Decision Trees

Decision Trees

Random Forrest

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Lesson Objectives

- Learn the following algorithms
  - Decision Trees
  - Random Forest

Where Are We?

Category	Sub Category	Example	Algorithms
supervised	Regressions	- Predict house prices	- Linear Regression - Logistic
	Classifications	- Cancer or not - Spam or not	- **➔ Trees (random forest ..etc)** - SVM
Unsupervised	Clustering	- Group customers (soccer mom, nascar dad)	- Kmeans - Hierarchical clustering
	Dimensionality reduction	- Reduce the number of attributes to consider	- PCA
Semi-supervised		(large amount of data, but only a very small subset is labelled)	

Exercise: College Admission

- Consider college application and admittance data
- **Inputs**:
  - GRE: max 800
  - GPA: 1.0 to4.0
  - Rank: 1 (better) to 4
- **Output**
  - Admitted : Yes or No
- We did this Multiple Logistics Regression before.
  Now we will try SVM

gre	gpa	rank	admitted
380	3.6	3	No
660	3.67	3	Yes
800	4	1	Yes
640	3.19	4	Yes
520	2.93	4	No
760	3.0	2	Yes
400	3.08	2	No
700	4.0	1	Yes
500	3.17	3	No

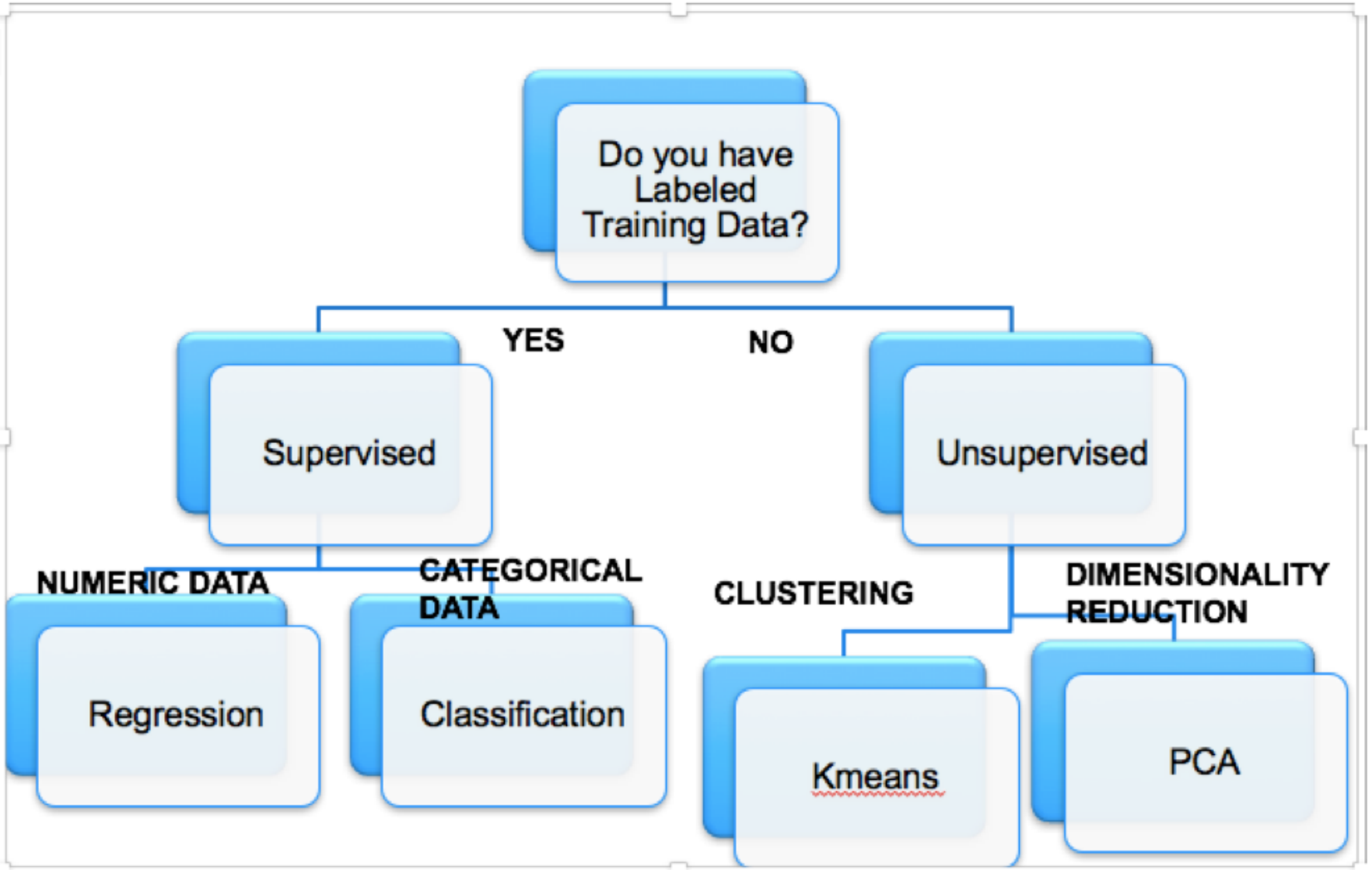
Decision Trees

➔ **Decision Trees**

Random Forrest

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Example of Decision Tree



Decision Trees Intro

- Decision Trees are in important algorithm in ML
  - Has long history
  - Classics : Binary Tree (guessing game, customer support)
  - Modern variations : Random Forest, Boosted trees, Bagged trees

- Trees can be used for both classification and regression. So called **'Classification and Regression Trees' (CART)**

- Decision trees are used mostly for classification – "**classification trees**"

Decision Trees Advantages / Disadvantages

Advantages	- Simple to understand and interpret - Requires little data preparation - Able to handle both numerical and categorical data - Performs well with large datasets - Easy to extract in order to move to another language
Disadvantages	- Not as precise as linear or logistic regression - Very sensitive to input data Tree may change significantly with a small change in data

Decision Tree Use Cases

- Medical diagnosis
  - Start from pattern of symptoms
  - Define classes as clinical subtypes or conditions
  - Result: patients with a condition who should receive different therapies.

