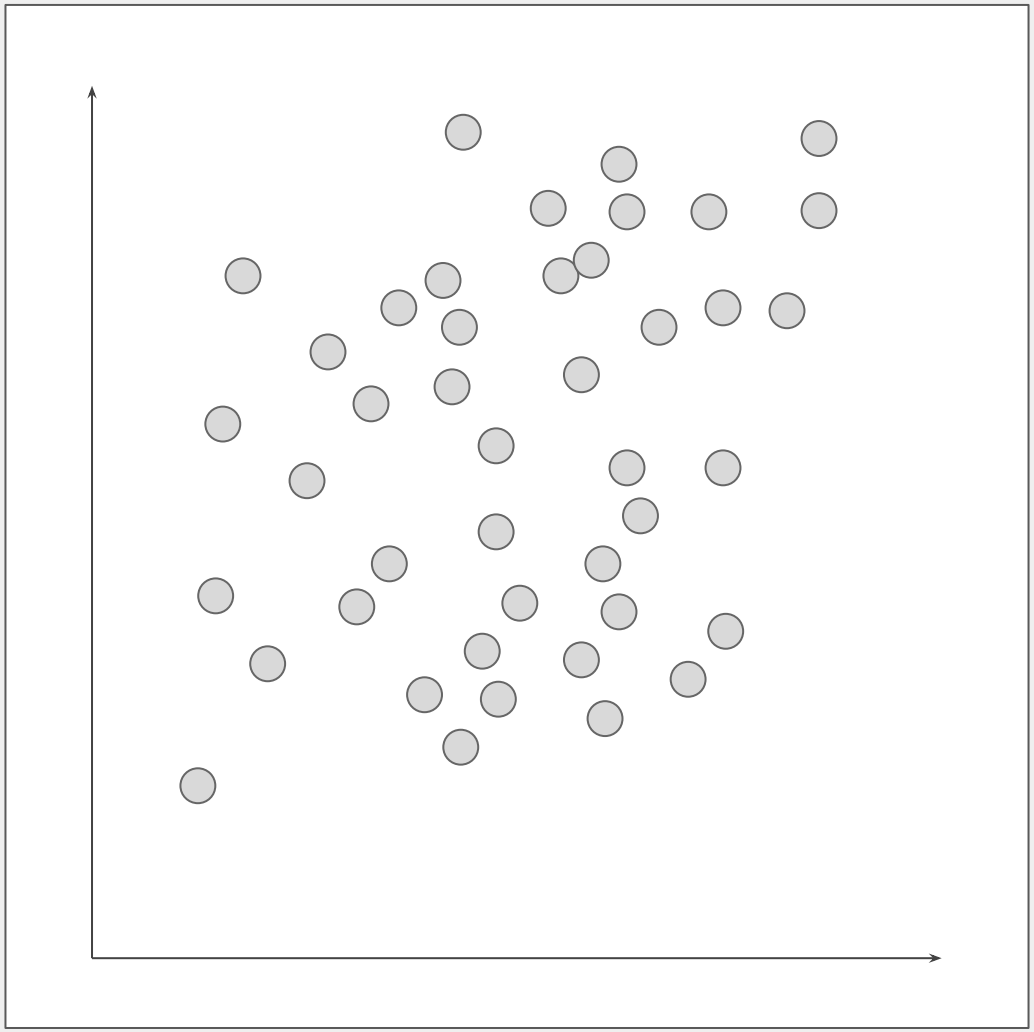
Classification Models

CEB Day 4: January 27, 2017

DistrictDataLabs

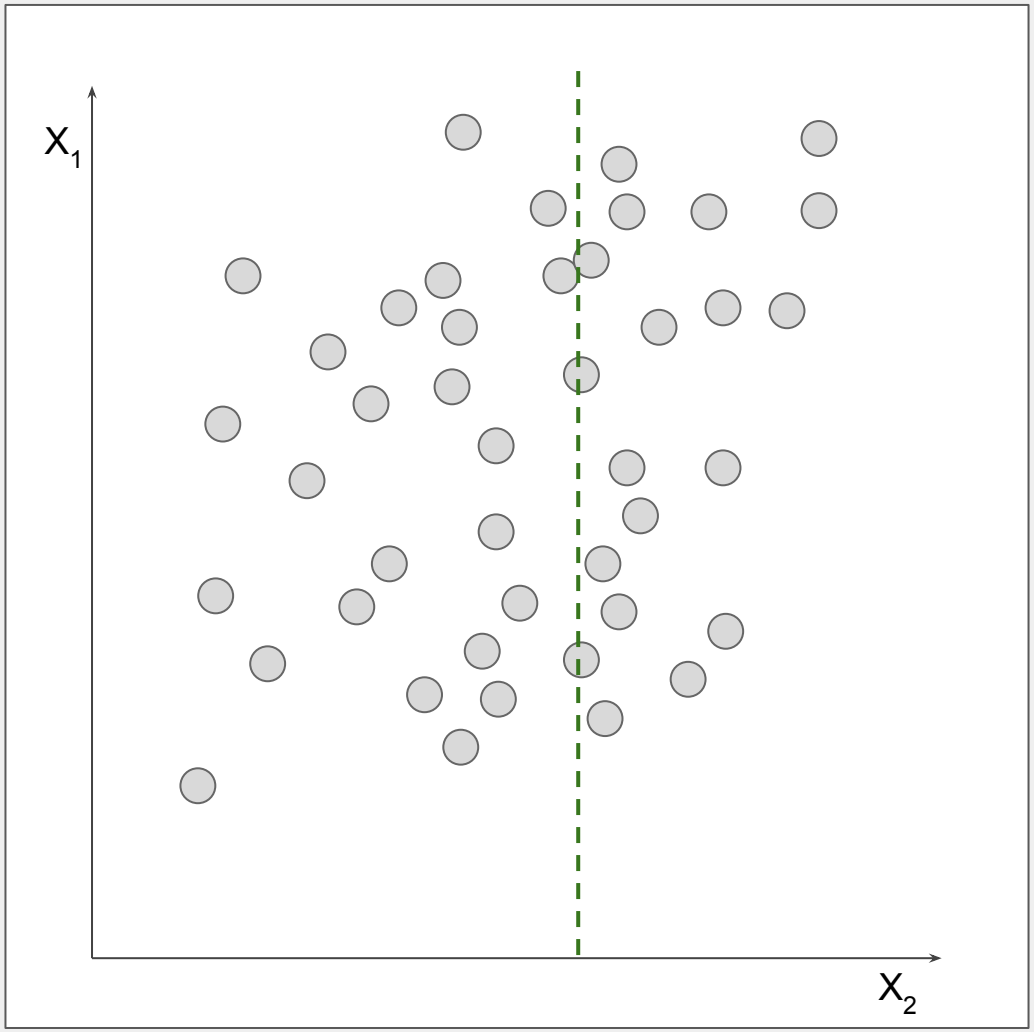
How do you make predictions from data?

DistrictDataLabs



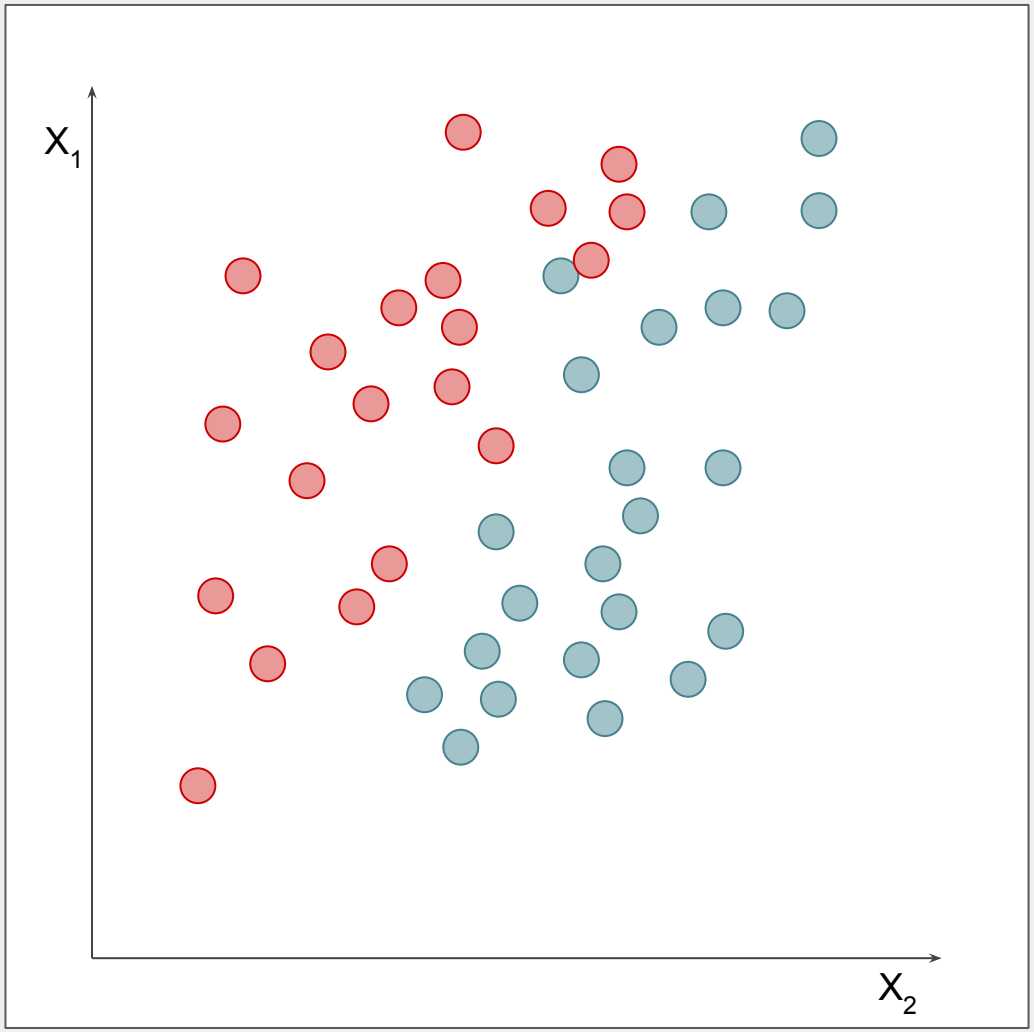
What patterns do you see?

DistrictDataLabs



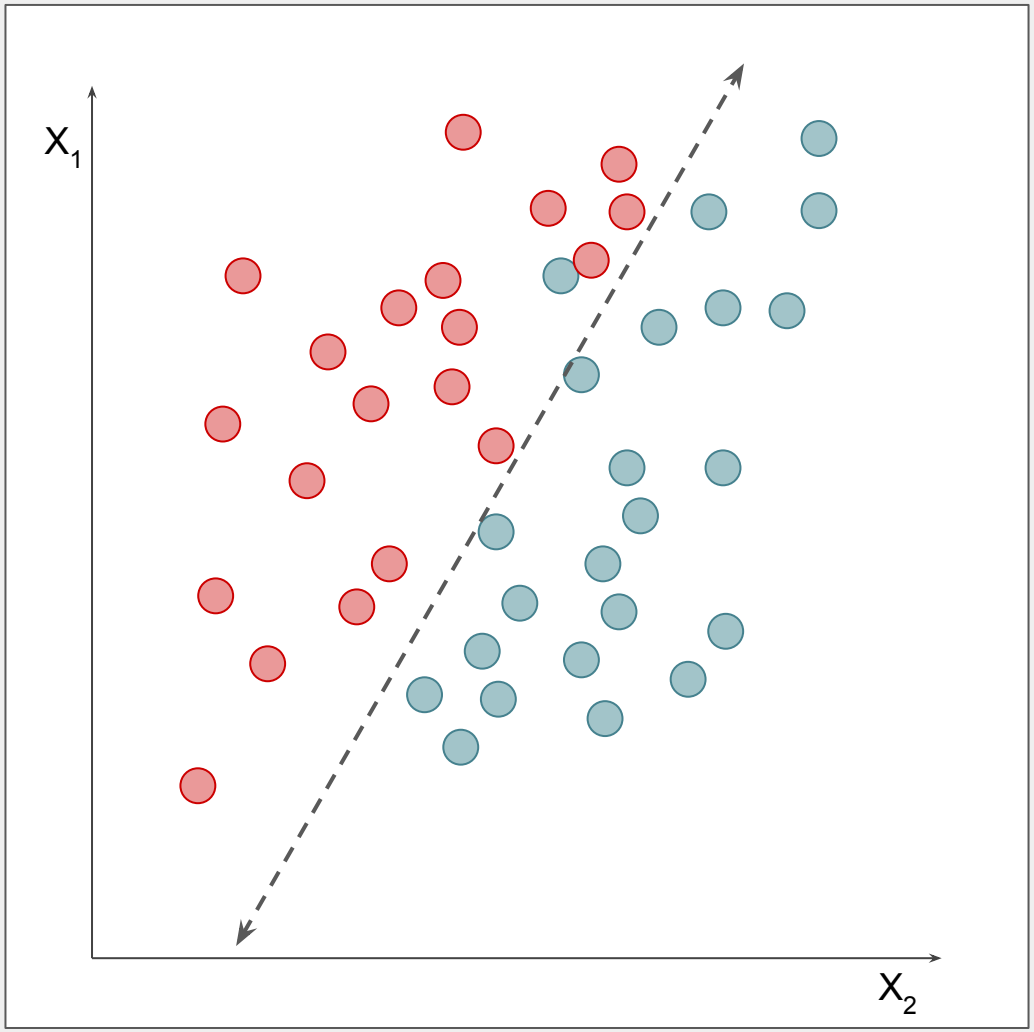
What is the X_1 value?

DistrictDataLabs



How do you determine red from blue?

DistrictDataLabs

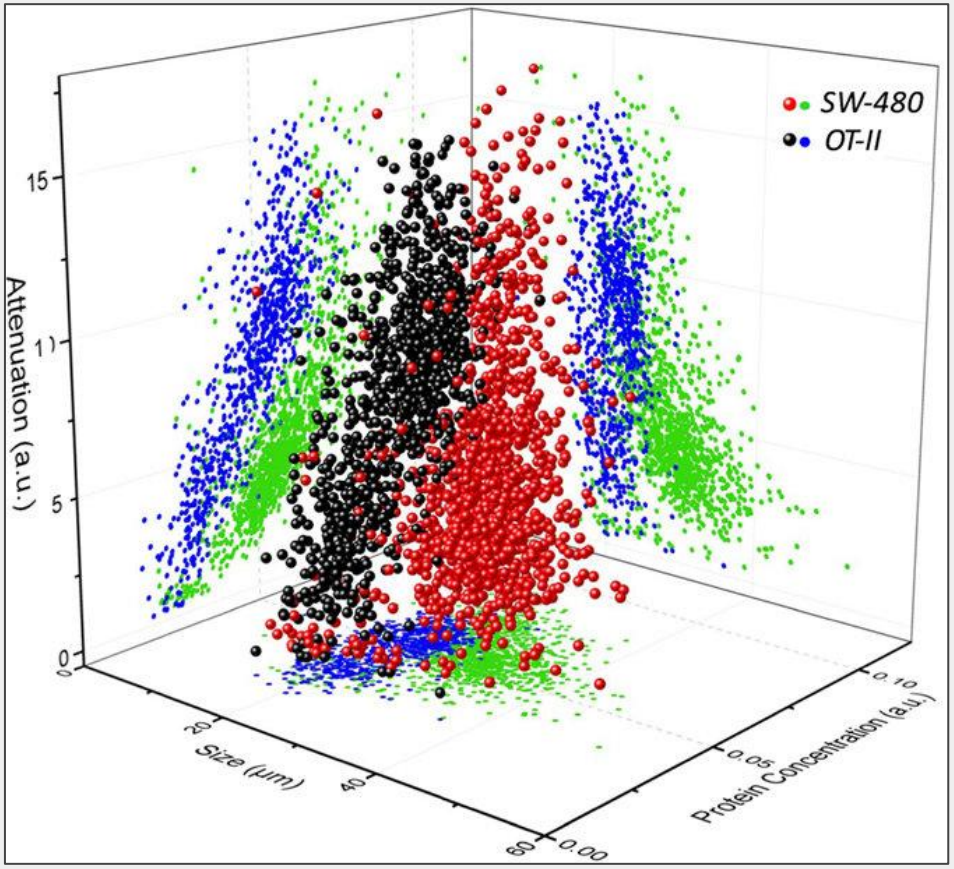


How accurate is this decision?
How do you define the decision?

