ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING

Pattern Recognition & Machine Learning

Model Combination - Trees

Panagiotis C. Petrantonakis

Assistant Professor

Dept. of Electrical and Computer Engineering

ppetrant@ece.auth.gr

Fall Semester

Until now…

- We have talked about many different models for classification.
- It is often the case that the overall performance could be improved by model combination!
- Example: You have multiple NN models trained on different datasets for the same regression problem. For a new, test sample, the average of the predictions made by each model would probably lead to a better result than the outcome of a single model.
- The group of models that are combined like this are called *committees*!
  - Why *committees* lead to better results?

Test error of a classifier

- You can think about test error in terms of a decomposition into **two terms**.

Test error of a classifier

- You can think about test error in terms of a decomposition into **two terms**.

- Let f be a learned classifier, selected from a set $\mathcal{F}$, i.e., all possible classifiers using a fixed representation, e.g., trees with certain depth, NNs with certain number of neurons in the hidden layer etc.

Test error of a classifier

- You can think about test error in terms of a decomposition into **two terms**.
- Let f be a learned classifier, selected from a set $\mathcal{F}$, i.e., all possible classifiers using a fixed representation, e.g., trees with certain depth, NNs with certain number of neurons in the hidden layer etc.
- Then the error of the classifier on a test set can be decomposed as:

$$error(f) = \mathbb{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x};\mathcal{D}) - h(\boldsymbol{x})\right)^2\right] =$$

$$= \left(\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] - h(\boldsymbol{x})\right)^2 + \mathbb{E}_{\mathcal{D}}[(f(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})])^2]$$

where $\mathcal{D}$ is a particular dataset and $h(\boldsymbol{x})$ is the function to be approximated.

Bias/Variance Trade-off

$$\mathbb{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x};\mathcal{D}) - h(\boldsymbol{x})\right)^2\right] = \underbrace{\left(\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] - h(\boldsymbol{x})\right)^2}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})])^2]}_{\text{Variance}}$$

- Bias: measures the quality of the model family (e.g., NNs with certain number of neurons in hidden layer).
  - Suppose someone gave you infinite data to train your model; How well would you do with this particular model family?

- Variance: measures how far the actual, learned classifier (train over limited data) is from the optimal classifier of the family, i.e., the one trained with infinite data.
  - In essence, this tells you how much you have to pay due to the fact that you do not have infinite amount of data!
  - Sensitivity of the model to the individual dataset it was trained on.

Bias/Variance Trade-off

$$\mathbb{E}_{\mathcal{D}}\left[\left(f(\boldsymbol{x};\mathcal{D}) - h(\boldsymbol{x})\right)^2\right] = \underbrace{\left(\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] - h(\boldsymbol{x})\right)^2}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})])^2]}_{\text{Variance}}$$

- There is a fundamental trade-off between Bias and Variance in practical applications.

- As you make your model family more complex, e.g., complex NN architecture, you make $\mathcal{F}$ bigger. This, will cause a decrease in bias term as the approximation with infinite data will be much better.

- On the other hand this will also cause and increase in the variance term as more complex representation with limited data will most probably lead to overfitting!

Bias/Variance Trade-off - Example

- Assume we have L datasets of N points to approximate the function:

$$h(x) = \sin(2\pi x) + \varepsilon$$

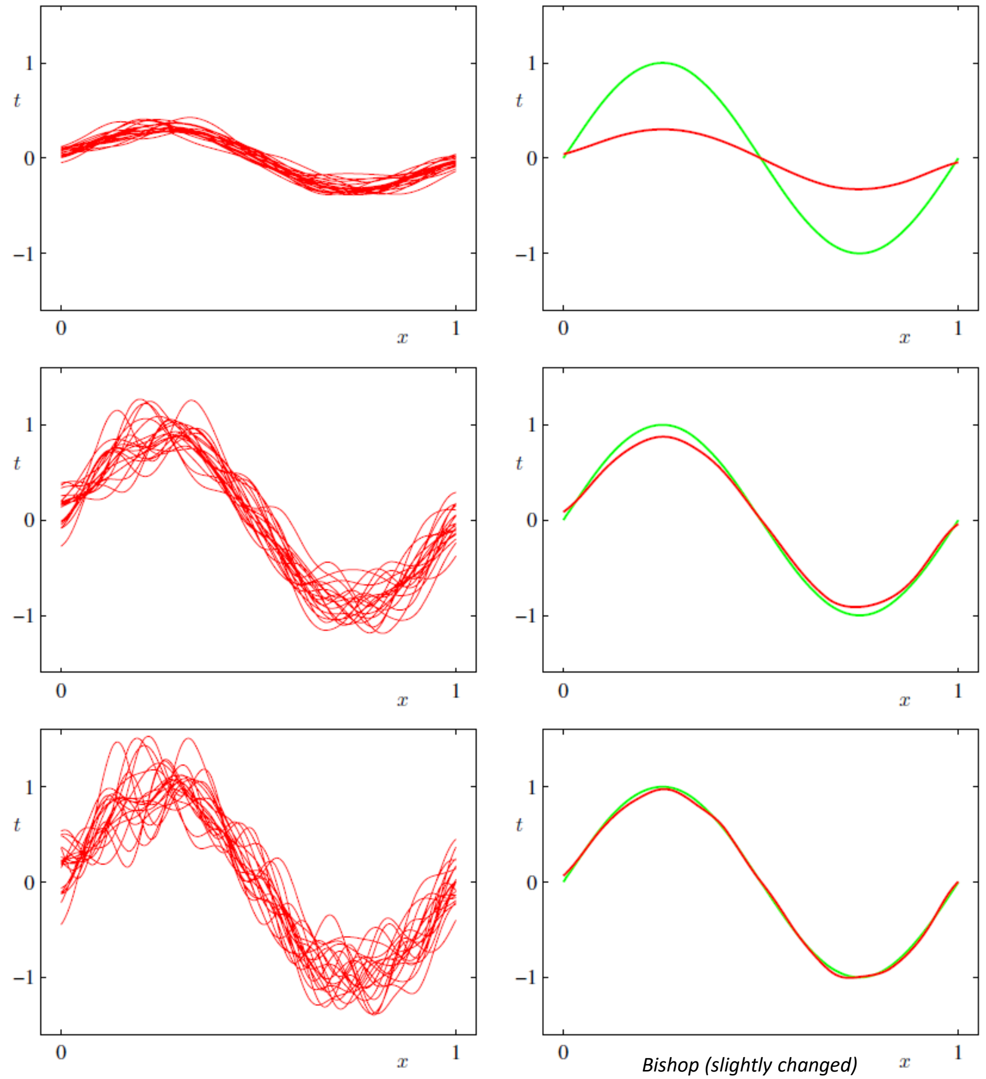
The final outcome of the committee will be:

$$\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x};\mathcal{D})] = \frac{1}{L}\sum_{l=1}^{L} f^{l}(x)$$

first row: high bias, low variance

second row: lower bias, high variance

third row: low bias, higher variance



Bishop (slightly changed)

Averaging predictions

- Averaging predictions of models trained on different datasets leads to reduction of the variance!
  - Low bias/high variance leads to low variance/low bias after averaging
- Thus, when we average a set of low-bias (high variance) models, i.e., complex models, we obtain accurate predictions. That's why committees work!
- Pitfall: we only have one single dataset!
- How could we artificially introduce variability between different models within a committee?

Averaging predictions

- Averaging predictions of models trained on different datasets leads to reduction of the variance!
  - Low bias/high variance leads to low variance/low bias after averaging
- Thus, when we average a set of low-bias (high variance) models, i.e., complex models, we obtain accurate predictions. That’s why committees work!
- Pitfall: we only have one single dataset!
- How could we artificially introduce variability between different models within a committee?
  - Bootstrap datasets!

Bootstrapping Datasets

- Suppose we have the dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$
- we can create B new datasets by simply drawing N points at random from $\boldsymbol{X}$ with replacement!
  - Some samples may appear multiple times in a new dataset after bootstrapping.
  - Some samples of the initial dataset may not appear at all.
- Thus, I get B new datasets: $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_B\}$.

Bootstrapping Datasets

- Suppose we have the dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$
- we can create B new datasets by simply drawing N points at random from $\boldsymbol{X}$ with replacement!
  - Some samples may appear multiple times in a new dataset after bootstrapping.
  - Some samples of the initial dataset may not appear at all.
- Thus, I get B new datasets: $\{\boldsymbol{X}_1, \dots, \boldsymbol{X}_B\}$.
- Now we can use these B training sets to train B different models (committee members).
- The final decision of the committee will be: $f_{com} = \frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x})$
  - Bootstrap aggregation/bagging

Regression with bootstrapping

- Suppose the ground truth function that we need to predict is $h(\boldsymbol{x})$.
- Each model gives a prediction: $f_b(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon_b(\boldsymbol{x})$
  - error of the model: $\epsilon_b(\boldsymbol{x})$
- The average sum-of-squares error of model b is :

$$\mathbb{E}_{\boldsymbol{x}}\left[\left(f_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = \mathbb{E}_{\boldsymbol{x}}[\epsilon_b(\boldsymbol{x})\ ^2]$$

- The average error of b individual models is:

$$E_{ave} = \frac{1}{B}\sum_{b=1}^{B} \mathbb{E}_{\boldsymbol{x}}[\epsilon_b(\boldsymbol{x})\ ^2]$$

Regression with bootstrapping

- The expected error of the committee is:

$$E_{com} = \mathbb{E}_{\boldsymbol{x}}\left[\left(f_{com}(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = \mathbb{E}_{\boldsymbol{x}}\left[\left(\frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right]$$

Regression with bootstrapping

- The expected error of the committee is:

$$E_{com} = \mathbb{E}_{\boldsymbol{x}}\left[\left(f_{com}(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = \mathbb{E}_{\boldsymbol{x}}\left[\left(\frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right]$$

- If we assume uncorrelated error between the different models then we can prove that:

$$E_{com} = \frac{1}{B} E_{ave}$$

Regression with bootstrapping

- The expected error of the committee is:

$$E_{com} = \mathbb{E}_{\boldsymbol{x}}\left[\left(f_{com}(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = \mathbb{E}_{\boldsymbol{x}}\left[\left(\frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right]$$

- If we assume uncorrelated error between the different models then we can prove that:

$$E_{com} = \frac{1}{B} E_{ave}$$

- Nevertheless, the way we defined the datasets after bootstrapping, it is not possible that the errors are not correlated.

Regression with bootstrapping

- The expected error of the committee is:

$$E_{com} = \mathbb{E}_{\boldsymbol{x}}\left[\left(f_{com}(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = \mathbb{E}_{\boldsymbol{x}}\left[\left(\frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right]$$

- If we assume uncorrelated error between the different models then we can prove that:

$$E_{com} = \frac{1}{B} E_{ave}$$

- Nevertheless, the way we defined the datasets after bootstrapping, it is not possible that the errors are not correlated.
- Even in the correlated case, it can be proved that: $E_{com} \leq E_{ave}$

Regression with bootstrapping

- The expected error of the committee is:

$$E_{com} = \mathbb{E}_{\boldsymbol{x}}\left[\left(f_{com}(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right] = \mathbb{E}_{\boldsymbol{x}}\left[\left(\frac{1}{B}\sum_{b=1}^{B} f_b(\boldsymbol{x}) - h(\boldsymbol{x})\right)^2\right]$$

- If we assume uncorrelated error between the different models then we can prove that:

$$E_{com} = \frac{1}{B} E_{ave}$$

- Nevertheless, the way we defined the datasets after bootstrapping, it is not possible that the errors are not correlated.
- Even in the correlated case, it can be proved that: $E_{com} \le E_{ave}$
- Basic strategy: choose B models with low bias (complex models that can overfit). Then bootstrap aggregated error will be lower than the average error of the individual models.

Feature Bagging

- Feature Bagging: create new datasets by sampling **subsets of features** of length $M < D$ for each learner.
    - Also called ‘random subspace method’

Feature Bagging

- Feature Bagging: create new datasets by sampling **subsets of features** of length $M < D$ for each learner.
  - Also called 'random subspace method'

- Example:
  - Assume dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ with samples $\boldsymbol{x}_n = \{x_1, x_2, x_3, x_4, x_5, x_6\}$. I create new datasets $\boldsymbol{Y}^1 = \left\{\boldsymbol{y}_1^1, \dots, \boldsymbol{y}_N^1\right\}$ with $\boldsymbol{y}_n^1 = \{x_1, x_2, x_6\}$ and $\boldsymbol{Y}^2 = \left\{\boldsymbol{y}_1^2, \dots, \boldsymbol{y}_N^2\right\}$ with $\boldsymbol{y}_n^2 = \{x_3, x_4, x_6\}$.
  - Train different models with different datasets after feature bagging

Feature Bagging

- Feature Bagging: create new datasets by sampling **subsets of features** of length $M < D$ for each learner.
  - Also called ‘random subspace method’

- Example:
  - Assume dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ with samples $\boldsymbol{x}_n = \{x_1, x_2, x_3, x_4, x_5, x_6\}$. I create new datasets $\boldsymbol{Y}^1 = \{\boldsymbol{y}_1^1, \dots, \boldsymbol{y}_N^1\}$ with $\boldsymbol{y}_n^1 = \{x_1, x_2, x_6\}$ and $\boldsymbol{Y}^2 = \{\boldsymbol{y}_1^2, \dots, \boldsymbol{y}_N^2\}$ with $\boldsymbol{y}_n^2 = \{x_3, x_4, x_6\}$.
  - Train different models with different datasets after feature bagging

- It works well if features are uncorrelated.

