Introduction to Regression

Lecture 2

Outline

- Lecture 2: Introduction to Regression
  - Definition, and comparison of Lp norms.
  - How do we find the solution? The special case of p=2.
  - When do unique solutions exist?
  - The problem of dependent features.
- General Solution Finding Procedure - Gradient Descent
  - Gradient Descent, and Introduction to Convex Optimization.
  - Concrete Examples of Linear Regression and Gradient Descent in Python in an iPython Notebook.
- Introduction to Decision Trees
  - How decision trees are constructed - variance reduction.
  - Concrete Examples of Decision Trees in Python with an iPython Notebook.

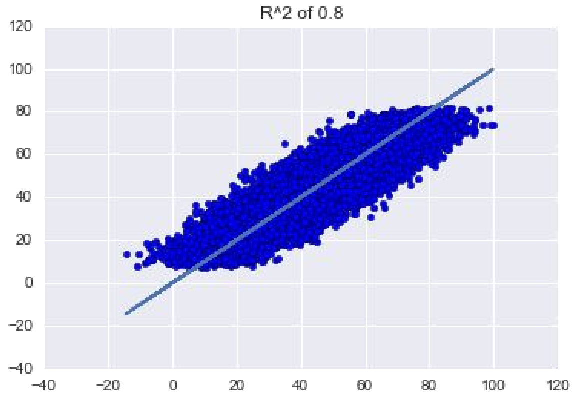
Introduction to Regression

Given a collection of points to learn from: (x_i, y_i)

Can we find a function $f : X \to Y$ minimizing the distance to the data.

$$\mathcal{L}(y, f) := \sum_{i=1}^{N} d(y_i, f(x_i))$$

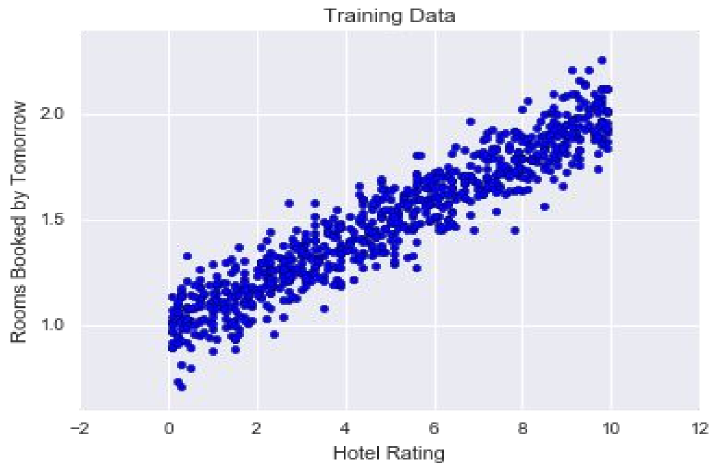
$d(x, y)$ - Is a metric (distance between two points)

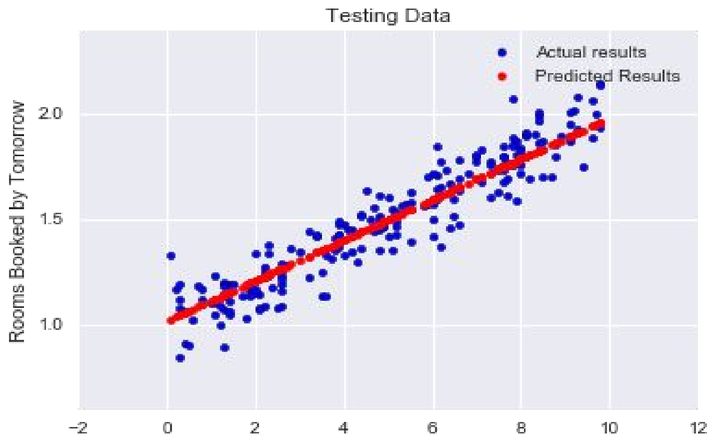


Testing/Training Split

We always learn from training data

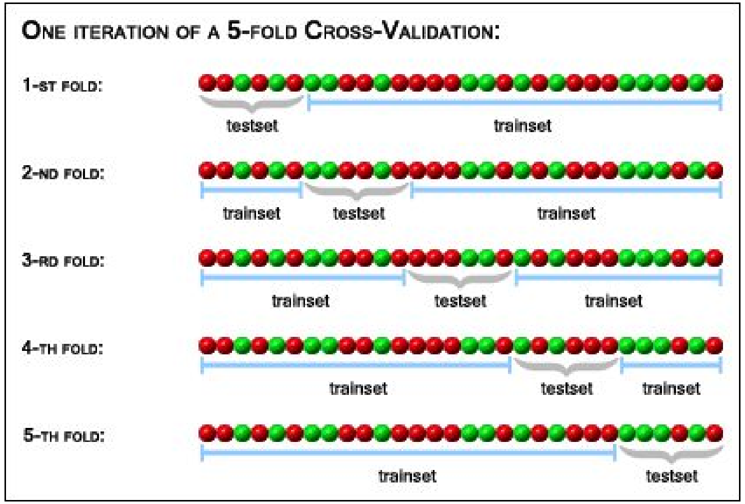
Splitting data into Testing and Training





- First, we randomly split our data into **testing and training subsets.**
- The most important thing in machine learning is **generalizability** - we must be able to **make predictions on unseen data** that works.
- If you make a model **complex enough**, you can **always get very good accuracy** on the **training data**, but it **won't generalize** (next lecture).

Cross Validation



- Generally instead of just taking say 80% of your data for training, 20% for testing (for example), we randomly split the data into several ‘folds’.

