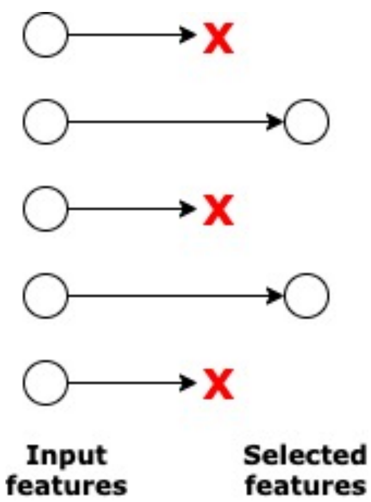
Dimensionality Reduction: Feature Selection

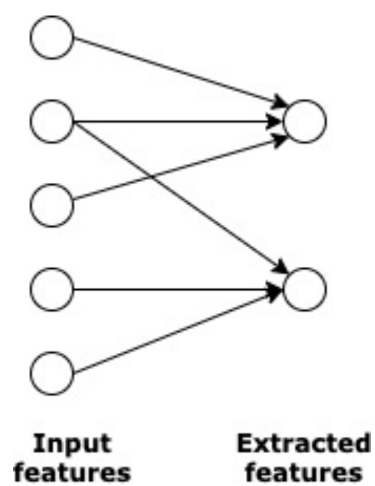
Tozammel Hossain

Data Science & Analytics
University of Missouri

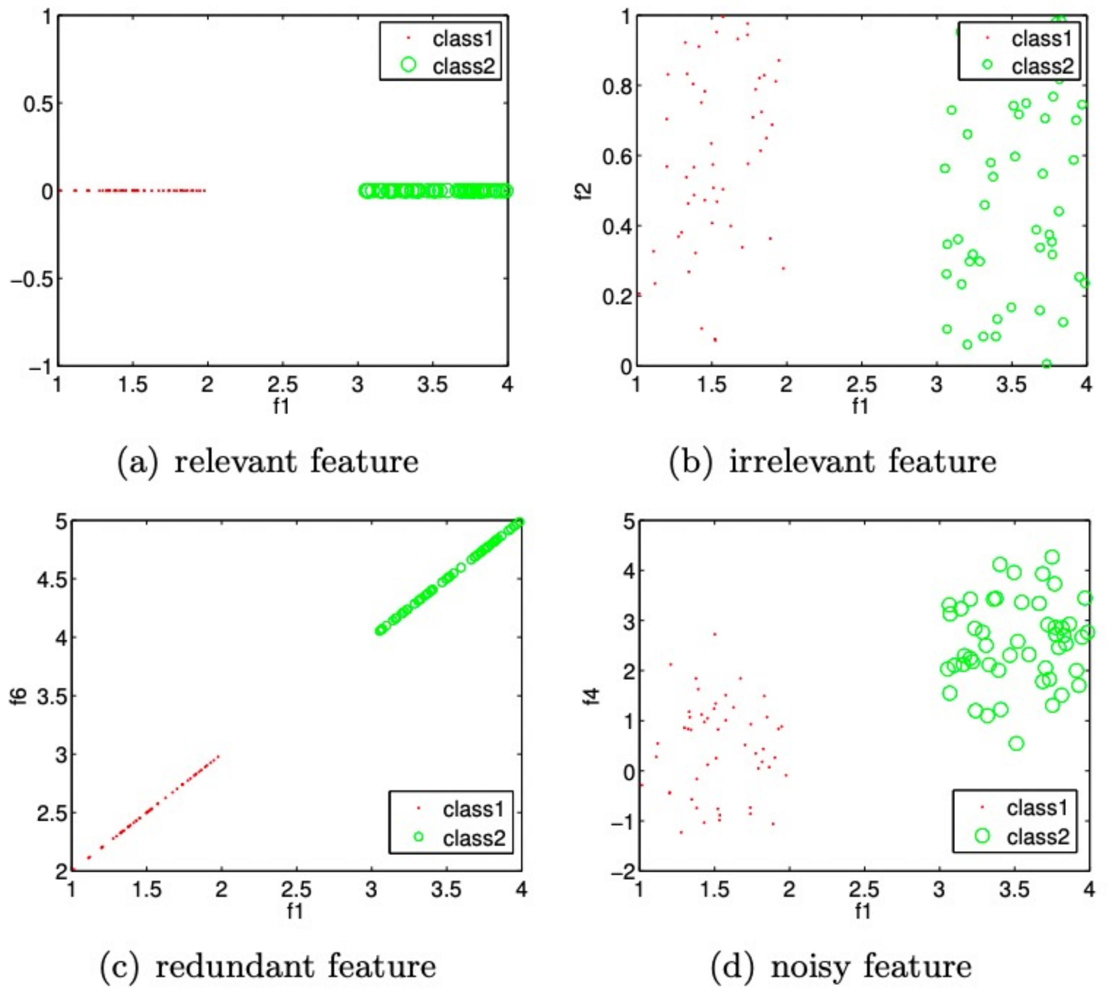
Dimensionality Reduction

- **Removes irrelevant, redundant, and noisy features**
  - Could be used with supervised and unsupervised settings
  - Learns character/structure of data
  - Used as a preprocessing step
- **Dimensionality Reduction**
  - Feature Selection
    - aka subset selection
    - finding a subset of features that give most information
  - Feature Extraction
    - finding a new subset of features that are combinations of original dimensions
