8-2021

Unboxing the Algorithm: A Process Model of an Algorithmic Solution

Marta Stelmaszak Rosa
Portland State University, stmarta@pdx.edu

Citation Details

Stelmaszak, M. (2021) Unboxing the Algorithm: A Process Model of an Algorithmic Solution. Americas Conference on Information Systems 2021, 9-13 August 2021.

Unboxing the Algorithm: A Process Model of an Algorithmic Solution

Completed Research

Marta Stelmaszak
Portland State University
stmarta@pdx.edu

Abstract

With the explosion of data, analytics and artificial intelligence, information systems research focuses on the use, management and consequences of algorithms. This far, only a handful of papers offer insights into how algorithmic solutions work. To address this gap, we studied the code making up 45 public data science Jupyter notebooks containing algorithmic solutions developed to predict customer churn in a credit card dataset on a data science platform Kaggle.com. We synthesized a process model of an algorithmic solution: preparing the environment, reading in data, cleaning data, exploratory data analysis, pre-processing the dataset, building and training the model, and testing and validating model. Unboxing the algorithm and investigating the process offers a more fine-tuned understanding and language to better conceptualize the use, management and consequences of algorithmic solutions. It also provides a scaffolding for research into the development of algorithmic solutions, highlighting their variability, experimentation and data scientist decisions.

Keywords

Algorithms, algorithmic solutions, data science, information systems development, process model

Introduction

Algorithms have, without a doubt, attracted research attention across a number of fields, from media studies, through sociology, to computer science. Management and information systems (IS) researchers study algorithmic solutions primarily in terms of their use, management and consequences for individuals in work contexts, in organizations, and in the wider society (Galliers et al. 2017; Markus 2017; Newell and Marabelli 2015). However, this research can greatly benefit from an improved understanding of how algorithmic solutions are developed, and thus there have been calls to focus more on theorizing their development (van den Broek et al. 2021). This far, only a handful of papers in IS offer insights into how algorithmic solutions work which is an essential link between understanding their use and their development. Against this background, this study aims to answer a simple question: what is the process of making an algorithmic solution work?

To uncover the building blocks and propose a process model, we studied 45 public data science Jupyter notebooks containing algorithmic solutions developed to predict customer churn in a credit card dataset on a popular data science and machine learning platform Kaggle.com (Dissanayake et al. 2015; Mangal and Kumar 2016). Referring to a common problem faced by many companies and often tackled by algorithmic solutions, the credit card dataset attracted over 200 notebooks with code and comments describing attempts to best predict customer churn. We selected 35 of the best-regarded notebooks, downloaded them and coded them using a grounded theory approach (Charmaz 2006; Glaser and Strauss 1967; Urquhart and Fernandez 2006). We then grouped the themes to identify the elements that made up each proposed algorithmic solution and distilled a process model of how they were developed.

Based on our findings, we propose a process model of making an algorithmic solution work encompassing: preparing the environment, reading in data, cleaning data, exploratory data analysis, pre-processing the dataset, building and training the model, and testing and validating the model. We contribute to information systems and management literature by developing a process model of an algorithmic solution that offers a more fine-tuned language to investigate not only the use, management and consequences of algorithmic solutions on individuals, organizations and societies, but also enables a further study of the design and development of such solutions from a socio-technical perspective.

Making Algorithmic Solutions Work

Recent technological (processing capabilities, big data, machine learning), societal (use of smartphones, attitudes towards data, social media) and organizational (phantomization, networks) developments contributed to the growth in use of various algorithms (Baptista et al. 2017; Berente et al. 2019). IS research in the socio-technical tradition has thus focused on the study of the use, management and consequences of algorithms on individual, organizational and societal levels (Galliers et al. 2017; Markus 2017; Newell and Marabelli 2015). However, far less attention has been paid so far to the understanding of how algorithms and algorithmic solutions based on them are developed (van den Broek et al. 2021). First papers begin to uncover how data scientists and subject matter experts need to work together in the development process (van den Broek et al. 2021), how the practices of data scientists in the banking industry rely on both subjectivity and objectivity in the production of information (Joshi 2020), and how data scientists engage in the practices of knowledge hiding (Ghasemaghaei and Turel 2021). In other words, while focusing predominantly on what happens after the algorithms are put to work, current literature offers little insight into how algorithms are *made* to work, that is what steps need to be in place for an algorithmic solution to work effectively. Such understanding is essential because the process of making an algorithmic solution work, as we show below, determines what kinds of insights and predictions it offers, thus influencing decisions.

