Teach and Try: A simple interaction technique for exploratory data modelling by end users

Advait Sarkar
Alan F Blackwell
Mateja Jamnik
Computer Laboratory, University of Cambridge
Cambridge, United Kingdom
{advait.sarkar, alan.blackwell, mateja.jamnik}@cl.cam.ac.uk

Martin Spott
BT Research and Technology
Adastral Park, Ipswich
martin.spott@bt.com

***Abstract*—The modern economy increasingly relies on exploratory data analysis. Much of this is dependent on data scientists – expert statisticians who process data using statistical tools and programming languages. Our goal is to offer some of this analytical power to end-users who have no statistical training through simple interaction techniques and metaphors. We describe a spreadsheet-based interaction technique that can be used to build and apply sophisticated statistical models such as neural networks, decision trees, support vector machines and linear regression. We present the results of an experiment demonstrating that our prototype can be understood and successfully applied by users having no professional training in statistics or computing, and that the experience of interacting with the system leads them to acquire some understanding of the concepts underlying exploratory statistical modelling.**

I. Introduction

There are many situations in which end-users need to perform simple exploratory and interactive data analysis. Typical examples include the analysis of trends in historical data in order to predict future values, estimation of missing data from a data set, or "sanity checking" of new data by comparison to known information. In this paper, we present a novel tool that supports these kinds of operations within a spreadsheet-like interaction paradigm.

Formally, the problem being addressed is mixed-initiative statistical inference or machine learning. Here, the goal is to construct a model that characterises a multivariate training data set, and then apply that model to a test data set to estimate missing values or gauge the correctness of preexisting values. Specialised software packages such as scikit-learn [5], programming languages such as R [6], and deep domain expertise are typically needed in order to build and apply such models. In contrast, our tool only requires the user to be familiar with simple spreadsheet manipulation operations.

II. Our interaction technique

Users mark a range of cells to indicate that they have confidence in those values. The marked cells are used to train a statistical model. Once the training set has been specified in this manner, the user can select any other range of cells, potentially overlapping with the training set, in order to apply the model to those cells. If cells in the new selection are empty, they will be filled in. Otherwise, they will be coloured according to their deviation from the model prediction.

This sequence of operations is illustrated in Figure 1. The user first makes a selection of a number of rows, and then clicks on the "Teach" button. At this point, the selected rows are added to a training dataset. The cells are visually marked as "taught" by colouring the text green, colouring the background a faint green, and placing a green check mark icon in the cell.

Next, the user selects cells from a single column and clicks on the "Try" button, as in panels 3 and 4 of Figure 1. The variable in the selected column is now interpreted as the *target* or *dependent* variable for the statistical model, and the variables in the unselected columns are interpreted as the *feature vector* or *independent* variables. The software interprets cell selection bounds as parameters for the model, and this allows the user to build and apply a relevant model with a single interaction. This is contingent on the data in the spreadsheet being laid out in a well-defined relational schema. Upon clicking the "Try" button, the software trains a statistical model on the rows of data previously taught, and applies the model to the rows containing the current selection. If the cells are empty, then the model's predictions are used to populate the cell contents. If cells in the new selection already contain values, as in Figure 2, their values are not altered. Instead, they are coloured on a red-green scale to reflect the deviation of those values from the expectations of the model, and a question mark icon is added.

Our prototype is implemented in Java, using standard components of the Swing UI library. We use the Java API provided by the WEKA Data Mining Software [4] to implement the algorithms for statistical inference. In our current prototype, we use the standard WEKA implementations of the multilayer perceptron, C4.5 decision tree, support vector machine, and simple linear regression.

III. Experimental study

We evaluated our prototype using a variant of the "Champagne Prototyping" (CP) method [1], in which an advanced spreadsheet feature is demonstrated to an end-user, who is then interviewed to determine whether they have understood the conceptual basis of what they have seen without explicit explanation. In this study, we extended the method, asking the user to then carry out tasks of a similar nature by themselves, in order to see whether they are able to generalise their conceptual understanding and develop the competence to apply it without

Sarkar, A., Blackwell, A. F., Jamnik, M., & Spott, M. (2014). Teach and try: A simple interaction technique for exploratory data modelling by end users. In 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 53–56). IEEE. doi:10.1109/VLHCC.2014.6883022

