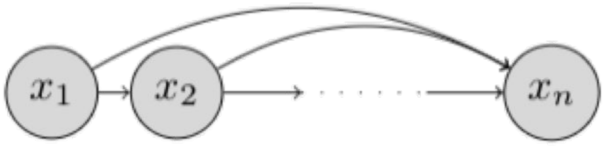
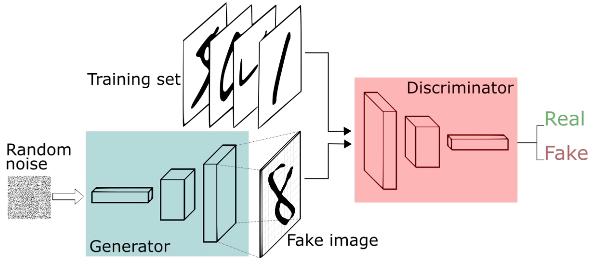
Deep Learning

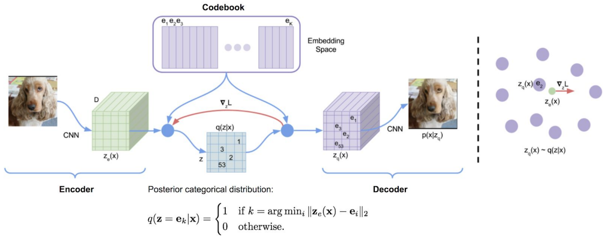
Lecture 13

Recap

- Autoregressive models
- GAN
- WGAN
- Image quality metrics
- GAN models
- VQ-VAE



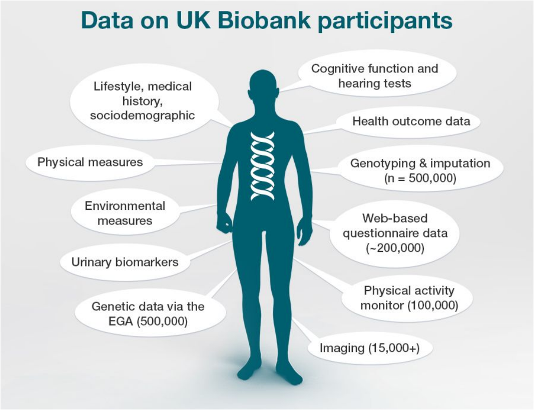




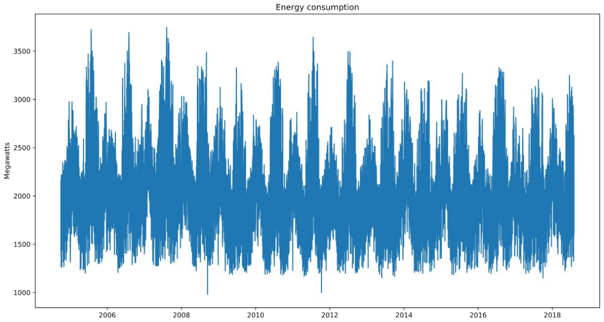
Tabular data

	app_id	amnt	currency	operation_kind	card_type	operation_type
0	0	0.465425	1	4	98	4
1	0	0.000000	1	2	98	7
2	0	0.521152	1	2	98	3
3	0	0.356078	1	1	5	2
4	0	0.000000	1	2	98	7

Transactional data



Medical data

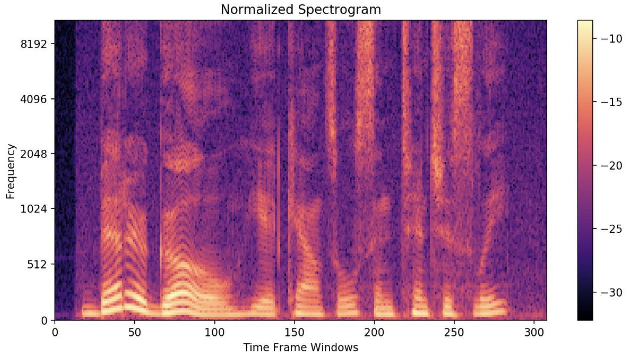


Time-series data

Tabular Data

	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amount
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95
6	USA	Finchley	3/01/2011	10394	10	442.00
7	UK	Gillingham	3/01/2011	10395	9	2,122.92
8	USA	Finchley	6/01/2011	10396	7	1,903.80
9	USA	Callahan	8/01/2011	10399	17	1,765.60
10	USA	Fuller	8/01/2011	10404	7	1,591.25
11	USA	Fuller	9/01/2011	10398	11	2,505.60
12	USA	Coghill	9/01/2011	10403	18	855.01
13	USA	Finchley	10/01/2011	10401	7	3,868.60
14	USA	Callahan	10/01/2011	10402	11	2,713.50
15	UK	Rayleigh	13/01/2011	10406	15	1,830.78
16	USA	Callahan	14/01/2011	10408	10	1,622.40
17	USA	Farnham	14/01/2011	10409	19	319.20
18	USA	Farnham	15/01/2011	10410	16	802.00

Heterogeneous data -
different sources

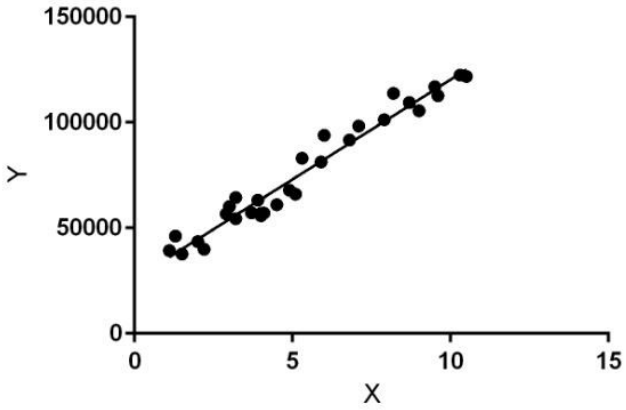


Homogeneous data -
only one source

Tabular data specifics

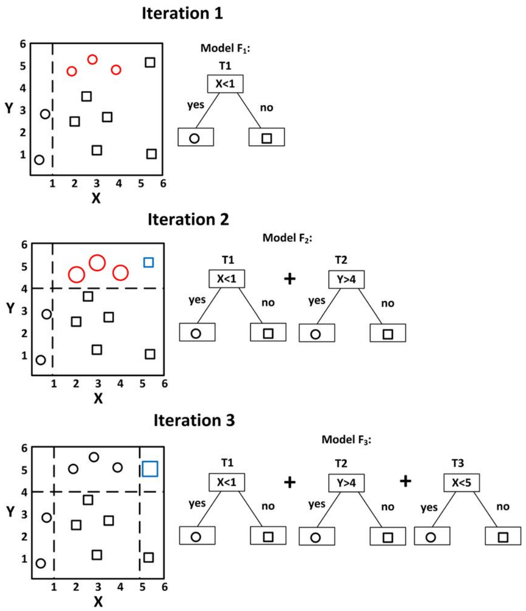
- Lack of transferability & no spatial dependencies (No inductive biases)
- Missing/Noisy data
- The role of one feature can be significant
- There is no standard benchmark (GLUE, ImageNet)

