Basic formulas for statistical measures

Law	Discrete	Continuous
Mean (μ) **Expected Value**	$\mu = E(X) = \Sigma\,(x\,f(x))$	$\mu = E(X) = \int (x\,f(x))\,dx$
Joint Mean **Expected Value**	$\mu_{g(x,y)} = \Sigma\Sigma\,(g(x,y) * f(x,y))$	$\mu_{g(x,y)} = \int\int (g(x,y) * f(x,y))\,dx\,dy$
Variance (σ^2)	$\sigma^2 = \Sigma\,(x^2 * f(x)) - \mu^2$	$\sigma^2 = \int (x^2 * f(x))\,dx - \mu^2$
	$\sigma^2 = E\,(x^2) - \mu^2$	
Covariance (σxy) **check linearity** $-\infty < \sigma_{xy} < \infty$	$\sigma_{xy} = \Sigma\Sigma\,[(xy)\,f(x,y)] - \mu_x\mu_y$	$\sigma_{xy} = \int\int [(xy)\,f(x,y)]\,dx\,dy - \mu_x\,\mu_y$
	$\mu_x = \Sigma\,(x * g(x))$	$\mu_x = \int (x * g(x))\,dx$
	$\mu_y = \Sigma\,(y * h(y))$	$\mu_y = \int (y * h(y))\,dy$
	$\sigma_{xy} = E(XY) - \mu_x\,\mu_y$	
Correlation (ρxy) **Measure linearity** $-1 < \rho_{xy} < 1$	$\sigma_x{}^2 = E(X^2) - \mu_x{}^2$	$\sigma_y{}^2 = E(y^2) - \mu_y{}^2$
	$\rho_{xy} = \sigma_{xy} / (\sigma_x\,\sigma_y)$	

Summary of Common Probability Distributions

Name	Probability Distribution	Mean	Variance
Discrete			
Uniform	$\dfrac{1}{n}, a \le b$	$\dfrac{(b+a)}{2}$	$\dfrac{(b-a+1)^2 - 1}{12}$
Binomial	$\dbinom{n}{x} p^x (1-p)^{n-x},$ $x = 0, 1, \ldots, n, 0 \le p \le 1$	np	$np(1-p)$
Geometric	$(1-p)^{x-1}p,$ $x = 1, 2, \ldots, 0 \le p \le 1$	$1/p$	$(1-p)/p^2$
Negative binomial	$\dbinom{x-1}{r-1}(1-p)^{x-r}p^r$ $x = r, r+1, r+2, \ldots, 0 \le p \le 1$	r/p	$r(1-p)/p^2$
Hypergeometric	$\dfrac{\dbinom{K}{x}\dbinom{N-K}{n-x}}{\dbinom{N}{n}}$ $x = \max(0, n-N+K), 1, \ldots$ $\min(K,n), K \le N, n \le N$	$np,$ where $p = \dfrac{K}{N}$	$np(1-p)\left(\dfrac{N-n}{N-1}\right)$
Poisson	$\dfrac{e^{-\lambda}\lambda^x}{x!}, x = 0, 1, 2, \ldots, 0 < \lambda$	λ	λ
Continuous			
Uniform	$\dfrac{1}{b-a}, a \le x \le b$	$\dfrac{(b+a)}{2}$	$\dfrac{(b-a)^2}{12}$
Normal	$\dfrac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ $-\infty < x < \infty, -\infty < \mu < \infty, 0 < \sigma$	μ	σ^2
Exponential	$\lambda e^{-\lambda x}, 0 \le x, 0 < \lambda$	$1/\lambda$	$1/\lambda^2$
Erlang	$\dfrac{\lambda^r x^{r-1} e^{-\lambda x}}{(r-1)!}, 0 < x, r = 1, 2, \ldots$	r/λ	r/λ^2
Gamma	$\dfrac{\lambda x^{r-1} e^{-\lambda x}}{\Gamma(r)}, 0 < x, 0 < r, 0 < \lambda$	r/λ	r/λ^2
Weibull	$\dfrac{\beta}{\delta}\left(\dfrac{x}{\delta}\right)^{\beta-1} e^{-(x/\delta)^\beta},$ $0 < x, 0 < \beta, 0 < \delta$	$\delta\Gamma\left(1+\dfrac{1}{\beta}\right)$	$\delta^2\Gamma\left(1+\dfrac{2}{\beta}\right)$ $-\delta^2\left[\Gamma\left(1+\dfrac{1}{\beta}\right)\right]^2$
Lognormal	$\dfrac{1}{x\sigma\sqrt{2\pi}}\exp\left(\dfrac{-[\ln(x)-\theta]^2}{2\omega^2}\right)$	$e^{\theta+\omega^2/2}$	$e^{2\theta+\omega^2}(e^{\omega^2}-1)$

