Supervised Learning – Classification

- In supervised learning, the algorithm is trained on a dataset that has input features (X) and labels/targets (y).
- The model learns a mapping function $f(X) \rightarrow y$.
- Later, when we feed new unseen data (X_new), it predicts the label (y_pred).

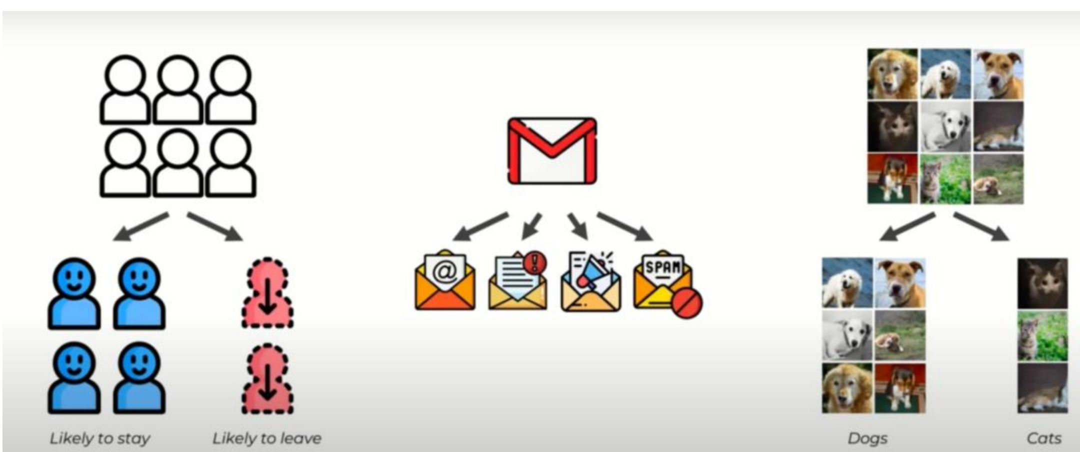
What is classification?

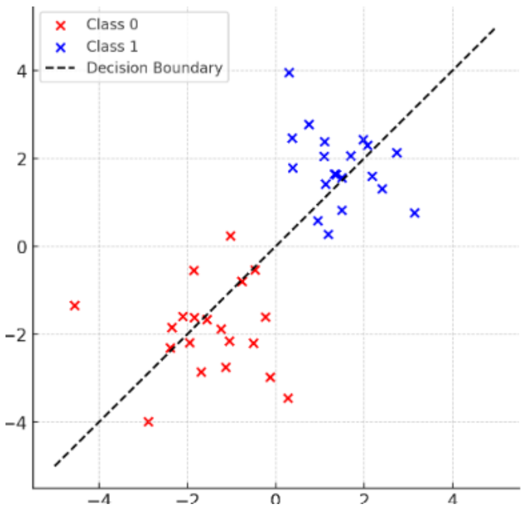
A process where a machine learning model is trained on labeled data to predict the category or class of new, unseen observations.

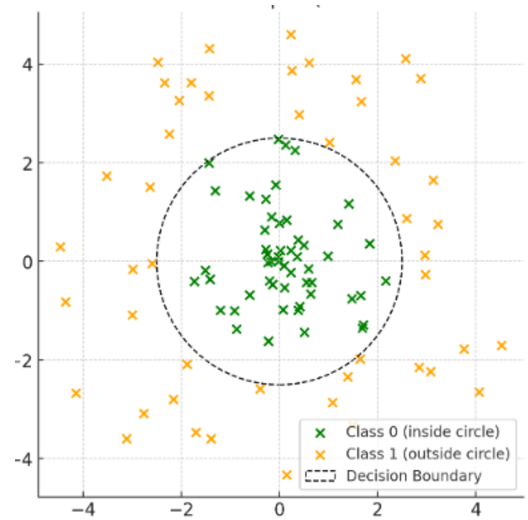
For example, in email spam filtering, a classification model is trained with emails labeled as "spam" or "not spam." Once trained, it predicts whether a new email falls into the "spam" or "not spam" category based on its features.

