

# Machine Learning Using Python

*Tan Kwan Chong*

*Senior Data Scientist, SparkBeyond*

---

# Workshop Objectives

---

- Collect data from a variety of sources
- Explore large data sets
- Clean and "munge" the data to prepare it for analysis
- Apply machine learning algorithms to gain insight from the data
- Visualize the results of your analysis
- Build your own library and Python scripts

---

# Day 1 - Developing the Fundamentals

---

- Introduction to Machine Learning (AM)
  - What is machine learning?
  - Installation and update of tools
- Exploring and using Data Sets (PM)
  - Learn the steps to pre-process a data set and prepare it for machine learning algorithms
  - Introduction to machine learning algorithms – linear & logistic regression

---

# Day 2 - Diving into Machine Learning

---

- Supervised vs Unsupervised Learning (AM)
  - Decision trees, random forests, K-nearest neighbors
  - K-means, DBScan clustering
- Model Evaluation (PM)
  - Feature engineering and model selection
  - Model evaluation metrics
  - Overfitting and bias-variance trade-off
  - Cross validation

# Diving into Machine Learning

# Machine Learning Predicts World Cup Winner

Intelligent Machines

## Machine learning predicts World Cup winner

Researchers have predicted the outcome after simulating the entire soccer tournament 100,000 times.

by Emerging Technology from the arXiv June 12, 2018

**T**he 2018 soccer World Cup kicks off in Russia on Thursday and is likely to be one of the most widely viewed sporting events in history, more popular even than the Olympics. So the potential winners are of significant interest.

One way to gauge likely outcomes is to look at bookmakers' odds. These companies use professional statisticians to analyze extensive databases of results in a way that quantifies the probability of different outcomes of any possible match. In this way, bookmakers can offer odds on all the games that will kick off in the next few weeks, as well as odds on potential winners.

An even better estimate comes from combining the odds from lots of different bookmakers. This approach suggests Brazil is the clear favorite to win the 2018 World Cup, with a probability of 16.6 percent, followed by Germany (12.8 percent) and Spain (12.5 percent).

But in recent years, researchers have developed machine-learning techniques that have the potential to outperform conventional statistical approaches. What do these new techniques predict as the likely outcome of the 2018 World Cup?



An answer comes from the work of Andreas Groll at the Technical University of Dortmund in Germany and a few colleagues. These guys use a combination of machine learning and conventional statistics, a method called **random-forest approach** to identify a different most likely winner.

First some background. The random-forest technique has emerged in recent years as a powerful way to analyze

large data sets while avoiding some of the pitfalls of other data-mining methods. It is based on the idea that some future event can be determined by a decision tree in which an outcome is calculated at each branch by reference to a set of training data.

However, decision trees suffer from a well-known problem. In the latter stages of the branching process, decisions can become severely distorted by training data that is sparse and prone to huge variation at this kind of resolution, a problem known as overfitting.

The random-forest approach is different. Instead of calculating the outcome at every branch, the process calculates the outcome of random branches. And it does this many times, each time with a different set of randomly selected branches. The final result is the average of all these randomly constructed decision trees.


<https://www.technologyreview.com/s/611397/machine-learning-predicts-world-cup-winner/>

# Error, Bias, Variance

---

# What is R-Squared? What is a Residual?

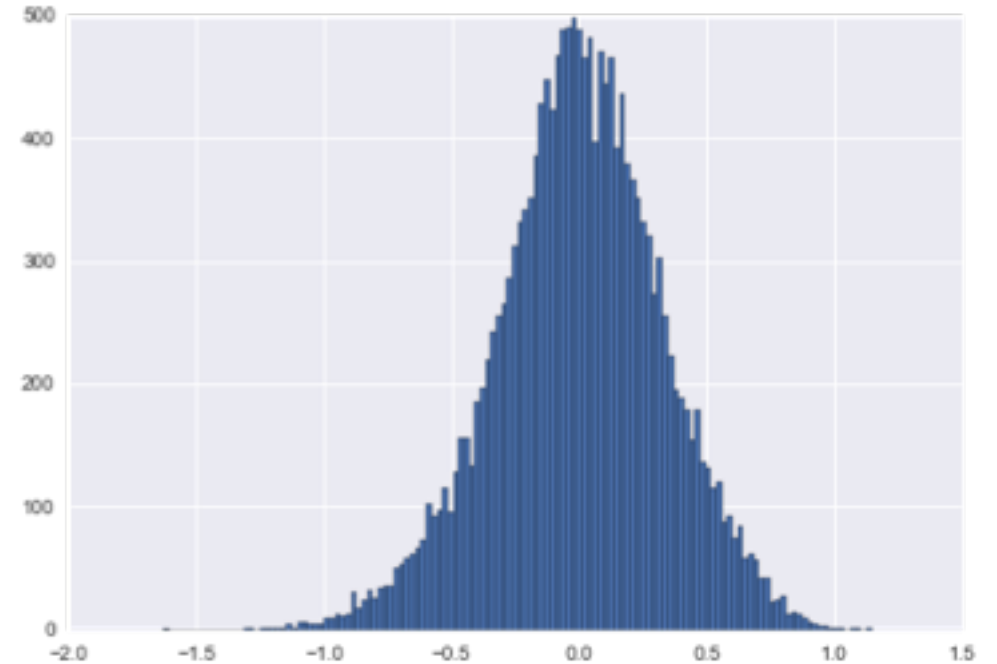
---

- R-squared, the central metric introduced for linear regression
- Which model performed better, one with an r-squared of 0.79 or 0.81?
- R-squared measures explain variance.
- But does it tell the magnitude or scale of error? 
- We'll explore loss functions and find ways to refine our model.



# Recall: What's Residual Error?

- In linear models, residual error must be normal with a median close to zero.
- Individual residuals are useful to see the error of specific points, but it doesn't provide an overall picture for optimization.
- We need a metric to summarize the error in our model into one value.



---

# R-Squared

---

- Recall that R-squared or fraction of variance explained is

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

- R-squared has very intuitive properties. When a model does no better than a null model then R-squared will be 0.
- When a model makes perfect predictions, R-squared will be 1.
- Commonly R-squared is only applied as a measure of training error and it is possible to overfit a model with a large number of parameters that are pure noise

---

# Adjusted R-Squared

---

- Adjusted R-squared adjusts for this problem by penalizing additional complexity

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p}$$

*where  $n$  is the number of observations and  $p$  the number of parameters*

- Thus the error predicted by Adjusted R-squared will start to increase as model complexity becomes very high
- However, Adjusted R-squared generally under-penalizes complexity
- R-squared values also don't tell us how far off our predictions are

---

# Mean Absolute Error (MAE)

---

- Mean Absolute Error is the mean of the absolute value of errors
- Calculate the difference between each target  $y$  and the model's predicted value  $\hat{y}$  (i.e. the residual)
- Apply the absolute function on the residual and take the mean of the resulting absolute error values
- MAE is easy to understand because it is the average error

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

---

# Mean Squared Error (MSE)

---

- Mean Squared Error is the mean of the squared errors
- Calculate the difference between each target  $y$  and the model's predicted value  $\hat{y}$  (i.e. the residual)
- Square each residual and take the mean of the squared errors
- MSE is more popular than MAE because MSE punishes large errors, which tends to be useful in the real world

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

---

# Root Mean Squared Error (RMSE)

---

- Root Mean Squared Error is the square root of the mean of the squared errors
- Calculate the MSE and take the square root of it
- RMSE is even more popular than MSE because RMSE is interpretable in the “y” units

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

---

# Advanced Classification Metrics

---

- We can split up the accuracy of each label by using the true positive rate and the false positive rate.
- For each label, we can put it into the category of a true positive, false positive, true negative, or false negative

|                    | Prediction Positive | Prediction Negative |
|--------------------|---------------------|---------------------|
| Condition Positive | True Positive (TP)  | False Negative (FN) |
| Condition Negative | False Positive (FP) | True Negative (TN)  |

# Advanced Classification Metrics

- True Positive Rate (TPR) asks, “Out of all of the target class labels, how many were accurately predicted to belong to that class?”
- For example, given a medical exam that tests for cancer, how often does it correctly identify patients with cancer?

|                    | Prediction Positive | Prediction Negative |
|--------------------|---------------------|---------------------|
| Condition Positive | True Positive (TP)  | False Negative (FN) |
| Condition Negative | False Positive (FP) | True Negative (TN)  |

True Positive Rate aka Sensitivity, Recall

$$TPR = \frac{TP}{P} = \frac{TP}{(TP + FN)}$$



# Advanced Classification Metrics

- False Positive Rate (FPR) asks, “Out of all items not belonging to a class label, how many were predicted as belonging to that target class label?”
- For example, given a medical exam that tests for cancer, how often does it trigger a “false alarm” by incorrectly saying a patient has cancer?

|                    | Prediction Positive | Prediction Negative |
|--------------------|---------------------|---------------------|
| Condition Positive | True Positive (TP)  | False Negative (FN) |
| Condition Negative | False Positive (FP) | True Negative (TN)  |

False Positive Rate aka Specificity

$$FPR = \frac{FP}{N} = \frac{FP}{(FP + TN)}$$

# Advanced Classification Metrics

- Precision reflects how many of the positive predictions are indeed positive

|                    | Prediction Positive | Prediction Negative |
|--------------------|---------------------|---------------------|
| Condition Positive | True Positive (TP)  | False Negative (FN) |
| Condition Negative | False Positive (FP) | True Negative (TN)  |

Positive Prediction Value aka Precision

$$PPV = \frac{TP}{(TP + FP)}$$

|                    | Prediction Positive | Prediction Negative |
|--------------------|---------------------|---------------------|
| Condition Positive | True Positive (TP)  | False Negative (FN) |
| Condition Negative | False Positive (FP) | True Negative (TN)  |

Negative Prediction Value

$$NPV = \frac{TN}{(TN + FN)}$$

---

# Advanced Classification Metrics

---

- A good classifier would have a true positive rate approaching 1 and a false positive rate approaching 0
- We can vary the classification threshold for our model to get different predictions. But how do we know if a model is better overall than another model?
- We can compare the FPR and TPR of the models, but it can often be difficult to optimize two numbers at once
- Logically we would like a single number for optimization

---

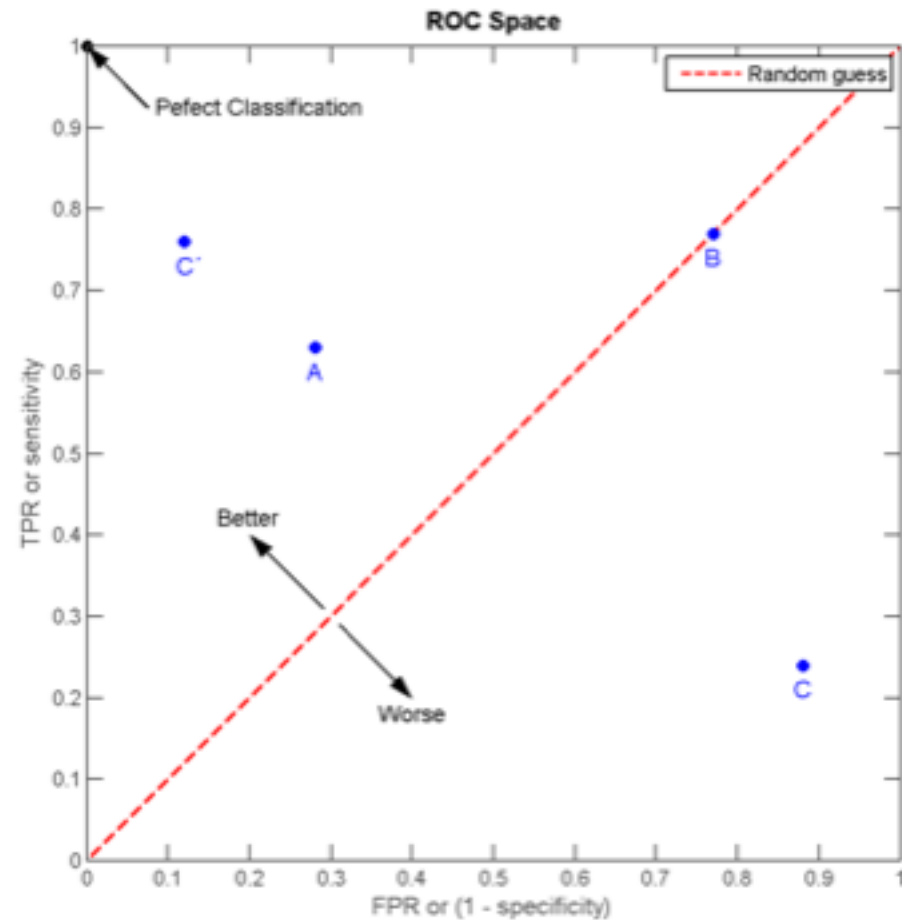
# Advanced Classification Metrics

---

- This is where the Receiver Operation Characteristic (ROC) curve comes in handy.
- The curve is created by plotting the true positive rate against the false positive rate at various model threshold settings.
- Area Under the Curve (AUC) summarizes the impact of TPR and FPR in one single value.