- Advantage: pushes learners to **not** over focus on features that predict well on certain training set but do not generalize well to new data!

- Works well when I have too many features and too few samples.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.
- *Boosting* algorithms use initial dataset but train, **sequentially**, different models.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.
- *Boosting* algorithms use initial dataset but train, **sequentially**, different models.
- In boosting, we use simple classifiers (weak learners). Each one of these classifiers is trained using a weighted form of the dataset.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.
- *Boosting* algorithms use initial dataset but train, **sequentially**, different models.
- In boosting, we use simple classifiers (weak learners). Each one of these classifiers is trained using a weighted form of the dataset.
- The weighting coefficient associated with each datapoint of the dataset, depends on the performance of the previous model.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.
- *Boosting* algorithms use initial dataset but train, **sequentially**, different models.
- In boosting, we use simple classifiers (weak learners). Each one of these classifiers is trained using a weighted form of the dataset.
- The weighting coefficient associated with each datapoint of the dataset, depends on the performance of the previous model.
- Samples that are misclassified are given greater weight for the subsequent classifier in the sequence.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.
- *Boosting* algorithms use initial dataset but train, **sequentially**, different models.
- In boosting, we use simple classifiers (weak learners). Each one of these classifiers is trained using a weighted form of the dataset.
- The weighting coefficient associated with each datapoint of the dataset, depends on the performance of the previous model.
- Samples that are misclassified are given greater weight for the subsequent classifier in the sequence.
- Thus,
  - Bootstrap-bagging: Complex model with low bias. Decrease variance by averaging.
  - Boosting: Simple models with high bias (underfit). Decrease bias (and variance) by focusing on error of previous models.

Boosting

- Bootstrapping and Feature Bagging methods create new datasets and train **in parallel** different models. The final decision was made by averaging.
- *Boosting* algorithms use initial dataset but train, **sequentially**, different models.
- In boosting, we use simple classifiers (weak learners). Each one of these classifiers is trained using a weighted form of the dataset.
- The weighting coefficient associated with each datapoint of the dataset, depends on the performance of the previous model.
- Samples that are misclassified are given greater weight for the subsequent classifier in the sequence.
- Thus,
  - Bootstrap-bagging: Complex model with low bias. Decrease variance by averaging.
  - Boosting: Simple models with high bias (underfit). Decrease bias (and variance) by focusing on error of previous models.
- Boosting, final decision: $Y_{com}(\boldsymbol{x}) = sign\left(\sum_{m=1}^{M} a_m y_m(\boldsymbol{x})\right)$, $y_m(\boldsymbol{x}) \in \{-1,1\}$

Boosting: Binary Classification

- Assume $\boldsymbol{X} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^{N}$ where $t_n \in \{-1,1\}$

Boosting: Binary Classification

- Assume $\boldsymbol{X} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^{N}$ where $t_n \in \{-1,1\}$
- Each data sample has an associated weight w_n. Initialization: $w_n = \frac{1}{N}$
- Assume that our base classifier produces labels $y_m(\boldsymbol{x}) \in \{-1,1\}$

Boosting: Binary Classification

- Assume $\boldsymbol{X} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^{N}$ where $t_n \in \{-1,1\}$
- Each data sample has an associated weight w_n. Initialization: $w_n = \frac{1}{N}$
- Assume that our base classifier produces labels $y_m(\boldsymbol{x}) \in \{-1,1\}$
- The most popular boosting algorithm is AdaBoost (Adaptive Boosting)
    - At each step a new classifier is trained on weighted dataset
    - Weights of misclassified data samples increase.
    - Committee decision is formed by weighted base classifiers.

AdaBoost Algorithm

1. Initialize weights: $w_n^1 = \frac{1}{N}, n = 1, \dots, N$

AdaBoost Algorithm

1. Initialize weights: $w_n^1 = \frac{1}{N}, n = 1, \dots, N$

2. for $m = 1, \dots, M$:

   a) Train $y_m(\boldsymbol{x})$ to minimize $J_m = \sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]$

   b) Compute weighted error rates $\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]}{\sum_{n=1}^{N} w_n^m}$

      and model weight $a_m = ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$

   a) Update weights $w_n^{m+1} = w_n^m \exp\{a_m I[y_m(\boldsymbol{x}_n) \neq t_n]\}$

AdaBoost Algorithm

1. Initialize weights: $w_n^1 = \frac{1}{N}, n = 1, \dots, N$

2. for $m = 1, \dots, M$:

   a) Train $y_m(\boldsymbol{x})$ to minimize $J_m = \sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]$

   b) Compute weighted error rates $\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]}{\sum_{n=1}^{N} w_n^m}$

      and model weight $a_m = ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$

   a) Update weights $w_n^{m+1} = w_n^m \exp\{a_m I[y_m(\boldsymbol{x}_n) \neq t_n]\}$

3. Make prediction by $Y_M(\boldsymbol{x}) = sign\left(\sum_{m=1}^{M} a_m y_m(\boldsymbol{x})\right)$

AdaBoost Algorithm

1. Initialize weights: $w_n^1 = \frac{1}{N}, n = 1, \dots, N$

2. for $m = 1, \dots, M$:

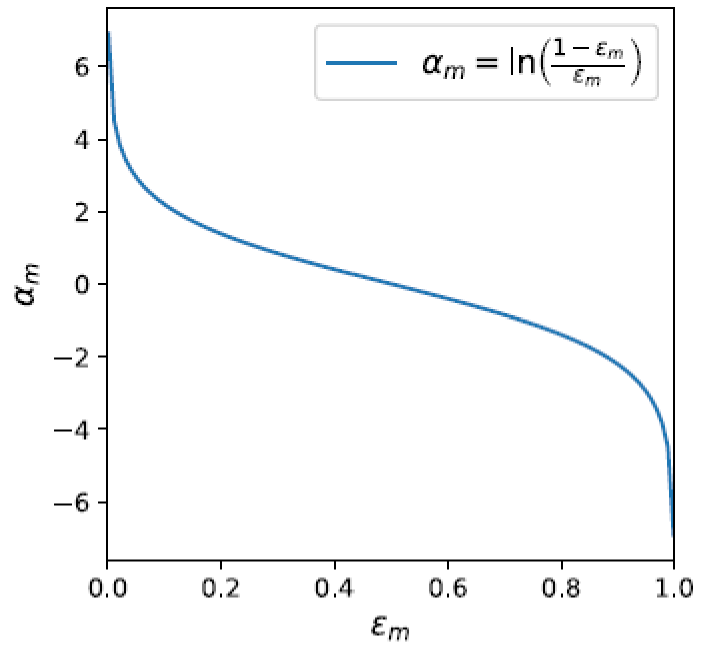
   a) Train $y_m(\boldsymbol{x})$ to minimize $J_m = \sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]$

   b) Compute weighted error rates $\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]}{\sum_{n=1}^{N} w_n^m}$

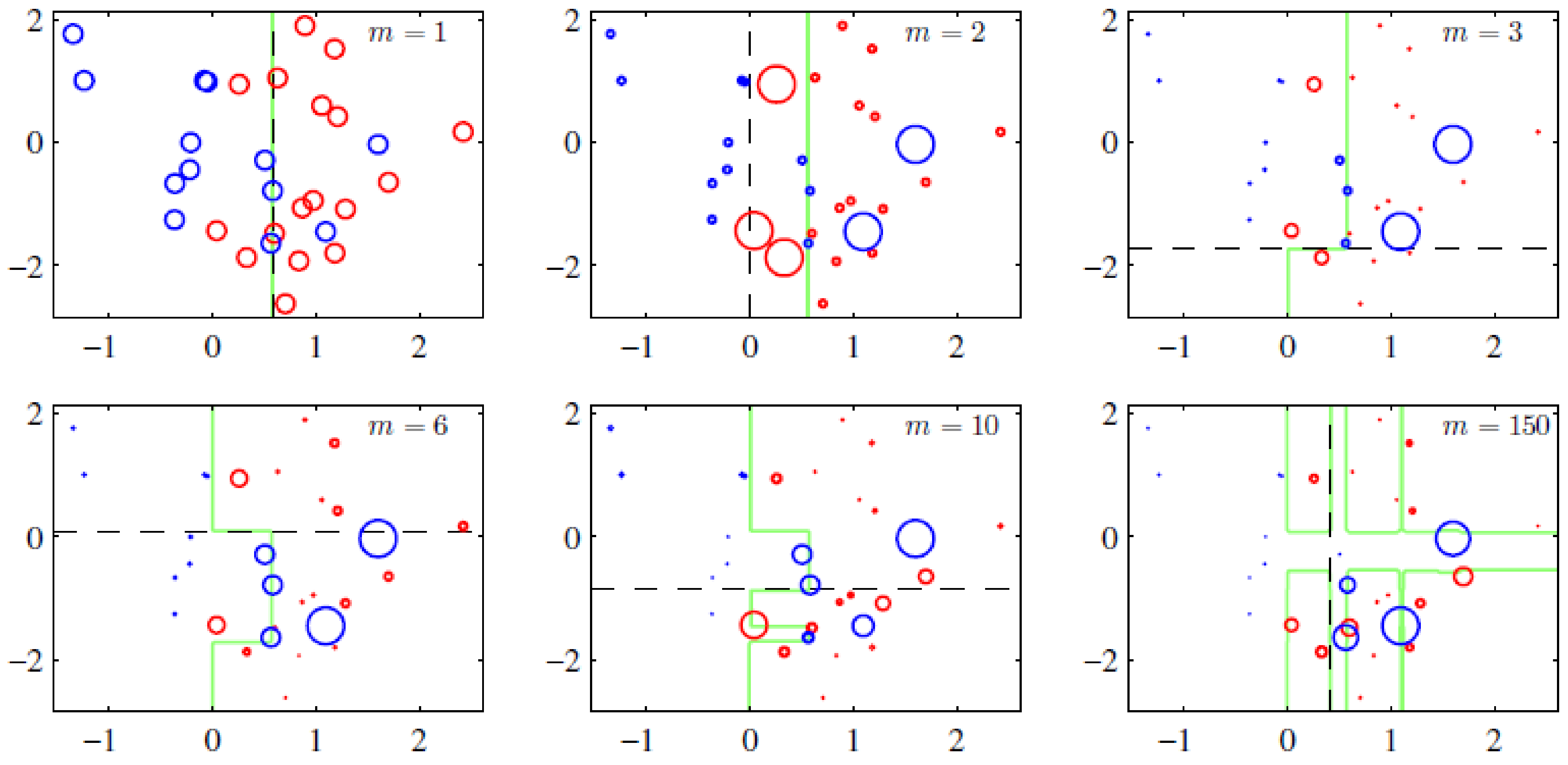
      and model weight $a_m = ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$

   a) Update weights $w_n^{m+1} = w_n^m \exp\{a_m I[y_m(\boldsymbol{x}_n) \neq t_n]\}$

3. Make prediction by $Y_M(\boldsymbol{x}) = sign\left(\sum_{m=1}^{M} a_m y_m(\boldsymbol{x})\right)$



AdaBoost Algorithm - Example



Bishop

Dashed line: current base classifier boundary, green line: committee's decision boundary, cycles: training samples, radius of cycles: weight of the sample for current base classifier.

Interpretation of AdaBoost

- AdaBoost is actually minimizing an exponential error function:

$$E_m = \sum_{n=1}^{N} \exp[-t_n f_m(\boldsymbol{x}_n)]$$

where $t_n f_m(\boldsymbol{x}_n) > 0$ if the sample $\boldsymbol{x}_n$is correctly classified and $t_n f_m(\boldsymbol{x}_n) \leq 0$ otherwise.

Interpretation of AdaBoost

- AdaBoost is actually minimizing an exponential error function:

$$E_m = \sum_{n=1}^{N} \exp[-t_n f_m(\boldsymbol{x}_n)]$$

where $t_n f_m(\boldsymbol{x}_n) > 0$ if the sample $\boldsymbol{x}_n$is correctly classified and $t_n f_m(\boldsymbol{x}_n) \leq 0$ otherwise. Moreover:

$$f_m(\boldsymbol{x}_n) = \frac{1}{2}\sum\nolimits_{i=1}^{m} a_i y_i(\boldsymbol{x}_n)$$

is the linear combination of base classifiers $y_i(\boldsymbol{x}_n)$, where m is the number of committee members.

Interpretation of AdaBoost

- AdaBoost is actually minimizing an exponential error function:

$$E_m = \sum_{n=1}^{N} \exp[-t_n f_m(\boldsymbol{x}_n)]$$

where $t_n f_m(\boldsymbol{x}_n) > 0$ if the sample $\boldsymbol{x}_n$ is correctly classified and $t_n f_m(\boldsymbol{x}_n) \leq 0$ otherwise. Moreover:

$$f_m(\boldsymbol{x}_n) = \frac{1}{2} \sum_{i=1}^{m} a_i y_i(\boldsymbol{x}_n)$$

is the linear combination of base classifiers $y_i(\boldsymbol{x}_n)$, where m is the number of committee members.

