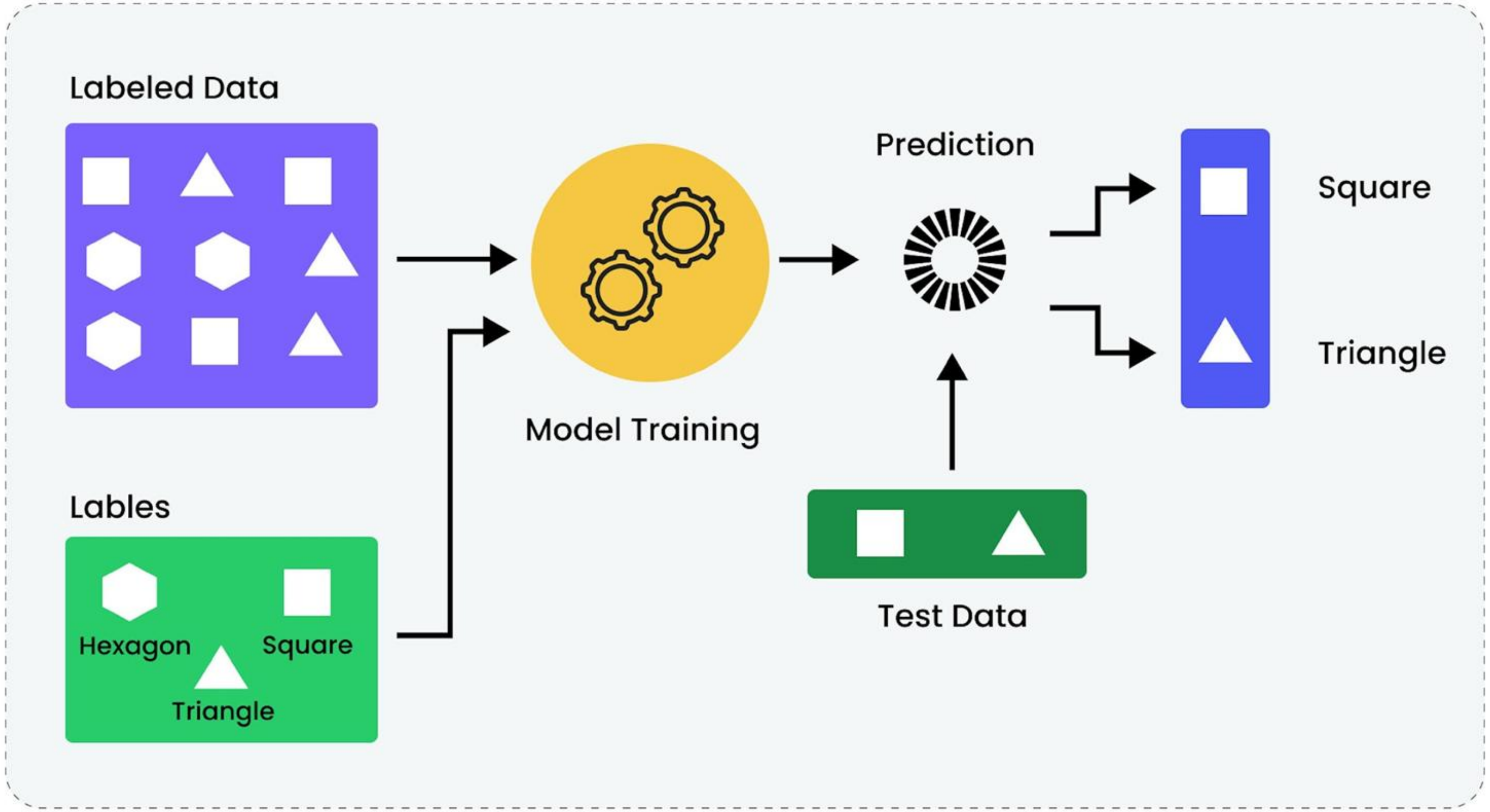
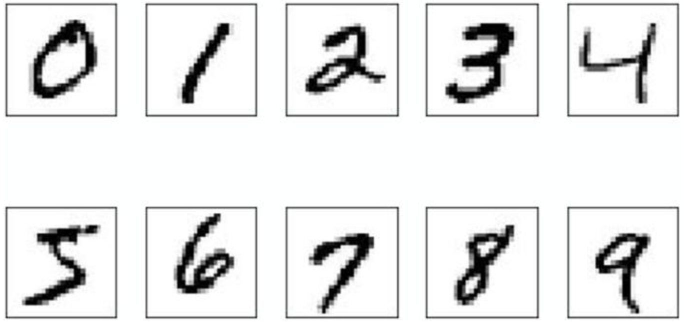
SUPERVISED LEARNING PART 1: CLASSIFICATION



Supervised Learning

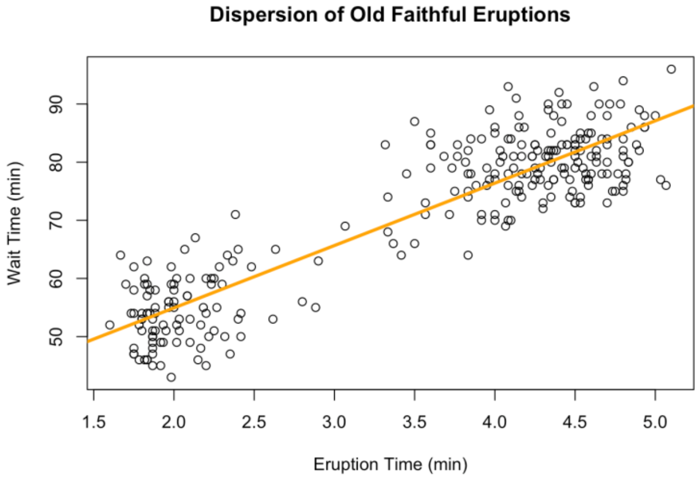
Classification

- Predict categories
- Validation:
  - Accuracy (and many more!)
- Example: Which digit is it?



Regression

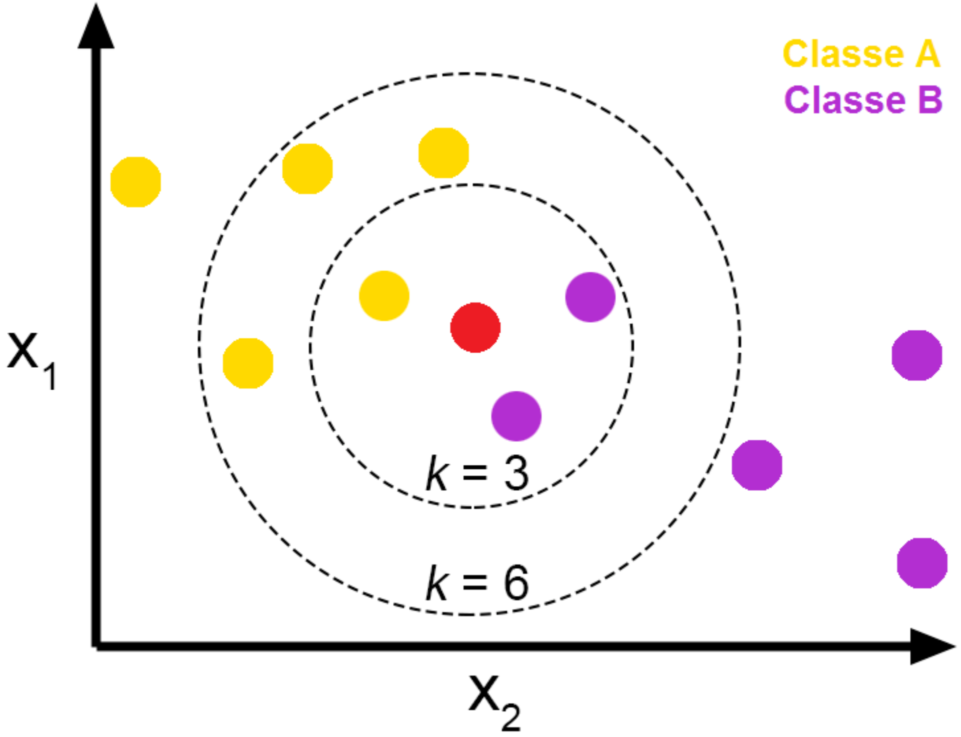
- Predict continuous values
- Validation:
  - Mean-squared error (more to come)
- Example: How long must we wait?



