Improving Feature Type Inference Accuracy of TFDV with SortingHat

Vraj Shah
University of California, San Diego
vps002@eng.ucsd.edu

Kevin Yang
University of California, San Diego
khy009@eng.ucsd.edu

Arun Kumar
University of California, San Diego
arunkk@eng.ucsd.edu

ABSTRACT

ML feature type inference is a critical data preparation (prep) step when applying ML over structured data. Features could be *Numeric*, *Categorical*, or something else. Tensorflow Data Validation (TFDV) uses conservative heuristics to infer feature types from the descriptive statistics about the column. It wrongly calls many *Categorical* features with integer values as *Numeric*, e.g. *ZipCode*. In this work, we discuss and compare different approaches to type inference to improve the accuracy of TFDV for columns with integer domain. We find that an ML-based approach using our tool, which we call SortingHat is more accurate than many rule-based heuristics. Moreover, we perform an empirical comparison of different approaches on a suite of downstream benchmark tasks. We find that the wrong feature type inference can often lead to a significant decrease in the downstream model's accuracy relative to their true accuracy. More importantly, we find that SortingHat can even help TFDV to improve the accuracy of the downstream model. Finally, we integrate the best performing model built on our labeled data with TFDV version 0.21.2.

1 FEATURE TYPE INFERENCE VOCABULARY MAPPING

We formalized the task of ML feature type inference by creating the first ever benchmark labeled dataset [5]. Our dataset has 9921 examples and a 9-class label vocabulary. The complete description of our label vocabulary is provided in the technical report [5]. Figure 1 shows the feature type vocabulary of existing tools such as TFDV [3], TransmogrifAI in Salesforce Einstein [2], and AutoGluon from Amazon AWS [4] and how they map to our label vocabulary.

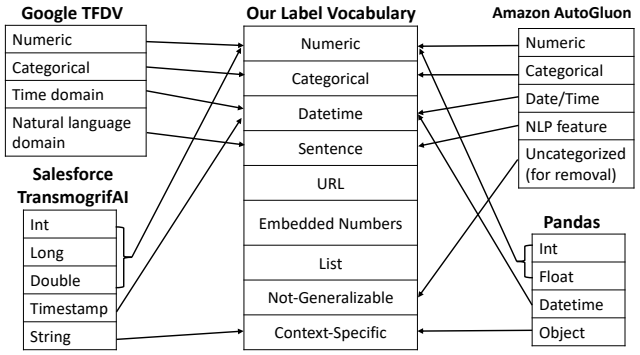

Figure 1: Feature type vocabulary mapping of TFDV, Pandas, TransmogrifAI, and AutoGluon to our vocabulary

Descriptive Stats
Total number of values
Number of nans and % of nans
Number of unique values and % of unique values
Mean and std deviation of the column values, word count, stopword count, char count, whitespace count, and delimiter count
Min and max value of the column
Regular expression check for the presence of url, email, sequence of delimiters, and list on the 5 sample values
Pandas timestamp check on 5 sample values

Table 1: List of descriptive statistics features

2 APPROACHES

The focus of this work is to study different type inference approaches for columns with integer domain on top of TFDV's existing type inference pipeline. Specifically, we identify the feature type of a column using each approach. We then populate the "schema.int_domain.is_categorical" field in the schema proto with the *True* value if the inferred type is *Catgeorical*. We discuss 4 approaches below.

(1) TFDV + Rule on number of distinct value. If the number of distinct values in a column is less than or equal to a certain threshold, then we mark such columns as *Categorical*. We tune the threshold using the methodology given in Section 4.1.

(2) TFDV + Rule on percentage of distinct values. If the percentage of unique values in a column is less than or equal to a certain threshold, then we mark such columns as *Categorical*. We tune the threshold using the methodology given in Section 4.1.

(3) TFDV + Rule on mutual information value. We compute the mutual information of the column with the target column. We randomly shuffle the column values and again compute their mutual information with the target. If the difference is exactly 0 then we mark the column as *Categorical*. We discuss why picking a threshold on mutual information is non-trivial in Section 4.3.

