Performing Regression Using Multiple Techniques

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

Overview

Choosing regression algorithms based on dataset size and features

Support Vector Machines (SVM)

Nearest Neighbors regression

Stochastic Gradient Descent

Decision Tree regression

Least-angle Regression (LARS)

Choosing Regression Algorithms

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

Choosing Regression Algorithms

Size of Dataset

	Small	Medium	Large
Many			
Moderate			
Few			

Number of Features

100K+ Data Points: Use SGD

Size of Dataset

	Small	Medium	Large
Many			Stochastic Gradient Descent (SGD)
Moderate			
Few			

Number of Features

More Features Than Samples: Use LARS

Size of Dataset

Number of Features

	Small	Medium	Large
Many	Least Angle Regression (LARS)		Stochastic Gradient Descent (SGD)
Moderate			
Few			

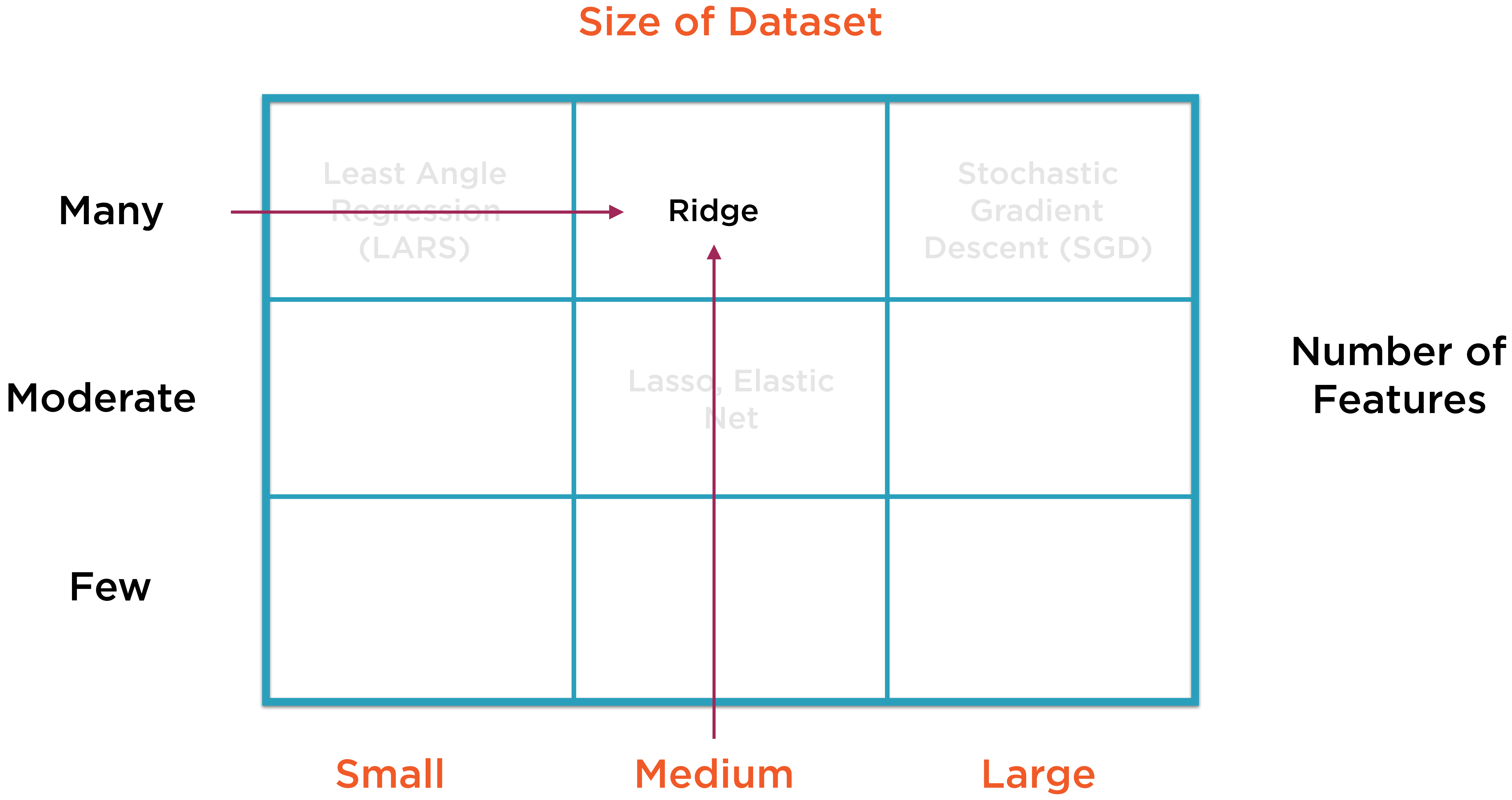
Many Features, Few Useful: Lasso, ElasticNet

Size of Dataset

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

Number of Features

Many Features, Most Useful: Ridge



Size of Dataset

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

Number of Features

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

	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)

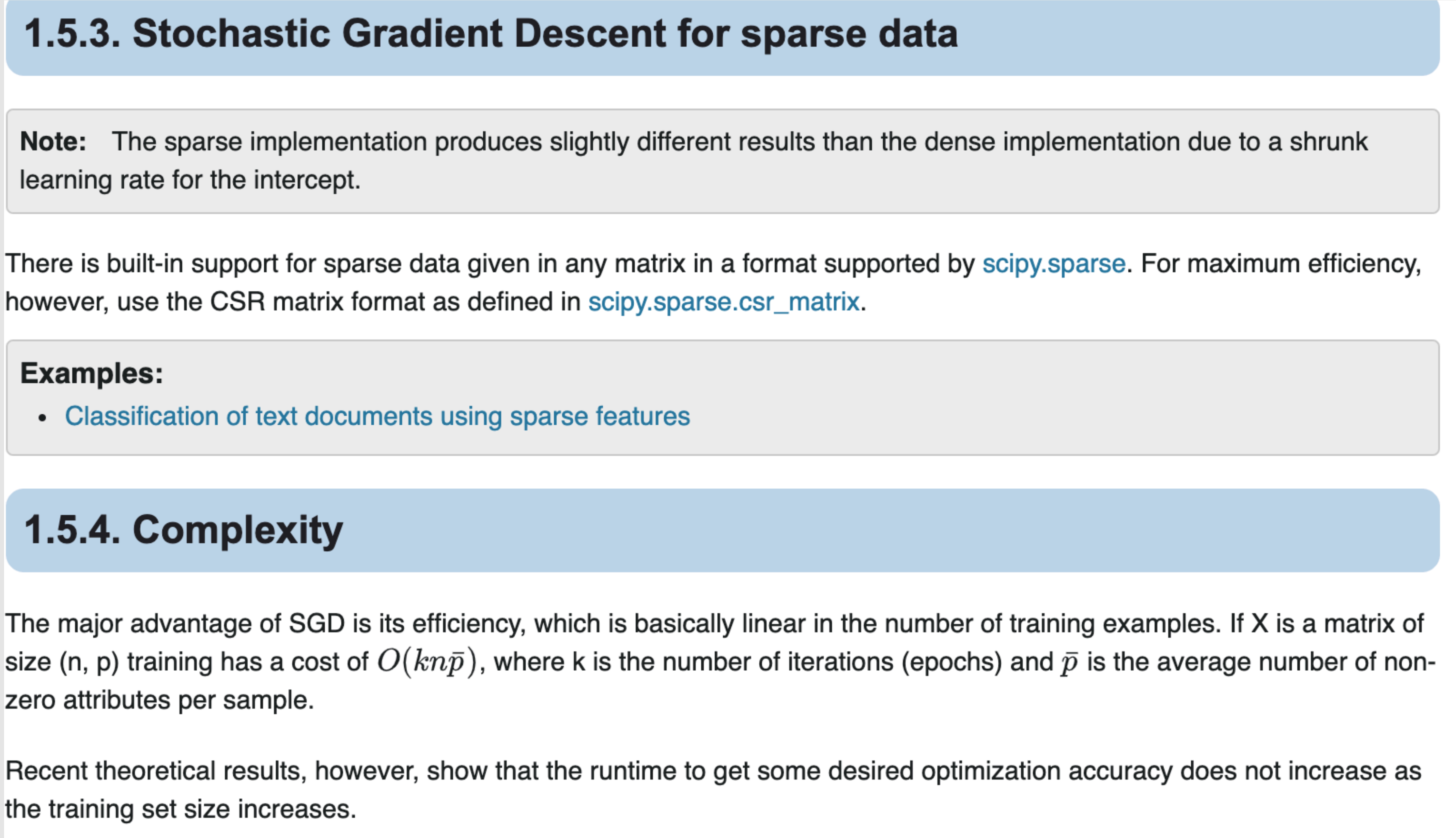
Number of Features

Choosing Regression Algorithms

Size of Dataset

	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)

Number of Features



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:

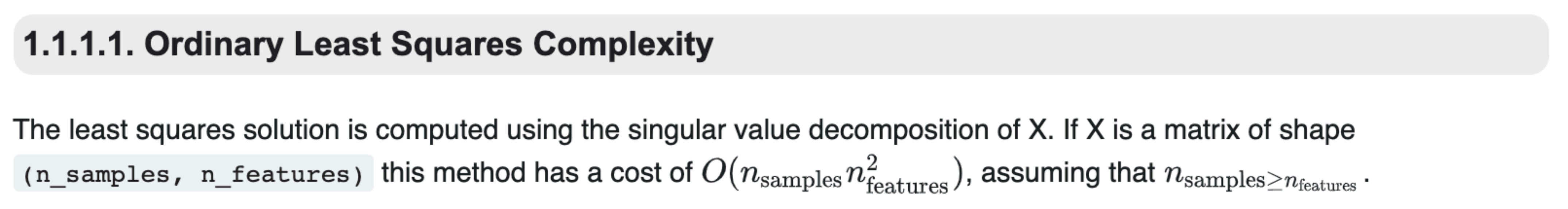
- Classification of text documents using sparse features

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.

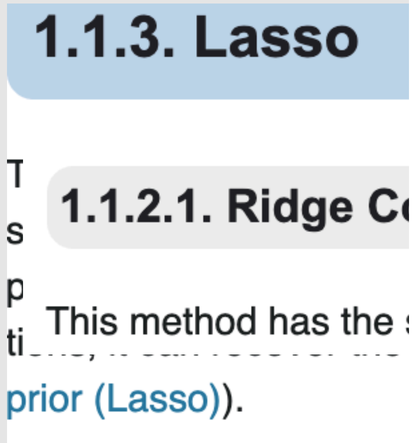
Recent theoretical results, however, show that the runtime to get some desired optimization accuracy does not increase as the training set size increases.

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

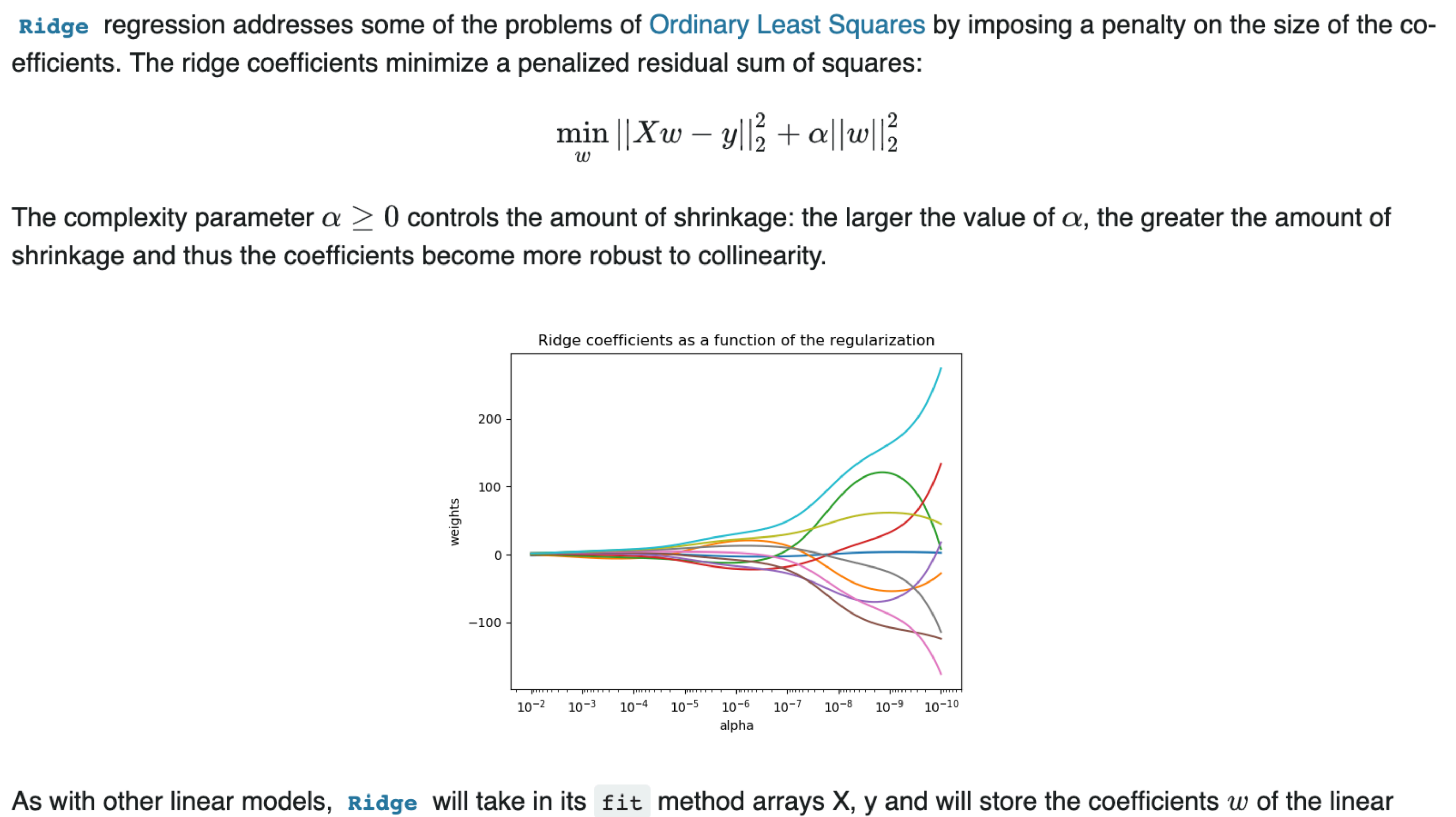
p

ti

1.1.2.1. Ridge C

This method has the

prior (Lasso)).

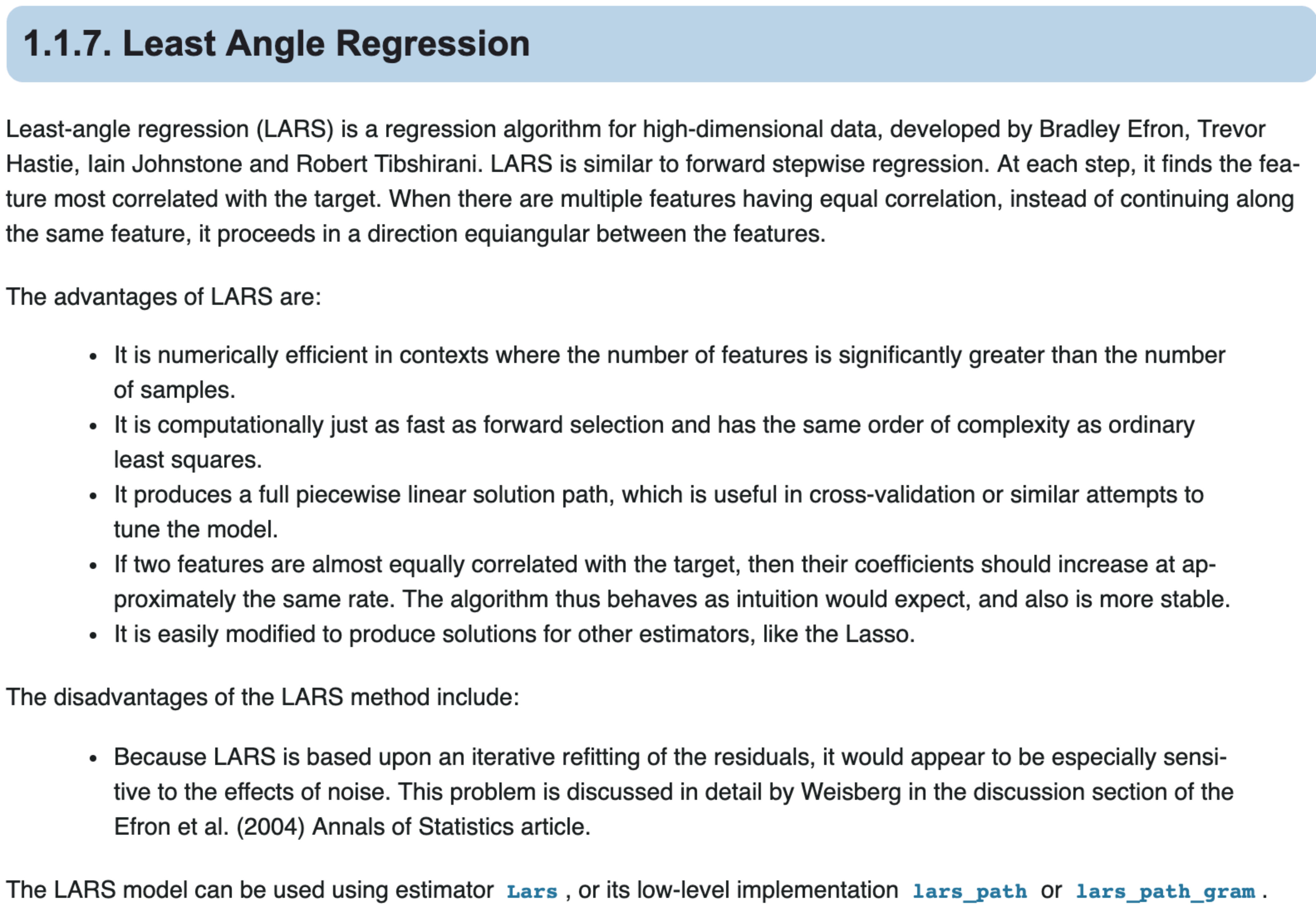


Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

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

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.

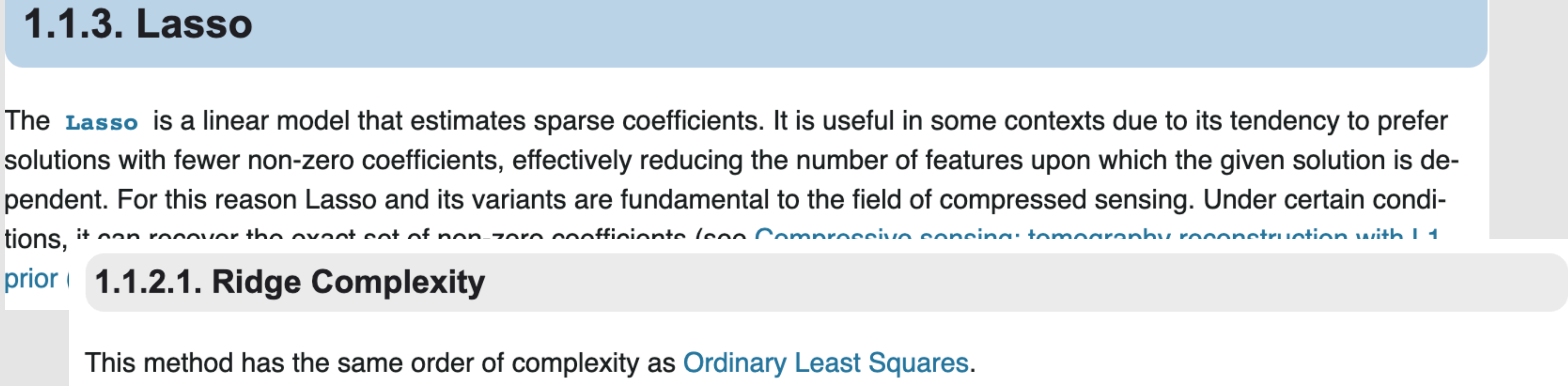
The advantages of LARS are:

- 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` .

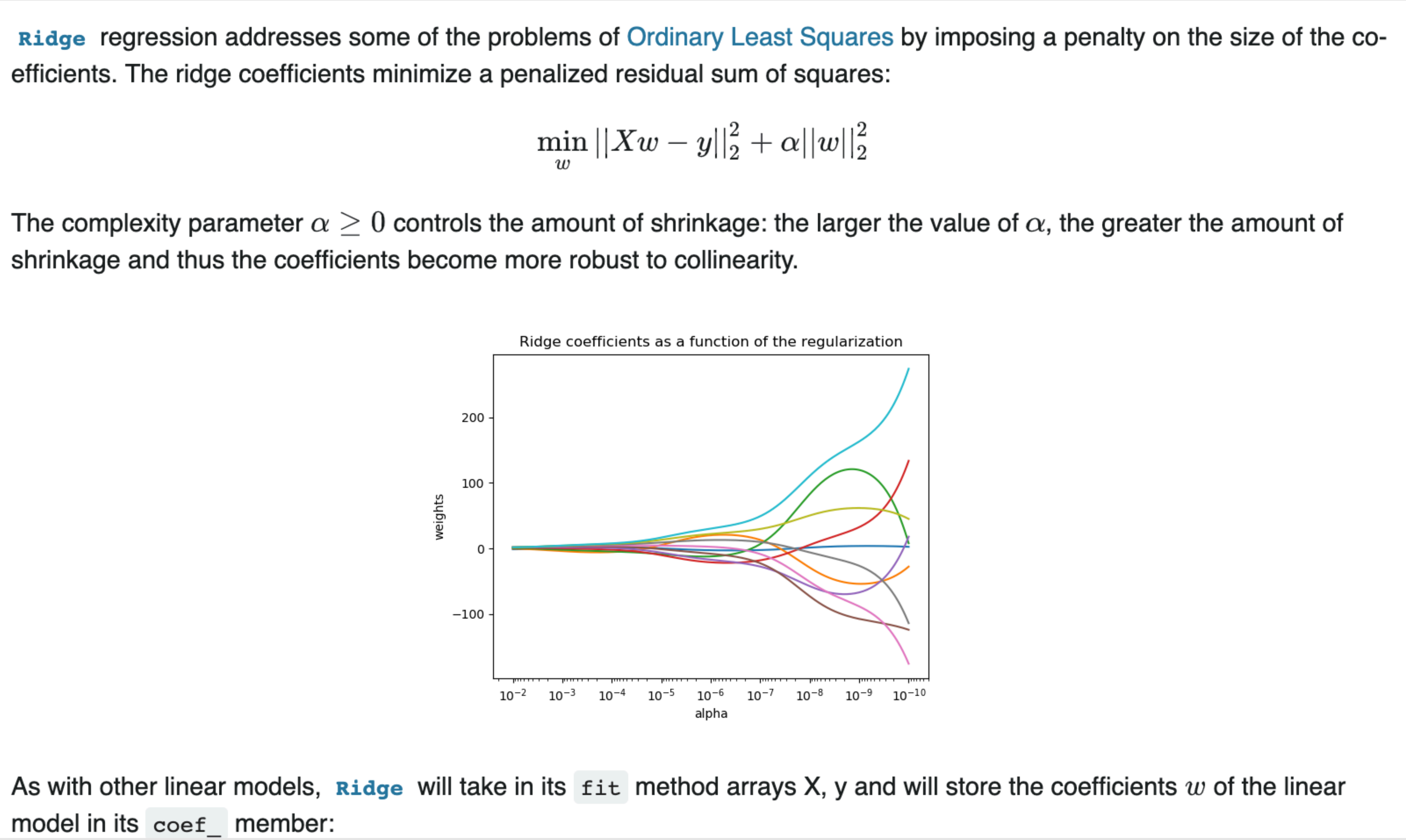


1.1.3. Lasso

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.



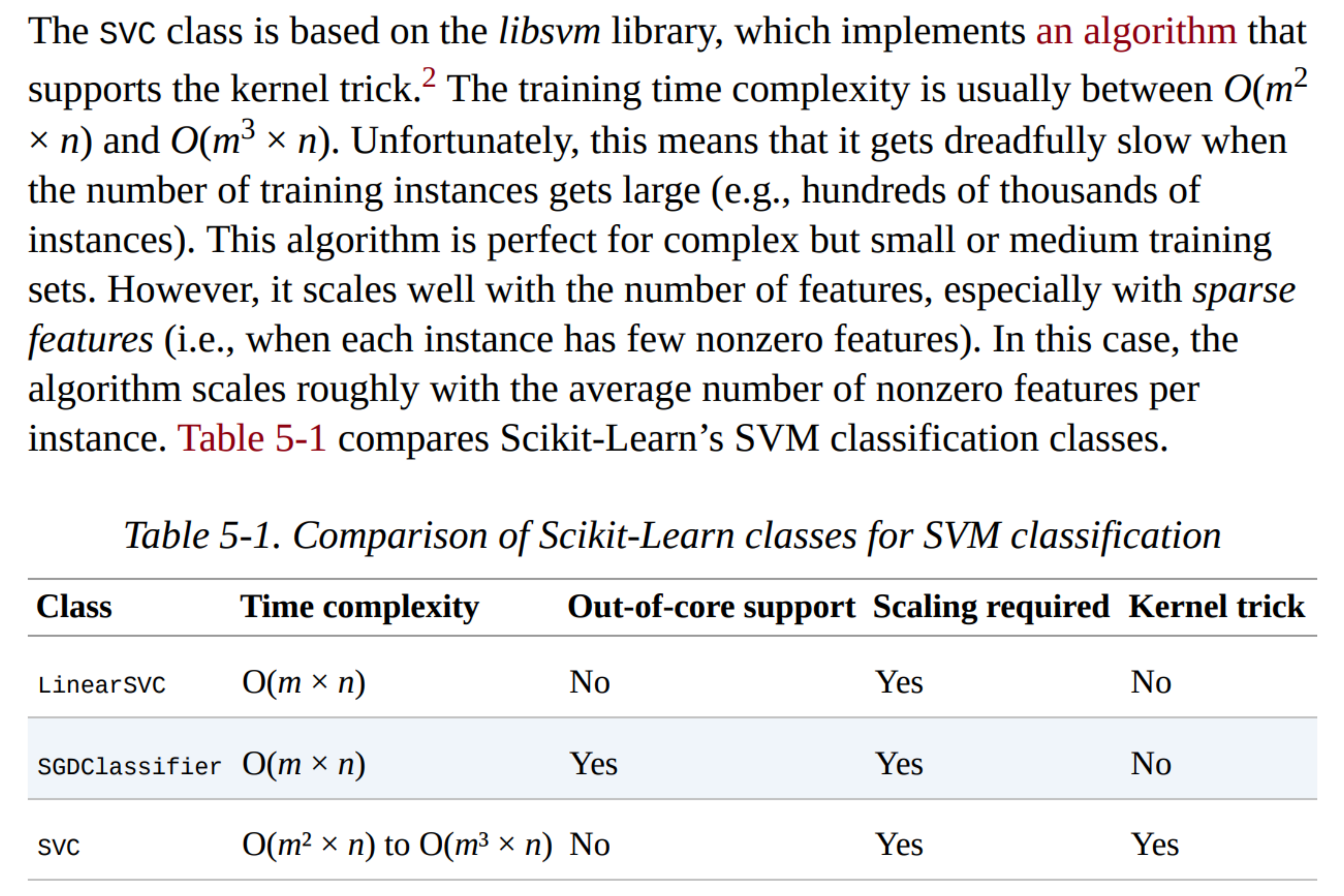
`Ridge` regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

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

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

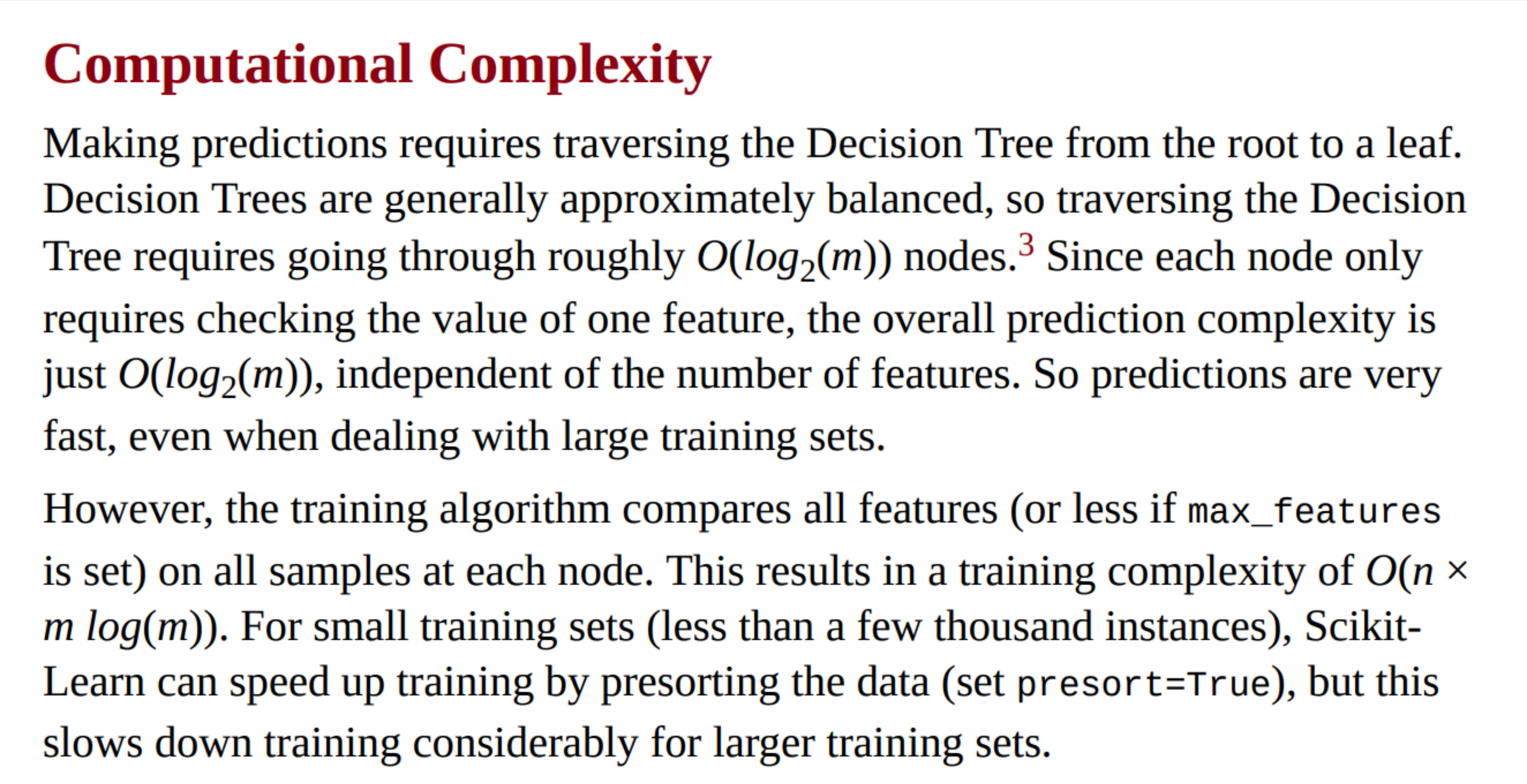


The `SVC` class is based on the *libsvm* library, which implements an algorithm that supports the kernel trick.[2] The training time complexity is usually between $O(m^2 \times n)$ and $O(m^3 \times n)$. Unfortunately, this means that it gets dreadfully slow when the number of training instances gets 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)

