Machine Learning Basics

Class Lecture Slides

Cedric Kiplimo

Centre for Data Science and Artificial Intelligence

6 July 2026

Course Overview

- Welcome to the Minds, Machines and Matrices (M3) workshop!

Course Overview

- Welcome to the Minds, Machines and Matrices (M3) workshop!
- **Objective:** Designing algorithms that automatically extract patterns from data.

Course Overview

- Welcome to the Minds, Machines and Matrices (M3) workshop!
- **Objective:** Designing algorithms that automatically extract patterns from data.
- **Approach:** Concepts, foundational mathematics, and engineering trade-offs.

What is Machine Learning?

- Traditional Programming: Rules + Data $\rightarrow$ Answers.

What is Machine Learning?

- Traditional Programming: Rules + Data $\rightarrow$ Answers.
- Machine Learning: Data + Answers $\rightarrow$ Rules.

What is Machine Learning?

- Traditional Programming: Rules + Data $\rightarrow$ Answers.
- Machine Learning: Data + Answers $\rightarrow$ Rules.
- ML algorithms extract valuable patterns from data, ideally without much domain-specific expertise

What is Machine Learning?

- Traditional Programming: Rules + Data $\rightarrow$ Answers.
- Machine Learning: Data + Answers $\rightarrow$ Rules.
- ML algorithms extract valuable patterns from data, ideally without much domain-specific expertise
- The goal is generalisation to unseen data environments.

Three Pillars of Machine Learning

Every machine learning system rests on three fundamental pillars:

1. Data

Three Pillars of Machine Learning

Every machine learning system rests on three fundamental pillars:

1. Data
2. Model

Three Pillars of Machine Learning

Every machine learning system rests on three fundamental pillars:

1. Data
2. Model
3. Learning

Pillar 1: Data

What constitutes data in the context of ML?

- The fuel of the machine learning system.

Pillar 1: Data

What constitutes data in the context of ML?

- The fuel of the machine learning system.
- Represented typically as numerical vectors.

Pillar 1: Data

What constitutes data in the context of ML?

- The fuel of the machine learning system.
- Represented typically as numerical vectors.
- Composed of individual instances and distinct features.

Pillar 1: Data

What constitutes data in the context of ML?

- The fuel of the machine learning system.
- Represented typically as numerical vectors.
- Composed of individual instances and distinct features.
- Results from some real but unknown data-generating process

Pillar 1: Data

What constitutes data in the context of ML?

- The fuel of the machine learning system.
- Represented typically as numerical vectors.
- Composed of individual instances and distinct features.
- Results from some real but unknown data-generating process
- Examples: Age, Income, Pixel intensities, Word embeddings.

Pillar 2: The Model

What is the model's role?

- The underlying mathematical architecture.

Pillar 2: The Model

What is the model's role?

- The underlying mathematical architecture.
- A mathematical function mapping inputs to predicted outputs.

Pillar 2: The Model

What is the model's role?

- The underlying mathematical architecture.
- A mathematical function mapping inputs to predicted outputs.
- Simplified version of the real data-generating process

Pillar 3: Learning

How does the system actually learn?

- The core optimisation phase.

Pillar 3: Learning

How does the system actually learn?

- The core optimisation phase.
- Find patterns and structure in data by optimising the model parameters.

Pillar 3: Learning

How does the system actually learn?

- The core optimisation phase.
- Find patterns and structure in data by optimising the model parameters.
- Minimise empirical loss, maximise a performance metric, and strive towards generalisation.

Mathematical Communication

- Machine learning requires a precise language.

Mathematical Communication

- Machine learning requires a precise language.
- Consistent notation prevents systemic confusion.

Mathematical Communication

- Machine learning requires a precise language.
- Consistent notation prevents systemic confusion.
- Let us baseline our linear algebra and matrix notations.

Notation: Scalars

- Represented by lowercase, italicized letters (a, b, c).

Notation: Scalars

- Represented by lowercase, italicized letters (a, b, c).
- Denotes single numerical values.

Notation: Scalars

- Represented by lowercase, italicized letters (a, b, c).
- Denotes single numerical values.
- Examples: Learning rate (α), individual weights, indexing counters.

Notation: Vectors

- Represented by bold lowercase letters ($\mathbf{x}, \mathbf{y}, \mathbf{w}$).

Notation: Vectors

- Represented by bold lowercase letters ($\mathbf{x}, \mathbf{y}, \mathbf{w}$).
- Denotes a single instance, data point, or weight sequence.

Notation: Vectors

- Represented by bold lowercase letters ($\mathbf{x}, \mathbf{y}, \mathbf{w}$).
- Denotes a single instance, data point, or weight sequence.
- Assumed to be column vectors by default in standard operations.

Notation: Matrices

- Represented by bold uppercase letters ($\mathbf{A}, \mathbf{X}, \mathbf{W}$).

Notation: Matrices

- Represented by bold uppercase letters ($\mathbf{A}, \mathbf{X}, \mathbf{W}$).
- Denotes the complete design matrix or collections of features.

Notation: Matrices

- Represented by bold uppercase letters ($\mathbf{A}, \mathbf{X}, \mathbf{W}$).
- Denotes the complete design matrix or collections of features.
- Rows typically represent samples; columns represent attributes.

Notation: Matrix Operations

- $\mathbf{x}^T$: Transpose operations.

Notation: Matrix Operations

- $\mathbf{x}^T$: Transpose operations.
- Flips rows and columns systematically.

Notation: Matrix Operations

- $\mathbf{x}^T$: Transpose operations.
- Flips rows and columns systematically.
- Crucial for calculating inner products ($\mathbf{w}^T\mathbf{x}$).

Notation: Distance Metrics

- $\|\cdot\|$: The Norm operation.

Notation: Distance Metrics

- $\|\cdot\|$: The Norm operation.
- Typically denotes the standard Euclidean norm.

Notation: Distance Metrics

- $\|\cdot\|$: The Norm operation.
- Typically denotes the standard Euclidean norm.
- Used for measuring magnitudes or distance between target vectors.

Mathematical Notation Summary

Symbol	Meaning	Contextual Example
a, b	Scalars	Learning rate, regularisers
$\mathbf{x}, \mathbf{y}$	Vectors	Single instance, target array
$\mathbf{A}, \mathbf{X}$	Matrices	Complete Data/Design Matrix
$\mathbf{x}^T$	Transpose	Row-column geometric inversion
$\|\| \cdot \|\|$	Norm	Euclidean distance magnitude

Translating Reality to Math

- Nature contains hidden, complex relationships between inputs and outputs.

