CoGrammar

Welcome to this session:

Task Walkthrough - Task 22 - 26 (Part 1)

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Data Science

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions]

Skills Bootcamp Data Science

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***
- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***
- We would love your feedback on lectures: ***Feedback on Lectures***
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- **Explain the role of image processing** in machine learning.
- **Preprocess and extract features** from image datasets.
- **Train a Random Forest Classifier** to classify images.
- **Explain how K-Means Clustering works** and its applications.
- **Use the elbow method and silhouette score** to determine the optimal number of clusters.
- **Apply K-Means Clustering** to real-world datasets and interpret the results.

Lecture Overview

- → Presentation of the Task
- → Image Processing
- → K-means Clustering
- → Task Walkthrough

Task Walkthrough

Imagine you work at a space research agency analyzing satellite images. Your mission is to identify different types of land cover (e.g., water bodies, vegetation, urban areas) using machine learning.

You will:

❖ Load a satellite image dataset and convert it into numerical values.

❖ Preprocess the image by converting it to grayscale and extracting features.

❖ Apply K-Means Clustering to categorize different land types.

❖ Determine the optimal number of clusters using the Elbow Method.

❖ Visualize the clustered image to interpret land patterns.

How does a machine interpret an image for machine learning?

A. As a single pixel value
B. As an array of numerical values
C. As a sequence of characters
D. As a 3D object

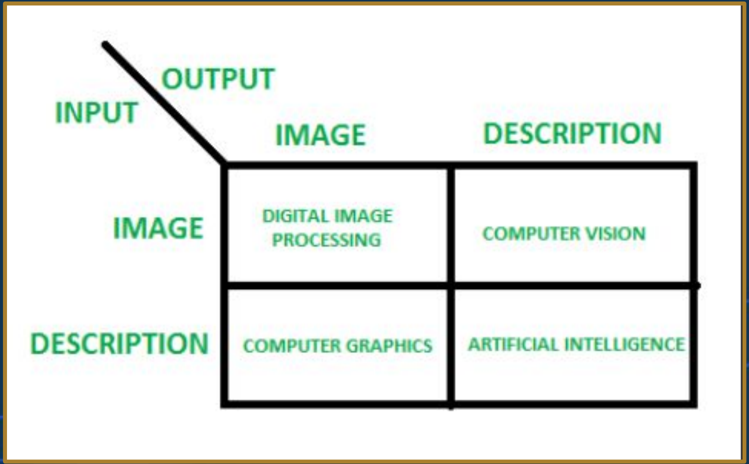
What is a key assumption of K-Means Clustering?

A. Data points are already labelled
B. The dataset is small
C. The dataset can be divided into distinct groups
D. The clusters are always the same size

CoGrammar

Image Processing

- Image processing refers to the **manipulation and analysis of digital images** to **extract meaningful information** and enhance visual content.
- Enables computers to **interpret and understand visual data**.
- Facilitates the development of intelligent systems capable of **automating tasks related to visual perception**.



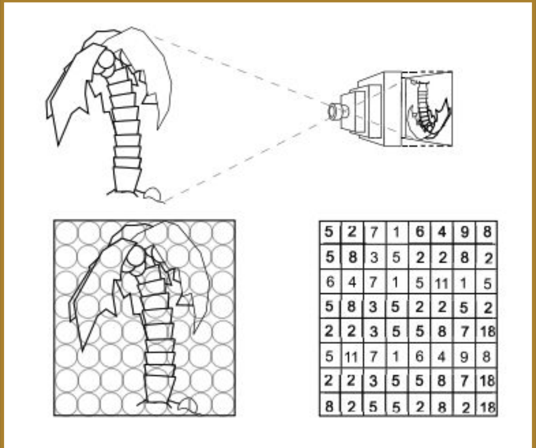
INPUT OUTPUT	IMAGE	DESCRIPTION
IMAGE	DIGITAL IMAGE PROCESSING	COMPUTER VISION
DESCRIPTION	COMPUTER GRAPHICS	ARTIFICIAL INTELLIGENCE

Source: GeeksForGeeks

Key Concepts

- **Digital Image:** A digital image is represented as a two-dimensional array of pixels, where each pixel contains intensity or colour values.

Source: [Clemson University]



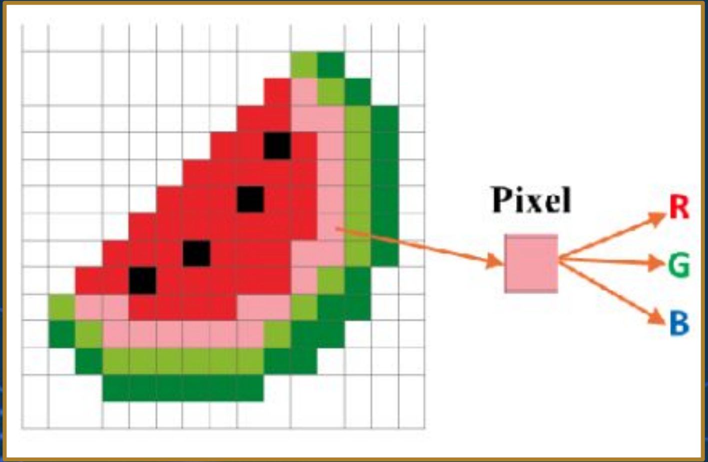
5	2	7	1	6	4	9	8
5	8	3	5	2	2	8	2
6	4	7	1	5	11	1	5
5	8	3	5	2	2	5	2
2	2	3	5	5	8	7	18
5	11	7	1	6	4	9	8
2	2	3	5	5	8	7	18
8	2	5	5	2	8	2	18

Figure 1.1: Capturing a 2D Continuous Image of a Scene

CoGrammar

Key Concepts

❖ **Pixel:** A pixel is the smallest unit of a digital image and is represented by **one or more numerical values**. In grayscale images, each pixel has a single intensity value, while in colour images, pixels typically have three values corresponding to the **red, green, and blue** colour channels.

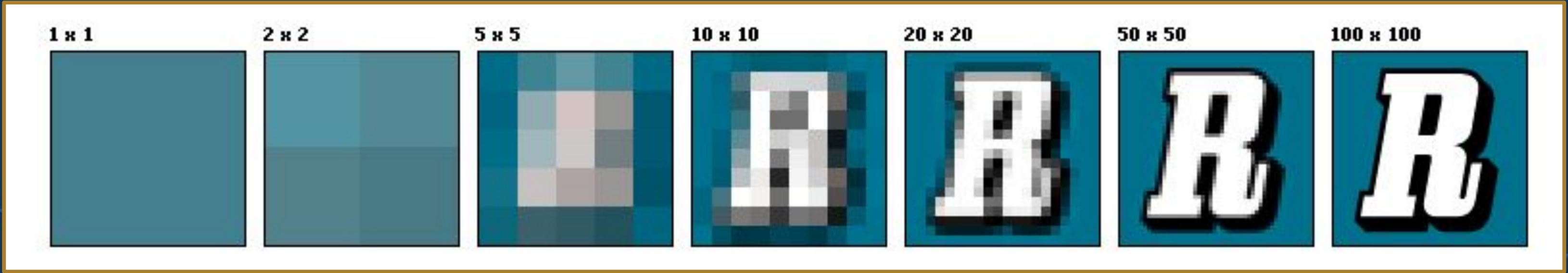


Source: [ResearchGate]

CoGrammar

Key Concepts

❖ **Image Resolution:** Image resolution refers to the number of pixels in an image, usually expressed in terms of **width and height** (e.g., 1024x768). Higher resolution images contain more pixels and provide **more detailed and clearer representations** of the visual content.

