



Week 13

Machine Learning

Dr Ammar Masood
Department of Cyber Security,
Air University Islamabad

Table of Contents

- Nonparametric Models
- Ensemble modeling
- Support vector machines
- Nearest neighbor modeling

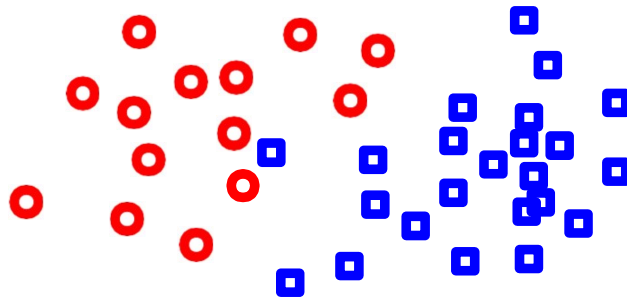
Nonparametric Models

- **Parametric Models:** Fixed number of parameters (e.g., linear regression)
- **Nonparametric Models:** Flexible, grow with data (e.g., nearest-neighbors)
- **Instance-based Learning:** Retain all training examples; no abstraction

Nearest-Neighbor Models

- **Table Lookup:** Return label for identical match → Poor generalization
- **k-Nearest Neighbors (kNN):**
 - Classify using the **majority label** of nearest k examples
 - Given a query x_q , instead of finding an example that is equal to x_q , find the **k examples** that are nearest to x_q .
 - Notation $NN(k, x_q)$ used to denote the set of k neighbors Nearest neighbors nearest to x_q .
 - For **classification**, find the set of neighbors $NN(k, x_q)$ and take the most common output Value.
 - for example, if $k=3$ and the output values are (Yes, No, Yes), then the classification will be Yes.
 - To do **regression**, we can take the **mean** or **median** of the k neighbors, or we can solve a linear regression problem on the neighbors..
 - Use **odd k** for binary classification to avoid ties

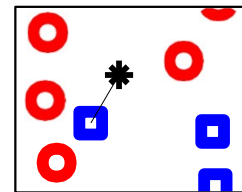
Nearest Neighbour Rule



Consider a two class problem where each sample consists of two measurements (x,y) .

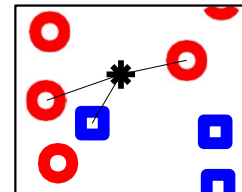
For a given query point q , assign the class of the nearest neighbour.

$k = 1$



Compute the k nearest neighbours and **assign the class by majority vote.**

$k = 3$



Nearest-Neighbor Models

- **Distance Metrics:**

- How do we measure the distance from a query point x_q to an example point x_j ?
- Minkowski distance or L^p norm, defined as

$$L^p(x_j, x_q) = (\sum_i |x_{j,i} - x_{q,i}|^p)^{1/p}.$$

- With $p=2$ this is Euclidean distance and with $p=1$ it is Manhattan distance
- With Boolean attribute values, the number of attributes on which the two points differ is called the **Hamming distance**.
- **Normalize** features to avoid scale bias
 - We can compute the mean μ_i and standard deviation σ_i of the values in each dimension, and rescale them so that $x_{j,i}$ becomes $(x_{j,i} - \mu_i)/\sigma_i$

Examples

- Measures distance between vectors:

- $x_j = (1, 2, 3)$
- $x_q = (4, 0, 3)$

- Manhattan Distance ($p = 1$)

$$L^1 = |1 - 4| + |2 - 0| + |3 - 3| = 3 + 2 + 0 = \boxed{5}$$

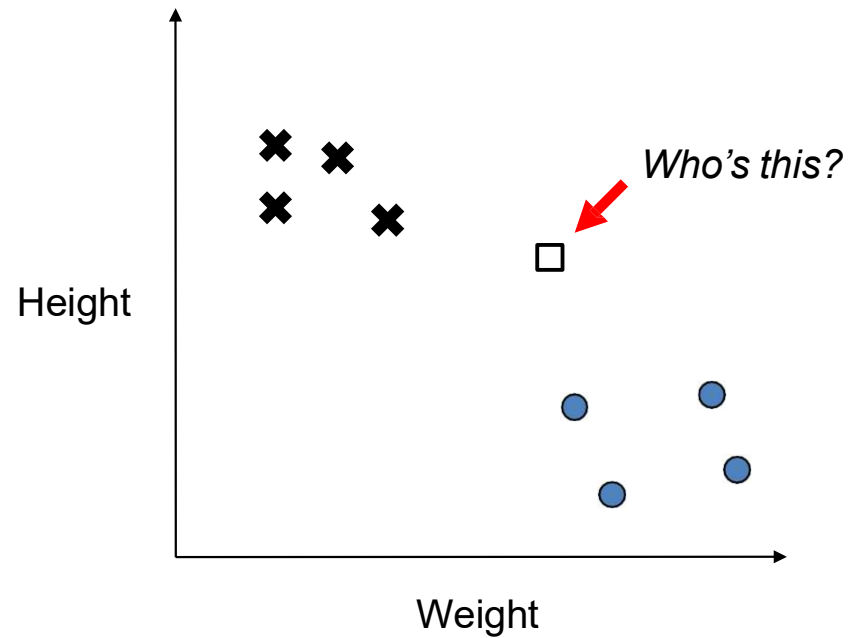
- Euclidean distance ($p=2$)

$$L^2 = \sqrt{(1 - 4)^2 + (2 - 0)^2 + (3 - 3)^2} = \sqrt{9 + 4 + 0} = \sqrt{13} \approx \boxed{3.61}$$

- Cubic Minkowski distance ($p=3$)

