# ML Predictability

Empirical Asset Pricing

**Mads Markvart Kjær**

Department of Economics and Business Economics, Aarhus University, CREATES

E-mail: mads.markvart@econ.au.dk

Spring 2022

# What is Machine Learning (ML)?

## Definition 1 (Samuel): Machine learning

Machine learning is the use of algorithms and statistical models that computer systems employ to perform a specific task without using explicit instructions. It relies on patterns and inference instead. Machine learning algorithms build a mathematical model based on sample data, known as “training data", in order to make predictions or decisions without being explicitly programmed to perform the task (Samuel, 1959).

## Definition 2 (Mitchell): Machine learning

A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$ if its performance at tasks in $T$, as measured by $P$, improves with experience $E$ (Mitchell, 1997).

# Another definition (sort of)

> Interviewer: What's your biggest strength?
>
> Me: I'm a fast learner.
>
> Interviewer: What's 11 * 11?
>
> Me: 65.
>
> Interviewer: Not even close. It's 121.
>
> Me: It's 121.

# Another definition (sort of)

# A different way of thinking

- Previos in the course, we started from a model (i.e., a DGP) and our outcome was to estimate the true parameters (like the case of the SDF)
- Now, we move into the area in which we (in the extreme case) do not care about the DGP but ultimately want to do predictions!

# ML in Empiricial Asset Pricing

# ML in empirical asset pricing

## Definition: Machine learning in practice

Machine learning is:

- A diverse collection of high-dimensional models for statistical prediction
- Combined with so-called regularization methods for model selection and the mitigation of overfit
- Efficient algorithms for searching among a vast number of potential model specifications.

# ML in empirical asset pricing

- Element i) enhances the flexibility relative to more traditional econometric techniques - brings hope of better approximating the unknown and likely complex (true) data generating process.
- Element ii) guards against overfitting that is likely to arise due to the higher flexibility (and no. of parameters) and low signal-to-noise environment.
- Element iii) describes ML techniques that are designed to approximate an optimal specification with manageable computational costs.

$\rightarrow$ For the following three reasons, ML is well-suitable for empirical asset pricing!

# Reason 1: The fundamental research agenda

- Empirical asset pricing centres around estimating the (conditional) expectation of asset returns (recall the course introduction):
  1. Time series dynamics of assets' risk premium (excess returns).
  2. Cross-sectional variation in assets' average excess returns.
- Both are fundamentally a question of predictability either in the time series or the cross-sectional direction.
- Which requires a specification/search or learning the true function for the conditionally expected return.

$\Rightarrow$ ML takes the view that the best approximation model for expected returns should be learned!

# Reason 2: The collection of relevant variables is enormous

- The profession has accumulated a staggering list of predictors and risk factors that has been argued are important for prediction/measuring assets' excess returns.

## Snip of relevant variables

- Harvey et al. (2016) have identified 316 risk factors.
- Green et al. (2017) have collected 94 stock-level predictive signals.
- Welch and Goyal (2008) have collected 20 (at various frequencies) predictors for the aggregate market risk premium.
- McCracken and Ng (2016) have collected in excess of 100 macroeconomic variables that are often used for predicting stock returns, both in the time series and in the cross-section.

...

# Reason 2: The collection of relevant variables is enormous

- Those variables are often correlated and do likely not represent in excess of, say, 300 orthogonal and important signals.
- Traditional econometric techniques would fail massively (and not even be feasible) in those cases.

$\Rightarrow$ ML techniques are well suited for so-called high-dimensional settings, by encompassing variable selection and dimension reduction techniques.

# Reason 3: The functional form is unknown and likely complex

- Further complicating the problem of empirical asset pricing is a fundamental ambiguity regarding the functional form through which the high-dimensional predictors should enter into excess returns.
- Should they enter linearly? Non-linearly? If so, how? Through interactions? In squared or cubed forms?
- This radically proliferates the number of predictors and the dimensionality of the problem!
- There is not much help in the literature of theoretical asset pricing in which it all comes down to an assumption...

# Reason 3: The functional form is unknown and likely complex

$\Rightarrow$ ML techniques are well suited for this, as it learns, with an emphasis on parsimony, the best specifications due to

1. considering several specifications,
2. incorporating non-linear models
3. penalizing additional parameters and using conservative model selection to avoid overfitting

# What ML cannot do

- Fundamentally, ML has great potential for improving the estimation of the (conditional) expectation of excess returns, that is, best approximate the form for $\mathbb{E}[r^e_{it+1}|\mathcal{F}_t]$.
- But this is just *measurements*. It is silent about economic mechanisms and equilibria.
- This requires putting structure on the problem and applying ML in an appropriate manner.
- The approach in Gu et al. (2020) is not generally about this, but we will address this a bit here anyway, and otherwise much more in the last topic of the course on cross-sectional asset pricing.

# A nice overview!

- Check out Masini et al. (2020) for an overview of ML methods in time series forecasting.

# ML Methods

# Return predictability via ML

## Return predictability in very general terms

In its most general form, assets' excess returns can always be expressed as

$$r^e_{i,t+1} = \mathbb{E}_t[r^e_{i,t+1}] + \varepsilon_{i,t+1}, \tag{1}$$

where

$$\mathbb{E}_t[r^e_{i,t+1}] = g^*(z_{i,t}), \tag{2}$$

$\varepsilon_{i,t+1}$ is a forecast error term and $z_{it}$ is a $P$-dimensional set of (possibly stock-specific) predictors, $i = 1, \ldots, N$, and $t = 1, \ldots, T$.

- The objective is to learn the best specification of the general function $g(\cdot)$, and we will later see numerous ML techniques that are quite successful in this.
- Note that this function is independent of $i$ and $t$.

# Return predictability via ML (in practice)

- The main focus in this lecture will be on out-of-sample predictability.

## Out-of-sample predictability

* Assessing predictability in-sample entails estimating the predictive regression using the full range of available observations
* Assessing out-of-sample predictability, conversely, entails using information available at time $t$ only to forecast returns at time $t+1$
* To emulate a forecaster in real-time, we split the total sample ($T$) in two parts: in-sample (initial) and out-of-sample

$$\underbrace{t = 1,2,\ldots,R}_{\text{In-sample}}, \underbrace{R+1,R+2,\ldots,T}_{\text{Out-of-sample}} \tag{41}$$

* One can either estimate the regression coefficients using a rolling or an expanding window of data (benefits/drawback?)
* Irrespective of choice, we end up with a sequence of forecasts $\{\widehat{r}_i\}_{i=R+1}^{T}$ and forecast errors $\{\widehat{\varepsilon}_i\}_{i=R+1}^{T}$ for evaluation

# Hyperparameters

- All methods we will go through depends on one (or more) choice (s) made by the researcher
- Many of these choices goes under the definition of hyperparameters:

## Definition: Hyperparameters

Most of the ML techniques depends on parameters that are not part of the model, but rather specifies parts of the estimation procedure.

- These typically control e.g. regularization (a defense against overfitting).
- Examples include penalty parameters in LASSO or number of random trees in a forest.

# Choosing hyperparameters

For instance, the following three approaches are common

- Cross-validation
- Information criteria (Related to model selection and regularization)
- They can be “tuned" to optimize out-of-sample performance

  $\rightarrow$ We will for now mostly focus on the tuning approach

# Return predictability via ML (in practice)

- We need to re-define the design of the out-of-sample study by introducing additional sample splitting and hyperparameters (or tuning parameters).

## Definition: Sample splitting for tuning hyperparameters

We can tune hyperparameters by splitting the sample into three disjoint (not the usual two) time periods that maintain the temporal ordering of data.

1. Training period: This period/data is used to estimate (train) the model, given a choice of hyperparameters.
2. Validation period: The trained model from the training period is then used to form predictions for the validation period. Here, the objective function from the forecast evaluation is computed (e.g. out-of-sample $R^2$).
3. Testing period: This sample period is not used in estimation/training, nor validation step, and is truly out-of-sample, used to evaluate the methods' predictive ability.

$$\underbrace{1,2,\ldots,R_1}_{\text{training}},\underbrace{R_1+1,R_1+2,\ldots,R_2}_{\text{validation}},\underbrace{R_2+1,R_2+2,\ldots,T}_{\text{testing}} \tag{3}$$

# Return predictability via ML (in practice)

- To determine hyperparameters, one iterates between the training and validation period to choose the hyperparameters that minimize the value of the objective function applied to the resulting forecast errors.
- Each time one re-estimates the model for the new candidate choice of hyperparameters.
- The principle is to simulate a true out-of-sample evaluation of the model to determine the best value for the hyperparameters.
- Roughly speaking, you can think of the joint training and validation period as “in-sample" and the testing period as “out-of-sample" in the terminology seen in the first lectures on return predictability.

# Return predictability via ML (in practice)

- As usual, it entails a choice about whether to use fixed, rolling, or expanding (recursive) estimation schemes.
- Gu et al. (2020) apply a hybrid that using a recursive scheme for the training period but a rolling scheme for the validation period.
- This continuously increases the sample size for training the model, yet always tune hyperparameters over a window with a fixed amount of data points, subsequent to the end of the training period.

# ML techniques

- We will now (quickly) go through several popular ML techniques that are useful in the context of empirical asset pricing.

## Three fundamental elements

Each ML technique may be described by three fundamental elements:

1. The model for $g^*(\cdot)$: Each technique implies some model for describing the general functional form for excess return predictions.
2. The objective function: All techniques employ mean squared prediction error (MSE) as objective function for estimation, yet some with penalty terms used for regularization.
3. Computational algorithms for optimal specification: When one considers non-linear transformations of predictors, their number proliferates and ML techniques entails an efficient approach to search all possible specifications of the model. This point is mostly out of the scope of this course.

## Types of (machine) learning

There exists broadly speaking three types of learning:

1. Supervised learning: Data contains both dependent and independent variables and models are trained to learn the relationship between those, typically used for prediction (example is OLS regression).
2. Unsupervised learning: Data contains only independent variables and models are constructed as to find a structure in this, like grouping or clustering of data (example is PCA).
3. Reinforcement learning: The models learn how to take actions, given data, in an environment through feedback and rewards (example is to play chess).

- Our focus will be mainly supervised learning, yet sometimes with an element of unsupervised learning semi-supervised learning is thus applied.

# Simple linear regression model

- The first method cannot really be classified as an ML technique (even though it is actually the starting point of many ML books!)
- This serves as reference point for the performance of the many ML techniques we will consider.

## Simple linear model

1. Model: The model is linear in both variables and parameters as

$$g^*(z_{it};\theta) = z_{it}'\theta. \tag{4}$$

2. The objective function: The typical $\mathbb{L}_2$ objective function

$$\mathcal{L}(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}(r_{it+1}^e - g(z_{it};\theta))^2. \tag{5}$$

Gu et al. (2020) write about several additional extensions to the objective function which are robust towards heavy tails, among other things.

# Penalized linear model

- In the presence of many predictors, the simple linear model is bound to fail since when the number of predictors approaches the number of time series observations, it is both inconsistent and inefficient.
- It will almost surely overfit to noise rather than extracting the signal.
- This is particularly relevant for return predictability where the signal-to-noise ratio is low.

  ➜ A natural solution is to reduce the number of parameters to estimate by sorting out the irrelevant variables known as regularization.

## Penalized linear model

1. Model: The model is (still) linear in both variables and parameters

$$g^*(z_{it};\theta) = z_{it}'\theta. \tag{6}$$

2. The objective function: The objective function, that we minimize, differs from before by adding a penalty term to the original loss function

$$\tilde{\mathcal{L}}(\theta;\lambda,\alpha) = \mathcal{L}(\theta) + \phi(\lambda,\alpha), \tag{7}$$

where $\lambda,\alpha$ are non-negative tuning parameters. We will work with the Elastic Net (ENet) formulation as per

$$\phi(\lambda,\alpha) = \lambda(1-\alpha)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\alpha\sum_{j=1}^{P}\theta_j^2, \tag{8}$$

effectively imposing a “budget constraint" on coefficients. In most cases $\alpha = 0.5$ is thought to be defined as ENet, $\alpha = 0$ as LASSO, and $\alpha = 1$ ridge regression.

## Penalized linear model (cont'd)

- LASSO sets coefficients equal to zero and perform, as such, variable selection.
- Ridge regressions shrinks coefficients to zero, but does not set them exactly to zero.
- ENet is somewhere in between, with the weight given by $\alpha$.

# Dimension reduction: PCR and PLS

- Penalized regressions uses regularization to handle the high-dimensionality of data.
- However, this has problems if the predictors are highly correlated.

## Possible issue with regularization

- Suppose, simplistically, we have a (possibly large) set of predictors and suppose they are all equal to the target variable plus iid noise.
- Performing regularization is suboptimal as opposed to taking just the average among all variables and use the average as the predictor.
- Why? Because the averaging cancels out the noise and “extracts" the true signal, yet regularization only removes some of the variables with noise and leaves many that (still) includes noise.

# Dimension reduction: PCR and PLS

- This idea with predictor averaging as opposed to predictor regularization/selection is the main idea behind dimension reduction techniques.
- These techniques form linear combinations, for instance the average, of the predictors to help reduce noise and isolate the de-correlated signals in predictors.
- We will see the use of Principal Components Regression (PCR) and Partial Least Squares (PLS).
- Both procedures consist of two steps …

# Dimension reduction: PCR and PLS

- …In the first step, PCA combines all regressors into a small set of linear combinations that seeks to extract the common signals among the variables.
- In the second step, those common signals are used in a standard predictive regression (that you have seen in previous lectures).
- That is, PCR regularizes the prediction problem, in some sense, by zeroing out low variance components.
- PCA is inherently unsupervised in the first step, as it does not incorporate the forecast objective when estimating components it condenses information *among* predictors.

# Dimension reduction: PCR and PLS

- PLS, on the other hand, conducts the dimension reduction while exploiting the covariance between the target variable and the predictors.
- That is, it essentially constructs components (linear combinations of predictors) that maximise the covariation with the target variable.

## PLS components (intuitively)

1. For each predictor $j$, estimate its univariate return prediction coefficient via OLS. This coefficient reflects the partial sensitivity of returns towards the $j$'th predictor.
2. Average now all predictors into a single aggregate component with weights proportional to the coefficients from prior step, placing most weight on the most important variable and vice versa.

$\Rightarrow$ as such, PLS performs dimension reduction with the ultimate forecasting objective in mind. To form more than one component, one orthogonalizes the target and the predictors w.r.t. previously constructed components and above procedure is repeated.

