Towards Benchmarking Foundation Models for Tabular Data With Text

Martin Mráz [*1] **Breenda Das** [*1] **Anshul Gupta** [*1] **Lennart Purucker** [1] **Frank Hutter** [1 2 3]

Abstract

Foundation models for tabular data are rapidly evolving, with increasing interest in extending them to support additional modalities such as free-text features. However, existing benchmarks for tabular data rarely include textual columns, and identifying real-world tabular datasets with semantically rich text features is non-trivial. We propose a series of simple yet effective ablation-style strategies for incorporating text into conventional tabular pipelines. Moreover, we benchmark how state-of-the-art tabular foundation models can handle textual data by manually curating a collection of real-world tabular datasets with meaningful textual features. Our study is an important step towards improving benchmarking of foundation models for tabular data with text.

1. Introduction

Foundation models have begun to transform tabular learning (Erickson et al., 2020b; Hollmann et al., 2025), echoing the trajectory of other research fields. A natural next step is mixed-modality tabular modeling, where structured columns may also include free-text fields such as job descriptions, clinical notes, or product summaries. Current tabular benchmarks, however, almost never contain textual columns, cf. (Bischl et al., 2019; Liu et al., 2024; McElfresh et al., 2024). Moreover, locating real-world datasets with semantically rich text features is exceptionally difficult, with even exhaustive searches of OpenML and Kaggle only yielding a handful of usable candidates (Shi et al., 2021). Consequently, current tabular foundation models are rarely evaluated for tabular data with text.

Pipelines that can handle tabular data with text vary greatly in their implementation. AutoGluon's AutoMLPipelineFeatureGenerator (Erickson et al., 2020a) converts text to sparse vectors; CARTE (Kim et al., 2024) applies fastText sentence embeddings (Bojanowski et al., 2017). and the TabPFNv2 API accepts raw text but does not disclose its methodology. These divergent choices raise a fundamental question: Which embedding strategy works best, and under what conditions?

To answer this question, we present the first systematic study of predictive machine learning with foundation models for tabular data with text. We study the performance of three representative embedding routes: fastText, Skrub's TableVectorizer, and AutoGluon's text encoder. We show qualitatively, with a simple synthetic counter-example, that both n-gram based and off-the-shelf sentence embeddings can fail to recover highly predictive semantic patterns. Moreover, quantitatively, we evaluate these methods on a manually curated set of real-world tabular benchmark that (i) contain genuinely informative free-text columns (ii) spans over a variety of domains and samples.

Our contributions are: (I) A qualitative study show casing the limitations of standard n-gram based and generic NLP-based embeddings for tabular tasks with text. **(II)** A manually curated set of real-world tabular datasets with semantically rich textual columns. **(III)** An empirical study of three text embedding pipelines for TabPFNv2 and XGBoost with the TabPFNv2 API and AutoGluon Tabular Predictor as baselines.

Our study reveals the limitations of current methods and underscores the need for new methods to handle tabular data with free text.

2. Related Work

Approaches incorporating free-text in tabular learning largely follow two paradigms. First, **row-as-text** methods serialize entire rows into prompts and delegate prediction to a large language model (LLM), as seen in TabLLM (Hegselmann et al., 2022), Table-LLM (Zhang et al., 2024). These work well when textual fields dominate or in few-shot settings. Second, **per-column embedding** strategies extract textual embeddings from a single or groups of features, while preserving the structural nature of tabular

[*]Equal contribution [1]Department of Computer Science, University of Freiburg, Freiburg, Germany [2]Prior Labs, Freiburg, Germany [3]ELLIS Institute, Tubingen, Germany. Correspondence to: Martin Mraz <mrazm@informatik.uni-freiburg.de>, Breenda Das <dasb@informatik.uni-freiburg.de>, Anshul Gupta <guptaa@informatik.uni-freiburg.de>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

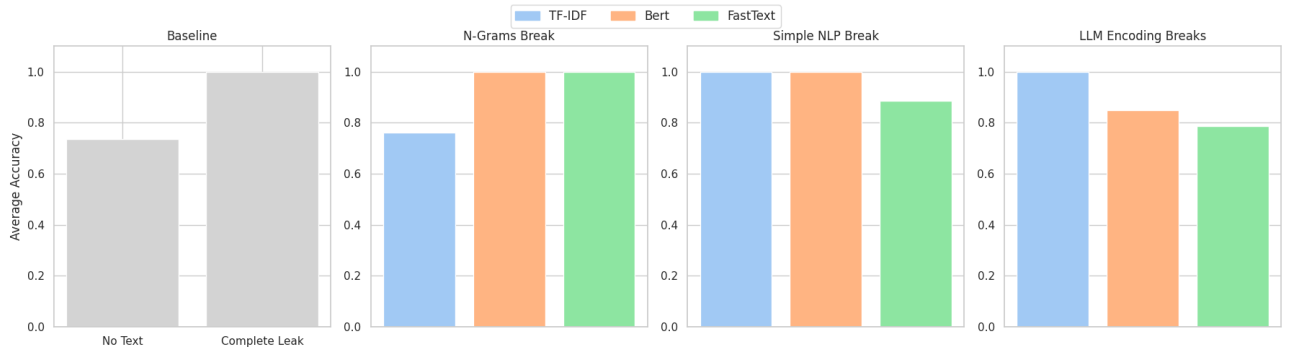


Figure 1. **Qualitative investigation of textual embeddings for tabular prediction tasks:** TabPFN v2 accuracy across five datasets. Baselines "No-Text" uses original input features, "Complete Leak" has target leakage, therefore 100% accuracy for all. Following tests embed targets into textual modality. "N-Gram Break" shows TF-IDF breaking under unseen synonyms, "Simple NLP break" shows FastText underperforming under noise, "LLM Breaks" shows BERT variant breaking under semantic ambiguity.

data. They embed each column using fastText or LLMs and then concatenate the resulting embeddings to the table, or replace the feature with its individual textual embedding, cf. (Koloski et al., 2025; Carballo et al., 2023; Kim et al., 2024). Grinsztajn et al. (2023) compare 30+ LLM and substring-based methods for embedding string columns, showing when LLMs yield better representations.

In this study, we investigate pre-column embeddings because we focus on a many-shot setting (e.g., more than 32 training samples), in which LLM-based predictive methods do not perform well, cf. (Hegselmann et al., 2023) and (Ma et al., 2024), or require a prohibitive amount of fine-tuning (Shysheya et al., 2025).

Most popular prior tabular benchmarks contain no free-text field, cf. (Bischl et al., 2019; Gijsbers et al., 2022; McElfresh et al., 2023). For tabular data with text, Shi et al. (2021) curated 18 datasets with mixed modalities, but many rely almost entirely on text, with only one or two tabular features, or features derived from the long text, e.g. *# of words* (Tang et al., 2024). Thus, they benchmark text-based models rather than tabular models, focusing on text data with tabular features. In contrast, we focus on tabular models extended to handle text, focusing on tabular data with additional text features. Other studies were often limited to an evaluation with just one or two datasets (Carballo et al., 2023; Lu et al., 2023). Overall, existing studies for tabular data with text lack a domain-agnostic benchmark where textual features complement, rather than dominate, tabular data. This motivates the new benchmark we present.

The CARTE Benchmark. The latest seminal work on benchmarking for tabular data with text is the CARTE (Kim et al., 2024) benchmark. It includes 51 datasets for tabular data with text. However, when we investigated these datasets more closely, we found that **at most** 11 **out of the** 51 **datasets are suited to evaluate tabular data with text**. Moreover, our manually curated collection of datasets for benchmarking only includes 1 out of the 51 datasets.
We share an extensive report of our investigation in Appendix G. In short, we found that most datasets **(a)** do not represent predictive machine learning tasks for tabular data with text; **(b)** are skewed towards text data representing many categories instead of longer free text; **(c)** were preprocessed manually per dataset with logic that seems to favor CARTE; **(d)** or were very similar datasets from the same domain, with the same features, or similar semantic content. Thus, while CARTE was a significant step toward benchmarking and learning for tabular data with text, it falls short in several aspects. Our work complements CARTE's efforts and aims to improve benchmarking for tabular data with text further.

3. Qualitative Investigation

In this section, we evaluate popular text embedding techniques: N-Grams, simple NLP models and LLMs for tabular prediction tasks. We hypothesise that N-Grams like TF-IDF break under synonym variations, simple NLP models like FastText sentence vectors (Bojanowski et al., 2017) underperform under noise and LLMs like the BERT variation (Wang et al., 2020) might get confused under semantic ambiguity. Consider the training sentence: "The movie was bad." and the test: "The popcorn was bad, but the movie was great." In TF-IDF, OOD words like "good" weaken inference. FastText averages all word vectors, so noise like "popcorn was bad" disrupts the target signal. LLMs may underperform due to conflicting sentiments across long sentences. Motivated by these limitations, we design experiments to mimic such text patterns and test embedding robustness.

Experimental Setup. We choose five OpenML binary classification datasets (Appendix E) and formulate two baselines

(see Figure 1). For baseline "No Text" the model predicts using original input features. In baseline "Complete Leak" we leak the target feature as an addtional input feature, which results in 100% accuracy. We test robustness of each embedding as follows: (1) We substitute the leaked binary labels with 2 semantically irrelevant words, "good" and "number" and for each sample we substitute with a synonym "great", "three", etc. The test split has different set of synonyms simulating OOD synonyms. (2) We substitute the binary labels with "positive" and "negative", and populate each sample with random words e.g. - "apple mountain positive girl" simulating random noise in long text. (3) We substitute with "positive" and "negative" as before but populate each sample with opposite semantic signals e.g. - "favourable positive sad charming" simulating semantic ambiguity. Appendix E has complete list of subtitutions.

Results and Takeaways. We observe in Figure 1"N-Grams Break" subplot that TF-IDF underperforms given OOD synonyms while FastText and BERT retain 100% accuracy demonstrating robustness to semantic variability. FastText breaks optimal performance under random noise Figure 1 "Simple NLP Break", whereas TF-IDF and BERT retain performance, demonstrating robustness to noise under clear target signal. In Figure 1 "LLM Breaks", semantic ambiguity obstructs BERT and FastText from identifying the target signal, whereas TF-IDF remains robust, relying on target signal frequency.

While we acknowledge that these tests may seem artificial, we also emphasise that semantic variability, random noise and semantic ambiguity are common in real world long texts and given these limitations of prevalent embedding techniques, we might fall short of optimal performance on tabular predictions. Hence we encourage research into resolving these limitations and thereby coming closer to an ideal text embedding technique which would enable optimal tabular predictions using textual fields.

4. Towards Quantitative Benchmarks

Existing collections of datasets for benchmarking tabular data with text do not yet suffice for a robust evaluation. Therefore, we curate a new corpus according to five rules: (i) **Real (free) text features.** We select datasets that are not merely short categorical codes. Instead, we include datasets with real text features that can contain any free text. (ii) **Dual-signal requirement.** Both textual and structural features must carry predictive information; otherwise, a task collapses into either a pure-NLP or pure-tabular test. (iii) **Tabular predictive task.** The dataset must be *primarily* a tabular predictive task. Thus, we exclude datasets that are, for example, recommender systems or text retrieval/look-up tasks. (iv) **Accessibility.** Restricted data, such as real-world patient records (e.g. MIMIC-III (Goldberger et al., 2000)), are omitted so that the benchmark remains easily accessible for all users. (v) **Domain and target diversity.** We do not reuse the same source datasets for multiple benchmark tasks. Instead, we cover commerce, reviews, finance, and sensor-augmented data, spanning regression and (binary/-multi-class) classification tasks.

Table 1. Overview of benchmark datasets with counts of categorical, numerical, and text columns, grouped by task type.

Dataset	Task	# Cat	# Num	# Text	# Rows
fraud	b-clf	4	4	7	17880
kick	b-clf	10	4	4	94125
osha	b-clf	15	1	3	4847
cards	m-clf	4	3	3	2810
complaints	m-clf	4	2	5	96935
spotify	m-clf	5	11	3	10000
airbnb	reg	32	13	11	3818
beer	reg	3	13	5	2914
houses	reg	21	5	4	44913
laptops	reg	29	1	26	984
mercari	reg	2	1	4	100000
permits	reg	12	12	5	90876
wine	reg	7	3	6	1281

Table 1 overviews our curated datasets. Besides bootstrapping on datasets from prior work (Kim et al., 2024; Shi et al., 2021), we manually searched for datasets on UCI, OpenML, and Kaggle portals. After deduplication, only Kaggle provided datasets that satisfied all four criteria. Preprocessing is deliberately minimal, emulating a first-pass data-scientist workflow (e.g., HTML stripping, basic imputation) and supplemented by explicit checks that no target information leaks into the inputs. We provide complete dataset schemas, acquisition scripts, and preprocessing details in Appendix A.
In comparison to prior work, our datasets (i) exhibit a healthier balance between textual and numeric/categorical information, (ii) span diverse domains and scales,and (iii) include transparent preprocessing to ensure reproducibility.

