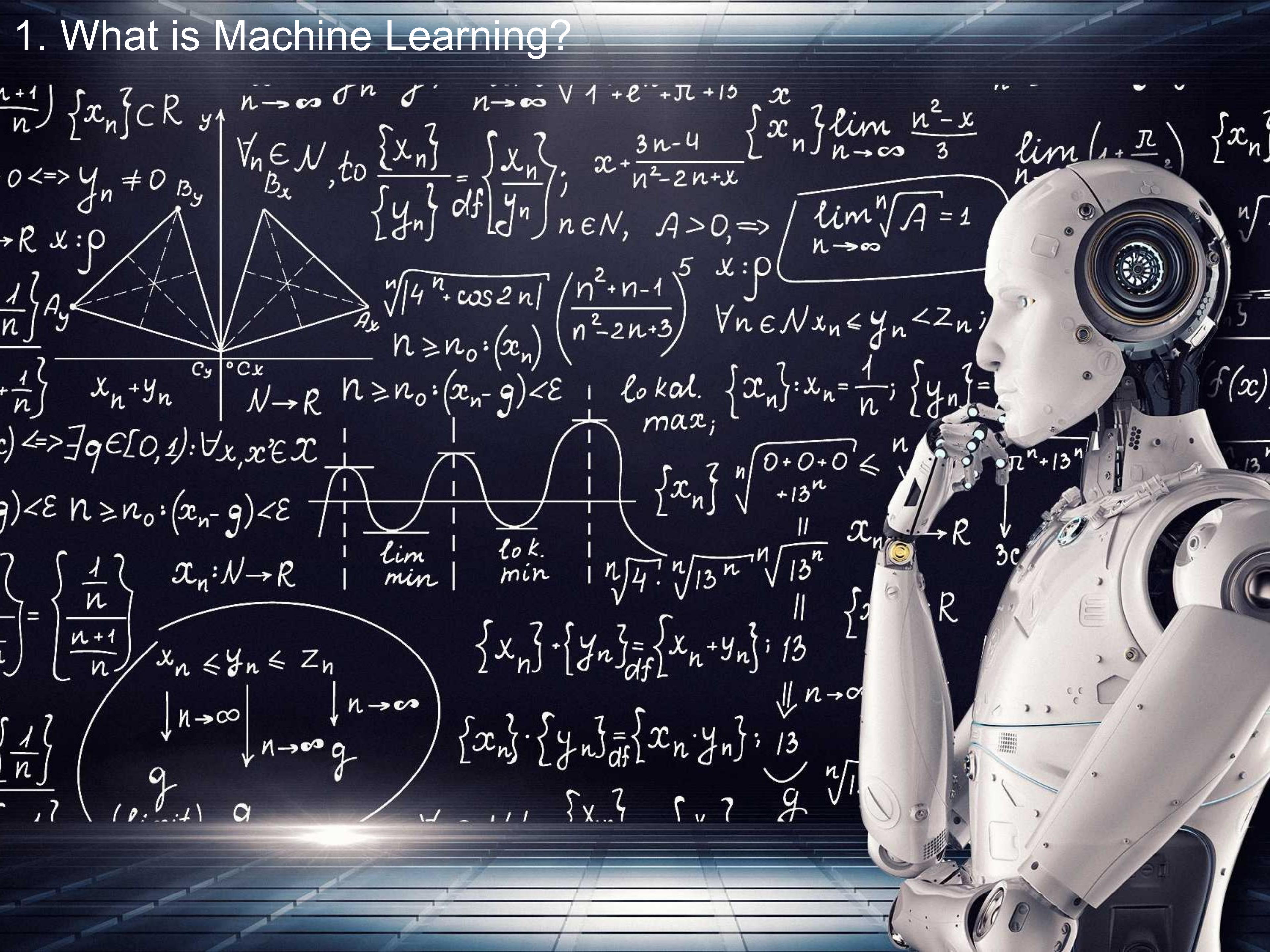
Lisbon School of Economics & Management
Universidade de Lisboa