Translating Reality to Math

- Nature contains hidden, complex relationships between inputs and outputs.
- We attempt to capture these interactions formally.

Translating Reality to Math

- Nature contains hidden, complex relationships between inputs and outputs.
- We attempt to capture these interactions formally.
- We assume an underlying true data-generating mechanism.

The Fundamental True Formula

We formalise natural phenomena using the following assumed functional relationship:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$$

The Fundamental True Formula

We formalise natural phenomena using the following assumed functional relationship:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$$

- $\mathbf{y}$: The observed real-world output.

The Fundamental True Formula

We formalise natural phenomena using the following assumed functional relationship:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$$

- $\mathbf{y}$: The observed real-world output.
- $\mathbf{x}$: The measurable inputs (features).

The Fundamental True Formula

We formalise natural phenomena using the following assumed functional relationship:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$$

- $\mathbf{y}$: The observed real-world output.
- $\mathbf{x}$: The measurable inputs (features).
- $f(\mathbf{x})$: The true, hidden mapping function.

Understanding Irreducible Noise

Focusing on the components of: $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$

- $\boldsymbol{\epsilon}$: The error term.

Understanding Irreducible Noise

Focusing on the components of: $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$

- $\boldsymbol{\epsilon}$: The error term.
- Defined explicitly as **irreducible noise**.

Understanding Irreducible Noise

Focusing on the components of: $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$

- $\boldsymbol{\epsilon}$: The error term.
- Defined explicitly as **irreducible noise**.
- Caused by unmeasured variables, random chance, or measurement errors.

Understanding Irreducible Noise

Focusing on the components of: $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$

- $\boldsymbol{\epsilon}$: The error term.
- Defined explicitly as **irreducible noise**.
- Caused by unmeasured variables, random chance, or measurement errors.
- Even a perfect model cannot predict or eliminate $\boldsymbol{\epsilon}$.

The Practical Prediction Formula

Because $f(\mathbf{x})$ is hidden, our actual engineering goal in ML is to construct $\hat{f}(\mathbf{x})$.

The Practical Prediction Formula

Because $f(\mathbf{x})$ is hidden, our actual engineering goal in ML is to construct $\hat{f}(\mathbf{x})$.

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$$

The Practical Prediction Formula

Because $f(\mathbf{x})$ is hidden, our actual engineering goal in ML is to construct $\hat{f}(\mathbf{x})$.

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$$

- $\hat{\mathbf{y}}$: Our predicted output.

The Practical Prediction Formula

Because $f(\mathbf{x})$ is hidden, our actual engineering goal in ML is to construct $\hat{f}(\mathbf{x})$.

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$$

- $\hat{\mathbf{y}}$: Our predicted output.
- $\hat{f}$: Our estimated approximation function.

The Core Structure of Data

- How do we group and feed features to the model?

The Core Structure of Data

- How do we group and feed features to the model?
- We structure objects using a data frame framework: $(\mathbf{X}, \mathbf{y})$.

The Core Structure of Data

- How do we group and feed features to the model?
- We structure objects using a data frame framework: $(\mathbf{X}, \mathbf{y})$.
- Let's review the precise spatial configuration.

Data Layout Example

Sample (i)	Feature 1 (Age)	Feature 2 (Income)	Target (y)
1	25	40,000	680 (Score)
2	42	85,000	720 (Score)
3	19	12,000	580 (Score)

- In this example, $\mathbf{X} \in \mathbb{R}^{3\times 2}$ and $\mathbf{y} \in \mathbb{R}^3$

Deep Dive: Matrix $\mathbf{X}$

- Organized as an $n \times d$ grid structure.

Deep Dive: Matrix $\mathbf{X}$

- Organized as an $n \times d$ grid structure.
- n: The absolute number of samples/instances available.

Deep Dive: Matrix $\mathbf{X}$

- Organized as an $n \times d$ grid structure.
- n: The absolute number of samples/instances available.
- d: The dimensionality or total feature count.

Deep Dive: Matrix $\mathbf{X}$

- Organized as an $n \times d$ grid structure.
- n: The absolute number of samples/instances available.
- d: The dimensionality or total feature count.
- Rows map to instances; columns map to unique attributes (features).

Deep Dive: Vector $\mathbf{y}$

- Structured as a column vector of total length n.

Deep Dive: Vector $\mathbf{y}$

- Structured as a column vector of total length n.
- Holds the ground-truth target values.

Deep Dive: Vector $\mathbf{y}$

- Structured as a column vector of total length n.
- Holds the ground-truth target values.
- In supervised setups, this provides our validation check.

Dials of the Model

- Models possess configuration settings that dictate behavior.

Dials of the Model

- Models possess configuration settings that dictate behavior.
- These configurations split into two distinct categories.

Dials of the Model

- Models possess configuration settings that dictate behavior.
- These configurations split into two distinct categories.
- Category 1: Internal Dials (Parameters).

Dials of the Model

- Models possess configuration settings that dictate behavior.
- These configurations split into two distinct categories.
- Category 1: Internal Dials (Parameters).
- Category 2: External Setup Dials (Hyperparameters).

Model Parameters

- Defined as the internal weights configuration.

Model Parameters

- Defined as the internal weights configuration.
- Denoted mathematically as (w, b).

Model Parameters

- Defined as the internal weights configuration.
- Denoted mathematically as (w, b).
- These are explicitly **learned** from the data.

Model Parameters

- Defined as the internal weights configuration.
- Denoted mathematically as (w, b).
- These are explicitly **learned** from the data.
- Updated automatically by optimisation algorithms during training.

Model Hyperparameters

- Defined as the foundational architectural settings.

Model Hyperparameters

- Defined as the foundational architectural settings.
- These are explicitly **chosen by the engineer**.

Model Hyperparameters

- Defined as the foundational architectural settings.
- These are explicitly **chosen by the engineer**.
- Must be fixed before a single training run executes.

Model Hyperparameters

- Defined as the foundational architectural settings.
- These are explicitly **chosen by the engineer**.
- Must be fixed before a single training run executes.
- Examples: Learning rate (α), maximum tree depth, batch size.

Parameters vs. Hyperparameters Rule

"We learn parameters; we search for hyperparameters."

Learning: The Three-Step Loop

The standard training progression is a continuous cycle:

1. Predict

Learning: The Three-Step Loop

The standard training progression is a continuous cycle:

1. Predict
2. Evaluate

Learning: The Three-Step Loop

The standard training progression is a continuous cycle:

1. Predict
2. Evaluate
3. Update

Step 1: Predict

- Termed the "Forward Pass" sequence.

Step 1: Predict

- Termed the "Forward Pass" sequence.
- The model ingests data matrix $\mathbf{X}$ using current weight sets.

