SC310005 Artificial Intelligence

Lecture 8: Unsupervised Learning

teerapong.pa@chula.ac.th

Reference

- https://nixustechnologies.com/unsupervised-machine-learning/
- https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c
- https://www.v7labs.com/blog/supervised-vs-unsupervised-learning
- https://www.labellerr.com/blog/supervised-vs-unsupervised-learning-whats-the-difference/
- https://medium.com/@farhansalimuddin/unleashing-the-power-of-feature-scaling-a-comprehensive-exploration-of-every-kind-b69cecef053a

Data in Supervised vs. Unsupervised Learning

Supervised Learning

Unsupervised Learning

Classical Machine Learning

Task Driven

Data Driven

Supervised Learning

(Pre Categorized Data)

Classification

(Divide the socks by Color)

Eg. Identity Fraud Detection

Regression

(Divide the Ties by Length)

Eg. Market Forecasting

Unsupervised Learning

(Unlabelled Data)

Clustering

(Divide by Similarity)

Eg. Targeted Marketing

Association

(Identify Sequences)

Eg. Customer Recommendation

Dimensionality Reduction

(Wider Dependencies)

Eg. Big Data Visualization

Obj: Predications & Predictive Models

Pattern/ Structure Recognition

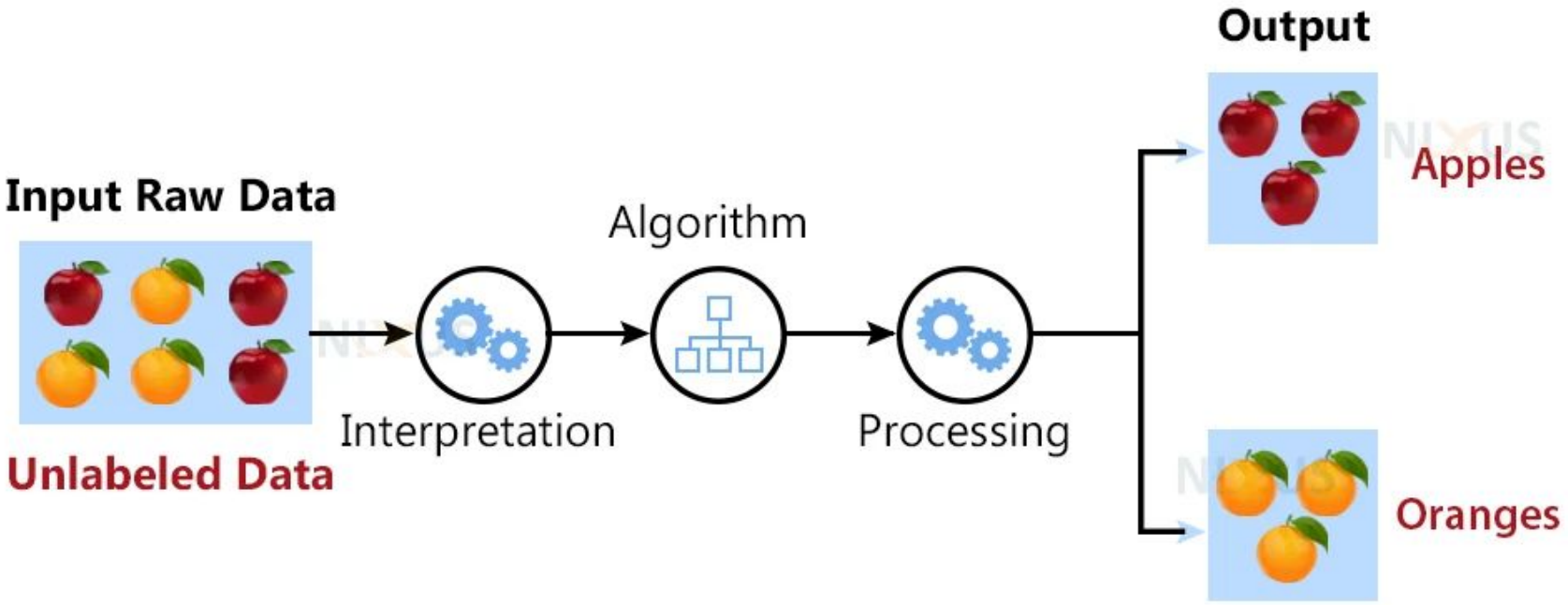
Unsupervised Learning

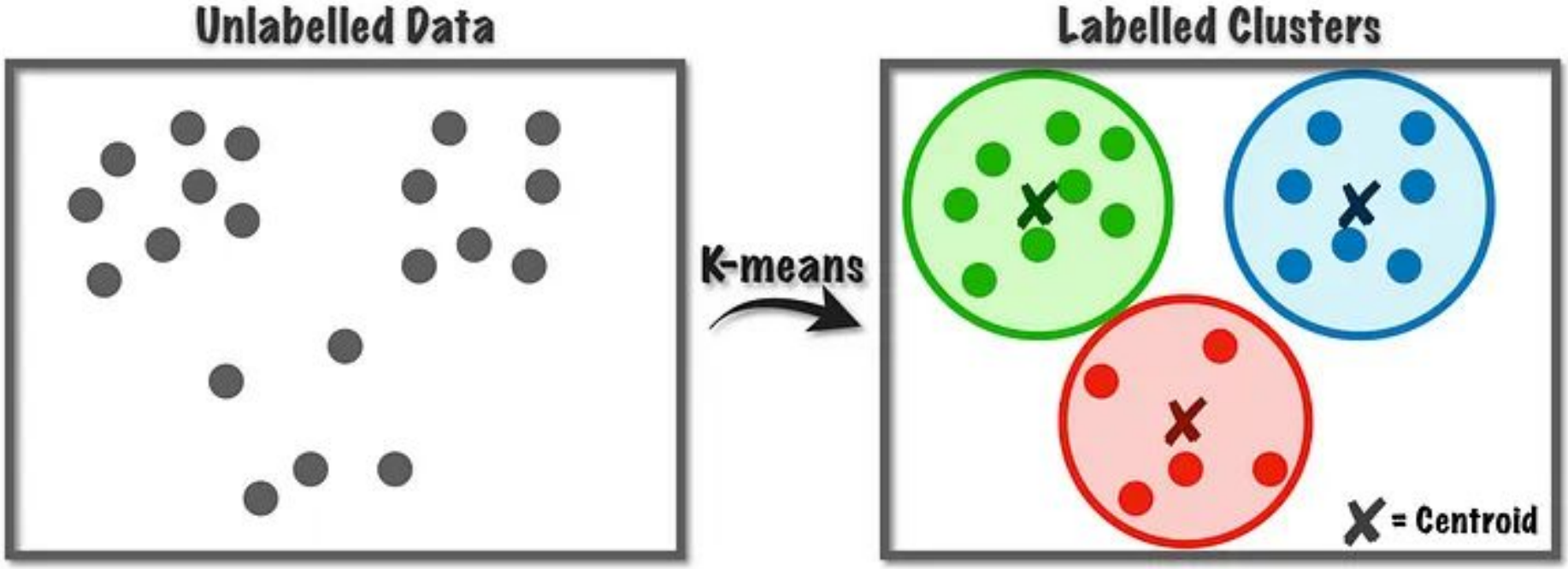
Unsupervised learning is another machine learning method in which patterns inferred from the unlabeled input data.

The goal of unsupervised learning is to **find the structure and patterns from the input data**.

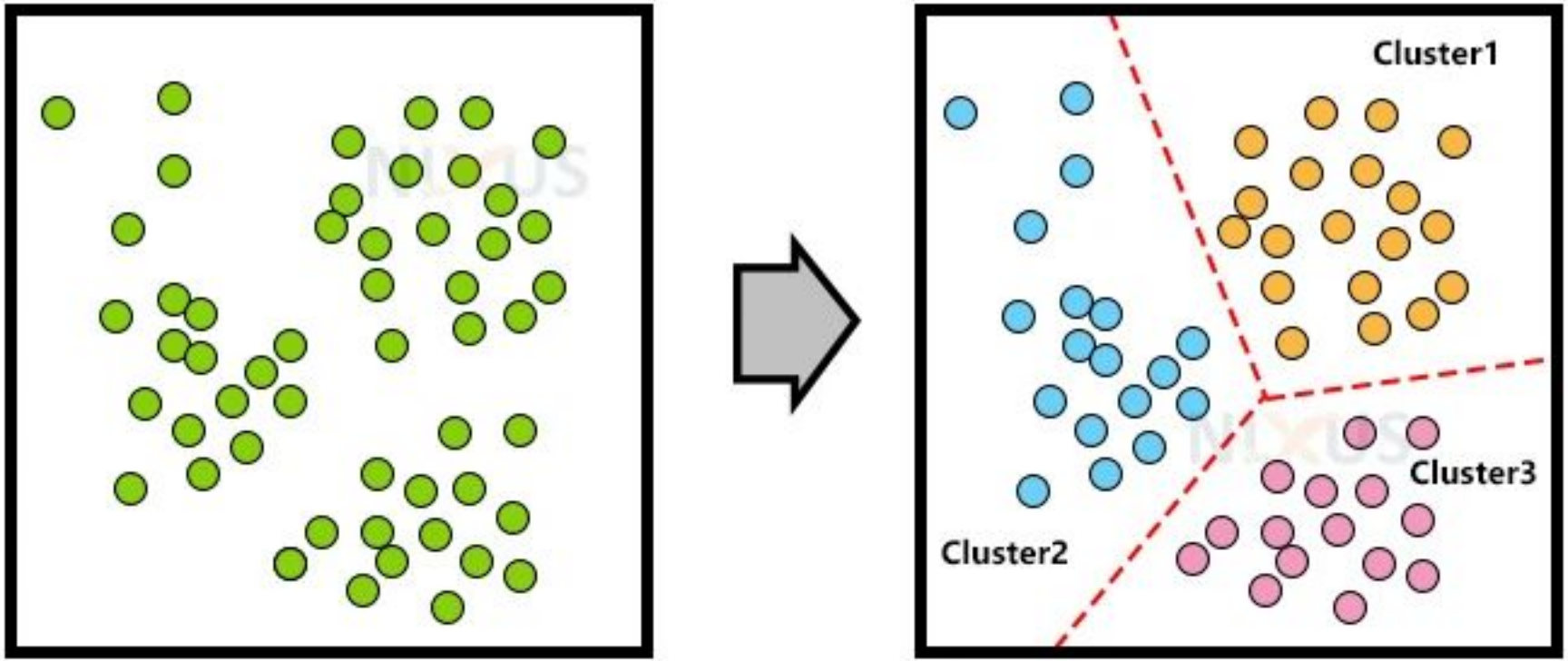
Unsupervised learning does not need any supervision. Instead, it finds patterns from the data by its own.

