Benchmarking Multimodal AutoML for Tabular Data with Text Fields

Xingjian Shi[*] xjshi@amazon.com
Jonas Mueller[*] jonasmue@amazon.com
Nick Erickson neerick@amazon.com
Mu Li mli@amazon.com
Alexander J. Smola alex@smola.org
Amazon Web Services

Abstract

We consider the use of automated supervised learning systems for data tables that not only contain numeric/categorical columns, but one or more text fields as well. Here we assemble 18 multimodal data tables that each contain some text fields and stem from a real business application. Our publicly-available benchmark[2] enables researchers to comprehensively evaluate their own methods for supervised learning with numeric, categorical, and text features. To ensure that any single modeling strategy which performs well over all 18 datasets will serve as a practical foundation for multimodal text/tabular AutoML, the diverse datasets in our benchmark vary greatly in: sample size, problem types (a mix of classification and regression tasks), number of features (with the number of text columns ranging from 1 to 28 between datasets), as well as how the predictive signal is decomposed between text vs. numeric/categorical features (and predictive interactions thereof). Over this benchmark, we evaluate various straightforward pipelines to model such data, including standard two-stage approaches where NLP is used to featurize the text such that AutoML for tabular data can then be applied. Compared with human data science teams, the fully automated methodology[3] that performed best on our benchmark (stack ensembling a multimodal Transformer with various tree models) also manages to rank 1st place when fit to the raw text/tabular data in two MachineHack prediction competitions and 2nd place (out of 2380 teams) in Kaggle's Mercari Price Suggestion Challenge.

1 Introduction

Despite recent data proliferation, the practical value of machine learning (ML) remains hampered by an inability to quickly translate raw data into accurate predictions. Automatic Machine Learning (AutoML) aims to address this via pipelines that can ingest raw data, train models, and output accurate predictions, all without human intervention [30]. Given their immense potential, many AutoML systems exist for data structured in tables, which are ubiquitous across science/industry [20, 26, 49].

Many data tables contain not only numeric and categorical fields (together referred to as *tabular* here), but also fields with free-form text. For example, Table 1 depicts actual data from the website Kickstarter. These contain multiple text fields such as the title and description of each funding proposal, numerical fields like the goal amount of funding and when the proposal was created, as well as categorical fields like the funding currency or country. This paper considers tables of this form

[*] Equal contribution.
[2] Benchmark is available at: `https://github.com/sxjscience/automl_multimodal_benchmark`
[3] Open-source available to easily run on your own data: `https://github.com/awslabs/autogluon`

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

name	desc	goal	country	currency	created_at	final_status
The Secret Order - The Game that gives back Gl...	Can you trust your friends? Solve the puzzle? ...	5000.0	GB	GBP	1424101105	0
Booker Family Foods. Home made, the way food s...	Community based, home-made-foods producer, to ...	2500.0	US	USD	1404617242	0
J.A.E.S.A : Next Generation Artificial Intelli...	A true next generation AI with the ability to ...	30000.0	CA	CAD	1399078600	1

Table 1: Example of data in our multimodal benchmark with text (*name*, *desc*), numeric (*goal*, *created_at*), and categorical (*country*, *currency*) columns. From these features, we want to predict if a Kickstarter project will reach its funding goal or not (*final_status*).

where rows contain IID training examples (each with a single numeric/categorical value to predict, i.e. regression/classification) and the columns used as predictive *features* can contain text, numeric, or categorical values. We refer to the value in a particular row and column as a *field*, where a single text field may actually contain a long text passage (e.g. a multi-paragraph item description). Despite their potential commercial value, there are currently few (automated) solutions for machine learning with this sort of data that jointly contain numeric/categorical and text features, which we here refer to as *multimodal* or *text/tabular* data. Applying existing AutoML tools to such data requires either manually featurizing text fields into tabular format [5, 24], or ignoring the text. Alternatively, one can model just the text with existing natural language processing (NLP) tools. [10, 22, 23, 29, 43].

This paper provides foundational tools aiming to spur a practical line of research that evaluates fundamental design choices for automated supervised learning with multimodal datasets that jointly contain text, numeric, and categorical features. Even though text commonly appears along with numeric/categorical fields in enterprise data tables, how to model such multimodal data has not been well studied in the literature. This stems from a lack of public benchmarks, as well as existing beliefs that basic featurization of the text should suffice for tabular models to exhibit strong performance [13, 24]. Here we introduce a new benchmark of 18 multimodal text/tabular datasets involving regression/classification tasks related to real business applications (Section 3), and provide a first systematic evaluation of some generic strategies for supervised learning with such data (Section 5).

Note that we write *AutoML* to describe any single modeling strategy that remains robustly performant across a diverse set of datasets without manual adjustments. The construction of an effective AutoML system critically relies on having an empirical benchmark of diverse datasets that are representative of real applications the system will be subsequently used for (in order to ensure the system performs well on the right types of data and not only on certain limited types of data). The experiments over our benchmark presented in this paper merely entail a preliminary evaluation of various straightforward (automated) multimodal modeling strategies that today's data scientists might consider for supervised learning with text/tabular data. Among other discoveries, our benchmark reveals that the conventional strategy of neural embeddings to featurize text for tabular models can be outperformed by simple alternatives. The strategy found to perform best in our benchmark (stack ensembling of tabular models with a multimodal Transformer network) should serve as a strong foundation for multimodal text/tabular AutoML[4], whose efficacy was subsequently verified in a few data science competitions (demonstrating that our benchmark and analysis have led to important new insights). That said, much further research is needed in this area, and we hope the public benchmark and open-source tooling introduced here will facilitate practical advances in important text/tabular modeling applications.

2 Related Work

Many ML courses teach data as vectors in $\mathbb{R}^d$, which is not the case in many practical applications. Thanks to the ubiquity of relevant benchmark data, substantial research has been conducted for properly handling categorical features in a unified manner that generalizes across datasets without sacrificing accuracy [21, 35, 40]. Given the prevalence of tabular data composed of numeric/categorical features, automating the ML process for such data has been the subject of extensive inquiry with major practical impact [20, 26, 30]. We hope our benchmark spurs similar progress on how to effectively handle text features in tabular data.

[4] A tutorial to easily run this method on your own text/tabular data is provided at: `https://auto.gluon.ai/stable/tutorials/tabular_prediction/tabular-multimodal-text-others.html`

Today, tools for automated learning with text data remain scarce (e.g. this dearth forced Blohm et al. [5] to turn to tabular AutoML tools for automated text prediction). Instead modern NLP applications primarily require experts who mostly favor Transformer networks as their model of choice for text [12, 41, 43]. However existing methods to input numeric/categorical features into Transformers remain rudimentary [43] and fail to outperform the best tree models for tabular prediction [28]. While multimodal text/tabular Transformer models have been utilized for *table understanding* tasks such as: semantic parsing of facts, cell filling, or relation extraction [11, 57], how to best adapt these models for standard classification/regression tasks with text/tabular features remains unstudied to our knowledge (a question that our benchmark might help answer). The use of tabular models together with Transformer-like text architectures has received limited attention [33, 53], and it remains unclear how to optimally leverage their complementary strengths for multimodal data (due to lack of benchmarks). In contrast, a number of entirely-neural architectures have been proposed for multimodal settings [31, 44, 45, 56]. However the vast majority of these are for $\{$image, text$\}$ data [2, 42, 46, 47], but the gap between neural networks and alternative models is far greater for images than for tabular data [28]. In short, it remains unclear what are the best generic ML pipelines for text/tabular data based on models available today.

Large, sufficiently diverse/representative, public benchmarks have spurred significant progress in tabular AutoML [14, 15, 20, 59] and NLP [18, 34, 39, 54]. However we are not aware of any analogous benchmarks for evaluating multimodal text/tabular ML. There do exist a few miscellaneous text/tabular datasets scattered throughout popular ML data repositories [1, 51], but these are mostly small academic datasets that are not representative of modern applications with significant practical value. In contrast, multiple prediction competitions each involving a single real-world text/tabular dataset have been held, but winning solutions have heavily relied on dataset/domain-specific tricks of limited generalizability (c.f. mercari [38] and jigsaw dataset descriptions in Section C). Here we aggregate multimodal datasets from competitions and other industry sources into one benchmark that aims to reveal unifying principles for powerful generic modeling of this form of data.

3 A Benchmark for Supervised Learning with Text/Tabular Data

In designing practical systems for real-world data tables that often contain text, the empirical performance of our design decisions is what ultimately matters. Representative benchmarks comprised of many diverse datasets are critical for proper evaluation of AutoML, whose aim is to reliably produce reasonable accuracy on arbitrary datasets without manual user-tweaking. Thus we introduce the first public benchmark for evaluating multimodal text/tabular ML, which is comprised of 18 tabular datasets, each containing at least one text field in addition to numeric/categorical columns. Our new benchmark is publicly available, as is the code to reproduce all results presented here.

