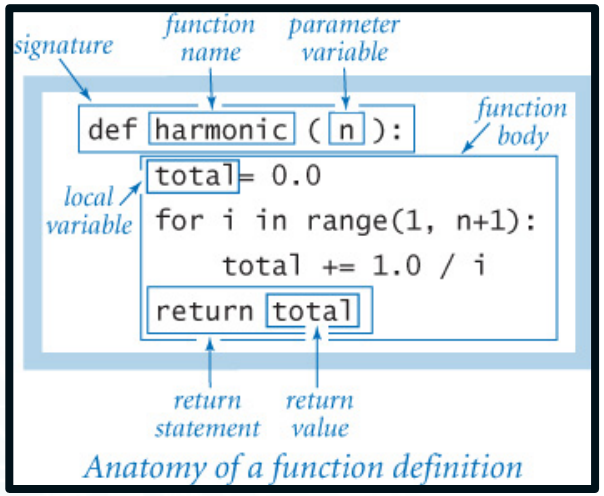
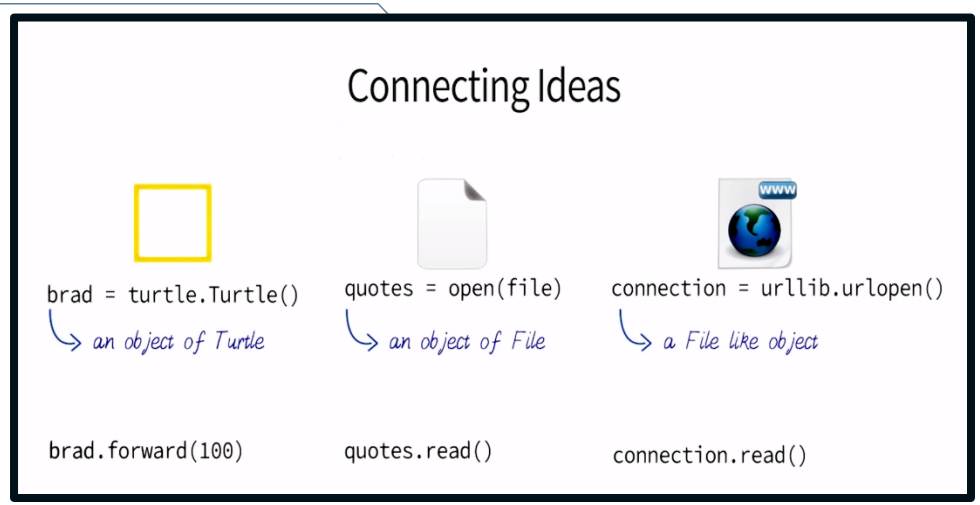
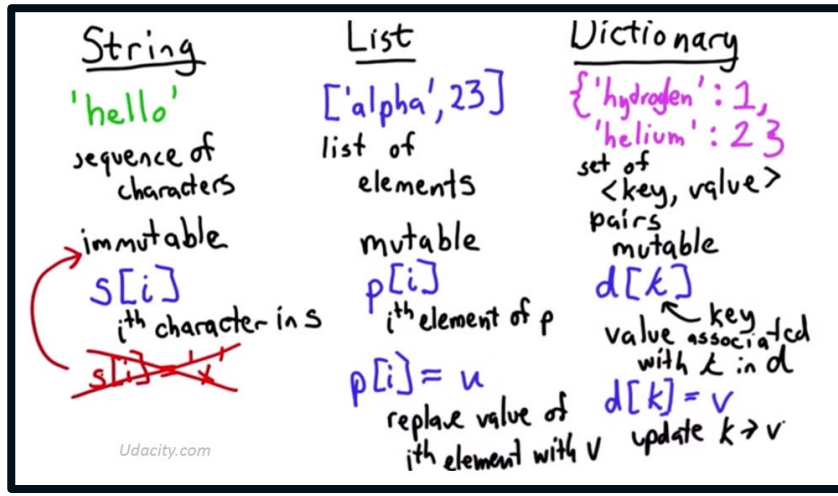
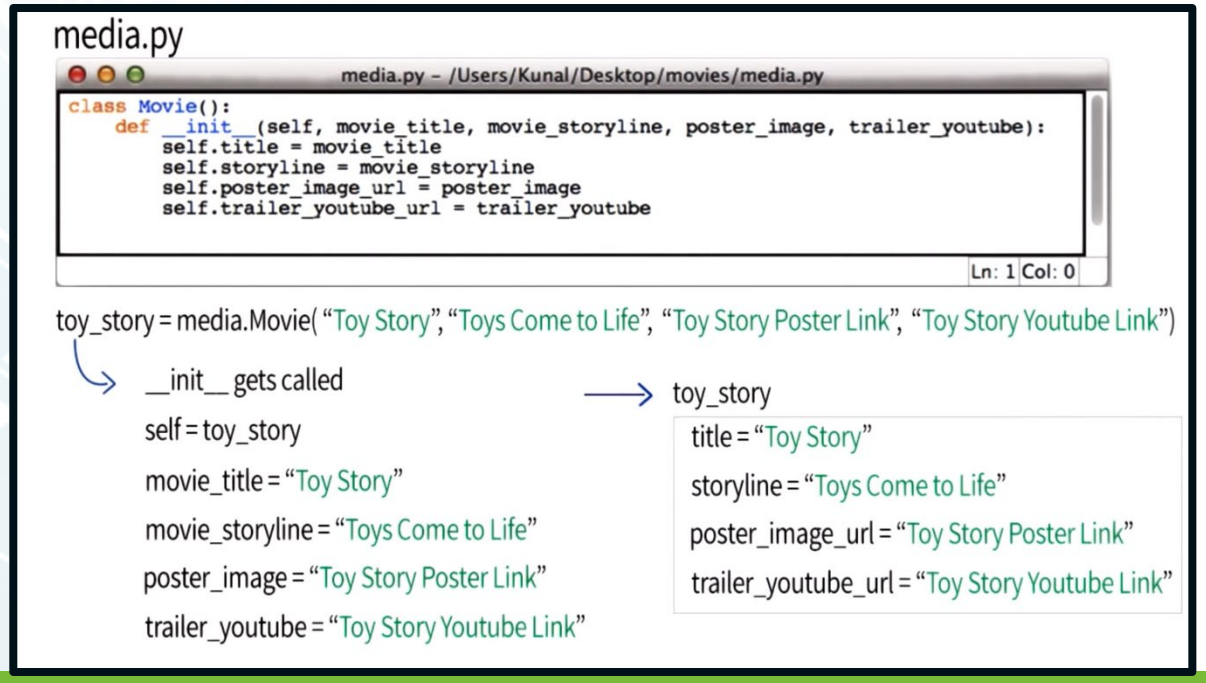
Confidential

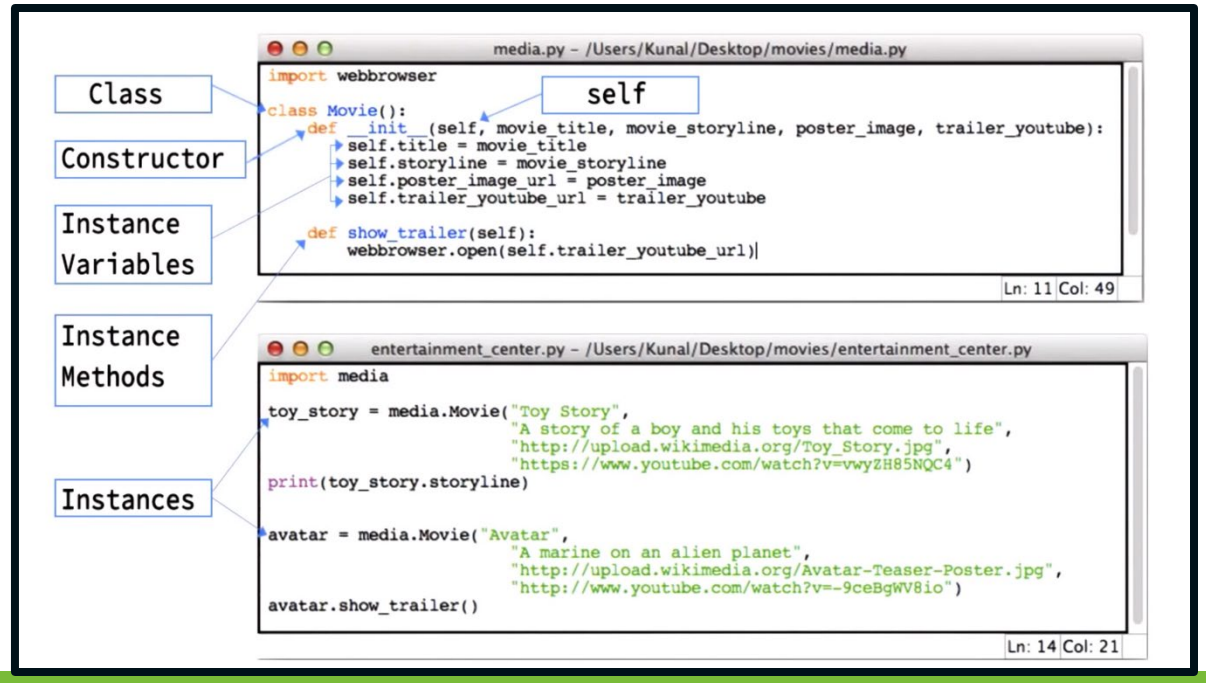
Python Foundation











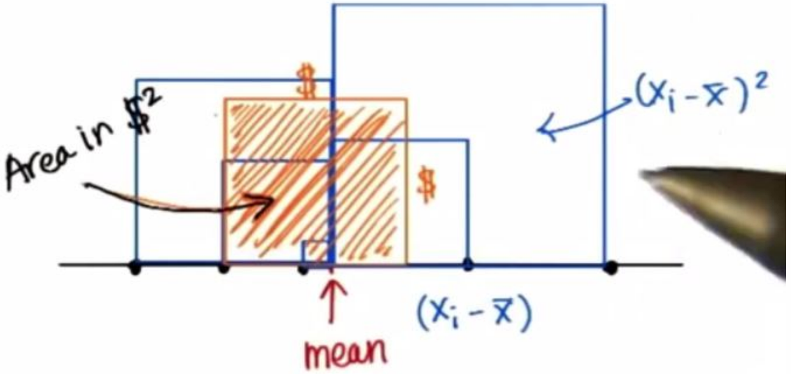
Measure of Variability

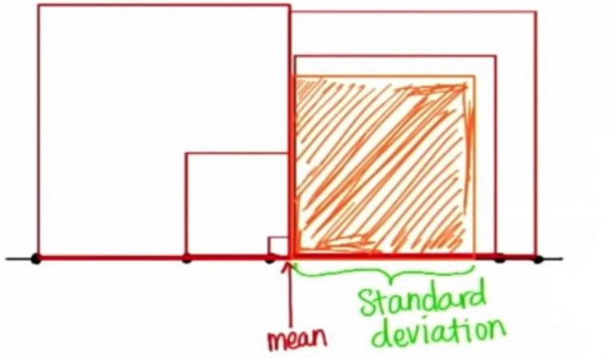
Sample	Deviation from Mean $(x_i - \bar{x})$	Squared Deviations
$33,219	-19,574.80	383,172,795
$36,254	-16,539.80	273,564,984
$38,801	-13,992.80	195,798,452
$46,335	-6,458.80	41,716,097
$46,840	-5953.80	35,447,734
$47,596	-5197.80	27,017,124
$55,130	2336.20	5,457,830
$56,863	4069.20	16,558,389
$78,070	25,276.20	638,886,286
$88,830	36,036.20	1,298,607,710

$\bar{x} = \frac{\Sigma x}{10} =$ $52,793.80

Avg. dev. $= \frac{\Sigma(x_i - \bar{x})}{10} = 0$

Avg. squared dev $= \frac{\Sigma(x_i - \bar{x})^2}{10} =$ 291,622,740 ← VARIANCE





$SS = \Sigma(\text{area of each square})$

$= \Sigma(x_i - \bar{x})^2$

Avg. squared deviation $= \frac{\Sigma(x_i - \bar{x})^2}{n}$

Transforming Skewed Continuous Features

A dataset may sometimes contain at least one feature whose values tend to lie near a single number, but will also have a non-trivial number of vastly larger or smaller values than that single number. Algorithms can be sensitive to such distributions of values and can underperform if the range is not properly normalized. For highly-skewed feature distributions such as 'capital-gain' and 'capital-loss', it is common practice to apply a logarithmic transformation on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers. Care must be taken when applying this transformation however: The logarithm of 0 is undefined, so we must translate the values by a small amount above 0 to apply the the logarithm successfully.

In addition to performing transformations on features that are highly skewed, it is often good practice to perform some type of scaling on numerical features. Applying a scaling to the data does not change the shape of each feature's distribution (such as 'capital-gain' or 'capital-loss' above); however, normalization ensures that each feature is treated equally when applying supervised learners. Note that once scaling is applied, observing the data in its raw form will no longer have the same original meaning

If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test, which calculates the best power transformation of the data that reduces skewness. A simpler approach which can work in most cases would be applying the natural logarithm.

After applying a natural logarithm scaling to the data, the distribution of each feature should appear much more normal.

Measure of Accuracy

Classification vs Regression Metrics

In classification we want to see how often a model correctly or incorrectly identifies a new example, whereas in regression we might be more interested to see how far off the model's prediction is from the real true value.

For the rest of this lesson we will discuss various performance metrics. For classification we will cover accuracy, precision, recall, and F-score. For regression we will go over mean absolute error and mean squared error.

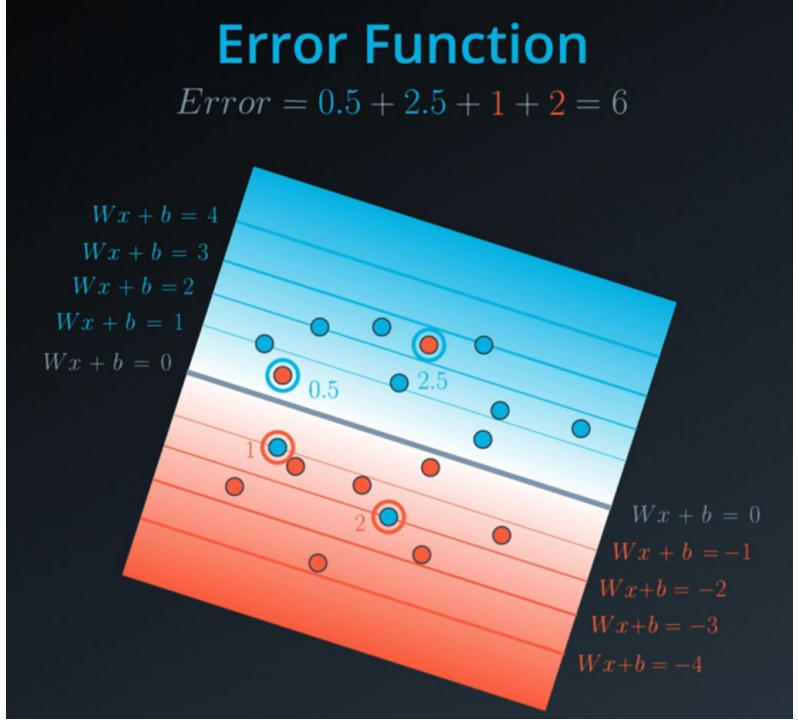
F1 Score

Now that you've seen precision and recall, another metric you might consider using is the F1 score. F1 score combines precision and recall relative to a specific positive class.

