Classification

Classification is a branch of supervised machine learning where the output variable belongs to a **category** rather than a continuous number. The goal is to teach a model to learn from past labeled examples and then decide which class a new observation falls into.

Difference from Regression:
Regression predicts values on a numerical scale, such as predicting the future sales of a product in dollars. Classification, in contrast, predicts labels such as “fraud” or “not fraud.”

Example of Classification: Determining whether a bank transaction is fraudulent.
Example of Regression: Predicting the expected rainfall (in millimeters) for a given week.

2A. Classification Algorithms

Logistic Regression

How it works: Logistic regression uses a mathematical function (sigmoid) to transform raw input values into probabilities between 0 and 1. If the probability is above a certain threshold (e.g., 0.5), the instance is assigned to one class; otherwise, it belongs to the other.
Use case: Predicting whether a voter will participate in an election (Yes/No).
Advantages: Simple to implement, interpretable.
Limitations: Works best for problems with a linear relationship; struggles with very complex data patterns.

Decision Trees

How it works: Decision trees split data into branches based on features, similar to a flowchart of yes/no questions. Each path leads to a final classification outcome.
Use case: Diagnosing whether a plant disease is present based on leaf color, size, and temperature.
Advantages: Easy to visualize and interpret.
Limitations: Prone to overfitting; small changes in data can create a different tree.

Random Forest

How it works: Random Forest is an ensemble of many decision trees. Each tree makes a prediction, and the final output is determined by the majority vote.
Use case: Predicting if a mobile app user will uninstall the app after installation.
Advantages: More accurate than a single tree, less prone to overfitting.
Limitations: More computationally demanding; less interpretable compared to a single tree.

2B. Extended Task – K-Nearest Neighbors (KNN)

Problem suitability: KNN is useful for classification tasks where similar objects tend to belong to the same group.

How it works: KNN stores all training data. When a new observation appears, the algorithm finds the "K" closest data points (neighbors) and assigns the most common class among them.

Application: Handwriting recognition — deciding which digit (0–9) a handwritten number represents.

Strengths: Simple, effective for small datasets, no training phase required.

Weaknesses: Computationally expensive for large datasets, sensitive to irrelevant features and scaling.

Compared to Logistic Regression, Decision Trees, and Random Forest:

KNN does not create an explicit model — it relies entirely on stored data.

It can adapt easily but is slower on large datasets.

3. Classification Metrics

Accuracy

The proportion of correct predictions out of all predictions made. Good for balanced datasets.

Precision

Of all predicted positives, how many are truly positive. Useful when false alarms must be minimized (e.g., detecting malware).

Recall

Of all actual positives, how many were correctly identified. Important in contexts where missing a positive is costly (e.g., diagnosing a disease).

F1-Score

The harmonic mean of precision and recall, providing a balance between the two.

Confusion Matrix

A 2×2 or larger table that summarizes predictions into true positives, true negatives, false positives, and false negatives.

Comparison Table

Metric	Focus	When to Use	Weakness
Accuracy	Overall correctness	Balanced datasets	Misleading on imbalance
Precision	Avoiding false positives	Security alerts, spam filters	Ignores false negatives
Recall	Avoiding false negatives	Medical tests, fraud detection	Ignores false positives
F1-Score	Balancing precision & recall	Imbalanced datasets	Harder to interpret alone
Confusion Matrix	Detailed error breakdown	Any classification task	Not a single score

4.Imbalanced Data Problem

Imbalanced data occurs when one class has far more examples than the other. For instance, in fraud detection, 99% of transactions may be normal while only 1% are fraudulent.

Why Accuracy Can Be Misleading: A model that predicts “not fraud” for every case will be 99% accurate but useless at detecting fraud.

More Reliable Metrics: Precision, Recall, and F1-Score are better suited because they evaluate performance specifically on minority classes that matter most.

5. Real-World Case Study: Credit Card Fraud Detection

Goal: Identify fraudulent credit card transactions in real time.
Data Used: Millions of anonymized transactions with features such as transaction amount, location, and time.
Model Applied: A Random Forest classifier trained with resampling techniques to handle class imbalance.
Key Results: The model achieved high recall (capturing most fraudulent cases) while keeping precision reasonable, which helped reduce financial losses without overwhelming investigators with false alarms.