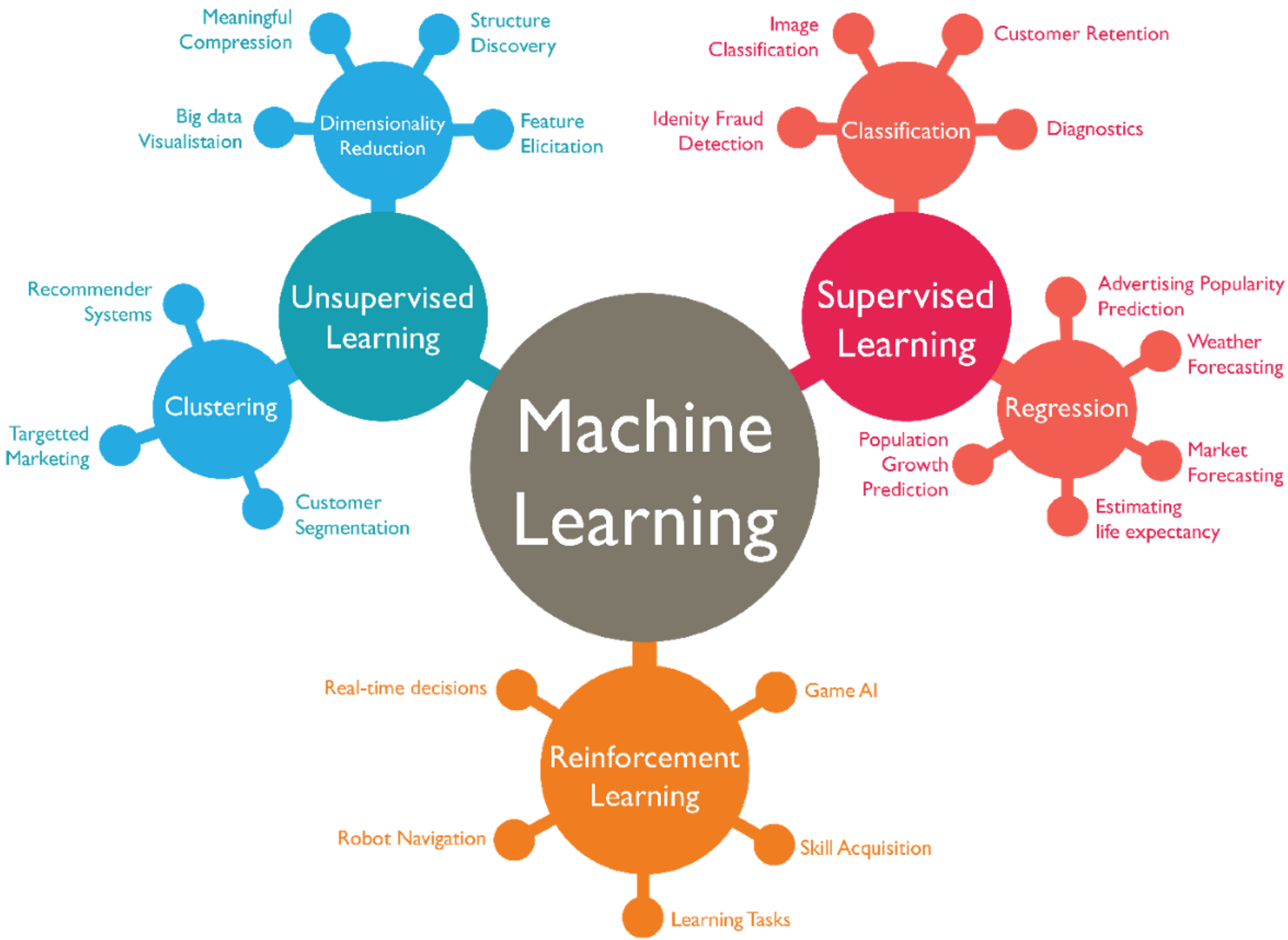
SME3006 Machine Learning – 2025 Fall

ML Basics and Data preprocessing

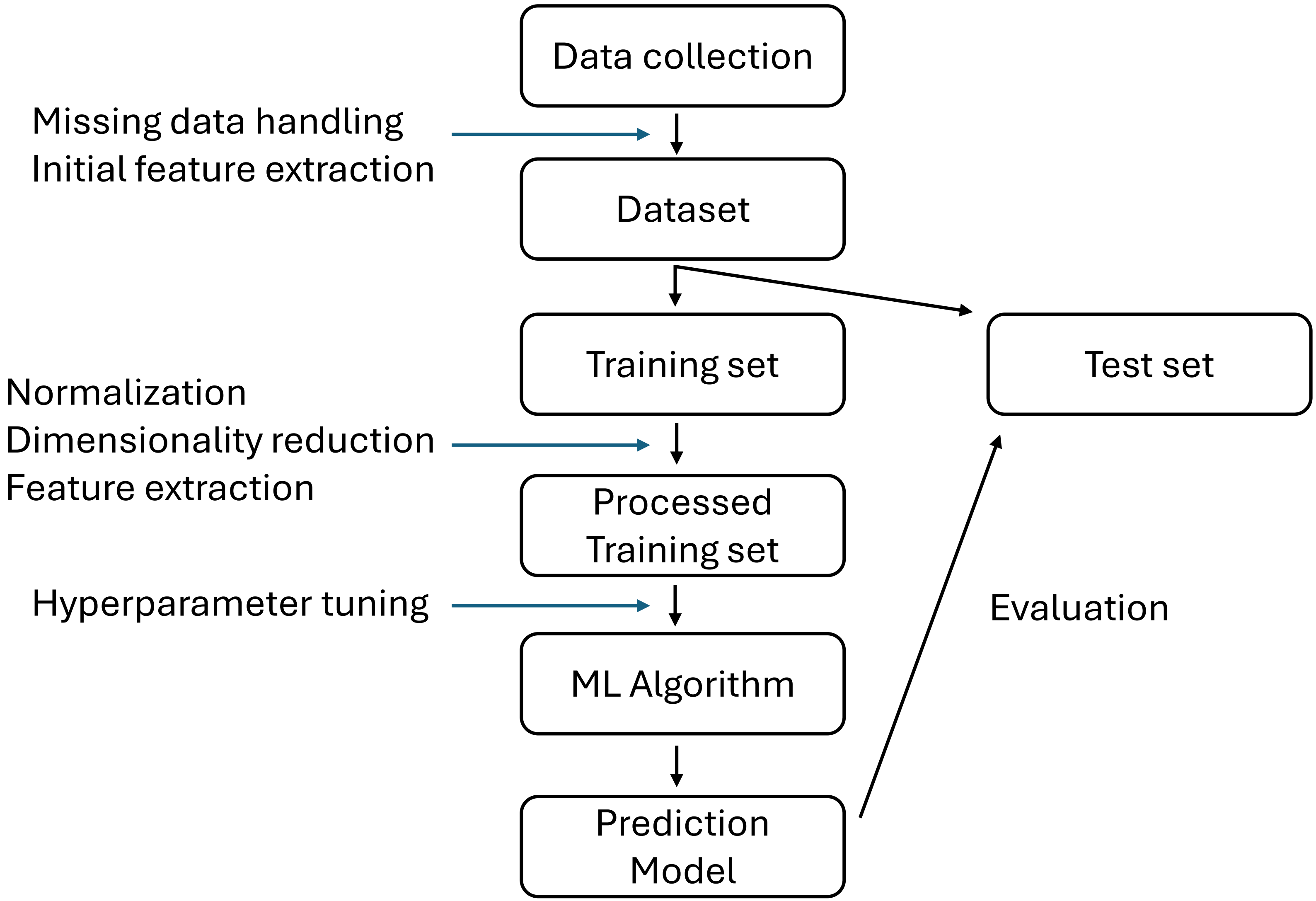
INHA UNIVERSITY

Types of ML



Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Machine learning system roadmap

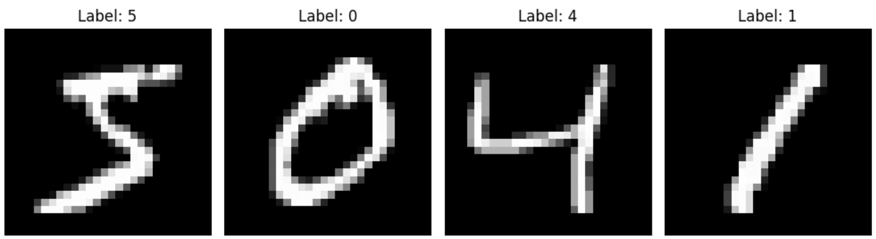


Terminology

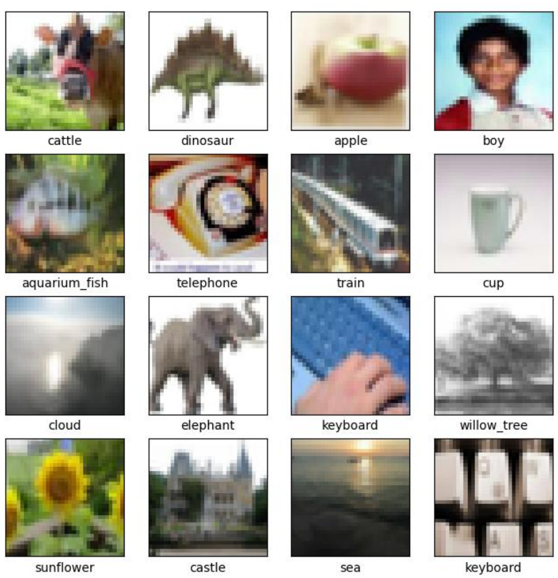
- Dataset
  - Instance, observation
  - Data sample, sample space
  - Feature, feature vector
- Learning, training
  - Training data, training sample, training set
  - Validation set, test set
  - Ground-truth
  - Label, label space
- Model
  - Inputs, input vectors
  - Outputs, targets

