APMA E4990.02: Decision Trees and Random Forests

Introduction to nonparametric models

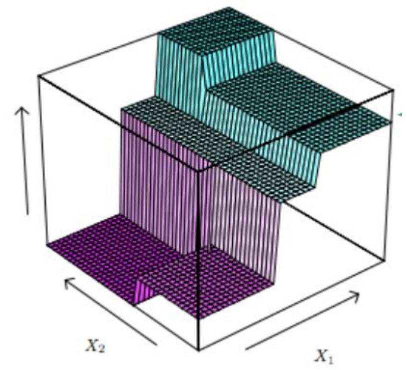
Announcements

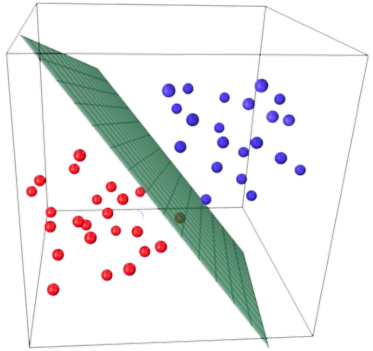
- Homework 2 due next week (it’s short).
- Homework 0 grades are up - good job!
- Today: Non-parametric models.
- Start looking into Big Query over the break.

Recall General Goal

More generally, joint random variables $(\mathbf{x}, y)$ with distribution $P(\mathbf{x}, y)$
we week to minimize:

$$\mathcal{L}(\hat{f}) := \mathbb{E}_P L(y, \hat{f}(\mathbf{x}))) \sim \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$





We will now introduce non-parametric models. That is, models that don't depend on learning a parameter, but rather how to intelligently split space.

$$\hat{f}(\mathbf{x}_i) = \mathbf{1}_{R_t(\mathbf{x}_i)}(\mathbf{x}_i) \text{ for } \mathbf{x}_i \in R_{t(i)}$$

$$\hat{f}(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$$

Decision Tree Classifiers

Decision Tree Classifiers

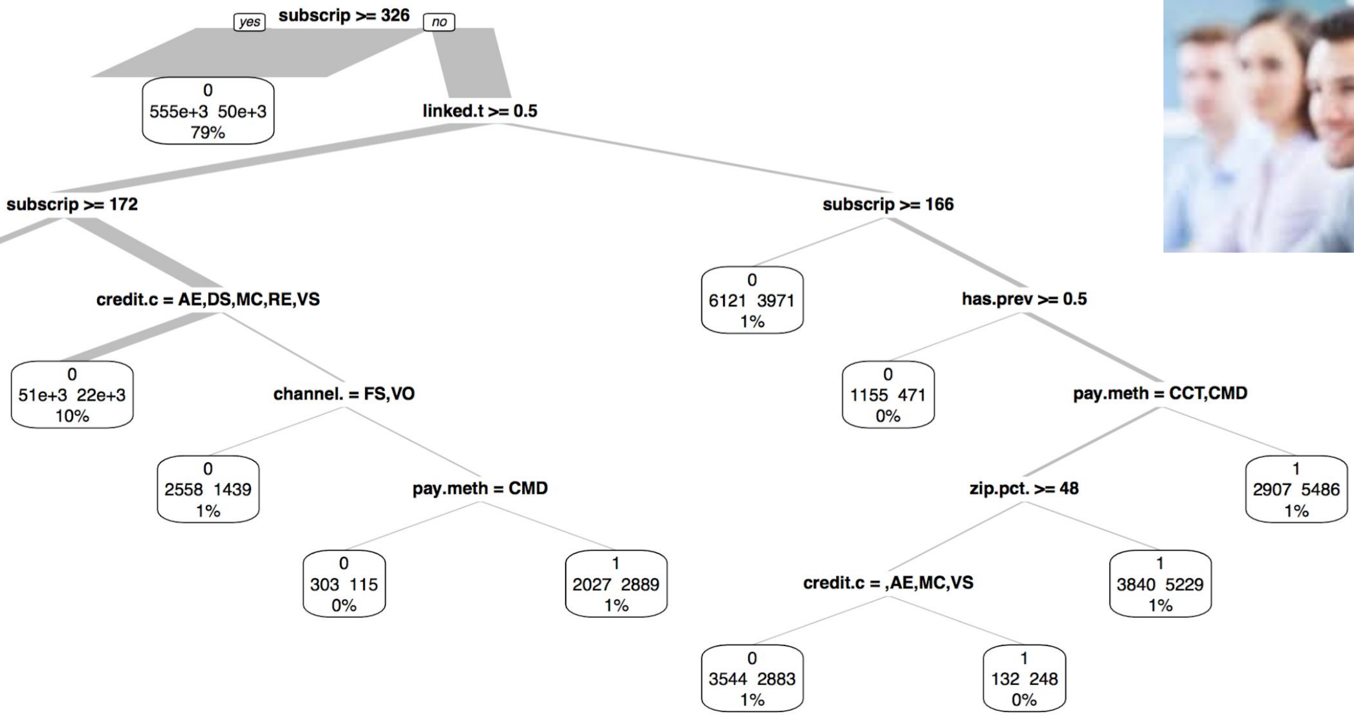
- Decision trees are simply a recursive set of decisions that result in bucketing every collection of attributes into a final node.

- Very easy to overfit. With categorical features, we can always find a deep enough decision tree to fit training data ‘exactly’.

- More like a human set of a decisions, therefore highly interpretable.

- Gives a robust, well defined notion of ‘most important features’.

The New York Times Churn Model API

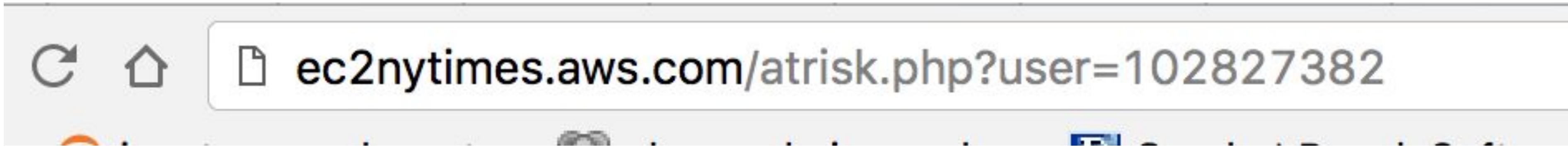


Sugar Care Center at The New York Times

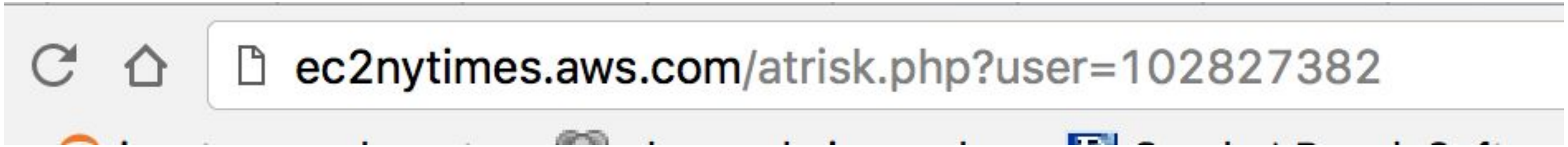
- Wrote an algorithm that aggregated the **‘reasons’ why somebody is at risk (**just collects them along the path down the decision tree).

- Based on the category they fell into, different actions were taken by people on the phone.

- The model didn’t perform as well as the linear model or the full Random Forest, but it was interpretable! And that’s often more important in industry.


ec2nytimes.aws.com/atrisk.php?user=102827382

Usage



ec2nytimes.aws.com/atrisk.php?user=102827382

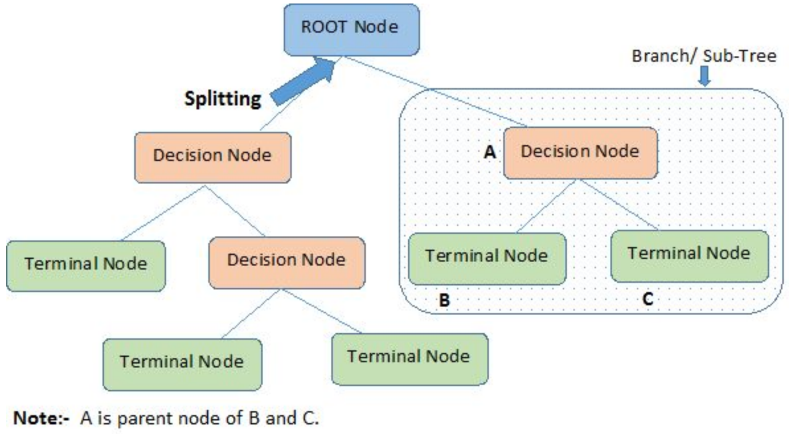
Risk score: 85%

Reasons:
Complained in past 60 days more than 3 times.
Subscriber for less than 3 months.
Zip code with delivery problems.

Advantages/Disadvantages?

- Decision Trees are **great for interpretability** - they’re even more **interpretable than linear models!**
- The rules are **much more ‘human like’**, meaning an iterated sequence of decisions to reach an outcome.
- Overall I find **Random Forests/Decision Trees to be better suited for classification than regression** - harder to approximate continuous values with discrete approximations.
- Decision Trees **can tend to over fit**, and often don’t have as much predictive power as linear models or Random Forests.
- Great variable interpretation.

Some terminology

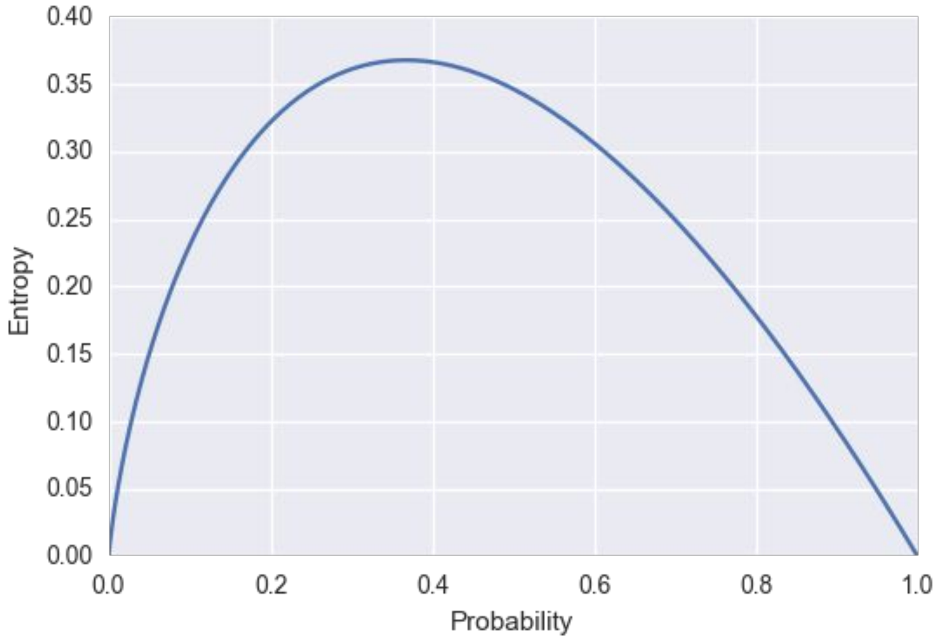


1. **Root Node:** It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
6. **Branch / Sub-Tree:** A sub-section of entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.

How do we construct the tree?

$$H = -\sum_{k=1}^{N} p_k \log p_k$$

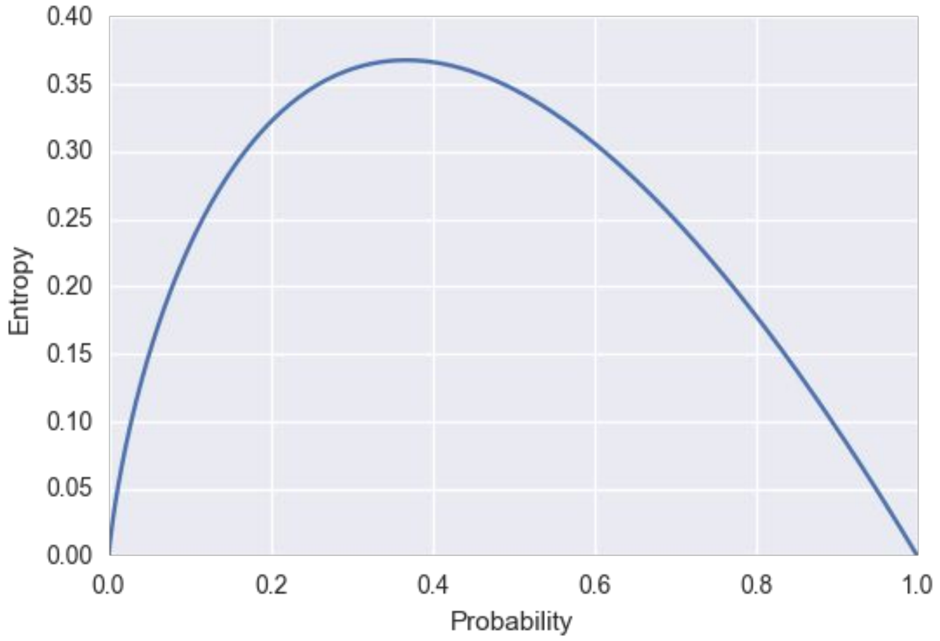
We now seek to minimize **information entropy** at each step. But what is it?



