AI & Machine Learning Model Reference

A Guide for Model Selection and Cost-Effective AI Solutions

Abstract

This document provides a structured reference to Machine Learning, Deep Learning, and Natural Language Processing models, focusing on their core algorithms, high-level mathematical foundations, and typical business use cases. The objective is to support informed model selection by balancing accuracy, cost, and explainability.

Darshana Bordoloi

darshana.b@thecodeyogi.com

Contents

1. Introduction & Purpose....2
2. Learning Paradigms in AI....3
3. Supervised Learning Models – Machine Learning....5
4. Unsupervised Learning Models – Machine Learning....12
5. Semi-Supervised Learning Models....18
6. Deep Learning Models – Supervised & Unsupervised....20
7. Natural Language Processing (NLP) Models (Non-LLM)....25
8. Time Series Models....29
9. Model Selection Guide....32

1. Introduction & Purpose

This document serves as a technical reference for **Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP) models**, focusing on:

- Core algorithms.
- High-level mathematical foundations.
- Typical business and industry use cases.

The document is intended to support client discussions and solution design, helping identify which model is appropriate for a given problem and why, without relying on code-level explanations.

1.1 Objective

- Provide a structured overview of commonly used ML, NLP, and DL models.
- Explain algorithmic logic and mathematical intuition at a high level.
- Enable cost-effective and explainable model selection.
- Avoid unnecessary use of complex or expensive models.

1.2 Target Audience

- Clients and business stakeholders
- CTOs and engineering leaders
- AI and data science teams

1.3 Model Selection Considerations

Model choice directly impacts:

- Cost – infrastructure, inference, and maintenance.
- Accuracy – suitability of the model to the problem.
- Explainability – transparency required for business and regulatory needs.

Simpler ML and NLP models often provide sufficient accuracy with lower cost and higher interpretability.

1.4 Scope of This Document

This document primarily covers:

- Machine Learning models
- Deep Learning models
- Classical Natural Language Processing (NLP) models

The focus is on **algorithms, high-level mathematical foundations, and typical use cases**.
Generative AI and Large Language Models (LLMs) are referenced only at a high level, where relevant.

1.5 Usage Guidance

This document should be used as a reference during client discovery and solution design to:

- Map business problems to model categories.
- Explain algorithm and math intuition clearly.
- Justify model choices based on constraints and requirements.

2. Learning Paradigms in AI

Learning paradigms describe how models learn from data, independent of the specific algorithms used. Understanding these paradigms helps in selecting an appropriate modelling approach based on data availability, labelling effort, and business constraints.

2.1 Overview of Learning Paradigms

AI models are commonly categorized into the following learning paradigms:

- Supervised Learning – models learn from labelled data.
- Unsupervised Learning – models identify patterns in unlabelled data.
- Semi-Supervised Learning – models learn from a mix of labelled and unlabelled data.
- Reinforcement Learning – models learn through interaction with an environment using reward signals.

These paradigms apply across Machine Learning (ML), Deep Learning (DL), and NLP models.

Example:
A sales forecasting model trained on past sales data is supervised learning, whereas grouping customers by purchasing behaviour without predefined labels is unsupervised learning.

2.2 Supervised Learning

Supervised learning models are trained on datasets where each input is associated with a known output label. The objective is to learn a mapping function that generalizes well to unseen data.

Core characteristics

- Requires labelled training data.
- Supports regression and classification tasks.
- Model performance can be quantitatively evaluated.

Example:
Predicting whether a customer will churn using historical customer data where churn outcomes are already known.

Common business use cases

- Sales and demand forecasting
- Fraud and risk detection
- Customer churn prediction
- Text classification and sentiment analysis

2.3 Unsupervised Learning

Unsupervised learning models operate on unlabelled data and aim to discover hidden structures or patterns within the data.

Core characteristics

- No labelled outputs required.
- Focus on structure discovery rather than prediction.

- Often used for exploratory analysis.

Example:
Segmenting customers into distinct groups based on purchasing behaviour without predefined customer categories.

Common business use cases

- Customer and market segmentation.
- Behavioural pattern discovery.
- Anomaly and outlier detection.
- Data visualization and compression.

2.4 Semi-Supervised Learning

Semi-supervised learning combines a small labelled dataset with a larger unlabelled dataset to improve learning performance while reducing labelling cost.

Core characteristics

- Useful when labelling data is expensive or time-consuming.
- Builds on supervised learning principles.
- Improves generalization using unlabelled data.

Example:
Classifying large volumes of documents where only a small subset has been manually labelled.

Common business use cases

- Document classification at scale.
- Medical and scientific data analysis.
- Enterprise datasets with partial annotations.

2.5 Reinforcement Learning

Reinforcement learning trains models to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.

Core characteristics

- No labelled dataset
- Learning driven by reward maximization
- Suitable for sequential decision-making problems

Example:
An automated recommendation system that adjusts suggestions based on user interactions and feedback over time.

2.6 Mapping Learning Paradigms to Business Problems

Choosing the appropriate learning paradigm depends primarily on data characteristics and problem formulation, rather than algorithm sophistication.

Business Scenario	Suitable Paradigm	Example
Historical labelled data available	Supervised Learning	Sales forecasting
No labels, pattern discovery needed	Unsupervised Learning	Customer segmentation
Limited labels, large datasets	Semi-Supervised Learning	Document classification
Sequential decisions with feedback	Reinforcement Learning	Recommendation optimization

This paradigm-first view ensures practical, cost-effective, and scalable AI solutions.

3. Supervised Learning Models – Machine Learning

3.1 Regression Models

Regression models are supervised learning techniques used to predict **continuous numerical outcomes**. They are widely used in business contexts due to their **interpretability, low computational cost, and strong baseline performance**.

