Choosing the Right Machine Learning Solution

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Choosing and evaluating

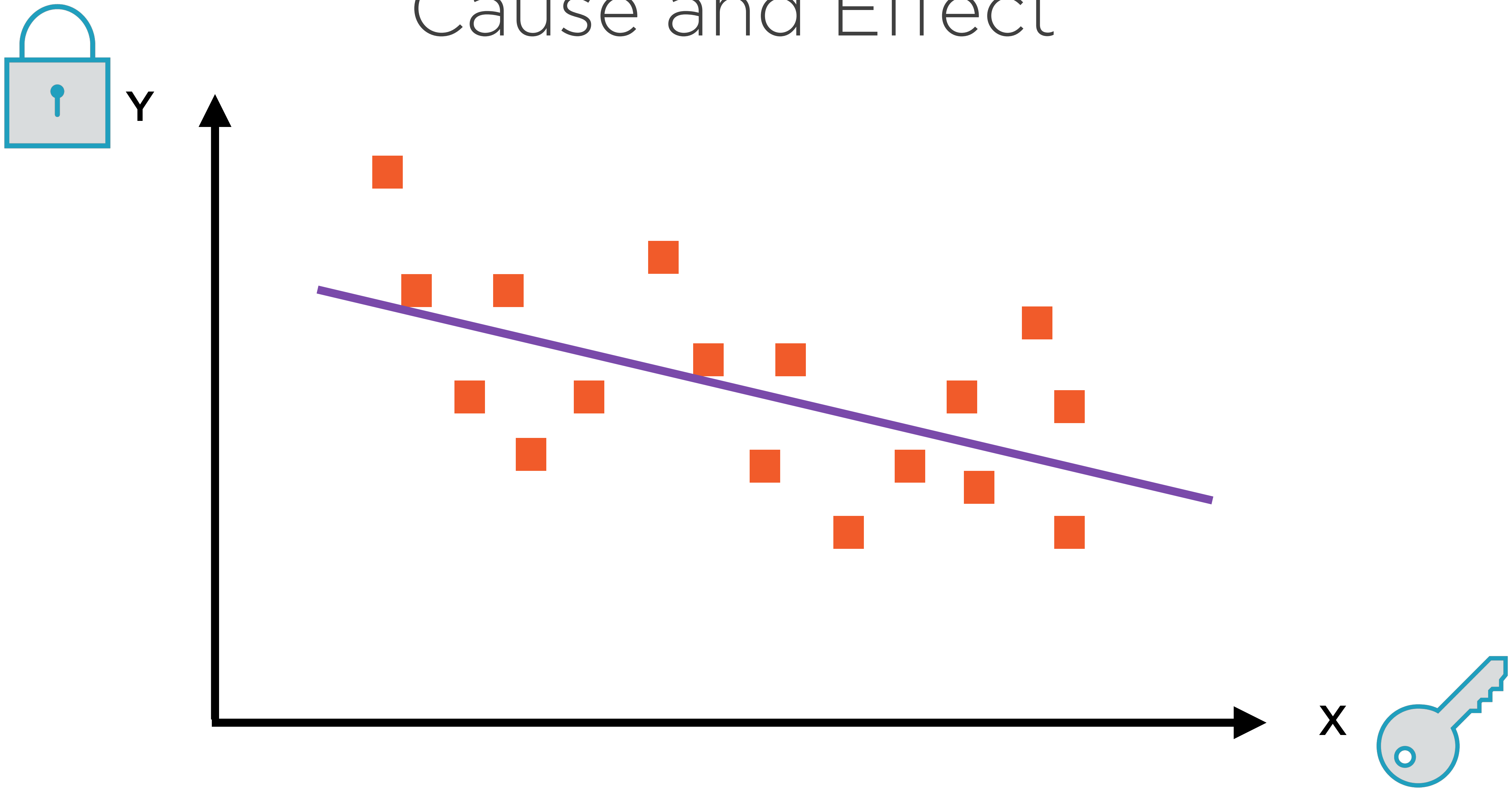
- Regression models
- Classification models
- Clustering models
- Dimensionality reduction techniques

Broad Problem Categories

Use-case	Problem
Predict continuous values	Regression
Predict categorical values	Classification
Find patterns within data - no y-values	Clustering, dimensionality reduction
Simplify complex x-data	Dimensionality reduction

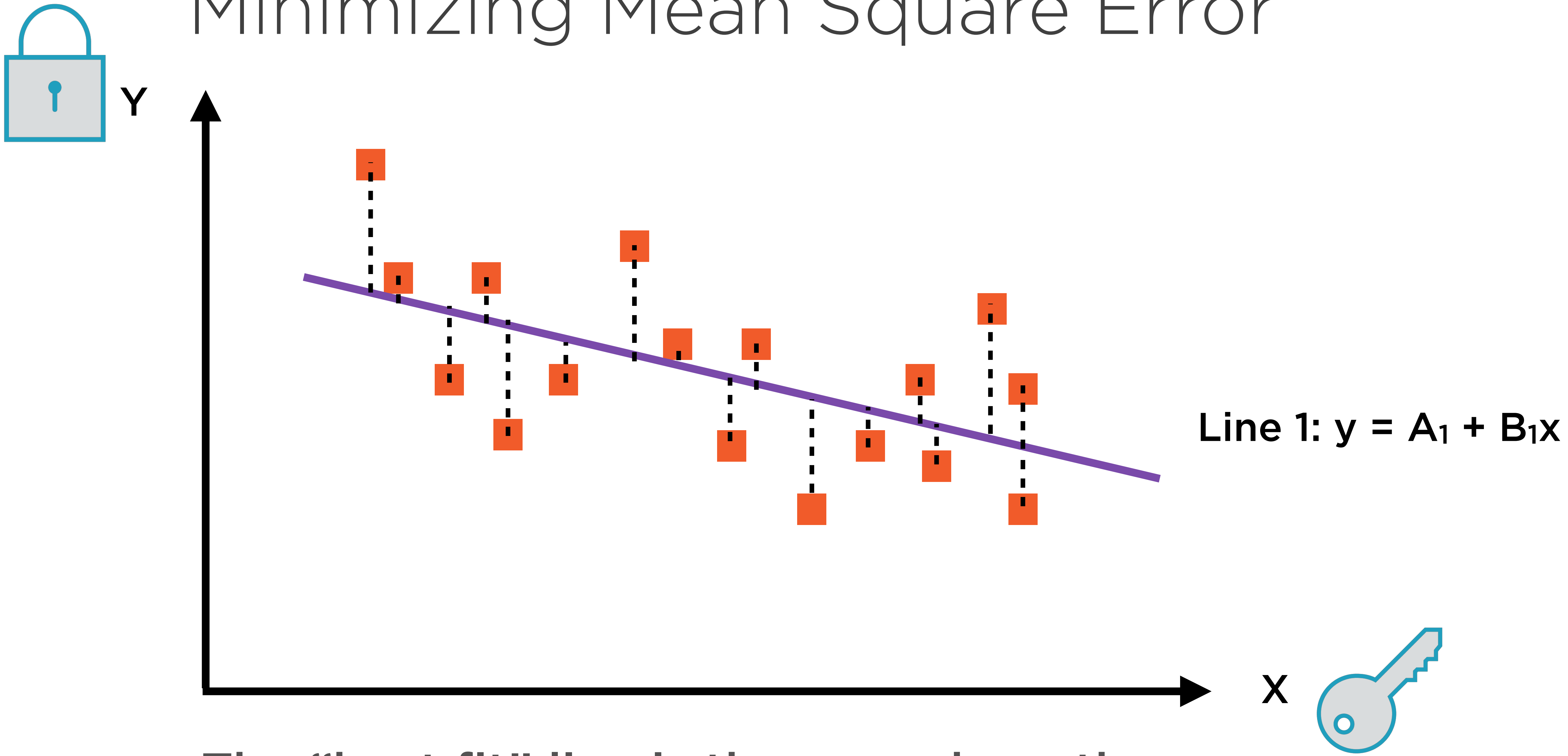
Regression Models

Cause and Effect



Linear Regression involves finding the “best fit” line

Minimizing Mean Square Error

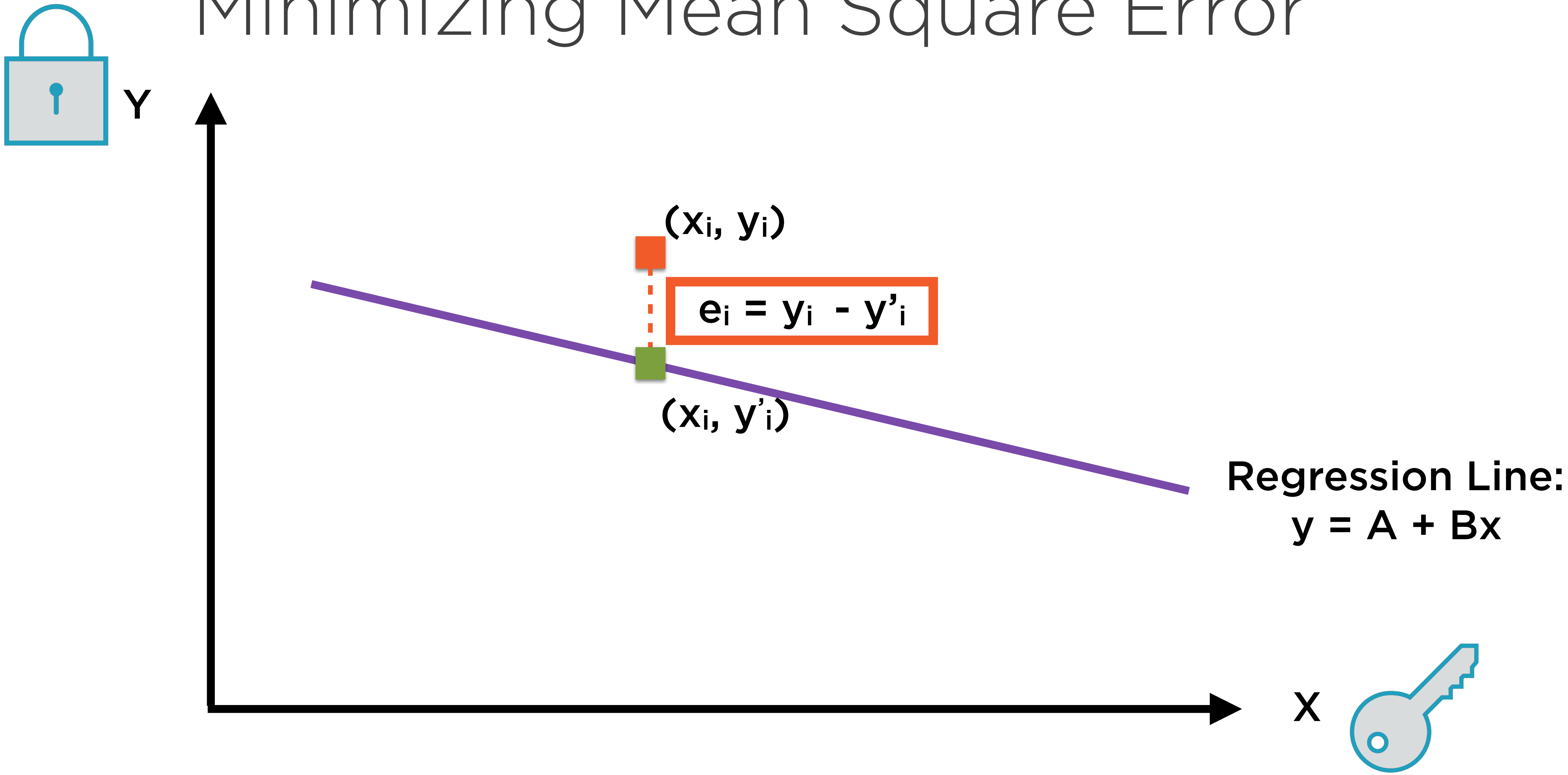


The "best fit" line is the one where the sum of the squares of the lengths of these dotted lines is minimum

The “best fit” line is the one where the sum of the squares of the lengths of the errors is minimized

Finding this line is the objective of the regression problem

Minimizing Mean Square Error

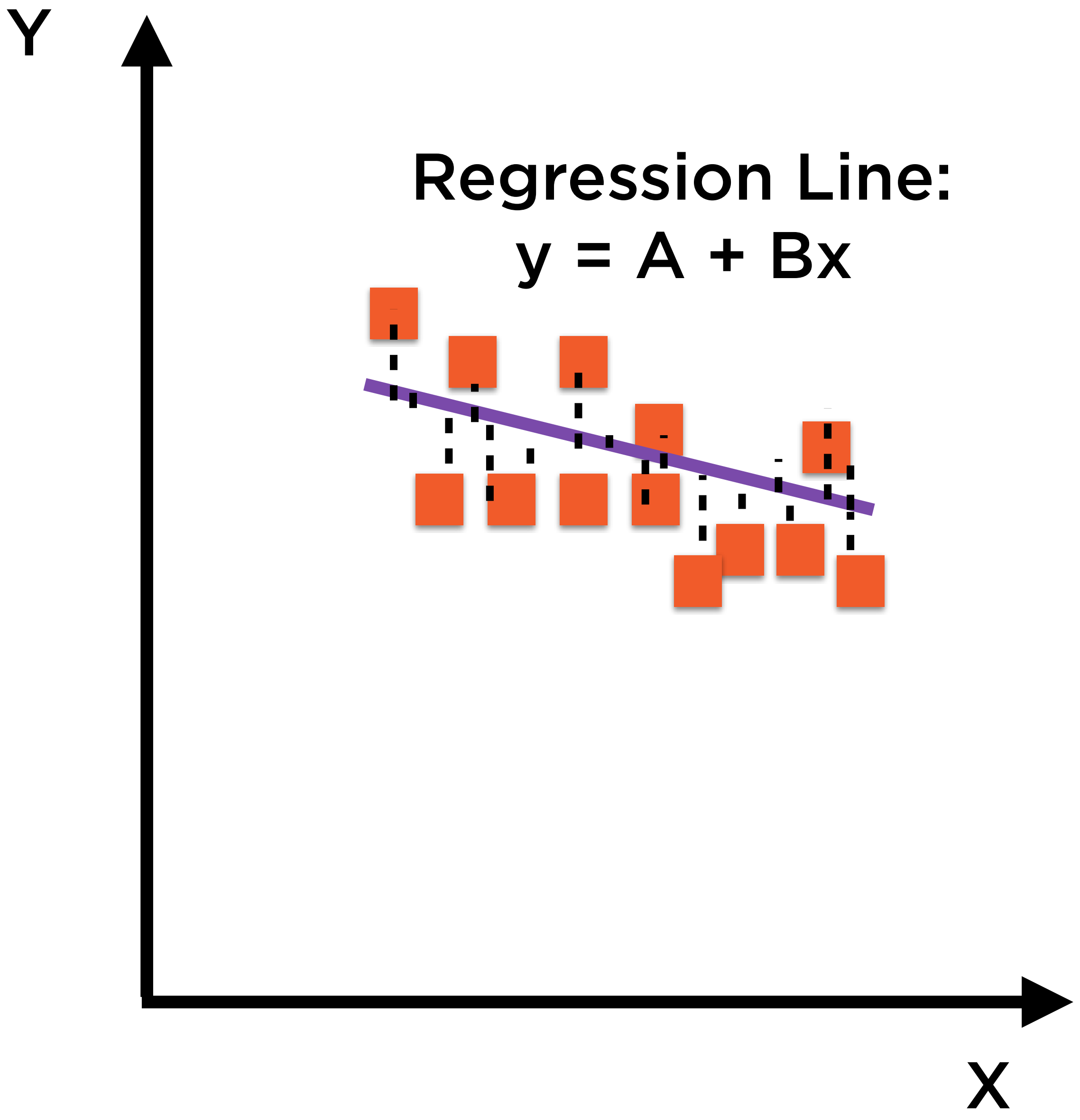


Residuals of a regression are the difference between actual and fitted values of the dependent variable

To find the "best fit" line we need to make some assumptions about regression error

There is a fine distinction between errors and residuals - but we can ignore it

Regression Assumptions



Ideally, residuals should

- have zero mean
- common variance
- be independent of each other
- be independent of x
- be normally distributed

