Machine Learning in Practice

Data Boot Camp

Lesson 19.2

Class Objectives

By the end of this lesson, you will be able to:

- Segment data.
- Prepare data for complex algorithms.
- Explain the importance of preprocessing data for unsupervised learning.
- Transform categorical variables into a numerical representation by using Pandas.
- Scale data by using the `StandardScaler` module from `scikit-learn`.


WELCOME


Recap

Recap

In the previous lesson, you learned:

- How to recognize the differences between supervised and unsupervised machine learning.
- What clustering is and how to use it in data science.
- How to apply the K-means algorithm to identify clusters in several datasets.
- How to determine the optimal number of clusters for a dataset by using the elbow method.

How is machine learning used in data analytics?

Machine Learning in Data

Examples include:

- Finding outliers
- Classifying fraudulent credit card charges
- Predicting customer preferences
- Algorithmic trading

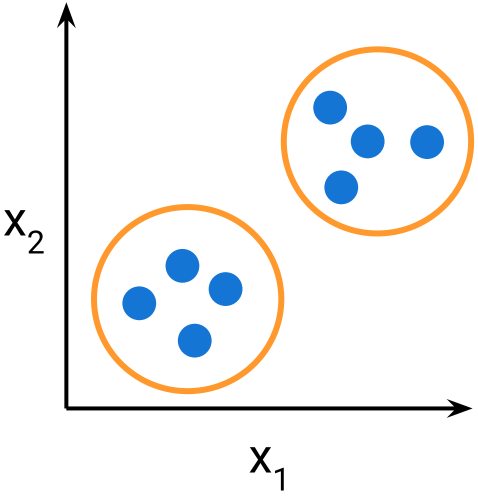
What is the difference between unsupervised and supervised learning?

Supervised Learning vs. Unsupervised Learning

The main distinction between the two approaches is the use of labeled datasets.

Supervised Learning

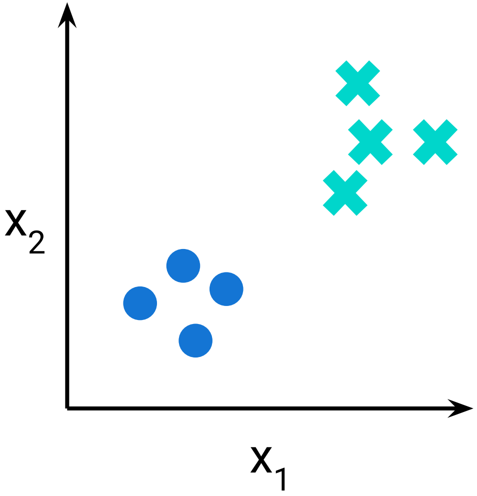
Uses labeled input and output data.



vs.

Unsupervised Learning

Only has labeled input data.



Instructor Demonstration

The K-Means Algorithm

Questions?

Activity: Warm-Up

In this activity, you will review some K-means concepts and code from the previous class.

Suggested Time:

15 Minutes

Time’s Up! Let’s Review.

Questions?

Instructor Demonstration

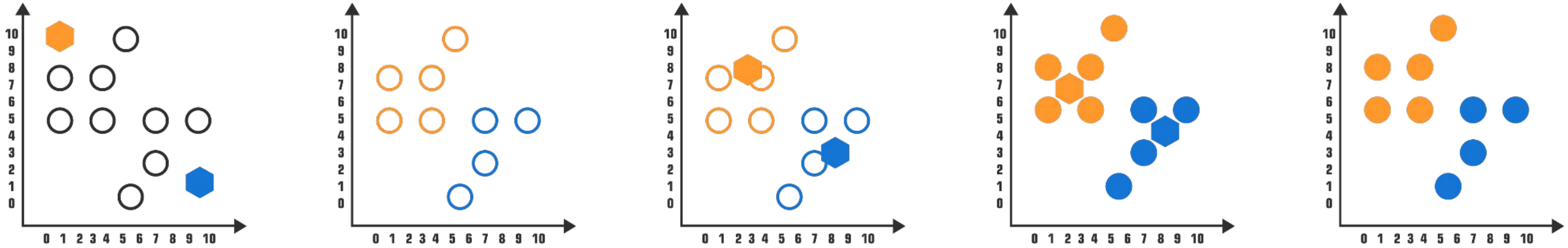
Review Warm-up

Questions?