- Goal: minimize E_m w.r.t. a_i and the parameters of $y_i(\boldsymbol{x}_n)$
  - *Sequential minimization:* fix $y_1(\boldsymbol{x}), \dots, y_{m-1}(\boldsymbol{x})$ and $a_1, \dots, a_{m-1}$ and minimize E_m w.r.t. a_m and the parameters of $y_m(\boldsymbol{x}_n)$

Derivation of AdaBoost - 1

$f_m(\boldsymbol{x}_n) = \frac{1}{2}\sum_{i=1}^{m} a_i y_i(\boldsymbol{x}_n)$

- Error function:

$$E_m = \sum_{n=1}^{N} \exp\left(-t_n f_{m-1}(\boldsymbol{x}_n) - \frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$
$$= \sum_{n=1}^{N} w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$

where $w_n^m = exp\left(-t_n f_{m-1}(\boldsymbol{x}_n)\right)$ is known (fixed) based on the trainings so far!

- Assume:
  - C_m to be the set of samples that are correctly classified by $y_m(\boldsymbol{x})$, i.e., $t_n y_m(\boldsymbol{x}_n) = 1$
  - M_m to be the set of samples that are misclassified by $y_m(\boldsymbol{x})$, i.e., $t_n y_m(\boldsymbol{x}_n) = -1$

- Then we can rewrite error function as:

Derivation of AdaBoost - 2

$$E_m = \sum_{n=1}^{N} w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right) = e^{-\frac{a_m}{2}} \sum_{n \in C_m} w_n^m + e^{\frac{a_m}{2}} \sum_{n \in M_m} w_n^m$$

$$= \left(e^{\frac{a_m}{2}} - e^{-\frac{a_m}{2}}\right) \sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n] + e^{-\frac{a_m}{2}} \sum_{n=1}^{N} w_n^m$$

- Thus, minimization of E_m w.r.t. $y_m(\boldsymbol{x}_n)$ minimizes $J_m = \sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]$

- minimization of E_m w.r.t. a_m: $\frac{\partial E_m}{\partial a_m} = 0$:

$$a_m = ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right) \text{ where } \varepsilon_m = \frac{\sum_{n=1}^{N} w_n^m I[y_m(\boldsymbol{x}_n) \neq t_n]}{\sum_{n=1}^{N} w_n^m}$$

Derivation of AdaBoost - 3

- As soon as we have found $y_m(\boldsymbol{x}_n)$ and a_m, we subsequently minimize E_{m+1} w.r.t. $y_{m+1}(\boldsymbol{x}_n)$ and a_{m+1}:

$$E_{m+1} = \sum_{n=1}^{N} \exp[-t_n f_{m+1}(\boldsymbol{x}_n)] =$$

$$= \sum_{n=1}^{N} w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right) \exp\left(-\frac{1}{2} t_n a_{m+1} y_{m+1}(\boldsymbol{x}_n)\right) =$$

$$= \sum_{n=1}^{N} w_n^{m+1} \exp\left(-\frac{1}{2} t_n a_{m+1} y_{m+1}(\boldsymbol{x}_n)\right)$$

where

$$w_n^{m+1} = w_n^m exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$

Derivation of AdaBoost - 4

- Weight updates:

$$w_n^{m+1} = w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$

Derivation of AdaBoost - 4

- Weight updates:

$$w_n^{m+1} = w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$

- If we use: $t_n y_m(\boldsymbol{x}_n) = 1 - 2I[y_m(\boldsymbol{x}_n) \neq t_n]$

Derivation of AdaBoost - 4

- Weight updates:

$$w_n^{m+1} = w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$

- If we use: $t_n y_m(\boldsymbol{x}_n) = 1 - 2I[y_m(\boldsymbol{x}_n) \neq t_n]$
- Then $w_n^{m+1} = w_n^m \exp\left(-\frac{a_m}{2}\right) \exp(a_m I[y_m(\boldsymbol{x}_n) \neq t_n]) =$

$$w_n^m \exp(a_m I[y_m(\boldsymbol{x}_n) \neq t_n])$$

- The term $\exp\left(-\frac{a_m}{2}\right)$ is independent of n and, thus, can be discarded.

Derivation of AdaBoost - 4

- Weight updates:

$$w_n^{m+1} = w_n^m \exp\left(-\frac{1}{2} t_n a_m y_m(\boldsymbol{x}_n)\right)$$

- If we use: $t_n y_m(\boldsymbol{x}_n) = 1 - 2I[y_m(\boldsymbol{x}_n) \neq t_n]$
- Then $w_n^{m+1} = w_n^m \exp\left(-\frac{a_m}{2}\right) \exp(a_m I[y_m(\boldsymbol{x}_n) \neq t_n]) =$

$$w_n^m \exp(a_m I[y_m(\boldsymbol{x}_n) \neq t_n])$$

- The term $\exp\left(-\frac{a_m}{2}\right)$ is independent of n and, thus, can be discarded.
- Finally, $sign(f_m(\boldsymbol{x}_n)) = sign\left(\frac{1}{2}\sum_{i=1}^m a_i y_i(\boldsymbol{x}_n)\right) = sign(\sum_{i=1}^m a_i y_i(\boldsymbol{x}_n))$

Pros and Cons of AdaBoost

- The selection of the exponential error function, makes AdaBoost a very simple algorithm with easy implementation.

- It is sensitive to large negative outliers of the term $t_n y_m(\boldsymbol{x}_n)$ (in case there is not suitable activation function to get +1 or -1).

- Exponential error function can not be interpreted as a well defined probabilistic model.

- AdaBoost does not generalize easily to $c > 2$
    - There are many alterations of the algorithm, though.

Decision Trees (revisited)

Combining models

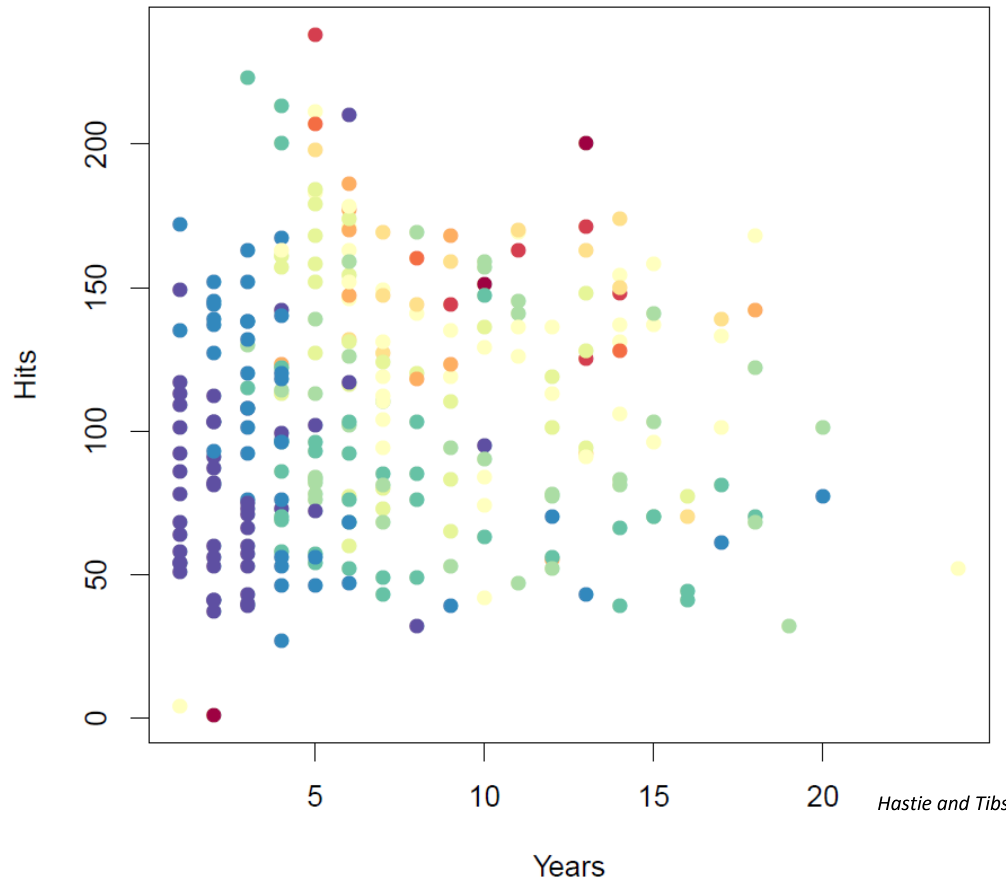
- Committees
  - Bootstrap Aggregation
  - Random Subspace methods (feature bagging)
  - Boosting
- We will now revisit the Decision Tree model and will extend the approach to Random Forests.
- In essence, decisions trees with bootstrapping and random subspaces methods lead to the random forests realization.

Decision Trees (DT)

- DT methods can be used both for regression and classification
- They result in stratification of segmentation of the feature space into a number of rectangular regions.
- The rules that segment the feature space into distinct regions can be summarized using tree, thus the term Decision Trees.
- DT are suitable for intuitive interpretation of the outcome!
- Nevertheless, they are in general weak classifiers (or regressors) and cannot compete with the supervised approaches that we have seen so far.
- Combining a large number of DT grown with bagging and/or boosting approaches can often result in great improvement in prediction accuracy at the expense of loss interpretation.

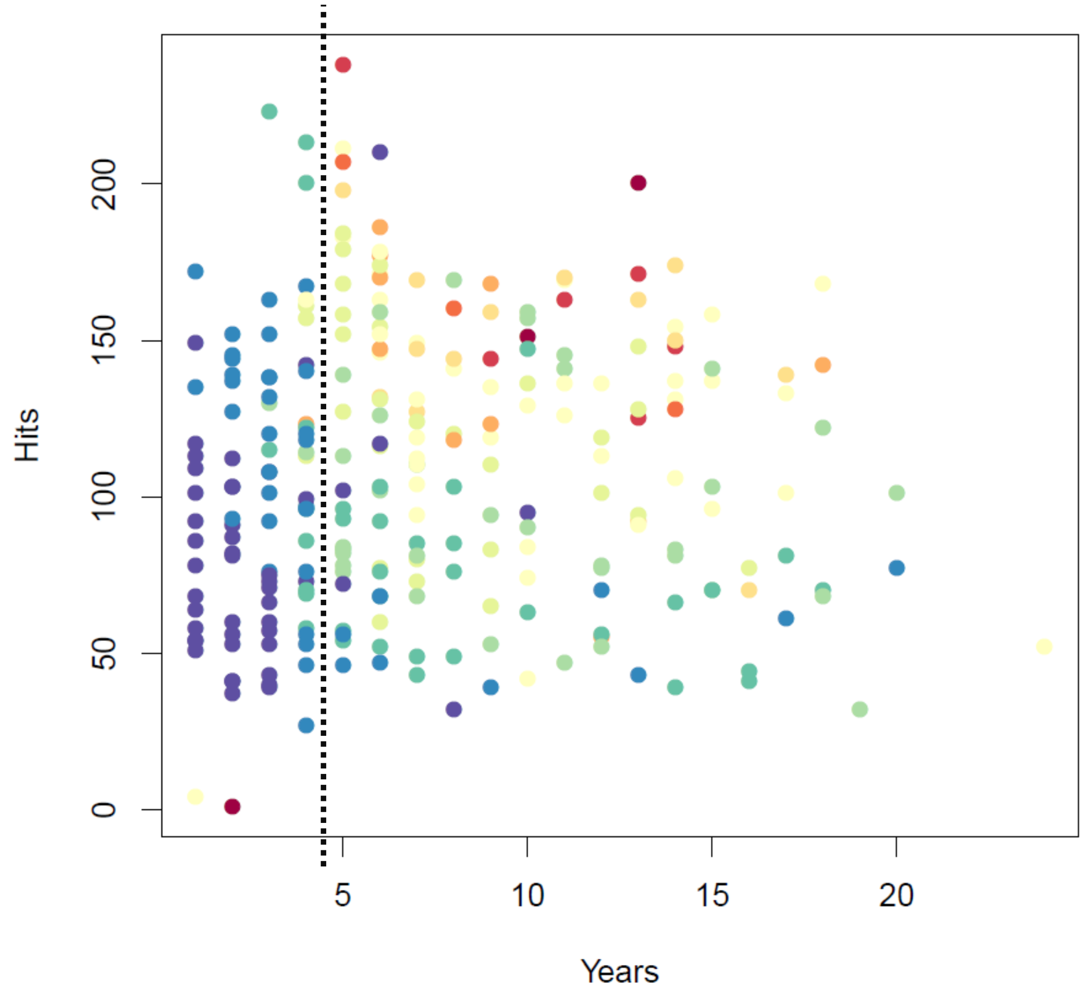
DT - Regression

- Baseball salary data
  - Salary is color-coded from low (blue, green) to high (yellow, red)