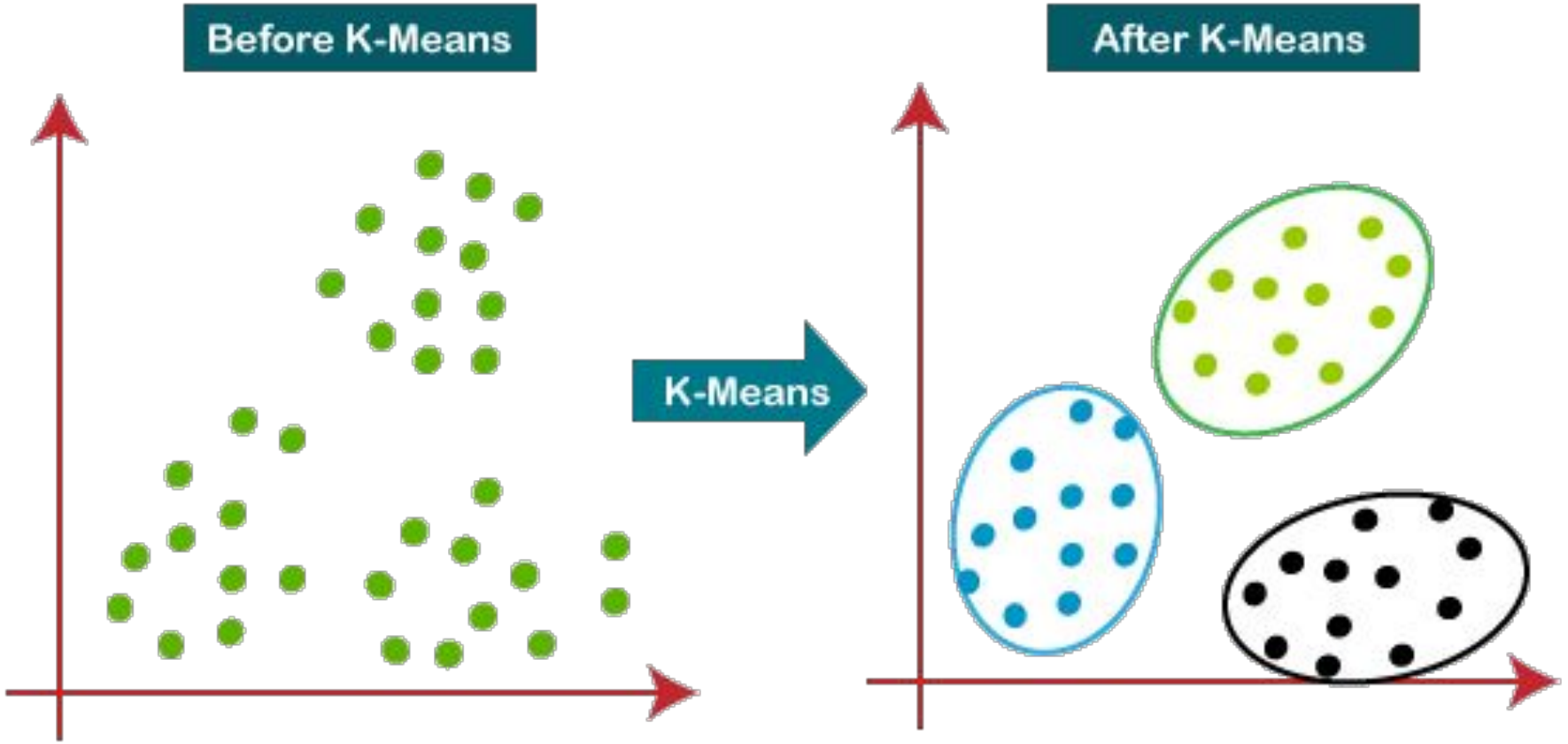
Unsupervised Machine Learning




Unlabelled Data
K-means
Labelled Clusters
= Centroid

Clustering



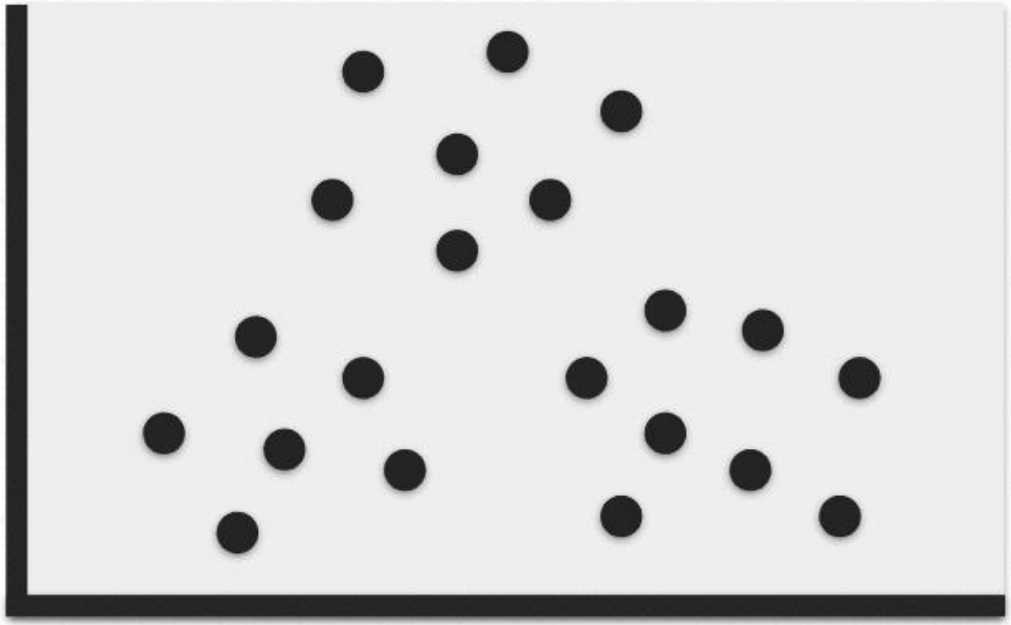

Before K-Means
K-Means
After K-Means

K-means Algorithm

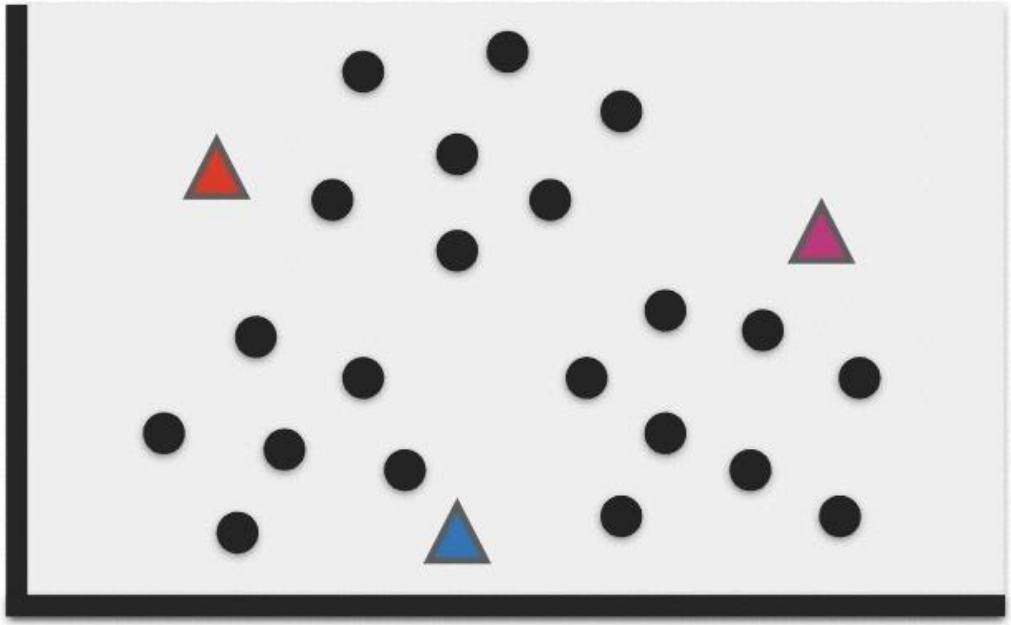
The intuition behind the algorithm is actually pretty straight forward. To begin, we choose a value for k (the number of clusters) and randomly choose an initial centroid (centre coordinates) for each cluster. We then apply a two step process:

1. **Assignment step** — Assign each observation to it's nearest centre.
2. **Update step** — Update the centroids as being the centre of their respective observation.

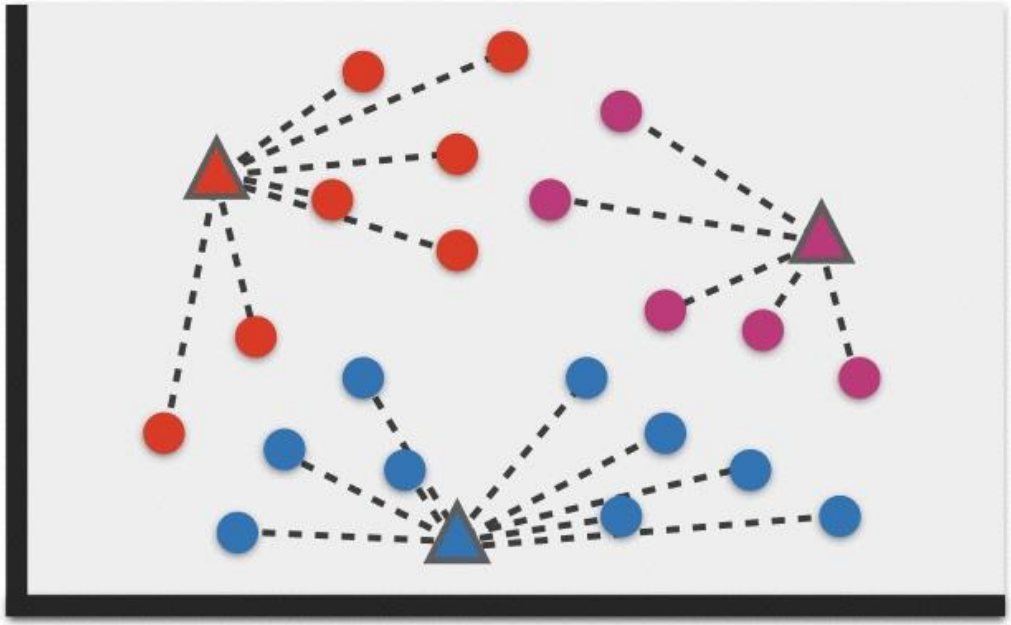
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



