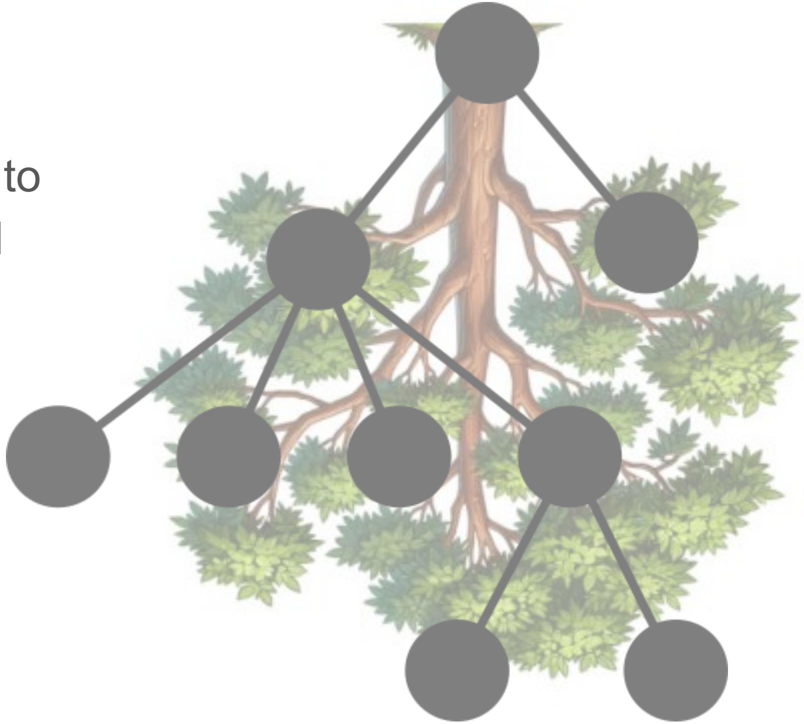
Neural Architecture and Hyperparameter Optimization of Time Series Forecasting Models with Tree Parzen Estimators

Andrew Garcia, PhD; Marco Vega, PhD

@ XLII ENCUENTRO DE ECONOMISTAS
DEL BANCO CENTRAL DE RESERVA DEL PERÚ
October 21, 2024

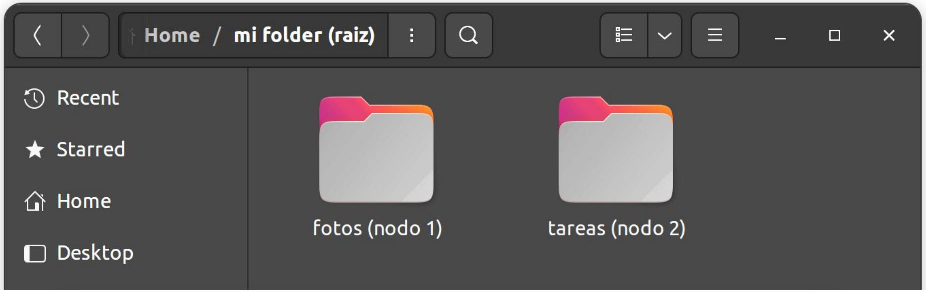
Trees in Computing

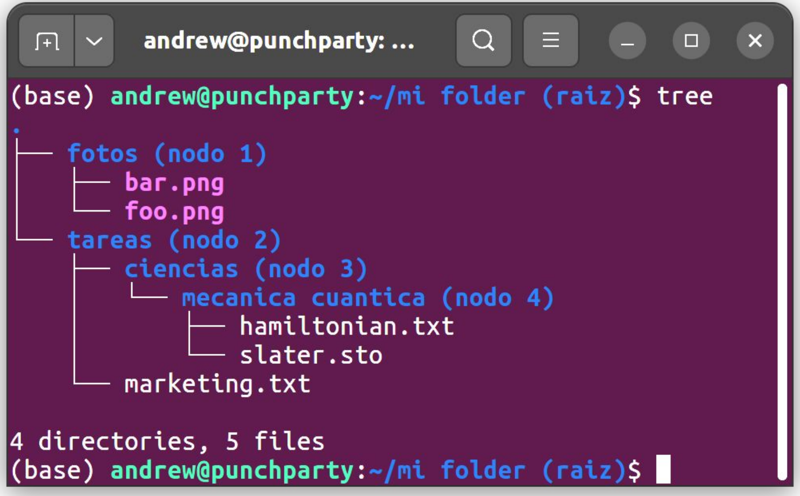
- **What is a Tree?**
    - **A Root:** The starting point of the tree.
    - **Branches:** Lines that connect the root to other points, like the branches of a real tree.
    - **Nodes:** The points or "endpoints" connected by branches. They can connect to other nodes.

