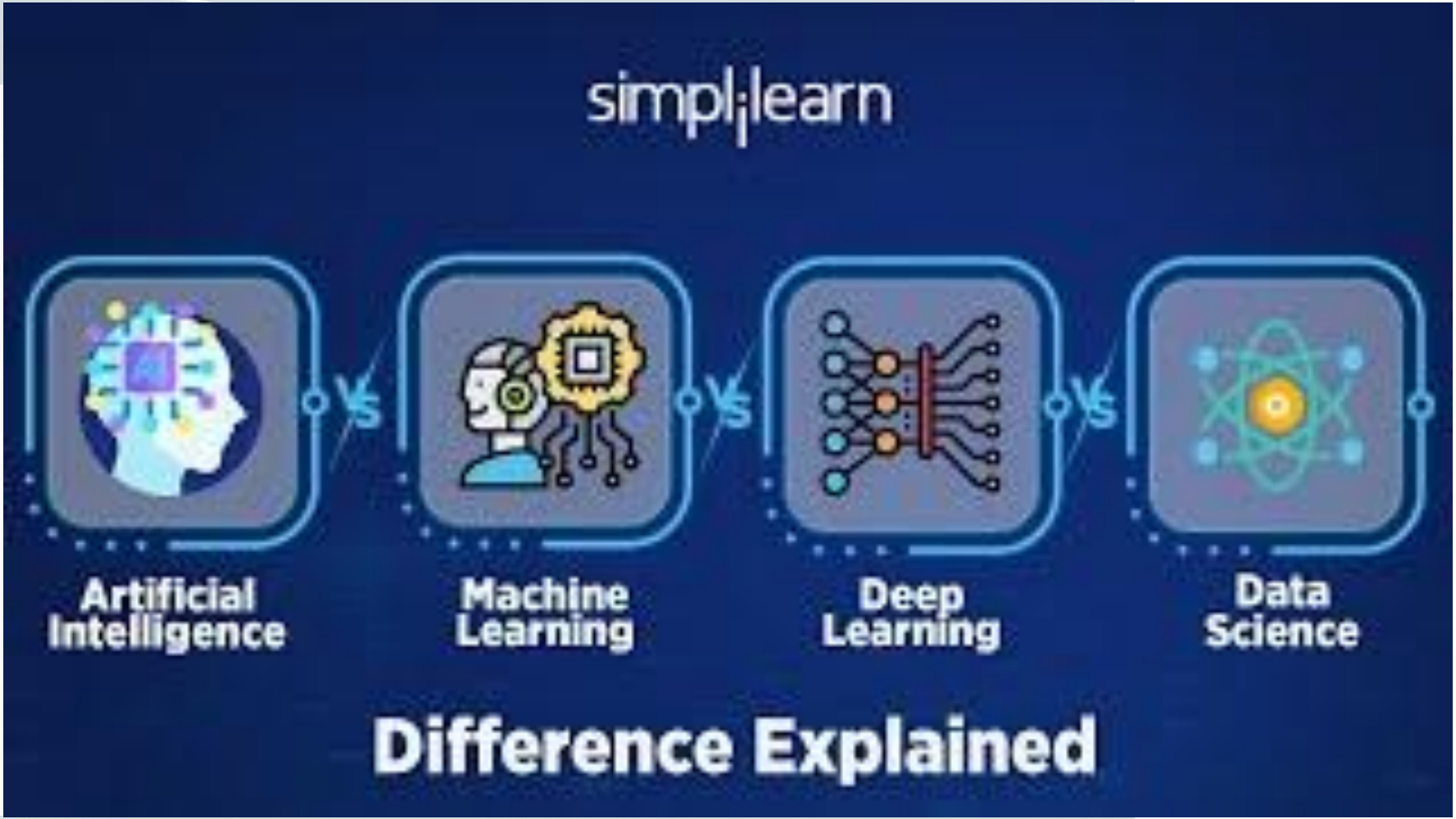
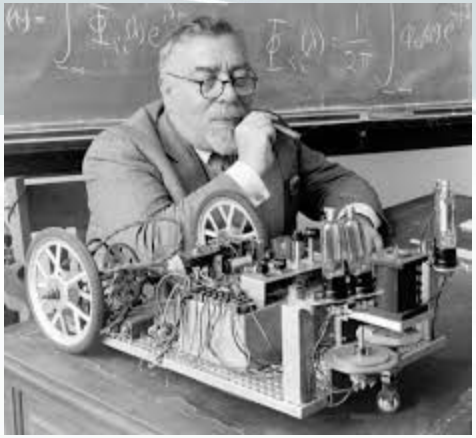
Introduction to Python