Our benchmark strives to represent the types of ML tasks that commonly arise in industry today. In creating the benchmark, we aimed to include a mix of classification vs. regression tasks and datasets from real applications (as opposed to toy academic settings) that contain a rich mix of text, numeric, and categorical columns. Table 2 shows it is comprised of datasets that are quite

Dataset ID	#Train	#Test	#Cat.	#Num.	#Text	Task	Metric	Prediction Target
prod	5,091	1,273	1	0	1	multiclass	accuracy	sentiment associated with product review
salary	15,841	3,961	1	0	5	multiclass	accuracy	salary range in data scientist job listings
airbnb	18,316	4,579	37	24	28	multiclass	accuracy	price label of Airbnb listing
channel	20,284	5,071	1	15	1	multiclass	accuracy	news category to which article belongs
wine	84,123	21,031	0	2	3	multiclass	accuracy	which variety of wine (type of grape)
imdb	800	200	0	7	4	binary	roc-auc	whether film is a drama
fake	12,725	3,182	2	0	3	binary	roc-auc	whether job postings are fake
kick	86,502	21,626	3	3	3	binary	roc-auc	whether proposed Kickstarter project will achieve funding goal
jigsaw	100,000	25,000	2	27	1	binary	roc-auc	whether social media comments are toxic
qaa	4,863	1,216	1	0	3	regression	R^2	subjective type of answer (in relation to question)
qaq	4,863	1,216	1	0	3	regression	R^2	subjective type of question (in relation to answer)
book	4,989	1,248	1	2	5	regression	R^2	price of books
jc	10,860	2,715	0	2	3	regression	R^2	price of JC Penney products on their website
cloth	18,788	4,698	2	1	3	regression	R^2	customer review score for clothing item
ae	22,662	5,666	3	2	6	regression	R^2	price of American-Eagle inner-wear items on their website
pop	24,007	6,002	1	2	1	regression	R^2	online popularity of news article
house	37,951	9,488	1	18	20	regression	R^2	sale price of houses in California
mercari	100,000	25,000	3	0	6	regression	R^2	price of Mercari online marketplace products

Table 2: The 18 multimodal datasets that comprise our benchmark. '#Cat.', '#Num.' and '#Text' count the number of categorical, numeric, and text features in each dataset, and '#Train' (or '#Test') count the training (or test) examples. In PDF, click on each Dataset ID for link to original data source.

diverse in terms of: sample-size, problem types, number of features, and type of features. 11 of the datasets contain more than one text field (with 28 text fields in the airbnb dataset). These text fields greatly vary in the amount of text they contain (e.g. short product names vs. lengthy product descriptions/reviews). The data (and text vocabulary) stem from a mix of of real-world domains spanning: e-commerce, news, social media, question-answering, and product listings (jobs, projects, films, Airbnb). Subsequent accuracy results from Table 3 indicate the 18 prediction problems also vary greatly in terms of both difficulty and how the predictive signal is divided between text/tabular modalities. To reflect real-world ML issues, we processed the data minimally (beyond ensuring the features/labels correspond to meaningful prediction tasks without duplicate examples) and thus there are arbitrarily-formatted strings and missing values all throughout. Methods/systems that perform well across the diverse set of 18 benchmark datasets are thus likely to provide real-world value for an important class of applications.

Each dataset in our benchmark is provided with a prespecified training/test split (usually 20% of the original data reserved for test set). Methods are not allowed to access the test set during training, and for validation (model-selection, hyperparameter-tuning, etc.) instead must themselves hold-out some data from the provided training data. As the choice of training/validation split is a key design decision in AutoML, we leave this flexible for different systems to choose in the learning process. To facilitate comparison between the different AutoML pipelines presented in this paper, we always used the same AutoGluon-provided training/validation split, which is stratified based on labels in classification tasks. Our use of other AutoML frameworks beyond AutoGluon (e.g. H2O) allows each framework to choose their own data splitting scheme.

The benchmark GitHub repository contains: (i) methods to easily retrieve the individual datasets and train/test splits, (ii) code to run all of the ML strategies studied in this paper and reproduce our results, and (iii) the scripts we used to produce each benchmark dataset from the original data source. Common modifications made to original data sources to produce the benchmark dataset versions included: defining a practically meaningful prediction task if there was not one associated with the original dataset, omitting duplicated rows, omitting non-predictive features (e.g. user ID) and those that were too correlated with the prediction target (making the benchmark too easy otherwise), applying log-transform to prediction targets that correspond to product prices (log-scale errors are more meaningful in most real pricing applications), and down-sampling overly large datasets (mercari, jigsaw) to ensure the benchmark remains computationally accessible. Additional descriptions of each dataset are provided in Appendix C.

4 Text/Tabular Modeling Pipelines

Using our benchmark, we conduct a systematic empirical analysis of various strategies for modeling text/tabular data. The strategy that performs best across the benchmark can serve as a promising starting point for automated supervised learning with multimodal data tables that contain text. Key choices include what models to use (and for which features), and how to optimally combine different models within an overall supervised learning pipeline. Our study considers popular modeling paradigms used by practictioners today, including: NLP models to featurize text for tabular models [5, 13, 24], ensembling of independently-trained text and tabular models [38], or end-to-end learning with neural networks that jointly operate on inputs across text and tabular modalities [31, 43, 44]. Below we merely outline the candidate modeling strategies, Appendix A provides full descriptions.

(Multimodal) Transformer Networks Given their dominance across NLP, the only models we consider for handling raw text are popular Transformer neural networks which have initially been pretrained in an unsupervised fashion over a massive text corpus [9, 12, 37, 43, 52, 54]. We investigate how the end-to-end deep learning paradigm can be leveraged for simultaneous text and tabular inputs by extending standard Transformer networks into *multimodal Transformer networks* that jointly operate on both text and tabular features. Three multimodal network variants depicted in Figure S2 are considered: (1) *All-Text* – in which all tabular features are converted to strings and input into the Transformer as text, (2) *Fuse-Early* – in which dense embedding layers map the tabular features into the same vector space as the embedded text tokens such that self-attention and other Transformer layers can be applied to learn low-level interactions across modalities, (3) *Fuse-Late* – a multi-tower network where one branch is a Transformer network for the text, other branches are multilayer perceptrons (MLP) for the numeric/categorical inputs, and the higher-level vector

representations of each branch are pooled into a single multimodal vector representation (near the output layer of the overall network) via concatenation.

Combining Transformers and Tabular Models As shown in Figure S3a, we consider featurizing the text into vector format followed by subsequent application of various tabular models [5]. Here the text embedding may stem from a pretrained Transformer network that has not been fine-tuned on our data (*Pre-Embedding*), a Transformer only trained on our text fields alone (*Text-Embedding*), or a multimodal Transformer network trained on both our text and tabular fields (*Multimodal-Embedding*). Note that 'tabular models' throughout are those trained on only numeric/categorical features, e.g. types of tree-based models. We also consider simple weighted ensembles that linearly combine the predictions of our Transformer network and various tabular models (*Weighted-Ensemble*, shown in Figure S3b) where each model takes as input the modalities it is suited for and is independently trained from the other models [7, 14]. Finally, stack ensembling is alternatively considered to nonlinearly aggregate predictions from the Transformer and tabular models (*Stack-Ensemble*, shown in Figure S3c), where an additional tabular 'stacker' model is trained using as its features the predictions output by the independently-trained Transformer and original tabular models [14, 50].

In our study, all tabular (numeric/categorical) modeling is simply done via AutoGluon-Tabular, an easy to use open-source tool for automated supervised learning on tabular data [14]. We chose AutoGluon because it has been found to produce highly accurate models for diverse tabular datasets [4, 16, 17, 58]. AutoGluon trains and ensembles a diverse suite of popular models for tabular data, including: Gradient Boosted Decision Trees [8, 32, 40], Extremely Randomized Trees [19], and MLP Neural Networks [14]. While neural networks are typically favored for unstructured data like text, decision tree ensembles have proven to be one of the most consistently performant models for tabular data [3, 16, 28]. Thus an effective strategy for text/tabular AutoML may need to appropriately combine the complementary strengths of Transformers and (tree-based) tabular models.

5 Experiments

To keep our study tractable, we adopt a sequential decision making process that decomposes the overall supervised learning pipeline design into three stages: 1) determine the appropriate Transformer backbone and fine-tuning strategy for text data alone (Appendix A.1), 2) determine the best way to extend this Transformer to text and tabular inputs (Appendix A.2), and 3) determine the best method to combine the best text and tabular models (Appendix A.3 and A.4). At each subsequent stage of the study, we explore modeling choices that are specific to that stage and simply use the best choice found in the empirical comparisons of the options available in previous stages.

For straightforward comparison, we employ the most commonly used classification/regression evaluation metrics that lie in $[0, 1]$ for reasonable predictions, with higher values indicating superior performance. We evaluate regression tasks via the coefficient of determination R^2, multiclass classification tasks via accuracy, and binary classification tasks via area under the ROC curve (AUC).

Choice of Transformer Backbone Our first decision concerns the Transformer network itself, including what architecture and pretraining objective to employ. Existing results may not translate to our setting, since Transformers are typically applied to datasets with at most a couple text fields per training example [54, 55]. Here we choose between the (standard, already pretrained) base version of RoBERTa [37] or ELECTRA [9], two popular backbones used across modern NLP applications.

We first fine-tune the pretrained Transformer models as our sole predictors, using only the text features in each dataset. This helps identify which model is better at handling the types of text in our multimodal datasets. During fine-tuning of both of the RoBERTa or ELECTRA networks, we additionally consider two tricks to boost performance: 1) Exponentially decay the learning rate of the network parameters based on their depth [48]. We use a per-layer learning rate multiplier of τ^d in which d is the layer depth and τ is the decay factor (set $= 0.8$ throughout). 2) Average the weights of the models loaded from the top-3 training checkpoints with the best validation scores [52].

The first section of Table 3 shows that ELECTRA performs better than RoBERTa across the text columns in our benchmark datasets. Our exponential decay and checkpoint-averaging tricks further boost performance, with the majority of additional gains produced by exponential decay. In subsequent experiments, we thus fix ELECTRA fine-tuned with both exponential decay and checkpoint-averaging as the model used to handle text features and call it *Text-Net*.

