Developing **Foundation Models** for Real-World **Tabular Data**

Marta Garnelo, Wojciech Marian Czarnecki

Fundamental

> *"One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great."*
>
> **The Bitter Lesson** - Richard Sutton

OUR NORTH STAR

It has been over six years since Richard Sutton wrote his seminal reflection on the superiority of scalable, learning based systems over solutions based on human intuition and knowledge [38]. With the advent of LLMs, that have not only redefined the field of natural language processing, but the industry at large, one could think that we have all internalized this mindset and the era of human-crafted solutions in machine learning is coming to an end. Perhaps disappointingly, when it comes to solving real life problems with supervised machine learning, we still rely on human intuitions and analysis. Enterprises still depend on decision trees built on hand crafted, engineered features. And even the biggest breakthroughs in supervised deep learning - a few of which are listed below - are often achieved with the help of large amount of human-knowledge, distilled into specific data pipelines, architectural choices, custom auxiliary losses etc.

- The AlphaFold model [22], which is able to predict 3D structures from protein sequences at an unprecedented quality and contributed to the Nobel prize in biology.
- The discovery of Halicin [37], a broad-spectrum antibiotic that was identified by a Deep Learning (DL) model that screened what was considered to be an intractable search space.
- The expansion of the known stable materials space (and therefore also of the catalogue of potential superconductors, batteries, photovoltaics, etc.) by an order of magnitude using supervised property prediction [27].
- The paradigm shift in weather forecasting as recent weather prediction models consistently beat flagship physics-based models while running on a single accelerator [25].
- Large-scale recommendation and ad-click prediction systems on major platforms (e.g. Google Play or YouTube) which are now built on supervised deep neural networks trained on massive sparse/dense feature logs [9, 10, 29, 28].

These breakthroughs, while seemingly unrelated, all have a few things in common. Firstly, they can all be framed as *tabular prediction problems*. Take the drug discovery example: here each molecular structure and its properties would constitute a row in a large table of chemical compounds. Given this table, the goal is to predict the target column

OUR NORTH STAR

for an unlabelled new row. This could mean regressing the value of the binding affinity or classifying whether or not the molecule would work as an antibiotic. Similarly, weather forecasting, material search and recommender systems can be cast onto tabular prediction problems. This seemingly trivial re-framing points towards a specific research direction with exciting future opportunity.

Another point in common between these milestones is their reliance on large amounts of resources in order to succeed. Generally speaking, one could expect that each of them called for a team of dozens of experts working together over several months, if not years. In addition to manpower they would also require spending huge amounts of resources on compute, most of which would be spent on the exploration and development of the models. Recent advances in DL research beg the question whether we are at a point where these efforts can be universally automated with a single prediction engine. In such a paradigm the responsibility of the researcher no longer lies in endlessly iterating over a single model to reach a particular objective. Instead, their task is to identify grand challenges that can be expressed as tabular prediction problems and use the automated model as a universal tool to solve them. Crucially, such a model would allow this (and any other) researcher to leverage the expertise, data and compute invested in its development. We will refer to this model as *universal predictor* going forward to reflect its capabilities.

The final commonality between the aforementioned discoveries is their grounding in the real world. This means that, similarly to recent advancements in natural language modelling research, a universal predictor would need to be equipped with semantic knowledge of the world. This, along with the desire to amortise universal knowledge across use cases in a single model, points towards universal predictors belonging to the foundation model family. With the ambitious goal of creating such a universal predictor we motivate our work towards solutions that combine all of the aspects mentioned above: work with structured data, automatically adapt to new tasks and accumulate knowledge about the world through pre-training on vast amounts of data and tasks. This leads us to *Tabular Foundation Models* (TFM).

FUNDAMENTAL TABULAR PROCESS

The field of TFMs is relatively young, with the first mentions of this term only appearing about two years ago [40]. The underlying concepts, however, are much older. From a purely machine learning (ML) perspective, the current instantiations can be seen as *meta-learning* where observations are sets of dimension-named vectors, or simply *k-shot classification/regression* tasks [32, 33, 14, 17]. As a result, there are many frameworks that one can use to describe these objects, the right choice will allow us to draw connections across models of interest and avoid unnecessary confusion.

We choose to approach this problem from the perspective of stochastic processes, the language used in the Neural Processes paper [17], as it provided a modern day foundation to reason about latent decompositions of the meta-learning problem. The first step will be to expand the original decomposition and then motivate why this view is helpful in guiding research in the field of TFMs. Our main argument is twofold: on one hand it provides a unified language to talk about relations between existing methods in the literature and ways in which they could be combined, and on the other it provides a learning theory argument to why certain components can have a dramatic effect on the model quality.