Fondren Library
Research Data Services

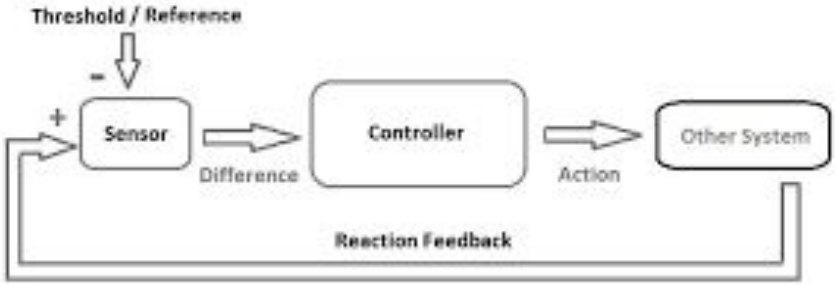

simplilearn
Artificial Intelligence
VS
Machine Learning
VS
Deep Learning
VS
Data Science
Difference Explained

History of Machine Learning



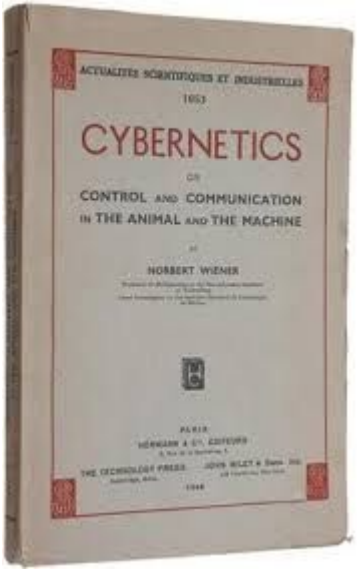
Key Words from this Time:

- Black Box
- Feedback Loops
- Networks
- System
- Input-Output
- Hidden Functions



A Cybernetic Loop

Published 1948



Macy Conferences (1941 to 1960)

https://en.wikipedia.org/wiki/Macy_conferences

Introduction to Machine Learning

Machine Learning (ML) has a rich history that traces back to the mid-20th century, rooted in the pursuit of artificial intelligence. Here are some key milestones:

1950s - The Dawn of AI:

- **1950:** Alan Turing proposes the Turing Test as a measure of machine intelligence.
- **1951:** Marvin Minsky and Dean Edmonds build the first neural network computer, the SNARC.

Arthur Samuel: Considered the "father of Machine Learning" for his pioneering work in the field, including the development of the checkers-playing program. Developed one of the earliest and most famous applications of machine learning, the checkers-playing program, in 1959.

Perceptron Model: Introduced by Frank Rosenblatt, an early neural network for binary classification.

Alan Turing: Proposed the concept of a machine that could simulate human intelligence.

Introduction to Machine Learning

Machine Learning (ML) has a rich history that traces back to the mid-20th century, rooted in the pursuit of artificial intelligence. Here are some key milestones:

1957 - Perceptron:

- Frank Rosenblatt invents the perceptron, an early neural network capable of learning binary classifiers.

1960s - AI Winter:

- Initial enthusiasm wanes due to limited computational power and unrealistic expectations, leading to reduced funding and interest.

1970s - Revival with Expert Systems:

- AI and ML see renewed interest with the development of expert systems, which use rules and logic to mimic human decision-making.