- In **k-fold cross-validation**, the original sample is randomly partitioned into **k equal sized subsamples**. Of the k subsamples, **a single subsample is retained as the validation data** for testing the model, and t**he remaining k − 1 subsamples are used as training data.**

Linear Regression

The special case of Lp

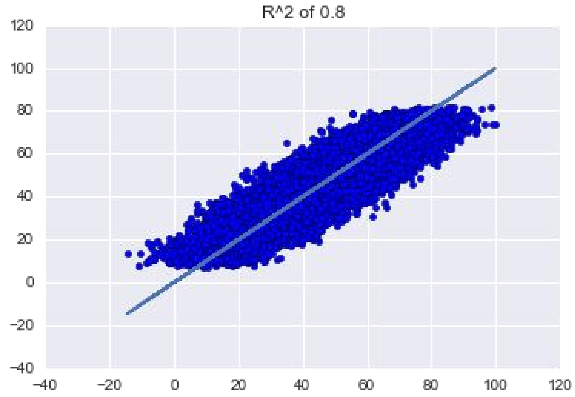
$$\mathcal{L}(y, \beta \cdot x) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^p$$

Most common case:

- p=2
  - Penalizes outliers much more.
  - Less sparse coefficients.
  - Has an **analytical solution.**
  - Unique solution when features are linearly independent.
  - Compatible with CLT (will see later!)
- p=1
  - More sparse coefficients (zero valued).
  - Helps eliminate collinear features.
  - Degenerate solutions.
  - Rarely used as a norm, but is used for penalizing coefficients

Linear assumption:

$$f(x_i) = \beta \cdot x_i$$



Question: Why would $p = \infty$ be a bad choice for machine learning?

Review of Convex Analysis (Calculus) - Part I

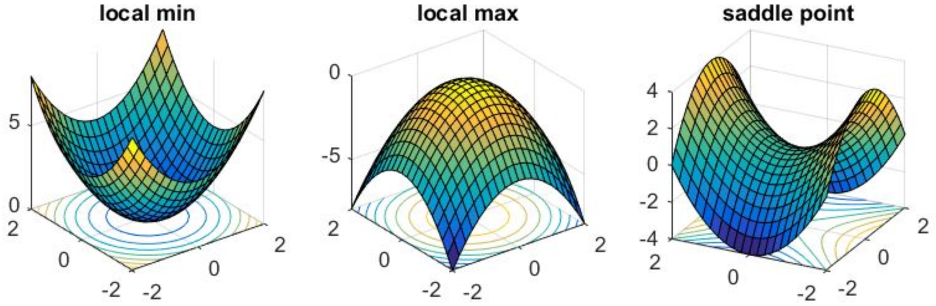
- Assume $\mathcal{L} : \mathbf{S} \rightarrow \mathbb{R}$ is continuously differentiable and convex, ie.

$$\mathcal{L}^{''}(\beta) \geq 0 \text{ for all } \beta \in \mathbf{S}$$

$$\exists \beta_0 \text{ such that } \mathcal{L}(\beta_0) \leq \mathcal{L}(\beta) \text{ for all } \beta \in S$$

- There f has a minimum value. Moreover, the minimum is unique when f is strictly convex, ie.

$$\mathcal{L}^{''}(\beta) \geq c > 0 \text{ for all } \beta \in \mathbf{S}$$



The Case of $p = 2$

Analytical Solutions and Stability Analysis

Analytical Solution to OLS when p=2

$$\mathcal{L}(\beta) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \beta \cdot x_i|^2$$

Claim: When p = 2 and the feature matrix $(X^T X)$ is invertible, the above is minimized uniquely by

$$\beta = (X^T X)^{-1} X^T y$$

The solution can make sense when $(X^T X)$ isn't invertible (in practice it often can be "almost not invertible", but solutions will be unstable and degenerate (more later)

Analytical Solution to OLS when p=2

Proof: $$\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N} |y_i - \beta \cdot x_i|^2$$

- Need linearly independent features for a unique inversion.

Differentiating, we have $$\frac{\partial \mathcal{L}}{\partial \beta_j} = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \beta \cdot x_i)x_j$$

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = 0 \Rightarrow \boxed{\frac{1}{N}\sum_{i=1}^{N}(\beta \cdot x_i)x_{ij} = \frac{1}{N}\sum_{i=1}^{N} y_i x_{ij}.} \Rightarrow \quad X^T X\beta = X^T y.$$

$$\Rightarrow \qquad \beta = (X^T X)^{-1} X^T y$$

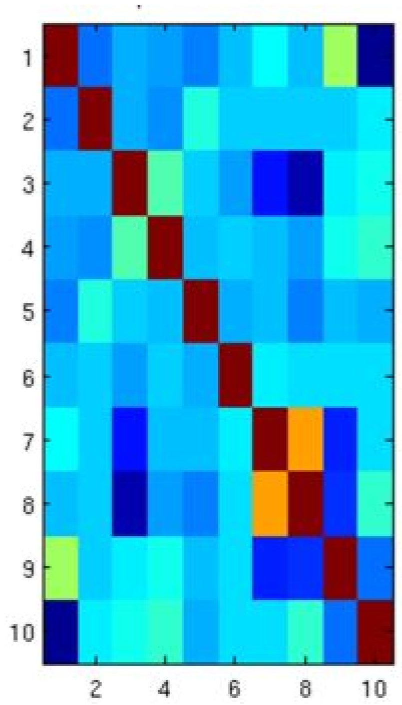
Analytical Solution to OLS when p=2

It easily follows that $$\frac{d^2\mathcal{L}}{d^2\beta} = \frac{2}{N}X^TX$$

- From this, it follows there is a **unique solution** iff the **eigenvalues** of the above matrix are **strictly positive**.

- This occurs precisely when **X has linearly independent features**.

- If X is **mean centered**, the above matrix is equivalent to the **correlation matrix**.



Dependent features - what goes wrong

Why we want to eliminate collinear/dependent features

Correlation results in instability

- $X^T X$ is **symmetric** and therefore has **non-negative eigenvalues.**

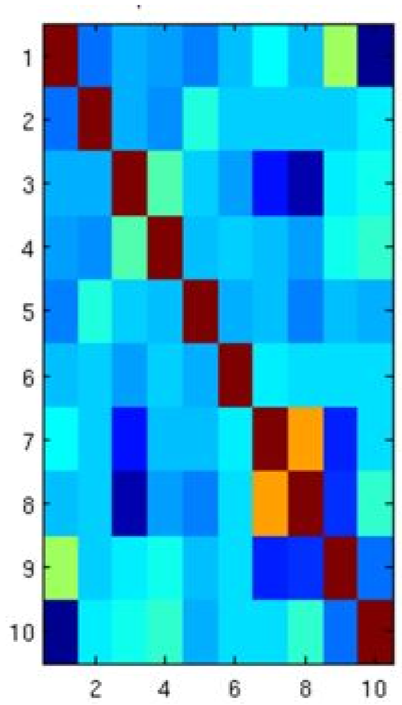
- They are **positive** precisely when the features of X are **linearly independent.**

