but only to a limited extent. Plotly provides open source Matlab, Python and R libraries for data scientists to create interactive charts in Matlab, Python, and R. These libraries seamlessly interface with hosting servers (cloud or on premises) for easy collaboration, code-free editing, and collaboration of data scientists with designers, and dashboard composition. It also offers different plans on library-specific support options including code review, hands-on assistance with implementation via email, chat and phone for developers, teams, and non-experts. Trifacta Wrangler automatically profiles the uploaded data, shows the percentage of the missing, mismatching, or inconsistent values and also gives data distribution using histograms. With Trifacta, stakeholders, analytics executives, IT leaders, and data analysts have visibility into the process of data preparation in order to see the impact of changes throughout the entire lifecycle. Users can instantly gain insights into their data quality using this tool but only on datasets with a limited size. All these tools, however, are only helpful for visualizing, preprocessing, and gaining insights on data. For gaining deeper insights and to create a model and build solution, we need the next group of the tools.

4.1.2 AIOps Tools

Most existing tools cover some aspects of AIOps. The first three tools chosen in this group are the well-known Cloud-based machine learning tools, Azure ML Studio, Amazon AWS ML, and Google Cloud ML. Azure ML Studio is a very efficient and easy to understand and use drag-and-drop tool. Also, for the benefit of data scientists or users familiar with programming, Python and R codes can be added to expand features where required. However, for non-expert users a basic knowledge of data science is required. Amazon AWS ML is also easy to understand and use but does not support embedding scripts and code. AWS ML is not very comprehensive in terms of the algorithms users can choose and also operations are only accessible by defining recipes, which is not an easy option for non-experts and users lacking basic knowledge of data science. Google Cloud ML offers more features than Azure and Amazon, but good knowledge of the TensorFlow [28] library is required to use the features, and it is not very easy to use for non-expert users as deep technical knowledge of ML is mandatory.

BigML is chosen as both a visual and a programmatic machine learning tool for developers. It is easy to use, but it is not possible to manipulate and change the algorithms. Therefore, it is not possible to use any algorithms and models other than those provided. Weka is an easy to use tool which is mostly used in research settings. Weka does not cover all data analysis algorithms and it is not possible to manipulate and add code to change the functionality of any of them. It is mostly suitable for smaller datasets and simulations, not large-scale proejcts and deployment.

TensorBoard offers visualization to make use of TensorFlow, an open-source software library widely used for dataflow programming across a range of tasks. TensorFlow programs are included in a suite of visualization tools called TensorBoard, which makes complex and confusing TensorFlow based computations such as training a massive deep neural network easier to understand, debug, and

optimize. TensorBoard visualizes the TensorFlow graph, plots quantitative metrics about the execution of the graphs, and shows additional data like images that pass through it. It is very comprehensive, and the programmers have the freedom to change everything based on their needs. However, a deep knowledge of data analytics and machine learning is needed in order to use this tool and also initial programming is required to display the model and components visually. Users need to install TensorFlow and TensorBoard through commands and the tool is not easy to use for non programmers and non data scientists.

MLJar offers some basic data preprocessing tasks such as imputing missing values and converting categorical attributes. It checks many different algorithms and tunes hyper-parameters separately for each model in parallel through MLJAR cloud. There are two types of interface available in MLJAR, an easy interface for beginner users with no knowledge of data science and programming and also a python wrapper over MLJAR API, enabling programmers and data scientists to write Python code. With MLJAR, it is possible to share the results with others and it provides a REST API for deploying the created model. The next tool, DeepCognition, simplifies and accelerates the process of working with deep learning across popular frameworks such as TensorFlow and MXNet. Data can be uploaded in several different formats, and can be also accessed from other cloud repositories such as AWS S3, Google Cloud and more. The simple drag and drop interface helps to easily design deep learning models. Users can also import model code and edit the model with the visual interface. One of the features of DeepCognition is that it generates code as users are building their model. Users can train their hyperparameters using DeepCognition's multi GPU training system and can also manually compare the results of different experiments. It offers one-click deployment of the model as a REST API and also generates a form based web application to showcase the final solution. However, as the name indicates, it only offers deep learning algorithms. Using these tools, users can build their data analytics solutions. Some of these tools cover DataOps as well and DevOps to some extent. However, not all of them are industry ready and usable for large scale problems. Industry-based tools fit in the next group of tools.