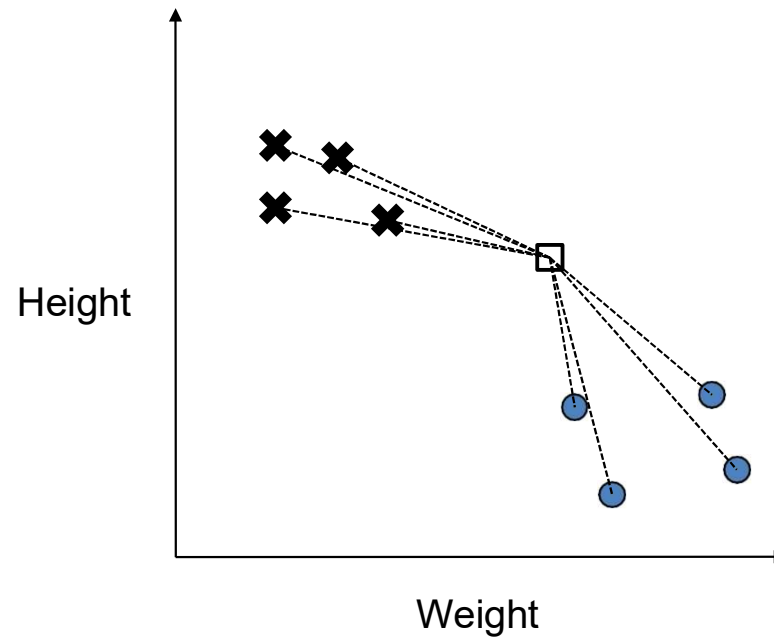
$$L^3 = (|1 - 4|^3 + |2 - 0|^3 + |3 - 3|^3)^{1/3} = (27 + 8 + 0)^{1/3} = 35^{1/3} \approx \boxed{3.27}$$

The K-Nearest Neighbour Algorithm



The K-Nearest Neighbour Algorithm

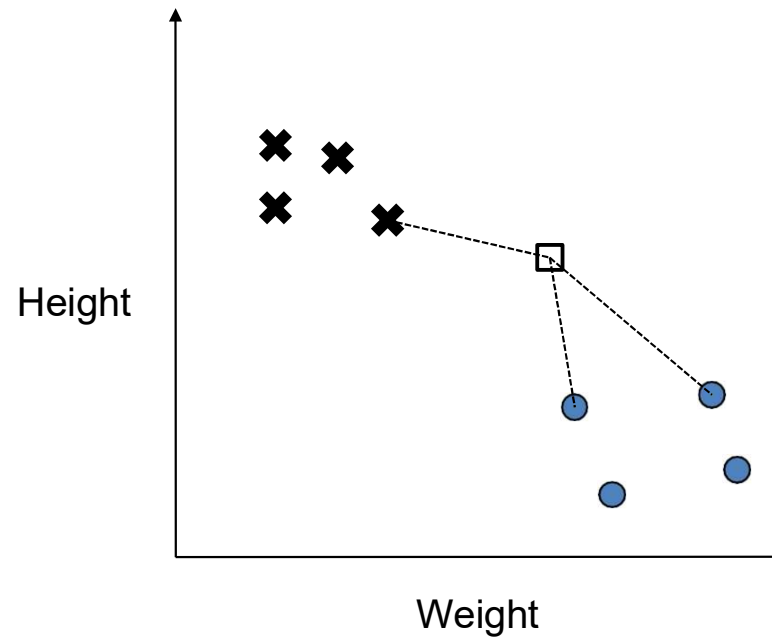
1. *Measure distance to all points*



The K-Nearest Neighbour Algorithm

1. Measure distance to all points
2. Find closest “k” points

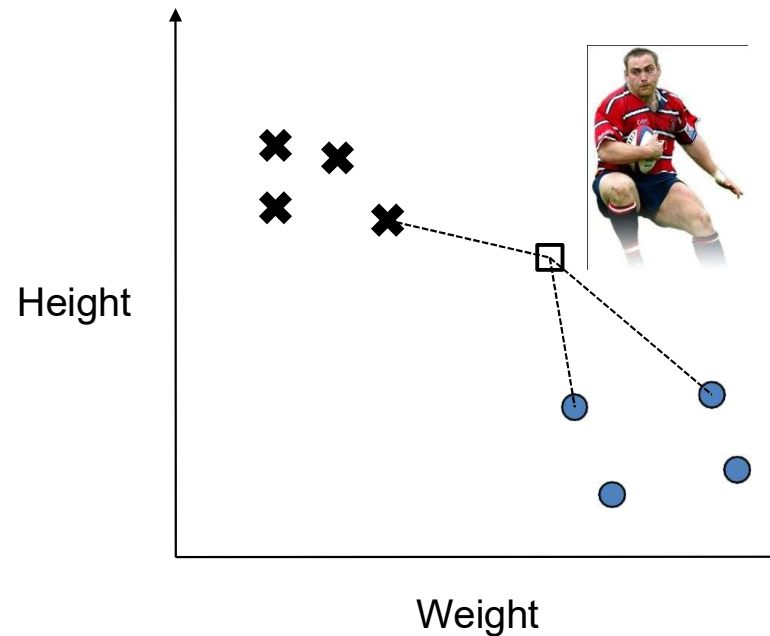
□ (here $k=3$, but it could be more)



The K-Nearest Neighbour Algorithm

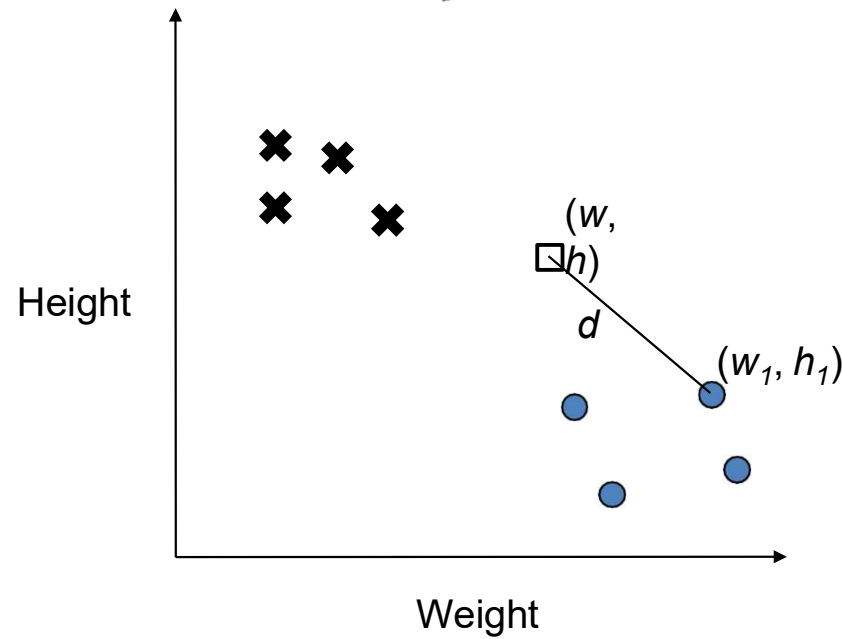
1. Measure distance to all points
2. Find closest “ k ” points
3. Assign majority class

