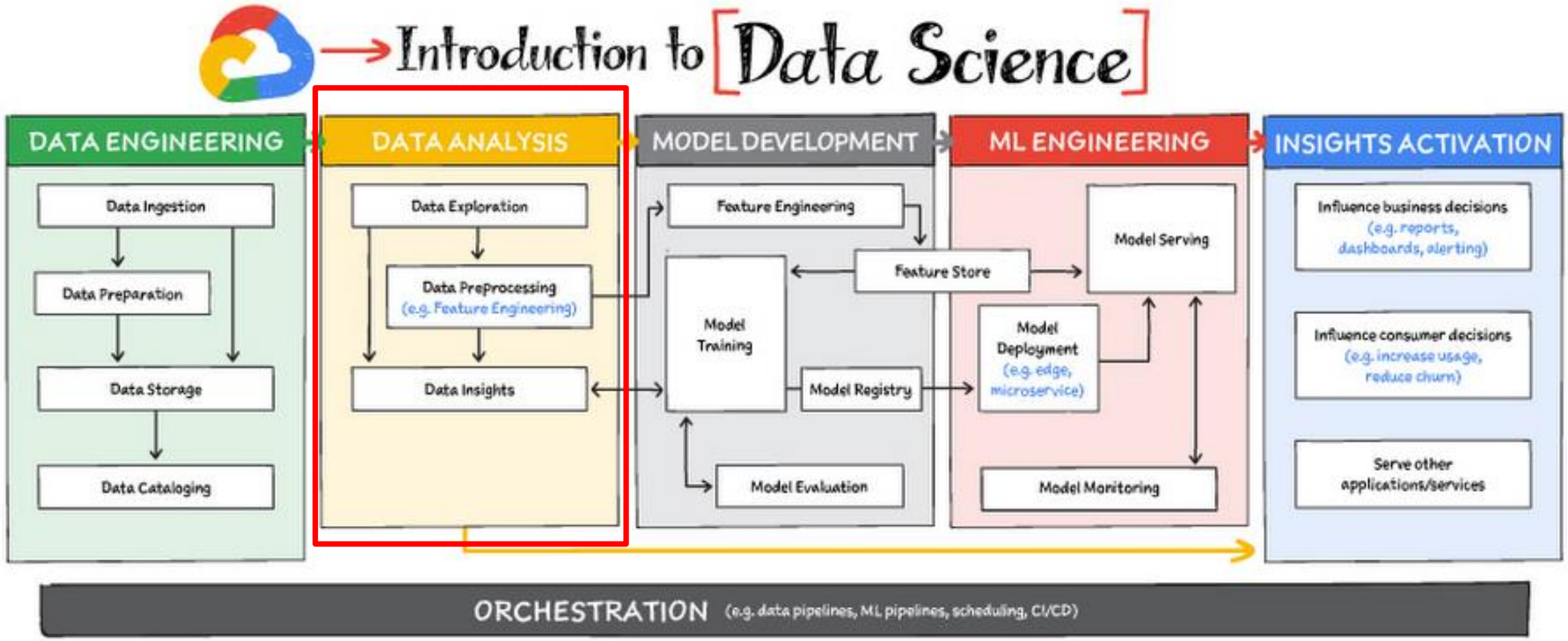
Feature selection and extraction

Szu-Chi Chung

Department of Applied Mathematics, National Sun Yat-sen University

The Pipeline



https://cloud.google.com/blog/topics/developers-practitioners/intro-data-science-google-cloud

How to select/extract important features after cleaning?

- *Feature selection* is the process of selecting a subset of relevant features for use in model construction. It is used for several reasons:
  - To avoid the curse of dimensionality
  - Shorter training times
  - Simplification of models to make them easier to interpret by researchers/users
- The central premise is that the data contains some features that are either *redundant* or *irrelevant* and can thus be removed without incurring much loss
- It should be distinguished from *feature extraction*. The latter creates new features from functions of the original features

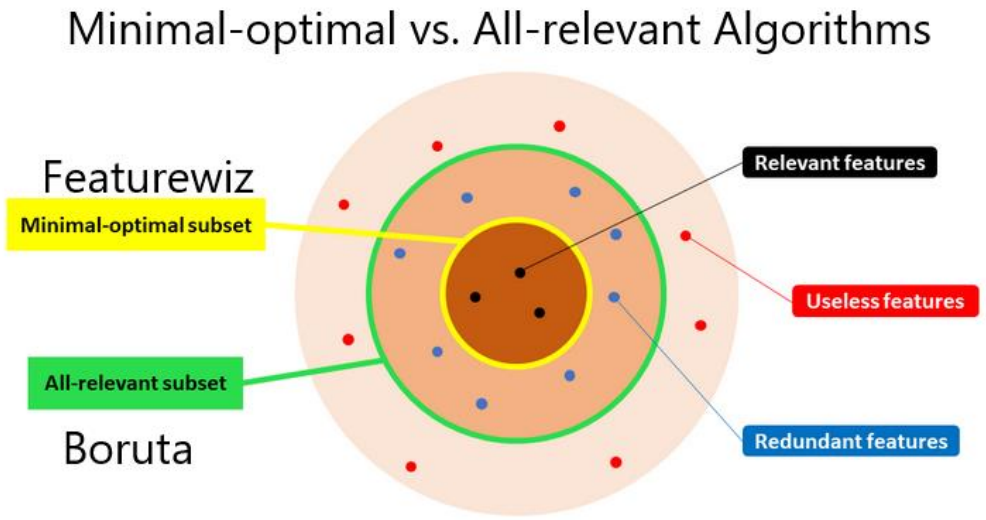


Figure created by Samuele Mazzanti

https://github.com/AutoViML/featurewiz

How to select/extract important feature?

1. Univariate and subset selection: We identify a subset of the p predictors that we believe to be related to the response. We then fit a model on the *reduced set of variables*
2. Select from model using importance metrics: We fit a model involving *all* p predictors, but the estimated coefficients or the importance of features are used to rank the features
3. Select from model shrinkage: We fit a model involving *all* p predictors, but the estimated coefficients are shrunken toward zero
4. Dimension Reduction/Clustering: We project/cluster the p predictors into an M-dimensional subspace, where $M < p$. Then these *M projections/clusters are used as predictors* to fit

