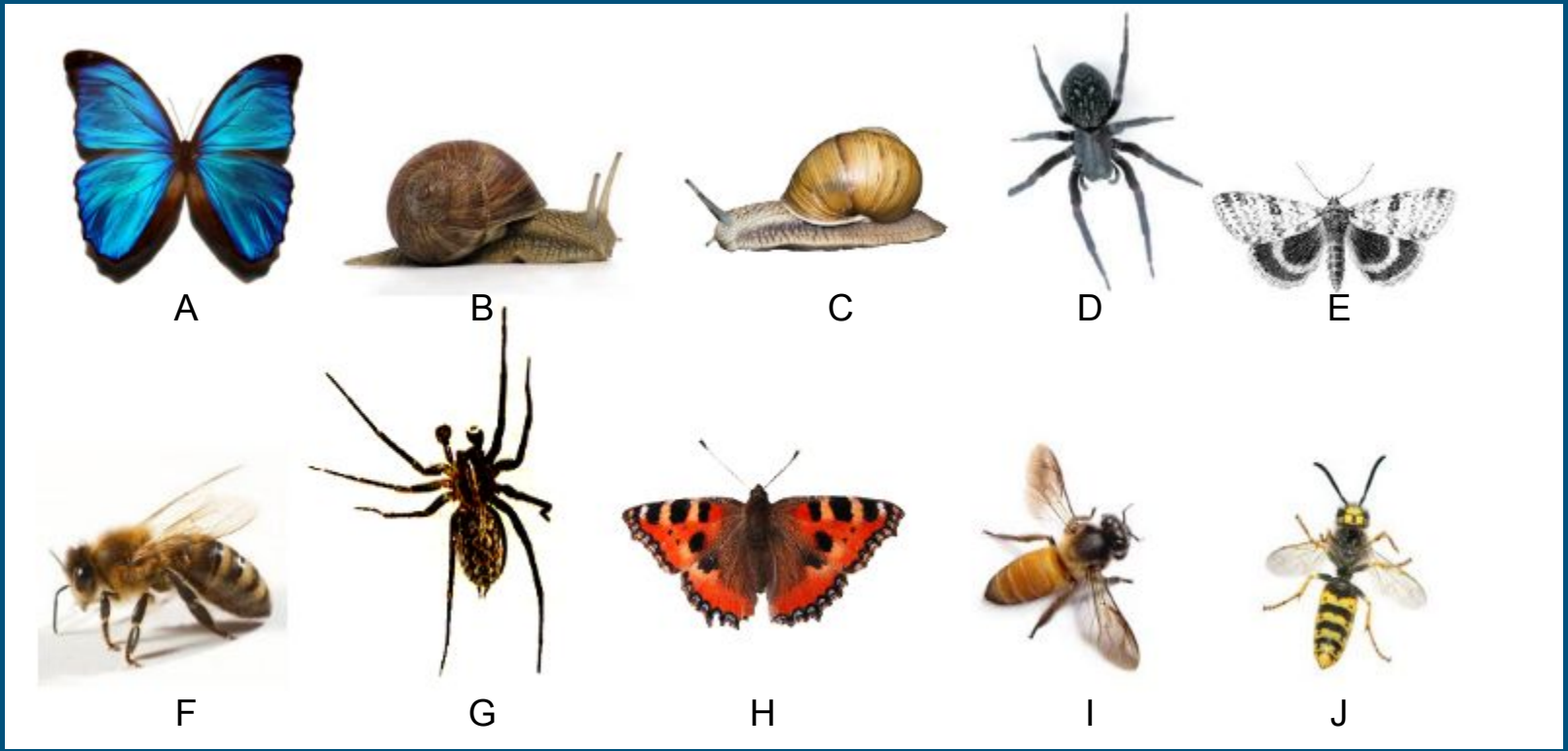
Clustering

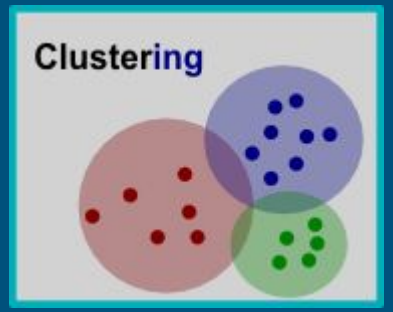
Sam D. Lojacono, Seth Adams, Kiana Herr, David Moste, Joel Bianchi, Maxwell Yearwood

How many different groups of creepy-crawlies do you see?