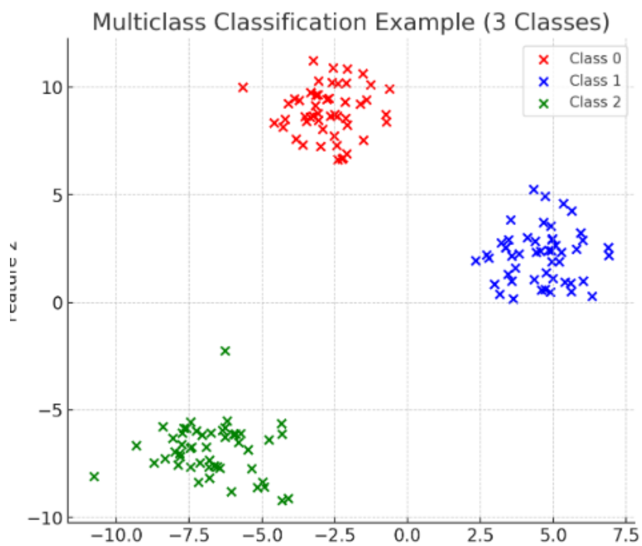
Key points about classification

- It is used when the output variable is categorical (not numeric), with two or more distinct classes (e.g., animal types, disease presence, or sentiment categories).
- The model trains on data pairs composed of feature sets and corresponding class labels.
- Common algorithms include logistic regression, decision trees, support vector machines (SVM), random forests, and k-nearest neighbors (KNN).
- The output of classification is a discrete value—a predicted category—rather than a continuous number as in regression tasks.

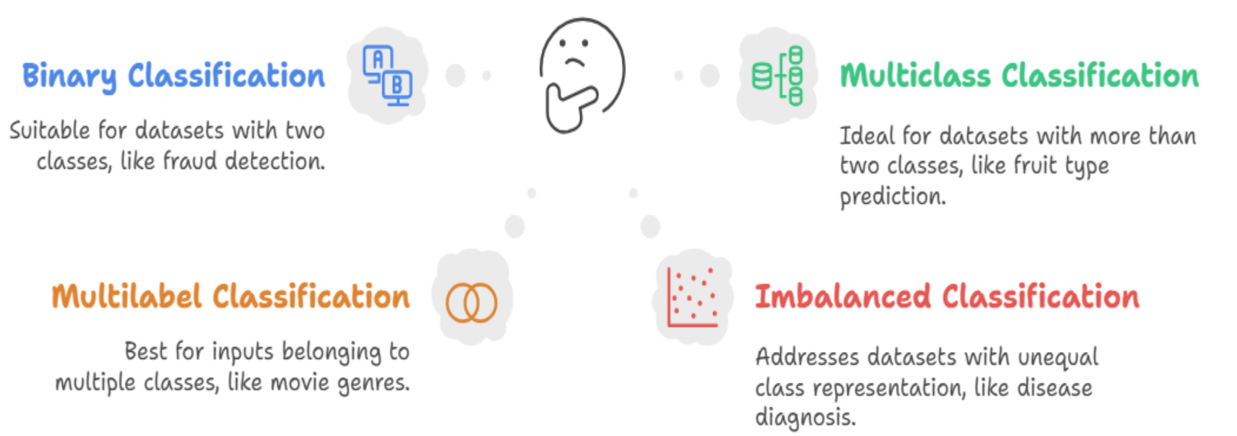








Types:



Example

1. Fraud detection
    - 99% transactions → legitimate
    - 1% transactions → fraudulent
    - If the model predicts "legitimate" for everything → it's 99% accurate but useless at detecting fraud.
2. Medical diagnosis
    - 95% patients → healthy
    - 5% patients → diseased
    - Model may always predict "healthy," missing sick patients.

Comparison between Regression and Classification

Aspect	Regression	Classification
Definition	Predicts a continuous numeric value.	Predicts a categorical label/class.
Output	Real numbers (e.g., 45.7, 100, 12.5).	Discrete categories (e.g., "Spam / Not Spam", "Cat / Dog / Horse").
Examples	- Predicting house prices 🏠- Predicting stock prices 💹- Predicting temperature 🌡	- Email spam detection 📧- Disease diagnosis (cancer vs no cancer) 🛡- Image recognition (cat/dog) 🐱🐶
Algorithms	- Linear Regression- Polynomial Regression- Ridge/Lasso Regression	- Logistic Regression- Decision Trees- Random Forest- SVM- Neural Networks
Evaluation Metrics	- MSE (Mean Squared Error)- RMSE- MAE- R2R^2 score	- Accuracy- Precision- Recall- F1-score- ROC-AUC
Decision Boundary	No explicit boundary (predicts numbers).	Yes, separates classes using a boundary.
Nature of Target Variable	Continuous (infinite possible values).	Categorical (finite classes).