## PCR and PLS models

1. Model: In vectorized form the linear regression model is

$$R = Z\theta + E, \tag{9}$$

where $R$ is $NT \times 1$, $Z$ is the $NT \times P$ stacked vector of predictors, and $E$ is a $NT \times 1$ error term (vectorization makes sense since $g(\cdot)$ is independent of $i$ and $t$). Both PCR and PLS condense all predictors into $K$ predictors, being linear combinations of all $P$ predictors, as per

$$R = (Z\Omega_K)\theta_K + \tilde{E}, \tag{10}$$

where $\Omega_K$ is $P \times K$ with columns $\omega_1, \omega_2, \ldots, \omega_K$ that contains the linear combination weights. As such, $Z\Omega_K$ is the dimension-reduced set of predictors.

## PCR and PLS models (cont'd)

**2.** The objective function: PCR chooses combination weights $\Omega_K$ recursively as per

$$\omega_j = \arg max_{\omega} \text{Var}[Z\omega], \tag{11}$$

subject to $\omega'\omega = 1$ and $\text{Cov}[Z\omega, Z\omega_l] = 0, l = 1,2,\ldots,j-1$. As such, PCR takes the $K$ linear combinations of $Z$ that most faithfully mimic the full predictor set.

PLS seeks the $K$ linear combinations of $Z$ that have maximal predictive association with the forecast target as per

$$\omega_j = \arg max_{\omega} \text{Cov}[R,Z\omega], \tag{12}$$

subject to $\omega'\omega = 1$ and $\text{Cov}[Z\omega, Z\omega_l] = 0, l = 1,2,\ldots,j-1$.

- Once $\Omega_K$ is estimated, the model is estimated by OLS and forecasts generated as in a simple predictive regressions.
- $K$ is the hyperparameter of the procedure.

# Generalized linear model

- If the true data generating process is not linear in the variables, but possibly still in the parameters, we may expand the set of predictor by including higher order transforms of them.
- This can be done in numerous ways - here we will consider the most simple way that just computes the square and cubes (or possibly higher orders) of the original predictors.
- However, this proliferates parameters, and requires an additional layer of regularization should be added, here using the ENet as above.
- Define $z^p$ (with abuse of notation) as the vector whose elements are each of $z$ raised to the $p$'th power.

## Generalized linear model

1. Model: The model has same format as the penalized linear regression model by including additively the higher-order transformations of the predictors as

$$g(z_{it};\theta) = z_{it}'\theta^{(1)} + z_{it}^{2'}\theta^{(2)},\ldots,z_{it}^{p'}\theta^{(p)}, \tag{13}$$

   with $\theta = (\theta^{(1)},\theta^{(2)},\ldots,\theta^{(p)})'$.

2. The objective function: The objective function has same format as the penalized linear model as per

$$\tilde{\mathcal{L}}(\theta;\lambda,\alpha) = \mathcal{L}(\theta) + \phi(\lambda,\alpha), \tag{14}$$

   with the ENet penalty term applied.

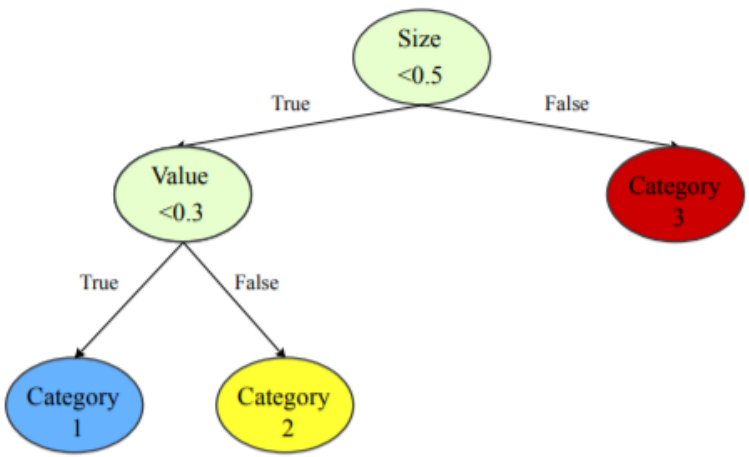
Gu et al. (2020) applies spline function expansions and a related penalty term that are out of the scope here.

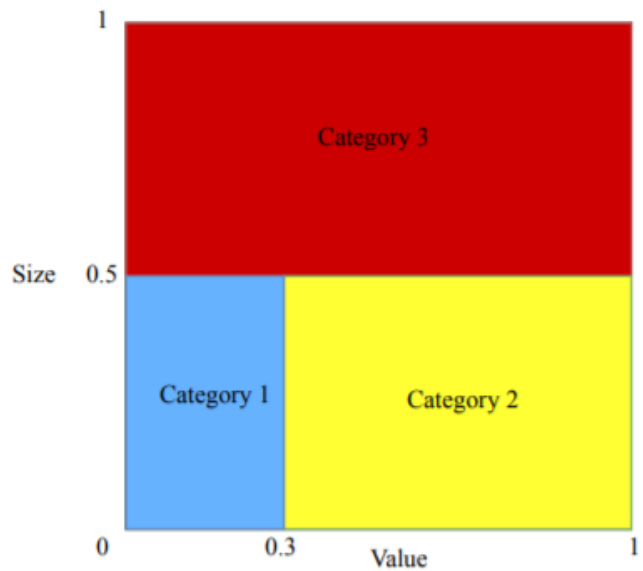
# Regression trees

- While the penalized linear model, PCR, and PCA only allow for linear effects of the predictor, the generalized linear model allow for higher-order effects.
- However, there is no way the generalized linear model would be well-specified if we allow it to include interactions as it increases parameters combinatorially.
- As an alternative, regression trees have become a popular ML technique for capturing these types of non-linearities.
- These are fully nonparametric and is fundamentally very different from the linear regression models seen so far.

# Regression trees

- Trees are designed to find groups of observations that behave similarly.
- Trees are grown sequentially, where at each step data is sorted into bins based on one of the predictor variables.
- This sequential branching slices the space of predictors into rectangular partitions, which approximates the unknown function $g^*(\cdot)$ with the average value of the outcome variable within each partition.




Note: This figure presents the diagrams of a regression tree (left) and its equivalent representation (right) in the space of two characteristics (size and value). The terminal nodes of the tree are colored in blue, yellow, and red, respectively. Based on their values of these two characteristics, the sample of individual stocks is divided into three categories.

## Regression trees

1. Model: The model or prediction of a tree $\mathcal{T}$ with $K$ leaves (terminal nodes - we had three in the example on former slide), and depth $L$ can be written as

$$g(z_{it};\theta,K,L) = \sum_{k=1}^{K} \theta_k 1_{\{z_{it} \in C_k(L)\}}, \tag{15}$$

   where $C_k(L)$ is one of the $K$ partitions (or regions) of data.

- Note that each partition is a product of up to $L$ (depth) many indicator functions of the predictions to move down the tree.
- The constant associated with the $k$'th partition, $\theta_k$, is the sample average of outcomes within the partition.

# Regression trees

- In the figure on the former slide, the prediction partition is

$$\begin{aligned} g(z_{it};\theta,3,2) = \theta_1 1_{\{\text{size}_{it}<0.5\}} 1_{\{\text{bm}_{it}<0.3\}} \\ + \theta_2 1_{\{\text{size}_{it}<0.5\}} 1_{\{\text{bm}_{it}\geq 0.3\}} \\ + \theta_3 1_{\{\text{size}_{it}\geq 0.5\}} \end{aligned} \tag{16}$$

- We will not look into the procedures to grow/estimate trees now other than saying they are grown to find the bins that best discriminate among the potential outcomes.
- However, trees are very prone to overfit and some regularization is needed to manage their flexibility.
- It can approximate potentially severe nonlineraities - for instance, a tree of depth $L$ can accommodate up to $(L-1)$-way interactions or very high orders of data!

# Regression trees

- We will consider two ways to regularize trees through a concept known as ensemble essentially a combination of models/predictions.

## Boosting ensemble learning

- Boosting combines forecasts from many over-simplified trees to form stable and more accurate (less biased) predictions.
- The procedure goes as follows:
  1. Fit a shallow tree (e.g. depth $L = 1$).
  2. Compute residuals and fit those with a second shallow tree of same depth.
  3. Add those two predictions together to form an prediction, but weight/shrink the prediction from the second tree by a factor (known as the learning rate) $\nu \in (0,1)$.
  4. At each new step, fit a shallow tree to the fitted residuals from the model of the prior step, and its predicted residuals are added to the total with a shrinkage $\nu$.
  5. Iterate for $B$ many ensemble trees.
- Tuning parameters are thus $L$,$\nu$,$B$.

## Random forests

- Random forests (RF) also combines forecasts from many different trees, yet it uses a concept known as bagging.
- The baseline tree bagging procedure draws $B$ different bootstrap samples of the data, fits a separate tree to each, and then average their forecasts.
- Trees for individual bootstrap samples tend to be deep and overfit, yet averaging stabilizes the performance and cancels out the fitting to noise.
- RF use a variant that aims at de-correlating trees to obtain larger benefits from the bagging procedure.

## Random forests (cont'd)

- For instance, if firm size is the dominant return predictor in data, most of the bagged trees will have shallow splits on size, resulting in substantial correlation between trees and their forecasts.
- RF de-correlates trees by dropout, which considers only a randomly drawn subset of predictors for splitting at each potential branch.
- This ensures that at least some trees will split ***not*** on size, lowering the correlation among trees and their output, which in turn lowers the variance of the prediction (averages out noise fitting).
- Now the depth $L$ and number of trees $B$ in the bagging are tuning parameters - in fact, so is the number of subset predictors at each split (default is $P/3$ for regressions).

# Neural networks

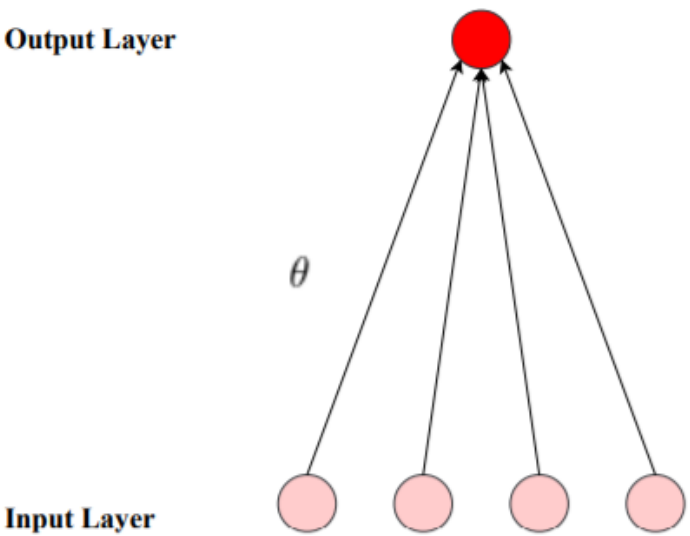
- Another nonlinear prediction method is the artificial neural networks (NN).
- They are typically also known as shallow or deep learning algorithms, being quite complex and typically the least interpretable among ML techniques.
- We focus here, as Gu et al. (2020), on traditional ***feed-forward networks***, which consist of an input layer of raw predictors, one or more hidden layers that interact and nonlinearly transform the predictors, and an output layer that aggregates hidden layers into an ultimate prediction.
- Each layer has a set of neurons, and those layers are connected by synapses hence the name neural network.

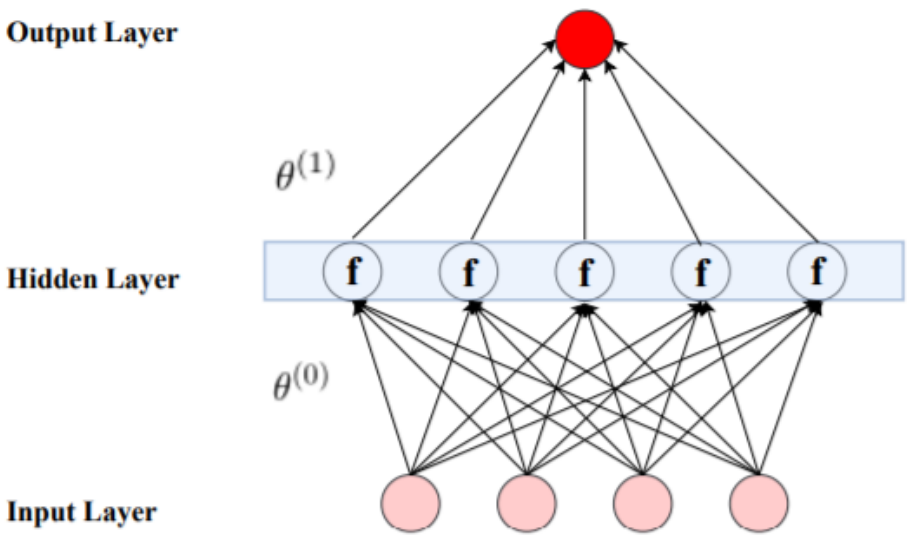
# Neural networks

- The number of units in the input layer is equal to the dimension of the predictors (the figure has 4).
- The left panel has no hidden layer and simply aggregates predictors into a prediction via

$$\theta_0 + \sum_{k=1}^{4} \theta_k z_k. \tag{17}$$

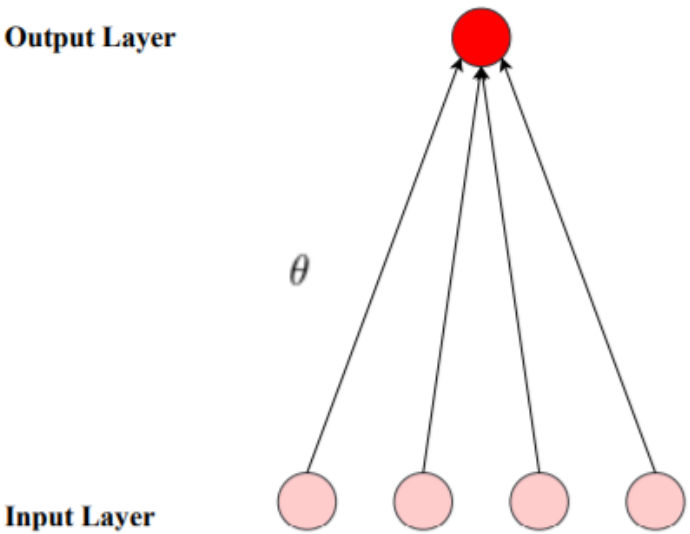
- …which is, indeed, just a linear regression model.