DistrictDataLabs

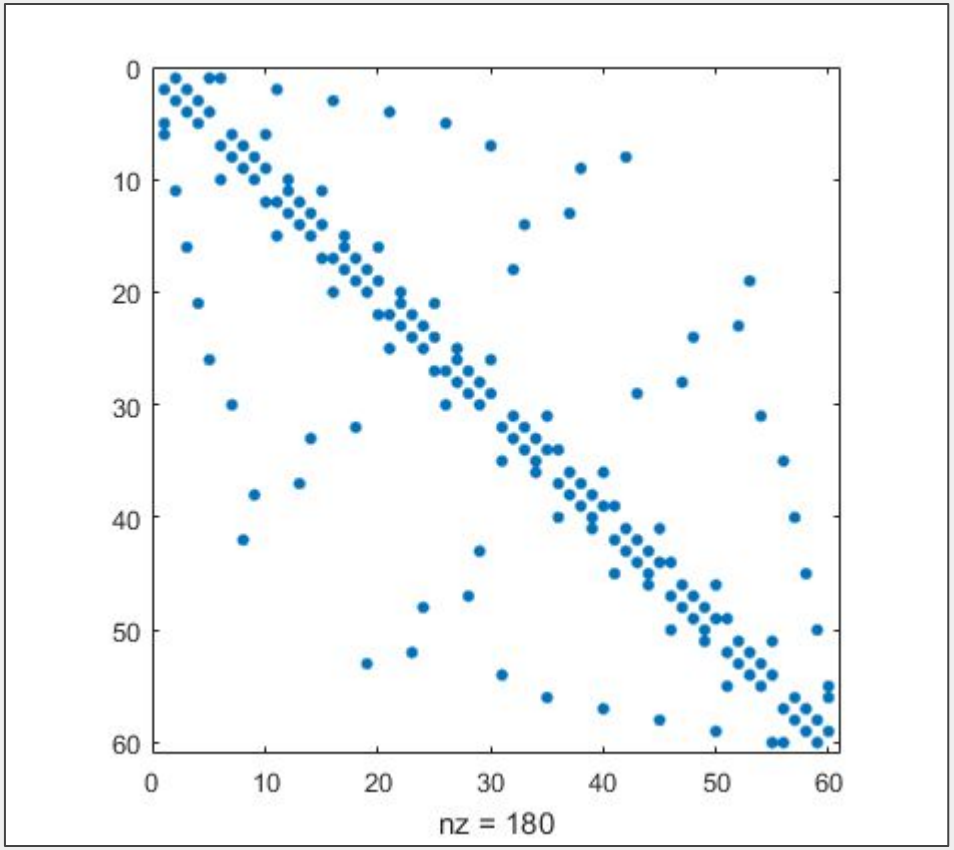
Separability

- Models are able to easily distinguish patterns in feature space.
- Patterns can be “separated” with a hyperplane or region
- Decreased covariance between features that describe instances.
- Loosely: “noiseless”.



Generalizability

- Models are able to make predictions on new inputs.
- Complete view of the decision region, no narrow windows of decision making or limited information gain (sparsity).
- Loosely: “model is sufficient”



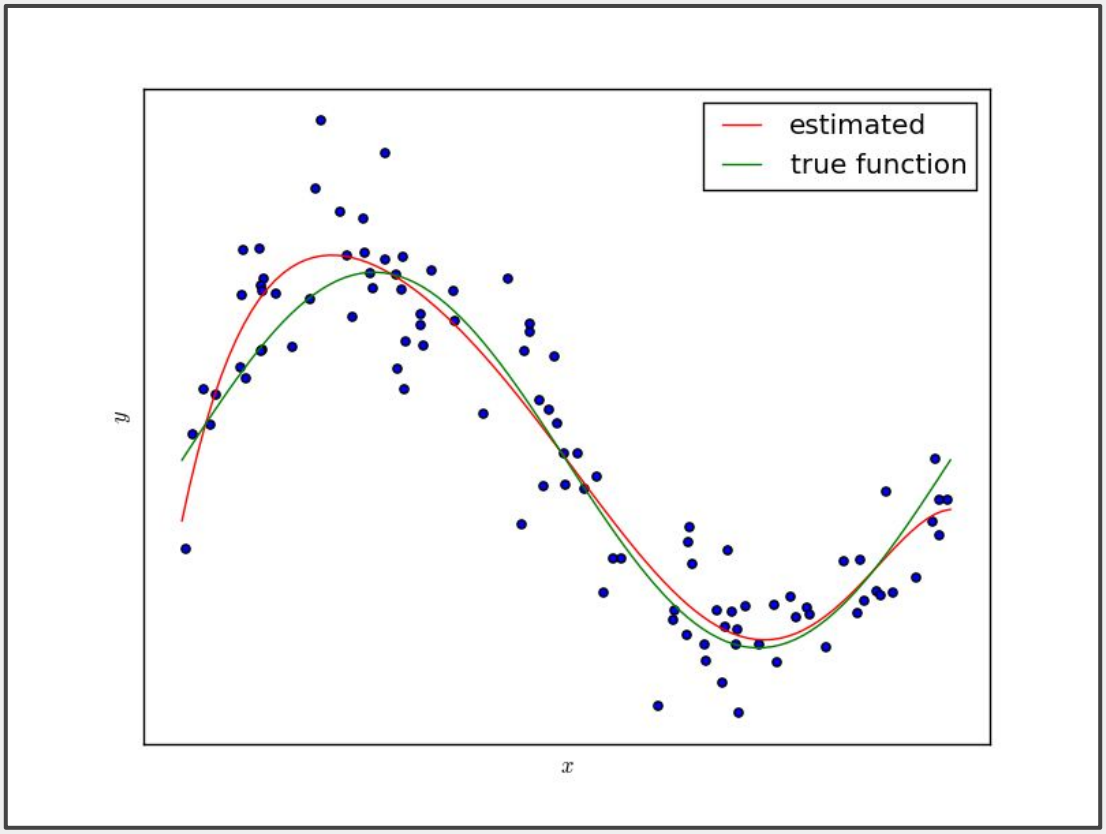
Types of Algorithms by Output

Input *training* data to *fit* a model which is then used to *predict* incoming inputs into ...

Type of Output	Algorithm Category
Output is one or more discrete classes	**Classification (supervised)**
Output is continuous	**Regression (supervised)**
Output is membership in a similar group	**Clustering (unsupervised)**
Output is the distribution of inputs	Density Estimation
Output is simplified from higher dimensions	Dimensionality Reduction

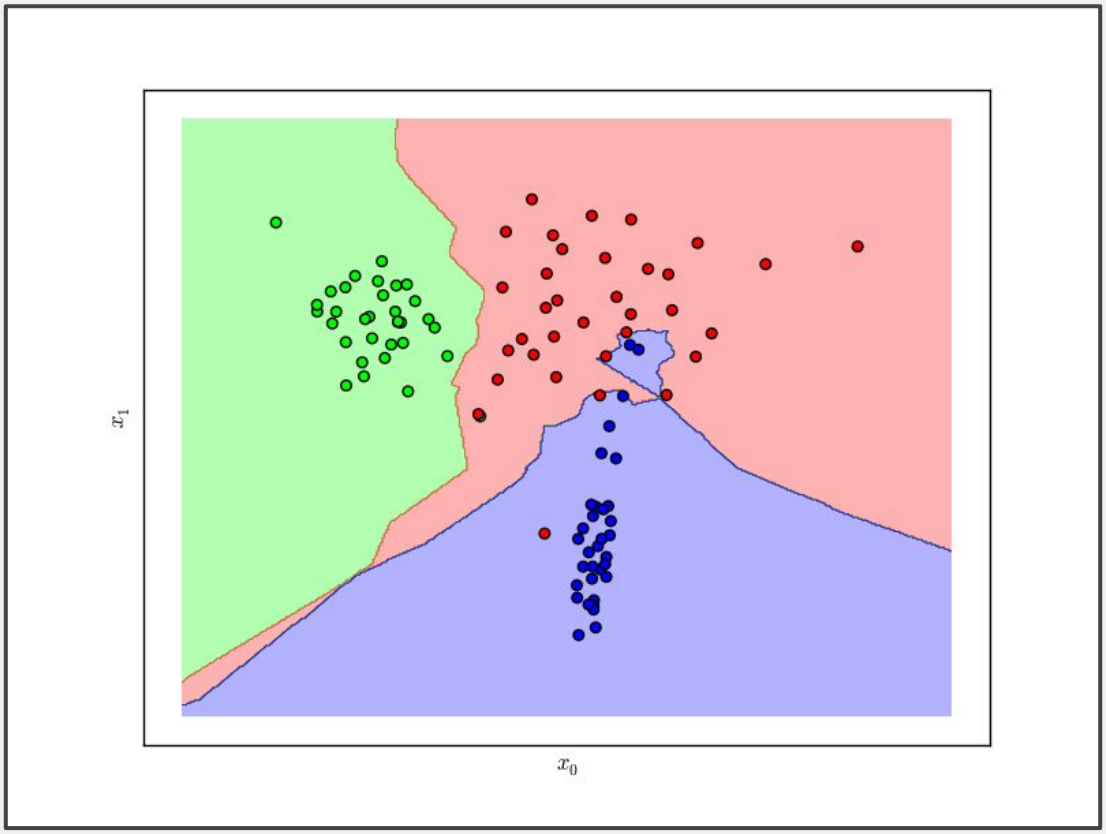
DistrictDataLabs

Regression



Given continuous input data fit a function that is able to predict the continuous value of input given other data.

Classification



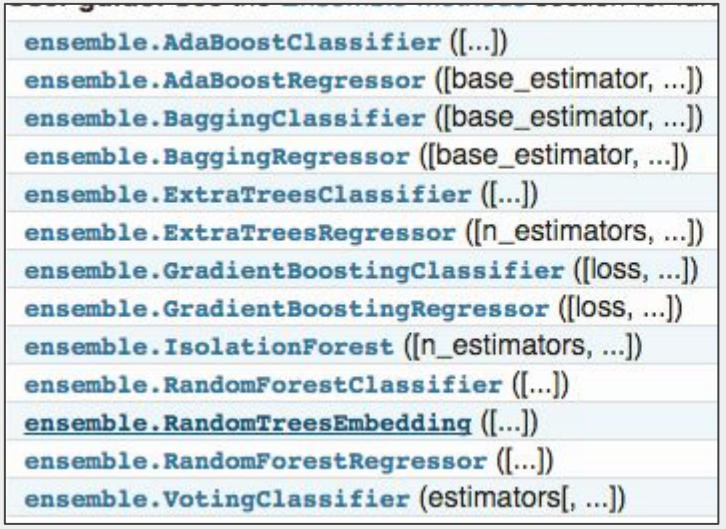
Given labeled input data (with two or more labels), fit a function that can determine for any input, what the label is.

Supervised Learning with Scikit-Learn

Regression and Classification are two sides of the same coin.

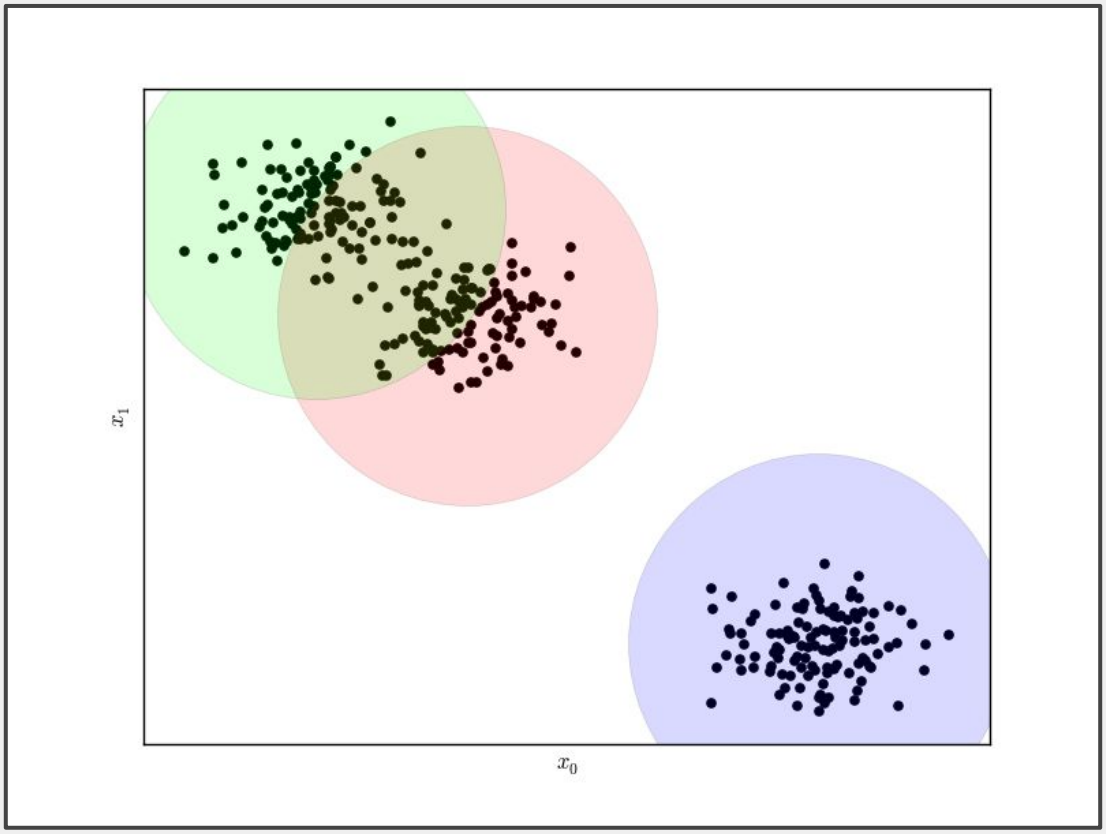
Many Scikit-Learn Regressors have an associated reciprocal Classifier and vice versa.

Often a solution to regression problems or investigations is to discretize into a classifier.



DistrictDataLabs

Clustering (Side Note)



Given data, determine a pattern of associated data points or clusters via their similarity or distance from one another.

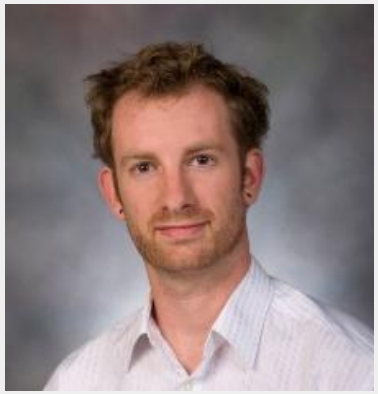
Hadley Wickham (2015)

"Model" is an overloaded term.

• **Model family** describes, at the broadest possible level, the connection between the variables of interest.

• **Model form** specifies exactly how the variables of interest are connected within the framework of the model family.

