Module 1

Introduction to Unsupervised Learning: Overview:
Understanding Unsupervised Learning

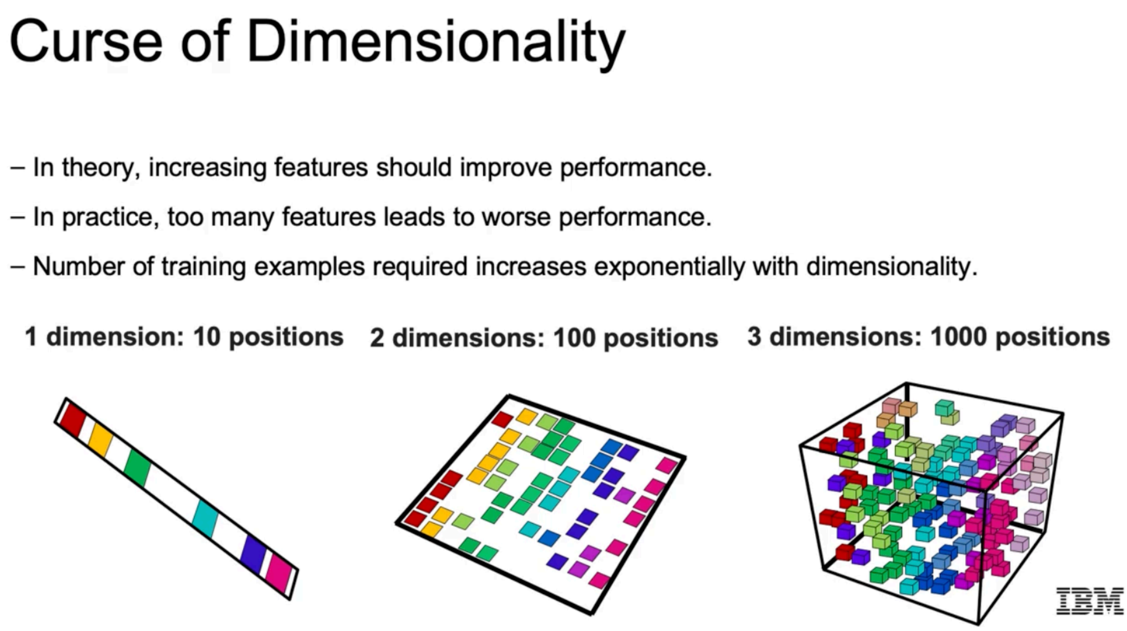
- Unsupervised learning focuses on finding structures within datasets without known outcomes, unlike supervised learning.
- Key use cases include clustering, which groups data points (e.g., customer segmentation), and dimensionality reduction, which simplifies datasets while retaining essential information.

Clustering Algorithms

- The course will cover several clustering algorithms, including:
  - K-means algorithm
  - Hierarchical agglomerative clustering
  - DBSCAN algorithm
  - Mean shift algorithm

Dimensionality Reduction Techniques

- Dimensionality reduction is crucial to mitigate the "curse of dimensionality," where too many features can lead to poor model performance.
- Techniques discussed include:
  - Principal Component Analysis (PCA)
  - Non-negative matrix factorization
- The course emphasizes the importance of reducing dimensions to improve model efficiency and interpretability.



Common Use Cases for Clustering

- **Classification**: Clustering can identify groupings in unlabeled data, such as distinguishing spam emails from regular ones or categorizing product reviews.
- **Anomaly Detection**: It helps detect unusual patterns, like identifying potentially fraudulent credit card transactions by clustering abnormal transaction behaviors.

Customer Segmentation

- Clustering can segment customers based on behaviors like purchase frequency and demographics, aiding in targeted marketing strategies.