3.1.1 Linear Regression

Algorithm Overview

Linear Regression models the relationship between one or more input variables and a continuous target variable by fitting a linear equation. The model estimates coefficients that minimize the difference between predicted and actual values.
In simple terms, it answers:
How does a change in one or more input variables affect the output?
The model assumes:

- A linear relationship between inputs and output
- Additive effects of features
- Independent observations

Mathematical Foundations (High-Level)

- **Linear algebra**
  - Input features represented as vectors and matrices.
  - Model expressed as a weighted sum of features.
- **Loss function**
  - Mean Squared Error (MSE) measures prediction error.
- **Optimization**
  - Coefficients are estimated by minimizing the loss function.
  - Solved using closed-form solutions or gradient descent.
- **Statistical interpretation**
  - Coefficients represent the expected change in output for a unit change in input

Note: The mathematical formulation is simple and interpretable, making Linear Regression suitable for regulated and audit-sensitive environments.

Typical Business Use Cases

- Sales and revenue forecasting.
- Price and cost estimation.
- Demand prediction.

- Financial trend analysis.
- Baseline predictive models before complex approaches.

Example:
Predicting monthly sales based on advertising spend, seasonality indicators, and historical sales data.

When to Use / Not Use

Use when:

- Relationships are approximately linear.
- Interpretability is important.
- Dataset size is small to medium.

Avoid when:

- Strong non-linear relationships dominate.
- High multicollinearity exists without regularization.

3.1.2 Ridge, Lasso & ElasticNet Regression

These models extend Linear Regression by introducing **regularization**, which helps address overfitting and multicollinearity.

Regularization Concepts (Algorithm Overview)

Regularization adds a **penalty term** to the loss function to constrain model complexity.

- *Ridge Regression (L2 regularization)*
  Penalizes large coefficients, shrinking them toward zero.
- *Lasso Regression (L1 regularization)*
  Encourages sparsity by forcing some coefficients to become exactly zero.
- *ElasticNet Regression*
  Combines L1 and L2 penalties to balance feature selection and stability.

These models improve generalization when dealing with many correlated features.

Mathematical Foundations (High-Level)

- **Bias–Variance Trade-off**
  - Regularization increases bias slightly.
  - Significantly reduces variance.
  - Improves performance on unseen data.
- **Penalty terms**
  - L1 norm: sum of absolute coefficient values.
  - L2 norm: sum of squared coefficient values.
- **Optimization impact**
  - Prevents coefficients from becoming excessively large.
  - Improves numerical stability.

Key intuition:
Regularization trades a small amount of accuracy on training data for **better real-world performance**.

Typical Business Use Cases

- Forecasting with many correlated features.
- Financial modelling with noisy variables.

- Marketing mix modelling.
- Risk scoring models.
- High-dimensional business datasets.

Example:
Predicting customer lifetime value using dozens of correlated behavioural and demographic features.

Model Selection Guidance

- **Ridge:**
  Use when all features are relevant but correlated.
- **Lasso:**
  Use when feature selection is required.
- **ElasticNet:**
  Use when datasets are high-dimensional and features are correlated.

3.2 Classification Models

Classification models are supervised learning techniques used to predict **categorical outcomes**. They estimate the probability that an observation belongs to a particular class and are widely used in business for **risk assessment, decision automation, and customer analytics**.

3.2.1 Logistic Regression

Algorithm Overview

Logistic Regression is a **linear classification model** that estimates the probability of a binary outcome. Instead of predicting a continuous value, it models the likelihood that an observation belongs to a given class using a logistic (sigmoid) function.

The model answers:

What is the probability that this observation belongs to a specific class?

It assumes:

- A linear relationship between input features and the log-odds of the outcome.
- Independent observations.

Mathematical Foundations (High-Level)

- **Sigmoid (logistic) function**
  - Transforms linear combinations of features into probabilities between 0 and 1.
- **Probability modelling**
  - Uses log-odds (logit) to model class membership.
- **Loss function**
  - Optimized using log loss (cross-entropy).
- **Optimization**
  - Parameters estimated via gradient-based optimization.

Key intuition:
Logistic Regression learns a decision boundary that separates classes while providing **probabilistic outputs**, not just class labels.

Typical Business Use Cases

- Fraud detection (fraud vs non-fraud)

- Customer churn prediction
- Credit risk assessment
- Medical diagnosis classification
- Spam detection

Example:
Estimating the probability that a customer will churn based on usage patterns, service history, and demographics.

When to Use / Not Use

Use when:

- Binary classification is required
- Interpretability is important
- Probabilities are needed for decision thresholds

Avoid when:

- Strong non-linear decision boundaries exist
- Classes are highly complex and overlapping

3.2.2 k-Nearest Neighbors (k-NN)

Algorithm Overview

k-Nearest Neighbors is a non-parametric, instance-based classification algorithm. It assigns a class to a new data point based on the majority class among its *k* closest data points in the feature space.

The model answers:

Which class do the most similar past observations belong to?

Mathematical Foundations (High-Level)

- **Distance metrics**
  - Euclidean, Manhattan, or cosine distance
- **Vector space representation**
  - Observations represented as points in a multi-dimensional space
- **Decision rule**
  - Majority voting among nearest neighbors

Key intuition:
Similar observations tend to belong to the same class.

Typical Business Use Cases

- Recommendation systems
- Customer segmentation classification
- Pattern recognition in small datasets
- Similarity-based matching

Example:
Classifying a new customer's risk category based on similarity to existing customers.

When to Use / Not Use

Use when:

- Dataset size is small to medium.
- Feature scaling is well-controlled.
- Interpretability is less critical.

Avoid when:

- Dataset is very large.
- High dimensionality exists (curse of dimensionality).
- Real-time inference is required.

3.2.3 Naive Bayes

Algorithm Overview

Naive Bayes is a probabilistic classification algorithm based on Bayes' Theorem. It assumes that features are conditionally independent given the class label.

Despite its simplicity, it performs well in many high-dimensional problems, particularly in text classification.

Mathematical Foundations (High-Level)

- **Bayes' Theorem**
  - Combines prior probability with likelihood.
- **Conditional probability**
  - Assumes independence between features.
- **Probability estimation**
  - Class predicted by maximizing posterior probability.

Key intuition:
Even with a strong independence assumption, aggregated evidence from multiple features can lead to accurate classification.

Typical Business Use Cases

- Spam email classification
- Document and text classification
- Sentiment analysis (baseline models)
- Intent detection in NLP systems

Example:
Classifying customer support emails into categories such as billing, technical, or general inquiries.