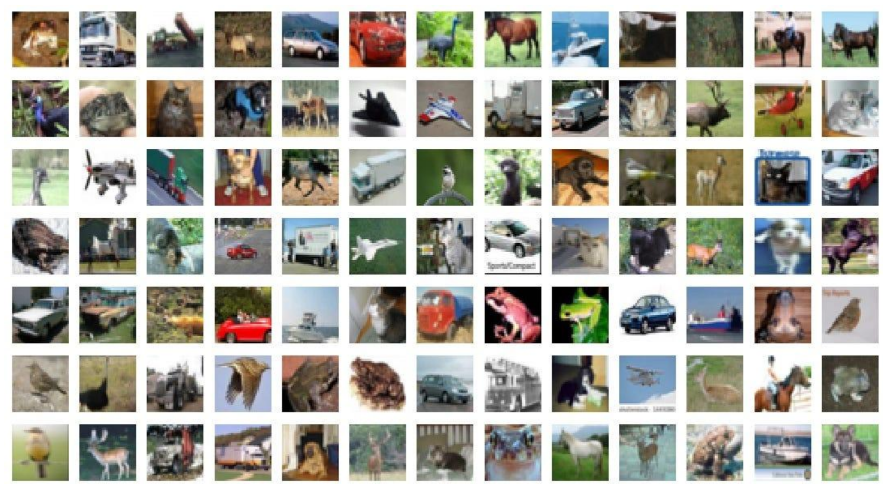
Terminology

> MNIST dataset



> CIFAR-10/100 dataset



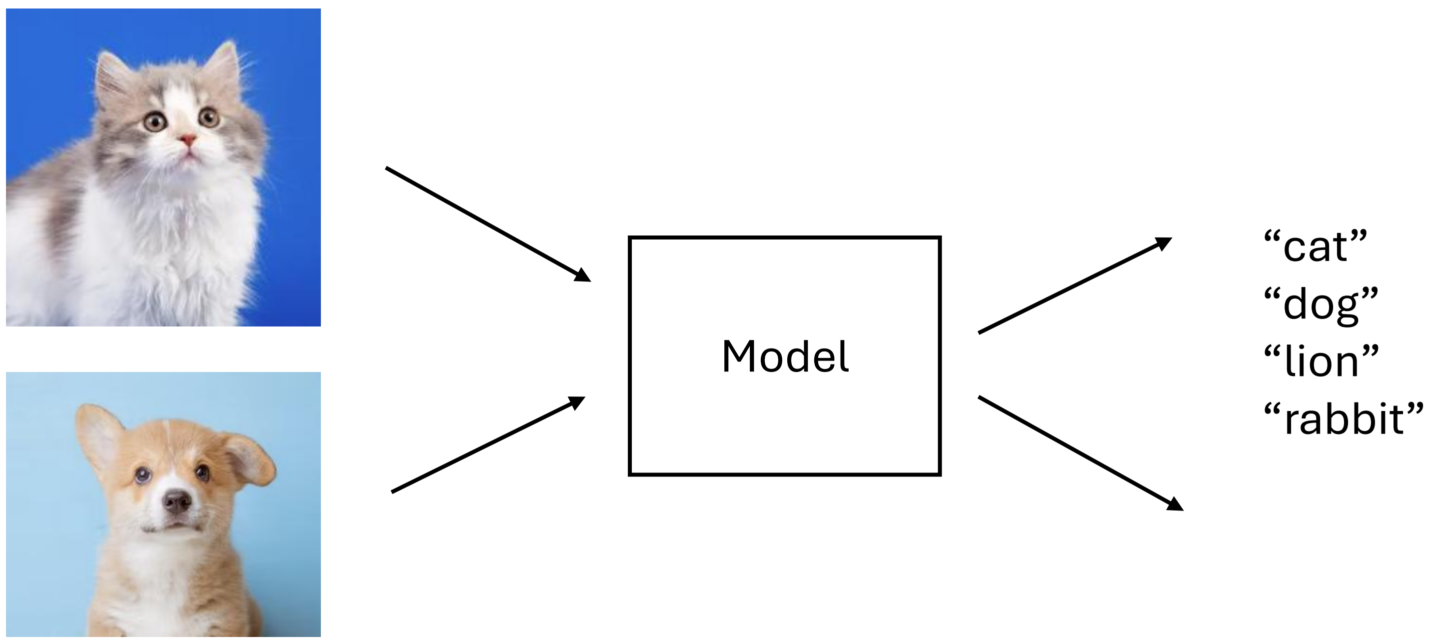


Terminology

- Sample space
  - coin tossing: head or tail (0 or 1)
  - MNIST: $256^{28\times28}$ (gray)
  - CIFAR10: $256^{32\times32\times3}$ (RGB)

- Label space
  - MNIST: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - CIFAR10: {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}

