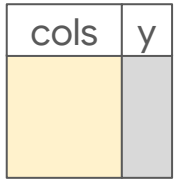
IngesTables

Scalable and Efficient Training of LLM-Enabled Tabular Foundation Models

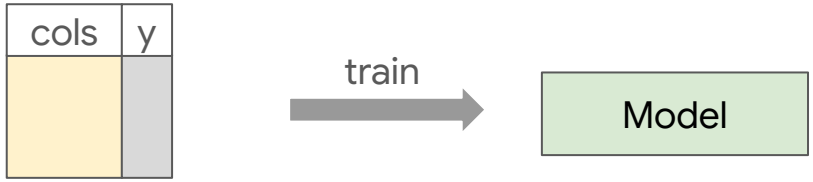
{scottyak,yihed,xavigonzalvo,soarik}@google.com

Google

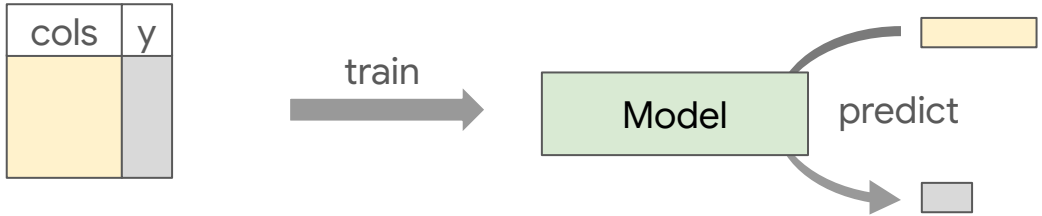
Standard tabular learning setup



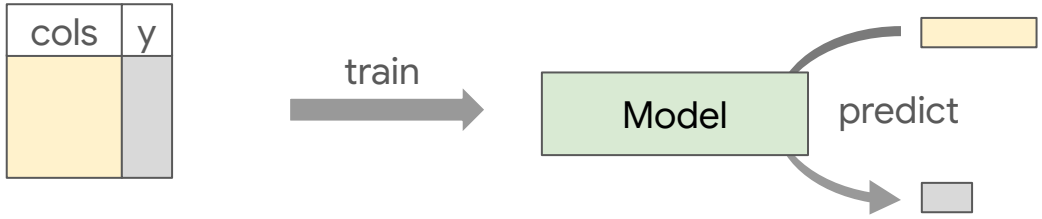
