Hybrid Sparse-Aware Chained Regressor with Neural Residuals: The Kap Formula

Guy Kaptue

September 17, 2025

Abstract

We introduce a novel chained regression architecture for tabular prediction that rigorously preserves sparsity throughout preprocessing and model chaining, while using dense representations only at neural residual stages. This method iteratively augments features with model outputs, creating a generalized stacked ensemble that is dimension-safe, sparse-aware, and highly suitable for high-cardinality categorical data. We formalize this sequential process as the **Kap Formula**, providing a precise mathematical operator to describe the architecture's flow. This paper provides the formal problem setup, a detailed training and inference procedure, and proofs of dimensional consistency, theoretical expressivity, and computational properties. We discuss the model's distinct advantages, particularly in handling high-dimensional, sparse tabular data, and its limitations.

1. Introduction

Predictive modeling on high-dimensional tabular datasets with mixed feature types presents unique challenges, including sparsity, high-cardinality categorical variables, and complex nonlinear dependencies. While traditional regressors such as tree-based models excel at handling structured data, they can struggle with intricate feature interactions. Conversely, deep neural networks, which are powerful universal approximators, require dense inputs and often incur significant computational and memory overhead when dealing with sparse, high-dimensional data.

In this work, we propose the ***Hybrid Sparse-Aware Chained Regressor with Neural Residuals***, a novel architecture that addresses these challenges by combining the strengths of different model families. The core idea is to sequentially chain base regressors, appending their predictions as new features at each stage. To capture complex, nonlinear patterns, neural modules are optionally inserted between stages to learn from the residual errors of the preceding model on an augmented feature space. A key innovation is a *sparsity-preserving* feature augmentation strategy that minimizes memory consumption.

Our main contributions are:

- A precise mathematical formulation of a sequential chained regression framework with sparsity-aware preprocessing and feature augmentation.
- The formal definition of the **Kap Formula**, a new mathematical operator that describes this sequential stacked generalization process.
- Proofs of dimensional safety, theoretical expressivity, and a detailed analysis of the model's computational complexity.

- A comprehensive discussion of the model's applicability, advantages, and limitations compared to existing ensemble and deep learning methods.

2. Related Work

Our proposed architecture represents a novel synthesis of several established machine learning concepts.

- **Stacked Generalization:** The core chaining mechanism extends Wolpert's stacked generalization [4]. Unlike classical stacking, which trains a meta-learner on the parallel outputs of base learners, our method constructs meta-features iteratively, using the output of each stage to enrich the input for the next.
- **Boosting:** The use of neural networks to model the output of a previous regressor is reminiscent of gradient boosting machines [2]. In our model, the neural residual stage serves a similar purpose to a weak learner in boosting, capturing patterns in the target signal that the preceding model failed to explain. However, it operates on an augmented feature space rather than a direct gradient step.
- **Hybrid Architectures:** The model falls within the category of hybrid machine learning architectures that combine heterogeneous learners. Similar to works like TabNet [1], we integrate a deep neural network component. Our specific approach, however, is uniquely defined by its rigorous focus on sparsity preservation through a specialized 'safe hstack hybrid' function, allowing it to scale to high-cardinality categorical features without a massive memory overhead.

3. Methodology

3.1. Problem Setup and Preprocessing

Let a dataset be given as $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We partition the feature space into numeric, categorical, and datetime columns. We define a preprocessing operator Φ that produces a design matrix suitable for learning, preserving sparsity where beneficial.

Numeric features: Let $\mathcal{X}_{\text{num}} \subseteq \{1, \ldots, d\}$ be the set of indices for numeric features.

$$\phi_{\text{num}}(\mathbf{x}) = \text{Standardize}(\mathbf{x}_{\mathcal{X}_{\text{num}}}) \in \mathbb{R}^{d_{\text{num}}}.$$

Categorical features: Let $\mathcal{X}_{\text{cat}} \subseteq \{1, \ldots, d\}$ be the set of indices for categorical features.

$$\phi_{\text{cat}}(\mathbf{x}) = \text{OneHot}(\mathbf{x}_{\mathcal{X}_{\text{cat}}}) \in \{0, 1\}^{d_{\text{cat}}},$$

where OneHot encoding maps each categorical value to a sparse binary vector.

