The effect of pipeline-collection-diversity on performance

Sean Carver @ Data Machines Corporation

August 26, 2020

Abstract

Do performers who submit *diverse* collections of pipelines tend to perform better than those who submit less diverse collections of pipelines? The answer for the Winter 2020 evaluation is effectively no. While there does exist statistically significant effect of increased diversity on increased score, the effect size is trivial. Thus, it seems that less diverse collections of pipelines, i.e. those that focus more on hyperparameter tuning, can as effective (or almost as effective, on average) as those that explore a greater diversity and/or ordering of primitives.

1 Introduction

In section 2, *Measures of Diversity*, we define several measures of the diversity within a collection of three or more pipelines. In section 3, *Results at a Glance*, we present a heat map showing briefly the (weak) connection between diversity and performance. In section 4, *Quantifying the Effect of Diversity*, we report on regression results which point to the measurable, though substantively trivial, effect of diversity on score. In a final section 5, *Visualizing Collections* we show a visualization of collections of pipelines for a problem together with a multiple alignment, for context.

2 Measures of Diversity

Before we can define *diversity* of a collection of (say 3 to 20) pipelines, we first need to define the distance between a pair of pipelines. We choose the Levenshtein edit distance as this measure between two pipelines. Specifically, we first express the pipelines as sequences of primitives, where primitives are written as "letters" in a large alphabet. The software we use accommodates all D3M primitives with two-letter pairs, each pair representing a single token (primitive) of the alphabet. The Levenshtein edit distance is the minimum number of substitutions, insertions and/or deletions needed to bring one sequence to coincide with the other. This measure satisfies the axioms for a distance.

But *distance* involves just two pipelines; *diversity* measures variation among a collection of three or more pipelines. We tried several alternatives for quantifying diversity. The most well-behaved measures (i.e. the ones that behave most closely to our expectations on synthetic data) were vector norms where the vector components were the Levenshtein distances for all possible unordered pairs of pipelines in the collection.

More precisely, we used l_p norms where p was a parameter varying between 1 and ∞. The different norms measure slightly different quantities. At the extreme, the l_∞ norm (maximum edit-distance component) is large when there is at least one pair of pipelines at great distance, regardless of the positions (great or small) of the other as-close or closer distances. On the other hand, the l_1 norm (sum of the edit-distance components) can still be relatively large if there are many pairs pipelines at moderate distance, even though there is no pair at great distance. We point out that the choice the different norms can sometimes matter: we have noted that the choice can order sets of collections—synthetic or real—differently in terms of diversity.

In the l_p norm, what value for the parameter p do we pick? If you are using only one measure of diversity, we recommend the l_2 norm as a nice trade off between the two extremes. That said, it is more informative to report two or more measures of diversity, in which case l_1 and l_∞ should be preferred because they are most independent.

Model	Coefficient	p-value	estimate	std error
A	l_1 (z-score)	0.00716 **	1.984e-01	7.355e-02
B	l_1 (z-score)	0.00763 **	1.984e-01	7.414e-02
C	l_1 (z-score)	multiple models, same story.		
D	l_1 (z-score)	0.00716 **	1.984e-01	7.355e-02
E	l_1 (z-score)	0.0155 *	1.778e-01	7.329e-02

Table 1: Summary of models predicting the z-score of a performer's best score in a submitted collection, based on l_1 and l_∞ measures of diversity of the collection and count of number of pipelines in the collection. We performed a regression analysis with 5 different models (A-E, see text). The table shows the model, the coefficient examined (in this case, always considered just l_1, among other predictors we used: l_∞ and count). Furthermore, the table shows p-values (two tailed, under the null hypothesis that coefficient is zero), estimates of coefficients and standard errors of the estimates. Note that several entries of the table are identical to our numerical precision. We attribute this result to category-level predictors that do not covary with response variable. We examine the effect of colinearity between l_1 and l_∞ and found that it did not change our results substantially. Significance codes: 0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 *[blank]* 1.

3 Results at a Glance

Figure 1 shows the diversity of performer pipeline submissions by problem type. The color-coding indicates diversity, with purple colors indicating small l_2 edit distance norms, trending towards yellow colors indicating larger l_2 edit distance norms.The number of asterisks in each cell captures the count of times a pipeline in that performer-category was the best (or tied). For instance, in the second row of the first column, UCB twice had the best-performing pipeline on a data set in the binary_classification problem category. Overall, the figure suggests (and we quantify below) a very small but measurable effect of diversity.

4 Quantifying the Effect of Diversity

We use regression-based methods to quantify the effect of diversity on performance. This analysis is complicated by several factors, which we address by various, overlapping methodological choices. Overall, our goal is to specify predictive models of performance on the bases of the l_1 and l_∞ norms, as well as the number of pipelines submitted. Our unit of analysis is the collection of pipelines submitted by a performer, which features diversity measures mentioned, a count, and the score obtained by its best member. We group these by problem across performer for analysis. The diversity measures and scores are not comparable across problems, or problem types; thus, we move to measuring all predictors and scores on a per-problem z-score basis.

Below, we report on five classes of statistical models. In the first, we group all problem types together for analysis in a "complete pooling" model; see MODEL A.

First, ran a complete pooling model including fixed effects for problem type, and saw no change in our key predictors.

Second we ran a fixed effects model (MODEL B) on category, where problems were classified in terms of type of problem (e.g. time series, binary classification, etc).

We then ran a no pooling model (MODEL C), which is actually numerous separate models, one for each problem-type category.

We then ran a hierarchical model with random effects on the intercept (MODEL D).

Finally, we ran a hierarchical model with random effects on both the intercept and slope (MODEL E).

