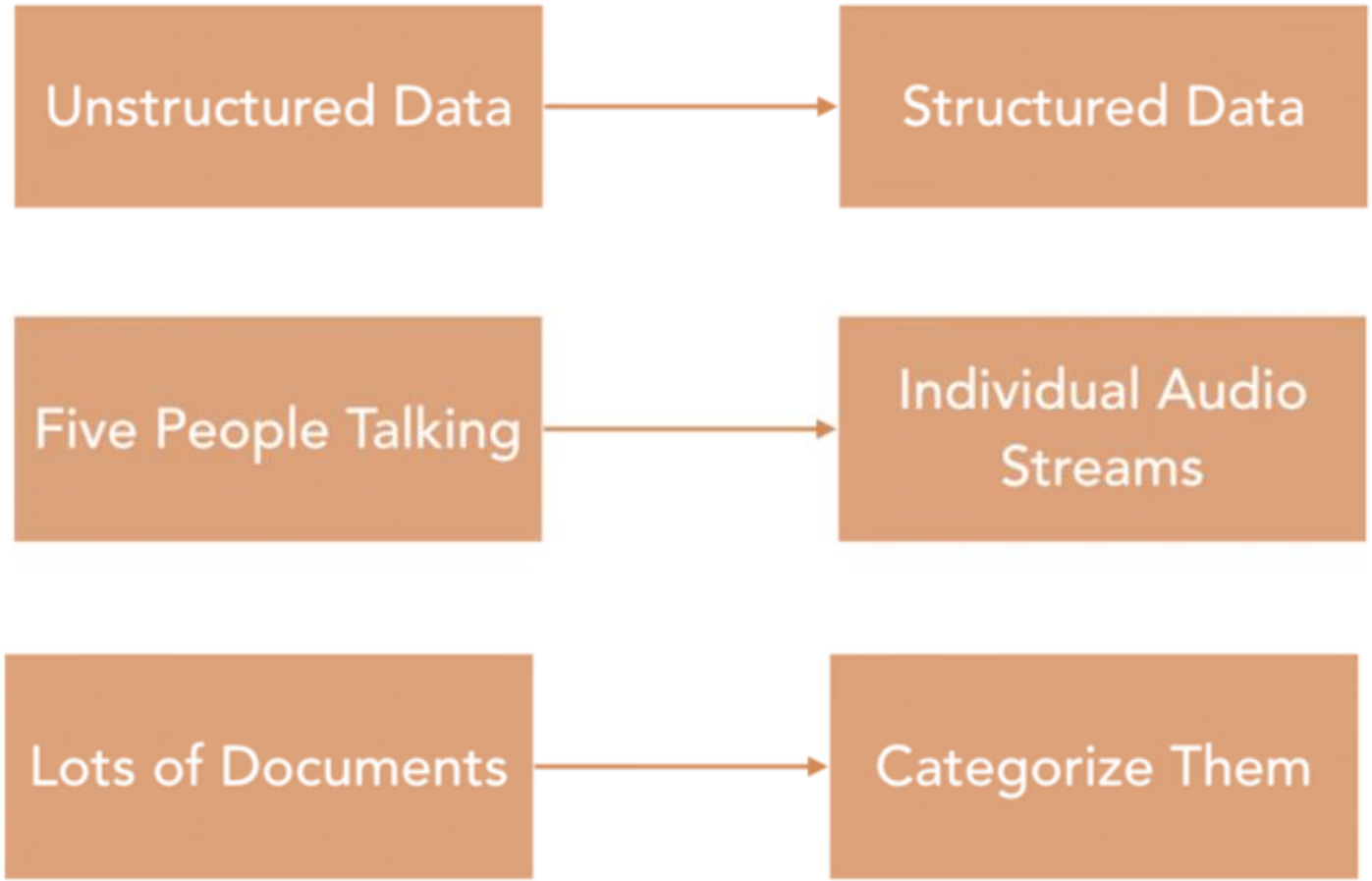
Unsupervised Machine Learning


Unstructured Data
Structured Data
Five People Talking
Individual Audio Streams
Lots of Documents
Categorize Them

Unsupervised Learning Identifies:

- Pattern
- Outliers
- Similarities
- Differences

Unsupervised learning provides correlations, but **your analytical skills** add meaning.