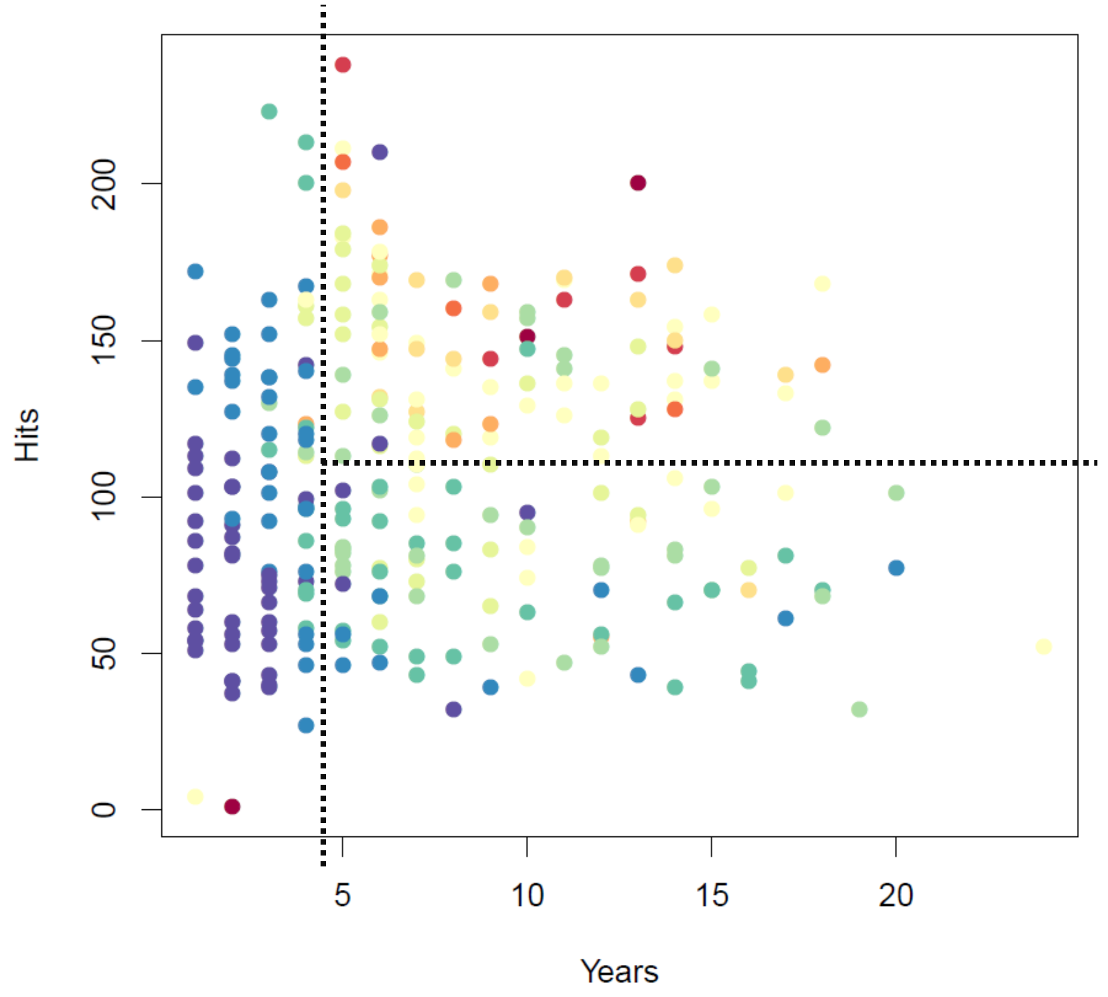


Hastie and Tibshirani

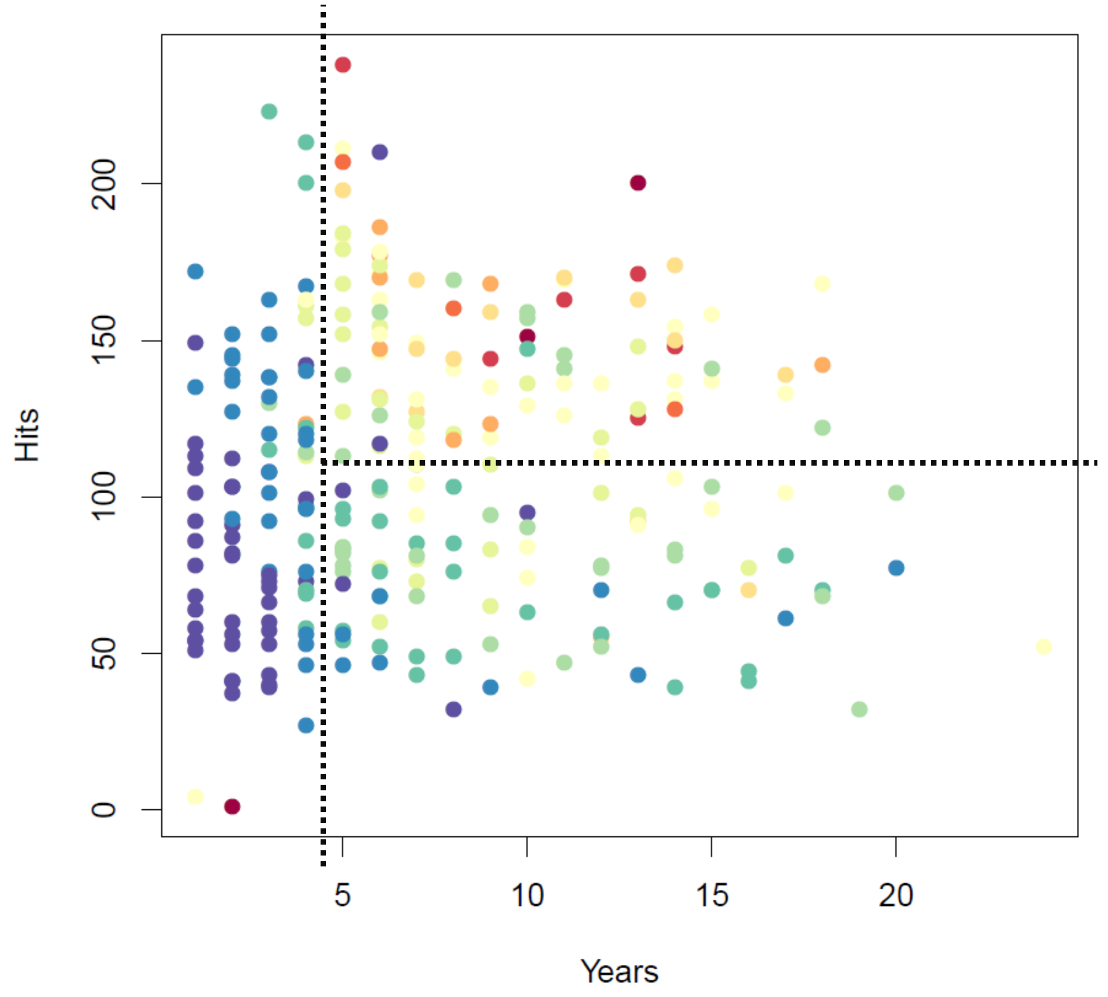
Baseball salary data

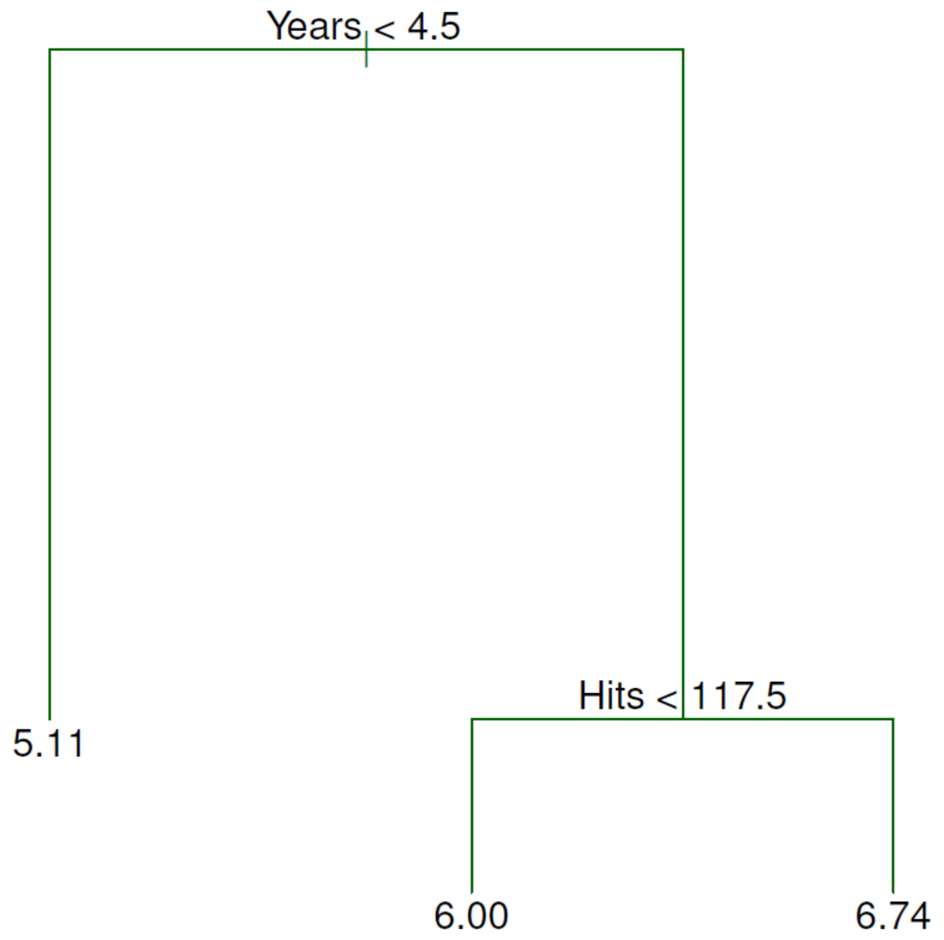


Baseball salary data



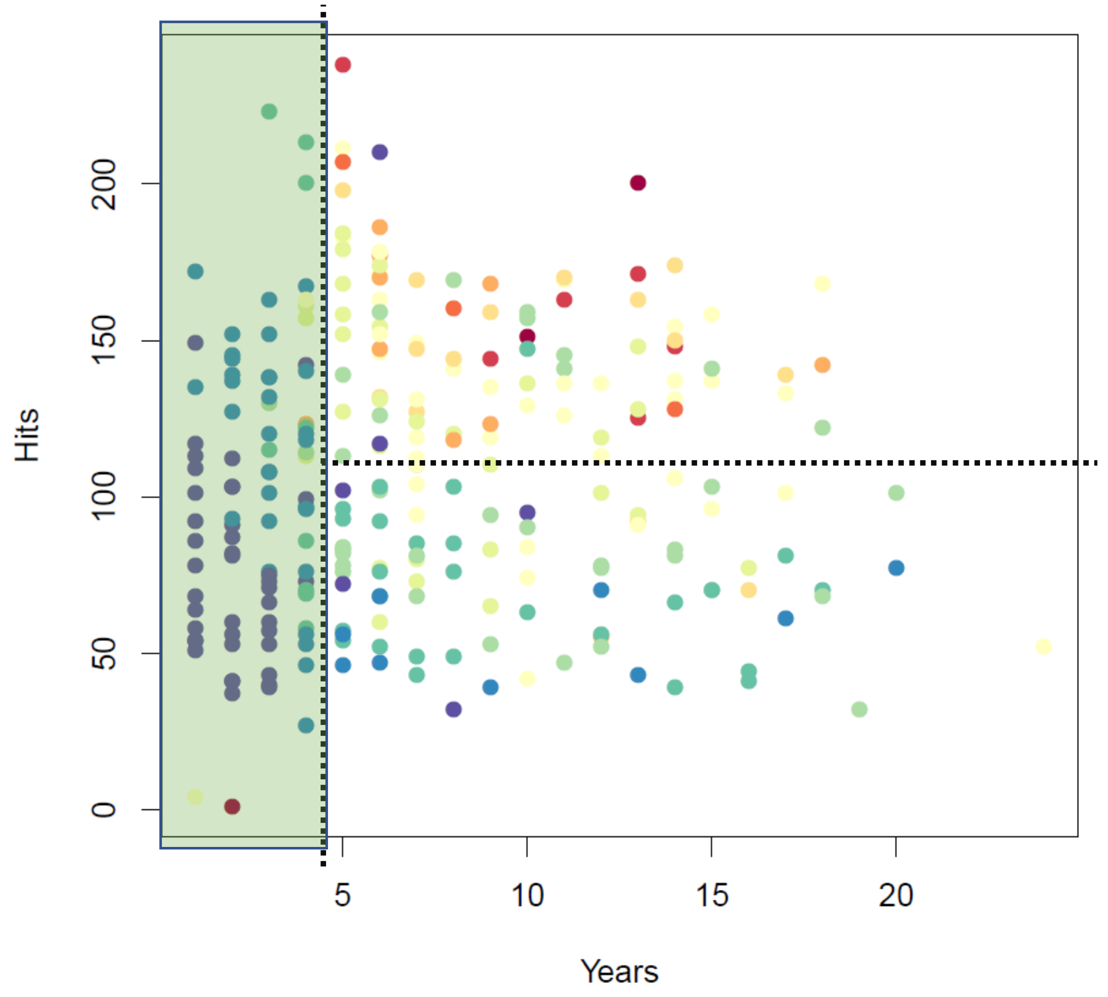
Baseball salary data

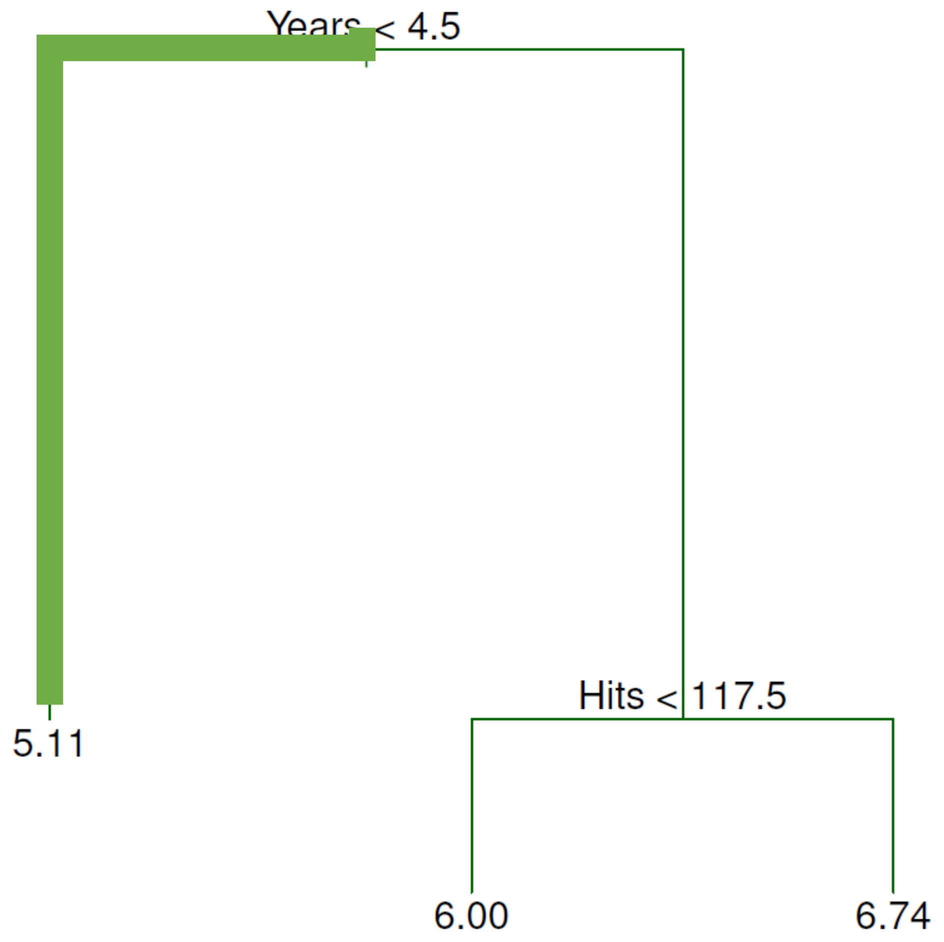




Hastie and Tibshirani

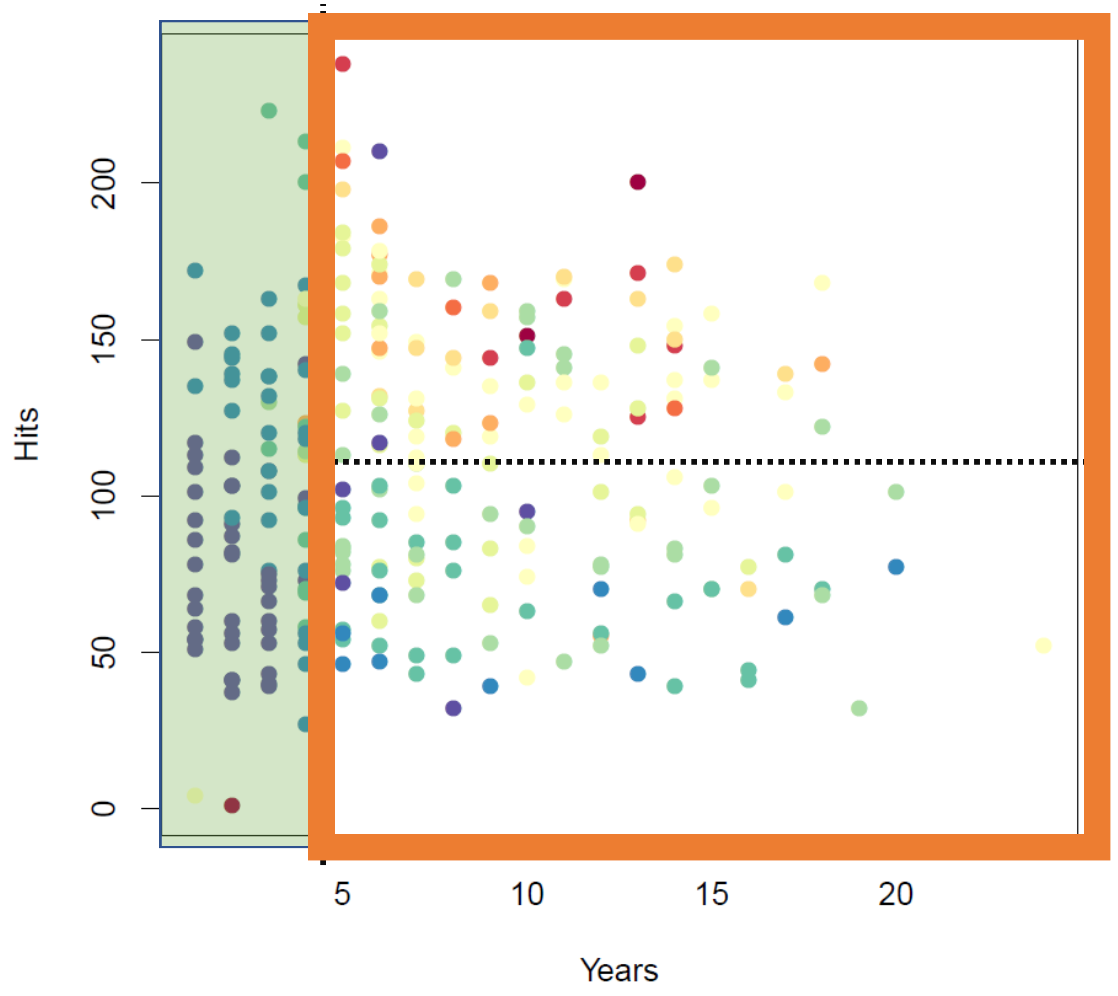
Baseball salary data