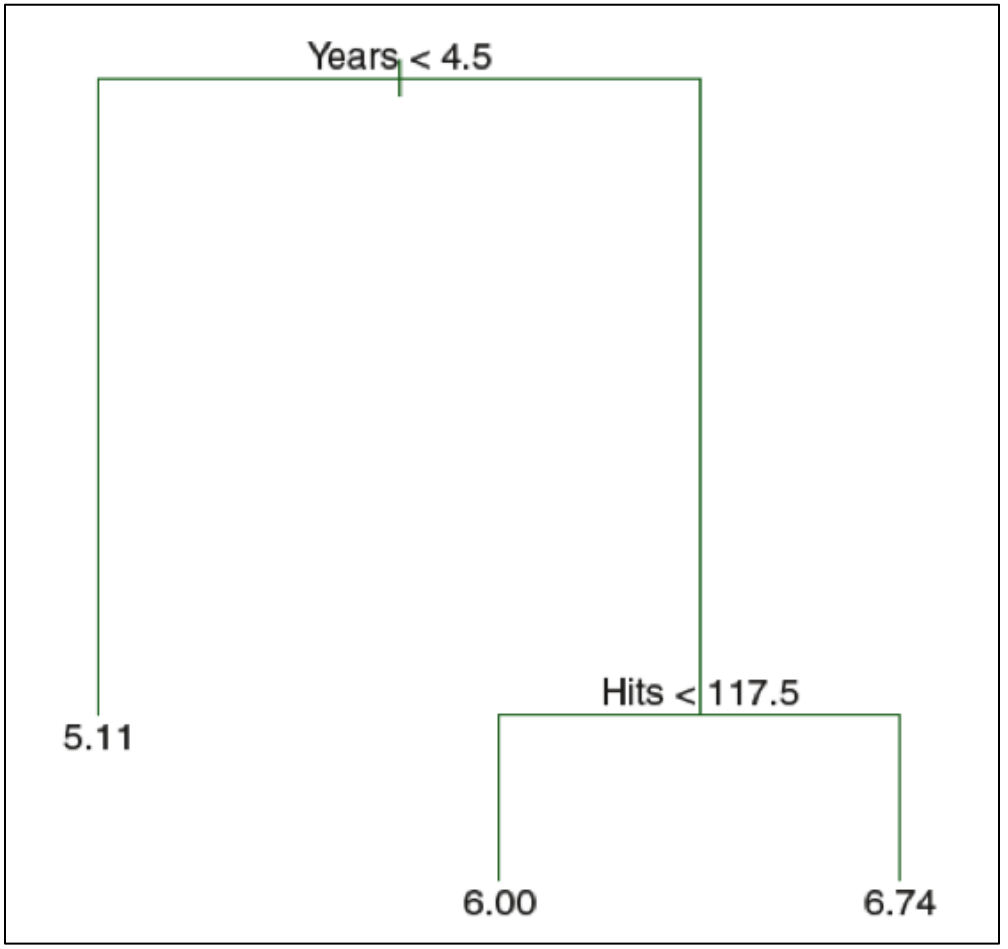
- Other examples
  - Business planning
  - Technical diagnostics
  - Customer support

Baseball Player Salary Prediction with Regression Tree

- We are looking at two attributes of a baseball player
  1 – years experience
  2 – hits

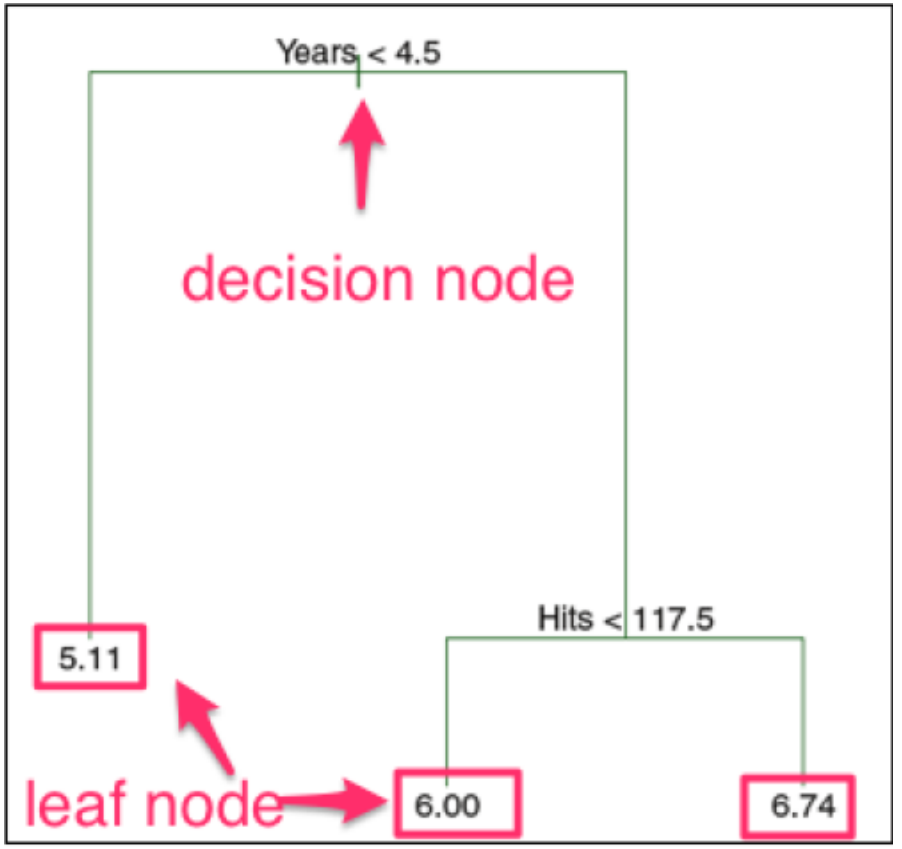
- Salary is in in log scale. Actual salary e^x thousands

Log salary	Salary (e^x)
5.11	165k
6	403k
6.74	845k



Tree Data Model

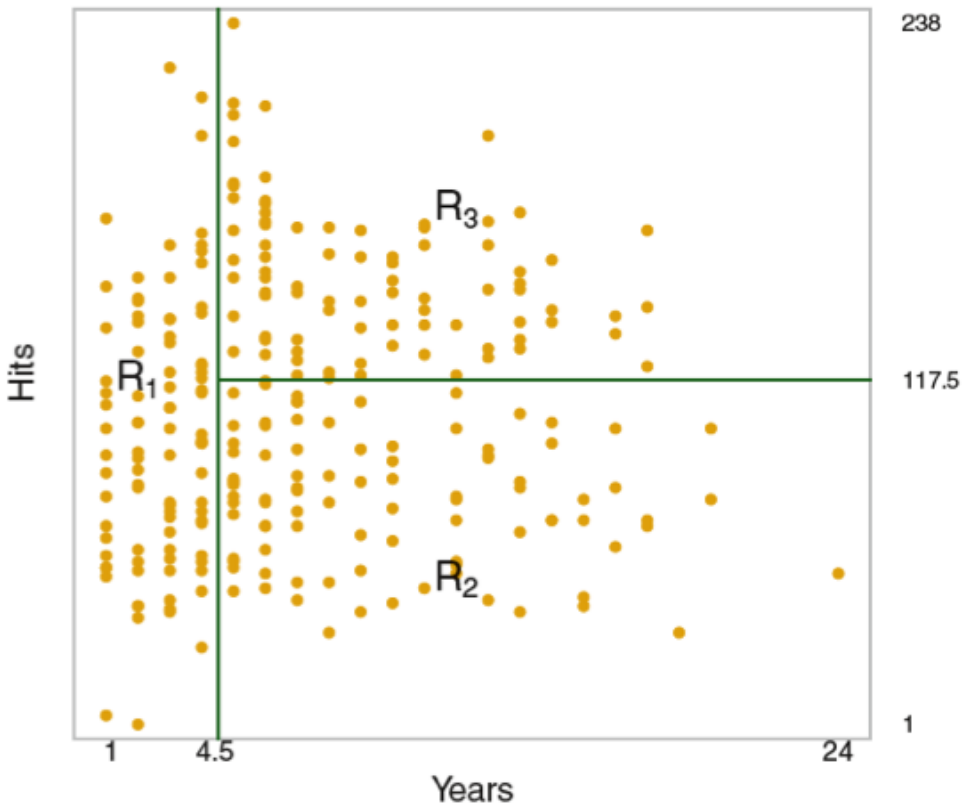
- Binary tree
- Each node has
  - Input : X
  - Split point (years < 4.5)
- Leaf Node has output variable (Y)