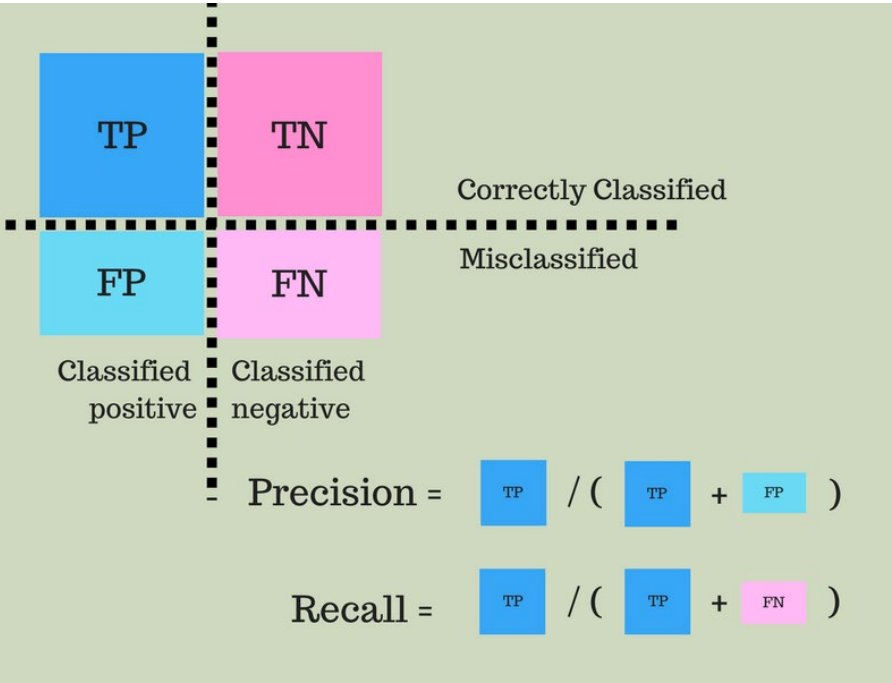
The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

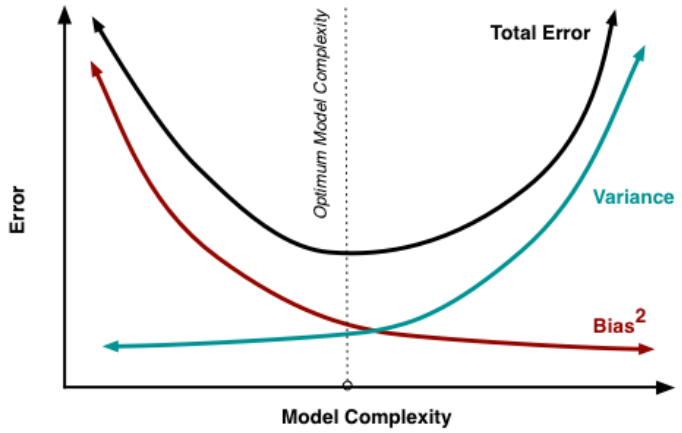
Precision tells us what proportion of messages we classified as spam, actually were spam. Recall(sensitivity) tells us what proportion of messages that actually were spam were classified by us as spam.

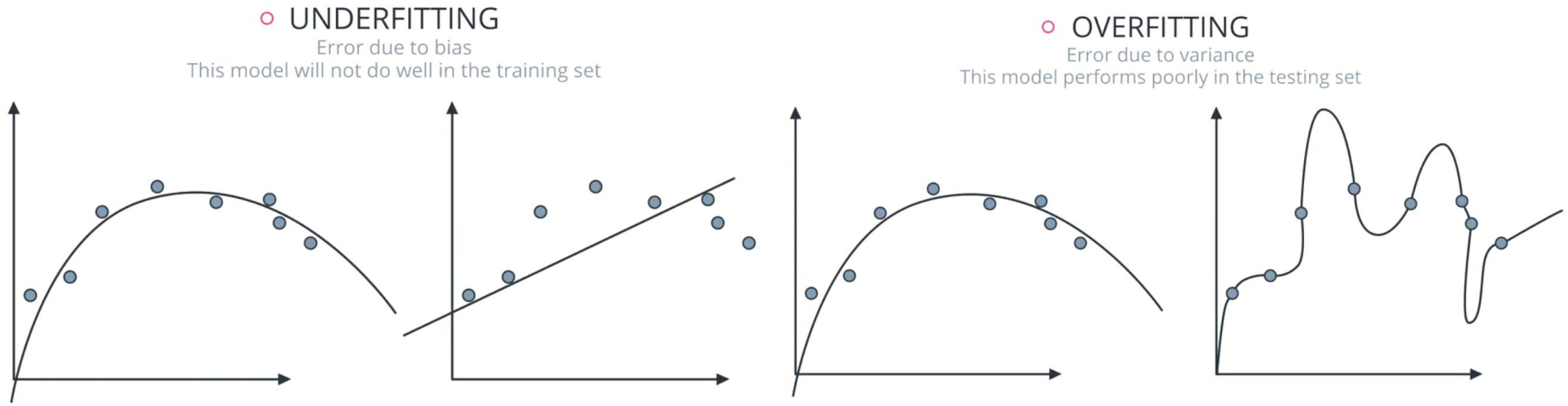


		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)



Tradeoff between bias and variance





Bias is due to a model being unable to represent the complexity of the underlying data. High bias pays litte attention to data.

Variance is due to a model being overly sensitive to the limited data. High variance pays too much attention to data

Parameter Optimization/Tuning

GridSearchCV is a way of systematically working through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance. The beauty is that it can work through many combinations in only a couple extra lines of code.

Here's an example from the sklearn documentation:

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svr = svm.SVC()
clf = grid_search.GridSearchCV(svr, parameters)
clf.fit(iris.data, iris.target)
```

Let's break this down line by line.

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
```

A dictionary of the parameters, and the possible values they may take. In this case, they're playing around with the kernel (possible choices are 'linear' and 'rbf'), and C (possible choices are 1 and 10).

Then all the following combinations of values for (kernel, C) are automatically generated: [('rbf', 1), ('rbf', 10), ('linear', 1), ('linear', 10)]. Each is used to train an SVM, and the performance is then assessed using cross-validation.

```
svr = svm.SVC()
```

This looks kind of like creating a classifier, just like we've been doing since the first lesson. But note that the "clf" isn't made until the next line--this is just saying what kind of algorithm to use. Another way to think about this is that the "classifier" isn't just the algorithm in this case, it's algorithm plus parameter values. Note that there's no monkeying around with the kernel or C; all that is handled in the next line.

```
clf = grid_search.GridSearchCV(svr, parameters)
```

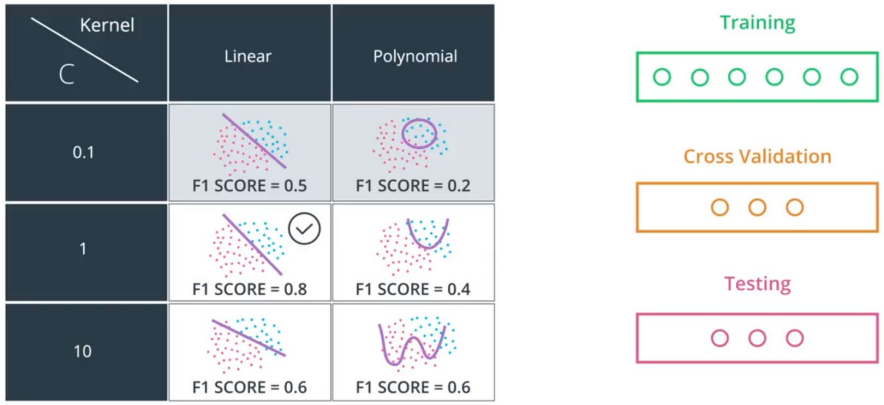
This is where the first bit of magic happens; the classifier is being created. We pass the algorithm (*svr*) and the dictionary of parameters to try (*parameters*) and it generates a grid of parameter combinations to try.

```
clf.fit(iris.data, iris.target)
```

And the second bit of magic. The fit function now tries all the parameter combinations, and returns a fitted classifier that's automatically tuned to the optimal parameter combination. You can now access the parameter values via *clf.bestparams*.

Choose a scikit-learn classifier (e.g., adaboost, random forests) that has a `feature_importance_` attribute, which is a function that ranks the importance of features according to the chosen classifier.

◦ GRID SEARCH CROSS VALIDATION



Learning Curves

Learning Curves

Let us explore the relationship between bias and variance on a model's ability to learn from data through visual graphs.

A learning curve in machine learning is a visual graph that compares the metric performance of a model on training and testing data over a number of training instances.

When we look a the relationship between data and error we should generally see a downward trend of error as the number of training points increases. This should make sense since we are trying to build models that learn from experience.

We separate training and testing sets so we can get a better idea whether the model can generalize to unseen data rather than fit to the data just seen.

You can verify in a learning curve when a model has learned as much as it can about the data when both the training and testing curves plateau and there is a consistent gap between the two error rates.

Learning Curves

Bias

When the training and testing errors converge and are quite high this usually means the model is biased. No matter how much data we feed it, the model cannot represent the underlying relationship and therefore has systematic high errors.

Variance

When we have a large gap between the training and testing error this generally means the model suffers from high variance. Unlike a bias model, models that suffer from variance can generally improve if we have more data to learn from or if we simplify the model to represent only the most important features of the data.

Ideal Learning Curve

The ultimate goal for a model is one that has low errors and generalizes pretty well for unseen data (testing data). We can see this when both the testing and training curves converge and where the error is extremely low. That is the model is very accurate on unseen data.

Model Complexity

Unlike a learning curve graph, a model complexity graph looks at how the complexity of a model changes the training and testing curves rather than the number of data points to train on. The general trend of is that as a model increases, the more variability exists in the model for a fixed set of data.

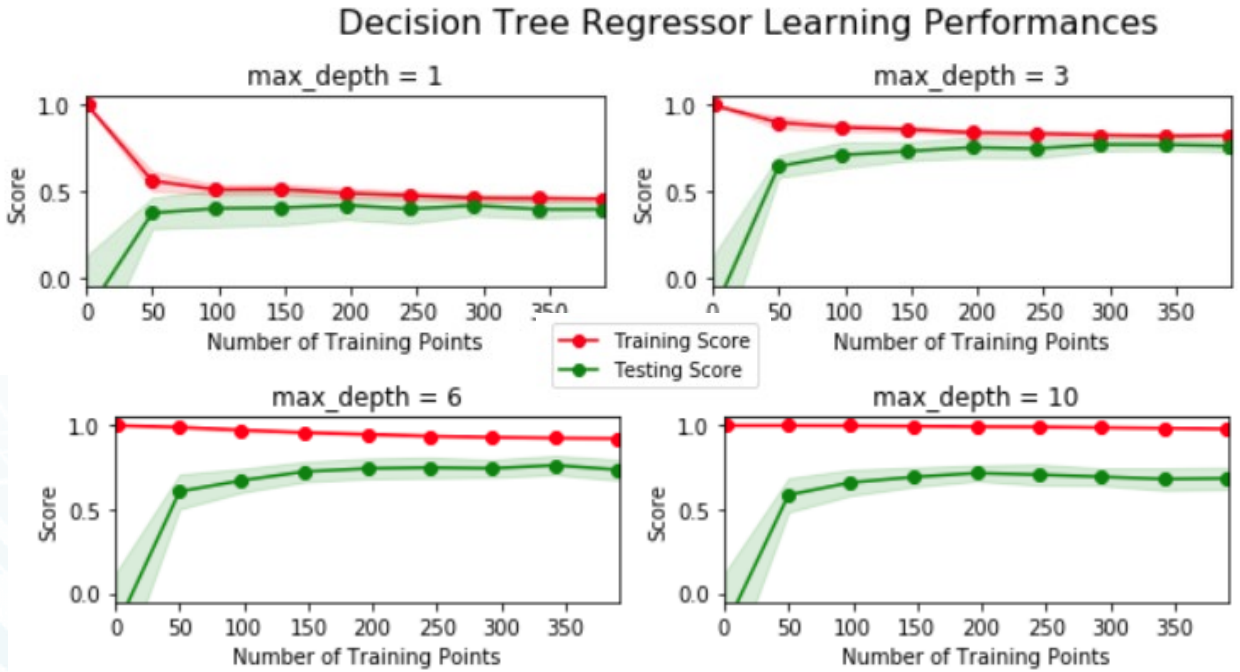
Learning Curves and Model Complexity

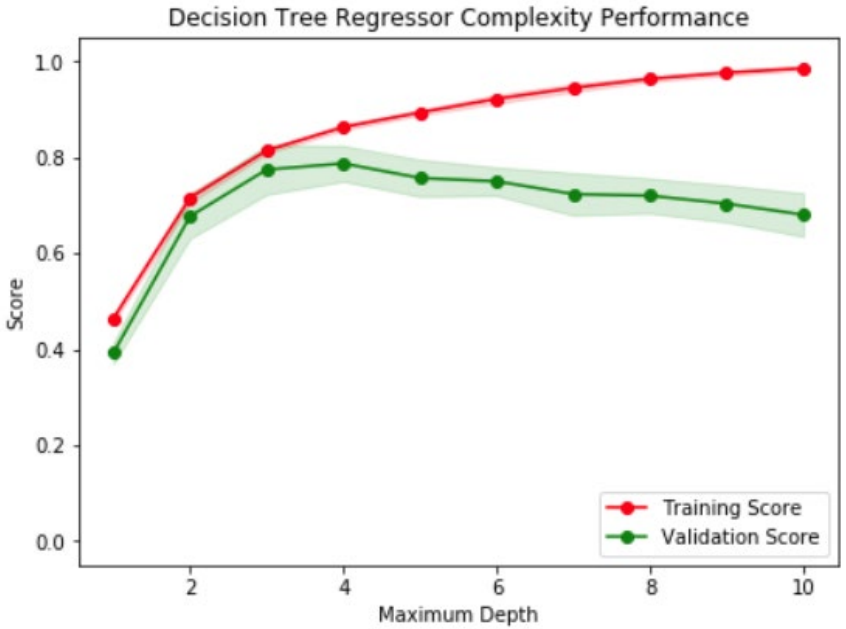
So what is the relationship between learning curves and model complexity?

If we were to take the learning curves of the same machine learning algorithm with the same fixed set of data, but create several graphs for the increase model complexity, all the learning curve graphs would represent the model complexity graph. That is, if we took the final testing and training errors for each model complexity and visualized them along the complexity of the model we would be able to see how well the model performs as the model complexity increases.

Practical use of Model Complexity

Knowing that we can identify issues with bias and variance by analyzing a model complexity graph, we now have a visual tool to help identify ways to optimize our models. This will supplement our use of GridSearchCV, allowing us to combine that practical tool with a more holistic, visual understanding of why and how specific models or sets of parameters perform well.





- LEARNING CURVES

Receiver Operating Characteristic: ROC curve

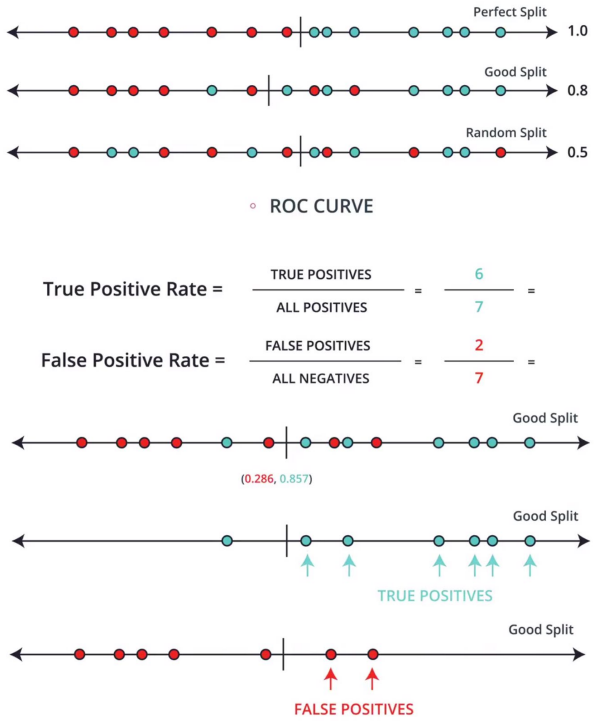
Come up with a metric or some number that is high for the perfect split, medium for the good split, and low for the random split. ROC curve will help us.