Choosing Regression Algorithms

Choosing Regression Algorithms

Size of Dataset

Number of Features	Small	Medium	Large
Many			
Moderate			
Few			

Number of Features

Choosing Regression Algorithms

Size of Dataset

	Small	Medium	Large
Many			
Moderate			
Few			

Number of Features

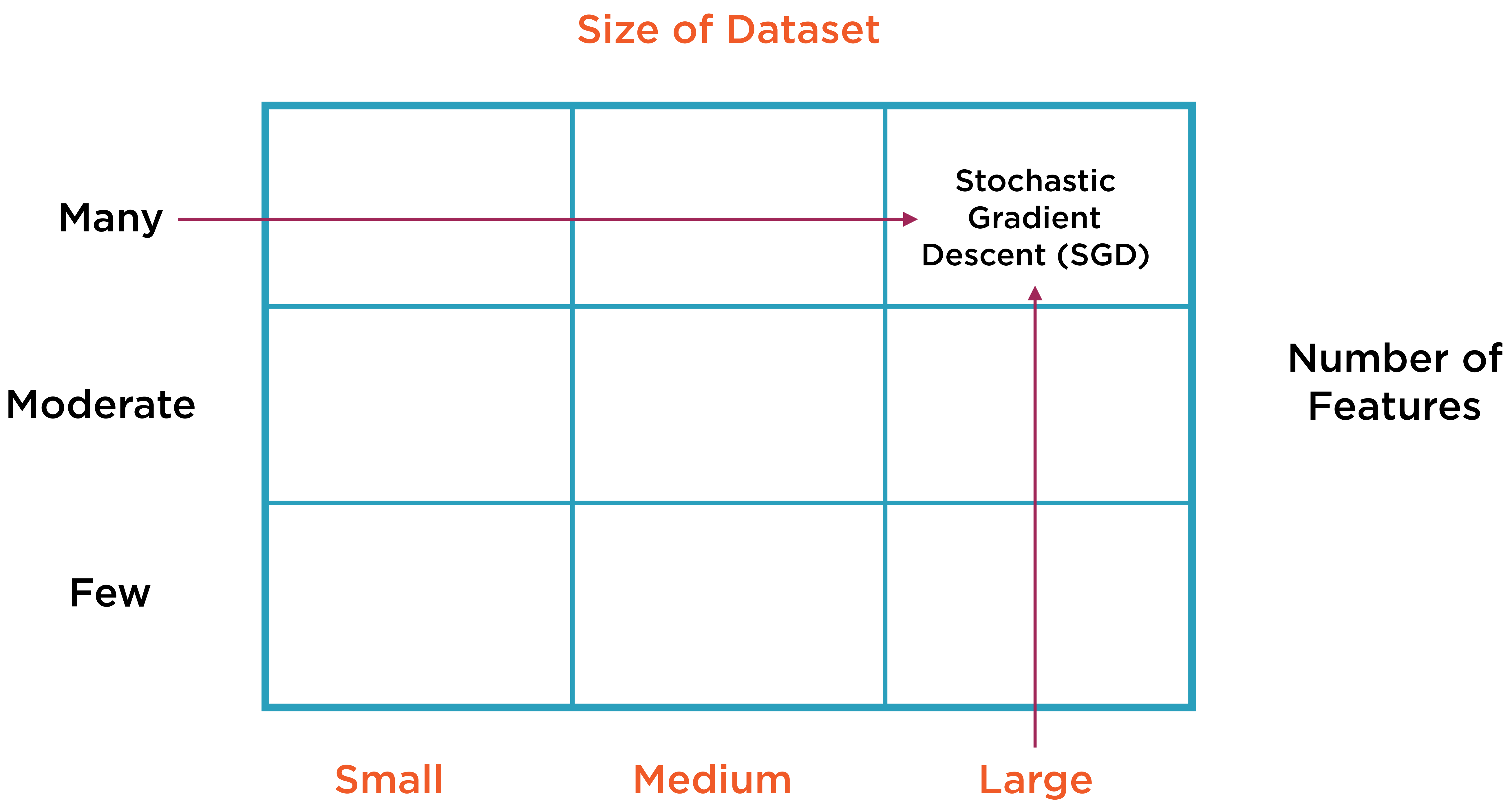
Choosing Regression Algorithms

Size of Dataset

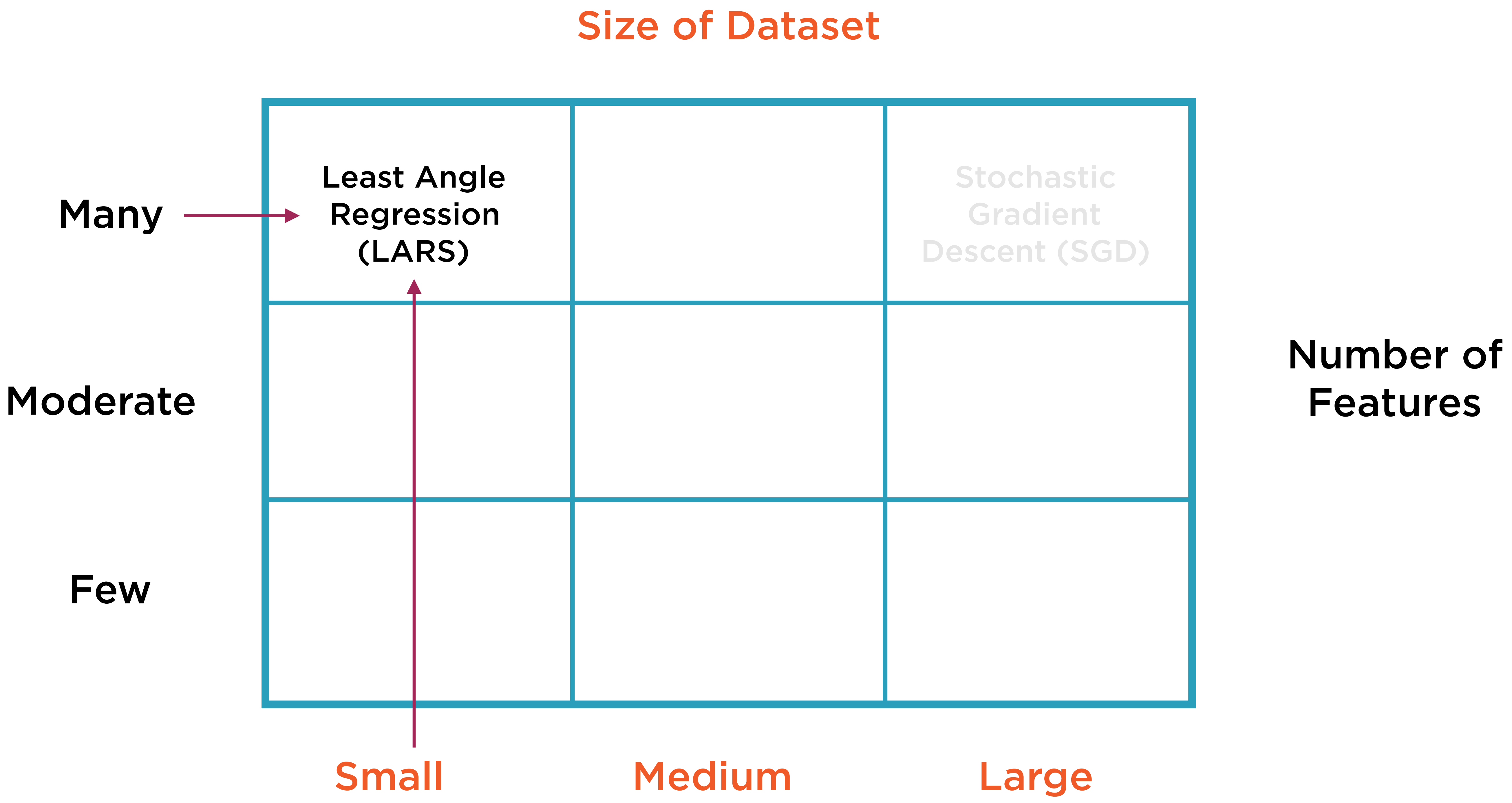
	Small	Medium	Large
Many			
Moderate			
Few			

Number of Features

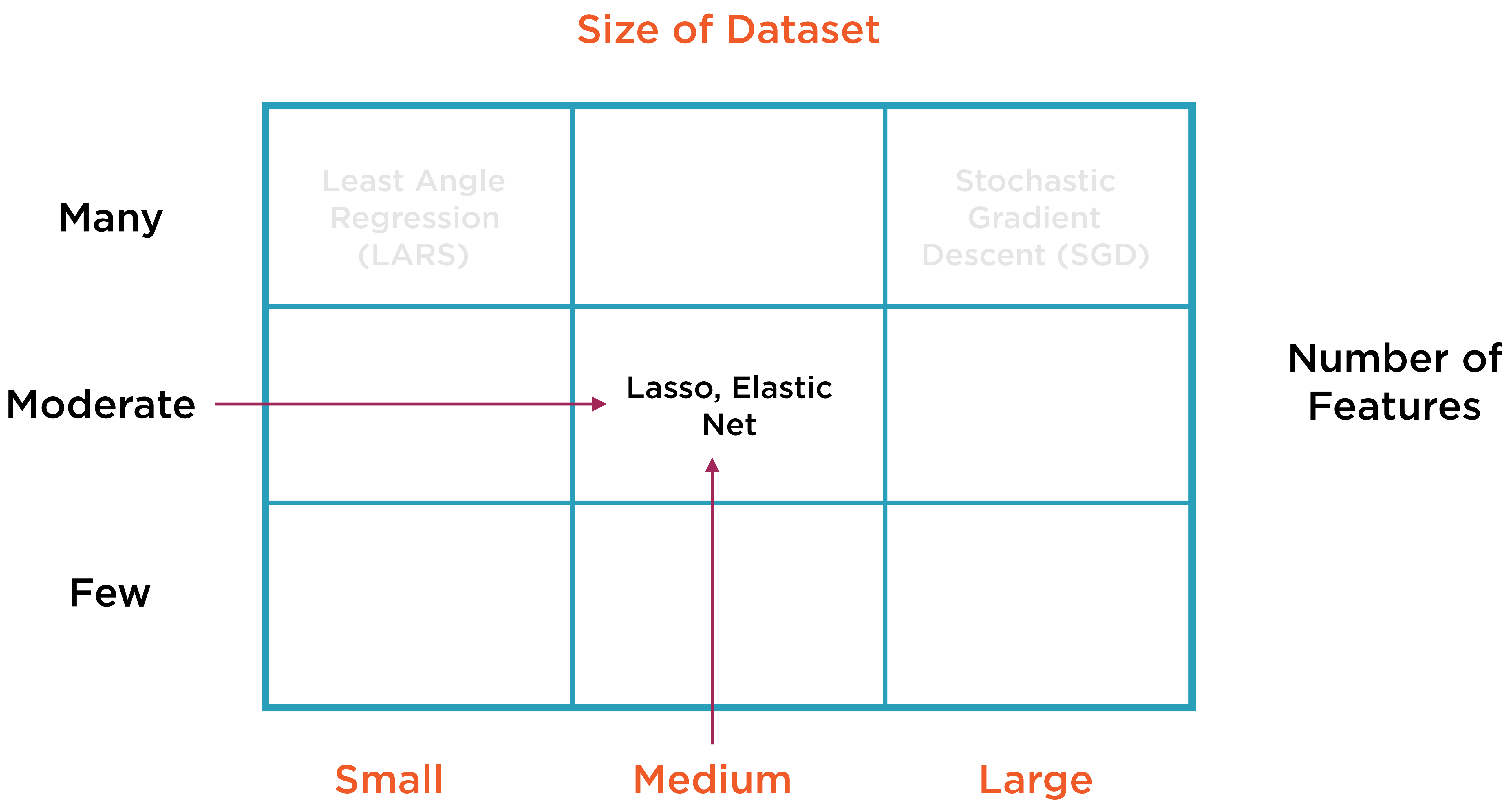
100K+ Data Points: Use SGD



More Features Than Samples: Use LARS



Many Features, Few Useful: Lasso, ElasticNet



Many Features, Most Useful: Ridge

Size of Dataset

Number of Features	Small	Medium	Large
Many	Least Angle Regression (LARS)	Ridge	Stochastic Gradient Descent (SGD)
Moderate		Lasso, Elastic Net	
Few			

Medium-sized Data with Non-linearity: SVR

Size of Dataset

Number of Features	Small	Medium	Large
Many	Least Angle Regression (LARS)	Ridge	Stochastic Gradient Descent (SGD)
Moderate	Support Vector Regression (Linear Kernel)	Lasso, Elastic Net	
Few			

Small Data with Non-linearity: SVR with RBF

Size of Dataset

Number of Features	Small	Medium	Large
Many	Least Angle Regression (LARS)	Ridge	Stochastic Gradient Descent (SGD)
Moderate	Support Vector Regression (Linear Kernel)	Lasso, Elastic Net	Support Vector Regression (Linear Kernel)
Few	**Support Vector Regression (RBF Kernel)**		

Many Features, Few Useful: Decision Trees

Size of Dataset

Number of Features	Small	Medium	Large
Many	Least Angle Regression (LARS)	Ridge	Stochastic Gradient Descent (SGD)
Moderate	Support Vector Regression (Linear Kernel)	Lasso, Elastic Net	Support Vector Regression (Linear Kernel)
Few	Support Vector Regression (RBF Kernel)	**Decision Trees and Ensemble Methods**	

Many Samples, Few Features: OLS

Size of Dataset

Number of Features	Small	Medium	Large
Many	Least Angle Regression (LARS)	Ridge	Stochastic Gradient Descent (SGD)
Moderate	Support Vector Regression (Linear Kernel)	Lasso, Elastic Net	Support Vector Regression (Linear Kernel)
Few	Support Vector Regression (RBF Kernel)	Decision Trees and Ensemble Methods	Ordinary Least Squares (OLS)

Choosing Regression Algorithms

Size of Dataset

Number of Features	Small	Medium	Large
Many	Least Angle Regression (LARS)	Ridge	Stochastic Gradient Descent (SGD)
Moderate	Support Vector Regression (Linear Kernel)	Lasso, Elastic Net	Support Vector Regression (Linear Kernel)
Few	Support Vector Regression (RBF Kernel)	Decision Trees and Ensemble Methods	Ordinary Least Squares (OLS)

1.5.3. Stochastic Gradient Descent for sparse data

Note: The sparse implementation produces slightly different results than the dense implementation due to a shrunk learning rate for the intercept.

There is built-in support for sparse data given in any matrix in a format supported by scipy.sparse. For maximum efficiency, however, use the CSR matrix format as defined in scipy.sparse.csr_matrix.

Examples:

- Classification of text documents using sparse features

1.5.4. Complexity

The major advantage of SGD is its efficiency, which is basically linear in the number of training examples. If X is a matrix of size (n, p) training has a cost of $O(kn\bar{p})$, where k is the number of iterations (epochs) and $\bar{p}$ is the average number of non-zero attributes per sample.

Recent theoretical results, however, show that the runtime to get some desired optimization accuracy does not increase as the training set size increases.

m = number of features n = size of training data

1.1.1.1. Ordinary Least Squares Complexity

The least squares solution is computed using the singular value decomposition of X. If X is a matrix of shape `(n_samples, n_features)` this method has a cost of $O(n_{\text{samples}} n_{\text{features}}^2)$, assuming that $n_{\text{samples}} \geq n_{\text{features}}$.

1.1.3. Lasso

T
s
p
ti

1.1.2.1. Ridge C

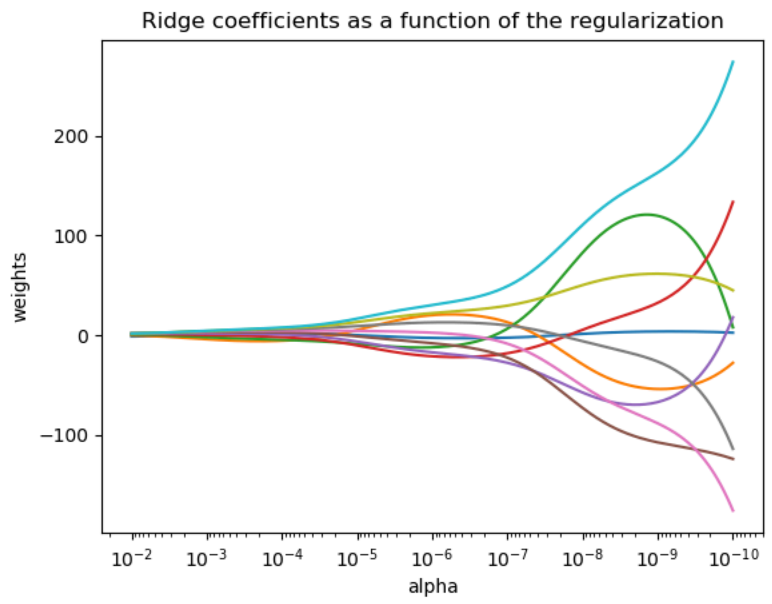
This method has the

prior (Lasso)).

`Ridge` regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2$$

The complexity parameter $\alpha \geq 0$ controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.



As with other linear models, `Ridge` will take in its `fit` method arrays X, y and will store the coefficients w of the linear

1.1.7. Least Angle Regression

Least-angle regression (LARS) is a regression algorithm for high-dimensional data, developed by Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani. LARS is similar to forward stepwise regression. At each step, it finds the feature most correlated with the target. When there are multiple features having equal correlation, instead of continuing along the same feature, it proceeds in a direction equiangular between the features.

The advantages of LARS are:

- It is numerically efficient in contexts where the number of features is significantly greater than the number of samples.
- It is computationally just as fast as forward selection and has the same order of complexity as ordinary least squares.
- It produces a full piecewise linear solution path, which is useful in cross-validation or similar attempts to tune the model.
- If two features are almost equally correlated with the target, then their coefficients should increase at approximately the same rate. The algorithm thus behaves as intuition would expect, and also is more stable.
- It is easily modified to produce solutions for other estimators, like the Lasso.

The disadvantages of the LARS method include:

- Because LARS is based upon an iterative refitting of the residuals, it would appear to be especially sensitive to the effects of noise. This problem is discussed in detail by Weisberg in the discussion section of the Efron et al. (2004) Annals of Statistics article.

The LARS model can be used using estimator `Lars` , or its low-level implementation `lars_path` or `lars_path_gram` .

1.1.3. Lasso

The `Lasso` is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent. For this reason Lasso and its variants are fundamental to the field of compressed sensing. Under certain conditions, it can recover the exact set of non-zero coefficients (see Compressive sensing: tomography reconstruction with L1 prior (

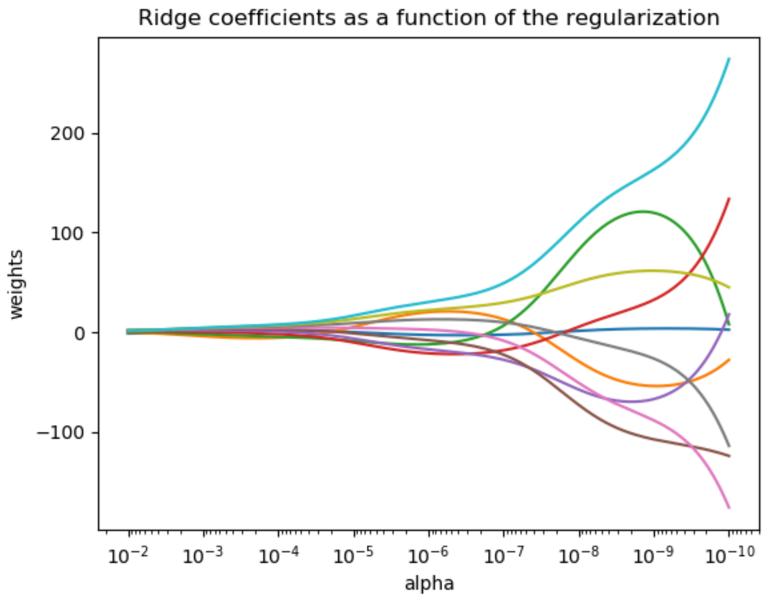
1.1.2.1. Ridge Complexity

This method has the same order of complexity as Ordinary Least Squares.

