Lecture 13

Dimensionality Reduction I: Feature Selection

STAT 451: Intro to Machine Learning, Fall 2021

Sebastian Raschka

https://sebastianraschka.com/teaching/stat451-fs2021/

Dimensionality Reduction

Why do we care?

Dimensionality Reduction

Why do we care?

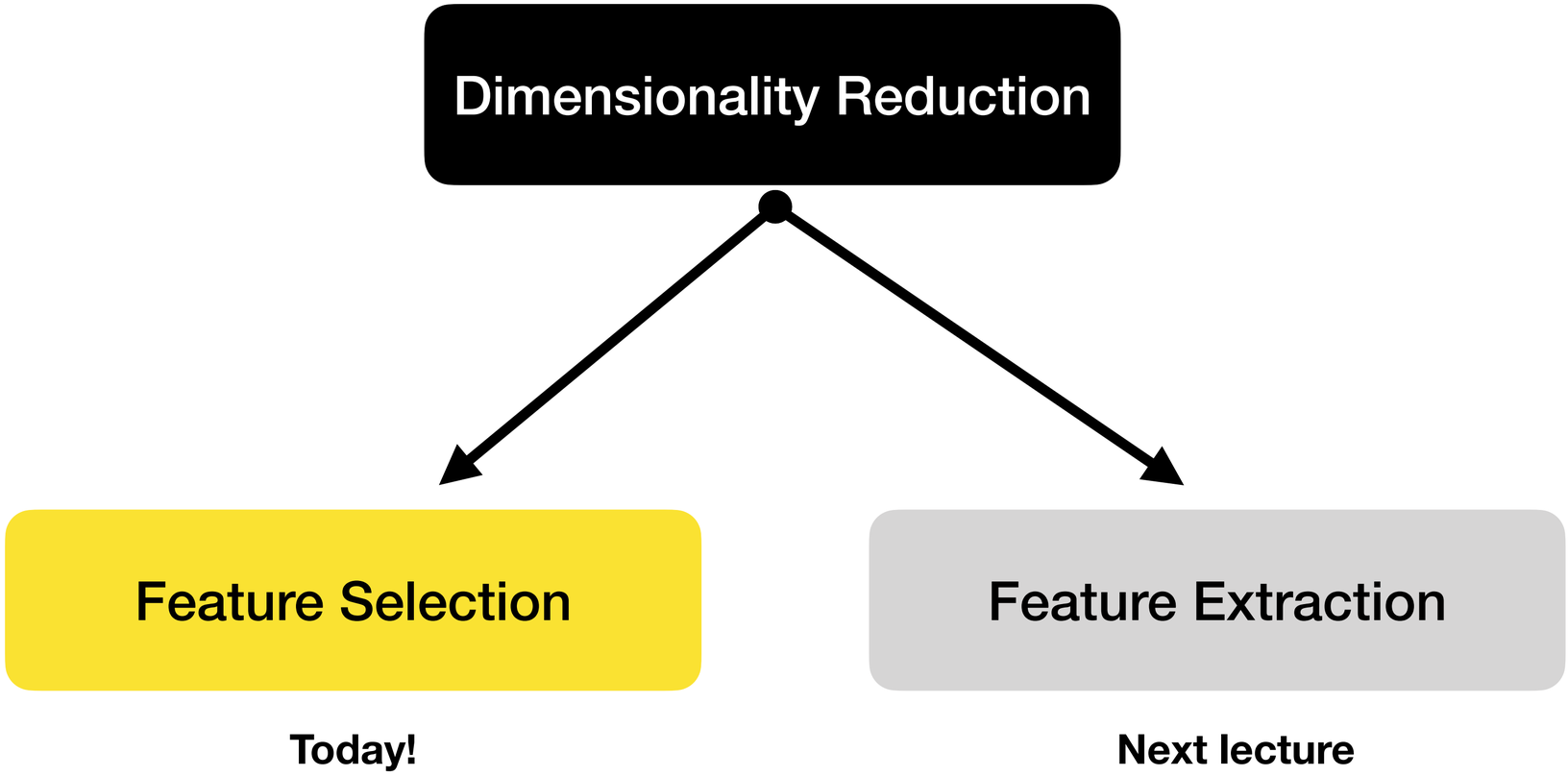
Curse of dimensionality

Computational efficiency

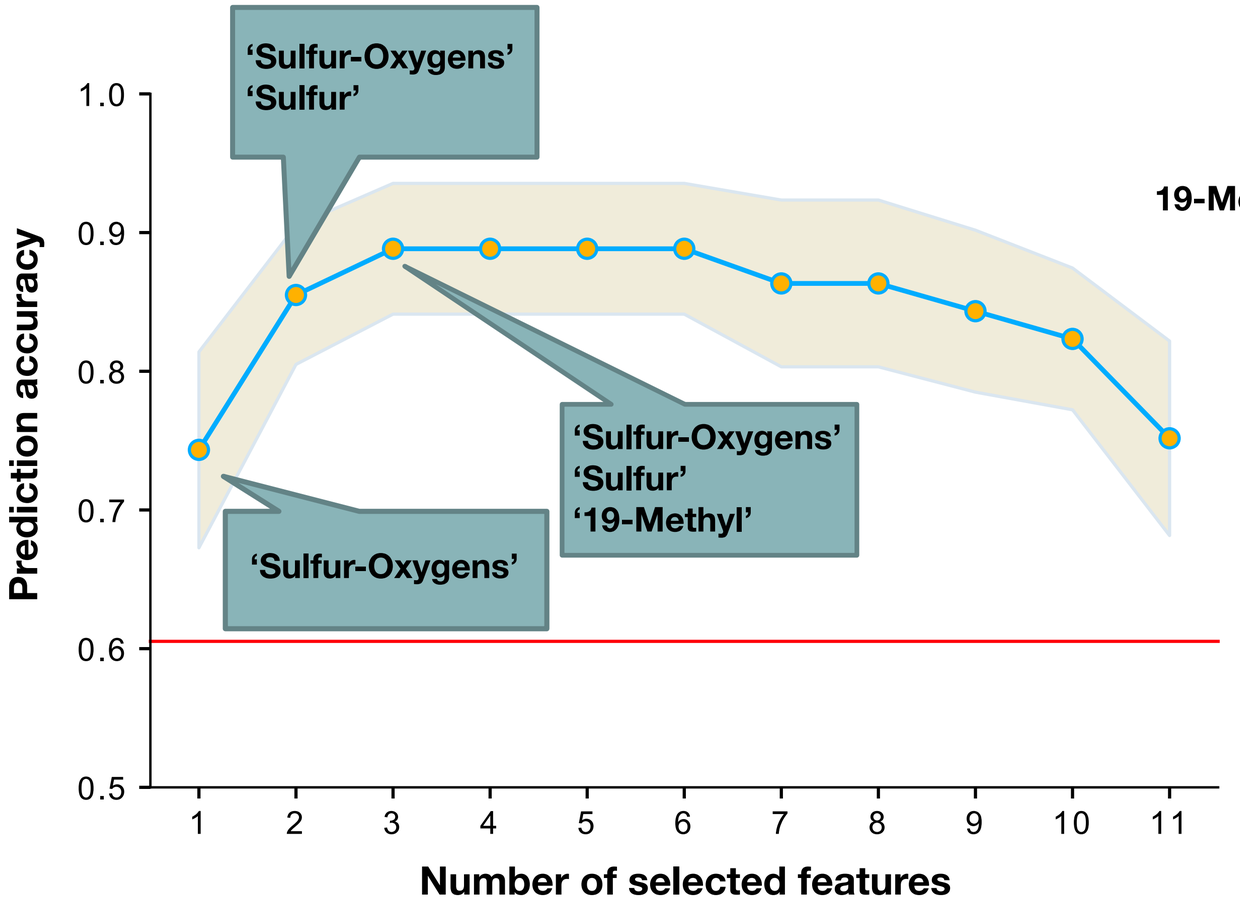
Easier data collection

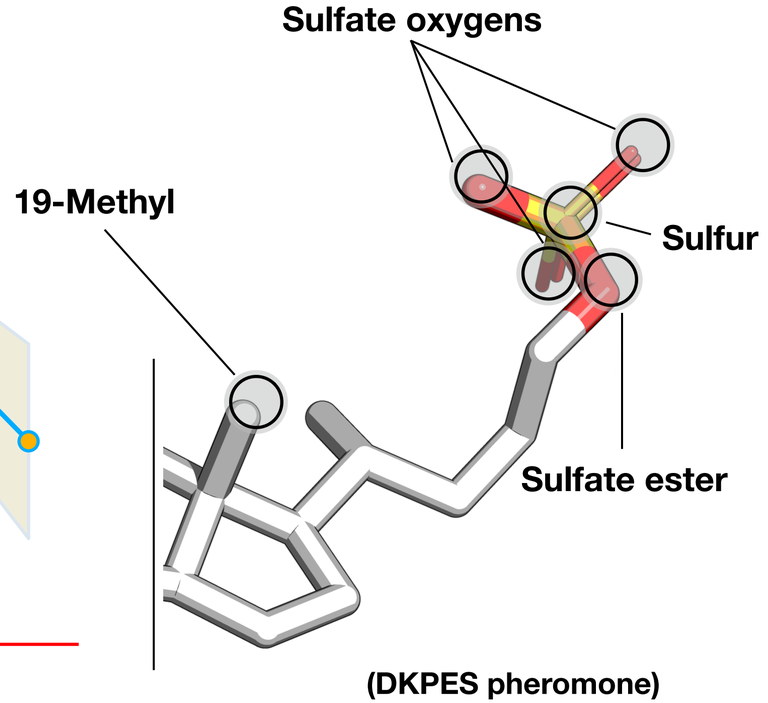
Storage space

Interpretability


Dimensionality Reduction
Feature Selection
Feature Extraction
Today!
Next lecture


Discovery of a pheromone receptor inhibitor for invasive species control (sea lamprey) in the Great Lakes
https://en.wikipedia.org/wiki/Sea_lamprey#/media/File:Sea_lamprey_on_brown_trout_flipped.jpg

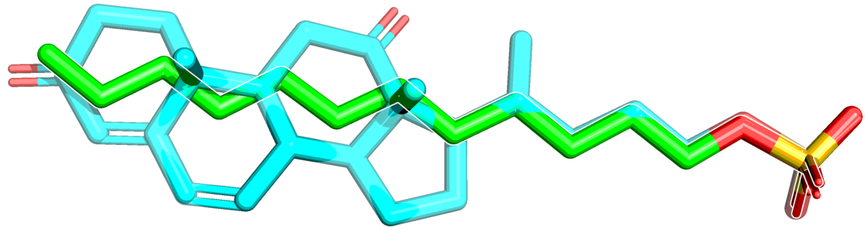




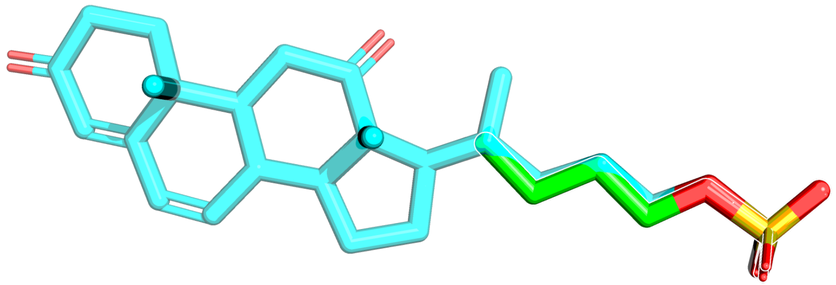
22

Sebastian Raschka, Nan Liu, Santosh Gunturu, Anne M. Scott, Mar Huertas, Weiming Li, and Leslie A. Kuhn (2018) *Facilitating the Hypothesis-driven Prioritization of Small Molecules in Large Databases: Screenlamp and its Application to GPCR Inhibitor Discovery.* Journal of Computer-Aided Molecular Design, 32(3), 415-433.