When to Use / Not Use

Use when:

- Text or high-dimensional data is involved
- Fast training and inference are required
- Baseline or benchmark models are needed

Avoid when:

- Feature dependencies are strong and critical
- High precision on complex boundaries is required

3.3 Tree-Based Models

Tree-based models are supervised learning techniques that make decisions by recursively splitting data based on feature values. They are widely used in business applications due to their interpretability, ability to model non-linear relationships, and strong performance on tabular data.

3.3.1 Decision Trees

Algorithm Overview

A Decision Tree models decisions as a hierarchical structure of rules. At each node, the algorithm selects a feature and a split point that best separates the data according to a purity measure. The process continues recursively until a stopping criterion is met.

The model answers:

Which sequence of rules best separates the data into distinct outcome groups?

Decision Trees naturally resemble human decision-making, making them intuitive and explainable.

Mathematical Foundations (High-Level)

- **Entropy**
  - Measures the uncertainty or impurity in a dataset.
- **Gini Index**
  - Measures how often a randomly chosen element would be incorrectly classified.
- **Information Gain**
  - Quantifies the reduction in impurity after a split.
- **Recursive partitioning**
  - Data is split repeatedly to maximize class separation.

Key intuition:
Each split increases the “purity” of the resulting subsets.

Typical Business Use Cases

- Credit approval and risk assessment
- Customer segmentation and profiling
- Business rule automation
- Fraud detection (rule-based systems)

Example:
Approving or rejecting a loan based on income, credit score, and employment status using interpretable decision rules.

When to Use / Not Use

Use when:

- Interpretability is critical
- Business rules need to be explainable
- Non-linear relationships exist

Avoid when:

- Dataset is very large and noisy
- Overfitting cannot be controlled

3.3.2 Random Forest

Algorithm Overview

Random Forest is an ensemble learning method that builds multiple decision trees and aggregates their predictions. Each tree is trained on a random subset of data and features, and the final prediction is obtained by voting (classification) or averaging (regression).

The model answers:

What is the most reliable prediction when multiple diverse decision trees are combined?

Mathematical Foundations (High-Level)

- **Ensemble learning**
  - Combines multiple weak learners into a strong learner.
- **Bagging (Bootstrap Aggregation)**
  - Random sampling with replacement reduces variance.
- **Variance reduction**
  - Averaging predictions stabilizes model output.
- **Feature randomness**
  - Prevents dominance of strong predictors.

Key intuition:
Many diverse models together are more robust than a single model.

Typical Business Use Cases

- Fraud detection
- Insurance risk modelling
- Customer churn prediction
- Credit scoring
- Feature-rich tabular datasets

Example:
Predicting customer churn by combining behavioural, demographic, and transactional features.

When to Use / Not Use

Use when:

- High accuracy is required.
- Data has complex feature interactions.
- Overfitting is a concern.

Avoid when:

- Model transparency is mandatory.
- Inference speed must be extremely fast.

3.3.3 Gradient Boosting (XGBoost, LightGBM – Conceptual)

Algorithm Overview

Gradient Boosting builds models sequentially, where each new tree focuses on correcting the errors made by the previous ones. Instead of independent trees, the trees are additively combined to minimize a specified loss function.

The model answers:

How can each new model improve upon the mistakes of the previous ones?

XGBoost and LightGBM are optimized implementations designed for performance and scalability.

Mathematical Foundations (High-Level)

- **Boosting principle**
  - Models are trained sequentially.
- **Loss minimization**
  - Each new tree reduces the overall prediction error.
- **Gradient descent in function space**
  - Gradients guide how the next tree is constructed.
- **Regularization**
  - Controls model complexity and overfitting.

Key intuition:
Small, incremental improvements lead to highly accurate models.

Typical Business Use Cases

- High-accuracy classification problems
- Ranking and recommendation systems
- Fraud and anomaly detection
- Structured business datasets

Example:
Predicting loan default probability with high precision using historical financial data.

When to Use / Not Use

Use when:

- Predictive performance is critical.
- Dataset is structured and feature-rich.
- Computational resources are available.

Avoid when:

- Explainability is a strict requirement.
- Simpler models already meet business needs.

Why Tree-Based Models Are Widely Used?

- Handle non-linear relationships naturally.
- Work well with mixed data types.
- Require minimal feature preprocessing.
- Strong performance on real-world business data.

4. Unsupervised Learning Models – Machine Learning

Unsupervised learning models operate on **unlabelled data** and are primarily used to discover **patterns, structures, or groupings** within the data. These models are especially valuable in early-stage analysis, segmentation tasks, and scenarios where labelled data is unavailable or expensive to obtain.

4.1 Clustering Models

Clustering models group data points based on **similarity or proximity** in a feature space. Unlike classification, clustering does not rely on predefined labels and instead reveals **natural groupings** within the data.

4.1.1 K-Means Clustering

Algorithm Overview

K-Means is a centroid-based clustering algorithm that partitions data into *K* distinct clusters. Each data point is assigned to the cluster whose centroid is closest, and centroids are iteratively updated until convergence.

The model answers:

How can data points be grouped so that items within a group are more similar to each other than to those in other groups?

Mathematical Foundations (High-Level)

- **Vector space representation**
  - Data points represented as vectors in a multidimensional space.
- **Distance minimization**
  - Typically uses Euclidean distance.
  - Objective is to minimize within-cluster variance.
- **Iterative optimization**
  - Assignment step: assign points to nearest centroid.
  - Update step: recompute centroids.

Key intuition:
Clusters are formed by minimizing the distance between data points and their respective cluster centres.

Typical Business Use Cases

- Customer segmentation
- Market segmentation
- Product categorization
- Behavioural analysis

Example:
Grouping customers into segments based on purchasing frequency, spend, and engagement metrics.

When to Use / Not Use

Use when:

- Number of clusters is known or can be estimated.
- Data is numeric and scaled.
- Clusters are roughly spherical.

Avoid when:

- Clusters vary significantly in size or density.
- Data contains significant noise or outliers.