Log salary	Salary (e^x)
5.11	165k
6	403k
6.74	845k

Tree Algorithm

- Tree partitions the input space
- Each input variable as a dimension on an p-dimensional space
- When p=2 → rectangle
  p > 2 → hyper rectangles (high dimensional)
- New data gets filtered through the tree and lands on one rectangle
  - That is the prediction
- Example
  - Input
    - Years > 4.5?
    - Hits > 117.5?
  - Output : R3



Creating a Decision Tree

- We divide all possible values for Y into regions (R_1, R_2 … R_j)
  - Distinct
  - Non-overlapping (one data point only assigned to one region)
- Regions R_1, R_2, …, R_J are high-dimensional rectangles
  - Also called “boxes”
- We want to find a set of boxes that will approximate our data
- Find the error (prediction vs actual) per each region
  - This is RSS (Residual Sum of Squares)
- Try to **minimize RSS across all regions**
  - The formula calculates RSS across all Regions (1 to J)

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

Decision Trees – Region Formulation

- We would need to consider every possible combination of breaking our data into regions R_1, R_2, …, R_J
  - But that is too hard ☺
- Instead, we will take a top-down, "greedy" approach
  - Also known as "recursive binary splitting"
- First, let us illustrate this with an example

Example: Guessing Game

- One person chooses a number in a certain range
- The other person guesses this number, using as few steps as possible
- What is the best strategy?

Guessing Game - Solution

- Let us say the first person choose a number in the range
  - 1 through 1000

- The second person should ask
  - Is it bigger than 500? – Say, the answer is “no”
  - Next: is it bigger than 250? – Say, the answer is “yes”
  - Next: is it bigger than 375?

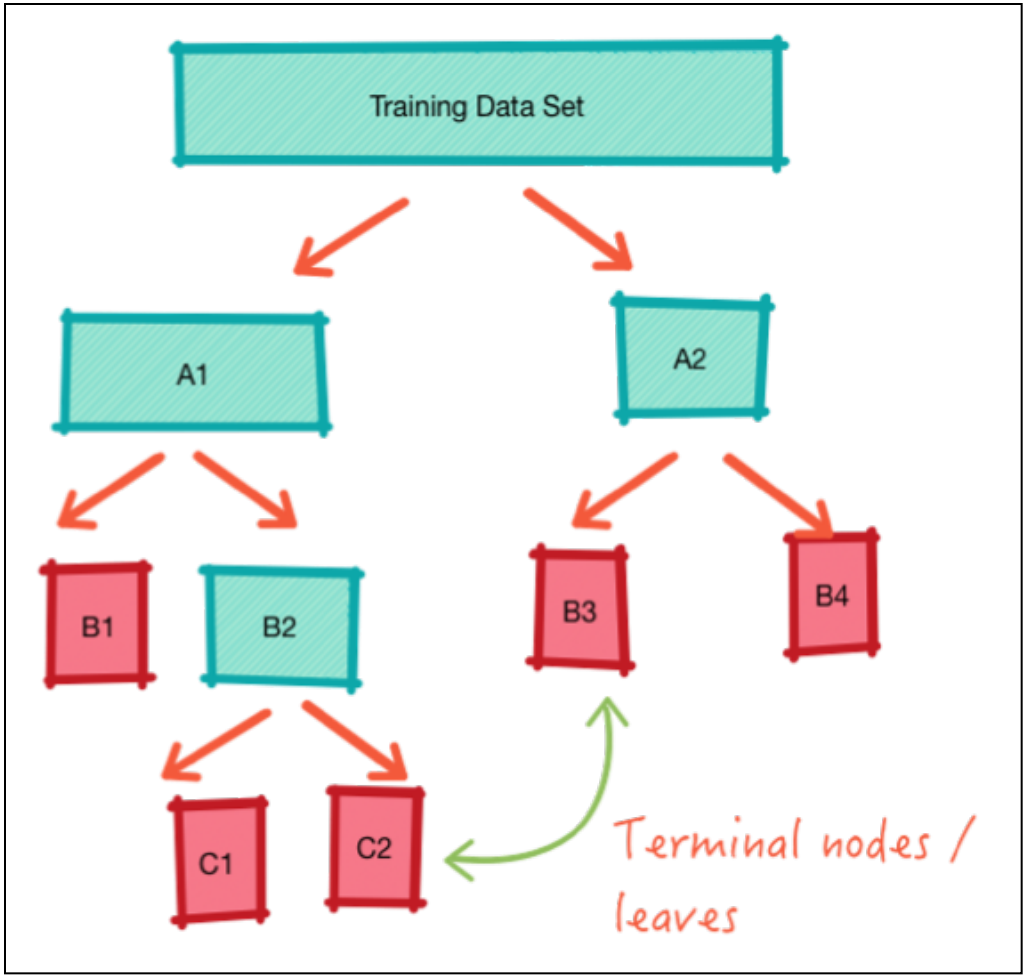
- “Greedy algorithm” for decision trees works in a similar way

How Greedy Decision Tree Algorithm Works

- Take all predictors X_1 through X_n
  - Find the set of dividers S_1 through S_n which will minimize RSS
- Take all predictors X_2 through X_n
  - Find the set of dividers S_2 through S_n which will minimize RSS
- Continue until
  - All nodes are leaf nodes or
  - No more improvement in RSS can be obtained
- You may also have to prune the tree later, to simplify it
- A complete solution is not known (NP-complete)
- Practical algorithms are based on some parameters to prevent runaway tree growth

What it means to be greedy?

- Select the *best split* from a set of possible splits
- Maximize the information gain at a tree node
- Greedy algorithm may not find the best tree
  - Grabs the best at each step
  - Does not consider all steps together