Support Vector Regression

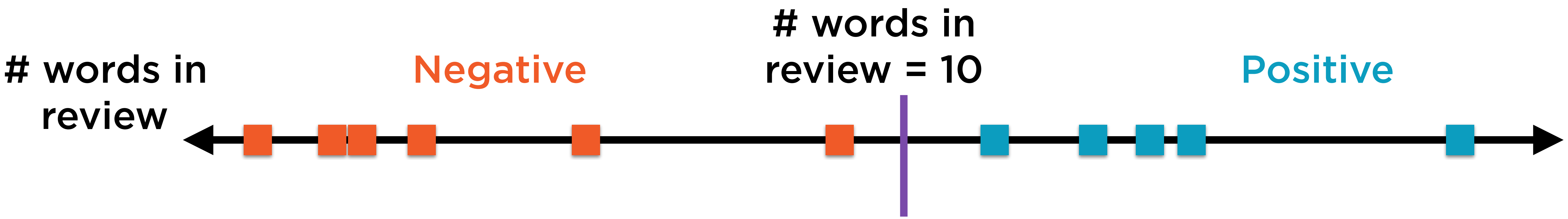
SVMs are typically used for classification problems

SVRs use the same underlying principles with a **different objective function**

Data in One Dimension

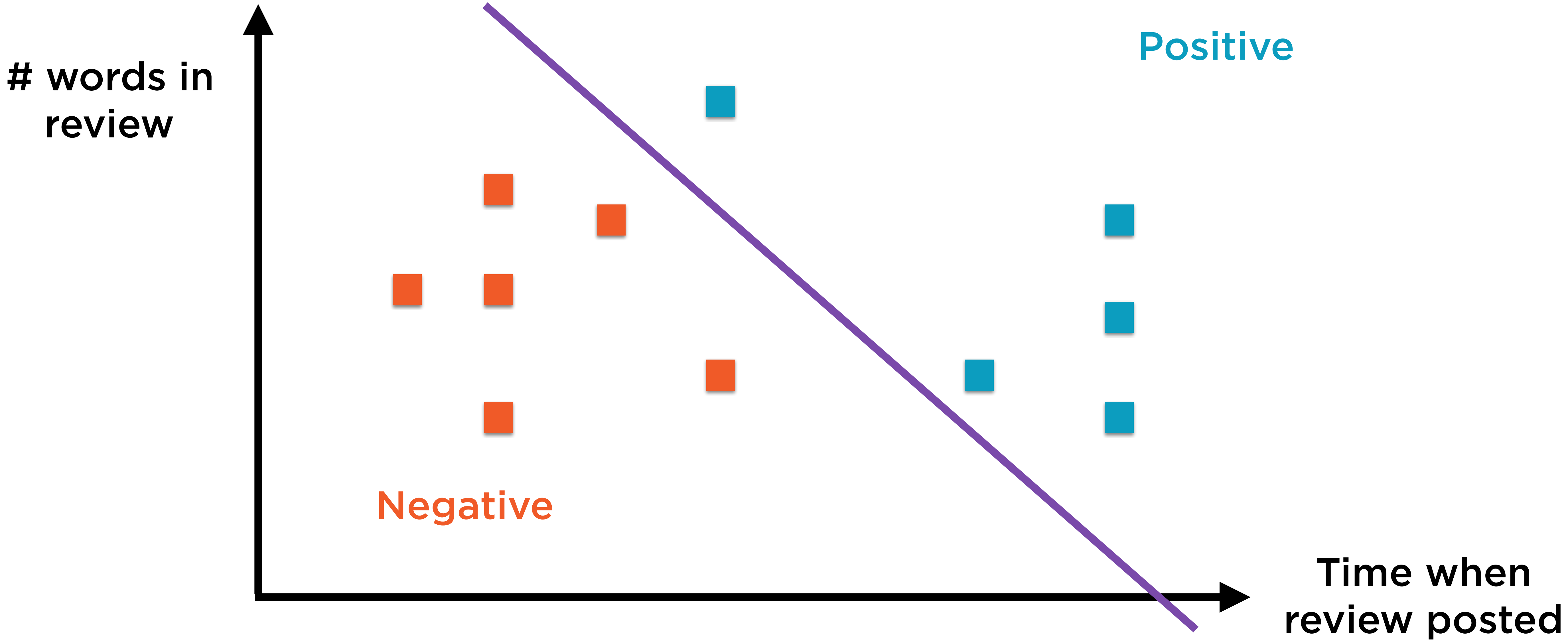
Unidimensional data points can be represented using a line, such as a number line

Data in One Dimension



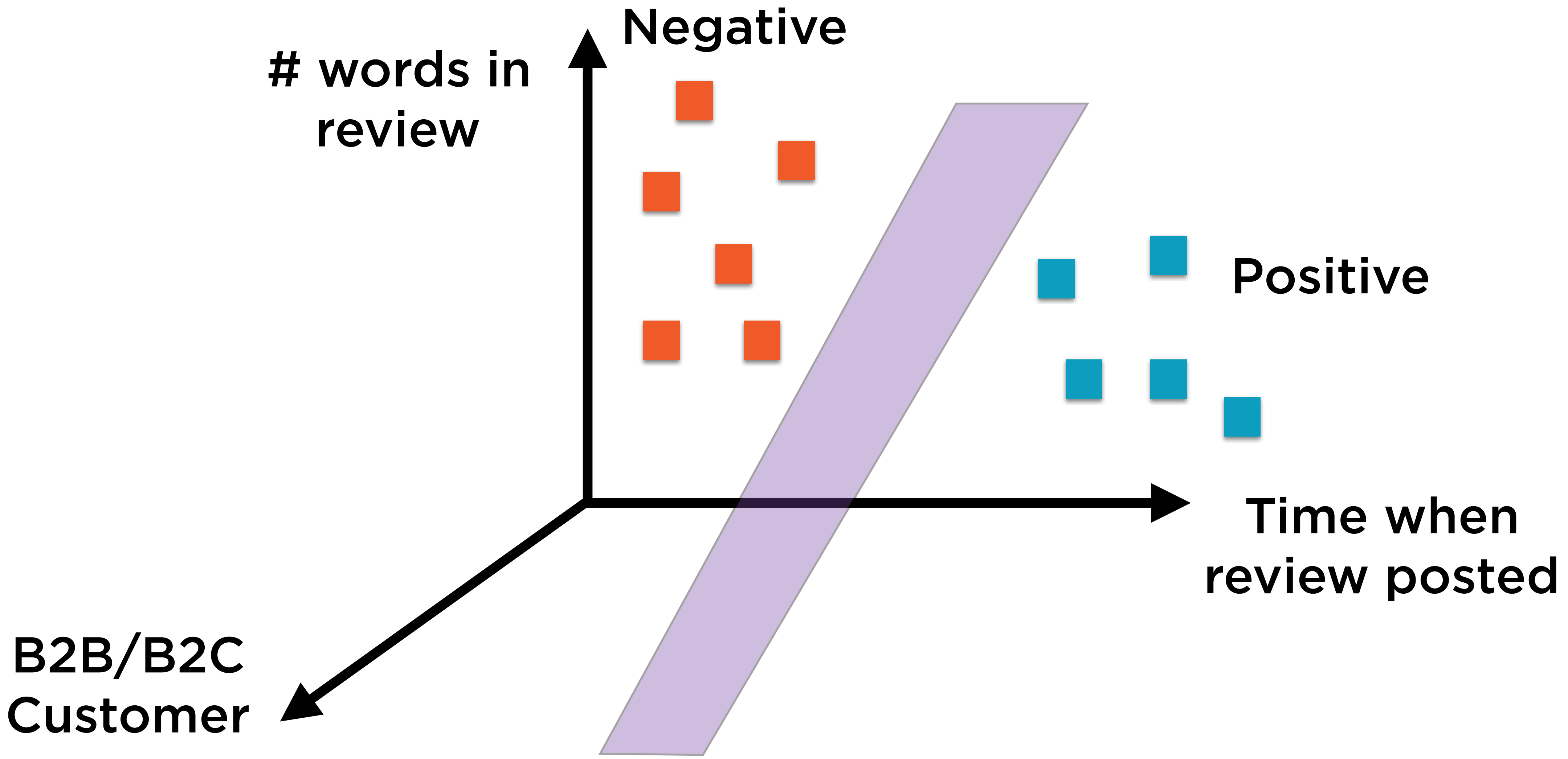
Unidimensional can also be separated, or classified, using a point

Data in Two Dimensions



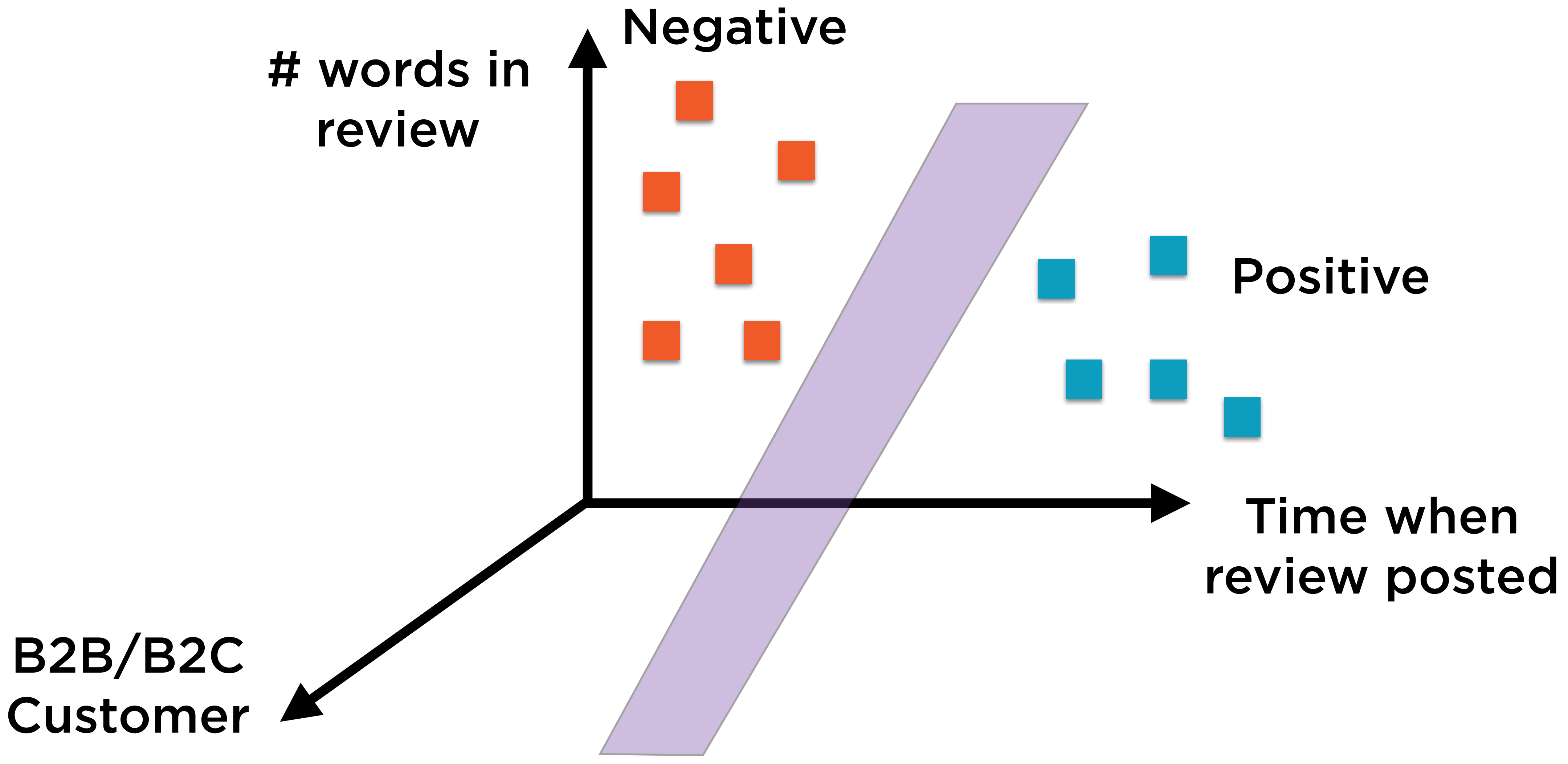
Bidimensional data points can be represented using a plane, and classified using a line

Data in N Dimensions



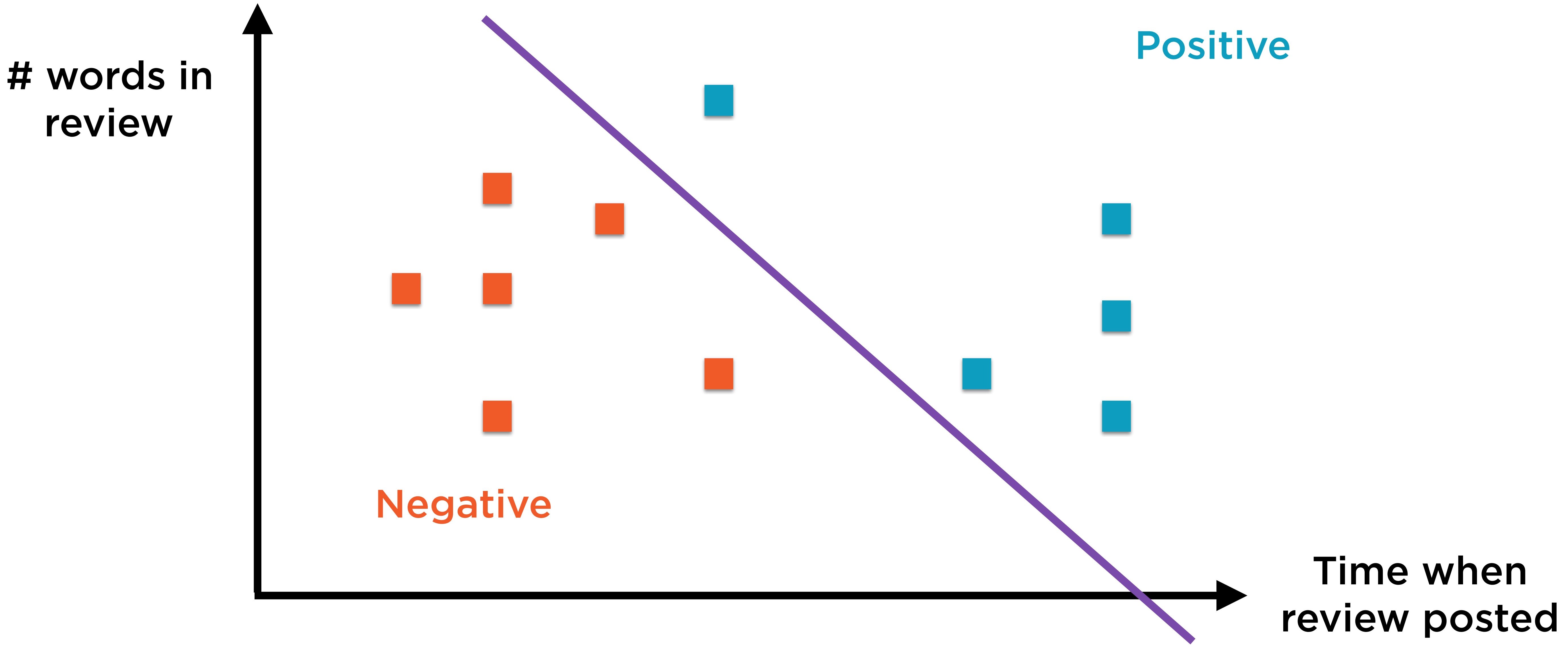
N-dimensional data can be represented in a hypercube, and classified using a hyperplane

Support Vector Machines



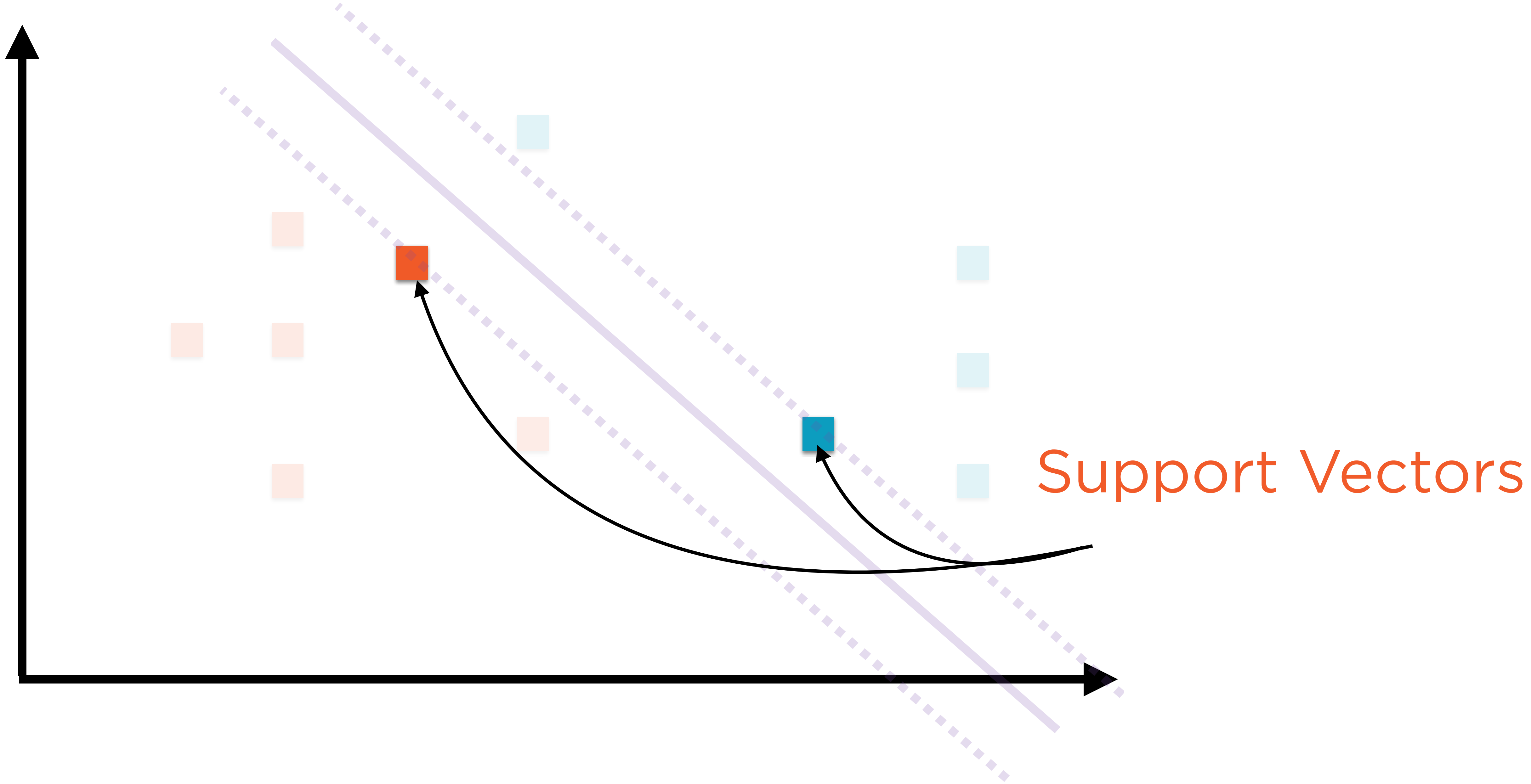
SVM classifiers find the hyperplane that best separates points in a hypercube

Classification



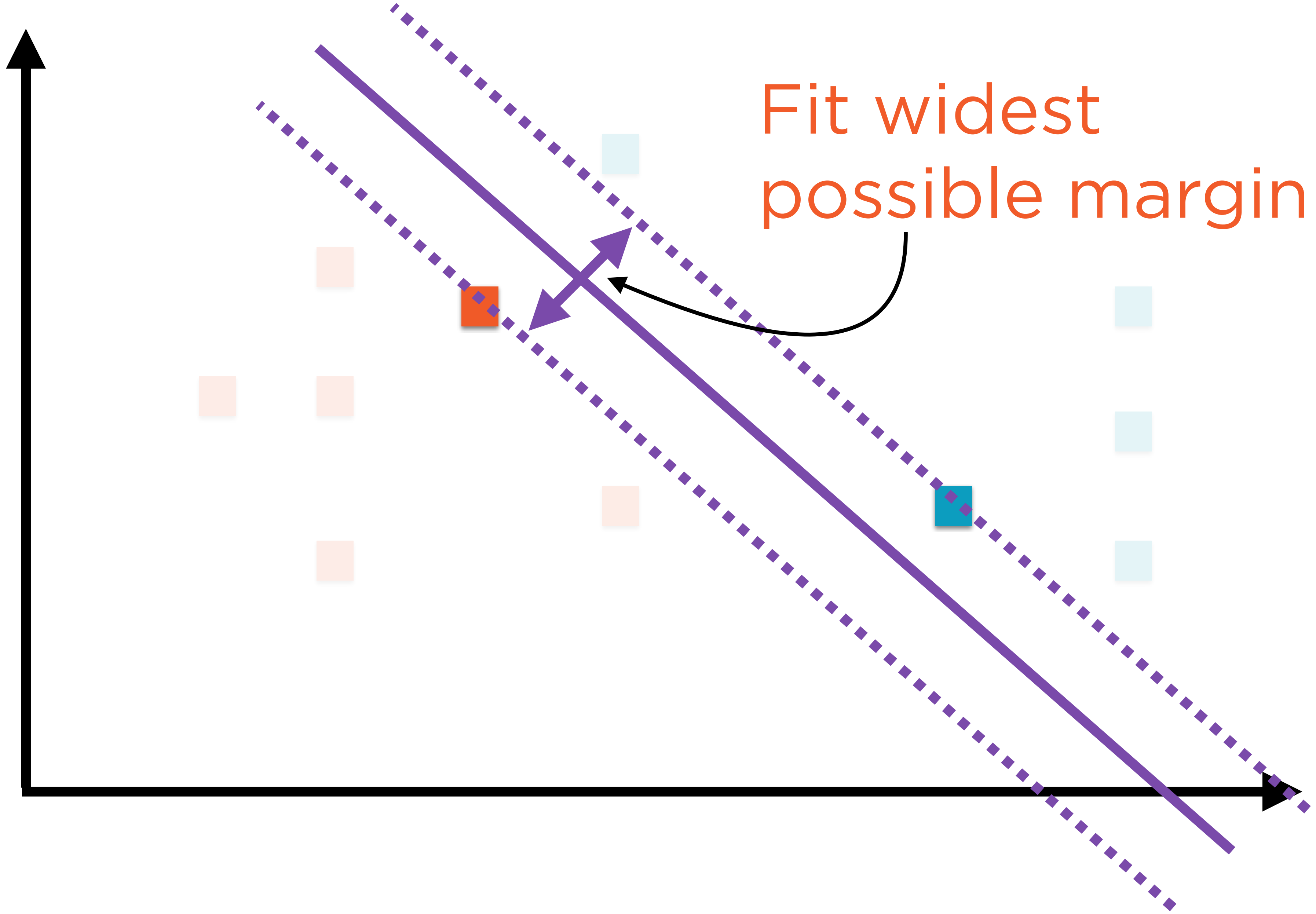
Ideally, data is linearly separable - hard decision boundary

Classification



The nearest instances on either side of the boundary are called the support vectors

Classification

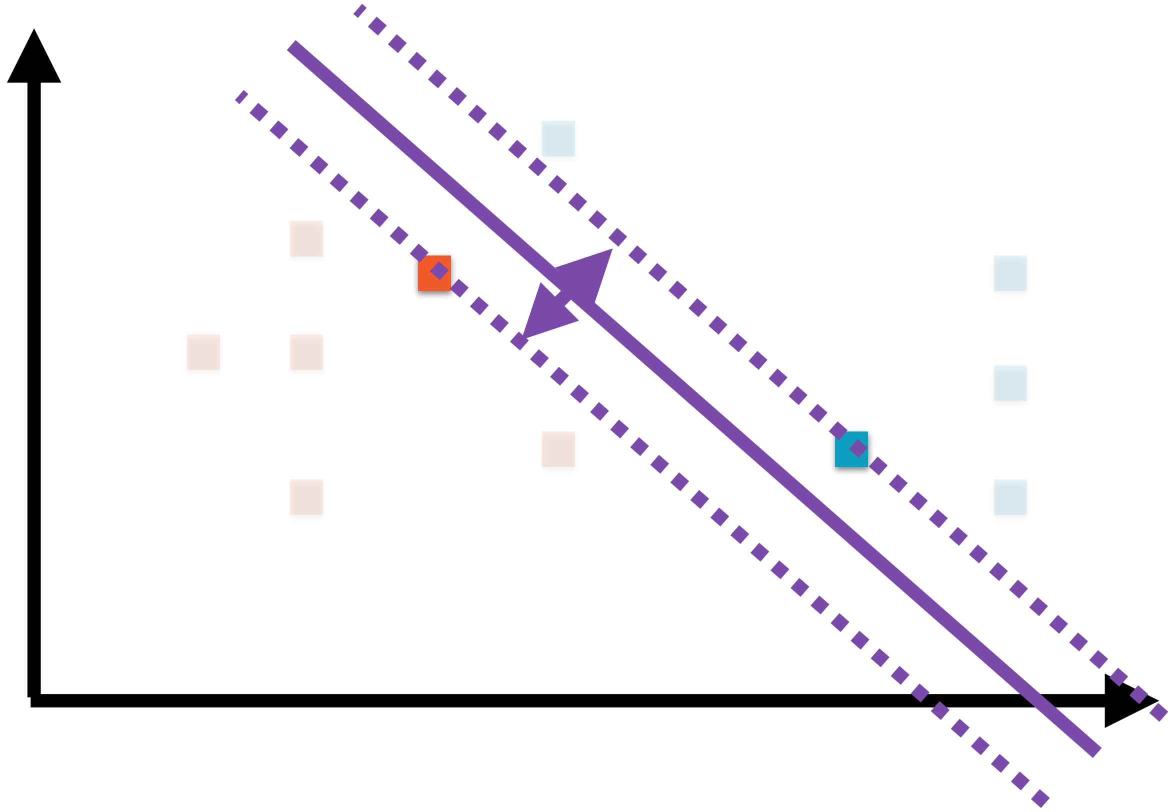


Fit widest possible margin

SVM finds the widest street between the nearest points on either side

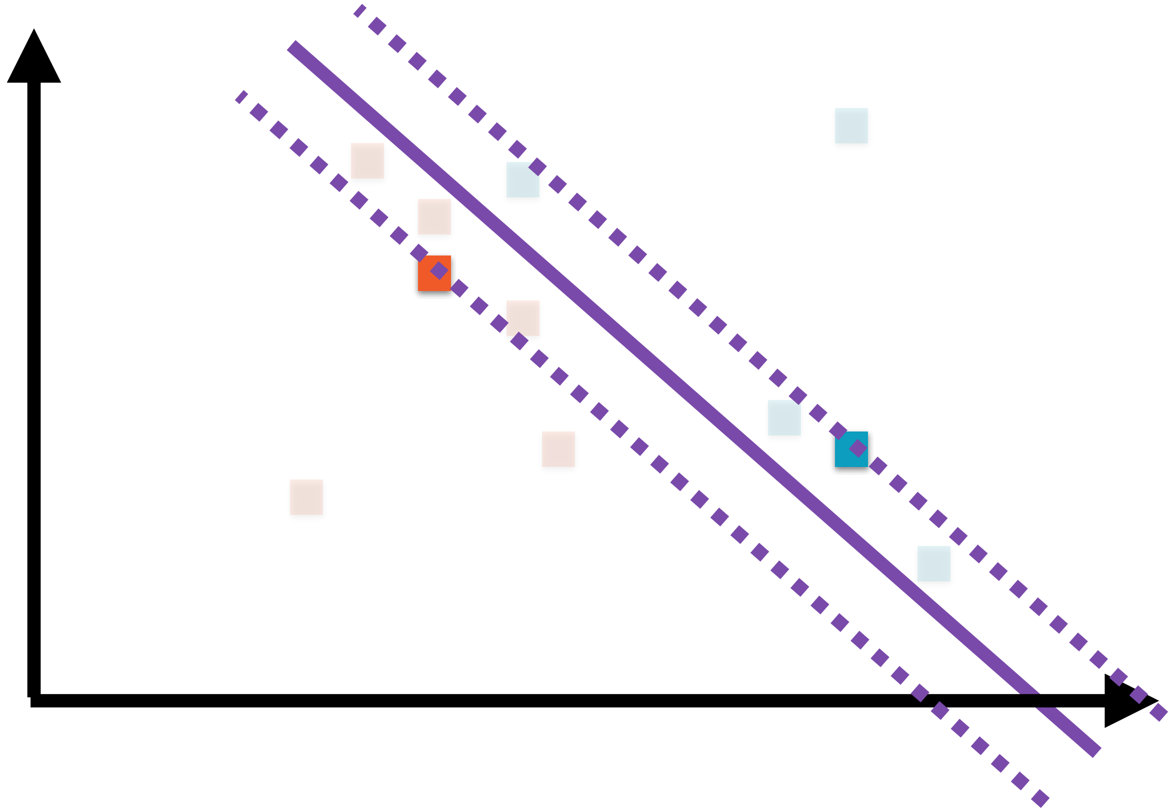
Similar, yet Different

SVM Classification



Find widest margin with most distance from nearest points (support vectors)