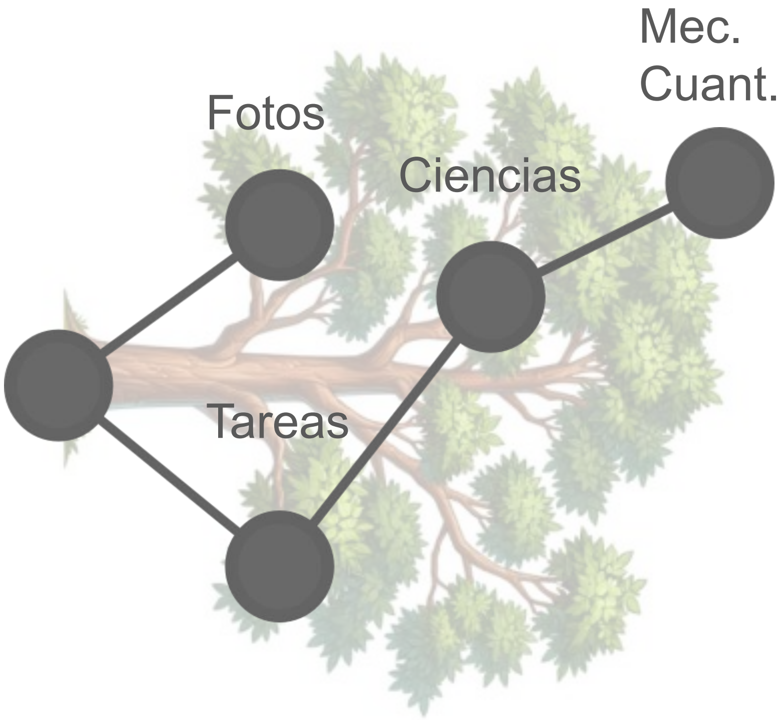


Trees in Computing

Folder Structure

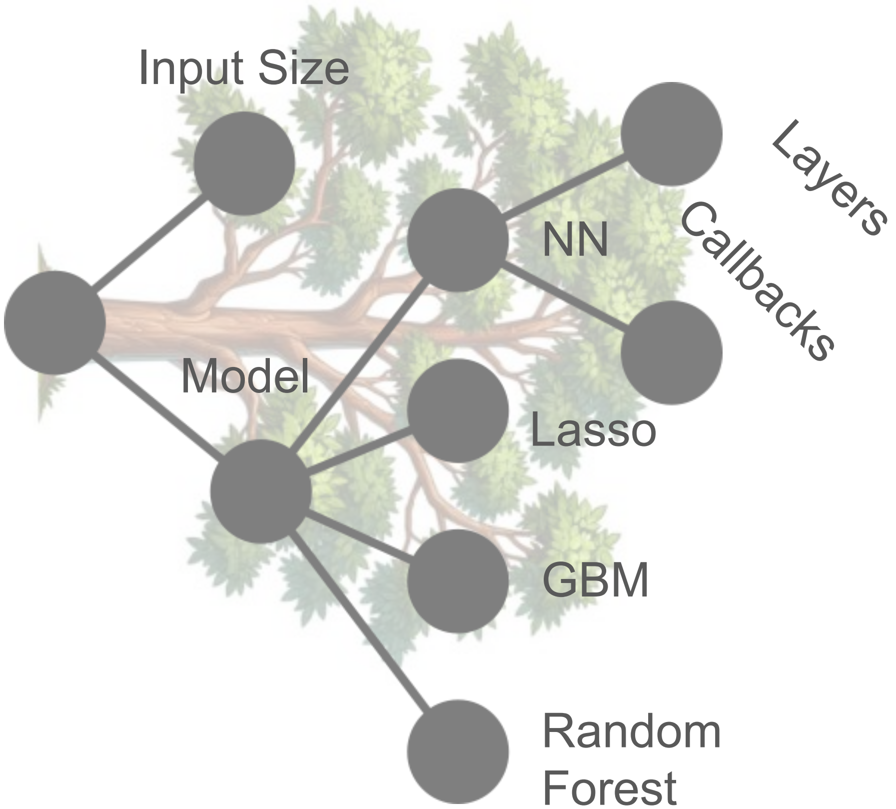






Trees in Computing

Hyperparameter Search Space




```
search_space {
    input size: range(10,50)
    model : CHOOSE [
     { Random Forest (RF),
       RF depth : range(10,30)
       RF ests : range (10,1000)
     }
     . . .
     { Neural Network (NN),
        Layers : CHOOSE [
        { Layer A, . . . }
        . . .
        { Layer Z, . . . }
     ]
     } ]
}
```

Why Tree Structured?
Hyperparameter Search Space

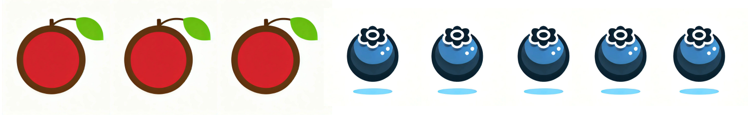
- Reduces the search space to **meaningful** models.
- **Why look for berries 🫐 in an apple 🍎 tree?**
  - The choice of a particular parameter can determine the relevance of another (dependency)

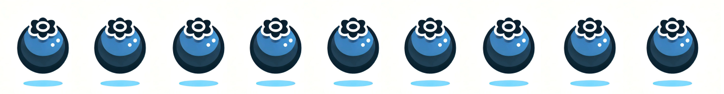
Why Tree Structured?

Hyperparameter Search Space

- Reduces the search space to **meaningful** models.
- **Why look for berries 🫐 in an apple 🍎 tree?**
  - The choice of a particular parameter can determine the relevance of another (dependency)
- Less number of trials to reach an optimum

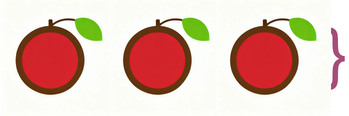
apple tree (linear) {





}

$$\mathcal{O}(k^n)$$

apple tree (tree) {  }

$$\mathcal{O}(k^d)$$

k:= choices per parameter
n:= number of parameters
d:= depth of search tree

(1) ML Regression Models

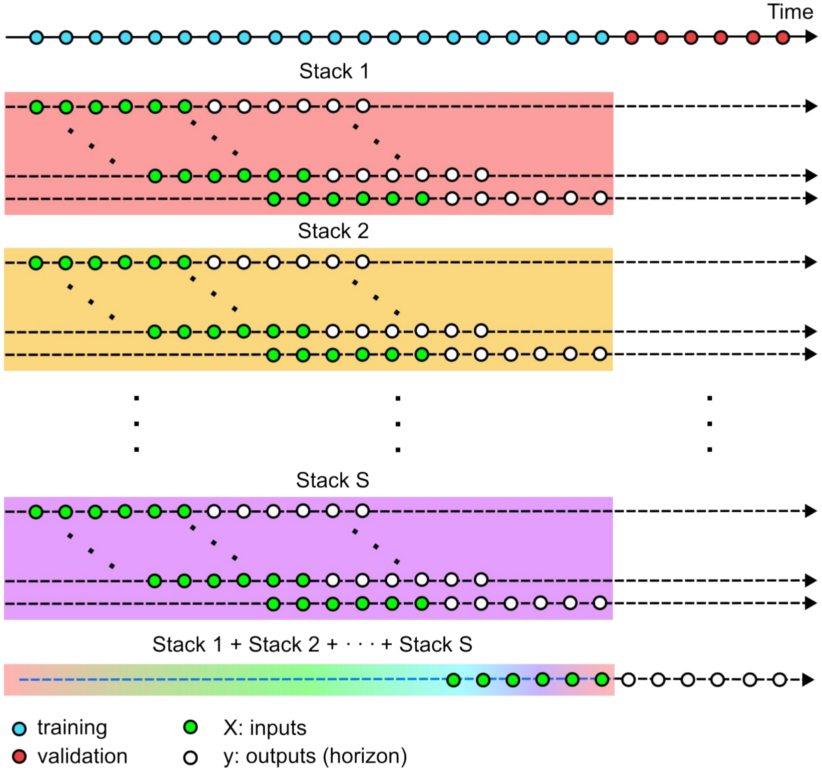
Hyperparameter	Feature value search space
Input Size	DiscreteRange (3, 108, 3)
Model*	{Random Forest, Gradient Boosting, Lasso}
Random Forest**	
Number of Estimators	DiscreteRange (10, 1000, 10)
Max Depth	{None, 10, 20, 30, 40, 50}
Min Samples Split	DiscreteRange (2, 20, 1)
Min Samples Leaf	DiscreteRange (1, 10, 1)
Max Features	{None, sqrt, log2}
Bootstrap	{True, False}
Gradient Boosting**	
Number of Estimators	DiscreteRange (10, 1000, 10)
Learning Rate	LogRange (-5, 0)
Max Depth	{None, 3, 5, 7, 9}
Min Samples Split	DiscreteRange (2, 10, 1)
Min Samples Leaf	DiscreteRange (1, 5, 1)
Subsample	Range (0.5, 1.0)
Lasso**	
Alpha	LogRange (-7, 2)