`Ridge` regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2$$

The complexity parameter $\alpha \geq 0$ controls the amount of shrinkage: the larger the value of α, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.



As with other linear models, `Ridge` will take in its `fit` method arrays X, y and will store the coefficients w of the linear model in its `coef_` member:

m = number of features n = size of training data

The `SVC` class is based on the *libsvm* library, which implements an algorithm that supports the kernel trick.[2] The training time complexity is usually between $O(m^2 \times n)$ and $O(m^3 \times n)$. Unfortunately, this means that it gets dreadfully slow when the number of training instances gets large (e.g., hundreds of thousands of instances). This algorithm is perfect for complex but small or medium training sets. However, it scales well with the number of features, especially with *sparse features* (i.e., when each instance has few nonzero features). In this case, the algorithm scales roughly with the average number of nonzero features per instance. Table 5-1 compares Scikit-Learn's SVM classification classes.

Table 5-1. Comparison of Scikit-Learn classes for SVM classification

Class	**Time complexity**	**Out-of-core support**	**Scaling required**	**Kernel trick**
`LinearSVC`	$O(m \times n)$	No	Yes	No
`SGDClassifier`	$O(m \times n)$	Yes	Yes	No
`SVC`	$O(m^2 \times n)$ to $O(m^3 \times n)$	No	Yes	Yes

m = number of features n = size of training data

Computational Complexity

Making predictions requires traversing the Decision Tree from the root to a leaf. Decision Trees are generally approximately balanced, so traversing the Decision Tree requires going through roughly $O(log_2(m))$ nodes.[3] Since each node only requires checking the value of one feature, the overall prediction complexity is just $O(log_2(m))$, independent of the number of features. So predictions are very fast, even when dealing with large training sets.

However, the training algorithm compares all features (or less if `max_features` is set) on all samples at each node. This results in a training complexity of $O(n \times m\ log(m))$. For small training sets (less than a few thousand instances), Scikit-Learn can speed up training by presorting the data (set `presort=True`), but this slows down training considerably for larger training sets.

Also, more features => Risk of overfitting with Decision trees

Can mitigate with Ensemble, but again slows down (more trees needed)

Evaluating Regression Models

Interpreting Results of Regression

Adjusted R^2

Residuals

F-statistic

T-statistics

R^2

Interpreting Results of Regression

Adjusted R^2

Residuals

F-statistic

T-statistics

R^2

Interpreting Results of a Simple Regression

R^2

Measures overall quality of fit - the higher the better (up to a point)

Residuals

Check if regression assumptions are violated

R^2

How well does the line represent the data?

How much of the variance in the data is captured by the line?

R^2

$$R^2 = \frac{\text{Explained variance}}{\text{Total variance}}$$

R^2

A higher R-square value indicates that a lot of the underlying variance is captured

Better-fit line

Interpreting Results of Regression

Adjusted R^2

Residuals

F-statistic

T-statistics

R^2

Adjusted-R^2 = R^2 x (Penalty for adding irrelevant variables)

Adjusted-R^2

Increases if irrelevant* variables are deleted

(*irrelevant variables = any group whose F-ratio < 1)

Adjusted-R^2 = R^2 x (Penalty for adding irrelevant variables)

Adjusted-R^2

Increases if irrelevant* variables are deleted

(*irrelevant variables = any group whose F-ratio < 1)

Interpreting Results of Regression

Adjusted R^2

Residuals

F-statistic

T-statistics

R^2

Regression T-statistics vs. F-statistic

T-statistics	F-statistic
One t-statistic for each regression coefficient	One F-statistic for the regression model as a whole
Null hypothesis: Corresponding coefficient value is zero	Null hypothesis: All coefficient values are equal to zero
Used to evaluate utility of specific variable in model	Used to evaluate overall quality of model
Widely used; standard error = coefficient/t-statistic	Relatively rarely used; R-squared or Adjusted R-squared preferred

Types of Classification

Types of Classification Tasks

Binary

“Yes/No”, “True/False”, “Up/Down”

Output is binary categorical variable

Multi-class

Digit classification

Output variable takes 1 of N (>2) values

Multi-label

(“True”, “Female”), (“False”, “Female”)

Output is tuple of multiple binary variables (not disjoint)

Multi-output

(“Sunday”, “January”)

Multiclass + multilabel

Multi-class Classification

Many classification algorithms are inherently binary

- Logistic regression
- Support Vector Machines

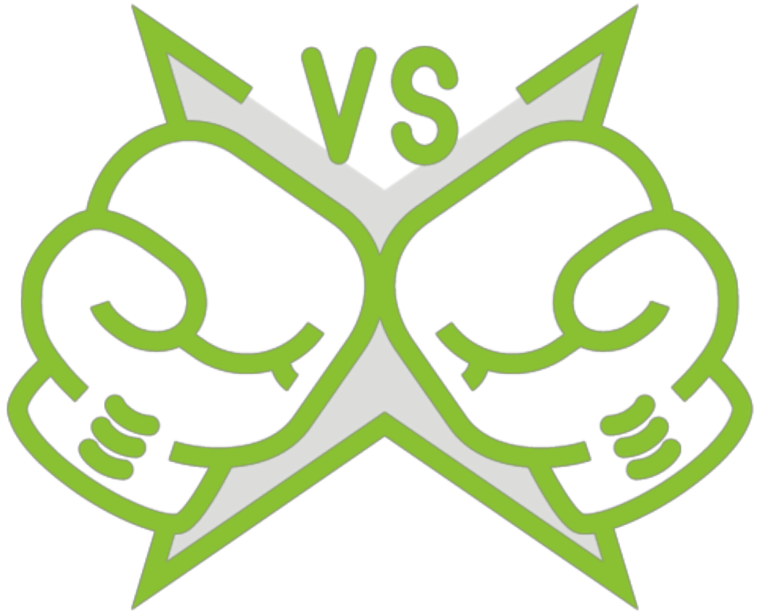
Inherently binary classifiers can be generalized for multi-class classification

Multi-class Classification

Some other algorithms are inherently multi-class

- Naive Bayes

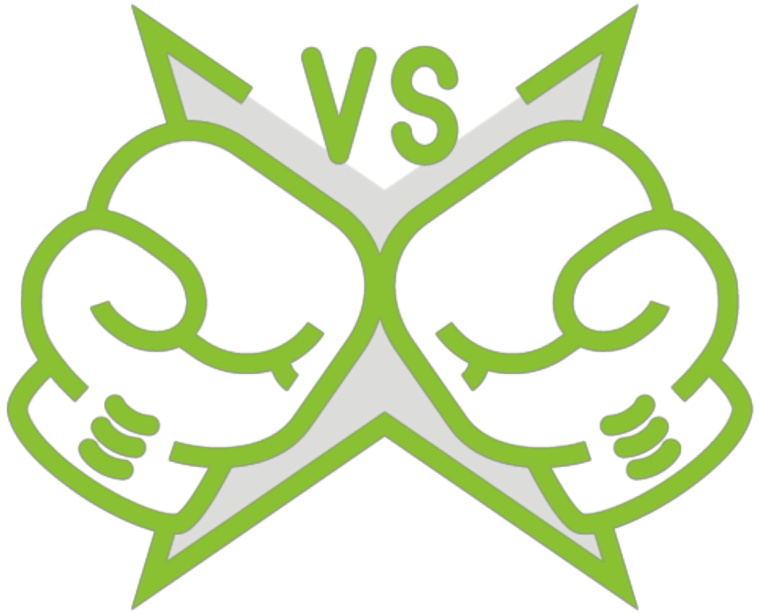
Multi-class Digit Classification



One-versus-all: Train 10 binary classifiers

- 0 or not 0
- 1 or not 1
- 2 or not 2
- Predicted label = output of detector with highest score

Multi-class Digit Classification



One-versus-one: Train 45 binary classifiers

One detector for each pair of digits

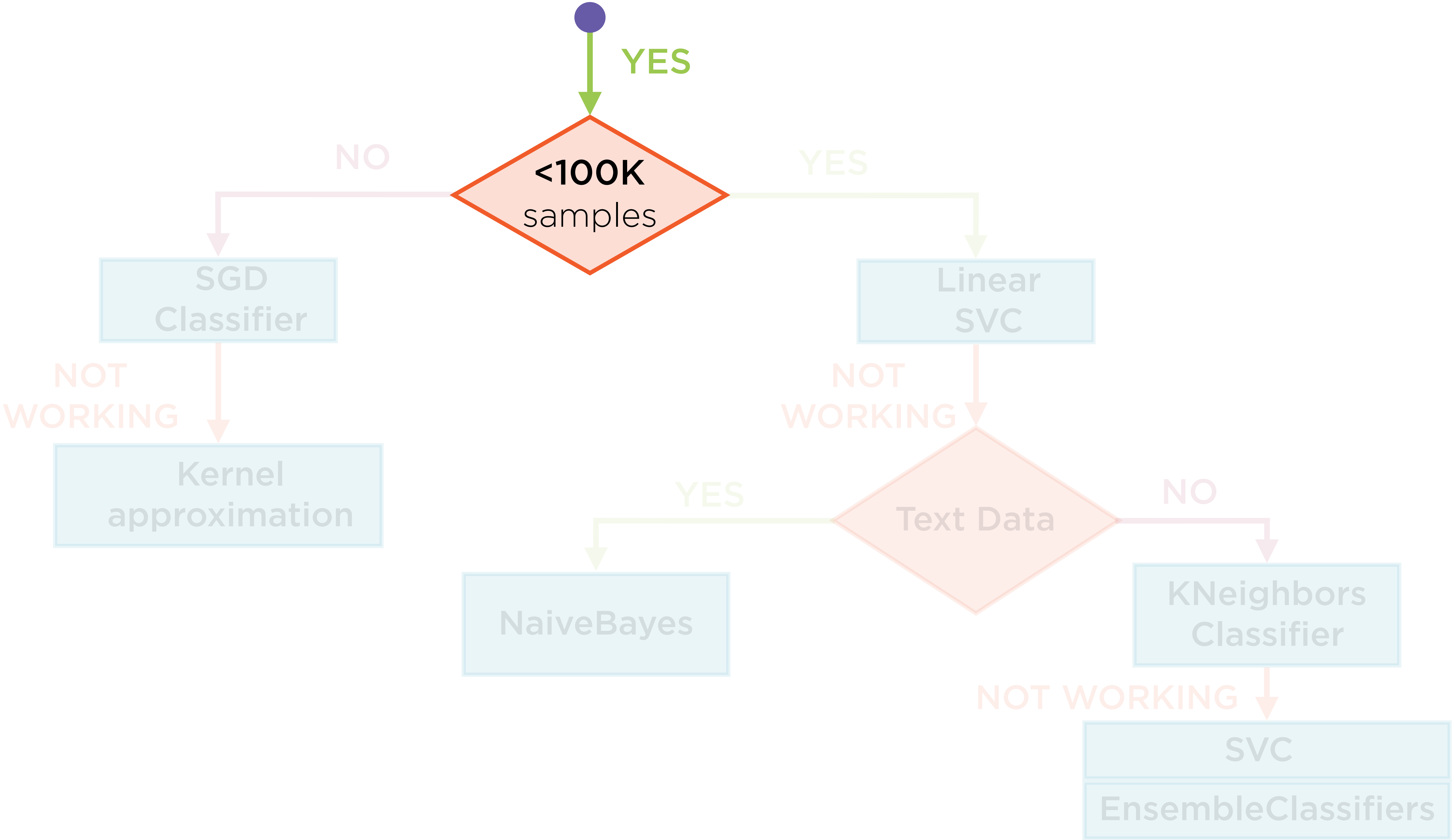
- 0 vs 1, 0 vs 2, 0 vs 3 and so on
- 1 vs 2, 1 vs 3 and so on

For N labels, need N(N-1)/2 classifiers

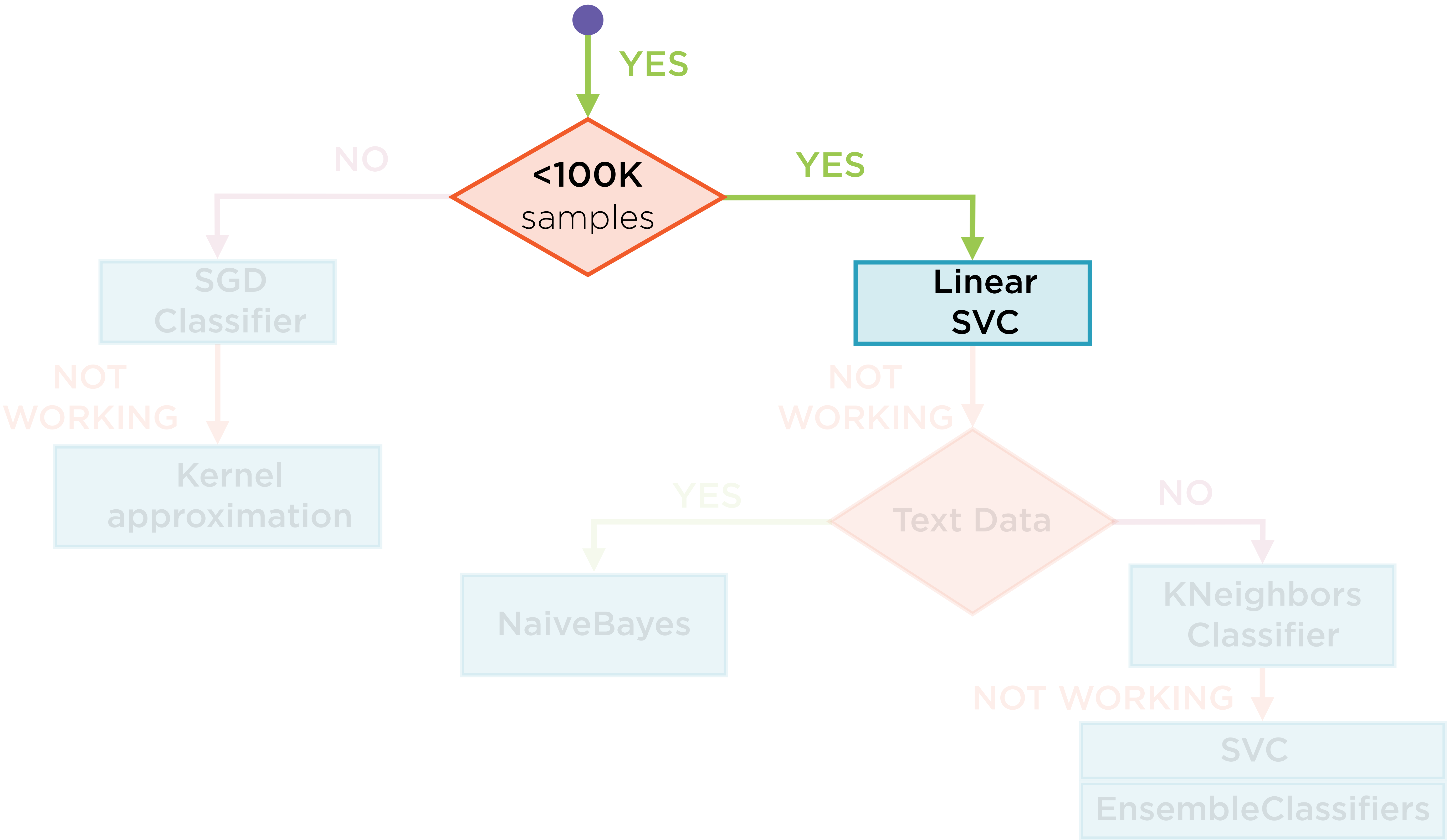
- Predicted label = output of digit that wins most duels