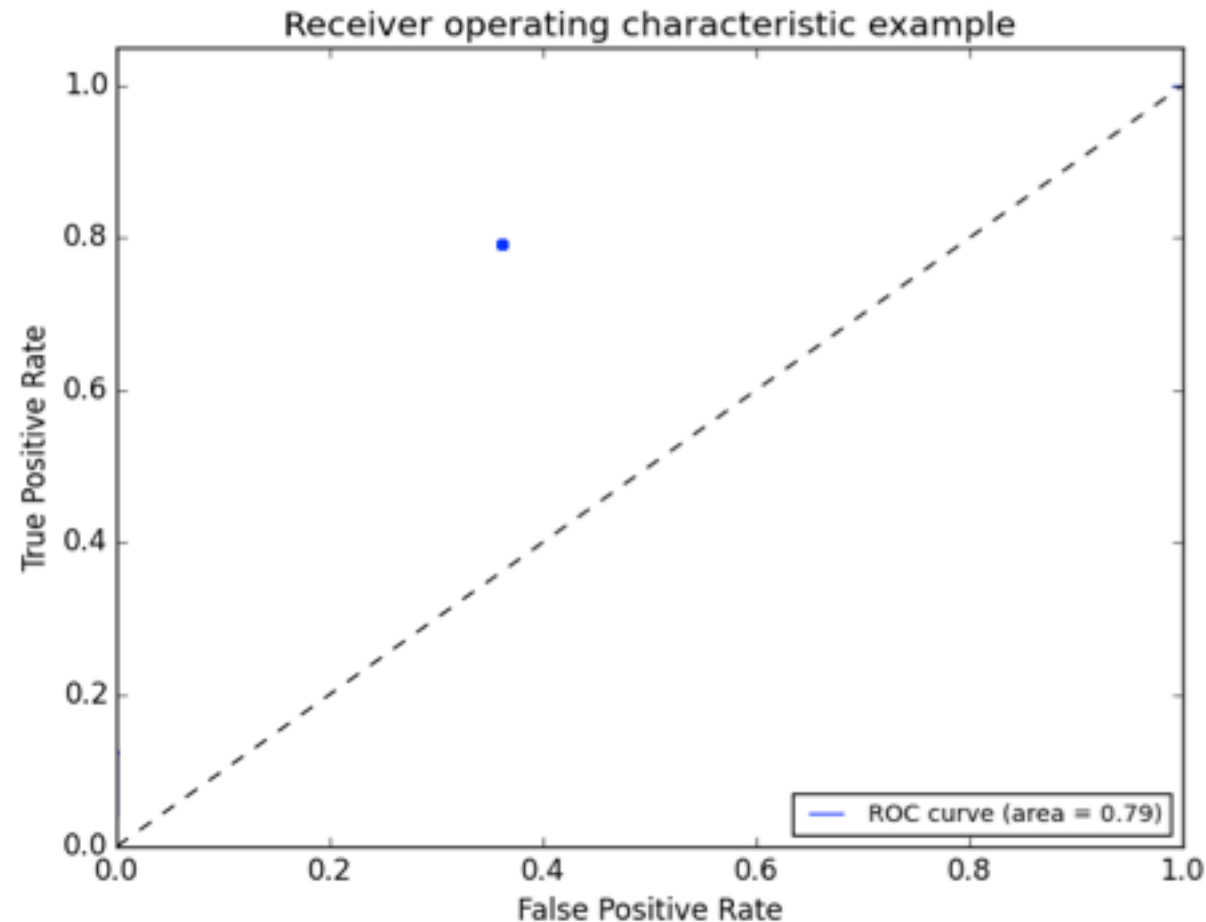
# Advanced Classification Metrics

- There can be a variety of points on a ROC curve



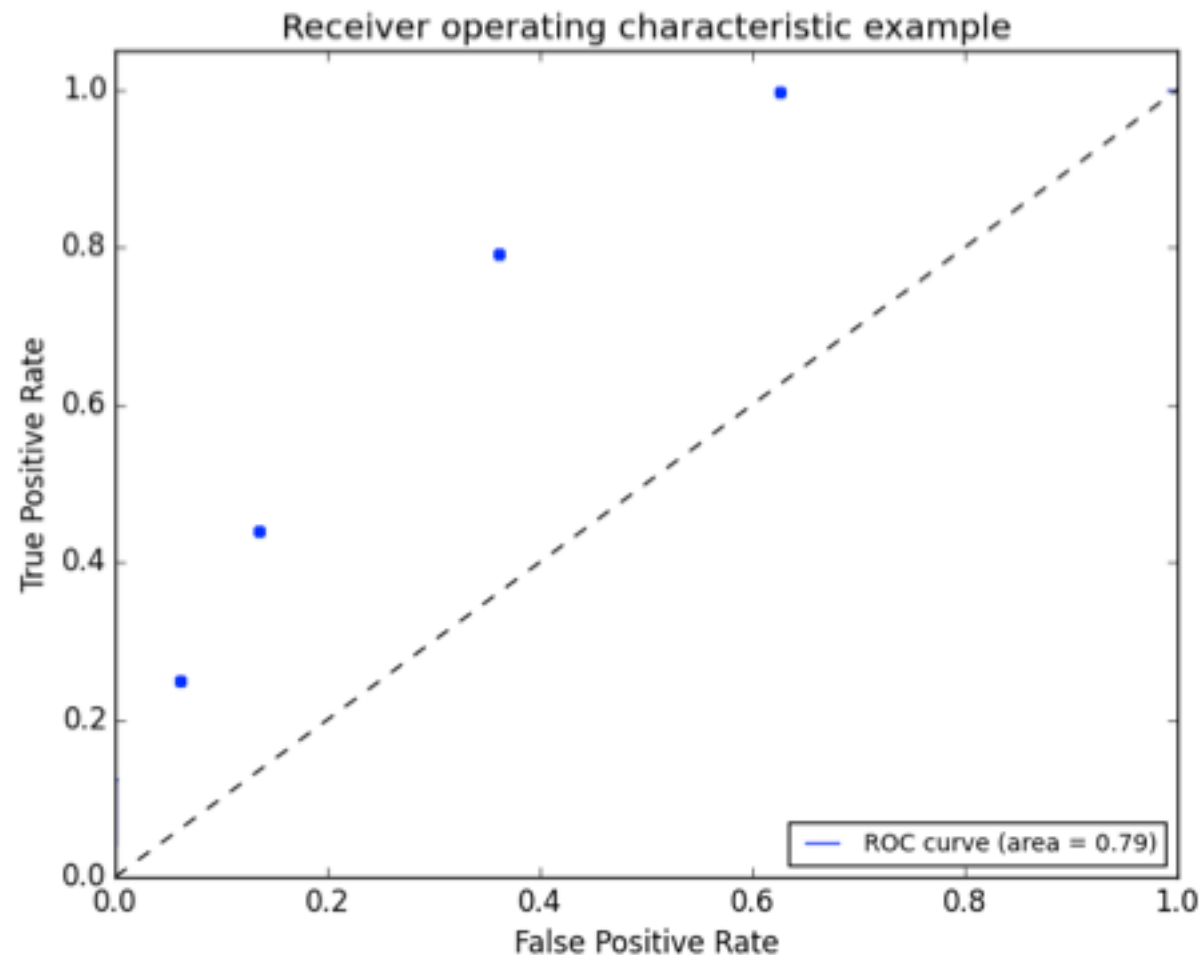
# Advanced Classification Metrics

- We can begin by plotting an individual TPR / FPR pair for one threshold



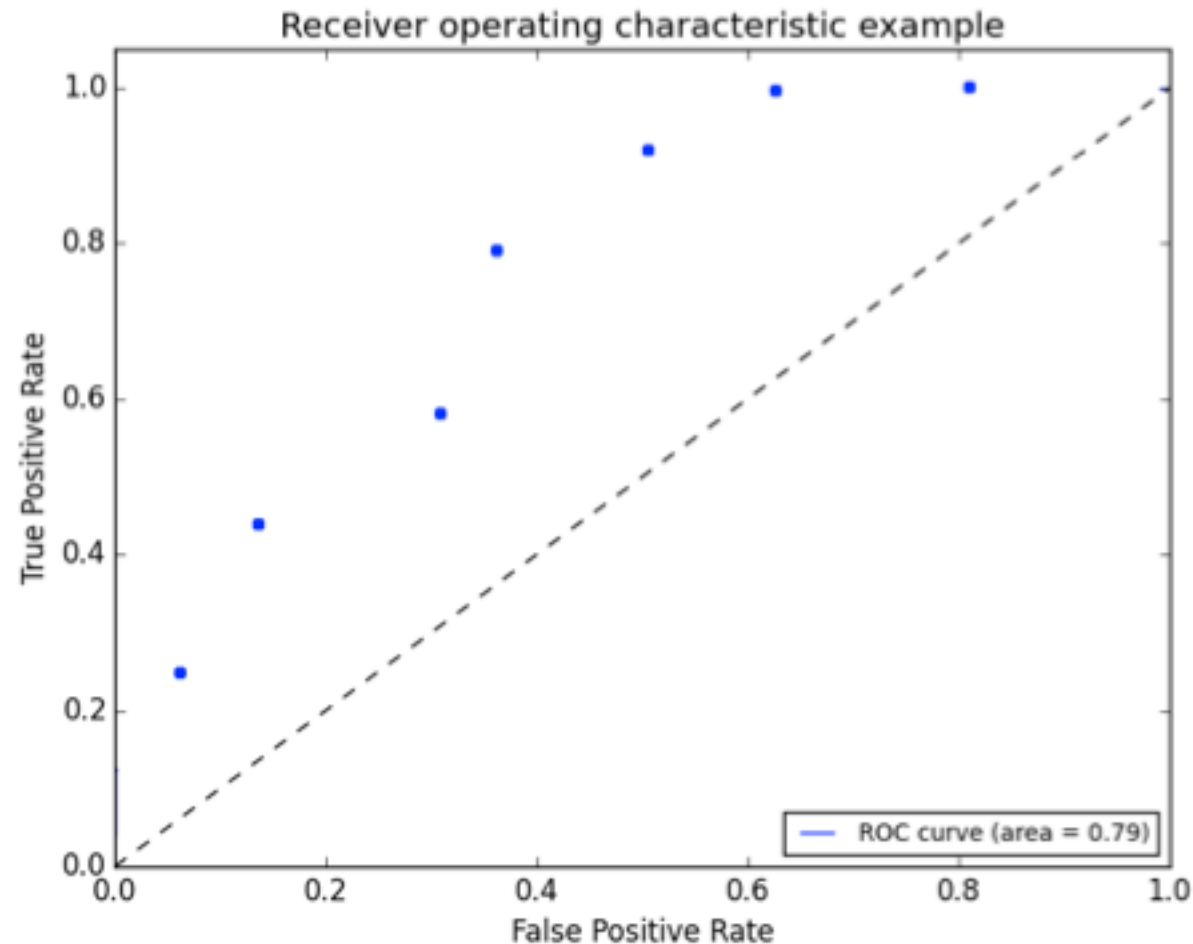
# Advanced Classification Metrics

- We can continue adding pairs for different thresholds



# Advanced Classification Metrics

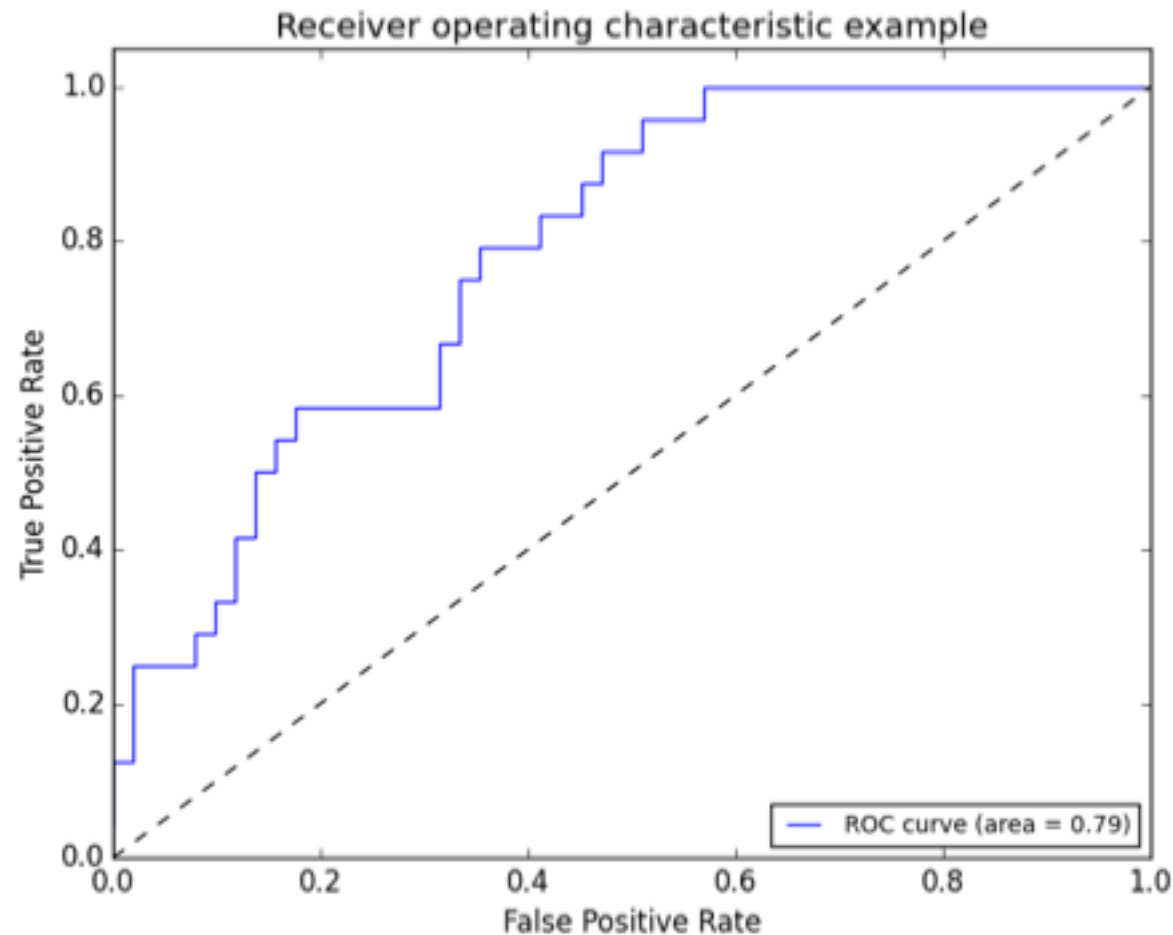
- We can continue adding pairs for different thresholds





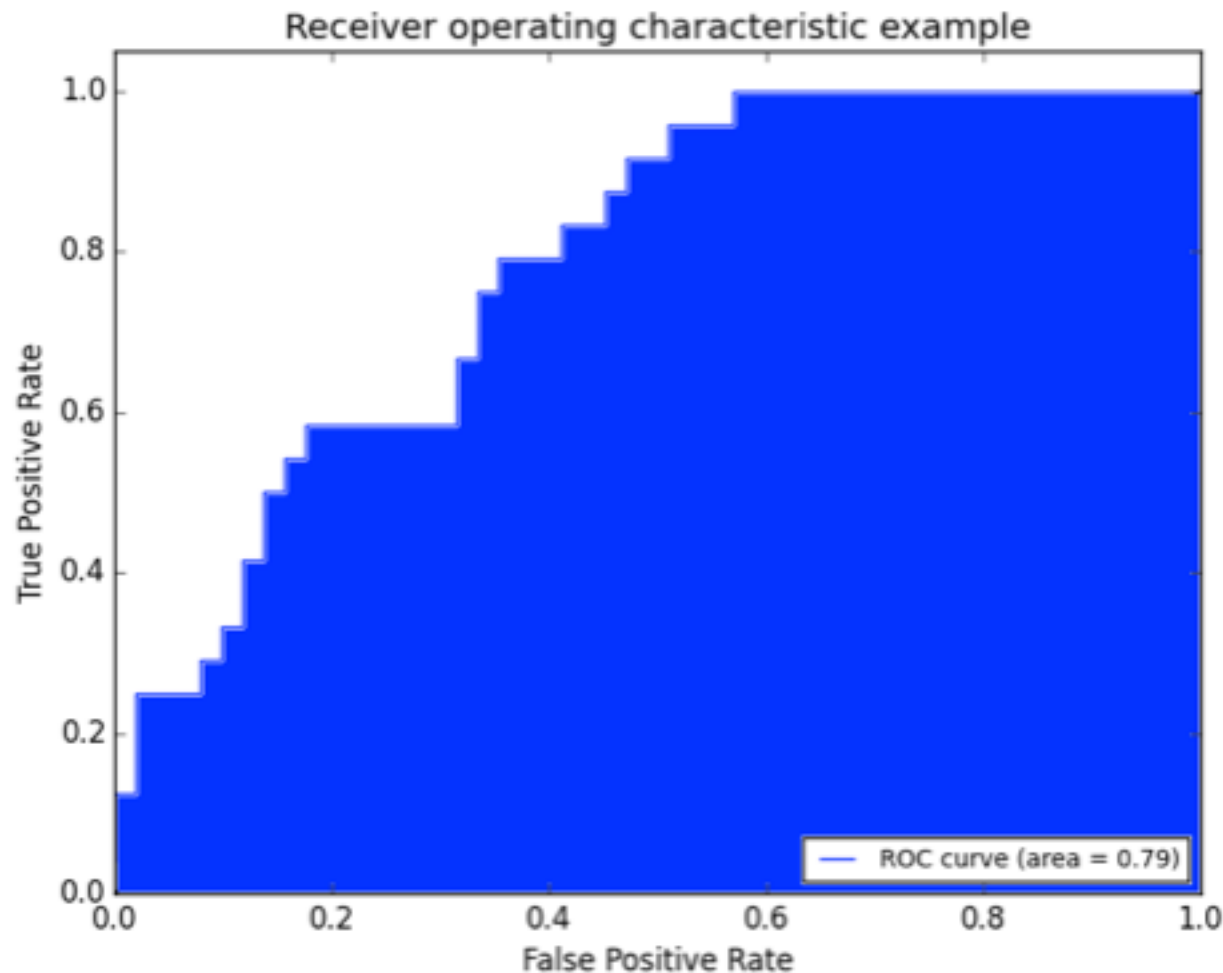
# Advanced Classification Metrics

- Finally, we create a full curve that is described by TPR and FPR



# Advanced Classification Metrics

- With this curve, we can find the Area Under the Curve (AUC)



---

# Advanced Classification Metrics

---

- If we have a TPR of 1 (all positives are marked positive) and FPR of 0 (all negatives are not marked positive), we'd have an AUC of 1. This means everything was accurately predicted.
