

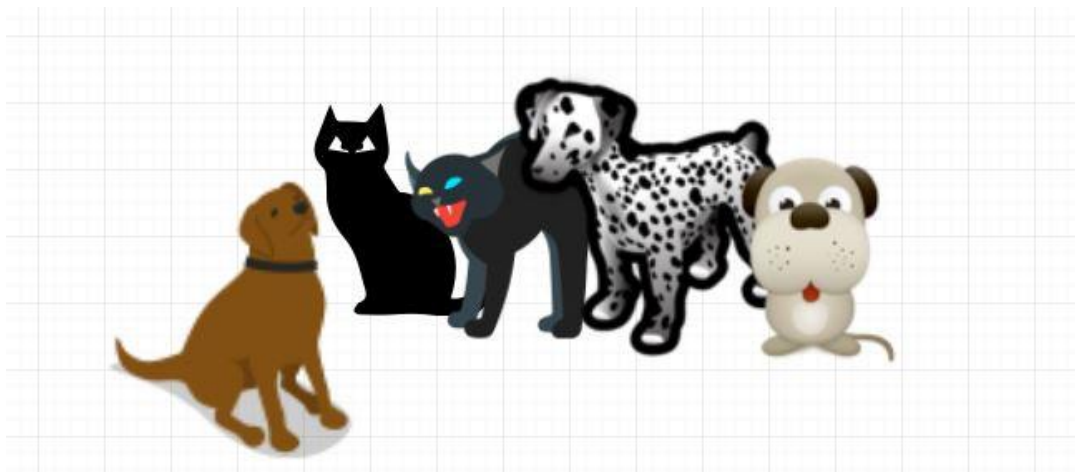
Week 9

Artificial intelligence and Machine Learning

What is Unsupervised Learning?

- **Unsupervised Learning** is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the unlabelled data.
- **Unsupervised Learning Algorithms** allow users to perform more complex processing tasks compared to supervised learning. Although, unsupervised learning can be more unpredictable compared with other natural learning methods. Unsupervised learning algorithms include clustering, anomaly detection, neural networks, etc.
- Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.
- Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.
- Unlike supervised learning, no teacher is provided that means no training will be given to the machine.
- Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.

For instance, suppose it is given an image having both dogs and cats which it has never seen.



- Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats'.
- But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts.
- The first may contain all pics having dogs in them and the second part may contain all pics having cats in them. Here you didn't learn anything before, which means no training data or examples.
- It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.
- Unsupervised learning is classified into two categories of algorithms:
- **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Common unsupervised learning approaches

- Unsupervised learning models are utilized for three main tasks—clustering, association, and dimensionality reduction. Below we'll define each learning method and highlight common algorithms and approaches to conduct them effectively.

Clustering

- Clustering is a data mining technique which groups unlabeled data based on their similarities or differences.
- Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information.
- Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic.

Exclusive and Overlapping Clustering

- Exclusive clustering is a form of grouping that stipulates a data point can exist only in one cluster.
- This can also be referred to as "hard" clustering. The K-means clustering algorithm is an example of exclusive clustering.
 - **K-means clustering** is a common example of an exclusive clustering method where data points are assigned into K groups, where K represents the number of clusters based on the distance from each group's centroid.

- The data points closest to a given centroid will be clustered under the same category.
- A larger K value will be indicative of smaller groupings with more granularity whereas a smaller K value will have larger groupings and less granularity.
- K-means clustering is commonly used in market segmentation, document clustering, image segmentation, and image compression.
- **Overlapping clusters** differs from exclusive clustering in that it allows data points to belong to multiple clusters with separate degrees of membership.
- “Soft” or fuzzy k-means clustering is an example of overlapping clustering.

Hierarchical clustering

- Hierarchical clustering, also known as hierarchical cluster analysis (HCA), is an unsupervised clustering algorithm that can be categorized in two ways; they can be agglomerative or divisive.
- **Agglomerative clustering** is considered a “bottoms-up approach.” Its data points are isolated as separate groupings initially, and then they are merged together iteratively on the basis of similarity until one cluster has been achieved. Four different methods are commonly used to measure similarity:
 1. **Ward’s linkage:** This method states that the distance between two clusters is defined by the increase in the sum of squared after the clusters are merged.
 2. **Average linkage:** This method is defined by the mean distance between two points in each cluster
 3. **Complete (or maximum) linkage:** This method is defined by the maximum distance between two points in each cluster
 4. **Single (or minimum) linkage:** This method is defined by the minimum distance between two points in each cluster
- **Euclidean distance** is the most common metric used to calculate these distances; however, other metrics, such as Manhattan distance, are also cited in clustering literature.
- **Divisive clustering** can be defined as the opposite of agglomerative clustering; instead it takes a “top-down” approach.
- In this case, a single data cluster is divided based on the differences between data points.
- Divisive clustering is not commonly used, but it is still worth noting in the context of hierarchical clustering.

- These clustering processes are usually visualized using a dendrogram, a tree-like diagram that documents the merging or splitting of data points at each iteration.

Probabilistic clustering

- A probabilistic model is an unsupervised technique that helps us solve density estimation or “soft” clustering problems.
- In probabilistic clustering, data points are clustered based on the likelihood that they belong to a particular distribution.
- The Gaussian Mixture Model (GMM) is the one of the most commonly used probabilistic clustering methods.
- **Gaussian Mixture Models** are classified as mixture models, which means that they are made up of an unspecified number of probability distribution functions.
- GMMs are primarily leveraged to determine which Gaussian, or normal, probability distribution a given data point belongs to.
- If the mean or variance are known, then we can determine which distribution a given data point belongs to.
- However, in GMMs, these variables are not known, so we assume that a latent, or hidden, variable exists to cluster data points appropriately.
- While it is not required to use the Expectation-Maximization (EM) algorithm, it is a commonly used to estimate the assignment probabilities for a given data point to a particular data cluster.

