CMPE 180A DSA with Python

Scientific Python packages
Introduction to neural networks

Sanja Damjanovic

Week 6

8/19/2025

CMPE 180A Week 6 Outline

- Scientific Python packages
  - Lab tutorials
  - Homework
- Introduction to machine learning and neural networks
  - Brief introduction to machine learning
  - Machine learning system and terminology
  - Data set
  - Multilayer artificial neural network
  - Lab tutorials
- Machine learning project

Structured data

-

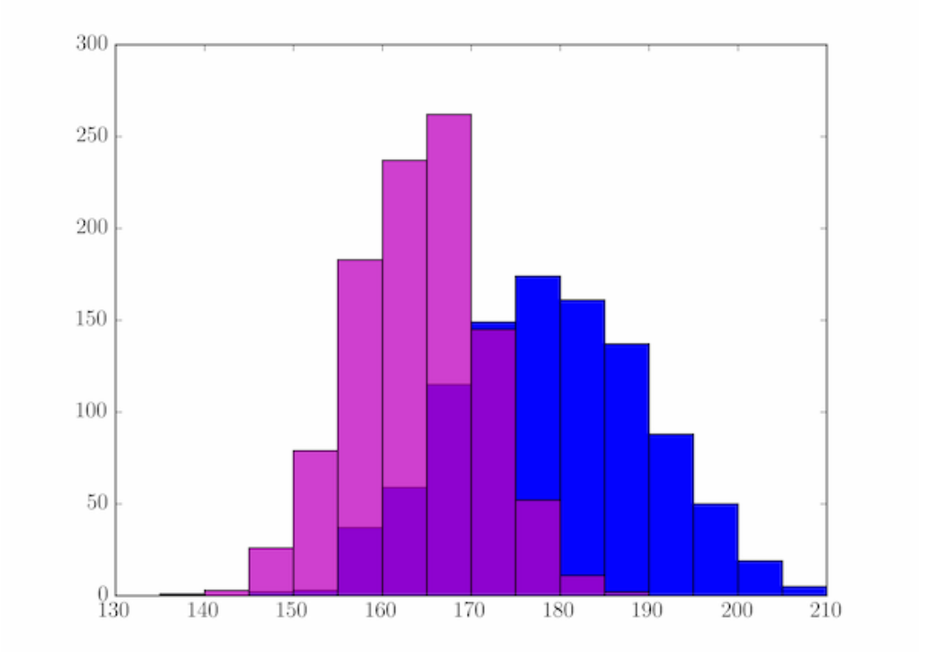
Name	Population	Mass /tonnes
Bowhead whale	9000	60
Blue whale	20000	120
Fin whale	100000	70
Humpback whale	80000	30
Gray whale	26000	35
Atlantic white-sided dolphin	250000	0.235
Pacific white-sided dolphin	1000000	0.15
Killer whale	100000	4.5
Narwhal	25000	1.5
Beluga	100000	1.5
Sperm whale	2000000	50
Baiji	13	0.13
North Atlantic right whale	300	75
North Pacific right whale	200	80
Southern right whale	7000	70

Homework 4

Histogram and Tabular Overview

Histogram should also have:

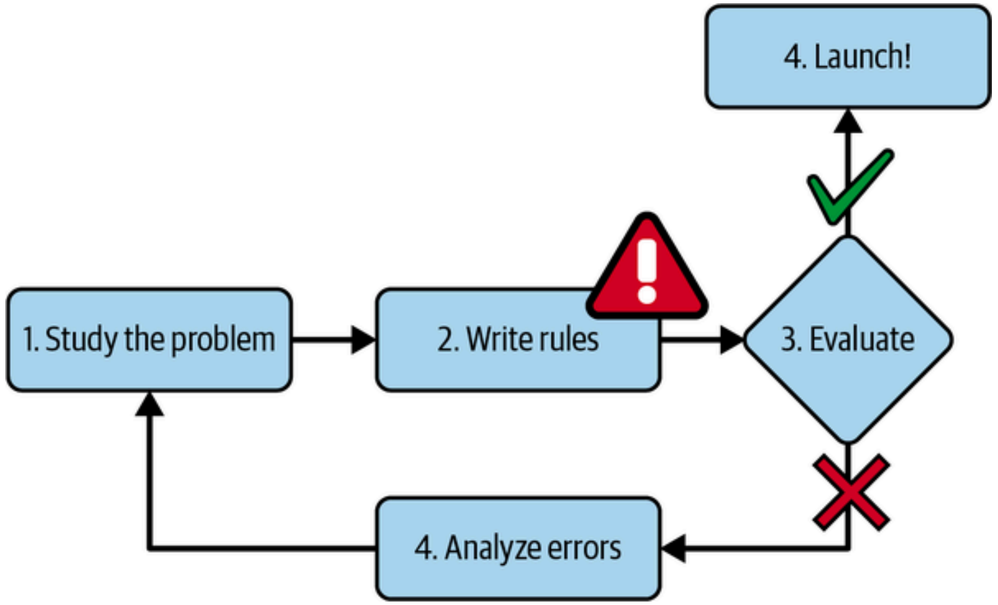
- Labeled axes with units
- Legend
- Title



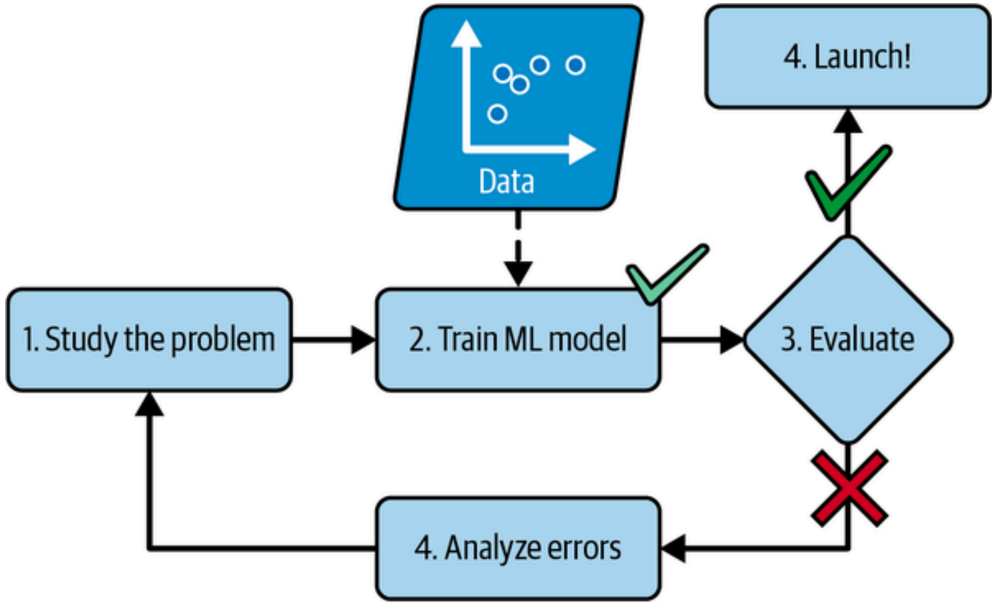
Height (cm)	Female	Male
135–140	0	1
140–145	3	0
145–150	26	2
150–155	79	3
155–160	183	37
160–165	237	59
165–170	262	115
170–175	145	149
175–180	52	174
180–185	11	161
185–190	2	137
190–195	0	88
195–200	0	50
200–205	0	19
205–210	0	5
Mean (cm):	164.1	178.8
Std (cm):	7.4	10.8