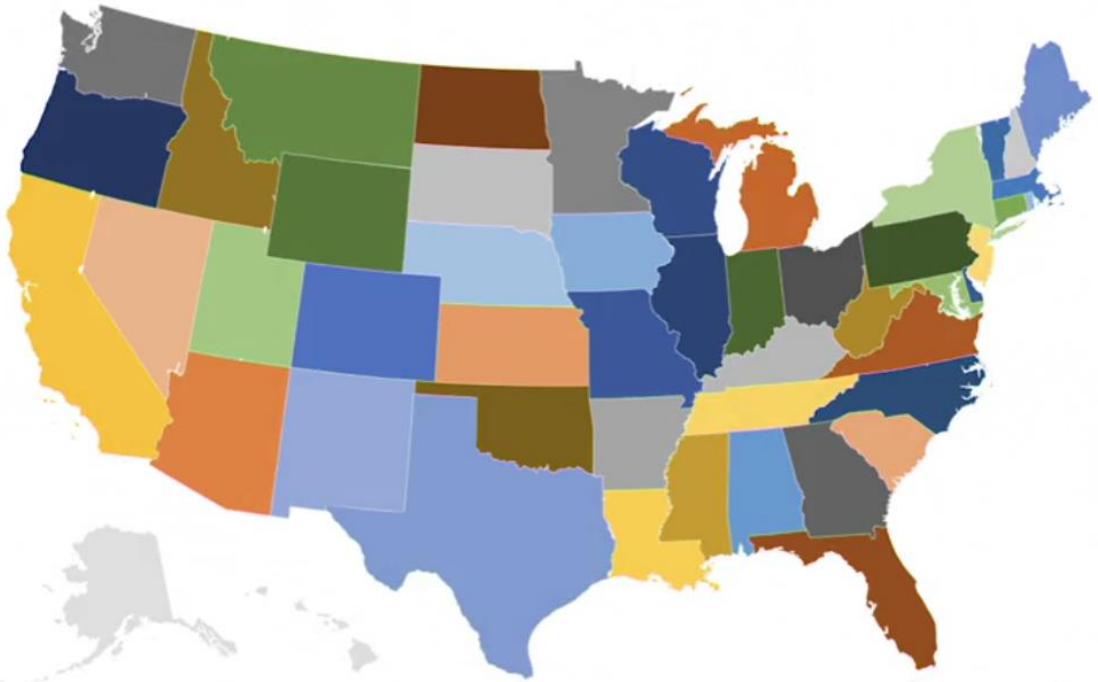
Clustering

1 Marketing Segments

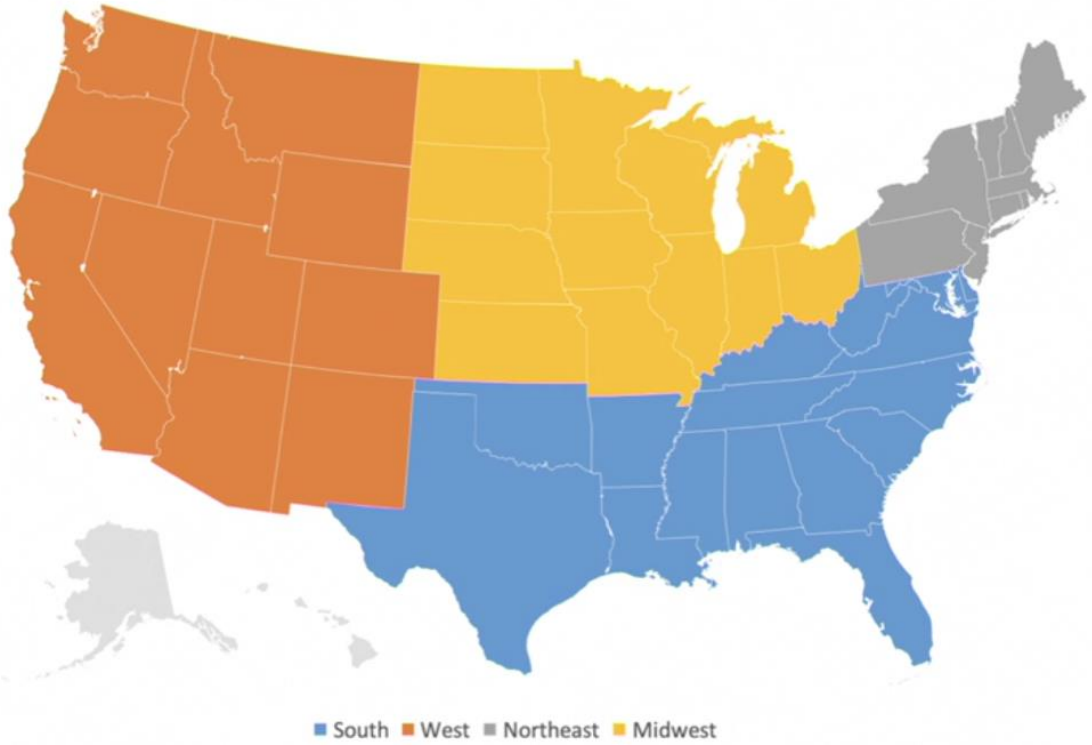
2 Exceptional Students

3 Medical Groups

Map of US States



Map of US States by Region



K-Dimensional Space

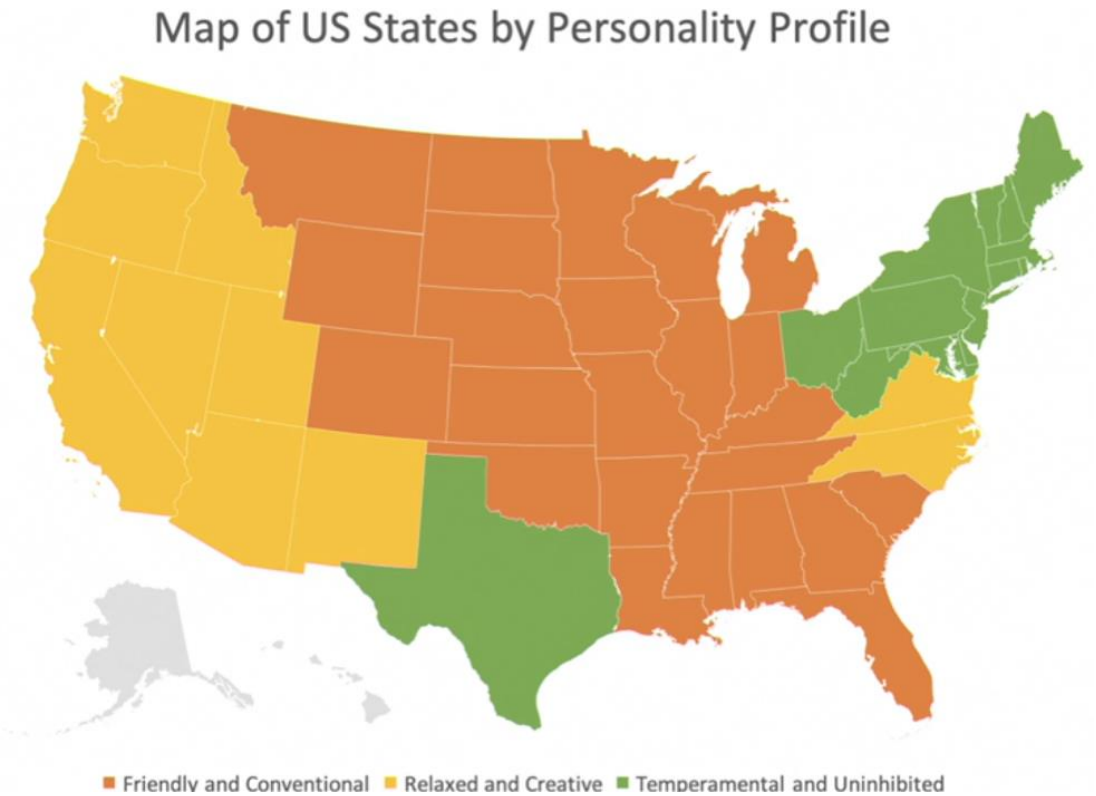
Locate each data point in a multidimensional space with k dimensions for k variables.

Measure Distances

Measure distance of each point to every other point; find clumps and gaps.

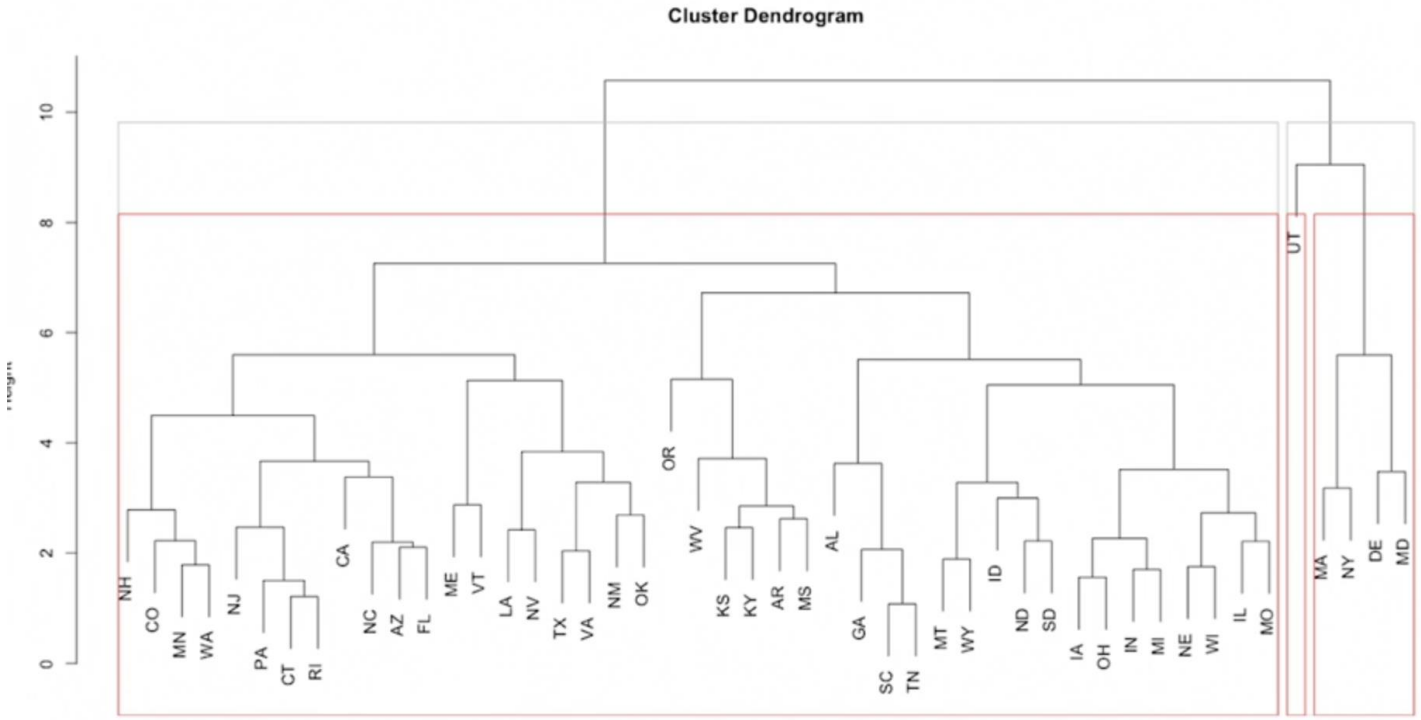
Big 5

- Common personality factors
- Extraversion
- Agreeableness
- Conscientiousness
- Neuroticism
- Openness



Google Correlate Search Terms

- Instagram
- Facebook
- Retweet
- Entrepreneur
- GDPR
- Privacy
- University
- Mortgage
- Volunteering
- Museum
- Scrapbook
- Modern dance



Alternatives

- K-means and group centroid models
- Density models
- Distribution models
- Linkage clustering models

K-Means Clustering Continued…

K-means clustering: unsupervised clustering algorithm where you know how many clusters are appropriate

Market Price and Cost Modeling

Hedge Fund Classification

Insurance Claim Fraud Detection

Customer Segmentation

Predictions are based on the number of centroids present (K) and nearest mean values, given an Euclidean distance measurement between observations.

When using K-means:

- Scale your variables
- Look at a *scatterplot* or the *data table* to estimate the appropriate number of centroids to use for the K parameter value

Precision: A measure of the model's relevancy

- 0: For all the points predicted to have a 0 label, 100% of the retrieved instances were relevant
- 1: For all the points predicted to have a 1 label, 74% of the retrieved instances were relevant

	precision	recall	fl-score	support
0	1.00	1.00	1.00	50
1	0.74	0.78	0.76	50
2	0.77	0.72	0.74	50
avg/total	0.83	0.83	0.83	150

High precision + High recall
= Highly accurate model results

Recall: A measure of the model's completeness

- Of all your points that were labeled 2, 72% of the results that were returned were truly relevant
- Of the entire dataset, 83% of the results that were returned were truly relevant

Hierarchical Clustering

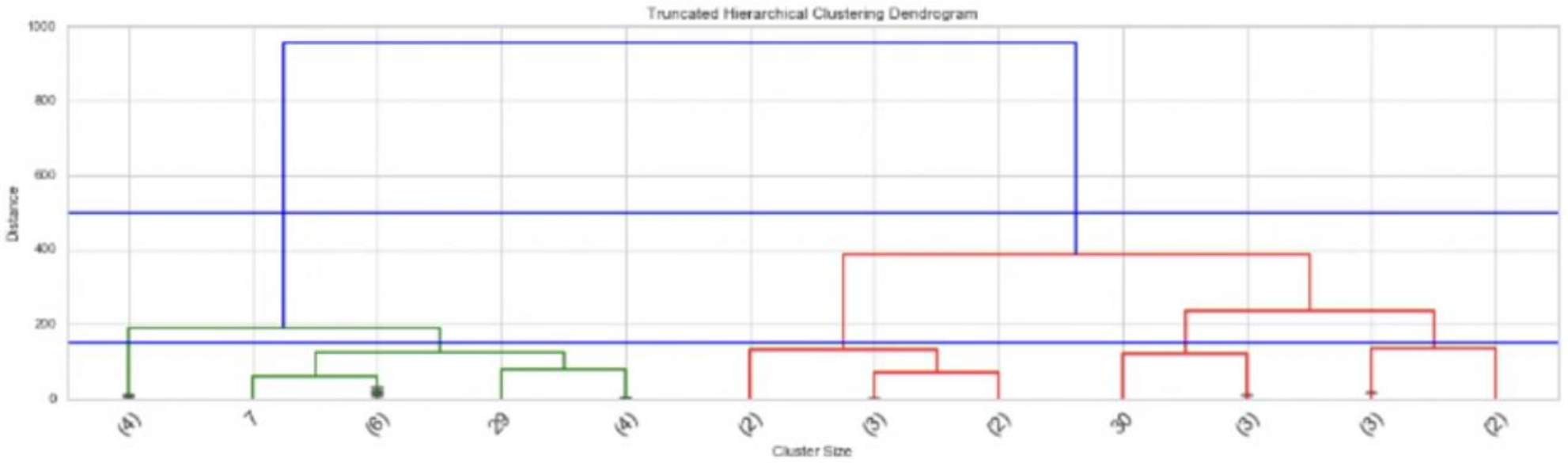
Hierarchical clustering methods predict subgroups within data by finding the distance between each data point and its nearest neighbors, and then linking the most nearby neighbors.

