Interactive visual machine learning in spreadsheets

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

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

***Abstract*—BrainCel is an interactive visual system for performing general-purpose machine learning in spreadsheets, building on end-user programming and interactive machine learning. BrainCel features multiple coordinated views of the model being built, explaining its current confidence in predictions as well as its coverage of the input domain, thus helping the user to evolve the model and select training examples. Through a study investigating users' learning barriers while building models using BrainCel, we found that our approach successfully complements the Teach and Try system [1] to facilitate more complex modelling activities.**

I. Introduction

Our goal is to make general-purpose machine learning tools accessible to non-expert end-users [2]. Current solutions (e.g., WEKA [3] or scikit-learn [4]) are designed for professional statisticians or computer scientists, and are conceptually complex compared to end-user data manipulation tools, such as spreadsheets. A rapidly increasing range of applications exposes end-users to machine learning (e.g., email filtering and recommender systems), where statistical models are manipulated implicitly through examples, rather than programmed explicitly. These are easily adopted by non-experts, but facilitate modelling only within limited problem domains.

Our solution for general-purpose interactive machine learning builds upon the spreadsheet, a versatile data manipulation paradigm already familiar to end-users. We have previously demonstrated a simple interface that enables non-expert end-users to build sophisticated machine learning models in spreadsheets [1]. However, we did not address how the user might work with large, noisy datasets, where data must be carefully selected to properly model the domain, avoiding pitfalls such as overfitting. How would the user know what examples to select? Whether the model has acquired a good coverage of the domain? Whether some training data is noisy?

The user should be able to *critically evaluate* the quality, capabilities, and outputs of the model. We present "BrainCel," an interface designed to facilitate this. BrainCel enables the end-user to understand:

1) *How their actions modify the model*, through visualisations of the model's evolution.
2) *How to identify good training examples*, through a colour-based interface which "nudges" the user to attend to data where the model has low confidence.
3) *Why and how the model makes certain predictions*, through a network visualisation of the k-nearest neighbours algorithm; a simple, consistent way of displaying decisions in an arbitrarily high-dimensional space.

II. Interface and system architecture

BrainCel is a browser-based prototype. Data is loaded from comma-separated files on the local filesystem, and must conform to a well-defined relational schema. A standard spreadsheet view (Fig. 1(a)) displays the file. An overview of the spreadsheet (Fig. 1(b)) is presented to its left. Hovering on the overview previews the rows in the proximity of the hover location (Fig. 1(c)), enabling the user to "peek" at various parts of the spreadsheet without losing their current position in the main spreadsheet. Clicking on the overview scrolls the main spreadsheet to the click position. This creates an augmented overview+detail [5], that is, *overview+detail+peeking*.

A. Model training and application

We implemented the two-step interface for training and applying models first introduced in Teach and Try [1]. The user first selects rows of data which they believe to be correct. By pressing the "Learn" button, selected rows are added to the training set. To indicate that rows are in the training set, their row numbers are coloured blue. With training data added, the user can select empty cells and click the "guess" button to apply the model and predict values for those cells.

Predictions are made using the k-nearest neighbours (k-NN) algorithm. We find the k rows in the training set most similar (closest in Euclidean distance) to the row where an empty cell is to be filled. Columns are heuristically characterised as containing either categorical or continuous data, based on whether the data in the column can be parsed as numeric. If the cell is in a numerical column, the values of the neighbours in the same column are averaged to provide a guess. If the cell is in a categorical column, the majority vote (mode) is taken. Other models, including more sophisticated implementations of k-NN, have not been considered within our current scope. Feature weights are not currently adjustable, however, future work may introduce additional parameter controls, such as bar charts for feature weighting [6], varying k, and editing the distance metric, e.g., through Brown et al.'s method [7].