4.1.2 Hierarchical Clustering

Algorithm Overview

Hierarchical Clustering builds a hierarchy of clusters either by **progressively merging smaller clusters (agglomerative)** or **splitting larger clusters (divisive)**. The result is a tree-like structure called a **dendrogram.**

The model answers:

How are data points related at different levels of similarity?

Mathematical Foundations (High-Level)

- **Distance metrics**
  - Euclidean, Manhattan, or cosine distance.
- **Linkage criteria**
  - Single, complete, average, or Ward linkage.
- **Hierarchical structure**
  - Nested clusters represented as a dendrogram.

Key intuition:

Clusters can be analysed at multiple levels of granularity.

Typical Business Use Cases

- Customer and product segmentation
- Taxonomy creation
- Document clustering
- Exploratory data analysis

Example:

Building a hierarchy of products based on similarity to support category planning.

When to Use / Not Use

Use when:

- Cluster hierarchy is meaningful.
- Number of clusters is unknown.
- Interpretability is important.

Avoid when:

- Dataset is very large.
- Real-time clustering is required.

4.1.3 DBSCAN (Density-Based Spatial Clustering)

Algorithm Overview

DBSCAN groups data points based on **density**, identifying clusters as regions of high point concentration separated by regions of low density. Unlike K-Means, it does not require specifying the number of clusters in advance.

The model answers:

Which data points form dense regions, and which points are noise or outliers?

Mathematical Foundations (High-Level)

- **Density estimation**
  - Uses neighbourhood radius (ε) and minimum points.
- **Core, border, and noise points**
  - Core points form dense clusters.
  - Noise points are treated as outliers.
- **Distance-based neighbourhood definition**

Key intuition:

Clusters are dense areas separated by sparse regions.

Typical Business Use Cases

- Anomaly and outlier detection
- Fraud detection
- Geospatial data analysis
- Network traffic analysis

Example:
Detecting unusual transaction patterns that deviate from normal customer behaviour.

When to Use / Not Use

Use when:

- Clusters have arbitrary shapes.
- Noise detection is important.
- Number of clusters is unknown.

Avoid when:

- Data density varies significantly across clusters.
- High-dimensional data makes distance metrics unreliable.

Why Clustering Models Are Valuable in Business?

- Do not require labelled data.
- Reveal hidden structure in datasets.
- Enable segmentation and exploratory insights.
- Serve as a foundation for downstream supervised models.

4.2 Dimensionality Reduction

Dimensionality reduction techniques are used to **reduce the number of features** in a dataset while preserving as much meaningful information as possible. These methods are commonly applied to improve **model performance, interpretability, and computational efficiency**, especially when working with high-dimensional data.

4.2.1 Principal Component Analysis (PCA)

Algorithm Overview

Principal Component Analysis is a linear dimensionality reduction technique that transforms the original feature space into a new set of orthogonal components, called **principal components**. These components are ordered such that the first few retain most of the variability present in the data.

The model answers:

How can high-dimensional data be represented using fewer, more informative dimensions?

Mathematical Foundations (High-Level)

- **Covariance matrix**
  - Captures variance and correlation between features.
- **Eigenvalues and eigenvectors**
  - Eigenvectors define the direction of maximum variance.
  - Eigenvalues indicate the amount of variance captured.
- **Variance maximization**
  - Principal components are selected to maximize explained variance.
- **Orthogonal transformation**
  - Ensures components are uncorrelated.

Key intuition:
PCA finds new axes that best explain the variability in the data.

Typical Business Use Cases

- Feature reduction before modelling.
- Data visualization and exploratory analysis.
- Noise reduction.

- Improving model training speed and stability.

Example:
Reducing hundreds of customer behavioural features into a smaller set of principal components before training a predictive model.

When to Use / Not Use

Use when:

- Data has many correlated features.
- Model performance suffers due to high dimensionality.
- Interpretability of original features is less critical.

Avoid when:

- Feature interpretability is essential.
- Data relationships are strongly non-linear.

4.2.2 t-SNE (Conceptual)

Algorithm Overview

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique designed primarily for **visualizing high-dimensional data** in two or three dimensions. It preserves local structure, making similar data points appear close together.

The model answers:

How can complex, high-dimensional data be visualized to reveal clusters or patterns?

Mathematical Foundations (High-Level)

- **Probability-based similarity**
  - Converts distances into conditional probabilities.
- **Local neighbourhood preservation**
  - Emphasizes maintaining local relationships.
- **Non-linear mapping**
  - Uses gradient-based optimization to minimize divergence.

Key intuition:
t-SNE focuses on preserving local similarity rather than global structure.

Typical Business Use Cases

- Visualizing customer segments.
- Exploring document or text embeddings.
- Inspecting clustering quality.
- Model diagnostics and exploratory analysis.

Example:
Visualizing high-dimensional customer embeddings to identify natural clusters for segmentation.

When to Use / Not Use

Use when:

- Data exploration and visualization are the goal.
- Understanding local patterns is important.

Avoid when:

- Results need to be reproducible across runs.
- Dimensionality reduction is required for downstream modelling.

Why Dimensionality Reduction Matters?

- Reduces computational complexity.
- Improves model performance.
- Enables effective visualization.
- Helps manage high-dimensional datasets.

4.3 Anomaly Detection

Anomaly detection models are used to identify **rare, unusual, or abnormal observations** that deviate significantly from normal patterns in data. These techniques are particularly valuable in scenarios where anomalous events are **critical but infrequent**, and labelled anomaly data is limited or unavailable.

4.3.1 Isolation Forest

Algorithm Overview

Isolation Forest is an ensemble-based anomaly detection algorithm that identifies anomalies by **isolating observations through random partitioning**. Unlike distance- or density-based approaches, it explicitly models anomalies as data points that are **easier to separate** from the rest of the data.

The model answers:

Which data points are significantly different from the majority of observations?

Mathematical Foundations (High-Level)

- **Random partitioning**
  - Randomly selects features and split values.
- **Isolation principle**
  - Anomalies require fewer splits to be isolated.
- **Path length analysis**
  - Shorter average path lengths indicate anomalies.
- **Ensemble averaging**
  - Multiple random trees improve robustness.