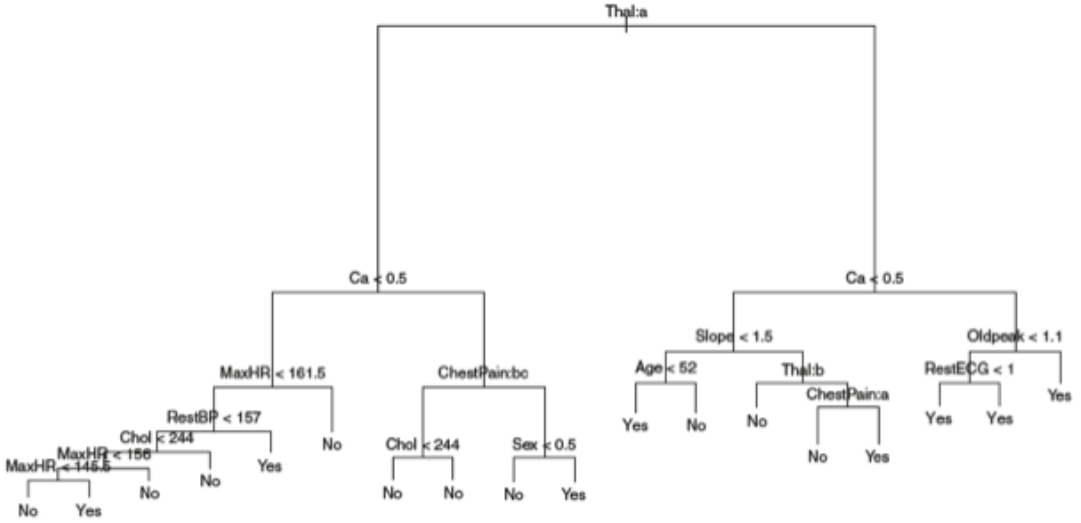
Tree Pruning

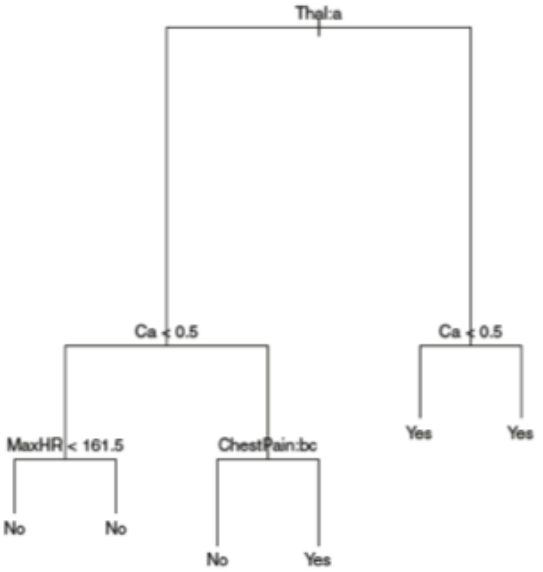
- Trees can grow arbitrarily large / deep
- While they predict training set well, they may overfit !
- So prune the tree
  - Reduce overfitting
  - Stable predictions, even if they are not most accurate
  - Make the tree simpler → easier to understand
- How to prune?
  - Walk through each leaf node
  - Evaluate the effect of removing it using hold-out test set
  - Stop removing when no further improvements can be made
- A 'sub tree' is chosen after pruning
  - We can not consider every possible sub-tree.. Too many!
  - Use a heuristic called 'minimize alpha / tuning parameter'

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Tree Pruning

- Heart condition tree
- To simplify
  - Cut away branches
  - Combine nodes





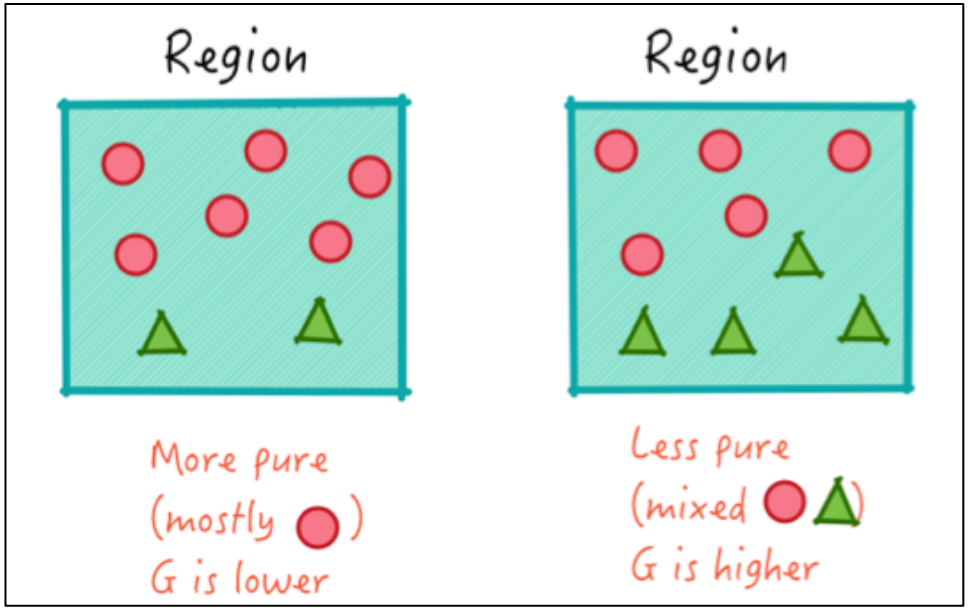
Classification Decision Trees

- Instead of predicting a number, predict a classification (Spam / NotSpam)
- In Regression Trees we use RSS (minimizing RSS) to find best regions
- In Classification Trees we can not use RSS (the observations may be not be numeric)
- Instead
  - We should predict the most common class
  - Therefore, let's optimize the error rate
- Two choices are
  - Gini Index
  - Entropy

Gini Index (G)

- Gini index measures the 'purity' of each node (how mixed data is in one Region)
- P_k = proportion of class k in that region
- A region with all classes of same type will have G = 0
- If all of p_{mk} are close to 0 or to 1, G is small
- If region has 50%-50% mix then G = 0.5 (worst purity)
- **Goal: minimize G score**

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



Gini Index Example

- Here we have a classifier : class-1 and class-2
- P_1 : proportion of class-1
- P_2 : proportion of class-2
- Calculating Gini index as follows

$$G = 1 - ({p_1}^2 + {p_2}^2)$$

Gini						
Scenario	Class 0	Class 1	Count	Class 0/Count	Class 1/Count	Gini (1 - p1*p1 - p2*p2)
1	10	10	20	0.5	0.5	0.5
2	19	1	20	0.95	0.05	0.095
3	1	19	20	0.05	0.95	0.095
4	15	5	20	0.75	0.25	0.375
5	5	15	20	0.25	0.75	0.375
6	11	9	20	0.55	0.45	0.495
7	20	0	20	1	0	0

About the "Prosper" Dataset

- https://www.prosper.com/

- America's first peer-to-peer lending marketplace
  - 2 million + members
  - $ 2 B + in funded loans

