**Machine Learning**

Week 2

**Decision Trees**

A **Decision Tree** is a **tree-structured flowchart** used to make decisions by splitting data into smaller groups based on feature values.

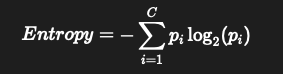
Each **internal node** represents a **feature**, each **branch** represents a **decision rule**, and each **leaf node** represents a **final output** (class label or value).

**How it Works?**

* **Start with all data at the root**
* **At each step**
  + Choose the best feature to split on (based on an impurity measure).
  + Split the dataset into branches.
* **Repeat recursively until**
  + All samples are classified
  + A stopping condition is met (max depth, min samples, etc.)

**Key Concepts**

* **Impurity Measures**
  + **Gini Impurity**
    - Measures how **Impure** a node is.
    - Lower is better.
  + **Entropy**



* + **Information Gain**



* + **Mean Squared Error (MSE)**
    - Used in Regression Trees.
* **Stopping Criteria**
  + Max depth of tree.
  + Minimum number of samples to split.
  + Minimum impurity decrease.
  + Leaf node purity.
* **Pruning (To Reduce Overfitting)**
  + **Pre-Pruning**: Stop tree growth early (e.g., limit depth)
  + **Post-Pruning**: Build full tree and **remove nodes** that don’t add value (based on validation set)

**Advantages**

* Easy to **visualize** and **interpret**
* Handles **non-linear data**
* Requires little data **preprocessing**
* Can handle **categorical** and **numerical** data
* Works well with **small to medium** datasets

**Disadvantages**

* **Overfits** easily on noisy data
* **Unstable**: small data changes can produce a very different tree
* **Greedy algorithm**: may not find the globally optimal tree
* Not great for **generalization** (unless pruned or regularized)

**Examples**

* Credit scoring
* Loan approval
* Medical diagnosis
* Customer churn prediction
* Fraud detection
* Marketing segmentation

**K-Nearest Neighbors**

k-Nearest Neighbors (kNN) is a non-parametric, lazy learning algorithm used for both classification and regression.

* It makes predictions based on similarity — that is, the idea that similar data points exist close to each other.
* It does not learn a model during training — instead, it stores the entire dataset and makes predictions at runtime.

**How it Works?**

1. **Calculate the distance** from the new point to all training data points (commonly using Euclidean distance).
2. **Select the k closest neighbors** (based on smallest distance).
3. For **classification**:
   1. Assign the **majority class** among the k neighbors.
4. For **regression**:
   1. Return the **average value** of the k neighbors.

**Distance Metrics**

* **Euclidean Distance** (default)



* **Manhattan Distance**



* **Cosine Similarity** (in high-dimensional text data)

## **Choosing the Right k**

* **Low k (e.g., 1–3)** → sensitive to noise, overfitting
* **High k (e.g., 10+)** → smoother boundary, but may underfit

Use **Cross-validation** to find optimal k

**Advantages**

* Very **simple** to implement and understand
* Naturally handles **multi-class classification**
* **No training time** (lazy learner)
* Performs well with **well-separated data**

**Disadvantages**

* Computationally expensive at inference (especially with large datasets)
* Slow prediction time (must compute distance to all points)
* Sensitive to irrelevant features & scale (feature scaling is crucial)
* Poor with high-dimensional data (curse of dimensionality)

**Examples**

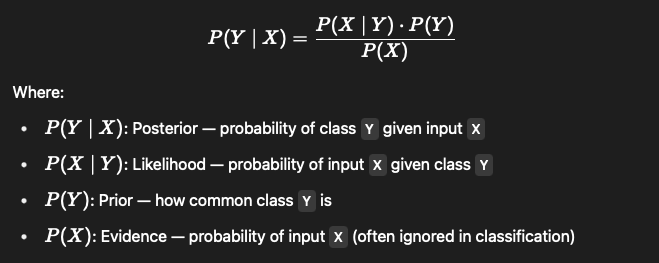
* Recommender Systems
* Text Categorization
* Handwriting Recognition (e.g., MNIST)
* Anomaly Detection
* Medical Diagnosis (e.g., tumor classification)

**Naive Bayes**

Naive Bayes is a family of probabilistic classifiers based on applying Bayes' Theorem with the naive assumption that features are independent given the class.