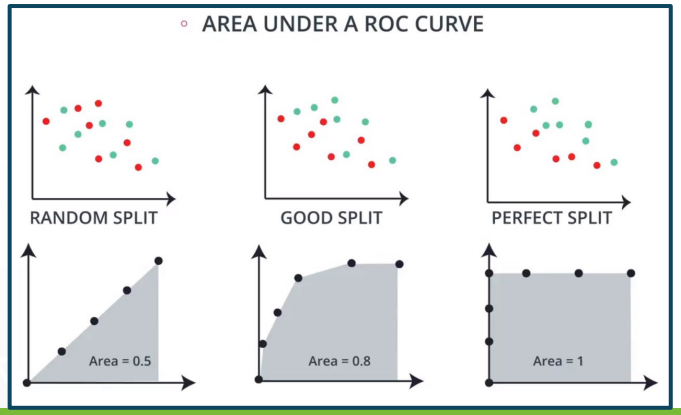
True positive rates: out of all the positively labeled points, how many did we classify correctly?

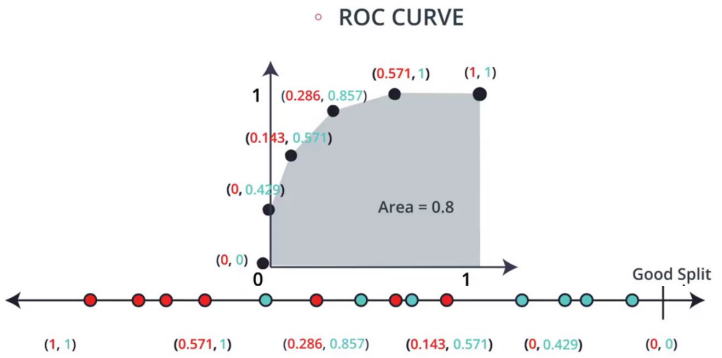
False positive rates: out of all the negative points, how many of them did the model incorrectly think they were positives?

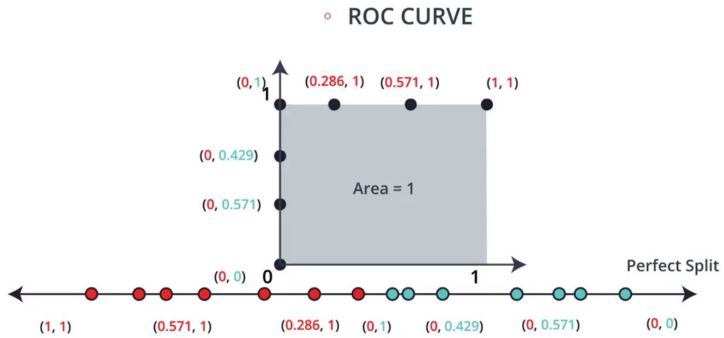
Remember these two numbers. Move this boundary around and calculate the same pair of numbers

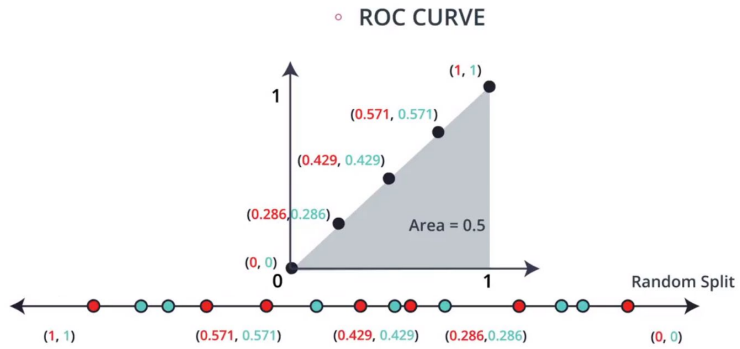
In summary, the closer your area under the ROC curve is to one, the better your model is



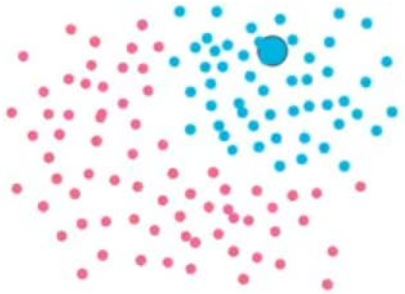


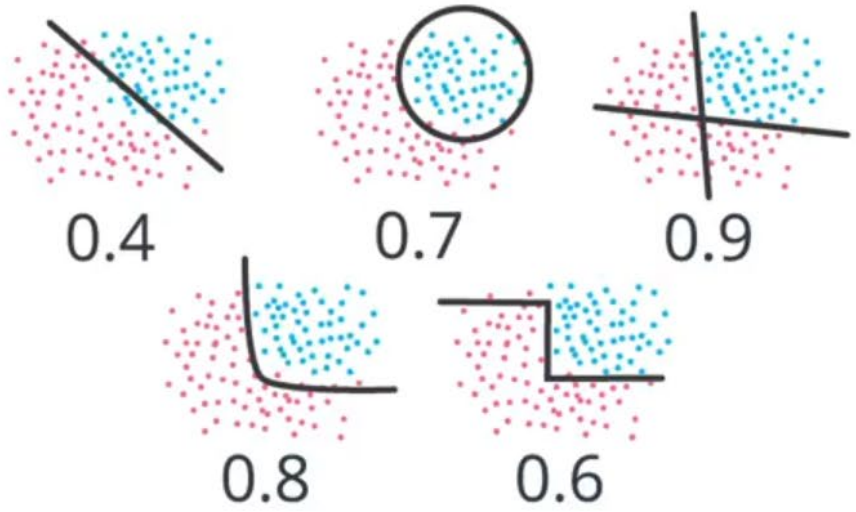


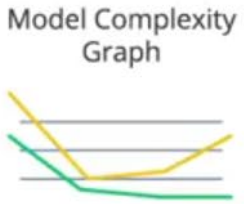


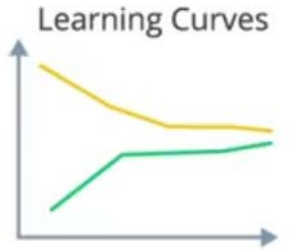


HOW TO USE MACHINE LEARNING



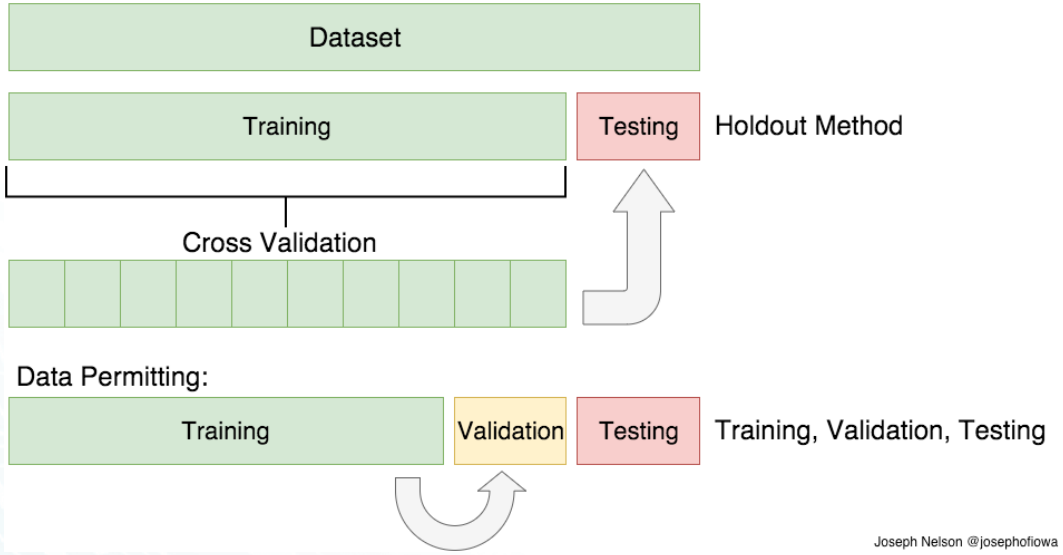


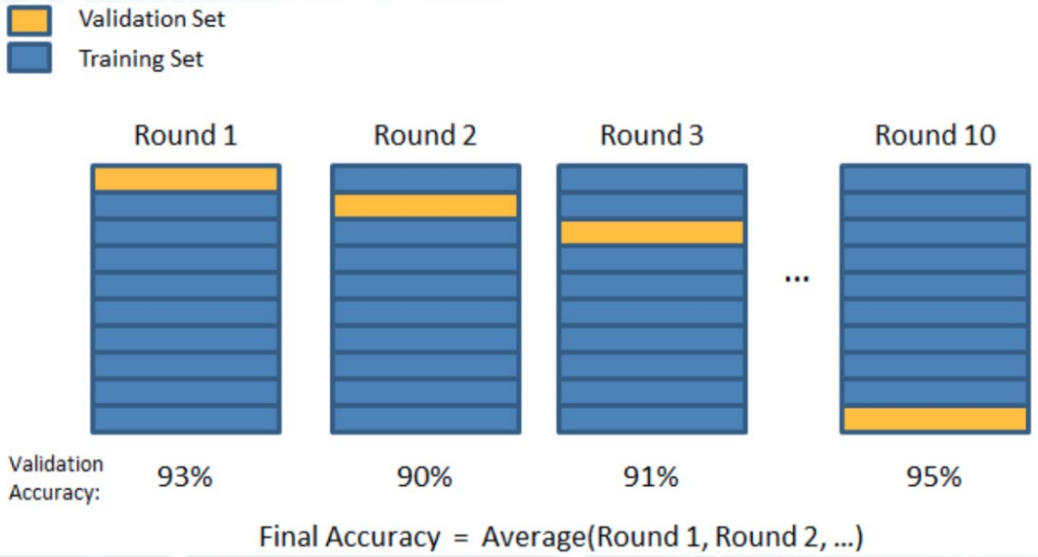




DATA **ALGORITHMS** **METRICS**

Cross Validation on Training Set





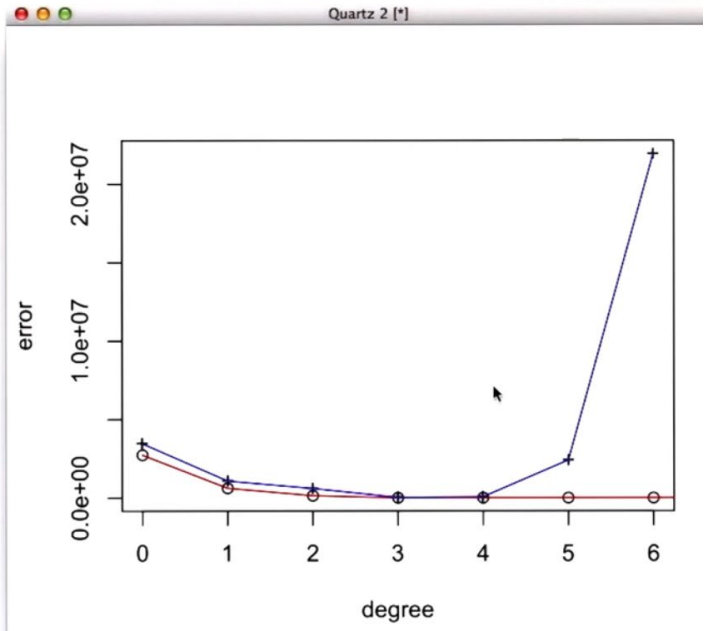
Housing Example Revisited

Degree of polynomial

Per-example error

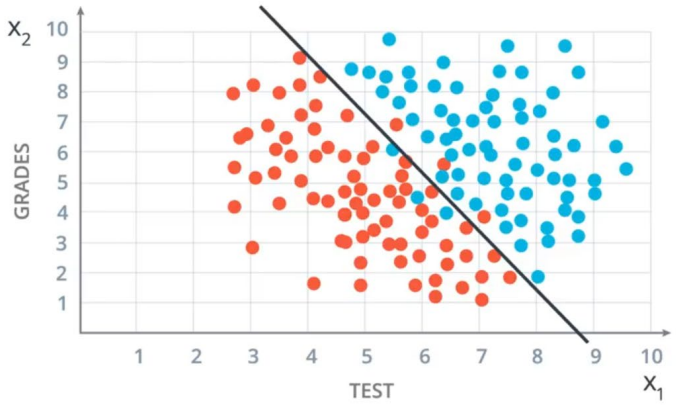
training error
→ improves

cross validation error
→ falls then rises!



Wx + b Explanation

Acceptance at a University



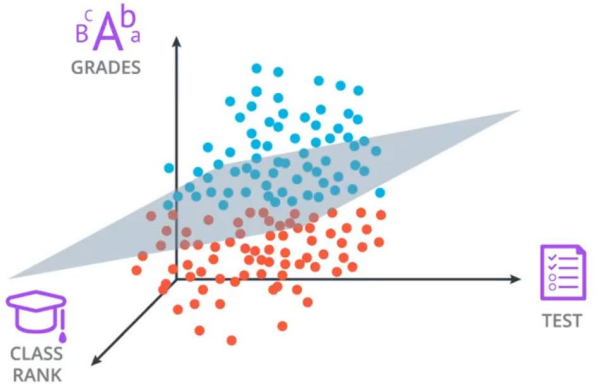
BOUNDARY:
A LINE

$w_1x_1 + w_2x_2 + b = 0$
$Wx + b = 0$
$W = (w_1, w_2)$
$x = (x_1, x_2)$
y = label: 0 or 1

PREDICTION:

$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$

Acceptance at a University



BOUNDARY:
A PLANE

$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$
$Wx + b = 0$

PREDICTION:

$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$

Acceptance at a University

	x_1 EXAM 1	x_2 EXAM 2	x_3 GRADES	...	x_n ESSAY	y PASS?
STUDENT 1	9	6	5	...	6	1(yes)
STUDENT 2	8	4	8	...	3	0(no)
...	...	...	...	...	...	
STUDENT n	6	7	2	...	8	1(yes)

← n columns →

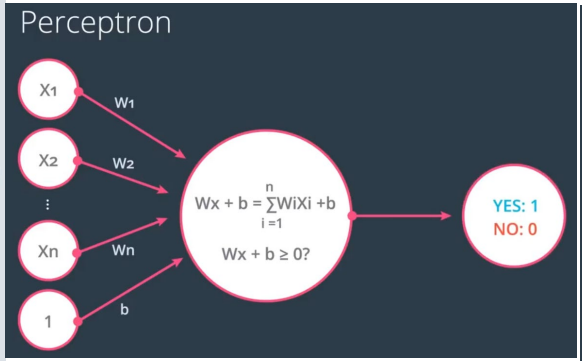
n-dimensional space
$x_1, x_2, \ldots, x_n$