□ (here $k=3$, but it could be more)



“Euclidean distance”

$$d = \sqrt{(w - w_1)^2 + (h - h_1)^2}$$



The K-Nearest Neighbour Algorithm

for each testing point

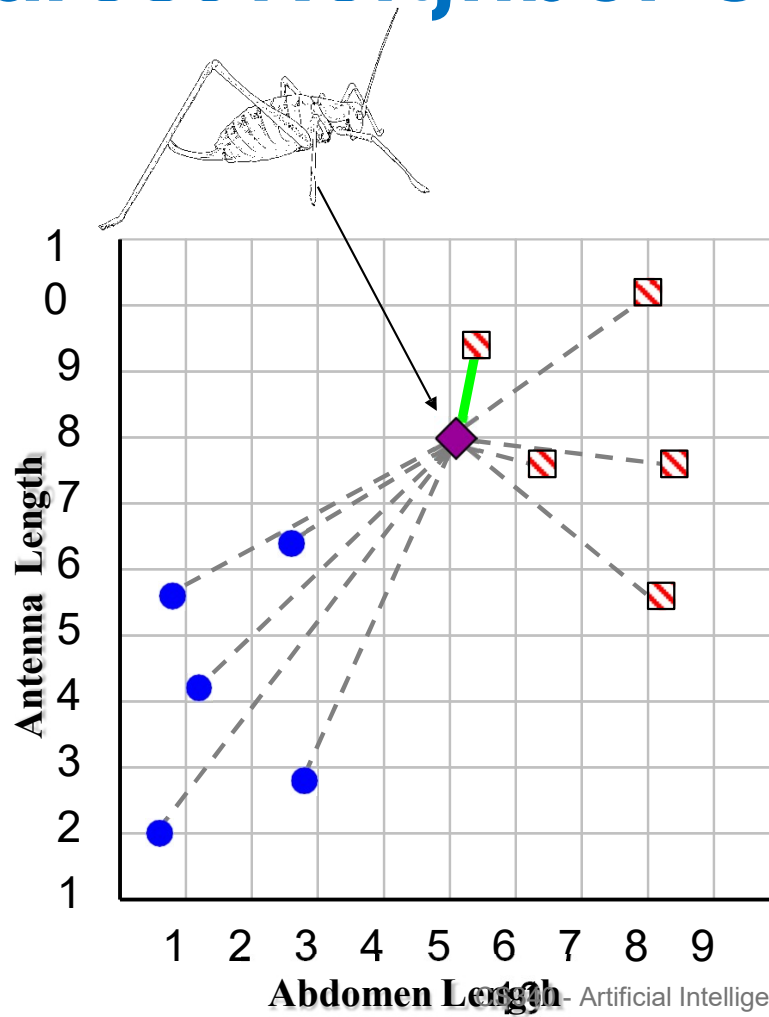
measure distance to every training point find the k closest points

identify the most common class among those k
predict that class

end

- **Advantage:** Surprisingly good classifier!
- **Disadvantage:** Have to store the entire training set in memory

Nearest Neighbor Classifier



If the **nearest** instance to the **previously unseen instance** is a **Katydid**
 class is **Katydid**
 else
 class is **Grasshopper**

▣ **Katydid**
 ● **Grasshoppers**

K-Nearest Neighbour Model

- Example : Classify whether a customer will respond to a survey question using a 3-Nearest Neighbor classifier

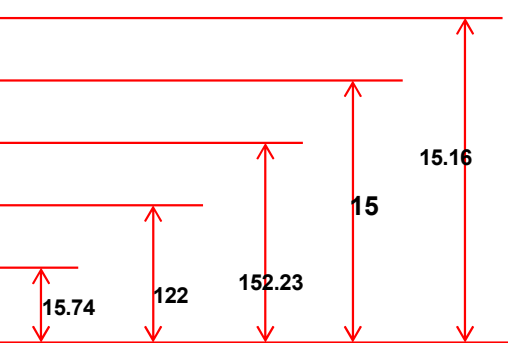
Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

K-Nearest Neighbour Model

- Example : 3-Nearest

Neigh

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?



Distances from David to neighbors:

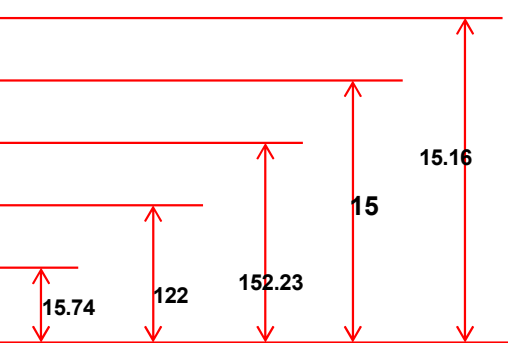
- Distance to Nellie: 15.74
- Distance to Rachel: 122
- Distance to Hannah: 152.23
- Distance to the 3rd neighbor (Hannah): 15.16

K-Nearest Neighbour Model

- Example : 3-Nearest

Neigh

Customer	Age	Income	No. credit cards	Response
John				No
Rachel				Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie				Yes
David	37	50K	2	?



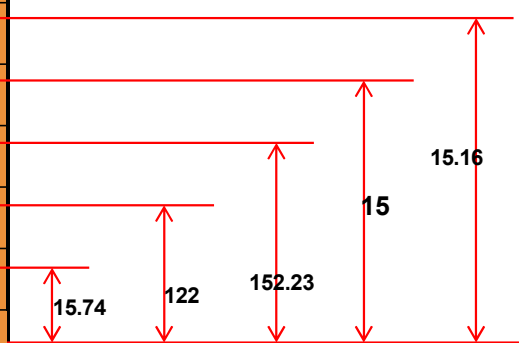
Three nearest ones to David are: No, Yes, Yes

K-Nearest Neighbour Model

- Example : 3-Nearest

Neigh

Customer	Age	Income	No. credit cards	Response
John				No
Rachel				Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie				Yes
David	37	50K	2	Yes?