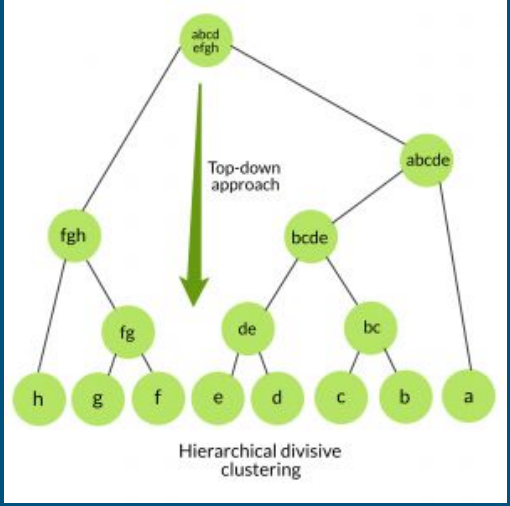
What is Clustering?

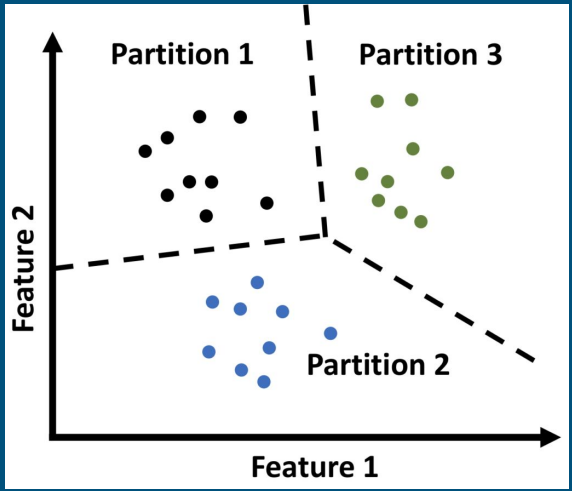
- **Clustering** is the process of dividing a dataset into groups in such a way that data points in the same cluster are similar to each other and different from those in other clusters
- Clustering is used in **machine learning** and **data science** to group similar data points together.
- The goal of clustering is to identify patterns in the data and group them into **meaningful** clusters.