Information Entropy

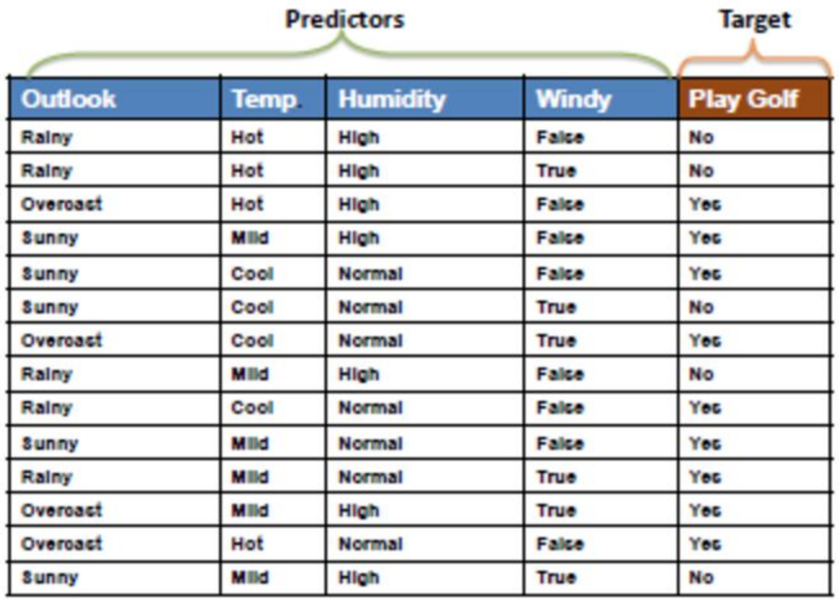
$$H = -\sum_{k=1}^{N} p_k \log p_k$$

- When **p is 1 or 0,** it means we have **no uncertainty** - we have all information available.
- When **p is ½, entropy is maximized** - we have the least information

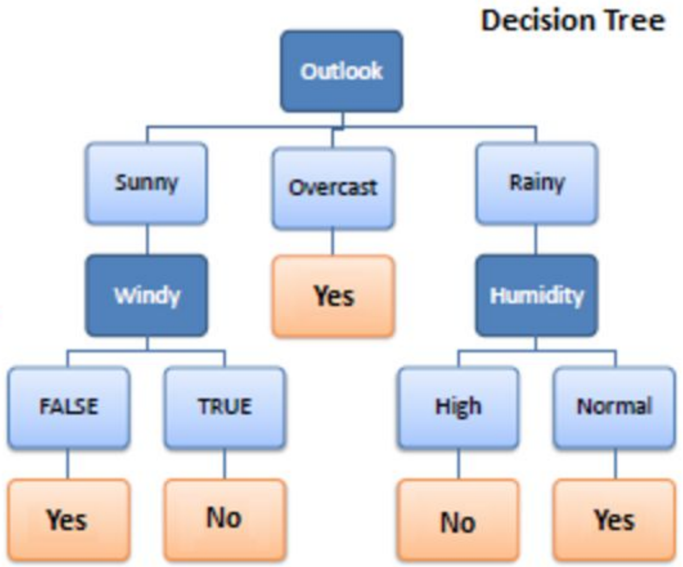


Recall thermodynamics principles if you have studied physics.

Example: Predict if somebody will play golf



Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



Class Balances: Yes: 9 entries, No: 5 entries

Overall Entropy of the dataset

$$Y = \text{Play Golf or Not}$$

$$H(Y) = -\frac{9}{14}\log\frac{9}{14} - \frac{5}{14}\log\frac{5}{14} = 0.94$$

Play Golf	
Yes	No
9	5

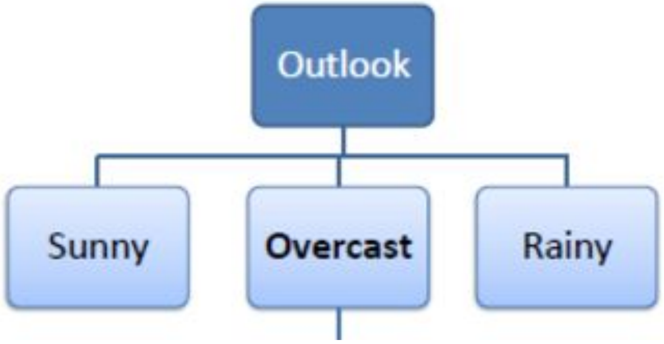
Conditional Entropy

$$H(Y|X) = \sum_x H(Y|X = x)p(x)$$
$$= \sum_x \sum_y p(y, x) \log p(y|X = x)$$
$$= \sum_{x,y} p(y, x) \log \frac{p(x)}{p(y, x)}$$

- The conditional entropy represents the entropy in the dataset conditioned on a particular variable
- When choosing the first variable to split on, we want to find the one that has the least entropy, or equivalently maximizes the information gain.

Minimize Entropy at Split

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



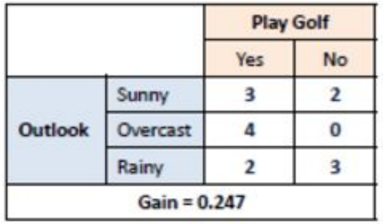
$$H(Y|X) = \sum_x H(Y|X = x)p(x)$$

$$H(Y|O) = p(\text{sunny})H(Y|\text{sunny}) + p(\text{overcast})H(Y|\text{overcast}) + p(\text{rainy})H(Y|\text{rainy})$$

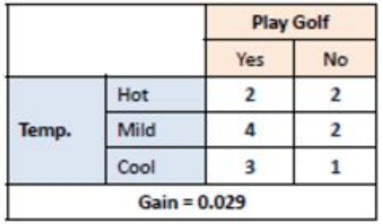
<u>Example:</u> $H(Y|\text{ Sunny }) = -\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}$

$$H_{Gain}(Y|O) = H(Y) - H(Y|O) = 0.94 - 0.693 = 0.247$$

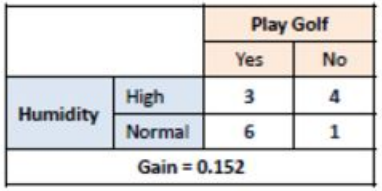
Comparing Entropy Gains for Split



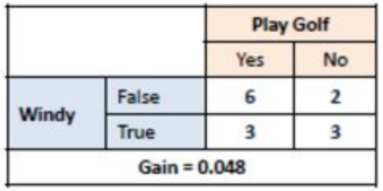
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			



		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

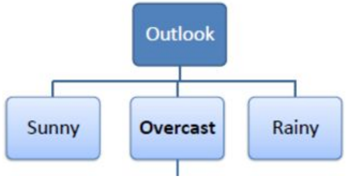


		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

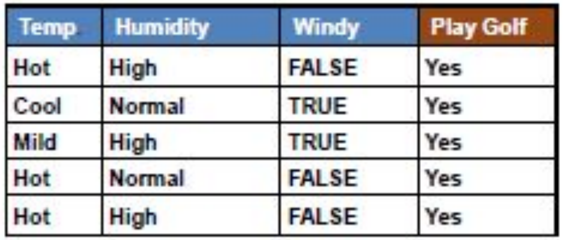
- We compute the information gain from each split. Then we choose the one which is the largest.

- Intuitively, we are choosing the split which reduces uncertainty the most - gives the 'purest' classes.

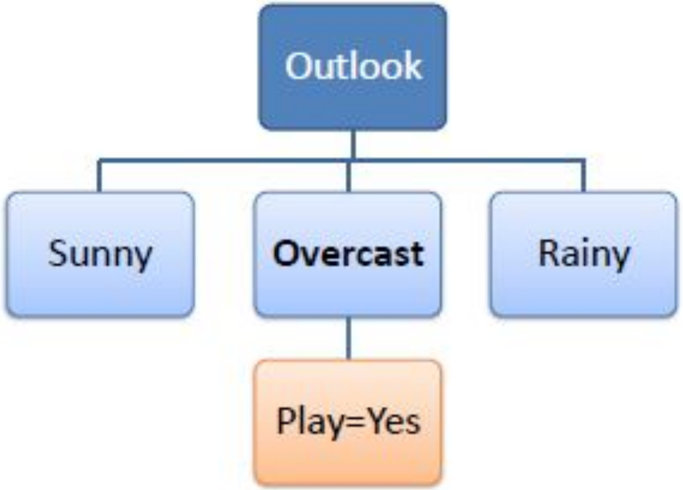
Outlook has the best split overall, so we split on that attribute.



Overcast is a pure class - terminal node

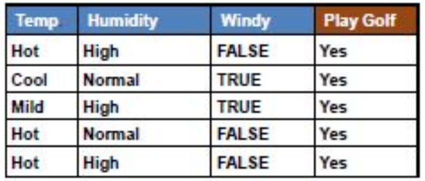


Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes
Hot	High	FALSE	Yes

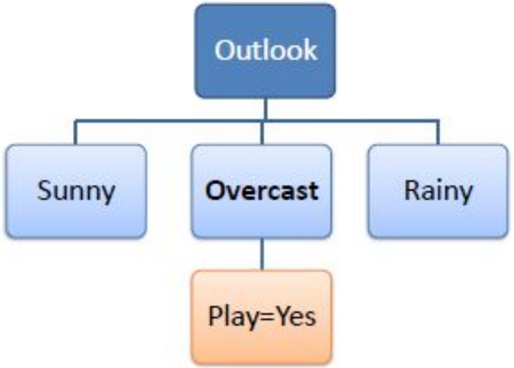


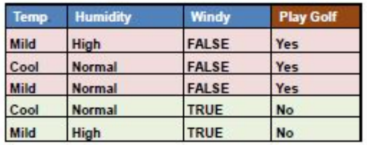
For each other node, we must continue to split until we have a terminal node (pure class).

Continue Recursively

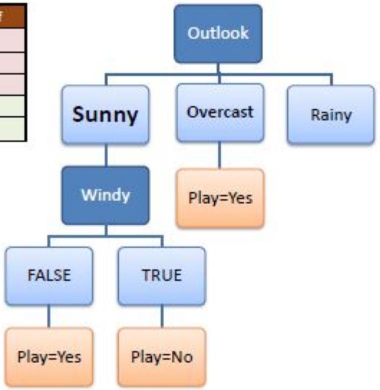


Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes
Hot	High	FALSE	Yes





Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



For each other node, we must continue to split until we have a terminal node (pure class).

Final Decision Tree

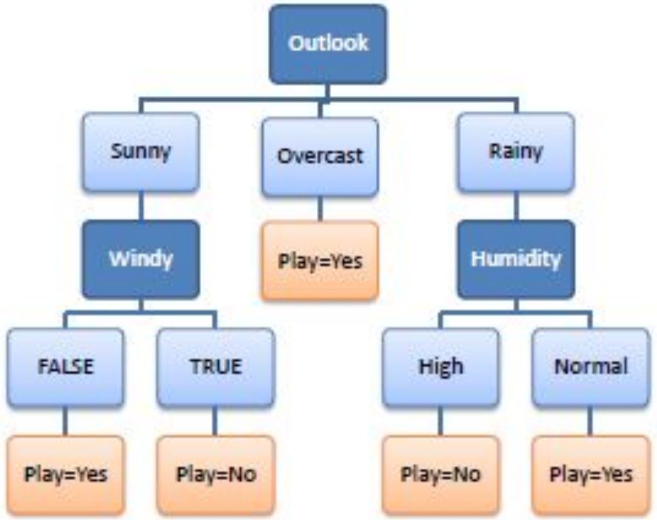
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



Summary

- The **information gain** is based on the **decrease in entropy** after a dataset is **split on an attribute**.

- Our **first attribute** is chosen to have the largest **information gain**, or largest **decrease in entropy**.

- Subsequent splits are **chosen recursively** on each sub-node selected from the first split, repeating the above.

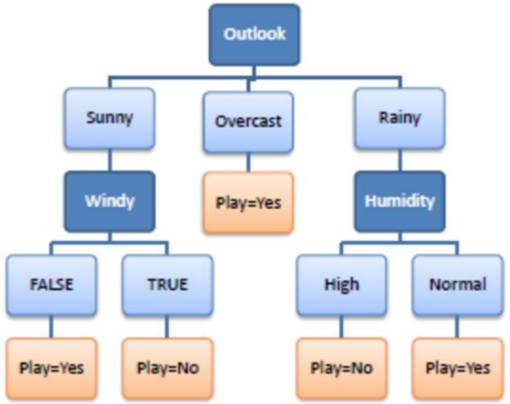
- The process is **continued** until we have a **pure class.**

- We **prevent over-fitting** by **optimizing performance on test data** using the **depth of the tree as a parameter** (will show examples when we cover a regression example next). We didn't do this here! So we fit our decision tree perfectly to our data (ROC of 1.0 on training data).

Variable Importance

Variable Importance

- We record the total amount that the entropy is decreased due to splits over a given predictor throughout the tree (add them).
- A large value indicates an important predictor.



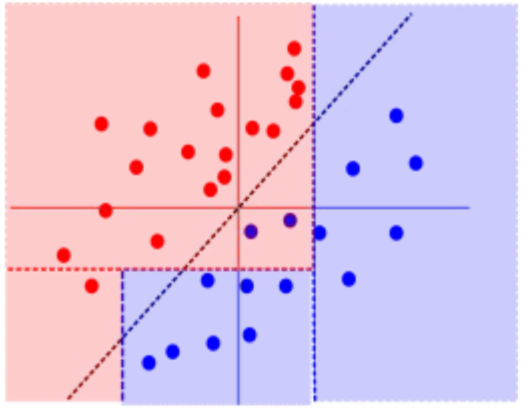
Common Questions

- **What about real valued inputs?** In general these splits are binary, then iterated through levels in the tree.

- **Are splits always binary?** In Python, yes, but not for a general algorithm. Any 3 way split can be seen as a one versus all split (ie. A&B or C versus A, B or C). It depends on the algorithm. **This can lead to suboptimal solutions.**

Decision Tree Classifier vs Logistic Regression

Which works better?