The algorithm uses the distance metric it calculates to predict subgroups.

To guess the number of subgroups in a dataset, first look at a dendrogram visualization of the clustering results.

Dendrogram: a tree graph that's useful for visually displaying taxonomies, lineages, and relatedness

Hierarchical Clustering Dendrogram



Hospital Resource Management

Business Process Management

Customer Segmentation

Social Network Analysis

Hierarchical Clustering Parameters

Distance Metrics

- Euclidean
- Manhattan
- Cosine

Linkage Parameters

- Ward
- Complete
- Average

Parameter selection method: use trial and error

DBSCAN: Outlier Detection

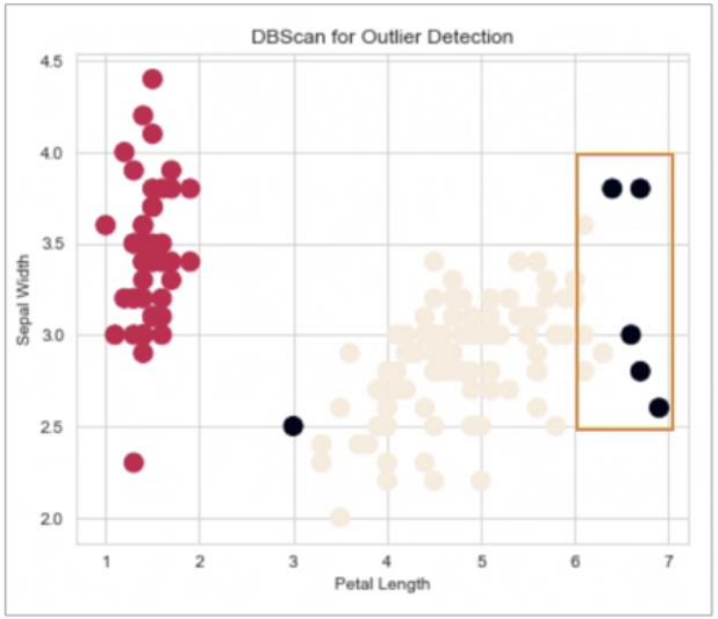
- Unsupervised method that clusters *core samples* (dense areas of a dataset) and denotes *non-core samples* (sparse portions of the dataset)
- Use to identify collective outliers
- Outliers should make up ≤ 5% of the total observations. Adjust model parameters accordingly

Important DBSCAN model parameters:

- *eps*
- *min_samples*

Counter ({1: 94, 0: 50, -1: 6})				
	0	1	2	3
98	5.1	2.5	3.0	1.1
105	7.6	3.0	6.6	2.1
117	7.7	3.8	6.7	2.2
118	7.7	2.6	6.9	2.3
122	7.7	2.8	6.7	2.0
131	7.9	3.8	6.4	2.0

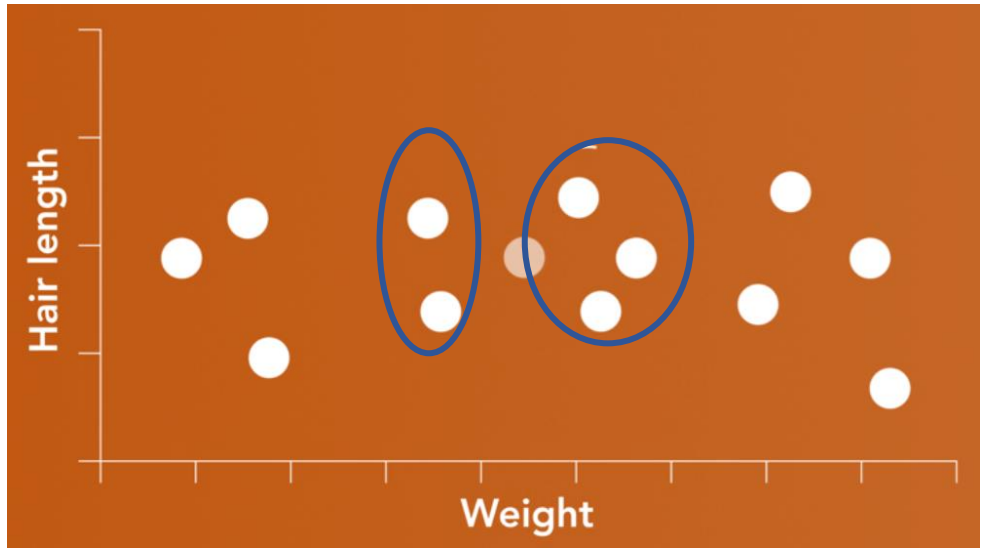
150 total observations
6 with -1 label (4%)

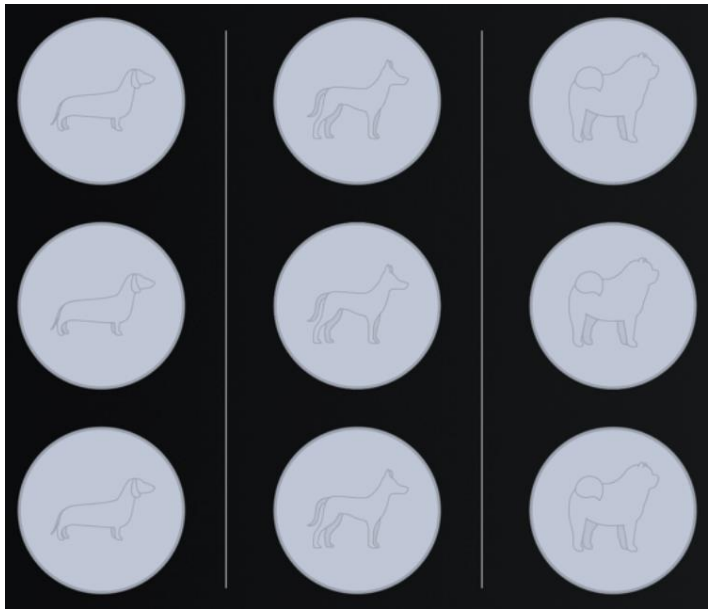


Collective Outliers

K-Means Clustering

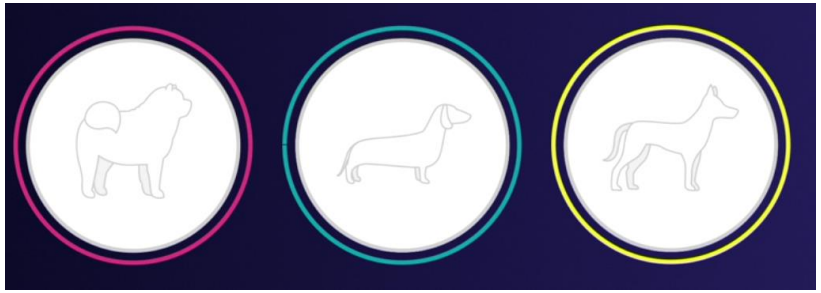
k-Means Clustering ≠ k-NN





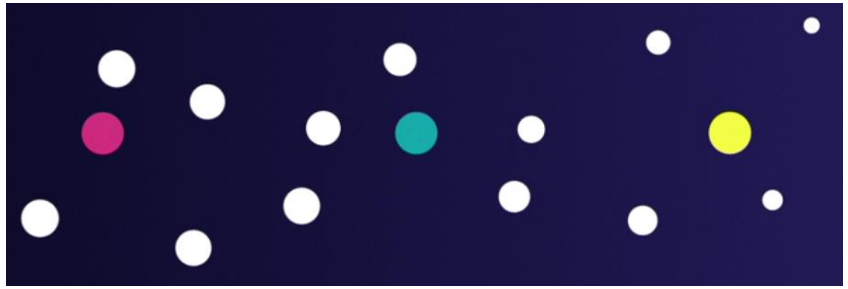
1.

K=3

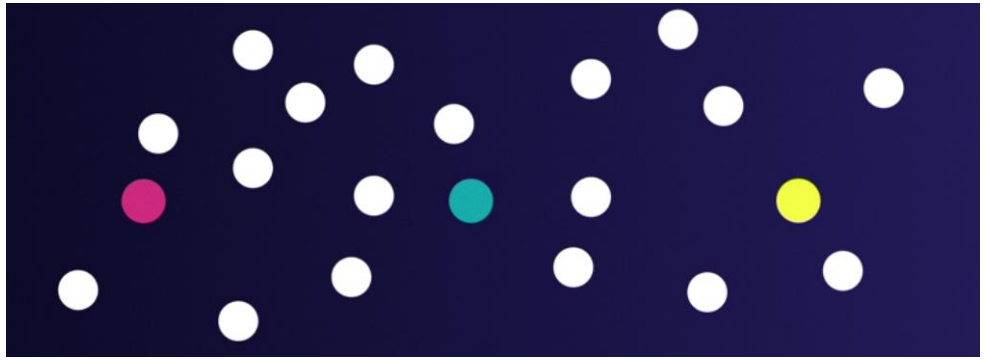


2.

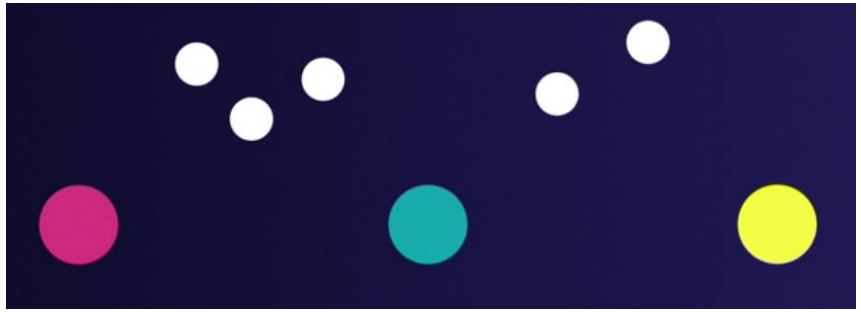
3.