Scaling Data

Scaling Data with StandardScaler

We can optimize data clustering by selecting the best value for k. The K-means algorithm is useful for grouping and understanding data.



Randomly select K-clusters

Each object assigned to similar centroid randomly

Clusters centers updated depending on renewed cluster mean

Re-assign data points; update cluster centers

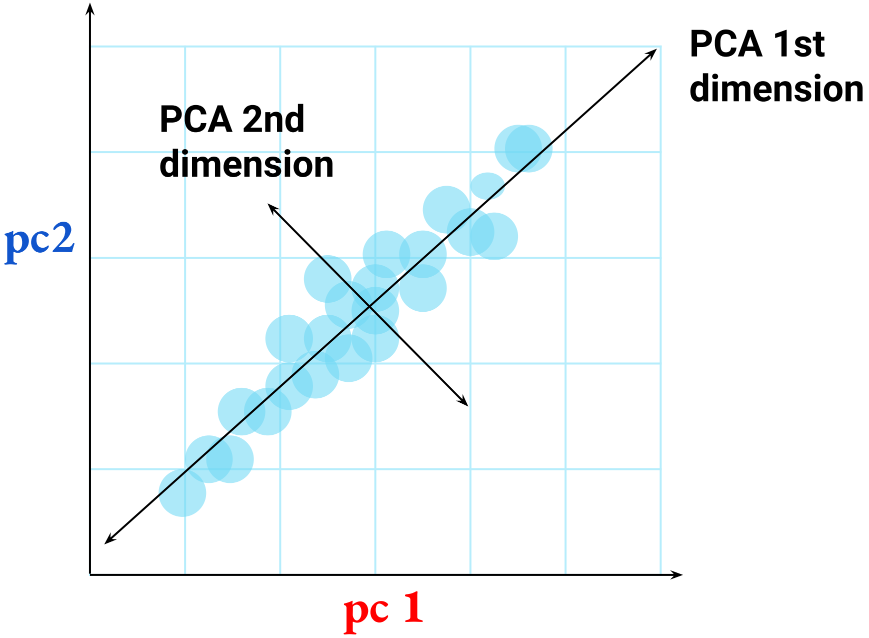
Re-assign data points

We can often enhance and optimize machine learning algorithms by applying Principal Component Analysis, or PCA.

Principal Component Analysis (PCA) is a statistical technique for streamlining the machine learning process when too many factors exist in the data.

Principal Component Analysis (PCA)

PCA reduces the number of factors by transforming a large set of features into a smaller one that contains MOST of the information of the original larger dataset.



Principal Component Analysis (PCA)

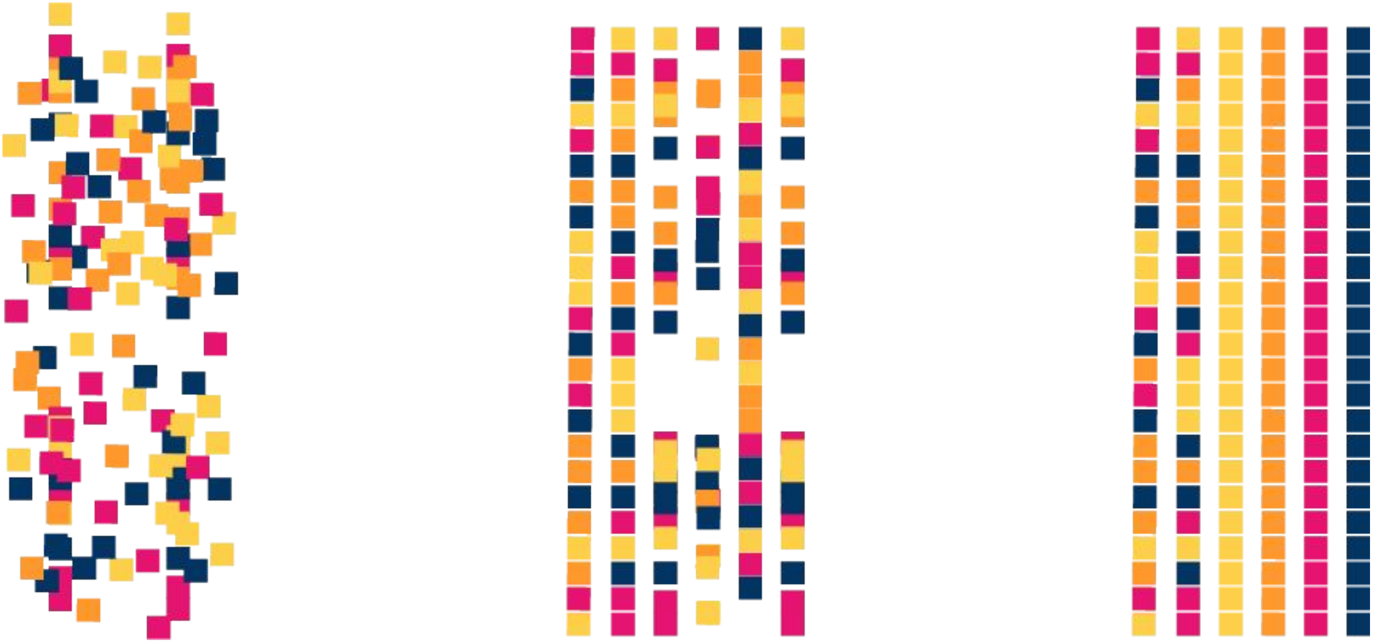
PCA is a dimensionality-reduction method that:

- Looks at all the dimensions (or data columns) in a dataset.
- Analyzes the weight of their contribution to the variance in the dataset.
- Reduces the dimensions to a smaller set that still contains as much of the information (the maximum variance) of the original dataset as possible.

PCA will NOT capture all the information from the original dataset, but it will capture as much as possible to maintain the predictive power and the meaning of the original dimensions.

Principal Component Analysis (PCA)

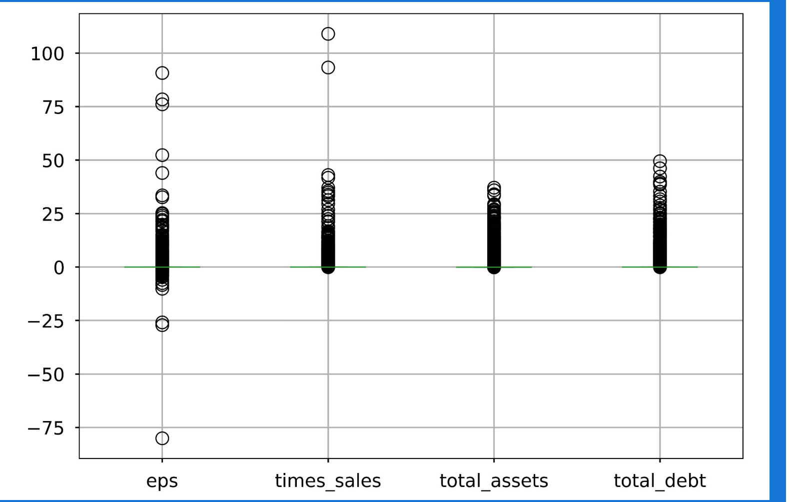
Reducing the number of factors, or **dimensional reduction**, comes at the expense of some accuracy, but the goal is to trade a little accuracy for simplicity.



Standard Scaling

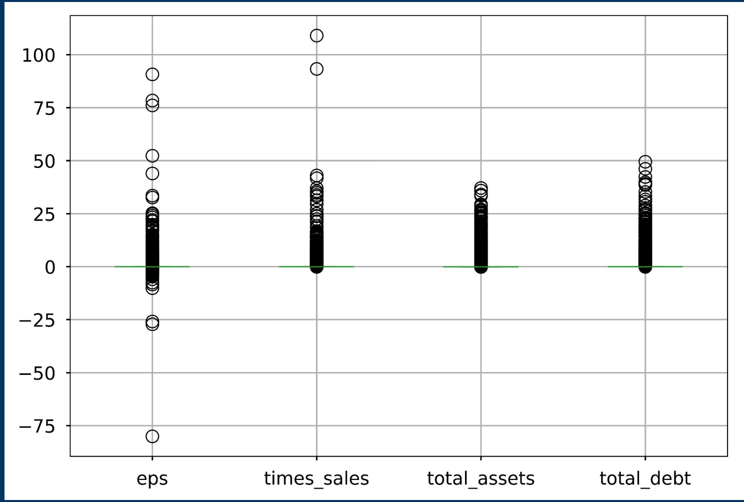
Before

Before using PCA, we'll apply standard scaling to learn how to transform the features of data.