4.1.3 DevOps Tools

In the DevOps tool category, a variety of tools have been produced, including Rapidminer, IBM Watson ML, SAS, KNIME, SageMaker, and DataRobot.

These tools support the development of industry ready solutions deployable on the cloud or in house, and they cover DataOps as well as AIOps. Rapidminer is currently a very comprehensive and easy to use tool for preprocessing, modeling, and deploying data analytics solutions. In terms of flexibility, it offers some advance parameter setting but not through code embedding. IBM Watson ML is easy to use and it offers both visual drag and drop for non-experts and code based modeling for experts. SAS is very comprehensive for industry-based usage. However, it is expensive and, since it is completely programming based, knowledge of programming is required. However, data manipulation is easier than other programming tools such as R and can be done visualy. KNIME is very comprehensive and easy to use. It is possible to change and manipulate the algorithms based on different users' requirements. It is also very efficient and quick, however a complete overview and knowledge for designing the workflow is required for users to be able to use KNIME effectively. In order to use all these tools, knowledge of data science is required.

DataRobot is a very comprehensive tool which applies exploratory data analysis, supports many algorithms provided by python, presents the accuracy for each model, chooses the best model based on a variety of factors, and visualizes and tests the final model. It does a complete preprocessing and has a complete brief description and implementation documentation for all models. It offers an easy to use interface with default setting for beginner users and it is feasible for a data scientist to change all settings and use Jupyter notebook to add their own code/algorithms. It has a model competition framework from different sources (Python, R, tensorFlow, etc), which uses many models in parallel and presents a leader board. It shows process visualization and workflow blue print, and proposes hyper parameter tuning. DataRobot is the only tool offering strategies for business problem description in order to communicate model designs with business end users, permitting them to incorporate their domain expertise, and gain an understanding of how the model is performing. In order to make the decision making explainable and prescriptive, DataRobot provides reasons for prediction at the prediction level, prediction likelihood of the target and also the driving factors for the decision. This makes models powerful and communicable to end users.

Finally, AWS SageMaker aims to remove all the barriers that typically slow down developers who want to use machine learning. It includes hosted Jupyter notebooks that make it easy to explore and visualize training data stored in Amazon S3. Users can connect directly to data in S3, or use AWS Glue to move data from Amazon RDS, Amazon DynamoDB, and Amazon Redshift into S3 for analysis in the Jupyter notebook. However, since SageMaker is dependent on Jupyter notebook, it is difficult and time intensive for users with no data science or programming knowledge to be able to explore and use this tool.

4.2 Tool Comparison and Analysis

To compare the tools introduced above we considered this set of representative example tools, downloaded and installed the tools, used the tools on our two use cases, property price prediction and stroke prediction, and then scored the tools against the criteria below.

4.2.1 Functionality Aspects

From a functionality perspective, the tools are categorized by the AI-SDLC phases, discussed in Section 3: business problem description, requirements, design, implementation, testing and deployment. These operations are compared for different tools in table 1. The ✓ symbol indicates the tool completely covers the requirements for the specific step, while ~ indicates the step is partially covered

and X indicates that the tool has no coverage of the specific step.

Based on the findings of the functionality assessment, shown in Table 1, none of the tools support business and requirements capture, modeling, and design. Most focus on low-level data analytics phases such as Dataops and AI-Ops and need a dataset to be uploaded in the first stage the user starts to work with the tool. However, there are many steps required before creating a dataset for a domain expert or end user. Also, they mostly focus on cloud with less focus on in house deployment.

4.2.2 Usability Aspects

We used several usability criteria including industry ready, cost, usability, comprehensiveness, flexibility, and data science knowledge required. The industry ready criteria examines whether the tool is ready and efficient to be used in industry. For cost, some tools offer a completely free product, some offer plan based or pay-as-go rates, while some offer a combination of free limited access and plan bassed extended access. Usability measures whether the tool is easy to use for different types of users including data scientists, software engineers, business owners, domain experts, researchers, etc. Comprehensiveness shows how complete and comprehensive the tool is in terms of the algorithms and models they offer and cover and whether new algorithms and models can be added by the user. The flexibility of a tool is measured by its ability to embed code in different languages, such as Python and R, making the tool effective for a variety of users including experts and beginners. Finally, usability for non-data science experts with no knowledge of data analytics and science is the last critiera we compare and discuss.