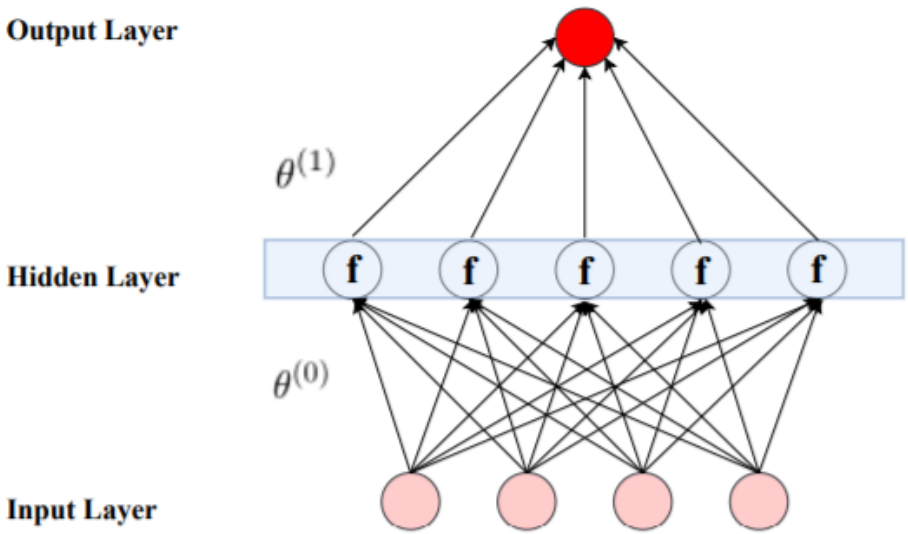


# Neural networks

- The right panel has one hidden layer, with each of the five neurons linearly drawing information from all predictors (like the left panel).
- Each neuron applies a nonlinear activation function $f$ that transforms the input before sending its output to the next layer.



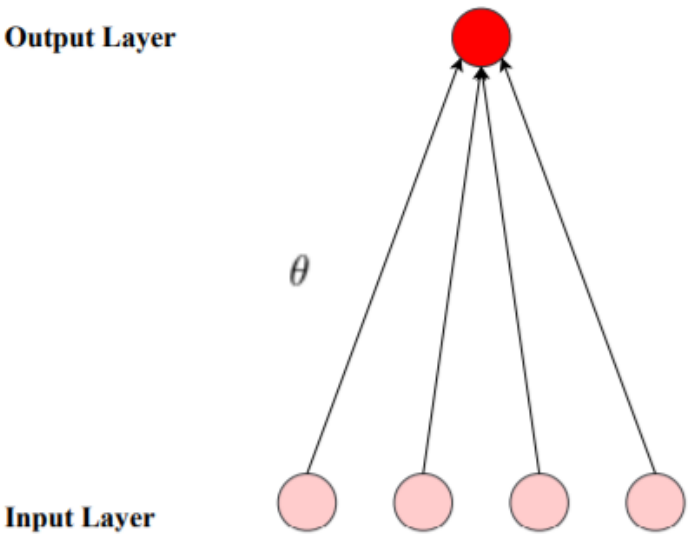
# Neural networks

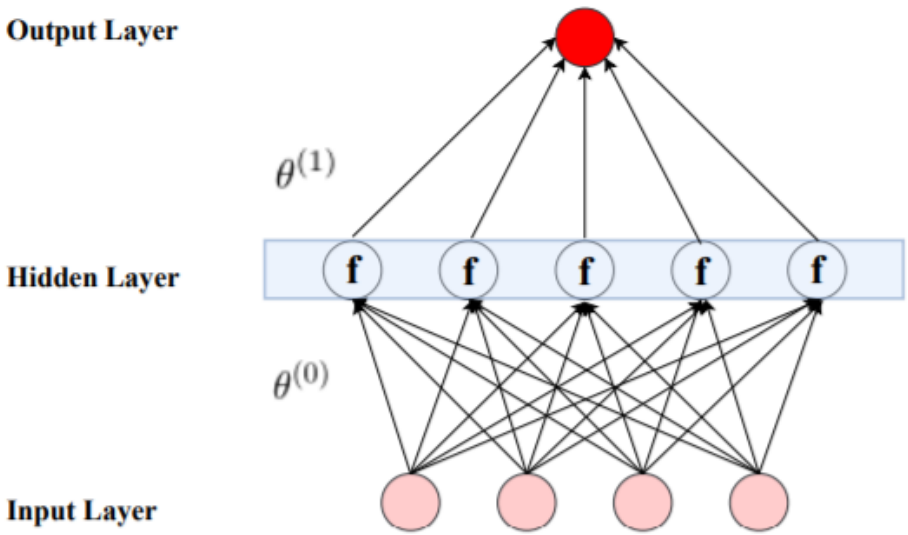
- For instance, the second neuron in the hidden layer transforms inputs into an output as

$$x_2^{(1)} = f\left(\theta_{2,0}^{(0)} + \sum_{k=1}^{4} \theta_{2,k}^{(0)} z_k\right), \tag{18}$$

which in the output layer is aggregated (using input from all neurons) to a final prediction

$$g(z;\theta) = \theta_0^{(1)} + \sum_{j=1}^{5} \theta_j^{(1)} x_j^{(1)}. \tag{19}$$



# Neural networks

- In this example, we have $(4+1)\cdot 5 + (5+1) = 31$ parameters with only four predictors, one hidden layer with five neurons!
- An NN that has few (or just one) hidden layers is referred to as shallow, whereas several layers defines deep learning.
- There are **soooo**...many choices to make in designing NNs - Gu et al. (2020) make several and we will not dwell with them here, but you should be aware of those when implementing the method.
- As with the tree-based methods, we will not consider the estimation of NNs, but focus on its implication for the functional form of expected excess returns for now.

# Targeting predictors

- You can naturally combine different methods such as targeting which simply introducing a selection step before training a specific model
- Dong et al. (2022) apply ENet before combining forecasts to predict market premium using anomalies (works well and is replicable)
- Borup et al. (2020a) apply ENet before a random forest

# Gu, Kelly and Xiu

# Overall comments

- We will now go through the paper of Gu et al. (2020)
- The paper is very well-written, and provides a good illustration of how ML can be used to predict returns
- Instead of trying to replicate the findings of Gu et al. (2020), we will take the findings as given focus on how they interpret their results

# Data: stock returns

- Monthly CRSP data from 1957 to end of 2016
- The total number of stocks is almost 30,000 and the average number of stocks per month exceeds 6,200.
- The monthly T-bill is used as proxy for the risk-free rate, used to compute excess returns.

# Data: predictors

- 94 stock-level predictive characteristics, some of which are updated annually, quarterly, and monthly, obtained from the joint CRSP-Compustat database.
- 74 industry dummies using SIC codes.
- 8 macro-financial predictors common to all stocks from Welch and Goyal (2008) (e.g. dividend-price ratio, term spreads, default spread, and stock market variance) obtained from Goyal's personal website.

$\Rightarrow$ totalling 102 predictors, excluding the 74 industry dummies, without any non-linear effects.

# Overarching predictive model

- All the ML techniques we will consider are designed to approximate the overarching model

$$\mathbb{E}_t[r^e_{it+1}|\mathcal{F}_t] = g^*(z_{it}) \tag{20}$$

  defined above.
- Denote the $P_c \times 1$ stock-level predictors by $c_{it}$ for each $i$, and the $P_x \times 1$ macro-financial predictors by $x_t$.
- Gu et al. (2020) then consider the total set of, now, stock-level set of predictors as per

$$z_{it} = c_{it} \otimes (1 \;\; x_t), \tag{21}$$

  which is $P_c \cdot (P_x + 1) \times 1$-dimensional and includes all characteristics and their interactions with the common predictors as input.

$\Rightarrow$ totalling $94 \cdot (8 + 1) = 846$ predictors, excluding the industry dummies.

# Overarching predictive model

- To see the relationship to standard $\beta$ representation models of asset pricing, we may write the conditional model with both time-varying $\beta$s and risk premia $\gamma$s as

$$\mathbb{E}_t[r^e_{it+1}] = \beta_{it}\gamma_t. \tag{22}$$

- Here, stock-level information enters $\beta$ (which is, indeed, stock-specific) and allows aggregate, common economic information through the dynamic risk premia $\gamma_t$.
- Suppose we have $K$ many factors, as usual, and let $\beta_{it} = \theta_1 c_{it}$ and $\gamma_t = \theta_2 x_t$ for some constant parameter matrices $\theta_1(K \times P_c)$ and $\theta_2(K \times P_x)$.

# Overarching predictive model

- The $\beta$ representation of the asset pricing model is the implied as per

$$\begin{aligned} g^*(z_{it}) = \mathbb{E}_t[r^e_{it+1}] = \beta_{it}\gamma_t &= c'_{it}\theta'_1\theta_2 x_t \\ &= (c_{it} \otimes (1 \;\; x_t))'\text{vec}(\theta'_1\theta_2) \\ &\equiv z'_{it}\theta, \end{aligned} \tag{23}$$

  where $\theta = \text{vec}(\theta'_1\theta_2)$.
- As such, the overarching model can be seen within the asset pricing framework.
- Yet, the ML techniques are not restricted to be linear in parameters as used in this example, allowing for a variety of transformations of the $z_{it}$ predictor set.

# Sample split

- The total sample period is (initially) split as follows

$$\underbrace{1957M1,\ldots,1974M12,}_{\text{training}}\underbrace{1975M1,\ldots,1986M12,}_{\text{validation}}\underbrace{1987M1,\ldots,2016M12,}_{\text{testing}}$$

  using the hybrid scheme between recursively expanding the training period, pushing forward the fixed length validation period to generate forecasts in the testing period.
- Since ML techniques are computationally intensive, they only re-estimate models once a year and not every month.
- The forecast horizon is set to one month or a year.

# Performance evaluation

- To assess the predictive ability of the ML techniques across all individual stock excess returns, Gu et al. (2020) use a modified version of the out-of-sample $R^2$ that you saw in the lectures on return predictability.
- This is defined as

$$R^2_{\text{OS}} = 1 - \frac{\sum_{t=R_2+1}^{T} \sum_{i=1}^{N_{\text{test},t}} (r^e_{it} - \hat{r}^e_{it})^2}{\sum_{t=R_2+1}^{T} \sum_{i=1}^{N_{\text{test},t}} (r^e_{it})^2}. \tag{24}$$

  where $N_{\text{test},t}$ is the number of stocks available at time $t$.
- This metric is taken only over the testing period.
- Note that the denominator is the sum of squared returns only and does *not* subtract the historical mean, as you have seen in previous lectures.

# Performance evaluation

- While subtracting the historical average is common and defines a natural benchmark, Gu et al. (2020) argues that it is flawed for individual stocks.
- They, instead, prefer using zero returns as a natural benchmark.
- The reason is that for individual stocks, the historical average is simply too noisy, being too sensitive to idiosyncratic movements.
- As such, it is so bad that it artificially lowers the bar for predictability.
- All the following results are generally shifted upwards by 3 percentage points when the “usual" $R^2_{\text{OS}}$ is applied.

# Performance evaluation

- To assess pairwise performance statistically, Gu et al. (2020) use a version of the Diebold-Mariano test.
- It is a cross-sectionally averaged version of the Diebold-Mariano test you have seen in previous lectures.
- The idea is that we construct a loss differential across all stocks under consideration, making a single time series.
- We then form a natural $t$-statistic upon that.

# Performance evaluation

## Aggregate Diebold-Mariano test

- To test the forecast performance of method 1 versus method 2, we define

$$
\begin{aligned}
d_{12t} &= \frac{1}{N_{\text{test},t}} \sum_{i=1}^{N_{\text{test},t}} (r_{it}^e - \hat{r}_{1it}^e)^2 - (r_{it}^e - \hat{r}_{2it}^e)^2 \\
&= \frac{1}{N_{\text{test},t}} \sum_{i=1}^{N_{\text{test},t}} \hat{e}_{1it}^2 - \hat{e}_{2it}^2, \qquad (25)
\end{aligned}
$$

  which is the average loss differential across all stocks at time $t$ (within the testing period).

[...]

## Aggregate Diebold-Mariano test (cont'd)

[...]

- This is, as such, a time series of average loss differentials and we can then test $\mathbb{H}_0 : \mathbb{E}[d_{12t}] = 0$ by running the regression

$$d_{12t} = \psi + \varepsilon_{12t} \tag{26}$$

  and performing a standard $t$-test on the constant $\psi$ using Newey and West (1987) HAC standard errors to evaluate the null $\psi = 0$.
- This directly takes into account cross-sectional correlation among stocks.

## Aggregate Diebold-Mariano test (cont'd)

[...]