- Let's assume for simplicity that **X is mean centered (fine but sometimes reasons why you might not).**

Imagine that X has two columns which are factors of one another. What can go wrong?

$$y = \alpha x_1 + \beta x_2$$

But our real rule is: $y = 5x_1$



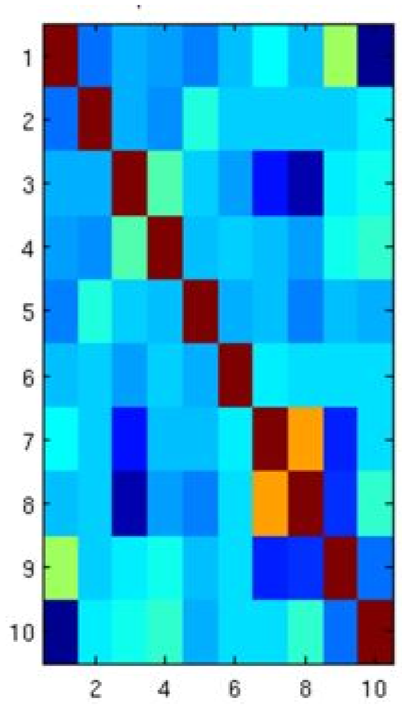
Correlation results in instability

$$y = \alpha x_1 + \beta x_2$$

But our real rule is: $y = 5x_1$

Then $y = -1000x_1 + 10005x_2$ is also a solution

Why is this a problem?



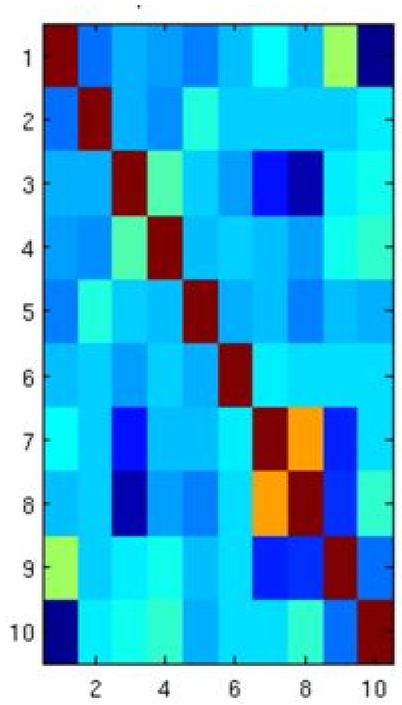
Correlation results in instability

Then $y = -1000x_1 + 10005x_2$ is also a solution

Imagine, more realistically, that $x_1 = x_2 + \text{noise}$

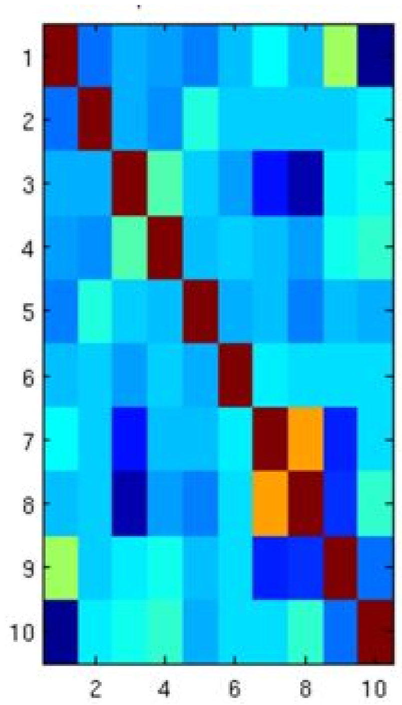
This creates **noisy output, and therefore decreased confidence.**

Conclusion: Make sure to eliminate collinear features in your model.



How to eliminate correlated features?

- Regularization (later in this lecture).
- Principle Component Analysis (next time).
- We made an assumption here that X was mean centered to tie things back to correlation, but we won’t always do this.



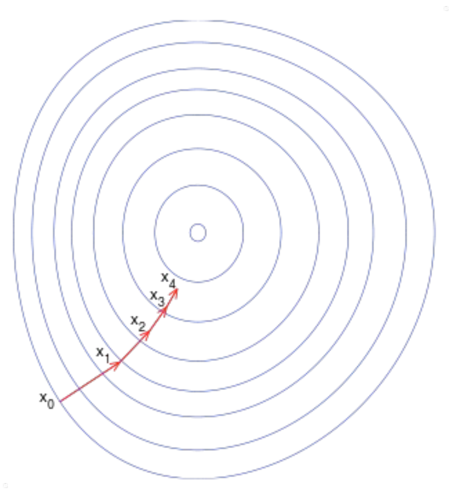
Gradient Descent

How do we minimize a function when there is no analytical solution?

Continuous Gradient Descent

Assume that $x(t)$ solves the equation:

$$\dot{x} = -\nabla f(x)$$



And that:

- Assume that f is strictly convex.
- That f is twice differentiable.

Then: $x(t)$ converges to the minimum of f exponentially fast.

Proof of Convergence of Gradient Descent

Differentiating and using the gradient flow, we have

$$\frac{d}{dt}(x - x_0)^2 = -2\nabla f(x) \cdot (x - x_0)$$

Next: How do we use strict convexity to obtain an estimate?

***Note*:** In reality we alway shave a discrete version of the above, and must choose a 'learning rate' (ie. time step size) - we will gloss over this for now.

Proof of Convergence of Gradient Descent

Using Taylor’s Law we have

$$f(x) - f(x_0) = \nabla f(x) \cdot (x - x_0) + \frac{1}{2}(x - x_0)^T D^2 f(\xi)(x - x_0).$$

$$f(x_0) - f(x) = \nabla f(x_0) \cdot (x_0 - x) + \frac{1}{2}(x - x_0)^T D^2 f(\tilde{\xi})(x - x_0).$$

Therefore:

$$-\nabla f(x) \cdot (x - x_0) = -(x - x_0)^T (D^2 f(\xi) + D^2 f(\tilde{\xi}))(x - x_0) \leq -M|x - x_0|^2$$

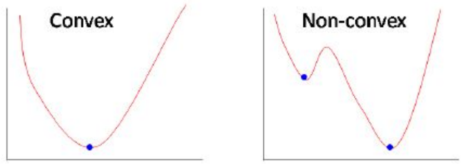
$$\frac{d}{dt}(x - x_0)^2 = -2\nabla f(x) \cdot (x - x_0)$$

$$\frac{d}{dt}(x - x_0)^2 \leq -m|x - x_0|^2$$

$$\Rightarrow |x(t) - x_0| \leq Ce^{-mt}$$

The importance of gradient descent.