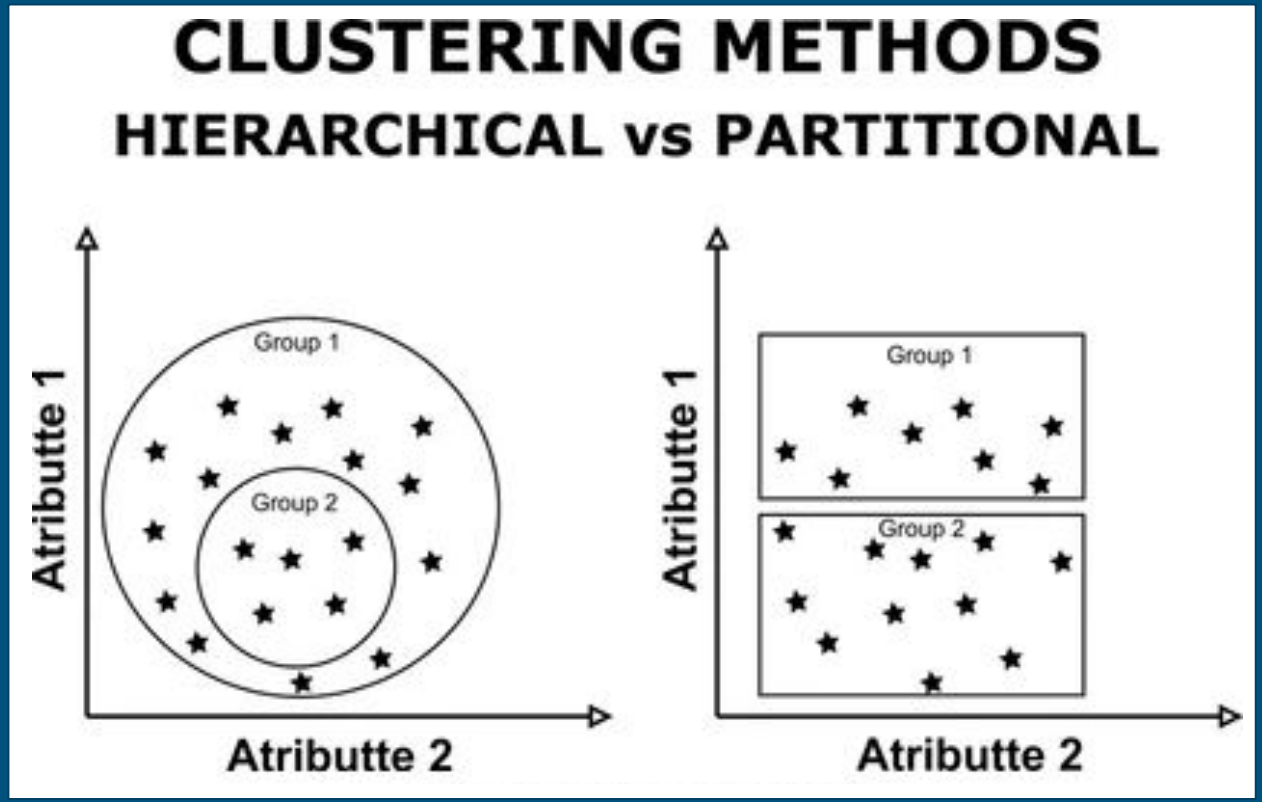
Types of Clustering

- There are two main types of clustering algorithms – hierarchical clustering and partitional clustering.
- **Hierarchical clustering** involves creating a tree-like structure of clusters, where smaller clusters are merged or nestled into larger ones.
- **Partitional clustering** involves dividing the data points into non-overlapping clusters, where each data point belongs to exactly one cluster.

