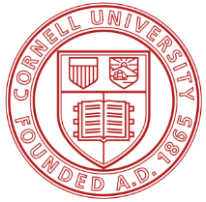


Advances in Financial Machine Learning: Lecture 4/10

Prof. Marcos López de Prado

Advances in Financial Machine Learning

ORIE 5256

What are we going to learn today?

- Ensemble Methods
  - The Three Sources of Error
  - Bootstrap Aggregation (Bagging)
  - Random Forest
  - Boosting
- Cross-Validation in Finance
  - Why K-Fold Cross-Validation Fails in Finance
  - Purged K-Fold Cross-Validation
- Feature Importance
  - With Substitution Effects
  - Without Substitution Effects
- Hyper-parameter Tuning with Cross-Validation
  - Grid Search
  - Randomized Search
  - Scoring

Ensemble Methods

The Three Sources of Errors

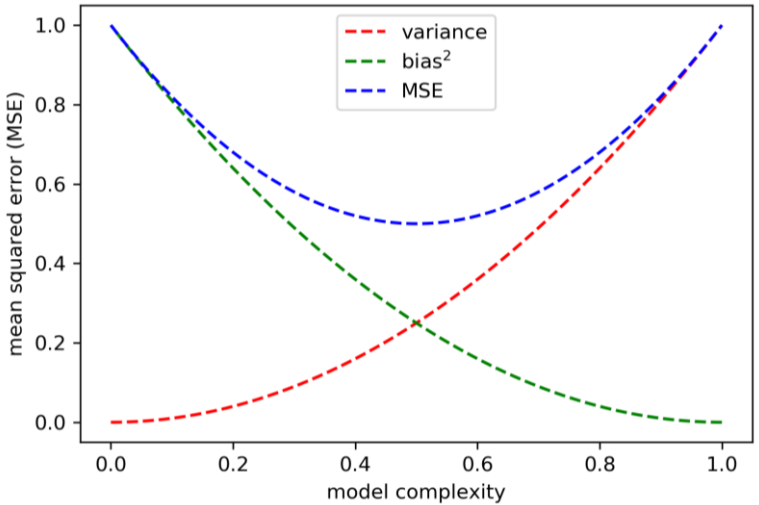
Consider a training set of observations $\{x_i\}_{i=1,\ldots,n}$ and real-valued outcomes $\{y_i\}_{i=1,\ldots,n}$. Suppose a function $f[x]$ exists, such that $y = f[x] + \varepsilon$, where ε is white noise with $\mathrm{E}[\varepsilon_i] = 0$ and $\mathrm{E}[\varepsilon_i^2] = \sigma_\varepsilon^2$. We would like to estimate the function $\hat{f}[x]$ that best fits $f[x]$, in the sense of making the variance of the estimation error $\mathrm{E}[(y_i - \hat{f}[x_i])^2]$ minimal (the mean squared error cannot be zero, because of the noise represented by σ_ε^2). This mean-squared error can be decomposed as

$$\mathrm{E}\left[(y_i - \hat{f}[x_i])^2\right] = \left(\underbrace{\mathrm{E}[\hat{f}[x_i] - f[x_i]]}_{bias}\right)^2 + \underbrace{\mathrm{V}[\hat{f}[x_i]]}_{variance} + \underbrace{\sigma_\varepsilon^2}_{noise}$$

An ensemble method is a method that combines a set of weak learners, all based on the same learning algorithm, in order to create a (stronger) learner that performs better than any of the individual ones. Ensemble methods help reduce bias and/or variance.

Ensemble Methods

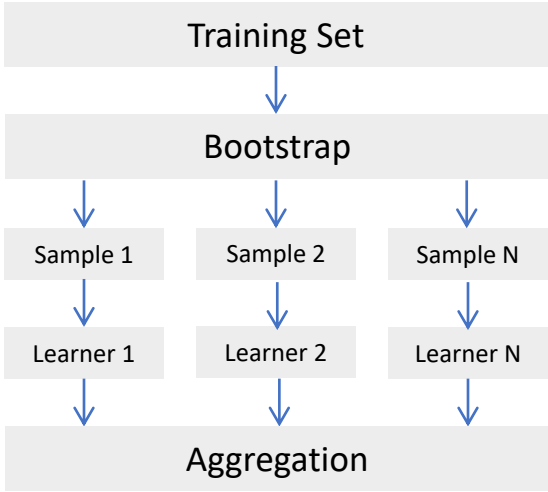
- Statistical models suffer from three kinds of estimation errors:
  - Bias: caused by specification errors (underfitting)
  - Variance: caused by confounding noise with signal (overfitting)
  - Noise: unpredictable changes or measurement errors
- Ensemble methods combine a set of weak learners in order to create a learner that performs better than the individual ones
- The three main types of ensemble methods are
  - Bagging (Bootstrap aggregation)
  - Boosting
  - Stacking



The bias-variance tradeoff: The more we try to reduce bias (underfitting), the more likely we are to increase variance (overfitting). Ensemble methods attempt to produce models that reduce both kinds of errors.

Bagging

- A bagging algorithm forms a strong learner by
  - training several weak learners independently (in parallel), and
  - aggregate their forecasts using a deterministic procedure
- For a sufficiently large number of uncorrelated learners, Bagging algorithms effectively reduce the **variance error**
- In addition, if the individual learners have a minimum accuracy, Bagging algorithms may also reduce the **bias error**
- For classification problems, two popular forms of aggregation are
  - Hard voting: Majority voting across all the forecasts
  - Soft voting: Average the probabilities of the individual forecasts, and select the class with highest average probability



Bagging almost always achieves variance reduction, and in some cases, also bias reduction. Because of its parallelization, bagging is a good default method to prevent overfitting.

Bootstrap Aggregation (Bagging) (1/2)

Consider a bagging classifier that makes a prediction on *k* classes **by majority voting** among *N* independent classifiers. We can label the predictions as {0,1}, where 1 means a correct prediction. The accuracy of a classifier is the probability *p* of attaining a 1. A sufficient condition for considering the classifier "informed" is that the sum of these labels is $X > N/2$. However, given that there are *k* classes, a necessary (non-sufficient) condition for considering the classifier "informed" is that $X > N/k$, which occurs with probability

$$\mathrm{P}\left[X > \frac{N}{k}\right] = 1 - \mathrm{P}\left[X \leq \frac{N}{k}\right] = 1 - \sum_{i=0}^{\lfloor N/k \rfloor} \binom{N}{i} p^i (1-p)^{N-i}$$

The implication is that for a sufficiently large *N*, say $N > p\left(p - \frac{1}{k}\right)^{-2}$, then $p > \frac{1}{k} \Rightarrow \mathrm{P}\left[X > \frac{N}{k}\right] > p$, hence the bagging classifier's accuracy exceeds the average accuracy of the individual classifiers. **This is a strong argument in favor of bagging any classifier in general, when computational requirements permit it**.

Bootstrap Aggregation (Bagging) (2/2)

$$\mathrm{V}\left[\frac{1}{N}\sum_{i=1}^{N}\varphi_i[c]\right] = \frac{1}{N^2}\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\sigma_{i,j}\right) = \frac{1}{N^2}\sum_{i=1}^{N}\left(\sigma_i^2 + \sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j}\right)$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\left(\bar{\sigma}^2 + \underbrace{\sum_{j\neq i}^{N}\bar{\sigma}^2\bar{\rho}}_{\substack{=(N-1)\bar{\sigma}^2\bar{\rho} \\ \text{for a fixed } i}}\right) = \frac{\bar{\sigma}^2 + (N-1)\bar{\sigma}^2\bar{\rho}}{N}$$

$$= \bar{\sigma}^2\left(\bar{\rho} + \frac{1-\bar{\rho}}{N}\right)$$

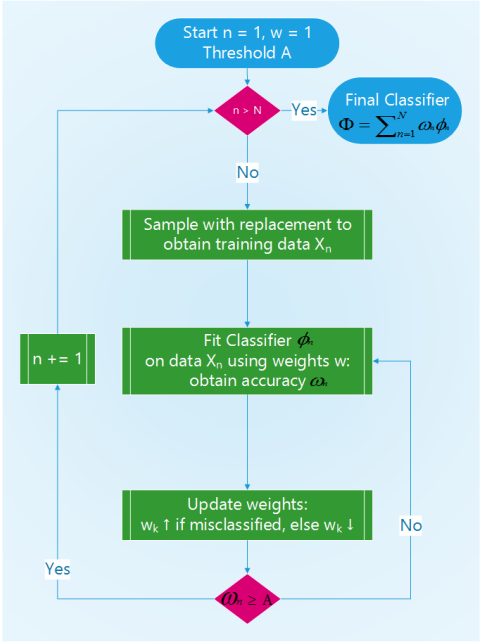
where $\sigma_{i,j}$ is the covariance of predictions by estimators i,j; $\sum_{i=1}^{N}\bar{\sigma}^2 = \sum_{i=1}^{N}\sigma_i^2 \Leftrightarrow \bar{\sigma}^2 = N^{-1}\sum_{i=1}^{N}\sigma_i^2$; and $\sum_{j\neq i}^{N}\bar{\sigma}^2\bar{\rho} = \sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j} \Leftrightarrow \bar{\rho} = (\bar{\sigma}^2 N(N-1))^{-1}\sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j}$.

Random Forest

- Decision trees are known to be prone to overfitting, which increases the variance of the forecasts.
- In order to address this concern, the random forest (RF) method was designed to produce ensemble forecasts with lower variance.
- RF shares some similarities with bagging, in the sense of **independently training individual estimators over bootstrapped subsets of the data**.
- The key difference with bagging is that random forests incorporate a second level of randomness: **When optimizing each node split, only a random subsample (without replacement) of the features will be evaluated**, with the purpose of further de-correlating the estimators.
- Advantages:
  - Like bagging, RF reduces forecasts' variance without overfitting (remember, as long as $\bar{p} < 1$).
  - RF evaluates feature importance on-the-fly.
  - RF provides out-of-bag accuracy estimates, however in financial applications they are likely to be inflated.
- Caveat:
  - Like bagging, RF will not necessarily exhibit lower bias than individual decision trees.

Boosting

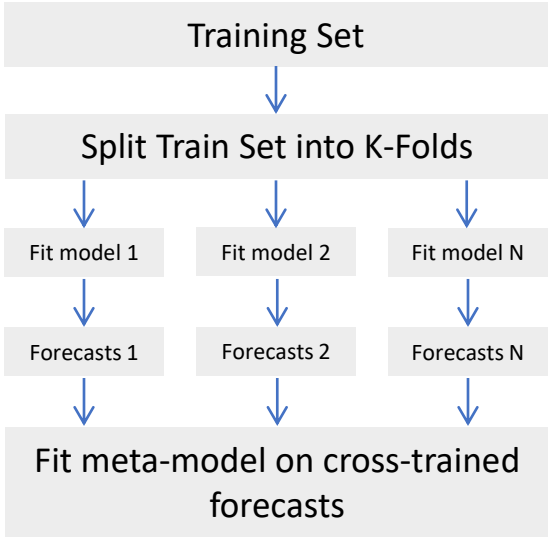
- A boosting algorithm forms a strong learner by
  - training several weak learners iteratively, where subsequent models learn from prior errors, and
  - aggregate their forecasts using a deterministic procedure
- Because of its adaptive nature, Boosting is generally more effective than bagging at reducing bias
- Boosting is can also be effective at reducing variance
- Unlike Bagging, boosted learners cannot be fit in parallel
  - Consequently, Boosting often takes longer to fit
- Two main types of Boosting algorithms
  - Adaboost: It adapts by updating the sample weights
  - Gradient Boosting: It adapts by updating the observations (residuals)



Adaboost decision flow.

Stacking

- A stacking algorithm forms a strong learner by
  - training several weak learners in parallel, and
  - train a meta-model on the individual predictions
- Differences with Bagging and Boosting:
  - Stacking often considers heterogeneous weak learners
  - Stacking uses a meta-model to combine the weak learners (rather than a deterministic method)
- Stacking often involves K-fold cross-training
  1. Split the train set into K-folds
  2. Train N models on K-1 folds, and make predictions on held-out
  3. Train meta-model, with the held-out predictions as features
- Training can be parallelized, however it is computationally expensive



Stacking allows the training of heterogeneous weak learners, which are combined by meta-model.

Numerai's Tournament

Numerai organizes weekly tournaments where researchers produce 1-month financial forecasts.

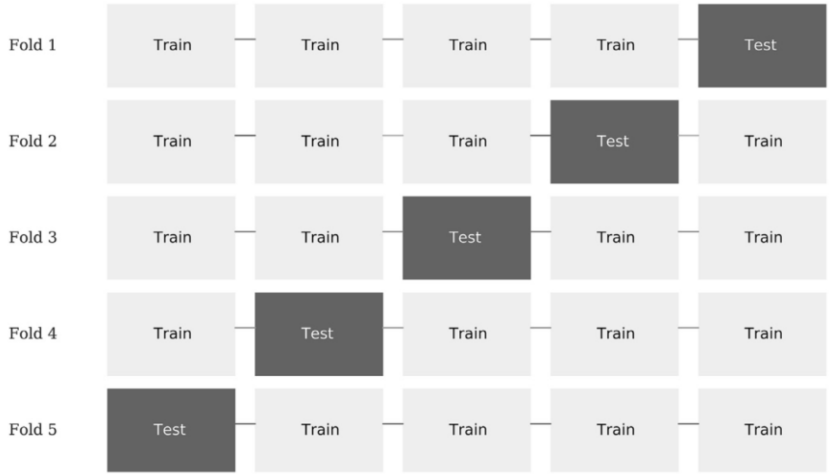
- The dataset is encrypted across company names and features names, so that
  - Vendors allow the distribution of the data
  - Researchers cannot use the testing set for training purposes
- The dataset is structured for cross-sectional studies, to prevent the mapping of companies across time, and defeat the encryption
  - Only relative levels are provided, with monthly frequency, without lags
  - This purposely prevents the modelling of time series characteristics, such as ECM.

When implementing ensemble models on this dataset, it makes sense to:

- **Balance performance across months**: Find month where a model performs poorly, exclude those months from model, and develop a model specifically for those (a kind of boosting).
- **Balance importance across features**: Avoid models that rely heavily on a few features. If those features cease to work, the model will perform poorly.
- **Balance performance across targets**: A robust model will predict different targets.
- **Calibrate bag size**: When bagging, form small bags while controlling that draws come from different months. That reduces the correlation across bags, hence reducing the error's variance.

Cross-Validation in Finance

Why K-Fold Cross-Validation Fails in Finance



Leakage takes place when the training set contains information that also appears in the testing set. Consider a serially correlated feature *X* that is associated with labels *Y* that are formed on overlapping data:

- Because of the serial correlation, $X_t \approx X_{t+1}$.
- Because labels are derived from overlapping datapoints, $Y_t \approx Y_{t+1}$.

Therefore, by placing *t* and *t + 1* in different sets, information is leaked: When a classifier is first trained on (X_t, Y_t), and then it is asked to predict $\mathrm{E}[Y_{t+1}|X_{t+1}]$ based on an observed X_{t+1}, this classifier is more likely to achieve $Y_{t+1} = \mathrm{E}[Y_{t+1}|X_{t+1}]$ even if *X* is an irrelevant feature.

Note that, for leakage to take place, it must occur that $(X_{train}, Y_{train}) \approx (X_{test}, Y_{test})$, and it does not suffice that $X_{train} \approx X_{test}$ or even $Y_{train} \approx Y_{test}$.

Purged K-Fold CV

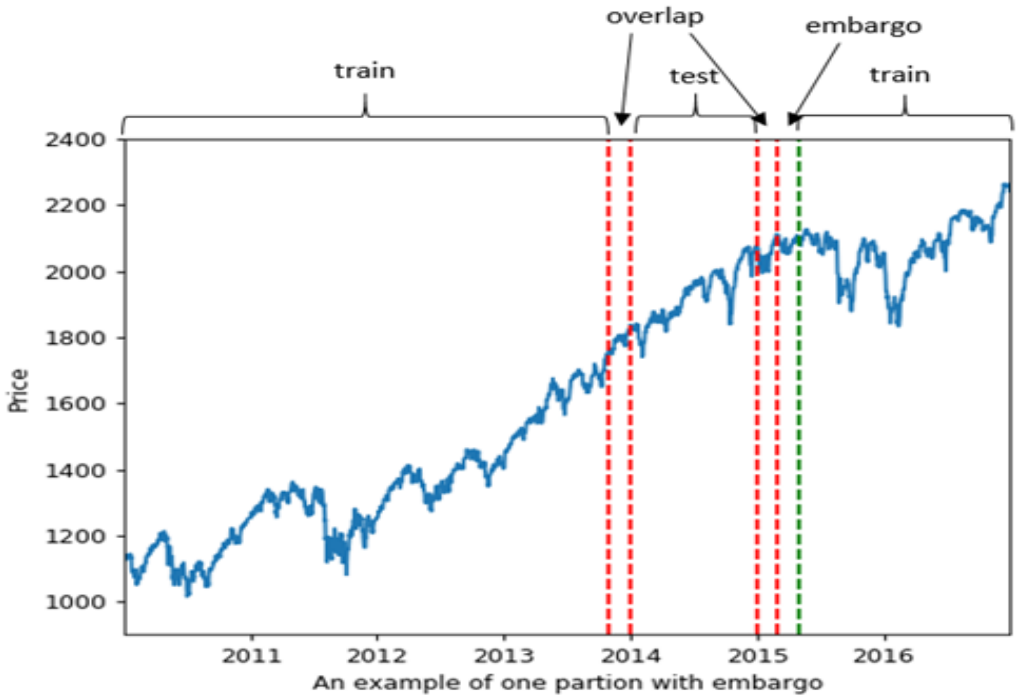
- One way to reduce leakage is to purge from the training set all observations whose labels overlap in time with those labels included in the testing set. I call this process *purging*.
- Consider a label Y_j that is a function of observations in the closed range $t \in [t_{j,0}, t_{j,1}]$, $Y_j = f\left[[t_{j,0}, t_{j,1}]\right]$.
  - For example, in the context of the triple barrier labeling method, it means that the label is the sign of the return spanning between price bars with indices $t_{j,0}$ and $t_{j,i}$, that is $\text{sgn}\left[r_{t_{j,0},t_{j,1}}\right]$.
- A label $Y_i = f\left[[t_{j,0}, t_{j,1}]\right]$ overlaps with Y_j if any of the three sufficient conditions is met:

$$t_{j,0} \leq t_{i,0} \leq t_{j,1}; t_{j,0} \leq t_{i,1} \leq t_{j,1};\ t_{i,0} \leq t_{j,0} \leq t_{j,1} \leq t_{i,1}$$

Embargoed K-Fold CV

- Since financial features often include series that exhibit serial correlation (like ARMA processes), we should eliminate from the training set observations that immediately follow an observation in the testing set. I call this process *embargo*.
  - The embargo does not need to affect training observations prior to a test, because training labels $Y_i = f\left[\left[t_{i,0}, t_{i,1}\right]\right]$, where $t_{i,1} < t_{j,0}$ (training ends before testing begins), contain information that was available at the testing time $t_{j,0}$.
  - We are only concerned with training labels $Y_i = f\left[\left[t_{i,0}, t_{i,1}\right]\right]$ that take place immediately after the test, $t_{j,1} \leq t_{i,0} \leq t_{j,1} + h$.
- We can implement this embargo period h by setting $Y_j = f\left[\left[t_{j,0}, t_{j,1} + h\right]\right]$ before purging. A small value $h \approx .01T$, where T is the number of bars, often suffices to prevent all leakage.

Example: Purging and Embargoing



This plot shows one partition of the K-Fold CV. The test set is surrounded by two train sets, generating two overlaps that must be purged to prevent leakage.

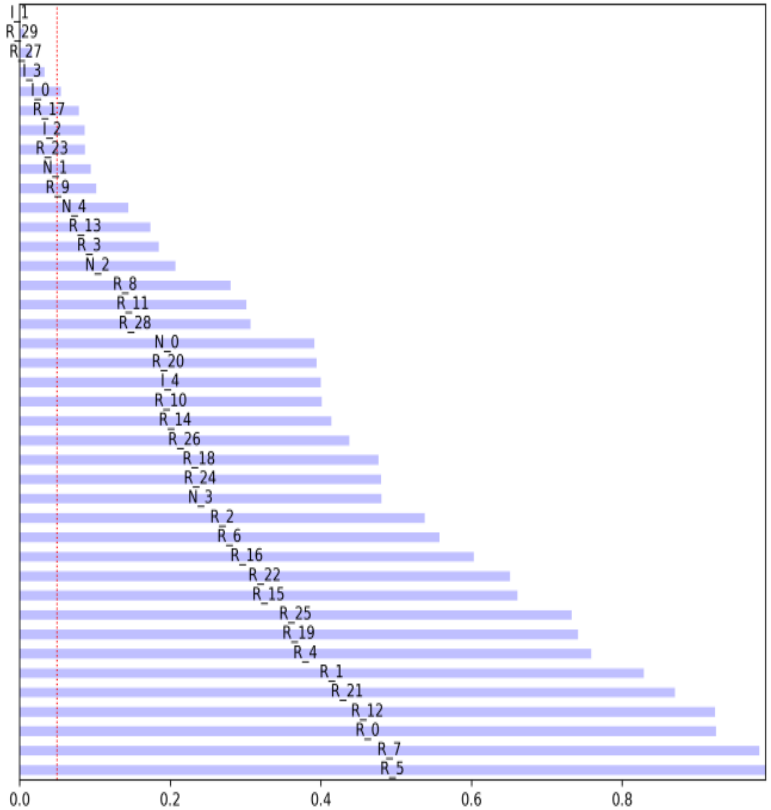
To further prevent leakage, the train observations immediately after the testing set are also embargoed.

Feature Importance

The Importance of Feature Importance

- **Backtesting is not a research tool**
  - It is trivial to fit an ML algorithm to maximize performance in a backtest.
- **Feature Importance is a research tool**
  - We should always try to understand what is the reason for the algorithm's predictive power.
  - In particular, in the context of financial markets, ask yourself: *"What is the risk-based or behavioral reason that explains that a market participant will systematically lose money to my advantage?"*
- Feature Importance methods can largely be classified depending on whether they control for **substitution effects** or not.
- A substitution effect takes place when the estimated importance of one feature is reduced by the presence of other related features.
- Substitution effects are the ML analogue of what the statistics and econometrics literature calls "multi-collinearity."
- One way to address linear substitution effects is to apply PCA on the raw features, and then perform the feature importance analysis on the orthogonal features.
- A better approach is to cluster the raw features and conduct the analysis at the cluster level.

A Numerical Example



Consider a binary random classification problem composed of 40 features, where 5 are informative, 30 are redundant, and 5 are noise.

Informative features (marked with the "I_" prefix) are those used to generate labels. **Redundant features** (marked with the "R_" prefix) are those that are formed by adding Gaussian noise to a randomly chosen informative feature. **Noise features** (marked with the "N_" prefix) are those that are not used to generate labels.

This plot shows the *p*-values that result from a logit regression on those features. The horizontal bars report the *p*-values, and the vertical dash line marks the 5% significance level. Only four out of the 35 non-noise features are deemed statistically significant: I_1, R_29, R_27, I_3. Noise features are ranked as relatively important (9, 11, 14, 18, and 26). Fourteen of the features ranked as least important are not noise.

In short, these *p*-values misrepresent the ground truth.

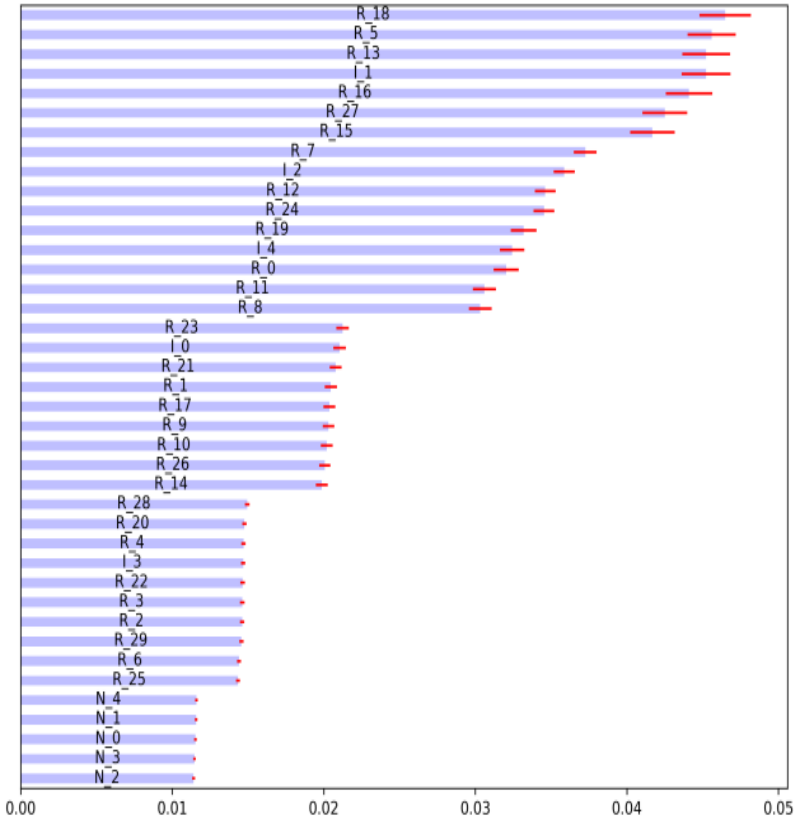
Mean Decrease Impurity (1/2)

Mean decrease impurity (MDI) is a fast, explanatory-importance (in-sample, IS) method specific to tree-based classifiers, like RF. At each node of each decision tree, the selected feature splits the subset it received in such a way that impurity is decreased. Therefore, we can derive for each decision tree how much of the overall impurity decrease can be assigned to each feature. And given that we have a forest of trees, we can average those values across all estimators and rank the features accordingly.

There are some important considerations you must keep in mind when working with MDI:

- By construction, MDI has the nice property that feature importances add up to 1, and every feature importance is bounded between 0 and 1.
- MDI dilutes the importance of substitute features, because of their interchangeability: The importance of two identical features will be halved, as they are randomly chosen with equal probability.
- MDI cannot be generalized to other non tree-based classifiers.
- Masking effects take place when some features are systematically ignored by tree-based classifiers in favor of others. In order to avoid them, set `max_features=int(1)`.

Mean Decrease Impurity (2/2)



This plots shows the result of applying MDI to the same random classification problem discussed earlier.

The horizontal bars indicate the mean of MDI values across 1,000 trees in a random forest, and the lines indicate the standard deviation around that mean. The more trees we add to the forest, the smaller becomes the standard deviation around the mean.

MDI does a good job, in the sense that all of the non-noisy features (either informed or redundant) are ranked higher than the noise features. Still, a small number of non-noisy features appear to be much more important than their peers. This is the kind of substitution effects that we anticipated to find in the presence of redundant features.

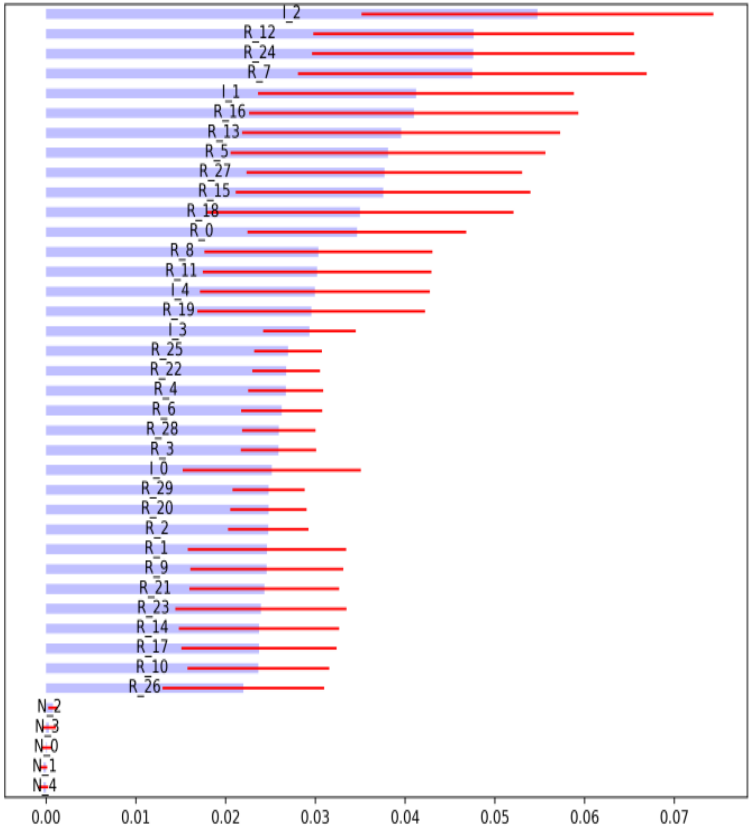
Mean Decrease Accuracy (1/2)

Mean decrease accuracy (MDA) is a slow, predictive-importance (out-of-sample, OOS) method. First, it fits a classifier; second, it derives its performance OOS according to some performance score (accuracy, F1, AUC, negative log-loss, etc.); third, it shuffles each column of the features matrix (X), one column at a time, deriving the performance OOS after each column's permutation. The importance of a feature is a function of the loss in performance caused by its column's permutation.

There are some important considerations you must keep in mind when working with MDA:

- This method can be applied to any classifier, not only tree-based classifiers.
- MDA is not limited to accuracy as the sole performance score.
- Like MDI, the procedure is also susceptible to substitution effects in the presence of correlated features. Given two identical features, MDA always considers one to be redundant to the other. Unfortunately, MDA will make both features appear to be outright irrelevant, even if they are critical.
- Unlike MDI, it is possible that MDA concludes that all features are unimportant. That is because MDA is based on OOS performance.
- The CV must be purged and embargoed, for the reasons explained earlier.

Mean Decrease Accuracy (2/2)



This plot shows the result of applying MDA to the same random classification problem we discussed earlier.

We can draw similar conclusions as we did in the MDI example.

First, MDA does a good job overall at separating noise features from the rest. Noise features are ranked last.

Second, noise features are also deemed unimportant in magnitude, with MDA values of essentially zero.

Third, although substitution effects contribute to higher variances in MDA importance, none is high enough to question the importance of the non-noisy features.

Single Feature Importance

Single feature importance (SFI) is a cross-section predictive-importance (out-of-sample) method. It computes the OOS performance score of each feature in isolation.

A few considerations:

- This method can be applied to any classifier, not only tree-based classifiers.
- SFI is not limited to accuracy as the sole performance score.
- Unlike MDI and MDA, no substitution effects take place, since only one feature
- is taken into consideration at a time.
- Like MDA, it can conclude that all features are unimportant, because performance
- is evaluated via OOS CV.

Orthogonalized Feature Importance

One way to partially address substitution effects is to **orthogonalize the features before applying MDI and MDA**. Consider a matrix $\{X_{t,n}\}$ of stationary features, with observations $t = 1, \ldots, T$ and variables $n = 1, \ldots, N$:

1. We compute the standardized features matrix Z, such that $Z_{t,n} = \sigma_n^{-1}(X_{t,n} - \mu_n)$, where μ_n is the mean of $\{X_{t,n}\}_{t=1,\ldots,T}$ and σ_n is the standard deviation of $\{X_{t,n}\}_{t=1,\ldots,T}$.
2. We compute the eigenvalues Λ and eigenvectors W such that $Z'ZW = W\Lambda$, where Λ is an NxN diagonal matrix with main entries sorted in descending order, and W is an NxN orthonormal matrix.
3. We derive the orthogonal features as $P = ZW$. We can verify the orthogonality of the features by noting that $P'P = W'Z'ZW = W'W\Lambda W'W = \Lambda$.

The diagonalization is done on Z rather than X, for two reasons: (1) centering the data ensures that the first principal component is correctly oriented in the main direction of the observations. It is equivalent to adding an intercept in a linear regression; (2) re-scaling the data makes PCA focus on explaining correlations rather than variances.

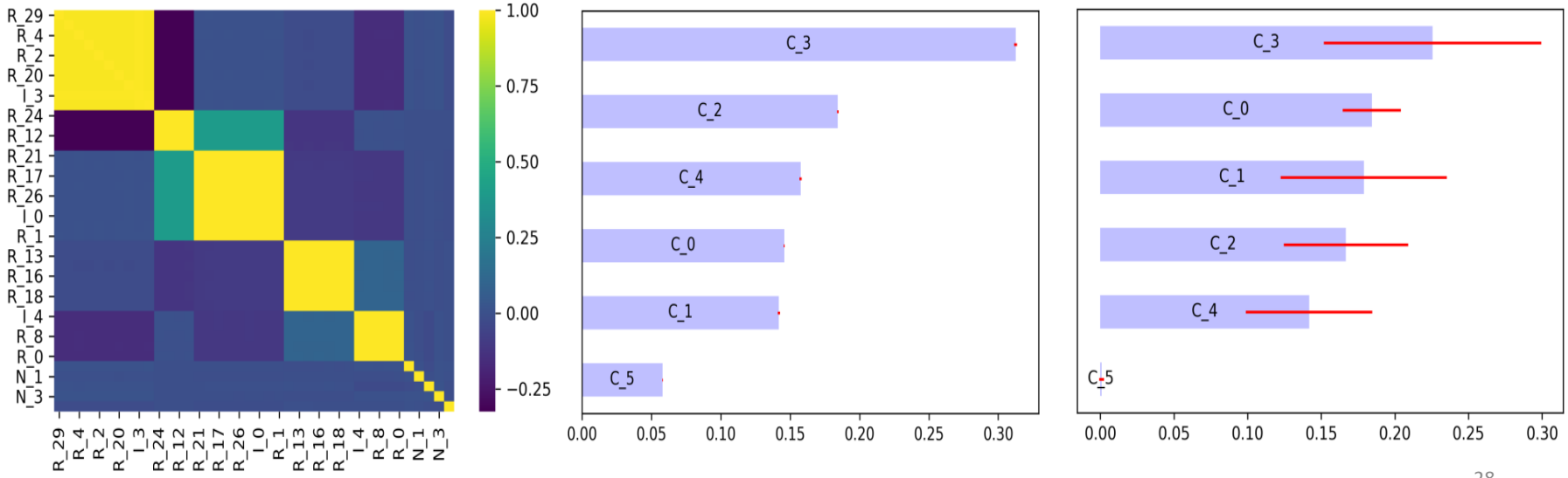
Clustered Feature Importance

- A better approach, which does not require a change of basis, is to cluster together *similar* features, and apply the feature importance analysis at the cluster level.
- By construction, clusters are mutually dissimilar, hence containing the substitution effects.
  - Intuitive results: the analysis is done on a partition of the features, without a change of basis.