After

After scaling, we'll combine PCA with the K-means algorithm. This will give us a strategy to better handle extremely large datasets.

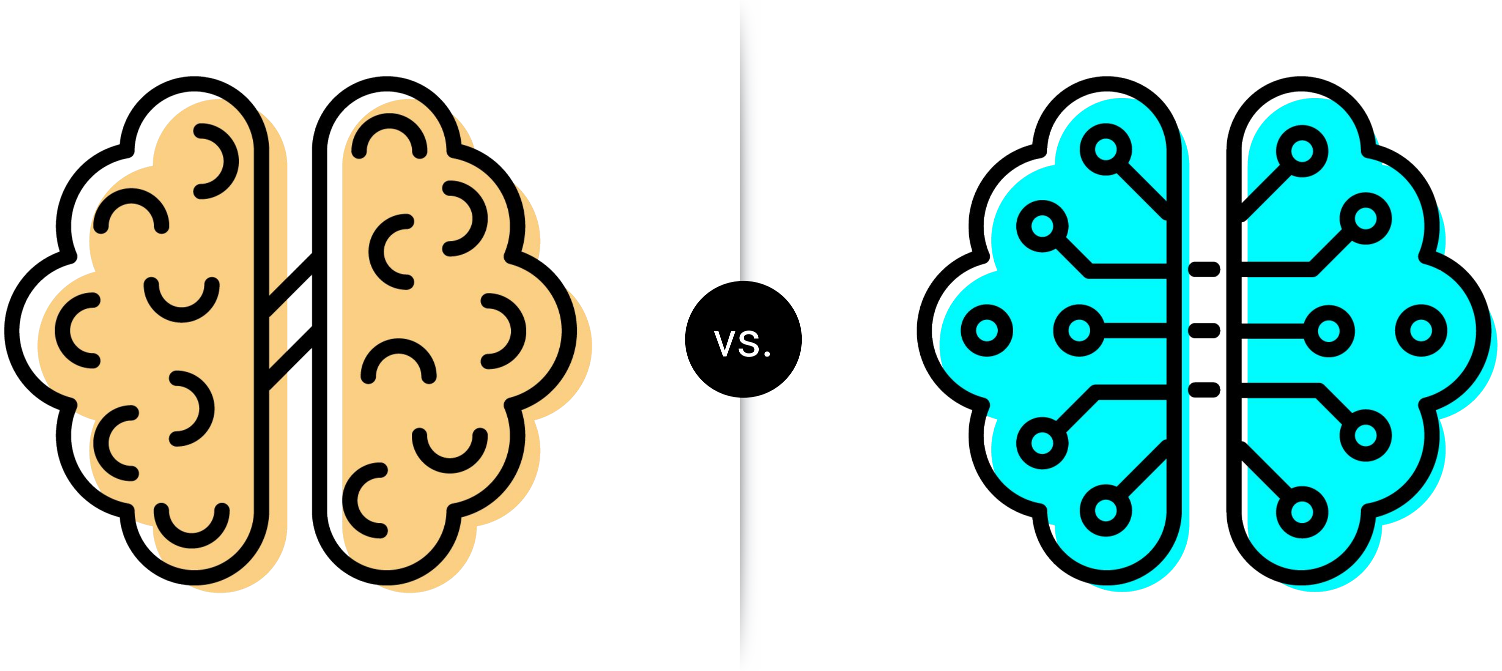


This Week’s Challenge Assignment

You’ll manage data and optimize the clusters with PCA, and then analyze the results. Next, you’ll propose the best number of cryptocurrency clusters for the portfolio.

Scaling Data

Manual preparation of data can be time-consuming. This is especially true if we have several columns in a DataFrame to transform.



Scaling Data

Remember, the K-means algorithm requires all the columns in a DataFrame to have numeric values.

- We should also ensure that the numeric values have the same scale.
- This prevents K-means from putting too much weight on any single variable.

Numeric Data Before Scaling

eps	times_sales	total_assets	total_debt
2.61	63.73	222822.05	46244.82
0.12	17.55	234.42	0.00
7.96	44.14	239.78	15.24
-21.25	109.27	16872.89	0.00
62.48	387.85	156035.77	41128.51

The Same Data After Scaling

eps	times_sales	total_assets	total_debt
-0.0575	-0.0797	-0.1134	-0.0864
-0.0570	0.0795	-0.1136	0.0864
-0.0594	-0.0796	-0.1136	-0.0864
-0.0567	-0.0770	0.1135	-0.0862
0.0484	0.2537	-0.0961	-0.0836

Scaling Data

When we **scale data**, we eliminate the measurement units and scale the numeric values to a similar scale.