Definition 1 (Fundamental Tabular Process (FTP)). *Given data context $C = \{(x_c^i, y_c^i)\}_i^N$ as well as a collection of facts $F = \{f_i\}_i^M$ and observations about the world $W = \{w\}_i^k$ we define a joint probability distribution*

$$p(y_P|x_P, C, F, W) = \iiint p(y_P|x_P, z_c)p(z_c|C, z_f)p(z_f|F, z_w)p(z_w|W)dz\ dz_f\ dz_w$$

where $p(y_P|x_P, z)$ is Kolmogorov-consistent, so that it is a correct stochastic process [30].

Despite this probabilistic definition, our goal is not to motivate or promote the need for a Bayesian approach here. Instead we resort to this decomposition of probabilities as a helpful tool to highlight the three critical components of an FTP:

1. **The Real World** Similar to other foundation models, a universal predictor contains knowledge about the world. In particular, this world is a single, consistent reality that imposes certain rules, structure, priors and meaning. We cannot directly observe the world as a whole and instead have access to a certain finite collection W of observations. An example of such observations could be the entire internet, which these days is often compressed into a Large Language Model (LLM), or into world models.
2. **Local Reality** Within this world, for any task there also exists a micro-world of a local reality specific to the task. For example, within a scientific domain this could be the constraints and specific characteristics of a particular laboratory. In an enterprise setting this would correspond to the information within and associated with a

■ FUNDAMENTAL TABULAR PROCESS

company. As with the real world it is impossible to directly observe this latent variable, but we have access to a collection of facts F about this micro-world. Quite crucially, some of the facts here might contradict the real-world knowledge. For example, 'Kraków' and 'Madrid' are two cities in the world, but within a company they could be the names of two meeting rooms. A universal predictor should be able to accommodate both realities and understand when to resort to which.

3. **Labelled examples** With these established, we get access to the context of the actual specific problem, represented as a collection of labelled observations C. Their interpretation and meaning now depends on the context provided by the aforementioned facts and world knowledge. An example of this could be a column within a table that contains distance information. This feature could be labelled with the word 'mile', which has a very specific meaning within the world we live in, that could further be affected by the fact that we might be solving a problem related to marine life (and thus it is in fact, a 'nautical mile').

Given all of the above, all potential future data points x_P can be successfully labelled.

From a purely representational perspective, a Neural Process [17] is enough to represent any functional mapping

$$p(y_P|x_P, C) = \int p(y_P|x_P, z)p(z|C)dz.$$

Why would one then introduce the two additional terms relating to W and F? The fact that they correspond to intuitive real-world objects is not enough justification to model them, we need a strict reason. The core argument we want to build is that introducing these quantities through focused research effort will lead to a significant *sample complexity reduction* in our tabular predictors. It is true that, in theory, we are able to learn any mapping defined by the context variable and without the extra terms if we have enough training data [5, 12]. However, the number of samples needed to reduce the error to an acceptable level might be prohibitively large. Especially in the real world, where the acquisition of new data can be arbitrarily expensive and sometimes even outright impossible due to a problem's constraints [34]. By acknowledging these two additional terms, we are identifying explicit opportunities to significantly reduce sample complexity. The following simple corollaries formalise this based on known results in statistical learning theory.

■ LEARNING TO LEARN

We begin with the simplest part, the motivation for learning $p(z_c|C, \cdot)$ through pre-training (or equivalently through learning to learn or meta-learning), as opposed to designing hand crafted algorithms like in classical machine learning. Most of the work in the current TFM landscape has been spent in this category [31, 20]. Conceptually, both designing an algorithm and pre-training on a massive set of problems accomplish the same thing. Both approaches boil down to creating some notion of a prior P over the function space, which should contain the function we want to model at inference. We know from standard PAC bounds that knowing a prior is directly related to our sample complexity, as formalised in the following corollary.

Corollary 1 (Benefits of Pre-training). *Let Q^* be a target distribution of functions and P be a prior. To learn a hypothesis with generalisation error ϵ and confidence $1 - \delta$, the sample complexity is bounded by*

$$\tilde{\mathcal{O}}\left(\frac{\mathrm{KL}(Q^*||P) + \log(1/\delta)}{\epsilon^2}\right)$$

which degenerates to

$$\tilde{\mathcal{O}}\left(\frac{-\log P(f^*) + \log(1/\delta)}{\epsilon^2}\right)$$

if Q^* is Dirac-delta around f^* (there is a single target function).

Proof. Direct consequence of bounds of Blumer [6] and McAllester [26] □

What this means in practice is that we can significantly reduce the sample complexity compared to classical ML algorithms. This can be achieved by setting up a pre-training regime such that the functions of practical interest occupy a meaningful part of the probability distribution. This contrasts with classical algorithms that are inherently designed to work uniformly across a wide range of hypotheses. Research in this space touches on many aspects of a model training pipeline:

- **Data generation** Training sets for pre-training tabular models are frequently synthetic, the generation of which boils down to manually defining the distribution P.