    - A type of feature engineering





Irrelevant, noisy, and redundant features



(a) relevant feature (b) irrelevant feature

(c) redundant feature (d) noisy feature

source: https://www.cse.msu.edu/~tangjili/publication/Feature_Selection.pdf

Why Feature Selection?

- **Ideal Case: the learning algorithm (e.g., classifier or regressor) should be able to use whichever features are necessary**
  - Doesn't work as feature selection is not inherent to many models
- **Feature selection reduces memory and computation**
  - Complexity (time and space) depends on the number of input dimension
- **Save the cost of extracting irrelevant features**

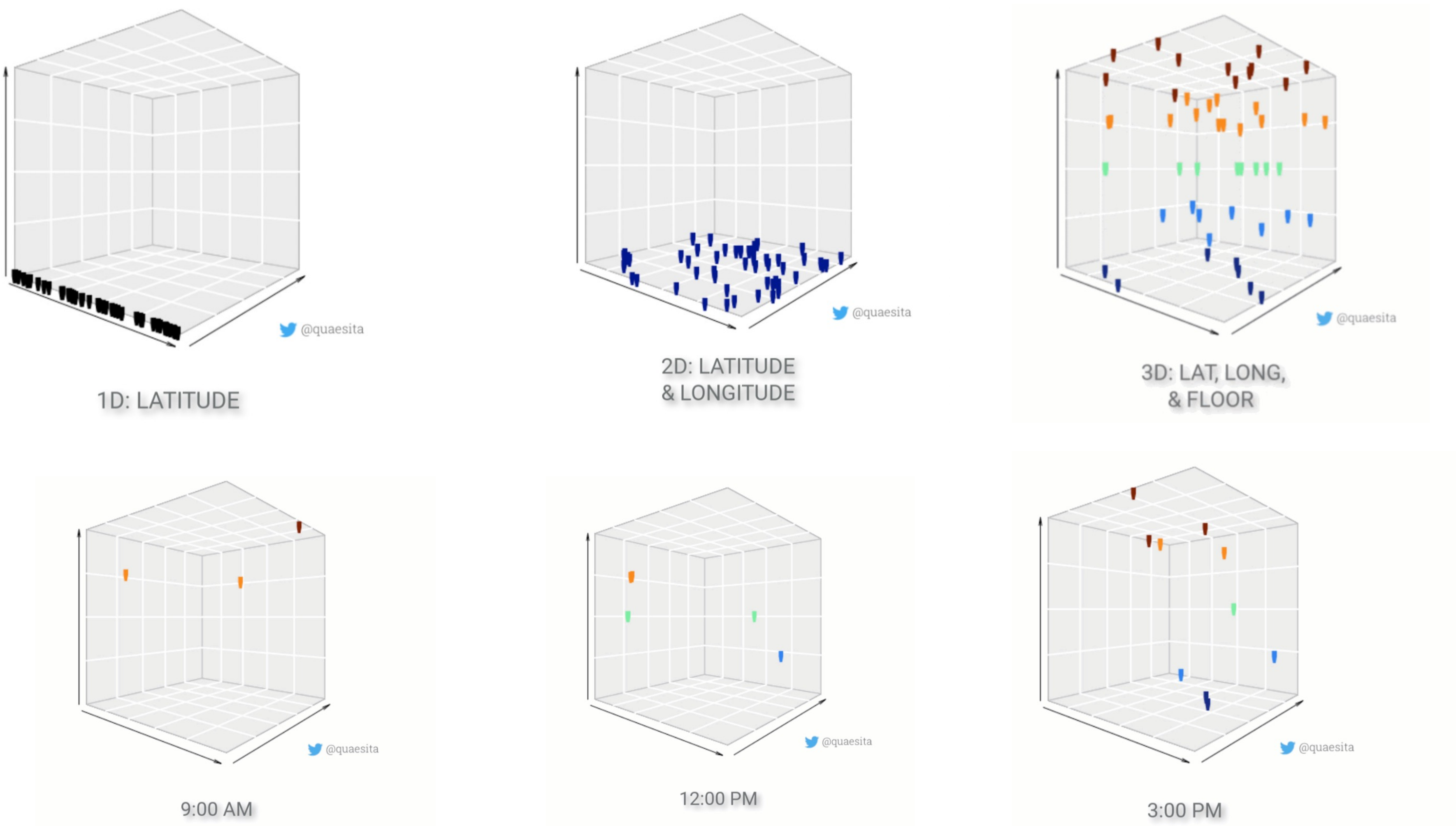
Why Feature Selection?