Step 1: Predict

- Termed the "Forward Pass" sequence.
- The model ingests data matrix $\mathbf{X}$ using current weight sets.
- Produces the current estimate output vector: $\hat{\mathbf{y}}$.

Step 2: Evaluate

- Requires a designated Loss Function: $L(\mathbf{y}, \hat{\mathbf{y}})$.

Step 2: Evaluate

- Requires a designated Loss Function: $L(\mathbf{y}, \hat{\mathbf{y}})$.
- Measures the delta between targets and estimates.

Step 2: Evaluate

- Requires a designated Loss Function: $L(\mathbf{y}, \hat{\mathbf{y}})$.
- Measures the delta between targets and estimates.
- Answers the question: How wrong are our current predictions?

Step 3: Update

- Triggers the optimisation engine.

Step 3: Update

- Triggers the optimisation engine.
- Calculates gradients against the error surface.

Step 3: Update

- Triggers the optimisation engine.
- Calculates gradients against the error surface.
- Tweaks internal parameters (w, b) to minimise upcoming loss values.

Visualising the Loop Optimisation

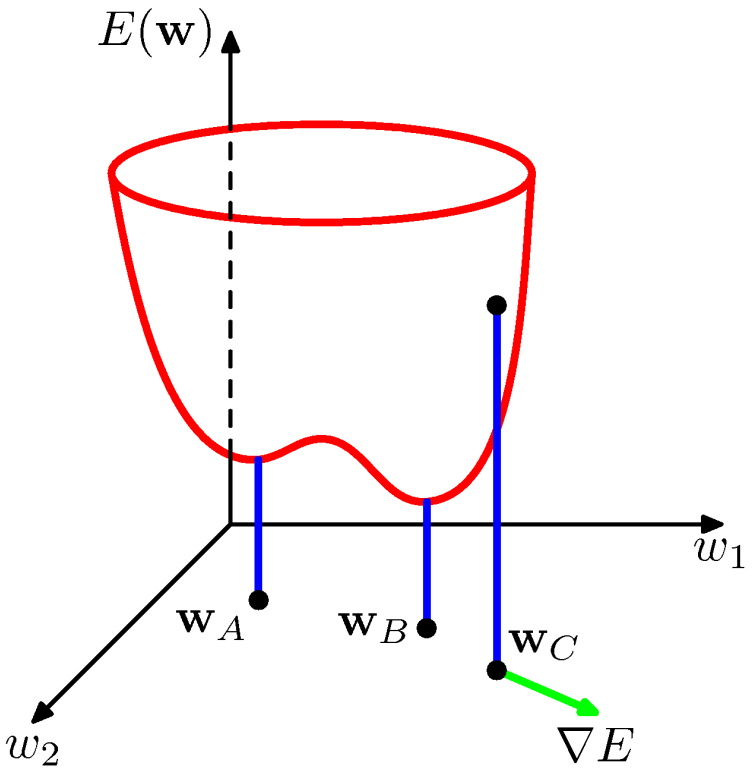

Figure: The error function $E(w)$ as a surface sitting over weight space. (Source: Bishop PRML)

Types of Machine Learning

- Algorithms are categorised by their data access profiles.

Types of Machine Learning

- Algorithms are categorised by their data access profiles.
- Paradigm 1: Supervised Learning.

Types of Machine Learning

- Algorithms are categorised by their data access profiles.
- Paradigm 1: Supervised Learning.
- Paradigm 2: Unsupervised Learning.

Types of Machine Learning

- Algorithms are categorised by their data access profiles.
- Paradigm 1: Supervised Learning.
- Paradigm 2: Unsupervised Learning.
- Paradigm 3: Frontier Frameworks.

Supervised Learning Overview

- Core property: Learning with an explicit "Teacher".

Supervised Learning Overview

- Core property: Learning with an explicit "Teacher".
- Ground-truth answers ($\mathbf{y}$) are present for every training item.

Supervised Learning Overview

- Core property: Learning with an explicit "Teacher".
- Ground-truth answers ($\mathbf{y}$) are present for every training item.
- Tasks fall into either Regression or Classification profiles.

Supervised Task: Regression

- Deals with prediction of continuous target values.

Supervised Task: Regression

- Deals with prediction of continuous target values.
- Model outputs real-valued continuous scalars.

Supervised Task: Regression

- Deals with prediction of continuous target values.
- Model outputs real-valued continuous scalars.
- Examples: Forecasting housing market prices, weather temperatures.

Supervised Task: Classification

- Deals with prediction discrete target categories.

Supervised Task: Classification

- Deals with prediction discrete target categories.
- Model assigns individual data observations into class buckets.

Supervised Task: Classification

- Deals with prediction discrete target categories.
- Model assigns individual data observations into class buckets.
- Examples: Email spam detection, MNIST handwritten digit grouping.

Visualising Labelled Data Profiles

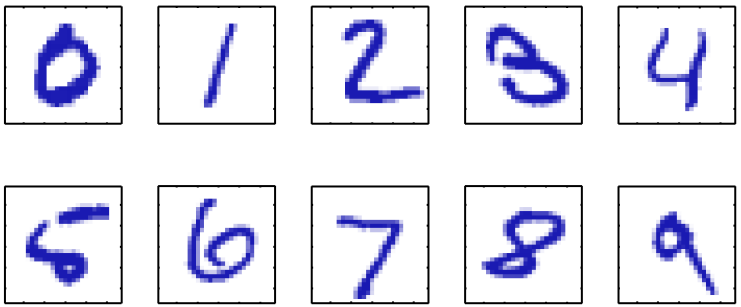

Figure: Examples of hand-written digits from the MNIST dataset (Source: Bishop PRML)

Unsupervised Learning Overview

- Core property: Learning via automatic discovery.

Unsupervised Learning Overview

- Core property: Learning via automatic discovery.
- No target labels ($\mathbf{y}$) are present in the dataset.

Unsupervised Learning Overview

- Core property: Learning via automatic discovery.
- No target labels ($\mathbf{y}$) are present in the dataset.
- Goal: Uncover hidden structural configurations within the data.

Unsupervised Task: Clustering

- Automatically grouping similar instances together.

Unsupervised Task: Clustering

- Automatically grouping similar instances together.
- Maximises intra-cluster similarity; minimizes inter-cluster similarity.

Unsupervised Task: Clustering

- Automatically grouping similar instances together.
- Maximises intra-cluster similarity; minimizes inter-cluster similarity.
- Example: Running targeted customer market segmentation.

Unsupervised Task: Dimensionality Reduction

- Compressing complex data matrices down.