- Binominal (Bernolli) used for ==x success== in n trials (true and false trials)
- Geometric is used for x trials to get the ==first== success
- Negative binonminal is used for ==kth== success in x trials
- Poission is used for number of outcomes in a region with ==average or rate==
- Uniform is used when the outcomes have ==equal likelihood==
- Exponential is used for ==time between events with average==
- Standard Normal Distribution has the following equation to convert its table0 values to any other normal distribution system

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}}$$

$$Z = \frac{X - \mu}{\sigma}$$

- Where μ is the mean of population
- σ is the standard deviation of the population
- x is the value from foreign system with normal distribution
- z is the result equivalent from standard normal distribution

---

Data Analysis

- **Descriptinve statisitics**
  - directly from population
  - values called parameters

- **Inference statistics**
  - calculated from sample
  - values called statistic

Measure of Centraility
- mean
- trimmed mean
- median
- mode

Measure of Variabliltiy
- range
- variance
- standard deviation

Measure of Position
- Quartile
- Percentile
- Box Plot

Measure of Shape
- skewness

Estimation
- Classical (Random Sample)
- Baysian (Prior Knowledge)
- Point Estimation
- Confidence Interval

Hypothesis

Frequency Table

- Range = Upper value – Lower value.
- Sturges's formula for class count: k =1+3.322log(n) → n number of items.
- Calculate class width: R / K.
- Assign data points to classes to build the frequency table.

Quartile

- Count the data points.
- Order them from smallest to largest.
- Locate the median (Q2).
- Divide the dataset into two at the median.
- For odd-sized datasets, exclude the median for the next steps.
- For even-sized datasets, include both middle values in the halves.
- The median of the lower half is Q1.
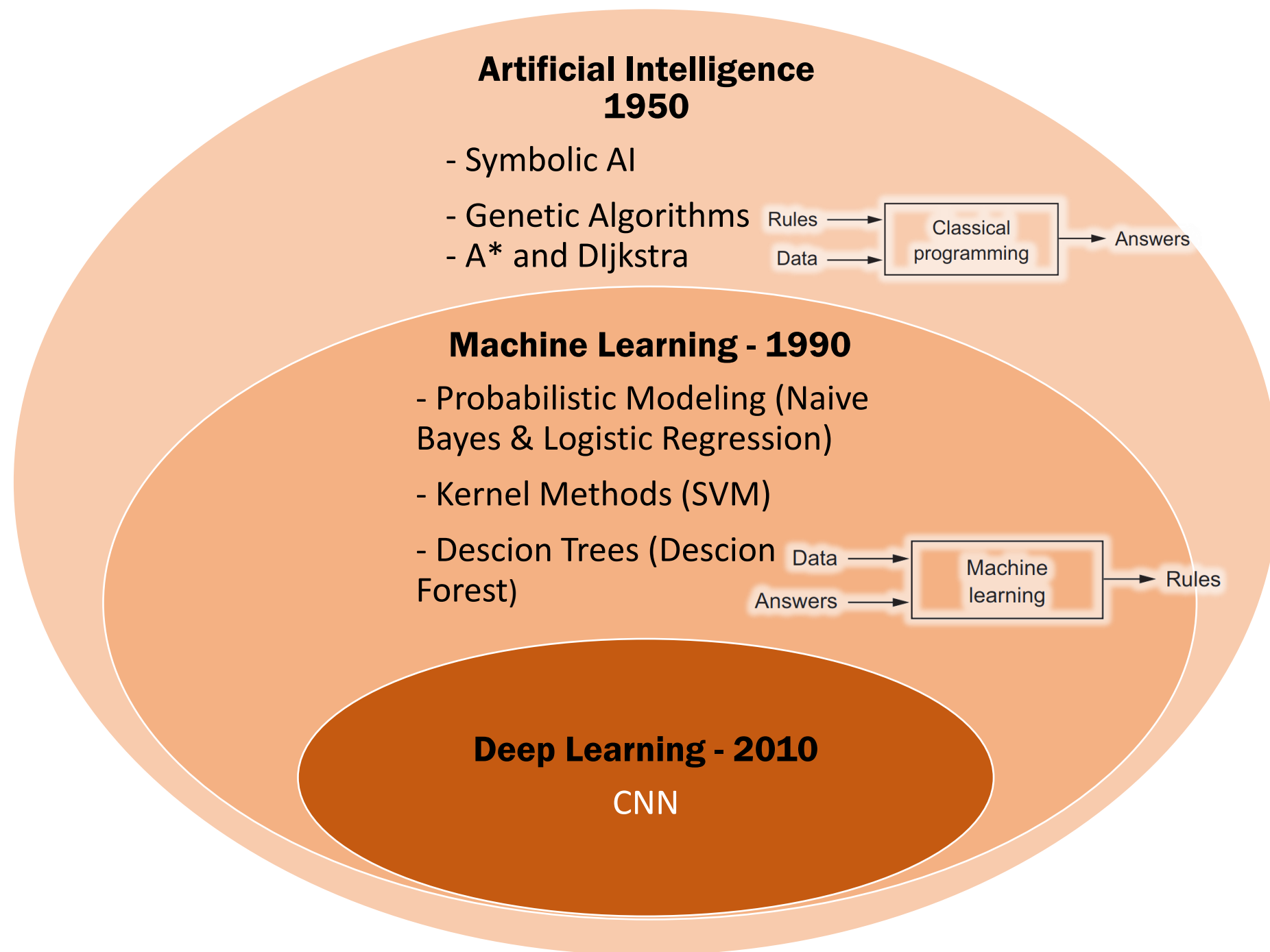- The median of the upper half is Q3