- Tree structured on the model type.
  - Each model is a separate node
  - If chosen, its hyperparameters may be chosen within the specified ranges
- Optimizes for **the best ML regression model**

Source: [Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)]

(2) NeuralForecast Models

Hyperparameter	Feature value search space
Input-to-Output Size	Range (1.0, 3.5)
Input Size (Input-to-Output Size)†	DiscreteRange (60, 210)
Max Steps	DiscreteRange (50, 200, 10)
Model*	{NHITS, NBEATS}
NBEATS**	
Number of Polynomials	{2, 3, 4}
Number of Harmonics	{1, 2}



- Suite of NBEATS and NHITS models
- Using **neuralforecast** package.
    - Source code left untouched
    - Handles data processing internally through BaseWindows

Source: Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)

(3) Neural Architecture Search (NAS) Models

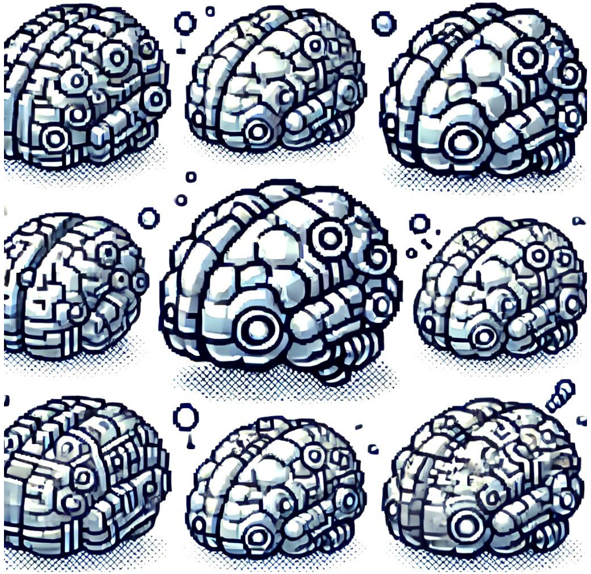
- **Topology**: Connectivity. What it takes to build a neural network
  - Layers
  - Network Components
- **Functional**: How a neural network should learn
  - Optimizers
  - Learning Rates
  - Callbacks

Hyperparameter	Detail	Feature value search space
	Global	
Random Seed for Initialization		DiscreteRange(1,100)
Adam Optimizer	Learning Rate	LogRange(-4, -2)
	β_1	Range(0.85, 0.95)
	β_2	Range(0.995, 0.999)
EarlyStopping Callback	Patience	DiscreteRange(20, 100, 5)
	Restore Best Weights	True
ReduceLROnPlateau Callback	Factor	0.5
	Patience	DiscreteRange(5, 20, 5)
	Min LR	0.0001
	Network-Specific	
Batch Size		DiscreteRange(1,10)
Input Size		DiscreteRange(3,40)
Activation Function		{None, ReLU, LeakyReLU, Swish}
L2 Layer Weight Regularization*		{False, True}
Neural Network Topology*		$\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$
Non-Local Blocks, Pre-MLP*		{False, True}
Non-Local Blocks, Post-MLP*		{False, True}
Time2Vec*		{False, True}
Scalers		{False, True}

Source: [Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)]

Machine Learning (ML)

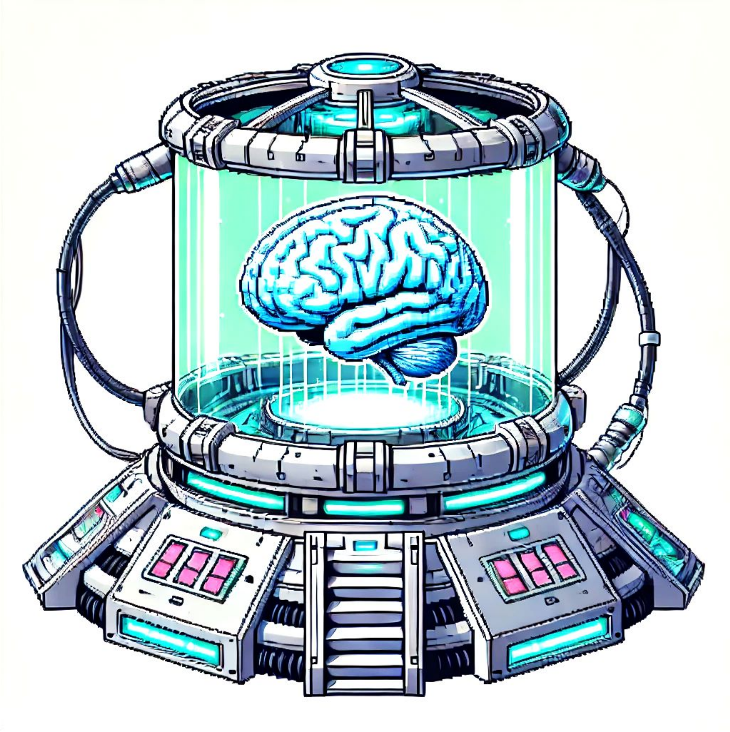
- Computers learn from data to make predictions or decisions
  - Use models based on
    - Mathematics
    - Statistics
- Typically, ML **models are pre-determined by humans** (should I use linear regression, or a neural network with 5 layers and ReLU functions?)