Three nearest ones to David are: No, Yes, Yes

K-Nearest Neighbour Model

- Example: For the example we saw earlier, pick the best K from the set {1, 2, 3} to build a K-NN classifier

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

<http://vision.stanford.edu/teaching/cs231n-demos/knn/>

HW: Solve this Problem

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Versicolor
7.3	2.9	Virginica
6.0	2.7	Versicolor
5.8	2.8	Virginica
6.3	2.3	Versicolor
5.1	2.5	Versicolor
6.3	2.5	Versicolor
5.5	2.4	Versicolor

Which species represents the query (5.7, 3.1)?

Curse of Dimensionality

- As dimensions increase:
 - **Nearest neighbors become far**
 - Most points lie in **outer shell**
 - Accurate interpolation becomes hard

k-d Trees

- **k-d Tree** = k-dimensional binary search tree
- Efficient lookup if **few dimensions & many examples**
- Splits on median values, alternates splitting dimension
- Nearest-neighbor search may need to explore both sides of split

Nonparametric regression

Methods

- **(a) Connect-the-dots** (piecewise linear): interpolates between immediate neighbors
- **(b) k-NN Average**: Mean of k nearest neighbors
- **(c) k-NN Linear Regression**: Fit line through k nearest points
- **(d) Locally Weighted Regression**: Weighted regression using **kernel functions**

Kernel-Based Regression

- **Kernel** assigns weight based on distance to query point:
$$K(\text{Distance}(x_j, x_q))$$
- Popular kernels: **Quadratic, Gaussian**
- **Kernel width** is a hyperparameter:
 - Too wide \rightarrow underfitting
 - Too narrow \rightarrow overfitting

Support Vector Machines

- **Why SVMs?**
- Once the top off-the-shelf ML algorithm (now overtaken by deep nets and forests)
- Combines advantages of **parametric** and **nonparametric** models

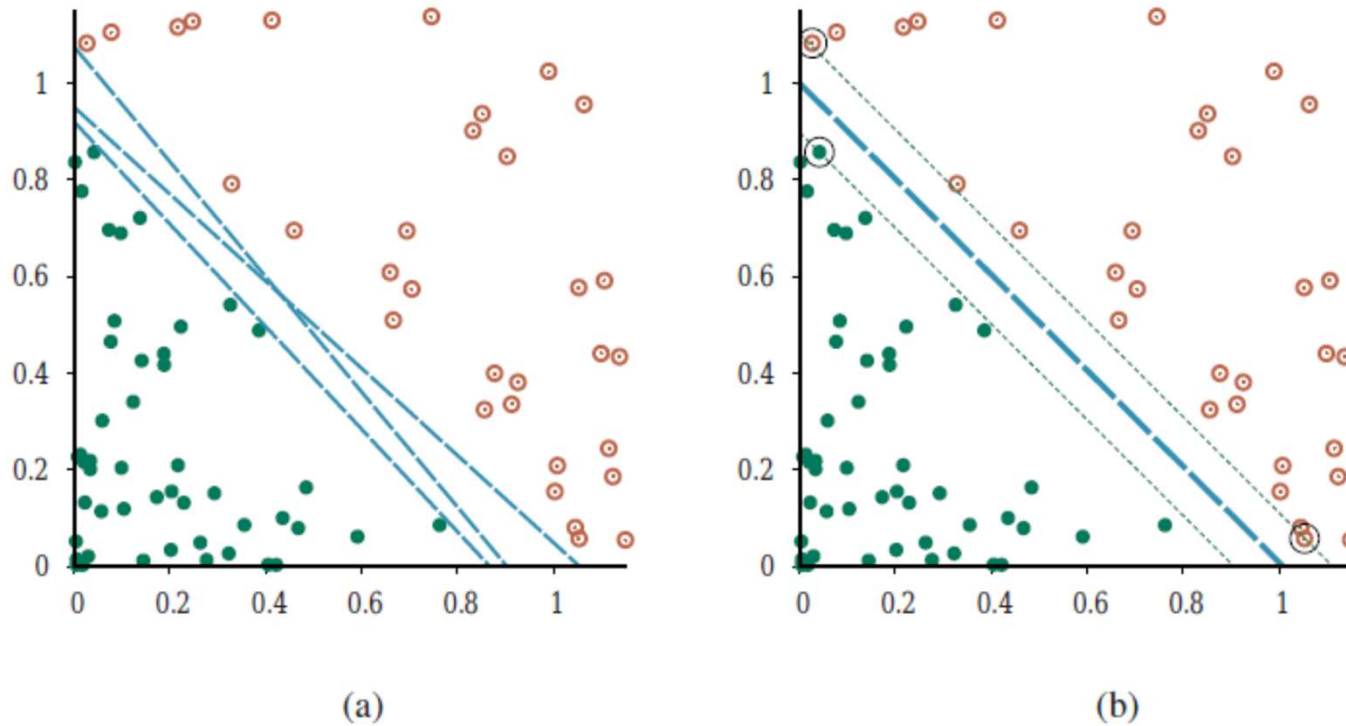


Figure 19.21 Support vector machine classification: (a) Two classes of points (orange open and green filled circles) and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the **margin** (area between dashed lines). The **support vectors** (points with large black circles) are the examples closest to the separator; here there are three.

