Using Multiple Models to Understand Data

Kayur Patel[‡], Steven M. Drucker[†], James Fogarty[‡], Ashish Kapoor[†], Desney S. Tan[†]

[‡]Computer Science and Engineering
DUB Group, University of Washington
Seattle, WA 98115
{kayur, jfogarty}@cs.uw.edu

[†]Microsoft Research
One Microsoft Way
Redmond, WA 98052
{sdrucker, akapoor, desney}@microsoft.com

Abstract

A human's ability to diagnose errors, gather data, and generate features in order to build better models is largely untapped. We hypothesize that analyzing results from multiple models can help people diagnose errors by understanding relationships among data, features, and algorithms. These relationships might otherwise be masked by the bias inherent to any individual model. We demonstrate this approach in our Prospect system, show how multiple models can be used to detect label noise and aid in generating new features, and validate our methods in a pair of experiments.

1 Introduction

Adoption of machine learning is hampered by the difficulties practitioners encounter in debugging their models [Patel *et al.* 2008]. Poor model performance can stem from a variety of underlying reasons: labels may be noisy, features may not be sufficiently descriptive, or a practitioner may have chosen an inappropriate algorithm. When a problem lies in data (e.g., label noise, insufficiently descriptive features), diagnosing and fixing poor performance often requires human input and domain expertise. For example, feature creation often involves inspecting data and applying domain knowledge to write discriminative features.

We propose a new method for debugging machine learning systems by aggregating results from a *collection of models*. We hypothesize that multiple models can effectively marginalize the bias of individual models, providing practitioners with information that is usually masked when inspecting any single model. We address the problem of choosing the correct algorithm by trying many configurations (e.g., feature sets, learner algorithms, learner parameters, cross-validation folds). Each configuration provides a predicted label for each example, and we analyze the distribution of labels for a dataset.

We test our hypothesis in a system called *Prospect*. Prospect automatically trains a collection of models, aggregates results from those models, and provides interactive visualizations to help practitioners understand and debug data. We illustrate and evaluate these methods in two problems: detecting label noise and generating new features. In our first experiment, we show that using multiple models to identify potential label noise can provide a threefold reduction in the number of spurious examples a practitioner examines. In our second experiment, we show that analyses of multiple models can identify examples that are significantly more likely to respond to additional informative features, thus helping a practitioner focus their attention the most relevant areas of a problem. Our user evaluation shows that practioners can effectively use Prospect to understand their data.

2 Related Work

A recent study examined difficulties faced by machine learning practitioners [Patel *et al.* 2008]. Key findings highlight that practitioners have trouble understanding relationships between data, features, and models. As a result, they often have little guidance on how to improve predictive performance. When models do not work, practitioners often spend an inordinate amount of time optimizing their classification algorithm without checking the quality of their data and features. Prospect addresses these difficulties in two ways. First, Prospect automates rote experimentation by trying multiple configurations of different algorithms. Second, Prospect enables new analyses that help practitioners link classification results back to noteworthy examples. Practitioners can inspect these examples and apply their expertise to determine how to best improve performance.

Most development support for machine learning takes the form of an API. Some machine learning APIs are bundled with GUIs that allow practitioners to load featurized data, select an algorithm, and gather results. Weka is a well-known, widely-used example of this type of tool [Witten *et al.* 2005]. Although Weka and some of its extensions allow practitioners to train many different models on one dataset [Mierswa *et al.* 2006; Berthold *et al.* 2008], comparisons between trained models are limited to comparing model accuracy. Prospect provides deeper analytic support through functionality for generating multiple models, gathering results, and inspecting data. Prospect also allows practitioners to compare and interpret results from multiple models through interactive visualizations.

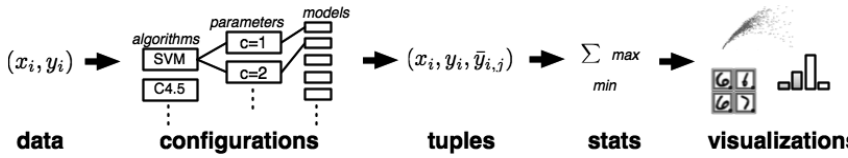

Figure 1: Given data, Prospect generates a set of models which output tuples containing predicted labels for each example in a dataset. Prospect computes statistics over the set of tuples and provides visualizations that help practitioners understand data and diagnose errors.