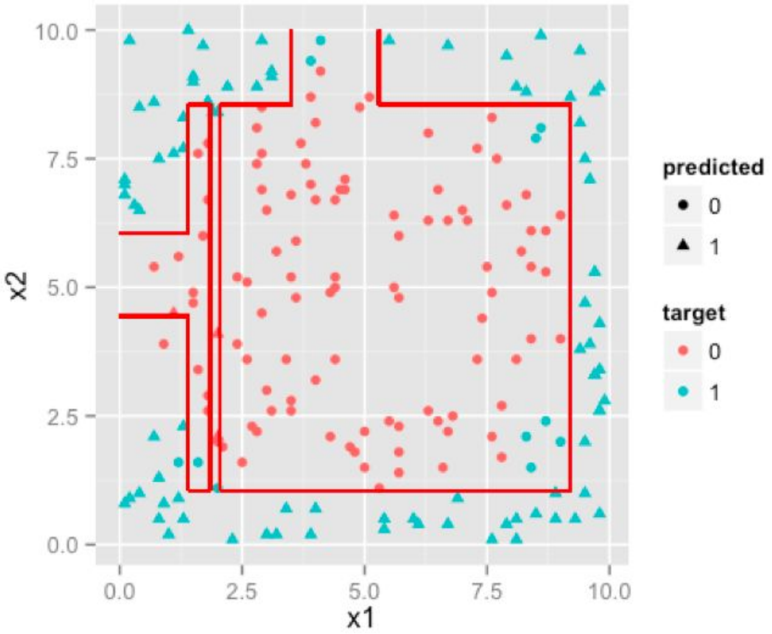


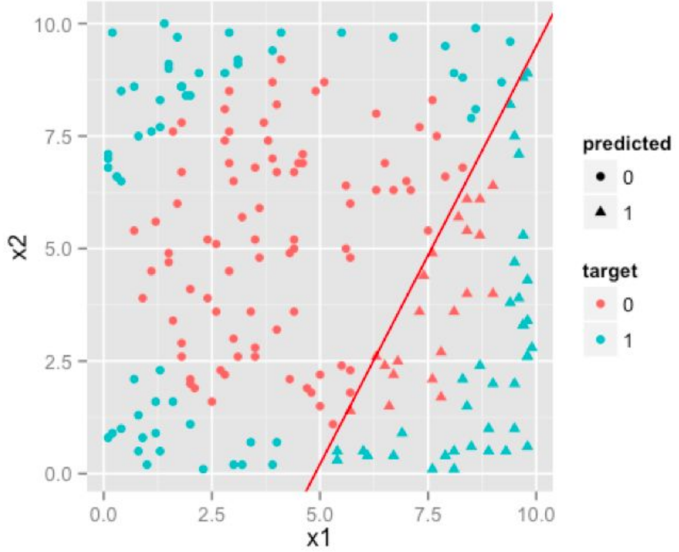
- **Logistic Regression:** assumes we can separate our data points by a single hyperplane.

- **Decision Trees:** Allow for non-linear boundaries, but make the additional assumption <u>that the decisions must be parallel to the axes.</u>

- **Answer:** It depends on your data! This will also be clearer when we cover regression next.

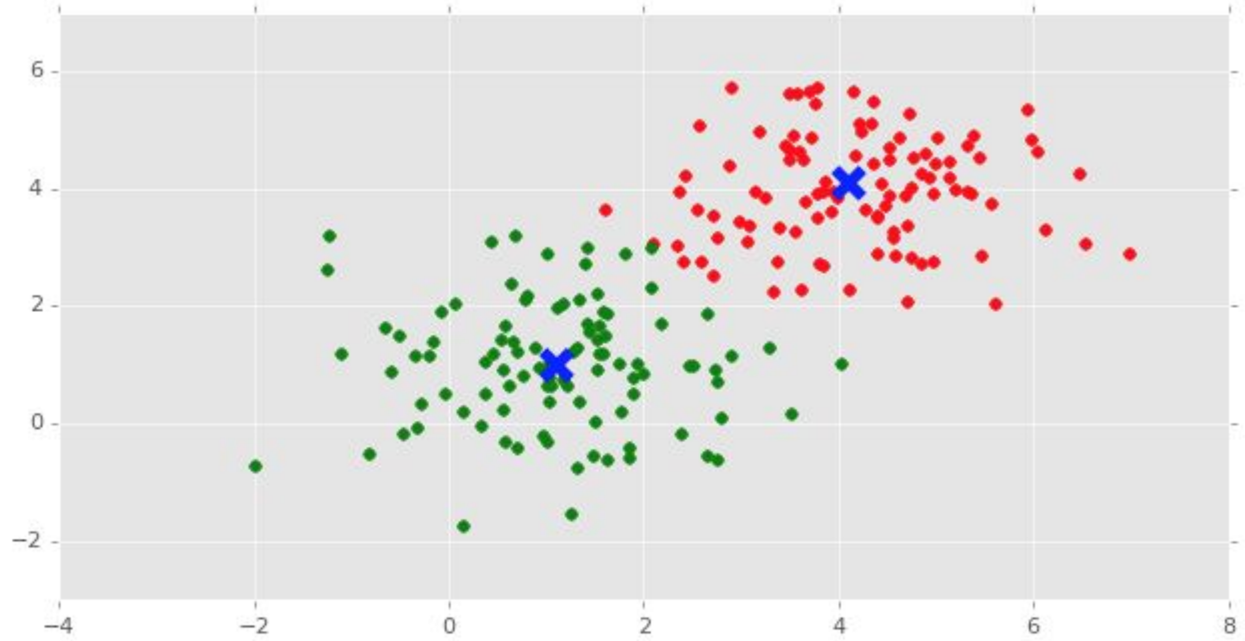
Decision Tree vs Linear



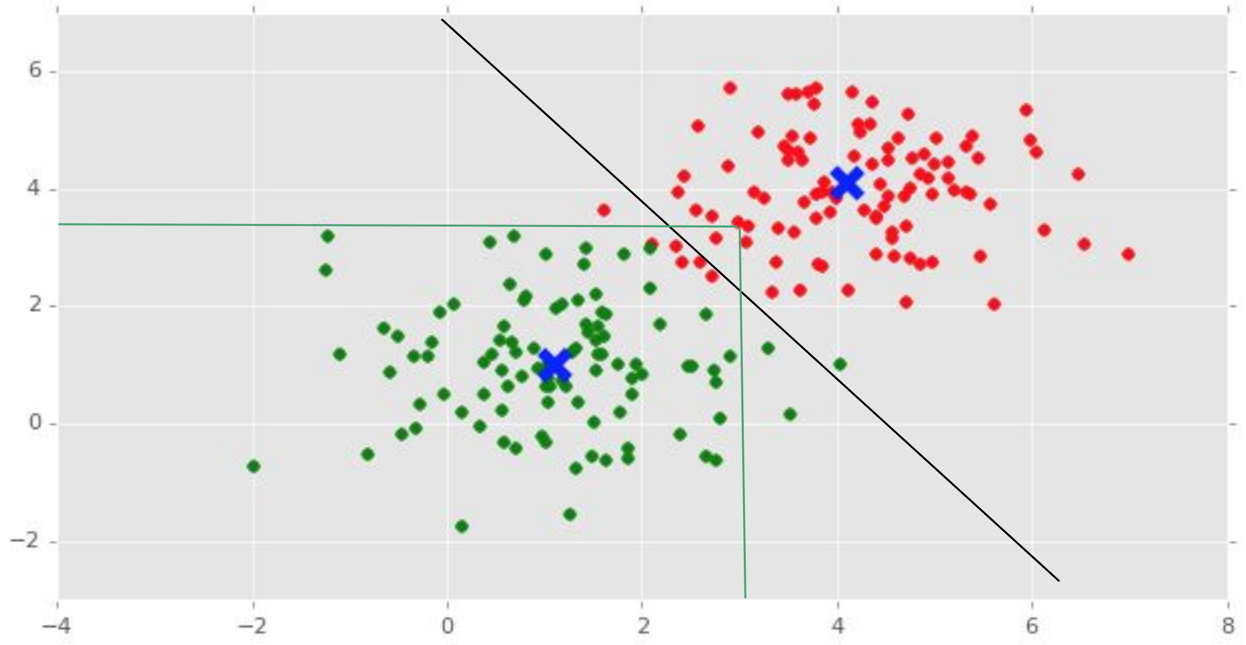


Sometimes we need a non-linear decision boundary. But the decision tree to the left seems inefficient no?

Which model would be better here?



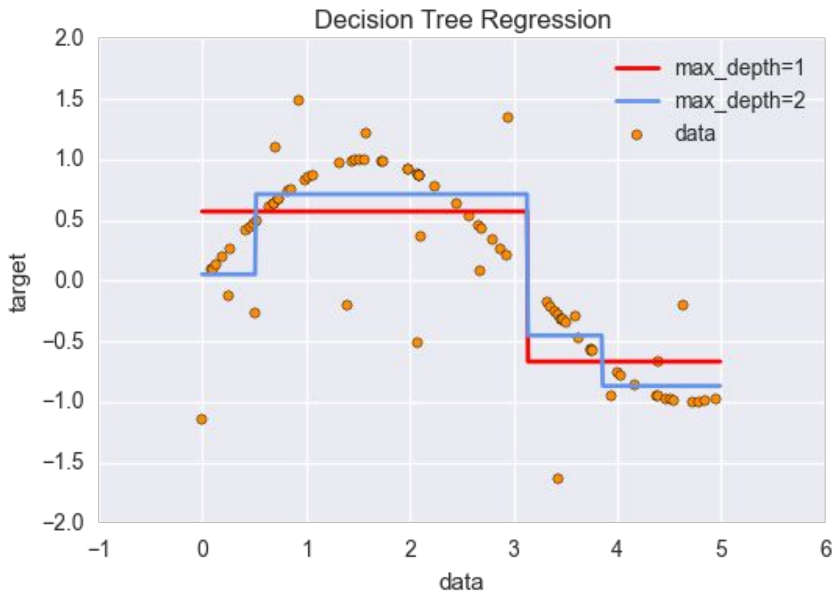
Which model would be better here?



Probably a linear model would work best here, but both could work.

Decision Tree Regression

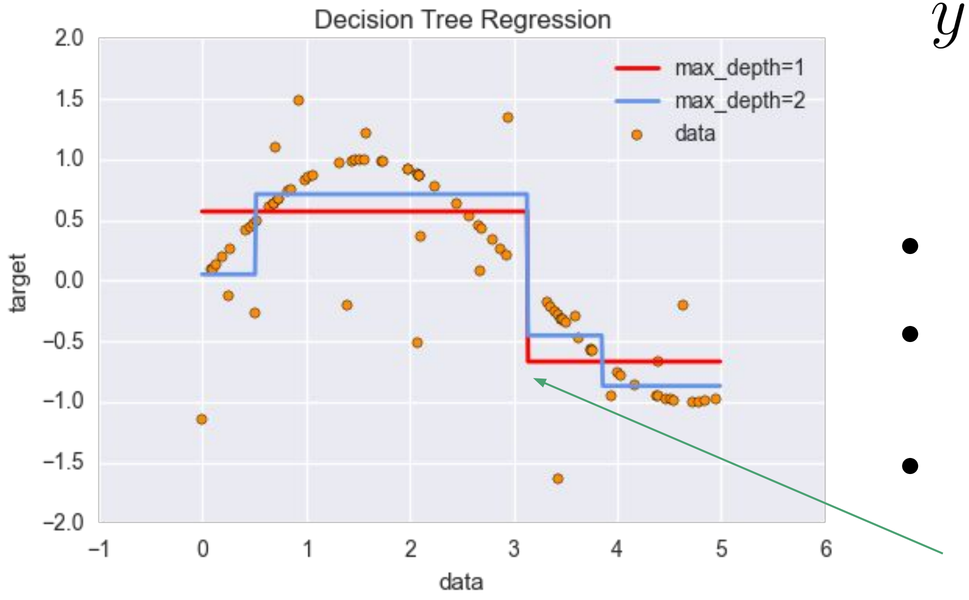
First: A 1d example



$$y = \sin(x) + \text{noise}$$

- Decision trees make a recursive series of decisions which lead to an outcome, real valued or labeled (for regression, real valued)
- The goal is to make splitting decisions on the data to minimize a norm, in particular, the L2 distance to the mean on that subset of the data, also known as the **variance.**

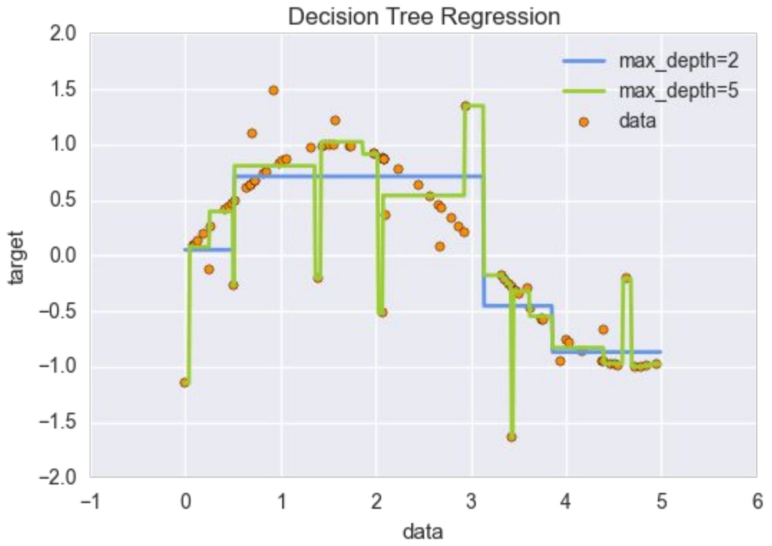
First: A 1d example



$$y = \sin(x) + \text{noise}$$

- For a depth of **zero**, one chooses the **mean**.
- How does one choose the splitting point when the depth is larger than zero?
- For a depth of one, the algorithm searches through all values between (-1,6) (in this example), and chooses the split which **minimizes the variance on the two segments which it creates.**