Code availability. The benchmark datasets (and more) acquisition scripts are available at TextTabBench repository.

A note to dataset creators. We encourage the community to release tabular data *before* heavy aggregation or category collapsing, so that the original free-text variation remains available for multimodal research.

5. Quantitative Experiments and Results

In this section we test SOTA tabular models on our dataset pool utilizing different text embedding techniques. We eval-

Table 2. **Comparison between with-text and without-text settings:** The Table compares best accuracy obtained by text embedding techniques used in tabular models on downsampling by SHAP for each dataset against accuracy obtained with non-textual features only. For each model or framework, higher score between with-Text and without-Text is bolded.

Dataset	TabPFNv2			XGBoost			TabPFNv2 API		AG Tabular Predictor	
	Emb	with Text	w/o Text	Emb	with Text	w/o Text	with Text	w/o Text	with Text	w/o Text
airbnb	AGPIPE	$\mathbf{0.692_{\pm 0.048}}$	$0.679_{\pm 0.045}$	AGPIPE	$\mathbf{0.603_{\pm 0.060}}$	$0.592_{\pm 0.048}$	$\mathbf{0.686_{\pm 0.037}}$	$0.673_{\pm 0.033}$	$0.638_{\pm 0.023}$	$\mathbf{0.645_{\pm 0.039}}$
beer	Fasttext	$\mathbf{0.646_{\pm 0.023}}$	$0.579_{\pm 0.020}$	Fasttext	$\mathbf{0.594_{\pm 0.036}}$	$0.468_{\pm 0.020}$	$\mathbf{0.643_{\pm 0.032}}$	$0.573_{\pm 0.034}$	$\mathbf{0.586_{\pm 0.027}}$	$0.512_{\pm 0.039}$
houses	Fasttext	$\mathbf{0.753_{\pm 0.035}}$	$0.733_{\pm 0.027}$	AGPIPE	$\mathbf{0.696_{\pm 0.020}}$	$0.673_{\pm 0.028}$	$\mathbf{0.745_{\pm 0.052}}$	$0.727_{\pm 0.082}$	$0.537_{\pm 0.154}$	$\mathbf{0.626_{\pm 0.121}}$
laptops	AGPIPE	$\mathbf{0.902_{\pm 0.023}}$	$0.881_{\pm 0.021}$	Skrub	$\mathbf{0.833_{\pm 0.046}}$	$0.801_{\pm 0.047}$	$\mathbf{0.900_{\pm 0.014}}$	$0.868_{\pm 0.014}$	$\mathbf{0.841_{\pm 0.020}}$	$0.827_{\pm 0.032}$
mercari	Fasttext	$\mathbf{0.237_{\pm 0.050}}$	$0.001_{\pm 0.016}$	Fasttext	$\mathbf{0.110_{\pm 0.062}}$	$0.001_{\pm 0.006}$	$\mathbf{0.262_{\pm 0.080}}$	$0.012_{\pm 0.027}$	$\mathbf{0.134_{\pm 0.037}}$	$-0.086_{\pm 0.095}$
permits	Fasttext	$0.494_{\pm 0.062}$	$\mathbf{0.506_{\pm 0.057}}$	Fasttext	$0.426_{\pm 0.040}$	$\mathbf{0.467_{\pm 0.065}}$	$\mathbf{0.492_{\pm 0.050}}$	$0.470_{\pm 0.055}$	$\mathbf{0.440_{\pm 0.093}}$	$0.427_{\pm 0.073}$
wine	Fasttext	$\mathbf{0.571_{\pm 0.115}}$	$0.423_{\pm 0.137}$	Fasttext	$\mathbf{0.394_{\pm 0.098}}$	$0.125_{\pm 0.266}$	$\mathbf{0.588_{\pm 0.058}}$	$0.455_{\pm 0.033}$	$\mathbf{0.460_{\pm 0.064}}$	$0.448_{\pm 0.054}$
complaints	FastText	$\mathbf{0.688_{\pm 0.024}}$	$0.646_{\pm 0.017}$	Fasttext	$\mathbf{0.672_{\pm 0.021}}$	$0.584_{\pm 0.023}$	$\mathbf{0.667_{\pm 0.008}}$	$0.639_{\pm 0.015}$	$\mathbf{0.679_{\pm 0.011}}$	$0.630_{\pm 0.024}$
frauds	AGPIPE	$\mathbf{0.962_{\pm 0.008}}$	$0.852_{\pm 0.006}$	AGPIPE	$\mathbf{0.958_{\pm 0.004}}$	$0.849_{\pm 0.015}$	$\mathbf{0.954_{\pm 0.003}}$	$0.845_{\pm 0.012}$	$\mathbf{0.960_{\pm 0.009}}$	$0.848_{\pm 0.014}$
cards	Fasttext	$\mathbf{0.703_{\pm 0.008}}$	$0.662_{\pm 0.007}$	Fasttext	$\mathbf{0.724_{\pm 0.011}}$	$0.632_{\pm 0.009}$	$\mathbf{0.705_{\pm 0.012}}$	$0.659_{\pm 0.020}$	$\mathbf{0.700_{\pm 0.023}}$	$0.655_{\pm 0.027}$
kick	Fasttext	$\mathbf{0.779_{\pm 0.016}}$	$0.702_{\pm 0.010}$	Fasttext	$\mathbf{0.769_{\pm 0.014}}$	$0.657_{\pm 0.013}$	$\mathbf{0.781_{\pm 0.013}}$	$0.686_{\pm 0.021}$	$\mathbf{0.776_{\pm 0.020}}$	$0.699_{\pm 0.020}$
osha	AGPIPE	$\mathbf{0.613_{\pm 0.006}}$	$0.566_{\pm 0.013}$	AGPIPE	$\mathbf{0.599_{\pm 0.020}}$	$0.539_{\pm 0.012}$	$\mathbf{0.547_{\pm 0.029}}$	$0.527_{\pm 0.015}$	$\mathbf{0.559_{\pm 0.019}}$	$0.544_{\pm 0.018}$
spotify	Fasttext	$\mathbf{0.815_{\pm 0.010}}$	$0.663_{\pm 0.016}$	Fasttext	$\mathbf{0.807_{\pm 0.012}}$	$0.636_{\pm 0.027}$	$\mathbf{0.841_{\pm 0.027}}$	$0.665_{\pm 0.016}$	$\mathbf{0.735_{\pm 0.006}}$	$0.636_{\pm 0.013}$

uate performance improvement obtained by each model when including textual features against dropping them.

Experimental Setup. We evaluate two models, TabPFNv2 (Hollmann et al., 2025) and XGBoost (Chen & Guestrin, 2016), with three text embedding methods: (1) FastText (Bojanowski et al., 2017) sentence vectors, (2) Skrub's TableVectorizer that uses GapEncoder by default, and (3) AutoGluon's AutoMLPipelineFeatureGenerator (Erickson et al., 2020b) that uses TextNgramFeatureGenerator. We extend these ablations to BERT embeddings (Wang et al., 2020) in Appendix F. We replace the textual features in the datasets with the corresponding embeddings. Due to memory constraints on our hardware introduced by TabPFN v2 (more details in Appendix D), we evaluate various feature selection and dimensionality reduction techniques for the text embeddings, including statistical tests (t-test, ANOVA), regularization (Lasso), variance-based selection, Principal Component Analysis (PCA), correlation filtering, and SHAP-based (Lundberg & Lee, 2017) importance. Further downsampling details are in Appendix B. We restrict number of features to 300 and rows to 3000 for all evaluations, ensuring consistency. We also test TabPFNv2-API (Hollmann et al., 2025) and AutoGluon's Tabular Predictor (Erickson et al., 2020b) frameworks that support automatic text handling. We report R^2 score for regression and accuracy score for classification, averaged over 5-fold cross-validation of each dataset. We do not conduct hyperparameter tuning of the text embedding generation processes for any datasets and stick to default settings. We expect performance differences upon doing so.

Results. In Table 2 we compare performance of each model and framework when including textual features against not including them. In this table we report best performance amongst all embedding techniques downsampled by SHAP. Appendix F shows all embedding and downsampling results.

(i) **Textual features boost accuracy.** In 11/13 datasets, text embeddings outperforms no-text, for all models and frameworks. (ii) **No single best embedding method.** FastText most frequently gives the best performance, on 7/13 datasets for all models and frameworks. No embedding method dominates universally. (iii) **No single best dowsampling technique.** SHAP-based selection most frequently gives the best performance, but it doesn't always dominate (detailed comparison in Appendix F). (iv) **Local models show higher gains than APIs, but APIs show consistent improvements with text.** TabPFNv2 API consistently outperforms no-text, but local models using our default embedding pipelines often can reach higher accuracy.Hence, we can expect improvements in terms of both magnitude and consistency, and conclude text in tabular models to be an unsolved problem.

Conclusion. In conclusion we present rules for creation an effective text for tabular data benchmark and curate a pool based on these rules. We evaluate different embedding and downsampling techniques, and find that text embeddings boost performance over no-text and downsampling is crucial for effective noise mitigation and reducing memory consumption. We discuss their limitations and thereby motivate creation of a robust text-modality pipeline.

Acknowledgements

L.P. acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under SFB 1597 (SmallData), grant number 499552394; and funding by ELSA – European Lighthouse on Secure and Safe AI funded by the European Union under Grant Agreement No. 101070617. Frank Hutter acknowledges the financial support of the Hector Foundation.

References

Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. Openml benchmarking suites. *arXiv:1708.03731v2 [stat.ML]*, 2019.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information, 2017. URL https://arxiv.org/abs/1607.04606.

Carballo, K. V., Na, L., Ma, Y., Boussioux, L., Zeng, C., Soenksen, L. R., and Bertsimas, D. Tabtext: A flexible and contextual approach to tabular data representation, 2023. URL https://arxiv.org/abs/2206.10381.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In Krishnapuram, B., Shah, M., Smola, A., Aggarwal, C., Shen, D., and Rastogi, R. (eds.), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16)*, pp. 785–794, 2016.

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

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., and Smola, A. Autogluon-tabular: Robust and accurate automl for structured data, 2020b. URL https://arxiv.org/abs/2003.06505.

Gijsbers, P., Bueno, M. L. P., Coors, S., LeDell, E., Poirier, S., Thomas, J., Bischl, B., and Vanschoren, J. Amlb: an automl benchmark, 2022. URL https://arxiv.org/abs/2207.12560.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23): e215–e220, 2000.

Grinsztajn, L., Oyallon, E., Kim, M. J., and Varoquaux, G. Vectorizing string entries for data processing on tables: when are larger language models better?, 2023. URL https://arxiv.org/abs/2312.09634.

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. Tabllm: Few-shot classification of tabular data with large language models, 2022. URL https://arxiv.org/abs/2210.10723.

Hegselmann, S., Parziale, A., Shanmugam, D., Tang, S., Asiedu, M. N., Chang, S., Hartvigsen, T., and Singh, H. Machine learning for health symposium 2023 – findings track, 2023. URL https://arxiv.org/abs/2312.00655.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Kim, M. J., Grinsztajn, L., and Varoquaux, G. Carte: Pre-training and transfer for tabular learning, 2024. URL https://arxiv.org/abs/2402.16785.

Koloski, B., Margeloiu, A., Jiang, X., Škrlj, B., Simidjievski, N., and Jamnik, M. Llm embeddings for deep learning on tabular data, 2025. URL https://arxiv.org/abs/2502.11596.

Liu, S.-Y., Cai, H.-R., Zhou, Q.-L., and Ye, H.-J. Talent: A tabular analytics and learning toolbox. *arXiv preprint arXiv:2407.04057*, 2024.

Lu, J., Qian, Y., Zhao, S., Xi, Y., and Yang, C. Mug: A multimodal classification benchmark on game data with tabular, textual, and visual fields. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5332–5346. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.354. URL http://dx.doi.org/10.18653/v1/2023.findings-emnlp.354.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Ma, J., Thomas, V., Hosseinzadeh, R., Kamkari, H., Labach, A., Cresswell, J. C., Golestan, K., Yu, G., Volkovs, M., and Caterini, A. L. Tabdpt: Scaling tabular foundation models, 2024. URL https://arxiv.org/abs/2410.18164.

Maćkiewicz, A. and Ratajczak, W. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993. ISSN 0098-3004. doi: https://doi.org/10.1016/0098-3004(93)90090-R. URL https://www.sciencedirect.com/science/article/pii/009830049390090R.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. When do neural nets outperform boosted trees on tabular data? In *Proceedings of the 37th International Conference on Advances in Neural Information Processing Systems (NeurIPS'23)*, pp. 76336–76369, 2023.

