Machine Learning for biology

V. Monbet

UFR de Mathématiques
Université de Rennes 1

Outline

Context

Data:

- "Responses":
  Regression Case: $y_i \in \mathbb{R}$ for $i = 1, \cdots, n$
  Classification Case: $y_i \in C$ for $i = 1, \cdots, n$ and $C = \{1, \cdots, k\}$
- "Explanatory Variables" or "Predictors," $x_i \in \mathbb{R}^p$ for $i = 1, \cdots, n$.
- Each pair (y_i, x_i) represents an experiment (individual).

Problem:

- Explain Y in terms of X.

Principle:

- Sample assumed to be iid: $(Y_1, X_1), (Y_2, X_2), \cdots, (Y_n, X_n)$
  $\rightarrow$ estimated model $Y \simeq \hat{f}(X)$.

Examples:

- Y: spam/non-spam, X: sender's address type, number of addresses, title word frequency, message word frequency, message word count, etc.
- Y: normal connection/attack connection, X: connection time, connection hour, click history, etc.

Bias-Variance Trade-off

Data:

- "Responses": y_i for $i = 1, \cdots, n$
- "Explanatory Variables" or "Predictors," $x_i \in \mathbb{R}^p$ for $i = 1, \cdots, n$.

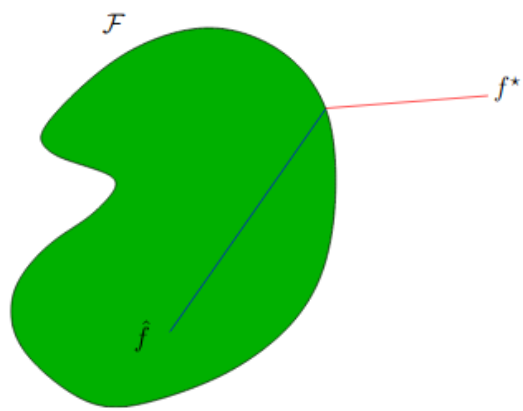
Problem:

- Estimate
$$f^* \in \arg\min_{f \in \mathcal{F}} \mathcal{R}(f) = E\left(\ell(Y, f(X))\right)$$
by $\hat{f}_n$ where the index n indicates that $\hat{f}_n$ is constructed from the available n data.
- Example of linear regression:
  - $\mathcal{F}$: set of linear mappings from $\mathbb{R}^p \to \mathbb{R}$
  - Loss function: $\ell(Y, f(X)) = (Y - f(x))^2$

Modeling means specifying a class of functions $\mathcal{F}$ and assuming that $f^* \in \mathcal{F}$

Errors:

$$\mathcal{R}(\hat{f}) - \mathcal{R}^* = \underbrace{\mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}}}_{\textit{estimation}} + \underbrace{\inf_{f \in \mathcal{F}} - \mathcal{R}^*}_{\textit{approximation}}$$



These two errors vary in opposite directions, and the goal of machine learning is to find the right compromise between the two terms.

Parametric vs. Non-parametric Approaches

In parametric approaches, such as:

- Multiple linear regression
- Discriminant analysis, logistic regression

strong assumptions are made about the form of the model f (i.e., about the set $\mathcal{F}$). The model depends on a parameter vector β, and we optimize a global criterion to estimate f

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} ||y_i - f(X_i; \beta)||^2$$

$\rightarrow$ low estimation error and high approximation error.

Modeling means specifying a class of functions $\mathcal{F}$ and assuming that $f^* \in \mathcal{F}$

In non-parametric[1] approaches, restrictions are weaker; the set $\mathcal{F}$ is broader. They involve considering observations x_i that are in the neighborhood of the point of interest x_0 to estimate

$$\hat{y}_0 = \hat{f}(x_0)$$

in other words, a local criterion is considered.
$\rightarrow$ high estimation error and low approximation error.

Examples of non-parametric methods:

- k-Nearest Neighbors Algorithm
- Decision Trees

[1] Data-Driven

Outline

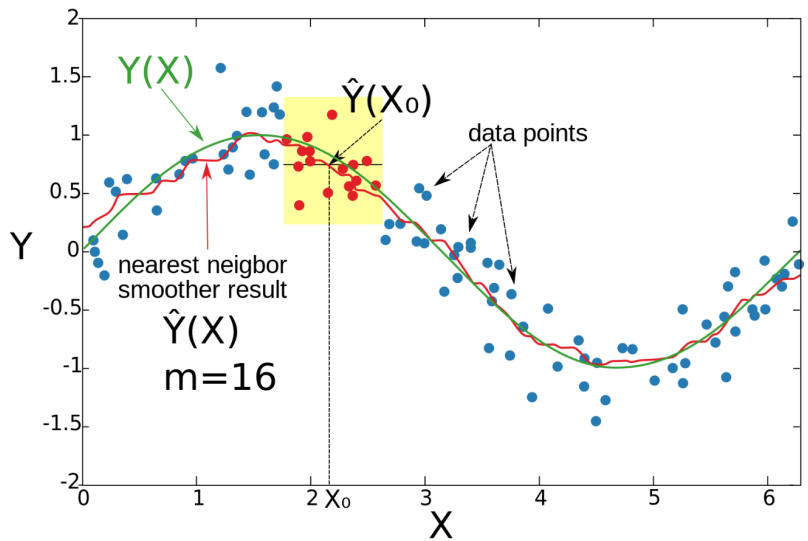
Principle of the k-NN Algorithm

The prediction associated with a new individual $\mathbf{x}_0 \in \mathcal{X}$ is obtained by searching the training set for the k instances that are closest to $\mathbf{x}_0$ and calculating the average of their corresponding responses.

$$\hat{Y}(\mathbf{x}_0) = \frac{1}{k} \sum_{i|\mathbf{x}_i \in V(\mathbf{x}_0)} y_i$$

where $V(x_0)$ is the neighborhood of x_0, i.e.,

$$V(x_0) = \{x \in \mathcal{X} | \text{dist}(x, x_0) < \text{dist}(x_{(k)}, x_0)\}$$



source: Wikipedia

Choosing Distances to Find Neighbors

To find the k nearest neighbors of x_0, we need to choose a distance metric.

- If all explanatory variables are quantitative, the usual distance metric is the Euclidean distance.

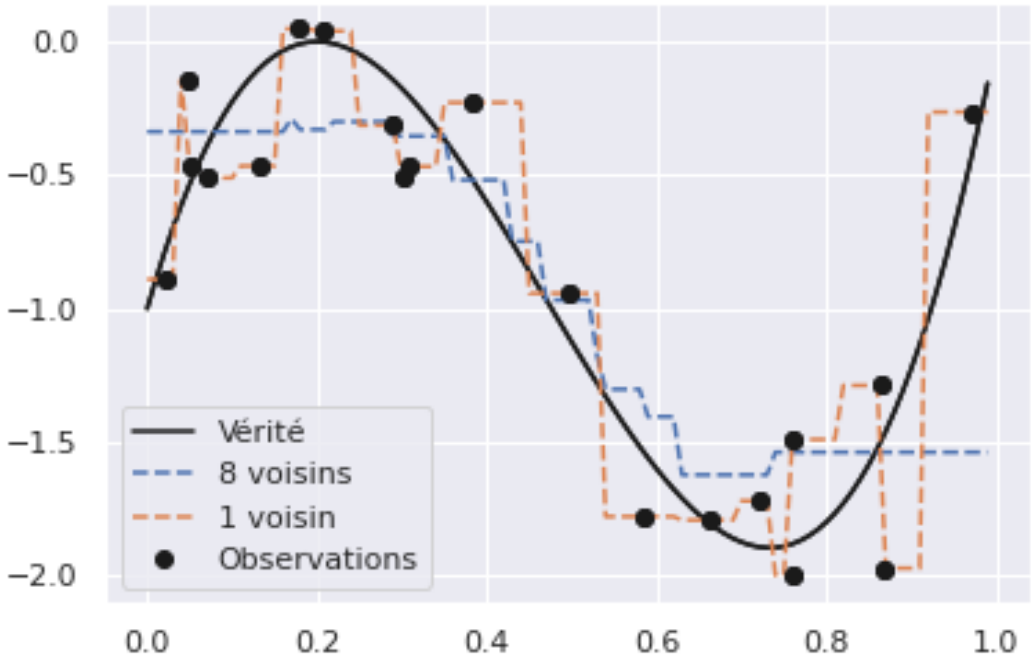
$$\text{dist}(\mathbf{x}_{i'}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{p} (x_{i'j} - x_{ij})^2}$$

  It's important to normalize the data so that all variables are on the same scale.
- If some of the explanatory variables are qualitative, they can be transformed into "dummy" variables, and the Euclidean distance can be used again.
- If all the data are qualitative, there are specific distance metrics like the chi-squared distance, Jaccard distance, etc.

Selection of the Number of Neighbors - Example

We estimate two models, one with 1 neighbor and the other with 8 neighbors, on simulated data:

$$y_i = (5x_i - 1)^2(x_i - 1) + u_i, \quad u_i \sim \mathcal{N}(0, 0.25)$$



- The smaller the number of neighbors, the higher the risk of overfitting. Estimation error (variance) dominates approximation error (bias).
- The larger the number of neighbors, the smoother the curve. Approximation error dominates estimation error.

Cross-Validation

In practice, we estimate the optimal number of neighbors using **K-fold cross-validation** or leave-one-out.

Repeat s times,

1. Randomly split the sample $(Z_1, \cdots, Z_n)$ into two sets: T (test) of size n/K, and A (training), the complement of T.
2. Estimate the model on A for each number of neighbors k.
3. Calculate the prediction error on T for each k

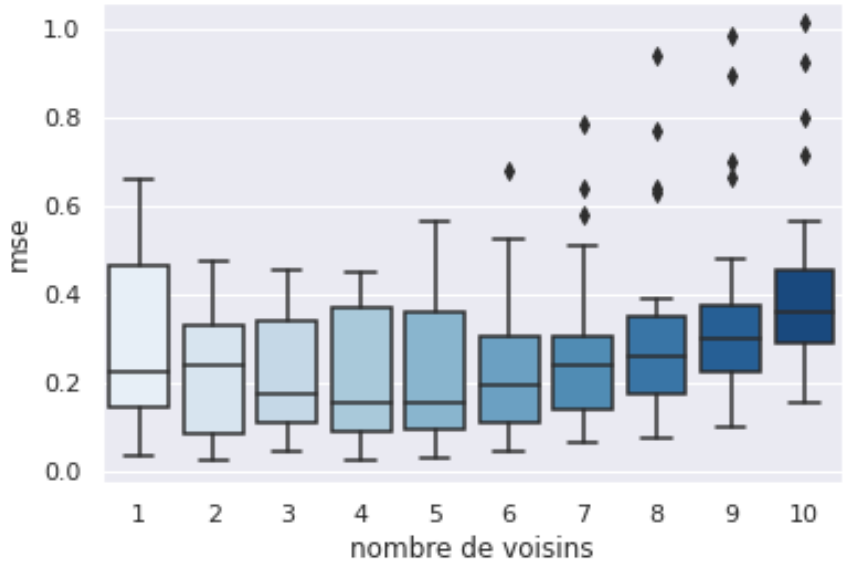
$$E(s,k) = \frac{K}{n} \sum_{i \in T} (\hat{y}_i(k) - y_i)^2$$

Estimate the average error for each value of k

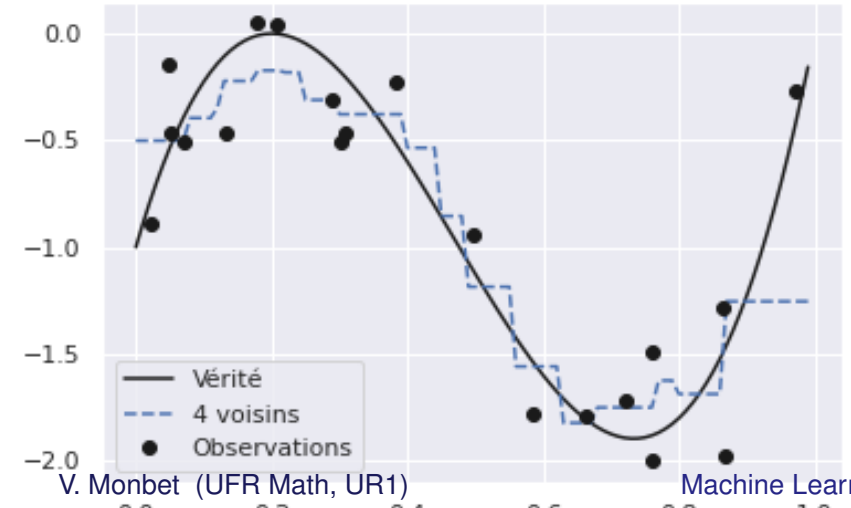
$$\widehat{MSE} = \frac{1}{S} \sum_{s} E(s,k)$$

Example

$$y_i = (5x_i - 1)^2(x_i - 1) + u_i, \quad u_i \sim \mathcal{N}(0, 0.25)$$



We repeat the estimation using the k-NN method for k ranging from 1 to 10, $S = 50$ times.
The boxplots in the top figure represent the distribution of $\widehat{MSE}$.