- This is equivalent to building the following test statistic

$$t(\bar{d}_{12}) = \frac{\bar{d}_{12}}{\sqrt{\text{Var}[\bar{d}_{12}]}} \xrightarrow{d} N(0,1), \tag{27}$$

where, $T_{\text{test}}$ being the length of the testing period,

$$\bar{d}_{12} = \frac{1}{T_{\text{test}}} \sum_{t=R_2+1}^{T} d_{12t}, \tag{28}$$

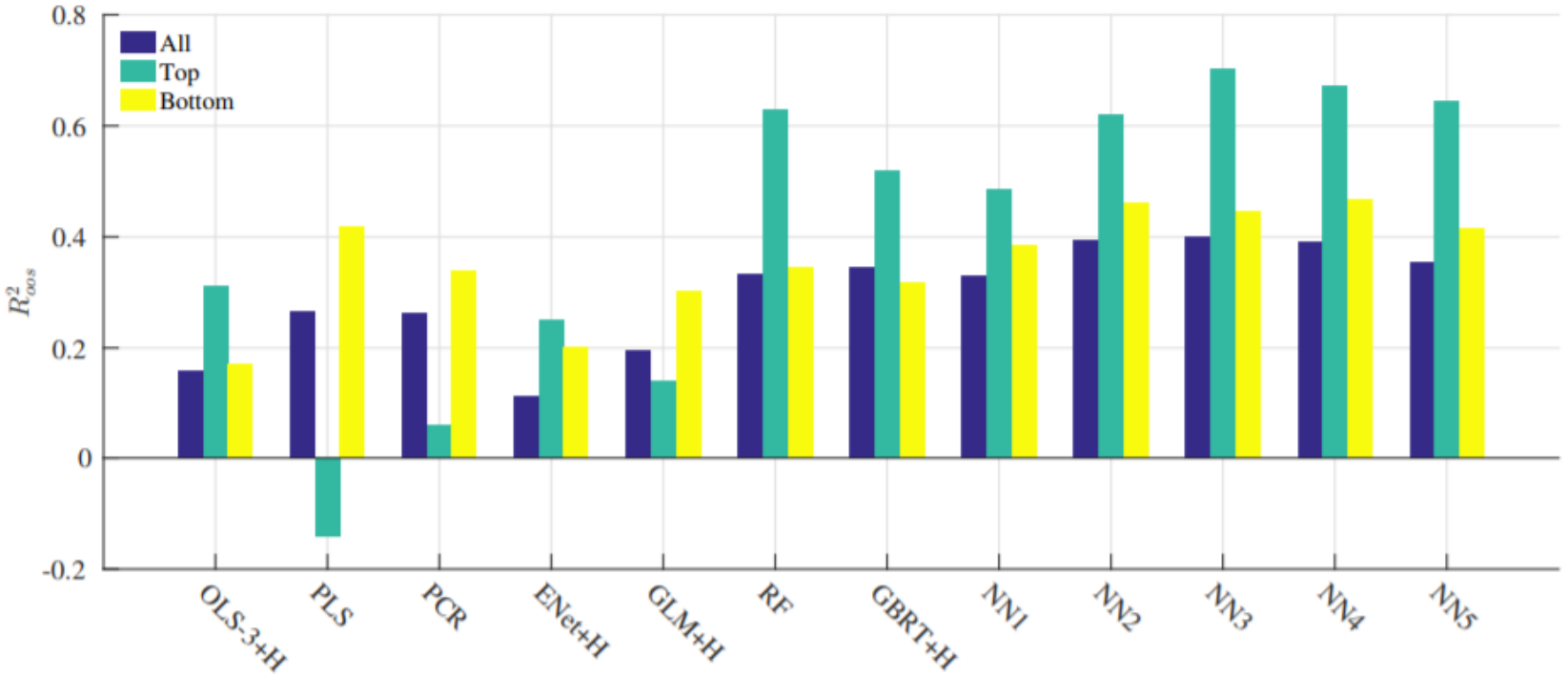
and $\text{Var}[\bar{d}_{12}]$ computed using the classical HAC estimator.

# Forecasting Accuracy

# Results for the cross-section of individual stocks

Fig. 1: Monthly out-of-sample predictability

| | OLS +H | OLS-3 +H | PLS | PCR | ENet +H | GLM +H | RF | GBRT +H | NN1 | NN2 | NN3 | NN4 | NN5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All | -3.46 | 0.16 | 0.27 | 0.26 | 0.11 | 0.19 | 0.33 | 0.34 | 0.33 | 0.39 | 0.40 | 0.39 | 0.36 |
| Top 1000 | -11.28 | 0.31 | -0.14 | 0.06 | 0.25 | 0.14 | 0.63 | 0.52 | 0.49 | 0.62 | 0.70 | 0.67 | 0.64 |
| Bottom 1000 | -1.30 | 0.17 | 0.42 | 0.34 | 0.20 | 0.30 | 0.35 | 0.32 | 0.38 | 0.46 | 0.45 | 0.47 | 0.42 |




Note: In this table, we report monthly $R^2_{oos}$ for the entire panel of stocks using OLS with all variables (OLS), OLS using only size, book-to-market, and momentum (OLS-3), PLS, PCR, elastic net (ENet), generalize linear model (GLM), random forest (RF), gradient boosted regression trees (GBRT), and neural networks with one to five layers (NN1–NN5). "+H" indicates the use of Huber loss instead of the $l_2$ loss. We also report these $R^2_{oos}$ within subsamples that include

# Results for the cross-section of individual stocks

## Summary of results

- OLS with all predictors are handily dominated by the naive benchmark of zero returns ($R^2_{\text{OS}} < 0$)
- Restricting OLS to a few variables or using penalization (ENet) improves predictability substantially.
- Dimension reduction via PCR or PLS improves predictability further.
- This suggests that single predictors (characteristics) are particularly redundant and fundamentally noisy signals, yet combining then into low-dimensional components averages out noise to better reveal their correlated signals.

# Results for the cross-section of individual stocks

## Summary of results (cont'd)

- The generalized linear model (that is linear in parameters, but allow for non-linear transformations of variables, yet no interactions) provides no substantial improvement of OLS-3 or ENet.
- Boosted trees and random forests are slightly outperforming PCR and PLS.
- Neural networks are the best performing non-linear method, and the best overall.
- This indicates the value of incorporating complex predictor interactions which are embedded in tree and neural network models, but missed by other techniques.
- Deep learning does not provide much, as NN4 and NN5 fails to improve over NN3.

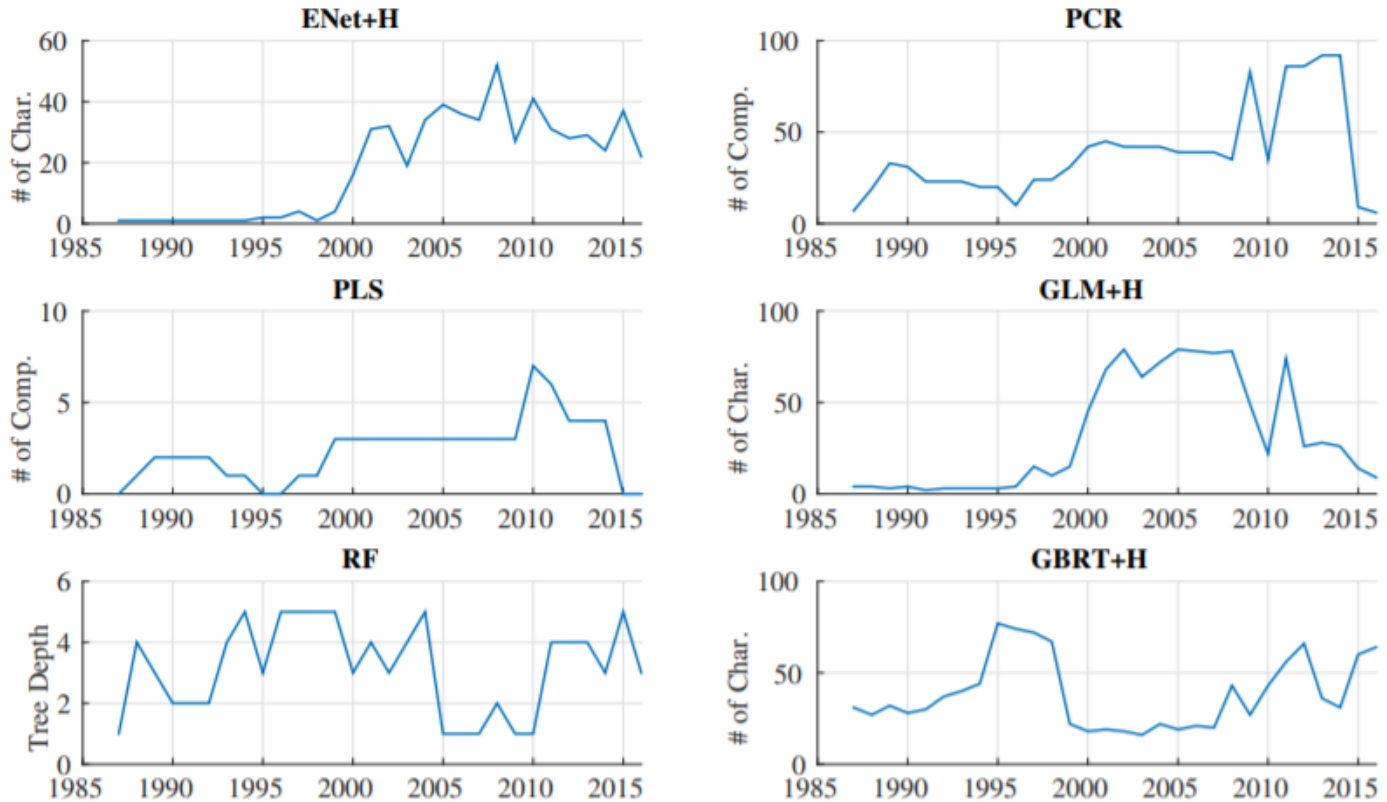
# Results for the cross-section of individual stocks

## Summary of results (cont'd)

- Separating predictability into that for the 1,000 largest and the 1,000 smallest firms based on market equity allows one to test whether predictability is driven by small, illiquid firms that are poorly priced in the financial market.
- The pattern from above remains in fact, RF and NN generates stronger performance for the largest firms and to a very large degree.
- Since the same pattern persists at annual forecasting horizon, it suggests that ML techniques are able to measure excess returns that persist over business cycle frequencies and not merely short-term inefficiencies.

# Results for the cross-section of individual stocks

Fig. 2: Time-varying model complexity




Note: This figure demonstrates the model complexity for elastic net (ENet), PCR, PLS, generalized linear model with group lasso (GLM), random forest (RF) and gradient boosted regression trees (GBRT) in each training sample of our 30-year recursive out-of-sample analysis. For ENet and GLM we report the number of features selected to have non-zero coefficients; for PCR and PLS we report the number of selected components; for RF we report the average tree depth; and for GBRT we report the number of distinct characteristics entering into the trees.

# Results for the cross-section of individual stocks

Fig. 3: Pairwise statistical comparison of methods

| | OLS-3 +H | PLS | PCR | ENet +H | GLM +H | RF | GBRT +H | NN1 | NN2 | NN3 | NN4 | NN5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OLS+H | **3.26*** | **3.29*** | **3.35*** | **3.29*** | **3.28*** | **3.29*** | **3.26*** | **3.34*** | **3.40*** | **3.38*** | **3.37*** | **3.38*** |
| OLS-3+H | | 1.42 | **1.87** | -0.27 | 0.62 | **1.64** | 1.28 | 1.25 | **2.13** | **2.13** | **2.36** | **2.11** |
| PLS | | | -0.19 | -1.18 | -1.47 | 0.87 | 0.67 | 0.63 | 1.32 | 1.37 | **1.66** | 1.08 |
| PCR | | | | -1.10 | -1.37 | 0.85 | 0.75 | 0.58 | 1.17 | 1.19 | 1.34 | 1.00 |
| ENet+H | | | | | 0.64 | **1.90** | 1.40 | **1.73** | **1.97** | **2.07** | **1.98** | **1.85** |
| GLM+H | | | | | | **1.76** | 1.22 | 1.29 | **2.28** | **2.17** | **2.68*** | **2.37** |
| RF | | | | | | | 0.07 | -0.03 | 0.31 | 0.37 | 0.34 | 0.00 |
| GBRT+H | | | | | | | | -0.06 | 0.16 | 0.21 | 0.17 | -0.04 |
| NN1 | | | | | | | | | 0.56 | 0.59 | 0.45 | 0.04 |
| NN2 | | | | | | | | | | 0.32 | -0.03 | -0.88 |
| NN3 | | | | | | | | | | | -0.32 | -0.92 |
| NN4 | | | | | | | | | | | | -1.04 |

Note: This table reports pairwise Diebold-Mariano test statistics comparing the out-of-sample stock-level prediction performance among thirteen models. Positive numbers indicate the column model outperforms the row model. Bold font indicates the difference is significant at 5% level or better for individual tests, while an asterisk indicates significance at the 5% level for 12-way comparisons via our conservative Bonferroni adjustment.

- To read the table, positive values indicate that the column method outperforms the row method.
- Bold values indicate significance at the 5% significance level.
- Asterisk adjusts for multiple testing via Bonferroni corrections.

# Results for the cross-section of individual stocks

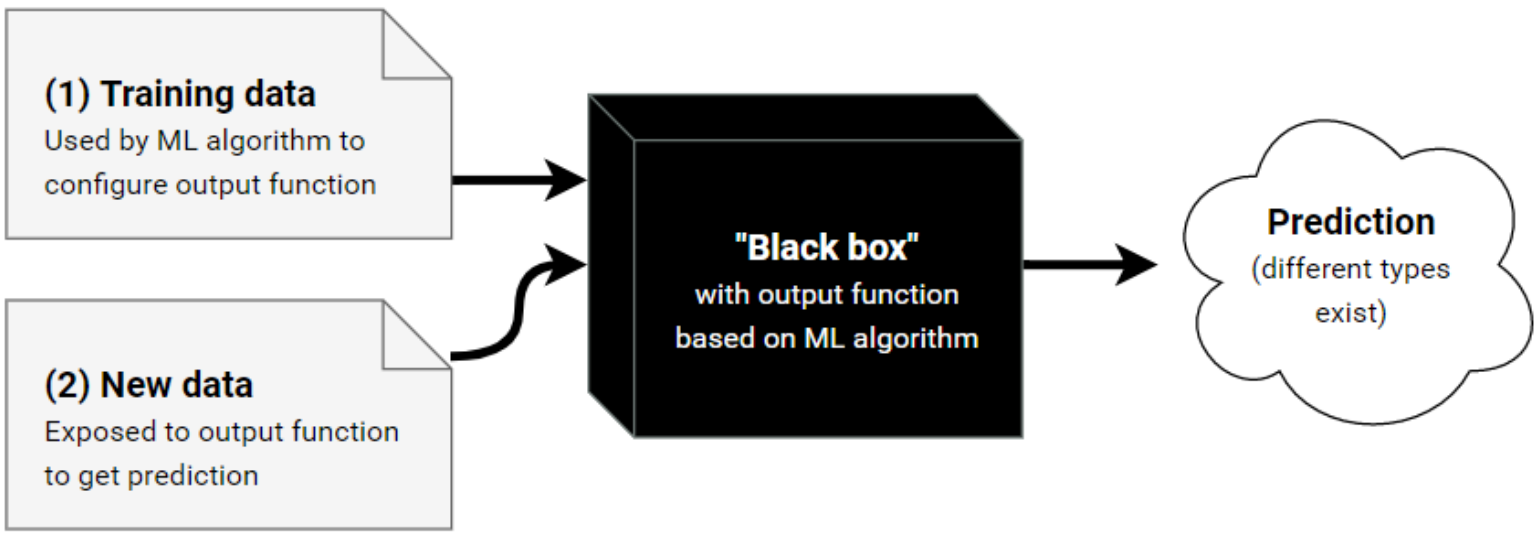
## Summary of results

- Constrained linear models - including OLS-3, ENnet, PCR and OLS - produces significant improvements over the unconstrained OLS.
- There is no statistical difference between the penalized OLS model and dimension reduction methods.
- Tree-based methods are marginally (at best) significantly superior to linear models.
- NN are the only models that produce large and significant improvements over linear (and generalized linear) models.
- . . . yet, they are not significantly better than tree-based methods.