- Dataset is public
  - 113,937 loans with 81 variables

Variables in the Prosper Dataset

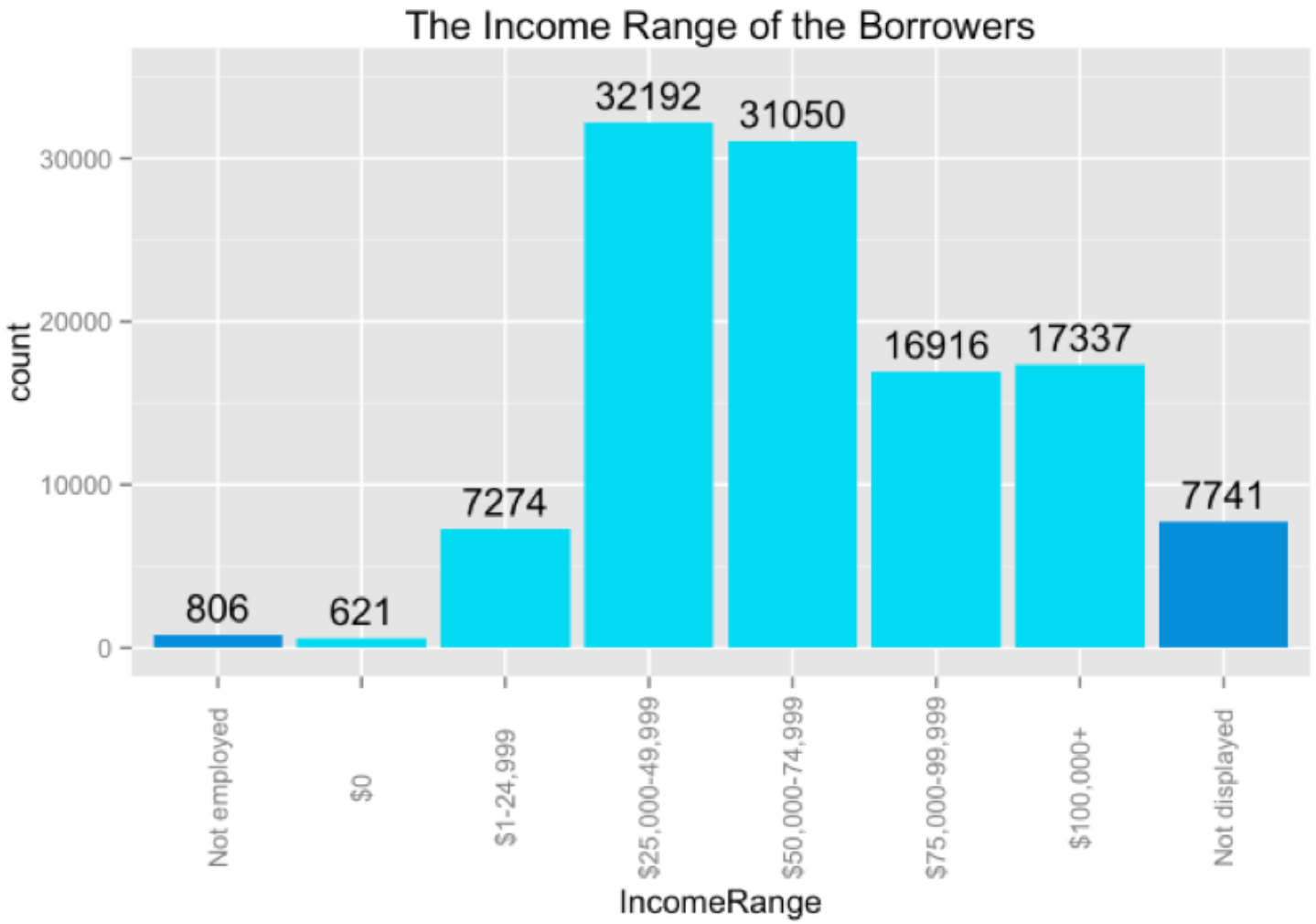
#	Name	#	Name
1	ListingKey	2	ListingNumber
3	ListingCreationDate	4	CreditGrade
5	Term	6	LoanStatus
7	ClosedDate	8	BorrowerAPR
9	BorrowerRate	10	LenderYield
11	EstimatedEffectiveYield	12	EstimatedLoss
13	EstimatedReturn	14	ProsperRating..numeric
15	ProsperRating..Alpha	16	ProsperScore
17	ListingCategory..numeric	18	BorrowerState
19	Occupation	20	EmploymentStatus
21	EmploymentStatusDuration	22	IsBorrowerHomeowner
23	CurrentlyInGroup	…	And so on, till #81

Prosper Data Examples

Value Name	Value Data
ListingKey	1021339766868145413AB3B
ListingNumber	193129
ListingCreationDate	2007-08-26 19:09:29.263000000
CreditGrade	C
Term	36
LoanStatus	Completed
ClosedDate	2009-08-14 00:00:00
BorrowerAPR	0.165
…	…

First Interesting Fact about Borrowers

- Borrowers' Incomes



Lab Preview

- In the lab, we will want to
  - Find interesting facts about the borrowers
  - Such as employment status below
  - (Note that unemployed can also get loans)

EmploymentStatus	count
Employed	18393
Part-time	1060
Self-employed	3045
Not employed	583
Other	924
Full-time	25016
Retired	703

Lab Goal

- Our real goal in the lab is:
- Build a decision tree which will tell us whether to fund or not to fund the loan

Lab 9.1 & 9.2 : Decision Trees

- **Overview**

  Create a classification decision tree

- **Approximate Time**

  30 mins

- **Instructions**

  Follow appropriate Python, R, or Scala instructions

  - **9.1 : decision-trees/decision-tree-1-college-admission**
  - **9.2 : decision-trees/decision-tree-3-prosper2-pipeline.ipynb**

Random Forrest

Decision Trees

➔ **Random Forrest**

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Decision Tree Problems

- **Decision Trees' pros**
  - Works rather well
  - Decision are visual and are easy to explain
  - Easy to scale to large datasets

- **Decision Trees' drawbacks**
  - They are not stable
  - They are not precise

- **In Machine Learning terms**
  - Decision Trees have high variance (bad)
  - Decision Trees have low bias (good)

Bias-Variance Tradeoff

- **Variance** is the amount that the estimate of the target function will change if different training data was used.
  - **Low Variance**: Suggests small changes to the estimate of the target function with changes to the training dataset.
  - **High Variance**: Suggests large changes to the estimate of the target function with changes to the training dataset.

- **Bias** are the simplifying assumptions made by a model to make the target function easier to learn
  - **Low Bias**: Suggests less assumptions about the form of the target function
  - **High-Bias**: Suggests more assumptions about the form of the target function.

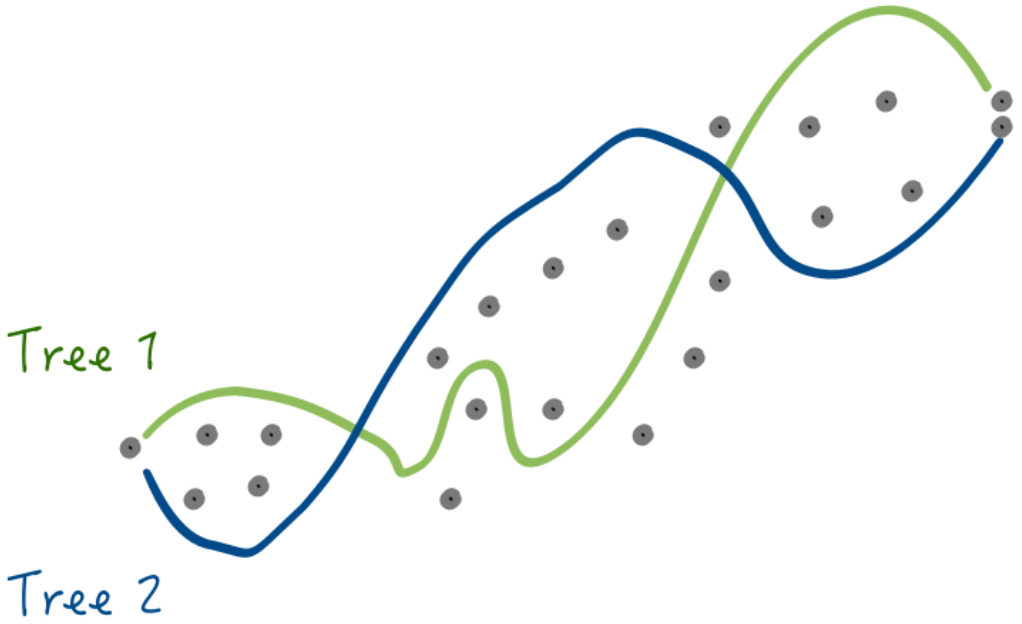
Bias-Variance Tradeoff

Low Bias (good)	High Bias (not good)
Decision Trees, k-Nearest Neighbors and Support Vector Machines	Linear Regression, Linear Discriminant Analysis and Logistic Regression
More able to adopt to complex data	May not be able to adopt to complex data

Low Variance (good)	High Variance (not good)
Modestly influenced by change of data	Strongly influenced by change of data
Parametric methods usually have low variance	nonparametric machine learning algorithms that have a lot of flexibility have a high variance
Decision Trees, k-Nearest Neighbors and Support Vector Machines	Decision Trees, k-Nearest Neighbors and Support Vector Machines.