The graph suggests retaining 3 or 4 neighbors. The estimation in the bottom figure corresponds to this choice.

One of the drawbacks of the k-Nearest Neighbors estimator is that it requires searching for neighbors. Even though there are algorithms for fast searches, the computational cost can still be significant.

Of course, there are other non-parametric approaches: kernel estimator, decision trees, etc.

Outline

Principle of Kernel Estimation

The prediction associated with a new individual $\mathbf{x}_0 \in \mathcal{X}$ is obtained by calculating a weighted average of all the responses in the training set.

$$\hat{Y}(\mathbf{x}_0) = \frac{1}{n} \sum_{i=1}^{n} w(x_0, x_i) y_i$$

where the kernel w gives stronger weights to points that are close to x_0.

Examples of kernels:

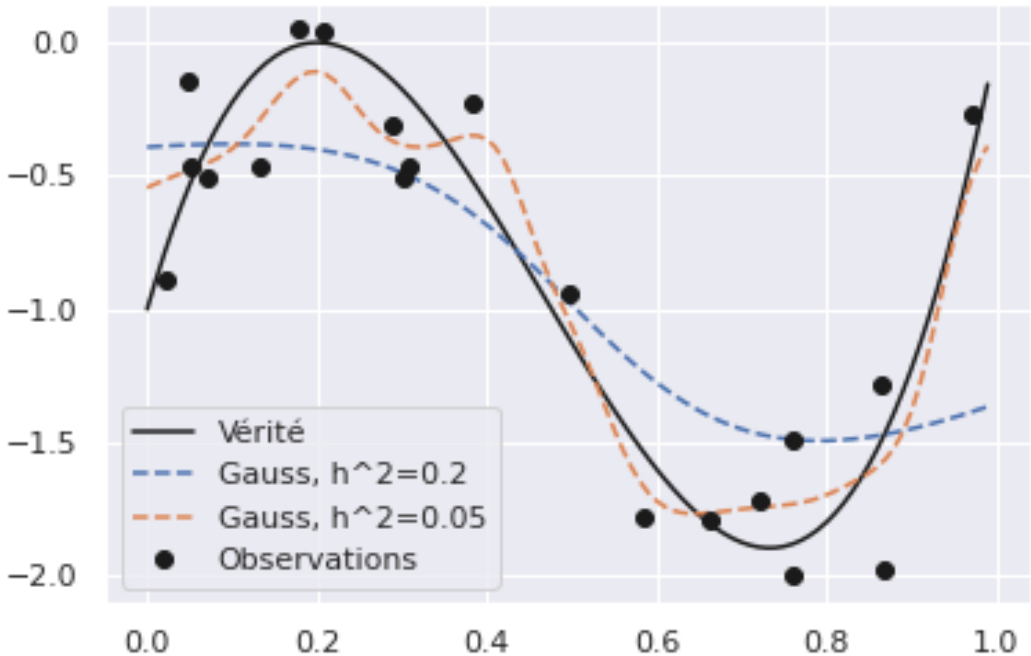
- Uniform kernel: $K(x_0, x) = \frac{1}{h} \mathbb{I}_{[x_0 - h/2, x_0 + h/2]}(x)$
  This is the histogram kernel.
- Gaussian (radial basis): $K(x_0, x) = \frac{1}{\sqrt{2\pi h^2}} \exp\left(- \frac{(x_0 - x)^2}{h^2} \right)$
- Tri-cube: $(1 - |x_0 - x|^3)^3 \mathbb{I}_{|x_0 - x| < h}$

The parameter h, called the bandwidth, controls the smoothness of the estimate.

Example

We estimate two models, one with $h^2 = 0.2$ and the other with $h^2 = 0.05$, on simulated data:

$$y_i = (5x_i - 1)^2(x_i - 1) + u_i, \quad u_i \sim \mathcal{N}(0, 0.25)$$



The choice of bandwidth h is crucial,

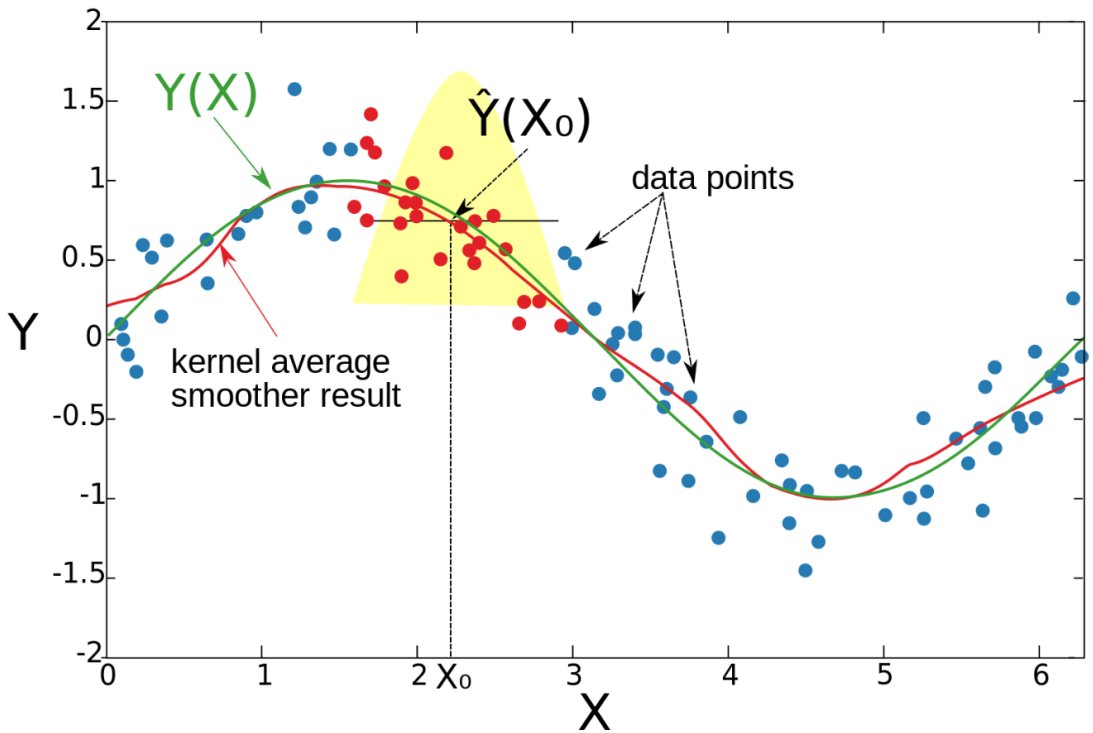
- Small h reduces bias but can lead to high variance,
- Large h controls variance but increases bias.

The estimate is smoother than that of the k-Nearest Neighbors method. However, computation time can become significant if n is large.

k-NN and Kernel

In machine learning, it's common to combine both methods to leverage their respective advantages,

- We select the k nearest neighbors,
- We calculate a weighted average to give more weight to neighbors closer to x_0.



Near Infrared Data

Predict the fat concentration in cookies from NIR spectra.
`https://drive.google.com/file/d/1vd4WVrmif5Sfy9VNacvzGw3D-4T5h_-L/view?usp=sharing`
using scikit learn (python)

Outline

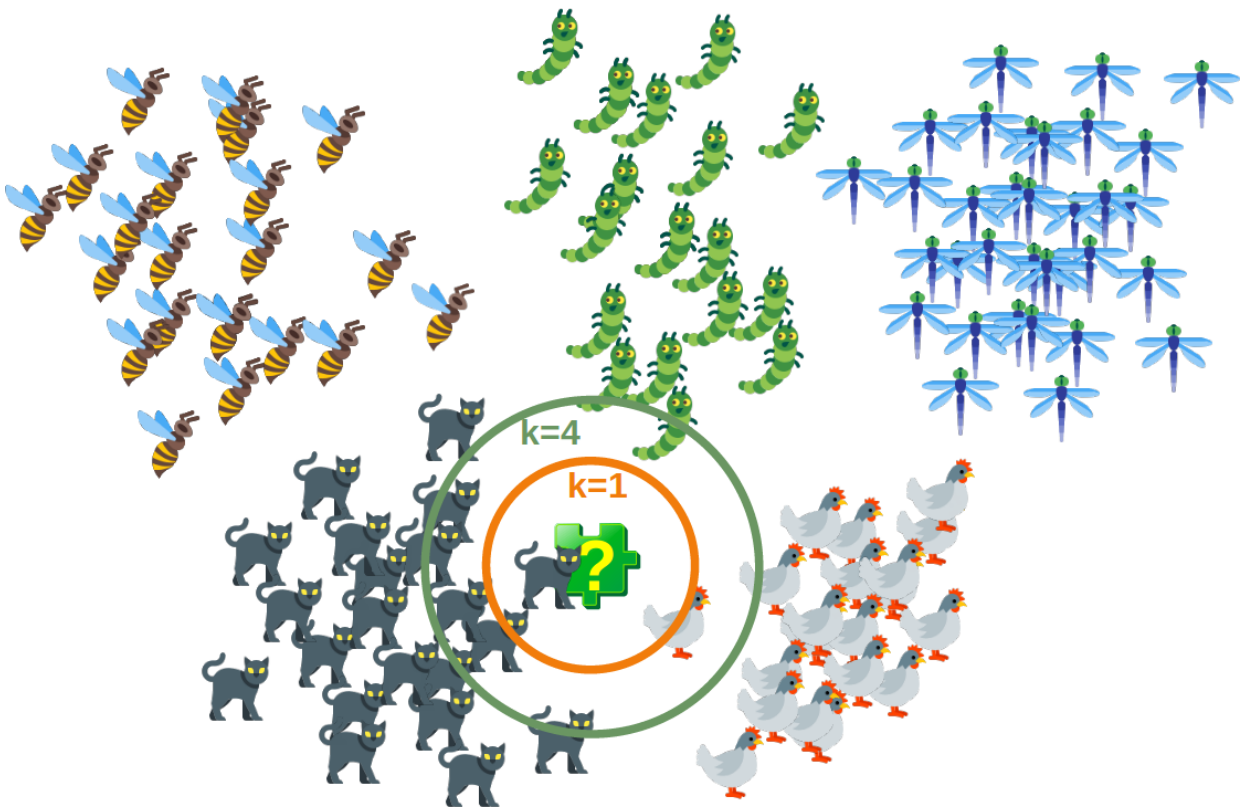
k-Nearest Neighbors for Classification

$$y_i \in C \text{ for } i = 1, \cdots, n \text{ and } C = \{1, \cdots, L\}$$

In a classification framework, the k-Nearest Neighbors method is similar to the regression case.