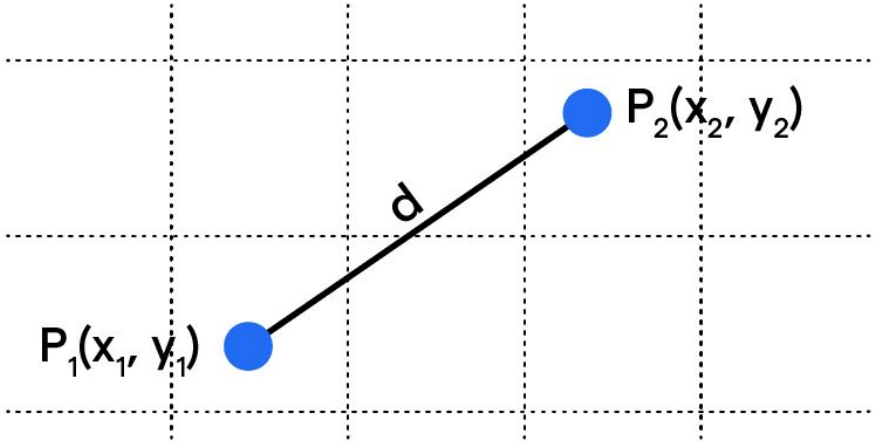
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



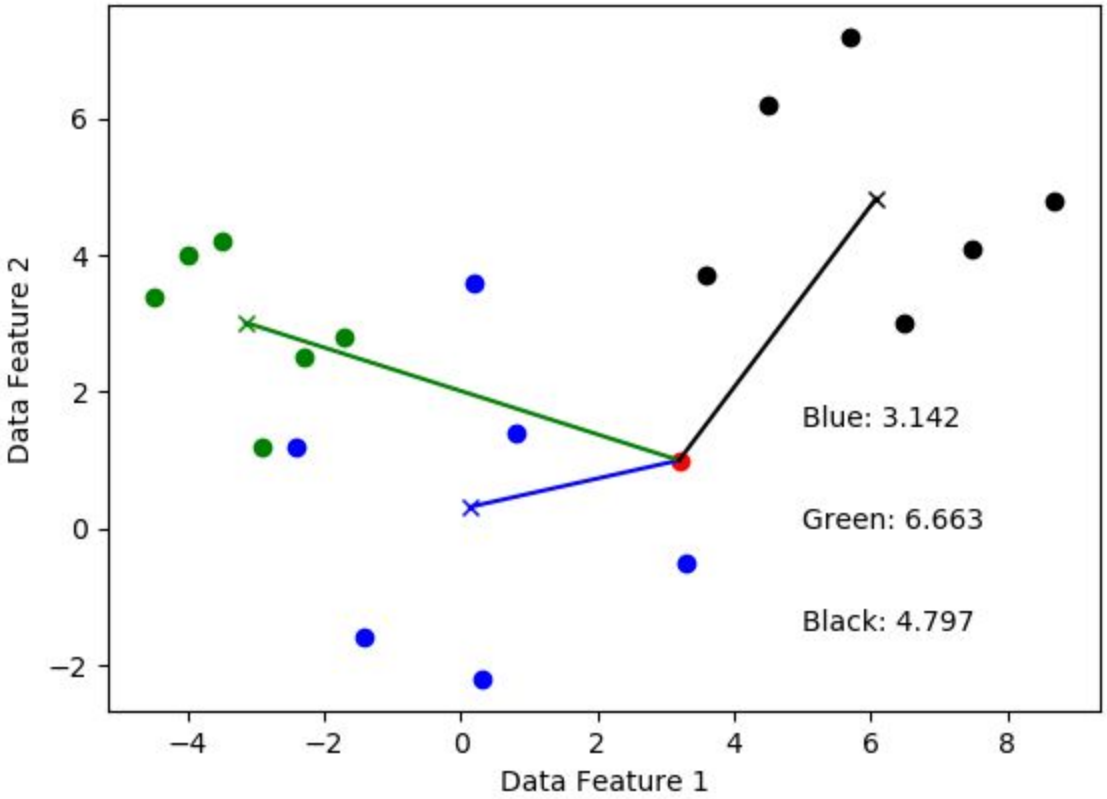
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End

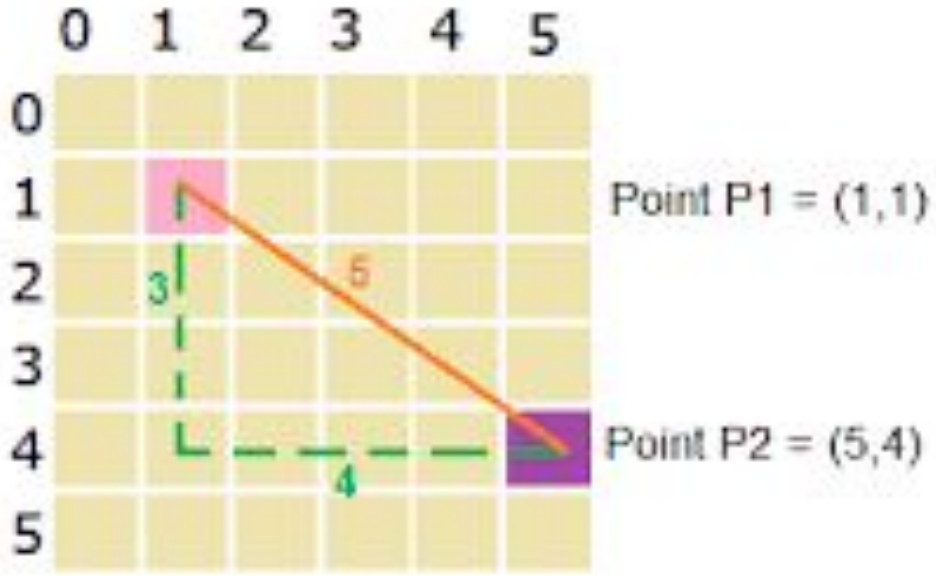


Euclidean Distance



$$\text{Euclidean Distance (d)} = \sqrt{(x_2 - y_1)^2 + (y_2 - y_1)^2}$$

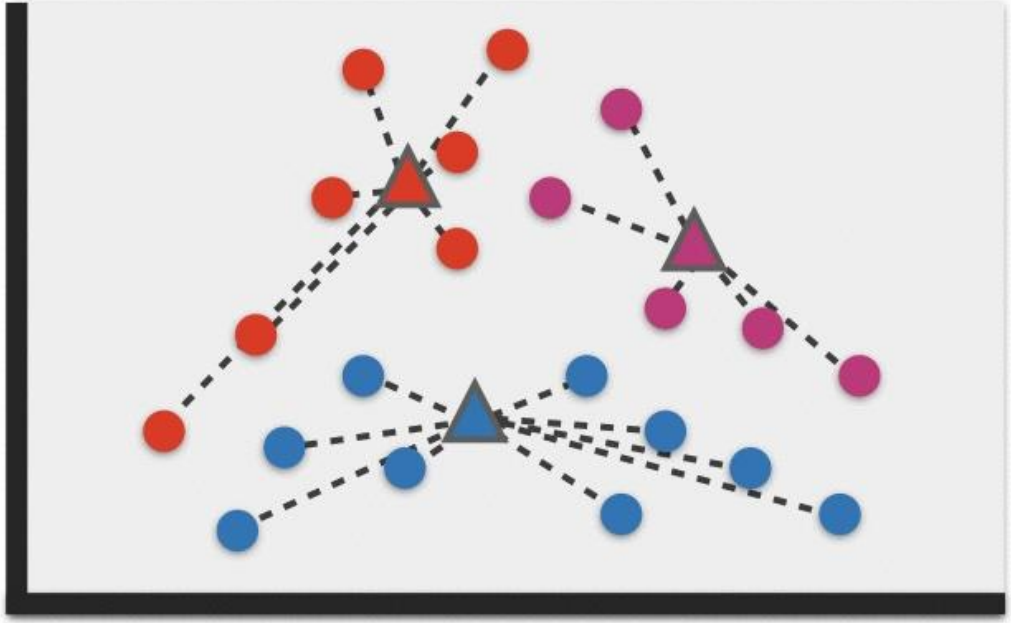

Data Feature 2
Data Feature 1
6
4
2
0
−2
−4
−2
0
2
4
6
8
Blue: 3.142
Green: 6.663
Black: 4.797



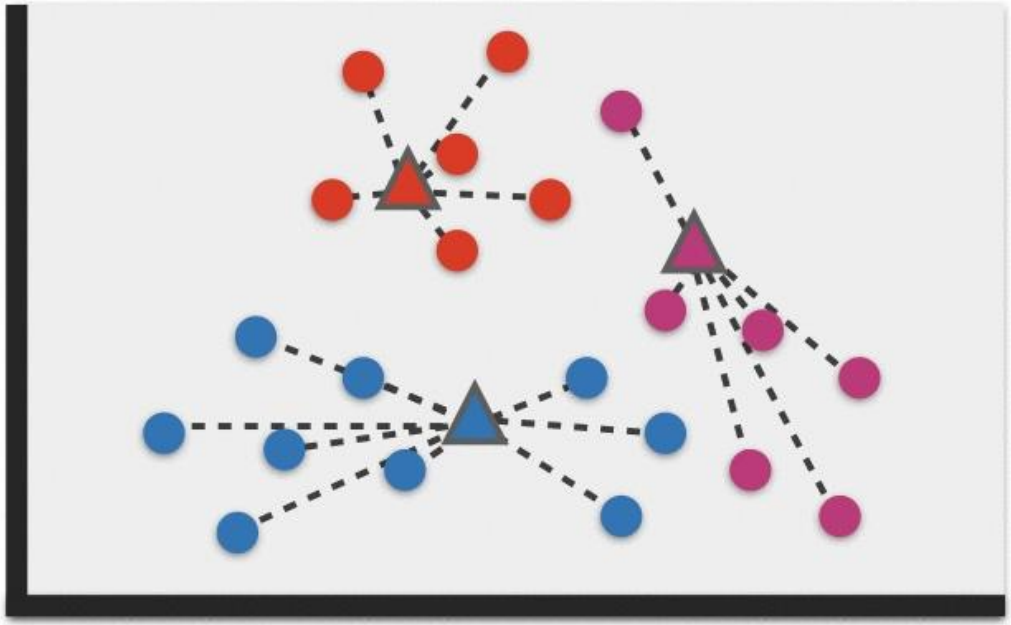
Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$