Other interactive machine learning tools have explored leveraging the complimentary strengths of humans and computers. By creating a synergistic relationship, interactive machine learning systems can be used to train better models [Raghavan *et al.* 2005; Kapoor *et al.* 2010; Talbot *et al.* 2009]. Several systems have explored this but their primary focus is direct interaction with the machine learning algorithm to improve its performance. Our ultimate goal is to train better models as well, but we believe tools must support the complex relationships between data, features, and algorithms. We are therefore deeply interested in the challenge of general methods to help practitioners to understand these relationships.

Recent work on the Gestalt system has also focused on aiding machine learning practitioners [Patel *et al.* 2010]. While Prospect and Gestalt share a common goal, they differ in approach. Gestalt supports the process of writing code to train a model, whereas Prospect allows practitioners to compare multiple models to understand label noise and generate new descriptive features. We feel that Prospect provides a complementary approach to Gestalt, both can be used in conjunction to help practitioners train better models.

There has been prior work on building visualizations that help practitioners interpret the behavior of machine learning algorithms [Caragea *et al.* 2001; Dai *et al.* 2008]. While successful, much of this work focuses on visualizing the innards of a single algorithm or a single type of data. This is limiting because some existing algorithms are intrinsically hard to interpret and because the space of datasets and algorithms is always increasing. Our methods abstract the algorithms and data to focus on the general problem of supervised learning (i.e., discrete labeled examples and models that provide predicted labels). As such, our technique scales to new models and diverse datasets and can work with models that are difficult to directly interpret.

The bias plus variance decomposition is a useful framework for evaluating supervised learning algorithms [Kohavi *et al.* 1996]. It defines three values. *Target noise* is inherent to data (i.e., it will exist even in perfect conditions). *Bias* is the difference between the expected value and optimal (i.e., structural error of the model). *Variance* is how much an algorithm's predictions vary based on different training data. All models have some inherent bias and variance, but these differ between algorithms.

Ensemble methods exploit intuitions about bias and variance to achieve better performance by combining results from multiple models. Combinations of classifier outputs based on simple rules (e.g., majority vote, sum, etc.) often produce results better than a single model [Kittler *et al.* 1998]. Other ensemble techniques, such as Boosting [Schapire 2002] and Bagging [Breiman 1996], automatically generate simple models and combine them to build more accurate ensemble models. However, this work is generally aimed at learning ensembles to directly improve accuracy. Our approach differs in that we leverage differences in biases to help practitioners better understand data independent of individual algorithms.

3 The Prospect System

In order to make correct choices about what algorithm to use, a practitioner must understand key properties inherent to the data. These properties are independent of any particular classification algorithm. Prospect lets practitioners better understand key properties of their data by first applying a collection of models and then analyzing the behavior and output of those models. The collection of machine learning models acts as a lens for scrutinizeing some of the key properties of data.

3.1 System Description

Figure 1 presents an overview of Prospect. Starting from data, Prospect first generates a set of *configurations*. Each configuration defines feature selection procedures, a learning algorithm, its parameters, and other specifications needed to completely determine a model creation process. Configurations are generated by systematically varying algorithms and parameters throughout the process. The goal is to create configurations that can be used to generate a collection of models that provide an *unbiased* perspective on the data. Our intuition is that by training multiple models using a diverse set of configurations, Prospect can marginalize the individual bias of any particular model.

Prospect considers each available configuration and uses k-fold cross-validation to generate classification results for the entire dataset. Formally, data consists of a set of examples $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, with labels $\boldsymbol{y} = \{y_1, \ldots, y_N\}$, and C, a set of different configurations. We use $\bar{y}_{ij}$ to denote the predicted label of $\boldsymbol{x}_i$ resulting from the j^{th} configuration. Thus cross validation for each configuration creates tuples of the form $(\boldsymbol{x}_i, y_i, \bar{y}_{ij})$. Interaction with Prospect is based on summarizing these tuples and allowing practitioners to visualize and analyze them.