Classical methods



Linear regression

$$\mathbf{y} = X\boldsymbol{\beta} + \varepsilon$$

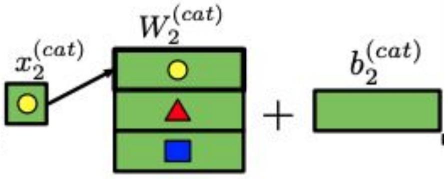


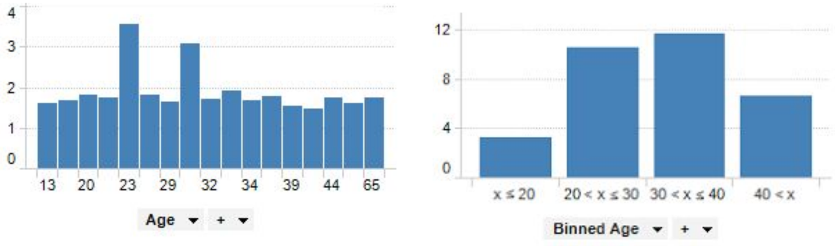
Gradient boosting

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x),$$

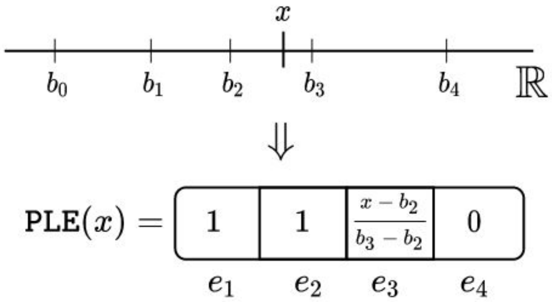
Encoding / Feature engineering

- Numerical features
  - Normalization
  - Discretization -> Embedding
  - Discretization -> Piecewise Linear Encoding
- Categorical features
  - Embedding





Discretization



[On Embeddings for Numerical Features in Tabular Deep Learning]()

Piecewise Linear Encoding

Encoding / Feature engineering

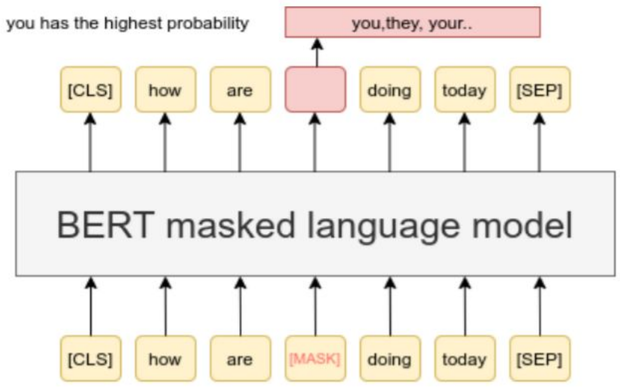
- Numerical discretization
  - Quantile transformation
  - Target-aware. Discretization is done by constructing decision tree

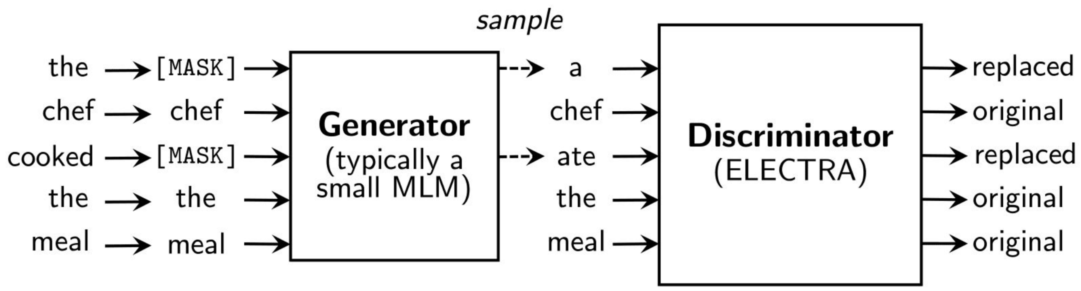
- Time Encoding
  - learnable time-dependent vector for position embedding

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$

[Time2Vec: Learning a Vector Representation of Time]

Pretraining: Masked Language Modelling



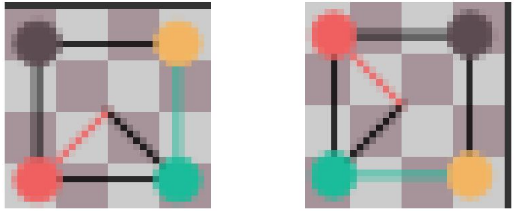


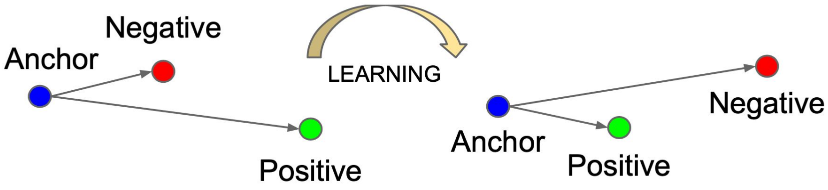
Learning representation which will be able to recover masked parts

Pretraining

The idea of pretraining is to start learning a task not from scratch but from some good representation:

- Invariance learning (Rotation invariance)
- Common sense (Semantically close objects should be close in latent space)





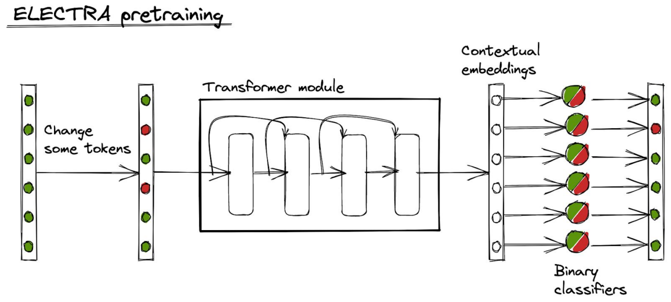
Pretraining

Reconstruction

- Reconstruction of the original input, given the corrupted input. Corruption can be done through feature resampling

Masking

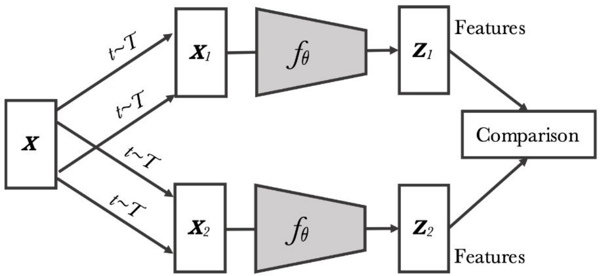
- Given masked input. Predict what column was masked