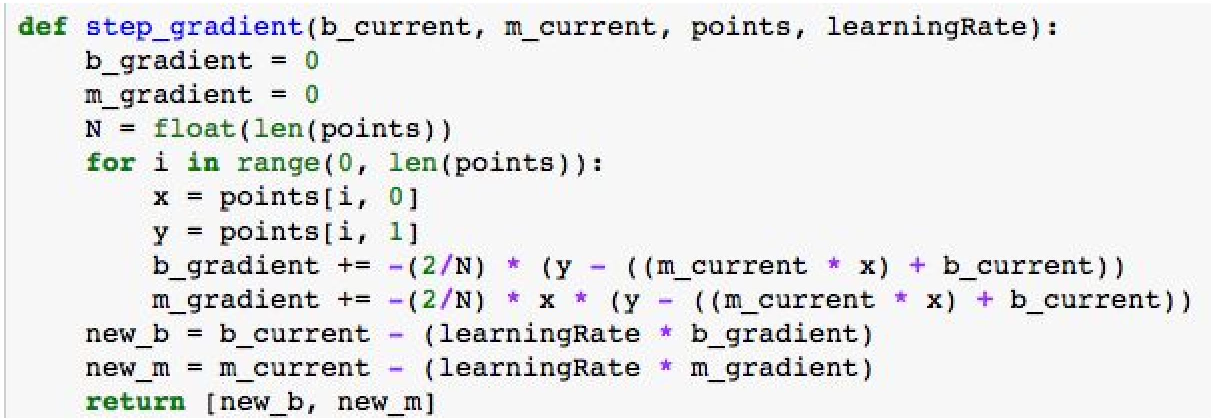
- Virtually all algorithms in predictive machine learning seek to minimize some a priori function of the data to the observed values.
- It only makes sense to try to minimize functions which are convex (at least locally). Otherwise it's common to get stuck in local minima.
- When there is no analytical solution, the solution must be obtained by following the steepest descent from a starting point to the minimum of the function.
- This will be true when we work with more advanced probabilistic methods as well, and more important to understand.



How is it computed in python?

$$\frac{\partial \mathcal{L}}{\partial \beta_j} = -\frac{2}{N} \sum_{i=1}^{N} (y_i - \beta \cdot x_i) x_j$$

$$\beta_n = \beta_{n-1} - \kappa \nabla_\beta \mathcal{L}(\beta_{n-1})$$



```
def step_gradient(b_current, m_current, points, learningRate):
    b_gradient = 0
    m_gradient = 0
    N = float(len(points))
    for i in range(0, len(points)):
        x = points[i, 0]
        y = points[i, 1]
        b_gradient += -(2/N) * (y - ((m_current * x) + b_current))
        m_gradient += -(2/N) * x * (y - ((m_current * x) + b_current))
    new_b = b_current - (learningRate * b_gradient)
    new_m = m_current - (learningRate * m_gradient)
    return [new_b, new_m]
```

κ is known as the learning rate - when computing we must choose a time step size (this can actually be important, but we won't focus on this)

Evaluating on testing data

We have a model - does it generalize?

How to evaluate on testing data?

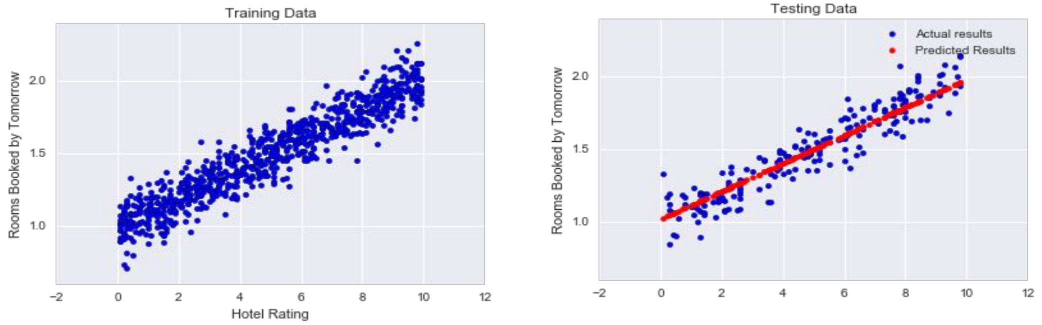
For regression problems, we generally use two possible metrics:

- **Root mean squared error:**

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$$

- **R-squared:**

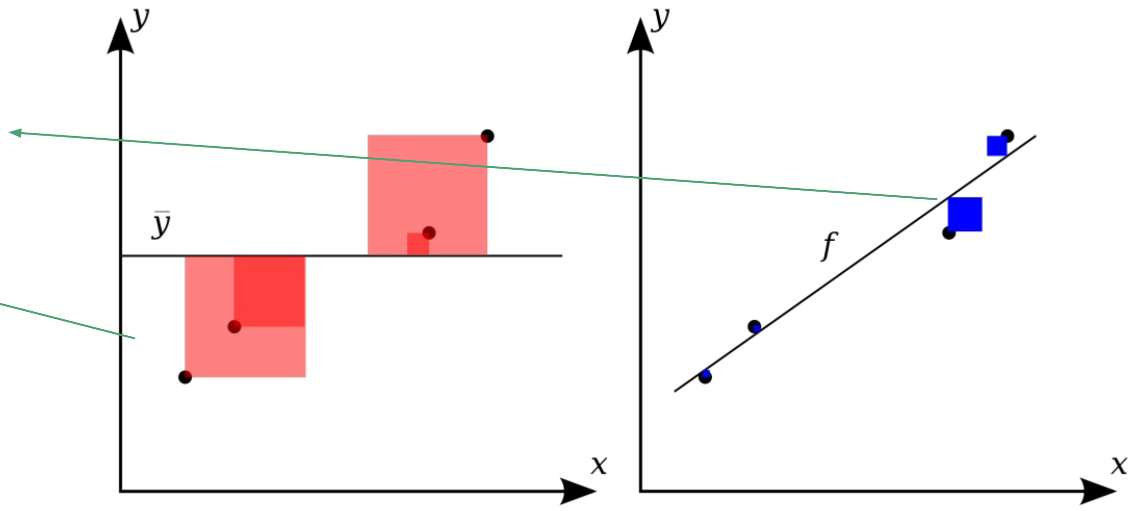
$$1 - \frac{\sum_{i=1}^{N}(f(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \longleftarrow$$



- This is the fraction of variance that our model is able to explain.