The usability aspect analysis is shown in table 2. For most of the tools, there is a clear trade-off between usability and comprehensiveness. If the tool is easy to learn and use, it only provides a limited list of algorithms while the tools with an extensive list of algorithms provided, are usually not easy for users lacking comprehensive knowledge of data science to learn and use the tool. Knowledge of data analytics and data science as well as programming languages such as Python and R are required for most tools.

4.2.3 Cognitive Effectiveness

Finally, in table 3, the "goodness" of visual symbols for each tool are evaluated in terms of their cognitive effectiveness. The term cognitive effectiveness is defined as "the speed, ease, and accuracy with which a representation can be processed by the human mind" [11]. Cognitive effectiveness determines the ability of visual symbols to communicate with a wide range of users, data scientists, and stakeholders. Physics of Notations (PoN) [29] was used to compare the cognitive effectiveness of these tools. PoN proposes a domain-specific theory for visual languges and defines a set of principles to evaluate, compare, and construct visual symbols. These principles include semiotic clarity, perceptual discriminability, semantic transparency, complexity management, cognitive integration, visual expressiveness, dual coding, graphic economy and cognitive fit.

Semiotic clarity specifies that a diagram should not have symbol redundancy, overload, excess and deficit. Perceptual discriminability is primarily determined by the *visual* distance between symbols. This is measured by the number of visual variables on which they differ and the size of these differences. Semantic transparency identifies the extent to which the meaning of a symbol should be inferred from its appearance. Complexity management restricts a diagram to have as few visual elements as possible to reduce its diagrammatic complexity. Cognitive integration identifies that the information from separate diagrams should be assembled into a coherent mental representation of a system; and it should be as simple as possible to navigate between diagrams. Visual expressiveness defines a range of visual variables to be used, resulting in a perceptually enriched representation that exploits multiple visual communication channels and maximizes computational offloading. Dual coding means that textual encoding should also be used, as it is most effective when used in a supporting role. Graphic economy discusses that the number of different visual symbols should be cognitively manageable. Finally, cognitive fit means that the diagram needs to have different visual dialects for different tasks or users.

Nine design rules are defined based on these principles in [30] as follows:

1. All visual symbols should have 1:1 correspondence to their referred concepts.
2. All symbols should use different shapes as their main visual variable, plus redundant coding such as color and/or textural annotation.
3. Icons need to be used to represent visual symbols and minimize the use of abstract geometrical shapes.
4. Hierarchical views need to be used for representation and users should be allowed to hide visual construct details for complex diagrams.
5. All the relationships between diagrams should be in a hierarchical tree structure, and child diagrams should be opened only from their parent diagram.
6. Various visual variables, such as shape, color, orientation, texture, etc should be used when designing visual symbols.
7. All visual symbols should have a textual annotation.
8. As few visual symbols as possible should be used in a DSVL.
9. All the symbols need to be usable for different users and tasks.

The tools are compared based on these aspects as shown in table 3. The ✓ symbol indicates that the tool fulfills the cognitive principle, while ~ indicates the principle has been partially fulfilled and there is space for improvement and X indicates that there is no visualization available for the tool or the rule is not supported. Based on our findings, some of the tools such as Azure ML Studio, DeepCognition, Rapidminer, KNIME, and DataRobot offer a completely drag and drop based fully automated tool, which are well designed from a cognitive effectiveness perspective. Some, such as Plotly, are completely code based, and others have a partially automated tool with a user interface

and different notations by which users can choose the steps and methods. These tools can lead to confusion in their use and users need to spend time getting to know the tool and notations. Moreover, a solution developed by a fully automated tool can not be reused later. On the other hand, no automation at all makes the tool difficult to learn and use for non-data scientist users.

