# Introduction into machine learning

* **Definition (Tom Mitchell)**: A system learns if its performance on a task improves with experience.
* Why ML?
  + Automates tasks that are too complex for explicit programming.
  + Adapts as the new data arrives
  + Handles noisy, high-dimensional data and generalizes to unseen examples.
  + Handles high-dimensional data: The model can work with data that has many features (columns), like a dataset with hundreds of input values per example.
  + Generalizes to unseen examples: After training, the model can make accurate predictions on new data it hasn’t seen before — it doesn’t just memorize the training examples.

**Supervised vs Unsupervised Learning**

* **Supervised Learning**
  + Uses **labeled** data (input + known output)
  + Classification: output is a **discrete** label (category) (spam or not spam, image recognition)
  + Regression: output is a **continuous** number (value) (house price)
* **Unsupervised Learning**
  + Uses **unlabeled** data(only inputs)
  + **Clustering:** group similar data points (e.g., customer segments).
  + **Dimensionality Reduction:** find lower-dimensional structure (e.g., PCA).

# Challenges in ML

**Data Quality Issues**

* *Some rows have missing test scores.*
* *Some names are in all caps, some are lowercase.*
* *Some ages are mistakenly recorded as 200.*
* Clean your data: remove or fix missing/incorrect values.
* Normalize and standardize features when needed.
* Use techniques like imputation or outlier detection.

**Data Representativeness Issues**

* *Training images are mostly from light-skinned patients.*
* *The model might perform poorly on dark-skinned patients.*
* Collect diverse and balanced datasets.
* Be aware of sampling bias or historical bias in data.
* Validate your model on different subgroups.

**Overfitting / Underfitting**

*A decision tree learns every detail of the training data:*

* *It even memorizes noise or rare outliers.*
* *But when shown new data, it fails to generalize.*

**Symptoms**

* High accuracy on training data
* Low accuracy on test data

**Solutions**

* Use simpler models
* Apply regularization (penalties)
* Use more data
* Try cross-validation

# ML Workflow and Methodologies

**ML Workflow Standard Steps:**

1. Data collection
   1. Gather input features and output labels
   2. Data must be representative – it should reflect the real-world situation where the model will be used
2. Model Selection
   1. Choose algorithm
   2. This depends on the problem (classification / regression)
3. Training
   1. Fit The Model to the training data
   2. Adjust weights and parameters to reduce error on training data
4. Evaluation
   1. Measure how well the model performs on unseen data
   2. Use metrics like accuracy, precision, recall, F1, MSE
5. Iteration
   1. Checks how well model generalizes to the new data
   2. Helps to catch overfitting and underfitting

**Why a Representative Training Set Matters**

* If the training data doesn’t reflect real-world cases, the model will perform poorly on new data (bad generalization)
* Example: Training a spam filter only on work emails won't generalize to personal emails.

**Purpose of the Evaluation Step**

* Checks how well the model generalizes to new data.
* Helps catch overfitting (model too tuned to training) or underfitting (model too simple)

**Test Set & Generalization Error**

* Test set = held-out data used once at the end to estimate real-world performance.
* Helps us measure the true generalization error: how well the model works on future data

**Hyperparameter Tuning**

* Hyperparameters = settings chosen before training (e.g., learning rate, tree depth, C in SVM).
* They control model complexity and behavior.
* We tune them to find the best performing model—not too simple, not too complex

**Validation Set During Tuning**

* A validation set is used to test different hyperparameter settings.
* Model is trained on the training set, evaluated on validation set.
* Best settings are chosen based on validation accuracy

**ML Methodologies Overview**

Train-Test Methodology

Split data into:

* Train set: fit the model.
* Test set: evaluate final model once.

Simple but unstable if dataset is small

**Validation Methodology**

Split into 3 parts: Train, Validation, Test

* Train model on training set.
* Tune hyperparameters on validation set.
* Evaluate final model on test set.