LEARNING TO LEARN

- **Data gathering** Real-world data play a crucial role both in training as well as in evaluation of models and data generators, as it provides a way to ground them closer to the desired real-world distribution Q^*. Careful selection of this data is therefore imperative.
- **Architectural design** The choice of architecture of the inference network itself imposes a prior over the functional space as it will make certain calculations easier or harder to express. In addition, explicitly imposing certain characteristics like invariances and equivariances when designing architectures will fold the space of functions and alter the distribution they will capture.

It is worth mentioning that current LLMs (the monolithic transformer-based models trained to carry out next-token prediction on a heterogeneous large corpus of human-written text) are likely unable to define a P that is well suited for our use case, since:

- The communication of the meta-task in LLMs is carried out in the form of a prompt. This approach inherently does not define a unique task for the model: the choice of wording of the prompt alone, for example, would alter the model's predictions. As a result, the support of the prior here would be unnecessarily spread.
- Certain mathematical properties of statistical learning theory are incompatible - e.g. the permutation invariance of columns in tables cannot be mathematically obtained without modifying the current architectures [1].

Consequently the P LLMs define will assign significantly lower probability to f^* than a model that explicitly respects this setup and consequently will have a worse sample complexity.

KNOWLEDGE OF THE WORLD

Another important component of the FTP is the term modelling the real world. We see it as a critical piece, since all the relevant problems we know of exist within this reality, and consequently share a lot of commonalities. This shared component and its potential for knowledge transfer can be formalised in many ways. We once again view this as another path to a significant *sample complexity reduction*. When pre-training a model to understand the concepts and relations of the real world, we can expect it to learn a new task on top much faster than if it had to learn it from scratch. More precisely, let us consider one of the most common applications in the tabular domain: the ability to reason about the actual values of categorical variables. In order to apply standard machine learning techniques we effectively need to treat categories as orthogonal vectors (e.g. through one-hot encoding, or using decision trees). However, in addition to being a different value to the other categoricals, text also carries meaning and this meaning might be shared between tasks. 'Kraków' will often represent a concept that is closer to 'Madrid' than to 'San Francisco', at least as long as the overall concept is related to geospatial reasoning. We can formalise this intuition by using standard techniques from learning theory and write down a following corollary:

Corollary 2 (Benefits of Categorical Embeddings). *Let $\mathcal{C} = C_1 \times \cdots \times C_k$ be a k-dimensional categorical metric space with $|C_i| = m_i$ and doubling dimensions $\Delta_1, \ldots, \Delta_k$. Assume the target function $f : \mathcal{C} \to \mathbb{R}$ is L-Lipschitz. To learn a hypothesis with generalisation error ϵ and confidence $1 - \delta$, the sample complexity when using the categoricals is bounded by*

$$\tilde{\mathcal{O}}\left(\frac{C_L + \log(1/\delta)}{\epsilon^2}\right) := \tilde{\mathcal{O}}\left(\frac{\prod_i^k (L/\epsilon)^{\Delta_i} + \log(1/\delta)}{\epsilon^2}\right),$$

while utilising one-hot encoding yields

$$\tilde{\mathcal{O}}\left(\frac{C_{\text{OH}} + \log(1/\delta)}{\epsilon^2}\right) := \tilde{\mathcal{O}}\left(\frac{\prod_i^k m_i + \log(1/\delta)}{\epsilon^2}\right).$$

***Proof** (sketch).* We use the general bound of sample complexity based on the metric entropy [2, 35]. Given the function is L-Lipschitz, the metric entropy scales linearly in the covering number of the domain itself [24]. Finally the covering dimension for the categorical (Lipschitz) case boils down to its doubling dimensions [18], while for one-hot covering dimension is just the volume of the space. □

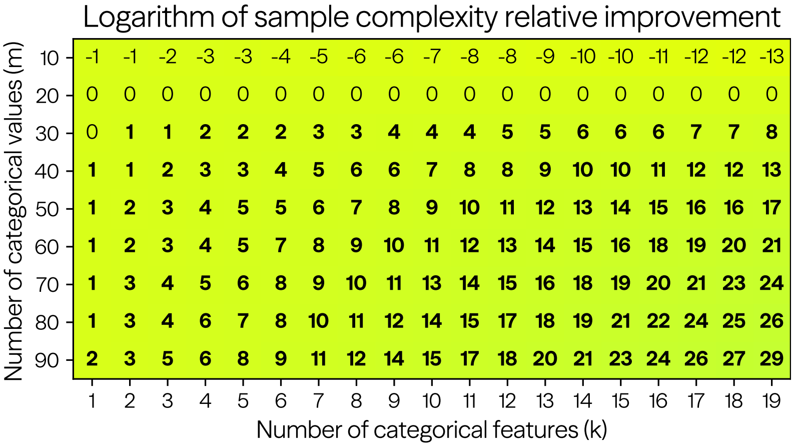
Using this formalisation, we infer that we can hope to obtain a significant reduction of the sample complexity under following assumptions:

- The number of categorical variables is large, both in terms of unique values (m_i) and/or number of categorical columns (k),
- the embedding space we learn during pre-training is very smooth (L is small) and is well behaved (has low dou-

■ bling dimension Δ_i).