Introduction to machine learning and neural networks

Traditional Programming v.s. Machine Learning Approach

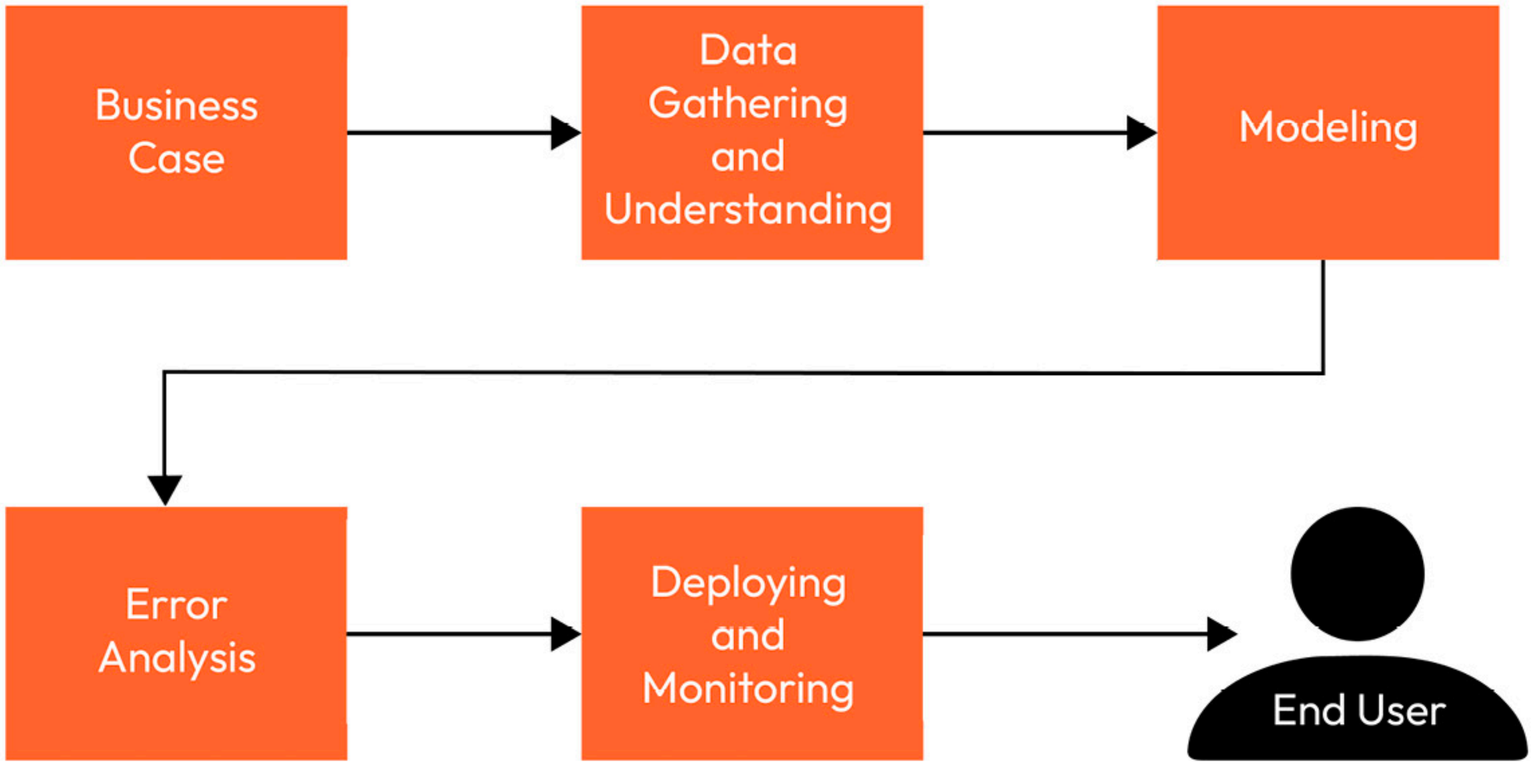


The traditional approach



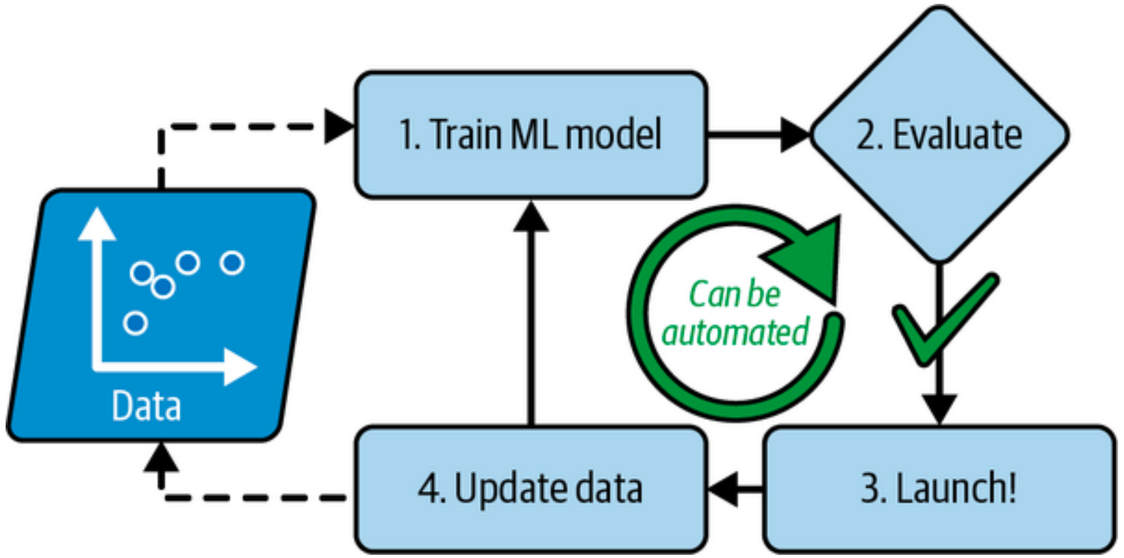
The machine learning approach

ML project lifecycle



ML model Automatic Update

- Automatically adapting to change

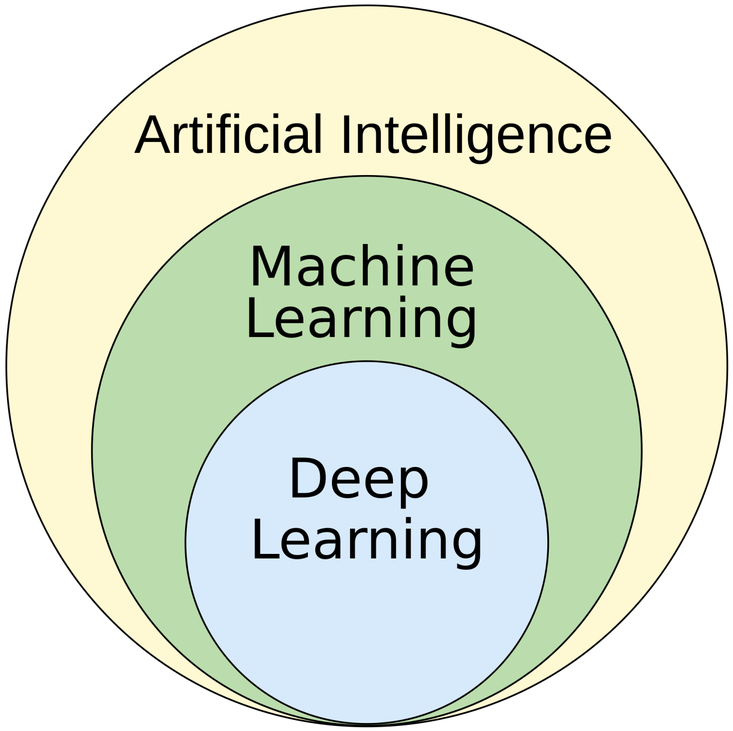


Machine learning

- Writing programs to learn from data
  - E.g. learn to distinguish ‘spam’ from ‘ham’
  - Predict house price, etc
- Machine learning as subfield of AI

Image source

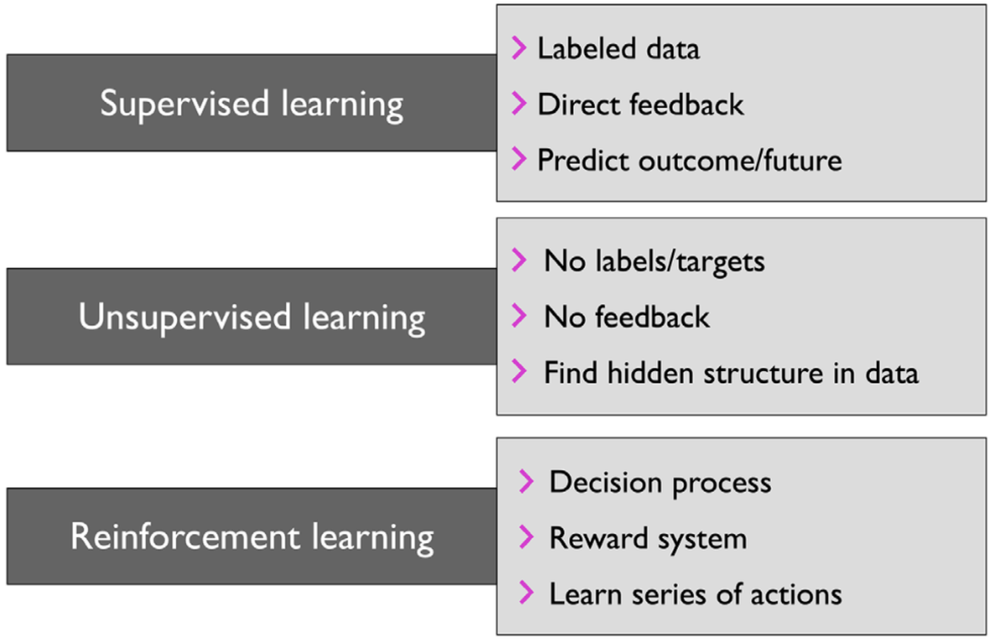
Machine learning as subfield of AI

- 

Types of Machine Learning Algorithms

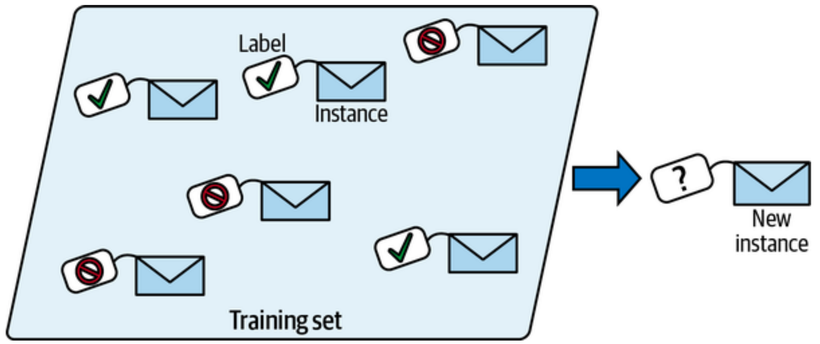
Based on training supervision

- Supervised learning
- Unsupervised learning
- Reinforcement learning

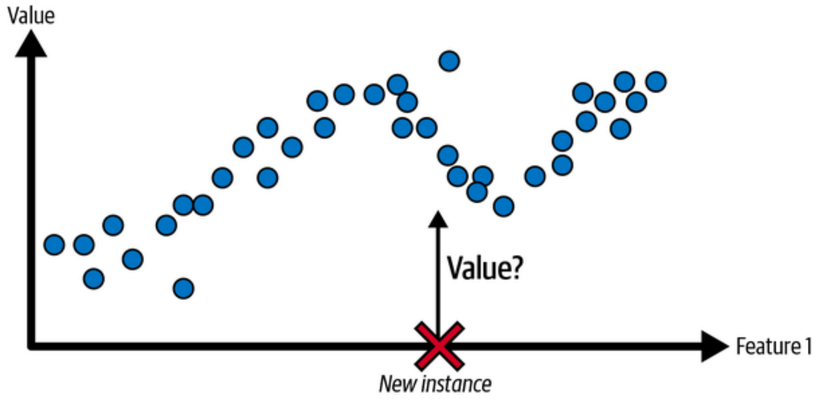


Supervised learning

- A labeled training set for spam classification (an example of supervised learning)