Unsupervised Task: Dimensionality Reduction

- Compressing complex data matrices down.
- Strips away redundant or noisy feature columns.

Unsupervised Task: Dimensionality Reduction

- Compressing complex data matrices down.
- Strips away redundant or noisy feature columns.
- Retains maximum structural information in a lower dimensional state.

Unsupervised Task: Anomaly Detection

- Identifying unusual data points that don't match standard profiles.

Unsupervised Task: Anomaly Detection

- Identifying unusual data points that don't match standard profiles.
- Used extensively in banking systems to discover fraud patterns.

Unsupervised Task: Anomaly Detection

- Identifying unusual data points that don't match standard profiles.
- Used extensively in banking systems to discover fraud patterns.
- Credit card fraud detection

Frontier Paradigms: Self-Supervised

- Blurs lines between supervised and unsupervised setups.

Frontier Paradigms: Self-Supervised

- Blurs lines between supervised and unsupervised setups.
- Generates internal labels automatically from raw unlabelled files.

Frontier Paradigms: Self-Supervised

- Blurs lines between supervised and unsupervised setups.
- Generates internal labels automatically from raw unlabelled files.
- Example: Large Language Models (LLMs) hiding words to predict the next token.

Frontier Paradigms: Reinforcement Learning

- Operationalised through a systematic action-reward feedback loop.

Frontier Paradigms: Reinforcement Learning

- Operationalised through a systematic action-reward feedback loop.
- An autonomous Agent interacts continuously with an Environment.

Frontier Paradigms: Reinforcement Learning

- Operationalised through a systematic action-reward feedback loop.
- An autonomous Agent interacts continuously with an Environment.
- Learns optimal paths via scalar feedback signals.

Frontier Paradigms: Reinforcement Learning

- Operationalised through a systematic action-reward feedback loop.
- An autonomous Agent interacts continuously with an Environment.
- Learns optimal paths via scalar feedback signals.
- Examples: Robotics, AlphaGo.

Data Preprocessing & Feature Engineering

- Raw data is rarely ready for model ingestion.

Data Preprocessing & Feature Engineering

- Raw data is rarely ready for model ingestion.
- Quality of preprocessing directly limits model performance.

Data Preprocessing & Feature Engineering

- Raw data is rarely ready for model ingestion.
- Quality of preprocessing directly limits model performance.
- Key Objective: Transforming real-world data into clean numerical vectors.

Feature Scaling

- Features often have vastly different ranges (e.g., Age vs. Income).

Feature Scaling

- Features often have vastly different ranges (e.g., Age vs. Income).
- Unscaled features cause optimization algorithms (like Gradient Descent) to oscillate.

Feature Scaling

- Features often have vastly different ranges (e.g., Age vs. Income).
- Unscaled features cause optimization algorithms (like Gradient Descent) to oscillate.
- **Standardization (Z-score normalization):** Scales data to have zero mean and unit variance.

Feature Scaling

- Features often have vastly different ranges (e.g., Age vs. Income).
- Unscaled features cause optimization algorithms (like Gradient Descent) to oscillate.
- **Standardization (Z-score normalization):** Scales data to have zero mean and unit variance.
- **Min-Max Scaling:** Compresses features strictly between a $[0, 1]$ boundary.

Handling Categorical Variables

- Algorithms natively require continuous numerical numeric values.

Handling Categorical Variables

- Algorithms natively require continuous numerical numeric values.
- **Label Encoding:** Assigns a unique integer to each category label.

Handling Categorical Variables

- Algorithms natively require continuous numerical numeric values.
- **Label Encoding:** Assigns a unique integer to each category label.
- Risk: Introduces an artificial, non-existent mathematical order.

Handling Categorical Variables

- Algorithms natively require continuous numerical numeric values.
- **Label Encoding:** Assigns a unique integer to each category label.
- Risk: Introduces an artificial, non-existent mathematical order.
- **One-Hot Encoding:** Creates binary columns for every individual category.

Handling Categorical Variables

- Algorithms natively require continuous numerical numeric values.
- **Label Encoding:** Assigns a unique integer to each category label.
- Risk: Introduces an artificial, non-existent mathematical order.
- **One-Hot Encoding:** Creates binary columns for every individual category.
- Solves the ordering issue but increases matrix dimensionality.

Preventing Leakage

- Strict separation of data partitions is mandatory.

Preventing Leakage

- Strict separation of data partitions is mandatory.
- Standard split: Training set, Validation set, and Testing set.

Preventing Leakage

- Strict separation of data partitions is mandatory.
- Standard split: Training set, Validation set, and Testing set.
- **Data Leakage:** When information from outside the training set is accidentally used to train the model.

Preventing Leakage

- Strict separation of data partitions is mandatory.
- Standard split: Training set, Validation set, and Testing set.
- **Data Leakage:** When information from outside the training set is accidentally used to train the model.
- Rule: Compute preprocessing parameters (like mean and scaling factors) **only** on the training split.

Quiz

Question

If we want to build an ML model to predict whether a customer will buy a product based on their 'Account Balance' (numeric) and 'Country of Residence' (categorical), what preprocessing steps must we take before hitting the train button?

Quiz

Question

If we want to build an ML model to predict whether a customer will buy a product based on their 'Account Balance' (numeric) and 'Country of Residence' (categorical), what preprocessing steps must we take before hitting the train button?

Answer: One-hot encode the Country column, scale the Account Balance column (since balances can vary by thousands), and split the dataset into Train/Val/Test before fitting any scalers.

The Fundamental Tradeoff

- How do we decompose a model's prediction errors?

The Fundamental Tradeoff

- How do we decompose a model's prediction errors?
- Total prediction error has three components: bias, variance, and noise.

The Fundamental Tradeoff

- How do we decompose a model's prediction errors?
- Total prediction error has three components: bias, variance, and noise.
- The following identity captures their contribution.

The Error Decomposition Identity

Expected generalisation error can be broken down mathematically:

$$\text{Total Expected Error} = \text{Bias}^2 + \text{Variance} + \epsilon$$

The Error Decomposition Identity

Expected generalisation error can be broken down mathematically:

$$\text{Total Expected Error} = \text{Bias}^2 + \text{Variance} + \epsilon$$

- Bias^2: Errors due to flawed model assumptions.

The Error Decomposition Identity

Expected generalisation error can be broken down mathematically:

$$\text{Total Expected Error} = \text{Bias}^2 + \text{Variance} + \epsilon$$

- Bias^2: Errors due to flawed model assumptions.
- Variance: Errors stemming from high sensitivity to small training variations.