Introduction to Machine Learning

The modern era of machine learning has been marked by significant breakthroughs, leading to widespread applications and advancements:

- **1980s - Backpropagation and Neural Networks:**
  a. The backpropagation algorithm is popularized, allowing for the training of multi-layer neural networks.
- **1990s - Support Vector Machines and Ensemble Methods:**
  a. SVMs become popular for their effectiveness in classification tasks.
  b. Ensemble methods like boosting and bagging are developed to improve model performance.
- **2000s - Big Data and Deep Learning:**
  a. The rise of big data and increased computational power leads to the resurgence of neural networks, now called deep learning.
  b. **2006:** Geoffrey Hinton and his team demonstrate deep belief networks, reigniting interest in deep learning.
- **2010s - AI Renaissance:**
  a. Deep learning achieves state-of-the-art results in various fields, including computer vision, natural language processing, and game playing.
  b. AI becomes ubiquitous in applications like speech recognition, image classification, and autonomous driving.

Introduction to Machine Learning


100 SECONDS OF
MACHINE
LEARNING

Introduction to Machine Learning

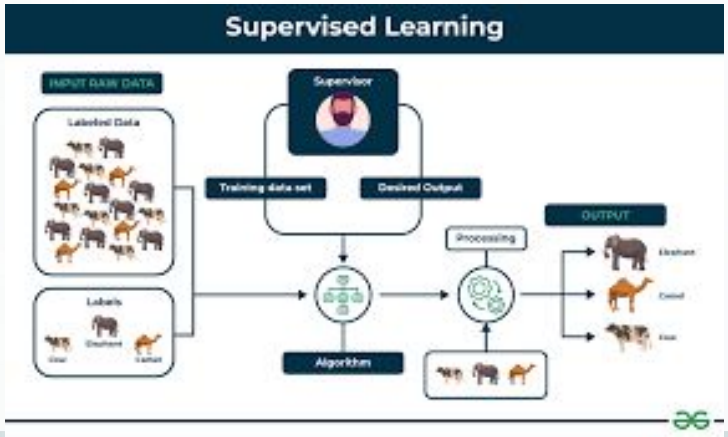
Machine Learning is a subset of artificial intelligence that trains algorithms to make predictions or decisions based on data.

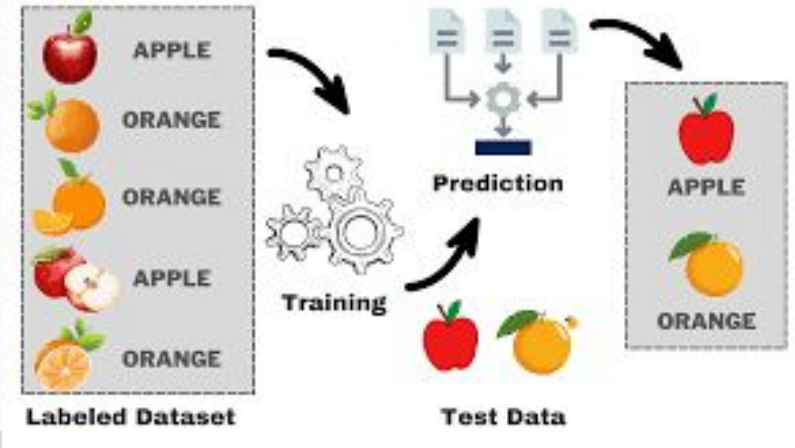
Types of Machine Learning:

- **Supervised Learning:** Uses labeled data to train the model. Examples include classification and regression tasks.
- **Unsupervised Learning:** Uses unlabeled data to identify patterns. Examples include clustering and dimensionality reduction.
- **Reinforcement Learning:** Uses trial and error to learn from actions and rewards. Often used in game playing and robotics.

Supervised Learning

Supervised learning is a type of machine learning where the model is trained on labeled data. This means that each training example is paired with an output label. The goal is for the model to learn to map inputs to outputs and make predictions on new, unseen data.





