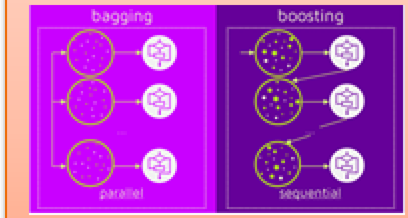
Focus for this lecture

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- Bagging and Boosting
- Other Ensemble Techniques
- Random Forests

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model



Bagging and Boosting

Ensemble Methods

Bagging and Boosting

Boosting and Adaboost

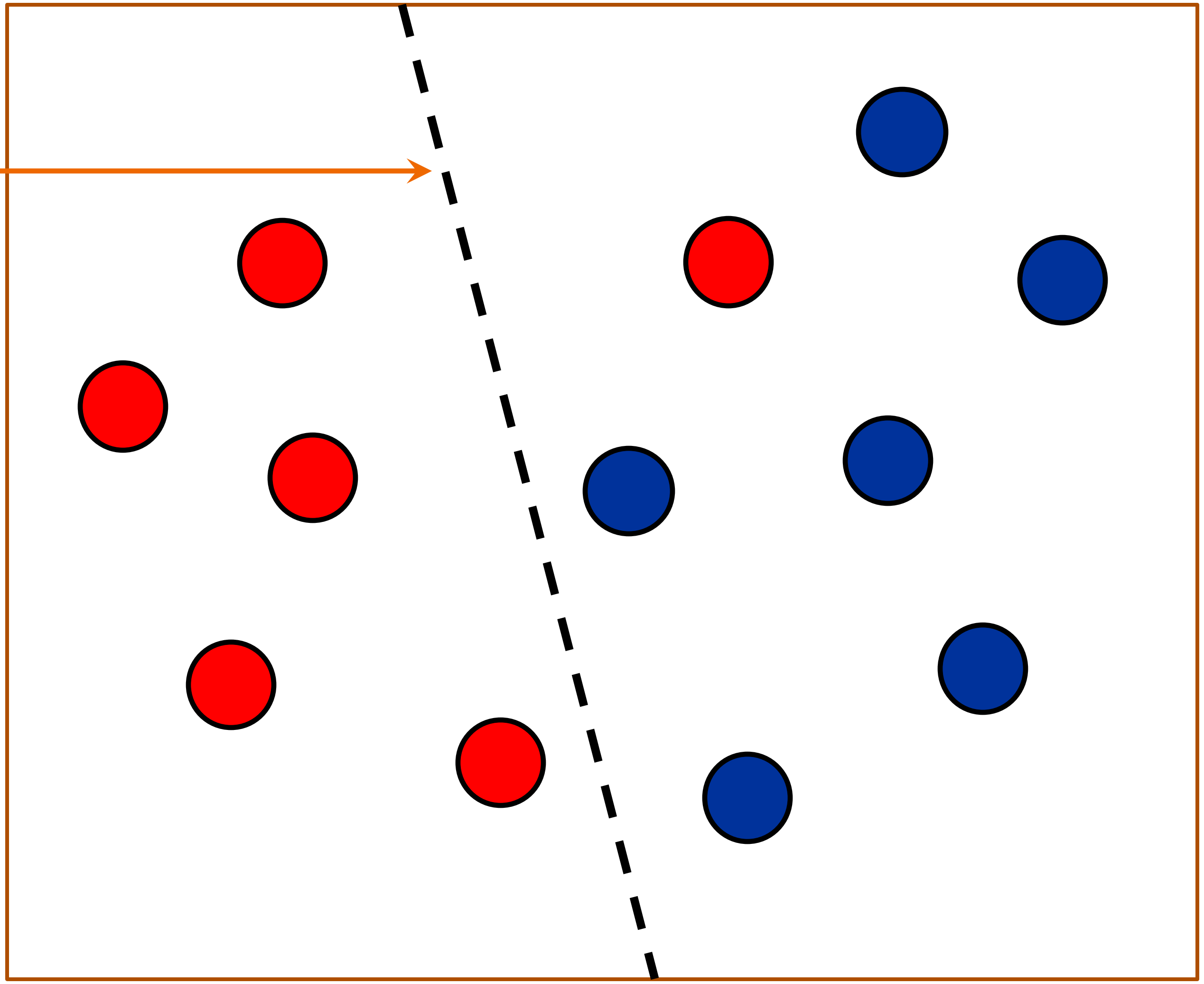
- Generate a set of weak classifiers
- Combine them using a weighted combination (probabilities)
- Weights proportional to their performance on the validation set
- **AdaBoost:**
  - A Popular variant of boosting
  - Generate classifiers by weighted sampling