- **Implements Occam's razor: the simplest explanation is usually the right one**
- **If data could be explained with fewer features, we get a better idea about the process that generates the data**
  - Facilitate knowledge extraction
- **Easier to visualize data with fewer features**

Curse of Dimensionality

- **With the increase of features**
  - Need more instances for reliable estimation of model; otherwise, overfitting problem occurs
  - Each data point resides into its own cluster
    - Data spreads out in the space
    - E.g., 8 features; each takes 10 values/bins => 10^8 = 100 M possible data points
  - Increase a dimension requires an exponentially-growing amount of data points to overcome spread out of data points

Curse of Dimensionality



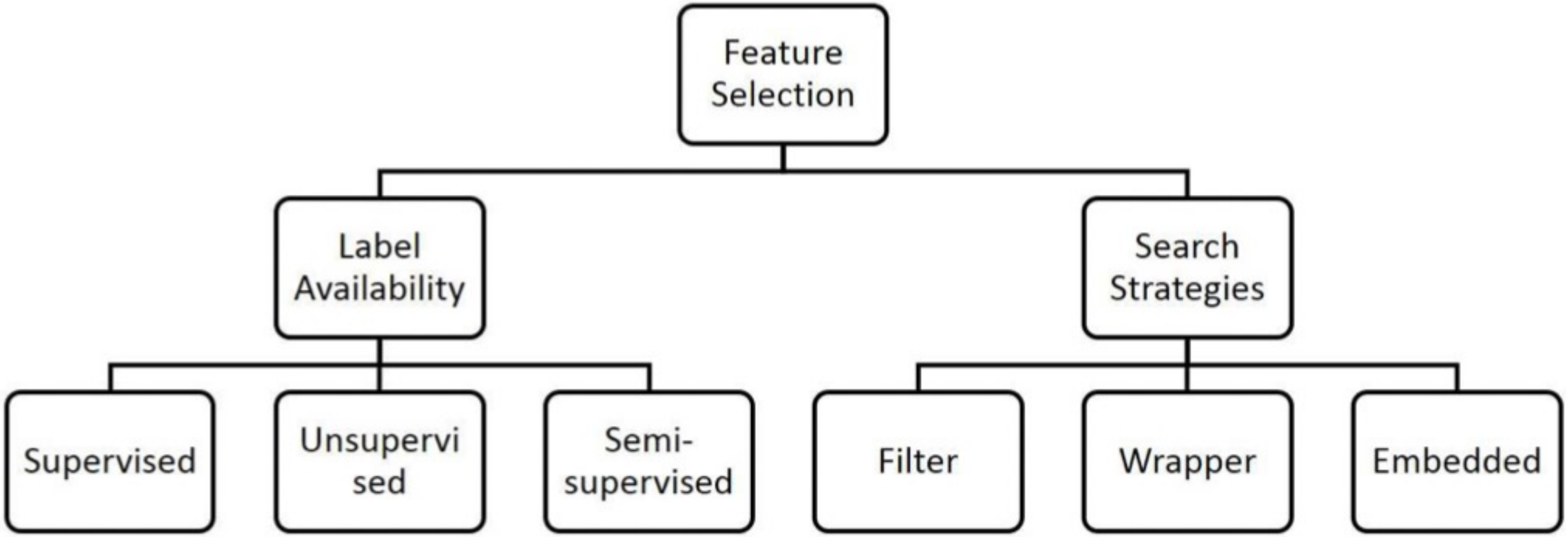
4D: Lat, Long, Floor & Time

https://towardsdatascience.com/the-curse-of-dimensionality-minus-the-curse-of-jargon-520da109fc87

Feature Selection Strategy

- **Goal: finding the "best" subset of the set of features**
- **Naive Strategy:**
  - for n features, there are 2^n possible subsets
  - take each subset and see its performance on the validation set
  - infeasible if n is large
- **Two components**
  - Selection criteria
  - Search methods