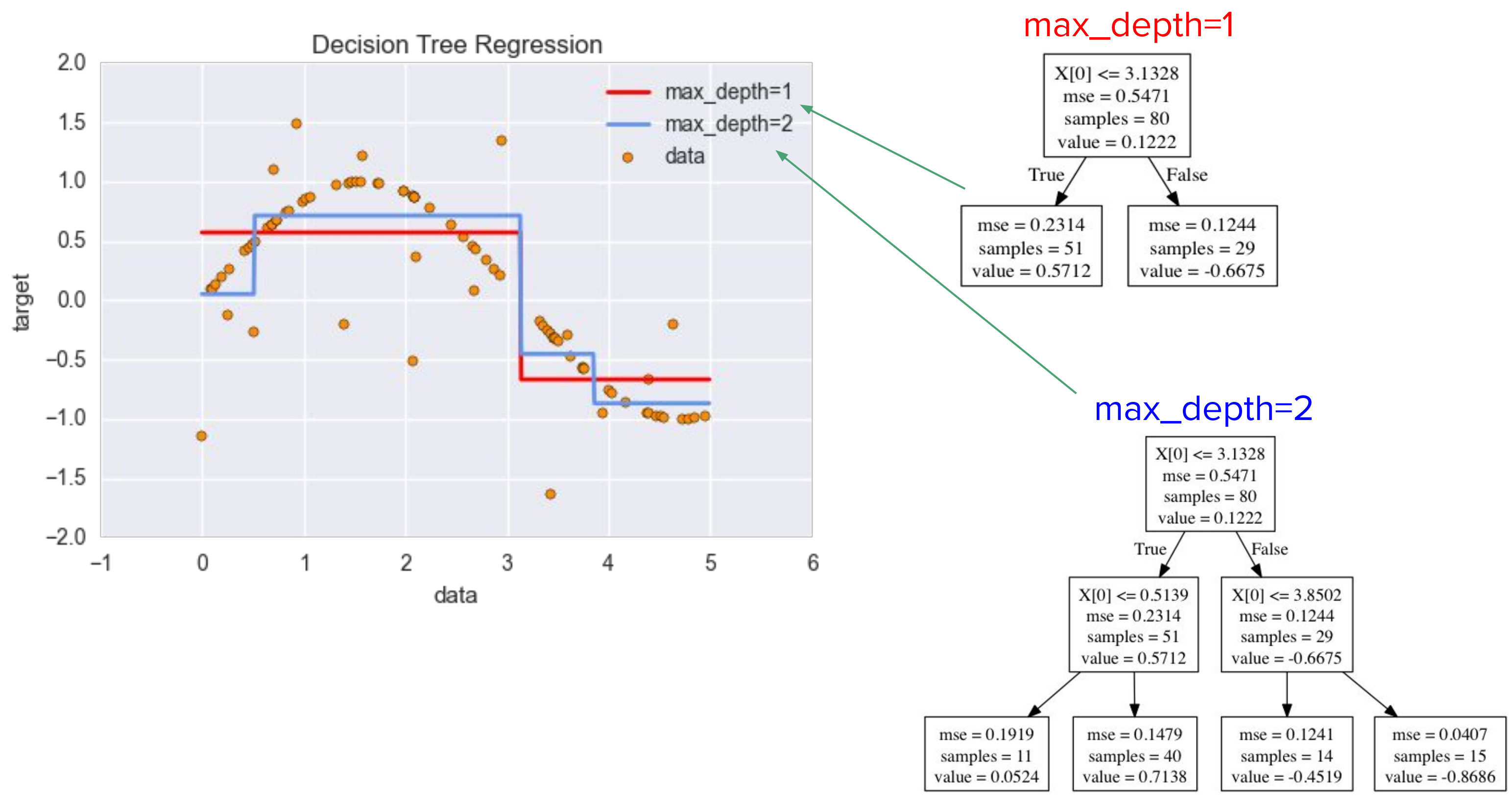
Increased depth can be bad



$$y = \sin(x) + \text{noise}$$

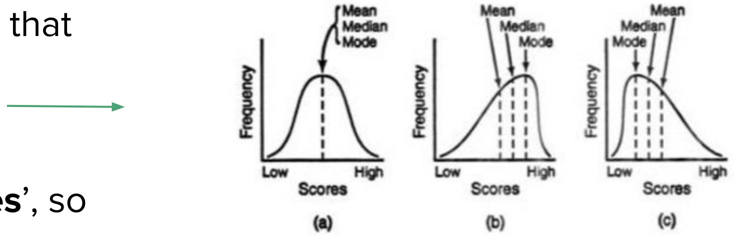
- In general, if one adds more depth to the tree, there is a risk of overfitting. Look at the unwanted variance we have picked up in this example.
- We will learn next lecture how to choose the perfect depth number!

How do the decisions work?



What are the advantages of regr. decision trees?

- Very easy to interpret.
- Makes **no a-priori assumptions about the structural form of the data** (as linear models do). For instance, works well with non-linear data.
- More “robust” than linear models, meaning **less sensitive to outliers**. This is for the same reason that the median is less sensitive to outliers than the median.
- Well defined notion of ‘**most significant variables**’, so good for data exploration (will explain).
- Handles categorical and real-valued variables.

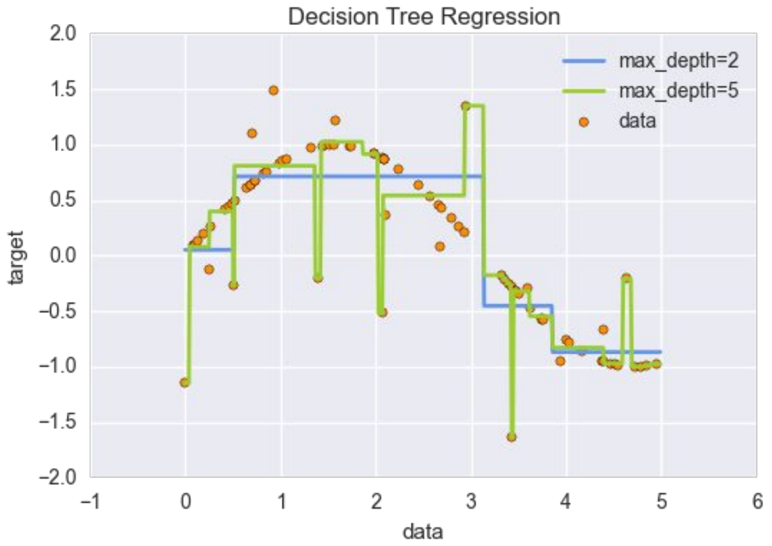


What are the disadvantages?

- Doesn’t handle continuous data as well sometimes - hard time with ‘new data’.
- Assumes rectangular decision boundaries which are parallel to the axes.
- Can over fit quite easily.
- Tends to have a bias towards variables with a wide range of values.

Regularization for Decision Trees

Increased depth can be bad

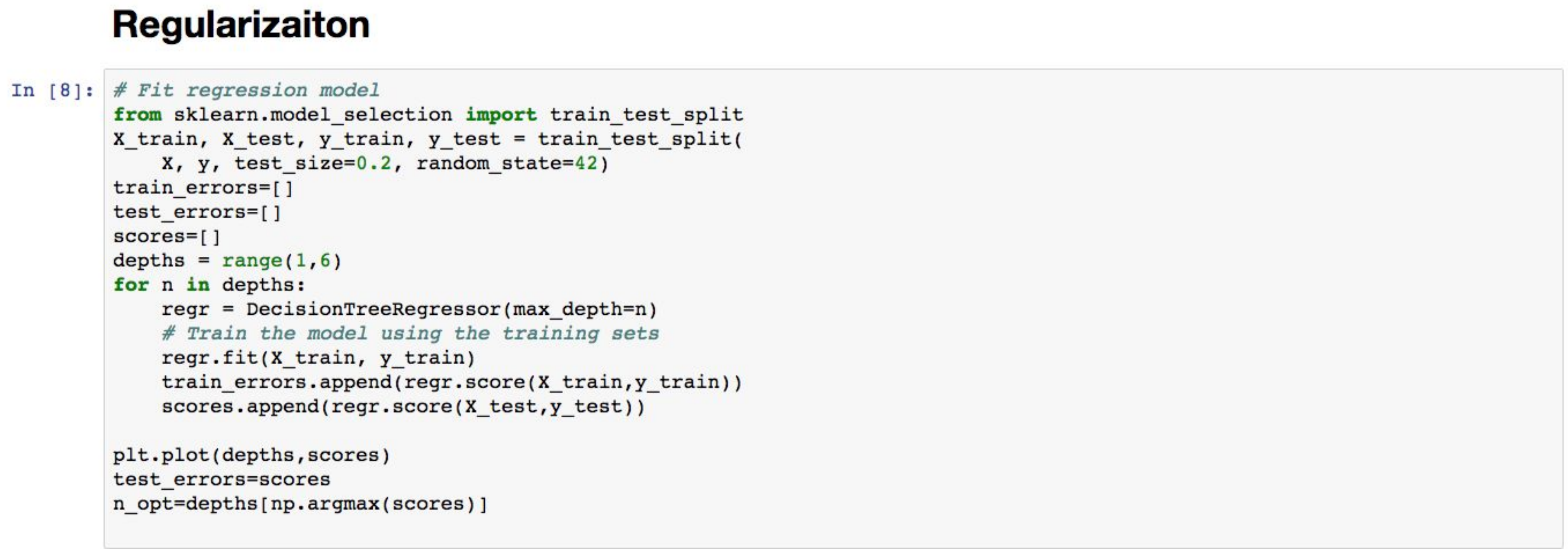


$$y = \sin(x) + \text{noise}$$

- In general, if one adds more depth to the tree, there is a risk of overfitting. Look at the unwanted variance we have picked up in this example.
- We will learn next lecture how to choose the perfect depth number!

Regularization is done by minimizing the depth of the tree.

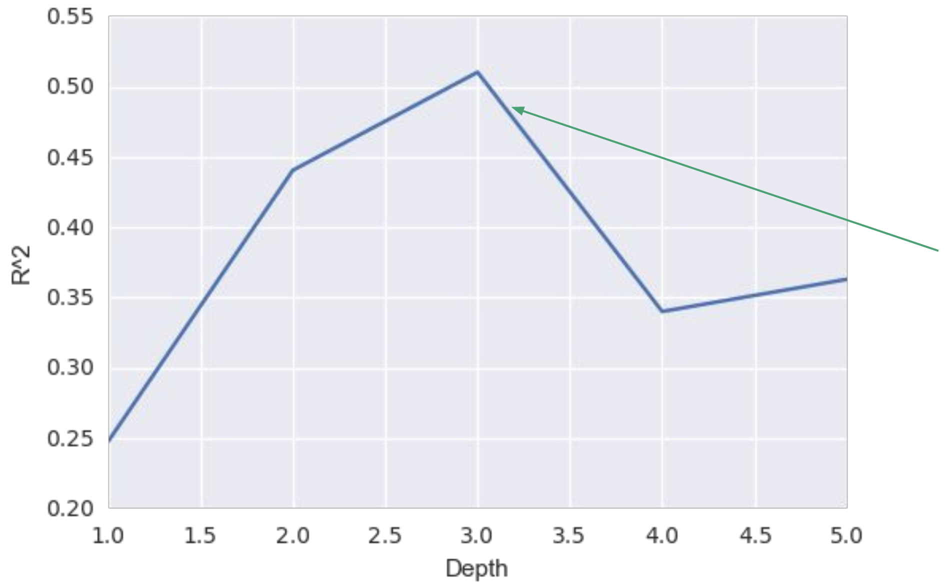
Regularizing a Decision Tree in Python



Regularizaiton

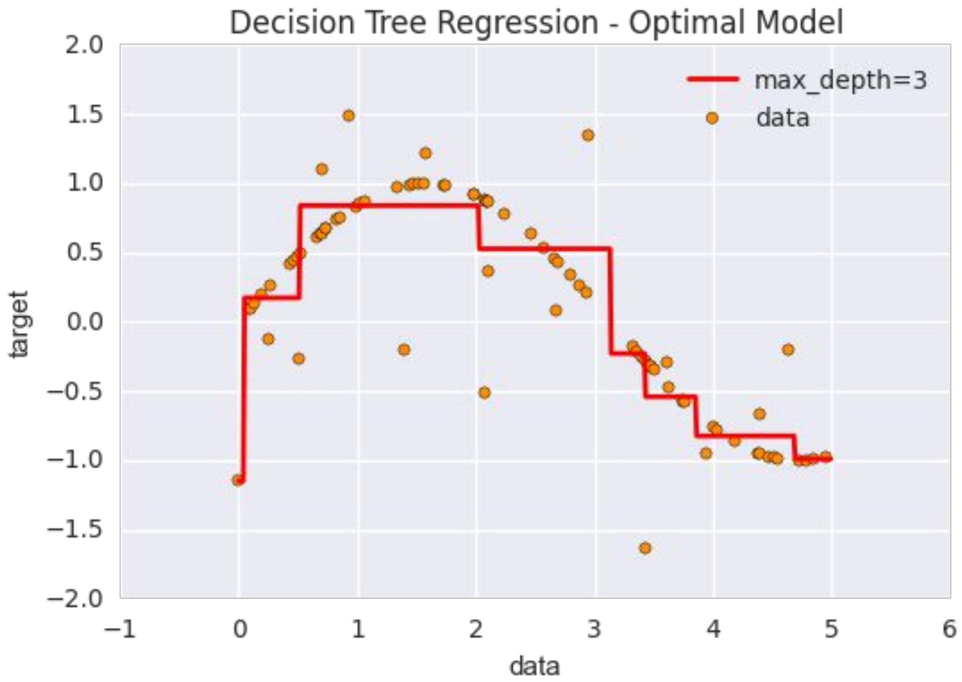
In [8]:
```
# Fit regression model
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
train_errors=[]
test_errors=[]
scores=[]
depths = range(1,6)
for n in depths:
    regr = DecisionTreeRegressor(max_depth=n)
    # Train the model using the training sets
    regr.fit(X_train, y_train)
    train_errors.append(regr.score(X_train,y_train))
    scores.append(regr.score(X_test,y_test))

plt.plot(depths,scores)
test_errors=scores
n_opt=depths[np.argmax(scores)]
```

Finding the best depth



Here the best depth is seen to be 3.