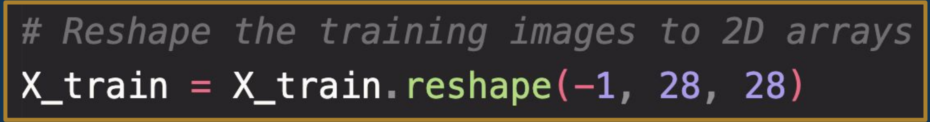


Source: [Wikipedia]

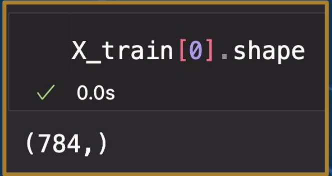
CoGrammar

Reshaping Images

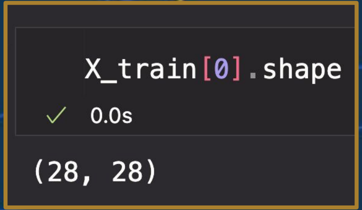
- ❖ The loaded image data is often flattened into a **1D array** for storage and processing purposes. However, for applying image processing techniques and visualisation, **it is necessary to reshape the data back into a 2D array representing the original image dimensions.**
- ❖ The -1 parameter in the reshape() function automatically calculates the number of images based on the total number of pixels.



```python
# Reshape the training images to 2D arrays
X_train = X_train.reshape(-1, 28, 28)
```



```
X_train[0].shape
✓ 0.0s
(784,)
```



```
X_train[0].shape
✓ 0.0s
(28, 28)
```

CoGrammar

Preprocessing

- Preprocessing techniques are applied to enhance the **quality of images**, **normalise pixel values**, and **remove any artifacts or noise** that may affect the performance of machine learning algorithms.

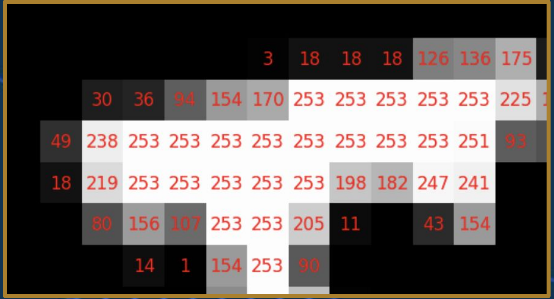
Preprocessing

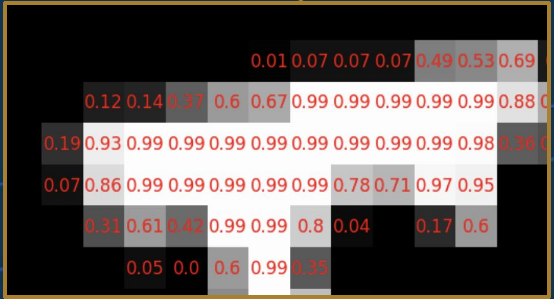
❖ Contrast stretching improves the contrast of an image by **spreading out the pixel intensities**. It can be achieved using the exposure module from the scikit-image library.

```python
# Apply contrast stretching
image_rescale = exposure.rescale_intensity(
    image, in_range=(0, 255), out_range=(0, 1)
)

# Clip the values to the valid range [0, 1]
image_rescale = np.clip(image_rescale, 0, 1)
```

This code calculates the 2nd and 98th percentile of pixel intensities and rescales the image intensity values to the range [0, 1] **(normalisation)**.





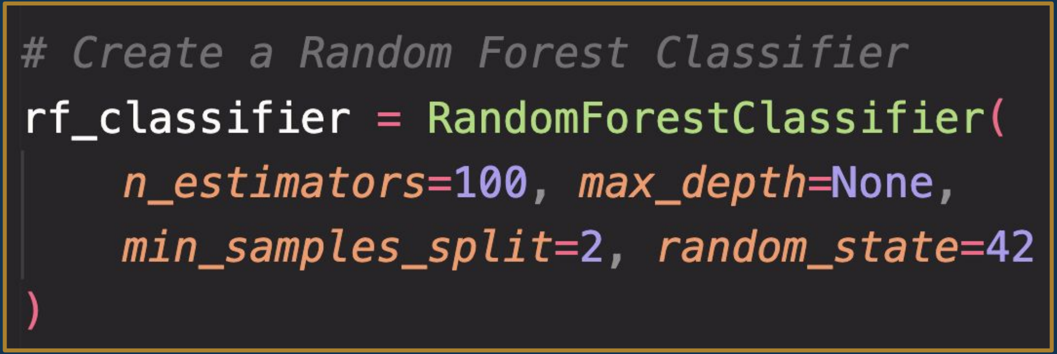
Random Forest Classifier

- The Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to make robust and accurate predictions. It is particularly well-suited for image classification tasks.
- The Random Forest Classifier has **several hyperparameters that can be tuned to optimise its performance, such as the number of trees (n_estimators), the maximum depth of each tree (max_depth), and the minimum number of samples required to split an internal node (min_samples_split)** → More on this at the end.

CoGrammar

Training

❖ To train the Random Forest Classifier, we first create an instance of the RandomForestClassifier class from the scikit-learn library, specifying the desired hyperparameters.

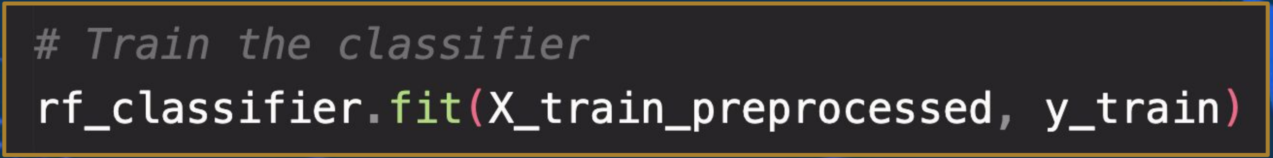


```python
# Create a Random Forest Classifier
rf_classifier = RandomForestClassifier(
    n_estimators=100, max_depth=None,
    min_samples_split=2, random_state=42
)
```

Training

❖ Next, **we fit the classifier to the preprocessed training data using the fit() method**. This involves providing the feature matrix (X_train_preprocessed) and the corresponding labels (y_train) to the classifier.

❖ During the training process, the Random Forest Classifier learns the underlying patterns and **relationships between the input features (pixel values) and the corresponding labels (digit classes)**.