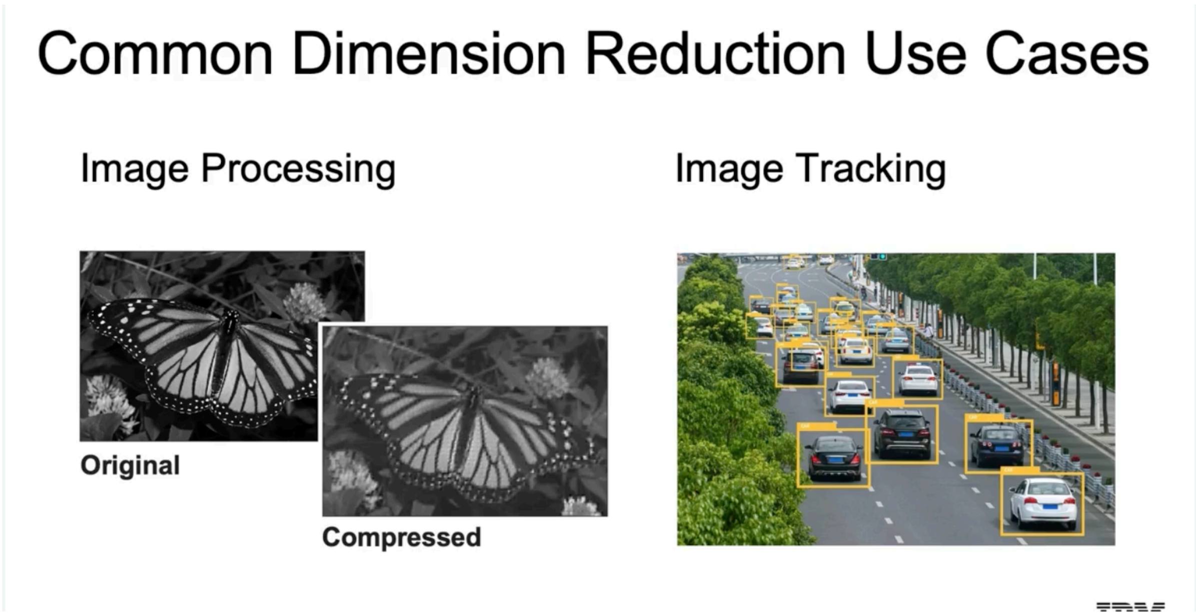
Improving Supervised Learning

- Clustering can enhance supervised learning models by training separate models for different data segments, potentially improving classification performance.

Dimension Reduction

- This technique is used to compress high-resolution images while retaining essential information, which is crucial for image processing and computational efficiency.

The lecture concludes by preparing learners for the next topic on the K-means algorithm, a specific clustering method.



Introduction to Clustering

Clustering Basics

- Clustering is used to group users into segments based on features, such as the number of visits.

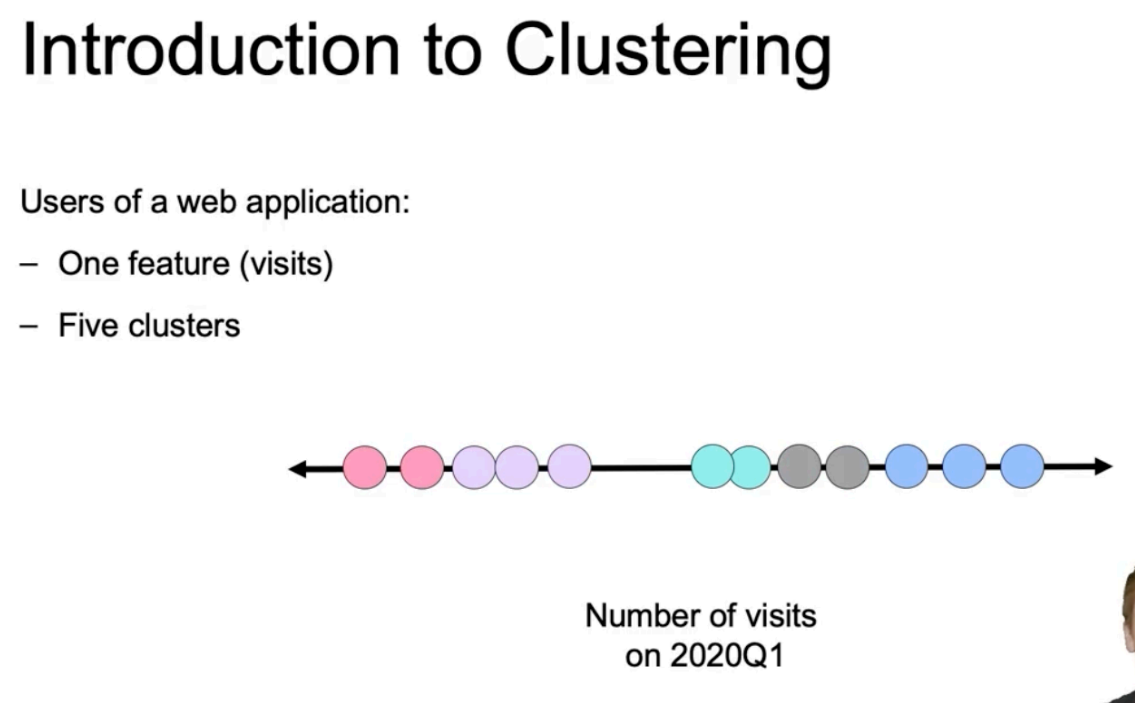
- Visual representation helps in determining the best way to draw lines between clusters.