Datetime features: Let $\mathcal{X}_{\text{time}} \subseteq \{1, \ldots, d\}$ be the set of indices for datetime features.

$$\phi_{\text{time}}(\mathbf{x}) = \text{Calendar}(\mathbf{x}_{\mathcal{X}_{\text{time}}}) \in \mathbb{R}^{d_{\text{time}}},$$

where the Calendar operator extracts fixed features such as year, month, and day.

Composite feature map: The overall feature transformation is a concatenation of the preprocessed subsets.

$$\Phi(\mathbf{x}) = \big[\phi_{\text{num}}(\mathbf{x}) \parallel \phi_{\text{cat}}(\mathbf{x}) \parallel \phi_{\text{time}}(\mathbf{x})\big] \in \mathbb{R}^D,$$

where ϕ_{cat} is stored sparsely. Let $X^{(0)} = \Phi(X)$ denote the preprocessed design matrix for the entire dataset, of shape $n \times D$.

3.2. Model Architecture and the Kap Formula

We define a chain of K base regressors $\{R_k\}_{k=1}^{K}$ and up to $K-1$ neural residual modules $\{N_k\}_{k=1}^{K-1}$. At each stage $k \in \{1, \ldots, K\}$, the feature matrix $X^{(k-1)}$ is augmented with the predictions from the previous model. The process starts with the preprocessed data $X^{(0)}$.

Stage features: Let $X^{(k)} \in \mathbb{R}^{n \times D_k}$ be the feature matrix at the beginning of stage k, where $D_0 = D$. The dimension of the next stage is $D_{k+1} = D_k + 1$.

Base regressor at stage k: A regressor R_k is trained on the features $X^{(k-1)}$ and produces a prediction vector $\hat{\mathbf{y}}_k^R \in \mathbb{R}^n$.

$$\hat{\mathbf{y}}_k^R = R_k\big(X^{(k-1)}\big).$$

Neural residual (optional) at stage k: A neural network module N_k is trained on the augmented features from stage $k-1$ and the predictions of the base regressor R_k.

$$\hat{\mathbf{y}}_k^N = N_k\big(\text{dense}(X^{(k-1)} \parallel \hat{\mathbf{y}}_k^R)\big) \in \mathbb{R}^n.$$

Feature augmentation rule: The feature matrix for the next stage is created by horizontally stacking the current feature matrix and the latest prediction vector. The choice of prediction vector depends on whether a neural residual is used.

$$X^{(k+1)} = \begin{cases} X^{(k)} \parallel \hat{\mathbf{y}}_k^R & \text{if } k = K, \text{ or no neural residual is used after } R_k, \\ X^{(k)} \parallel \hat{\mathbf{y}}_k^N & \text{otherwise.} \end{cases}$$