1. Univariate feature selection (filtering method)

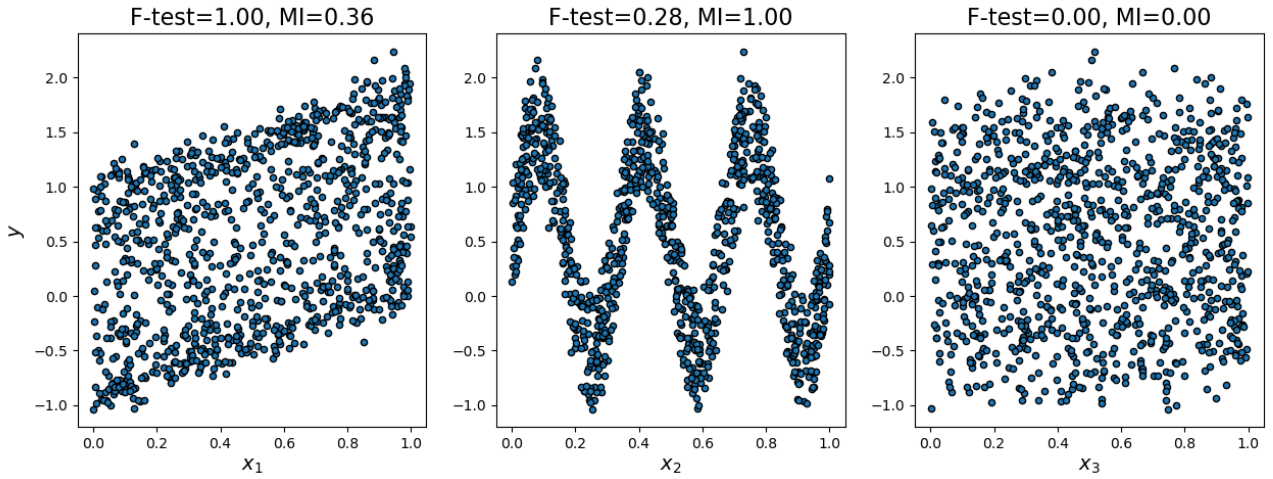
- A great first step is to construct a ranking with a feature utility metric, a function measuring associations *between a feature and the target*
  - The simple baseline is to remove all features whose variance doesn't meet some threshold. For instance, zero-variance features (have the same value in all samples) can be removed

- Univariate feature selection often works by selecting the best features based on univariate statistical tests following by filtering steps:
  - Select *K* best features: removes all but the highest scoring features
  - Select by percentile: removes all but a user-specified highest-scoring percentage of features

Univariate feature selection

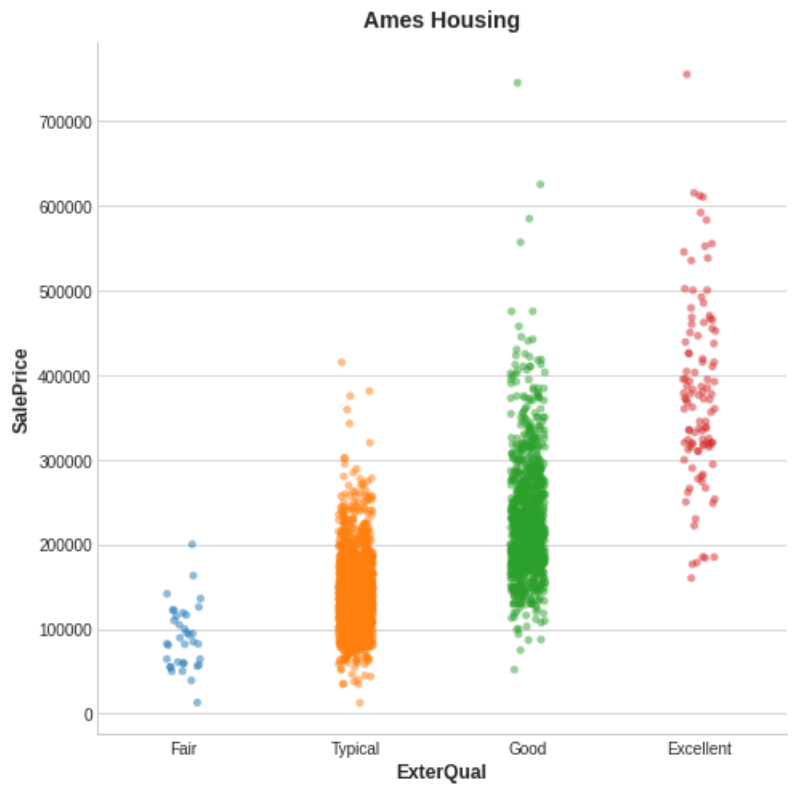
- The statistical test can return univariate scores or p-values:
  - For regression: Cross-correlation, F-test or mutual information
  - For classification: χ^2, F-test or mutual information
- F-test estimate the degree of linear dependency between two random variables
- Mutual information methods can capture any kind of statistical dependency, but being nonparametric, they require more samples for accurate estimation

$$F = \left(\frac{n-p-1}{p}\right)\left(\frac{r^2}{1-r^2}\right), p = 1$$

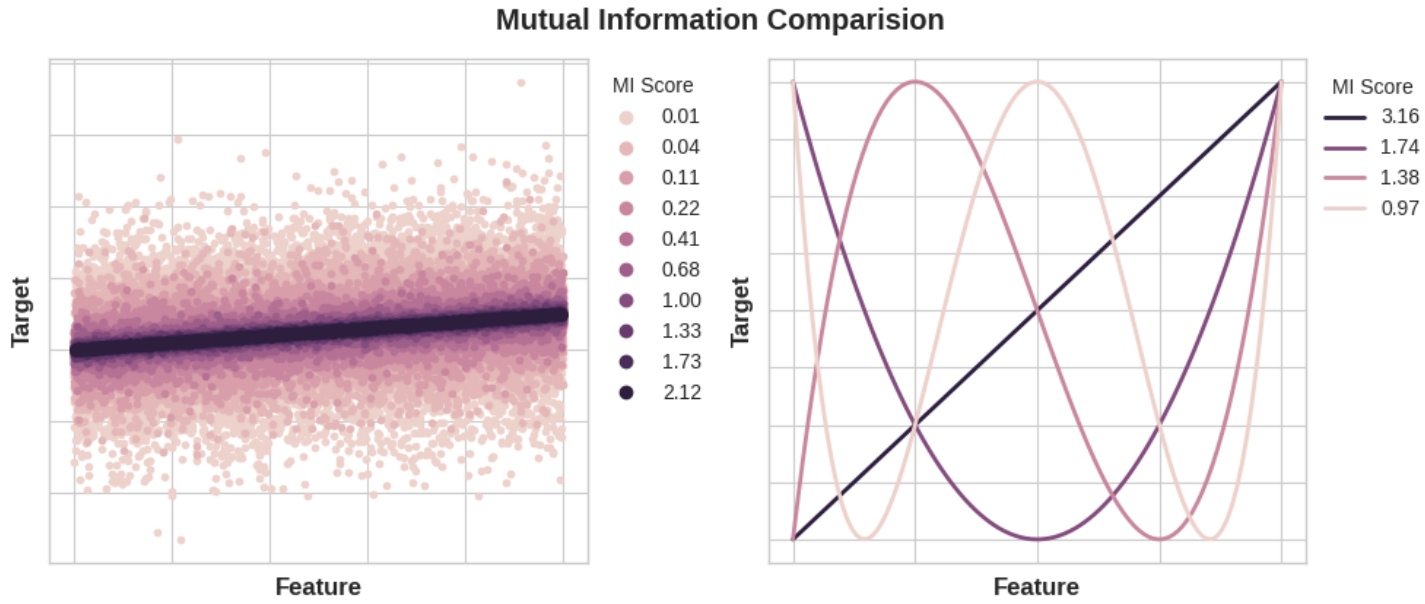


Univariate feature selection - Mutual information

- Mutual information (MI) is a lot like correlation in that it measures a relationship between two quantities
  - The MI between two quantities is a measure of the extent to which knowledge of one quantity reduces uncertainty about the other
  - We can see that knowing the value of `ExterQual` should make you more certain about the corresponding `SalePrice` - each category of `ExterQual` tends to concentrate `SalePrice` to within a certain range



