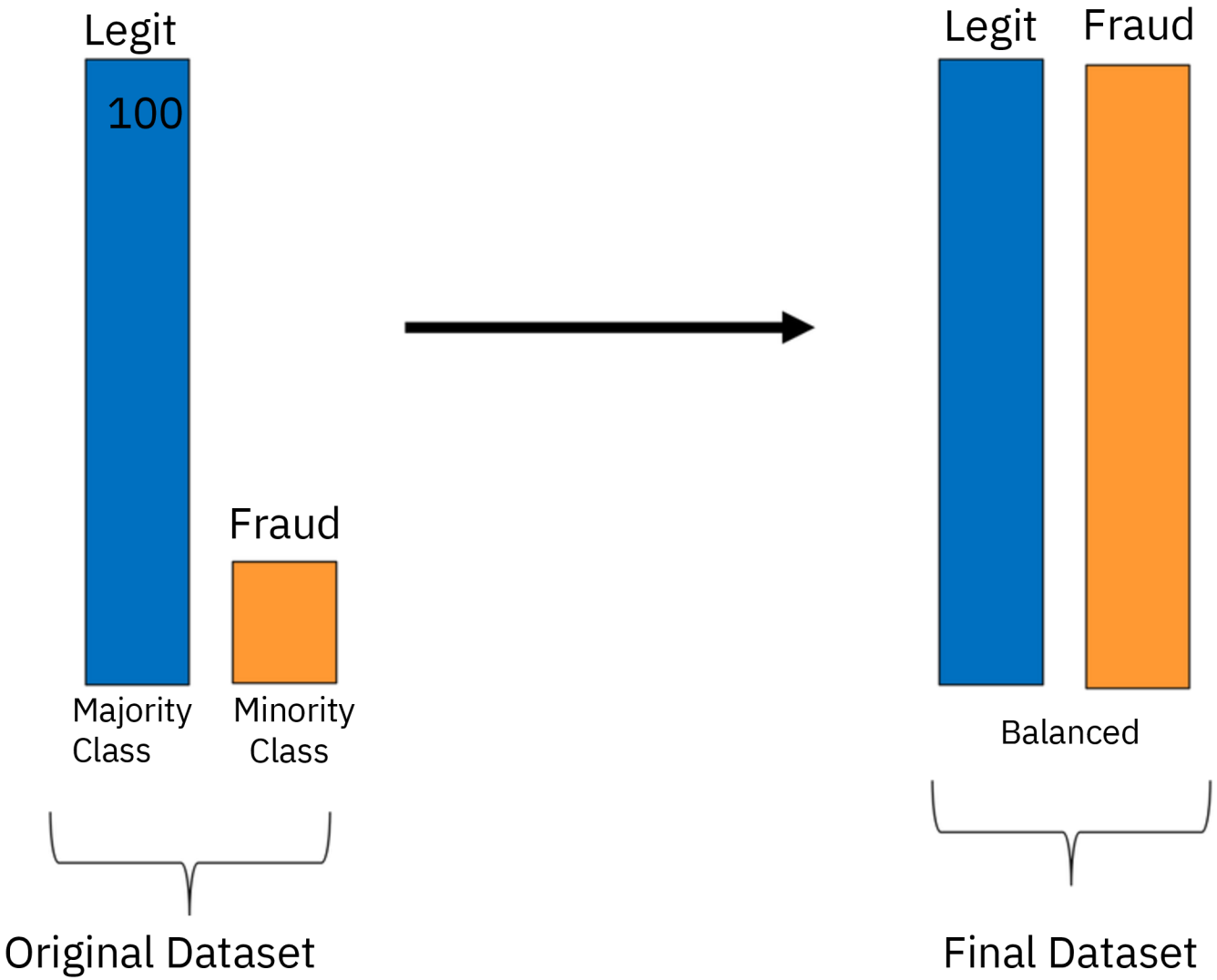
DATA PREPROCESSING A STRATEGY FOR MODEL OPTIMIZATION

Handling Class Imbalance in ML

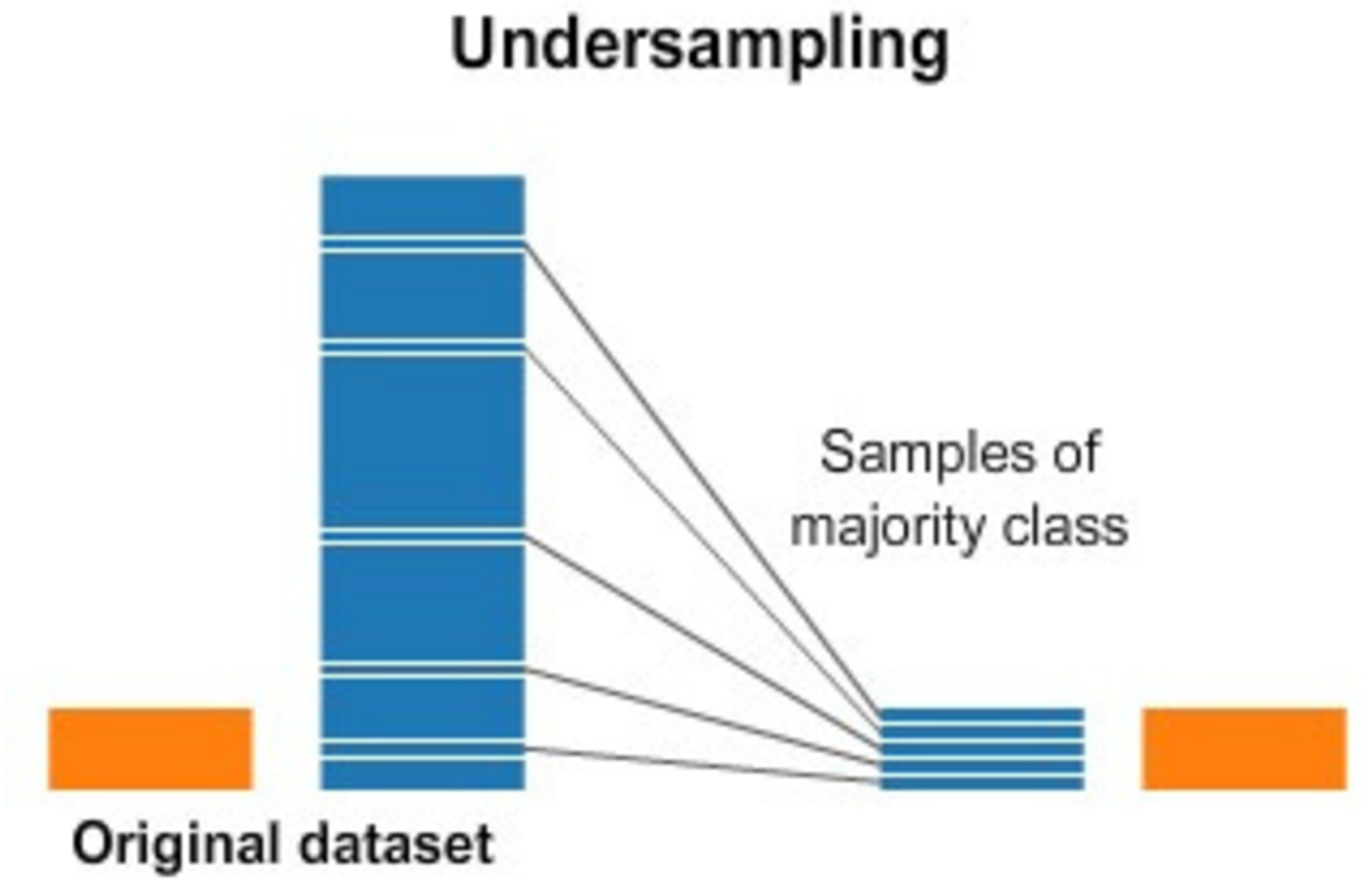
Class Imbalance Techniques



Some techniques to combat imbalance and avoid classification bias towards the majority class:

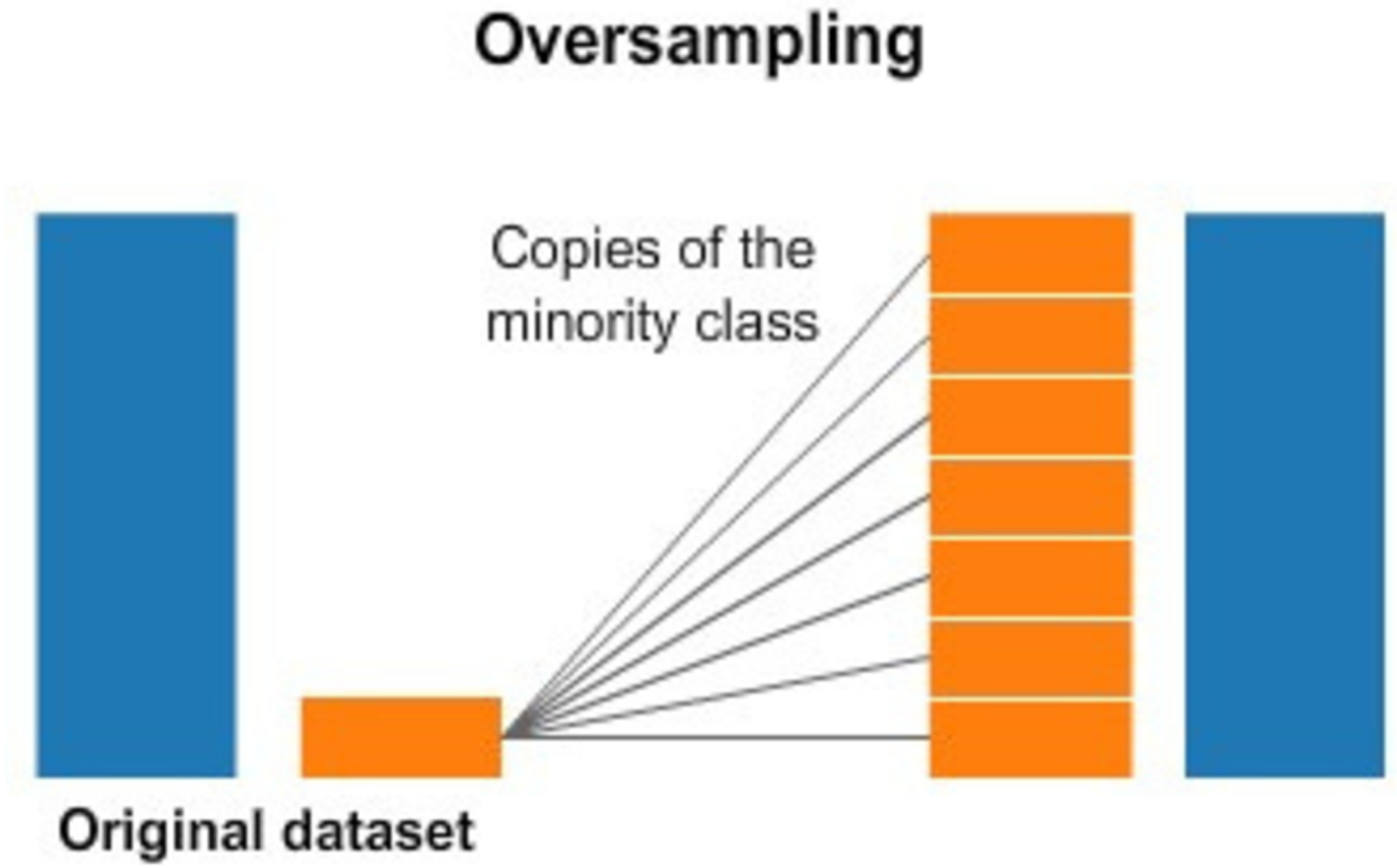
1) UndersamplingMajority Class
2) Oversampling Minority Class
3) Generate Synthetic Data
4) Balanced Class Weight

