GENERALI ASSET MANAGEMENT

CORE MATTERS

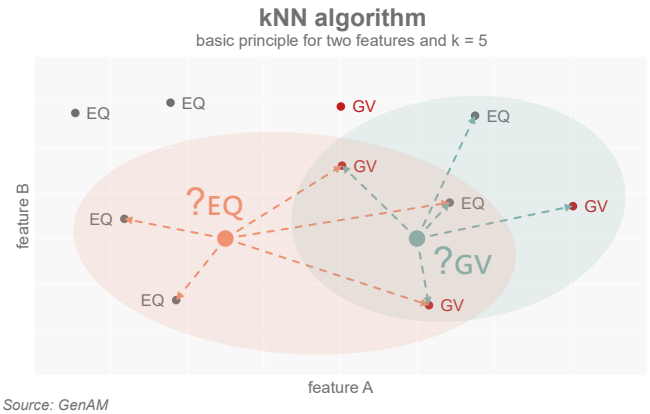
Machine Learning and Tactical Asset Allocation – Part II: Decomposing the Machine Learning Signals

Thorsten Runde

March 04, 2024

Our Core Matters series provides thematic research on macro, investment, and insurance topics

- In part I of this Core Matter we showed that integrating machine learning signals as an additional input layer into an already existing and proven tactical asset allocation process added value.
- So far, we mechanically applied the signals according to the chosen overlay strategy, thus accepting the 'black box' machine learning is widely associated which. That said, there is scope to shed light into that box.
- The algorithm that is used in our model is called: k-Nearest-Neighbours. Based on macroeconomic timeseries data, our signals are designed as a weighted majority vote amongst the representatives in the neighbourhood of the current query point i.e., the algorithms identifies periods in the past that are most similar to the current one. We can take this information about the neighbourhood from the model to contrast it with our own intuition or put it into a historical perspective by comparing it to past forecasts.



Source: GenAM

- The darker part of the box is the identification of the drivers behind the model's choice. It is not directly related to the k-Nearest-Neighbour algorithm itself but to one step in the data pre-processing. The model's choice is not directly based on the initial data but on a reduced number of (Kernel-based) principal components without knowing anything about their composition. We "unmask" the so-called Kernel-Trick and are thus able to map the model's choice back to the initial data.
- There is always a need to critically review pure model results. Hence, with the knowledge of the structure of the neighbourhood and the main drivers that led to its choice we offer a complete set of analytics allowing for a human valuation of the machine learning signals.

CONTENT

1. Basic idea and motivation

In part I of this Core Matter we showed how machine learning (ML) techniques aiming at forecasting the upcoming market regimes from freely available data can add value to a tactical asset allocation (TAA) process of a simplified portfolio consisting of US Treasuries and US equities. We explained how to tailor the ML training setup to ensure a smooth integration into an existing TAA approach. We concluded with back-testing the results showing the added-value of ML-enhanced TAA process versus a traditional one.

ML is very often associated with a black box. Thus, there seems to be an intrinsic field of tension between the application of machine learning signals and the healthy human scepticism towards any black box.

> *Confidence in any kind of model results is crucial*

Confidence in any kind of model results is crucial when it comes to applying them i.e., making decisions based on them. In that respect, we try to reduce the tensions and raise the confidence by taking a glimpse behind the scenes of the signals produced by the ML algorithm. We give the users of the model as much background information as possible to enable them to qualitatively value each signal derived by the model.

The extent to which this is possible depends on several factors. First, of course, there is the algorithm itself. Second, algorithm-independent data pre-processing steps also turn out to distort the view of the essential.

In the following, we take a closer look at the underlying structure of the signals and the driving forces behind them. Based on that, we present some standard analyses of the model we can provide to the user.

2. The model specification

The terms artificial intelligence (AI) and machine learning are widely and wrongly used synonymously. Likewise, the association of ML with a black box is exaggerated.

ML covers lots of algorithms ranging from deep neural networks to linear regressions. Not all of them are equally opaque.

> *Not all ML algorithms are equally opaque*

The algorithm that is used in our model, the so-called k-Nearest-Neighbours (kNN) algorithm, allows for a more detailed glimpse behind the scenes.

2.1 The kNN algorithm

The kNN algorithm is a so-called (a) supervised (b) instance-based (c) classifier.