K-Nearest Neighbors (KNN)

K-NEAREST NEIGHBORS (KNN)

- **K-Nearest Neighbors (KNN)** is a **non-parametric**, **instance-based** learning algorithm used for classification and regression
- **(KNN)** algorithm is considered a **supervised learning** algorithm because it uses labeled training data to make predictions about new, unseen data points.
- **Main Idea:** When given a new data point, KNN looks at the *k* closest data points in the training set (neighbors) and makes predictions based on their labels.
- Instead of building a mathematical model or decision boundary during training, KNN stores the labeled training data.

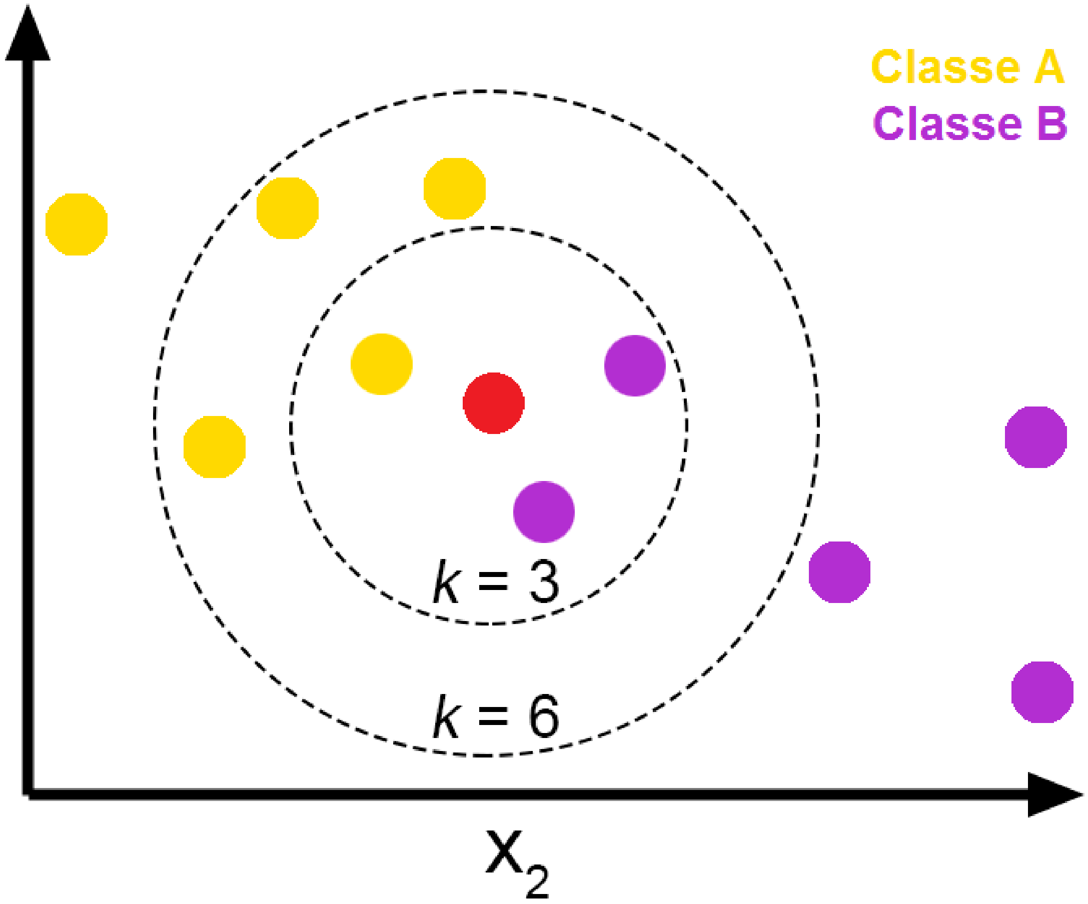


K-NN Algorithm

Steps:

1. Calculate the distance between the new point and all training points.
2. Sort the distances and find the k nearest neighbors.
3. Predict the label based on the neighbors.

- For classification, the majority label among the neighbors is chosen.
- For regression, the average of the neighbors' values is used.



K-NN SUMMARY

Choosing k :

- Small k : Sensitive to noise, can overfit.
- Large k : Smooths boundaries, but may underfit.

Distance Metrics: Common ones are:

- Euclidean Distance (default in Scikit-learn)
- Manhattan Distance
- Minkowski Distance (generalization)
- Cosine Distance (based on angles)

IMPORTANT Scaling Features: Since KNN is distance-based, ensure features are on a similar scale (e.g., use Min-Max Scaling or Standardization).

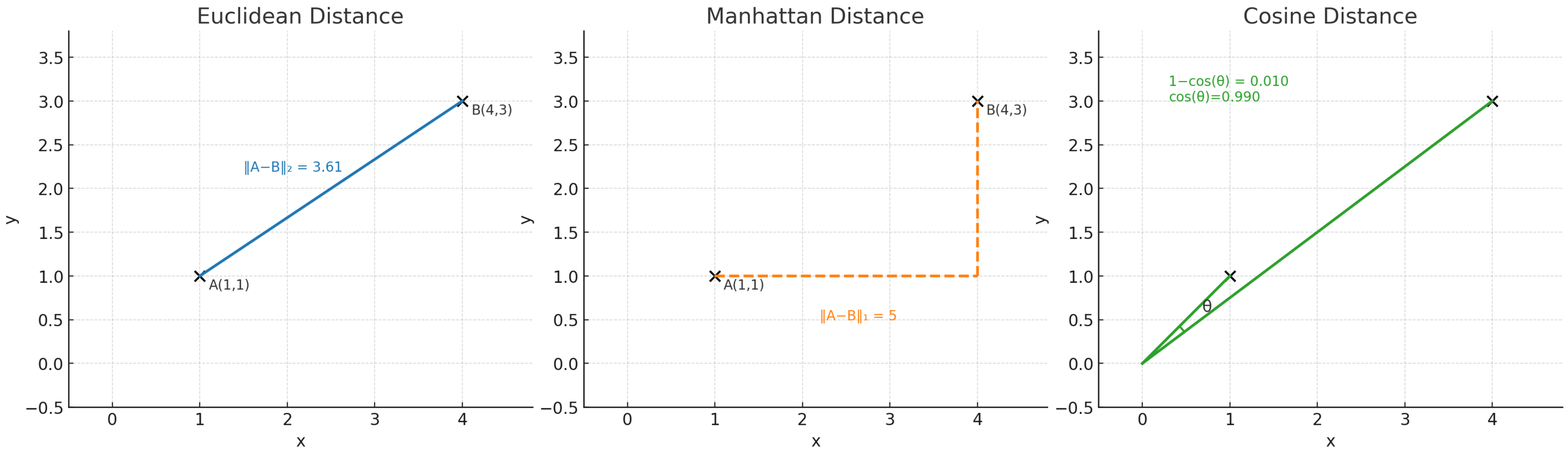
Advantages:

- Simple to understand and implement.
- No explicit training phase; training data is directly used.

Disadvantages:

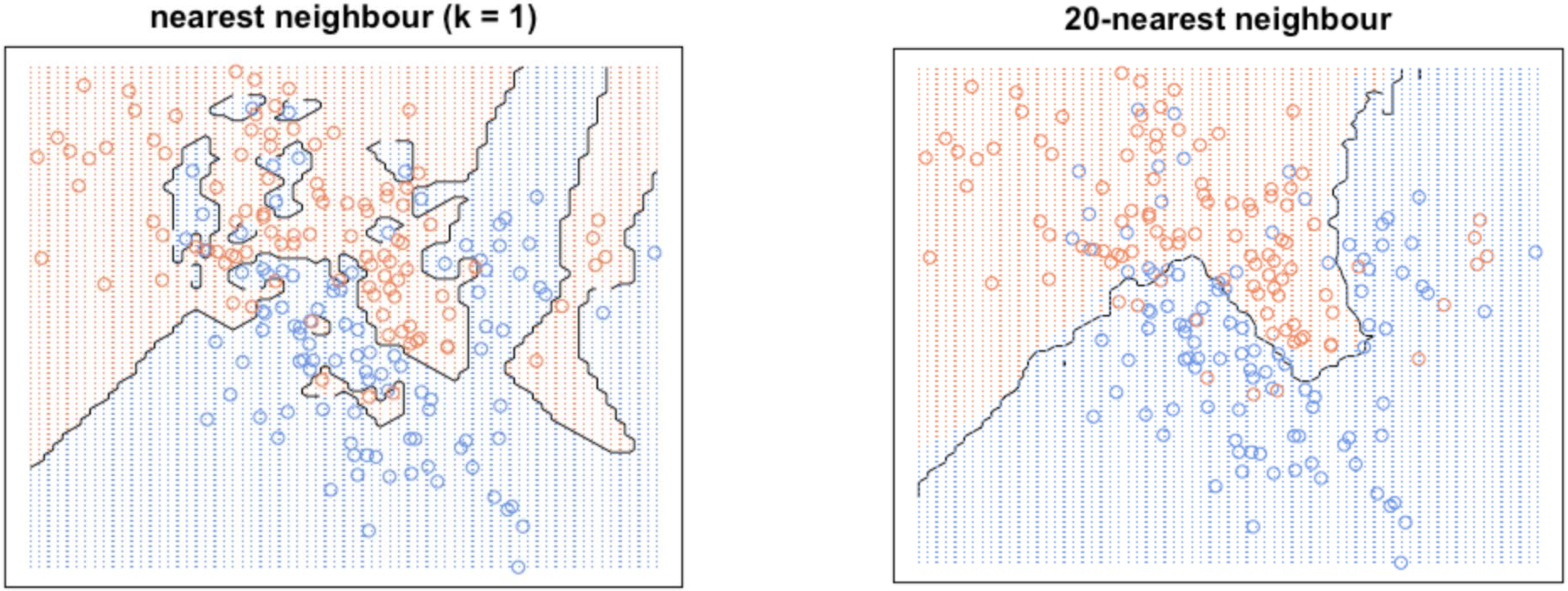
- Computationally expensive for large datasets.
- Sensitive to irrelevant features and feature scaling.