Terminology

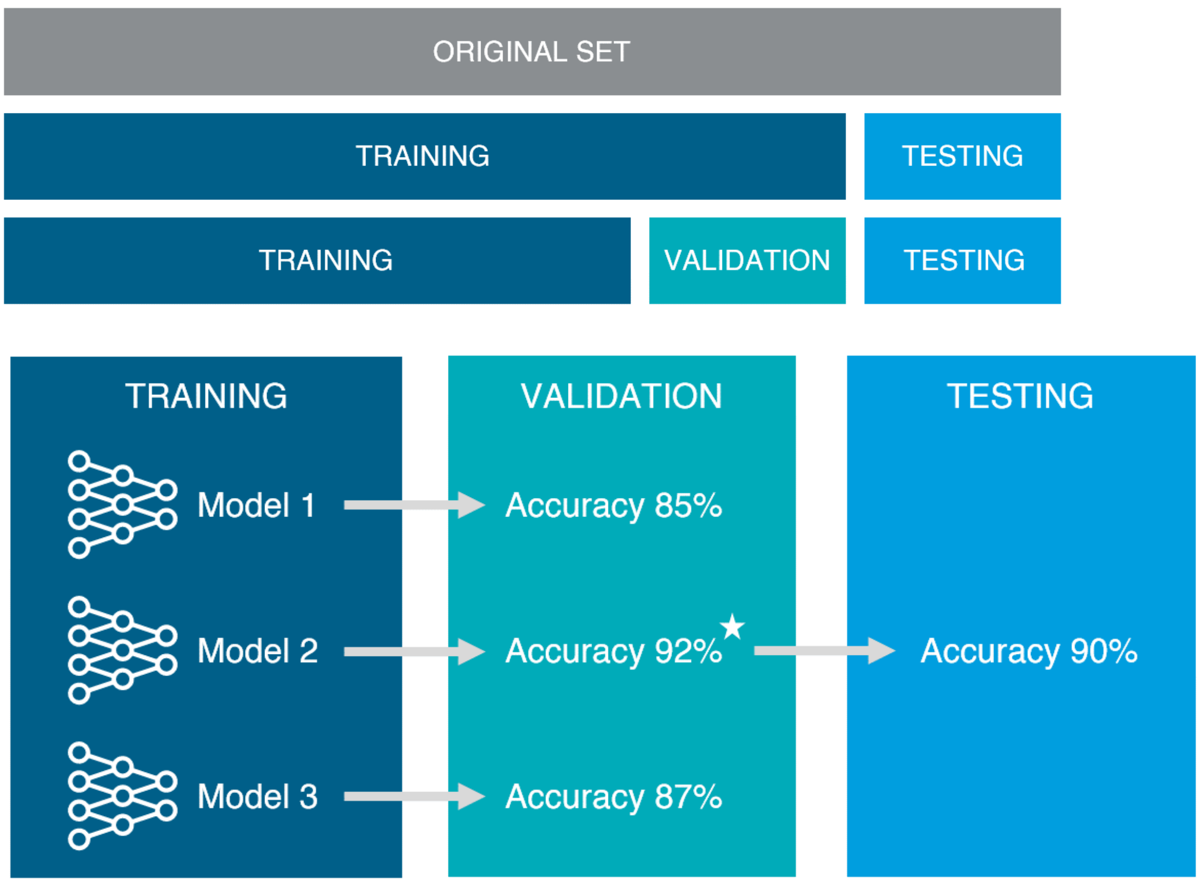


Instance, observation

Label

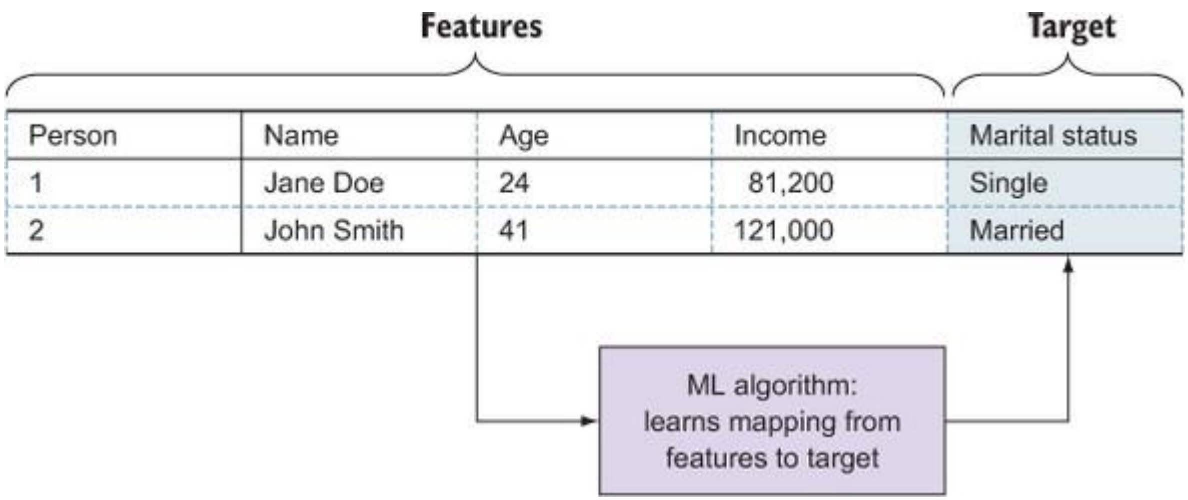
Terminology

> Dataset splitting: training / validation / testing

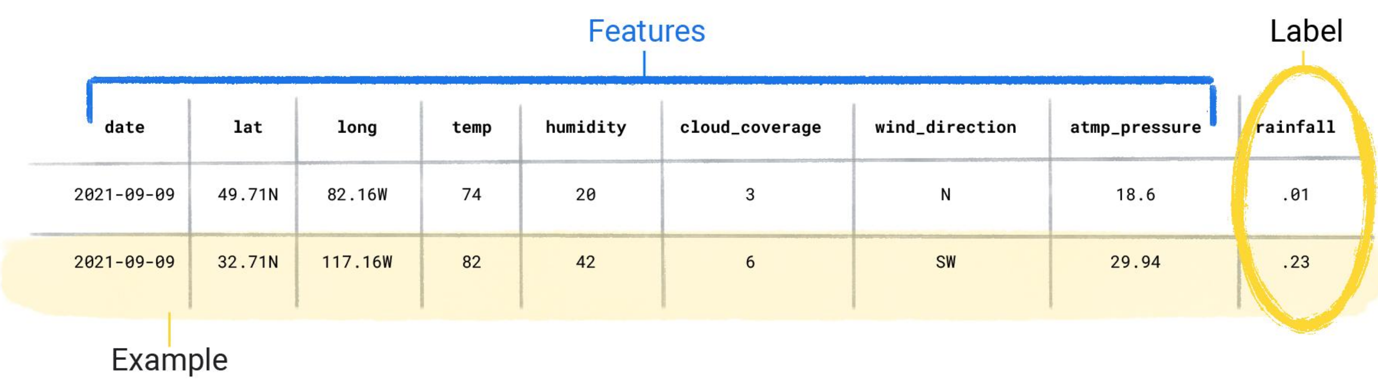


Terminology

> Features



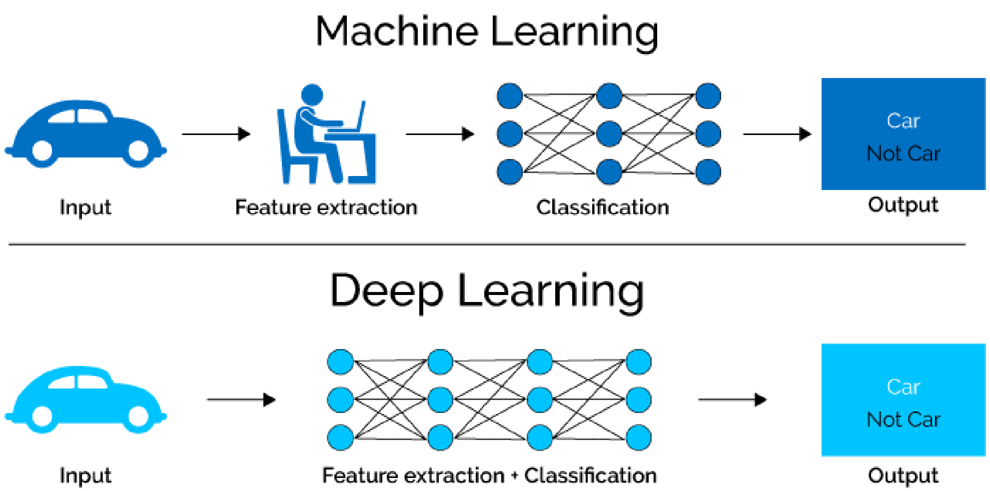
Person	Name	Age	Income	Marital status
1	Jane Doe	24	81,200	Single
2	John Smith	41	121,000	Married

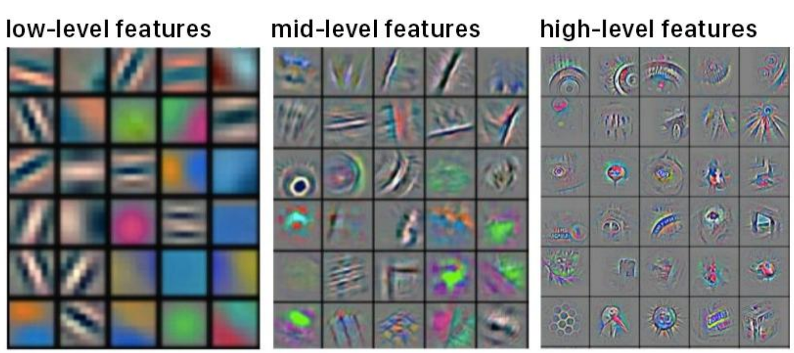


date	lat	long	temp	humidity	cloud_coverage	wind_direction	atmp_pressure	rainfall
2021-09-09	49.71N	82.16W	74	20	3	N	18.6	.01
2021-09-09	32.71N	117.16W	82	42	6	SW	29.94	.23

Terminology

- > Features
  - nose shape, whisker, pupils, ear shape, muzzle length



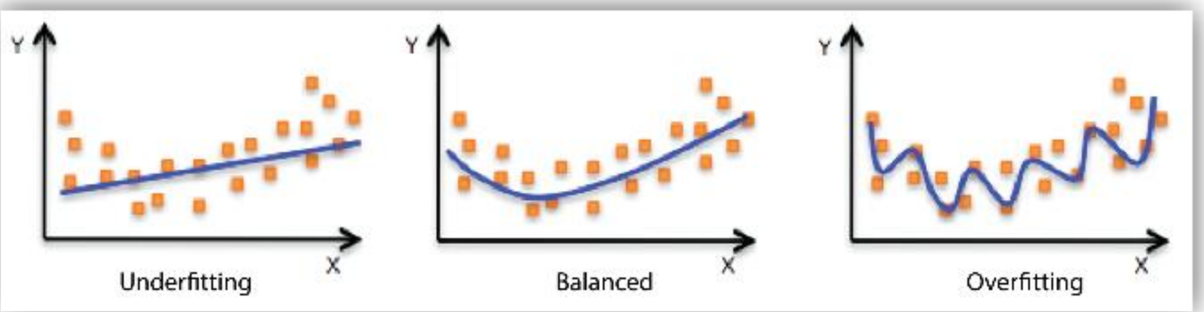


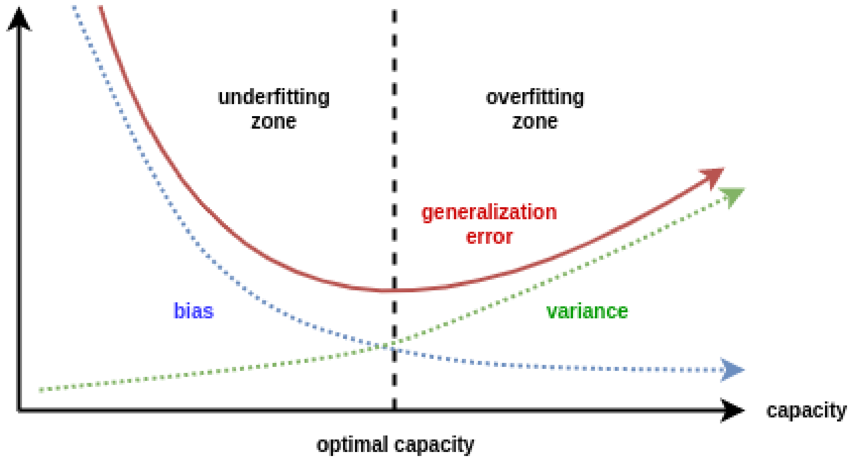
Terminology

> Generalization
  - Over-training, overfitting
  - Underfitting
  - Interpolation, extrapolation
  - Uncertainty
  - Inductive bias