Key Properties

1. **Maximum Margin Classifier**

1. Finds the hyperplane **farthest** from nearest training examples
2. Better generalization

2. **Kernel Trick**

1. Maps data to **higher-dimensional** space where it is linearly separable

3. **Sparse Representation**

1. Decision boundary depends only on **support vectors**, not all data

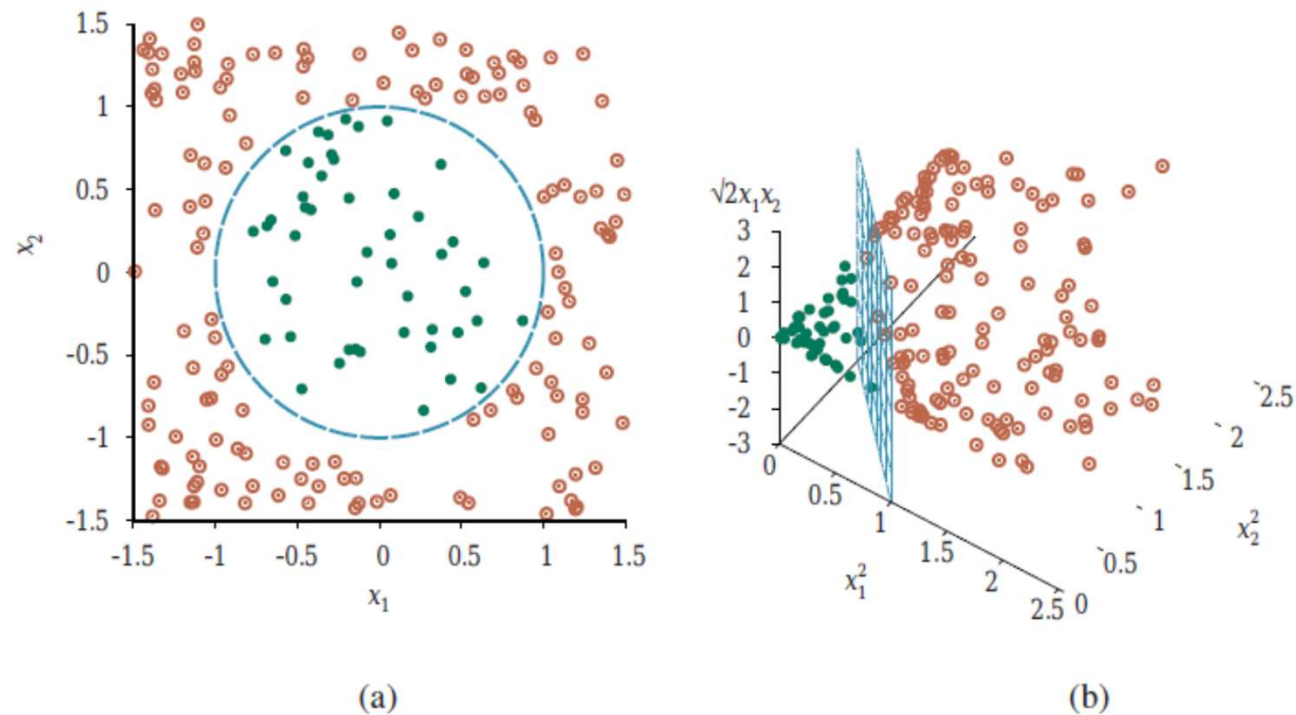


Figure 19.22 (a) A two-dimensional training set with positive examples as green filled circles and negative examples as orange open circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions. Figure 19.21(b) gives a closeup of the separator in (b).

Ensemble models

Core Concept

- Combine multiple base models to form a stronger **ensemble model**
- **Base Model:** Simple, possibly weak individual hypothesis
- **Ensemble:** Combines predictions (via voting, averaging, or another learner)

Introduction to Ensemble Learning

- Ensemble learning combines multiple hypotheses (base models) to make predictions
- Uses averaging, voting, or additional machine learning layers
- Base models form an ensemble model for improved performance
- Key benefits: Reduces bias and variance in predictions

Why Use Ensemble Learning?

- Reduces Bias:
 - Overcomes limitations of individual model restrictions
 - More expressive than single models
 - Can represent complex decision boundaries
- Reduces Variance:
 - Multiple models reduce chance of individual model errors
 - Majority voting improves accuracy
 - Reduces overfitting risk