Automated Machine Learning (AutoML)

- A computer's build-your-own-ML feat.
- Here, we **humans build the process to automate** the selection and construction of the **best ML model** to solve the problem.

(should I include GBM and linear regression in the candidates to allow the computer to choose?)



TPE-Guided Hyperparameter Optimization

Can be Generalized to a Bi-Level Optimization Problem

Let a model == a set of λ hyperparameters

$$\hat{\lambda} = \arg\min_{\lambda} \mathcal{L}^{\text{val}}(\mathcal{D}; \hat{\theta}, \lambda)$$

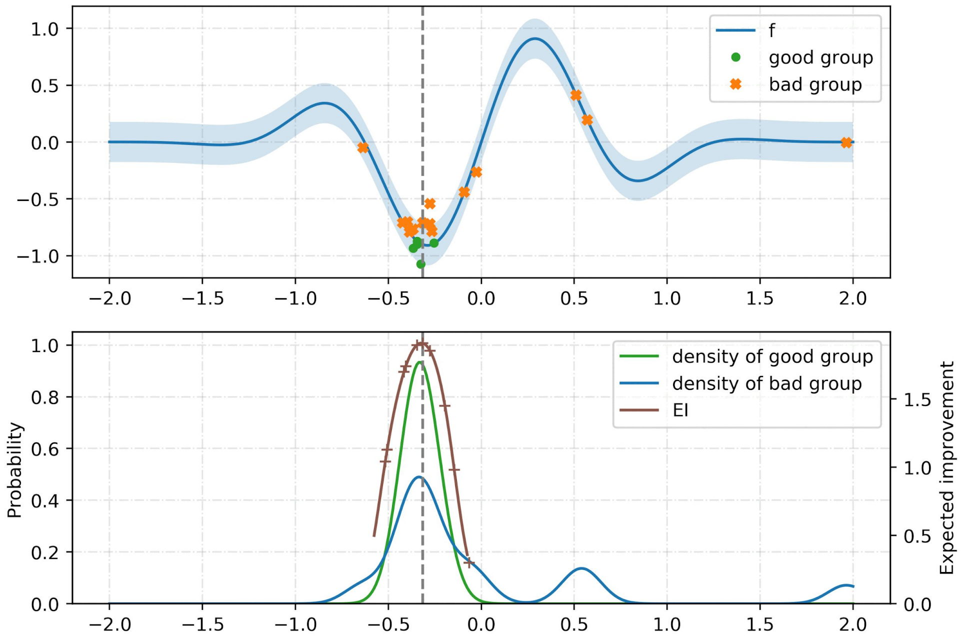
Minimize the forecasting loss wrt all models

$$\text{s.t.:} \quad \hat{\theta} = \arg\min_{\theta} \mathcal{L}^{\text{train}}(\mathcal{D}; \theta, \lambda)$$

Train a ML model

Source: [Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)]

The Method
Tree-structured Parzen Estimators (TPE)



$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases}$$

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy$$

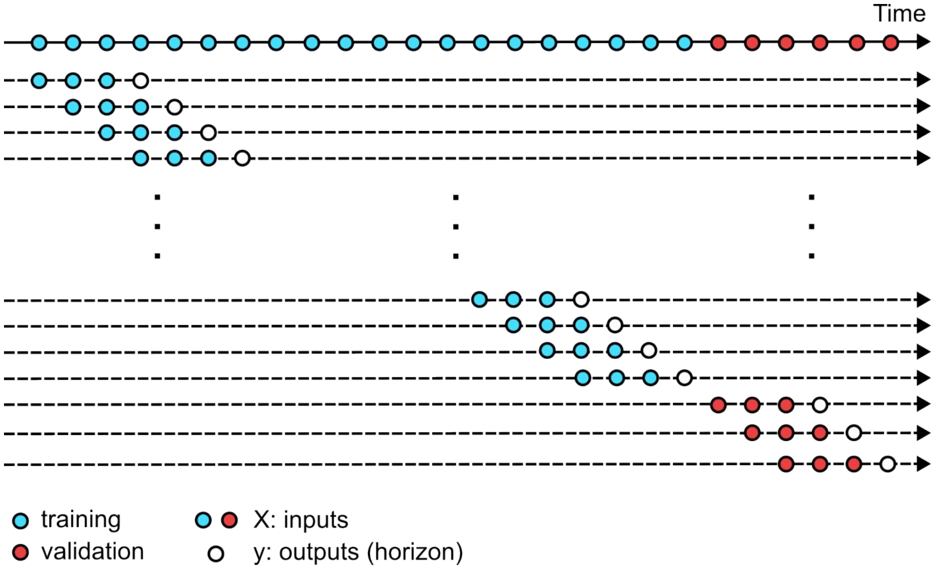
$$\propto \left(\gamma + \frac{g(x)}{\ell(x)}(1-\gamma)\right)^{-1}$$

Source: [Hyper-parameter optimization algorithms: a short review]

Forecasting
Data Preparation for Model Training

$$\hat{\lambda} = \arg\min_{\lambda} \mathcal{L}^{\text{val}}(\mathcal{D}; \hat{\theta}, \lambda)$$
$$\text{s.t.:} \quad \hat{\theta} = \arg\min_{\theta} \mathcal{L}^{\text{train}}(\mathcal{D}; \theta, \lambda)$$

- Split the time series into a training and a validation set.
- Each set is then further split into X-y samples
  - X: autoregressive inputs
  - y: horizon output
- Model search optimized by validation set.

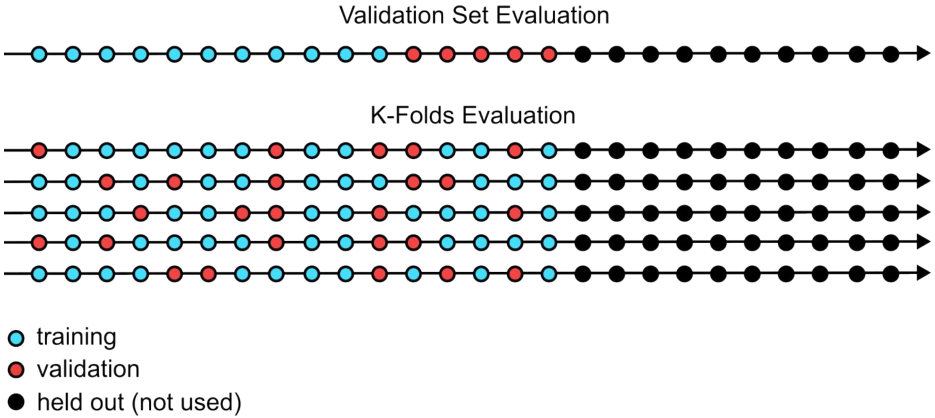


Source: [Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)]

Forecasting
Data Preparation for Model Training

$$\hat{\lambda} = \arg\min_{\lambda} \mathcal{L}^{\text{val}}(\mathcal{D}; \hat{\theta}, \lambda)$$
$$\text{s.t.:} \quad \hat{\theta} = \arg\min_{\theta} \mathcal{L}^{\text{train}}(\mathcal{D}; \theta, \lambda)$$

- Is there a better manifestation of validation for ML forecasting?
- Aside from the former strategy, we also consider K-Fold sets for validation.