- Let us introduce one algorithm that implements this idea. The **clustered feature importance (CFI)** algorithm involves three steps:
  1. Define a similarity measure across all features.
     - The similarity of features can be assessed using (non-linear) information-theoretic measures.
  2. Find the number and constituents of the clusters of features.
     - E.g., apply the ONC algorithm to the observed features.
  3. Apply the feature importance analysis on groups of similar features, rather than on individual features.
     - Cluster MDI is the sum of the MDIs across the constituents of a cluster.
     - Cluster MDA is obtained by shuffling simultaneously all features in a given cluster.

CFI Experimental Results

- In this experiment we test the clustered MDI and MDA procedures on the same dataset we used on the non-clustered versions of MDI and MDA. We apply CFI to control for substitution effects.
- The left plot shows that ONC correctly recognizes that there are 6 relevant clusters (one cluster for each informative feature, plus one cluster of noise features), and it assigns the redundant features to the cluster that contains the informative feature from which the redundant features were derived.
- MDI (center plot) and MDA (right plot) recognize that C_5 contains uninformative features.



Hyper-Parameter Tuning with CV

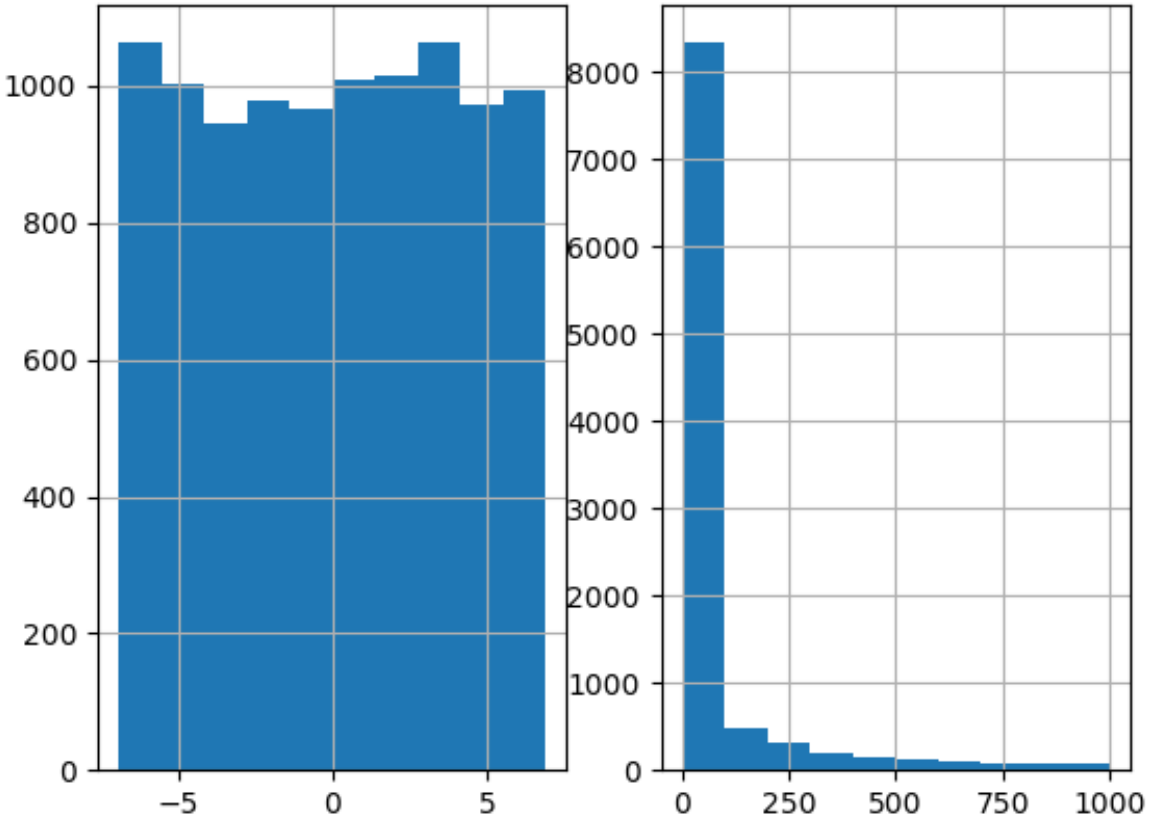
Grid Search CV

- Grid search cross-validation conducts an exhaustive search for the combination of parameters that maximizes the CV performance *on the validation set*, according to some user-defined score function.
- When we do not know much about the underlying structure of the data, this is a reasonable first approach.
- In the context of meta-labeling, you should avoid scoring by accuracy or negative log-loss:
  - Suppose a sample with a very large number of negative (i.e., label ‘0’) cases.
  - A classifier that predicts all cases to be negative will achieve high 'accuracy' or 'neg_log_loss', even though it has not learned from the features how to discriminate between cases.
  - In fact, such a model achieves zero recall and undefined precision.
  - The 'f1' score corrects for that performance inflation by scoring the classifier in terms of precision and recall.
- For other (non-meta-labeling) applications, it is fine to use 'accuracy' or 'neg_log_loss', because we are equally interested in predicting all cases.

Randomized Search CV

- For ML algorithms with a large number of parameters, a grid search cross-validation (CV) becomes computationally intractable.
- In this case, an alternative with good statistical properties is to sample each parameter from a distribution. This has two benefits:
  - We can control for the number of combinations we will search for, regardless of the dimensionality of the problem (the equivalent to a computational budget).
  - Having parameters that are relatively irrelevant performance-wise will not substantially increase our search time, as would be the case with grid search CV.
- It is common for some ML algorithms to accept **non-negative hyper-parameters** only. That is the case of some parameters, such as C in the SVC classifier and gamma in the RBF kernel.
  - We could draw random numbers from a uniform distribution bounded between 0 and some large value, say 100. That would mean that 99% of the values would be expected to be over 1.
  - That is not necessarily the most effective way of exploring the feasibility region of parameters whose functions do not respond linearly. For example, an SVC can be as responsive to an increase in C from 0.01 to 1 as to an increase in C from 1 to 100, so sampling C from a *U* [0, 100] (uniform) distribution will be inefficient.

Log-Uniform Distribution (1/2)



In those instances, it seems more effective to draw values from a distribution *where the logarithm of those draws will be distributed uniformly*.

Log-Uniform Distribution (2/2)

- A random variable *x* follows a log-uniform distribution between $a > 0$ and $b > a$ if and only if $\log[x] \sim U[\log[a], \log[b]]$. This distribution has a CDF:

$$F[x] = \begin{cases} \dfrac{\log[x] - \log[a]}{\log[b] - \log[a]} & \text{for } a \leq x \leq b \\ 0 & \text{for } x < a \\ 1 & \text{for } x > b \end{cases}$$

- From this, we derive a PDF:

$$f[x] = \begin{cases} \dfrac{1}{x \log[b/a]} & \text{for } a \leq x \leq b \\ 0 & \text{for } x < a \\ 0 & \text{for } x > b \end{cases}$$

Note that the CDF is invariant to the base of the logarithm, since $\dfrac{\log\left[\frac{x}{a}\right]}{\log\left[\frac{b}{a}\right]} = \dfrac{\log_c\left[\frac{x}{a}\right]}{\log_c\left[\frac{b}{a}\right]}$ for any base *c*, thus the random variable is not a function of *c*.

Scoring Functions

Useful Classification Scores

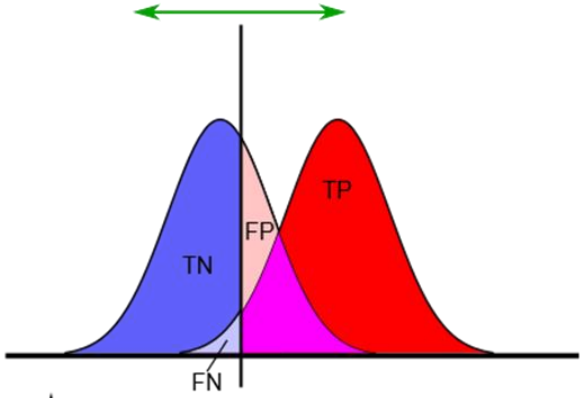
- **Receiver Operating Characteristics (ROC)**: True Positive Rate (recall) as a function of False Positive Rate.
- **Accuracy**: The fraction of opportunities correctly labeled.
- **Precision**: The fraction of true positives among the predicted positives.
- **Recall**: The fraction of true positives among the positives.
- **F1**: The (equally weighted) harmonic mean of *precision* and *recall*.
- **Log-loss (cross-entropy loss)**: It computes the log-likelihood of the classifier given the true label, which takes predictions' probabilities into account. Log loss can be estimated as follows:

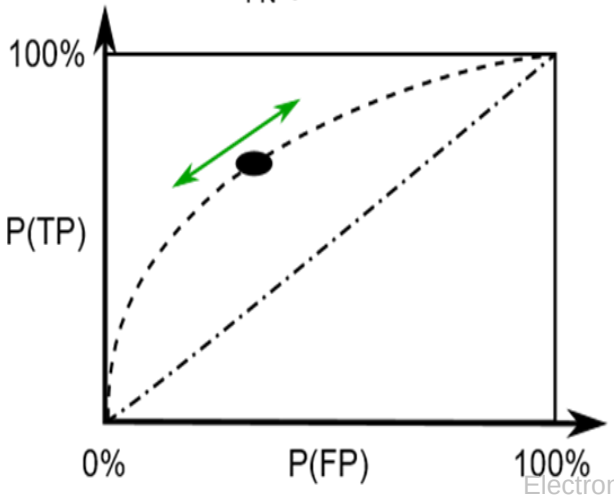
$$L[Y,P] = -\log\left[\text{Prob}[Y|P]\right] = -N^{-1}\sum_{n=0}^{N-1}\sum_{k=0}^{K-1} y_{n,k}\log\left[p_{n,k}\right]$$

where:

- $p_{n,k}$ is the probability associated with prediction *n* of label *k.*
- Y is a 1-of-*K* binary indicator matrix, such that $y_{n,k} = 1$ when observation n was assigned label k out of *K* possible labels, and 0 otherwise.

Area Under the Curve (AUC)





We can derive the ROC curve by plotting the True Positive Rate (TP/P) as a function of the False Positive Rate (FP/N), as we scroll threshold τ from right to left.

When distributions f_0 and f_1 perfectly overlap each other, $FP/N = TP/P$ for every τ, and the ROC curve is a straight diagonal line with 45 degrees of slope. The smaller the overlap between f_0 and f_1, the more the ROC separates from the diagonal.

The ROC curve shows the ability of the classifier to rank the positive instances relative to the negative instances, even if the consensus forecast is weak (big advantage compared to other scores). Also, ROC curves are invariant to changes in class distribution (class skew) and asymmetric error costs.

AUC (area under the ROC curve) is the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. For comparing AUCs from different classifiers, their respective distributions can be derived via bootstrap (Fawcett [2005]).

Cross-Entropy Loss

- Investment strategies profit from predicting the right label with high confidence: Gains from good predictions with low confidence will not suffice to offset the losses from bad predictions with high confidence.
- For this reason, **accuracy does not provide a complete scoring of the classifier's performance**.
- Conversely, log loss (aka cross-entropy loss) computes the log-likelihood of the classifier given the true label, which takes predictions' probabilities into account.

$$L[Y,P] = -\log\left[\text{Prob}[Y|P]\right] = -N^{-1}\sum_{n=0}^{N-1}\sum_{k=0}^{K-1} y_{n,k}\log\left[p_{n,k}\right]$$

where

- $p_{n,k}$ is the probability associated with prediction *n* of label *k.*
- Y is a 1-of-*K* binary indicator matrix, such that $y_{n,k} = 1$ when observation n was assigned label k out of *K* possible labels, and 0 otherwise.

Probability-Weighted Accuracy

- One disadvantage, however, is that log-loss scores are not easy to interpret and compare. A possible solution is to compute the probability-weighted accuracy (PWA) as

$$PWA = \sum_{n=0}^{N-1} y_n (p_n - K^{-1}) \Big/ \sum_{n=0}^{N-1} (p_n - K^{-1})$$

where $p_n = \max_k \{p_{n,k}\}$, and y_n is an indicator function, $y_n \in \{0,1\}$, where $y_n = 1$ when the prediction was correct and $y_n = 0$ otherwise.

- This is equivalent to standard accuracy when $p_n = 1$ for all *n*.

- PWA punishes a high confidence bad prediction more strongly than accuracy, but less strongly than cross-entropy.

Binary Scores on Multi-Class Labels

- Some scores can only be computed on binary labels

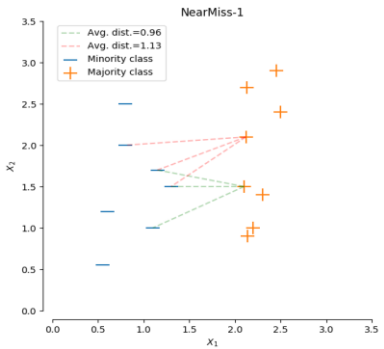
- When a label is multi-class, we can binarize the labels and compute the binary scores on them
  - Form a labels matrix with one row per observation and one column per class
  - For each column, set 1 if the class associated with that column took place, and 0 otherwise

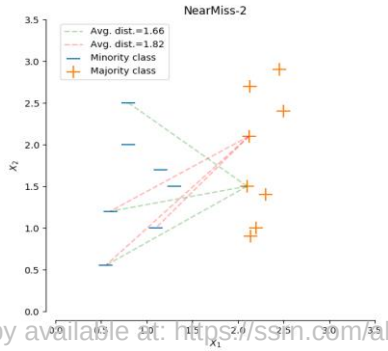
- There are two main approaches
  - **Macro**:
    - Compute the binary score on each column of the labels matrix, and average the scores
    - The average could be weighted by the relative number of occurrences per class
  - **Micro**:
    - Stack all columns in the labels matrix (a single column), and compute the combined binary score

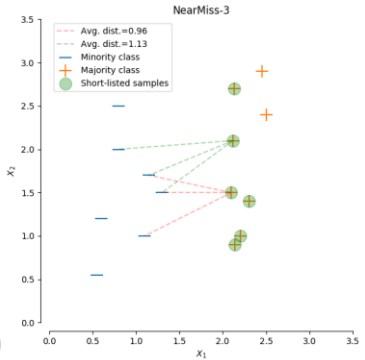
Unbalanced Classes

Near-Miss

- The Near-Miss algorithm undersamples the majority class
- Three popular heuristics for choosing samples
  - Version 1 : It selects samples of the majority class for which average distances to the k *closest* instances of the minority class is smallest.
  - Version 2 : It selects samples of the majority class for which average distances to the k *farthest* instances of the minority class is smallest.
  - Version 3 : It works in 2 steps.
    1. For each minority class instance, their M nearest-neighbors will be stored.
    2. The majority class instances are selected for which the average distance to the N nearest-neighbors is the largest.

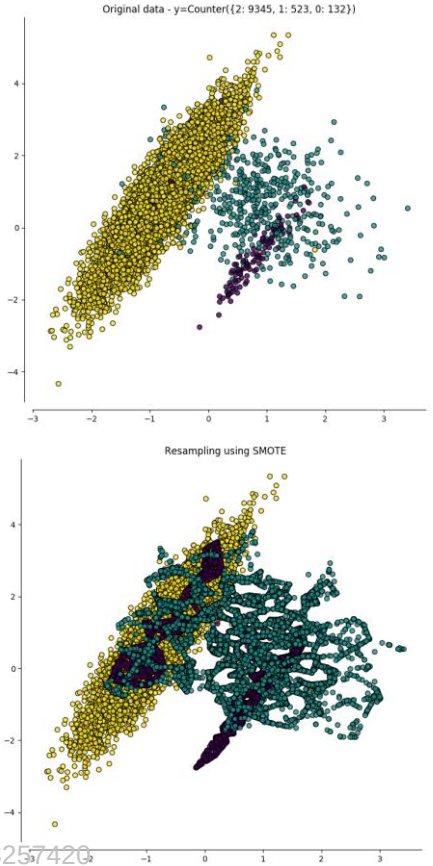




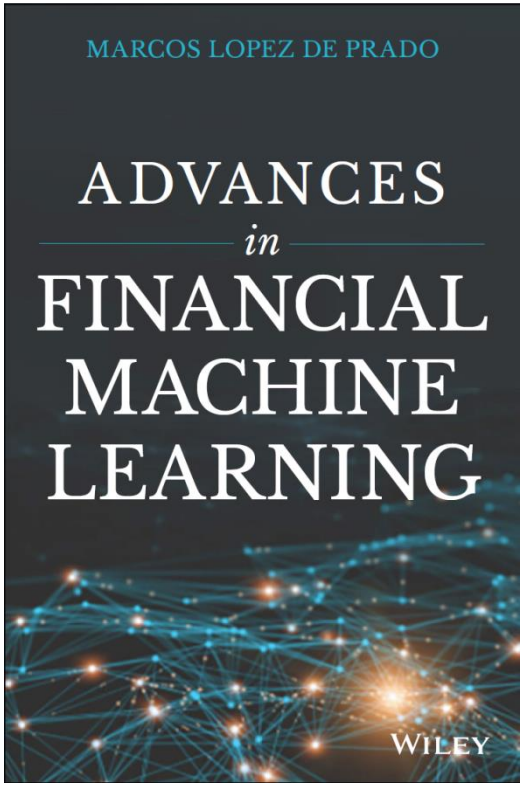


SMOTE

- SMOTE (Synthetic Minority Oversampling Technique) oversamples the minority class.
- It aims to balance class distribution by randomly increasing minority class examples by *replicating* them.
- SMOTE synthesizes new minority instances between existing minority instances.
    - It generates the virtual training records by linear interpolation for the minority class.
    - These synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class.
    - After the oversampling process, the data is reconstructed and several classification models can be applied for the processed data.



For Additional Details

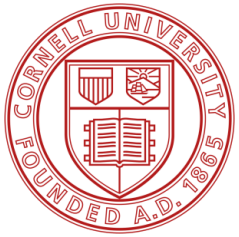


The first wave of quantitative innovation in finance was led by Markowitz optimization. Machine Learning is the second wave and it will touch every aspect of finance. López de Prado's Advances in Financial Machine Learning is essential for readers who want to be ahead of the technology rather than being replaced by it.
— Prof. **Campbell Harvey**, Duke University. Former President of the American Finance Association.

Financial problems require very distinct machine learning solutions. Dr. López de Prado's book is the first one to characterize what makes standard machine learning tools fail when applied to the field of finance, and the first one to provide practical solutions to unique challenges faced by asset managers. Everyone who wants to understand the future of finance should read this book.
— Prof. **Frank Fabozzi**, EDHEC Business School. Editor of The Journal of Portfolio Management.

Disclaimer

- The views expressed in this document are the authors' and do not necessarily reflect those of the organizations he is affiliated with.
- No investment decision or particular course of action is recommended by this presentation.
-



Codependence

Prof. Marcos López de Prado
Advances in Financial Machine Learning
ORIE 5256

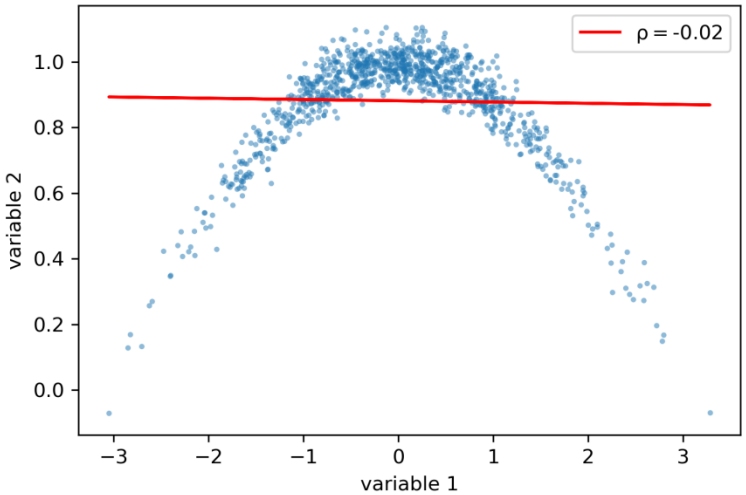
Key Points

- Two random variables are codependent when knowing the value of one helps us determine the value of the other
  - This should not me confounded with the notion of causality
  - Y may tell us something about Z because both Y and Z are caused by X
- Correlation is perhaps the best-known measure of codependence in Econometric studies
  - There is no theoretical reason to justify this choice
- **Despite its popularity among economists, correlation has many known limitations in the contexts of financial studies**
  - **Correlations are often unstable due to wrongly assuming linearity (misspecification error)**
- In this seminar we will explore more modern measures of codependence, based on Information Theory, which overcome some of the limitations of correlations

Correlation

Limitations of Correlation

- Consider two random vectors $\{X, Y\}$, and a correlation estimate $\rho[X, Y]$, with the only requirement that $\sigma[X, Y] = \rho[X, Y]\sigma[X]\sigma[Y]$, where $\sigma[X, Y]$ is the covariance between the two vectors, and $\sigma[.]$ is the standard deviation
  - Pearson's correlation is one of several correlation estimates to satisfy this requirement
- Correlations present three important caveats:
  1. Correlation quantifies the *linear* codependency between two random variables. It neglects non-linear relationships
  2. Correlation is highly influenced by outliers
  3. The application of correlations beyond the multivariate Normal case is questionable. We may compute the correlation between any two real variables, however that correlation is typically meaningless unless the two variables follow a bivariate Normal distribution



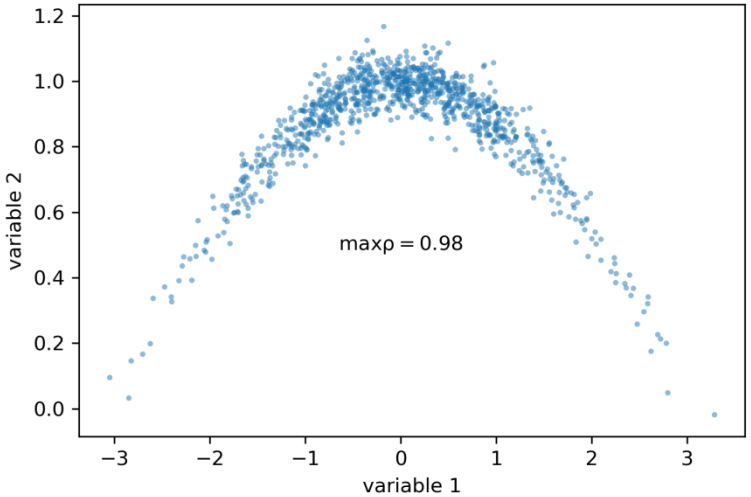
Correlation is a flawed measure of financial codependence. Many financial relationships are non-linear, and correlation fails to recognize them

Maximal Correlation (1/2)

- In 1959, mathematicians Hirschfeld, Gebelein and Rényi proposed a non-linear generalization of Pearson's correlation:

$$\rho_{max}[X,Y] = \sup_{\substack{f:\mathcal{X}\to\mathbb{R}, g:\mathcal{Y}\to\mathbb{R} \\ \mathrm{E}[f[X]]=\mathrm{E}[g[Y]]=0 \\ \mathrm{E}[f^2[X]]=\mathrm{E}[g^2[Y]]=1}} \mathrm{E}[f[X]g[Y]]$$

- This definition has the properties:
  - $0 \le \rho_{max}[X,Y] \le 1$
  - $\rho_{max}[X,Y] = 0$ if and only if X and Y are independent
  - $\rho_{max}[X,Y] = 1$ if there exist functions such that $f[X] = g[Y]$
  - $\rho_{max}[X,Y] = |\rho[X,Y]|$ if X and Y are jointly Gaussian
- The [ACE algorithm] finds $f[.], g[.]$



Maximal correlation extends the notion of Pearson correlation non-linear relationships. One drawback is that its estimation is relatively computational expensive

Maximal Correlation (2/2)

```
import numpy as np
from ace import model # https://pypi.org/project/ace/
#--------------------------------------------------------------------------
def max_correlation(x:np.array,y:np.array)->float:
    # Get max correlation using ace package
    # https://mlfinlab.readthedocs.io/en/latest/implementations/codependence.html
    ace_model=model.Model()
    ace_model.build_model_from_xy([x],y)
    return np.corrcoef(ace_model.ace.x_transforms[0],
                       ace_model.ace.y_transform)[0][1]
```

Python implementation for the estimation of maximal correlation.

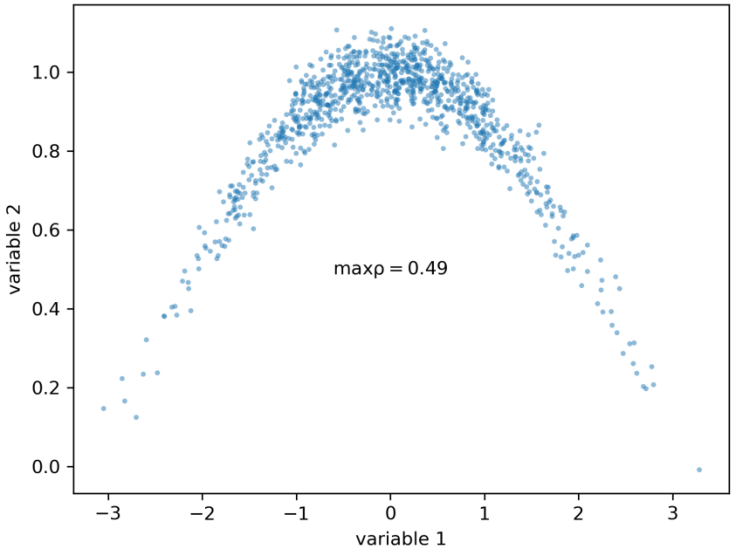
Distance Correlation (1/2)

- In 2005, Gábor Székely introduced the notion of distance correlation, also as a non-linear generalization of Pearson's correlation:

$$\rho_{dist}[X,Y] = \frac{dCov[X,Y]}{\sqrt{dCov[X,X]dCov[Y,Y]}}$$

where $dCov[X,Y]$ can be interpreted as the average Hadamard product of the doubly-centered Euclidean distance matrices of X, Y

- Then
  - $0 \leq \rho_{dist}[X,Y] \leq 1$
  - $\rho_{dist}[X,Y] = 0$ if and only if X and Y are independent



Distance correlation is another non-linear extension of Pearson's correlation. Like Maximal correlation, its estimation can be computational expensive

Distance Correlation (2/2)

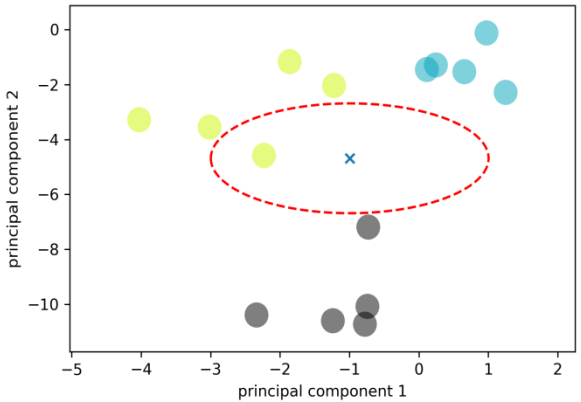
```
import numpy as np,copy
from scipy.spatial.distance import pdist, squareform
#-----------------------------------------------------------------------------
def distcorr(Xval, Yval, pval=True, nruns=500):
    # https://gist.github.com/wladston/c931b1495184fbb99bec
    X,Y=np.atleast_1d(Xval),np.atleast_1d(Yval)
    if np.prod(X.shape) == len(X):X = X[:, None]
    if np.prod(Y.shape) == len(Y):Y = Y[:, None]
    X,Y=np.atleast_2d(X),np.atleast_2d(Y)
    n=X.shape[0]
    if Y.shape[0] != X.shape[0]:raise ValueError('Number of samples must match')
    a,b=squareform(pdist(X)),squareform(pdist(Y))
    A = a - a.mean(axis=0)[None, :] - a.mean(axis=1)[:, None] + a.mean()
    B = b - b.mean(axis=0)[None, :] - b.mean(axis=1)[:, None] + b.mean()
    dcov2_xy = (A * B).sum() / float(n * n)
    dcov2_xx = (A * A).sum() / float(n * n)
    dcov2_yy = (B * B).sum() / float(n * n)
    dcor = np.sqrt(dcov2_xy) / np.sqrt(np.sqrt(dcov2_xx) * np.sqrt(dcov2_yy))
    if pval:
        greater = 0
        for i in range(nruns):
            Y_r = copy.copy(Yval)
            np.random.shuffle(Y_r)
            if distcorr(Xval, Y_r, pval=False) > dcor:
                greater += 1
        return (dcor, greater / float(nruns))
    else:
        return dcor
```

Python implementation for the estimation of distance correlation.

When pval=True, the function also returns the *p*-value associated with the estimated distance correlation, based on the number of simulations set by nruns.