• A **fitted model** is a concrete instance of the model form where all parameters have been estimated from data, and the model can be used to generate predictions.



http://had.co.nz/stat645/model-vis.pdf

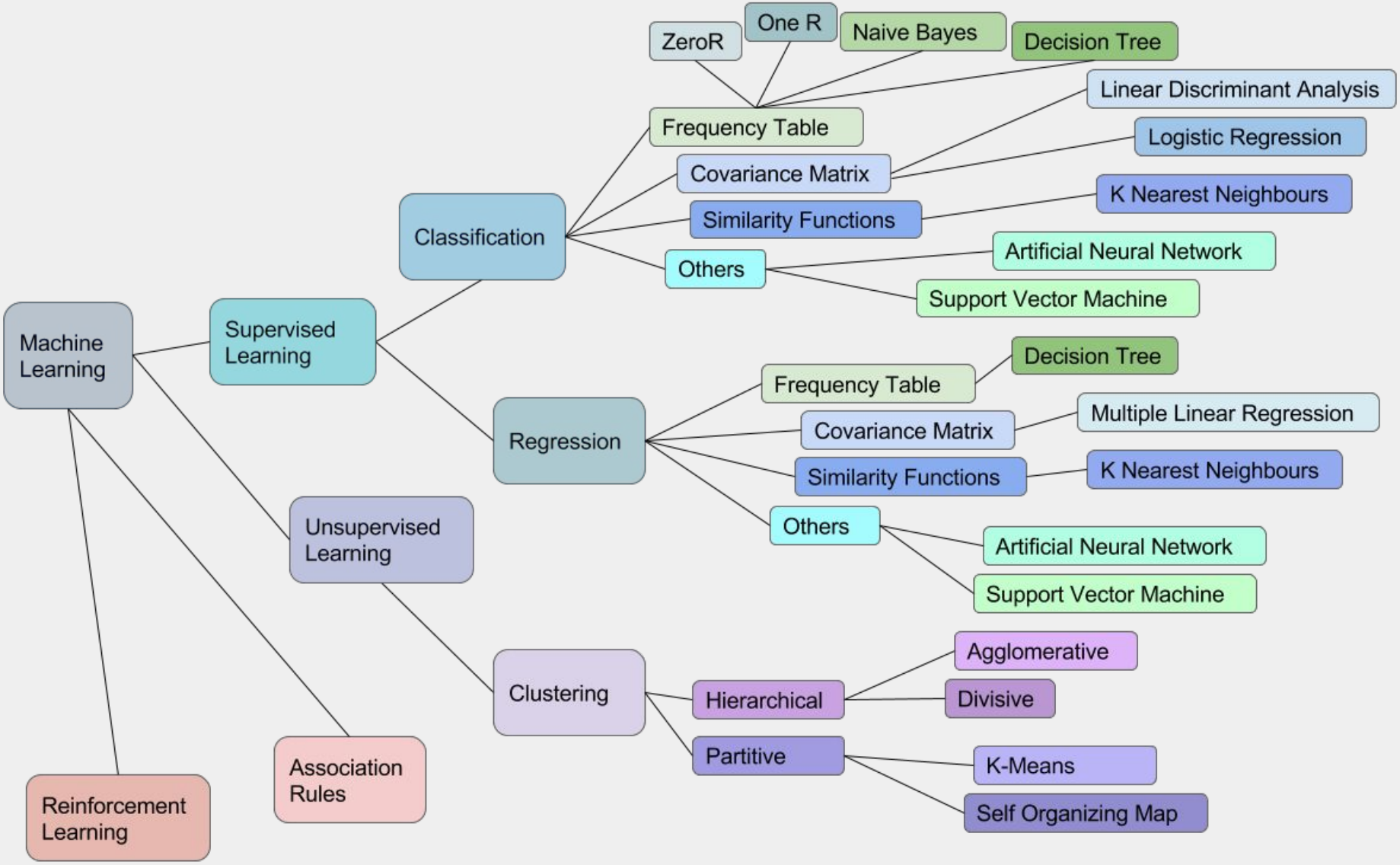
DistrictDataLabs

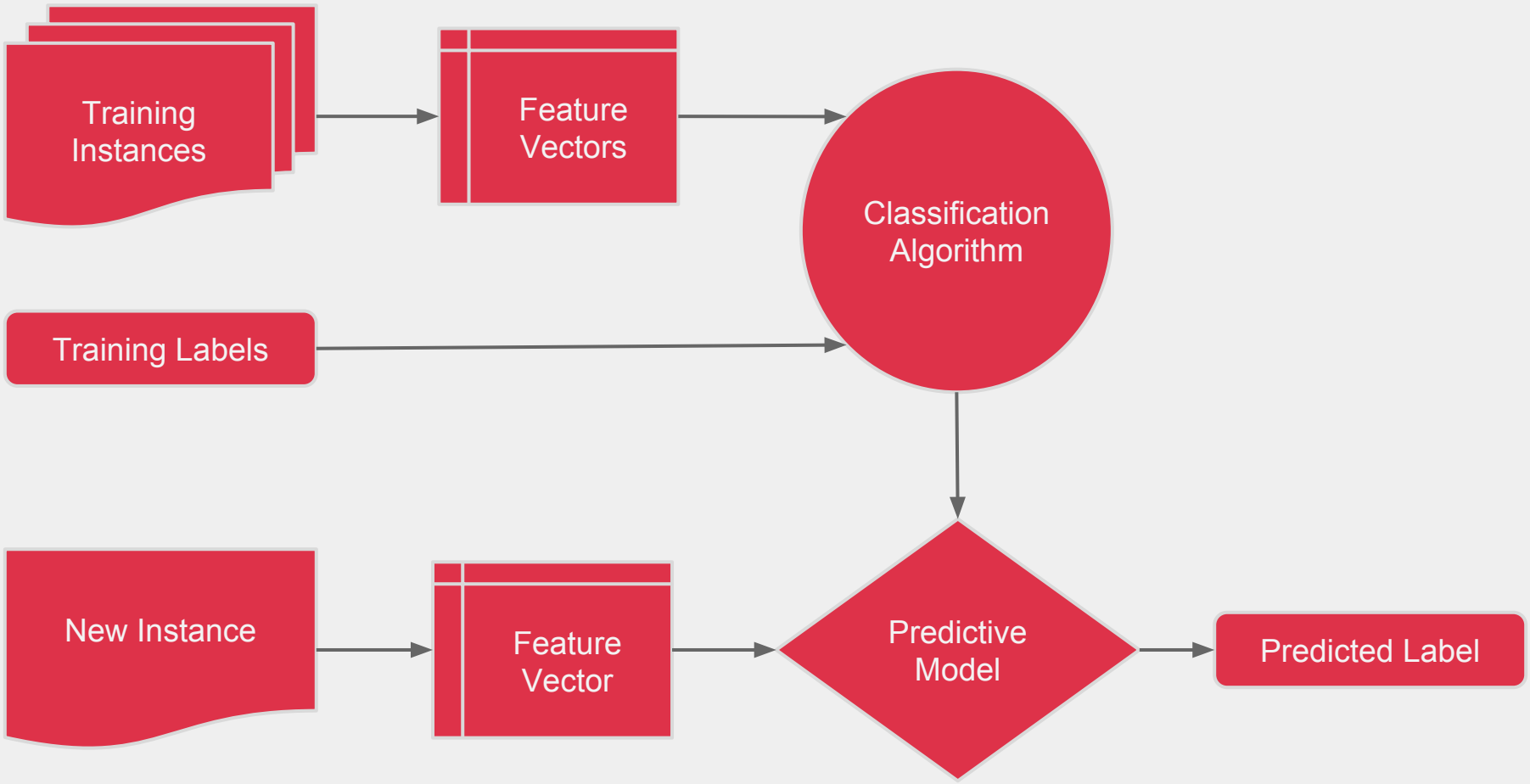
Your Task

Given a data set of instances of size *N*, create a ***model form*** that is fit from the data (built) by extracting features and dimensions. Use that ***model form*** to create a ***fitted model*** and predict outcomes on new data …

1. Data Wrangling (normalization, standardization, imputing)
2. Feature Analysis/Extraction
3. Model Selection/Building
4. Model Evaluation
5. Operationalize Model

DistrictDataLabs


Machine Learning
Supervised Learning
Classification
ZeroR
One R
Naive Bayes
Decision Tree
Frequency Table
Linear Discriminant Analysis
Logistic Regression
Covariance Matrix
K Nearest Neighbours
Similarity Functions
Artificial Neural Network
Others
Support Vector Machine
Regression
Frequency Table
Decision Tree
Multiple Linear Regression
Covariance Matrix
K Nearest Neighbours
Similarity Functions
Artificial Neural Network
Others
Support Vector Machine
Unsupervised Learning
Clustering
Hierarchical
Agglomerative
Divisive
Partitive
K-Means
Self Organizing Map
Association Rules
Reinforcement Learning



Classification Pipeline

DistrictDataLabs

Class Problem – Labels

Binary Classification

- yes/no
- true/false
- 1/0

Assumed that P(1) = 1-P(0)

`Binarizer` - sets target to 0 or 1 based on a threshold value.

Multi-Class Classification

- A, B, or C
- 0, 1, 2, 3

Assumed that classes are independent. Often composed of many binary classification problems.

`LabelEncoder` - convert target to numeric values.

DistrictDataLabs

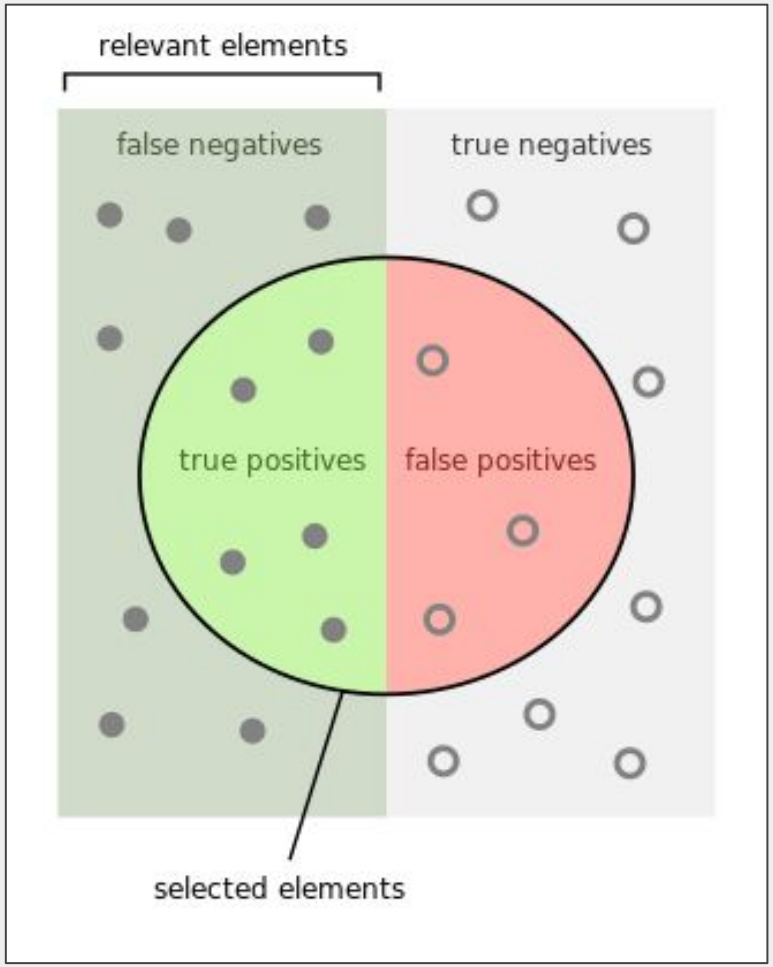
Evaluation

Because classification is supervised (we have the correct answers). We can evaluate performance via cross validation.

Type I error: false positive

Type II error: false negative

Recall (sensitivity): portion of instances that are retrieved.

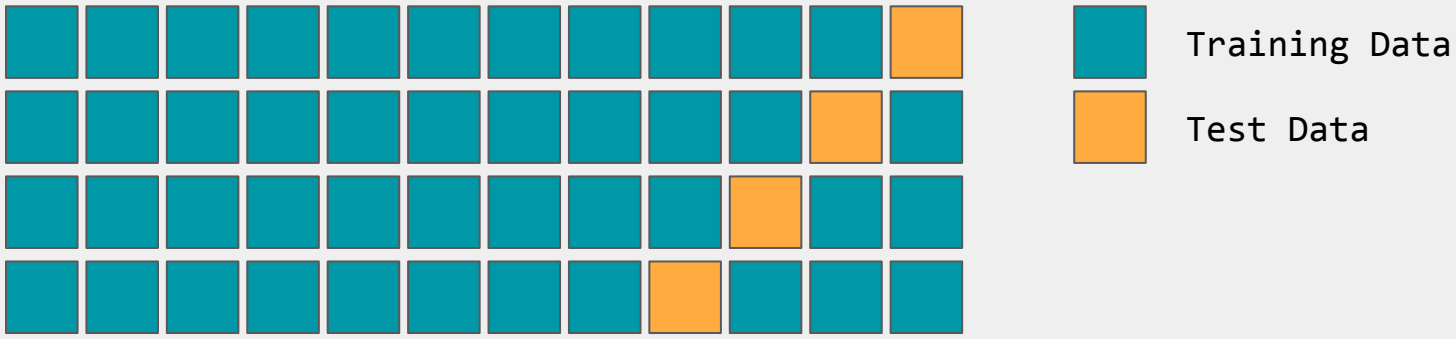


DistrictDataLabs

Cross Validation (classification)

Assess how model will generalize to independent data set (e.g. data not in the training set).

1. Divide data into training and test splits
2. Fit model on training, predict on test
3. Determine *accuracy*, *precision* and *recall*
4. Repeat *k* times with different splits then average as *F1*



DistrictDataLabs

Classification Metrics & Confusion Matrix

$$precision = \frac{A_{true}}{A_{true} + A_{false}}$$

$$accuracy = \frac{A_{true} + B_{true}}{total}$$

$$recall = \frac{A_{true}}{A_{true} + B_{false}}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

	Predicted Class A	Predicted Class B	
Actual A	True A	False B	#A
Actual B	False A	True B	#B
	#P(A)	#P(B)	total

DistrictDataLabs

```
from sklearn import metrics
from sklearn import cross_validation as cv

splits     = cv.train_test_split(X, y, test_size=0.083)
X_train, X_test, y_train, y_test = splits

model      = ClassifierEstimator()
model.fit(X_train, y_train)

expected   = y_test
predicted  = model.predict(X_test)

print metrics.classification_report(expected, predicted)
print metrics.confusion_matrix(expected, predicted)
print metrics.f1_score(expected, predicted)
```