Key intuition:

Anomalies are isolated faster than normal data points in randomly generated trees.

Typical Business Use Cases

- Fraud detection
- Network intrusion detection
- System and operational monitoring
- Financial transaction analysis

Example:

Detecting abnormal financial transactions that deviate from a customer's usual spending behaviour.

When to Use / Not Use

Use when:

- Dataset is large and high-dimensional
- Anomalies are rare
- Real-time or near real-time detection is required

Avoid when:

- Dataset is very small
- Clear distance-based patterns dominate

4.3.2 One-Class SVM

Algorithm Overview

One-Class Support Vector Machine is a boundary-based anomaly detection model that learns a decision boundary enclosing the majority of normal data points. Observations lying outside this boundary are classified as anomalies.

The model answers:

Which observations fall outside the region representing normal behaviour?

Mathematical Foundations (High-Level)

- **Kernel methods**
  - Enables modelling non-linear boundaries.
- **Margin maximization**
  - Separates normal data from the origin or boundary.
- **Hyperplane construction**
  - Defines a compact region of normality.

Key intuition:
Normal data occupies a well-defined region, and anomalies lie outside it.

Typical Business Use Cases

- Manufacturing defect detection
- Sensor data monitoring
- Network security
- Quality control systems

Example:
Identifying defective products based on sensor readings that fall outside normal operating ranges.

When to Use / Not Use

Use when:

- Normal behaviour is well-defined
- Dataset size is moderate
- Kernel-based modelling is beneficial

Avoid when:

- Dataset is very large
- Model interpretability is critical

Why Anomaly Detection Models Are Business-Critical?

- Enable early detection of rare but high-impact events.
- Reduce financial loss and operational risk.
- Often work without labelled anomaly data.
- Applicable across finance, operations, and security.

5. Semi-Supervised Learning Models

Semi-supervised learning combines **a small labelled dataset** with **a large unlabelled dataset** to train models more effectively than supervised learning alone. It is commonly used when labelling data is expensive, slow, or requires domain expertise.

5.1 Concept & Motivation

Algorithmic Perspective

Semi-supervised learning augments supervised models by **exploiting the structure of unlabeled data** during training. The learning process alternates between:

- Learning from labelled data
- Refining decision boundaries using unlabelled data

Mathematical Foundations (High-Level)

- **Cluster assumption**: data points form clusters, and points in the same cluster share labels
- **Manifold assumption**: data lies on a lower-dimensional manifold
- **Smoothness assumption**: similar inputs should produce similar outputs

Key intuition:

Unlabelled data constrains where decision boundaries should lie.

5.2 Self-Training Approaches

Algorithm Overview

1. Train a supervised model on labelled data.
2. Predict labels for unlabelled data.
3. Select high-confidence predictions.
4. Add them to the labelled dataset.
5. Retrain and repeat.

This creates a **bootstrapping loop**.

Mathematical Foundations (High-Level)

- **Probability thresholds** on model confidence.
- **Iterative risk minimization.**
- **Regularization** to limit error amplification.

Key intuition:

Confident predictions approximate true labels and can be reused for training.

Business Example

Automatically classifying large volumes of customer emails when only a few thousand are manually labelled.

5.3 Label Propagation

Algorithm Overview

Label propagation spreads labels from labelled data points to unlabelled ones based on similarity, typically modelled as a graph.

Mathematical Foundations (High-Level)

- **Graph representation**: nodes = data points, edges = similarity.
- **Weighted adjacency matrix.**
- **Iterative diffusion process** until convergence.

Key intuition:

Labels diffuse across dense regions of similar data.

Business Example

Categorizing product reviews by sentiment using similarity between text embeddings.

5.4 Graph-Based Learning

Algorithm Overview

Graph-based methods explicitly encode relationships between observations and optimize label assignments over the graph structure.

Mathematical Foundations (High-Level)

- **Graph Laplacian** to capture connectivity.
- **Energy minimization** enforcing smooth labels.
- **Regularization terms** penalizing label inconsistency.

Key intuition:

Connected data points should behave consistently.

Business Example

Detecting fraud rings in transaction networks where relational patterns matter more than individual records.

5.5 Typical Business Use Cases

- Large-scale document classification.
- Medical imaging with limited labelled data.
- Fraud detection with sparse confirmed cases.
- Enterprise data annotation pipelines.

6. Deep Learning Models – Supervised & Unsupervised

6.1 Fundamentals of Deep Learning

Deep Learning models are a class of machine learning models that learn **hierarchical representations** of data using multiple layers of neural networks. Unlike classical ML models, deep learning models can **automatically learn features from raw or minimally processed data**, making them suitable for complex and unstructured data.

6.1.1 Neural Networks Overview

Algorithm Overview

A neural network consists of interconnected layers of artificial neurons. Each neuron computes a weighted sum of its inputs, applies a non-linear transformation, and passes the result to the next layer.
A typical neural network includes:

- Input layer
- One or more hidden layers
- Output layer

The model answers:

How can complex patterns be learned through successive transformations of data?

Mathematical Foundations (High-Level)

- **Linear transformations**
  - Weighted sum of inputs using matrix multiplication.
- **Bias terms**
  - Shift activation functions to improve flexibility.

- **Function composition**
  - Each layer applies a transformation to the output of the previous layer.

Key intuition:
Neural networks approximate complex functions by stacking simple transformations.

Business Context

Neural networks are used when relationships between inputs and outputs are **highly non-linear** and difficult to capture using traditional ML models.

6.1.2 Activation Functions

Algorithm Overview

Activation functions introduce **non-linearity** into neural networks. Without activation functions, neural networks would behave like linear models regardless of depth.
Common activation functions include:

- ReLU
- Sigmoid
- Tanh

Mathematical Foundations (High-Level)

- **Non-linear mappings**
  - Enable networks to learn complex patterns.
- **Differentiability**
  - Required for gradient-based optimization.
- **Output constraints**
  - Some functions bound outputs (e.g., sigmoid outputs probabilities).

Key intuition:
Activation functions decide whether and how strongly a neuron should activate.