"Sulfate-tail" sufficient for bioactivity



69% signal inhibition

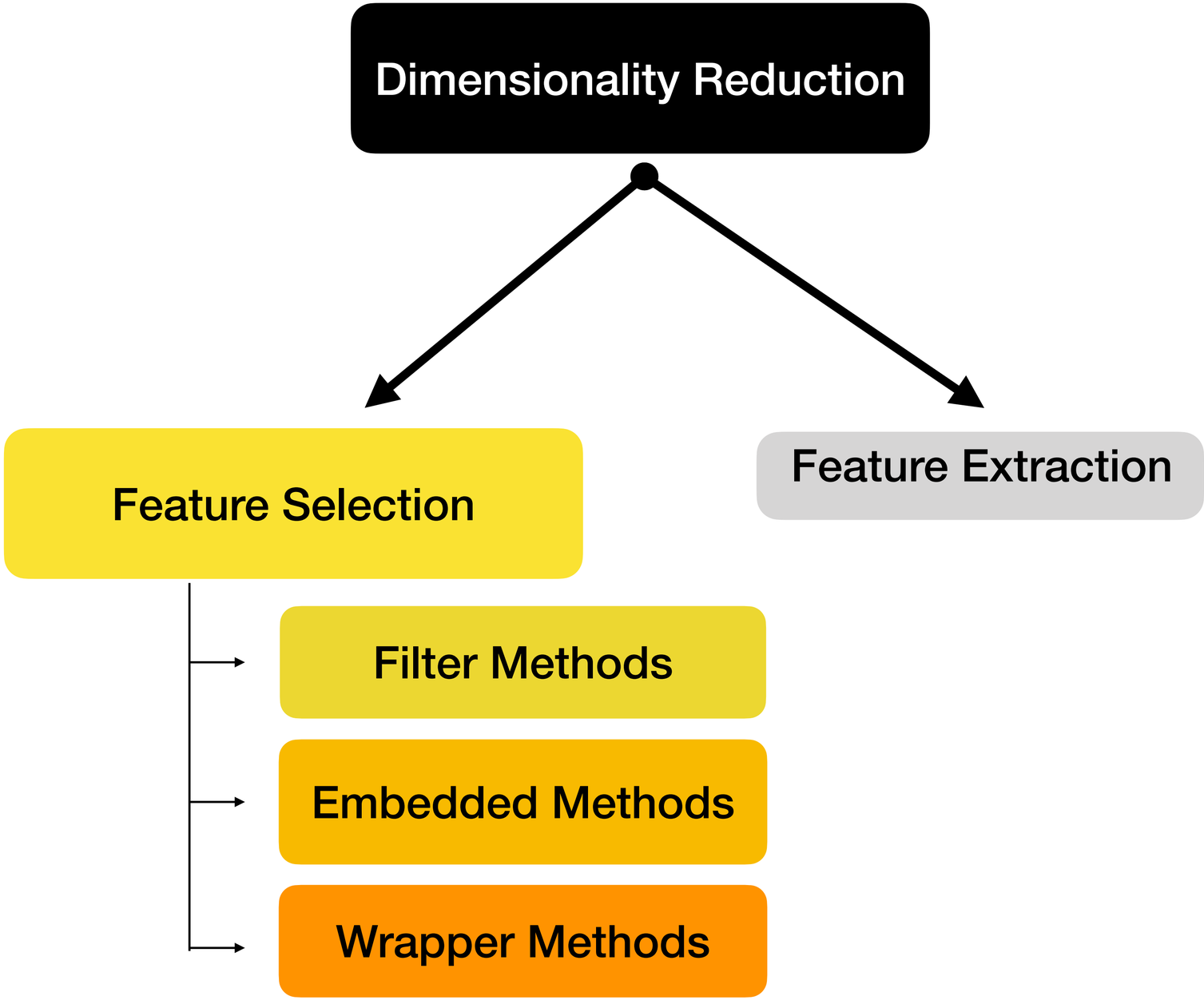


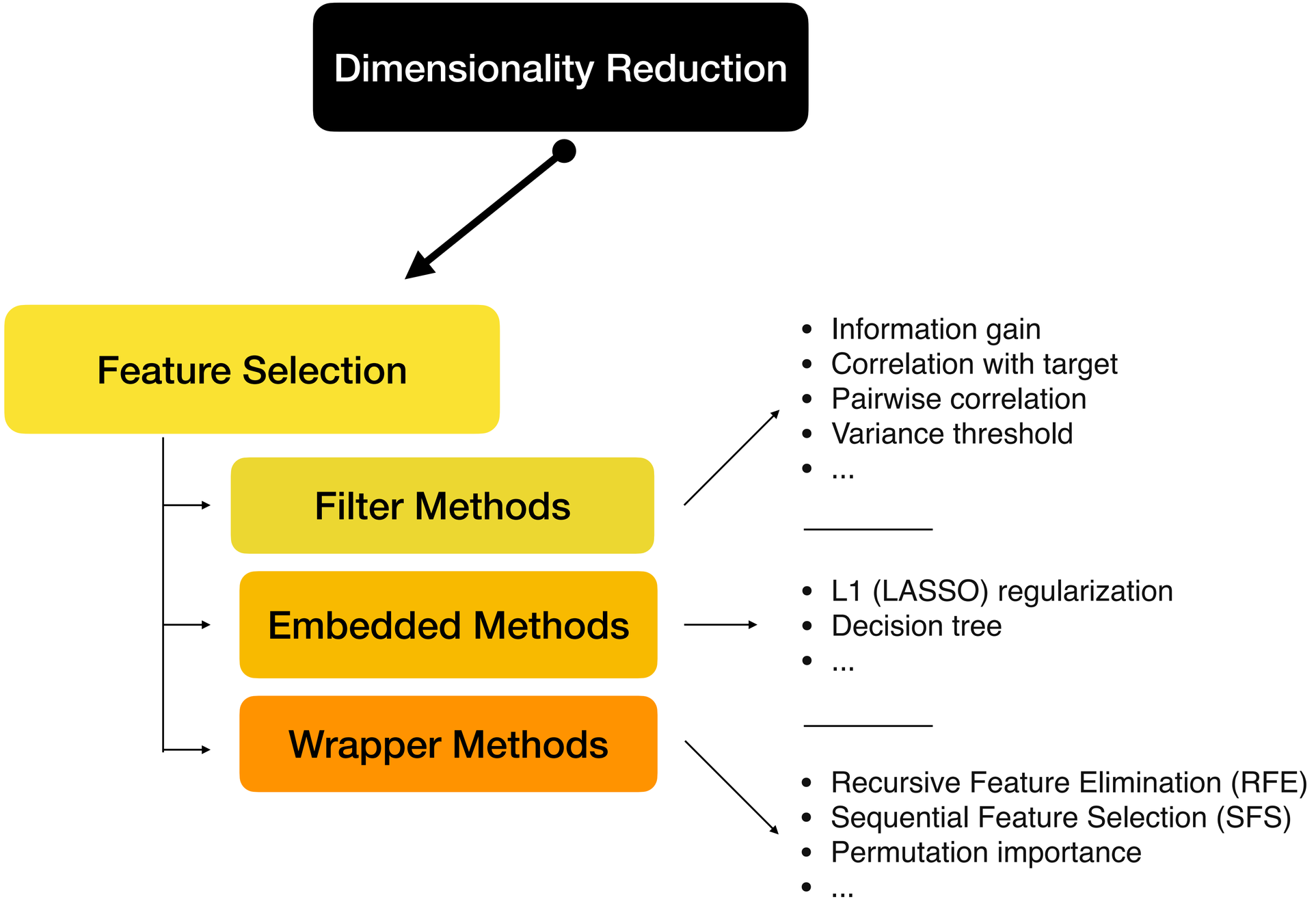
62% signal inhibition

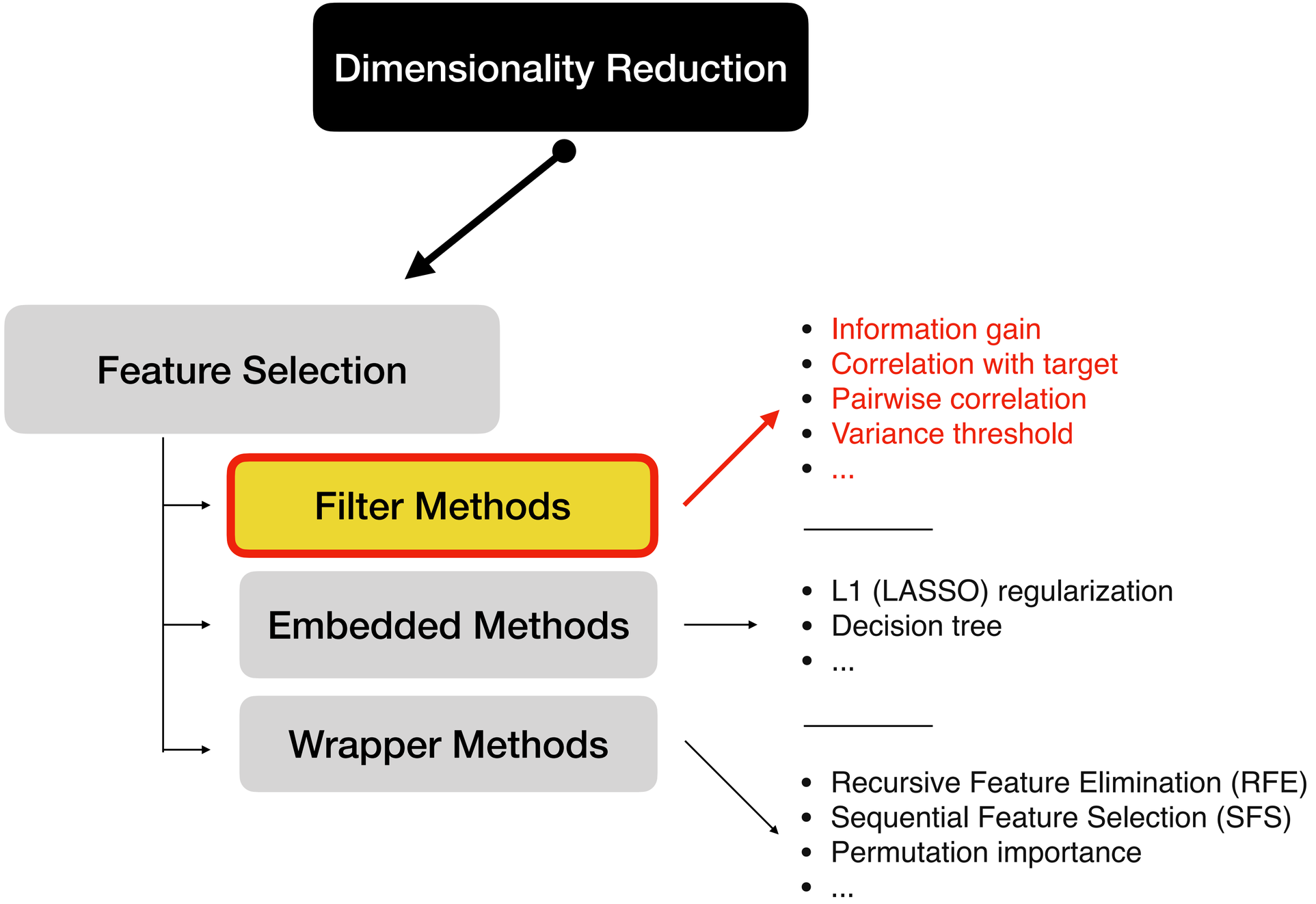
23

1. Different categories of feature selection
2. Filter methods
3. Embedded methods
    3.1. L1-regularized logistic regression
    3.2. Random forest feature importance
4. Wrapper methods
    4.1. Recursive feature elimination
    4.2. Permutation importance
    4.3. Permutation importance code example
    4.4. Sequential feature selection
    4.5. Sequential feature selection code example