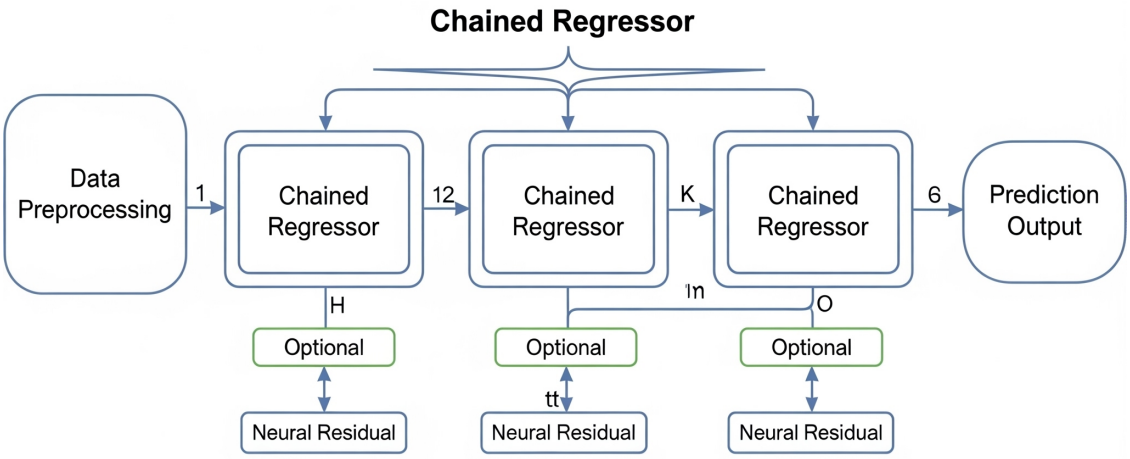

Figure 1: Overall workflow of the Hybrid Sparse-Aware Chained Regressor. The model sequentially processes data through chained regressors, with optional neural residual modules at each stage, leveraging predictions from previous stages as augmented features.

Theorem 1 (Kap Formula, 2025)**.** Let $\{R_k\}_{k=1}^{K}$ be a chain of base regressors and $\{N_k\}_{k=1}^{K-1}$ be a set of optional neural refiners. The sequential feature augmentation process is defined by:

$$X^{(0)} = \Phi(X), \quad X^{(k)} = X^{(k-1)} \parallel \begin{cases} R_k(X^{(k-1)}), & \text{if stage } k \text{ has no neural residual,} \\ N_k(\text{dense}(X^{(k-1)} \| R_k(X^{(k-1)}))), & \text{otherwise.} \end{cases}$$

The final prediction is then given by the output of the last regressor:

$$\hat{\mathbf{y}} = R_K(X^{(K-1)}).$$

This operator, denoted $\mathbf{Kap}(x; \{R_k, N_k\})$, formally defines a sequential stacked generalization process.

The Kap Formula distinguishes itself from classical stacking by explicitly defining the iterative construction of meta-features, which allows for the progressive refinement of predictions across stages.

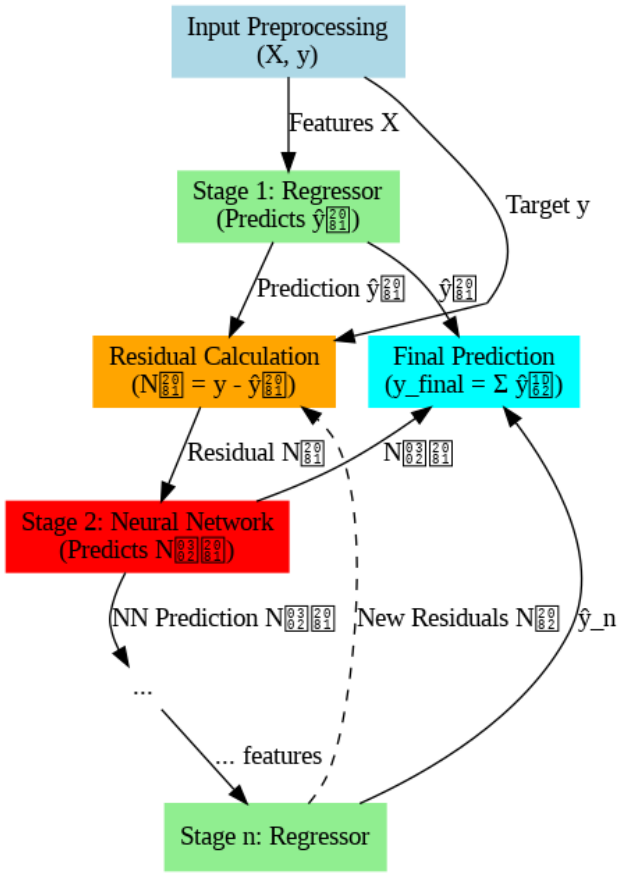

Figure 2: Conceptual workflow of the Kap Formula: A sequential chained regressor. The model progresses through stages, with each new stage leveraging predictions from the previous one as an augmented feature.

3.3. Training Objectives and Procedure

We assume that base regressors are trained using a generic empirical risk minimization framework, while neural modules are optimized with mean-squared error (MSE).