Terminology

> Generalization





Terminology

> Uncertainty

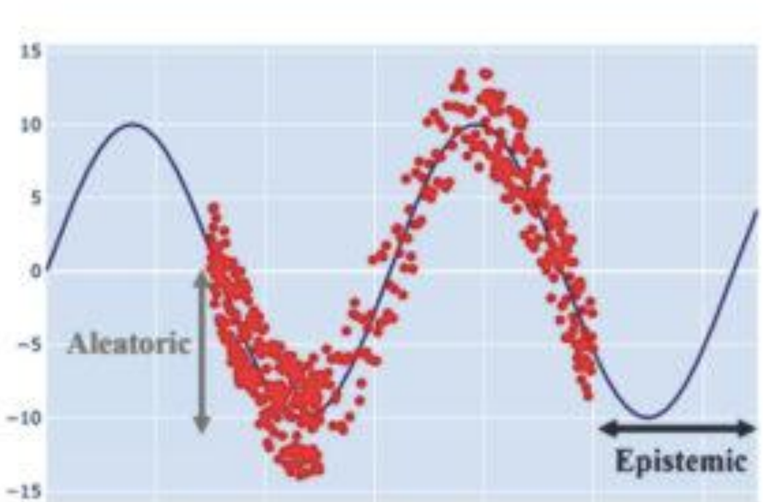
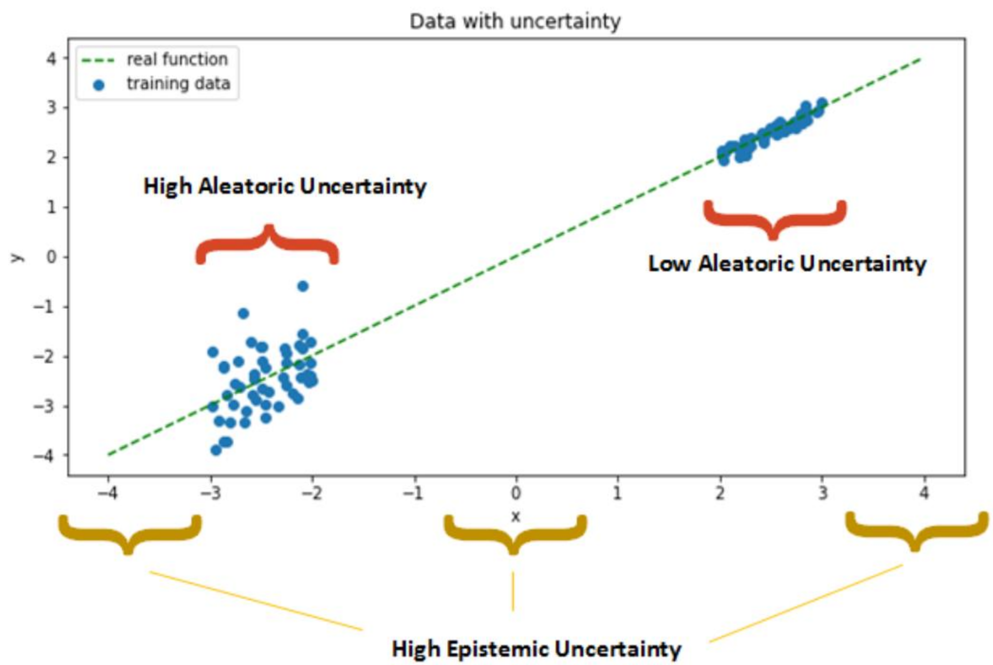
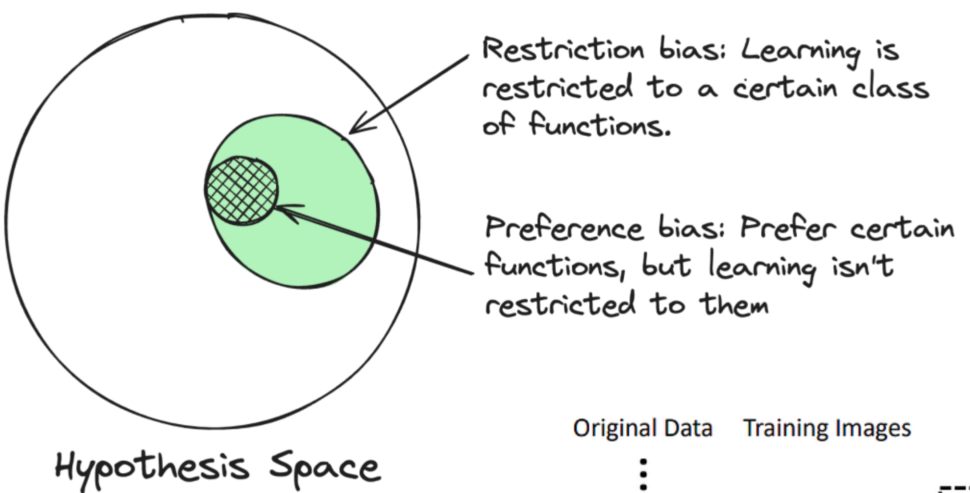


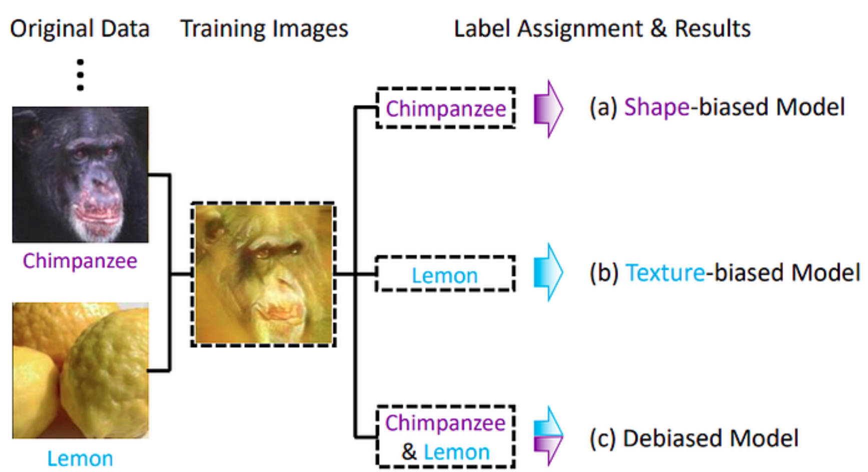
Fig. 1: A schematic view of main differences between aleatoric and epistemic uncertainties.



Terminology

> Inductive bias
  - set of assumptions a learning algorithm makes in order to generalize from limited data to unseen cases



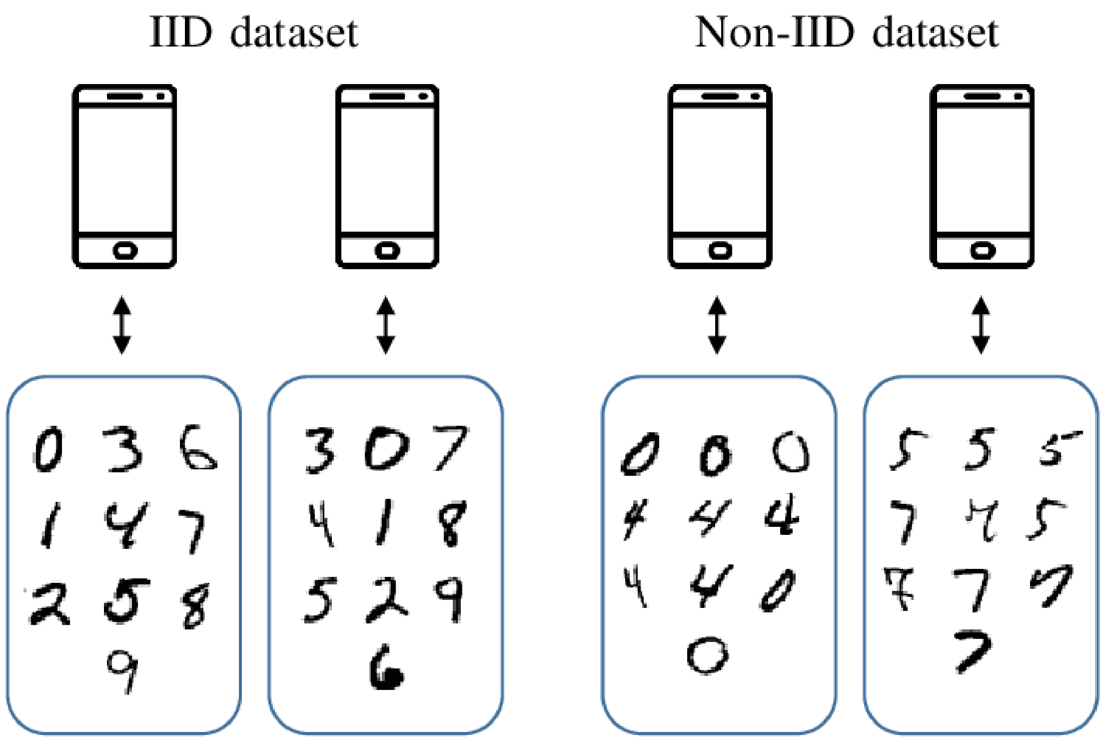


Terminology

- Distribution
  - Independent and Identically distributed (i.i.d)
  - Distribution shift, covariate shift
  - Out-of-distribution (OOD)
- Accuracy
  - Sensitivity, specificity, precision, recall, F1
  - Mean square error(MSE), mean average error(MAE)
  - Bias-variance tradeoff

