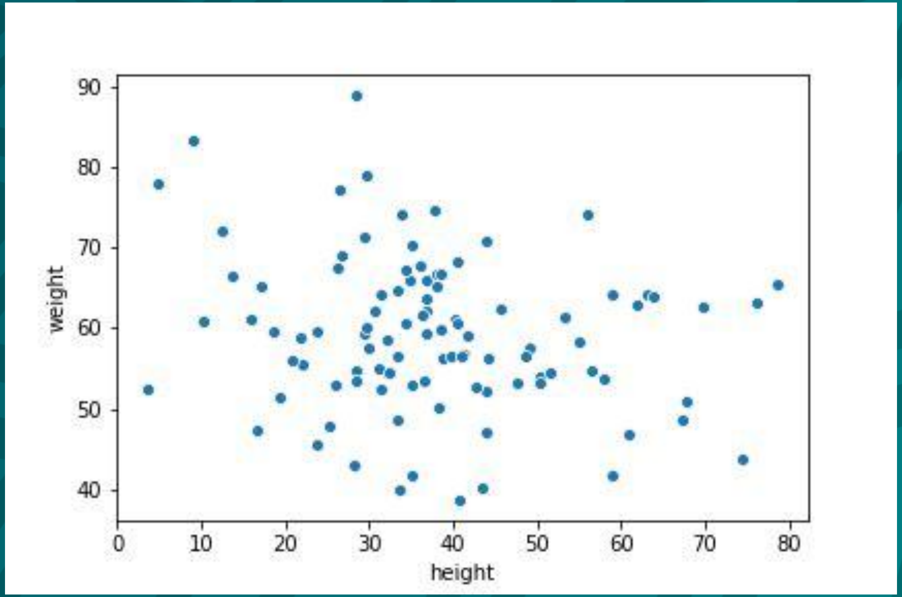
Warm Up

- Even if you don't have any prior knowledge on machine learning, try to use your reasoning skills to figure out whether the given scenario is a form of supervised learning or unsupervised learning.
- 1. In a set of data containing every fish currently on planet Earth, a machine learning model sorts each one by determining their similarity in features such as their general shape or size.
- 2. An AI is given the task of determining the resulting 3D shape of proteins based on the sequence of amino acids that make up the protein. As a starter, it is fed the 3D structure of existing amino acid sequences to give it a reference for the rest.
- 3. A scatter plot of data (image to the left if you don't know what a scatter plot looks like) plots the relationship between an individuals' height and their weight. An AI was assigned to filter and cluster the data into groups alike to each other.
- 4. In reCAPTCHA tests, a person is asked to click on images relating to a certain keyword; for example, you may have been asked by google to "click on all images with cars in them." This is a machine learning model that learns based on the data fed by millions of users around the world proving that they are not robots.



Answers

1. This is unsupervised learning. The machine learning model is given features to differentiate by, but no given labels or categories to sort the fishes by; it simply sorts on its own by similarity. There isn't anyone classifying the fishes for the model to see and learn by example, making it unsupervised.

2. This is supervised learning, as the model is fed a training dataset, an initial sample of intended outputs by human intervention that teaches the AI how other proteins might fold based on the training set.

3. This is unsupervised learning, as the machine learning model is given unlabeled data with no prior classifications. It is up to it to determine where lines between clusters are drawn, with no human intervention intended to point it in the right direction.

4. This is supervised learning. The model is taking human-interpreted, mostly-accurate data fed by you, the user, and using it in order to classify the various objects it asks you to spot. By answering reCAPTCHAS, you are refining and teaching a machine learning model!

Supervised vs. Unsupervised Learning

Supervised Learning

Supervised learning is a machine learning approach defined by its use of labeled datasets. These datasets are designed to train or “supervise” algorithms into classifying data or predicting outcomes accurately, using labeled inputs and outputs specified by humans. Here are some prominent types of supervised learning:

- Classification, which uses an algorithm to accurately assign test data into specific categories, such as separating apples from oranges.
- Regression, which uses an algorithm to understand the relationship between dependent and independent variables. Regression models are helpful for predicting numerical values based on different data points.