Univariate feature selection - Mutual information



- The actual usefulness of a feature depends on the model you use it with. A feature is only useful to the extent that its relationship with the target is *one your model can learn*. Just because a feature has a high MI score doesn't mean your model will be able to do anything with that information. You may need to transform the feature first to expose the association
- It's possible for a feature to be informative when interacting with other features, but not so informative all alone. **MI can't detect interactions between features**

Multivariate method - subset selection (wrapper method)

Best subset selection procedures

1. Let M_0 denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation
2. For $k = 1,2, \ldots p$:
   a) Fit all $\binom{p}{k}$ models that contain exactly k predictors
   b) Pick the best among these $\binom{p}{k}$ models, and call it M_k. Here best is defined as having the smallest RSS or loss
3. Select a single best model from among $M_0, \ldots, M_p$ using cross-validated score, C_p (AIC), BIC, or adjusted R^2

- Note that we need to replace RSS with deviance for classification setting

Multivariate method - stepwise selection

- For computational reasons, best subset cannot be applied with large p
- Best subset selection may also suffer from statistical problems when p is large: the larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data
  - Thus, an enormous search space can lead to overfitting and high variance
- Stepwise methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection
  - *Forward stepwise selection* begins with a model containing no predictors, and then adds predictors to the model, one at a time, until all of the predictors are in the model
  - In particular, at each step, the variable that gives the greatest additional improvement to the fit is added to the model

In detail

Forward Stepwise Selection

1. Let M_0 denote the null model, which contains no predictors
2. For $k = 0, \ldots p - 1$:
   a) Consider all $p - k$ models that augment the predictors in M_k with one additional predictor
   b) Choose the best among these $p - k$ models, and call it M_{k+1}. Here best is defined as having the smallest RSS or loss
3. Select a single best model from among $M_0, \ldots, M_p$ using cross-validated prediction score, C_p (AIC), BIC, or adjusted R^2

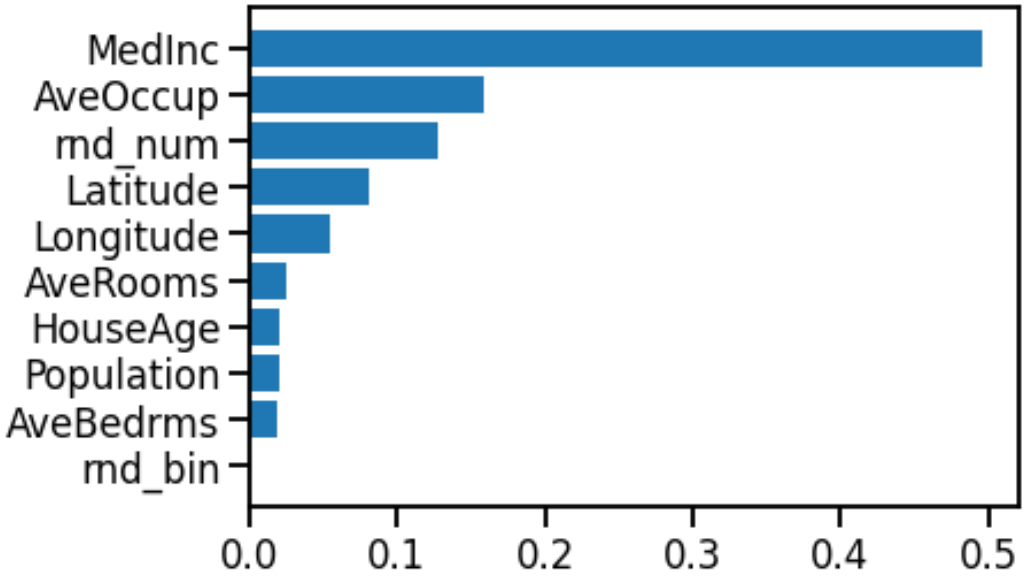
- Though forward stepwise selection considers $p(p + 1)/2 + 1$ models, it performs a *guided search* over model space, and so the effective model space considered contains substantially more than $p(p + 1)/2 + 1$ models

More on forward stepwise selection

- ✓ Computational advantage over best subset selection is clear
- ✓ For high dimensional data with $p > n$, the forward selection can still be applied by considering only $M_1, \ldots, M_n$
  - In some models, like linear regression, this will be a strength
- x It is not guaranteed to find the best possible model out of all 2^p models containing subsets of the p predictors
  - For instance, suppose that in a given data set with $p = 3$ predictors, the best possible one-variable model contains X_1, and the best possible two-variable model instead contains X_2 and X_3. Then forward stepwise selection will fail to select the best possible two-variable model, because M_1 will contain X_1, so M_2 must also contain X_1 together with one additional variable

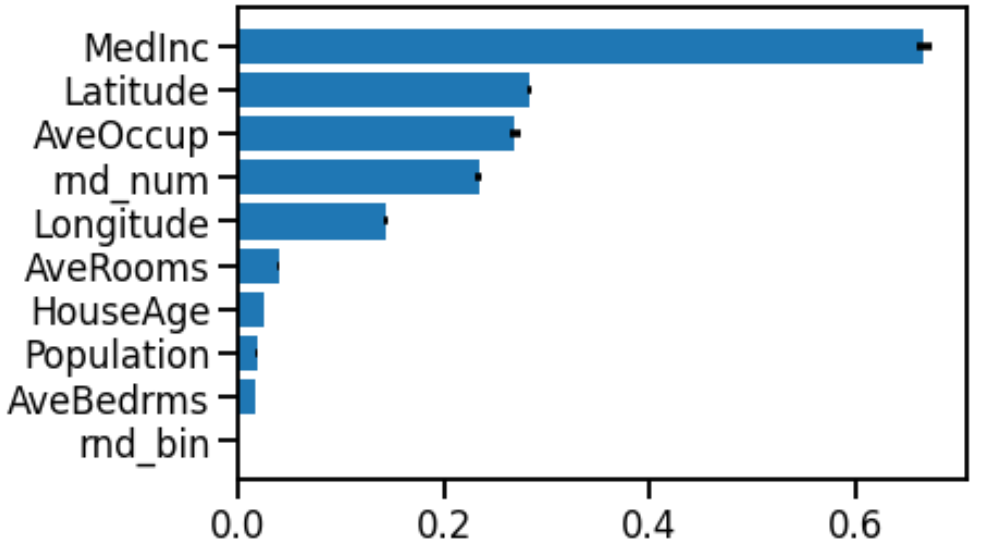
2. Select from model using importance metrics (embedding method)

- Any estimator that assigns importance to each feature through a specific attribute (such as *feature importance*)
  - The features are considered unimportant and removed if the corresponding importance of the feature values is below the provided *threshold*
  - For tree base model the importance of a feature may be how much this feature is used in each tree of the forest. Formally, it is computed as the (normalized) *total reduction of the criterion brought by that feature*
  - It will have a small bias toward high cardinality features (typically numerical features)



Select from model - Permutation importance

- A technique to evaluate the feature importance of any given *fitted model*. It basically shuffles a feature and sees how the model changes its prediction. The change in prediction will correspond to the *feature importance*
  - We will shuffle this specific feature, keeping the other feature as is, and run our same model (already fitted) to predict the outcome. *The decrease in the score shall indicate how the model used this feature to predict the target*.
  - If the feature is not used by the model, the score shall remain the same, thus the feature importance will be close to 0!