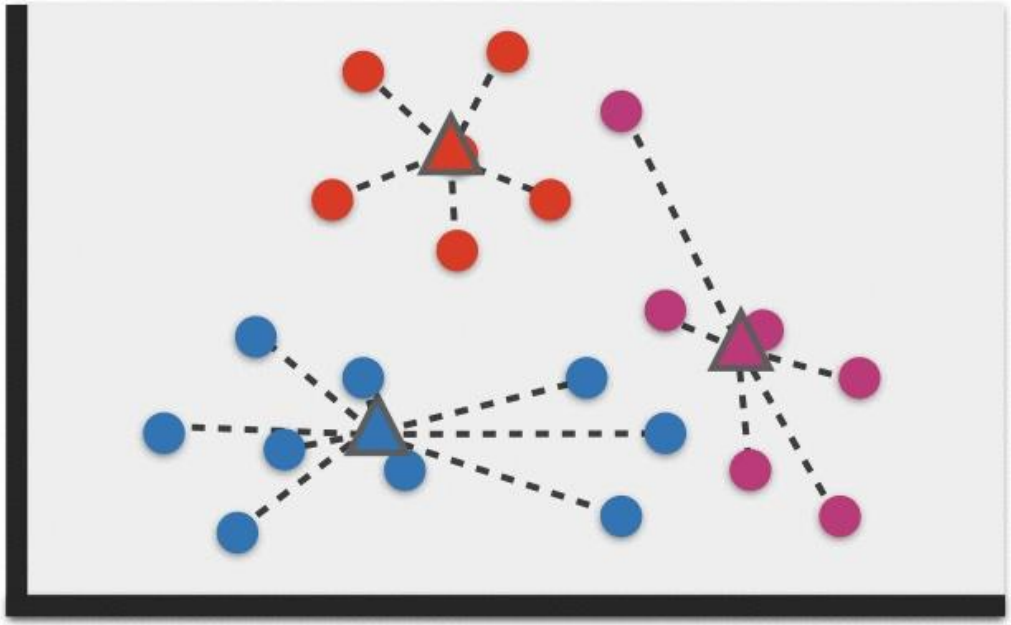
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



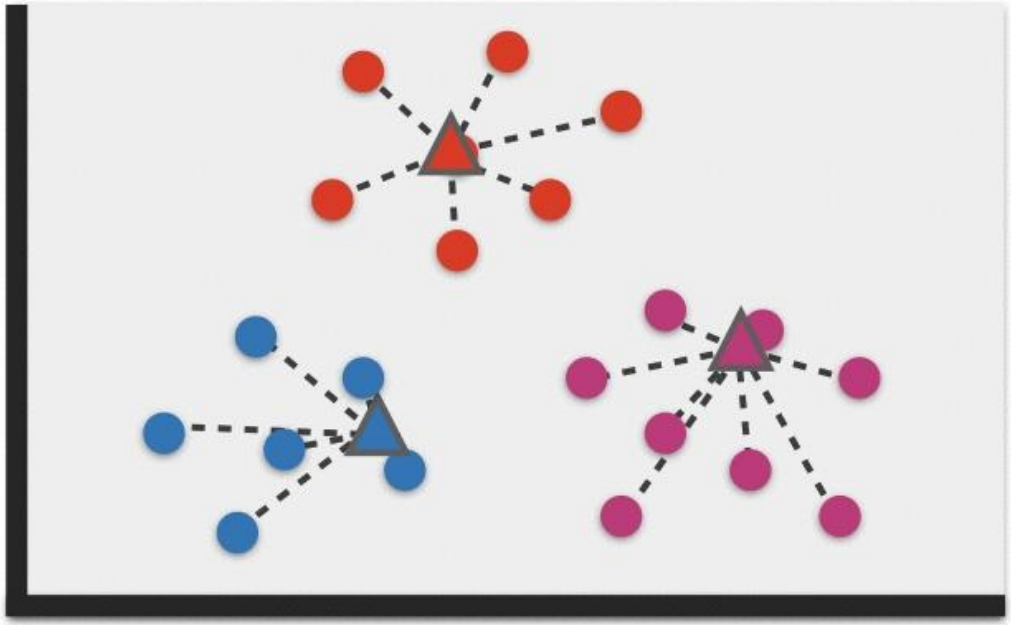
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



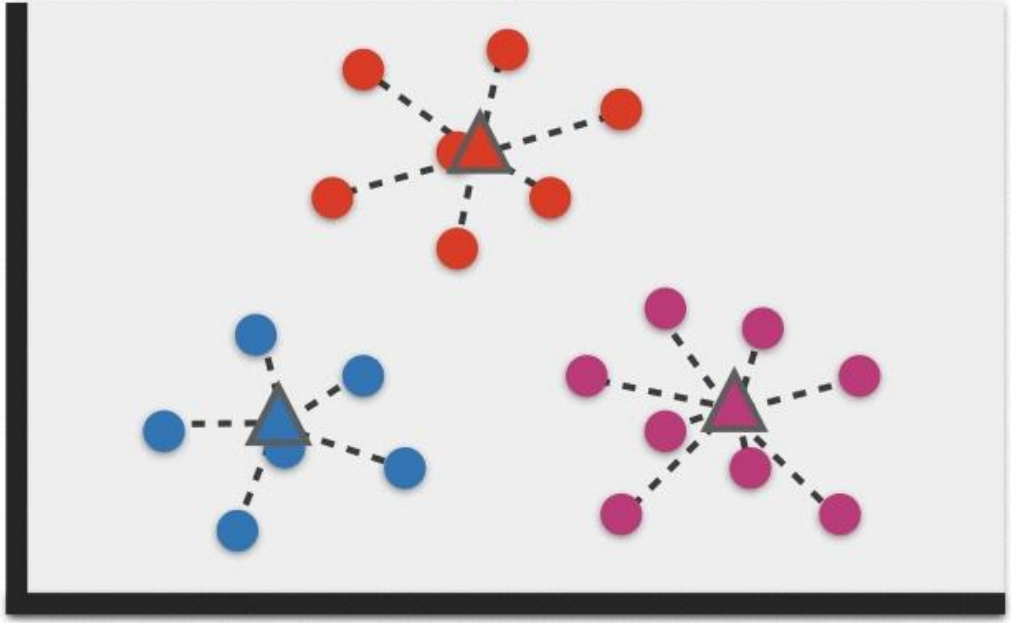
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



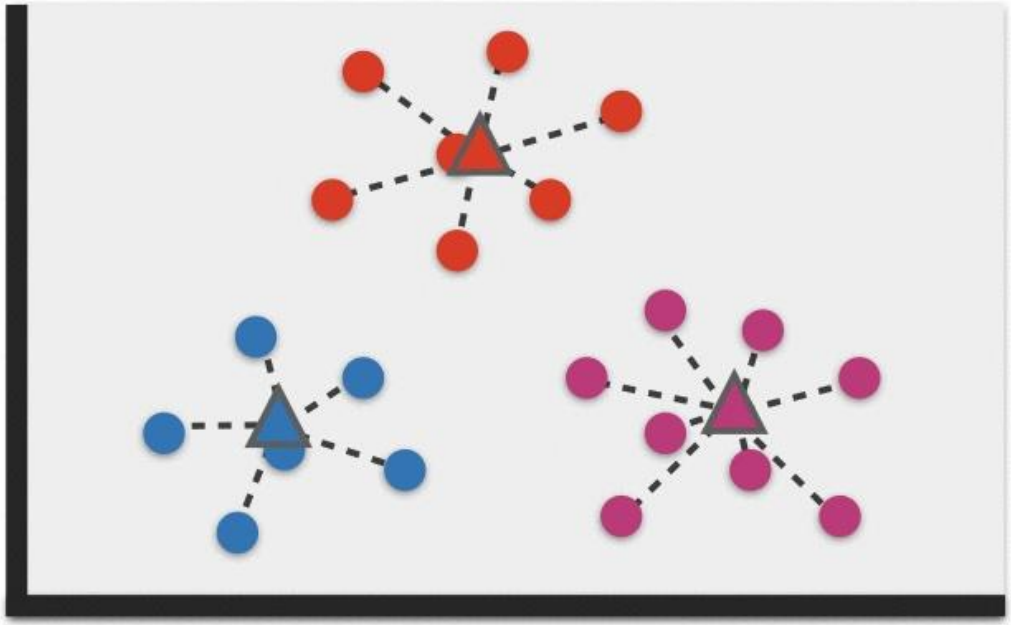
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



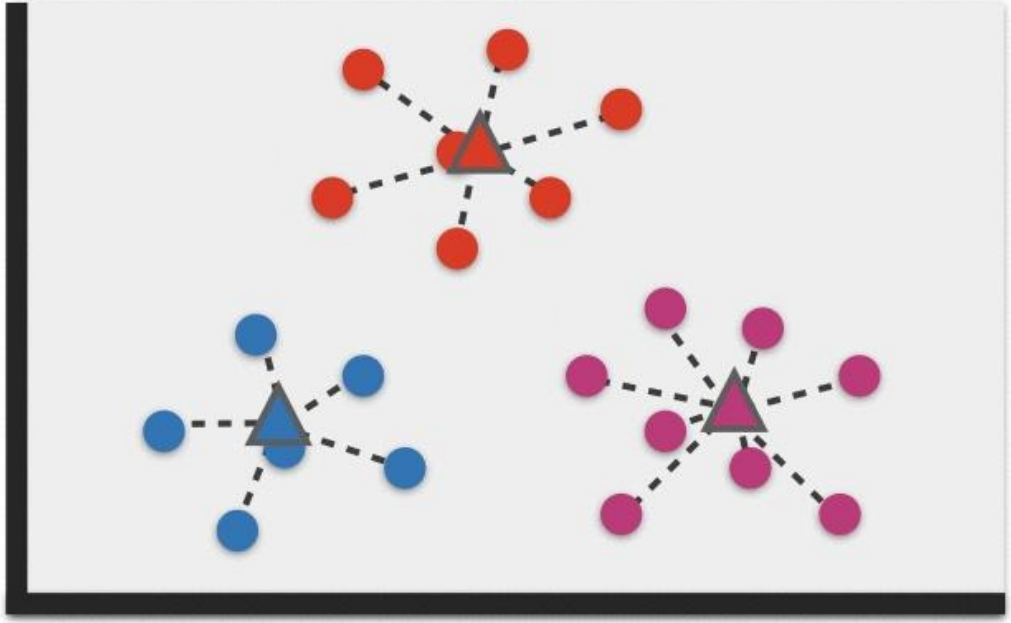
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