Undersampling



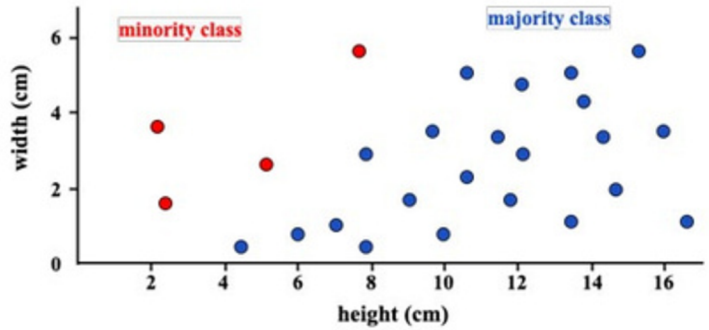
Pull a random sample from
the Majority Class

Oversampling

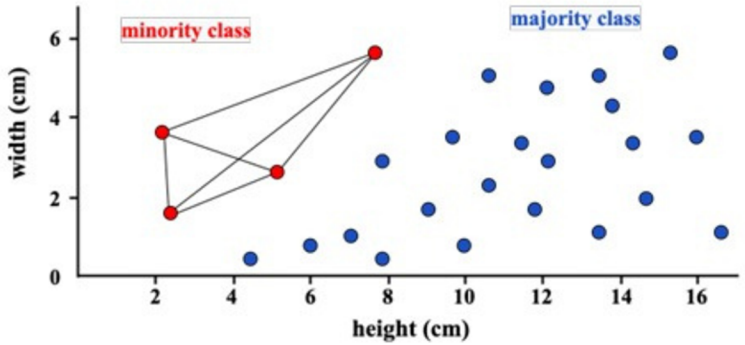


Duplication of the Minority Class

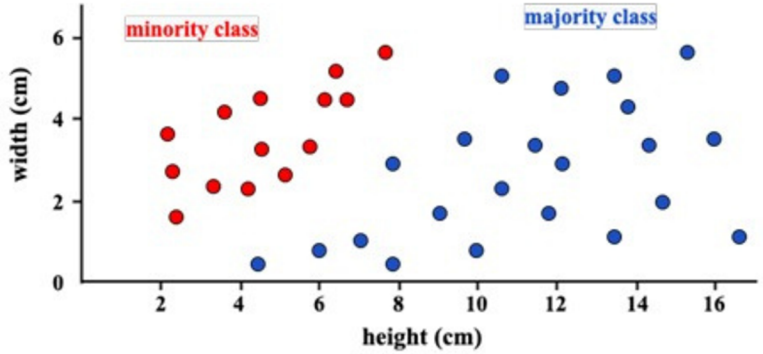
SMOTE (Synthetic Minority Oversampling Technique)



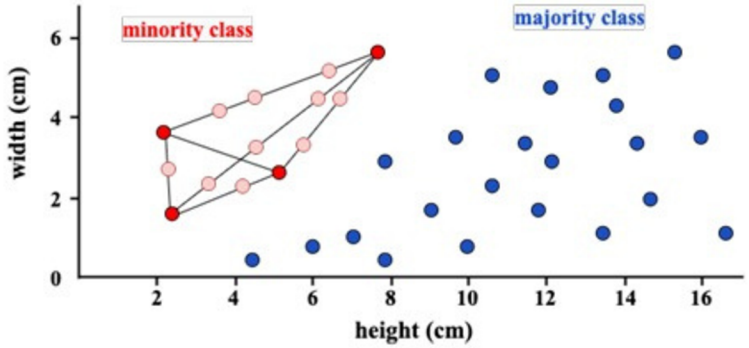
(a) Dataset with class imbalance



(b) Lines between each pair of points in minority class

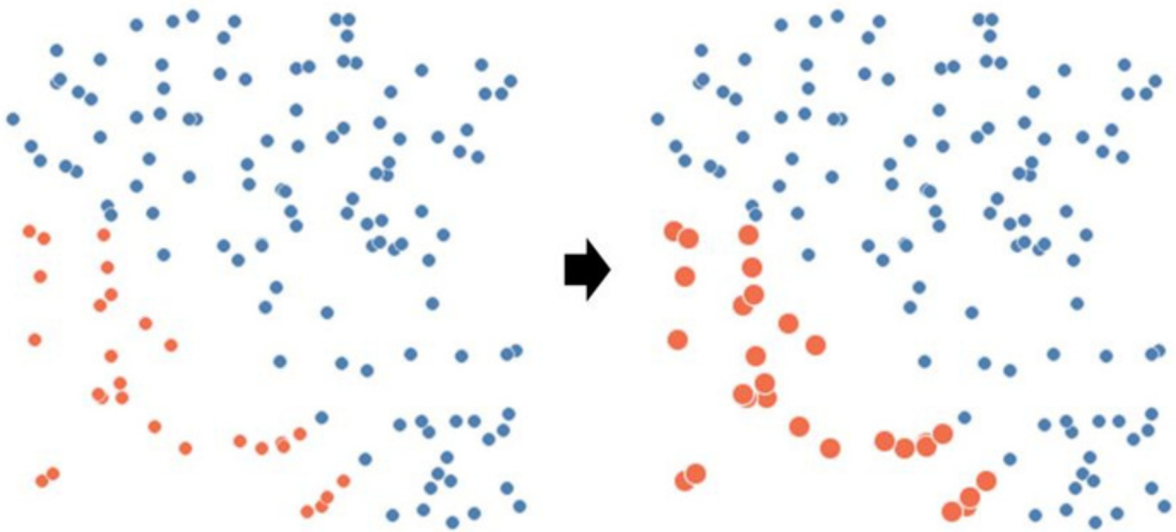


(d) Synthetic dataset with balanced classes



(c) Interpolation of data points

Class Weights



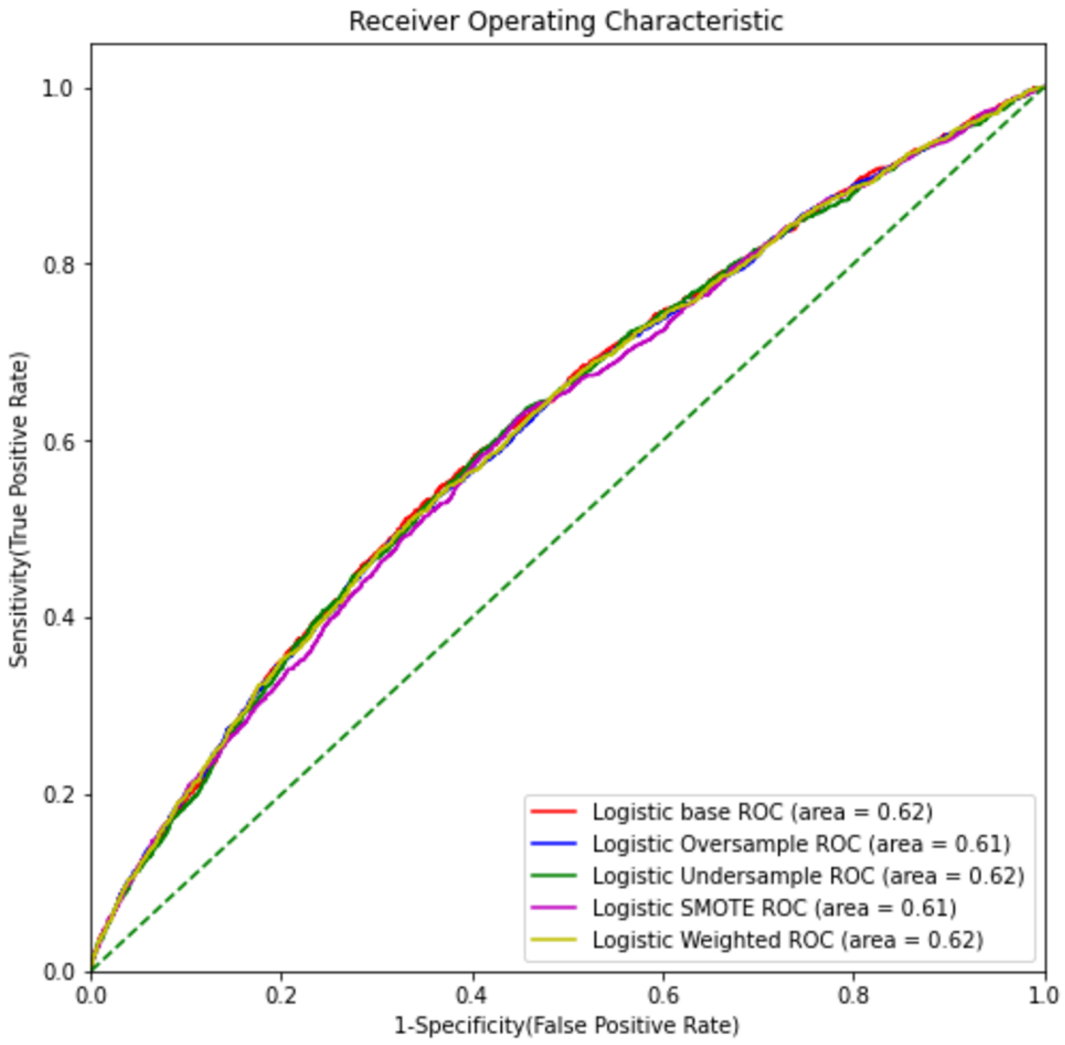
Assign weights to each class so that the weighted sum of observations is equal for all classes. For two classes where n is the number of observations and w is the weight:

$$n_{minority} \times w_{minority} = n_{majority} \times w_{majority}$$

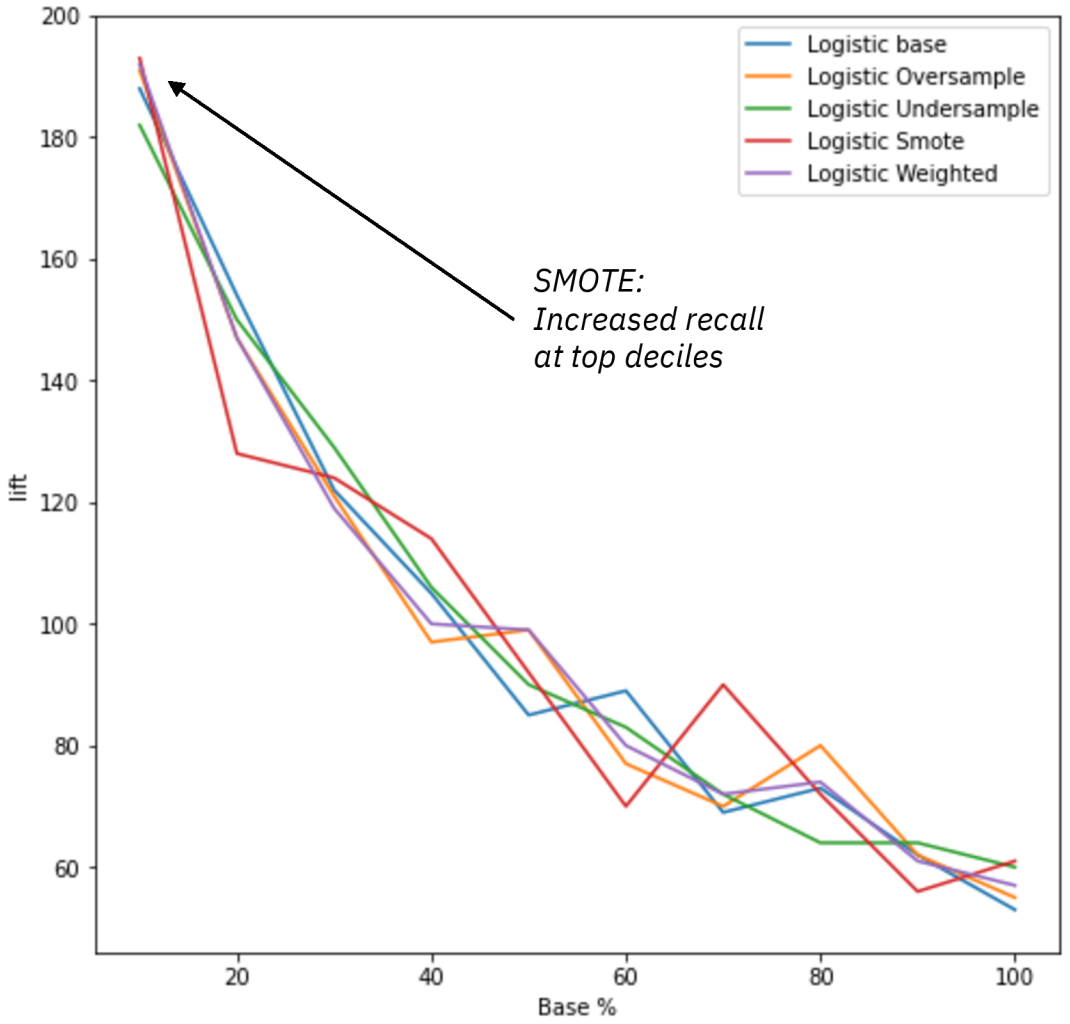
During training, the error is multiplied by the weight of the point. Without weights set, the model treats each point as equally important.

NB: <u>All</u> the records are assigned weight (no sampling)

ROC Evaluation



LIFT Evaluation



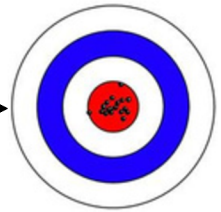
Feature Selection For ML

The ‘What’ and ‘Why’ of Feature Selection

Feature selection is the process of isolating the most consistent, non-redundant, and relevant features to use in model construction

Decreases Over-Fitting
Reduced chances of making decisions based on noise

Reduces Training Time
Less data means faster algorithms



Improves Model Accuracy
Reduced variance leads to higher precision

Improved Model Interpretability
Simpler models are more explainable

Feature Selection Techniques

- Filter Methods
- Wrapper Methods
- Embedded Methods
- Hybrid Methods

Filter Methods

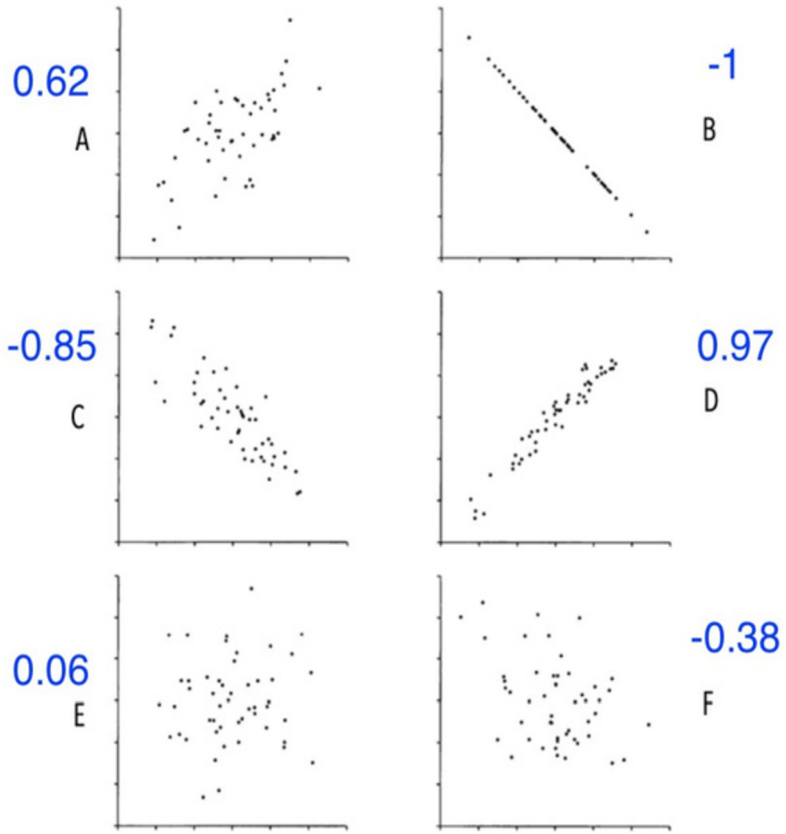
In the Filter methods, the features are selected based on statistical measures.

Filter methods are **independentof the learning algorithm** or the evaluation metric. Thus, the extracted features are "generic, having incorporated few assumptions"

1. Correlation Coefficient

(Identify features having a high correlation with the target variable and uncorrelated with each other)

Step 1: Calculate correlation coefficient of each feature with the target
Step 2: Rank order based on the absolute value of the correlation coefficient
Step 3: Establish a cut-off point and remove features below the threshold (or keep the top N features)



Reading: **Filter Methods for Feature Selection in Supervised Machine Learning Applications --Review and Benchmark**
Hopf, Reifenrath

2. Chi-Sq Test

A chi-square test is used in statistics to test the independence of two events. Given the data of two variables (e.g., Churn vs Gender), we can get observed count O and expected count E. Chi-Square measures how expected count E and observed count O deviates each other.

Contingency Table

Observed

	Churned	Not Churned	
Male	2	68	70
Female	1812		30
	20	80	10
			0

Expected

	Churned	Not Churned	
Male	1456		70
Femal	624		30
e	2080		10
			0

The Formula for Chi Square Is

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

c = degrees of freedom

O = observed value(s)

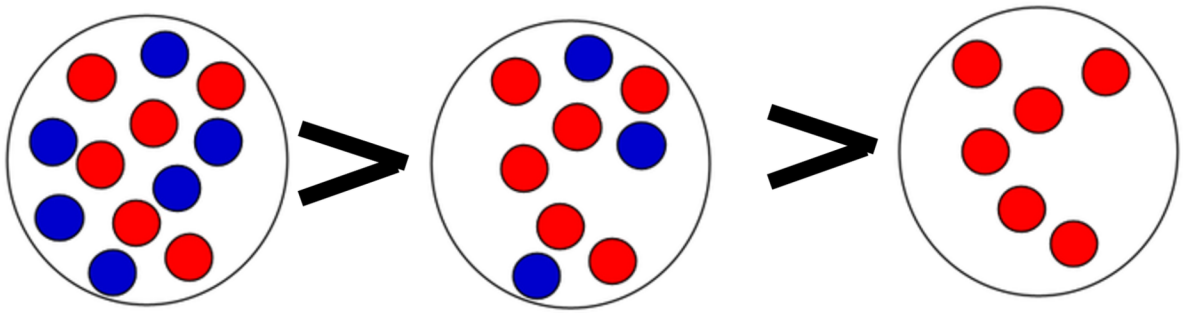
E = expected value(s)

When two features are independent, the observed count is close to the expected count, thus we will have smaller Chi-Square value. High Chi-Square value indicates that the hypothesis of independence is incorrect.