Business Context

Choosing appropriate activation functions impacts:

- Training stability
- Convergence speed
- Model performance

6.1.3 Loss Functions

Algorithm Overview

Loss functions quantify the difference between predicted outputs and actual targets. Training aims to **minimize this loss**.

Different tasks use different loss functions:

- Regression → Mean Squared Error
- Classification → Cross-Entropy Loss

Mathematical Foundations (High-Level)

- **Error measurement**
  - Scalar value representing prediction quality.
- **Optimization objective**
  - Loss serves as the function to be minimized.

- **Differentiability**
    - Enables gradient computation.

Key intuition:
The loss function tells the model how wrong its predictions are.

Business Context

Loss functions directly influence:
- Model accuracy
- Training behaviour
- Sensitivity to outliers

6.1.4 Backpropagation

Algorithm Overview

Backpropagation is the algorithm used to compute gradients of the loss function with respect to model parameters. It propagates error information **backwards through the network**, enabling weight updates.

The algorithm answers:

How should each parameter change to reduce prediction error?

Mathematical Foundations (High-Level)

- **Chain rule (calculus)**
    - Gradients computed layer by layer.
- **Partial derivatives**
    - Measure sensitivity of loss to parameters.
- **Gradient flow**
    - Error signals move from output to input layers.

Key intuition:
Errors are distributed backward so each parameter learns its contribution to the mistake.

Business Context

Backpropagation enables deep networks to:
- Learn from large datasets
- Continuously improve performance
- Adapt to complex patterns

6.1.5 Gradient Descent

Algorithm Overview

Gradient Descent is an optimization method that updates model parameters in the direction that reduces the loss function. Variants differ in how much data is used per update.
Common variants include:
- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent

Mathematical Foundations (High-Level)
- **Gradient vectors**
    - Indicate direction of steepest ascent.
- **Learning rate**
    - Controls step size during updates.
- **Iterative optimization**

- Repeated updates converge to a minimum.

Key intuition:
Gradient descent gradually moves the model toward better solutions.

Business Context

Optimization strategy impacts:
- Training time
- Convergence stability
- Computational cost

6.2 Feedforward Neural Networks (ANN)

6.2.1 Algorithm Overview

Feedforward Neural Networks **(also called Multilayer Perceptrons)** consist of stacked layers of neurons where information flows **only in one direction:** input → hidden layers → output.

Each layer applies a linear transformation followed by a non-linear activation. Learning happens by adjusting weights to minimize prediction error.

The model answers:

How can complex non-linear relationships be learned from structured or feature-engineered data?

6.2.2 Mathematical Foundations (High-Level)

- **Matrix multiplication** for weighted sums.
- **Activation functions** to introduce non-linearity.
- **Loss minimization** (e.g., MSE, cross-entropy).
- **Backpropagation** using the chain rule.
- **Gradient descent** for parameter updates.

Key intuition:
ANNs approximate complex functions by composing many simple transformations.

6.2.3 Typical Business Use Cases

- Customer churn prediction
- Credit risk modelling
- Recommendation systems (tabular features)
- Pricing and demand forecasting

Example:
Predicting customer churn using usage metrics, demographics, and transaction history when relationships are highly non-linear.

6.3 Convolutional Neural Networks (CNN)

6.3.1 Convolution & Pooling

Algorithm Overview

CNNs are designed for **grid-like data** such as images and scanned documents. They use convolutional filters to detect local patterns (edges, shapes) and pooling layers to reduce spatial dimensions.

6.3.2 Mathematical Foundations (High-Level)

- **Convolution operation** (sliding filters).

- **Shared weights** reduce parameter count.
- **Pooling** for spatial invariance.
- **Hierarchical feature extraction.**

Key intuition:
CNNs learn local patterns first, then combine them into higher-level concepts.

6.3.3 Typical Business Use Cases

- Image classification
- Object detection
- Optical Character Recognition (OCR)
- Document processing and layout analysis

Example:
Extracting text and structure from scanned invoices and documents.

6.4 Recurrent Neural Networks (RNN)

6.4.1 Sequential Modelling

Algorithm Overview

RNNs process data **sequentially,** maintaining a hidden state that captures information from previous time steps. This makes them suitable for ordered data.

6.4.2 Mathematical Foundations (High-Level)

- **Recurrent connections** reuse weights over time.
- **Hidden state updates** via matrix operations.
- **Backpropagation Through Time (BPTT).**

Vanishing Gradient Problem:
Gradients shrink as they propagate backward through many time steps, making long-term learning difficult.

6.4.3 Typical Business Use Cases

- Time-series forecasting
- Sequential event prediction
- Basic language modelling

Example:
Predicting next-day demand using recent historical demand values.

6.5 Long Short-Term Memory (LSTM) & GRU

6.5.1 Gating Mechanisms

Algorithm Overview

LSTM and GRU are specialized RNNs designed to overcome the vanishing gradient problem. They use **gates** to control what information is remembered or forgotten.

- **LSTM:** input, forget, output gates.
- **GRU:** update and reset gates.

6.5.2 Mathematical Foundations (High-Level)

- **Sigmoid gates** control information flow.
- **Cell state** preserves long-term memory.
- **Element-wise operations** regulate gradients.

Key intuition:
Gates act like valves that protect important information over long sequences.

6.5.3 Typical Business Use Cases

- Sales and demand forecasting
- Financial time-series analysis
- User behaviour sequence modelling

Example:
Forecasting monthly sales by capturing long-term seasonality and trends.

6.6 Autoencoders

6.6.1 Representation Learning

Algorithm Overview

Autoencoders are neural networks trained to **reconstruct their input.** They consist of:

- Encoder → compresses data
- Decoder → reconstructs data

The bottleneck layer learns a compact representation.

6.6.2 Mathematical Foundations (High-Level)

- **Dimensionality reduction** via encoding.
- **Reconstruction loss** (e.g., MSE).
- **Latent space learning.**

Key intuition:
If a model cannot reconstruct a data point well, it may be anomalous.

6.6.3 Typical Business Use Cases

- Anomaly detection
- Data compression
- Feature extraction

Example:
Detecting unusual system behaviour when reconstruction error exceeds normal thresholds.