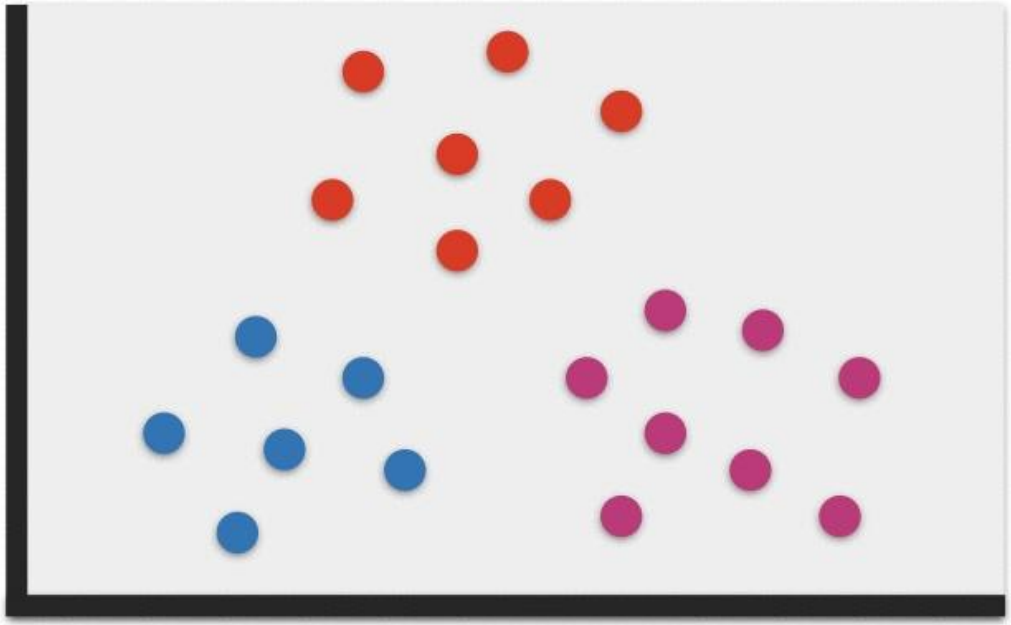
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End

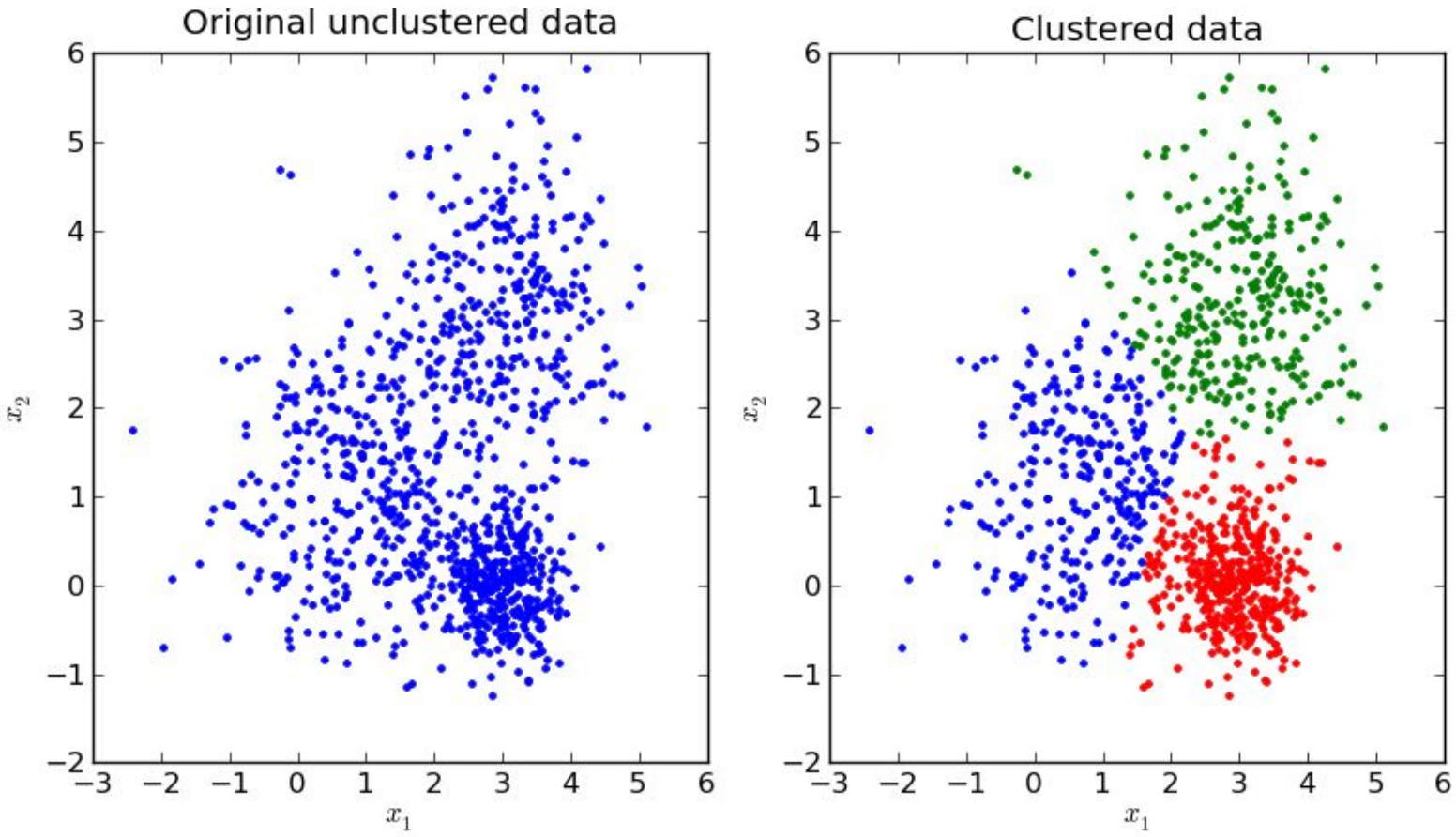


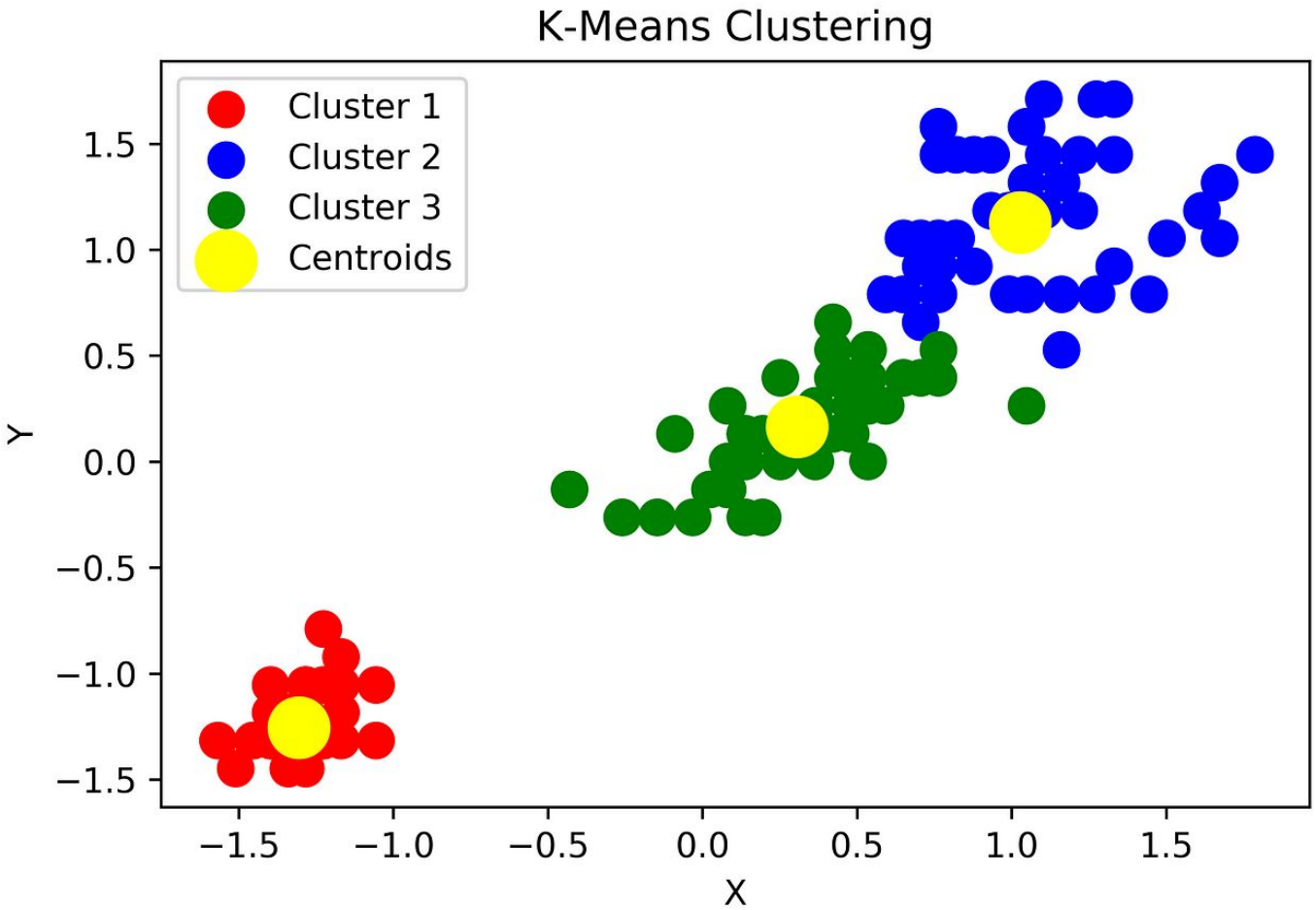
1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End

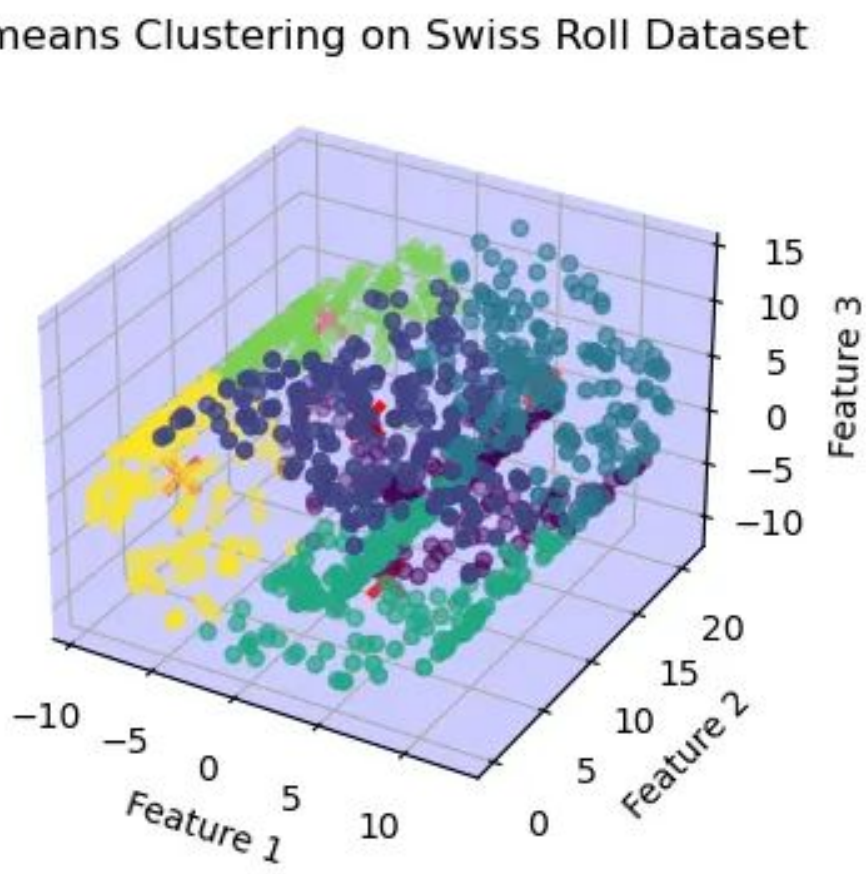


1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End




Original unclustered data
Clustered data
x_1
x_2
6
5
4
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
4
5
6


K-Means Clustering
Cluster 1
Cluster 2
Cluster 3
Centroids
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
X


K-means Clustering on Swiss Roll Dataset
15
10
5
0
−5
−10
Feature 3
20
15
10
5
0
Feature 2
−10
−5
0
5
10
Feature 1

Feature Scaling

Feature scaling is a data preprocessing technique used in machine learning to bring all numerical features in a dataset to a consistent scale or range.