Choosing the Number of Clusters

- Depending on business objectives, you may need to create two, three, or even five clusters.
- The course will cover various clustering algorithms and how to select the appropriate number of clusters for your data.

Introduction to K-means

- The next video will introduce K-means, the first unsupervised learning model to be explored in the course.
- Understanding K-means will provide foundational knowledge for applying clustering techniques effectively.



K-Means Algorithm Overview

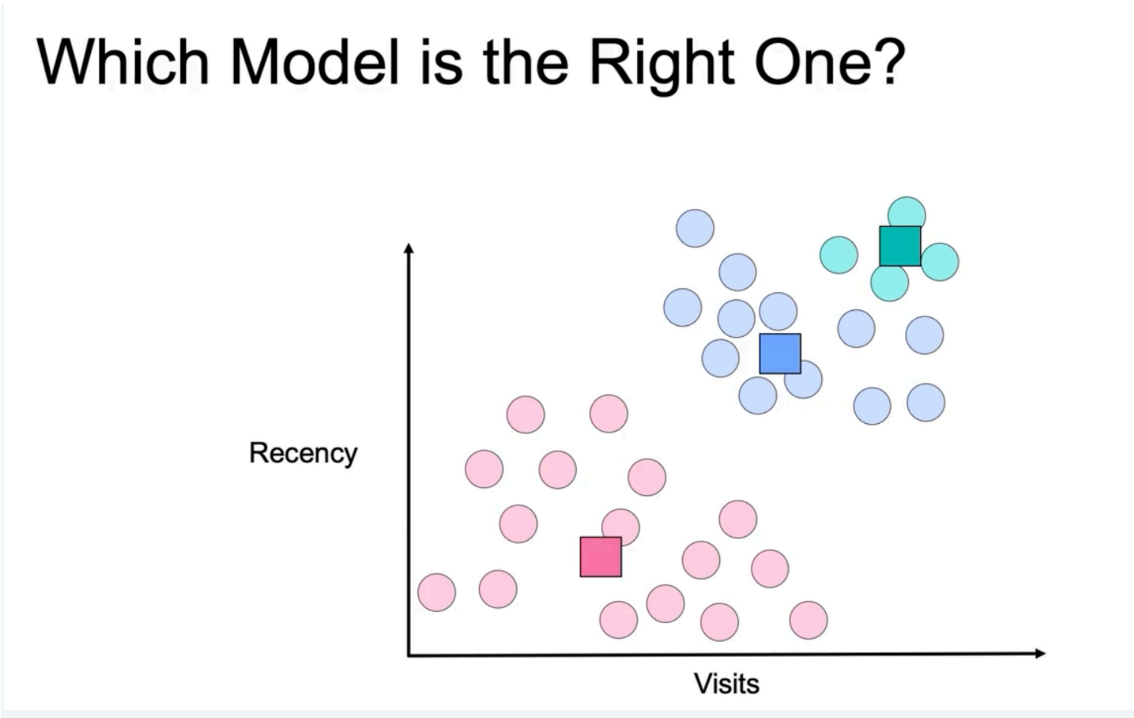
- The algorithm begins by selecting two random points as centroids for the clusters.
- Each data point is assigned to the nearest centroid, forming initial clusters.