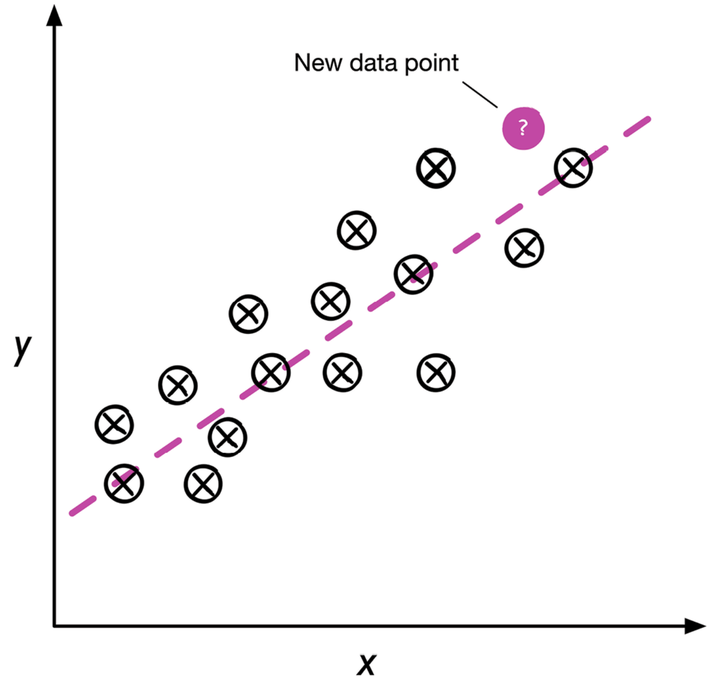


- A regression problem: predict a value, given an input feature (there are usually multiple input features, and sometimes multiple output values)



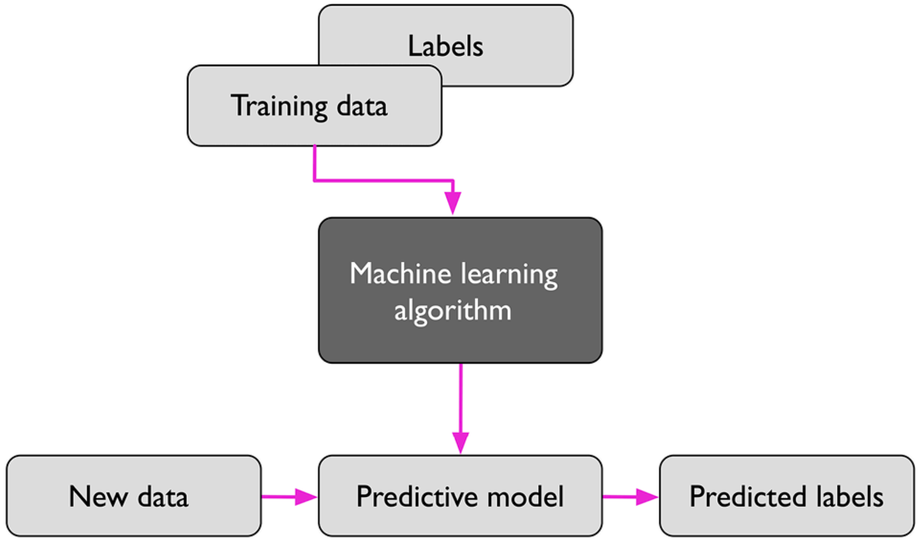
Regression

- supervised learning is the prediction of continuous outcomes



Supervised learning process

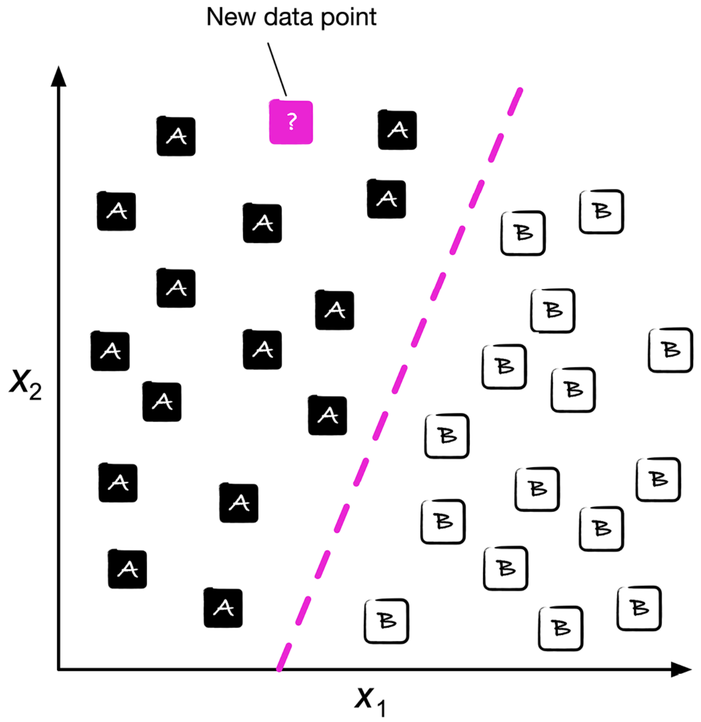
- learn a model from labeled training data that allows us to make predictions about unseen or future data
  - classification
  - Regression



Classification

Supervised

- Classifying a new data point

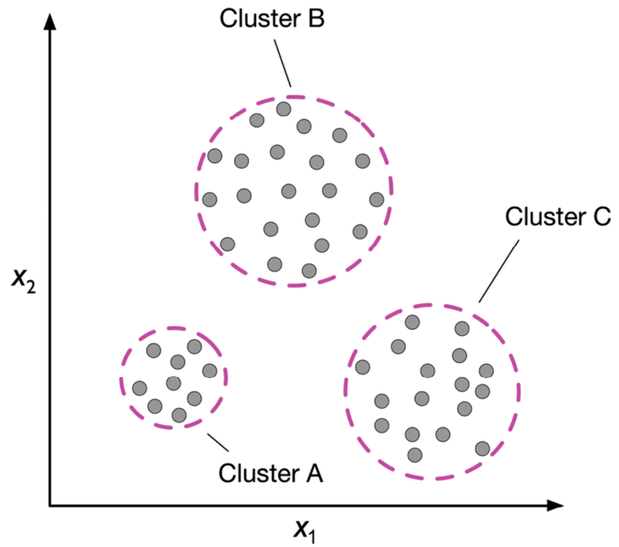


Clustering

Unsupervised learning

- **Clustering** is an exploratory data analysis or pattern discovery technique that allows us to organize a pile of information into meaningful subgroups (**clusters**)

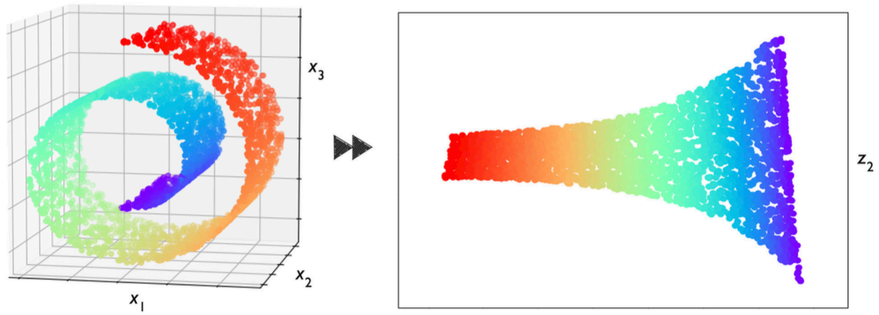
- for structuring information and deriving meaningful relationships from data



Dimensionality Reduction

Unsupervised learning

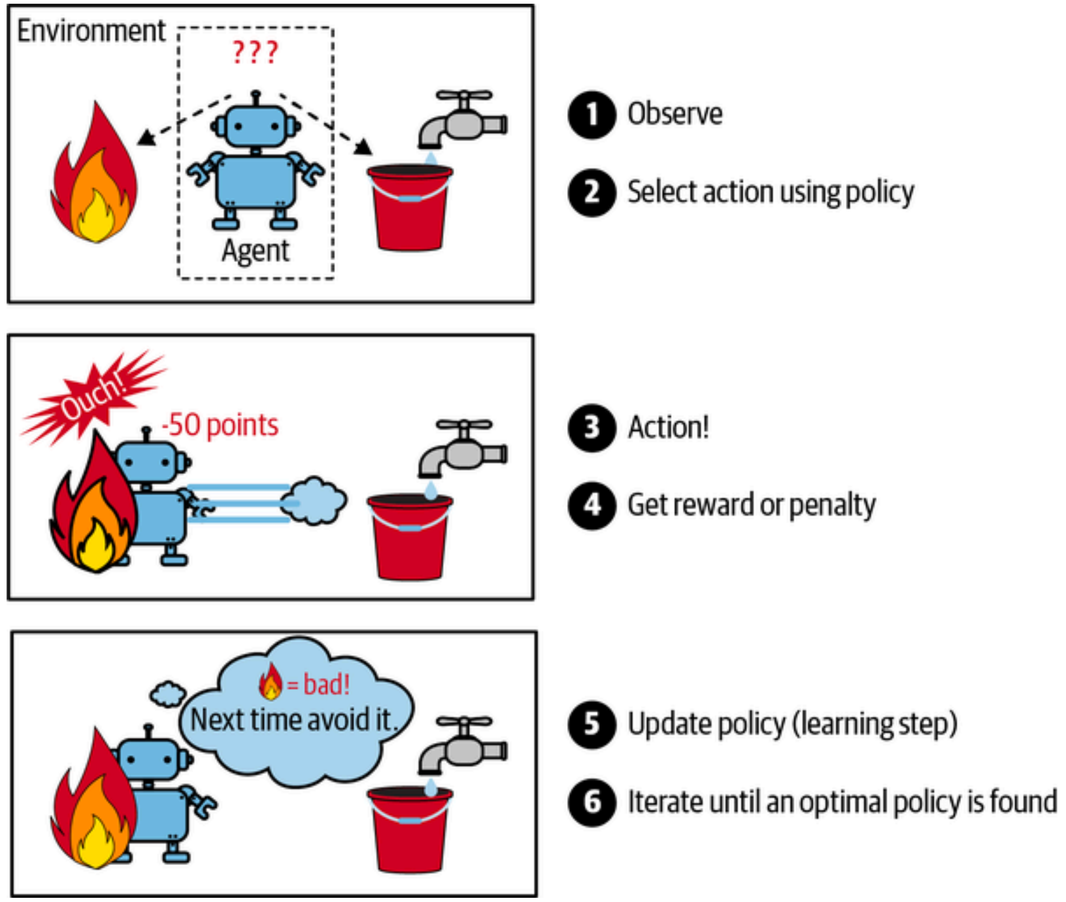
- For compression
- Example 3D —> 2D:



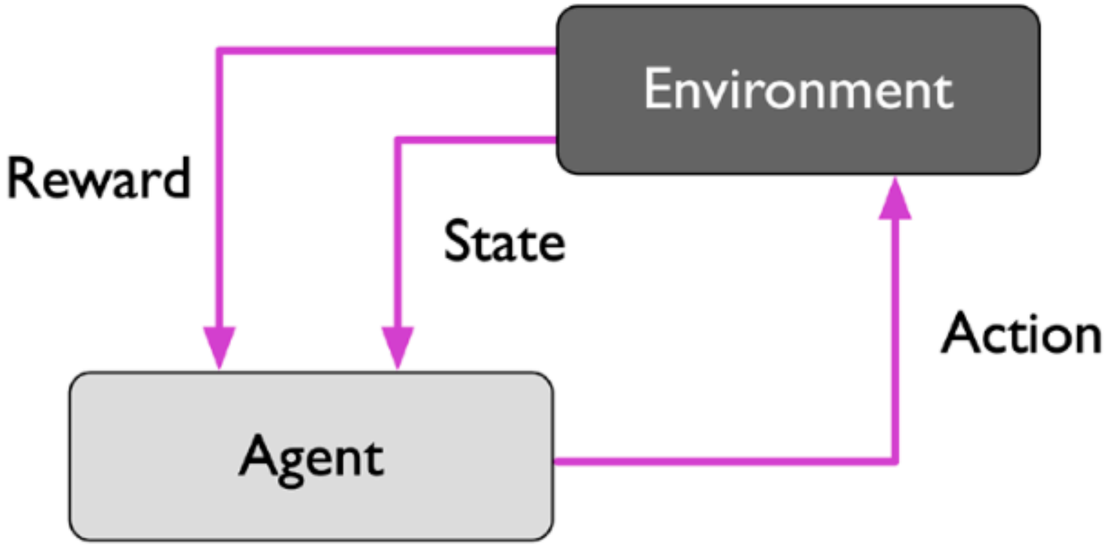
Reinforcement learning

The agent learns the optimal policy

- Interactive
- Develop a system - agent - that improves there performance based on interactions with the environment
- e. g. Learning to play chess



Reinforcement Learning Process



ML project

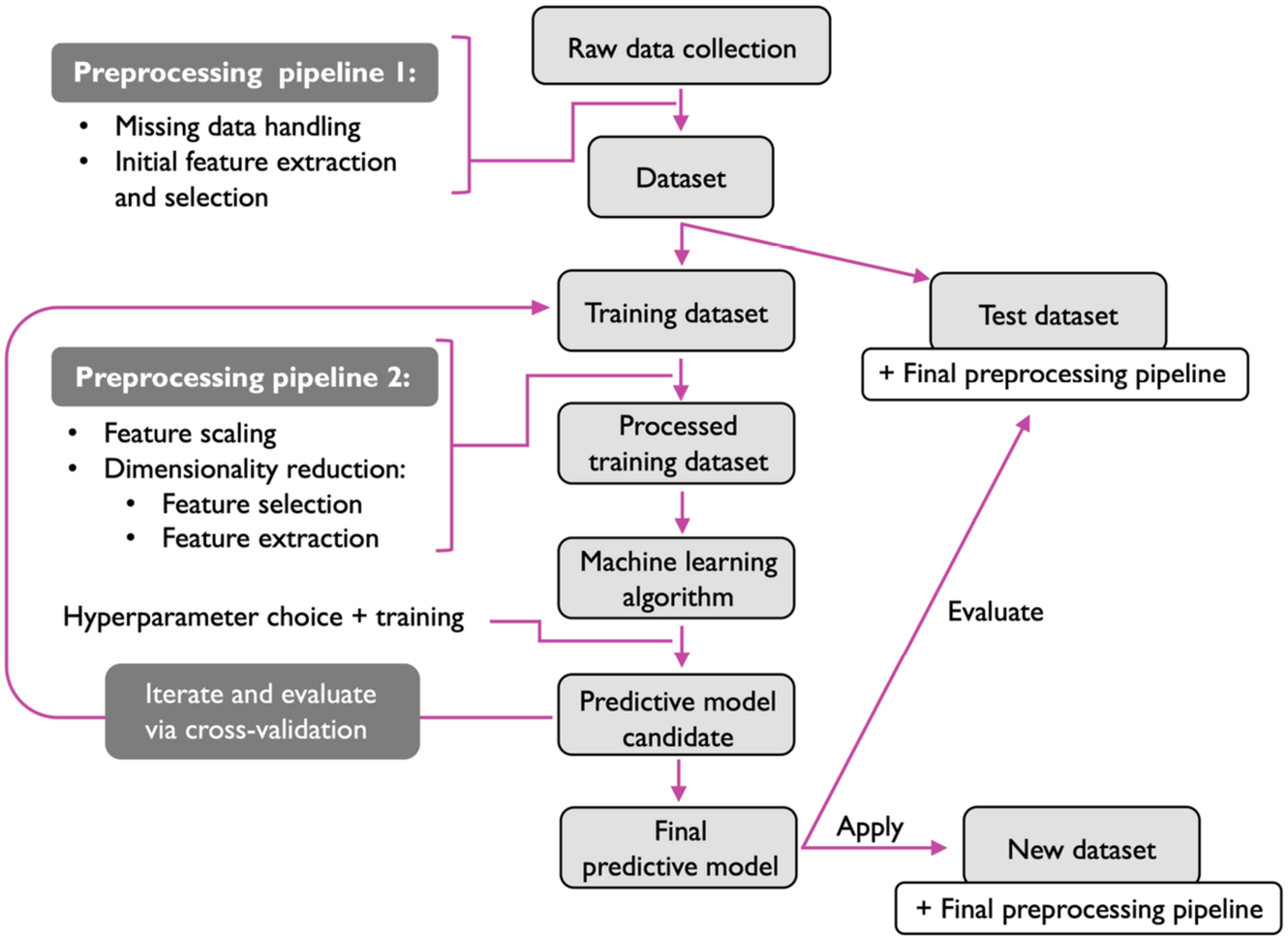
- You studied the data.
- You selected a model.
- You trained it on the training data (i.e., the learning algorithm searched for the model parameter values that minimize a cost function).
- Finally, you applied the model to make predictions on new cases (this is called *inference*), hoping that this model will generalize well.