TABLE 1
FUNCTIONALITY COMPARISON OF DIFFERENT DATA ANALYTICS TOOLS

End User Tool Examples				Tableau	Plotly	Trifacta	Azure ML Studio	Amazon AWS ML	Google Cloud ML	BigML	Weka	TensorBoard	MLJar	DeepCognition	Rapidminer	IBM Watson ML	SAS	KNIME	DataRobot	AWS Sagemaker
SDLC	Business problem description			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	~	X
	Requirements			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Design			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Implementation	DataOps		✓	✓	✓	✓	~	✓	✓	~	X	~	✓	✓	✓	✓	✓	✓	✓
		AIOps		X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Testing			X	X	X	X	X	X	X	X	X	X	X	X	✓	X	X	✓	✓
	Deployment	DevOps	Cloud based	✓	✓	✓	✓	✓	✓	✓	~	✓	✓	✓	✓	✓	✓	✓	✓	✓
			On premises	✓	✓	X	X	X	X	✓	✓	X	✓	X	✓	✓	✓	X	✓	X

TABLE 2
USABILITY COMPARISON OF DIFFERENT DATA ANALYTICS TOOLS

End User Tool Examples			Tableau	Plotly	Trifacta	Azure ML Studio	Amazon AWS ML	Google Cloud ML	BigML	Weka	TensorBoard	MLJar	DeepCognition	Rapidminer	IBM Watson ML	SAS	KNIME	DataRobot	AWS Sagemaker
Tool Usability	Industry ready		✓	✓	X	X	X	~	X	X	✓	~	~	✓	✓	✓	✓	✓	✓
	Cost	Free trial/for limited access	✓	✓	✓	✓	X	✓	✓	✓	✓	~	✓	✓	X	X	✓	X	✓
		Plan based/pay as you go	✓	✓	X	✓	✓	✓	✓	X	X	✓	✓	✓	✓	✓	X	✓	✓
	Usability		✓	X	✓	✓	✓	X	✓	✓	X	✓	✓	✓	✓	X	~	✓	~

	Comprehensiveness	✓	✓	~	X	X	✓	X	X	✓	X	~	✓	✓	✓	✓	✓	✓
	Flexibility	X	✓	~	✓	X	X	✓	X	✓	✓	~	~	✓	✓	✓	✓	✓
	No Data science knowledge required	✓	X	✓	X	X	X	X	X	X	~	~	X	X	X	X	~	X

TABLE 3
COGNITIVE DIMENSIONS OF DIFFERENT DATA ANALYTICS TOOLS

End User Tool Examples		**Tableau**	**Plotly**	**Trifacta**	**Azure ML Studio**	**Amazon AWS ML**	**Google Cloud ML**	**BigML**	**Weka**	**TensorBoard**	**MLJar**	**DeepCognition**	**Rapidminer**	**IBM Watson ML**	**SAS**	**KNIME**	**DataRobot**	**AWS Sagemaker**
Physics of Notations Rules	**Semiotic Clarity**	✓	X	~	✓	~	X	✓	~	~	~	✓	✓	~	~	✓	✓	~
	Perceptual Discriminability	✓	X	~	~	~	X	✓	~	~	~	✓	✓	~	~	~	✓	~
	Semantic Transparency	✓	X	~	✓	~	X	✓	✓	~	✓	✓	✓	✓	~	✓	✓	~
	Complexity Management	✓	X	✓	✓	~	X	✓	~	✓	✓	✓	✓	✓	~	~	✓	~
	Cognitive Integration	✓	X	✓	✓	~	X	✓	~	✓	✓	✓	✓	✓	~	~	✓	~
	Visual Expressiveness	✓	X	~	✓	~	X	✓	~	~	~	✓	✓	~	~	✓	✓	~
	Dual Coding	✓	X	✓	✓	✓	X	✓	✓	~	✓	✓	✓	✓	~	✓	✓	~
	Graphic Economy	✓	X	✓	✓	✓	X	✓	✓	~	✓	✓	✓	✓	✓	~	✓	✓
	Cognitive Fit	✓	X	✓	✓	X	X	✓	~	~	✓	✓	✓	✓	X	✓	✓	X

4.3 Analysis of the Findings

Based on the strengths and weaknesses of different tools discussed in Section 4.1 and shown in tables 1-3, we analyse our findings for each of the tools in this section.