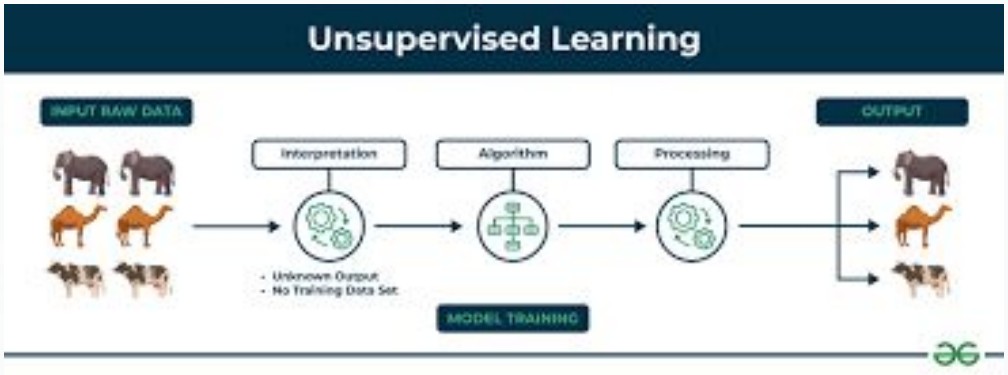
Supervised Learning

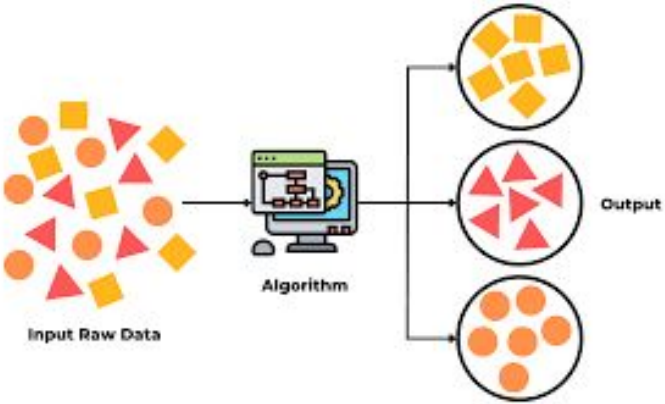
Supervised learning is a type of machine learning where the model is trained on labeled data. This means that each training example is paired with an output label. The goal is for the model to learn to map inputs to outputs and make predictions on new, unseen data.

```
# A simple example of supervised learning concept
from sklearn.linear_model import LinearRegression
# Dummy data
X = [[1], [2], [3], [4], [5]]
y = [1, 3, 5, 7, 9]

# Model training
model = LinearRegression()
model.fit(X, y)

# Predicting
```

Unsupervised Learning

Unsupervised learning involves training a model on data without labeled responses. The model tries to learn the patterns and structure from the data. Common tasks include clustering, dimensionality reduction, and association.