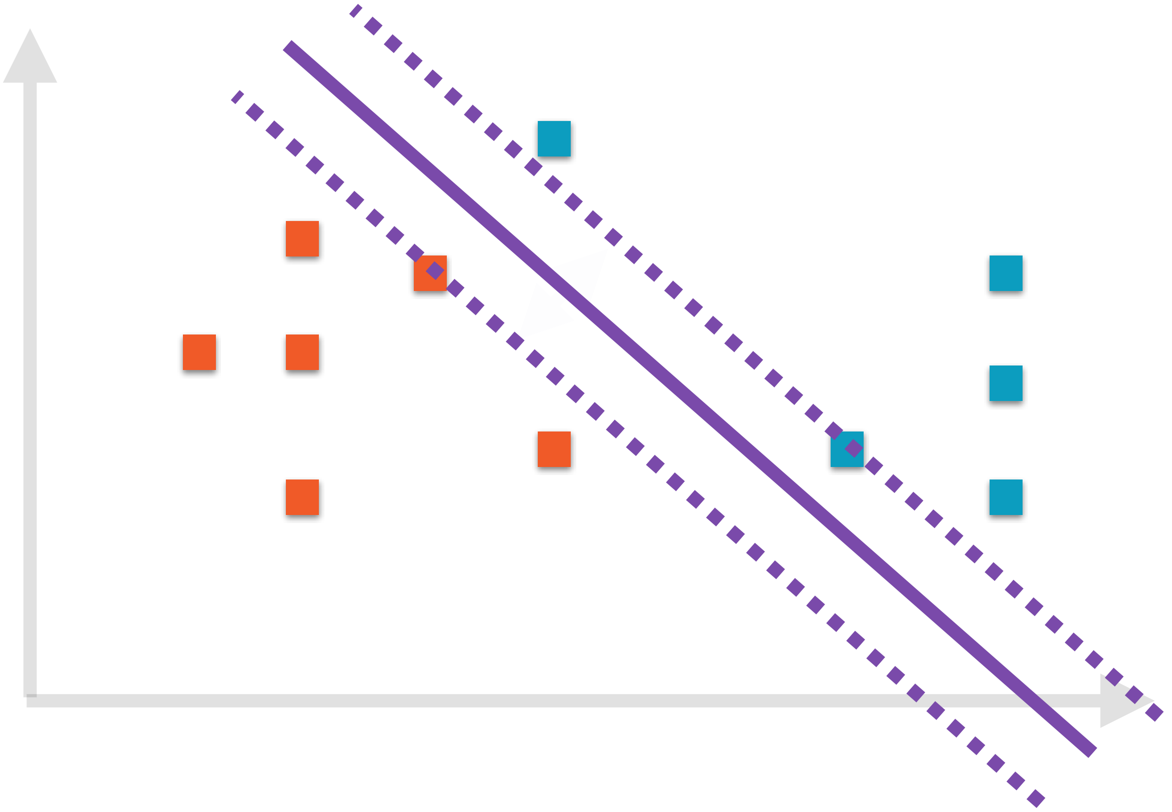
SVM Regression



Find line that “best fits” the points

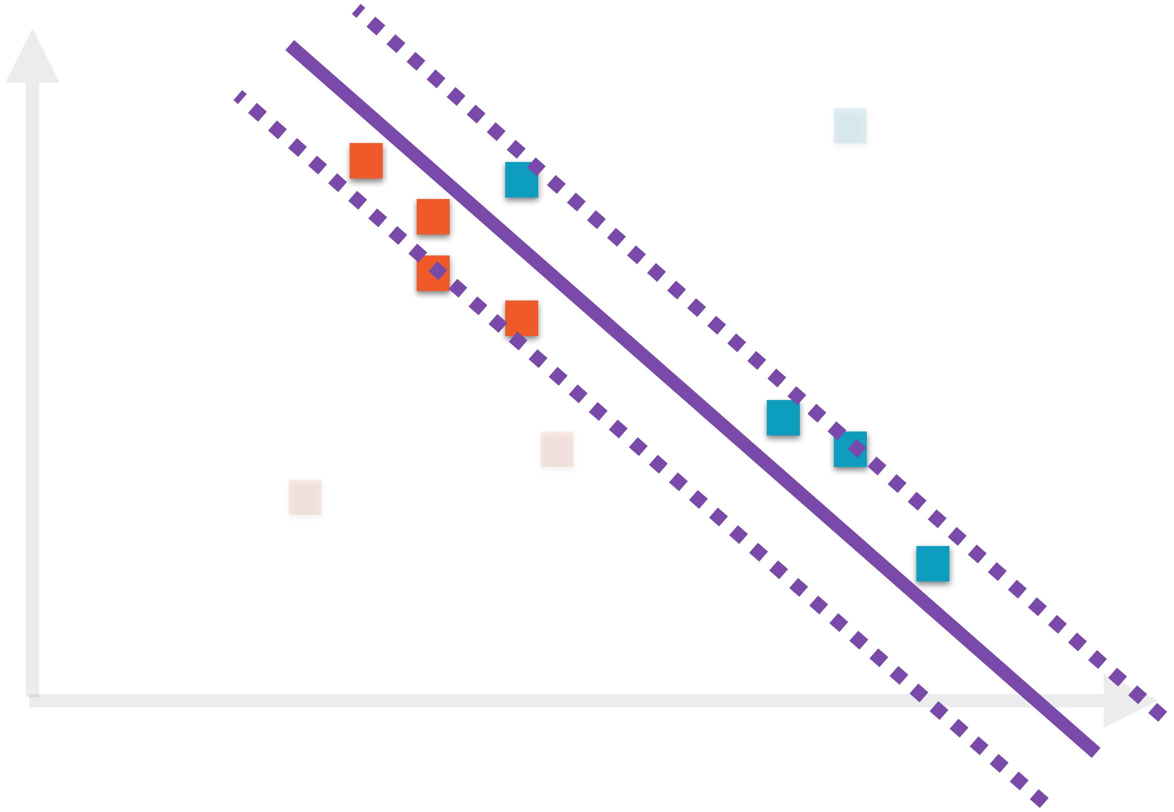
Similar, yet Different

SVM Classification



No points are inside the margin

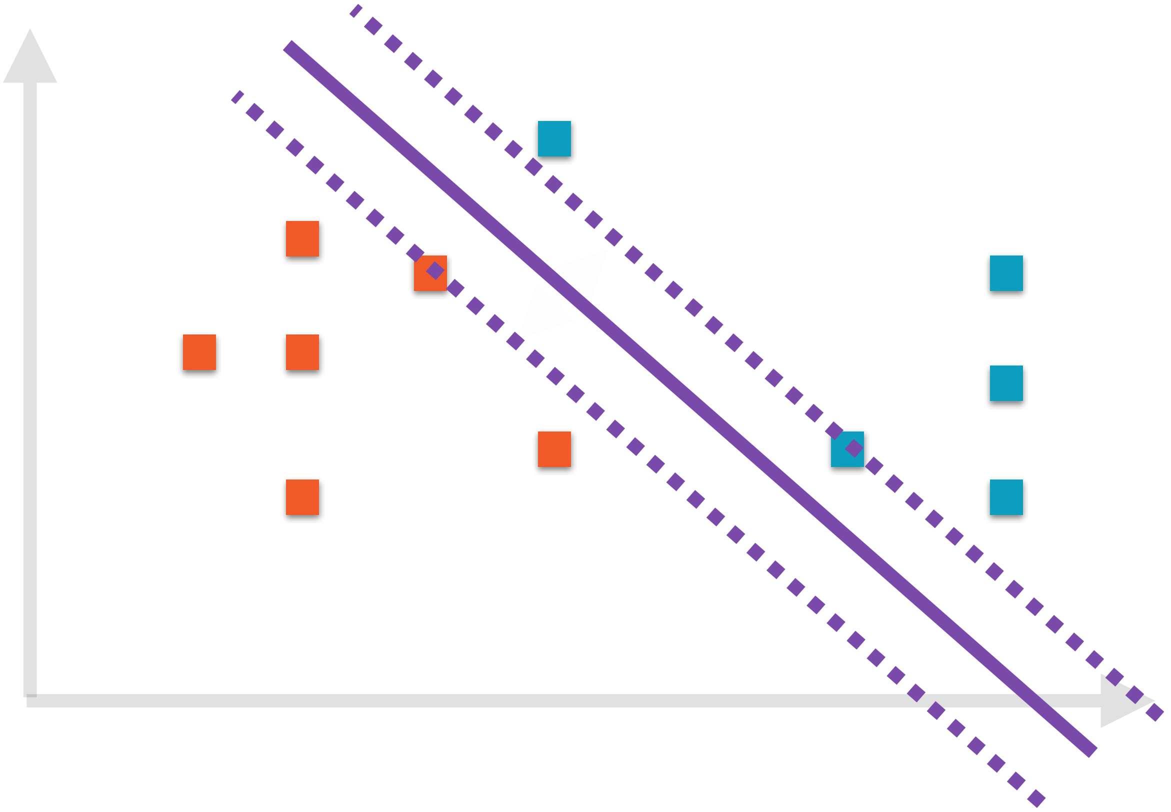
SVM Regression



Seek to maximize the number of points inside the margin

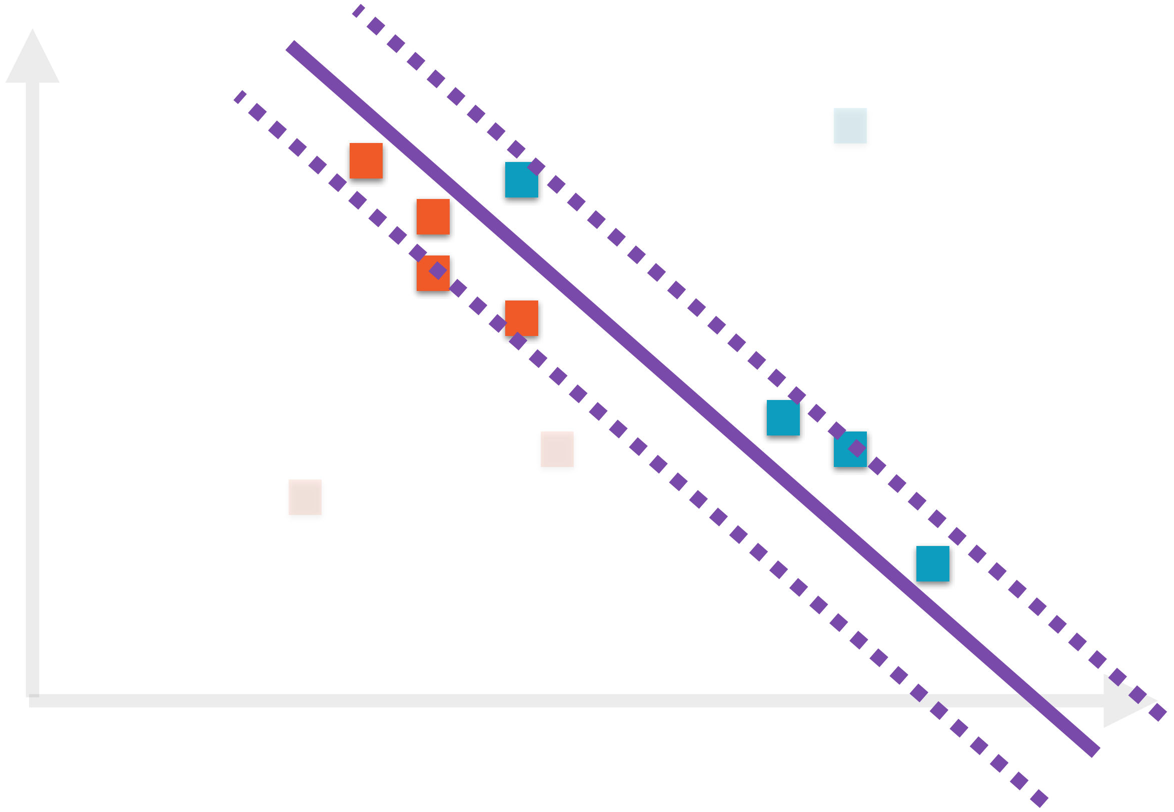
Similar, yet Different

SVM Classification



Points far from the margin are “good” (improve objective function value)

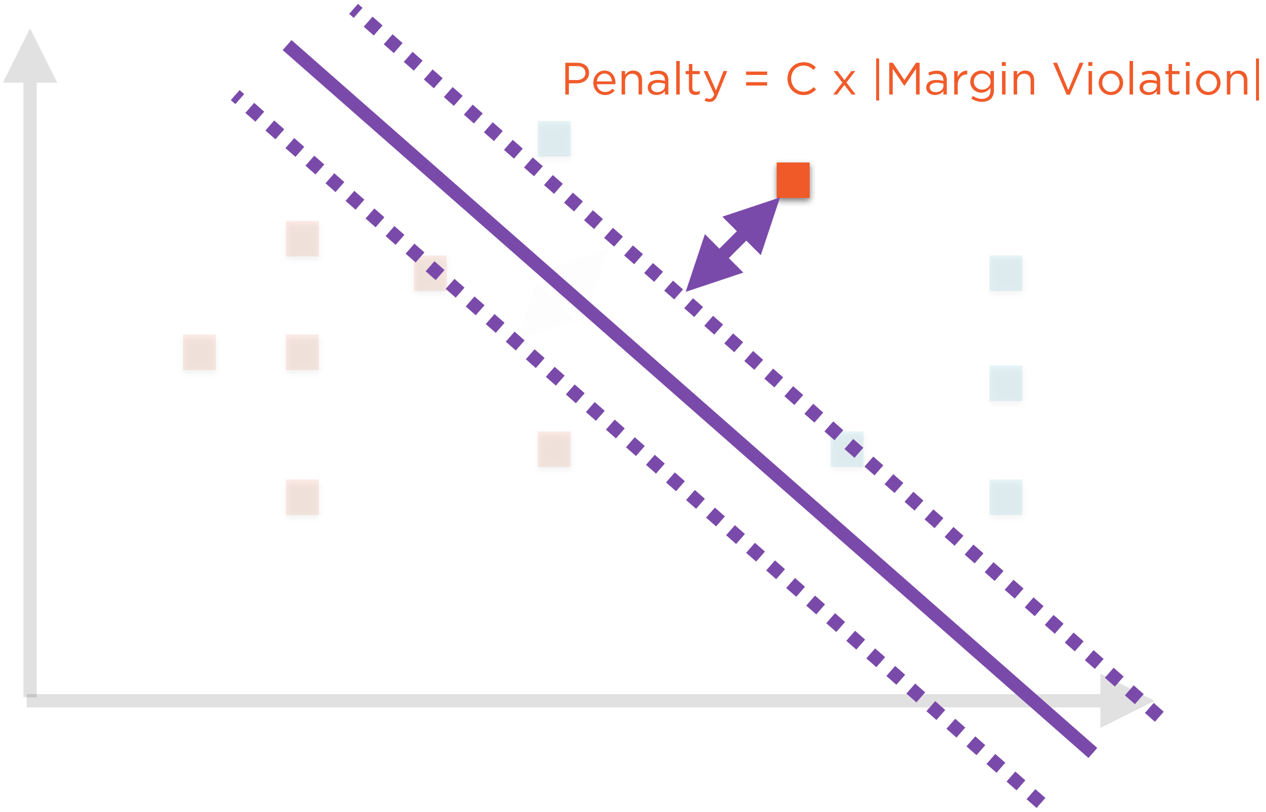
SVM Regression



Points far from the margin are “bad” (worsen objective function value)

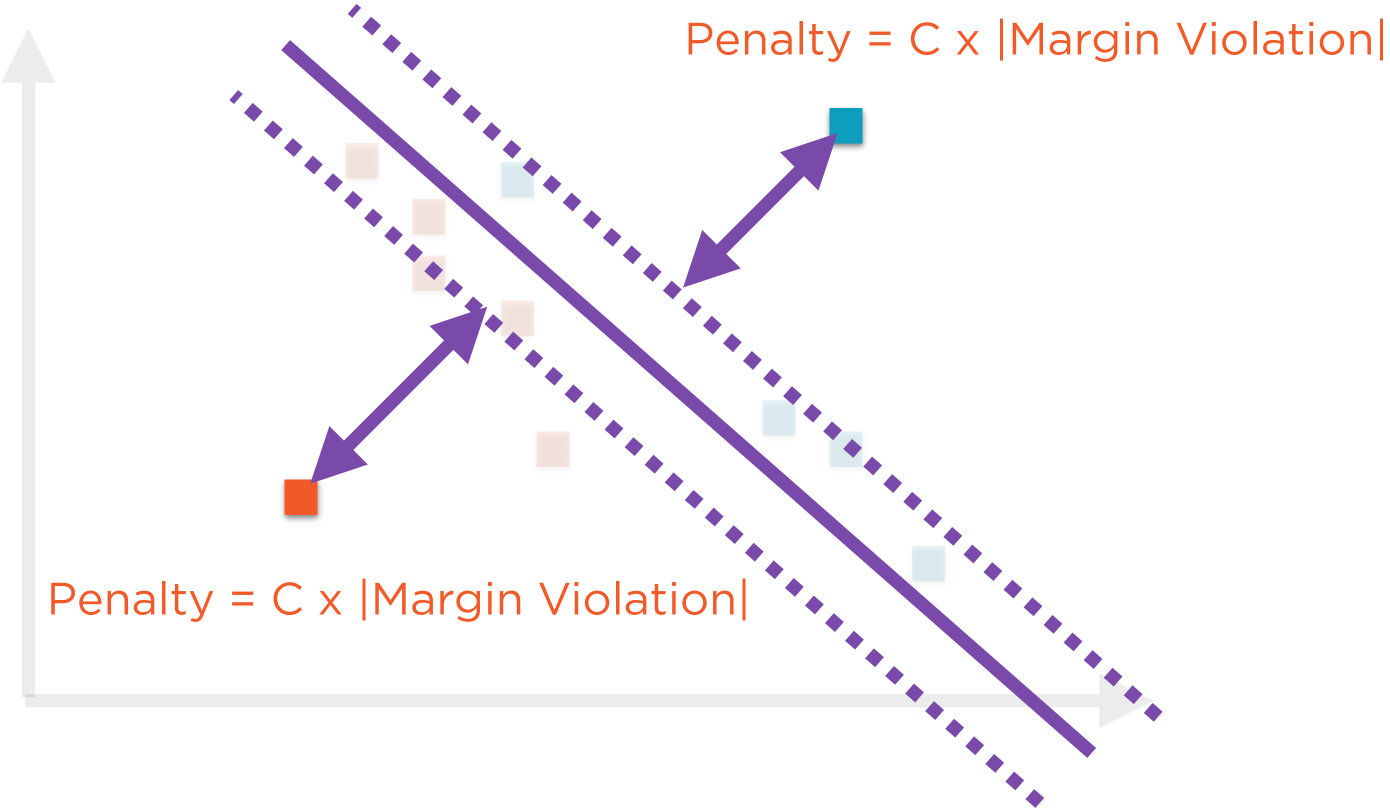
Similar, yet Different

SVM Classification



Outliers on “wrong” side of line are penalized

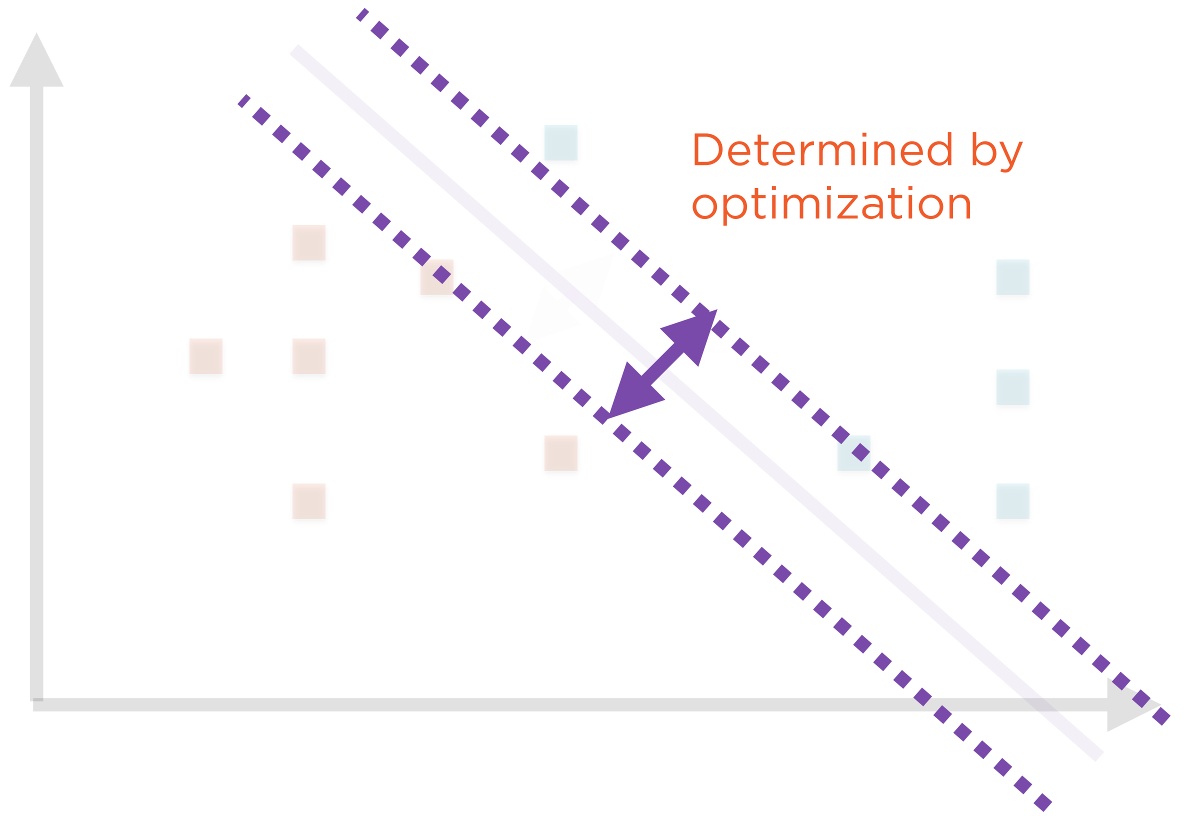
SVM Regression



Points far from the margin are penalized

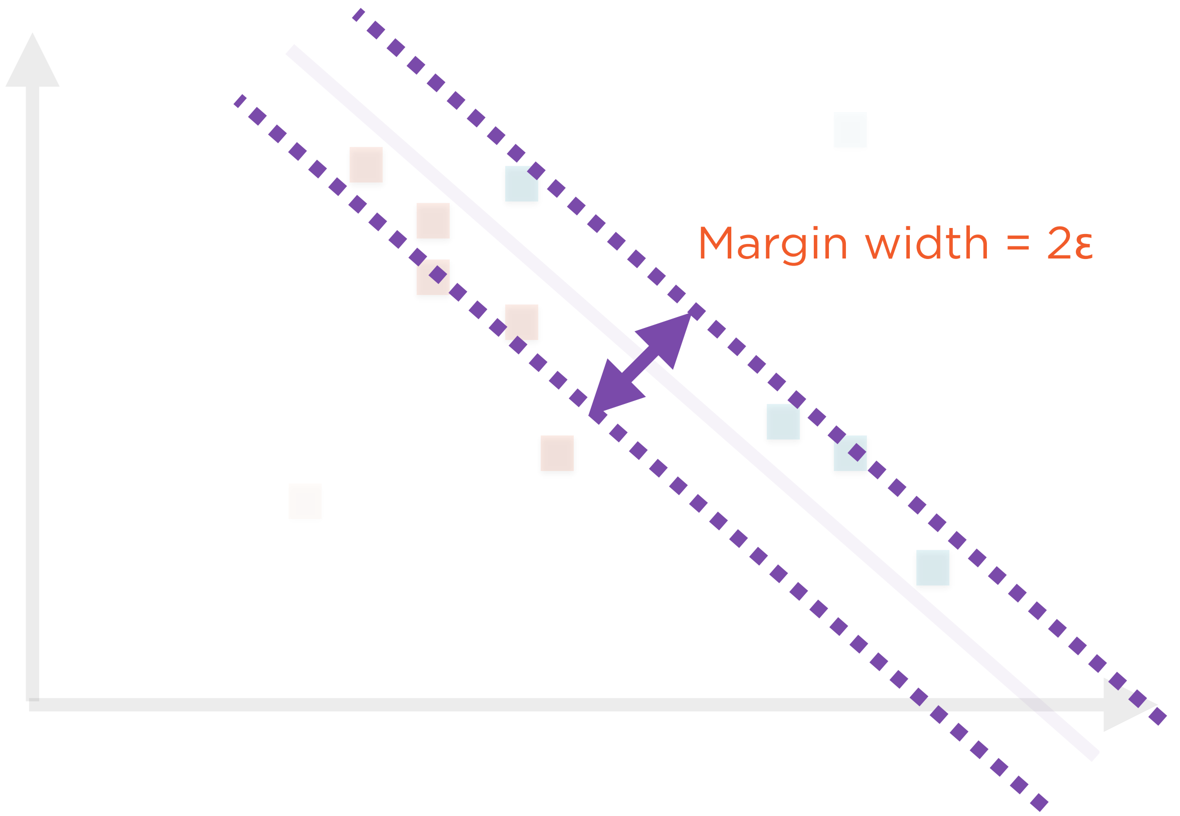
Similar, yet Different

SVM Classification



Width of margin found by optimizer (make as wide as possible)

SVM Regression



Width of margin specified in model (requires another hyperparameter ε)

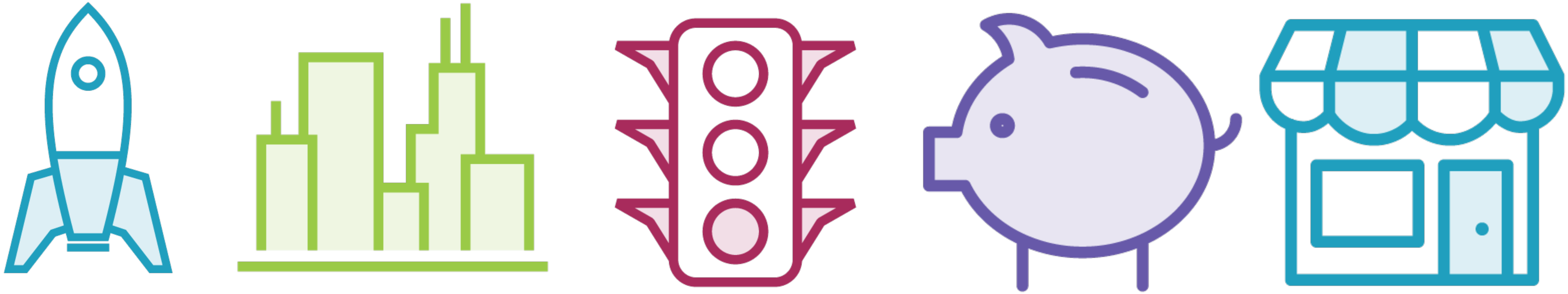
Demo

Support Vector regression

Nearest Neighbors Regression

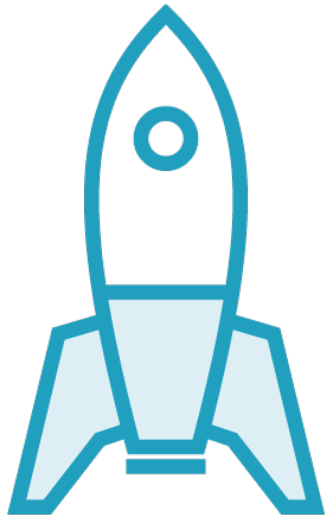
Nearest Neighbors Regression uses training data to find what is most similar to the current sample

Nearest Neighbors Regression



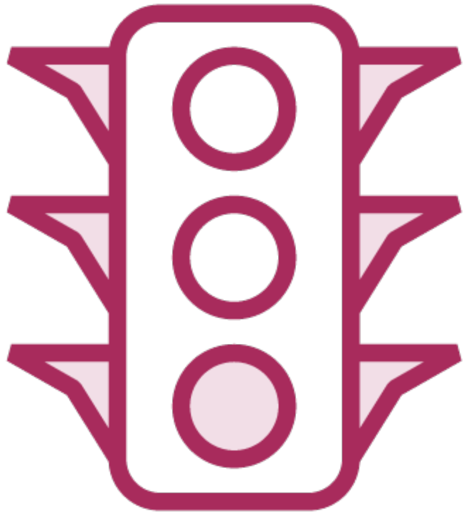
Uses the entire training dataset as a model

Nearest Neighbors Regression

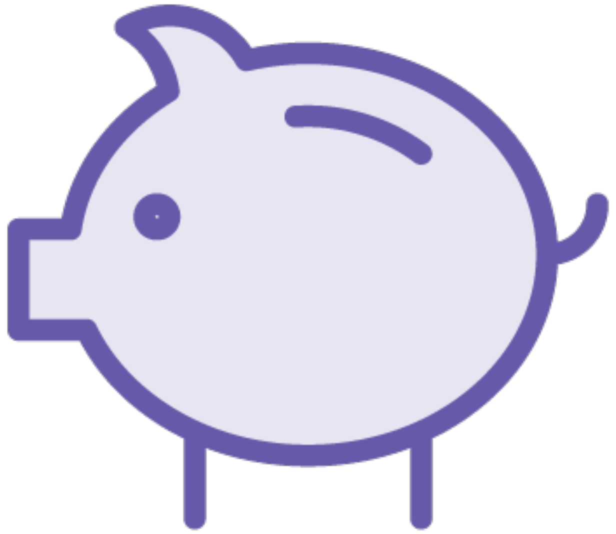


Rocket

Buildings



Signal

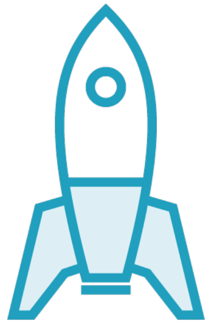


Pig

Shop

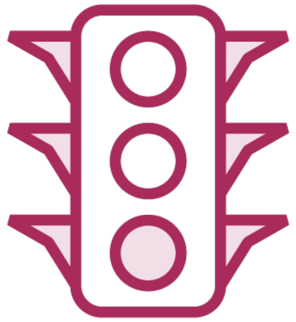
Each element in training data has an associated y value