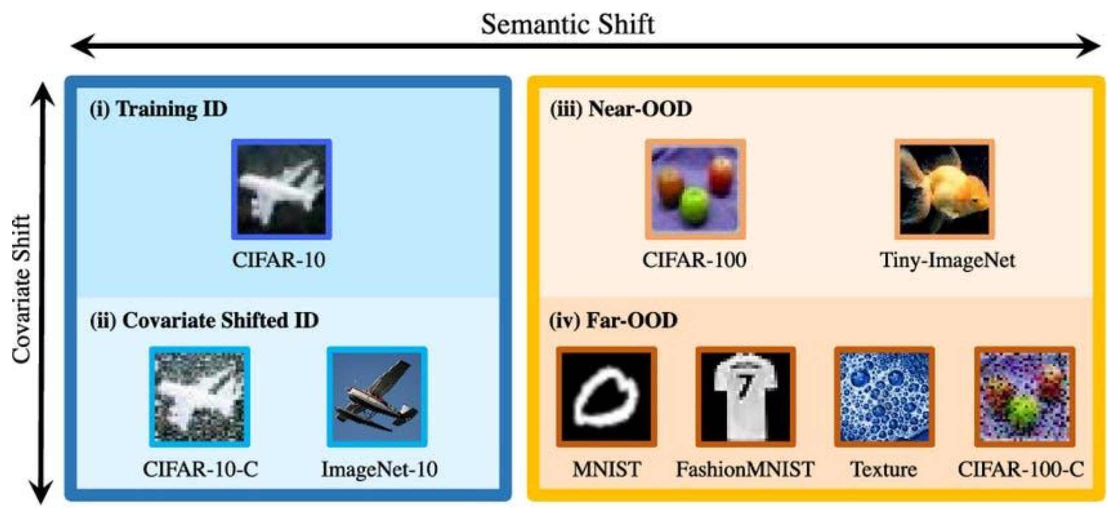
Terminology

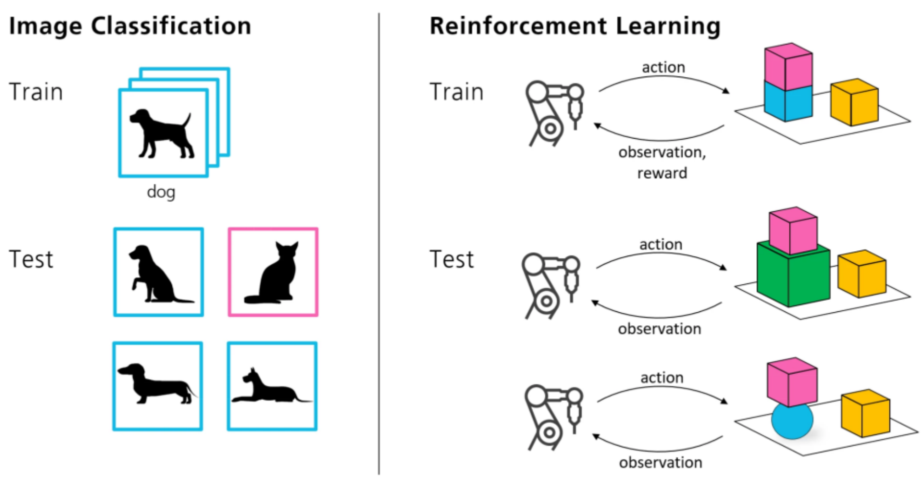
> Independent and Identically distributed (i.i.d)



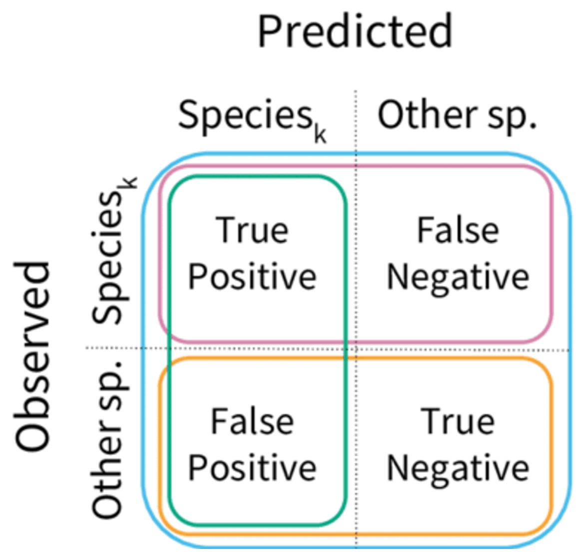
Terminology

> Distribution shift, covariate shift, out-of-distribution





Terminology

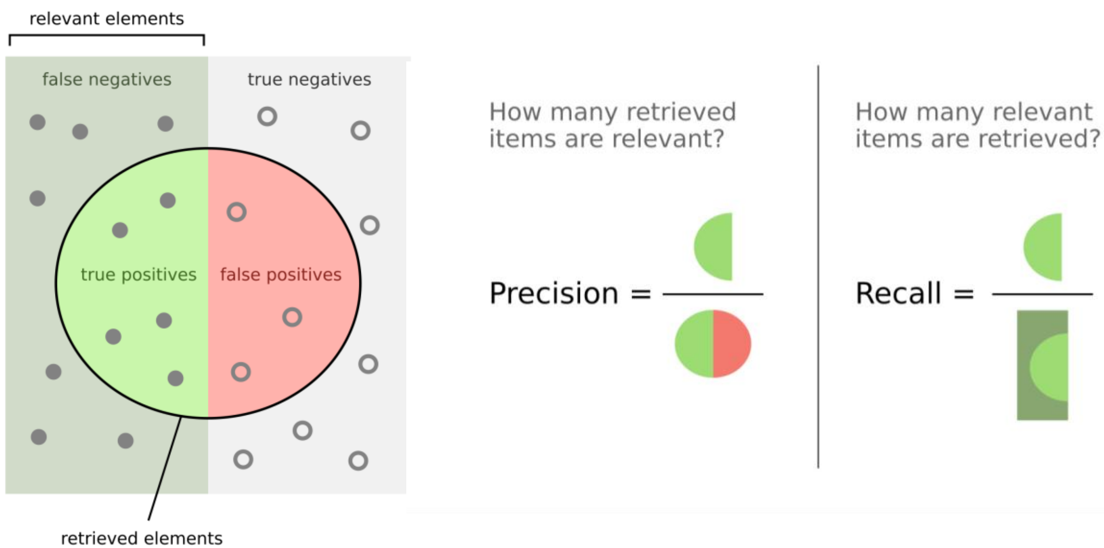


$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$



$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

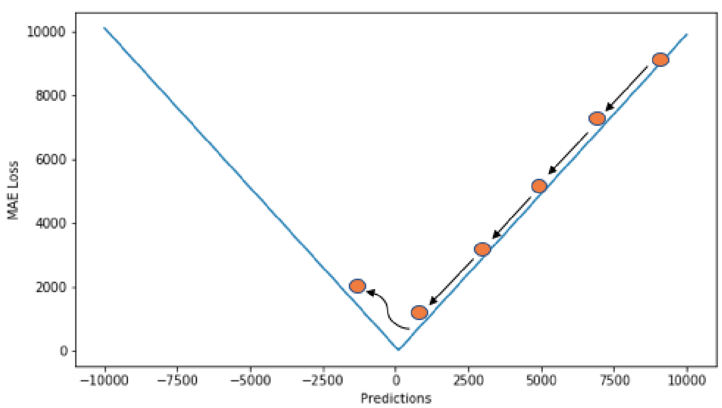
Terminology

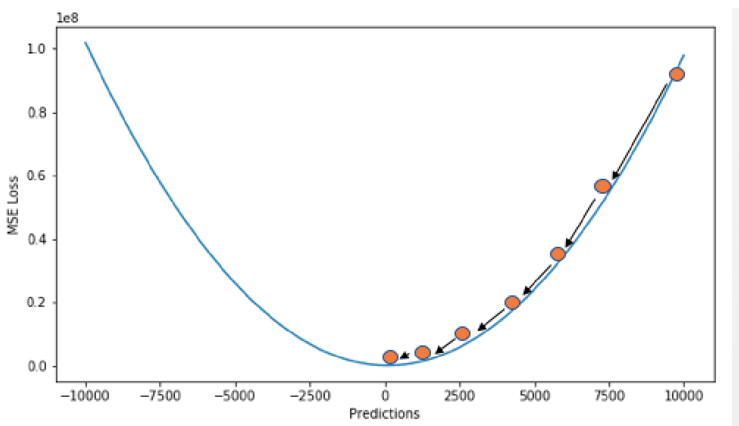
Epoch	Prediction	Target
1	[0, 4, 9]	[3, 5, 7]
2	[2, 4, **2**]	[3, 5, **7**]
3	[3, 5, 6]	[3, 5, 7]

$$MSE = \frac{1}{N}\sum_{i}^{N}(pred_i - target_i)^2$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(pred_i - target_i)^2}$$

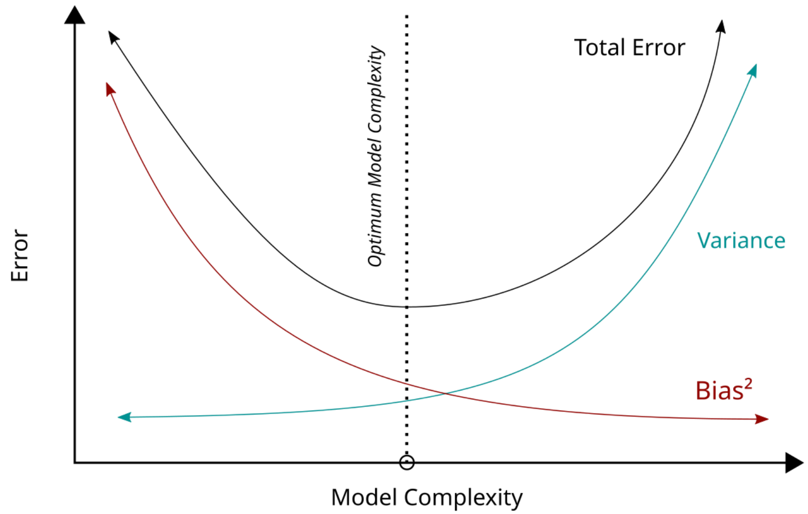
$$MAE = \frac{1}{N}\sum_{i}^{N}|(pred_i - target_i)|$$