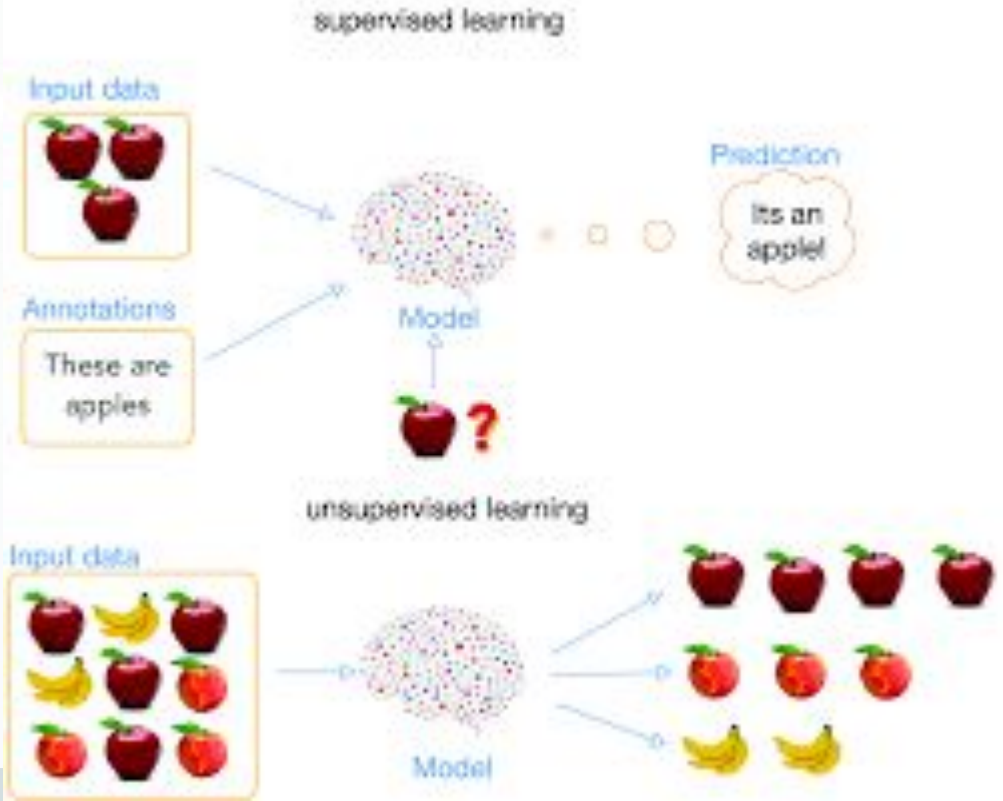


Unsupervised Learning

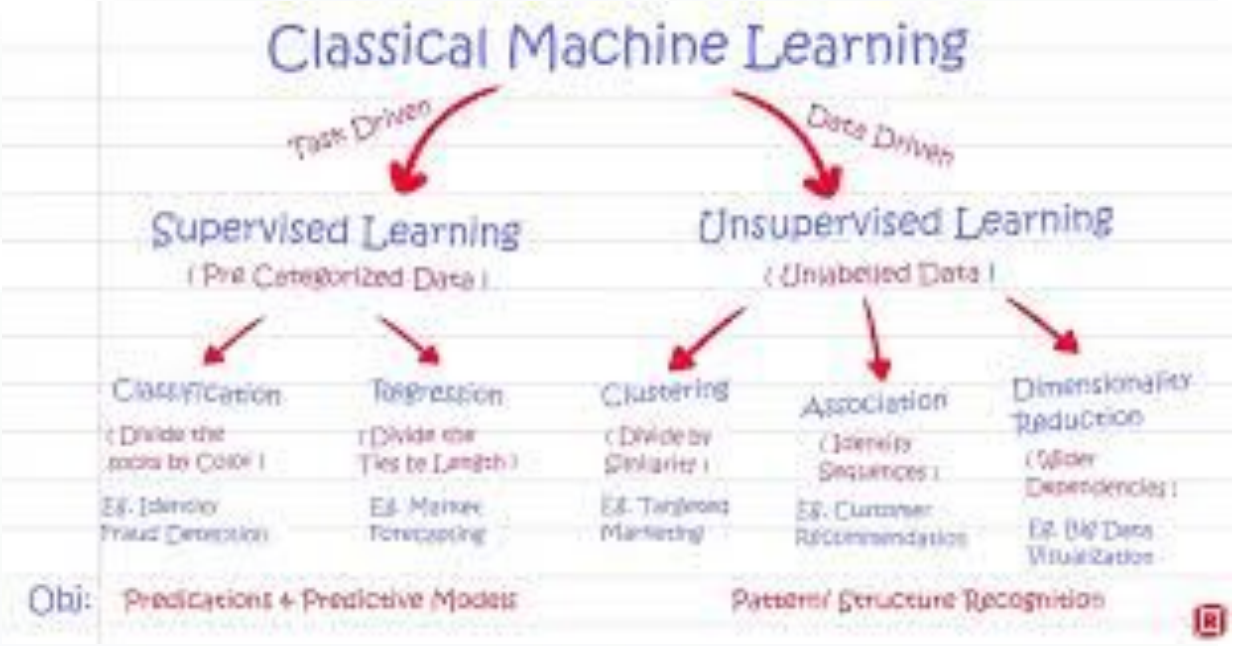
Unsupervised learning involves training a model on data without labeled responses. The model tries to learn the patterns and structure from the data. Common tasks include clustering, dimensionality reduction, and association.

```
# A simple example of unsupervised learning concept
from sklearn.cluster import KMeans
import numpy as np

# Dummy Data
X = np.array ([[1, 2], [1, 4], [1, 0],
               [4, 2], [4, 4], [4,0]])

# Model Training
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# Predicting clusters

clusters = kmeans.predict(X)
print(f"Clusters: {clusters}")
```