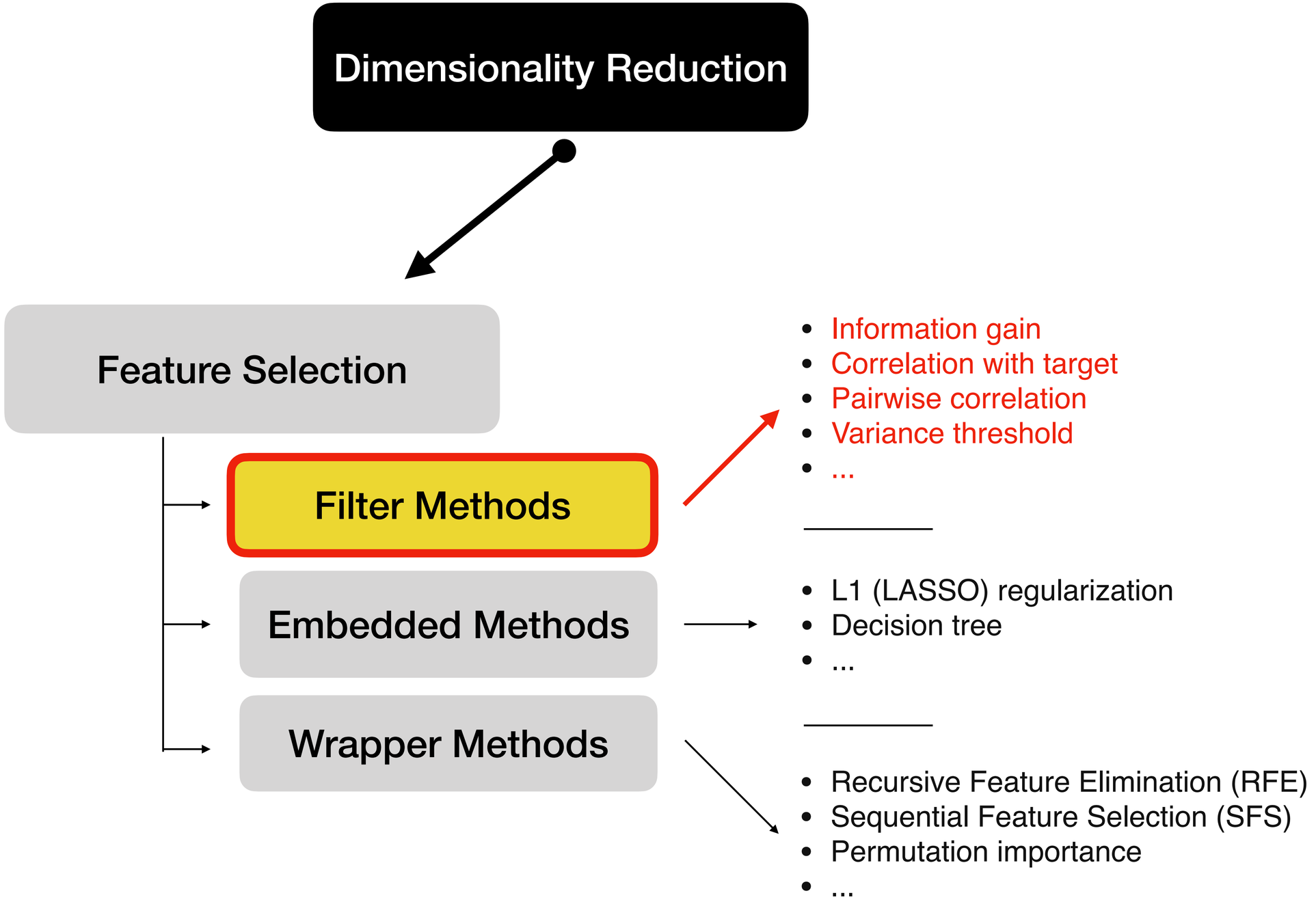
1. **Different categories of feature selection**
2. Filter methods
3. Embedded methods
    - 3.1. L1-regularized logistic regression
    - 3.2. Random forest feature importance
4. Wrapper methods
    - 4.1. Recursive feature elimination
    - 4.2. Permutation importance
    - 4.3. Permutation importance code example
    - 4.4. Sequential feature selection
    - 4.5. Sequential feature selection code example


Dimensionality Reduction
Feature Selection
Feature Extraction
Filter Methods
Embedded Methods
Wrapper Methods


Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
• Information gain
• Correlation with target
• Pairwise correlation
• Variance threshold
• ...
• L1 (LASSO) regularization
• Decision tree
• ...
• Recursive Feature Elimination (RFE)
• Sequential Feature Selection (SFS)
• Permutation importance
• ...

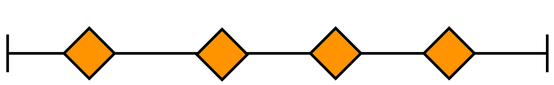

Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
Information gain
Correlation with target
Pairwise correlation
Variance threshold
...
L1 (LASSO) regularization
Decision tree
...
Recursive Feature Elimination (RFE)
Sequential Feature Selection (SFS)
Permutation importance
...

1. Different categories of feature selection
2. **Filter methods**
3. Embedded methods
    - 3.1. L1-regularized logistic regression
    - 3.2. Random forest feature importance
4. Wrapper methods
    - 4.1. Recursive feature elimination
    - 4.2. Permutation importance
    - 4.3. Permutation importance code example
    - 4.4. Sequential feature selection
    - 4.5. Sequential feature selection code example

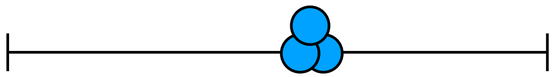

Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
Information gain
Correlation with target
Pairwise correlation
Variance threshold
...
L1 (LASSO) regularization
Decision tree
...
Recursive Feature Elimination (RFE)
Sequential Feature Selection (SFS)
Permutation importance
...

Variance Threshold (Filter)

- Compute the variance of each feature
- Assume that features with a higher variance may contain more useful information



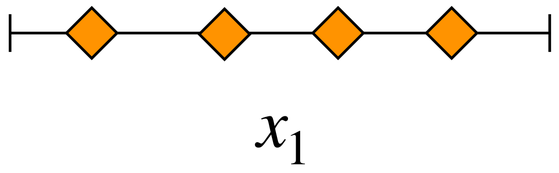
x_1

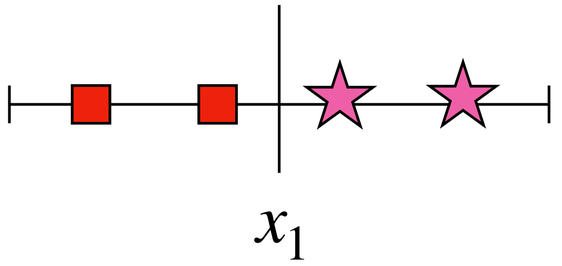


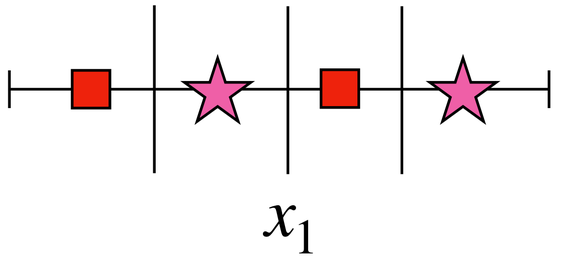
x_2

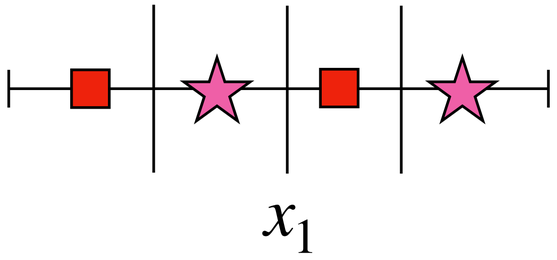
Variance Threshold (Filter)

- Compute the variance of each feature
- Assume that features with a higher variance may contain more useful information





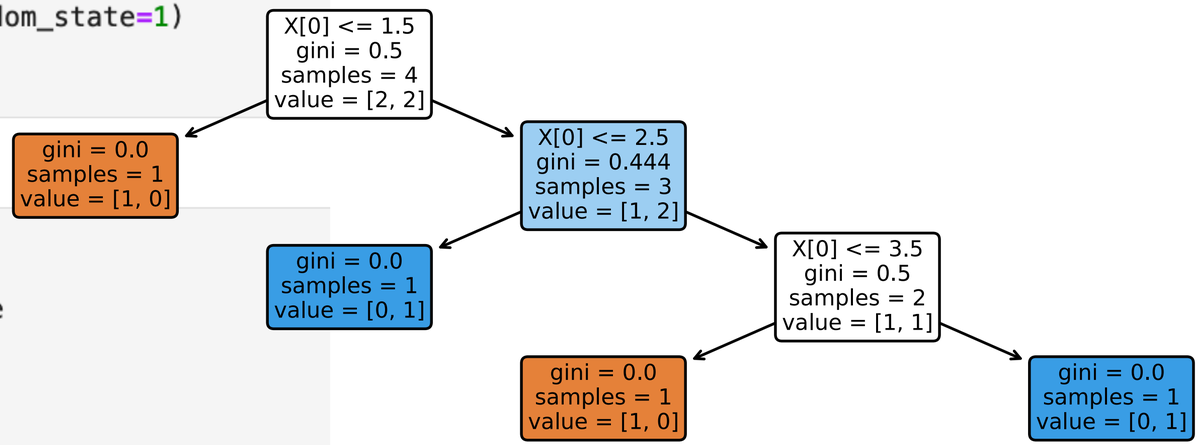





```
import numpy as np
from sklearn.tree import DecisionTreeClassifier

X = np.array([[1.], [2.], [3.], [4.]])
y = np.array([0, 1, 0, 1])


tree = DecisionTreeClassifier(random_state=1)
tree.fit(X, y)
tree.score(X, y)
```

```
1.0
```