Method	prod	qaq	qaa	cloth	airbnb	ae	mercari	jigsaw	imdb	fake	kick	jc	wine	pop	channel	salary	book	house	avg ↑	mrr ↑
Choosing Text-Net:	NLP Backbones and Fine-tuning Tricks (Section A.1)																			
RoBERTa	0.588	0.412	0.268	0.700	0.344	0.953	0.561	0.960	0.731	0.929	0.751	0.615	0.811	-0.000	0.301	0.396	0.151	0.821	0.572	0.07
ELECTRA	0.705	0.410	0.356	0.718	0.349	0.955	0.586	0.965	0.750	0.824	0.754	0.606	0.813	0.003	0.315	0.457	0.466	0.857	0.605	0.09
+ Exponential Decay $\tau = 0.8$	0.728	0.436	0.431	0.743	0.337	0.953	0.579	0.963	0.852	0.963	0.760	0.664	0.808	0.004	0.308	0.447	0.568	0.841	0.632	0.12
+ Average 3 ★	0.729	0.451	0.432	0.746	0.350	0.954	0.581	0.965	0.858	0.961	0.766	0.656	0.807	0.004	0.307	0.445	0.571	0.841	0.635	0.14
Choosing Multimodal-Net:	Fusion Strategy (Section A.2, Figure S2)																			
All-Text	0.907	0.454	0.419	0.746	0.366	0.957	0.599	**0.967**	0.840	0.967	**0.799**	0.645	0.810	0.013	0.480	0.465	0.585	0.892	0.662	0.28
Fuse-Early	**0.913**	0.441	0.418	0.745	0.377	0.953	0.596	**0.967**	0.843	0.960	0.770	0.653	0.806	0.013	0.474	0.458	0.548	0.901	0.658	0.21
Fuse-Late ★	0.907	0.449	**0.445**	0.747	0.395	0.958	0.603	0.966	0.857	0.961	0.773	0.639	0.812	0.015	0.481	0.468	0.571	0.907	0.664	0.22
Choosing Aggregation:	Multimodal Model Aggregation (Sections A.3 and A.4, Figure S3)																			
Pre-Embedding	0.895	0.216	0.247	0.642	0.449	0.972	0.433	0.586	0.871	0.926	0.743	0.491	0.680	0.012	0.526	0.460	0.581	0.939	0.593	0.11
Text-Embedding	0.867	0.446	0.432	0.748	0.430	0.972	0.434	0.587	0.855	0.962	0.790	0.658	0.830	0.008	0.502	0.438	0.594	0.932	0.638	0.17
Multimodal-Embedding	0.907	0.439	0.437	0.749	0.438	0.974	0.432	0.587	0.847	0.967	0.794	**0.683**	0.829	0.007	0.517	0.451	0.595	0.934	0.644	0.25
Weighted-Ensemble	0.907	0.439	0.429	0.744	0.453	0.976	0.597	0.957	0.876	0.923	0.787	0.641	0.814	0.018	0.554	0.483	0.620	0.941	0.676	0.25
Stack-Ensemble ★	0.909	**0.456**	0.438	**0.751**	0.459	0.977	**0.605**	**0.967**	**0.878**	0.964	0.797	0.624	0.836	**0.020**	**0.556**	0.496	**0.638**	**0.943**	**0.684**	**0.69**
	Tabular AutoML + Feature Engineering Baselines (Section A.3)																			
AG-Weighted	0.891	0.046	0.076	-0.002	0.426	0.841	0.098	0.587	0.845	0.686	0.668	0.004	0.173	0.016	0.549	0.226	0.222	0.934	0.405	0.08
AG-Stack	0.891	0.046	0.077	0.001	0.435	0.841	0.098	0.587	0.844	0.697	0.670	0.003	0.175	0.017	0.550	0.226	0.233	0.934	0.407	0.09
AG-Weighted+ N-Gram	0.892	0.426	0.382	0.610	0.450	0.978	0.526	0.909	0.842	0.966	0.772	0.357	0.829	0.019	0.546	0.484	0.591	0.941	0.640	0.17
AG-Stack+ N-Gram	0.895	0.414	0.383	0.654	**0.466**	**0.979**	0.569	0.915	0.850	**0.968**	0.775	0.612	**0.842**	**0.020**	0.548	0.494	0.600	0.943	0.663	0.43
H2O AutoML	0.869	0.247	0.159	0.163	0.329	0.976	0.430	0.531	0.813	0.756	0.669	0.411	0.478	0.014	0.530	0.525	0.444	0.939	0.516	0.11
H2O AutoML + Word2Vec	0.859	0.244	0.285	0.624	0.347	0.973	0.534	0.847	0.827	0.943	0.755	0.443	0.778	0.013	0.524	**0.528**	0.586	0.932	0.613	0.14
H2O AutoML + Pre-Embedding	0.846	0.227	0.312	0.644	0.367	0.969	0.282	0.572	0.874	0.893	0.738	0.549	0.571	0.007	0.483	0.483	0.523	0.933	0.572	0.09

Table 3: Accuracy (and R^2, AUC) of AutoML strategies over our multimodal benchmark. Column **avg** lists each method's average score across datasets (i.e. how *much* methods differ in overall performance) and **mrr** its mean reciprocal rank among all evaluated methods (i.e. how *often* a method outperforms others). Each subsection encapsulates a design stage (★ marks variant with best avg).

Best Multimodal Network Next, we explore the best way to extend the *Text-Net* model to operate across numeric/categorical inputs in addition to text fields (among the options in Figure S2).

Across our datasets, Table 3 shows that the *Fuse-Late* strategy outperforms *Text-Net* and the alternative *All-Text*/*Fuse-Early* options for producing predictions from multimodal inputs using a single neural network. We thus fix this *Fuse-Late* model as our *Multimodal-Net* used in subsequent experiments.

Aggregating Transformers and Tabular Models Having identified a good neural network architecture for multimodal text/tabular inputs, we now study combinations of such models with classical learning algorithms for tabular data (among the options in Figure S3). Where not specified, the tabular models are those trained by AutoGluon-Tabular (see Appendix B.4). The third section of Table 3 illustrates that *Stack-Ensemble* is overall the best aggregation strategy. Ensembling the predictions of *Multimodal-Net* and tabular models is better than instead using the Transformer for text embedding. As expected, *Text-Embedding* and *Multimodal-Embedding* outperform *Pre-Embedding*, demonstrating how domain-specific fine-tuning improves the quality of learned embeddings. *Multimodal-Embedding* performs better than *Text-Embedding* on some datasets and similarly across the rest, showing it can be beneficial to use text representations contextualized on numeric/categorical information.

AutoGluon Baselines As many of our results use the tabular models in AutoGluon [14], we also compare different variants of AutoGluon-Tabular (without our *Multimodal-Net*) as baselines:

AG-Weighted / *AG-Stack*: We train AutoGluon with weighted / stack ensembling of its tabular models, here ignoring all text columns. Thus, baseline ML performance of tabular models without using any text fields can be established via the AG-Weighted/Stack numbers in Table 3 (without N-Gram).

AG-Weighted + N-Gram / *AG-Stack + N-Gram*: Similar to *AG-Weighted* / *AG-Stack*, except we first use AutoGluon's N-Gram featurization [13] to encode all text in tabular form.

The performance gap between AutoGluon-Tabular with and without N-Grams can reveal (an approximate lower bound for) how much extra predictive value is provided by the text features in each dataset. Inspecting these gaps, we find that, compared to the tabular features, text features contain most of the predictive signal in some datasets (qaq, qaa, cloth, mercari, jc), and far less signal in other datasets (prod, imdb, channel), again highlighting the diversity of our benchmark. Note that our proposed *Stack-Ensemble* performs relatively well across all types of datasets, regardless how the predictive signal is allocated between text and tabular features.

H2O Baselines In addition to AutoGluon, we also run another popular open-source AutoML tool offered in H2O [25]. Since H2O AutoML is not designed for the text in our multimodal data tables, we try combining H2O's NLP tool [24] and tabular AutoML tool [36].

H2O AutoML: We run H2O AutoML directly on the original data of our benchmark. As a tabular AutoML framework, H2O AutoML is assumed to ignore text features, but H2O categorizes feature types differently than us and automatically treats some columns we consider to be text as categorical instead.

H2O AutoML + Word2Vec: We run H2O's word2vec algorithm to featurize text fields and then H2O AutoML on the featurized data, following their recommended procedure [24].

H2O AutoML + Pre-Embedding: We featurize each text field using embeddings from a pretrained ELECTRA Transformer, as in *Pre-Embedding*, followed by H2O AutoML on the featurized data table.

The last section of Table 3 shows that while these powerful AutoML ensemble predictors can outperform our individual neural network models (particularly for datasets with more predictive signal in the tabular features), our *Stack-Ensemble* and *Weighted-Ensemble* are superior overall.

Table 3 shows the accuracy for some datasets significantly improves when models utilize the text features rather than ignoring them. Predictive performance of *AG-stack* (baseline tabular model that ignores text) vs. *Stack-Ensemble* (our extension that leverages text) is 0.098 vs. 0.605 on mercari, 0.670 vs. 0.797 on kick, and 0.175 vs. 0.836 on wine. On these datasets, modeling the tabular features brings clear improvements over the text alone given the performance of *Text-Net* (our best text Transformer model that ignores tabular features) is only: 0.581 on mercari, 0.766 on kick, and 0.807 on wine. In certain applications, accuracy improvements of this magnitude may have significant commercial value, thus highlighting the benefits of multimodal modeling of text/tabular data.

Performance in Real-world ML Competitions Some datasets in our multimodal benchmark originally stem from previous ML competitions. For these (and other recent competitions with text/tabular data), we fit the automated strategy that performed best in our benchmark (*Stack-Ensemble*) to the official competition dataset, without manual adjustment or data processing. We then submit its resulting predictions on the competition test data to be scored, which enables us to see how they fare against the manual efforts of human data science teams.

The *Stack-Ensemble* strategy achieves 1st place historical leaderboard rank in two MachineHack prediction competitions: *Product Sentiment Classification*[5] and *Predict the Data Scientists Salary in India*[6], and 2nd place in another: *Predict the Price of Books*[7]. This same strategy also achieves 2nd place on the historical leaderboard of two Kaggle competitions: *California House Prices*[8] and *Mercari Price Suggestion Challenge*[9], where the latter was a very popular Kaggle competition in which 2380 teams participated (with $100,000 prize offered to winner). These results show that a straightforward AutoML strategy identified via preliminary analysis of our benchmark is already competitive with data scientists on real-world text/tabular datasets that possess great commercial value. Extensive studies of the benchmark will presumably reveal even more effective strategies.

Feature Importance Analysis Feature importance can help us understand what drives a ML system's accuracy and whether text fields in a dataset are worth their overhead. For two representative datasets from our benchmark, we compute *permutation feature importance* [6] for our trained models, which is defined as the drop in prediction accuracy after values of only this feature (which are entire text fields for a text column) are shuffled in the test data (across rows). We only shuffle original column values so our importance scores are not biased by preprocessing/featurization decisions (except in how these directly affect model accuracy).