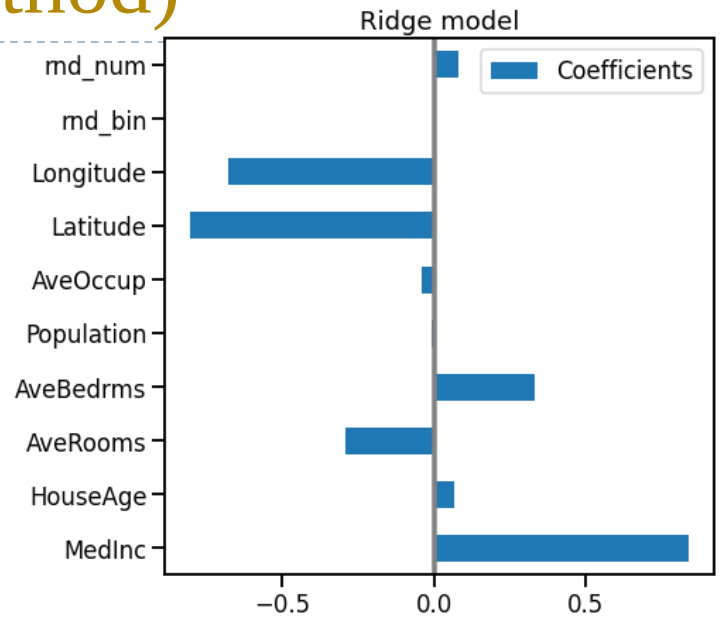
Select from model - Permutation importance

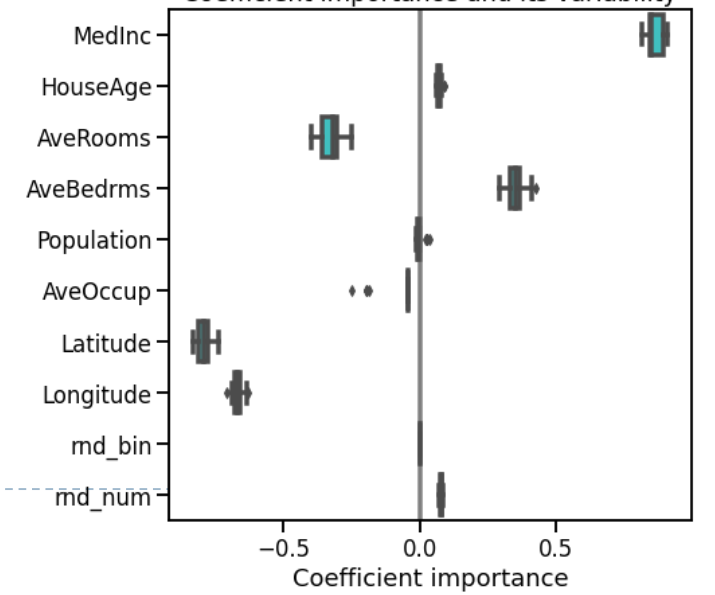
- Tree-based models can give high importance to features that may not be predictive of unseen data when the model is overfitting. Permutation-based feature importance avoids this issue *since it can be computed on unseen data*

- When two features are correlated and one of the features is permuted, the model will still have access to the feature through its correlated feature. This will result in a lower importance value for both features, where they might be important. One way to handle this is to cluster features that are correlated and only keep one feature from each cluster

3. Select from model shrinkage (embedding method)

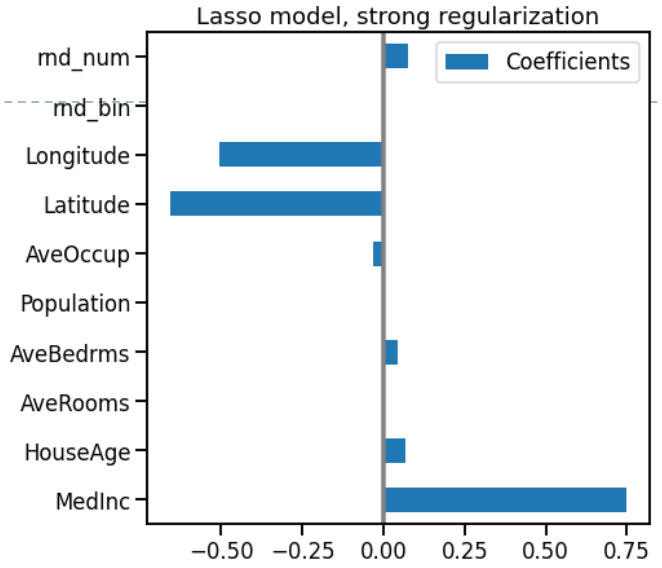
- In linear models, coefficients represent the relationship between the given feature X and the target y, assuming that all the other features remain constant
  - In practice, *ridge regression* is often used
    - Before any interpretation, we may need to scale each column
  - We can check the coefficient variability through cross-validation. If coefficients vary significantly when changing the input dataset, their robustness is not guaranteed, and they should probably be interpreted with caution

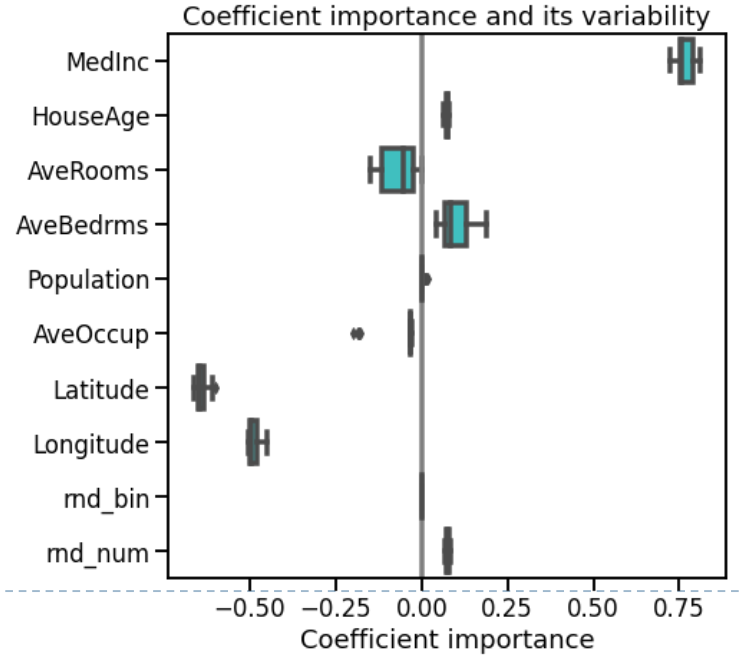




Select from model shrinkage

- Above methods have one obvious disadvantage: unlike subset selection, which will select models that involve just a subset of the variables, they will include all p predictors in the model!
- As with ridge regression, the *lasso* shrinks the coefficient estimates toward zero
  - However, in the case of the lasso, the l_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large. Hence, much like best subset selection, the lasso performs *variable selection*
  - We say that the lasso yields *sparse models*