Most researchers who in broad strokes describe what goes into making algorithmic solutions work in their papers refer to certain aspects with varying consistency: the fact that algorithms process data (Balasubramanian and Ye 2021; van den Broek et al. 2020; Galliers et al. 2017; Gregory et al. 2020; Grønsund and Aanestad 2020; Lebovitz 2020; Lycett 2013; Newell and Marabelli 2015; Pachidi et al. 2021; Shrestha et al. 2019) in an automated or preprogramed way (Galliers et al., 2017; Grønsund & Aanestad, 2020; Günther et al., 2017; Shrestha et al., 2019) to learn models (Balasubramanian and Ye 2021; Li et al. 2019; Shrestha et al. 2019) leading to new insights (Günther et al., 2017; Günther & Joshi, 2020; Pachidi et al., 2021), decisions (Balasubramanian and Ye 2021; van den Broek et al. 2020; Galliers et al. 2017; Newell and Marabelli 2015) or predictions (Lebovitz 2020; Li et al. 2019; Shrestha et al. 2019). This offers a punctuated and incomplete picture of the elements involved in developing algorithms that can be subsequently used in business settings.

A handful of papers offer insights into the essential elements of what happens inside algorithmic solutions. Pachidi et al. (Pachidi et al. 2021) provide a detailed description, covering various elements that are at play in a predictive model:

> “The model combined a number of internal and external data sources, such as time series of customer transactions, Nielsen market data, Gartner ICT spending predictions, financial data, and usage data. The output of the model was represented in a spreadsheet format that contained a list of all medium-sized customers and predictions regarding potential sales opportunities. The CLM model allocated customers to different customer segments (A, B, C, D) based on their historical and predicted sales with TelCo. For each TelCo product line (e.g., business telephone systems, mobile phone packages, fixed line setups), the CLM model assigned a position in the customer sales lifecycle (inform, specify, sell, maintain), each of which entailed a different contact strategy. Thus, the model output consisted of a ranking of opportunities, with a prioritized action list for account managers.”

Grønsund and Aanestad (2020, p. 7) are similarly detailed:

> “The algorithm-supported analysis system was designed to automate both data acquisition and the processing of data for subsequent analysis. Acquisition of data was automated by the system pulling streams of data on ship activity from the satellite-AIS data provider, along with additional data such as vessel descriptions and geospatial data, into a Hadoop-based data warehouse repository. Here the data were extracted and consolidated, then classified using rule-based NLP (Natural Language Processing) classification, and finally presented in BI tools that allowed human interpretation of the output.”

While the descriptions both point to obtaining, compiling and processing of data, further analysis and classification, they reveal differences in how the solutions work, and do not offer a complete picture. Taking a more general view, Orlikowski and Scott define an algorithm as "a set of step-by-step instructions to achieve a desired result in a finite number of moves" (2015, p. 210). Acknowledging this more traditional definition of an algorithm - a program containing a fixed sequence of instructions executed until a solution is reached - rooted in computer science (Hopcroft and Ullman 1983), Faraj et al. (2018) 'update' and broaden the scope of this definition by conceptualizing learning algorithms as "an emergent family of technologies that build on machine learning, computation, and statistical techniques, as well as rely on large data sets to generate responses, classifications, or dynamic predictions that resemble those of a knowledge worker" (p. 62). A similar definition of artificial intelligence algorithms is put forward by Tarafdar et al. (2020, p. 1): "We define AI algorithms as those that extract insights and knowledge from big data sources; computational and statistical techniques such as machine learning (ML) and deep learning embedded in such algorithms, aim to 'teach' computers the ability to do detect patterns in big data".

While these definitions offer a good starting point and an initial overview of the elements in the process of making algorithmic solutions work, they are partial and divergent in their focus. These differences in the definition, understanding, scope and scale of the steps and elements required to make algorithms work hamper the development of the understanding of the use, management and consequences of algorithms, and at the same time make uncovering their development more difficult. For IS research to systematically progress in this area it is thus fundamental to ask: *what is the process of making an algorithmic solution work*?

Research Setting and Methods

To answer this question, we studied 45 public data science notebooks containing algorithmic solutions developed to predict customer churn in a credit card dataset on a popular data science and machine learning platform Kaggle.com. Below we describe the research setting, as well as data collection and analysis methods.

Research Setting

Kaggle.com is a popular platform for data scientists and machine learning engineers where they can develop and improve their skills, as well as participate in corporate-sponsored competitions by addressing a variety of problems related to datasets published on the platform. Kaggle.com, part of Alphabet Inc, allows to upload datasets, set specific tasks for them and create interactive Jypyter notebooks where users can develop their algorithmic solutions. Kaggle.com was selected as a setting because of its public availability and openness in sharing notebooks that allows an unprecedented access to the inner workings of algorithmic solutions. Others have used Kaggle.com for research purposes as well (Dissanayake et al. 2015; Mangal and Kumar 2016).