# Interpretability

# Variable importance

- A very popular critique of ML techniques is that they are simply black box.




- Yet, with better understanding of the algorithms comes more interpretability.

# Variable importance

- There are several ways to obtain a sense of the importance of the variables, $VI_j$ used in the ML prediction.
- We consider two sorts for measuring $VI_j$:
  1. Reduction in $R^2_{OS}$ by setting all values of predictor $j$ to zero, holding the remaining model estimates fixed.
  2. Sum of squared partial derivatives (SSD),

$$SSD_j = \sum_{t=R_2+1}^{T} \sum_{i=1}^{N_{\text{test},t}} \left( \frac{\partial g(z;\theta)}{\partial z_j} \bigg|_{z_{it}} \right)^2, \tag{29}$$

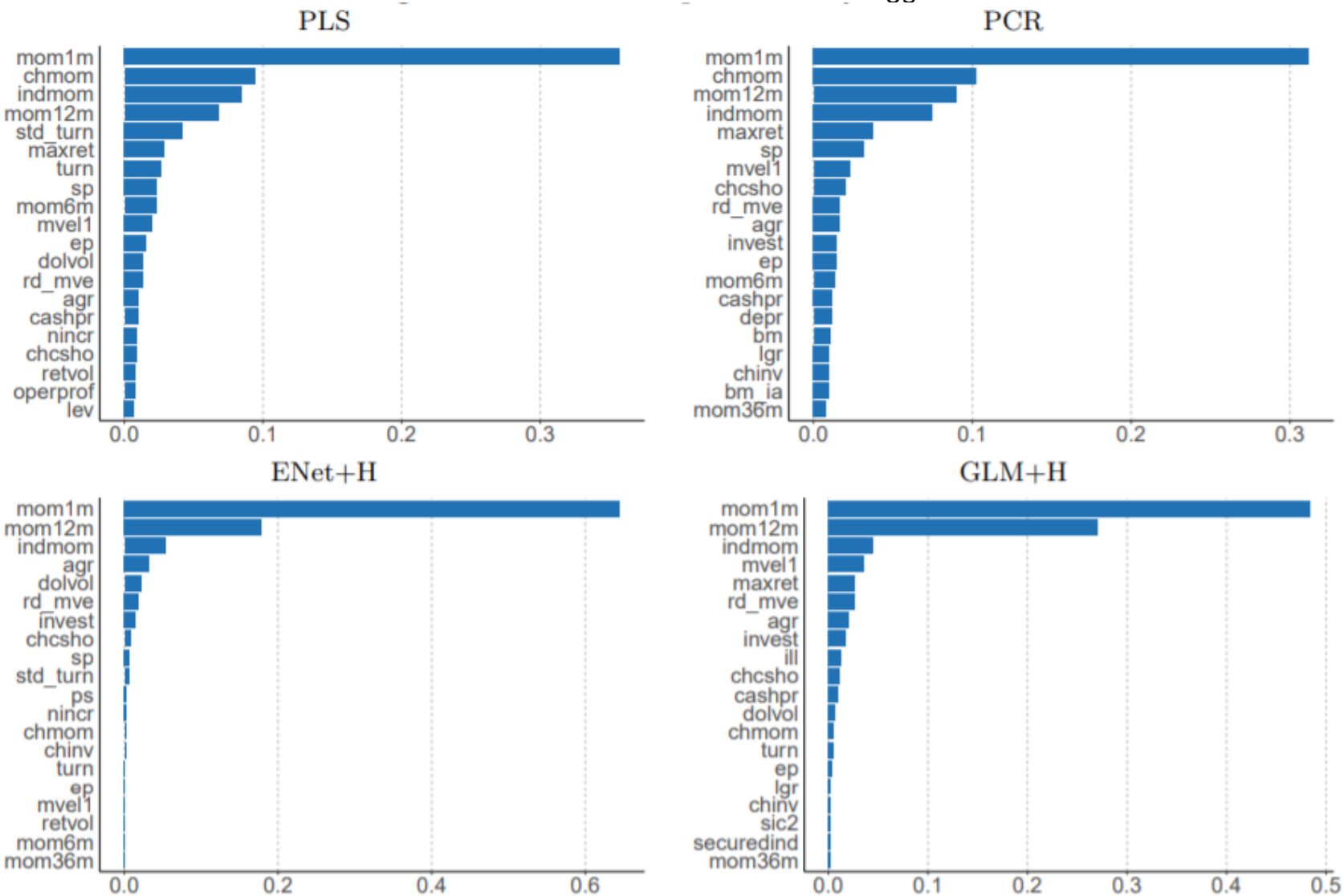
     which summarizes the sensitivity of the model fits to the changes in that variable.
- SSD is not defined for tree-based models (as they are not differentiable) in this case they use the reduction in impurity ($\approx$ MSE).

# A note on interpretability

- Here are some very useful references that provides model-agnostic methods:
  1. Interpretable machine learning by Christoph Molnar, link here: https://christophm.github.io/interpretable-ml-book/.
  2. Borup et al. (2020c) for application of partial dependency plots and associated variable importance the intuition is quite similar to approaches we will see below by Gu et al. (2020)

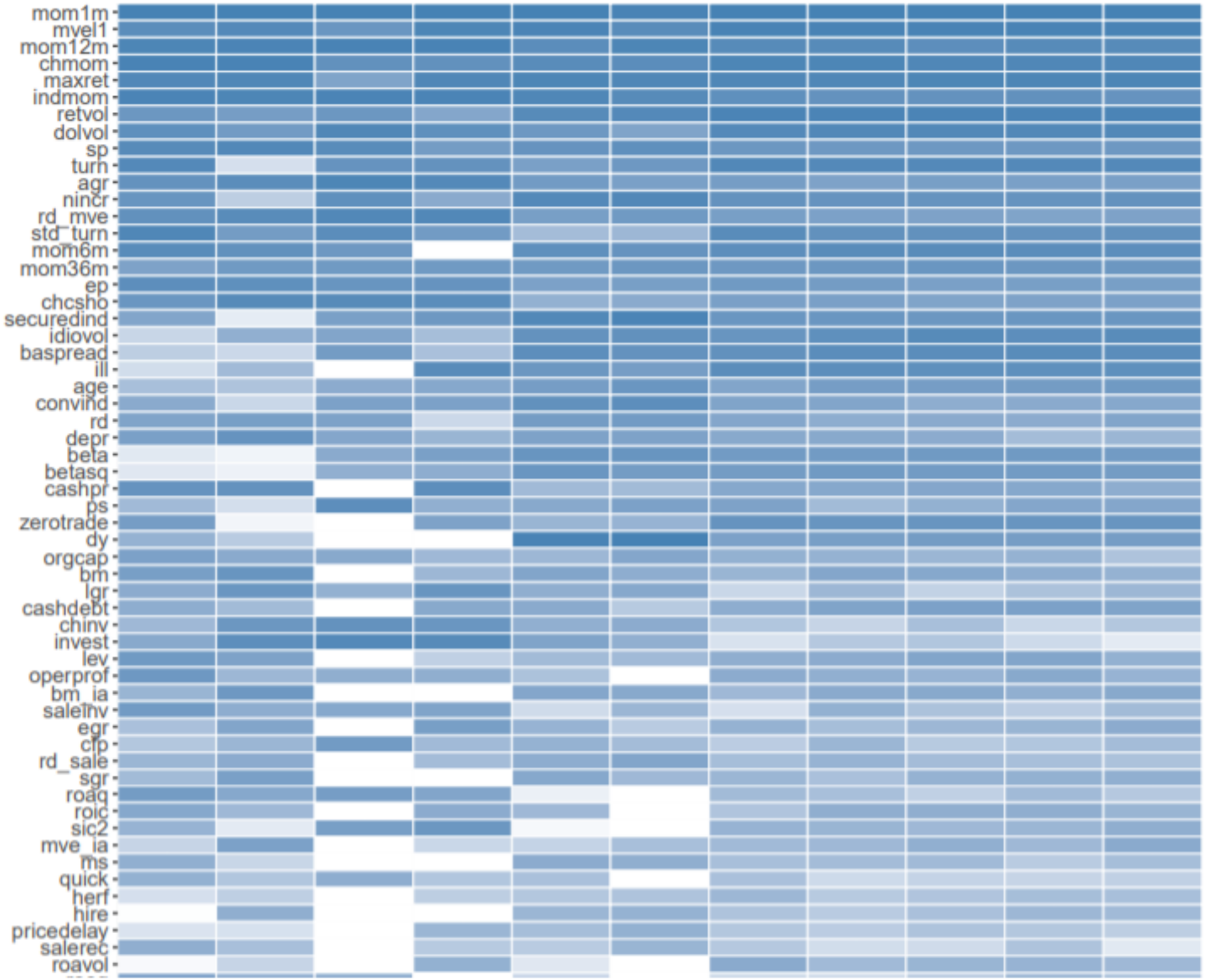
# Variable importance

Fig. 4: Variable importance (by $R^2_{\text{OS}}$)



# Variable importance

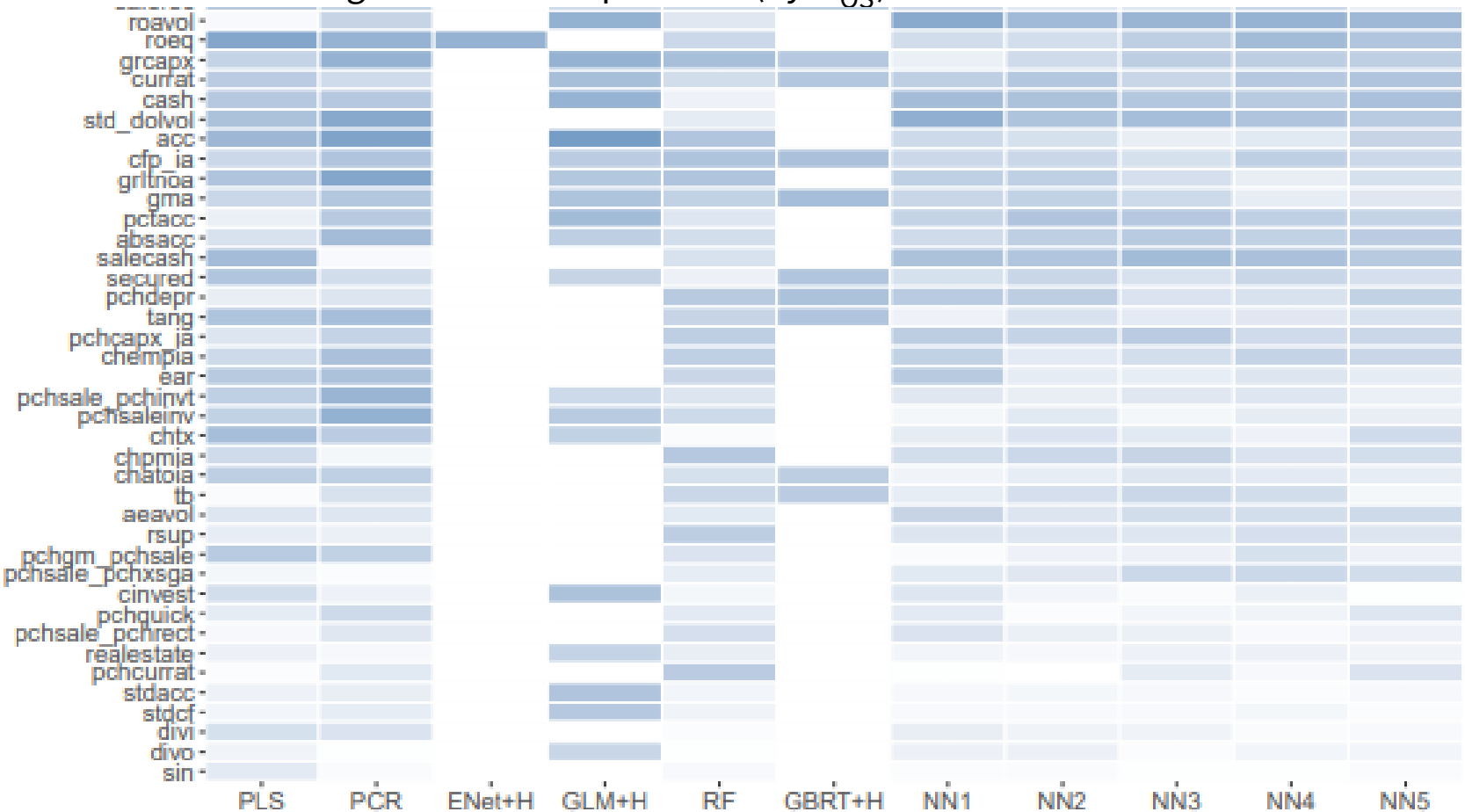
Fig. 5: Variable importance (by $R^2_{OS}$) across all models




Note: Rankings of 94 stock-level characteristics and the industry dummy (sic2) in terms of overall model contribution

# Variable importance

Fig. 6: Variable importance (by $R^2_{OS}$) across all models



# Variable importance

## Summary of results

- We may categorize the top predictors into four groups:
  1. Recent price trends: short-term reversal, stock momentum, momentum change, industry momentum, recent maximum return, long-term reversal.
  2. Liquidity variables: turnover, turnover volatility, log market equity, dollar volume, Amihud liquidity, number of zero trading days, bid-ask spread.
  3. Risk measures: total and idiosyncratic volatility, market beta, beta-squared.
  4. Valuation ratios and fundamental signals: earnings-to-price, sales-to-price, asset growth, no. of recent earnings increases.
- Results remain using the SSD measure for variable importance.

# Variable importance

- If the top predictors are truly informative, their ranking should be unaffected by the presence of irrelevant variables.

## The placebo principle

- To rule out that a given method or procedure mechanically generate the realized results, the placebo principle simulates irrelevant variables that share dynamics with the proposed relevant ones, but they are completely unrelated to the object of interest. Conducting the experiment as with the real data many times using the placebo variables generates an empirical distribution in which one can evaluate the realized values. If the empirical probability of significance is low of the realized values, there is little likelihood that results are spurious and they may be considered robust.
- Use of the placebo principle includes Gu et al. (2020), Borup and Schütte (2021), Borup and Schütte (2020), Delikouras and Kostakis (2019), Kelly and Pruitt (2013) in both cross-sectional asset pricing and (time series) return predictability.