Boruta – using concept similar to permutation test

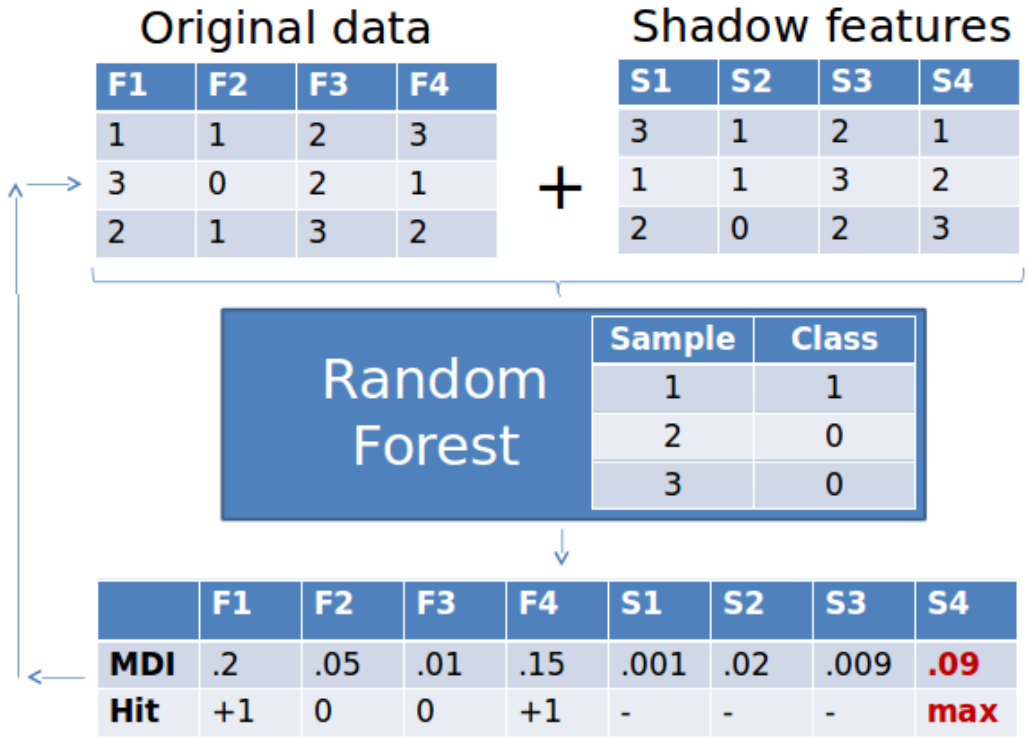
- Try to capture all the interesting features in your dataset with respect to target
  - Boruta is an all relevant feature selection method. This makes it suited for biomedical data analysis, where we regularly collect measurements of thousands of features and we have no clue where we should *cut off the threshold*
  - Boruta uses the binomial test to examine the following hypothesis

    H_0 : Feature importance of a feature is the same as the maximum of random features

    H_a : Feature importance of a feature is larger than the maximum of random features
  - As we do this for thousands of features we need to correct for multiple testing. The original method uses the rather conservative Bonferroni correction for this. Or you could set a threshold for it

Boruta – using concept similar to permutation test

1. First, we duplicate our dataset, and shuffle the values in each column, these are called *shadow features*
2. We train models on merged training data and shadow features and check for each of our real features if they have higher importance than the best of the shadow features
3. Then we remove feature that did not pass test or does not hit for a large number of iteration from the original data matrix and continue with another iteration



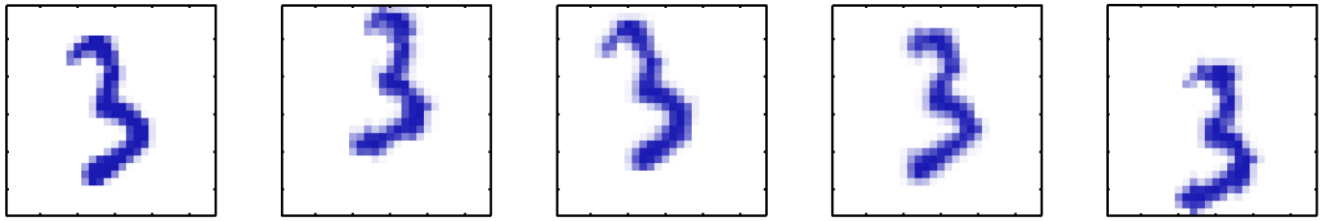
	F1	F2	F3	F4	S1	S2	S3	S4
MDI	.2	.05	.01	.15	.001	.02	.009	**.09**
Hit	+1	0	0	+1	-	-	-	**max**
Hit	+1	+1	0	0	-	**max**	-	-
Hit	+1	0	0	+1	-	-	-	**max**

SULOV from featurewiz

- SULOV is Searching for Uncorrelated List Of Variables
- Feature are selected to be far away from each other while still have high correlation to the target (Minimal Optimal algorithm)
  1. Find all pairs of highly correlated features that exceed threshold (e.g. 0.7)
  2. For each pair find their mutual information to the target variable and eliminate one with lower score from the pair
- Recursively select best features using the feature importance score from XGBoost

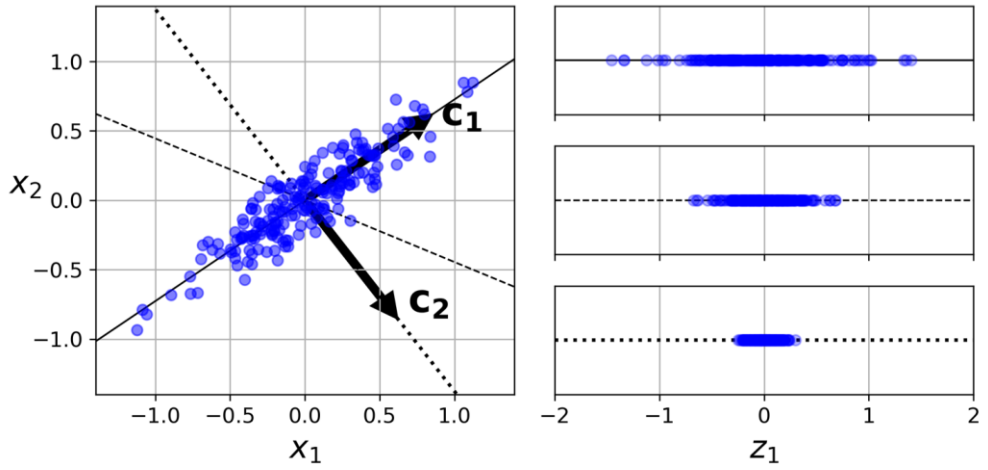
4. Feature extraction - Dimensional reduction by linear projection

- In most real-world problems, training instances are not spread out uniformly across all dimensions. Many features are almost constant, while others are highly correlated. As a result, all training instances actually lie within a much lower-dimensional *subspace* of the high-dimensional space
  - Dimensionality reduction tries to preserve various measures or structures in high dimensional space like distance, topology, density, etc
  - We would like to find a low-dimensional representation of the data that captures as much of the information as possible



Principal Components Analysis (PCA)