Data set

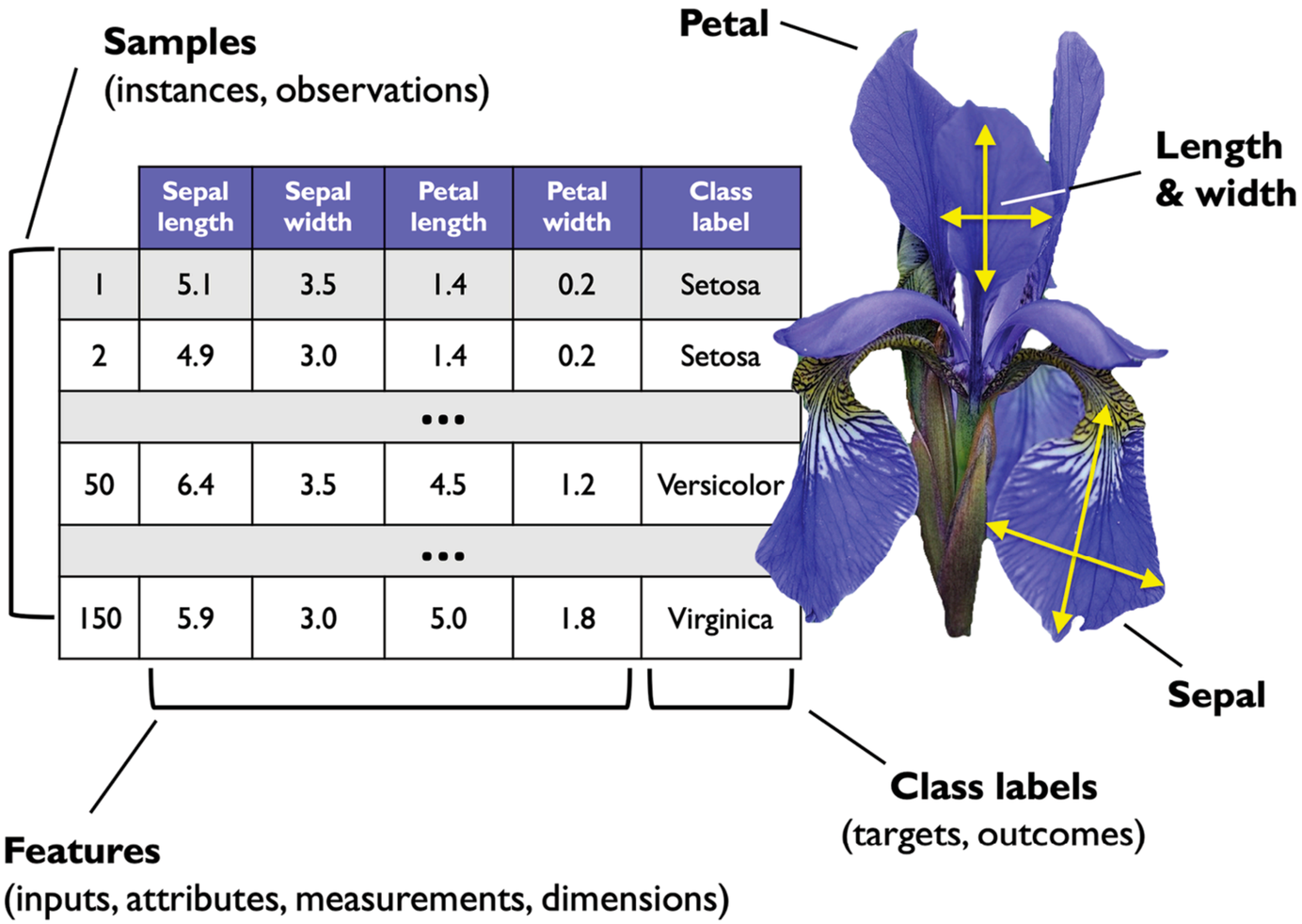
- data exploration to gain insights into the data
- data preprocessing to handle various issues such as:
  - formatting problems, missing values, duplicates, removal of irrelevant features, and handling outliers, imbalanced data, and categorical features

A roadmap for building machine learning systems

Workflow



The Iris Dataset



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

- The **Iris dataset** contains the measurements of 150 Iris flowers from three different species—*Setosa, Versicolor, and Virginica*
- each flower example represents one row in the dataset
- the flower measurements in centimeters are stored as columns - the **features** of the dataset

https://archive.ics.uci.edu/dataset/53/iris

IRIS Data set

- Training examples: The i-th training example, the j-th dimension

For example, $x_1^{(150)}$ refers to the first dimension of flower example 150, the sepal length. Each row in matrix $\boldsymbol{X}$ represents one flower instance and can be written as a four-dimensional row vector, $\boldsymbol{x}^{(i)} \in \mathbb{R}^{1\times 4}$:

$$\boldsymbol{X}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

And each feature dimension is a 150-dimensional column vector, $\boldsymbol{X}^{(i)} \in \mathbb{R}^{150\times 1}$. For example:

$$\boldsymbol{x_j} = \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \\ \cdots \\ x_j^{(150)} \end{bmatrix}$$

- Target variables: class labels
  - 150 -dimensional column vector

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \cdots \\ y^{(150)} \end{bmatrix}, \text{ where } y^{(i)} \in \{\text{Setosa, Versicolor, Virginica}\}$$

Machine learning terminology

- **Training example**: A row in a table representing the dataset: synonymous observation, record, instance, or sample (in most contexts, sample refers to a collection of training examples)
- **Training**: Model fitting, for parametric models similar to parameter estimation
- F**eature, abbrev. *x***: A column in a data table or data (design) matrix: synonymous with predictor, variable, input, attribute, or covariate
- **Target, abbrev. *y***: Synonymous with outcome, output, response variable, dependent variable, (class) label, and ground truth
- **Loss function**: Often used synonymously with a ***cost* function or an *error* function**
  - the term “loss” refers to the loss measured for a single data point, and the cost is a measurement that computes the loss (average or summed) over the entire dataset.

Good ML Model

- a good model will generalize well to new cases
- Splitting your data into sets:
  - the *training set*
  - *validation set*
  - the *test set*

ML Challenges

Leading to a trained model that is unlikely to make accurate predictions

- Insufficient quantity of training data
- Nonrepresentative training data
- Poor quality data
- Irrelevant features
- Overfitting the training data
- Underfitting the training data

Questions

- Q1: What type of algorithm would you use to allow a robot to walk in various unknown terrains?
- Q2: What type of algorithm would you use to segment your customers into multiple groups?
- Q3: Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?
- Answer options:
  - A: Supervised
  - B: Unsupervised
  - C: Reinforcement learning

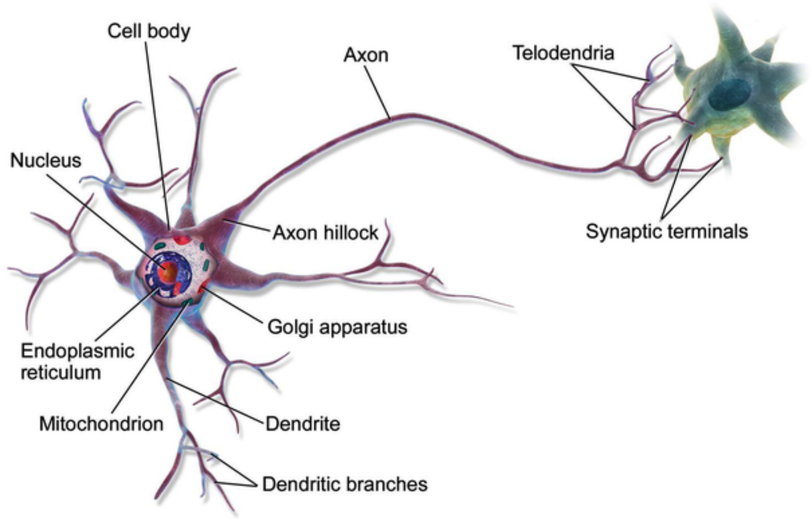
Multilayer Artificial Neural Network

Multilayer Artificial Neural Network

- Deep learning subfield of ML
- Training artificial NN
- Neurons - building blocks of NN
- Modeling complex functions with artificial neural networks

A Biological Neuron

- Human brain
  - inspiration for Artificial Neural Networks (ANNs)
  - organized in a vast network of billions, with each neuron typically connected to thousands of other neurons