Figure 1 shows that both our *Multimodal-Net* and *Stack-Ensemble* containing this network may rely more heavily on text features than the *AG-Stack+N-Gram* baseline. With more powerful modeling of text fields, models often begin to rely more heavily on the text fields. An exception here is the *brand_name* feature in mercari, but this feature usually contains just a single word in its fields. Furthermore, the *Multimodal-Net* places less importance on the tabular features, demonstrating how purely neural network approaches are less effective for modeling numeric/categorical data compared to alternative tree-based tabular models. It is thus useful to combine both multimodal-Transformer and tabular models in order to ensure we are most effectively modeling both the text and tabular features.

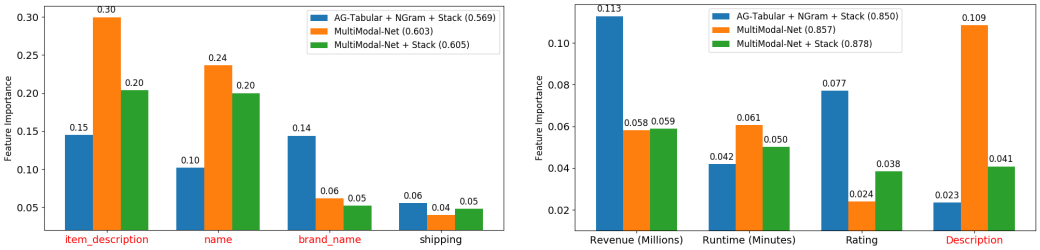
[5] `https://www.machinehack.com/hackathons/product_sentiment_classification_weekend_hackathon_19/overview` ("Anonymous Submission ID 1556" entry)

[6] `https://machinehack.com/hackathons/predict_the_data_scientists_salary_in_india_hackathon/overview` ("Xingjian Shi" entry)

[7] `https://machinehack.com/hackathons/predict_the_price_of_books/overview`

[8] `https://www.kaggle.com/c/california-house-prices` ("sxjscience" entry)

[9] `https://github.com/sxjscience/automl_multimodal_benchmark/blob/main/competition_submissions/mercari_submission_screenshot.png`



(a) Permutation importance in "mercari".

(b) Permutation importance in "imdb".

Figure 1: Importance of text vs. tabular features for three models in two datasets (text features in red). Here *MultiModal-Net* + *Stack* corresponds to the *Stack-Ensemble* method from Figure S3c.

6 Discussion

Lacking public benchmarks, academic research on ML for multimodal text/tabular data has not matched industry demand to derive practical value from such data. This paper provides evidence that generic best practices for such data remain unclear today: we simply evaluated a few basic strategies on our benchmark and found a single automated strategy that turns out to outperform human data scientists in numerous historical prediction competitions involving diverse text/tabular data. This strategy uses a stack ensemble (Appendix A.4) of tabular models trained on top of predictions from other tabular models and a *Multimodal-Net* (depicted in Figure S3c). The latter network is based on a *Fuse-Late* architecture (depicted in Figure S2c) with concatenation of text, numeric, and categorical representations (where text representations are produced via the ELECTRA Transformer backbone) and is trained via fine-tuning with exponential learning rate decay and checkpoint averaging.

Note that this stack ensemble strategy was simply the method that performed best over our benchmark datasets. The fact that this strategy is also highly competitive in numerous text/tabular prediction competitions highlights the utility of our benchmark in revealing performant modeling techniques, indicating that the benchmark is sufficiently diverse and representative of real-world text/tabular prediction tasks. Our benchmark analysis challenges certain conventional beliefs:

- Neural embedding of text followed by tabular modeling (*Pre/Text-Embedding*) [5, 24] is often outperformed by N-gram featurization (*AG-Stack* + *N-Gram*) or leveraging predictions from text neural networks (*Stack-Ensemble*) rather than their representations (embeddings). Given the success of pretrained Transformers across NLP, we are surprised to find both N-Grams and word2vec here provide superior text featurization than *Pre-Embedding*.
- In the architecture of multimodal networks for classification/regression, newer ideas to fuse modalities in early layers (i.e. *Fuse-Early/All-Text* Transformers with cross-modality attention [27, 43, 46]) are not necessarily superior to older multi-tower *Fuse-Late* architectures that fuse representations in higher layers closer to the output [2, 31, 44].
- An end-to-end multimodal neural network is surpassed by stack ensembling this *Multimodal-Net* with tabular models trained in separate stages rather than end-to-end (*Stack-Ensemble*).

Previously anticipated conclusions that are empirically validated by our benchmark include:

- Text featurization is better via fine-tuned networks (*Text-Embedding*) than pretrained ones (*Pre-Embedding*), and slightly better via a fine-tuned multimodal network (*Multimodal-Embedding*), whose text embeddings benefit from contextualization on the tabular features.
- Able to exploit predictive interactions between different modalities, stack ensembling outperforms simple weighted ensembling, yet it still facilitates modular system design.

Further analysis of our text/tabular benchmark can reveal many more practical ML insights, including under which data conditions certain methods perform better than others. Future research should investigate different data preprocessing pipelines, which are known to play an important role in AutoML. Other important questions not considered in our preliminary study include *how to best*: Handle many long text fields? Perform multimodal feature selection? Apply feature engineering that combines synergistically with learned neural network representations? Allocate limited training/HPO time between cheaper tabular models and more expensive text neural networks? We hope our public benchmark spurs the AutoML community to broaden their methods' applicability to more data types.

References

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2007. URL http://archive.ics.uci.edu/ml.

[2] N. Audebert, C. Herold, K. Slimani, and C. Vidal. Multimodal deep networks for text and image-based document classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 427–443. Springer, 2019.

[3] S. Bansal. Data science trends on kaggle. https://www.kaggle.com/shivamb/data-science-trends-on-kaggle#5.-XgBoost-vs-Keras, 2018.

[4] O. Bezrukavnikov and R. Linder. A neophyte with automl: Evaluating the promises of automatic machine learning tools. *arXiv preprint arXiv:2101.05840*, 2021.

[5] M. Blohm, M. Hanussek, and M. Kintz. Leveraging automated machine learning for text classification: Evaluation of automl tools and comparison with human performance. *arXiv preprint arXiv:2012.03575*, 2020.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *International Conference on Machine Learning*, 2004.

[8] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 785–794. ACM, 2016.

[9] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[10] G. Cloud. Features and capabilities of automl natural language. 2021. URL https://cloud.google.com/natural-language/automl/docs/features.

[11] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. Turl: table understanding through representation learning. *Proceedings of the VLDB Endowment*, 14(3):307–319, 2020.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT*, 2019.

[13] J. Eisenstein. Natural language processing, 2018.

[14] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[15] H. J. Escalante, W.-W. Tu, I. Guyon, D. L. Silver, E. Viegas, Y. Chen, W. Dai, and Q. Yang. Automl@ neurips 2018 challenge: Design and results. In *The NeurIPS'18 Competition*, pages 209–229. Springer, 2020.

[16] R. Fakoor, J. Mueller, N. Erickson, P. Chaudhari, and A. J. Smola. Fast, accurate, and simple models for tabular data via augmented distillation. In *Advances in Neural Information Processing Systems*, 2020.

[17] S. Feldman. Which machine learning classifiers are best for small datasets? an empirical study. https://www.data-cowboys.com/blog/which-machine-learning-classifiers-are-best-for-small-datasets, 2021.

[18] S. Gehrmann, T. Adewumi, K. Aggarwal, P. S. Ammanamanchi, A. Anuoluwapo, A. Bosselut, K. R. Chandu, M. Clinciu, D. Das, K. D. Dhole, et al. The GEM benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*, 2021.

[19] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1): 3–42, 2006.

[20] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren. An open source AutoML benchmark. In *ICML Workshop on Automated Machine Learning*, 2019.

[21] C. Guo and F. Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[22] J. Guo, H. He, T. He, L. Lausen, M. Li, H. Lin, X. Shi, C. Wang, J. Xie, S. Zha, et al. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7, 2020.

[23] S. Gupta and V. Khare. Enhanced text classification and word vectors using amazon sagemaker blazingtext. 2018. URL `https://aws.amazon.com/blogs/machine-learning/enhanced-text-classification-and-word-vectors-using-amazon-sagemaker-blazingtext/`.

[24] H2O.ai. NLP with H2O. `https://docs.h2o.ai/h2o-tutorials/latest-stable/h2o-world-2017/nlp/index.html`, 2017.

[25] H2O.ai. *H2O, Fast Scalable Machine Learning, for python*, January 2021. URL `https://github.com/h2oai/h2o-3`. version 3.32.0.3.

[26] X. He, K. Zhao, and X. Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[27] R. Hu and A. Singh. UniT: Multimodal multitask learning with a unified transformer. *arXiv preprint arXiv:2102.10772*, 2021.

[28] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[29] HuggingFace. Auto training and fast deployment for state-of-the-art nlp models. 2021. URL `https://huggingface.co/autonlp`.

[30] F. Hutter, L. Kotthoff, and J. Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. 2018.

[31] H. Jin, Q. Song, and X. Hu. Auto-keras: An efficient neural architecture search system. In *KDD*, 2019.

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 2017.

[33] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *KDD*, 2019.

[34] S. M. Lakew, M. Negri, and M. Turchi. Low resource neural machine translation: A benchmark for five african languages. *arXiv preprint arXiv:2003.14402*, 2020.

[35] M. Larionov. Sampling techniques in Bayesian target encoding. *arXiv preprint arXiv:2006.01317*, 2020.

[36] E. LeDell and S. Poirier. H2o automl: Scalable automatic machine learning. In *ICML Workshop on Automated Machine Learning*, 2020.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[38] K. Lopuhin and P. Jankiewicz. 1st place solution to mercari price suggestion challenge. `https://github.com/pjankiewicz/mercari-solution`, 2018.

[39] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[40] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, 2018.

[41] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26, 2020.

[42] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[43] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[44] J. Rodriguez. pytorch-widedeep, deep learning for tabular data i: data preprocessing, model components and basic use. `https://jrzaurin.github.io/infinitoml/2020/12/06/pytorch-widedeep.html`, 2020.

[45] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

[46] A. Singh, V. Goswami, V. Natarajan, Y. Jiang, X. Chen, M. Shah, M. Rohrbach, D. Batra, and D. Parikh. Mmf: A multimodal framework for vision and language research. `https://github.com/facebookresearch/mmf`, 2020.