Category		Algorithm	When to Use
		Bubble Sort	Simple sorting, small datasets
		Insertion Sort	Small datasets, mostly sorted data
		Selection Sort	Small datasets, simplicity over efficiency
	Sorting	Merge Sort	Large datasets, require stable sort
		Quick Sort	Large datasets, average case speed
		Heap Sort	Need to sort large datasets in place
		Radix Sort	Large datasets with fixed-length keys
		Bucket Sort	Large datasets, uniformly distributed
		External Sort	Very large datasets that don't fit in memory
		Linear Search	Unsorted data, small datasets
	Searching	Binary Search	Sorted data, large datasets
		Depth-First Search (DFS)	Graphs, exploring edges/node paths
		Breadth-First Search (BFS)	Graphs, finding shortest path
	Data Structures		Arrays, Linked Lists, queue, stack, priority queue
	String Manipulation		
Hierarchical data	Tree and BST	Tree Traversal	Accessing/Searching/Manipulating tree data
		BST Operations	Efficient search, insert, delete in sorted data
		AVL Trees	Self-balancing BST where lookup speed is critical
networks, paths, and relationships	Graph	Dijkstra's Algorithm	Shortest path in weighted graph without negative edges
		Kruskal's Algorithm	Minimum spanning tree in sparse graphs
		Prim's Algorithm	Minimum spanning tree in dense graphs
sequence of decisions	Greedy		
Can be divided to sub problems	Divide and conquer	Quick Sort	When average case performance is key
		Merge Sort	Large datasets where stability is key
		Binary Search	Quickly finding an item in sorted data
		Strassen's Matrix Multiplication	Large matrix multiplication
		Karatsuba Algorithm	Fast multiplication of large numbers
Permutations Combinations	Backtracking		
Overlapping Subproblems Optimal Substructure	Dynamic Programming		Fibonacci, knapsack and palindromes

**Artificial Intelligence
1950**

- Symbolic AI

- Genetic Algorithms
- A* and DIjkstra

Rules → Classical programming → Answers
Data →

Machine Learning - 1990

- Probabilistic Modeling (Naive Bayes & Logistic Regression)

- Kernel Methods (SVM)

- Descion Trees (Descion Forest)

Data → Machine learning → Rules
Answers →

Deep Learning - 2010

CNN

1. **Type**:
   - **Supervised** Learning: Algorithms learn from labeled data, with input-output pairs provided during training.
   - **Unsupervised** Learning: Algorithms learn patterns and structures in unlabeled data without explicit supervision.
   - **Reinforcement** Learning: Agents learn to interact with an environment to maximize rewards through trial and error.

