Challenges in Integrating Data Science Pipelines in an End User Software Product

Abstract

In the last few years, we have experienced an increasing use of data science and machine learning techniques in various domain softwares. With the advent of Big Data and recently improved learning algorithms, as seen in the field of deep learning, it became possible to overcome existing limitations on real world applications. Nevertheless, including Big Data and machine learning techniques beneficially in a user-oriented product faces recurring issues naturally given through computational challenges and data complexity.
In the following, we present a data engineering pipeline step by step from raw data to the software ready for deployment. First, we face data quality issues during the data cleaning and integration. Subsequently, the correctness of data transformation, selection and projection to task relevant data depends on domain knowledge and understanding of data. Therefore data exploration is a necessary tool with recurring hand-crafted analytics to achieve valuable insights. A resilient understanding of given data and a robust mathematical representation of domain specific relationships is essential so that the upcoming feature engineering can be tackled efficiently. As well as for the selection of a predictive model, we have to design an appropriate way to evaluate different feature sets and models with regard to the domain requirements.
Finally, we have to combine the complex and computational expensive data science pipeline with a lightweight, user-friendly and responsive front-end. In the end, to unfold the entire value of the software, we need to provide a reasonable visualization and explanation of the prediction outcome.

Data Cleaning

What is Big Data?

When your data sets become so large and diverse that you have to start innovating around how to collect, store, process, analyze and share them

Figure 1: A definition of Big Data given by Martin Grund -Software Engineer at Amazon.

In the beginning, the main challenge is to sum up unorganized, distributed data sources, which do not share standards for same semantic units. Initially, preparing the data for later steps in the process is one of the most time-consuming tasks for data scientists. Moreover domain knowledge is required to organize data semantically correct.
One of the main issues to tackle at data cleaning are missing values and incorrect data. In addition, common sub-tasks are to handle different data formats, to deal with little (labeled) data or data protection issues.

Data Transformation

Feature Engineering

$$Power_t^i = \frac{\left(\frac{M_t^i}{100} * A_M\right) * \left(\frac{n_t^i}{100} * A_n\right)}{9550} * n_t^i \ \text{[kW]}$$

$$Energy = \sum_{Motor\ i=1}^{i=4} \sum_{t_{start}}^{t_{end}} \frac{Power_t^i}{3.6E3} \ \text{[kWh]}$$

M_t^i [%] - Mechanical moment mean value at time t, for engine i
A_M [Nm] - Mechanical moment nominal value
n_t^i [%] - Mean rotational speed at time t, for engine i
A_n [rpm] - Nominal rotational speed

Figure 2 shows a highly domain specific feature. It computes energy consumption during coal mining. It turned out to be the best energy related feature. Often, Data Scientists do not know much about the underlying domain, so that the main considerations were supported by customers. This implies difficulties in evaluating the performance of not only features, but the whole model. Without solid domain knowledge, it is hard to represent customers KPIs to a mathematically reasonable loss or evaluation function. Even if we gained domain knowledge, often it is not trivial to transform real world requirements to a correct mathematical model.