Classification models

Classification models are algorithms used in supervised learning to predict categorical outputs.

Types of Classification Models

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. k-Nearest Neighbors (k-NN)
6. Support Vector Machine (SVM)
7. Neural Networks (Deep Learning)

Parametric and Non Parametric algorithms

Feature	Parametric	Non-Parametric
Assumption	Fixed form (e.g., linear)	No fixed form
Parameters	Finite, fixed size	Grow with data
Examples	Logistic Regression, Naive Bayes	k-NN, Decision Trees
Flexibility	Less flexible	Very flexible
Data Need	Works well with small data	Needs more data
Speed	Fast training, prediction	Slower (esp. with large data)
Overfitting	Less prone	More prone

explore how classification is applied in different industries, identify the type of classification problem (binary / multiclass / multilabel), in atleast two industries- T

Logistic Regression

🔹 1. Overview

- Logistic Regression is a classification algorithm (not regression, despite the name).
- It predicts the probability that a given input belongs to a particular class.
- Output is between 0 and 1, using the sigmoid function.

🔹 2. Types of Logistic Regression

1. Binary Logistic Regression – output has 2 classes (Yes/No, Spam/Not Spam).
2. Multinomial Logistic Regression – output has 3+ classes without order (Iris dataset: Setosa, Versicolor, Virginica).
3. Ordinal Logistic Regression – output has ordered categories (e.g., rating = low, medium, high).

🔹 3. How It Works

- Instead of fitting a line like in linear regression, logistic regression fits a sigmoid curve.
- The equation:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots)}}$$

where

- p = probability of belonging to class 1
- $\beta_0, \beta_1, \dots$ = coefficients learned from data
- If $p > 0.5$ → predict class 1, else class 0.

🔹 4. Sigmoid Function

- Maps values to the range (0, 1).

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

- Useful to model probabilities.

🔹 5. Assumptions

1. The dependent variable is binary (for simple logistic regression).
2. Independent variables are linearly related to the log-odds.
3. Observations are independent.
4. No multicollinearity among independent variables.
5. Large sample size preferred.

🔹 6. Evaluation Metrics

Since logistic regression is classification, we evaluate using:

- Confusion Matrix
- Accuracy, Precision, Recall, F1-score
- ROC Curve & AUC
- Log-Loss

Logistic Regression

Pros

1. Simple and efficient

Easy to implement and computationally efficient for binary classification tasks.

2. Interpretable results

Provides clear insights into the relationship between features and class probabilities.

3. Probability prediction

Predicts the likelihood of an instance belonging to a specific class.