Standard tabular learning setup



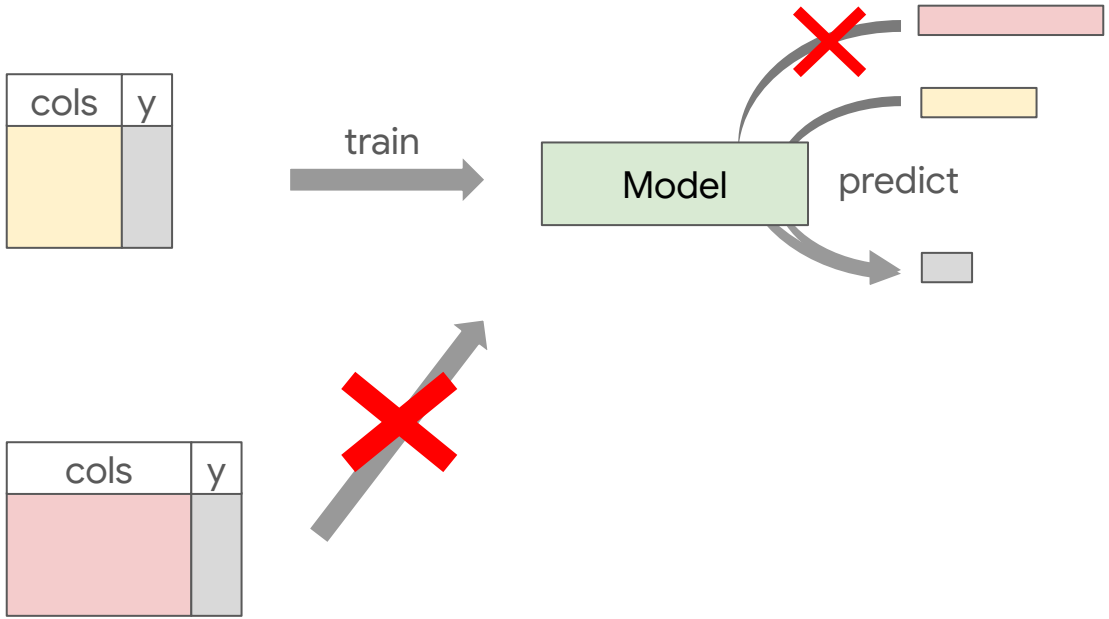
Standard tabular learning setup



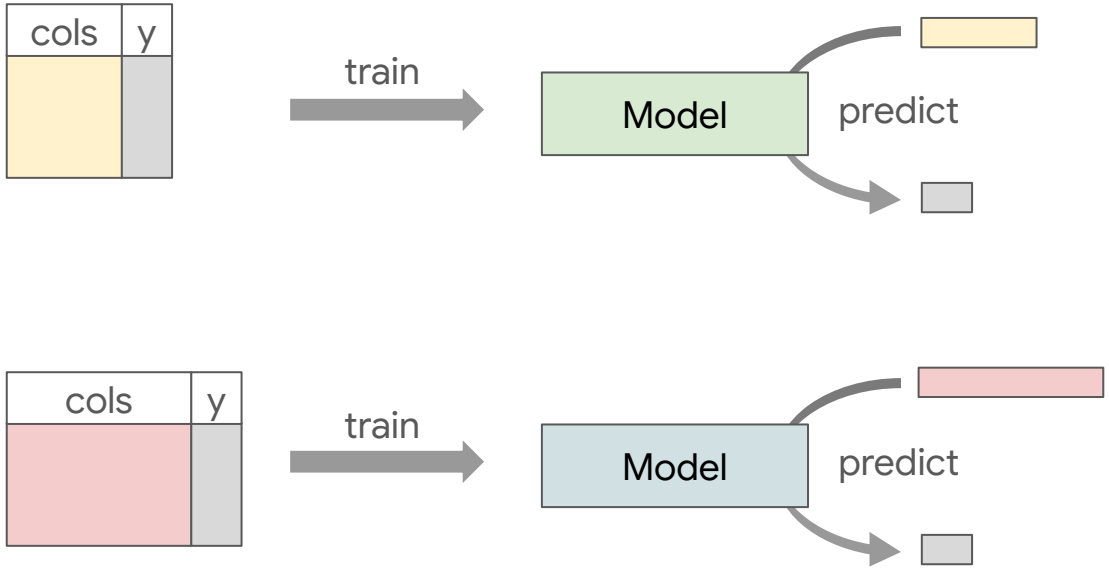
Standard tabular learning setup



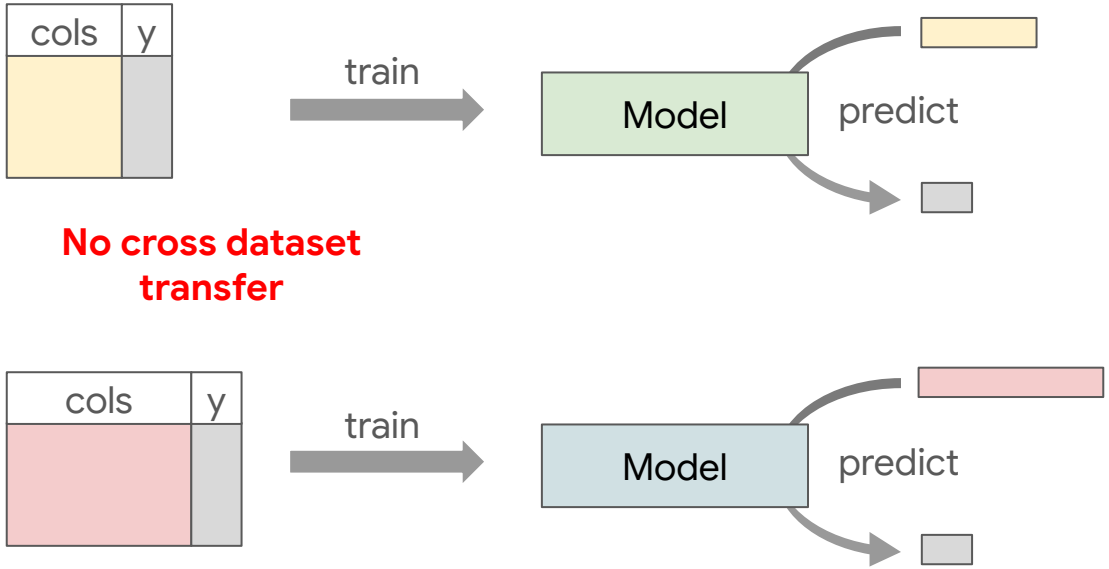
Standard tabular learning setup



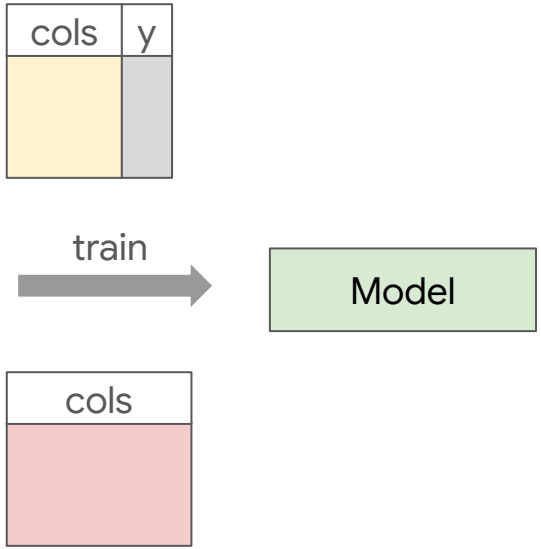