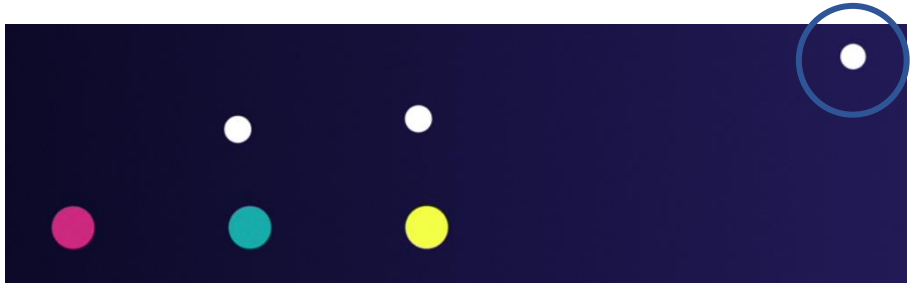
4.



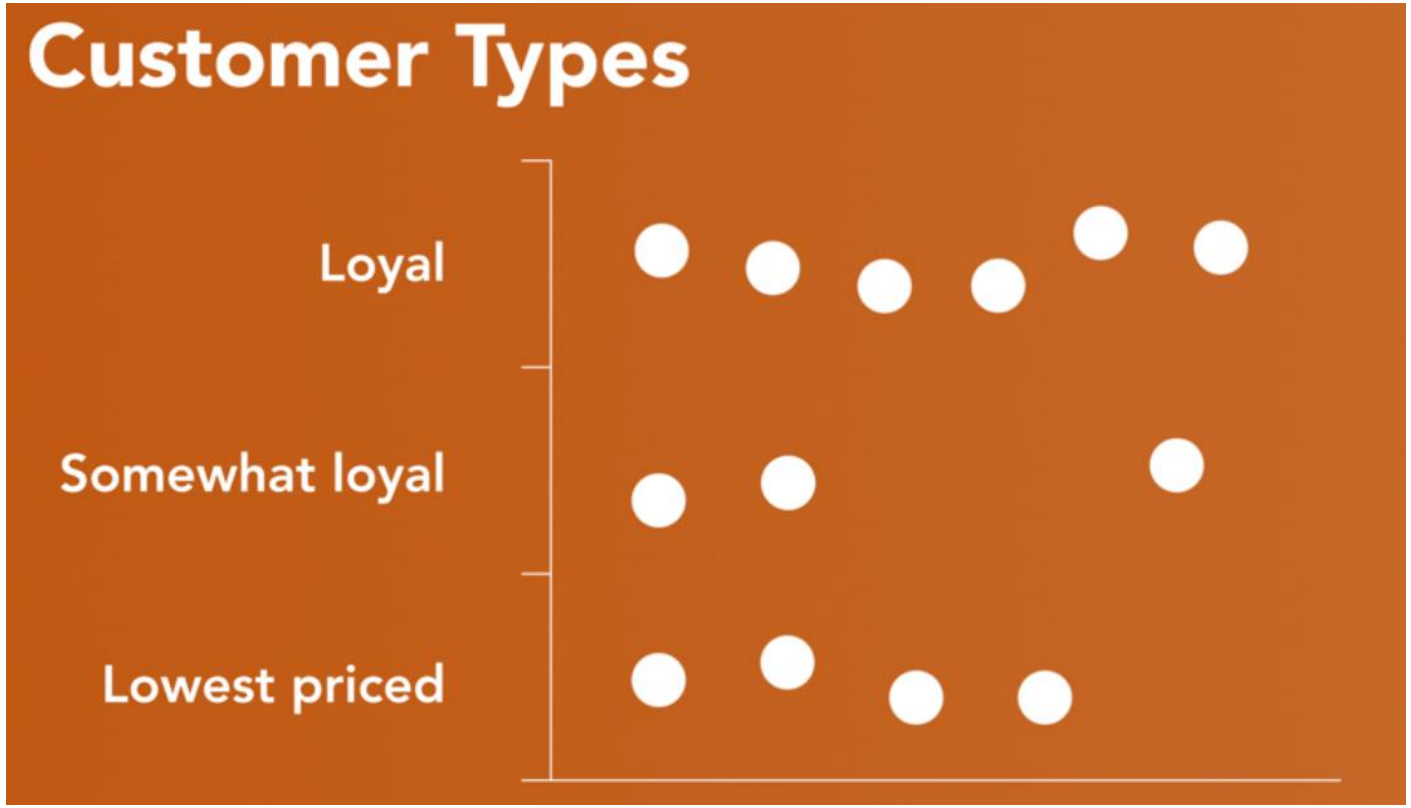
5.



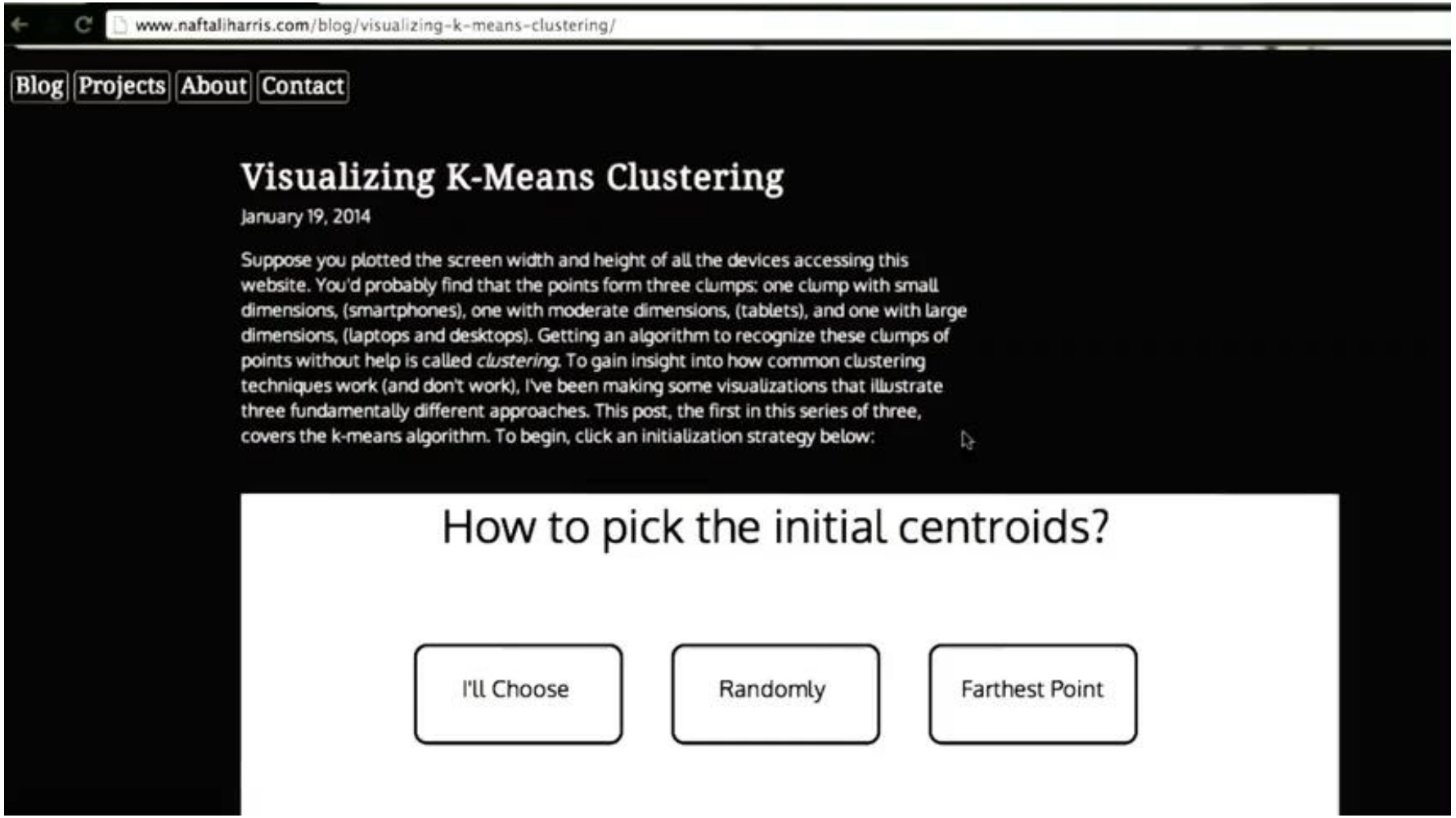
6.

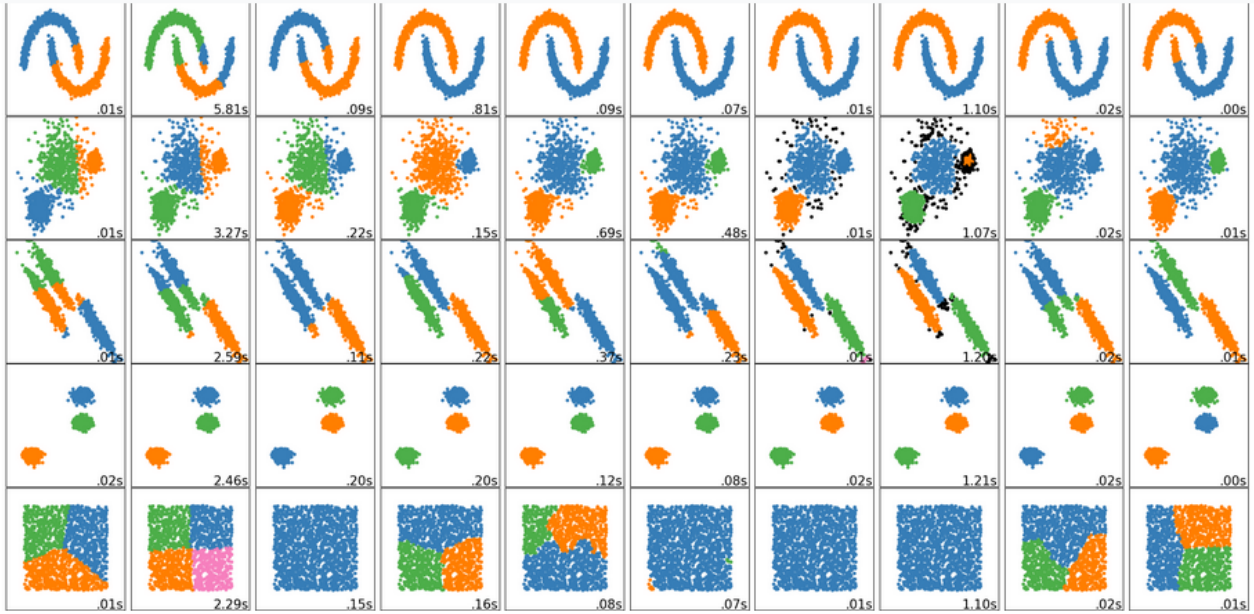


Outlier


Customer Types
Loyal
Somewhat loyal
Lowest priced

K-Means Continued





A comparison of the clustering algorithms in scikit-learn

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points

sklearn.cluster.KMeans

class `sklearn.cluster.KMeans`(*n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True, n_jobs='deprecated', algorithm='auto'*) [source]

K-Means clustering.