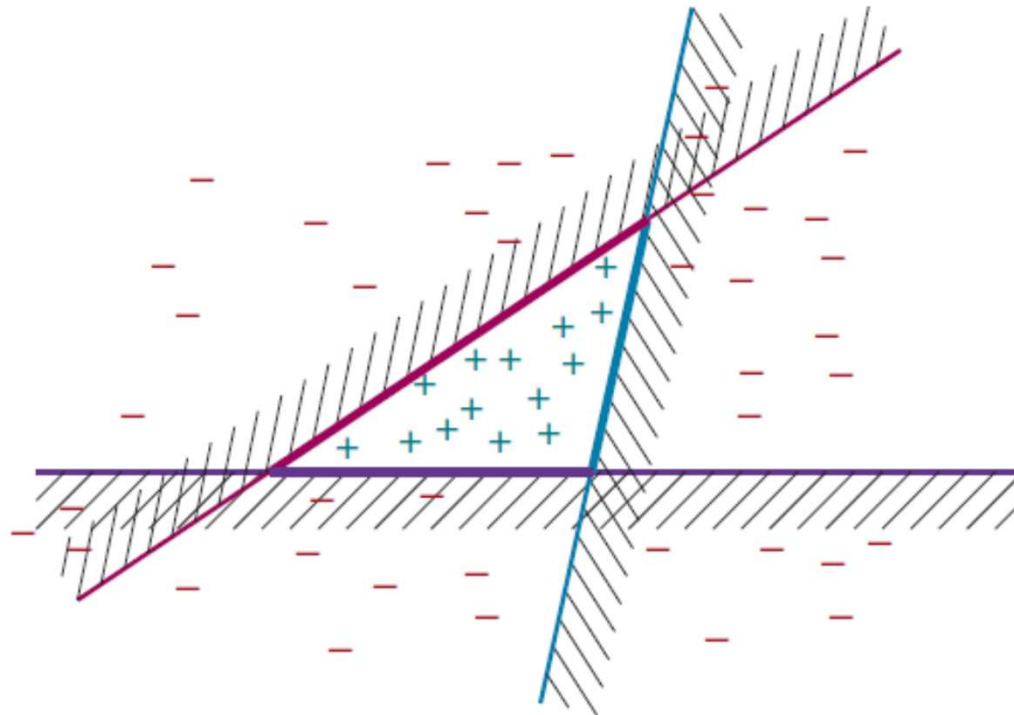


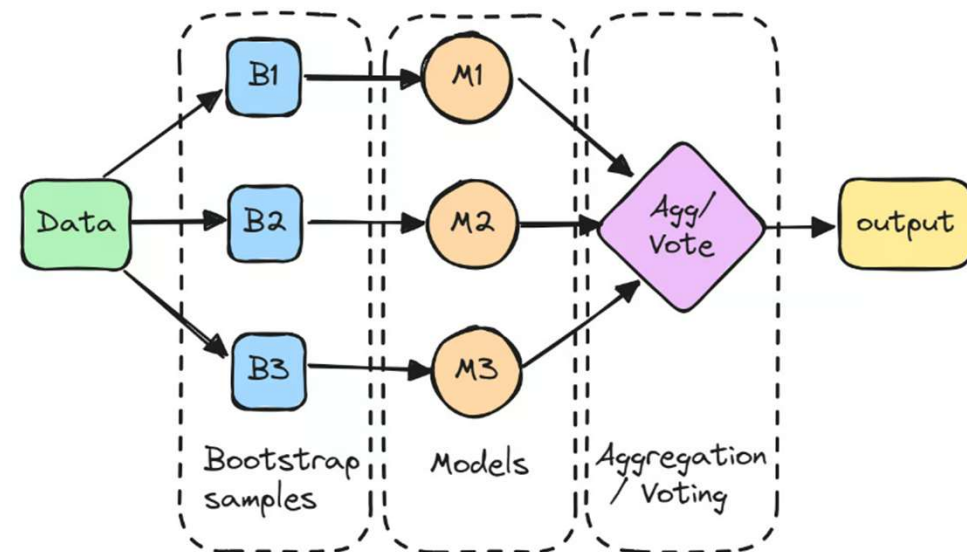
Figure 19.23 Illustration of the increased expressive power obtained by ensemble learning. We take three linear threshold hypotheses, each of which classifies positively on the unshaded side, and classify as positive any example classified positively by all three. The resulting triangular region is a hypothesis not expressible in the original hypothesis space.

Types of Ensemble Methods

- Four main approaches:
 - Bagging (Bootstrap Aggregating)
 - Random Forests
 - Stacking (Stacked Generalization)
 - Boosting
- Each method has unique characteristics and advantages

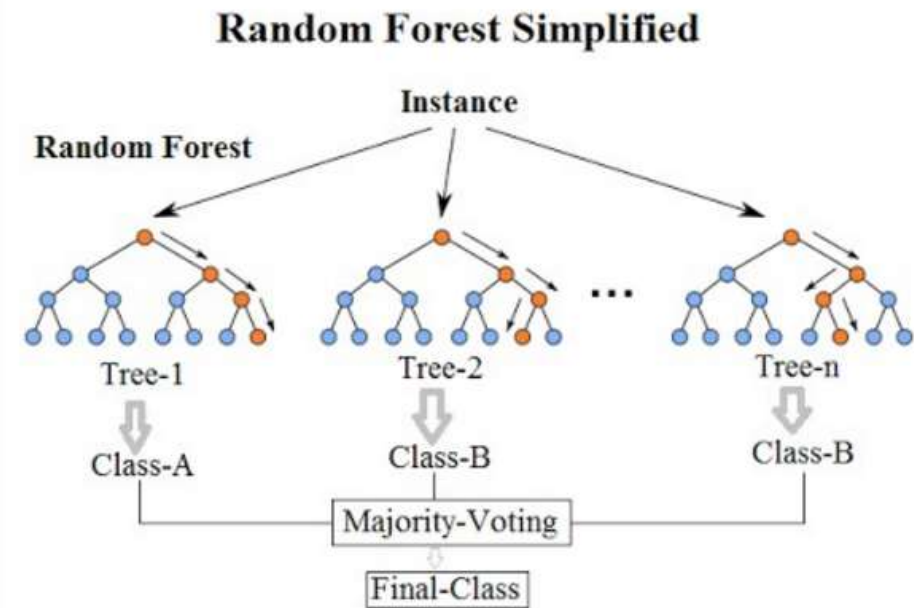
Bagging (Bootstrap Aggregating)

- Creates **K distinct training** sets by sampling with replacement
- Trains model on each set independently
- Aggregates predictions through:
 - Plurality vote (classification)
 - Average (regression)
- Particularly effective with decision trees
- Parallel processing capable



Random Forests

- Extension of bagging specifically for **decision trees**
- Introduces **randomness in attribute selection**
- At each split point:
 - Randomly selects **\sqrt{n}** attributes for classification
 - Selects **$n/3$** attributes for regression
- Uses extremely randomized trees (ExtraTrees)



ExtraTrees

In **traditional decision trees**, for **numerical features**, the algorithm:

- Sorts the values,
- Checks **every possible split point** (e.g., midpoints),
- Chooses the one with the **highest information gain**.

In **ExtraTrees**, instead of evaluating all split points:

1. It randomly picks a few **candidate split values** from the feature's range.
2. Among those random splits, it selects the one that gives the **highest information gain**.
3. This makes each tree **more diverse**, which improves the performance of the ensemble (forest).

Example

Imagine we have a feature called "**Age**" with values:

[22, 25, 28, 30, 35, 40, 45]

In a traditional tree:

- All possible midpoints like 23.5, 26.5, ..., 42.5 are tested.

In **ExtraTrees**:

- Randomly pick, say, 3 values in the range (22 to 45): **27, 33, 39**
- Compute information gain for each
- Pick the best among them

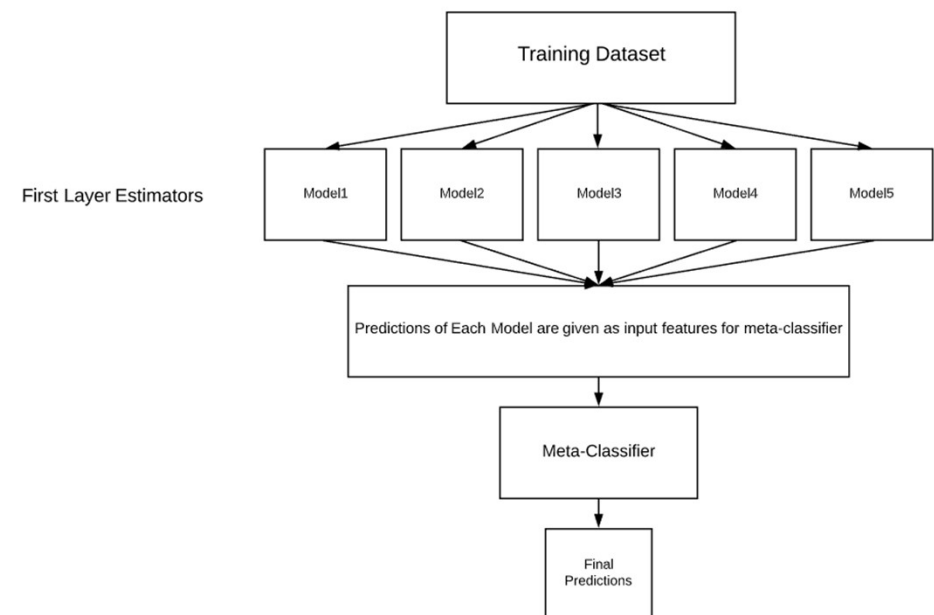
This **adds randomness** to each tree and ensures that **no two trees are exactly the same** — even with the same dataset.