LISBOA
UNIVERSIDADE DE LISBOA

Ano Letivo 2025/2025

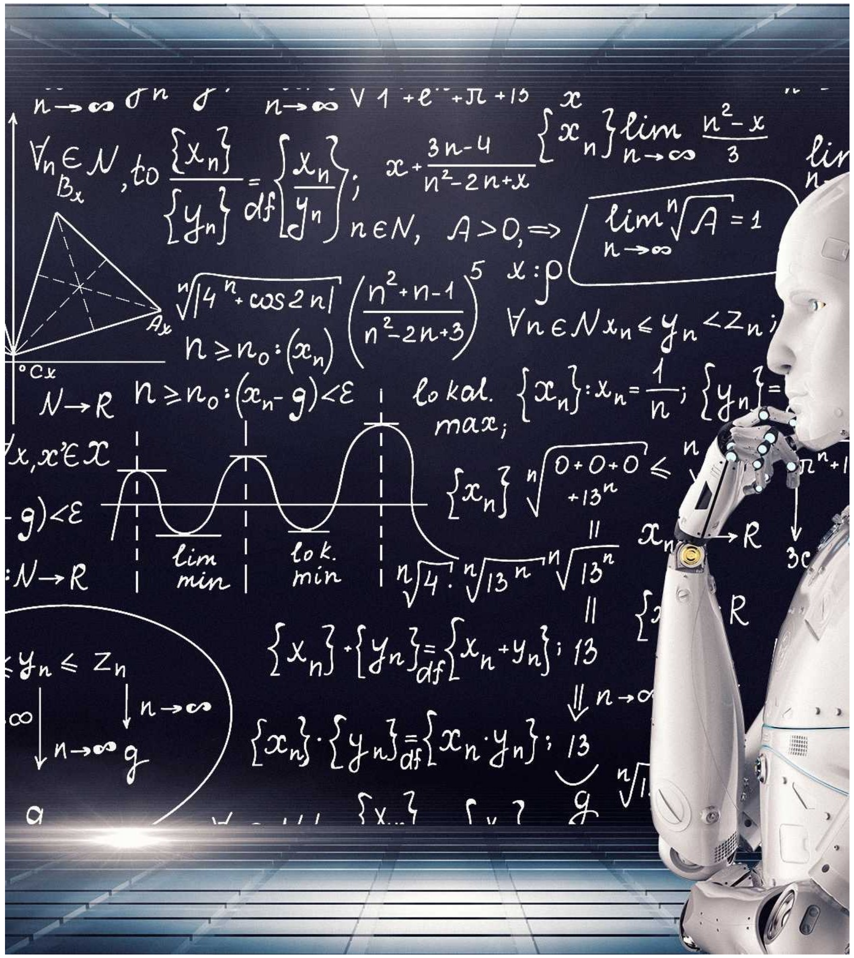
MACHINE LEARNING

1. What is Machine Learning?



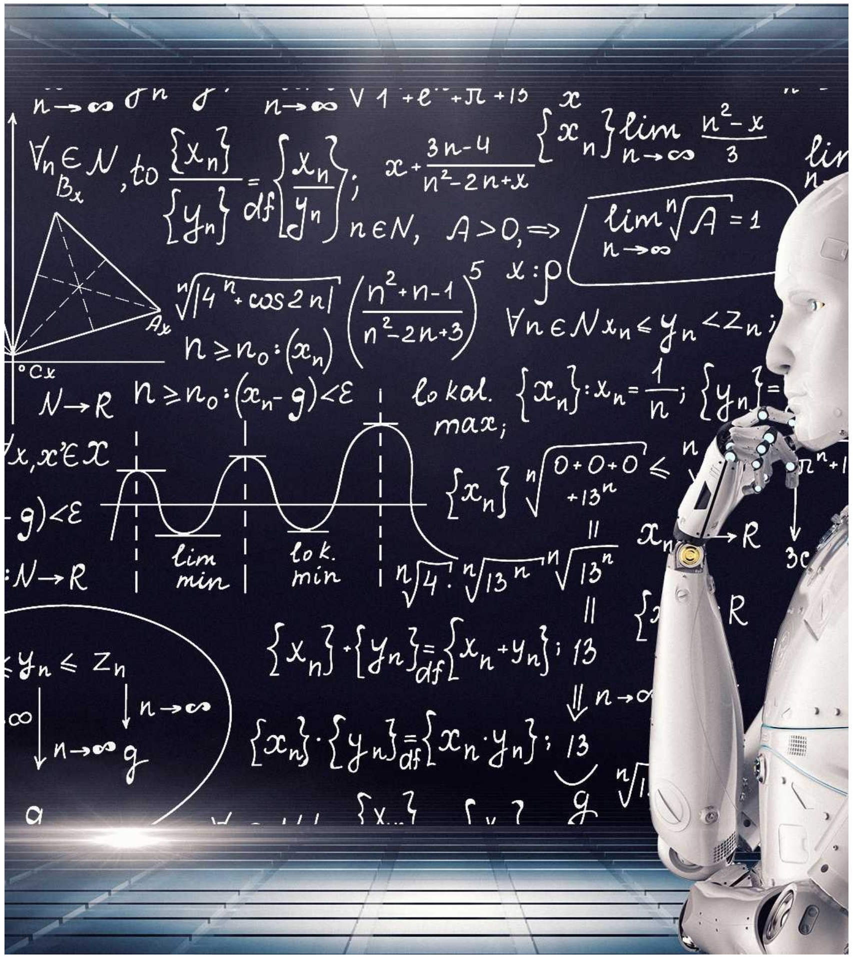
1. What is Machine Learning?