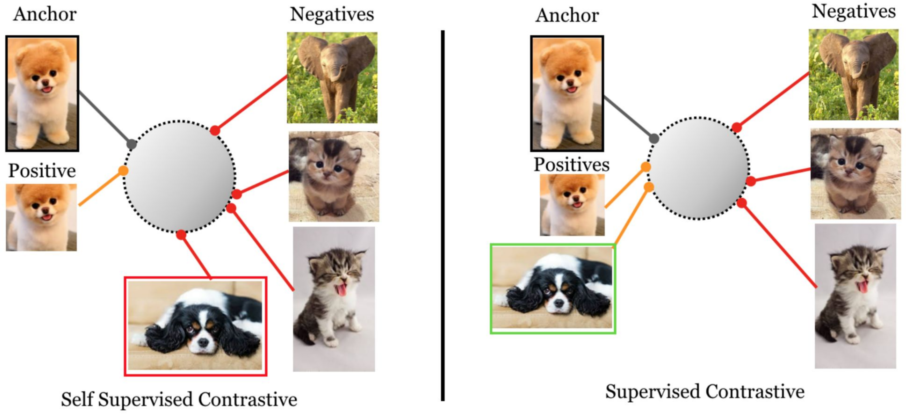


Name	Age	Gender
Vadim	50	Male
[MASK]	14	Prefer not to say

Revisiting Pretraining Objectives for Tabular Deep Learning

Pretraining





Contrastive

- Forcing to different views of object be close to each other

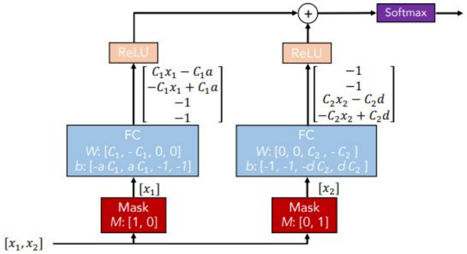
Supervised/Target-aware

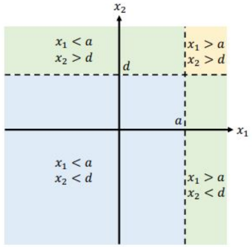
- Augmentation or regularization through self-pretraining
- Target prediction as mask | Resampling label conditioned distribution

Models

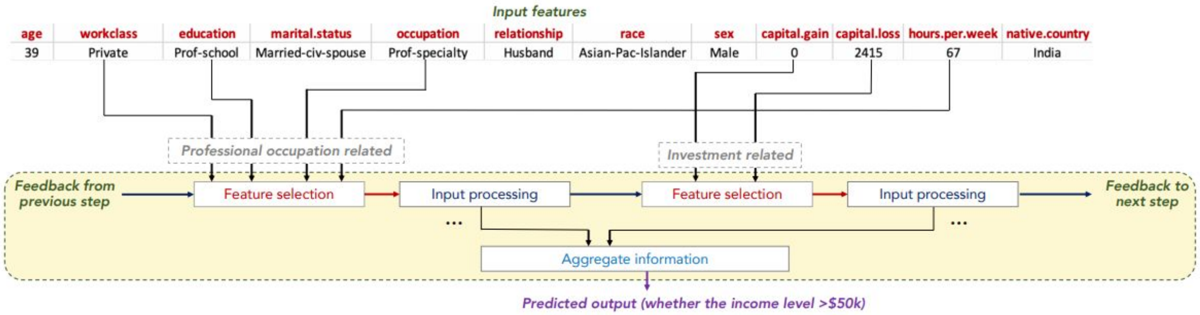
TabNet

Motivation: replace decision trees with neural network





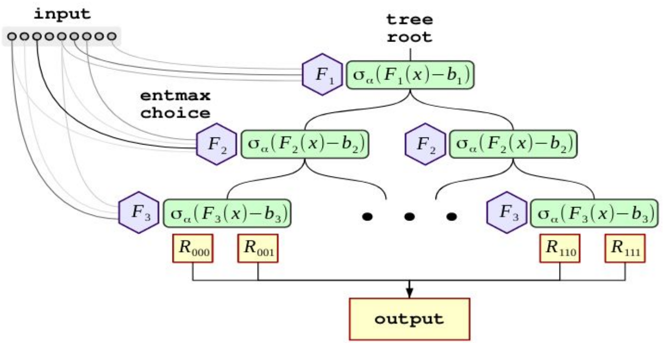
TabNet



$$\mathbf{M[i]} = \mathrm{sparsemax}(\mathbf{P[i-1]} \cdot \mathrm{h}_i(\mathbf{a[i-1]}))$$

$$P[i] = \prod_{j=1}^{i=1} (\gamma - M[j]),$$

Neural Oblivious Decision Ensembles

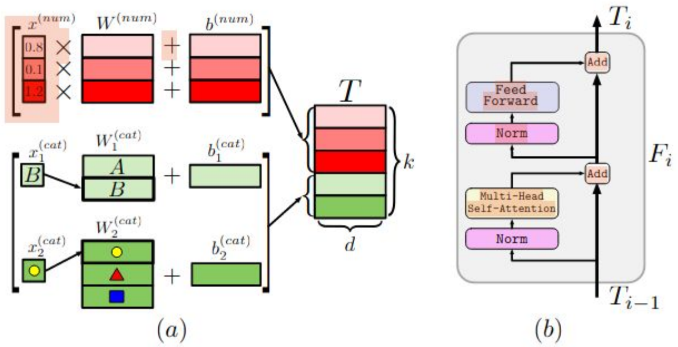


$$\hat{h}(x) = \sum_{i_1,\dots i_d \in \{0,1\}^d} R_{i_1,\dots,i_d} \cdot C_{i_1,\dots,i_d}(x)$$

$$\hat{f}_i(x) = \sum_{j=1}^{n} x_j \cdot entmax_\alpha(F_{ij})$$

[Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data]