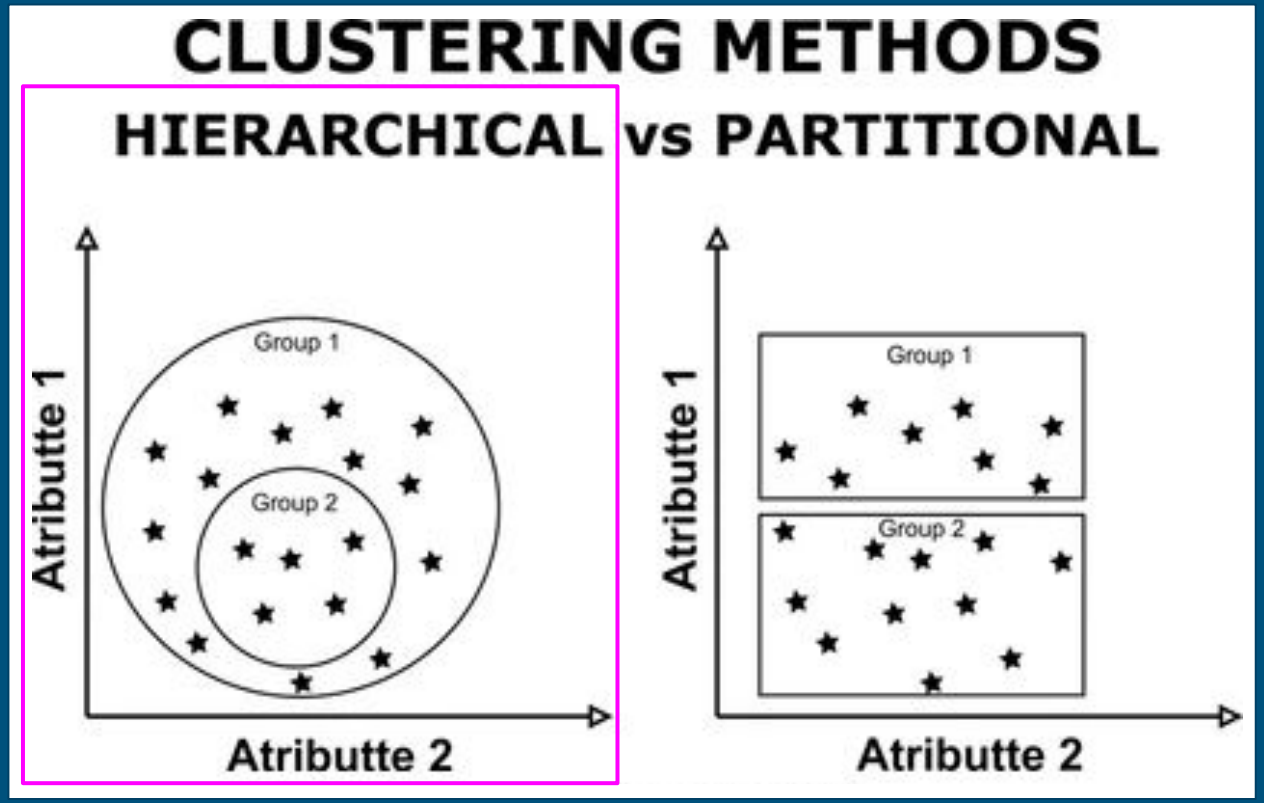




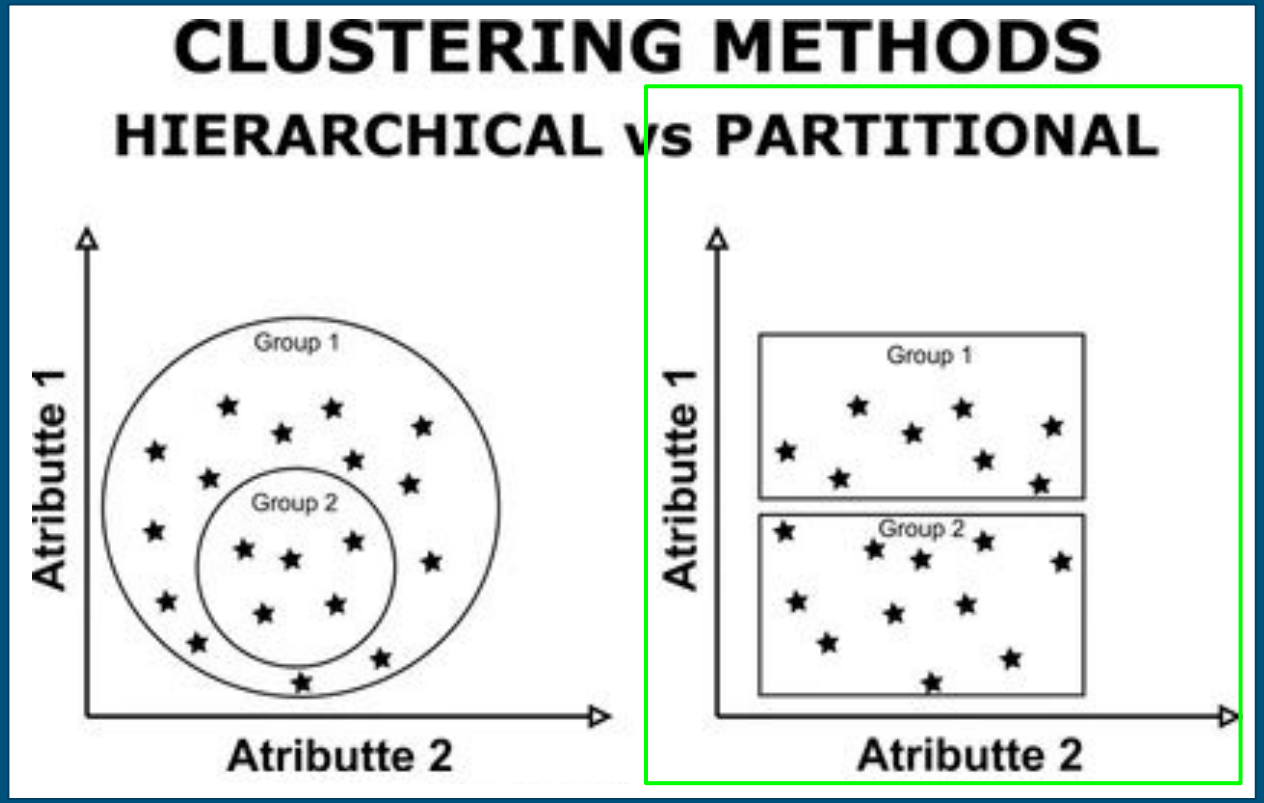
Types of Clustering



Types of Clustering

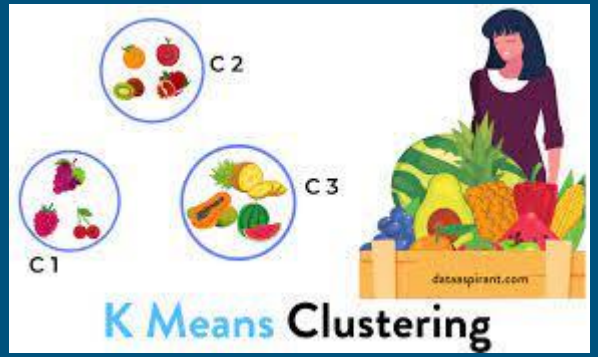


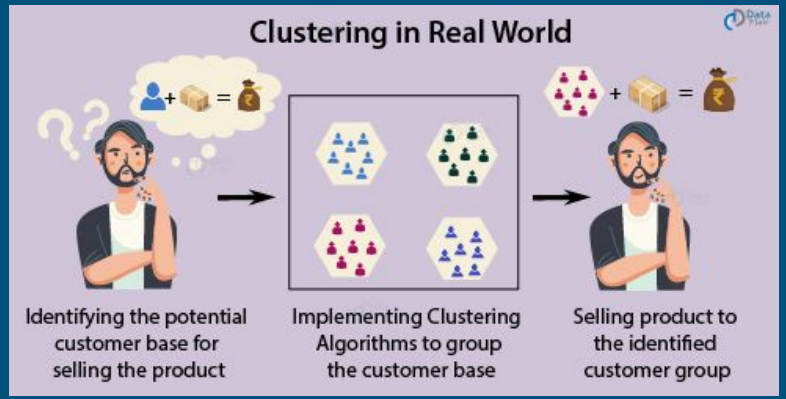
Types of Clustering



Applications Clustering

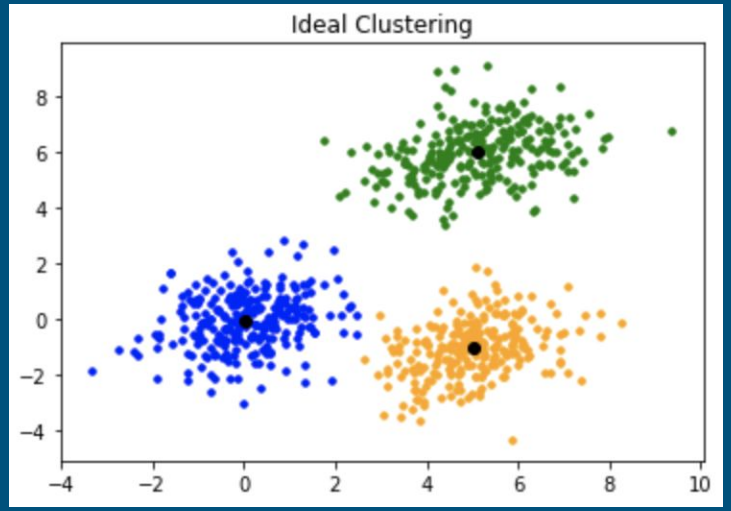
- Image segmentation and compression