Read more in the User Guide.

Parameters:

n_clusters : *int, default=8*
The number of clusters to form as well as the number of centroids to generate.

init : *{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++'*
Method for initialization:

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

'random': choose `n_clusters` observations (rows) at random from data for the initial centroids.

If an array is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

If a callable is passed, it should take arguments X, n_clusters and a random state and return an initialization.

n_init : *int, default=10*
Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

max_iter : *int, default=300*
Maximum number of iterations of the k-means algorithm for a single run.

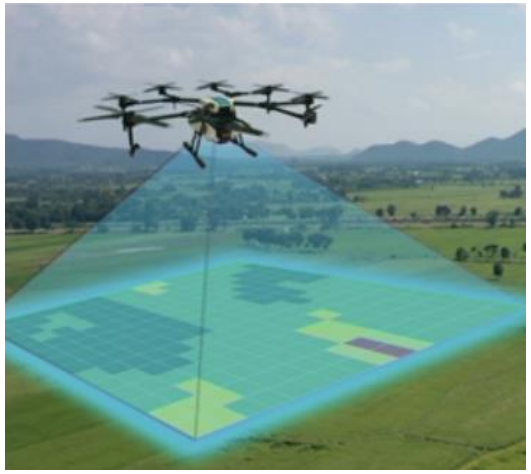
tol : *float, default=1e-4*
Relative tolerance with regards to Frobenius norm of the difference in the cluster centers of two consecutive iterations to declare convergence.

precompute_distances : *{'auto', True, False}, default='auto'*
Precompute distances (faster but takes more memory).

'auto' : do not precompute distances if n_samples * n_clusters > 12 million. This corresponds to about 100MB overhead per job using double precision.

Feature Scaling

Example: Count the number of species with a drone.

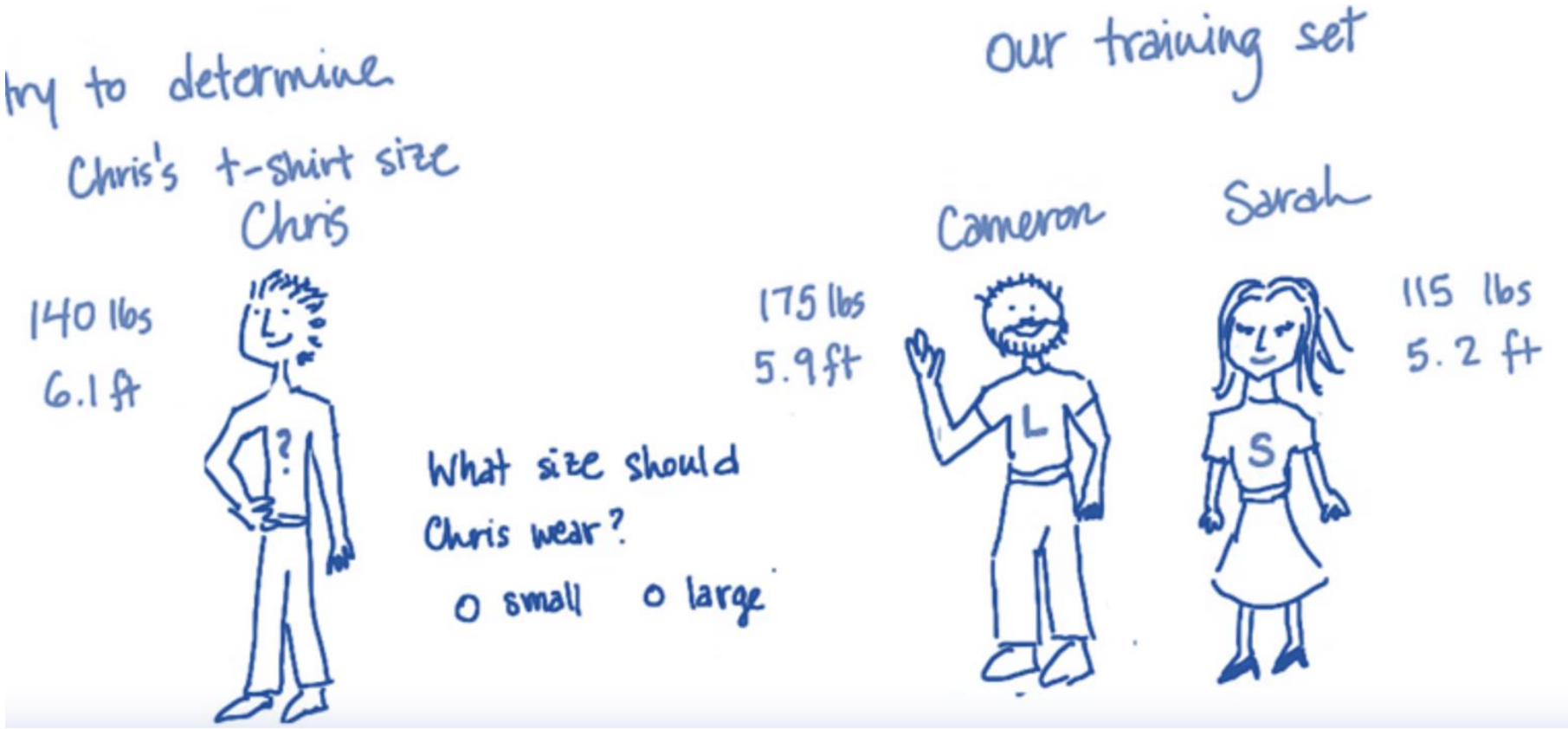


$$z = \frac{x - \min(x)}{[\max(x) - \min(x)]}$$

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean
σ = Standard deviation

Intuition


try to determine
Chris's t-shirt size
Chris
140 lbs
6.1 ft
?
What size should Chris wear?
o small o large
our training set
Cameron
175 lbs
5.9 ft
L
Sarah
115 lbs
5.2 ft
S

try to determine Chris's t-shirt size

height + weight

our training set

Chris

140 lbs
6.1 ft

What is height + weight for Chris?

146.1

Cameron

175 lbs
5.9 ft

Sarah

115 lbs
5.2 ft

try to determine height + weight our training set

Chris's t-shirt size

Chris

140 lbs

6.1 ft

146.1

What is height + weight for Sarah?