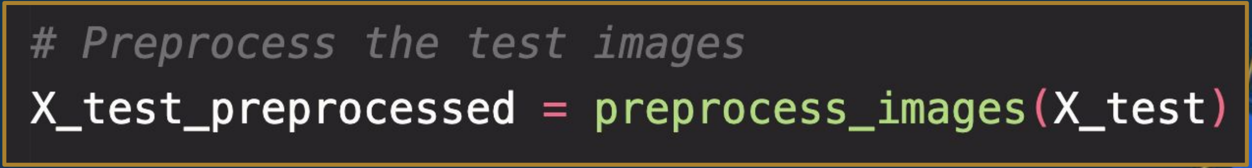


```python
# Train the classifier
rf_classifier.fit(X_train_preprocessed, y_train)
```

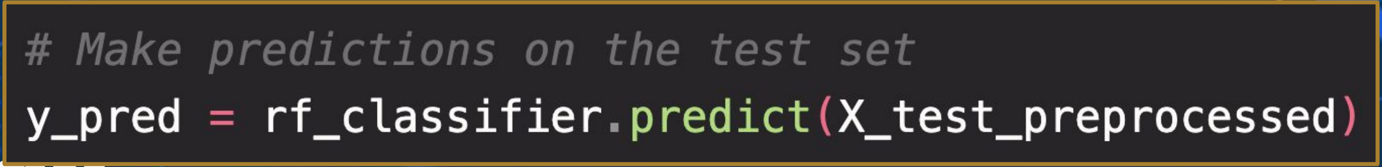
Making Predictions

❖ Once the Random Forest Classifier is trained, we can use it to make predictions on **new, unseen images**.

❖ To make predictions, we first **preprocess the test images in the same way as the training images**, applying the necessary reshaping and preprocessing steps.

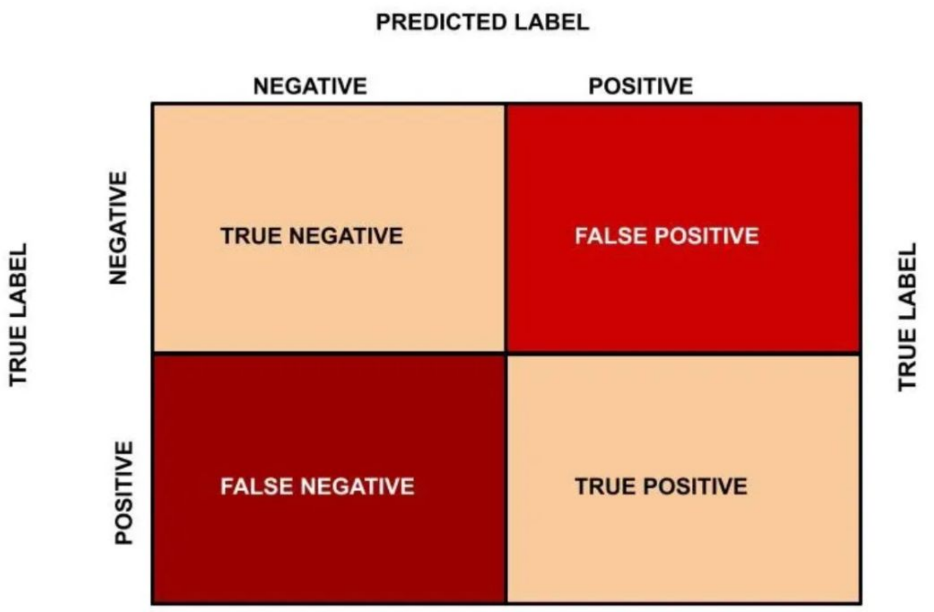


```python
# Preprocess the test images
X_test_preprocessed = preprocess_images(X_test)
```



```python
# Make predictions on the test set
y_pred = rf_classifier.predict(X_test_preprocessed)
```

Evaluation



		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	TRUE NEGATIVE	FALSE POSITIVE
	POSITIVE	FALSE NEGATIVE	TRUE POSITIVE

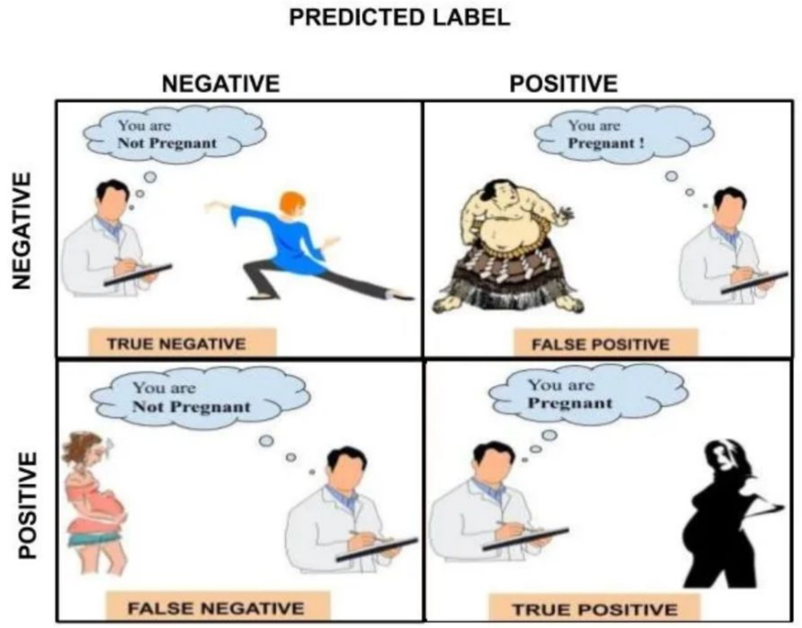


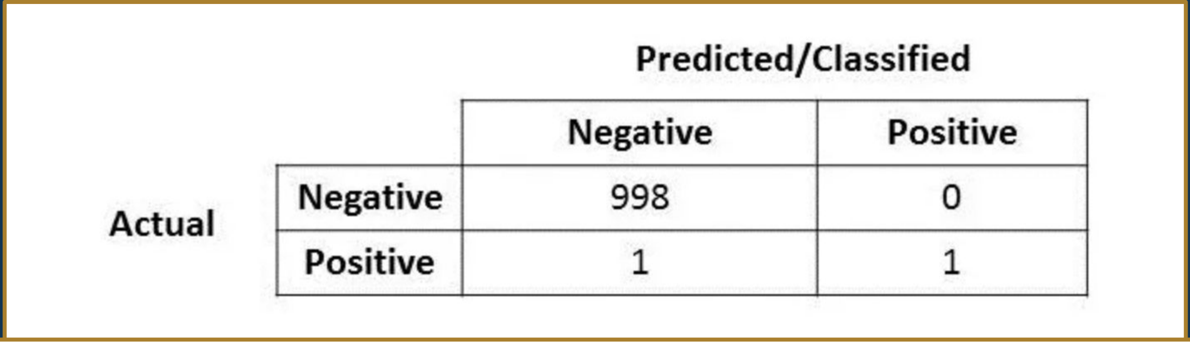
Figure 5: Confusion Matrix.

Source: Medium

Evaluation

❖ **Accuracy:** Accuracy measures the overall correctness of the model's predictions by calculating the **proportion of correctly classified samples out of the total number of samples**.