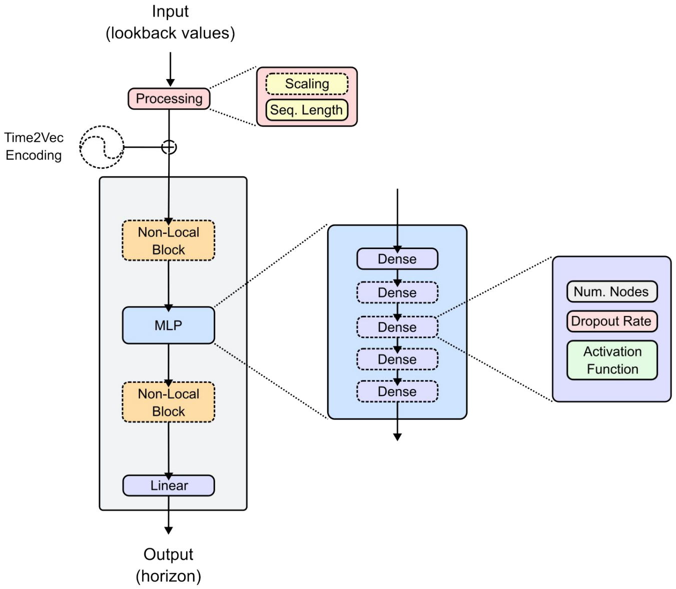
Source: [Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)]

The Models

- Three model collections are evaluated in isolation between each other through this methodology
  - ML Regression Model Collection
    - Model Selection
  - NeuralForecast Model Collection
    - Model Selection
  - Neural Network Topology and Functional Hyperparameters
    - Neural Architecture Search (NAS)

Source: [Algorithms for Hyper-Parameter Optimization]

Neural Network Topology and Functional Hyperparameters (NAS) - Let's Go Deeper



Detail	Feature value search space
MLP Layer Features	
Activation Function	{None, ReLU, LeakyReLU, Swish}
L2 Layer Weight Regularization*	{False, True}
L2 Strength**	LogRange(-5, -1)
Neural Network Topology*	$\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$
For each Layer $i \in \{1, \ldots, j\}$, for each τ_j Neural Network Topology	
Number of Nodes in Layer i**	DiscreteRange(1, 50)
Non-Local Blocks	
Non-Local Blocks, Pre-MLP*	{False, True}
Compression Rate**	Range(0, 1)
Non-Local Blocks, Post-MLP*	{False, True}
Compression Rate**	Range(0, 1)
Time2Vec (t2v) Layer	
Time2Vec Layer*	{False, True}
Number of Trend Terms**	1
Number of Periodic Terms**	DiscreteRange(4, 100, 2)

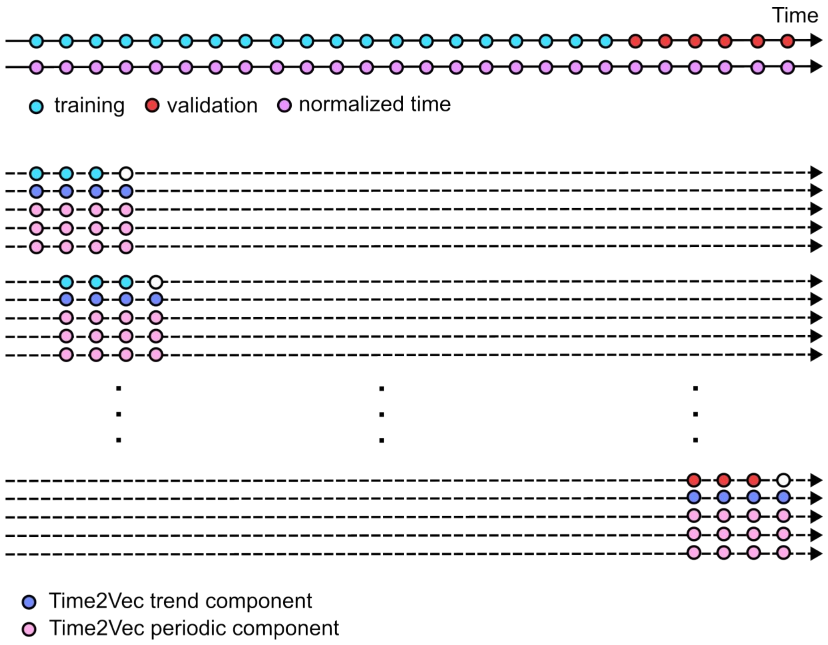
* Hyperparameters with an asterisk have additional hyperparameters.
** Hyperparameters or sets with double asterisks are the children of said hyperparameters or children sets thereof.

Source: [Forecasting inflation with a framework for model and neural architecture search with tree-structured search spaces (Garcia and Vega 2024)]

Time2Vec

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

- When Time2Vec is selected, the **time series is tagged with** a normalized **time** vector **prior** to any splitting.
  - A record of the relative location of each sample
- As it passes through the network, normalized time vector is encoded into **Time2Vec components** with **weights to be learned** by the NN.