Mutual Information

Entropy

Entropy is a measure of disorder or impurity



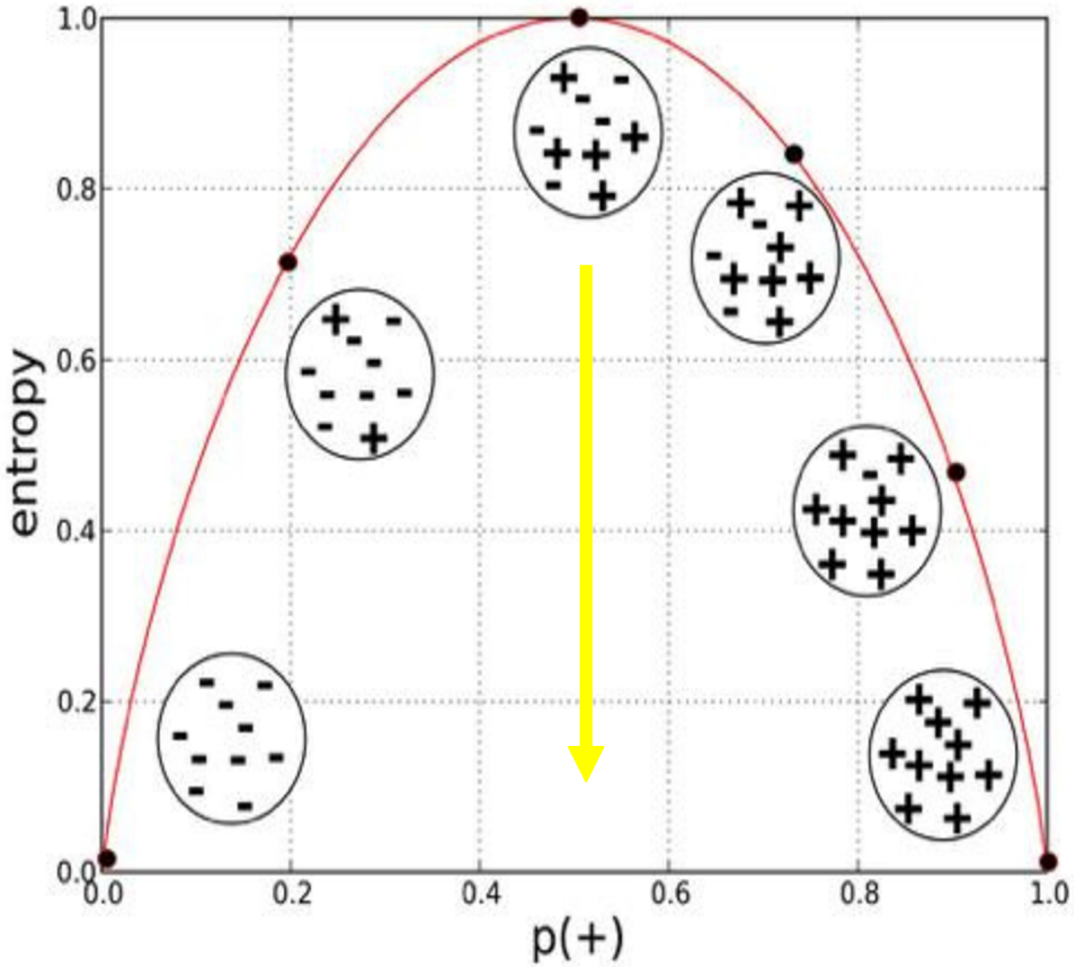
$P_{red} = 0.5$
$P_{blue} = 0.5$

$P_{red} = 1$
$P_{blue} = 0$

H(X1) =1

H(X3) =0

Information Gain



Entropy Calculation:

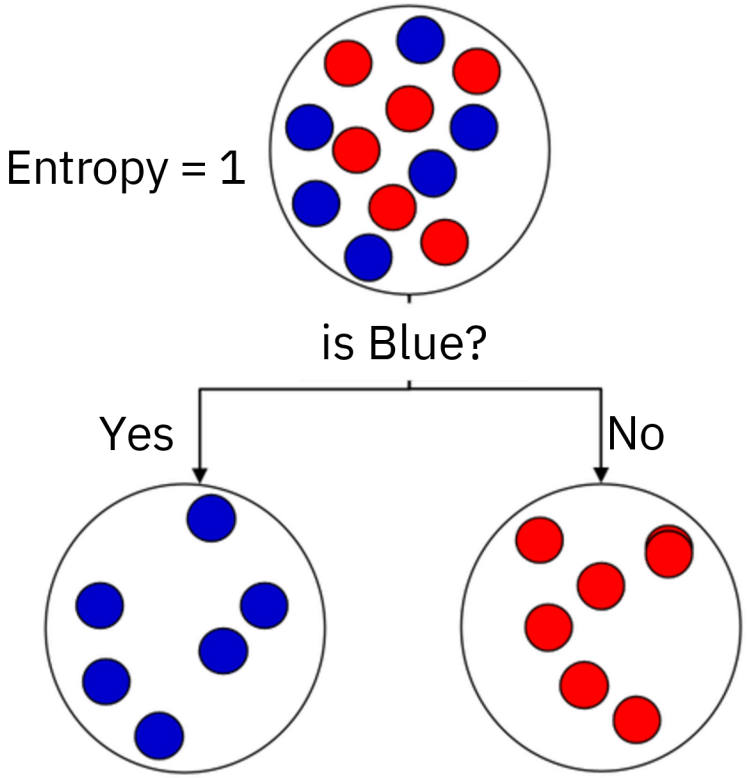
$$H(X) = -[(p * \log p) + (q * \log q)]$$

where

pi= Probability of Y = 1 i.e.probability of success of the event

qi= Probability of Y = 0 i.e.probability of failure of the event

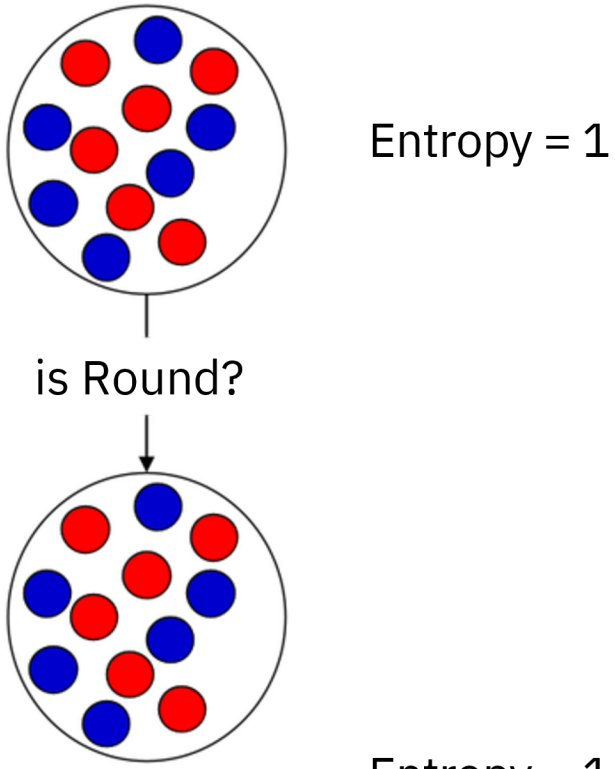
3. Mutual Information



Entropy = 0 (calculated by weighted average of the two nodes)

Information gain = 1 -0 = 1

Large Information Gain means the feature is powerful



Information gain = 1 -1 = 0

No information gain means the feature is useless

Mutual information quantifies the amount of information one can obtain from one random variable about another.

$\mathbf{I(X\,;\,Y)} = H(X) - H(X|Y)$

Independent features have

$I(X\,;\,Y) = 0$

(keep features with low mutual information)

4. Variance Threshold

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

A Univariate Approach (examine statistics of a single feature)

Compute the variance of each feature and remove features below a specified threshold

Assume that features with a higher variance may contain more useful information

	X1	X2
	1	1
	7	1
	4	1
	8	1
	9	1
	3	1
Mean	5.33	1

Var(X1) =8.22

Var(X2)=0

X2 has no value for modeling

Scikit-Learn Implementation : **VarianceThreshold**

Recap: Filter Method Examples:

- Correlation Coefficient
- Chi-Square
- Mutual Information
- Variance Threshold

Lot's more Filter Methods...

`sklearn.feature_selection`: Feature Selection

The `sklearn.feature_selection` module implements feature selection algorithms. It currently includes univariate filter selection methods and the recursive feature elimination algorithm.

User guide: See the Feature selection section for further details.

`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, *[, ...])	Recursive feature elimination with cross-validation to select 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, *[, ...])	Univariate linear regression tests returning F-statistic and p-values.
`feature_selection.r_regression`(X, y, *[, ...])	Compute Pearson's r for each features and the target.
`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.

https://scikit-learn.org/stable/modules/classes.html?highlight=feature%20selection#module-sklearn.feature_selection

SIDEBAR.....

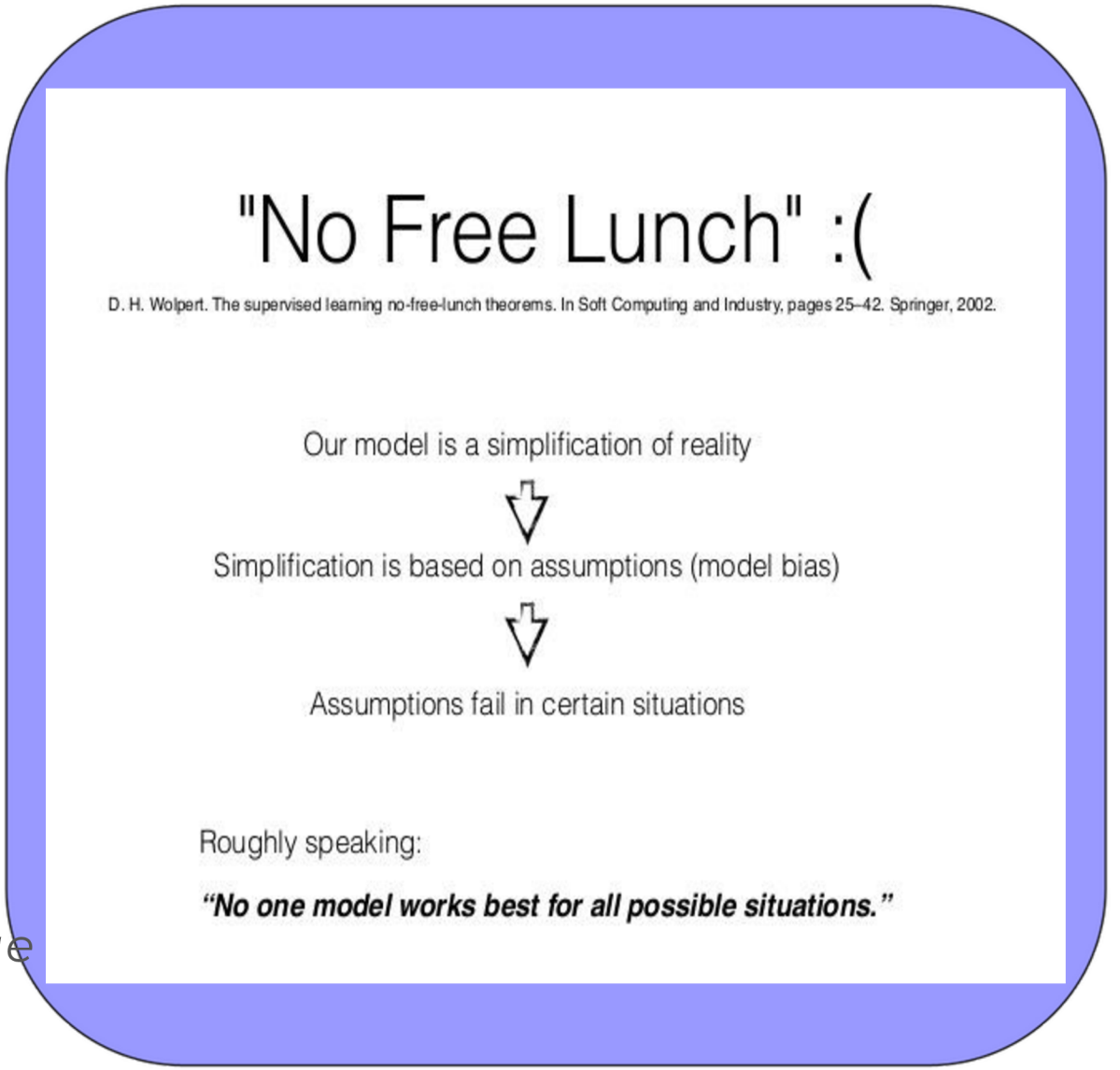
No Free Lunch Theorems (Wolpert and Macready 1997)