2. **Functionality**:
   - **Classification**: Algorithms predict discrete class labels for given input data.
   - **Regression**: Algorithms predict continuous values based on input data.
   - **Clustering**: Algorithms group similar data points together based on some similarity measure.
   - **Dimensionality Reduction**: Techniques reduce the number of input variables while preserving important information.

3. **Model Complexity**:
   - **Parametric Models**: Models have a fixed number of parameters and a predefined structure, such as linear regression.
   - **Non-parametric Models**: Models have a flexible number of parameters, such as decision trees and k-nearest neighbors.
   - **Deep Learning**: Neural networks with multiple hidden layers, capable of learning intricate patterns and representations from data.
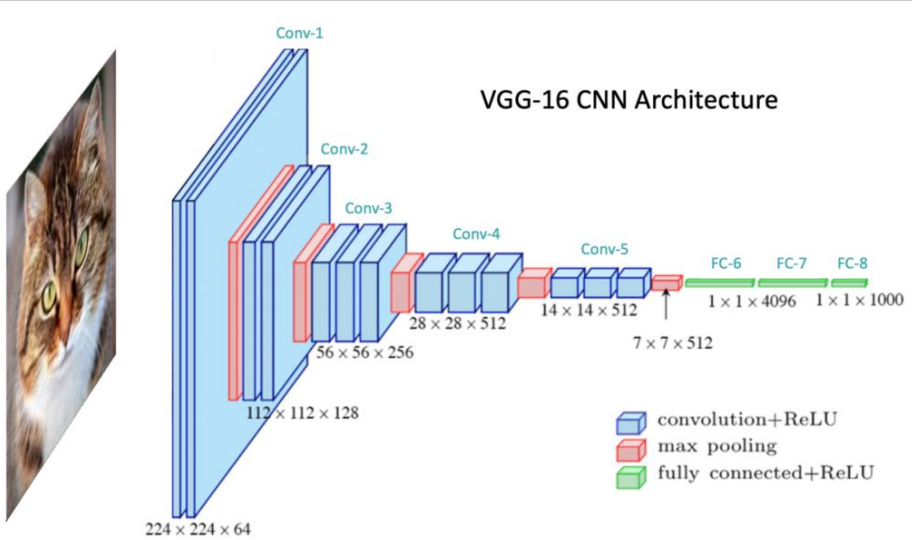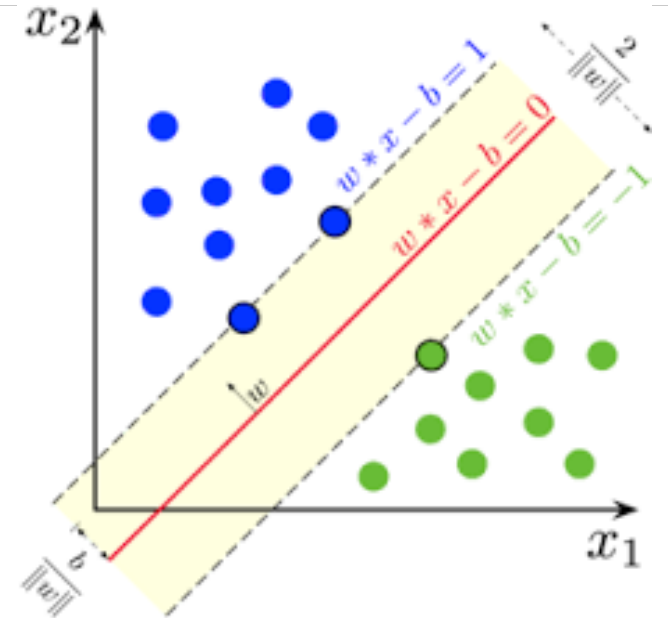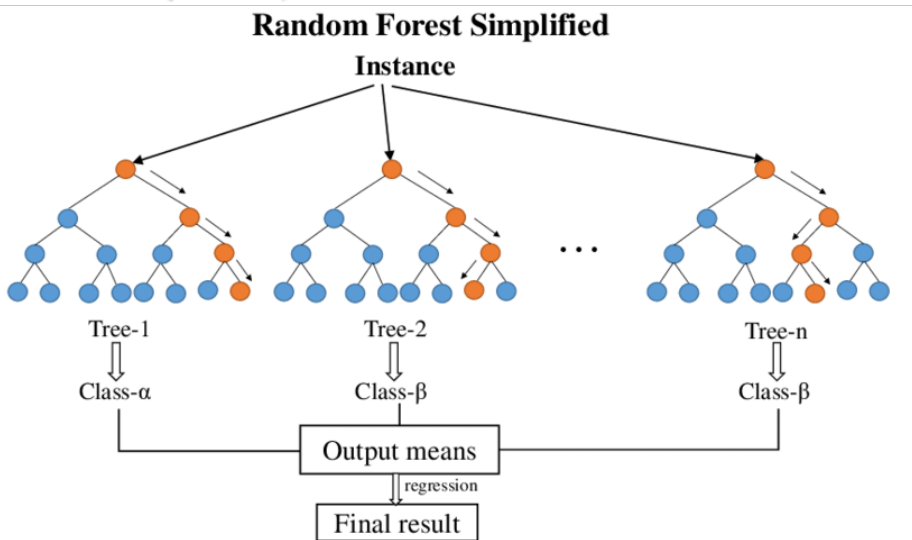
4. **Domain or Application**:
   - **Computer Vision**: Algorithms designed to process and interpret visual data.
   - **Natural Language Processing (NLP)**: Algorithms designed to understand and process human language.
   - **Speech Recognition**: Algorithms designed to transcribe spoken language into text.
   - **Recommendation Systems**: Algorithms designed to recommend items or content to users based on their preferences.