- We select the k nearest neighbors as before.
- We estimate the response by majority voting:

$$\hat{y}_0 = \arg \max_{\ell \in \{1, \cdots, L\}} \sum_{i \in V(x_0)} \mathbb{I}_{y_i = \ell}.$$



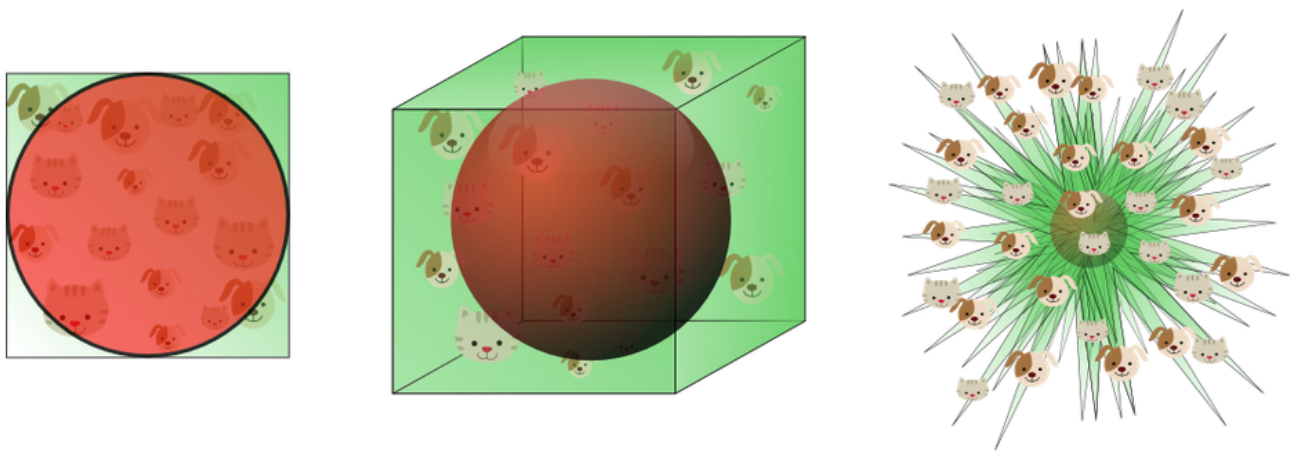
Outline

Conclusion/discussion

- Non-parametric methods are easy to implement as they mainly rely on the distance between observations.
- The choice of hyperparameters (number of neighbors, bandwidth) is crucial because it allows finding the right balance between bias and variance.
- The k-Nearest Neighbors and kernel estimation methods lead to models that are not interpretable: they do not help in understanding the underlying phenomena.

Curse of dimansionality

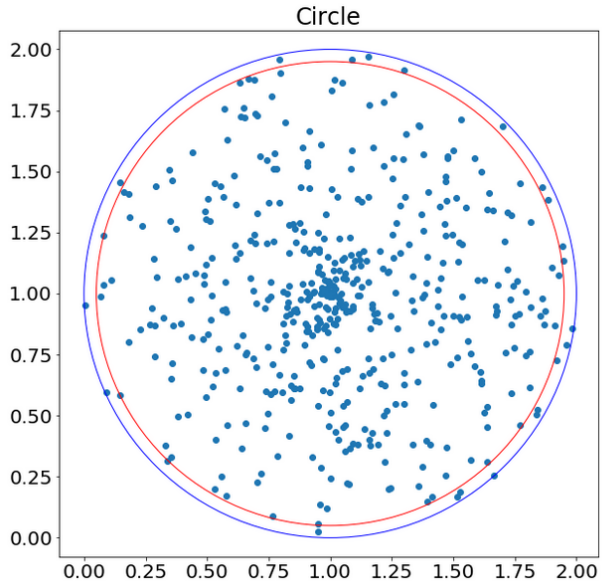
- Data driven methods always suffer of the curse of dimensionality : when the dimension increases, the number of points in the neiborhood of x_0 decreases as well as the rate of convergence (ie more observations are required for a given prediction qaulity).

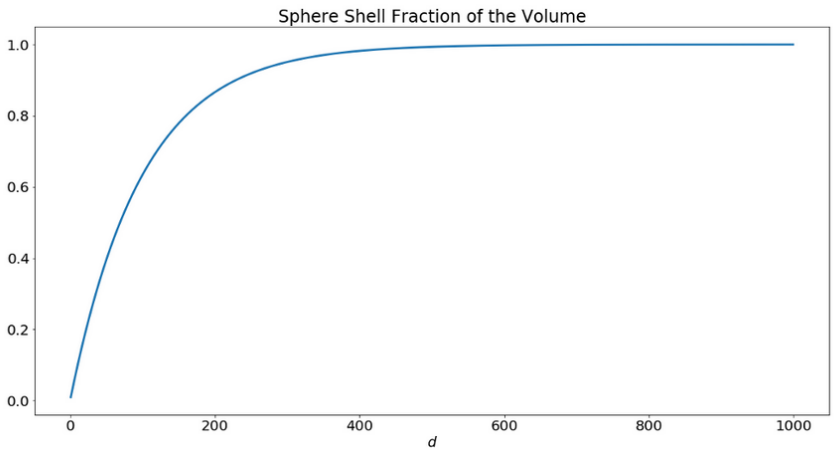


https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

Example of a ball

Voume of the sphere in $\mathbb{R}^d$: $V = cr^d$.





Large circle $r_o = 1$,
Small circle $r_i = .95$

$V = cr_o^d = c$,
$V_{\text{shell}} = c - cr_i^d$,

Volume of the margin of the sphere:

$$\frac{V_{\text{shell}}}{V} = 1 - r_i^d$$

When the dimension increases, most points belong to the 'edge' fraction of the sphere. The volume inside the small sphere is almost empty...

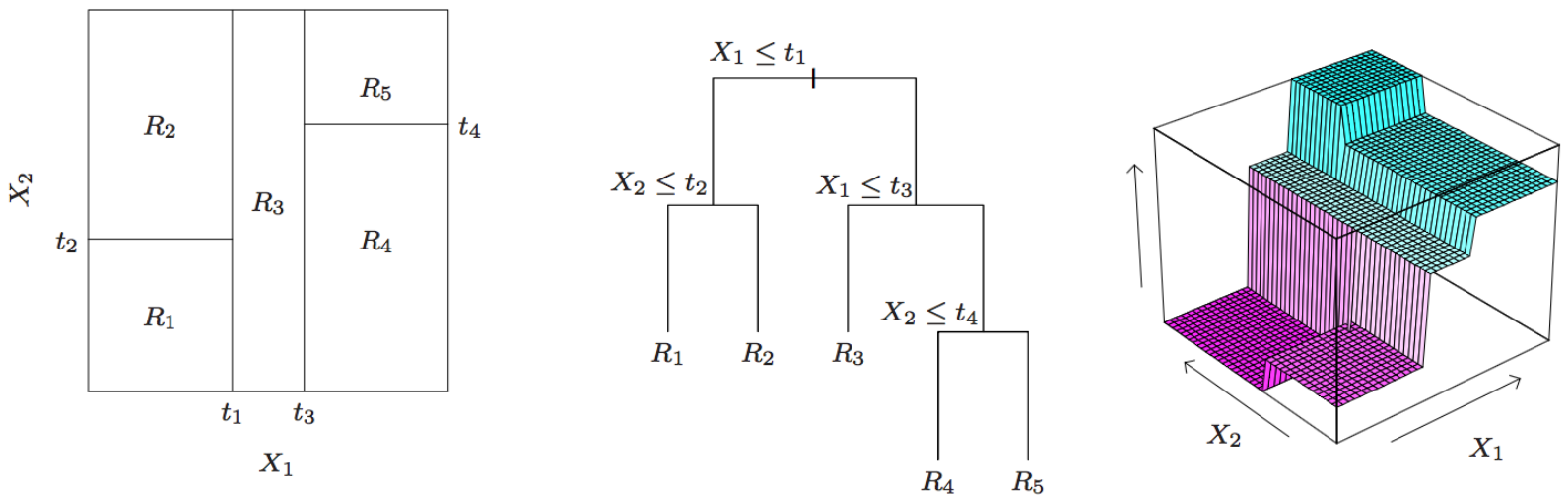
It is difficult to find good neighbors in this case!

Outline

Principle of Decision Trees

- The principle of decision tree algorithms is to partition the space into rectangles and fit a (very) simple model in each region.
- The concept is simple but often effective.
- For example, consider a regression problem to predict a continuous response $Y \in \mathcal{Y}$ from 2 inputs X_1 and X_2, each taking values in the unit interval. In each cell R_m, Y is predicted by the mean value c_m of the observed values in the cell.

$$\hat{f}(x) = \sum_{m=1}^{M} c_m \mathbb{I}_{R_m}(x_1, x_2)$$

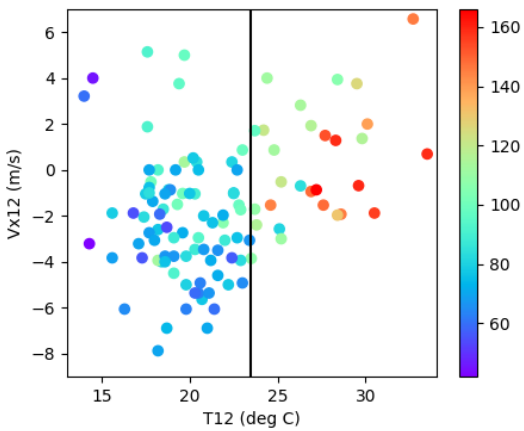


Figures from Hastie's book.

Outline

Building a decision tree: greedy algorithm

- Find the partition which minimize the mean saquare error $\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2$, with f a decision tree, is numerically too expensive.
- Trees are estimated by an iterative algorithm which builds nested partitions.
- At each iteration, the algorithm selects the variable and threshold that result in a new subdivision for a subregion of the partition.
  Splitting Criterion: We seek a division such that the weighted sum of variances of the child regions is less than the variance of the parent region.
- Example: Predicting the daily maximum ozone.



Variance before dividing: 794
Variances mean after dividing: 299

- When the final partition is obtained, the regression function is

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{I}_{R_m}(\mathbf{x}) \text{ avec } c_m = \sum_{\{i|\mathbf{x}_i \in R_m\}} y_i$$

Algorithme

- Considering all the data within the region R_m, a splitting variable x_j, and a threshold s, we define the subregions

$$R_{m1}(j,s) = \{\mathbf{x} \in R_m | x_j \leq s\} \text{ and } R_{m2}(j,s) = \{\mathbf{x} \in R_m | x_j > s\}$$

- The variable x_{j^*} and the threshold s^* are searched such that

$$(j^*, s^*) = arg \min_{j,s} \left(\sum_{\{i|\mathbf{x}_i \in R_{m1}(j,s)\}} (y_i - c_1)^2 + \sum_{\{i|\mathbf{x}_i \in R_{m2}(j,s)\}} (y_i - c_2)^2 \right).$$

with

$$c_1 = \sum_{\{i|\mathbf{x}_i \in R_1\}} y_i \text{ et } c_2 = \sum_{\{i|\mathbf{x}_i \in R_2\}} y_i$$

- For each variable, the choice of threshold s is (numerically) very fast, and the selection of the best pair (j, s) is feasible.

Stopping Criteria
The divisions are repeated until at least one stopping criterion is met.