If one algorithm performs better than another algorithm on one class of problems, then it will perform worse on another class of problems

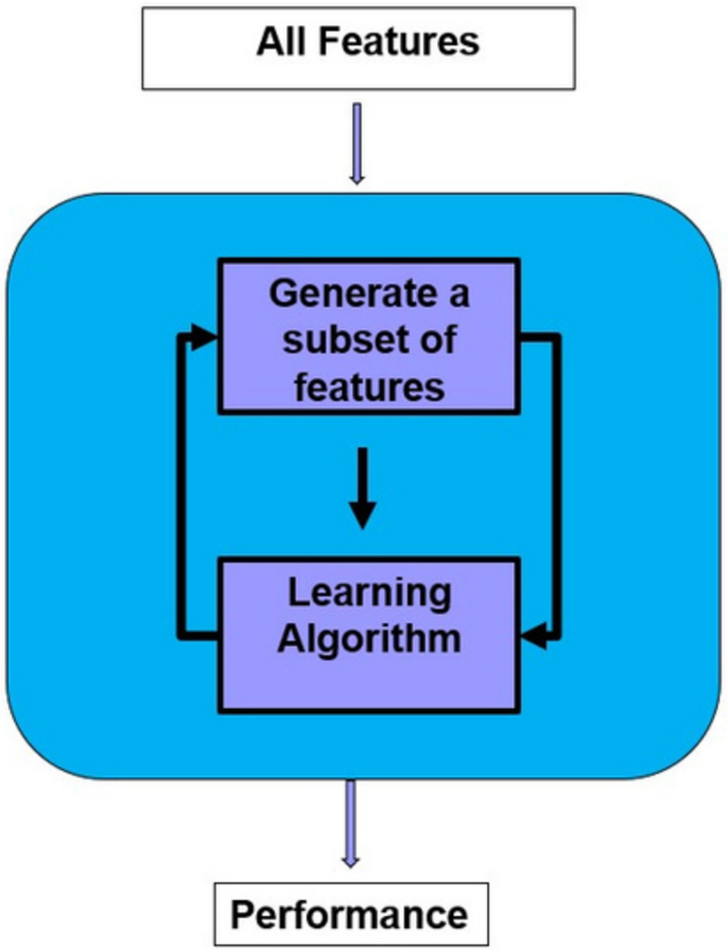
There is provably no single best optimization algorithm or machine learning algorithm.

all optimization algorithms perform equally well when their performance is averaged across all possible problems.

The NFL stated that within certain constraints, over the space of all possible problems, every optimization technique will perform as well as every other one on average (including Random Search)



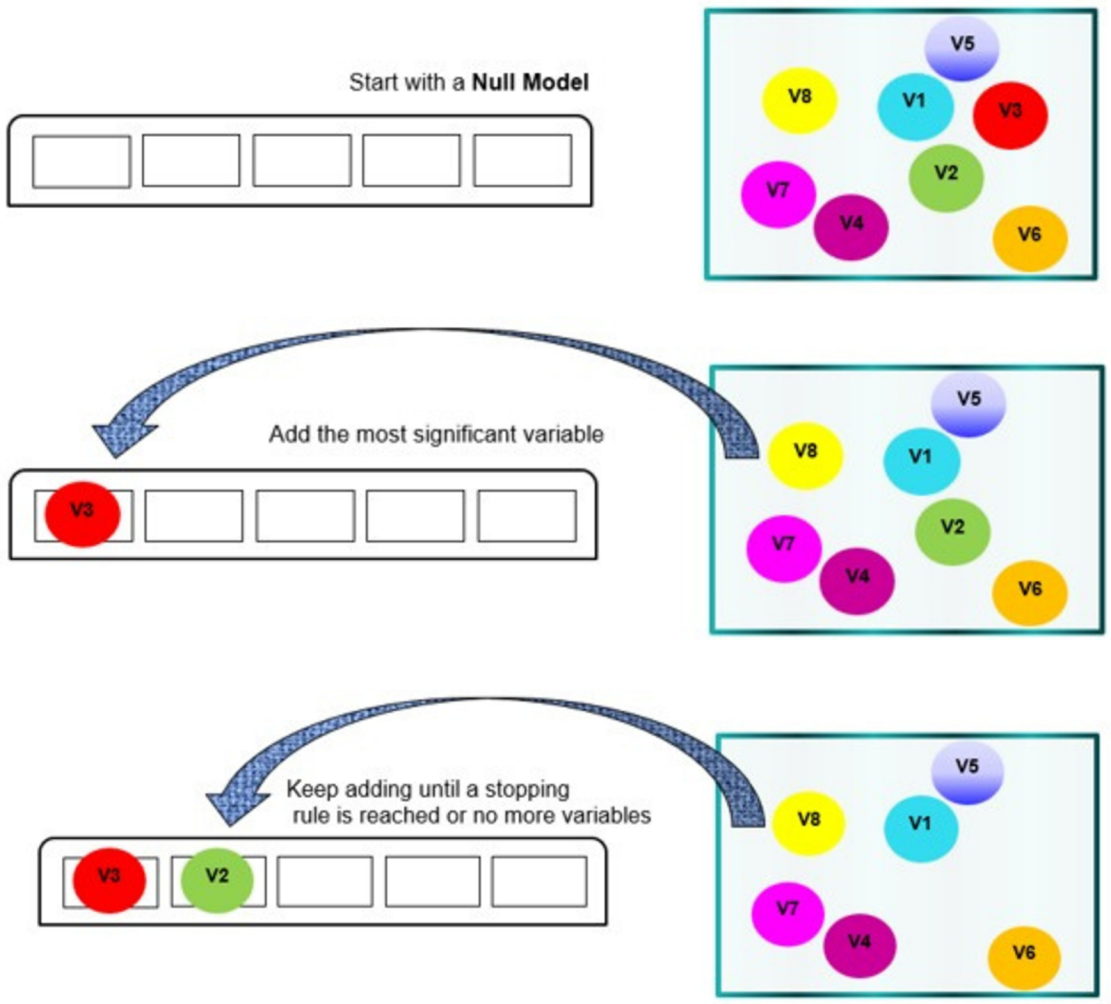
Wrapper Methods



These methods evaluate features onspecific algorithms to find the best setof features. Examples are:

- **Sequential Feature Selection**
  - **Sequential Forward Selection**
  - **Sequential Backward Selection**
  - **Sequential Floating Methods**
- **Recursive Feature elimination**
- **Permutation Importance**

1a. Sequential Forward Selection (SFS)



An iterative method wherein we start a NULL model then add the best performing variable against the target.

Next, we select another variable that gives the best performance **in combination** with the first selected variable.
This process continues until a preset criterion is achieved or we run out of variables