(a) **Supervised**: the so-called ground truth i.e., the actual outcome is known in the training phase (in our case the market regime we want to forecast)

(b) **Instance-based**: the current instance (in our case observation i.e., month) of the data is compared with past instances of the data to produce a signal. In particular, there is no explicit generalization in a sense that an underlying functional relationship is estimated.

(c) **Classifier**: producing a signal means categorizing the current instance into a set of classes (in our case 1 (EQ) if equities are expected to outperform bonds or 0 (GV) vice versa)

The algorithm is non-parametric i.e., no assumptions about the distribution or level of measurement of the data are made. The only assumption needed is that similar instances (past observations) belong to similar classes (market regimes i.e., months in which equities outperform govies or vice versa).

The kNN algorithm

The algorithm itself is performed on a given number of neighbours *k*. The optimal value for *k* is determined via hyper parameter tuning in the training phase of the model.

The choice of the neighbours is based on the standard Euclidean distance.

Let *p* be the number of features. Let $\boldsymbol{x}$ denote an instance with $(x_1, x_2, \ldots, x_p)$ i.e., the input variables observed at a given point in time. The standard Euclidean distance between two instances $\boldsymbol{x}$ and $\boldsymbol{x}'$ is defined as:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{p} (x_i - x'_i)^2}$$

Let *D* denote the set of all instances and S_x the set of the *k* nearest neighbours of instance $\boldsymbol{x}$ with $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ each of which assigned with a class *c*:

$$S_x \subseteq D \text{ s.t. } |S_x| = k \text{ and}$$

$$\forall (\mathbf{x}', c') \in D \backslash S_x,\ d(\mathbf{x}, \mathbf{x}') \geq \max_{(\mathbf{x}'', c'') \in S_x} d(\mathbf{x}, \mathbf{x}'')$$

I.e., for any instance $\boldsymbol{x}' \in D$ and $\notin S_x$ the distance to any instance $\boldsymbol{x}$ in the neighbourhood S_x is larger than the largest distance within S_x.

Given a function *m()* that performs a binary mapping for each class *c* in the neighbourhood's classes *c''* defined as:

$$m(c, c'') = \begin{cases} 1 \text{ if } c'' = c \\ 0 \text{ otherwise} \end{cases}$$

In case of the standard Euclidean distance used as a weighting function, the classifier *h()* can finally be defined as:

$$h(\mathbf{x}) = \arg \max_{c} \sum_{j=1}^{k} \frac{1}{d(\mathbf{x}, \mathbf{x}'')} m(c, c'')$$

Based on selected features (in our case monthly macroeconomic timeseries variables), the kNN signal is designed as a majority vote amongst the representatives in the neighbourhood of the current query point i.e., assigning the class which has the most representatives. In the graph of the cover page the basic principle is shown for two features and five past instances[1].

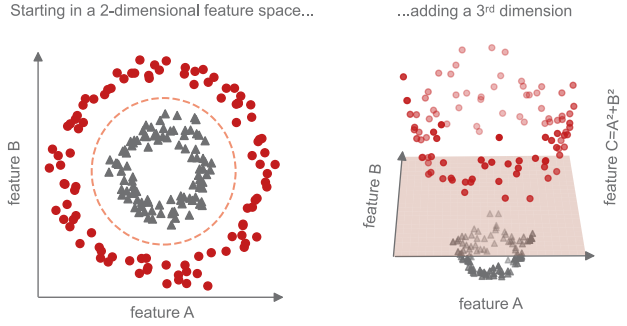
> *kNN: a simple but versatile and effective classifier*

"Despite its simplicity, nearest neighbours has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular."[2]

There are just two parameters which are optimized in the training phase via so-called hyperparameter tuning[3]. Firstly, the number of neighbours to be considered in the voting and secondly if and how the distance of the representatives in the neighbourhood shall be considered in voting i.e., the shorter the distance the higher the weight in the voting[4]. In our case the optimal combination turned out to be a weighted[5] voting amongst eleven neighbours.

With the neighbours being the core of the signal, it is straightforward to extract and scrutiny them. Yet to apply some qualitative valuation to the neighbourhood, we also need to identify the features that led to this choice. This turns out to be far less straightforward.