Tableau is a very comprehensive and easy to use tool broadly used in industry for data analysis, however, it only offers data operations and needs to be used with another tool for modeling and training purposes. Plotly is a visualization library embeddable in a variety of data analytics programming languages and obviously usable by the programmers of these languages only. Trifacta also covers DataOps only and is mostly recomended for small training/education based problems.

Azure ML Studio is a very easy drag and drop tool mostly for training/education purposes with a basic knowledge of data science required. Amazon AWS ML, Weka, BigML, and MLJar do not provide as many algorithms and are not a completely drag and drop tool like Azure ML Studio, however they are easy to use tools for training/education purposes. AWS Sagemaker is another AWS machine learning tool that is very comprehensive and industry ready, however it is completely based on programming in Jupyter notebook and both difficult to use for end-users and not the best option for data scientists since they can use the Jupyter notebook directly and move their final models to AWS if required. Google Cloud ML is comparable to AWS ML, however more comprehensive, but not as easy to use as AWS ML. IBM Watson ML offers an easy to use, comprehensive and industry ready tool, however for someone with knowledge of data science. TensorBoard and DeepCognition provide comprehensive tools for deep learning specifically but they both need knowledge of deep learning to be usable.

DataRobot, Rapidminer and KNIME are among the most comprehensive and easiest tools to use. These three tools offer a comprehensive list of algorithms and they are scalable to be used in industry. KNIME can be downloaded free of charge, Rapidminer offers a limited free access and different plans for extended access, while DataRobot can not be used without buying the product. DataRobot is the only tool with some initial basic attempts to include business knowledge by enabling documentation. However, these tools still need considerable knowledge of data analytics and data science as well as a basic knowledge of programming to be applicable to a broader range of domains. The problem with most of these tools is that they are complicated for a non-expert to make use of. At the same time,

they are over-simplified and also a black-box for expert data scientists and software engineers who are interested to see inside the box. Data science experts usually do not use these tools because they find writing their own programs using programming languages such as Python and R more comprehensive, applicable to different domains, are re-usable, and without further cost.

Moreover, all of the tools reviewed only cover a few phases of data verification, feature extraction, and machine learning specified in Figure 3. In terms of the data verification phase, a dataset containing the features and examples needs to be collected and provided before getting started with any of these tools. Based on [31], the common issues with tools such as Weka, R [32], Orange [33], KNIME, Scikit learn [34], Mahout [35], and Spark Mllib [36] are that they are over complicated for non-experts, over-simplified for real world problems, and poorly engineered for real world or high scale usage. Additionally, commercial tools such as SAS and SPSS [37] inherit the same issues while they are expensive as well.

5 Research Directions

Based on the analysis of our findings discussed in Section 4.2, there are several key directions for the future research in developing tools for data analytics problems:

At present most tools focus on low-level data analytics process design, coding and visualization of results. There is a **need to better capture requirements**, changes in the requirements, and adaptation of the specified process. In many ways these AI-based systems are no different in terms of their need for good requirements and high-level design models than are other more traditional systems. So saying, there is a need to better support domain expert end users in their requirements management for AI-based systems, providing approaches to capture their requirements, not so much with respect to the software solution but the domain problem and available data and business intelligence needed to solve it.

To unleash the value of big data analytics, data analytics and machine learning steps need to be tightly connected to the control and management of business and requirement engineering processes. However, as mentioned earlier, the primary focus of big data and data analytics tools and technologies is currently on storage, processing, and particular data analysis and machine learning tasks. Current data analytics tools rarely focus on the improvement of end-to-end processes. Data science approaches tend to be process agonistic whereas process science approaches tend to be model-driven without considering the "evidence" hidden in the data. **Bringing scalable process mining analytics on top of big data toolkits, while enabling them to be easily tailored to domain-specific requirements** is encouraged in [38]. Therefore, tools filling these gaps would be very useful for end-users and data scientists, for the discovery and exploration phase, to be able to model the problem, extract insights/patterns, and develop predictive and clustering models if it is feasible before involving software engineers.