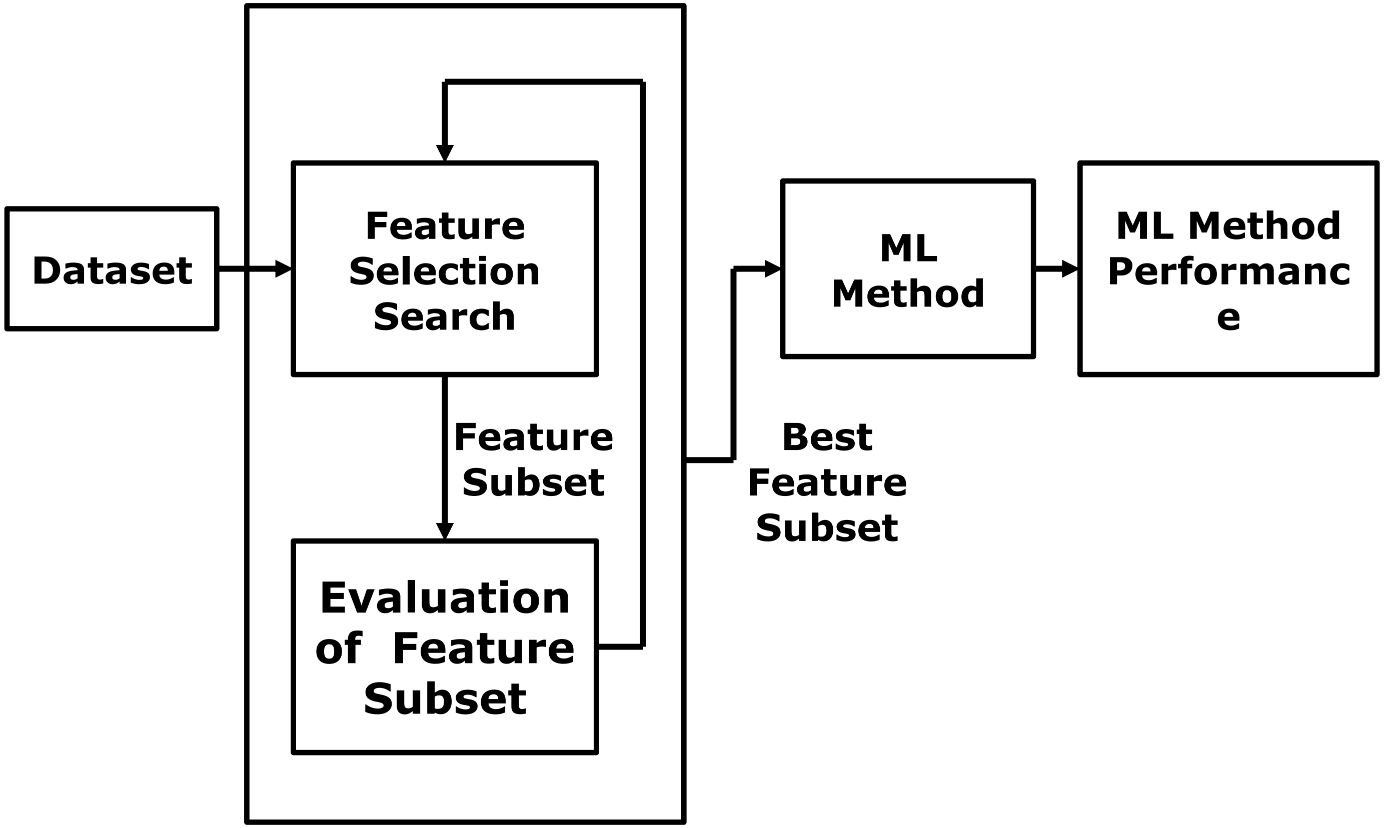
Feature Selection Types



Filter Methods

- **Aka variable ranking method; univariate feature selection**
- **Runs independent of the learning algorithm**
  - classification/regression/clustering
- **Select the features using the ”character” of the data**
  - identify relationship between each feature and target