The Error Decomposition Identity

Expected generalisation error can be broken down mathematically:

$$\text{Total Expected Error} = \text{Bias}^2 + \text{Variance} + \epsilon$$

- Bias^2: Errors due to flawed model assumptions.
- Variance: Errors stemming from high sensitivity to small training variations.
- ϵ: Irreducible ambient noise.

Analysing High Bias

- Corresponds directly to **Under-fitting**.

Analysing High Bias

- Corresponds directly to **Under-fitting**.
- Model introduces overly simplistic assumptions.

Analysing High Bias

- Corresponds directly to **Under-fitting**.
- Model introduces overly simplistic assumptions.
- Fails to capture meaningful patterns present in the training data.

Analysing High Bias

- Corresponds directly to **Under-fitting**.
- Model introduces overly simplistic assumptions.
- Fails to capture meaningful patterns present in the training data.
- Performance is systematically poor on both training and validation sets.

Analysing High Variance

- Corresponds directly to **Over-fitting**.

Analysing High Variance

- Corresponds directly to **Over-fitting**.
- Model memorises specific details and random noise within the training set.

Analysing High Variance

- Corresponds directly to **Over-fitting**.
- Model memorises specific details and random noise within the training set.
- Fails to generalise effectively to unseen data instances.

Analysing High Variance

- Corresponds directly to **Over-fitting**.
- Model memorises specific details and random noise within the training set.
- Fails to generalise effectively to unseen data instances.
- Training loss is low, but validation loss spikes dangerously.

Finding Generalisation

- The engineering goal is seeking the optimal balance point.

Finding Generalisation

- The engineering goal is seeking the optimal balance point.
- Minimising both components simultaneously is mathematically constrained.

Finding Generalisation

- The engineering goal is seeking the optimal balance point.
- Minimising both components simultaneously is mathematically constrained.
- Target: The Sweet Spot minimising validation curve metrics.

Bias-Variance Trade-off Example: Polynomial Curve Fitting

- Original function is $t(x) = \sin(2\pi x)$ with x drawn from $[0, 1]$ with some Gaussian noise added.

Bias-Variance Trade-off Example: Polynomial Curve Fitting

- Original function is $t(x) = \sin(2\pi x)$ with x drawn from $[0, 1]$ with some Gaussian noise added.
- We desire to fit the data using a polynomial of the form:

$$y(x, \mathbf{w}) = w_0 + w_1 + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Bias-Variance Trade-off Example: Polynomial Curve Fitting

- Original function is $t(x) = \sin(2\pi x)$ with x drawn from $[0, 1]$ with some Gaussian noise added.
- We desire to fit the data using a polynomial of the form:

$$y(x, \mathbf{w}) = w_0 + w_1 + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- We fit by minimising the delta between the value of $y(x, \mathbf{w})$ and the original ground truth data.
- Captured by the error (loss) function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \Big[y(x_n, \mathbf{w}) - t_n \Big]^2$$

Bias-Variance Trade-off Example: Polynomial Curve Fitting

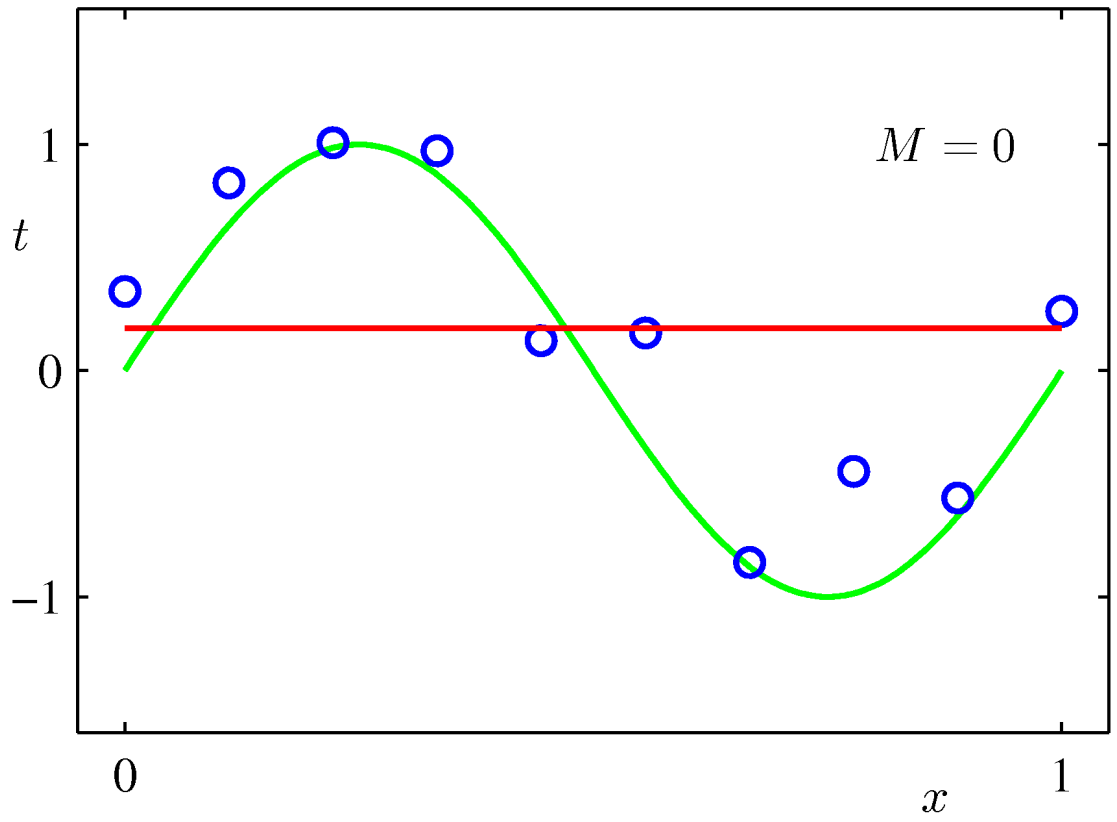

Figure: Plots of polynomial having order $M = 0$ shown as a red curve. Results in **High Bias** or **Under-fitting** (Source: Bishop PRML)