McElfresh, D., Khandagale, S., Valverde, J., C, V. P., Feuer, B., Hegde, C., Ramakrishnan, G., Goldblum, M., and White, C. When do neural nets outperform boosted trees on tabular data?, 2024. URL https://arxiv.org/abs/2305.02997.

Shi, X., Mueller, J., Erickson, N., Li, M., and Smola, A. J. Benchmarking multimodal automl for tabular data with text fields, 2021. URL https://arxiv.org/abs/2111.02705.

Shysheya, A., Bronskill, J., Requeima, J., Siddiqui, S. A., Gonzalez, J., Duvenaud, D., and Turner, R. E. Jolt: Joint probabilistic predictions on tabular data using llms, 2025. URL https://arxiv.org/abs/2502.11877.

Tang, Z., Fang, H., Zhou, S., Yang, T., Zhong, Z., Hu, T., Kirchhoff, K., and Karypis, G. Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models, 2024. URL https://arxiv.org/abs/2404.16233.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Zhang, X., Luo, S., Zhang, B., Ma, Z., Zhang, J., Li, Y., Li, G., Yao, Z., Xu, K., Zhou, J., Zhang-Li, D., Yu, J., Zhao, S., Li, J., and Tang, J. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios, 2024. URL https://arxiv.org/abs/2403.19318.

A. TextTabBench Datasets

We applied a consistent preprocessing pipeline to all datasets prior to training, aiming to ensure quality, reduce noise, and standardize formats across diverse sources.

General Preprocessing. After downloading the raw CSV files, each dataset was cropped to a maximum of **100,000 rows** to reduce memory and compute overhead. This limit was chosen as it extensively exceeds the typical size needed to train a performant in-context learner in our setting.

We then applied the following preprocessing steps:

- **Missing value filtering:** Columns with more than 50% missing values were dropped (except in a few borderline cases)
- **Constant column removal:** Columns containing only a single unique value (often placeholder IDs or corrupted entries) were removed.
- **Duplicate removal:** Exact duplicate rows were dropped to avoid sampling bias.
- **Target validation:** Rows with missing target labels were discarded.
- **Unnamed column cleanup:** Spurious columns labeled "Unnamed" (typically from CSV formatting issues) were dropped.

Column Type Classification. To determine appropriate encoding strategies, we automatically classified columns into *numerical*, *categorical*, and *textual* types using the following heuristics:

- **Numerical columns:**
  - Columns explicitly typed as numeric were preserved.
  - String columns were also considered numerical if their values were mostly numeric after stripping relatively few (towards total char length) formatting characters (e.g., "15s", $\rightarrow$ 15).
  - Additionally, if the non-numeric part of a string column was repetitive across rows (e.g., ["ABV 12%", "ABV 15%", "ABV 10%"] $\rightarrow$ [12, 15, 10]), the column was interpreted as numeric.
  - After being classified as *numerical*, these columns were automatically cleaned of non-numeric artefacts.
- **Categorical columns:**
  - The threshold for categoricity was typically set to 50 unique values for large datasets, or computed as 5% of the number of rows for smaller ones.
  - In rare cases, this boundary was manually adjusted, for example, when the number of unique values slightly exceeded the default threshold (e.g., 51 categories in a dataset with several thousand rows).
- **Textual columns:**
  - String columns with a large number of unique values and little to no numeric structure were classified as textual.

These heuristics ensured robust and interpretable categorisation across the heterogeneous collection of datasets. The category assignments were manually verified for correctness, and in a small number of cases required reassessment and tailored adjustments. For example, string placeholders like "no-data" were mapped to `NaN`, zip codes embedded in longer strings (e.g., "1234XXX") were trimmed to their numeric prefix ("1234"). In features that were predominantly numerical, if only a few rows contained irregular textual entries (e.g. "3 or 4, not sure"), these were manually cleaned and converted to approximate numeric values (e.g., "3.5") rather than dropping the entire feature, to preserve data utility.

Manual Dataset-Specific Adjustments. Following the general preprocessing and type classification, we applied an additional round of light manual cleaning. These modifications were dataset-specific and aimed at removing irrelevant or non-essential columns (e.g., constant metadata, indices, or redundant identifiers), which could otherwise introduce noise or imbalance.

To ensure fairness across model types, we explicitly avoided transformations that would favor a particular modality. For instance, we *did not* split compound textual columns into multiple fields, as such engineering might disproportionately benefit text-aware models.

The types of changes applied in this step included:

- Dropping hand-identified non-informative columns (IDs, replicated features, etc.).
- Converting clearly timestamp-formatted strings into UNIX numeric time representations.
- Personalised preprocessing of numerical features (e.g. a conversion ['700ml', '8dl', '0.65l', ...] $\rightarrow$ [700, 800, 650, ...])

These dataset-specific adjustments were kept minimal and only made when necessary to ensure compatibility, reduce noise, or preserve valid numerical representations, without imposing any domain expertise.We can summarise these choices as "What would any reasonable person do, given no knowledge about which model is to be used over the data?"

Reproducibility. All such dataset-specific changes are clearly documented in the corresponding preprocessing notebooks within our benchmark repository. The code to reproduce our results is to be found at Towards TextTabBench repository.

Classification Datasets

- **Consumer Complaints Dataset** This dataset contains records of consumer complaints filed against financial service companies, featuring structured fields such as product type, state, and submission method, alongside rich textual attributes like sub-product, issue, sub-issue, and company name. The prediction target is the company's response to the complaint, consolidated into four meaningful classes: CLOSED WITH EXPLANATION, CLOSED WITH NON-MONETARY RELIEF, CLOSED WITH MONETARY RELIEF, AND CLOSED WITHOUT RELIEF. This focused setup maintains a challenging multi-class classification task while allowing for a fair assessment of how textual embeddings can enhance tabular model performance in real-world customer service scenarios. The original dataset has approx. 1.42m rows, which we randomly downsample to 100k to include a fair representation of rare classes while keeping the size feasible for training.
- **Hearthstone Cards Dataset**

  Describes collectable cards from the game *Hearthstone*, with features covering gameplay stats, categorical traits, and rich textual descriptions. The prediction target is the *player class* to which a card belongs (10 categories), with a notable class imbalance. *Neutral* cards dominate, while other classes have significantly fewer samples. This natural imbalance reflects the game design and poses a realistic challenge for classification tasks. Text fields like *text* (card effects) and *flavor* (lore snippets) provide valuable semantic signals, making this dataset well-suited to benchmark how textual features can improve tabular model performance, especially for minority classes.
- **Job Posting Fraud Detection**

  This dataset contains job advertisements with structured attributes (e.g., employment type, required experience) and rich textual content (e.g., job title, description, company profile). The prediction target is *fraudulent* (binary classification: real vs fake). Textual fields such as *description*, *requirements*, and *company_profile* carry essential semantic cues for detecting fraudulent postings, making this dataset highly suitable for evaluating text-augmented tabular models. The target is notably imbalanced, with legitimate postings outnumbering fraudulent ones.
- **Spotify Genres Dataset**

  This dataset contains audio features and metadata for a large collection of songs, combining numerical attributes (e.g., danceability, tempo), categorical musical traits (e.g., key, time signature), and rich textual fields such as *track name*, *album name*, and *artist*. The prediction target is *track genre*, we downsample the original 114 balanced genres to only 10, easier to differentiate classes:

  ['CLASSICAL', 'COUNTRY', 'ELECTRONIC', 'HIP-HOP', 'INDIE', 'JAZZ', 'METAL', 'POP', 'R-N-B', 'ROCK']

- **Kickstarter Success Dataset**

  This dataset contains metadata from Kickstarter crowdfunding projects, including campaign characteristics such as *funding goal*, *duration*, and *category information*, as well as descriptive textual fields like *project name*, *sub-category*, and *city*. The prediction target is the binary *status* (successful vs. failed). Temporal features like *launched_at* and *deadline* are represented as numerical timestamps.

- **OSHA Accidents Dataset**

  This dataset captures detailed records of workplace accidents reported to the Occupational Safety and Health Administration (OSHA), combining structured attributes such as *project type*, *degree of injury*, and *part of body affected* with rich textual descriptions including *abstract summaries*, *event descriptions*, and *associated keywords*. The prediction target is whether the injured employee was performing a *regularly assigned task* at the time of the incident (binary classification).

Regression Datasets

- **Airbnb Rates Dataset**
  This dataset includes Airbnb listings from Seattle with structured features such as *property type*, *room type*, *coordinates*, and *host metrics*, alongside rich textual fields like *listing summaries*, *neighborhood overviews*, and *amenities*. The prediction target is the *listing price* (continuous regression). It is a mix of structured and textual inputs making it an excellent benchmark for assessing how well tabular models can leverage textual embeddings in price prediction tasks.

- **California Housing Prices**
  The dataset includes structured property details such as *lot size*, *year built*, and *school quality*, along with free-text descriptions of *home features* (e.g., *appliances*, *flooring*, *heating*). The prediction target is *Total Interior Livable Area*, shifting focus away from price toward physical home size estimation. Price-related columns are excluded to prevent leakage. This setting tests how well textual features help infer structural characteristics.

- **Laptop Dataset**
  This dataset contains laptop listings with *specifications*, *textual descriptions*, and target prices. Inputs include categorical traits (e.g., *Processor Brand*), numerical specs (e.g., *RAM*, *Screen Size*), and verbose text (e.g., *Sales Package*, *Additional Features*). Rather than filtering, all features are retained to simulate real-world noise and redundancy. It serves as a robust stress test for tabular foundation models under messy conditions.

- **Mercari Price Suggestion Dataset**
  This dataset comes from a consumer-to-consumer marketplace and includes listings with structured fields (e.g., *item condition*, *shipping*) and high-cardinality textual inputs like *product names*, *brand names*, and *descriptions*. The goal is to predict the *item price*. Hierarchical category fields (e.g., *Women/Jewelry/Necklaces*) are retained as textual inputs. The dataset is ideal for evaluating semantic reasoning in text-rich, user-generated data.

- **San Francisco Building Permits Dataset**
  The task is to predict *time_to_approve* (in days) for permit applications. Features include structured fields (e.g., *permit type*, *construction type*, *estimated cost*) and descriptive metadata (e.g., *project description*, *location*). While many permits have zero delay (often legitimate), these are retained. The target's long-tail distribution together with high frequency of zero-day-permits poses a meaningful challenge for modelling bureaucratic latency with mixed inputs.

- **Wine Cost Dataset**
  This dataset includes structured product data (e.g., *grape variety*, *vintage*, *ABV*) and rich textual descriptions such as sensory profiles and marketing text. The prediction target is *price*. Given the subjective nature of descriptions and influence on perceived value, this setting benchmarks how well tabular models can integrate free text for subjective value estimation.

B. Downsampling Strategies

We select a diverse set of feature downsampling techniques to evaluate across datasets and embedding strategies. These include both *supervised* methods that rely on the target variable and *unsupervised* methods that operate independently of it. Some techniques are task-specific—for example, statistical tests require classification targets, and correlation-based

filters are only meaningful for regression. They also vary in complexity: while some are zero-shot filters based on simple statistics (e.g., variance, correlation), others require training a predictive model to estimate feature importance (e.g., SHAP, L1 regularization). This diversity allows us to compare the effectiveness of different selection strategies under varied conditions and modeling setups.

Table 3. **Applicability and properties of feature selection strategies.** A checkmark (✓) indicates support or applicability; a cross (×) indicates that the method is not suitable for that task type or does not have the specified property.

Selector	Regression	Binary Clf	Multi-class Clf	Zero-Shot	Supervised
t-test	×	✓	×	✓	✓
ANOVA	×	✓	✓	✓	✓
L1 (Lasso)	✓	✓	✓	×	✓
Variance	✓	✓	✓	✓	×
PCA	✓	✓	✓	✓	×
Correlation	✓	×	×	✓	✓
SHAP	✓	✓	✓	×	✓
Random	✓	✓	✓	✓	×

T-test This method checks whether the average value of each feature is different between binary classes. It runs a statistical test (t-test) on each feature and selects those with the most significant differences. Only works for binary classification. No model training is needed.

ANOVA This method checks if the average values of a feature are different across three or more classes. It runs a statistical test (ANOVA) on each feature and keeps the ones with the strongest differences. Works for both binary and multi-class classification. No model training is needed.

Variance This method selects features that vary the most across all rows. It keeps the top-k features with the highest variance, assuming that more variable features may carry more information. It is unsupervised and does not use the target variable.

PCA (Maćkiewicz & Ratajczak, 1993) This method uses Principal Component Analysis (PCA) to find combinations of features that explain how the data varies the most. Each feature gets a score based on how much it contributes to these combinations. This score is computed by taking the feature's contribution (called a loading), multiplying it by how much that combination explains, and summing across all combinations. The features with the highest total scores are kept. This method does not use the target variable.