Boosting Illustration

Weak Classifier 1

$$h_1(x) = 0$$

$$H(x)\colon\ sign(h_1(x))$$

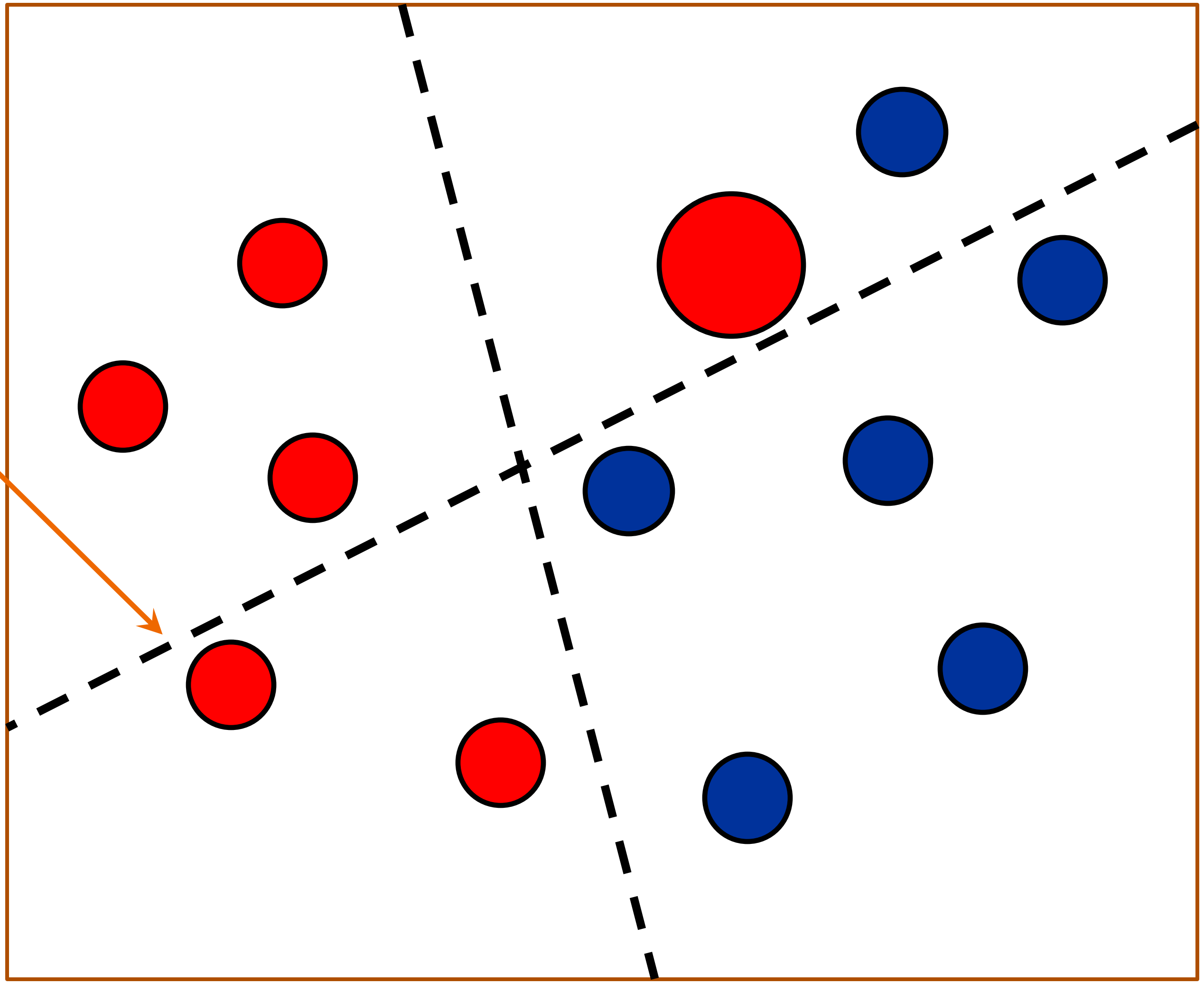


Boosting Illustration

Weak Classifier 2

$$h_2(x) = 0$$

$$H(x)\text{: } sign(h_2(x))$$

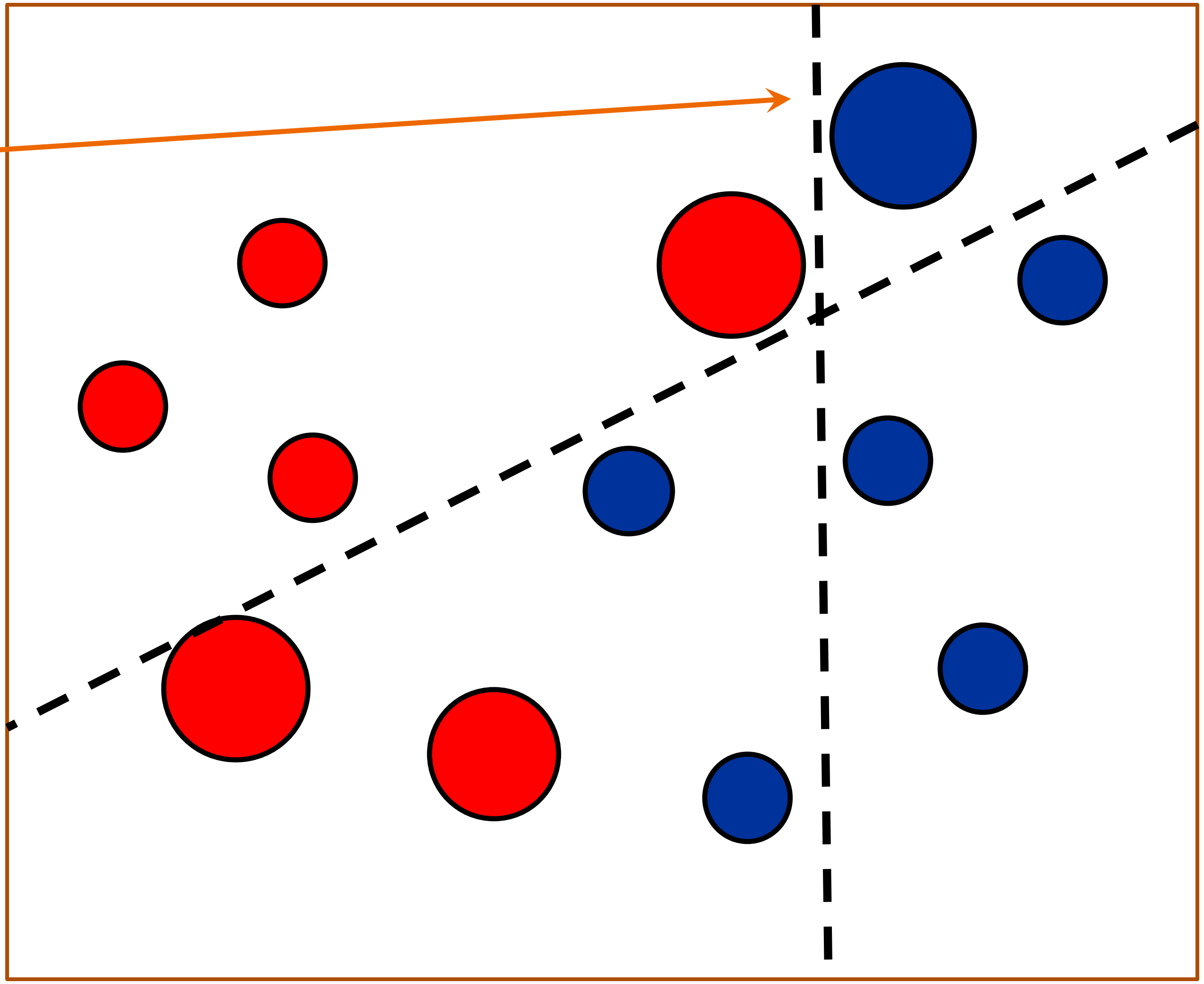


Boosting Illustration

Weak Classifier 3

$$h_3(x) = 0$$

$$H(x)\text{: } sign(h_3(x))$$

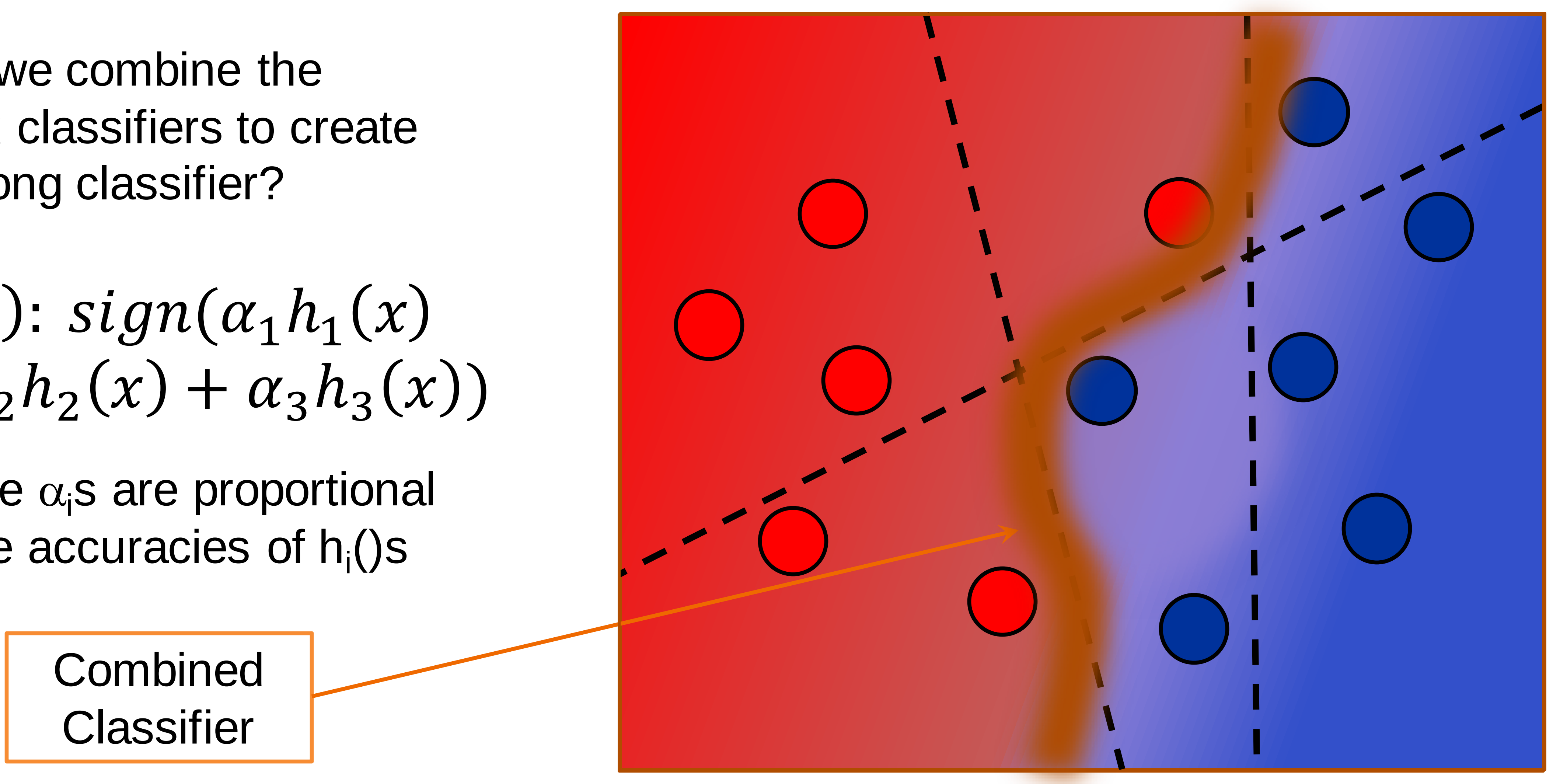


Boosting Illustration

Can we combine the weak classifiers to create a strong classifier?