Comparing Correlations

- Codependence attempts to measure how closely associated two random variables are
- However, **correlation is not a metric**, because it does not necessarily satisfy two conditions:
  - non-negativity: $-1 \leq \rho[X,Y] \leq 1$
  - subadditivity: $\rho[X,Z] \not\leq \rho[X,Y] + \rho[Y,Z]$
- Comparing non-metric measurements of codependence can lead to rather incoherent outcomes
  - For instance, the difference between correlations $(0.9,1.0)$ is the same as $(0.1,0.2)$, even though the former involves a greater difference in terms of codependence
- We cannot cluster directly on correlations. We can either:
  a) define a metric based on correlation
  b) apply a metric on correlations (e.g., compute the Euclidean distance on observed correlations)



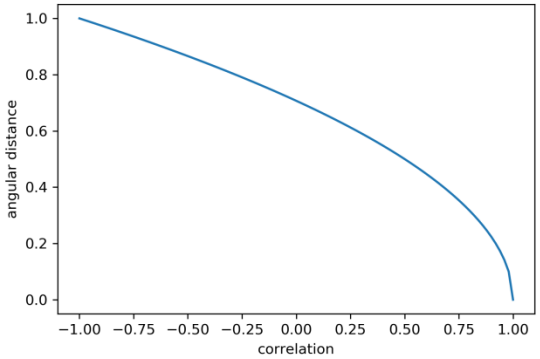
Metric functions are important because they induce an intuitive topology on a given set, which allows us to apply notions of "proximity" or "similarity"

Correlation-based Distance (1/2)

- For Pearson's correlation $\rho[X,Y]$, consider the measure

$$d_\rho[X,Y] = \sqrt{\frac{1}{2}(1-\rho[X,Y])}$$

- This measure is known as the angular distance. It is a metric, because it is a linear multiple of the Euclidean distance between the vectors $\{X,Y\}$ (after standardization)
  - See López de Prado [2016] for a proof
- The metric $d_\rho[X,Y]$ is normalized, $d_\rho[X,Y] \in [0,1]$, because $\rho[X,Y] \in [-1,1]$

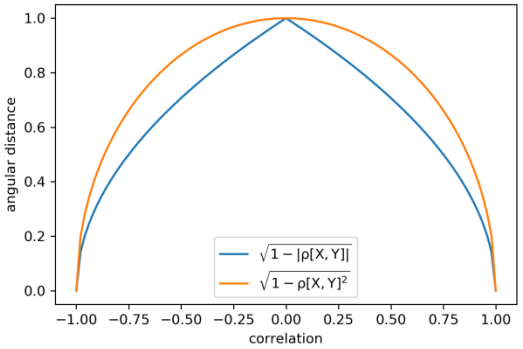


The angular distance satisfies all the conditions of a true metric

Correlation-based Distance (2/2)

- The metric $d_\rho[X,Y]$ deems more distant two random variables with negative correlation than two random variables with positive correlation, regardless of $|\rho[X,Y]|$
- This property makes sense in many applications. For example, we may wish to build a long-only portfolio, where holdings in negative-correlated securities can only offset risk, and therefore should be treated as different for diversification purposes
- In other instances, like in long-short portfolios, we often prefer to consider highly negatively-correlated securities as similar, because the position sign can override the sign of the correlation
- In those cases, we can define alternative normalized correlation-based distance metrics, such as

$$d_{|\rho|}[X,Y] = \sqrt{1-|\rho[X,Y]|}, \text{ or } d_{\rho^2}[X,Y] = \sqrt{1-\rho[X,Y]^2}$$



In some financial applications, it makes more sense to apply a modified definition of angular distance, such that the sign of the correlation is ignored

Information-Theoretic Codependence

Entropy

- Let X be a discrete random variable that takes a value x from the set S_X with probability $p[x]$. The entropy of X is defined as

$$H[X] = -\sum_{x \in S_X} p[x] \log\big[p[x]\big]$$

- A few observations:
  - The value $\frac{1}{p[x]}$ measures how surprising an observation is, because surprising observations are characterized by their low probability
  - Entropy is the expected value of those surprises, where the $\log[.]$ function prevents that $p[x]$ cancels $\frac{1}{p[x]}$ and endows entropy with desirable mathematical properties
  - Accordingly, entropy can be interpreted as the amount of uncertainty associated with X. Entropy is zero when all probability is concentrated in a single element of S_X. Entropy reaches a maximum at $\log[\|S_X\|]$ when X is distributed uniformly, $p[x] = \frac{1}{\|S_X\|}, \forall x \in S_X$

Joint Entropy

- Let Y be a discrete random variable that takes a value y from the set S_Y with probability $p[y]$. Random variables X and Y do not need to be defined on the same probability space. The joint entropy of X and Y is

$$H[X,Y] = -\sum_{x,y \in S_X \times S_Y} p[x,y] \log\big[p[x,y]\big]$$

- In particular, we have that $H[X,Y] = H[Y,X]$, $H[X,X] = H[X]$, $H[X,Y] \geq \max\{H[X], H[Y]\}$, and $H[X,Y] \leq H[X] + H[Y]$
- It is important to recognize that Shannon's entropy is finite only for discrete random variables
  - In the continuous case, one should use the limiting density of discrete points (LDDP), or discretize the random variable, as explained later (Jaynes [2003])

Conditional Entropy

- The conditional entropy of X given Y is defined as

$$H[X|Y] = H[X,Y] - H[Y] = -\sum_{y \in S_Y} p[y] \sum_{x \in S_X} p[x|Y=y] \log\left[p[x|Y=y]\right]$$

  where $p[x|Y=y]$ is the probability that X takes the value x conditioned on Y having taken the value y

- Following this definition, $H[X|Y]$ is the uncertainty we expect in X if we are told the value of Y
- Accordingly, $H[X|X] = 0$, and $H[X] \geq H[X|Y]$

Kullback-Leibler Divergence

- Let p and q be two discrete probability distributions defined on the same probability space. The Kullback-Leibler (or KL) divergence between p and q is

$$D_{KL}[p\|q] = -\sum_{x \in S_X} p[x] \log\left[\frac{q[x]}{p[x]}\right] = \sum_{x \in S_X} p[x] \log\left[\frac{p[x]}{q[x]}\right]$$

  where $q[x] = 0 \Longrightarrow p[x] = 0$.
- Intuitively, this expression measures how much p diverges from a reference distribution q
- The KL divergence is *not* a metric: Although it is always non-negative ($D_{KL}[p\|q] \geq 0$), it violates the symmetry ($D_{KL}[p\|q] \neq D_{KL}[q\|p]$) and triangle inequality conditions
- Note the difference with the definition of joint entropy, where the two random variables did not necessarily exist in the same probability space
- KL divergence is widely used in variational inference

Cross Entropy

- Let p and q be two discrete probability distributions defined on the same probability space. Cross entropy between p and q is

$$H_C[p\|q] = -\sum_{x \in S_X} p[x]\log\big[q[x]\big] = H[X] + D_{KL}[p\|q]$$

- Cross entropy can be interpreted as the uncertainty associated with X, where we evaluate its information content using a wrong distribution q rather than the true distribution p
- Cross entropy is a popular scoring function in classification problems, and it is particularly meaningful in financial applications (López de Prado [2018], section 9.4)

Mutual Information (1/3)

- Mutual information is defined as the decrease in uncertainty (or informational gain) in X that results from knowing the value of Y,

$$I[X,Y] = H[X] - H[X|Y] = H[X] + H[Y] - H[X,Y] = \sum_{x \in S_X} \sum_{y \in S_Y} p[x,y] \log \left[\frac{p[x,y]}{p[x]p[y]} \right]$$

$$= D_{KL}[p[x,y] \| p[x]p[y]] = \sum_{y \in S_Y} p[y] \sum_{x \in S_X} p[x|y] \log \left[\frac{p[x|y]}{p[x]} \right] = \mathrm{E}_Y \left[D_{KL}[p[x|y] \| p[x]] \right]$$

$$= \sum_{x \in S_X} p[x] \sum_{y \in S_Y} p[y|x] \log \left[\frac{p[y|x]}{p[y]} \right] = \mathrm{E}_X \left[D_{KL}[p[y|x] \| p[y]] \right]$$

- From the above we can see that $I[X,Y] \geq 0$, $I[X,Y] = I[Y,X]$, and that $I[X,X] = H[X]$

Mutual Information (2/3)

- When X and Y are independent, $p[x, y] = p[x]p[y]$, hence $I[X, Y] = 0$
- An upper boundary is given by $I[X, Y] \leq \min\{H[X], H[Y]\}$
- However, mutual information is *not* a metric, because it does not satisfy the triangle inequality: $I[X, Z] \not\leq I[X, Y] + I[Y, Z]$
- An important attribute of mutual information is its grouping property,

$$I[X, Y, Z] = I[X, Y] + I[(X, Y), Z]$$

  where (X, Y) represents the joint distribution of X and Y.
- Since X, Y and Z can themselves represent joint distributions, the above property can be used to decompose mutual information into simpler constituents
  - This makes mutual information a useful similarity measure in the context of agglomerative clustering algorithms and forward feature selection

Mutual Information (3/3)

```
import numpy as np,scipy.stats as ss
from sklearn.metrics import mutual_info_score
#-------------------------------------------------------------------------
def mutualInfo(x,y,bXY,norm=False):
    cXY=np.histogram2d(x,y,bXY)[0]
    iXY=mutual_info_score(None,None,contingency=cXY) # mutual information
    if norm:
        hX=ss.entropy(np.histogram(x,bXY)[0]) # marginal
        hY=ss.entropy(np.histogram(y,bXY)[0]) # marginal
        iXY/=min(hX,hY)
    return iXY
```

Python implementation for the estimation of $I[X,Y]$:

- Pass the number of bins as bXY (needed for discretizing x and y)
- Pass norm=True if for the normalized mutual information (bounded between 0 and 1)

Variation of Information (1/3)

- Variation of information is defined as

$$VI[X,Y] = H[X|Y] + H[Y|X] = H[X] + H[Y] - 2I[X,Y] = 2H[X,Y] - H[X] - H[Y]$$

- This measure can be interpreted as the uncertainty we expect in one variable if we are told the value of another
- It has a lower bound in $VI[X,Y] = 0 \Leftrightarrow X = Y$, and an upper bound in $VI[X,Y] \leq H[X,Y]$
- Variation of information is a metric, because it satisfies the axioms: (a) non-negativity, $VI[X,Y] \geq 0$; (b) symmetry, $VI[X,Y] = VI[X,Y]$; and (c) triangle inequality,$VI[X,Z] \leq VI[X,Y] + VI[Y,Z]$
- Because $H[X,Y]$ is a function of the sizes of S_X and S_Y, $VI[X,Y]$ does not have a firm upper bound
  - This is problematic when we wish to compare variations of information across different population sizes

Variation of Information (2/3)

- The following quantity is a metric bounded between zero and one for all pairs (X, Y),

$$\widetilde{VI}[X,Y] = \frac{VI[X,Y]}{H[X,Y]} = 1 - \frac{I[X,Y]}{H[X,Y]}$$

- Following Kraskov et al. [2008], a sharper alternative bounded metric is

$$\widetilde{\widetilde{VI}}[X,Y] = \frac{\max\{H[X|Y], H[Y|X]\}}{\max\{H[X], H[Y]\}} = 1 - \frac{I[X,Y]}{\max\{H[X], H[Y]\}}$$

where $\widetilde{\widetilde{VI}}[X,Y] \leq \widetilde{VI}[X,Y]$ for all pairs (X, Y).

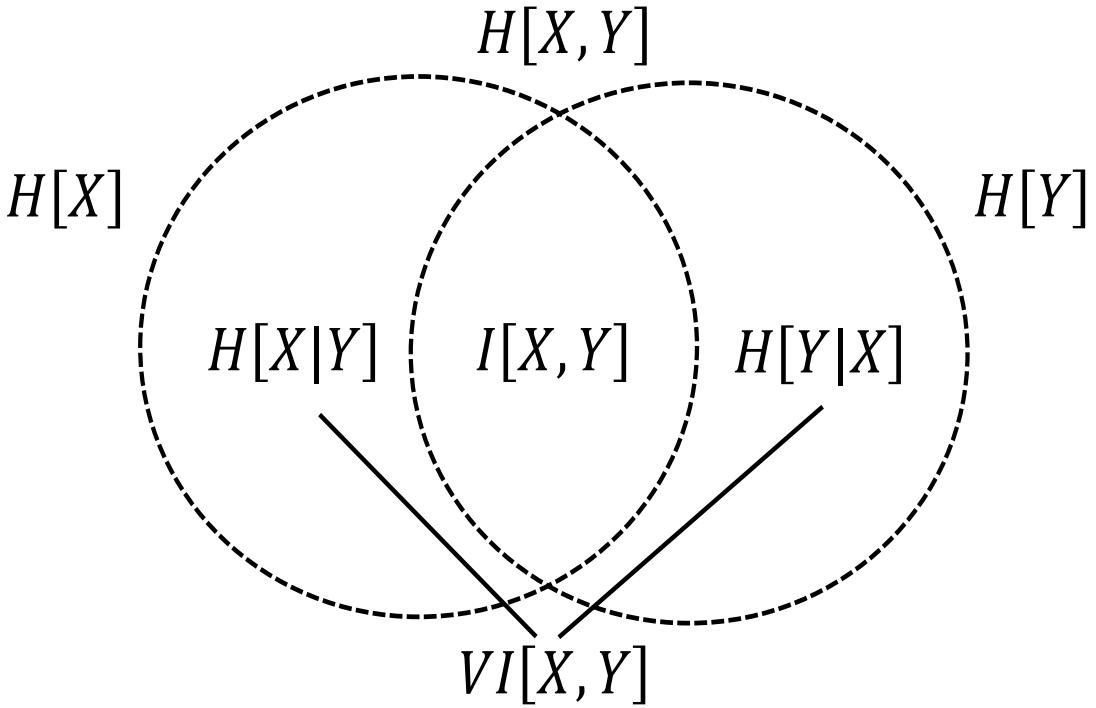
Variation of Information (3/3)

```
def varInfo(x,y,bXY,norm=False):
    cXY=np.histogram2d(x,y,bXY)[0]
    iXY=mutual_info_score(None,None,contingency=cXY) # mutual information
    hX=ss.entropy(np.histogram(x,bXY)[0]) # marginal
    hY=ss.entropy(np.histogram(y,bXY)[0]) # marginal
    vXY=hX+hY-2*iXY # variation of information
    if norm:
        hXY=hX+hY-iXY # joint
        vXY/=hXY # normalized variation of information
    return vXY
```

Python implementation for the estimation of $VI[X,Y]$:

- Pass the number of bins as bXY (needed for discretizing x and y)
- Pass norm=True if for $\widetilde{VI}[X,Y]$ (bounded between 0 and 1)

Information Correspondences



The correspondence between joint entropy, marginal entropies, conditional entropies, mutual information and variation of information

Discretization

Discretization (1/4)

- Throughout this section, we have assumed that random variables were discrete
- For the continuous case, we can quantize (coarse-grain) the values, and apply the same concepts on the binned observations
- Consider a continuous random variable X, with probability distribution functions $f_X[x]$. Shannon defined its (differential) entropy as

$$H[X] = -\int_{-\infty}^{\infty} f_X[x] \log\left[f_X[x]\right] dx$$

- The entropy of a Gaussian random variable X is $H[X] = \frac{1}{2}\log[2\pi e\sigma^2]$, thus $H[X] \approx 1.42$ in the standard Normal case

Discretization (2/4)

- One way to estimate $H[X]$ on a finite sample of real values is to divide the range spanning the observed values $\{x\}$ into B_X bins of equal size Δ_X, $\Delta_X = \frac{\max\{x\} - \min\{x\}}{B_X}$, giving us

$$H[X] \approx -\sum_{i=1}^{B_X} f_X[x_i] \log\left[f_X[x_i]\right] \Delta_X$$

  where $f_X[x_i]$ represents the frequency of observations falling within the *i*th bin.
- Let $p[x_i]$ be the probability of drawing an observation within the segment Δ_X corresponding to the *i*th bin
- We can approximate $p[x_i]$ as $p[x_i] \approx f_X[x_i]\Delta_X$, which can be estimated as $\hat{p}[x_i] = \frac{N_i}{N}$, where N_i is the number of observations within the *i*th bin, $N = \sum_{i=1}^{B_X} N_i$, and $\sum_{i=1}^{B_X} \hat{p}[x_i] = 1$

Discretization (3/4)

- This leads to a discretized estimator of entropy of the form

$$\widehat{H}[X] = -\sum_{i=1}^{B_X} \frac{N_i}{N} \log\left[\frac{N_i}{N}\right] + \log[\Delta_X]$$

$$\widehat{H}[X,Y] = -\sum_{i=1}^{B_X}\sum_{j=1}^{B_Y} \frac{N_{i,j}}{N} \log\left[\frac{N_{i,j}}{N}\right] + \log[\Delta_X \Delta_Y]$$

- From the estimators $\widehat{H}[X]$ and $\widehat{H}[X,Y]$, we can derive estimators for conditional entropies, mutual information and variation of information
- As we can see from these equations, results may be biased by our choice of B_X and B_Y

Discretization (4/4)

- For the marginal entropy case, Hacine-Gharbi et al. [2012] found that the following binning is optimal,

$$B_X = \text{round}\left[\frac{\zeta}{6} + \frac{2}{3\zeta} + \frac{1}{3}\right]$$

$$\zeta = \sqrt[3]{8 + 324N + 12\sqrt{36N + 729N^2}}$$

- For the joint entropy case, Hacine-Gharbi and Ravier [2018] found that the optimal binning is given by,

$$B_X = B_Y = \text{round}\left[\frac{1}{\sqrt{2}}\sqrt{1 + \sqrt{1 + \frac{24N}{1 - \hat{\rho}^2}}}\right]$$

where $\hat{\rho}$ is the estimated correlation between X and Y.

Python Implementation

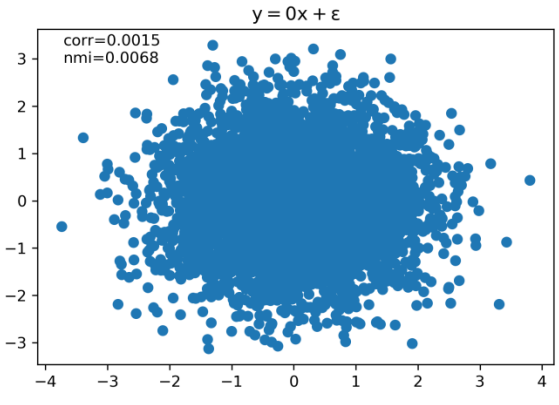
```
def numBins(nObs,corr=None):
   if corr is None: # univariate case
      z=(8+324*nObs+12*(36*nObs+729*nObs**2)**.5)**(1/3.)
      b=round(z/6.+2./(3*z)+1./3)
   else: # bivariate case
      b=round(2**-.5*(1+(1+24*nObs/(1.-corr**2))**.5)**.5)
   return int(b)
#--------------------------------------------------------------------------
bXY=numBins(x.shape[0],corr=np.corrcoef(x,y)[0,1])
```

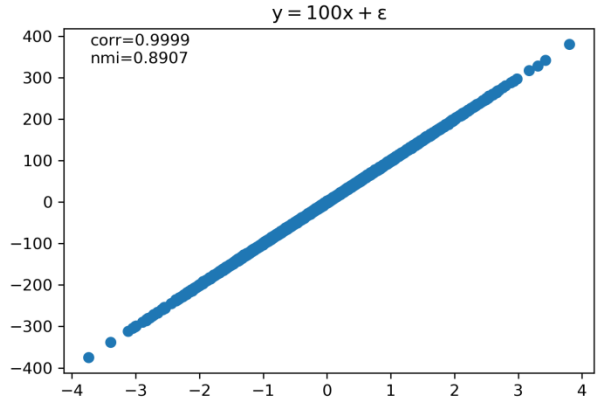
Use this function to estimate the bXY argument required by mutualInfo() and varInfo().

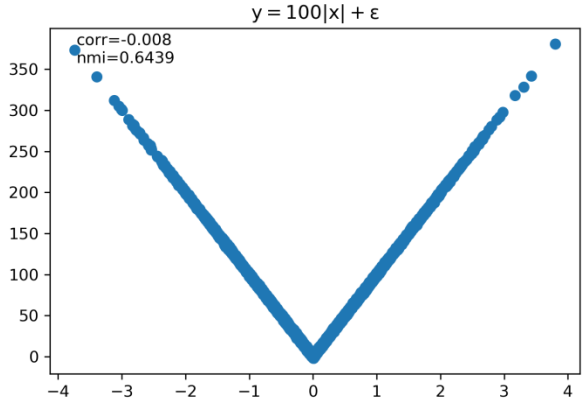
Numerical Examples

Does It Work?

- The normalized mutual information is the information theoretic analogue to linear algebra's correlation coefficient
- Consider two arrays x and e, of random numbers from a standard Gaussian distribution. We evaluate the normalized mutual information (NMI) and correlation between x and various y

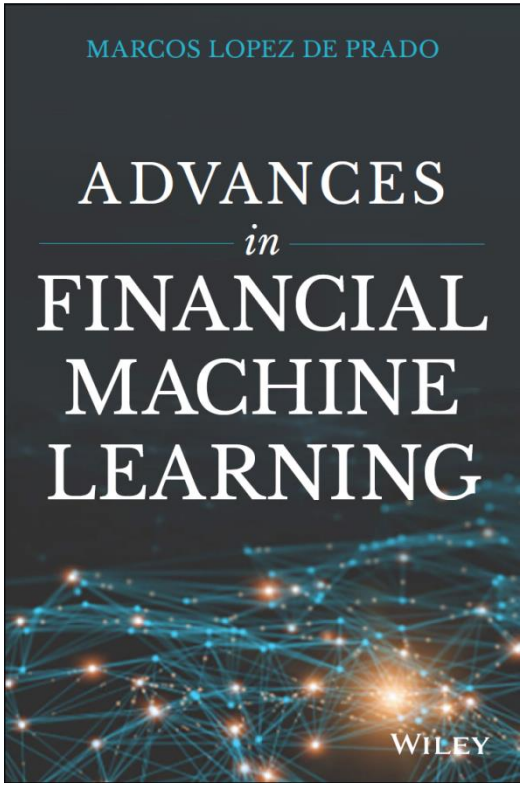






Correlations fail to recognize the strong relationship that exists between x and y when that relationship is non-linear. In contrast, mutual information recognizes that we can extract a substantial amount of information from x that is useful to predict y, and *vice versa*.

For Additional Details



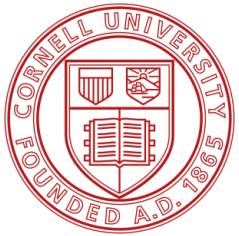
The first wave of quantitative innovation in finance was led by Markowitz optimization. Machine Learning is the second wave and it will touch every aspect of finance. López de Prado's Advances in Financial Machine Learning is essential for readers who want to be ahead of the technology rather than being replaced by it.
— Prof. **Campbell Harvey**, Duke University. Former President of the American Finance Association.

Financial problems require very distinct machine learning solutions. Dr. López de Prado's book is the first one to characterize what makes standard machine learning tools fail when applied to the field of finance, and the first one to provide practical solutions to unique challenges faced by asset managers. Everyone who wants to understand the future of finance should read this book.
— Prof. **Frank Fabozzi**, EDHEC Business School. Editor of The Journal of Portfolio Management.

Disclaimer

- The views expressed in this document are the authors' and do not necessarily reflect those of the organizations he is affiliated with.
- No investment decision or particular course of action is recommended by this presentation.
-

www.QuantResearch.org



Clustering

Prof. Marcos López de Prado
Advances in Financial Machine Learning
ORIE 5256

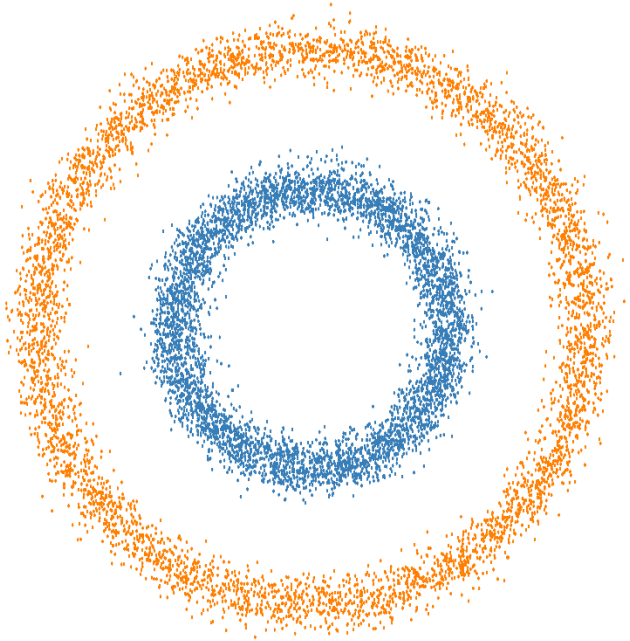
Key Points

- Many problems in finance require the clustering of variables or observations:
  - Factor investing, relative value analysis (e.g., forming quality minus junk portfolios)
  - Risk management, portfolio construction (e.g., deriving the efficient frontier)
  - Dimensionality reduction (e.g., decomposing bond return drivers)
  - Modelling of multicollinear systems (e.g., computing *p*-values)
- Despite its usefulness, clustering is almost never taught in Econometrics courses
  - None of the major Econometrics textbooks, and only a handful of academic journal articles, discuss the clustering of financial datasets
- In this seminar we review two general clustering approaches:
  - Partitional
  - Hierarchical
- Different features and/or similarity metrics will lead to different clusterings
  - **It is key to formulate the problem in a way that results have economic meaning and interpretability**

Introduction

What is Clustering?

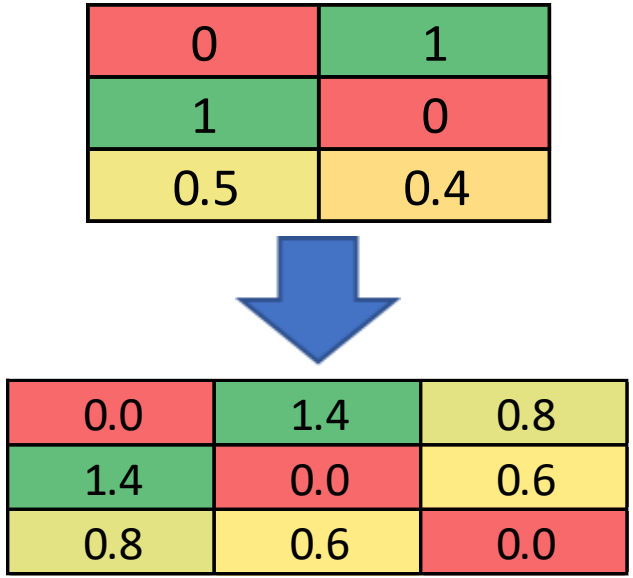
- A clustering problem is defined by a set of *objects* and a set of *features* associated with those objects
- Goal: Separate the objects into groups (called clusters) using the features, such that intra-group similarities are maximized, and inter-group similarities are minimized
- Clustering is a form of unsupervised learning
  - we do not provide examples to assist the algorithm in solving this task
- Clustering problems appear naturally in finance, at every step of the investment process



In this example, an agglomerative clustering algorithms recognizes two types of points based on 2 features (their coordinates)

Proximity Matrix

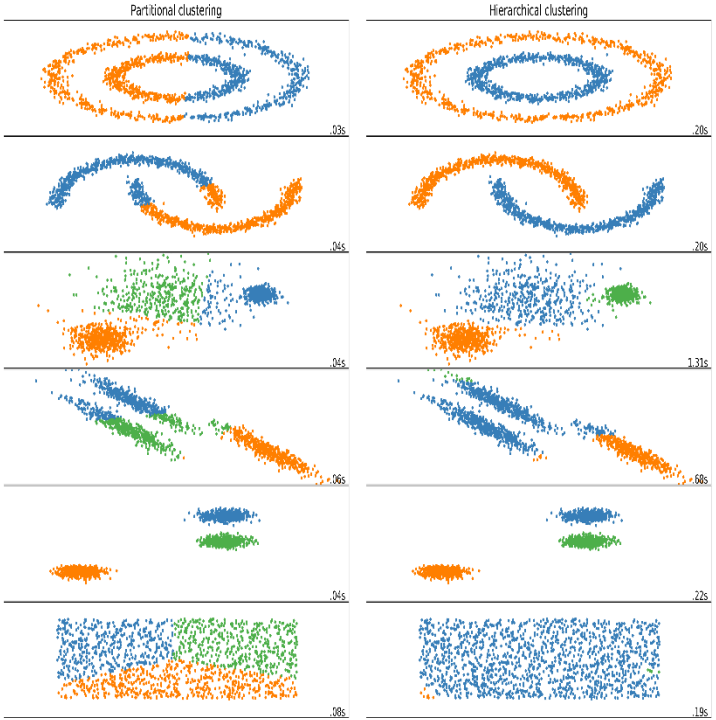
- Consider a data matrix X, of order NxF, where N is the number of objects and F is the number of features
- We use the F features to compute the proximity between the N objects, as represented by an NxN matrix
  - The proximity measure can indicate either *similarity* (e.g., correlation, mutual information) or *dissimilarity* (e.g., a distance metric).
  - It is convenient but not strictly necessary that dissimilarity measures satisfy the conditions of a metric
- When forming the proximity matrix, it is a good idea to standardize the input data, to prevent that one feature's scale dominates over the rest



A data matrix with 3 objects described by 2 features implies a 3x3 proximity matrix. The 3rd object is slightly more similar to the 2nd than to the 1st

Main Types of Clustering Methods

- There are two main classes of clustering algorithms: partitional and hierarchical
  - **Partitional** techniques create a one-level (un-nested) partitioning of the objects (each object belongs to one cluster, and to one cluster only)
  - **Hierarchical** techniques produce a nested sequence of partitions, with a single, all-inclusive cluster at the top and singleton clusters of individual points at the bottom.
    - Hierarchical clustering algorithms can be **divisive** (top-down) or **agglomerative** (bottom-up)
- By restricting the growth of a tree, we can derive a partitional clustering from any hierarchical clustering
  - However, one cannot generally derive a hierarchical clustering from a partitional one



Comparison of partitional vs hierarchical clustering on different datasets

A Partitional Algorithm: K-Means

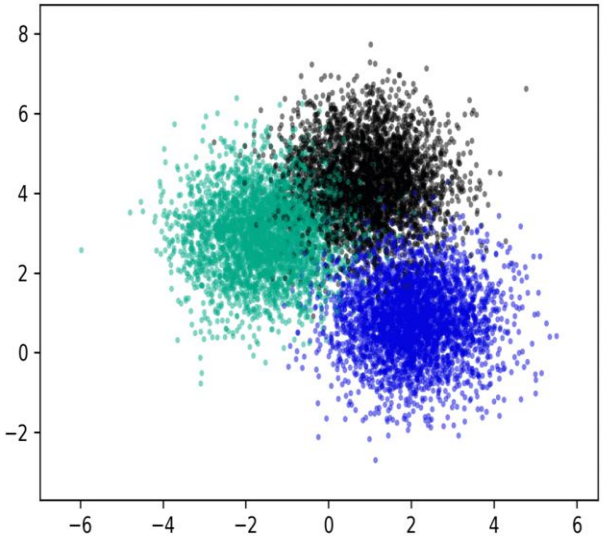
The Algorithm

- K-Means is a vector quantization model
  - It attempts to split the samples (rows) of X into a pre-determined number of clusters K

1. Initialize a random set of K centroids, $\{\mu_k\}_{k=1,\dots,K}$
2. Assign each sample X_n to one cluster, such that the within-clusters variance is minimized

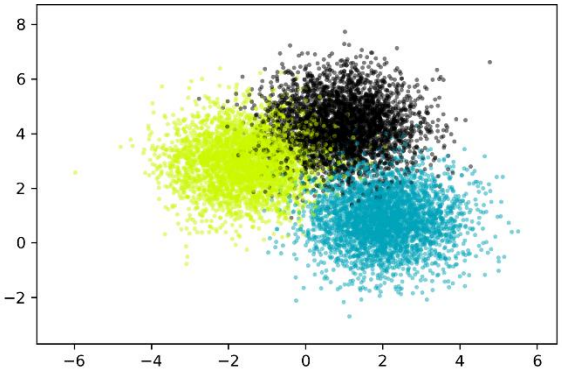
$$\sum_{n=1}^{N} \min_{k\in[1,K]} [\|X_n - \mu_k\|^2]$$

3. Update $\{\mu_k\}$, based on the clusters from (2)
4. Repeat steps (2)-(3) until convergence

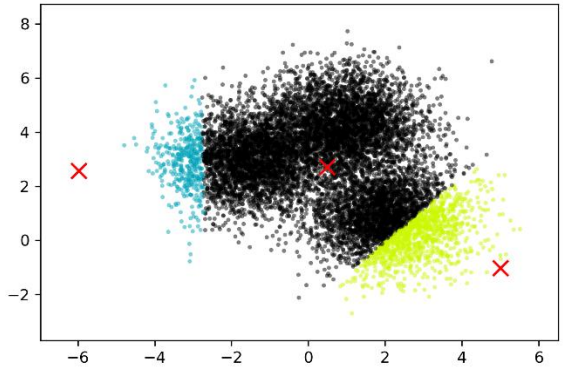


Convergence of K-Means on 3 blobs

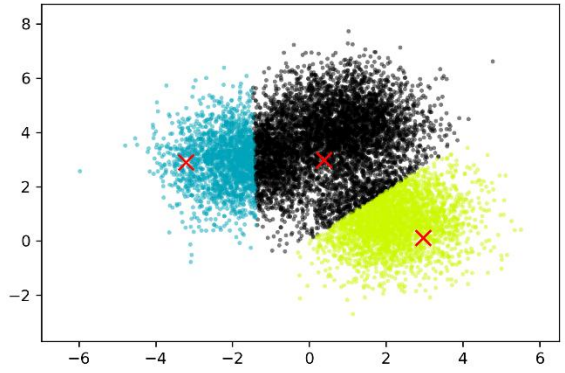
The Assignment-Update Cycle



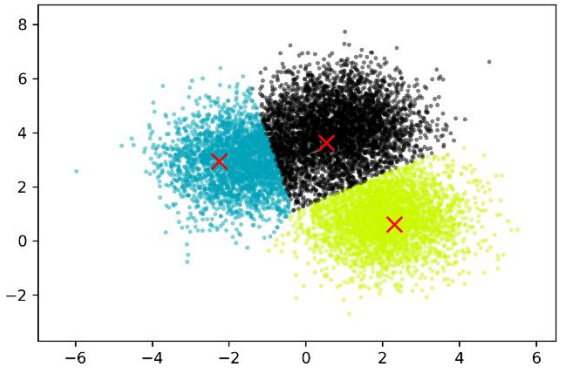
Original data



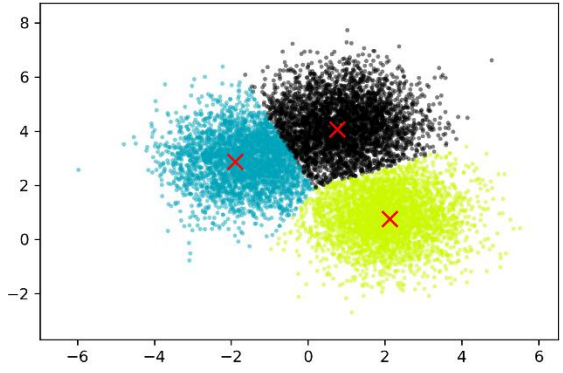
Iteration 1



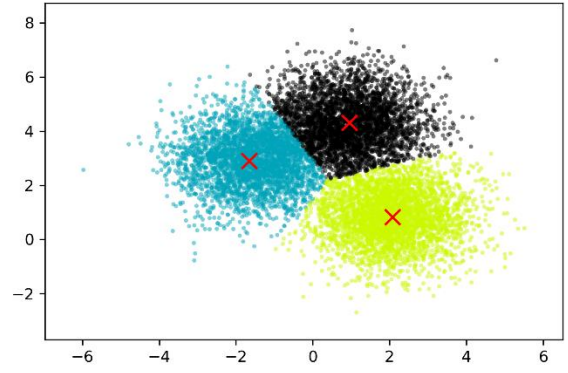
Iteration 2



Iteration 3



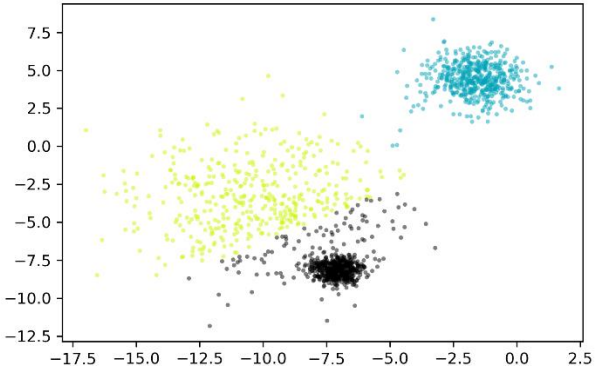
Iteration 4



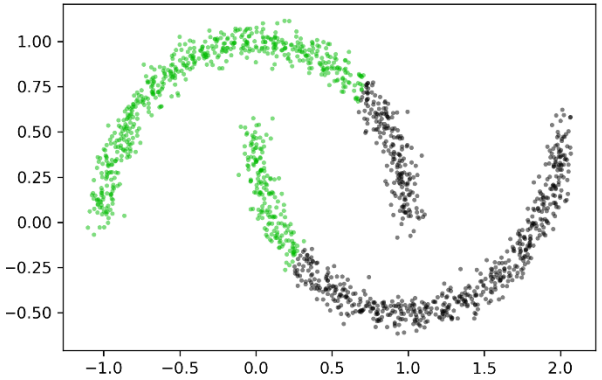
Iteration 5

A Few Considerations

- K-Means assumes that the clusters are convex, isotropic, and with similar variance
  - Features should be standardized prior to clustering
  - Other algorithms may perform better when clusters are elongated or irregular
- Within-cluster variance is not a normalized metric
  - Curse of dimensionality: When X has many columns, variances are inflated, and outcomes may be biased. One solution is to apply a dimensionality reduction technique (e.g., PCA) prior to clustering
- K-Means will always converge, however the outcome may be a local minimum
  - One solution is to run multiple instances in parallel, with different seed centroids



Three blobs with difference variance



Two anisotropic blobs

Python Implementation

```
import matplotlib.pyplot as mpl,matplotlib.cm as cm
from sklearn import datasets
from sklearn.cluster import KMeans
X,_=datasets.make_blobs(n_samples=10000,n_features=2,centers=3,random_state=0)
clusterer=KMeans(n_clusters=3).fit(X)
colors=cm.nipy_spectral(clusterer.labels_.astype(float)/n_clusters)
mpl.scatter(X[:,0],X[:,1],marker='.',s=30,lw=0,alpha=.5,c=colors,edgecolor='k’)
mpl.savefig('kmeans.png',dpi=300)
```

The above code generates a features matrix X, with values centered around 3 blobs. Then applies K-Means on the X matrix, targeting 3 clusters, and plots the predicted clusters.