Source: Medium



		Predicted/Classified	
		Negative	Positive
Actual	Negative	998	0
	Positive	1	1

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

CoGrammar

Evaluation

❖ **Precision:** Precision focuses on the quality of the model's positive predictions. It is calculated as the ratio of true positive predictions to the total number of positive predictions (true positives + false positives). **High precision indicates that when the model predicts a specific class, it is more likely to be correct.**

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

$$= \frac{\textit{True Positive}}{\textit{Total Predicted Positive}}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

True Positive + False Positive = Total Predicted Positive

Source: Medium

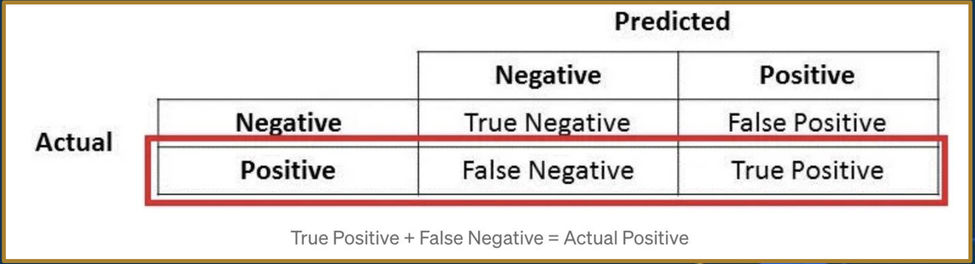
CoGrammar

Evaluation

❖ **Recall:** Recall, also known as sensitivity or true positive rate, measures the model's ability to correctly identify positive instances. It is calculated as the ratio of true positive predictions to the total number of actual positive instances (true positives + false negatives). **High recall indicates that the model is able to capture a large proportion of the positive instances.**

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

$$= \frac{\textit{True Positive}}{\textit{Total Actual Positive}}$$



		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

True Positive + False Negative = Actual Positive

Source: Medium

CoGrammar

Evaluation

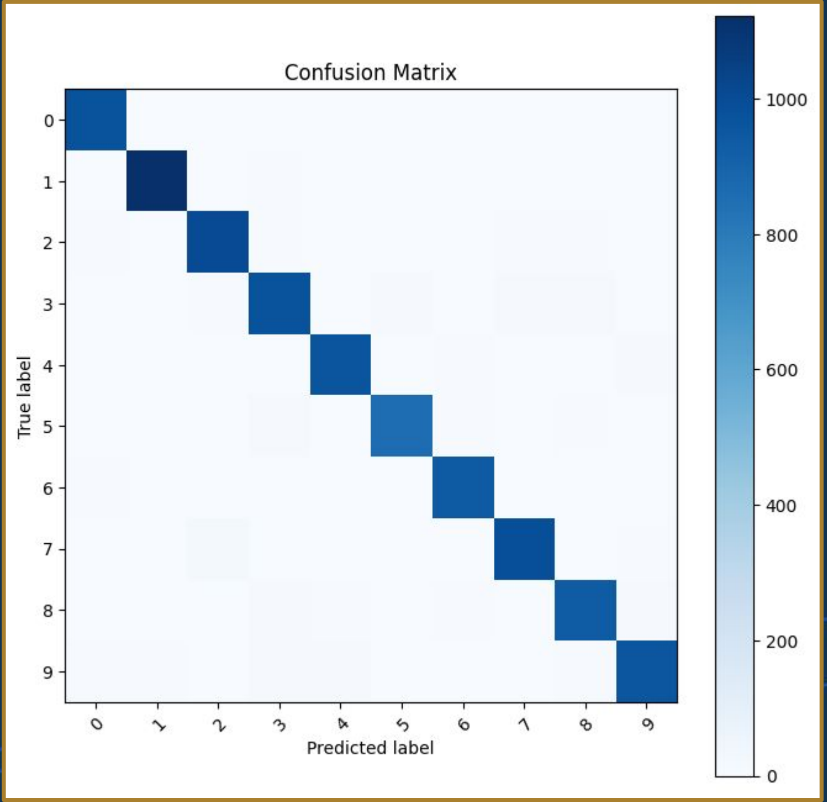
❖ **F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balanced measure that considers both the quality and completeness of the model's predictions. **It is especially useful when dealing with imbalanced datasets or when equal importance is given to precision and recall.**

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

CoGrammar

Confusion Matrix

❖ A confusion matrix is a powerful tool for visualising the **performance of a classification model** by providing a tabular summary of the model's predictions compared to the true labels.



CoGrammar

Hyperparameters

- Hyperparameters are the **adjustable settings** of a machine learning algorithm that are **not learned from the data but are set by the user before training**. These hyperparameters can significantly impact the performance and behaviour of the model.
- Proper tuning of hyperparameters is crucial for optimising the model's performance and achieving the best possible results. It involves **finding the right combination of hyperparameter values that strike a balance between model complexity and generalisation ability**.

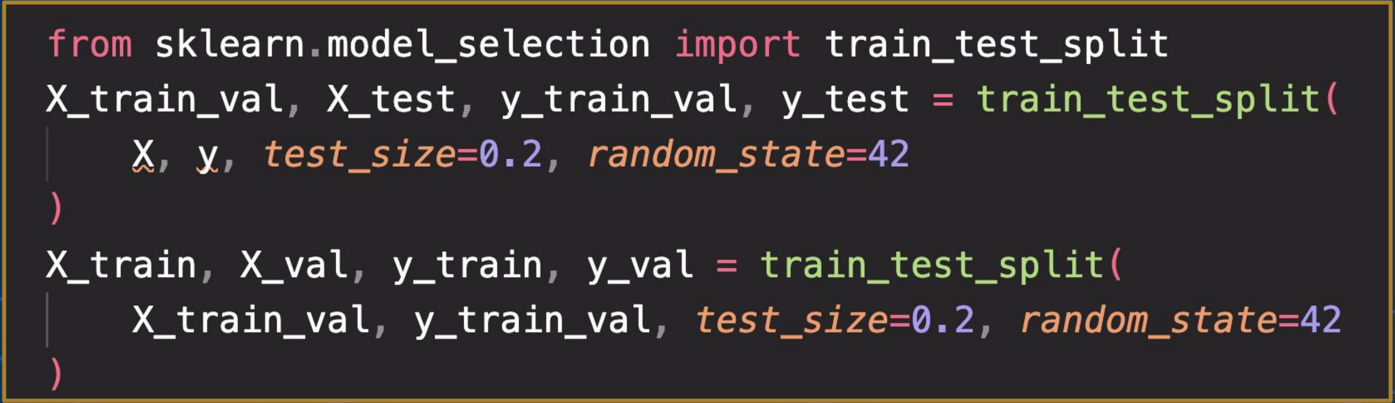
Data Splits

- ❖ To properly tune hyperparameters and evaluate the model's performance, it is essential to split the data into three sets: **training, validation, and test**.
- ❖ The training set is used to **train the model**, the validation set is used to **tune the hyperparameters and assess the model's performance during the tuning process**, and the test set is used to **evaluate the final model's performance on unseen data.**

CoGrammar

Data Splits

- In this example, we first split the data into training/validation (X_train_val, y_train_val) and test (X_test, y_test) sets using a 80-20 split. Then, we further split the training/validation set into training (X_train, y_train) and validation (X_val, y_val) sets using another 80-20 split.