Comparison of Euclidean, Manhattan, and Cosine Distances



Choice of Distance

- Euclidean most common
- Cosine often used for high-dimensional data



nearest neighbour (k = 1)

20-nearest neighbour

Effect of k on Decision Boundary

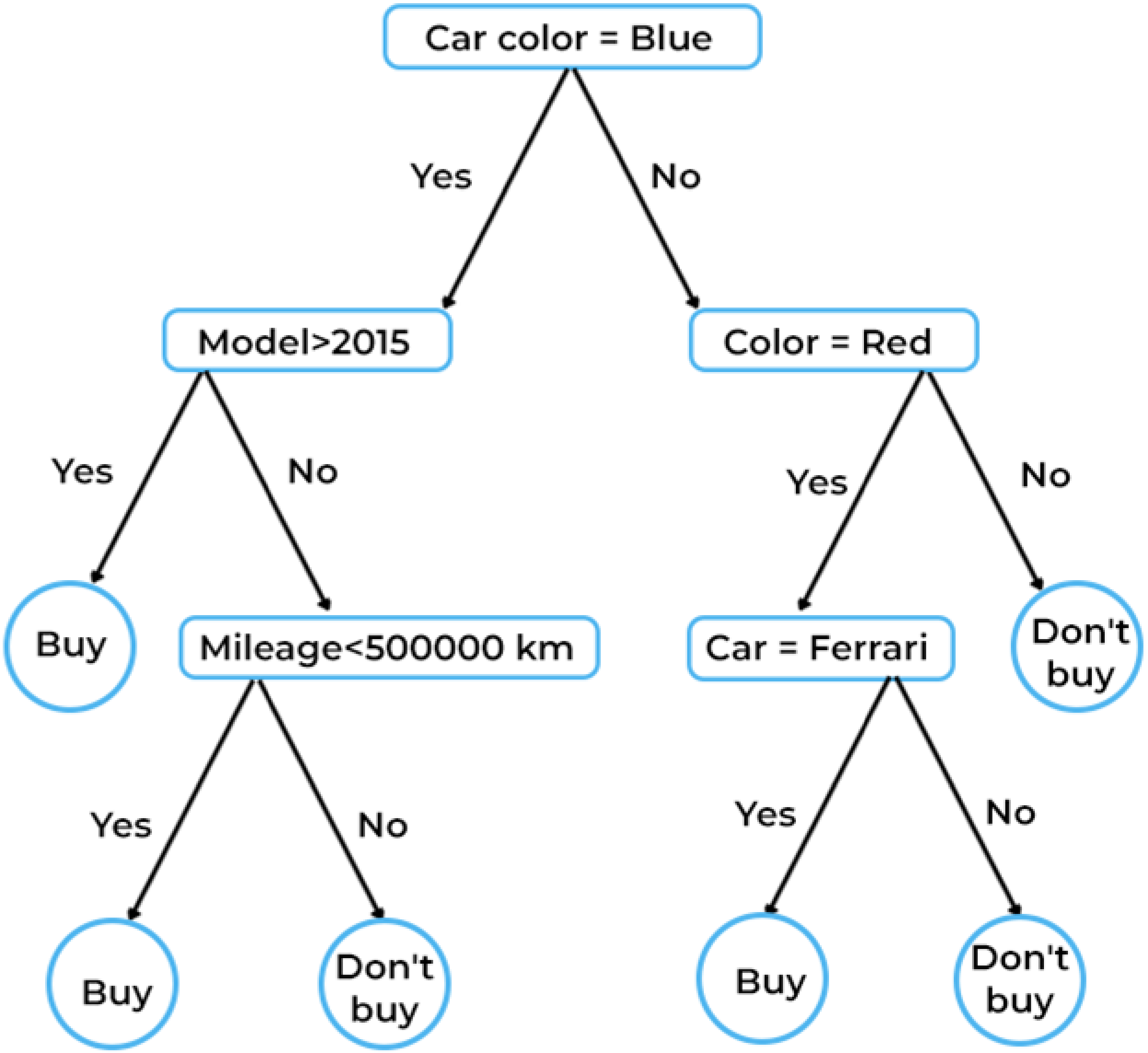
https://fderyckel.github.io/machinelearningwithr/knnchapter.html

Decision Trees

DECISION TREES

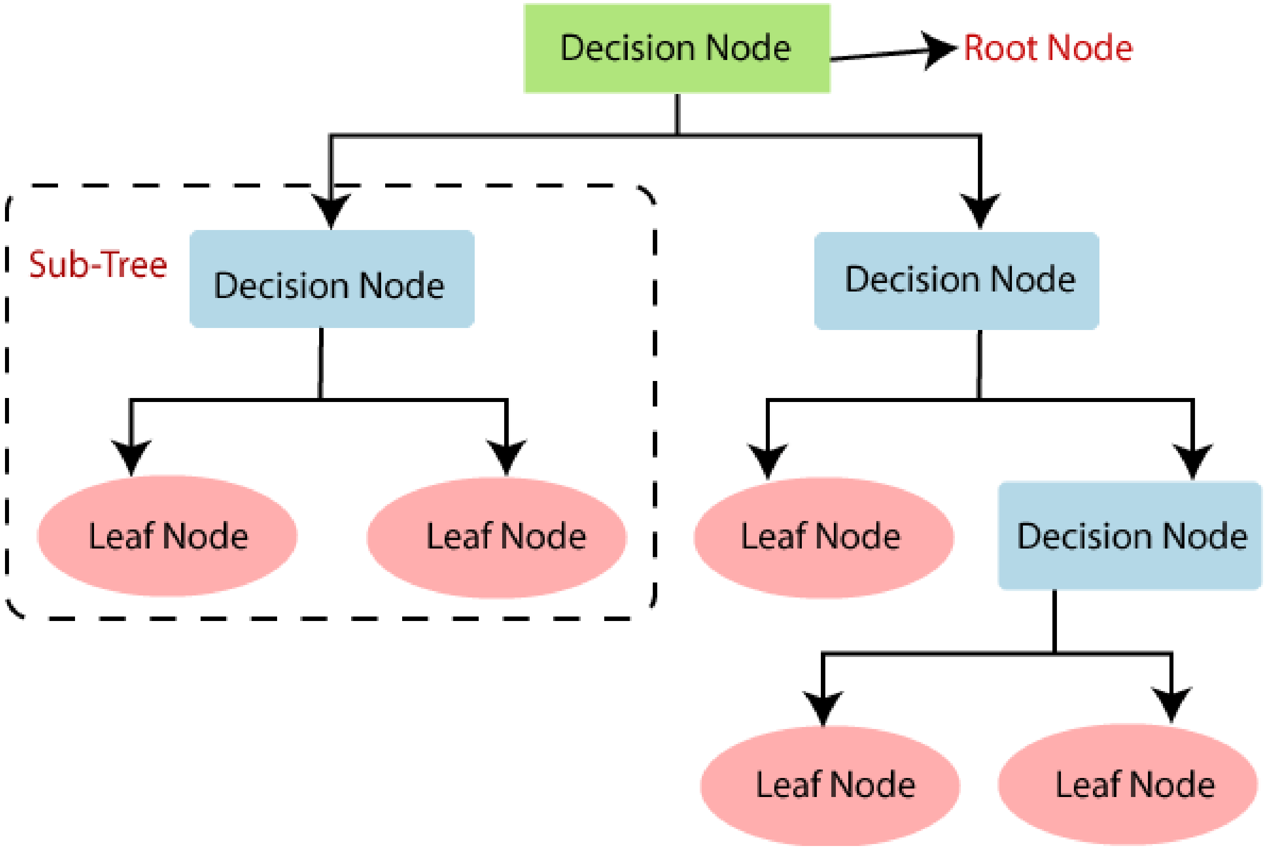
- **Classification** and **regression** tasks.
- They are explainable
- Can model non-linear relationships
- Handles numeric and categorical data
- Works with missing data*
- No feature scaling required
- Prone to data overfitting, instability, bias