- Branch of Artificial Intelligence
- focuses on creating systems that can **learn from data** rather than relying on explicit programming.
- Instead of writing step-by-step instructions we provide data and algorithms that **detect patterns**, make predictions, or group observations.



1. What is Machine Learning?

- **Key idea:** Algorithms approximate relationships in data.
- **Common applications:**
  - forecasting demand,
  - fraud detection,
  - customer segmentation,
  - recommendation systems,
  - sentiment analysis.



2. Categories of Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

2.1 Supervised Learning

- **Definition:** Learning from labeled data, i.e., input-output pairs (X, y).
- **Goal:** Train a model that maps inputs X to outputs y.

2.1 Supervised Learning

- **Examples:**
  - Predicting house prices (regression)
  - Classifying emails as spam/not spam (classification)
- **Main algorithms:** Linear Regression, Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, Neural Networks.

2.2 Unsupervised Learning

- **Definition:** Learning from unlabeled data, i.e., only inputs X.
- **Goal:** Discover hidden structures or patterns.

2.2 Unsupervised Learning

- **Examples:**
  - Customer segmentation (clustering)
  - Topic modeling in documents
  - Dimensionality reduction (e.g., PCA)
- **Main algorithms:** K-Means, Hierarchical Clustering, PCA, t-SNE, Autoencoders.

Example of Regression

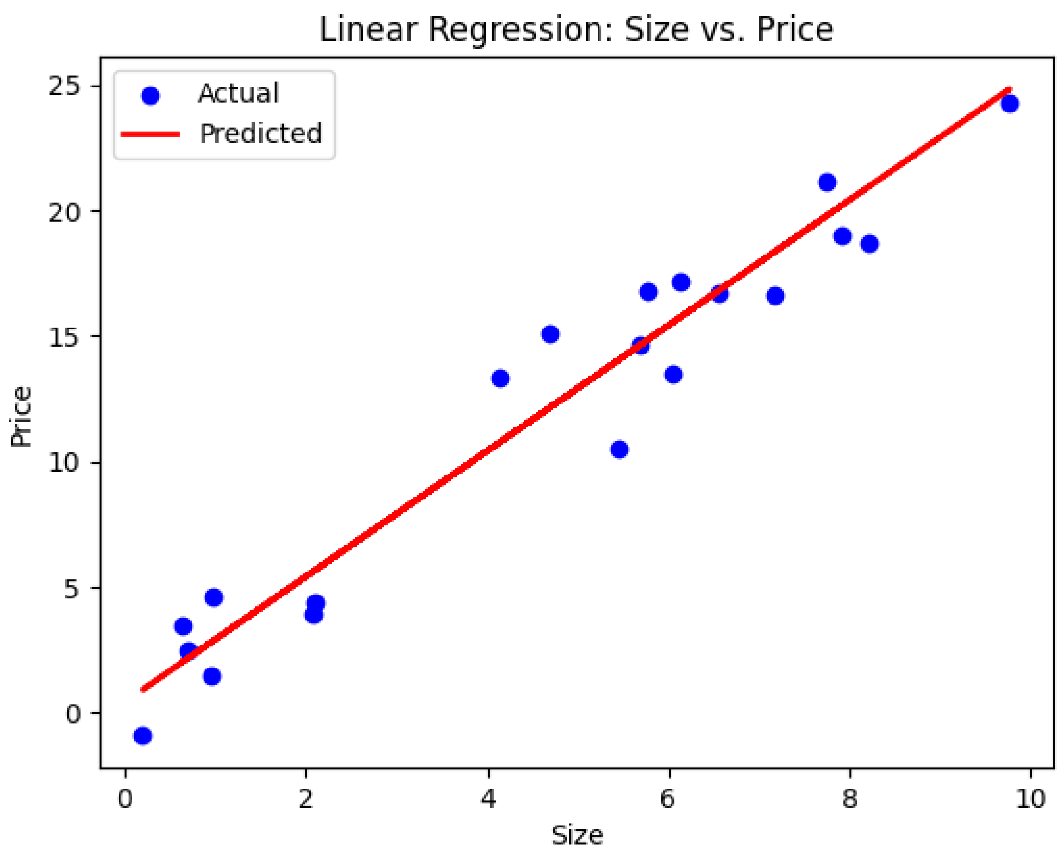
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score