KNOWLEDGE OF THE WORLD

While these conditions are quite intuitive, we find it helpful to consider a more concrete example to understand this difference in practice. For example, let's take dimension $k = 10$ and each categorical $m_i = 35$. This corresponds to a table containing 10 categorical columns with 35 distinct categorical values each, which falls comfortably within the usual parameters one would encounter in the real world. If we choose a very smooth metric $L = \Delta_i = 1$ and a 5% error $\epsilon = 0.05$ we obtain $C_{\mathrm{L}} = (1/0.05)^k = 2 \cdot 10^{10}$ compared to $C_{\mathrm{OH}} = 35^{10} \approx 2 \cdot 10^{15}$. This means that you can hope for over 100,000× lower sample complexity if you pretrained the categorical embeddings to respect the assumptions of **Corollary 2**. Now, if we consider a common case, where some categoricals are actually free-form text, this means that $m_i = n$, and we obtain $C_{\mathrm{OH}} = n^k$ which grows polynomially to infinity as the number of training samples n grows. **Figure 1** visualises how the sample complexity improves as the dimensions and number of categories grow. It is clear from that figure how quickly these bounds diverge, as the number of categorical values grows.



m \ k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
10	-1	-1	-2	-3	-3	-4	-5	-6	-6	-7	-8	-8	-9	-10	-10	-11	-12	-12	-13
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	**1**	**1**	**2**	**2**	**2**	**3**	**3**	**4**	**4**	**4**	**5**	**5**	**6**	**6**	**6**	**7**	**7**	**8**
40	**1**	**1**	**2**	**3**	**3**	**4**	**5**	**6**	**6**	**7**	**8**	**8**	**9**	**10**	**10**	**11**	**12**	**12**	**13**
50	**1**	**2**	**3**	**4**	**5**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**16**	**17**
60	**1**	**2**	**3**	**4**	**5**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**18**	**19**	**20**	**21**
70	**1**	**3**	**4**	**5**	**6**	**8**	**9**	**10**	**11**	**13**	**14**	**15**	**16**	**18**	**19**	**20**	**21**	**23**	**24**
80	**1**	**3**	**4**	**6**	**7**	**8**	**10**	**11**	**12**	**14**	**15**	**17**	**18**	**19**	**21**	**22**	**24**	**25**	**26**
90	**2**	**3**	**5**	**6**	**8**	**9**	**11**	**12**	**14**	**15**	**17**	**18**	**20**	**21**	**23**	**24**	**26**	**27**	**29**

Figure 1: A comparison of sample complexity improvement coming from **Corollary 2**. Once the ratio between number of unique categorical values to L/ϵ crosses 1 (which in this case happens at 20 and corresponds to any number in bold), the benefit grows extremely fast with both more categorical values and variables. And in the most extreme case reaches 10^{20} improvement. All numbers here are rough estimates, and the purpose of the visualisation is only to give a sense of the orders of magnitude, not the exact values.

The only way in which we can hope to obtain embeddings with the universal properties mentioned above is by training models beyond a single classification/regression problem. In addition, this model needs to capture a latent variable representing the real world and make use of that knowledge towards its predictions. As discussed, this model does not only capture consistent objects and concepts, but also the relationships between them. Lipschitzness served as a convenient exemplification of the types of relations we referred to in our arguments. Of course there are other possible regularities, for which one could derive analogous bounds.

■ LOCAL REALITY

While it is undeniable that all humanly relevant problems exist within a single, shared reality, not every instance of a concept should be interpreted identically. Even the motivating example above - of city names being present as categorical variables will on rare occasions be completely wrong. What if the task of interest in fact concerns predictions within a company where 'Kraków' and 'Madrid' are not cities, but meeting room names? This type of information will never be available to a model developer. Crucially in this *local reality* the underlying geometry is completely different, and as a result the L-Lipschitz assumption of **Corollary 2** will not hold. This is the main argument towards needing the $p(z_f|F, z_w)$ component. It is not enough to just learn a single consistent model of everything, especially since a *local reality* can change arbitrarily fast – companies can rename their meeting rooms at any point in time. Consequently, the focus of this additional component is on learning the mapping and on learning how to unify the model of the world z_w with any additional facts that might overwrite parts of it. From a mathematical perspective, the argument reiterates **Corollary 2**.

