

UNIT 3



ALGORITHMS, TRAINING & TESTING DATA | FEATURES | MODEL, IMPORTANT PYTHON LIBRARIES FOR MACHINE LEARNING AND INSTALLATION STEPS

1. Algorithms

- An algorithm is a set of rules or instructions given to an AI system to help it learn from data.

Example: Decision Tree is an algorithm used for classification and regression task.

$$2 * 2 = 4$$

$$4 * 2 = 8$$

$$6 * 2 = 12$$

$$8 * 2 = 16$$

$$x * 2 = x + x \quad (\text{rule})$$

$$2x = x + x \quad (\text{rule})$$

$$3x = x + x + x \quad (\text{rule})$$

$$2x, 3x \quad (\text{algorithm})$$

$$2x + 3y - 4y = z \quad (\text{algorithm})$$

x	y	=	z
$2(x) + 3$	$- 4(y)$	$=$	z

x ₁	x ₂	x ₃	=	y
Hair (cm)	Height	Bia	=	Gender
24	160	Yes	=	Female
36	165	Yes	=	Female
48	170	Yes	=	Female
8	175	No	=	Male

y is categorical variable (classification)

2. Training Data, Testing Data, Features and Model

- Training Data:** The dataset used to train the machine learning model. It's labeled data for supervised learning.

Example: A set of images of cat and dog, each labeled with "cat" and "dog".

- Testing Data:** Data used to evaluate the performance of a model after training. It's unseen by the model during training.

Example: A new set of images not included in the training data, used to check the accuracy of a trained model.

- **Features:** Individual measurable properties or characteristics of a phenomenon being observed, used as input variables in a model.
Input (features like x_1, x_2, x_3) and output (labeled like y)
Example: In a dataset for *house price prediction (labeled)*, features might include *square footage, number of bedrooms, and age of the house (features)*.
- **Model:** In machine learning, a model refers to the specific representation (like $2x + 3y - 4y = z$) learned from data, based on which predictions or decisions are made.
Example: A neural networks trained to identify objects in an image.

3. Overfitting vs Underfitting

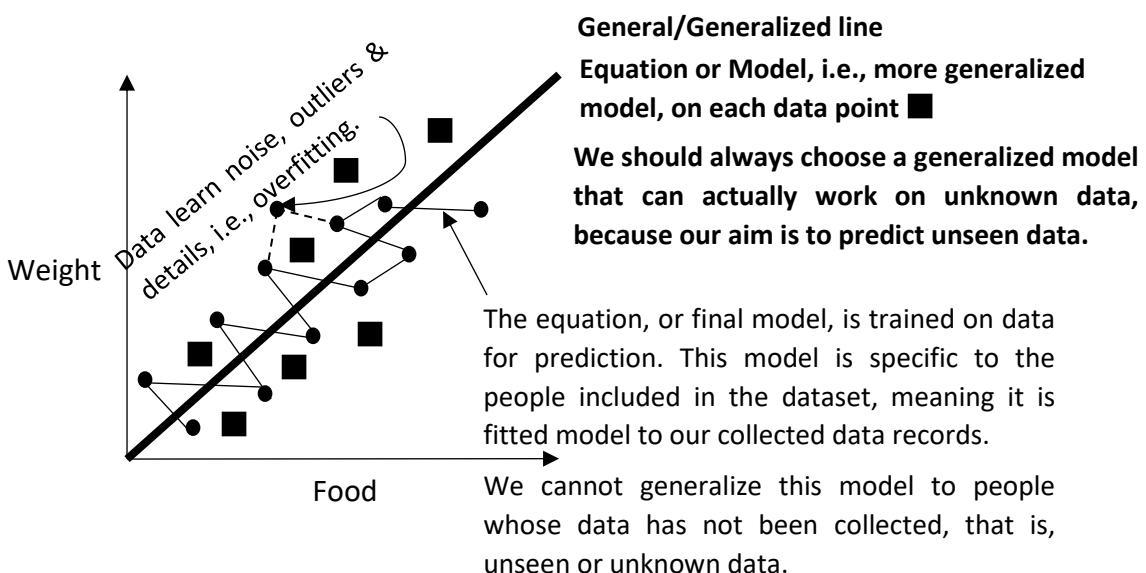


Figure 3.1 Overfitting vs Underfitting.

- **Overfitting:** A modeling error, that occurs when a model is too complex and learns the noise and details in training data to the extent that it negatively impacts the performance of the model on new data.
Example: A model that performs exceptionally well on the training data but poorly on the testing data.
- **Underfitting:** This occurs when a model is too simple, unable to capture the underlying pattern of the data, and therefore performs poorly on both training and new data.
Example: A liner regression model trying to fit non-liner data.