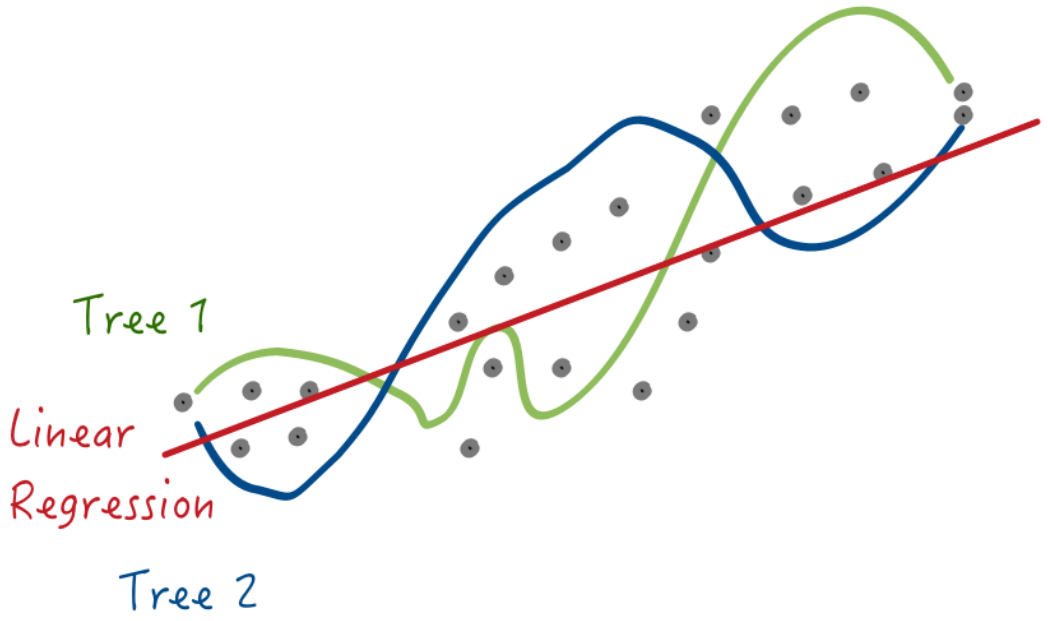
High Variance

- A tree may fit some of the data well
- But another tree may fit the other part of the data
- Neither of them will work well in the real world



Low Bias

- Decision Trees have low bias, and that is good
- By contrast, Linear Regression has high bias
  - It has less capacity to reflect complex data



Bootstrap Aggregation (Bagging)

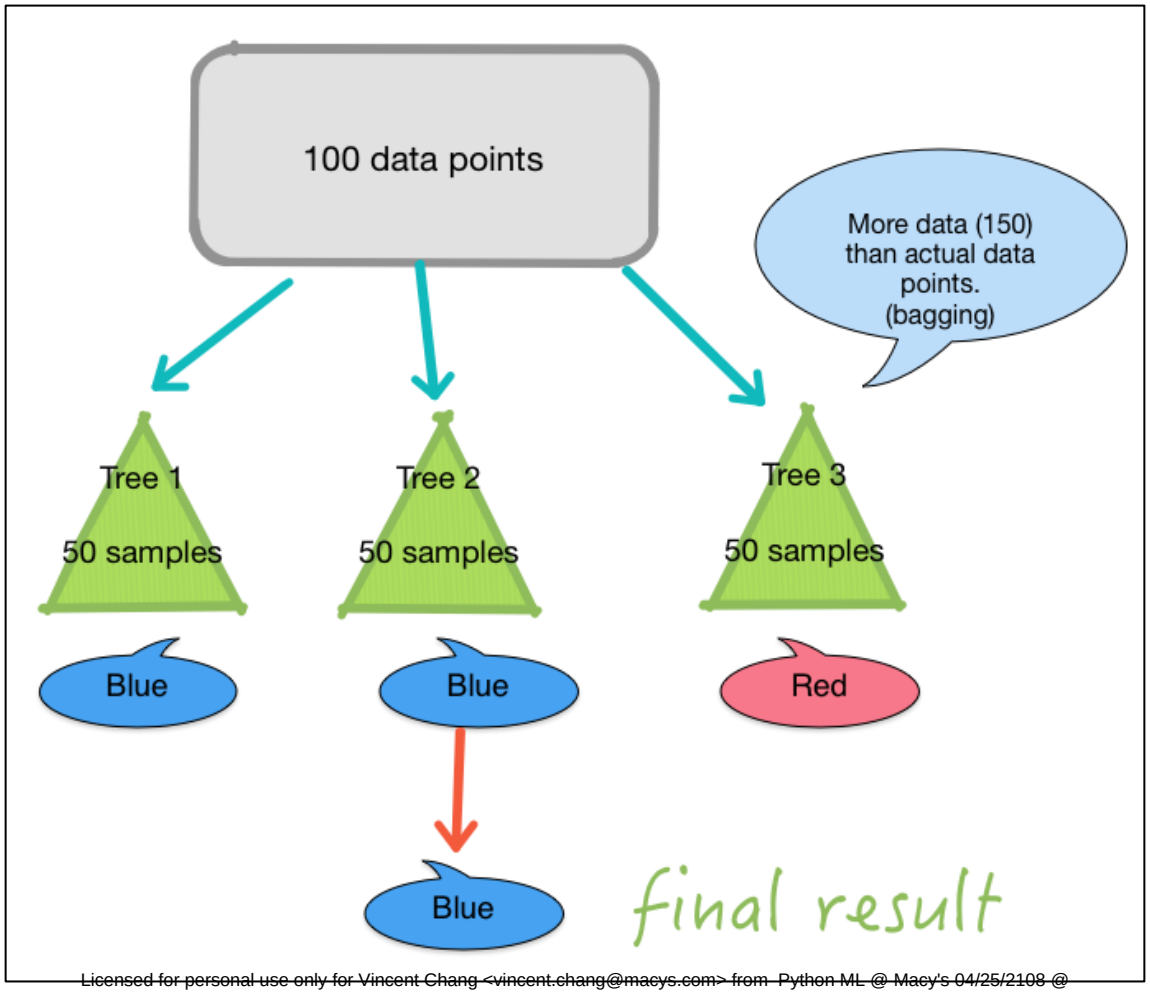
- Bagging is a simple and very powerful ensemble method.
- Ensemble method is a technique that combines predictions from multiple ML algorithms
- This results in more accurate prediction than any individual model

Data Bagging

- Say, we have only one set of data
- But, we want many sets
  - Let us randomly select a subset
  - This is our training data set
  - Return the elements back into the main set
  - Repeat
  - One element may be picked multiple times
  - This is called "**Bootstrapping**" (Sampling With Replacement)
- Thus, we generate many set of data for training
- This process is called "**data bagging**"

Bagging (Boosting Aggregation) In Action

- We create 3 trees with **Boosting**
- Each tree is predicting blue or red
- Final result is **aggregated** (**bagging**)



Data Bagging Performance

- We specify number of trees to create
- More trees will take more time
- Stop creating more trees when accuracy stops increasing
  - This can be validated by **cross-validation** testing

- Pros:
  - It will not over-fit the data.
    Because data is randomly split

- Cons:
  - Even with bagging trees will have lot of structural similarities
  - Their predictions will be highly co-related

Improving Bagging

- Data Bagging (selecting random subsets of data) solved overfitting problem
  - But we still have a problem of trees predicting highly correlated results

- Solution:
  We also select features randomly!