L1 regularisation This method trains a linear model with an ℓ_1 penalty, which forces some feature weights to become exactly zero. For regression tasks, it uses Lasso (linear regression with L1 regularisation). For classification, it uses logistic regression with an L1 penalty. After training, it selects the top-k features with the largest non-zero weights. This method is supervised and requires fitting a model.

Disclaimer: We added this downsampling method only for the additional BERT embeddings experiment.

Correlation This method computes the correlation between each feature and the target, using either Pearson or Spearman correlation. It ranks features by the absolute correlation and selects the top-k. It is supervised and mainly used for regression. It can also be applied to binary classification, but is less reliable than methods like the t-test.

SHAP (Lundberg & Lee, 2017) This method trains a model and then uses SHAP values to measure how much each feature contributes to the model's predictions. It computes the average absolute SHAP value for each feature across the dataset and selects the top-k most important ones. It is supervised and requires training a model such as XGBoost or logistic regression.

C. Hyperparameters and Training Details

Training Settings Summary

TabPFNv2: The TabPFNv2 model is used in inference-only mode with the argument `ignore_pretraining_limits=True`, allowing it to bypass dataset constraints from pretraining.

XGBoost: The XGBoost model uses default hyperparameters unless otherwise specified. For binary classification, it uses the `binary:logistic` objective and `logloss` as the evaluation metric. For multi-class classification, it uses `multi:softprob` with `mlogloss`. Regression uses the default objective (`reg:squarederror`). Label encoding is disabled via `use_label_encoder=False`. Early stopping, max depth, and learning rate are not explicitly set, so the defaults apply (`max_depth=6`, `learning_rate=0.3`).

TabPFNv2 API: This refers to the standardized inference interface provided in the TabPFN (Hollmann et al., 2025), which enables users to leverage the pretrained TabPFNv2 model without the need of external hardware like GPU. For best results, we kept similar samples to other experiments and ensure that all datasets were passed as raw tables (with basic preprocessing as already stated in the paper except for 'osha' where the API does not support "Abstract Text" column and crashes) to the API in our experiments with and without text.

AutoGluon's Tabular Predictor: It is included as a strong AutoML baseline. It is trained per fold with `presets=best quality` and `time_limit=360` seconds across most datasets on `eval_metrics` of R^2 score for regression and `accuracy` for classification tasks. These adjustments allowed AutoGluon's internal model selection, hyperparameter tuning, and ensembling mechanisms to perform more effectively and fairly reflect the model's potential.

Our time constraints were selected to ensure computational feasibility and comparability across datasets. A more generous budget could allow these systems to explore a richer search space and potentially yield higher predictive performance, however we prefer shorter budgets to compare to the ICL performance of TabPFN. Conducting a fair comparison would then necessitate tuning TabPFN as well—an approach that has already demonstrated superior performance compared to AutoGluon's Tabular Predictor under default configurations.

D. Hardware Specifications

We ran all experiments on a single NVIDIA GeForce RTX 2080 Ti GPU with 11 GB RAM for each experiment on TabPFNv2 and single CPU core with 6 GB RAM for each XGBoost experiment, and 32GB RAM for AutoGluon's Tabular Predictor.

Table 4. Classification accuracies across break points and embedding methods.

Break Type	Dataset ID	TF-IDF	BERT	FastText (FT)
TF-IDF Breaks	31	80.0	100.0	100.0
	42193	55.0	100.0	100.0
	1461	80.0	100.0	100.0
	1590	89.5	100.0	100.0
	Average	76.1	100.0	100.0
FastText Breaks	31	100.0	100.0	85.0
	42193	100.0	100.0	95.0
	1461	100.0	100.0	80.0
	1590	100.0	100.0	94.7
	Average	100.0	100.0	88.7
BERT Breaks	31	100.0	85.0	75.0
	42193	100.0	80.0	70.0
	1461	100.0	80.0	80.0
	1590	100.0	94.7	89.5
	Average	100.0	84.9	78.6

E. Motivational Setup

Datasets Used: We used four OpenML datasets for aggregating the scores of the three tested embedding techniques for the motivational setup. We subsample 100 rows randomly in each dataset for these test scenarios. The table 4 details the scores for each dataset and embedding.

Complete list of Dilution Strategies. We present the complete lists of target replacement strategies used in the three test scenarios (excluding the baselines). The full set of synonyms for the terms 'good' and 'number' used in the TF-IDF breakpoint simulation is shown in Listing 1. The full set of random words used for the FastText breakpoint simulation appears in Listing 2. The complete list of synonyms appended to 'positive' or 'negative' in the Bert-based test is provided in Listing 3.

```
self.synonyms = {
    'train': {
        "good": [
            "positive", "great", "excellent", "favorable", "pleasant",
            "admirable", "beneficial", "wonderful", "commendable", "worthy"
        ],
        "number": [
            "one", "three", "four", "five", "six",
            "seven", "eight", "nine", "ten", "eleven"
        ]
    },
    'test': {
        "good": ["nice"],
        "number": ["two"]
    }
}
```

Listing 1. TF-IDF Breakpoint Simulation

```
 self.random_words = [
    "breeze", "crystal", "jungle", "sunset", "clock",
    "river", "pencil", "butterfly", "cloud", "guitar", "forest",
    "echo", "mirror", "flame", "galaxy", "shadow", "storm", "pearl",
    "ember", "whisper", "velvet", "feather", "lantern", "cherry", "fog",
    "nutmeg", "rocket", "canyon", "harbor", "planet", "sketch", "compass",
    "dream", "saddle", "maple", "python", "quartz", "cactus", "ladder",
    "amber", "panther", "blanket", "marble", "candle", "helmet",
    "anchor", "sand", "ocean", "lemon", "boulder", "ink", "ribbon",
    "nest", "basket", "flute", "meadow", "thunder", "vine", "shell",
    "drift", "carpet", "sapphire", "tiger", "honey", "blossom", "stream",
    "mountain", "lighthouse", "cliff", "pebble", "tunnel", "bubble",
    "apple", "silver", "chalk", "frost", "comet", "antler", "bramble",
    "ripple", "beacon", "groove", "hazel", "dune", "harvest",
    "twig", "cobweb", "glider", "ivory", "petal",  "plume",
    "island", "whistle", "puzzle", "snowflake", "cradle",
    "nail", "window", "tassel"
 ]
```

Listing 2. FastText Breakpoint Simulation

```
self.random_words =  [
   # Positive Synonyms
   "favorable", "happy", "joyful", "pleased", "delighted", "cheerful", "content",
       "grateful", "optimistic", "upbeat", "ecstatic",
   "radiant", "thrilled", "hopeful", "enthusiastic", "elated", "blissful",
       "satisfied", "charming", "agreeable", "nice",
   "awesome", "fabulous", "fantastic", "glorious", "marvelous", "splendid",
       "superb", "terrific", "admirable", "commendable",
   "noble", "excellent", "great", "incredible", "lively", "lovely", "magnificent",
       "outstanding", "peaceful",
   "kind", "rejoicing", "serene", "soothing", "supportive", "sympathetic",
       "tender", "vibrant", "warmhearted", "winsome"

   # Negative Synonyms
   "harsh", "sad", "angry", "upset", "depressed", "bitter", "gloomy", "anxious",
       "worried", "hostile", "resentful",
   "unhappy",  "irritable", "moody", "pessimistic", "fearful", "dismal",
       "horrible",  "awful", "nasty", "unpleasant",
   "terrible", "mean", "cruel",  "hurtful", "jealous", "malicious", "miserable",
       "regretful",  "scornful",
   "troubled","spiteful", "tense",  "vindictive", "vulgar", "wicked", "wretched",
       "abrasive", "agonizing",
   "evil", "brutal", "callous", "coldhearted", "disrespectful", "frustrated",
       "hateful", "hostile", "intolerant", "nervous",
   "repulsive"
]
```

Listing 3. Bert Breakpoint Simulation

F. Embedding and Downsampling Ablations

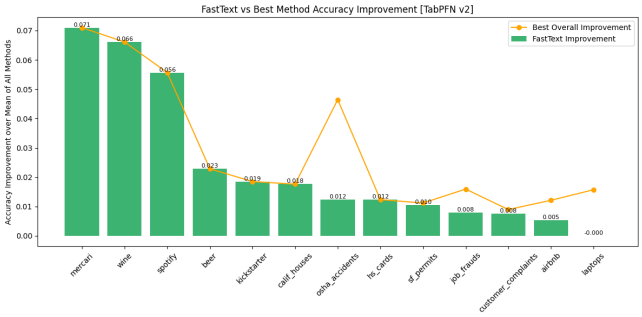


Figure 2. **Performance improvement with FastText over mean accuracy across all ablations on TabPFN v2**: FastText generally gives the best scores, with a few exceptions.These exceptions generally have longer length of texts than average.

We present our complete study of each embedding's performance under every downsampling technique. Tables 6, 7 and 8 compare performance of TabPFN and XGBoost under different downsampling techniques for 3 embeddings FastText, Skrub and Autogluon feature generators respectively. The blanks generally mean the features were below the max-features threshold (300), or at times it's task specific, for e.g: T-Test and Anova are classification specific downsampling techniques so these techniques are skipped for regression tasks.

Comparison of FastText with Mean accuracy across all embeddings. Figure 2 demonstrates improvement of FastText combined with the best performing downsampling technique over mean performance of all ablations for particular dataset. Here, it is clear FastText generally is the best performing embedding technique with a few exceptions.

Comparison of SHAP with Mean accuracy across all downsampling techniques. Figure 4 shows improvement of SHAP downsampling technique over mean accuracy across all downsampling techniques over FastText embeddings. SHAP generally achieves the highest score, with a few exceptions, although the magnitude of difference from the best score on these exceptions are low.

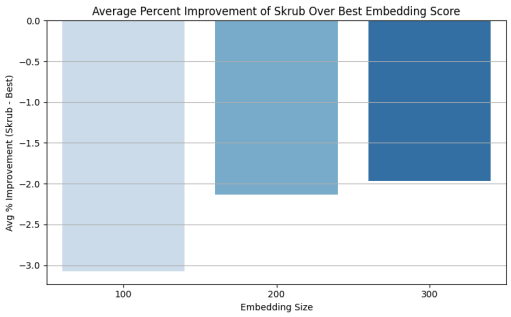
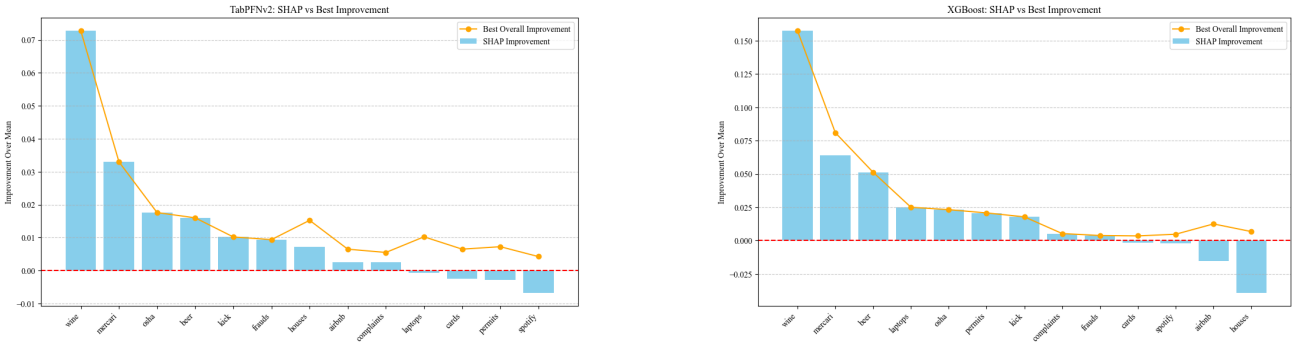


Figure 3. **Ablation over larger embedding sizes for Skrub GapEncoder over five datasets**: beer, calif_houses, laptops, sf_permits, wine. Bars show percentage difference between best score and best skrub embedding score (across all downsampling techniques), averaged across all chosen datasets. Skrub's performance improves with larger embedding dimensions, though the improvement slows down, and any further increase in dimensionality doesn't seem well motivated.



(a) TabPFN v2 with SHAP

(b) XGB with SHAP

Figure 4. **SHAP performance improvement over mean accuracy across all ablations on FastText embeddings:** SHAP generally coincides with the best score using FastText for most datasets, on few exceptions, the performance is slightly below the best score.