One way Prospect summarizes these tuples is with descriptive statistics about examples. In the case of systems based on a single model, the predicted label is the only new information about an example. In contrast, Prospect computes a distribution of predicted labels for each example. This distribution allows Prospect to compute example-centric statistics like agreement (the percentage of configurations that agree on a label for an example) and max label (the label picked by majority of configurations).

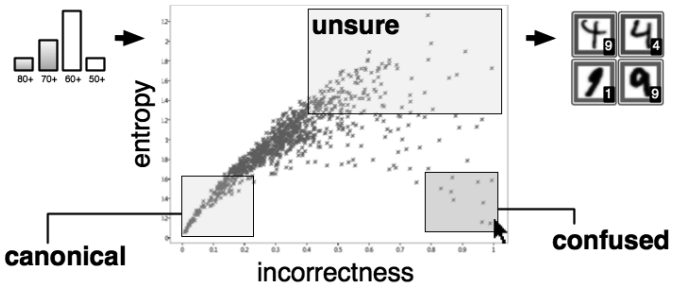

Figure 2: The middle image shows the incorrectness vs. entropy plot. To find label noise remove low accuracy configurations and inspect examples in the confused region.

These statistics can then be used to filter or visualize data. Prospect provides a library of common visualizations, such as histograms, scatter plots, and confusion matrices. Practitioners can interact with these multiple visualizations to filter their data. For example, they can select and filter examples on a scatter plot. Instead of describing the full set of visualizations and interactions enabled by Prospect, the following sections focus on how the summary statistics and interactive visualizations can be used to address two problems: helping detecting label noise and helping provide insight for generating new features.

3.2 Experimental Setup

Both experimental studies were conducted on three multi-class datasets taken from different domains: the 20 *newsgroups* dataset [Lang 1995], the MNIST *digit* recognition dataset [Lecun *et al.* 1998], and the *flowers* image recognition dataset [Nilsback *et al.* 2006]. We use the same scheme for generating configurations in both evaluations. This section describes these common datasets and configurations.

Datasets: The newsgroups dataset consists of posts collected from 20 different newsgroups. We select 1000 articles balanced by class and use the Weka API [Witten *et al.* 2005] to tokenize documents, stem words, and compute word count features (2057 features in total).

The digits dataset consists of 28x28 images of handwritten numerical digits. We select 1000 images balanced by class (100 for each digit) and use pixel values for features (784 features in total).

The flowers dataset consists of images of 17 types of flowers. Images vary in scale, pose, and lighting with large within-class variations. We select 680 examples balanced by class (40 per class). We use precomputed shape, color and texture features provided by Nilsback *et al.* [2006].

Configurations: To create different configurations, we systematically vary our feature space, classification algorithm, training and testing split, and other parameters. Specifically, we create multiple feature spaces using feature selection based on information gain. We apply three different classification algorithms (support vector machines, decision trees, and Naïve Bayes) and vary parameters for each algorithm. We also randomly vary the composition of cross-validation folds. Our implementation uses feature selection and classification algorithms provided by Weka.

We prune our set of configurations using two heuristics. First, in the interest of speed, we launch each configuraiton in a thread that terminates after one minute. If the executaiton does not terminate and produce a model we remove it from our list of configurations. Second, because we want a varied distribution of predicted labels, we remove configurations that apply the same classification algorithm and yield the same distribution of predicted labels.

Our current process for automatically generating configurations is limited, as there is no guarantee our set of configurations adequately samples the space of possible configurations. Optimal sampling remains future work. Even with this current limitation, our results are promising.

4 Detecting Label Noise

Ground truth labels are often imperfect for a variety of reasons. For instance, humans are prone to make errors when fatigued, when distracted, or when presented with inherently ambiguous data. Noisy labels can adversely impact results, and they are often expensive to find because detecting them often requires human inspection. Prospect can use multiple models to *reduce the number of examples* a practitioner inspects in debugging label noise.