Cons

1. Linear assumption

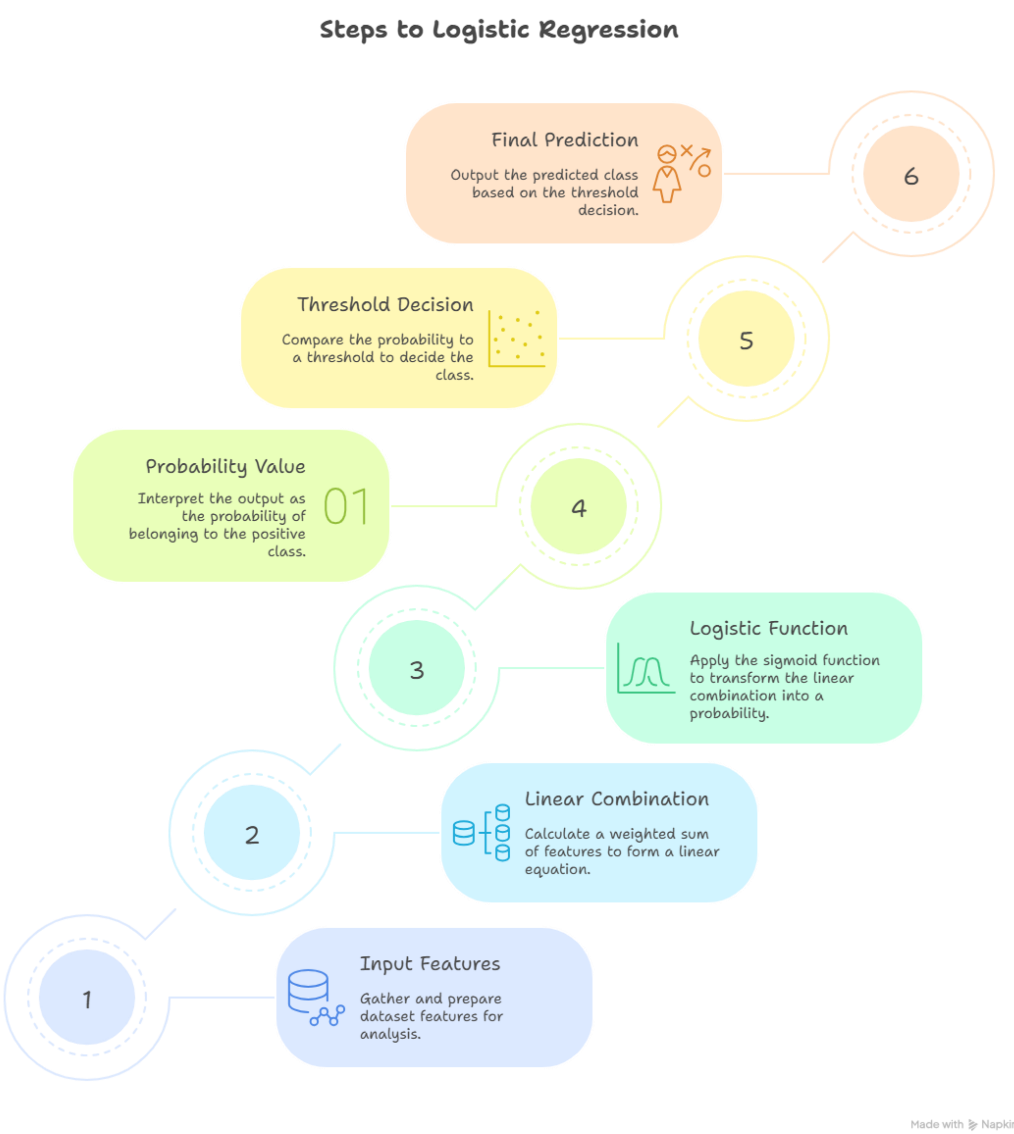
Assumes a linear relationship between features and class probabilities.

2. Sensitive to outliers

Outliers can significantly affect the model's performance.

3. Not suitable for complex data

Struggles with datasets that have complex, non-linear relationships.


Steps to Logistic Regression
Final Prediction
Output the predicted class based on the threshold decision.
6
Threshold Decision
Compare the probability to a threshold to decide the class.
5
Probability Value
Interpret the output as the probability of belonging to the positive class.
01
4
3
Logistic Function
Apply the sigmoid function to transform the linear combination into a probability.
2
Linear Combination
Calculate a weighted sum of features to form a linear equation.
1
Input Features
Gather and prepare dataset features for analysis.
Made with Napkin

Explore LogisticRegression() function

```
LogisticRegression(
    penalty='l2',
    dual=False,
    tol=1e-4,
    C=1.0,
    fit_intercept=True,
    intercept_scaling=1,
    class_weight=None,
    random_state=None,
    solver='lbfgs',
    max_iter=100,
    multi_class='auto',
    verbose=0,
    warm_start=False,
    n_jobs=None,
    l1_ratio=None
)
```

🔹 Key Parameters

1. penalty

- Type of regularization applied.
- Options:
  - `'l1'` → Lasso (feature selection, sparse solution).
  - `'l2'` → Ridge (default, prevents overfitting).
  - `'elasticnet'` → Combination of L1 + L2.
  - `'none'` → No regularization.

---

2. dual

- `False` (default) → primal optimization.
- `True` → dual optimization (only for `'l2'` with `liblinear` solver).
- Rarely changed.

3. tol

- Tolerance for stopping criteria (default `1e-4`).
- Smaller → more precise but slower training.

---

4. C

- Inverse of regularization strength.
  - Smaller `C` → stronger regularization (simpler model).
  - Larger `C` → weaker regularization (fits more to training data).

5. fit_intercept

- `True` → add an intercept term (bias).
- `False` → no intercept.
- Usually keep `True`.

---

6. class_weight

- Adjusts for imbalanced data.
- Options:
  - `None` (default, no weighting).
  - `'balanced'` → automatically balances by class frequency.
  - Or custom dict: `{0: 1, 1: 5}` → weights class 1 more.