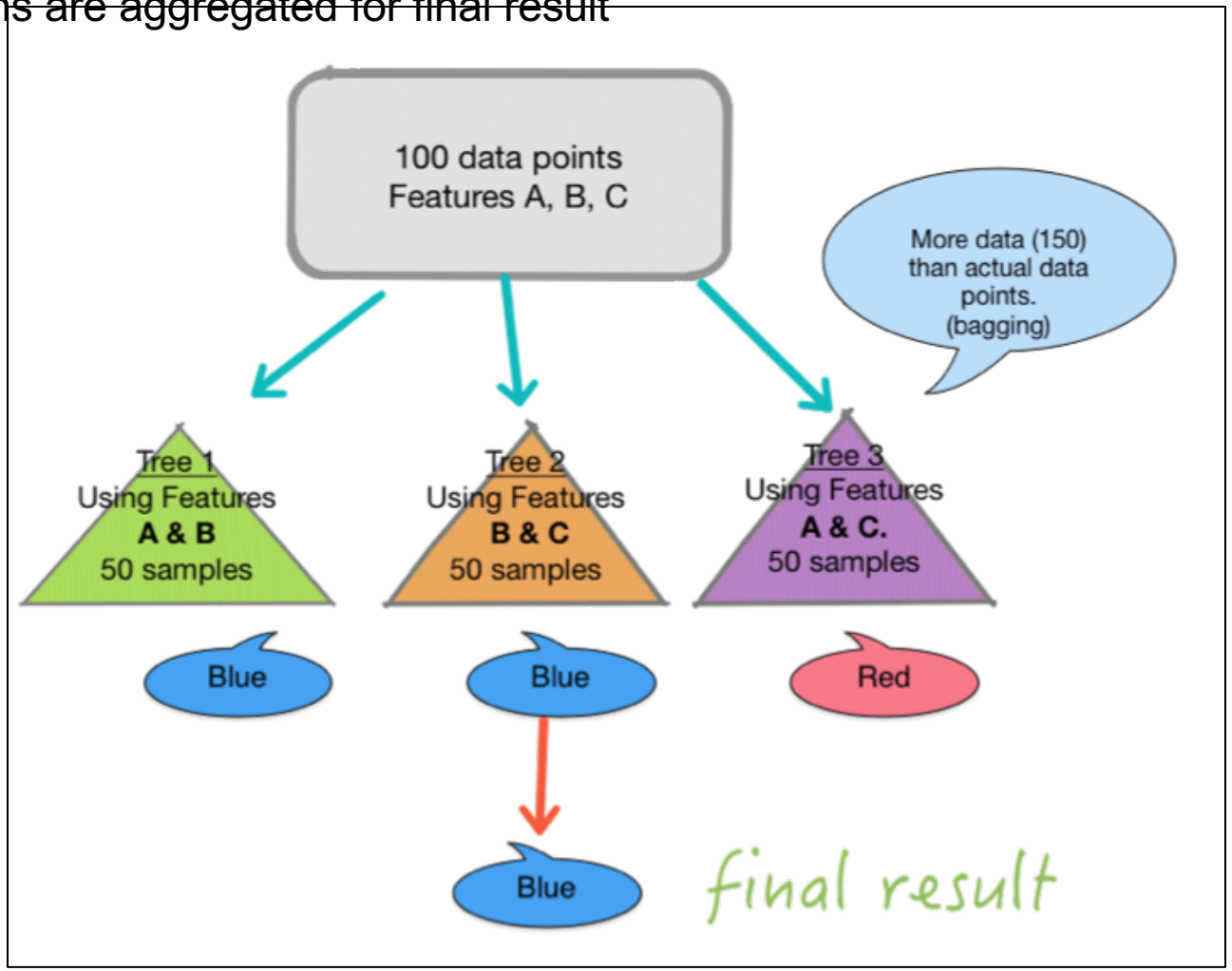
- For each tree, we will select a different set of features

- Then we will average the results

- This is called "**feature bagging**"

Data & Feature Bagging in Action

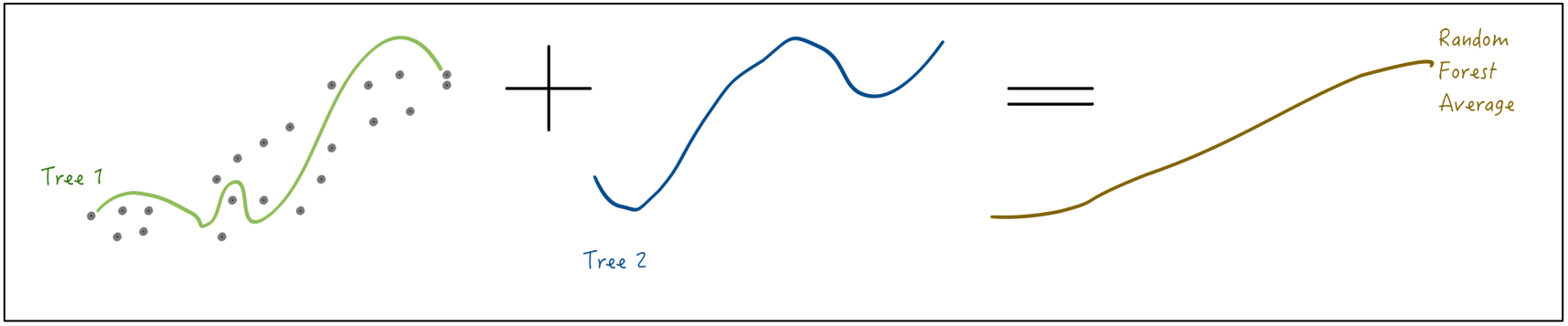
- We have 3 trees each operating on
  - Randomly selected subset of data
  - And randomly selected features (A,B,C)
- Their predictions are aggregated for final result



Random Forests Idea

- We want to keep low bias but add low variance

- **Approach**
  - Generate many decision trees
  - Each tree will operate on
    - Randomly bootstrapped subset of data (minimize overfitting)
    - Randomly chosen feature set (reduce correlation)
  - Each tree will be random and deep, and not pruned
  - And average their predictions



Random Forest Advantages

- Each random tree might have high variance
- When averaged, the combined variance is reduced