We can then compare data of **differing natures**.

Scaling Data

Instead of manually transforming our data, we can use functions from Pandas and the scikit-learn library to simplify our data preparation.

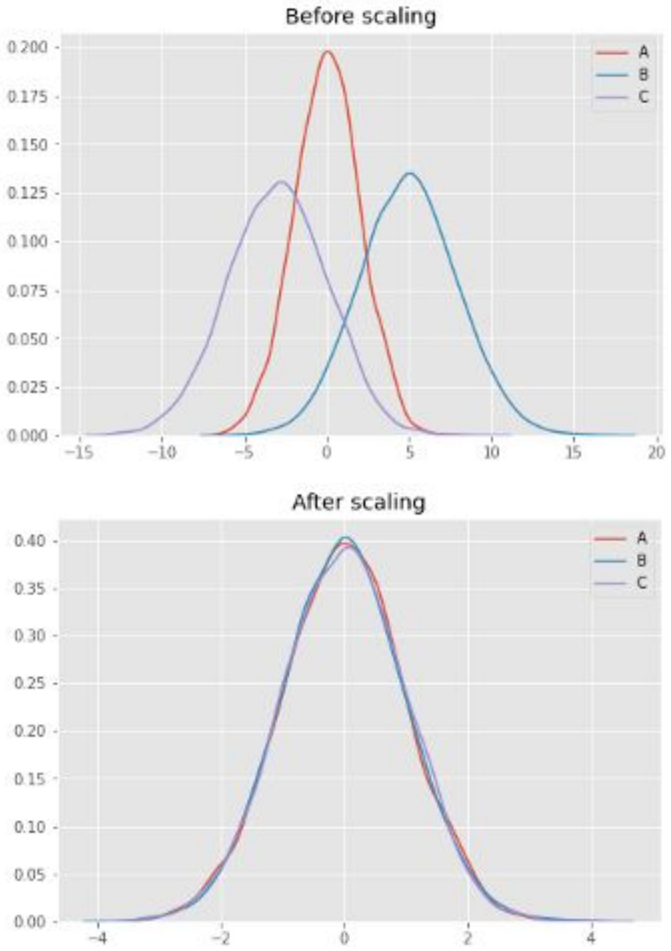
Scaling Data

The most common way to scale data is to apply **standard scaling**, which is a method of centering values around the mean.

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

μ = Mean

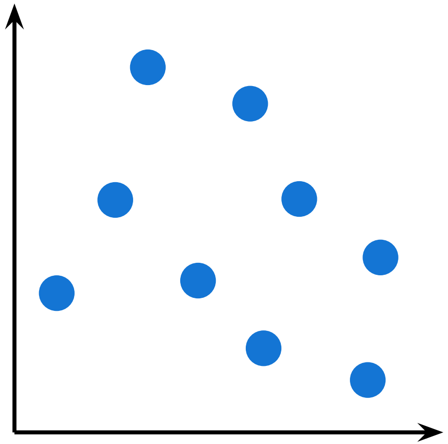
σ = Standard deviation



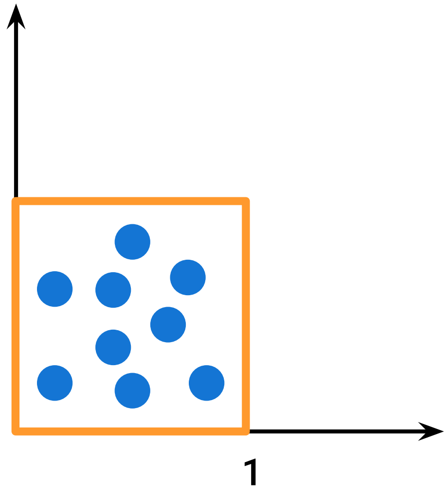
Scaling Data

Data standardization is a common practice in the data preprocessing steps that occur before training a machine learning model.