The basic principle of Kernel-PCA



Source: http://www.eric-kim.net, GenAM

2.2 Kernel PCA

In general ML algorithms suffer from the so-called "curse of dimensionality". Simply stated, this means that the amount of data points needed to make the model learn accurately is

[1] E.g., let the big orange dot represent the current query point or instance. Identify the five nearest past instances as the neighbourhood. Perform the majority vote amongst the representatives in that neighbourhood: 3x EQ and 2x GV ⇒ signal = EQ.

[2] https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification

[3] In ML hyperparameter tuning refers to the process of finding the optimal parameter setup for a given algorithm.

[4] For a formal representation of the algorithm s. grey box on this page.

[5] The standard Euclidean distance is used to weigh the classes of the neighbours.

exponentially increasing as the number of features increases. In case of kNN it can be shown that with a growing number of features i.e., dimensions, the only assumption made[6] can no longer be maintained.

(Kernel) PCA

Mathematically, PCs can be extracted through an eigen-decomposition of the covariance matrix Σ of the centred (de-meaned) data X:

$$\Sigma = \mathrm{cov}(X) \equiv \frac{X^T X}{n} \text{ with } X \in \mathbb{R}^{n \times p}$$

with n instances (observations) and p features (variables). The i^{th} PC is given by solving the following equation for $\mathbf{e}_i$:

$$\Sigma \mathbf{e}_i = \lambda_i \mathbf{e}_i \text{ s.t. } \|\mathbf{e}_i\| = 1$$

with $\mathbf{e}_i$ being the unit length i^{th} eigenvector and λ_i the i^{th} eigenvalue of X. PCs are extracted for λ_i sorted in descending order. Projections of X onto the PCs can be derived by calculating the dot product between a de-meaned instance $\boldsymbol{x}$ and the desired eigenvector $\mathbf{e}_i$.

Kernel PCA applies a non-linear transformation Φ to the data X i.e., it maps it to a higher dimensional space, before doing the eigendecomposition:

$$\phi(X) : \mathbb{R}^p \rightarrow \mathbb{R}^m \text{ with } m \gg p$$

But, instead of directly applying Φ to the data X the transformation is performed through a pairwise application of a corresponding Kernel κ to all instances $\boldsymbol{x}$, the so called Kernel-Trick:

$$\kappa(\mathbf{x}, \mathbf{x}') = \phi\left(\mathbf{x}^T \mathbf{x}'\right)$$

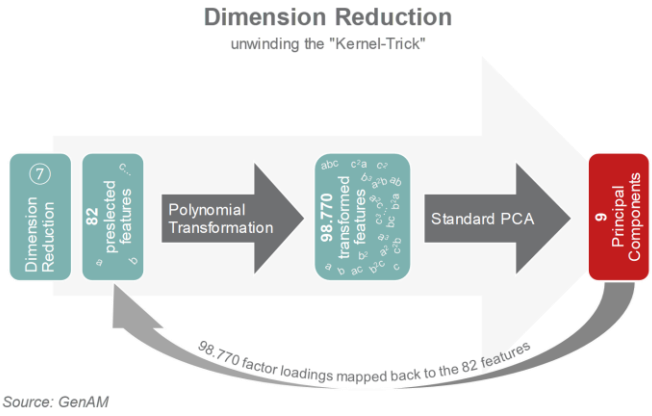
which results in the symmetric kernel matrix $K \in \mathbb{R}^{n \times n}$. By applying an eigendecomposition to a de-meaned version of K, PCs can be extracted like above. However, as the covariance matrix of $\Phi(X)$ is not calculated explicitly, no principal component axes are yielded. That said, the resulting eigenvectors already represent the projections of the data X onto the respective PCs.

In fact, it can even be shown that the distribution of the distances collapses to an extremely small and thus little representative range.

Hence, dimension reduction is a standard data pre-processing step in ML. In our case principal component analysis (PCA) turns out to be the most efficient way to reduce the dimensionality, more precisely, Kernel-PCA[7].

Roughly speaking, PCA aims at the revelation of data inherent dependencies. It reduces the dimensionality by identifying orthogonal principal components (PCs) that are linear combinations of the original features. In doing so, a sufficiently large share of the variance in the original data can be explained by a distinctively smaller number of PCs. In our case nine PCs were extracted[8] out of 82 input features.