The optimal decision tree

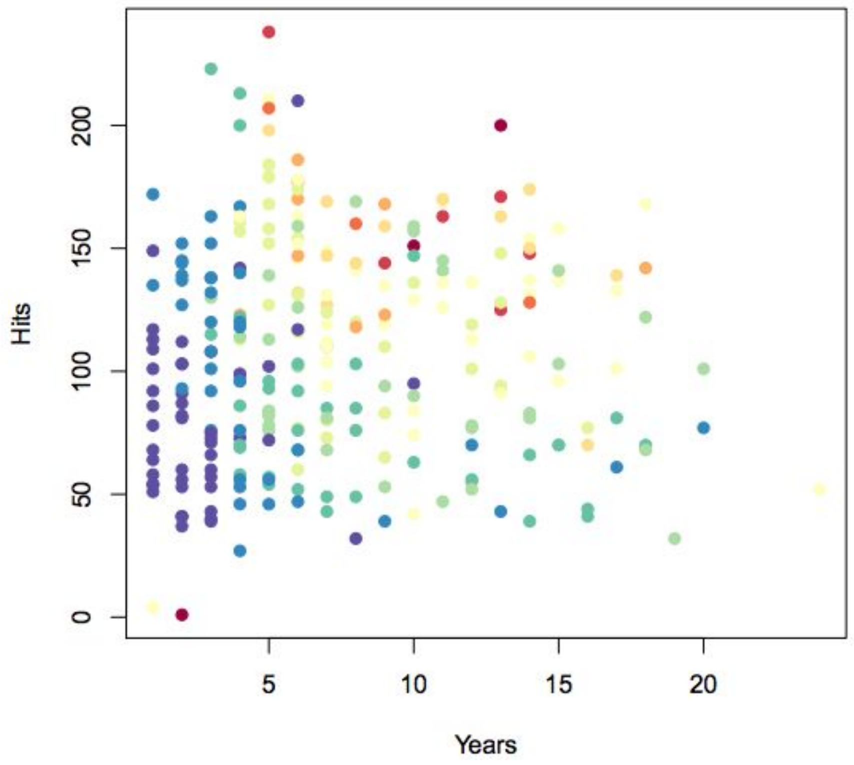


- Notice that this is consistent with our intuition of what would be the ‘right’ fit.

- The same procedure we did for linear models was applied here, but the regularization strength parameter replaced with depth.

More than one variable

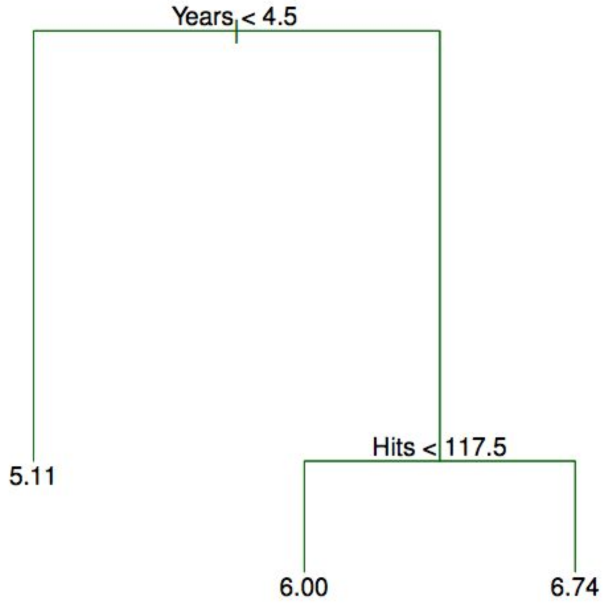
Example 1: Predicting Baseball Salaries

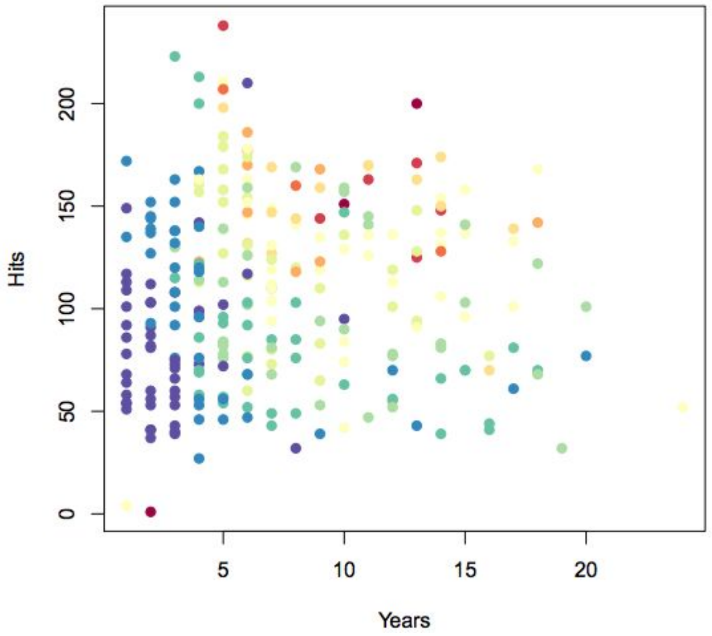


- When more than one variable is in the model, decision 'boundaries' are made.

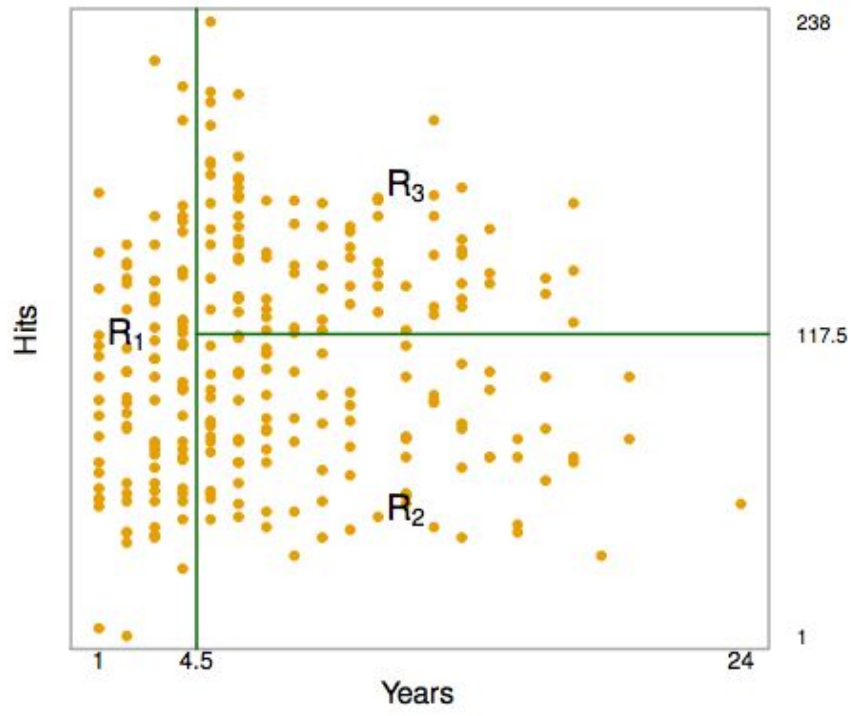
- To the left, blue means a lower salary, and red means a higher salary.

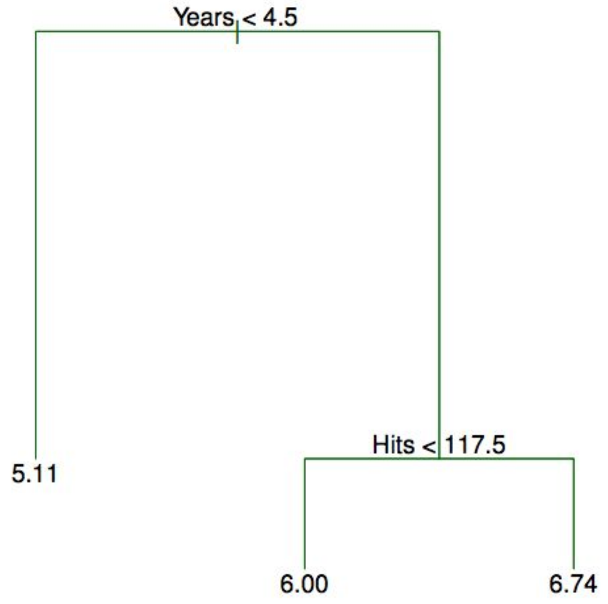
Depth 2 decision tree





Depth 2 decision tree



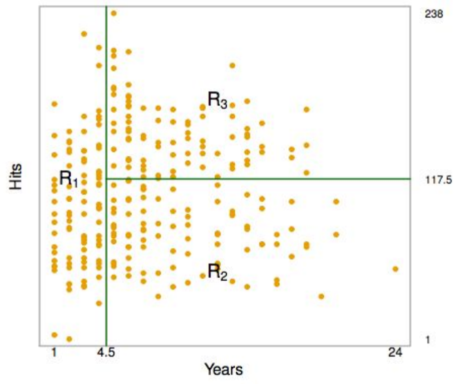


What are we optimizing for?

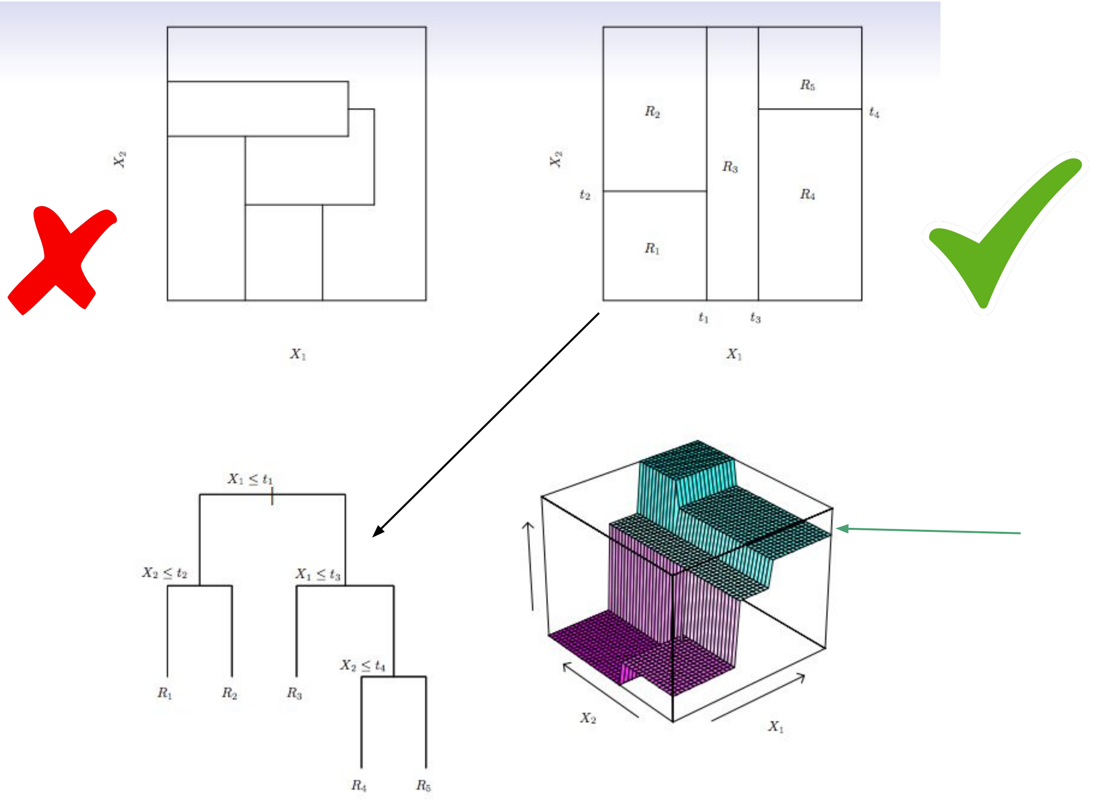
- We divide the predictor space — that is, the set of possible values for X1, X2, . . , Xp — into J distinct and non-overlapping regions, R1, R2, . . . , RJ.