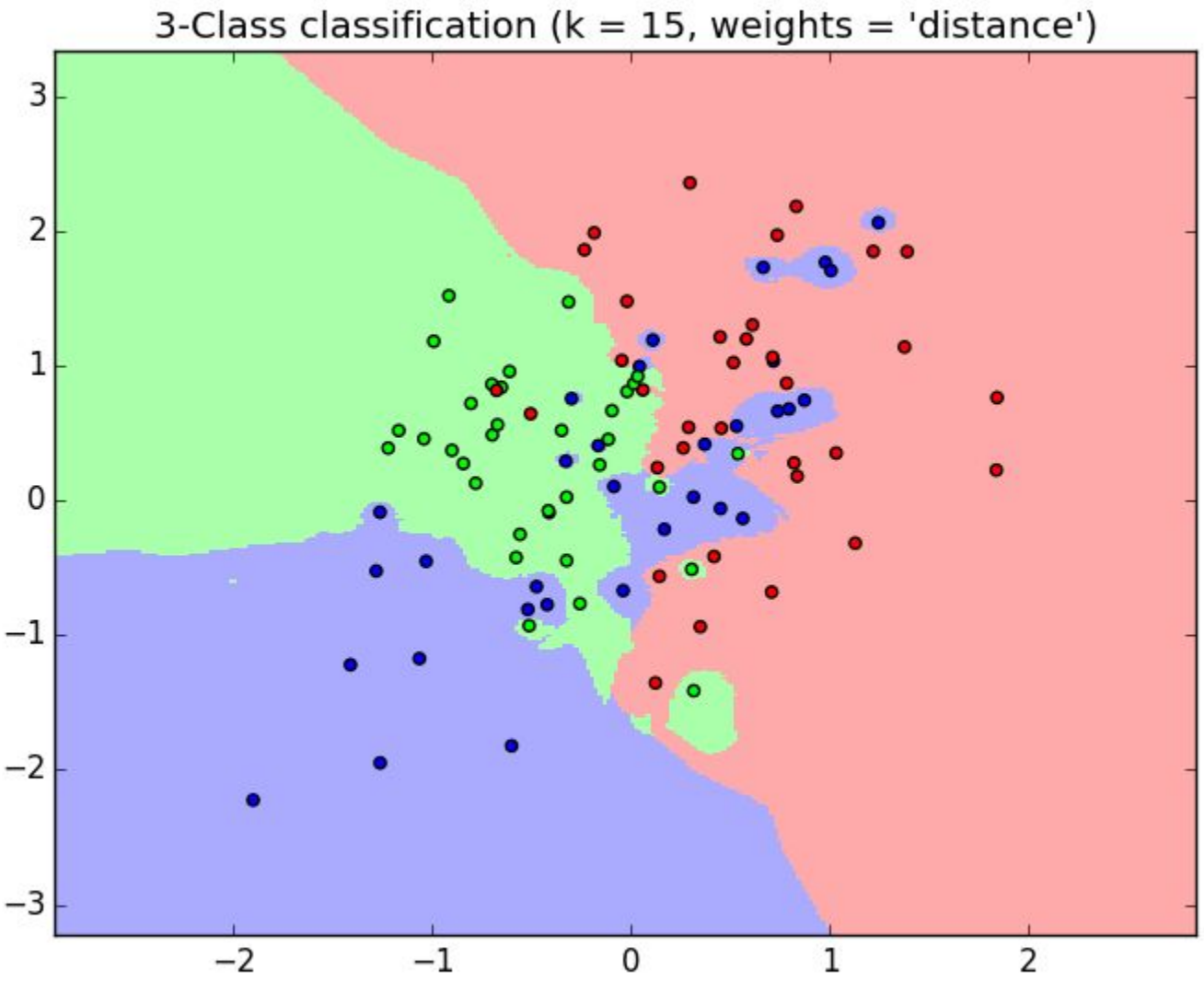
k-Nearest Neighbors

DistrictDataLabs

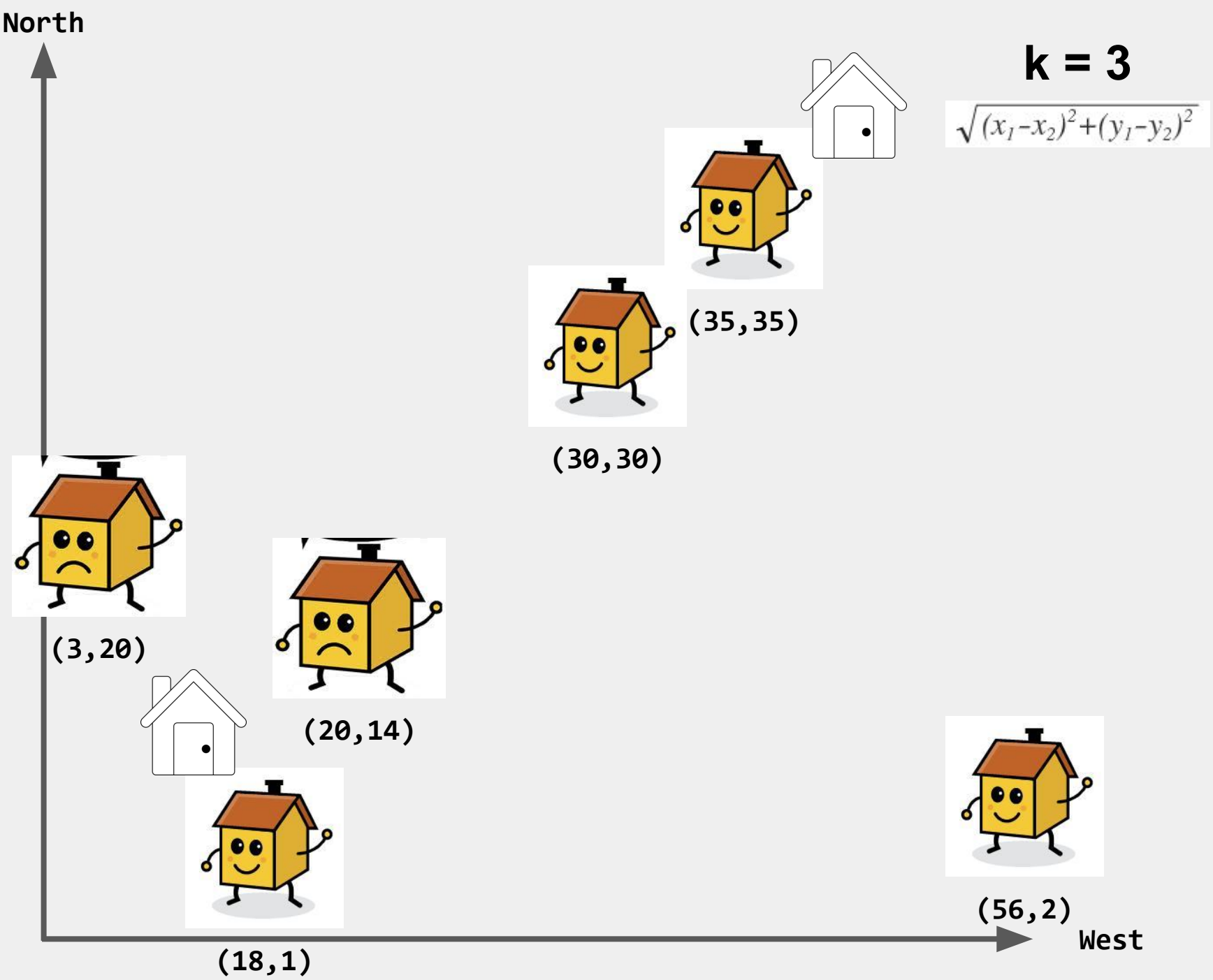
K Nearest Neighbors (kNN)

- In Scikit-Learn the `neighbors` package provides both supervised and unsupervised nearest neighbors methods - these are simple and effective!
- Find a predefined (k) samples closest in distance to the input point and predict the label from those.
- Distance == similarity in this case, and can be any metric measure. Euclidean distance is most common.
- Non-generalizing: must “remember” all training data.

DistrictDataLabs



DistrictDataLabs

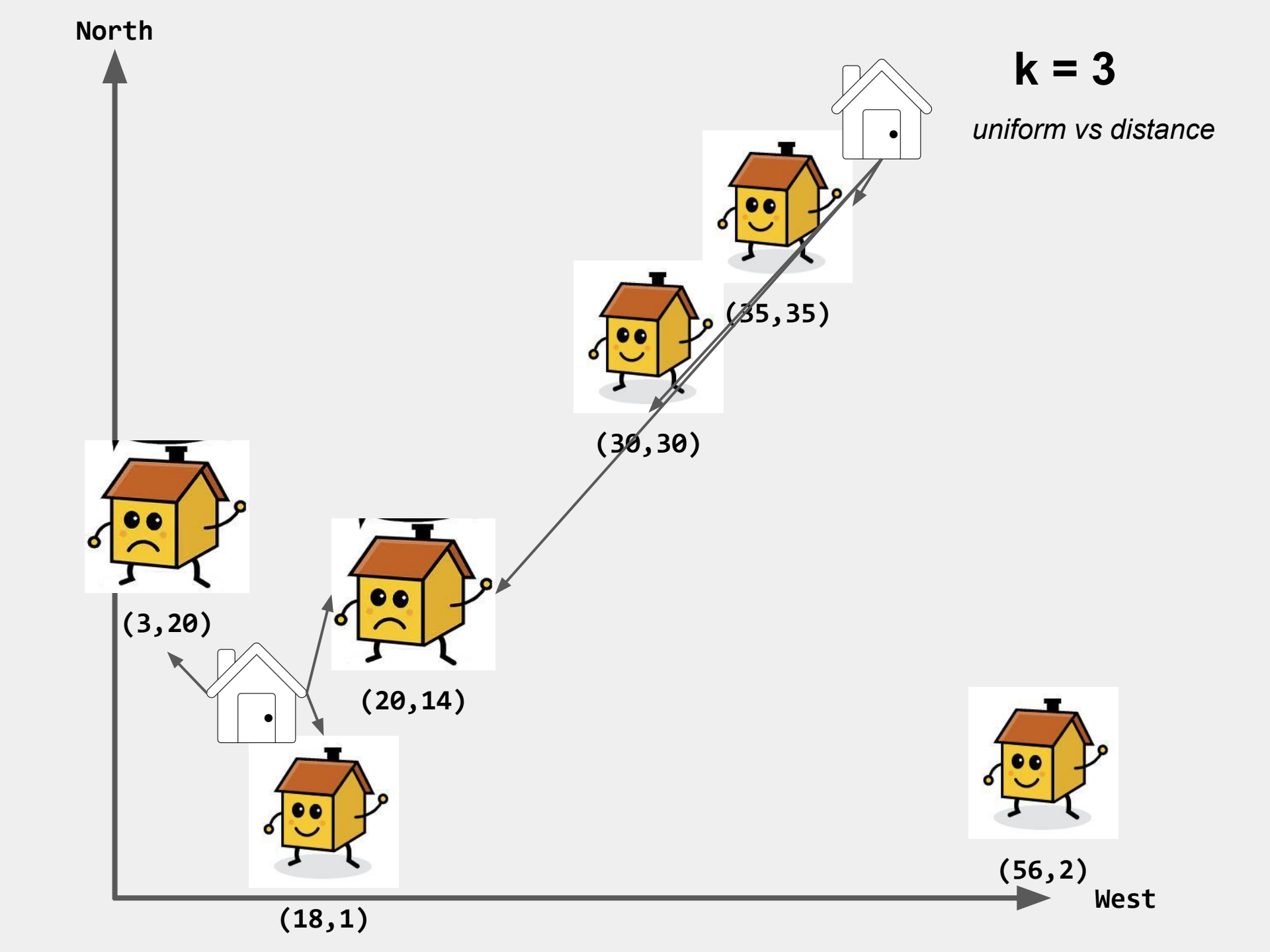

North
k = 3
$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$
(35,35)
(30,30)
(3,20)
(20,14)
(18,1)
(56,2)
West

```
from sklearn.metrics.pairwise import
euclidean_distances

X = [[3,20],[20,14],[18,1],[30,30],[35,35],[56,2]]
y = [[10,10],[40,40]]

euclidean_distances(X,y)

array([[ 12.20655562,  42.05948169],
       [ 10.77032961,  32.80243893],
       [ 12.04159458,  44.77722635],
       [ 28.28427125,  14.14213562],
       [ 35.35533906,   7.07106781],
       [ 46.69047012,  41.23105626]])
```


North
k = 3
uniform vs distance
(35,35)
(30,30)
(3,20)
(20,14)
(18,1)
(56,2)
West

Pros and Cons of kNN

Pros

- Works well with distance-sensitive data
- Simple and effective for a wide array of tasks
- Non parametric - works well with irregular decision boundaries.
- Often a first approach
- No worse than 2x Bayes error rate as data $\rightarrow \infty$

Cons

- kNN can be very arbitrary and requires hand-holding to model.
- How do you choose k?
- What is the definition of neighbor/similarity?
- Suffers from the curse of dimensionality.

DistrictDataLabs

Choosing K

K is an arbitrary number between 1 and the number of instances in the data set. Three options for choosing:

1. Guess
2. Heuristic
3. Optimization

Tips:

- Avoid an even k with only 2 classes (tie break)
- Choose k >= # classes + 1
- Choose as low a k as possible

DistrictDataLabs

Use coprime class and k

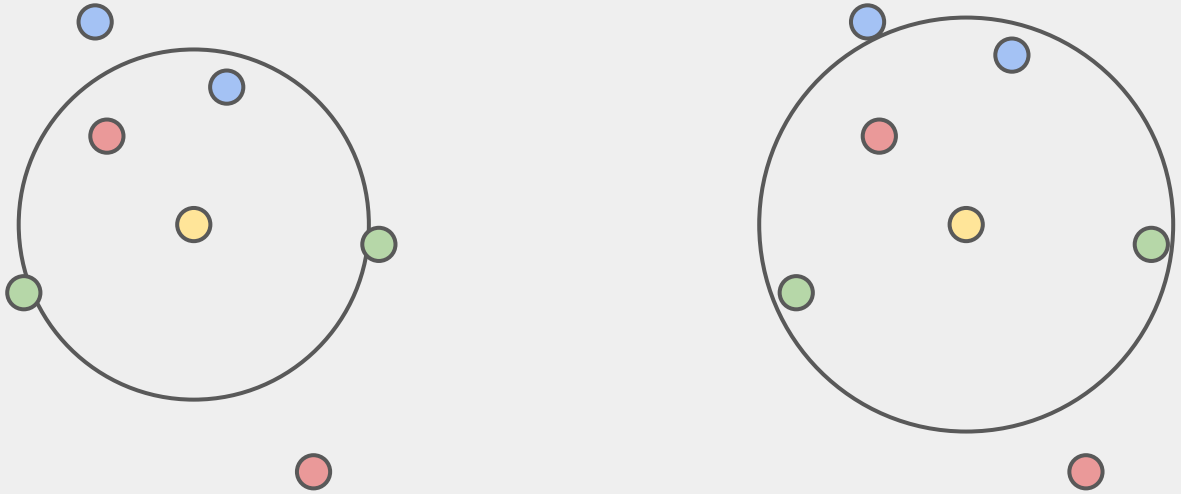
Coprime numbers don't share common divisors except for 1. So 4 and 9 are coprime but not 3 and 9.

Classes = 2 (3 red, 3 green)
k = 6 and k=5 respectively

DistrictDataLabs

Use number of classes + 1

With k < # of classes, there is no chance that all classes will be represented - but we still want to prevent ties.



Classes = 3 (2 red, 2 green, 2 blue)
k = 2 and k=4 respectively

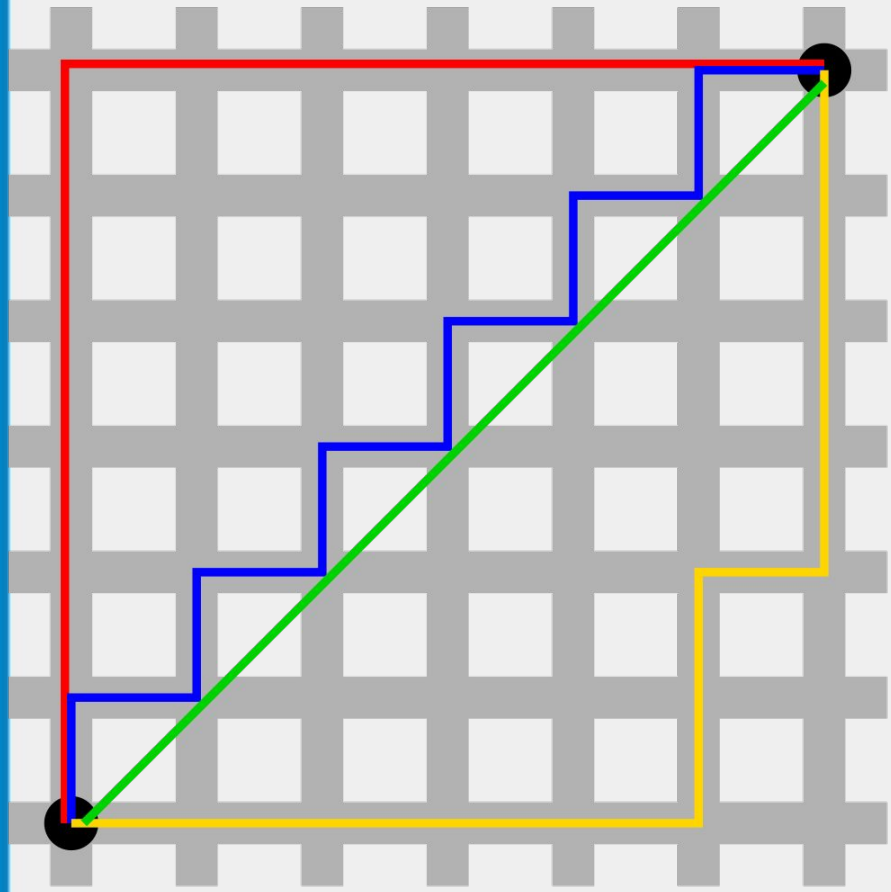
DistrictDataLabs

Optimization

- Many people assert k should be chosen via *domain knowledge* - as K increases, the *complexity* increases and slows down performance.
- Minimize error by trying different values of k = Hill Climbing problem. Algorithms: Genetic parameter optimization and simulated annealing.
- Iterate 2x through 1% of the data with a variety of K and compute error curve for minimization.

DistrictDataLabs

Distance Measures I



Taxicab (Manhattan) Distance is the Red, Yellow, and Blue lines.