B. Expressing confidence through colour

For each row, the mean distance to its k neighbours is taken to be its confidence value [8]. A high mean distance (the row's nearest neighbours are relatively far from it) is interpreted as *low* confidence; conversely, a low mean distance is interpreted as *high* confidence. Rows are coloured on a red-green scale, where red indicates low confidence. Additionally, lightness is scaled so that a higher confidence is given a higher lightness; this de-emphasises the visual salience of green and makes it safe for red-green colour blindness. Through colour, the overview summarises confidence over the entire spreadsheet;

Sarkar, A., Jamnik, M., Blackwell, A. F., & Spott, M. (2015). Interactive visual machine learning in spreadsheets. In 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 159–163). IEEE.
DOI: 10.1109/VLHCC.2015.7357211

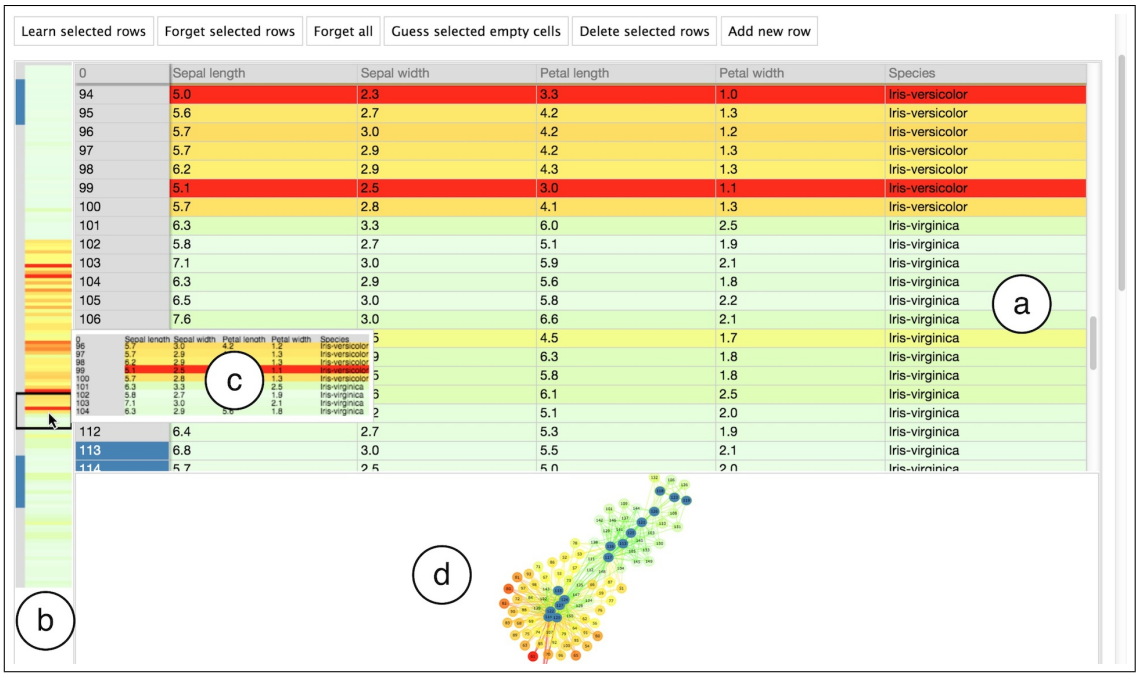

Fig. 1. Part of the BrainCel interface showing (a) the core spreadsheet, (b) the overview, (c) peeking at the spreadsheet contents, and (d) the explanatory network visualisation. Rows with row numbers coloured blue have been added by the user to the training set, and all rows have been coloured according to the model's confidence. Other parts of the interface, such as the distributions in Fig. 2 and the progress graphs in Fig. 4, are visible upon scrolling down.

green areas are well-modelled by the training data, whereas red areas are dissimilar to their nearest neighbours in the training set, marking them out either as being inadequately represented in the training set or as outliers.

C. Visually explaining the k-NN algorithm

A force-directed network visualisation of the model is displayed in the lower area of the interface (Fig. 1(d)). Each node represents a row, and has k edges going to its k nearest neighbours in the training set. Edge lengths are proportional to the distances to the k neighbours. We thus project the n-dimensional rows onto a 2D space. Since in order to explain the k-NN algorithm's behaviour it suffices to represent *proximity*, rather than variation along any particular dimension, we sacrifice concrete interpretations of the spatial axes in favour of expressing "nearness" and "farness."

This visualisation has several advantages. First, it facilitates *why* and *why not* explanations for any given prediction; the answer to "why was this cell value guessed?" is that the model drew upon the rows directly connected to it. Second, it provides a *how* explanation for k-NN; the general answer to "how is a value guessed?" is that all the rows are arranged by their similarity to each other and each guess draws upon the most relevant rows. Third, the emergent clusters visually reify the abstract model. For example in a 3-class classification problem, as in the Iris dataset [9], as rows are added to the training data, the network first branches from one cluster into two, and then converges at three (Fig. 3). Fourth, a colour scheme consistent with the spreadsheet explains how confidence is computed, since high-confidence rows (green nodes) occur close to rows in the training set (blue nodes).