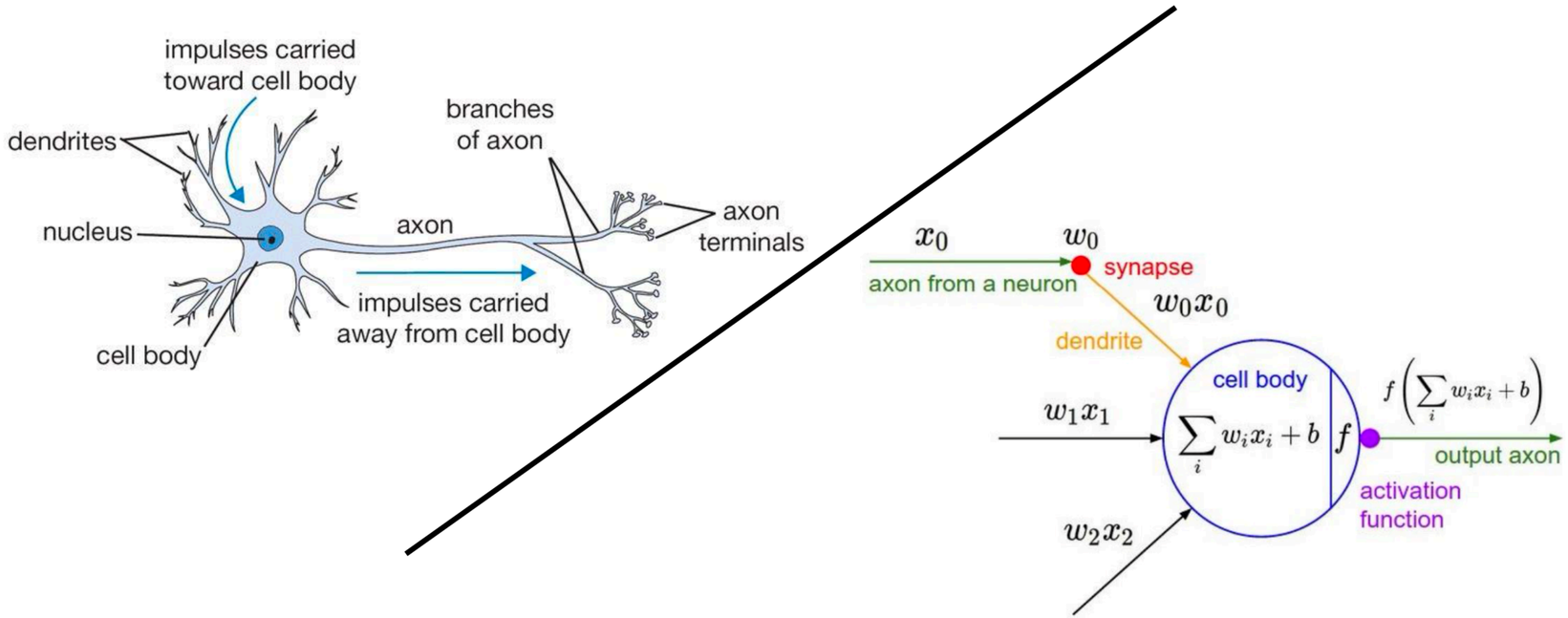


Credit: Stanford CS 231n

The Perceptron

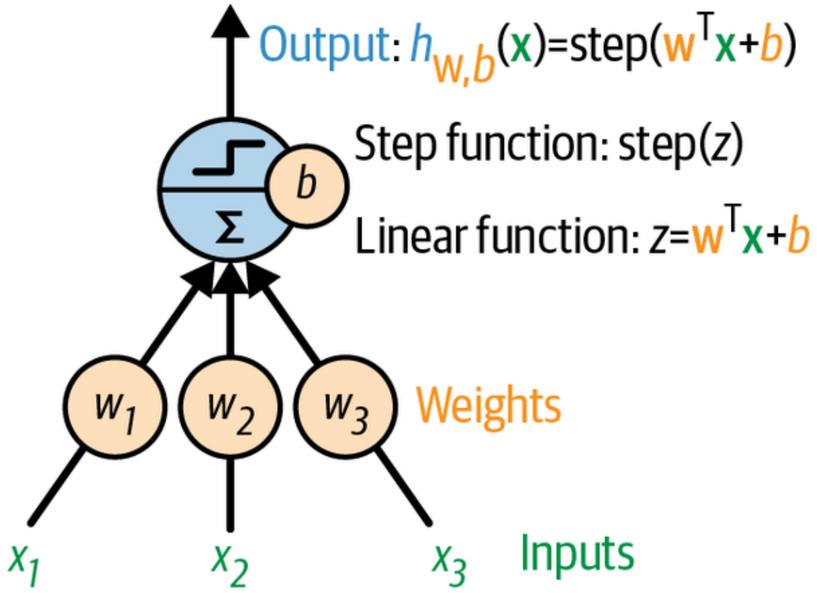
The structural building block of deep learning

- Simple computational model for neuron
- Weights
- Bias
- Activation function



Credit: Stanford CS 231n

The Perceptron with TLU



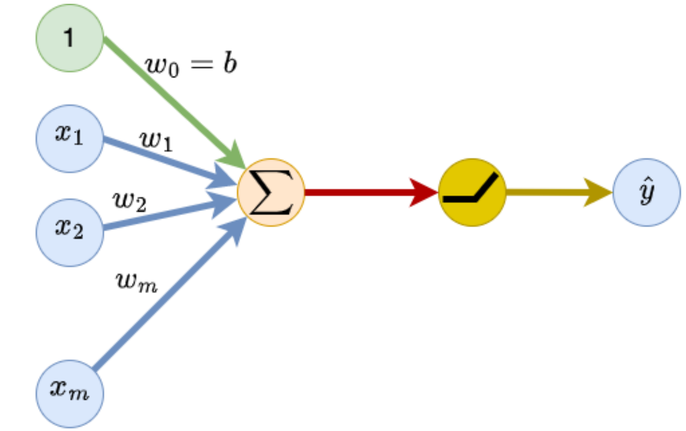
- The simplest ANN
- a threshold logic unit (TLU)
- $z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b = \mathbf{w}_T \mathbf{x} + b$
- *Parameters:* the input weights **w** and the bias term *b*
- *Step function:* $h_w(\mathbf{x}) = \text{step}(z)$

$$\text{heaviside}\,(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}\,(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

The Perceptron

- Weights $w = [w_0, w_1, w_2, \ldots, w_m]$
- Bias $w_0 = b$
- Training example/instance:
  - $x = [1, x_1, x_2, \ldots x_m]$
- Output $\hat{y}$
- The perceptron algorithm

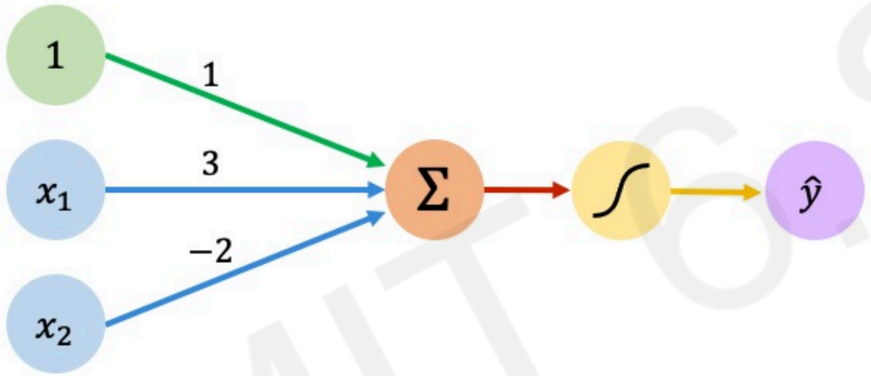
Feed-Forward Propagation



A Single Unit Perceptron

- Dot product: $w \cdot x = \sum_{i=1}^{m} w_i \cdot x_i$
- Linear units: Elements that compute $b + w \cdot x$
- Non-linearity: $\hat{y} = \begin{cases} 1 & \text{if } b + w \cdot x > 0 \\ 0 & \text{if otherwise} \end{cases}$
- Binary classification output $\hat{y} = f(\mathbf{w} \cdot \mathbf{x})$
- Training: $w = ?$
  - System parameters: hyperparameters
  - the perceptron algorithm: function approximation
  - Training examples

Perceptron Example



We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

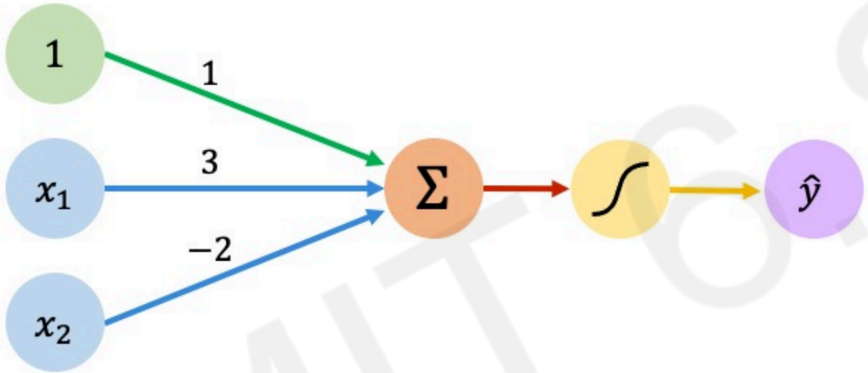
This is just a line in 2D!

Source pdf: MIT S191

Source video: MIT S191

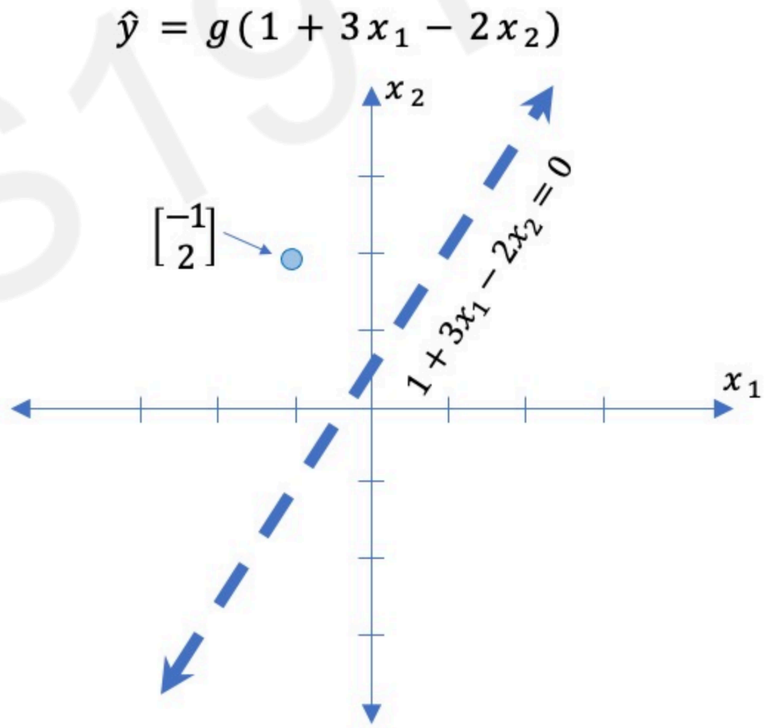
Perceptron Example

Inference



Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

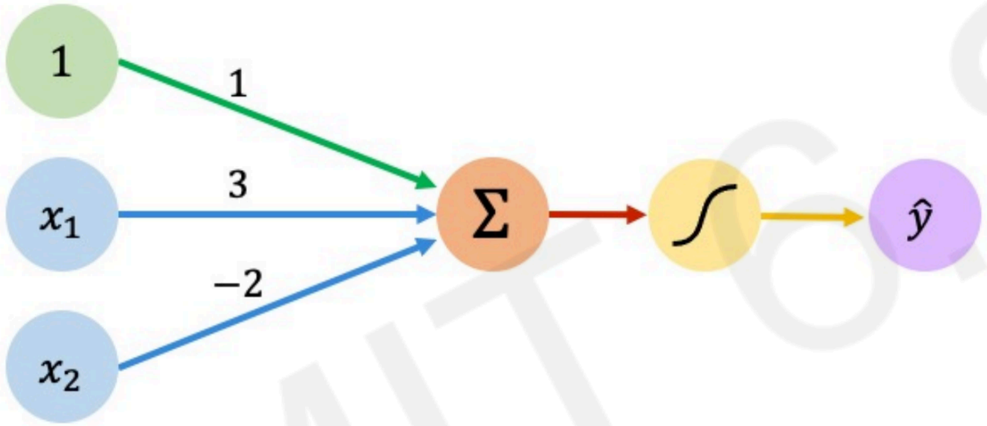
$$\hat{y} = g\left(1 + (3 * -1) - (2 * 2)\right)$$
$$= g(-6) \approx 0.002$$