- If we have a TPR of 0 (all positives are not marked positive) and an FPR of 1 (all negatives are marked positive), we'd have an AUC of 0. This means nothing was predicted accurately.
- An AUC of 0.5 would suggest randomness (somewhat) and is an excellent benchmark to use for comparing predictions (i.e. is my AUC above 0.5?).

---

# Bias Variance Tradeoff

---

- When our error is **biased**, it means the model's prediction is consistently far away from the actual value.
- This could be a sign of poor sampling and poor data or if the model is not a suitable representation of the true relationship.
- One objective of a biased model is to trade bias error for generalized error. We prefer the error to be more evenly distributed across the model.
- This is called error due to **variance**.
- We want our model to *generalize* to data it hasn't seen even if doesn't perform as well on data it has already seen.

# Bias vs Variance

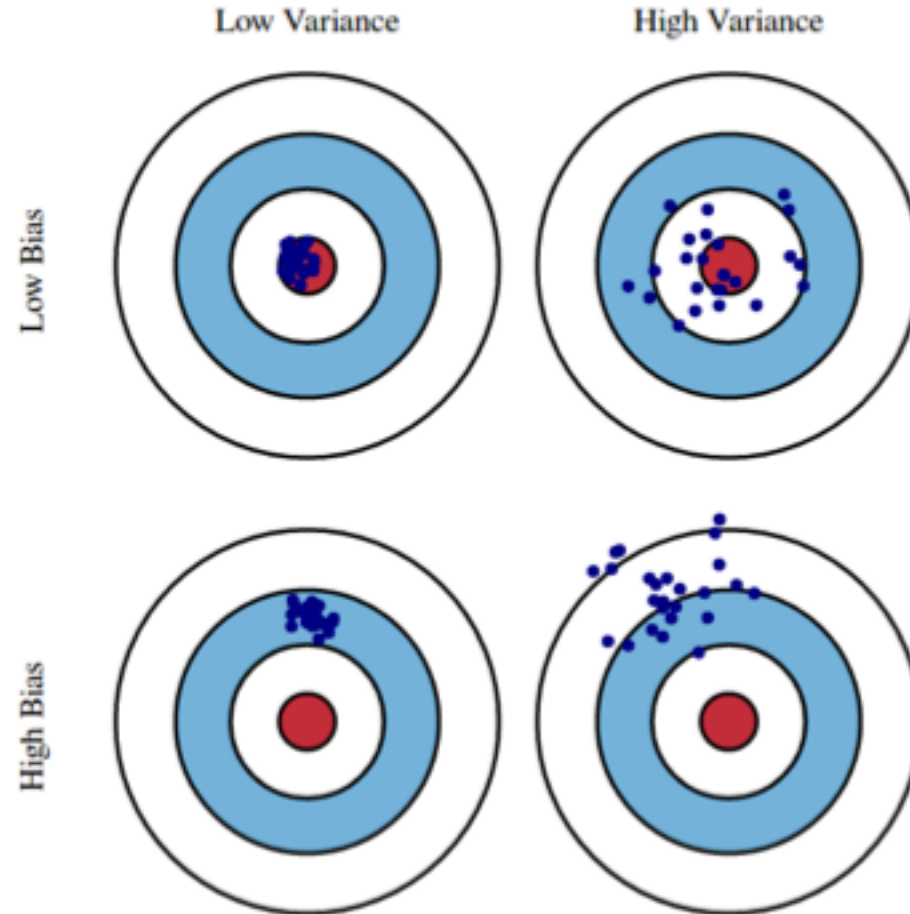
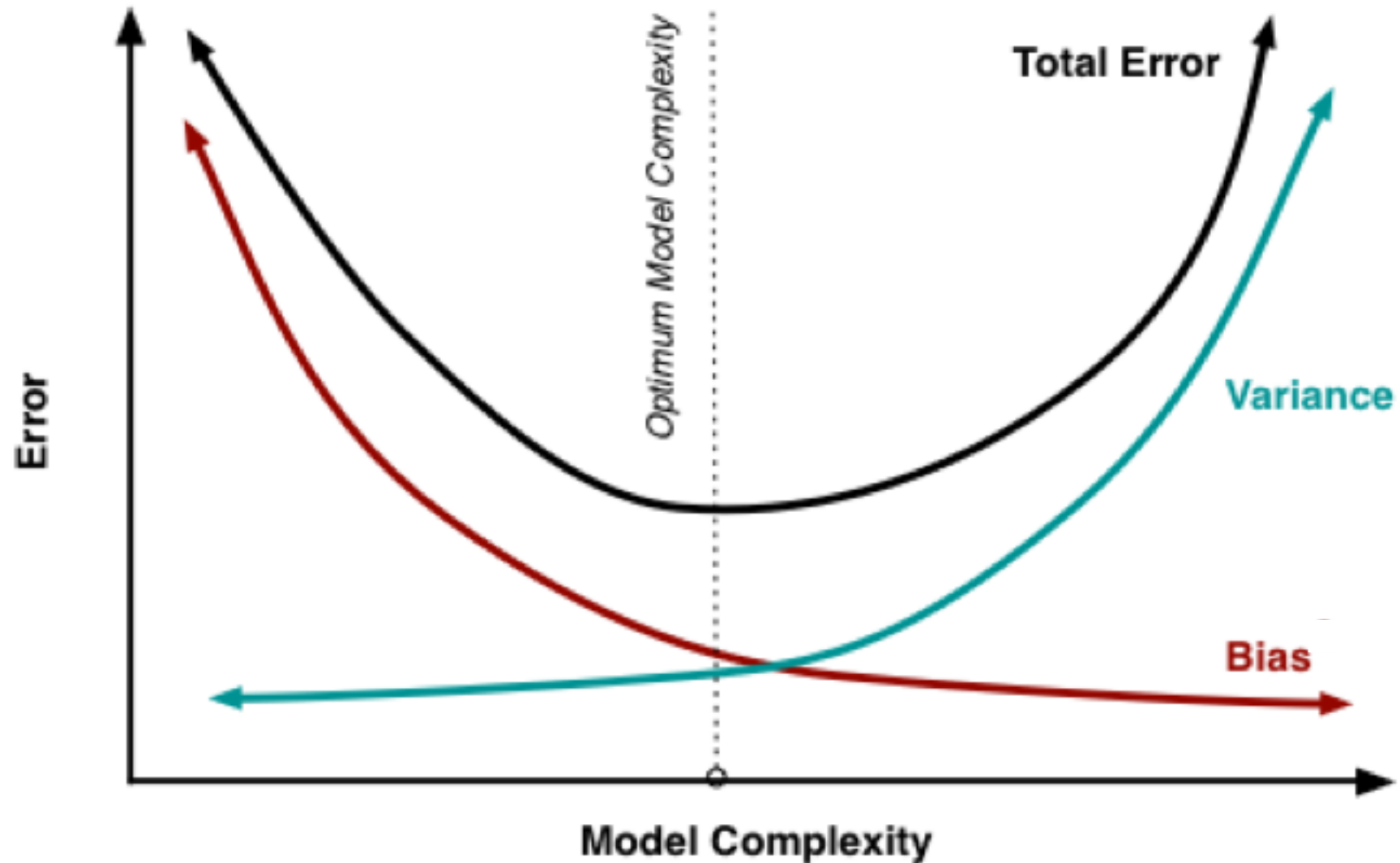