Association Rules

- An association rule is a rule-based method for finding relationships between variables in a given dataset.
- These methods are frequently used for market basket analysis, allowing companies to better understand relationships between different products. Understanding consumption habits of customers enables businesses to develop better cross-selling strategies and recommendation engines.
- Examples of this can be seen in Amazon’s “Customers Who Bought This Item Also Bought” or Spotify’s "Discover Weekly" playlist.
- While there are a few different algorithms used to generate association rules, such as Apriori, Eclat, and FP-Growth, the Apriori algorithm is most widely used.

Dimensionality reduction

- While more data generally yields more accurate results, it can also impact the performance of machine learning algorithms (e.g. overfitting) and it can also make it difficult to visualize datasets.

- Dimensionality reduction is a technique used when the number of features, or dimensions, in a given dataset is too high.
- It reduces the number of data inputs to a manageable size while also preserving the integrity of the dataset as much as possible.
- It is commonly used in the preprocessing data stage, and there are a few different dimensionality reduction methods that can be used, such as:

Principal component analysis

- Principal component analysis (PCA) is a type of dimensionality reduction algorithm which is used to reduce redundancies and to compress datasets through feature extraction.
- This method uses a linear transformation to create a new data representation, yielding a set of "principal components."
- The first principal component is the direction which maximizes the variance of the dataset.
- While the second principal component also finds the maximum variance in the data, it is completely uncorrelated to the first principal component, yielding a direction that is perpendicular, or orthogonal, to the first component.
- This process repeats based on the number of dimensions, where a next principal component is the direction orthogonal to the prior components with the most variance.

•

Applications of unsupervised learning

- Machine learning techniques have become a common method to improve a product user experience and to test systems for quality assurance.
- Some of the most common real-world applications of unsupervised learning are:
- **News Sections:** Google News uses unsupervised learning to categorize articles on the same story from various online news outlets. For example, the results of a presidential election could be categorized under their label for "US" news.
- **Computer vision:** Unsupervised learning algorithms are used for visual perception tasks, such as object recognition.
- **Medical imaging:** Unsupervised machine learning provides essential features to medical imaging devices, such as image detection, classification and segmentation, used in radiology and pathology to diagnose patients quickly and accurately.
- **Anomaly detection:** Unsupervised learning models can comb through large amounts of data and discover atypical data points within a dataset. These

anomalies can raise awareness around faulty equipment, human error, or breaches in security.

- **Customer personas:** Defining customer personas makes it easier to understand common traits and business clients' purchasing habits. Unsupervised learning allows businesses to build better buyer persona profiles, enabling organizations to align their product messaging more appropriately.
- **Recommendation Engines:** Using past purchase behavior data, unsupervised learning can help to discover data trends that can be used to develop more effective cross-selling strategies. This is used to make relevant add-on recommendations to customers during the checkout process for online retailers.

Working of K-Means Algorithm

We can understand the working of K-Means clustering algorithm with the help of following steps –

- **Step 1** – First, we need to specify the number of clusters, K, need to be generated by this algorithm.
- **Step 2** – Next, randomly select K data points and assign each data point to a cluster. In simple words, classify the data based on the number of data points.
- **Step 3** – Now it will compute the cluster centroids.
- **Step 4** – Next, keep iterating the following until we find optimal centroid which is the assignment of data points to the clusters that are not changing any more

—
4.1 – First, the sum of squared distance between data points and centroids would be computed.

4.2 – Now, we have to assign each data point to the cluster that is closer than other cluster (centroid).

4.3 – At last compute the centroids for the clusters by taking the average of all data points of that cluster.

- K-means follows **Expectation-Maximization** approach to solve the problem.
- The Expectation-step is used for assigning the data points to the closest cluster and the Maximization-step is used for computing the centroid of each cluster.

While working with K-means algorithm we need to take care of the following things

—

- While working with clustering algorithms including K-Means, it is recommended to standardize the data because such algorithms use distance-based measurement to determine the similarity between data points.
- Due to the iterative nature of K-Means and random initialization of centroids, K-Means may stick in a local optimum and may not converge to global optimum. That is why it is recommended to use different initializations of centroids.

How Many Clusters?

Methods for choosing the right number of clusters

Introduction

Clustering is an unsupervised machine learning method that can identify groups of similar data points, known as clusters, from the data itself. For some clustering algorithms, such as K-means, one needs to know how many clusters there are beforehand. If the number of clusters is incorrectly specified, the results are not very informative (see Figure 1).

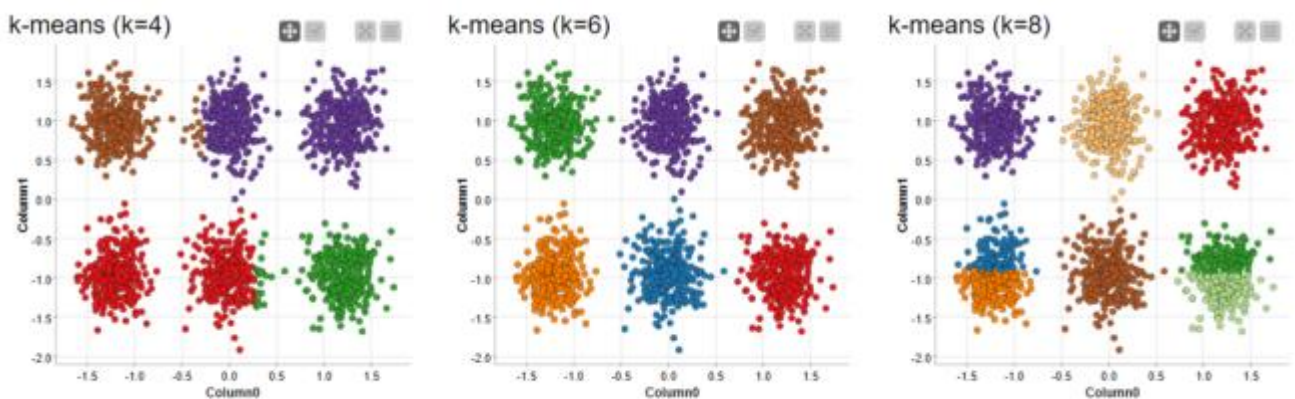


Figure 1: Clustering with different number of clusters, k=4, 6, & 8. Simulated data with 6 clusters. Image by author.