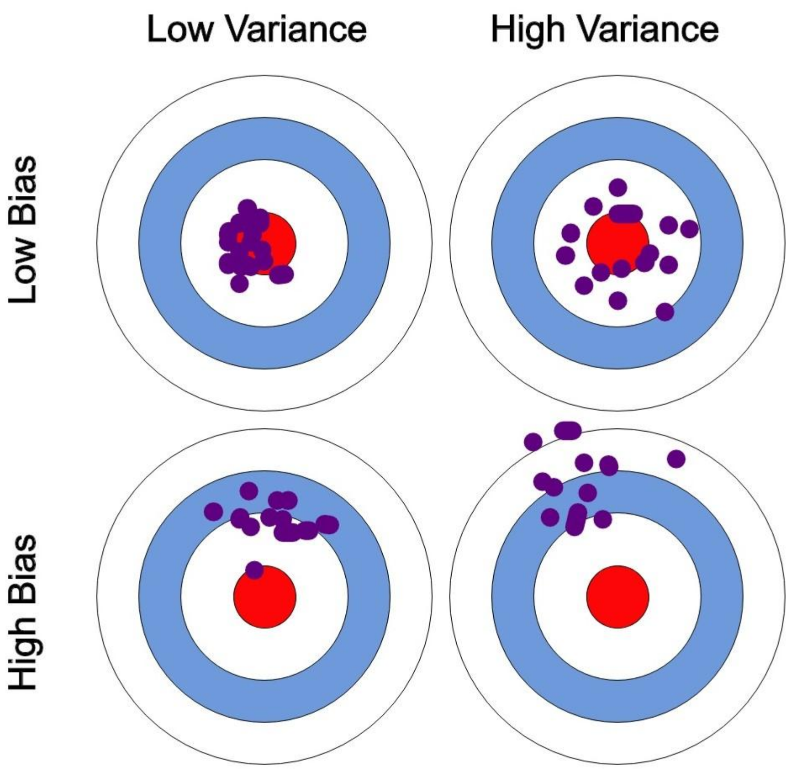




Terminology

> Bias-variance tradeoff

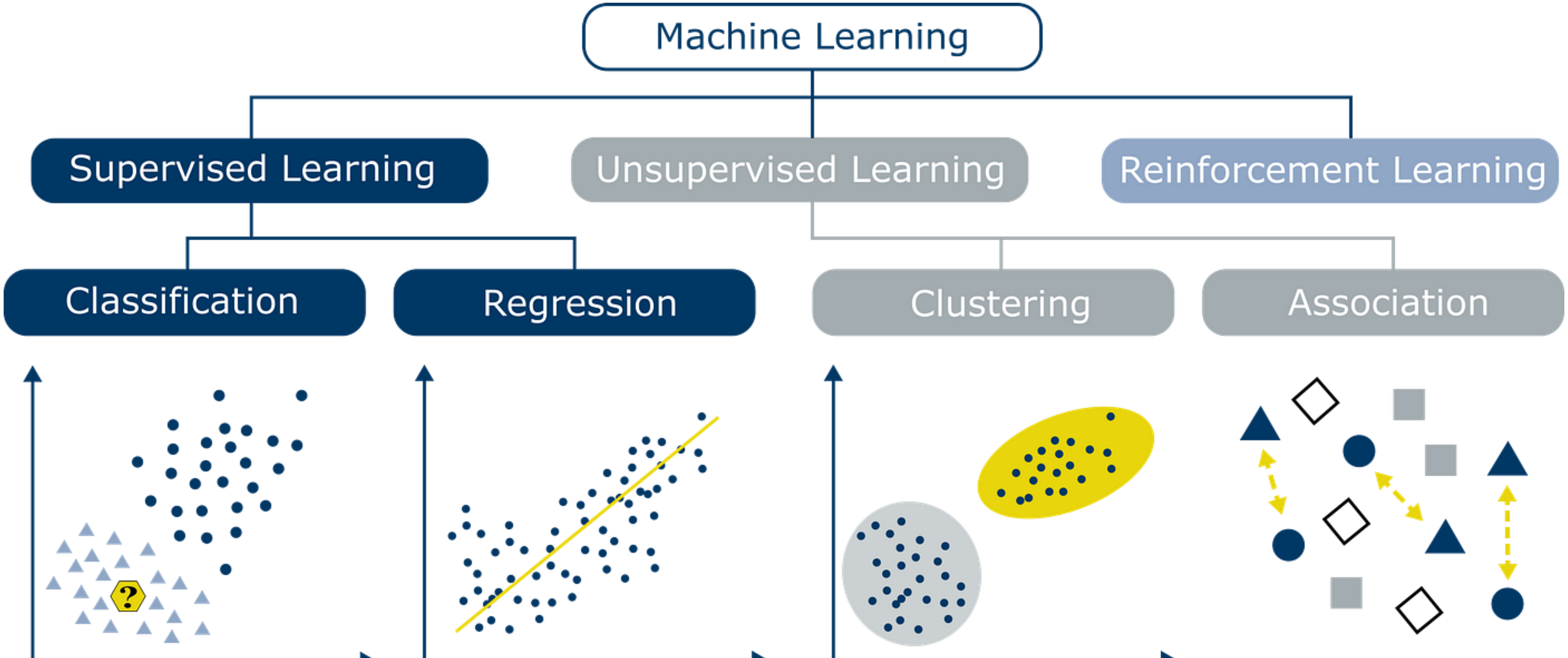




Terminology

- Task
  - Classification
  - Regression
  - Clustering
  - Dimensionality reduction
  - Generative task

Terminology



General principles in ML

> No Free Lunch Theorem (NFL)
  - There is no universally best learning algorithm for all problems

> Curse of Dimensionality
  - As the number of dimensions increases, the volume of the space grows exponentially, making the data sparse

> Occam's Razor
  - Among competing hypotheses, the one with the fewest assumptions should be selected.

> Bias-Variance Tradeoff
  - Balancing simplicity and flexibility is essential for generalization

Python packages

> Numpy
  - numerical python.
  - array and matrix operations, linear algebra, random number generation

> Pandas
  - handling structured data
  - reading CSVs, data cleaning, filtering

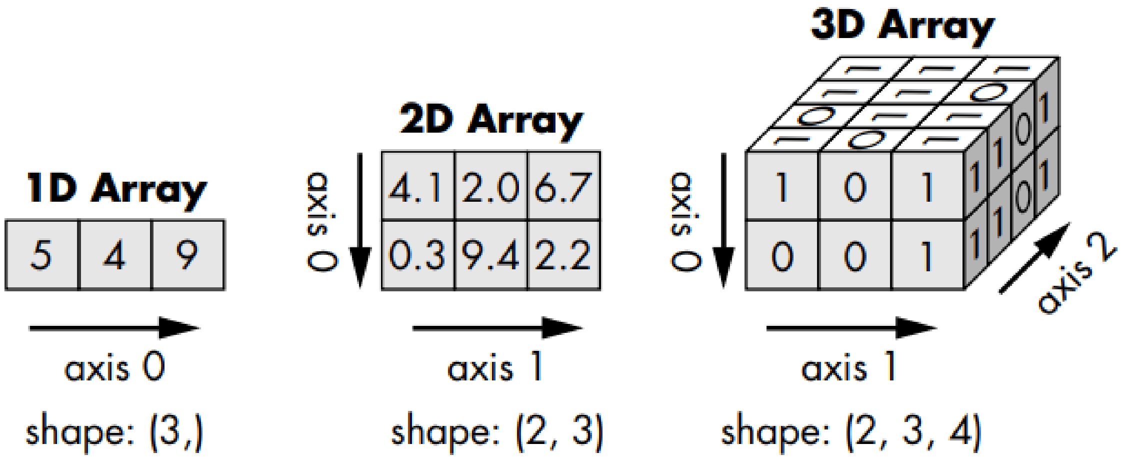
> Matplotlib
  - basic plotting library

> Sklearn (Scikit-learn)
  - a machine learning library
  - classification, regression, clustering, preprocessing, model evaluation

> Seaborn, Scipy, TensorFlow, PyTorch, MLflow, Optuna, …

Numpy basics

```
import numpy as np

array1 = np.array([5,4,9])
array2 = np.array([[4.1,2.0,6.7], [0.3,9.4,2.2]])
array3 = np.array([[[1,0, 1],[0, 0, 1]], [[0, 1, 1], [...]], ....  ]]])

print(array1.shape)
print(array2.shape)
```



Numpy basics

- ndarray
  - int8, int16, int32, float16, float32, string, object...
  - but, has to be same format in one array

- list
  - multiple types can be combined

```
array1 = np.array([1, 2.0, 3.0])
array2 = np.array([1,2,3])

print(array1.dtype)
print(array2.dtype)

list1 = [1, 2, ‘hi’]

array1_2 = array1.astype(‘float64’)
```

Numpy basics

```
array = np.arange(10)
print(array)
-> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

array = np.zeros((2,3), dtype='float32')
print(array)
-> [[0. 0. 0.]
    [0. 0. 0.]]

array = np.ones((1,2), dtype ='int32')
print(array)
-> [[1 1]]
```

Numpy basics

```
array1 = np.arange(20)
array2 = array1.reshape(4,5)
array3 = array2.reshape(-1,2)
array4 = array1[2:5]
array5 = array2[[1,2], [2,3]]
array6 = array1[array1>7]


array = np.array([3, 1, 9, 5])
sorted = np.sort(array)
index = np.argsort(array)
```

Numpy basics

```
A = np.array([[1, 2, 3], [1, 1, 1], [3, 2, 1]])
B = np.array([[2, 1, 2,], [1, 2, 4], [2, 1, 0]])

A+B
A*B
A@B, np.matmul(A,B)
np.transpose(A), A.T

np.linalg.det(A)
np.linalg.inv(A)

np.append(A, [[3,4,5]], axis=0)
np.vstack([A, A])
np.hstack([A, A])
A.flatten()
B = np.expand_dims(A, axis = 0)
A = np.squeeze(B)

np.random.rand(2,3)
np.where(A>1, A, 0)
```

Pandas basics

https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html

Series: 1D labeled array holding data of any type
DataFrame: 2D data structure that holds data like a table

```
import pandas as pd

s = pd.Series([1, 3, 5, np.nan, 6, 8])
df = pd.DataFrame({"A": 1.0, "B": pd. Timestamp("20250702"), "C": np.array([2]*2),
"D": pd.Categorical(["train", "test"]}))

df.head(n) # view the top rows of the frame        # n is the number of rows
df.tail() # view the bottom rows of the frame
df.index # 0, 1
df.columns # A, B, C, D

df.to_numpy()
```

Pandas basics

```
dates = pd.date_range("20130101", periods=6)
df = pd.DataFrame(np.random.randn(6,4), index = dates, columns=list("ABCD"))

df["A"]
df.loc[dates[0]]
df.iloc[3]
df.iloc[3:5, 0:2]
```