Step 1. Initialize $X^{(0)} = \Phi(X)$.

Step 2. For $k = 1$ to K:

- Fit R_k on $X^{(k-1)}$ and compute $\hat{\mathbf{y}}_k^R$.
- If $k < K$ and a neural residual is enabled:
  - Form dense augmented features $Z^{(k)} = \text{dense}(X^{(k-1)} \parallel \hat{\mathbf{y}}_k^R)$.
  - Fit N_k on $Z^{(k)}$ by minimizing MSE and compute $\hat{\mathbf{y}}_k^N$.
  - Update $X^{(k)} = X^{(k-1)} \parallel \hat{\mathbf{y}}_k^N$ using sparse-aware concatenation.
- Else:
  - Update $X^{(k)} = X^{(k-1)} \parallel \hat{\mathbf{y}}_k^R$.

Step 3. Output the final prediction $\hat{\mathbf{y}} = R_K(X^{(K-1)})$.

4. Inference and Dimensional Consistency

4.1. Inference Procedure

Given a new batch of data X_{new}, the inference procedure mirrors the training process.

Step 1. Compute $X_{\text{new}}^{(0)} = \Phi(X_{\text{new}})$.
Step 2. For $k = 1$ to K:

- Evaluate $\hat{\mathbf{y}}_{k,\text{new}}^{R} = R_k(X_{\text{new}}^{(k-1)})$.
- If stage k has a neural residual:
  - Form $Z_{\text{new}}^{(k)} = \text{dense}(X_{\text{new}}^{(k-1)} \parallel \hat{\mathbf{y}}_{k,\text{new}}^{R})$.
  - Evaluate $\hat{\mathbf{y}}_{k,\text{new}}^{N} = N_k(Z_{\text{new}}^{(k)})$.
  - Set $X_{\text{new}}^{(k)} = X_{\text{new}}^{(k-1)} \parallel \hat{\mathbf{y}}_{k,\text{new}}^{N}$.
- Else:
  - Set $X_{\text{new}}^{(k)} = X_{\text{new}}^{(k-1)} \parallel \hat{\mathbf{y}}_{k,\text{new}}^{R}$.

Step 3. Return the last stage's output, $\hat{\mathbf{y}}_{\text{new}} = \hat{\mathbf{y}}_{K,\text{new}}^{R}$.

4.2. Dimensional Consistency

Proposition 1 (Dimension Safety)**.** Let $X^{(k-1)} \in \mathbb{R}^{n \times D_{k-1}}$ be either dense or sparse. Suppose the prediction vector $\hat{\mathbf{y}} \in \mathbb{R}^n$ is reshaped to $\mathbb{R}^{n \times 1}$ prior to concatenation. Then the update $X^{(k)} = X^{(k-1)} \parallel \hat{\mathbf{y}}$ is well-defined and yields $X^{(k)} \in \mathbb{R}^{n \times D_{k-1}+1}$, provided the concatenation operator is chosen appropriately based on the operands' storage formats.

Proof of Proposition. By construction, both operands share the same row dimension n. The prediction vector is explicitly reshaped to a single column, ensuring compatible second dimensions for horizontal concatenation. The choice of concatenation operator (*sparse_hstack* or *np.hstack*) preserves the row dimension and increases the column dimension by exactly one, thus proving the operation is always defined and produces the stated shape. □

5. Properties and Theoretical Guarantees

Expressivity: If any neural module N_k is a universal approximator on compact subsets of its input space (e.g., a multi-layer perceptron with a non-linear activation function), as per the work of Hornik et al. [3], then the overall Kap Formula-based chain inherits universal approximation capacity on the augmented feature space. This provides a strong theoretical foundation for its predictive power.

Computational Complexity: Let $T(R_k; n, D_k)$ be the training time of a base learner and $T(N_k; n, D_k)$ for the neural module. The total training time is the sum of the times for each stage:

$$\sum_{k=1}^{K} T(R_k; n, D_{k-1}) \; + \; \sum_{k=1}^{K-1} T(N_k; n, D_{k-1} + 1).$$

Memory usage is optimized by preserving sparsity for all $X^{(k-1)}$ except within the transiently created dense matrix $Z^{(k-1)}$ at neural stages.