$$H(x):\ sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

where α_is are proportional to the accuracies of $h_i()$s

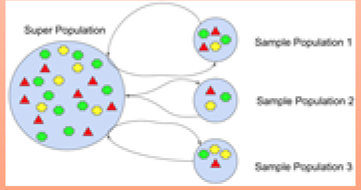


Boosting: Summary

- Weak Learners combine to form strong classifiers
- Does not work that well with strong learners
- Boosting combined with trees gives some of the most powerful classifiers

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- ✓ Bagging and Boosting
- Other Ensemble Techniques



Other Ensemble Methods

Other solutions from a group of solvers

Ensemble Clustering

- Generate a set of weak clusters
  - Could be done efficiently
  - Need not be too accurate on number of clusters

- Combine the clusters together
  - How to generate consensus ?
  - Can give effective ways to determine number of clusters

Generation of Ensemble by Weak Clustering

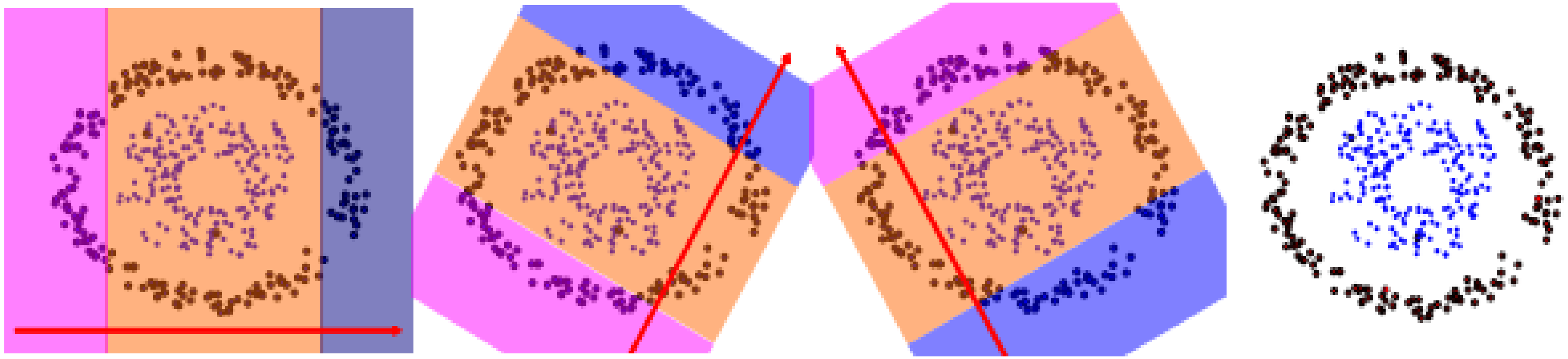
- Weak clustering is defined as a partition that is only slightly better than a random partition of the data
- Weak clusterings can be generated efficiently compared to sophisticated clustering algorithms
- Weak clusterings can be obtained by
  - clustering in low dimensional projections of data
  - by random “cuts” of the data,
  - using sub-samples of data

Ensemble Generation by Random Projections

- Random subspaces provide us with different views of the multidimensional data; each random subspace can be of very low dimensionality (e.g., 1-D)

- Clustering in 1-D space is computationally inexpensive and can be implemented by k-means algorithm

Ensemble Generation by Random Projections



Different 3-cluster partitions of 2-dim data
resulting from projections onto random lines

Concentric circular
clusters can be perfectly
detected by an ensemble
of 50-100 partitions

Co-association As Consensus Function

- Similarity between objects can be estimated by the number of clusters shared by two objects in all the partitions of an ensemble

- This similarity definition expresses the strength of co-association of n objects by an n x n matrix

Co-association As Consensus Function

$$C_{ij} = C\left(x_i, x_j\right) = \frac{1}{N}\sum_{k=1}^{N} I\left(\pi_k(x_i) = \pi_k(x_j)\right)$$

- x_i : the i-th pattern; $\pi_k\left(x_i\right)$: cluster label of x_i in the k-th partition; $I()$: Indicator function; N = no. of different partitions

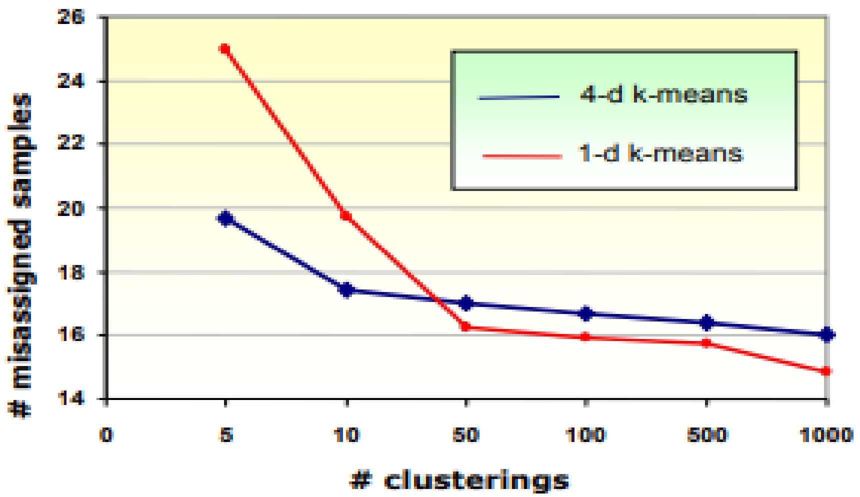
- This consensus function eliminates the need for solving the label correspondence problem

Results for Ensembles of Random Projections

"Galaxy/Star" data (4600 points in 14 dimensions, 2 classes)

H, # of components	k, # of cl. in component	Hypergraph methods: HGPA	Hypergraph methods: MCLA	Median partition, QMI: k-means
5	2	49.7	20.0	20.4
10	2	49.7	23.5	21.1
20	2	49.7	21.0	18.0
5	3	49.7	22.0	21.7
10	3	49.7	17.7	13.7
20	3	49.7	15.8	13.3
5	4	49.7	19.7	16.7
10	4	49.7	16.9	15.5
20	4	49.7	14.1	13.2
5	5	49.7	22.0	22.5
10	5	49.7	17.7	17.4
20	5	49.6	15.2	12.9

Type of Consensus Function



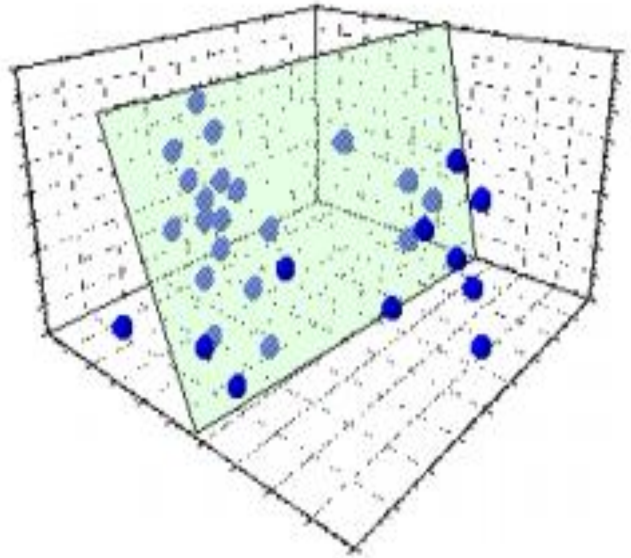
Ensemble finds novel and better clustering solutions compare with regular k-means that has more than 30% error rate, on average, for Galaxy data

Partitions by Random Cuts

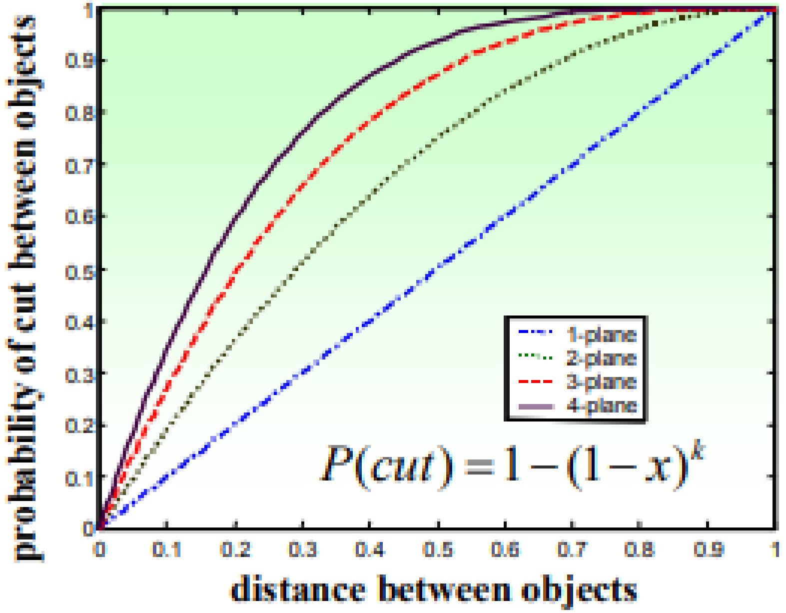
- This approach pushes the notion of the weak clustering to the extreme.