Bias-Variance Trade-off Example: Polynomial Curve Fitting

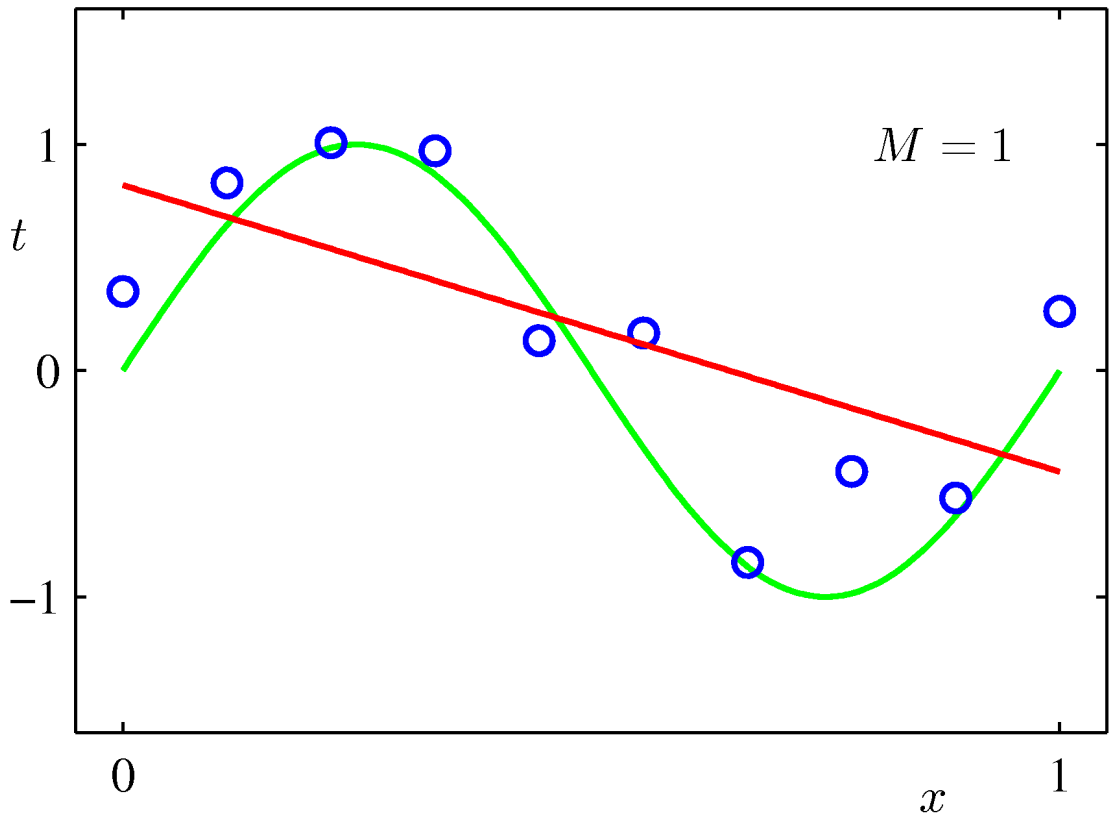

Figure: Plots of polynomial having order $M = 1$ shown as a red curve. Results in **High Bias** or **Under-fitting** (Source: Bishop PRML)

Bias-Variance Trade-off Example: Polynomial Curve Fitting

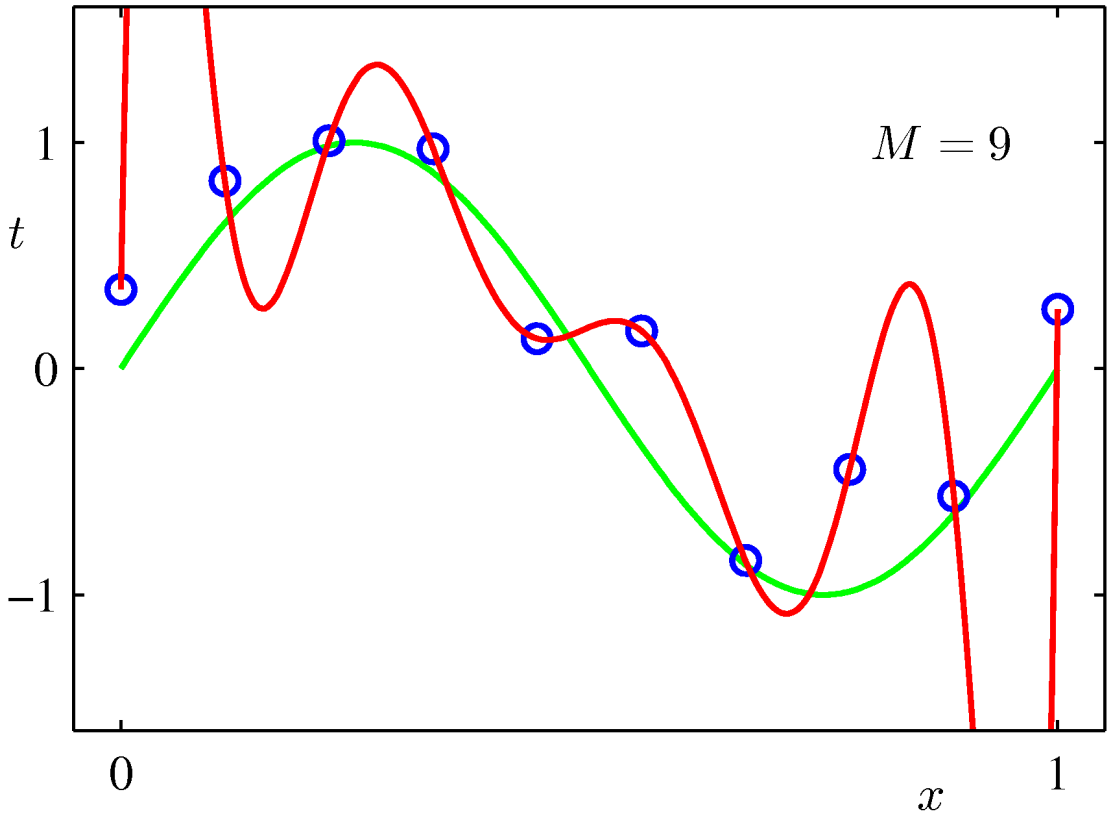

Figure: Plots of polynomials having order $M = 9$ shown as a red curve. Results in **High Variance** or **Over-fitting** (Source: Bishop PRML)