Standard PCA is a linear approach. But, very often, like in our case, the data classes (EQ, GV) are not linearly separable in the original feature space. Via Kernel transformation additional dimensions are added, thus achieving linear separability in the transformed, higher dimensional, feature space. Projecting this back to the original features leads to a non-linear separation.[9]



Applying a non-linear transformation to the data before performing the PCA is computationally extremely costly. Thus, it is infeasible to do so in the training phase of the model, where thousands of specifications must be calculated to figure out the optimal parameterization.

[6] Similar instances belong to similar classes. (see page 2)
[7] Compared to standard PCA Kernel PCA improved the results significantly. Feature selections directly aiming at identifying the best combination of a prespecified number of features proved computationally not feasible.
[8] The optimal number of PCs is evaluated within the model training.
[9] See chart on the previous page which gives an extreme yet illustrative example. It is impossible to draw a line in the left sub-chart that separates the red dots from the grey triangles. Adding a 3rd dimension as the sum of the squared original features allows for linear separation by constructing a two-dimensional plane between the dots and the triangles (s. right sub-chart). Projecting the plane back to the original feature space leads to the dashed circle which is of course non-linear and perfectly separates the dots from the triangles.

The Kernel-Trick: blessing and curse at the same time

The solution for this dilemma is the so-called Kernel-Trick[10] which allows to pass on the transformation of the original data and to directly operate in the low-dimensional PC space. The blessing is: you do not even have to know the transformations which is indispensable in case their number is infinite[11]. At the same time, it is a curse, as not knowing the transformations also means not knowing anything about the composition of the PCs. Thus, in that respect, Kernel-PCA is a black box.

In our case, the best Kernel turned out to be a polynomial of degree three i.e., the number of transformations, though large, is at least finite[12]. Furthermore, we only want to analyse the composition of the PCs for selected signals.

We "unwind" the Kernel-Trick by applying the transformation represented by the kernel directly to our input data. In a second step, we perform standard PCA on this transformed data. Now we do have the full information about the compositions of the PCs that we map back to the original features (see chart on previous page). Hence, we are enabled to figure out the economic drivers behind the choice of the neighbourhood and thus of the final signal.

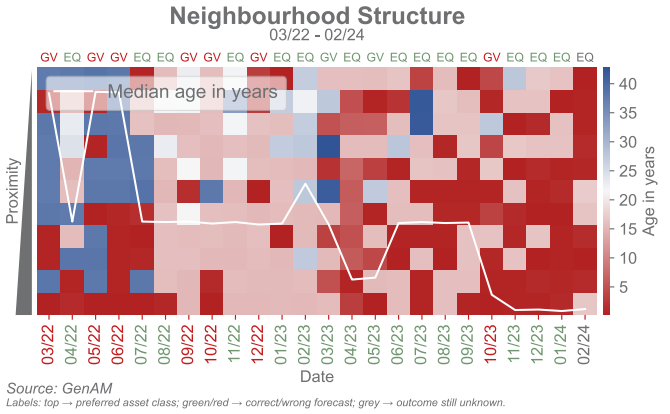
3. Decomposing the signals

So far, the signals stemming from the ML model were just ones (EQ or equities preferred over next month) or zeros (GV or government bonds) and the user was faced with a take it or leave it situation. But, given the findings from the previous chapters, we can now start to decompose the signals and analyse its key drivers in terms of variables.

3.1 The neighbourhood

The HeatMap on the right shows the structure of the neighbourhoods behind the ML signals over the past two years. The neighbours are sorted vertically by proximity (Euclidian distance) with the nearest neighbour in the lowest row. The age of the neighbours i.e., the time distance, is represented by the colour shading. Red neighbours are young and blue neighbours old.

Particularly until July 2022 we find quite some very old neighbours, partly 35 years old or even older. This might be considered a bit surprising at the first moment as we are analysing timeseries data.