A Hierarchical Algorithm: Agglomerative Clustering

Agglomerative Algorithm

1. Apply a distance metric to X
2. Combine into a cluster the pair with lowest distance
   - The pair can be composed of two items, two clusters, or one item and a cluster
3. Reduce the distance matrix
   a. Remove the 2 rows and columns associated with the pair
   b. Apply a **linkage criterion** to determine the distance between the new cluster and the rest of objects, e.g.:
      - Single linkage: minimum distance to any object in the pair
      - Complete linkage: maximum distance to any object in the pair
4. Repeat 2 and 3 until the distance matrix has been reduced to only one object

0.0	1.4	0.8
1.4	0.0	0.6
0.8	0.6	0.0

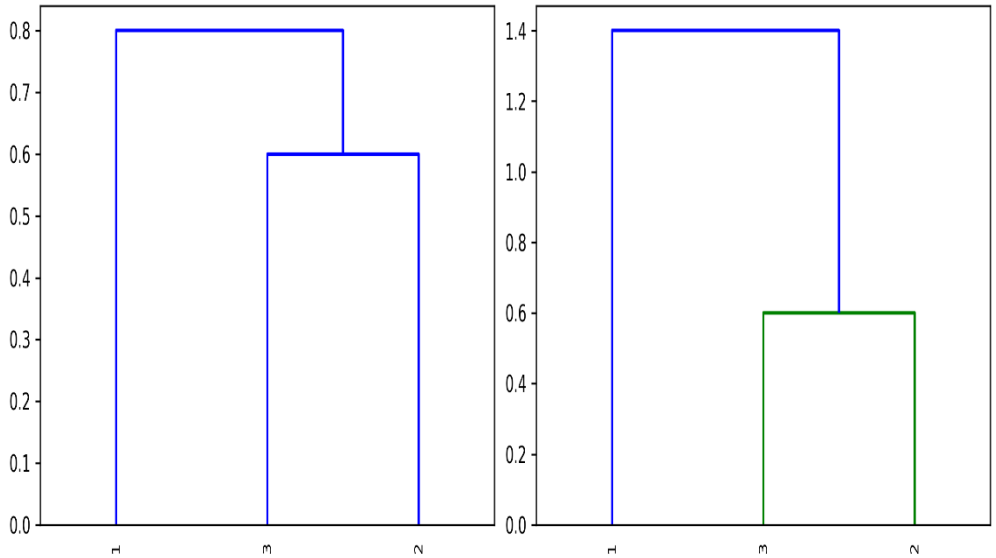
0.0	0.8
0.8	0.0

0.0

Reduction of the distance matrix: First we combine objects 2 and 3. Applying a single-linkage criterion, we determine that the distance between the cluster and object 1 is 0.8

Dendrogram

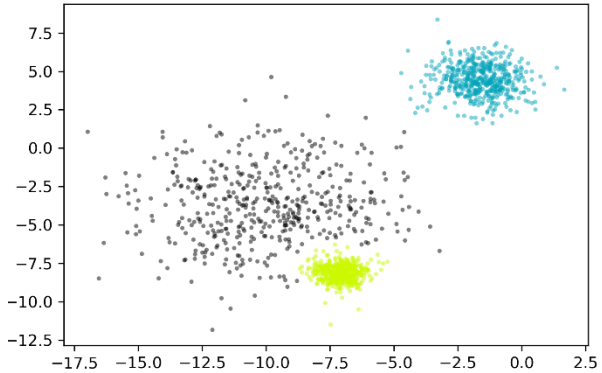
- A dendrogram is a tree graph that displays the hierarchical composition of the clusters
- The y-axis indicates the distance between the two objects that form a new cluster
- A linkage matrix characterizes a dendrogram
  - For N items, a linkage matrix has $N - 1$ rows (one row per cluster)
  - Three columns:
    - Integer identifying object 1
    - Integer identifying object 2
    - Distance between objects 1 and 2 (based on linkage criterion)



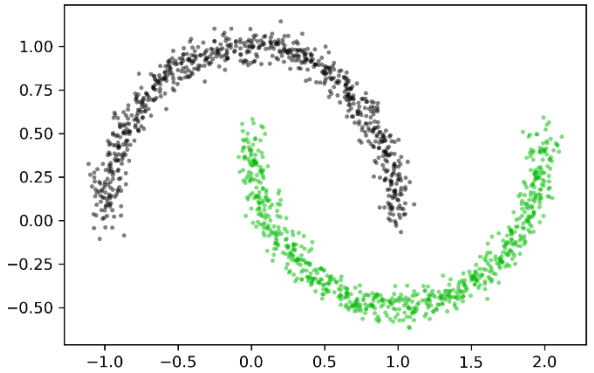
Dendrograms associated with the previous example, using the single-linkage (left) and complete-linkage (right) criteria. Note how the second clustering occurs at the distance of 0.8 for single-linkage, but 1.4 for complete-linkage

A Few Considerations

- Hierarchical algorithms can handle clusters that are non-convex, anisotropic, with unequal variance
  - This includes clusters within clusters
- Hierarchical algorithms allow connectivity constraints
  - Connectivity constraints cluster together only adjacent points. This links together points even if the centroid is not part of the cluster
- However, hierarchical algorithms may not handle properly elongated blobs
  - One solution is to orthogonalize the features (e.g., PCA without dimensionality reduction) prior to clustering
- The appropriate linkage method can be chosen via cross-validation, or [cophenetic correlation]



Three blobs with unequal variance



Two anisotropic blobs

Python Implementation

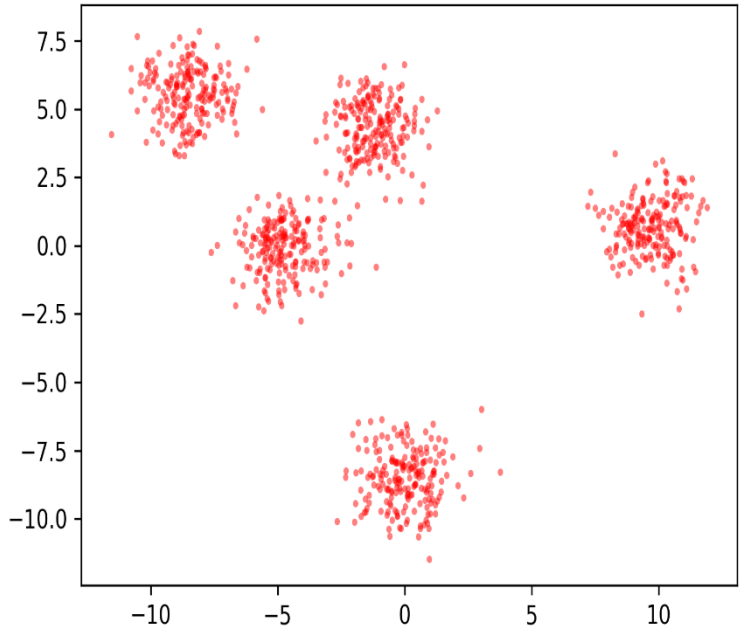
```
import matplotlib.pyplot as mpl,seaborn as sns
import scipy.cluster.hierarchy as sch
link=sch.linkage(ssd.squareform(dist0,force='tovector'),optimal_ordering=True,method='single')
dist1=dist0.iloc[sch.leaves_list(link),sch.leaves_list(link)] # apply clusters
sns.heatmap(dist1,cmap='viridis',xticklabels=False,yticklabels=False)
mpl.savefig('clusters.png',dpi=300)
mpl.clf();mpl.close() # reset pylab
dendro=sch.dendrogram(link,leaf_rotation=90.,leaf_font_size=8.,
                                    labels=dist.columns,distance_sort=True)
mpl.savefig('dendro.png',dpi=300)
mpl.clf();mpl.close() # reset pylab
```

The above code takes a distance matrix `dist0` (a dataframe), derives the linkage matrix using a single-linkage criterion, applies the clusters to produce a reordered `dist1` distance matrix, plots `dist1`, and plots the associated dendrogram. See also Scikit-Learn's implementation.

Optimal Number of Clusters

Cluster Scoring

- In order to determine the optimal number of clusters, we first need to define a function that scores the output of a scoring algorithm
- In general, there are two types of clustering scoring functions:
  a) External: those that require ground-truth labels
  b) Internal: those that don't require it
- Because clustering is an unsupervised learning problem, internal scores are more natural. Three of the most used internal scoring functions are:
  - Calinski-Harabasz index (or variance ratio)
  - Davies-Boulding index
  - Silhouette scores



How many blobs are there? On 2-D, this is an easy question for a human. On higher dimensions, machines are more likely to win

Calinski-Harabasz Index (Variance Ratio)

Given N objects, centered around c, and grouped into K clusters

- The within-cluster dispersion (W_K) is

$$W_K = \sum_{k=1}^{K} \sum_{x \in C_k} (x - c_k)^T (x - c_k)$$

  where C_k is the set of objects in cluster k, c_k is the center of cluster k, and x is coordinates of an object.

- The between-cluster dispersion (B_K) is

$$B_K = \sum_{k=1}^{K} N_k (c_k - c)^T (c_k - c)$$

  where N_k is the number of objects in C_k.

- The Variance Ratio is defined as

$$s = \frac{B_K}{W_K} x \frac{N - K}{K - 1}$$

Davies-Bouldin Index

- This index is defined on K cluster as

$$s = \frac{1}{K}\sum_{i=1}^{K}\max_{i\neq j} R_{i,j}$$

  where $R_{i,j}$ is the (symmetric, non-negative) similarity between two clusters.

- A common choice is $R_{i,j} = \frac{\delta_i+\delta_j}{d_{i,j}}$, where
  - δ_i is the average distance between objects in cluster i and that cluster's centroid
  - $d_{i,j}$ is the distance between the centroids of clusters i and j
- Accordingly, this index can be interpreted as the average similarity between each cluster C_i and its most similar cluster C_j
  - Note: A lower index means better clustering. The more dissimilar the clusters, the better the clustering

Silhouette Scores

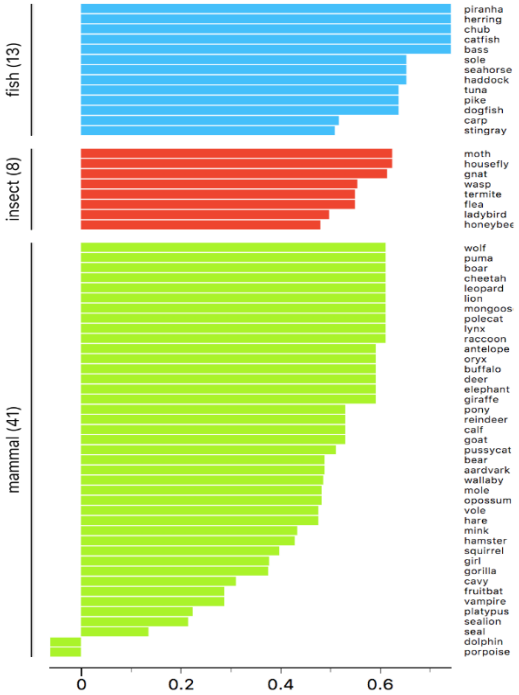
- Silhouette scores are defined for each sample, $\{s_n\}_{n=1,\ldots,N}$

$$s_n = \frac{b_n - a_n}{\max\{a_n, b_n\}}$$

where

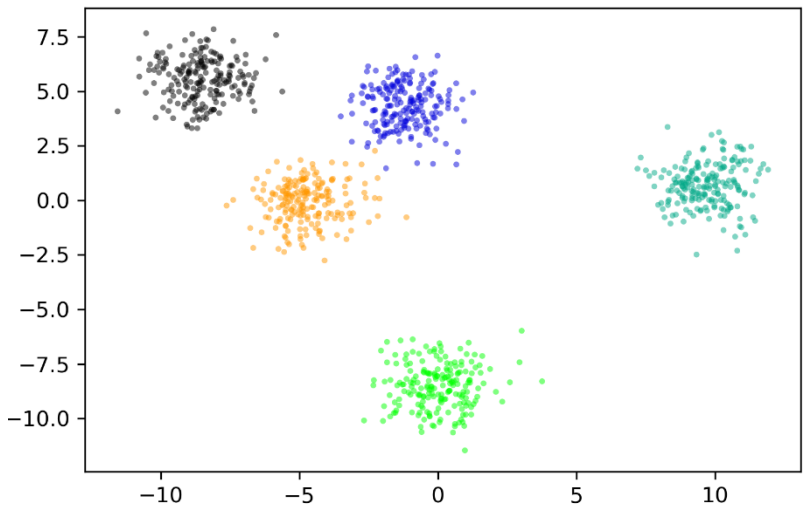
  - a_n: mean distance between object n and other objects in its cluster
  - b_n: mean distance between object n and objects in the nearest cluster

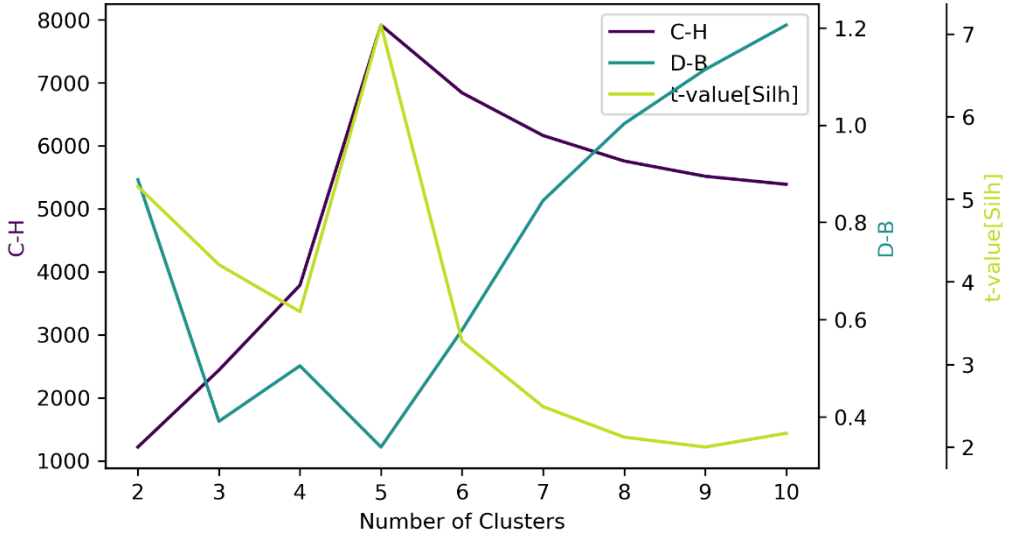
- Advantages:
  - The scores are bounded: $-1 \leq s_n \leq 1$
  - Because we have one score per sample, we can reallocate specific objects to better clusters
  - Clusters with average $s_n \approx 0$ are overlapping, and could be merged
  - We can use $\{s_n\}$ to derive a distribution of scores, and make inference (p-values). For example, we can compute the t-value, $s = \frac{\mathrm{E}[\{s_n\}]}{\sqrt{\mathrm{V}[\{s_n\}]}}$



Silhouette scores for dolphins highlight that there is something "fishy" about these mammals

Optimal Number of Clusters



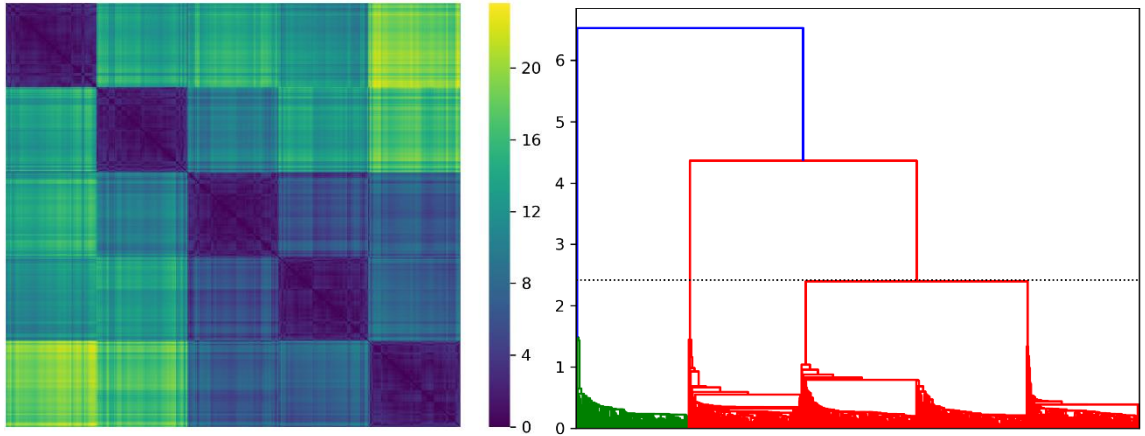


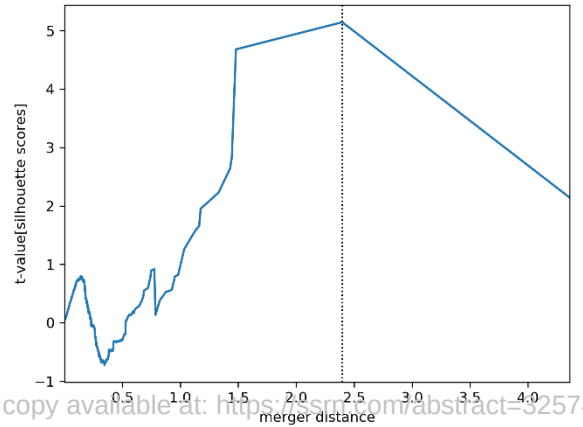
How many blobs are there? First, we applied the K-Means algorithm, setting as target 2,3,…,10 clusters. Second, we computed the three scores on each of the resulting groups for 2,…,10 clusters. Third, we plotted the scores for each number of clusters.

All scoring functions agree that the optimal number of clusters is 5. D-B points at the possibility of 3 clusters, which is not an unreasonable answer considering that 2 blobs are in close proximity.

Deriving Partitions from a Hierarchical Algorithm

- Dendrograms give us the distance at which each merger has taken place
- We can evaluate the scoring function at the distance of each merger, and derive the optimal merging distance
- That distance implies the number of clusters
- The procedure is fast, however not necessarily the most accurate





If we apply a hierarchical clustering method on the previous 5 blobs example, we recover 4 clusters (with the right answer, 5 clusters, being a close alternative)

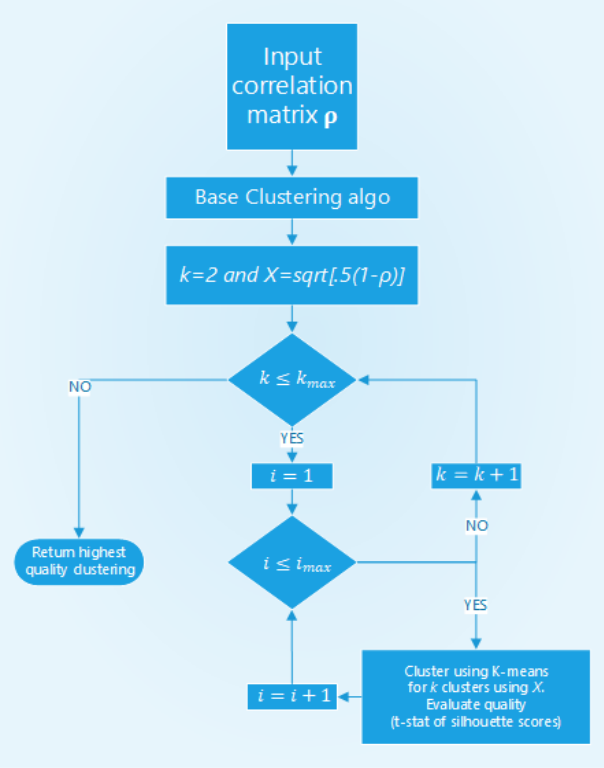
The ONC Algorithm (1/4)

- Clustering steps (find here an implementation):
  1. Base level clustering
  2. Recursive improvements to clustering

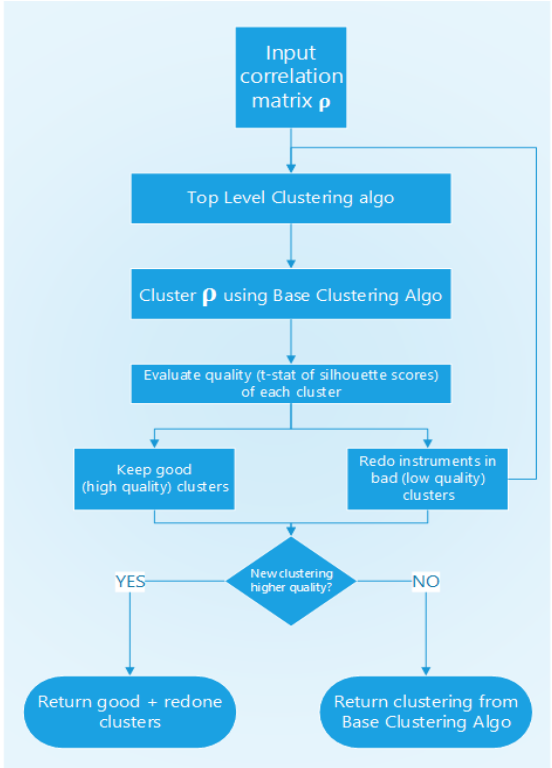
- Base Clustering:
  1. For each k in $2, \ldots, N-1$, and l in $1, \ldots, n_{init}$:
     - Apply K-Means to extract k clusters using distance matrix
     - Evaluate silhouette scores s_n, $n = 1, \ldots, N$, for the clustering
     - Evaluate quality score $q_{k,l} = \frac{\mathrm{E}[s_n]}{\sqrt{\mathrm{V}[s_n]}}$
  2. Choose optimal clustering, with $K = \mathrm{argmax}_k \left\{ \max_l \{q_{k,l}\} \right\}$

- Recursive improvement to clustering:
  1. Run Base Clustering to derive K
  2. Keep clusters where $q_i \geq \mathrm{E}[q_i]$
     - Recursively rerun Base Clustering for rest of clusters
     - Keep new clustering only if re-clustering improves average quality score

The ONC Algorithm (2/4)



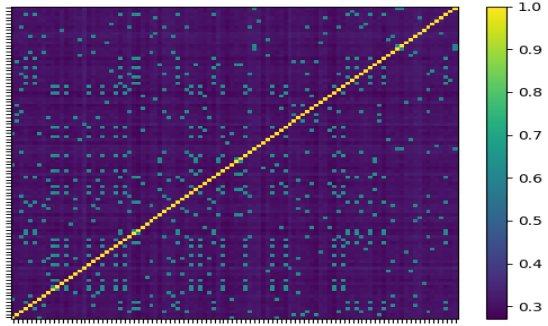
Base clustering



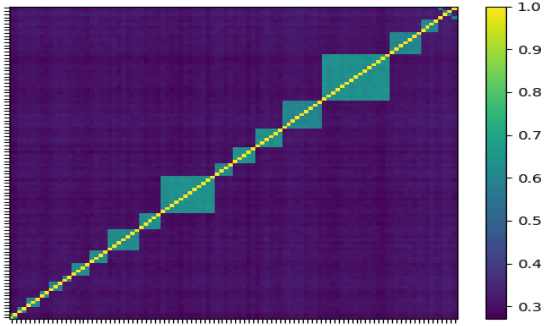
Higher-level clustering

The ONC Algorithm (3/4)

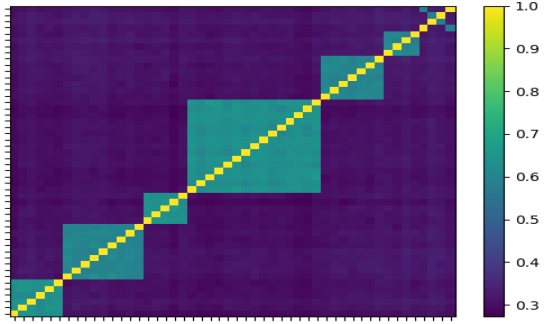
Initial Scrambled Random Block Covariance



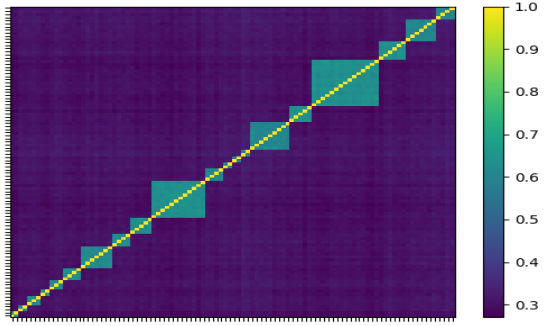
After Base Clustering



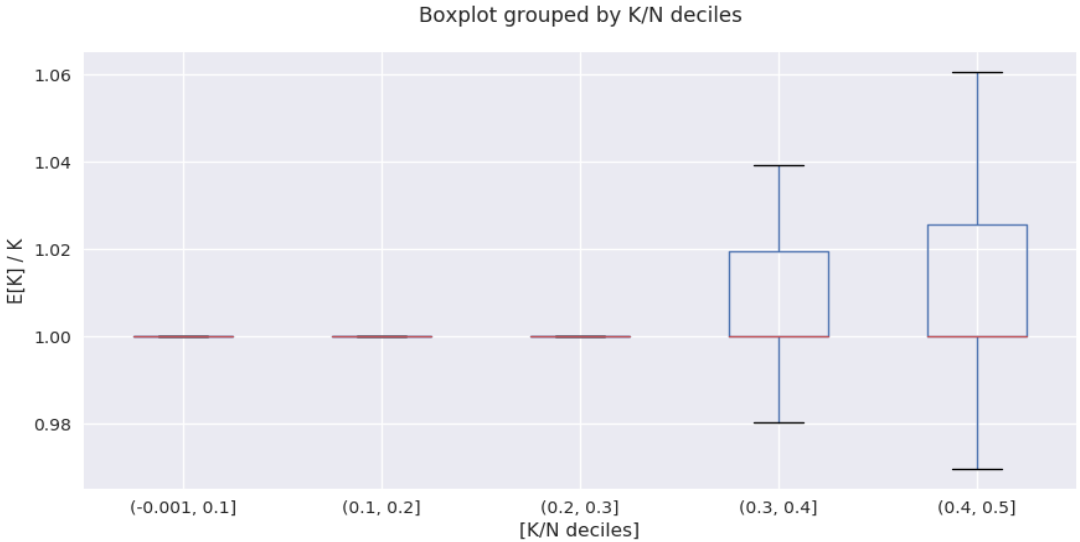
Recursively Re-evaluate Clustering



Final Clustering



The ONC Algorithm (4/4)



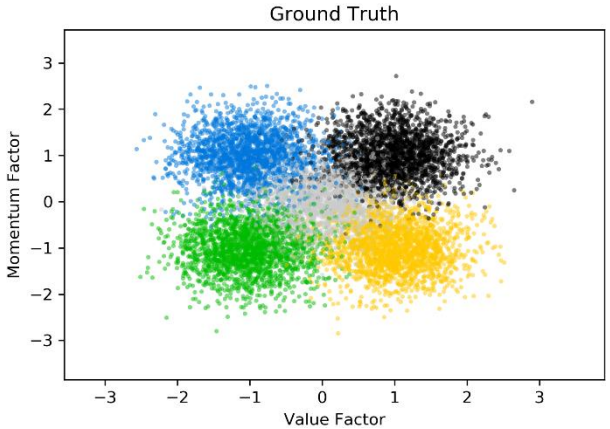
- Create a random block covariance matrix
  - Size NxN
  - K blocks of random size
- Add global noise
- Run Clustering
- Compare expected cluster count K to number of estimated clusters

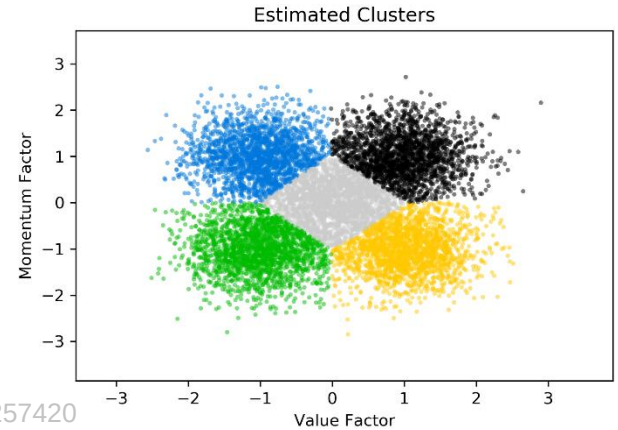
The boxplots show the results from these simulations. In particular, for $\frac{K}{N}$ in a given decile, we display the boxplot of the ratio of K predicted by the clustering to the actual $\mathrm{E}[K]$ predicted by clustering. Ideally, this ratio should be near 1. Monte Carlo simulations confirm that ONC is effective at recovering the ground truth.