Classification Algorithms

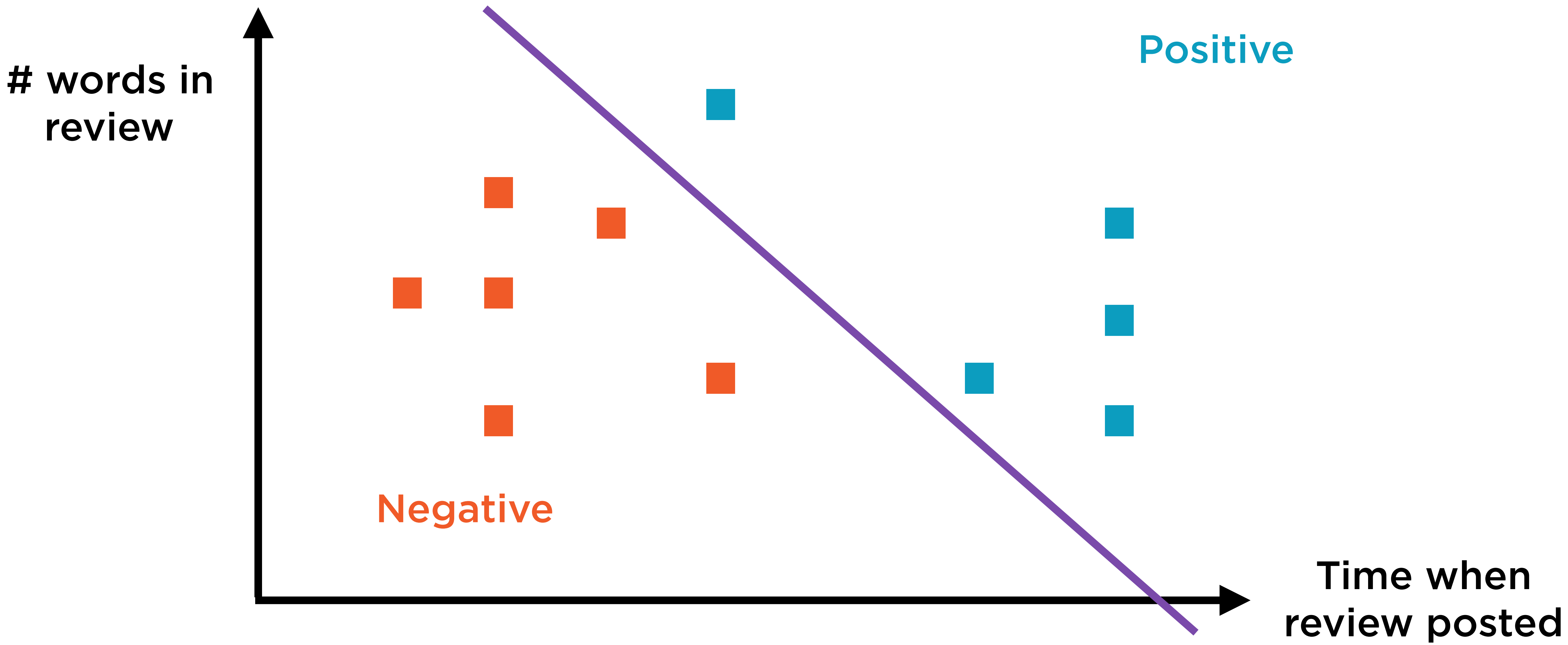
Classification



Classification

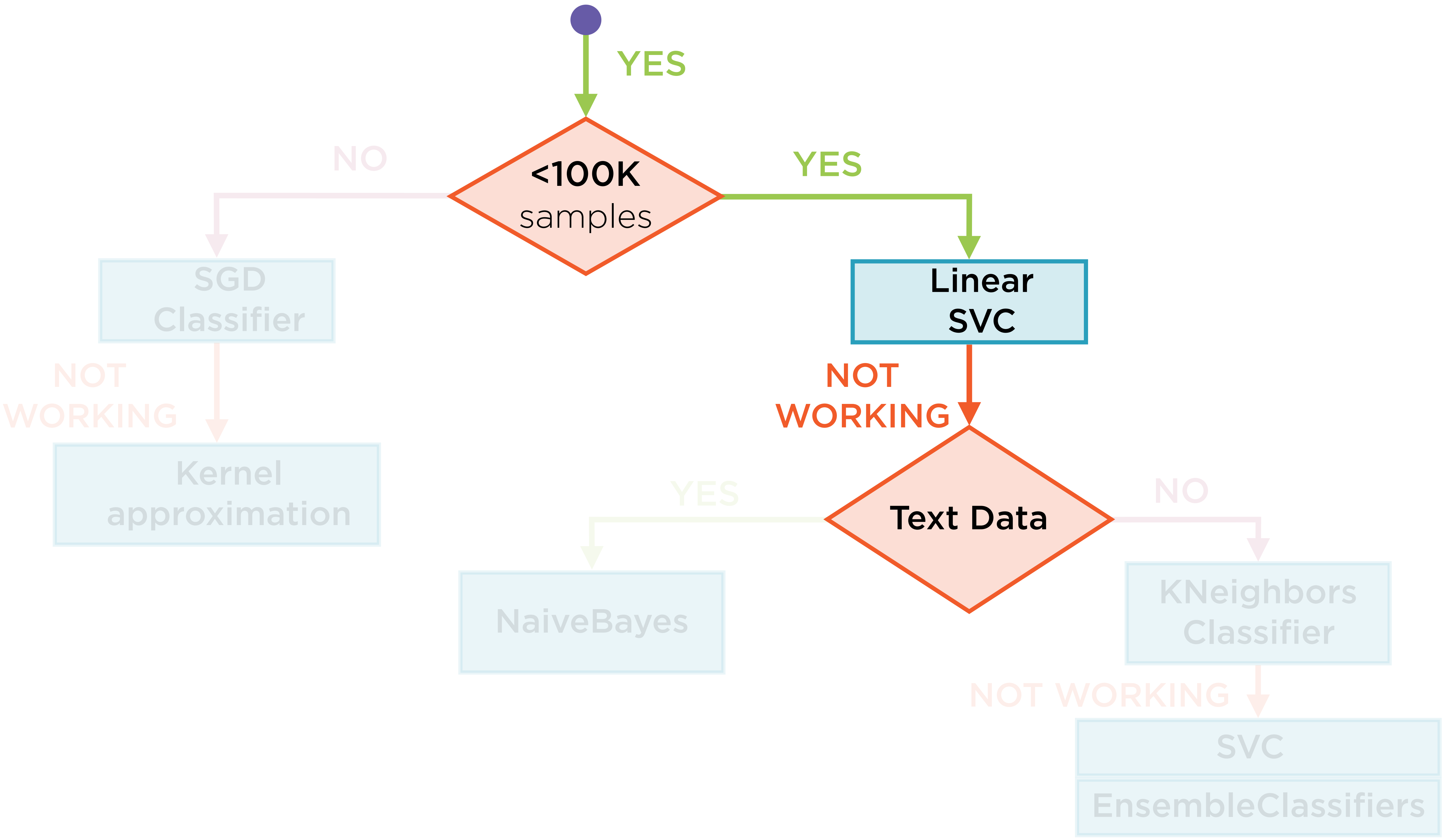


Linearly Separable Data

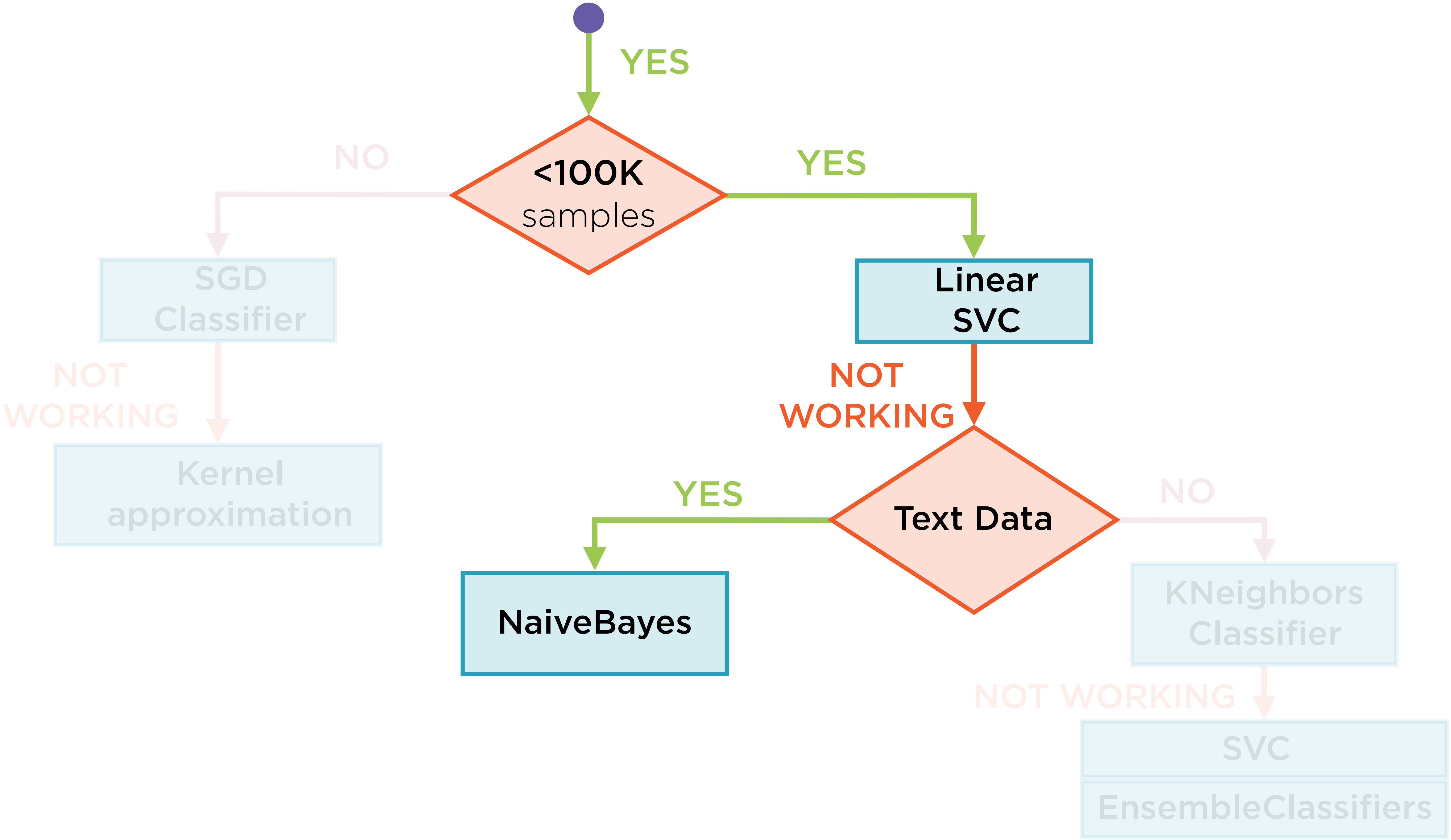


A linear SVC fits a straight line to separate classes in the data

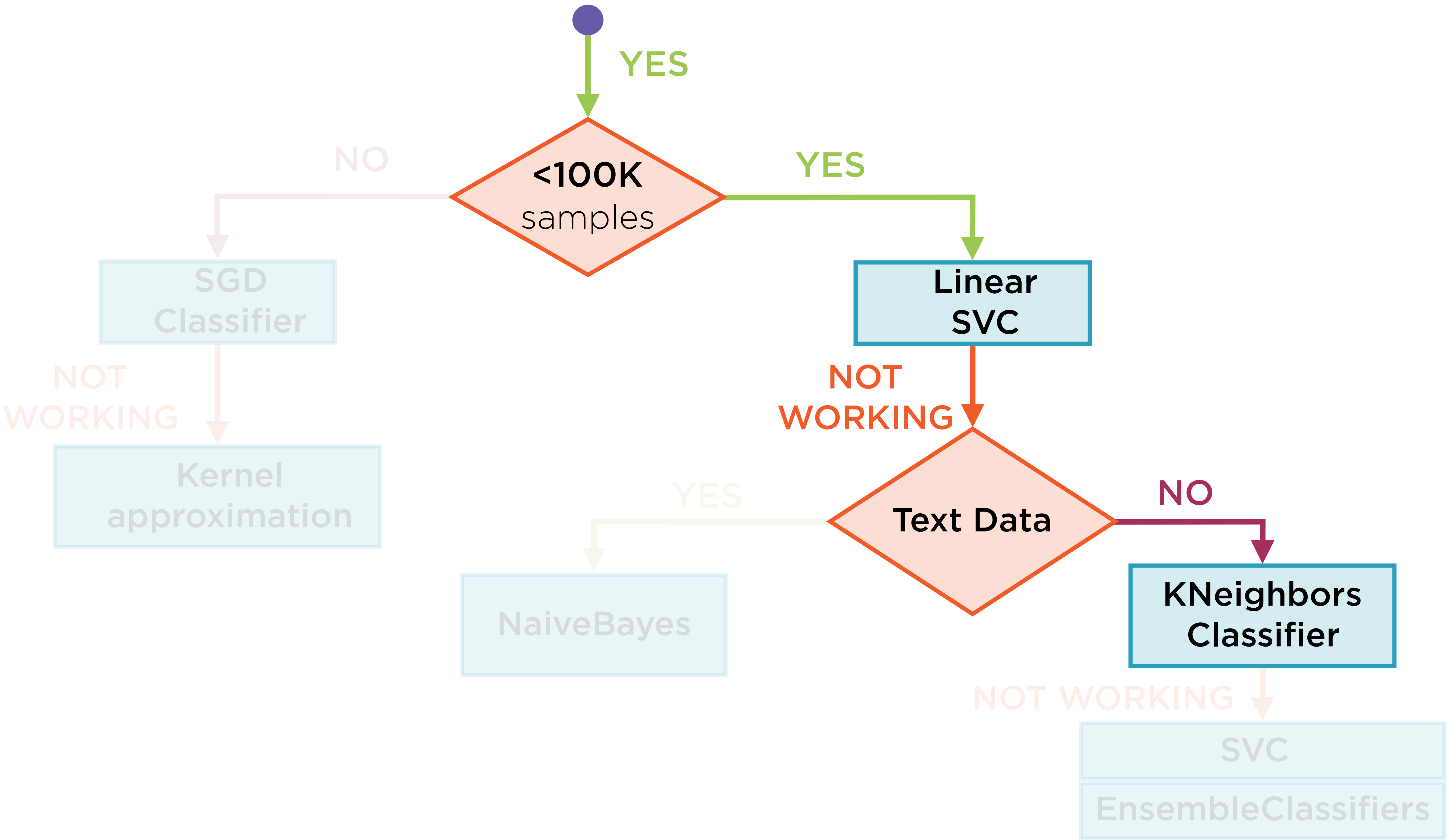
Classification



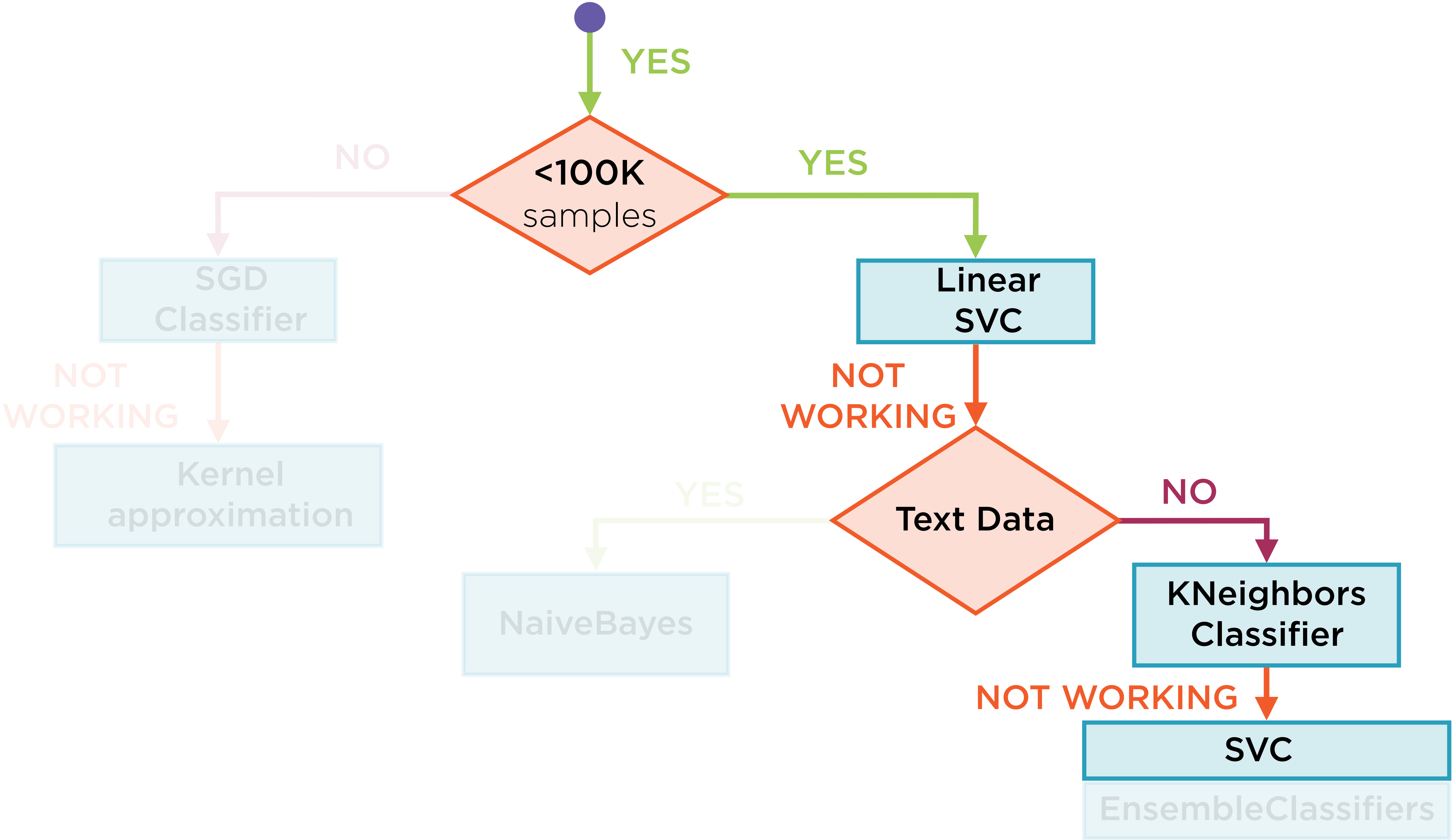
Classification



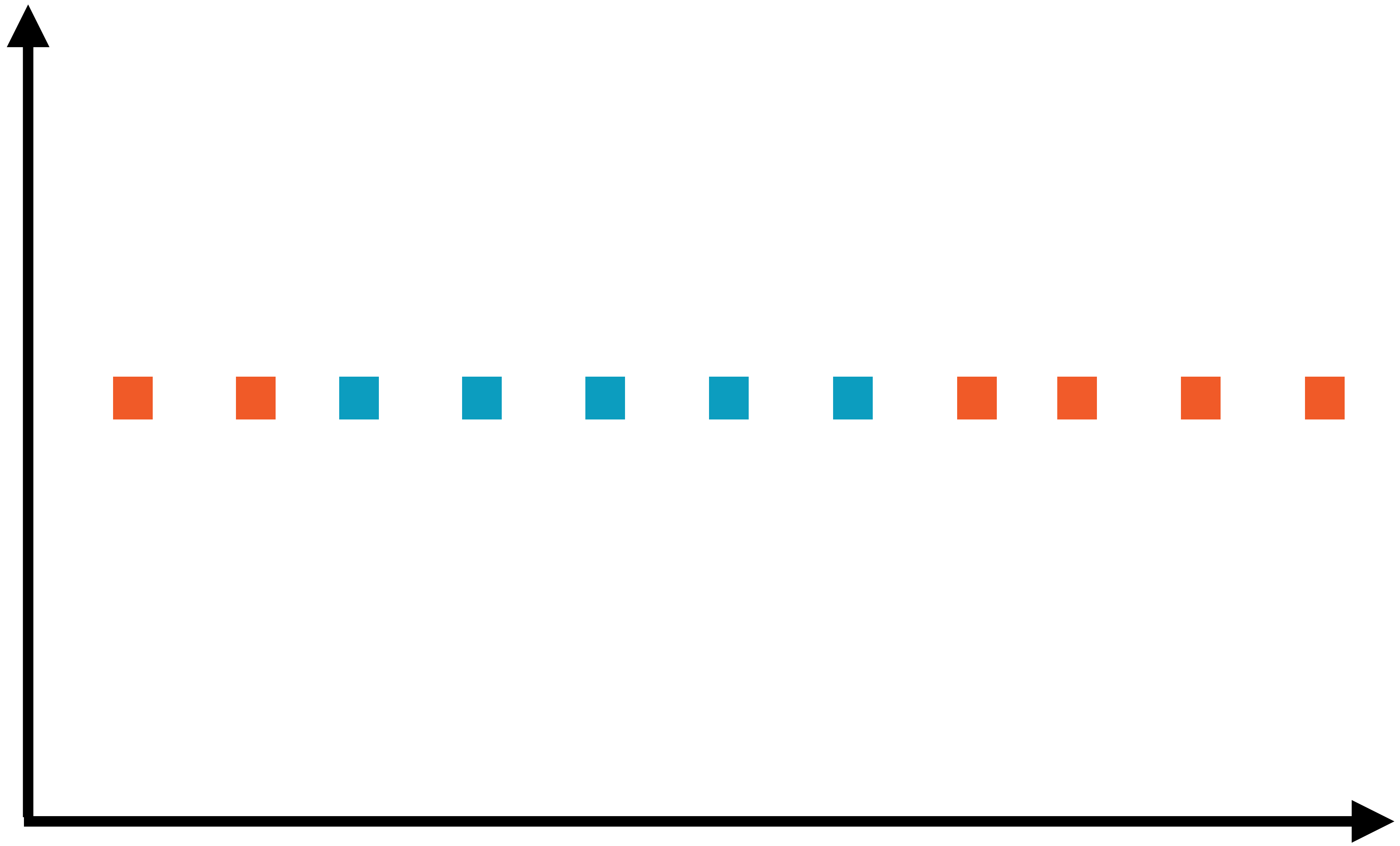
Classification



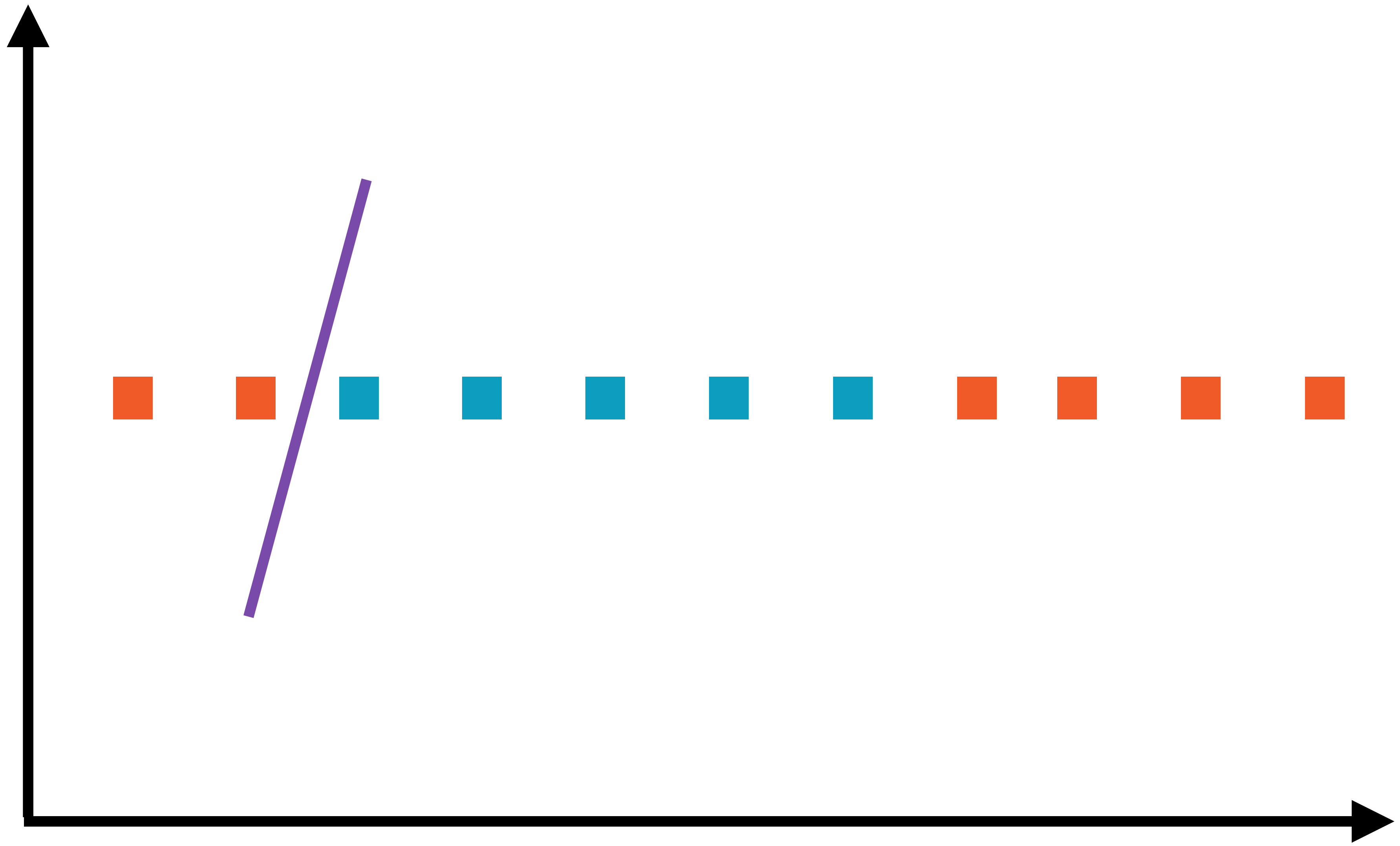
Classification



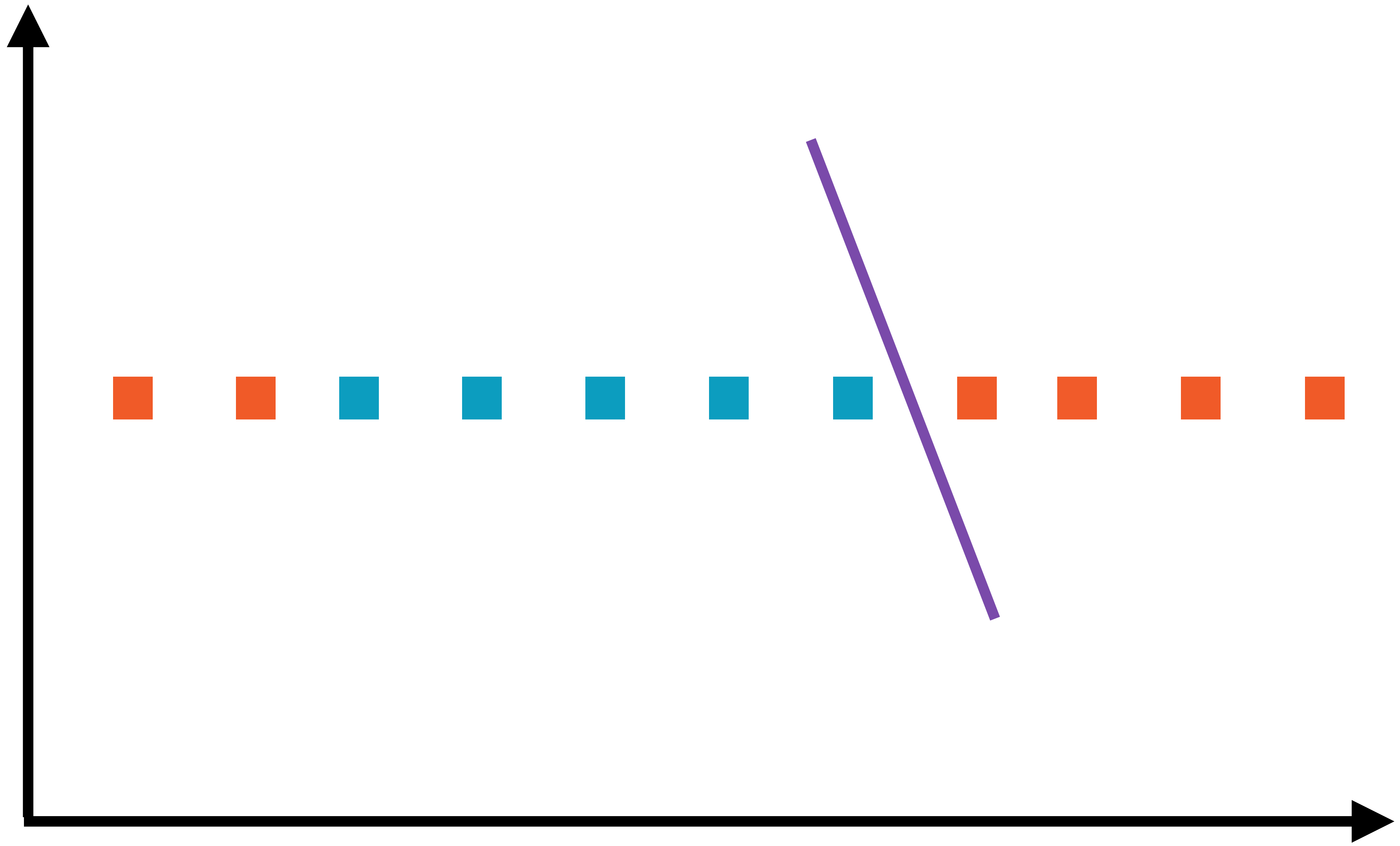
Non-separable Data



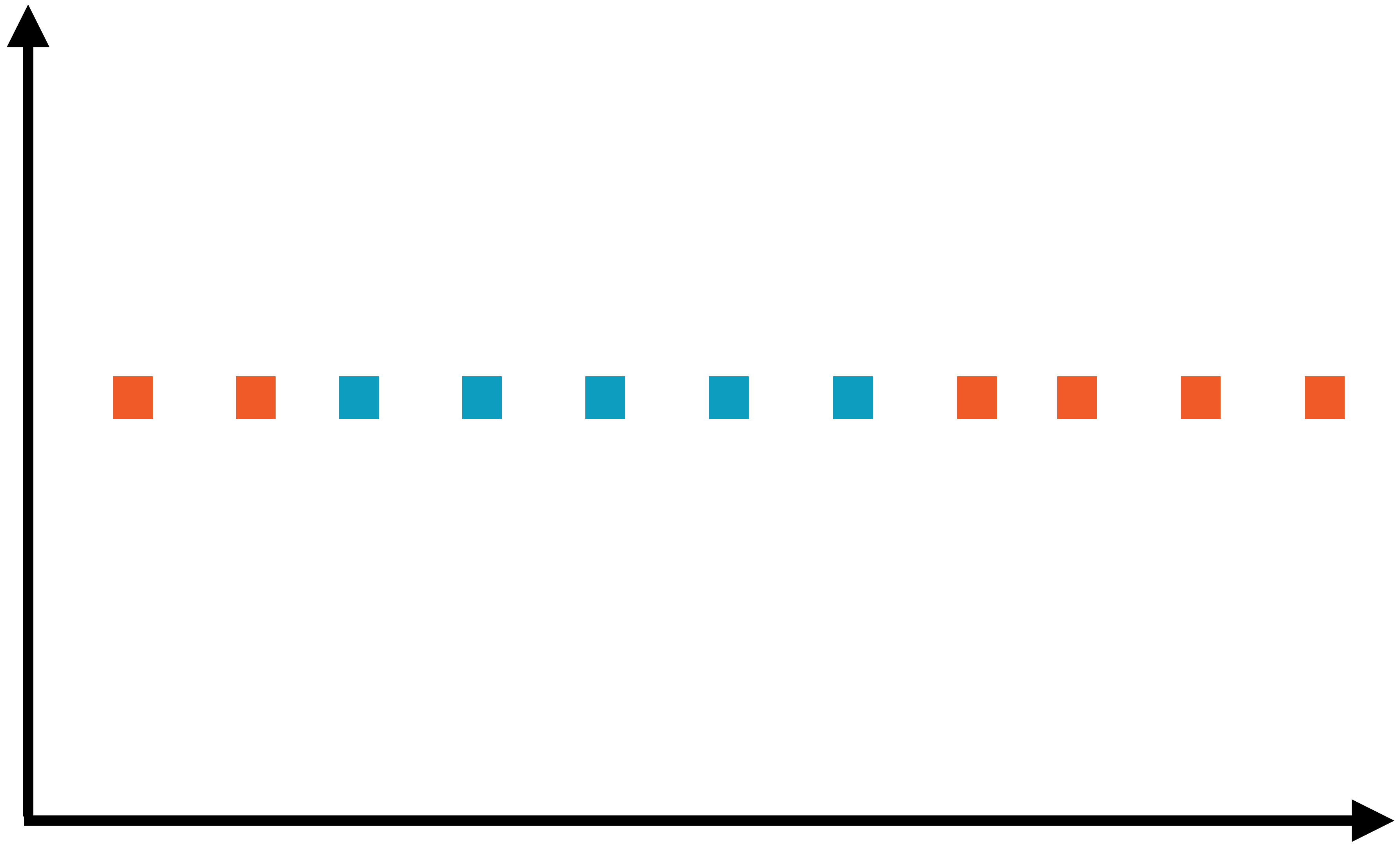
Non-separable Data



Non-separable Data

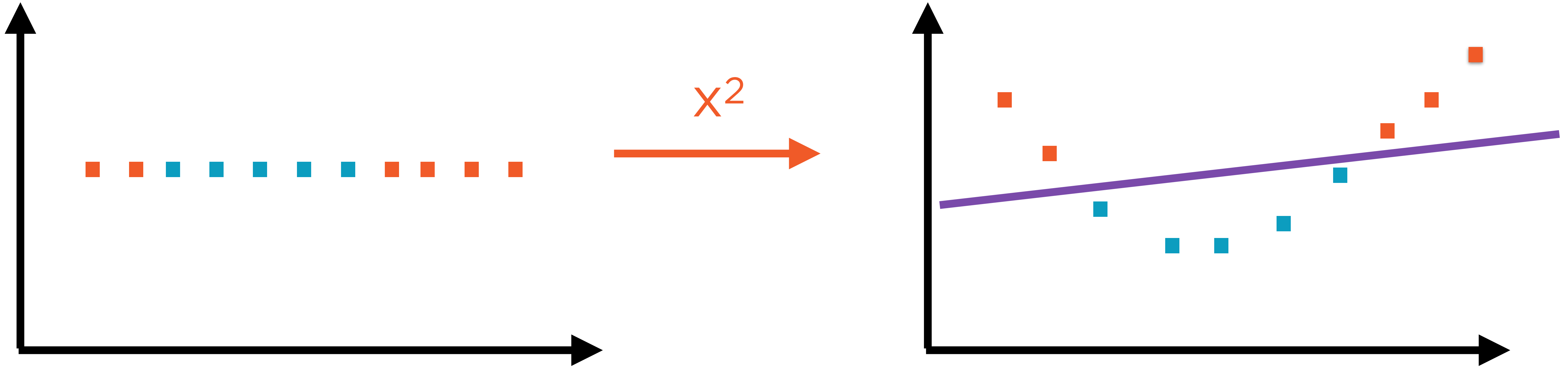


Non-separable Data



Transform data using the kernel trick such that it is separable

Nonlinear SVM



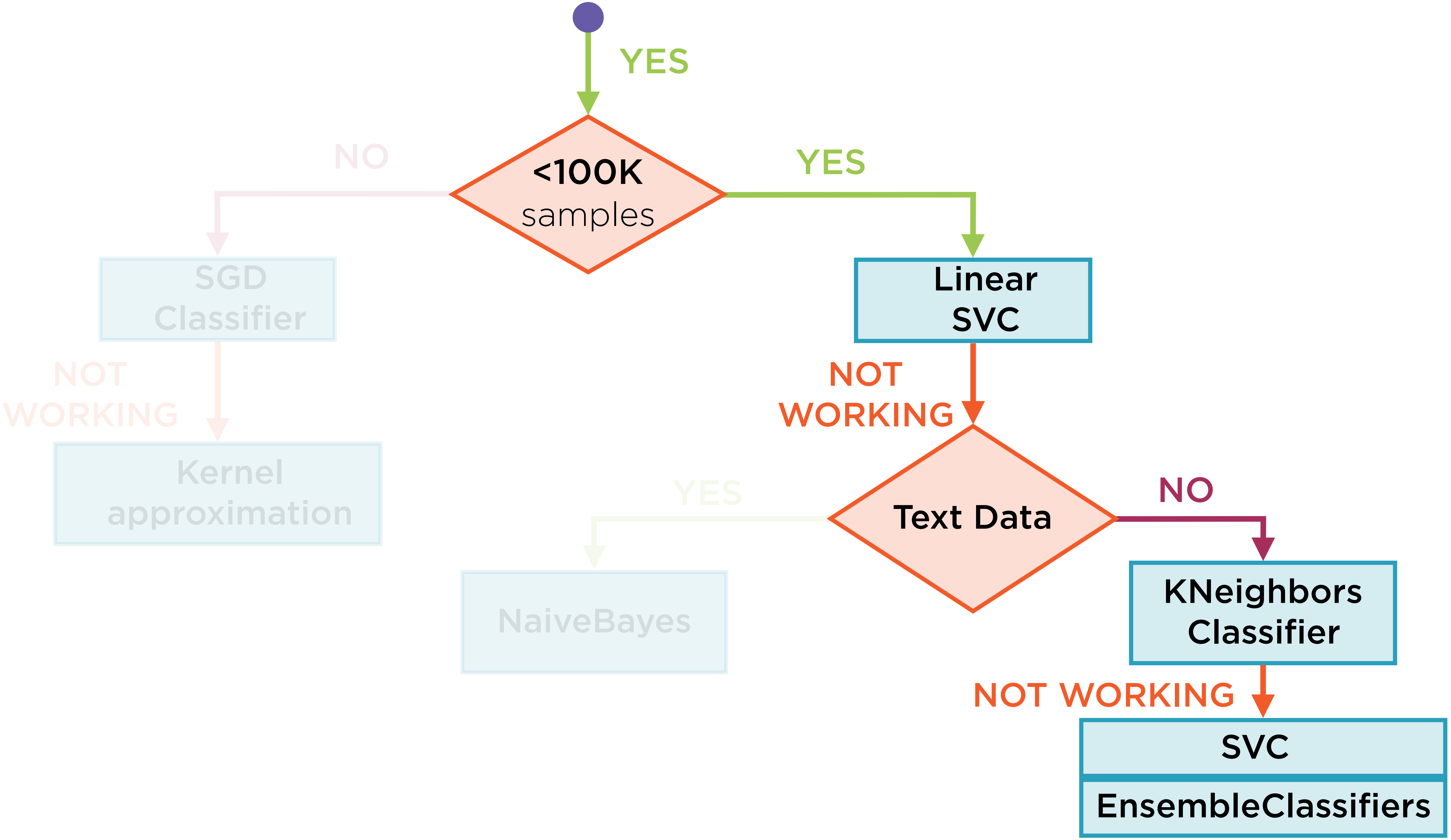
Original Data

Not linearly separable

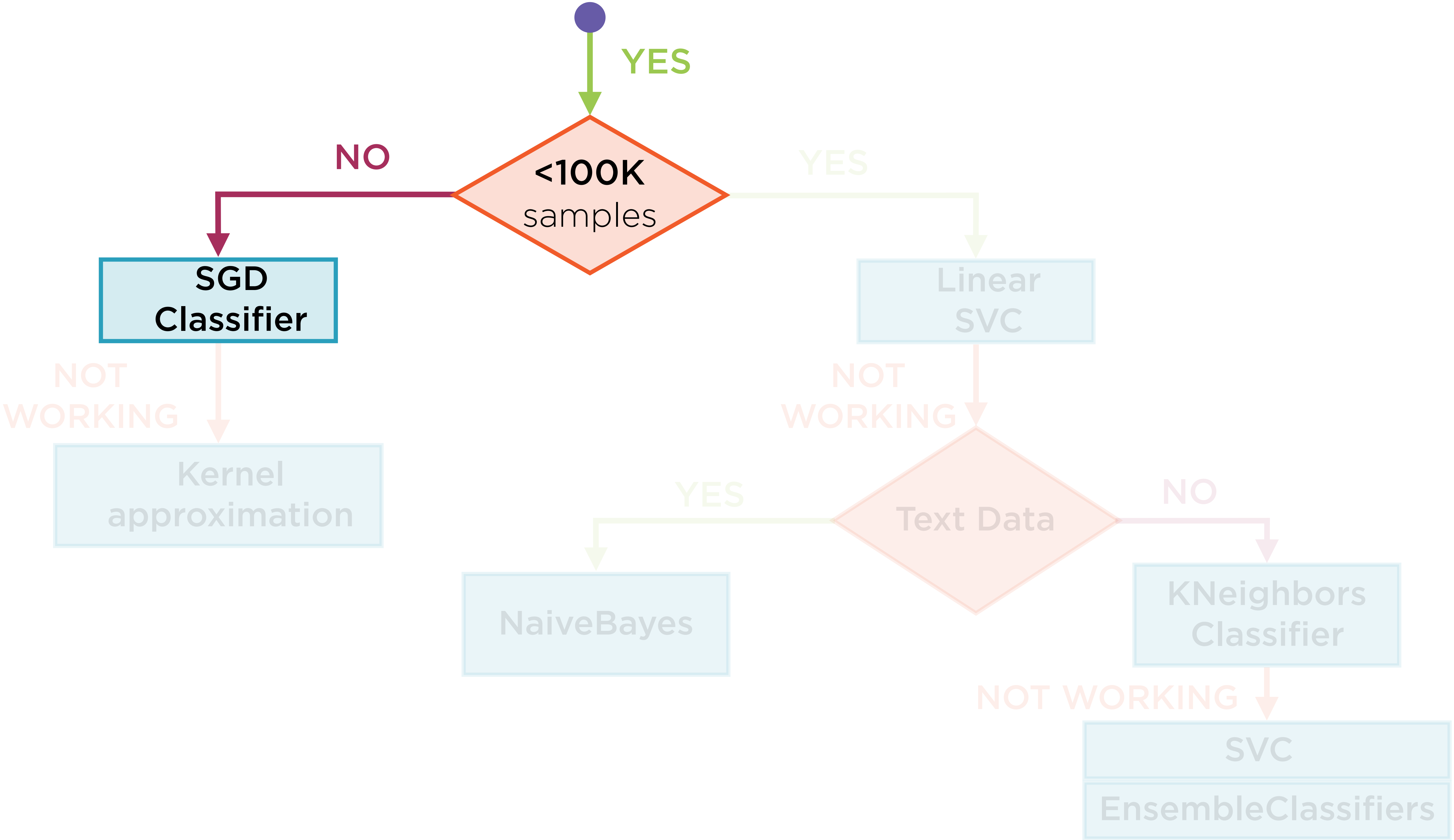
Square of original data

Now linearly separable!

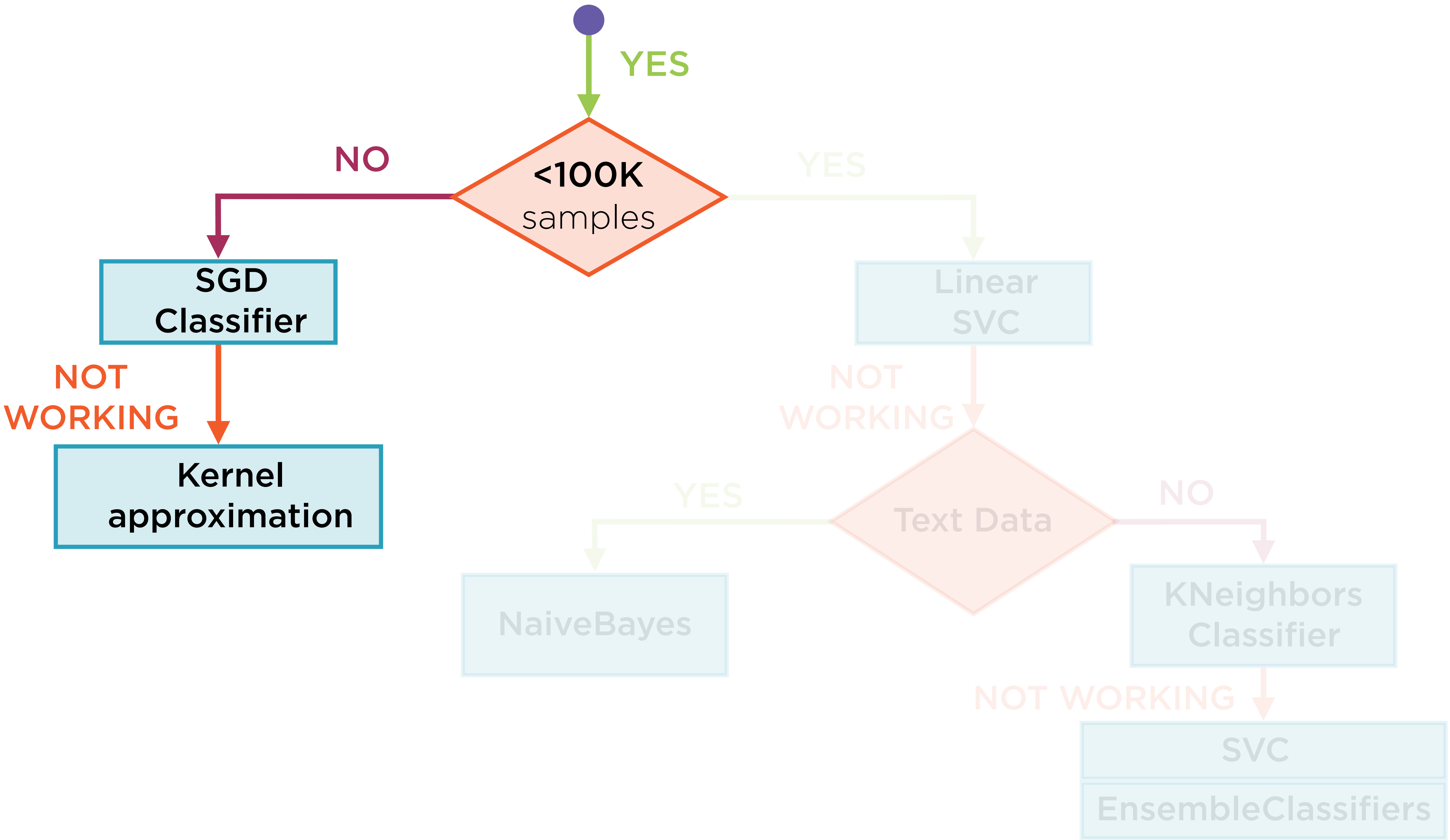
Classification



Classification