- Customer segmentation in marketing
- Document clustering in natural language processing
- Anomaly detection in cybersecurity
- Simple and efficient technique for finding groups of similar observations in large and high-dimensional data sets





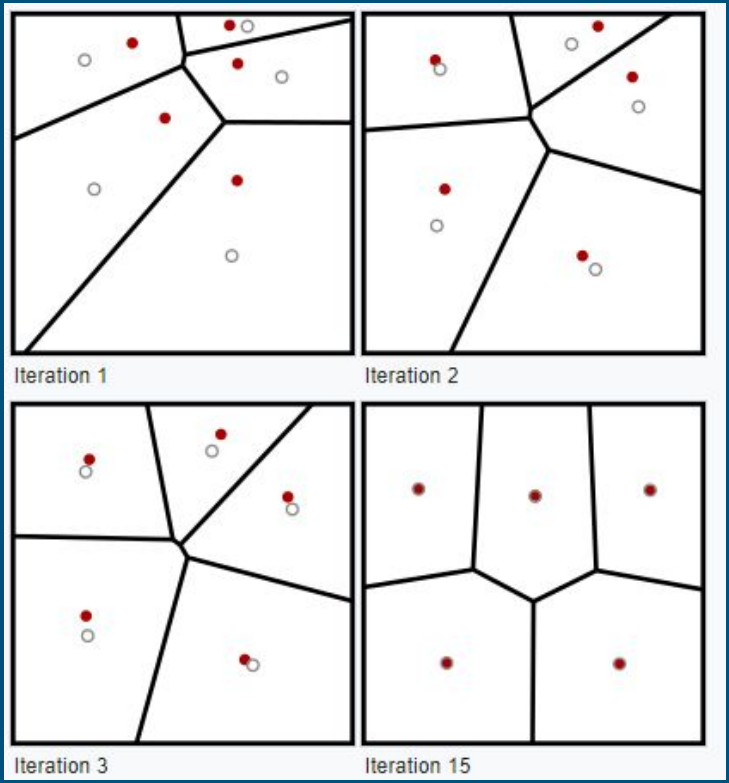
What is k-Means Clustering?