Standard tabular learning setup



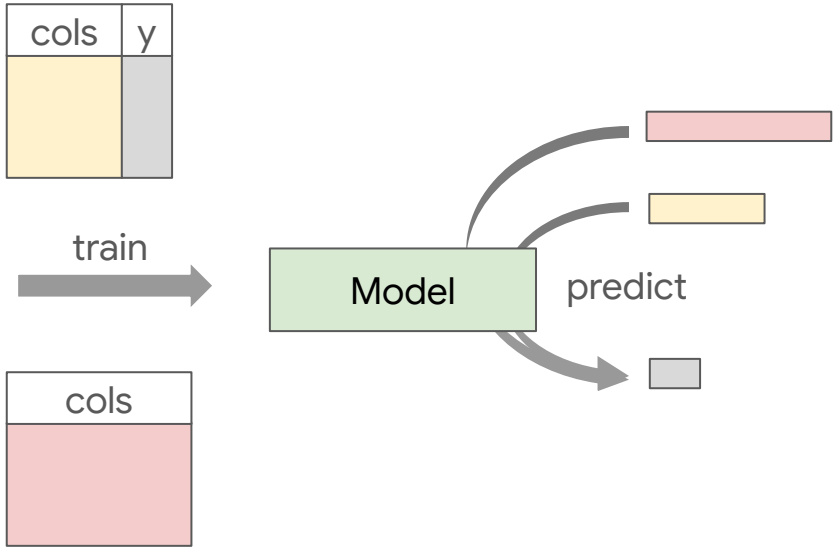
Standard tabular learning setup



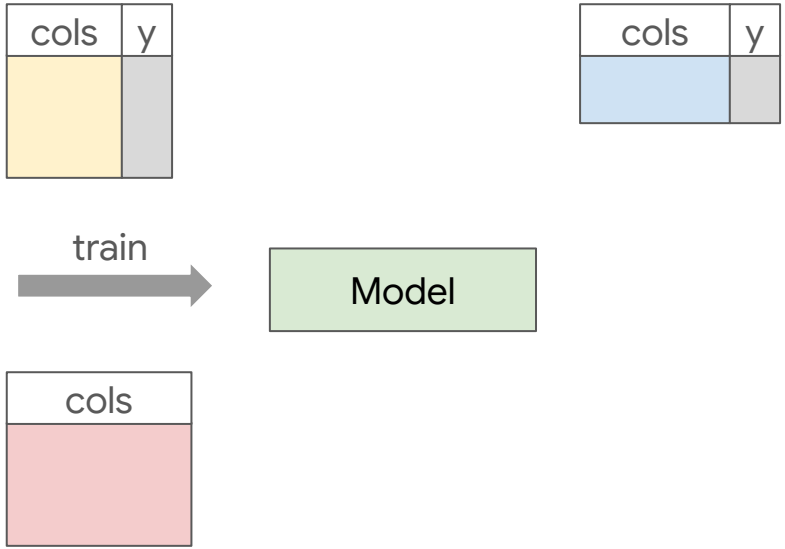
TransTab setup [Wang & Sun, 2022]



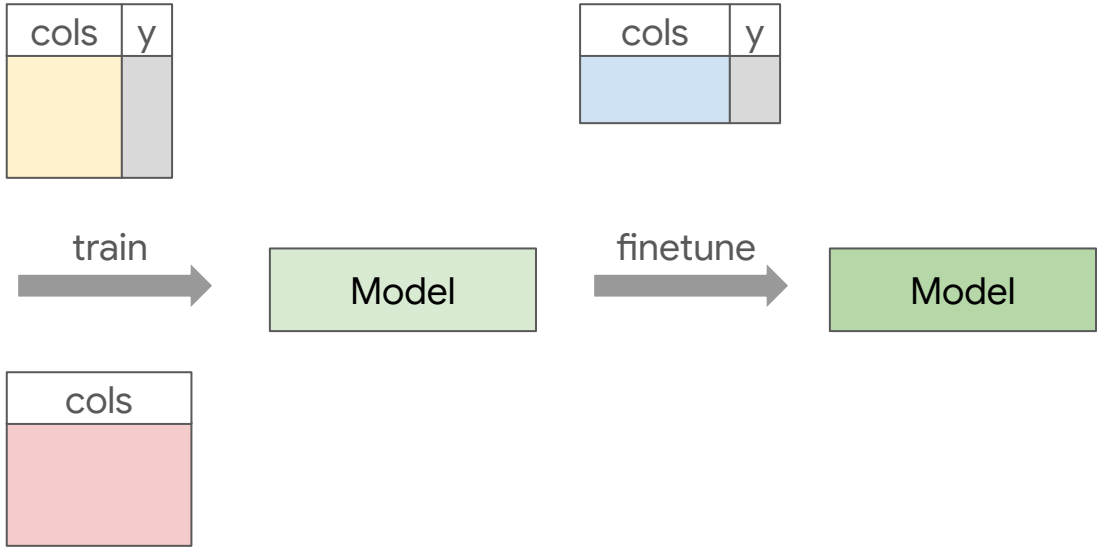
TransTab setup [Wang & Sun, 2022]



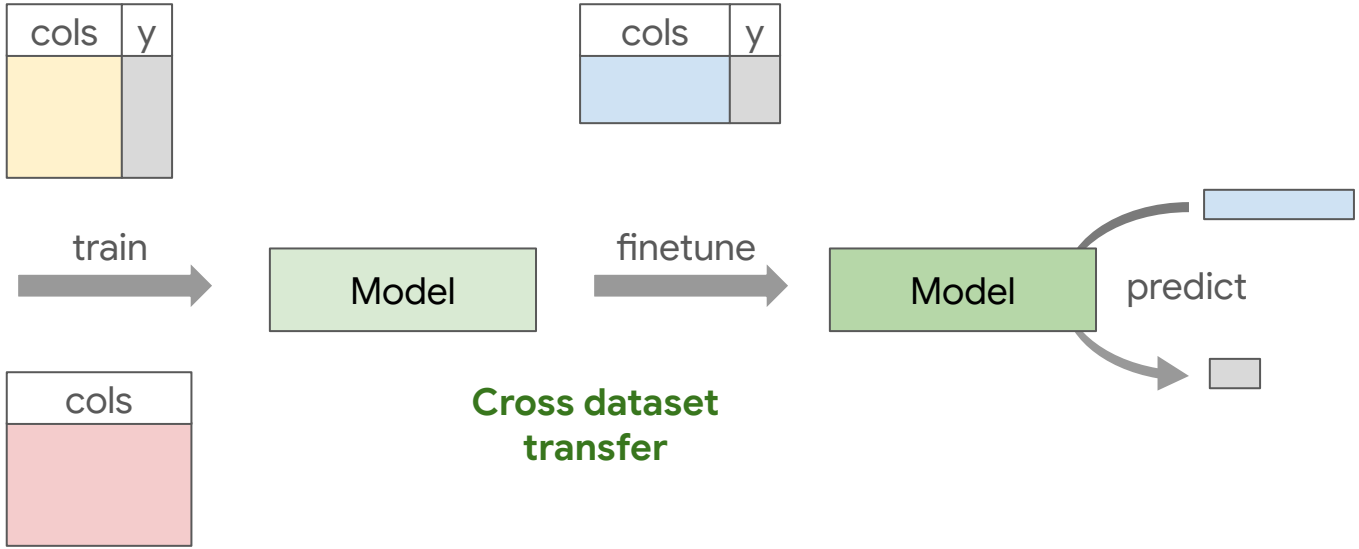
TransTab setup [Wang & Sun, 2022]