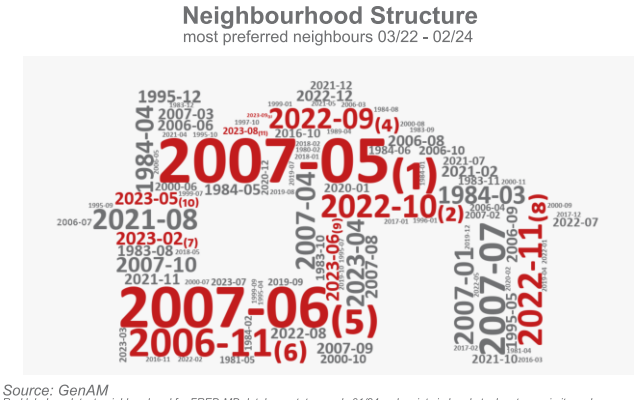


Source: GenAM
Labels: top → preferred asset class; green/red → correct/wrong forecast; grey → outcome still unknown.

Proximity and age do not necessarily coincide

That said, as the data passes through a series of pre-processing steps before entering the algorithm, including detrending and standardization, the chances for each past observation to be included in the neighbourhood are equal.

Taking a closer look at the favourite neighbours (see WordCloud below), it is obvious that some neighbours are more preferred than others. For the possible (11 neighbours x 24 months =) 264 places, just eighty-one candidates were finally selected, with a clear preference for spring / summer 2007 and the end of 2006. The two most popular neighbours of the last two years (2007-05, 2006-11), although not on top ranks, are also found in the neighbourhood behind the current ML signal (the forecast period ending mid-February 2024).



Source: GenAM
Red labels → latest neighbourhood for FRED-MD database status: early 01/24; subscripts in brackets denote proximity ranks

[10] For more details about Kernel-PCA and the Kernel-Trick see grey box on previous page..
[11] E.g., this is true for the for the Radial Base Function (RBF), a kernel which projects into an infinite dimensional space.
[12] Even just a polynomial of degree three already boosts the number of features from 82 to 98.770 (see chart on previous page).

3.2 The drivers

So far we just looked at the structure of the neighbourhoods, i.e., we know which past observations have been picked as the most similar ones to the current query point. Yet we still do not know in which aspects these instances are similar.

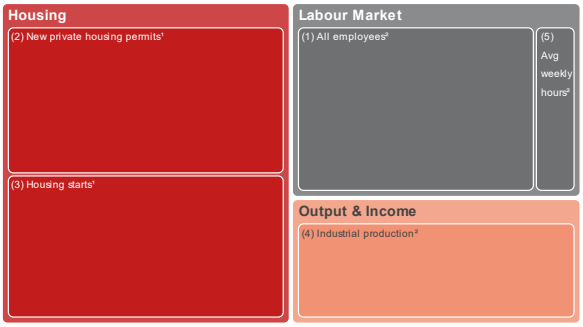
In any case, similarity cannot be measured directly through the features but only indirectly through the identified PCs. This leaves us with the necessity to find a "link" between the PCs and the factor loadings i.e., the coefficients of the features within the PCs.

To aggregate the factor loadings across PCs we can either use the PC's share of explained variance or the relative SHAP[13] values. Whereas the former helps to identify the driving forces behind the model itself, the latter is focused on single concrete forecasts.

3.2.1 Model drivers

The TreeMap chart below condenses the composition of the nine PCs based on their share in explaining the overall variance of the features to the TOP 5 model drivers[14]. It can be seen that the neighbour selection is primarily based on their similarity with respect to "Housing", "Labour Market", and "Industrial production".

TOP5 Model Drivers
Forecast period: 01/24 - 02/24



Source: GenAM
FRED-MD database status: early 01/24; PCA loadings aggregated by share of explained variance; ¹sum across regions; ²sum across sectors

Although referring to the concrete forecast period ranging from mid-January 2024 to mid-February 2024, this composition proved pretty stable over time[15]. This is not too surprising as the model is updated incrementally i.e., a new observation added to a growing database will impact the variance decomposition less and less.

[13] SHAP = **SH**apley **A**dditive ex**P**lanations leverages on findings in cooperative game theory and adopts Lloyd Shapley's solution concept to find a fair dividing of the winnings of a game amongst its players, already introduced in 1951. For a few more details see grey box on this page.

3.2.2 Forecast drivers (SHAP)

Although it is essential to know what the model relies on in general additional insights can be achieved by identifying the drivers behind a single concrete forecast. To do so, we switch from the share of explained variance to relative SHAP values to aggregate the factor loadings across PCs.

SHAP values

In the following we directly apply the terminology from machine learning to briefly explain the concept.