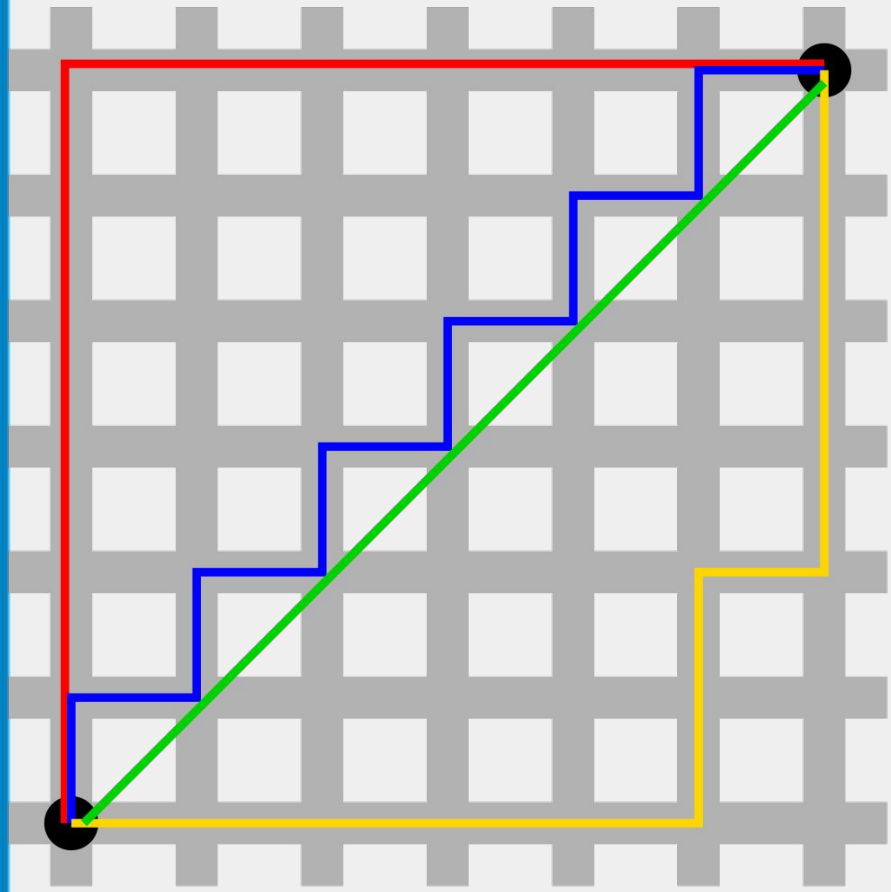
$D_{manhattan} = ?$

Green line is Euclidean distance.

$D_{euclidean} = ?$

DistrictDataLabs

Distance Measures I



Taxicab (Manhattan) Distance is the Red, Yellow, and Blue lines.

$D_{manhattan} = 12$

Green line is Euclidean distance.

$D_{euclidean} = 6\sqrt{2} = 8.49$

DistrictDataLabs

Minkowski Distance

Can we generalize taxicab and euclidean distance?

$$d_{manhattan}(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

$$d_{euclidean}(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{\frac{1}{2}}$$

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

DistrictDataLabs

Mahalanobis Distance

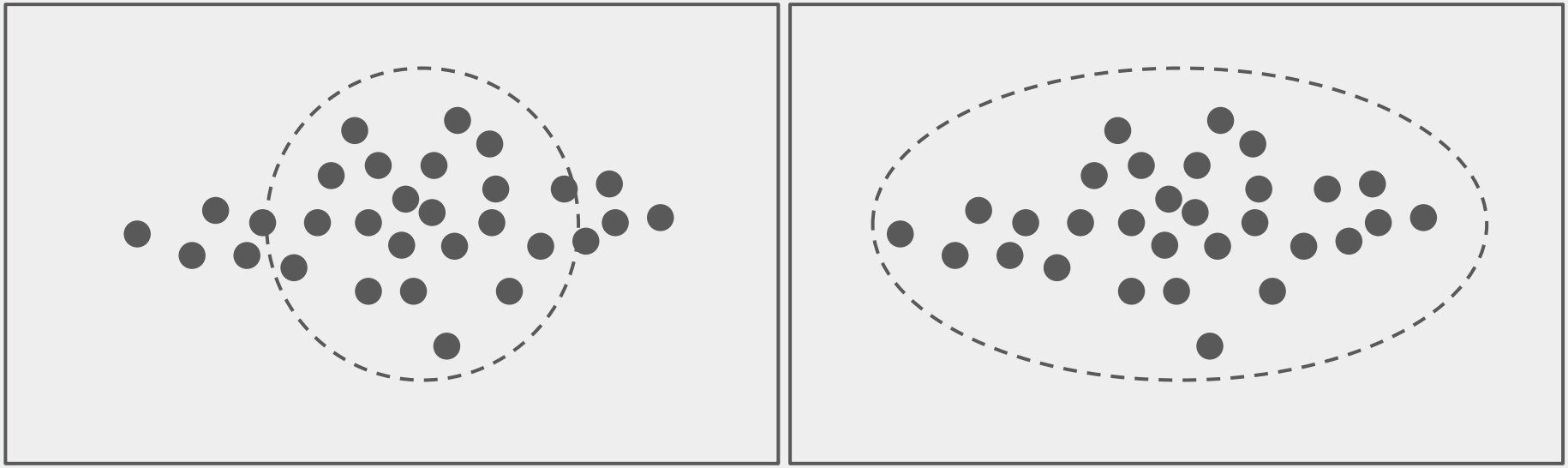
Minkowski distance assumes all data is symmetric (distance is the same in all dimensions).

Mahalanobis considers the volatility of each dimensions; creating a variable s_i for each dimension, the standard deviation of that set.

$$d(x,y)=\sqrt{\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{s_i^2}}$$

DistrictDataLabs

Mahalanobis Distance

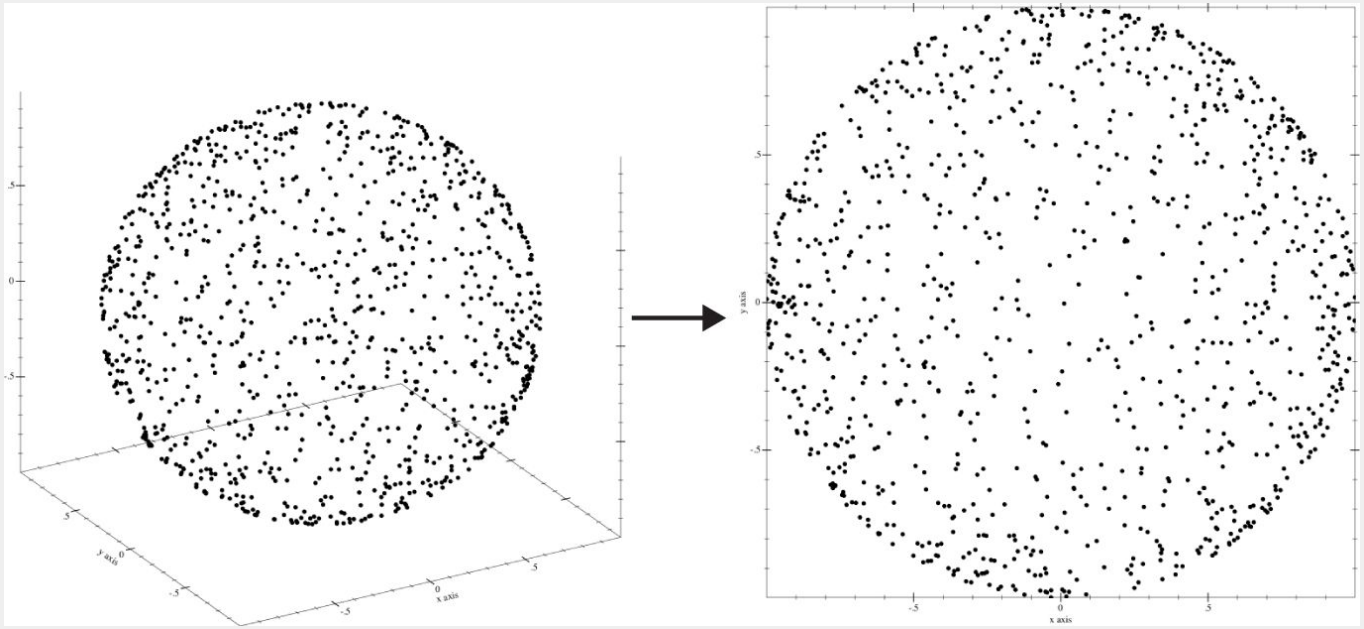


DistrictDataLabs

Curse of Dimensionality

High dimensional data tends to be sparse and far apart.

Algorithms that are based in locality need to determine nearness, but more or less dimensions can skew results.



DistrictDataLabs

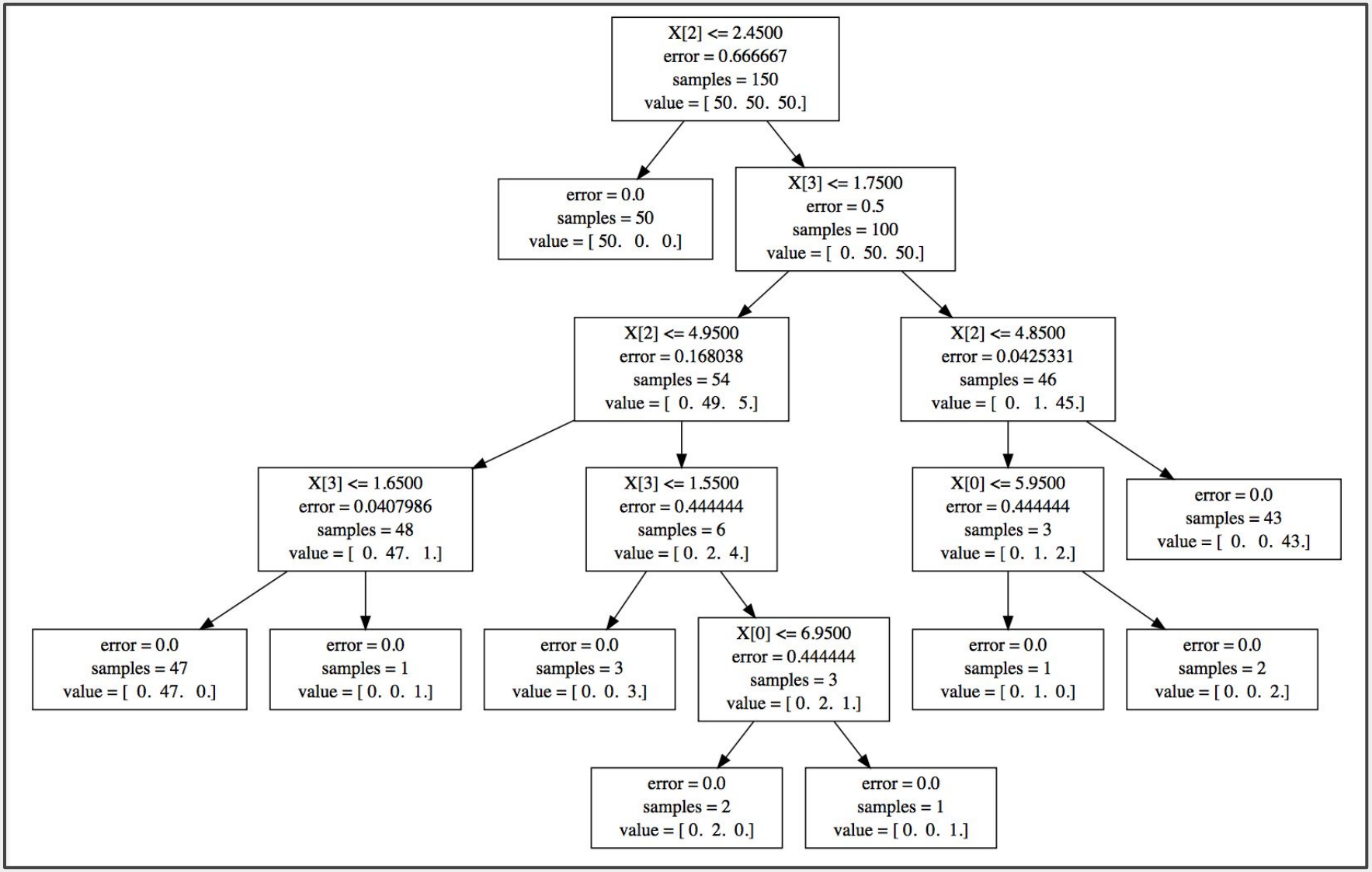
Decision Trees

DistrictDataLabs

Decision Trees

- In Scikit-Learn the `tree` package provides classifiers and regressors based on a decision tree model.
- Another example of non-parametric inductive learning - but one that generalizes better.
- Creates a graphical model of rules that partitions the data until a decision is reached at one of the leaf nodes.
- Complexity is related to the amount of data and the partitioning method.

DistrictDataLabs



Decision Tree example from Iris Data Set

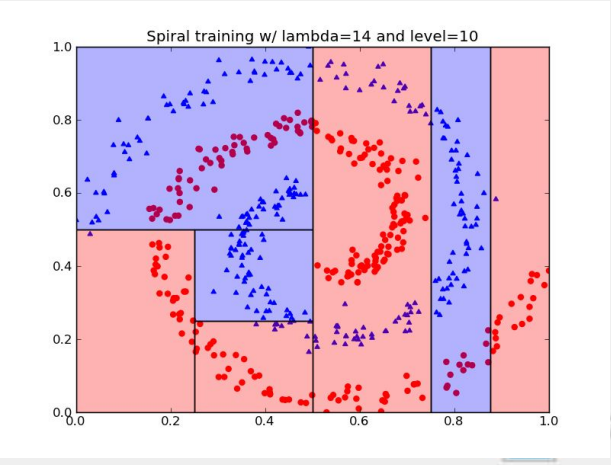
DistrictDataLabs

Decision Tree Algorithm

Partition data as follows:

1. Start with whole training data
2. Select attribute along dimension that gives “best” split
3. Create child nodes based on split
4. Recurse on each child using child data until stop

What is the “best” split?



trictDataLabs

Split Metrics

- **Gini Impurity (used by CART)**

Measures how often randomly chosen elements would be incorrectly labeled according to distribution in partition. Zero value means all cases fall into single target

- **Information Gain (ID3 and C5)**

Uses entropy - the amount of information contained in every partition that represents some part of data.

- **Variance Reduction (CART)**

Maximally select the split where the most variance would be reduced, used for continuous values of data.