Despite the unrealistic assumption of independence, Naive Bayes works surprisingly well in many real-world scenarios, especially text classification (e.g., spam detection, sentiment analysis)

**Bayes Theorem**



**Types**

1. **Gaussian**
   * **Used When:** Features are continuous and normally distributed
   * **Examples:** Medical diagnosis
2. **Multinomial**
   * **Used When:** Features are discrete counts
   * **Examples:** Text classification
3. **Bernoulli**
   * **Used When:** Binary features (0/1)
   * **Examples:** Spam detection, sentiment
4. **Complement**
   * **Used When:** Modified version for imbalanced data
   * **Examples:** Text classification

**Advantages**

* **Fast and efficient** on large datasets
* Performs well even with **limited training data**
* **Works well with text** data (bag of words)
* Requires **very little training time**
* Handles **multi-class** problems naturally

**Disadvantages**

* Assumes **independence** between features (not always realistic)
* **Struggles with highly correlated** features
* May be **outperformed by more complex** models on structured/tabular data

**Train/Test Split**

The **Train/Test Split** is a **basic technique** to evaluate your machine learning model by dividing the dataset into:

* **Training Set** → used to **train** the model.
* **Test Set** → used to **evaluate** the model's performance on **unseen data**.

**Typical Split Ratios:**

* **80% Train / 20% Test**
* **70% Train / 30% Test**
* Depends on dataset size and problem type.

### **Advantages:**

* Simple and fast to implement
* Gives a quick estimate of model performance

### **Disadvantages:**

* **High variance**: performance depends on **how the data was split**
* May not generalize well if the dataset is small

**Cross-Validation (CV)**

Cross-validation is a **resampling technique** used to **evaluate model performance** more reliably by splitting the data **multiple times**.

**How it works?**

1. Split data into **k equal parts** (folds)
2. Train the model on **k-1 folds**
3. Test it on the **remaining fold**
4. **Repeat k times**, each time using a different fold as test
5. Average the scores

### **Advantages:**

* **More reliable** estimate of model performance
* Reduces **overfitting** risk
* Useful when you have **limited data**

**Disadvantages:**

* **Computationally expensive**
* Can be slow for large datasets and complex models

**Feature Scaling**

Feature Scaling is the process of **normalizing** or **standardizing** the range of independent variables (features) so that they contribute equally to the model.

**Why is it Needed?**

Many ML algorithms (like **KNN**, **SVM**, **Gradient Descent**, **Logistic Regression**) are **sensitive to the scale** of features.

* Example: A feature with values ranging from 1 to 1000 will dominate a feature ranging from 0 to 1.

**Types**

1. **Min-Max Scaling**

Scales values to a fixed range [0, 1].

1. **Standardization**

Centers data around 0 with standard deviation 1.

1. **MaxAbs Scaling**

Good for data that is already centered.

**Handle Missing Data**

Missing values can:

* **Distort model training**
* Cause **errors** in algorithms that don’t handle NaNs (like SVM, KNN)

**Types**

1. **Remove rows/columns**

**Example:** When missing data is small in percentage

1. **Mean/Median Imputation**

**Example:** For numerical features

1. **Mode Imputation**

**Example:** For categorical features

1. **KNN Imputer**

**Example:** Estimates missing values using neighbors

1. **Model-Based Imputer**

**Example:** Predict missing values using regression etc.

**Encoding Categorical Features**

Machine Learning models **only work with numbers**. So, we convert text (categorical data) into numeric values.

**Types**

1. **Label Encoding**

Converts categories to 0, 1, 2, 3, etc.

1. **One-Hot Encoding**

Creates binary columns for each category.

1. **Ordinal Encoding**

Maps values with a defined order.

1. **Target Encoding**

Uses mean of the target for encoding

**Feature Selection**

Feature selection is the process of **identifying and choosing a subset** of the most relevant features (variables or predictors) from the original dataset.

**Why is it Needed?**

* To remove **irrelevant, redundant, or noisy** features.
* To reduce **overfitting** and improve **model performance**.
* To **speed up training** and make the model more **interpretable**.

**Types**

* **Filter Methods**
  + Use statistical tests (e.g., chi-square, ANOVA) to score features.
  + Independent of ML model.
  + Example: SelectKBest
* **Wrapper Methods**
  + Use the model's performance to select features.
  + Computationally expensive.
  + Example: Recursive Feature Elimination (RFE)
* **Embedded Methods**
  + Perform selection **during** model training.
  + Example: Feature importance from Decision Trees, Lasso regression.

**Dimensionality Reduction**

Dimensionality reduction transforms the data from a **high-dimensional space** to a **lower-dimensional space**, while **preserving as much information as possible**.

**Why is it Needed?**

* To deal with the **curse of dimensionality**.
* To **remove multicollinearity**.
* To **compress data** while keeping important patterns.
* To **visualize** high-dimensional data in 2D/3D.

**Types**

* **Principal Component Analysis (PCA)**
  + Projects data onto **orthogonal axes (principal components)** that maximize variance.
  + Creates **new features**, not subsets of old ones.
  + Example: Reducing 100 features to 10 PCs that explain 95% of variance.
* **Linear Discriminant Analysis (LDA)**
  + Similar to PCA but uses **class labels** to maximize separation between categories.
* **t-SNE / UMAP**
  + Used for **visualization** (non-linear reductions).
  + Not good for predictive modeling.