Most current tools assume data is in a form more or less amenable to processing by ML algorithms. However, many real-world datasets are not "clean" nor "integrated", and great effort is **needed to source the data, integrate and harmonize it, pre-process and cleanse it**, organize it according to required algorithm needs, and then provide it for processing. As pointed out in [39], some of the data related challenges data scientists are facing are 1) data quality due to the bugs in data collection and sampling procedure, 2) data availability, which is described as a lack of data, missing data values, delayed data, locating the right data sources for analysis, and gaining data permission and access 3) data preparation, i.e. the integration of data from multiple sources and shaping of the data into a right format. Complex, large datasets introduce many issues that need to be supported, including handling partial and incomplete datasets.

Many existing tools **need better support to be extensible based on the user' requirements**. Only a few tools, notably Azure ML Studio, BigML, IBM Watson ML, SAS, KNIME, MLJar, DataRobot, AWS SageMaker and TensorBoard offer the ability for a data science expert to embed new code and expand the algorithms and usability based on their own needs. Live programming is an alternative, proposed in [40], where the user's edits immediately and automatically update the script results. However, including these features must not make the tool over complicated so it remains usable for non-expert users with no programming background. Therefore, both simplicity for non-experts with no data science and programming knowledge, and support for expansion and tailoring for data science experts need to be provided.

Some of the tools such as Azure ML Studio, DeepCognition, Rapidminer, KNIME, and DataRobot offer a completely drag and drop based fully automated tool while other tools such as Google Cloud ML, IBM Watson ML, and SageMaker have a partially automated tool with a user interface and different notations by which users can choose the steps and methods. A solution developed by a fully or partly automated tool can not be reused later. On the other hand, some tools such as Plotly, are completely code based, which helps users to reuse their code in their other future projects, however, a pure code based tool with no automation at all makes the tool difficult to learn and use for non-data scientist users. There is currently no **fully or partially automated tool which generates code from visualization and can be easily re-used later**.

Most current tools only cover the DataOps, AIOps, and DevOps parts of the data analytics life cycle. Some, such as Tableau and TensorBoard, cover one phase and some such as Rapidminer and KNIME, try to cover all the phases. However, as discussed above, data verification, feature extraction, and ML code are only small parts of the whole systems development life cycle. At present there is **no tool covering all the phases including analysis of requirements, design, implementation, testing, and maintenance**. DataRobot covers some initial steps of business problem description but there is still a gap in providing tool features to capture the requirements and changes in

requirements as well as adapting the solution based on these changes.

Many real-world problems require: large datasets to be processed; computationally expensive algorithms; and thus deployment of solutions on complex, powerful computing infrastructure. Some tools hide all of these details from the user but provide limited configuration and flexibility; some tools require users to have detailed knowledge of configuring and deploying solutions; and many only support limited single desktop solutions that do not scale. According to [39], batch processing jobs like Hadoop can take a while due to the huge data size. Therefore, iterative work flow can be expensive and quick visualization of large data sets is an obstacle. Despite a large suite of diverse tools, it is difficult for data scientists to access the right tools, because generic tools do not work for specific problems that they have. When it comes to building predictive models, difficulty of knowing key tricks related to feature engineering and evaluating different models for optimal performance is one of the challenges for data scientists. It is also difficult to infer the right signal from the data and what confidence level should be appropriate for the analysis [39]. Thus, **scaling and distribution needs to be supported for real-world applications**, while considering limited end user knowledge of computing platforms.

Many tools provide a variety of visualization support approaches to support business decision making with the information produced by AI and ML algorithms. However, these are typically limited to built-in visualization options or the specification of complex visualizations require programming knowledge. Some visualizations allow end user information exploration while others are static displays. Further **enhancement of information visualization capabilities is needed**, including interactive exploration and end user specification of complex visualizations for their target domain.

Last and the most challenging issue is that in the large organizations, there are many different parts and different data analysts and scientists are working separately on different projects without communicating and reusing the existing information and models. Moreover, it is not easy to communicate the results with the higher level managers and showcase the data science models and outcomes before they are finalized and deployed due to the lack of common language and dialect. Data scientists need to wait until they have their analysis and models finalized to be able to report their evaluation results and model accuracies to the higher level managers and non data scientists. Furthermore, many organizations prefer to have a simpler and less accurate while more explainable model because of the explainability issue of the complex models. Another challenge reported by data scientists in [39] is that it is hard to convey the resulting insights to leaders and stakeholders in an effective manner and convincing teams that data science actually is helpful. Communicating to the team and getting all stakeholders on the same page is reported to be very challenging for data scientists in this paper. Therefore, there is **need for tools enabling collaboration between all people involved in data analytics projects in different parts of an organization with a same language and dialect**.