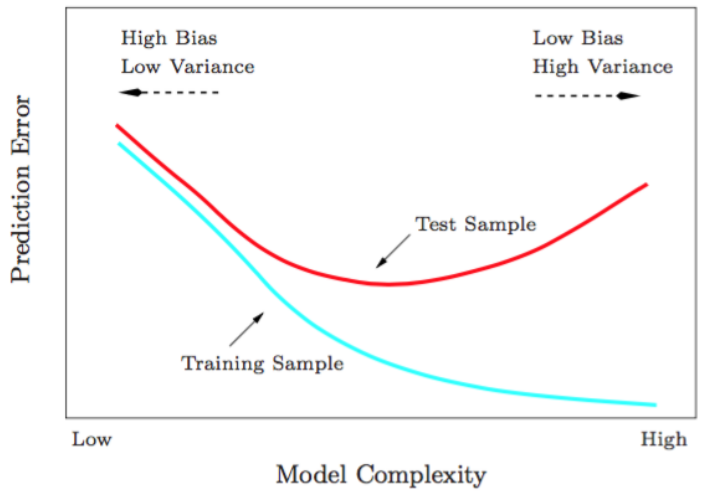
The classical stopping criteria are:

- Minimum number of observations to perform a split: If the number of observations is low, the estimation of variances may be biased and noisy.
- Minimum number of observations per leaf: If the number of observations is low, the prediction of the target variable will be imprecise (high estimation variance).
- Maximum depth: A tree that is too deep can lead to overfitting.

An alternative approach is to consider a validation set for cross-validation. In this case, the algorithm is stopped when the prediction error on the validation set no longer decreases significantly.

Another way to frame the question: What is the right size for a decision tree?

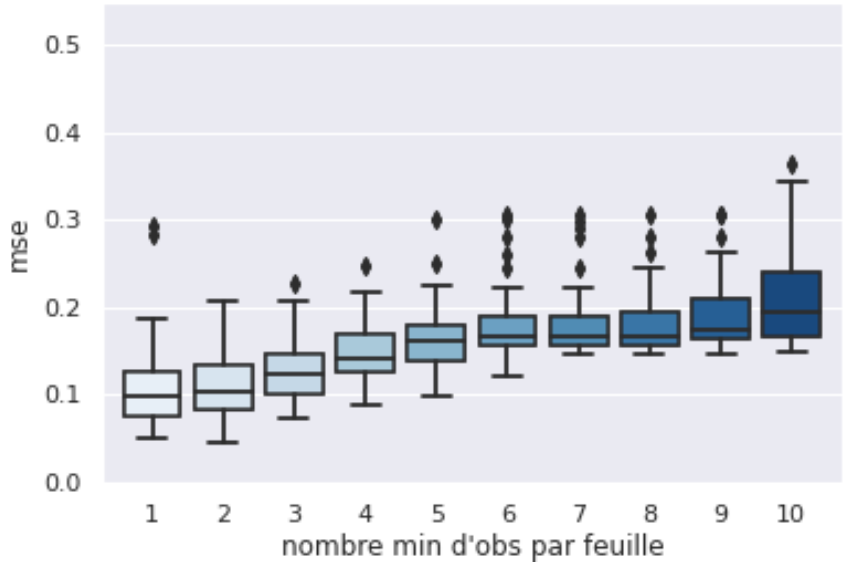
- Bias-Variance Trade-off: A deep tree may risk overfitting, while a tree that is too small may not capture important structures."



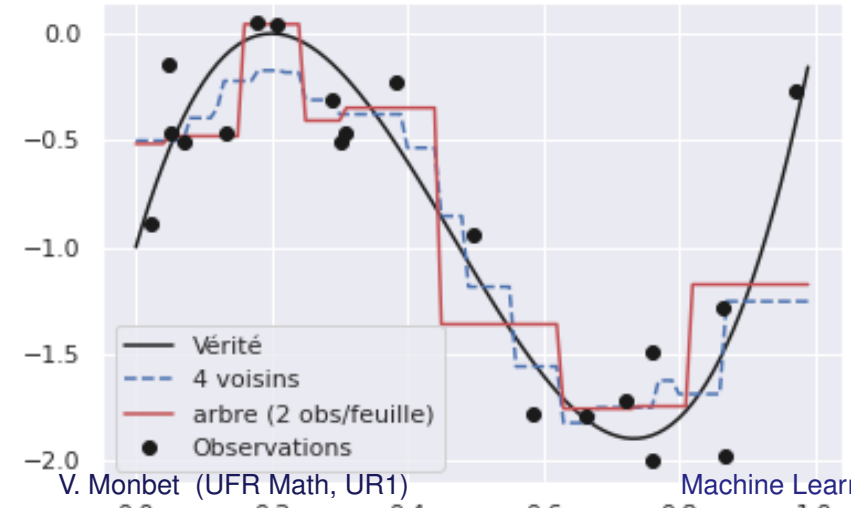
- The tree depth controls the balance between bias and variance.
  -Low depth (= a few divisions) leads to trees with a low variance but a large bias
  -Large depth (= a lot of divisions) leads to trees with a large variance and a low biais.

Example

$$y_i = (5x_i - 1)^2(x_i - 1) + u_i, \quad u_i \sim \mathcal{N}(0, 0.25)$$



One repeats $S = 50$ times the estimation of a tree with a minimum number of observations in each leaf varying from 1 to 10.
Boxplots ot the top panel represents the distribution of $\widehat{MSE}$.



The plot suggests to keep 1 or 2 observations y leaf. One gets the figure of the bottom panel.

Optimizing the Depth of a Decision Tree

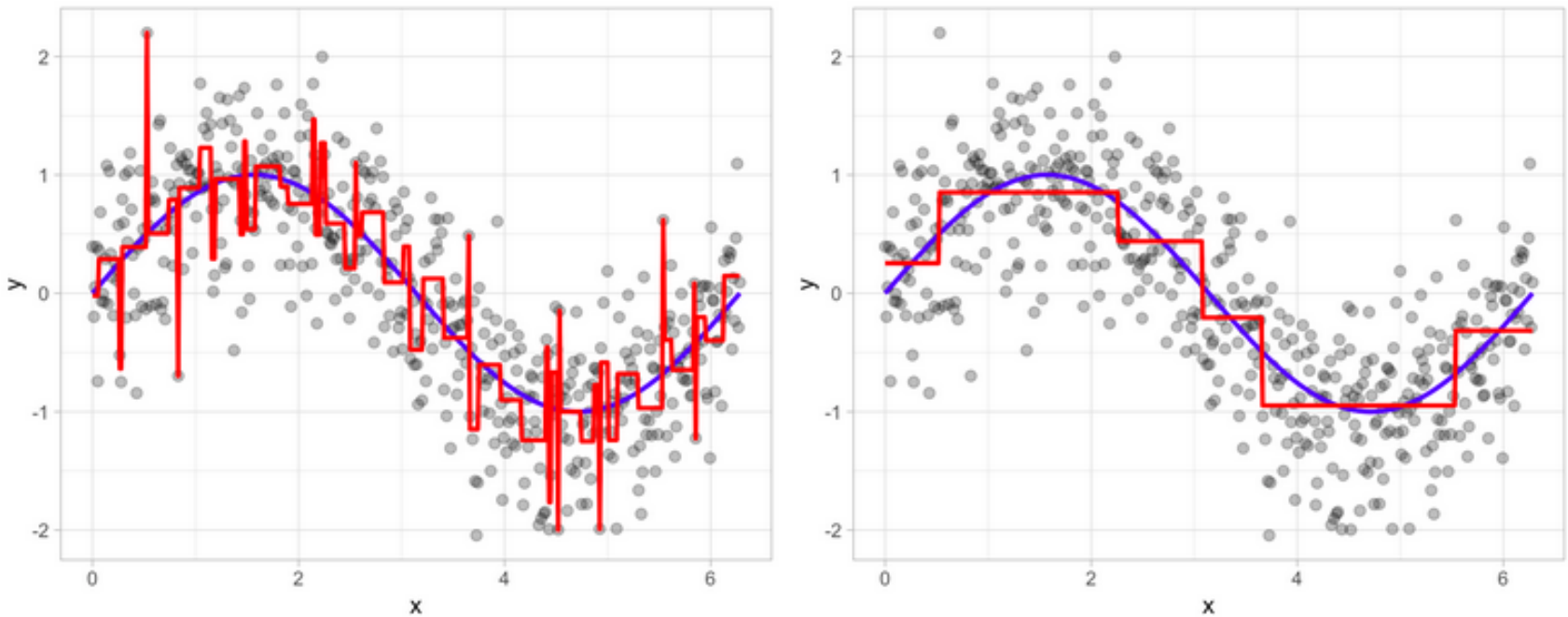
- The classical approach is to construct a tree that is initially too deep, for example, by stopping the algorithm when the minimum number of observations per node is reached. Then, the tree is pruned to optimize a cost-complexity criterion.
- The cost-complexity criterion is defined as

$$C_\alpha(\mathbb{T}) = \underbrace{\sum_{m=1}^{|\mathbb{T}|} N_m Q_m(\mathbb{T})}_{\text{cost}} + \underbrace{\alpha|\mathbb{T}|}_{\text{complexity}}$$

  where $N_m = \text{Card}(\{\mathbf{x}_i \in R_m\})$ and $Q_m(T) = \frac{1}{N_m}\sum_{\{i|\mathbf{x}_i \in R_m\}}(y_i - \hat{c}_m)^2$. $|\mathbb{T}|$ defines the depth of the tree.
- The idea is to find, for each α, the subtree $\mathbb{T}_\alpha$ of $\mathbb{T}$ that minimizes $C_\alpha(\mathbb{T})$.
- α is called the complexity parameter. It plays a similar role to the regularization constant introduced in penalized regression.
- In practice, α is selected using 5- or 10-fold cross-validation.

Pruning Example

Too deep tree on the left and pruned tree on the right.

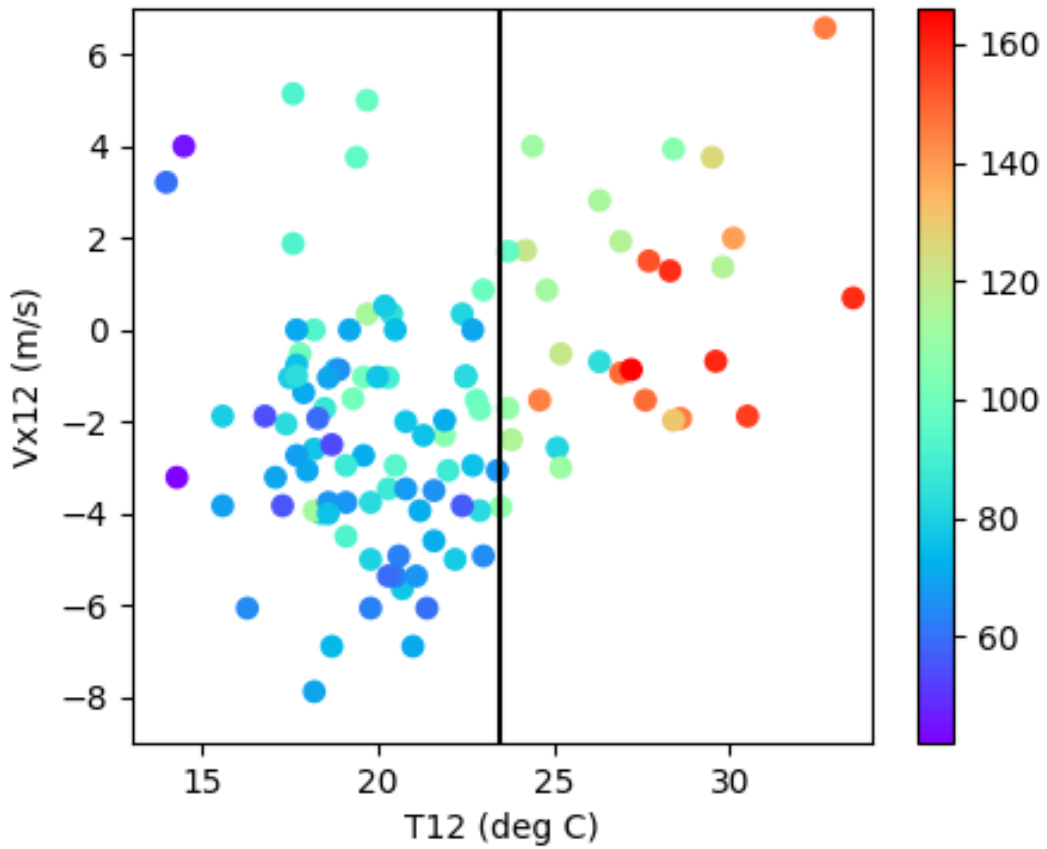


Curve to approximate (blue), noisy observations (black), and estimation (red).