Validation accuracy helps you choose the best configuration

**Cross-Validation**

Repeatedly split data into train/validation k times (k-fold).

Every data point gets to be validation once.

Average results to choose best hyperparameters.

More reliable than a single split

**Golden Rule: Test Set Must Be Isolated**

Never use test data to make decisions (no peeking!).

Keeps the evaluation honest and unbiased

# Data Preparation

**Stages of Data Preparation**

1. Data Exploration
2. Data Cleaning
3. Feature Selection
4. Feature Engineering
5. Data Transformation

**Data Exploration**

Goal: Understand the data before modifying it.

Activities**:**

* **Identify feature types**: Categorical vs. Continuous
* **Check for missing values and outliers**
* **Identify the target** variable (for supervised learning)
* **Visualize data**: Distributions, histograms, boxplots
* **Study correlations** between features and target
* **Identify transformations** to normalize or encode features

**Data Cleaning**

Outlier Handling:

* Use boxplots or IQR to find them.
* Actions:
* Keep (if valid)
* Remove (if error)
* Transform (e.g., log, sqrt to reduce skew)

Missing Values:

Types:

* MCAR: Missing completely at random
* MAR: Missing at random (depends on other variables)
* MNAR: Missing not at random (depends on its own value)

Categorical Features:

* Drop feature (if mostly missing)
* Fill with **mode**
* Create "Missing" category
* Random sampling from known values

**Feature Selection**

Goal: Remove irrelevant or redundant features.

Methods:

* Correlation Analysis: Drop one of two highly correlated features.
* Model-based: Use feature importance (e.g., from Decision Trees, Lasso).
* Domain knowledge
* Data quality: Drop features with many missing values or low variance.

**Feature Engineering & Data Transformation**

Engineering Techniques:

* Discretize continuous features (e.g., income bins)
* Interaction terms: Add ratios or combinations of features
* Date decomposition: Split into year, month, weekday
* Cyclic encoding for time: sine/cosine transforms

Transformations:

* log(x), sqrt(x), x² to reduce skew
* One-hot encoding for categorical data
* Scaling:
  + Min-Max scaling: [0, 1]
  + Standard scaling: Mean 0, SD 1
  + Normalize: Unit vector length

Dimensionality Reduction:

* PCA: Reduce feature count while keeping most variance

**Timing of Preparation Steps**

Correct approach:

* Apply data cleaning, feature engineering, scaling only on training + validation data.
* Keep test data untouched until the very end (to avoid leakage).

Why?

* Prevent test data from influencing model decisions.

Sometimes:

* Practitioners do cleaning on the full dataset before splitting due to convenience—but this may introduce bias.

# Classification Overview

**What is Classification?**

Classification is a supervised learning task where the goal is to predict discrete class labels (like "spam" vs. "not spam", or digit 5 vs. not 5) based on input features.

* Input: Features (e.g. pixel values, age, income)
* Output: A category (e.g. class 0, 1, 2...)
* Example: Email → Predict "spam" or "not spam"

**Classification Models**

k-NN (k-Nearest Neighbors)

* k-NN classifies a new data point by looking at the 'k' closest examples from the training set and using majority vote to pick the class.
* **Instance based** Because k-NN doesn't actually learn a model. It just memorizes the data and uses it at prediction time.
* How it works:
  + You have a dataset with labeled points (like “spam” or “not spam”).
  + For a new point, compute the distance to every training point.
  + Pick the k closest points (the "neighbors").
  + Count how many are from each class.
  + Assign the most frequent class.
* Distance Metric
* This is how we measure closeness:
  + Euclidean distance (straight line between points)
  + Other examples: Manhattan distance, cosine similarity
* Scaling Matters!
  + If one feature (e.g., income) ranges from 0–100,000 and another (e.g., age) from 0–100, the income will dominate the distance. To fix this:
    - Use standard scaling (mean 0, std 1) or
    - min-max scaling (0 to 1)

Naive Bayes

A black text on a white background

AI-generated content may be incorrect.