np.random.seed(0)
X = np.random.rand(100, 1) * 10   # size in 0-10
y = 2.5 * X.squeeze() + np.random.randn(100) * 2  # price with noise

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Regression coefficients:", model.coef_, model.intercept_)
print("MSE:", mean_squared_error(y_test, y_pred))
```

Example of Regression



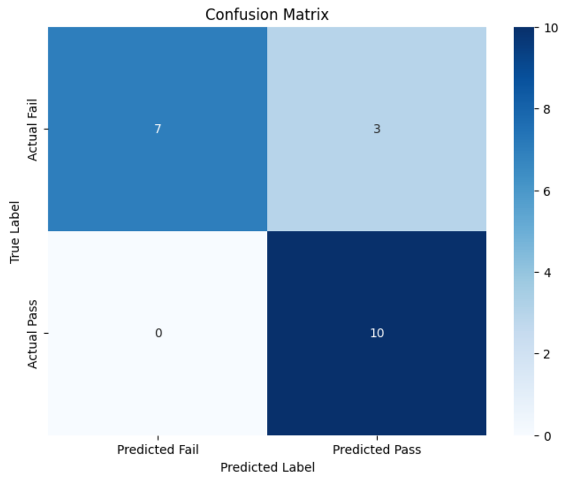
Example of Classification

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Example 2: Classification - Predicting pass/fail based on study hours
hours = np.random.rand(100, 1) * 10
labels = (hours.squeeze() + np.random.randn(100)) > 5  # Pass if > 5h effective study
X_train, X_test, y_train, y_test = train_test_split(hours, labels, test_size=0.2)

clf = LogisticRegression().fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```



Accuracy: 0.95

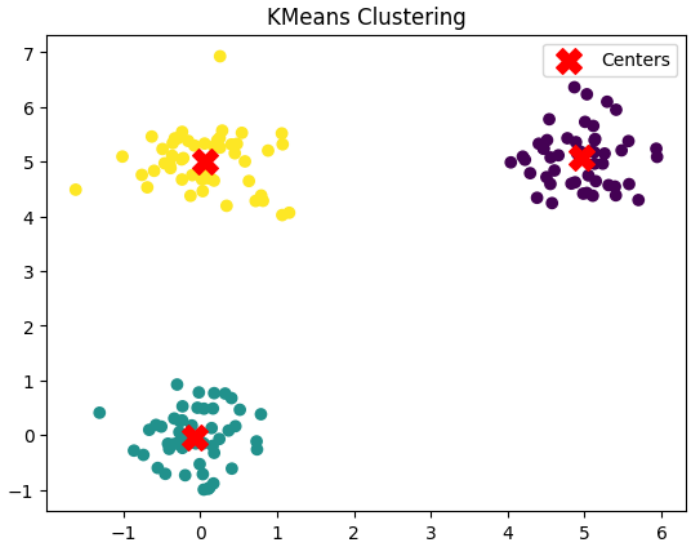
Example of Clustering

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Generate synthetic data: 3 clusters
np.random.seed(42)
X = np.vstack([
    np.random.normal([0,0], 0.5, (50,2)),
    np.random.normal([5,5], 0.5, (50,2)),
    np.random.normal([0,5], 0.5, (50,2))
])

# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=0).fit(X)
labels = kmeans.labels_

# Visualization
plt.scatter(X[:,0], X[:,1], c=labels, cmap="viridis")
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],
            c="red", marker="X", s=200, label="Centers")
plt.title("KMeans Clustering")
plt.legend()
plt.show()
```