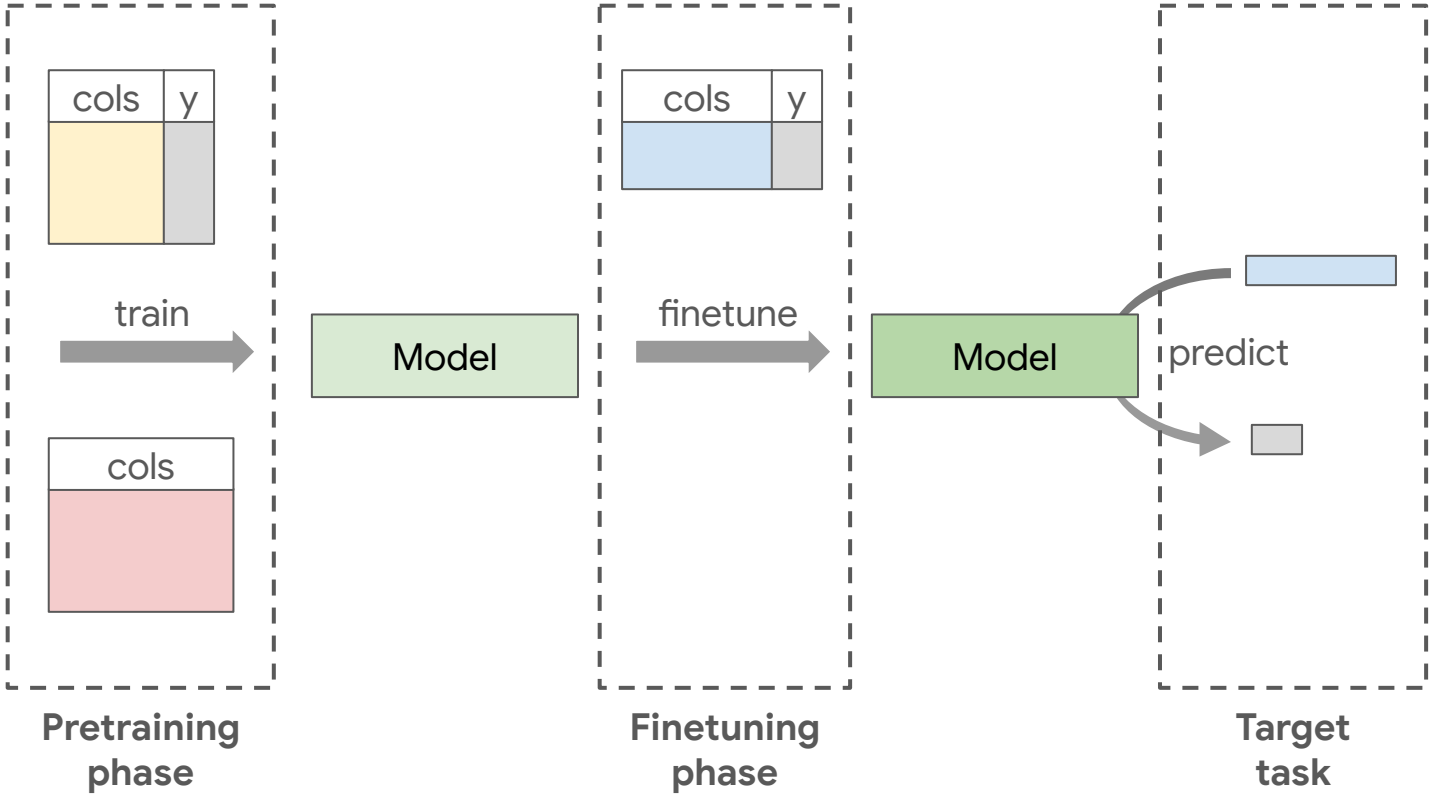
TransTab setup [Wang & Sun, 2022]



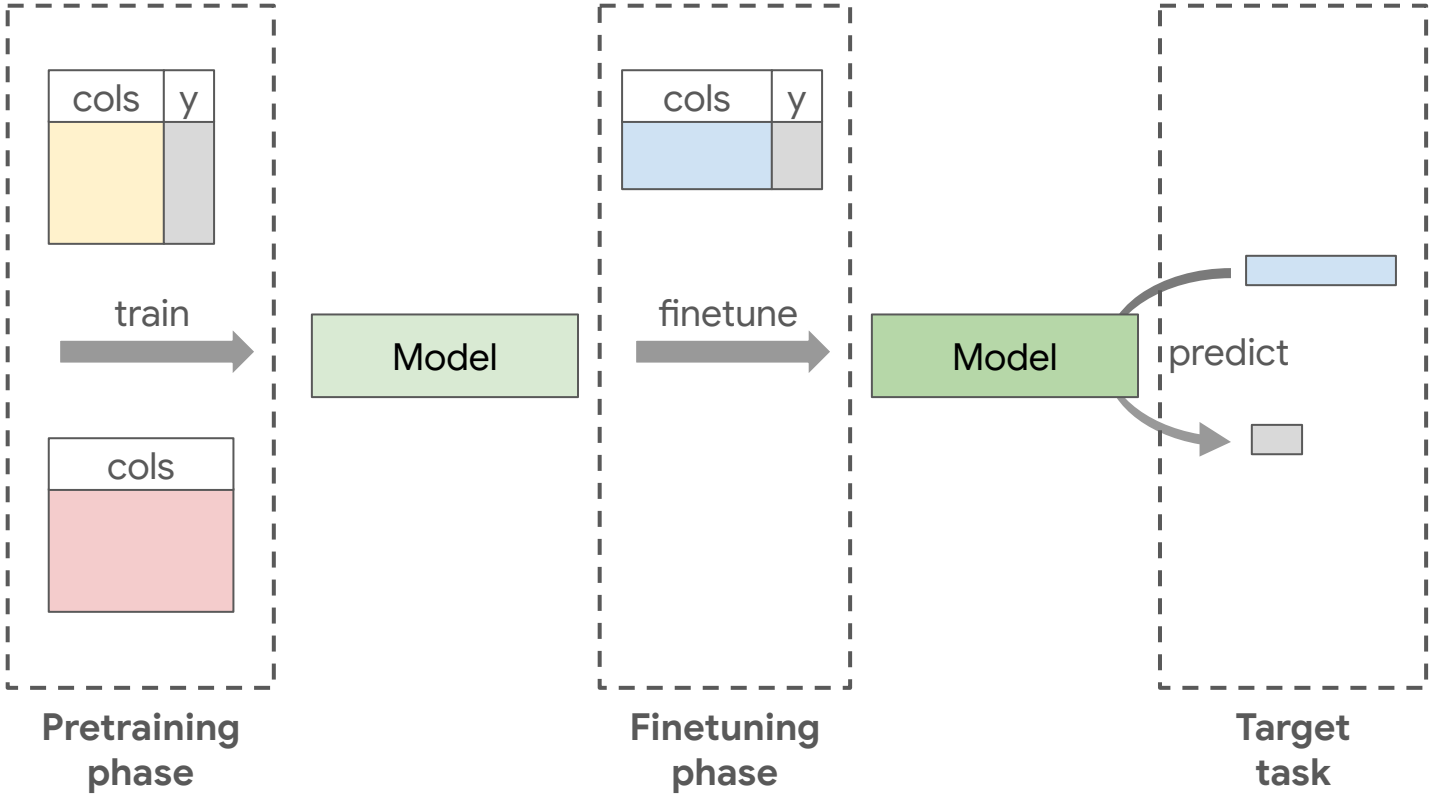
TransTab setup [Wang & Sun, 2022]