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree


plt.figure(figsize=(10, 3))

plot_tree(tree,
          filled=True,
          rounded=True)

plt.show()
```



Variance

Variance of discrete random variable:

$$\text{Var}(X) = \sum_{i=1}^{n} p_i \cdot \left(x_i - \mu\right)^2$$

E.g., dataset with n datapoints (for sample variance, *n-1*)

$$\text{Var}(X) = \frac{1}{n} \sum_{i=1}^{n} \left(x_i - \mu\right)^2$$

E.g., dataset with n datapoints (for sample variance, *n-1*)

$$\text{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu\right)^2$$

Variance of Bernoulli variable (Boolean feature, e.g., after one-hot encoding)

$$\text{Var}(X) = p(1-p)$$

```
data_var = np.var(50*[0] + 50*[1]) # i.e., p = 0.5
data_var
```

```
0.25
```

```
0.5 * 0.5
```

```
0.25
```

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.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 returning F-statistic and p-values.
feature_selection.r_regression(X, y, *[, center])	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

VarianceThreshold $0.8 \times (1 - 0.8) = 0.16$

```
from sklearn.preprocessing import OneHotEncoder


X = [['blue'], ['green'], ['blue'], ['blue'], ['green'], ['red'], ['blue'], ['green']]
y = [0, 0, 1, 0, 0, 1, 0, 0]

enc = OneHotEncoder(drop='first')
enc.fit(X)
X_ohe = enc.transform(X)
X_ohe.toarray()
```

```
array([[0., 0.],
       [1., 0.],
       [0., 0.],
       [0., 0.],
       [1., 0.],
       [0., 1.],
       [0., 0.],
       [1., 0.]])
```

```
from sklearn.feature_selection import VarianceThreshold

sel = VarianceThreshold(threshold=(.8 * (1 - .8)))

sel.fit(X_ohe)
sel.transform(X_ohe).toarray()
```

```
array([[0.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [1.]])
```

Be aware of feature scaling!

E.g., dataset with n datapoints (for sample variance, *n-1*)

$$\text{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu\right)^2$$

```
np.random.seed(123)

data = np.random.random_sample(100)
np.var(data)
```

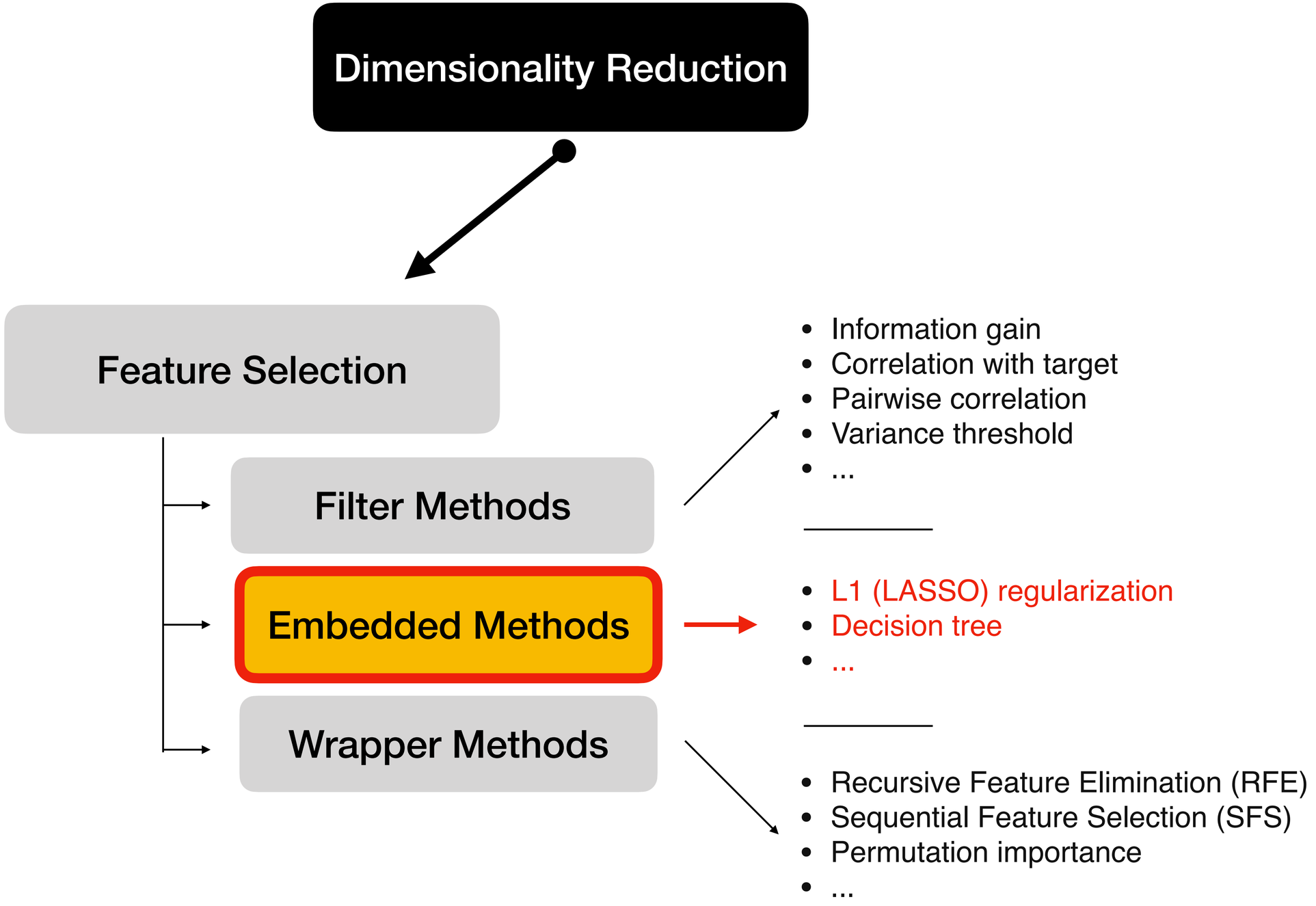
```
0.060211775685055 76
```