Filter Methods



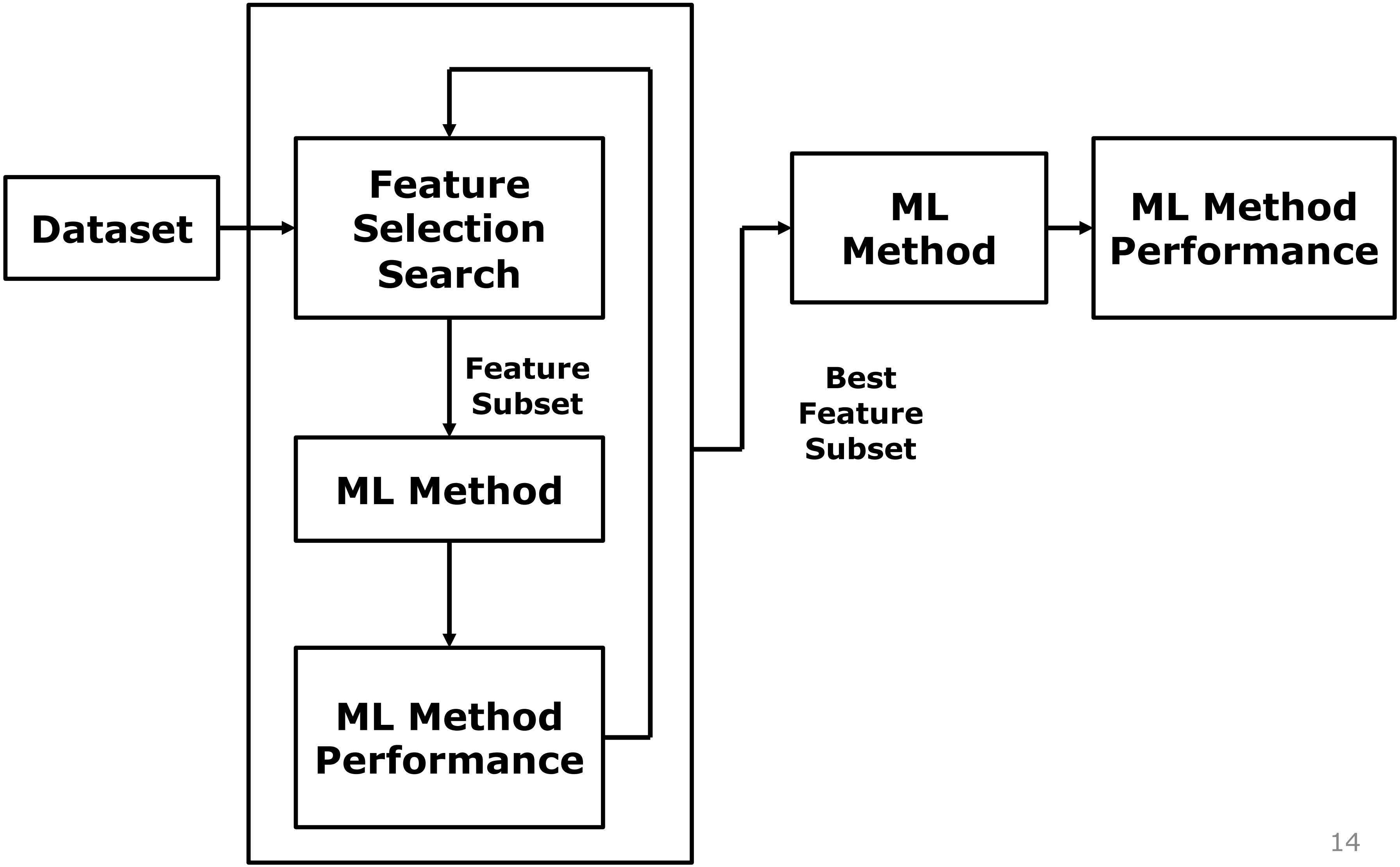
Filter Methods

- **Pros**
  - Independent of the learning method
    - Bias of the learning algorithm does not interact with the bias inherent in the feature selection method
  - Fast to compute
    - No need to build a learning model
- **Cons**
  - May miss features that are relevant for the target learning algorithm
  - Each feature is considered individually
    - A feature could be informative when combined with another feature

Wrapper Methods

- **Feature selection is “wrapped around” a learning algorithm**
- **Utilize the estimated accuracy from the learning algorithm as the measure of goodness**
- **Bias of the learning algorithm interacts with the bias inherent in the feature selection method**

Wrapper Methods



Wrapper Methods

- **Pros**
  - Takes insights from the learning algorithm
    - Predictive accuracy estimates are used for scoring a subset
  - May perform better given a learning algorithm

- **Cons**
  - Computationally expensive
    - Naïve approach is exponential
    - Remedy: different search strategy
      - hill-climbing, best-first, branch-and-bound, and genetic algorithms

Embedded method

- **Feature selection is embedded in the process of model construction**
- **A tradeoff between filter and wrapper methods**
- **No additional strategy is required for selecting the features**
- **Examples:**
  - Ridge Regression, LASSO, Elasticnet, Logistic Regression, Decision Tree

Embedded Method

- **Model provides some measure of importance for each of the features**