DistrictDataLabs

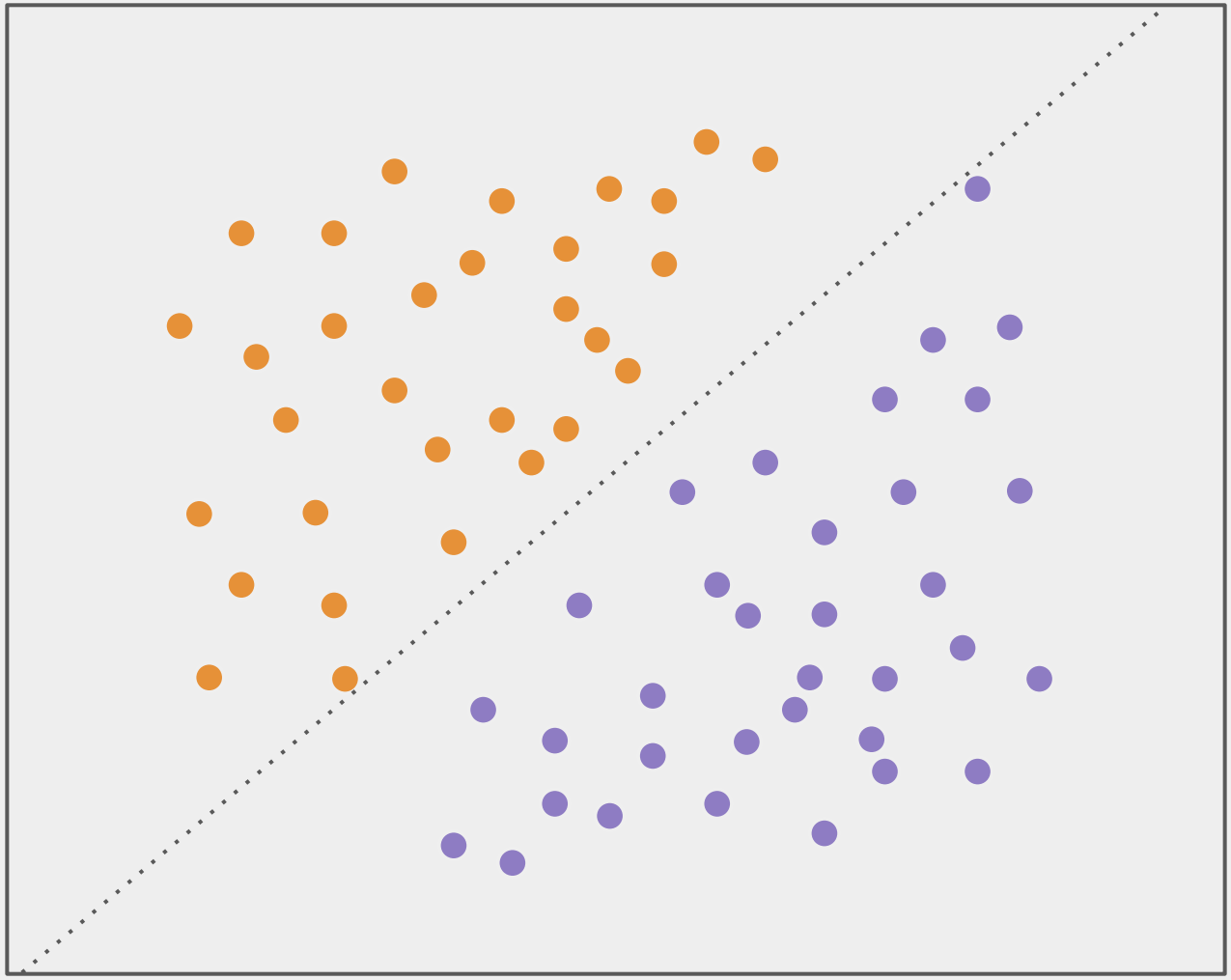
Gini

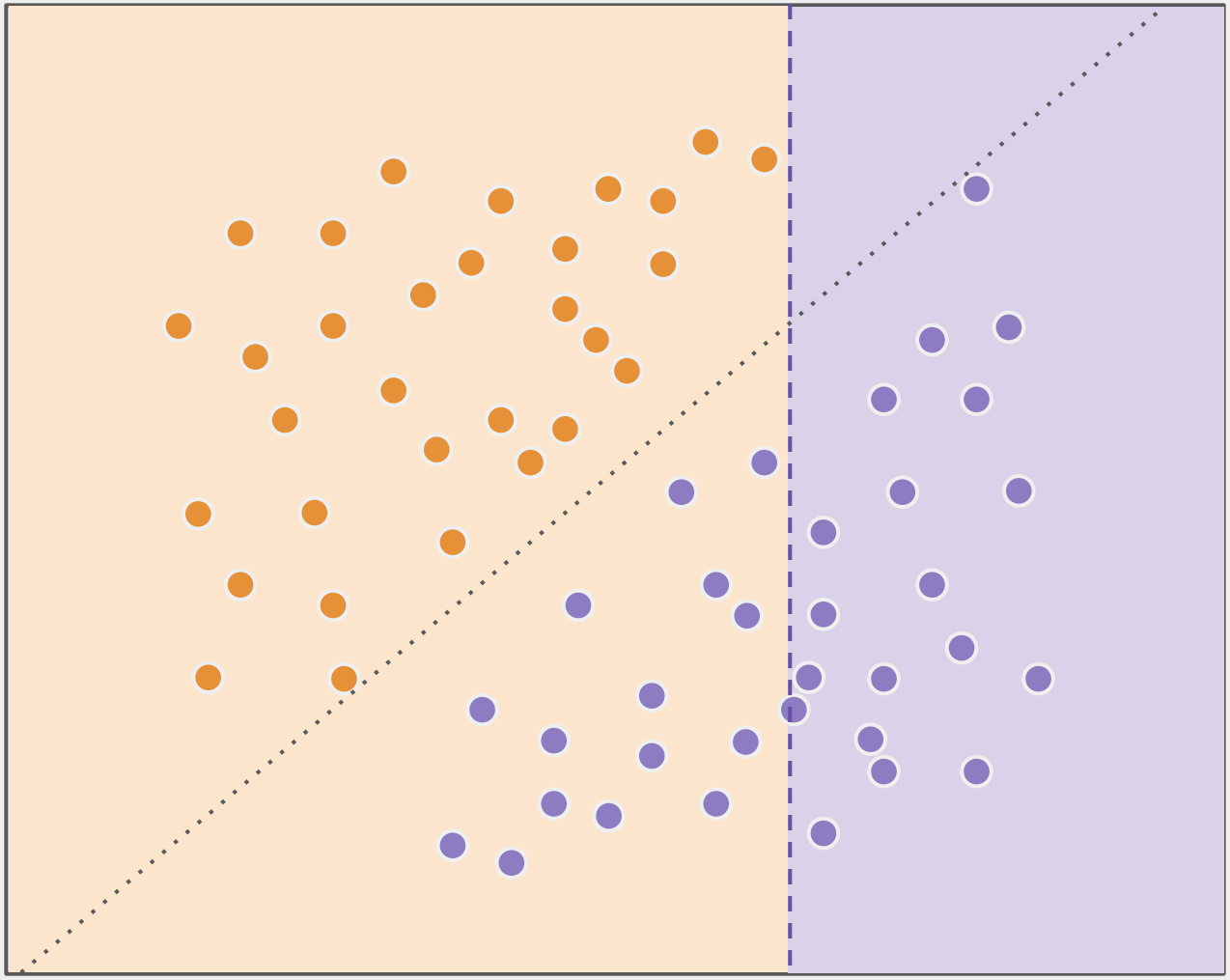
For a set of items where i $\in$ {1,2, ..., m} and f_i is the proportion of items in the split labeled i, the Gini Impurity is computed as follows:

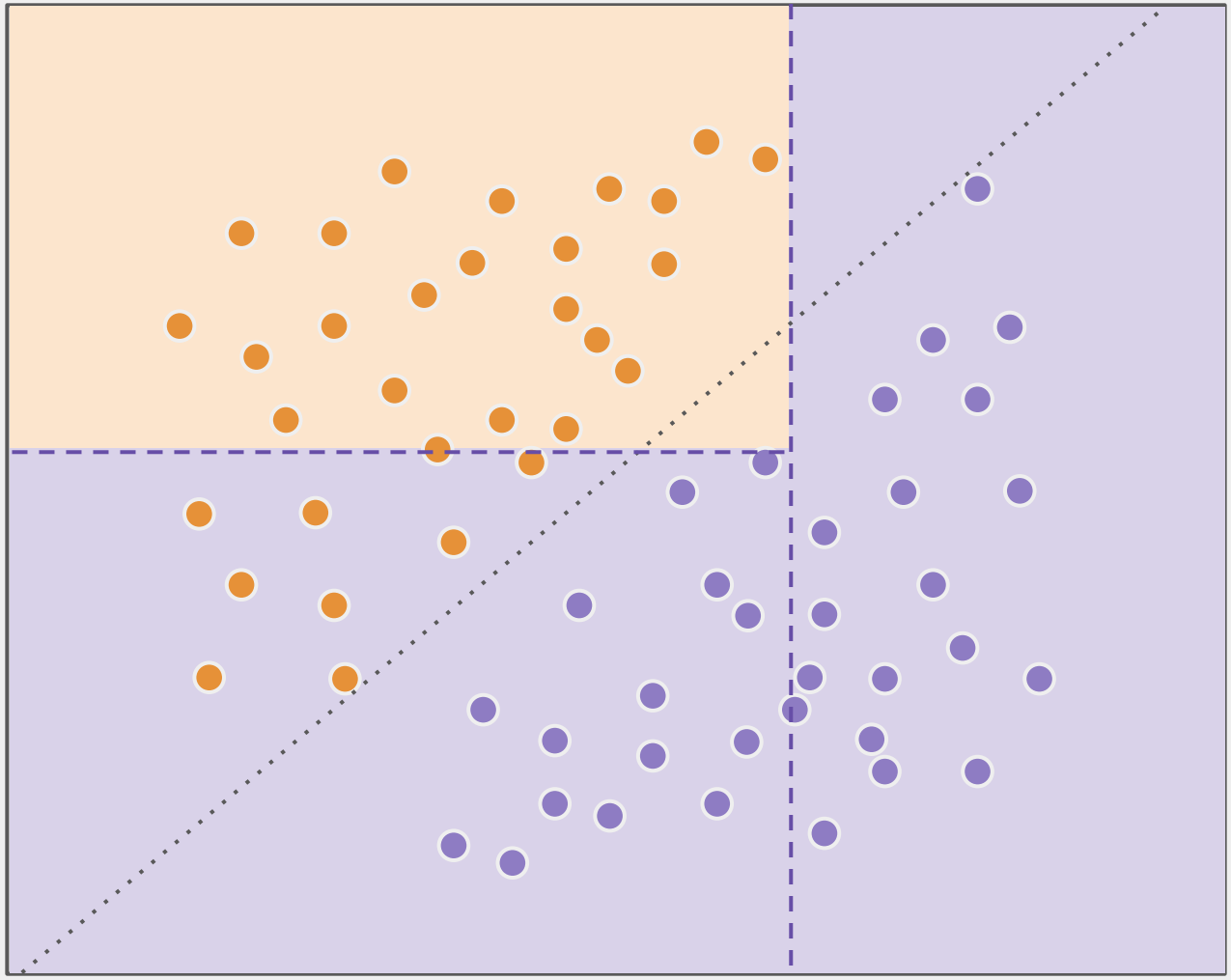
$$I_G(f) = 1 - \sum_{i=1}^{m} f_i^2$$

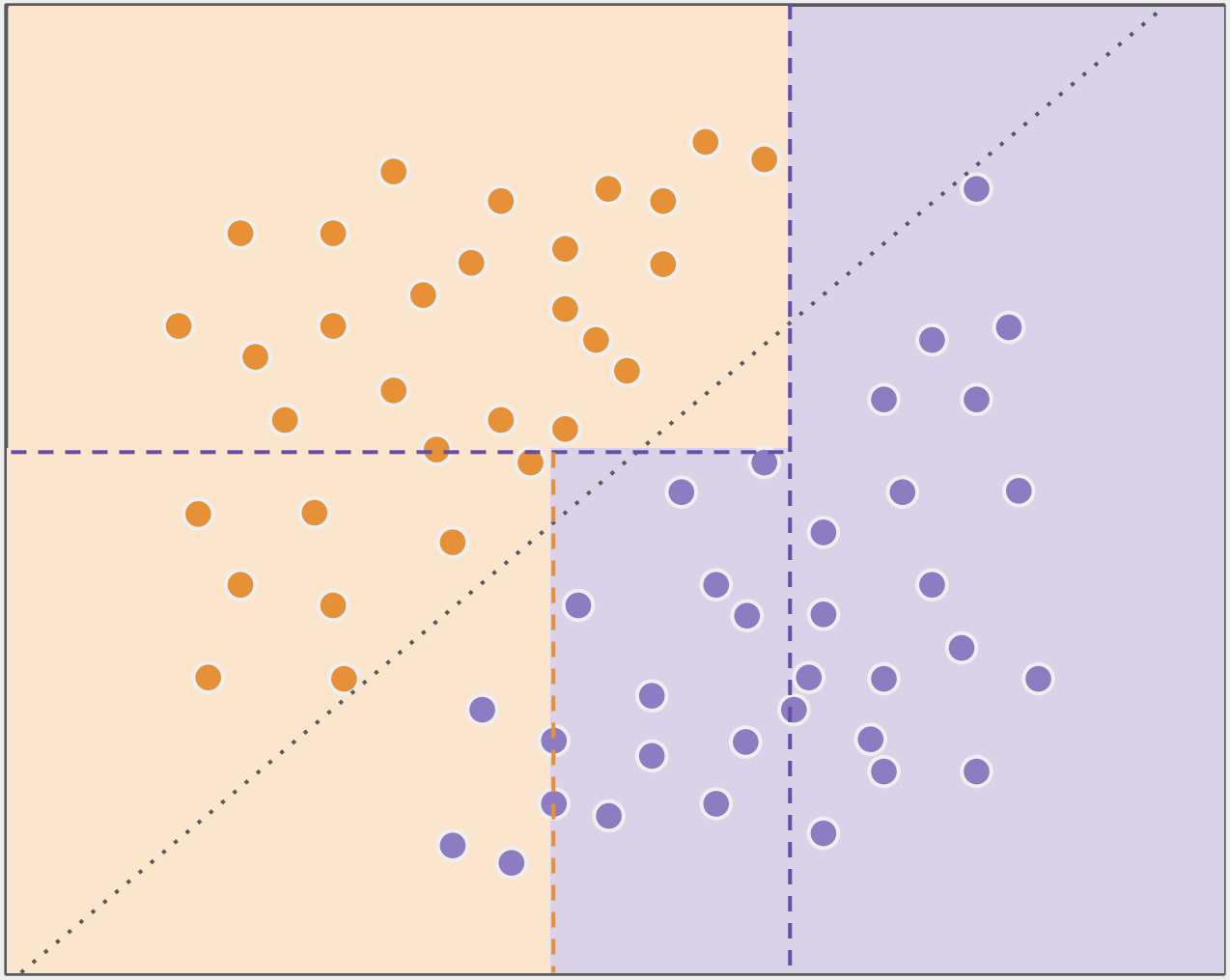
Essentially the sum of the probability of an item being chosen times the probability of a mistake being made.

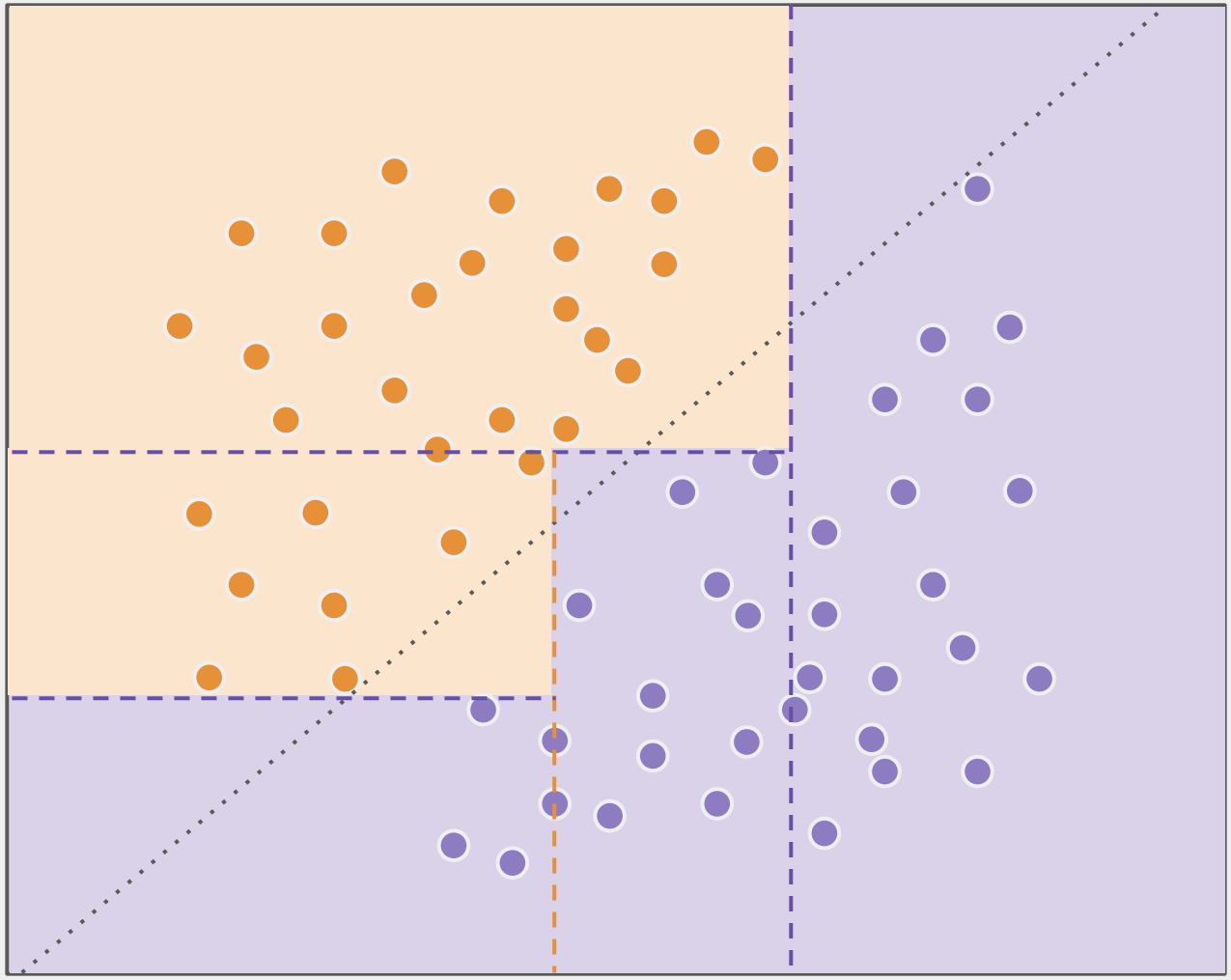
DistrictDataLabs











Pros and Cons of Decision Trees

Pros

- Simple to understand; no representational mysticism. Trees can be drawn out.
- No need to normalize or standardize (need to deal with missing values).
- Low cost
- Handle multi-output
- Validate with Statistics!
- Handles wrong assumptions.