The dataset we selected for this study is a well-regarded and popular dataset with high usability. It contains the details of around 10,000 credit card customers of a bank, whereby a portion of customers churned. The goal is to identify, based on 18 variables such as age, salary, credit card limit and similar, what makes a customer churn (give up a credit card) to be able to predict customers at risk of churning in the future, as well as to identify the variables that are most predictive of the risk of churn ("Kaggle.Com" 2021). When the dataset was investigated for the purposes of this research project, there were around 210 notebooks submitted that contained algorithmic solutions pertaining to this dataset, with constant daily activity in existing notebooks and new notebooks being added.

We selected an open and public dataset rather than a competition because the majority of notebooks submitted for competitions are private and thus visible only to sponsor companies, and competitions are usually very specific and limit the number of potential algorithmic solutions applied. In contrast, public notebooks allow good access to a variety of notebooks containing fairly unrestricted solutions and allow for much more experimentation on the part of users. From the many datasets available on Kaggle.com, we selected the credit card customers dataset because it is related to a common problem that many companies and businesses face, and it is a problem that is often tackled by developing algorithmic solutions, thus it is a good representative sample of what researchers in information systems and management would consider of interest.

Data collection

In January and February 2021, we collected 57 Jupyter notebooks that were created using the credit card customer dataset in Python as the set programming language. The notebooks were arranged from the 'hottest' (a measure used on Kaggle.com to define notebooks with most activity, edits and highest votes by the community, *Kaggle.Com*, 2021) to the least hot, and thus those that we collected were considered among the 'hottest' at the time. We decided to select the 'hottest' notebooks as these were assessed as high quality by the community, thus were likely to contain well-developed algorithmic solutions. We discarded notebooks in R to eliminate differences in programming languages, and notebooks that contained only partial solutions, for example only analyzed data without building actual models. We ended up with 45 suitable notebooks. Using a feature available on Kaggle.com, we downloaded all of the selected notebooks and converted them to PDF documents to analyze them in nVivo.

Data analysis

Since our study is rooted in grounded theory (Charmaz 2006; Glaser and Strauss 1967; Urquhart and Fernandez 2006), we proceeded by inductively coding the notebooks to identify the different elements of code they contained by what these elements of code did. We coded each segment of code in each notebook to identify its function. Verbal descriptions of data scientists sometimes provided additional information as to the role of each code segment, so these were coded too. However, the descriptions were mostly useful in the second stage of data analysis, where we grouped the codes we obtained into higher-level elements of the process, as they explained the flow of the process. For example, in the notebooks data scientists would sometimes indicate they were proceeding to exploratory data analysis, and we used these comments to group elements of code identified under the element 'Exploratory Data Analysis'.

Because of the inductive nature of our study, we oscillated between data analysis and further data collection. After coding the first 30 notebooks, we began to group the codes to start building the model. We then proceeded with coding and analyzing notebooks one by one to supplement and verify the model that was emerging from our analysis. When we reached notebook number 35, the subsequent 10 notebooks did not add any new codes to the codebook and at this point we decided to stop coding and analyzing the notebooks as we reached the point of saturation.

Unboxing the Algorithm

In this section, we present the elements of the process of making an algorithmic solution work that we identified in the data. Each element is discussed in turn by showing what kinds of operations were performed in every element.

Preparing the Environment

Notebooks begin with setting the environment in which the development of the algorithmic solution takes place: programming language, acceleration and connection to the internet. The notebooks we observed were all set up in a Python 3 environment, which "*comes with many helpful analytics libraries installed*" (Notebook 002) and allows to write up to 20 GB to the working directory. Notebooks give the possibility to turn on an accelerator, such as a GPU, for faster processing, and to connect to the internet for access to external files. In some notebooks, data scientists use verbal comments to identify and restate the problem.

After this initial setup, various necessary libraries are imported, that is pre-packaged functions designed for specific purposes that can be deployed by data scientists without the need to code such functions from scratch. Invariably, the notebooks featured "*numpy*" (Notebook 005), a Python library for linear algebra and "*pandas*" (Notebook 007) allowing for data processing and for example reading in CSV files, among others. These two libraries are essential to develop the algorithmic solution. Other libraries imported include data visualization packages, such as "*seaborn*" or "*matplotlib*" (Notebook 029), which are fairly standard and popular libraries for this purpose. In some notebooks, all required packages are imported in the beginning of the notebook, including "*sklearn*" and "*keras*" (Notebook 014) that are used for building models, while other notebooks import additional libraries as and when needed. Libraries are imported with simple code: "*import numpy as np*" (Notebook 001), for example. Importing libraries is a standard procedure and there are not substantial comments regarding this step. There exists a variety of libraries used in developing algorithmic solutions that are widely used, and they encapsulate and abstract out the complexity behind such tasks like training a specific model, as explained below.