- A method of partitioning observations into clusters
- Clusters are characterized by their centroids, which serve as markers for the members of the cluster
- Each observation belongs to the cluster with the nearest mean (cluster centers or centroid)



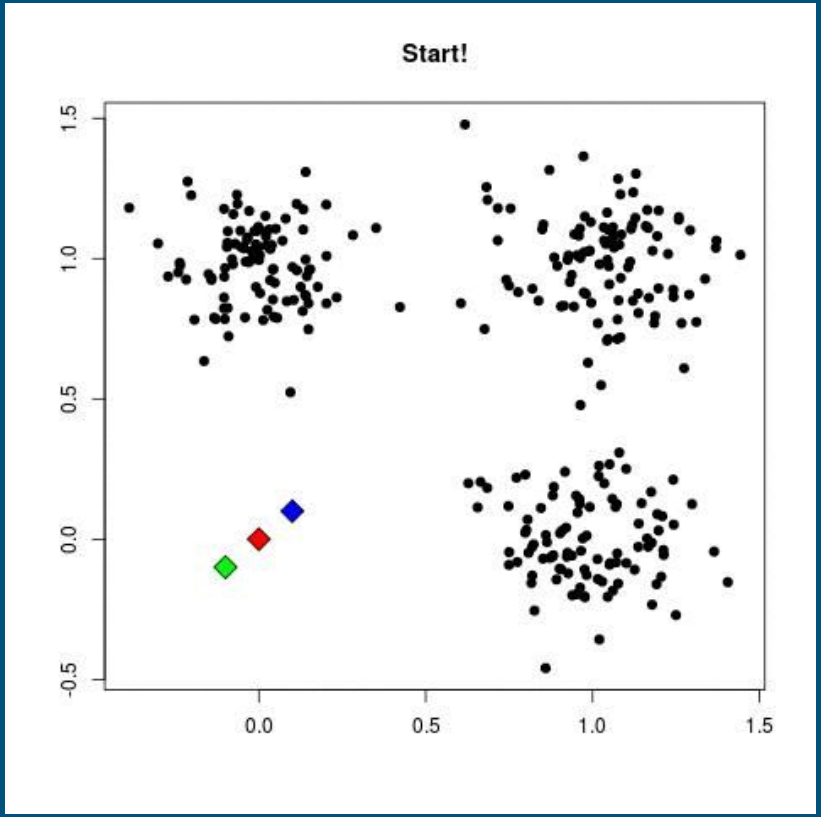
The Iterative Refinement Technique

- The most common algorithm for k-means clustering
- Also known as "Lloyd's algorithm" or "naïve k-means"
- Alternates between assigning data points to the nearest cluster and recomputing the cluster means
- Converges when the centroids no longer change



Steps to the K-means Clustering Algorithm

1. Choose the number of clusters to form (k)
2. Randomly select k observations and assign them as the initial centroids
3. Assign each observation to the nearest centroid using a distance measure
4. Recalculate the centroids of each cluster as the mean of all observations in that cluster
5. Repeat steps 3 and 4 until convergence



Limitations of k-Means Clustering

- Does not guarantee to find the optimum solution
- Determining the optimal number of clusters can be subjective or based on external criteria

“Notebooks”: Popular Data Science Tools

python™

Type Python in a **.py** file

Language we are programming in

jupyter

Type Python blocks & MD blocks in a **.ipynb** file

“Notebook” for Python code that allows testing small bits at a time

replit

Free online coding environment for Python

Google colab

Free online coding environment that can handle Python notebooks

Code-Along Starter Code Notebook

CODE-ALONG

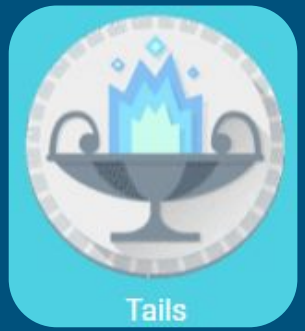
TO DO LIST

- [ ] 1. Flip a Coin
- [ ] 2. Join a Room
- [ ] 3. Open Copy of Starter Code Notebook
- [ ] 4. Code along!
- [ ]