(4) TFDV + SortingHat. SortingHat is our tool that uses our best performing Random Forest model to automatically infer the feature types from a raw CSV file. Our Random Forest model uses the column name, 5 randomly sampled unique values from the column, and 25 descriptive statistics for a column such as percentage of distinct values, percentage of NaNs, mean, standard deviation, minimum value, maximum value, and an average number of whitespace-separated tokens. We provide the complete list of these 25 features in Table 1.

3 SORTINGHAT INTEGRATION WITH TFDV

To compute the descriptive stats, we use the same apache beam pipeline but with a separate run over the data. This allows us to avoid having to edit the various generators present in the existing codebase. In the generate_statistics_from_csv method in the stats_gen_lib.py file, we added our code to grab and count distinct values using "beam.transforms.util.Distinct" and "beam.combiners.Count.Globally." To gather the 5 sample values, we use the method: "Sample.FixedSizeGlobally(5)." Next, we created a sorting_hat.proto file to hold the values that we extract in the above pipeline. With all of this established, we process the data and populate each individual proto, while finally creating a list of them and returning it to the user. Once we have both the statistics proto as well as the sortinghat proto, we send them to the infer_schema method in the validation_api.py file. Since both protos hold the attribute name field, we can use that to merge the two protos into a vector representation. Using this vectorized data, we compute several descriptive stats that we need to run our Random Forest model. If the output of our model is *Categorical*, we populate the "schema.int_domain.is_categorical" field in each schema proto. Overall, we have made changes to the stats_gen_lib.py and validation_api.py files to integrate our SortingHat model with TFDV version 0.21.2.

How to use our integration code?

(1) Pull TFDV version 0.21.2.

(2) Replace stats_gen_lib.py (under utils/) and validation_api.py (under api/) files in their respective locations with our files. The usage of the "generate_statistics_from_csv" function now has an extra parameter of "sorting_hat_output_path" which is a path of a file that serves as a sink for the apache beam output. The "infer_schema" function has now two extra parameters: "sorting_hat_stats" and "model_name." The former represents the SortingHat proto and the later represents the name of the model with the current default being 'rf' (denoting Random Forest). In addition, the "infer_schema" function returns a dictionary of predictions made by the Random Forest model on the columns with integer domain. The key of this dictionary denotes a feature type from our 9-class label vocabulary and the value is a list of the columns inferred with that type.

(3) Add the sorting_hat_pb2.py file under tensorflow-metadata's proto/v0 folder with the other proto files. Add the RandomForest model and the dictionary file under the directory: "api/".

(4) We provide an example script to test our integration code in tfdv-test.py.

4 EMPIRICAL STUDY

4.1 Threshold Tuning

We curated and hand-labeled a large meta-dataset of 9921 columns from 1240 real data files with 9-class vocabulary for benchmarking feature type inference [5]. We use our labeled dataset to select 3447 columns with integer domain. We use this subset of our labeled data to find the threshold for the rule-based approaches. Note that the integer domain columns can correspond to any label from 5 classes: *Numeric*, *Catgeorical*, *Datetime*, *Not-Generalizable*, and *Context-Specific*. We partition our labeled data subset into a train and test set with an 80:20 ratio. We use a standard grid search for tuning the threshold where the train set is used for validation. We pick the threshold with the highest train accuracy.

# Distinct Values	Train	Test
2	0.4977	0.4617
3	0.5717	0.5428
5	**0.5901**	**0.5560**
10	0.5782	0.5310
15	0.5757	0.5339
20	0.5706	0.5265
25	0.5681	0.5206
50	0.5544	0.5044
75	0.5457	0.5015
100	0.5287	0.4926
150	0.5078	0.4912
200	0.4930	0.4779

% Distinct Values	Train	Test
0.1	0.5334	0.4912
0.2	**0.5749**	**0.5369**
0.5	0.5457	0.5133
0.75	0.5294	0.5059
1	0.5139	0.4941
2	0.4872	0.4690
3	0.4771	0.4543
5	0.4623	0.4484
10	0.4373	0.4366
20	0.4142	0.4041
30	0.4020	0.3835
60	0.3749	0.3540

Table 2: 5-class train and test accuracy on our labeled dataset (only the integer columns) with different thresholds on the absolute number of distinct values and percentage distinct values in the column

Table 2 presents the grids and 5-class train and test accuracy on different thresholds of absolute domain size and the percentage of unique values in the column. We find that the training accuracy is highest with a grid value of 5 on the absolute number of distinct values. Thus, we set its threshold to be 5. Similarly, we pick the threshold value of 0.2 on the percentage of distinct values. We present the cumulative distribution of the absolute number of unique values, total values, and the percentage of unique values for the integer columns of our labeled dataset in the appendix.