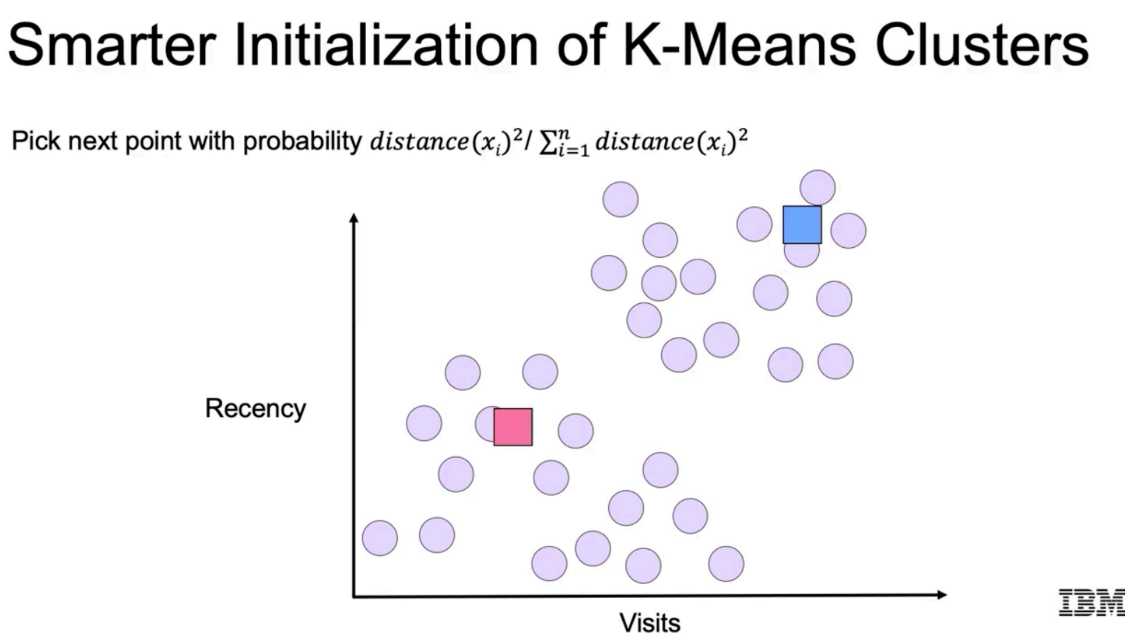
Iteration Process

- After the initial assignment, centroids are recalculated as the mean of the points in each cluster.
- This process of assigning points and updating centroids is repeated until the centroids no longer change, indicating convergence.

Challenges and Considerations

- The K-Means algorithm can yield different results based on the initial choice of centroids.
- Multiple convergence points may exist, leading to different cluster configurations, which can affect the algorithm's effectiveness.





The Elbow Method

- The elbow method identifies an inflection point in a graph of the number of clusters versus inertia/distortion, indicating a suitable number of clusters (K).
- Before the inflection point, inertia or distortion decreases rapidly; after it, the rate of decrease slows significantly.

Implementing K-Means in Python

- K-Means can be implemented by importing the class from sklearn.cluster and initializing it with hyperparameters, including the number of clusters.
- The fitting process involves calling `.fit()` on the data and using `.predict()` to determine cluster assignments.

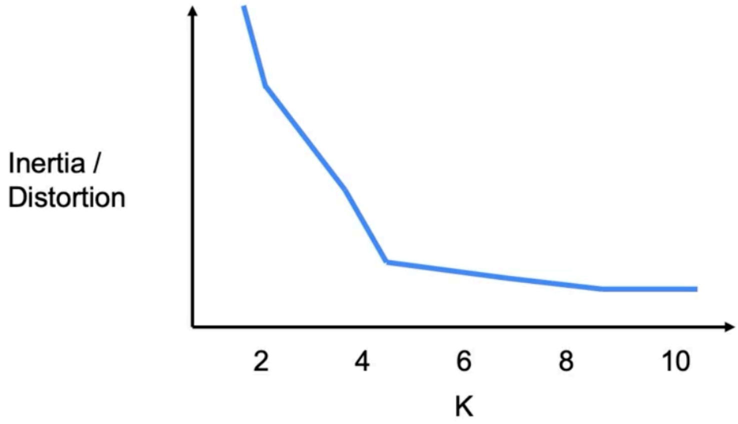
Using the Elbow Method in Practice

- To apply the elbow method, fit K-Means for various cluster counts (1-10) and save the inertia values.
- Plotting these values helps visualize the elbow point, guiding the selection of the optimal number of clusters.

Choosing the Right Number of Clusters

Inertia measures distance of point to cluster.

Value decreases with increasing K as long as cluster density increases.



K-Means: The Syntax

Import the class containing the clustering method.

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

Create an instance of the class.

```
kmeans = KMeans(n_clusters=3,
                init='k-means++')
```

Fit the instance on the data and then predict clusters for new data.

```
kmeans = kmeans.fit(X1)
y_predict = kmeans.predict(X2)
```

Can also be used in batch mode with **MiniBatchKMeans**.

K-Means: Elbow Method Syntax

To implement elbow method, fit K-Means for various levels of *k*, save inertia values.

```
inertia = [ ]
list_clusters = list(range(10))
for k in list_clusters:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    inertia.append(km.inertia_)
```