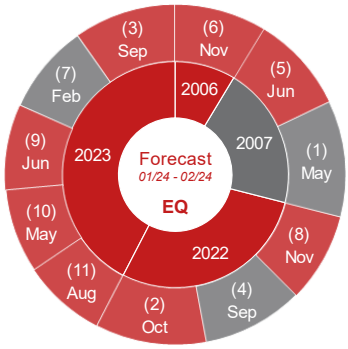
Again, let p be the number of features. Let $\boldsymbol{x}$ denote an instance with $(x_1, x_2, \ldots, x_p)$. In essence, a SHAP value $\varphi_i(f)$ gives the expected marginal contribution of a specific feature x_i to a concrete forecast $f(\boldsymbol{x})$.

$$\varphi_i(f) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{x_i\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(f(S \cup \{x_i\}) - f(S)\right)$$

S denotes a subset of features and $|S|$ the number of features in this subset. $f(S \cup \{x_i\})$ is the forecast including feature x_i and $f(S)$ the forecast excluding feature x_i. The sum is across all potential subsets to which feature x_i could be added.

The SunBurst chart below reveals the neighbourhood structure behind the ML signal for the same forecast period as before[16]. The size of the outer segments represents the neighbours' weights in the model's voting scheme.

Neighbourhood Structure



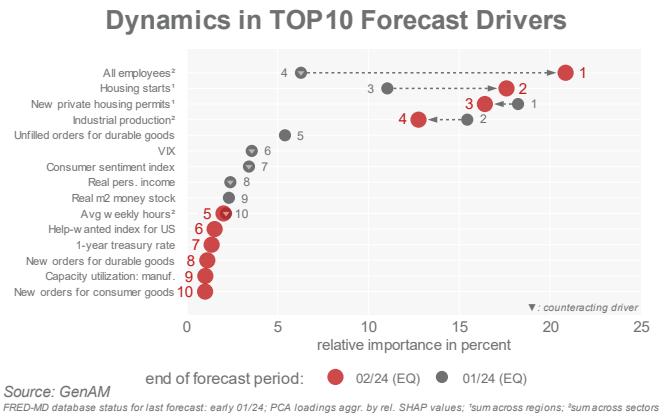
Source: GenAM
FRED-MD database status: early 01/24; red/grey segments in favour of EQ/GV

In this forecast, which favours Equities, there is a preference for the very recent past, as can be seen by the share of the years 2023 and 2022 which roughly accounts for two thirds of

[14] The TOP 5 is roughly 2.5 times as important as the remainder of the input features together.
[15] See chapter 3.3 The historical perspective.
[16] These are the same neighbours shown in red in the WordCloud on page 5.

the voting[17]. Furthermore, the voting in favour of equities is fairly clear as just three of the eleven neighbours (May/2007, Sep/2022, and Feb/2023) are associated with a Government Bond regime.

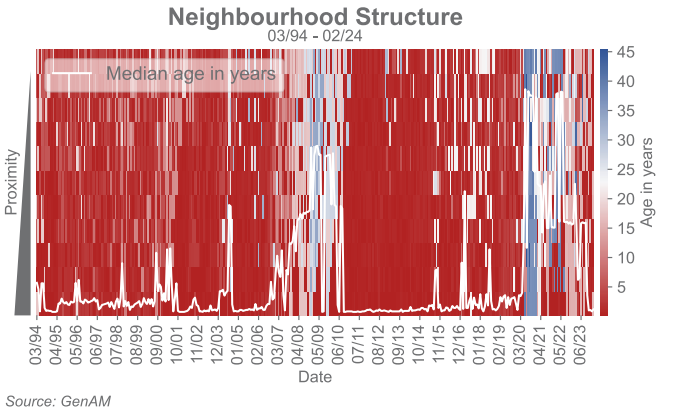
To identify the crucial features that led to the choice of these neighbours and thus to the forecast in favour of equities we look for some "implicit confidence" behind the forecast. It gives us an idea of how clear the voting result was rather than the forecast itself[18]. It can be derived from the weights[19] with which each neighbour contributes to the voting result and thus, the forecast.