Reading in Data

The next element in the process in an algorithmic solution is to read in the required data. The first step here, quite logically, includes loading data in. Because the dataset that the notebooks use is uploaded to Kaggle, it can be attached to each notebook with a simple search within the interface, and then imported by executing a command from the "*pandas*" library "*read_csv*" (Notebook 001).

Inspecting the data follows, usually through function "*head*", displaying first five (by default) rows of the dataset and corresponding columns with column headers, and sometimes function "*shape*" displaying the dimensions of the dataset (number of rows and number of columns) as well as function "*columns*", giving the names of columns in the dataset. In just one notebook, we observed explicitly looking for duplicate entries in the dataset. Commands to perform these functions are pre-packaged and take forms of "*df.head()*", "*df.shape*" or "*df.columns*" (Notebook 003). This stage of the process also involves checking data types present in the dataset, performed by using functions "*info*" or "*dtypes*" that indicate which columns contain integer (whole numbers), float (fractions with decimal points) or object (text or mixed numeric and non-numeric values) data type. This is important as most algorithmic solutions work only with numerical values. As part of reading in data, simple descriptive statistics of the data are obtained through function "*describe*", resulting in displaying the number of rows, mean, standard deviation, minimum value, quartiles, and maximum value for each column.

Conducting the these steps is essential to load the dataset and obtain basic information about the data needed to confirm that the data is loaded correctly, contains the expected columns and rows, and to gain initial familiarity with the dataset.

Cleaning Data

After reading in the dataset, data is cleaned to prepare it for further processing. This is essential before any analysis can take place. Steps at this stage tend to be taken in various orders across the notebooks, and are reported here in no particular order.

Missing values are identified and dealt with: that is NULL values in the dataset have to be resolved before any analysis can take place. This is done by using the function "*isnull*", listing all columns with the number of missing values (Notebook 001). The customer churn dataset contained no null values, so in this case there was no need to deploy solutions to solve this problem. Missing values have to be resolved as the majority of algorithms cannot deal with datasets containing missing values. One of the ways to solve this problem that is presented in the notebooks is the method of imputation, that is replacing the missing or null value with an existing value from the dataset. In the solution proposed in the notebook this is done based on the nearest neighbor of the missing value, but since no missing values were detected, the solution is not implemented.

In the notebooks, we found sometimes columns are renamed if their names are not intuitive enough or simply too long. Certain columns containing variables that are not needed for the analysis are removed. For example, the customer churn dataset contains two columns with Naïve Bayes Classifier by default, and the author of the dataset suggests removing these columns before proceeding with analysis. At different points in data cleaning, exploratory data analysis or pre-processing the dataset various columns are also removed if they are not contributing to the model (for example, removing customer ID: "*data=data.drop(columns=['CLIENTNUM']*", Notebook 015). In some notebooks, outliers are removed from the data using a common statistical measure of z-score, indicating how far from the mean a given data point is. In the only notebook we observed that removed outliers, this resulted in removing 810 rows from the dataset.

All notebooks we studied transform data types as part of cleaning data. This step, sometimes referred to as feature engineering, is required when the dataset contains object data types, which are categorical variables typical in many datasets, such as marital status, level of education or gender. These data types have to be transformed into numerical variables in order to be analyzed. This is conducted by using pre-existing functions to encode these variables as integers (e.g. primary education as 1, secondary as 2, tertiary as 3) or using popular one-hot encoding where there is no natural ordinal relationship between categories and dummy variables are created (e.g. male is 0, female is 1). Cleaned data is an essential element of any algorithmic solution, as without the steps taken in this element, data either results in erroneous analysis and model training, or simply cannot be used to train models.

Exploratory Data Analysis

The next step in the algorithmic solution process is exploratory data analysis, whereby actions are taken to learn about the relationship between the dependent variable of interest (here: customer churn or attrition) and independent variables that may help build the predictive model. This step is essential to uncover what model will be the most appropriate for the dataset and which variables can be potentially of interest.

The first step is to identify the dependent variable (a trivial matter in the given dataset), and to analyze independent variables. This is very often performed by visualizing them independently, in relation to each other, or in relation to the dependent variable. In most cases, such visualizations were implemented using functions from visualization libraries, such as "*seaborn*", "*matplotlib*" or rarely "*plotly*". Visualizing data is the part that takes up the most code in nearly all notebooks we analyzed. Various visualizations are produced, such as box plots, pie charts, histograms, in order to help identify which variables may be useful in building the model. Visualizations are often accompanied by comments such as "*Females are slightly more likely to churn with 17% compared to males with 15%, we'll convert this 9 feature to 1-0*" (Notebook 013). Some notebooks contain more comprehensive comments on the learnings from visualizations.