Source: [Time2Vec](#) GitHub: [time2vec](#) Demo: [Colab](#)

Results

Forecasting Performance

- For all trainings, the validation window ends in January 2019
- Two POOS testing windows are reserved
  - Jan 2019 to Dec 2023 (60 months)
  - Jan 2022 to Dec 2023 (24 months)

ML Methods	Total CPI Annual Var Dec-23	Out-of-Sample RMSE Jan-22 to Dec-23	Out-of-Sample RMSE Jan-19 to Dec-23
Single Task (Univariate) Models - Π_t			
NeuralForecast, Validation Vector	2.857	0.393	0.393
ML Regressor, Validation Set	3.126	0.442	0.401
ML Regressor, K-fold	3.045	0.423	0.391
NAS-UV, Validation Set	3.093	0.457	0.418
NAS-UV, K-fold	3.128	0.420	0.382
NAS-UV (shuffled), Validation Set	3.167	0.497	0.425
NAS-UV (shuffled), K-fold	2.951	0.545	0.413
Dual Task (Component-Based) Models - Π_t^{sae} and Π_t^{ae}			
NeuralForecast, Validation Vector	2.921	0.374	0.369
ML Regressors, Validation Set	2.807	0.478	0.414
ML Regressors, K-fold	2.964	0.478	0.397
NAS-CB, Validation Set	3.055	0.409	0.370
NAS-CB, K-fold	3.061	0.454	0.380
NAS-CB (shuffled), Validation Set	3.060	0.406	0.358
NAS-CB (shuffled), K-fold	3.038	0.428	0.379
Expected Value (Benchmark)			
Ground-Truth	3.237	-	-
AR(2)	3.405	0.483	0.450
AR(1)	3.638	0.559	0.470
Random Walk	3.638	0.559	0.463

NAS = “neural architecture search”, UV = “univariate model”, CB = “component-based model”.

Forecasting Performance

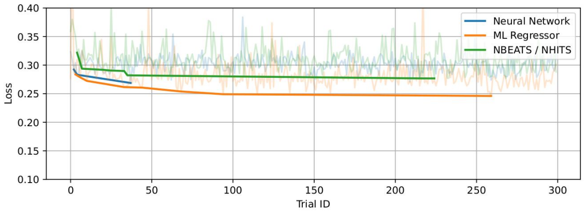
- NeuralForecast models overall the best performing.
- For univariate (single task) case
  - Models optimized through K-fold validation better than standard.
- For component-based (dual task) case
  - NAS outperforms ML regressors
  - Standard validation > K-fold

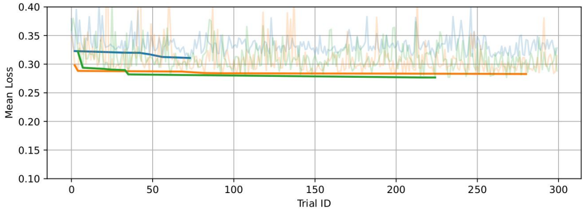
ML Methods	Total CPI Annual Var Dec-23	Out-of-Sample RMSE Jan-22 to Dec-23	Jan-19 to Dec-23
Single Task (Univariate) Models - Π_t			
NeuralForecast, Validation Vector	2.857	0.393	0.393
ML Regressor, Validation Set	3.126	0.442	0.401
ML Regressor, K-fold	3.045	0.423	0.391
NAS-UV, Validation Set	3.093	0.457	0.418
NAS-UV, K-fold	3.128	0.420	0.382
NAS-UV (shuffled), Validation Set	3.167	0.497	0.425
NAS-UV (shuffled), K-fold	2.951	0.545	0.413
Dual Task (Component-Based) Models - Π_t^{sae} and Π_t^{ae}			
NeuralForecast, Validation Vector	2.921	0.374	0.369
ML Regressors, Validation Set	2.807	0.478	0.414
ML Regressors, K-fold	2.964	0.478	0.397
NAS-CB, Validation Set	3.055	0.409	0.370
NAS-CB, K-fold	3.061	0.454	0.380
NAS-CB (shuffled), Validation Set	3.060	0.406	0.358
NAS-CB (shuffled), K-fold	3.038	0.428	0.379
Expected Value (Benchmark)			
Ground-Truth	3.237	-	-
AR(2)	3.405	0.483	0.450
AR(1)	3.638	0.559	0.470
Random Walk	3.638	0.559	0.463

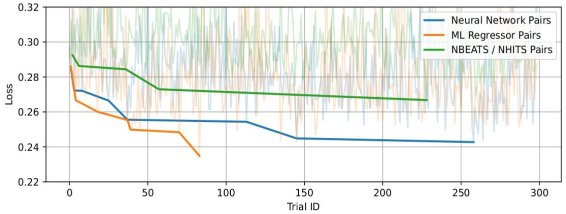
NAS = "neural architecture search", UV = "univariate model", CB = "component-based model".

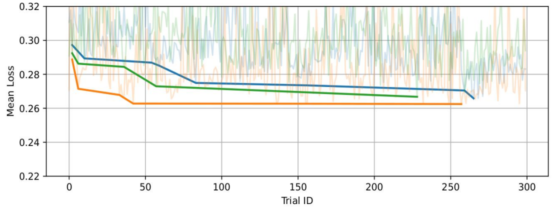
Shapley Values Study with SHAP

- We have accumulated large amounts of models
  - 1,800 models for NAS
  - 1,800 models for ML
  - 900 models for NBEATS / NHITS
- Use their **hyperparameter sets** and corresponding **forecasting errors** for a Shapley values study.