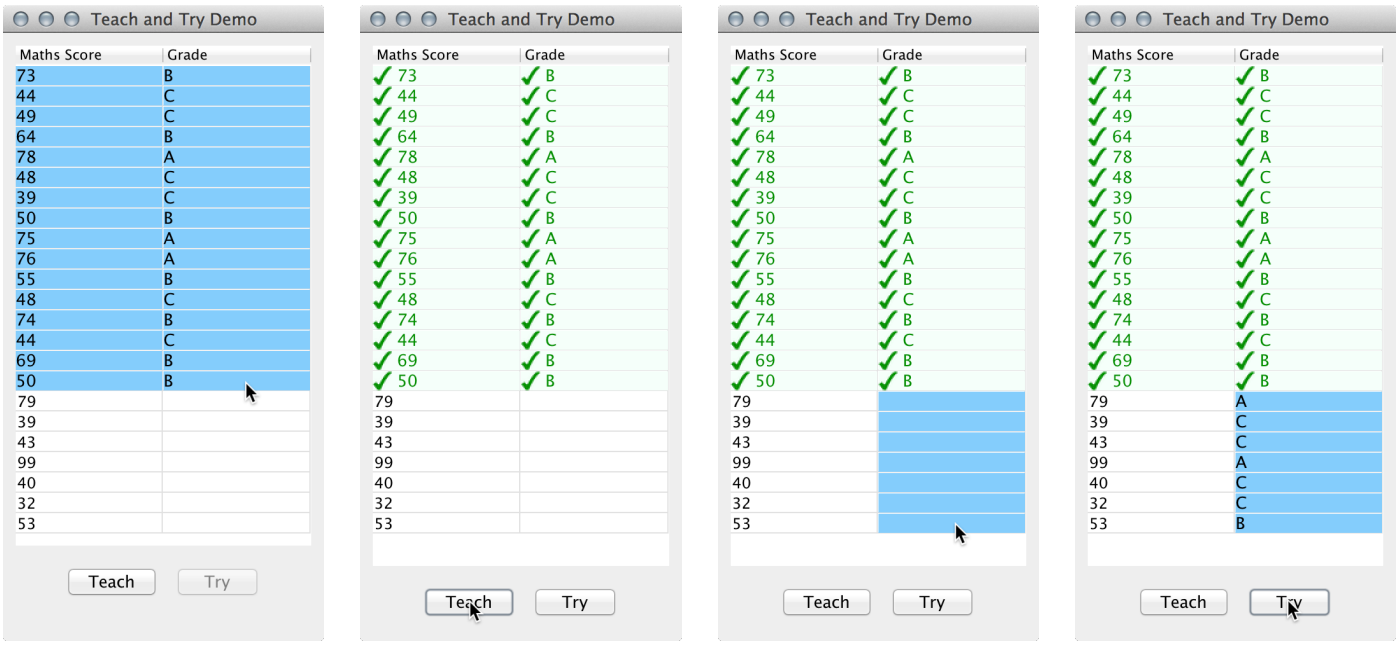

Fig. 1: The "filling" capability of the prototype, depicted as a sequence of actions from left to right. This resembles an actual scenario presented to the participants of our study, in which they were asked to imagine themselves as a maths teacher grading students based on their score on a recent exam. Each row is a student. The first column is their score, and the second column is their grade. A score of 75 or above gets an A, 50 to 74 gets B, and 49 or below gets C. Some of the Grade column is pre-populated, and the participants were asked to use the software to "quickly grade the remaining students". The model being implicitly built in these figures is a decision tree classifier.