Figure 2 (middle) shows a scatter plot visualization of two summary statistics for examples: *incorrectness* and *label entropy*. Incorrectness (x-axis) is the percentage of configurations that misclassify an example. Label entropy (y-axis) is the entropy of the distribution of labels predicted by each configuration for an example. Given the indicator function *1{}*, the set of configurations *C,* and the set of labels *L;* these values are computed as follows:

$$inc(x_i) = \sum_{j}^{|C|} \frac{1\{y_i \neq \bar{y}_{i,j}\}}{|C|}$$

$$ent(x_i) = -\sum_{l \in L} \frac{\sum_{j}^{|C|}\{\bar{y}_{i,j} = l\}}{|C|} \times \log \frac{\sum_{j}^{|C|}\{\bar{y}_{i,j} = l\}}{|C|}$$

Figure 2 highlights three regions in the scatter plot. The *canonical* region contains examples that most configurations classify correctly (i.e., low-incorrectness, low-entropy). The *unsure* region contains examples for which different configurations generate widely varying predicted labels (i.e., high entropy). The *confused* region contains examples with high incorrectness and low entropy. These are the examples for which most configurations agree on a predicted label, but the predicted label is not the same as the ground truth label. These *confused* examples are of the most interest for detecting label noise, as the consistent misclassification by many different models suggests the ground truth label may be incorrect.

Illustration of the Procedure: Figure 2 shows the exact steps a practitioner would take within Prospect to find potential label noise. First, a practitioner uses a configuration accuracy histogram to filter configurations with poor performance; this removes noisy models and reduces variance. Second, they select a set of relevant examples from the confused region of the scatter plot.

	inc+ent	inc	single classifer
news	0.9978	0.9963	0.9929
digits	0.9952	0.9935	0.9878
flowers	0.9910	0.9804	0.9837

Table 1: Area under the ROC curve.

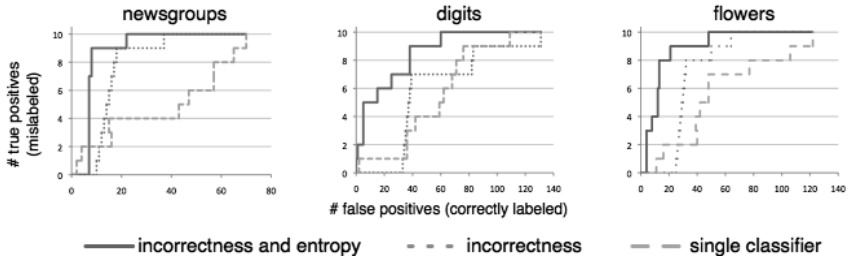

Figure 3: True positive vs. false positive for the three datasets.

Finally, they create a grid view to inspect those examples together with their ground truth labels. In this case, we can see that an example labeled '9' is clearly a '4' and that an example labeled '1' may be a '9'.

4.1 Experimental Evaluation

The *confused* region can be seen as a classifier that attempts to determine whether examples are *incorrectly labeled*. Because we want to minimize the examples a human inspects when identifying label noise, we want to maximize *incorrectly* labeled examples (true positives) while minimizing *correctly* labeled examples (false positives) in the confused region. We examine this tradeoff in comparison to two experimental baseline thresholds.

We first compare to sampling according to the posterior probability of the highest-accuracy model in the collection (sampling in order of increased confidence). This finds examples about which the single best model has the least confidence. We choose this for two reasons. First, practitioners often debug using their highest-accuracy model. Second, good training performance is often indicative of low bias, and a model with low bias is likely to have a good true positive / false positive tradeoff. This comparison thus shows the value of explicitly considering multiple models.

Our second comparison is to a threshold considering only our incorrectness statistic. Our *confused* region is differentiated from the *unsure* region according to the entropy of labels predicted by different configurations. This comparison confirms that both statistics provide distinct information to help in identifying noisy labels.

Procedure Details: To simulate label noise, we randomly selected 10 examples and randomly changed their ground truth label. Consistent with the process illustrated in Figure 2, we removed poorly performing models by using only the top quartile of configurations (ordered by model accuracy).

Our baseline conditions classify examples based threshold cutoffs. Examples below the threshold are correclty labeled, and examples above are incorrectly labeled. To compute ROC curves for the baselines, we set a high threshold value such that all examples are correctly labeled and then lower the threshold to compute the true positive and false positive rates.

We compare the baseline conditions to the confused region classifier. The confused region is a rectangle defined by two points. The bottom-right point is anchored, and the top-left point is variable. Examples within the rectangle are classified as incorrectly labeled. To compute the ROC curve for the confused region, we generate a set of possible confused regions by sampling different values for the top-left point and compute true-positive and false-positive rates for each possible confused region.