The next step in exploratory data analysis is to identify correlations between variables. Identifying correlations is an important step in exploratory data analysis, as from this decisions can be made as to which features to include in pre-processing the dataset for model building, as described below. For example, Notebook 022 based on the identification of correlations decides to *"#Drop some features which have less than 0.01 correlation and greater than -0.01 correlation*". Exploratory data analysis is a required step of building an algorithmic solution as it provides the necessary insight into the dataset for the purposes of model building. It is at this stage that the importance of variables with respect to the target variable is assessed.

Pre-processing the Dataset

The following step in the process is to pre-process the dataset, which involves preparing the dataset according to the requirements of model building. First, data need to be scaled, which may involve actual scaling, that is changing the range of variables to a common range, e.g. between 0 and 1, or normalizing the variables following a normal distribution. Scaling is performed to ensure that no variable is interpreted as more predictive than it actually is just because its numerical values are on a different from other variables. Scaling is routinely performed using standard pre-packaged functions, such as "*StandardScaler*" from the popular "*sklearn*" library (Notebook 026).

The dataset should be resampled if it is not balanced, that is if one category is present much more frequently than another. In the case of the dataset investigated, customers who attired occurred much less frequently, as identified in exploratory data analysis, so resampling was required. This is usually done by oversampling from the group of attired customers, most frequently using a pre-packaged function 'SMOTE' (Synthetic Minority Oversampling Technique) which creates additional data points to increase the number of observations in the less frequent category. Alternatively, undersampling can be performed, that is reducing the number of observations of non-attired customers. In these cases, two separate datasets are retained for model building and validation, and later evaluated for better predictive results.

Formatting the dataset is a common step when pre-processing the dataset for model building. This step encompasses concatenating datasets together (e.g. adding columns with dummy variables) or dropping other unnecessary columns. Importantly, at this stage columns carrying the attrition flag (whether customer attired or not) are removed from the training dataset – this is the target variable for the identification of which the model is later be trained.

Finally, pre-processing the dataset involves splitting it into training (data that the model is trained on) and test (data that the trained model's performance is tested on) datasets. This is an essential step and it is present in all notebooks studied. It is performed using a pre-packaged function which allows to decide the percentage of the dataset retained for testing. This value varied from 0.2 to 0.4 in the notebooks studied. The function is executed in a simple line of code: "*X_train, _X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 25)*" (Notebook 012). Following this step, the datasets are ready for model building.

Building and Training the Model

The model building stage contains a number of optional steps. Model building may start with setting up a pipeline, that is a set of instructions that are to be applied to a number of models selected (e.g. to automatically deploy the model first on the training and then on the test dataset). In some cases, the pipeline includes steps that otherwise belong to the pre-processing of the dataset, such as transforming data types or resampling the dataset. Pipelines are usually created using a pre-packaged "*pipeline*" function from "*sklearn*" (Notebook 028), but they can be coded from scratch. Another step that is present in the minority of notebooks is parameter selection, whereby baseline parameters for the models are set manually, for example the number of layers in a decision tree, learning rate, or number of estimators. In the majority of notebooks, parameters were not defined but left to the defaults pre-set in models applied.

Since these two steps are optional and present only in the minority of notebooks, most notebooks introduce model building and training right after pre-processing the dataset. Model building involves selecting and creating an instance of the actual algorithm that subsequently processes data iteratively. In all notebooks studied, such algorithms – commonly referred to as models in the data science community – are deployed by importing them from pre-existing libraries. For example, in Notebook 001 we observe "*from sklearn.linear_model import LogisticRegression // logmodel = LogisticRegression()*", whereby the algorithm for logistic regression is imported from the "*sklearn*" library in one line of code, and then invoked as "*logmodel*" in another line of code. Thus "*logmodel*" is an instance of the logistic regression algorithm that is created in the notebook. In the next line of code, this model is trained: "*logmodel.fit(X_train_pd,y_train)*" – the instance of the logistic regression algorithm is "*fit*" to the training dataset. After this operation, the model would have been trained, that is it learned how the predictive variables are correlated with the target variable.

Few notebooks contain only a single model selected for the algorithmic solution, while more commonly notebooks contain several models that are trained subsequently and their performance is later compared. In both scenarios, data scientists import relevant models from pre-existing libraries and apply them to the pre-processed dataset in one or at most few lines of code. Since the dataset we studied contained a supervised classification problem, all models used in the notebooks were applicable to this type of a predictive problem. The types of models we observed were some simpler and common ones, such as logistic regression, a k-nearest-neighbor classifier, Naïve Bayes or decision tree classifier, to more advanced classifiers such as AdaBoost, Bernoulli NB, Extra Trees Classifier, Gradient Boosting Classifier, LGBM Classifier, Principal Component Analysis, Random Forest Classifier, Support Vector Classifier or XGBoost classifier. One notebook experimented with using a neural network for this task. Regardless of the type of the model selected, all these were implemented using pre-packaged functions.