The network visualisation fits the k-NN model well, but what about "explaining" other machine learning algorithms? In future work, one could try and explain other algorithms as though they were k-NN [10]. Another option is to use model-specific visualisations, such as Kulesza et al.'s bar charts for naïve Bayes [6]. The general-purpose, modular nature of BrainCel makes it a flexible testbed for comparing different visual "explanations" of the same model.

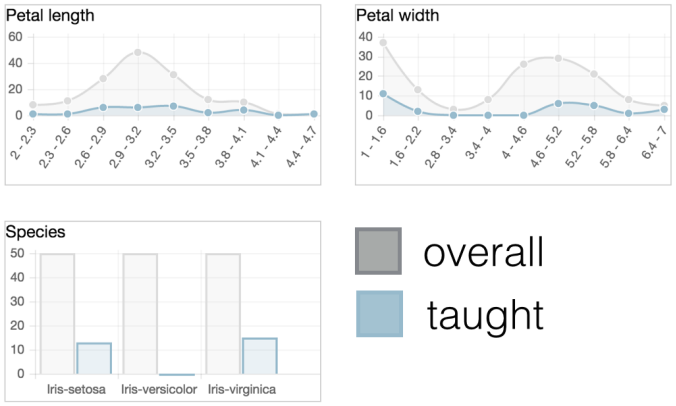

Fig. 2. Distributions of taught vs overall data. The "Species" graph shows that the class "Iris-versicolor" is underrepresented in the training data.

D. Expressing training set representativeness

BrainCel displays the value distributions within the columns in the overall dataset, compared to the value distributions in the training set (Fig. 2). These charts expose whether certain classes or types of data are under- or overrepresented, either in the underlying dataset, or in the training data. Thus, the user can assess whether the distribution of taught data is representative of the overall spreadsheet.

E. Visualisations of machine understanding progression

To help the user understand how the model's quality evolved with their actions, BrainCel displays two line graphs

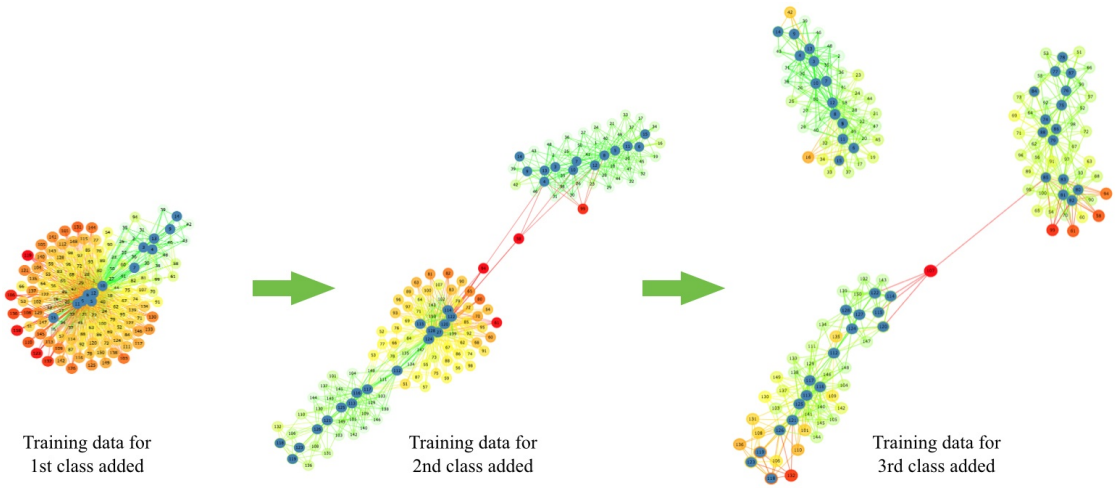

Fig. 3. Evolution of the model as shown by BrainCel's network visualisation when data from the three classes in the Iris dataset [9] is incrementally added. Rows in the training set are depicted in dark blue, and all other nodes are coloured according to their mean distance from their k nearest neighbours.