6.7 Business Relevance and Model Selection Considerations

- Deep learning models are most effective when data is large-scale, complex, and unstructured, and when classical ML models fail to capture underlying patterns.
- Increased model complexity introduces higher training cost, infrastructure requirements, and maintenance overhead.
- Deep learning should be adopted selectively, with clear justification based on accuracy gains and business impact.
- In regulated or audit-sensitive environments, explainability requirements may favor simpler ML models over deep learning.

7. Natural Language Processing (NLP) Models (Non-LLM)

Classical NLP models focus on **representing text numerically** and applying statistical or machine learning techniques to extract meaning. These approaches are often **far more cost-effective, predictable, and easier to deploy** than Large Language Models, while still delivering strong business value.

7.1 Text Representation Techniques

Text representation techniques convert unstructured text into **numerical feature vectors** that can be used by ML models.

7.1.1 Bag of Words (BoW)

Algorithm Overview

Bag of Words represents a **document as a vector of word counts,** ignoring grammar and word order. Each unique word corresponds to a feature, and the value represents its frequency in the document.

The model answers:

Which words appear in a document, and how often?

Mathematical Foundations (High-Level)

- **Vocabulary construction**
- **Frequency vectors**
- **Sparse matrix representation**

Key intuition:

Meaning is approximated by word presence and frequency.

Typical Business Use Cases

- Basic text classification
- Spam detection (baseline)
- Keyword analysis

Example:

Classifying emails as spam or non-spam based on word occurrence.

7.1.2 TF-IDF (Term Frequency – Inverse Document Frequency)

Algorithm Overview

TF-IDF improves upon BoW by weighting words based on their importance. Words common in a document but rare across the corpus receive higher weights.

The model answers:

Which words best distinguish this document from others?

Mathematical Foundations (High-Level)

- **Term Frequency (TF):** word frequency in a document
- **Inverse Document Frequency (IDF):** rarity across documents
- **Logarithmic scaling**
- **Vector space representation**
- **Cosine similarity for comparison**

Key intuition:

Important words are frequent in a document but rare globally.

Typical Business Use Cases

- Document classification
- Resume screening
- Search relevance
- Document similarity

Example:
Matching resumes to job descriptions based on keyword importance.

7.2 Word Embeddings

Word embeddings **represent words as dense vectors** that capture semantic meaning and relationships.

7.2.1 Word2Vec

Algorithm Overview

Word2Vec learns word embeddings using shallow neural networks based on word co-occurrence.

Two architectures:

- **CBOW:** predicts a word from its context.
- **Skip-gram:** predicts context words from a target word.

Mathematical Foundations (High-Level)

- **Shallow neural networks**
- **Vector optimization**
- **Cosine similarity**
- **Semantic distance**

Key intuition:
Words appearing in similar contexts have similar meanings.

Typical Business Use Cases

- Semantic search
- Recommendation systems
- Document similarity

Example:
Finding similar products based on textual descriptions.

7.2.2 GloVe (Global Vectors)

Algorithm Overview

GloVe learns word embeddings using **global word co-occurrence statistics** rather than local context windows.

Mathematical Foundations (High-Level)

- **Co-occurrence matrix**
- **Matrix factorization**
- **Weighted least squares optimization**

Key intuition:
Global word statistics capture meaningful semantic relationships.

Typical Business Use Cases

- Topic clustering
- Text similarity
- NLP feature engineering

7.3 Topic Modelling

7.3.1 Latent Dirichlet Allocation (LDA)

Algorithm Overview

LDA is a probabilistic model that **represents documents as mixtures of topics**, where **each topic is a distribution over words.**

The model answers:

What topics exist in this collection of documents?

Mathematical Foundations (High-Level)

- **Probabilistic generative modelling**
- **Dirichlet distributions**
- **Bayesian inference**

Key intuition:
Documents are composed of multiple hidden topics.

Typical Business Use Cases

- Topic discovery in large document collections
- Customer feedback analysis
- Content categorization

Example:
Identifying common complaint themes from thousands of customer reviews.

7.4 NLP + ML Pipelines

Classical NLP pipelines combine **text representation techniques with machine learning models.**

7.4.1 Sentiment Analysis

- TF-IDF / embeddings + Logistic Regression / Naive Bayes.
- Used for brand monitoring and customer feedback analysis.

Example:
Analyzing customer reviews as positive, neutral, or negative.

7.4.2 Text Classification

- Emails, tickets, documents categorized automatically.

Example:
Routing support tickets to the correct department.

7.4.3 Document Similarity

- Cosine similarity on TF-IDF or embeddings.

Example:
Detecting duplicate or near-duplicate documents.

7.4.4 Intent Detection

- Classifying user intent from short text.

Example:
Understanding user intent in chatbots without using LLMs.

Why Classical NLP Models Are Critical for Businesses?

- Low inference cost.
- Predictable behaviour.
- Easy to explain and debug.
- Suitable for production at scale.

7.5 Business Relevance and Model Selection Considerations

- Many NLP use cases can be effectively solved using **classical text representation techniques** (TF-IDF, embeddings) combined with ML models.
- Large Language Models are often unnecessary for tasks such as classification, similarity, routing, and keyword extraction.
- Classical NLP approaches offer **predictable behaviour, lower latency, and easier production deployment**.
- Model choice should prioritize **cost efficiency, interpretability, and scalability**, especially in high-volume enterprise systems.

8. Time Series Models

Time series models are used to analyze and forecast data points indexed over time. These models explicitly account for **temporal dependency**, making them essential for forecasting, planning, and monitoring tasks across business domains.

8.1 Classical Time Series Models

Classical time series models rely on **statistical assumptions** and are well-suited for problems with limited data, strong interpretability requirements, and stable temporal patterns.

8.1.1 ARIMA (Autoregressive Integrated Moving Average)

Algorithm Overview

ARIMA models a time series using:

- **Autoregression (AR):** dependence on past values.
- **Integration (I):** differencing to remove trends.
- **Moving Average (MA):** dependence on past forecast errors.

The model answers:

How can future values be predicted from past observations and past errors?