Pandas basics

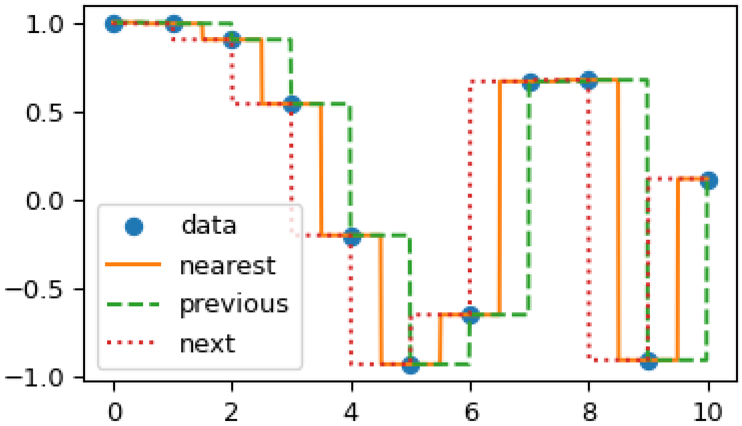
```
df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ["E"])
df1.loc[dates[0] : dates[1], "E"] = 1
df1

pd.isna(df1)

df1.dropna(how="any")
df.dropna(thresh=4)
df.dropna(subset=["C"])
```

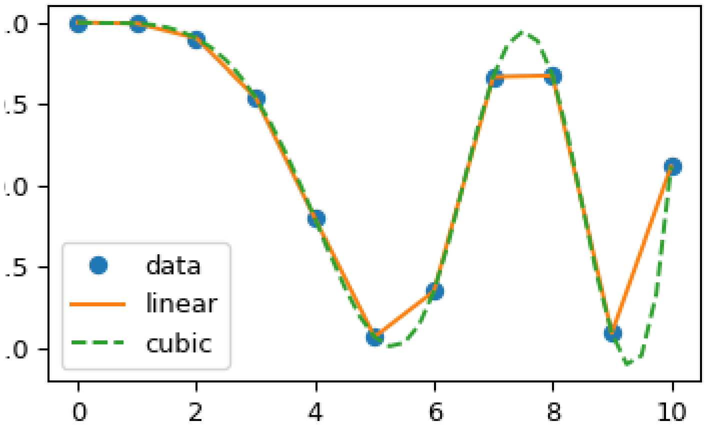
Working with missing data

```
from io import StringIO
csv_data = \
'''A,B,C,D
 1.0,2.0,3.0,4.0
 5.0, 6.0,,8.0
 10.0,11.0,12.0,'''
df = pd.read_csv(StringIO(csv_data))

df.isnull().sum()

df.fillna(5) # replace NA with a scalar value
df.ffill() # fill gaps forward
df.bfill() # fill gaps backward

df.fillna(df.mean()) #
```

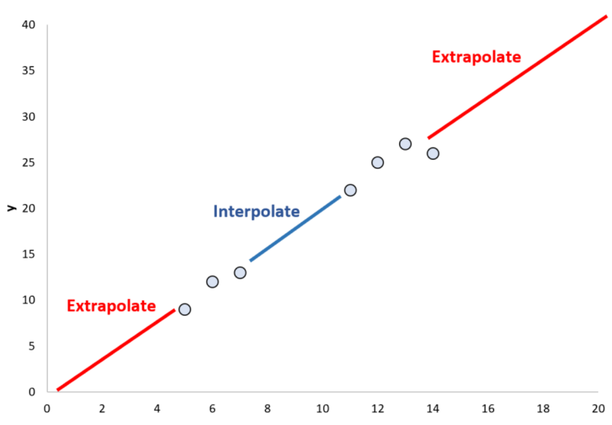
Working with missing data

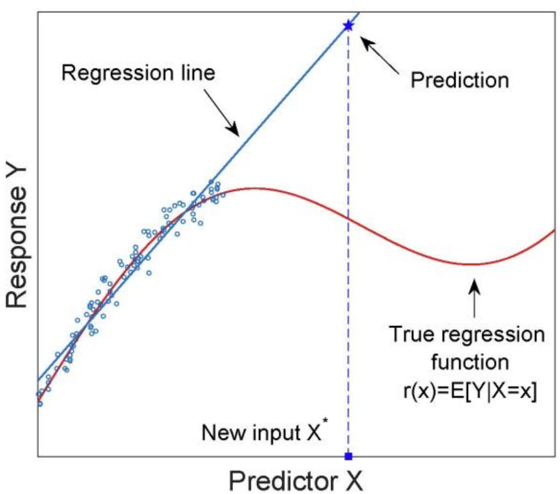
> Interpolation





> Extrapolation





Working with missing data

```
from sklearn.impute import SimpleImputer

imr = SimpleImputer(missing_values= np.nan, strategy='mean') #median,
most_frequent
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)


from sklearn.impute import KNNImputer

kimr = KNNImputer() # basic n_neighbors = 5
kimr.fit_transform(df.values)

df.interpolate('linear')
df.interpolate('quadratic')
df.interpolate('cubic')
df.interpolate(method='spline', order = 2)
df.interpolate(method='polynomial", order = 3)
```

Categorical data

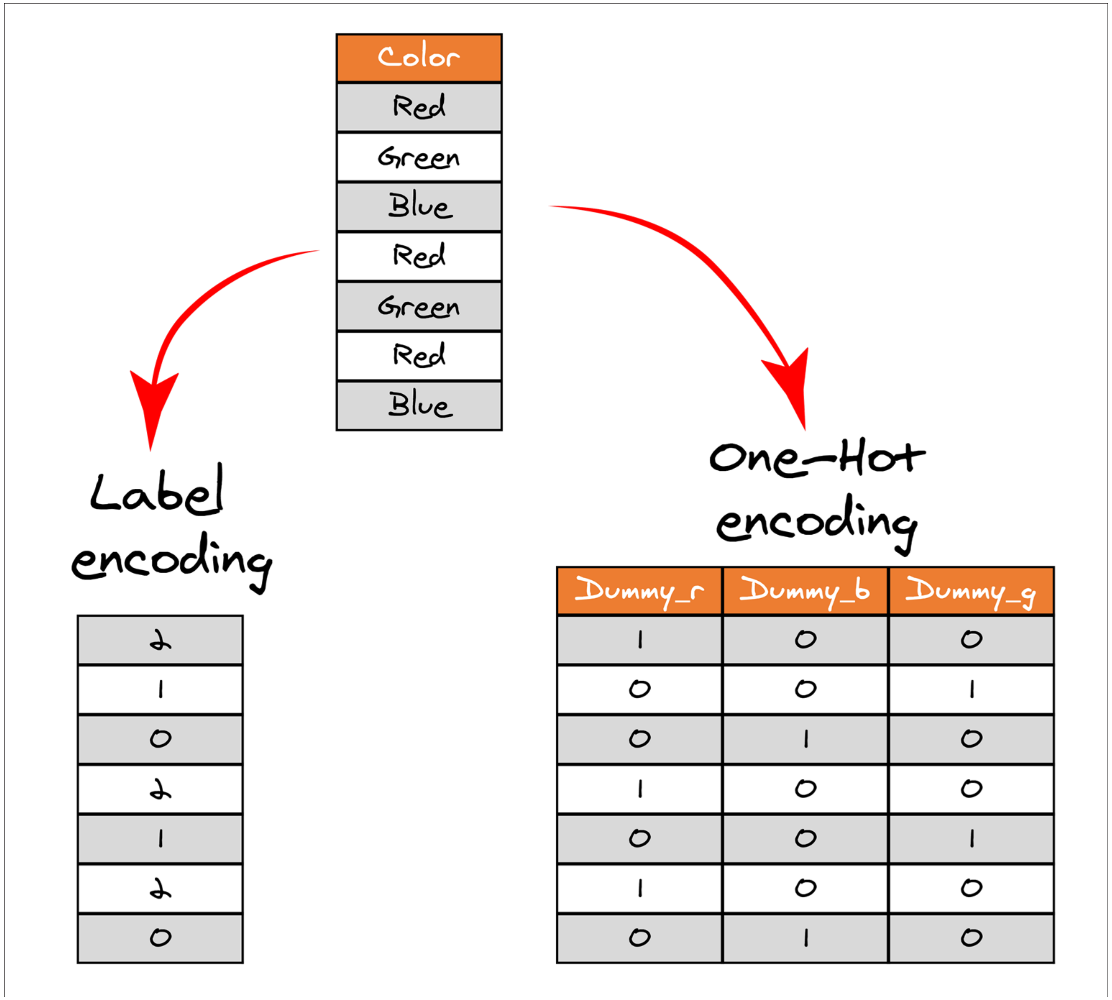
```
df = pd.DataFrame([[‘green’, ‘M’, 10.1, ‘class2’],
                   [‘red’, ‘L’, 13.5, ‘class1’],
                   [‘blue’, ‘XL’, 15.3, ‘class2’]])
df.columns = [‘color’, ‘size’, ‘price’, ‘classlabel’]

size_mapping = {‘M’:1, ‘L’:2, ‘XL’:3}
df[‘size’] = df[‘size’].map(size_mapping)

inv_size_mapping = {v: k for k, v in size_mapping.items()}
df[‘size’].map(inv_size_mapping)
```

Categorical data

> Label encoding (integer encoding)
  - ['cat', 'dog', 'bird'] -> [0, 1, 2]
  - simple and memory efficient
  - Implies ordinal relationship which can confuse the models

> One-hot encoding
  - cat: [1, 0, 0]
  - dog: [0, 1, 0]
  - bird: [0, 0, 1]
  - No ordinal assumption
  - High dimensionality (memory and computation cost)