120.2

180.9

175 lbs

5.9 ft

Cameron

Sarah

115 lbs

5.2 ft

try to determine Chris's t-shirt size

height + weight

our training set

Chris

140 lbs

6.1 ft

146.1

Cameron

175 lbs

5.9 ft

180.9

Sarah

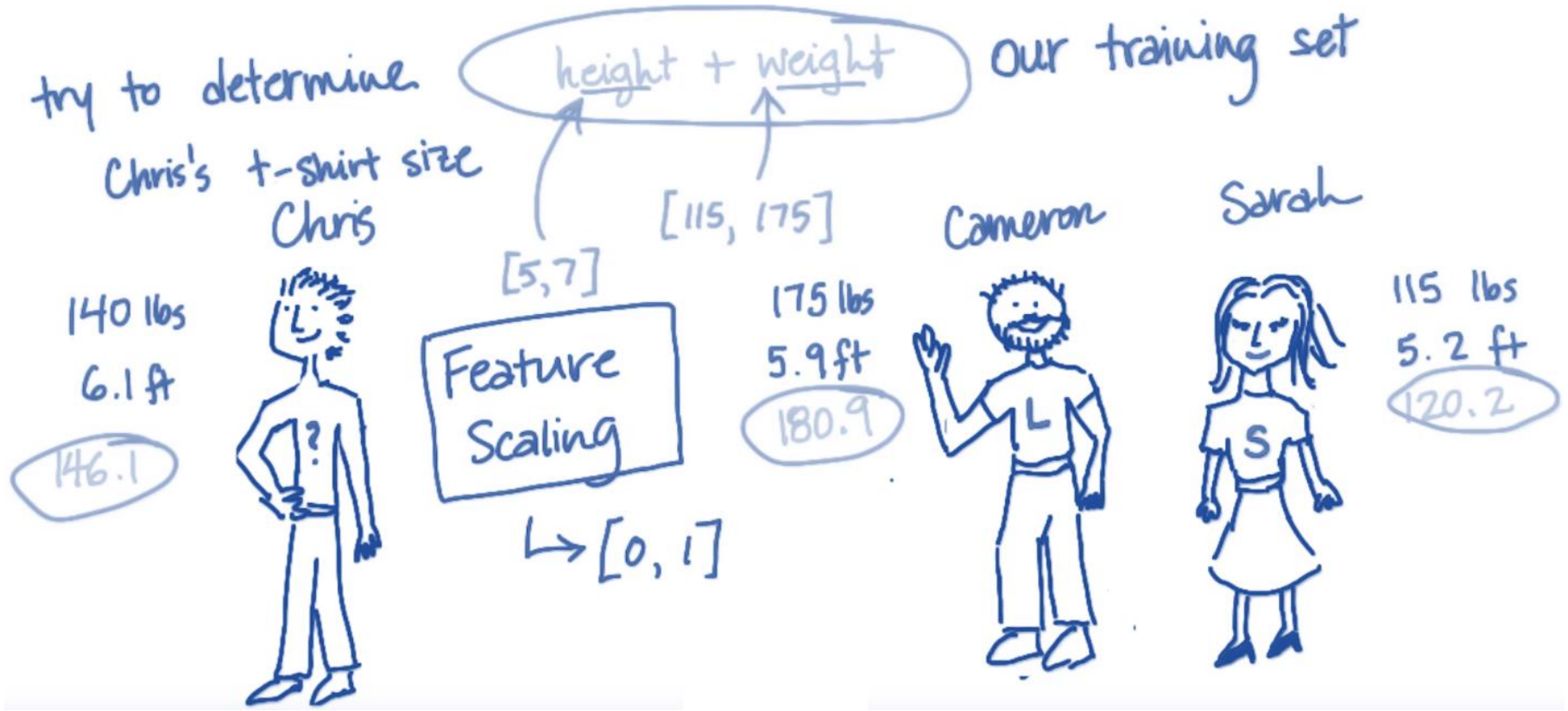
115 lbs

5.2 ft

120.2

Who is Chris closer to in height + weight

- ○ Cameron (large shirt)
- ✓ Sarah (small shirt)


try to determine
Chris's t-shirt size
height + weight
our training set
Chris
[115, 175]
Cameron
Sarah
[5,7]
140 lbs
6.1 ft
146.1
Feature Scaling
↳ [0, 1]
175 lbs
5.9 ft
180.9
115 lbs
5.2 ft
120.2
L
S

Feature Scaling Formula

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

info taken from old feature(s)

x': new (rescaled) feature

Quiz

old weights: [115, 140, 175]

what is x_{min}?

what is x_{max}?

what is x'_{140}?

Write a min/max rescaler that transforms features to have range $[0, 1]$ according to the formula

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

6.3.1.1. Scaling features to a range

An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size. This can be achieved using `MinMaxScaler` or `MaxAbsScaler`, respectively.

The motivation to use this scaling include robustness to very small standard deviations of features and preserving zero entries in sparse data.

Here is an example to scale a toy data matrix to the `[0, 1]` range:

```
>>> X_train = np.array([[ 1., -1.,  2.],
...                     [ 2.,  0.,  0.],
...                     [ 0.,  1., -1.]])
...
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[0.5       , 0.        , 1.        ],
       [1.        , 0.5       , 0.33333333],
       [0.        , 1.        , 0.        ]])
```

The same instance of the transformer can then be applied to some new test data unseen during the fit call: the same scaling and shifting operations will be applied to be consistent with the transformation performed on the train data:

```
>>> X_test = np.array([[-3., -1.,  4.]])
>>> X_test_minmax = min_max_scaler.transform(X_test)
>>> X_test_minmax
array([[-1.5       ,  0.        ,  1.66666667]])
```

It is possible to introspect the scaler attributes to find about the exact nature of the transformation learned on the training data:

```
>>> min_max_scaler.scale_
array([0.5       , 0.5       , 0.33...])

>>> min_max_scaler.min_
array([0.        , 0.5       , 0.33...])
```

If `MinMaxScaler` is given an explicit `feature_range=(min, max)` the full formula is:

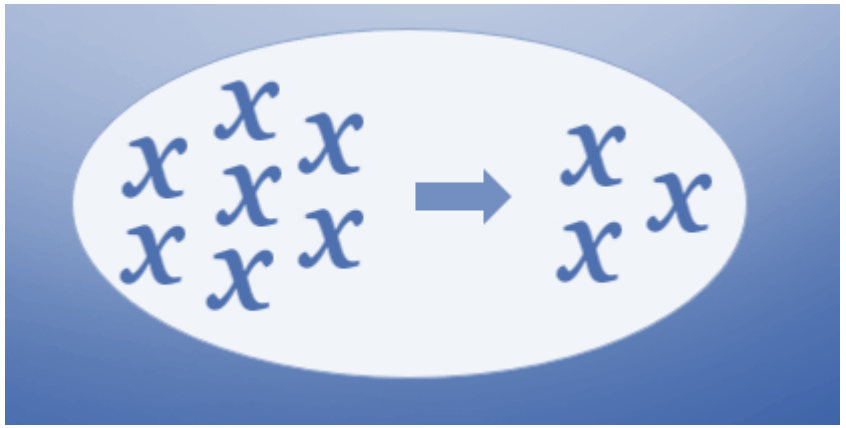
```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X_scaled = X_std * (max - min) + min
```

`MaxAbsScaler` works in a very similar fashion, but scales in a way that the training data lies within the range `[-1, 1]` by dividing

```
>>> from sklearn.preprocessing import MinMaxScaler
>>> import numpy
>>> weights = numpy.array([[115], [140], [175]])
>>> scaler = MinMaxScaler()
>>> rescaled_weight = scaler.fit_transform(weights)
/Users/anaconda/lib/python2.7/site-packages/sklearn/utils/validation.py:278: UserWarning: Mi
nMaxScaler assumes floating point values as input, got int64
  "got %s" % (estimator, X.dtype))
>>> weights = numpy.array([[115.], [140.], [175.]])
>>> rescaled_weight = scaler.fit_transform(weights)
>>> rescaled_weight
array([[ 0.        ],
       [ 0.41666667],
       [ 1.        ]])
>>>
```

Feature Selection

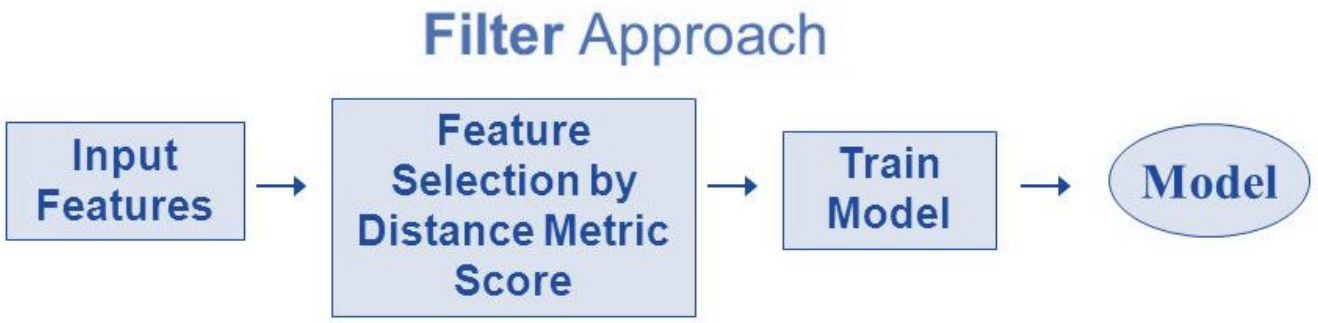


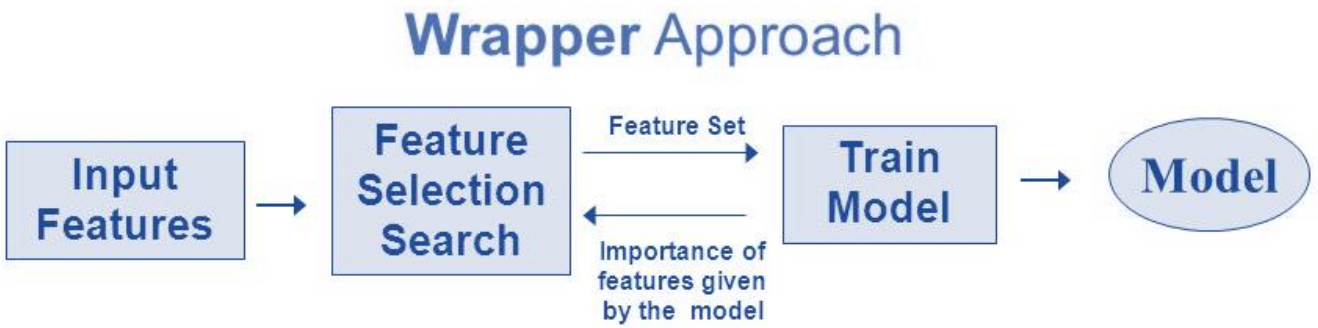
Problems due to unnecessary features

- Unnecessary resource allocation for these features.
- These features act as a noise for ML model
- The model takes more time to get trained.

the importance of feature selection

- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces Overfitting.





Filter methods	Wrapper methods	Embedded methods
Generic set of methods which do not incorporate a **specific machine learning algorithm.**	Evaluates on a **specific machine learning algorithm** to find optimal features.	Embeds (fix) features during **model building process.** Feature selection is done by observing each iteration of model training phase.
Much **faster** compared to Wrapper methods in terms of time complexity	**High computation time** for a dataset with many features	Sits **between Filter methods and Wrapper methods** in terms of time complexity
Less prone to **over-fitting**	High chances of **over-fitting** because it involves training of machine learning models with different combination of features	Generally used to reduce **over-fitting** by **penalizing** the coefficients of a model being too large.
Examples – **Correlation, Chi-Square test, ANOVA, Information gain** etc.	Examples - **Forward Selection, Backward elimination, Stepwise selection** etc.	Examples - **LASSO, Elastic Net, Ridge Regression** etc.

Important: you should include the feature selection step before feeding the data to the model for training especially when you are using accuracy estimation methods such as cross-validation. This ensures that feature selection is performed on the data fold right before the model is trained.