BUYING A CAR



DECISION TREES DEFINITIONS

- **Root Node:** All data starts here.
- **Branch or Sub-Tree:** A subset (partition) of the tree.
- **Splitting**: Dividing a node into two or more sub-nodes.
- **Decision Node:** Created by splitting.
- **Leaf or Terminal Node:** Decisions are mad here.
- **Pruning:** Removing a sub-node from the tree is called pruning.

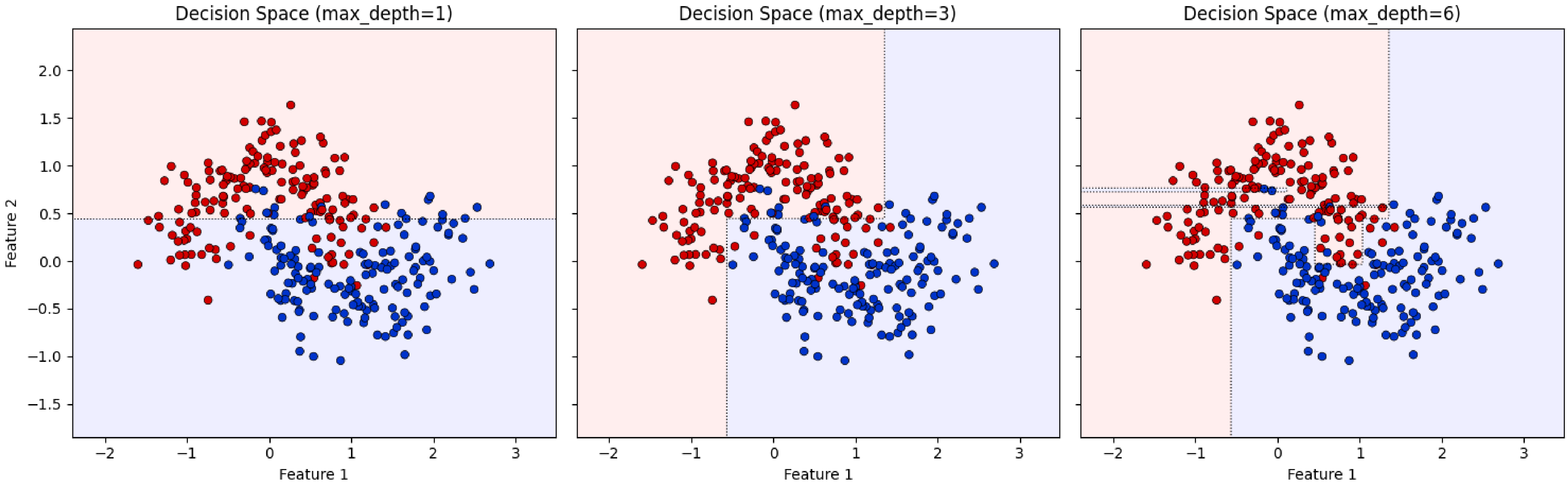


BYU

DECISION TREE HYPERPARAMETERS (Python)

- **max_depth**: The maximum depth of the tree. Limiting this helps prevent overfitting.
- **min_samples_split**: The minimum number of samples required to split an internal node.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node.
- **max_features**: The number of features to consider when looking for the best split.

Decision Tree Decision Boundary by Max Depth



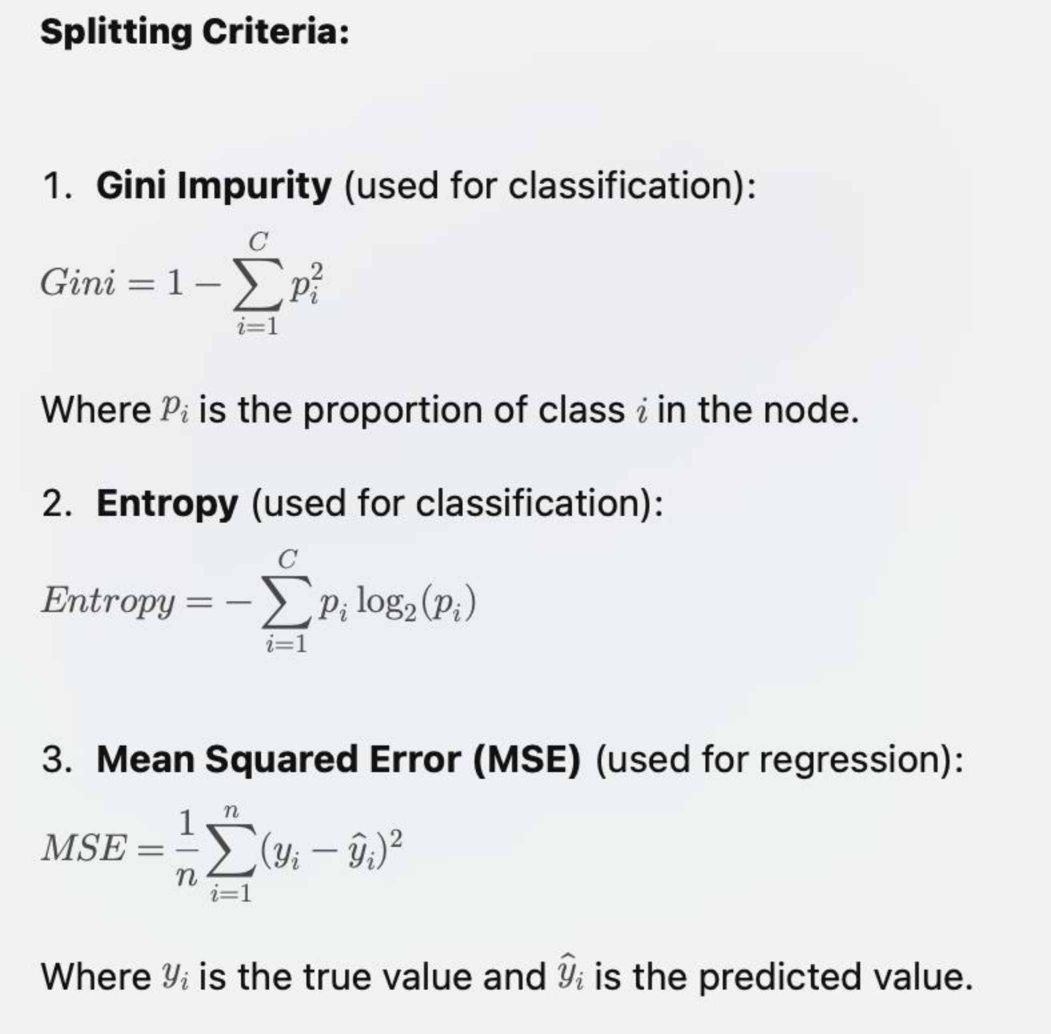
BYU

TREE BUILDING PROCESS

1. **Start at the root**: Evaluate all possible splits based on features.

2. **Choose the best split** based on a criterion (How to choose best?).

3. **Repeat** the process recursively until a stopping criterion is met (e.g., max depth, minimum samples per leaf).



Splitting Criteria:

1. **Gini Impurity** (used for classification):

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

Where p_i is the proportion of class i in the node.