We then cover three approaches to find the optimal number of clusters:

- The elbow method
- The optimization of the silhouette coefficient
- The gap statistic

Quality of Clustering Outcome

- Before getting into different methods to determine the optimal number of clusters, we shall see how we can quantitatively assess the quality of clustering outcomes.
- Imagine the following scenarios. The same data set is clustered into three clusters (see Figure 2). As you can see, the clusters are defined well on the left, whereas the clusters are identified poorly on the right.

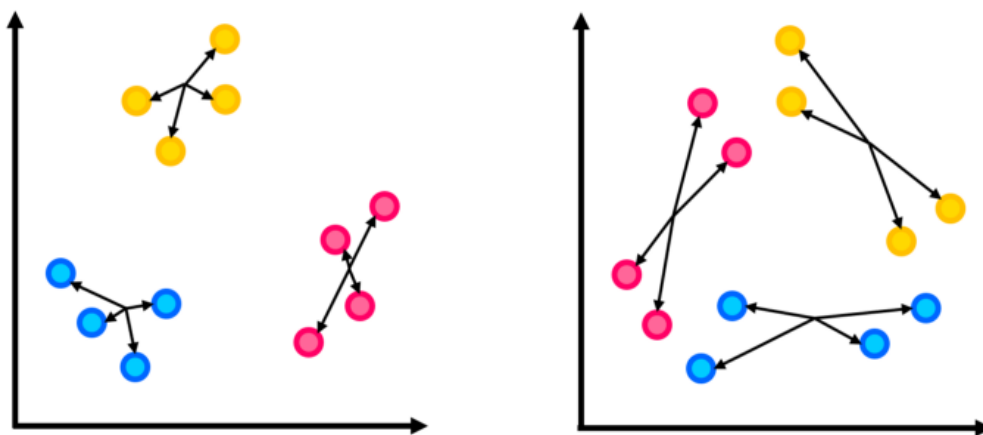


Figure 2: Examples of well-defined clusters (left) and poorly-defined clusters (right) based on the same data set. The arrows indicate the distance between the data points and their cluster centers. Image by author.

Elbow Method

- The inertia is a decreasing function of the number of clusters k .
- However, its rate of decrease is different above or below the optimal number of clusters K .
- For $k < K$, the inertia decreases rapidly, whereas the decrease is slow for $k > K$.
- Thus, by plotting the inertia over a range of k , one can determine where the curve bends, or elbows, at K .

- Figure 4 shows an inertia plot from our example in Figure 1. We can clearly see a bend, or the elbow, at $k=6$.

This method, however, is somewhat subjective, as different people may identify the elbow at different locations. In our example in Figure 4, some may argue that $k=4$ is the elbow. Moreover, the elbow may not be always apparent, as we shall see later.

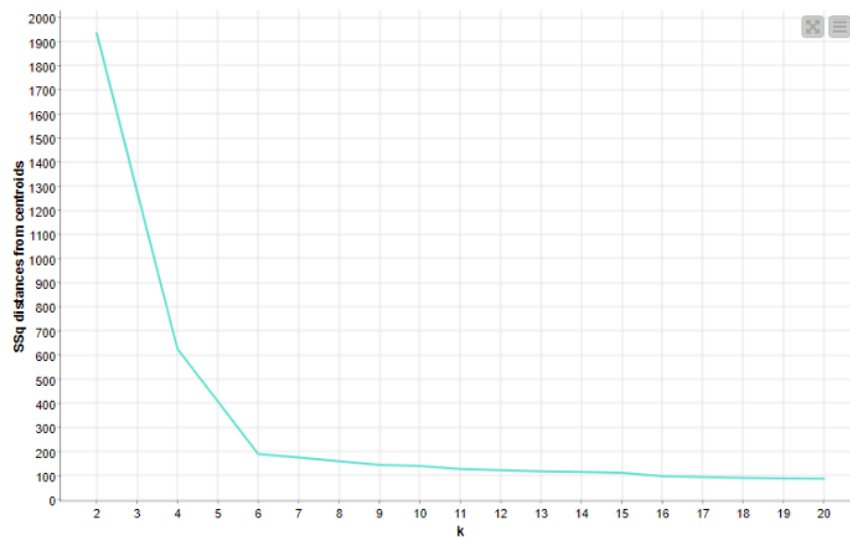


Figure 4: The plot of the inertia for different k , for the data set presented in Figure 1. Image by author.

Silhouette Method

- The silhouette coefficient may provide a more objective means to determine the optimal number of clusters.
- This is done by simply calculating the silhouette coefficient over a range of k , and identifying the peak as the optimum K .
- A KNIME component [Optimized K-Means \(Silhouette Coefficient\)](#) does exactly that. It performs K-Means clustering over a range of k , finds the optimal K that produces the largest silhouette coefficient, and assigns data points to clusters based on the optimized K .
- figure 5 shows an example of a silhouette coefficient plot from our example data presented in Figure 1.

- As it can be seen, the silhouette coefficient peaks at $k=6$, and thus it is determined as the optimum K .