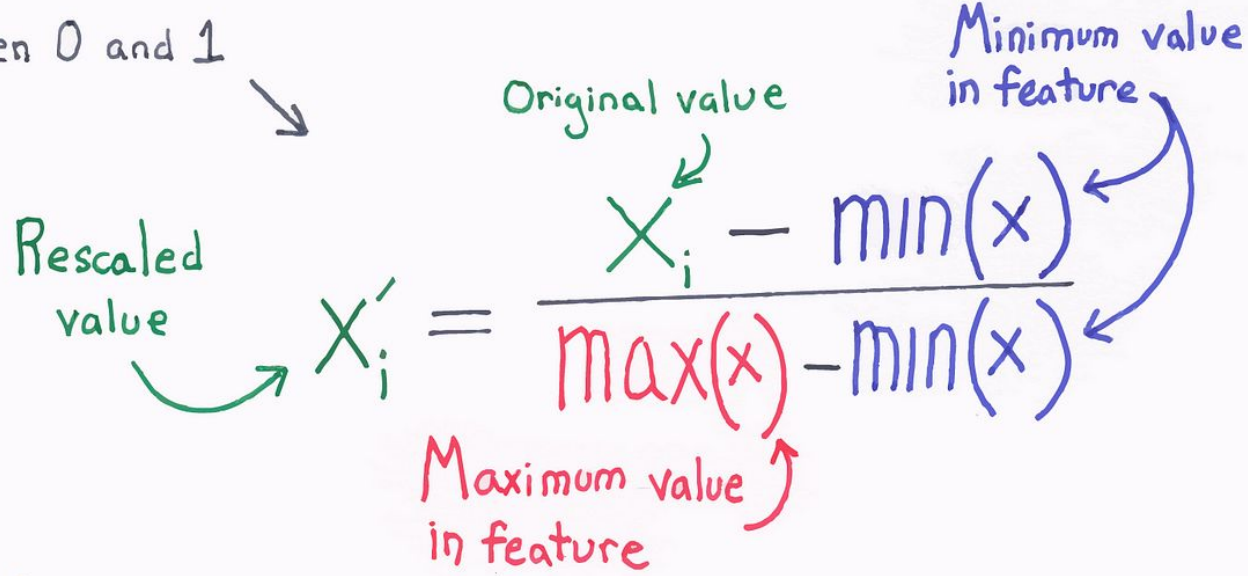
In many real-world datasets, features may have different units of measurement or widely varying magnitudes.

Feature Scaling: Min-Max Scaler

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

MINMAX SCALING

Rescales feature values to between 0 and 1



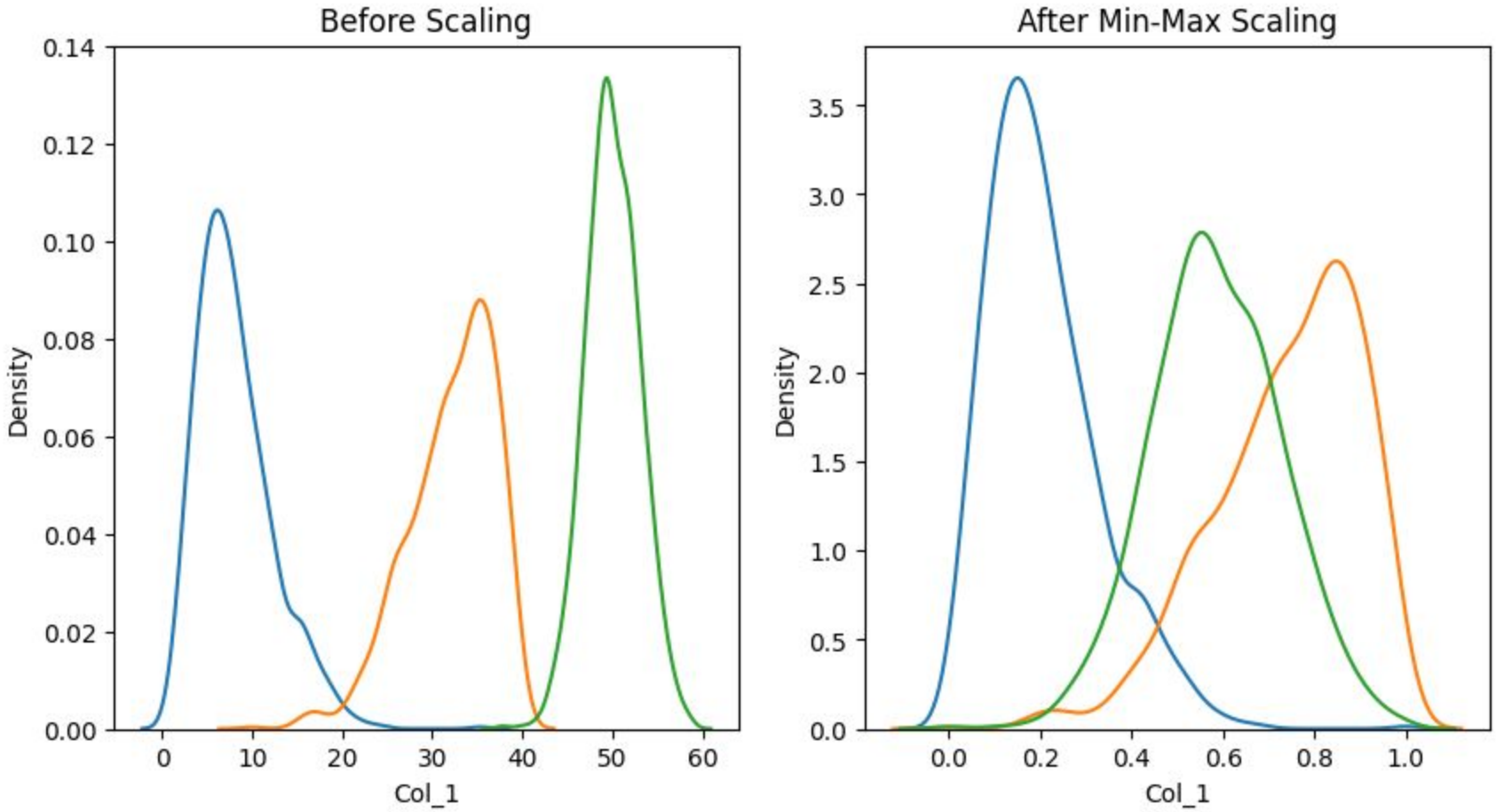
Chris Albon

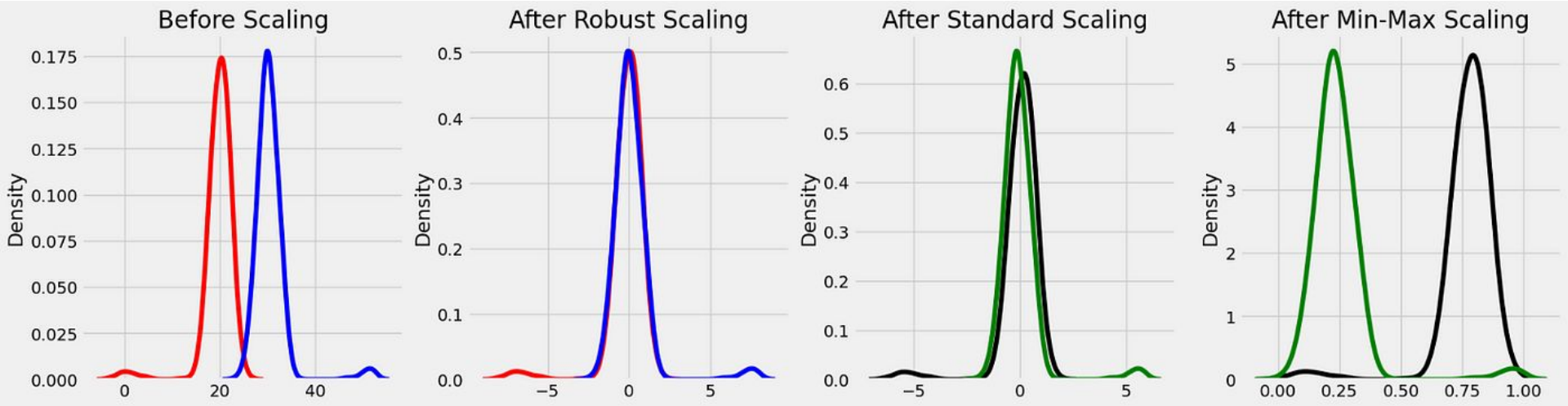
Feature Scaling: Standard Scaler

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

μ = Mean

σ = Standard Deviation


Before Scaling
After Min-Max Scaling
Density
Col_1
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
10
20
30
40
50
60
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8


Before Scaling
After Robust Scaling
After Standard Scaling
After Min-Max Scaling
Density
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
20
40
0.5
0.4
0.3
0.2
0.1
0.0
−5
0
5
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−5
0
5
5
4
3
2
1
0
0.00
0.25
0.50
0.75
1.00

Recap: One-Hot Encoding

id	color
1	red
2	blue
3	green
4	blue

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html

Pandas.get_dummies vs LabelEncoder

pandas.get_dummies

```
pandas.get_dummies(data, prefix=None, prefix_sep='_',
dummy_na=False, columns=None, sparse=False, drop_first=False,
dtype=None)
```

[source]

Convert categorical variable into dummy/indicator variables.

Each variable is converted in as many 0/1 variables as there are different values. Columns in the output are each named after a value; if the input is a DataFrame, the name of the original variable is prepended to the value.

sklearn.preprocessing.LabelEncoder

class `sklearn.preprocessing.LabelEncoder` [source]

Encode target labels with value between 0 and n_classes-1.

This transformer should be used to encode target values, *i.e.* `y`, and not the input `X`.

Read more in the User Guide.