Results: Table 1 shows the area under the ROC curve for our three datasets. The area is high for all three conditions, but that is because ROC looks at the false positive and true positive *rates*. We have an unbalanced set of positive and negative examples, so the rates will not correspond to the actual number of false positives. Because we are trying to reduce the human cost of finding label noise, measured in the number of examples a human must verify, the *number* of false positives is important. Each new false positive is another example that must be manually verified.

To show the number of examples a human must inspect, we plotted the tradeoff between number of true positives (incorrectly labeled examples detected) and false positives (correctly labeled examples classifed as mislabeled examples). Figure 3 shows this tradeoff for our three datasets. The plots show that looking at the confused region (the solid red incorrectness and entropy line) results in fewer false positives for each true positive detected than the baseline conditions. Focusing on the confused region can reduce the number of examples a person must inspect by up to a factor of 3.

5 Generating New Features

Feature discovery is a complex process, with practitioners often needing to acquire specialized domain knowledge before they can discover discriminative features. We emphasize that the key to discovering new discriminative features lies in understanding properties of the data that distinguish one class from another. Such understanding can be developed through deep analysis of the features or through analysis of how different examples are classified. Automated feature selection methods generally focus on analysis of the feature set, but this can be non-trivial for humans (especially in a high-dimensional feature space). On the other hand, looking at how different examples are classified is generally easier to comprehend and can provide insight into deficiencies of a feature set. Prospect provides visualizations to examine aggregate statistics regarding how different examples are classified and misclassified. This can in turn be used to identify situations where the current feature set is not sufficiently descriptive.

Figure 4 illustrates how Prospect supports the interactive process of finding examples to help build intuition for new

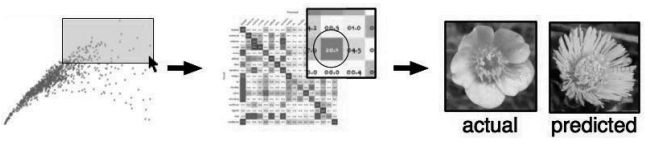

Figure 4: Looking at the confused examples in the unsure region can help practitioners understand their data and create features.

features. The key idea is to enable practitioners to observe a summarization of how different examples are classified, which should then help them develop insights about the differentiable characteristics of data. Figure 2's incorrectness vs. entropy plot again plays an important role in this process. To help develop insight into new potential features, we focus on the *unsure* region of the plot (examples with high entropy). These are misclassified by many configurations, but are not predominately mistaken for any particular class. These seem crucial to the process of feature discovery because their high entropy suggests that the available features cannot support reliable differentiation between classes. Focusing on the development of new features that are relevant to these examples should therefore provide new discriminative power to models.

Illustration of the Procedure: Figure 4 shows the exact steps a practitioner would take within Prospect to examine unsure examples to help develop insight for new features. A practitioner first uses a rubberband tool to focus on a set of examples with high entropy and high incorrectness. After filtering to focus only upon these examples, they create an aggregate confusion matrix. This presents the normalized sum of confusion matrices for each configuration, highlighting which classes are commonly confused across all models. The practitioner can then click into a cell in this aggregate confusion matrix to directly inspect examples that might benefit from new features. Prospect presents a representative example of a class together with each confused example. Here we can see that two examples have similar shape and color, but could likely be better differentiated by a new feature capturing image texture.

5.1 Experimental Evaluation

Human intuition is difficult to measure, and developing effective intuition about a feature often requires in-depth understanding of the problem. Instead of attempting to directly measure intuition, we assess how *relevant* examples are to the creation of new features. Relevant examples are those with the most capacity for improvement, those that will be "most helped" by the addition of new features. These should be most responsive to new features, exhibiting a larger change in correctness than other examples.

We claim that the *unsure* region contains relevant examples. To evaluate this claim, we conduct an experiment adding new descriptive features to a dataset and measuring the change in correctness for examples from the unsure region relative to the remainder of examples. We establish that examples from the unsure region are significantly more responsive to new descriptive features. This validates that practitioner focus on the unsure region can be an effective strategy for developing insight to generate new features.