The Dumbbell chart above shows the ten most relevant features for the choice of the neighbourhood[20]. In this case the forecast drivers are more or less in line with the overall model drivers[21] (large red dots). Yet, the comparison with the previous forecast (small grey dots) shows there was quite some fluctuation amongst the drivers. Furthermore, although the ranking of the top four remained quite stable there are some significant shifts in their importance (dashed arrows). The downward pointing triangles indicate counteracting features i.e., features that work in the direction of a less clear voting result with respect to the forecasted market phase. There were five counteracting features identified for the previous forecast but none for the current one.

3.3 The historical perspective

So far, our consideration represented just a snapshot or referred to a rather short period of time. To add a further layer of interpretability we are now resorting to a broader historical context i.e., we analyse how neighbourhoods and model drivers evolved over time.



Expanding the view on the neighbourhoods to the past 30 years we can find further periods with a high median age of the neighbourhood (e.g.: 11/2008 – 05/2010). In the period from 06/2020 until 05/2021 we even yielded a median age of slightly below 40 years (see HeatMap chart above) which coincides with the findings for the first half of 2022. In that respect, one might conclude that the neighbourhood structures over the past two years, while not unique, are at least somehow outstanding.

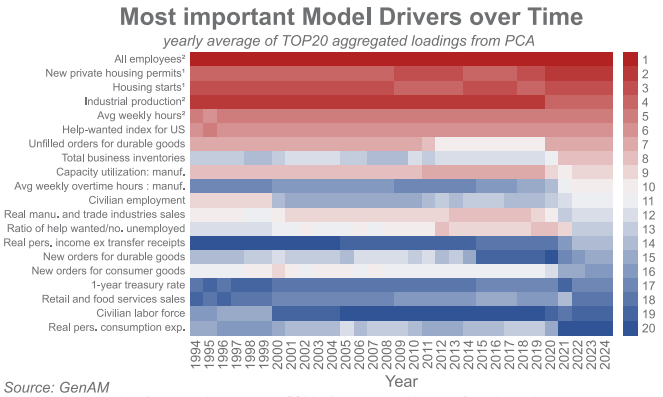
The median age of all neighbours across the past 30 years is 20 months. Thus, the age of the neighbourhood behind the more recent forecast discussed in 3.2 is clearly below average. In fact, just one third of all neighbourhoods in the past 30 years was even younger.

Model drivers pretty stable over time

Putting the model drivers into a longer historical perspective (see upper HeatMap chart on the next page) we can see that the TOP 20 did (nearly) not change in the past two years. However, moving further into the past, quite some movements can be observed in the lower ranks but also to some degree even in the top ranks[22] compared to the last two years. That said, there always seem to be periods of two to three years lengths in which the structure of the PCs appears quite stable. After some sort of transition phase in 2020 and 2021

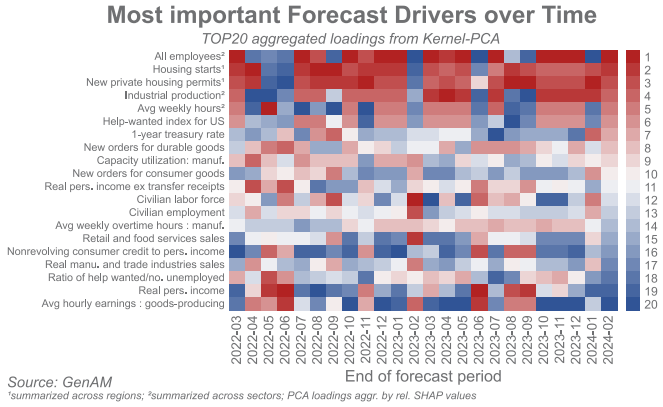
[17] In fact, the median age is slightly above one year.
[18] As our forecast are either one ore zero there is no meaningful direct application of SHAP values to the forecast itself.
[19] See SunBurst chart on page 6. With 82% the implicit confidence behind the current forecast is distinctively higher than the 30y-average of 66%.
[20] Please note that in game theory Shapley's values sum up to the overall winnings of a game. This requires some kind of natural zero point i.e., if nobody plays nothing will be won. Such a natural zero point generally does not exist in the context of ML. Instead, the approach is applied to deviations from an average. In our case the average probability is calculated based on the 15 years preceding the forecast under scrutiny.
[21] See TreeMap chart on page 6
[22] Please keep in mind that just the number of PCs was fixed during the training of the model. Their composition is of course depending on the data available until the query period. As the model was trained to perform best on average over a 30-year-period, too much variation in the top drivers should not be expected.

we seem to have re-entered such a stable phase in the most recent years. From 1994 onwards "Industrial Production" (No.2 until 2019) and "All employees"[23] (No. 1 until now) held the top two positions. That said, although never holding the top rank, it is the housing market ("New private housing permits", "Housing Starts") that dominates the PC composition.