* What’s “Naive”?
  + It **assumes all features are independent** given the class label.
  + Example: If you're classifying emails as spam or not, it assumes that words like “free”, “win”, and “money” appear independently.
* How it works:
  + Calculates how likely the features are for each class.
  + Multiplies the probabilities.
  + Picks the class with the highest final probability.
* Great For:
  + Text data (spam filtering, sentiment analysis)
  + Fast, works well with lots of features

Logistic Regression

* Despite the name, it is a classification algorithm.
* A black text on a white background

  AI-generated content may be incorrect.
* z is a linear combination of features: just weights times features plus bias
* σ(z) is the sigmoid function, maps any number to a value between 0 and 1 → interpreted as probability
* Decision Rule:
  + If output > 0.5 → class 1
  + Else → class 0
* Why sigmoid?
  + It squashes any number into the range (0, 1), making it perfect for representing probability of being in class 1.
* Uses
  + Simple and interpretable
  + Gives probability estimates
  + Linear decision boundary

Support Vector Machines (SVM)

* Core idea:
  + Find the optimal boundary (hyperplane) that separates classes with the maximum margin
* Terms
  + Hyperplane: A line (2D), plane (3D), or higher-dimensional equivalent.
  + Margin: Distance between the boundary and the closest point from any class.
  + Support Vectors: Points that lie on the edge of the margin – they define the boundary.
* Why use it?
  + Good for high-dimensional data
  + Focuses only on critical points (support vectors)
  + Can be made nonlinear using kernels
* Kernels (Shortcut for feature transformation):
  + Polynomial kernel
  + RBF (Gaussian) kernel

These allow SVMs to find nonlinear boundaries by mapping inputs to a higher-dimensional space without actually computing all the features (this is called the kernel trick).

Tree-Based Models

A. Decision Trees

* Splits the data based on questions like “Is age > 30?”
* Builds a tree where each path leads to a class
* Leaf = final decision (class)
* Terms:
  + Gini impurity: Measures how mixed the classes are in a node. Lower is better.
  + Pre-pruning: Stop growing the tree early to avoid overfitting (e.g., max depth = 5)

B. Random Forest

* Ensemble of many decision trees
* Each tree is trained on a random sample (bootstrapping)
* At each split, only a random subset of features is considered
* Randomness:
  + Different data for each tree
  + Different features for each split
* Result: More diversity, less overfitting
* Prediction:
  + For classification: vote among all trees
  + Better accuracy, more stable than a single tree

C. Gradient Boosted Trees

* Builds trees sequentially, each trying to fix errors of the previous
* More sensitive and accurate than Random Forest, but slower
* Boosting is like learning from mistakes — each tree looks at what the last tree got wrong.

Neural Networks

* Structure:
  + Layers of artificial neurons:
    - Input → Hidden → Output
* Neuron:
* A black text on a white background

  AI-generated content may be incorrect.
* Common types:
  + Feedforward NN: basic architecture
  + CNN: images
  + RNN: sequences (text, time series)
* Why good for classification?
  + Can learn complex patterns
  + Can output probabilities (softmax for multiclass)

**Evaluation Metrics**

* If 95% of your data is class 0, and you always predict class 0:
  + Accuracy = 95% (sounds good)
  + But you never detect class 1, so it's useless.
* This happens in imbalanced datasets.