  - sklearn:

Attributes:	**classes_ : *ndarray of shape (n_classes,) or list of ndarray*** The classes labels (single output problem), or a list of arrays of class labels (multi-output problem). **`feature_importances_` : *ndarray of shape (n_features,)*** Return the feature importances.

Embedded methods

- **Pros**
  - Combines the power of two methods
    - filter models and wrapper models
  - Computationally less expensive compared to wrapper method
    - Learning model is executed once
  - Includes the interaction with the learning model
- **Cons**
  - Relatively slow compared to filter method

Ranking Features

- **Statistical Measures**
  - Classifiction
    - Pearson χ^2 test
    - ANOVA F-value
  - Pearson correlation (r^2) coefficient
    - Regression
- **Information Theoretic Measure**
  - Mutual Information
    - Applicable to both classification and regression

Filter Methods: Pearson's χ^2 test

- **Measures dependence between two variables**
  - Feature is categorical/count, target is categorical
  - Applicable to classification
  - sklearn: chi2

FS: ANOVA F-value

- **Measures dependence between two variables**
  - Feature is continuous and target is discrete
  - The F-value scores
    - group the numerical feature by the target
    - check whether the means for each group are significantly different
  - sklearn: **f_classif**

FS: Filter Methods

- **Pearson's correlation (r^2)**
  - Measures **(linear)** dependence between two variables
  - Feature is continuous and target is continuous
  - sklearn: **f_regression**