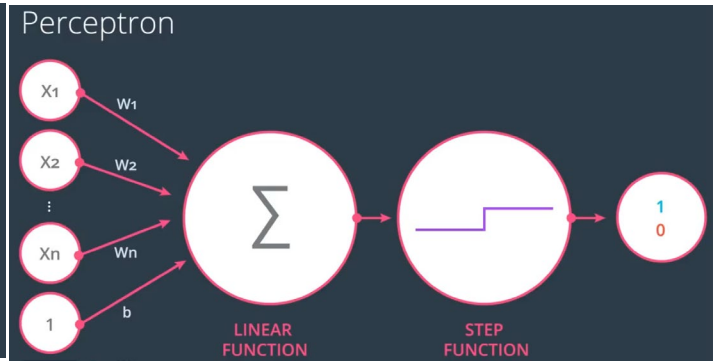
BOUNDARY:
n-1 dimensional hyperplane

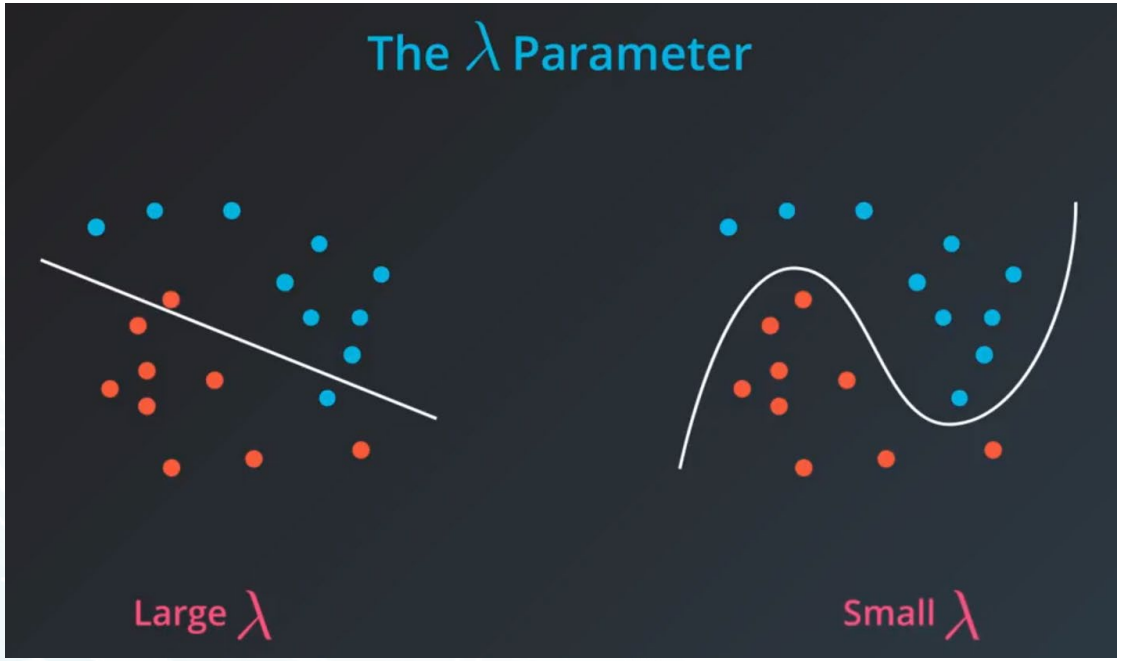
$w_1x_1 + w_2x_2 + w_nx_n + b = 0$
$Wx + b = 0$

PREDICTION:

$$\hat{y} = \begin{cases} 1 \text{ if } Wx + b \geq 0 \\ 0 \text{ if } Wx + b < 0 \end{cases}$$







Lambda is amount of punishing complexity in regression

L1 vs L2 Regularization

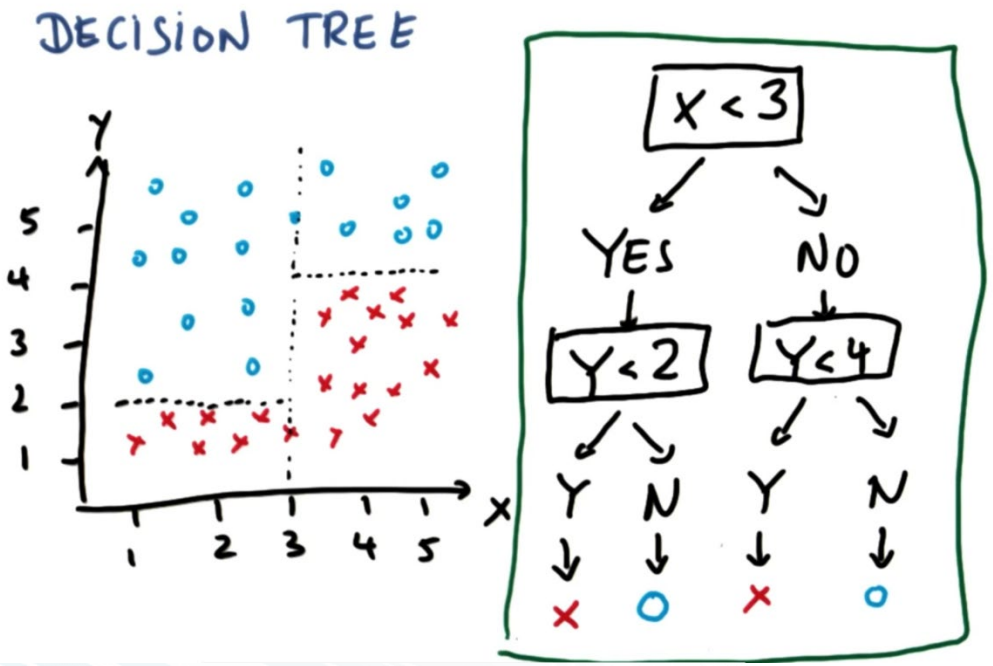
L1 Regularization	L2 Regularization
Computationally Inefficient (unless data is sparse)	Computationally Efficient
Sparse Outputs	Non-Sparse Outputs
Feature Selection	No Feature Selection

L1 is sum of absolute coeffs in regression

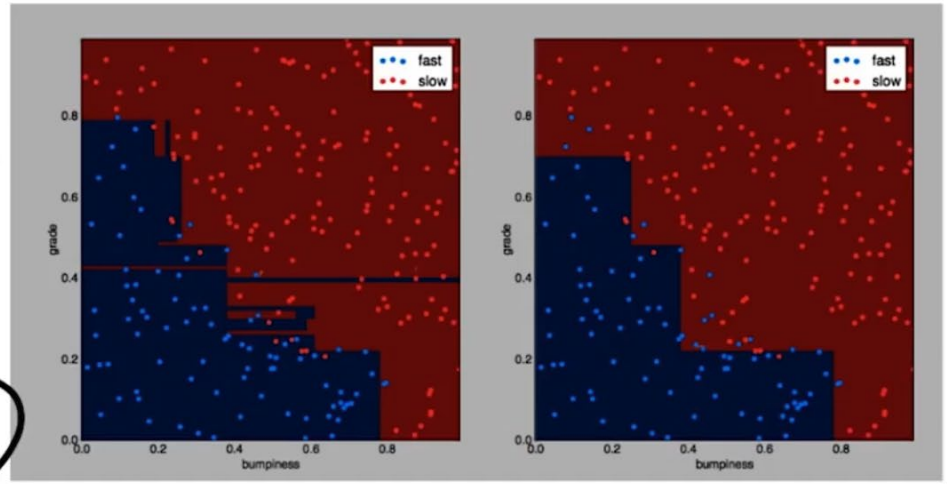
L2 is sum of squared coeffs in regression

In the end, L is added to error

Decision Tree Hyperparameters



min_samples_split and overfitting



2

90.8%

50

91.2%

What is the accuracy?

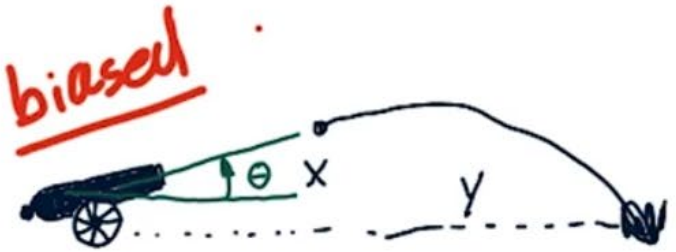
Depth = 1 Depth = 2 Depth = 3 Depth = 4

Maximum depth of a decision tree

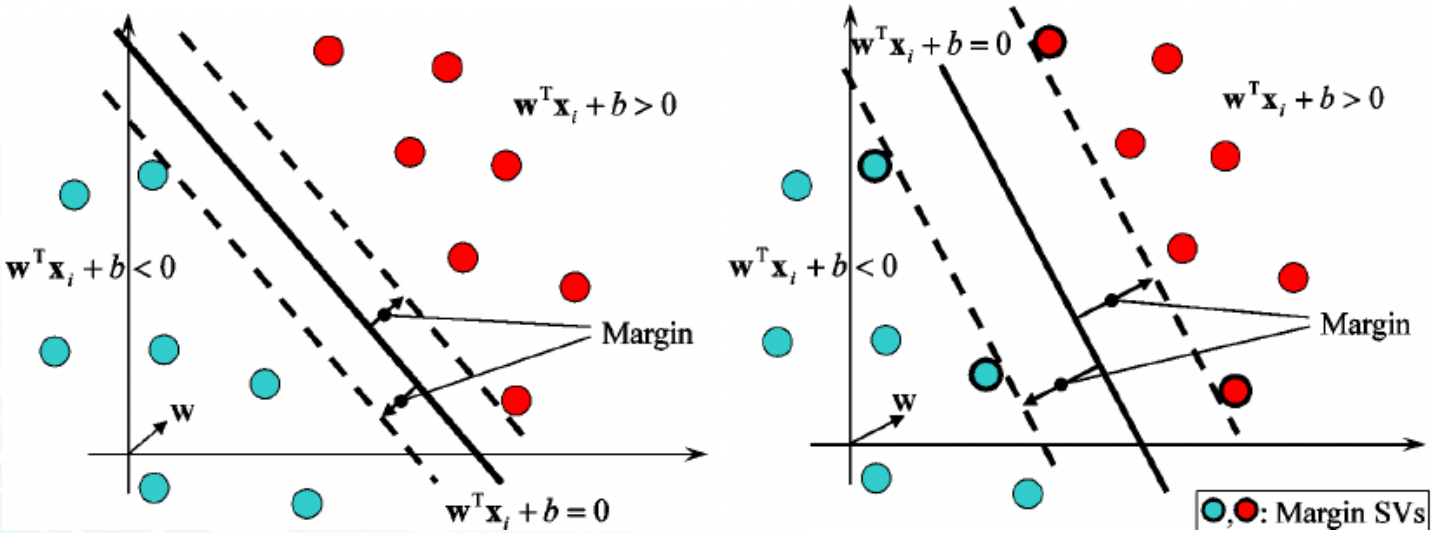
Parametric or Non Model

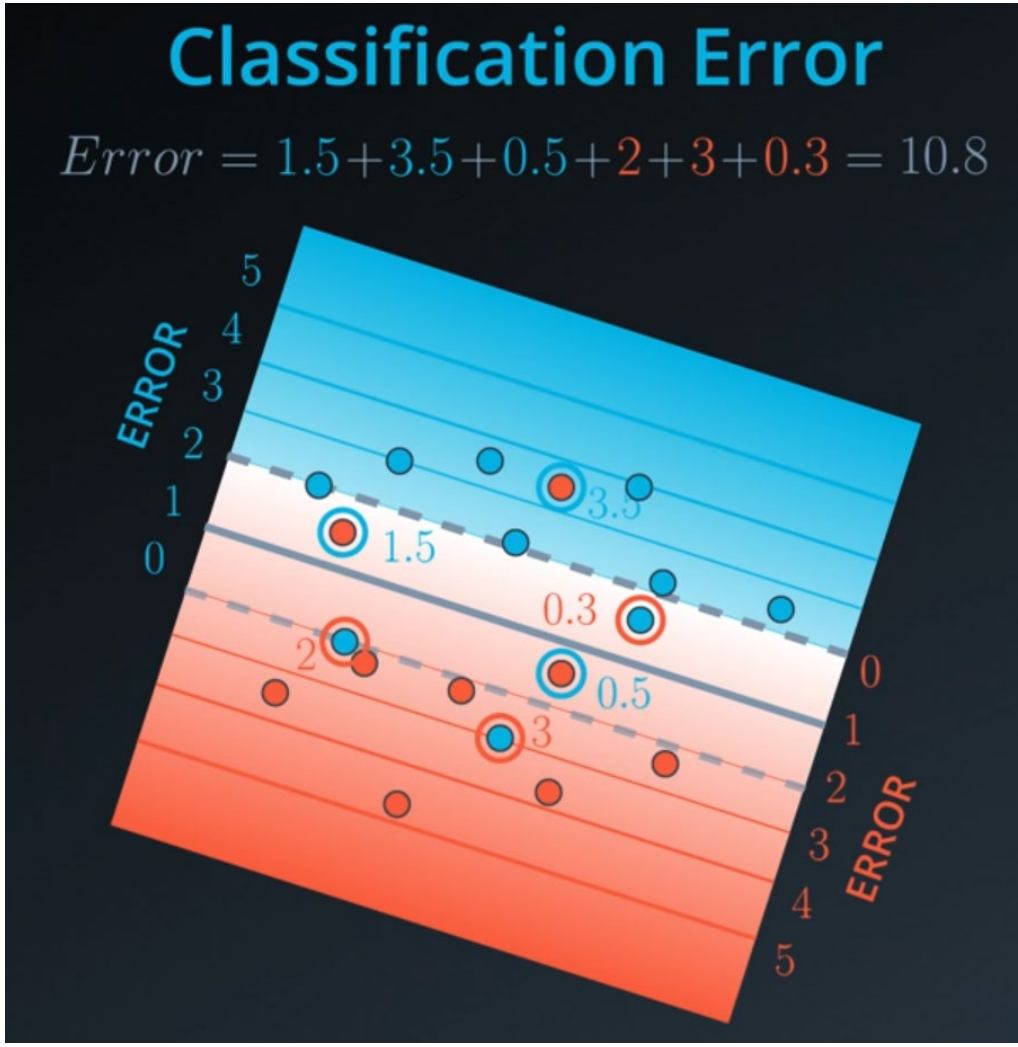
Q: Parametric or non?

	parametric	non
cannon ball	☑	☐
honey bee	☐	☑

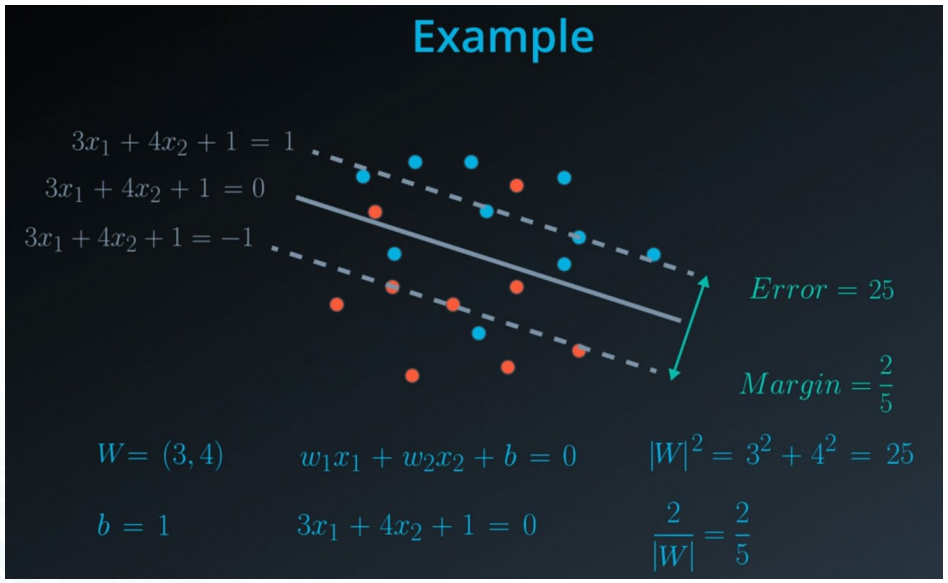


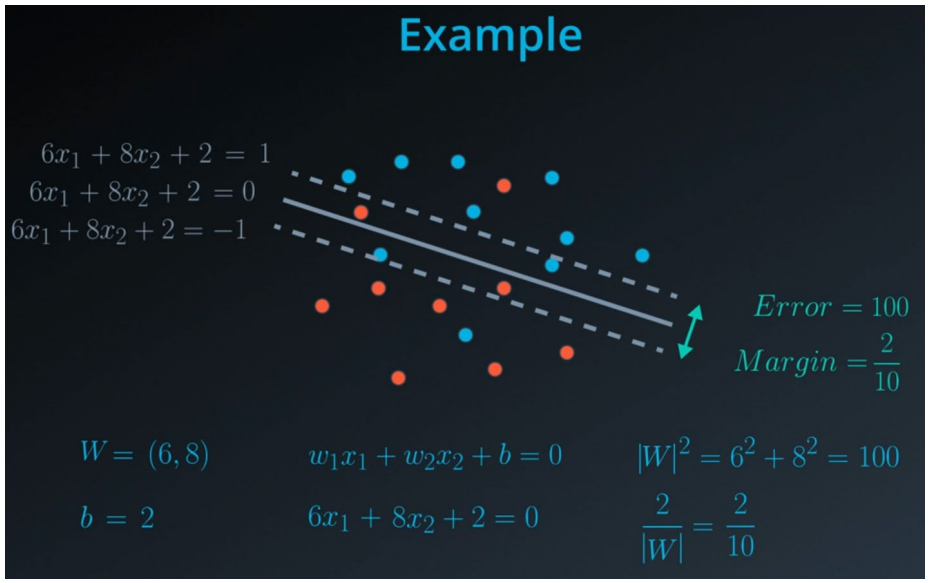
SVMs


$\mathbf{w}^T\mathbf{x}_i + b > 0$
$\mathbf{w}^T\mathbf{x}_i + b < 0$
Margin
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x}_i + b = 0$
$\mathbf{w}^T\mathbf{x}_i + b = 0$
$\mathbf{w}^T\mathbf{x}_i + b > 0$
$\mathbf{w}^T\mathbf{x}_i + b < 0$
Margin
$\mathbf{w}$
: Margin SVs