Classification



Evaluating Classifiers

All-is-well Binary Classifier

Here, accuracy for rare cancer may be 99.9999%, but…

Accuracy

Some labels maybe much more common/rare than others

Such a dataset is said to be skewed

Accuracy is a poor evaluation metric here

Confusion Matrix

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 instances	4 instances
No Cancer	5 instances	1000 instances

Confusion Matrix

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	**10**	**4**
No Cancer	**5**	**1000**

True Positive

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 TP	4
No Cancer	5	1000

Actual Label = Predicted Label

False Positive

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10	4
No Cancer	5 FP	1000

Actual Label ≠ Predicted Label

True Negative

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10	4
No Cancer	5	1000 TN

Actual Label = Predicted Label

False Negative

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10	4 FN
No Cancer	5	1000

Actual Label ≠ Predicted Label

Confusion Matrix

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

Accuracy

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

Actual Label = Predicted Label

Accuracy

	Predicted Labels	
Actual Label	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

$$\text{Accuracy} = \frac{\text{TP + TN}}{\text{Num Instances}} = \frac{1010}{1019} = 99.12\%$$

Accuracy is not a good metric to evaluate whether this model performs well

Precision

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

Precision

Actual Label	Predicted Labels: Cancer	Predicted Labels: No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

Precision = Accuracy when classifier flags cancer

Precision

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{10}{15} = 66.67\%$$

Recall

Predicted Labels

Actual Label

	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

Recall = Accuracy when cancer actually present

Recall

Actual Label \ Predicted Labels	Cancer	No Cancer
Cancer	10 TP	4 FN
No Cancer	5 FP	1000 TN

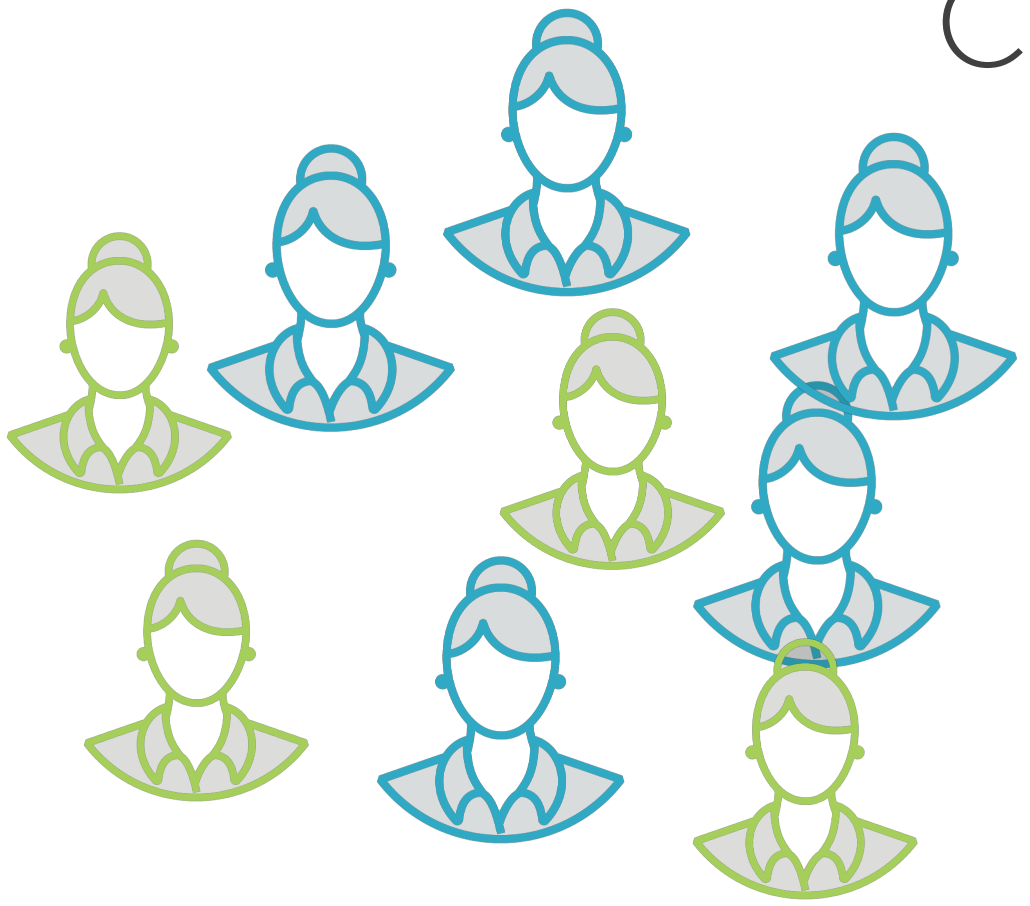
$$\text{Recall} = \frac{TP}{TP + FN} = \frac{10}{14} = 71.42\%$$