4. Important Python libraries for Machine Learning

1. [scikit-learn library](#)
2. [TensorFlow library](#)
3. [Keras library](#)

4. [PyTorch library](#)
5. [NLTK :: Natural Language Toolkit library](#)
6. [openCV library](#)
7. Many more...
 - All models are available within libraries, along with their user guides, documentation, code, and definitions. The scikit-learn library is widely used for data preprocessing. Feature extraction is also provided by the scikit-learn library. Classification, regression, clustering, dimensionality reduction, model selection, and preprocessing all come under the scikit-learn library.

5. Installation steps for machine learning environment

5.1. Bash Terminal Commands

Create Conda Environment

```
# Syntax for creating a new conda environment with Python 3.11
# and naming it "bash_terminal"
conda create -n bash_terminal python=3.11
```

Activate Environment

```
conda activate bash_terminal
```

Install Libraries

```
conda install numpy pandas matplotlib seaborn scikit-learn
tensorflow pytorch scikit-learn jupyter
```

Import Libraries in Python File

```
# Core scientific libraries
import numpy as np
import pandas as pd

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning
```

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Deep Learning
import tensorflow as tf
import torch

# Jupyter (only needed in notebooks)
from IPython.display import display
```

Deactivate Current Environment

```
conda deactivate
```

Remove Environment

```
conda remove -n bash_terminal --all
```

5.2. Jupyter Notebook Cell

Create Conda Environment

```
# Cell: Create environment
!conda create -n jupyter_notebook python=3.11 -y
```

Activate Environment

```
# You can't truly activate in a cell, but you can use conda run
# Or install as a kernel (recommended)
```

```
# Install ipykernel in the environment
!conda install -n jupyter_notebook ipykernel -y
```

```
# Register environment as Jupyter kernel
!python -m ipykernel install --user --name=jupyter_notebook --
display-name="Python (jupyter_notebook)"
```

```
# Then: Go to Kernel → Change Kernel → Python  
(jupyter_notebook) in Jupyter menu
```

Install Libraries

```
# After changing kernel, install directly  
!conda install numpy pandas matplotlib seaborn scikit-learn  
tensorflow pytorch scikit-learn jupyter -y
```

Import Libraries in Python File

```
# Core scientific libraries  
import numpy as np  
import pandas as pd  
  
# Visualization  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Machine Learning  
import sklearn  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import accuracy_score  
  
# Deep Learning  
import tensorflow as tf  
import torch  
  
# Jupyter (only needed in notebooks)  
from IPython.display import display
```

Deactivate Current Environment

```
# Switch back to base kernel from menu: Kernel → Change Kernel  
→ Python 3
```

```
# Note: In cells, you don't deactivate - you just change kernel
```

Remove Environment

```
# First, switch to a different kernel (not Jupyter_notebook)
# Then run:
!conda env remove -n jupyter_notebook -y

# Also remove the kernel
!jupyter kernelspec uninstall jupyter_notebook -y
```

5.3. Google Colab Cell

Colab doesn't natively support conda, but you can install it.

Create Conda Environment

```
# Install Conda First
# Cell 1: Install condacolab
!pip install -q condacolab
import condacolab
condacolab.install()
```

Activate Environment

```
# Run something inside your conda environment
!conda run -n colab_notebook python --version

# Activation does NOT persist
```

Install Libraries

```
# After changing kernel, install directly
!conda install numpy pandas matplotlib seaborn scikit-learn
tensorflow pytorch scikit-learn jupyter -y
```

Import Libraries in Python File

```
# Core scientific libraries
import numpy as np
import pandas as pd
```

```

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Deep Learning
import tensorflow as tf
import torch

# Jupyter (only needed in notebooks)
from IPython.display import display

```

Deactivate Current Environment

```

# Or just restart runtime: Runtime → Restart runtime
# There is no proper activate/deactivate workflow in Colab,
because each !command runs in a separate shell session.
# !command runs in a temporary shell
# Activation does NOT persist
# There is nothing to deactivate

```

Remove Environment

```

# Step 1: Restart Runtime
# Runtime → Restart runtime (Restart Session)

```

```

# Step 2: Then remove the environment
!conda env remove -n colab_notebook -y

```

```

# Verify removal
!conda env list

```

5.4. Kaggle Notebook Cell

 **Kaggle does NOT support conda environments at all.**

Kaggle comes with pre-installed packages. You can only use pip for additional packages.

Check Available Packages

```
# See what's already installed  
!pip list
```

Install Additional Libraries

```
# Install using pip only  
# !pip install some-package-name  
!pip install numpy pandas matplotlib seaborn scikit-learn  
tensorflow pytorch scikit-learn jupyter
```