Principal Component Analysis

Customer Name	Customer Age	Customer Income	Product Name	Product Price	Sale Status
Ron	55	$45,000	Toothpaste	4.29	1
Tiffany	72	$15,000	Shampoo	5.99	0
Jennifer	47	$65,000	Hair Color	8.99	1

Key factors that influence customer purchasing behavior

Probabilities that products will be purchased based on the key influencing factors

Principal Components

Uncorrelated features that embody a dataset's important information (its "variance") with the redundancy, noise, and outliers stripped out

Using Factors and Components

- Both factors and components represent what is left of a dataset after information redundancy and noise is stripped out
- Use them as input variables for machine learning algorithms to generate predictions from these compressed representations of your data

PCA Use Cases

Decoding what components to keep

- The *explained variance ratio* tells us how much information is compressed into the first few components
- When deciding how many components to keep, look at the percent of cumulative variance. Make sure to retain at least 70% of the dataset's original information

Explained Variance Ratio

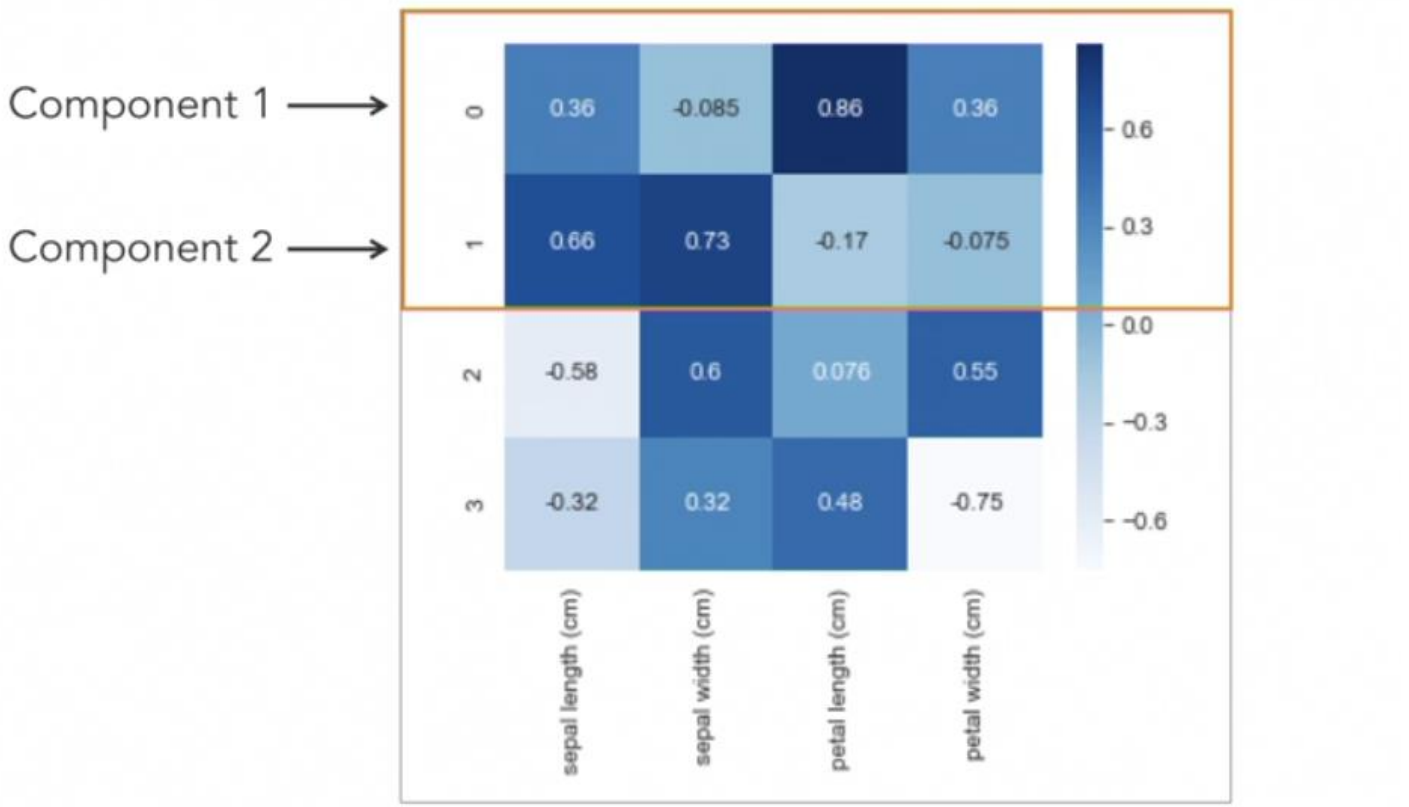
```
array([ 0.92461621,  0.05301557,  0.01718514,  0.00518309])
```

```
pca.explained_variance_ratio_.sum()
```

```
1.0
```

Cumulative Variance

PCA Results


Component 1
Component 2
0
1
2
3
0.36
-0.085
0.86
0.36
0.66
0.73
-0.17
-0.075
-0.58
0.6
0.076
0.55
-0.32
0.32
0.48
-0.75
0.6
0.3
0.0
−0.3
−0.6
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)

PCA Review

→ systematized way to transform input features into principal components

→ use principal components as new features

→ PCs are directions in data that maximize variance (minimize information loss) when you project/compress down onto them

→ more variance of data along a PC, higher that PC is ranked

→ most variance/most information → first PC
second-most variance (without overlapping w/ first PC) → second PC

→ max no. of PCs = no. of input features

When to use PCA

→ latent features driving the patterns in data

→ dimensionality reduction
- → visualize high-dimensional data
- → reduce noise
- → make other algorithms (regression, classification) work better b/c fewer inputs

Linear Regression

Linear Regression

Linear regression is a statistical machine learning method you can use to quantify, and make predictions based on, relationships between numerical variables.

- Simple linear regression
- Multiple linear regression

Sales Forecasting

Resource Consumption Forecasting

Supply Cost Forecasting

Telecom Services Lifecycle Forecasting

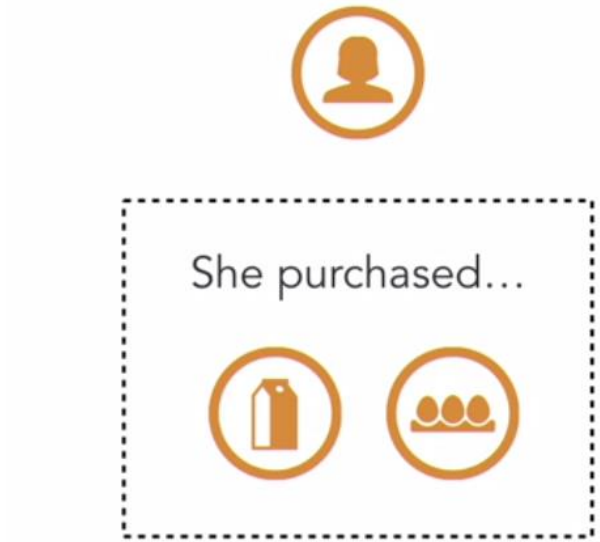
Linear Regression Assumptions

- All variables are continuous numeric, not categorical
- Data is free of missing values and outliers
- There's a linear relationship between predictors and predictant
- All predictors are independent of each other
- Residuals (or prediction errors) are normally distributed

Association Rules Models with Apriori

Association Rule Mining

Association rule mining is a process that deploys pattern recognition to identify and quantify relationships between different, yet related items.



A Simple Association Rules Use Case

How likely is it that she will purchase bread too?

- Popular use case: product placement optimization at both brick and mortar and ecommerce stores

A Simple Association Rules Use Case

Say, for example, eggs and bread are frequently purchased together. With that finding, you can increase sales by:

- Advertising to buyers of either eggs or butter in order to increase that person's propensity to purchase the (paired) other product
- Offer discounts on both eggs and butter if the customer buys both of them in one purchase

Expressing an Association Rule

Association rule:

- "If item eggs are purchased, then the possibility of buying is bread is ____"

Can also be represented as:

- {eggs} → {bread}

Advantages of Association Rules

- Fast
- Works with small data
- Few (if any) feature engineering requirement

Feature Engineering

The term *feature engineering* refers to the process of engineering data into a predictive feature that fits the requirements (and/or improves the performance) of a machine learning model.

Three Ways to Measure Association

1. Support
2. Confidence
3. Lift

Illustrating with Bread and Eggs

Scenario: 5,000 total transactions in a supermarket

- A = Bread purchases = 500 transactions
- C = Eggs purchases = 350 transactions
- (A → C) Both bread and eggs purchased = 150 transactions

Support

Support is the relative frequency of an item within a dataset.
Support for an item can be calculated as: $\text{support}(A \rightarrow C) = \text{support}(A \cup C)$

Example:

- *support(bread) = (transactions containing bread)/(total number of transactions)*
- *support(bread) = 500/5000 = 0.1*

Confidence

Confidence is the probability of seeing the consequent item (a "then" term) within data, given that the data also contains the antecedent (the "if" term) item.

In other words, confidence tells you:

THEN How likely it is for 1 item to be purchased given that,

IF another item is purchased.

Confidence and if-then Statements

Confidence determines how many if-then statements are found to be true within a dataset.

$$\text{confidence}(A \rightarrow C) = \frac{\text{support}(A \rightarrow C)}{\text{support}(A)}$$

** where A is the antecedent and C is the consequent

Confidence with Bread and Eggs

$$\text{confidence}(A \rightarrow C) = \frac{\text{support}(A \rightarrow C)}{\text{support}(A)}$$

$$\text{confidence(bread} \rightarrow \text{eggs)} = \frac{(150/5000)}{(500/5000)} = 0.3 = 30\%$$

So, there is a 30% likelihood that eggs will be bought if bread is purchased

Lift

Lift is a metric that measures how much more often the antecedent and consequent occur together rather than them occurring independently.

$$\text{lift}(A \to C) = \frac{\text{confidence}(A \to C)}{\text{support}(C)}$$

Lift Scores

- Lift score > 1: A is highly associated with C. If A is purchased, it is likely that C will also be purchased
- Lift score < 1: If A is purchased, it is unlikely that C will be purchased
- Lift score = 1: Indicates that there is no association between items A and C

Lift with Bread and Eggs

$$\text{lift}(A \to C) = \frac{\text{confidence}(A \to C)}{\text{support}(C)}$$

$$\text{lift}(\text{bread} \to \text{eggs}) = \frac{0.3}{(350/5000)} = 4.28$$

With a lift of 4.28, your rule would be "If a customer buys bread, then they're likely to also buy eggs."