- Suppose that we wish to visualize n observations with measurements on a set of p features, $X_1, X_2, \ldots, X_p$, as part of an exploratory data analysis
  - We could do this by examining two-dimensional scatterplots of the data, each containing the n observations on two of the features. However, there are $\binom{p}{2}$ such scatterplots
  - Most likely, none of them will be informative since they each contain just a small fraction of the total information present in the data set!
  - PCA produces a *low-dimensional representation* of a dataset. It finds a sequence of linear combinations of the variables that have *maximal variance*



Interpretation of principal components

- The first principal component loading vector solves the optimization problem

$$\max_{\Phi_{11},\ldots,\Phi_{p1}} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\Phi_{j1}x_{ij}\right)^2 \text{ subject to } \sum_{j=1}^{p}\Phi_{j1}^2 = 1$$

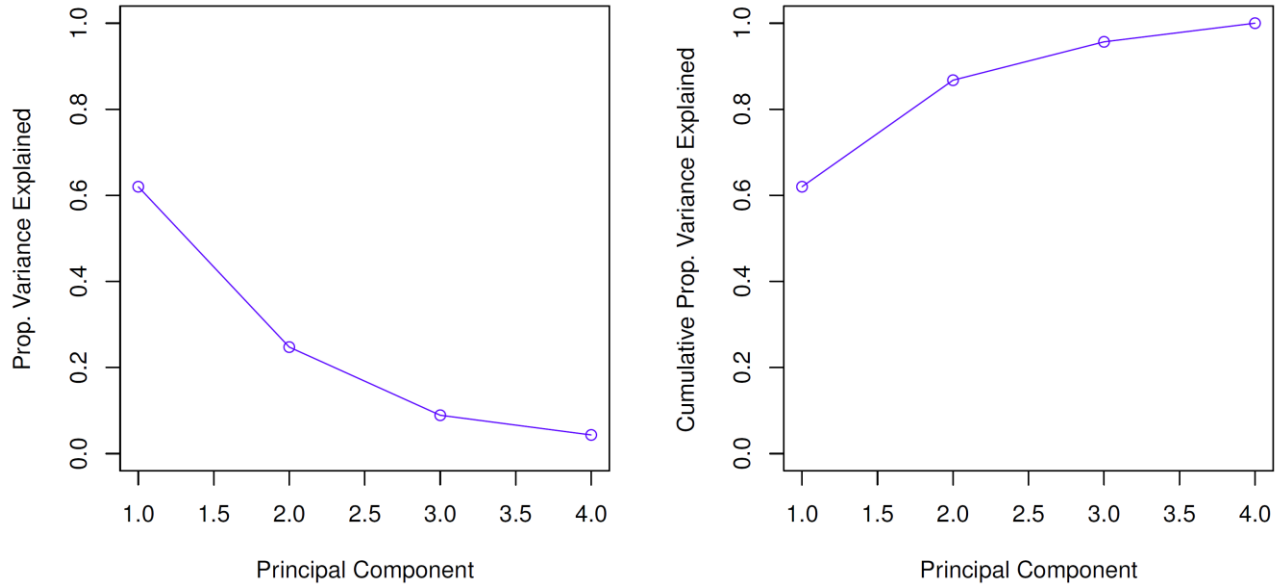
  - This problem can be solved via a *singular-value decomposition* of the data matrix X or the *eigenvalue decomposition* of the covariance matrix X
- The loading vector Φ_1 with elements $\Phi_{11}, \Phi_{21}, \ldots \Phi_{p1}$ defies a direction in feature space along which the data vary the most
- If we project the n data points $x_1, x_2, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$
- The second principal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance among all linear combinations that are uncorrelated with Z_1

Proportion Variance Explained

- The *proportion of variance explained (PVE)* of the mth principal component is given by the positive quantity between 0 and 1

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2} = \frac{\sum_{i=1}^{n}(\sum_{j=1}^{p} \Phi_{jm} x_{ij})^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}$$

- The PVEs sum to one. We sometimes display the cumulative PVEs

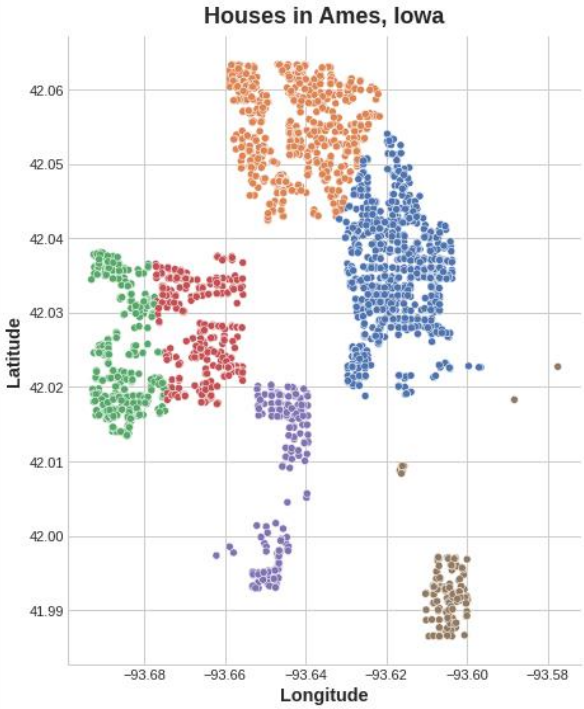


Cluster analysis

- Clustering refers to a very broad set of techniques for finding subgroups, or clusters, in a data set. We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other
  1. Once a dataset has been clustered, it is possible to measure each instance's *affinity* with each cluster which reduces to *k* dimensional space (*representation learning*)
  2. We can cluster observations on the basis of the features in order to identify subgroups among the observations, or we can cluster features on the basis of the observations in order to discover subgroups among the features (*feature clustering*/feature agglomeration)
  3. For semi-supervised learning: if you only have a few labels, you could perform clustering and propagate the labels to all the instances in the same cluster
  4. For *anomaly detection*: any instance that has a low affinity to all the clusters is likely to be an anomaly
- Taxonomy of clustering algorithms

Clustering as a preprocessing method

- Clustering can be an efficient approach to dimensionality reduction, or as a preprocessing step before a supervised learning algorithm
  - *Cluster labels* as a feature
    - The motivating idea for adding cluster labels is that the clusters will break up complicated relationships across features into simpler chunks. Our model can then just learn the simpler chunks one-by-one instead of having to learn the complicated whole all at once
  - *Distance to cluster* centers can also be a good feature
  - Another use case is for feature agglomeration by pooling function



Longitude	Latitude	Cluster	D_C0	…	D_C3
-93.619	42.054	3	1.25		5.24
-93.619	42.053	3	3.56		3.33
-93.638	42.060	1	5.21		0.89
-93.602	41.988	0	7.83		4.31

Clustering for semi-supervised learning

- Another use case for clustering is *in semi-supervised learning* when we have plenty of unlabeled instances and very few labeled instances
  - Just like active learning, we can first cluster all the data points and ask for the label of the images that are most close to the cluster center
    - Using these images to train classifiers may be much better than random instances
  - We can also propagate the labels to all the other instances in the same cluster. This is called *label propagation*

Short summary

- For unsupervised algorithms. To make the *affinity* concrete, we must define what it means for two or more observations to be similar or different
  - Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied
  - Supervised metrics are also possible, you can refer to similarity learning
  - Metric learning may also be helpful in this case
- Both clustering and PCA seek to simplify the data via a small number of summaries, but their mechanisms are different:
  - PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance
  - Clustering looks for homogeneous subgroups among the observations