A Few Use Cases

Factor Investing / Relative Value

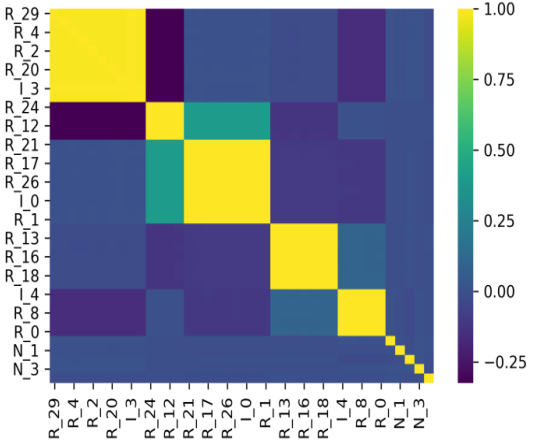
- Factor investing attempts to price assets that share some common characteristics
- Traditionally, economists group assets according to a single characteristic
  - E.g.: value, size, momentum, quality, liquidity, carry, etc.
- This misses known interaction effects, such as value vs. momentum, and hierarchical dependencies
- A natural solution is to cluster assets on multiple characteristics (features), and let the algorithm find the optimal number of clusters
  - We can then evaluate the performance of each cluster, and assess whether the risk-premium is statistically significant
  - This approach is also useful for relative value strategies

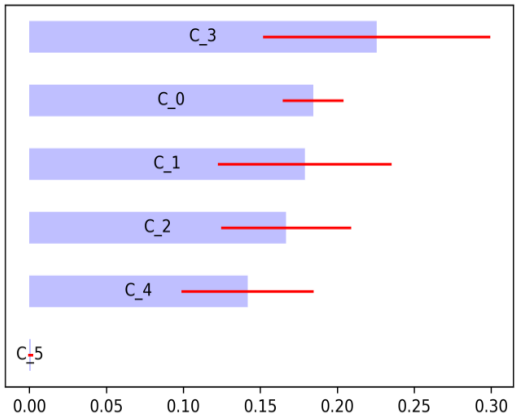




Feature Importance Analysis

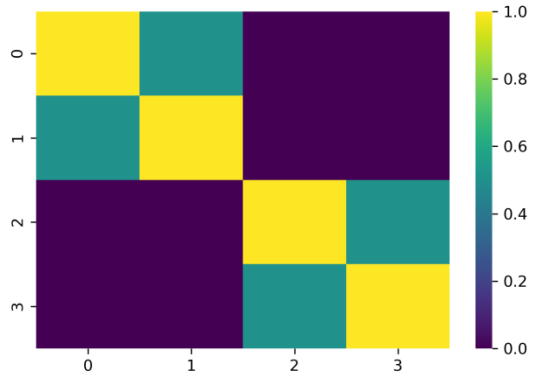
- Consider a binary random classification problem composed of 40 features, where 5 are informative, 30 are redundant, and 5 are noise
  - **Informative features** (marked with the "I_" prefix) are those used to generate labels
  - **Redundant features** (marked with the "R_" prefix) are those that are formed by adding Gaussian noise to a randomly chosen informative feature
  - **Noise features** (marked with the "N_" prefix) are those that are not used to generate labels
- A clustering algorithm prevents that *substitution effects* bias the MDA or MDI analyses, by
  - finding the optimal number of clusters (see top right plot)
  - bundling together the features that are redundant to an informative one (see bottom right plot)





Portfolio Construction

- When K securities form a correlation cluster, convex optimization methods (Markowitz, Black-Litterman, etc.) cannot distinguish between them
- This is an example of signal-induced instability
- One solution is to apply the NCO algorithm:
  1. cluster the correlation matrix,
  2. compute the optimal intra-cluster allocations,
  3. compute the optimal inter-cluster allocations,
  4. derive the optimal weights as the dot-product of (2) and (3)
- Steps (1) and (2) allow us to transform a "Markowitz-cursed" problem into a well-behaved problem

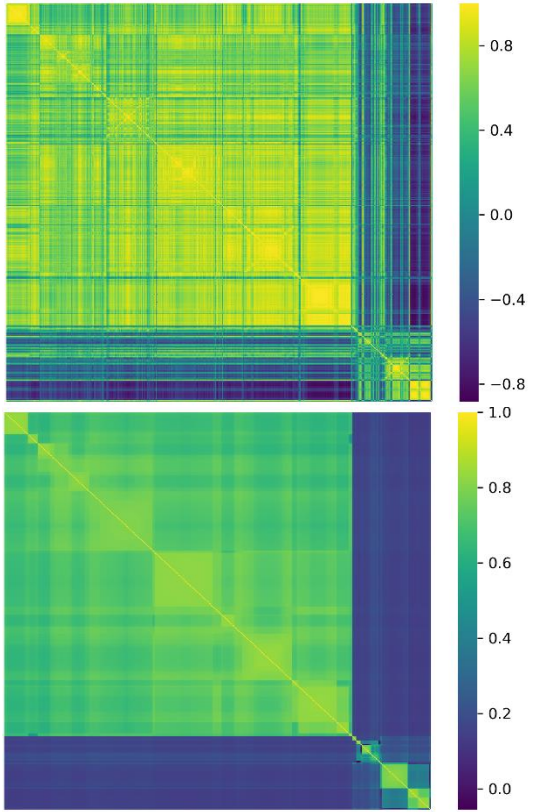


	Markowitz	NCO
Raw	7.02E-02	3.17E-02
Shrunk	6.54E-02	5.72E-02

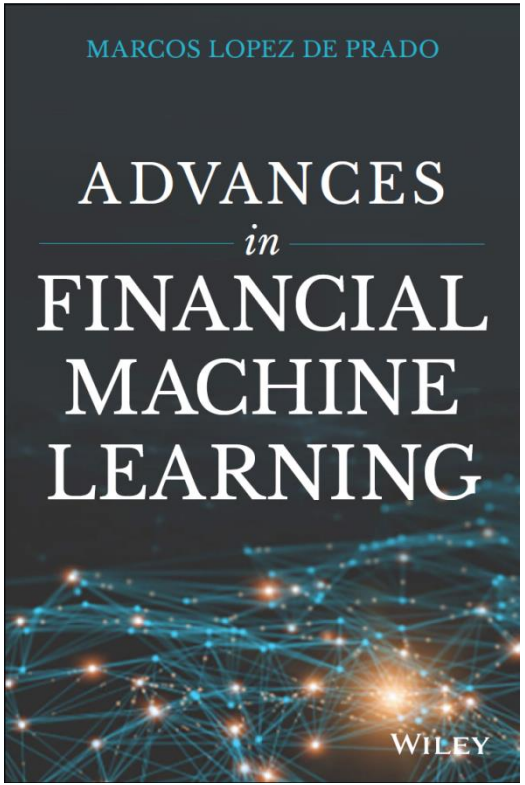
NCO computes the maximum Sharpe ratio portfolio with 45.17% of Markowitz's RMSE, i.e. a 54.83% reduction in RMSE

Forward-Looking Correlation Matrices

- It is widely acknowledged that empirical correlations have:
  a) poor numerical properties that lead to unreliable estimators; and
  b) poor predictive power
- Additionally, factor-based correlation matrices have their own caveats. In particular, estimated factors are typically:
  - non-hierarchical, and
  - do not allow for interactions at different levels
- We can derive a forward-looking correlation matrix from a knowledge graph
  - On the right, plots of a correlation matrix before (top) and after (bottom) imposing a theory-implied structure (GICS)
  - Adding signal through a theoretical dendrogram makes correlation patterns smoother and less noisy, while preserving the hierarchical structure



For Additional Details



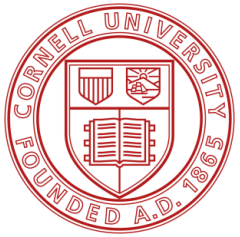
The first wave of quantitative innovation in finance was led by Markowitz optimization. Machine Learning is the second wave and it will touch every aspect of finance. López de Prado's Advances in Financial Machine Learning is essential for readers who want to be ahead of the technology rather than being replaced by it.
— Prof. **Campbell Harvey**, Duke University. Former President of the American Finance Association.

Financial problems require very distinct machine learning solutions. Dr. López de Prado's book is the first one to characterize what makes standard machine learning tools fail when applied to the field of finance, and the first one to provide practical solutions to unique challenges faced by asset managers. Everyone who wants to understand the future of finance should read this book.
— Prof. **Frank Fabozzi**, EDHEC Business School. Editor of The Journal of Portfolio Management.

Disclaimer

- The views expressed in this document are the authors' and do not necessarily reflect those of the organizations he is affiliated with.
- No investment decision or particular course of action is recommended by this presentation.
-

www.QuantResearch.org



Three Machine Learning Solutions to the Bias-Variance Dilemma

Prof. Marcos López de Prado

Advances in Financial Machine Learning

ORIE 5256

Background

- The error of a supervised learning algorithm has three components:
  - **Bias**: error caused by erroneous assumptions
  - **Variance**: error caused by overfitting
  - **Noise**: irreducible error, due to the random nature of the variable
- A central problem in classical statistics (e.g., Econometrics) is to find the MVUE
  - Fit the estimator with **m**inimum **v**ariance among all **u**nbiased **e**stimators
- A central problem in Machine Learning (ML) is to minimize overall errors
  - Capture the regularities in the data (low bias), while
  - generalizing well to unseen data (low variance)
- This is a key difference between Econometrics and ML
  - Econometric models are willing to sacrifice performance in exchange for zero bias
  - ML models find a balance between bias and variance, in order to maximize performance
- In this seminar, we will study how ML finds this balance

The Problem

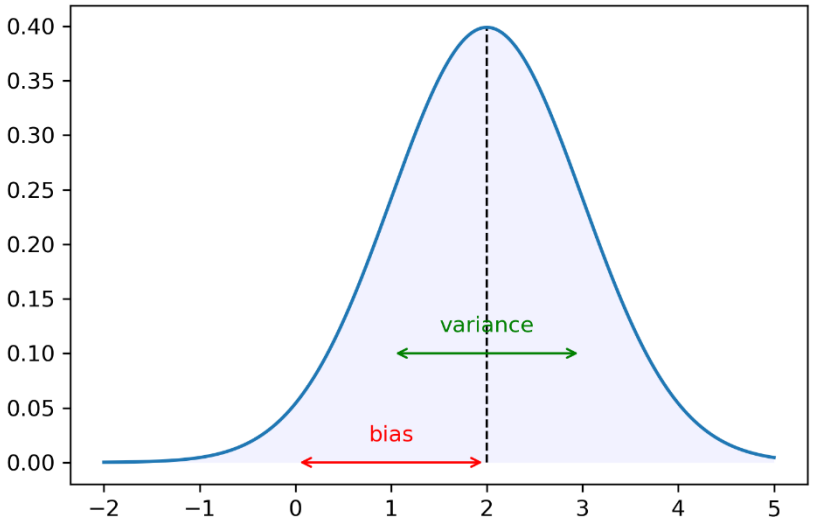
Error Decomposition

- Consider N observations, $\{(x_n, y_n)\}_{n=1,\ldots,N}$, and a function $f[x_n]$ that predicts y_n, such that errors $\varepsilon_n = y_n - f[x_n]$ are unpredictable, with $\varepsilon_n \sim N[0, \sigma_\varepsilon^2]$
- This "true model" $f[x_n]$ is unknown to us, so instead we propose a statistical model $\hat{f}[x_n]$ that approximates $f[x_n]$, with error $e_n = y_n - \hat{f}[x_n]$
- In fitting the statistical model, we would like to minimize the mean squared *true* error $\mathrm{E}\left[\left(f[x_n] - \hat{f}[x_n]\right)^2\right]$. But because $f[x_n]$ is unknown to us, the best we can do is to minimize the mean squared *empirical* error (MSE) associated with $\hat{f}[x_n]$,

$$MSE = \mathrm{E}[e_n^2] = \underbrace{\left(\mathrm{E}\left[f[x_n] - \hat{f}[x_n]\right]\right)^2}_{bias^2} + \underbrace{\mathrm{V}\left[\hat{f}[x_n]\right]}_{variance} + \underbrace{\sigma_\varepsilon^2}_{noise}$$

Visualizing Bias and Variance

- We generate 10,000 trial datasets, $\{(x_n, y_n)\}_{n=1,\ldots,N}$, where $N = 1{,}000$
- We generate one out-of-sample (OOS) pair,$\{(\bar{x}, \bar{y})\}$
- For each trial dataset, we
  - fit the statistical model, $\hat{f}[x_n]$
  - compute the true error, $\epsilon = f[\bar{x}] - \hat{f}[\bar{x}]$
    - Note: The true error does not incorporate noise, because we use $f[\bar{x}]$ instead of $\bar{y}$
- Then,
  - the estimated bias is $\mathrm{E}[\epsilon]$ across all trials
  - the estimated variance is $\mathrm{V}[\epsilon]$ across all trials



Monte Carlo experiments help us visualize the bias and variance associated with a particular statistical model, $\hat{f}[x_n]$.

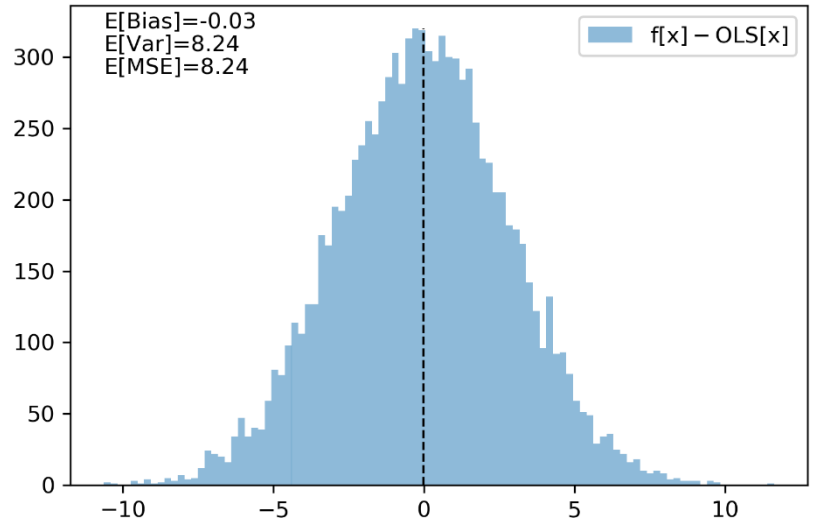
Gauss-Markov Theorem (GMT)

- GMT assumes that:
  - $f[x_n]$ is linear
  - $\hat{f}[x_n]$ incorporates all k variables
  - $\{e_n\}$ is white noise, and uncorrelated to $\{x_n\}$
- Then, the ordinary least squares (OLS) estimator has the lowest $\mathrm{V}\left[\hat{f}[x_n]\right]$ among all *linear unbiased* estimators

$$\text{bias:} = f[x_n] - \mathrm{E}\left[\hat{f}[x_n]\right] = 0$$

$$\text{variance:} = \mathrm{V}\left[\hat{f}[x_n]\right] \propto \frac{\sigma_\varepsilon^2}{N} k$$

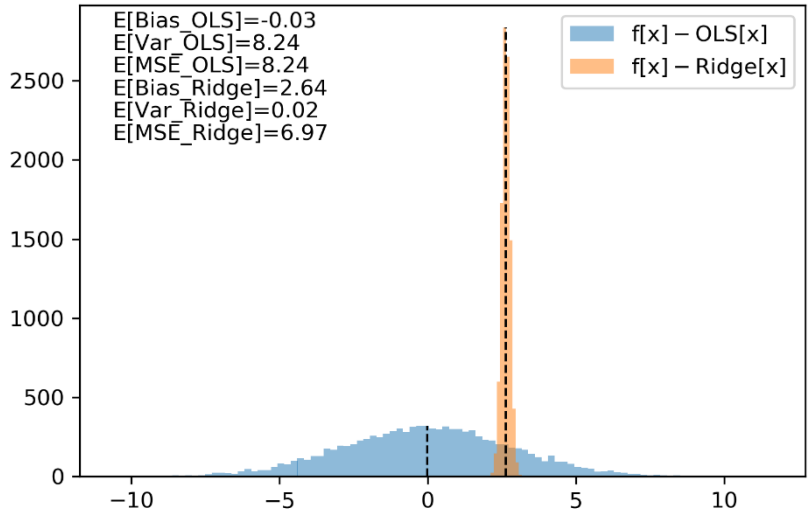
- GMT is the fundamental reason why OLS is the statistical workhorse in Econometrics



When the true model is linear, there are no omitted variables, and errors are white noise, then OLS gives us the MVUE (minimum variance estimate among all linear unbiased).

When GMT Assumptions Are True

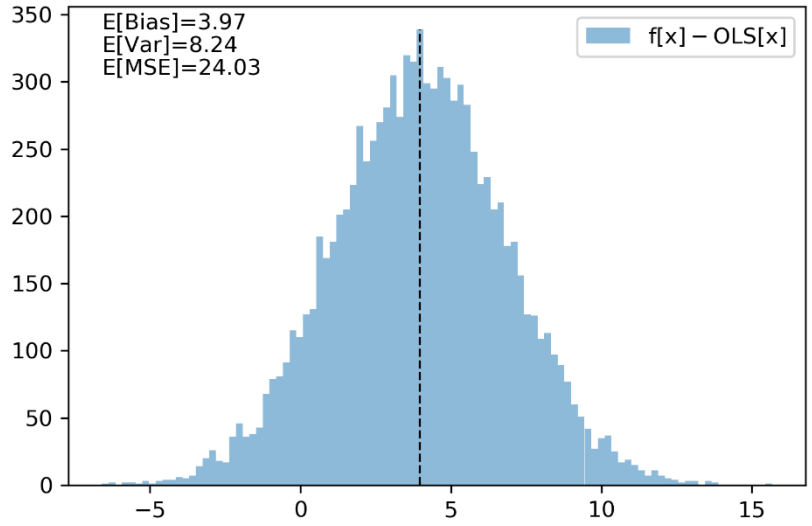
- GMT tells us that OLS has minimum variance among *linear unbiased* estimators, however
  - there are non-linear biased estimators with lower variance (e.g., James-Stein estimator)
  - there are linear biased estimators with lower MSE (e.g., Ridge, LASSO)
- **Even if all GMT assumptions are correct, OLS does not necessarily yield the minimum MSE**
- Econometrics' reliance on unbiased estimators (including, but not limited to OLS) is not justified by model performance



In the above experiment, all GMT assumptions are true, and yet OLS underperforms a biased ML estimator (Ridge), because one of the regressors is spurious. As a result of selection bias, many financial studies include spurious regressors, thus setting OLS for delivering poor predictions.

When GMT Assumptions Are False

- GMT does not apply under three common assumption violations in financial datasets:
  - **Specification errors**: The true model is not linear in its parameters, and the statistical model omits variables or incorporates spurious variables
  - **Stochastic regressors**: The explanatory variables are measured with error, are endogenous, are mutually dependent, etc.
  - **Non-spherical errors**: The errors are heteroskedastic or autocorrelated
- As a result, Econometric models often are inefficient (high bias and/or variance)
- One motivation for using ML in finance is that it relies on more realistic assumptions

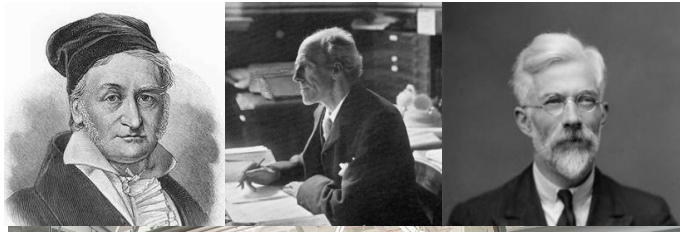


Because financial systems are extremely complex, Econometric models are often misspecified and miss variables (particularly interaction effects). In the above experiment, the OLS specification is correctly linear, however it is missing one interaction, leading to high bias and variance.

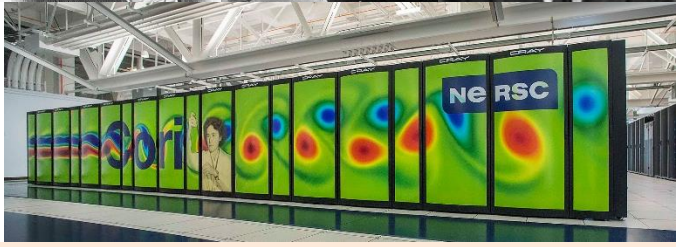
ML Solutions to the Bias-Variance Dilemma

- In finance, there is no strong theoretical reason to discard biased models
  - Financial systems are too complex for expecting fully well-specified (unbiased) models
- Accordingly, minimizing variance subject to zero bias (MVUE) is the wrong objective
- ML methods can achieve lower MSE than Econometric models through
  - Cross-validation
  - Complexity reduction
    - Regularization
    - Feature selection, dimensionality reduction
  - Ensemble methods
    - Bootstrap aggregation (Bagging)
    - Boosting
    - Stacking

The History of Empirical Research



The Future of Quantitative Finance



What underlies these three solutions is computational statistics. When Gauss, Pearson, Fisher, and others created the tools used by econometricians, they did not have access to computers. They had no choice but to rely on closed-form solutions based on strong (unrealistic) assumptions. Today, we have more practical tools to solve the bias-variance dilemma.

Solution 1:
Cross-Validation

A Different MSE Decomposition

- Recall from the Problem statement that we wish to find the statistical model ($\hat{f}$) that minimizes the mean squared *true* error, $\mathrm{E}\left[\left(f[x_n] - \hat{f}[x_n]\right)^2\right]$
- The problem is, f is unknown, and so is the true error
- Instead, we minimize the mean squared *empirical* error,

$$\mathrm{E}\left[\left(y_n - \hat{f}[x_n]\right)^2\right] = \underbrace{\mathrm{E}\left[\left(f[x_n] - \hat{f}[x_n]\right)^2\right]}_{\text{true error}} + \underbrace{\sigma_\varepsilon^2}_{\text{noise}} + \underbrace{2\mathrm{E}\left[\left(f[x_n] - \hat{f}[x_n]\right)\varepsilon_n\right]}_{\text{cross-term}}$$

- A couple of notes:
  - The noise (σ_ε^2) does not displace the minimum, because it is a constant
  - However, the cross-term ($2\mathrm{E}\left[\left(f[x_n] - \hat{f}[x_n]\right)\varepsilon_n\right]$) is not necessarily a constant, and may lead to a function $\hat{f}$ that does not minimize the true error
- We need to study the cross-term

Cross-Term Under Test Set Errors

- Consider a datapoint (x_0, y_0) that is not part of the **train set** used to fit $\hat{f}$, $(x_0, y_0) \notin \{(x_n, y_n)\}_{n=1,\dots,N}$
- In this case, $\mathrm{E}\left[\left(f[x_0] - \hat{f}[x_0]\right)\varepsilon_0\right] = 0$, because
  - f is not fitted, so it does not depend on any observations (train set or test set)
  - $\hat{f}$ depends on $\{(x_n, y_n)\}_{n=1,\dots,N}$, but it does not depend on (x_0, y_0), because (x_0, y_0) is not part of the train set
  - The value of ε_0 does depend on y_0, because $\varepsilon_0 = y_0 - f[x_0]$
- Thus,

$$\mathrm{E}\left[\left(y_0 - \hat{f}[x_0]\right)^2\right] = \mathrm{E}\left[\left(f[x_0] - \hat{f}[x_0]\right)^2\right] + \sigma_\varepsilon^2$$

- In conclusion, minimizing MSE on observations *outside the train set* (i.e., a test set) is equivalent to minimizing the mean squared true error
  - This important result is the foundation for **cross-validation**

Cross-Validation (CV)

- CV is a validation technique that assesses how the forecasts of a model generalize on unseen data (test set). The observations are rows of (X,y). A score is computed on the forecasts across multiple test sets

Type	Name	Description
Exhaustive	**Leave-one-out**	Iterate one observation as the test set, and the remaining observations as the train set. Out of N observations, there are N train-test splits.
	Leave-p-out	Iterate p observations as the test set, and the remaining observations as the train set. Out of N observations, there are $\binom{N}{p}$ train-test splits.
Non-exhaustive	**Holdout**	Randomly assign observations to the train and test sets. Caveat: There is a single run, and the result may not be representative.
	K-Fold	Split the N observations into K regular folds, and apply a leave-one-out CV on the folds. The data may or may not be shuffled prior to forming the folds.
	Combinatorial K-Fold	Split the N observations into K regular folds, and apply a leave-p-out CV on the folds. The data may or may not be shuffled prior to forming the folds.
	Monte Carlo	Train sets are generated from random subsampling (random sampling without replacement) of the rows of (X,y)

Solution 2:
Complexity Reduction

Cross-Term Under Train Set Errors

- Let us compute $\mathrm{E}\left[\left(f[x_0] - \hat{f}[x_0]\right)\varepsilon_0\right]$ when (x_0, y_0) is part of the **train set** used to fit $\hat{f}$, $(x_0, y_0) \in \{(x_n, y_n)\}_{n=1,\ldots,N}$
- Applying Stein's lemma we obtain that

$$\mathrm{E}\left[\left(f[x_0] - \hat{f}[x_0]\right)\varepsilon_0\right] = -\sigma_\varepsilon^2 \mathrm{E}\left[\frac{\partial \hat{f}[x_0]}{\partial y_0}\right]$$

- We can think of $D_0 = \mathrm{E}\left[\frac{\partial \hat{f}[x_0]}{\partial y_0}\right]$ as a measure of model complexity relative to the train set
  - As N grows, $\hat{f}$ is less sensitive to changes in y_0
  - The more complex $\hat{f}$ is, the more sensitive it becomes to changes in y_0
- Thus,

$$\mathrm{E}\left[\left(y_0 - \hat{f}[x_0]\right)^2\right] = \mathrm{E}\left[\left(f[x_0] - \hat{f}[x_0]\right)^2\right] + \sigma_\varepsilon^2(1 - 2D_0)$$

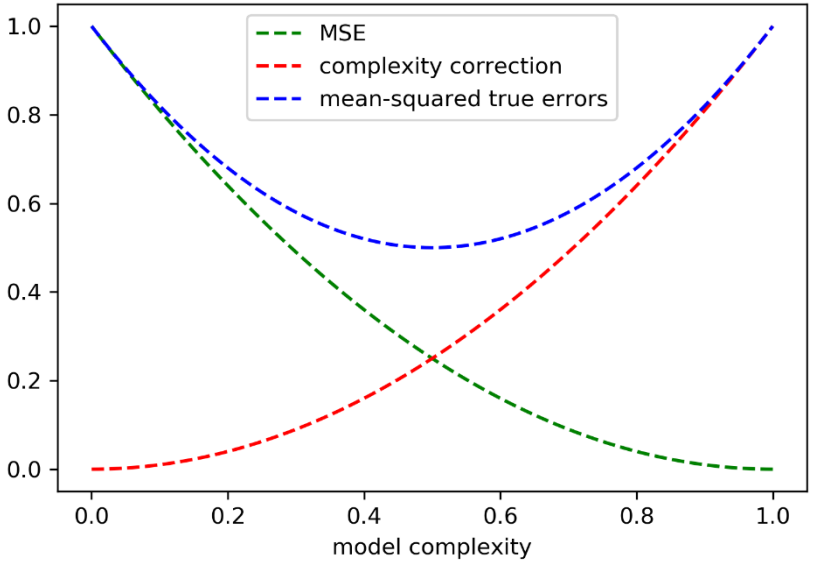
Stein's Unbiased Risk Estimator (SURE)

- In the context of train set errors,

$$\frac{1}{N}\sum_{n=1}^{N}\left(f[x_n]-\hat{f}[x_n]\right)^2 =$$

$$\underbrace{\frac{1}{N}\sum_{n=1}^{N}\left(y_n-\hat{f}[x_n]\right)^2}_{\text{MSE}} + \underbrace{\sigma_\varepsilon^2\left(\frac{2}{N}\sum_{n=1}^{N}D_n-1\right)}_{\text{complexity correction}}$$

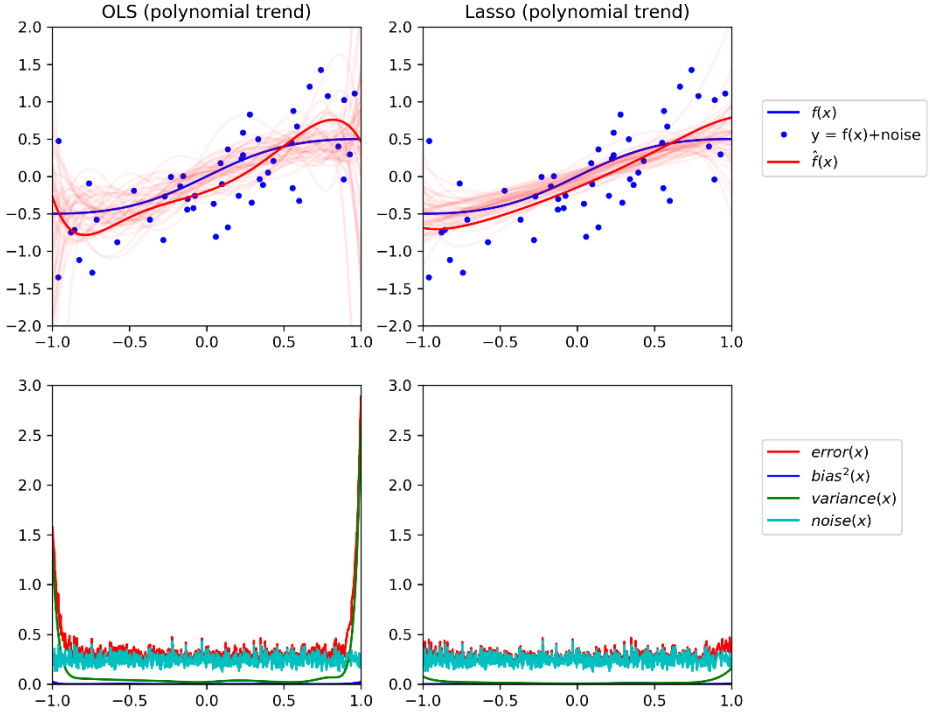
- When D_n is unknown, we can improve our estimate of $\mathrm{E}\left[\left(f[x_n]-\hat{f}[x_n]\right)^2\right]$ by adding a penalty for complexity to MSE's estimate
  - This important result is the foundation for **regularization** *and* **feature selection**



In the train set, MSE monotonically decreases with complexity, until $MSE = 0$ for a perfectly overfit model. There is an optimal level of complexity that minimizes the mean-squared true errors.

Regularization

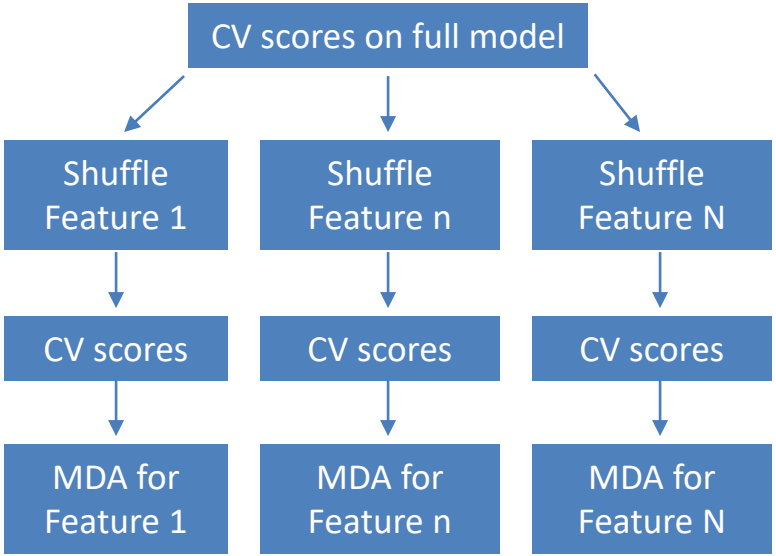
- Common forms of ML regularization include
  - **Early stopping**: The training step is stopped before convergence, in order to prevent an overly complex (overfit) model
    - E.g., place a limit to the growth of trees in a random forest
  - **Tikhonov regularization**: Also known as Ridge regression, it adds as a penalty to the objective function the L_2-norm of the vector of estimated parameters
    - E.g., penalize solutions with larger norms
  - **Sparsity regularizers**: Similarly to Tikhonov regularization, it adds as a penalty to the objective function the L_1-norm of the vector of estimated parameters
    - E.g., penalize solutions with many non-zero parameters



In the above example, LASSO (a sparsity regularizer) zeroes the estimated coefficients for unnecessary regressors, thus achieving a lower MSE than OLS.

Feature Selection

- Another way to reduce the complexity of a model is by eliminating unnecessary variables
- Feature selection is a difficult task for classical statistical models
  - An important variable will be discarded under the wrong specification
  - An unimportant variable will be selected under the wrong specification
- **ML decouples the specification search from the variable search**, hence enabling the reduction of complexity via feature selection

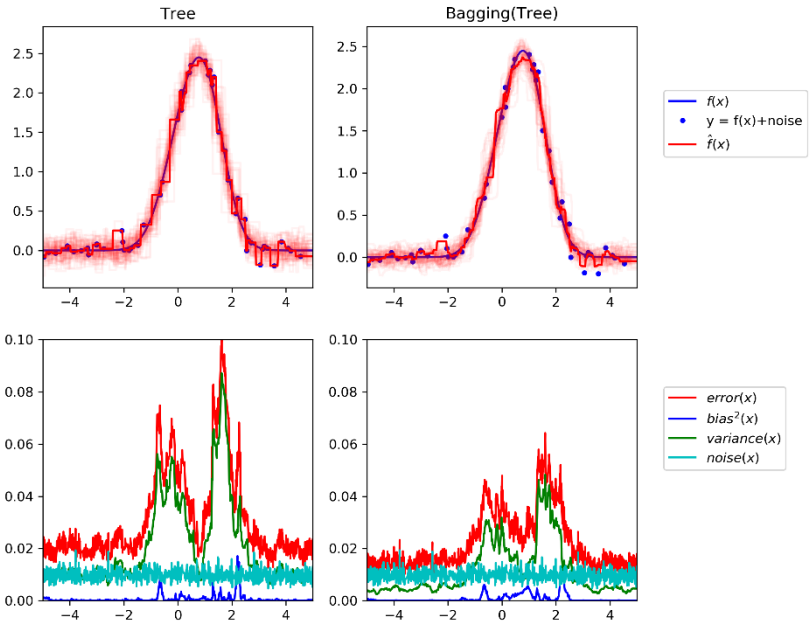


Permutation tests (e.g., MDA) allow us to determine the predictive power associated with each feature. This helps us identify which variables are important, regardless of the true model's specification.

Solution 3:
Ensemble Estimators

Overcoming the Bias-Variance Tradeoff

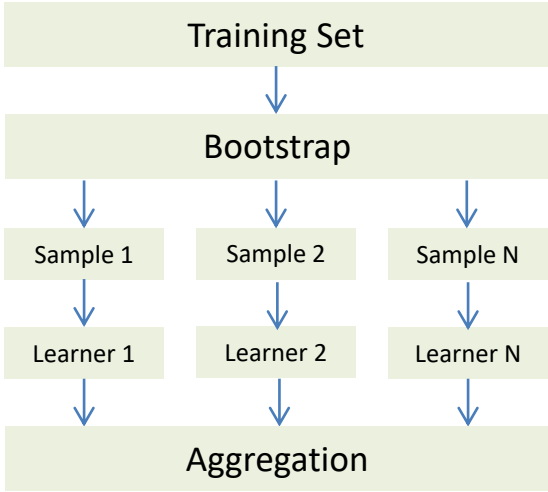
- In general, models with lower variance exhibit higher bias, and models with lower bias exhibit higher variance
- One solution to break this bias-variance tradeoff is to
  1. Overfit a large number of uncorrelated predictors (each of them will have low bias)
  2. Generate an ensemble forecast, based on forecasts from (1)
- Ensemble forecasts exhibit lower variance, and under some circumstances also lower bias, than the individual forecasts



On a non-linear dataset, we fit a decision tree and an ensemble of decision trees. The ensemble achieves lower variance and somewhat lower bias, particularly around the turning points of f.

Bagging

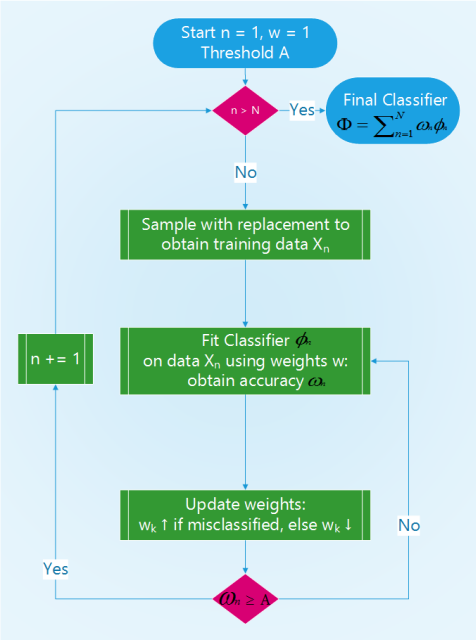
- A bagging algorithm forms a strong learner by
  - training several weak learners independently (in parallel), and
  - aggregating their forecasts using a deterministic procedure
- For a sufficiently large number of uncorrelated learners, bagging algorithms effectively reduce the **variance**
  - For the details, see section 6.3.1 of AFML
- In addition, if the individual learners have a minimum accuracy, bagging algorithms may also reduce the **bias**
  - For the details, see section 6.3.2 of AFML
- In classification problems, two popular forms of aggregation are
  - Hard voting: Majority voting across all the forecasts
  - Soft voting: Average the probabilities of the individual forecasts, and select the class with highest average probability



Bagging almost always achieves variance reduction, and in some cases, also bias reduction. Because of its parallelization, bagging is a good default method to prevent overfitting.