LOCAL REALITY

Observation 1 (Benefits of Local Knowledge). *Let $\mathcal{C} = C_1 \times \cdots \times C_k$ be a k-dimensional categorical metric space with $|C_i| = m_i$ and let $\phi_F : \mathcal{C} \to \mathcal{C}_{\mathcal{F}}$ be a mapping to another k-dimensional categorical metric space with doubling dimensions $\Delta_1, \ldots, \Delta_k$. Assume the target function $f : \mathcal{C} \to \mathbb{R}$ can be expressed as $f = g \circ \phi_F$, and that g is L-Lipschitz. To learn a hypothesis with generalisation error ϵ and confidence $1 - \delta$, the sample complexity when using the projected space is bounded by*

$$\tilde{\mathcal{O}}\left(\frac{\prod_i (L/\epsilon)^{\Delta_i} + \log(1/\delta)}{\epsilon^2}\right),$$

while in the worst case scenario (i.e. when the metric on $\mathcal{C}$ degenerates the space to just orthogonal unit vectors) using the original space gives

$$\tilde{\mathcal{O}}\left(\frac{\prod_i m_i + \log(1/\delta)}{\epsilon^2}\right).$$

Proof. A direct consequence of **Corollary 2**. □

This motivates further sample complexity reduction when modelling $p(z_f|F, z_w)$. A big qualitative difference here is that while learning $p(z_w|W)$ can boil down to training an LLM which embeds a consistent view of the world in its weights, for example, training $p(z_f|F, z_w)$ has to learn a transformation to be applied during inference. This results from the fact that different collections of F are contradictory, preventing us from marginalising over different tasks. Consequently, F exists only in the context of inference on a particular task

SAMPLE COMPLEXITY

Given the above structure and arguments, the main message crystallises:

> *By modelling each of the components of the Fundamental Tabular Process, one can hope to drastically reduce the sample complexity of novel supervised learning tasks, far beyond what any classical ML algorithm, or a current-day LLM, can achieve.*

The main goal of our reasoning so far is not to provide a narrow, overly formalised specification, but rather to motivate on an intuitive and a mathematical level, the decomposition of a universal predictor. By tying everything to the concept of sample complexity reduction we are able to think about all of the ongoing exciting research avenues in this space as striving towards the same goal, rather than a collection of competing entities defining their own objectives. We are able to group papers, methods and results into clusters that share coherent mathematical objectives. Of course, we do not claim this view is optimal, it is simply a helpful framework.

EXISTING LANDSCAPE

Each of the three terms of an FTP also allows us to characterise different approaches in this space based on whether they are present or not. In addition, if a model approximates a term we can further distinguish between those that approximate it implicitly, i.e. by conditioning on information related to this term, or explicitly by encoding that information in their weights or as a latent variable.
Let us look at each term of FTPs. We won't go deep into $p_y := p(y_P|x_P, z_c)$ since it can be thought of as an "inference" step. It is, however, worth mentioning that most of the architectural differences between existing TFMs can be seen as different implementations of this term. The term $p(z_c|C, z_f)$ describes the task of distilling the information provided by the entirety of the context (this includes information from C, F and W) into a representation that allows us to make good predictions on the task specified by said context. In the case of traditional machine learning methods, z_c corresponds to the learned parameters (e.g. weights of a neural network) which result from training on a context C (the labelled examples). In the case of Neural Processes this step constitutes the summarisation of the context observations into a fixed dimensional latent space, which can be then queried to make new predictions. As per our earlier definition we refer to this as explicit modelling of p_c. Implicit modelling of p_c on the other hand is what many Tabular Foundation Models resort to. They do so by directly comparing the predictions to the context data (e.g. by passing them through a transformer model) thereby collapsing the terms for p_y and p_c.
Most approaches do not go beyond modelling the probability of the target variables given some task-defining context. There are, however, a few attempts at incorporating some world knowledge W as well as some user-specific knowledge F into model predictions. Some notable examples include Kumo [15] that implicitly models $p(z_f|F, z_w)$ by using message passing neural networks over graphs spanning multiple tables or databases. Given that it is designed to work specifically with relational databases, the index/key-based relations can be thought of as additional context that provides facts about every data point and helps the model with interpreting its meanings. Others like CARTE [23] model $p(z_w|W)$ explicitly, by pre-training on knowledge graphs.

Considering the results reported in the papers above, it is apparent that the biggest gains can be achieved by pushing the frontier of incorporating $p(z_f|F, z_w)$ and $p(z_w|W)$. This can seem somewhat contradictory to what is currently a big focus of the community: improving statistical reasoning, the parts of the model that approximate p_y and p_c. The main reason behind this is the lack of really strong baselines to evaluate our ability to model p_f and p_w. When it comes to pure statistical reasoning decades worth of work have resulted in strong baselines. Algorithms like CatBoost and XGBoost form a strong frontier here. When it comes to moving beyond the standard statistical learning theory and incorporating knowledge of the world, however, classical methods are not applicable, and thus we are left with very naive approaches as our only point of comparison.