Nearest Neighbors Regression

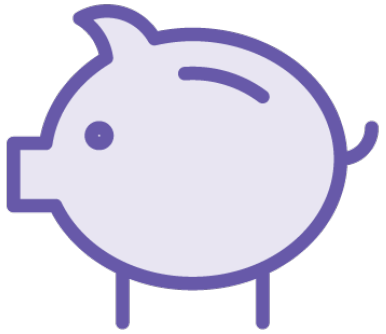


Rocket

Buildings



Signal



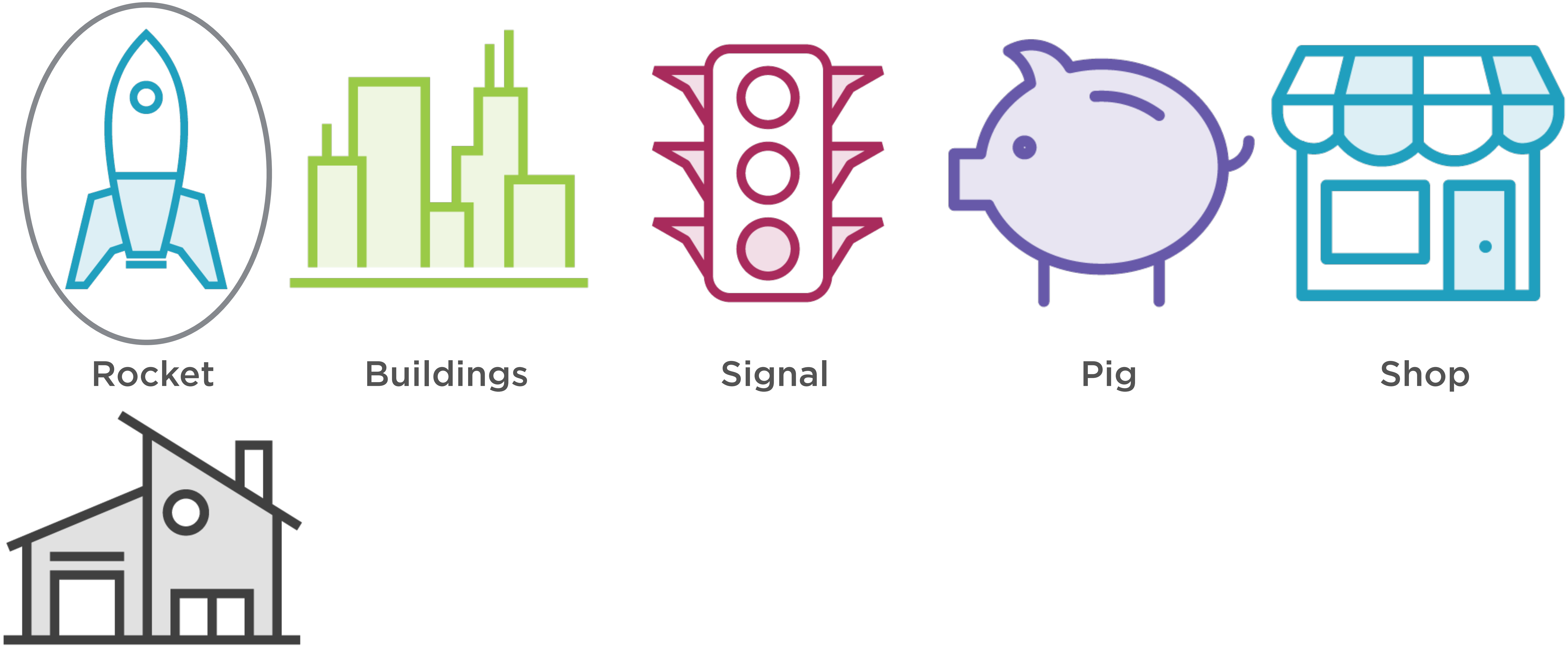
Pig

Shop

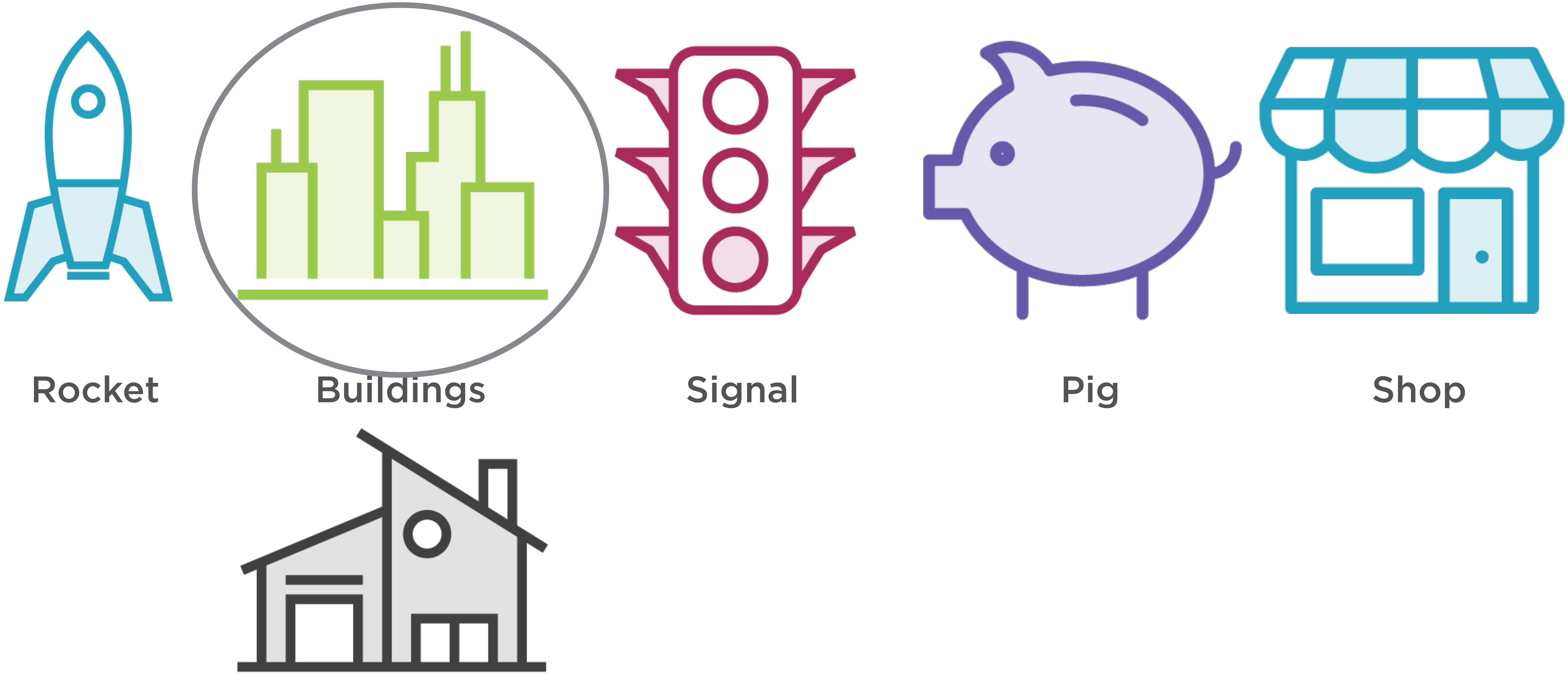
Predictions for a new sample involves figuring out which element in the training data it is similar to

The nearest neighbor

Nearest Neighbors Regression



Nearest Neighbors Regression



Rocket

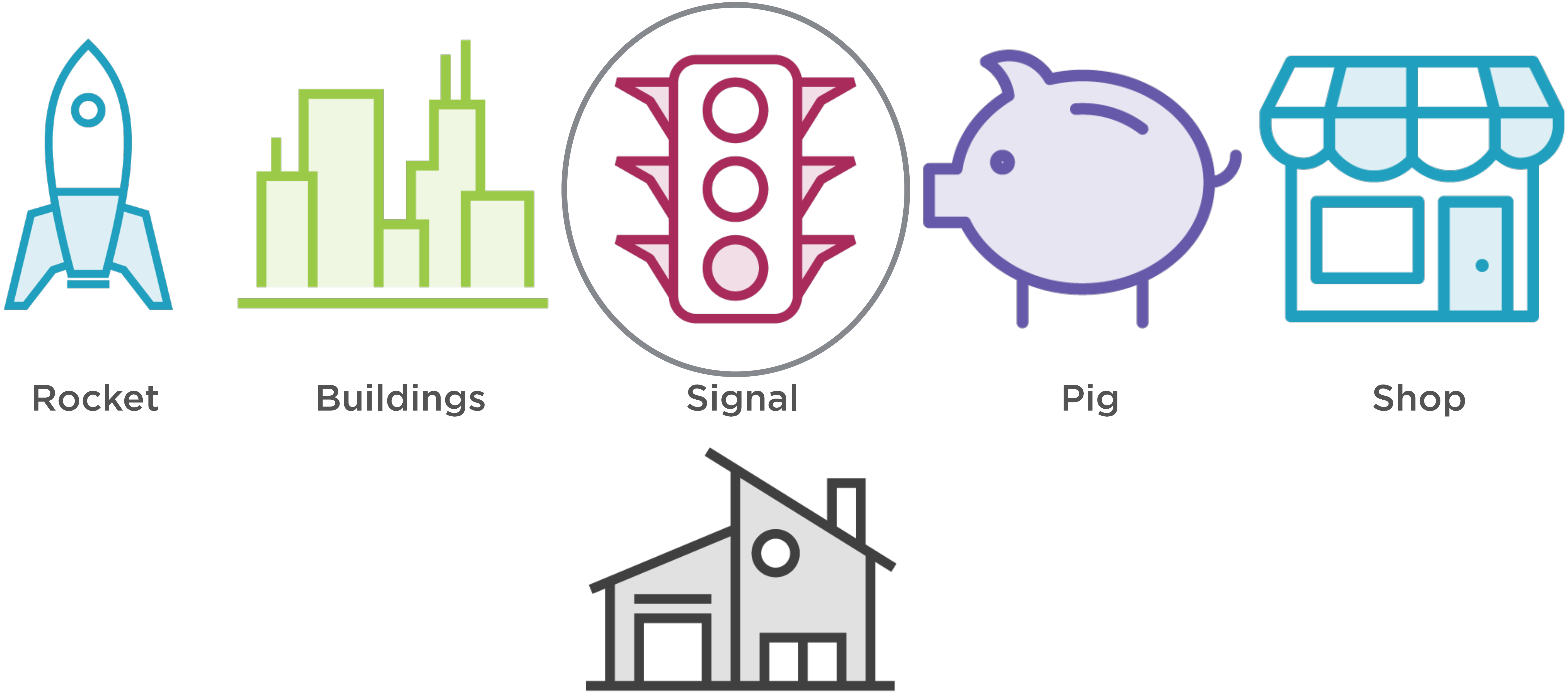
Buildings

Signal

Pig

Shop

Nearest Neighbors Regression



Rocket

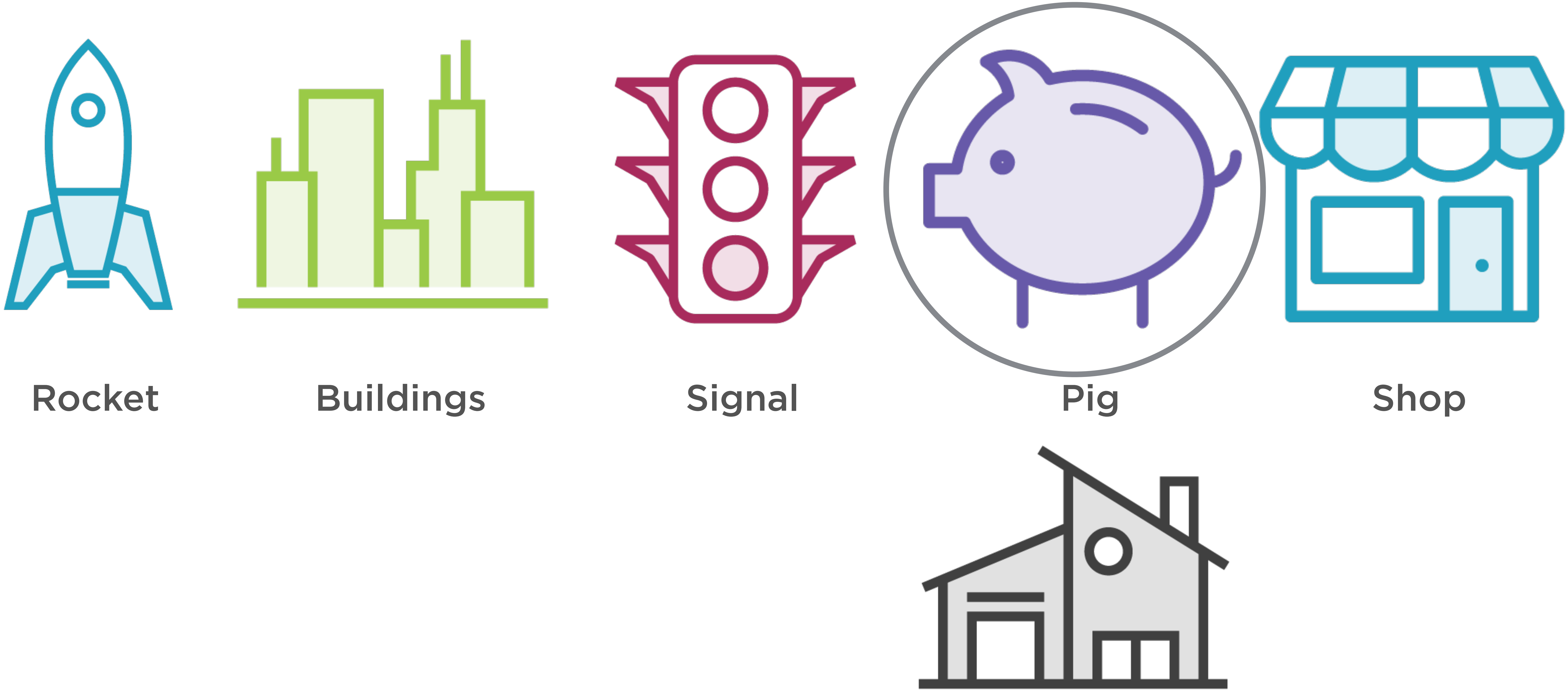
Buildings

Signal

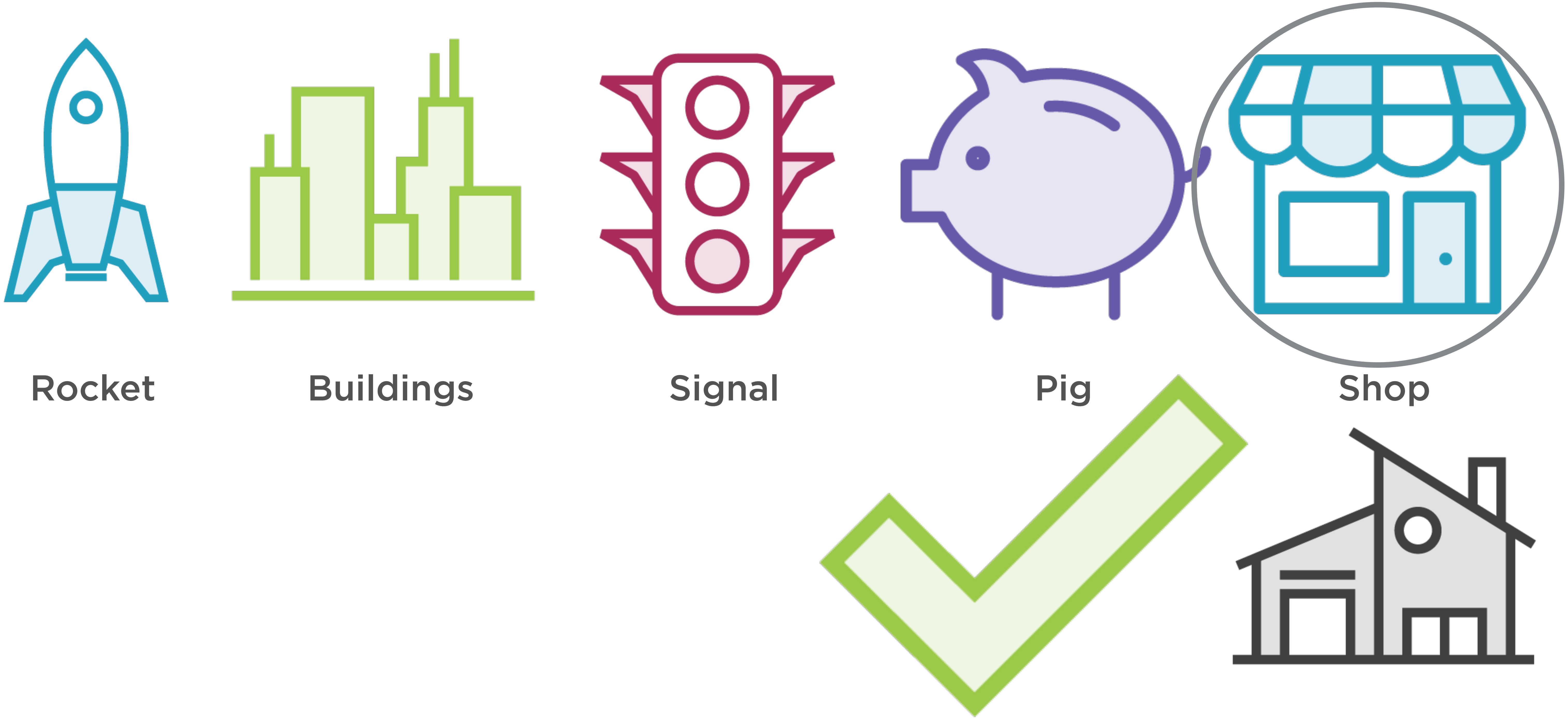
Pig

Shop

Nearest Neighbors Regression



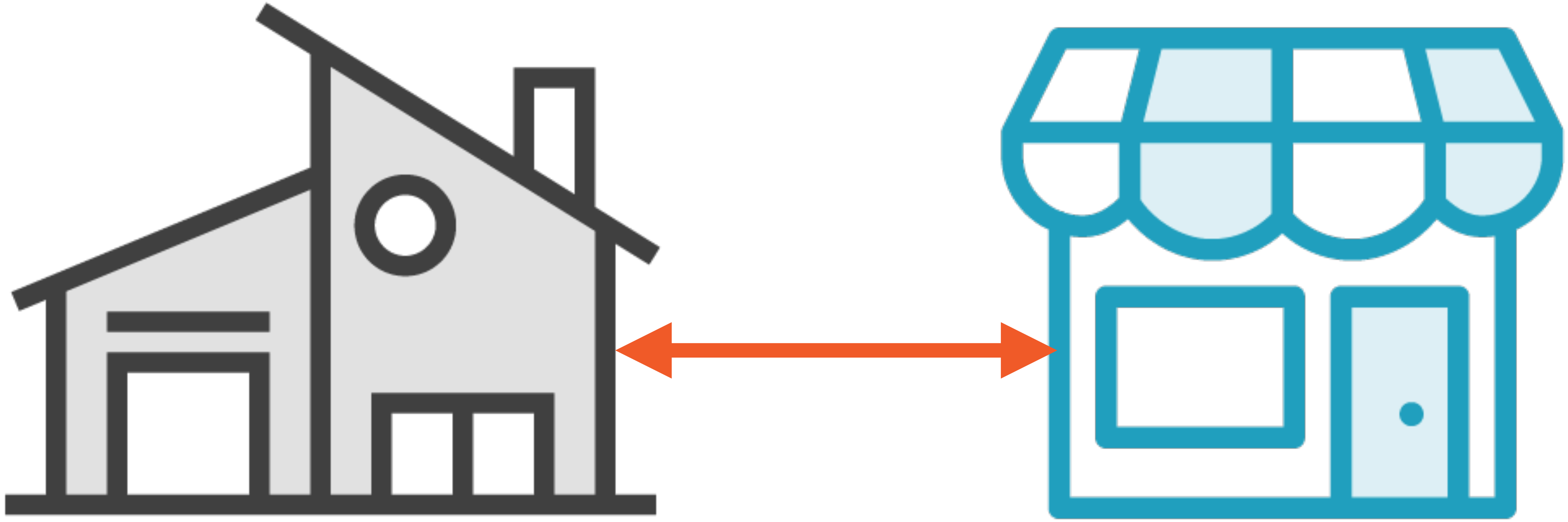
Nearest Neighbors Regression



Nearest Neighbors Regression

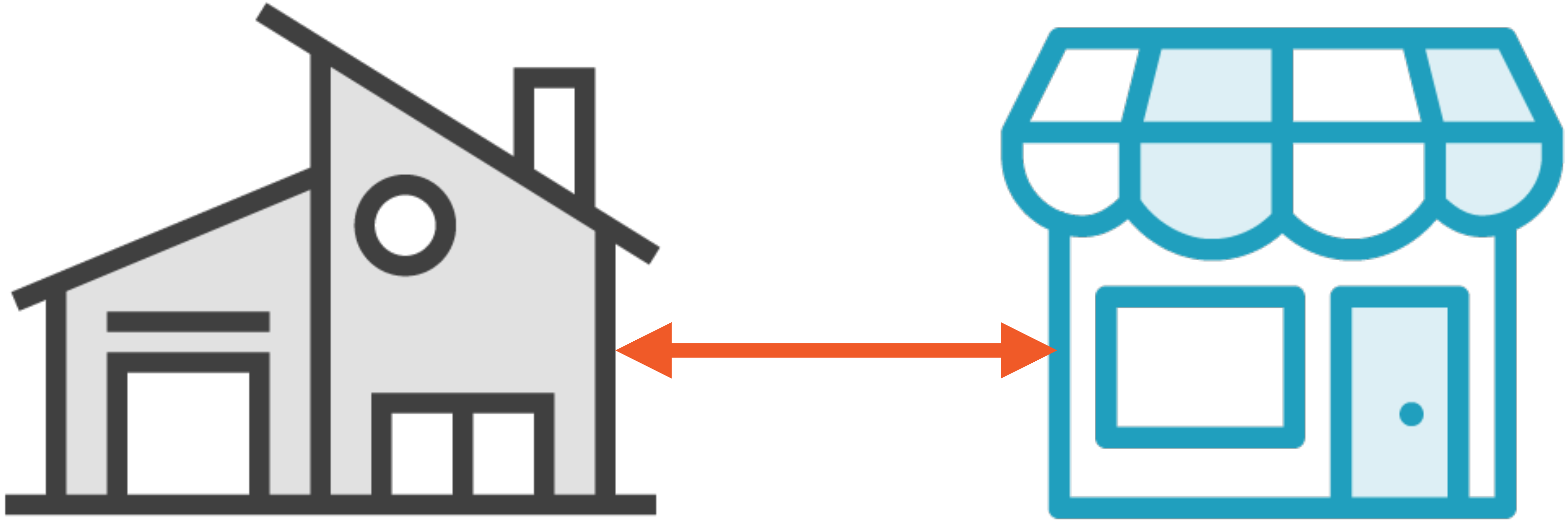
How do we calculate neighbors of a sample?

Nearest Neighbors Regression



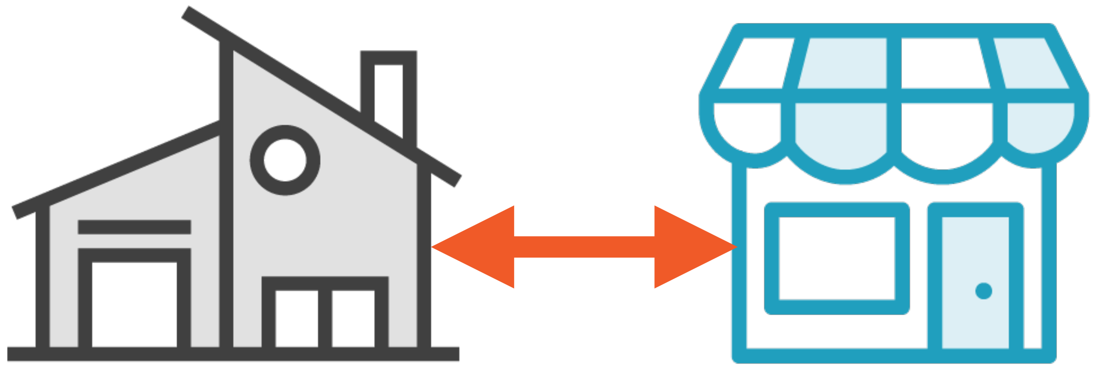
Distance measures

Nearest Neighbors Regression



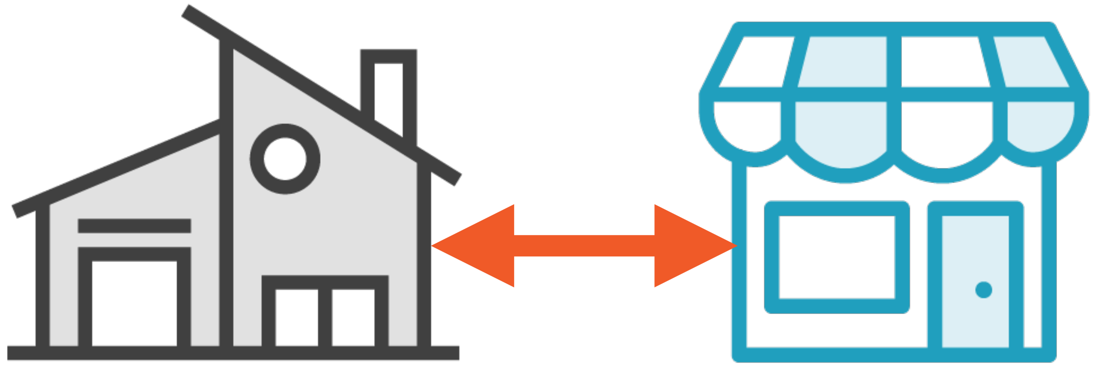
Euclidean distance, **Hamming** distance, **Manhattan** distance

Distance Measures



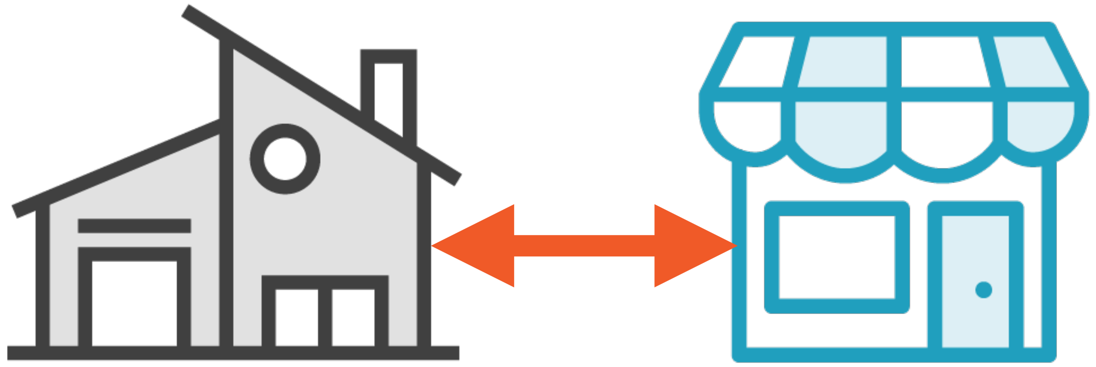
```
EuclideanDistance(x, xi) = sqrt(sum((xj – xij)^2))
```

Distance Measures



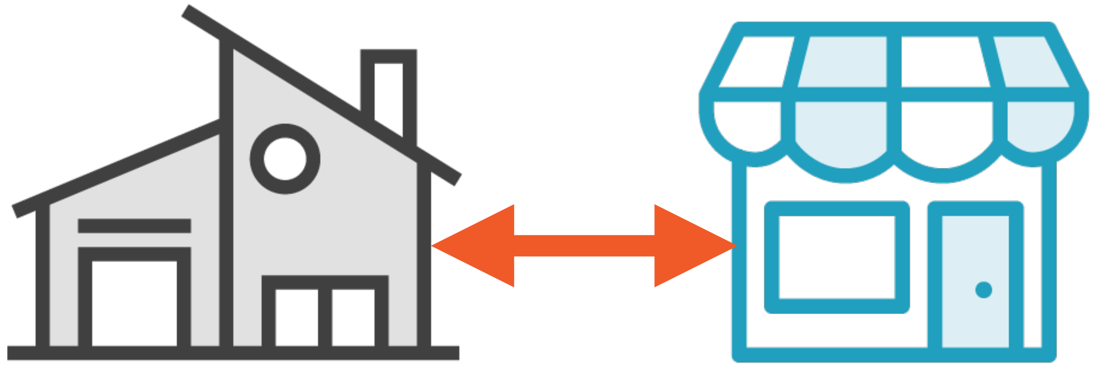
`EuclideanDistance(x, xi) = sqrt(sum((xj – xij)^2))`

Distance Measures



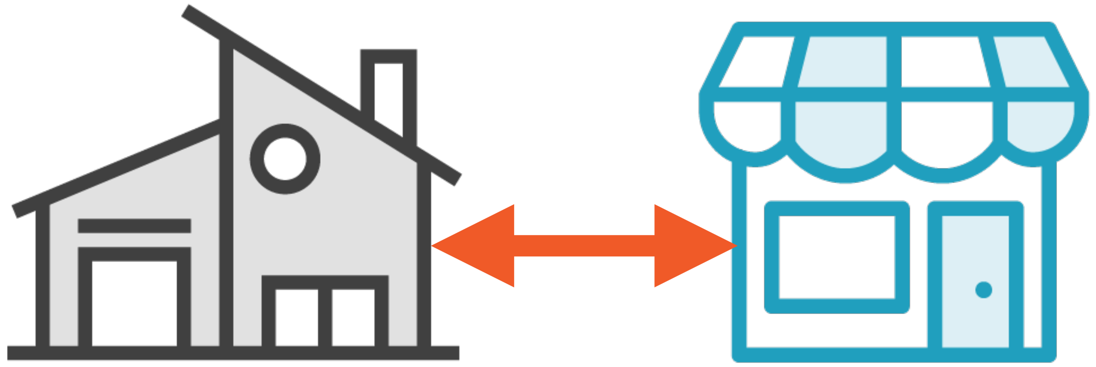
`EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))`

Distance Measures



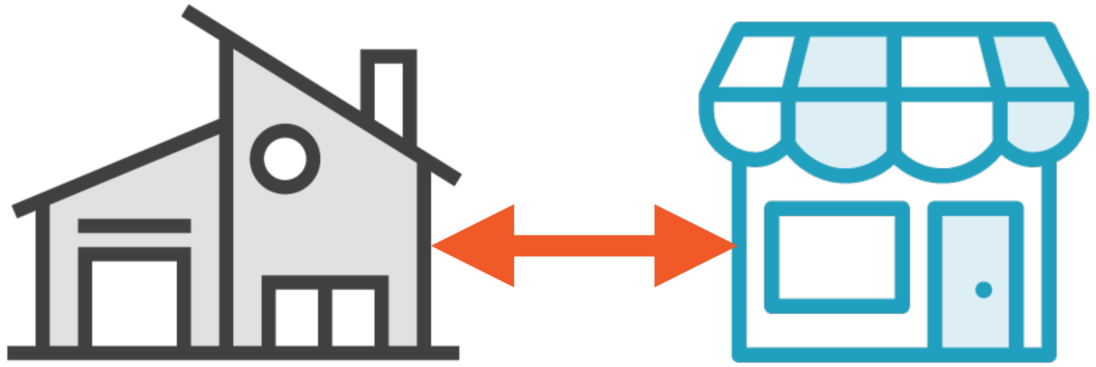
```
EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))
```

Distance Measures



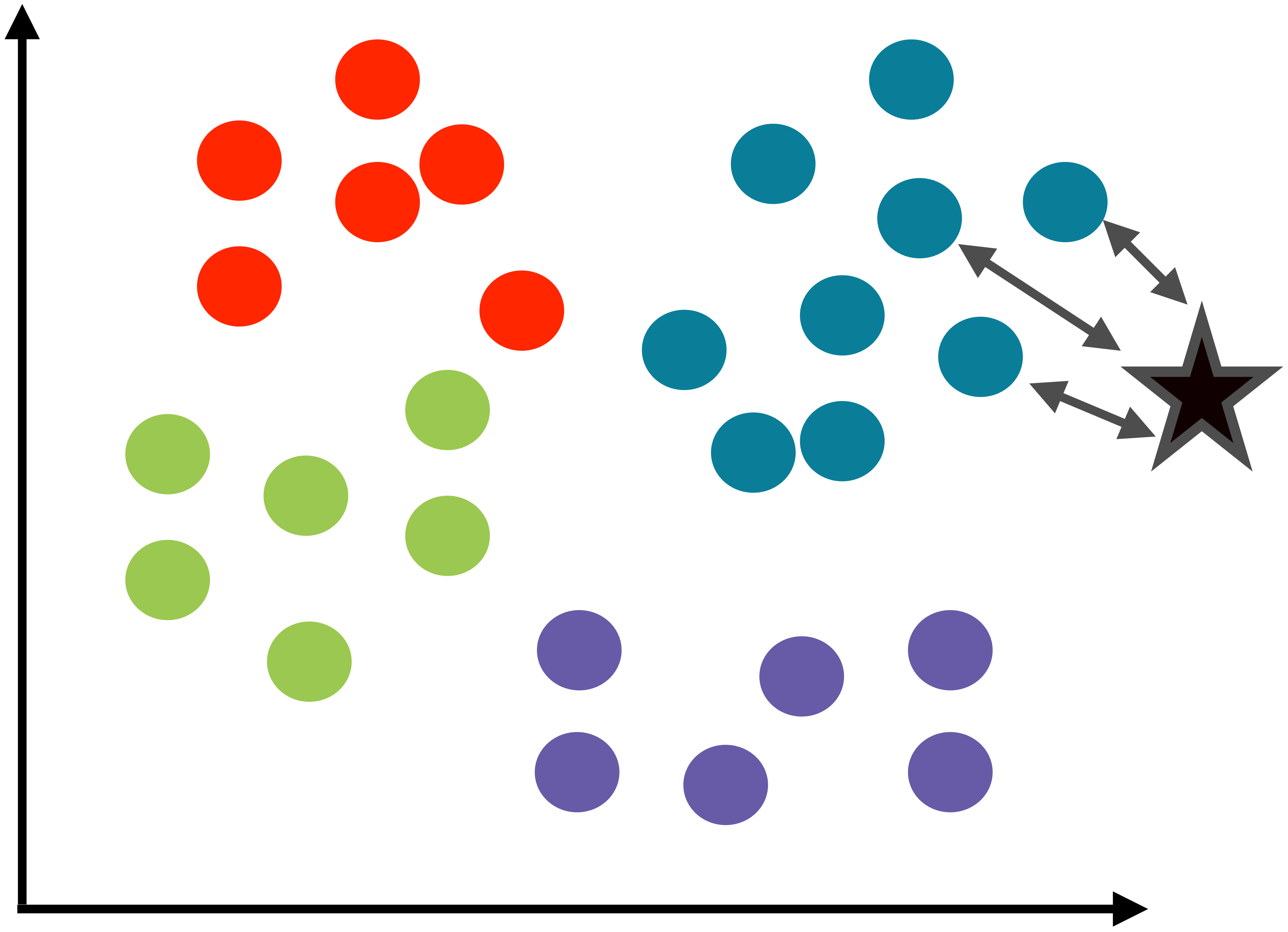
```
EuclideanDistance(x, xi) = sqrt(sum((xj - xij)^2))
```