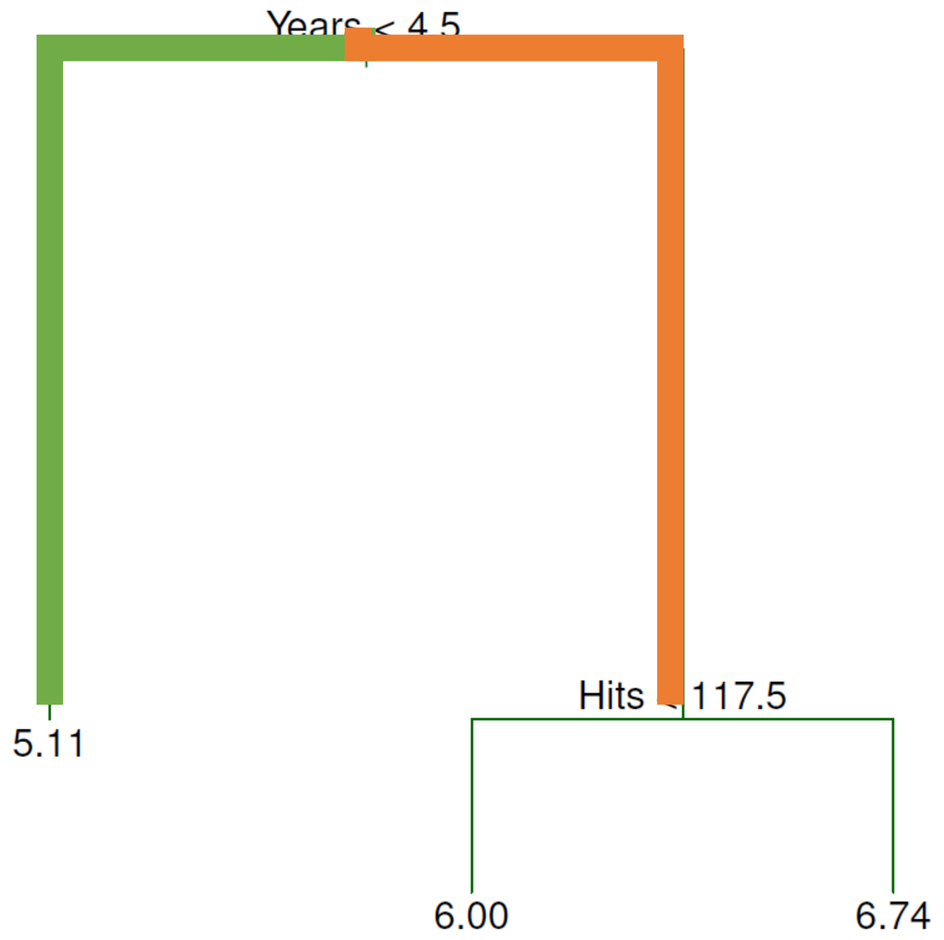




Hastie and Tibshirani

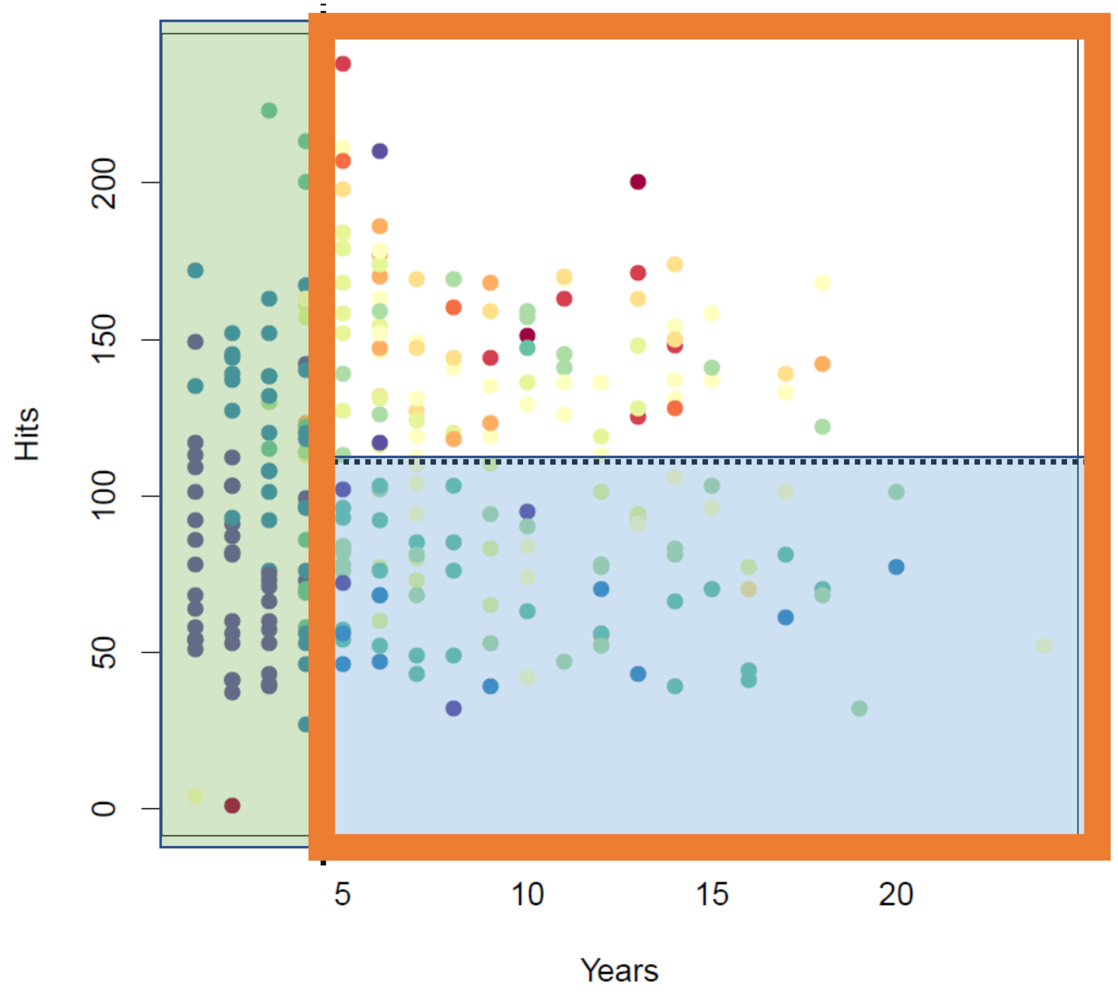
Baseball salary data

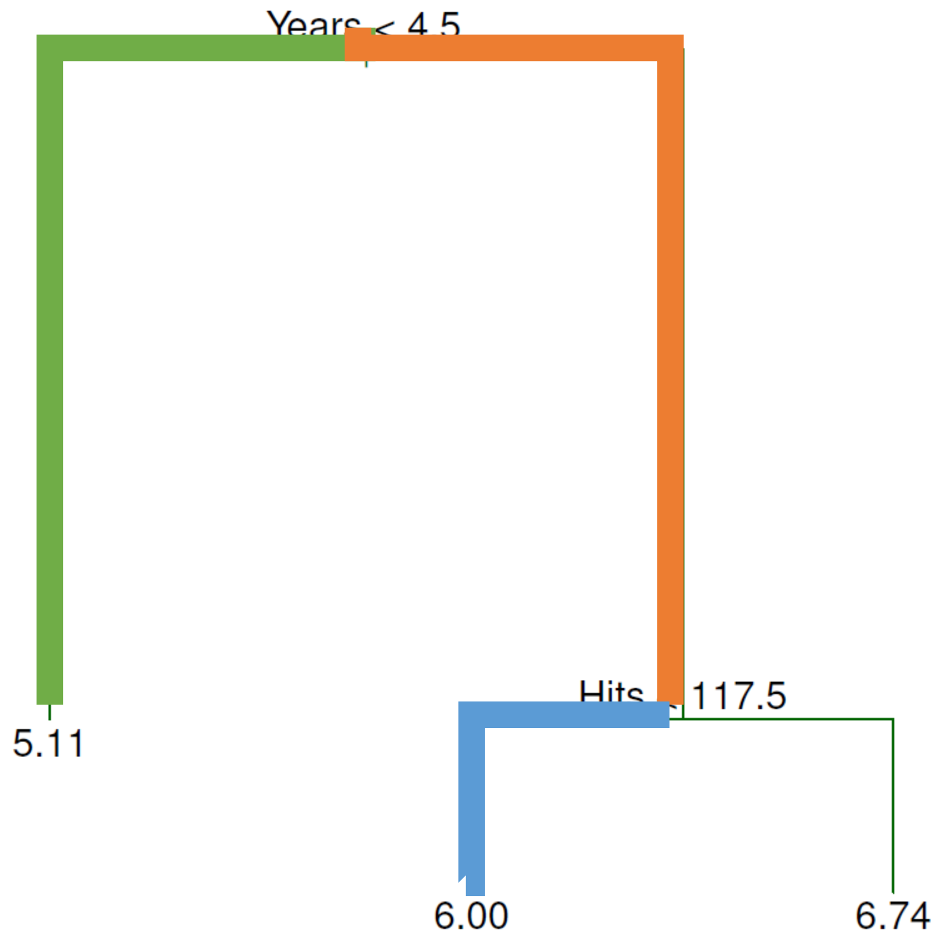




Hastie and Tibshirani

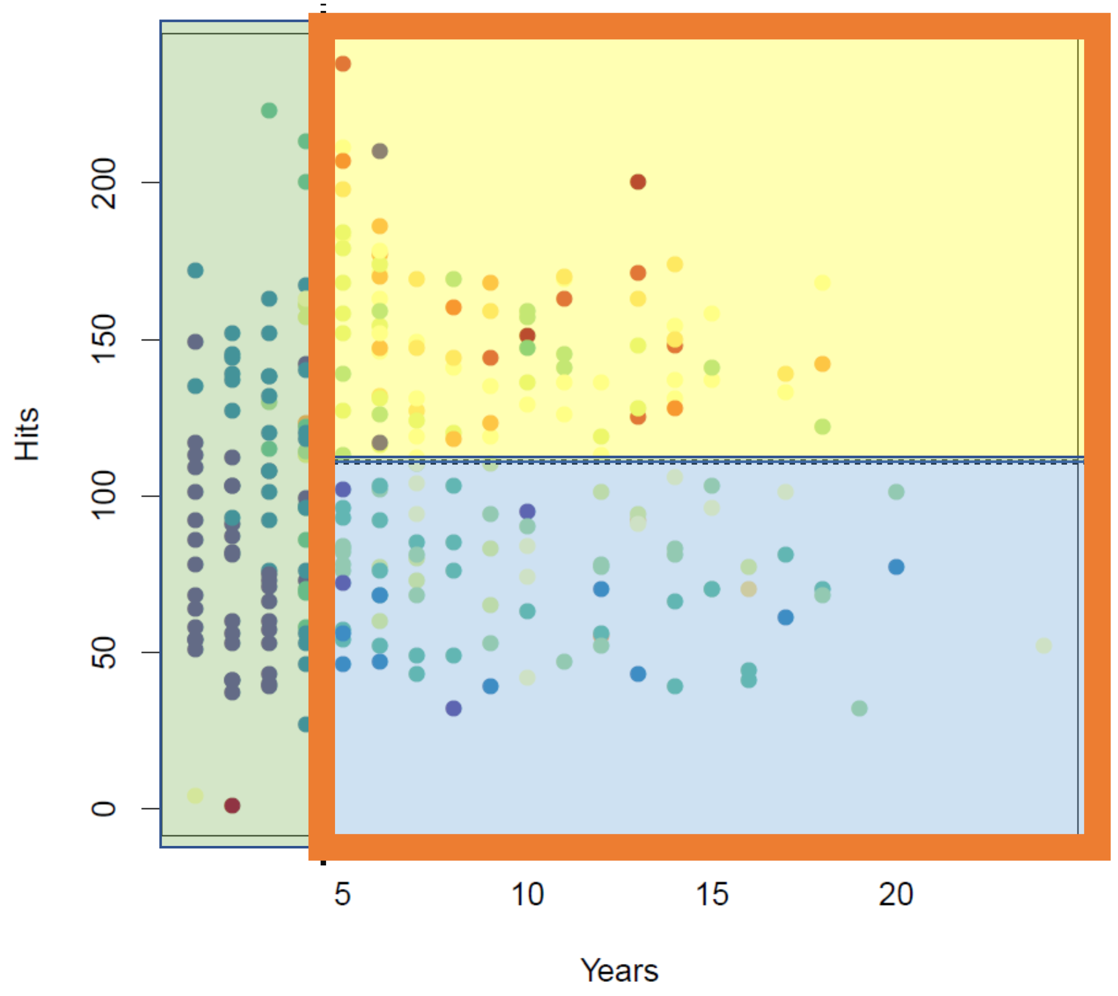
Baseball salary data

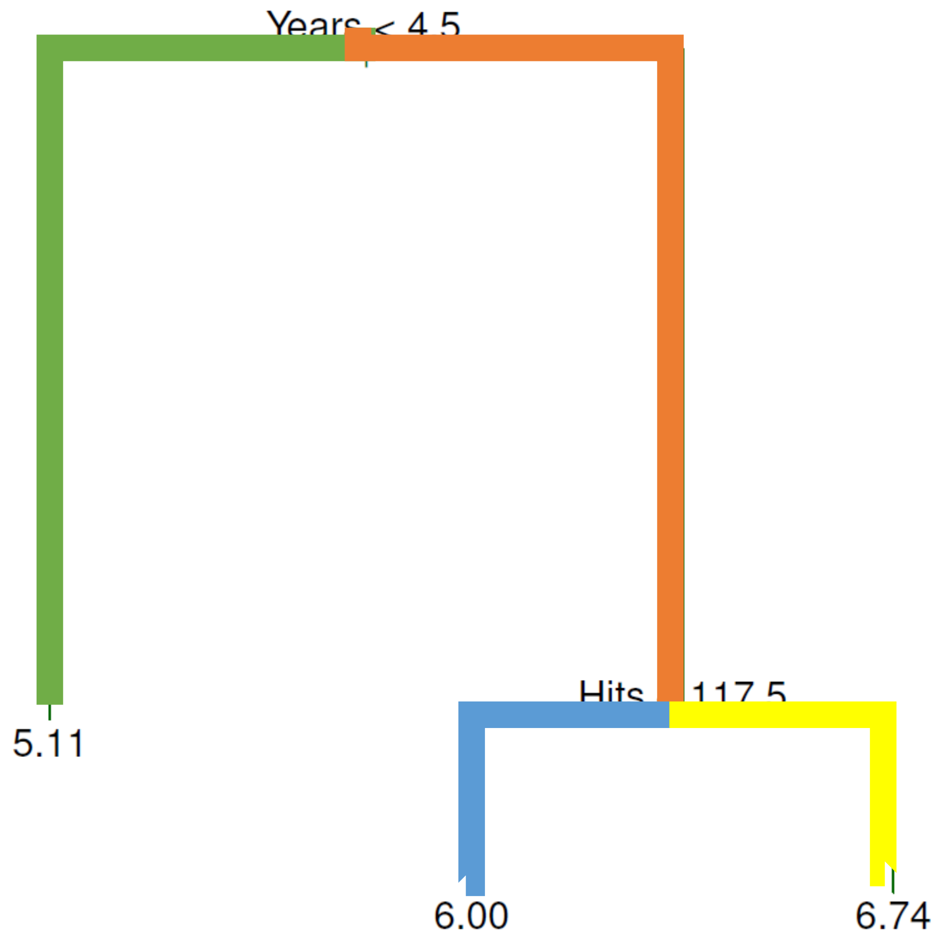




Hastie and Tibshirani

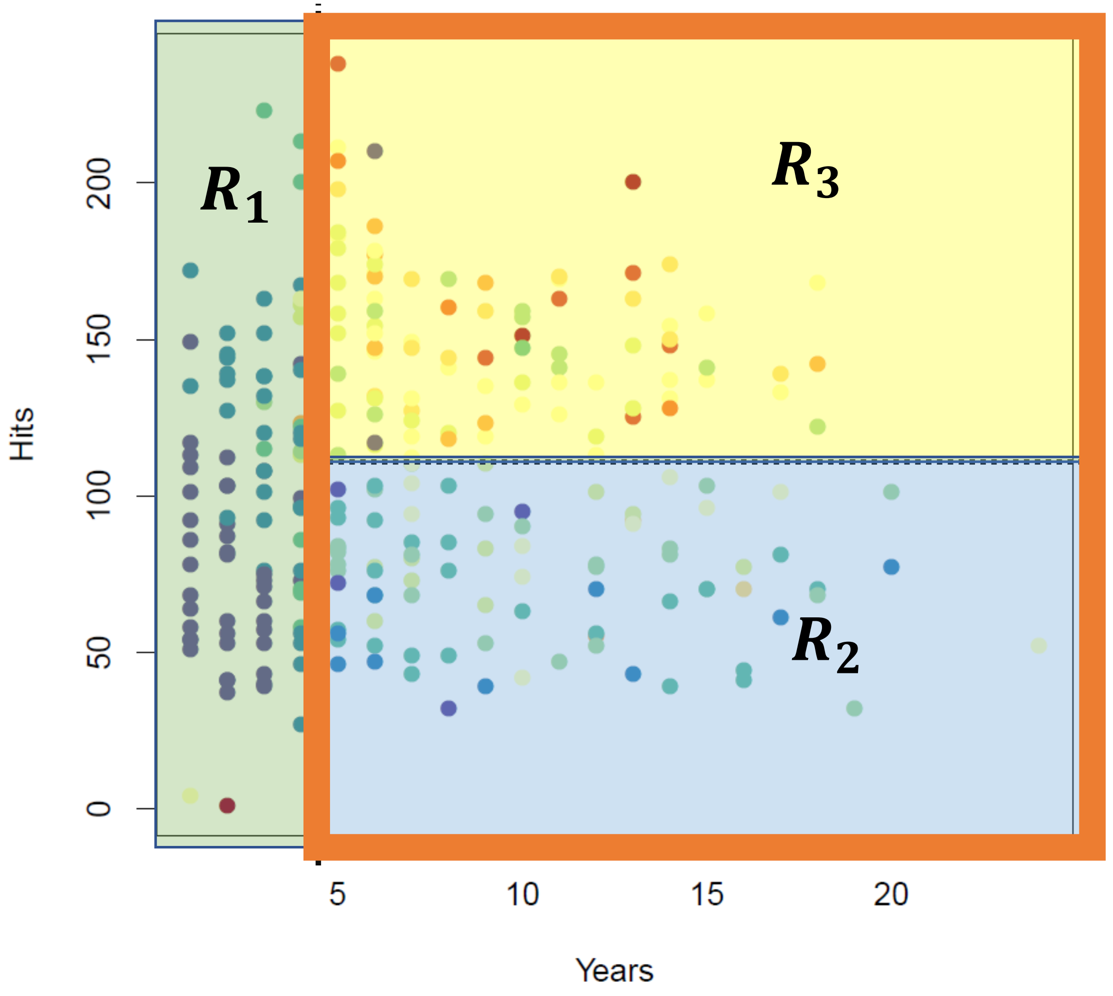
Baseball salary data

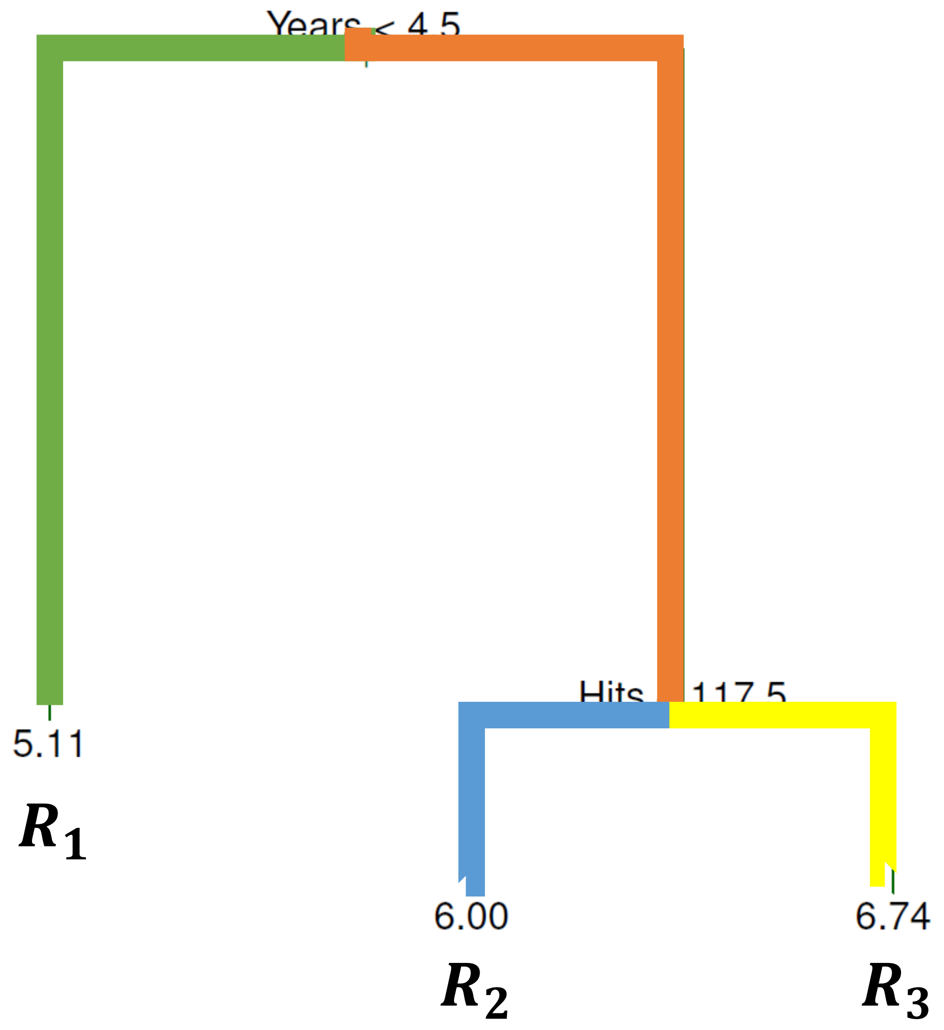