Cons

- Prone to overfit (deep trees don’t generalize)
- Minor variations in data cause big changes in tree structure (unstable) - fix with ensemble methods.
- Create biases if some classes dominate
- Some functions are impossible to model

DistrictDataLabs

Tips for using Decision Trees

- Ratio of samples to number of features is important, trees in high dimensional space is very likely to overfit.
- Perform dimensionality reduction to give tree a better chance of finding features that are discriminative (PCA, ICA, Feature Selection)
- Visualise your tree as you are training by using the export function. Use max_depth=3 as an initial tree depth to get a feel for how the tree is fitting to your data, and then increase the depth.
- Remember that the number of samples required to populate the tree doubles for each additional level the tree grows to. Use max_depth to control the size of the tree to prevent overfitting.
- Balance your dataset before training to prevent the tree from creating a tree biased toward the classes that are dominant.

DistrictDataLabs

Tips for using Decision Trees

Or, just use a Random Forest...

```
from sklearn.ensemble import RandomForestClassifier

splits = cv.train_test_split(digits.data,digits.target,
test_size=0.2)
X_train, X_test, y_train, y_test = splits

model = RandomForestClassifier()
model.fit(X_train, y_train)

expected = y_test
predicted = model.predict(X_test)

print classification_report(expected, predicted)
```

DistrictDataLabs

Naive Bayes

DistrictDataLabs

Detecting Fraud

- Online store with at least 1000 orders per month and where 10% of orders are fraudulent
- 60 seconds to check an order, $15 per hour to a customer service representative = $3k per year!
- Goal: construct a model of probability that detects if an order is 50% likely to be fraudulent (reduce checking)

How likely is an order to be fraudulent?

What if we have more information about frauds?

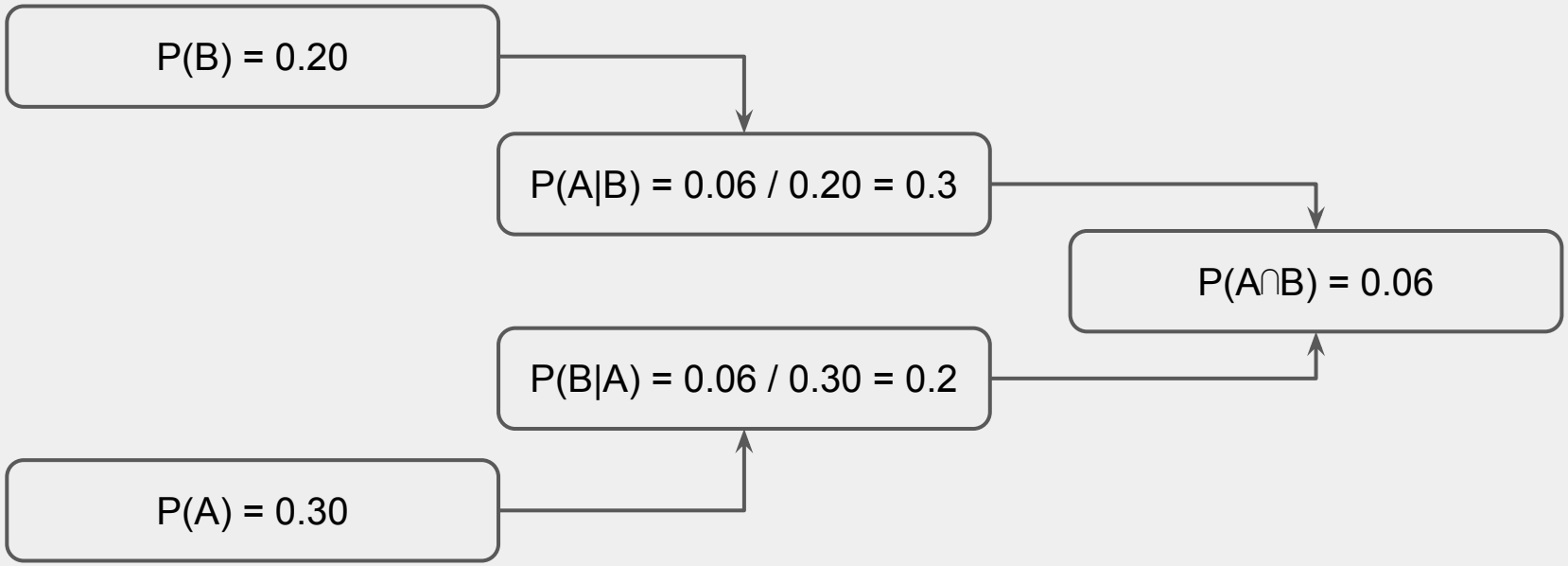
Frauds are likely to use gift cards and promo codes.

DistrictDataLabs

Conditional Probability

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

The probability of A *given* B is equal to the probability of A *and B* divided by the probability of B.



P(A|B) sits between P(A), P(B) and P(A∩B)

DistrictDataLabs

Detecting Fraud

We can now frame our problem in terms of the following probabilities:

$$P(Fraud|Giftcard) = \frac{P(Fraud \cap Giftcard)}{P(Giftcard)}$$

$$P(Fraud|Promo) = \frac{P(Fraud \cap Promo)}{P(Promo)}$$

But how do we compute P(Fraud∩Giftcard)?

DistrictDataLabs

Bayes Theorem

In the 18th century, Reverend Thomas Bayes came up with the research that Pierre-Simon Laplace would beautify:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Because:

$$P(B|A) = \frac{\frac{P(A \bigcap B)P(B)}{P(B)}}{P(A)} = \frac{P(A \bigcap B)}{P(A)}$$

DistrictDataLabs

Bayes and Fraud Detection

Now we can compute the following probabilities:

$$P(Fraud|Giftcard) = \frac{P(Giftcard|Fraud)P(Fraud)}{P(Giftcard)}$$

$$P(Fraud|Promo) = \frac{P(Promo|Fraud)P(Fraud)}{P(Promo)}$$

And these are easily computed by frequency. Let's say that gift cards are used 10% of the time, and 60% of gift card use is fraudulent: What is the probability of P(Fraud|Giftcard)?

DistrictDataLabs

Bayesian Classification

You might have guessed it: use of a promo or gift card during fraud are *features* of our Bayesian model.

What we want to solve now is: P(Fraud|Giftcard,Promo)

Using *joint probability* we can derive the *chain rule*:

$$P(A_1,A_2,\ldots,A_n)=P(A_1)P(A_2|A_1)P(A_3|A_1,A_2)\cdots P(A_n|A_1,A_2,\ldots,A_{n-1})$$

With Bayes, we can feed lots of information into our model. Which is now as follows:

$$P(Fraud|Giftcard,Promo)=\frac{P(Giftcard,Promo|Fraud)P(Fraud)}{P(Giftcard,Promo)}$$

DistrictDataLabs

Let's Be Naive!

Our model is still tough to solve, but let's look at the denominator - P(Giftcard, Promos) - this doesn't seem to be related to fraud.

We will make an ***assumption of independence*** and drop the denominator from our model. Using the chain rule:

$$P(Fraud,Giftcard,Promo)=P(Fraud)P(Gift|Fraud)P(Promo|Fraud,Gift)$$

Now we have to deal with P(Promo|Fraud,Gift), using the naive assumption again, along with a parameter, Z (sum of probabilities of all classes) we get the following:

$$P(Fraud|Giftcard,Promo)=\frac{1}{Z}P(Fraud)P(Gift|Fraud)P(Promo|Fraud)$$

DistrictDataLabs

Classification

- Building our model:
  - compute the probabilities of all features from a training set
  - compute the probabilities of all classes from training set
  - compute the parameter Z
  - Build a truth table
- Using our model:
  - compute “probabilities” of input vector
  - use truth table, trained probabilities and parameters to compute likelihood of class

	Fraud	Not Fraud
Gift Card	0.6	0.1
Promos	0.5	0.3
Class	0.1	0.9

DistrictDataLabs

Smoothing

- What happens if we receive new information?
- If the information has a probability = 0, then the independence assumption will cause skew.

Smoothing: Donating some of the probability density to unknown information.

Easiest smoothing technique: *laplacian (additive)*

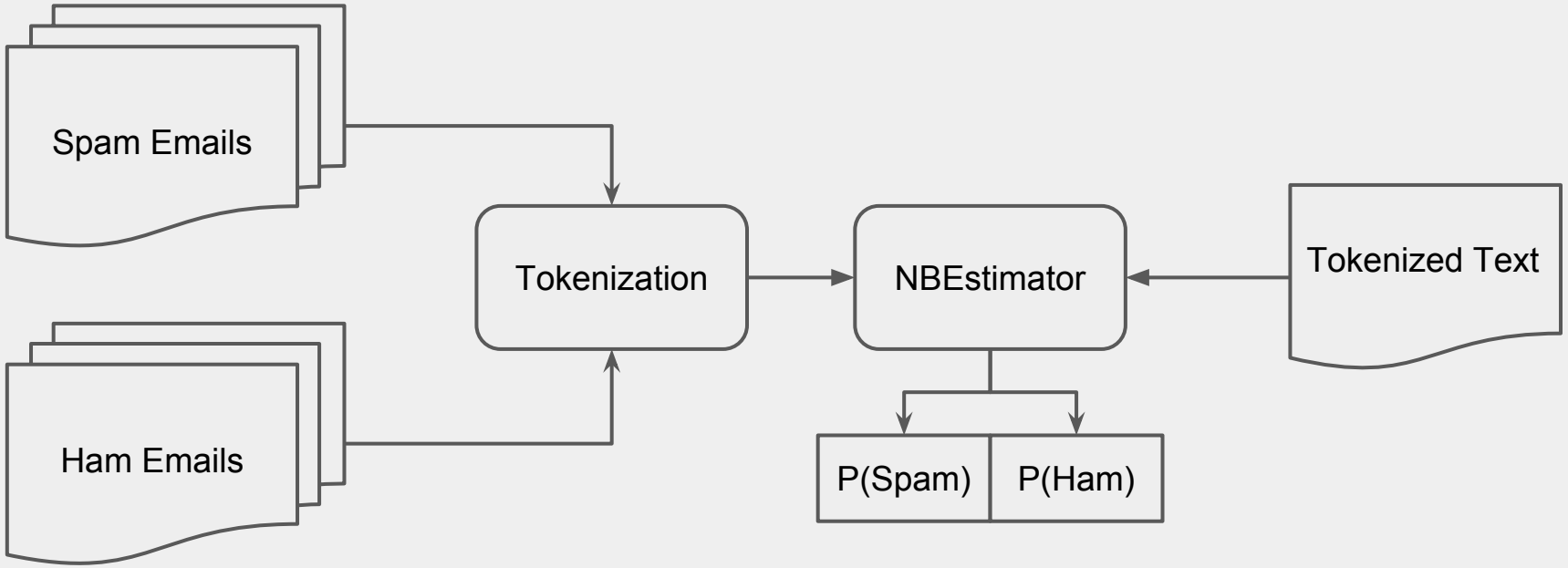
Simply give all unknown items a frequency of 1 - then recompute probabilities accordingly.

DistrictDataLabs

Text Classification with NB

Spam/Ham is the canonical Naive Bayesian classifier for text and is popular because if it's wide use.

Compute using "bag of words" - each word is a feature.



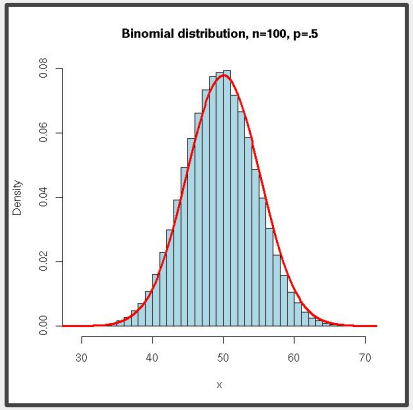
DistrictDataLabs

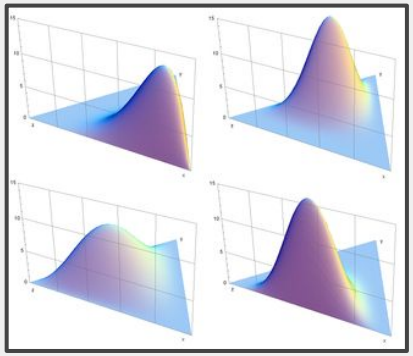
Naive Bayes in Scikit-Learn

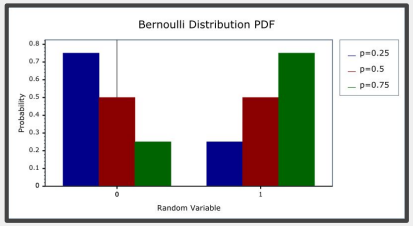
GaussianNB: the likelihood of the features is assumed to be Gaussian (standardized).

MultinomialNB: for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice).

BernoulliNB: for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable







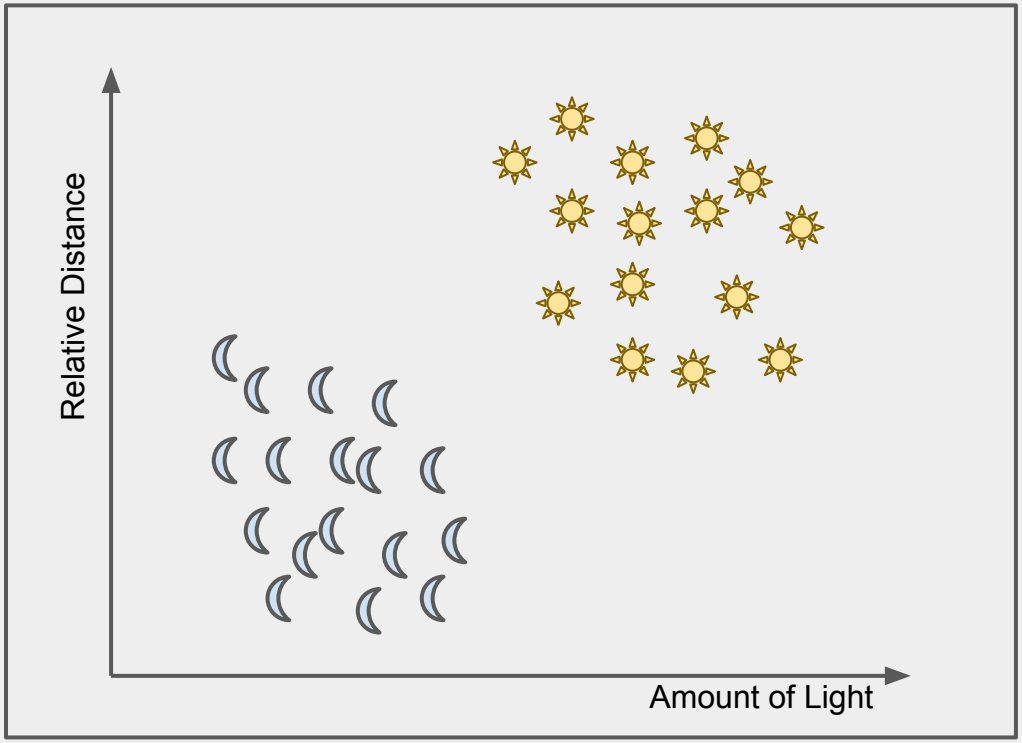
DistrictDataLabs

Support Vector Machines

DistrictDataLabs

Stars or Moons?