Distance Measures

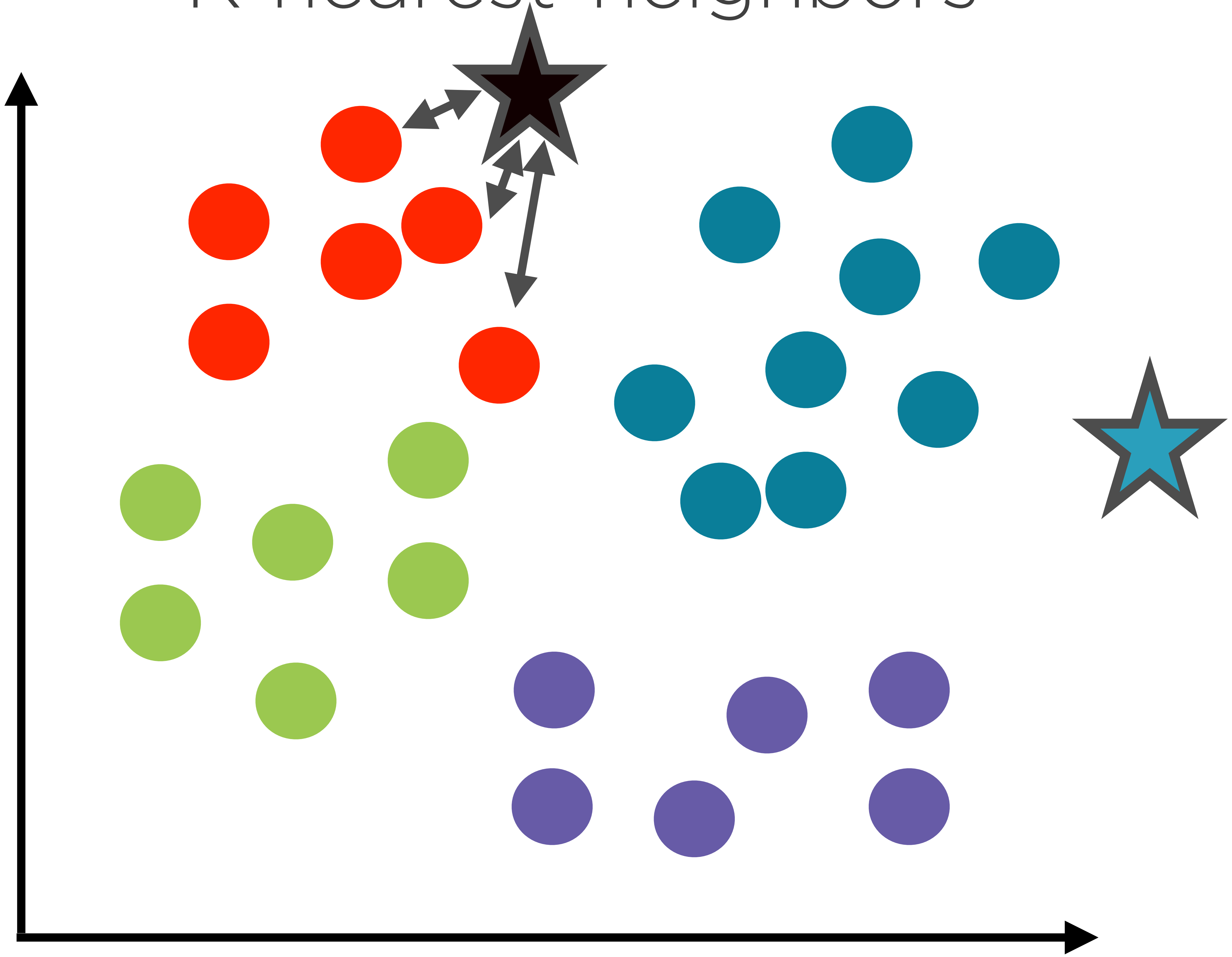


```
EuclideanDistance(x, xi) = sqrt(sum((xj – xij)^2))
```