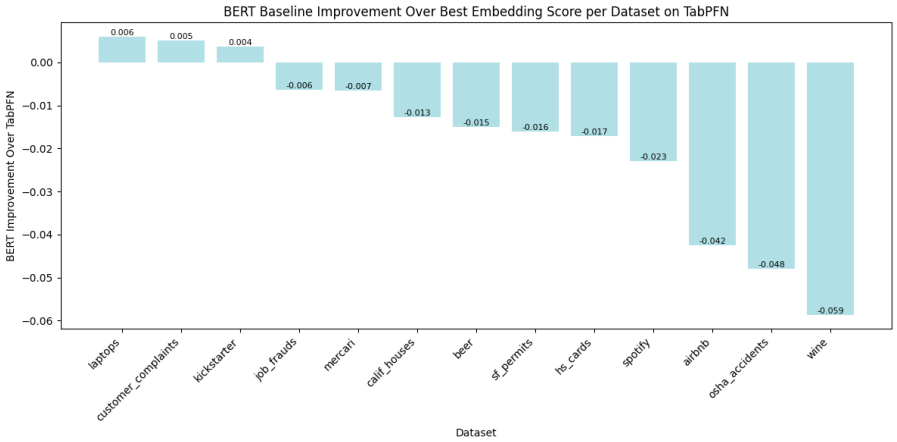


Figure 5. **Performance improvement with BERT variant over best accuracy across all ablations on TabPFN v2**: BERT is sub-optimal compared to best scores over other embeddings: FastText, Skrub and AGPIPE per dataset. This can be possibly due to higher embedding dimensions than the rest, hence downsampling drastically might have caused loss of information.

Extended Ablations on higher dimensional embeddings using Skrub. Since default configuration of Skrub's TableVectoriser uses GapEncoder with only 30 n-components, we include an extended ablation (see Figure 3) for Skrub with higher

embedding sizes to compare with the best score per dataset as in Table 2. We observe that Skrub generally fails to outperform the best score on average across 5 datasets (beer,calif houses, laptops, sf permits and wine) from the pool, although the gap minimizes with higher embedding sizes.

Extended Ablations using BERT. We additionally tested BERT embeddings on our dataset pool across all downsampling techniques. The difference between average BERT scores with best scores per dataset from 2 is in Figure 5. Complete ablation per embedding is shown in Table 5.

Surprisingly, our experiments show that BERT embeddings do not generally outperform our baseline methods. This contrasts with prior work (Grinsztajn et al., 2023), which finds LLM embeddings more effective for rich textual features. Given that our datasets contain high-quality, well-structured text fields, the underperformance may be due to our substantial downsampling of BERT embeddings in our setup, imposed by hardware constraints. This opens up interesting directions for further investigation.

Table 5. Comparison of BERT using various feature selection techniques on TabPFN v2. **Bolded** values indicate the best performing method per dataset. Standard deviations (±) are shown as subscripts.

Dataset	t-test	anova	variance	pca	corr.	shap	rand.	lasso
airbnb	–	–	$0.641_{\pm 0.060}$	$0.640_{\pm 0.063}$	**0.650**	$0.648_{\pm 0.061}$	0.643	$0.639_{\pm 0.060}$
beer	–	–	$0.614_{\pm 0.013}$	$0.624_{\pm 0.011}$	0.597	$\mathbf{0.631_{\pm 0.008}}$	0.621	$0.628_{\pm 0.010}$
houses	–	–	$0.733_{\pm 0.057}$	$0.736_{\pm 0.053}$	0.732	$\mathbf{0.748_{\pm 0.053}}$	0.732	$0.738_{\pm 0.054}$
laptops	–	–	$0.896_{\pm 0.016}$	$0.900_{\pm 0.021}$	0.903	$\mathbf{0.914_{\pm 0.020}}$	0.905	$0.913_{\pm 0.025}$
mercari	–	–	$0.184_{\pm 0.032}$	$0.170_{\pm 0.032}$	**0.230**	$0.223_{\pm 0.032}$	0.185	$0.181_{\pm 0.024}$
permits	–	–	$0.484_{\pm 0.063}$	$0.486_{\pm 0.061}$	**0.489**	$0.476_{\pm 0.059}$	0.488	$0.482_{\pm 0.059}$
wine	–	–	$0.454_{\pm 0.149}$	$0.455_{\pm 0.156}$	0.481	$\mathbf{0.512_{\pm 0.132}}$	0.479	$0.473_{\pm 0.141}$
complaints	–	$\mathbf{0.697_{\pm 0.005}}$	$0.689_{\pm 0.008}$	$0.692_{\pm 0.008}$	–	$0.689_{\pm 0.007}$	0.688	$0.687_{\pm 0.010}$
frauds	$0.945_{\pm 0.007}$	$0.947_{\pm 0.007}$	$0.948_{\pm 0.008}$	$0.949_{\pm 0.009}$	–	$\mathbf{0.955_{\pm 0.008}}$	0.949	$0.954_{\pm 0.006}$
cards	–	$0.683_{\pm 0.011}$	$0.677_{\pm 0.010}$	$0.673_{\pm 0.009}$	–	$\mathbf{0.695_{\pm 0.011}}$	0.683	$0.690_{\pm 0.009}$
kickstarter	$0.772_{\pm 0.010}$	$0.779_{\pm 0.013}$	$0.777_{\pm 0.010}$	$0.776_{\pm 0.010}$	–	$0.780_{\pm 0.012}$	**0.783**	$0.775_{\pm 0.010}$
accidents	$0.550_{\pm 0.036}$	$0.561_{\pm 0.037}$	$0.523_{\pm 0.031}$	$0.529_{\pm 0.031}$	–	$\mathbf{0.565_{\pm 0.035}}$	0.533	$0.528_{\pm 0.030}$
spotify	–	$0.796_{\pm 0.027}$	$0.803_{\pm 0.023}$	$\mathbf{0.803_{\pm 0.023}}$	–	$0.799_{\pm 0.023}$	0.796	$0.801_{\pm 0.021}$

Table 6. Comparison of TabPFNv2 and XGBoost on FastText data embedding across downsampling techniques. **Bolded** values indicate the best performing method per model. Standard deviations (±) are shown as subscripts.

Dataset	TabPFNv2								XGBoost							
	t-test	anova	variance	pca	corr.	shap	rand.	all	t-test	anova	variance	pca	corr.	shap	rand.	all
airbnb	–	–	$0.675_{\pm 0.043}$	$0.675_{\pm 0.044}$	$\mathbf{0.686_{\pm 0.050}}$	$0.682_{\pm 0.054}$	$0.682_{\pm 0.049}$	–	–	–	$0.582_{\pm 0.057}$	$\mathbf{0.595_{\pm 0.036}}$	$0.570_{\pm 0.037}$	$0.567_{\pm 0.053}$	$0.583_{\pm 0.045}$	–
beer	–	–	$0.632_{\pm 0.013}$	$0.640_{\pm 0.015}$	$0.617_{\pm 0.018}$	$\mathbf{0.646_{\pm 0.023}}$	$0.631_{\pm 0.021}$	–	–	–	$0.519_{\pm 0.024}$	$0.520_{\pm 0.024}$	$0.570_{\pm 0.020}$	$\mathbf{0.594_{\pm 0.036}}$	$0.562_{\pm 0.023}$	–
houses	–	–	$0.736_{\pm 0.040}$	$0.741_{\pm 0.045}$	$\mathbf{0.761_{\pm 0.039}}$	$0.753_{\pm 0.035}$	$0.745_{\pm 0.042}$	–	–	–	$0.702_{\pm 0.035}$	$0.683_{\pm 0.039}$	$\mathbf{0.706_{\pm 0.032}}$	$0.660_{\pm 0.074}$	$0.706_{\pm 0.032}$	–
laptops	–	–	$0.880_{\pm 0.016}$	$0.874_{\pm 0.024}$	$0.884_{\pm 0.009}$	$0.882_{\pm 0.014}$	$\mathbf{0.893_{\pm 0.017}}$	–	–	–	$0.761_{\pm 0.033}$	$0.770_{\pm 0.031}$	$0.806_{\pm 0.048}$	$\mathbf{0.811_{\pm 0.037}}$	$0.807_{\pm 0.037}$	–
mercari	–	–	$0.199_{\pm 0.050}$	$0.197_{\pm 0.041}$	$0.221_{\pm 0.036}$	$\mathbf{0.237_{\pm 0.050}}$	$0.199_{\pm 0.042}$	–	–	–	$-0.006_{\pm 0.194}$	$-0.014_{\pm 0.227}$	$\mathbf{0.127_{\pm 0.046}}$	$0.110_{\pm 0.062}$	$0.077_{\pm 0.125}$	–
permits	–	–	$\mathbf{0.504_{\pm 0.058}}$	$0.500_{\pm 0.058}$	$0.489_{\pm 0.066}$	$0.494_{\pm 0.062}$	$0.494_{\pm 0.062}$	–	–	–	$0.406_{\pm 0.041}$	$0.400_{\pm 0.052}$	$0.412_{\pm 0.034}$	$\mathbf{0.426_{\pm 0.040}}$	$0.403_{\pm 0.061}$	–
wine	–	–	$0.486_{\pm 0.110}$	$0.479_{\pm 0.113}$	$0.512_{\pm 0.115}$	$\mathbf{0.571_{\pm 0.115}}$	$0.516_{\pm 0.093}$	–	–	–	$0.149_{\pm 0.237}$	$0.153_{\pm 0.223}$	$0.343_{\pm 0.177}$	$\mathbf{0.394_{\pm 0.098}}$	$0.301_{\pm 0.202}$	–
complaints	–	$\mathbf{0.691_{\pm 0.025}}$	$0.681_{\pm 0.027}$	$0.683_{\pm 0.029}$	–	$0.688_{\pm 0.024}$	$0.687_{\pm 0.027}$	–	–	$0.666_{\pm 0.021}$	$0.667_{\pm 0.026}$	$0.662_{\pm 0.029}$	–	$\mathbf{0.672_{\pm 0.021}}$	$0.672_{\pm 0.029}$	–
frauds	$0.945_{\pm 0.005}$	$0.949_{\pm 0.005}$	$0.941_{\pm 0.014}$	$0.937_{\pm 0.012}$	–	$\mathbf{0.954_{\pm 0.005}}$	$0.951_{\pm 0.007}$	–	$0.947_{\pm 0.001}$	$0.948_{\pm 0.005}$	$0.944_{\pm 0.012}$	$0.939_{\pm 0.010}$	–	$\mathbf{0.949_{\pm 0.005}}$	$0.948_{\pm 0.007}$	–
cards	–	$0.709_{\pm 0.011}$	$0.707_{\pm 0.008}$	$\mathbf{0.712_{\pm 0.009}}$	–	$0.703_{\pm 0.008}$	$0.694_{\pm 0.008}$	–	–	$0.726_{\pm 0.005}$	$\mathbf{0.729_{\pm 0.010}}$	$0.729_{\pm 0.012}$	–	$0.724_{\pm 0.011}$	$0.718_{\pm 0.008}$	–
kick	$0.772_{\pm 0.009}$	$0.772_{\pm 0.013}$	$0.764_{\pm 0.010}$	$0.766_{\pm 0.011}$	–	$\mathbf{0.779_{\pm 0.016}}$	$0.770_{\pm 0.015}$	–	$0.755_{\pm 0.007}$	$0.755_{\pm 0.012}$	$0.749_{\pm 0.016}$	$0.750_{\pm 0.023}$	–	$\mathbf{0.769_{\pm 0.014}}$	$0.747_{\pm 0.011}$	–
osha	$0.575_{\pm 0.008}$	$0.566_{\pm 0.009}$	$0.551_{\pm 0.023}$	$0.563_{\pm 0.019}$	–	$\mathbf{0.579_{\pm 0.015}}$	$0.552_{\pm 0.018}$	–	$0.561_{\pm 0.029}$	$0.551_{\pm 0.016}$	$0.538_{\pm 0.006}$	$0.535_{\pm 0.018}$	–	$\mathbf{0.571_{\pm 0.009}}$	$0.554_{\pm 0.017}$	–
spotify	–	$0.822_{\pm 0.013}$	$0.823_{\pm 0.016}$	$\mathbf{0.826_{\pm 0.014}}$	–	$0.815_{\pm 0.010}$	$0.816_{\pm 0.012}$	–	–	$\mathbf{0.814_{\pm 0.007}}$	$0.812_{\pm 0.009}$	$0.811_{\pm 0.009}$	–	$0.807_{\pm 0.012}$	$0.800_{\pm 0.010}$	–

Table 7. Comparison of TabPFNv2 and XGBoost using Skrub features across downsampling techniques. **Bolded** values indicate the best performing method per model. Standard deviations (±) are shown as subscripts.