1. How do we determine 'significance'?

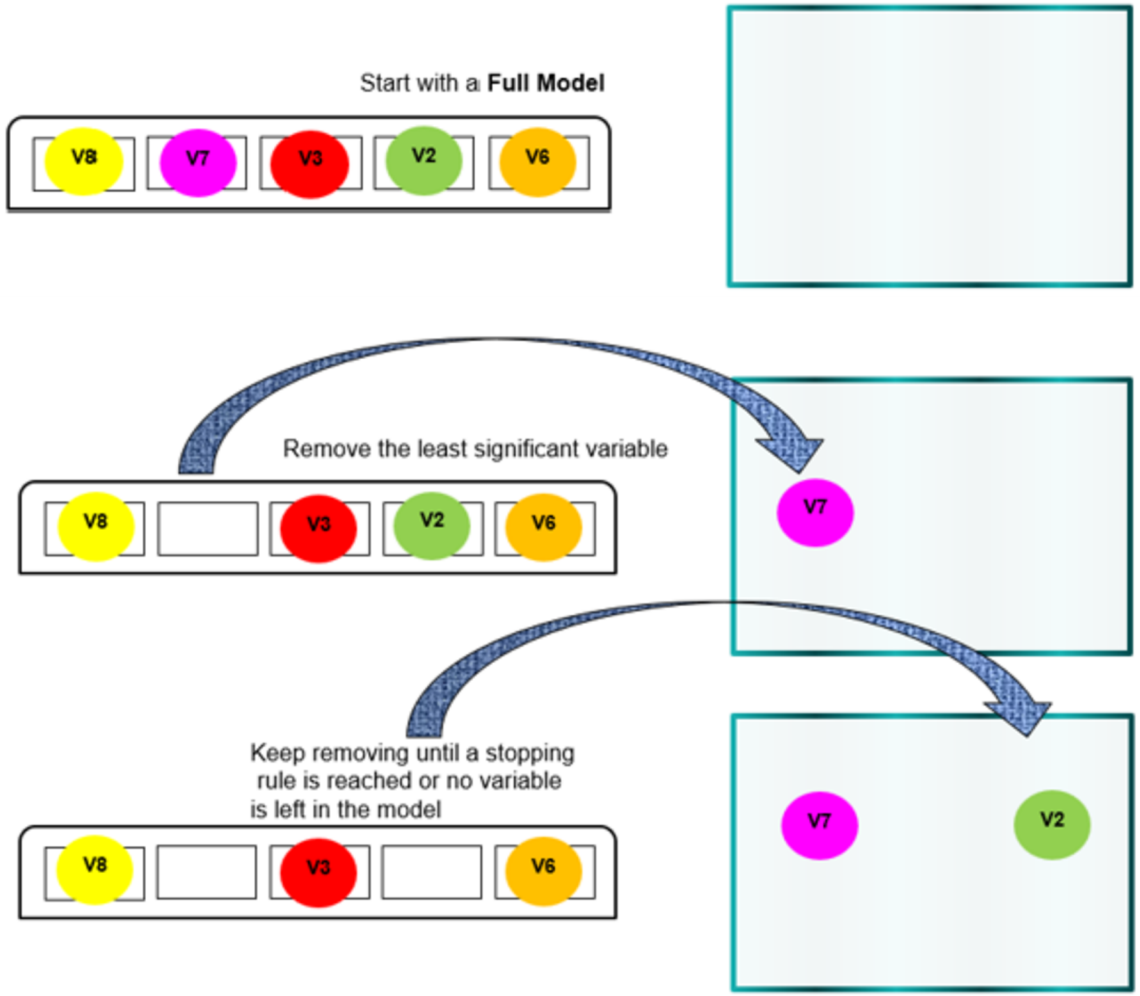
   a. Smallest P-Value
   b. Provides the highest drop in model error (or highest increase in accuracy)

2. How do we determine stopping criterion?

   a. Specified maximum P-value. Variables will no longer be chosen if their P-values exceed the threshold
   b. No more improvement in accuracy can be achieved
   b. Model achieves the number of desired variables

Scikit-Learn Implementation : **SequentialFeatureSelector**

1b. Sequential Backward Selection (SBS)



An iterative method wherein we start with a FULL model i.e., all features in the model

Next, we remove the variable from the model which gives the worst evaluation measure value.

This process continues until a preset criterion is achieved or all the variables are removed from the model

1. How do we determine least 'significance'?

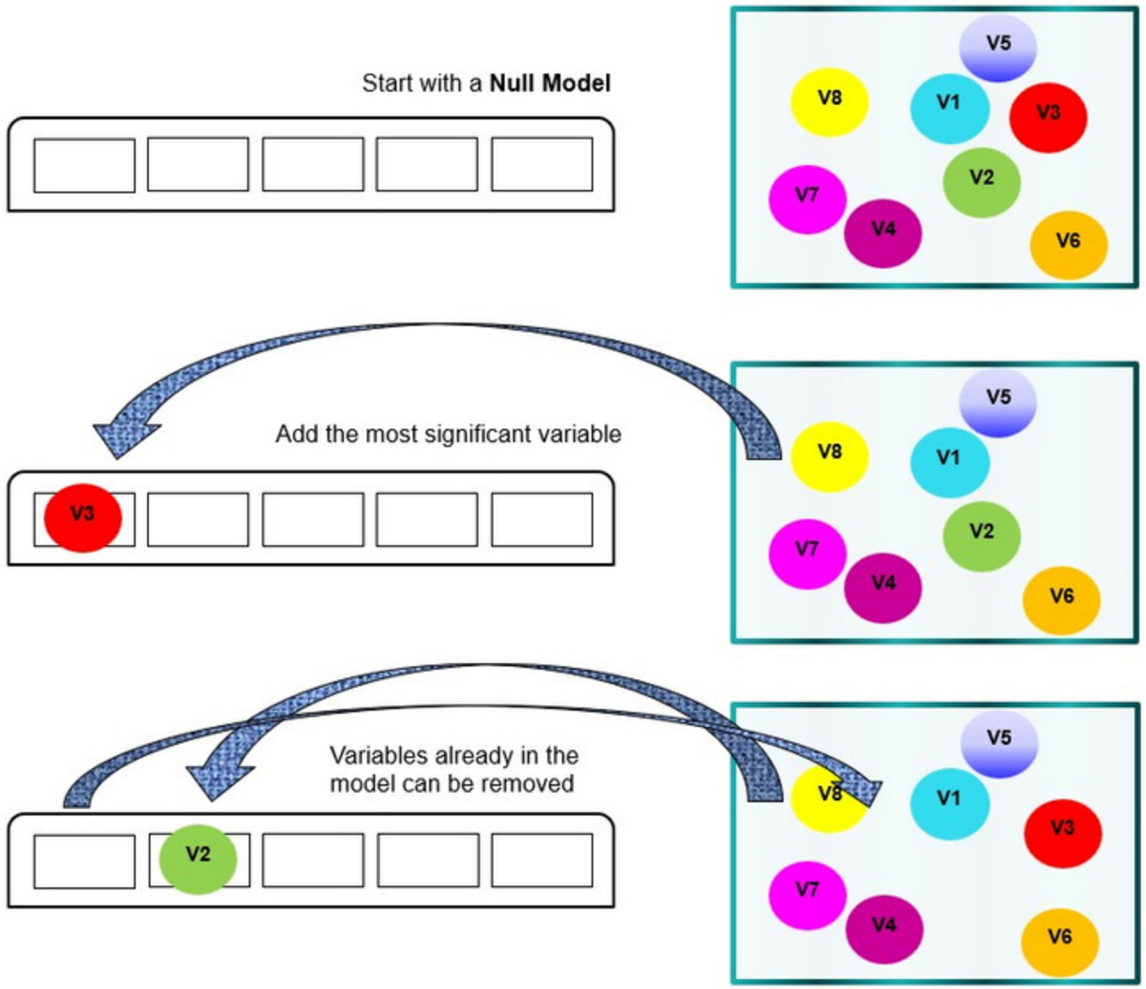
   a. Highest P-Value in the model
   b. It's elimination from the model causes the lowest increase in error (lowest decrease in accuracy)

2. How do we determine stopping criterion?

   a. When all remaining variables have a P-value lower than some specified value
   b. No more improvement in accuracy can be achieved
   c. Model achieves the number of desired variables

Scikit-Learn Implementation : **SequentialFeatureSelector**

1c&d. Sequential Floating Versions of Forward/Backward



1c. Sequential Forward Floating Selection (SFFS)
1d. Sequential Backward Floating Selection (SBFS)

A combination of Forward and Backward selection wherein variables already selected for the model can be removed in next iterations or removed features can be added back .

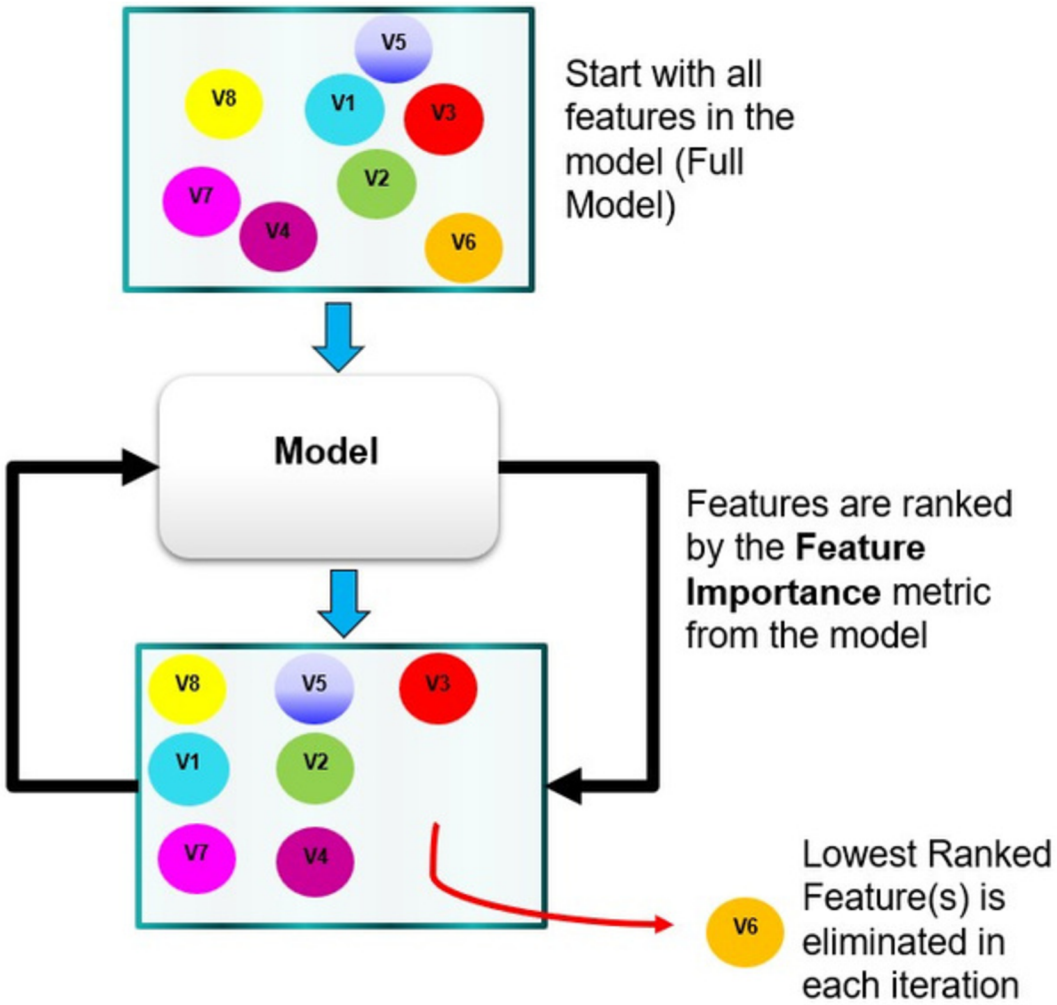
This process continues until a preset criterion is achieved.