Bias-Variance Trade-off Example: Polynomial Curve Fitting

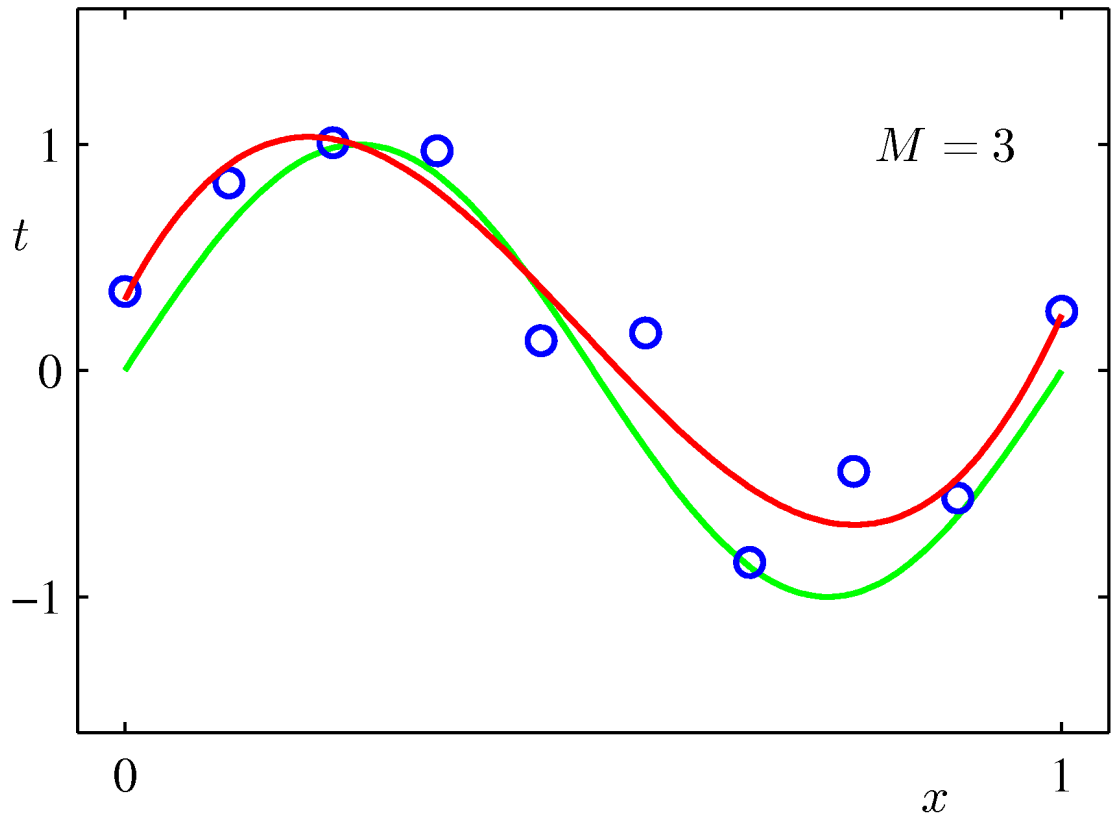

Figure: Plots of polynomials having order $M = 3$ shown as a red curve. Results in the best fit or **Generalisation** (Source: Bishop PRML)

Visualising the Error Curves

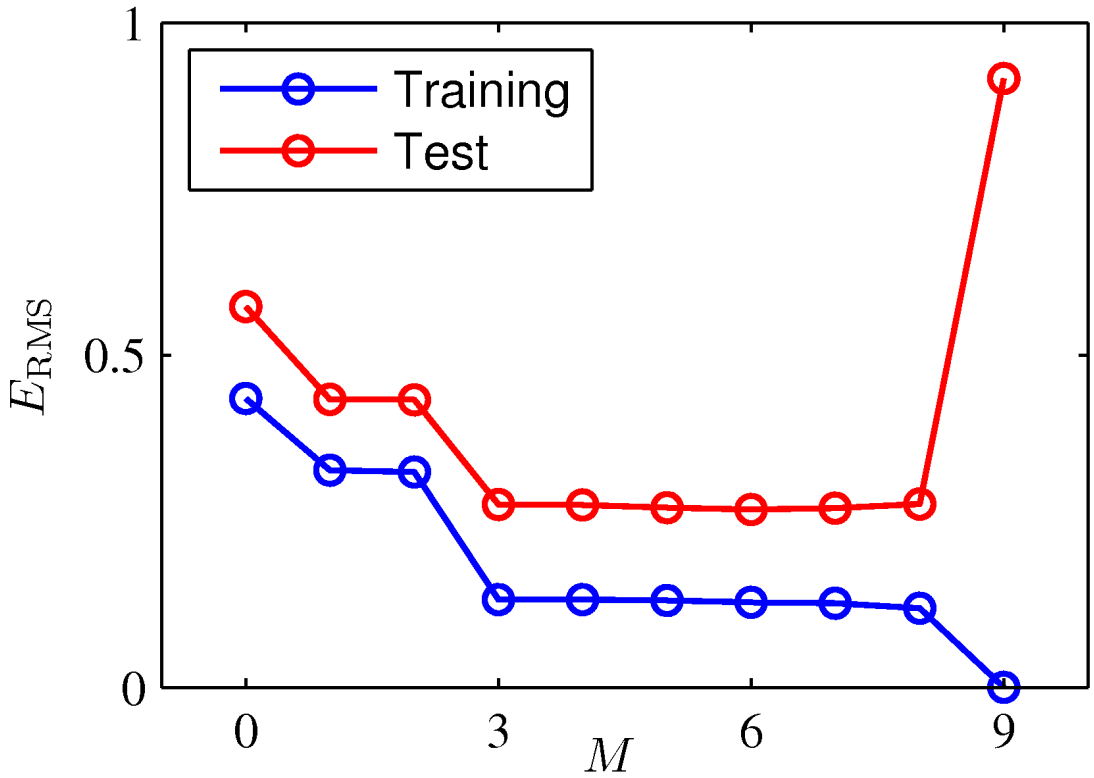

Figure: Training and test error for various values of M. (Source: Bishop PRML)

The Confusion Matrix

- Simple accuracy can be deceptive on imbalanced datasets.

The Confusion Matrix

- Simple accuracy can be deceptive on imbalanced datasets.
- We must evaluate systems using more granular metrics.

The Confusion Matrix

- Simple accuracy can be deceptive on imbalanced datasets.
- We must evaluate systems using more granular metrics.
- These metrics trace back to the **Confusion Matrix**.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	**True Positive (TP)**	False Negative (FN)
	Negative	False Positive (FP)	**True Negative (TN)**

The Matrix Quadrants

Our evaluation terms depend on four underlying outcomes from the confusion matrix:

1. True Positives (TP)

The Matrix Quadrants

Our evaluation terms depend on four underlying outcomes from the confusion matrix:

1. True Positives (TP)
2. True Negatives (TN)

The Matrix Quadrants

Our evaluation terms depend on four underlying outcomes from the confusion matrix:

1. True Positives (TP)
2. True Negatives (TN)
3. False Positives (FP)

The Matrix Quadrants

Our evaluation terms depend on four underlying outcomes from the confusion matrix:

1. True Positives (TP)
2. True Negatives (TN)
3. False Positives (FP)
4. False Negatives (FN)

Precision

What is the core intuition of Precision?