6 Conclusions and Future Work

Many new data analytics tools covering different data analytics phases, including DataOps, AIOps, and DevOps, have been developed for both research and industry usage. We have introduced, summarized and compared a number in this paper. Such tools help data scientists to specify, integrate and apply complex data analytics and visualization techniques to build a range of big data applications. Based on our analysis of these tools we see that most tools currently focus on the data analysis and machine learning modeling and implementation phases. This is only a small part of the AI-software development life cycle. Furthermore, the tools are complicated for a domain expert with no data science and programming background, and are not designed to allow for collaboration between key stakeholders (team members) involved in the development of the AI-powered software systems. In our future research, we will be looking at alternative approaches to provide domain experts with the tools they can directly use to cover the whole lifecycle of AI-based systems development.

Acknowledgment

This work was supported by ARC Discovery grant DP170101932.

References

[1] G. Piatetsky. (2017). *Gartner 2017 Magic Quadrant for Data Science Platforms: gainers and losers*. Available: https://www.kdnuggets.com/2017/02/gartner-2017-mq-data-science-platforms-gainers-losers.html

[2] I. Portugal, P. Alencar, and D. Cowan, "A Preliminary Survey on Domain-Specific Languages for Machine Learning in Big Data," presented at the IEEE International Conference on Software Science, Technology and Engineering (SWSTE), Beer-Sheva, Israel, 2016.

[3] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin, "A Survey of Open Source Tools for Machine Learning with Big Data in the Hadoop Ecosystem," *Journal of Big Data*, vol. 2, no. 24, 2015.

[4] J. B. Rollins, "Foundational Methodology for Data Science," IBM Analytics2015.

[5] C. E. Sapp, "Preparing and Architecting for Machine Learning," Gartner Technical Professional Advice2017.

[6] *Microsoft Azure Machine Learning Studio*. Available: https://studio.azureml.net/

[7] *Machine Learning at AWS - Amazon AWS*. Available: https://aws.amazon.com/machine-learning/

[8] *Predictive Analytics - Cloud Machine Learning Engine | Google Cloud*. Available: https://cloud.google.com/products/machine-learning/

[9] *BigML.com is Machine Learning Made Easy*. Available: https://bigml.com/

[10] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "A Survey of Current End-user Data Analytics Tool

Support," in *IEEE International Congress on Big Data 2018*, San Francisco, USA, 2018, pp. 41-48.

[11] J. H.Larkin and H. A.Simon, "Why a Diagram is (Sometimes) Worth Ten Thousand Words," *Cognitive Science,* vol. 11, no. 1, pp. 65-100, 1987.

[12] Pete Chapman *et al.*, *CRISP-DM 1.0: Step-by-step Data Mining Guide*. SPSS, 2000.

[13] Pekka Pääkkönen and D. Pakkala, "Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems," *Big Data Research* vol. 2, pp. 166-186, 2015.

[14] D. Sculley *et al.*, "Hidden Technical Debt in Machine Learning Systems," presented at the 28th International Conference on Neural Information Processing Systems (NIPS), Montreal, Canada, 2015.

[15] *Tableau Software: Business Intelligence and Analytics*. Available: https://www.tableau.com/

[16] *Plotly: Modern Visualization for the Data Era*. Available: https://plot.ly/

[17] *Trifacta: Data Wrangling Tools & Software*. Available: https://www.trifacta.com/

[18] *Weka 3 - Data Mining with Open Source Machine Learning Software*. Available: https://www.cs.waikato.ac.nz/ml/weka/

[19] *TensorPort · GitHub*. Available: https://github.com/tensorport

[20] *MLJAR: Platform for Building Machine Learning Models*. Available: https://mljar.com/