Classification Error
$Error = 1.5+3.5+0.5+2+3+0.3 = 10.8$
ERROR
0
1
2
3
4
5
3.5
1.5
0.3
2
0.5
3
ERROR

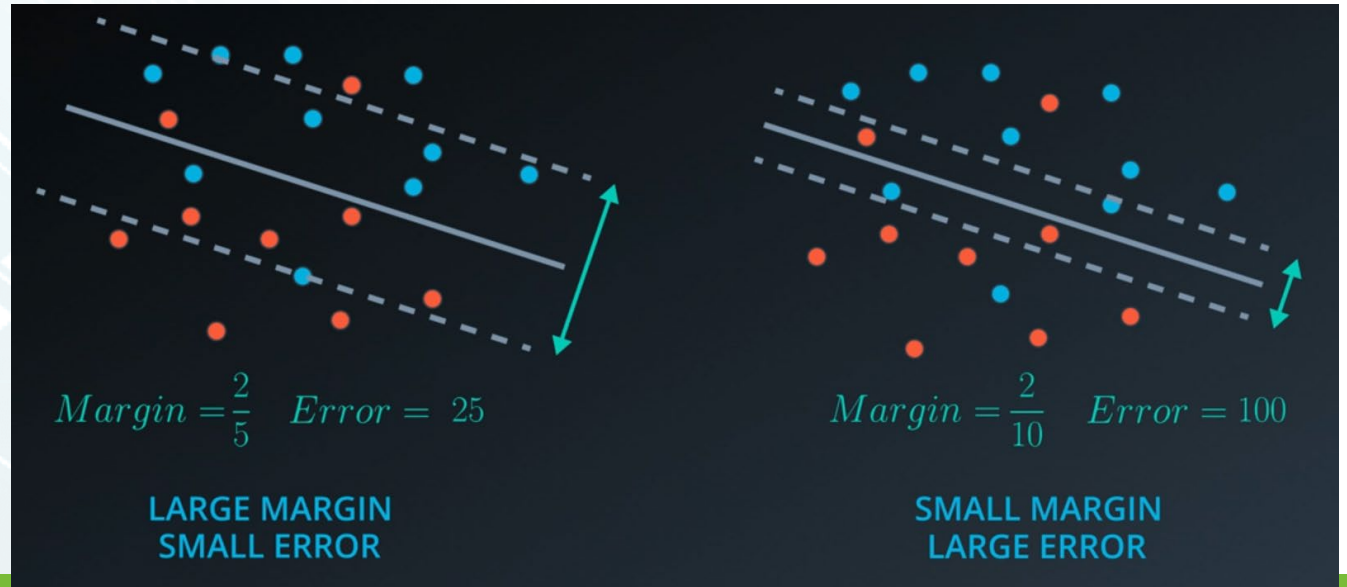
SVM-Example of margin error





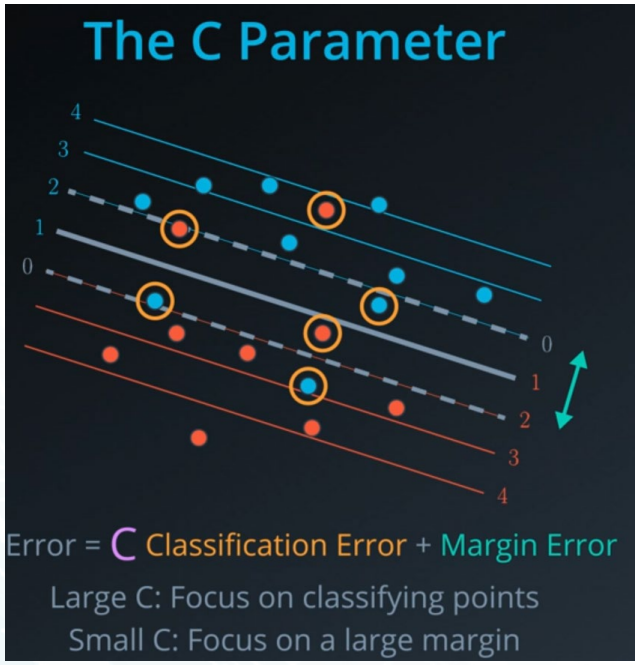
L2 Regularization

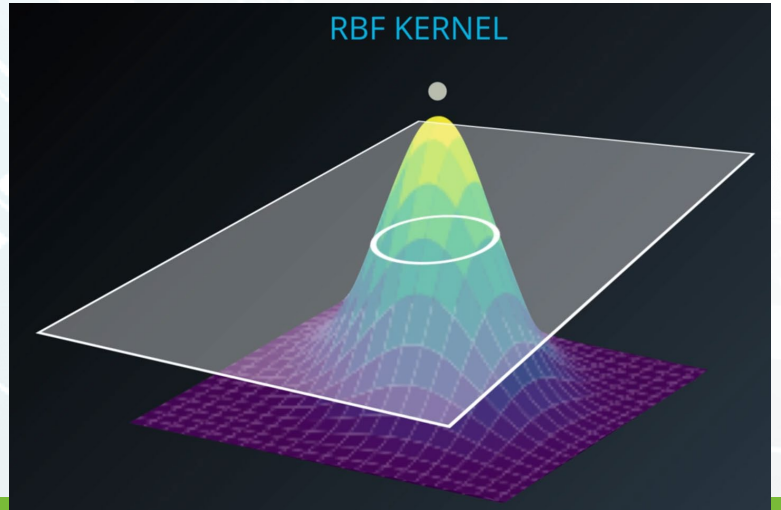
Eror term

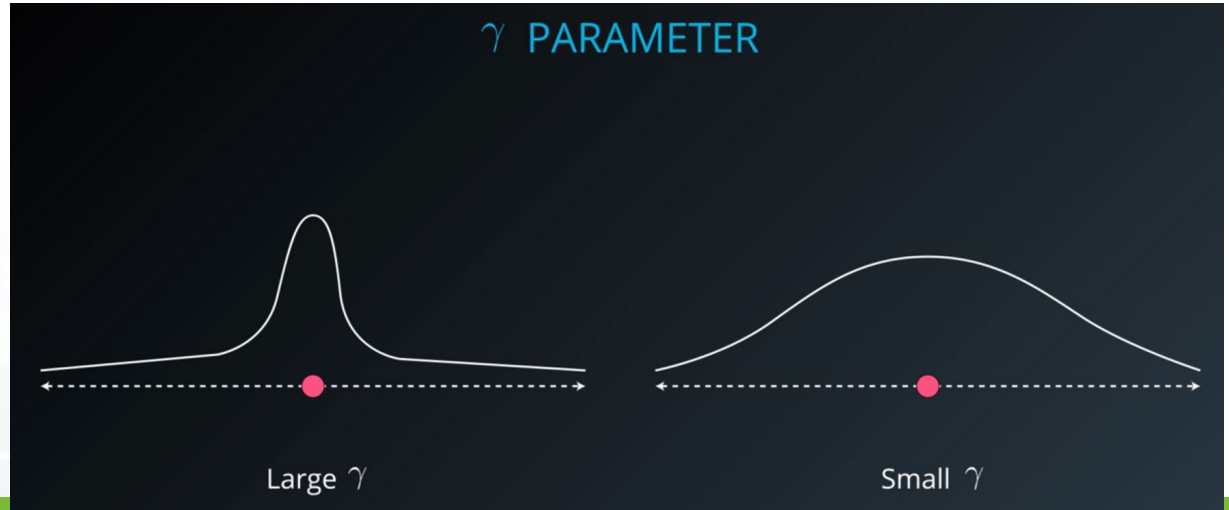


Total error is the sum of classification error and margin eror. Minimize this by using gradient descent

C Parameter and Y in Radial Basis Function







Bias

Restriction bias: Representational power of ML data structure. It tells what we are able to represent with ML algorithm. Answers the question of why do you use this algorithm over others?

- Perceptron is suitable for linearly seperable dataset

Preference bias: ML algorithm's selection of one representation over another

Answer the question of how do we start algorithm?

- Shorted trees preferred over longer and complex ones in decision tree to avoid overfitting
- Gradient descent prefers small inital weights over larger ones to avoid overfitting

Joint Probability Distribution

$$P(X = x_i, Y = y_j) = p_{ij} \qquad i, j = 1, 2, \cdots$$

X \ Y	y_1	y_2	$\cdots$	y_j	$\cdots$
x_1	p_{11}	p_{12}	$\cdots$	p_{1j}	$\cdots$
x_2	p_{21}	p_{22}	$\cdots$	p_{2i}	$\cdots$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	
x_i	p_{i1}	p_{i2}	$\cdots$	p_{ij}	$\cdots$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	

(1) $p_{ij} \geq 0$; (2) $\sum_i \sum_j p_{ij} = 1$

Conditional Independence

Definition: X is conditionally independent of Y given Z if the probability distribution governing X is independent of the value of Y given the value of Z; that is, if

$\forall x,y,z \quad P(X=x \mid Y=y, Z=z) = P(X=x \mid Z=z)$

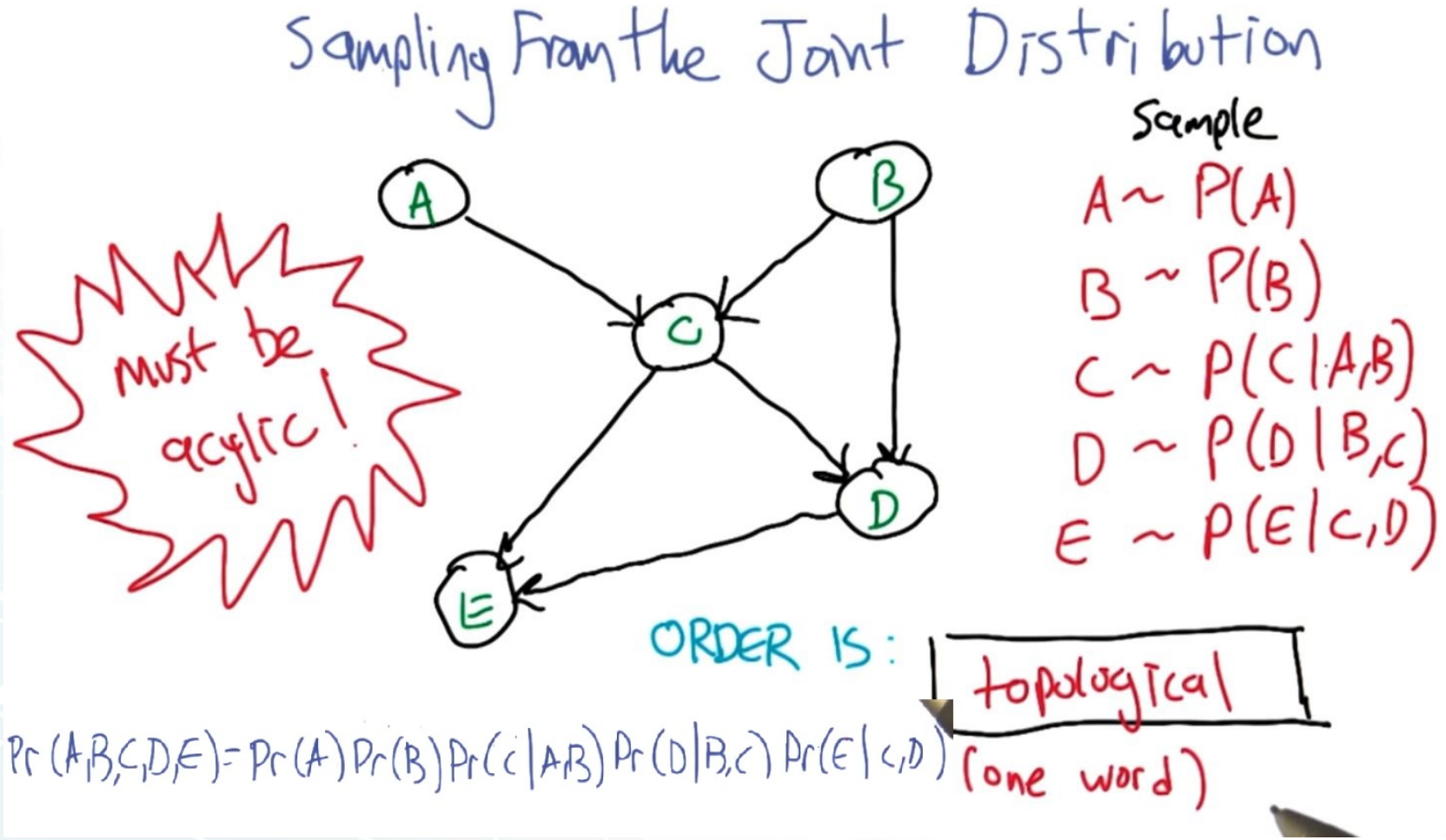
more compactly we write $P(X \mid Y, Z) = P(X \mid Z)$

Independence: $Pr(x,y) = Pr(x) \cdot Pr(y)$

chain rule: $Pr(x,y) = Pr(x|y) \cdot Pr(y)$

$\therefore Pr(x|y) = Pr(x)$ ✓

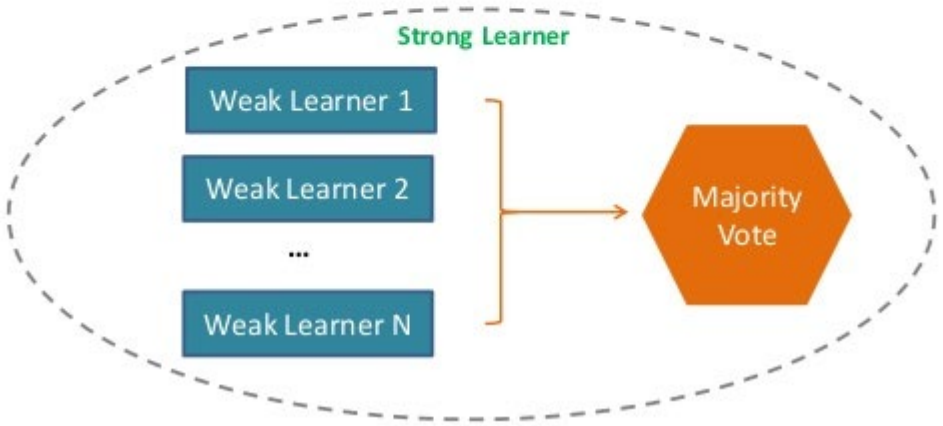
Sampling


Sampling From the Joint Distribution
Sample
A ~ P(A)
B ~ P(B)
C ~ P(C|A,B)
D ~ P(D|B,C)
E ~ P(E|C,D)
A
B
C
D
E
must be acyclic!
ORDER IS:
topological
(one word)
Pr(A,B,C,D,E) = Pr(A) Pr(B) Pr(C|A,B) Pr(D|B,C) Pr(E|C,D)