CNN and RNNS for high-dimensional, large, complex dataset (low interpretability and higher computational resources)

➢ Let's say you're working on predicting the price of a house based on various features such as square footage, number of bedrooms, number of bathrooms, and location. In this case, you have a continuous target variable (price) that you want to predict, and the relationship between the features and the target variable is likely to be linear.

➢ Using linear regression would be appropriate in this scenario because it models the relationship between the input features and the target variable as a linear function. Linear regression would allow you to estimate the coefficients of each feature, which represent the change in the target variable for a one-unit change in each feature, holding all other features constant.

➢ On the other hand, logistic regression is used for binary classification problems where the target variable is categorical (e.g., yes/no, 1/0). If the target variable is not binary or if the relationship between the features and the target variable is not linear, logistic regression would not be suitable.
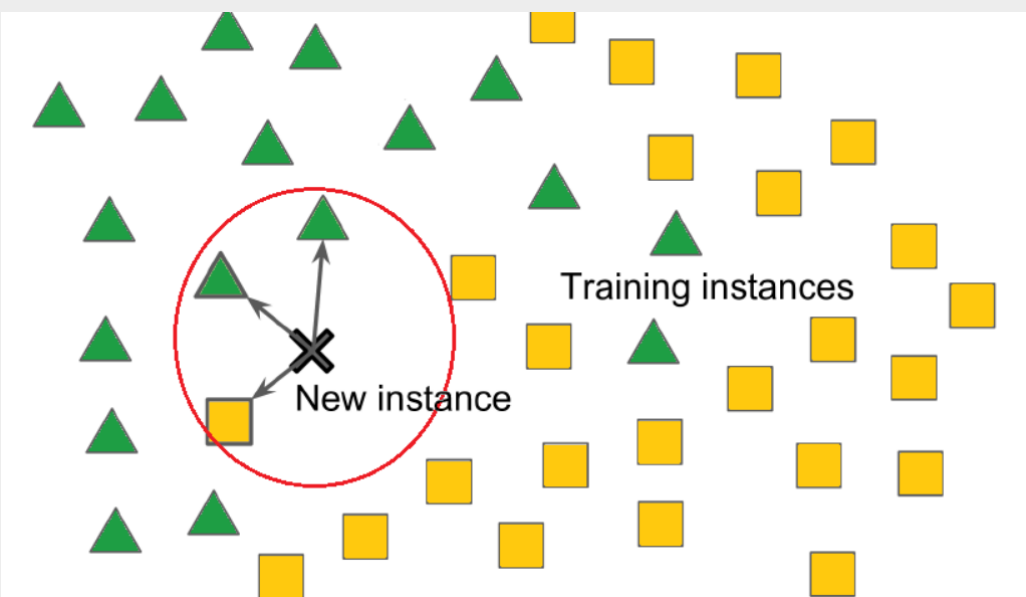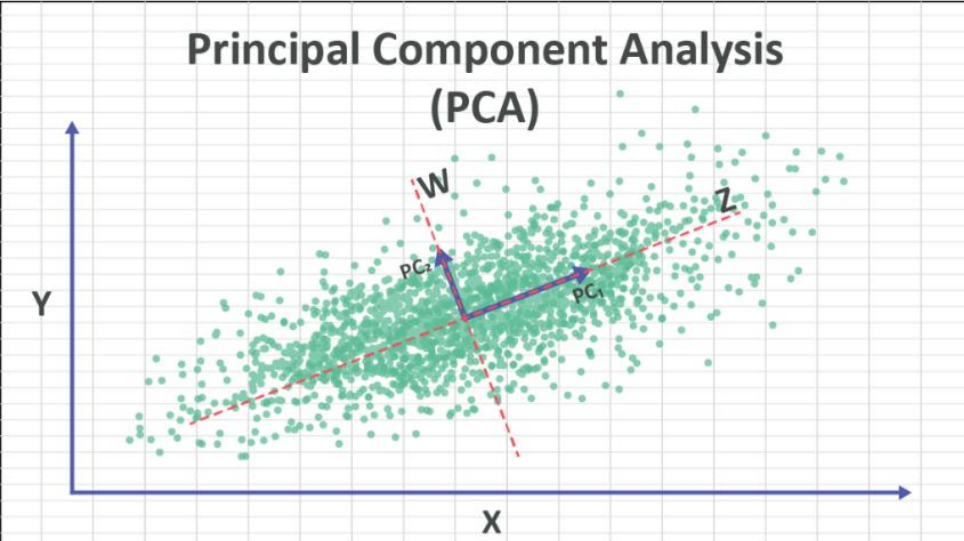
➢ Let's consider a scenario where you have a dataset with high-dimensional features, such as images represented as pixel values. You want to perform binary classification to distinguish between images of cats and dogs. Each image is represented by a large number of features (e.g., pixel values), making the dataset high-dimensional.