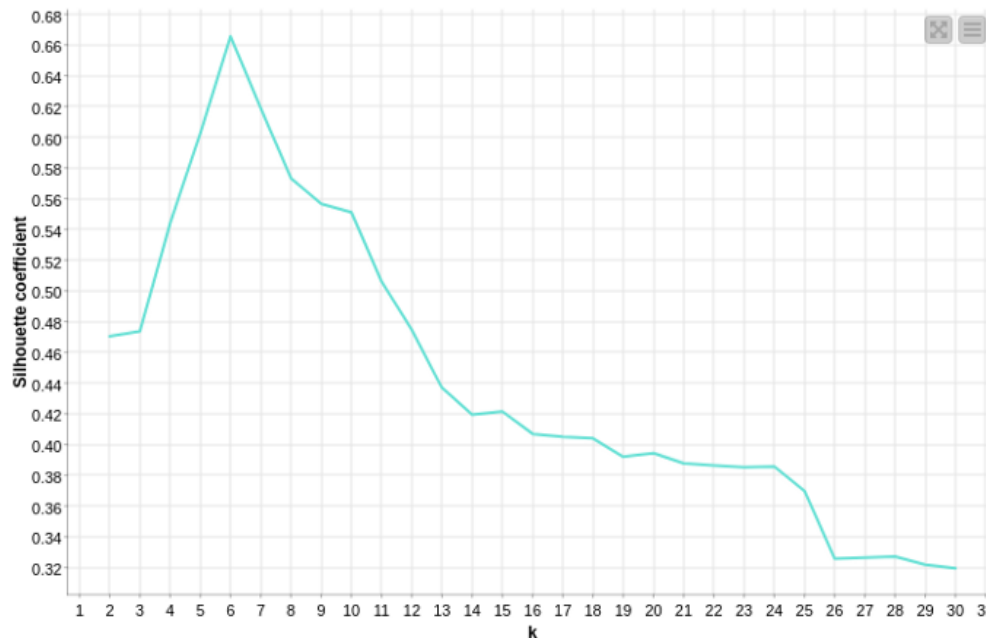


Figure 5: The plot of the silhouette coefficient for different k , for the data set presented in Figure 1.
Image by author.

Gap Statistic

- To talk about gap statistics, let's consider clustering of a random data set with no cluster organization whatsoever.
- Say a random data set is clustered into k clusters, and the inertia is calculated based on the resulting clustering (see Figure 6).
- Despite the lack of underlying cluster organization, the clustered random data produces steadily decreasing inertiae (plural of inertia) as k increases.
- This is because the more cluster centers there are, the smaller the distance becomes between data points to the cluster centers, producing decaying inertiae.
- In contrast, as we have already seen in Figure 4, the rate of decrease in inertia varies whether k is below or above the optimum number of clusters K in a data set with cluster organization.

- When the inertia for the observed and random data are plotted together, the difference becomes apparent (see Figure 7).
- The gap statistic is calculated by comparing the inertiae from a (hopefully) clustered data set and a corresponding random data set covering the same ranges in the data space (Tibshirani et al., (2001)).

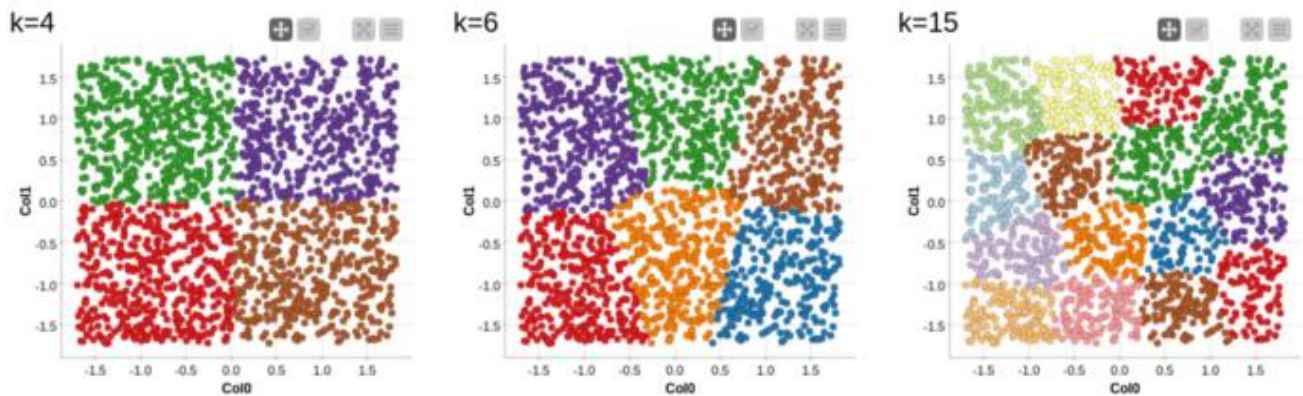


Figure 6: Uniformly distributed random data clustered into $k=4$ (left), 6 (center), and 15 (right) clusters. Image by author.

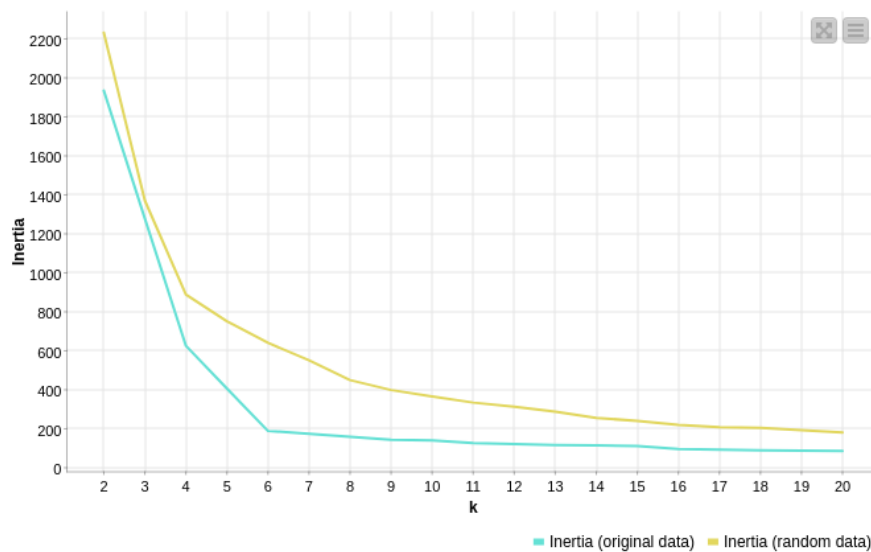


Figure 7: How the inertia decreases for the original data (from Figure 1) vs. the random data over a range of k . Image by author.