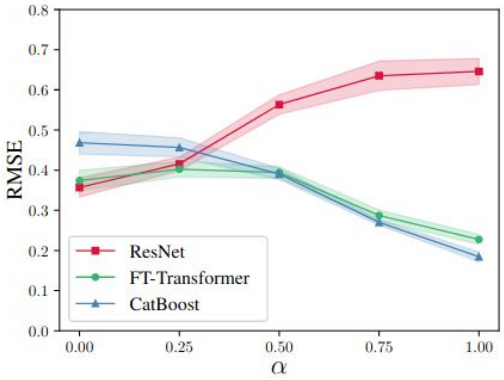
FT-Transformer



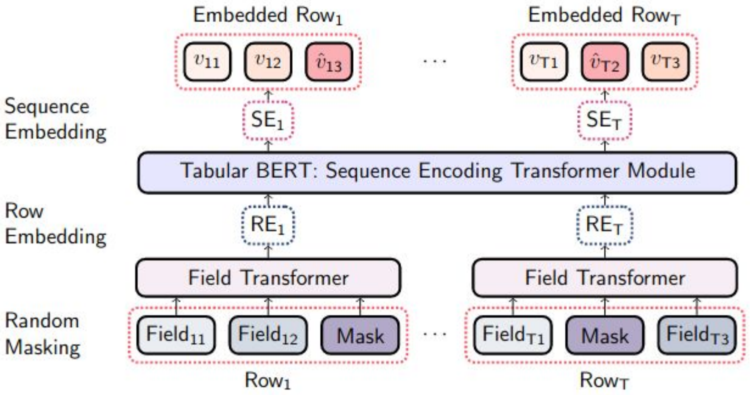
$$\texttt{ResNet}(x) = \texttt{Prediction}(\texttt{ResNetBlock}(\ldots(\texttt{ResNetBlock}(\texttt{Linear}(x)))))$$
$$\texttt{ResNetBlock}(x) = x + \texttt{Dropout}(\texttt{Linear}(\texttt{Dropout}(\texttt{ReLU}(\texttt{Linear}(\texttt{BatchNorm}(x))))))$$
$$\texttt{Prediction}(x) = \texttt{Linear}(\texttt{ReLU}(\texttt{BatchNorm}(x)))$$

$$x \sim \mathcal{N}(0, I_k), \qquad y = \alpha \cdot f_{GBDT}(x) + (1 - \alpha) \cdot f_{DL}(x).$$

[Revisiting Deep Learning Models for Tabular Data]()

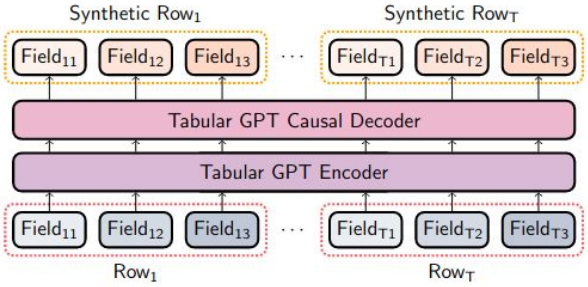


TabGPT/TabBERT



First, we consider **intra-transaction** relationships (how is this feature connected to another).

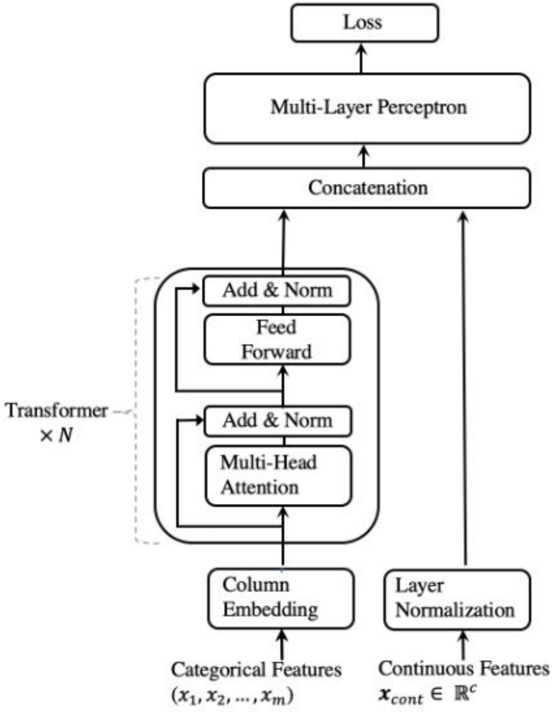
Second, we consider **inter-transaction** relationships (how do these transactions connected with each other)



Transaction generation using TabGPT

[Tabular Transformers for Modeling Multivariate Time Series]

TabTransformer

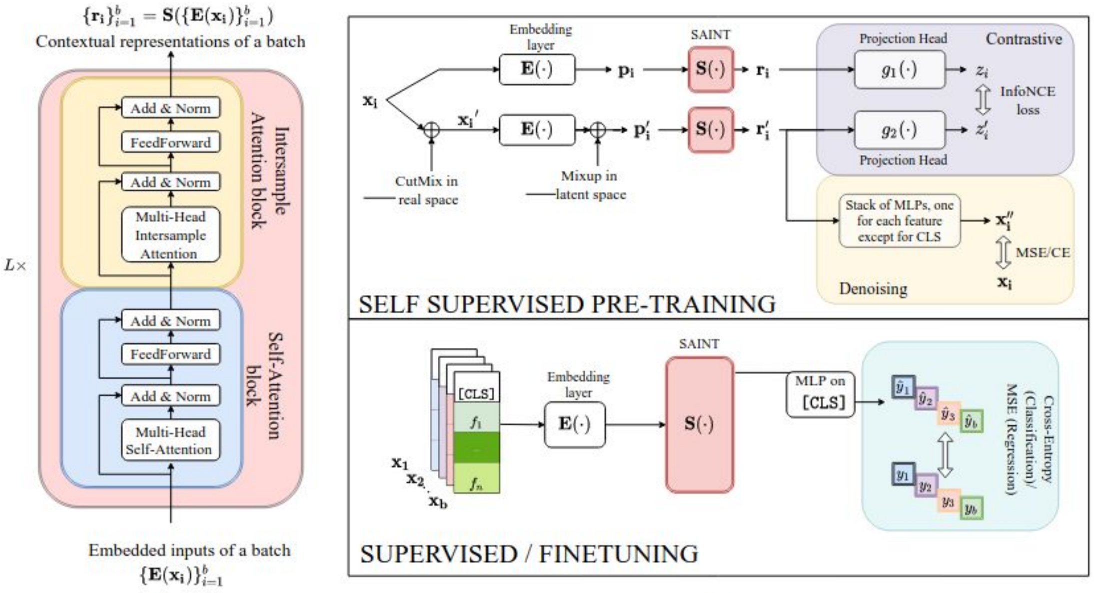


Categorical features can be interpreted as text:
rare categories ~ rare words, similar categories ~ synonyms

Adding context in features is crucial: 2 month dog ≠ 12 year dog

[TabTransformer: Tabular Data Modeling Using Contextual Embeddings]

SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training



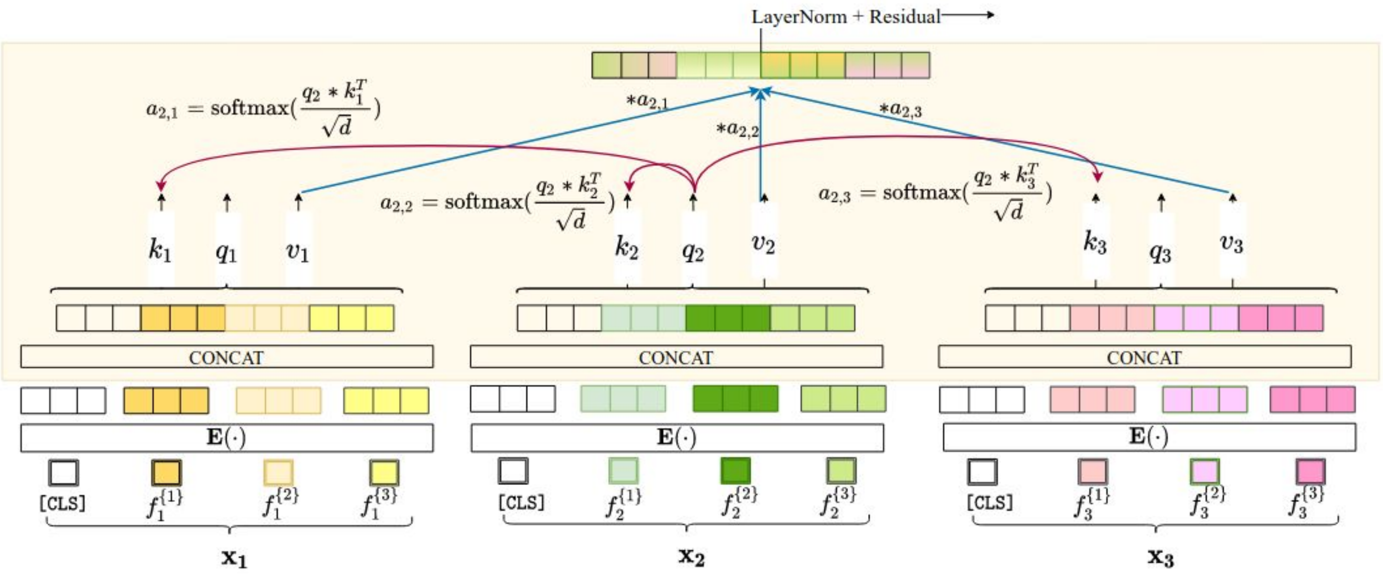
(a)

(b)

SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training

SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training

Intersample attention



SAINT

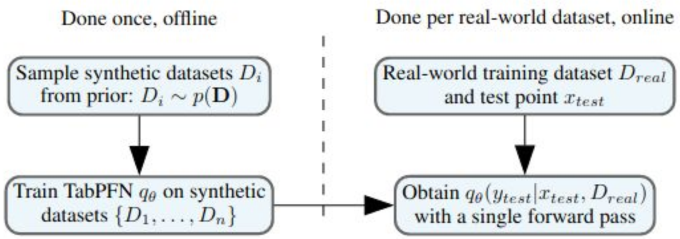
Dataset size	45,211	7,043	452	200	495,141	12,330	32,561	58,310	60,000	
Feature size	16	20	226	783	49	17	14	147	784	
Model \ Dataset	Bank	Blastchar	Arrhythmia	Arcene	Forest	Shoppers	Income	Volkert†	MNIST†	Mean
Logistic Reg.	90.73	82.34	86.22	91.59	84.79	87.03	92.12	53.87	89.89*	89.25
Random Forest	89.12	80.63	86.96	79.17	98.80	89.87	88.04	66.25	93.75	89.52
XGBoost [4]	92.96	81.78	81.98	81.41	95.53	92.51	92.31	68.95	94.13*	91.06
LightGBM [22]	93.39	83.17	88.73	81.05	93.29	**93.20**	**92.57**	67.91	95.2	90.13
CatBoost [10]	90.47	84.77	87.91	82.48	85.36	93.12	90.80	66.37	96.6	90.73
MLP	91.47	59.63	58.82	90.26	96.81	84.71	92.08	63.02	93.87*	84.59
VIME [49]	76.64	50.08	65.3	61.03	75.06	74.37	88.98	64.28	95.77*	76.07
TabNet [1]	91.76	79.61	52.12	54.10	96.37	91.38	90.72	56.83	96.79	83.88
TabTransf. [18]	91.34	81.67	70.03	86.8	84.96	92.70*	90.60*	57.98	88.74	90.86
SAINT-s	**93.61**	**84.91**	93.46	86.88	99.67	92.92	91.79	62.91	90.52	92.59
SAINT-i	92.83	84.46	**95.8**	**92.75**	99.45	92.29	91.55	**71.27**	**98.06**	93.09
SAINT	93.3	84.67	94.18	91.04	**99.7**	93.06	91.67	70.12	97.67	**93.13**

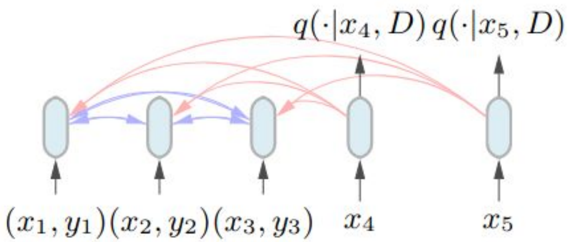
TabPFN

Let's adjust classical likelihood

$$p(y|x, D) \propto \int_{\Phi} p(y|x, \phi)p(D|\phi)p(\phi)d\phi.$$

And train the model to approximate given likelihood. Then, we can predict values on new datasets in **zero-shot manner**

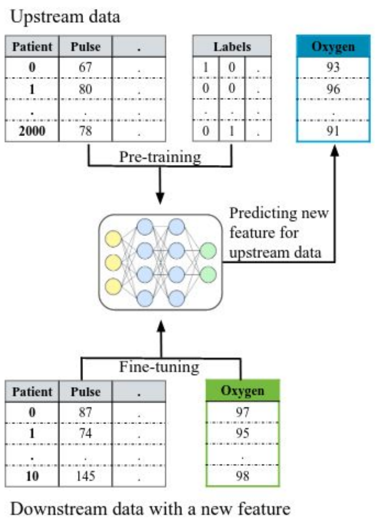




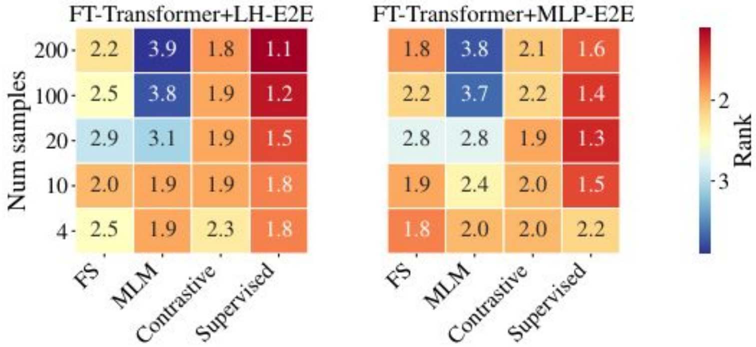
[TabPFN: A Transformers That Solve Small Tabular Classification Problems in a Second]()

Transfer Learning with Deep Tabular Models

How to add new feature in the model?