Scikit-Learn Implementation :

SequentialFeatureSelector

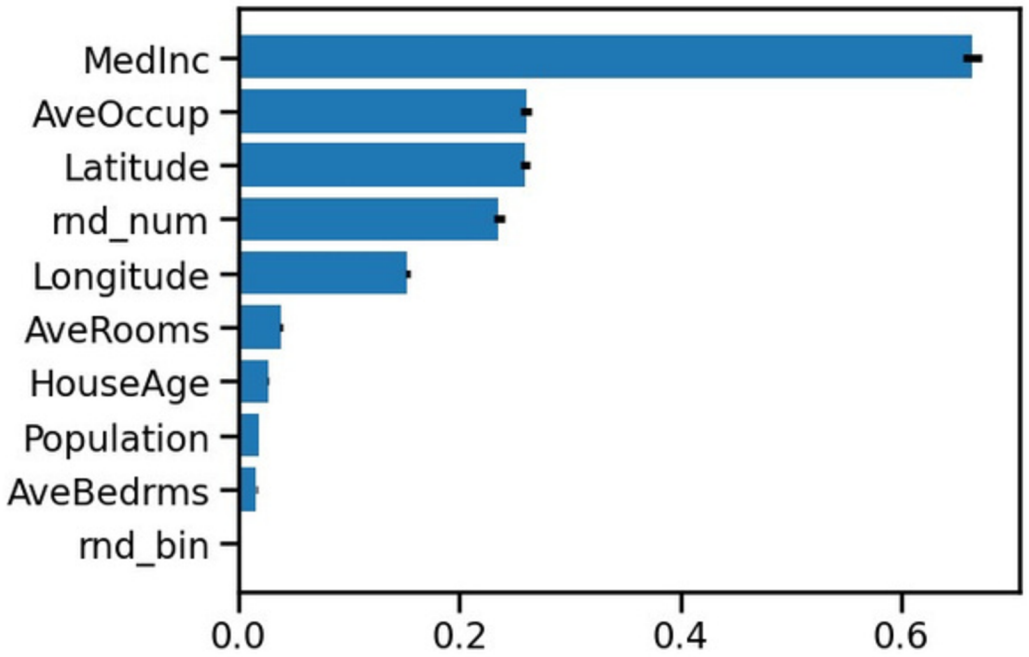
2. Recursive Feature Elimination RFE)

Word Definition: “involvingdoing or saying the same thing several times in order to produce a particular result or effect”



Feature Importance Score

RFE is applicable to models for which we can compute a**feature importance score**.



Examples of Feature Importance Scores

A. Model Coefficients

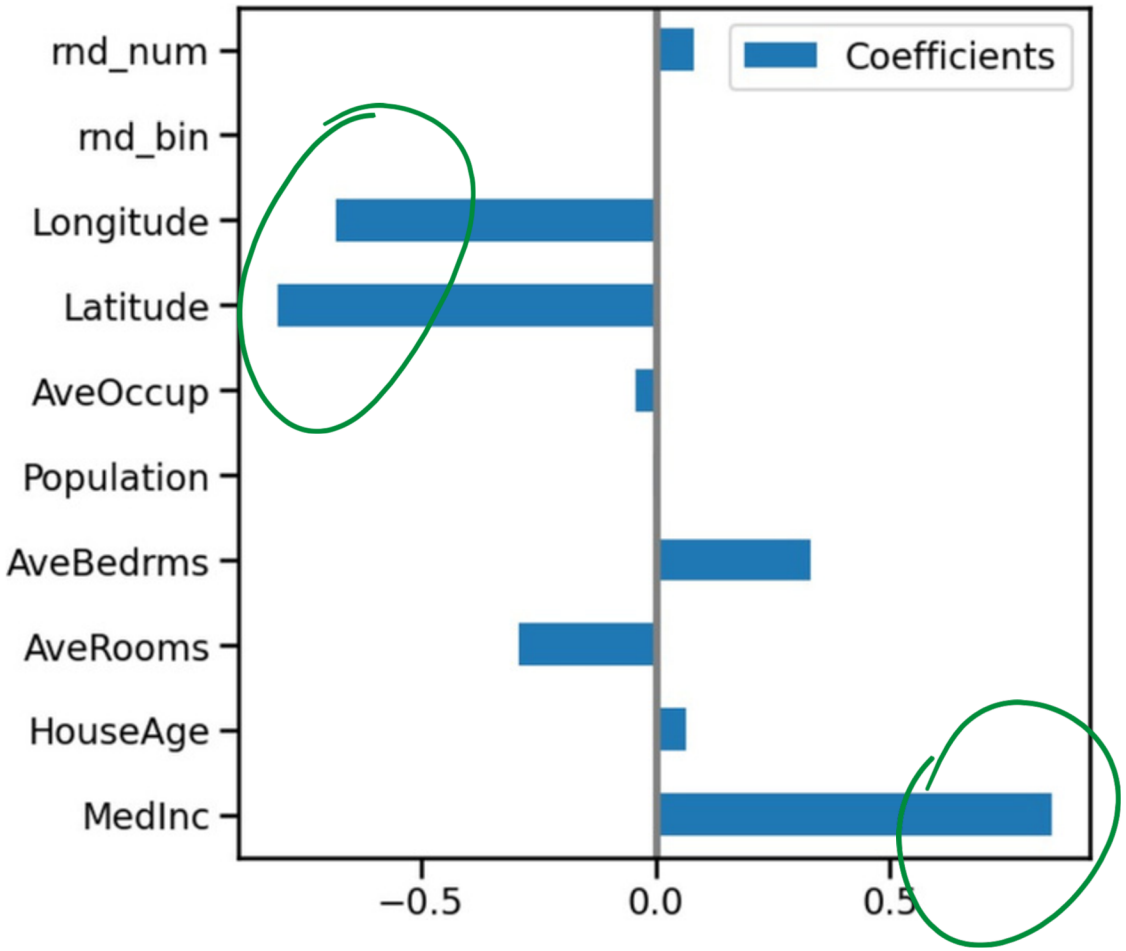
Feature importance in models such as Linear or Logistic regression can be obtained by examining the size of the coefficients.

Notes:

1.The coefficients must be from a standardized dataset

2.Comparison is made on absolutevalues

3.Caution: Highly correlated features can introduce instability of the coefficients



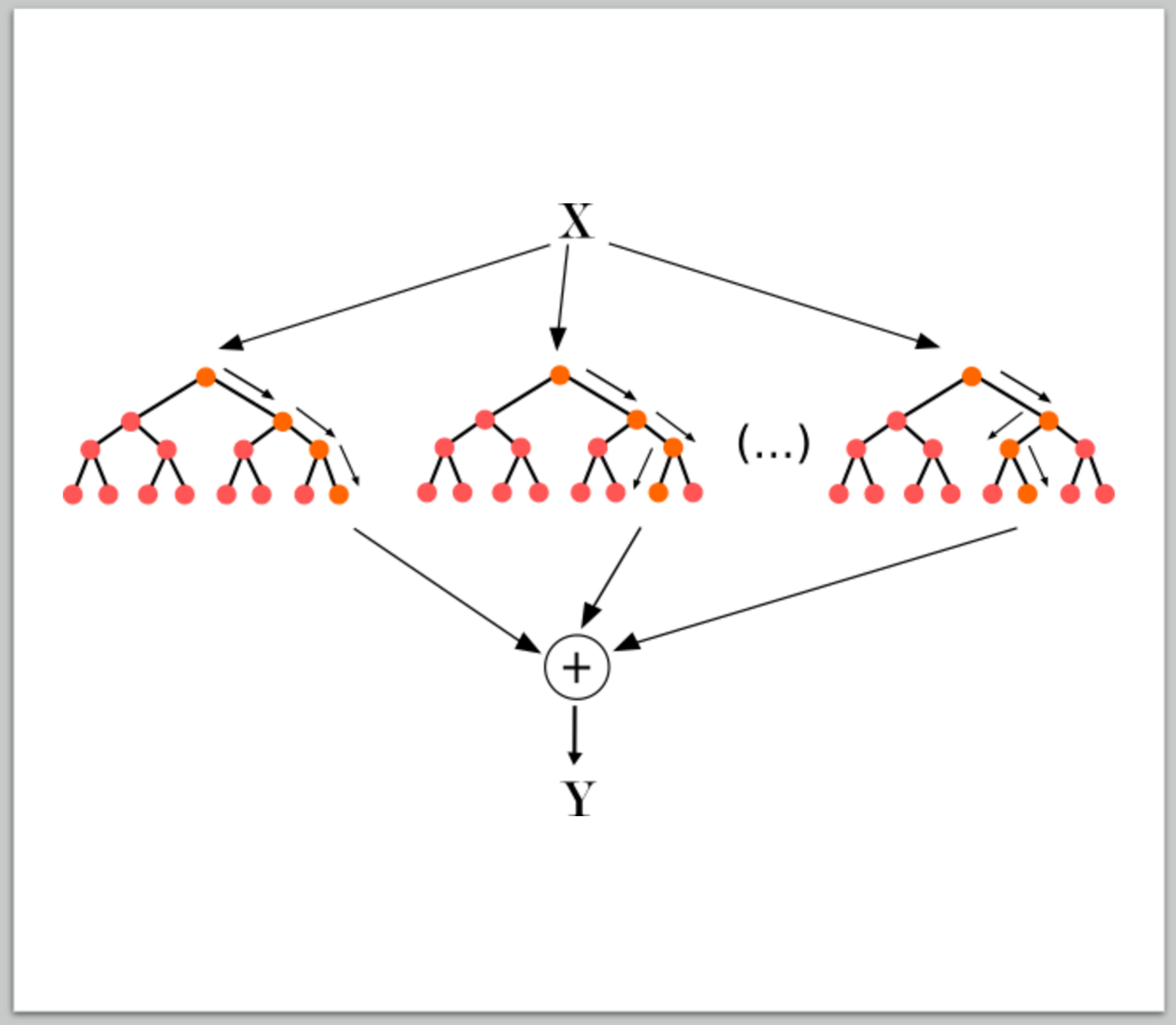
B. Gini importance

(or mean decrease impurity/Entropy)