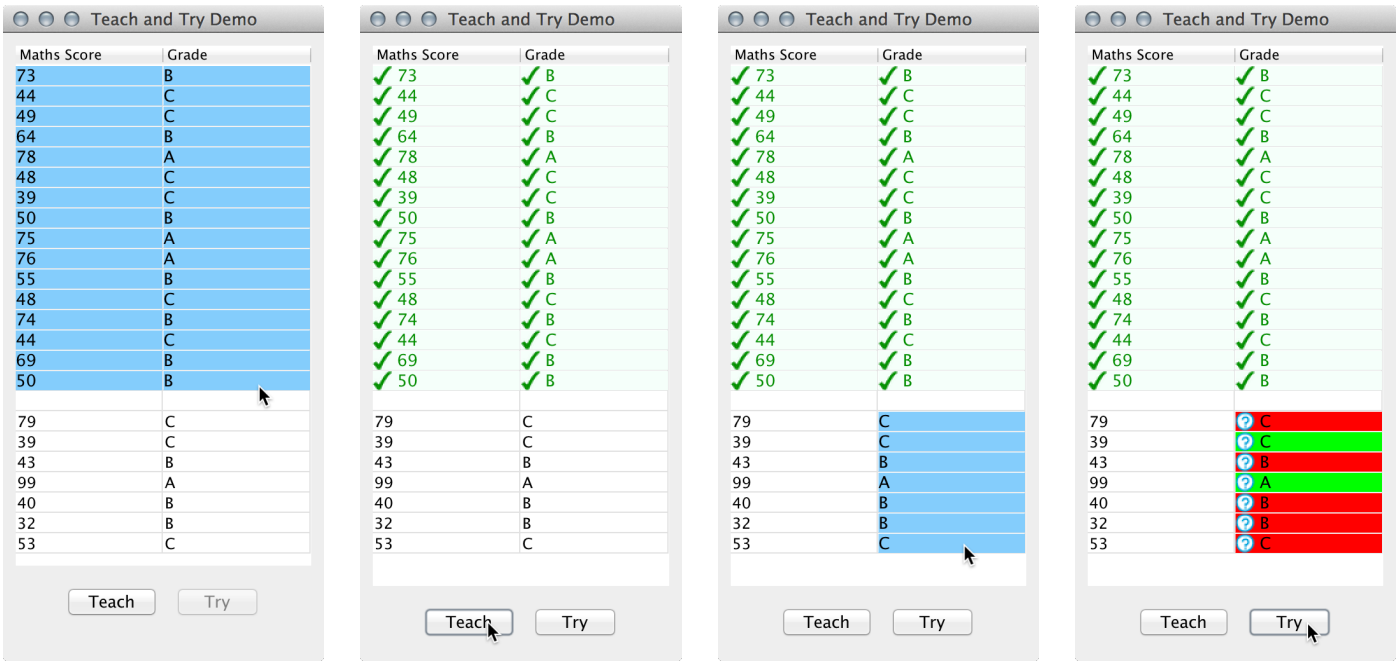

Fig. 2: The "evaluation" capability of the prototype, depicted as a sequence of actions from left to right. This resembles an actual scenario presented to the participants of our study. They were asked to imagine themselves as a maths teacher assessing the competency of a colleague, who had graded the latter section of the students. The first section of the Grade column is pre-populated with "correct" data as per the marking scheme from Fig. 1, and the participants were asked to use the software to "check whether the new teacher understands how to grade papers". Here, the machine learning task is that of classification, so there are only two colours assigned to the cells; green for "correct" and red for "incorrect". In other scenarios, where the target variable was numerical and continuous, the cells were coloured according their percentage error on a linear red-green scale.

further assistance. A further distinction between our study and CP is that in CP, "the core feature of interest is not actually executable by the computer or even by the researcher". In contrast, the feature of interest in our prototype is completely functional and is executed on the computer.

We recruited 21 participants, largely administrative staff, from the University of Cambridge Computer Laboratory and BT Research. Of these, 8 participants had some prior exposure to statistical concepts. The remaining 13 participants had no prior experience with either machine learning or statistical modelling, but had a basic familiarity with spreadsheet environments. We enforced this categorisation through a post-experiment interview about users' prior knowledge.

We created seven small tasks with corresponding datasets. Each task presented the user with a hypothetical scenario in which they used the software to perform a simple statistical procedure, without any explicit acknowledgement that they were doing so. Of the seven tasks, three were so-called "filling" tasks. These required the user to use existing data to fill in missing information. A further three were "evaluation" tasks, which required the user to use known high-quality data to assess or evaluate the quality or correctness of certain other data. A final task combined both filling and evaluation into one spreadsheet.

The experimenter demonstrated the use of the software for one of the evaluation and filling tasks each. The user was then asked to complete the remaining tasks, and was asked after each to explain what they had done. The order of the tasks was counterbalanced across participants. User responses were recorded, transcribed and analysed. For each user we recorded the time taken to complete the task, and observed unprompted references to statistical concepts.