[47] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. VideoBERT: A joint model for video and language representation learning. In *ICCV*, pages 7464–7473, 2019.

[48] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune BERT for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[49] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. In *IEEE 31st international conference on tools with artificial intelligence*, 2019.

[50] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

[51] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL `http://doi.acm.org/10.1145/2641190.2641198`.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[53] A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, H. Jin, S. Petryk, S. A. Bargal, and J. E. Gonzalez. NBDT: Neural-backed decision tree. In *International Conference on Learning Representations*, 2021.

[54] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, 2019.

[55] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

[56] Y. Wang, S. Joty, M. R. Lyu, I. King, C. Xiong, and S. C. Hoi. VD-BERT: A unified vision and dialog transformer with BERT. In *Empirical Methods in Natural Language Processing*, 2020.

[57] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, 2020.

[58] J. Yoo, T. Joseph, D. Yung, S. A. Nasseri, and F. Wood. Ensemble squared: A meta automl system. *arXiv preprint arXiv:2012.05390*, 2020.

[59] M.-A. Zöller and M. F. Huber. Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research*, 70:409–472, 2021.

Appendix

A Descriptions of Text/Tabular Modeling Pipelines

Here we fully describe the various straightforward modeling strategies that we evaluate over our benchmark in order to identify performant baselines for automated supervised learning with multi-modal data tables that contain text. Recall our study aims to cover popular variants of text/tabular modeling used in practice today, including: NLP models to featurize text for tabular models [5, 13, 24], ensembling of independently-trained text and tabular models [38], or end-to-end learning with neural networks that jointly operate on inputs across text and tabular modalities [31, 43, 44]. We first consider the latter paradigm of multimodal neural network models, which in subsequent sections are also considered for text featurization and ensembling with tabular models.

A.1 Transformer Models for Text

We first consider solely inputting the text into our neural network and then discuss how to extend the network to additional numeric/categorical inputs in Section A.2. While many neural architectures have been proposed to model text, pretrained Transformer networks now dominate modern NLP. These models are first pretrained in an unsupervised manner on a massive text corpus before being fine-tuned over our (smaller) labeled dataset of interest [12, 43]. This allows our supervised learning to benefit from information gleaned from the external text corpus that would otherwise not be available in our limited labeled data. The Transformer also effectively aggregates information from various aspects of a training example, using a *self-attention* mechanism to contextualize its intermediate representations based on particularly informative features [52]. Since BERT [12] first demonstrated the power of Transformer pretraining via Masked Language Modeling (MLM), superior pretraining techniques have been developed. RoBERTa [37] dynamically generates masks and pretrains on a larger corpus for a longer time, employing the same MLM objective as BERT in which random tokens are masked for the Transformer to guess their original value. ELECTRA [9] is an alternative pretraining technique in which a simple generative model randomly replaces tokens and the Transformer must classify which tokens were replaced.

Given a dataset with multiple text columns, we feed the tokenized text from all columns jointly into our Transformer (with special [SEP] delimiter tokens between fields and a [CLS] prefix token appended at the start [12]), as detailed in the next paragraph. A single embedding vector for all text fields is obtained from the Transformer's representation at the [CLS] position after feeding the merged input into the network [12]. Similarly, just a single text field can be embedded via the Transformer's vector representation at the [CLS] position, after feeding only this field into the network.

Handling Multiple Text Fields in the Transformer Given multiple text columns, we feed the tokenized text from all columns jointly into our Transformer, as illustrated in Figure S1. We follow the usual method to format text from multiple passages [12]: tokenized inputs from different text fields are merged with special [SEP] delimiter tokens between fields and a [CLS] prefix token is subsequently appended at the start of merged input. To further ensure that the network distinguishes boundaries between adjacent text fields, we alternate 0s and 1s as the segment IDs. Here segment IDs and the [SEP] token were previously used to demarcate boundaries between passages during pre-training [12]. After feeding the merged inputs into the Transformer, we can extract its intermediate representations at each position as token-level embeddings (each token has one embedding, which has been contextualized based on information from the other tokens).

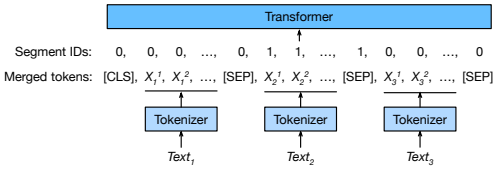
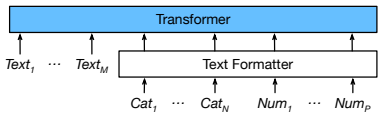
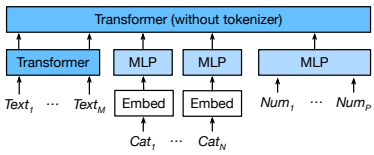


Figure S1: Inputting data from 3 text fields into Transformer.

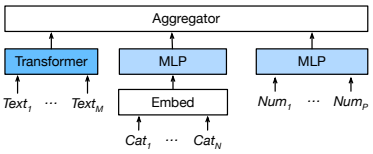
When the total length of tokenized text fields exceed the maximum allowed length (set to be 512 throughout this work), we truncate the input by repeatedly removing one token from the longest



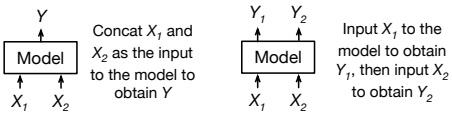
(a) *All-Text*. Convert numeric and categorical values into additional text tokens.



(b) *Fuse-Early*. Transformer operates on learned embeddings for each feature.



(c) *Fuse-Late*. Separate branches encode each modality, and aggregated via concatenation.



(d) Notation used in these figures.

Figure S2: Options for fusing modalities in *Multimodal-Net* (Section A.2). Two dense layers (not shown) are added on top of each network in (a)-(c) to output a prediction (real value for regression, logit vector for classification). Over our benchmark, option (c) performs best and is the chosen *Multimodal-Net* architecture that we subsequently try combining with tabular models.

individual text field until the length constraint is met. Since self-attention is permutation equivariant, a common practice is to assign an additional vector that encodes each position (namely positional encoding) so that the Transformer can distinguish between identical tokens occurring at different locations [52]. After merging multiple text fields into a single input, we simply assign positional encodings based on this larger input.

A.2 Extending Transformer Architectures to Multimodal Inputs

In many multimodal datasets, some of the predictive signal solely resides in text fields, while other predictive information is restricted to tabular feature values, or complex interactions between text and tabular values. To enjoy the benefits of end-to-end learning without sacrificing accuracy, we consider how to adapt a Transformer network to simultaneously operate on inputs from both modalities, referring to the resulting network as *Multimodal-Net*. A natural approach in our setting is to enhance the Transformer such that its attention mechanism can contextualize representations of individual text tokens based not only on other parts of the text, but also on the values of relevant tabular features as well. Below we discuss three different options for implementing the *Multimodal-Net* that are depicted in Figure S2 (with details in Appendix B.2). These options differ in whether information is fused across text and tabular modalities: at the input layer (*All-Text*), in the earlier layers of the network near the input (*Fuse-Early*), or in the later layers of the network near the output (*Fuse-Late*).

All-Text A simple (yet crude) option is to convert numeric and categorical values to strings and subsequently treat their columns also as text fields [43]. Through its byte-pair encoding, a pretrained Transformer can handle most categorical strings and may be able to crudely represent numeric values within a certain range (here we round all numbers to 3 significant digits in their string representation).

Fuse-Early Rather than casting them as strings, we can allow our model to adaptively learn token representations for each numeric and categorical feature via backpropagation (see Figure S2b). We introduce an extra factorized embedding layer [21, 86] to map categorical values into the same $\mathbb{R}^d$ vector representation encoded by the pretrained Transformer backbone for text tokens (with different embedding layers used for different categorical columns in the table). All numeric features are encoded via a single-hidden-layer Multi-layer Perceptron (MLP) to obtain a unified $\mathbb{R}^d$ vector representation. The resulting d-dimensional vector representations from each modality are jointly fed into a 6-layer Transformer encoder whose self-attention operations can model interactions between the embeddings of text tokens, categorical values, and numeric values. We refer to this strategy as *Fuse-Early* because only a minimal (yet adaptive) input processing layer is added to convert the tabular features into a common vector form which can be jointly fed through many shared Transformer layers. Huang et al. [28] considered a similar strategy for applying Transformers to entirely numeric/categorical data, albeit without text components that are a major focus here.

Fuse-Late Rather than aggregating information across modalities in early network layers, we can perform separate neural operations on each data type and only aggregate per-modality representations into a single representation near the output layer (see Figure S2c). This multi-branch design allows each branch to extract higher-level representations of the values from each modality, before the network needs to consider how modalities should be fused. Here we use a multi-tower architecture in which numeric and categorical features are fed into separate MLPs for each modality. The text features are fed into a (pretrained) Transformer network. The topmost vector representations of all three networks are pooled into a single vector from which predictions are output via two dense layers. As pooling operators, we considered mean/max pooling or concatenation as options. Experiments show these pooling methods perform similarly on each dataset, with concatenation exhibiting slightly better overall performance, and we thus fix concatenation as our pooling method in the Fuse-Late architecture.

A.3 Featurizing Text for Tabular Models

Despite their success for modeling text, the application of Transformer architectures to tabular data remains limited [16, 28, 68]. The use of tabular models together with Transformer-like text architectures has also received little attention [33, 53]. Recall that 'tabular models' throughout are those trained on only numeric/categorical features, e.g. different types of decision tree ensembles fit by AutoGluon-Tabular.

To allow tabular models to access information in text fields, the text is typically first mapped to a continuous vector representation which replaces a text column in our data table with multiple numeric columns (one for each vector dimension). One can treat each text column as a document, and each individual text field as a paragraph within the document, such that each text field can be featurized via NLP methods for computing text representations [13, 45, 91] before the tabular models are trained.

Rather than classical NLP methods like N-grams or word embeddings [13], a Transformer can instead be used to map the text fields into a vector representation via contextual embedding [5, 12]. Subsequently, the text fields are replaced in the data table by additional numeric columns corresponding to each dimension of the embedding vector (Figure S3a). Our study considers three ways to featurize text using a Transformer.