7. random_state

- Controls randomness (e.g., in solver).
- Fix a number for reproducibility.

---

8. solver

- Optimization algorithm.
- Options:
  - `'lbfgs'` → (default) good for multiclass & large datasets.
  - `'liblinear'` → small datasets, supports L1.
  - `'saga'` → supports L1, L2, elasticnet (large datasets).
  - `'newton-cg'`, `'sag'` → advanced options.

---

9. max_iter

- Maximum number of iterations (default = 100).
- Increase if model doesn't converge (`ConvergenceWarning`).

---

10. multi_class

- Strategy for multiclass problems.
- Options:
  - `'auto'` → automatically chooses best (default).
  - `'ovr'` (One-vs-Rest).
  - `'multinomial'` (softmax, better for multiclass).

11. n_jobs

- Number of CPU cores used.
- `-1` → use all cores (parallelism).

---

12. l1_ratio

- Used when `penalty='elasticnet'`.
- Mix ratio between L1 and L2:
  - `0` → pure L2.
  - `1` → pure L1.
  - Between → mix.

Parameter	Meaning	When to Use
penalty	Type of regularization (l1, l2, elasticnet, none)	Use l2 (default) for general; l1 for feature selection; elasticnet for mixed.
dual	Solve dual optimization problem (only for liblinear)	Rare; use only if dataset has more features than samples.
tol	Tolerance for stopping criterion	Decrease if convergence issues; smaller = more accurate but slower.
C	Inverse of regularization strength (default=1.0)	Lower C → stronger regularization; higher C → less regularization.
fit_intercept	Add intercept (bias) term	Keep True unless data is already centered.
intercept_scaling	Scaling factor for intercept (used with liblinear)	Rarely needed; ignore in most cases.
class_weight	Handle imbalanced data	Use 'balanced' for imbalanced datasets, or custom {class: weight}.

random_state	Seed for reproducibility	Set a fixed number (e.g., 42) for consistent results.
solver	Optimization algorithm (lbfgs, liblinear, saga, etc.)	lbfgs (default, good for multinomial); liblinear (small data, L1); saga (large data, L1/L2/elasticnet).
max_iter	Max number of iterations	Increase (e.g., 500) if you get ConvergenceWarning.
multi_class	Multi-class strategy (auto, ovr, multinomial)	Use multinomial for softmax-style multiclass; ovr if dataset is small.
n_jobs	Number of CPU cores	Use -1 to run in parallel on all cores.
l1_ratio	Ratio of L1 vs L2 (only if penalty='elasticnet')	Between 0–1; 0 = pure L2, 1 = pure L1.

Logistic Regression Parameter Selection Guide

1️⃣ Step 1 – Check Data Size

- Small dataset (< few 1000 samples):
  - Use solver='liblinear' (fast, supports L1/L2).
- Medium to large dataset (10k+ samples):
  - Use solver='lbfgs' or solver='saga'.

2️⃣ Step 2 – Regularization (penalty)

- L2 (default):
  - Best for most cases (prevents overfitting, stable).
- L1 (feature selection):
  - Forces some coefficients to zero → picks important features.
  - Must use solver='liblinear' or solver='saga'.
- ElasticNet (mix of L1 & L2):
  - Use if you want both sparsity and stability.
  - Requires solver='saga' + l1_ratio.

---

3 Step 3 – Multi-Class Problem?

- Binary classification: Default works (ovr = one-vs-rest).
- Multi-class (>2 classes):
  - Use multi_class='multinomial', solver='lbfgs' (preferred).
  - Or saga if dataset is very large/sparse.

---

4 Step 4 – Imbalanced Data?

- If class distribution is skewed:
  - Add class_weight='balanced'.

📝 Flowchart (Text Version)

```pgsql
Dataset small (< few 1000)?
 ├─ Yes → Use solver='liblinear'
 │      ├─ Want feature selection? → penalty='l1'
 │      └─ Otherwise → penalty='l2'
 └─ No (large dataset) → Use solver='lbfgs' (or 'saga' if sparse)
        ├─ Multiclass? → multi_class='multinomial'
        ├─ Need feature selection? → solver='saga', penalty='l1'
        ├─ Want ElasticNet? → solver='saga', penalty='elasticnet', set l1_ratio
        └─ Otherwise → penalty='l2'
```