Unsupervised Learning

Unsupervised learning uses machine learning algorithms to analyze and cluster *unlabeled* data sets. These algorithms discover hidden patterns in data without the need for human intervention. Here are some prominent types of unsupervised learning:

- Clustering is a data mining technique for grouping unlabeled data based on their similarities or differences.
- Association uses different rules to find relationships between variables in a given dataset. These methods are frequently used for market basket analysis and recommendation engines.

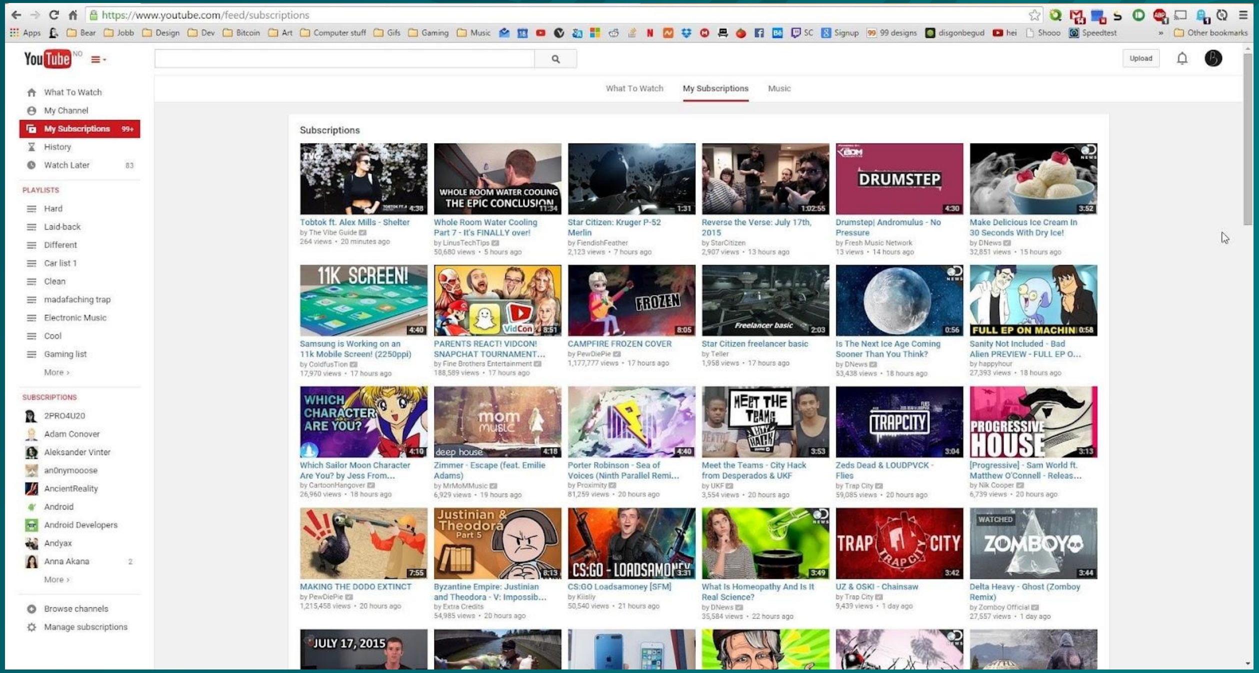
Basically, the main differences boil down to whether labels are used in the datasets.

This project will focus on clustering, specifically K-Means Clustering, a relatively simple, unsupervised machine learning algorithm that tries to partition data points into separate groups. Clustering is a vital technique with applications in areas such as social network analysis, search result grouping, medical imaging, and anomaly detection.

Real Life Example

The YouTube recommendation engine, which suggests a wide variety of videos from content creators to users, is a great example of how clustering algorithms can cluster videos by popularity, and then suggest to the user a mix of unpopular and popular videos.