```
np.var(data*10)
```

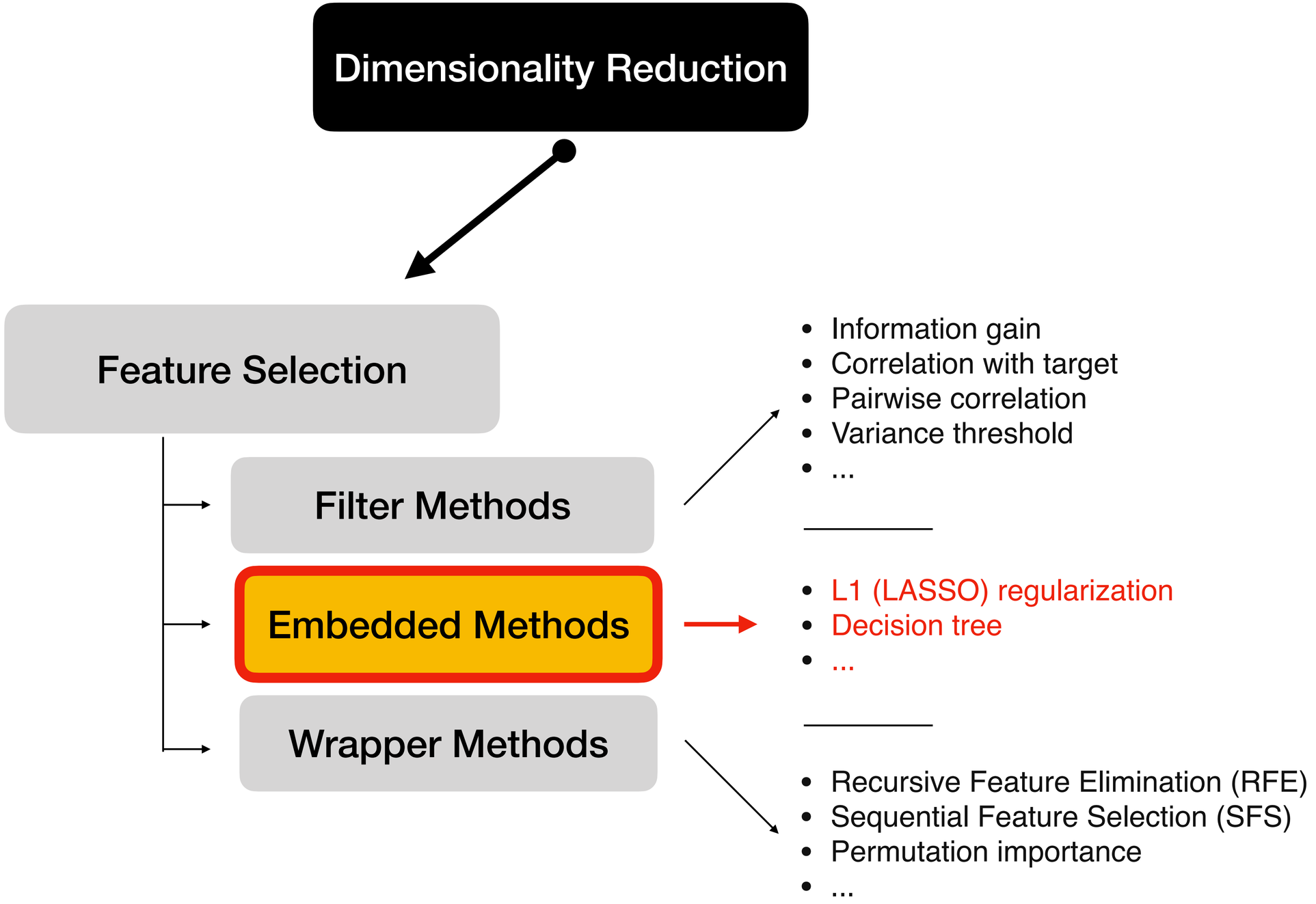
```
6.021177568505576
```

Variance Threshold (Filter)

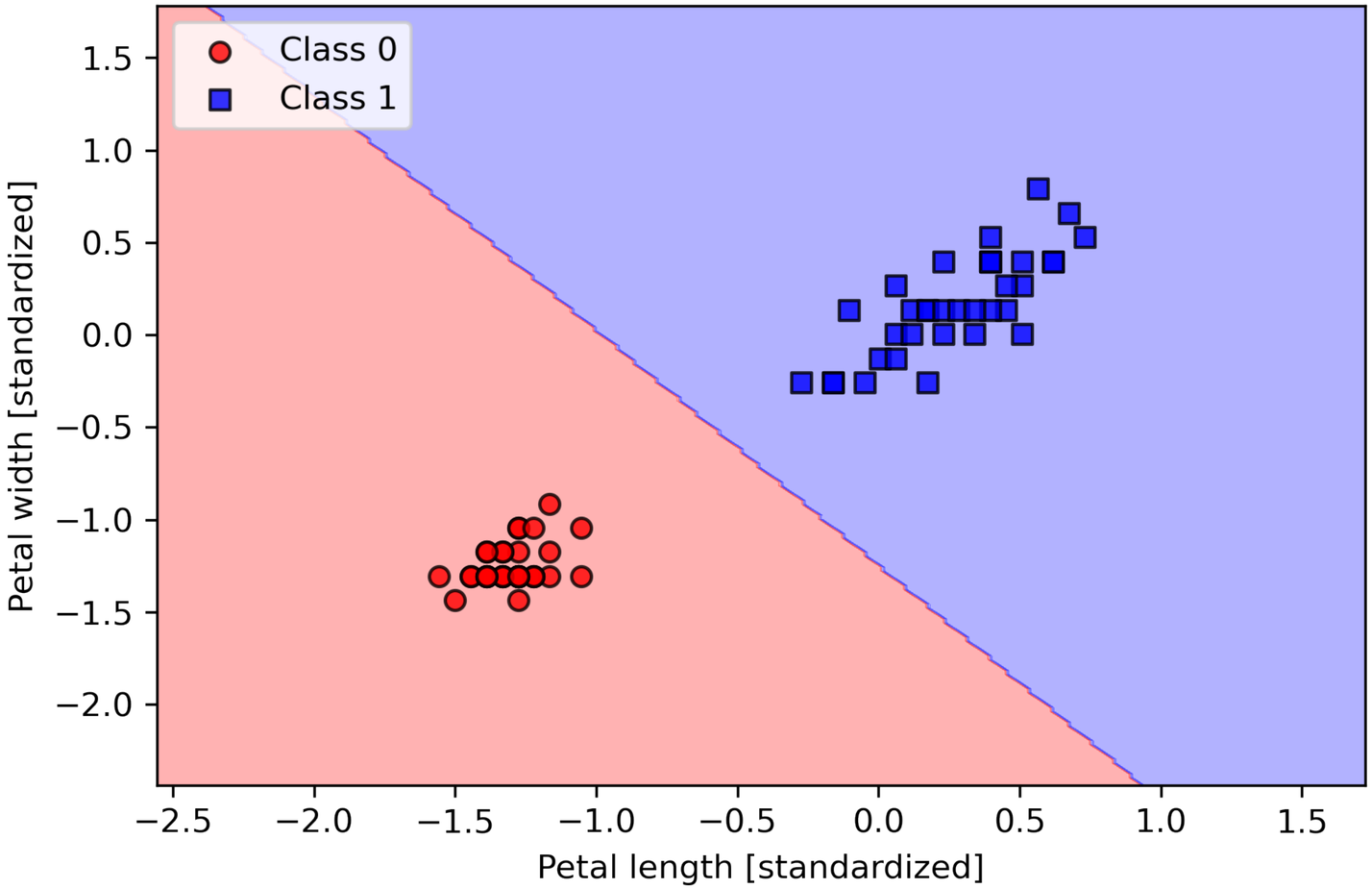
- Compute the variance of each feature
- Assume that features with a higher variance may contain more useful information
- Select the subset of features based on a user-specified threshold ("keep if greater or equal to *x*" or "keep the the top *k* features with largest variance")
- **Good:** fast!
- **Bad:** does not take the relationship among features into account


Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
• Information gain
• Correlation with target
• Pairwise correlation
• Variance threshold
• ...
• L1 (LASSO) regularization
• Decision tree
• ...
• Recursive Feature Elimination (RFE)
• Sequential Feature Selection (SFS)
• Permutation importance
• ...

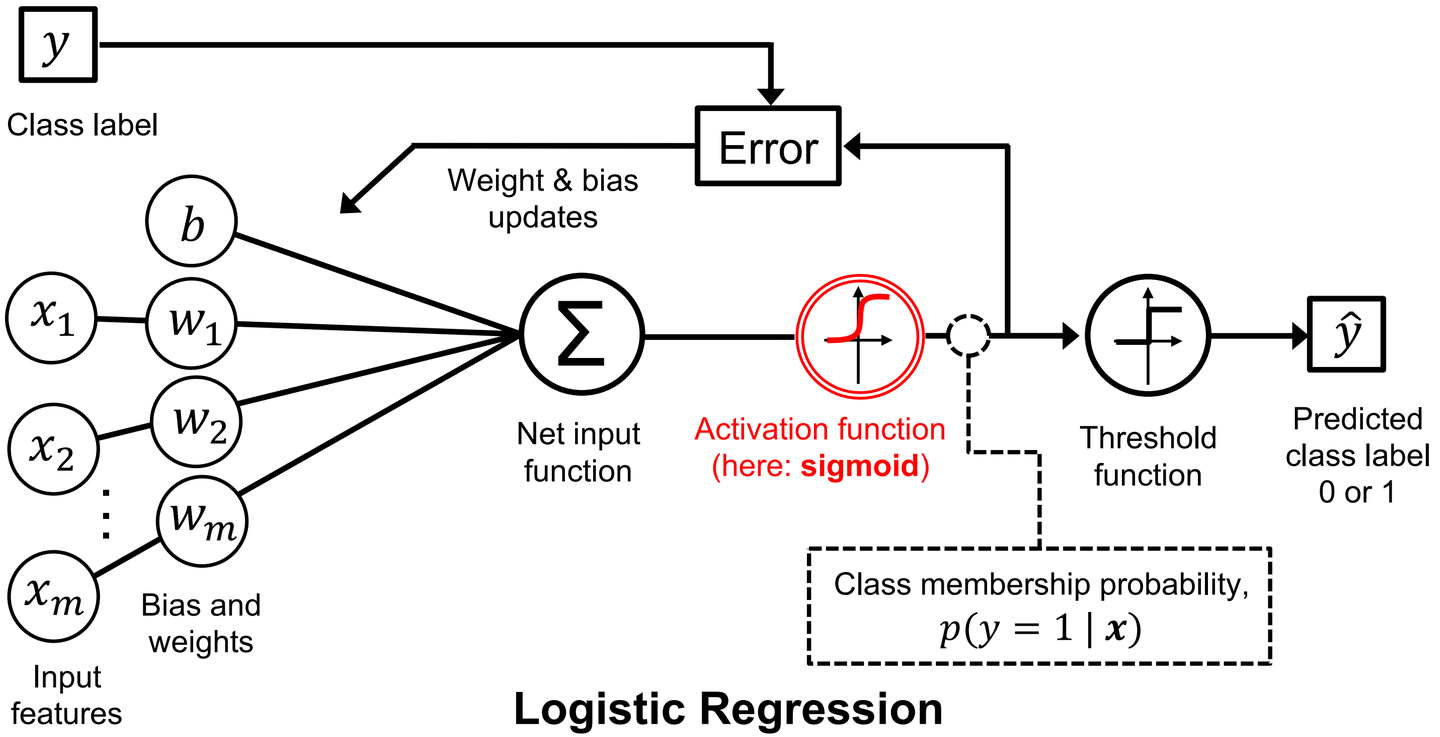
1. Different categories of feature selection
2. Filter methods
3. **Embedded methods**
    - **3.1. L1-regularized logistic regression**
    - 3.2. Random forest feature importance
4. Wrapper methods
    - 4.1. Recursive feature elimination
    - 4.2. Permutation importance
    - 4.3. Permutation importance code example
    - 4.4. Sequential feature selection
    - 4.5. Sequential feature selection code example


Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
• Information gain
• Correlation with target
• Pairwise correlation
• Variance threshold
• ...
• L1 (LASSO) regularization
• Decision tree
• ...
• Recursive Feature Elimination (RFE)
• Sequential Feature Selection (SFS)
• Permutation importance
• ...

Logistic Regression



Logistic Regression



Source: Raschka, Liu, and Mirjalili. M*achine Learning with PyTorch and Scikit-Learn, Ch 3*

Logistic Regression Hyperparameters

Regular loss function to minimize during training

$$L(\mathbf{w}, b \mid \mathbf{x}) = -\sum_{i=1} \left[y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(z^{(i)}\right)\right) \right]$$

L1 Norm /
LASSO (Least Absolute Shrinkage and Selection Operator)

L1 norm: $\lambda \, ||\mathbf{w}||_1 = \lambda \sum_{j=1}^{m} |w_j|$

L1 Regularization / LASSO (Embedded)

Least Absolute Shrinkage and Selection Operator

L1 penalty against complexity

$$\lambda \, ||\mathbf{w}||_1 = \lambda \sum_{j=1}^{m} |w_j|$$

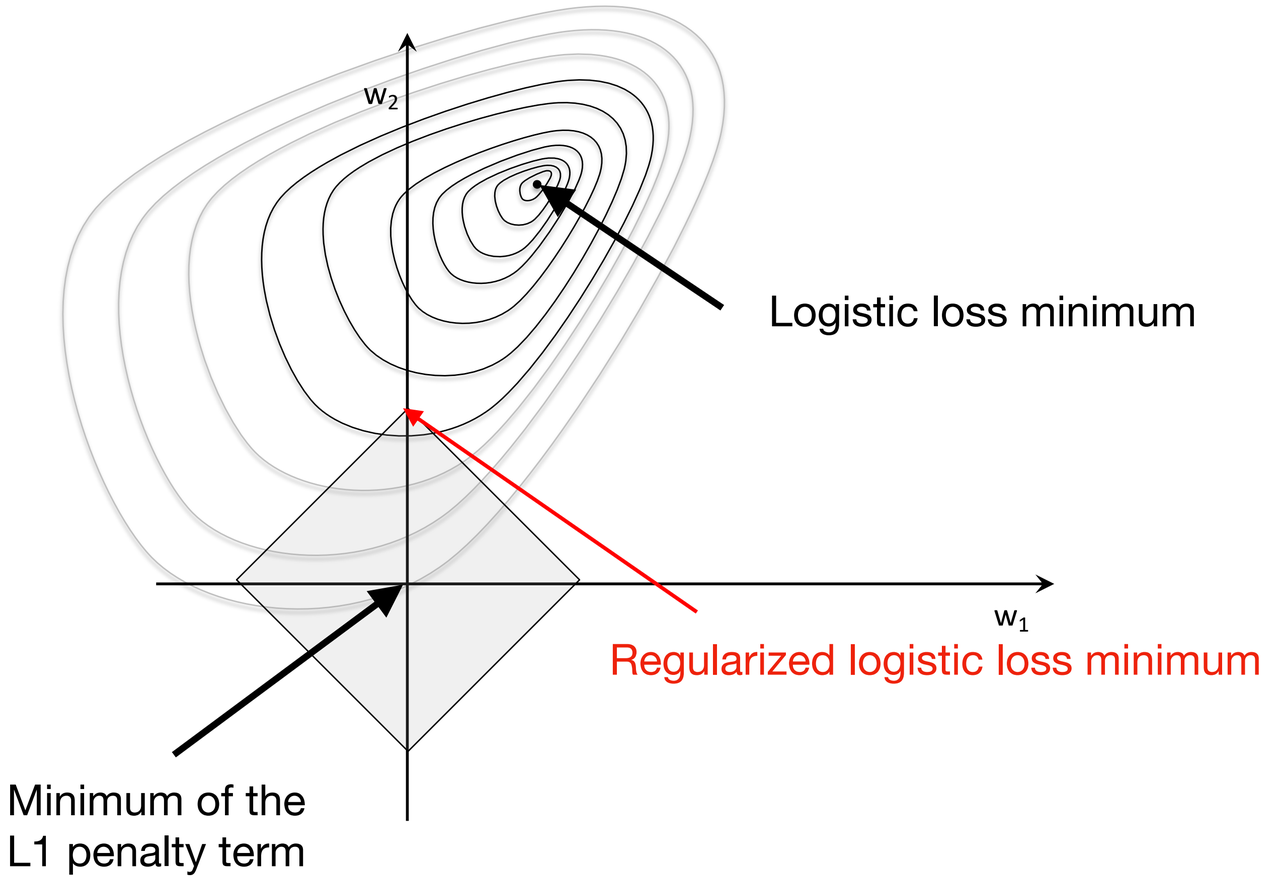
hyperparameter

L1-penalized loss

$$L_{L1}(\mathbf{w}, b \mid \mathbf{x}) = -\sum_{i=1} \left[y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(z^{(i)}\right)\right) \right] + \lambda \|w\|_1$$

L1 Regularization / LASSO (Embedded)

Least Absolute Shrinkage and Selection Operator



For more details, see Tibshirani, Ryan, and L. Wasserman. "A closer look at sparse regression." *Lecture notes* (2016).

L1 Regularization / LASSO (Embedded)

Least Absolute Shrinkage and Selection Operator

Wine Dataset

https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