➢ In this case, using SVM with a kernel function, such as the radial basis function (RBF) kernel, can effectively handle high-dimensional data and find complex nonlinear decision boundaries between the classes. SVM with the RBF kernel is capable of capturing intricate patterns in the data, which may be necessary for distinguishing between images of cats and dogs with high accuracy.

➢ On the other hand, PCA is a dimensionality reduction technique that aims to reduce the dimensionality of the dataset by finding a lower-dimensional representation while preserving most of the variance in the data. However, PCA may not be as effective in capturing complex nonlinear relationships in the data, especially in cases where the decision boundary between classes is highly nonlinear or intricate.

➢ Consider a text classification problem where you want to classify emails as either spam or not spam based on the words contained in the email. Each email is represented by a bag-of-words representation, where the presence or absence of words in the email is used as features for classification. In this scenario:

➢ Naive Bayes: Naive Bayes classifiers are well-suited for text classification tasks because they assume that features are conditionally independent given the class label. Despite this simplifying assumption, Naive Bayes classifiers often perform well on text data and are computationally efficient. They calculate the probability of each class given the features and select the class with the highest probability.

Algorithm	Type	Functionality	Deep Learning	Model Complexity	Code example	image
| **CNN** | Supervised | Classification, Regression | Yes | Deep | ```python
from keras.models import Sequential
from keras.layers import Conv2D,
MaxPooling2D, Flatten, Dense
model = Sequential()
model.add(Conv2D(32, (3, 3),
activation='relu', input_shape=(32, 32,
3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3),
activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))
``` |  |
| **SVM (Support Vector Machine)** | Supervised | Classification, Regression | No | Parametric | ```python
from sklearn import datasets
from sklearn.model_selection
import train_test_split from sklearn.svm
import SVC
iris = datasets.load_iris()
X_train, X_test, y_train, y_test =
train_test_split(iris.data, iris.target,
test_size=0.3, random_state=42)
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
accuracy = svm_model.score(X_test, y_test)
print("Accuracy:", accuracy)
``` |  |
| **Decision Forests** | Supervised | Classification, Regression | No | Non-parametric | ```python
from sklearn import datasets
from sklearn.model_selection
import train_test_split
from sklearn.ensemble import
RandomForestClassifier
iris = datasets.load_iris()
X_train, X_test, y_train, y_test =
train_test_split(iris.data, iris.target,
test_size=0.3, random_state=42)
rf_model =
RandomForestClassifier(n_estimators=100)
rf_model.fit(X_train, y_train)
accuracy = rf_model.score(X_test, y_test)
print("Accuracy:", accuracy)
``` |  |

| Naive Bayes | Supervised | Classification | No | Parametric | ```from sklearn import datasets
from sklearn.model_selection
import train_test_split
from sklearn.naive_bayes
import GaussianNB
iris = datasets.load_iris()
X_train, X_test, y_train, y_test =
train_test_split(iris.data, iris.target,
test_size=0.3, random_state=42)
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
accuracy = nb_model.score(X_test, y_test)
print("Accuracy:", accuracy)``` | |
|---|---|---|---|---|---|---|
| **Linear Regression** | Supervised | Regression | No | Parametric | ```from sklearn.datasets import load_boston
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
LinearRegression
boston = load_boston()
X_train, X_test, y_train, y_test =