2. **Entropy** (used for classification):

$$Entropy = - \sum_{i=1}^{C} p_i \log_2(p_i)$$

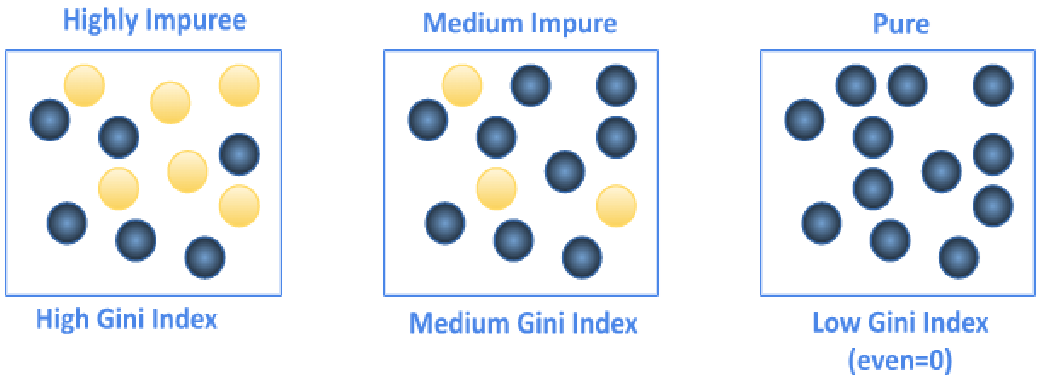
3. **Mean Squared Error (MSE)** (used for regression):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

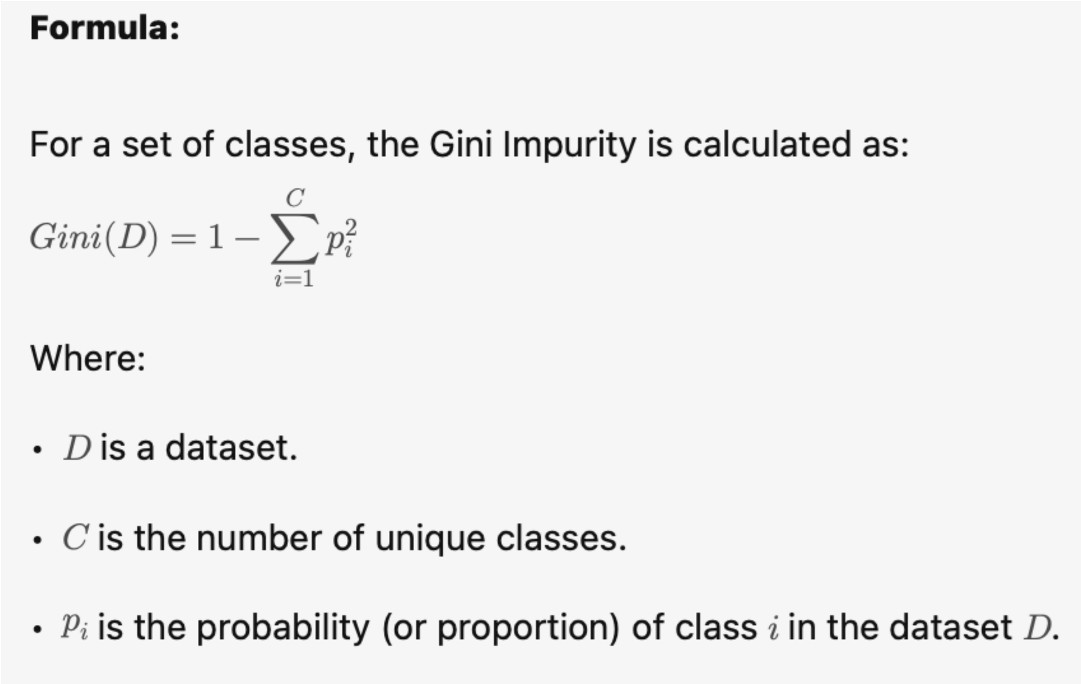
Where y_i is the true value and $\hat{y}_i$ is the predicted value.

GINI IMPURITY

- The **Gini Impurity** measure quantifies the “impurity” or **impurity score** of a node.
- A **lower Gini score** means a **purer** node.



- If all data points in a node belong to a single class, the Gini Impurity is 0 (perfect purity).
- If the data points are evenly distributed among all classes, the Gini Impurity is at its maximum value (1 – 1/C)



Formula:

For a set of classes, the Gini Impurity is calculated as:

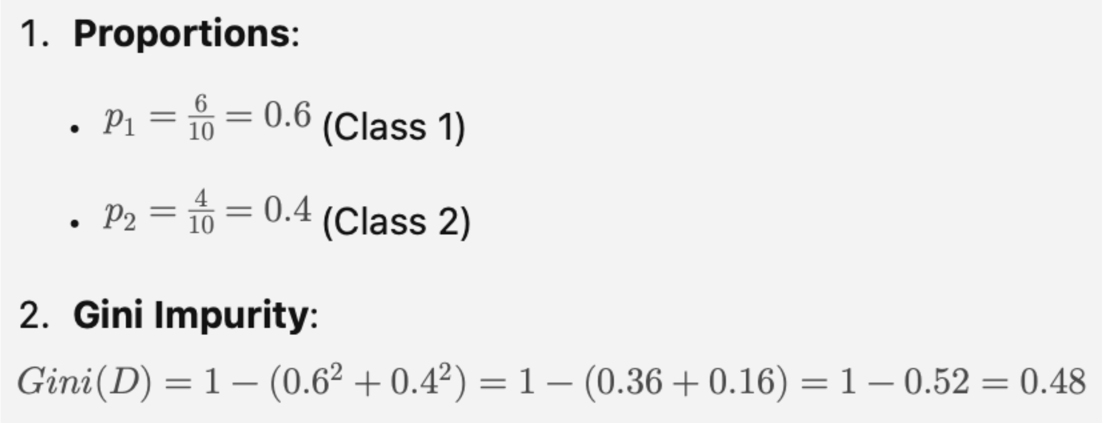
$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2$$

Where:

- D is a dataset.
- C is the number of unique classes.
- p_i is the probability (or proportion) of class i in the dataset D.

Example Calculation:

Let’s say we have a dataset of 10 samples belonging to two classes: Class 1 (6 samples) and Class 2 (4 samples)

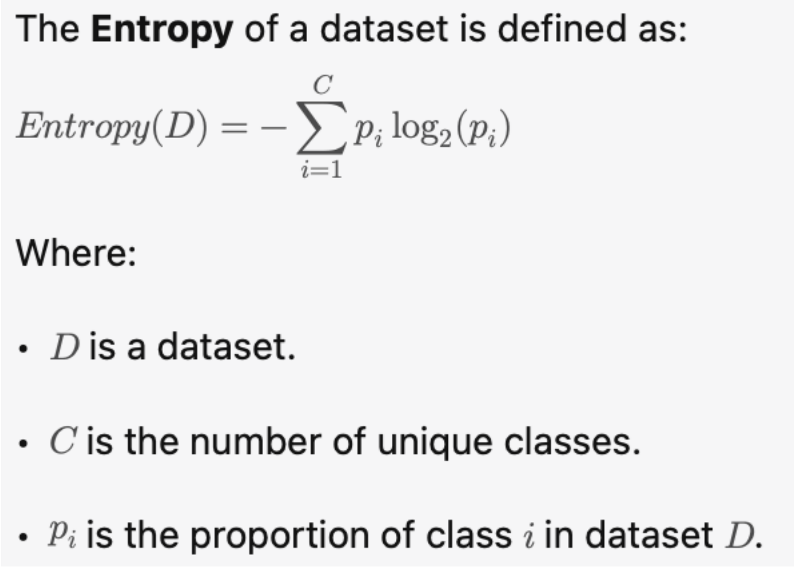


1. **Proportions**:
   - $p_1 = \frac{6}{10} = 0.6$ (Class 1)
   - $p_2 = \frac{4}{10} = 0.4$ (Class 2)
2. **Gini Impurity**:

$$Gini(D) = 1 - (0.6^2 + 0.4^2) = 1 - (0.36 + 0.16) = 1 - 0.52 = 0.48$$

ENTROPY

- **Entropy** is another measure of impurity or disorder, derived from information theory. It is used in decision trees to decide the best split by evaluating how much uncertainty (or entropy) is present in the data.



The **Entropy** of a dataset is defined as:

$$Entropy(D) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Where:

- D is a dataset.
- C is the number of unique classes.
- p_i is the proportion of class i in dataset D.