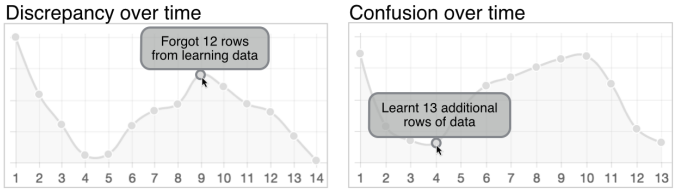

Fig. 4. Summary charts of training set representativeness and overall confidence. The x-axis shows interaction history, with data points being created on each edit. Tooltips show the action which created the point.

which update over time (Fig. 4). The first summarises how well the training data represents the overall dataset. New points are added whenever the training set is modified, computed by summing the Hellinger distances [11] between the distributions of attribute values in the training set and their corresponding distributions in the overall dataset. This quantity decreases as the distribution of training data becomes more similar to that of the overall dataset. This graph is presented as the "Discrepancy" between taught and overall data. The second shows the mean confidence over all rows as a function of interaction history: a new data point is added whenever the spreadsheet is modified. We vertically invert this graph to match the other graph, where lower values are more desirable, and present it as the machine's "confusion."

In both graphs, tooltips describe the action which led to a given data point being added. This provides a chronological account of how the model evolved in response to user actions. Since the vertical scale on these graphs is not directly related to the user's problem domain, the y-axis is unlabelled to encourage the user to think of the quantities as merely increasing or decreasing, rather than focussing on exact values.

III. Exploratory User Study

We conducted a user study modelled after Kulesza et al. [12], not as a summative evaluation, but rather to understand how our approach could support end-user machine learning. Thus, we observed (1) the learning barriers users encountered, and (2) which parts of the interface were most useful and why.

The study used a think-aloud protocol. Participants completed two equally-difficult randomised tasks, where they were given a spreadsheet (either the Iris or Zoo dataset [13]) containing empty cells, and were asked to fill the missing information using BrainCel. The first task was done with a spreadsheet-only version of BrainCel without visualisations (only item (a) in Fig. 1, without confidence-based colouring, i.e., the Teach & Try interface [1]), and the second with the complete BrainCel interface. As with the Whyline [14], comparing a full-featured version to a version with reduced functionality helped reveal how augmenting the standard spreadsheet interface affected users' exploration of their statistical models.

We recruited participants from humanities departments at Cambridge University. We screened out participants with prior exposure to statistics or machine learning, leaving 7 in the final analysis. All 7 successfully completed both tasks (populated the spreadsheets with correct values) in under 45 minutes.

We used Kulesza et al.'s coding scheme (a subset of Ko et al.'s learning barriers [15]). Briefly, the codes are as follows: *Design barrier*: the user's goals are unclear. *Selection barrier*: the goal is clear but the programming tools required to achieve this goal are unclear. *Use barrier*: the tools required are clear, but the user does not know how to use them properly. *Coordination barrier*: the tools required are clear, but the user cannot make them work together. *Understanding barrier*: the user thinks they know what to do, but their actions have surprising results. We applied the codes to sentences. Two researchers first independently coded random 5-minute transcript excerpts, iteratively refining coding rules until mean Jaccard index agreement reached 86% for a 5-minute transcript section, and 83% for a complete 40-minute transcript. The remaining transcripts were then coded.

A. Results: learning barriers

Fig. 5 shows the distributions of learning barriers encountered by our participants when using BrainCel with and without the additional visualisations. We confirm Kulesza et al.'s observations that Selection and Coordination barriers are the most common. Selection barriers revealed unclear aspects of the interface (e.g., P1: *Can I select these to learn?*, P3: *They've highlighted, so that must mean they're okay, or does it?*). Coordination barriers occurred when the model responded to changes in the training data in non-obvious ways (e.g., P3:

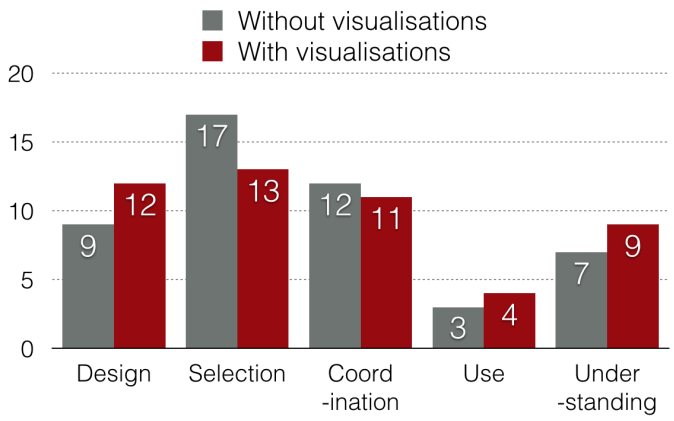

Fig. 5. Count of learning barriers encountered in all transcripts.