K-nearest-neighbors



K-nearest-neighbors



K-nearest-neighbors

Nearest Neighbors Regression

K-nearest-neighbors Regression

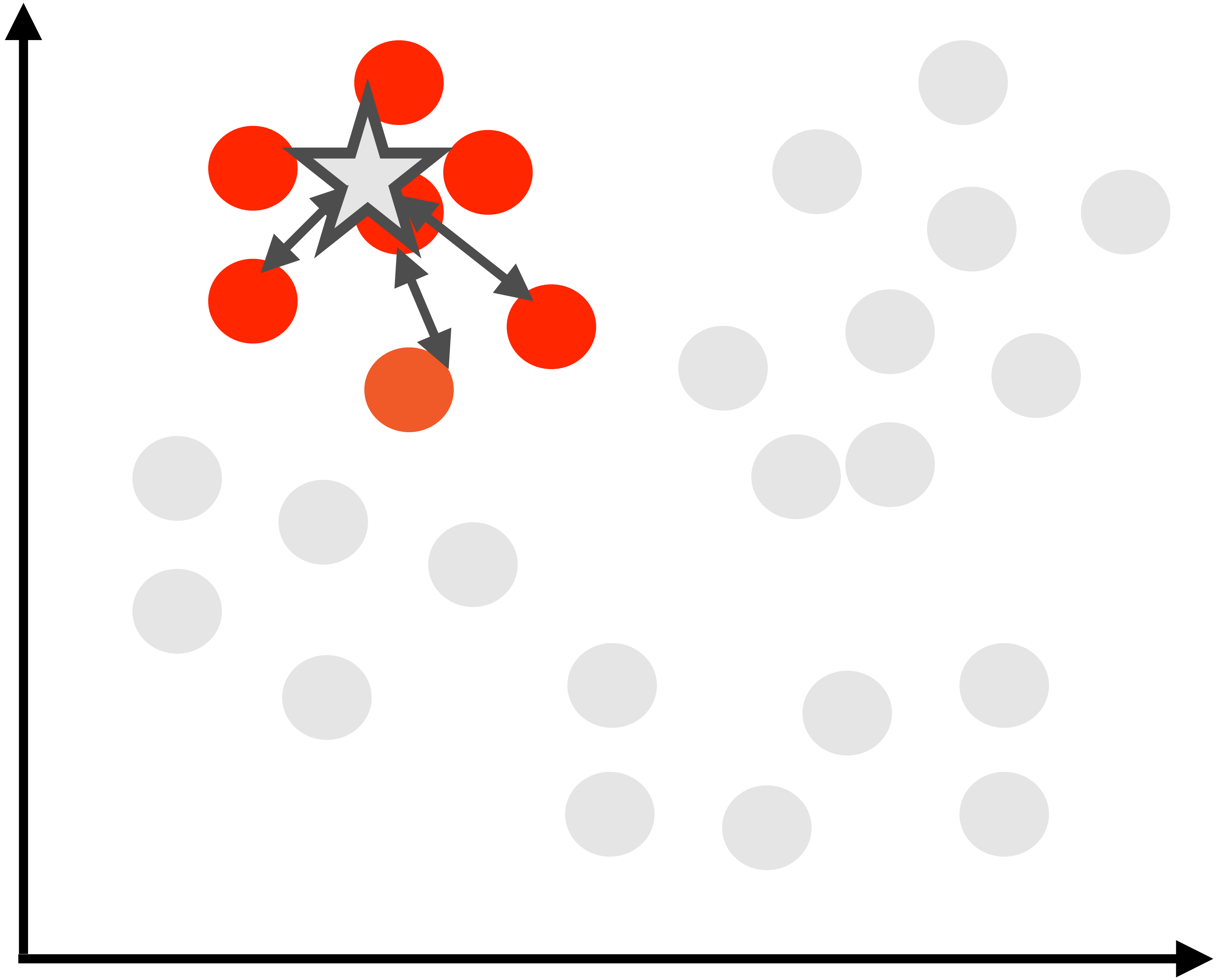
Radius Neighbors Regression

Nearest Neighbors Regression

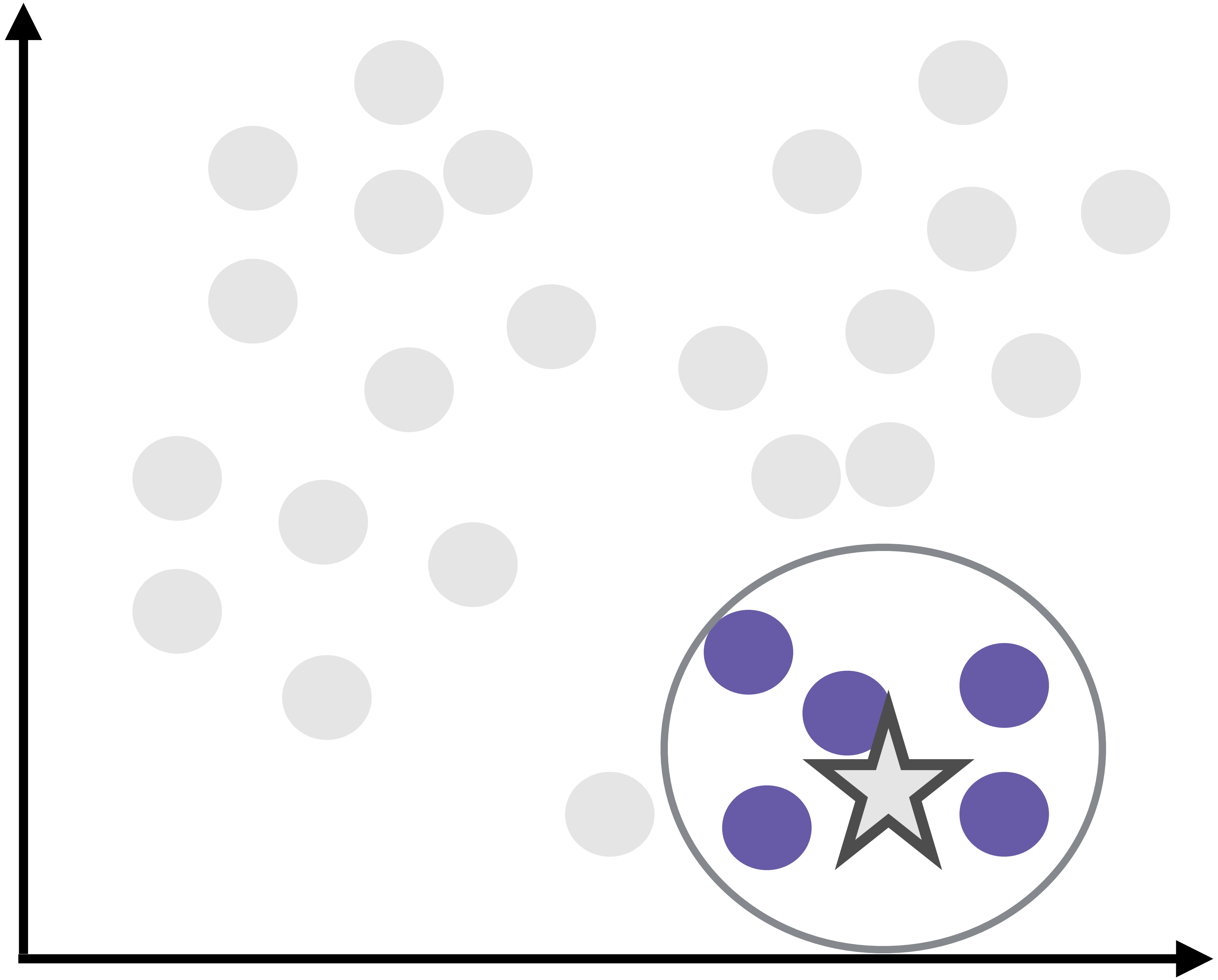
Average y-value of K Nearest Neighbors

Average y-values of Neighbors Within Radius

K-nearest-neighbors



Radius Neighbors

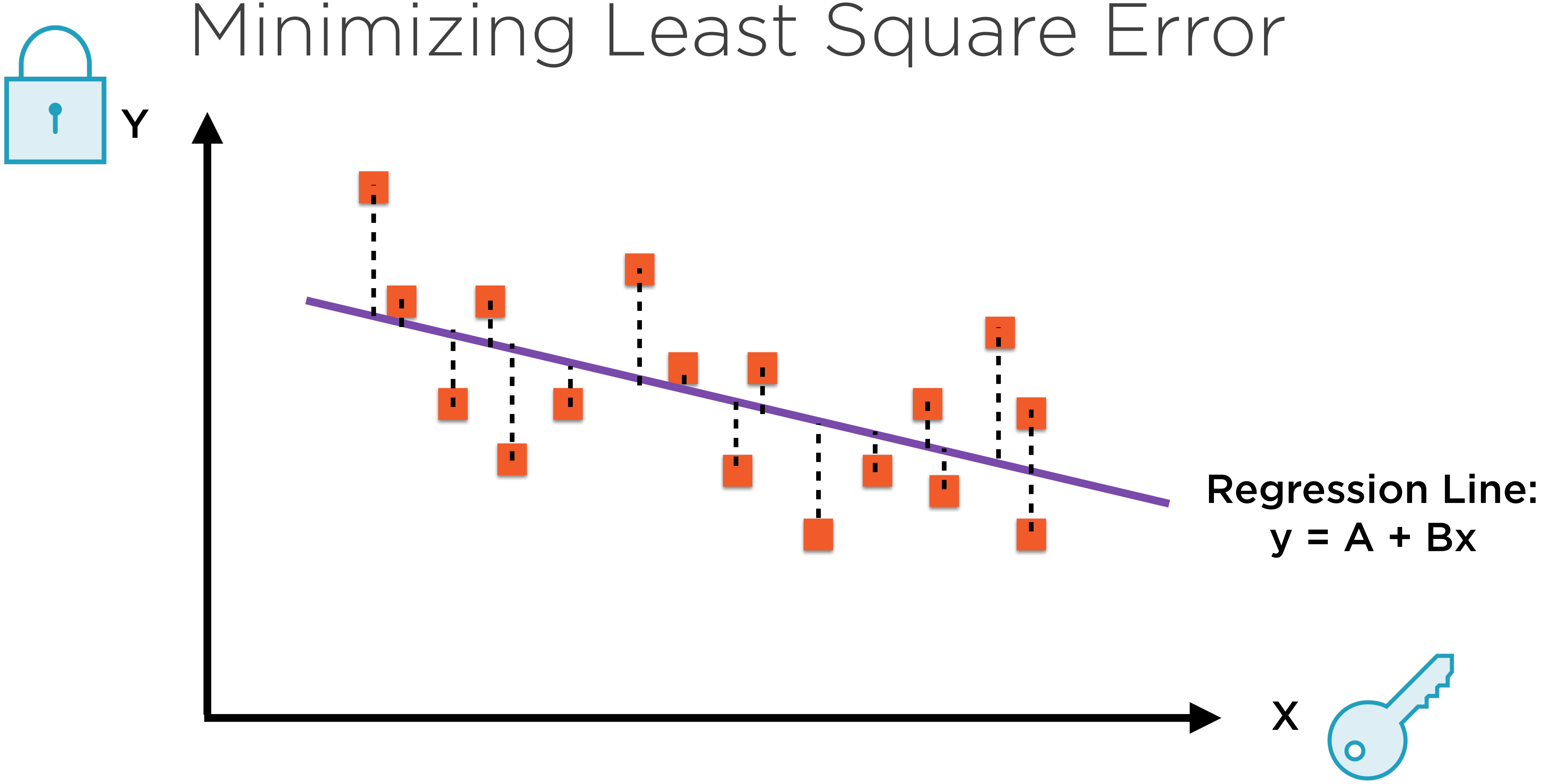


Demo

K-nearest-neighbors regression

SGD Regression

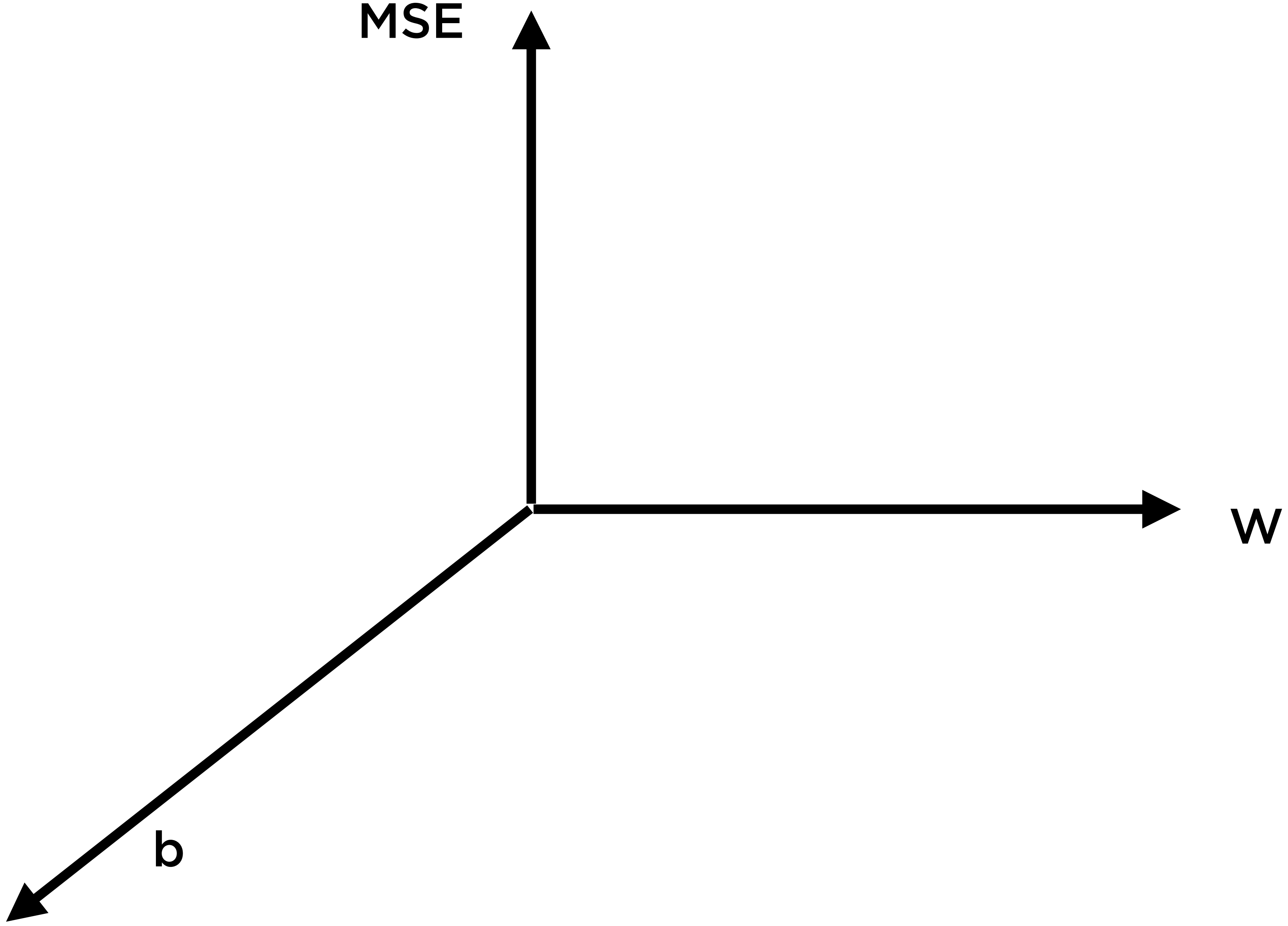
Minimizing Least Square Error



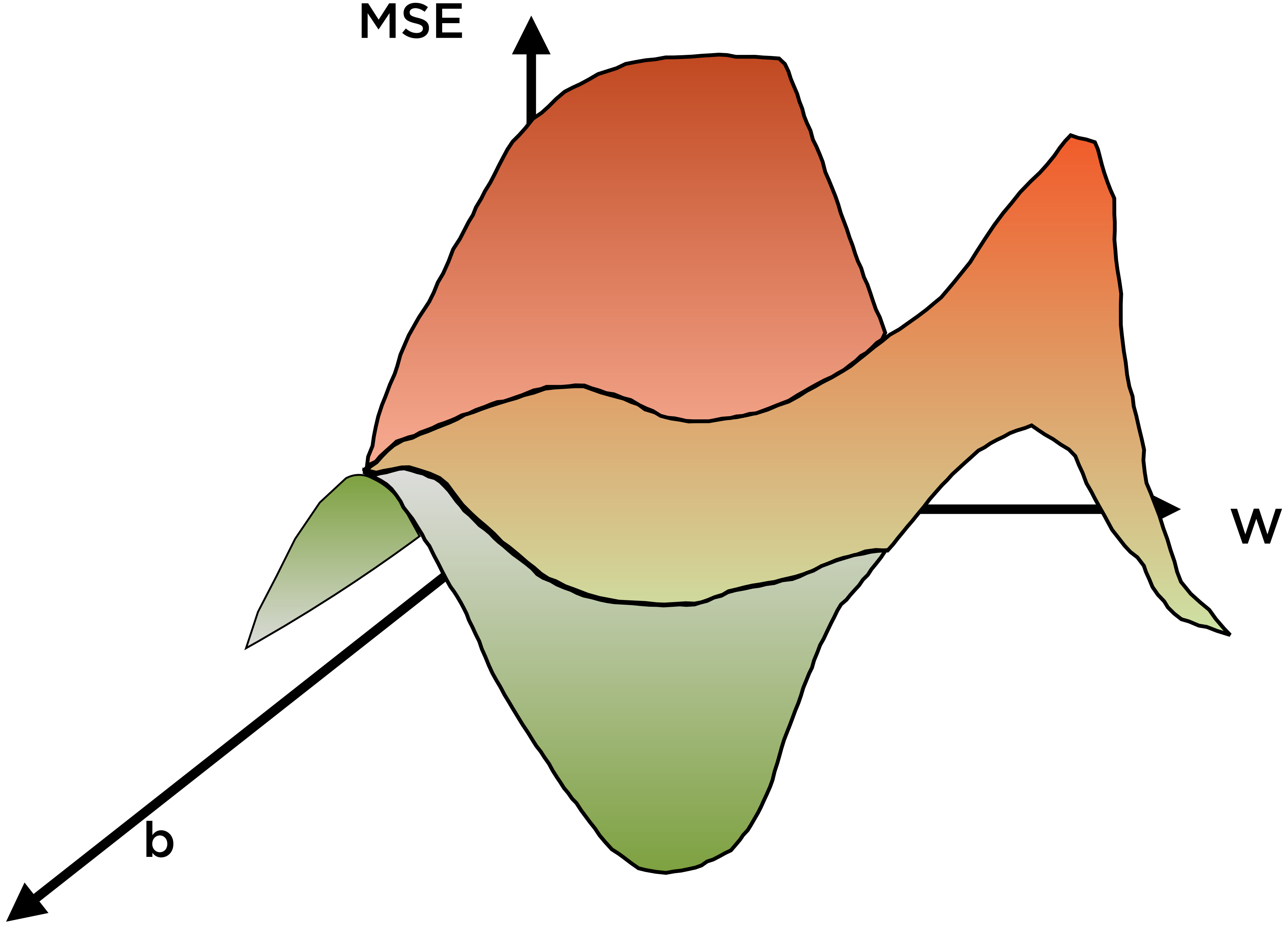
Regression Line:
$y = A + Bx$

The “best fit” line is called the regression line

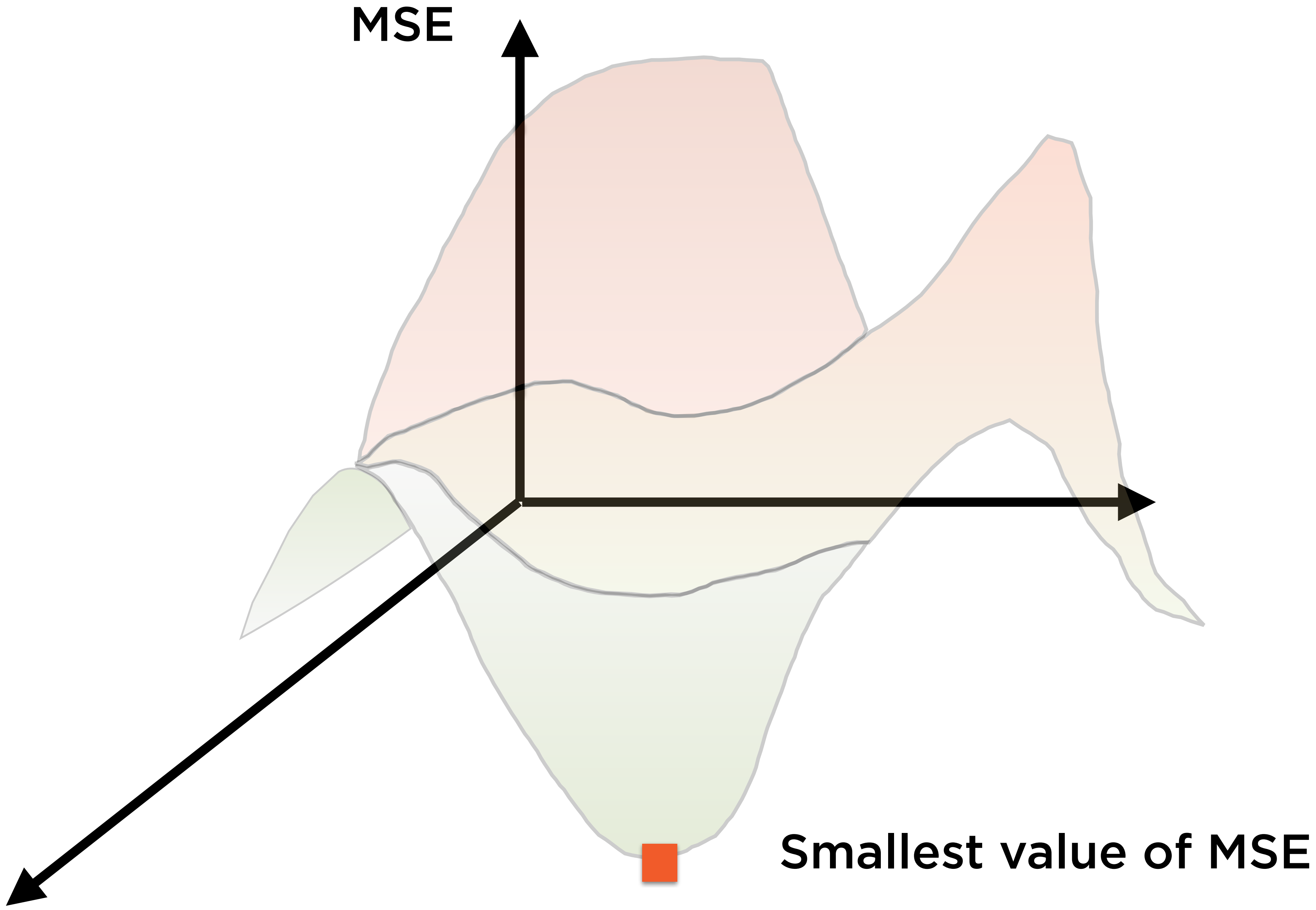
Minimizing MSE



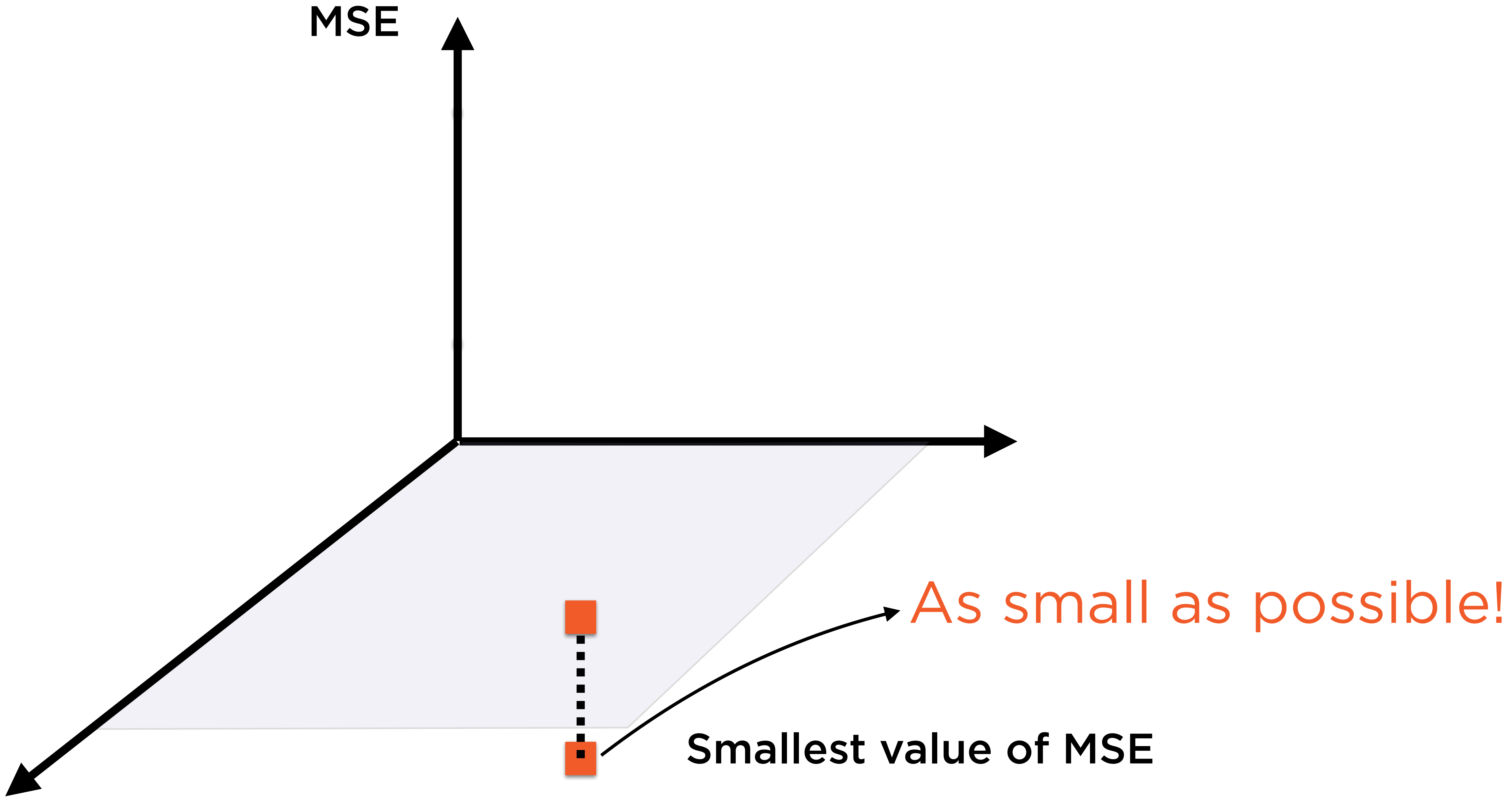
Minimizing MSE



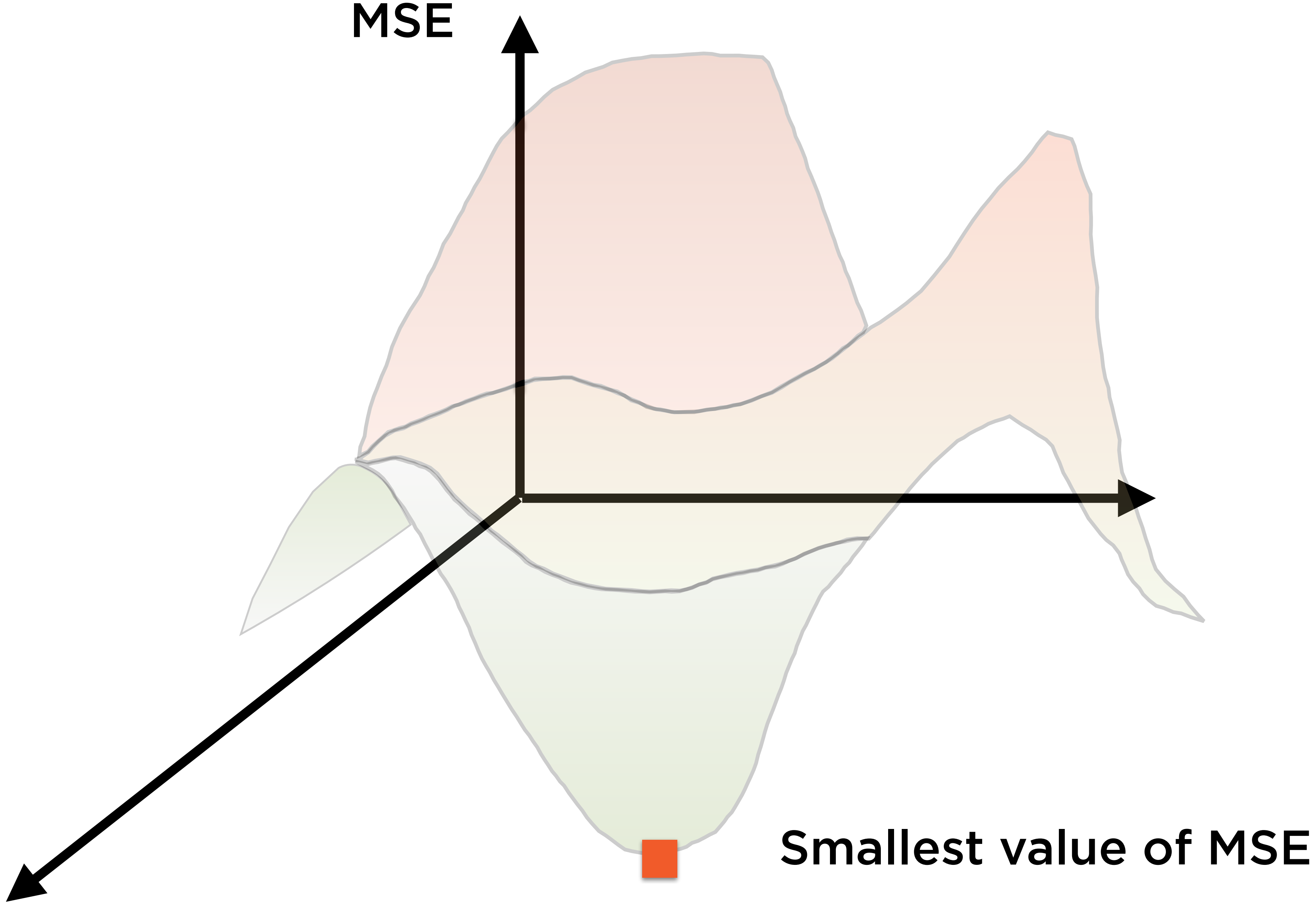
Minimizing MSE



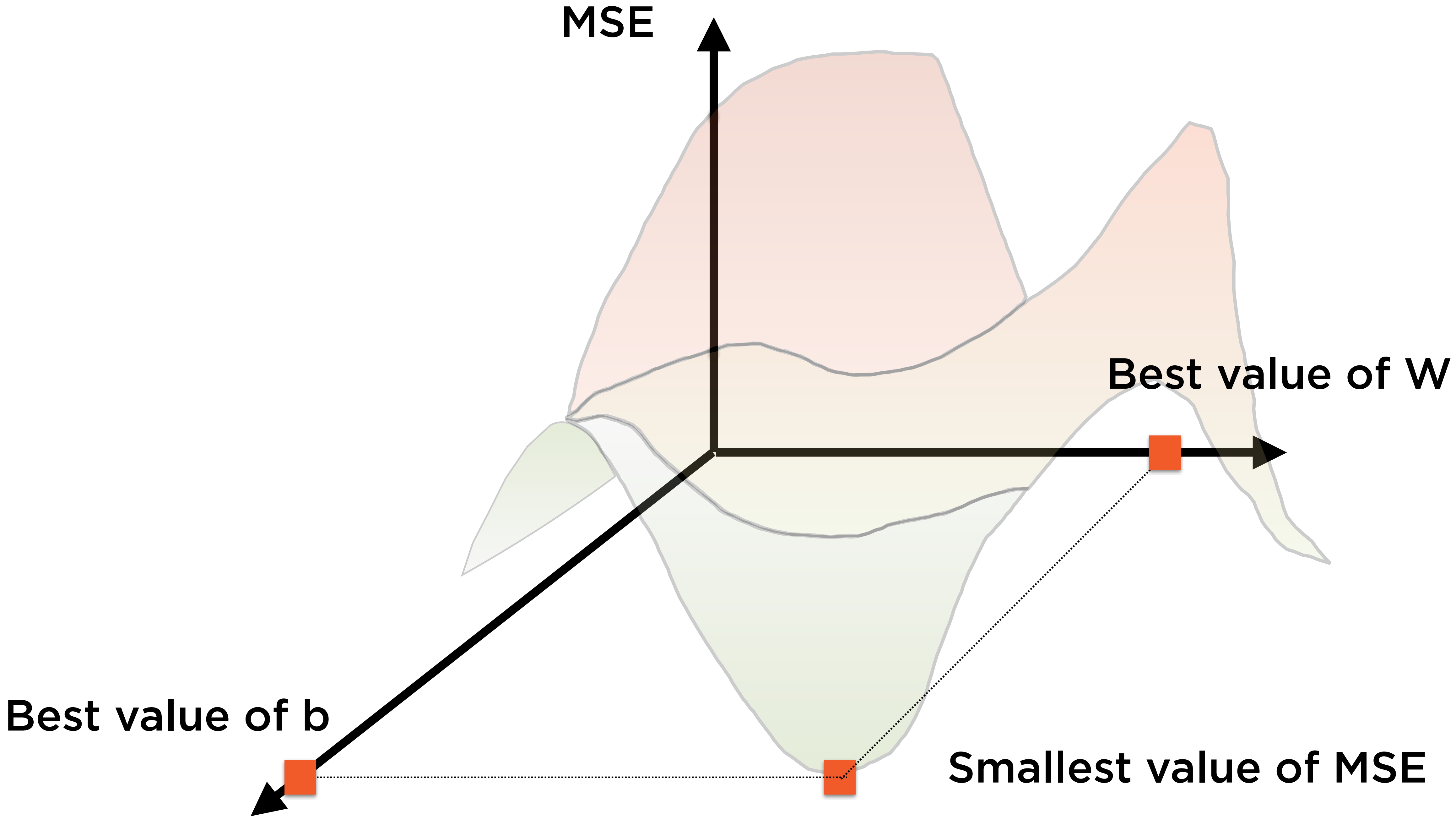
Minimizing MSE



Minimizing MSE

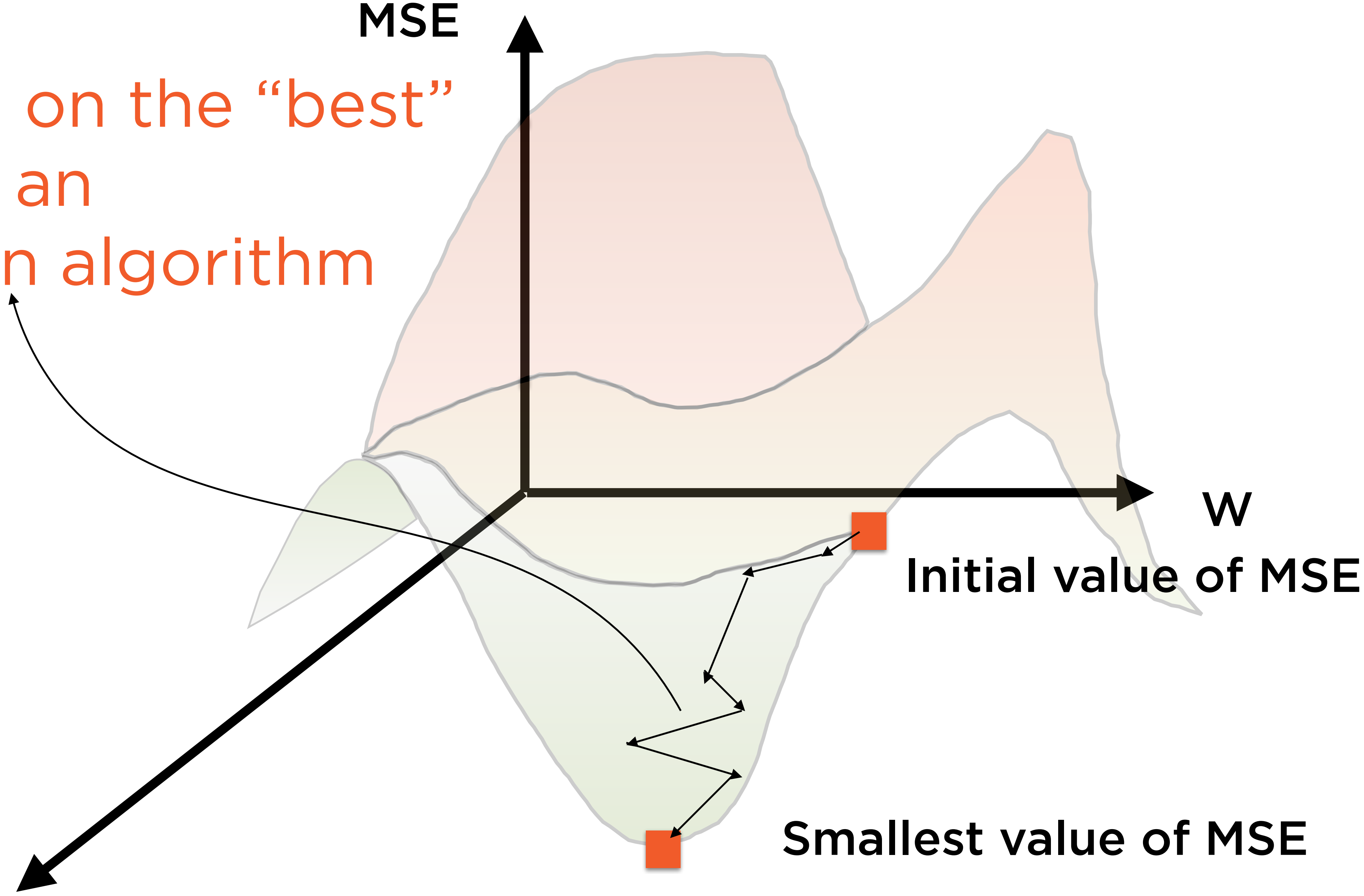


Minimizing MSE

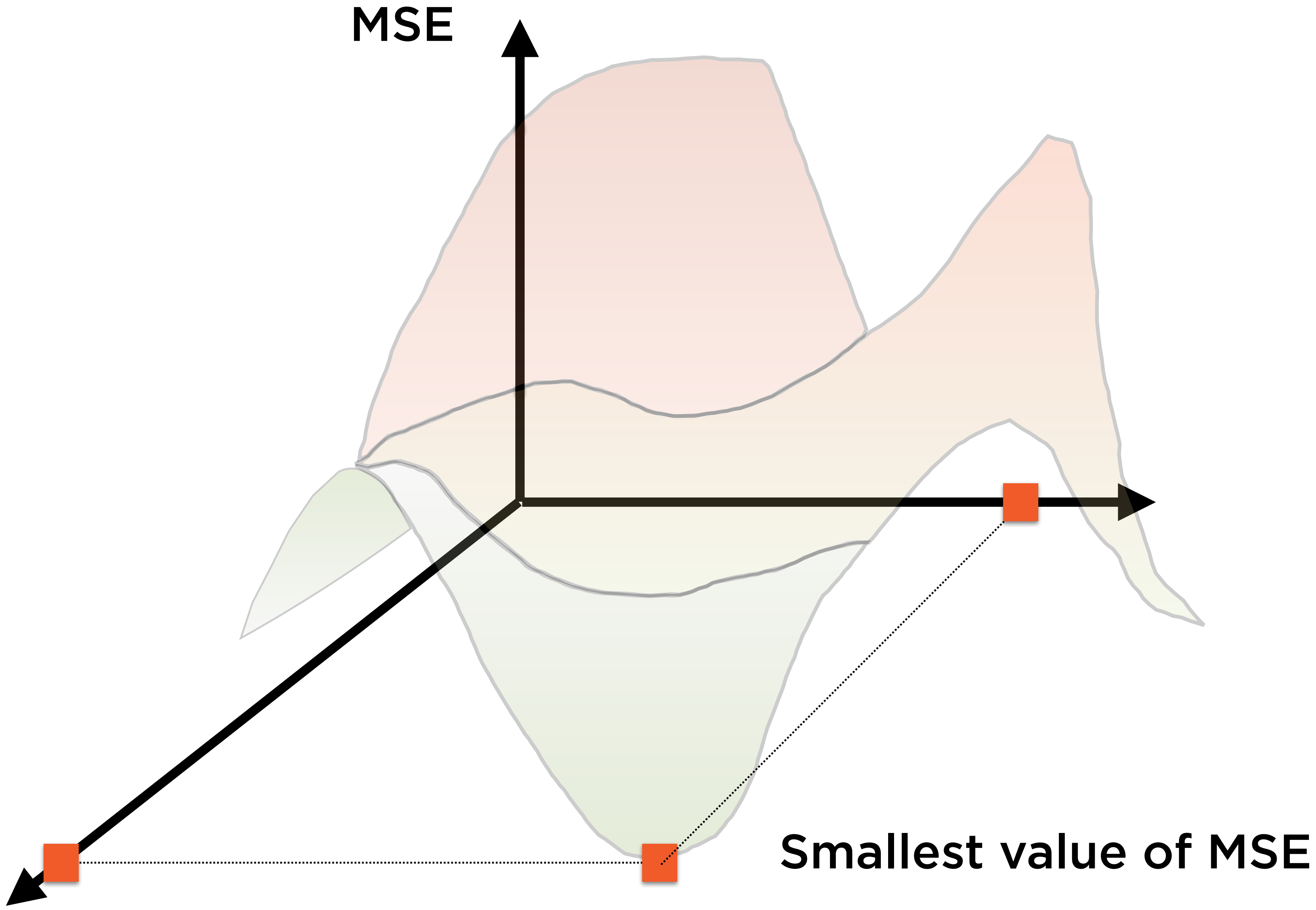


“Gradient Descent”

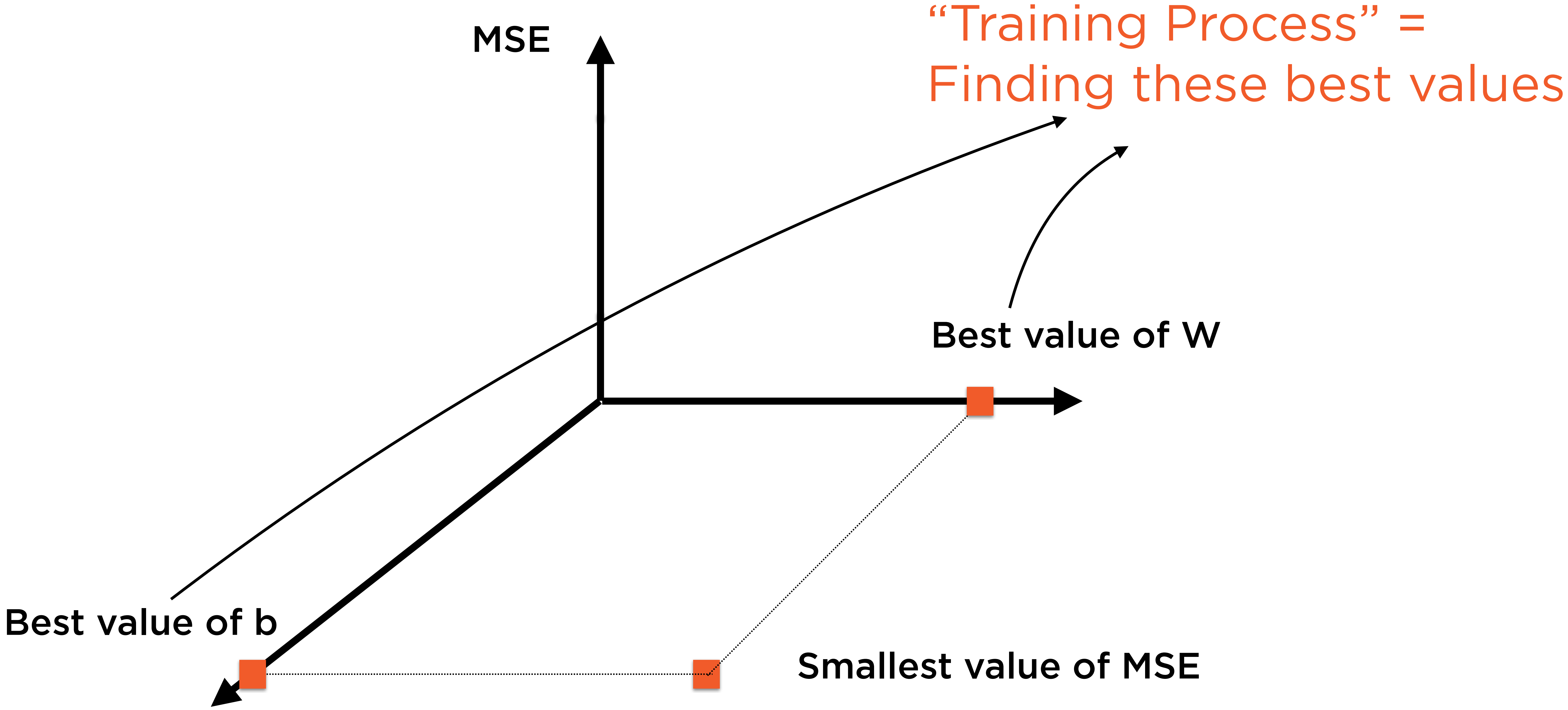
Converging on the “best” value using an optimization algorithm



Minimizing MSE

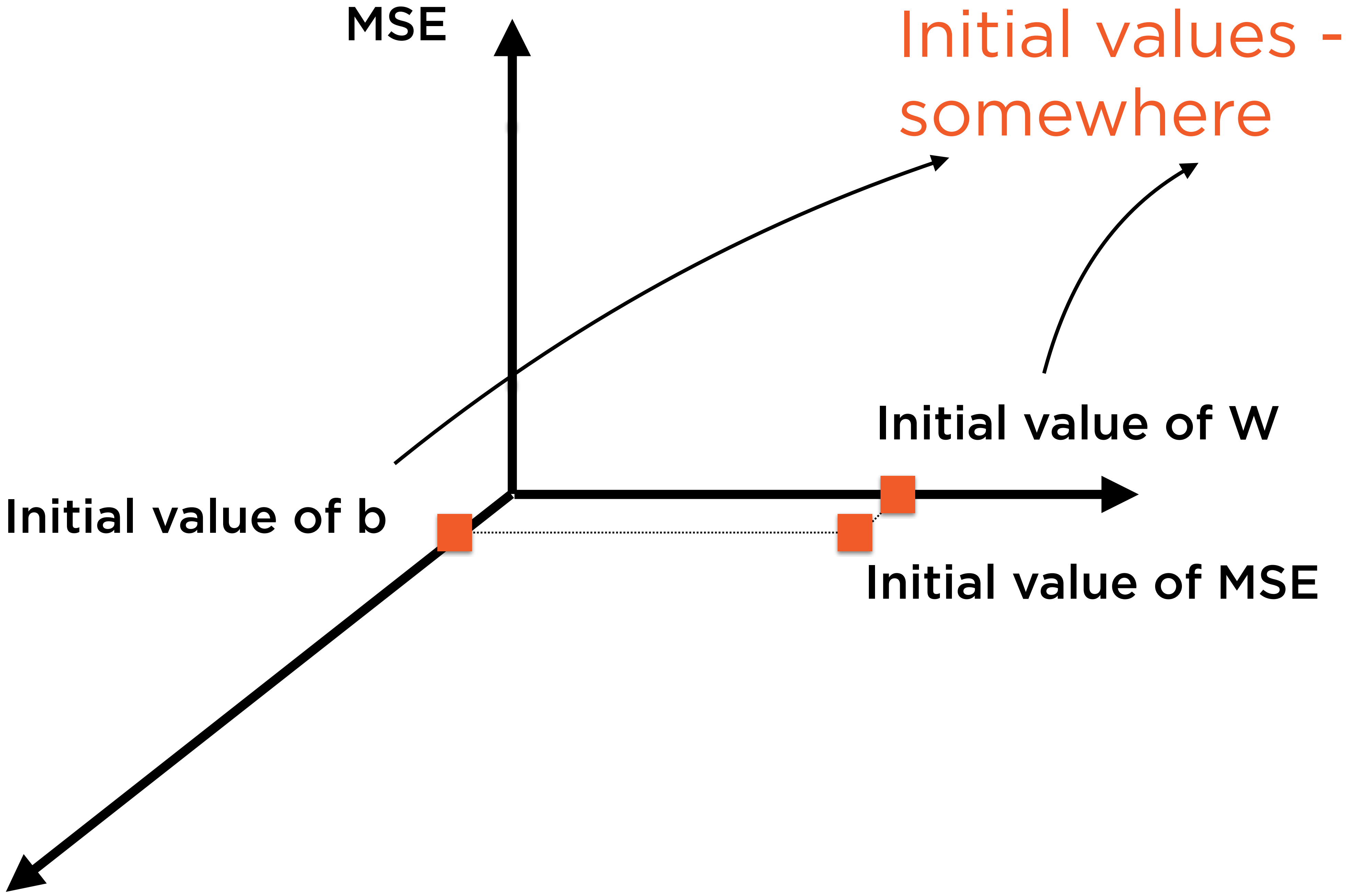


“Training” the Algorithm



“Training Process” =
Finding these best values

Start Somewhere



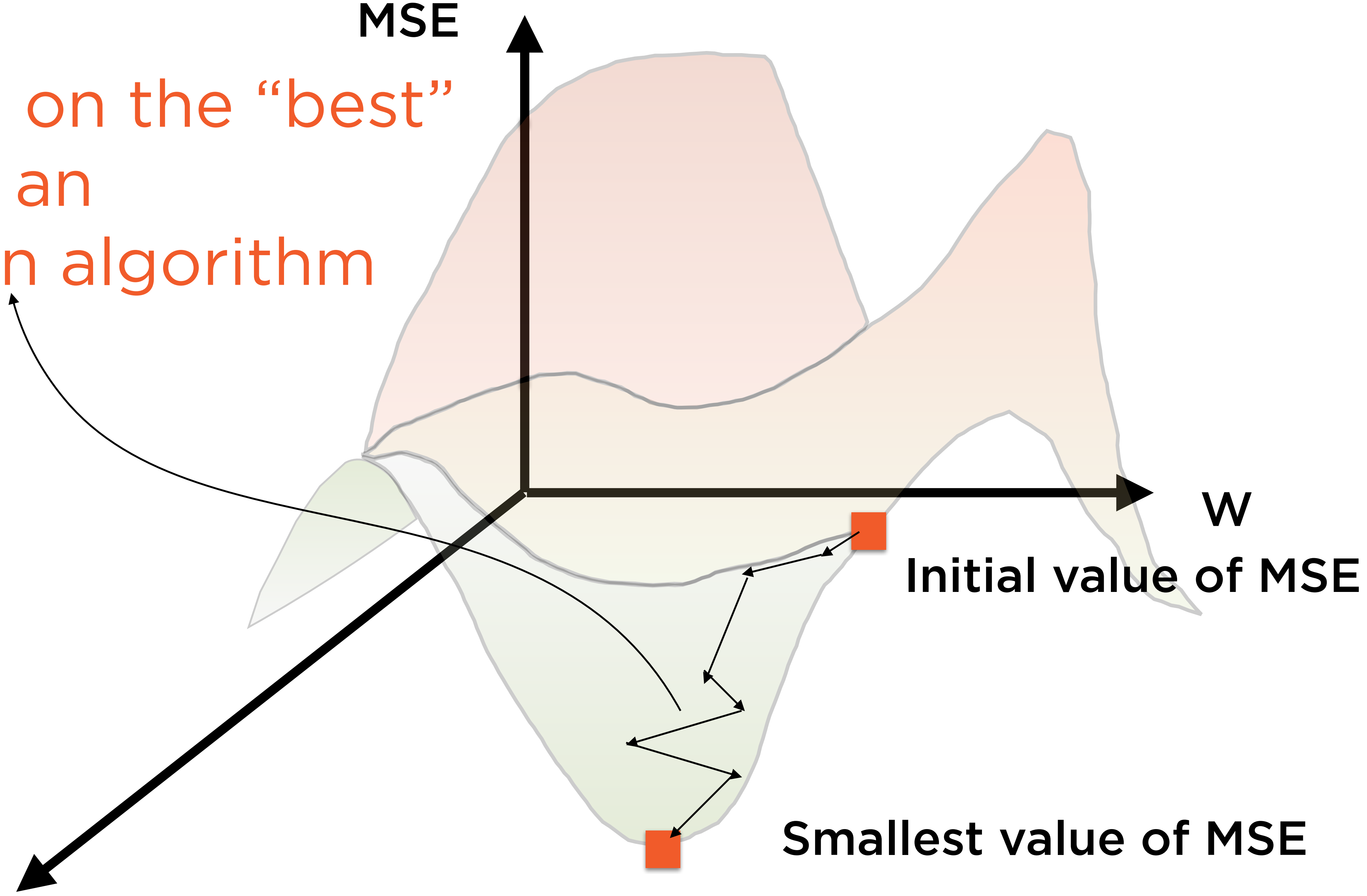
Initial values - have to start somewhere

Initial value of W

Initial value of b

Initial value of MSE

“Gradient Descent”



Converging on the “best” value using an optimization algorithm

Stochastic Gradient Descent **iteratively** converges to the best model

SGD Regressor

Can use different loss functions

MSE loss yields OLS regressor

Can also implement Lasso, Ridge, Elastic Net

SGD Training works well for very large datasets

Demo

Stochastic Gradient Descent regression

Decision Trees for Regression

Jockey or Basketball Player?

Jockeys

Tend to be light to meet horse carrying limits

Basketball Players

Tend to be tall, strong and heavy

Jockey or Basketball Player?