Categorical data

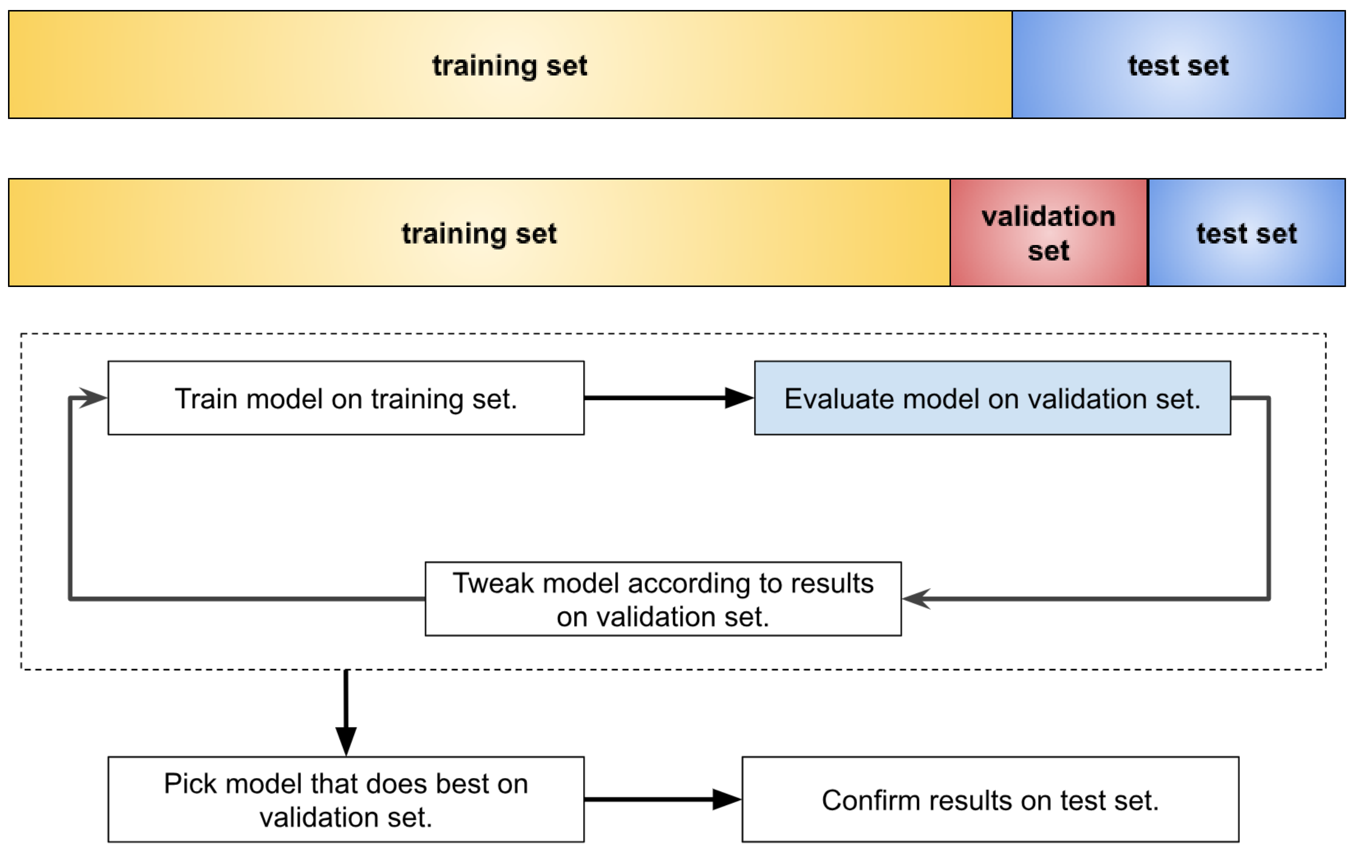


Categorical data

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder

order_enc = col_trans = ColumnTransformer([(‘ord_enc’, OrdinalEncoder(dtype = int)
, [‘color’])])
X_trans = col_trans.fit_transform(df)
X_trans

X = df[[‘color’, ‘size’, ‘price’]].values
c_transf = ColumnTransformer([(‘onehot’, OneHotEncoder(dtype=int), [0]), (‘nothing’,
‘passthrough’, [1, 2])])

c_transf.fit_transform(X)

pd.get_dummies(df[[‘price’, ‘color’, ‘size’]])
```

Dataset splitting

> Training, validation, and test sets
  - A validation set performs the initial testing on the model as it is being trained



Dataset splitting

> https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

```
from sklearn.model_selection import train_test_split

df_wine = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data', header=None)
X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0, stratify = y)
```

Data scaling

> Normalization
  - Linear scaling: when the feature is mostly uniformly distributed

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

  - Z-score scaling: when the feature is normally distributed

$$x' = \frac{x - \mu}{\sigma}$$

  - Log scaling: when the feature distribution is heavy skewed

$$x' = \log(x)$$

  - Clipping: when the feature contains extreme outliers

$$x' = clip(x, min, max)$$

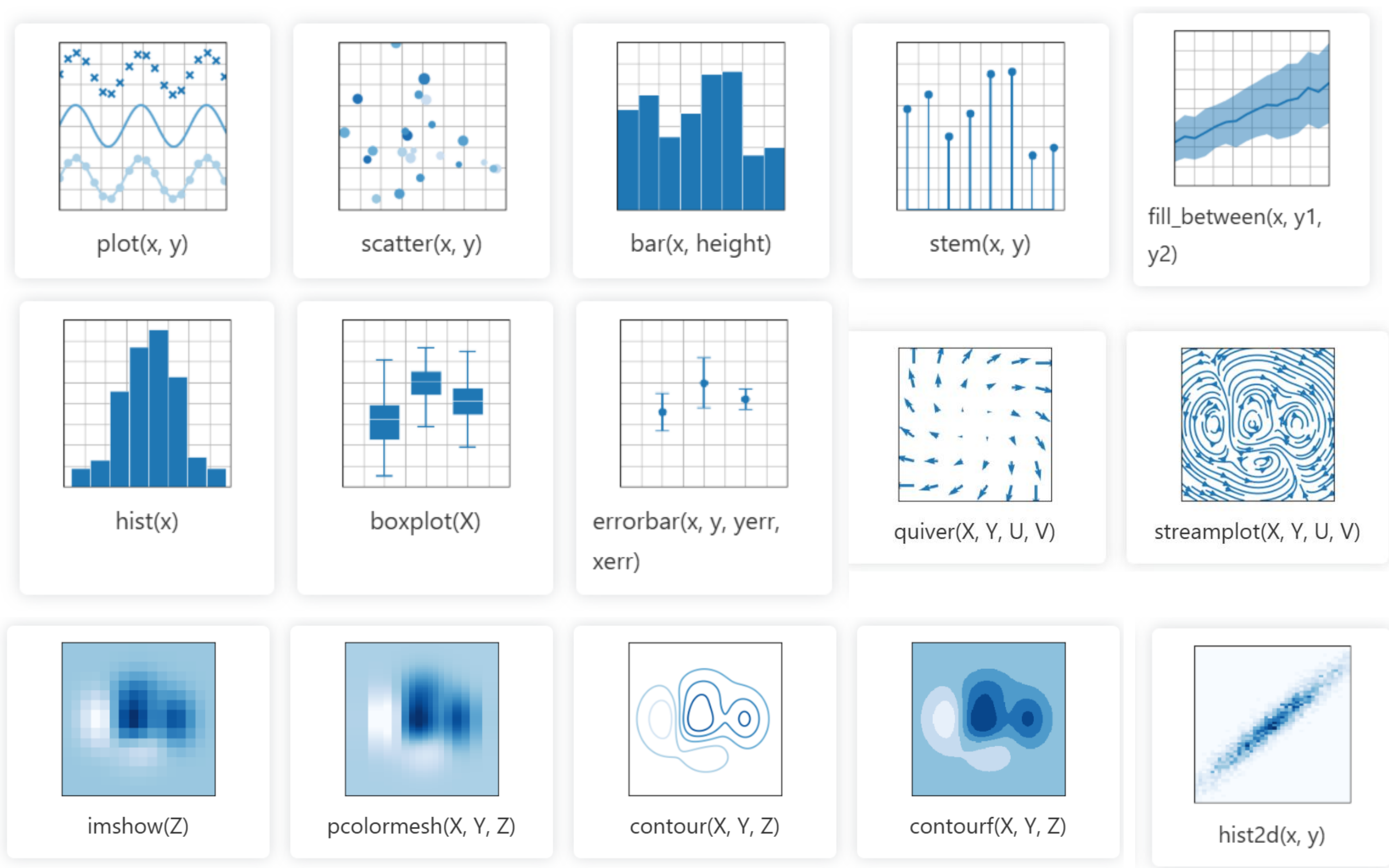
Data scaling

> https://scikit-learn.org/stable/api/sklearn.preprocessing.html

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

from sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler = StandardScaler() # MinMaxScaler, RobustScaler, MaxAbsScaler
scaler.fit(iris_df)
iris_scaled = scaler.transform(iris_df)
```

Visualization

> https://matplotlib.org/stable/plot_types/index.html



Recommended reading

- https://developers.google.com/machine-learning/crash-course/numerical-data
- https://amueller.github.io/COMS4995-s20/slides/aml-02-matplotlib/#p1

Reference

- 알고리즘 중심의 머신러닝 가이드, Chapter 2
- 머신러닝 교과서 파이토치편, Chapter 4
- 파이썬 머신러닝 완벽가이드 Chapter 1, 2
- UC Berkeley, Concise Machine Learning, Chapter 1

- Python for data analysis — ML Engineering
- Recap: Data preprocessing — ML Engineering
- Python for scientific computing — ML Engineering