Evaluation Metrics

K-Means: Inertia

Inertia measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster.

A good model is one with low inertia AND a low number of clusters (K). However, this is a tradeoff because as K increases, inertia decreases.

To find the optimal K for a dataset, use the *Elbow method*; find the point where the decrease in inertia begins to slow. $K=3$ is the “elbow” of this graph.

- **Dunn’s Index**

- Dunn’s Index (DI) is another metric for evaluating a clustering algorithm.
- Dunn’s Index is equal to the minimum inter-cluster distance divided by the maximum cluster size.
- Note that large inter-cluster distances (better separation) and smaller cluster sizes (more compact clusters) lead to a higher DI value.
- A higher DI implies better clustering.
- It assumes that better clustering means that clusters are compact and well-separated from other clusters.

Dimensionality Reduction using PCA in python

Steps to Apply PCA in Python for Dimensionality Reduction

We will understand the step by step approach of applying Principal Component Analysis in Python with an example. In this example, we will use the iris dataset, which is already present in the sklearn library of Python.

Step-1: Import necessary libraries

All the necessary libraries required to load the dataset, pre-process it and then apply PCA on it are mentioned below:

- Python3

```
# Import necessary libraries
from sklearn import datasets # to retrieve the iris
Dataset
import pandas as pd # to load the dataframe
from sklearn.preprocessing import StandardScaler # to
standardize the features
from sklearn.decomposition import PCA # to apply PCA
import seaborn as sns # to plot the heat maps
```

Step-2: Load the dataset

After importing all the necessary libraries, we need to load the dataset. Now, the iris dataset is already present in sklearn. First, we will load it and then convert it into a pandas data frame for ease of use.

- Python3

```
#Load the Dataset
iris = datasets.load_iris()
#convert the dataset into a pandas data frame
df = pd.DataFrame(iris['data'], columns =
iris['feature_names'])
#display the head (first 5 rows) of the dataset
df.head()
```

Output:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

iris dataset

Step-3: Standardize the features

Before applying PCA or any other Machine Learning technique it is always considered good practice to standardize the data. For this, Standard Scalar is the most commonly used scalar. Standard Scalar is already present in sklearn. So, now we will standardize the feature set using Standard Scalar and store the scaled feature set as a pandas data frame.

- Python3

```
#Standardize the features
#Create an object of StandardScaler which is present in
sklearn.preprocessing
scalar = StandardScaler()
scaled_data = pd.DataFrame(scalar.fit_transform(df)) #scaling
the data
scaled_data
```

Output:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444

Scaled iris dataset

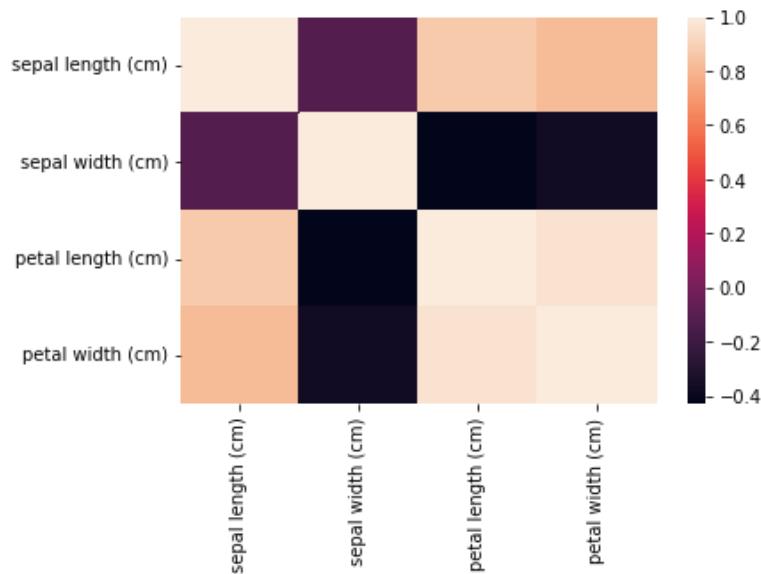
Step-3: Check the Co-relation between features without PCA (Optional)

- Now, we will check the co-relation between our scaled dataset using a heat map. For this, we have already imported the seaborn library in Step-1.
- The correlation between various features is given by the *corr()* function and then the heat map is plotted by the *heatmap()* function.
- The colour scale on the side of the heatmap helps determine the magnitude of the co-relation.
- In our example, we can clearly see that a darker shade represents less co-relation while a lighter shade represents more co-relation.
- The diagonal of the heatmap represents the co-relation of a feature with itself – which is always 1.0, thus, the diagonal of the heatmap is of the highest shade.

• Python3

```
#Check the Co-relation between features without PCA
sns.heatmap(scaled_data.corr())
```

Output:



Co-relation Heatmap of Iris dataset without PCA

We can observe from the above heatmap that sepal length & petal length and petal length & petal width have high co-relation. Thus, we evidently need to apply dimensionality reduction. If you are already aware that your dataset needs dimensionality reduction – you can skip this step.

Step-4: Applying Principal Component Analysis

We will apply PCA on the scaled dataset. For this Python offers yet another in-built class called PCA which is present in `sklearn.decomposition`, which we have already imported in step-1. We need to create an object of PCA and while doing so we also need to initialize `n_components` – which is the number of principal components we want in our final dataset. Here, we have taken `n_components = 3`, which means our final feature set will have 3 columns. We fit our scaled data to the PCA object which gives us our reduced dataset.