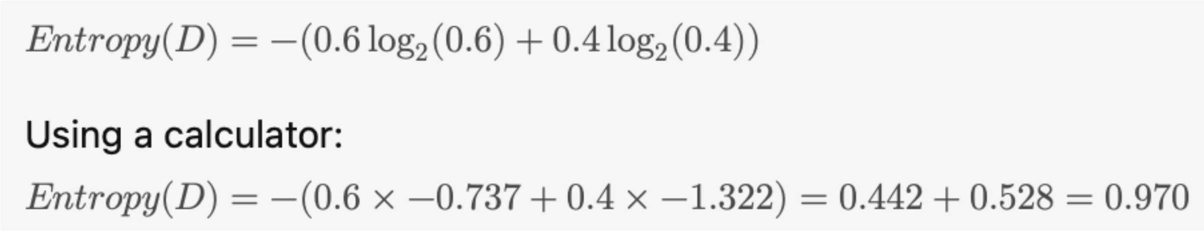
How to Interpret Entropy:

• If all samples in a node belong to the same class, entropy is 0 (pure node).

• If the samples are evenly distributed among all classes, entropy reaches its maximum, which is $\log_2(C)$ (where C is the number of classes).

Example Calculation:

Let's consider the same dataset of 10 samples with 6 samples of Class 1 and 4 samples of Class 2.



$$Entropy(D) = -(0.6\log_2(0.6) + 0.4\log_2(0.4))$$

Using a calculator:

$$Entropy(D) = -(0.6 \times -0.737 + 0.4 \times -1.322) = 0.442 + 0.528 = 0.970$$

Entropy vs. Gini Impurity

- Gini Impurity
  - Faster to compute (no logarithms).
  - Prefers pure splits, leading to shallower trees.
  - Default in CART (e.g., Scikit-learn).

- Entropy
  - Slower (uses logarithms) but handles rare classes better.
  - Encourages more balanced splits, sometimes leading to deeper trees.
  - Used in ID3, C4.5 (Information Gain-based methods).

- Key Takeaways:
  - Use Gini when speed and simplicity matter.
  - Use Entropy for imbalanced datasets and finer splits.
  - Differences are usually small in practice.

Logistic Regression

LOGISTIC REGRESSION MATH BACKGROUND

LOGISTIC REGRESSION MATH BACKGROUND

- Logistic Regression models the **probability** that an input belongs to a particular class.
- The linear model outputs real-valued numbers.
- Predicted values are mapped between 0 and 1.
- The logistic (sigmoid) function is defined as:

$$\sigma(t) = \frac{1}{1+e^{-t}}$$

Odds and Log-Odds:

- Odds represent the ratio of the probability of an event occurring to it not occurring:

$$\text{Odds} = \frac{p}{1-p}$$

- Taking the logarithm of the odds yields the log-odds (logit function):

$$\text{Logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

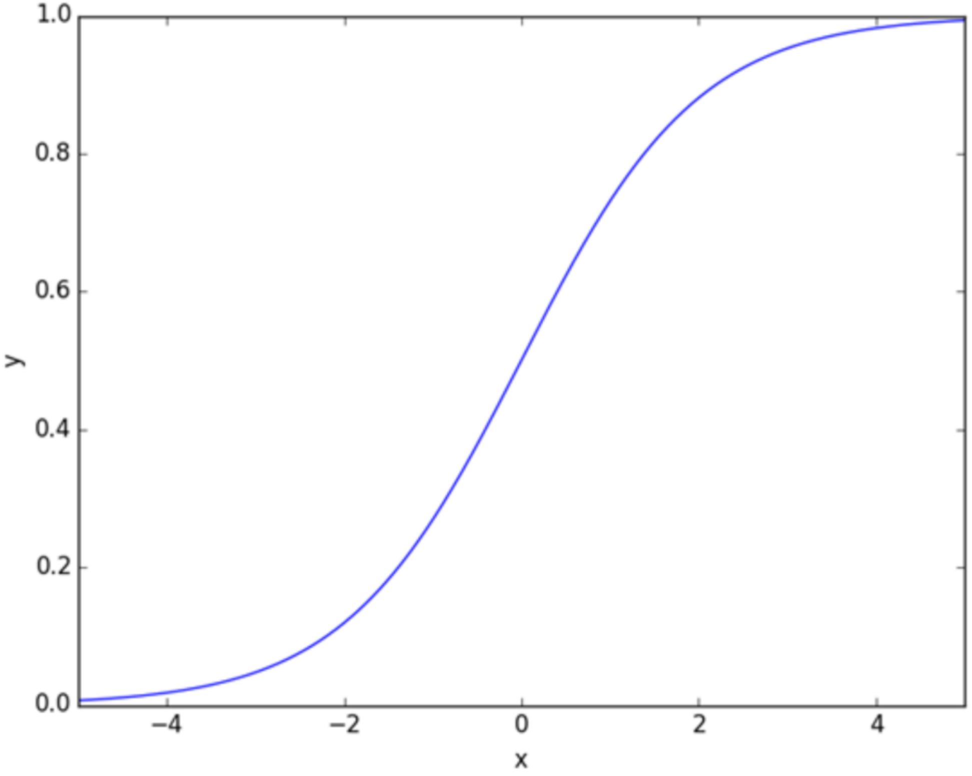
- This linear relationship between log-odds and input features forms the basis of logistic regression.

BYU

LOGISTIC REGRESSION FUNCTIONS

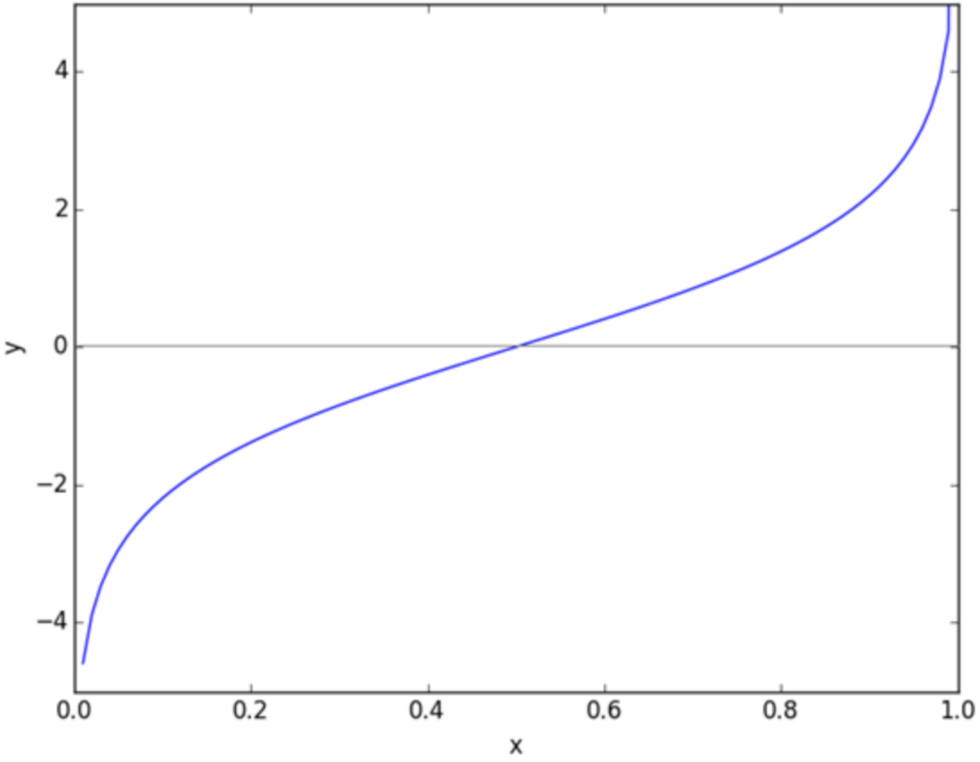
- **Sigmoid Function – Number to Probability**

$$\sigma(t) = \frac{1}{1+e^{-t}}$$



- **Logit Function – Probability to Number**

$$\text{Logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$



LOGISTIC REGRESSION

Problem Types:

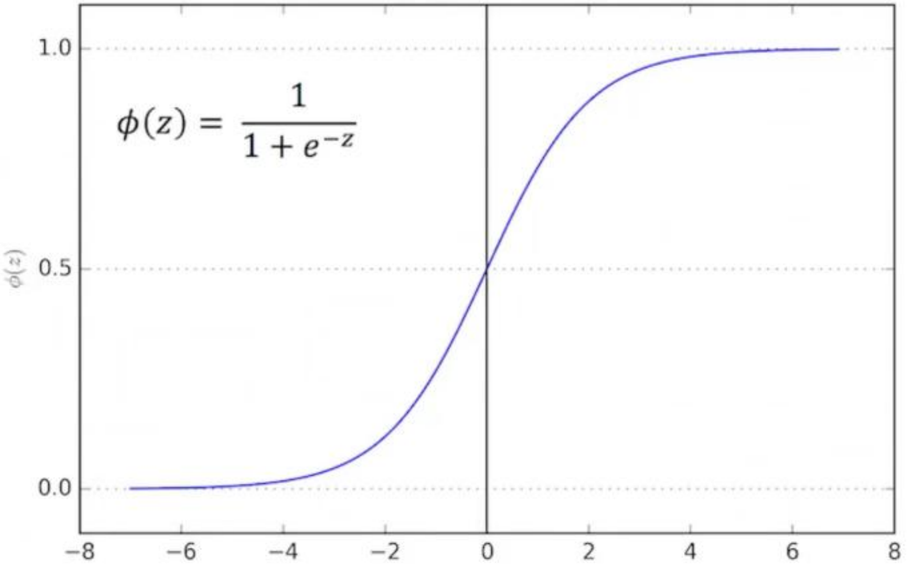
- **Binary Classification**: Predicts one of two possible outcomes (e.g., spam vs. non-spam emails).
- **Multi-Class Classification**: Extended by using techniques like One-vs-Rest or Softmax for handling multiple classes.