So without further ado, let's begin with our journey into basic artificial intelligence!



The Initial Analysis

First of all, because some actions that we take won't cause critical errors but will instead yield warnings, we'll import Python's built-in warnings class and ignore them. These are not errors and will not stop your program, but may become annoying.

```
# lets import the warnings library so that we can avoid warnings
import warnings
warnings.filterwarnings('ignore')
```

The Initial Analysis

Now that we've gotten warnings out of the way, we'll begin choosing our data that we want to analyze. For this project, we want to perform clusters of customers who share similar behavior; for that, we'll select the columns Spending Score and Annual Income.

When clustering, we always pick groups of data that may share traits to hopefully guarantee an observable relationship. You wouldn't choose to use AI to find the relationship between the number of lizards born per year and the number of cars manufactured per year, right? After all, there's clearly no relationship there.

We used pandas' loc function to obtain the columns of data we wanted.

```
# Lets select the Spending score, and Annual
Income Columns from the Data
x = data.loc[:, ['Spending Score (1-100)',
'Annual Income (k$)']].values
```

Cont.

We will also write this up:

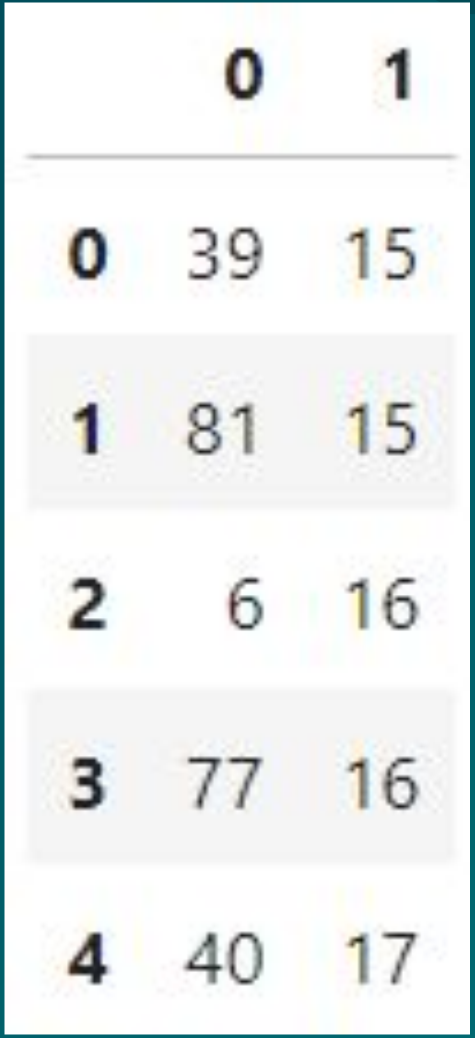
```
# let's check the shape of x
print(x.shape)
```

The shape function returns a tuple (an immutable iterable, think of it as a type of list) in the format (number of rows, number of columns). In this case, you will get (200, 2).

As we did at the very start of this project, we'll check our values just to be sure…

```
x_data   = pd.DataFrame(x)
x_data.head()
```

Result



	0	1
0	39	15
1	81	15
2	6	16
3	77	16
4	40	17

The Elbow Method

Now that we've picked out our desired data and made sure of its accuracy, let's get down to business.

Here's a great video on KMeans clustering that we're going to watch first:
https://www.youtube.com/watch?v=4b5d3muPQmA

And here's some quick background info on important terminology:

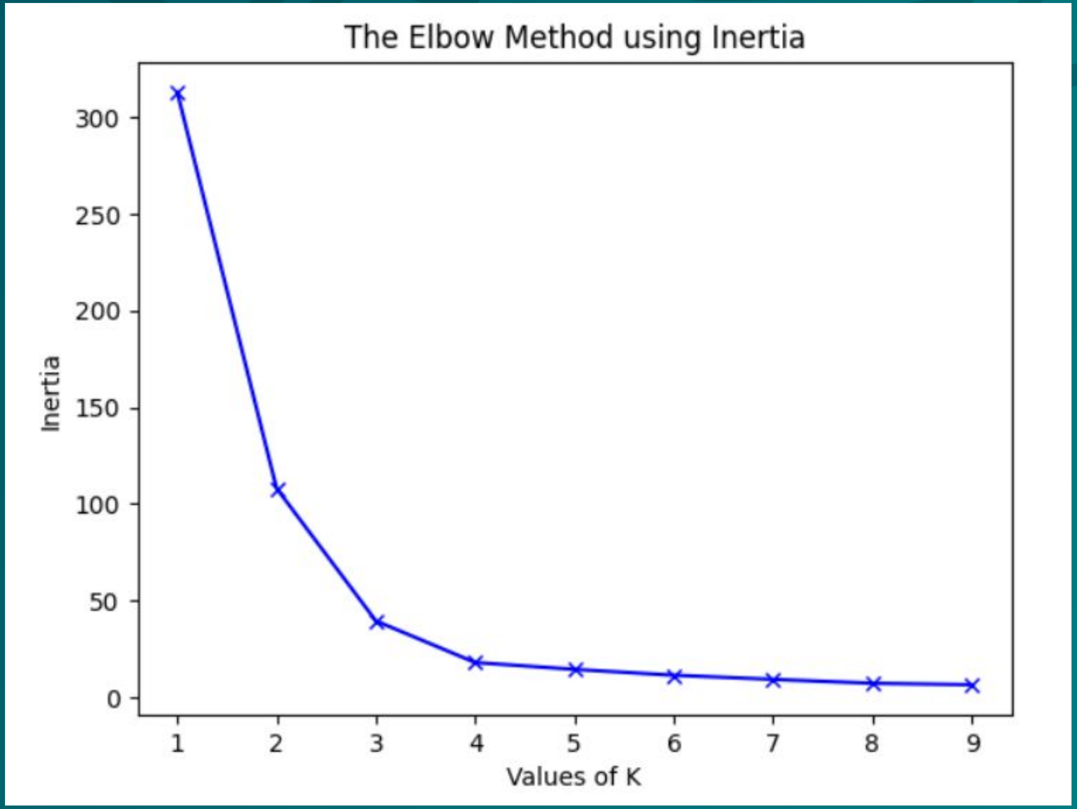
- Centroids are data points within the data set that are closest to every data point within their respective cluster. They are what determine the number of clusters as they are the centers of clusters.
- The WCSS value, or within-cluster sum of squares value, is the sum of square distances between the centroids and each point.

The Elbow Method is a way of finding out the optimal number of clusters (k) to partition a given dataset into. To determine the optimal k-value, we continuously iterate for k = 1 to k = n. For every value of k, we calculate the within-cluster sum of squares (WCSS) value, found by running the KMeans algorithm with k clusters. Then, we graph every value of k against every WCSS value.

The Elbow Method

Here's an example of an Elbow Method graph. The "elbow" of this graph can be spotted at k=4. Beyond values of k = 4, the WCSS decreases in a linear fashion. Thus, we can conclude that the optimal number of clusters in this graph is k = 4.

That's a *lot* of information to digest. Let's go through it in Python step-by-step:



The Elbow Method

First, we'll import the KMeans method from the scikit-learn package we installed earlier. If you recall, the scikit-learn package contains the tools for us to utilize an AI model with our analysis. We'll be using the package's functions to do the heavy lifting instead of coding a machine learning model from scratch. Building on existing tools–that's what programming is all about!

```
from sklearn.cluster import KMeans
```

The Elbow Method

Next, we need to create an array of WCSS values, and fill it with calculated WCSS values.

Here's the breakdown of everything going on in this codeblock.