Variability in forecast drivers distinctively higher

In contrast to these findings for the model drivers the drivers for concrete forecasts vary substantially even over shorter period of time (see HeatMap below).



Although the top ranks might be quite stable for some months in a row the variability generally spans across all the twenty drivers shown in the chart above[24]. E.g., "All employees" holds the top rank in twelve out of twenty-four months but it also finds itself amongst the bottom five in eight months.

3.4 Further Findings

Since the voting is at the heart of the ML signal generated by the kNN algorithm, we dug even a bit deeper by further analysing the "implicit confidence" already used to calculate the SHAP values[25].

Unfortunately, an analysis of the model accuracy conditional on quantiles of this implicit confidence did not reveal any statistically robust relationship. i.e., a high "implicit confidence" does not necessarily coincide with a higher likelihood of a correct forecast. We performed similar analyses on the average age of the neighbourhoods. Again, no statistically robust relationship could be found.

What may seem disappointing at first glance also leaves room for a more constructive view when compared to traditional econometric approaches. In traditional econometrics, residuals not containing any noticeable patterns any longer are taken as an indication for a well specified model. Not being able to identify statistically robust relationships in the structures of the ML signals might thus be regarded in the same way. There seems to be no additional information beyond the signals which could be systematically taken advantage of. Hence, the model already appears to exhaustively exploit the information contained in the input features.

4. Conclusions

Confidence in any kind of model results is crucial, we stated at the beginning of this paper. Depending on the algorithm applied, ML needs not necessarily be a black box. In fact, the kNN algorithm allows for high transparency by revealing the structures and the drivers of the model signals. The user is enabled to validate his/her own view against that of the machine and thus to decide whether to make use of the signal or not. In that sense, the user is put into the position to perform a quality check before taking a decision, which was the ambition of this paper.

As a matter of fact, the emphasis is on quality. There were no options detected to exploit the information behind the signals quantitatively. That being said, we take this as an indication for a sufficiently well parameterized model.

All in, both enabling a quality check of the signals and confirming the model parameterization underpins our confidence in the model results.

[23] The underlying raw variable measures the total number of employees.
[24] Over the past ten years the members of the top five model drivers did not change whereas 35 different forecast drivers had made it into the top five at least once within the same period of time.

[25] See 3.2.2 Forecast drivers (SHAP)

Imprint

Issued by:	**Generali Asset Management S.p.A.** **Società di gestione del risparmio, Research Department**
Head of Research:	**Vincent Chaigneau**
Head of Macro & Market Research:	**Dr. Thomas Hempell, CFA**
Team:	**Elisabeth Assmuth** | Research Operations **Elisa Belgacem** | Head of Cross-Asset Quant & Dev, Senior Credit Strategist **Radomír Jáč** | GI CEE Chief Economist **Jakub Krátký** | GI CEE Financial Analyst **Michele Morganti** | Head of Insurance & AM Research, Senior Equity Strategist **Vladimir Oleinikov, CFA** | Senior Quantitative Analyst **Dr. Thorsten Runde** | Senior Quantitative Analyst **Dr. Christoph Siepmann** | Senior Economist **Dr. Florian Späte, CIIA** | Senior Bond Strategist **Guillaume Tresca** | Senior Emerging Market Strategist **Dr. Martin Wolburg, CIIA** | Senior Economist **Paolo Zanghieri, PhD** | Senior Economist
Head of Insurance and AM Research:	**Michele Morganti**
Team:	**Raffaella Bagata** | Research Operations **Alberto Cybo-Ottone, PhD** | Senior Economist **Mattia Mammarella** | Research Analyst **Roberto Menegato** | Senior Insurance Research Analyst **Antonio Salera, PhD** | Economist, Pension Expert **Federica Tartara, CFA** | Senior Economist
Head of Credit Research:	**Vivek Tawadey**