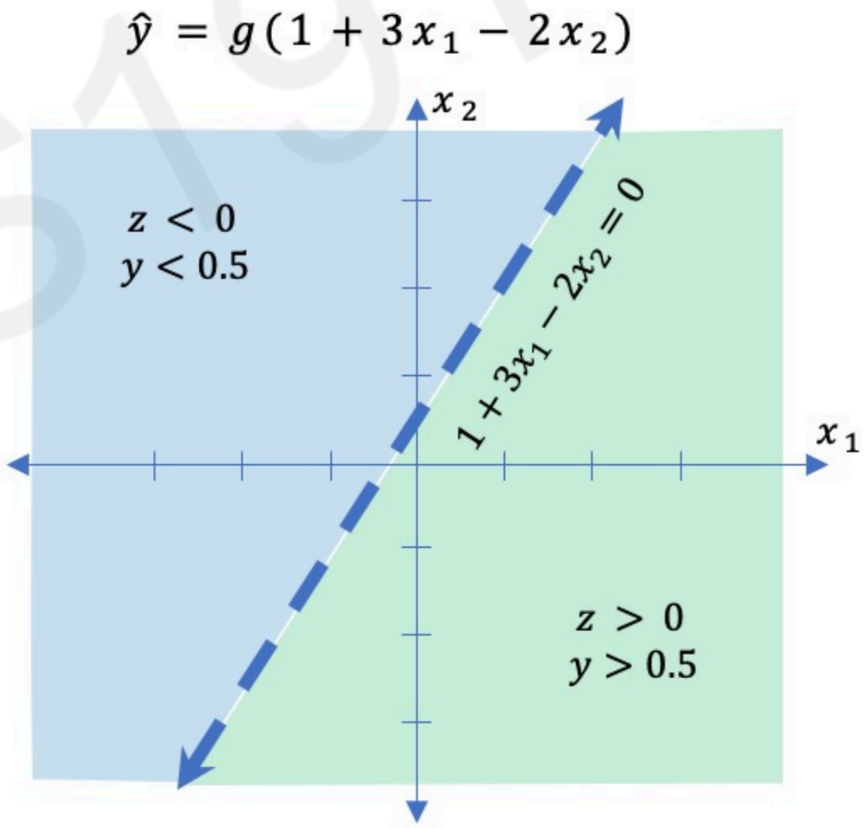


Source pdf: MIT S191

Source video: MIT S191

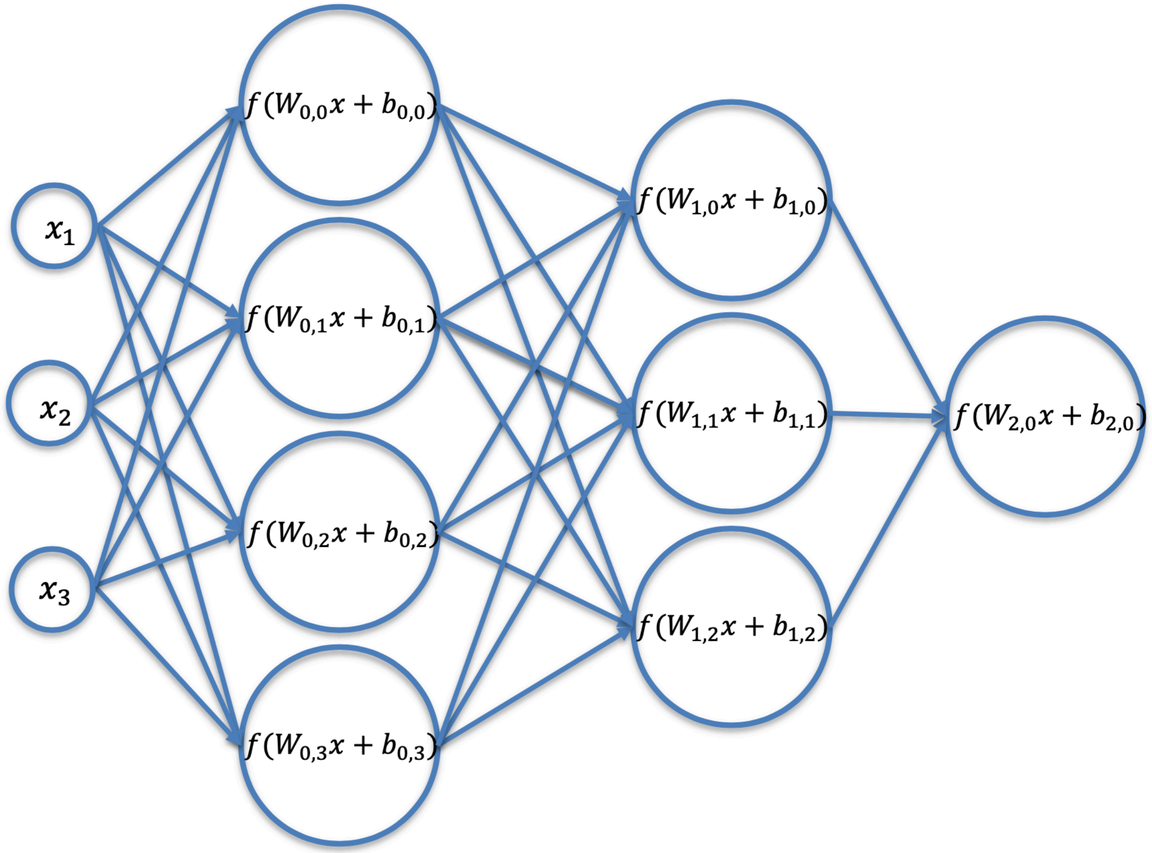
Perceptron Example



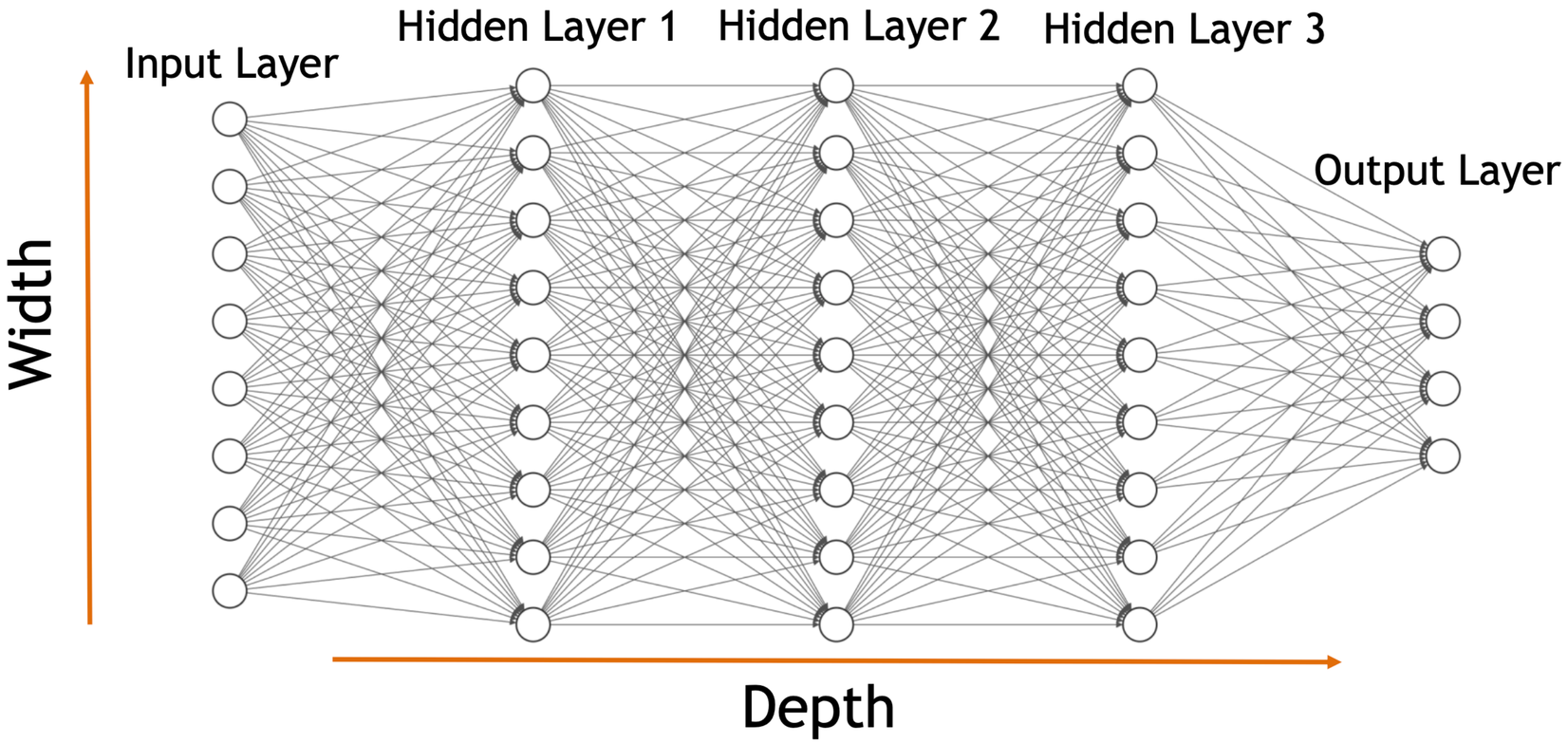


Net of Artificial Neurons

- **Inspired by biological neurons**



Neural Network



Scikit-learn

Python library

scikit learn Install User Guide API Examples Community More Go

scikit-learn

Machine Learning in Python

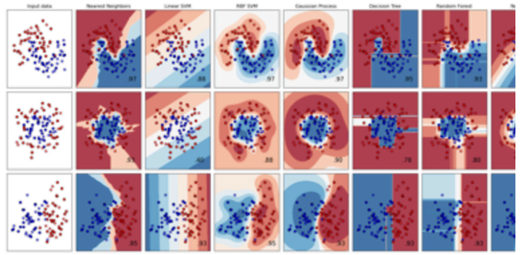
Getting Started | Release Highlights for 1.3 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

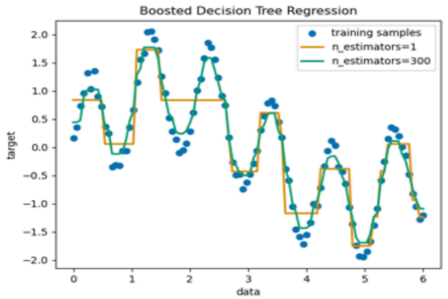


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

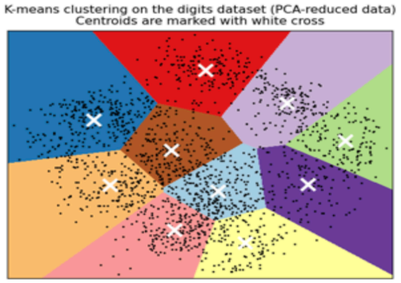


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...