Conclusions

- The principal advantage of selecting features within a data science pipeline is to reduce the time to train this pipeline and its time to predict, simplification of models to make them easier to interpret and avoid the curse of dimensionality
  - Univariate and subset selection
  - Select from the model using importance or using shrinkage
  - Dimension reduction/clustering
- Unsupervised learning is important for understanding the variation and grouping structure of a set of unlabeled data and can be a useful pre-processor for supervised learning
  - It is intrinsically more difficult than supervised learning because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy)

References

[1] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition Chapter 8,9

[2] https://inria.github.io/scikit-learn-mooc/python_scripts/dev_features_importance.html

[3] https://www.kaggle.com/learn/feature-engineering

[4] An Introduction to Statistical Learning, second edition

[5] Model selection 1, 2

[6] Unsupervised learning 1, 2

Appendix

Resources

- Feature selection
  - https://scikit-learn.org/stable/modules/feature_selection.html#
  - https://scikit-learn.org/stable/modules/permutation_importance.html
  - https://scikit-learn.org/stable/modules/unsupervised_reduction.html
  - https://github.com/feature-engine/feature_engine
  - https://github.com/AutoViML/featurewiz
  - https://github.com/jaswinder9051998/zoofs
- Boruta
  - https://danielhomola.com/feature%20selection/phd/borutapy-an-all-relevant-feature-selection-method/
- Core set selection
  - https://dcai.csail.mit.edu/lectures/growing-compressing-datasets/

Resources

- Clustering
  - https://developers.google.com/machine-learning/clustering
  - https://github.com/microsoft/ML-For-Beginners/blob/main/5-Clustering/1-Visualize/README.md#introduction-to-clustering
- Manifold learning
  - https://scikit-learn.org/stable/modules/manifold.html
- Spectral clustering
  - https://jlmelville.github.io/smallvis/spectral.html
- Mixture models
  - https://cs229.stanford.edu/syllabus.html
  - https://cs229.stanford.edu/notes2021fall/cs229-notes7b.pdf
  - Variational Bayesian Gaussian Mixture

Backward stepwise selection: details

Backward Stepwise Selection

1. Let M_p denote the full model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   a) Consider all k models that contain all but one of the predictors in M_k , for a total of $k-1$ predictors
   b) Choose the best among these k models, and call it M_{k-1}. Here best is defined as having the smallest RSS
3. Select a single best model from among $M_0, \ldots, M_p$ using cross-validated score, C_p (AIC), BIC, or adjusted R^2

More on backward stepwise selection

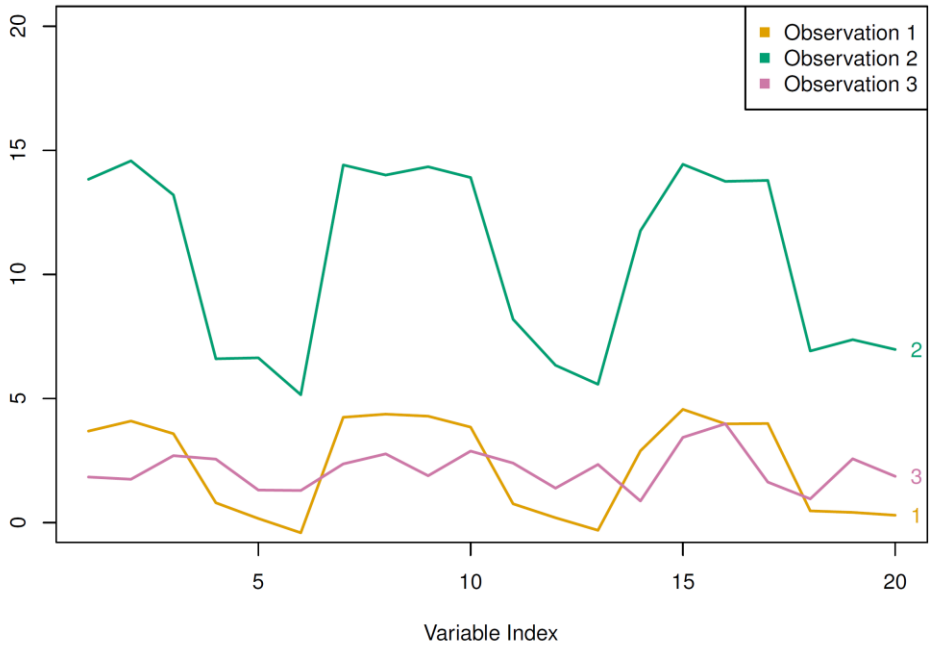
- Like forward stepwise selection, the backward selection approach searches through only $1 + p(p + 1)/2$ models, and so can be applied in settings where p is too large to apply best subset selection
- Like forward stepwise selection, backward stepwise selection is not guaranteed to yield the best model containing a subset of the p predictors
- Backward selection requires that the number of samples n is larger than the number of variables p (so that the full model can be fit). In contrast, forward stepwise can be used even when $n < p$, and so is the only viable subset method when p is very large

Good EDA clustering algorithm

1. **Don't be wrong**: If you are doing EDA you are trying to learn and gain intuitions about your data. In that case it is far better to get no result at all than a result that is wrong. This means a good EDA clustering algorithm needs to conservative in it's clustering
2. **Intuitive Parameters**: If you know little about your data it can be hard to determine what value or setting a parameter should have. This means parameters need to be intuitive enough that you can hopefully set them without having to know a lot about your data
3. **Stable Clusters**: If you run the algorithm twice with a different random initialization, you should expect to get roughly the same clusters back. If you are sampling your data, taking a different random sample shouldn't change much. If you vary the clustering algorithm parameters you want the clustering to change in a somewhat stable predictable fashion
4. **Performance**: You can sub-sample, but ultimately you need a clustering algorithm that can scale to large data sizes. A clustering algorithm isn't much use if you can only use it if you take such a small sub-sample that it is no longer representative of the data at large!

Choice of Dissimilarity Measure

- So far have used Euclidean distance
- An alternative is correlation-based distance which considers two observations to be similar if their features are highly correlated
  - Correlation-based distance focuses on the shapes of observation profiles rather than their magnitudes
  - More metrics