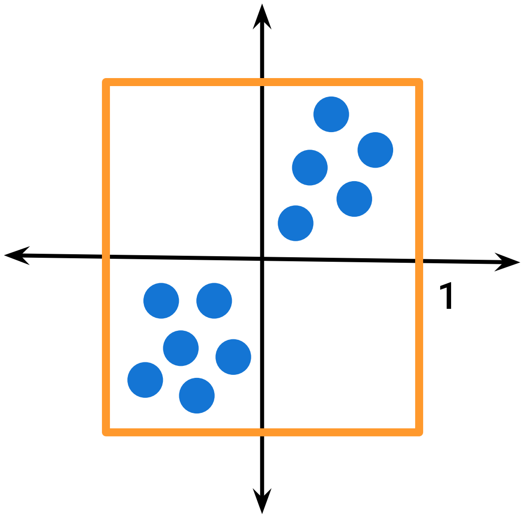
Actual data



After scaling



After standardization



Scaling Data

Let's review one of our credit card spending datasets to illustrate how standard scaling works.

```
# Read in the CSV file and create the Pandas DataFrame
df_shopping = pd.read_csv(
    Path("../Resources/shopping_data.csv")
)

# Review the DataFrame
df_shopping.head()
```

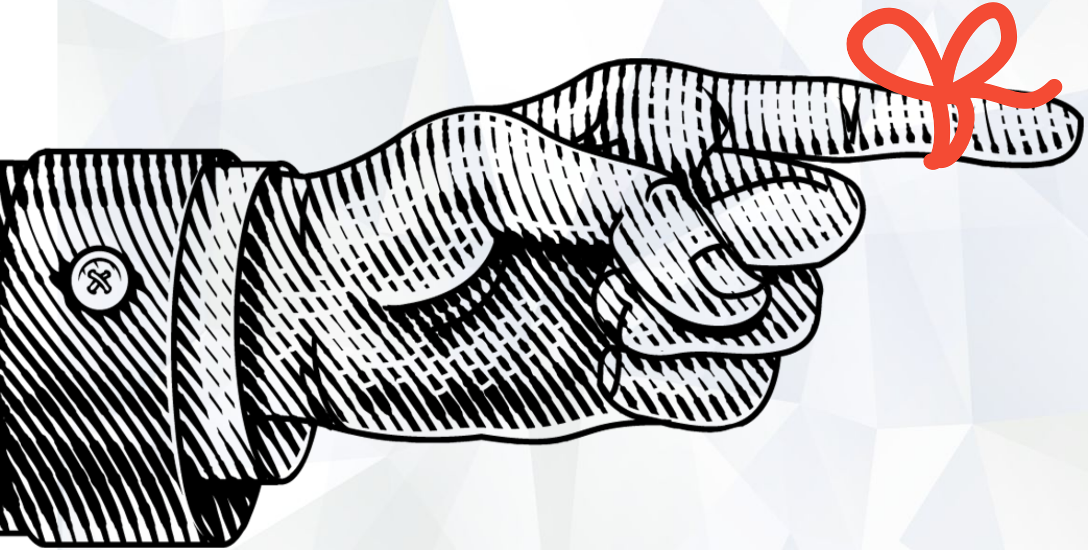
	CustomerID	Card Type	Age	Annual Income	Spending Score
0	1	Credit	19	15000	39
1	2	Credit	21	15000	81
2	3	Debit	20	16000	6
3	4	Debit	23	16000	77
4	5	Debit	31	17000	40

Instructor Demonstration

Applying Standard Scaling

Questions?

Preprocessing Data



Remember,

an important aspect of machine learning is data preparation, or data preprocessing.

Preprocessing Data

We can import a dataset into a Pandas DataFrames, but that doesn't mean all the data is ready for immediate analysis by a machine learning model.

Preprocessing Data

We should consider the following factors when feeding data to a machine learning model:

01

Most machine learning models cannot directly work with data that is in the form of strings or text.

We must encode, or convert, these elements into numeric categories.

02

Machine learning algorithms have trouble learning about data with wildly different scales.

03

Missing values are difficult for machine learning models to navigate.

Preprocessing Data

There is a saying, **"Garbage in, garbage out."**

The data going into a machine learning model must be clean for the predictions coming out of it to be accurate.

Encoding is a preprocessing technique for creating a numeric representation of a categorical variable.

Time to Code

Preprocessing Data

Suggested Time:

20 minutes


Questions?


Break

Activity: Standardizing Stock Data

In this activity, you will use the K-means algorithm to segment customer data for mobile versus in-person banking service ratings.

Suggested Time:

25 Minutes

Time’s Up! Let’s Review.


Questions?