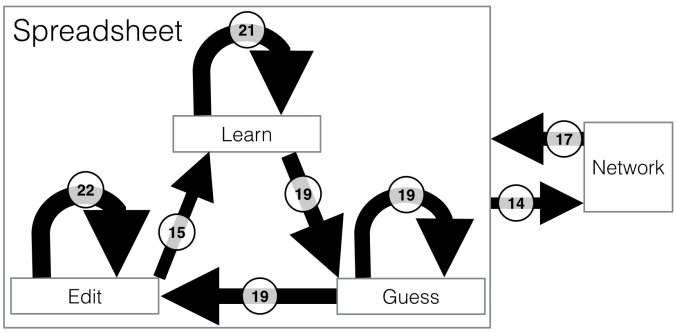

Fig. 6. Most common activity transitions. Numbers and arrow widths give the total observed count of transitions. "Learn" corresponds to adding training data, "Guess" to invoking the model, and "Edit" to editing values. "Spreadsheet" and "Network" refer to inspecting those areas respectively.

So let's delete the [classifications] it got wrong and try again... no, they're still wrong., P4: *So, what are [the model's] problem areas?*). Design barriers occurred when choosing a strategy for debugging incorrect predictions, e.g., whether one should add or remove training data, manually correct the prediction, or correct the prediction and then add it as training data.

Our small sample precludes statistical comparisons. However, it appears that Selection and Coordination barriers (found by Kulesza et al. to be the most prevalent) are both less frequent when visualisations are introduced. With visualisations, Design, Use and Understanding barriers were observed *more frequently*. This increase can be attributed to the fact that once users have their attention drawn to the structure of the model rather than surface features, the barriers they encounter become more interesting. For example, P5 expressed the following relatively mundane understanding barrier when working without the visualisations: *Why is that row [misclassified]?* However, with visualisations, P5 described more complex understanding barriers, e.g.: *It's gotten that correct, but why is it still not confident about it?* The visualisations helped end-users extend the zone of proximal development [16] past the simpler concepts of the spreadsheet training paradigm to more sophisticated conceptual issues regarding critical assessment of the model. Thus, an analysis of the difference in barrier distributions *with*- and *without*-visualisations is not sufficiently nuanced to show the way in which the visualisations helped. This suggests an additional dimension to the learning barriers, capturing a notion of "hardness" or "sophistication," acknowledging that some barriers are higher than others.

B. Results: user activity flows with visualisations

We recorded participants' transitions between interface activities in the *with*-visualisation tasks and present the most common transitions (each accounting for $>5\%$ of all transitions) in Fig. 6. The self-loops on learning, guessing and editing correspond to an incremental approach commonly adopted by participants where training data was added one row at a time, and cell values were guessed one at a time, to inspect the direct consequence of adding or removing training rows. Similarly, the pattern Edit→Learn→Guess also appeared frequently: manually correcting mispredictions, adding the corrected row to the training set, and seeing if other incorrect guesses were now correct. This suggests that manual effort may be saved if guessed cells were not static, but constantly recalculated like spreadsheet formulae. Participants frequently alternated between the spreadsheet and the network visualisation, using the network to diagnose mispredictions and study the model structure. Interestingly, participants overwhelmingly preferred to incrementally *add* to the training data, rather than remove rows, to improve the model.

IV. Related work

Amershi et al. [17], and Lim & Dey [18] identify what types of information intelligent applications should give to end-users, and Kulesza et al. [19] demonstrate that this information is critical for the formation of users' mental models. BrainCel incorporates several such information types; the network visualisation answers the *how* and *why* questions about predictions, and the distribution charts show what the system "knows."

BrainCel's summary line charts of model evolution build on Behrisch et al. [20], where a live visualisation shows how much of the data passes a confidence threshold, enabling users to assess convergence. Similarly, our use of confidence to highlight rows on which the user might wish to focus, is based on Groce et al.'s experimental evidence that model confidence is an effective way of selecting testing examples [21].