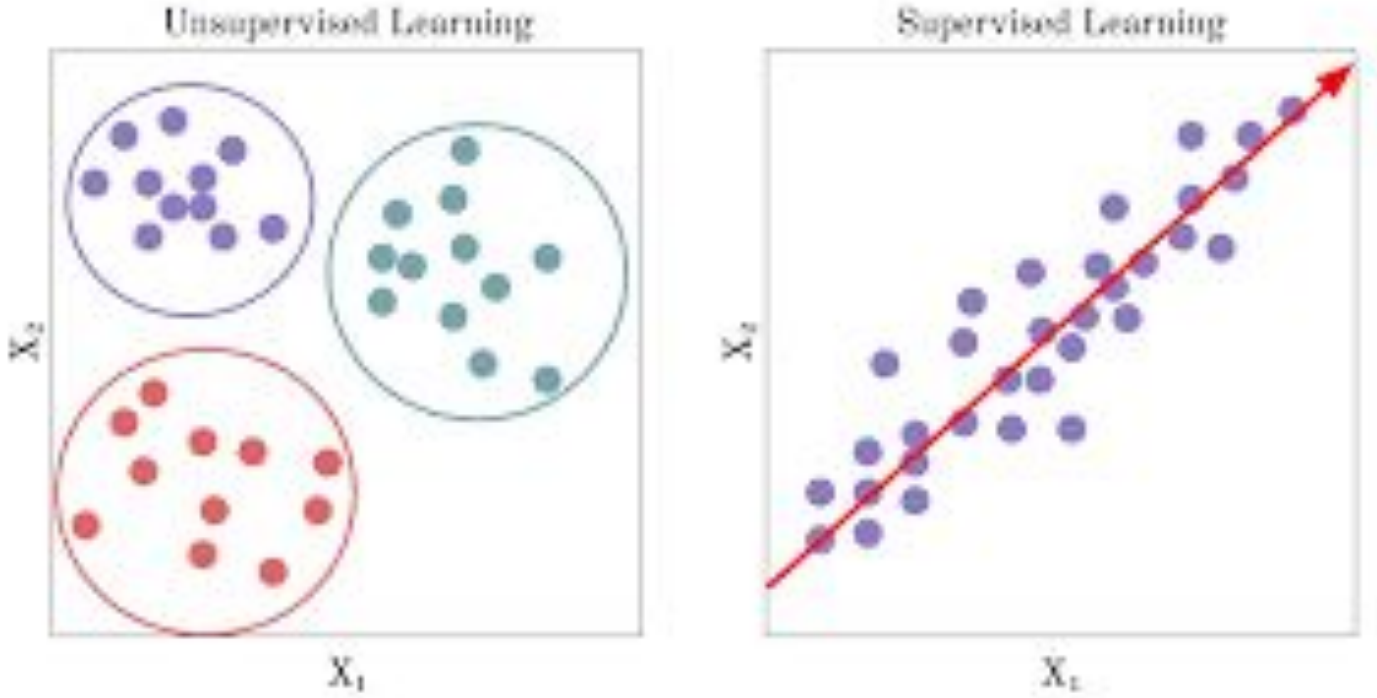
Unsupervised Learning vs Supervised Learning



Unsupervised Learning vs Supervised Learning



Unsupervised Learning vs Supervised Learning



Dealing with Missing Data

adding new columns in dataframes

- we can create new columns in our dataframe as well

adding new columns in dataframes

- we can create new columns in our dataframe as well

```
df['new'] = df['W'] + df['Y']
```

```
df
```