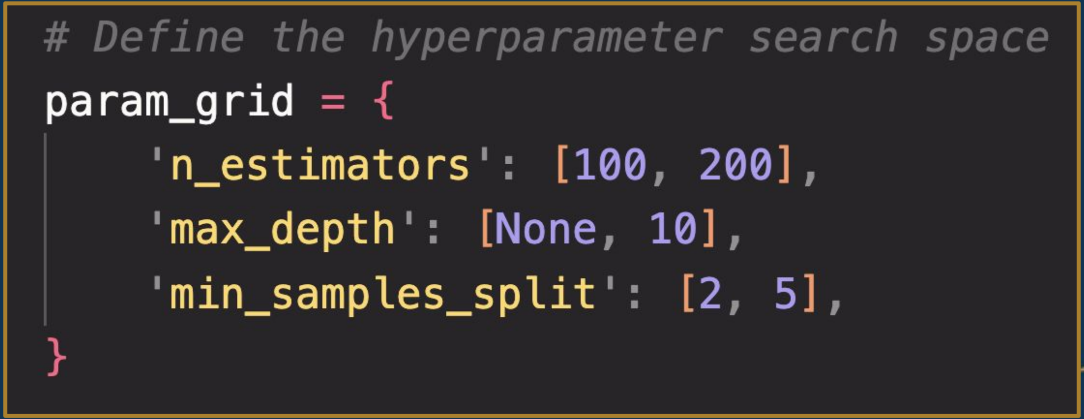


```python
from sklearn.model_selection import train_test_split
X_train_val, X_test, y_train_val, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
X_train, X_val, y_train, y_val = train_test_split(
    X_train_val, y_train_val, test_size=0.2, random_state=42
)
```

CoGrammar

Process

❖ Define the hyperparameter search space

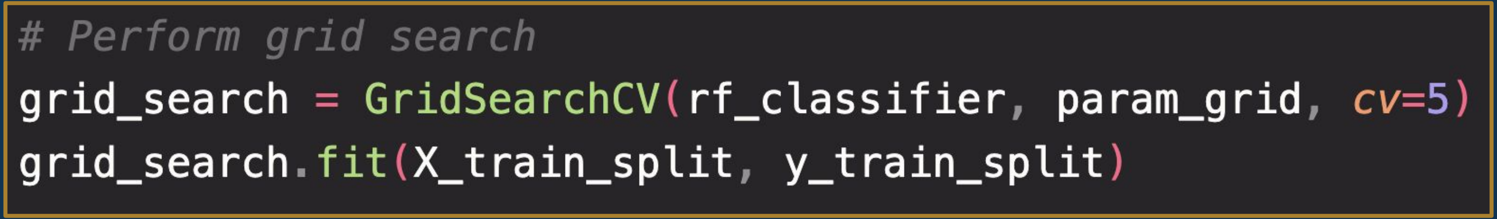


```
# Define the hyperparameter search space
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5],
}
```

CoGrammar

Process

❖ Create an instance of the model and perform a grid search

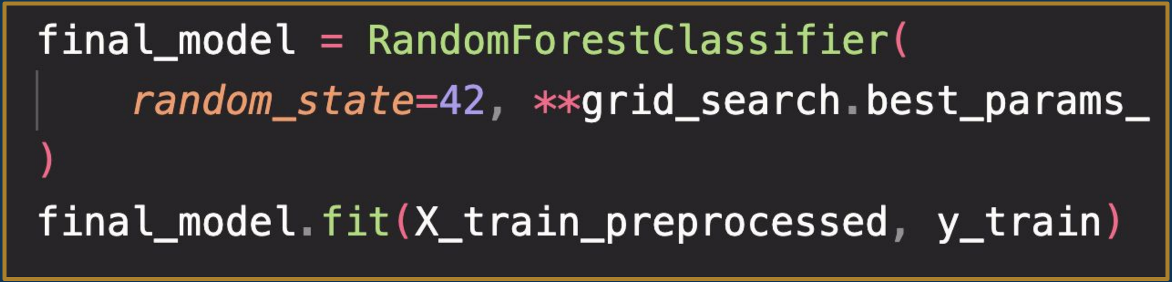


```python
# Perform grid search
grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(X_train_split, y_train_split)
```

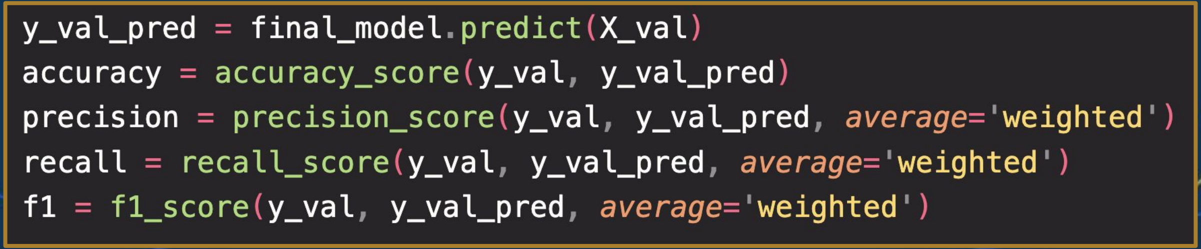
CoGrammar

Process

❖ Evaluate the model's performance on the validation set



```python
final_model = RandomForestClassifier(
    random_state=42, **grid_search.best_params_
)
final_model.fit(X_train_preprocessed, y_train)
```



```python
y_val_pred = final_model.predict(X_val)
accuracy = accuracy_score(y_val, y_val_pred)
precision = precision_score(y_val, y_val_pred, average='weighted')
recall = recall_score(y_val, y_val_pred, average='weighted')
f1 = f1_score(y_val, y_val_pred, average='weighted')
```

CoGrammar

Overfitting

- If the model achieves **high accuracy on the training set but performs poorly on the validation set**, it indicates overfitting.
- Example:
  - **Training Accuracy: 0.98**
  - **Validation Accuracy: 0.75**
- In this case, the model is **too complex** and has learned noise or specific patterns from the training data that do not generalise well to unseen data.
- To mitigate overfitting, you can try reducing the model complexity

CoGrammar

Good Fit

❖ If the model achieves **high accuracy on both the training and validation sets**, it indicates a good fit.

❖ Example:

➢ **Training Accuracy: 0.95**

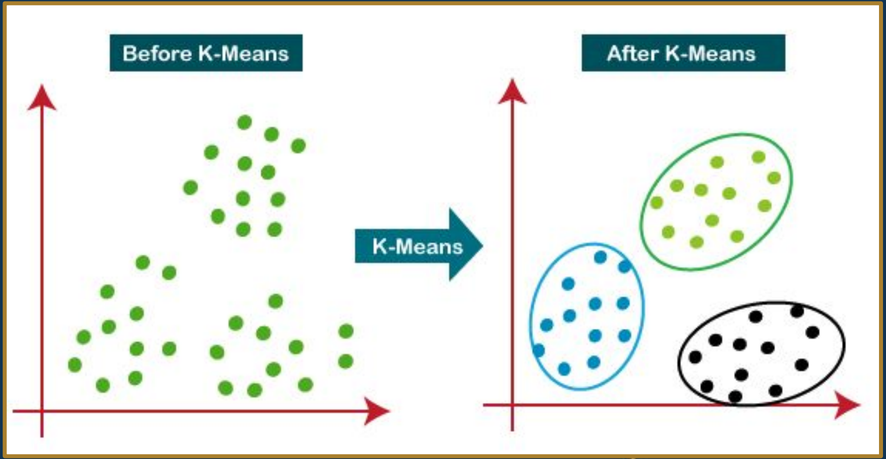
➢ **Validation Accuracy: 0.93**

❖ In this case, the model has found a **good balance between capturing the relevant patterns in the data and generalising well to unseen data**.

❖ You can proceed with evaluating the model on the test set to assess its final performance.