Depending on model performance, experiments may be run with different models or different parameters. Changing parameters is often referred to as parameter tuning, and is usually done by using pre-packaged functions or writing up own functions to train a large number of models with different parameters and compare their performance to select the best set of parameters. Developing different models can involve selecting a different model (that is, a different algorithm) and training it on the same dataset or performing so-called feature selection or dimensionality reduction, where a pre-packaged function is used to identify the variables that contribute the most to obtaining predictions and keep these in the model, while discarding other variables that could potentially negatively impact model performance. Running experiments with different models or parameters is optional and not present universally across the notebooks. However, model building and training are an essential element of developing an algorithmic solution.

Testing and Validating the Model

The final stage is model validation or testing, whereby the results of an algorithmic solution are evaluated. When various models are developed, the steps in model testing are often interspersed with model building, as this allows to quickly compare the results of different models. After training relevant models, they are then applied to the unseen portion of the data, that is to the test dataset withheld earlier when the dataset was pre-processed. This is customarily done in one line of code, e.g. "*ypred = model_1.predict(X_test)*" (Notebook 017), whereby the trained model "*model_1*" is applied onto "*X_test*" dataset to yield predictions saved in "*ypred*". The results of this prediction are then used to assess the performance of the model.

All notebooks contain a way to evaluate model performance. This is usually done both for the training and test dataset, albeit it is commonly accepted to use the performance results on the test dataset as final. The most common measure of model performance in the notebooks is accuracy, that is how many times the model predicted the correct classification of the dependent variable in comparison to the known but withheld labels in the test dataset. The second common measure in this dataset is recall, that is how often the model correctly identified churning customers – this measure is particularly relevant to the problem set in the dataset. Sometimes models are evaluated based on the time it took to train them. Evaluating model performance is done using pre-packaged functions, most often imported from "*sklearn*", for example: "*from sklearn.metrics import classification_report, confusion_matrix // print (classification_report(y_test,predictions)) // print(confusion_matrix(y_test,predictions))*" (Notebook 011). In notebooks where various models are trained, it is customary to evaluate and compare the performance between different models.

In many notebooks, evaluating and comparing performance is also represented visually, using a number of graphs and diagrams to show how the models performed. Visualizing results is often complementary to showing numerical performance results. In some notebooks, the final step in this element includes indicating clearly and verbally the best performing model, e.g. "*Out of all Random Forest outperformed other algorithms with accuracy of 97%*" (Notebook 002) or "*We can see that combining the features generated with PCA with the others is what gives the better results, this can be due to higher degrees of freedom for the model*" (Notebook 028). A large number of notebooks do not make any comments on the best performing model, either because there is only one model trained or because the numerical results are considered as self-explanatory.

Process Model of an Algorithmic Solution

This research project set out to uncover the process of making an algorithmic solution work. The investigation of data science notebooks with algorithmic solutions in response to a credit card customer churn problem helped us identify the elements involved in the process of developing an algorithmic solution. Our investigation showed that algorithmic solutions entail a number of steps in the process of making them work: preparing the environment, reading in data, cleaning data, exploratory data analysis, pre-processing the dataset, building and training the model, and testing and validating model. All these elements need to be present in this order for an algorithmic solution to work. The model summarizing our findings is presented below.

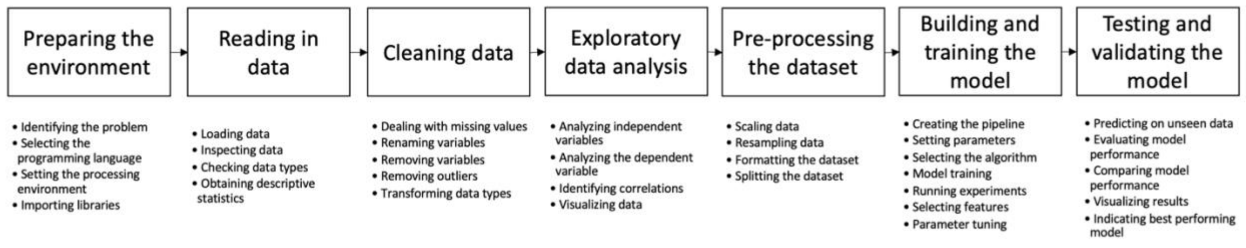
Preparing the environment
• Identifying the problem
• Selecting the programming language
• Setting the processing environment
• Importing libraries
Reading in data
• Loading data
• Inspecting data
• Checking data types
• Obtaining descriptive statistics
Cleaning data
• Dealing with missing values
• Renaming variables
• Removing variables
• Removing outliers
• Transforming data types
Exploratory data analysis
• Analyzing independent variables
• Analyzing the dependent variable
• Identifying correlations
• Visualizing data
Pre-processing the dataset
• Scaling data
• Resampling data
• Formatting the dataset
• Splitting the dataset
Building and training the model
• Creating the pipeline
• Setting parameters
• Selecting the algorithm
• Model training
• Running experiments
• Selecting features
• Parameter tuning
Testing and validating the model
• Predicting on unseen data
• Evaluating model performance
• Comparing model performance
• Visualizing results
• Indicating best performing model