Random Forest Advantages

- Resistant to overfitting
- Efficient parallel processing
- No pruning required
- Strong performance in competitions
- Applications:
 - Financial prediction
 - Medical diagnosis
 - Bioinformatics
 - Remote sensing

Stacking (Stacked Generalization)

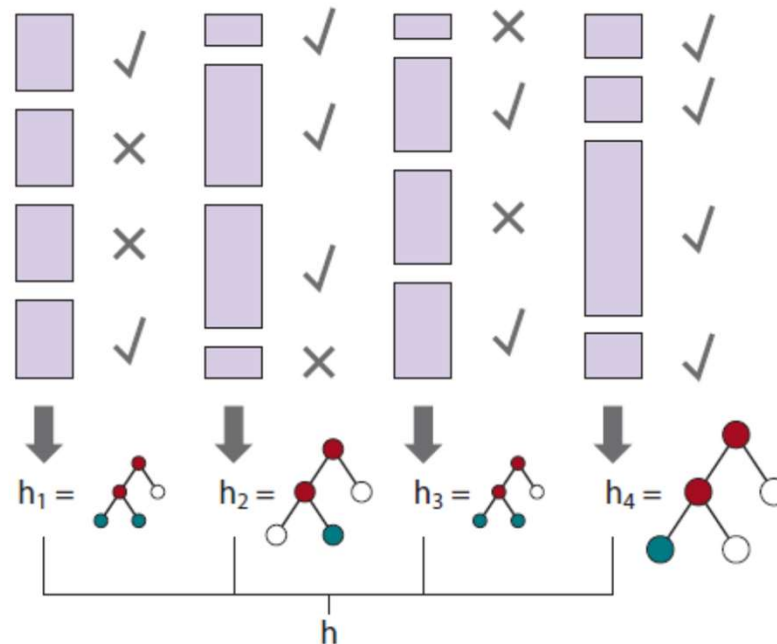
- Combines different types of models
- Process:
 - Train **multiple base models**
 - Use predictions as new features
 - Train meta-model on combined predictions
- Often outperforms individual models
- Popular in data science competitions



Boosting Fundamentals

- Sequential ensemble method
- Weighted training approach
- Each example has weight $w_j \geq 0$
- Focuses on difficult examples
- Increases weights of misclassified examples
- Decreases weights of correctly classified examples

Boosting



$$h(\mathbf{x}) = \sum_{i=1}^K z_i h_i(\mathbf{x})$$

where z_i is the weight of the i th hypothesis.

AdaBoost Algorithm

- Popular boosting implementation
- Usually applied with **decision trees** as the component hypotheses
- Creates weighted majority vote
- Properties:
 - Can achieve perfect training accuracy
 - Works with weak learners
 - Sequential processing required
 - Adaptive weighting scheme

function ADABOOST(*examples*, *L*, *K*) **returns** a hypothesis

inputs: *examples*, set of *N* labeled examples $(x_1, y_1), \dots, (x_N, y_N)$

L, a learning algorithm

K, the number of hypotheses in the ensemble

local variables: *w*, a vector of *N* example weights, initially all $1/N$

h, a vector of *K* hypotheses

z, a vector of *K* hypothesis weights

$\epsilon \leftarrow$ a small positive number, used to avoid division by zero

for *k* = 1 **to** *K* **do**

h[*k*] $\leftarrow L(\text{examples}, \mathbf{w})$

error $\leftarrow 0$

for *j* = 1 **to** *N* **do** *// Compute the total error for h[k]*

if *h*[*k*](*x*_{*j*}) $\neq y_j$ **then** *error* $\leftarrow \text{error} + \mathbf{w}[j]$

if *error* > 1/2 **then break** from loop

error $\leftarrow \min(\text{error}, 1 - \epsilon)$

for *j* = 1 **to** *N* **do** *// Give more weight to the examples h[k] got wrong*

if *h*[*k*](*x*_{*j*}) = *y*_{*j*} **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot \text{error} / (1 - \text{error})$

$\mathbf{w} \leftarrow \text{NORMALIZE}(\mathbf{w})$

z[*k*] $\leftarrow \frac{1}{2} \log((1 - \text{error}) / \text{error})$ *// Give more weight to accurate h[k]*

return *Function*(*x*) : $\sum \mathbf{z}_i \mathbf{h}_i(x)$

Gradient Boosting

- Uses gradient descent principles
 - Attention not given to specific examples, but to the gradient between the right answers and the answers given by the previous hypotheses.
- Popular implementations:
 - GBM (Gradient Boosting Machines)
 - GBRT (Gradient Boosted Regression Trees)
- Focuses on gradient between **predictions** and **actual values**
 - Parameters of the existing model not updated; parameters of the next tree updated
 - But we must do that in a way that reduces the loss by moving in the right direction along the gradient
- Highly effective for tabular data

XGBoost

- eXtreme Gradient Boosting
- Industry standard implementation
- Features:
 - Efficient memory usage
 - Parallel processing
 - Built-in regularization
 - Pruning capabilities

Online Learning

Online learning is a type of machine learning where the model is trained incrementally on a stream of data, one example at a time. This approach is particularly useful for handling large datasets, real-time data streams, and dynamic environments.