Outline

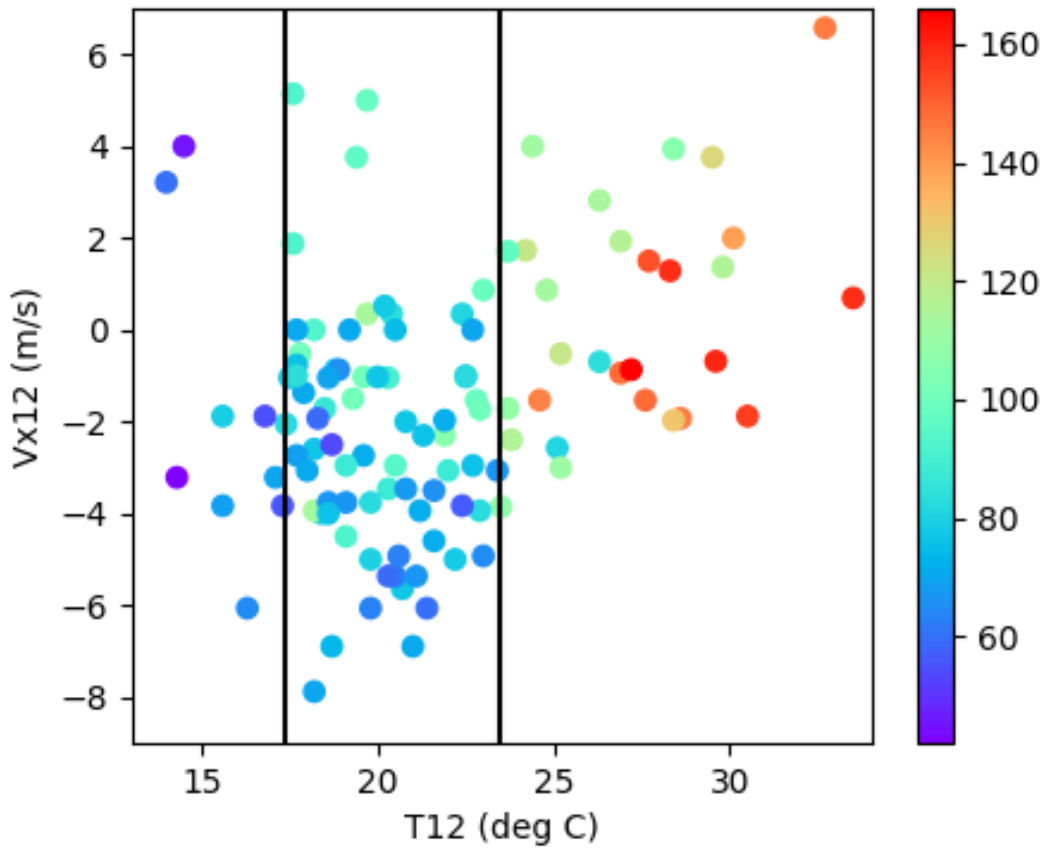
Example: Ozone (2 explanatory variables)

- Daily maximum O3 in relation to temperature and wind speed at 12h.



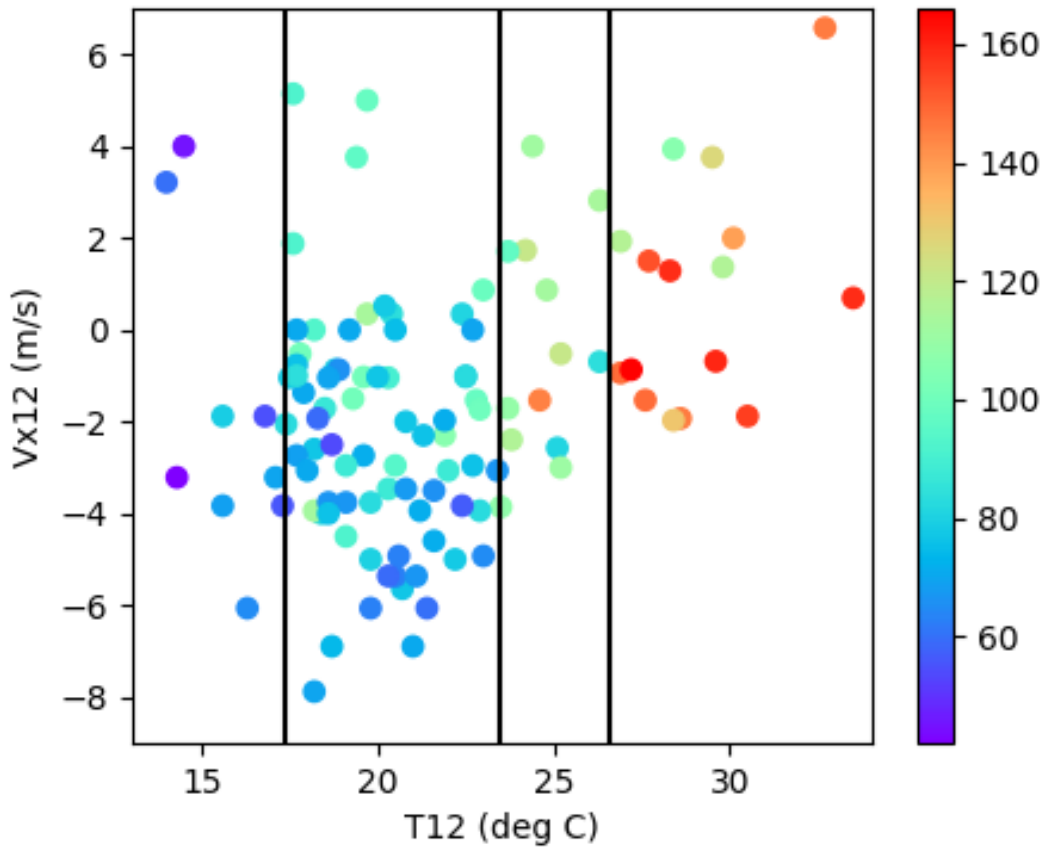
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



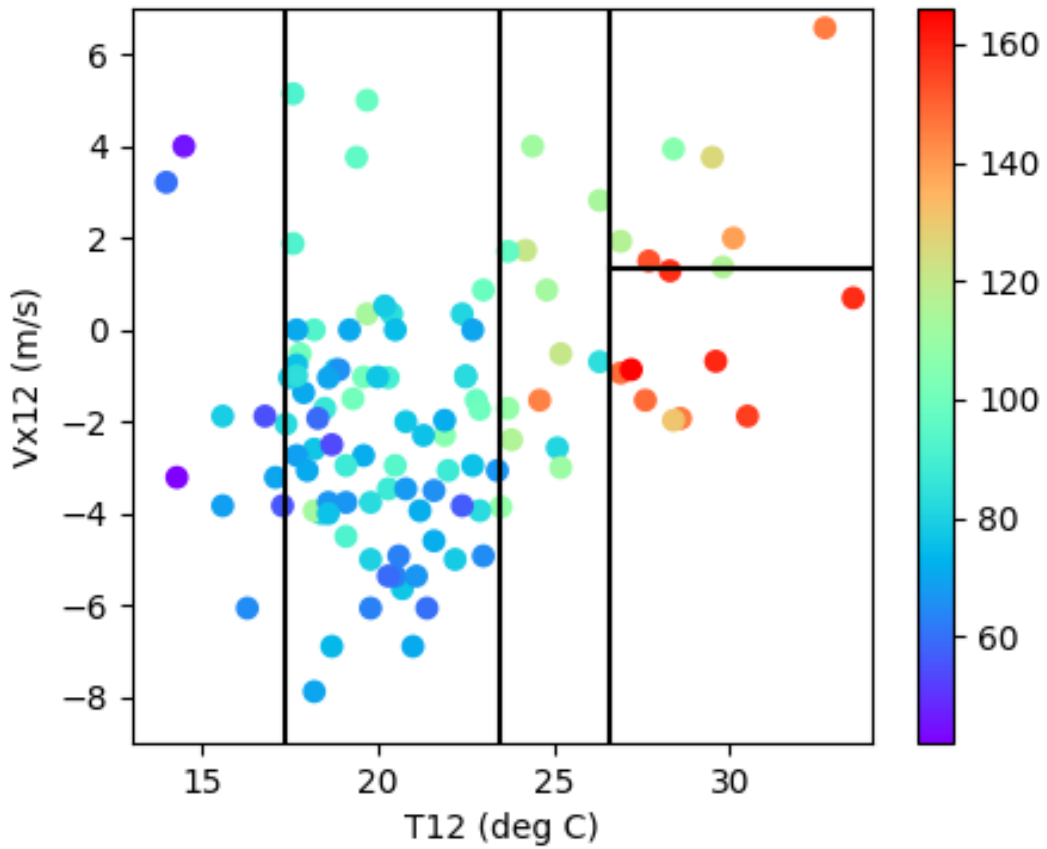
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



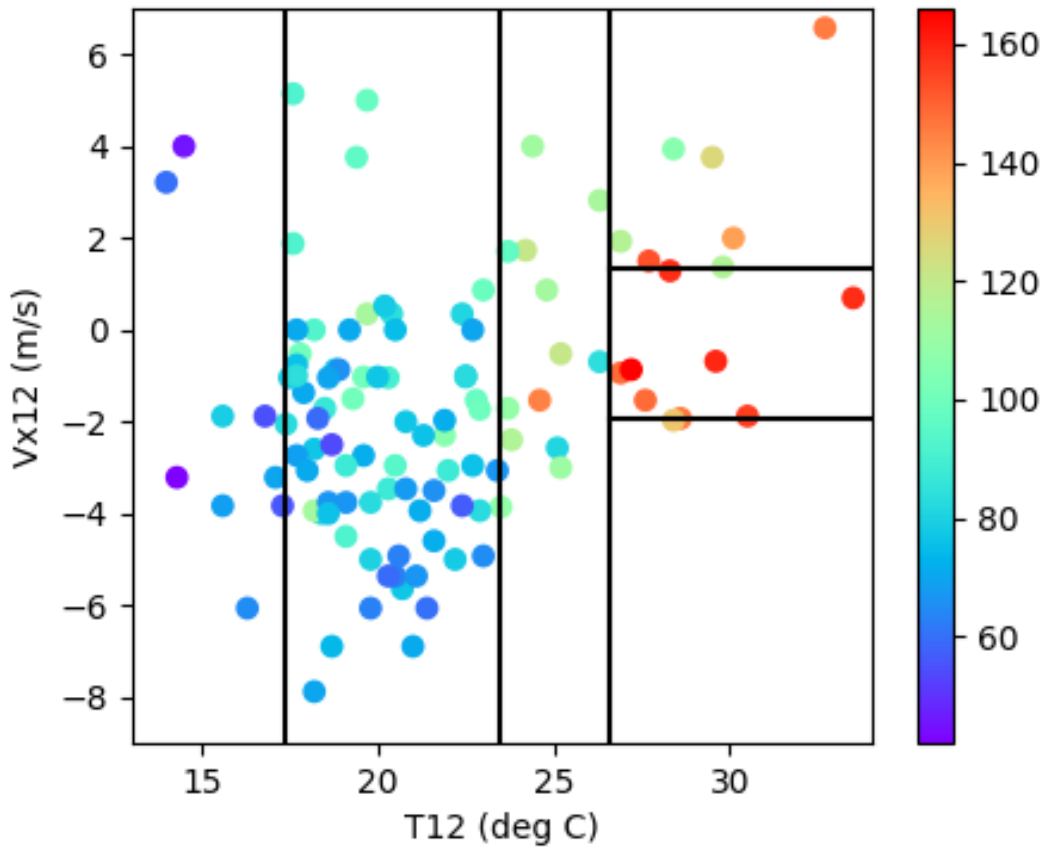
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