- For every observation that falls into the region Rj , we make the same prediction, which is simply the mean of the response values for the training observations in Rj . We then want to minimize:

$$\min \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2$$



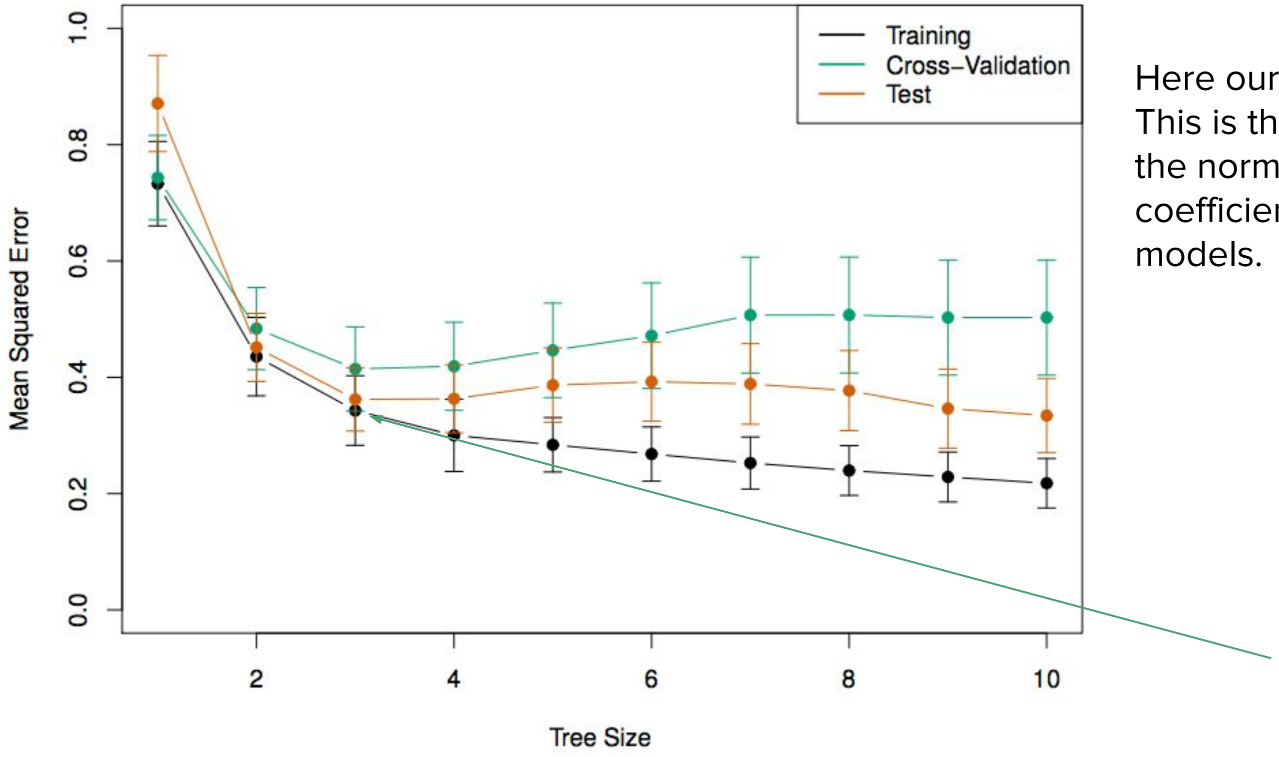
How do we make predictions?



$$\min \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2$$

Choose the value which is the mean over the values in that region.

How to optimize for depth? Regularization



Here our parameter is tree depth. This is the non-parametric analogue of the normed penalties on the coefficients when we studied linear models.

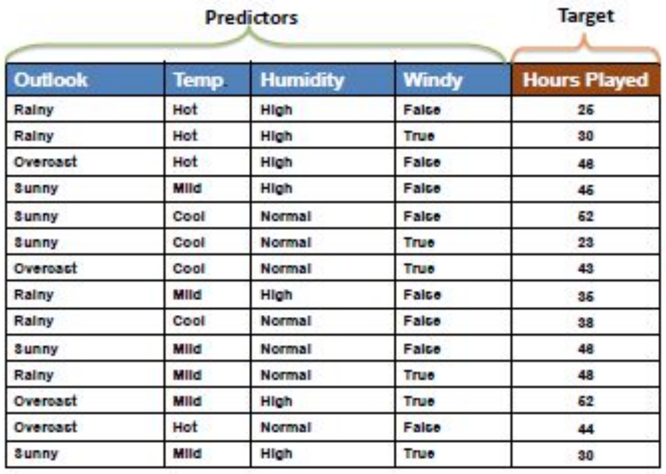
Sweet spot!

Announcements

- Very special lecture next week by Harold Li (data scientist at Lyft) on experimental design and SQL. https://www.linkedin.com/in/harold-li-ba21ba44
- Today: HW0 grades posted, HW1 soon to follow.
- After next week's special lecture, we will assign the final project, which will be due the final day of class.

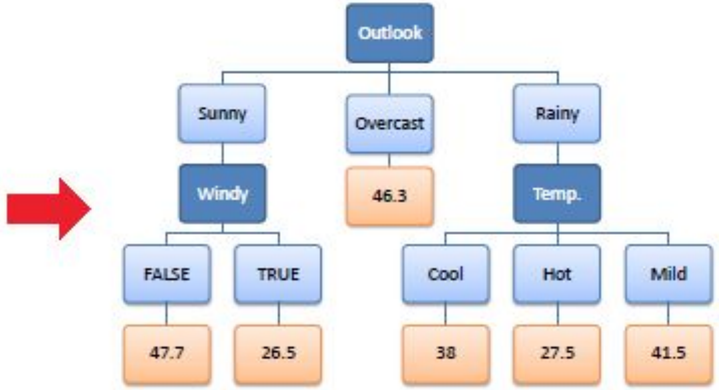
The Algorithm

Hours Played on Golf Course



Predictors | Target

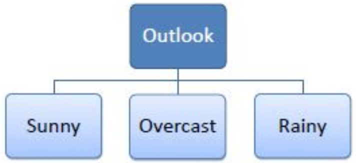
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

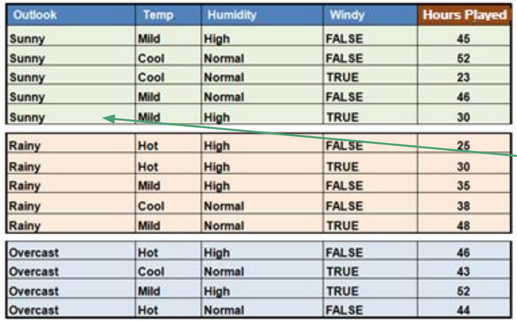


Machine Learning Objective: Let's try to predict the number of hours played on a golf course in a single day by all of the golfers.

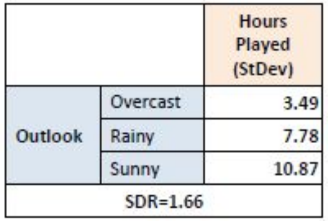
Example taken from: http://chem-eng.utoronto.ca/~datamining/dmc/decision_tree_reg.htm

How do we choose the root node?

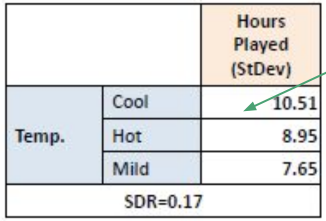




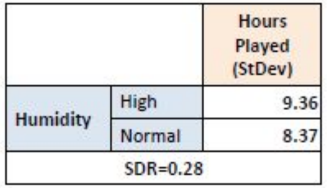
Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44



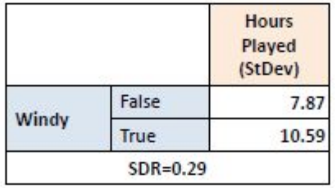
		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		



		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		



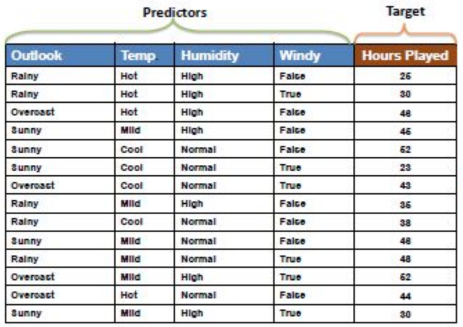
		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		



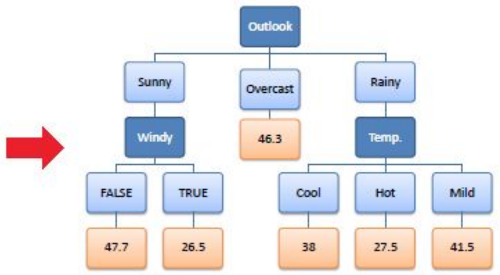
		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

- Here we've tried splitting first by the variable "Outlook".
- From here, we can compute the conditional standard deviations in the three subsets created.
- Our goal is to find the variable that, when split, reduces the stdev the most.
- Original stdev of "Hours Played" is 9.33.
- We can try this for every variable, and then choose the one which reduces the stdev the most.
- Subsequent decisions are made by recursively iterating this procedure.

More precise description



Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



For zero depth we minimize:

$$\mathrm{Var}(y) := \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

Solution is: $\hat{y} = \bar{y}$

$$\mathrm{Var}(Y|X = x_j) = \sum_{i=1}^{N}(y_i - \bar{y}_j)^2 p(y_i|x_j)$$

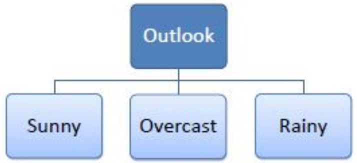
$$= \frac{1}{N}\sum_{i,x_i=x_j}(y_i - \bar{y}_j)^2 |X = x_j|$$

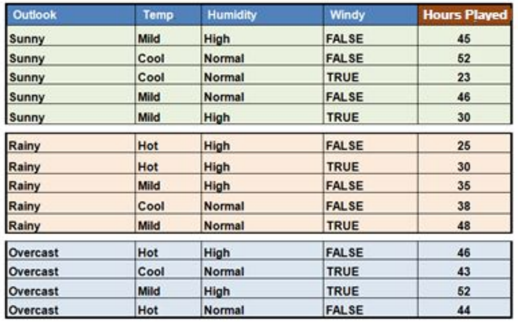
For depth one we minimize the conditional variance for each variable:

$$\mathrm{VarRed}(Y, X) = \mathrm{Var}(Y) - \sum_j \mathrm{Var}(Y|X = x_j)$$

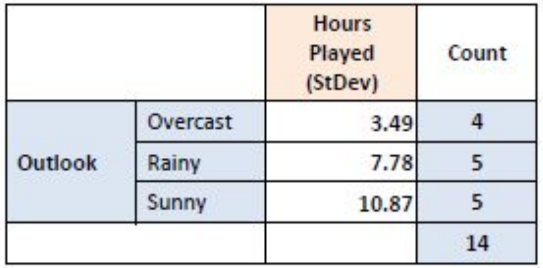
Variance Reduction

Let’s try Outlook first





Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44



		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

$$\mathrm{Var}(y) := \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 = 9.32^2$$

$$\sum_j \mathrm{Var}(Y|\mathrm{Outlook}) = \frac{4}{14}(3.49)^2 + \frac{5}{14}(7.78)^2 + \frac{5}{14}(10.87)^2$$

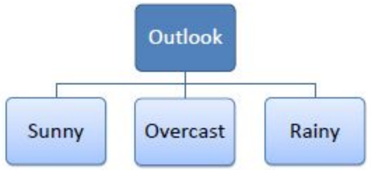
$$\mathrm{Var}(Y|X = x_j) = \sum_{i=1}^{N}(y_i - \bar{y}_j)^2 p(y_i|x_j)$$

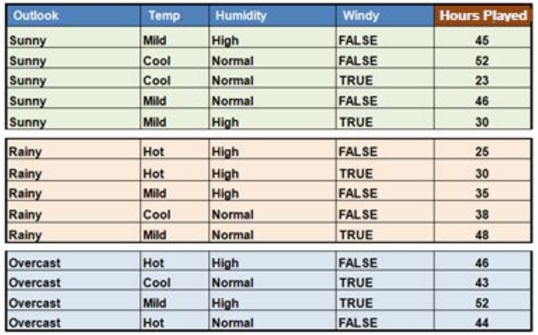
$$= \frac{1}{N}\sum_{i,x_i=x_j}(y_i - \bar{y}_j)^2 |X = x_j|$$

$$\mathrm{VarRed}(Y, X) = \mathrm{Var}(Y) - \sum_j \mathrm{Var}(Y|X = x_j)$$

- Therefore there is a variance reduction of **19.56 for the outlook variable.**

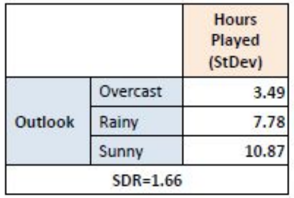
Now we've split based on Outlook



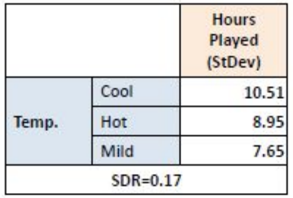


Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

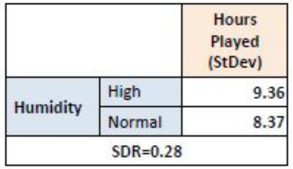
- These groups now have less variance in the data. And Outlook reduced variance the most. Now we continue recursively.