	W	X	Y	Z	new
A	-1.085631	0.997345	0.282978	-1.506295	-0.802652
B	-0.578600	1.651437	-2.426679	-0.428913	-3.005279
C	1.265936	-0.866740	-0.678886	-0.094709	0.587050
D	1.491390	-0.638902	-0.443982	-0.434351	1.047408
E	2.205930	2.186786	1.004054	0.386186	3.209984

removing columns in dataframes

- always mind the syntax
- **df.drop('new', axis=1, inplace = True)**
- why are these parameters necessary?
- in python, **axis=1** refers to column identification and **axis=0** refers to row identification
- **inplace = True** tells python to modify the existing dataframe (save vs. save as)

creating subsets of original dataframes

- **df2 = df[[‘W’, ‘X’]]**
- our new dataframe **df2** will now only have two columns
- when would we typically subset?

dropping rows in dataframes

- what do you notice about the syntax below?

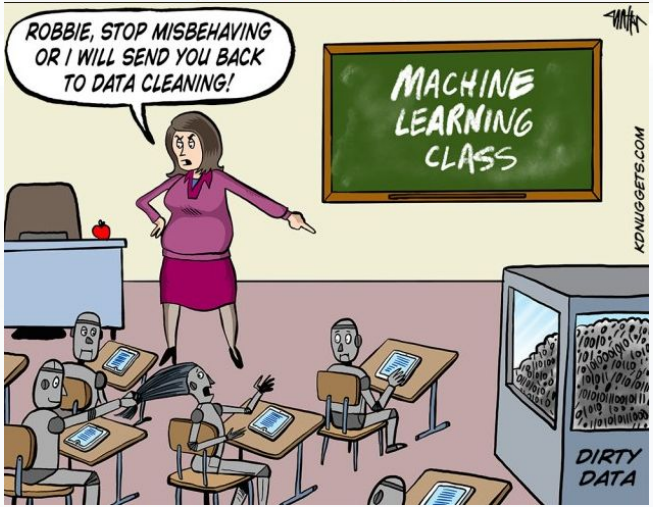
```
df.drop('E', axis=0)
```

	W	X	Y	Z	new
A	-1.085631	0.997345	0.282978	-1.506295	-0.802652
B	-0.578600	1.651437	-2.426679	-0.428913	-3.005279
C	1.265936	-0.866740	-0.678886	-0.094709	0.587050
D	1.491390	-0.638902	-0.443982	-0.434351	1.047408

renaming columns in dataframes

- messy data sometimes means messy columns

```
df = df.rename(columns={'Unnamed: 0': 'newName1', 'oldName2': 'newName2'})
# Or rename the existing DataFrame (rather than creating a copy)
df.rename(columns={'oldName1': 'newName1', 'oldName2': 'newName2'}, inplace=True)
```



missing values

- there are a few techniques to check for missing values
- let’s create a sample dataframe with some missing values to work

```
import pandas as pd
import numpy as np
```

```
df = {'A':[1,2,np.nan], 'B':[3,np.nan, np.nan]}
df = pd.DataFrame(df)
```

missing values simple example

- first let’s check to see if there are any missing values

	A	B
0	1.0	3.0
1	2.0	NaN
2	NaN	NaN

```
df.isnull()
```

	A	B
0	False	False
1	False	True
2	True	True

- **null** function is one way

missing values simple example

- we can sum up total number of missing values by column

```
df.isnull().sum()
```

```
A    1
B    2
dtype: int64
```

```
df.isnull().sum().sum()
```

```
3
```

- we can also sum up total number of missing values for the entire dataframe

missing values simple example

- let's focus on simply dropping missing values

```
df.dropna()
```

	A	B
0	1.0	3.0

- **df.dropna()** drops all missing values
- **df.dropna(axis=1)** drops na values only from columns that contain na values (if a column doesn't have any missing values, it won't be dropped)
- **df.dropna(thresh=2)** sets a threshold (na values will only be dropped if that threshold is met)