# Variable importance

- We can simulate placebo characteristics that have no relation to returns by construction, but have similar properties as the true characteristics as per

$$c_{ijt}^{\text{placebo}} = \frac{2}{N+1}\text{CSrank}(\tilde{c}_{ijt}) - 1, \quad \tilde{c}_{itj} = \rho_j \tilde{c}_{ijt-1} + \epsilon_{ijt},$$

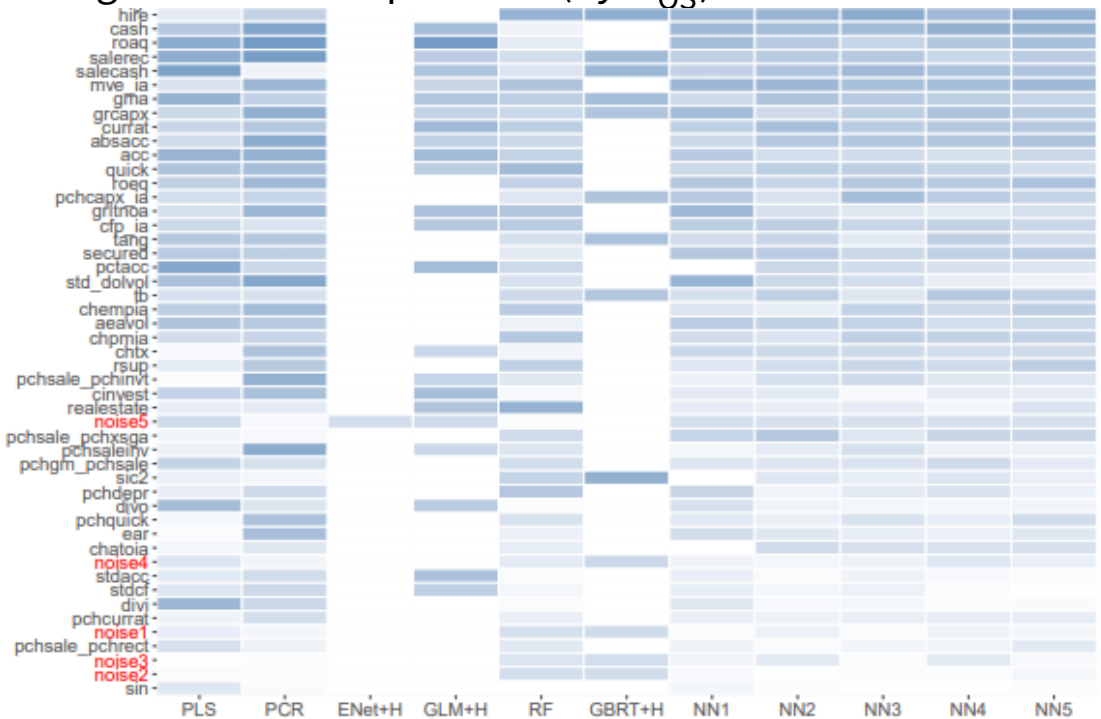
  where $\rho_j \sim U[0.9,1]$, $\tilde{c}$ is the actual value of the characteristic, and $\text{CSrank}(\cdot)$ is a cross-sectional rank function such that $c_{ijt}^{\text{placebo}} \in [-1,1]$.
- If we include those placebo variables in the prediction problem as additional predictors and compute the variable importance, we would appreciate them being ranked lowest without affecting the ranking of the remaining variables.

# Variable importance (top ones)

Fig. 7: Variable importance (by $R^2_{\text{OS}}$) across all models

# Variable importance (bottom ones)

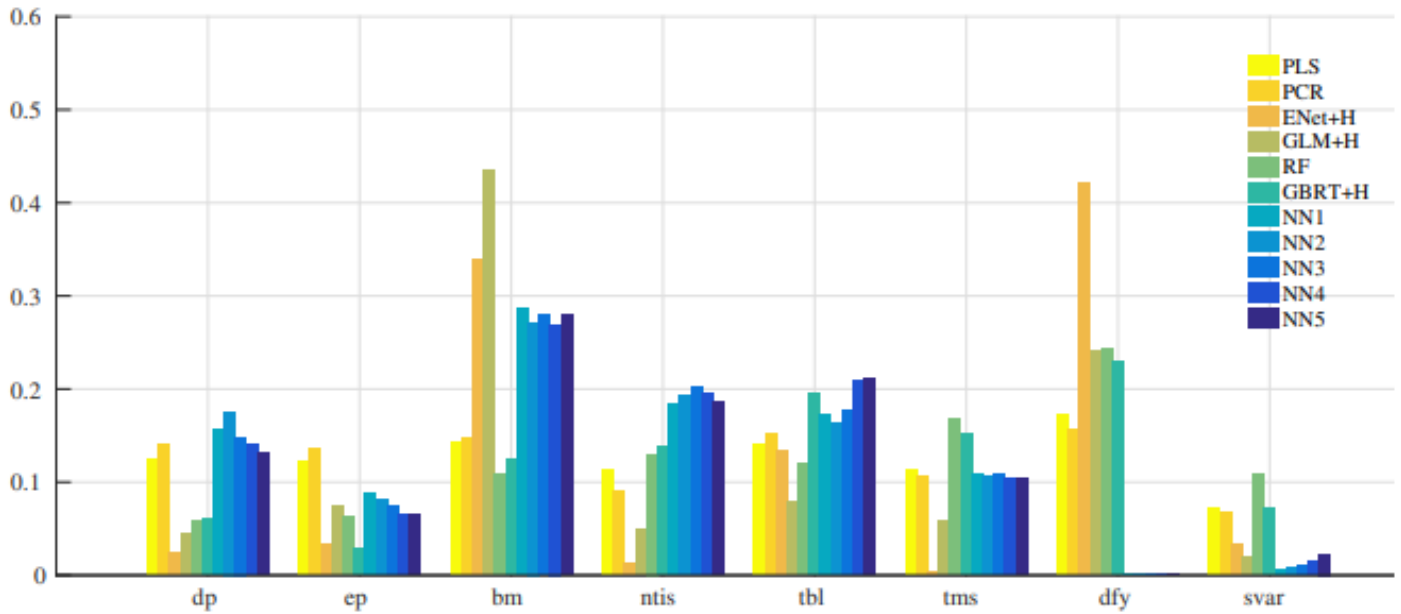
Fig. 8: Variable importance (by $R^2_{OS}$) across all models



# Importance of macro-financial variables

Fig. 9: Variable importance (by $R^2_{OS}$)

| | PLS | PCR | ENet+H | GLM+H | RF | GBRT+H | NN1 | NN2 | NN3 | NN4 | NN5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| dp | 12.52 | 14.12 | 2.49 | 4.54 | 5.80 | 6.05 | 15.57 | 17.58 | 14.84 | 13.95 | 13.15 |
| ep | 12.25 | 13.52 | 3.27 | 7.37 | 6.27 | 2.85 | 8.86 | 8.09 | 7.34 | 6.54 | 6.47 |
| bm | 14.21 | 14.83 | 33.95 | 43.46 | 10.94 | 12.49 | 28.57 | 27.18 | 27.92 | 26.95 | 27.90 |
| ntis | 11.25 | 9.10 | 1.30 | 4.89 | 13.02 | 13.79 | 18.37 | 19.26 | 20.15 | 19.59 | 18.68 |
| tbl | 14.02 | 15.29 | 13.29 | 7.90 | 11.98 | 19.49 | 17.18 | 16.40 | 17.76 | 20.99 | 21.06 |
| tms | 11.35 | 10.66 | 0.31 | 5.87 | 16.81 | 15.27 | 10.79 | 10.59 | 10.91 | 10.38 | 10.33 |
| dfy | 17.17 | 15.68 | 42.13 | 24.10 | 24.37 | 22.93 | 0.09 | 0.06 | 0.06 | 0.04 | 0.12 |
| svar | 7.22 | 6.80 | 3.26 | 1.87 | 10.82 | 7.13 | 0.57 | 0.85 | 1.02 | 1.57 | 2.29 |




Note: Variable importance for eight macroeconomic variables in each model. Variable importance is an average over all training samples. Variable importances within each model are normalized to sum to one. The lower panel provides a complementary visual comparison of macroeconomic variable importances.

# Variable importance

## Summary of results

- The book-to-market ratio is a critical predictor across all models.
- Market volatility has little role in any model.
- PCR and PLS puts some weight on all other variables in general, likely because they are highly correlated.
- Linear and generalized linear models strongly favour bond market variables, like the default spread and T-bill.
- Nonlinear models like RF and NN place great emphasis on exactly those ignored by linear models.

# Marginal association

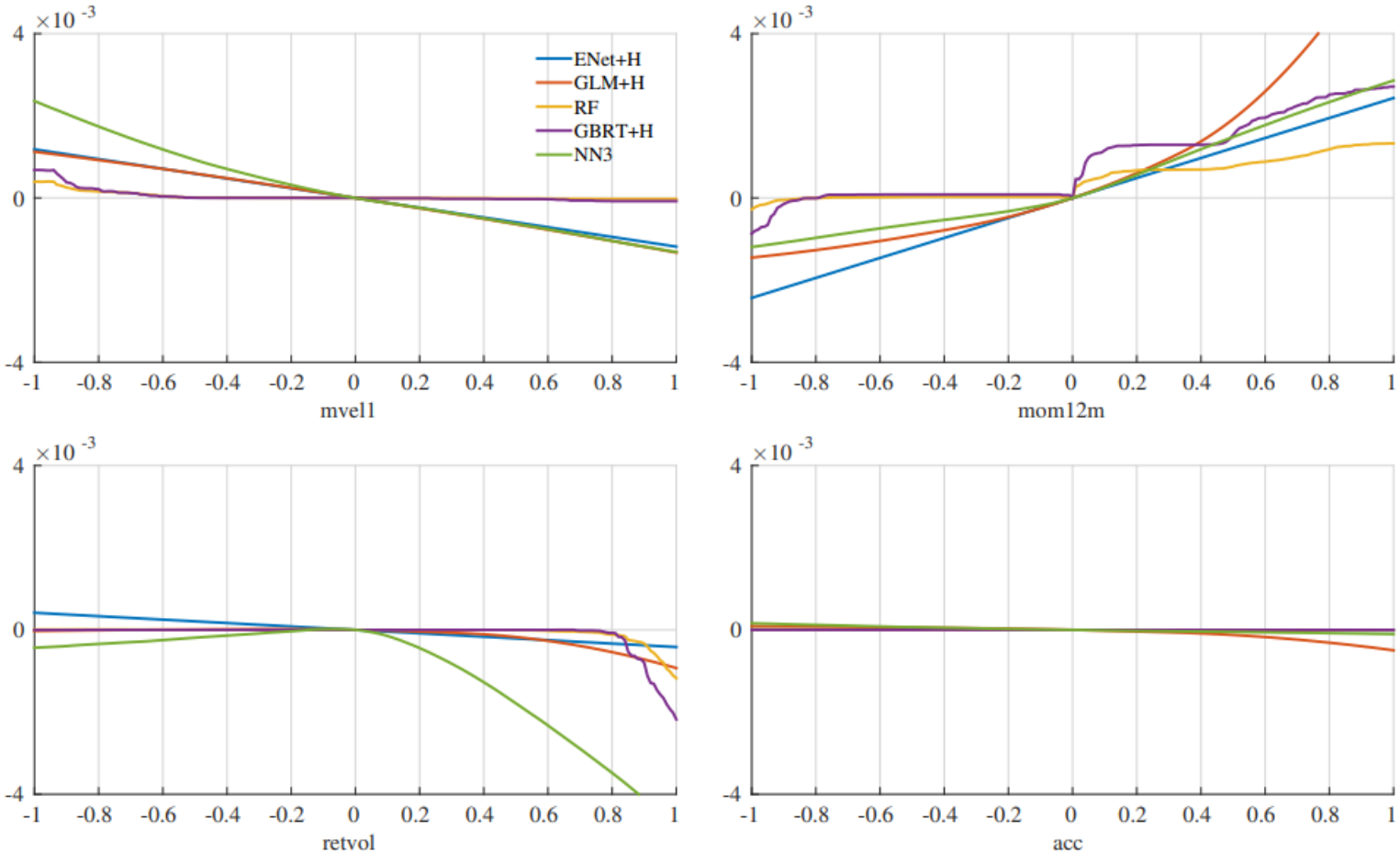
## Marginal impact of variable on expected returns

- To trace out the estimated relationship between certain variables and expected returns, Gu et al. (2020) fix all characteristics to their median values except the one variable one is interested in (e.g. size or volatility).
- One then varies the value of the variable under consideration from its minimum to its maximum (for characteristics this is between $[-1,1]$) and compute the fitted value of returns.
- Plotting all fitted values implied from using values in the support of the variables allows one to assess the marginal association.

- This has strong resemblance to partial dependency plots, see e.g. Borup et al. (2020c)

# Importance of macro-financial variables

Fig. 10: Marginal association




Note: Sensitivity of expected monthly percentage returns (vertical axis) to individual characteristics (holding all other covariates fixed at their median values).

# Marginal association

## Summary of results

- ML techniques generally show patterns that are consistent with some well-known phenomena.
- Expected returns are generally decreasing in size, increasing in past one-year returns, and decreasing in volatility.
- Penalized linear regressions (ENet) finds no association between size or volatility with expected returns, strongly opposing nonlinear methods that find large impacts of volatility.