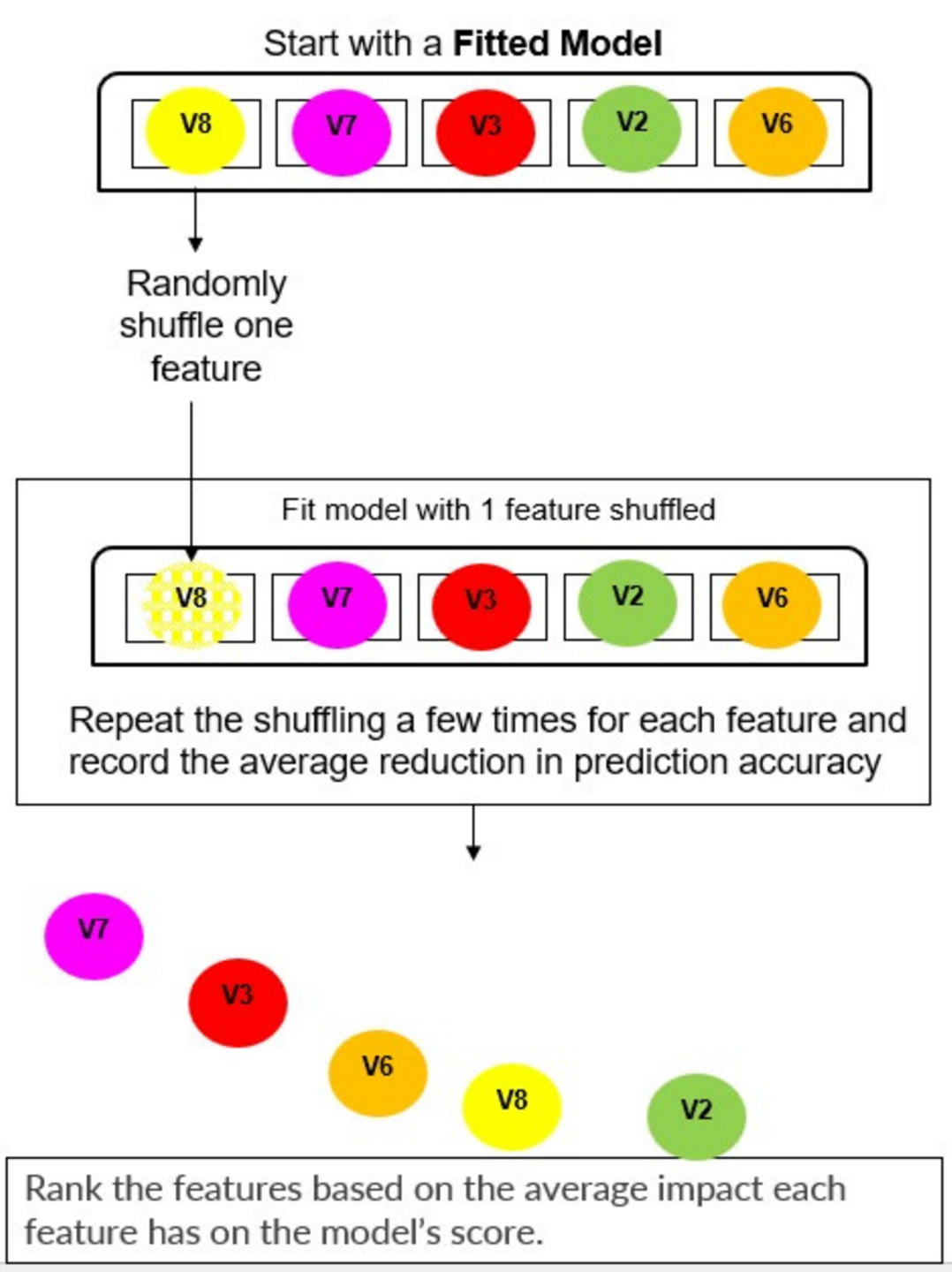
For Decision Tree based algorithms such as Random Forests

For each feature, calculate the ***total decrease in node impurity****and the average over all trees of the ensemble.*

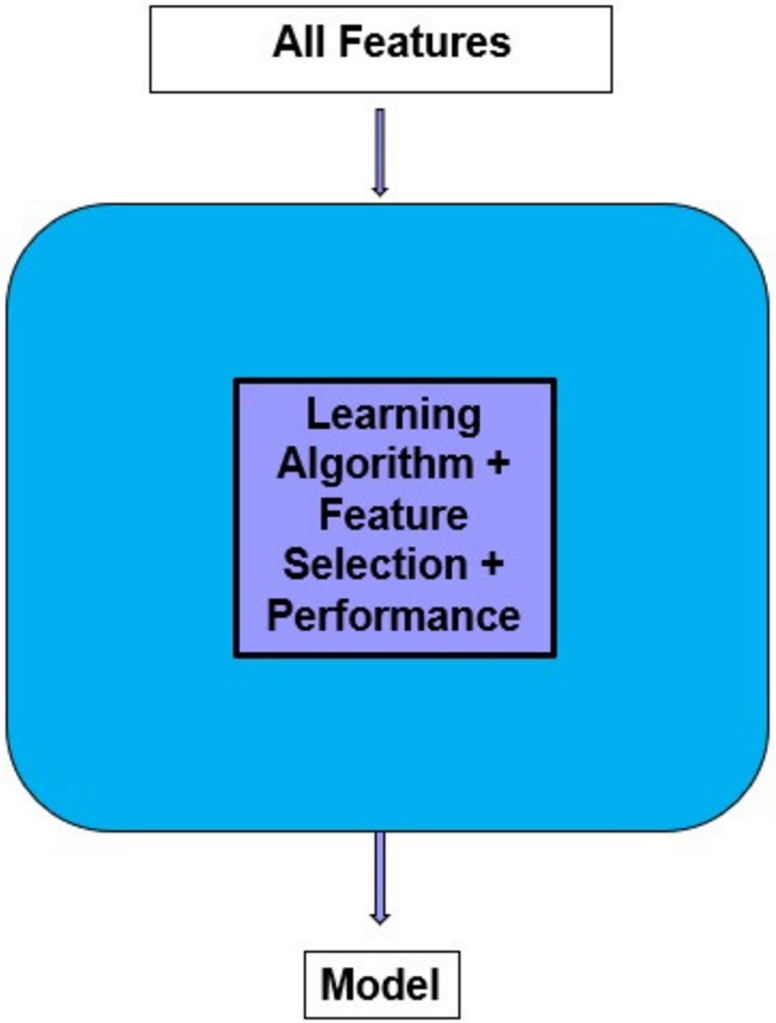
Alternative tweaks:
-How many times the feature is used in a DT/Forest
-Average positioning



C. Permutation importance



Embedded Methods



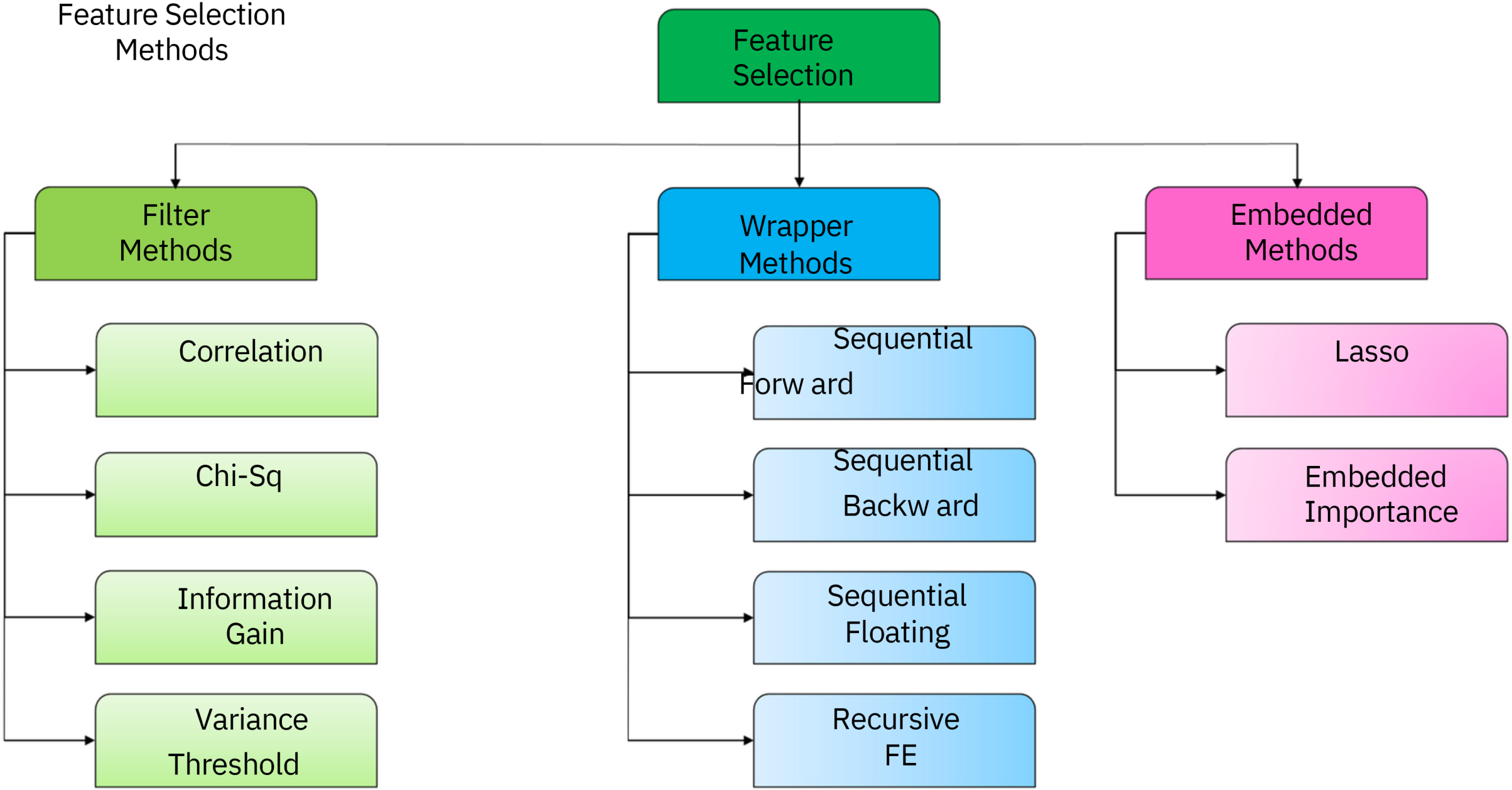
The feature selection function isin-built within the algorithm:

- **Regularization**
  - **Lasso (L1)**

- **Max_featuresin decision trees, Random Forests etc. (embedded feature importance)**

Scikit-Learn Implementation : **SelectFromModel**

SUMMARY
Feature Selection Methods




Let’s make some
variables disappear