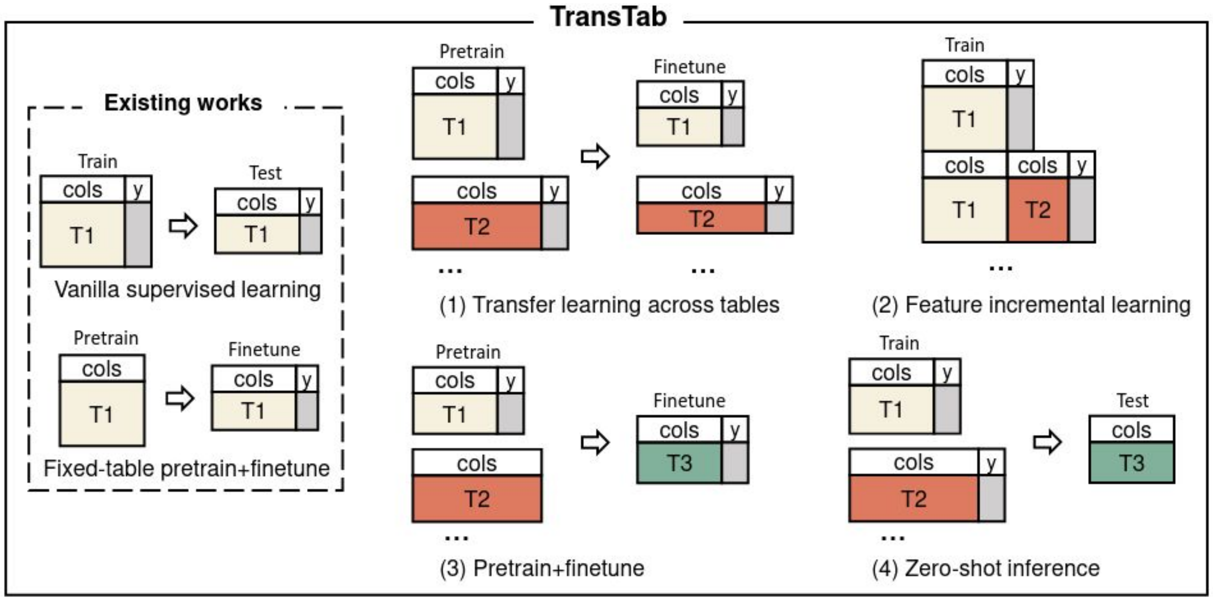
What pretraining method is the best?



[Transfer Learning with Deep Tabular Models]

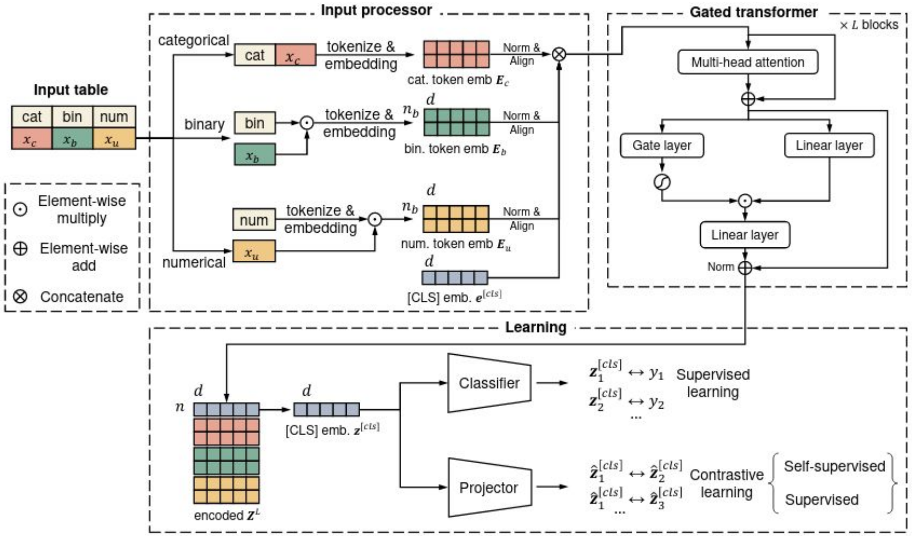
FS = from scratch

TransTab

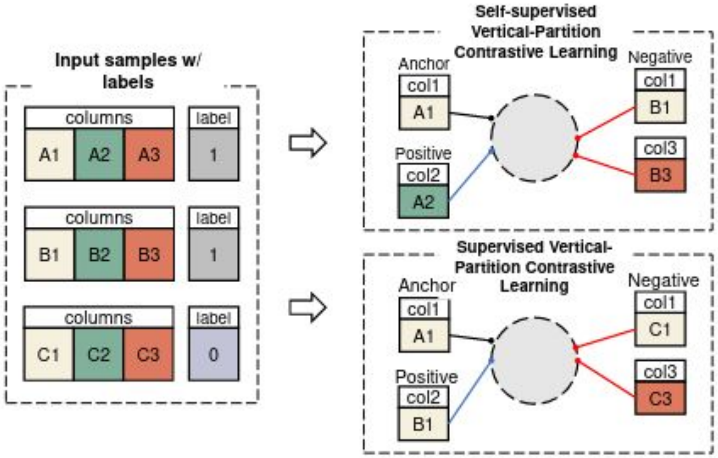


[TransTab: Learning Transferable Tabular Transformers Across Tables]

TransTab



Input processor
Gated transformer
× L blocks
Input table
cat
bin
num
categorical
binary
numerical
tokenize & embedding
Norm & Align
cat. token emb E_c
bin. token emb E_b
num. token emb E_u
[CLS] emb. $e^{[cls]}$
Multi-head attention
Gate layer
Linear layer
Norm
Element-wise multiply
Element-wise add
Concatenate
Learning
encoded Z^L
[CLS] emb. $z^{[cls]}$
Classifier
Projector
Supervised learning
Contrastive learning
Self-supervised
Supervised



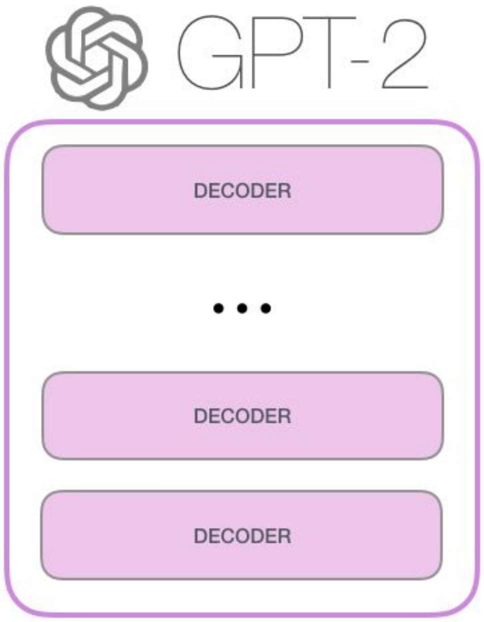
Input samples w/ labels
columns
label
A1 A2 A3
B1 B2 B3
C1 C2 C3
1
1
0
Self-supervised Vertical-Partition Contrastive Learning
Supervised Vertical-Partition Contrastive Learning
Anchor
Positive
Negative
col1
col2
col3