4.2 Datasets

We use 23 datasets from our downstream benchmark suite that has at least one integer column [5]. In total, there are 19 classification tasks and 4 regression tasks. The datasets and their source details are available on our Github repo [1].

4.3 Mutual Information Approach

The approach based on mutual information requires the user to manually annotate the target column in the dataset. Our labeled dataset does not contain information about the target for any of the labeled examples. For many columns in our labeled dataset, it is not trivial to decide the target column in their corresponding source data file. Thus, it is not clear how a threshold on mutual information should be picked.

We show the scatter plots of mutual information value between every column in the downstream datasets and their corresponding target in Figure 2 (A). We find that categorical and numerical columns are distinctively separated by their domain size, where numerical columns have typically more number of unique values

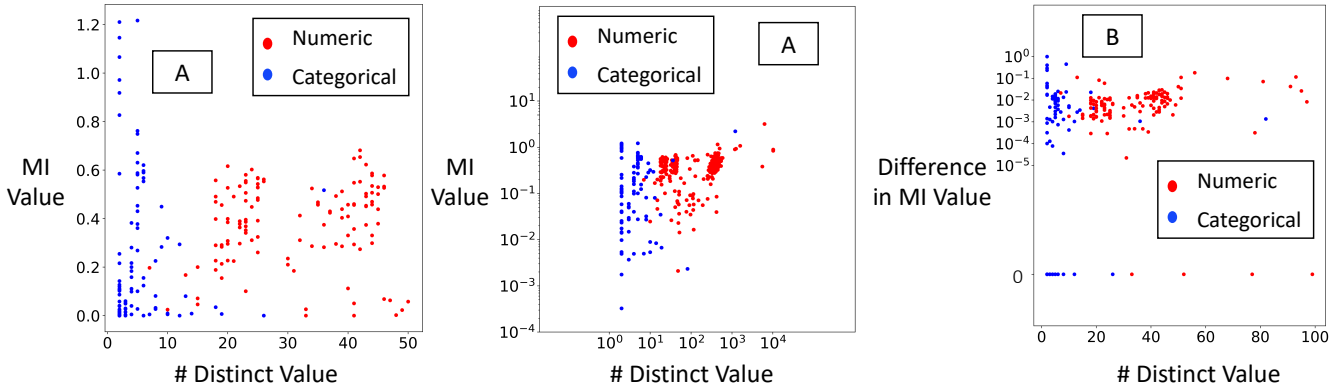

Figure 2: (A) Scatter plots of mutual information of *Numeric* and *Categorical* columns of the downstream datasets with their respective target. (B) Scatter plot of difference in mutual information between original columns with target and shuffled columns with target on the downstream datasets.

than categorical attributes. However, we do find a clear demarcation of mutual information value between numeric and categorical columns. Figure 2 (B) shows the difference in mutual information value between the original raw columns with target and shuffled raw columns with the target. We observe that there are more categorical columns with a mutual information difference of zero than numeric columns. But it is not clear how we can quantify the mutual information change in terms of the type of the column and/or data properties.

4.4 Type Inference Accuracy Results

Table 4 presents the type inference accuracy of all approaches on 23 downstream datasets. We observe that the accuracy of standard TFDV is almost 83%. A rule-based approach on the absolute number of distinct values with TFDV leads to a lift in the accuracy of nearly 6% compared to standard TFDV. A rule-based approach on the percentage of distinct values performs only marginally better than standard TFDV. Overall, among all approaches, the type inference accuracy of TFDV is highest with the SortingHat approach.