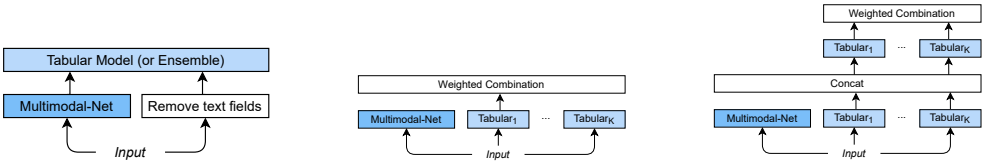
Pre-Embedding Most straightforward is to embed text via a pretrained Transformer (not fine-tuned on our labeled data), and subsequently train tabular models over the featurized data table [5].

Text-Embedding The *Pre-Embedding* strategy is not informed about our particular prediction problem and the domain of the text data. In *Text-Embedding*, we further fine-tune the pretrained Transformer to predict our labels from only the text fields, and use the resulting Text-Net to embed the text. By adapting to the domain of the specific prediction task, *Text-Embedding* is able to extract more relevant textual features that can improve the performance of tabular models. This is particularly true in settings where the target only depends on one out of many text fields, since the fine-tuning process can produce representations that vary more based on the relevant field vs. irrelevant text.

Multimodal-Embedding Text representations may improve when self-attention is informed by context regarding numeric/categorical features. Thus we also consider embedding text via our best multimodal network from Section A.2 (depicted in Figure S2c). These models are again fine-tuned using the labeled data and now produce a single vector representation for *all* columns in the dataset, regardless of their type. Since Transformers are better suited for modeling text than tabular features, we only replace the text fields with the learned vector, all other non-text features are kept and used for subsequent tabular learning. Thus the sole difference between *Text-Embedding* and *Multimodal-Embedding* is that the embeddings used to replace text are additionally contextualized on numeric/categorical feature values in the latter method.

A.4 Aggregating Text & Tabular Models

Rather than merely leveraging the Transformers for their embedding vector representations as in Section A.3, an alternative multimodal text/tabular modeling strategy is to instead consider their predictions and ensemble these with predictions from tabular models. Utilized by most AutoML frameworks [14, 36, 72], model ensembling is a straightforward technique to boost predictive accuracy. Ensembling is particularly suited for multimodal data, where different models may be trained with



(a) *Embedding-as-Feature* (b) *Weighted-Ensemble* (c) *Stack-Ensemble*

Figure S3: Options for combining *Multimodal-Net* with classical tabular models. Five particular tabular models are used in this paper: extremely randomized trees, a simple MLP, and three different types of gradient boosted decision trees. Over our benchmark, option (c) performs the best and is chosen as the strategy for aggregating text and tabular models in our proposed AutoML solution.

different modalities. However, the resulting ensemble may then be unable to exploit nonlinear predictive interactions between features from different modalities. To remedy this, we advocate for the use of our multimodal Transformers (from Section A.2) that fuse information from text and tabular inputs. Here we specifically consider ensembling the multimodal Transformer model with the various standard tabular models used by AutoGluon-Tabular. Furthermore, we propose stack ensembling with nonlinear aggregation of model predictions that can exploit inter-modality interactions between different base models' predictions, even when base models do not overlap in modality.

Weighted-Ensemble We first consider straightforward aggregation via a weighted average of the predictions from our Transformer model and various tabular models (like those trained by AutoGluon-Tabular). Here, our Transformer and other models are independently trained using a common training/validation split. Subsequently, we apply *ensemble selection*, a forward-selection algorithm to fit aggregation weights over all models' predictions on the held-out validation data [7]. Unlike regression for fitting the aggregation weights [36, 50], ensemble selection is favored by many tabular AutoML tools like AutoGluon as it is more computationally efficient, less prone to overfitting, and naturally favors sparse weights [14, 73].

Stack-Ensemble Rather than restricting the aggregation to a linear combination, we can use stacking [104]. This trains another ML model to learn the best aggregation strategy. The features upon which the 'stacker' model operates are the predictions output by all base models (including our Transformer), concatenated with the original tabular features in the data. Following Erickson et al. [14], we try each type of tabular model in AutoGluon-Tabular as a stacker model (see Appendix B.4). To output predictions, a weighted ensemble is constructed via ensemble selection applied to the tabular stacker models (Figure S3c). We do not consider our larger (multimodal) Transformer model as a stacker since lightweight aggregation models are preferred in practice. Overfitting is a key peril in stacking, and we ensure that stacker models are only trained over *held out* predictions produced from base models via 5-fold cross-validation (bagging) [14, 50].

B Additional Experiment/Implementation Details

B.1 Data Processing

For tabular features/models, we can simply rely on the same preprocessing as AutoGluon-Tabular, which has been found to also work well for other learning algorithms [14]. For our subsequently introduced multimodal neural networks that operate on both text and tabular features, we simply rescale and center numeric features and impute their missing values via their average. Missing values for categorical features (and previously unseen categories encountered during inference) are represented via an additional `Unknown` category in order to avoid unrealistic missing at random assumptions. Missing text fields are handled as empty strings in our preprocessing pipeline. The tabular MLP networks in AutoGluon-Tabular (the AutoML solution around which our experiments are based) also only use this simple preprocessing. We thus also utilized the same preprocessing for other neural networks evaluated in our experiments for controlled comparison. Note this was only done for the experiments presented here; the actual datasets in our benchmark have not been preprocessed in this manner, and the benchmark leaves preprocessing as a challenge for future AutoML systems to address as they see fit.

Method	binary	multiclass	regression
RoBERTa	0.843	0.511	0.501
ELECTRA	0.823	0.545	0.519
+ Exponential Decay $\tau = 0.8$	0.885	0.545	0.544
+ Average 3 ★	0.887	0.548	0.546
Choosing Multimodal-Net:		Fusion Strategy	
All-Text	0.893	0.641	0.548
Fuse-Early	0.885	0.643	0.546
Fuse-Late ★	0.889	0.649	0.551
Choosing Aggregation:		Multimodal Model Aggregation	
Pre-Embedding	0.781	0.638	0.430
Text-Embedding	0.798	0.657	0.528
Multimodal-Embedding	0.799	0.673	0.532
Weighted-Ensemble	0.886	0.682	0.549
Stack-Ensemble ★	**0.902**	**0.690**	**0.553**
Baselines:		Tabular AutoML + Feature Engineering	
AG-Weighted	0.697	0.510	0.154
AG-Stack	0.700	0.513	0.155
AG-Weighted+ N-Gram	0.872	0.679	0.471
AG-Stack+ N-Gram	0.877	0.688	0.519
H2O AutoML	0.692	0.551	0.343
H2O AutoML + Word2Vec	0.843	0.627	0.445
H2O AutoML + Pre-Embedding	0.769	0.567	0.427

Table S1: Alternative summary of AutoML results over our multimodal benchmark, where performance on each dataset is separately averaged over the **binary** classification tasks (i.e. average AUC), **multiclass** classification tasks (i.e. average accuracy), and **regression** tasks (i.e. average R^2). See Table 3 for additional details.

AutoGluon also automatically infers the type of each feature via simple yet effective heuristics. One decision particular to our multimodal applications is when to designate a column of string values as a categorical vs. text feature. In this work, we simply threshold based on the number of unique values in the column, such that commonly reoccurring strings are treated as discrete categories rather than unstructured text. We choose the threshold to be 20 in all presented experiments, based on visually confirming the inferred feature types with this threshold agree with our intuition regarding which columns should be handled as text.

While the aforementioned steps are used to report the feature types for each dataset listed in Table 2, we note that our benchmark does *not* require systems to treat certain columns as particular data types. Feature type inference is instead left up to individual methods, since automatically identifying the best way to treat certain columns remains an important research question.

B.2 Network Architectures

In this paper, we used a single-hidden-layer MLP as the basic building block for encoding features and projecting the hidden states. It has one bottleneck layer and uses layer normalization. We use the leaky ReLU activation (with slope set to 0.1) for all basic MLP layers mentioned throughout the paper. For the 6-layer Transformer model in *Fuse-Early*, we used the GeLU activation like Devlin et al. [12]. We set the number of units, heads, and hidden size of FFN (the feedforward layers) in this Transformer to be 64, 4, 256 correspondingly. For the categorical features, we use an encoding network that is similar to the factorized embedding in ALBERT [86], in which we use an embedding layer with 32 units and then project it with a basic MLP layer that has 64 bottleneck units. We further set the number of output units in the basic MLP to be the same as the token-embeddings used in the pretrained Transformer model (i.e., ELECTRA or RoBERTa) so that all vectors belong to the same space.

In the *Fuse-Late* variant, we further concatenate all encoded categorical features and encode them with a second basic MLP layer. Numeric features are concatenated and encoded with one basic MLP layer. These MLP layers all utilize 128 bottleneck units and their output unit number matches the dimensionality of token embeddings for the pretrained Transformer. The total number of parameters for the *All-Text*, *Fuse-Late*, and *Fuse-Early* multimodal network variants are: 109.0 million, 109.1 million, and 109.3 million correspondingly. Thus, these three model variants have comparable costs.

B.3 Neural Network Optimization

All text/multimodal neural networks are trained with the slanted triangular learning rate scheduler [79] with initial learning rate set to 0.0, the maximal learning rate set to 5×10^{-5} and warmup set to 0.1. We use a batch size of 128, 10^{-4} weight decay, and the AdamW optimizer. Text/multimodal networks are trained for 10 epochs and we early stop based on their validation performance. These learning rate and weight decay values were determined via grid search on a single smaller (subsampled) dataset that we used for early initial experiments.

B.4 Details of AutoGluon Tabular Models in the Stack Ensemble

For better efficiency, we considered just the following tabular models when running AutoGluon [14]:

- Fully-connected Neural Network (MLP) with ReLU activations [14].
- LightGBM model with default hyperparameters (GBM) [32].
- A second LightGBM model with a different set of hyperparameter values. By default, AutoGluon uses this second model in conjunction with the first LightGBM model.
- An implementation of Extremely Randomized Trees from the LightGBM library [19].
- CatBoost gradient boosted trees for sophisticated handling of categorical features [40].