IV. Results

A. Task durations

We did not observe a significant difference in task durations between the inexperienced and the experienced participants, suggesting that the software is equally usable by users with knowledge of machine learning or statistics as well as users without any prior exposure to such concepts.

We observed a significant effect of task order on task duration within task type; specifically, the second task of any type ("filling" or "evaluation") takes *less* time than the first task of the same type. The task durations were not normally distributed, so the Wilcoxon rank sum test is applied to yield a highly significant location shift ($p < 0.004$). The size of the effect is a median improvement of 2.9s from the first task to the second (the mean improvement is 33.6s, but as the data is highly non-normal this measure is biased by outliers). A plot of the task durations is shown in Figure 3. We interpret this gain in speed as evidence of the *learnability* of our interaction mechanism; the experience of interacting with the software leads the user to quickly acquire the competence to apply it in a new context.

B. Conveyance of statistical concepts

We now argue that the participants developed a nontrivial understanding of how the software works. Upon completion of the tasks, the participants were asked two questions in order to establish what kind of understanding of the system the user had acquired; specifically, these were (1) "How might the computer be doing this?", and (2) "Why might the computer make mistakes?".

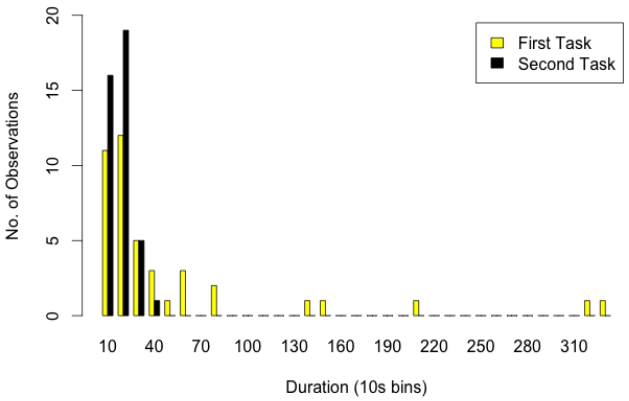

Fig. 3: Frequency distribution of task duration, broken down by task order. Observe the decreased spread in the second task.

Because of the potential bias from previous experience, we now deal exclusively with the utterances of the participants whom we consider to be *inexperienced* with respect to machine learning and statistical concepts. Note that we present the following observations not to make strong claims about the taxonomy of users' beliefs, but rather to demonstrate that users had gained a sufficient appreciation of the machine learning paradigm (that of building a model using trusted data and applying it for inference) to enable more confident use than if the software's behaviour seemed difficult to predict.

1) *Mathematical*: On 3 occasions, participants used mathematical terminology to describe their understanding of the software, guessing that it was "solving a system of linear equations", "finding some sort of correlation", or "plotting a graph".
2) *Technical / software constructs*: On 3 occasions participants explained the software's behaviour in terms of technology they were familiar with; in particular, these participants thought that the computer might be constructing complicated spreadsheet formulae, SQL queries or conditional formatting rules.
3) *Case-based reasoning*: On 3 occasions participants informally described the case-based reasoning, or nearest-neighbour prediction algorithms ("If a statement has been shown to be true in the past, then in the future, this statement must also be true" / "It checks to see if there is a precedent"). This suggests that nearest-neighbour matching or case-based reasoning may be among the most intuitive machine learning algorithms.
4) *Non-technical*: On 6 occasions, participants described the software's behaviour in abstract, non-technical terms, saying that it "spotted patterns", "makes different connections between the numbers", "deduces rules", "makes assumptions about how things should look and extrapolates", or "accumulates experience".