Fig. 1 Graphical illustration of bias and variance.

# Bias Variance Tradeoff



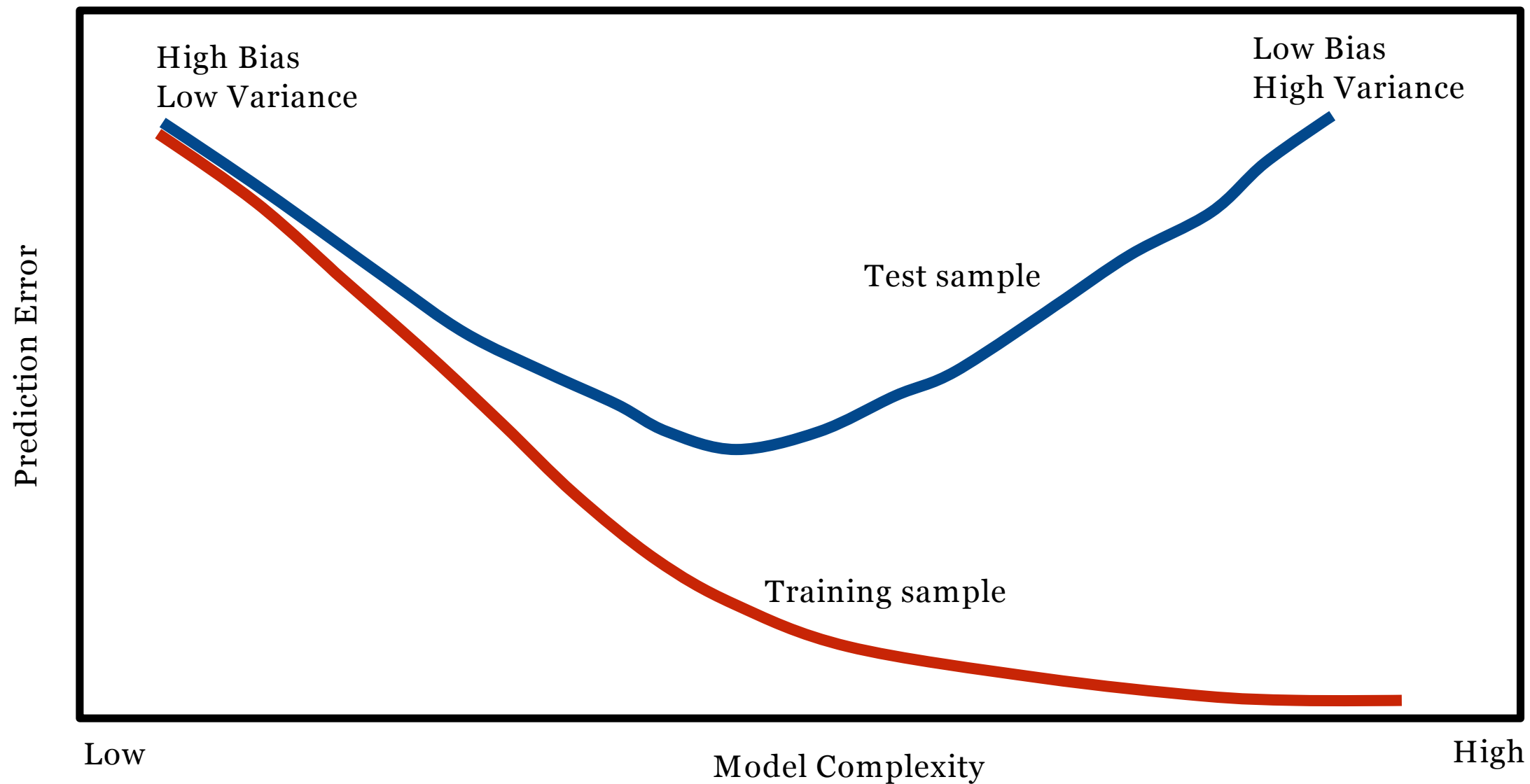
---

# Bias Variance Tradeoff

---

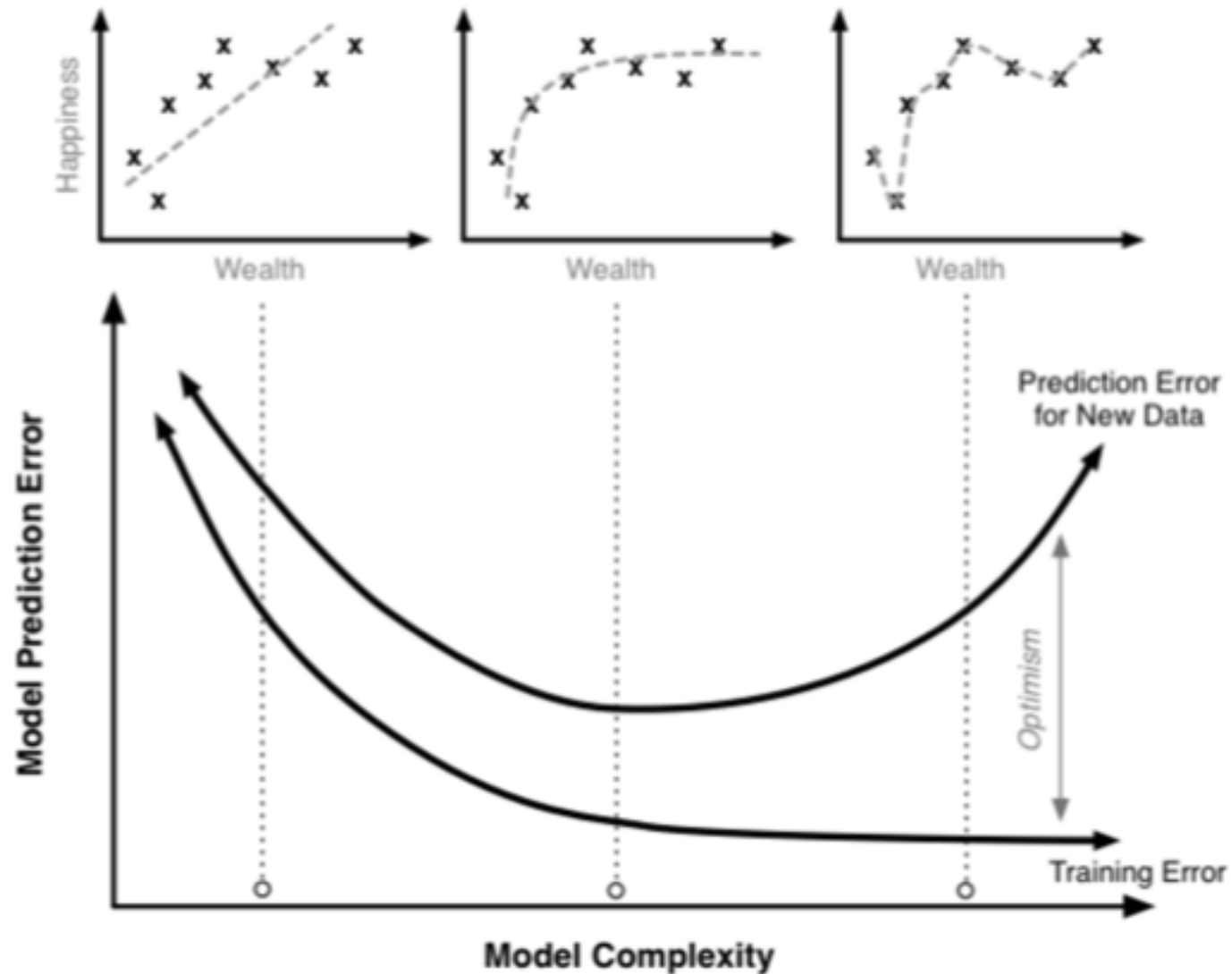
- As model flexibility increases, training MSE will decrease but the test MSE may not
- When a given method yields a small training MSE but a large test MSE, we are said to be overfitting the data
- This happens when the model is working too hard to find patterns in the training data and may be picking up patterns that are just caused by random chance rather than true properties of the unknown function

# Bias Variance Tradeoff





# Bias Variance Tradeoff



---

# Model Validation

---

## Validation Set

- We divide the available data into two parts: a training set and a validation set
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set
- The resulting validation set error provides an estimate of the test error

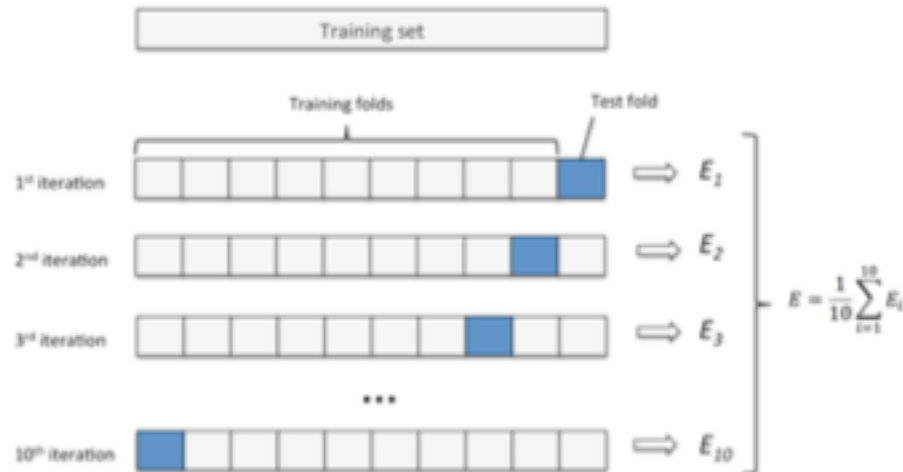
## Limitations

- The validation estimate can be highly variable, depending on which observations are included in the training and validation sets
- Only a subset of the observations are used to train the model

# Model Validation

## K-fold Cross-Validation

- Randomly divide the data into K equal-sized parts
- We leave out part k, fit the model to the other K-1 parts (combined) and then obtain predictions for the left out part
- This is done in turn for each part  $k = 1, 2, \dots, K$  and the results combined