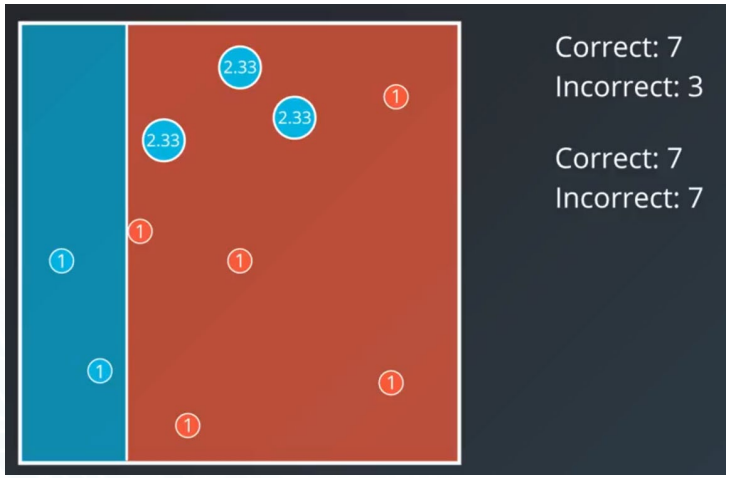
Ensemble Learning

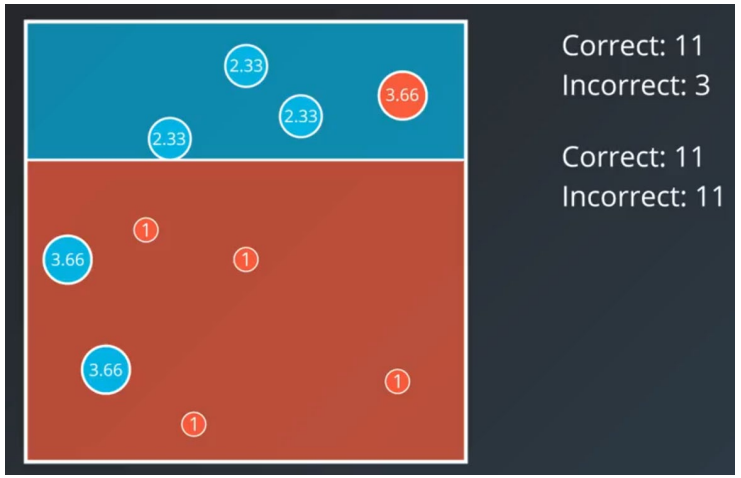
Weak Learners

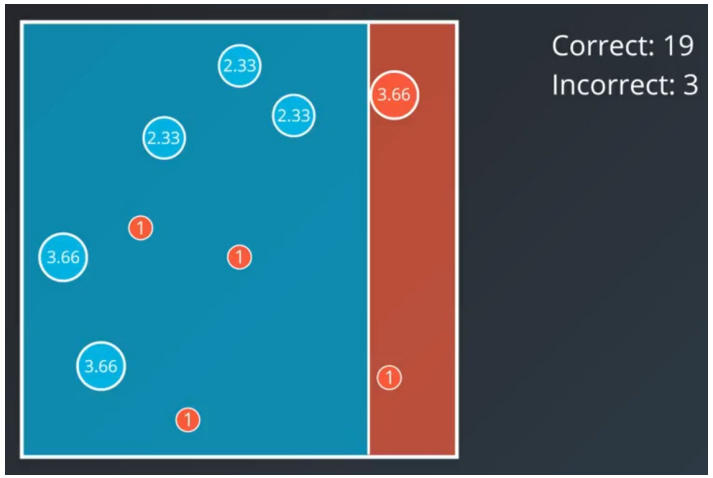
- In an Ensemble method, one combines multiple weak learners to make a strong learning model.
- A weak learner is any model that has an accuracy of better than random, even if it is just slightly better (e.g., 0.51).

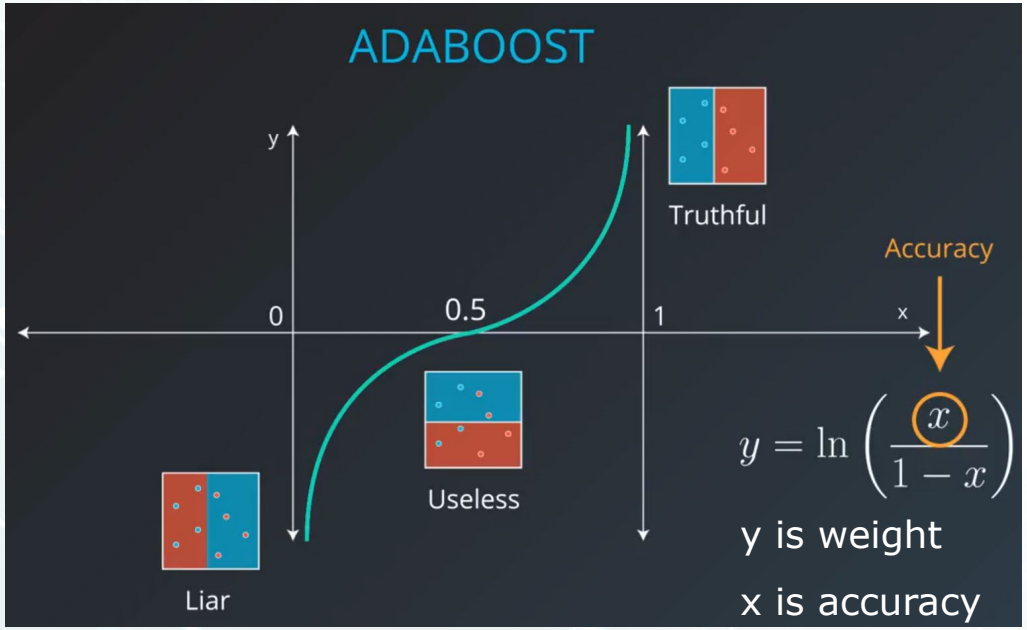


Weighting the data & the model





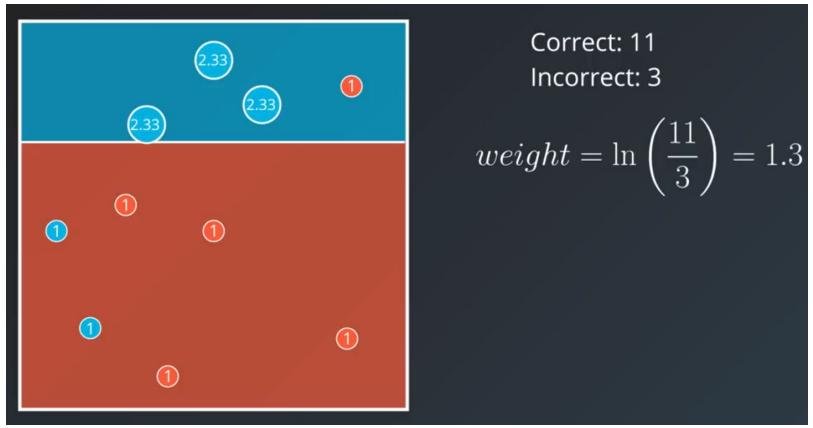




Combine the models

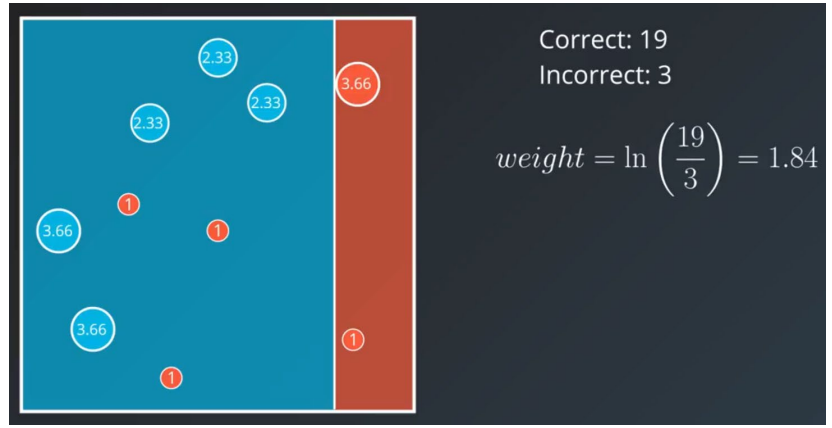
Correct: 7
Incorrect: 3

$$weight = \ln\left(\frac{7}{3}\right) = 0.84$$



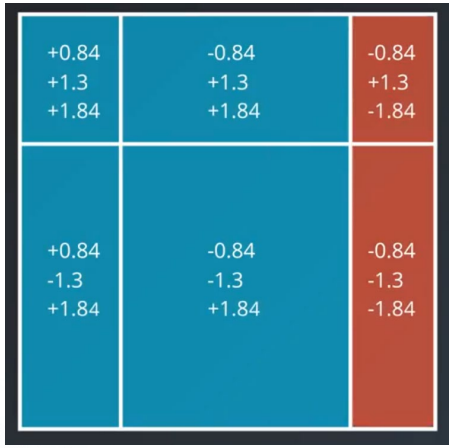
Correct: 11
Incorrect: 3

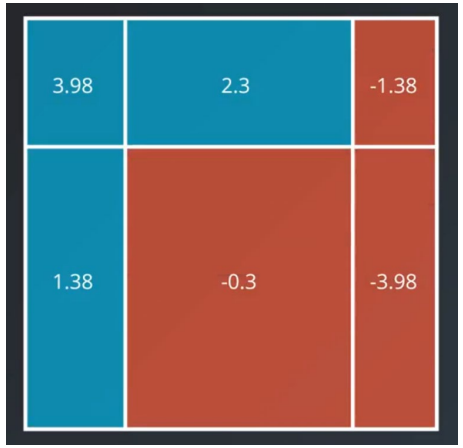
$$weight = \ln\left(\frac{11}{3}\right) = 1.3$$

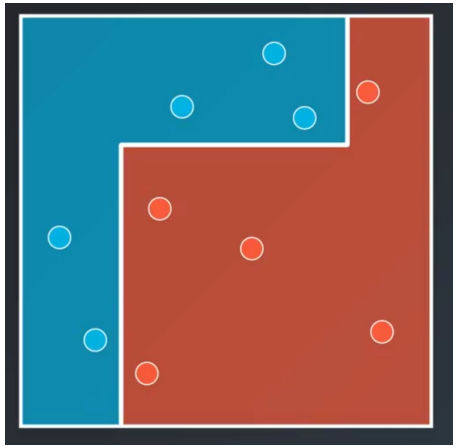


Correct: 19
Incorrect: 3

$$weight = \ln\left(\frac{19}{3}\right) = 1.84$$







PCA for facial recognition

PCA for Facial Recognition

What makes facial recognition in pictures good for PCA?

- ☑ pictures of faces generally have high input dimensionality (many pixels)
- ☑ faces have general patterns that could be captured in smaller number of dimensions (two eyes on top, mouth/chin on bottom, etc.)
- ☐ facial recognition is simple using machine learning (humans do it easily)

RANDOM PROJECTION

from d dimensions to k dimensions

Johnson-Lindenstrauss lemma

A dataset of N points in high-dimensional Euclideanspace can be mapped down to a space in much lower dimension in a way that preserves the distance between the points to a large degree.

Epsilon is the input the function that we can use to say preserve the distances by this degree

12,000

N

docID	a2i	aaa	aaai	aapo	aat	aazhang	abandonment	abbott	…	zorn	zornetzer	zoubin	zucker	zue	zur	zurich	zwislocki
1																	
2																	
3	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0

$$\times \begin{bmatrix} r_{11} & \dots & r_{1d} \\ \dots & \dots & \dots \\ r_{k1} & \dots & r_{kd} \end{bmatrix}$$

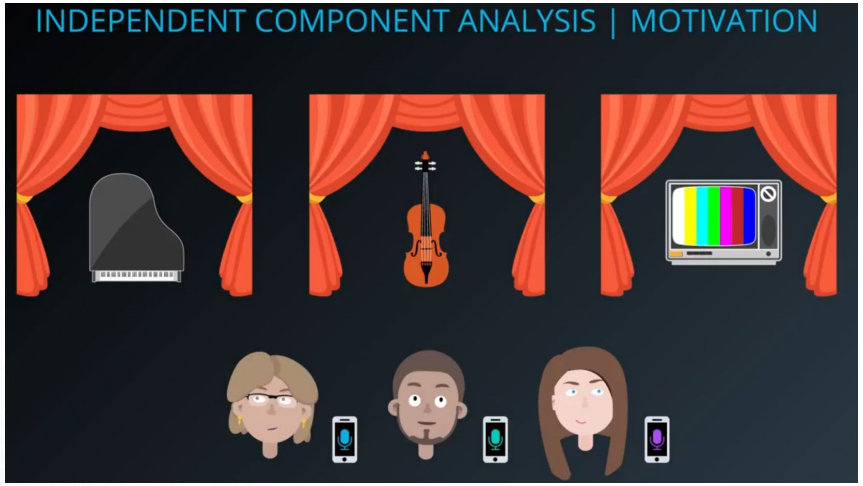
6,268

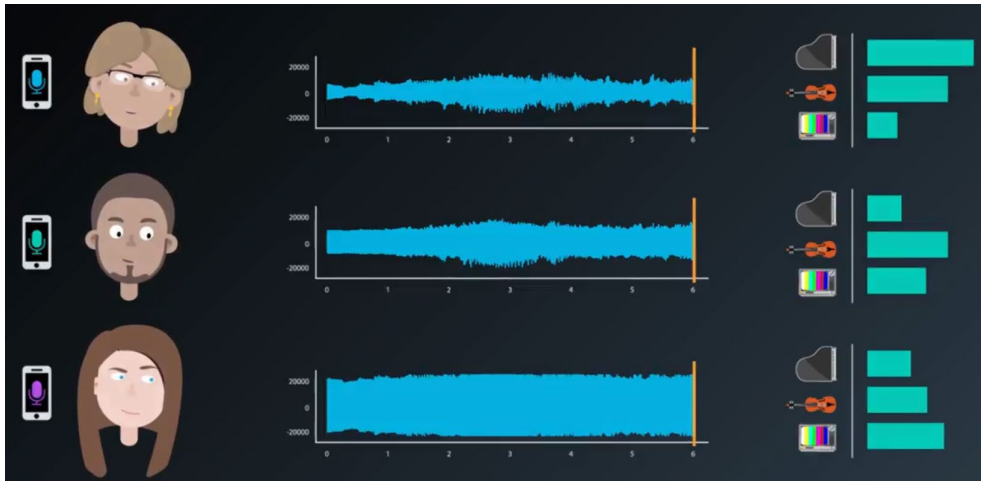
N

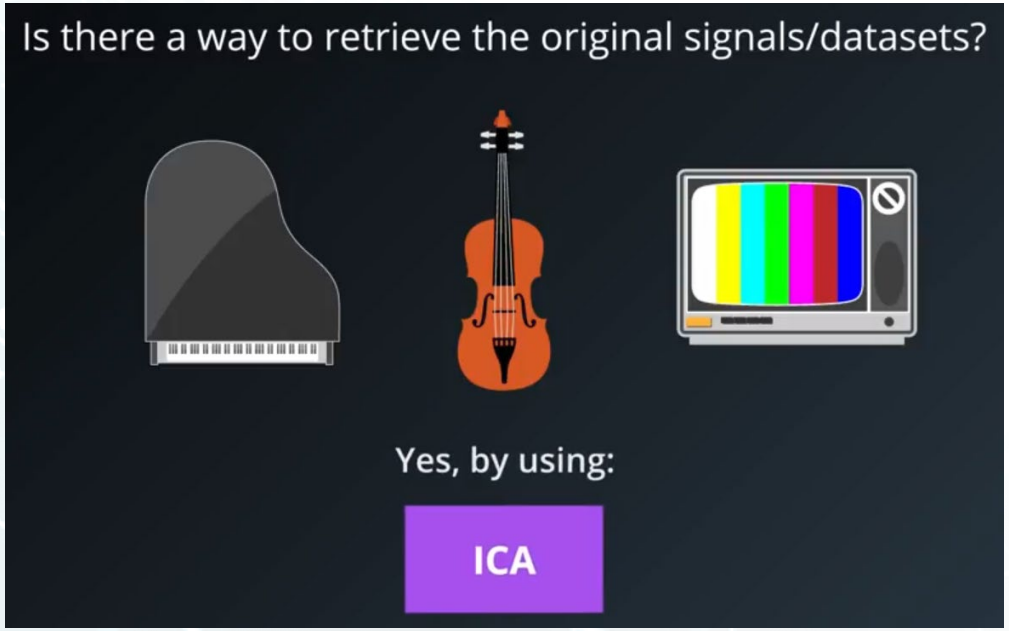
	0	1	2	3	4	…	6263
0							
1							
2	-0.932545	0.000000	0.133221	0.666104	0.133221	…	0.133221
3	0.133221	-2.131532	0.399662	0.532883	-0.399662	…	1.065766
4	0.399662	-2.131532	0.000000	-0.532883	0.399662	…	1.065766