The research trajectory of TFMs reflects a common pattern in machine learning which is reminiscent of the early days of DL image processing. Initial research overwhelmingly focuses on architectural breakthroughs (in the past this produced convolutional networks, skip connections for image recognition or transformers for natural language). Soon after, the focus shifts to optimising the data and training regimes instead. This match in trends also serves as an indication that the research field of TFMs is still in an early stage. As it stands the two big priorities in TFM architecture research are maximising the scale of the context window (being able to consume large quantities of data during inference) and inducing various invariances and equivariances in transformer-based models [19], which, as mentioned before, can be thought of as a way to sharpen the P distribution (see **Corollary 1**). Models that specifically tackle the latter, like TabICL, are nonetheless not truly feature order invariant yet [31]. These two challenges alone are proof that there is still a lot of work to be done in this field. In terms of architectural choices for incorporating facts and world knowledge, graph neural networks [23, 15] and LLMs seem to be the most common ones. This heterogeneity does not seem to stem from any grounded, theoretical justification, but rather simple practicalities - it is much easier to reuse pre-existing models like LLMs, than to train a unified, homogeneous model that is able to jointly model the world and to carry out statistical reasoning from scratch. That being said, one of the biggest lessons in deep learning is that unified, end-to-end trained systems always beat human designed hybrids and multi-stage systems, which points to an extensive, underexplored research direction.

Class	**PT**	**TST**	**Examples**	p_c	p_f	p_w
Traditional models		⬢	XGBoost [8]	■		
		⬢	CatBoost [11]	■		
		⬢	Single-Task Neural Network	■		
Zero-shot models	⬢$_S$	⬡	TabPFN [20]	□		
	⬢$_S$	⬡	TabICL [31]	□		
	⬢$_S$	⬡	Kumo [15]	□	□	
Fine-tuned models	⬢$_W$	⬢	CARTE [23]	□		■
	⬢$_W$	⬢	TabStar [3]	□		■
	⬢$_W$	⬢	XTab [41]	□		
Task-specific models	⬡$_W$	⬢	TabNet [4]	■		
	⬡$_W$	⬢	RealMLP [21]	■		

Table 1: Characterisation of the tabular prediction model landscape. PT and TST stand for pre-training and task-specific training respectively and refer to the training phases that a model can be trained with. ⬢ indicates that this training phase is necessary for a model, ⬡ that it is optional. p_c, p_f, and p_w correspond to the probabilities from Definition 1 with $p_c = p(z_c|C, z_f)$, $p_w = p(z_w|W)$ and $p_f = p(z_f|F, z_w)$, respectively. ■ signals that the probability is being modelled explicitly while □ stands for implicit modelling. $\cdot_S$ and $\cdot_W$ stands for synthetic or real-world respectively.

There are two conceptually distinct phases that can be present when training TFMs: pre-training (PT) and task-specific training (TST). In the context of LLMs the latter would be referred to as fine-tuning. Here we choose to adopt the term TST instead, given that for some of the models this stage carries more purpose than just adapting to a specific task. In some cases, in fact, it constitutes the entire training regime. Let us take a look at 4 main classes of approaches:

1. **Traditional models** for tabular data, such as CatBoost and XGBoost, are trained exclusively in the TST phase because models in this category are built from scratch for every new table or task. This also includes all the incredible neural network based models that took years to develop for a specific application, such as AlphaFold [36].
2. **Zero-shot models** amortise the few-shot adaptation requirements of meta learning. This means the models learn to adapt to task-specific data at inference time without any additional training steps. The overarching family of this type of amortised meta-learning models is called Conditional Neural Processes [16]. Instances of this family applied to tabular prediction include different flavours of PFNs [20, 31].

EXISTING LANDSCAPE

3. **Fine-tuned models** rely on an initial pre-training phase, which can either be carried out on the same type of data as the TST phase (TabStar, XTab) or on a different one (CARTE). This pre-training leverages the adaptability of amortised meta-learning models while the follow-up fine-tuning stage allows the model to improve its predictions and further strengthen its adaptation to the task at hand. Needless to say, given that this group of models relies on two training stages they are generally the most computationally expensive. Nonetheless, since these models benefit from the upsides of both training phases it is likely that a universal predictor will fall in this category.

4. **Task-specific models** The training regime of this group of models is similar to the traditional models in that the majority of the training happens in the task-specific stage. Unlike traditional models, however, approaches like TabNet and RealMLP can share information between tasks as the hyperparameters are tuned jointly for all tasks in what could be considered a pre-training phase. Nonetheless each model is trained *from scratch* for every task.

Finally, the type of data used to pre-train models has become another major differentiating factor. Models that only rely on pre-training require large quantities of data to learn flexible models, especially since they can't rely on additional training at inference time to improve their performance. Tabular data for predictive tasks, however, is not as readily available as text or image data. As a result, most pre-trained models rely on synthetic data as it can be generated in any quantity. Despite the challenge that matching generated data to real data poses, models trained on synthetic data have performed extremely well on real-world data, often outperforming approaches trained on real data. The downside of synthetic data is that we can only create algorithms that produce numerical values. It is not possible to generate meaningful textual or categorical data as we are not able to capture its statistics in the same way.