```
from sklearn.model_selection import train_test_split

X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values

X_train, X_test, y_train, y_test =\
    train_test_split(X, y,
                     test_size=0.3,
                     random_state=0,
                     stratify=y)
```

```
from sklearn.preprocessing import StandardScaler


sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(penalty='l1', C=1.0, solver='liblinear', multi_class='ovr')
# Note that C=1.0 is the default. You can increase
# or decrease it to make the regulariztion effect
# stronger or weaker, respectively.
lr.fit(X_train_std, y_train)
print('Training accuracy:', lr.score(X_train_std, y_train))
print('Test accuracy:', lr.score(X_test_std, y_test))
```

```
Training accuracy: 1.0
Test accuracy: 1.0
```

```
lr.intercept_
```

```
array([-1.26363107, -1.21610924, -2.37035486])
```

```
np.set_printoptions(8)
```

```
lr.coef_[lr.coef_!=0].shape
```

```
(23,)
```

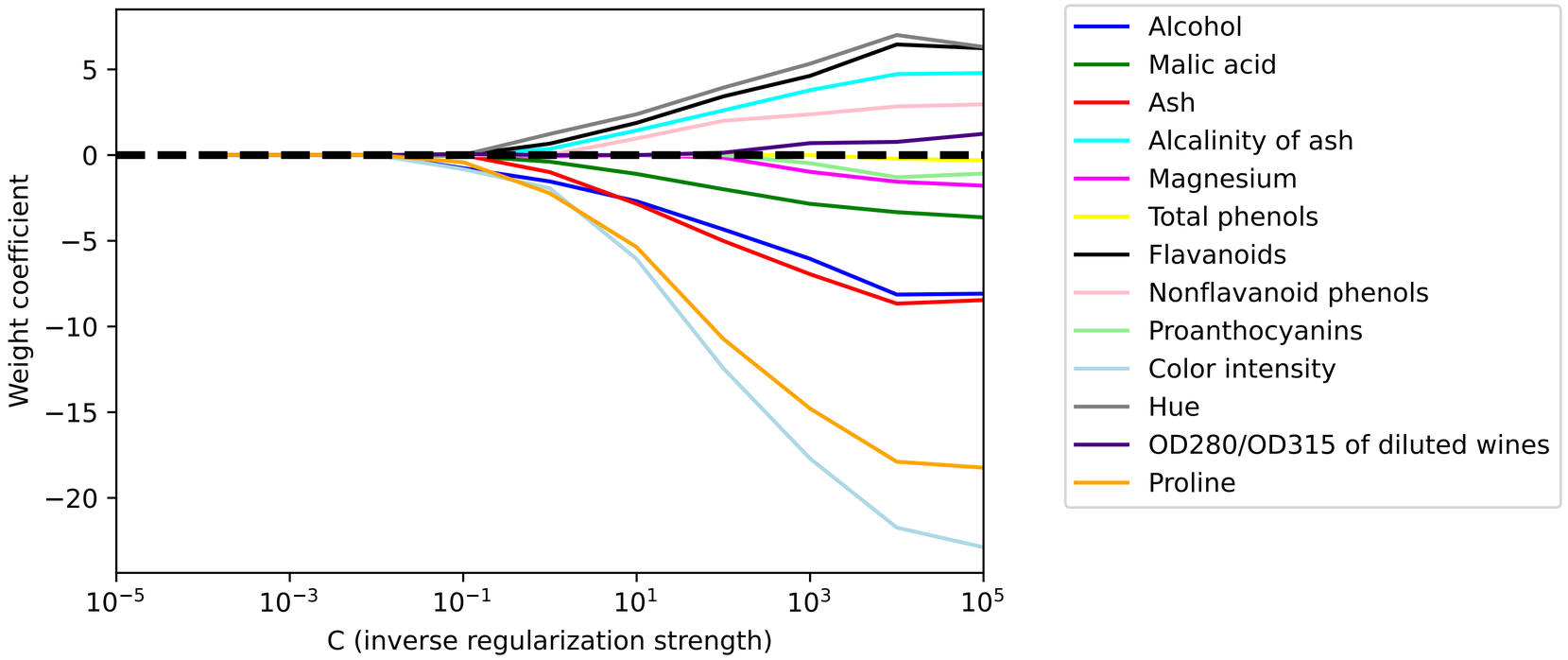
```
lr.coef_
```

```
array([[ 1.2455466 ,  0.18072432,  0.74213192, -1.15948629,  0.        ,
         0.        ,  1.17434899,  0.        ,  0.        ,  0.        ,
         0.        ,  0.54353185,  2.51127873],
       [-1.53786975, -0.38667962, -0.9954337 ,  0.36456123, -0.05923747,
         0.        ,  0.66763266,  0.        ,  0.        , -1.93321837,
         1.23529768,  0.        , -2.23229101],