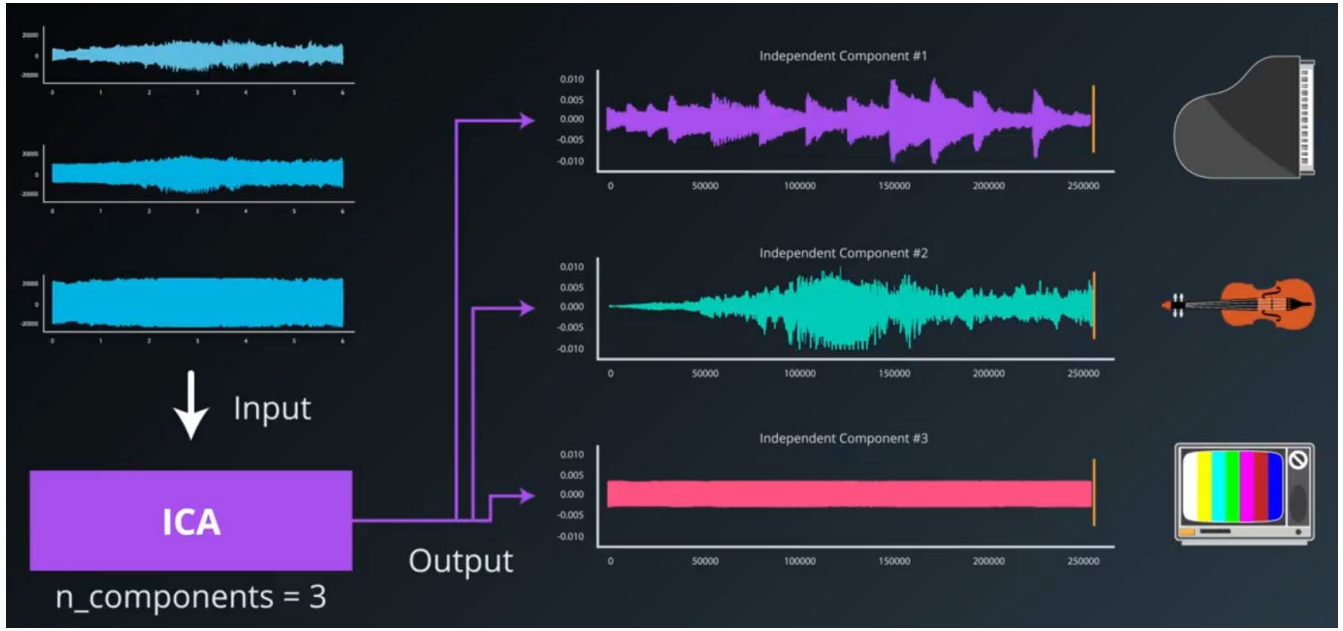
$$\underset{0.9 \times (125.6)^2}{(1-eps)\,\|u-v\|^2 \overset{0.1 \quad 125.6}{<}} \underset{(125.8)^2}{\overset{125.8}{\|p(u)-p(v)\|^2}} \underset{1.1 \times (125.6)^2}{\overset{0.1 \quad 125.6}{<}(1+eps)\,\|u-v\|^2}$$

ICA-Blind source separation problem

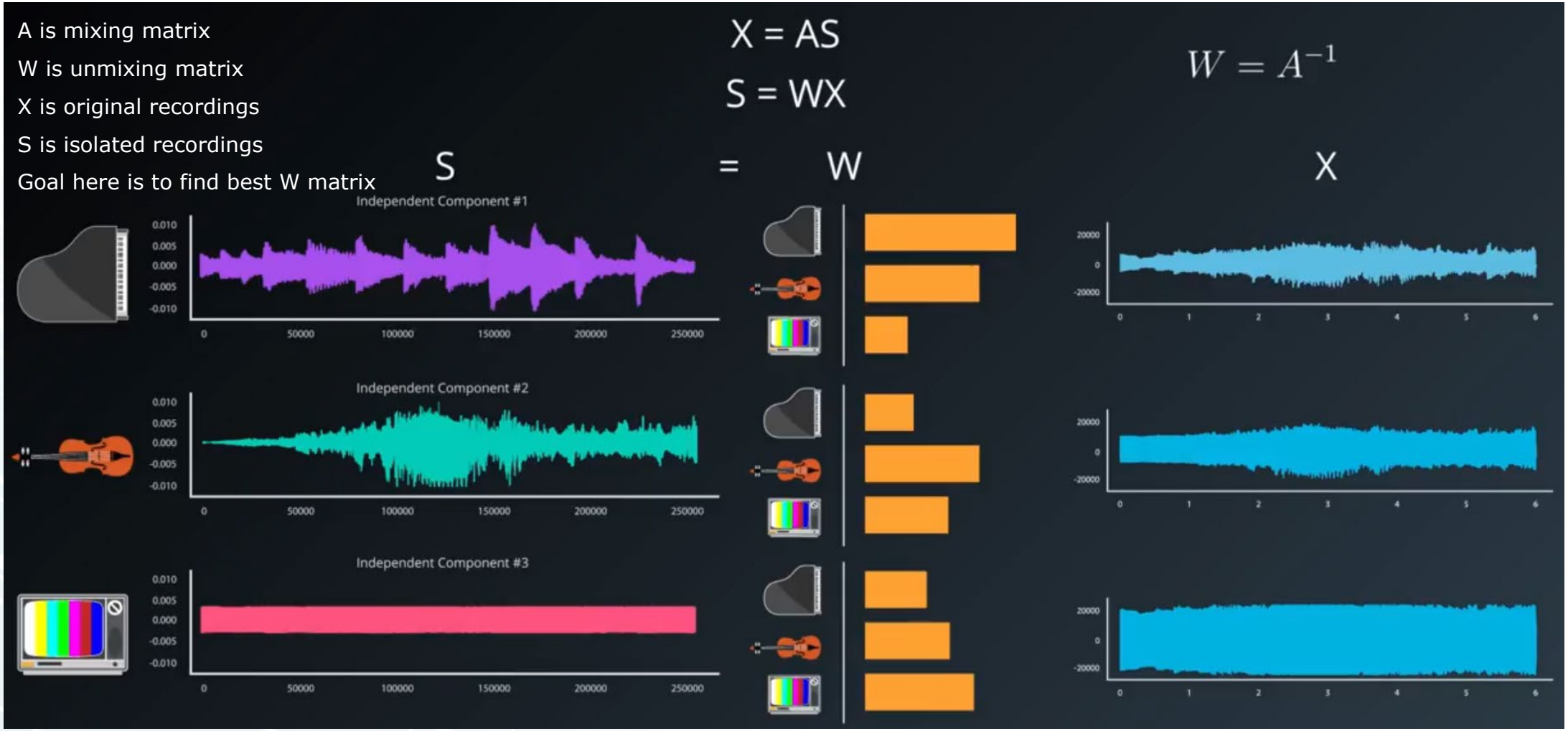




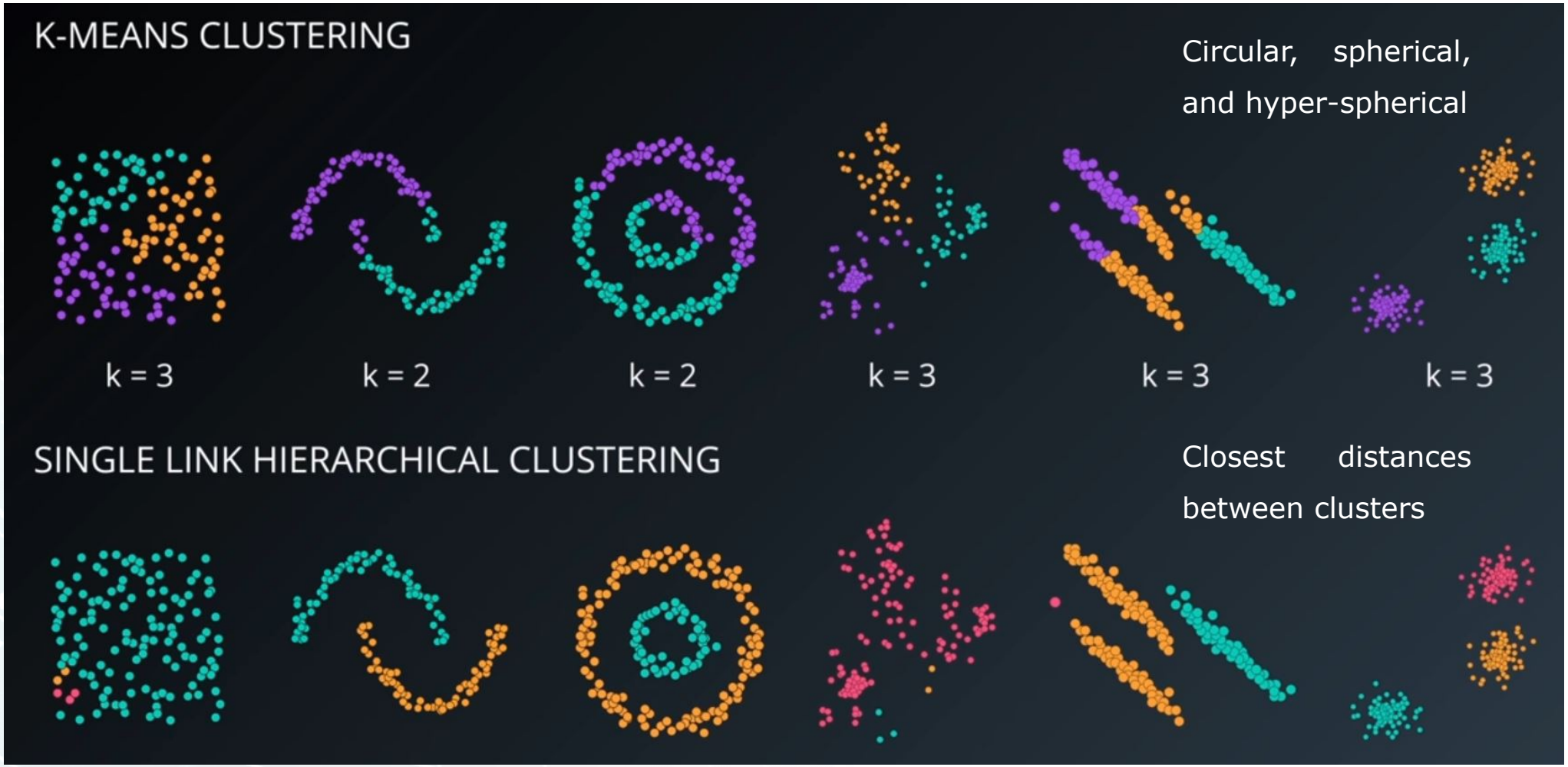




How to find isolated sounds

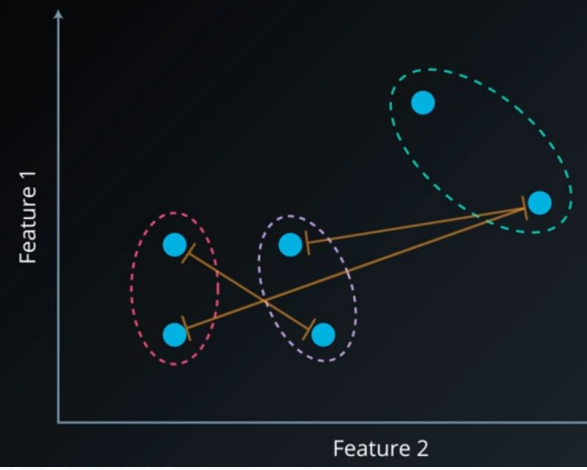


K-means vs Single link clustering result



Clustering

DISTANCE MEASURE | COMPLETE LINK

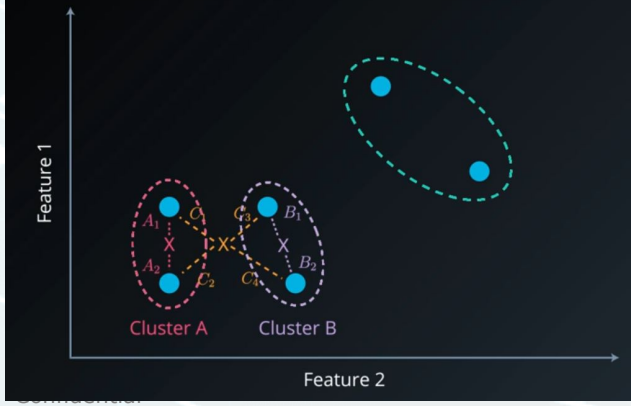


Farthest distance between clusters

DISTANCE MEASURE | AVERAGE LINK

Average all possible distances between clusters

WARD'S METHOD



DISTANCE BETWEEN CLUSTERS A AND B

$$\Delta(A,B) = C_1^2 + C_2^2 + C_3^2 + C_4^2 - A_1^2 - A_2^2 - B_1^2 - B_2^2$$

Minimize variance

Default in scikit

HIERARCHICAL CLUSTERING

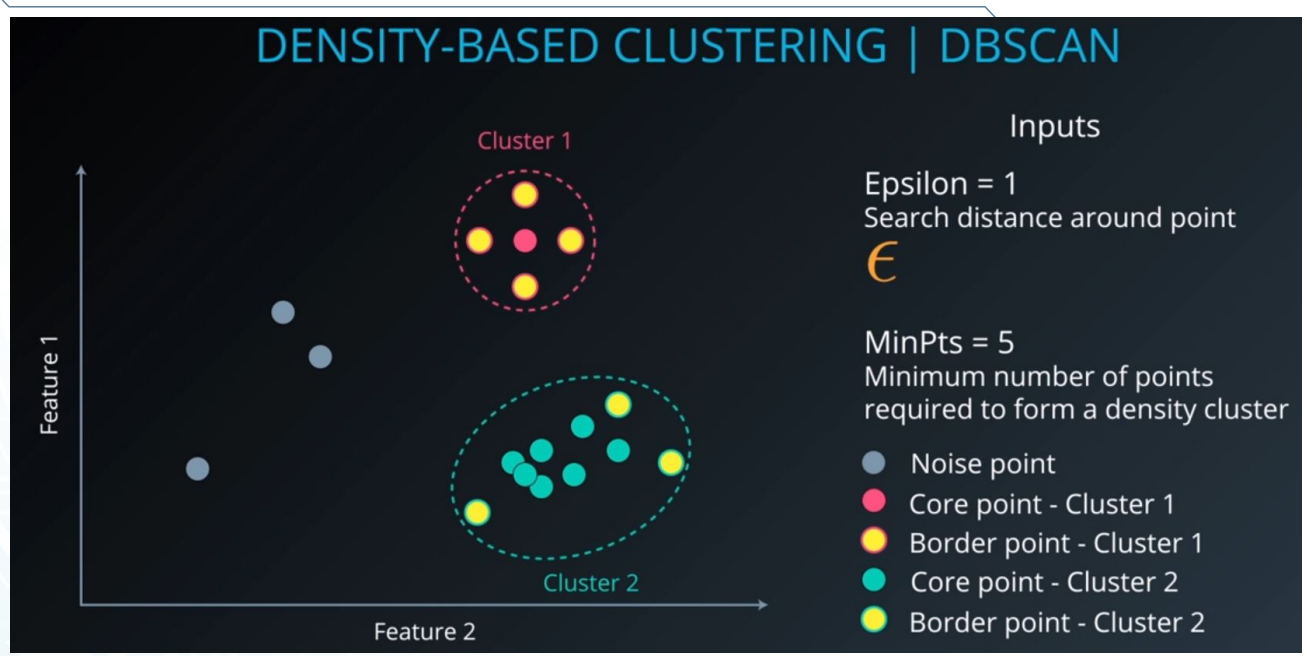
ADVANTAGES:

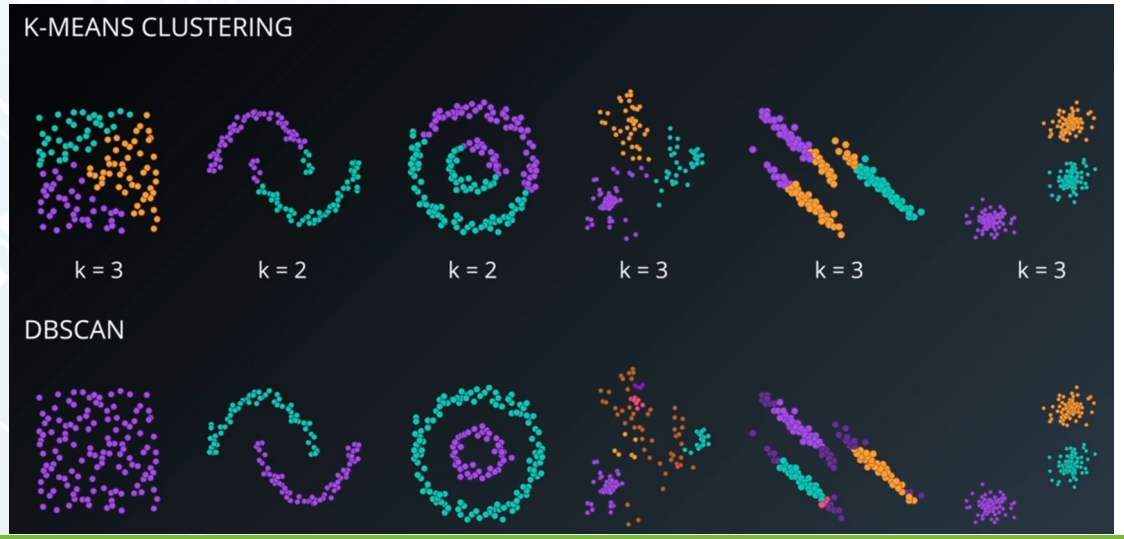
- Resulting hierarchical representation can be very informative
- Provides an additional ability to visualize
- Especially potent when the dataset contains real hierarchical relationships (e.g. Evolutionary biology)

DISADVANTAGES:

- Sensitive to noise and outliers
- Computationally intensive $O(N^2)$

DBSCAN

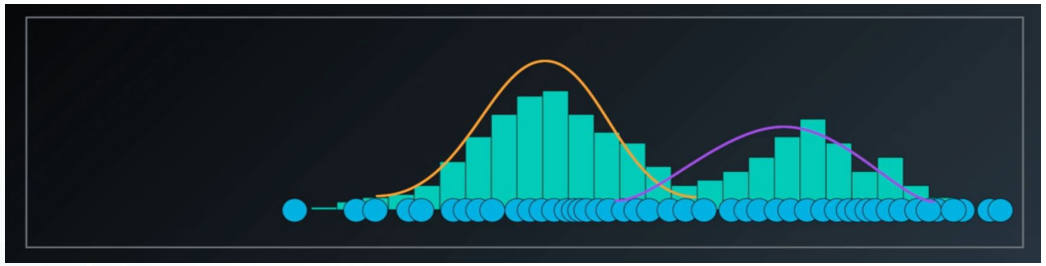




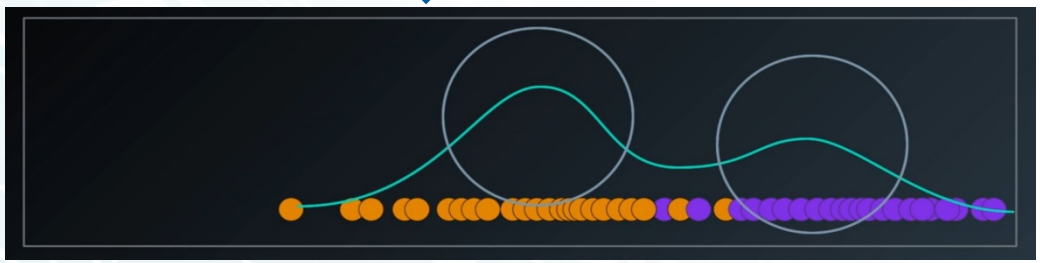
ADVANTAGES:
- We don't need to specify the number of clusters
- Flexibility in the shapes & sizes of clusters
- Able to deal with noise
- Able to deal with outliers

DISADVANTAGES:
- Border points that are reachable from two clusters
- Faces difficulty finding clusters of varying densities

Gaussian mixture model clustering 1D/2D exp

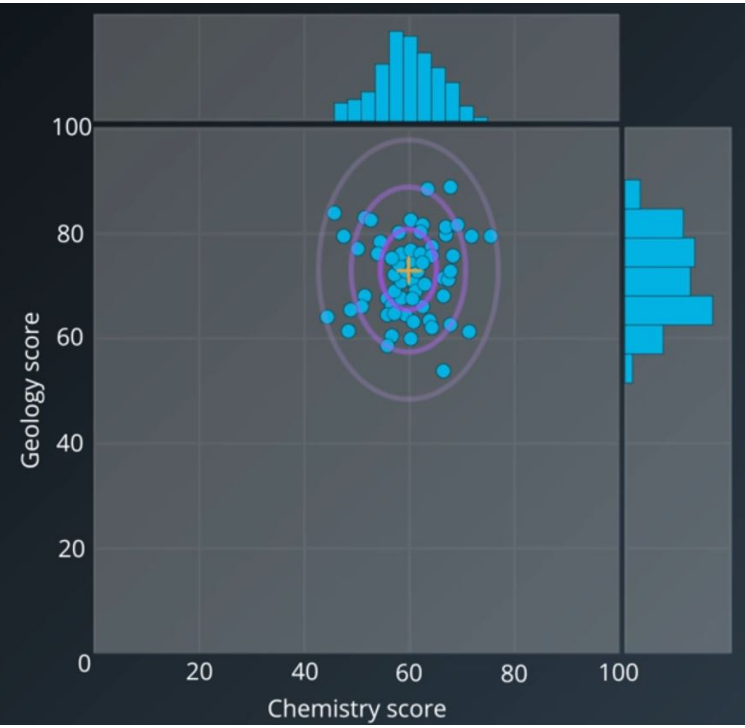


Two different gaussian distributions are together without their labels. Feed this into sckitlearn to give two clusters



MULTIVARIATE GAUSSIAN DISTRIBUTION

Student ID	Chemistry Score	Geology Score
1	66	83
2	61	73
3	57	87
4	61	72
...	...	...
99	60	76
MEAN	59	78
STD DEVIATION	5	8



Gaussian mixture algorithm

Expectation - Maximization For Gaussian Mixtures:

Dataset to cluster into two clusters

STEP #1: INITIALIZE K GAUSSIAN DISTRIBUTIONS

STEP #2: SOFT-CLUSTER DATA - "EXPECTATION"

STEP #3: RE-ESTIMATE THE GAUSSIANS - "MAXIMIZATION"

STEP #4: EVALUATE LOG-LIKELIHOOD TO CHECK FOR CONVERGENCE

REPEAT FROM STEP #2 UNTIL CONVERGED

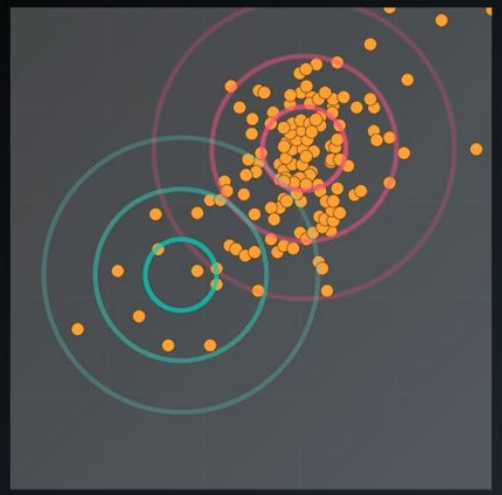
GAUSSIAN MIXTURE MODEL CLUSTERING

Advantages:
- Soft-clustering (sample membership of multiple clusters)
- Cluster shape flexibility

Disadvantages:
- Sensitive to initialization values
- Possible to converge to a local optimum
- Slow convergence rate

How algorithm works

Step 1 - Initialize Gaussian Distributions



Initial Gaussian Parameters

Cluster	μ	σ^2
A	(64.63, 76.30)	100
B	(46.02, 51.30)	57

Step 2 - Soft-cluster the data points - "Expectation" step

SOFT CLUSTERING ("RESPONSIBILITIES")

Point #	feature 1	feature 2	Cluster A	Cluster B
1	62	71	0.99976	0.00024
2	58	81		
3	52	74		
...	...	...		
N	52	78		

$$E[Z_{1A}] = \frac{N(X_i|\mu_A, \sigma_A^2)}{N(X_i|\mu_A, \sigma_A^2) + N(X_i|\mu_B, \sigma_B^2)} = \frac{0.001288}{0.001288 + 0.0000038}$$

$$N(X|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^2} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Cluster	μ	σ^2
A	(64.63, 76.30)	100
B	(46.02, 51.30)	57

Step 3 - Re-estimate parameters of Gaussians - "Maximization" step

NEW GAUSSIAN PARAMETERS

Cluster	new μ	new σ^2
A	(64.4872457, 76.3074590)	103.92494596
B	(46.0271498, 51.3087720)	67.10773268

$$\text{new } \sigma_A^2 = \frac{\sum_{i=1}^{N} E[Z_{iA}](X_i - \mu_A^{new})(X_i - \mu_A^{new})^T}{\sum_{i=1}^{N} E[Z_{iA}]}$$

$$= 103.92494596$$

Point #	feature 1	feature 2	Cluster A	Cluster B
1	62	71	0.99937	0.00063
2	58	81	0.9998	0.0002
3	52	74	0.55818	0.44182
...	...	...	...	...
N	52	78	0.99133	0.00867

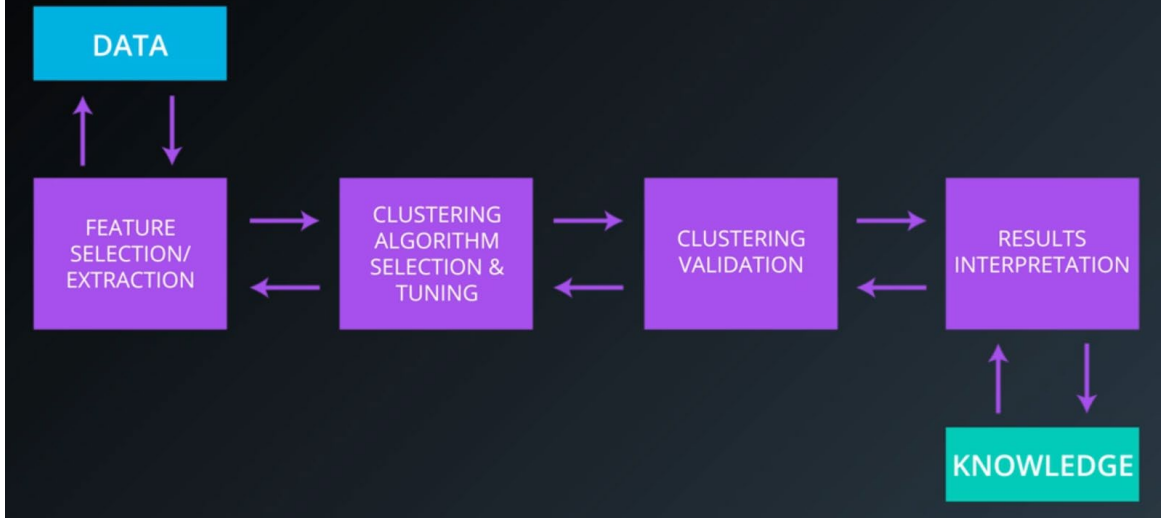
$$\text{new } \mu_A = \frac{\sum_{i=1}^{N} E[Z_{ij}]X_i}{\sum_{i=1}^{N} E[Z_{ij}]} = \frac{0.99937\text{x}(62\ 71) + 0.9998\text{x}(58\ 81) + \ldots}{0.99937 + 0.9998 + 0.55818 + \ldots}$$

$$= (64.4872457, 76.3074590)$$

Step 4 - Evaluate log-likelihood

$$\ln p(X|\mu, \sigma^2) = \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k N(X_i|\mu_k, \sigma_k^2)\right)$$

CLUSTER ANALYSIS



CLUSTER VALIDATION | EXTERNAL INDICES

Matching a clustering structure to information we know beforehand.

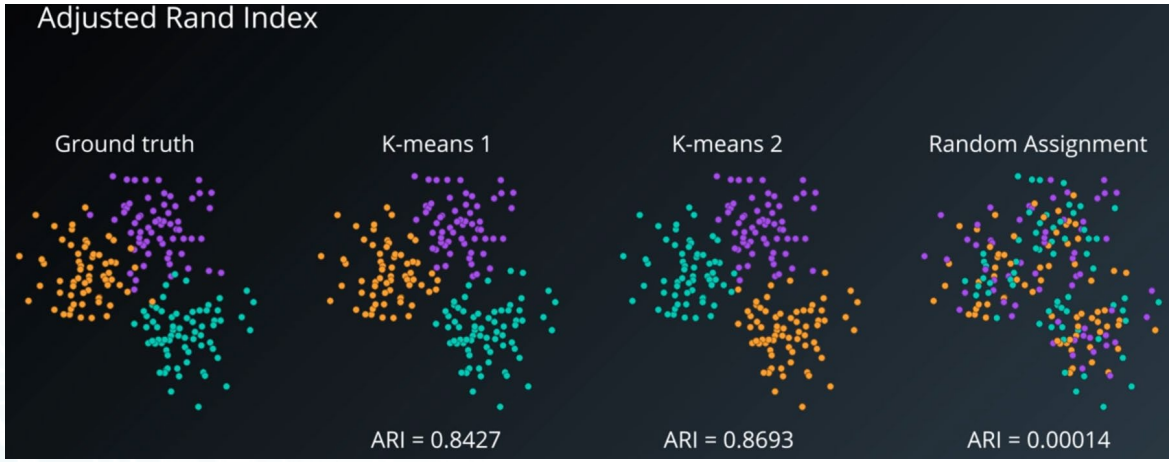
Index	Range	Available in sklearn
Adjusted Rand Score	[-1,1]	✓
Fawlks and Mallows	[0,1]	✓
NMI measure	[0,1]	✓
Jaccard	[0,1]	✓
F-measure	[0,1]	✓
Purity	[0,1]	

CLUSTER VALIDATION | INTERNAL INDICES

Measure the fit between the data and the structure using only the data

Index	Range	Available in sklearn
Silhouette index	[-1,1]	✓
Calinski-Harabasz		✓
BIC		
Dunn Index		

Silhiutte coefficient



Clustering can be used for measuring the responses of the specific customer segments to any change in a product. It is called A/B tests.