train_test_split(boston.data, boston.target,
test_size=0.3, random_state=42)
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
accuracy = lr_model.score(X_test, y_test)
print("Accuracy:", accuracy)``` | |
| **Logistic Regression** | Supervised | Classification | No | Parametric | ```from sklearn import datasets
from sklearn.model_selection
import train_test_split
from sklearn.linear_model
import LogisticRegression
iris = datasets.load_iris()
X_train, X_test, y_train, y_test =
train_test_split(iris.data, iris.target,
test_size=0.3, random_state=42)
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
accuracy = lr_model.score(X_test, y_test)
print("Accuracy:", accuracy)``` | |

For Naive Bayes, the figure shows Bayes' theorem:

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

with annotations: "Likelihood of the Evidence given that the Hypothesis is True" (P(E|H)), "Prior Probability of the Hypothesis" (P(H)), "Posterior Probability of the Hypothesis given that the Evidence is True" (P(H|E)), and "Prior Probability that the evidence is True" (P(E)).

| KNN (K-Nearest Neighbors) | Supervised | Classification | No | Non-parametric | ```from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.3, random_state=42)
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
accuracy = knn_model.score(X_test, y_test)
print("Accuracy:", accuracy)``` | |
|---|---|---|---|---|---|---|
| PCA (Principal Component Analysis) | Unsupervised | Dimension Reduction | No | Non-parametric | ```from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
iris = load_iris()
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(iris.data)``` | |
| K-Means | Unsupervised | Clustering | No | Non-parametric | ```from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)``` | |

| GANs (Generative Adversarial Networks) | Unsupervised | Generation | Yes | Deep | | |
| Autoencoder | Unsupervised | Dimension Reduction, Generation | Yes | Deep | | |

Advanced CNNs

Model	Type	Best Case to Use	Model Size
SqueezeNet	CNN	When limited computational resources are available	Small
MobileNet	CNN	Mobile and edge devices, real-time applications	Small
BiLSTM (Bidirectional LSTM)	RNN	Sequential data with bidirectional context	Medium
GRU (Gated Recurrent Unit)	RNN	Sequences with less complex dependencies	Medium
LSTM (Long Short-Term Memory)	RNN	Long sequences with complex dependencies	Medium
DenseNet	CNN	Image classification tasks	Large
EfficientNet	CNN	State-of-the-art performance with efficient use of resources	Large
NASNet (Neural Architecture Search Network)	CNN	High-performance image classification tasks	Large
AlexNet	CNN	Legacy image classification tasks	Large
GoogLeNet (Inception)	CNN	Tasks requiring strong generalization	Large
VGG (Visual Geometry Group) networks (e.g., VGG16, VGG19)	CNN	Image classification tasks, benchmarking	Large
Attention-based models (e.g., Transformer with Attention Mechanisms)	RNN/CNN	NLP tasks, sequence-to-sequence learning	Varies
Transformer	RNN/CNN	NLP tasks, language translation	Varies

Simple list:

1) SVM (Support Vector Machine)
2) Decision Forests
3) Naive Bayes
4) Linear Regression
5) Logistic Regression
6) KNN (K-Nearest Neighbors)
7) PCA (Principal Component Analysis)
8) GANs (Generative Adversarial Networks)
9) Autoencoder
10) K-Means
11) AlexNet
12) VGG (Visual Geometry Group) networks (e.g., VGG16, VGG19)
13) GoogLeNet (Inception)
14) ResNet (Residual Network)
15) DenseNet
16) MobileNet
17) EfficientNet
18) NASNet (Neural Architecture Search Network)
19) SqueezeNet
20) LSTM (Long Short-Term Memory)
21) GRU (Gated Recurrent Unit)
22) BiLSTM (Bidirectional LSTM)
23) Transformer
24) Attention-based models (e.g., Transformer with Attention Mechanisms)