Where Apriori Fits In

- The Apriori algorithm is the algorithm that you use to implement association rule mining over structured data.

K-Nearest Neighbors

K-Nearest Neighbor Classification

A supervised classifier that memorizes observations from within a test set to predict classification labels for new, unlabeled observations

KNN makes predictions based on how similar training observations are to the new, incoming observations.

The more similar the observation values, the more likely they will be classified with the same label.

Stock Price Prediction

Predictive Trip Planning

Credit Risk Analysis

Recommendation Systems

KNN Model Assumptions

- Dataset has little noise
- Dataset is labeled
- Dataset only contains relevant features
- Dataset has distinguishable subgroups
- Avoid using KNN on large datasets
  It will probably take a long time

	Precision	Recall	fl-score	Support
0	0.80	1.00	0.89	4
1	1.00	0.67	0.80	3
accuracy			0.86	7
macro avg	0.90	0.83	0.84	7
weighted avg	0.89	0.86	0.85	7

Recall: a measure of your model's completeness

- Of all your points that were labeled 1, only 67% of the results that were returned were truly relevant
- Of the entire dataset, 83% of the results that were returned were truly relevant

High precision + Low recall = Few results returned, but many of the label predictions that are returned are correct

Naïve Bayes Classification

Naïve Bayes Classifiers

Naïve Bayes is a machine learning method you can use to predict the likelihood that an event will occur given evidence that's present in your data.

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

Three Types of Naïve Bayes Model

- Multinomial
- Bernoulli
- Gaussian

Spam Detection

Customer Classification

Credit Risk Protection

Health Risk Protection

Naïve Bayes Assumptions

Predictors are independent of each other.

A priori assumption: the assumption the past conditions still hold true; when we make predictions from historical values we will get incorrect results if present circumstances have changed.

- All regression models maintain a priori assumption as well

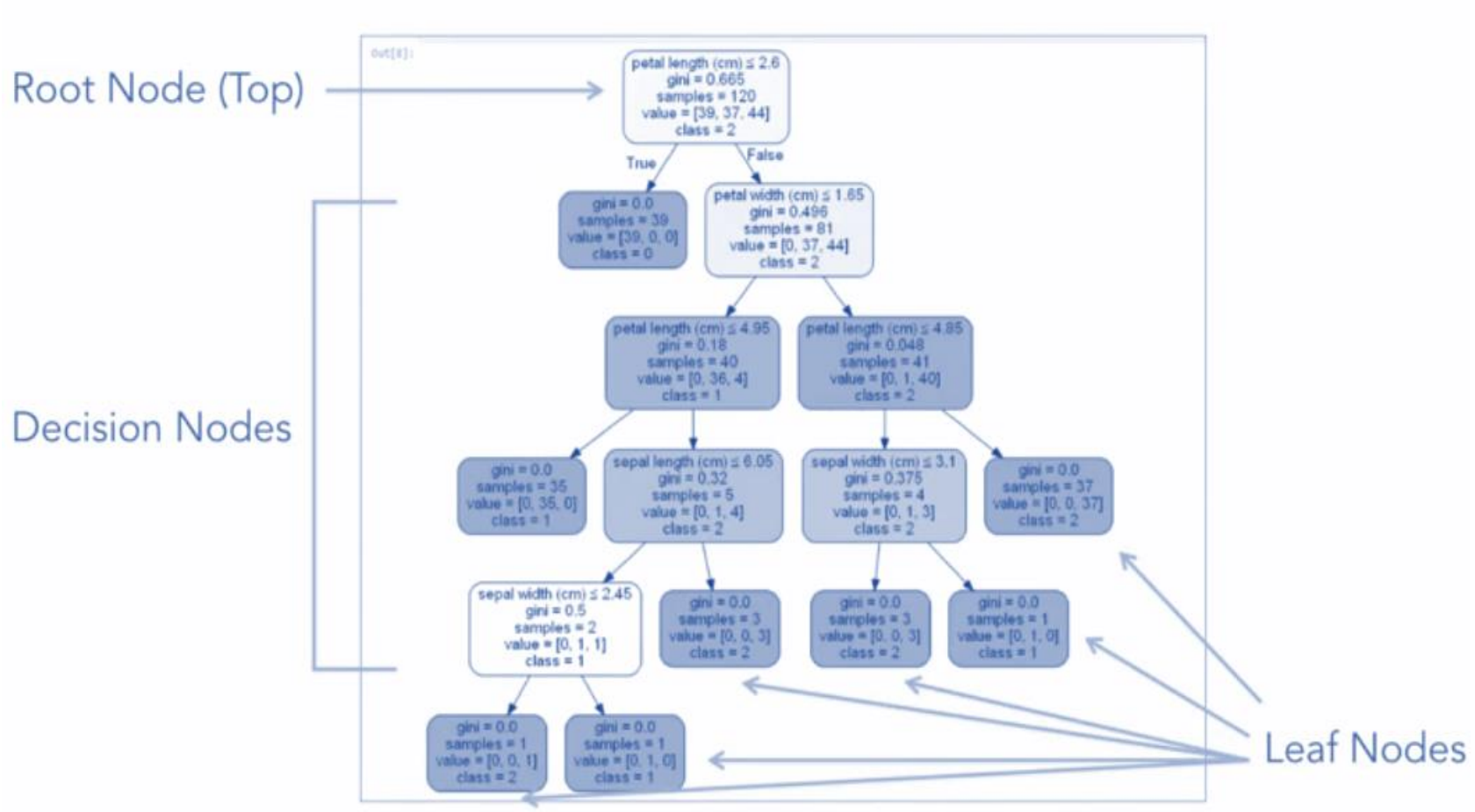
Decision Trees

Decision Tree

A decision tree is a decision-support tool that models decisions in order to predict probable outcomes of those decisions.

Decision Trees Have Three Types of Nodes

- Root node
- Decision nodes
- Leaf nodes



Decision Tree Algorithms

A class of supervised machine learning methods that are useful for making predictions from nonlinear data

Decision tree algorithms are an appropriate fit for:

- *Continuous* input and/or output variables
- *Categorical* input and/or output variables

Two Types of Decision Trees

- Categorical variable decision tree (or classification trees): when you use a decision tree to predict for a categorical target variable
- Continuous variable decision tree (or regression trees): when you use a decision tree to predict for a continuous target variable

Main Benefits of Decision Trees Algorithms

- Decision trees are nonlinear models
- Decision trees are very easy to interpret
- Trees can be easily represented graphically, helping in interpretation
- Decision trees require less data preparation

Assumptions of Decision Trees Algorithms

- Root node = Entire training set
- Predictive features are either categorical, or (if continuous) they're binned prior to model deployment
- Rows in the dataset have a recursive distribution based on the values of attributes

How Decision Trees Algorithms Work

1. Deploy recursive binary splitting to stratify or segment the predictor space into a number of simple, nonoverlapping regions.
2. Make a prediction for a given observation by using the mean or the mode of the training observations in the region to which it belongs.
3. Use the set of splitting rules to summarize the tree.

Recursive Binary Splitting

Recursive binary splitting is the process that's used to segment a predictor space into regions in order to create a binary decision tree.

How Recursive Binary Splitting Works

At every stage we split the region into two, and we do this by the following criteria:

- In regression trees: use the SSE to calculate the loss function, thus identifying the best split
- In classification trees: use the Gini index to calculate the loss function, thus identifying the best split

Characteristics of Recursive Binary Splitting

- Top-down
- Greedy

Disadvantages of Decision Tree Algorithms

- Very non-robust
- Sensitive to training data
- Globally optimum tree not guaranteed

Using Decision Trees for Regression

Regression trees are appropriate when:

- Target is a continuous variable
- Linear relationship between features and target

Output values from terminal nodes represent the mean response:

- Values of new data points will be predicted from that mean

Recursive Splitting in Regression Trees

At every stage in a regression tree, the region is split into two according to sum of squares error:

- The model begins with the entire data set, S, and searches every distinct value of every predictor to find the predictor and split value that partitions the data into two groups (S1 and S2), such that the overall sums of squares error are minimized:

$$\text{SSE} = \sum_{i \in S_1} (y_i - \overline{y}_1)^2 + \sum_{i \in S_2} (y_i - \overline{y}_2)^2$$

Using Decision Trees for Classification

Classification trees are appropriate when:

- Target is a binary, categorical variable

Output values from terminal nodes represent the mode response:

- Values of new data points will be predicted from that mode

Recursive Splitting in Classification Trees

At every stage in a classification tree, the region is split into two according to a user-defined metric, for example:

- The Gini index (G) is a measure of total variance across the K classes; it measures the probability of misclassification

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

- *G* takes on a small value if all of the p_{mk}'s are close to zero or one
- "Measure of node purity": where a small *G* value indicates that a node contains predominantly observations from a single class

Tree Pruning

Tree pruning is the process that's used to overcome model overfitting by removing subnodes of a decision tree (that is, replacing a whole subtree by a leaf node).

Two Popular Tree Pruning Methods

- Hold-out test: Fastest, simplest pruning method
- Cost-complexity pruning

Why Prune a Decision Tree

Why tree pruning is necessary:

- Deep tree = Model overfitting = Bad performance
- If expected error rate (in subtree) > Single leaf