[21] *DeepCognition - Become an AI-Powered Organization Today*. Available: https://deepcognition.ai/

[22] *RapidMiner: Data Science Platform*. Available: https://rapidminer.com/

[23] *Watson Machine Learning - Overview | IBM Cloud*. Available: https://www.ibm.com/cloud/machine-learning

[24] *SAS: Analytics, Business Intelligence and Data Management*. Available: https://www.sas.com/en_au/home.html

[25] *KNIME - Open for Innovation*. Available: https://www.knime.com/

[26] *DataRobot: Automated Machine Learning for Predictive Modeling*. Available: https://www.datarobot.com/

[27] *Machine Learning Models & Algorithms | Amazon SageMaker on AWS*. Available: https://aws.amazon.com/sagemaker/

[28] *Tensorflow*. Available: https://www.tensorflow.org/

[29] D. Moody, "Theory Development in Visual Language Research: Beyond the Cognitive Dimensions of Notations," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Corvallis, OR, USA, 2009, pp. 151-154.

[30] J. Liu, "Model-Driven Endpoint Development for Testing Environment Emulation," Doctor of Philosophy, School of Software and Electrical Engineering, Swinburne University of Technology, 2017.

[31] (2015). *The Past, Present, and Future of Machine Learning APIs*. Available: https://www.slideshare.net/bigml/the-past-present-and-future-of-machine-learning-apis

[32] *R: The R Project for Statistical Computing*. Available: https://www.r-project.org/

[33] *Orange – Data Mining Fruitful & Fun*. Available: https://orange.biolab.si/

[34] *Scikit-learn: Machine Learning in Python*. Available: http://scikit-learn.org/stable/

[35] *Apache Mahout*. Available: https://mahout.apache.org/

[36] *Apache Spark™ - Lightning-Fast Cluster Computing*. Available: https://spark.apache.org/mllib/

[37] *IBM SPSS - IBM Analytics*. Available: https://www.ibm.com/analytics/au/en/technology/spss/

[38] W. v. d. Aalst and E. Damiani, "Processes Meet Big Data: Connecting Data Science with Process Science," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 810-819, 2015.

[39] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "Data Scientists in Software Teams: State of the Art and Challenges," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1024-1038, 2018.

[40] R. DeLine and D. Fisher, "Supporting Exploratory Data Analysis with Live Programming," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2015, pp. 111-119.

Hourieh Khalajzadeh received the PhD degree in Computer Science from Swinburne University of Technology, Australia, in 2018. She is currently a research fellow at Monash University, Australia. Her research interests include data analytics, machine learning, deep learning, domain specific visual langiages, data management in distributed systems, and cloud computing. She is a member of IEEE.

Mohamed Abdelrazek received the Ph.D. degree from Swinburne University in 2014. He is currently an Associate Professor in software engineering and IoT with the School of Information Technology, Deakin University, Australia. He has over 10 years' experience in building software solutions. His research interests include software engineering, security, and artificial intelligence.

John Grundy received the BSc (Hons), MSc, and PhD degrees in computer science from the University of Auckland, New Zealand. He is currently Senior Deputy Dean for the Faculty of Information Technology and a Professor of Software Engineering at Monash University, Melbourne, Australia. He is an associate editor of the IEEE Transactions on Software Engineering, the Automated Software Engineering Journal, and IEEE Software. His current interests include domain-specific visual languages, model-driven engineering, large-scale systems engineering, and software engineering education.

John Hosking received BSc (1976) and PhD (1985) degrees from the University of Auckland. Previously he was a lecturer, senior lecturer, associate professor and professor at the University of Auckland and Dean of Engineering and Computer Science at the Australian National University. Currently he is a professor and Dean of Science at the University of Auckland. He is a Fellow of the Royal Society of New Zealand and has produced close to 250 refereed publications. His current interests include domain-specific visual languages, model-driven engineering, and automated software engineering. He is a Member of IEEE and the IEEE Computer Society.

Qiang He received his first Ph. D. degree from Swinburne University of Technology (SUT), Australia, in 2009 and his second Ph. D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2010. He is a senior lecturer at Swinburne University of Technology. His research interests include software engineering, cloud computing, services computing, big data analytics and green computing.