Mfeat dataset has 216 *Numeric* integer columns, containing almost 50% of the total columns. Thus, excluding *Mfeat* dataset, the overall type inference accuracy drops to 66% with standard TFDV. Moreover, we find that the overall accuracy with the rule-based approach on the absolute domain size is higher with a threshold of 20 than with a threshold of 5. However, on including *Mfeat*, the overall accuracy of the same rule-based approach with a threshold of 5 is higher than a threshold of 20. Thus, rule-based approaches are highly sensitive to a chosen threshold and dataset characteristics.

4.5 Downstream Benchmark Suite

We now empirically study the benefit of doing feature type inference accurately with TFDV-based approaches on the downstream model accuracy. We refer the reader to our technical report for a comprehensive downstream benchmark with other tools and other feature types [5].

	TFDV		TFDV + Rule (#DV <=5)		TFDV + SH	
	LR	RF	LR	RF	LR	RF
Underperform truth	16	14	14	13	11	11
Match or outperform truth	7	9	9	10	12	12
Match TFDV	-		17	17	8	9
Outperform TFDV			5	3	12	10
Best performing approach for a dataset	12	13	16	13	21	18

Table 3: Number of downstream datasets (out of 23) where tools underperform, match, or outperform the ground truth downstream performance, or standard TFDV, or the best performing tool. *#DV* is the number of distinct values in the column. SH is our SortingHat tool that uses the Random Forest for type inference trained on our data. LR denotes downstream linear model (Logistic/Linear regression) and RF denotes downstream Random Forest.

4.5.1 Models and Metrics. In terms of downstream model evaluation, we present both extremes of bias-variance tradeoff: L2-regularized Logistic regression (high bias, low variance) for classification, L2-regularized Linear regression (high bias, low variance) for regression, and Random Forest (low bias, high variance) for both classification and regression. Thus, we have 46 downstream models in total. We use the accuracy metric scaled to 100 for the classification tasks and root mean squared error (RMSE) metric for the regression tasks.

4.5.2 Downstream Model Performance. Table 5 presents the end-to-end comparison of downstream models built with feature types inferred by two approaches: TFDV with a rule on the absolute domain size with a threshold of 5 (TFDV + DV) and TFDV with SortingHat (TFDV + SortingHat). We compare their downstream performance relative to that with feature types inferred with standard TFDV and true feature types. Table 3 presents summary

Feature Types	Raw Attribute Types	Dataset	|Y|	|A|	|AC|	Type Inference Accuracy					
						TFDV	TFDV + Rule (#DV <=5)	TFDV + Rule (#DV <=20)	TFDV + Rule (%DV <=0.2)	TFDV + MI (diff = 0)	TFDV + SH
NU	Int, Float	Cancer	2	9	9	100	+0	+0	+0	-33.3	+0
	Int	Mfeat	10	216	216	100	+0	-10.2	+0	-0.5	-3.2
CA	Int	Hayes	3	4	4	0	+100	+100	+0	+100	+100
	Int	Supreme	2	7	7	71.4	+0	+14.3	+0	+0	+28.6
	Int, String	Flares	2	10	10	30	+70	+70	+0	+10	+70
	Int, String	Kropt	18	6	6	50	+16.7	+50	+16.7	+16.7	+50
	Int, String	Boxing	2	3	3	66.7	+0	+33.3	+0	+0	+0
CA + NG	Int, String	Apnea2	2	3	3	66. 7	+0	+33.3	+0	+0	+0
NU + CA	Int, String	Flags	2	28	28	60.7	17.9	+39.3	+0	+14.3	+25
	Int,Float,String	Diggle	2	8	8	87.5	+0	+12.5	+0	+0	-25
	Int, Float	Hearts	2	13	13	61.5	+38.5	+38.5	+0	+23.1	+38.5
	Int, Float	Sleuth	2	10	10	80	+20	-10	+0	+20	+20
NU + CA + ST	Int, String	Auto-MPG	3	8	8	62.5	+12.5	+25	+0	+0	+12.5
NU + CA + ST + NG	Int,String,PK	Clothing	5	10	9	66.7	+0	+0	+11.1	+0	+11.1
NU + DT + NG	Int, String, Time, PK	IOT	2	4	2	100	+0	+0	-50	+0	-50
NG + CA	Int,String, PK	Zoo	5	17	13	92.3	+7.7	+7.7	+0	+0	+7.7
NU + CA + EN + NG	Int,Float,String	PBCseq	2	18	12	83.3	+8.4	+8.4	+8.4	+16.7	-8.3
NU + CA + LST + NG + CS	Int, Float, String, PK	Pokemon	36	40	35	37.1	+0	+11.5	+0	-2.8	+51.5
NU +CA+DT+ URL +NG+CS	Int,Float,Date, String, Time	President	57	26	16	100	+0	-6.2	-6.2	+0	+0
CA	Int	MBA	R	2	2	50	+0	+0	+0	+50	+50
NU + CA	Int	Vineyard	R	3	3	66. 7	+0	+0	+0	-33.4	+33.3
	Int, String	Apnea	R	3	3	66. 7	+0	+33.3	+0	+0	+0
NU + CA + EN + NG	Int, String	Car Fuel	R	11	6	100	+0	+0	+0	-16.7	+0
Mean accuracy on 23 downstream datasets				449	426	**83.3**	+6.3	+4.5	+0.2	+2.6	+9.9
Mean accuracy on 22 datasets (Mfeat excluded)				233	210	**66.2**	+12.9	+19.5	+0.5	+5.7	+23.3