Key Concepts

1. Batch Learning

- Traditional machine learning approach where the model is trained on a fixed dataset in one go. Requires periodic retraining on entire datasets.

2. Online Learning

- Incremental training of the model on a stream of data, one example at a time.
- Enables real-time adaptation to changing data distributions.

3. Streaming Data

- Data arrives in a continuous stream, often in real-time.
- Requires online learning to handle the dynamic nature of the data.

Types of Online Learning

1. Supervised Online Learning

- The model is trained on labeled data, one example at a time.
- The goal is to minimize the loss function on the training data.

2. Unsupervised Online Learning

- The model is trained on unlabeled data, one example at a time.
- The goal is to discover patterns or structure in the data.

3. Semi-Supervised Online Learning

- The model is trained on a combination of labeled and unlabeled data.
- The goal is to leverage the benefits of both supervised and unsupervised learning.

Advantages/Challenges of Online Learning

1. Real-Time Adaptation

- The model can adapt to changing data distributions in real-time.

2. Efficiency

- Online learning can be more efficient than batch learning, especially for large datasets.

3. Scalability

- Online learning can handle large volumes of data and high data throughput.

Challenges of Online Learning

1. Concept Drift

- The underlying data distribution changes over time, affecting the model's performance.

2. Model Stability

- The model may become unstable or overfit the data if not properly regularized.

3. Data Quality

- The quality of the data can affect the model's performance and stability.

Applications of Online Learning

1. Real-Time Recommendation Systems

- Online learning can be used to build real-time recommendation systems that adapt to user behavior.

2. Fraud Detection

- Online learning can be used to detect fraudulent activities in real-time.

3. Predictive Maintenance

- Online learning can be used to predict equipment failures and maintenance needs in real-time.

Online Learning

- Handles non-i.i.d. data
- Sequential prediction process
- Receives immediate feedback
- Adapts to changing patterns
- Useful for streaming data

Weighted Majority Algorithm

Initialize a set of weights $\{w_1, \dots, w_K\}$ all to 1.

for each problem to be solved **do**

1. Receive the predictions $\{\hat{y}_1, \dots, \hat{y}_K\}$ from the experts.
2. Randomly choose an expert k^* in proportion to its weight: $P(k) = w_k$.
3. **yield** \hat{y}_{k^*} as the answer to this problem.
4. Receive the correct answer y .
5. For each expert k such that $\hat{y}_k \neq y$, update $w_k \leftarrow \beta w_k$
6. Normalize the weights so that $\sum_k w_k = 1$.

Here β is a number, $0 < \beta < 1$, that tells how much to penalize an expert for each mistake.

Introduction to Machine Learning Project Development

- Machine learning projects require a structured methodology
- Different from traditional software development
- Focus on both theoretical understanding and practical implementation
- Importance of systematic approach to ensure success

1. Problem Formulation

- Define specific user problems to solve
- Identify components suitable for ML solutions
- Establish clear loss functions
- Consider supervised, unsupervised, or reinforcement learning approaches
- Example: "Help users find photos with specific labels" vs. vague "organize photos"

Types of Learning Approaches

- Supervised Learning: Labeled data training
- Unsupervised Learning: Pattern discovery
- Semisupervised Learning: Few labeled examples + many unlabeled
- Reinforcement Learning: Action-based learning with rewards
- Weakly Supervised Learning: Noisy or imprecise labels

2. Data Collection Fundamentals

- Source identification (public datasets, user data, crowdsourcing)
- Data quantity requirements
- Transfer learning possibilities
- Privacy considerations
- Data provenance tracking
- Legal compliance

Data Quality and Management

- Error detection and correction
- Missing data handling
- Adversarial data considerations
- Terminology standardization
- Data pipeline integrity
- Regular monitoring of data feeds

Handling Data Challenges

- Unbalanced classes
- Undersampling and oversampling techniques
- Outlier detection and management
- Data augmentation methods
- SMOTE and ADASYN for synthetic data generation

Feature Engineering

- Data preprocessing techniques
- Quantization of continuous values
- One-hot encoding for categorical data
- Domain knowledge integration
- Creation of derived features
- Normalization methods

Exploratory Data Analysis (EDA)

- Visualization techniques
- Summary statistics
- Histogram and scatter plot analysis
- Cluster visualization
- Dimensionality reduction
- t-SNE (t-distributed stochastic neighbor embedding) mapping

3. Model Selection Criteria

- Random forests for categorical features
- Nonparametric methods for large datasets
- Logistic regression for linear separation
- Support vector machines for smaller datasets
- Neural networks for pattern recognition
- Ensemble methods considerations

Hyperparameter Optimization

- Experience-based selection
- Systematic search methods
- Multiple validation sets
- Avoiding overfitting during tuning
- Performance metrics tracking
- ROC curve analysis
 - Receiver operating characteristic curve **plots false positives versus true positives** for each value of the hyperparameter, helping one visualize values that would be good choices for the tradeoff.

Model Evaluation Metrics

- False positives vs. false negatives
- ROC curves and AUC
- Confusion matrices
- Computational cost considerations
- Resource utilization
- Performance-cost tradeoffs

4. Trust and Interpretability

Interpretability: We say that a machine learning model is interpretable if you can inspect the actual model and understand why it got a particular answer for a given input, and how the answer would change when the input changes.

- Model transparency
- Decision explanation capability
- Stakeholder considerations
- Regulatory compliance
- User trust building
- Accountability measures

Model Explainability

An **explainable** model is one that can help you understand “why was this output produced for this input?” In book terminology, interpretability derives from inspecting the actual model, whereas explainability can be provided by a separate process.

- LIME (Local Interpretable Model-agnostic Explanations)
- Feature importance analysis
- Black box model interpretation
- Decision tree visualization
- Linear model coefficients
- Explanation generation methods

5. Monitoring and Maintenance

- Performance dashboards
- Alert systems
- Metric tracking
- Human evaluation
- Quality assurance
- Regular updates

Dealing with Nonstationarity

- Adapting to changing patterns
- Model freshness requirements
- Update frequency determination
- Automated testing processes
- Release management
- Version control

Best Practices for Deployment

- Automated testing pipelines
- Canary deployments
- Rollback capabilities
- Performance monitoring
- Resource usage tracking
- Quality regression prevention