Feature Selection

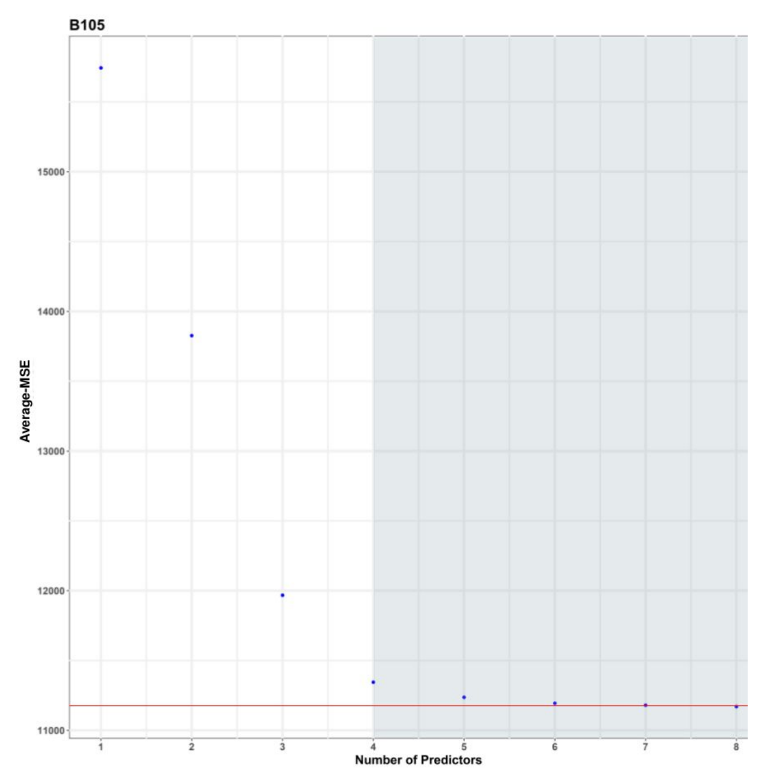
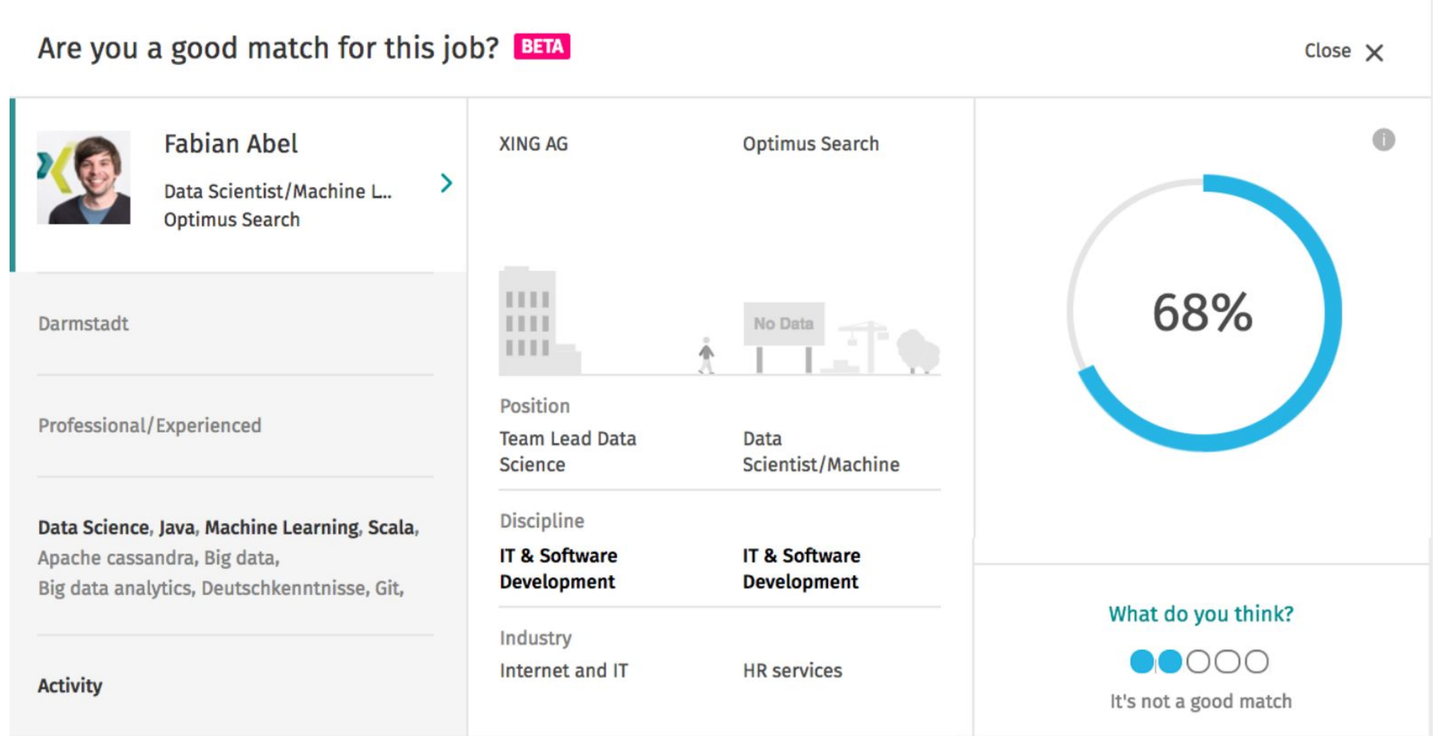


Figure 3 represents an average-MSE curve to pre-select the best feature subset to predict power consumption in coal mining. A brute-force approach over all possible subsets evaluates for each subset its performance through a 10-fold cross validation. This approach is computationally expensive. Moreover as long as features are engineered hand-crafted, we cannot be sure to find an optimal feature space, that supports our prediction model best. We can only evaluate a very limited amount of features, because of engineering workload, computational costs and often a rare understanding of the domain as a Data Scientist. To tackle these issues, we could try to learn representations of features through auto-encoders, but in the end these learned feature spaces are hard to understand, even for a domain expert, so that they are not trustworthy enough for a human level interpretation of our prediction.

Visualization and Explanation of Knowledge


Are you a good match for this job? BETA
Close ×

Fabian Abel
Data Scientist/Machine L...
Optimus Search

Darmstadt

Professional/Experienced

Data Science, **Java**, **Machine Learning**, **Scala**, Apache cassandra, Big data, Big data analytics, Deutschkenntnisse, Git,

Activity

	XING AG	Optimus Search
Position	Team Lead Data Science	Data Scientist/Machine
Discipline	**IT & Software Development**	**IT & Software Development**
Industry	Internet and IT	HR services

Figure 4 shows a job recommendation dashboard on XING. Basic information about a potentially new job are displayed. A small window illustrates a matching score given a job and the user's profile and an opportunity to rate the recommendation. All in all, most space is needed for job information. More interesting to show is the explanation of the score. In order to convince the user why this offer is a good one for her, we could visualize decision points and features contributing most to the prediction. This enables the user to rate the relevance of a certain feature in detail and which decision points are wrong. Another idea is to use knowledge not only regarding the correctly predicted class. We can make use of knowledge implicitly learned about uninteresting offers and could visualize a whole picture of important features and in which way they influence the prediction. In the end, it fits our made effort to engineer such a complex data science pipeline best, if we provide the whole range of knowledge we learned implicitly and explicitly to the end user in order to achieve the best possible information gain and experience with our data science pipeline based software.

Student

Julius Rückin
IT-Systems Engineering (Bachelor)

HPI Hasso Plattner Institut