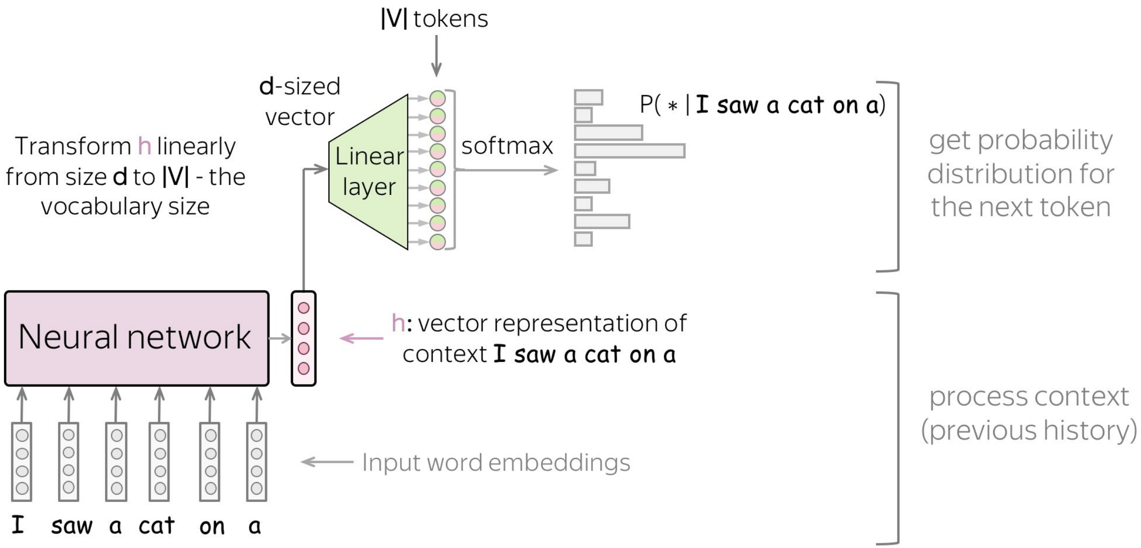
Notes

- DL models are not better than gradient boosting but have a potential
- DL models + XGBoost = Performance boost
- Better results = Proper hyperparameter search + regularization

Tabular data as a text

Small reminder



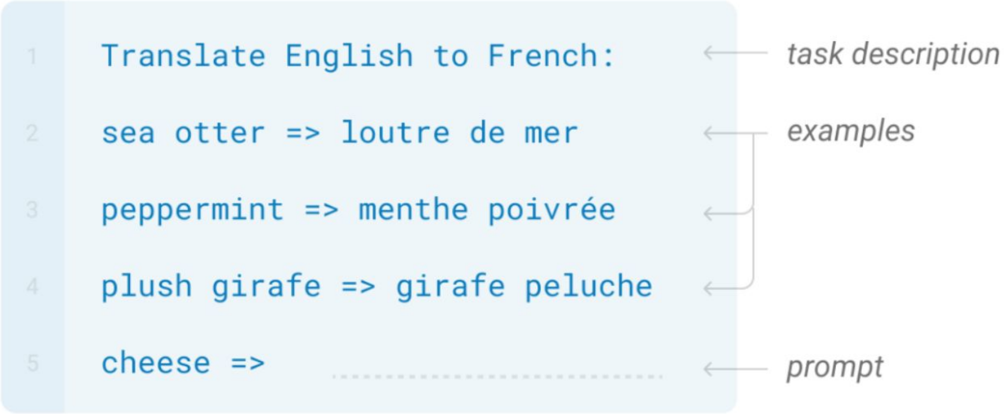


Q: Who is Batman?
A: Batman is a fictional comic book character.