6. Advantages and Limitations

6.1. Advantages

- **Enhanced Predictive Performance:** The sequential chaining allows each subsequent regressor to learn from and correct the errors of the previous ones on an enriched feature space. The neural residuals are particularly effective at capturing complex, nonlinear relationships.
- **Sparsity Preservation:** The model efficiently handles high-dimensional, sparse data, such as that from one-hot encoding of high-cardinality categorical features. By preserving sparsity throughout most of the pipeline, it reduces memory consumption and leverages the performance of sparse matrix operations.
- **Flexibility and Customization:** The framework allows for a high degree of customization. Users can select any combination of base regressors (R_k), order them based on domain knowledge, and decide where to insert the neural residual stages.

6.2. Limitations

- **Increased Computational Cost:** The sequential nature of the pipeline means that each regressor and neural network must be trained one after another. This makes the overall training time significantly longer than a single-model approach.
- **Complexity and Hyperparameter Tuning:** The model introduces a new layer of complexity. In addition to tuning each base regressor, users must also tune the hyperparameters of the neural networks and decide on the optimal sequence of models.
- **Interpretability Challenges:** While individual base regressors may offer some interpretability, the overall chained pipeline is a black-box model. The final prediction is the result of a complex series of transformations and model interactions, making it difficult to understand which features or stages are driving the final output.

7. Applicability Fields

7.1. When the Model is Advantageous

- **High-dimensional Tabular Data:** The model is ideal for datasets with many categorical features, such as those found in e-commerce, finance, and bioinformatics. The sparsity preservation minimizes memory overhead and computational cost.
- **Complex Nonlinear Relationships:** When a single model cannot adequately capture the intricate, nonlinear patterns, the chained approach with neural residuals can be highly effective.
- **High-Stakes Prediction Tasks:** In applications like fraud detection or financial forecasting where a small improvement in accuracy is critical, the potential for superior performance justifies the increased cost and complexity.

7.2. When the Model Should Be Avoided

- **Simple, Low-Dimensional Data:** For simple datasets, the complexity and overhead of the chained model are unnecessary. A single, well-tuned model would likely suffice.
- **Real-Time or Low-Latency Applications:** The multi-stage, sequential nature of the pipeline makes inference significantly slower. Applications requiring very fast predictions would be better served by a simpler, more efficient model.
- **When Interpretability is Critical:** In fields where model decisions need to be explainable and auditable (e.g., regulatory compliance), the black-box nature of the full pipeline is a major drawback.

8. Conclusion

We have presented the ***Hybrid Sparse-Aware Chained Regressor with Neural Residuals***, a novel and robust architecture for tabular regression. Our key contribution is the ***Kap Formula***, a new operator that formalizes a sequential stacked generalization process. Unlike traditional ensembles that train meta-learners in parallel, our method builds an enriched feature space iteratively, with each stage learning from the predictions of the previous. This approach is uniquely designed to handle high-dimensional, sparse tabular data, a common but challenging data type in real-world applications.

By preserving sparsity throughout the model pipeline, we address a critical bottleneck of applying standard neural networks to wide tabular datasets, significantly reducing memory consumption and leveraging the efficiency of sparse matrix operations. The optional neural residual stages provide the model with the ability to capture complex, nonlinear relationships that might be missed by simpler base regressors. This hybrid design combines the interpretability and efficiency of traditional models with the powerful approximation capabilities of deep learning.

In summary, the Kap Formula represents a powerful, generalized framework for building scalable and high-performing models on mixed-type tabular data. Its core strengths lie in its ***sparsity-aware design***, ***iterative feature augmentation***, and ***hybrid learning approach***. Future work will focus on extensive empirical benchmarking against state-of-the-art models like TabNet and Gradient Boosting Machines, as well as developing automated methods for optimizing the sequence and type of base regressors and neural residuals within the chain.

References

[1] Sercan O. Arik and Ansgar Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[2] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1230, 2001.

[3] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[4] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.