In all models we see that there is a significant effect of diversity on performance, but the coefficient is small (less than .2) To interpret this result, an increase in l_1 diversity of one correponds to an increase of the z-score of the max score by less than 0.2.

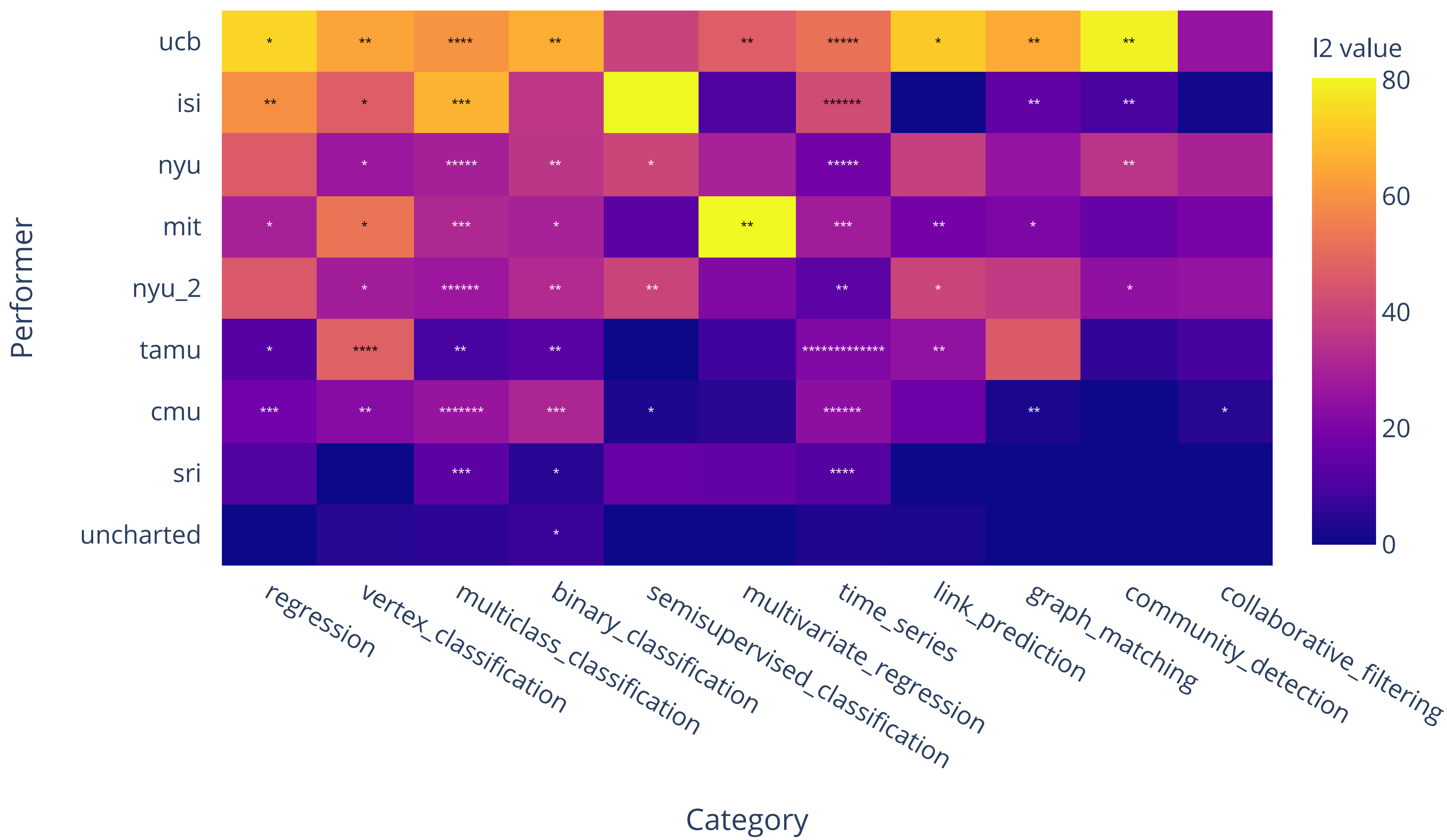

Figure 1: Heat map of l_2 diversity see section “Measures of Diversity” for a discussion of this and other measures we use to quantify diversity. The horizontal axis is the problem category and the vertical axis is performer for the Winter 2020 evaluation. The number of asterisks in each cell is the number of problem instances in the corresponding category-performer which was the best (or tied for best) performing pipeline for any problem. There were 37 ties (including multi-way ties), corresponding to an additional 37 asterisks in the figure beyond the total number of 103 problems.

See
Following
Pages

Figure 2: The next several pages show a cartoon figure of a multiple sequence alignment of all submitted pipelines (all performers) for one randomly selected problem **semi_1217_click_prediction_small_MIN_METADATA**. The circles in the first pane illustrate the pipelines and are referenced with the number that corresponds to the label in the multiple alignment in the second pane. The colors correspond to different performers (also labeled in the second pane). The alphabet used to label primitives is shown on the third pane, which was randomly generated. The position of the circles was determined by a spring layout procedure with pipelines close in edit distance supposedly remain close as nodes. That said, usually impossible solve this problem perfectly (at least in two dimensions) and we have noted that the software returns some misleading results while presumably trying to draw a pretty picture (what it was designed to do). Nevertheless, we found these figures useful for keeping pipelines in memory—as long as the image is interpreted as a "cartoon" and not a "topographical map." Cartoons for the other problems are available upon request.

5 Visualizing Collections

In this final section, we show a cartoon visualization of collections of pipelines for a problem together with a multiple alignment (see Figure 2, and its caption).