Procedure Details: Our experiment requires two versions of each dataset: a *before* dataset and an *after* dataset created by the addition of new features. We define our base datasets (Section 3.2) to be the *after* datasets, then create the *before* datasets by removing features. For the flowers dataset, we remove the shape feature. The newsgroup features are not as clearly divisible, so we remove a random 75% of the features. Random removal is ineffective in the digits data because adjacent pixel values provide redundant information. We therefore define the before features to be only the top-left region of each digit.

We define the unsure region in terms of the before dataset, thresholding at the top quartile of examples (according to the entropy of distributions from the before configurations). This splits our set of examples $\mathbf{X}$ into the unsure region $\mathbf{U}$ and the not unsure region $\mathbf{X} - \mathbf{U}$. We then compute δ_f, the change in correctness for each example resulting from adding additional features to create the after dataset. To test that $\mathbf{U}$ contains more relevant examples, we use a t-test to determine whether δ_f is significantly greater in $\mathbf{U}$ than in $\mathbf{X} - \mathbf{U}$.

It is possible greater δ_f in $\mathbf{U}$ is due to extreme entropy of the examples rather than relevance to a new feature (i.e., an effect similar to regression to the mean). To test this, we generate a *noise* dataset by changing the random seed used to create cross-validation folds in the *before* dataset.

Changing the seed produces different training and validation sets, and different training sets will lead to changes in models, predicted labels, and entropy values. However, the underlying features are the same, it is only configuration of examples in the training and verification sets that differs. Since the features are the same, we refer to this difference as noise, and we use the noise value to determine if changes in entropy caused by the introduction of new features are significantly different than changes in entropy cause by the introduction of noise. We define δ_n as the difference due to noise, and use a t-test to determine whether δ_n is significantly greater in $\mathbf{U}$ than in $\mathbf{X} - \mathbf{U}$.

Because we find that examples in $\mathbf{U}$ are more responsive to noise, it is important to establish that the improvement to correctness that we see when adding relevant features δ_f is not explained by the noise δ_n. We examine this by testing that $\delta_f - \delta_n$ is significantly greater in $\mathbf{U}$ than in $\mathbf{X} - \mathbf{U}$**.** This validates the *relevance* of examples in the unsure region.

Results: Table 2 shows a table of the improvement in correctness resulting from the addition of new features. The correctness of unsure examples increases significantly more than other examples (newsgroups: $p<0.01$, digits: $p<0.01$, and flowers: $p<0.04$). After accounting for variation due to noise, we observe that this effect is significant for the newsgroups ($p<0.01$) and digits ($p<0.01$) datasets. Although the improvement in correctness is not significant for the flowers dataset ($p\approx0.27$), it is also not any worse.

	rest	unsure
news	-0.008	0.507
digits	0.138	0.247
flowers	0.057	0.090

Table 2: Improvement in correctness by adding new features

Consequently, we can conclude that focusing on the unsure examples can indeed provide practitioners with insights critical to the process of interactive feature discovery.

In summary, our experiments indicate that focusing on examples in the unsure region can only help. Consequently, the unsure region is a good place to start when trying to generate new features. While automatic discovery of new discriminative features is still a hard problem, Prospect can help a human practitioner to explore the properties of the dataset and reason about such exogenous variables.

6 User study

We conducted a user study where participants provided us data they had collected and were trying to model. After processing their data with Propsect and creating raw data visualizations, we brought participants back to the lab and asked questions about examples in the unsure and confused region. We had three participants (P1-P3). To maintain anonymity, we refrain from describing their datasets in detail. P2 and P3 were working on multi-class problems, while P1 was working on a binary classification problem.

Automation Saves Time and Effort: P1 had just started analyzing and modeling, and Prospect allowed them to concentrate on features and data without worrying about tweaking parameters. P2 was in the middle of analyzing and refining their data using a single algorithm. Prospect helped them explore the space of configurations to find one that provided a significant accuracy boost (around 30%). P3 was near the end of their project and had painstakingly performed a manual exploration of configurations. They commented that Prospect would have reduced the effort of manual exploration. In all three cases Prospect either saved or would have saved participants time and effort.