       [ 0.13570425,  0.16821283,  0.35724291,  0.        ,  0.        ,
         0.        , -2.43809231,  0.        ,  0.        ,  1.56377541,
        -0.81940109, -0.49234846,  0.        ]])
```

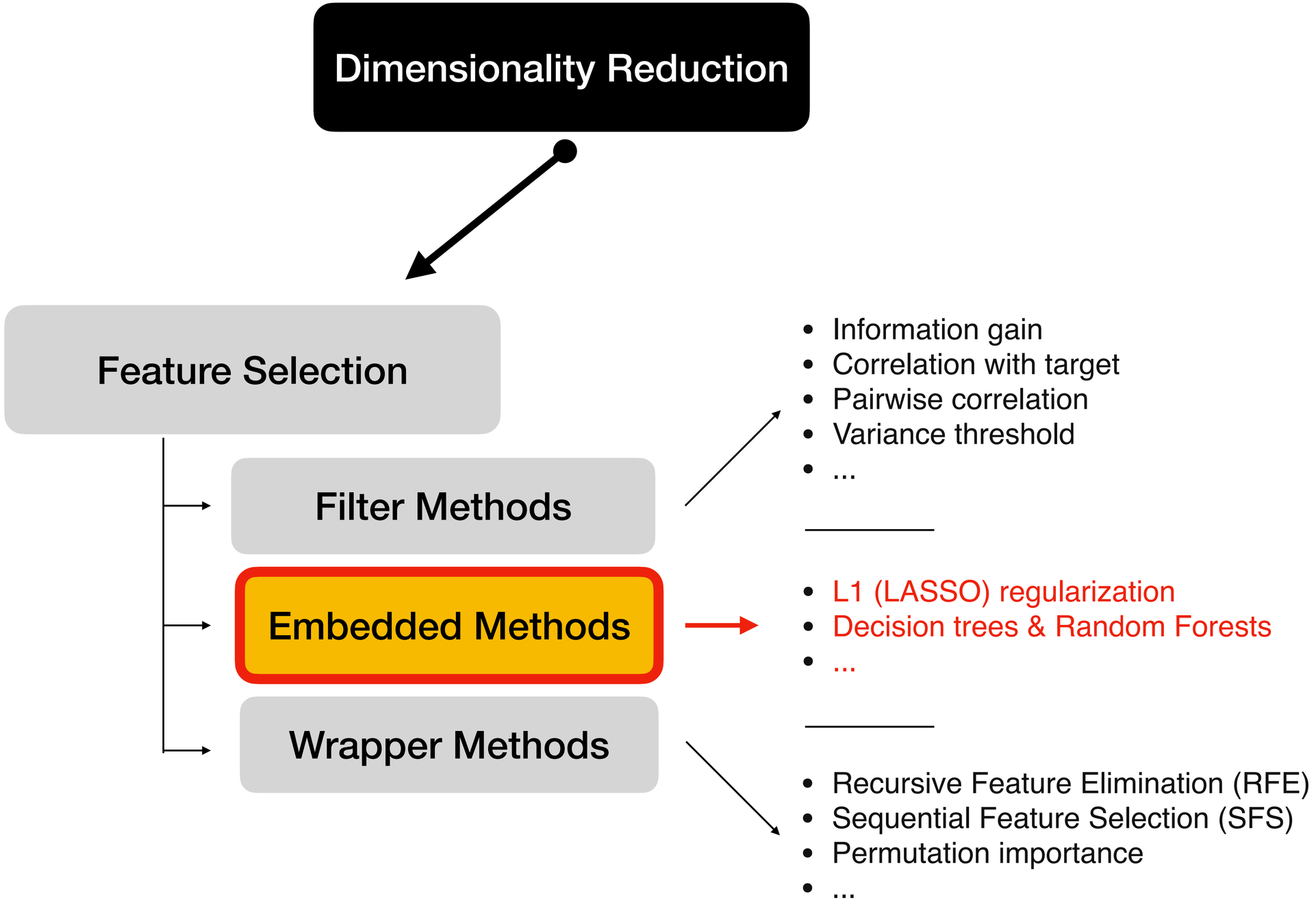
L1 Regularization / LASSO (Embedded)

Least Absolute Shrinkage and Selection Operator

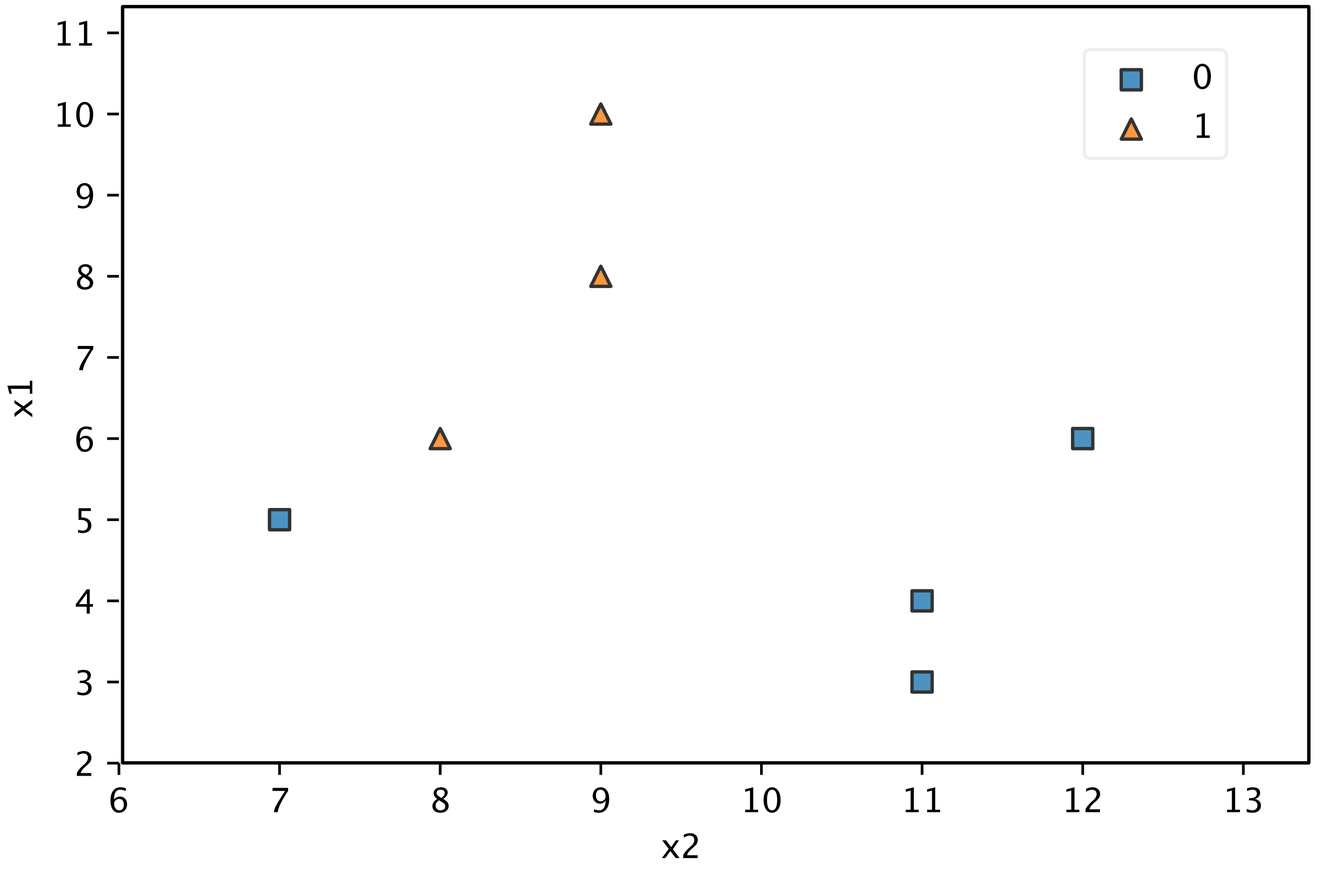
LASSO Path



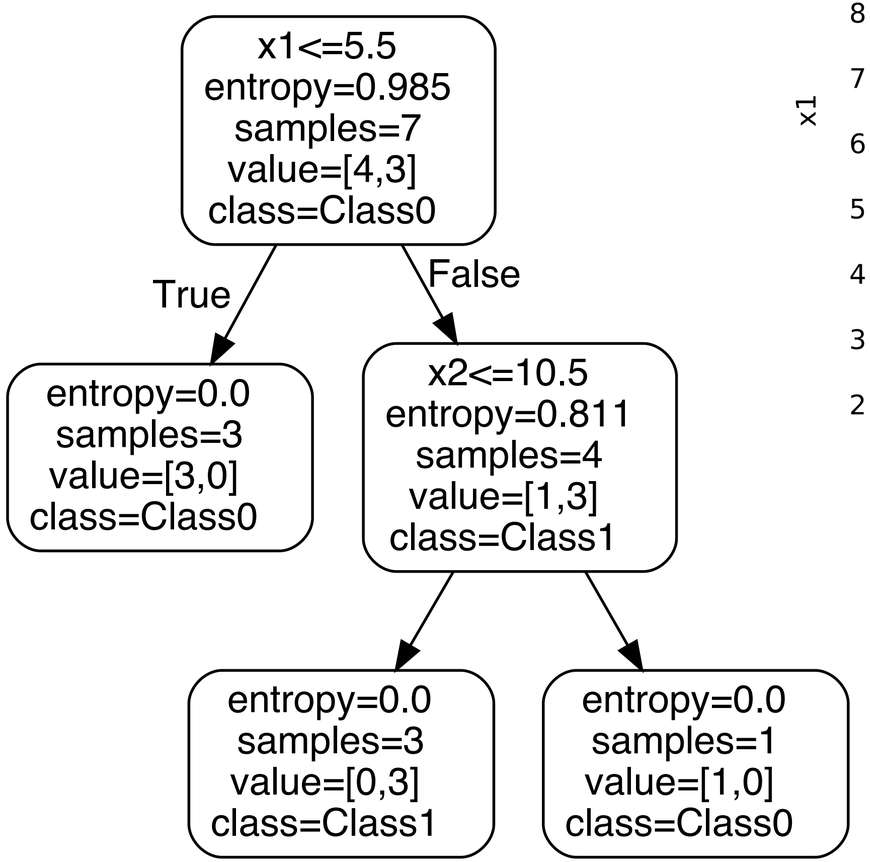
1. Different categories of feature selection
2. Filter methods
3. Embedded methods
    3.1. L1-regularized logistic regression
    3.2. Decision trees & random forest feature importance
4. Wrapper methods
    4.1. Recursive feature elimination
    4.2. Permutation importance
    4.3. Permutation importance code example
    4.4. Sequential feature selection
    4.5. Sequential feature selection code example

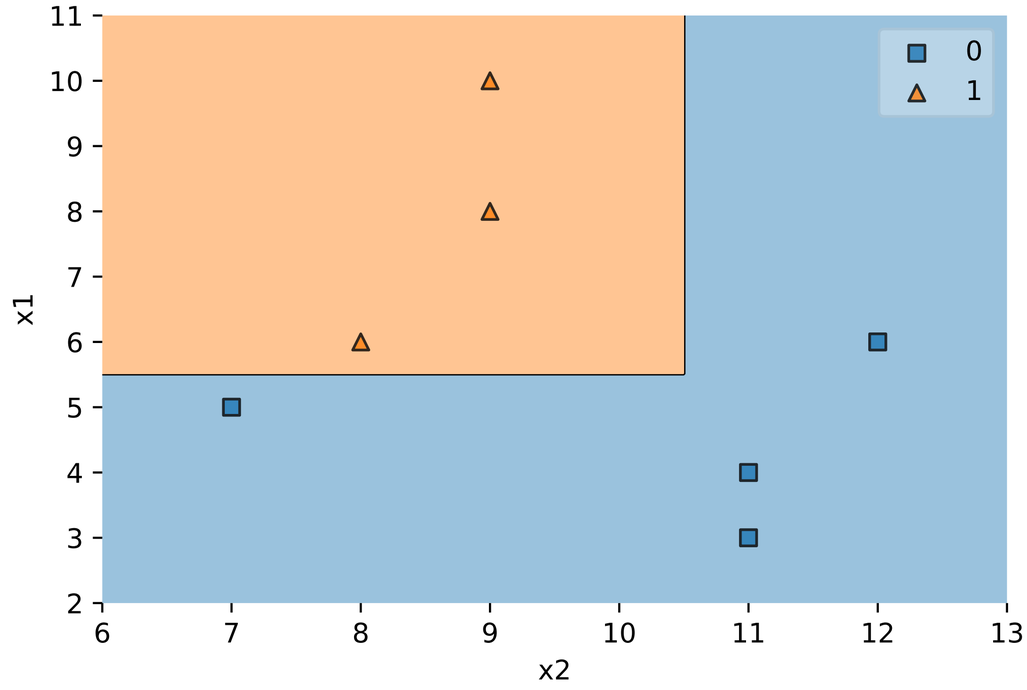

Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
• Information gain
• Correlation with target
• Pairwise correlation
• Variance threshold
• ...
• L1 (LASSO) regularization
• Decision trees & Random Forests
• ...
• Recursive Feature Elimination (RFE)
• Sequential Feature Selection (SFS)
• Permutation importance
• ...

Feature Selection in Decision Trees (1)

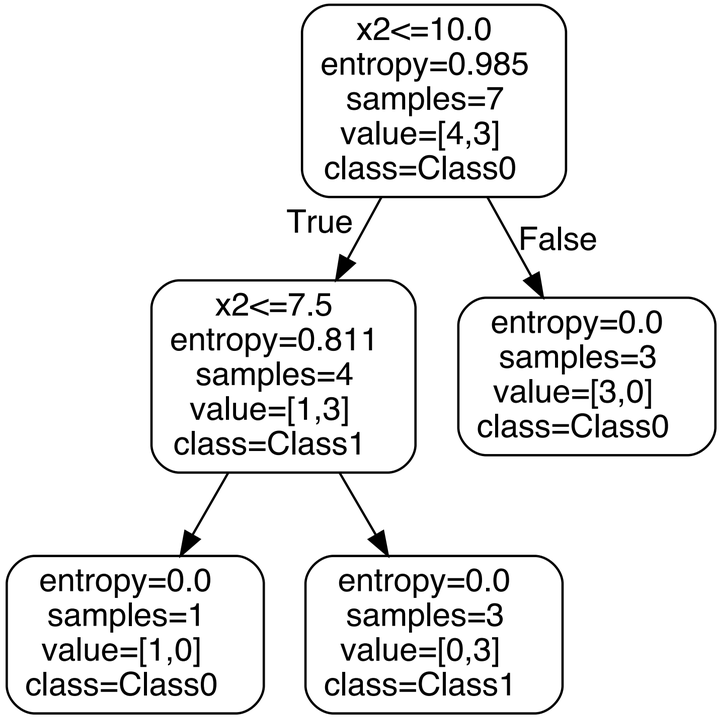


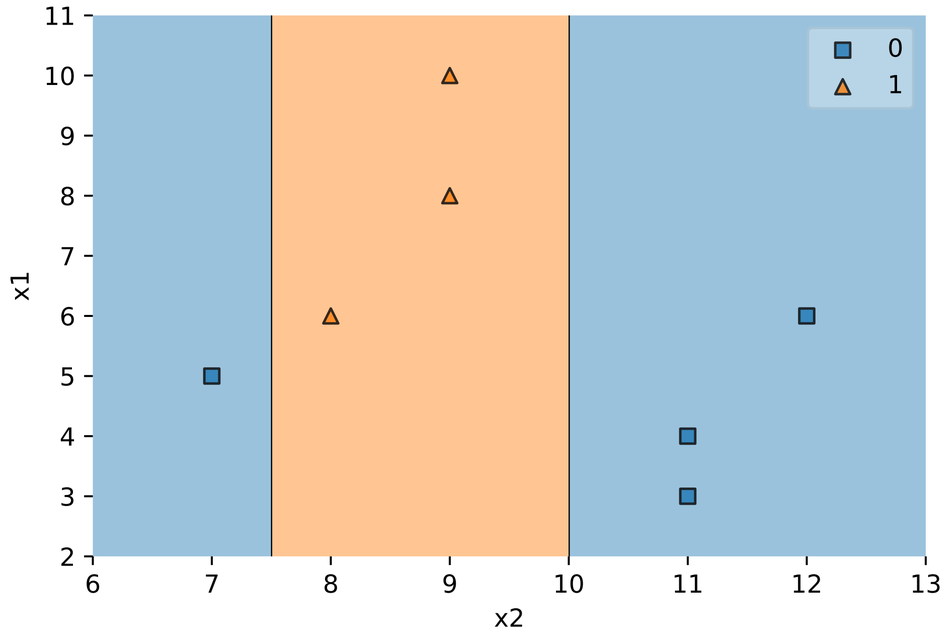
Feature Selection in Decision Trees (2)





Feature Selection in Decision Trees (3)



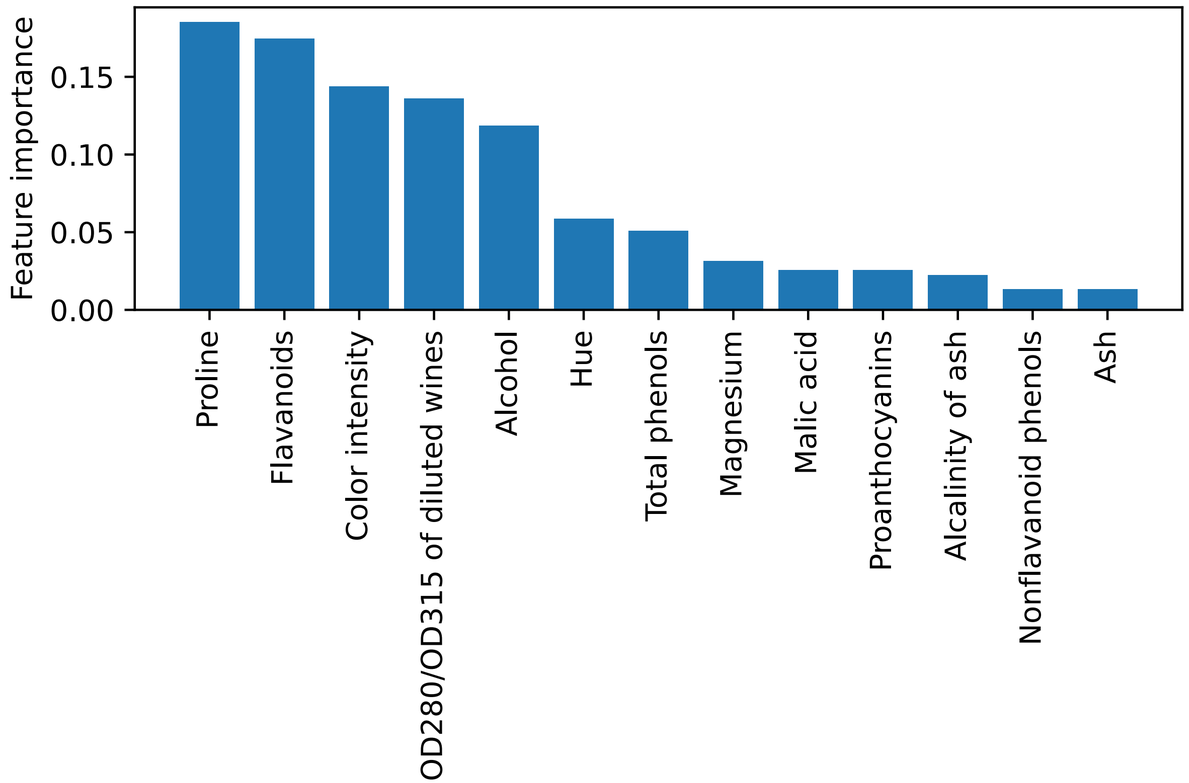