Examples

Reference #1: MLbook1

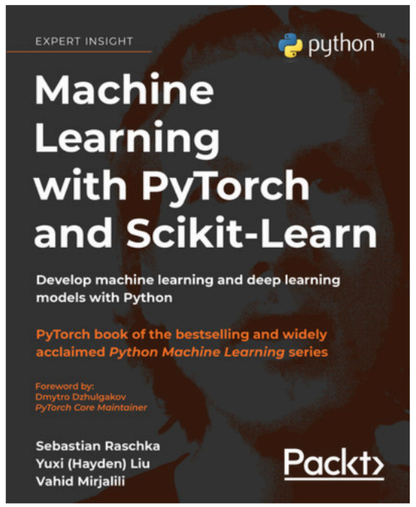
Available on https://www.oreilly.com/ or app with SJSU log in

- Machine Learning with PyTorch and Scikit-Learn , S. Raschka, Y. Liu, V. Mirjalili, GitHubRepo
- Chapters: 2, 11
- beginner friendly

Machine Learning with PyTorch and Scikit-Learn

★★★★★ 5 reviews

By Sebastian Raschka, Yuxi Liu, Vahid Mirjalili, Dmytro Dzhulgakov



TIME TO COMPLETE:
21h 56m

TOPICS:
PyTorch

PUBLISHED BY:
Packt Publishing

PUBLICATION DATE:
February 2022

PRINT LENGTH:
774 pages

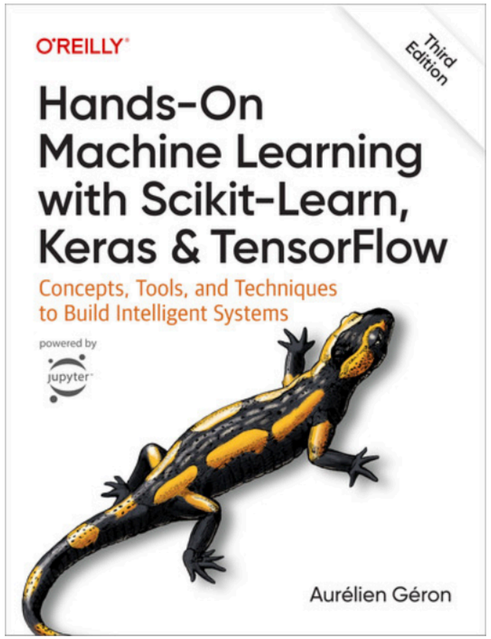
Reference #2: MLbook2

Available on https://www.oreilly.com/ or app with SJSU log in

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition

★★★★★ 16 reviews

By Aurélien Géron



TIME TO COMPLETE:
25h 32m

TOPICS:
Machine Learning

PUBLISHED BY:
O'Reilly Media, Inc.

PUBLICATION DATE:
October 2022

PRINT LENGTH:
861 pages

- GitHub repository
- Intermediate/advanced level
- Chapters: 3, 4

K-Means

Documentation

- https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html
- Color Quantization using K-Means: https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html#sphx-glr-auto-examples-cluster-plot-color-quantization-py
- 7.4. Loading other datasets

Machine Learning Project

ML Project Roadmap (1)

1. Understand the Problem, Set Up Your Environment, Download the Data
2. Data Preprocessing
    1. Data Cleaning: Handle missing values, remove duplicates, and correct errors.
    2. Feature Engineering: Create new features that might be useful for prediction based on domain knowledge and initial data analysis.
3. Data Transformation: Normalize or standardize data if necessary, encode categorical variables, etc.
4. Exploratory Data Analysis (EDA)
    1. Statistical Analysis: Get a statistical summary of the variables.
    2. Visualization: Use plots (histograms, box plots, scatter plots) to understand distributions and relationships between features.
    3. Correlation Analysis: Determine the relationships between features and the target variable.
5. Model Selection
    1. Choose Models: Based on the problem type, select appropriate algorithms (e.g., linear regression, random forests, gradient boosting machines, neural networks).
    2. Train Models: Split the data into training and validation sets to train and tune your models.
    3. Model Evaluation: Use cross-validation and the competition's evaluation metric to assess performance.
6. Model Tuning
    1. Hyperparameter Tuning: Use techniques like grid search or random search to find the optimal model settings.
    2. Feature Selection: Identify the most important features and remove irrelevant or less important features to improve model performance.
7. Ensemble Methods
    1. Combine Models: Use techniques like stacking, blending, or bagging to combine the predictions of several models to improve accuracy.

ML Project Roadmap (2)

8. Finalize the model
    1. Retrain on full dataset: Once you have your best model configuration, retrain it on the entire training dataset to make full use of the data.
    2. Validation: Validate the final model using a hold-out sample or via cross-validation to ensure robustness.
9. Submit prepared results to Kaggle
10. Reiterate
    1. Review based on leaderboard performance and feedback, refine your models and preprocessing steps
    2. Experiment: Try different models, features, and tuning parameters to improve your score.
11. Document along the way: approaches, experiments, and results

Exploratory Data Analysis (EDA)

- Histograms: Visualize distributions of numerical variables with histograms.
- Box Plots: Use box plots to see distributions with respect to categories and identify outliers.
- Scatter Plots: Plot relationships between numerical variables using scatter plots.
- Pair Plots: Use Seaborn’s `sns.pairplot()` for pairwise relationships between features.
- Heatmaps: Create correlation matrices and visualize them with heatmaps to check how variables correlate with each other.

Transform data

- combining data from different sources (N/A)
- **data cleaning**
- data types
- parsing dates
- file encodings
- **missing data**
- **duplicate data**
- dummy variables
- **remove outliers**
- scaling features
- engineering features

Feature engineering (1)

1. Imputation: Filling in missing values in the dataset. This can be done using various strategies such as filling with the mean, median, mode, or using more complex methods like regression or k-nearest neighbors to estimate the missing values.
2. Categorical Encoding: Converting categorical variables into numerical formats that can be understood by machine learning algorithms. Common methods include:
    1. One-hot encoding: Creating a new binary column for each category.
    2. Label encoding: Assigning each category a unique integer.
3. Normalization and Standardization:
    1. Normalization (Min-Max scaling): Scaling the feature to a fixed range, typically 0 to 1.
    2. Standardization (Z-score normalization): Scaling the feature so that it has a mean of 0 and a standard deviation of 1. This is useful for algorithms that assume data is normally distributed.

Feature Engineering (2)

4. Feature Creation:
    1. Interaction terms: Creating new features by combining two or more existing features, which can help in capturing interactions between variables that the original features might miss.
    2. Polynomial features: Generating polynomial and interaction features by raising existing features to various powers and creating interaction terms.
5. Dimensionality Reduction:
    1. Principal Component Analysis (PCA): Reducing the number of variables in data by selecting new components that capture the maximum variance in the data.
    2. t-Distributed Stochastic Neighbor Embedding (t-SNE): Reducing dimensions while trying to keep similar instances close and dissimilar instances apart, often used for visualization.
    3. Autoencoders: Using neural networks to learn a dense representation of the input features.
6. Binning or Bucketing: Converting continuous variables into categorical variables by dividing the range of the variable into fixed number of bins or intervals. This can help handle outliers and nonlinear effects.

Model Selection for Regression

1. **Linear Regression Models**

- Linear Regression: The simplest form of regression, linear regression fits a straight line to data.

- Ridge Regression: Extends linear regression by adding a regularization penalty to the loss function, which helps to reduce model complexity and prevent overfitting.

- Lasso Regression: Similar to Ridge but uses L1 regularization which can lead to sparse models where some coefficients can become exactly zero.

- Elastic Net: Combines L1 and L2 regularization terms, which can be beneficial when dealing with highly correlated data.

2. **Support Vector Machines**

- SVR (Support Vector Regression): Applies the principles of Support Vector Machines (SVM) to regression problems, focusing on fitting the error within a certain threshold.

3. **Stochastic Gradient Descent**

- SGDRegressor: Implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for regression, allowing it to be highly customizable.

4. **Nearest Neighbor Regressors**

- KNeighborsRegressor: Regression based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated with the nearest neighbors.

5. **Decision Tree Regressors**

- Decision Tree Regressor: A non-parametric method that builds regression trees. Each leaf of the tree represents a value predicted in that region.

- Extra Trees Regressor: Similar to random forests, providing an ensemble of unpruned decision trees, but created from random subsets of the features.

- Random Forest Regressor: A meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Prediction

- Train the model:
  - Split the training set into: training and test
  - Inference using the model