K = 5, 10 are typically used

# K Nearest Neighbors

---

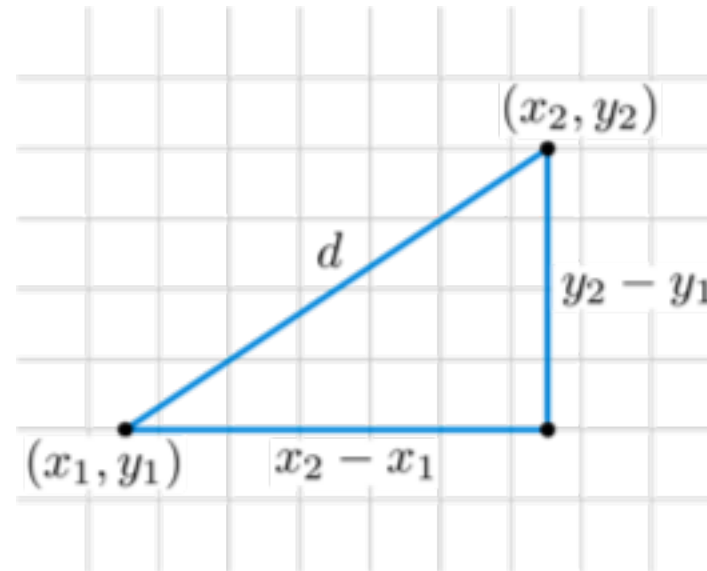
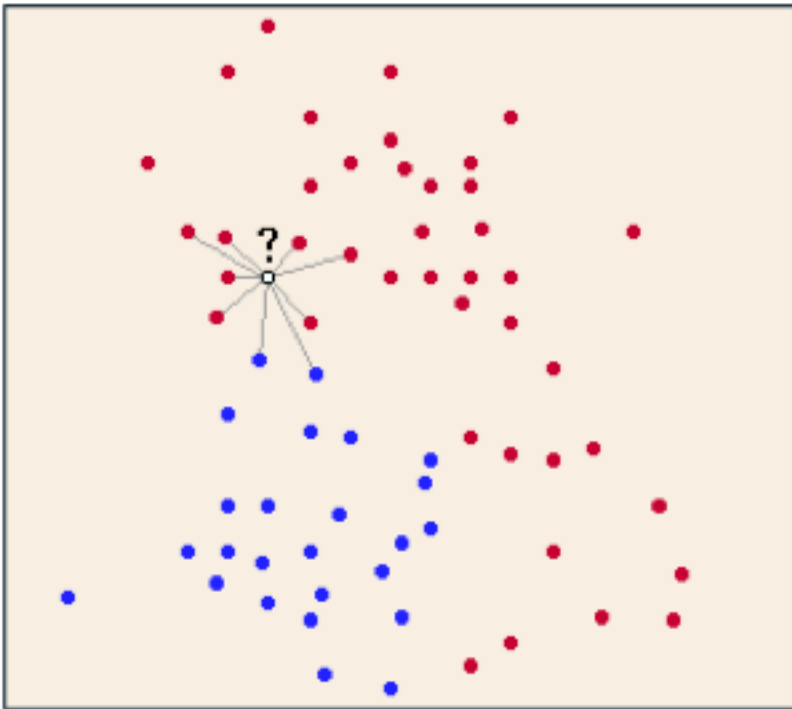
# K Nearest Neighbors

---

- K Nearest Neighbors (KNN) is a classification algorithm that makes a prediction based upon the closest data points.
- The KNN algorithm:
  - For a given point, calculate the distance to all other points.
  - Given those distances, pick the k closest points.
  - Calculate the probability of each class label given those points.
  - The original point is classified as the class label with the largest probability (“votes”).

# K Nearest Neighbors

- KNN uses distance to predict a class label. This application of distance is used as a measure of similarity between classifications.
- We're using shared traits to identify the most likely class label.



---

# K Nearest Neighbors

---

- Suppose we want to determine your favorite type of music. How might we determine this without directly asking you?
- Generally, friends share similar traits and interests (e.g. music, sports teams, hobbies) We could ask your five closest friends what their favorite type of music is and take the majority vote.
- This is the idea behind KNN: we look for things similar to (or close to) our new observation and identify shared traits. We can use this information to make an educated guess about a trait of our new observation.

---

# What Happens in Ties?

---

- What happens if two classes get the same number of votes?
- This could happen in binary classification if we use an even number for  $k$ . This could also happen if there are multiple class labels.
- In sklearn, it will choose the class that it first saw in the training set.
- We could also implement a *weight*, taking into account the distance between the point and its neighbors.
- This can be done in sklearn by changing the weights parameter to “distance”.



---

# What Happens in High Dimensionality?

---

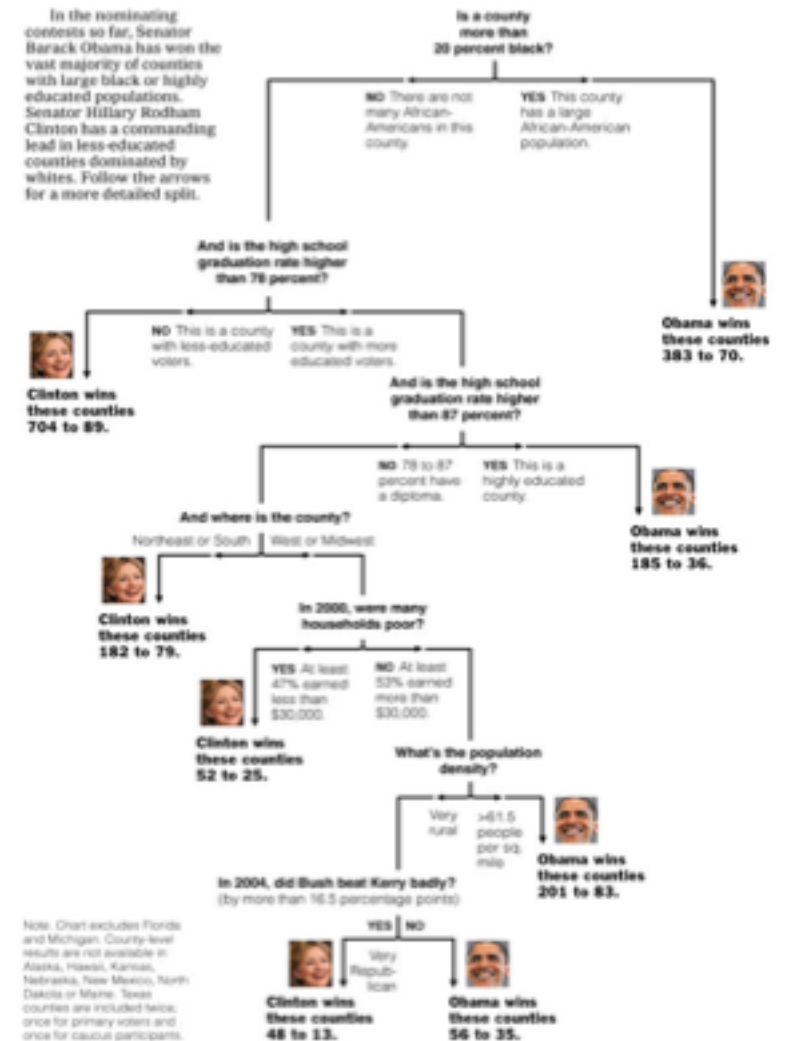
- Since KNN works with distance, higher dimensionality of data (i.e. more features) requires significantly more samples in order to have the same predictive power.
- Consider this: with more dimensions, all points slowly start averaging out to be equally distant. This causes significant issues for KNN.
- Keep the feature space limited and KNN will do well. Exclude extraneous features when using KNN.

# Decision Trees and Random Forests

# Intuition Behind Decision Trees

- Decision trees are like the game “20 questions”. They make decision by answering a series of questions, most often binary questions (yes or no).
- We want the smallest set of questions to get to the right answer.
- Each questions should reduce the search space as much as possible.

Decision Tree: The Obama-Clinton Divide

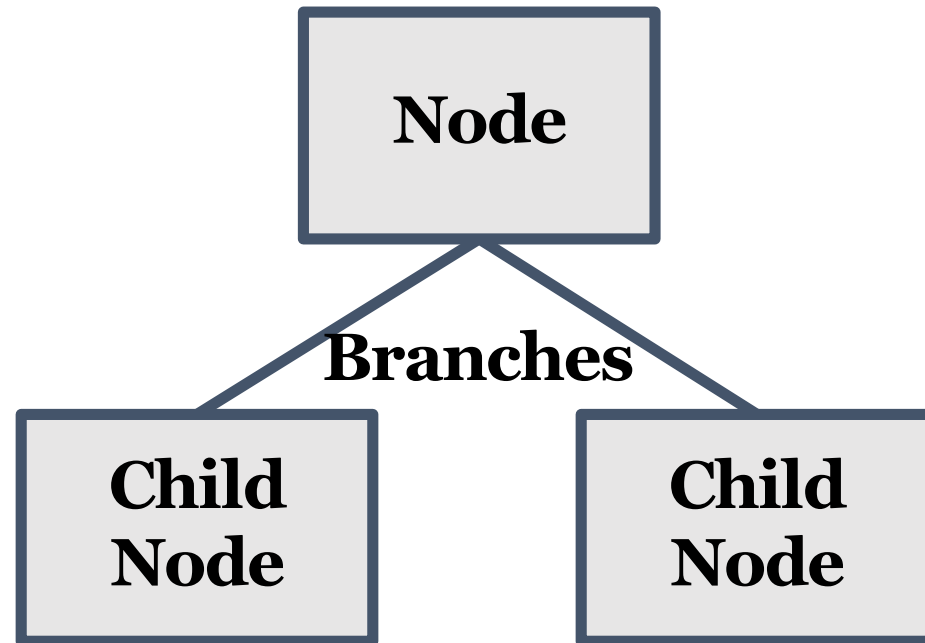


---

# Trees

---

- Trees are a data structure made up of nodes and branches.
- Each node typically has two or more branches that connect it to its children.

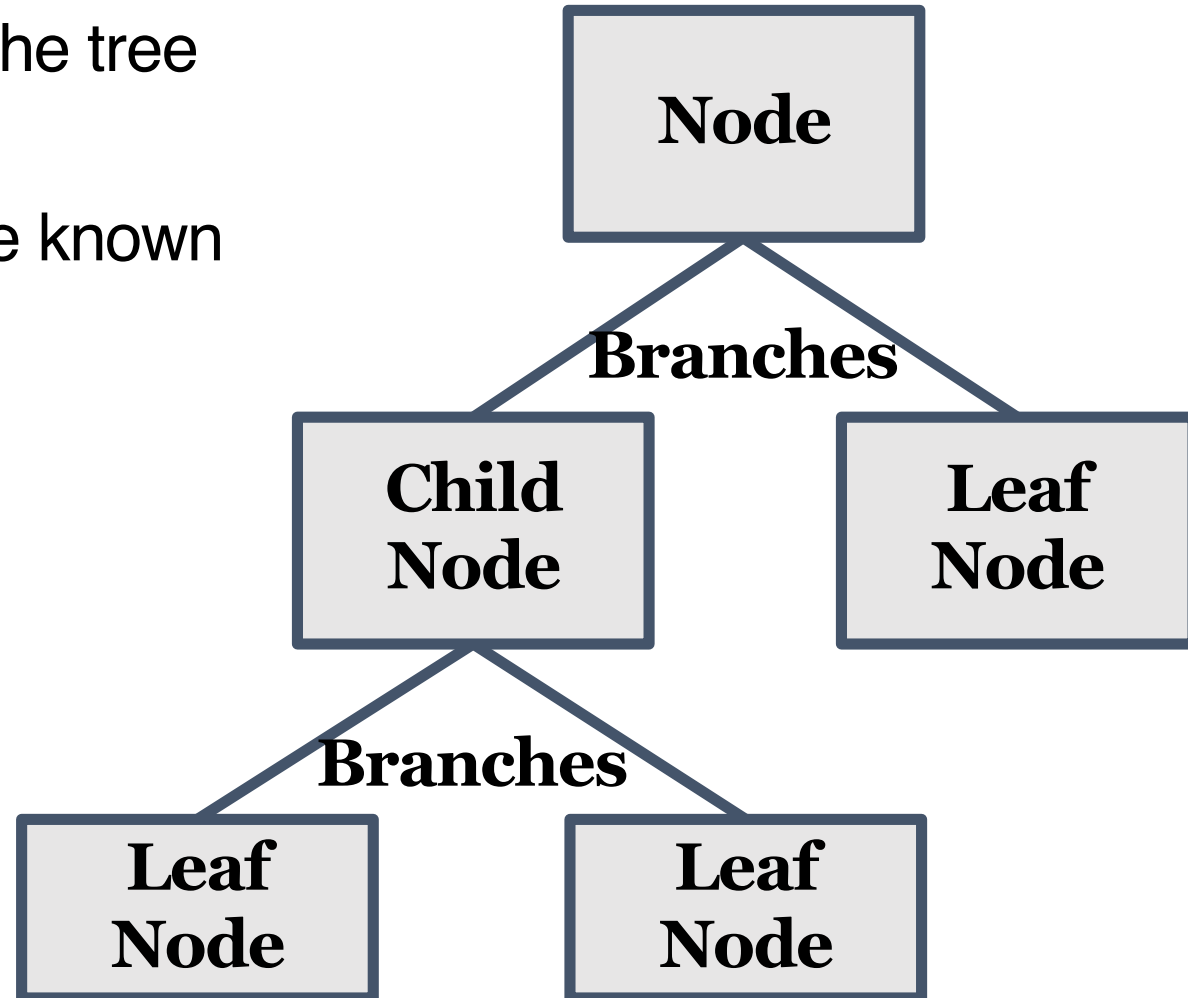


---

# Trees

---

- Each child is another node in the tree and contains its own subtree.
- Nodes without any children are known as leaf nodes.