WHERE DO LLMS LIVE?

At this point it could be natural to wonder about the role of LLMs in all of this. Do they belong into this picture and categorisation? We argue that they violate too many assumptions to currently be a viable candidate as a stand-alone model. However, they might constitute a valuable component of a larger system. As such, work on Tabular Foundation Models cannot be reduced to LLM research. It is about building complex statistical models, which might incorporate LLMs as components within its stack. Nonetheless such models would constitute a profound evolutions from the original LLM format.
There are many reasons for this [13], one of them is their inability to scale their context window to the sizes common in tabular data. **Figure 2** clearly depicts the severity of this issue. For each table on Kaggle we calculate the minimum number of tokens required to represent it. We then plot the frequency over all the number of tokens required for all the tables in Kaggle. Within the resulting curve we have highlighted the portion of tables that the average and a top-tier LLM could take as input respectively. We have also marked a few sizes of interest: the datasets that lead to four breakthroughs (AlphaGo [36], AlphaFold [22], AlphaStar [39] and GPT3 [7]) as well as the size of a table with 10M rows and 100 columns, which constitutes a relatively common scale for corporate data in the real world. The fact that the strongest LLMs are currently not able to process a large portion of the Kaggle tables *individually* and that the majority of recent breakthroughs had data sizes more than 2 orders of magnitude above what LLMs can handle clearly shows that this is not a problem that can just be thrown at language models as they are. This is not to say LLMs cannot be part of the solution. As mentioned above they have the potential to bring in knowledge about the world as part of a larger model.

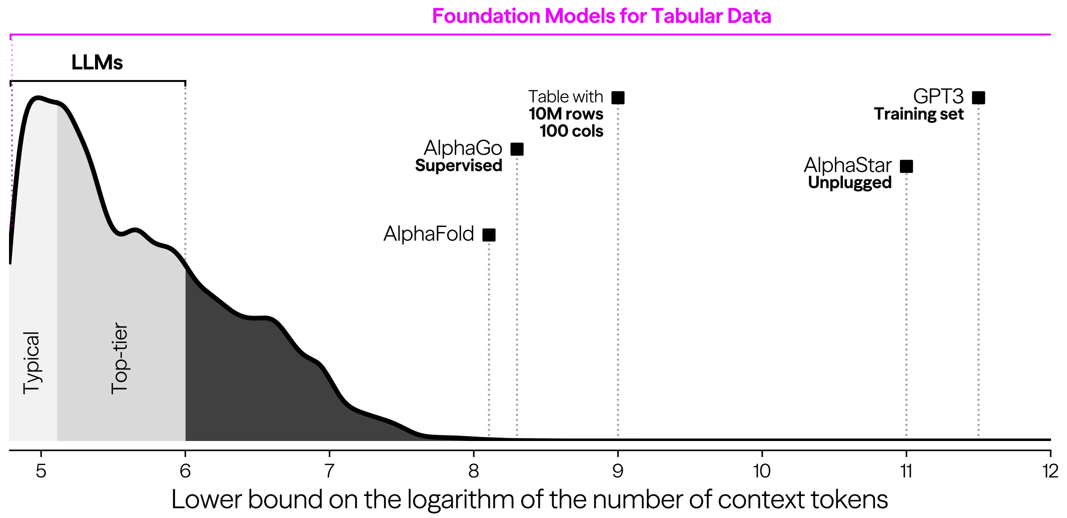


Figure 2: Visualisation shows (in log scale) the distribution of the minimum number of tokens required to process a *single* table from the Kaggle repository (limited to datasets up to 100MB). We have included call-outs for big breakthroughs of

■ WHERE DO LLMS LIVE?

supervised-learning models. With the current LLMs one could fit at best up to 10^6 tokens. Showing a dramatic discrepancy between the scale of the data of interest, and capacities of (even the most powerful) LLMs.

Another crucial point to keep in mind is that even if they were scaled significantly in terms of their context window, the core argument we are trying to form here is that the game of building the universal predictor is about the *sample complexity*. General, language based models, while powerful, are inherently misaligned in terms of their training objective, of forming a sharp prior over the functions that would serve this purpose (see **Corollary 1**). This does not mean that one cannot use LLM-like models, but rather that the general path taken by current research in the LLM space is misaligned with this objective. In practice the distinction of what is and is not an LLM becomes philosophical, and our only argument is that creating a proper solution requires more insights and developments than just scale.