TransTab setup [Wang & Sun, 2022]



🤔 More pretraining data => Foundation Model? 🤔



🤔 More pretraining data => Foundation Model? 🤔

- Not so fast!
- Schema-independent tabular architecture is **necessary**, but **not sufficient**

Our approach

- New architecture to exploit tabular datasets' symmetries.
- New learning setup for training this model architecture.
- Harness embedding LLMs to build on LLMs' growing world knowledge.
- Include task descriptions and feature names to allow cross-table transfer learning and avoid negative transfer.
- Efficient use of LLMs through aggressive caching, so the number of LLM forward-passes scales with the schema, not the data.

It's cheap!!!

To exemplify the cost difference, consider 10 datasets, each with 10 million rows and 100 features (50 categorical with up to 100 categories, 50 numeric), assuming each cell contains 4 tokens, and using the pricing page from OpenAI [19]. We compare three ways of using LLMs:

- Decoder-only LLM (GPT-3.5 Turbo 4K): $10 \cdot 10$ million $\cdot\ 100 \cdot 4 \cdot (\$0.002 / 1000) = \$80{,}000$
- Embedding model (Ada v2, no caching): $10 \cdot 10$ million $\cdot\ 100 \cdot 4 \cdot (\$0.0001 / 1000) = \$4000$
- Embedding model (Ada v2, with caching): $10 \cdot (100 \cdot 4 + 50 \cdot 100) \cdot (\$0.0001 / 1000) < \$0.01$

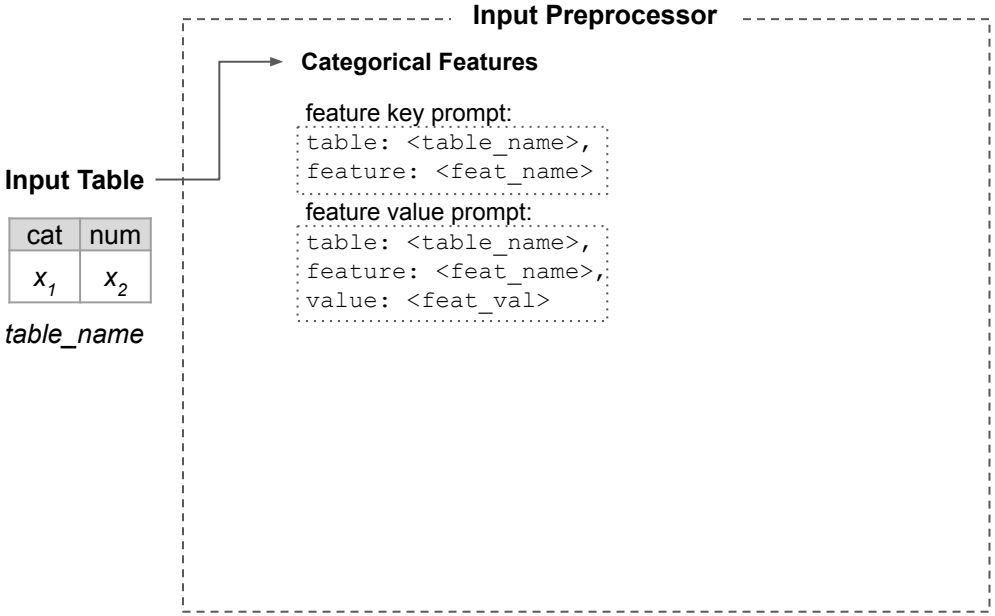
Such significant difference highlights the importance of employing embedding models to encode key table attributes with caching, as the fundamental design choice of `IngesTables`.