Confusion Matrix

|  | **Predicted Positive** | **Predicted Negative** |
| --- | --- | --- |
| **Actual Positive** | True Positive (TP) | False Negative (FN) |
| **Actual Negative** | False Positive (FP) | True Negative (TN) |
| Each term:   * TP: predicted positive, actually positive * FP: predicted positive, actually negative (false alarm) * FN: predicted negative, actually positive (missed detection) * TN: predicted negative, actually negative   Key Metrics     * "Of all positive predictions, how many were correct?" * High precision = few false positives * Example: Used in spam filters — we don’t want to mark legit emails as spam.   Recall (Sensitivity)     * "Of all actual positives, how many did we catch?" * High recall = few false negatives * Example: In medical tests, missing a disease (false negative) can be dangerous.   F1 Score     * Harmonic mean of precision and recall. * High only when both are high. * Best when you need a balance.   **Precision-Recall Trade-Off**   * Most models output scores/probabilities, not hard classes. * You must pick a threshold to convert scores into 0/1.   + Raise the threshold → higher precision, lower recall   + Lower the threshold → higher recall, lower precision * Plotting Precision vs. Recall at different thresholds = Precision-Recall Curve |  |  |

# Regression Details

**Regression** is a supervised learning technique where we try **to predict continuous (any real number) numbers** (not categories).

Think of it like writing a program that guesses a house price based on features like size, location, and age.

The **goal** is to find a function that maps inputs (called features) to outputs (the target), minimizing the prediction error.

**Standard Linear Regression (Core Idea)**

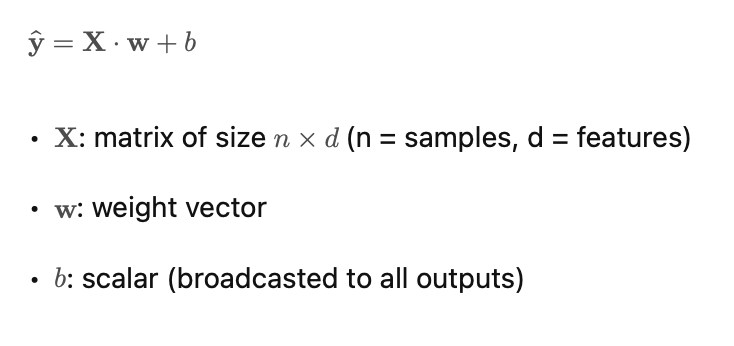
In Linear Regression, we assume the relationship between input features and the output is linear (a straight line in 2D, a hyperplane in higher dimensions).

A white background with black text

AI-generated content may be incorrect.

**Vectorized Prediction**

For many samples and features, we use matrix notation:



**OLS (Ordinary Least Squares)**

Find weights w that minimize the total squared error between actual and predicted values.

A white background with black text

AI-generated content may be incorrect.

This means:

• Small errors → good prediction.

• Big errors → penalized heavily (because of squaring).

OLS gives an exact solution using linear algebra:

A black text on a white background

AI-generated content may be incorrect.

This finds the best possible weights that minimize the MSE without needing loops or iteration.

Conditions:

• This formula only works if XtX is invertible.

• If not (e.g. multicollinearity), OLS becomes unstable or fails → use SVD or regularization.

• Fast and exact

• Very interpretable

Weaknesses

• Can overfit if:

• Too many features

• Highly correlated features

• Not enough data

A white background with black text

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

knn, ridge, lasso depend on the scale of the data

data preparation:

1. feature selection (remove unnecesary stuff)
2. missing data;
   1. if the feature is mostly NAN (>20%) remove it (row/col)
   2. impute
      1. numeric
         1. mean, median, mode ;
         2. 0
         3. knn / linear regression
         4. MICE
      2. categorical mode
         1. Mode
         2. "Missing"
         3. knn or other classifiers
         4. MICE
3. outliers
   1. Keep them if mild (>1.5 IQR ; Extreme >3 IQR)
   2. delete / impute => depends if they mean something
   3. transform => if they come from sharedness
4. **RMSE (Root Mean Squared Error)** tells us how far off, on average, our predictions are from the real values. For example, if RMSE is 50,000 DKK, that means the model usually predicts house prices with an error of about 50,000 DKK. Smaller RMSE means better predictions.
5. **R² (R-squared)** shows how well the model explains the variation in the data. If R² is 0.86, it means that 86% of the changes in house prices can be explained by the features (like size, number of rooms, etc.). The closer R² is to 1, the better the model is at explaining the data.