■ AN EXCITING CHALLENGE

It is an extremely rare opportunity in the world of machine learning to be at the forefront of a nascent field which is nearly guaranteed to work. Tabular Foundation Models, as of today, satisfy this requirement. This is a very young field, with a lot of exciting, open questions, spanning from theoretical considerations (especially around various forms of invariances and equivariances), through tough engineering ones (e.g. how does one scale to billions of in-context points) to open-ended practical scientific endeavours (what are the best training regimes to model each of the components? How do we avoid the risks of repeating 'The Bitter Lesson' [38]). At the same time the number of positive results, and the maturity of related areas proves that the challenge is ripe for the taking. In addition, the community working on these challenges is refreshingly small compared to the extremely oversaturated world of Large Language Models.

In a not too distant future, we will be able to make superhuman predictions, and obtain scientific and economic breakthroughs using universal predictors and we will use them as commonly and easily as every enterprise uses tree based methods today.

References

[1] Rohan Alur, Chris Hays, Manish Raghavan, and Devavrat Shah. The impossibility of inverse permutation learning in transformer models, 2025. URL `https://arxiv.org/abs/2509.24125`.

[2] Martin Anthony and Peter L Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, 1999.

[3] Alan Arazi, Eilam Shapira, and Roi Reichart. Tabstar: A tabular foundation model for tabular data with text fields. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[4] Sercan O Arik and Thomas Pfister. Tabnet: Attentive interpretable tabular learning. *arXiv preprint arXiv:1908.07442*, 2019.

[5] Andrew R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14①: 115–133, 1994. doi: 10.1007/BF00993164. URL `https://doi.org/10.1007/BF00993164`.

[6] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information Processing Letters*, 24⑥:377–380, 1987.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794. Association for Computing Machinery, 2016.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS)*, New York, NY, USA, 2016. ACM. URL `https://arxiv.org/abs/1606.07792`. arXiv:1606.07792.

[10] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, pages 191–198, New York, NY, USA, 2016. ACM. doi: 10.1145/2959100.2959190.

[11] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[12] Selina Drews and Michael Kohler. On the universal consistency of an over-parametrized deep neural network estimate learned by gradient descent. *Annals of the Institute of Statistical Mathematics*, 76③:361–391, 2024. doi: 10.1007/s10463-024-00898-6. URL `https://doi.org/10.1007/s10463-024-00898-6`.

[13] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. Large language models (llms) on tabular data: Prediction, generation, and understanding–a survey. *arXiv preprint arXiv:2402.17944*, 2024.

[14] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28④:594–611, 2006.

[15] Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. Kumorfm: A foundation model for in-context learning on relational data. 2025.

[16] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International conference on machine learning*, pages 1704–1713. PMLR, 2018.

[17] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[18] Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient regression in metric spaces via approximate lipschitz continuity. *IEEE Transactions on Information Theory*, 60⑨:5790–5799, 2014.

[19] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations 2023*, 2023.

[20] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025.

[21] David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37:26577–26658, 2024.

[22] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[23] Myung Jun Kim, Léo Grinsztajn, and Gaël Varoquaux. Carte: pretraining and transfer for tabular learning. *arXiv preprint arXiv:2402.16785*, 2024.

[24] Andrei N Kolmogorov and Vladimir M Tikhomirov. ϵ-entropy and ϵ-capacity of sets in function spaces. *Uspekhi Matematicheskikh Nauk*, 14②:3–86, 1959. English translation in American Mathematical Society Translations (Series 2), Vol. 17, pp. 277–364 (1961).

[25] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

[26] David A McAllester. Some pac-bayesian theorems. *Machine Learning*, 37③:355–363, 1999.

[27] Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023. doi: 10.1038/s41586-023-06735-9.

[28] Meta Engineering. Sequence learning: A new paradigm for personalized ads and recommendations. `https://engineering.fb.com/`, 2024. Meta Engineering Blog, accessed 2025-11-28.

[29] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep learning recommendation model for personalization and recommendation systems. *CoRR*, abs/1906.00091, 2019. URL `https://arxiv.org/abs/1906.00091`.

[30] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Universitext. Springer, Berlin, 6 edition, 2003.

[31] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

[32] Jürgen Schmidhuber. Evolutionary principles in self-referential learning. Diploma thesis, Technische Universität München, Munich, Germany, 1987.

[33] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4①:131–139, 1992.

[34] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[35] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, Cambridge, 2014.

[36] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[37] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackerman, et al. A deep learning approach to antibiotic discovery. *Cell*, 180④:688–702, 2020. doi: 10.1016/j.cell.2020.01.021.

[38] Richard S. Sutton. The bitter lesson. 3 2019. URL `http://incompleteideas.net/IncIdeas/BitterLesson.html`. Accessed: 2025-11-28.

[39] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

[40] Zifeng Wang, Chufan Gao, Cao Xiao, and Jimeng Sun. Anypredict: Foundation model for tabular prediction. *CoRR*, abs/2305.12081, 2023. doi: 10.48550/arXiv.2305.12081. URL `https://arxiv.org/abs/2305.12081`.

[41] Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.