Dataset	TabPFNv2								XGBoost							
	t-test	anova	variance	pca	corr.	shap	rand.	all	t-test	anova	variance	pca	corr.	shap	rand.	all
airbnb	–	–	$0.680_{\pm 0.047}$	$0.680_{\pm 0.049}$	$0.679_{\pm 0.049}$	$0.681_{\pm 0.048}$	$\mathbf{0.681_{\pm 0.047}}$	–	–	–	$0.552_{\pm 0.027}$	$0.541_{\pm 0.047}$	$\mathbf{0.568_{\pm 0.043}}$	$0.562_{\pm 0.047}$	$0.545_{\pm 0.059}$	–
beer	–	–	–	–	–	–	–	$\mathbf{0.634_{\pm 0.005}}$	–	–	–	–	–	–	–	$\mathbf{0.560_{\pm 0.021}}$
houses	–	–	–	–	–	–	–	$\mathbf{0.738_{\pm 0.037}}$	–	–	–	–	–	–	–	$\mathbf{0.689_{\pm 0.012}}$
laptops	–	–	$0.895_{\pm 0.025}$	$0.896_{\pm 0.023}$	$\mathbf{0.908_{\pm 0.022}}$	$0.900_{\pm 0.022}$	$0.896_{\pm 0.024}$	–	–	–	$0.809_{\pm 0.044}$	$0.825_{\pm 0.040}$	$0.829_{\pm 0.053}$	$\mathbf{0.833_{\pm 0.046}}$	$0.822_{\pm 0.041}$	–
mercari	–	–	–	–	–	–	–	$\mathbf{0.155_{\pm 0.028}}$	–	–	–	–	–	–	–	$\mathbf{0.099_{\pm 0.046}}$
permits	–	–	–	–	–	–	–	$\mathbf{0.505_{\pm 0.060}}$	–	–	–	–	–	–	–	$\mathbf{0.437_{\pm 0.053}}$
wine	–	–	–	–	–	–	–	$\mathbf{0.467_{\pm 0.145}}$	–	–	–	–	–	–	–	$\mathbf{0.213_{\pm 0.263}}$
complaints	–	–	–	–	–	–	–	$\mathbf{0.692_{\pm 0.027}}$	–	–	–	–	–	–	–	$\mathbf{0.678_{\pm 0.037}}$
frauds	–	–	–	–	–	–	–	$\mathbf{0.953_{\pm 0.007}}$	–	–	–	–	–	–	–	$\mathbf{0.952_{\pm 0.007}}$
cards	–	–	–	–	–	–	–	$\mathbf{0.689_{\pm 0.011}}$	–	–	–	–	–	–	–	$\mathbf{0.723_{\pm 0.013}}$
kick	–	–	–	–	–	–	–	$\mathbf{0.763_{\pm 0.019}}$	–	–	–	–	–	–	–	$\mathbf{0.747_{\pm 0.023}}$
osha	–	–	–	–	–	–	–	$\mathbf{0.560_{\pm 0.015}}$	–	–	–	–	–	–	–	$\mathbf{0.535_{\pm 0.018}}$
spotify	–	–	–	–	–	–	–	$\mathbf{0.810_{\pm 0.007}}$	–	–	–	–	–	–	–	$\mathbf{0.760_{\pm 0.021}}$

Table 8. Comparison of TabPFNv2 and XGBoost using AutoMLPipelineFeatureGenerator (AGPIPE) across various downsampling techniques. **Bolded** values indicate the best performing method per model. Standard deviations (±) are shown as subscripts.

Dataset	TabPFNv2								XGBoost							
	t-test	anova	variance	pca	corr.	shap	rand.	all	t-test	anova	variance	pca	corr.	shap	rand.	all
airbnb	–	–	$0.673_{\pm 0.048}$	$0.675_{\pm 0.044}$	$0.684_{\pm 0.047}$	$\mathbf{0.692_{\pm 0.048}}$	$0.678_{\pm 0.047}$	–	–	–	$0.603_{\pm 0.043}$	$0.601_{\pm 0.034}$	$\mathbf{0.617_{\pm 0.057}}$	$0.603_{\pm 0.060}$	$0.598_{\pm 0.040}$	–
beer	–	–	$0.609_{\pm 0.016}$	$0.612_{\pm 0.015}$	$0.609_{\pm 0.019}$	$\mathbf{0.618_{\pm 0.014}}$	$0.603_{\pm 0.020}$	–	–	–	$\mathbf{0.559_{\pm 0.010}}$	$0.540_{\pm 0.023}$	$0.548_{\pm 0.024}$	$0.553_{\pm 0.030}$	$0.539_{\pm 0.023}$	–
houses	–	–	$0.736_{\pm 0.037}$	$0.738_{\pm 0.030}$	$0.741_{\pm 0.032}$	$\mathbf{0.747_{\pm 0.039}}$	$0.735_{\pm 0.026}$	–	–	–	$0.678_{\pm 0.044}$	$0.687_{\pm 0.041}$	$\mathbf{0.707_{\pm 0.033}}$	$0.696_{\pm 0.020}$	$0.678_{\pm 0.027}$	–
laptops	–	–	$0.897_{\pm 0.027}$	$0.899_{\pm 0.029}$	$0.890_{\pm 0.021}$	$\mathbf{0.902_{\pm 0.023}}$	$0.895_{\pm 0.018}$	–	–	–	$0.804_{\pm 0.041}$	$0.799_{\pm 0.047}$	$\mathbf{0.822_{\pm 0.016}}$	$0.815_{\pm 0.042}$	$0.804_{\pm 0.037}$	–
mercari	–	–	$0.119_{\pm 0.021}$	$0.123_{\pm 0.023}$	$0.151_{\pm 0.022}$	$\mathbf{0.173_{\pm 0.037}}$	$0.051_{\pm 0.021}$	–	–	–	$0.096_{\pm 0.092}$	$0.072_{\pm 0.042}$	$0.096_{\pm 0.192}$	$\mathbf{0.096_{\pm 0.070}}$	$-0.266_{\pm 0.270}$	–
permits	–	–	$0.491_{\pm 0.061}$	$0.482_{\pm 0.061}$	$\mathbf{0.494_{\pm 0.065}}$	$0.492_{\pm 0.058}$	$0.488_{\pm 0.052}$	–	–	–	$0.437_{\pm 0.046}$	$0.437_{\pm 0.025}$	$\mathbf{0.451_{\pm 0.050}}$	$0.420_{\pm 0.054}$	$0.436_{\pm 0.032}$	–
wine	–	–	$0.505_{\pm 0.124}$	$0.503_{\pm 0.114}$	$\mathbf{0.531_{\pm 0.110}}$	$0.527_{\pm 0.135}$	$0.456_{\pm 0.134}$	–	–	–	$0.283_{\pm 0.305}$	$0.281_{\pm 0.288}$	$0.317_{\pm 0.354}$	$\mathbf{0.339_{\pm 0.243}}$	$0.184_{\pm 0.302}$	–
complaints	–	$0.673_{\pm 0.029}$	$0.681_{\pm 0.034}$	$\mathbf{0.682_{\pm 0.027}}$	–	–	$0.677_{\pm 0.030}$	$0.678_{\pm 0.029}$	–	$0.666_{\pm 0.021}$	$\mathbf{0.669_{\pm 0.018}}$	$0.650_{\pm 0.014}$	–	$0.664_{\pm 0.019}$	$0.662_{\pm 0.020}$	–
frauds	$0.950_{\pm 0.006}$	$0.947_{\pm 0.007}$	$0.945_{\pm 0.006}$	$0.946_{\pm 0.005}$	–	$\mathbf{0.962_{\pm 0.008}}$	$0.915_{\pm 0.007}$	–	$0.940_{\pm 0.006}$	$0.946_{\pm 0.002}$	$0.953_{\pm 0.005}$	$0.952_{\pm 0.005}$	–	$\mathbf{0.958_{\pm 0.004}}$	$0.919_{\pm 0.008}$	–
cards	–	–	–	–	–	–	–	$\mathbf{0.683_{\pm 0.010}}$	–	–	–	–	–	–	–	$\mathbf{0.698_{\pm 0.015}}$
kick	$0.756_{\pm 0.021}$	$0.700_{\pm 0.017}$	$0.762_{\pm 0.015}$	$0.763_{\pm 0.018}$	–	$\mathbf{0.767_{\pm 0.015}}$	$0.757_{\pm 0.017}$	–	$0.721_{\pm 0.020}$	$0.679_{\pm 0.014}$	$0.721_{\pm 0.008}$	$0.721_{\pm 0.012}$	–	$\mathbf{0.731_{\pm 0.010}}$	$0.729_{\pm 0.020}$	–
osha	$0.568_{\pm 0.012}$	$0.556_{\pm 0.011}$	$0.560_{\pm 0.018}$	$0.565_{\pm 0.014}$	–	$\mathbf{0.613_{\pm 0.006}}$	$0.557_{\pm 0.013}$	–	$0.557_{\pm 0.011}$	$0.535_{\pm 0.013}$	$0.564_{\pm 0.015}$	$0.559_{\pm 0.025}$	–	$\mathbf{0.599_{\pm 0.020}}$	$0.542_{\pm 0.007}$	–
spotify	–	$0.713_{\pm 0.011}$	$\mathbf{0.718_{\pm 0.006}}$	$0.713_{\pm 0.007}$	–	$0.716_{\pm 0.011}$	$0.704_{\pm 0.015}$	–	–	$0.712_{\pm 0.028}$	$\mathbf{0.713_{\pm 0.015}}$	$0.711_{\pm 0.023}$	–	$0.706_{\pm 0.019}$	$0.703_{\pm 0.023}$	–

G. Previous Benchmarks Accessment

We provide a detailed review of two prior tabular benchmarks involving text features: CARTE (Kim et al., 2024) and Amazon's Multimodal AutoML for Tabular Data (Shi et al., 2021). Our primary focus is CARTE, as it is the more recent and comprehensive of the two.

The CARTE benchmark includes 40 regression and 11 classification datasets. However, upon close examination, we find that many of these datasets are ill-suited for benchmarking systems that aim to handle tabular data with semantically rich free-text features. Below, we highlight common issues and explain why a significant portion of the benchmark is incompatible with our goal of an evaluation design representative of real-world tabular data with text.

G.1. TL;DR

We review 51 datasets from the CARTE benchmark and 18 datasets from the "Multimodal AutoML for Tabular Data with Text Fields" paper. While both were instrumental in shaping our benchmark design, our detailed inspection revealed substantial limitations in their suitability for real-world evaluation of text-tabular systems.

For the 18 datasets, the primary issue was their strong bias toward text-heavy datasets, rendering most tasks better suited for language models than for general-purpose tabular learners. Nonetheless, we retained, with some degree of modifications, seven of their datasets that exhibited balanced multimodal features.

The CARTE benchmark, despite its breadth, suffers from domain redundancy, overuse of artificial targets, and limited semantic depth in its text columns. Many string features serve as identifiers or location codes, offering little benefit to textual encoders. Moreover, several datasets undergo significant undocumented preprocessing, making reproduction difficult.

Based on the discussed shortcomings, we defined a short list of lightweight selection rules to guide our manual dataset curation, as detailed in the main body of the paper. While defining these rules, we closely examined the limitations of prior benchmarks. To support and illustrate these decisions, the following section explores several examples in more detail. Some of the reasoning, such as assessing feature similarity across datasets or the quality of textual signals, requires a degree of human-like meta-understanding. To approximate this process in a transparent and relatively independent way, we incorporate outputs from a highly capable LLM (ChatGPT-4o via API), along with the corresponding prompts.

Ultimately, we used both benchmarks as a springboard, retaining or adapting a few datasets and collecting new ones to construct a more principled and representative benchmark. Our final selection prioritizes semantic textual inputs, domain diversity, and predictive alignment.

G.2. Domain Redundancy and Target Overlap

One of the most immediate issues we observed is the reuse of the same dataset under different targets. For example, several wine review datasets are evaluated both on price and rating targets:

Table 9. List of datasets reused with different target columns.

Dataset	Target 1	Target 2	Retains the Other Target?
Wine.com	Rating	Prices	Yes
Wine Enthusiasts	Points	Price	Yes
Wine Vivino	Price	Rating	Yes

From a modeling perspective, targets like price and rating are often strongly correlated and are unlikely to offer an independent benchmarking signal. Including both only serves to overweight particular domains and can bias results toward models that perform well in those areas.

G.3. Limited Domain Diversity

Although the CARTE benchmark includes over 50 datasets, many of them share the same prediction targets and come from overlapping domains. These two factors are often entangled, leading to much lower effective coverage than the "50+ dataset" label suggests.

When grouped by thematic content or feature-space similarity, the 51 datasets consolidate into approximately 14 to 20 distinct domains. The exact number depends on how strictly one chooses to group datasets. In certain borderline cases, the topics may differ, but the feature schemas and prediction targets are highly similar, which provides a reasonable justification for merging. For example, coffee and alcohol reviews exhibit nearly identical feature schemas and prediction targets, and can reasonably be grouped under a single "consumables/sensory" domain.

To reduce manual bias in domain classification, we leverage large language models for an independent grouping. Specifically, we use OpenAI's GPT-4o to assess dataset similarity based solely on feature names, representative values, and a prompt describing the benchmark context. This LLM-driven process yields 14 coherent domain groupings, aligning with the lower bound of our manual estimate.