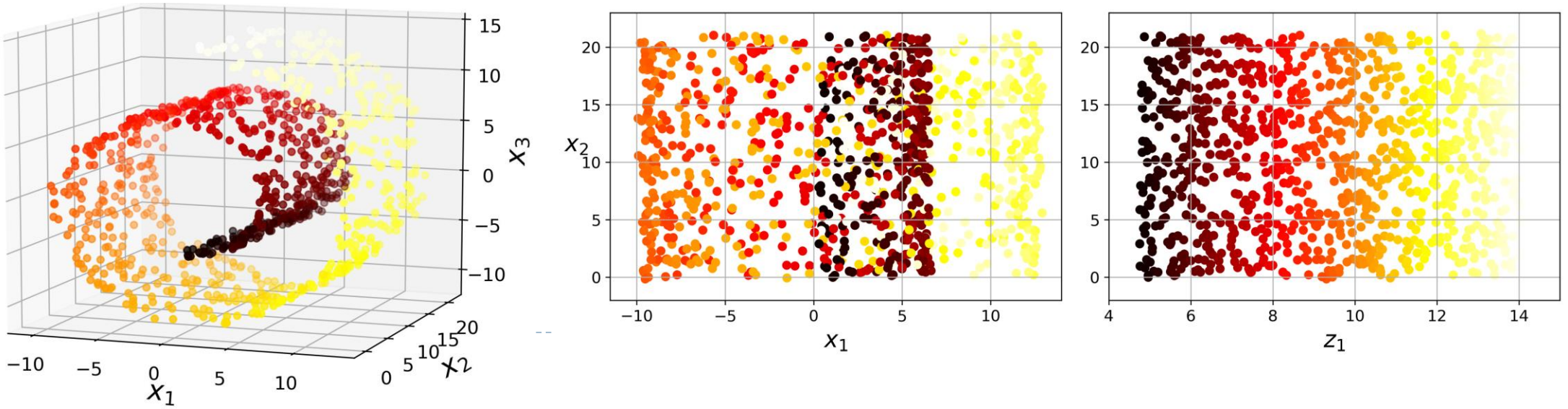


Choice of Dissimilarity Measure

- Consider an online retailer interested in clustering shoppers based on their past shopping histories
  - The data takes the form of a matrix where the rows are the shoppers and the columns are the items available for purchase; the elements of the data matrix indicate the number of times a given shopper has purchased a given item
  - If Euclidean distance is used, then shoppers who have bought very few items overall will be clustered together. This may not be desirable. On the other hand, if correlation-based distance is used, then shoppers with similar preferences will be clustered together

Manifold learning

- A *M*-dimensional manifold is part of a *p*-dimensional space (where $M < p$) that *locally resembles* an *M*-dimensional hyperplane
  - In the case of the Swiss roll, $M = 2$ and $p = 3$: it locally resembles a 2D plane, but it is bent and twisted in the third dimension
  - Simply projecting onto a plane would squash different layers of the Swiss roll
- Many dimensionality reduction algorithms work by modeling the manifold on which the training instances lie; this is called *manifold learning*



t-SNE (t-distributed Stochastic Neighbor Embedding)

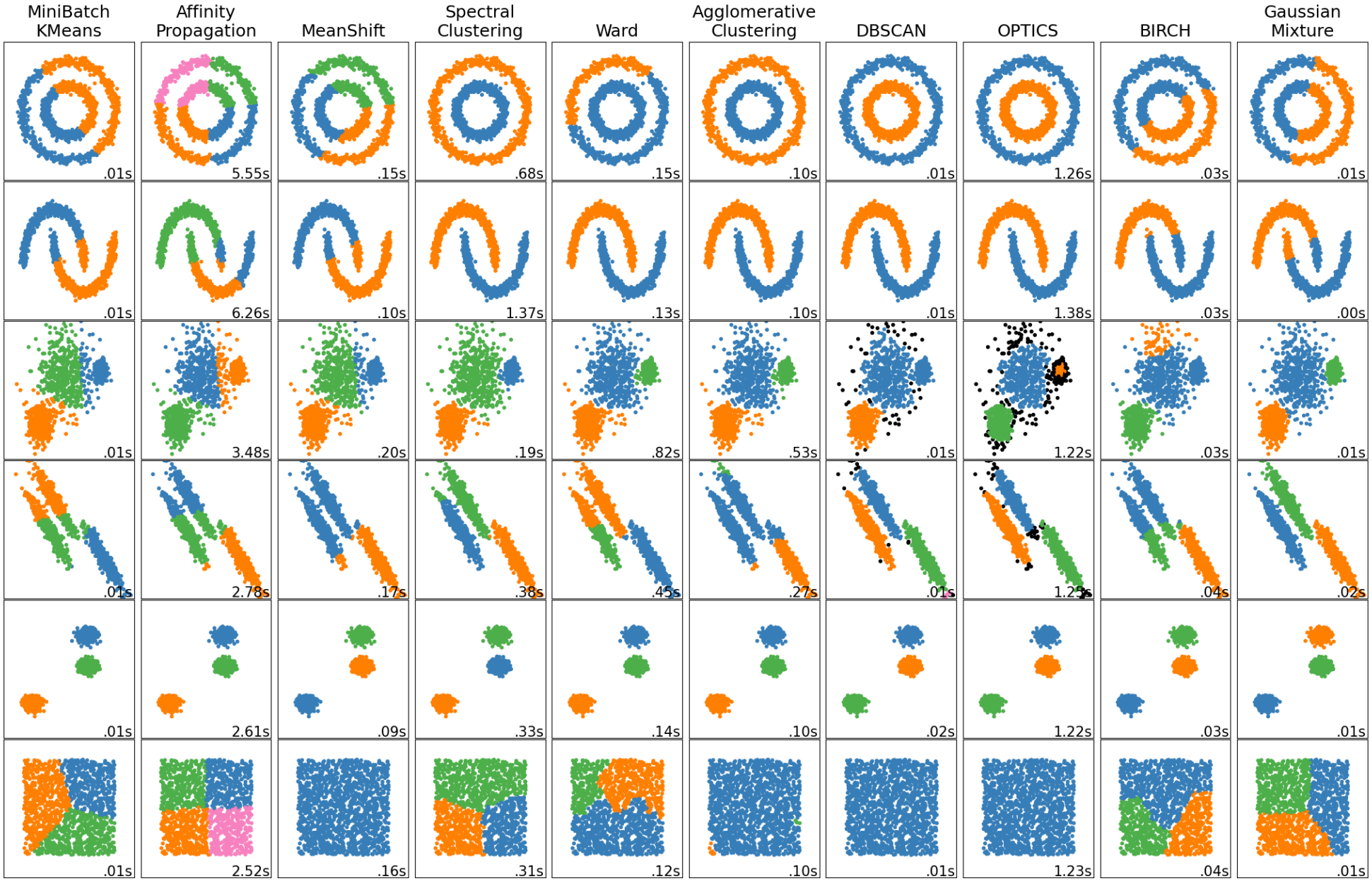
- t-SNE is a manifold learning method that is specialized for visualization and EDA
  1. It is a statistical method for visualizing high-dimensional data by giving each data point a location in a two or three-dimensional map
  2. It converts affinities of data to probabilities and tends to preserve topology that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points. It is particularly sensitive to local structure and can reveal the structure at many scales or data lie on several manifolds on a single map
  3. However, the global structure is not explicitly preserved and we should use it with caution for downstream analysis

t-Stochastic Neighbor Embedding

9

Caveats of t-SNE

- t-SNE does not work well for general dimensionality problems where the embedded dimension is greater than 2D or 3D
  - O(Mn^2) computational complexity. But can be reduced to $O(Mn \log n)$ using Barnes-Hut t-SNE which only works for the target dimensionality smaller than 3 and dense dataset. The flt-tsne or UMAP can be used to give linear scalability
- There are more and more theoretical guarantees for t-SNE. See here and here
  - t-SNE is generally very good at capturing local structure and best tool for visualization
  - Noise may appear to have some structure and clusters may not discover by t-SNE
  - Generally, we do not use t-SNE for clustering purposes
  - See here for more experiments
- UMAP, on the other hand, can be used as a preprocessing method



https://scikit-learn.org/stable/modules/clustering.html