- Data set is cut by random hyperplanes. Points separated by hyperplanes are declared to be in different clusters

Partitions by Random Cuts



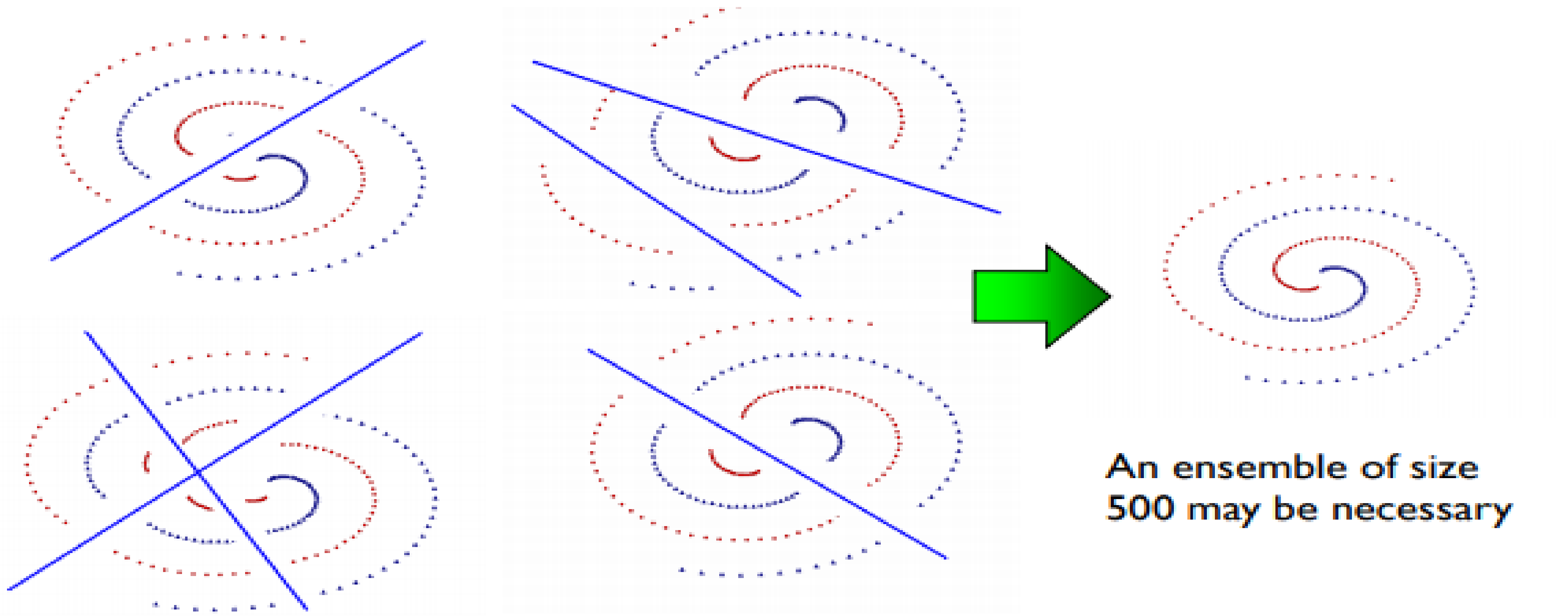
Even random cuts can uncover inter-pattern similarity values and provide relevant cluster information



Probability of separating two patterns using different numbers of random cuts

Results for Ensemble of Random Cuts

We can correctly identify two spirals by combining partitions resulting from random cuts using co-association consensus function with SL



An ensemble of size 500 may be necessary

Ensemble Methods: Summary

- A group of weak learners do better than an expert

- Generate a set of solutions
  – Ensure the solutions are weak
    - Linear models, simple decision trees, projecting on a line, etc.

  – Ensure the solutions are diverse
    - Could be done through data sampling, feature dropping, etc.

Ensemble Methods: Summary

- Combine the solutions
  - Intelligent combination that weighs solutions by their performance
- Effective approach in a variety of Machine Learning problems
- Ensemble methods are often highly parallelizable
- Using an ensemble of simple solutions tend to reduce overfitting

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- ✓ Bagging and Boosting
- ✓ Other Ensemble Techniques
- Random Forests

Random Forests

A simple way of combining Decision Trees

Random Forests

- Ensemble classifier
- Consists of many decision trees
- Selects features by bagging
- Outputs the class that is the mode of the class’s output by individual decision trees

CART Algorithm: A Recap

- Classification and Regression Trees (Leo Breiman)

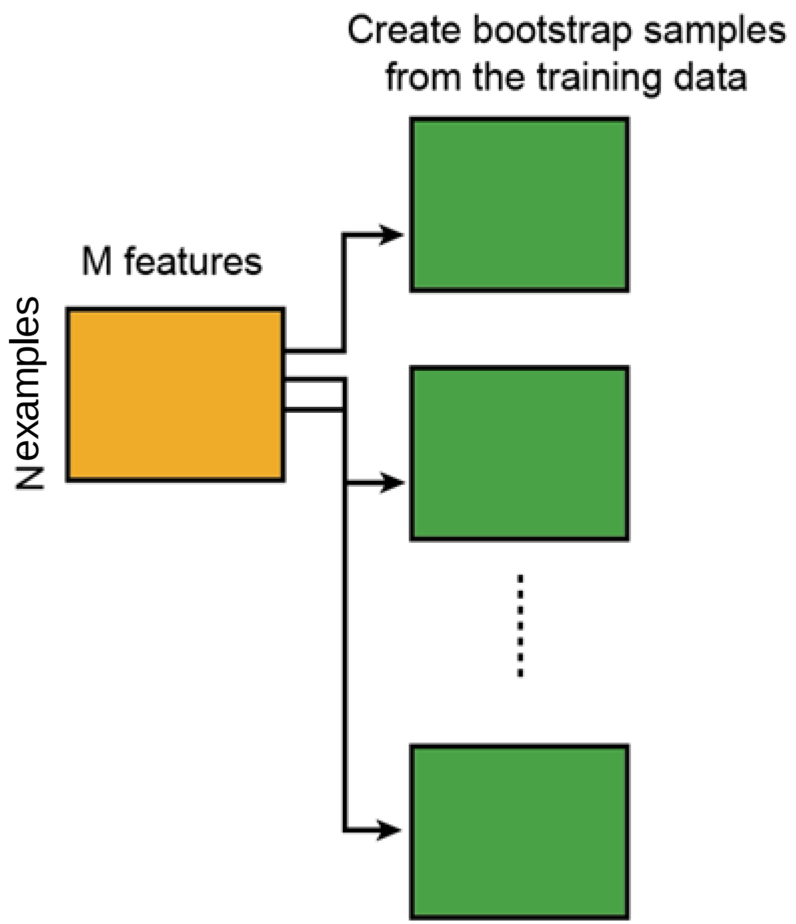
- Recursive Binary Splitting: Greedy Algorithm
  - All values of an attribute are sorted and all split points are tested
  - Test all such attributes and select the split with the lowest cost