Figure 1 Model of an algorithmic solution

Discussion and Conclusions

The model we synthesize from the research conducted provides a more complete and comprehensive picture of the process that algorithmic solutions follow from the moment datasets are identified, to the point at which these solutions offer predictions. Our analysis revealed several important contingencies in the process of making algorithmic solutions work.

First, there is a high degree of optionality in terms of how each step of the process unfolds. While the process has to follow the stages we identified in this order for the algorithmic solution to work, what actually happens at each stage varies. There are actions that seem compulsory, and others that are optional, as they were present in some but not all notebooks we studied. This introduces variability around the process, and induces changes in the outputs at all stages. Interestingly, notebooks did not conclude by offering the same solutions with the same accuracy, thus suggesting that variability in the process results in differences in outcomes. Further research is needed to investigate potential explanations of this variability and how optionality impacts the outcomes.

Second, we uncovered that the algorithm *per se* is only a small part of the entire algorithmic solution, and one that is concealed in a packaged library, invoked with one line of code, and applied mechanistically to the dataset. In this sense, an algorithm is indeed a set of step-by-step instructions that may take the form of a nearest-neighbor or logistic regression classifier where the algorithm performs specific calculations until a stopping condition is reached. In comparison to the complexity of the other elements, the algorithm is an unproblematic, out-of-the box solution. However, the training of the algorithm involves the entire process as we described.

Third, our findings indicate that at every stage and in every element, human reasoning and decision-making is involved. Data scientists make individual decisions, from what libraries to import, through what descriptive statistics to pay attention to, to how to resample, format and split the dataset, to which algorithm to apply. At the same time, their design choices are heavily constrained and limited by the environment, data, and the algorithm they choose to work with. For example, the choice of the algorithm will influence how data are cleaned and what form the pre-processed dataset takes. Further, we discovered that the process can be characterized by a high degree of experimentation, whereby data scientists test and develop various models and compare their performance. More research is needed into the understanding of how data scientists make such decisions in the process of developing algorithmic solutions.

The model we present can be fruitfully used to organize and synthesize extant IS research around the use, management and consequences of algorithms. A processual understanding of the workings of algorithmic solutions offers a more detailed way of discussing what exactly is used, has to be managed and results in certain consequences. For information systems, our model offers a fruitful avenue for further study. First and foremost, we offer a model that unifies a number of perspectives and thus can allow research to progress. Second, all elements and steps in themselves give rise to a plethora of questions of relevance to the discipline. Finally, by providing a process model of an algorithmic solution, we facilitate further IS research that goes beyond the use, management and consequences of algorithmic systems. We provide a starting point for IS studies into the design and development of such systems. Unboxing the algorithm makes it possible to study how these systems are made, with information systems research having a unique focus on the socio-technical aspects of developing algorithmic solutions.

REFERENCES

Balasubramanian, N., and Ye, Y. 2021. "Substituting Human Decision-Making with Machine Learning: Implications for Organizational Learning," *Academy of Management Review* (Forthcomin).

Baptista, J., Stein, M.-K., Lee, J., Watson-Manheim, M. B., and Klein, S. 2017. "Call for Papers: Strategic Perspectives on Digital Work and Organizational Transformation," *Journal of Strategic Information Systems* (26:4), i–iii.

Berente, N., Gu, B., Recker, J., and Santhanam, R. 2019. "Managing AI," *MIS Quarterly*, pp. 1–5.

van den Broek, E., Sergeeva, A., and Huysman, M. 2020. "Hiring Algorithms: An Ethnography of Fairness in Practice," *40th International Conference on Information Systems, ICIS 2019*.

van den Broek, E., Sergeeva, A., and Huysman, M. 2021. "When the Machine Meets the Expert: An Ethnography of Developing AI for Hiring," *MIS Quarterly*, pp. 1–50.

Charmaz, K. 2006. *Constructing Grounded Theory*, Thousand Oaks, CA: SAGE Publications.