Below, we present our manually derived, more permissive domain categorization of datasets from the CARTE benchmark:

Regression Dataset Domains

- **Movies and Shows**: Anime Planet, Japanese Anime, K-Drama, Filmtv Movies, Mydramalist, Rotten Tomatoes
- **Kid Products**: Babies R Us, Buy Buy Baby
- **Alcohol Reviews**: Beer Rating, Wikiliq (Beer & Spirits), Wina Poland, Wine.com, WineEnthusiasts, WineVivino
- **Bike & Car Listings**: Bikewale, Bikedekho, Cardekho, Used Cars 24, Used Cars Benz Italy, UsedCars.com, Used Cars Pakistan, Used Cars Saudi Arabia
- **Salaries**: Employee remuneration, Employee Salaries, ML/DS Salaries
- **Academic Text**: Journal Score JCR, Journal Score SJR
- **Other**: Clear Corpus, Company Employees, Fifa22 Players, Museums, Prescription Drugs, Videogame Sales, US Accidents

Classification Dataset Domains

- **Restaurants**: Zomato, Yelp, Michelin
- **Cross-domain**: Whisky (Alcohol), Roger Ebert (Movies), US Accidents (also used in regression)
- **Miscellaneous**: NBA Draft, Spotify, Ramen rating, Chocolate Bars Ratings, Coffee Ratings

G.4. Qualitative Evaluation via Semantic Matching

To better understand the extent of feature overlap across datasets, we performed a semantic feature comparison using OpenAI's *gpt-4o*. Given the cost of evaluating all pairwise combinations, we present a smaller pilot study using a standardized prompt to extract semantically aligned and misaligned features.

```
prompt_template = """
    Analyze and compare the feature space of two datasets: {dataset1_name} and
        {dataset2_name}.
    Identify relationships between feature names using semantic reasoning.

    **Your Task:**
    1. **Find Similar Features**: Identify columns that represent the same concept, even
        if their names differ.
       - Match based on **data type, structure, and naming conventions**, rather than
           relying solely on example values.
       - Consider cases where **one feature in a dataset maps to multiple features** in
           the other dataset.
       - Note: When a column represents an inherent property of the entity (such as its
           name, title, or composition/build/materials), treat it as similar across
           datasets unless context clearly indicates a different meaning.

    2. **Identify Dissimilar Features**: Columns that do not have a meaningful equivalent
        in the other dataset.
       - Consider **data type mismatches** (e.g., numeric vs. categorical).
```

```
        - Features that belong to completely different contexts should be classified as
            dissimilar.
        - For instance, even if the same term (e.g., "location") is used in both datasets,
            they should only be considered similar if their contexts align.

    ### Additional Guidelines:
    - **Return column names, with original example values.** Do not assume or generate
        example values.
    - Consider **semantic similarity** beyond direct string matching.
    - Account for **differences in feature naming conventions** (e.g., "price" vs.
        "cost", "region" vs. "province").
    - **Preserve structured output strictly in JSON format** - avoid any additional text
        or explanations.
    - Make sure you always return a pair of features for the "similar_features" section.

    ### Expected Output:
    {format_template}

    ### Dataset 1: {dataset1_name}
    {dataset1_values}

    ### Dataset 2: {dataset2_name}
    {dataset2_values}
"""
```

Listing 4. Prompt for feature similarity decision via ChatGPT's 4o model. The <format template> is excluded from the prompt to save space, its structure is visible from the example return in Listing 5.

```
"carte_ramen_ratings vs carte_coffee_ratings": {
    "similar_features": [
        {
            "dataset1_col_name": {"Brand": "MIT"},
            "dataset2_col_name": {"roaster": "A.R.C."},
            "reason": "Both represent the product's manufacturer or origin."
        },
        ...
    ],
    "dissimilar_features": {
        "dataset1": [{"col_name": "Style", ...}],
        "dataset2": [{"col_name": "origin", ...}]
    }
}
```

Listing 5. Sample output: comparing coffee vs ramen datasets

This structured approach highlights redundancy in several domains within the CARTE benchmark and reinforces the need for more careful dataset selection.

Broad domain diversity is important for evaluating how well models generalize across varying tasks and data distributions. When a benchmark heavily reuses a specific domain or target type, models may appear to perform well simply by overfitting to recurring patterns, rather than demonstrating true adaptability. This can lead to inflated performance estimates and misleading conclusions about general-purpose effectiveness.

Visualizing Feature Overlap. Figures 6 and 7 visualize the directional feature coverage between dataset pairs using both continuous and binary heatmaps. Each matrix cell $(A \to B)$ quantifies the proportion of features in dataset A that were semantically matched to features in dataset B, computed as $\#\text{similar features}/(\#\text{similar} + \#\text{dissimilar features in } A)$. This directional score captures asymmetries due to differing feature schema sizes.

The left heatmaps display continuous similarity scores, with masked cells where no semantic match was found. The right heatmaps binarize these scores using a 0.5 threshold to indicate strong vs. weak directional similarity. Blue cells indicate weak or no similarity, green cells indicate strong directional similarity (≥ 0.5), and light grey diagonal entries denote self-comparisons.

The vehicle-related datasets (Figure 6) form a tightly connected group, exhibiting high inter-coverage and structural overlap. In contrast, cross-domain comparisons (Figure 7) such as between healthcare, music, and automotive datasets show low feature alignment.

Together, these visualisations support our core observation: while some dataset groups exhibit redundancy and high overlap, the broader benchmark pool lacks diversity and coherent structure, underscoring the need for more principled dataset selection in text-tabular benchmarks.

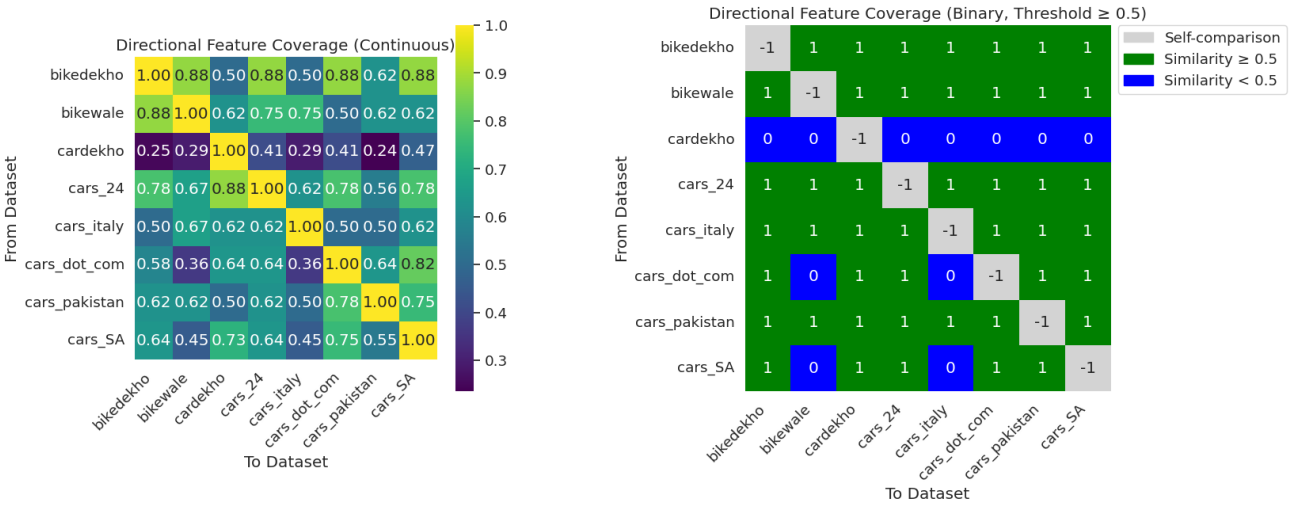


Figure 6. Directional feature coverage between used vehicle listing datasets. Left: proportion of semantically similar features from dataset A to B. Right: binary thresholded version where coverage ≥ 0.5 is highlighted in blue. Grey cells indicate missing comparisons.

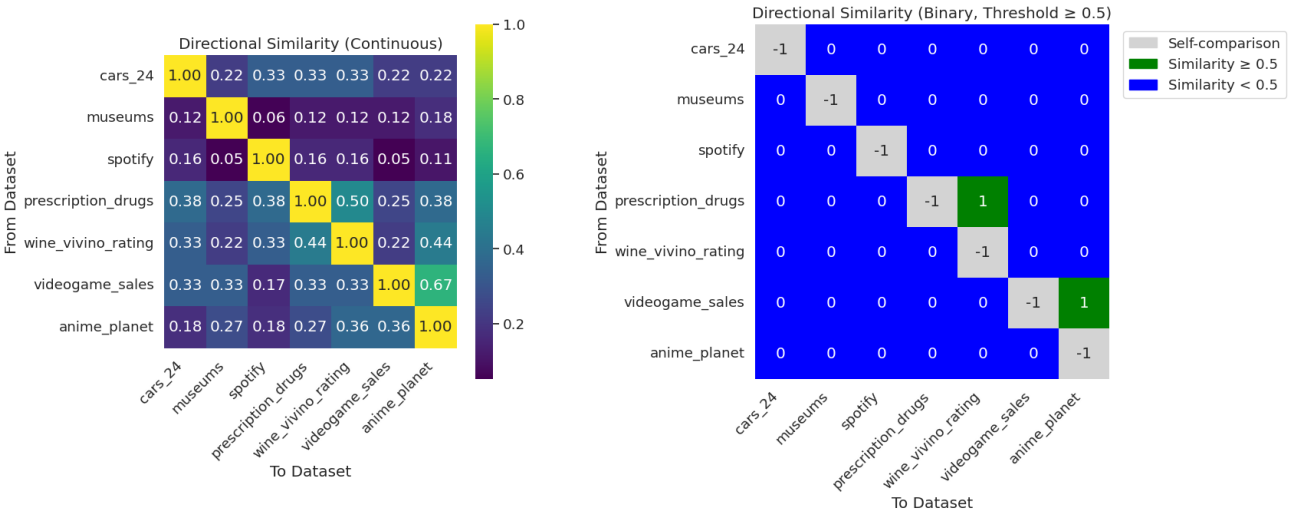


Figure 7. Directional feature coverage between cross-domain datasets with low expected similarity. The left heatmap shows continuous coverage from dataset A to B; the right applies a binary threshold of ≥ 0.5. Grey cells indicate unpaired or unmatched datasets.

Directional Table Comparisons. Tables 11 and 10 provide the raw counts underlying the directional similarity scores in Figures 6 and 7. Each cell reports the number of semantically matched features divided by the total number of features in the source dataset (row).

These counts demonstrate why directionality is essential. A large dataset may match many features in a smaller one, but this overlap may still represent a small fraction of its own schema. For example, `cars_24` matches 2 features in `museums`, yielding 2/17 in one direction and 2/9 in the other—highlighting a twofold difference in perceived similarity.

We observe similar effects even among closely related datasets. In Table 11, comparisons with `cardekho` (17 features) consistently yield lower scores from other datasets, despite sharing similar content. Without directionality, such asymmetries would be obscured, potentially misleading benchmark interpretations.

Table 10. Matched feature counts for dissimilar datasets. Row = source dataset, column = target dataset.

	cars_24	museums	spotify	prescript_drugs	wine_vivino_rating	videogame_sales	anime_planet
cars_24	–	2 / 17	3 / 19	3 / 8	3 / 9	2 / 6	2 / 11
museums	2 / 9	–	1 / 19	2 / 8	2 / 9	2 / 6	3 / 11
spotify	3 / 9	1 / 17	–	3 / 8	3 / 9	1 / 6	2 / 11
prescript_drugs	3 / 9	2 / 17	3 / 19	–	4 / 9	2 / 6	3 / 11
wine_vivino_rating	3 / 9	2 / 17	3 / 19	4 / 8	–	2 / 6	4 / 11
videogame_sales	2 / 9	2 / 17	1 / 19	2 / 8	2 / 9	–	4 / 11
anime_planet	2 / 9	3 / 17	2 / 19	3 / 8	4 / 9	4 / 6	–

G.5. Dataset-Specific Pre-processing

A less visible, but highly consequential, aspect of CARTE is the amount of *dataset-specific* pre-processing embedded in the official dataset loaders shared by the authors. These routines do far more than generic cleaning: while many datasets receive only light scrubbing, a non-trivial subset is significantly modified: columns are renamed, features merged or split, categorical values recoded with expert priors, and even the target variable transformed. While such changes may boost model performance by providing richer semantics, the paper itself mentions only "minimal preprocessing," without elaboration. Because these steps are hard-coded and undocumented, reproducing CARTE results with an independent pipeline is effectively impossible.

Not all datasets are affected. While many datasets in CARTE undergo only minimal hygiene steps (e.g., dropping high-null columns or coercing obvious types), we find that a substantial subset, roughly 10 out of the 51, are subject to heavier, dataset-specific transformations. These go beyond what we would consider acceptable for a general-purpose benchmark and instead introduce domain-specific engineering choices. Our concern is not with basic cleanup, but with the lack of transparency and consistency in how deeper interventions are applied. For details on what we consider minimal and acceptable preprocessing, see Section A.