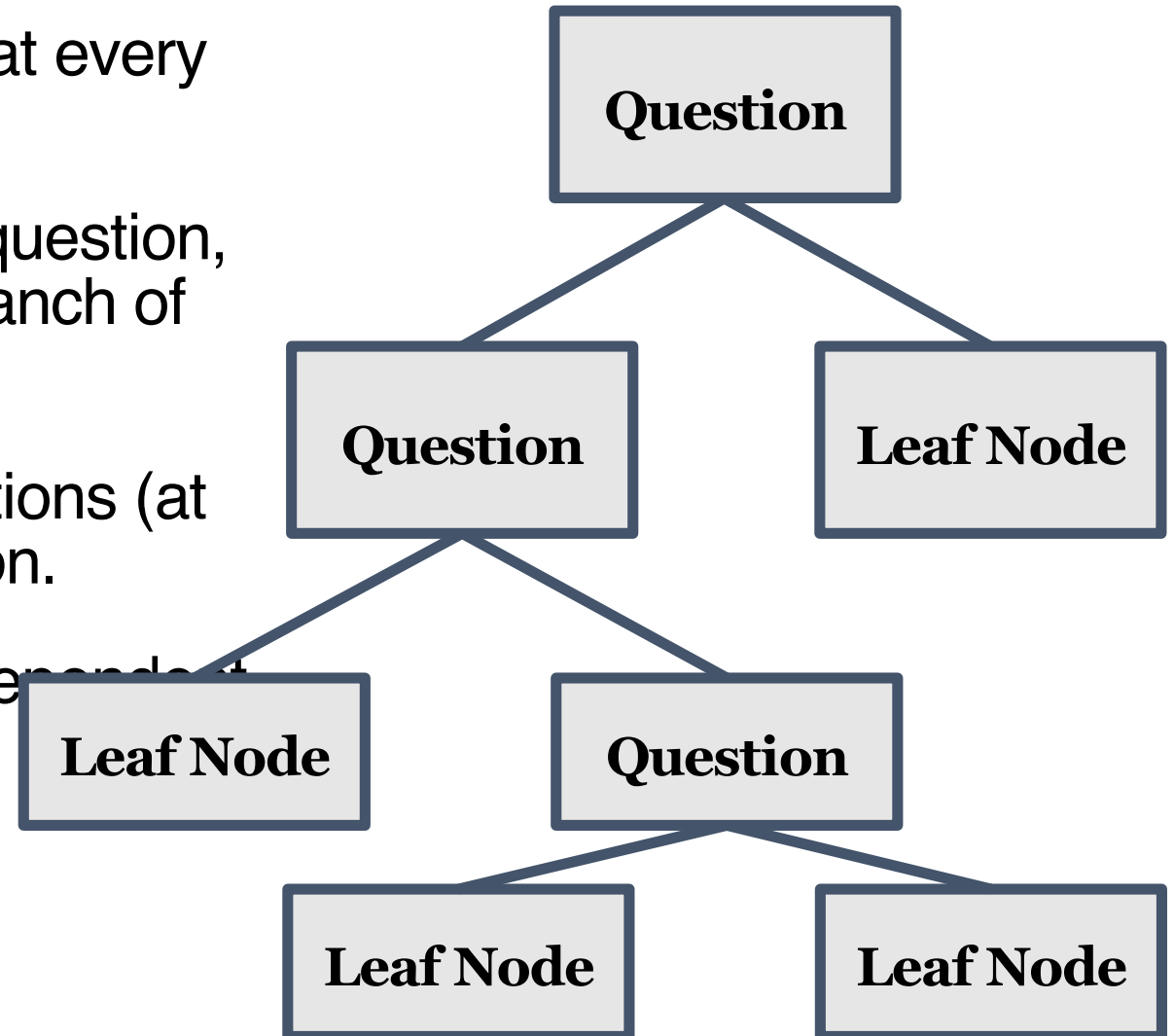


---

# Decision Trees

---

- A *decision tree* contains a question at every node.
- Depending upon the answer to the question, we proceed down the left or right branch of the tree and ask another question.
- Once we don't have any more questions (at the *leaf* nodes), we make a prediction.
- Note: The next question is always dependent on the last.



---

# Decision Trees

---

- Let's suppose we want to predict if an article is a news article.
- What questions should we ask to make a prediction?
- How many questions should we ask?

---

# Decision Trees

---

- We may start by asking: does it mention a President?
- If it does, it must be a news article.
- If not, let's ask another question: does the article contain other political features?
- If not, does the article contain references to political topics?
- We could keep going on in this manner until we were satisfied.



---

# Comparison to Previous Models

---

- Decision trees are *non-linear*, an advantage over logistic regression.
- A *linear* model is one in which a change in an input variable has a constant change on the output variable.

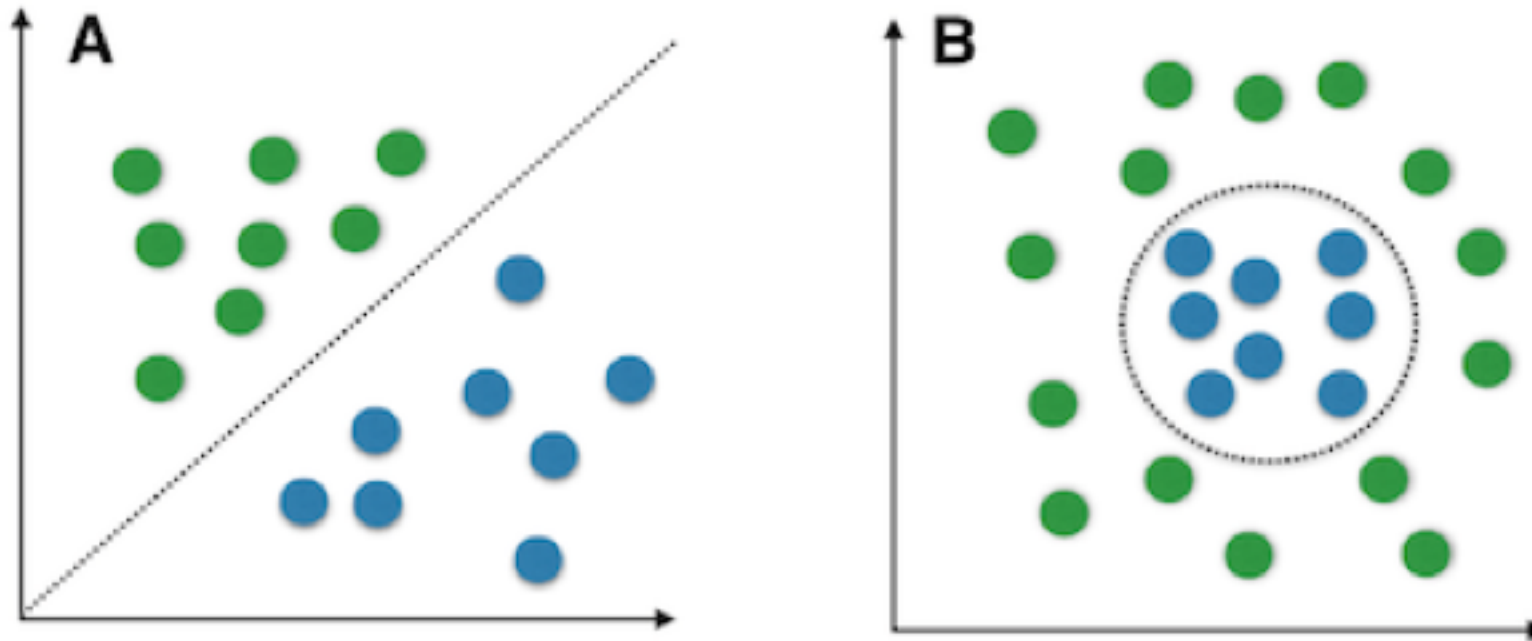
---

# Comparison to Previous Models

---

- Linear vs. non-linear classification models

Linear vs. nonlinear problems



---

# Comparison to Previous Models

---

- An example of this difference is the relationship between years of education and salary. In a *linear* model, the increase in salary from 10 to 15 years of education would be the same as the increase in salary from 15 to 20 years of education. In a *non-linear* model, salary can change dramatically for years 0-15 and negligibly from years 15-20.
- Trees automatically contain interaction of features, since each question is dependent on the last.

---

# Training a Decision Tree Model

---

- Training a decision model is deciding the best set of questions to ask.
- A good question will be one that best segregates the positive group from the negative group and then narrows in on the correct answer.
- For example, in our news article decision tree, the best question is one that creates two groups, one that is mostly news stories and one that is mostly non-news stories.

---

# Training a Decision Tree Model

---

- We can quantify the *purity* of the separation of groups using Classification Error, Entropy, or Gini Coefficient.
- We want to choose the question that gives us the best *change* in our purity measure. At each step, we can ask, “Given our current set of data points, which question will make the largest change in purity?”
- This is done *recursively* for each new set of two groups until we reach a stopping point.

---

# Training a Decision Tree Model

---

- Classification Error: Fraction of training observations in the region that do not belong to the most common class

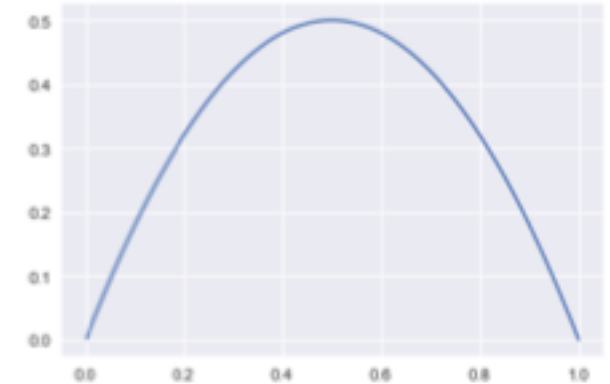
$$E = 1 - \max_k(\hat{p}_{mk})$$

- $\hat{p}_{mk}$  represents the proportion of training observations in the mth region that are from the kth class
- Classification error however is generally not sufficiently sensitive for tree-growing and in practice the two other measures are preferable

# Training a Decision Tree Model

- Gini Index is defined by:

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



- The Gini Index takes on a small value if all the  $\hat{p}_{mk}$  are close to zero or one. For this reason the Gini Index is referred to as a measure of node purity

$$0.9, 0.1 \Rightarrow G = 0.9 * (1-0.9) + 0.1 * (1-0.1) = 0.18$$

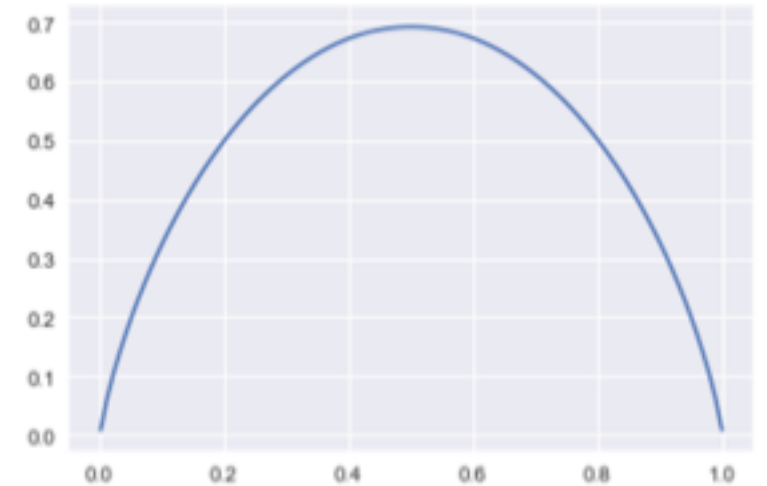
$$0.4, 0.6 \Rightarrow G = 0.4 * (1-0.4) + 0.6 * (1-0.6) = 0.48$$

# Training a Decision Tree Model

- Cross Entropy is defined by:

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

- Since  $0 \leq \hat{p}_{mk} \leq 1$  it follows that  $0 \leq -\hat{p}_{mk} \log(\hat{p}_{mk})$
- The Cross Entropy will take on a value near zero if all the  $\hat{p}_{mk}$  are all near zero or near one



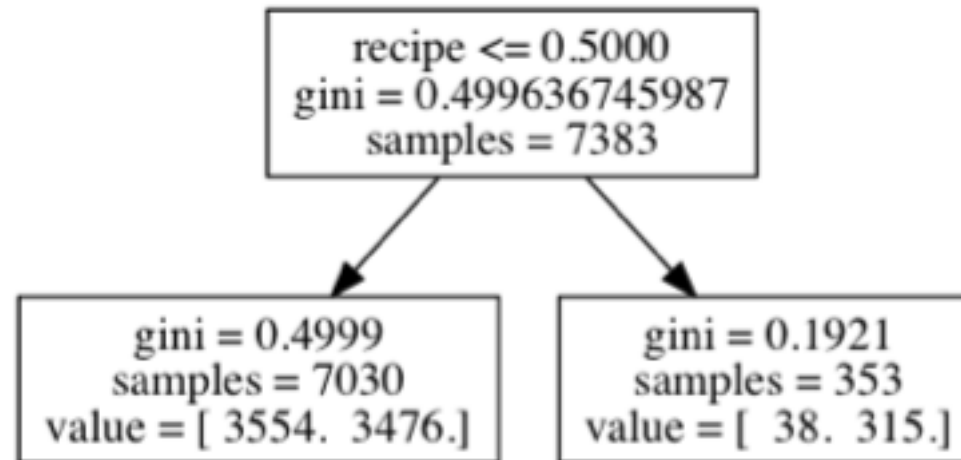


---

# Training a Decision Tree Model

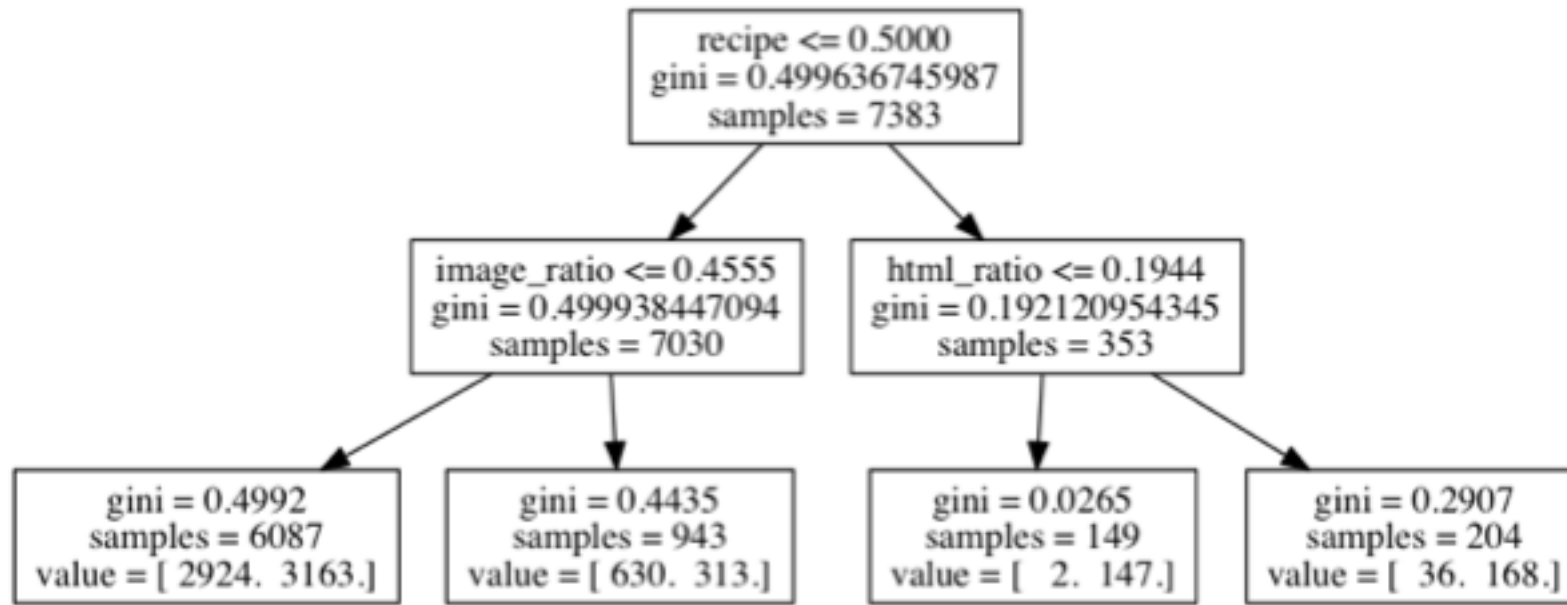
---

- Let's build a sample tree for our evergreen prediction problem. Assume our features are whether the article contains a recipe, the image ratio, the html ratio.
- First, let's choose the feature that gives us the highest purity, the recipe feature.



# Training a Decision Tree Model

- We can take each side of the tree and repeat the process.



- We can continue this process until we have asked as many questions as we want or until our leaf nodes are completely pure.

---

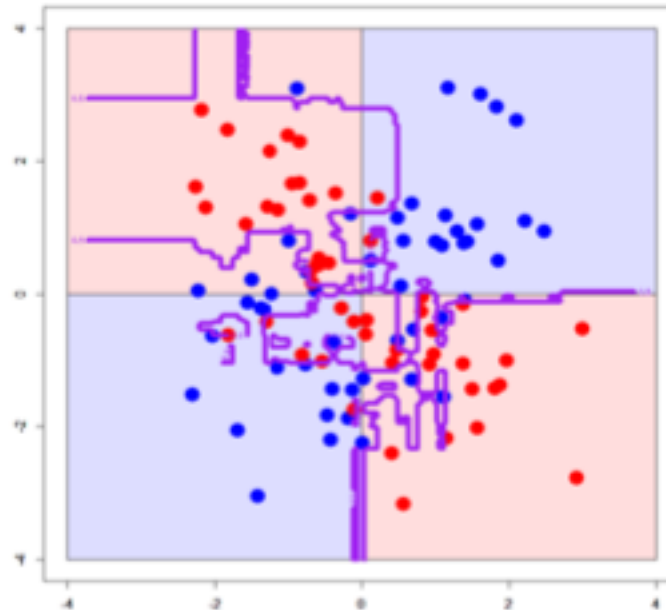
# Making Predictions from a Decision Tree

---

- Predictions are made by answering each of the questions.
- Once we reach a leaf node, our prediction is made by taking the majority label of the training samples that fulfill the questions.
- In our sample tree, if we want to classify a new article, ask:
  - Does the article contain the word recipe?
  - If it doesn't, does the article have a lot of images?
  - If it does, then 313 / 943 article are evergreen.
  - So we can assign a 0.33 probability for evergreen sites.

# Overfitting in Decision Trees

- Decision trees tend to be weak models because they can easily memorize or overfit to a dataset.
- A model is *overfit* when it memorizes or bends to a few specific data points rather than picking up general trends in the data.



---

# Overfitting in Decision Trees

---

- An unconstrained decision tree can learn an extreme tree (e.g. one feature for each word in a news article).
- We can limit our decision trees using a few methods.
  - Limiting the number of questions (nodes) a tree can have).
  - Limiting the number of samples in the leaf nodes.

---

# Advantages and Disadvantages of Decision Trees

---

## **Advantages:**

- Trees are very easy to explain to people
- Some people believe that decision trees more closely mirror human decision-making than other approaches
- Trees can be displayed graphically and are easily interpreted even by a non-expert

## **Disadvantages:**

- Trees generally do not have the same level of predictive accuracy as other approaches
- Trees can be very non-robust, i.e. a small change in the data can cause a large change in the final estimated tree

---

# Random Forests

---

- Random forest models are one of the most widespread classifiers used.
- They are relatively simple to use and help avoid overfitting.
- Random Forests are an *ensemble* or collection of individual decision trees.

---

# Training a Random Forest

---

- Training a random forest model involves training many decision tree models.
- Since decision trees overfit easily, we use many decision trees together and randomize the way they are created.



---

# Training a Random Forest

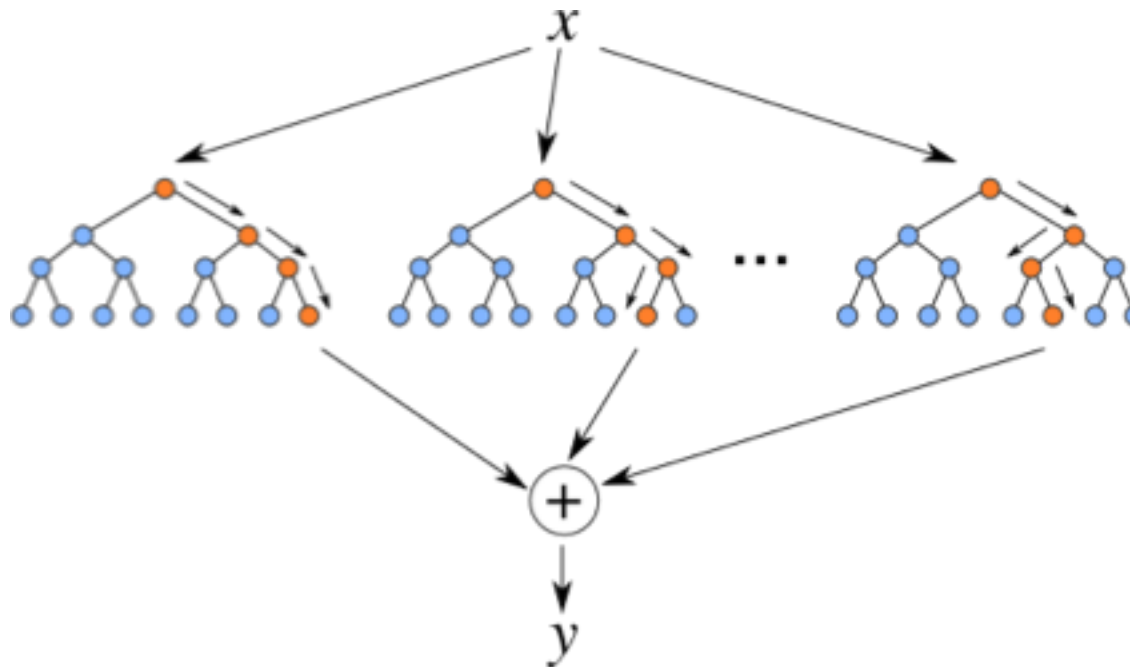
---

## Random Forest Algorithm

1. Take a bootstrap sample of the dataset.
2. Train a decision tree on the bootstrap sample. For each split/feature selection, only evaluate a *limited* number of features to find the best decision tree.
3. Repeat this for N trees.

# Predictions Using a Random Forest

- Predictions for a random forest model come from each decision tree.
- Make an individual prediction with each decision tree.
- Combine the individual predictions and take the majority vote.