- Python

```
#Applying PCA
#Taking no. of Principal Components as 3
pca = PCA(n_components = 3)
```

```
pca.fit(scaled_data)
data_pca = pca.transform(scaled_data)
data_pca = pd.DataFrame(data_pca, columns=['PC1', 'PC2', 'PC3'])
data_pca.head()
```

Output:

	PC1	PC2	PC3
0	-2.264703	0.480027	-0.127706
1	-2.080961	-0.674134	-0.234609
2	-2.364229	-0.341908	0.044201
3	-2.299384	-0.597395	0.091290
4	-2.389842	0.646835	0.015738

PCA Dataset

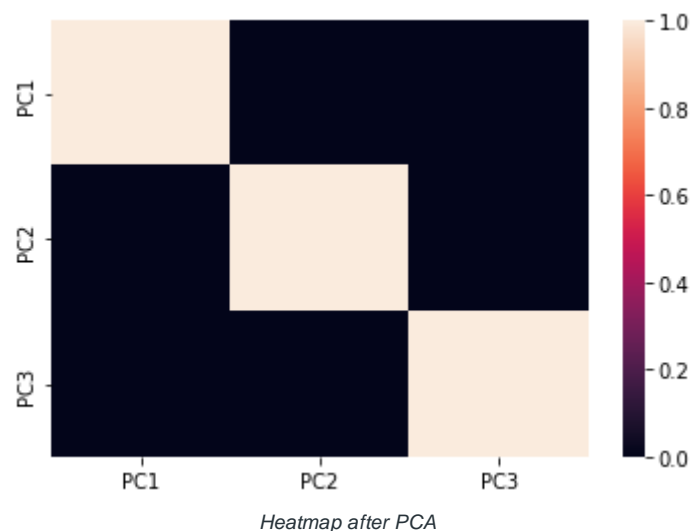
Step-5: Checking Co-relation between features after PCA

Now that we have applied PCA and obtained the reduced feature set, we will check the co-relation between various Principal Components, again by using a heatmap.

- Python3

```
#Checking Co-relation between features after PCA
sns.heatmap(data_pca.corr())
```

Output:



The above heatmap clearly depicts that there is no correlation between various obtained principal components (PC1, PC2, and PC3). Thus, we have moved from

higher dimensional feature space to a lower-dimensional feature space while ensuring that there is no correlation between the so obtained PCs is minimum. Hence, we have accomplished the objectives of PCA.

What is MLOps?

Machine Learning Operations involves a set of processes or rather a sequence of steps implemented to deploy an ML model to the production environment. There are several steps to be undertaken before an ML Model is production-ready. These processes ensure that your model can be scaled for a large user base and perform accurately. 📖

Why do we need MLOps?

Creating an ML model that can predict what you want it to predict from the data you have fed is easy. *However, creating an ML model that is reliable, fast, accurate, and can be used by a large number of users is difficult.*

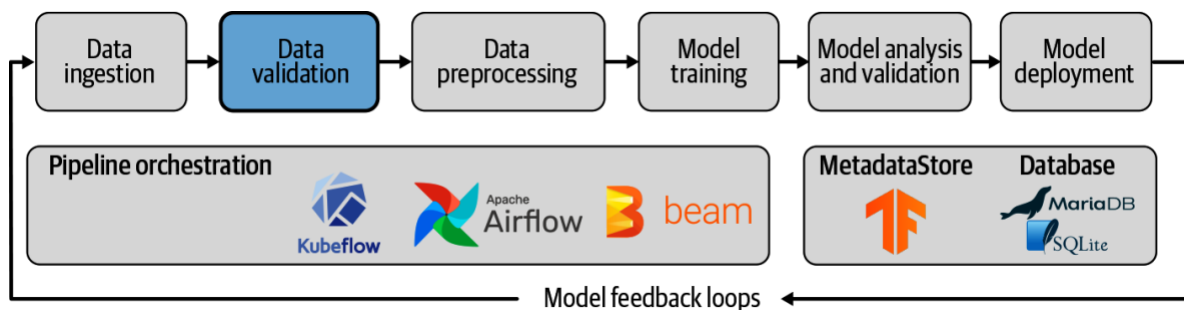
The necessity of MLOps can be summarized as follows:

- ML models ***rely on a huge amount of data***, difficult for a single person to keep track of.
- ***Difficult to keep track of parameters*** we tweak in ML models. Small changes can lead to enormous differences in the results.
- We have to keep track of the features the model works with, feature engineering is a separate task that contributes largely to model accuracy.
- Monitoring an ML model isn't like monitoring a deployed software or web app.
- ***Debugging an ML model is an extremely complicated art***

- Models rely on real-world data for predicting, ***as real-world data changes, so should the model.*** This means we have to keep track of new data changes and make sure the model learns accordingly.


Pipeline

- A machine learning pipeline is a way to control and automate the workflow it takes to produce a machine learning model.
- Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment.
- Machine learning pipelines are iterative as every step is repeated to continuously improve the accuracy of the model and achieve the end goal.



An example of ML Pipeline O'Reilly

- The term **Pipeline** is used generally to describe the independent sequence of steps that are arranged together to achieve a task.
- This task could be machine learning or not. Machine Learning Pipelines are very common but that is not the only type of pipeline that exists.
- Data Orchestration Pipelines are another example. According to Microsoft docs, there are three scenarios:

Scenario	Primary persona	Azure offering	OSS offering	Canonical pipe	Strengths
Model orchestration (Machine learning)	Data scientist	Azure Machine Learning Pipelines	Kubeflow Pipelines	Data -> Model	Distribution, caching, code-first, reuse
Data orchestration (Data prep)	Data engineer	Azure Data Factory pipelines	Apache Airflow	Data -> Data	Strongly typed movement, data-centric activities
Code & app orchestration (CI/CD)	App Developer / Ops	Azure Pipelines 	Jenkins	Code + Model -> App/Service	Most open and flexible activity support, approval queues, phases with gating