- Cost Functions:
  - Regression: MSE
  - Classification: Gini

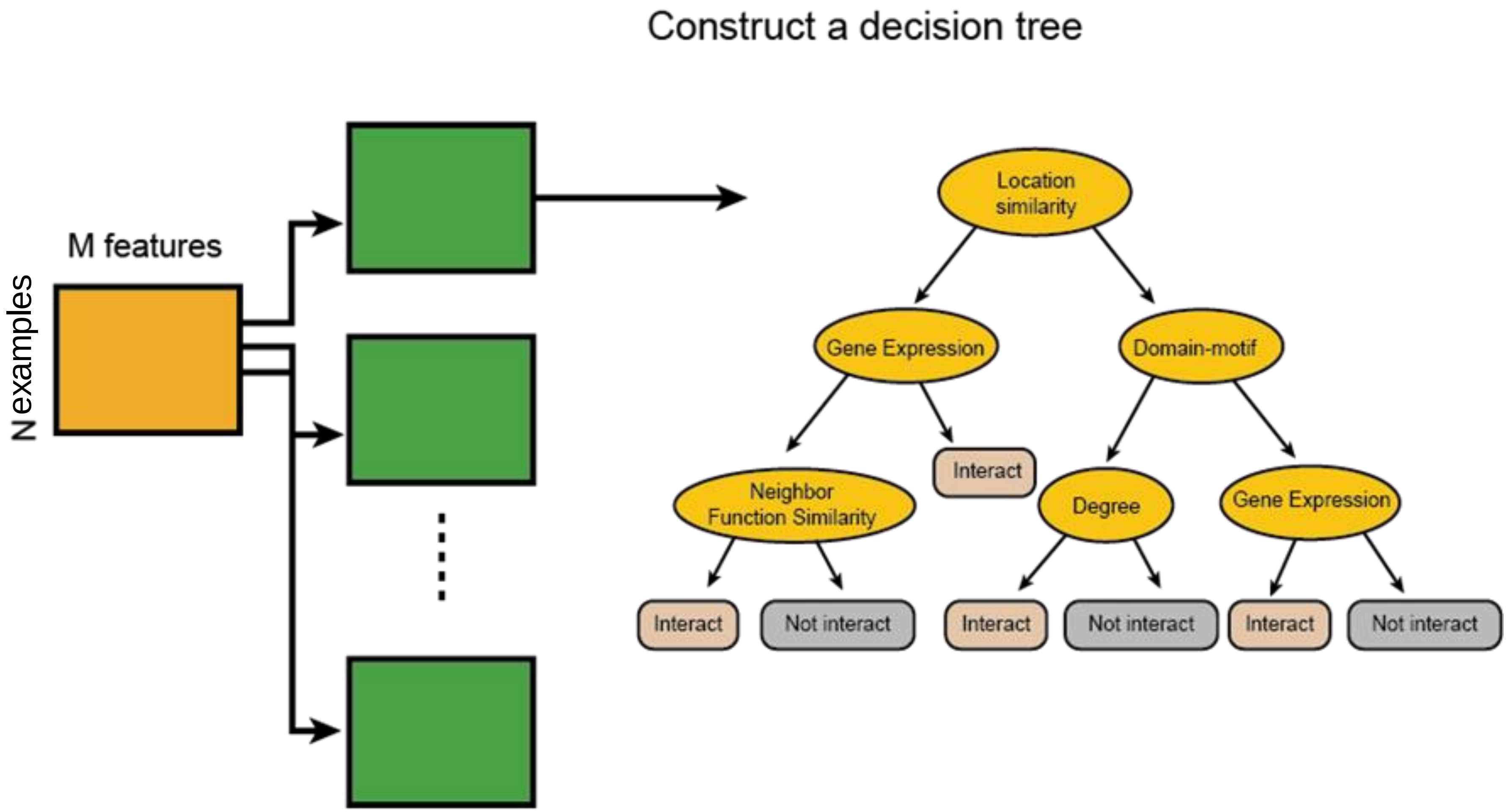
Random Forest

- **Random forest** (or **random forests**) is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- The term came from **random decision forests** that was first proposed by Tin Kam Ho of Bell Labs in 1995.
- The method combines Breiman's "bagging" idea and the random selection of features.