Furthermore, when asked why the computer might make mistakes, the participants provided multiple explanations which have familiar implications in statistics and machine learning, such as the following:

1) *Insufficient examples* (7 cases): "There's not enough information available [...] to exactly predict the pattern" / "What you taught it didn't cover it" / "The more data you have, the better will be the outcome"
2) *Noisy training data* (7 cases): "What you taught was wrong" / "It could be getting mixed messages from the data" / "There is a contradiction in the data"
3) *Insufficient discriminatory power* (between statistical classes) (4 cases): "Maybe the [data] overlap and they're quite ambiguous".
4) *Outliers* (3 cases): "It might be exceptional data"
5) *Incorrect model* (3 cases): "The method isn't yielding the correct answer"
6) *Insufficient dimensions* (3 cases): "There might be other factors besides those listed in the data that influence the outcome"

C. Usability issues

Some participants initially misunderstood how the selection bounds for the "Try" action were being interpreted. The most common error (3 cases) was to select *all* the cells in the target rows, because they had done exactly that for the "Teach" action. The next most common error (2 cases) was to select all cells in the target row *except* the cells in the target column, to instruct the computer to "Try to use *this* data...". This suggests that while the "Teach" selection is immediately intuitive, further work might be conducted in order to establish a more intuitive mode for the "Try" selection. One possible alternative is to invert the order in which the actions are taken, so that the "Teach" and "Try" buttons are pressed *before* making the selection. Here, the cursor would allow the user to "paint" regions of the spreadsheet as training or test data.

Additionally, some participants found the "Teach" and "Try" labels confusing; "Teach" is an action taken by the user, whereas "Try" is an action taken by the computer. This semantic irregularity caused some initial difficulty, which was overcome once the participant had completed the first task; however, future work might investigate alternative labels, such as "Train" and "Test", or "Learn" and "Apply".

V. Related work

The cell annotation aspect of our technique is related to WYSIWYT [7], in which users of a spreadsheet test it by marking "correct" values in individual cells, allowing the system to synthesise boundary conditions. However, WYSIWYT allows users without spreadsheet programming knowledge to debug their data, whereas our system allows users without statistical knowledge to build and apply statistical models.

The filling behaviour may appear similar to the string processing algorithm for spreadsheets due to Gulwani [3]; however, the capabilities of the underlying systems are different. While Gulwani's system synthesises a specific class of string manipulation functions, our interaction technique allows for the application of many kinds of statistical models and inference algorithms. For inexperienced users, an appropriate model can often be inferred from heuristic characterisation of the selected regions. For instance, if the "Try" region contains categorical data, a classifier is likely to be appropriate.

The Oracle Spreadsheet add-in for Predictive Analytics (SPA) [2] provides a spreadsheet-based interface for estimating the explanatory power of one variable with respect to another, and for performing SVM-based classification/regression. However, the output of the system operations is displayed in a separate spreadsheet to the original data, which reduces the directness of its manipulation. Furthermore, operations are triggered through a graphical wizard where the dependent variable and feature vector are manually specified, whereas we exploit the information present in the user's selection bounds to achieve the same effect. Finally, being targeted towards Business Intelligence (BI) professionals, it exposes statistical concepts that require domain knowledge to be interpreted, and thus it is unusable by those without such knowledge.

VI. Conclusion

Our goal was to make advanced analytical techniques available to end-users with no statistical training. We have described a simple spreadsheet-like interaction technique that can be used to train and apply powerful statistical models. We have presented results of an experiment demonstrating that the system can be understood and successfully applied by users having no professional training in statistics or computing, and that the experience of interacting with the system leads them to acquire some understanding of the concepts underlying exploratory data analysis methods.

Acknowledgments

Advait Sarkar is funded through an EPSRC Industrial CASE studentship sponsored by BT Research and Technology, and also by a Premium Studentship from the University of Cambridge Computer Laboratory. Many thanks to all the participants for their valuable time and effort.

References

[1] Alan F. Blackwell, Margaret M. Burnett, and Simon Peyton Jones. Champagne prototyping: A research technique for early evaluation of complex end-user programming systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, VLHCC '04, pages 47–54, Washington, DC, USA, 2004. IEEE Computer Society.

[2] Marcos M Campos, Peter J Stengard, and Boriana L Milenova. Data-centric automated data mining. In *Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on*, pages 8–15. IEEE Computer Society, 2005.

[3] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. In *ACM SIGPLAN Notices*, volume 46, pages 317–330. ACM, 2011.

[4] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[5] F. Pedregosa and et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[7] Gregg Rothermel, Lixin Li, Christopher DuPuis, and Margaret Burnett. What you see is what you test: A methodology for testing form-based visual programs. In *Proceedings of the 20th International Conference on Software Engineering*, ICSE '98, pages 198–207, Washington, DC, USA, 1998. IEEE Computer Society.