Redesigning Data Collection: P1 found that their data was not rich enough to model their classes. Prospect helped them conclude that a redesign of their data collection mechanism would help create better models.

Finding Label Noise: P1 took steps to gather and verify clean data. P3 provided us an early version of their data with label noise, and the noise was easy to spot in the confused region (Figure 2). P2 had yet to perform a through data cleaning, and after reflection, they identified confused examples as label noise. Data cleaning can be problematic, so Prospect's ability to find label noise was useful.

Reinforcing Feature Ideas: P3 noted that the unsure region contained examples that were particularly hard to classify, and that they were modifying their approach to be robust to these types of errors. Reflecting on their data, we were able to co-design a potential new feature. In fact, all three of our participants either came up with ideas for new features or reinforced ideas they already had. In this way, Prospect was able to guide the feature creation process.

7 Conclusion and Future Work

Input from multiple models can provide new and unique insight that can help developers understand various properties of the data. Prospect provides one instance of a tool that can leverage these insights to build better models. While we have shown that Prospect works well for two common debugging tasks (detecting label noise and generating new features), we believe there is potential to leverage the same intuitions to develop new techniques, tools, and visualizations that help other hard data understanding tasks (e.g., determining whether to collect more data). We are focusing on many additional interactive machine learning opportunities enabled by the ideas in this work, including new methods to better sample configurations, interactively generating task-specific collections of classifiers, and richer interaction scenarios.

Acknowledgments

We would like to thank Oren Etzioni and Dan Weld for their feedback on early versions of this work. This work was supported in part by the National Science Foundation under grant IIS-0812590 and by Kayur Patel's Microsoft Research Fellowship.

References

[Berthold *et al.* 2008] Michael R. Berthold *et al.* "KNIME: The Konstanz Information Miner." *Data Analysis Machine Learning and Applications*, vol. 11.

[Breiman 1996] Leo Breiman. "Bagging Predictors." *Machine Learning*, vol. 24.

[Caragea *et al.* 2001] Doina Caragea, Dianne Cook, and Vasant G. Honavar. "Gaining insights into support vector machine pattern classifiers using projection-based tour methods." *KDD 2001*.

[Dai and Cheng 2008] Jianyong Dai, and Jianlin Cheng. "HMMEditor: a visual editing tool for profile hidden Markov model." *BMC Genomics,* vol. 9.

[Kapoor *et al.* 2010] Ashish Kapoor, Bongshin Lee, Deseny Tan, and Eric Horvitz. "Interactive optimization for steering machine classification." *CHI 2010*.

[Kittler *et al.* 1998] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. "On combining classifiers." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20.

[Kohavi and Wolpert 1996] Ron Kohavi and David H. Wolpert. "Bias Plus Variance Decomposition for Zero-One Loss Functions." *ICML 1996*.

[Lang 1995] Ken Lang "NewsWeeder: Learning to Filter Netnews." *ICML 1995*.

[Lecun *et al.* 1998] Yann Lecun, L'eon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-Based Learning Applied to Document Recognition." *Proceedings of the IEEE*, vol. 86.

[Mierswa *et al.* 2006] Ingo Mierswa *et al.* "YALE: Rapid Prototyping for Complex Data Mining Tasks." *KDD 2006*.

[Nilsback and Zisserman 2006] Maria-Elena Nilsback and Andrew Zisserman. "A Visual Vocabulary for Flower Classification." *CVPR 2006*.

[Patel *et al.* 2010] Kayur Patel *et al.* "Gestalt: Integrated Support for Implementation and Analysis in Machine Learning."*Proceedings of the Symposium on User Interface Software and Technology*, *2010*.

[Patel *et al.* 2008] Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. "Investigating Statistical Machine Learning as a Tool for Software Development." *CHI 2008*.

[Raghavan *et al.* 2005] Hema Raghavan, Omid Madani, and Rosie Jones. "InterActive feature selection." *IJCAI 2005*.

[Schapire 2002] Robert E. Schapire. "The boosting approach to machine learning: an overview." *MSRI Workshop on Nonlinear Estimation and Classification*, *2002*.

[Talbot *et al.* Tan 2009] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S Tan. "EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers." *CHI 2009*.

[Witten and Frank 2005] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd ed.