To avoid overfitting in stacking, all models are trained with 5 fold cross-validation (bagging) as described by Erickson et al. [14]. For classification tasks, the outputs of each base model which are aggregated in the ensemble are taken to be predicted class probabilities.

B.5 Notes on Hyperparameter Tuning

Note that hyperparameter tuning was not a major focus in the preliminary study conducted in this paper. Standard hyperparameter tuning strategies [95] are readily applicable to our multimodal setting, and the experiments presented here could easily employ the advanced Bayesian optimization techniques available in AutoGluon [97]. We expect the performance of all of our proposed AutoML strategies will grow even better with time devoted to hyperparameter tuning. However in this paper we did not conduct such a search and simply used the default hyperparameters supplied by AutoGluon for tabular models, which are already highly performant [14], and the text/multimodal network hyperparameters are listed here and are viewable in our released code. Over just a few datasets, we found that relative performance of different strategies did not qualitatively differ with other reasonable manually-chosen hyperparameter settings (i.e. hyperparameter values known to generally work well for these specific models such as alternative popular learning rate schedules or small changes to the size of the networks).

Rather than only reporting a couple thoroughly-tuned results, we instead preferred to spend our time/compute budget to explore more modeling strategies over more datasets. Note that all H2O AutoML variants reported in Table 3 relied on extensive hyperparameter sweeps (automatically used within H2O), and yet were still unable to outperform some of the other untuned methods we considered. This further supports the claim that our benchmark has helped us identify a broadly performant strategy for multimodal AutoML.

B.6 Compute Details

All experiments were run on Amazon Web Services EC2 cloud instances (P3.2xlarge). Each instance has two NVIDIA V100 Tensor Core GPUs. About 2000 hours of total compute was required for all experiments presented in this paper (18 instances used for about a week). Given a limited compute budget, we believe more meaningful conclusions may be drawn by running more algorithms over more datasets rather than replicate runs of different seeds/splits on just a few (less diverse) datasets. We also did not include any small datasets in our benchmark for which replicate runs would otherwise be required to get statistically stable results.

C Dataset Details

Each dataset can be easily loaded into Python (or another programming language) as the standard dataframe format used by `pandas`. All tables in our benchmark are appropriately formatted for supervised learning, with the first row serving as a standard header whose columns specify the names of each feature. We release our modified versions of the datasets in our benchmark under a **CC BY-NC-SA** license, and note that any data from this benchmark which has previously been published elsewhere falls under the original license from which the data originated (links to the original sources are provided). We the authors bear all responsibility in case of violation of rights. Long-term preservation of our benchmark is ensured by hosting the repository on GitHub, such that users can contribute their own improvements or publicly raise issues for us to address. The data files are hosted in AWS Simple Cloud Storage (S3), a reliable medium that will ensure researchers can easily obtain these files. Appendix C.1 provides a datasheet [74] for our overall benchmark.

prod: Classify the sentiment (4-way classification) of user reviews of products based on the review text and product type (e.g. Tablet, Mobile, etc.). Intuitively, we expect most of the predictive signal to lie in the text, but predictions can be further improved by accounting for the fact that certain types of products tend to receive certain user sentiment. Representing a relatively simple multimodal task with only a single text feature and one categorical feature, this dataset originally stems from a 2020 MachineHack prediction competition: `https://machinehack.com/hackathons/product_sentiment_classification_weekend_hackathon_19/overview`

salary: Predict the salary range listed in data scientist job postings (in India) given the job description as well as other features like skill requirements and location. Intuitively, the best models will learn to identify valuable requirements from the text and high salary locations (via categorical modeling) as well as predictive interaction-effects. Representing a task with many text fields, this dataset originally stems from a 2018 MachineHack prediction competition: `https://machinehack.com/hackathons/predict_the_data_scientists_salary_in_india_hackathon/overview`

airbnb: Predict the price label of AirBnb listings (in Melbourne, Australia) based on information from the listing page including various text descriptions and many numeric features (e.g. host's response-rate, number of bed/bath-rooms) and categorical features (e.g. property type, superhost or not). Representing a complex classification task with many features from each modality, the original version of this dataset was released via the InsideAirbnb initiative: `https://www.kaggle.com/tylerx/melbourne-airbnb-open-data`

channel: Predict which news category (i.e. channel) a Mashable.com news article belongs to based on the text of its title, as well as auxiliary numerical features like the number of words in the article, its average token length, how many keywords are listed, etc. Representing a task with one text field but many tabular (numeric) features, the original version of this dataset was collected by [70]: `https://archive.ics.uci.edu/ml/datasets/online+news+popularity`

wine: Classify the variety of wines based on tasting descriptions from sommeliers, and numeric features like price and categorical features like country-of-origin. The original version of this dataset was collected from WineEnthusiast: `https://www.kaggle.com/zynicide/wine-reviews`

imdb: Predict whether or not a movie falls within the Drama category based on text features like its name, description, actors/directors, and numerical features like its release year, runtime, etc. Representing a task with smaller sample-size, the original version of this dataset was collected from IMDB (the most popular movies): `https://www.kaggle.com/PromptCloudHQ/imdb-data`*

fake: Predict whether online job postings are real or fake based on their text and additional tabular features like amount of salary offered and degree of education required. Representing an imbalanced binary classification task, these data stem from the Employment Scam Aegean Dataset collected by [101]: `https://www.kaggle.com/shivamb/real-or-fake-fake-jobposting-prediction`

kick: Predict whether a proposed Kickstarter project will achieve funding goal based on text features like its title, description, numeric features like the amount of money requested, date posted, and categorical features like the country, currency, etc. This dataset represents a complex task where models must consider interactions between modalities to address a core question of Kickstarter's

*PromptCloud released the original version of the data from which we created this benchmark dataset.

business: https://www.kaggle.com/codename007/funding-successful-projects

jigsaw: Predict whether online social media comments are toxic based on their text and additional tabular features providing information about the post (e.g. likes, rating, date created, etc.). This dataset originates from a 2019 Kaggle competition (https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification) in which the 1st place solution[10] utilized dataset-specific tricks such as a Bucket Sequencing Collator, auxiliary domain-specific prediction tasks for models, and a custom mimic loss function for training.

qaa: Given a question and an answer (from the Crowdsource team at Google) as well as an additional category feature, predict the (subjective) type of the answer in relation to the question. Representing a predominantly NLP task that requires deep language understanding (though the most accurate models must also consider the category), this dataset stems from a 2019 Kaggle competition: https://www.kaggle.com/c/google-quest-challenge

qaq: Given a question and an answer (from the Crowdsource team at Google) as well as additional category features, predict the (subjective) type of the question in relation to the answer. These data stem from the same source as **qaa**, where the different labels were both prediction targets in the original (multi-label) Kaggle competition.

book: Predict the sale price of books based on text features like their title, author, synopsis, categorical features like genre and numeric features like customer reviews and overall rating. This dataset originally stems from a 2019 MachineHack prediction competition: https://machinehack.com/hackathons/predict_the_price_of_books/overview

jc: Predict the sale price of items sold on the website of the retailer JC Penney based on text features like its title/description, and numeric features like its rating. Representing an important (e)commerce task, this data was originally collected using information from the online page for each product: https://www.kaggle.com/PromptCloudHQ/all-jc-penny-products*

cloth: Predict the score of a customer review of clothing items (sold by an anonymous retailer) based on the review text, how much positive feedback the review has received (numeric), and additional features about the product like its department (categorical). The data were collected by [60]: https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews

ae: Predict the price of inner-wear items sold by retailer American Eagle based on text features like their product name, description, categorical features like brand, and numeric features like rating, review count. Representing an important (e)commerce task, this data was originally collected using information from the online page for each product: https://www.kaggle.com/PromptCloudHQ/innerwear-data-from-victorias-secret-and-others*

pop: Predict the popularity (number of shares on social media, on log-scale) of Mashable.com news articles based on the text of their title, as well as auxiliary numerical features like the number of words in the article, its average token length, and how many keywords are listed, etc. This dataset represents a very difficult prediction problem with only weak signal offered by the observed features. It is fundamentally hard to forecast how popular an article will be based only on its title and crude numerical summary statistics. To be comprehensive, an AutoML benchmark should contain at least one challenging problem like this. While **pop** stems from the same original data source as **channel**, the two have different labels to predict and do not share exactly the same set of features.

house: Predict sale prices of California homes sold in 2020 based on a text summary written by the seller and various tabular features (e.g. bedroom number, home type, location, year built, parking). Representing a regression task with many features that are text and numeric, this dataset originally stems from a 2021 Kaggle prediction competition: https://www.kaggle.com/c/california-house-prices

mercari: Predict the price of items sold in the online marketplace of Mercari based on information from the product page like name, description, free shipping availability, etc. This data originates from a 2017 Kaggle competition (https://www.kaggle.com/c/

[10] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/discussion/103280

mercari-price-suggestion-challenge/), in which 1st place[11] and 3rd place[12] engineered dataset-specific text features such as customized bag-of-words and character N-grams, carefully tuned learning-rate/batch-size schedules, and specially ensembled models in a dataset-specific manner.

C.1 Datasheet for our Multimodal Text/Tabular Benchmark

To avoid redundancy, we only provide details here not covered elsewhere in the paper or our benchmark repository. Table 2 lists statistics of each dataset. For details on how each dataset was collected, please refer to the original source linked in our benchmark repository.

How were datasets selected for the benchmark? The 18 datasets in our benchmark represent all of the public text/tabular datasets we could find that do not violate our exclusion criteria and satisfy our main desiderata: the dataset must entail a meaningful prediction problem with real enterprise data (as opposed to contrived toy task without real-world application). Note that we only consider tabular datasets that contain text fields, which is a small fraction of publicly available tabular datasets (even though such data are ubiquitous in private enterprises). Our dataset search was conducted over the following sources: Kaggle, MachineHack, UCI ML Repository; the first two are the best sources of publicly available enterprise datasets (with meaningful prediction problems) that we are aware of.