K-means clustering

- K-means clustering is an **unsupervised learning algorithm used to partition a dataset into K distinct clusters.**
- The goal is to **minimise the within-cluster variation** and **maximise the separation between clusters**.
- It is a popular technique for **exploratory data analysis** and **pattern discovery**.



Source: Javapoint

CoGrammar

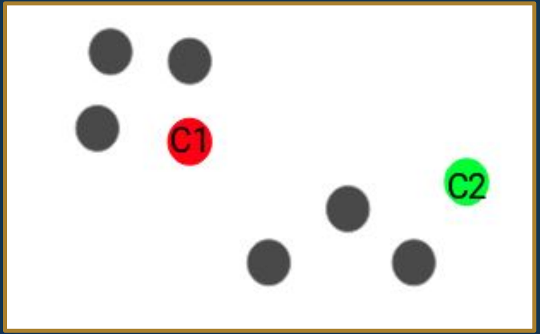
Some Applications

- **Customer Segmentation:** Grouping customers based on their purchasing behaviour, demographics, or preferences.
- **Image Compression:** Reducing the colour palette of an image by clustering similar colours.
- **Anomaly Detection:** Identifying unusual or anomalous data points that do not belong to any cluster.
- **Document Clustering:** Grouping similar documents based on their content or topics.

CoGrammar

Step 1

- Initialise **K centroids randomly** or using a specific strategy (e.g., K-means++).

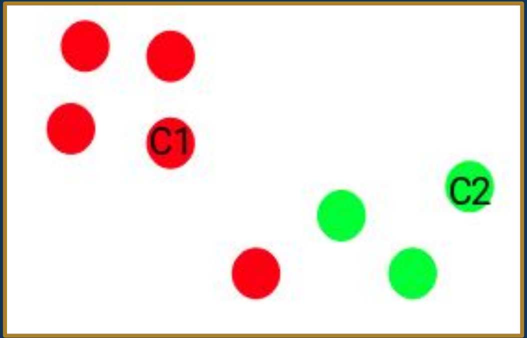


Source: Analytics Vidhya

CoGrammar

Step 2

- Assign each data point to the **nearest centroid** based on a distance metric (e.g., Euclidean distance).

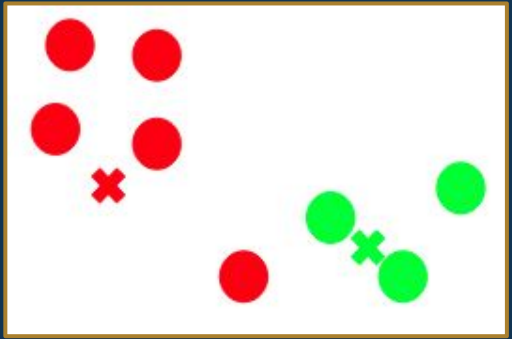


Source: [Analytics Vidhya]

CoGrammar

Step 3

- Update the centroids by calculating the **mean of the data points assigned to each cluster**.

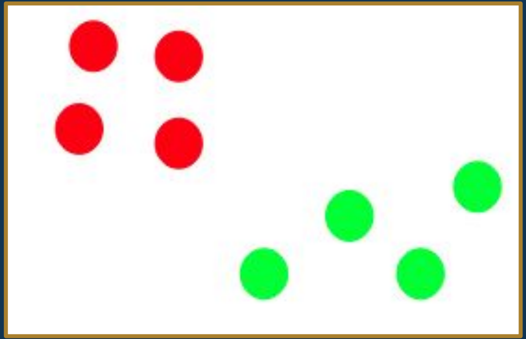


Source: Analytics Vidhya

CoGrammar

Step 4

❖ Repeat steps 2 and 3 **until convergence (centroids no longer change)** or a **maximum number of iterations** is reached.



Source: Analytics Vidhya

CoGrammar

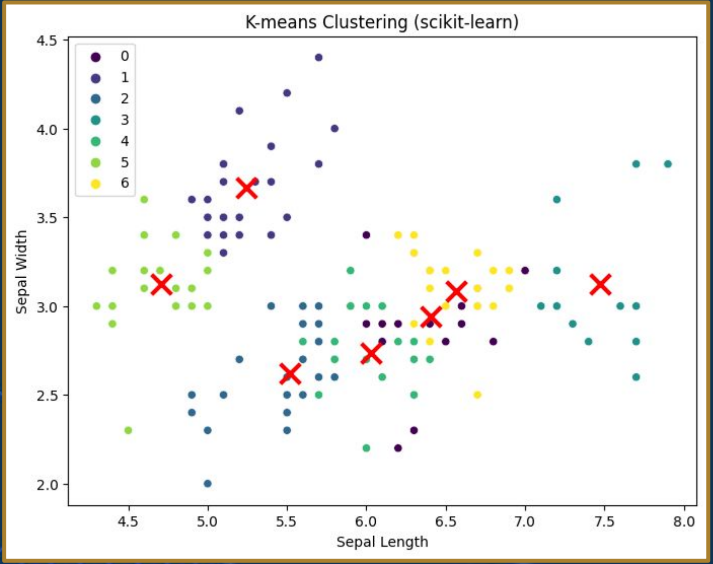
Goals

❖ The algorithm aims to minimise the **sum of squared distances** between data points and their assigned centroids.

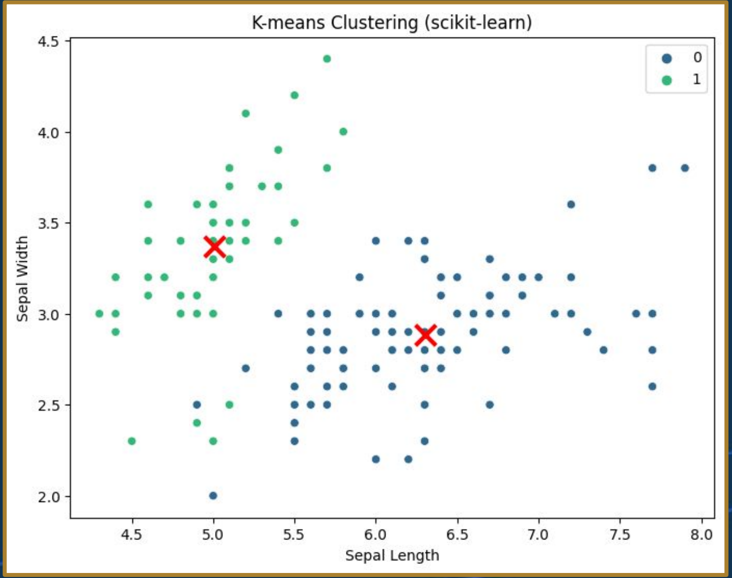
CoGrammar

Goals

❖ Selecting an appropriate number of clusters (K) is crucial for effective clustering.