Clustering Algorithms

Clustering

A set of points, each representing a user

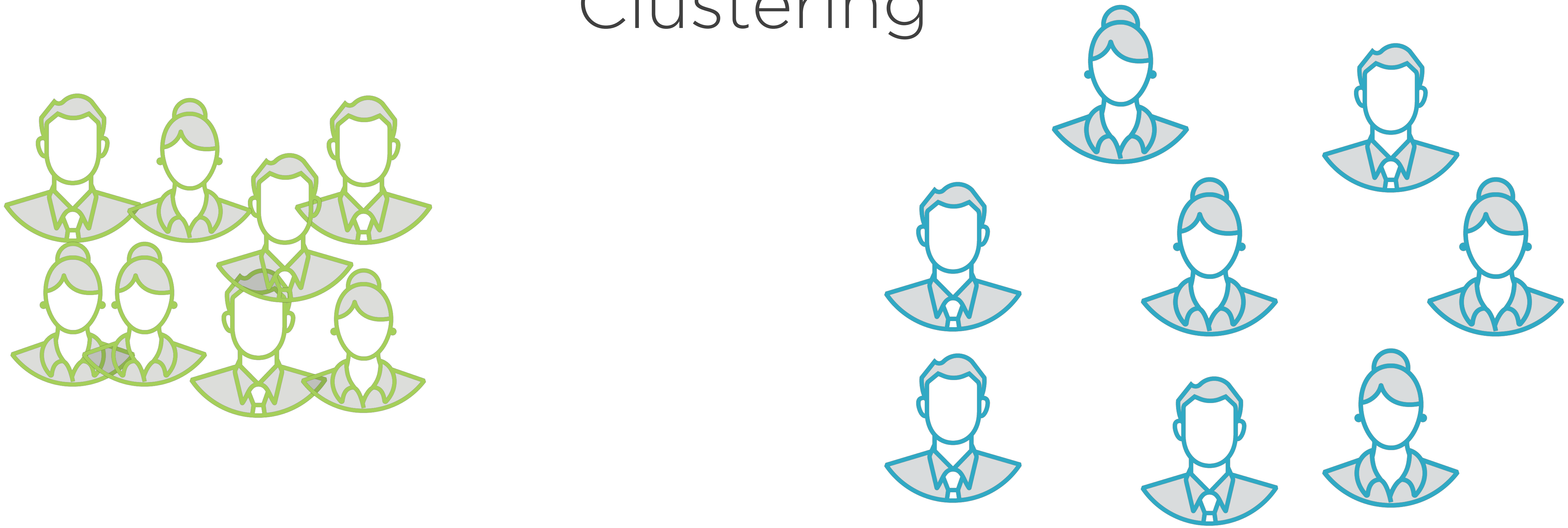
Clustering



Same group = similar

Different group = different

Clustering



Same group = similar

Different group = different

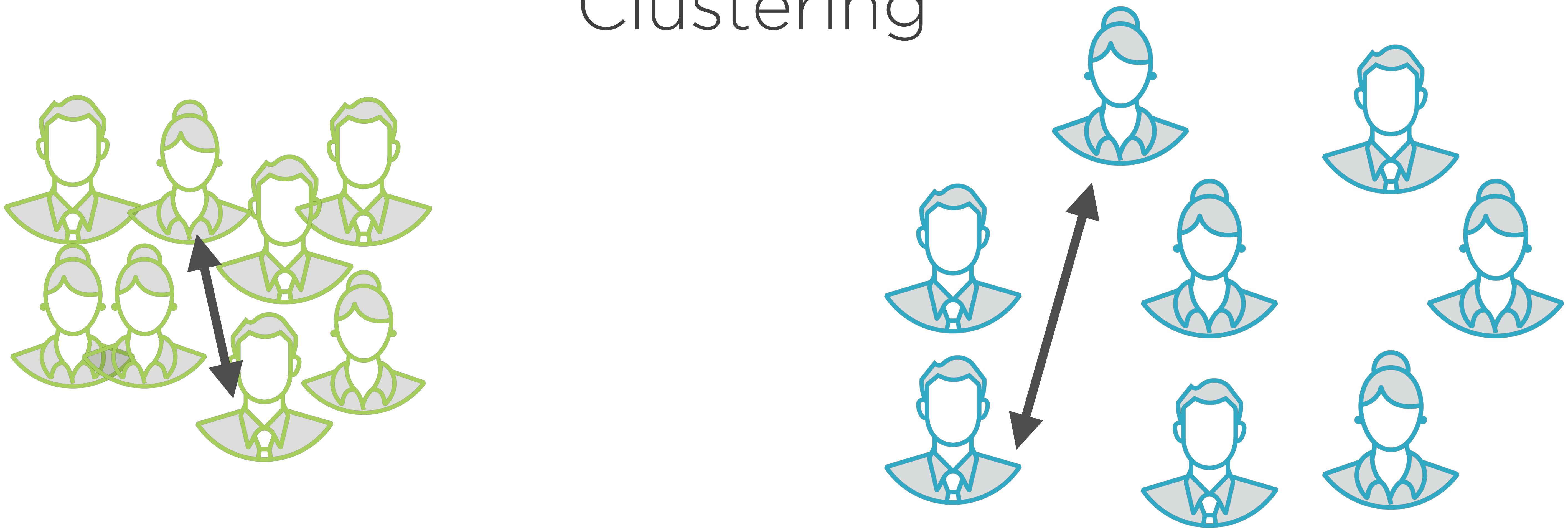
Users in a Cluster

May like the same kind of music

May have gone to the same high school

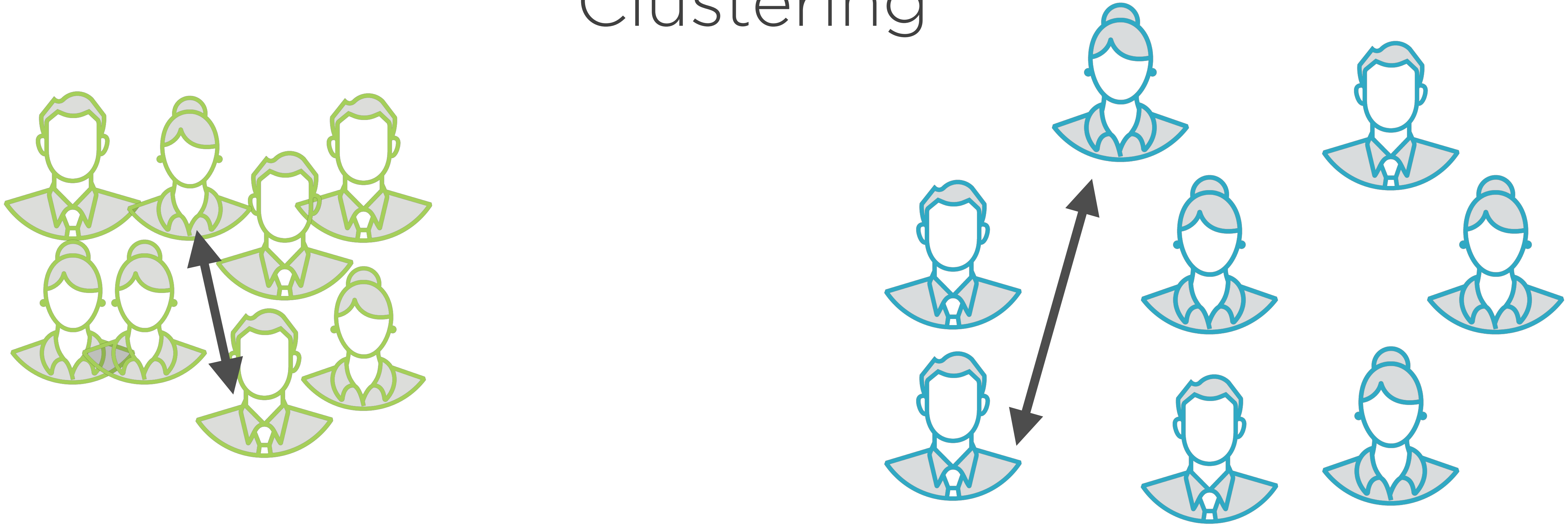
May have kids of the same age

Clustering



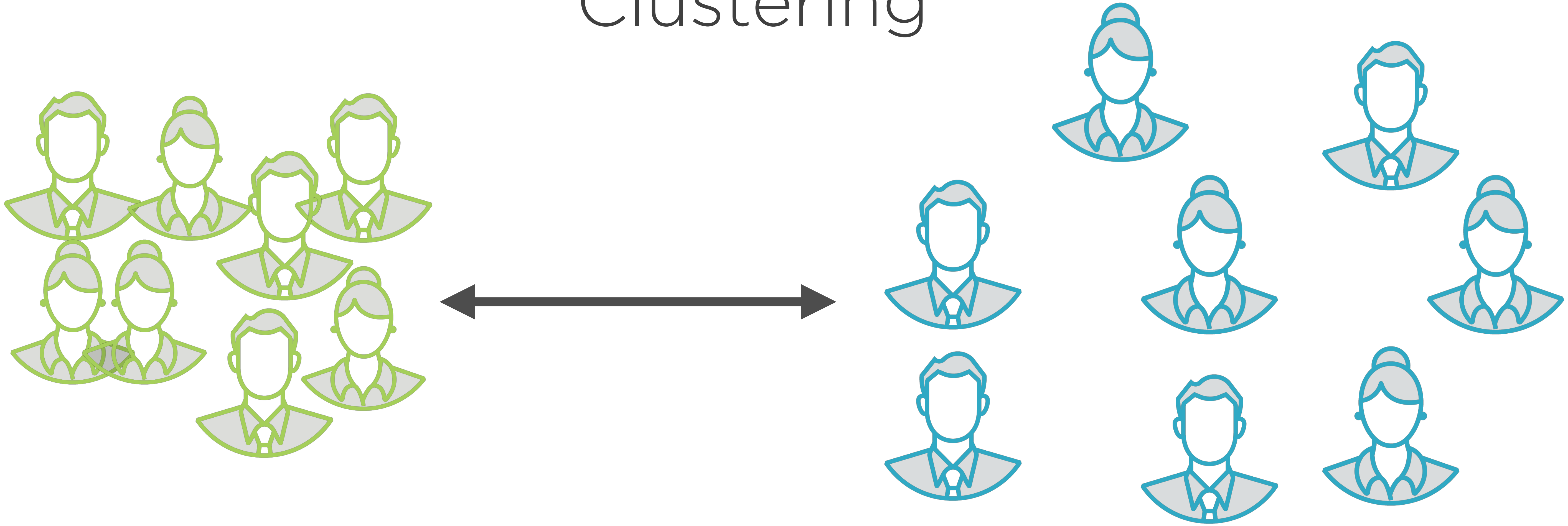
The **distance** between users in a cluster indicates how **similar** they are

Clustering



Maximize intra-cluster similarity

Clustering



Minimize **inter**-cluster similarity

Entities in the **same group are very similar** and entities in **different groups are very different**

Choosing Clustering Algorithms

Size of Dataset

Number of Clusters	Small	Medium	Large
Many			
Moderate			
Few			

Choosing Clustering Algorithms

Size of Dataset

Number of Clusters	Small	Medium	Large
Many			Birch Agglomerative
Moderate			
Few			

Birch, Agglomerative Clustering

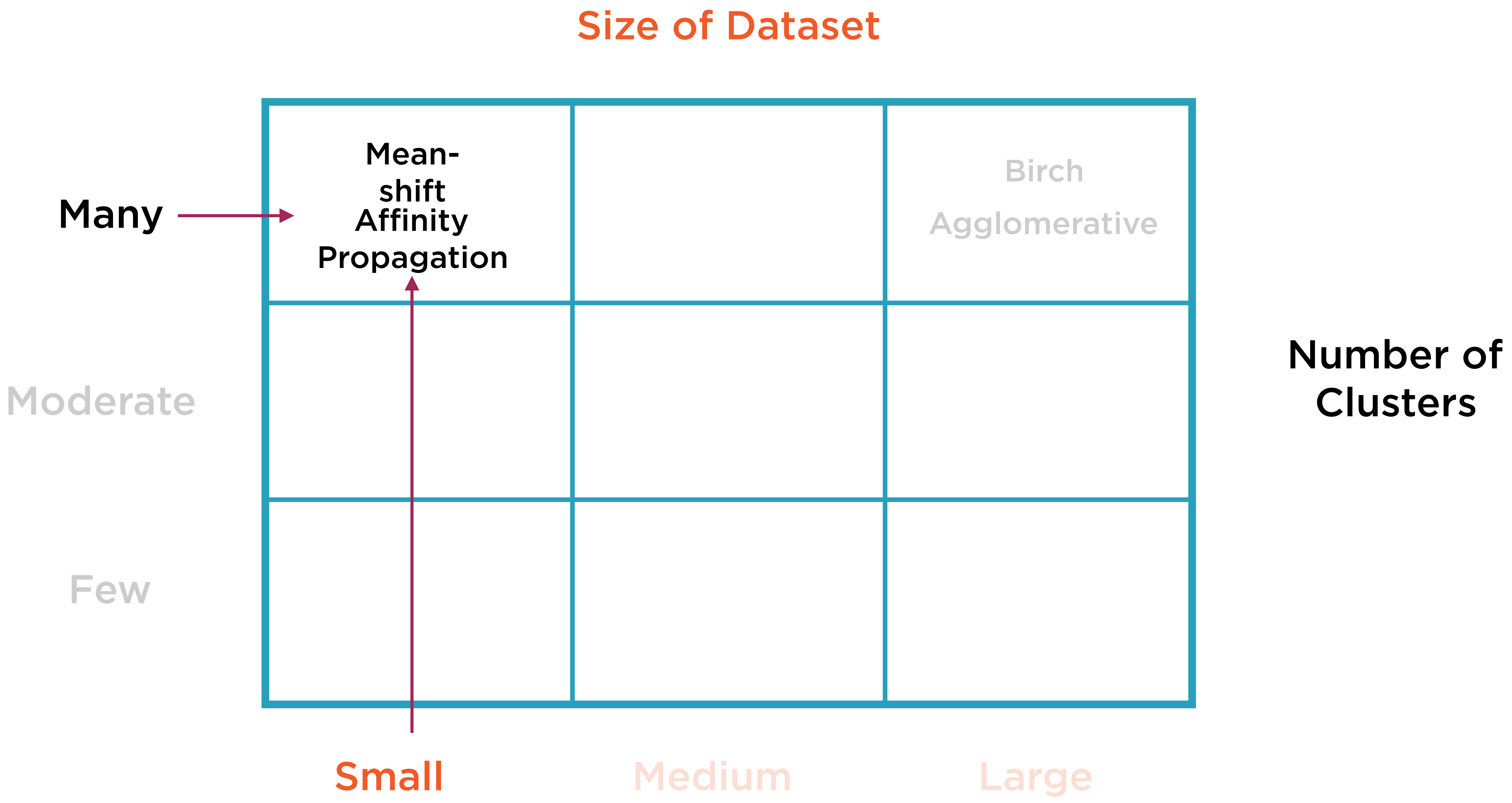
Large datasets, large number of clusters

Birch detects and removes outliers

Also incrementally processes incoming data and updates clusters

Agglomerative clustering works even in absence of Euclidean distance

Choosing Clustering Algorithms



Mean-shift, Affinity Propagation

Small datasets, large number of clusters

Both work well with uneven cluster sizes and manifold shapes

Mean-shift uses pairwise distances between points

Affinity Propagation does not need number of clusters to be specified

Choosing Clustering Algorithms

Size of Dataset

Number of Clusters	Small	Medium	Large
Many	Mean-shift Affinity Propagation		Birch Agglomerative
Moderate			K-means DBSCAN
Few			

K-means, DBSCAN

Large datasets, moderate number of clusters

K-means for even cluster sizes and flat surfaces

Mini-batch K-means tweaks algorithm to be much faster, almost as good

DBSCAN for uneven cluster sizes and manifolds

Choosing Clustering Algorithms

Size of Dataset

Number of Clusters	Small	Medium	Large
Many	Mean-shift Affinity Propagation		Birch Agglomerative
Moderate			K-means DBSCAN
Few		Spectral	

Spectral Clustering

Small datasets, small number of clusters

Simple to implement

Intuitive results for data exploration

Even cluster sizes

Fine for manifolds

Relies on distances between points

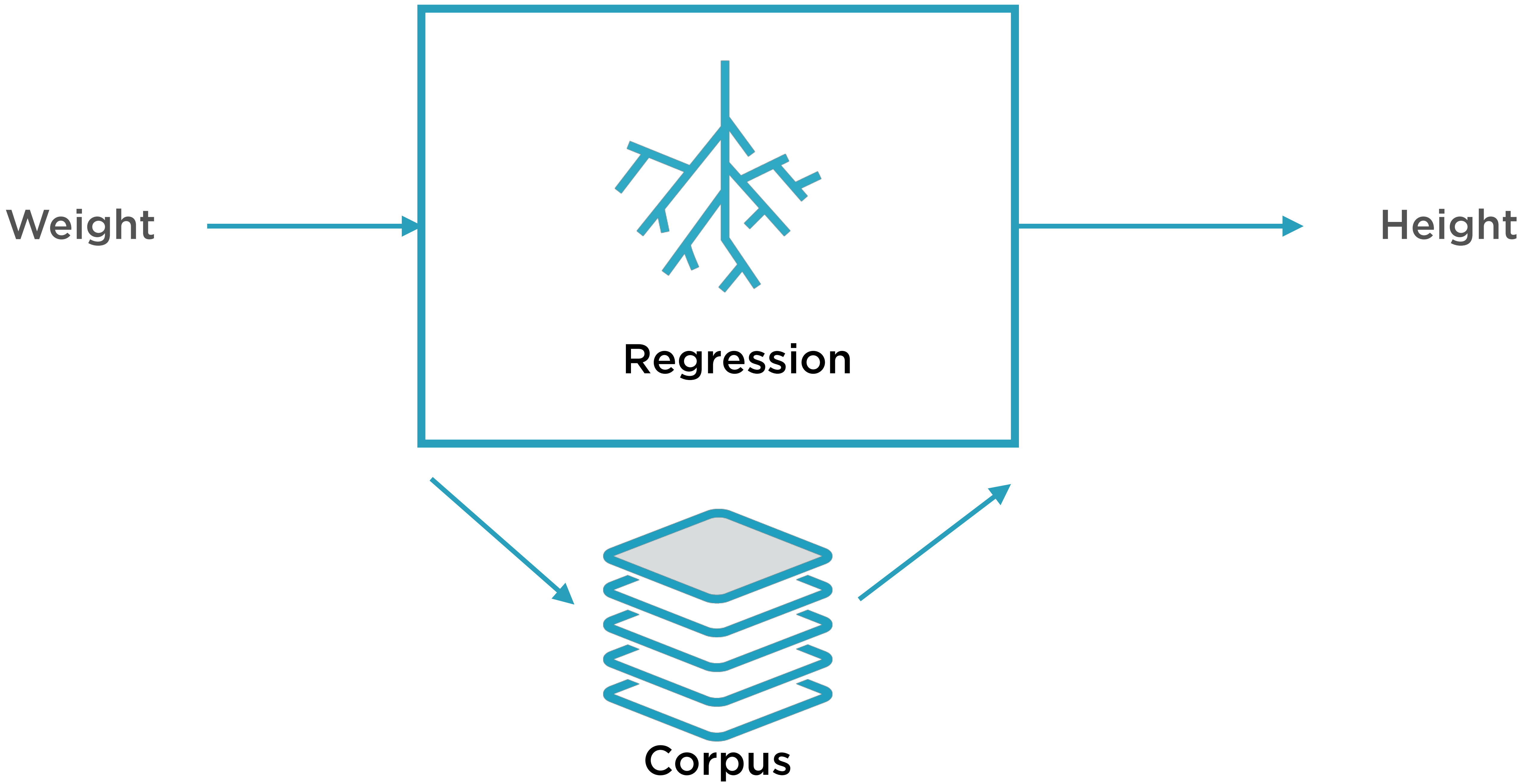
Choosing Clustering Algorithms

Size of Dataset

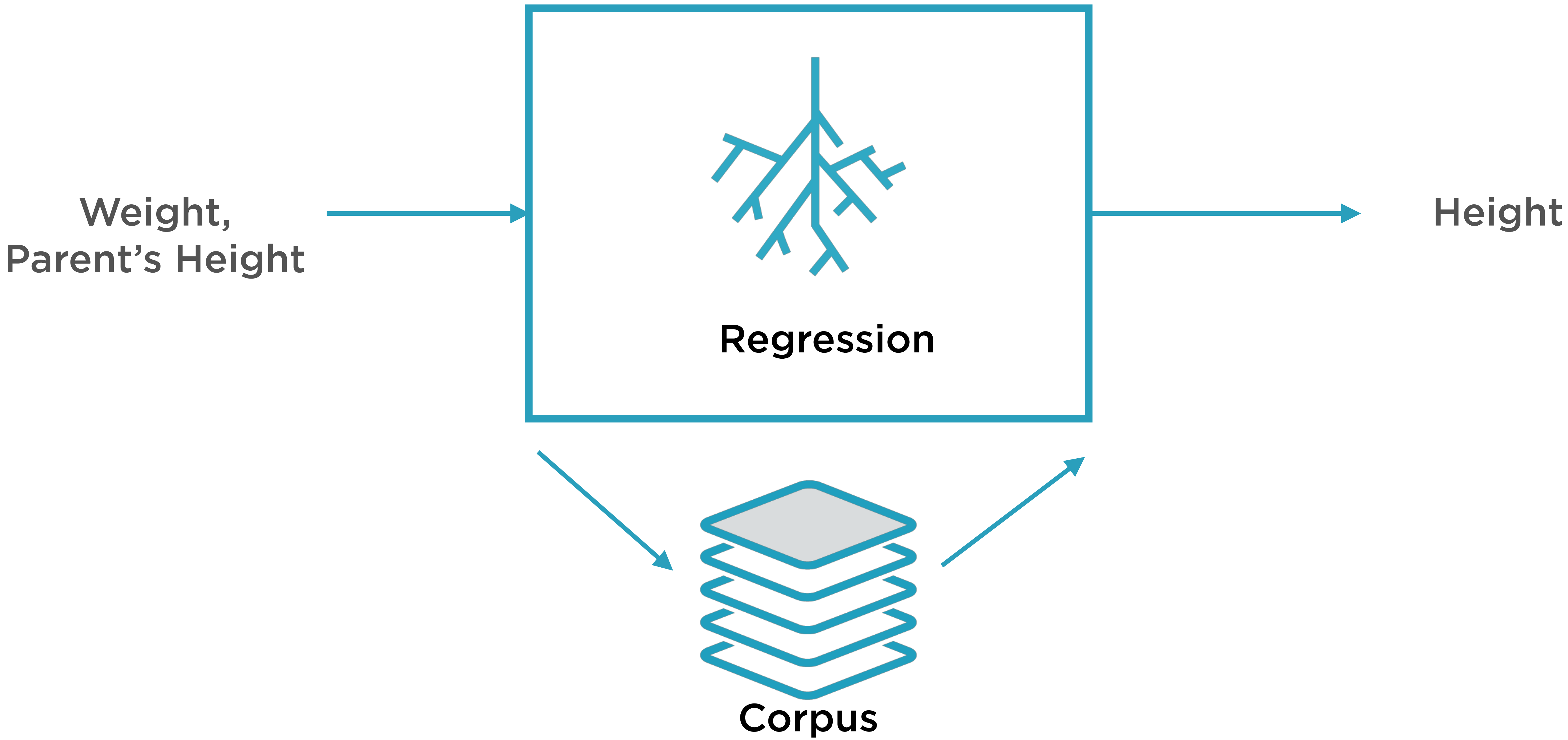
Number of Clusters	Small	Medium	Large
Many	Mean-shift Affinity Propagation		Birch Agglomerative
Moderate			K-means DBSCAN
Few		Spectral	