# Marginal association

## Summary of results (cont'd)

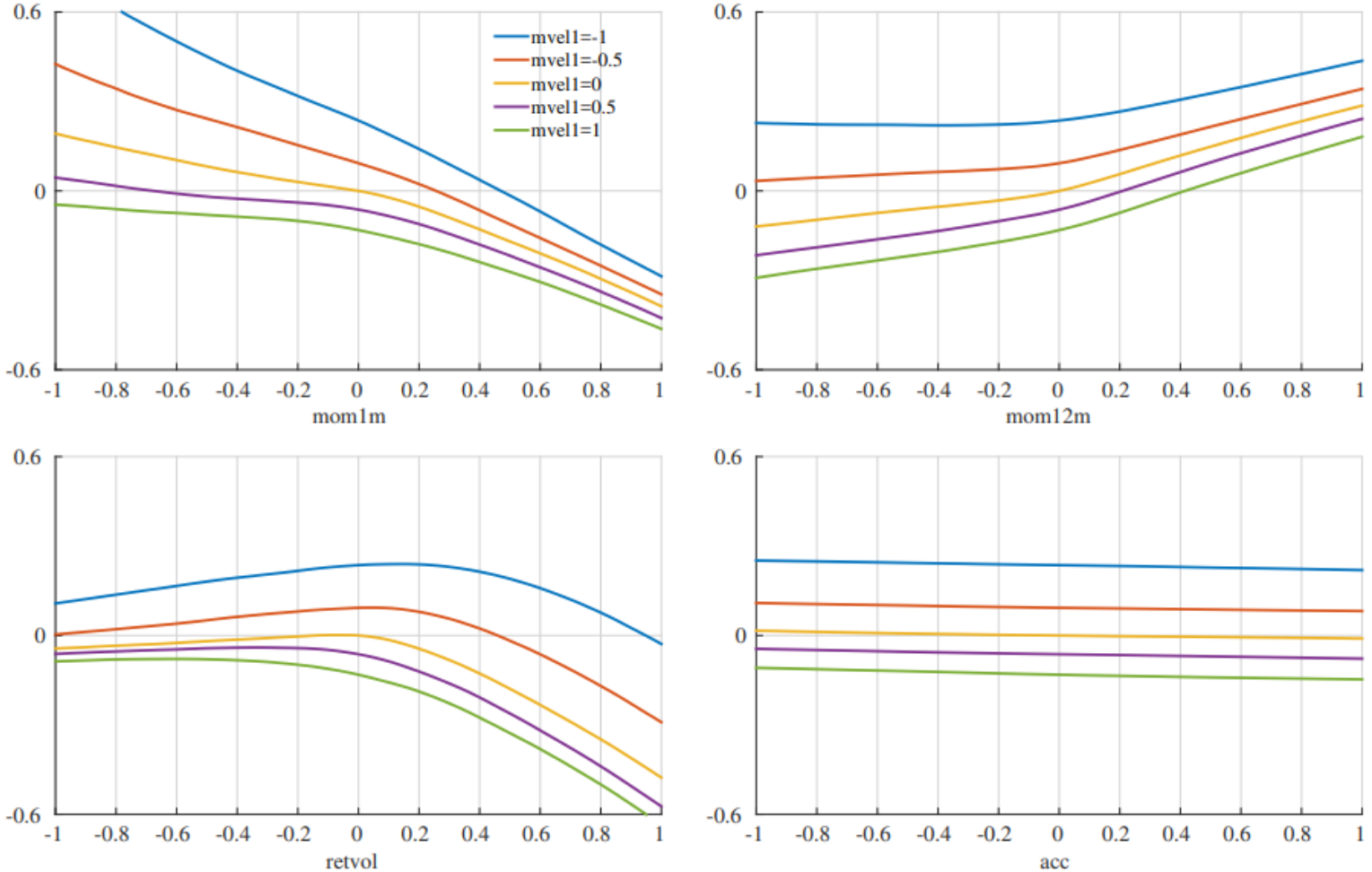
- For example, a firm that drops from the median to the 20th percentile of the size distribution has an increase in expected return (annualized) of approx. 2.4% according to the NN3.
- …and a firm whose volatility increases from the median to the 80th percentile experience an increase in expected return of approx. 3.0% (annualized), according to the NN3.
- Linear models cannot allow for any curvature on these marginal relationships their inability to capture nonlinearities can lead them to prefer a zero association instead, which might explain their differences in performance.

# Interaction effects

- The favourable performance of trees and neural networks indicates a benefit to allowing for potentially complex interactions among predictors.
- To understand those effects, we can do as above, yet now we simply vary two variables, simultanously, over their support $[-1,1]$, while still keeping all other variables at their median values.
- Gu et al. (2020) show results for the interaction between size and four variables, namely short-term reversal, momentum, total volatility, and accruals (note that Gu et al. (2020) has a small typo in their text as to which variables the figure includes).
- They also consider the interaction between size and total volatility with aggregated book-to-market ratios and net equity issuance.
- They are all based on NN3.

# Marginal association through interactions

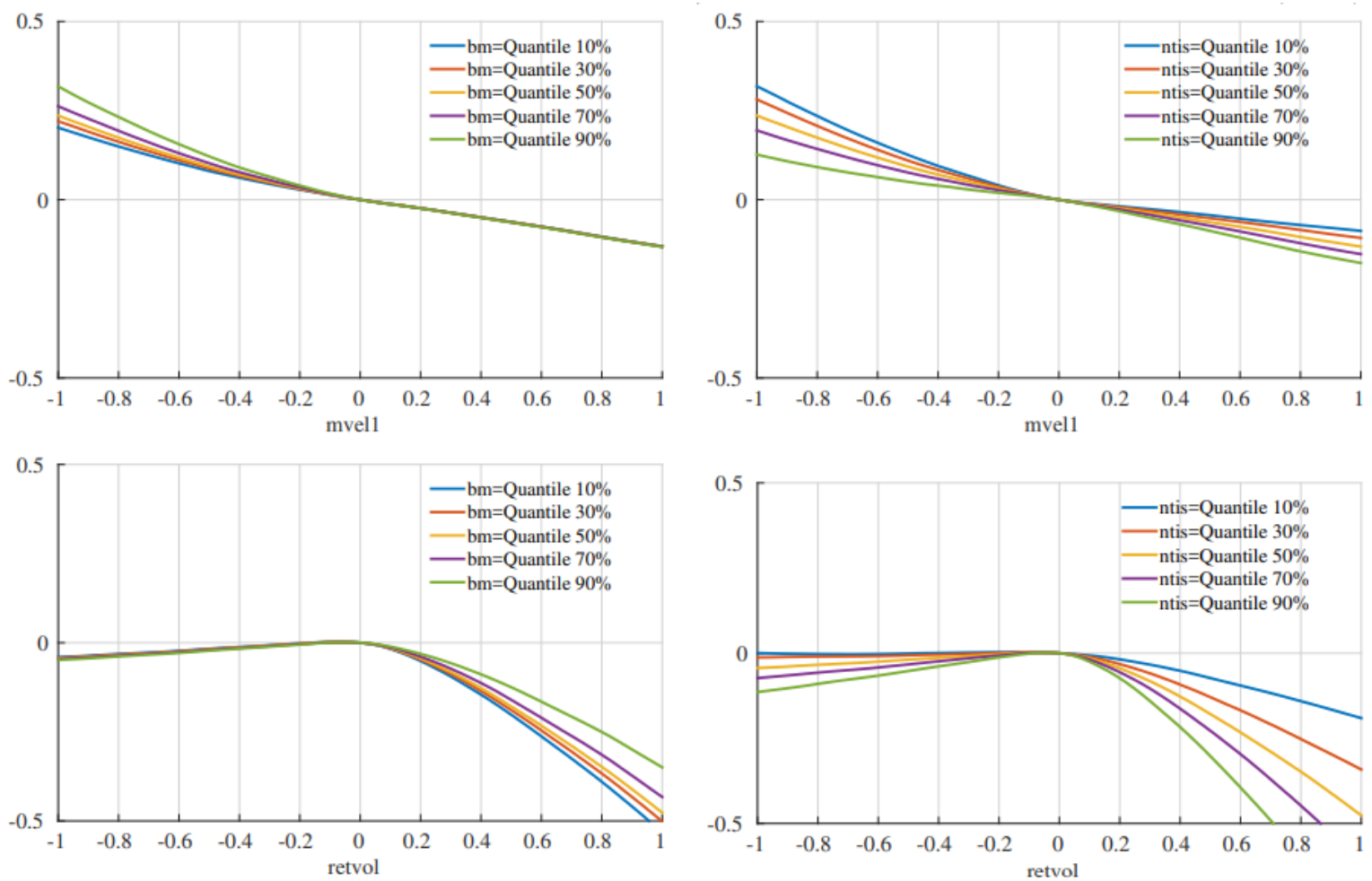
Fig. 11: Marginal association through interactions among characteristics




Note: Sensitivity of expected monthly percentage returns (vertical axis) to interactions effects for mvel1 with mom1m, mom12m, retvol, and acc in model NN3 (holding all other covariates fixed at their median values).

# Marginal association through interactions

Fig. 12: Marginal association through interactions among characteristics and macro-financial variable




Note: Sensitivity of expected monthly percentage returns (vertical axis) to interactions effects for mvel1 and retvol with bm and ntis in model NN3 (holding all other covariates fixed at their median values).

# Marginal association through interactions

## Summary of results

- There are clear interactions among size and short-term reversal, momentum, and total volatility, but none with accruals.
- There are clear interactions among size and aggregate valuations (book-to-market ratios) and equity issuance, and among total volatility and the former two variables.
- Gu et al. (2020) also find that the most important characteristic vs. macro-financial interactions come from interacting a stock's recent price trends (short-term reversal, momentum, industry momentum) with aggregate asset price levels (valuation ratios or T-bill rates).
- …and they are stable over time.

# Economic magnitude

# Machine learning portfolios

- Rather than assessing pre-specified portfolios, we may also build Machine learning portfolios that seek to explicitly exploit the forecasts.

## Machine learning portfolios

1. For each method and at each time point $t$, gather all one-month forecasts across all stocks.
2. Sort all stocks into deciles based on their forecasted returns (using a univariate portfolio sort)
3. Within each decile portfolio value-weight all stocks.
4. Construct a zero-net investment portfolio that buys the highest expected return stocks (decile 10) and sells the lowest (decile 1).

- Note, that Gu et al. (2020) have not calculated breakpoints using NYSE stocks only...

# Machine learning portfolios

Fig. 13: Realized returns in portfolios

| | OLS-3+H | | | | PLS | | | | PCR | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pred | Avg | Std | SR | Pred | Avg | Std | SR | Pred | Avg | Std | SR |
| Low(L) | -0.17 | 0.40 | 5.90 | 0.24 | -0.83 | 0.29 | 5.31 | 0.19 | -0.68 | 0.03 | 5.98 | 0.02 |
| 2 | 0.17 | 0.58 | 4.65 | 0.43 | -0.21 | 0.55 | 4.96 | 0.38 | -0.11 | 0.42 | 5.25 | 0.28 |
| 3 | 0.35 | 0.60 | 4.43 | 0.47 | 0.12 | 0.64 | 4.63 | 0.48 | 0.19 | 0.53 | 4.94 | 0.37 |
| 4 | 0.49 | 0.71 | 4.32 | 0.57 | 0.38 | 0.78 | 4.30 | 0.63 | 0.42 | 0.68 | 4.64 | 0.51 |
| 5 | 0.62 | 0.79 | 4.57 | 0.60 | 0.61 | 0.77 | 4.53 | 0.59 | 0.62 | 0.81 | 4.66 | 0.60 |
| 6 | 0.75 | 0.92 | 5.03 | 0.63 | 0.84 | 0.88 | 4.78 | 0.64 | 0.81 | 0.81 | 4.58 | 0.61 |
| 7 | 0.88 | 0.85 | 5.18 | 0.57 | 1.06 | 0.92 | 4.89 | 0.65 | 1.01 | 0.87 | 4.72 | 0.64 |
| 8 | 1.02 | 0.86 | 5.29 | 0.56 | 1.32 | 0.92 | 5.14 | 0.62 | 1.23 | 1.01 | 4.77 | 0.73 |
| 9 | 1.21 | 1.18 | 5.47 | 0.75 | 1.66 | 1.15 | 5.24 | 0.76 | 1.52 | 1.20 | 4.88 | 0.86 |
| High(H) | 1.51 | 1.34 | 5.88 | 0.79 | 2.25 | 1.30 | 5.85 | 0.77 | 2.02 | 1.25 | 5.60 | 0.77 |
| H-L | 1.67 | 0.94 | 5.33 | 0.61 | 3.09 | 1.02 | 4.88 | 0.72 | 2.70 | 1.22 | 4.82 | 0.88 |
| | ENet+H | | | | GLM+H | | | | RF | | | |
| | Pred | Avg | Std | SR | Pred | Avg | Std | SR | Pred | Avg | Std | SR |
| Low(L) | -0.04 | 0.24 | 5.44 | 0.15 | -0.47 | 0.08 | 5.65 | 0.05 | 0.29 | -0.09 | 6.00 | -0.05 |
| 2 | 0.27 | 0.56 | 4.84 | 0.40 | 0.01 | 0.49 | 4.80 | 0.35 | 0.44 | 0.38 | 5.02 | 0.27 |
| 3 | 0.44 | 0.53 | 4.50 | 0.40 | 0.29 | 0.65 | 4.52 | 0.50 | 0.53 | 0.64 | 4.70 | 0.48 |
| 4 | 0.59 | 0.72 | 4.11 | 0.61 | 0.50 | 0.72 | 4.59 | 0.55 | 0.60 | 0.60 | 4.56 | 0.46 |
| 5 | 0.73 | 0.72 | 4.42 | 0.57 | 0.68 | 0.70 | 4.55 | 0.53 | 0.67 | 0.57 | 4.51 | 0.44 |
| 6 | 0.87 | 0.85 | 4.60 | 0.64 | 0.84 | 0.84 | 4.53 | 0.65 | 0.73 | 0.64 | 4.54 | 0.49 |
| 7 | 1.01 | 0.87 | 4.75 | 0.64 | 1.00 | 0.86 | 4.82 | 0.62 | 0.80 | 0.67 | 4.65 | 0.50 |
| 8 | 1.16 | 0.88 | 5.20 | 0.59 | 1.18 | 0.87 | 5.18 | 0.58 | 0.87 | 1.00 | 4.91 | 0.71 |
| 9 | 1.36 | 0.80 | 5.61 | 0.50 | 1.40 | 1.04 | 5.44 | 0.66 | 0.96 | 1.23 | 5.59 | 0.76 |
| High(H) | 1.66 | 0.84 | 6.76 | 0.43 | 1.81 | 1.14 | 6.33 | 0.62 | 1.12 | 1.53 | 7.27 | 0.73 |
| H-L | 1.70 | 0.60 | 5.37 | 0.39 | 2.27 | 1.06 | 4.79 | 0.76 | 0.83 | 1.62 | 5.75 | 0.98 |