Coefficient of Determination (R^2)

$$1 - \frac{\sum_{i=1}^{N}(f(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$



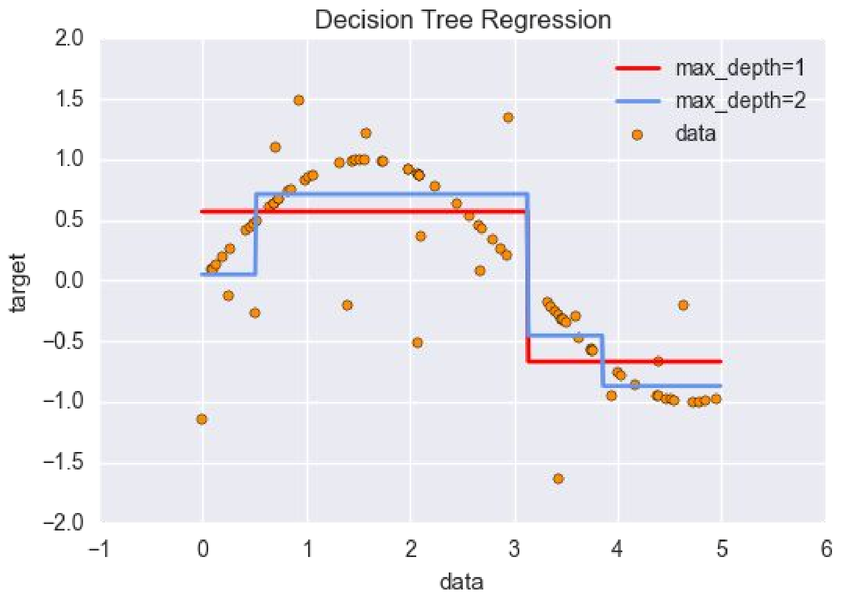
- This is the percentage of variance that our model is able to explain
- A perfect model would have the second term be 0, resulting in 1.
- If the model was simply the mean, the result would be 0.

What have we shown?

- Given any norm, by iterating under its gradient flow, we can obtain a solution to the minimization problem (it’s unique iff the norm is strictly convex).
- Most minimization problems don’t have analytical solutions like the L2 norm - **this is how scikit-learn codes these algorithms (with GD).**
- We will now see a live demo of this in Python.
- https://github.com/Columbia-Intro-Data-Science/APMAE4990-/blob/master/notebooks/Lecture1%20-%20Introduction-to-Regression.ipynb
-