Dissanayake, I., Zhang, J., and Gu, B. 2015. "Task Division for Team Success in Crowdsourcing Contests: Resource Allocation and Alignment Effects," *Journal of Management Information Systems* (32:2), pp. 8–39.

Faraj, S., Pachidi, S., and Sayegh, K. 2018. "Working and Organizing in the Age of the Learning Algorithm," *Information and Organization* (28:1), Elsevier, pp. 62–70.

Galliers, R. D., Newell, S., Shanks, G., and Topi, H. 2017. "Datification and Its Human, Organizational and Societal Effects: The Strategic Opportunities and Challenges of Algorithmic Decision-Making," *Journal of Strategic Information Systems* (26:3), pp. 185–190.

Ghasemaghaei, M., and Turel, O. 2021. "Possible Negative Effects of Big Data on Decision Quality in Firms: The Role of Knowledge Hiding Behaviours," *Information Systems Journal* (31:2), pp. 268–293.

Glaser, B. G., and Strauss, A. L. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research*, New York: Aldine.

Gregory, R. W., Henfridsson, O., Kaganer, E., and Kyriakou, H. 2020. "The Role of Artificial Intelligence and Data Network Effects for Creating User Value," *Academy of Management Review*, pp. 1–40.

Grønsund, T., and Aanestad, M. 2020. "Augmenting the Algorithm: Emerging Human-in-the-Loop Work Configurations," *Journal of Strategic Information Systems* (29:2), Elsevier, p. 101614.

Günther, W. A., Rezazade Mehrizi, M. H., Huysman, M., and Feldberg, F. 2017. "Debating Big Data: A Literature Review on Realizing Value from Big Data," *Journal of Strategic Information Systems* (26:3), pp. 191–209.

Günther, W., and Joshi, M. P. 2020. "Algorithmic Intelligence in Research: Prevalent Topic Modeling Practices and Implications for Rigor in IS and Management Research," *ICIS 2020 Conference Proceedings*, pp. 1–9.

Hopcroft, J. E., and Ullman, J. D. 1983. *Data Structures and Algorithms*, Boston, MA: Addison-Wesley.

Joshi, M. P. 2020. "Custodians of Rationality: Data Science Professionals and the Process of Information Production in Organizations," *AMCIS 2020 Proceedings*, pp. 0–1.

"Kaggle.Com." 2021. *Kaggle.Com.* [Accessed on: 15 January 2021]. Available at: www.kaggle.com.

Lebovitz, S. 2020. "Diagnostic Doubt and Artificial Intelligence: An Inductive Field Study of Radiology Work," *40th International Conference on Information Systems, ICIS 2019*.

Li, K. G., Mithas, S., Zhang, Z., and Tam, K. Y. 2019. "How Does Algorithmic Filtering Influence Attention Inequality on Social Media?," *Proceedings of International Conference on Information Systems 2019*, pp. 1–9.

Lycett, M. 2013. "'Datafication': Making Sense of (Big) Data in a Complex World," *European Journal of Information Systems* (22:4), pp. 381–386.

Mangal, A., and Kumar, N. 2016. "Using Big Data to Enhance the Bosch Production Line Performance: A Kaggle Challenge," *Proceedings of 2016 IEEE International Conference on Big Data*.

Markus, M. L. 2017. "Datification, Organizational Strategy, and IS Research: What's the Score?," *Journal of Strategic Information Systems* (26:3), pp. 233–241.

Newell, S., and Marabelli, M. 2015. "Strategic Opportunities (and Challenges) of Algorithmic Decision-Making: A Call for Action on the Long-Term Societal Effects of 'datification'," *Journal of Strategic Information Systems* (24), Elsevier B.V., pp. 3–14.

Orlikowski, W. J., and Scott, S. V. 2015. "The Algorithm and the Crowd: Considering the Materiality of Service Innovation," *MIS Quarterly* (39:1), pp. 201–216.

Pachidi, S., Berends, H., Faraj, S., and Huysman, M. 2021. "Make Way for the Algorithms: Symbolic Actions and Change in a Regime of Knowing," *Organization Science* (32:1), pp. 18–41.

Shrestha, Y. R., Ben-Menahem, S. M., and von Krogh, G. 2019. "Organizational Decision-Making Structures in the Age of Artificial Intelligence," *California Management Review*, pp. 66–83.

Tarafdar, M., Teodorescu, M., Tanriverdi, H., Robert Jr., L. P., and Morse, L. 2020. "Seeking Ethical Use of AI Algorithms: Challenges and Mitigations," *Proceedings of the 41th International Conference on Information Systems, December 13-16, India.*, pp. 0–7.

Urquhart, C., and Fernandez, W. 2006. "Grounded Theory Method: The Researcher as Blank Slate and Other Myths," in *Proceedings of the 27th International Conference on Information Systems*, Milwaukee, WI.