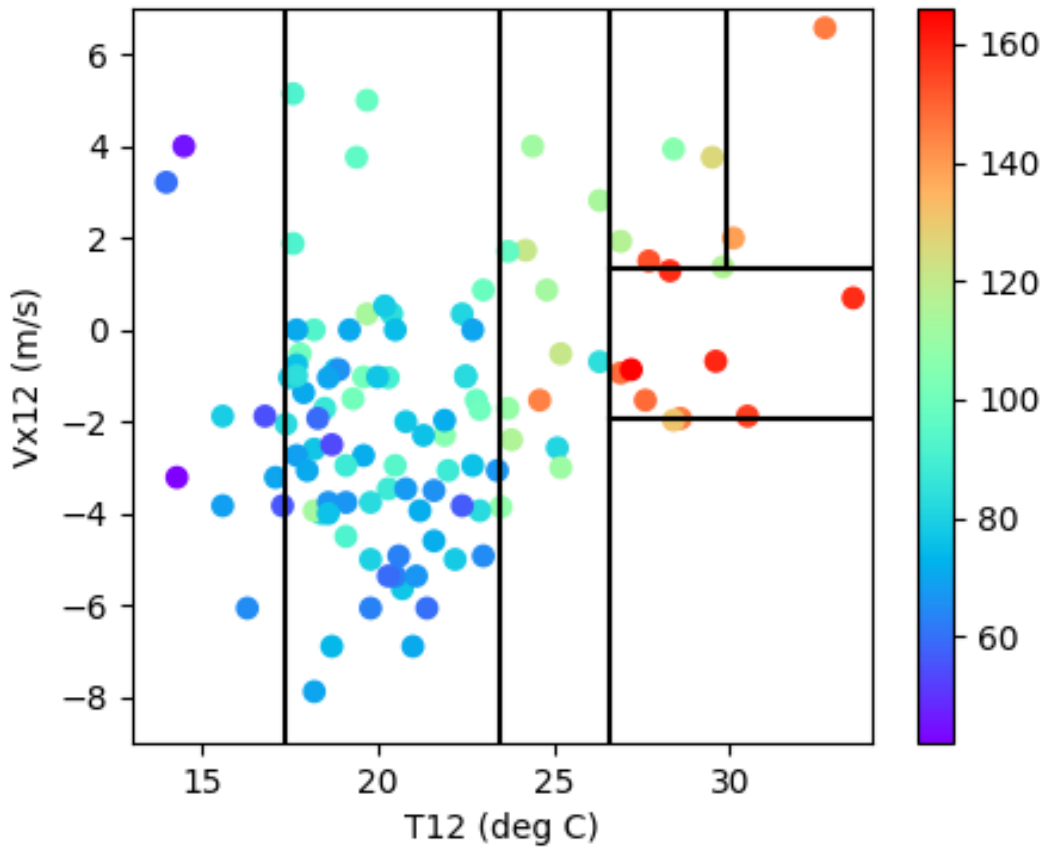
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



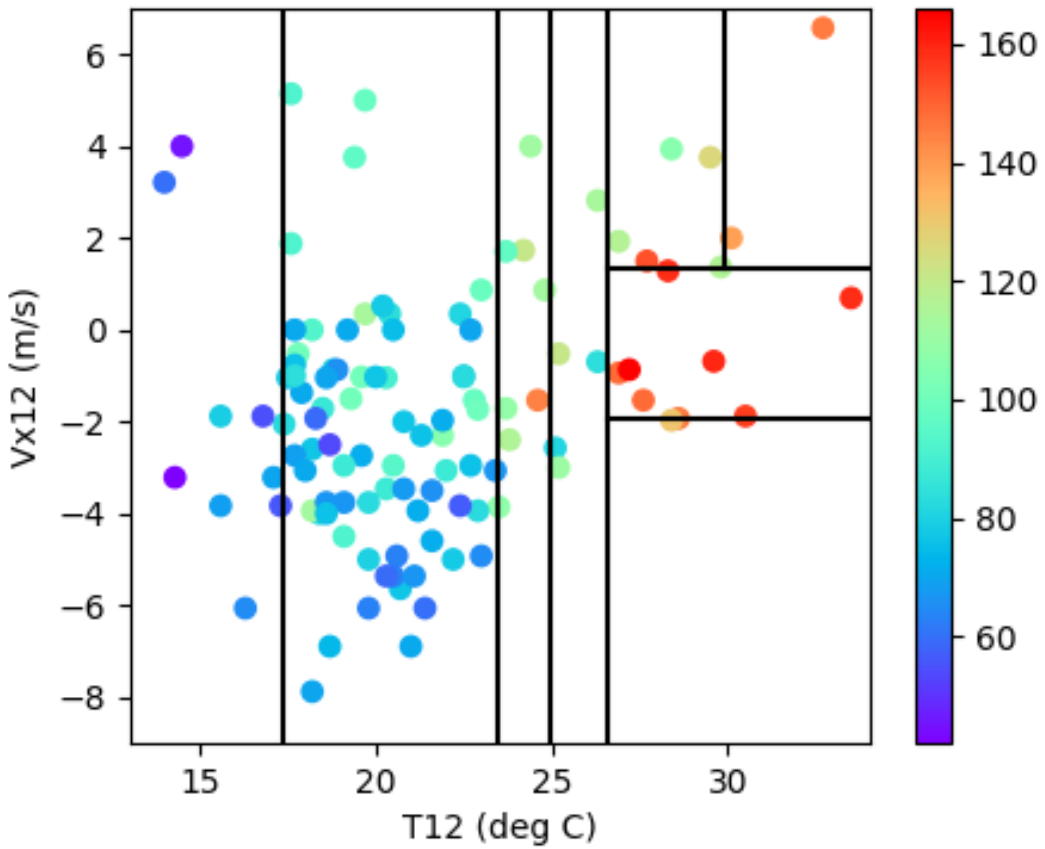
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



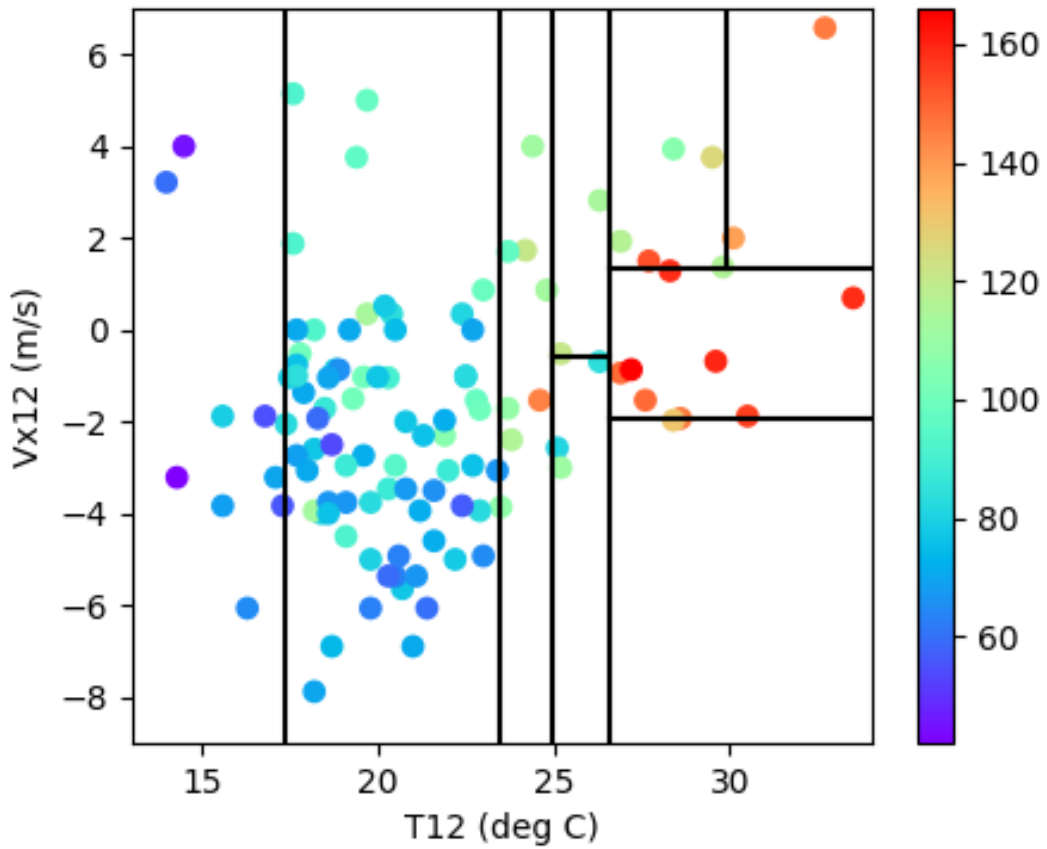
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



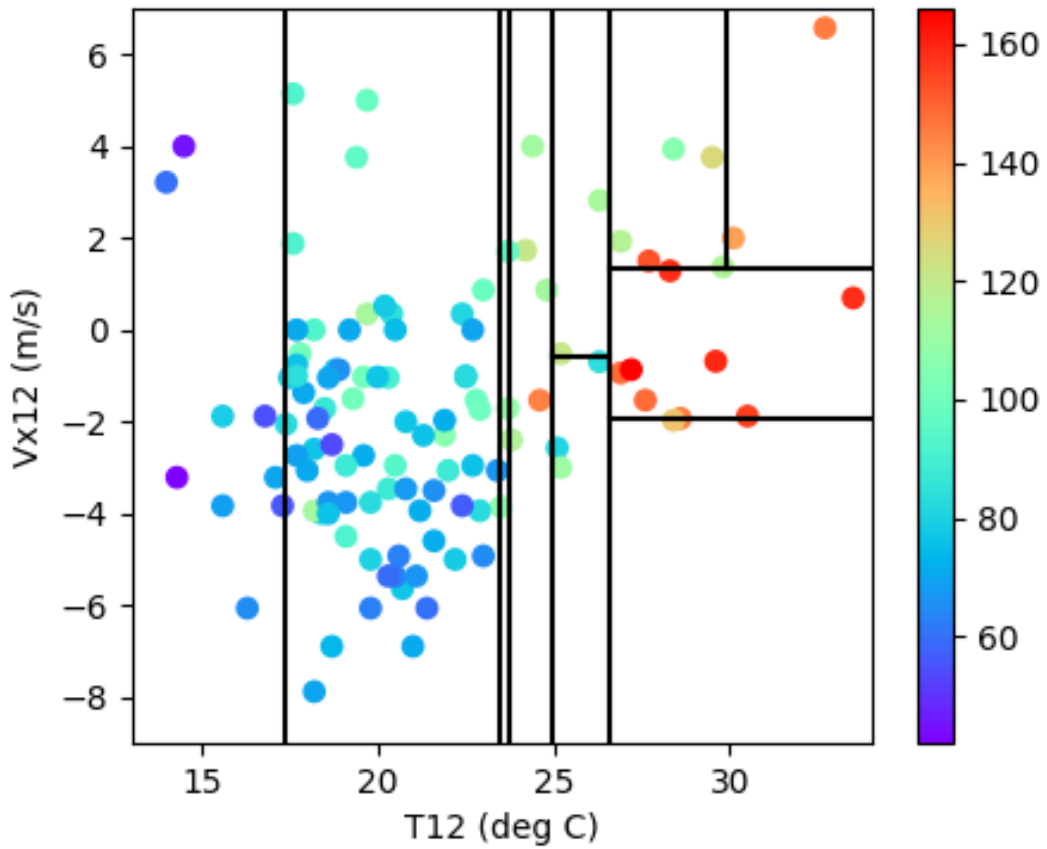
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