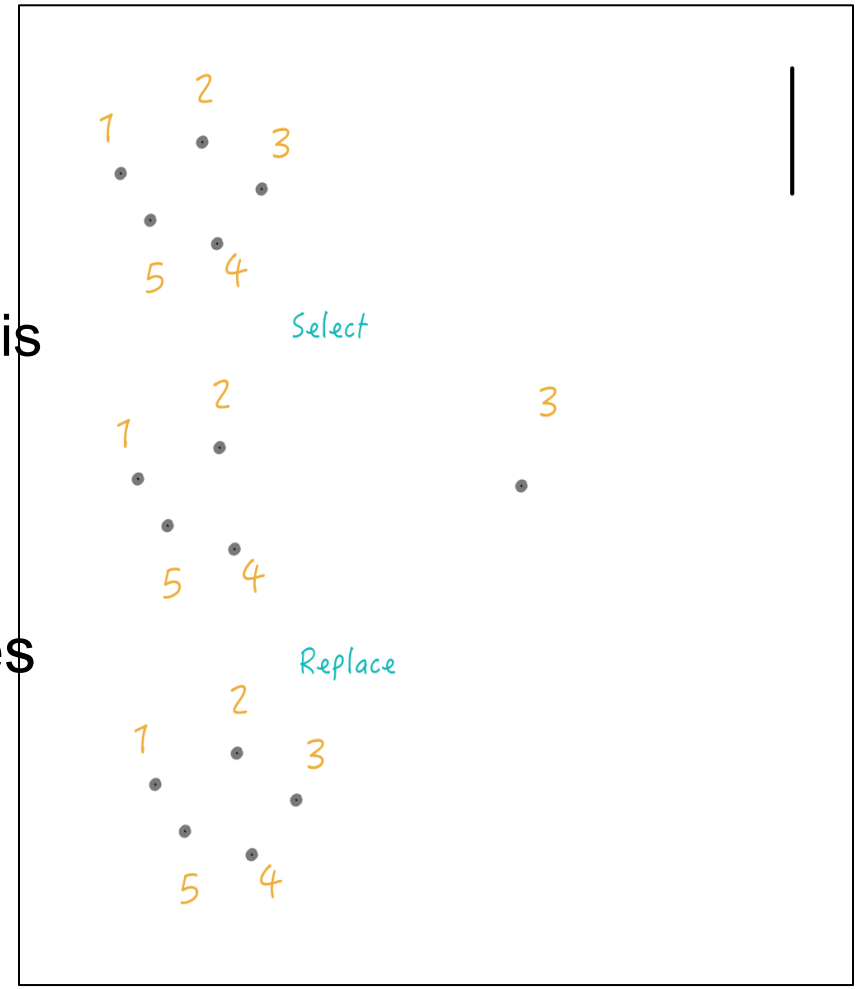
  `Variance of a forest = Variance of a tree / Number of trees`
- Bagging results in the improvement of accuracy

- Aggregating regression trees
  - Average out results
  - Tree1 predicts 10, Tree2 predicts 15, Tree3 predicts 12
  - Final result = average = (10 + 15 + 12) / 3 = 12.33

- Aggregating classification trees
  - Select majority-vote
  - Tree predictions : blue, blue, blue, red, red
  - Final result : blue

Bagging with Replacement

- We select a point at random
  - The probability of a point NOT being selected is 4/5
- Return point 3
- Select again
  - The probability of missing again is $(4/5)^2$
- …
- Select again
- The probability of missing 5 times is
  - $(4/5)^5 = 0.32768$
  - Approximately 1/3
- The rule
  - 1/3 of all points never gets selected



"Out of Bag" Observations

- The rule above:
  - 33% of all points never gets selected in bagging
  - And thus, it is never used for training
- These 33% of measurements are "Out of Bag" observations
  - "OOB" can be used for verification!
- For each OOB observation
  - Do prediction with the forest
  - Compute Mean Square Error (MSE)
- It is called OOB MSE measure

Dataset: Presidential Elections Contributions

CMTE_ID	COMMITTEE ID
CAND_ID	CANDIDATE ID
CAND_NM	CANDIDATE NAME
CONTBR_NM	CONTRIBUTOR NAME
CONTBR_CITY	CONTRIBUTOR CITY
CONTBR_ST	CONTRIBUTOR STATE
CONTBR_ZIP	CONTRIBUTOR ZIP CODE
CONTBR_EMPLOYER	CONTRIBUTOR EMPLOYER
CONTBR_OCCUPATION	CONTRIBUTOR OCCUPATION
CONTB_RECEIPT_AMT	CONTRIBUTION RECEIPT AMOUNT
CONTB_RECEIPT_DT	CONTRIBUTION RECEIPT DATE
RECEIPT_DESC	RECEIPT DESCRIPTION
MEMO_CD	MEMO CODE
MEMO_TEXT	MEMO TEXT
FORM_TP	FORM TYPE
FILE_NUM	FILE NUMBER
TRAN_ID	TRANSACTION ID
ELECTION_TP	ELECTION TYPE/PRIMARY GENERAL INDICATOR

2012 Presidential Election Campaign Contribution Dataset

- **Question for the class**:
  What can we predict?

CMTE_ID	CAND_ID	CAND_NM	CONTBR_NM	CONTBR_CITY	CONTBR_ST	CONTBR_ZIP	CONTBR_EMPLOYER	CONTBR_OCCUPATION	CONTB_RECEIPT_AMT	CONTB_RECEIPT_DT	RECEIPT_DESC	MEMO_CD	MEMO_TEXT	FORM_TP	FILE_NUM	TRAN_ID
C00431445	P80003338	Obama, Barack	COHEN, AMY	PALM SPRINGS	CA	92264	NOT EMPLOYED	DISABLED	56	12-Oct-12				SA17A	846396	C25804066
C00431445	P80003338	Obama, Barack	LAMATTINA, LISA	PRINCETON JUNCTION	NJ	85502831	SILVERMEDIA	ACTOR	500	15-May-12		X	* OBAMA VICTORY FUND 2012	SA18	791603	C16489775
C00495820	P80000748	Paul, Ron	MARCELLA, DANIEL	ORCHARD BEACH	MD	212262034	UNITED STATES AIR FORCE	DISABLED VETERAN	201.2	12-Jan-12				SA17A	779231	98519
C00431445	P80003338	Obama, Barack	HOLDEN, LARRY	LINCOLN	MA	17736347	SELF-EMPLOYED	LAW	100	16-Jul-12		X	* OBAMA VICTORY FUND 2012	SA18	806136	C18958762
C00494393	P20002556	Pawlenty, Timothy	DEVRIES, TIMOTHY C. MR.	LONG LAKE	MN	553569732	NORWEST EQUITY PARTNERS	PRIVATE EQUITY	2500	24-May-11	REDESIGNATION FROM PRIMARY	X	REDESIGNATION FROM PRIMARY	SA17A	748365	SA17.25570

What Can We Predict?

- Here are the possible predication candidates
  - Occupations
  - Employer
  - State
  - Candidate
- Here is what we predict in our lab
  - Contribution amount

Lab: Random Forest

- **Overview**

- **Approximate Time**
  30 mins

- **Instructions**
  Follow appropriate Python, R, or Spark instructions

Review Questions

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Backup Slides

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Example of a Decision Tree