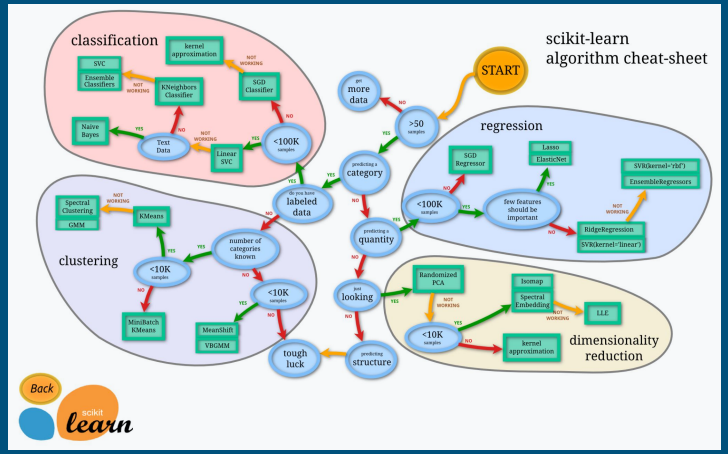
CLICK TO FLIP A COIN!

ROOM 1

ROOM 2



Intro to Scikit Learn



- Machine learning package for python
- Heavily used in data science
- Has built-in methods for various classification, regression, and clustering algorithms
- Also has built-in methods for the undersides of data science as well (preprocessing)

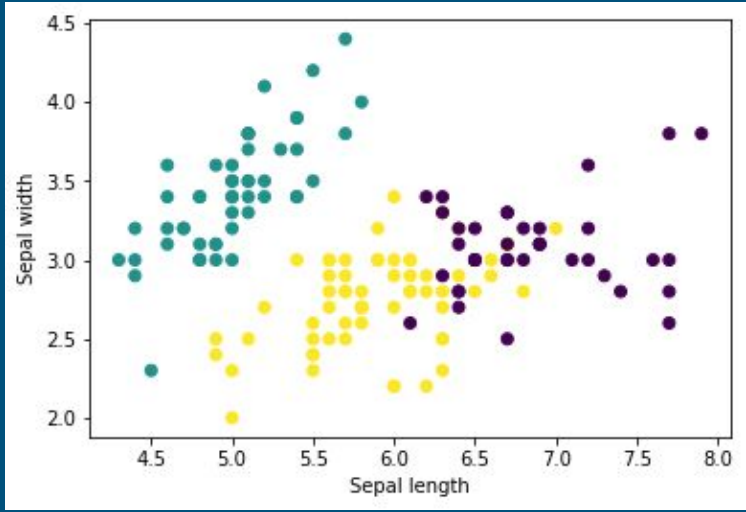
Scikit Learn Code

```python
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Create an implementation of kmeans
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)

# Plot the results
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```



Scikit Learn Code

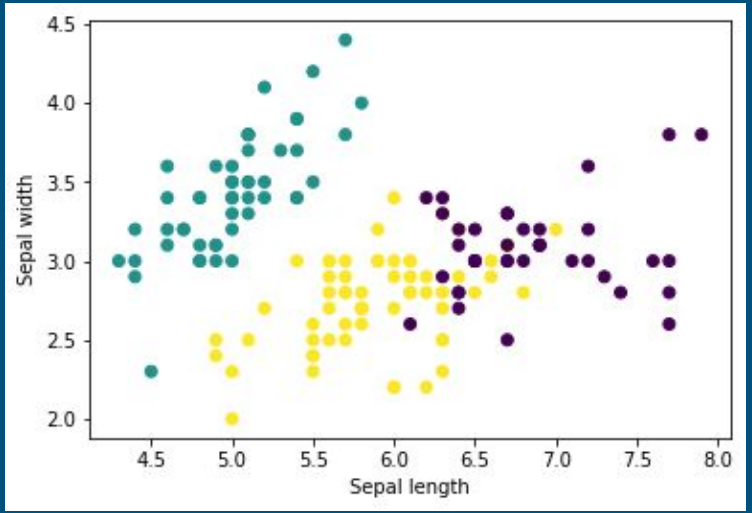
```python
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Create an implementation of kmeans
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)

# Plot the results
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

These lines import both the kmeans algorithm and a prepackaged dataset from scikit learn