Decision Tree Regression

First: A 1d example

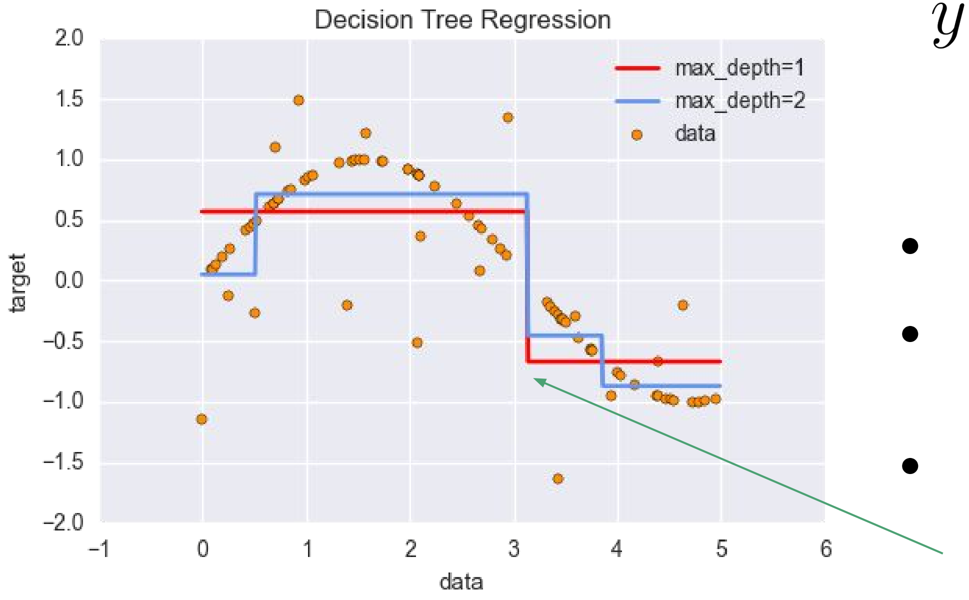


$$y = \sin(x) + \text{noise}$$

- 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.**

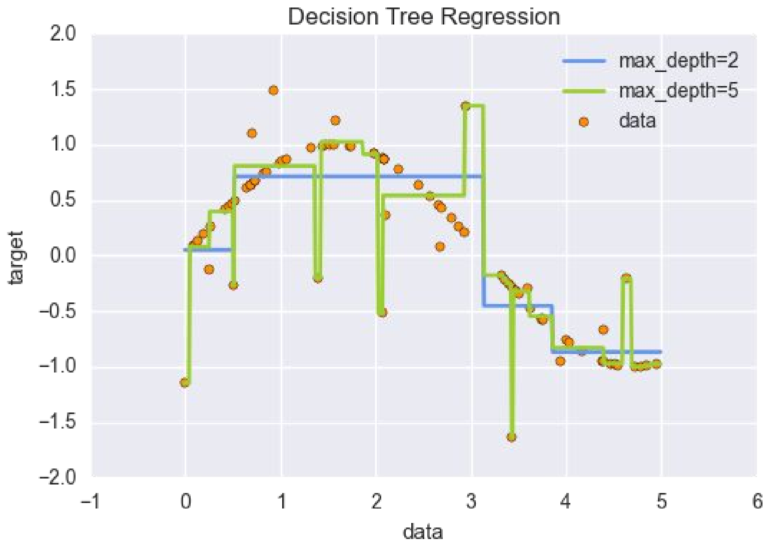
First: A 1d example



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

- 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.**

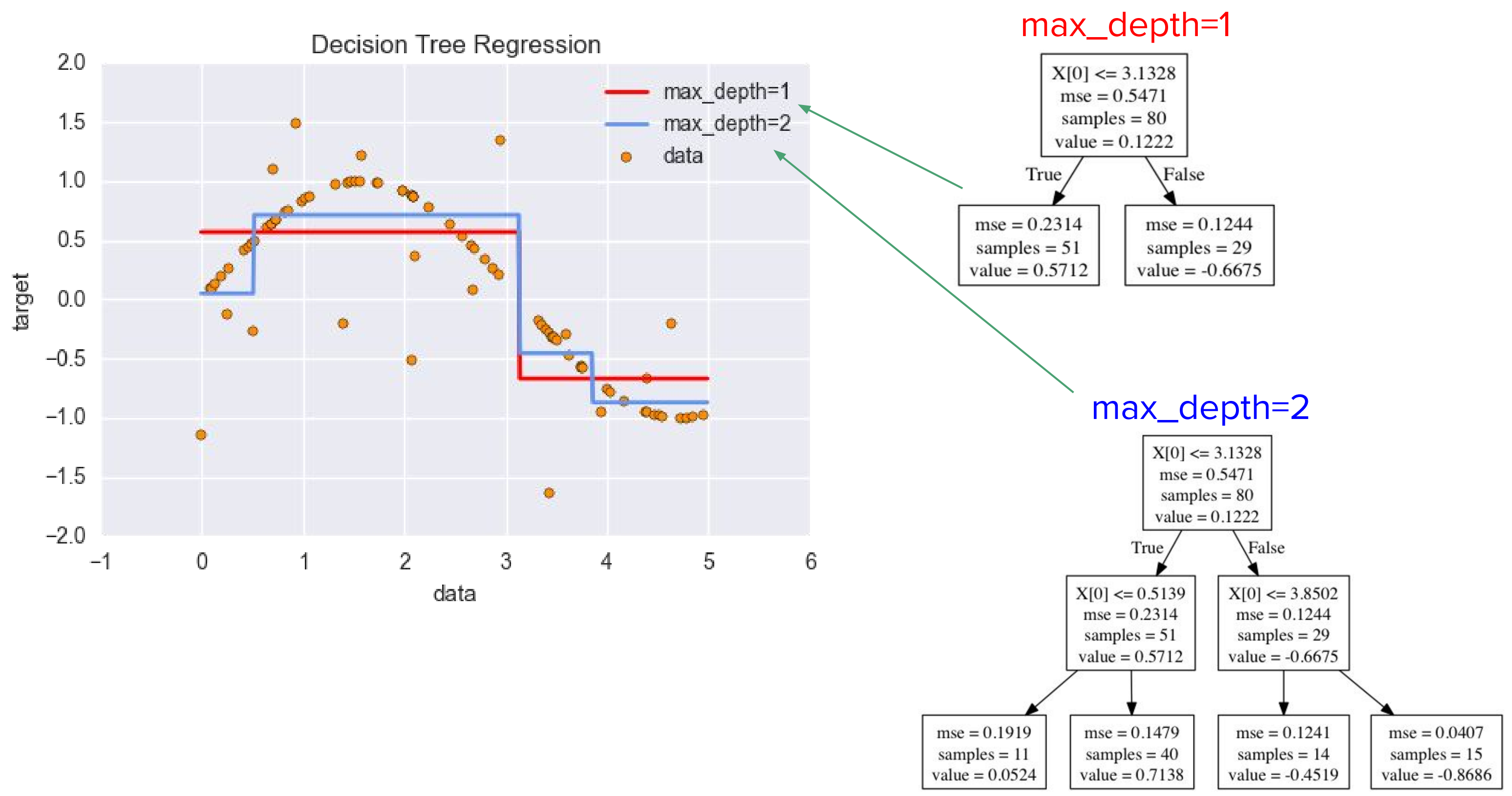
Increased depth can be bad



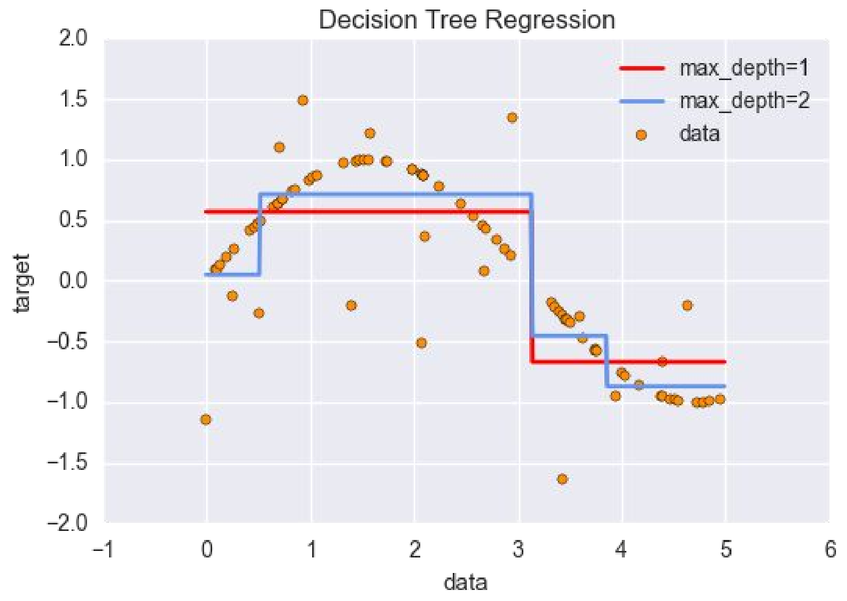
$$y = \sin(x) + \text{noise}$$

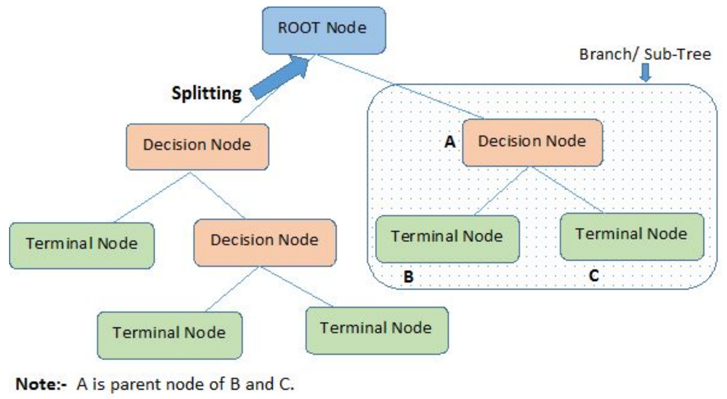
- In general, if one adds more depth to the tree, there is a risk of over-fitting. Look at the unwanted variance we have picked up in this example.
- We will learn next lecture how to choose the perfect depth number!