Table 4: Type Inference Accuracy of different approaches on the downstream datasets. *Numeric* (*NU*), *Categorical* (*CA*), *Datetime* (*DT*), *Sentence* (*ST*), *Not-Generalizable* (*NG*), *Embedded Number* (*EN*), *URL*, *List* (*LST*), and *Context-Specific* (*CS*) are feature types. |A| is the number of columns/attributes in that dataset. |AC| is the number of columns covered by TFDV with its type inference vocabulary. |Y| is the number of target classes. *R* denotes regression tasks and PK denotes primary keys. *#DV* and *%DV* are the number of distinct values and the percentage of distinct values in the column respectively. *MI* and *SH* denotes the mutual information-based and SortingHat approach (using Random Forest trained on our labeled data) respectively.

statistics on how these approaches perform relative to the ground truth, standard TFDV, and best performing tool for a given dataset. We find that for a given dataset and a downstream model, TFDV + SortingHat underperforms the best performing approach for only 7 out 46 downstream models, in contrast to 17 for TFDV + DV. In addition, TFDV + SortingHat outperforms the standard TFDV for 22 downstream models, in contrast to 8 for TFDV + DV.

From Table 5, we find that wrong type inference often leads to a drop in accuracy compared to the accuracy with true feature types. For instance, wrong type inference leads standard TFDV to underperform on 30 out of 46 downstream models. This lead to a drop of