Mathematical Foundations (High-Level)

- **Autoregression:** linear regression on lagged values.
- **Differencing:** transforms non-stationary series into stationary ones.
- **Moving average:** models residual error structure.
- **Stationarity:** constant mean and variance over time.

Key intuition:
Future values are a weighted combination of past values and past errors, assuming stable statistical properties.

Typical Business Use Cases

- Sales and demand forecasting.
- Inventory planning.
- Capacity forecasting.
- Financial trend analysis.

Example:
Forecasting weekly product demand using historical sales data.

When to Use / Not Use

Use when:

- Dataset is small to medium.
- Patterns are linear and stable.
- Explainability is important.

Avoid when:

- Strong non-linear patterns exist.
- Multiple external variables dominate behaviour.

8.1.2 SARIMA (Seasonal ARIMA)

Algorithm Overview

SARIMA extends ARIMA by explicitly modelling **seasonal patterns** through additional seasonal autoregressive and moving average terms.

The model answers:

How can repeating seasonal patterns be incorporated into forecasting?

Mathematical Foundations (High-Level)

- **Seasonal differencing**
- **Seasonal autoregressive terms**
- **Seasonal moving average terms**
- **Periodic stationarity**

Key intuition:
Seasonal effects are predictable and can be modelled separately from overall trends.

Typical Business Use Cases

- Monthly or quarterly sales forecasting
- Seasonal demand planning
- Energy consumption forecasting

Example:
Forecasting retail sales with strong yearly seasonality.

8.2 Deep Learning for Time Series

Deep learning models are used for time series when patterns are **non-linear**, influenced by **long-term dependencies**, or when multiple interacting signals exist.

8.2.1 LSTM for Forecasting

Algorithm Overview

LSTM networks are designed to **model sequential data while retaining information over long time horizons.** They use gated mechanisms to decide what information to keep or discard at each time step.

The model answers:

How can long-term temporal dependencies be captured in complex sequences?

Mathematical Foundations (High-Level)

- **Gating mechanisms:** control information flow
- **Cell state:** preserves long-term memory
- **Non-linear transformations**
- **Backpropagation Through Time (BPTT)**

Key intuition:

LSTM selectively remembers important past information while ignoring noise.

Typical Business Use Cases

- Long-term sales forecasting.
- Financial market modelling.
- Sensor and IoT data forecasting.
- User behaviour sequence prediction.

Example:

Forecasting monthly revenue by learning long-term trends and seasonality from historical data.

8.2.2 Comparison: ARIMA vs LSTM

Aspect	ARIMA / SARIMA	LSTM
Model type	Statistical	Deep learning
Data requirement	Low–Medium	High
Pattern handling	Linear	Non-linear
Seasonality	Explicit	Learned
Interpretability	High	Low
Computational cost	Low	High
Production complexity	Low	Medium–High

Model Selection Guidance

- Use **ARIMA / SARIMA** when:
  - Data is limited.
  - Interpretability is required.
  - Patterns are stable.
- Use **LSTM** when:
  - Data volume is large.
  - Long-term dependencies exist.
  - Accuracy is prioritized over explainability.

8.3 Business Relevance and Model Selection Considerations

- Classical models such as **ARIMA and SARIMA** are often sufficient when time series data is limited, patterns are stable, and explainability is required.
- Deep learning models such as **LSTM** should be considered only when long-term dependencies, non-linear patterns, or large volumes of historical data are present.
- Model selection should align with **data maturity**, forecast horizon, and operational constraints.
- For many business forecasting problems, simpler statistical models deliver **comparable business value at significantly lower cost**.

9. Model Selection Guide

This section provides a **practical decision framework** for selecting appropriate AI models based on business requirements, data characteristics, and operational constraints. The goal is to **maximize business value while minimizing unnecessary complexity and cost**.

9.1 Use Case → Model Mapping Matrix

The table below maps common business use cases to recommended model categories, highlighting when simpler models are sufficient.

Business Use Case	Recommended Model Type	Notes
Sales forecasting	ARIMA / SARIMA	Prefer when data is limited and patterns are stable
Demand forecasting (complex)	LSTM	Use only with sufficient historical data
Customer churn prediction	Logistic Regression / Random Forest	Interpretable and cost-effective
Fraud detection	Random Forest / Gradient Boosting	Handles non-linear patterns well
Anomaly detection	Isolation Forest / Autoencoders	Choose based on data scale
Text classification	TF-IDF + ML	Low cost, fast, scalable
Sentiment analysis	NLP + ML	LLMs usually unnecessary
Document similarity	TF-IDF / Embeddings	Deterministic and efficient
Image classification	CNN	Use when image data is primary
Recommendation systems	ML / DL (context-dependent)	Complexity depends on scale

Guiding principle:

Start simple and increase complexity only when required by data and performance needs.

9.2 When NOT to Use Deep Learning

Deep learning models should **not** be the default choice.
Avoid deep learning when:

- Dataset size is small or medium.
- Relationships are largely linear or rule-based.
- Interpretability and auditability are required.
- Latency and infrastructure cost are critical constraints.
- Classical ML models already meet accuracy requirements.

In many enterprise scenarios, tree-based or linear models provide sufficient performance with far lower operational overhead.

9.3 When LLMs Are Overkill

Large Language Models are powerful but expensive and difficult to control. They are often unnecessary for:

- Text classification
- Sentiment analysis
- Keyword extraction
- Document similarity
- Intent detection
- Support ticket routing

In these cases, classical NLP pipelines offer:

- Predictable behaviour
- Lower latency

- Lower cost
- Easier production deployment

LLMs should be reserved for tasks that require:

- Language generation
- Multi-step reasoning
- Conversational intelligence
- Open-ended responses

9.4 Cost vs Accuracy vs Explainability

Model selection involves balancing three competing factors:

Factor	Simple Models (ML / NLP)	Complex Models (DL / LLM)
Cost	Low	High
Accuracy	Moderate–High	High
Explainability	High	Low
Deployment complexity	Low	High
Maintenance	Low	High

Key takeaway:

The most accurate model is not always the most valuable model.
For many business problems, **explainability, cost control, and operational simplicity** outweigh marginal accuracy gains.