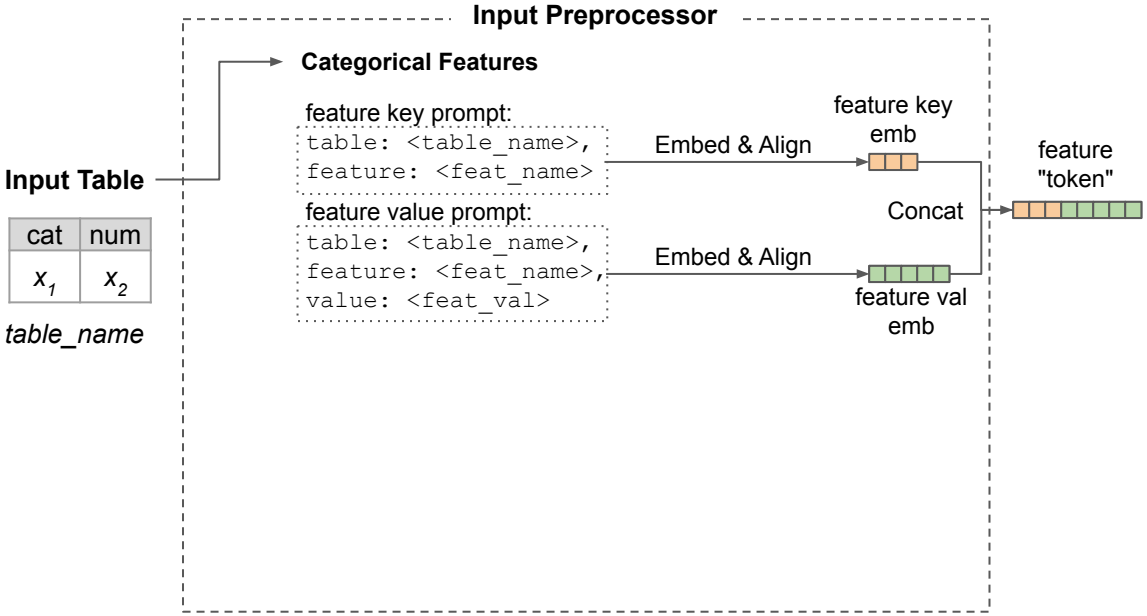
ours

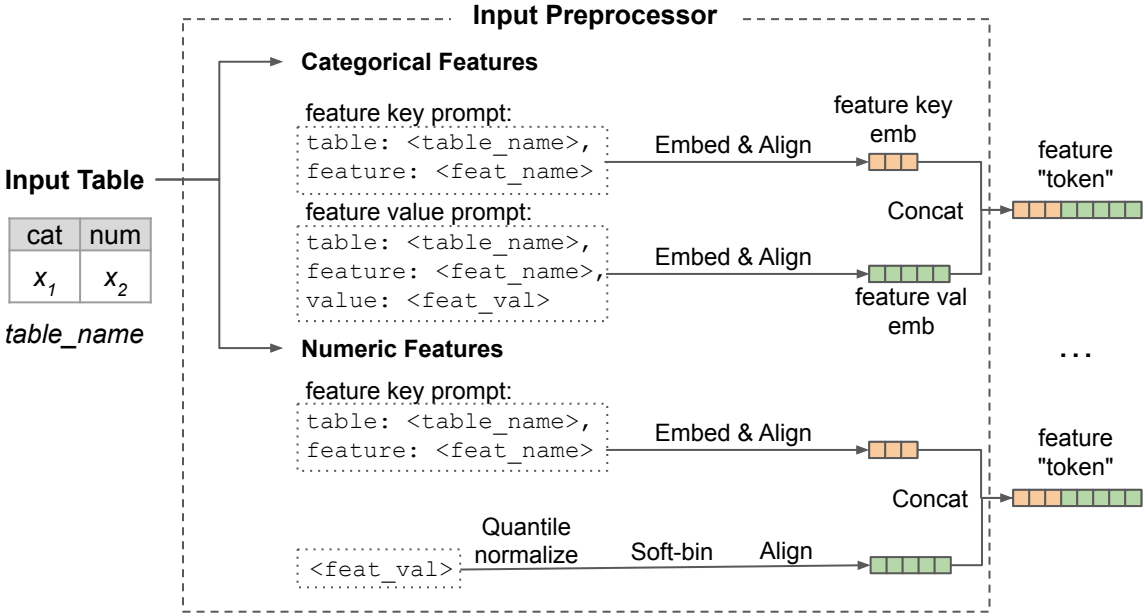
Input Table

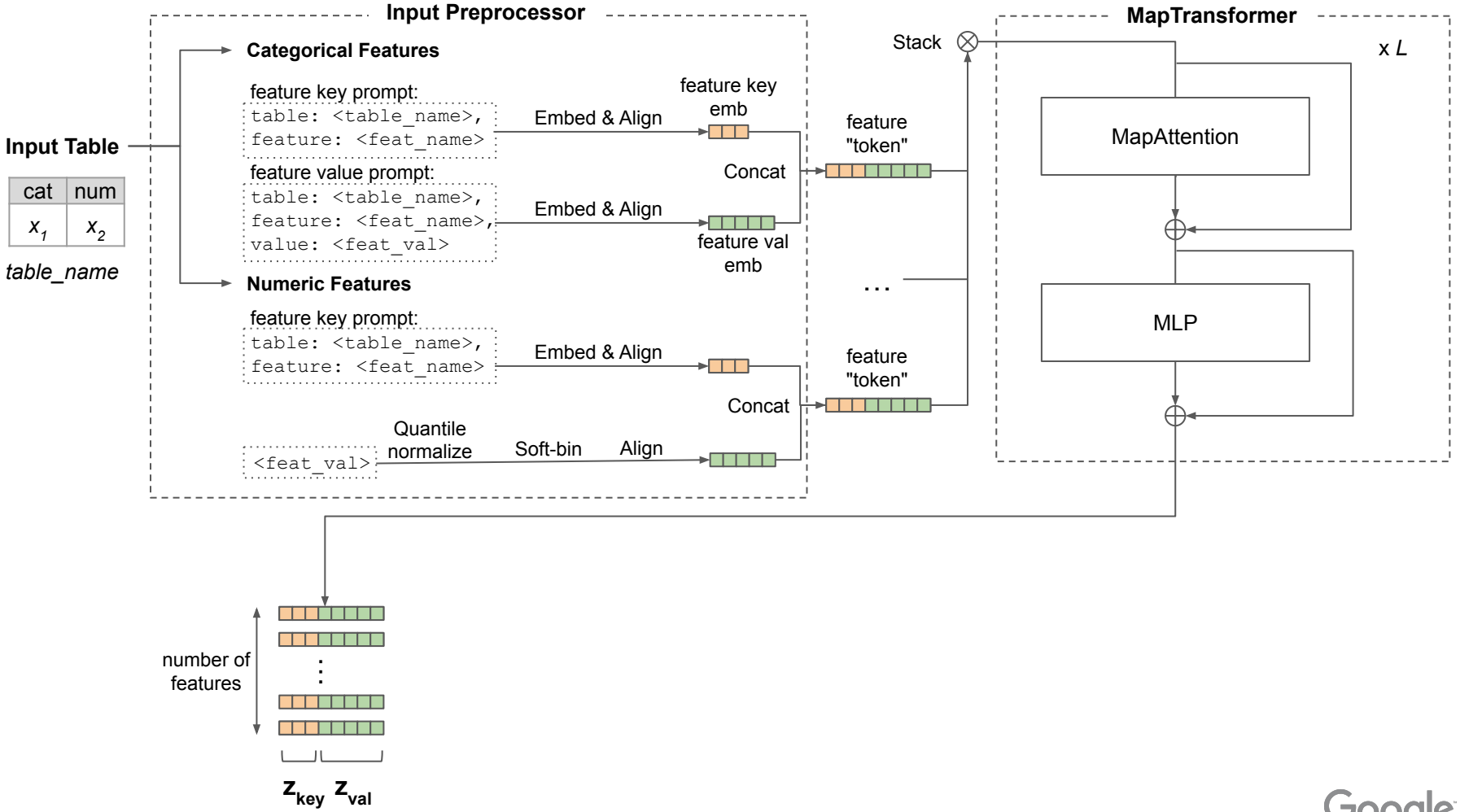
cat	num
x_1	x_2

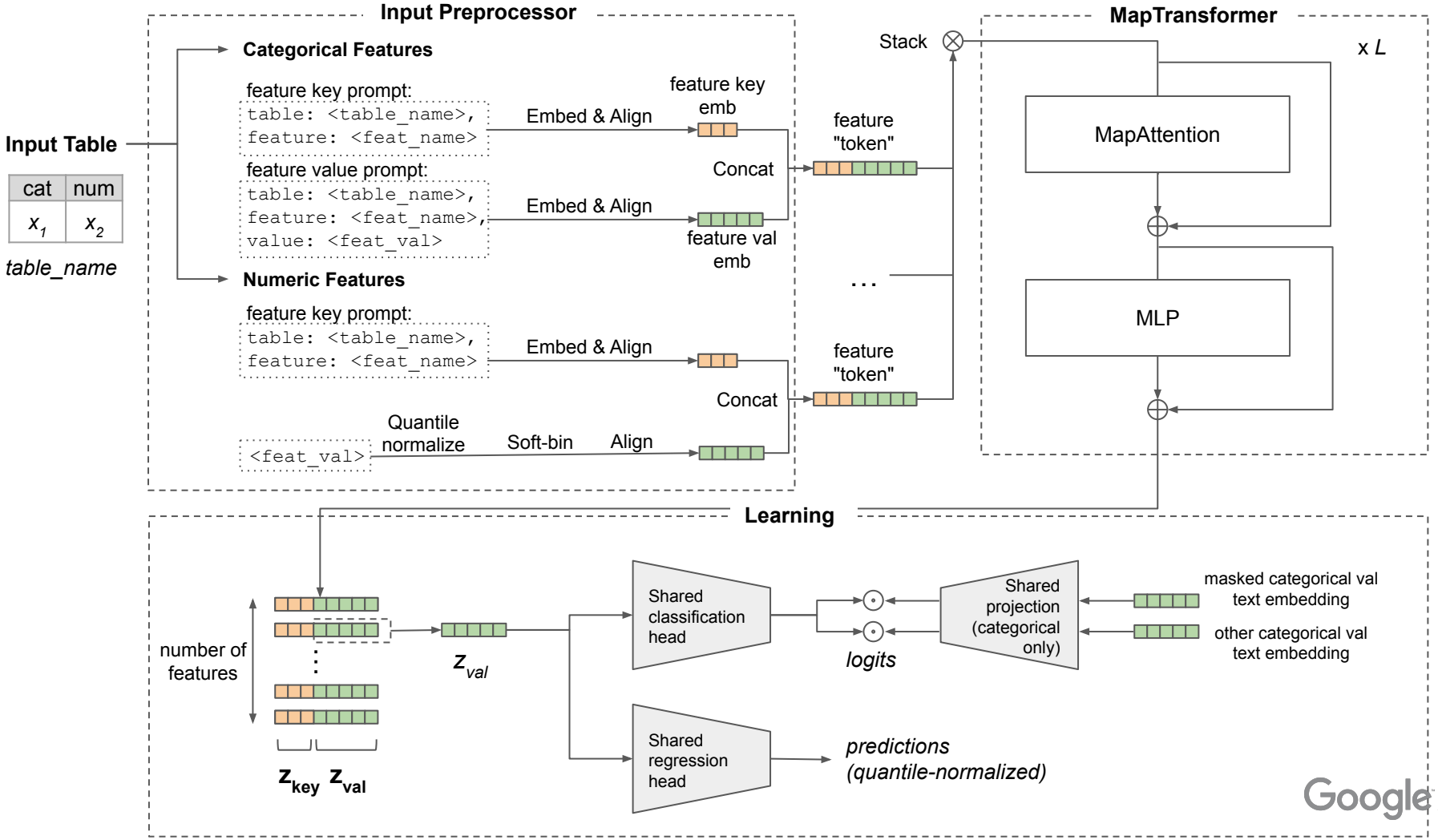
table_name


Input Preprocessor
Categorical Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
feature value prompt:
table: <table_name>,
feature: <feat_name>,
value: <feat_val>
Input Table
cat
num
x_1
x_2
table_name