Boosting

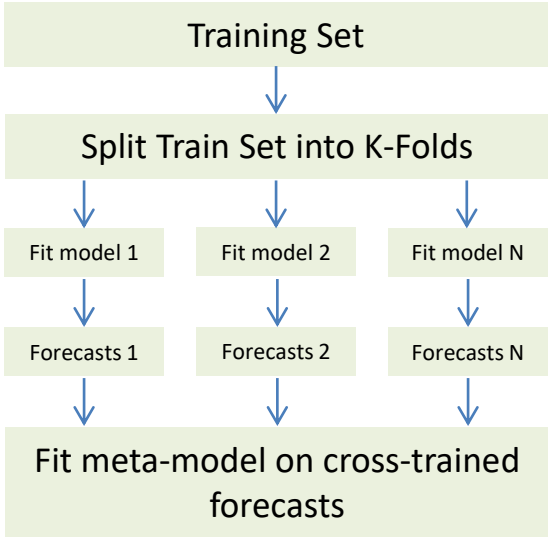
- A boosting algorithm forms a strong learner by
  - training several weak learners iteratively, where subsequent models learn from prior errors, and
  - aggregating their forecasts using a deterministic procedure
- Because of its adaptive nature, boosting is generally more effective than bagging at reducing **bias**
- Boosting can also be effective at reducing **variance**
- Unlike bagging, boosted learners cannot be fit in parallel
  - Consequently, boosting often takes longer to fit
- Two main types of boosting algorithms
  - Adaboost: It adapts by updating the sample weights
  - Gradient boosting: It adapts by updating the observations (residuals)



Adaboost's decision flow.

Stacking

- A stacking algorithm forms a strong learner by
  - training several weak learners in parallel, and
  - training a meta-model on the individual predictions
- Differences with bagging and boosting:
  - Stacking often considers heterogeneous weak learners
  - Stacking uses a meta-model to combine the weak learners (rather than a deterministic method)
- Stacking often involves K-fold cross-training
  1. Split the train set into K-folds
  2. Train N models on K-1 folds, and make predictions on held-out
  3. Train meta-model, with the held-out predictions as features
- Training can be parallelized, however it is computationally expensive



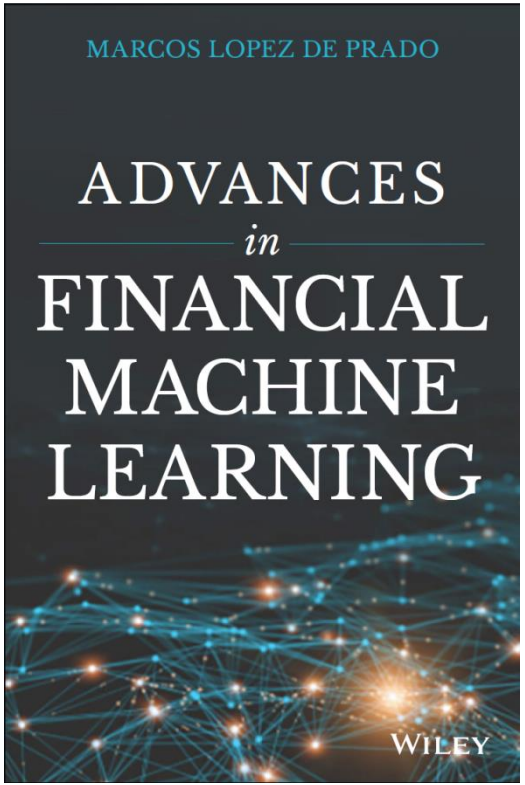
Stacking allows the training of heterogeneous weak learners, which are (non-deterministically) combined by a meta-model.

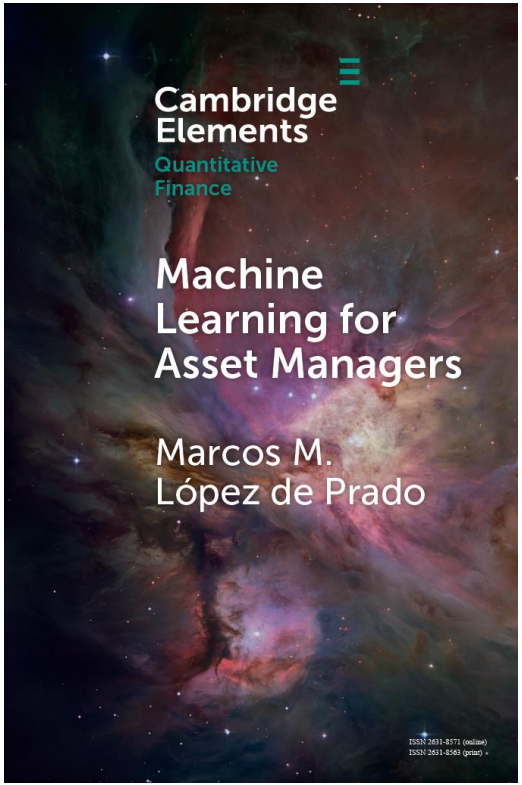
Conclusions

ML has an Important Role to Play in Finance

- Classical statistics (e.g., Econometrics) relies on assumptions that are often unrealistic in finance
  - In particular, the assumptions that: (i) we have perfect knowledge about the model's specification, and (ii) that we know all the variables involved in a phenomenon (including all interaction effects)
- Classical statistical models applied to financial series often result in poor performance (high bias and high variance)
  - Low *p*-values are meaningless when researchers wrongly assume a linear relationship between variables, or omit interaction effects
    - E.g., there is no theoretical reason to expect a linear relationship between financial returns and risk factors, hence the low *p*-values found in the factor literature can be misleading
- **In the context of financial datasets, ML is a more appropriate quantitative tool**
  - It relies on fewer assumptions, and does not impose a specification, thanks to its numerical approach
  - It maximizes performance, because it does not restrict itself to unbiased estimators
  - It decouples the specification search from the variable search, which is conducive to better theories

For Additional Details





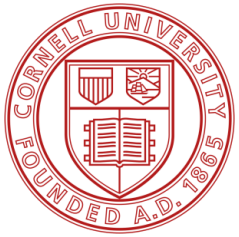
The first wave of quantitative innovation in finance was led by Markowitz optimization. Machine Learning is the second wave and it will touch every aspect of finance. López de Prado's Advances in Financial Machine Learning is essential for readers who want to be ahead of the technology rather than being replaced by it.
— Prof. **Campbell Harvey**, Duke University. Former President of the American Finance Association.

Financial problems require very distinct machine learning solutions. Dr. López de Prado's book is the first one to characterize what makes standard machine learning tools fail when applied to the field of finance, and the first one to provide practical solutions to unique challenges faced by asset managers. Everyone who wants to understand the future of finance should read this book.
— Prof. **Frank Fabozzi**, EDHEC Business School. Editor of The Journal of Portfolio Management.

Disclaimer

- The views expressed in this document are the author's, and do not necessarily reflect those of the organizations he is affiliated with.
- No investment decision or particular course of action is recommended by this presentation.
-

www.QuantResearch.org



Clustered Feature Importance

Prof. Marcos López de Prado
Advances in Financial Machine Learning
ORIE 5256

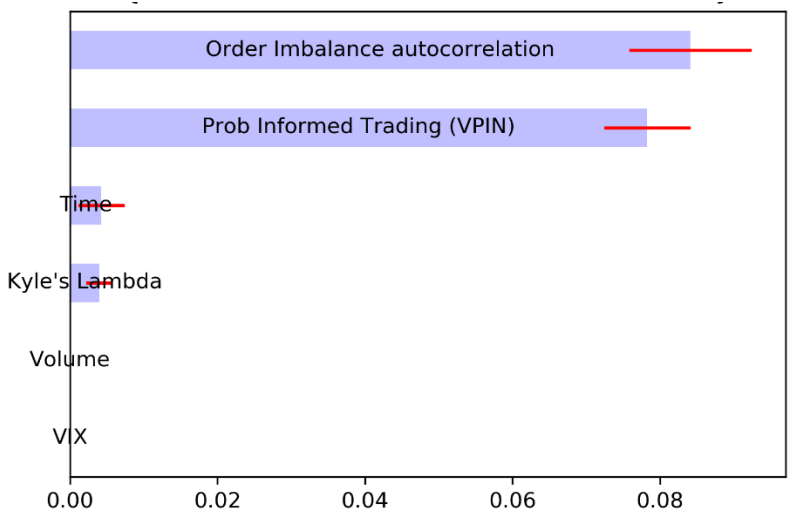
Key Points

- A primary goal of empirical research is to identify the variables involved in a phenomenon
  - The identification of these variables is a prerequisite to the formulation of a theory
- In classical statistics (e.g., Econometrics), the significance of variables is established through *p*-values
- *p*-values suffer from multiple flaws, which have led to the acknowledgement that most discoveries in finance are false
- In this seminar, we explain how Machine Learning (ML) methods overcome the flaws of *p*-values, and facilitate the discovery of new theories
- **This application demonstrates that ML is *not* necessarily a "black box", contrary to popular perception**

What is Feature Analysis?

The Role of Feature Analysis

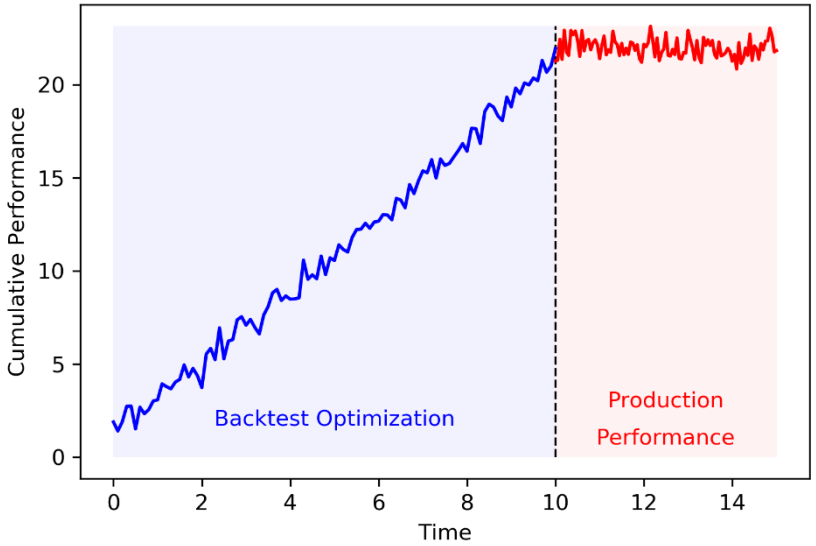
- With the help of ML, feature analysts find which variables are able to predict a particular outcome
  - ML algorithms decouple the specification search from the variable search
- The patterns identified by feature analysts do not imply causation
  - Strategists/Economists must hypothesize the cause-effect mechanism that underlies the pattern uncovered by feature analysts
  - That hypothesis can then be tested, in order to discriminate **causation** from mere **codependence**



In this feature analysis, researchers found that volatility bursts can be predicted with the help of two microstructural variables. Other variables were shown to have little or no predictive power. This finding could lead to the formulation of a hypothesis, which can then be backtested.

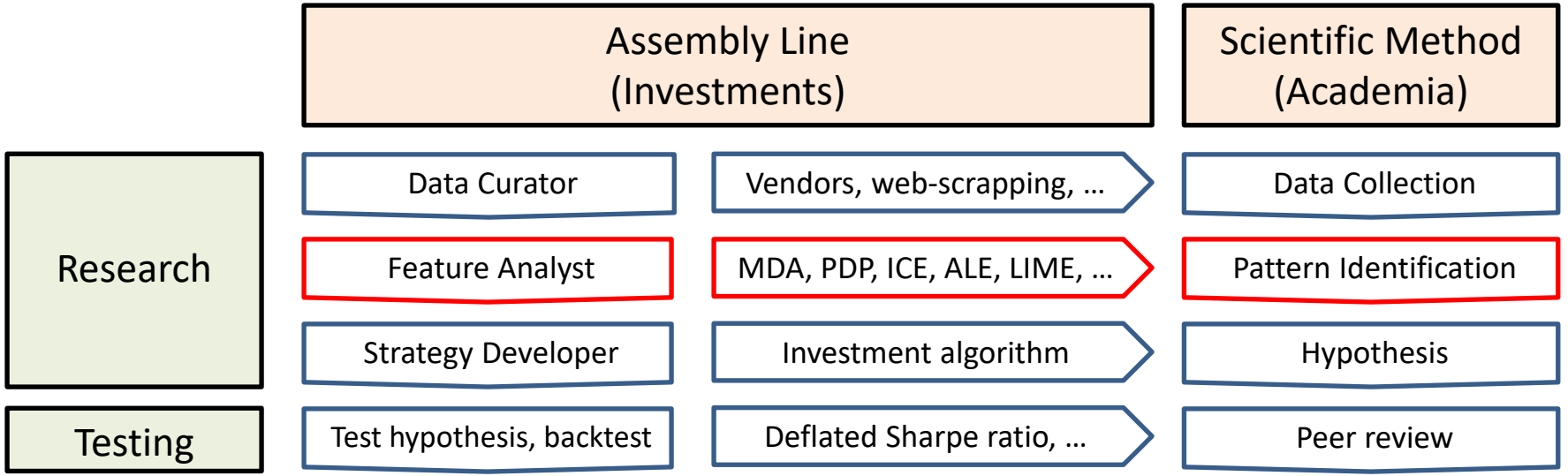
Feature Analysis vs. Backtesting

- **Backtesting is not a research tool**. It is a validation tool
  - By the time we backtest, research should have concluded
  - Building a hypothesis through backtesting leads to selection bias and false discoveries
- The goal of backtesting is not to form a hypothesis
  - On the contrary, one goal of backtesting is to **deconstruct a hypothesis, and to prove the researcher wrong** through counter-examples
- To form a hypothesis, first we must find the variables involved in a phenomenon
  - This is the key role of Feature Analysis



Backtest overfitting results from confounding Research with Validation. The failure of most quantitative funds can be traced back to this basic misunderstanding of the scientific method.

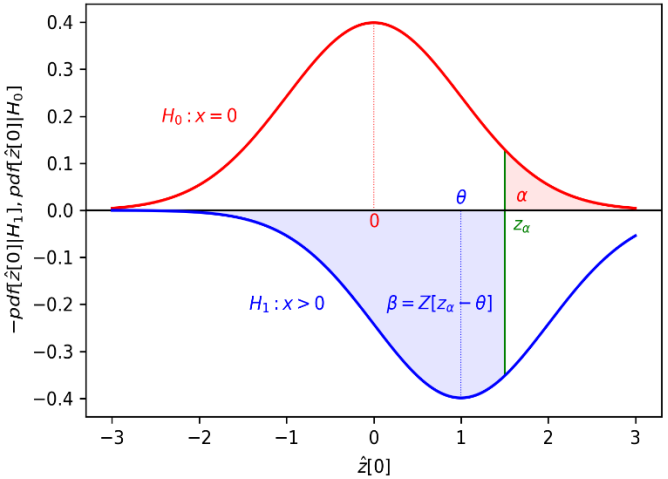
How Feature Analysis Fits in the Scientific Method



Classical Feature Importance: *p*-Values

Hypothesis Testing

- The null hypothesis (H_0) represents the absence of a phenomenon (a negative)
- The alternative hypothesis (H_1) represents the existence of a phenomenon (a positive)
- A test rejects a null hypothesis H_0 with confidence $(1-\alpha)$ when the test's statistic exceeds a value τ that, should H_0 be true, could only occur with probability α
- Two possible errors:
  - Type I: Reject a true H_0 (false positive probability, α)
  - Type II: Reject a true H_1 (false negative probability, β)
- ***p*-value is the α associated with an observed τ**



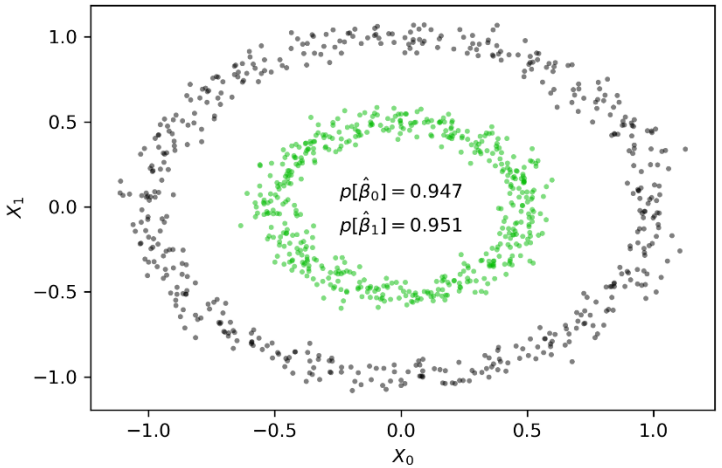
Example of a right-sided test:

$P\left[\hat{x} \geq \tau \middle| H_0 : x = 0\right] \leq \alpha$ (red area)

$P\left[\hat{x} \leq \tau \middle| H_1 : x = \theta\right] \leq \beta$ (blue area)

Pitfall #1: Specification-Significance Entanglement

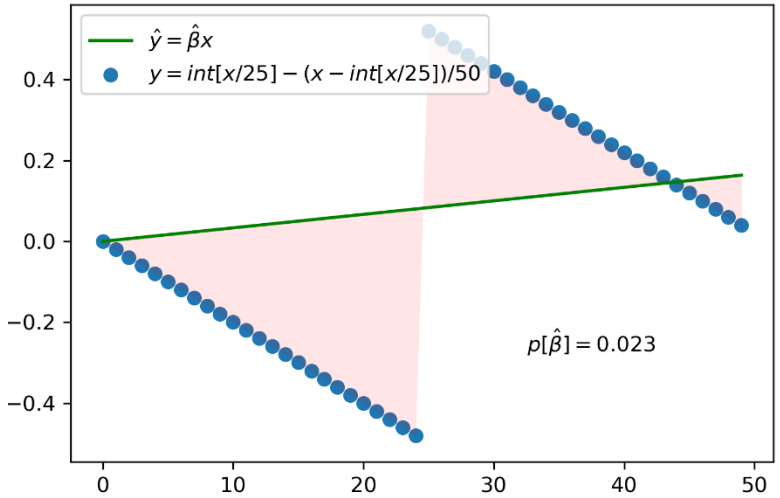
- *p*-values are typically computed on the estimated coefficients of a given model
- When the *p*-value is less than 5%, the variable associated with that coefficient is deemed *statistically significant*, under the assumption that the model is correctly specified
- Thus, *p*-values cannot tell us whether a variable is significant *per se*. They cannot decouple the specification search from the significance search
- In finance, where systems are so complex that researchers can only guess the correct specification, *p*-values are likely to lead to false conclusions



Features X_0 and X_1 can be jointly used to perfectly separate these two classes. However, a logistic regression deems both features uninformative (a false negative).

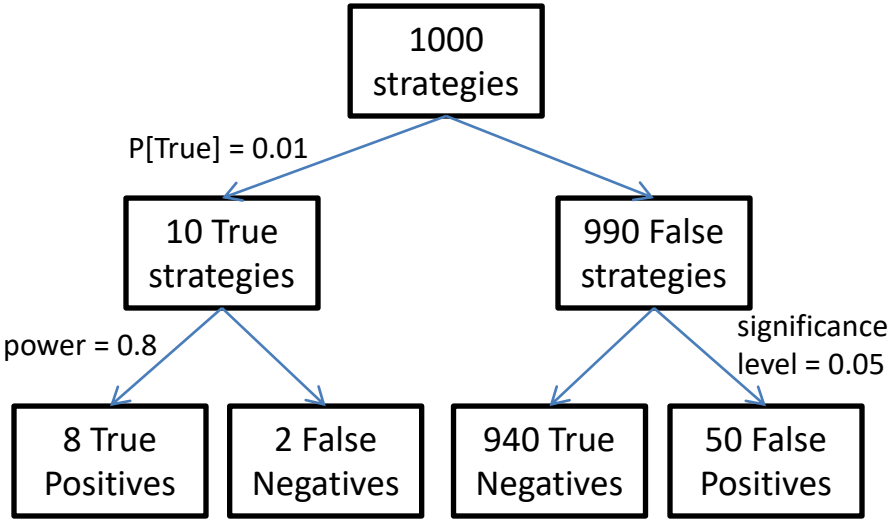
Pitfall #2: In-Sample Estimate

- OLS regressions attempt to adjudicate variance in-sample, not forecast
  - **Informational leakage: Each observation is used for fitting the model and evaluating the model**
- *p*-values are derived from *estimation* errors (in-sample), not *generalization* errors (out-of-sample)
  - A variable that appears to be significant ($p < .05$) may indeed have little predictive power
- In this example, an OLS regression (green line) attempts to minimize the in-sample error (red area)
  - The *p*-value is low, even though the model gets the sign of the relationships wrong



The relationship appears to be linearly ascending (in-sample). Cross-validation would have shown that the model performs poorly on unseen data

Pitfall #3: A Dubious Probability



Suppose that the probability of a backtested strategy being profitable is 1%.

Then, at the standard thresholds of 5% significance and 80% power, researchers are expected to make 58 discoveries out of 1000 trials, where 8 are true positives and 50 are false positives.

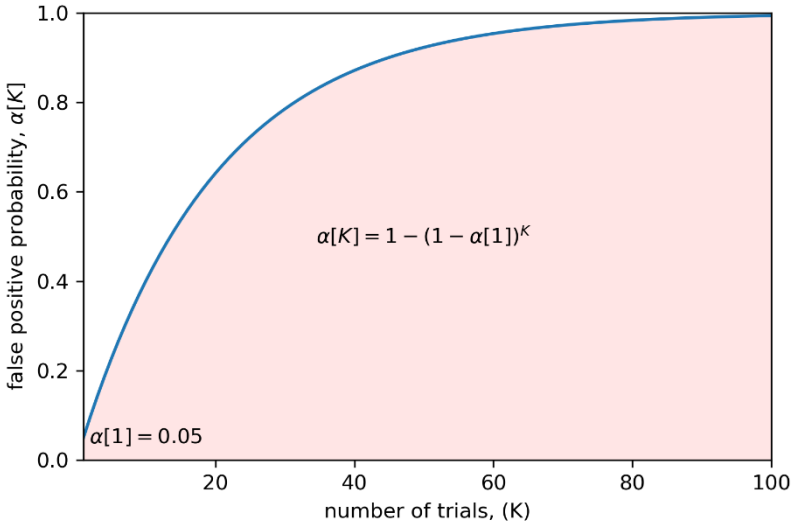
Under these circumstances, **a *p*-value of 5% implies that at least 86% of the discoveries are false!**

The problem is, *p*-values evaluate a somewhat irrelevant probability: $\mathrm{P}[X > x|H_0]$.

What we really care about is a different probability: $\mathbf{P}[\boldsymbol{H_1}|\boldsymbol{X > x}]$.

Pitfall #4: *p*-Hacking

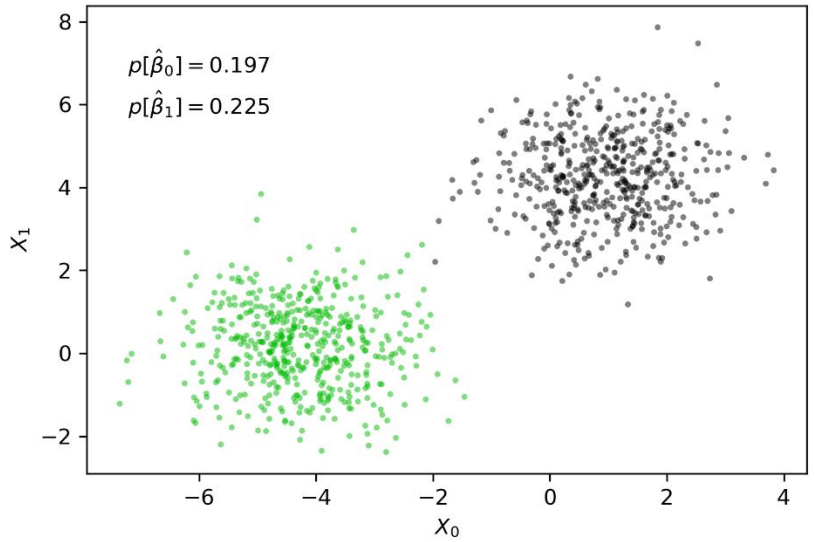
- If hypotheses are tested more than once on a given dataset, the probability of a false positive exceeds α
- Selecting the model with lowest *p*-values out of multiple trials leads to **selection bias**
- In finance, where datasets are limited, it is highly likely that a researcher reuses a datasets multiple times
- Virtually no paper published in financial academic journals controls for the number of trials involved in a discovery (K)
- **It is highly likely that published *p*-values are artificially low (false positives)**



The false positive probability quickly rises after the first trial. Journal articles present findings as if they had been the result of a single trial. Because that is almost never the case, most discoveries in finance are false.

Pitfall #5: Substitution Effects

- In addition to correct model specification, *p*-values require (among other assumptions):
  - Uncorrelated regressors (no multicollinearity)
  - White noise residuals
  - Normally-distributed residuals
  - No outliers
- In particular, *p*-values are not robust to multicollinearity (linear substitution effects)
- Because of this lack of robustness, important variables can be assigned high *p*-values, and be wrongly dismissed as irrelevant



These two blobs can be easily separated with either X_0 or X_1. When we pass both features to a logistic regression, both are deemed insignificant, as a result of substitution effects.

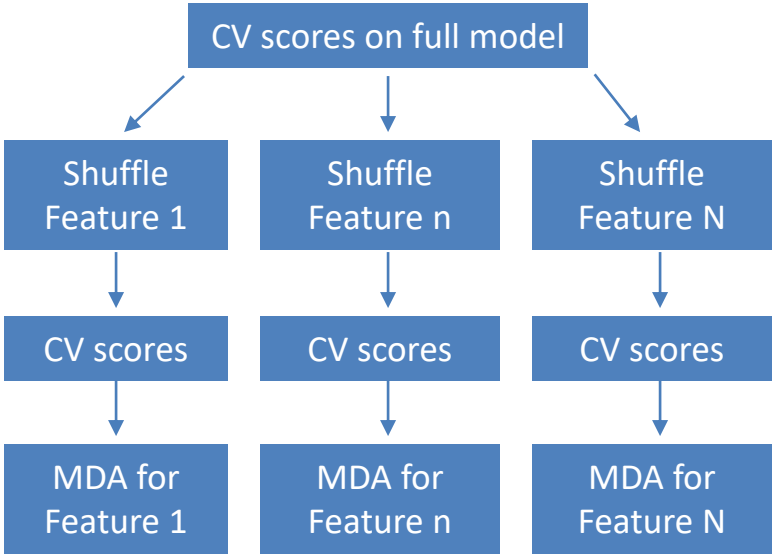
Machine Learning Feature Importance: Mean Decrease Accuracy (MDA)

Mean Decrease Accuracy (MDA)

- MDA is one particular feature analysis:

1. Fit a model using all features, and cross-validate the prediction's scores
2. For each feature
   a) Shuffle the feature, refit the model and cross-validate the new prediction's scores
   b) Compute the distribution of the difference between scores (2.a) and (1)

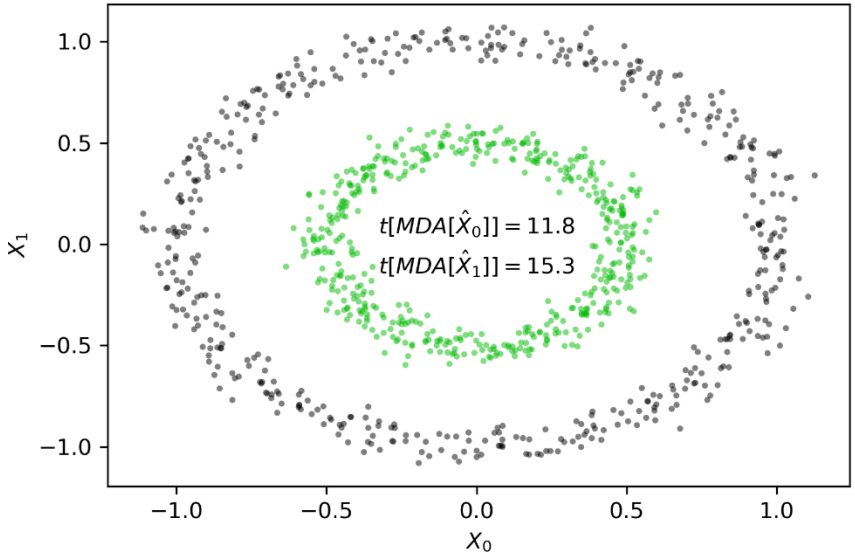
- Shuffling an important feature causes a large loss in model performance
- MDA is also known as *permutation importance*, because it can be derived on any score (not only accuracy)



We can compute K losses in performance in a K-fold cross-validation. That allows us to bootstrap the distribution of the generalization error.

Specification-Significance Disentanglement

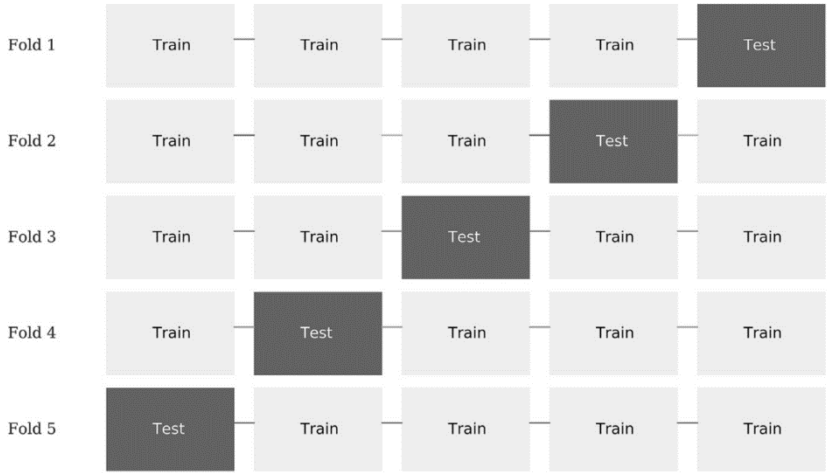
- MDA relies on a particular algorithm to determine the loss in performance (e.g., a random forest), however these algorithms can fit a very wide range of specifications
  - Interestingly, a black-box ML algorithm can be a tool for insight and model interpretability!
- Unlike *p*-values, MDA determines the importance of a feature irrespective of the specification
- Once we know the variables involved in a phenomenon, we can search for the correct specification, and formulate a hypothesis



An MDA analysis of the two concentric classes correctly concludes that both features are extremely informative (above, t-values computed as the ratio bean mean[MDA] and std[MDA]).

Out-of-Sample Measurement

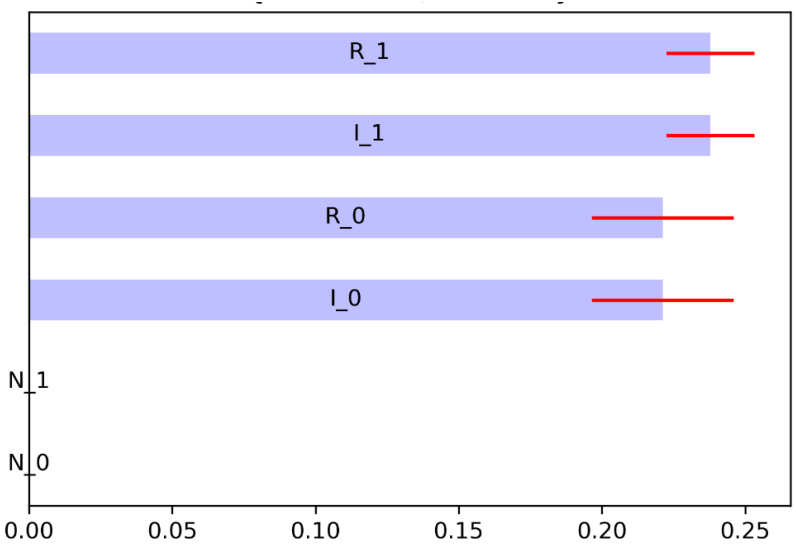
- Unlike *p*-values, MDA does not evaluate the importance of a feature in-sample
  - Observations used to fit the algorithm are not used by MDA to evaluate its performance
- Instead, MDA relies on cross-validation to determine a feature's importance
  - The importance is derived from the increase in the *generalization* error (not the *estimation* error) that results from shuffling a feature
- In the event that X exhibits serial dependence and y is formed on overlapping periods, the train set must be **purged** and **embargoed** (see CPCV)



Data splits in a K-Fold cross-validation experiment. Regardless of the particular cross-validation approach used, MDA always derives out-of-sample importance.

Interpretability

- MDA estimates the drop in performance (out-of-sample) that would result from sampling a variable randomly
- Unlike *p*-values, MDA has a clear interpretation, which
  - does not rely on strong assumptions
  - can be extended to any scoring function
  - is model agnostic
- This is an intuitive, general and flexible approach, that enables the comparison among very different models



MDA's flexibility is a consequence of its experimental nature. Rather than relying on strong (and potentially unrealistic) inferential assumptions, the generalization error is estimated via computational methods.