The Curse of Dimensionality

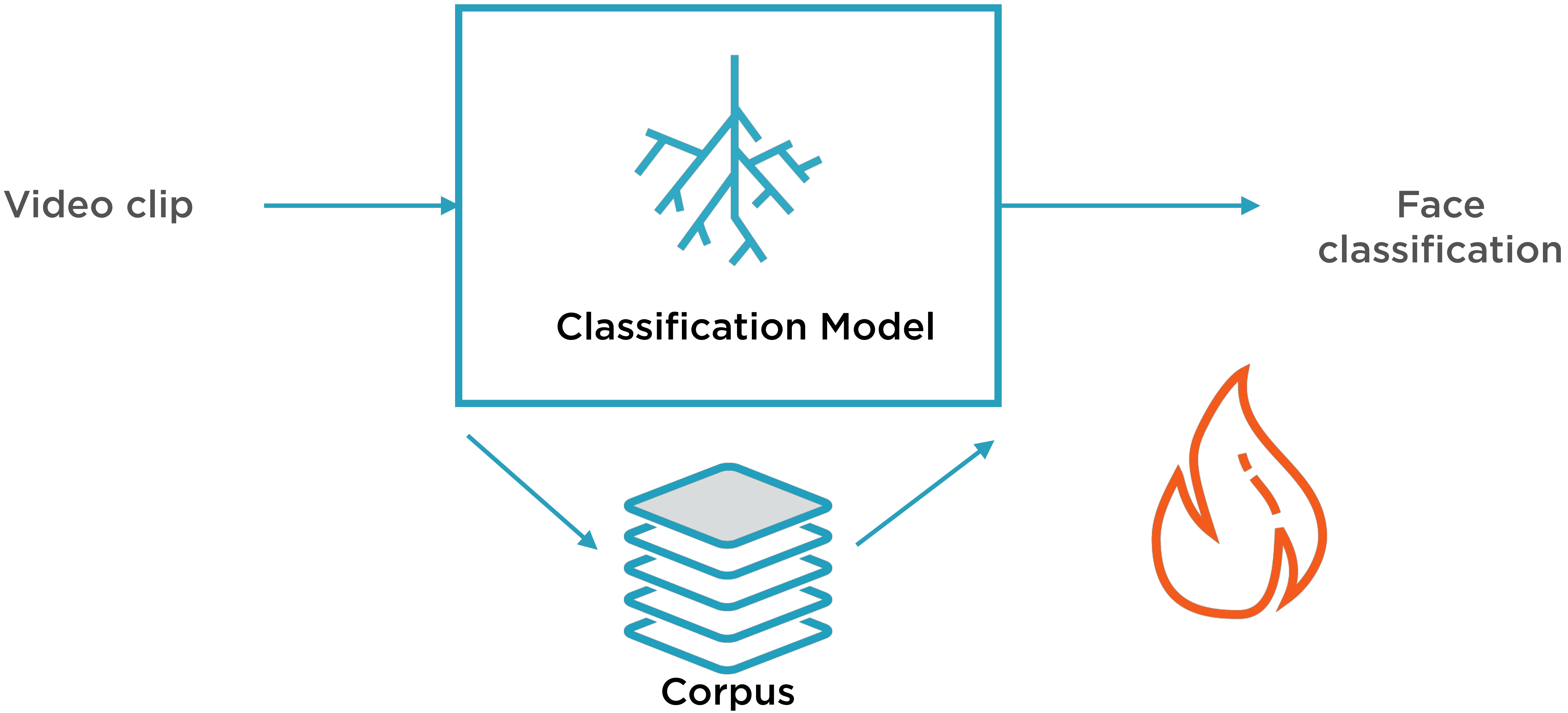
One X Variable



Two X Variables



Dimensionality Explosion



Curse of Dimensionality: As number of **x** variables grows, several problems arise

Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Problems in Visualization

Exploratory Data Analysis (EDA) is an essential precursor to model building

Essential for

- identifying outliers
- detecting anomalies
- choosing functional form of relationships

Problems in Visualization

Two dimensional visualizations are powerful aids in EDA

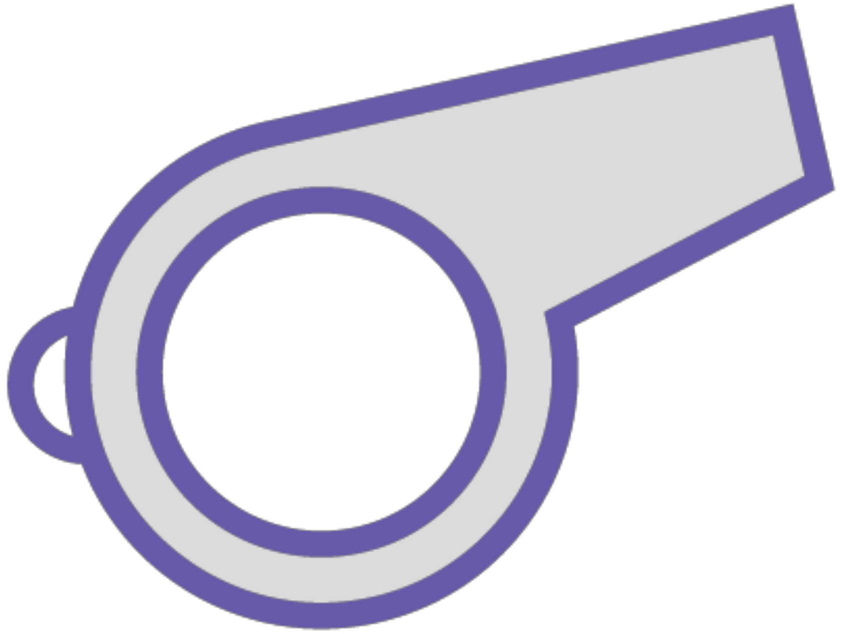
Even three-dimensional data is hard to meaningfully visualize

Higher dimensional data is often imperfectly explored prior to ML

Curse of Dimensionality

Problems in Visualization	**Problems in Training**	**Problems in Prediction**

Problems in Training



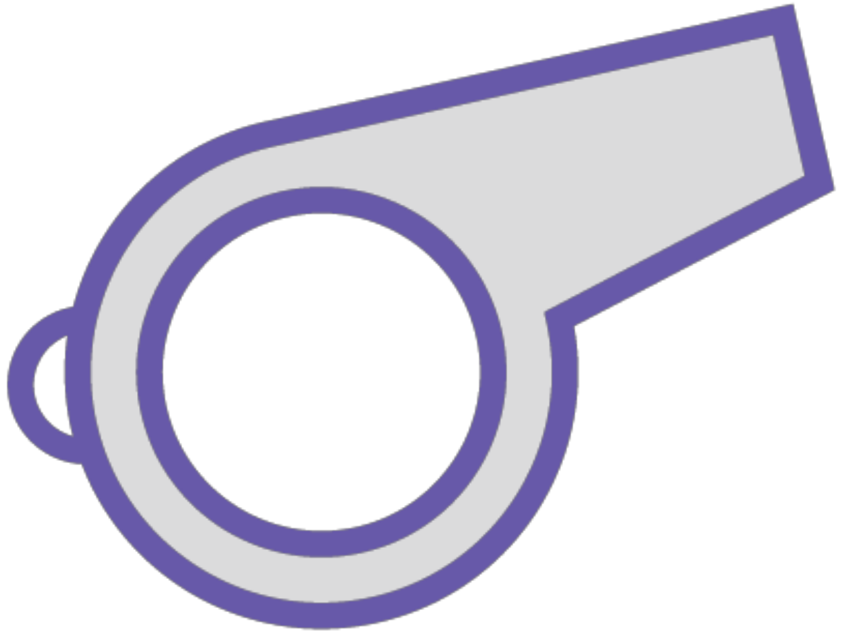
Training is the process of finding best model parameters

Complex models have thousands of parameter values

Many parameters may be useless or noisy

Training for too little time leads to bad models

Problems in Training



Number of parameters to be found grows rapidly with dimensionality

Extremely time-consuming

For on-cloud training, also extremely expensive

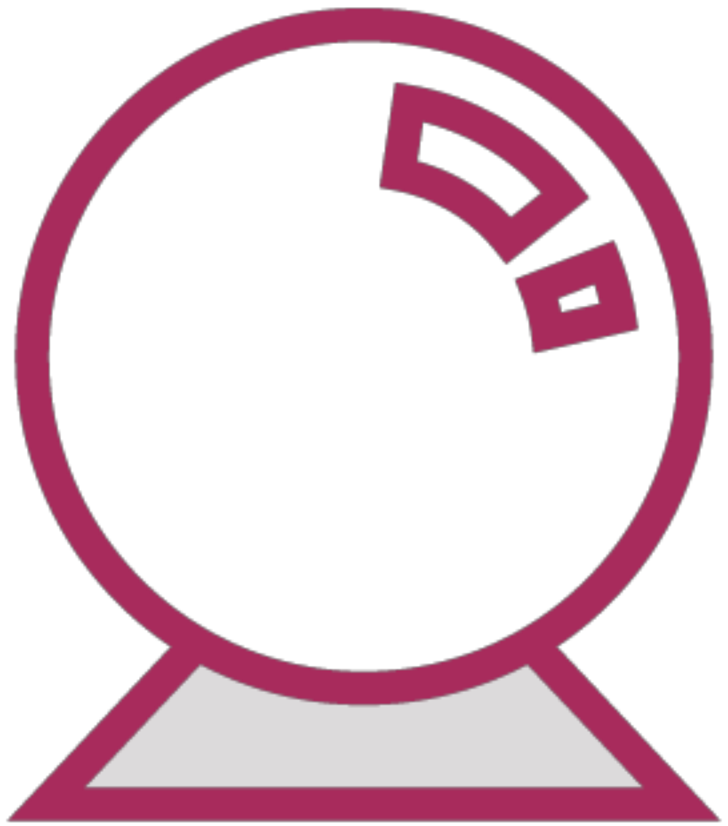
Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Problems in Prediction

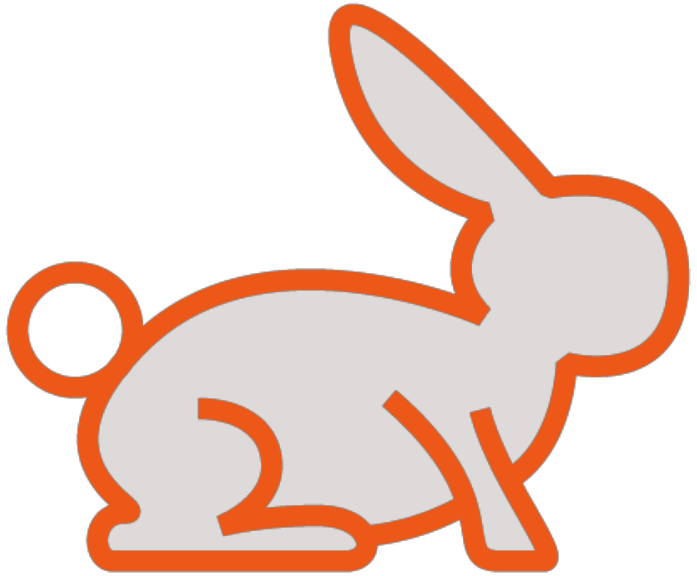


Prediction involves finding training instances similar to test instance

As dimensionality grows, size of search space explodes

Higher the number of X variables, higher the risk of overfitting

Overfitting



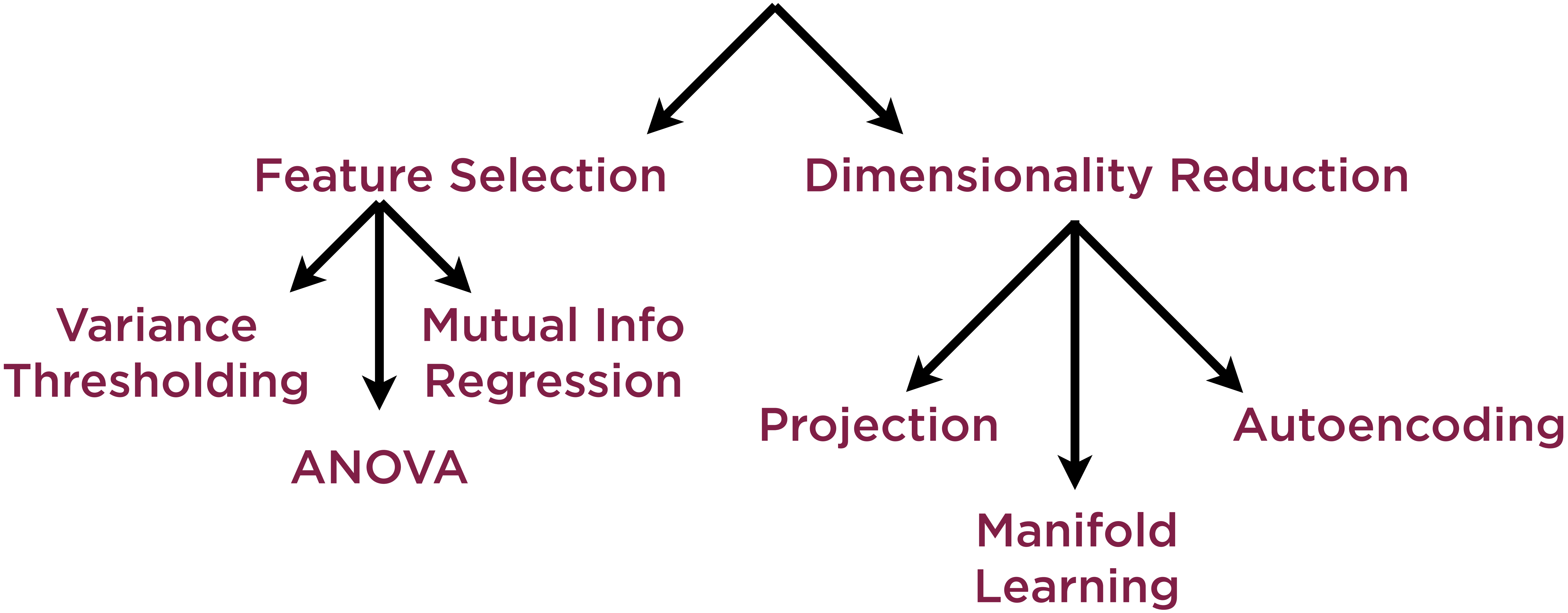
Low Training Error

Model does very well in training...

High Test Error

...but poorly with real data

Reducing Complexity



Reducing Complexity

Feature Selection

Dimensionality Reduction

Choose a subset of original X variables

Projection

Manifold Learning

Autoencoding

Reducing Complexity

Feature Selection

Dimensionality Reduction

Transform original X variables into new dimensions

Projection

Manifold Learning

Autoencoding

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Find new, better axes
and re-orient data

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

e.g. PCA, Factor Analysis, LDA, QDA

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Works best with linear data (can use kernel trick to extend to non-linear data)

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Autoencoding

Manifold Learning

Unroll the data so that twists and turns are smoothened out

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Autoencoding

Manifold Learning

Works best when data lies along a rolled-up surface such as a Swiss Roll or S-curve

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Autoencoding

e.g. MDS, Isomap,
LLE, Kernel PCA

Manifold Learning

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Build neural networks to simplify the data

Summary

Choosing and evaluating

- Regression models
- Classification models
- Clustering models
- Dimensionality reduction techniques