Sentiment Analysis

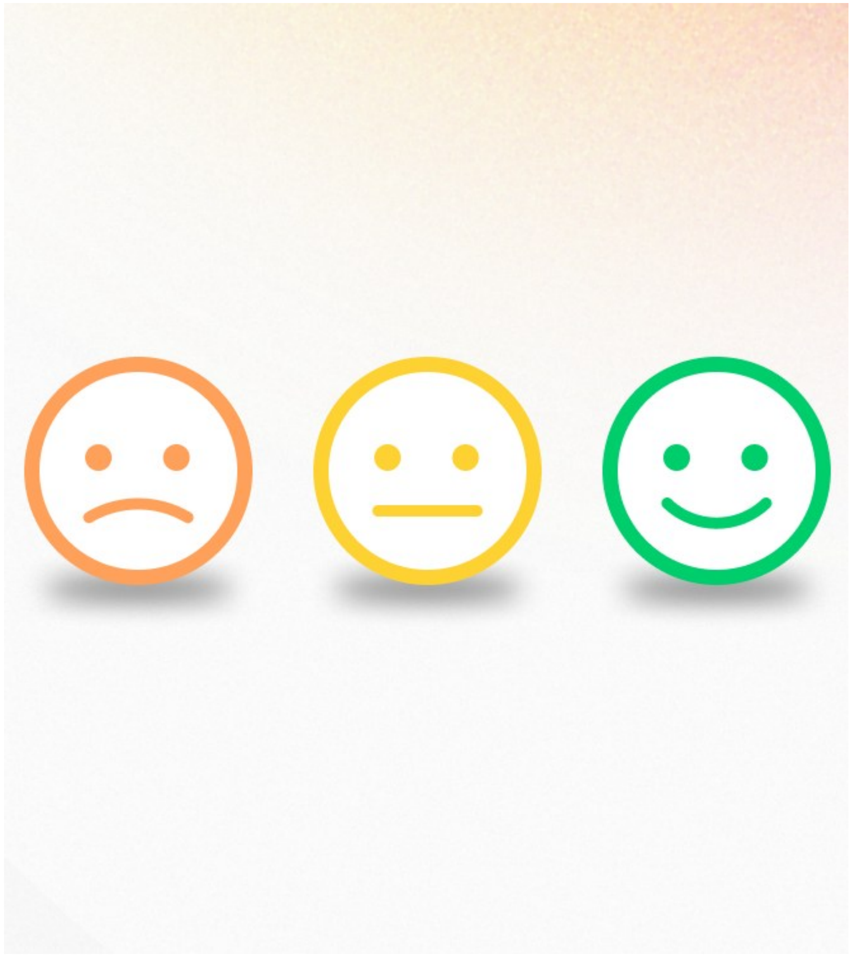
```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

# Download VADER model (only needed once)
nltk.download('vader_lexicon')

# Initialize sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Example texts
texts = [
    "I love this product, it’s fantastic!",
    "This is the worst experience I’ve ever had.",
    "The movie was okay, not great but not terrible either.",
    "Amazing customer support, very satisfied.",
    "Totally disappointed, waste of money."
]

# Analyze each text
for t in texts:
    score = sia.polarity_scores(t)
    sentiment = "Positive" if score['compound'] > 0.05 else ("Negative"
if score['compound'] < -0.05 else "Neutral")
    print(f"Text: {t}\n Score: {score} → Sentiment: {sentiment}\n")
```



Sentiment Analysis

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# Example labeled dataset
reviews = [
    "I love this phone", "This product is terrible",
    "Fantastic quality, very happy", "Worst service ever",
    "Not bad, but could be better", "Absolutely wonderful"
]
labels = [1, 0, 1, 0, 0, 1]  # 1 = Positive, 0 = Negative

# Convert text into numerical features (Bag of Words)
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(reviews)

# Split train/test
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.3, random_state=42)

# Train Naive Bayes classifier
clf = MultinomialNB().fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Predictions:", y_pred)
```

Sentiment Analysis

- How to use?

```
new_text = ["It is so good"]
X_new = vectorizer.transform(new_text)
prediction = clf.predict(X_new)

print("Prediction:", prediction)
```

4. Summary

- **Supervised Learning**: learns from labeled data (prediction, classification).
- **Unsupervised Learning**: finds hidden structures in unlabeled data (clustering, dimensionality reduction).
- Python provides powerful libraries like scikit-learn for easy implementation.