Avoidance of Selection Bias

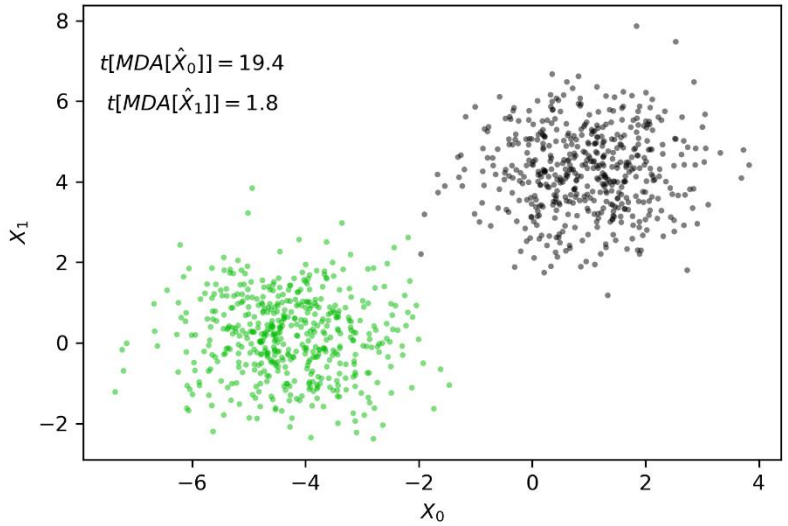
- When implemented correctly, MDA can prevent selection bias
- The key is to avoid reusing a single test set multiple times
- For instance
  - Repeated CV: K-Fold can be applied on shuffled rows of (X, y), after purging, thus preventing that a model is selected out of a particular split scheme
  - Monte Carlo CV: Cross-validation can be conducted on subsampled rows of (X, y) (random sampling without replacement)
  - Leave-p-out CV: This is an exhaustive cross-validation, where all possible combinations of p-sized test sets are evaluated

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	Paths
G1	x	x	x	x	x											5
G2	x					x	x	x	x							5
G3		x				x				x	x	x				5
G4			x				x			x			x	x		5
G5				x				x			x		x		x	5
G6					x				x			x		x	x	5

Combinatorially-purged cross-validation (CPCV) is a variation of *leave-p-out*, where the train set in purged and embargoed.
For example, by holding 2 out of 6 folds, we can form 15 different train-test splits, resulting in 5 full paths (where the 6-Fold path is only one of them). The number of paths can be raised, by increasing the value of *K* or the value of *p*. The end effect is an arbitrarily small chance of overfitting.

Substitution Effects

- When two important features share information, shuffling one may not result in a material reduction in model performance
  - MDA may wrongly dismiss one or both as uninformed
- Typically MDA is more robust to substitution effects than *p*-values (see right plot), however this is a potential vulnerability
  - One possibility is to shuffle together all features with mutual information
- The rest of the presentation explains how **Clustered MDA addresses substitution effects**



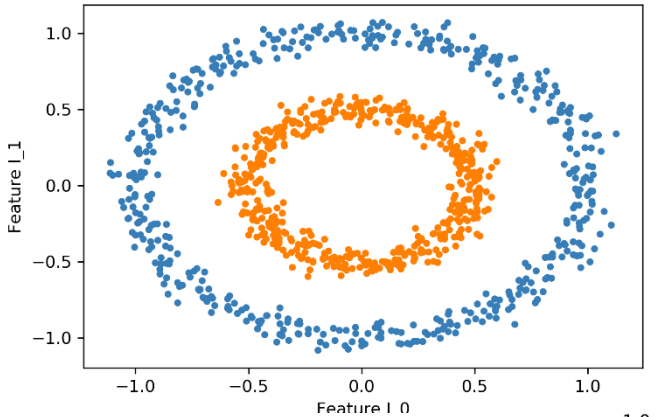
These two blobs can be easily separated with either X_0 or X_1. MDA shows a high t-value for both, however we can appreciate a substitution effect whereby the importance of X_1 is undermined by X_1.

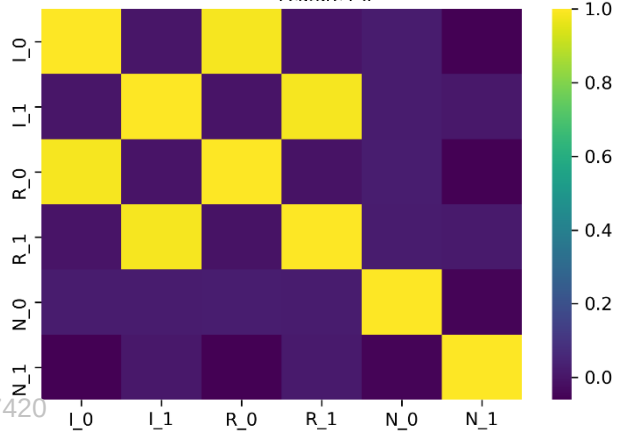
Clustered Feature Importance under Linear Substitution Effects

Linear Substitution Effects

- Consider 1,000 binary labels that can be clearly separated as a function of two informed features, I_0 and I_1
  - This is not an artificial example. Credit markets display this type of concentric patterns
- We add two redundant features:
  - R_0 is a linear function of I_0
  - R_1 is a linear function of I_1
- We add two noise feature, N_0 and N_1

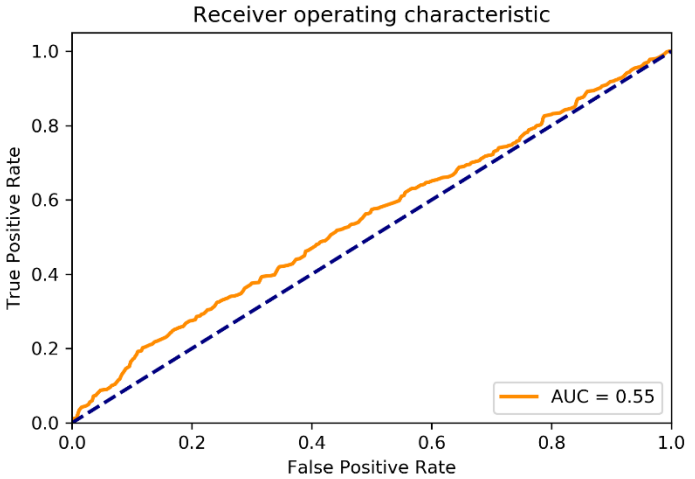
- The correlation matrix evidences that the system is multicollinear
  - Notice the off-diagonal blocks (I_0,R_1) and (I_1,R_1)



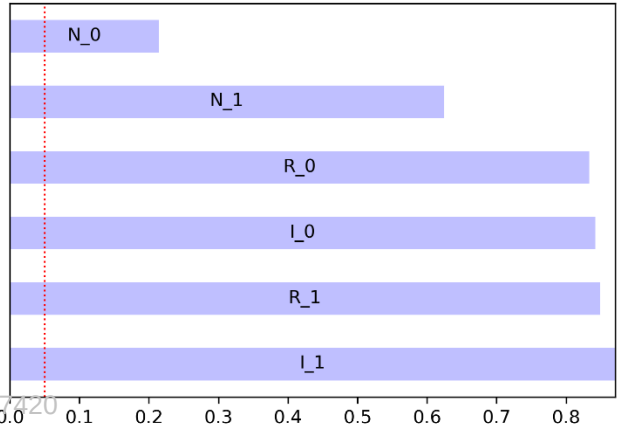


p-Values

- All *p*-values exceed the 5% threshold
- *p*-values mislead us into concluding that *all* features are noise (a false negative)
- Moreover, N_0 and N_1 exhibit lower *p*-values than the informed and redundant features



- Why do *p*-values mislead us?:
  - The model is misspecified
    - AUC > 0.5, but it should have been 1, because the labels are perfectly separable
  - The system is multicollinear
  - *p*-values do not disentangle the specification search from the significance search

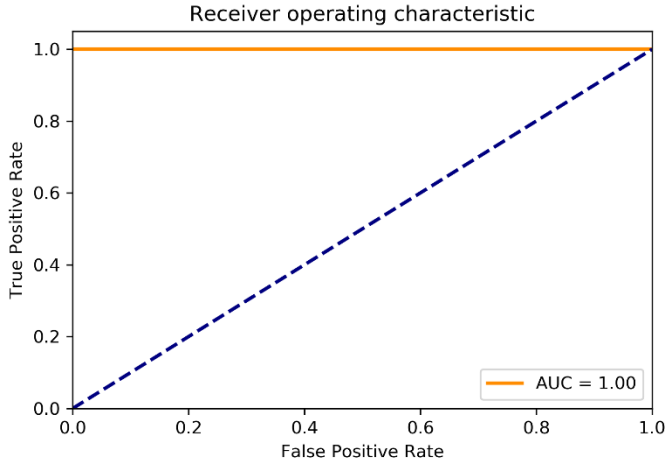


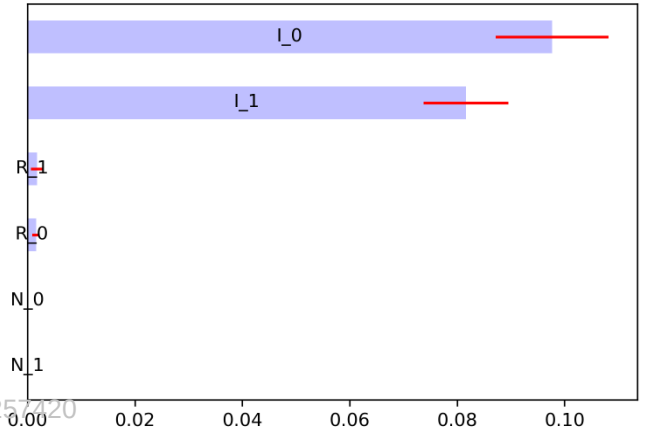
MDA on Random Forest

- MDA correctly identifies I_0 and I_1 as informed
- However, MDA incorrectly dismisses R_0 and R_1 as noise

- MDA (partially) failed because of multicollinearity
  - (I_0, I_1) prevent a reduction of AUC when (R_0, R_1) are shuffled

- Note that, unlike with *p*-values, misspecification is not an issue: AUC=1



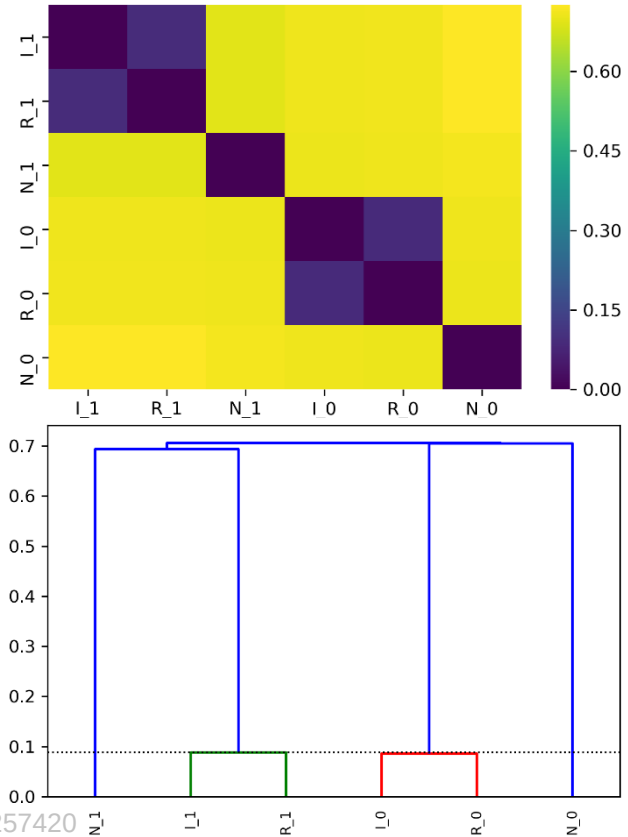


Features Clustering

- Apply a single-linkage agglomerative clustering algorithm on a distance matrix

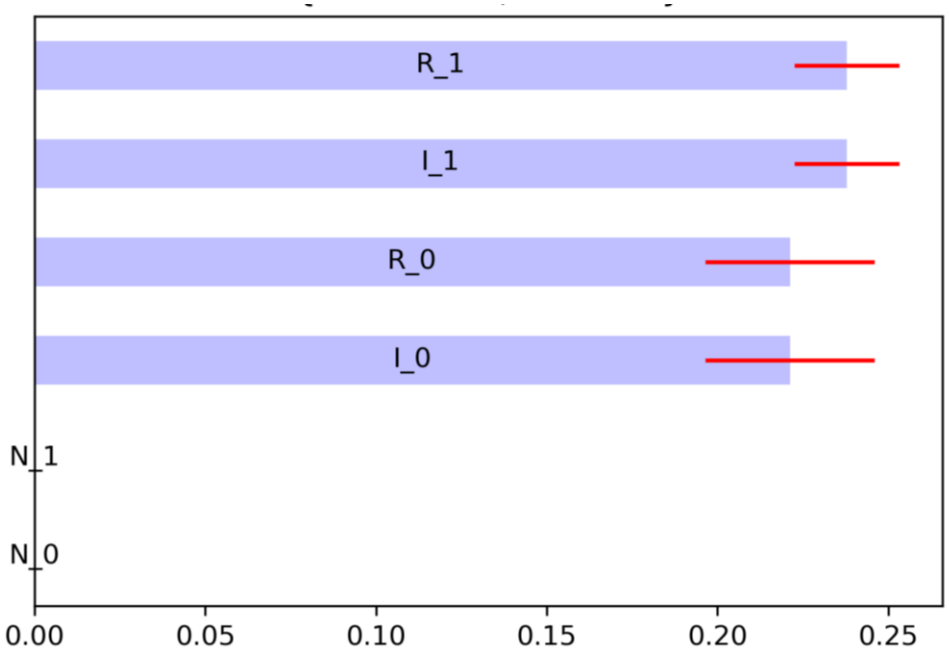
$$d_{i,j} = \sqrt{\frac{1}{2}(1 - \rho_{i,j})}$$

- We base the distance on correlation, because the substitution effect is linear
- The algorithm recognizes that
  - The optimal number of clusters is 4
  - R_0 is redundant to I_0
  - R_1 is redundant to I_1
- **The system formed by the clusters is not multicollinear (no off-diagonal blocks)**



Clustered MDA on Random Forest

- Instead of shuffling each variable individually, we shuffle together all variables within a cluster
- Clustered MDA gives the right answer:
  - All informed and redundant features are important
  - N_0 and N_1 have zero contribution to the model's performance
- Why did Clustered MDA work?
  - MDA decouples the specification search from the significance search
  - The clusters are not multicollinear

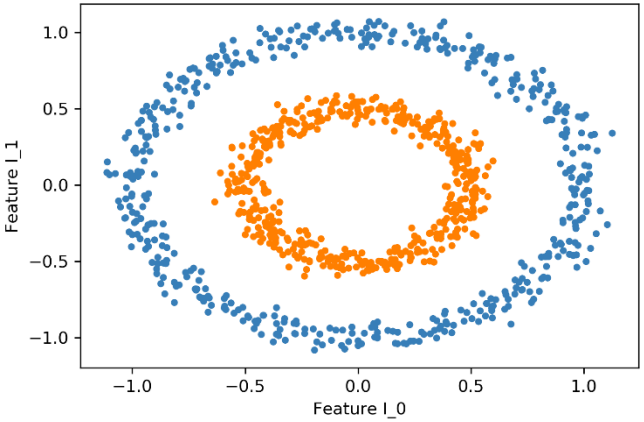


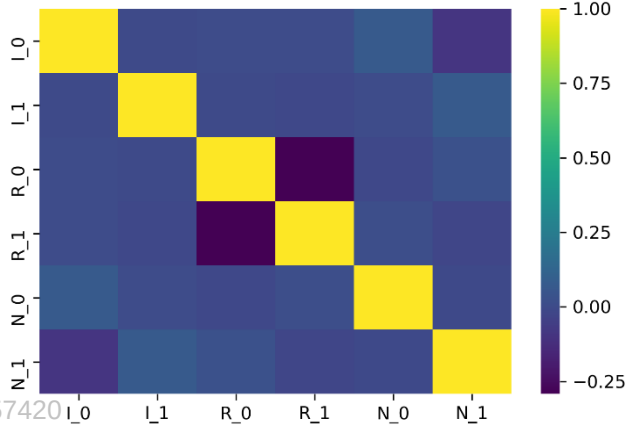
Clustered Feature Importance under Non-Linear Substitution Effects

Non-Linear Substitution Effects

- Consider 1,000 binary labels that can be clearly separated as a function of two informed features, I_0 and I_1
- We add two non-linear redundant features:
  - R_0=cos[I_0]
  - R_1=cos[I_1]
- We add two noise feature, N_0 and N_1

- The system is not multicollinear, however there are strong non-linear substitution effects between the features





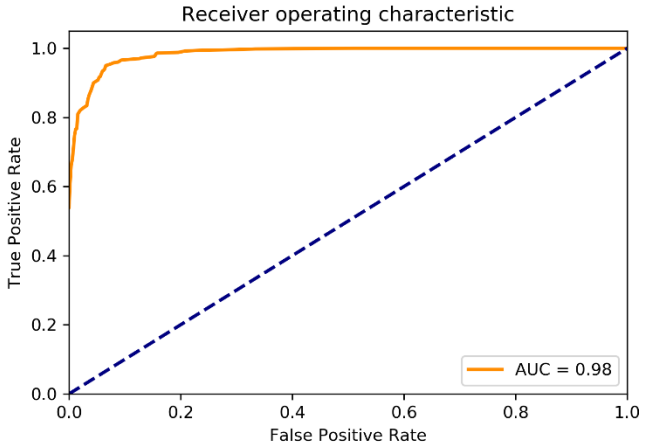
p-Values

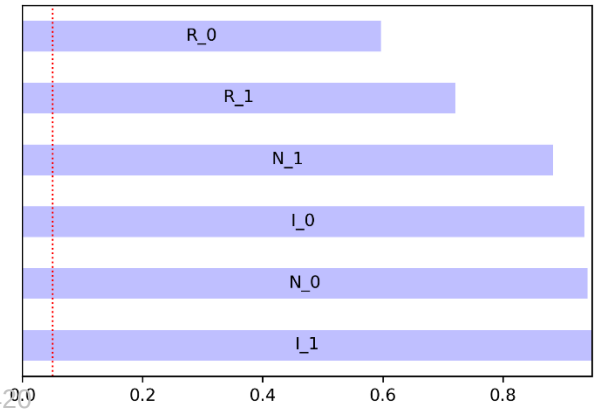
- The model is correctly specified
  - The redundant features allow logit to separate the labels well, with an ROC of almost 1

- Despite of the high ROC, *all* *p*-values exceed the 5% threshold. Why?

- The reason is, the substitution effects do not allow logit to estimate the parameters robustly

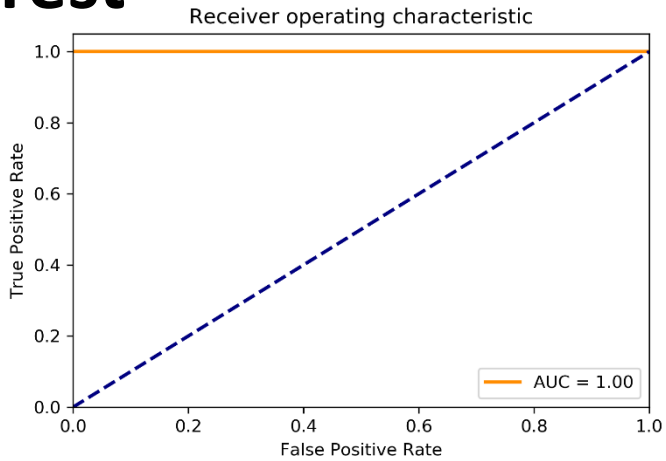
- One again, *p*-values are unable to select the important features

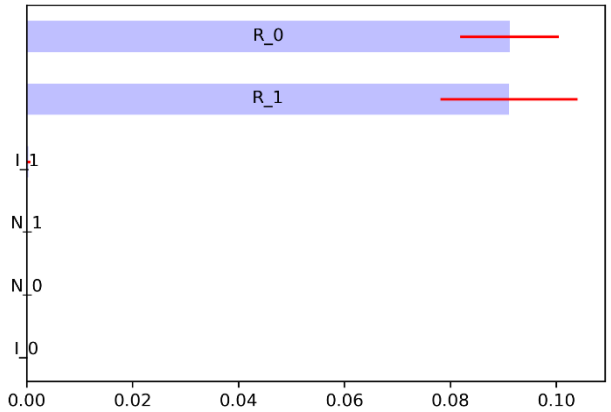




MDA on Random Forest

- At AUC=1, the model is correctly specified
- MDA correctly identifies R_0 and R_1 as important
- However, MDA incorrectly dismisses I_0 and I_1
- MDA (partially) failed because of the non-linear substitution effects
  - (R_0, R_1) prevent a reduction of AUC when (I_0, I_1) are shuffled

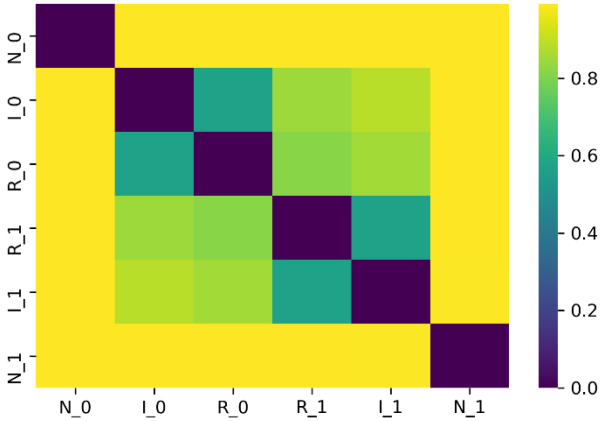


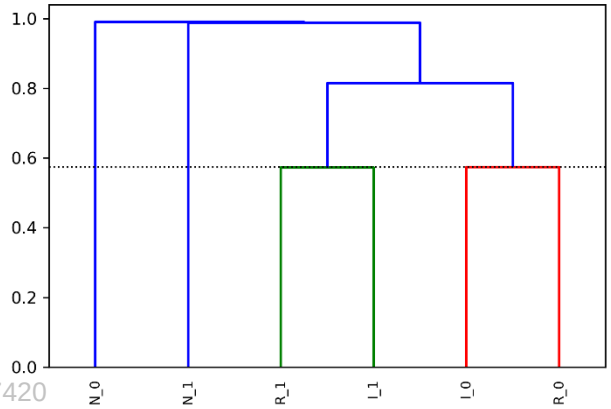


Features Clustering

- Apply a single-linkage agglomerative clustering algorithm on a **variation of information (VI) matrix**
- VI can effectively measure linear as well as non-linear codependence
- The algorithm recognizes that
  - The optimal number of clusters is 4
  - R_0 is redundant to I_0
  - R_1 is redundant to I_1

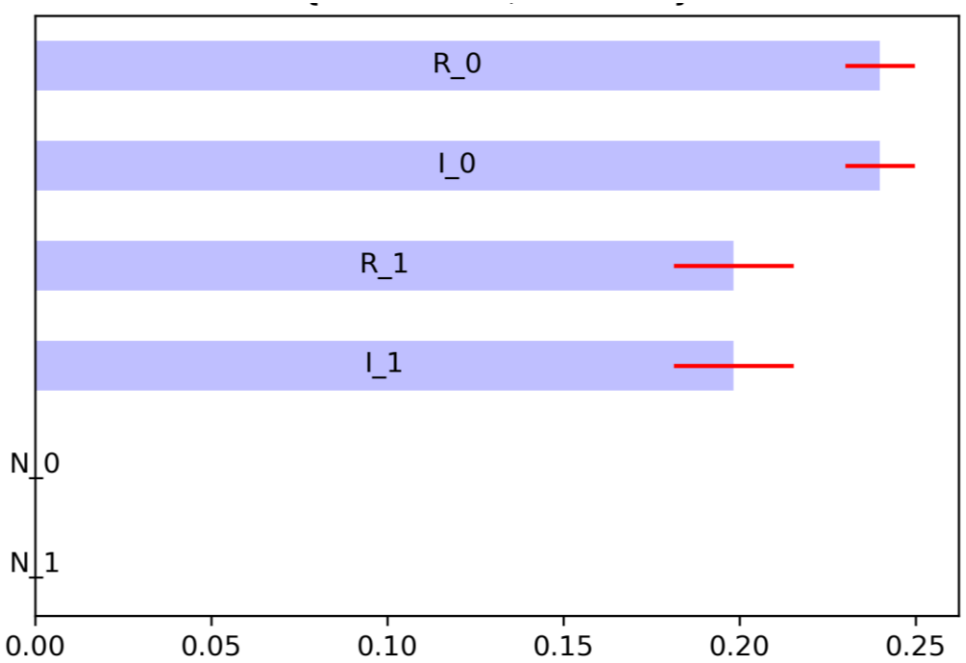
- **The system formed by the clusters does not exhibit substitution effects (no off-diagonal blocks)**





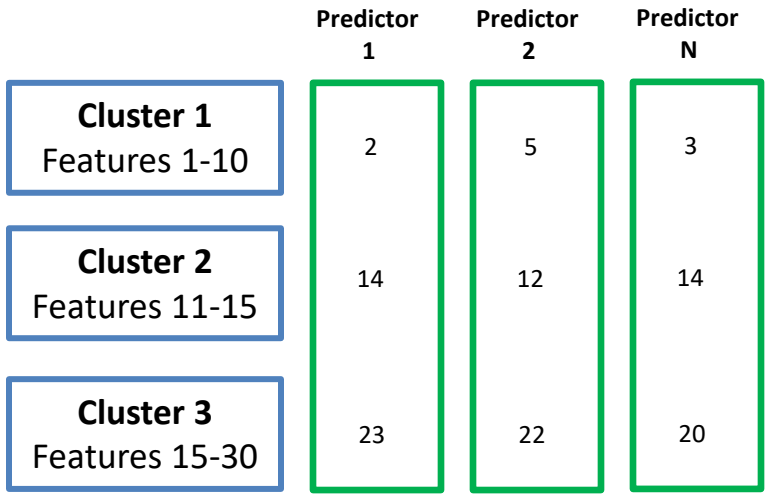
Clustered MDA on Random Forest

- Instead of shuffling each variable individually, we shuffle together all variables within a cluster
- Clustered MDA gives the right answer:
  - All informed and redundant features are important
  - N_0 and N_1 have zero contribution to the model's performance
- Why did Clustered MDA work?
  - MDA decouples the specification search from the significance search
  - Clusters concentrate most of the mutual information, muting substitution effects

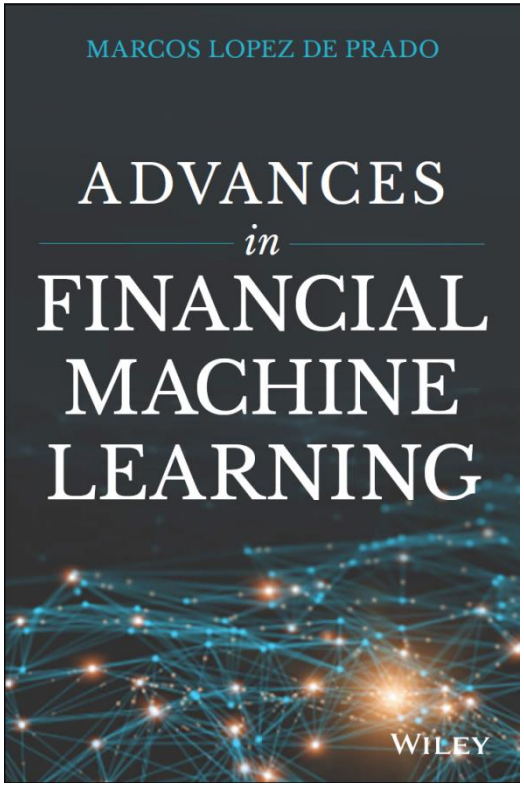


Uses of Clustered MDA

- Clustered MDA identifies the variables involved in a phenomenon
  - With that knowledge, we can hypothesize a particular cause-effect mechanism that binds those variables together
- Clustered MDA avoids substitution effects
  - It provides an intuitive form of regularization
- Moreover, Clustered MDA also helps us fit better ensemble models
  - Each individual predictor is fit by randomly drawing one feature per cluster
  - As a result, the individual predictors exhibit lower correlation, thus reducing the variance of the ensemble predictions



For Additional Details



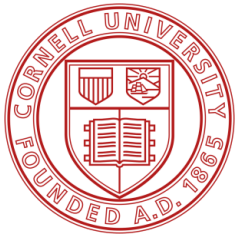
The first wave of quantitative innovation in finance was led by Markowitz optimization. Machine Learning is the second wave and it will touch every aspect of finance. López de Prado's Advances in Financial Machine Learning is essential for readers who want to be ahead of the technology rather than being replaced by it.
— Prof. **Campbell Harvey**, Duke University. Former President of the American Finance Association.

Financial problems require very distinct machine learning solutions. Dr. López de Prado's book is the first one to characterize what makes standard machine learning tools fail when applied to the field of finance, and the first one to provide practical solutions to unique challenges faced by asset managers. Everyone who wants to understand the future of finance should read this book.
— Prof. **Frank Fabozzi**, EDHEC Business School. Editor of The Journal of Portfolio Management.

Disclaimer

- The views expressed in this document are the authors' and do not necessarily reflect those of the organizations he is affiliated with.
- No investment decision or particular course of action is recommended by this presentation.
-

www.QuantResearch.org



Interpretable Machine Learning: Shapley Values

Prof. Marcos López de Prado

Advances in Financial Machine Learning

ORIE 5256

Key Points

- Machine learning (ML) algorithms utilize the power of computers to solve tasks that are beyond the grasp of classical statistical methods
- However, ML is often perceived as a black-box, hindering its adoption
- **In this presentation, we demonstrate the use of Shapley values to interpret the outputs of ML models**
- Shapley values interpret a model's prediction in terms of
  - attribution to the various features (sign and size)
  - features that are most important overall
  - dependence on the feature's value
  - interactions between features
  - similarity (supervised clustering)
- With the help of interpretability methods, ML is becoming the primary tool of scientific discovery, through induction as well as abduction

The Role of ML in Modern Science

Induction & Abduction Through ML

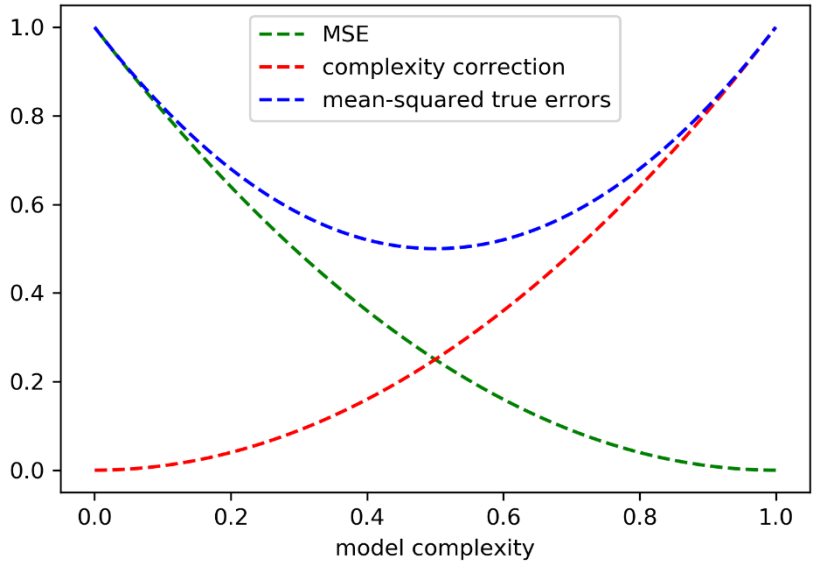
- Scientists discover through induction and abduction
- **Induction**: From cause (X) and effect (y), derive a mechanism (f) such that $y = f[X]$
  - Classical statistics primarily learns by induction
- **Abduction**: From effect (y) and mechanism (f), derive a *plausible* cause (X), such that $y = f[X]$
  - ML is particularly helpful at abduction, because it isolates the causes possibly involved in an effect
- Abduction requires interpretability of f
  - We can only point to plausible causes if we understand the mechanism (no black-box)

Reasoning	Description
Abduction	Researchers propose plausible causes of an effect
Induction	Researchers propose, test, and validate a clear cause-effect mechanism
Deduction	In presence of a cause, an effect is deduced ($A \rightarrow B$). The absence of an effect implies the absence of its cause (not $B \rightarrow$ not A)

Unlike classical statistics, ML decouples the search for X from the search for f, because the researcher finds the X predictive of y without imposing a f. ML's decoupling enables learning X by **abduction**. Once X is fixed, we can learn by **induction** what mechanism f is supported by empirical evidence. Finally, for a given $f[X]$, we can postulate future values of y by **deduction**.

Performance vs. Interpretability Dilemma

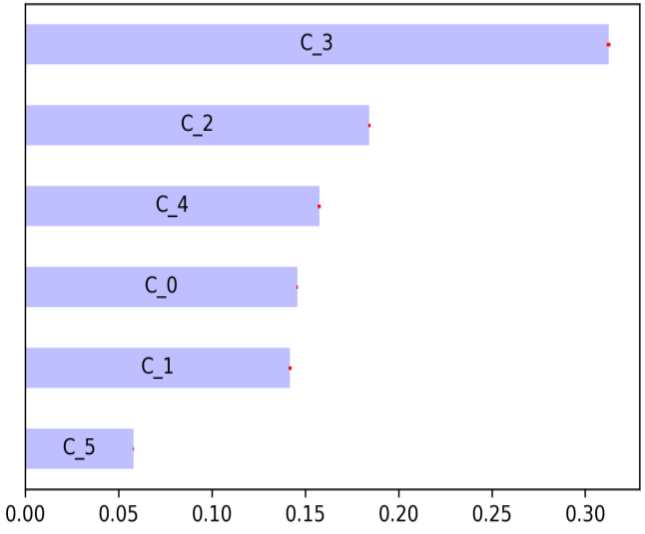
- A model is interpretable when we can understand its decisions
  - Given a model's prediction, we can abduct a likely cause for that predicted effect
- Linear models are popular because they are intrinsically interpretable
  - An effect is the weighted sum of individual causes
- Dilemma:
  - linear models are intrinsically interpretable, but perform poorly
  - nonlinear models are powerful, but not intrinsically interpretable
- Solution: Use approaches that make ML models interpretable (post hoc)



Due to their simplicity, classical statistical models are intrinsically interpretable. However, that interpretability comes at the cost of poor performance, because linear models rarely achieve the minimum mean squared error (MSE).