Intuitively know

Jockeys tend to be light

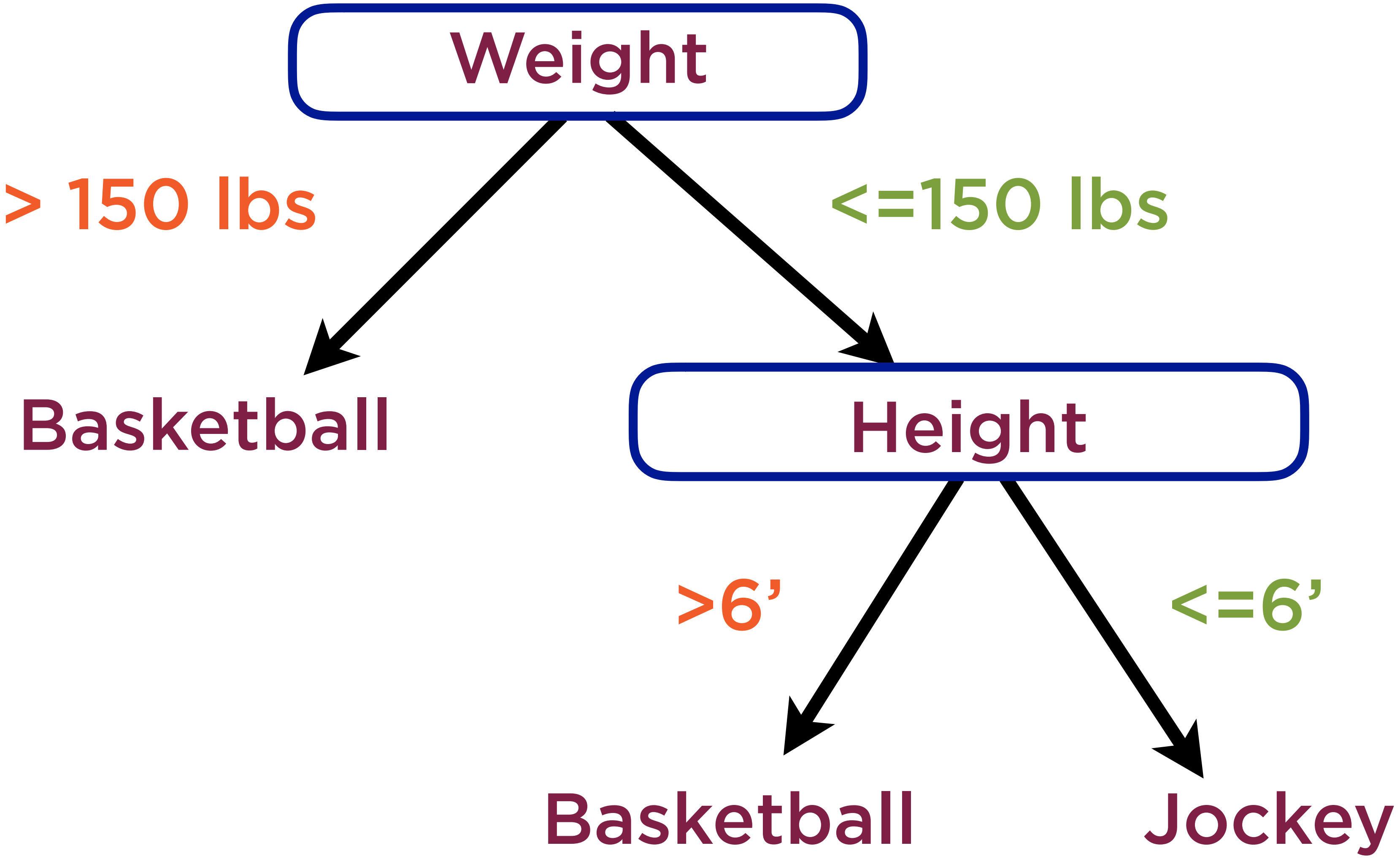
And not very tall

Basketball players tend to be tall

And also quite heavy

Decision trees set up a tree structure on training data which helps make **decisions** based on **rules**

Fit Knowledge into Rules



Weight

> 150 lbs

<=150 lbs

Basketball

Height

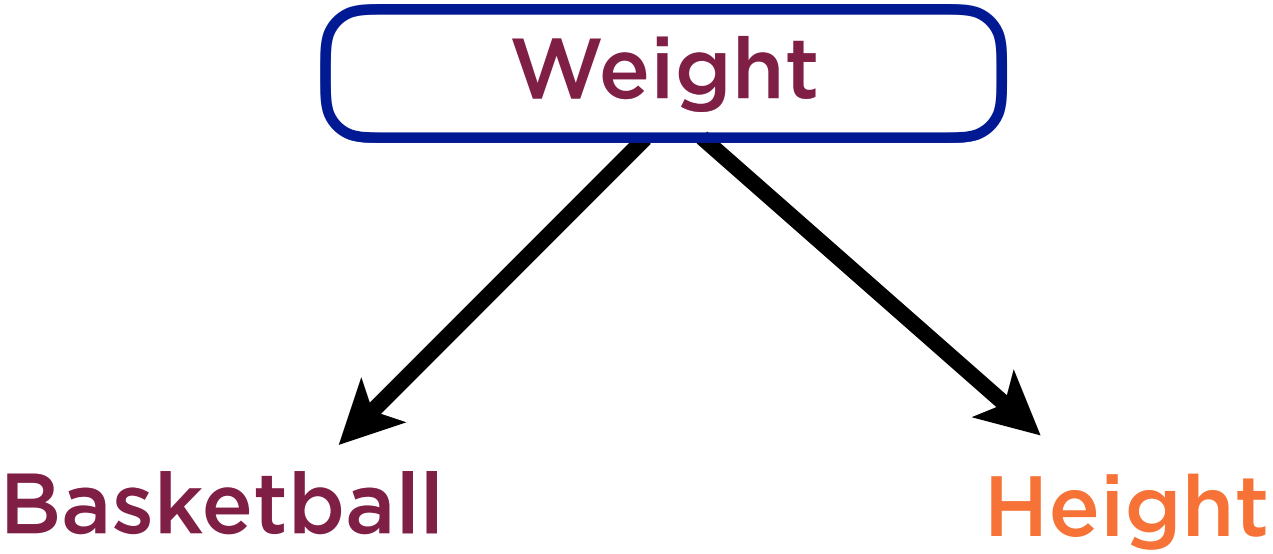
>6’

<=6’

Basketball

Jockey

Decision Based on Weight

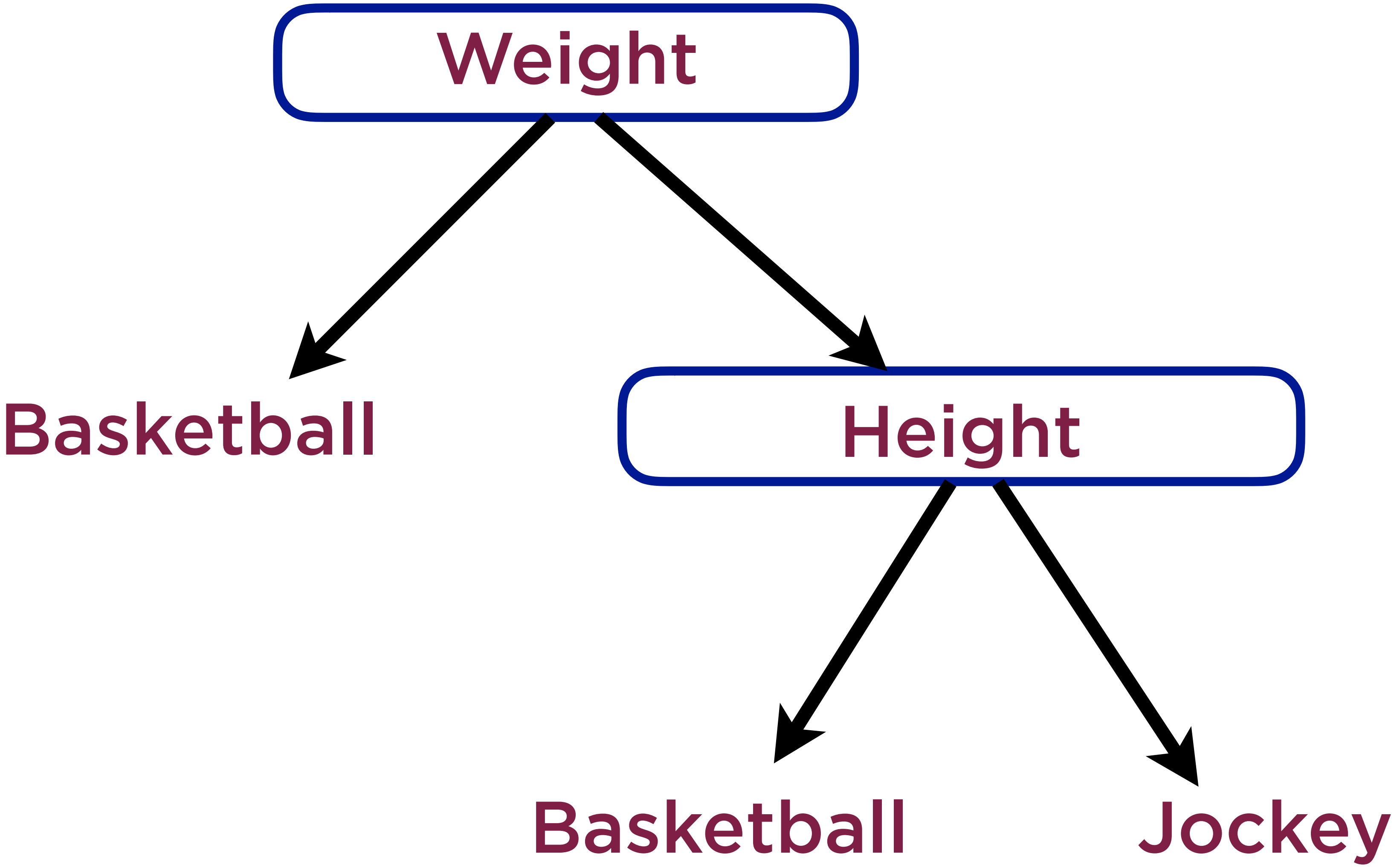


Weight

Basketball

Height

Decision Based on Height


Weight

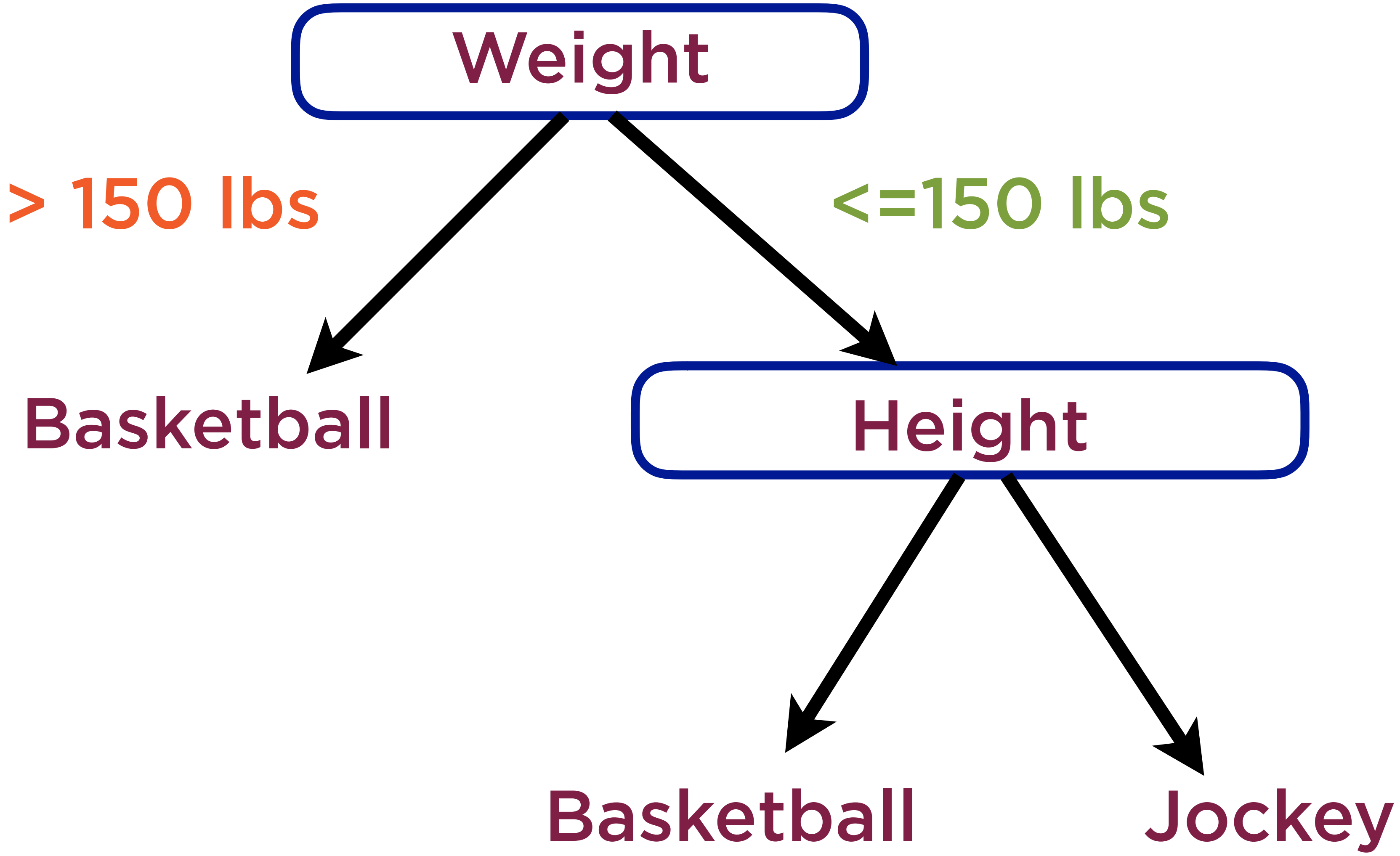
Basketball

Height

Basketball

Jockey

Fit Knowledge into Rules



Weight

> 150 lbs

<=150 lbs

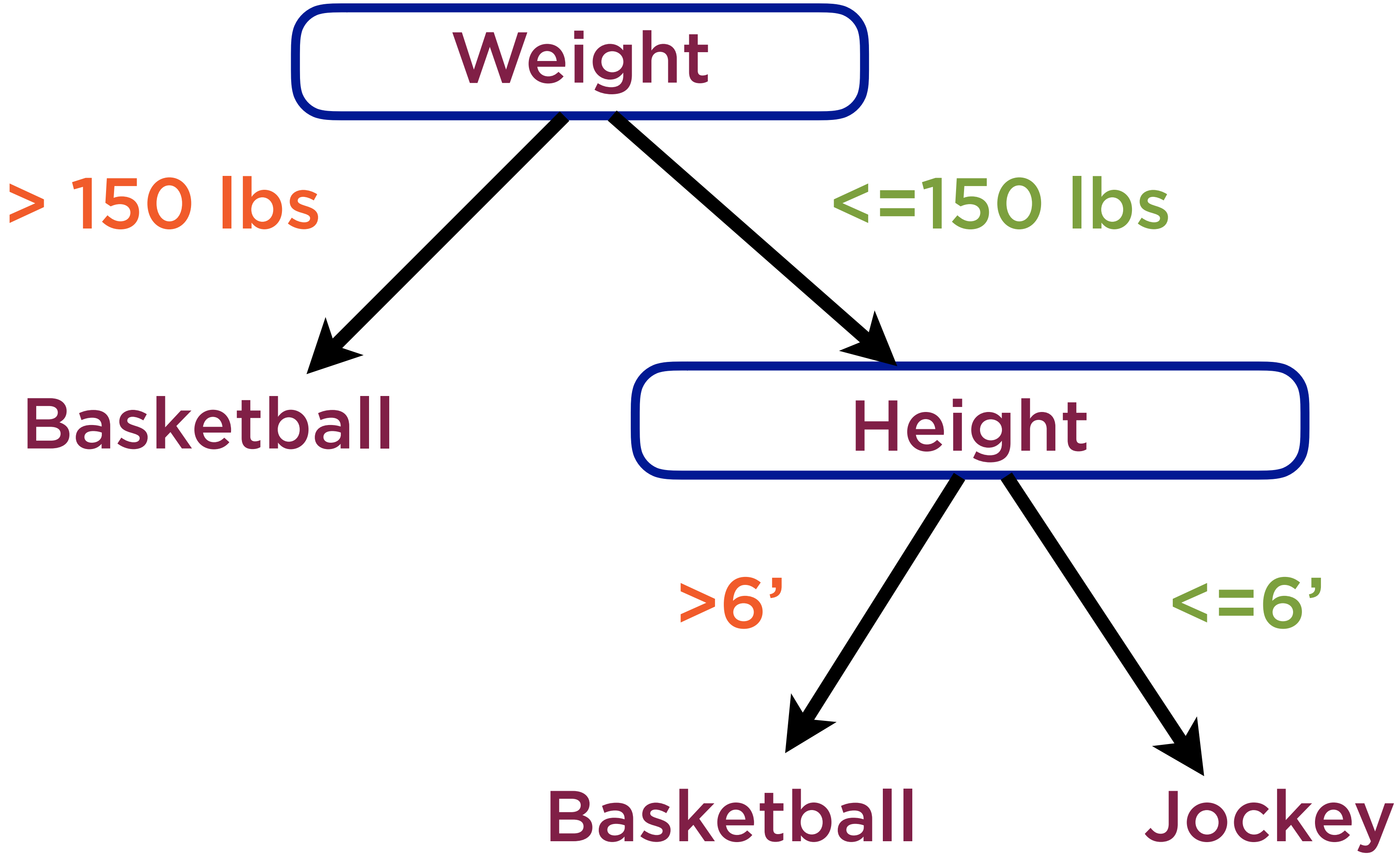
Basketball

Height

Basketball

Jockey

Fit Knowledge into Rules



Weight

> 150 lbs

<=150 lbs

Basketball

Height

>6’

<=6’

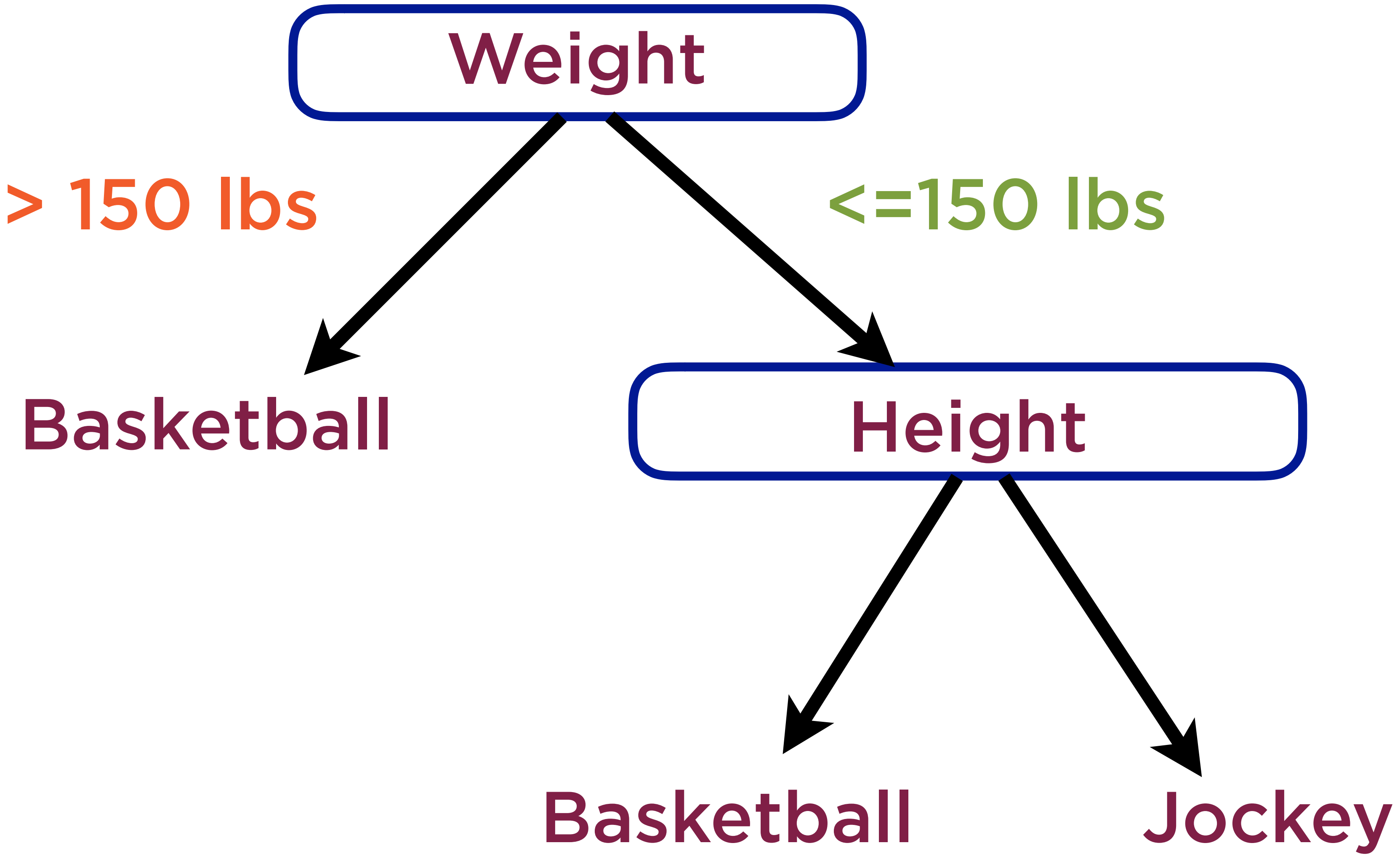
Basketball

Jockey

Decision Tree

Fit knowledge into rules

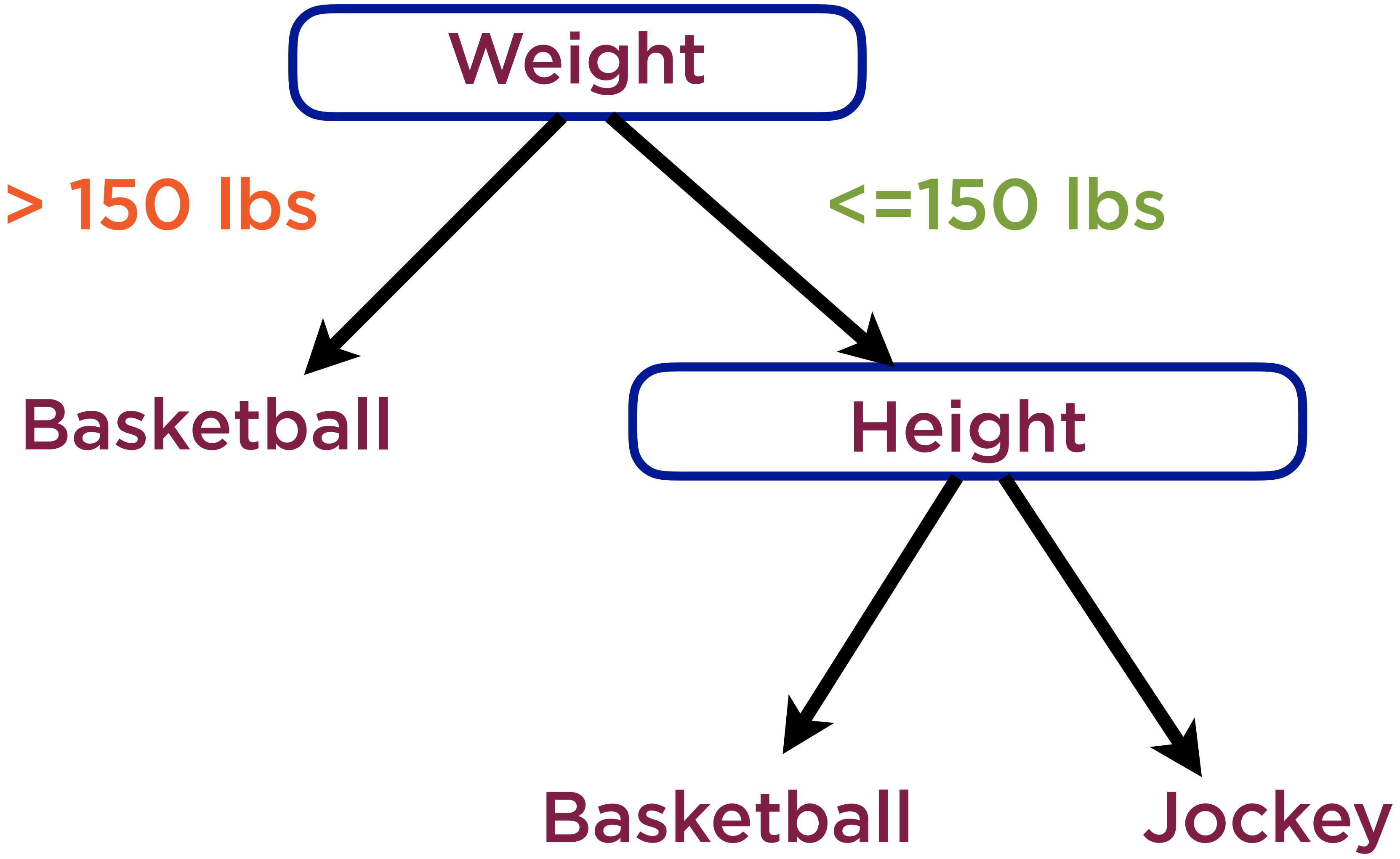
Each rule involves a threshold



Decision Tree

Order of decision variables matters

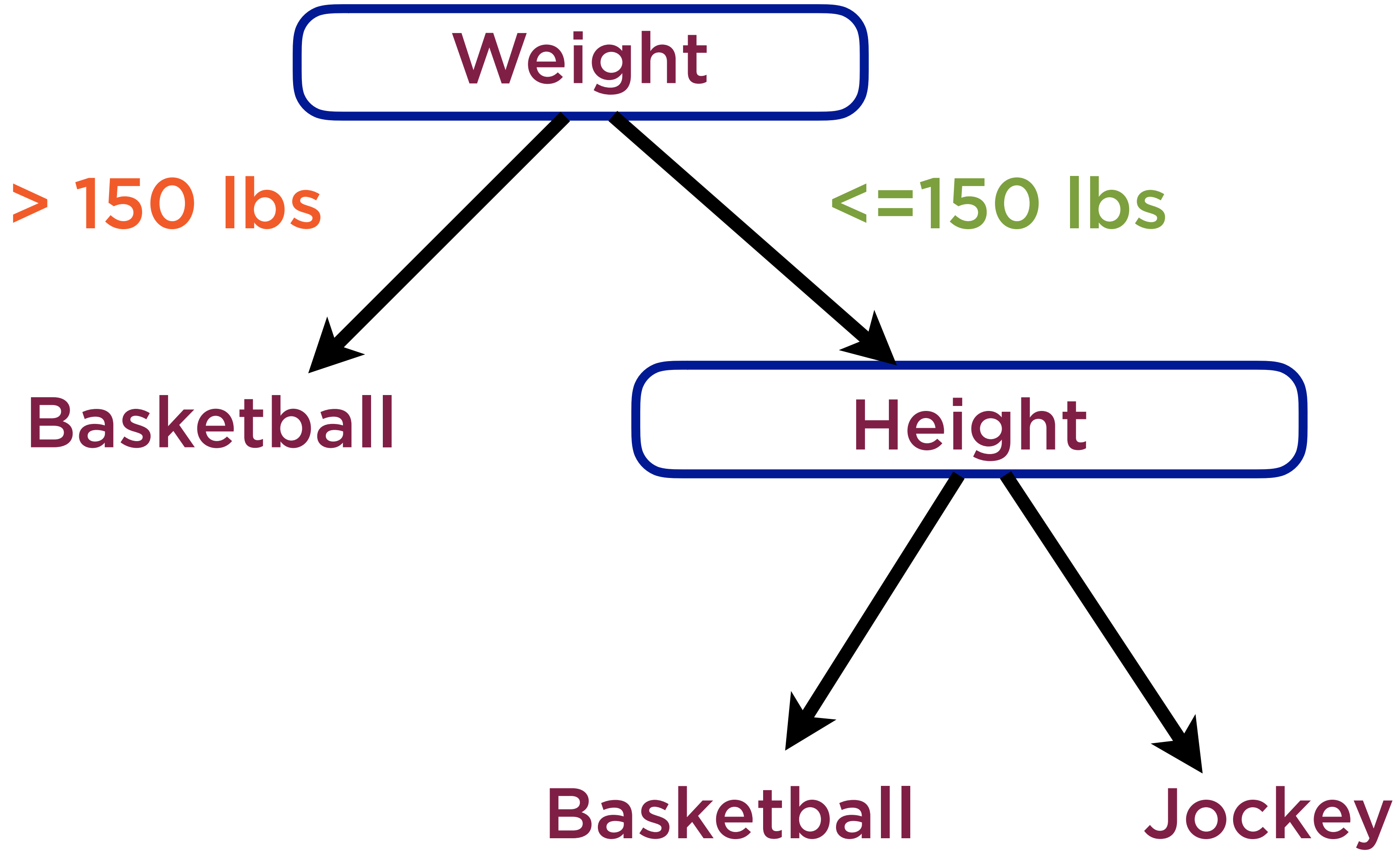
Rules and order found using ML



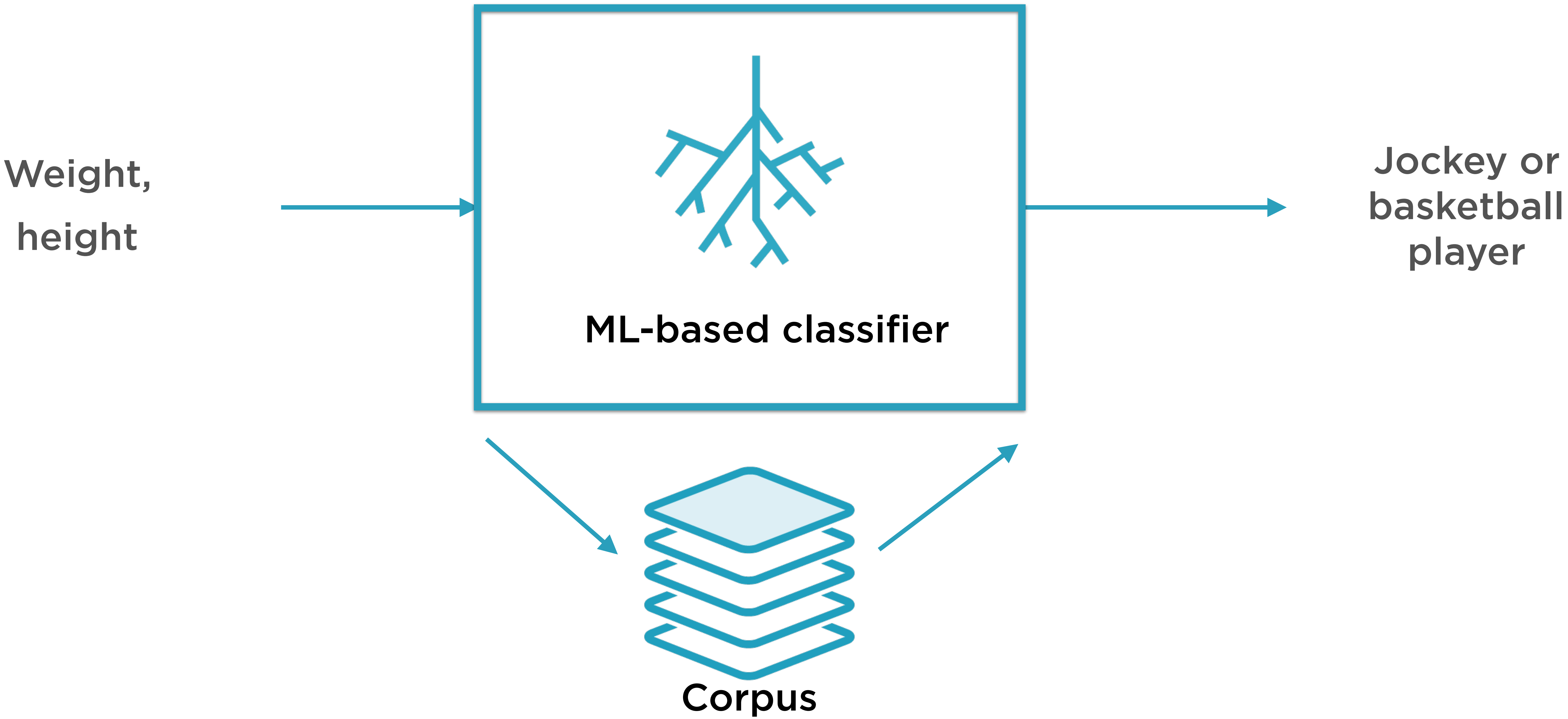
Decision Tree

“CART”