Advantages:

- **Quick**: Able to apply to very large datasets
- **Interpretability**: Weights provide insights into feature importance.
- **Probabilistic Outputs**: Predicts probabilities for confidence estimation.

Limitations:

- **Linearity**: Assumes a linear relationship between input features and log-odds of the outcome.
- **Outliers**: Sensitive to outliers; regularization helps mitigate this.
- **Multicollinearity**: Correlated features can reduce interpretability.



BYU

INTERPRETING LOGISTIC REGRESSION COEFFICIENTS

- Each coefficient (β_i) represents the **change in the log-odds** of the outcome for a one-unit increase in X_i, **holding other variables constant**.
- Exponentiating gives the **odds ratio**:
  - $e^{\beta_i} > 1$: increase odds of the event
  - $e^{\beta_i} < 1$: decrease odds of the event
- Example:
  - If $\beta_1 = 0.7$, then $e^{\beta_1} = e^{0.7} \approx 2.01$
  - A unit increase in X_1doubles the odds of the outcome!

$$\text{Logit}(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

Support Vector Machine (SVM)

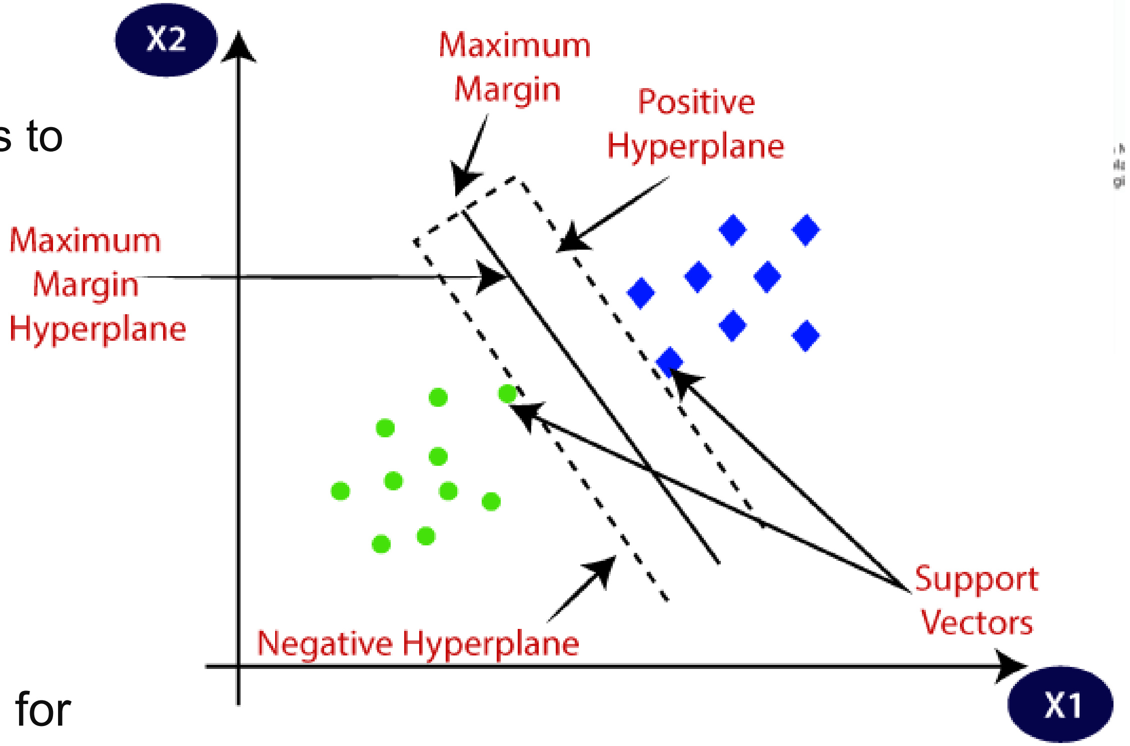
SUPPORT VECTOR MACHINE (SVM)

Core Idea: SVM finds the hyperplane that maximizes the (soft) margin between the closest data points (support vectors) of different classes.

- For non-linearly separable data, SVM uses kernel functions to project data into a higher-dimensional space where it becomes linearly separable.

Key Concepts:

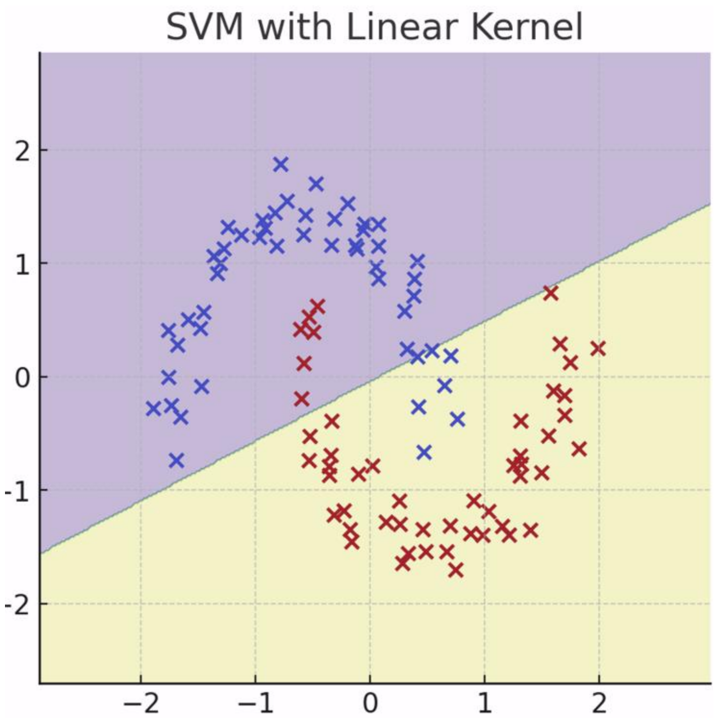
- **Hyperplane**: A decision boundary.
- **Margin**: Distance between the hyperplane and the nearest data points from either class.
- **Kernel Trick**: Transforms data into higher dimensions for linear separability. (This is often used for non-linear problems)