(A) Feature Types	Dataset	|Y|	Logistic Regression				Random Forest			
			Truth	TFDV *% change over Truth*	TFDV + Rule (#DV <=5) *% change over TFDV*	TFDV + SH *% change over TFDV*	Truth	TFDV *% change over Truth*	TFDV + Rule (#DV <=5) *% change over TFDV*	TFDV + SH *% change over TFDV*
NU	Cancer	2	60.8	+0	+0	+0	66.7	+0	+0	+0
	Mfeat	10	92.5	+0	+0	-2.7	91.8	+0	+0	-2.3
CA	Hayes	3	74.1	-14.1	+14.1	+14.1	78.5	+0	+0	+0
	Supreme	2	99.3	-17.1	+0	+17.1	99.4	+0	+0	+0
	Flares	2	90.8	+0	+0	+0	89.2	+0.3	-0.3	-0.3
	Kropt	18	39.4	-6.9	+0	+6.9	68.8	-3.4	+0.4	+3.4
	Boxing	2	80.7	-25.2	+0	+0	78.5	-11.9	+0	+0
CA + NG	Apnea2	2	92	-0.6	+0	+0	90.1	-0.8	+0	+0
NU + CA	Flags	2	68.2	-3.6	+0.5	+3.1	75.9	-2.6	+0	+0.5
	Diggle	2	99.9	+0	+0	-5.5	99.9	+0	+0	-0.3
	Hearts	2	84.9	-1.6	+1.6	+1.6	86.2	-3	+3	+3
	Sleuth	2	68.9	-3.3	+3.3	+3.3	76.7	+0	+0	+0
NU + CA + ST	Auto-MPG	3	89.1	-8.6	+0	-7.3	95.2	-18.9	-0.3	-1.5
NU + CA + ST + NG	Clothing	5	66.7	-9.1	+0	+0	64.2	-4.9	+0	+1
NU + DT + NG	IOT	2	83.8	+0	+0	+0	93.8	+0	+0	+0
NG + CA	Zoo	5	75.6	-11.1	+8.9	+8.9	77.8	-8.9	+4.4	+2.2
NU + CA + EN + NG	PBCseq	2	68.6	+0.5	-1.4	+8.5	73	-0.1	-0.1	+2.2
NU + CA + LST + NG + CS	Pokemon	36	65.84	-52.4	+0	+1.7	88.1	-3.2	+0	+1.4
NU +CA+DT+ URL +NG+CS	President	57	39.5	-7.9	+0	+1.5	81.7	-23.1	+0	+0.2

(B) Feature Types	Dataset	Linear Regression – L2 Regularization				Random Forest			
		Truth	TFDV *change over Truth*	TFDV + Rule (#DV <=5) *change over TFDV*	TFDV + SH *change over TFDV*	Truth	TFDV *change over Truth*	TFDV + Rule (#DV <=5) *change over TFDV*	TFDV + SH *change over TFDV*
CA	MBA	0.363	+0.05	-0	-0.05	0.384	+0.08	-0	-0.08
NU + CA	Vineyard	2.97	+2	-0	-2	2.7	+0.37	-0	-0.37
	Apnea	2206.2	-0	-0	-0	1355.7	-0	-0	-0
NU + CA + EN + NG	Car Fuel	11.3	+0.16	-0	-0	11.7	+1.1	-0	-0

Table 5: Accuracy comparison of downstream models using inferred types from Random Forest trained on our labeled data (SH) and rule-based approach on the number of distinct values, against standard TFDV, relative to accuracy with true feature types. Datasets involve (A) Classification tasks with accuracy metric (B) Regression tasks with RMSE metric. *Numeric* (*NU*), *Categorical* (*CA*), *Datetime* (*DT*), *Sentence* (*ST*),*Not-Generalizable* (*NG*), *Embedded Number* (*EN*), *URL*, *List* (*LST*), and *Context-Specific* (*CS*) are feature types. \|Y\| is the number of target classes. PK denote primary keys. *#DV* is the number of distinct values in the column. *SH* denotes the SortingHat approach (using Random Forest trained on our labeled data).

an average 6.4% and up to 52% in accuracy compared to the ground truth-based model. We find that the drop in accuracy due to wrong type inference is higher for a linear model (average of 8.5%) compared to a higher-capacity Random Forest model (average of 4.2%). Overall, TFDV + SortingHat and TFDV + DV approaches underperform the truth for 22 and 27 models respectively. Thus, SortingHat can even help TFDV to improve its downstream performance.

5 PUBLIC RELEASE

We have released a public repository on GitHub with our entire labeled data for the ML feature type inference task [1]. Also, we released the raw 1240 CSV files that we used to create our labeled data. The repository also contains the downstream benchmark suite with all the datasets and the associated code for running the benchmark.

6 TAKEAWAYS

(1) From our end-to-end experiments on our labeled data, we find that Random Forest model (scikit-learn implementation) achieves the highest 9-class classification accuracy compared to models like CNN, Logistic Regression, RBF-SVM, and k-NN. The difference in accuracy on our held-out test set between Random Forest and CNN is almost 5%, which is significant. Thus, we decided to integrate Random Forest rather than the CNN with TFDV.

(2) The integration code we wrote is for TFDV version 0.21.2. Thus, it would need to be ported to the latest TFDV version.