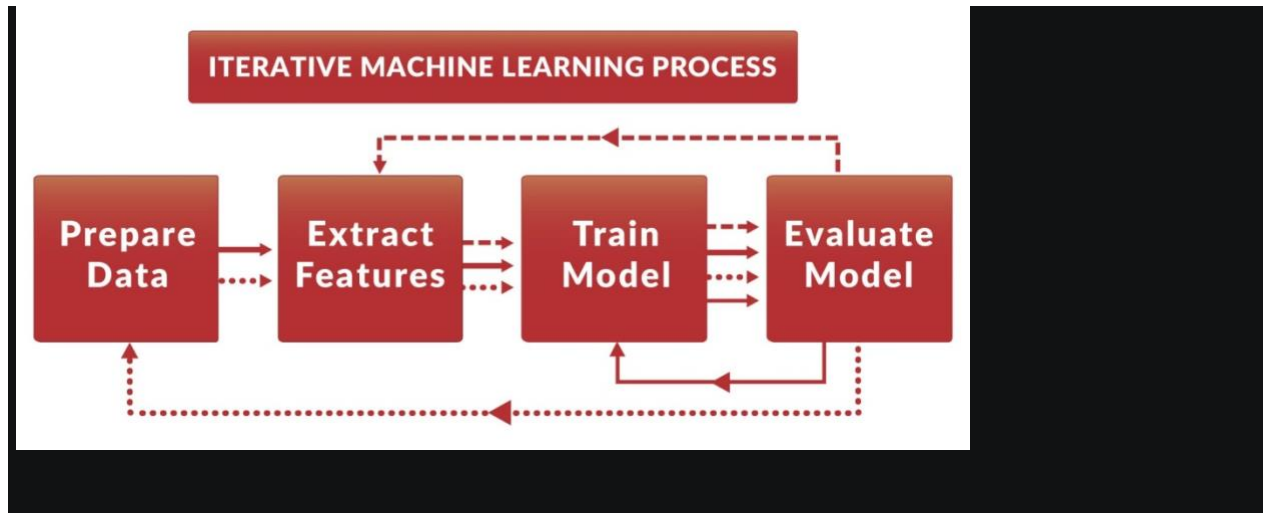
Version Control

- Version control is important in any software development environment, and even more so in machine learning.
In ML, the development process is very complex. It includes huge amounts of data, testing of multiple models, optimization of parameters, tuning of features, and more.
- Version Control is the process of tracking and managing software changes over time. Whether you're building an app or an ML model, you need to track every modification done by members of the software team to fix bugs and avoid conflicts.
- To achieve that, you can use a version control framework. Frameworks were developed to track every individual change by each contributor and save it in a special kind of database, where you can identify the differences and help prevent conflicts in concurrent work while merging.

Why do we need version control in ML?

- The machine learning development process includes a lot of iterative work, where the developers are searching for the best performing model while

changing hyperparameters, code, and data. It's important to keep a history of these changes to track model performance relative to the parameters, which saves you the time you'd spend retraining the model for experimentation.



- Machine learning version control has three parts:

Code

- There's modelling code, and there's implementation code. Modelling code is used to implement the model, and implementation code is used for inference. They can both be written in different programming languages, but it might make it more difficult to maintain all of your code and dependencies.

Data

- There's metadata, which is information about your data and model. Then there's the actual data, the datasets you use to train and run your model. Metadata can change without any change in the data, and versioning should link the data to the appropriate meta.

Model

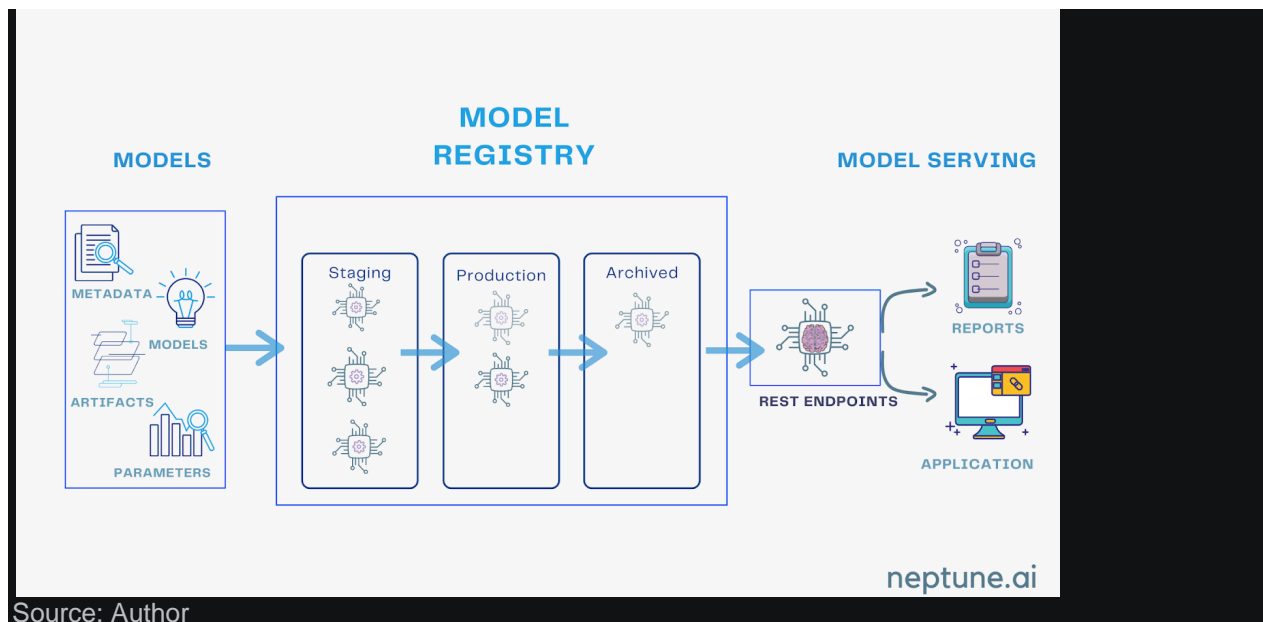
- The model connects all of the above with model parameters and hyperparameters.