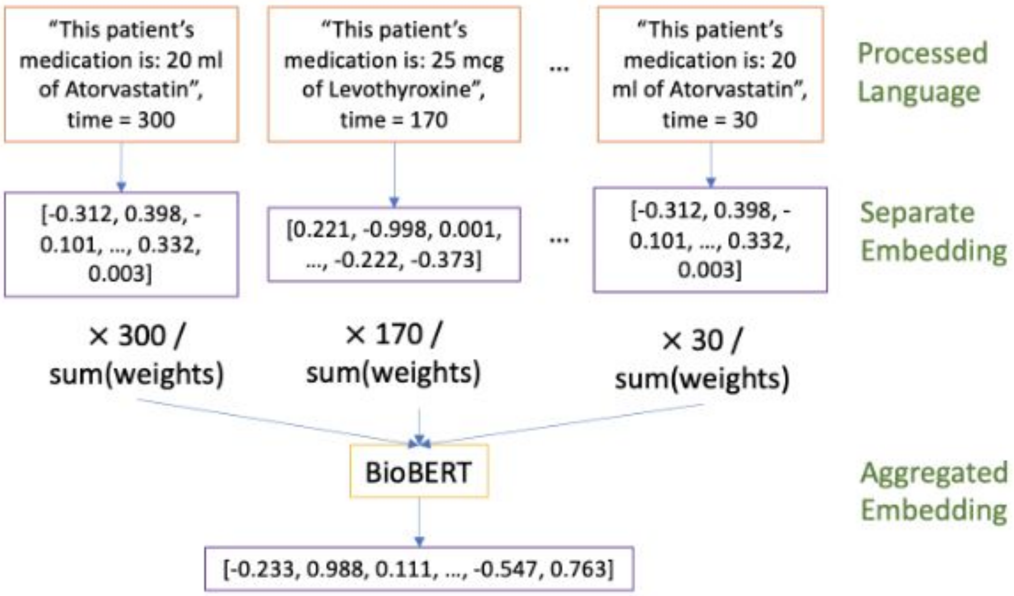
Prompt engineering

Few-shot

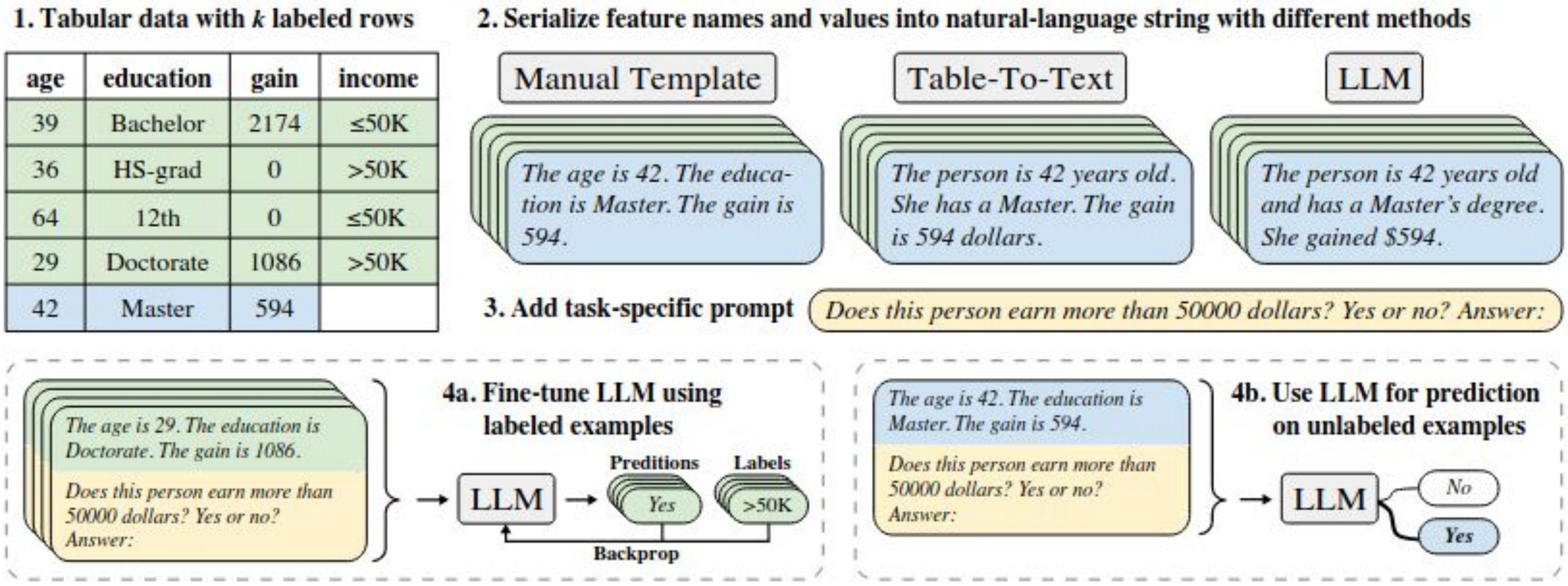
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



TabText

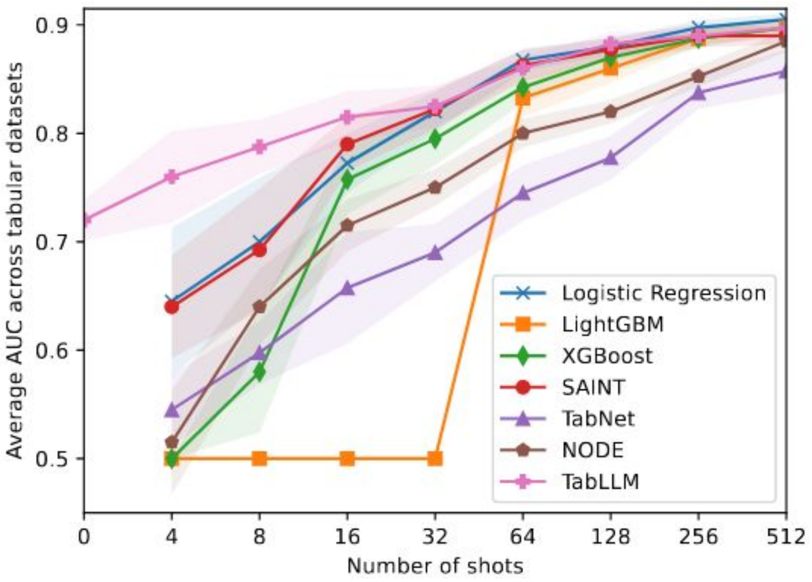

"This patient's medication is: 20 ml of Atorvastatin", time = 300
"This patient's medication is: 25 mcg of Levothyroxine", time = 170
...
"This patient's medication is: 20 ml of Atorvastatin", time = 30
Processed Language
[-0.312, 0.398, -0.101, ..., 0.332, 0.003]
[0.221, -0.998, 0.001, ..., -0.222, -0.373]
...
[-0.312, 0.398, -0.101, ..., 0.332, 0.003]
Separate Embedding
× 300 / sum(weights)
× 170 / sum(weights)
× 30 / sum(weights)
BioBERT
Aggregated Embedding
[-0.233, 0.988, 0.111, ..., -0.547, 0.763]

Tab LLM



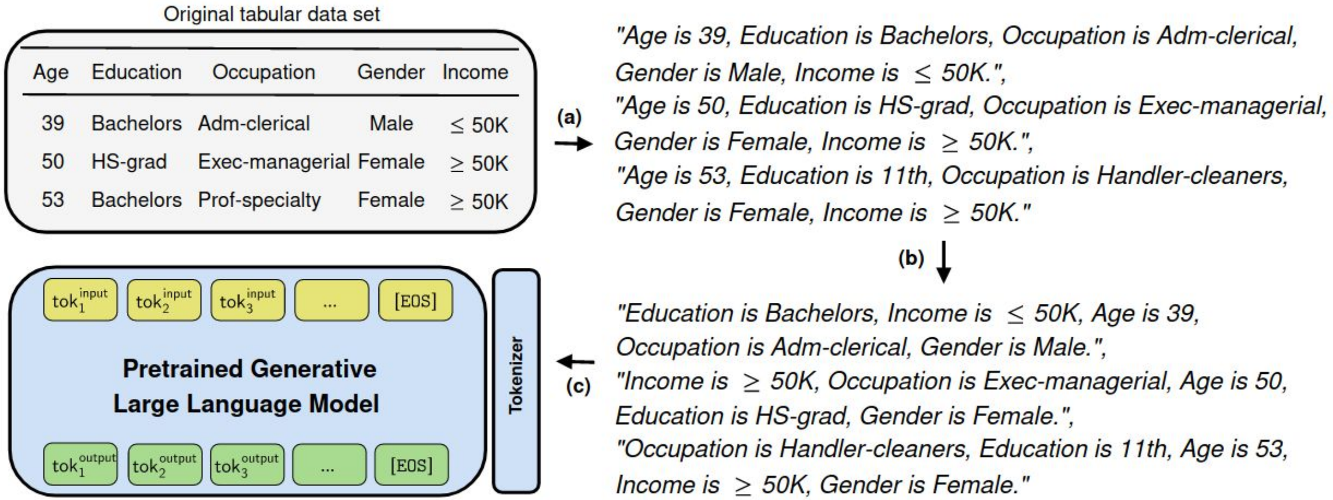
[TabLLM: Few-shot Classification of Tabular Data with Large Language Models]()

Tab LLM



Average AUC across tabular datasets
0.9
0.8
0.7
0.6
0.5
0
4
8
16
32
64
128
256
512
Number of shots
Logistic Regression
LightGBM
XGBoost
SAINT
TabNet
NODE
TabLLM

Language Models are Realistic Tabular Data Generators



[Language Models are Realistic Tabular Data Generators]

Recap

- Encoding
- Pretraining
- Tabular DL
- Tabular DL as text