How do the decisions work?

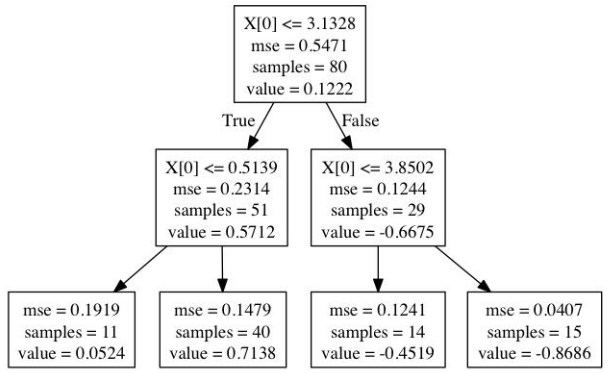


Some terminology

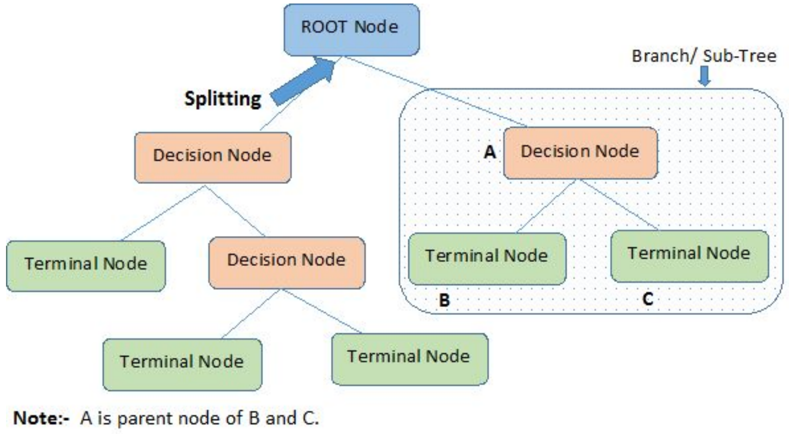




max_depth=2



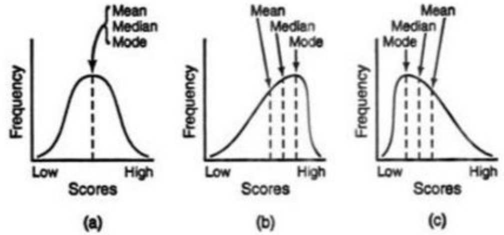
Some terminology



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.

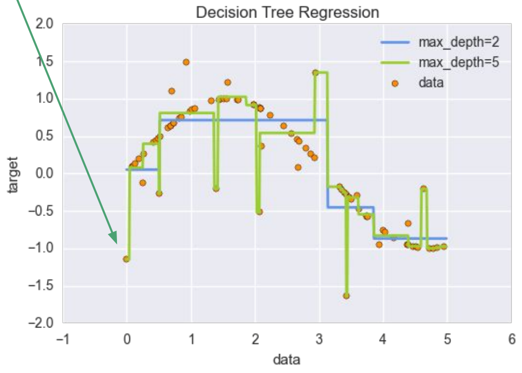
What are the advantages of decision trees?

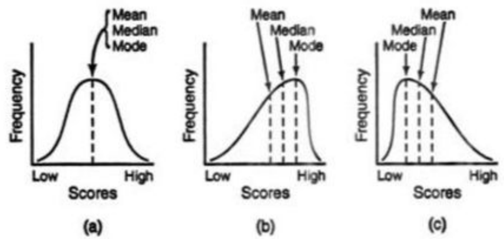
- Very easy to interpret.
- Makes **no a-priori assumptions about the structural form of the data** (as linear models do). For instance, works well with non-linear data.
- More "robust" than linear models, meaning **less sensitive to outliers**. This is for the same reason that the median is less sensitive 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 (doesn't require one-hot encoding as linear models always do - Exercise: Why?)



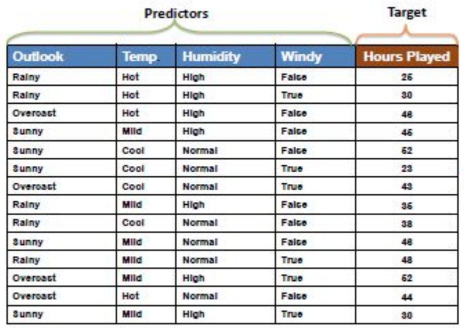
What are the disadvantages?

- Prone to **over-fitting** (more on this next lecture).
- Can lose information when dealing with continuous variables, since it needs to make a finite number of splits.

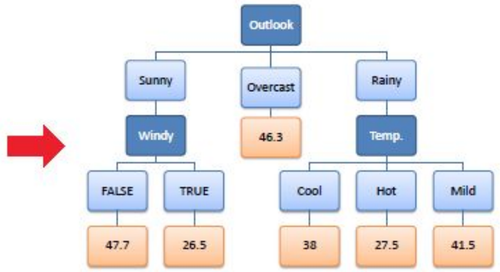




More than one variable?



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

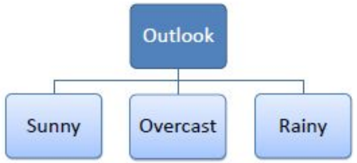


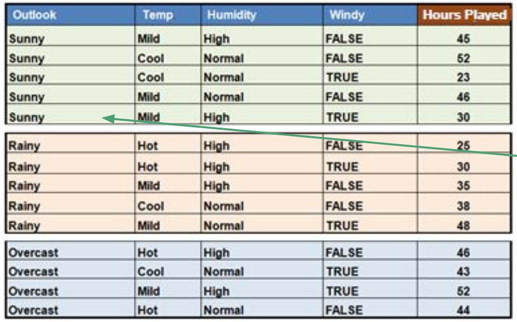
- A decision tree is simply a recursive set of decisions which leads to a result.
- It can be a real value or a class.
- Decision trees are very simple to understand, but are somewhat complex to explain when multiple features are involved.