New in version 0.12.

Attributes:

classes_ : *ndarray of shape (n_classes,)*
Holds the label for each class.

See also:

`OrdinalEncoder`
Encode categorical features using an ordinal encoding scheme.

`OneHotEncoder`
Encode categorical features as a one-hot numeric array.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

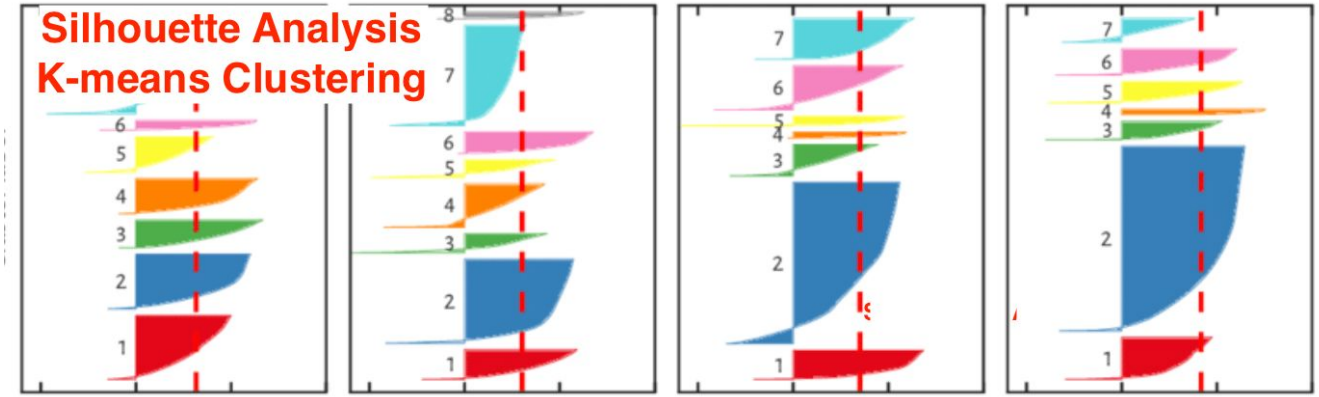
It can also be used to transform non-numerical labels (as long as they are hashable and comparable)

```
>>> le = LabelEncoder()
>>> le.fit(["paris", "paris", "tokyo", "amsterdam"])
LabelEncoder()
>>> list(le.classes_)
['amsterdam', 'paris', 'tokyo']
>>> le.transform(["tokyo", "tokyo", "paris"])
array([2, 2, 1]...)
>>> list(le.inverse_transform([2, 2, 1]))
['tokyo', 'tokyo', 'paris']
```

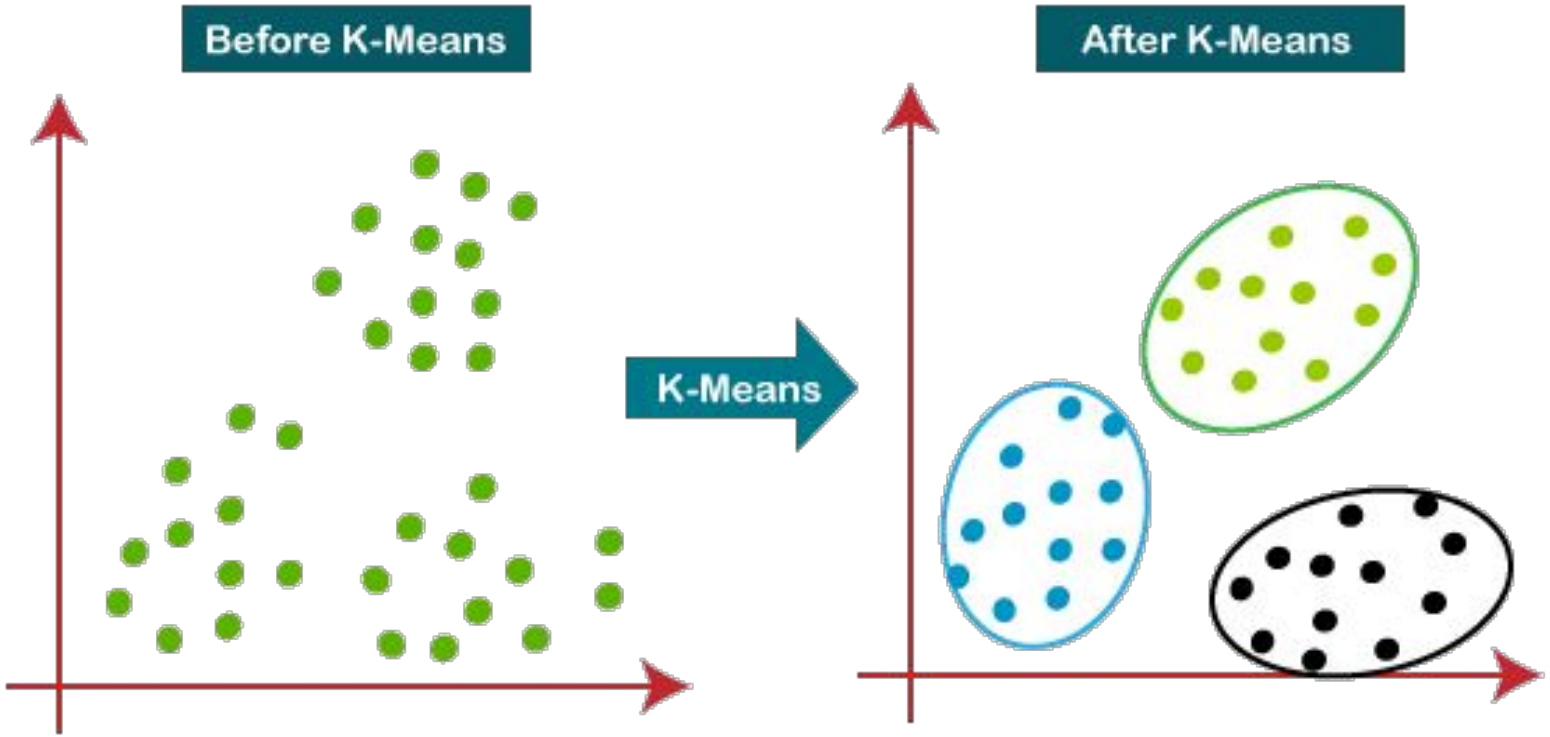
https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Silhouette Score

The silhouette coefficient or silhouette score kmeans is a measure of how similar a data point is within-cluster (cohesion) compared to other clusters (separation). The Silhouette score can be easily calculated in Python using the metrics module of the scikit-learn/sklearn library.