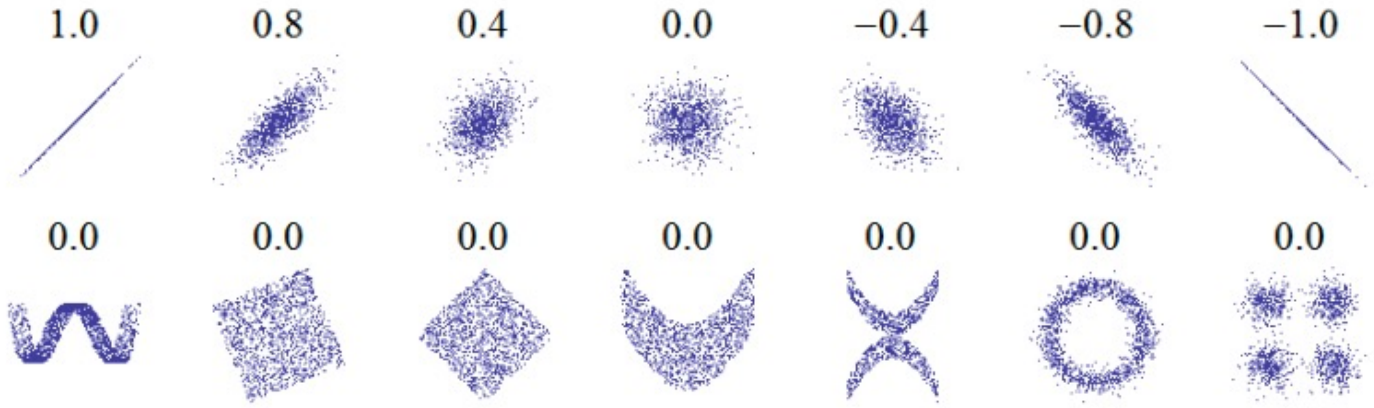
FS: Filter Methods

- **Mutual Information**
  - Reduction of uncertainty in a variable given the other one
  - Defined in terms of entropy
    - Entropy measures uncertainty in a variable
    - $\boldsymbol{M(C, Xi) = H(C) - H(C|Xi)}$
  - MI(X,Y) = 0 → Independent
  - MI(X,Y) > 0 → Dependency
- **Q: Any other usage of this measure?**

Linear vs Non-linear Dependency

Measure of *linear* dependence between r.v.'s X and Y.

$$\rho(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{\mathrm{E}[XY] - \mathrm{E}[X]\cdot\mathrm{E}[Y]}{\sigma_x \cdot \sigma_Y}.$$



How to select given ranking?

1.13.2. Univariate feature selection

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the `transform` method:

- `SelectKBest` removes all but the k highest scoring features
- `SelectPercentile` removes all but a user-specified highest scoring percentage of features
- using common univariate statistical tests for each feature: false positive rate `SelectFpr`, false discovery rate `SelectFdr`, or family wise error `SelectFwe`.
- `GenericUnivariateSelect` allows to perform univariate feature selection with a configurable strategy. This allows to select the best univariate selection strategy with hyper-parameter search estimator.

Baseline Feature Selection Method

`VarianceThreshold` is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by

$$\mathrm{Var}[X] = p(1 - p)$$

so we can select using the threshold `.8 * (1 - .8)`:

FS: Wrapper Methods

- **Forward selection**
  - Start with no features and at each step add one that decreases the error the most
  - Continue until any further addition does not decrease the error (or decrease it only slightly)
- **Backward elimination**
  - Start with all variables and at each step remove one that decrease the error the most
  - Continue until any further removal increases the error significantly
  - Recursive Feature Elimination

sklearn Feature Selection

`feature_selection.GenericUnivariateSelect`([...])	Univariate feature selector with configurable strategy.
`feature_selection.SelectPercentile`([...])	Select features according to a percentile of the highest scores.
`feature_selection.SelectKBest`([score_func, k])	Select features according to the k highest scores.
`feature_selection.SelectFpr`([score_func, alpha])	Filter: Select the pvalues below alpha based on a FPR test.
`feature_selection.SelectFdr`([score_func, alpha])	Filter: Select the p-values for an estimated false discovery rate
`feature_selection.SelectFromModel`(estimator, *)	Meta-transformer for selecting features based on importance weights.
`feature_selection.SelectFwe`([score_func, alpha])	Filter: Select the p-values corresponding to Family-wise error rate
`feature_selection.SequentialFeatureSelector`(...)	Transformer that performs Sequential Feature Selection.
`feature_selection.RFE`(estimator, *[, ...])	Feature ranking with recursive feature elimination.
`feature_selection.RFECV`(estimator, *[, ...])	Feature ranking with recursive feature elimination and cross-validated selection of the best number of features.
`feature_selection.VarianceThreshold`([threshold])	Feature selector that removes all low-variance features.

`feature_selection.chi2`(X, y)	Compute chi-squared stats between each non-negative feature and class.
`feature_selection.f_classif`(X, y)	Compute the ANOVA F-value for the provided sample.
`feature_selection.f_regression`(X, y, *[, center])	Univariate linear regression tests.
`feature_selection.mutual_info_classif`(X, y, *)	Estimate mutual information for a discrete target variable.
`feature_selection.mutual_info_regression`(X, y, *)	Estimate mutual information for a continuous target variable.