---

# Regression Trees

---

- For regression trees, the prediction is the mean value of all the values in the node
- The goal is to find nodes  $R_1, \dots, R_J$  based on minimizing the RSS

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

---

# Visual Introduction to Machine Learning

---

- <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>
- <http://www.r2d3.us/visual-intro-to-machine-learning-part-2/>

# Clustering

---

# Machine Learning Categories

---

|                   | <i>Supervised Learning</i>       | <i>Unsupervised Learning</i> |
|-------------------|----------------------------------|------------------------------|
| <i>Discrete</i>   | classification or categorization | clustering                   |
| <i>Continuous</i> | regression                       | dimensionality reduction     |

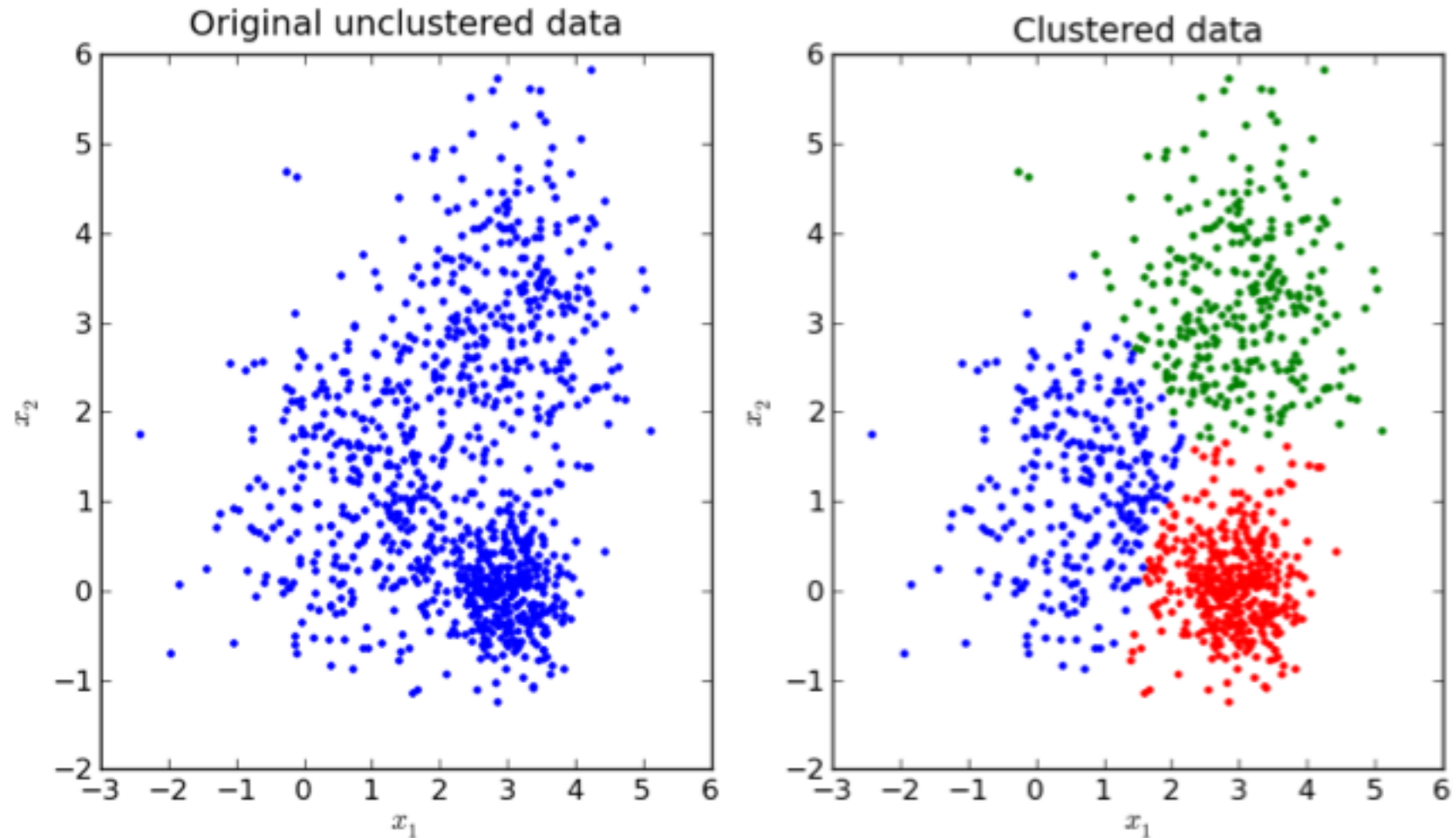
---

# Unsupervised Learning

---

- So far all the algorithms we have used are *supervised*: each observation (row of data) came with one or more *labels*, either *categorical variables* (classes) or *measurements* (regression)
- **Unsupervised learning** has a different goal: **feature discovery**
- **Clustering** is a common and fundamental example of unsupervised learning
- **Clustering** algorithms try to find meaningful groups within data

# Clustering: Centroids



<http://stackoverflow.com/questions/24645068/k-means-clustering-major-understanding-issue>



---

# K-Means Clustering

---

- k-Means clustering is a popular centroid-based clustering algorithm
- Basic idea: find k clusters in the data centrally located around various mean points
- Demo <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

---

# K-Means Clustering

---

- k-Means seeks to minimize the sum of squares about the means
- Precisely, find  $k$  subsets  $S_1, \dots, S_k$  of the data with means  $\mu_1, \dots, \mu_k$  that minimizes:

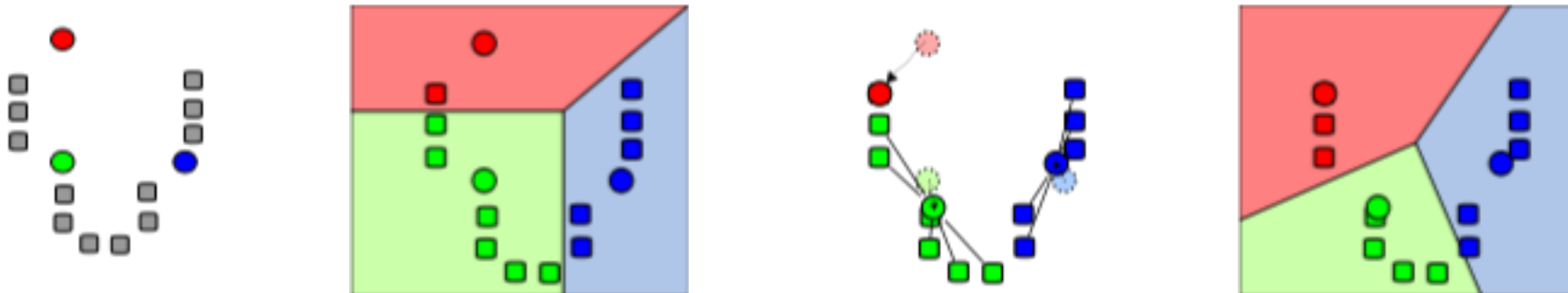
$$\sum_{i=1}^k \sum_{x \in S_i} |x - \mu_i|^2$$

---

# K-Means Clustering

---

- This is a computationally difficult problem to solve so we rely on heuristics
- The “standard” heuristic is called “Lloyd’s Algorithm”:
  - Start with k initial mean values
  - Data points are then split up into a Voronoi diagram
  - Each point is assigned to the “closest” mean
  - Calculate new means based on centroids of points in the cluster
  - Repeat until clusters do not change



---

# K-Means Clustering

---

- Assumptions are important! k-Means assumes:
  - k is the correct number of clusters
  - data is isotropically distributed (circular/spherical distribution)
  - variance is the same for each variable
  - clusters are roughly the same size
- Nice counterexamples / cases where assumptions are not met:
  - <http://varianceexplained.org/r/kmeans-free-lunch/>
  - [http://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_assumptions.html](http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html)

---

# DBSCAN Clustering

---

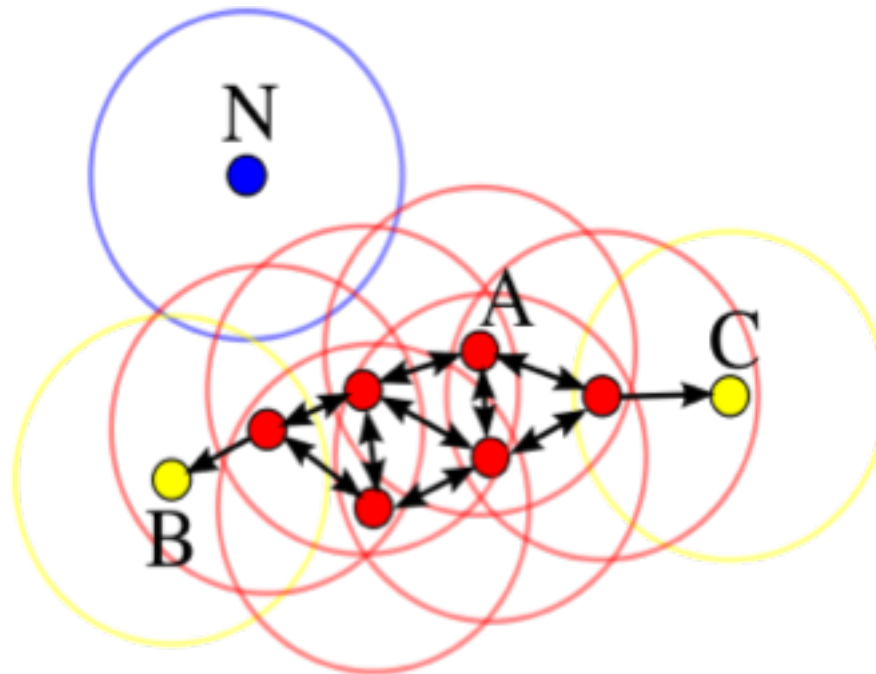
- Density-based spatial clustering of applications with noise
- Main idea: Group together closely-packed points by identifying
  - Core points
  - Reachable points
  - Outliers (not reachable)
- Two parameters:
  - Min\_samples
  - Eps

---

# DBSCAN Clustering

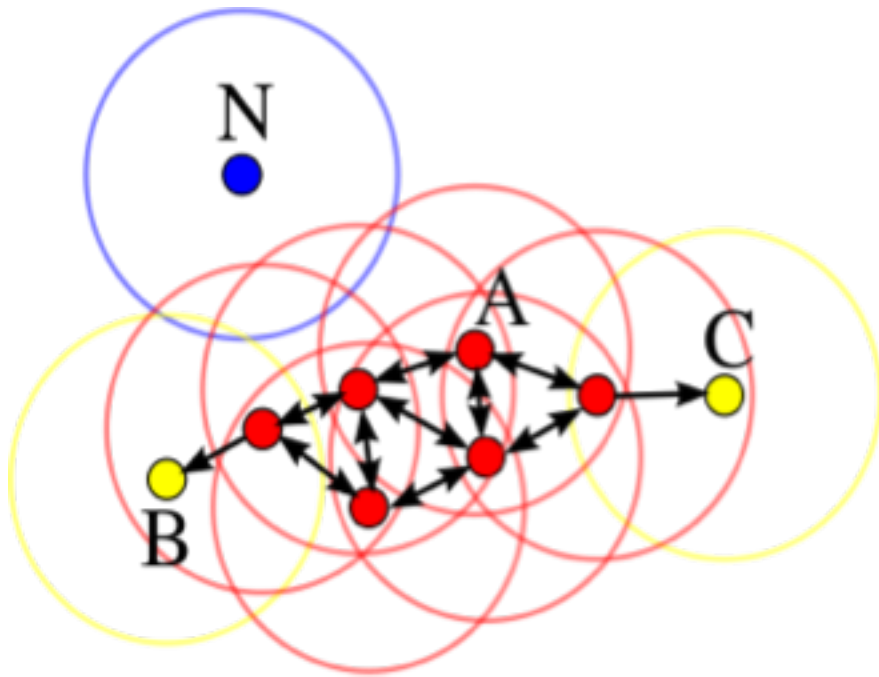
---

- Core point: at least  $\text{min\_samples}$  points within  $\text{eps}$  of the core point
  - Such points are directly reachable from the core point
- Reachable: point  $q$  is reachable from  $p$  if there is a path of core points from  $p$  to  $q$
- Outlier: not reachable



# DBSCAN Clustering

- A cluster is a collection of connected core and reachable points



- In this diagram,  $\text{minPts} = 4$ . Point A and the other red points are **core** points, because the area surrounding these points in an  $\epsilon$  radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster.
- Points B and C are not core points, but are **reachable** from A (via other core points) and thus belong to the **cluster** as well.
- Point N is a **noise** point that is neither a core point nor directly-reachable.

# Concluding Remarks