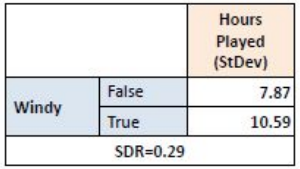
		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		



		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		



		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		



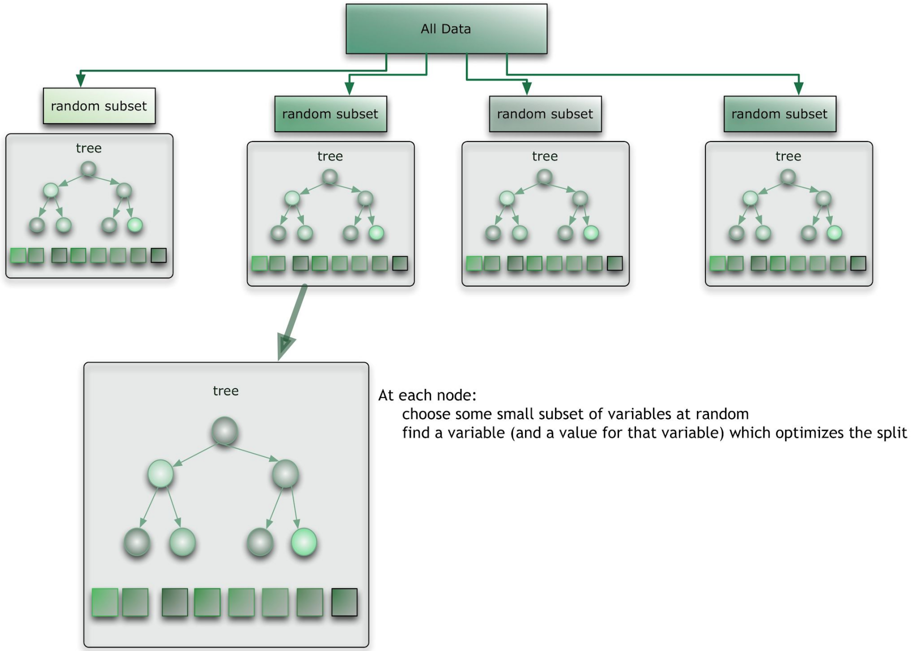
		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

So mean squared error replaces information entropy from classification trees - that's really the only difference.

Random Forests

Random Forest Introduction

-


All Data
random subset
random subset
random subset
random subset
tree
tree
tree
tree
tree
At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

A random forest generalizes decision trees in the following way:

1. It <u>splits the data into random subsets.</u>
2. In the random subsets, <u>it chooses a random set of features to build a decision tree.</u>
3. The <u>mean of the values from each decision tree</u> is taken as the value.
4. For a <u>classification model</u>, the <u>mode</u> can be taken, or the <u>mean.</u>

Advantages/Disadvantages to Random Forests

- The main advantage of Random Forests is that they prevent overfitting by taking random subsets of the features to use to build the decision trees. This makes it unlikely that you will pick up too much variance.

- They’re very robust and easy to use - no parameter optimization is really needed.

- The major drawback is that they’re slow - they’re not practical for any kind of real time decision making.

- They’re also not good if the model needs to be interpretable.

Some more

- Random Forests **aren't always good at generalizing to cases with completely new data**. For example, if I tell you that 1 chocolate costs $1, 2 chocolates cost $2, and 3 chocolates cost $3, how much do 10 chocolates cost? A linear regression can easily figure this out, while a Random Forest has to guess more.

- If a variable is a categorical variable with multiple levels, random forests are **biased towards the variable having multiple levels**

How to fine tune a random forest?

Two parameters are important in the random forest algorithm:

1. Number of trees used in the forest (the more the better).
2. Number of random variables used in each tree (generally optimal is around the square root of the number of features total).
3. The depth of each tree.

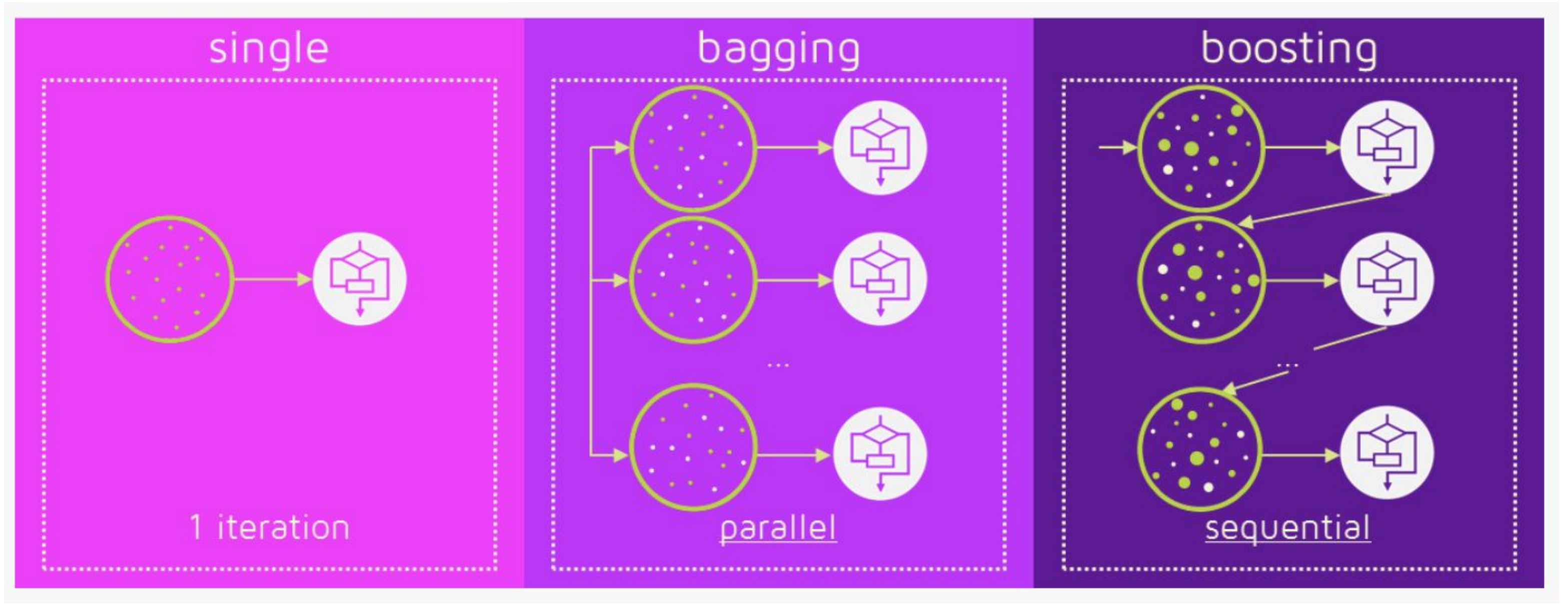
Variable Importance for Random Forests

- For RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all trees.

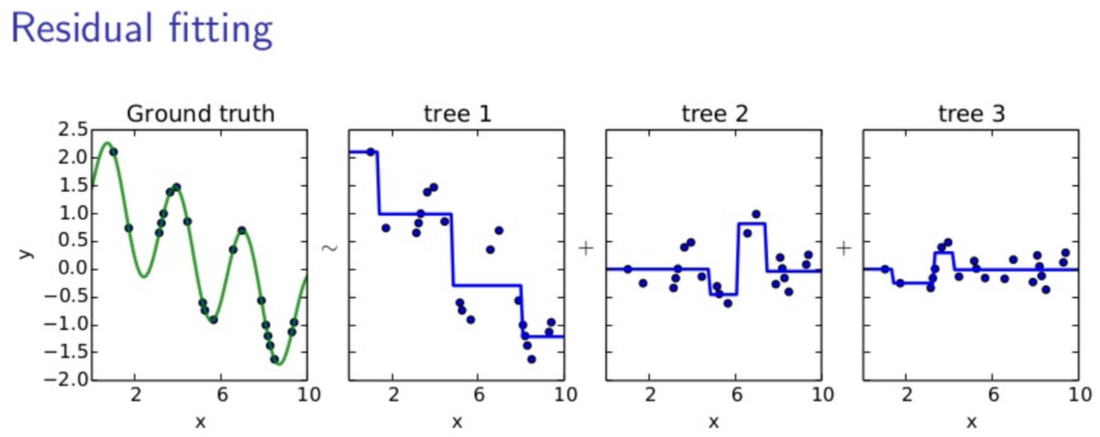
- A large value indicates an important predictor.

Gradient Boosted Trees

Boosting vs Bagging



Fitting the residuals for regression

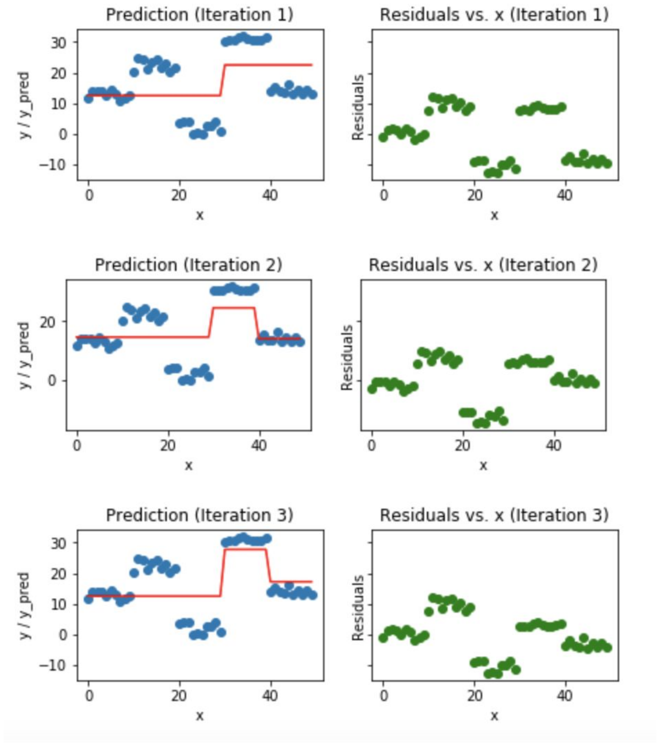


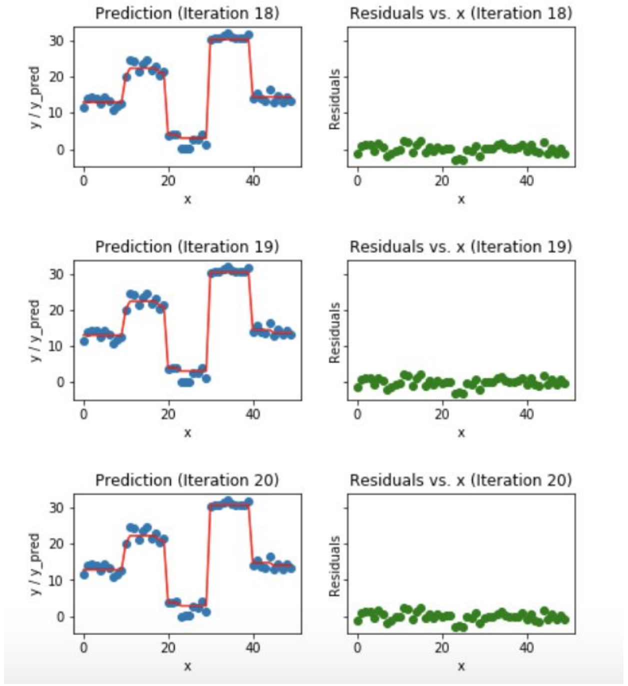
1. **“Tree 1”** is a best-fit-line of the data.

2. **“Tree 2”** is a curve that plots the errors from the graph in “Tree 1”. These errors are based on how “Tree 1” misrepresented the original plot (“Ground truth” in this case).

3. **“Tree 3”** is a combination of ‘Tree 1’ and ‘Tree 2’.

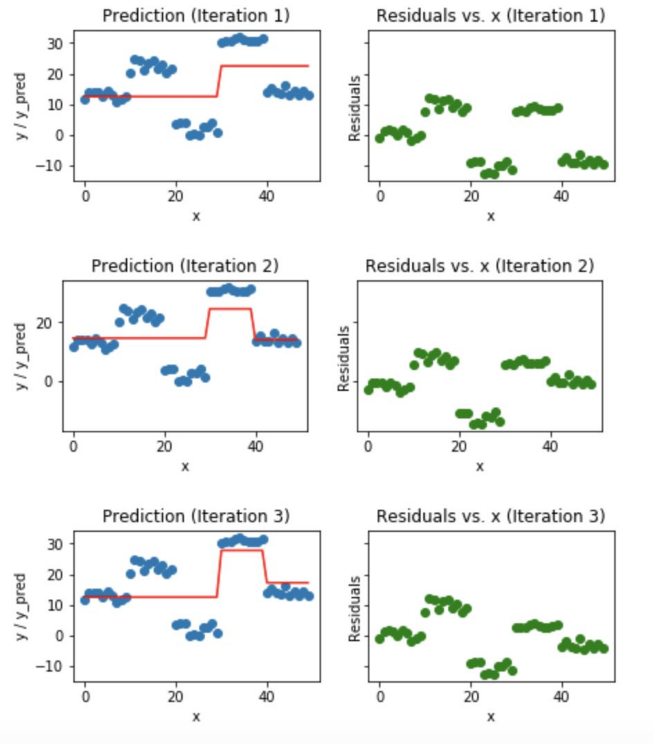
Fitting the residuals for regression





A way to correct errors from residuals

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$
$$\cdots$$
$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$



How to find the next f?

$$\begin{aligned}\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant\end{aligned}$$

We choose the next f to minimize the total loss given the previous prediction.

How to find the next f?

$$\hat{y}^t = \hat{y}^{t-1} + \text{argmin}_h \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i^{t-1} + h(x_i)).$$

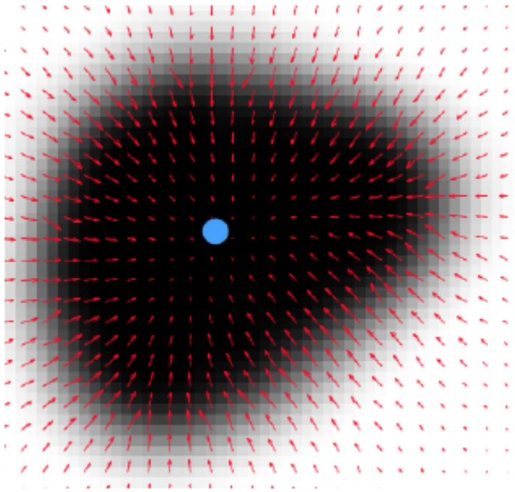
The above problem is not well posed though! Why?