(3) We believe that TFDV can benefit by expanding their type inference vocabulary to include *Not-Generalizable* type. *Not-Generalizable* columns do not contain any informative values to be useful as features. e.g., primary keys, columns with only one unique value, etc. Such columns offer no discriminative power and are thus useless. For more examples, please refer to our tech report [5]. From our downstream benchmark experiments in the tech report, we find that using such features that offer no discriminative power to build the downstream model, leads to a drop in accuracy compared to excluding them completely for building the model. Thus, recognizing such columns correctly can improve downstream performance.

REFERENCES

[1] 2020. Github Repository for ML Feature Type Inference https://github.com/pvn25/MLDataPrepZoo/tree/master/MLFeatureTypeInference.

[2] 2020. TransmogrifAI: Automated machine learning for structured data https://transmogrif.ai/.

[3] Denis Baylor et al. 2017. Tfx: A tensorflow-based production-scale machine learning platform. In *SIGKDD*. ACM.

[4] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint arXiv:2003.06505* (2020).

[5] Vraj Shah, Jonathan Lacanlale, Premanand Kumar, Kevin Yang, and Arun Kumar. 2020. Towards Benchmarking Feature Type Inference for AutoML Platforms (Technical Report). https://adalabucsd.github.io/papers/TR_2020_SortingHat.pdf.

APPENDIX

A CUMULATIVE DISTRIBUTION ON OUR LABELED DATA

Figure 3, Figure 4, and Figure 5 presents the cumulative distribution of the number of distinct values, total values, and the percentage of distinct values for all integer columns by class in our labeled dataset. Figure 6 presents the mutual information value of all integer columns in our downstream benchmark suite by class.

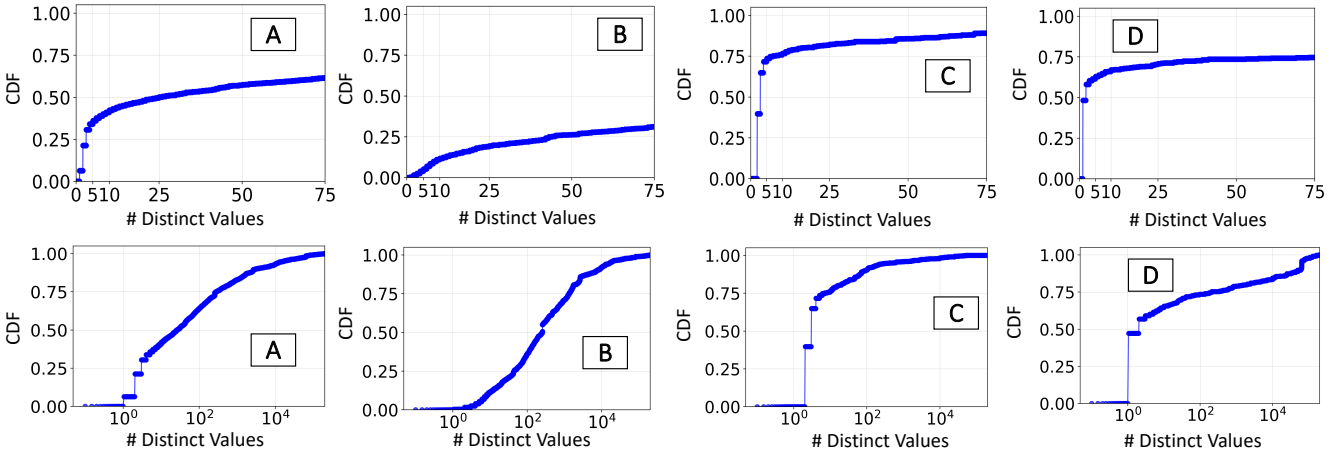

Figure 3: Cumulative distribution of the number of distinct values for all integer columns in our labeled dataset that belongs to (A) Any of the 5 class, (B) *Numeric* class, (C) *Categorical* class, and (D) *Not-Generalizable* class.