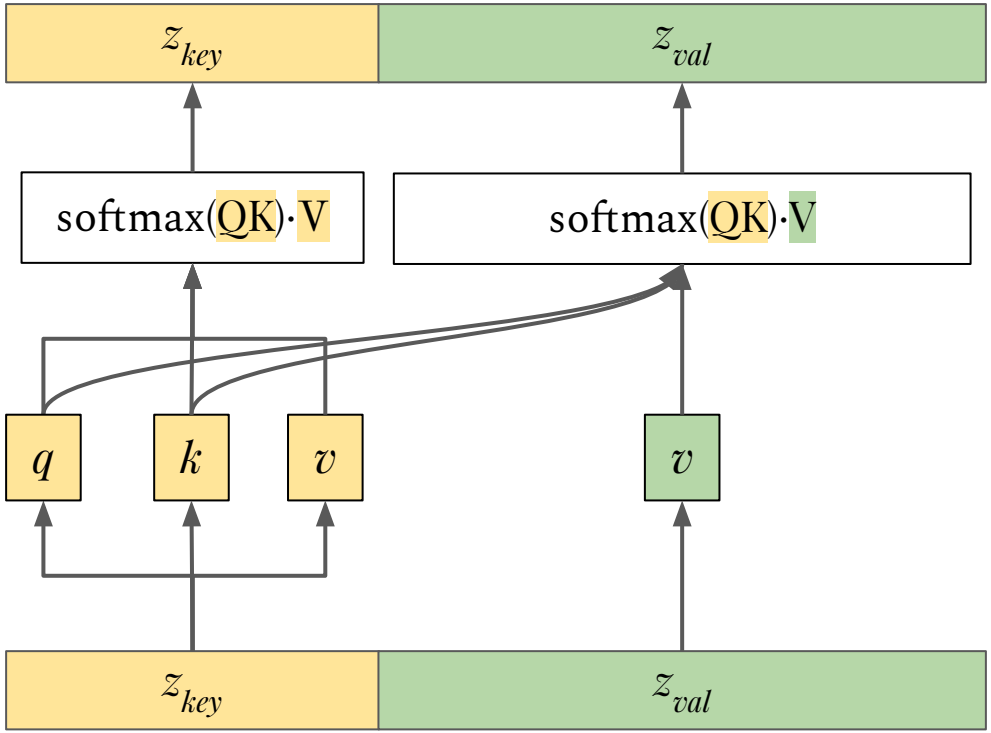

Input Preprocessor
Categorical Features
Input Table
cat
num
x_1
x_2
table_name
feature key prompt:
table: <table_name>,
feature: <feat_name>
feature value prompt:
table: <table_name>,
feature: <feat_name>,
value: <feat_val>
Embed & Align
Embed & Align
feature key emb
feature val emb
Concat
feature "token"


Input Preprocessor
Input Table
cat
num
x_1
x_2
table_name
Categorical Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
Embed & Align
feature key emb
feature value prompt:
table: <table_name>,
feature: <feat_name>,
value: <feat_val>
Embed & Align
feature val emb
Concat
feature "token"
Numeric Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
Embed & Align
Concat
feature "token"
Quantile normalize
Soft-bin
Align
<feat_val>
. . .


Input Preprocessor
MapTransformer
Input Table
cat
num
x_1
x_2
table_name
Categorical Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
Embed & Align
feature key emb
feature value prompt:
table: <table_name>,
feature: <feat_name>,
value: <feat_val>
Embed & Align
feature val emb
Concat
feature "token"
Numeric Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
Embed & Align
Concat
feature "token"
<feat_val>
Quantile normalize
Soft-bin
Align
Stack
x L
MapAttention
MLP
number of features
z_{key} z_{val}


Input Preprocessor
MapTransformer
Input Table
cat
num
x_1
x_2
table_name
Categorical Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
Embed & Align
feature key emb
feature value prompt:
table: <table_name>,
feature: <feat_name>,
value: <feat_val>
Embed & Align
feature val emb
Concat
feature "token"
Numeric Features
feature key prompt:
table: <table_name>,
feature: <feat_name>
Embed & Align
Concat
feature "token"
<feat_val>
Quantile normalize
Soft-bin
Align
Stack
x L
MapAttention
MLP
Learning
number of features
z_{val}
z_{key} z_{val}
Shared classification head
Shared regression head
logits
predictions (quantile-normalized)
Shared projection (categorical only)
masked categorical val text embedding
other categorical val text embedding

New building block: **`MapAttention`**



Results (AirBnB price prediction 2023)

	IngesTables		
	No pretraining	Pretrained w/ AirBnB 2019	
	Fully supervised	Zero-shot	Fully supervised
RMSE (AirBnB 2023)	0.1042	0.1009	**0.0884**

Pretraining helps!

Results (Clinical trial outcome prediction)

Trial Data	Metrics	XGBoost	FFNN	IngesTables	
Phase I	AUROC	0.518	0.550	0.647 ± 0.009	
	PRAUC	0.513	0.547	0.683 ± 0.007	+20%
Phase II	AUROC	0.600	0.611	0.655 ± 0.003	
	PRAUC	0.586	0.604	0.700 ± 0.002	+15%
Phase III	AUROC	0.667	0.681	0.720 ± 0.006	
	PRAUC	0.697	0.747	0.874 ± 0.003	+15%

outperforms general methods

Results (Clinical trial outcome prediction)

Trial Data	Metrics	XGBoost	FFNN	DeepEnroll	COMPOSE	HINT	SPOT	IngesTables
Phase I	AUROC	0.518	0.550	0.575	0.571	0.576	0.660	0.647 ± 0.009
	PRAUC	0.513	0.547	0.568	0.564	0.567	0.689	0.683 ± 0.007
Phase II	AUROC	0.600	0.611	0.625	0.628	0.645	0.630	0.655 ± 0.003
	PRAUC	0.586	0.604	0.600	0.604	0.629	0.685	0.700 ± 0.002
Phase III	AUROC	0.667	0.681	0.699	0.700	0.723	0.711	0.720 ± 0.006
	PRAUC	0.697	0.747	0.777	0.782	0.811	0.856	0.874 ± 0.003

comparable to specialized architectures

Results (Clinical trial outcome prediction)

Trial Data	Metrics	XGBoost	FFNN	DeepEnroll	COMPOSE	HINT	SPOT	AnyPredict	IngesTables
Phase I	AUROC	0.518	0.550	0.575	0.571	0.576	0.660	0.699	0.647 ± 0.009
	PRAUC	0.513	0.547	0.568	0.564	0.567	0.689	0.726	0.683 ± 0.007
Phase II	AUROC	0.600	0.611	0.625	0.628	0.645	0.630	0.706	0.655 ± 0.003
	PRAUC	0.586	0.604	0.600	0.604	0.629	0.685	0.733	0.700 ± 0.002
Phase III	AUROC	0.667	0.681	0.699	0.700	0.723	0.711	0.734	0.720 ± 0.006
	PRAUC	0.697	0.747	0.777	0.782	0.811	0.856	0.881	0.874 ± 0.003

underperforms LLM + pretrained BERT

Conclusion

- We present a new building block and learning setup to enable training Foundation Models for tabular datasets.
- We demonstrate that world knowledge can be harnessed cheaply by aggressively caching pretrained embedding LLM outputs.