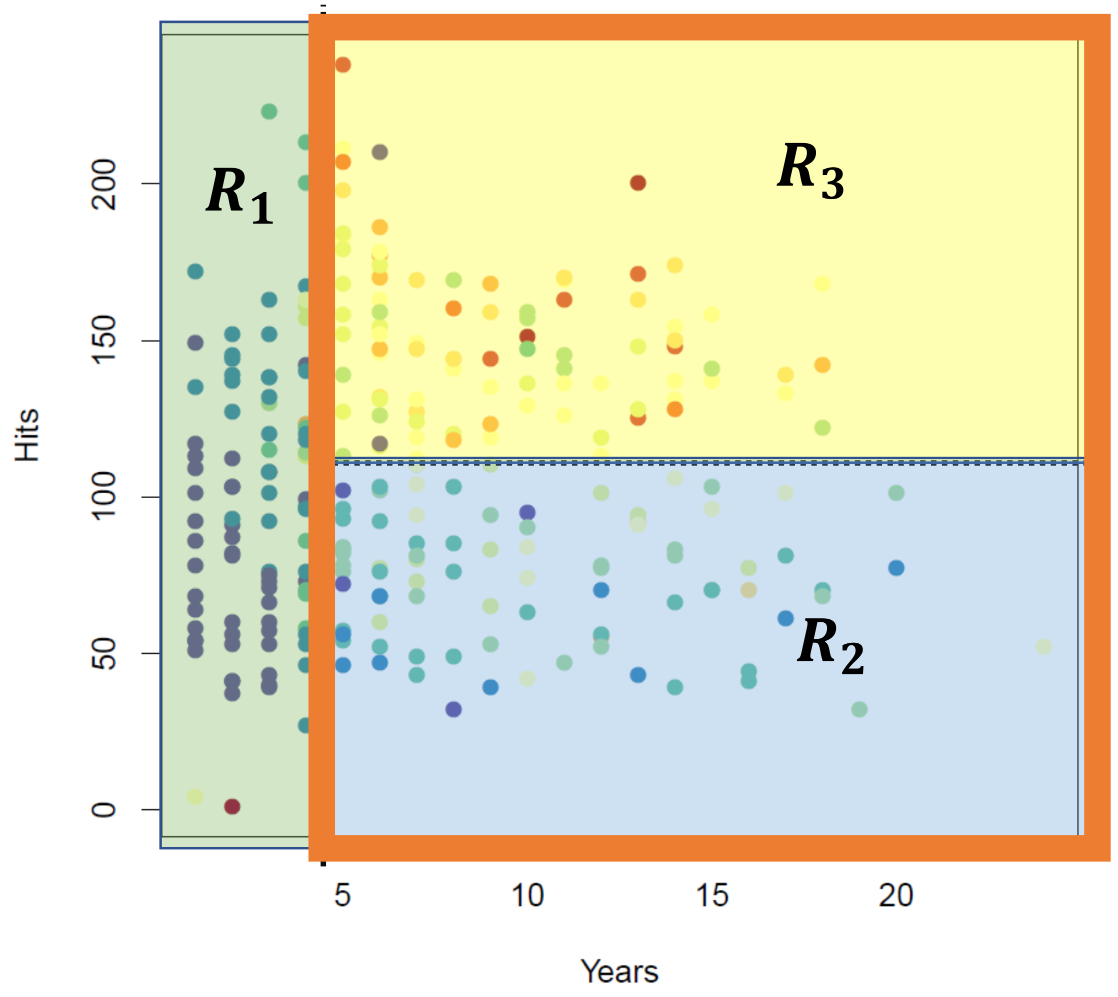
Hastie and Tibshirani

Baseball salary data





Baseball salary data



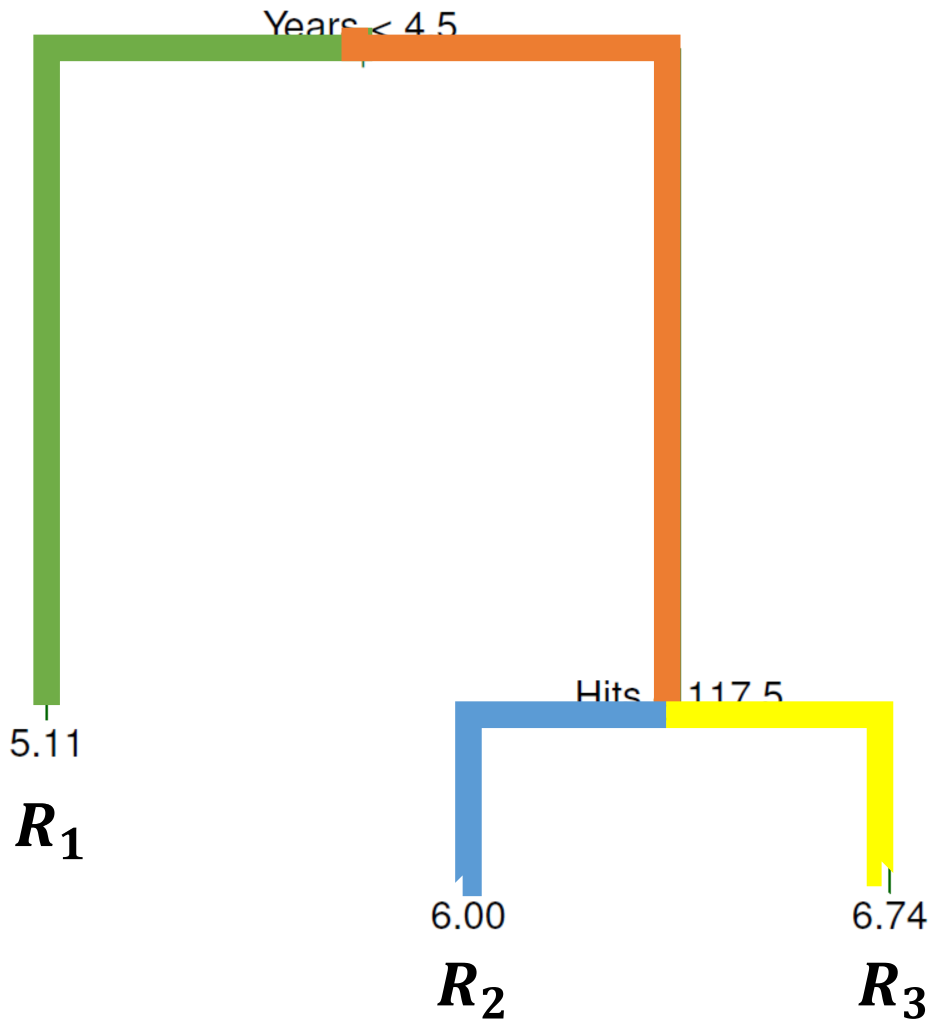
Overall the tree segments the feature space into three regions:

$$R_1 = \{\boldsymbol{x}; Years < 4.5\}$$

$$R_2 = \{\boldsymbol{x}; Years < 4.5, Hits < 117.5\}$$

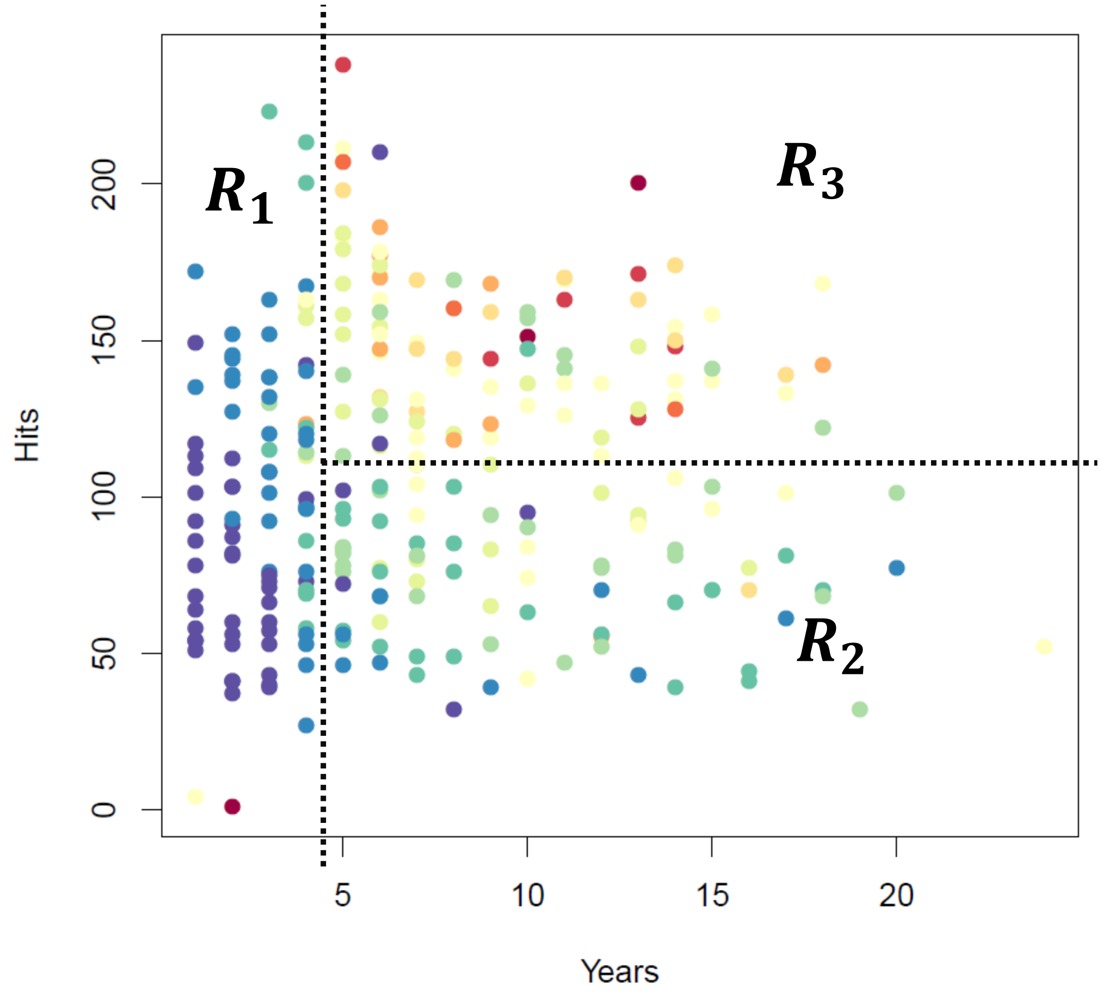
$$R_3 = \{\boldsymbol{x}; Years < 4.5, Hits \geq 117.5\}$$

Baseball salary data



- Overall the tree segments the feature space into three regions:
- $R_1 = \{\boldsymbol{x}; Years < 4.5\}$
- $R_2 = \{\boldsymbol{x}; Years < 4.5, Hits < 117.5\}$
- $R_3 = \{\boldsymbol{x}; Years < 4.5, Hits \geq 117.5\}$
- In tree terminology, R_1, R_2, R_3 are known as terminal nodes whereas the nodes where the feature space is split are referred to as internal nodes.