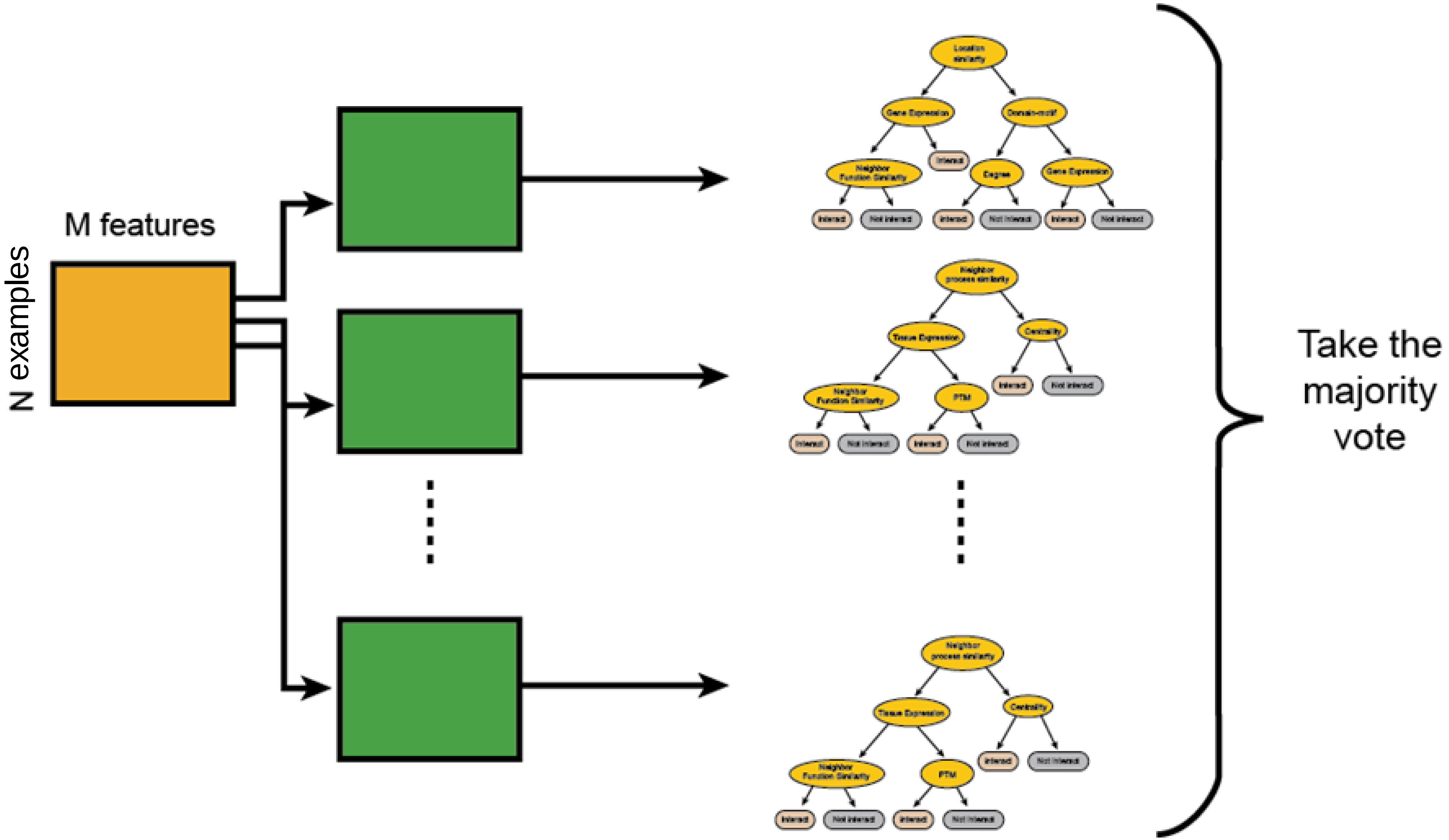
Random Forest



Random Forest

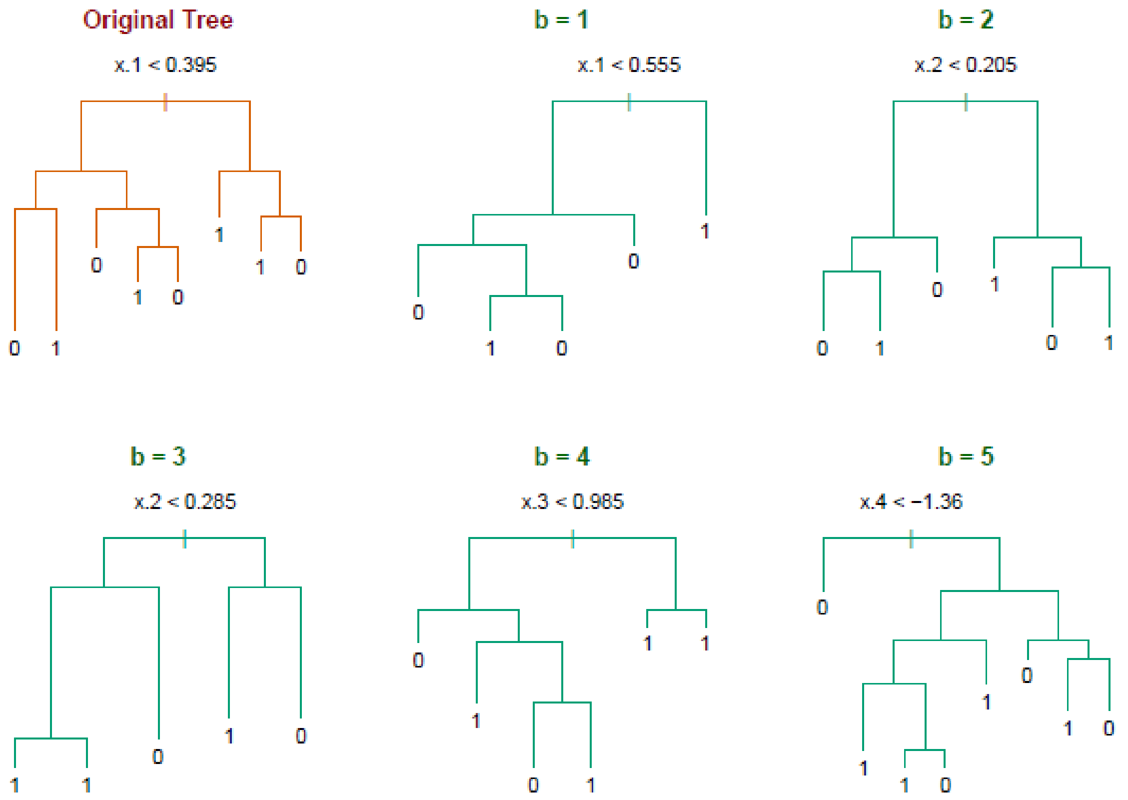


Random Forest



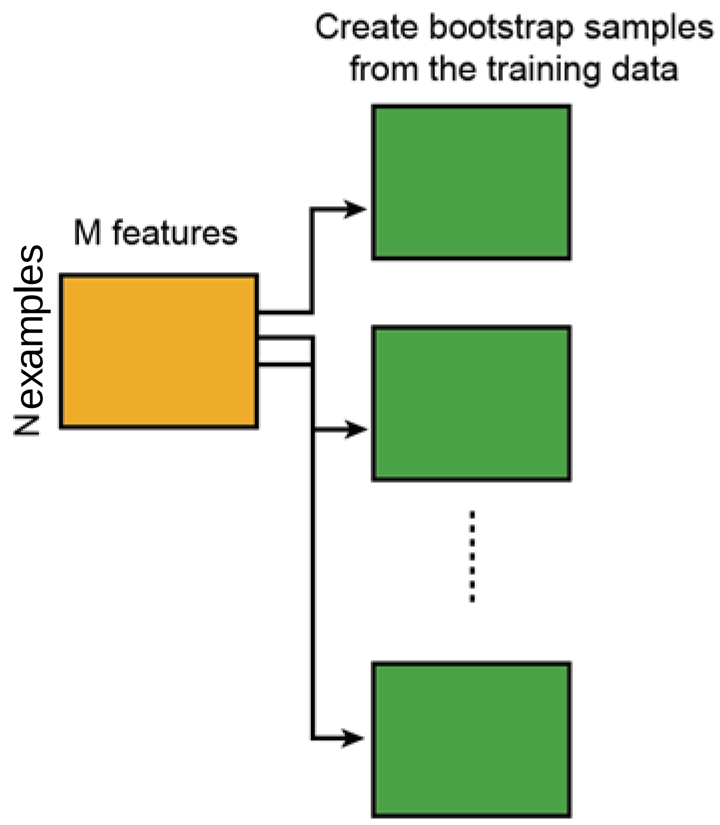
Random Forest

- Random forest classifier, an extension to bagging which uses *de-correlated* trees.