Taxonomies of ML Interpretability Approaches

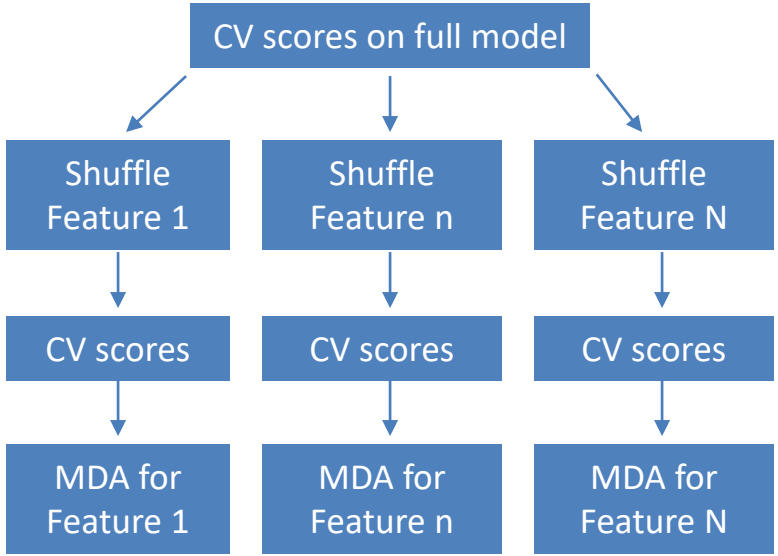
- Types of interpretability
  - **Intrinsic**: The model is interpretable as the result of restricting its complexity (e.g., linear models)
  - **Post hoc**: The model becomes interpretable after applying methods that explain its behavior (e.g., MDA)
- Types of interpretability methods
  - **Specific**: Applicable to certain classes of models (e.g., MDI)
  - **Agnostic**: Applicable to all classes of models (e.g., MDA)
- Interpreted output
  - **Global**: Importance across all observations (e.g., MDI, MDA)
  - **Local**: Importance for a given observation (e.g., LIME)
- Estimation
  - **In-sample**: Importance derived from train-set errors (e.g., MDI)
  - **Cross-validated**: Importance derived from test-set errors (e.g., MDA)



Mean decreased impurity (MDI) is an example of a specific approach (it can only be derived for tree-based models). A feature's importance is measured by its contribution to global impurity reduction at node splits (in-sample).

Example: Permutation Importance (MDA)

- Permutation importance compares the cross-validated performance of a model fit on X, with the performance of the same model after shuffling a particular feature
  - If the feature is important, we should observe a substantial reduction in cross-validated performance
- Advantages
  - Post-hoc, agnostic, global, cross-validated
  - Consistent: if the model changes such that it relies more on a feature, its importance will raise
- Disadvantages
  - No local interpretability: the sign of the effect may change locally, and the approach will not show it
  - Purely experimental: limited theoretical properties

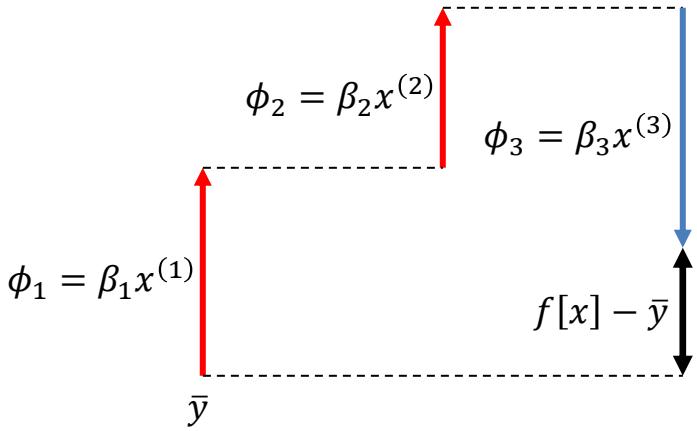


We can compute K losses in performance in a K-fold cross-validation. That allows us to bootstrap the distribution of the generalization error. Next, we will study how **Shapley values** overcome some of MDA's limitations.

Shapley Values

The Problem

- Consider a dataset with
  - N features, $X = \{X^{(1)}, \dots, X^{(N)}\}$, and
  - a real-valued target variable, y
- Given an instance $x \in X$, a model f forecasts the target as $f[x]$
- Question: Can we attribute the departure of a prediction $f[x]$ from its average value $\bar{y}$ in terms of the observed instance x?
- In a linear model, the answer is trivial
  - $f[x] - \bar{y} = \beta_1 x^{(1)} + \cdots + \beta_N x^{(N)}$
- We would like to do a similar *local* decomposition for any (nonlinear) ML model



An attribution of the departure of a model's prediction, $f[x]$, from its average value, $\bar{y}$. Because addition is commutative, the sequence of the effects does not alter the result. Hence, in a linear model, the attribution is invariant to the sequence of effects.

Coalitional Game Theory (1/2)

- Shapley values answer this attribution problem through game theory
  - Features are treated as players in a game
  - The players form coalitions in order to achieve an outcome
  - We wish to find a "fair" attribution of each individual player's contribution
- First, we consider the 2^N possible coalitions (interactions) between the players (features), where
  - some players (features) may not participate (remain at their average value), and
  - some players (features) may participate (depart from their average value)

$x^{(1)} \neq \bar{x}^{(1)}$	$x^{(2)} \neq \bar{x}^{(2)}$	$x^{(3)} \neq \bar{x}^{(3)}$	$f[x\|\ldots] - \bar{y}$
0	0	0	$f[x\|000] - \bar{y}$
0	0	1	$f[x\|001] - \bar{y}$
0	1	0	$f[x\|010] - \bar{y}$
0	1	1	$f[x\|011] - \bar{y}$
1	0	0	$f[x\|100] - \bar{y}$
1	0	1	$f[x\|101] - \bar{y}$
1	1	0	$f[x\|110] - \bar{y}$
1	1	1	$f[x\|111] - \bar{y}$

In a model with 3 features, there are $2^3 = 8$ possible interactions. For each interaction, we compute the departure of the model's forecast from its baseline (average value). We encode as "1" a feature that is not at its average value (it forms part of a coalition), and "0" a feature that is at its average value.

Coalitional Game Theory (2/2)

- Second, we use the coalitions (interactions) table to compute the marginal contribution of each player (feature) conditional to the other players
  - The marginal impact of changing $x^{(i)}$ after changing $x^{(j)}$ may differ from the marginal impact of changing $x^{(j)}$ after changing $x^{(i)}$
  - Accordingly, we must account for all possible $N!$ sequences of conditional effects
- Third, the Shapley value of a player (feature) is the average conditional marginal contribution of that player across all the possible $N!$ ways of conditioning $f[.]$

Sequence	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
$x^{(1)}, x^{(2)}, x^{(3)}$	$f[x\|100] - f[x\|000]$	$f[x\|110] - f[x\|100]$	$f[x\|111] - f[x\|110]$
$x^{(1)}, x^{(3)}, x^{(2)}$	$f[x\|100] - f[x\|000]$	$f[x\|111] - f[x\|101]$	$f[x\|101] - f[x\|100]$
$x^{(2)}, x^{(1)}, x^{(3)}$	$f[x\|110] - f[x\|010]$	$f[x\|010] - f[x\|000]$	$f[x\|111] - f[x\|110]$
$x^{(2)}, x^{(3)}, x^{(1)}$	$f[x\|111] - f[x\|011]$	$f[x\|010] - f[x\|000]$	$f[x\|011] - f[x\|010]$
$x^{(3)}, x^{(1)}, x^{(2)}$	$f[x\|101] - f[x\|001]$	$f[x\|111] - f[x\|101]$	$f[x\|001] - f[x\|000]$
$x^{(3)}, x^{(2)}, x^{(1)}$	$f[x\|111] - f[x\|011]$	$f[x\|011] - f[x\|001]$	$f[x\|001] - f[x\|000]$

In a model with 3 features, there are $3! = 6$ possible sequences. We can use the interactions table to derive the marginal contribution of each feature in each sequence. The Shapley values are the averages per column.

A Numerical Example

Coalitions table

$f[x\|\ldots]-\bar{y}$	Observations
$f[x\|000]-\bar{y}$	0
$f[x\|001]-\bar{y}$	20
$f[x\|010]-\bar{y}$	25
$f[x\|011]-\bar{y}$	30
$f[x\|100]-\bar{y}$	10
$f[x\|101]-\bar{y}$	28
$f[x\|110]-\bar{y}$	37
$f[x\|111]-\bar{y}$	50

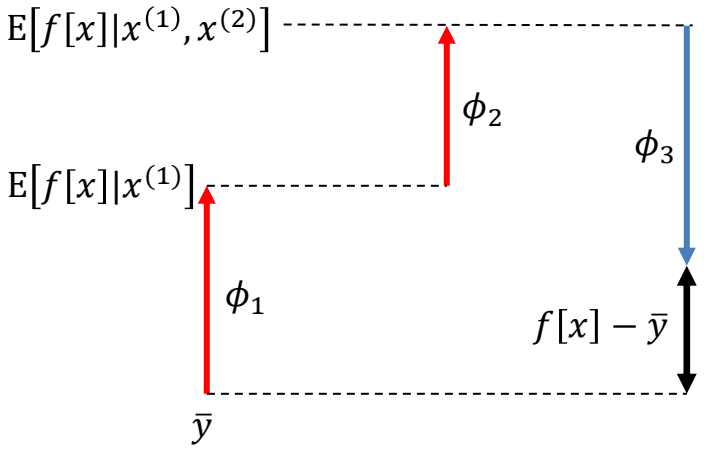
Marginal conditional contributions table

Sequence	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	Sum
$x^{(1)},x^{(2)},x^{(3)}$	10-0=10	37-10=27	50-37=13	50
$x^{(1)},x^{(3)},x^{(2)}$	10-0=10	50-28=22	28-10=18	50
$x^{(2)},x^{(1)},x^{(3)}$	37-25=12	25-0=25	50-37=13	50
$x^{(2)},x^{(3)},x^{(1)}$	50-30=20	25-0=25	30-25=5	50
$x^{(3)},x^{(1)},x^{(2)}$	28-20=8	50-28=22	20-0=20	50
$x^{(3)},x^{(2)},x^{(1)}$	50-30=20	30-20=10	20-0=20	50
Average	13.34	21.83	14.83	50

Example of Shapley values for an observation (x, y) in a model f with 3 features.

Properties of Shapley Values

- Shapley values ($\{\phi_i\}_{i=1,\dots,N}$) provide the only "fair" attributions, i.e. attributions with the following properties
  - **Efficiency**: $f[x] - \bar{y} = \sum_{i=1}^{N} \phi_i$
  - **Symmetry**: $\phi_i = \phi_j$ IIF $x^{(i)}$ and $x^{(j)}$ contribute equally to all possible coalitions
  - **Missingness**: if $x^{(i)}$ does not change $f[x]$ regardless of the coalition, then $\phi_i = 0$
  - **Additivity**: a function with combined outputs has as Shapley values the sum of the constituent ones
- Shapley values and MDA values are **consistent** (unlike MDI values)
  - if the model changes such that it relies more on a feature, that feature's importance will rise



The marginal conditional contribution for a feature depends on the particular sequence of effects. Shapley values achieve **efficiency** by averaging across all possible sequences (hence the above $\mathrm{E}[f[x]|\dots]$ attributions).

From Permutations to Combinations

- As we can appreciate from the numerical example, some calculations are redundant
  - E.g., the marginal contribution of $x^{(3)}$ on sequence $x^{(1)}, x^{(2)}, x^{(3)}$ must be the same as the marginal contribution of $x^{(3)}$ on sequence $x^{(2)}, x^{(1)}, x^{(3)}$, because
    - in both cases $x^{(3)}$ comes in third position, and
    - the permutations of $\{x^{(1)}, x^{(2)}\}$ do not alter that marginal contribution
- The Shapley value of $x^{(i)}$ can be derived as the average contribution of $x^{(i)}$ across all possible coalitions S, where S does not include i

$$\phi_i = \frac{1}{N} \sum_{S \subseteq (X \setminus \{i\})} \binom{N-1}{\|S\|}^{-1} (f[S \cup \{i\}] - f[S])$$

$$= \sum_{S \subseteq (X \setminus \{i\})} \frac{\|S\|!\,(N - \|S\| - 1)!}{N!} (f[S \cup \{i\}] - f[S])$$

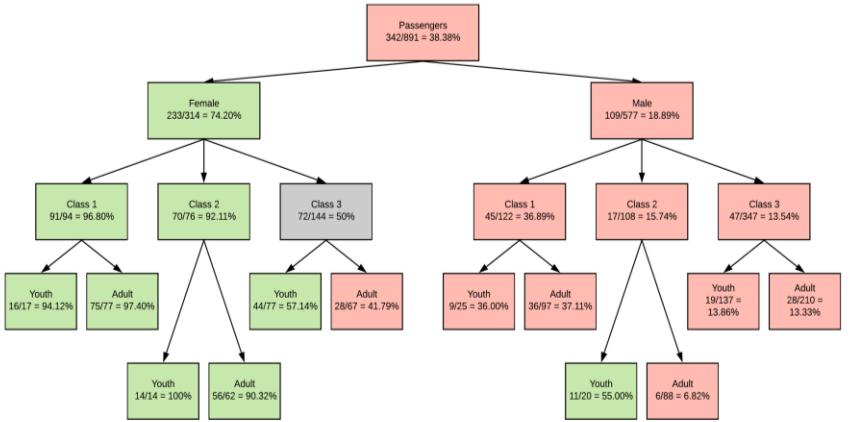
Coalitions can have sizes $\|S\| = 0,1, \ldots, N-1$. A coalition S produces non-redundant marginal contributions $f[S \cup \{i\}] - f[S]$ for $\binom{N-1}{\|S\|}$ combinations. The above equation computes the exact Shapley values by grouping the marginal conditional contributions in terms of coalitions (S). For large N, Strumbelj et al. [2014], Lundberg and Lee [2016], and Lundberg et al. [2019] have developed fast algorithms for the estimation of ϕ_i.

Interaction Effects

- An interaction effect occurs when the effect of one variable on the output depends on the value of another variable
- The estimation of interaction effects requires having an accurate attribution of the individual effect of the variables involved
  - Shapley values are particularly useful for estimating interaction effects
  - For $i \neq j$,

$$\phi_{i,j} = \sum_{S \subseteq (X \setminus \{i,j\})} \frac{\|S\|!\,(N - \|S\| - 2)!}{2(N-1)!}\,\delta_{i,j}[S]$$

where $\delta_{i,j}[S] = f[S \cup \{i,j\}] - f[S \cup \{i\}] - f[S \cup \{j\}] + f[S]$

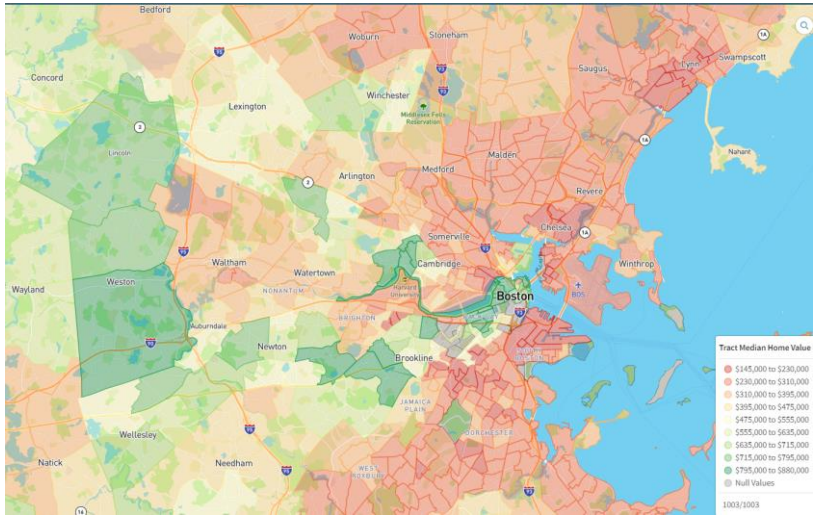


The survival probability of RMS Titanic passengers. There are interaction effects between gender and ticket class (e.g., females in class 3 relative to females in classes 1 & 2), and age and ticket class (e.g., young males in class 2 relative to adult males in class 2). Shapley values help us quantify the strength of these interactions.

Case Study:
Boston Housing Prices

The Data

- Target: prices for 506 residential properties
- Features (14):
  - **CRIM**: per capita crime rate by town
  - **ZN**: proportion of residential land zoned for lots over 25,000 sq.ft.
  - **INDUS**: proportion of non-retail business acres per town
  - **CHAS**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
  - **NOX**: nitric oxides concentration (parts per 10 million)
  - **RM**: average number of rooms per dwelling
  - **AGE**: proportion of owner-occupied units built prior to 1940
  - **DIS**: weighted distances to five Boston employment centers
  - **RAD**: index of accessibility to radial highways
  - **TAX**: full-value property-tax rate per $10,000
  - **PTRATIO**: pupil-teacher ratio by town
  - **B**: $1000(Bk - 0.63)^2$, where Bk is the proportion of blacks by town
  - **LSTAT**: proportion of the population that is poor
  - **MEDV**: Median value of owner-occupied homes in $1000's

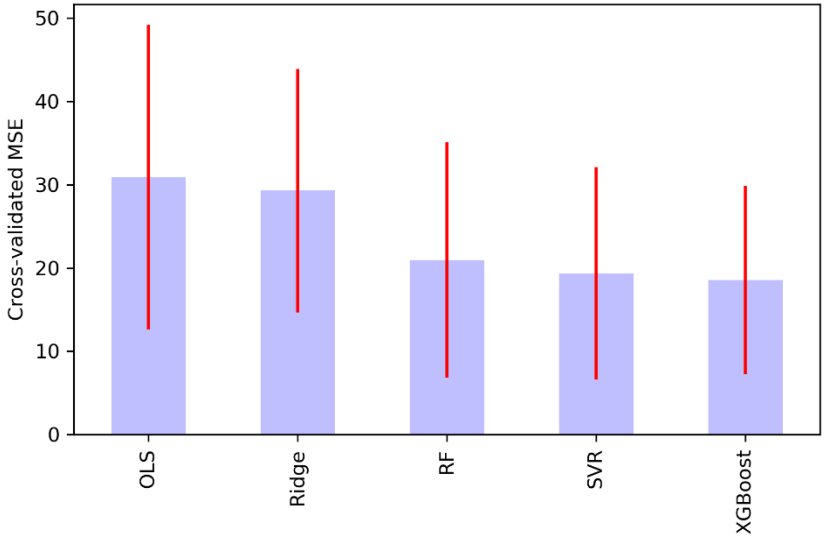


Source: Harvard University

The Boston house-price data, collected by Harrison and Rubinfeld [1978], has been used in multiple of ML studies, tournaments and research papers.

The Model

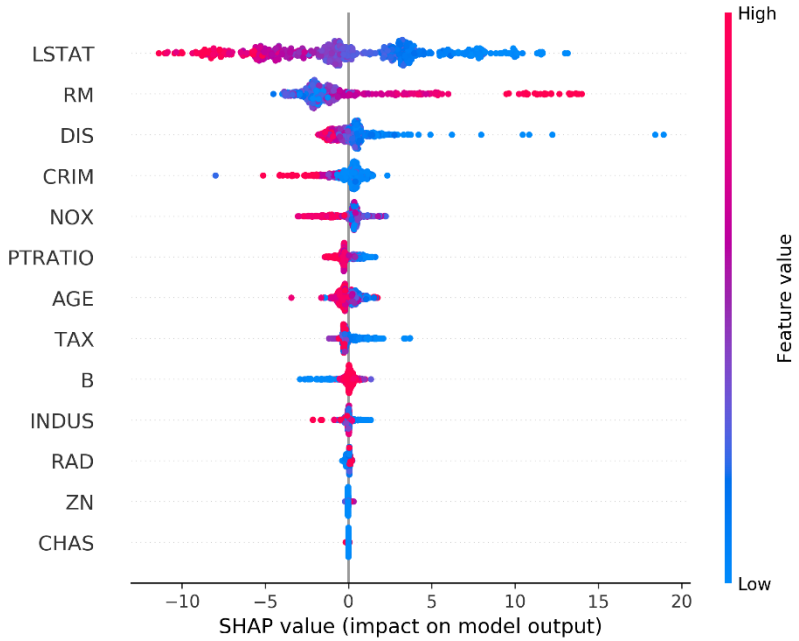
- The data exhibits strong non-linearities, which are not well captured by linear models (OLS, Ridge)
- The best performing model (XGBoost) is not intrinsically interpretable
- We can develop an understanding of the uncovered patterns by attributing the effect (housing prices) to the various features
  - In a linear model, we would simply look at the size and sign of the estimated coefficients
  - In a nonlinear model, we must study this attribution on a case-by-case basis: **Shapley values**
- By **abduction**, this attribution gives us the plausible drivers of housing prices



Above is a bar plot of the mean squared errors (with 1-std error bars) derived through K-Fold cross-validation. Linear models (OLS, Ridge) performed considerably worse than non-linear ML algorithms (Random Forest regression, Support Vector Regressor, XGBoost) on this small dataset.

Feature Importance Plot

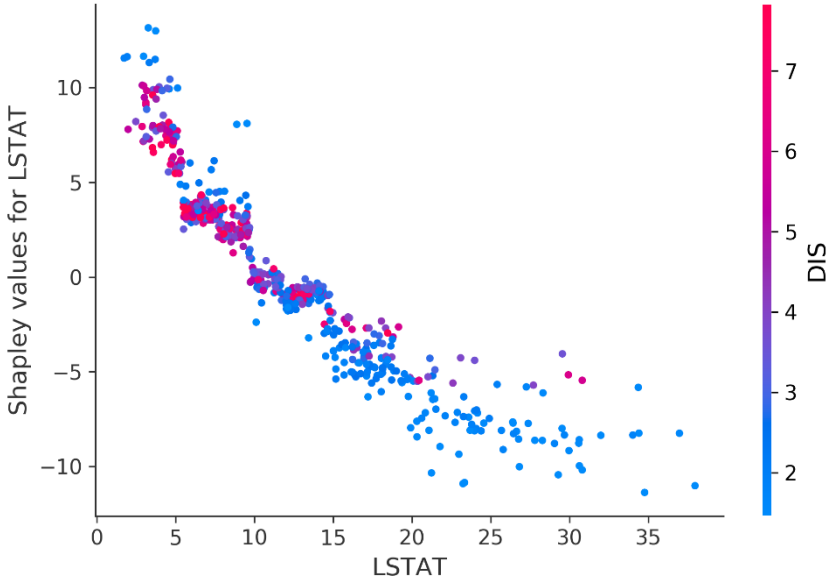
- It displays the Shapley value for every observation, grouped by feature
  - The **y-axis** ranks the features by importance
    - Importance is measured as average absolute Shapley values
  - The **x-axis** shows the magnitude of the impact
    - When many observations for a given feature have a similar impact, the line of dots widens
  - The **color** shows the magnitude of the feature
- In this example, we learn that
  - The top drivers of housing prices are poverty rate (LSAT), number of rooms (RM), distance to employment centers (DIS), crime rate (CRIM), and air pollution (NOX)
  - All of these drivers impact prices negatively, except for RM



The signs of the relationships match intuition, providing support that the XGBoost model has found patterns with a theoretical foundation.

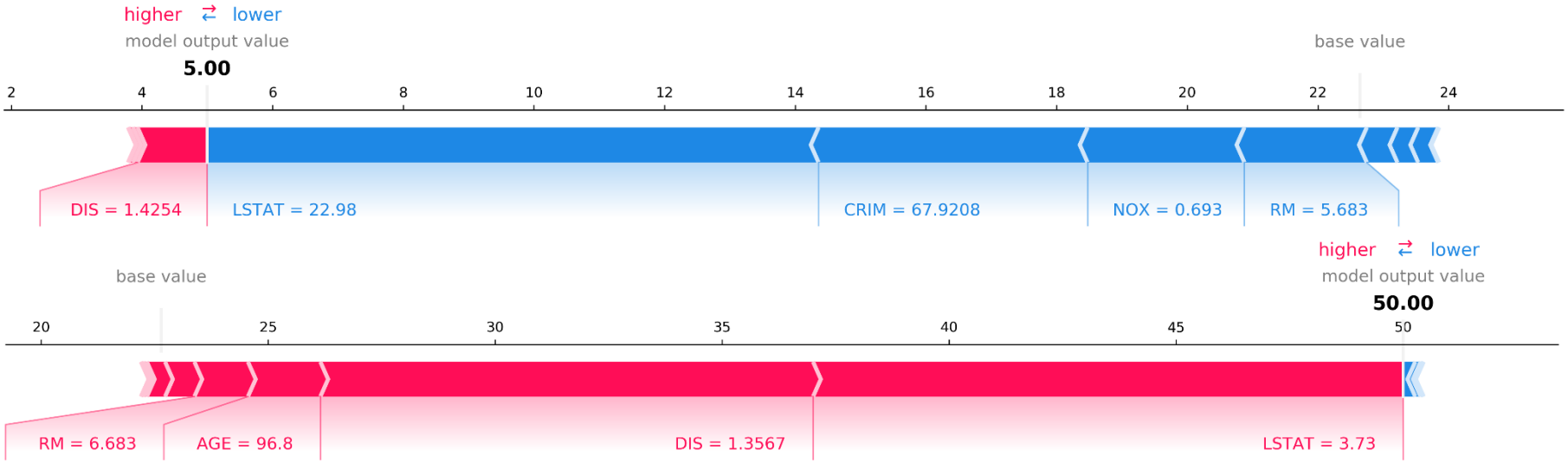
Dependence Plot

- For any *pair* of features, it displays
  - the Shapley values (y-axis) as a function of one feature's value (x-axis)
    - E.g., a linear, negative, monotonic effect
  - the interaction between that feature (x-axis) and another (dot color)
    - E.g., a color separation in dots across the y-axis
- In this example, we learn that
  - the poverty rate (LSAT) has a negative impact on prices (consistent with the importance plot)
  - the effect is almost linear for LSAT between 5% and 20%, but more acute around the extremes
  - there are strong interactions between LSAT and DIS (distance to employment centers):
    - **High LSAT is detrimental, particularly with low DIS (inner cities)**
    - **Low LSAT is beneficial, particularly with low DIS (downtown)**



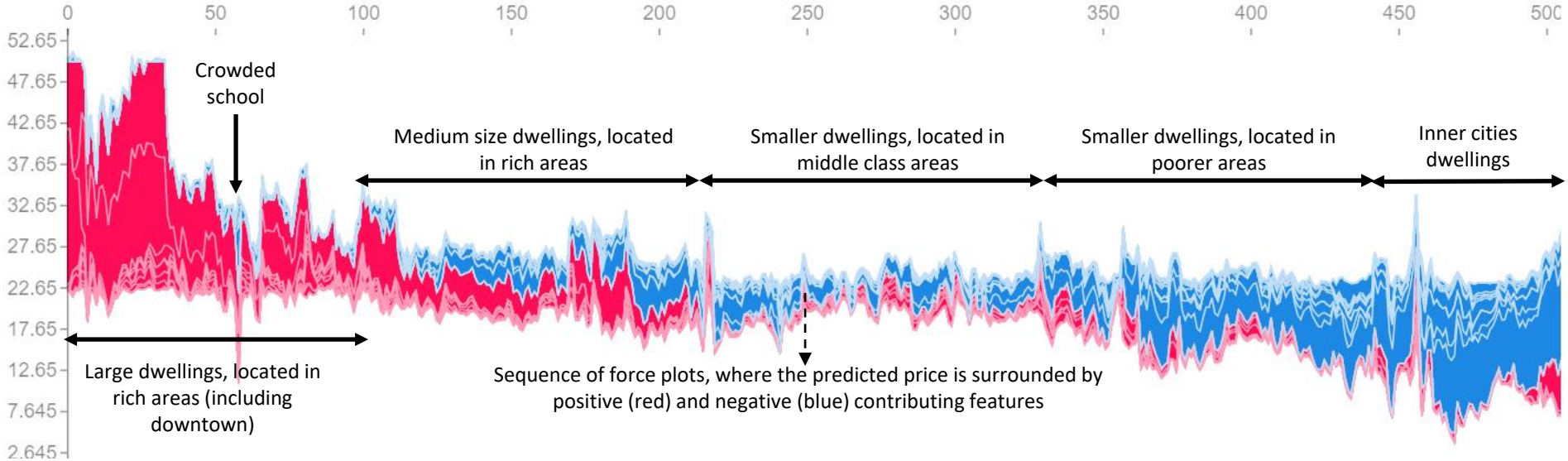
The interaction effect between LSAT and DIS is not monotonic. It flips sign around LSAT of 12%: for higher LSAT values, red dots are on top, and for lower LSAT values, red dots are at the bottom.

Force Plot



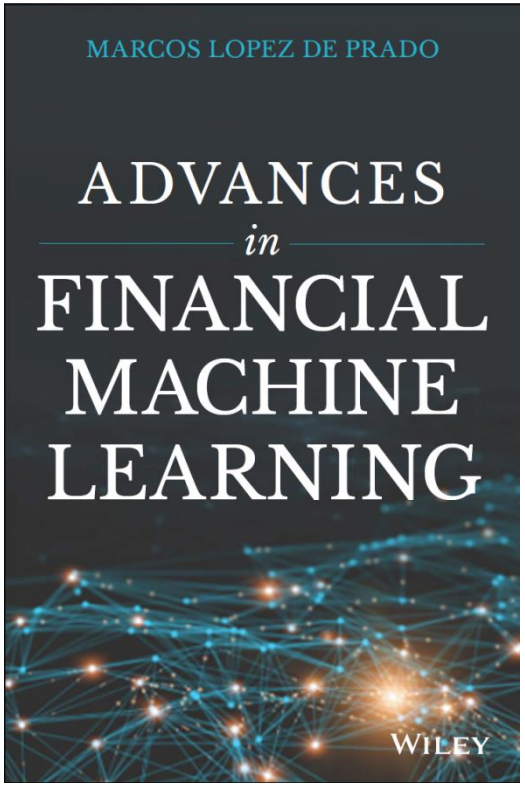
A force plot breaks down the contribution per feature to a particular prediction. The top plot shows the attribution for the house with lowest predicted price (5.00), and the bottom plot shows the attribution for the house with highest predicted price (50.00). The base value is the average prediction (22.64). Features colored in blue detract from the predicted price (e.g., high LSAT, of 22.98), and featured colored in red add to the predicted price (e.g., low LSAT, of 3.73). Features are ordered by the magnitude of their impact on the prediction. Note that the low DIS contributed to increase the price prediction in both examples, in congruence with the feature importance plot.

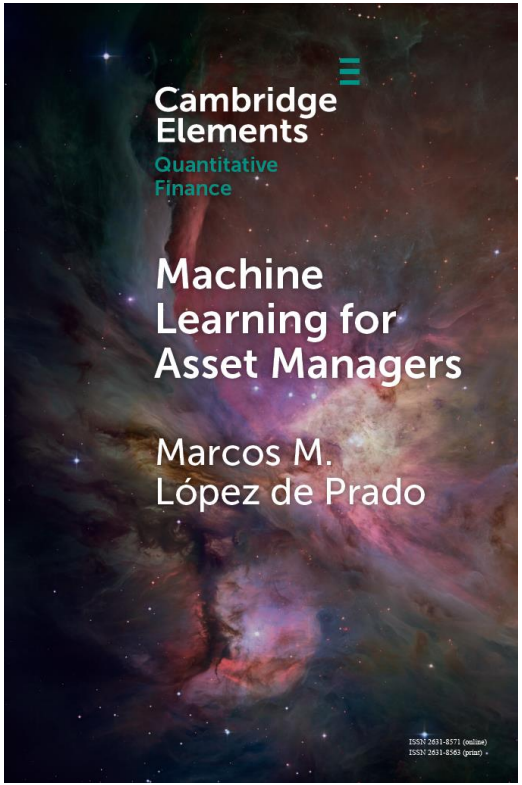
Supervised Clustering



A unit change in one feature's Shapley value is comparable to a unit change in another feature's Shapley value, because all Shapley values are expressed in the same (target variable, y) unit. We can use the matrix of Shapley values to form a distance matrix between observations. The hierarchical clustering of that distance matrix gives us a new sequence of observations, ordered by similarity of attributions (observations with similar reasons for a predicted outcome are placed together). This supervised clustering allows us to intuitively summarize all housing prices into 5 major categories.

For Additional Details





The first wave of quantitative innovation in finance was led by Markowitz optimization. Machine Learning is the second wave and it will touch every aspect of finance. López de Prado's Advances in Financial Machine Learning is essential for readers who want to be ahead of the technology rather than being replaced by it.
— Prof. **Campbell Harvey**, Duke University. Former President of the American Finance Association.

Financial problems require very distinct machine learning solutions. Dr. López de Prado's book is the first one to characterize what makes standard machine learning tools fail when applied to the field of finance, and the first one to provide practical solutions to unique challenges faced by asset managers. Everyone who wants to understand the future of finance should read this book.
— Prof. **Frank Fabozzi**, EDHEC Business School. Editor of The Journal of Portfolio Management.

Disclaimer

- The views expressed in this document are the author's and do not necessarily reflect those of the organizations he is affiliated with.
- No investment decision or particular course of action is recommended by this presentation.
-

www.QuantResearch.org