Clustering Complex Data

Clustering Complex Data

Sometimes, complex or unusual datasets might require alternative algorithms for clustering.

In this demonstration, we'll introduce two:

01 Birch

02 Agglomerative clustering

Clustering Complex Data: BIRCH

BIRCH stands for:

Balanced **I**terative **R**educing and **C**lustering using **H**ierarchies

Clustering Complex Data: BIRCH

BIRCH is an unsupervised data mining algorithm that is similar to K-means, with a few differences.

- In particular, BIRCH uses hierarchical clustering.
- This approach may start out with many clusters, but then over the process of learning, BIRCH combines these clusters until there is only the specified number left.
- The inventors of BIRCH designed it for use with extremely large datasets. This is still its main purpose because it tends to be memory-efficient.

Data

Phase 1:
Load data into memory by building a CF tree

Initial CF tree

Phase 2:
Condense data by building a smaller CF tree (optional)

Smaller CF tree

Phase 3:
Global clustering

Good cluster

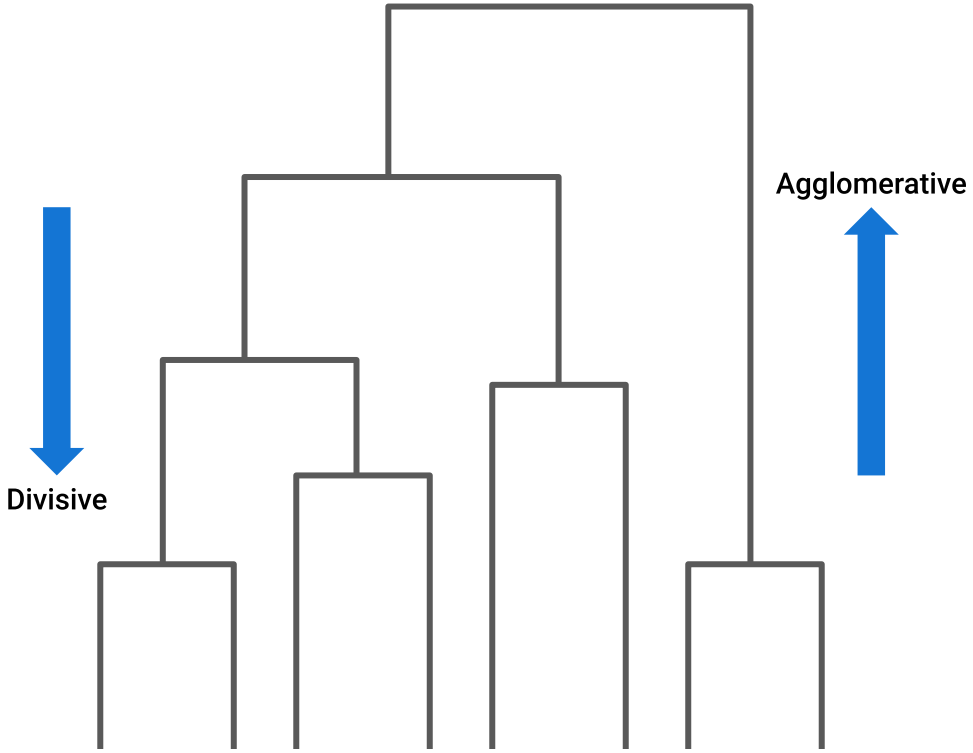
Phase 4:
Cluster refinement (optional)

Better cluster

Clustering Complex Data: Agglomerative Clustering

Agglomerative clustering is like BIRCH.

- Neither one requires you to specify the appropriate cluster count k, unlike K-means.
- While you can specify a specific cluster count with these two approaches, they are flexible enough to divide the data into categories without much input from you.
- Sometimes there isn't a single, catch-all answer when deciding to use one clustering routine over another.
- Instead, data scientists often try multiple algorithms to find out which one appears to work best on their specific data.



We'll try out all three clustering methods (K-means, BIRCH, Agglomerative) on a single dataset and preview the resulting labels.

Instructor Demonstration

Compare and Contrast Alternative Clustering Algorithms

Questions?

Activity: Segmenting Customer Data

In this activity, you will use BIRCH, agglomerative clustering, and the K-Means model to segment a dataset on thousands of consumer credit card holders, courtesy of a Kaggle competition.

Suggested Time:

25 Minutes

Time’s Up! Let’s Review.

Questions?

The End