Model Registry Makes MLOps Work: Here's Why

- Model Registry is a part of the machine learning lifecycle or MLOps.
- It is a service that manages multiple model artifacts, tracks, and governs models at different stages of the ML lifecycle.
- Model registry is a collaborative hub where teams can work together at different stages of the machine learning lifecycle, starting from the experimentation phase to the production phase.
- It makes approval, governance, and monitoring seamless and improves workflow performance.
- It helps you manage the full development life cycle of an ML Model, and standardize deployment.
- In the next few years, 75% of companies/startups are planning to go from pilot to operations for their machine learning projects.
- So, when it comes to production we sometimes use ML Tools that are completely unfit for data scientists or ML workflows.
- So, the **Model Registry is a kind of a linchpin** to make MLOps work. We'll explore a few platforms and the key features of Model registry.

What is a Model Registry?

- The Model Registry is a system that allows machine learning engineers and data scientists to publish, test, monitor, govern and share them for collaboration with other teams.
- Essentially, the model registry is used when you're done with your experimentation phase, and ready to share with the team and stakeholders.



Source: Author

Why we need Model Registry

- Let's say you've spent a lot of resources developing an ML algorithm that works well and has good potential to impact your business outcome.
- Rolling out ML to production is painfully slow, companies take one, sometimes two years to release a single model.
- What we lack is transparency and a way to collaborate with other team members.
- So, what if you had a Central Repo for staging all your production-ready models? That would streamline the entire production and workflow process.
- With a model registry, you can ensure that all the key values (including data, configurations, environment variables, model code, versions, and docs) are in one place, where everyone has access.
- Lack of governance and security is a major issue in many industries. It eventually slows down production, and then companies have to go back to the whiteboard to think about what went wrong and how to fix it. There are many real-life cases where lack of proper management leads to serious issues. Proper management, governance, and testing might not create any issue in the first place. Thanks to the model registry, you can:
 - Manage model lifecycle
 - Model risks and approval of workflows
 - Roll-out faster and seamlessly

- Easily collaborate and manage

What Is Model Monitoring?

- ML model monitoring is the practice of tracking the performance of ML models in production to identify potential issues that can add negative business value. These practices help proactively monitor prediction quality issues, data relevance, model accuracy, and bias.
- ML monitoring constitutes the subset of AI observability where it showcases a bigger picture with testing, validation, explainability, and exploring unforeseen failure modes.
- The performance of ML models starts degrading over time. It can be due to data inconsistencies, skews, and drifts, making deployed models inaccurate and irrelevant. Appropriate ML monitoring helps identify precisely when the model performance started diminishing. Such proactive monitoring helps take required actions like retraining models or replacing models. It helps foster users' trust in ML systems.

Why is ML Model Monitoring Important?

- ML monitoring helps fix the issue of poor generalization that arises due to training models on smaller subsets of data. With poor generalization, ML models fail to attain the required accuracy. Changes in data distributions over time also affect model performance.
- A model is trained and optimized based on the variables and parameters provided to it. These same parameters may fail to hold ground truth or become insignificant when the model is deployed. Monitoring practices help analyze how the deployed model performs on real-time data over a long period.
- ML models involve complex pipelines and automated workflows. Without an appropriate monitoring framework, these transformations become error-prone and hamper the model's performance over time.
- Inputs fed to ML models can destabilize ML systems due to changes in sampling methods or hyper-parameter configuration changes. Tracking different stability metrics helps stabilize the ML system.
- ML model monitoring helps understand and debug production models and the derived insights help reduce complications raised due to the black-box nature of ML models.

MLOPS deployment

Successful ML deployments generally take advantage of a few key MLOps principles, which are built on the following pillars:

- **Tracking** – ML models are software artifacts that need to be deployed. Tracking provenance is critical for deploying any good software and typically handled through version control systems. But, building ML models depends on complex details such as data, model architectures, hyperparameters, and external software. Keeping track of these details is vital, but can be simplified greatly with the right tools, patterns, and practices.
- **Automation & DevOps** – Automation is key to modern DevOps, but it's more difficult for ML models. In a traditional software application, a continuous integration and continuous delivery (CI/CD) pipeline would pick up some versioned source code for deployment. For an ML application, the pipeline should not only automate training models, but also automate model retraining along with archival of training data and artifacts.
- **Monitoring/Observability** – Monitoring software requires good logging and alerting, but there are special considerations to be made for ML applications. All predictions generated by ML models should be logged in such a way that enables traceability back to the model training job. ML applications should also be monitored for invalid predictions or data drift, which may require models to be retrained.
- **Reliability** – ML models (especially deep learning models) can be harder to test and more computationally expensive than traditional software. It is important to make sure your ML applications function as expected and are resilient to failures. Getting reliability right for ML requires some special considerations around security and testing.

A Comparative Study of Clustering Algorithms

Clustering is basically defined as division of data into groups of similar objects. Each group called a cluster consists of objects that are similar between themselves and dissimilar compared of other groups. Lets compare among different type of clusters. The algorithms under discuss are: ***k-means algorithm, hierarchical***

clustering algorithm, self organizing maps algorithm, and expectation maximization clustering algorithm.

Comparison Metrics:

Now I would like to decide the factors on which I would discuss the comparison amongst the clustering algorithms:

1. size of dataset
2. number of clusters
3. type of dataset and type of software used
4. performance of the algorithm
5. accuracy of the algorithm
6. quality of the algorithm

	Size of Dataset	Number of Clusters	Type of Dataset	Type of Software
<i>k</i> -means Alg.	Huge Dataset & Small Dataset	Large number of clusters & Small number of clusters	Ideal Dataset & Random Dataset	LNKnet Package & Cluster and TreeView Package
HC Alg.	Huge Dataset & Small Dataset	Large number of clusters & Small number of clusters	Ideal Dataset & Random Dataset	LNKnet Package & Cluster and TreeView Package
SOM Alg.	Huge Dataset & Small Dataset	Large number of clusters & Small number of clusters	Ideal Dataset & Random Dataset	LNKnet Package & Cluster and TreeView Package
EM Alg.	Huge Dataset & Small Dataset	Large number of clusters & Small number of clusters	Ideal Dataset & Random Dataset	LNKnet Package & Cluster and TreeView Package