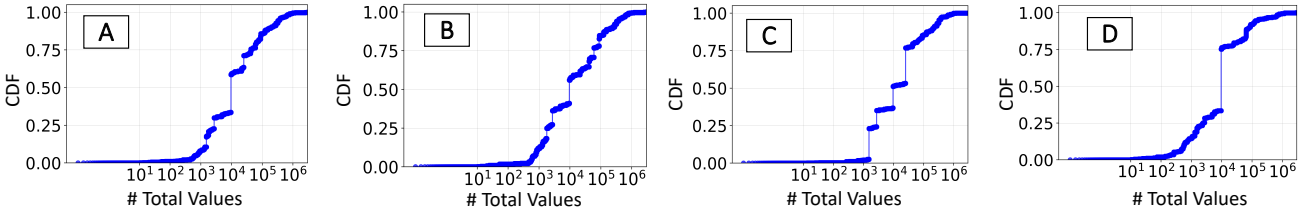

Figure 4: Cumulative distribution of the total number of values for all integer columns in our labeled dataset that belongs to (A) Any of the 5 class, (B) *Numeric* class, (C) *Categorical* class, and (D) *Not-Generalizable* class.

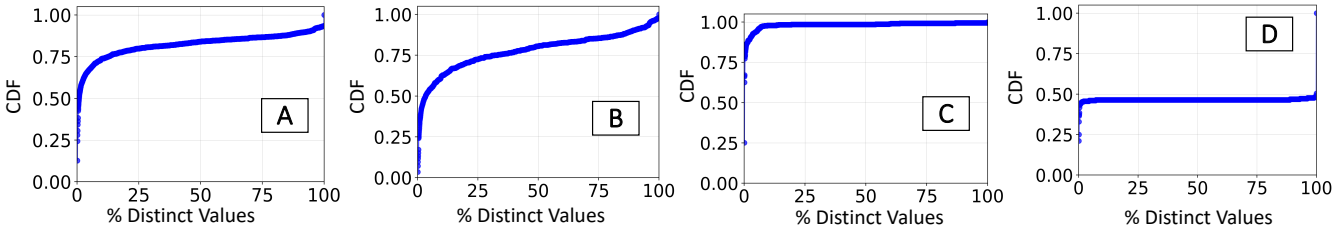

Figure 5: Cumulative distribution of percentage of distinct values for all integer columns in our labeled dataset that belongs to (A) Any of the 5 class, (B) *Numeric* class, (C) *Categorical* class, and (D) *Not-Generalizable* class.

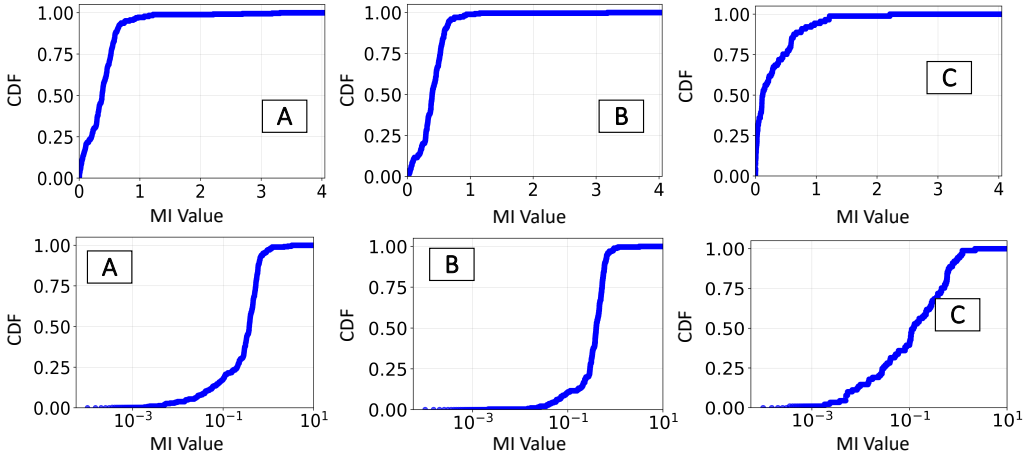

Figure 6: Cumulative distribution of the mutual information value for all integer columns in our downstream benchmark suite that belongs to (A) Any class, (B) *Numeric* class, and (C) *Categorical* class.