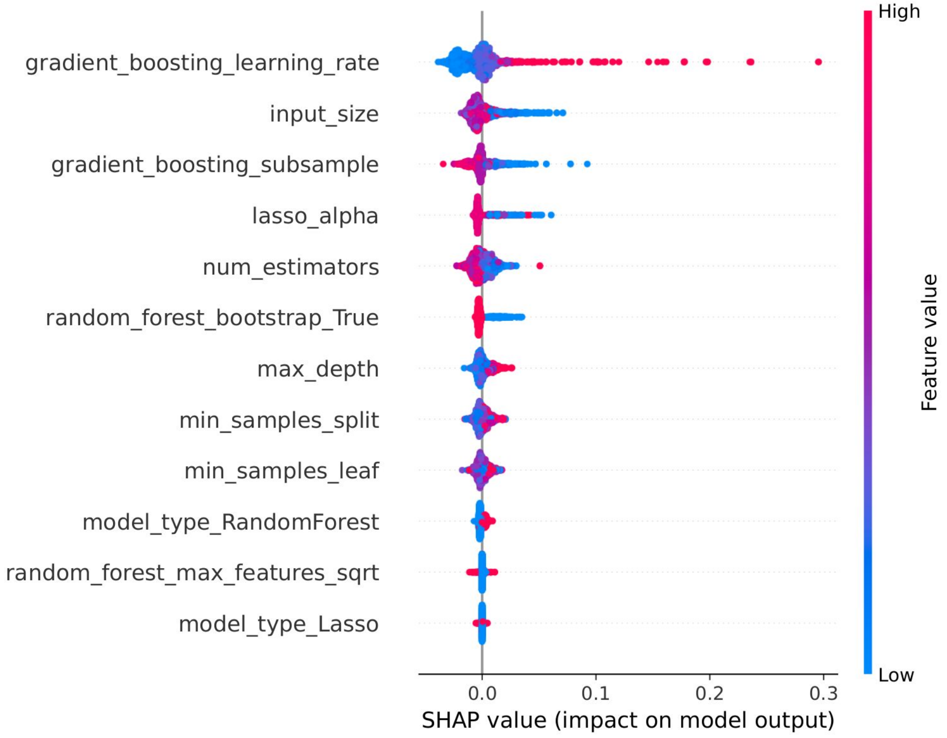
SHAP Study - ML Regressors

- ML Regressors dominate in performance for the univariate case.
- GBM models dominate as best over the other options.

B.3 Models Optimized with Testing Set Evaluation

Table 8: Univariate Machine Learning Regressor

Hyperparameter	Feature value
Input Size	57
Model Type	Gradient Boosting
Max Depth	9
Number of Estimators	790
Minimum Samples Split	10
Minimum Samples Leaf	5
Learning Rate	0.02165
Subsample	0.97788
Performance	
$\mathcal{L}^{val}(\mathcal{D};\Theta,\Lambda)$	0.24601

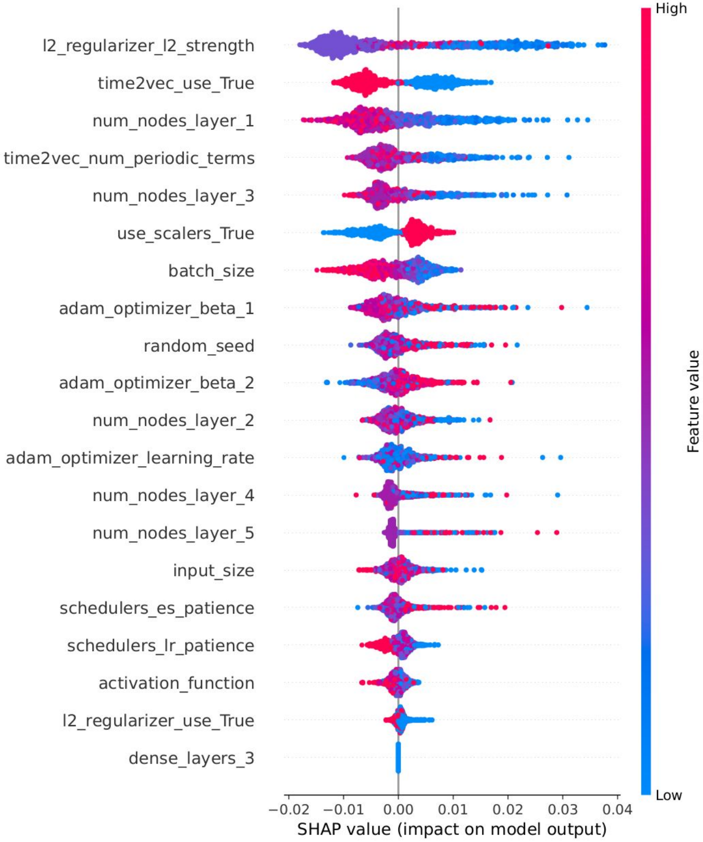


SHAP Study - NAS

- Best NAS model to forecast inflation is a component-based model.
- Both SAE and AE components have Time2Vec operations.
    - SAE modeled by a NN
    - AE by a non-NN model

Table 11: Component-Based Neural Architecture Search (NAS-CB) model

Hyperparameter	Feature value	
	SAE Model	AE Model
Adam Learning Rate	0.00291	
Adam β_1	0.87802	
Adam β_2	0.99731	
ES Patience	55	
LR Patience	10	
Input Size	18	
Batch Size	5	10
Number of Dense Layers	3	3
Number of Layer Nodes	[36,13,16]	[29,12,25]
Activation Function	ReLU	Linear (None)
Pre-MLP NLB Compression L_{pool}	9	-
Scalers, Use	False	False
Number of Periodic Terms (Time2Vec)	48	40
Performance		
$\mathcal{L}^{val}(\mathcal{D};\Theta,\Lambda)$	0.24272	
$\mathcal{L}^{val,sae}(\mathcal{D}^{(1)};\Theta^{(1)},\Lambda^{(1)})$	0.13550	
$\mathcal{L}^{val,ae}(\mathcal{D}^{(2)};\Theta^{(2)},\Lambda^{(2)})$	0.50156	

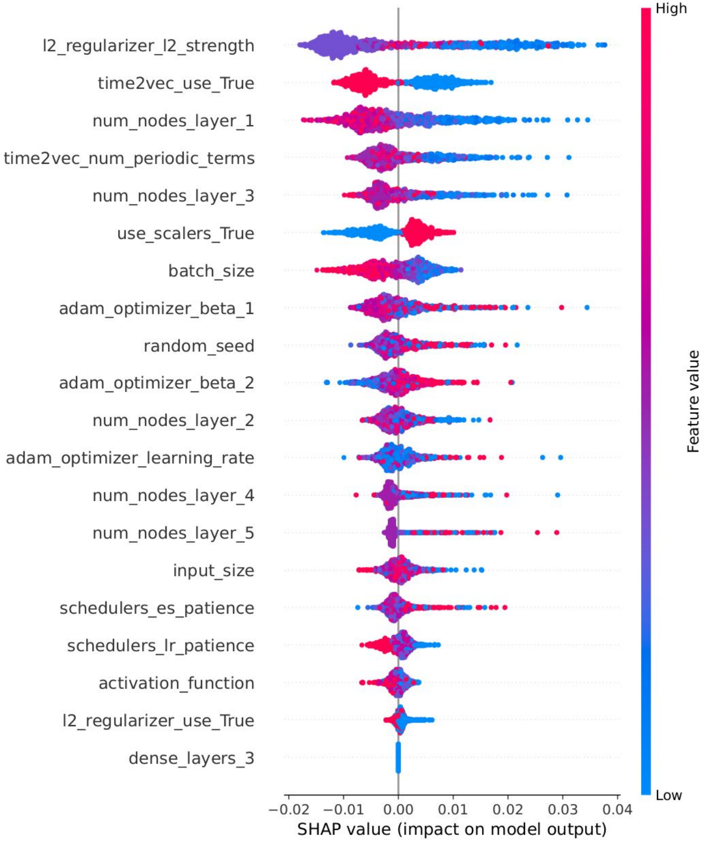


SHAP Study - NAS

- The presence of Time2Vec in neural networks reduces the forecasting loss for the given inflation data.
  - More periodic terms modeled by sin cos functions are preferred