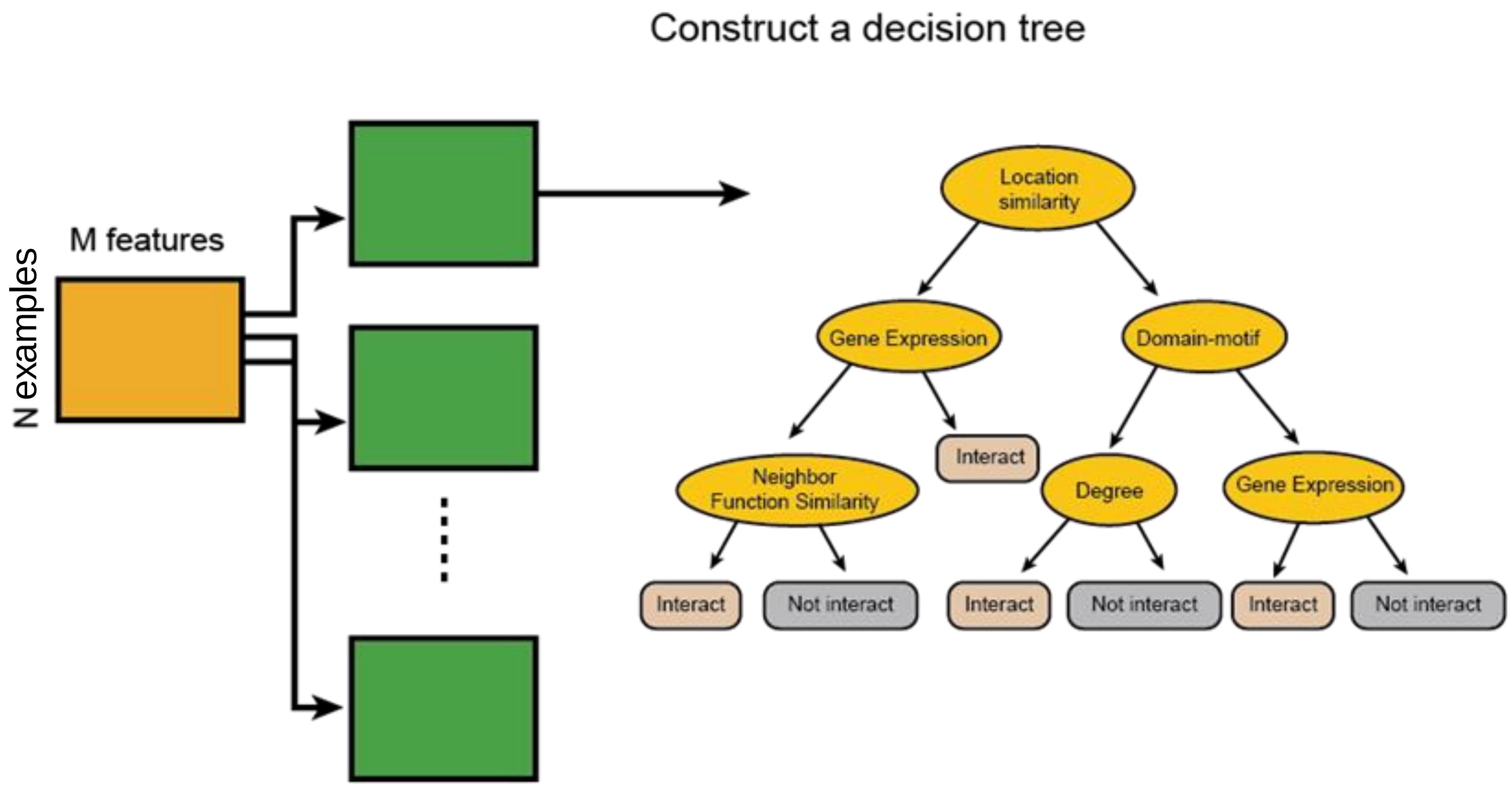


Random Forest Classifier

Random Forest Classifier

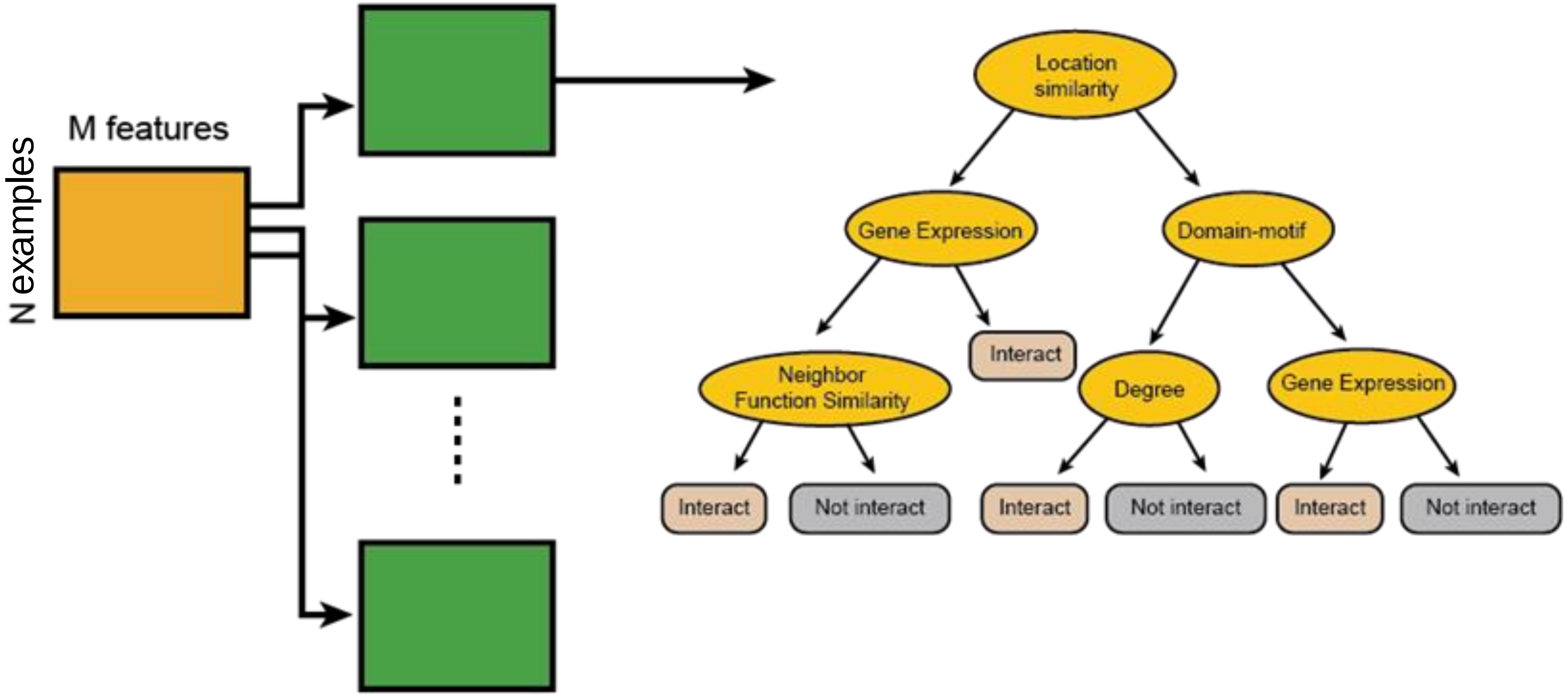


Random Forest Classifier

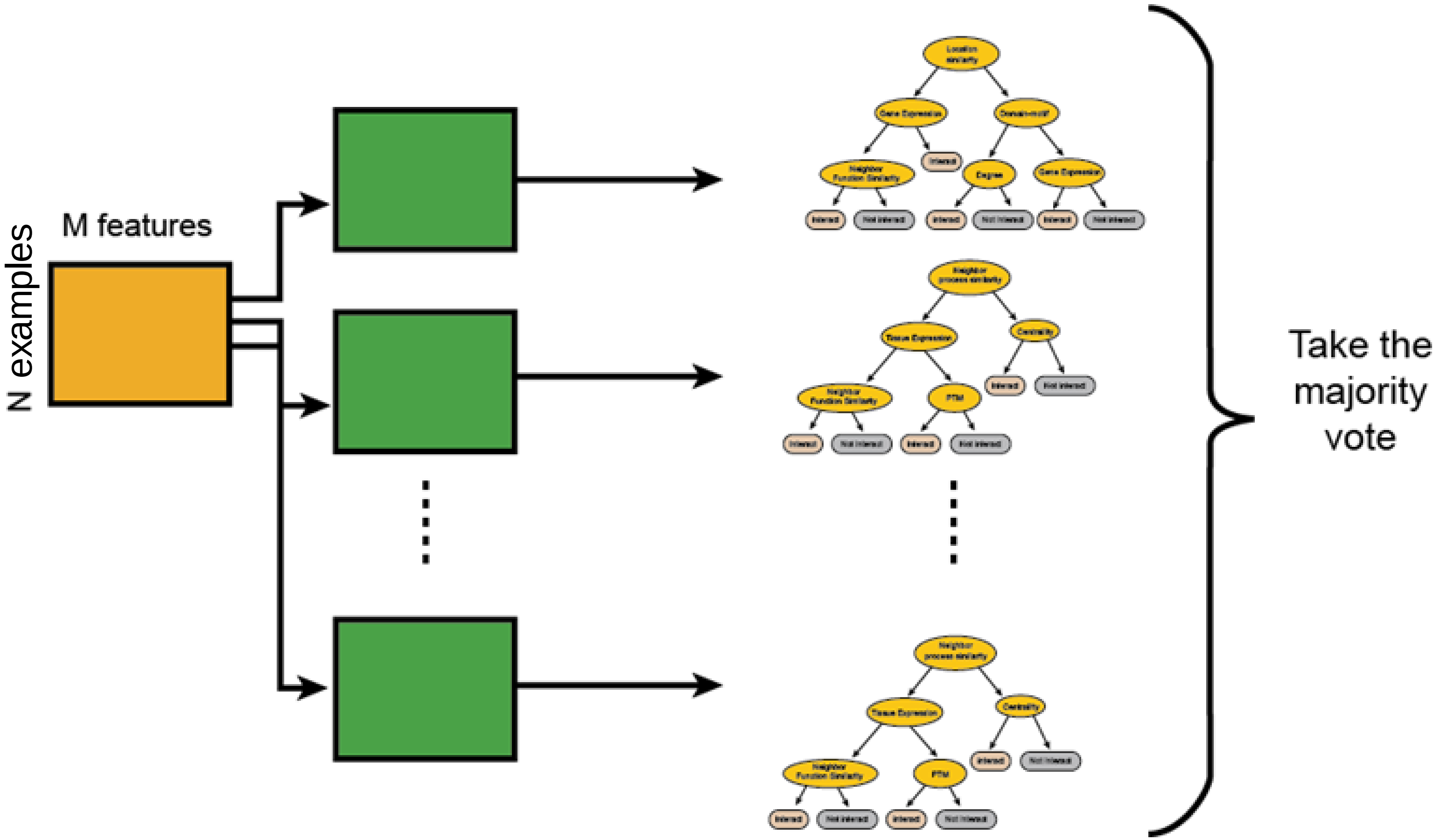


Random Forest Classifier

At each node in choosing the split feature choose only among m<M features



Random Forest Classifier



Example: Email Classification: Spam vs Ham

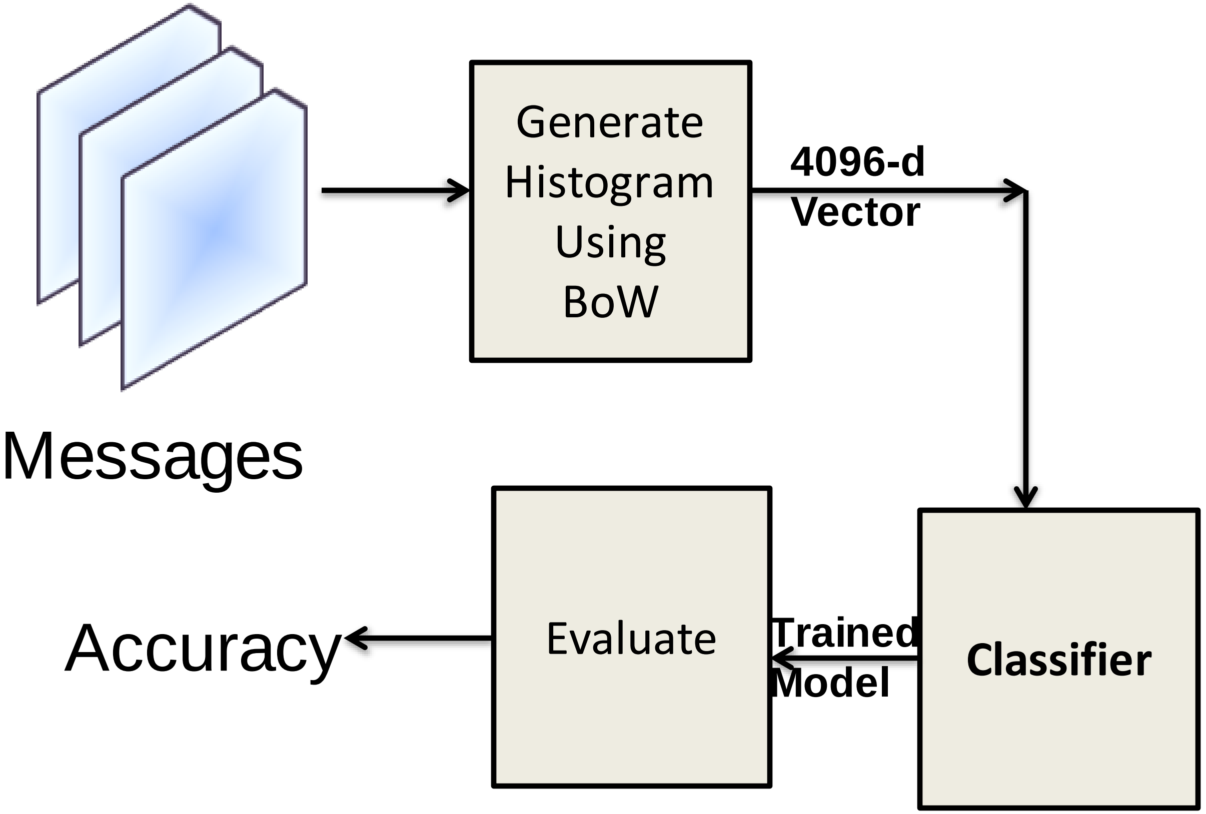
- Features: 2-million dimensional (one-hot)
- Each tree selects a subset of features (words)
  - Select the best from the subset
- Should the words be uniformly sampled?
- Automatically selects relevant words
- What is the equivalent in classification with Gene data.

Casestudy

Code Available

Case Study: RF & DT

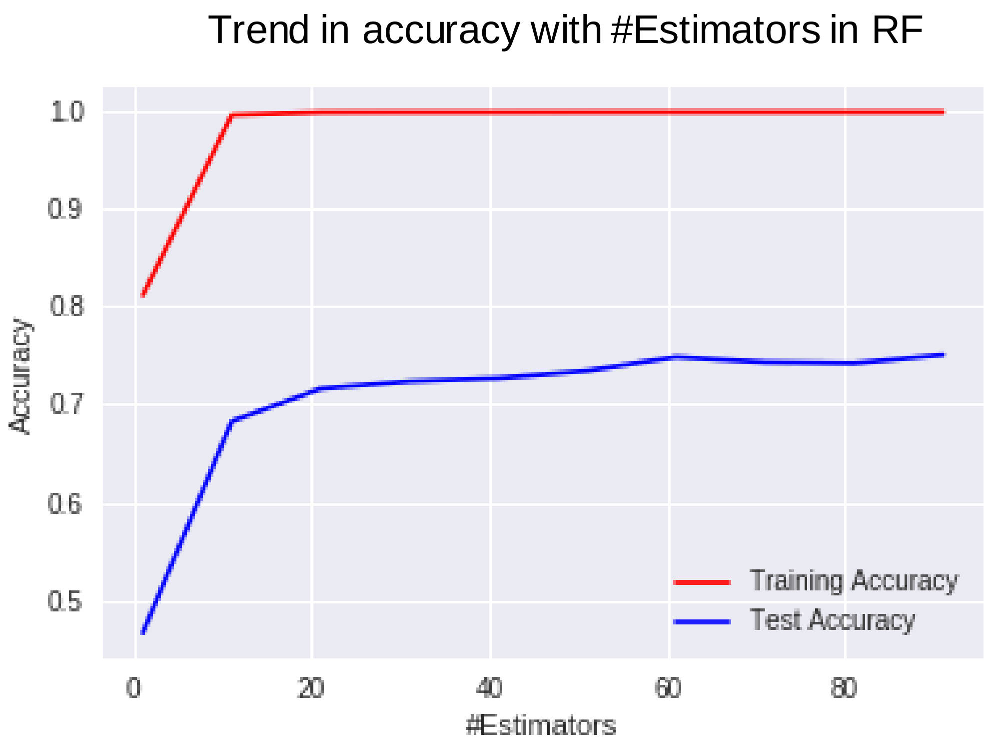
- Text data with 20 classes