SVM Kernel Functions

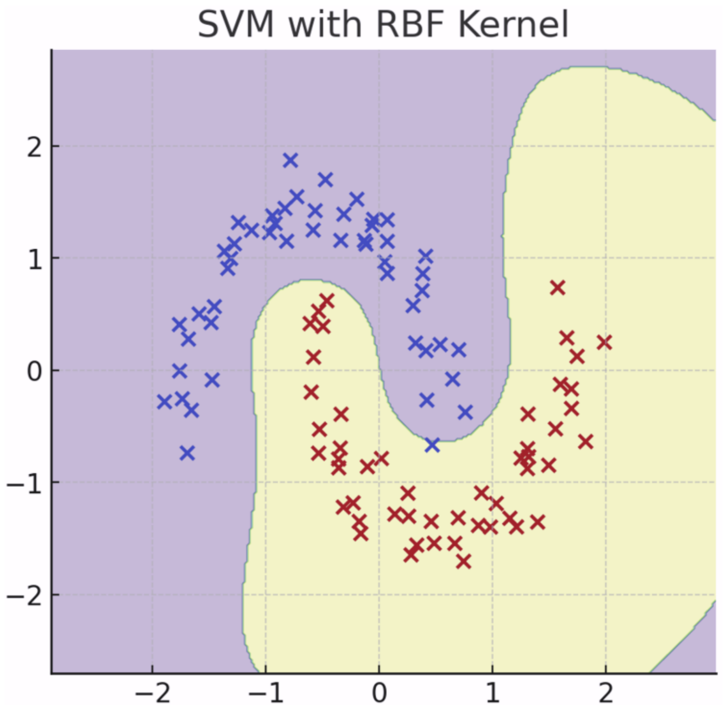
- Linear Kernel

$$K(x_i, x_j) = x_i^T x_j$$


SVM with Linear Kernel

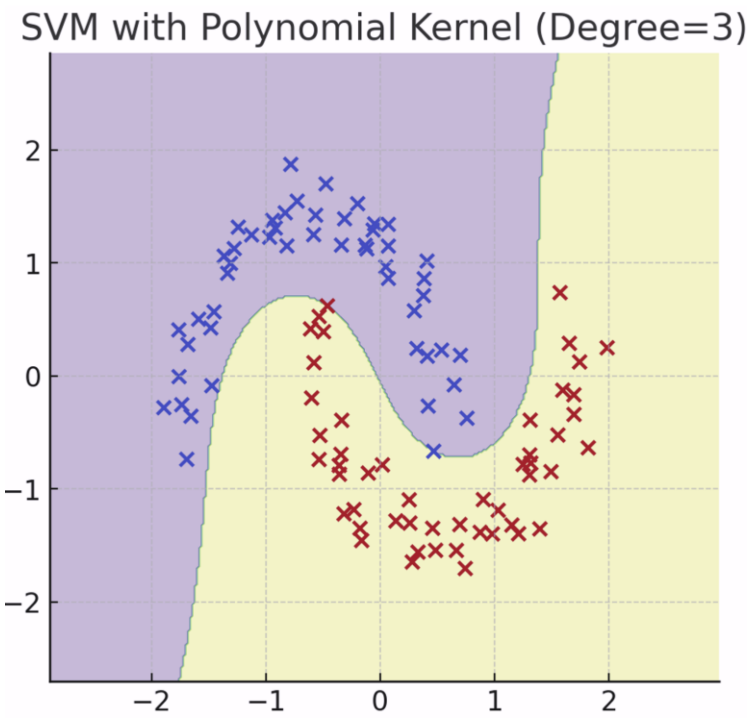
- Radial Basis Function (RBF)

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$


SVM with RBF Kernel

- Polynomial Kernel

$$K(x_i, x_j) = \left(x_i^T x_j + c\right)^d$$


SVM with Polynomial Kernel (Degree=3)

SVM KEY PARAMETERS

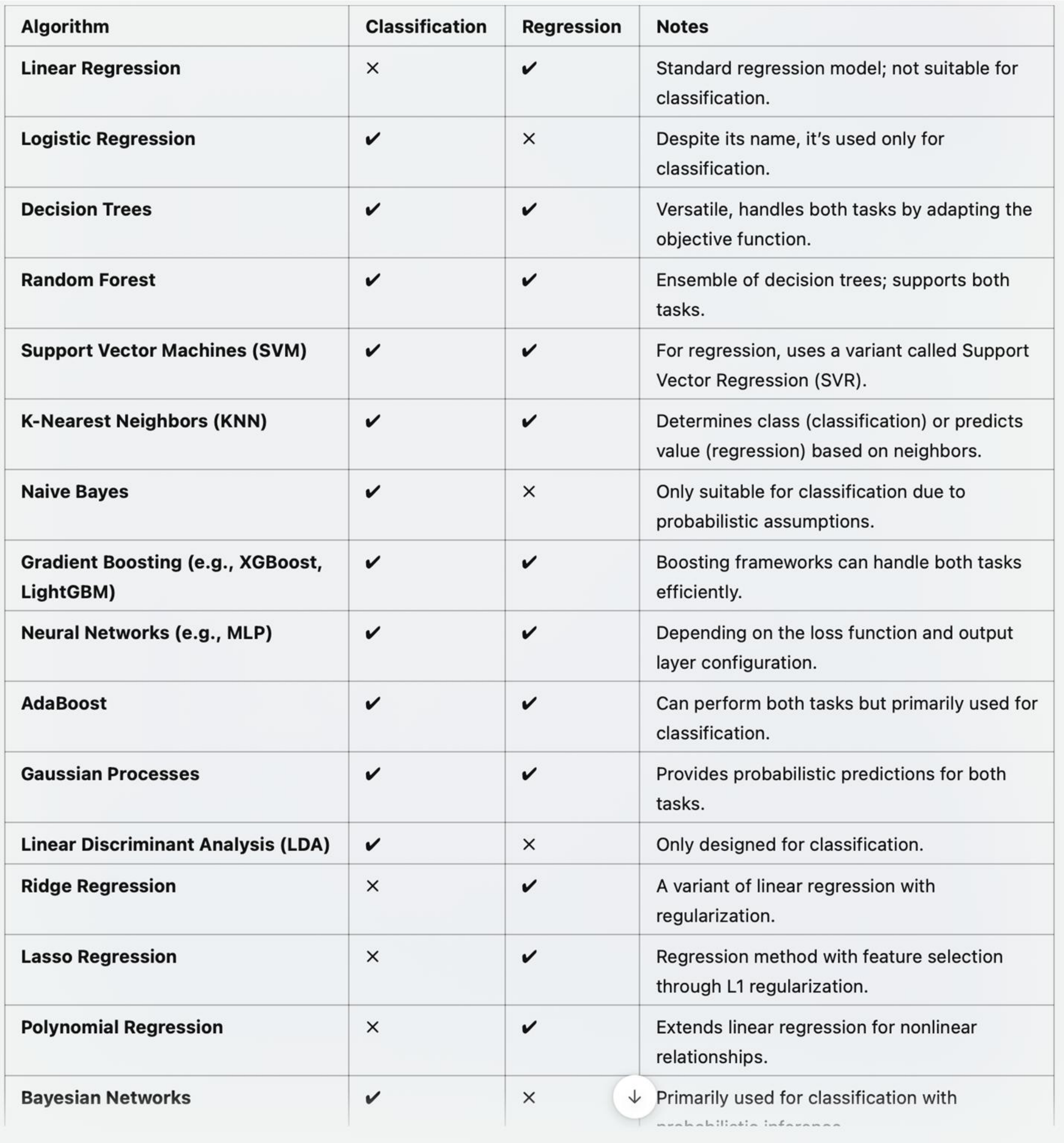
- **C**: Regularization parameter.
  - High **C** means the model tries to classify all points correctly (narrow margin), while
  - Low **C** allows more misclassifications (wide margin).
  - **Opposite effects of choice of k w/ k-NN**
- **kernel**: Defines the type of hyperplane (linear, rbf, poly, etc.).
- **gamma**: Defines the influence of a single training point (applicable for RBF and polynomial kernels).
- **degree**: Degree of the polynomial kernel.

Advantages:

- Effective for high-dimensional data.
- Works well with a clear margin of separation.

Disadvantages:

- Computationally expensive for large datasets.
- Choice of kernel and hyperparameters can be highly problem dependent.



Algorithm	Classification	Regression	Notes
Linear Regression	×	✔	Standard regression model; not suitable for classification.
Logistic Regression	✔	×	Despite its name, it's used only for classification.
Decision Trees	✔	✔	Versatile, handles both tasks by adapting the objective function.
Random Forest	✔	✔	Ensemble of decision trees; supports both tasks.
Support Vector Machines (SVM)	✔	✔	For regression, uses a variant called Support Vector Regression (SVR).
K-Nearest Neighbors (KNN)	✔	✔	Determines class (classification) or predicts value (regression) based on neighbors.
Naive Bayes	✔	×	Only suitable for classification due to probabilistic assumptions.
Gradient Boosting (e.g., XGBoost, LightGBM)	✔	✔	Boosting frameworks can handle both tasks efficiently.
Neural Networks (e.g., MLP)	✔	✔	Depending on the loss function and output layer configuration.
AdaBoost	✔	✔	Can perform both tasks but primarily used for classification.
Gaussian Processes	✔	✔	Provides probabilistic predictions for both tasks.
Linear Discriminant Analysis (LDA)	✔	×	Only designed for classification.
Ridge Regression	×	✔	A variant of linear regression with regularization.
Lasso Regression	×	✔	Regression method with feature selection through L1 regularization.
Polynomial Regression	×	✔	Extends linear regression for nonlinear relationships.
Bayesian Networks	✔	×	Primarily used for classification with probabilistic inference