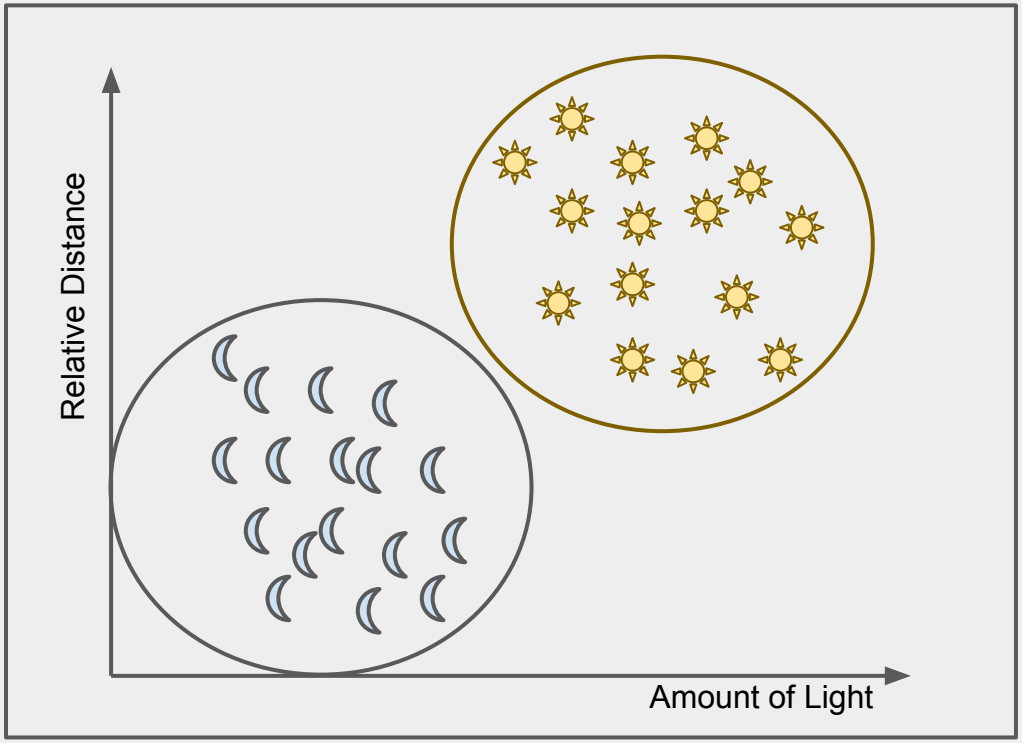
Let's say we know we have two discernible groups (a strong hypothesis):



DistrictDataLabs

Stars or Moons?

Using kNN we're highly dependent on the instances, and we get centers:



DistrictDataLabs

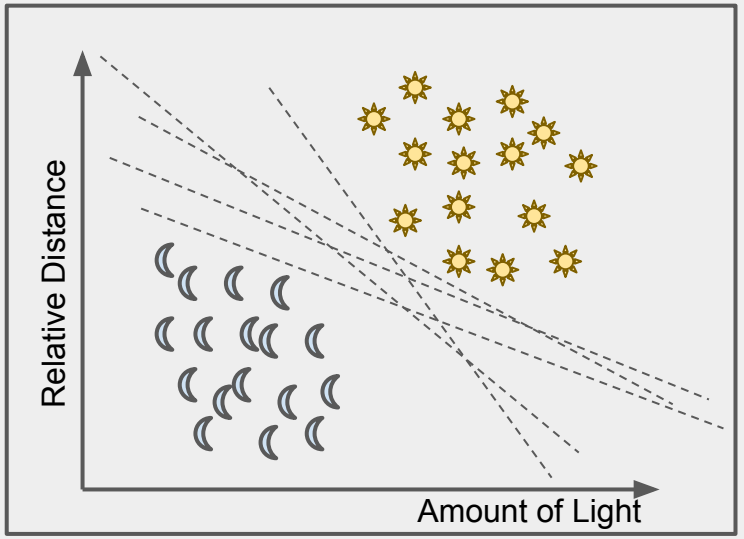
Decision Boundary

In order to *generalize* this model, we really want to find a *decision boundary* - a line between two classes of data.

How many lines fit between moons and stars?

Instead of picking an arbitrary line, we attempt to maximize the distance between classes.

Definition
In geometry a **hyperplane** is a *subspace* of one dimension less than its ambient space. If a space is 3-dimensional then its **hyperplanes** are the 2-dimensional planes, while if the space is 2-dimensional, its **hyperplanes** are the 1-dimensional lines.



DistrictDataLabs

Support Vector Machines

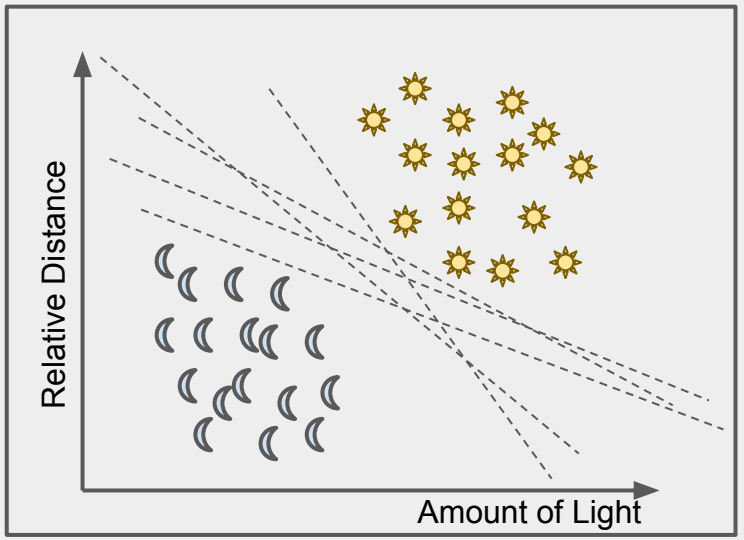
- Introduced by Vladimir Vapnik in 1980s
- Modern interpretation is less stringent (added slack)

Designed to solve a two-group classification:

- boolean (true, false)
- ids (3,4)
- sign (1, -1)

Desirable:

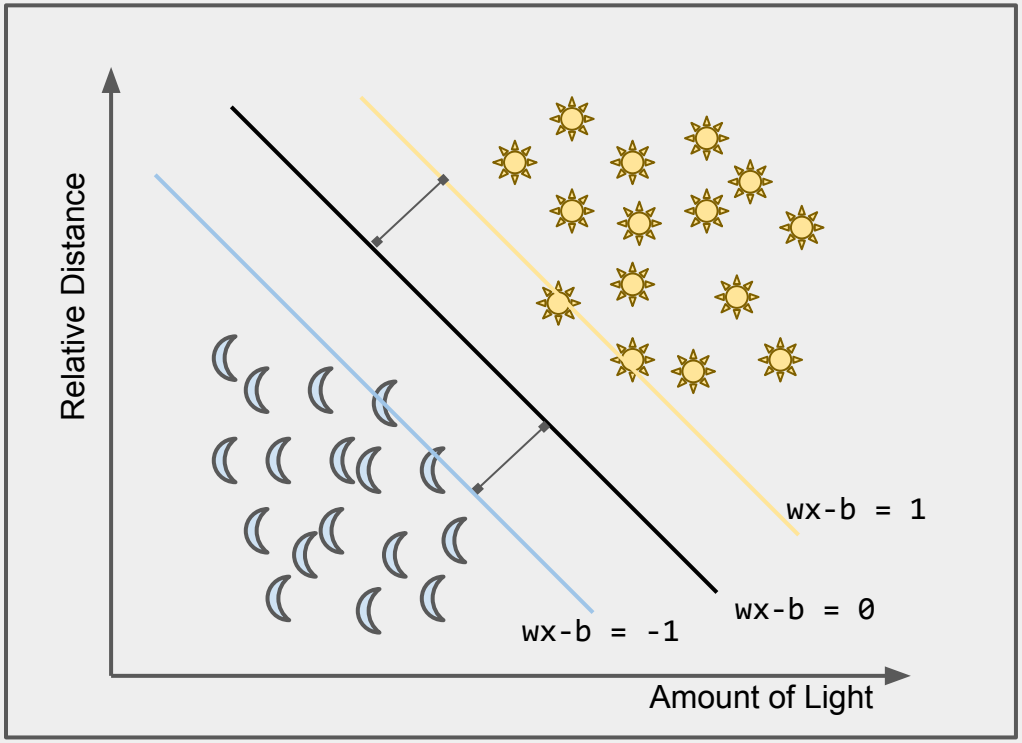
- Avoids curse of dimensionality
- Works in high dimensional space
- FAST to compute



DistrictDataLabs

Maximizing Distance

Solve for w for the function wx-b=0 to determine the hyperplane maximizing the distance between two groups.



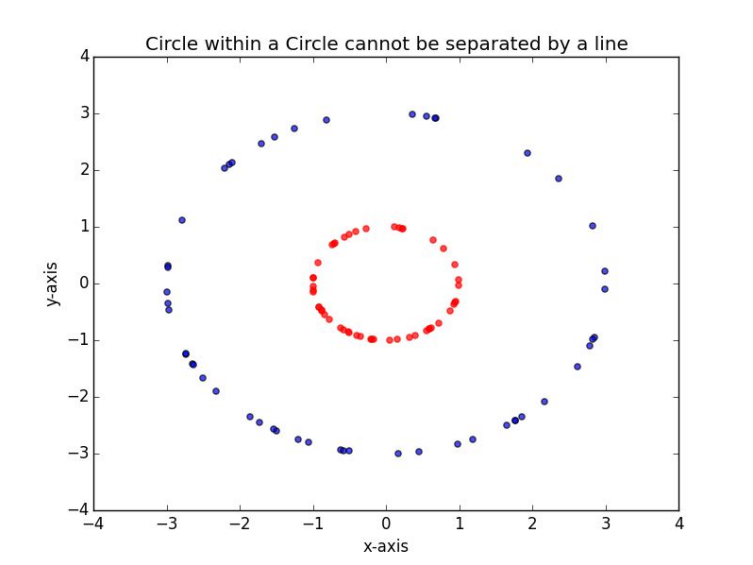
Solvable by finding the perpendicular hyperplane to all three parallel planes and dividing by 2, which happens to be:

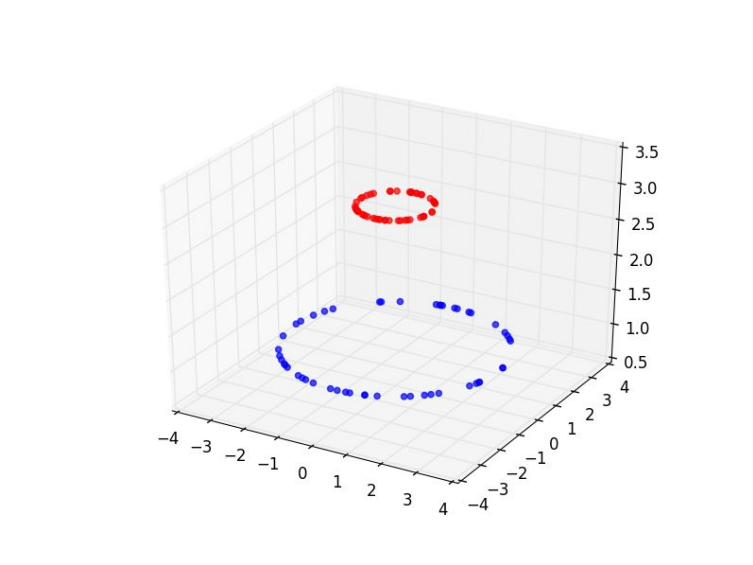
```
2 / ||w||
```

DistrictDataLabs

The Kernel Trick

- Examples are linear, real data is not (and also usually in a sparse, high-dimensional space).
- Kernel trick: transpose data into a higher dimensional space. E.g. for circles (polynomial kernel):





$\langle x, y \rangle$

$\langle x^2, \sqrt{2}xy, y^2 \rangle$

DistrictDataLabs

Trickier Kernel

Take a linear x and transform it into $\phi(x)$ or some function that better suits the data. Points to consider:

- $\phi(x_i)\phi(x_j)$ might take a long time to compute
- ϕ might be a complicated function
- Extensive training data will be slow

So, we try another trick: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$

Mapping the dot product + cheap function (e.g. inner dot product is already calculated) = *kernel function*

DistrictDataLabs

Kernel Functions

Homogeneous Polynomial
d is the degree, any number > 0
works best in most cases.

$$K(x_i, x_j) = (x_i^T x_j)^d$$

Heterogeneous Polynomial
Add a non-negative, non-zero constant, c
Increases relevance of higher order features

$$K(x_i, x_j) = (x_i^T x_j + c)^d$$

Radial Basis Function (RBF)
Often used more: high-D performance
Can create infinite new dimensions
Numerator is euclidean distance
σ is a free parameter

$$K(x_i, x_j) = exp\left(\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$$

DistrictDataLabs

Kernel Functions, Scikit-Learn

Linear Kernel:
C is slack variable

$$\gamma \langle x_i, x_j \rangle$$

Polynomial Kernel:
d is degree, r is coef0

$$(\gamma \langle x_i, x_j \rangle + r)^d$$

RBF Kernel:
gamma keyword > 0

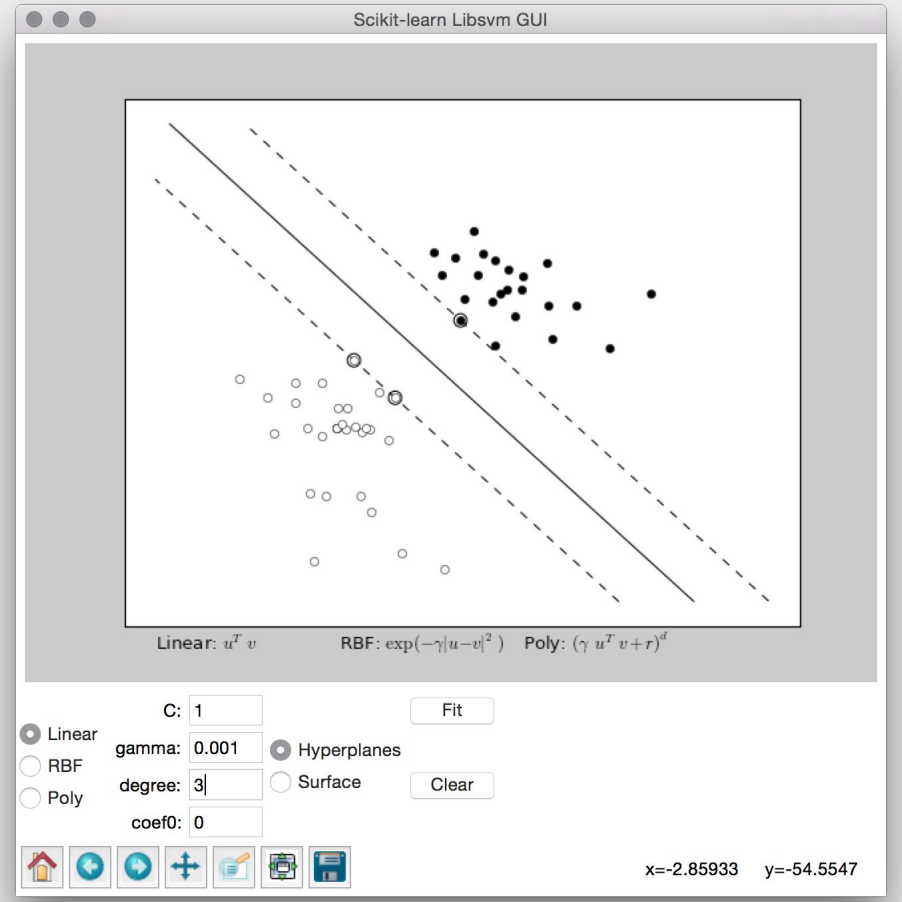
$$exp(-\gamma |x_i - x_j|^2)$$

Sigmoid Kernel:
r is specified by coef0

$$\tanh(\gamma \langle x_i, x_j \rangle + r)$$

Define your own! Specify a Python function or precomputed Gram matrix.

DistrictDataLabs



SVM Demo

DistrictDataLabs