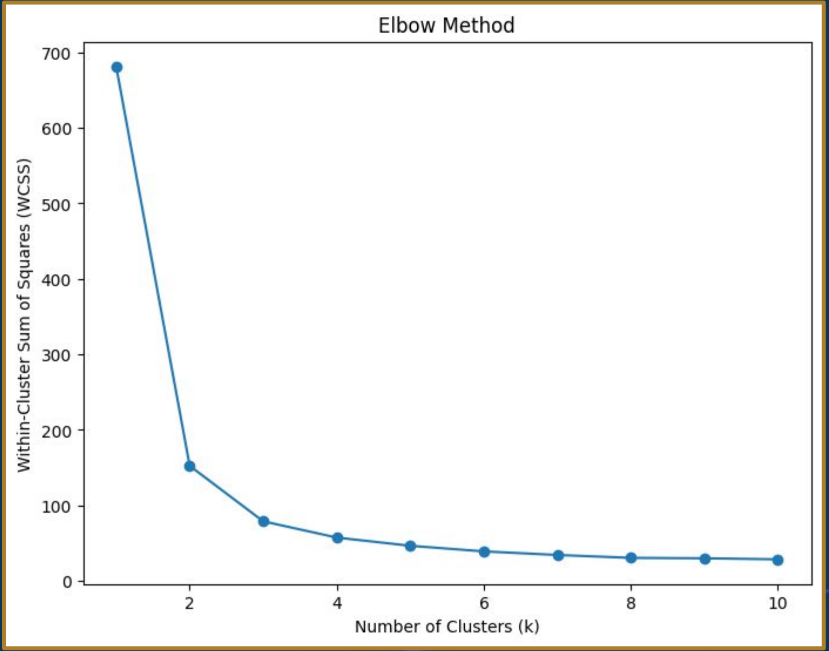
VS



CoGrammar

Techniques

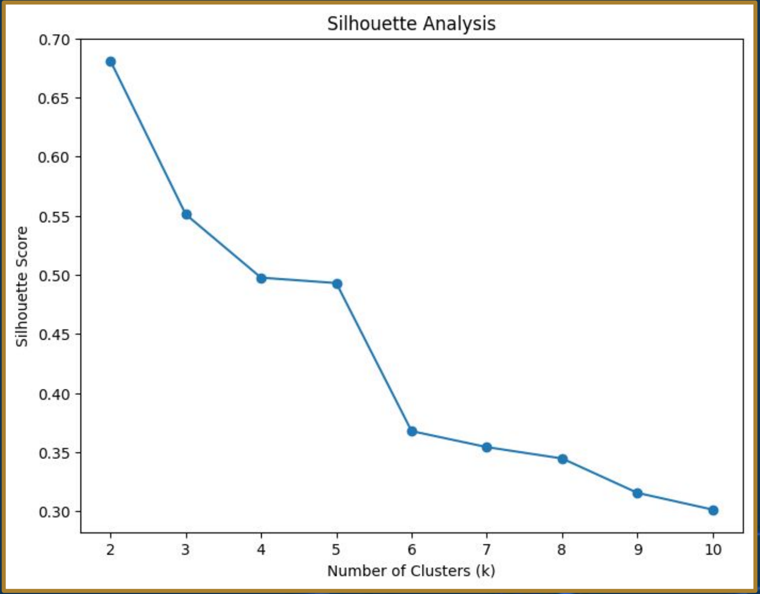
- Elbow Method:
  - Plot the within-cluster sum of squares (WCSS) against different values of K.
  - Look for the **"elbow point"** where the rate of decrease in WCSS slows down significantly.



CoGrammar

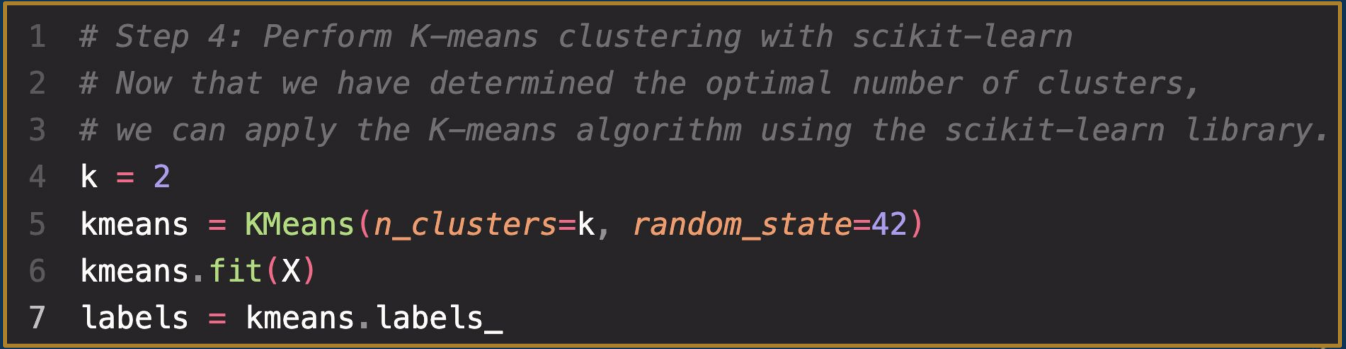
Techniques

- Silhouette Analysis:
  - Measure the quality of clustering based on the compactness and separation of clusters.
  - Calculate the silhouette score for each data point and average them for different values of K.
  - Choose the K value that **maximises the average silhouette score.**