Interpretation of Results



- *Years* is the most important factor in determining salary. Players with less experience earn lower salaries than more experienced players.

- Given that a player is less experienced, the number of *Hits* that he made in the previous year seems to play little role in his salary.

- Among players who have been in the major leagues for five or more years, the number of *Hits* does affect salary

Tree-building process - Regression

- Divide feature space into high-dimensional rectangles.
- The goal is to find rectangular regions $R_1, R_2, \ldots, R_J$ minimize error function:

$$E = \sum_{j=1}^{J} \sum_{i \in R_j} \left(x_i - \bar{x}_{R_j} \right)^2$$

where $\bar{x}_{R_j}$ is the mean of the training samples within region R_j .

Tree-building process - Regression

- Divide feature space into high-dimensional rectangles.

- The goal is to find rectangular regions $R_1, R_2, \dots, R_J$ minimize error function:

$$E = \sum_{j=1}^{J} \sum_{i \in R_j} \left(x_i - \bar{x}_{R_j}\right)^2$$

where $\bar{x}_{R_j}$ is the mean of the training samples within region R_j .

- Well… this is computationally infeasible! Thus we follow a greedy approach:
  - top-down approach: successively split feature space; each split is indicated via two new branches further down on the tree.
  - At each step, best split is made at that particular step, instead of looking ahead and picking a split that will lead to a better tree in some future step. (greedy approach)

Tree-building process - 2

- First select the feature x_i and the cutpoint s in order to split the feature space into two regions: $\{\boldsymbol{x}: x_i < s\}$ and $\{\boldsymbol{x}: x_i \geq s\}$ for which E is minimized.

Tree-building process - 2

- First select the feature x_i and the cutpoint s in order to split the feature space into two regions: $\{\boldsymbol{x}: x_i < s\}$ and $\{\boldsymbol{x}: x_i \geq s\}$ for which E is minimized.

- Repeat the process, i.e., best feature and best cutpoint to further minimize E (largest decrease) within each the segmented regions. Split the corresponding region.

Tree-building process - 2

- First select the feature x_i and the cutpoint s in order to split the feature space into two regions: $\{\boldsymbol{x}: x_i < s\}$ and $\{\boldsymbol{x}: x_i \geq s\}$ for which E is minimized.

- Repeat the process, i.e., best feature and best cutpoint to further minimize E (largest decrease) within each the segmented regions. Split the corresponding region.

- Each time, we split one of the two previously identified regions.

Tree-building process - 2

- First select the feature x_i and the cutpoint s in order to split the feature space into two regions: $\{\boldsymbol{x}: x_i < s\}$ and $\{\boldsymbol{x}: x_i \geq s\}$ for which E is minimized.

- Repeat the process, i.e., best feature and best cutpoint to further minimize E (largest decrease) within each the segmented regions. Split the corresponding region.

- Each time, we split one of the two previously identified regions.

- The whole process continues until a stopping criterion is reached.
  - A specific value for E
  - number of samples in each regions above a threshold.

Tree-building process - 2

- First select the feature x_i and the cutpoint s in order to split the feature space into two regions: $\{\boldsymbol{x}: x_i < s\}$ and $\{\boldsymbol{x}: x_i \geq s\}$ for which E is minimized.

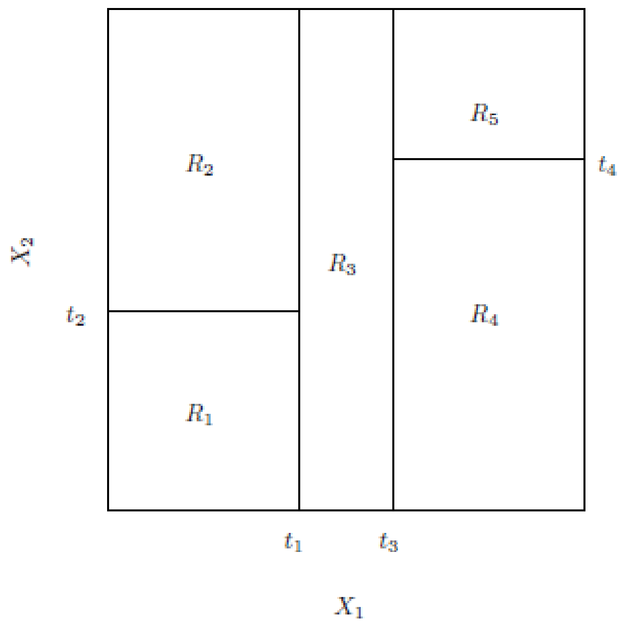
- Repeat the process, i.e., best feature and best cutpoint to further minimize E (largest decrease) within each the segmented regions. Split the corresponding region.

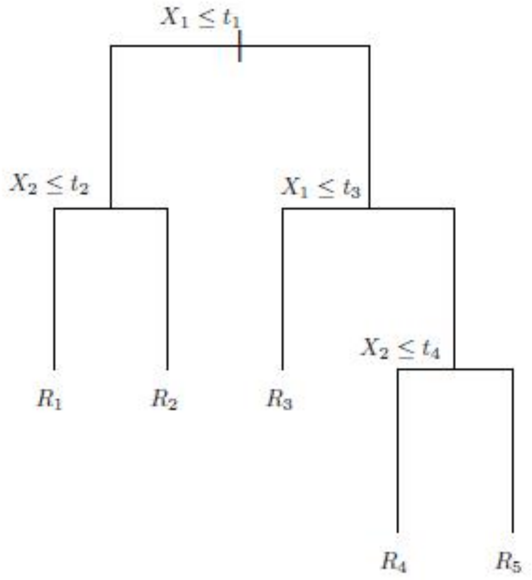
- Each time, we split one of the two previously identified regions.

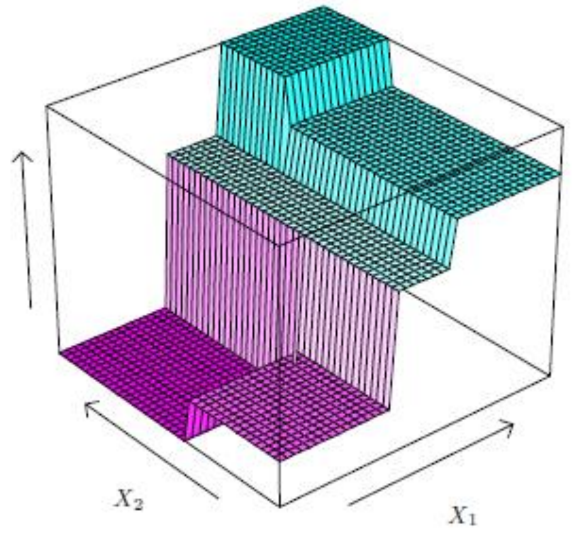
- The whole process continues until a stopping criterion is reached.
  - A specific value for E
  - number of samples in each regions above a threshold.

- Predict the response for a given test sample by assigning the mean (regression) of the training samples in the region where the test sample belongs.

Example







Hastie and Tibshirani

- Left: The output of recursive binary splitting on a two-dimensional example.
- Center: A tree corresponding to the partition in the top right panel.
- Right: A perspective plot of the prediction surface corresponding to that tree.

Tree Pruning

- The abovementioned process may result in overfitting (really large trees). (remember with shallow trees I may get underfitted models).

- An alternate approach would be to grow a tree only so long as E decreases more than a certain threshold.

- This would result in medium sized trees but would cause the loss of larger decrease later on the training process.
    - seemingly worthless split early on in the tree might be followed by a very good split later on.

- A better strategy would be to grow a very large tree T_0 and then prune it in order to obtain a subtree that meet certain restrictions.
    - Grow T_0 and then prune it to T with a smaller number of terminal nodes $|T|$.

Decision Tree – Classification (K classes)

- Same as in the regression case but we now split the feature space by minimizing

$$E = \sum_{j=1}^{J} Q_j$$

where Q_j can be one of the following options:

- Misclassification rate: $Q_j = 1 - \max_k(p_{jk})$
- Negative cross entropy: $Q_j = -\sum_{k=1}^{K} p_{jk} ln p_{jk}$
- Gini index: $Q_j = \sum_{k=1}^{K} p_{jk}(1 - p_{jk})$

where p_{jk} represents the proportion of training observations in the jth region that are from the kth class.

Ensemble methods

- DT are easily interpretable and nice to visualize. It is said that DTs more closely mirror human decision-making than other ML approaches.

- Unfortunately, DT have not the same level of predictive accuracy as other classification approaches we have seen so far.

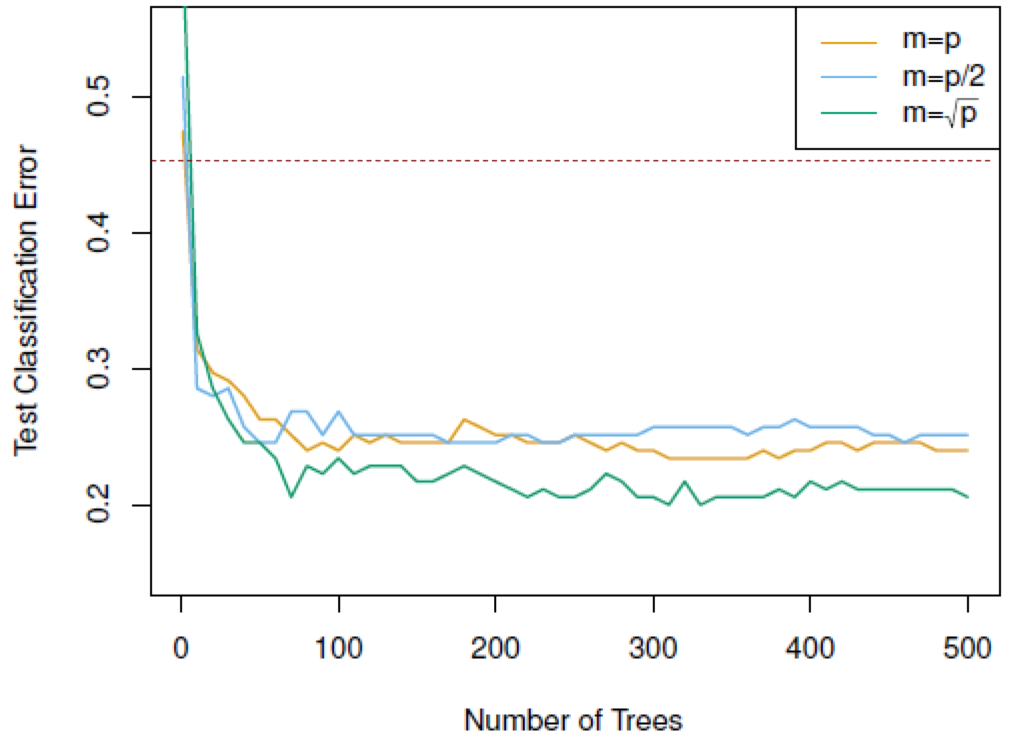
- Solution: Create ensembles of DTs!
  - Bagging/bootstrap aggregation with trees
  - Random Forests: bagging + random subspace methods
  - Boosting

Random Forests

- Bootstrapping can lead to highly correlated DT.
  - If there are a few very strong features in the dataset all bootstrapped trees will use these features to split space.
- Solution:
  - Build an ensemble of trees by bootstrapping the dataset.
  - Use a feature bagging/random subspace method: for each tree, every time a split is under consideration, a random selection of m out of p features is chosen as a split candidate.
  - At each split a new selection is made

Random Forests - Example

- Gene expression dataset
- Task: classify cancer type based on $p = 500$ gene expressions (features).
- Random Forests $(m < p)$
- Just bootstrapping $(m = p)$
- Random Forests exhibit small improvement over just bootstrapping.
- Single tree performance (red dashed line) is much worse (45.7%)



Hastie and Tibshirani

Sum up

- We can combine multiple models using different methods
  - Bootstrapping/Bagging
  - Random subspace methods/Feature Bagging
  - Boosting

- Trees are weak classifiers but with nice interpretability.

- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees.

- Random Forests and boosting are among the state-of-the-art methods for supervised learning but their results can be difficult to interpret.
  - The more complexity we add the more difficult it becomes to interpret out results.

ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF ENGINEERING

Questions?

Pattern Recognition & Machine Learning

Model Combination - Trees

Fall Semester