- Preprocessing:
  - – Find the Histograms for Each Document using Bag of words
- Train the classifier on the reduced data
- Find the Accuracy to Evaluate the model



Code Available

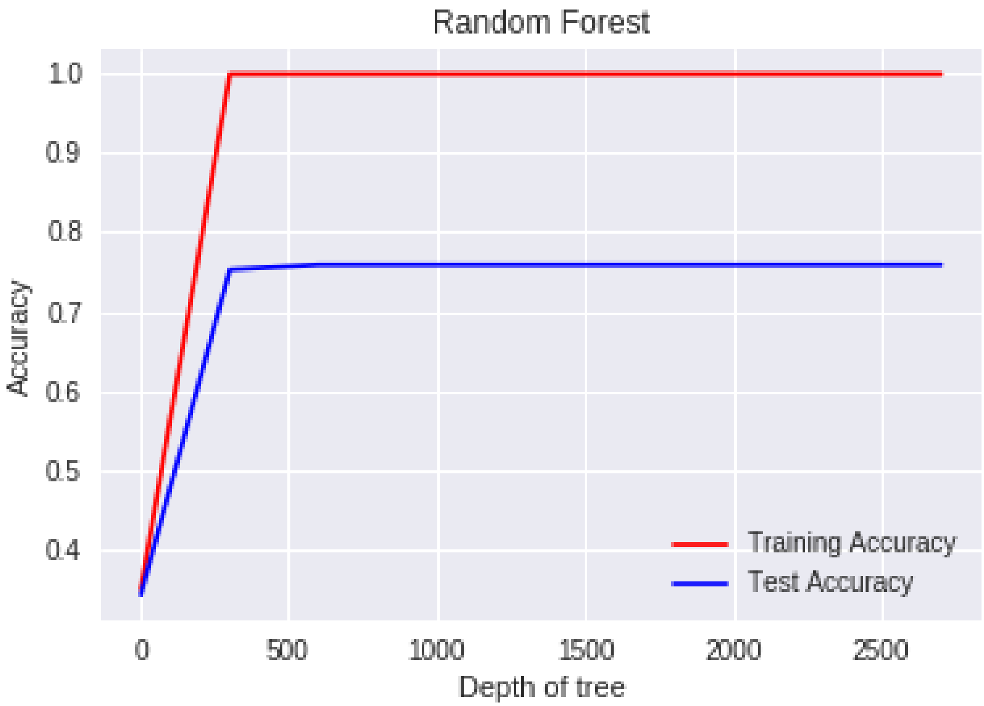
Random Forest vs Decision Tree

- DT:
  - Training Accuracy: 99.8
  - Test Accuracy: 58.4
  - Training time : 1min 51s
  - Testing time: 14.1 ms
- RF (Estimators=100):
- Training Accuracy: 99.8
- Test Accuracy: 75.9
- Training time : 2min 9s
- Testing time: 151 ms



Code Available

Effect of Depth



Summary

Acquire Raw Data → Prepare / Clean Data, Visualization → Feature Engineering → Pick Model and Hyper-params for Task → Model Training / Optimization → Evaluate Model Performance → Deploy Model

- ✓ Bagging and Boosting
- ✓ Other Ensemble Techniques
- ✓ Random Forests

Thanks!!

Questions?