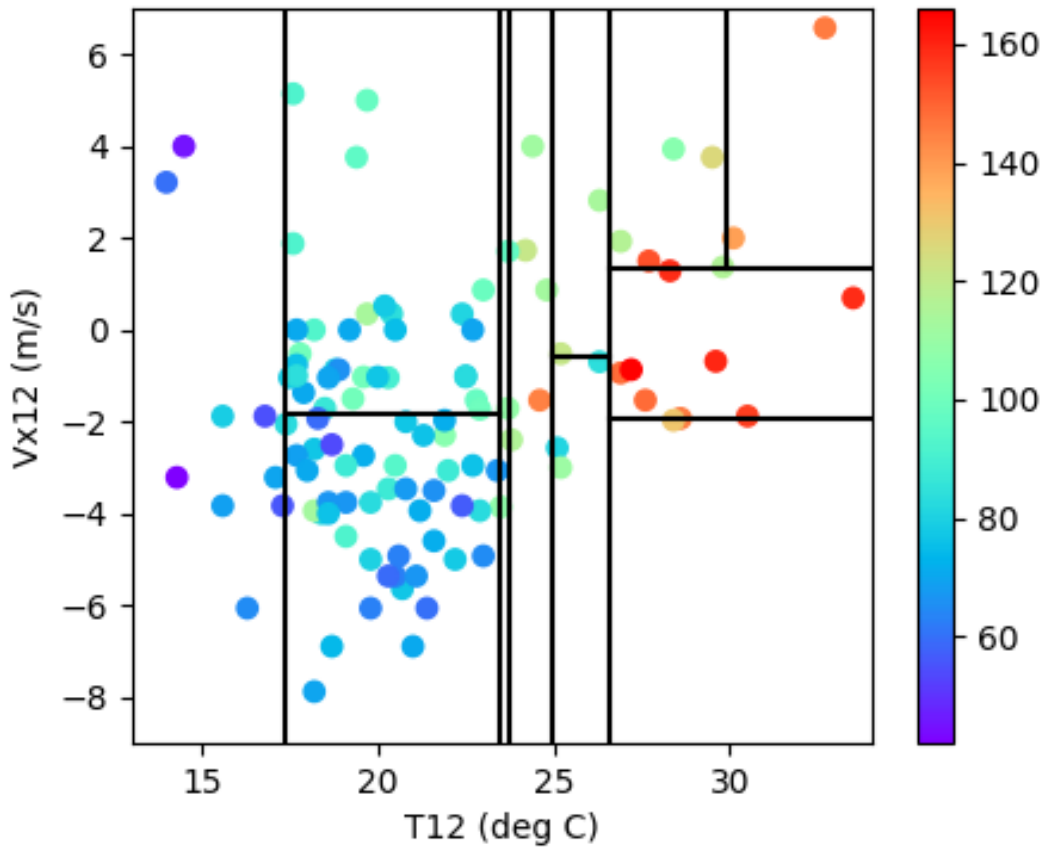
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



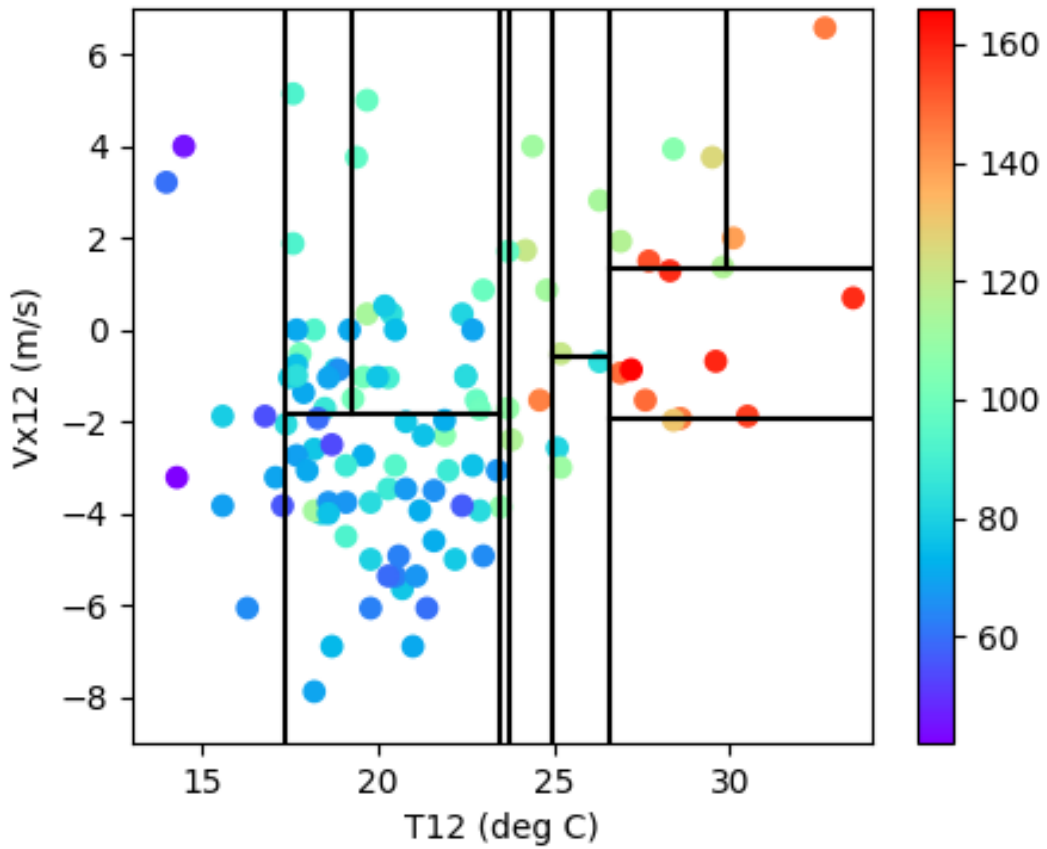
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



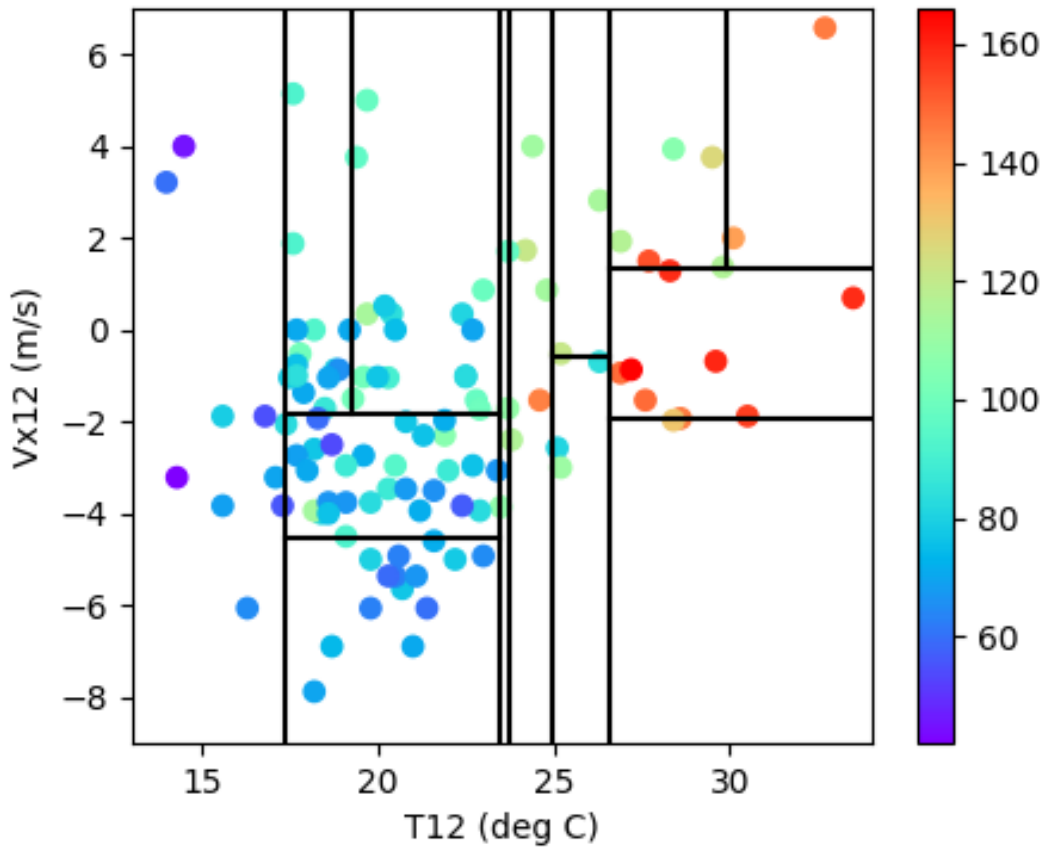
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



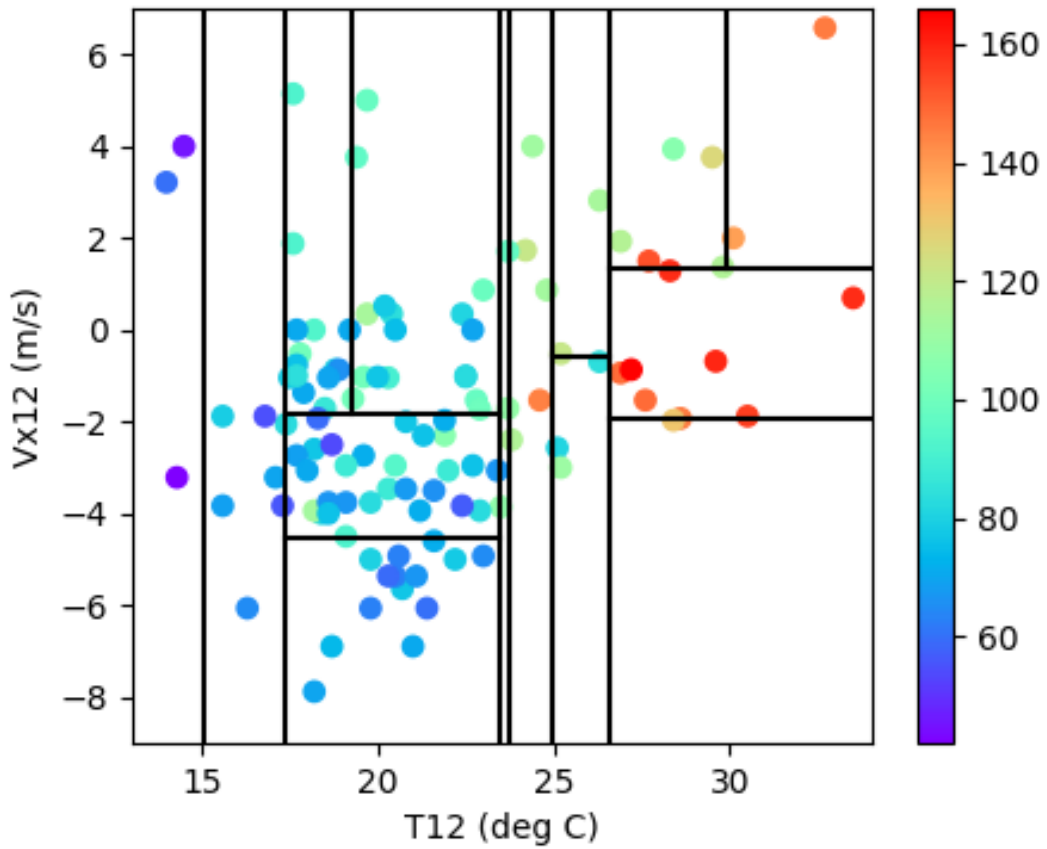
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