Within each source, we searched for datasets matching the keyword "text" in their meta-data/descriptions for consideration in our benchmark (although the majority such datasets either had no tabular features or failed to provide the original raw text presenting only a featurized version such as bag-of-words). We also conducted some dataset searches via Google, but did not find serious candidates for the benchmark via this avenue. Beyond the primary requirement that data must stem from a real enterprise application with a meaningful classification/regression task, our other exclusion criteria ensured each dataset in the benchmark has: IID examples, non-prohibitive licensing, some text fields beyond just 1-2 words and in the English language (for simplicity), sample size of at least 1000, and predictive signal across both text and tabular (numeric+categorical) modalities (meaning one modality does not appear entirely useless for the prediction problem, evaluated via preliminary *AG-Stack+Ngram* runs without each modality).

For what purpose were the benchmark datasets created? We collected the datasets in this benchmark to evaluate supervised machine learning (classification/regression) algorithms designed to jointly operate on text and tabular features. The original versions of these data were also initially created primarily for a similar purpose.

Who created this benchmark? Who funded its creation? The authors of this paper, all scientists employed by Amazon, curated this benchmark. Curating the benchmark did not cost significant money, and the benchmark data are currently hosted on cloud servers (S3) provided by Amazon. The original data sources were created/curated/funded by various companies/individuals, please refer to each individual source for more details.

Do the datasets contain all possible instances or are they a sample (not necessarily random) of instances from a larger set? Each dataset is a sample of instances from a larger set. We caution these samples may not be at all representative of the larger set, and thus the benchmark should not be used to draw domain-specific conclusions/insights through scientific data analysis of individual datasets.

Is any information missing from individual instances? Yes there are many missing fields in certain datasets. It is unclear why they are missing or if the missingness mechanism satisfies the missing at random assumption.

Are relationships between individual instances made explicit? For evaluating ML performance, we simply assume the data are IID. However this may be violated by certain datasets. For example, product datasets may contain near duplicate products and products may be related (reviewed by the same users, price of a product can affect price of others, etc.). We do not explicitly know the

[11] https://www.kaggle.com/c/mercari-price-suggestion-challenge/discussion/50256

[12] https://www.kaggle.com/c/mercari-price-suggestion-challenge/discussion/50272

relationships between instances in these data.

Are there recommended data splits (e.g., training, development/validation, testing)? Yes the benchmark provides a recommended training/test split, but ML systems are free to split validation data from the training set as they see fit. The split was done randomly (stratified based on labels for classification) to best reflect an IID setting for which supervised learning methods are primarily intended.

Does the benchmark contain data that might be considered confidential? Not to our knowledge, but it is possible that a person entered confidential information into the text fields (although they knew these would be publicized).

Does the benchmark contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? The data are mostly non-offensive data used for business purposes. Exceptions are the text fields in the *jigsaw* dataset, which contain toxic online comments, and the *channel/pop* datasets, which contain news article titles that may be anxiety-inducing. Furthermore, some of the user reviews of products may be offensive to certain people, although we did not spot any.

Does the benchmark relate to people? Yes some datasets contain information from people. These all stem from commercial sources where people upload their data intentionally to share it with the world (e.g. user reviews, Kickstarter fundraising, public questions, etc.). There is no sensitive/personal information in these data, beyond what a person intended to publicize.

Is it possible to identify individuals, either directly or indirectly from the benchmark? Yes it may be possible as some datasets contain text fields where an individual may have entered arbitrary information (although they knew the information would appear publicly).

Does the benchmark contain data that might be considered sensitive in any way? Not to our knowledge given all this data was already publicly available, but it is possible given the nature of free form text fields.

How did you process the data from the original sources? Is the software used to preprocess/clean the datasets available? We processed each dataset from the original source using the publicly available scripts in the **scripts/data_processing/** folder of our benchmark GitHub repository. To create versions for our benchmark, we omitted certain features (columns), badly formatted or duplicated rows and subsampled overly large datasets.

Have the benchmark data been used for any tasks already? Yes many of the datasets have been used to evaluate ML systems, some through formal prediction competitions. Other datasets have been used to demonstrate data analysis techniques. For the datasets originally stemming from Kaggle, one can find some of the previously considered tasks in the discussion forum or notebooks associated with the original dataset.

What (other) tasks could the benchmark data be used for? Are there tasks for which these data should not be used? We recommend these datasets only be used for evaluation of machine learning algorithms. One could select different target variables in each dataset to create new prediction tasks to evaluate, but these will likely be less practically meaningful (i.e. representative of a real application) than the target variable we have selected for each dataset. Also note that none of the datasets has extremely large sample-size (say over a million), so modeling conclusions drawn based on this benchmark may not translate to applications with massive datasets.

Will the benchmark be distributed to third parties outside of the entity on behalf of which the dataset was created? Yes the benchmark is made publicly available.

Have any third parties imposed IP-based or other restrictions on the data? Yes please refer to the licenses corresponding to each original data source (linked from our repository) for more details.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? Not to our knowledge.

How can the curators of the benchmark be contacted? You can open a GitHub issue at the benchmark repository, or email the authors of this paper.

Will the benchmark be updated (e.g. to correct errors, add new datasets, add/delete instances)? Yes updates will be done via GitHub and publicly announced there.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? Yes anybody may open Pull Request with desired changes on GitHub.

C.2 Limitations and Societal Impact

Since our benchmark only contains text in the English language and primarily from commercial domains, its conclusions will only hold for particular types of applications. To ensure similar advancements for text/tabular data with low-resource languages [34, 78, 82], we encourage the development of a similar benchmark with non-English text. We also caution that analysis of text fields may raise privacy concerns as such fields may expose arbitrary personal information [62, 71]. Since text fields may contain arbitrary information, they are also prone to introducing spurious correlations in training data that may harm accuracy during deployment [98] and may be undesirably coupled to protected attributes such as race, gender, or socioeconomic status [103]. Basing automated business decisions on customer-generated text could also be more susceptible to adversarial manipulation [88] than tabular features that customers cannot as easily control.

Additional References for the Appendix

[60] A. F. Agarap. Statistical analysis on e-commerce reviews, with sentiment classification using bidirectional recurrent neural network (rnn). *arXiv preprint arXiv:1805.03687*, 2018.

[5] M. Blohm, M. Hanussek, and M. Kintz. Leveraging automated machine learning for text classification: Evaluation of automl tools and comparison with human performance. *arXiv preprint arXiv:2012.03575*, 2020.

[62] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*, 2020.

[7] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *International Conference on Machine Learning*, 2004.

[9] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL HLT*, 2019.

[13] J. Eisenstein. Natural language processing, 2018.

[14] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[68] R. Fakoor, P. Chaudhari, J. Mueller, and A. J. Smola. Trade: Transformers for density estimation. *arXiv preprint arXiv:2004.02441*, 2020.

[16] R. Fakoor, J. Mueller, N. Erickson, P. Chaudhari, and A. J. Smola. Fast, accurate, and simple models for tabular data via augmented distillation. In *Advances in Neural Information Processing Systems*, 2020.

[70] K. Fernandes, P. Vinagre, and P. Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer, 2015.

[71] N. Fernandes, M. Dras, and A. McIver. Generalised differential privacy for text document processing. In *International Conference on Principles of Security and Trust*, pages 123–148. Springer, Cham, 2019.

[72] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, 2015.

[73] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*, pages 113–134. Springer, 2019.

[74] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford. Datasheets for datasets. *arXiv preprint arXiv:1803.09010*, 2018.

[19] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1): 3–42, 2006.

[21] C. Guo and F. Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[24] H2O.ai. NLP with H2O. `https://docs.h2o.ai/h2o-tutorials/latest-stable/h2o-world-2017/nlp/index.html`, 2017.

[78] M. A. Hedderich, L. Lange, H. Adel, J. Strötgen, and D. Klakow. A survey on recent approaches for natural language processing in low-resource scenarios. In *NAACL HLT*, 2021.

[79] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[28] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[31] H. Jin, Q. Song, and X. Hu. Auto-keras: An efficient neural architecture search system. In *KDD*, 2019.

[82] K. Kann, K. Cho, and S. Bowman. Towards realistic practices in low-resource natural language processing: The development set. In *Empirical Methods in Natural Language Processing*, 2019.

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 2017.

[33] G. Ke, Z. Xu, J. Zhang, J. Bian, and T.-Y. Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *KDD*, 2019.

[34] S. M. Lakew, M. Negri, and M. Turchi. Low resource neural machine translation: A benchmark for five african languages. *arXiv preprint arXiv:2003.14402*, 2020.

[86] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[36] E. LeDell and S. Poirier. H2o automl: Scalable automatic machine learning. In *ICML Workshop on Automated Machine Learning*, 2020.

[88] J. Li, S. Ji, T. Du, B. Li, and T. Wang. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium*, 2019.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[38] K. Lopuhin and P. Jankiewicz. 1st place solution to mercari price suggestion challenge. `https://github.com/pjankiewicz/mercari-solution`, 2018.

[91] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[40] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, 2018.

[43] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[44] J. Rodriguez. pytorch-widedeep, deep learning for tabular data i: data preprocessing, model components and basic use. `https://jrzaurin.github.io/infinitoml/2020/12/06/pytorch-widedeep.html`, 2020.

[95] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[45] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

[97] L. C. Tiao, A. Klein, C. Archambeau, and M. Seeger. Model-based asynchronous hyperparameter optimization. *arXiv preprint arXiv:2003.10865*, 2020.

[98] L. Tu, G. Lalwani, S. Gella, and H. He. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633, 2020.

[50] M. J. Van der Laan, E. C. Polley, and A. E. Hubbard. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[101] S. Vidros, C. Kolias, G. Kambourakis, and L. Akoglu. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, 9(1):6, 2017.

[53] A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, H. Jin, S. Petryk, S. A. Bargal, and J. E. Gonzalez. NBDT: Neural-backed decision tree. In *International Conference on Learning Representations*, 2021.

[103] B. A. Williams, C. F. Brooks, and Y. Shmargad. How algorithms discriminate based on data they lack: Challenges, solutions, and policy implications. *Journal of Information Policy*, 8: 78–115, 2018.

[104] D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.