Illustrative examples. Listings 6 and 7 show two typical cases:

- **`whisky`** 1. maps a five-level price code ($–$$$$$) to coarse text ranges, 2. replaces a one-letter flavour *cluster* with a multi-word description, 3. drops three columns, fills missing values, binarises the `Meta_Critic` target at 8.6, and 4. prunes high-null and single-unique features.
- **`used_cars_pakistan`** 1. concatenates `Brand`, `Model`, and `Version` into one string, 2. log-transforms the price (base 100), 3. coerces several numeric columns to strings or floats, and 4. applies the same null- and uniqueness-based column drops.

Table 11. Matched feature counts for similar datasets. Row = source dataset, column = target dataset.

	bikedekho	cardekho	cars_24	cars_pakistan	bikewale	cars_italy	cars_SA	cars_dot_com
bikedekho	–	4 / 16	7 / 9	5 / 8	7 / 8	4 / 8	7 / 11	7 / 12
cardekho	4 / 8	–	7 / 8	4 / 8	5 / 8	5 / 8	8 / 11	7 / 11
cars_24	7 / 8	7 / 17	–	5 / 8	6 / 8	5 / 8	7 / 11	7 / 11
cars_pakistan	5 / 8	4 / 17	5 / 9	–	5 / 8	4 / 8	6 / 11	7 / 11
bikewale	7 / 8	5 / 17	6 / 9	5 / 8	–	6 / 9	5 / 11	4 / 11
cars_italy	4 / 8	5 / 17	5 / 8	4 / 8	6 / 8	–	5 / 11	4 / 11
cars_SA	7 / 8	8 / 17	7 / 9	6 / 8	5 / 8	5 / 8	–	9 / 11
cars_dot_com	7 / 8	7 / 17	7 / 9	7 / 9	4 / 8	4 / 8	9 / 12	–

```python
# --- recode price brackets (5 -> 6 text ranges) ------------------
data["Cost"] = data["Cost"].map({
    "$$$$$+": "over 300 CAD",
    "$$$$$" : "125-300 CAD",
    "$$$$"  : "70-125 CAD",
    "$$$"   : "50-70 CAD",
    "$$"    : "30-50 CAD",
    "$"     : "< 30 CAD"
})

# --- expand one-letter flavour clusters -------------------------
cluster_map = {"A": "... sweet, fruity, spicy",
               "B": "... floral, malty",
               "J": "dry, very smoky, pungent", ...}
data["Cluster"] = data["Cluster"].map(cluster_map)

# --- binarise target ----------------------------------------------
data["Meta_Critic"] = (data["Meta_Critic"] > 8.6).astype(int)
```

Listing 6. CARTE's key preprocessing steps for `whisky`

```python
# --- standardise column names & dtypes -------------------------
data.rename(columns={"Make":"Brand","Make_Year":"Year","CC":"Engine_Capacity"},
    inplace=True)
data = data.astype({"Year":str, "Engine_Capacity":float, "Mileage":float})

# --- join brand/model/version into one text feature ------------
data["Model"] = data["Brand"] + " " + data["Model"] + ", " + data["Version"]
data.drop(columns=["Brand","Version"], inplace=True)

# --- log-transform target (base-100) ---------------------------
data["Price"] = np.emath.logn(100, data["Price"])
```

Listing 7. CARTE's key preprocessing steps for `used_cars_pakistan`

Why this matters. Such tailored transformations effectively inject domain knowledge and can inflate performance for models that rely on these engineered signals (e.g., composite model names or price buckets). At the same time, other models that might have performed better with the original data are negatively represented. We argue that one would either perform manual preprocessing for all models individually, or for none at all.

G.6. Artificial Targets and Label Construction

Another concern in CARTE is the presence of *artificial or weakly motivated targets*, particularly in their classification tasks. Of the 11 classification datasets, many are derived from regression settings where continuous labels are simply thresholded into binary classes.

This design choice introduces several problems:

- **Loss of task fidelity:** Binarising a continuous outcome (e.g., review scores or numerical metrics) removes useful information and distorts the underlying distribution.
- **Unclear target motivation:** In several cases, the classification target does not appear meaningfully aligned with the available features. For example, in `nba_draft`, the label indicates whether a player had "positive value over replacement". This is a long-term outcome influenced by team context, opportunity, and external factors not represented in the dataset. Without such contextual features, it's unclear how this target can be reliably predicted.
- **Reduced benchmark utility:** These constructions skew the task distribution toward binary classification, limiting the benchmark's diversity and making it a poor proxy for real-world classification challenges.

Not all classification targets in CARTE are problematic, e.g., `spotify` uses "major vs minor key" which, while narrow, is at least categorical by nature. But overall, we find that most of the classification datasets do not represent native classification problems and instead reflect arbitrary thresholding over numerical data.

For a robust benchmark, classification tasks should arise from semantically grounded, categorical outcomes rather than being retrofitted from regression labels.

G.7. Data Accessibility and Reproducibility

While not a primary concern for CARTE, we believe it's important to highlight a broader issue in multimodal tabular benchmarks: *dataset accessibility*. For a benchmark to be truly useful and reproducible, its datasets should be easy to access, ideally via direct download or a minimal sign-up process.

Unfortunately, several works in this domain, such as the AutoML Benchmark (Shi et al., 2021), include datasets sourced from platforms like MachineHack or private competitions. These often require non-trivial setup: creating platform accounts, joining specific challenges, or navigating manual approval processes.

The situation is even more restrictive for medical datasets like `MIMIC-III`, where individual institutional review and data use agreements are needed. While these restrictions are understandable in domain-specific contexts, they pose a significant barrier for community benchmarks that aim to support widespread experimentation and reproducibility.

We emphasize that we have no issues with commonly used sources such as Kaggle, OpenML, UCI, or direct download via `wget`. These platforms offer broad accessibility and minimal setup, making them reasonable choices for benchmark construction. The real challenge arises when datasets require significantly more effort to access—such as navigating opaque approval processes, participating in gated competitions, or relying on unstable or temporary hosting. Such sources are inherently less reliable; datasets may become inaccessible without notice, undermining the longevity and reproducibility of benchmark results.

In our view, benchmark datasets should be:

- publicly accessible without approval steps;
- accompanied by scripts or loaders that work out of the box
- stable over time (i.e., not dependent on ephemeral hosting or event-based portals).

Without these guarantees, reproducing results becomes cumbersome, and sharing new methods depends on access to fragile or gated data sources, undermining the core purpose of open and lasting benchmarks.

G.8. Dual Signal

G.9. Free Text Is Not Always Free Text

When building benchmarks for tabular models that incorporate textual inputs, it is essential to verify that the so-called "text" columns are truly *semantic* in nature, not merely strings in format. Many datasets include string-valued fields that function more like categorical variables or identifiers, such as product codes, model names, or brand labels. While technically textual, these features typically do not require or benefit from text-aware modeling.

To better characterize this, our internal analysis includes basic statistics such as the number of unique values per column. This helps reveal whether string columns behave more like free text or structured categories. For instance, a column with only a few dozen repeated string values is unlikely to encode the kind of nuanced semantics that justify using advanced text encoders.

In the case of CARTE, to be fair, the benchmark does not explicitly position itself as targeting free-text reasoning. However, it has often been interpreted that way within the community. Based on our analysis, only around half of the datasets (depending on the threshold applied) contain string features with enough diversity to be meaningfully treated as textual inputs. This highlights the importance of validating the modality of features, rather than assuming all string-typed columns warrant semantic modeling.

Yet, at least one example of a dataset with non-semantically rich free text features: 'us-accidents-counts' dataset in the

CARTE benchmark has textual columns such as 'ZIPCODE' or 'AIRPORT-CODE' which have no semantic value or concrete relevance to the target feature of accident counts in the city.

G.10. How to Choose a Good Prediction Task Dataset?

To assess the suitability of datasets for prediction tasks, we conducted an analysis on both CARTE's benchmark datasets and those in our own benchmark. This analysis leveraged GPT-4o to evaluate whether each dataset is well-aligned with the intended task objectives. The prompt used for this evaluation is shown in Listing 8.

We illustrate GPT's assessment using two example datasets: CARTE's `us-accidents` and our own `beer-rating`. Based on GPT's evaluations, the majority of CARTE's datasets were categorized as either Red or Yellow, indicating potential issues in task alignment. In contrast, our benchmark datasets were predominantly classified as Green, suggesting a stronger fit for the intended prediction tasks.

These findings support our motivation to develop a new benchmark that addresses some of CARTE's limitations, as identified through GPT's interpretations. It is important to note that these assessments were based on GPT-4o at the time of testing and may evolve with future model updates.

```
prompt_template = """
I am a researcher in the field of AI and I want to create a benchmark for tabular
    datasets with meaningful textual features. The textual features would be replaced
    with textual embeddings and the dataset will be used to benchmark a set of different
    tabular models.  For each dataset I will provide you with column names, first few
    rows and declaration of the target feature. Based on conditions below, review each
    dataset and classify it as Green (meets all conditions), Yellow (meets some), or Red
    (meets none) based on its fitness to be included in the benchmark, then justify your
    choice.
The general conditions are:
(1) suitability for regression/classification (not recommendation or look up table tasks),
(2) prediction is to be boosted by both textual and non textual features
(3) target feature is native to the prediction task and relavant to the feature space
(4) Textual features are semantically rich (e.g., 'item_condition' > 'seller_name').
(5) The features must contain enough signal for the model to go beyond predicting the
    mean/target statistics.
Always explain your reasoning
(e.g., 'Yellow: meets 1, 3, and 5 but lacks long-text features' and hence is not a good
    fit for the benchmark).
"""
```

Listing 8. Prompt for analysing dataset fit for Text for Tabular data benchmark.

```
Answer: """
Classification: RED

Analysis:

Condition 1: Suitability for regression or classification

Met. The target column "Counts" is a numeric value, which makes the dataset suitable for
    a regression task.

Condition 2: Prediction is to be boosted by both textual and non-textual features

Not met. While the dataset includes textual features like "City", "County", and "State",
    these are short categorical labels. They are not semantically rich and are unlikely
    to benefit from textual embeddings. Features like "Zipcode", "Code", and
    "Airport_Code" are identifiers or codes, not meaningful text.

Condition 3: Target is native to the task and relevant to the features

Partially met. The target "Counts" might represent some countable event or measurement,
    but there is no clear link between the features and the target. For example, the same
    city and zipcode appear with very different count values, suggesting missing context.
```

```
Condition 4: Textual features are semantically rich

Not met. The textual features in the dataset are names of locations and codes. These are
    not descriptive, long-form, or semantically meaningful in a way that text embeddings
    would capture. This limits their utility in a benchmark focused on rich textual input.

Condition 5: Dataset has enough signal to go beyond predicting the mean

Partially met. While 'Counts' varies across rows, the available features do not clearly
    explain that variance. This limits a model's ability to learn meaningful patterns
    beyond simple statistics.

Conclusion:
This dataset is not a good candidate for a benchmark focused on textual embeddings in
    tabular models. It lacks rich textual features and does not offer strong predictive
    links between inputs and target values.

Final Rating: RED: Only condition 1 is clearly met.
"""
```

Listing 9. GPT's verdict on 'us-accidents-counts' dataset from CARTE benchmark.

```
Answer: """
Classification: GREEN

Analysis:

Condition 1: Suitability for regression or classification
Met. The target column "review_overall" is a numeric value, which makes this a valid
    regression task.

Condition 2: Prediction is to be boosted by both textual and non-textual features
Met. The dataset includes a rich textual feature ("Description") that contains tasting
    notes and product details. In addition, there are structured numeric features like
    "ABV", "Min IBU", "Max IBU", and flavor attributes such as "Sweet", "Sour", and
    "Hoppy". This combination supports modeling with both types of features.

Condition 3: Target is native to the task and relevant to the feature space
Met. The "review_overall" rating is directly related to the beer's style, taste, and
    description. It is a natural outcome of the given features, which makes it highly
    relevant.

Condition 4: Textual features are semantically rich
Met. The "Description" field includes full sentences and tasting notes. This is
    semantically rich text that can benefit from text embeddings and contributes
    meaningfully to model performance.

Condition 5: Dataset has enough signal to go beyond predicting the mean
Met. There is variability in the review scores and the number of reviews. Combined with
    detailed feature data, the model has enough signal to learn meaningful patterns
    rather than just guessing the average.

Conclusion:
This dataset is well-suited for benchmarking tabular models that integrate textual
    embeddings. It includes a strong combination of structured and unstructured features,
    a meaningful target, and enough variability to support learning.

Final Rating: GREEN - Meets all conditions.
"""
```

Listing 10. GPT's verdict on 'beer-rating' dataset from our benchmark.