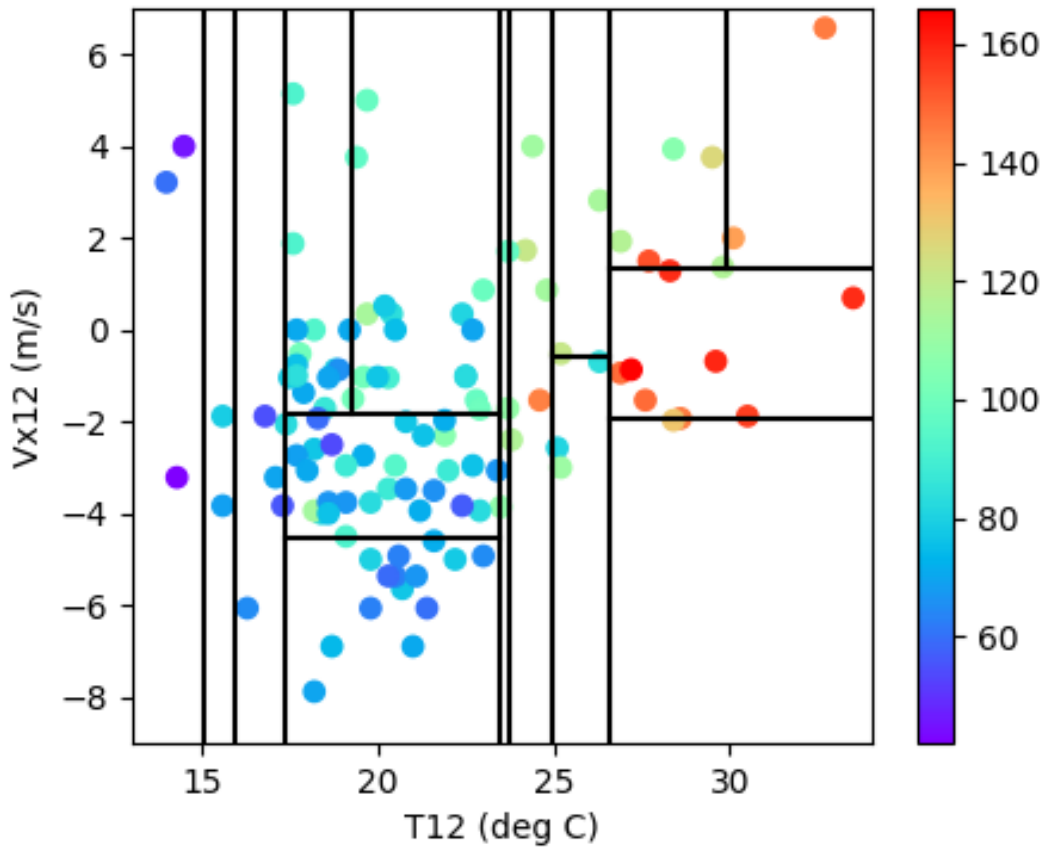
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



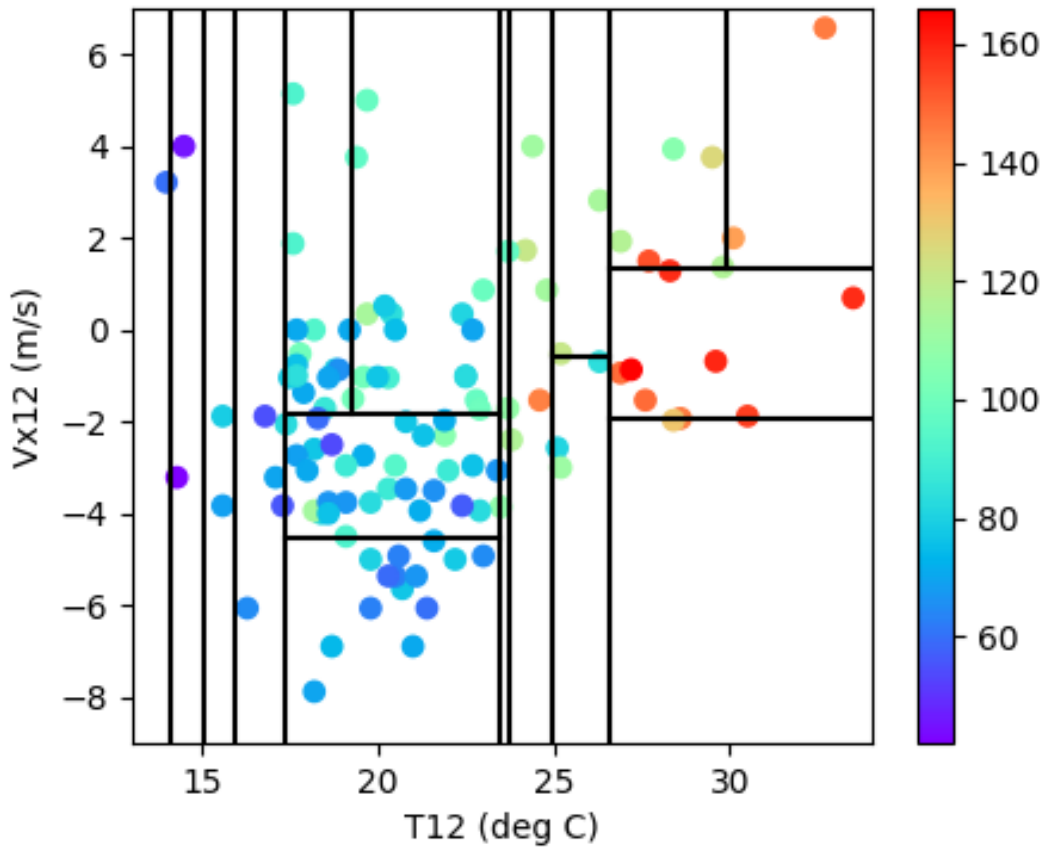
Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.

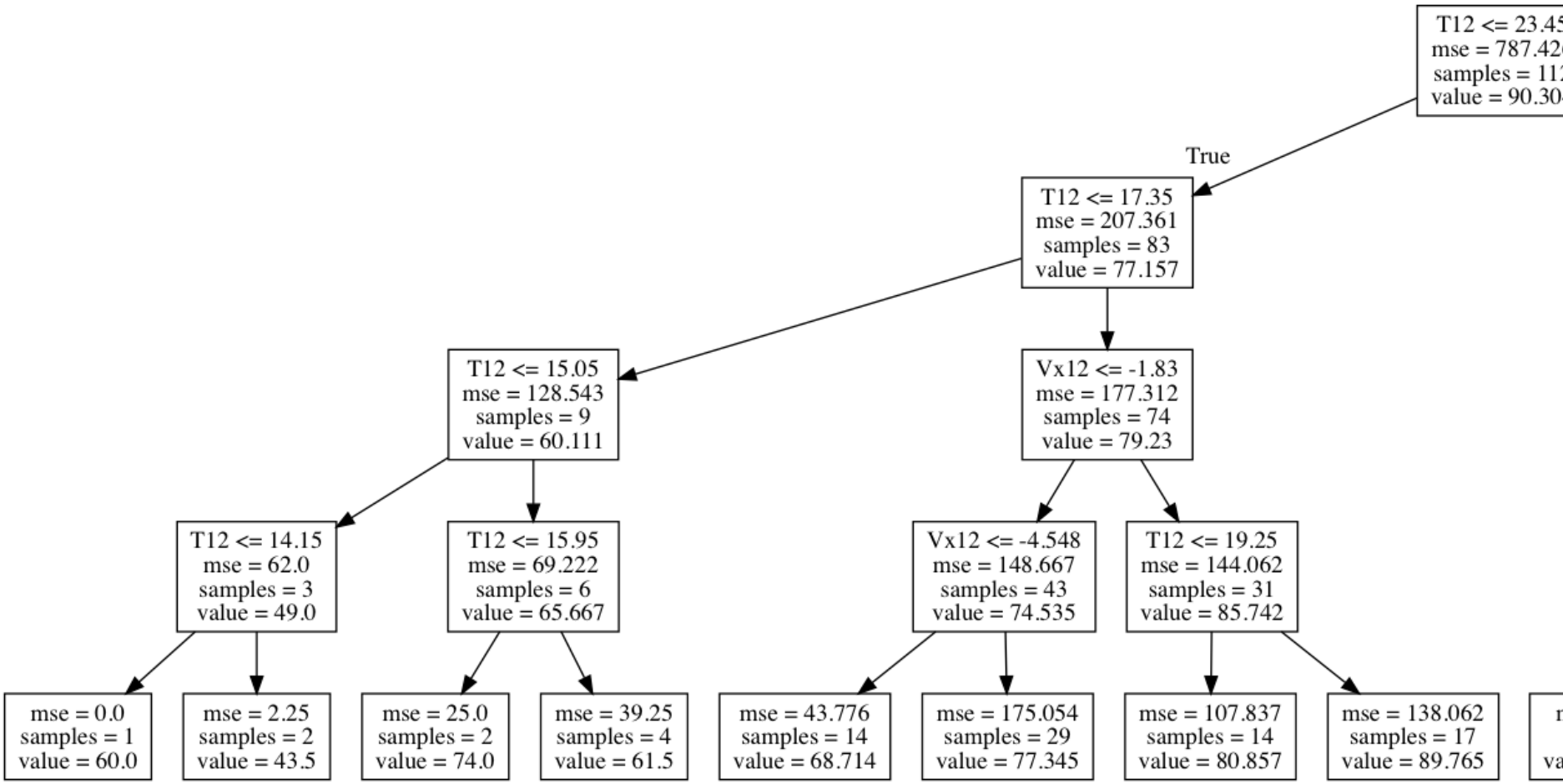


Example: Ozone

- Daily maximum O3 in relation to temperature and wind speed at 12h.



Representation



- In each node, we have the name of the dividing variable, its split threshold, the least squares error, the number of observations in the subregion, and the associated prediction.
- Terminal nodes are referred to as "leaves."

Outline

Classification Trees

- When $\mathcal{Y} = \{1, \cdots, K\}$, i.e., Y is a qualitative variable, tree algorithms are similar to the regression case for a quantitative variable.

- The main difference is in the choice of the criterion to optimize for selecting the splitting variable and threshold.[2] We select a criterion I for measuring homogeneity from the ones proposed below and search for the pair (j^*, s^*) that provides the best homogeneity gain.

- The prediction in a leaf R_m is determined by majority vote:

$$\hat{y} = \arg\max_{k \in \mathcal{Y}} \hat{p}_{mk}$$

where

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i | \mathbf{x}_i \in R_m} \mathbb{I}(y_i = k)$$

- Criteria: Gini index or entropy [CART algorithm], chi-squared test [CHAID algorithm].

[2] In the case of regression, variance is used.

Impurity Measures

- The **Gini index** is a measure of inequality. It was introduced to measure income inequality. It is equal to 0 in the case of perfect equality (perfect homogeneity) and 1 in the case of perfect inequality.

$$I_m = 1 - \sum_{k=1}^{K} p_{mk}^2$$

  where $p_{mk} = \frac{1}{N_m} \sum_{i|x_i \in R_m} \mathbb{I}_{\{y_i = k\}}$, m is the subregion number, and k is the class number.
  In the case of a binary variable, the Gini index calculates variance, so it is very similar to the criterion used in regression.

- **Entropy** can be interpreted as a measure of disorder. Zero entropy corresponds to a perfectly ordered environment (i.e., a perfectly homogeneous subregion).

$$E_m = \sum_{k=1}^{K} p_{mk} \log p_{mk}$$

- In general, classification error is not used because it is less sensitive and tends to select splits that lead to impure nodes.

Outline

Closing Remarks

Decision trees have several advantages:

- They are non-parametric models (very similar to k-Nearest Neighbors) that can capture non-linear relationships between variables.
- Decision trees require little or no preprocessing. For example, there is no need for marginal transformations. In particular, outliers in the predictor variables have relatively little impact on split decisions.
- Decision trees can handle categorical predictor variables with more than 2 levels. Refer to J. Friedman, Hastie, and Tibshirani (2001).
- Missing values also pose no problem. A common solution is to create a "missing" category for each variable.

However, the predictive performance of decision trees is often slightly below that of other algorithms. This is because trees consist of very simple rules that lead to non-smooth decisions. Furthermore, deep trees have high variance, and shallow trees have high bias.