Scikit Learn Code

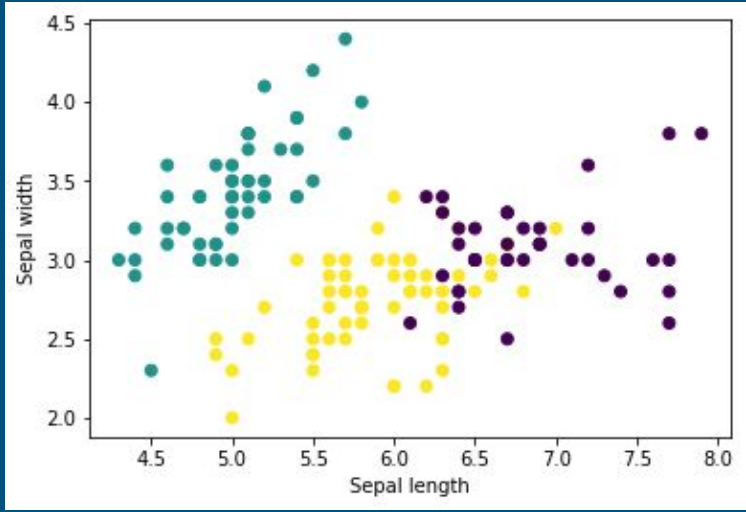
```python
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Create an implementation of kmeans
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)

# Plot the results
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

These lines bring the iris dataset into the working code



Scikit Learn Code

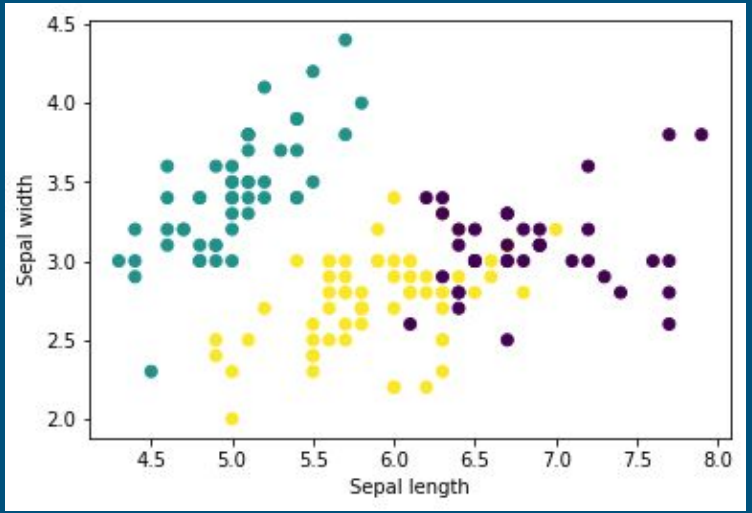
```python
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Create an implementation of kmeans
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)

# Plot the results
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

These lines of code run the entire algorithm we just worked on…



Scikit Learn Code

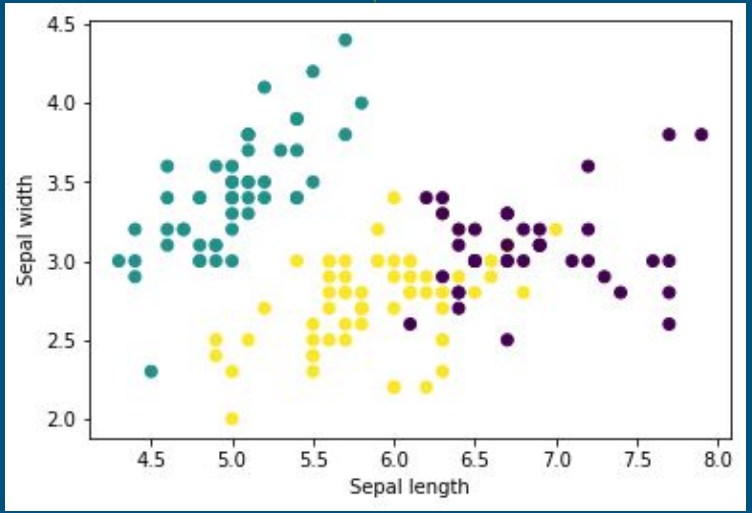
```python
# Import required libraries
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data

# Create an implementation of kmeans
kmeans = KMeans(n_clusters=3)
labels = kmeans.fit_predict(X)

# Plot the results
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

These lines print out the plot shown below



Homework Options

Click Here for Homework