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.

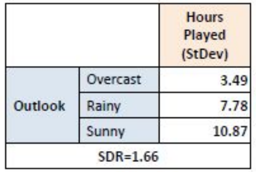
Example taken from: http://chem-eng.utoronto.ca/~datamining/dmc/decision_tree_reg.htm

How do we choose the root node?

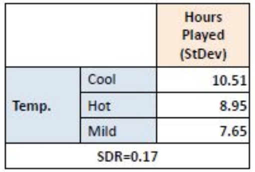




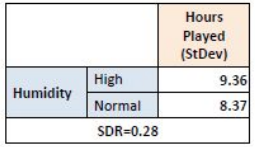
Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast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



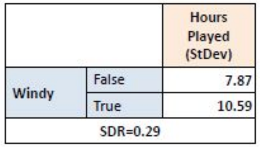
		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		



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



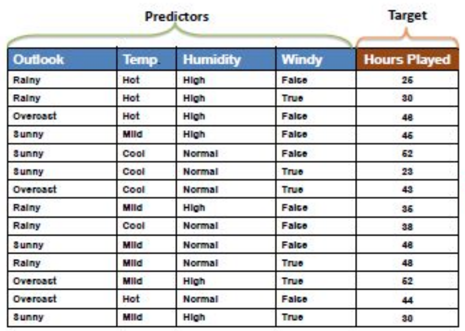
		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		



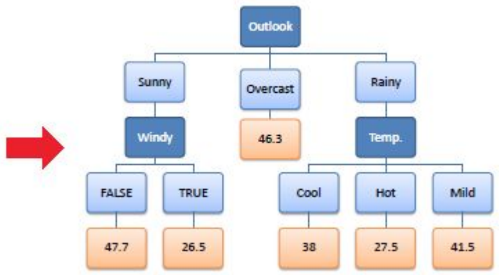
		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

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

More precise description



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



For zero depth we minimize:

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

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

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

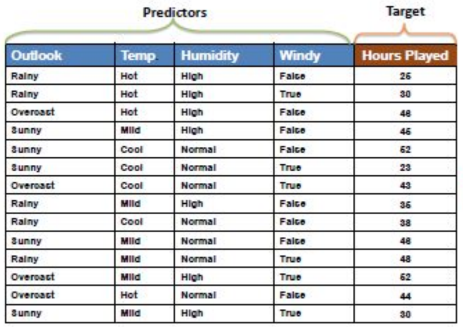
$$= \sum_{i, x_i = x_j}(y_i - \bar{y}_j)^2 \frac{1}{|X = x_j|}$$

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

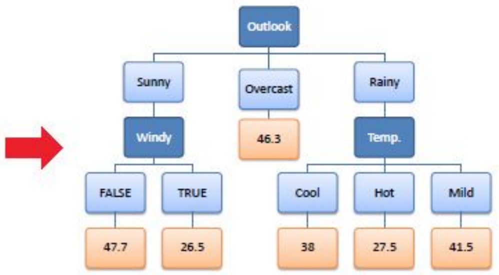
$$\mathrm{VarRed}(Y, X) = \mathrm{Var}(Y) - \sum_{j}\mathrm{Var}(Y|X = x_j)$$

Variance Reduction

Let’s try Outlook first



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



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

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

$$\sum_j \mathrm{Var}(Y|X = x_j) = \frac{4}{14}(3.49) + \frac{5}{14}(7.78) + \frac{5}{14}(10.87) = 7.66$$

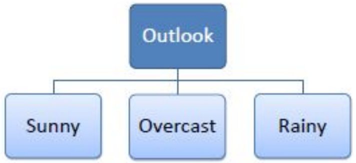
$$\mathrm{Var}(Y|X = x_j) = \sum_{i=1}^{N}(y_i - \bar{y}_j)^2 p(y_i|x_j)$$

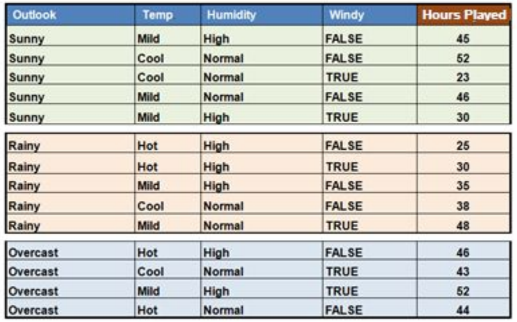
$$= \sum_{i, x_i = x_j}(y_i - \bar{y}_j)^2 \frac{1}{|X = x_j|}$$

Therefore there is a variance reduction of **1.66 for the outlook variable.**

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

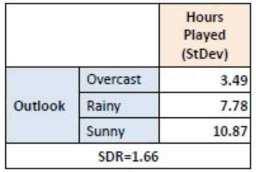
Now we've split based on Outlook



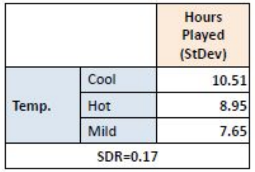


Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48
Overcast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

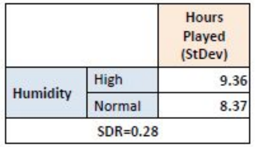
- These groups now have less variance in the data. And Outlook reduced variance the most. Now we continue recursively.
- A branch set with standard deviation more than 0 needs further splitting - we need pure classes on the final leaf nodes.



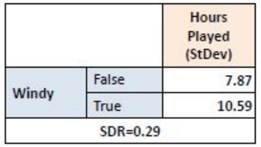
		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		



		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		

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.

Follow the notebook here:

https://github.com/doriang102/Columbia_Data_Science/blob/master/notebooks/Lecture1%20-%20Introduction-to-Regression.ipynb

Appendix - mean center without loss of generality

$$y = \beta \cdot X + c$$

$$y = \beta \cdot (X - \bar{X} + \bar{X}) + c$$

$$y = \beta \cdot (X - \bar{X}) + c + \beta \cdot \bar{X}$$

$$y = \beta \cdot (X - \bar{X}) + d$$