How to Find the Optimal Number of Clusters

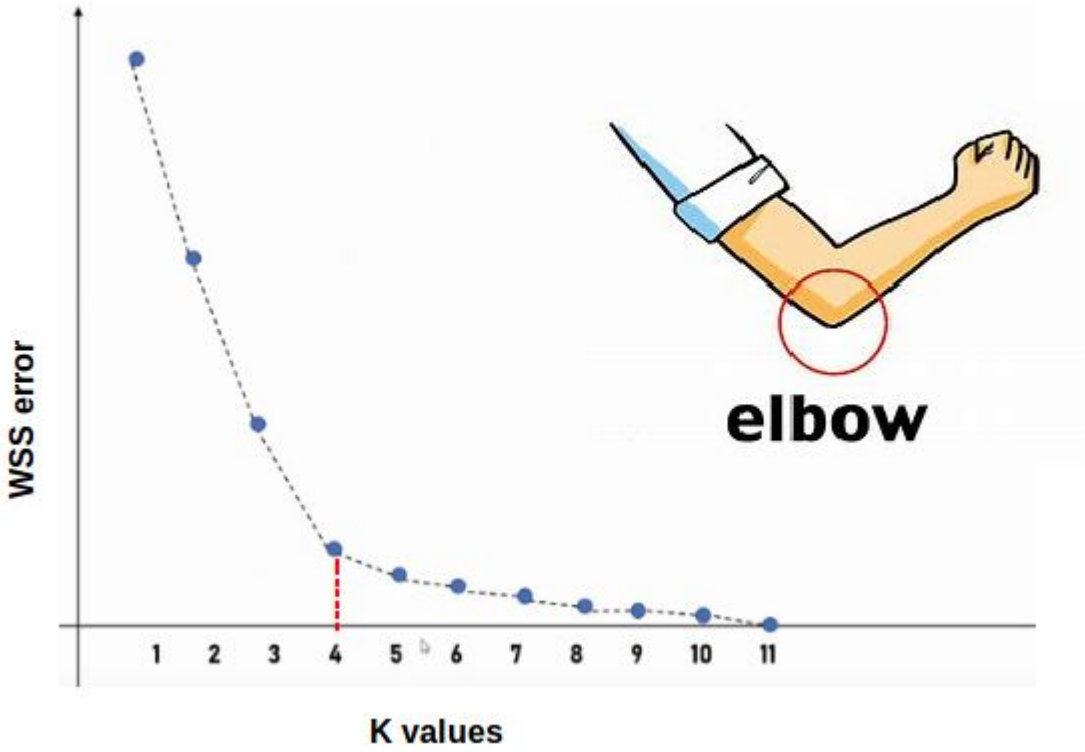


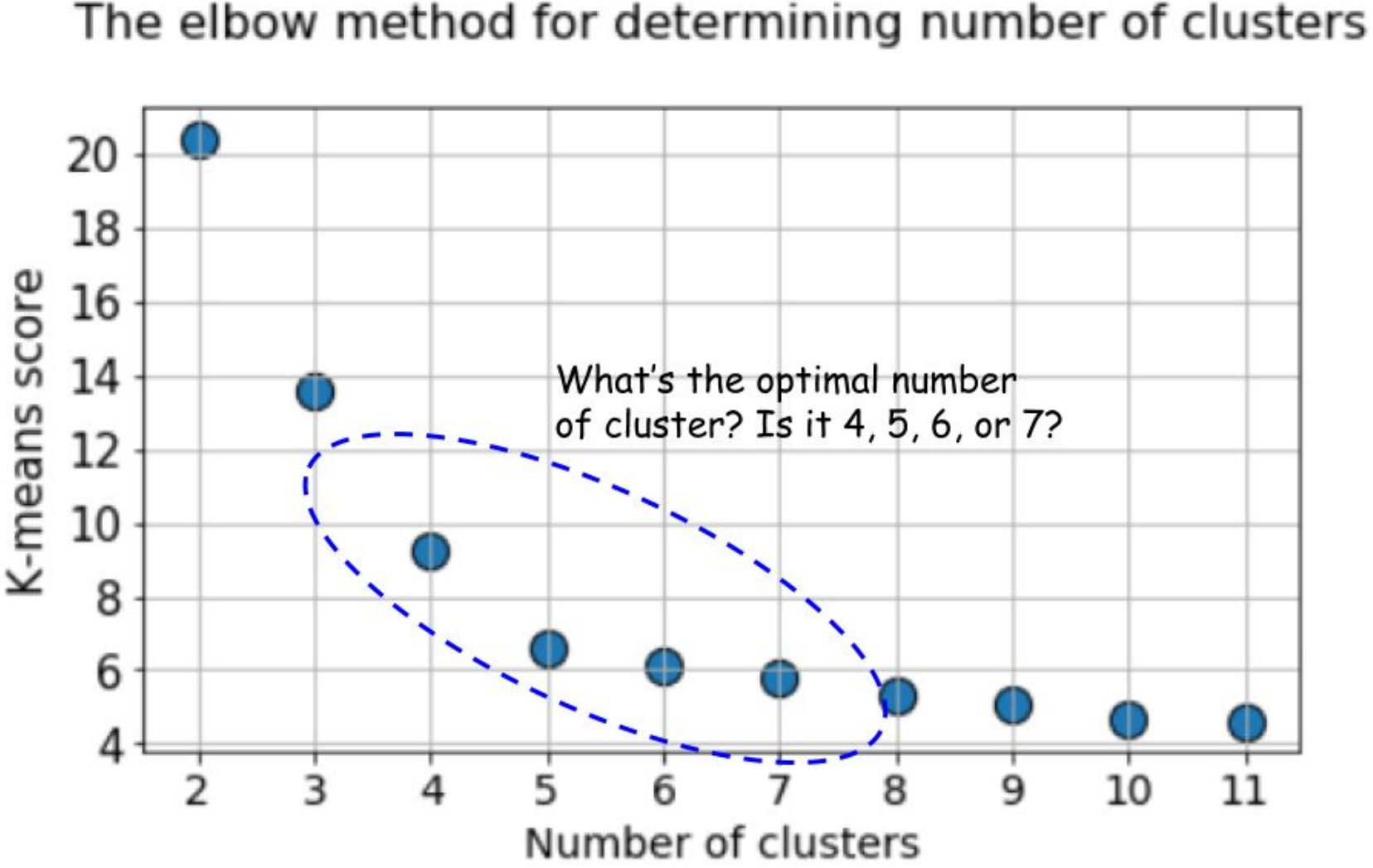
Elbow Method in K-means Clustering

The elbow method is a graphical method for finding the optimal K value in a k-means clustering algorithm.

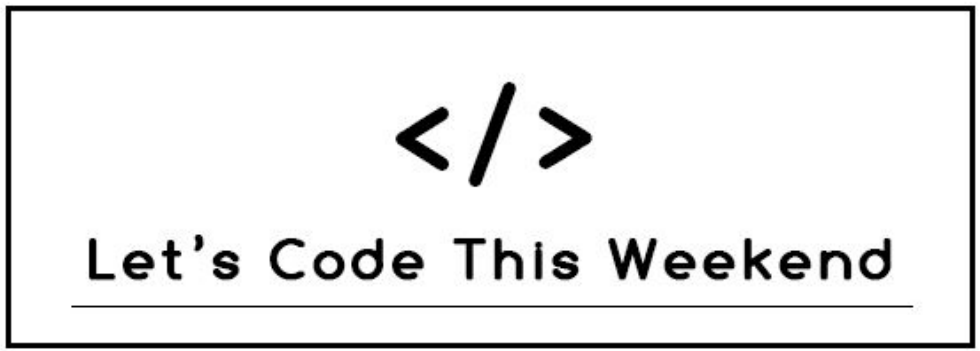
The elbow graph shows the within-cluster-sum-of-square (WCSS) values on the y-axis corresponding to the different values of K (on the x-axis). The optimal K value is the point at which the graph forms an elbow.

Elbow method




The elbow method for determining number of clusters
K-means score
Number of clusters
What's the optimal number of cluster? Is it 4, 5, 6, or 7?
20
18
16
14
12
10
8
6
4
2
3
4
5
6
7
8
9
10
11

Let's Code!



✓ 0s

[1]
```
import warnings
warnings.filterwarnings("ignore")
```

[2]
```
# # Necessary libraries using:
!pip install scikit-learn pandas numpy matplotlib yellowbrick
```

[3] ✓ 1s

```python
# Import Libraries

import pandas as pd
import numpy as np

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score

from yellowbrick.cluster import KElbowVisualizer

import matplotlib.pyplot as plt
import seaborn as sns
```

```
[4]  1 # Load Titanic dataset
     2 titanic_data = pd.read_csv('titanic_dataset.csv')
```

```
[5]  1 titanic_data.columns
```

0s

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
[6]  # Drop irrelevant columns or handle missing values
     titanic_data.drop(['PassengerId', 'Name', 'Ticket', 'Cabin', 'Embarked'], axis=1, inplace=True)
     titanic_data['Age'].fillna(titanic_data['Age'].median(), inplace=True)
     titanic_data['Fare'].fillna(titanic_data['Fare'].median(), inplace=True)
```

```
[7]  # Convert categorical features to numerical
     titanic_data['Sex'] = titanic_data['Sex'].map({'male': 0, 'female': 1})
```

```
[8]  # Select relevant features
     features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Survived']
     X = titanic_data[features]
```

[9] 0s

```
# Drop rows with missing values in the selected features
X.dropna(inplace=True)

# Separate the target variable
y = X['Survived']
X = X.drop('Survived', axis=1)
```

[10] 0s

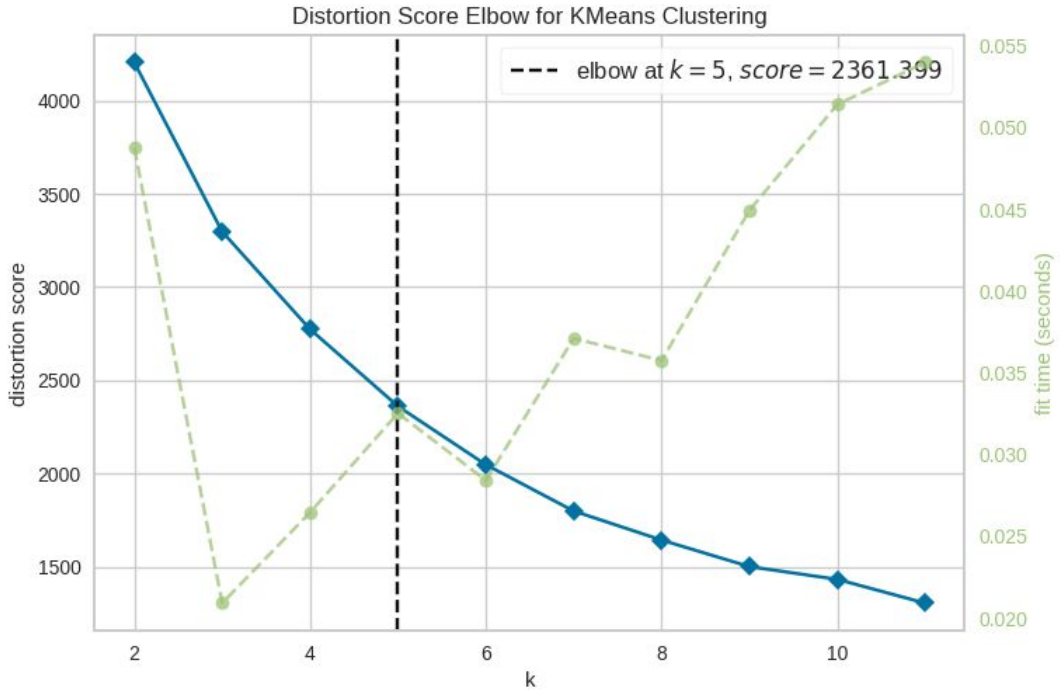
```
# Min-Max scaling
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

[11] 0s

```
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

✓ 1s

```
# Determine the optimal number of clusters (k) using the elbow method
visualizer = KElbowVisualizer(KMeans(), k=(2, 12))
visualizer.fit(X_scaled)
visualizer.show()
```



[13]
```
# Choose the optimal k based on the elbow plot or silhouette score
optimal_k = visualizer.elbow_value_
optimal_k
```

```
5
```

[14]
```
# Apply k-means clustering
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
labels = kmeans.fit_predict(X_scaled)
```

```
# Evaluate clustering using silhouette score
silhouette_avg = silhouette_score(X_scaled, labels)
print(f'Silhouette Score: {silhouette_avg}')
```

```
Silhouette Score: 0.3716355864098721
```

[16]
```
# Add cluster labels to the original DataFrame
titanic_data['Cluster'] = labels
titanic_data.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cluster
0	0	3	0	22.0	1	0	7.2500	3
1	1	1	1	38.0	1	0	71.2833	0
2	1	3	1	26.0	0	0	7.9250	2
3	1	1	1	35.0	1	0	53.1000	0
4	0	3	0	35.0	0	0	8.0500	3

```
# Visualize the clusters on the original data points with labels
plt.figure(figsize=(10, 6))
scatter = plt.scatter(titanic_data['Fare'], titanic_data['Age'], c=labels, cmap='viridis', alpha=0.5)
plt.title('K-Means Clustering Results on Original Data Points')
plt.xlabel('Fare')
plt.ylabel('Age')

legend_labels = np.unique(labels)
legend_elements = [plt.Line2D([0], [0], marker='o', color='w', label=f'Cluster {label}',
                              markerfacecolor=scatter.to_rgba(label), markersize=10) for label in legend_labels]
plt.legend(handles=legend_elements, loc='upper right')

plt.show()
```

[18]
```
# Interpret the clusters
cluster_means = titanic_data.groupby('Cluster').mean()
cluster_means.head()
```

Cluster	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0.885417	1.010417	0.833333	31.009583	0.572917	0.656250	133.717534
1	0.335526	1.263158	0.085526	44.299342	0.296053	0.125000	38.953838
2	0.725389	2.585492	1.000000	25.222798	0.414508	0.461140	15.971417
3	0.150127	2.798982	0.000000	26.935751	0.211196	0.111959	12.162360
4	0.122807	2.964912	0.491228	17.491228	3.561404	2.192982	36.382312

[20]
0s

```
# Analyze cluster characteristics, e.g., survival rate
if 'Survived' in titanic_data.columns:
    survival_rate_per_cluster = titanic_data.groupby('Cluster')['Survived'].mean()
    print('\nSurvival Rate per Cluster:')
    print(survival_rate_per_cluster)
else:
    print("Survived column not found in the clustered data.")
```

```
Survival Rate per Cluster:
Cluster
0    0.885417
1    0.335526
2    0.725389
3    0.150127
4    0.122807
Name: Survived, dtype: float64
```

Week 8: Assignment

Apply k-means clustering to group regions based on selected COVID-19 features.

The main objective of this assignment is to explore patterns in COVID-19 data using k-means clustering and interpret the results.