Precision

What is the core intuition of Precision?

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

Precision

What is the core intuition of Precision?

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

- Focus: Out of all predicted positives, how many were actually correct?

Precision

What is the core intuition of Precision?

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

- Focus: Out of all predicted positives, how many were actually correct?
- Critical when the cost of False Alarms (FP) is extremely high.

Precision

What is the core intuition of Precision?

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

- Focus: Out of all predicted positives, how many were actually correct?
- Critical when the cost of False Alarms (FP) is extremely high.
- Example case: Email spam filtering architectures.

Recall

What is the core intuition of Recall?

Recall

What is the core intuition of Recall?

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

Recall

What is the core intuition of Recall?

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

- Focus: Out of all actual positive items, how many did we successfully catch?

Recall

What is the core intuition of Recall?

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

- Focus: Out of all actual positive items, how many did we successfully catch?
- Critical when missing cases (FN) leads to catastrophic real-world results.

Recall

What is the core intuition of Recall?

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

- Focus: Out of all actual positive items, how many did we successfully catch?
- Critical when missing cases (FN) leads to catastrophic real-world results.
- Example case: Medical diagnostic screening algorithms (e.g., cancer detection).

Regression Metrics

- Evaluating classification algorithms involves comparing discrete variables (e.g., True vs. False).

Regression Metrics

- Evaluating classification algorithms involves comparing discrete variables (e.g., True vs. False).
- Regression requires quantifying a continuous numerical distance.

Regression Metrics

- Evaluating classification algorithms involves comparing discrete variables (e.g., True vs. False).
- Regression requires quantifying a continuous numerical distance.
- Metrics measure the size and distribution of prediction mistakes.

Summary of Regression Metrics

Metric	Mathematical Formula	Core Characteristic

Summary of Regression Metrics

Metric	Mathematical Formula	Core Characteristic
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	Treats all errors equally; robust to outliers

Summary of Regression Metrics

Metric	Mathematical Formula	Core Characteristic
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	Treats all errors equally; robust to outliers
MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Heavily penalizes large outlier errors

Summary of Regression Metrics

Metric	Mathematical Formula	Core Characteristic
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	Treats all errors equally; robust to outliers
MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Heavily penalizes large outlier errors
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	Returns error scale to original target units

Quiz

Question

Imagine an autonomous vehicle braking system. If it detects an obstacle, it slams on the brakes. Do we design the ML model powering this system to maximise *Precision* or *Recall*? Why?

Quiz

Question

Imagine an autonomous vehicle braking system. If it detects an obstacle, it slams on the brakes. Do we design the ML model powering this system to maximise *Precision* or *Recall*? Why?

Answer: *Recall*. A False Positive means the car brakes randomly for a shadow or a plastic bag (annoying but safe). A False Negative means the car fails to brake for an actual pedestrian (catastrophic). We must catch every single true positive obstacle.

Choosing Your Strategy

- Algorithms are divided based on foundational mathematical design philosophies.

Choosing Your Strategy

- Algorithms are divided based on foundational mathematical design philosophies.
- Strategy 1: Parametric Models.

Choosing Your Strategy

- Algorithms are divided based on foundational mathematical design philosophies.
- Strategy 1: Parametric Models.
- Strategy 2: Non-Parametric Models.

Parametric Strategy Profile

- Relies on rigid, clear predefined structural assumptions.

Parametric Strategy Profile

- Relies on rigid, clear predefined structural assumptions.
- Models possess a fixed, predetermined number of internal parameters.

Parametric Strategy Profile

- Relies on rigid, clear predefined structural assumptions.
- Models possess a fixed, predetermined number of internal parameters.
- Highly computationally efficient and requires fewer training entries.

Parametric Strategy Profile

- Relies on rigid, clear predefined structural assumptions.
- Models possess a fixed, predetermined number of internal parameters.
- Highly computationally efficient and requires fewer training entries.
- Examples: Linear Regression, Linear Discriminant Analysis (LDA).

Non-Parametric Strategy Profile

- Operates without rigid initial functional shape assumptions.

Non-Parametric Strategy Profile

- Operates without rigid initial functional shape assumptions.
- Parameter totals scale flexibly according incoming dataset volumes.

Non-Parametric Strategy Profile

- Operates without rigid initial functional shape assumptions.
- Parameter totals scale flexibly according incoming dataset volumes.
- Highly expressive but structurally data-hungry.

Non-Parametric Strategy Profile

- Operates without rigid initial functional shape assumptions.
- Parameter totals scale flexibly according incoming dataset volumes.
- Highly expressive but structurally data-hungry.
- Examples: k-Nearest Neighbors (k-NN), Some Decision Tree variations.

The Realities of Machine Learning Engineering

- Data quality and composition dictate ultimate algorithmic success.

The Realities of Machine Learning Engineering

- Data quality and composition dictate ultimate algorithmic success.
- **Golden rule:** Garbage In, Garbage Out.

The Realities of Machine Learning Engineering

- Data quality and composition dictate ultimate algorithmic success.
- **Golden rule:** Garbage In, Garbage Out.
- Nearly 80% of real-world project execution is spent purely on Preprocessing.

Parametric vs. Non-Parametric Models

Property	Parametric Models	Non-Parametric Models

Parametric vs. Non-Parametric Models

Property	**Parametric Models**	**Non-Parametric Models**
Assumptions	Strong functional form	No prior shape assumptions

Parametric vs. Non-Parametric Models

Property	**Parametric Models**	**Non-Parametric Models**
Assumptions	Strong functional form	No prior shape assumptions
Parameters	Fixed number	Flexible; scales with data size

Parametric vs. Non-Parametric Models

Property	**Parametric Models**	**Non-Parametric Models**
Assumptions	Strong functional form	No prior shape assumptions
Parameters	Fixed number	Flexible; scales with data size
Speed	Fast training and inference	Slower computation profiles

Parametric vs. Non-Parametric Models

Property	**Parametric Models**	**Non-Parametric Models**
Assumptions	Strong functional form	No prior shape assumptions
Parameters	Fixed number	Flexible; scales with data size
Speed	Fast training and inference	Slower computation profiles
Best Use Case	Smaller datasets	Complex, non-linear datasets

Course Summary

- Machine Learning maps structural inputs to target landscapes.

Course Summary

- Machine Learning maps structural inputs to target landscapes.
- Success balances understanding mathematical foundations and optimisation trade-offs.

Course Summary

- Machine Learning maps structural inputs to target landscapes.
- Success balances understanding mathematical foundations and optimisation trade-offs.
- Next: Linear Regression.

Questions & Discussion

Thank you for your time.