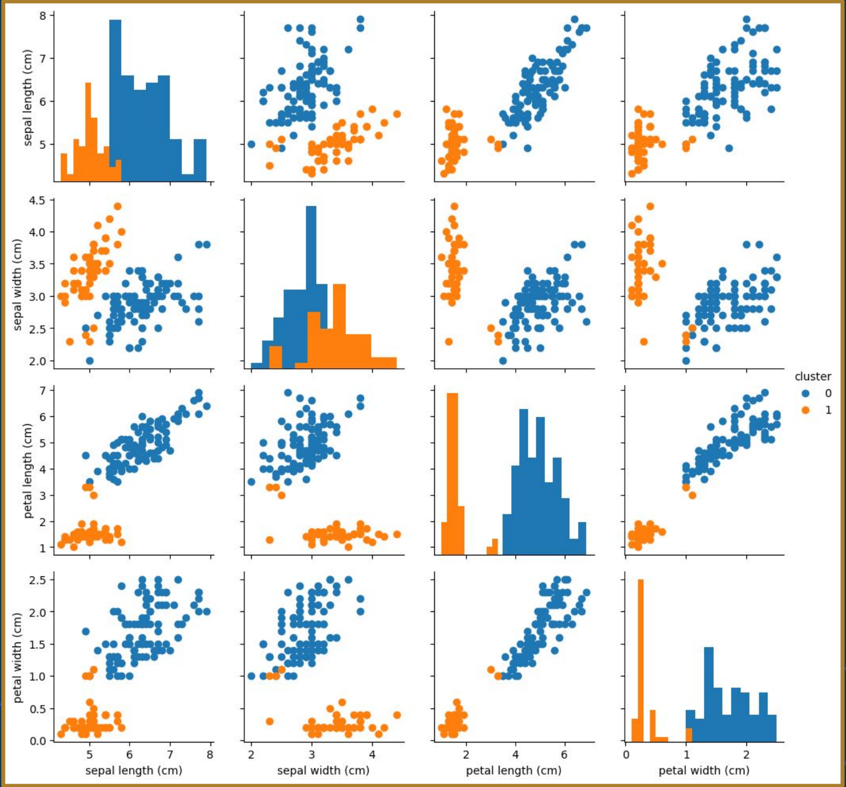
CoGrammar

Implementation



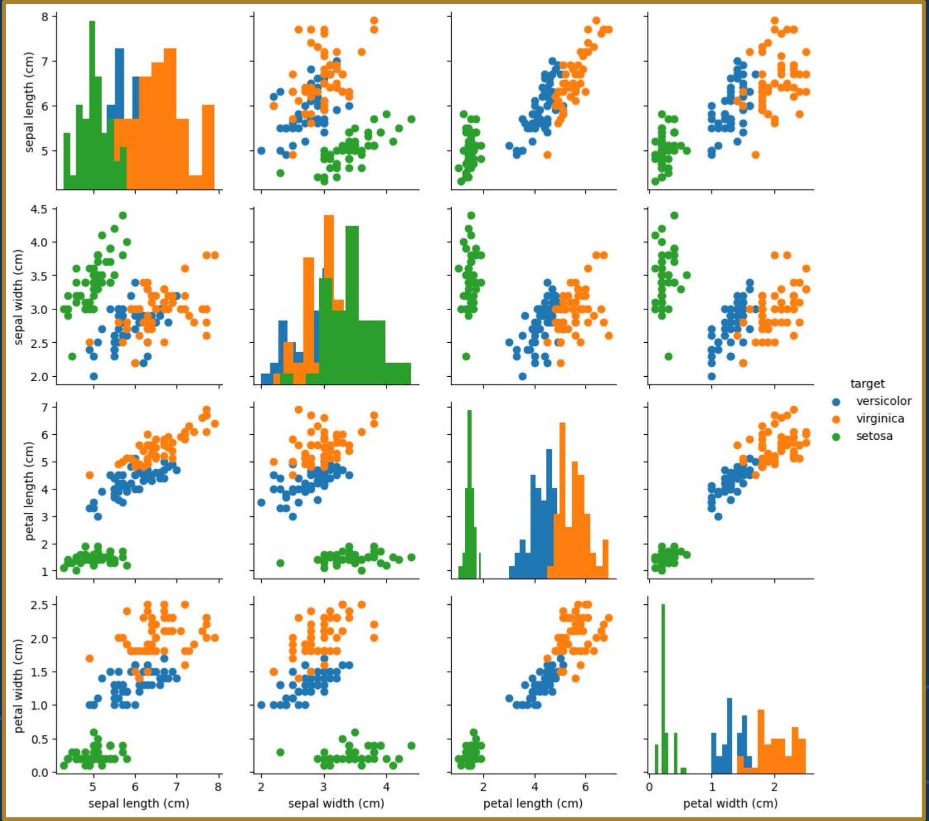
```python
# Step 4: Perform K-means clustering with scikit-learn
# Now that we have determined the optimal number of clusters,
# we can apply the K-means algorithm using the scikit-learn library.
k = 2
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
```

Calculated Clusters



CoGrammar

Actual Species



CoGrammar

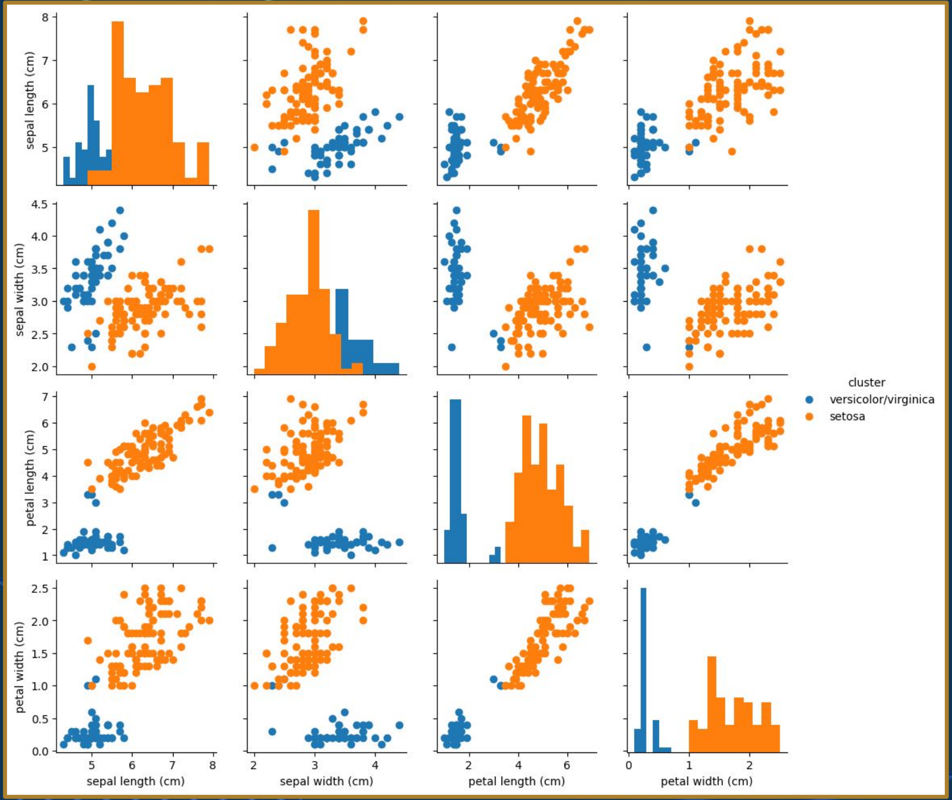
Importance

❖ After obtaining the clustering results, it's important to **interpret and analyse the clusters.**

❖ Examine the characteristics and centroids of each cluster to **understand their distinguishing features.**

❖ **Assign meaningful labels or descriptions to the clusters based on domain knowledge and the patterns observed in the data.**

Interpretation



It is easy to distinguish between setosa and the other two species, which could guide the type of model you build later on. It also indicates that differentiating between versicolor and virginica would be more difficult, although could be made easier when choosing petal metrics instead of sepal metrics given the separation is more apparent.

CoGrammar

Limitations

- The algorithm is **sensitive to the initial positions of the centroids**, which can lead to different results in each run.
- It assumes that **clusters are spherical and have equal sizes, which may not always be the case in real-world data**.
- **Outliers and noise** in the data can affect the clustering results.
- **Scaling and normalisation of features** may be necessary to ensure equal contribution to the distance calculations.
- Handling **high-dimensional data** can be challenging due to the curse of dimensionality.

CoGrammar

Task Walkthrough

Imagine you work at a space research agency analyzing satellite images. Your mission is to identify different types of land cover (e.g., water bodies, vegetation, urban areas) using machine learning.

You will:

- ❖ Load a satellite image dataset and convert it into numerical values.
- ❖ Preprocess the image by converting it to grayscale and extracting features.
- ❖ Apply K-Means Clustering to categorize different land types.
- ❖ Determine the optimal number of clusters using the Elbow Method.
- ❖ Visualize the clustered image to interpret land patterns.

Why is it important to preprocess images before applying machine learning?

A. It reduces noise and improves model accuracy
B. It makes images more visually appealing
C. It removes all pixel values
D. It decreases file size

What does the Elbow Method help determine in K-Means Clustering?

A. The best accuracy score
B. The optimal number of clusters
C. The best feature for prediction
D. The number of training iterations

Summary

★ **Image Processing** is used to convert images into numerical data for machine learning.

★ **K-Means Clustering** is an unsupervised learning technique that groups similar data points.

★ **The Elbow Method and Silhouette Score** help determine the optimal number of clusters.

★ Combining these techniques allows us to **classify different land types in satellite images.**

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education