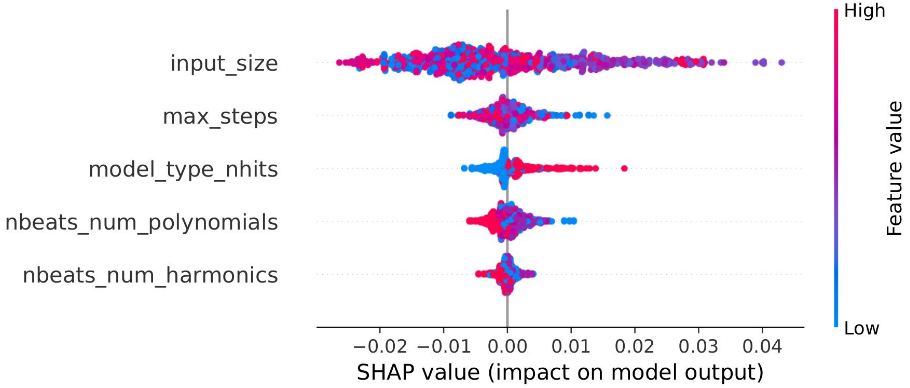
Table 11: Component-Based Neural Architecture Search (NAS-CB) model

Hyperparameter	Feature value	
	SAE Model	AE Model
Adam Learning Rate	0.00291	
Adam β_1	0.87802	
Adam β_2	0.99731	
ES Patience	55	
LR Patience	10	
Input Size	18	
Batch Size	5	10
Number of Dense Layers	3	3
Number of Layer Nodes	[36,13,16]	[29,12,25]
Activation Function	ReLU	Linear (None)
Pre-MLP NLB Compression L_{pool}	9	-
Scalers, Use	False	False
Number of Periodic Terms (Time2Vec)	48	40
Performance		
$\mathcal{L}^{val}(\mathcal{D};\Theta,\Lambda)$	0.24272	
$\mathcal{L}^{val,sae}(\mathcal{D}^{(1)};\Theta^{(1)},\Lambda^{(1)})$	0.13550	
$\mathcal{L}^{val,ae}(\mathcal{D}^{(2)};\Theta^{(2)},\Lambda^{(2)})$	0.50156	



SHAP Study - NeuralForecast

- **Input size** affects forecasting loss, but in terms of direction it is a mixed bag.
- **More training steps** typically results in better forecasting.
- **NBEATS models are favored** over NHITS
- For NBEATS the following typically yield better forecasting
  - **More polynomial** vectors
  - **More harmonic** [periodic] vectors



Conclusions

Conclusions

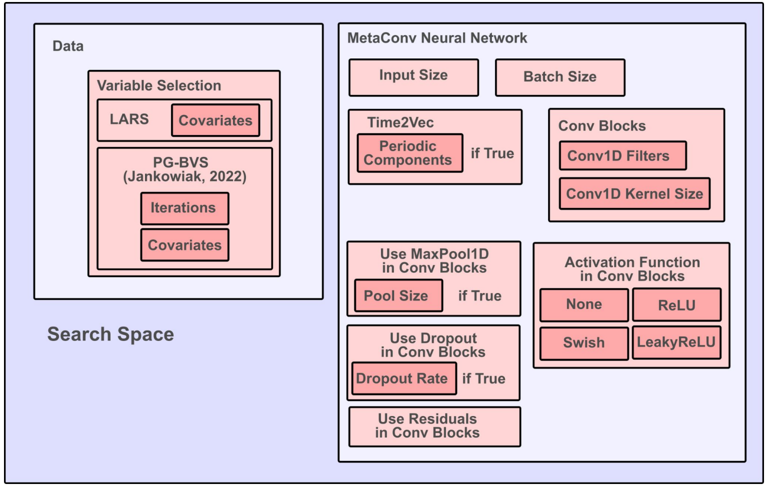
1. In neural networks, **preserving sequential memory** of input samples in time series through mechanisms like **Time2Vec** significantly improves forecast accuracy
2. NBEATS and NHITS models, have **built-in sequential analysis mechanisms**, and are consequently the best performing
3. Best practices
    - K-fold cross-validation for hyperparameter optimization in parsimonious univariate models
    - Standard validation for component-based models

Future Work

Multivariate AutoML

- Use **automated variable selection methods** for macroeconomic variables to forecast inflation
- Apply the **usage of CNN architectures in AutoML**
  - Believed to be better suited for making inferences on multivariate data

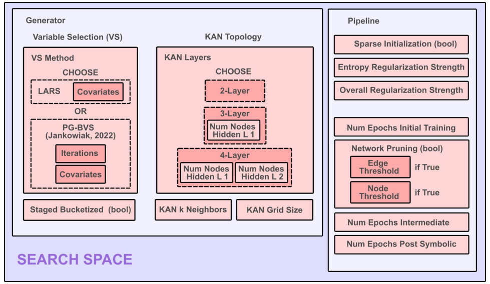
Convolutional Neural Networks (CNN)

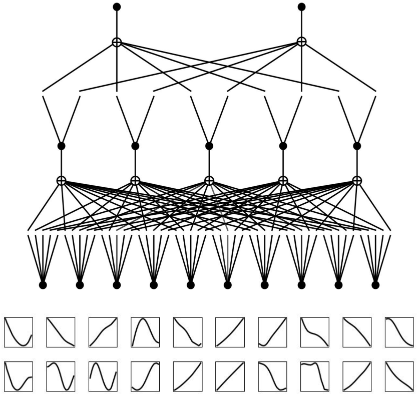


Future Work

NAS and Pipeline Optimization (AutoML) for Symbolic Representation of Inflation with Kolmogorov-Arnold Networks (KANs)

- Reverse-engineering the `pykan` repo to develop useful features to analyze inflation at a more granular level:
  - Selective input locking and
  - Input bypass
- New **symbolic representations** of headline inflation from macroeconomic variables and lags thereof with KANs





Gracias