- The for-loop creates 10 data points for our Elbow Method graph, from k = 1 to k = 10
- The KMeans function's parameters specify our desired settings (in order):
    - Create i clusters/centroids (i is the current iteration of the for-loop)
    - Select initial centroids based on a probability distribution and each points' inertia (you don't have to know how this setting works)
    - Sets the max number of iterations that the KMeans algorithm will run (300 times)
    - Determines the number of times the KMeans algorithm is run with different centroid seeds (10 times)
    - Select that no randomness will decide the initial centroid locations
- The fit method trains the model with the dataset with the settings we just applied
- Finally, we append the inertia (same as the WCSS value) of the ith iteration of the trained model to the list of WCSS values

```
for i in range(1, 11):
    km = KMeans(n_clusters = i, init =
'k-means++', max_iter = 300, n_init = 10,
random_state = 0)
    km.fit(x)
    wcss.append(km.inertia_)
```

The Elbow Method

Essentially, we're creating and calculating the WCSS values for a KMeans clustering algorithm model with i clusters, where i is an integer ranging from 1 to 10, inclusive. This way, we can plot them all on a curve.

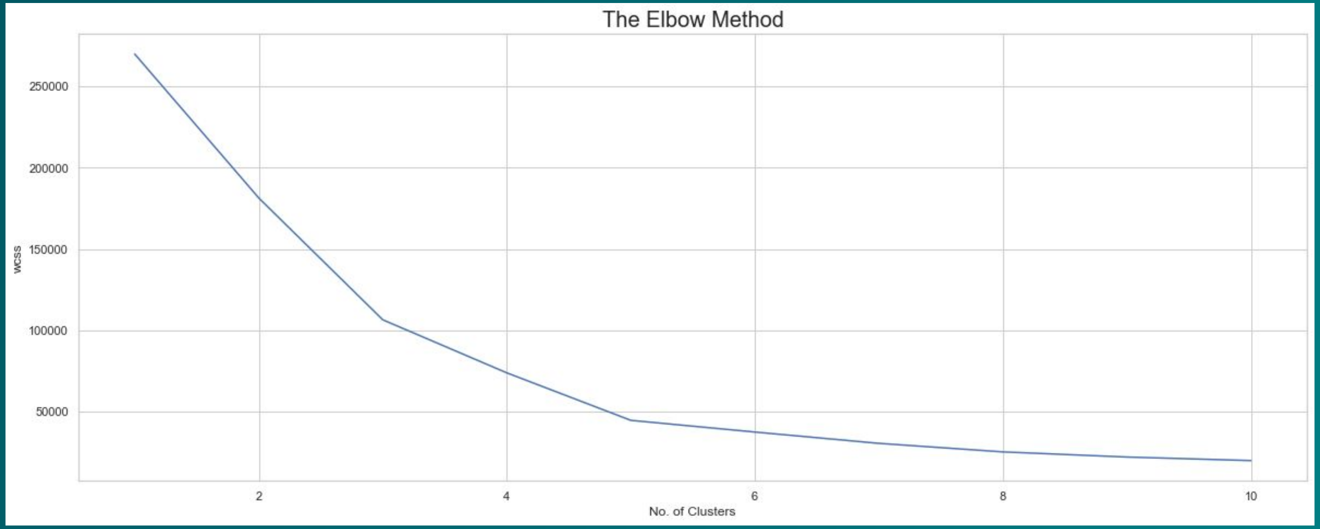
Let's do just that. Here's what we wrote…

```
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method',
fontsize = 20)
plt.xlabel('No. of Clusters')
plt.ylabel('wcss')
plt.show()
```

Result

We can see that the elbow of the curve is at k = 5 clusters, since the curve decreases almost linearly after k = 5. The optimal number of clusters/centroids to put into our clustering model is thus 5.

When we want to run our model, we simply call the KMeans function with the optimal number of clusters, as shown below.



```
km = KMeans(n_clusters = 5, init = 'k-means++',
max_iter = 300, n_init = 10, random_state = 0)
```

Homework

Practice applying the Elbow Method on the Mall Customer dataset's Spending Score and Annual Income data. Determine the optimal cluster number for these variables using the Elbow Method.

Thank You!