BrainCel emphasises how concepts evolve in the "mind" of the *computer* (consider Fig. 3). Conversely, Kulesza et al. present interfaces to help refine models in the mind of the *user*, who may not have well-defined mental concepts [22]. The intersection of our two approaches suggests *joint* concept evolution, visualising a shared man-machine understanding.

V. Conclusion

We have presented BrainCel, an interactive visual system for general-purpose machine learning in spreadsheets. BrainCel's multiple coordinated views of the model explain its current confidence, its coverage of the input domain, and provide *why* and *how* explanations for predictions, helping the user debug the model and select training examples. We reported an exploratory user study confirming that BrainCel successfully exhibits properties desirable in interactive machine learning systems, but within the general purpose spreadsheet paradigm previously proposed in the Teach and Try system.

Acknowledgments

The authors would like to thank the participants for their time. 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.

REFERENCES

[1] A. Sarkar, A. F. Blackwell, M. Jamnik, and M. Spott, "Teach and try: A simple interaction technique for exploratory data modelling by end users," in *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*. IEEE, Jul. 2014, pp. 53–56.

[2] A. Sarkar, M. Jamnik, A. F. Blackwell, and M. Spott, "Spreadsheet interfaces for usable machine learning," in *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 283–284.

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] A. Cockburn, A. Karlson, and B. B. Bederson, "A review of overview+ detail, zooming, and focus+ context interfaces," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 2, 2008.

[6] T. Kulesza, M. Burnett, W.-k. Wong, and S. Stumpf, "Principles of Explanatory Debugging to Personalize Interactive Machine Learning," in *Proceedings of the 20th International Conference on Intelligent User Interfaces - IUI '15*, 2015, pp. 126–137.

[7] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang, "Dis-function: Learning distance functions interactively," in *Visual Analytics Science and Technology (VAST), IEEE Conference on*. IEEE, 2012, pp. 83–92.

[8] S. J. Smith, M. O. Bourgoin, K. Sims, and H. L. Voorhees, "Handwritten character classification using nearest neighbor in large databases," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 9, pp. 915–919, 1994.

[9] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[10] A. Sarkar, "Confidence, command, complexity: metamodels for structured interaction with machine intelligence," in *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)*, Jul. 2015, pp. 23–36.

[11] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen." *Journal für die reine und angewandte Mathematik*, vol. 136, pp. 210–271, 1909.

[12] T. Kulesza, S. Stumpf, W.-K. Wong, M. M. Burnett, S. Perona, A. Ko, and I. Oberst, "Why-oriented end-user debugging of naive bayes text classification," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 1, no. 1, p. 2, 2011.

[13] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[14] A. J. Ko and B. A. Myers, "Designing the whyline: a debugging interface for asking questions about program behavior," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 151–158.

[15] A. J. Ko, B. A. Myers, and H. Aung, "Six Learning Barriers in End-User Programming Systems," in *2004 IEEE Symposium on Visual Languages - Human Centric Computing*. IEEE, 2004, pp. 199–206.

[16] L. Vygotsky, "Zone of proximal development," *Mind in society: The development of higher psychological processes*, vol. 5291, 1987.

[17] S. Amershi, J. Fogarty, A. Kapoor, and D. S. Tan, "Effective end-user interaction with machine learning." in *AAAI*, 2011.

[18] B. Lim and A. Dey, "Assessing demand for intelligibility in context-aware applications," *Proceedings of the 11th international conference on Ubiquitous computing*, p. 195, 2009.

[19] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong, "Too much, too little, or just right? Ways explanations impact end users' mental models," in *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, 2013, pp. 3–10.

[20] M. Behrisch, F. Korkmaz, L. Shao, and T. Schreck, "Feedback-driven interactive exploration of large multidimensional data supported by visual classifier," in *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*. IEEE, 2014, pp. 43–52.

[21] A. Groce, T. Kulesza, C. Zhang, S. Shamasunder, M. Burnett, W.-K. Wong, S. Stumpf, S. Das, A. Shinsel, F. Bice, and K. McIntosh, "You Are the Only Possible Oracle: Effective Test Selection for End Users of Interactive Machine Learning Systems," *IEEE Transactions on Software Engineering*, vol. 40, no. 3, pp. 307–323, 2014.

[22] T. Kulesza, S. Amershi, R. Caruana, D. Fisher, and D. Charles, "Structured labeling for facilitating concept evolution in machine learning," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 2014, pp. 3075–3084.