How to find the next f?

$$\hat{y}^t = \hat{y}^{t-1} + \text{argmin}_h \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i^{t-1} + h(x_i)).$$

We can't just use all h - so let's follow the gradient flow!

$$\hat{y}_j^t = \hat{y}_j^{t-1} - \nu\alpha_t \sum_{i=1}^{N} \nabla\mathcal{L}_{\hat{y}_i^{t-1}}(y_i, \hat{y}_i^{t-1}),$$



Is that it?

So why can't we just solve the steepest descent problem? The main issue is explaiend well in *The Elements of Statistical Learning* by Hastie et al.

"If minimizing the loss of the training data were the only goal, steepest descent would be the prefered strategy. The gradient is trivial to calculate for any differentiable loss function (y,f(x)L(y,f(x) *)…Unfortunately the gradient is only defined at training points, whereas our goal is to generalize* y ˆ ^{t-1} and y^t *to new data not represented in the training set."*

Regress on the gradients

$$r_{ij} = \nabla_{\hat{y}_i^{t-1}} l(y_i, \hat{y}_i^{t-1})$$

- If we only needed a model on training examples, we wouldn’t need to do this. But we have to generalize to new data, so we build very shallow decision trees for each correction term.
-
- Why decision trees and not linear models? (think about this for a second).

The Algorithm

Step 1: Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

Step 2: For all m**: Compute so-called pseudo-residuals:**

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

In other words, fit $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.

The Algorithm Continued

Step 4: Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

Step 5: Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

Output $F_M(x)$..

Different Boosting Methods

Both are boosting algorithms which means that they convert a set of weak learners into a single strong learner. They both initialize a strong learner (usually a decision tree) and iteratively create a weak learner that is added to the strong learner. They differ on **how they create the weak learners** during the iterative process.

Adaboost: Weights the “wrong” examples more heavily in the subsequent iterations.

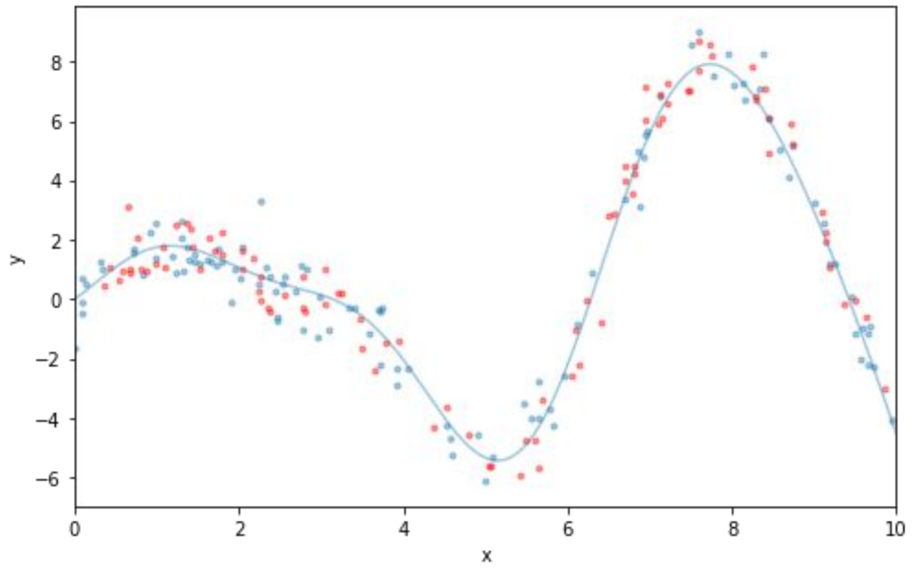
Gradient Boost (original): Computes residuals and optimizes for those alone.

How does it compare?

1. RF uses decision trees, which are very prone to overfitting. In order to achieve higher accuracy, RF decides to create a large number of them based on bagging. The basic idea is to resample the data over and over and for each sample train a new classifier. Different classifiers overfit the data in a different way, and through voting those differences are averaged out.

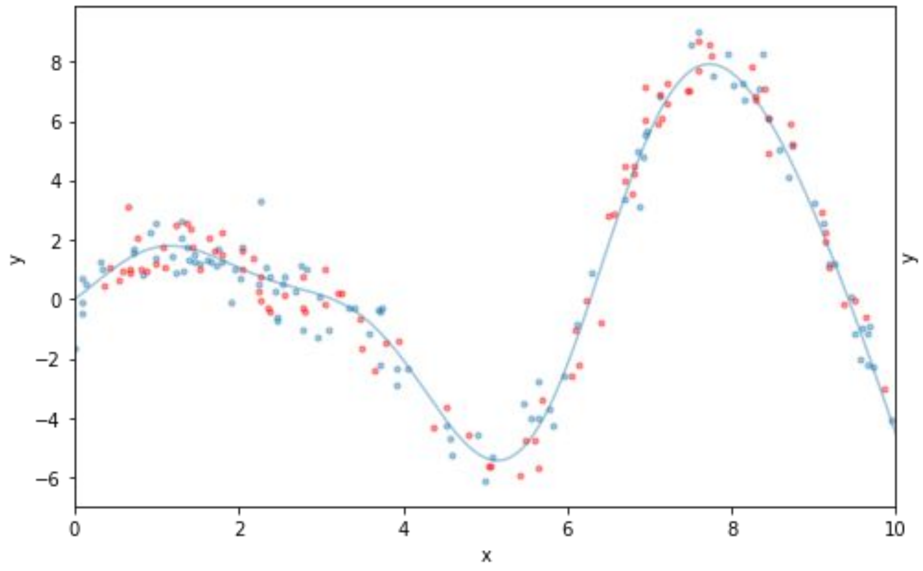
2. GBM is a boosting method, which builds on weak classifiers. The idea is to add a classifier at a time, so that the next classifier is trained to improve the already trained ensemble. Notice that for RF each iteration the classifier is trained independently from the rest.

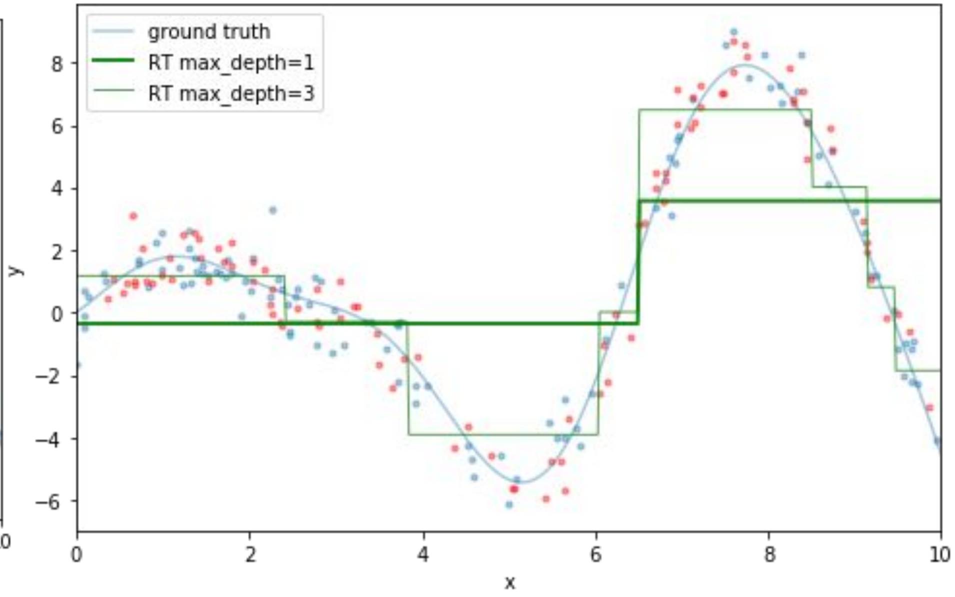
Decision Tree Regressor



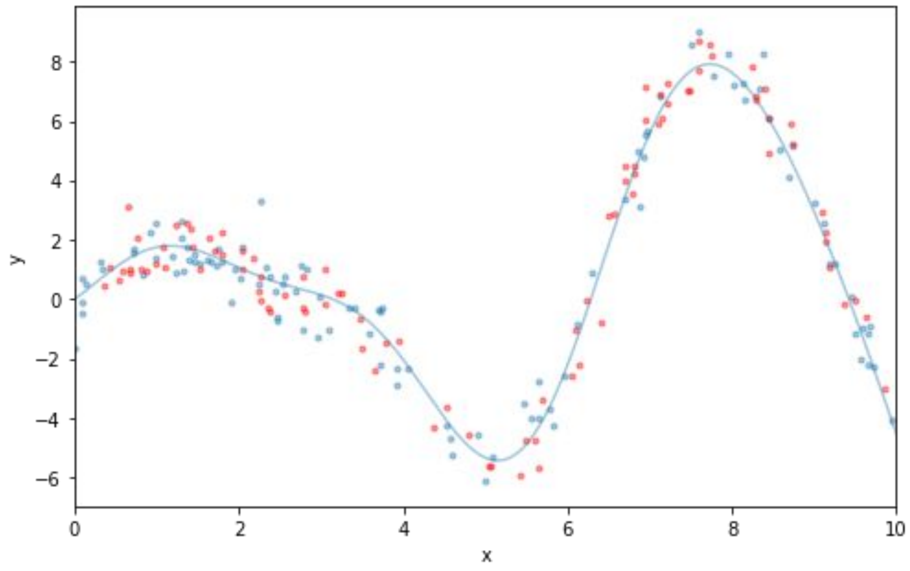
?

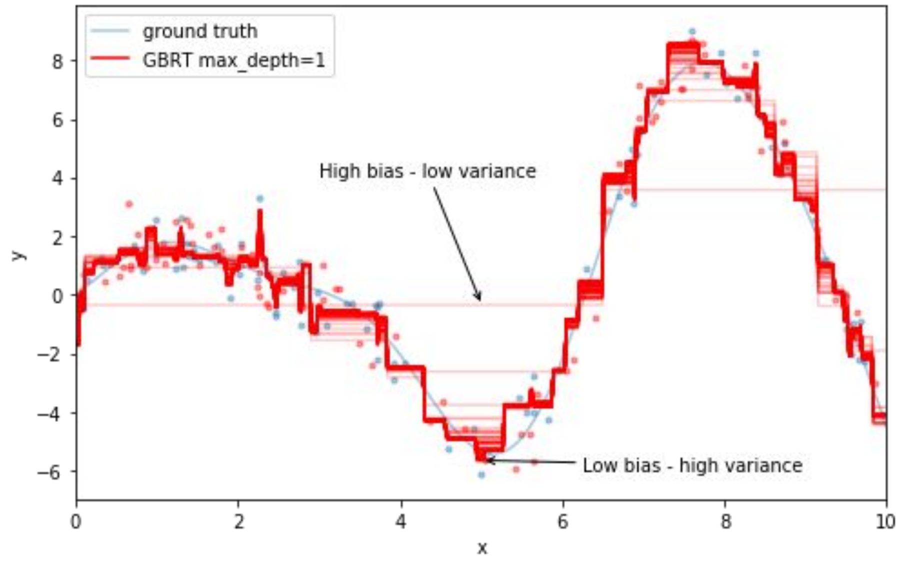
Decision Tree Regressor



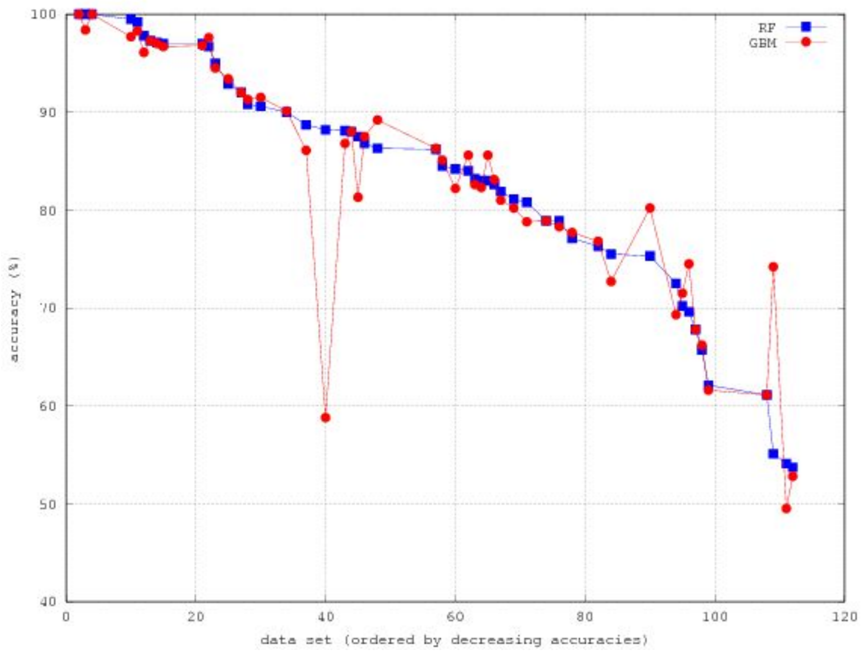


Gradient Boosted Regressor





Comparison on 100 datasets



- GBM offers slightly better results most of the time.
- GBM is much harder to tune however since you need to tune learning rate, decision tree depth, etc.

Summary

- Random Forests perform almost as well and don’t require fine tuning as much. You can also parallelize them easily (why?)

- GBT will usually perform better since you are approximating the function in a more fine grained way.

- If simplicity and speed are what is needed - Random Forest. If you’re looking to squeeze juice out of your data and improve performance, then GBT might be the way to go.

What is used more commonly?

- The secret truth is that almost all models that aren’t based on image or sound processing use some combination of random forest and gradient boosting.

- Mastering these two models means you’ve mastered a large percentage of the tools used in practical data science problems.

- Tree based methods are simpler to use and handle a wider variety of data than linear models do.