*C*lassification *A*nd
*R*egression *T*ree



Decision Trees for Classification



Weight,

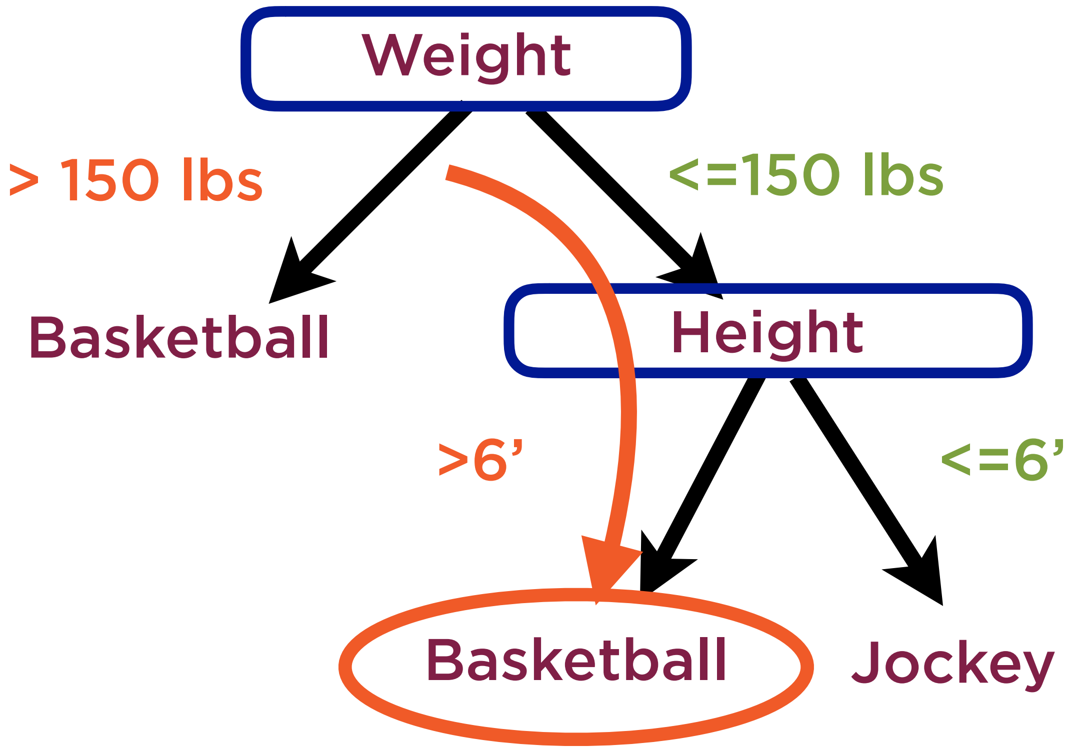
height

ML-based classifier

Jockey or basketball player

Corpus

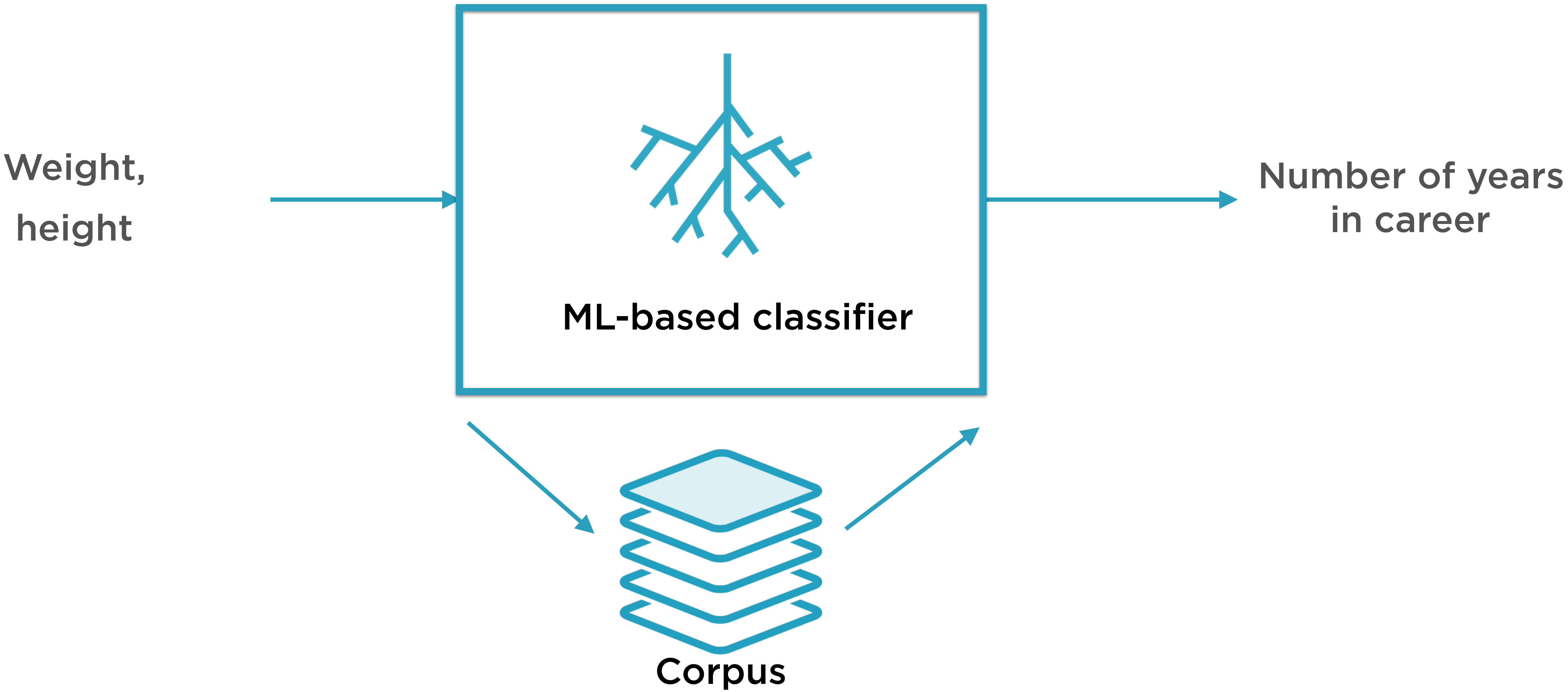
Decision Trees for Classification



Traverse tree to find right node

Return most frequent label of all training data points in that node

Decision Trees for Regression



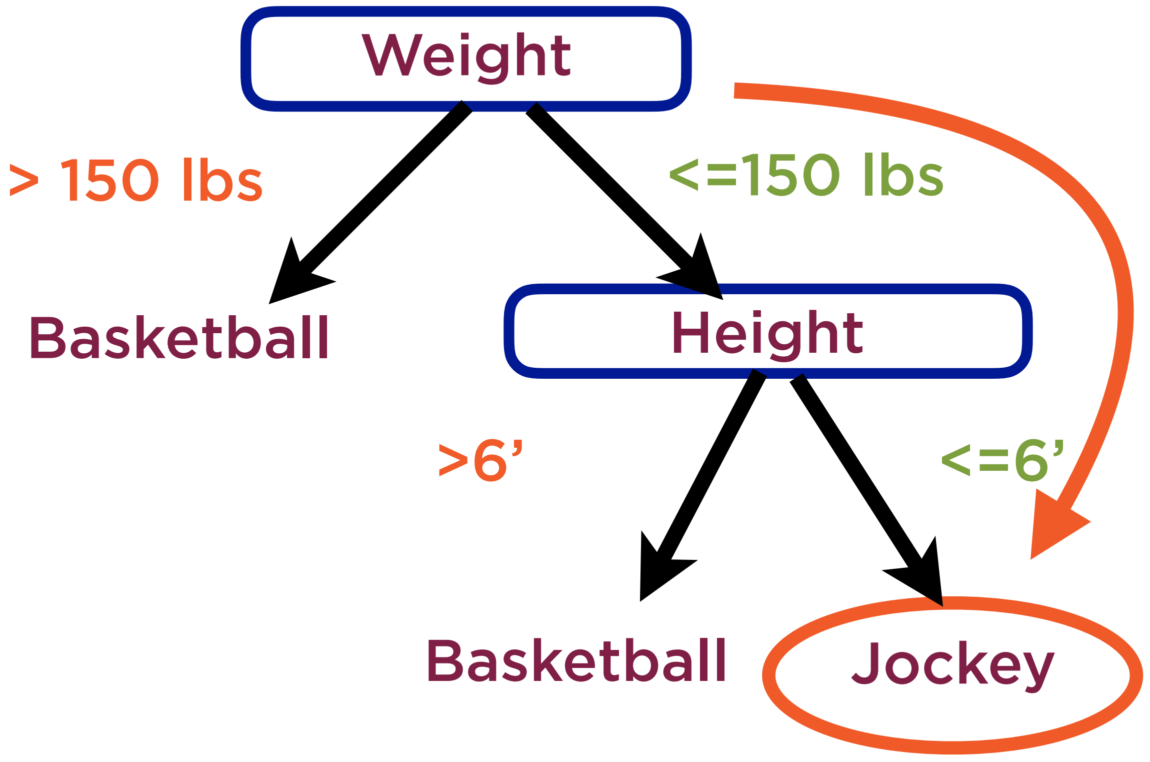
Weight, height

ML-based classifier

Number of years in career

Corpus

Decision Trees for Regression



Traverse tree to find right node

Return the average of number of years of all training data points in that node

Demo

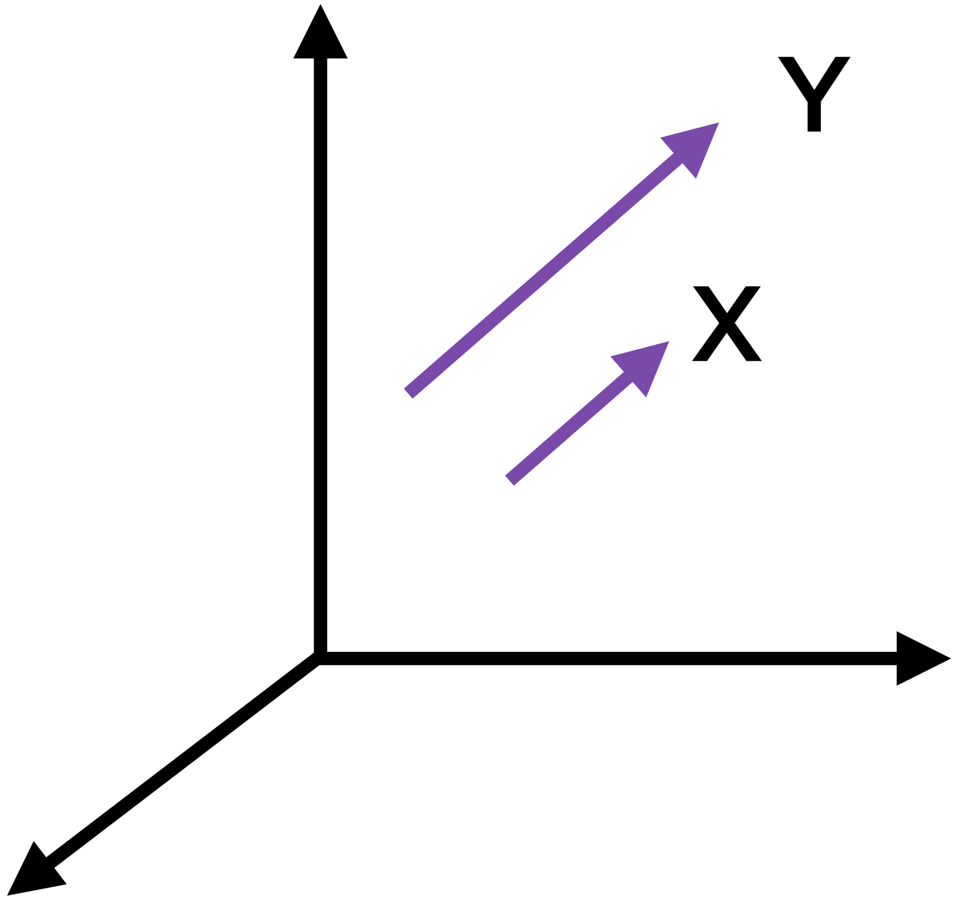
Decision tree regression

Least Angle Regression

Least Angle Regression

A regression technique that relies on selecting x-variables that have the highest correlation (least angle) with the unexplained y-variable

Aligned Vectors



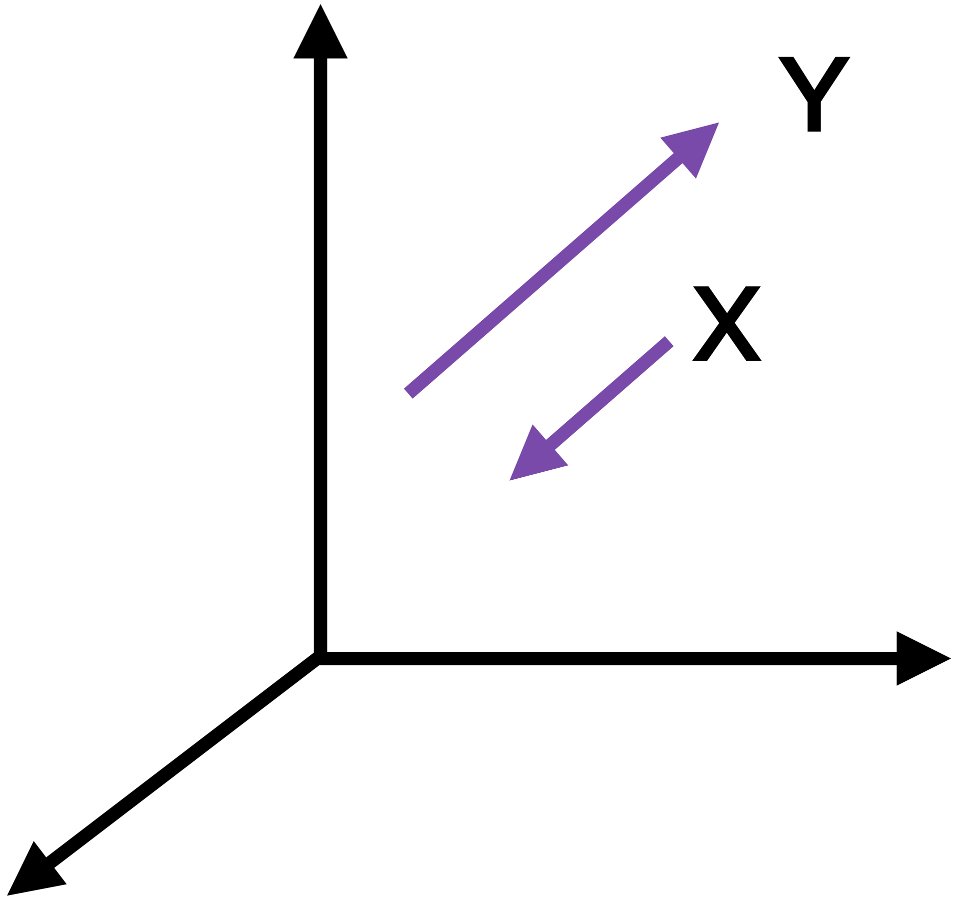
Vectors X and Y are parallel

Angle between them is 0 degrees

Perfectly aligned

Correlation of 1 (highest possible)

Opposite Vectors



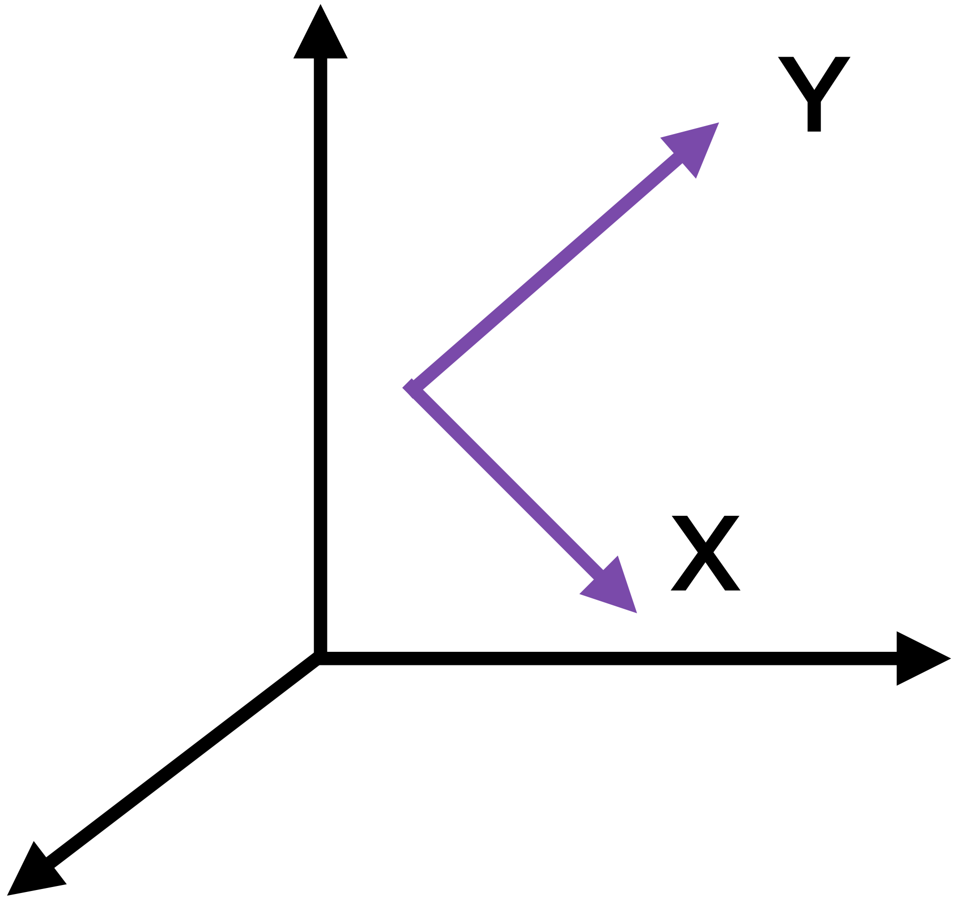
Vectors A and B point in opposite directions

Angle between them is 180 degrees

Perfectly opposed

Correlation of -1 (lowest possible)

Orthogonal Vectors



Vectors X and Y are at 90 degrees

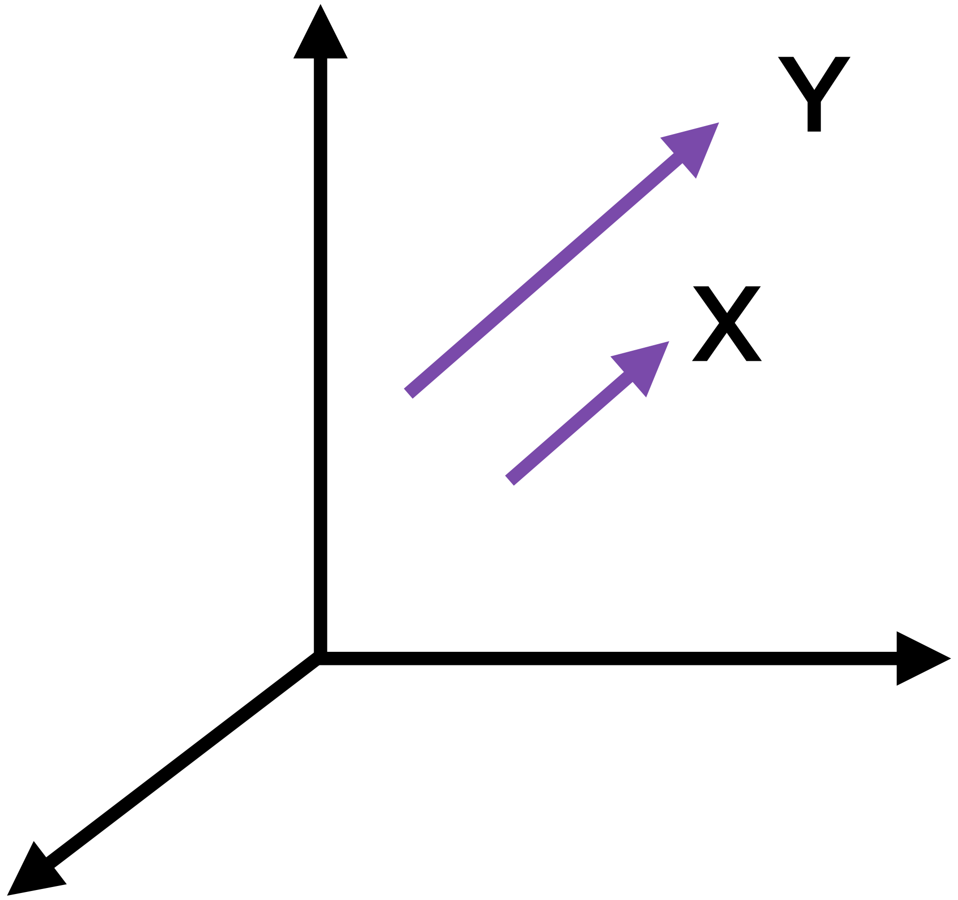
Orthogonal vectors represent uncorrelated data

X and Y are unrelated, independent

LARS Regression

```
Start with all coefficients β equal to zero

Find predictor Xj most correlated with y

Increase coefficient of βj
  Until:
  Some other Xi is more highly correlated than Xj

Increase coefficient of βi , βj

Continue until all X variables are in model
```

Advantages of LARS



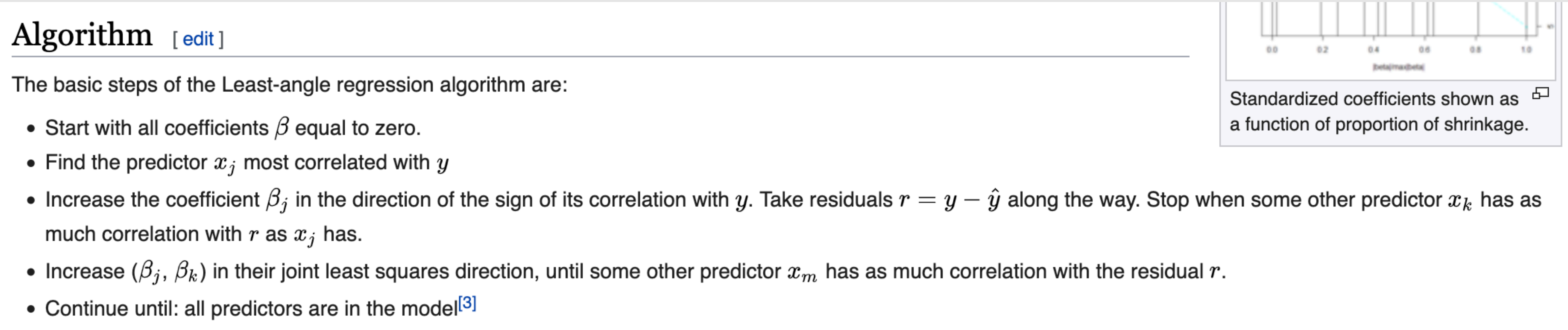
Works well when number of dimensions >> number of points

Intuitive and stable

Equivalent to forward stepwise regression

- Add variables in one-by-one

Has problems dealing with highly correlated x variables



Standardized coefficients shown as a function of proportion of shrinkage.

Algorithm [edit]

The basic steps of the Least-angle regression algorithm are:

- Start with all coefficients β equal to zero.
- Find the predictor x_j most correlated with y
- Increase the coefficient β_j in the direction of the sign of its correlation with y. Take residuals $r = y - \hat{y}$ along the way. Stop when some other predictor x_k has as much correlation with r as x_j has.
- Increase (β_j, β_k) in their joint least squares direction, until some other predictor x_m has as much correlation with the residual r.
- Continue until: all predictors are in the model[3]

Demo

Least angle regression

Regression with Polynomial Relationships

```
y = Wx + b
```

$$f(x) = Wx + b$$

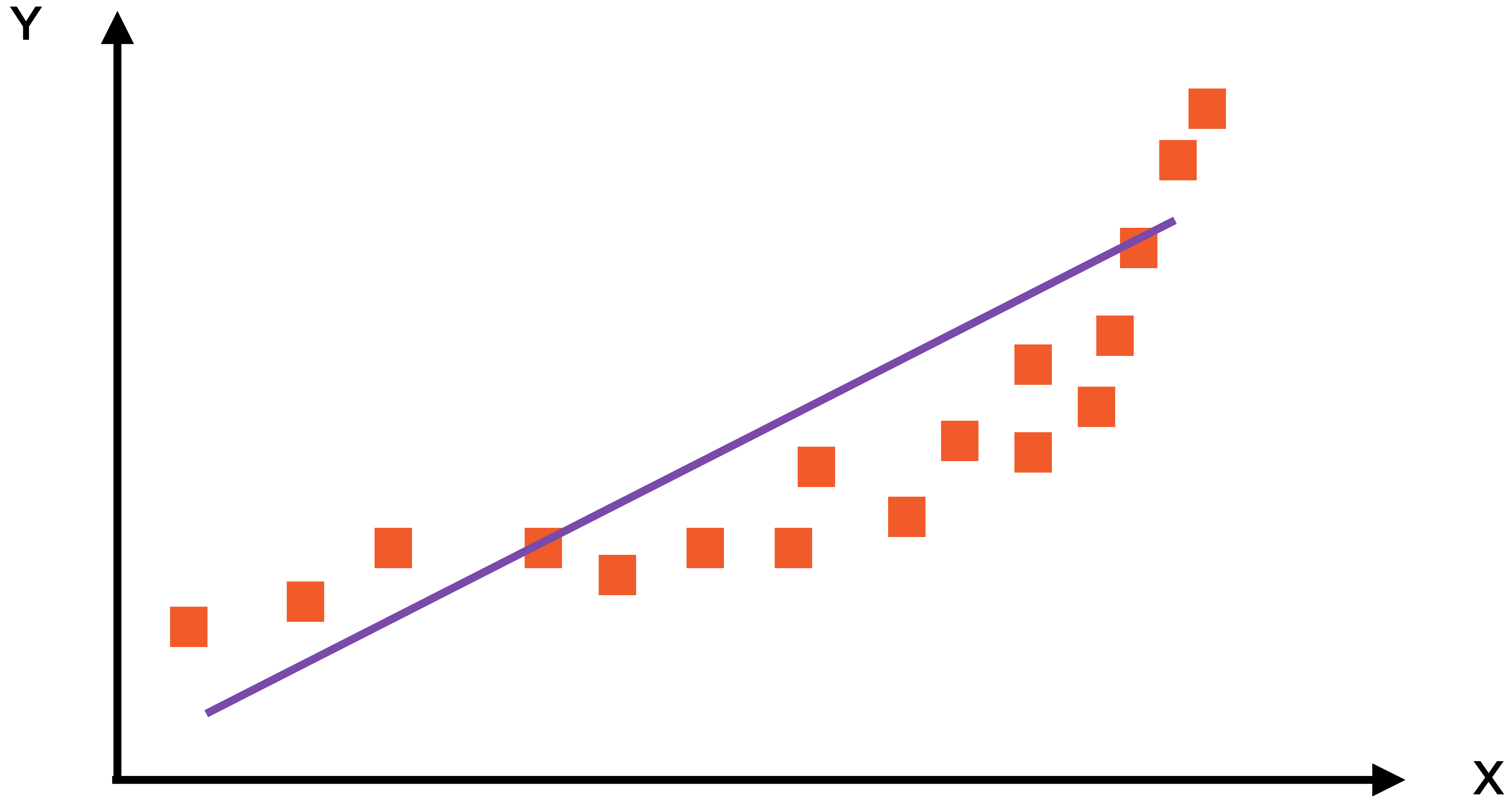
Relationship between y and x is a polynomial of degree 1

$$y = Vx^2 + Wx + b$$

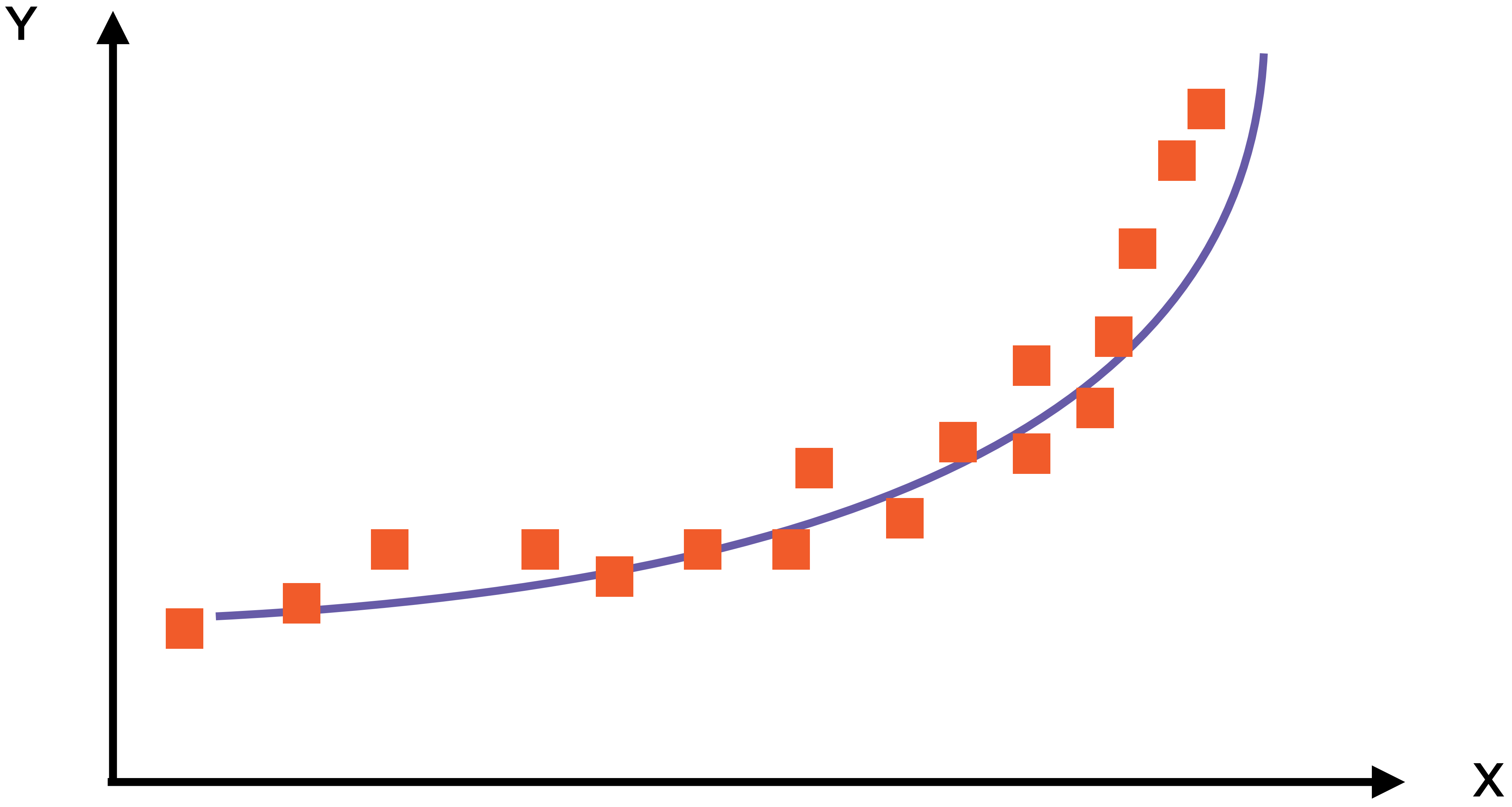
$$f(x) = Vx^2 + Wx + b$$

Now relationship between y and x is a polynomial of degree 2

Linear Fit Performs Poorly



Quadratic Fit Performs Well



Generate polynomials of a certain degree of all input features

Fit a simpler model on this polynomial data

Summary

Choosing regression algorithms based on dataset size and features

Support Vector Machines (SVM)

Nearest neighbors regression

Stochastic Gradient Descent

Decision Tree regression

Least-angle regression (LARS)