# Machine learning portfolios

Fig. 14: Realized returns in portfolios

| | GBRT+H | | | | NN1 | | | | NN2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pred | Avg | Std | SR | Pred | Avg | Std | SR | Pred | Avg | Std | SR |
| Low(L) | -0.45 | 0.18 | 5.60 | 0.11 | -0.38 | -0.29 | 7.02 | -0.14 | -0.23 | -0.54 | 7.83 | -0.24 |
| 2 | -0.16 | 0.49 | 4.93 | 0.35 | 0.16 | 0.41 | 5.89 | 0.24 | 0.21 | 0.36 | 6.08 | 0.20 |
| 3 | 0.02 | 0.59 | 4.75 | 0.43 | 0.44 | 0.51 | 5.07 | 0.35 | 0.44 | 0.65 | 5.07 | 0.44 |
| 4 | 0.17 | 0.63 | 4.68 | 0.46 | 0.64 | 0.70 | 4.56 | 0.53 | 0.59 | 0.73 | 4.53 | 0.56 |
| 5 | 0.34 | 0.57 | 4.70 | 0.42 | 0.80 | 0.77 | 4.37 | 0.61 | 0.72 | 0.81 | 4.38 | 0.64 |
| 6 | 0.46 | 0.77 | 4.48 | 0.59 | 0.95 | 0.78 | 4.39 | 0.62 | 0.84 | 0.84 | 4.51 | 0.65 |
| 7 | 0.59 | 0.52 | 4.73 | 0.38 | 1.11 | 0.81 | 4.40 | 0.64 | 0.97 | 0.95 | 4.61 | 0.71 |
| 8 | 0.72 | 0.72 | 4.92 | 0.51 | 1.31 | 0.75 | 4.86 | 0.54 | 1.13 | 0.93 | 5.09 | 0.63 |
| 9 | 0.88 | 0.99 | 5.19 | 0.66 | 1.58 | 0.96 | 5.22 | 0.64 | 1.37 | 1.04 | 5.69 | 0.63 |
| High(H) | 1.11 | 1.17 | 5.88 | 0.69 | 2.19 | 1.52 | 6.79 | 0.77 | 1.99 | 1.38 | 6.98 | 0.69 |
| H-L | 1.56 | 0.99 | 4.22 | 0.81 | 2.57 | 1.81 | 5.34 | 1.17 | 2.22 | 1.92 | 5.75 | 1.16 |
| | NN3 | | | | NN4 | | | | NN5 | | | |
| | Pred | Avg | Std | SR | Pred | Avg | Std | SR | Pred | Avg | Std | SR |
| Low(L) | -0.03 | -0.43 | 7.73 | -0.19 | -0.12 | -0.52 | 7.69 | -0.23 | -0.23 | -0.51 | 7.69 | -0.23 |
| 2 | 0.34 | 0.30 | 6.38 | 0.16 | 0.30 | 0.33 | 6.16 | 0.19 | 0.23 | 0.31 | 6.10 | 0.17 |
| 3 | 0.51 | 0.57 | 5.27 | 0.37 | 0.50 | 0.42 | 5.18 | 0.28 | 0.45 | 0.54 | 5.02 | 0.37 |
| 4 | 0.63 | 0.66 | 4.69 | 0.49 | 0.62 | 0.60 | 4.51 | 0.46 | 0.60 | 0.67 | 4.47 | 0.52 |
| 5 | 0.71 | 0.69 | 4.41 | 0.55 | 0.72 | 0.69 | 4.26 | 0.56 | 0.73 | 0.77 | 4.32 | 0.62 |
| 6 | 0.79 | 0.76 | 4.46 | 0.59 | 0.81 | 0.84 | 4.46 | 0.65 | 0.85 | 0.86 | 4.35 | 0.68 |
| 7 | 0.88 | 0.99 | 4.77 | 0.72 | 0.90 | 0.93 | 4.56 | 0.70 | 0.96 | 0.88 | 4.76 | 0.64 |
| 8 | 1.00 | 1.09 | 5.47 | 0.69 | 1.03 | 1.08 | 5.13 | 0.73 | 1.11 | 0.94 | 5.17 | 0.63 |
| 9 | 1.21 | 1.25 | 5.94 | 0.73 | 1.23 | 1.26 | 5.93 | 0.74 | 1.34 | 1.02 | 6.02 | 0.58 |
| High(H) | 1.83 | 1.69 | 7.29 | 0.80 | 1.89 | 1.75 | 7.51 | 0.81 | 1.99 | 1.46 | 7.40 | 0.68 |
| H-L | 1.86 | 2.12 | 6.13 | 1.20 | 2.01 | 2.26 | 5.80 | 1.35 | 2.22 | 1.97 | 5.93 | 1.15 |

Note: In this table, we report the performance of prediction-sorted portfolios over the 30-year out-of-sample testing period. All stocks are sorted into deciles based on their predicted returns for the next month. Column "Pred", "Avg", "Std", and "SR" provide the predicted monthly returns for each decile, the average realized monthly returns, their standard deviations, and Sharpe ratios, respectively. All portfolios are value weighted.

# What is ML capturing?

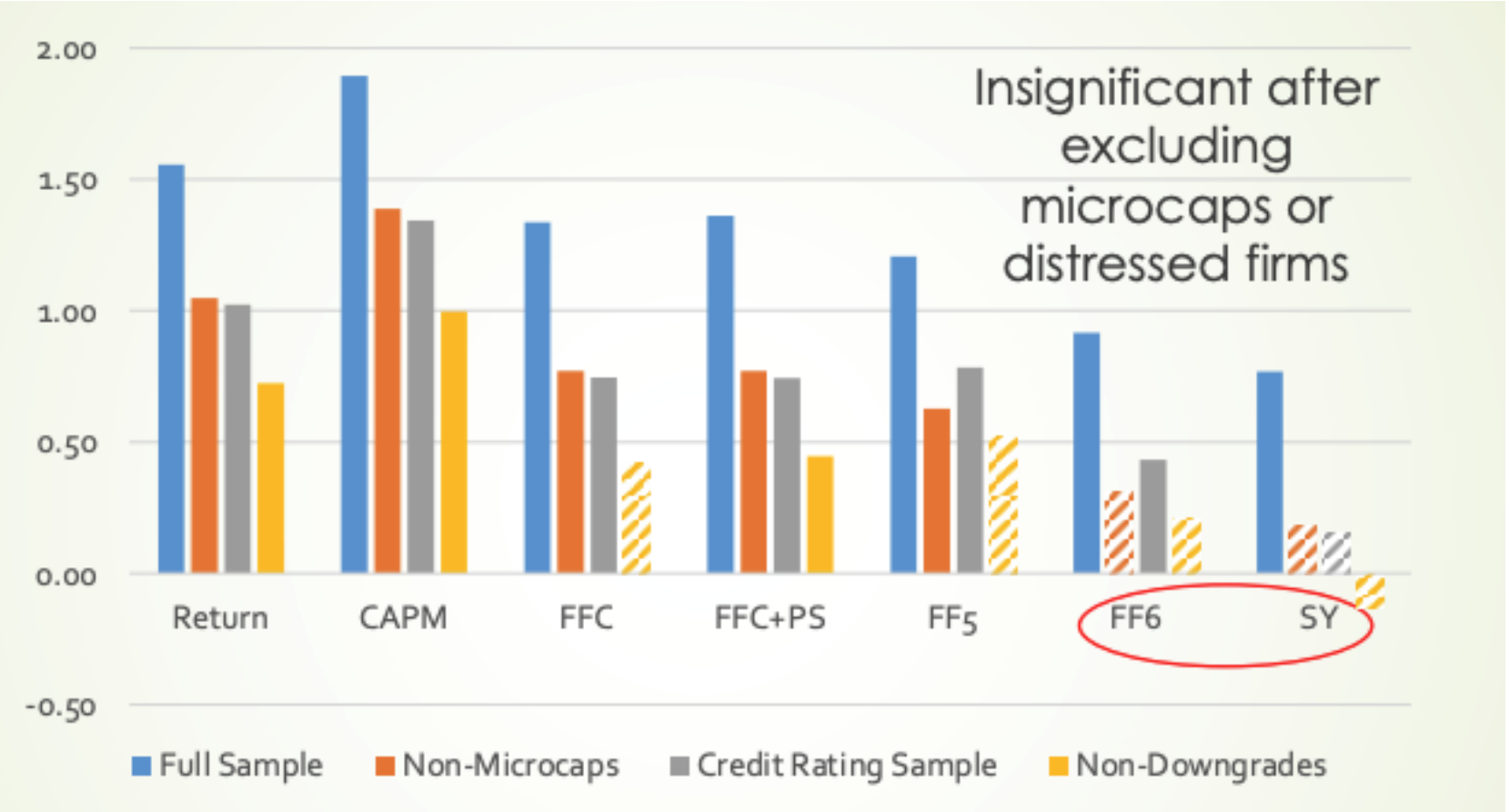
- Avramov et al. (2022) exemines whether machine learning methods clear the common economic restrictions in asset pricing

  ➜ Very similar to your discussion on anomalies!
- Avramov et al. (2022) reexamine the same setup as Gu et al. (2020) for their NN3 model
- Their economic restrictions refers to:

  1. Exclude microcaps: market cap smaller than the 20th NYSE size percentile
  2. Exclude firms with no data on S&P long-term issuer credit rating
  3. Exclude distressed firms: -12 and 12 months around an issuer credit rating downgrade

# What is ML capturing?

Fig. 15: After imposing economic restrictions (directly stolen from Avramov et al. (2022))




➜ Returns are roughly halfed when imposing the economic restrictions!

➜ FF6 (FF5 + momentum) $\alpha$ decreases 78% when imposing the restrictions!

# Other comments

- Avramov et al. (2022) only compare for linear methods like the IPCA method of Kelly et al. (2019)
- Their evidence suggest that nonlinearities are useful for difficult-to-value/difficult-to-arbitrage stocks!

  ➜ Very similar to our discussion on anomalies!
- Unlike anomalies ML seems to generate returns in the post-2001 period

# Other asset classes

- You can find (more or less) a copy-paste of Gu et al. (2020) within other asset classes:
  - Bonds: Bianchi et al. (2021b) or Bianchi et al. (2021a) for the corrected version....
  - Currencies: Filippou et al. (2022)

# Examples of ML techniques

- Let us consider numerous examples in the Matlab live script *mlPredictability.mlx*.

# Potential projects

# About projects

1. Asset return predictability with one or several ML techniques (e.g. on bonds, currencies, equity, etc.)
2. Asset return predictability with other ML type techniques like James-Stein shrinkage, forecast combination (see e.g. Rapach and Zhou (2013), or Borup et al. (2020b)), targeted random forests (Borup et al., 2020a), among other things.
3. Understanding several ML methods' assessment of the variable importance or functional relationship of some important predictors, like the dividend-price ratio.
4. Machine learning portfolios.
5. . . . or be creative.

$\Rightarrow$ there are several data sources with interesting candidate predictors listed in the document that describes the project.

# References


Avramov, D., S. Cheng, and L. Metzker (2022): "Machine learning vs. economic restrictions: Evidence from stock return predictability," *Management Science*.

Bianchi, D., M. Büchner, T. Hoogteijling, and A. Tamoni (2021a): "Corrigendum: Bond risk premiums with machine learning," *The Review of Financial Studies*, 34, 1090–1103.

Bianchi, D., M. Büchner, and A. Tamoni (2021b): "Bond risk premiums with machine learning," *The Review of Financial Studies*, 34, 1046–1089.

Borup, D., B. J. Christensen, N. Mühlbach, and M. S. Nielsen (2020a): "The effects of targeting predictors in a random forest regression model," *Available at SSRN 3551557*.

Borup, D., J. N. Eriksen, M. M. Kjær, and M. Thyrsgaard (2020b): "Predicting bond return predictability," *Available at SSRN*.

Borup, D., D. E. Rapach, E. C. M. Schütte, et al. (2020c): "Now-and backcasting initial claims with high-dimensional daily internet search-volume data," *Available at SSRN 3690832*.

Borup, D. and E. C. M. Schütte (2020): "In search of a job: Forecasting employment growth using Google Trends," *Journal of Business & Economic Statistics*, 1–15.

——— (2021): "Asset pricing with data revisions," *Journal of Financial Markets*, Forthcoming.

Delikouras, S. and A. Kostakis (2019): "A single-factor consumption-based asset pricing model," *Journal of Financial and Quantitative Analysis*, 54, 789–827.

Dong, X., Y. Li, D. E. Rapach, and G. Zhou (2022): "Anomalies and the expected market return," *The Journal of Finance*, 77, 639–681.

Filippou, I., D. Rapach, M. P. Taylor, and G. Zhou (2022): "Out-of-Sample Exchange Rate Prediction: A Machine Learning Perspective," *Available at SSRN 3455713*.

Green, J., J. R. M. Hand, and X. F. Zhang (2017): "Thecharacteristics that provide independent information about average U.S. monthly stock returns," *The Review of Financial Studies*, 30, 4389–4436.

Gu, S., B. Kelly, and D. Xiu (2020): "Empirical asset pricing via Machine Learning," *Review of Financial Studies*, Forthcoming.

Harvey, C. R., Y. Liu, and H. Zhu (2016): "… and the Cross-Section of Expected Returns," *Review of Financial Studies*, 29, 5–68.

Kelly, B. and S. Pruitt (2013): "Market expectations in the cross-section of present values," *The Journal of Finance*, 68, 1721–1756.

Kelly, B. T., S. Pruitt, and Y. Su (2019): "Characteristics are covariances: A unified model of risk and return," *Journal of Financial Economics*, 134, 501–524.

Masini, R. P., M. C. Medeiros, and E. F. Mendes (2020): "Machine Learning Advances for Time Series Forecasting," *arXiv preprint arXiv:2012.12802*.

McCracken, M. W. and S. Ng (2016): "FRED-MD: A Monthly Database for Macroeconomic Research," *Journal of Business & Economic Statistics*, 34, 574–589.

Mitchell, T. (1997): *Machine Learning*, McGraw Hill.

Newey, W. K. and K. D. West (1987): "A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix," *Econometrica*, 55, 703–708.

Rapach, D. and G. Zhou (2013): "Forecasting Stock Returns," in *Handbook of Economic Forecasting*, ed. by G. Elliott and A. Timmermann, Elsevier, vol. 2, chap. 6, 328–383.

Samuel, A. L. (1959): "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, 3, 210–229.

Welch, I. and A. Goyal (2008): "A comprehensive look at the empirical performance of equity premium prediction," *The Review of Financial Studies*, 21, 1455–1508.