```
import pandas as pd
import numpy as np


df_wine = pd.read_csv('https://archive.ics.uci.edu/'
                      'ml/machine-learning-databases/wine/wine.data',
                      header=None)


df_wine.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
                   'Alcalinity of ash', 'Magnesium', 'Total phenols',
                   'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
                   'Color intensity', 'Hue', 'OD280/OD315 of diluted wines',
                   'Proline']

print('Class labels', np.unique(df_wine['Class label']))
df_wine.head()
```

```
Class labels [1 2 3]
```

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

```
from sklearn.ensemble import RandomForestClassifier



forest = RandomForestClassifier(n_estimators=500,
                                random_state=1)

forest.fit(X_train, y_train)
importances = forest.feature_importances_

indices = np.argsort(importances)[::-1]

plt.ylabel('Feature importance')
plt.bar(range(X_train.shape[1]),
        importances[indices],
        align='center')

feat_labels = df_wine.columns[1:]
plt.xticks(range(X_train.shape[1]),
           feat_labels[indices], rotation=90)

plt.xlim([-1, X_train.shape[1]])
```



```
forest.estimators_
```

```
[DecisionTreeClassifier(max_features='auto', random_state=1791095845),
 DecisionTreeClassifier(max_features='auto', random_state=2135392491),
 DecisionTreeClassifier(max_features='auto', random_state=946286476),
 DecisionTreeClassifier(max_features='auto', random_state=1857819720),
 DecisionTreeClassifier(max_features='auto', random_state=491263),
 DecisionTreeClassifier(max_features='auto', random_state=550290313),
 DecisionTreeClassifier(max_features='auto', random_state=1298508491),
 DecisionTreeClassifier(max_features='auto', random_state=2143362693),
 DecisionTreeClassifier(max_features='auto', random_state=630311759),
 DecisionTreeClassifier(max_features='auto', random_state=1013994432),
 DecisionTreeClassifier(max_features='auto', random_state=396591248),
 DecisionTreeClassifier(max_features='auto', random_state=1703301249),
```

max_features : *{"auto", "sqrt", "log2"}, int or float, default="auto"*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `round(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=sqrt(n_features)`.
- If "sqrt", then `max_features=sqrt(n_features)` (same as "auto").
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Random Forest Feature Importance

Method A: Impurity-based feature importance

(this is used in scikit-learn)

Usually measured as follows:

- for a given feature
  - for each tree
    - compute impurity decrease (Gini, Entropy)
    - weight by number of examples at that node
    - averaged over all trees
- normalize importances so that sum of feature importances sum to 1

Random Forest Feature Importance

Caveats

- Impurity-based feature importance are inflated for categorical features with lots of unique values (we will cover permutation-based performance later, which addresses this)
- Correlated features share importance

Bootstrap Sampling

Random Forest Feature Importance

Method B: Permutation Importance

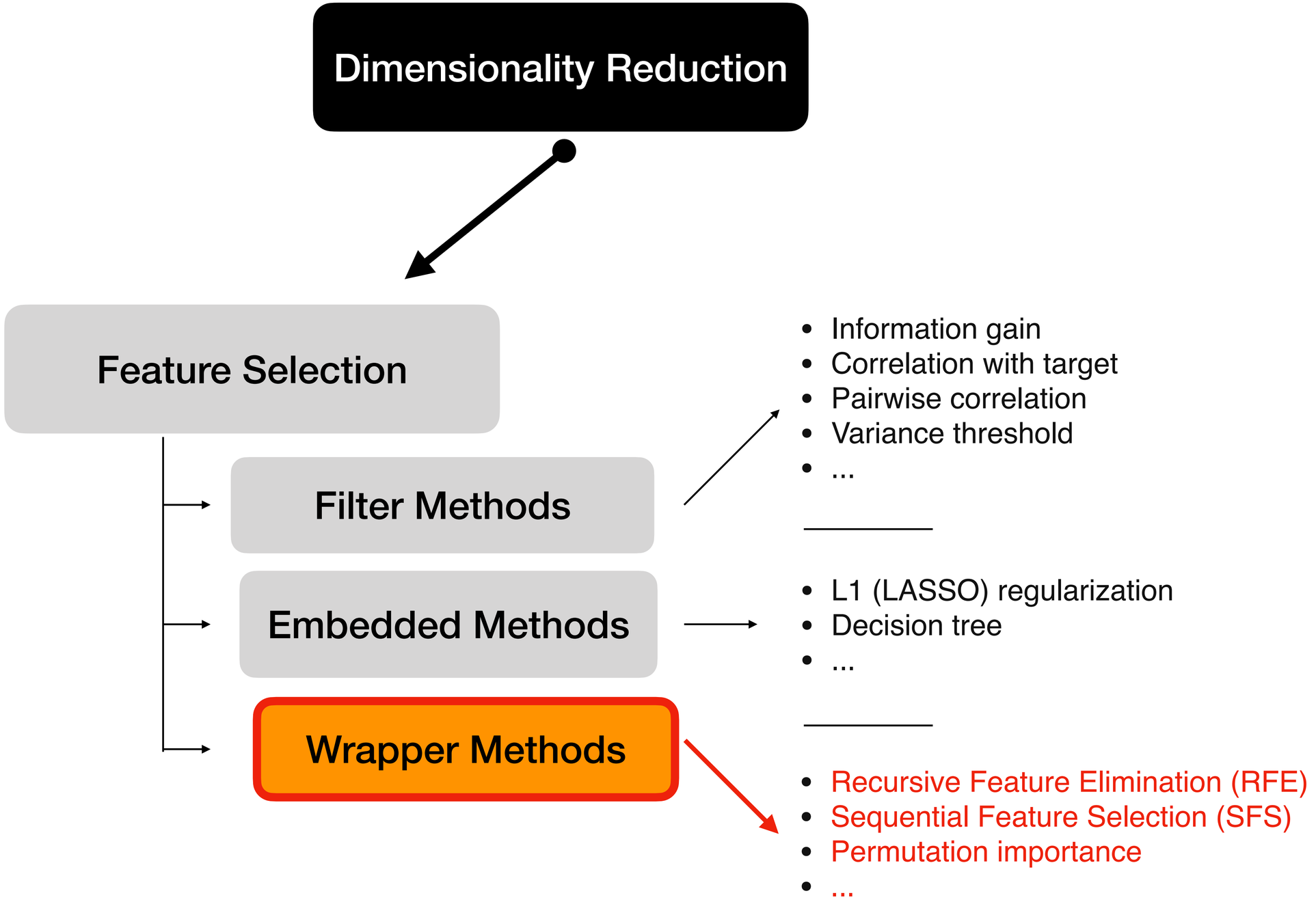
Out-of-bag accuracy:

- During training, for each tree, make prediction for OOB sample (~1/3 of the training data)
- Based on those predictions where example *i* was OOB, compute label via majority vote among the trees that did not use example *i* during model fitting
- The proportion over all examples where the prediction (by majority vote) is correct is the OOB accuracy estimate

Out-of-bag feature importance via permutation:

(we will also cover a generalized version with a hold out set later)

- Count votes for correct class
- Given feature *i,* permute this feature in OOB examples of a tree
- Compute the number of correct votes after permutation from the number of votes before permutation for given tree
- Repeat for all trees in the random forest and average the importance
- Repeat for other features


Dimensionality Reduction
Feature Selection
Filter Methods
Embedded Methods
Wrapper Methods
• Information gain
• Correlation with target
• Pairwise correlation
• Variance threshold
• ...
• L1 (LASSO) regularization
• Decision tree
• ...
• Recursive Feature Elimination (RFE)
• Sequential Feature Selection (SFS)
• Permutation importance
• ...