Text Classification and Responsible Classification

Applied Text Mining

Dr. Maryam Movahedifar

14-17 July 2025

University of Bremen, Germany
movahedm@uni-bremen.de

Outline

Introduction to Classification

Types of Learning: Supervised vs. Unsupervised

- Supervised Learning:
  - The model is trained on labeled data, meaning each input has a corresponding output (label).
  - Example: Classifying emails as "Spam" or "Not Spam."
- Unsupervised Learning:
  - The model is trained on unlabeled data, meaning no predefined outputs are provided.
  - Example: Clustering news articles based on topics without knowing the categories.

Supervised Learning

- **Definition:** Learning from labeled data where each input has a known output (label).
- **Label:** The output or correct answer the model tries to predict.
- **Examples:**
  - **Text Classification:**
    - Input: *"This book is amazing!"*
    - Label: *Positive sentiment*
  - **Spam Detection:**
    - Input: *"You have won a prize!"*
    - Label: *Spam*
  - **Document Classification:**
    - Input: A news article about politics
    - Label: *Politics*

Unsupervised Learning

- **Definition:** Learning from unlabeled data, where the model discovers patterns without predefined outputs.
- **No Labels:** There are no predefined categories or answers provided.
- **Examples:**
  - **Clustering:**
    - Input: A collection of customer reviews
    - Output: Grouping them into clusters like *"Positive"* or *"Negative"* (discovered automatically).
  - **Topic Modeling:**
    - Input: A set of news articles
    - Output: Discovering topics such as *"Sports"*, *"Politics"*, etc., without prior labels.

Key Concepts in Supervised Learning

Features

Input variables used to describe and distinguish the data. In text mining, these can be word counts or TF-IDF values.

Parameters

Internal values learned during training, like weights in a linear model or split points in a tree.

Prediction

The model's output based on input features. Compared with true labels to evaluate performance.

Hyperparameters

User-defined settings such as learning rate or number of neighbors. Chosen before training.

Types of Classification

★ **Classification** is a core task in supervised learning. The goal is to assign data points to predefined categories. There are two primary types:

Binary Classification

Involves exactly two classes.

Examples:

- Spam vs. Ham
- Disease vs. No Disease
- Yes vs. No

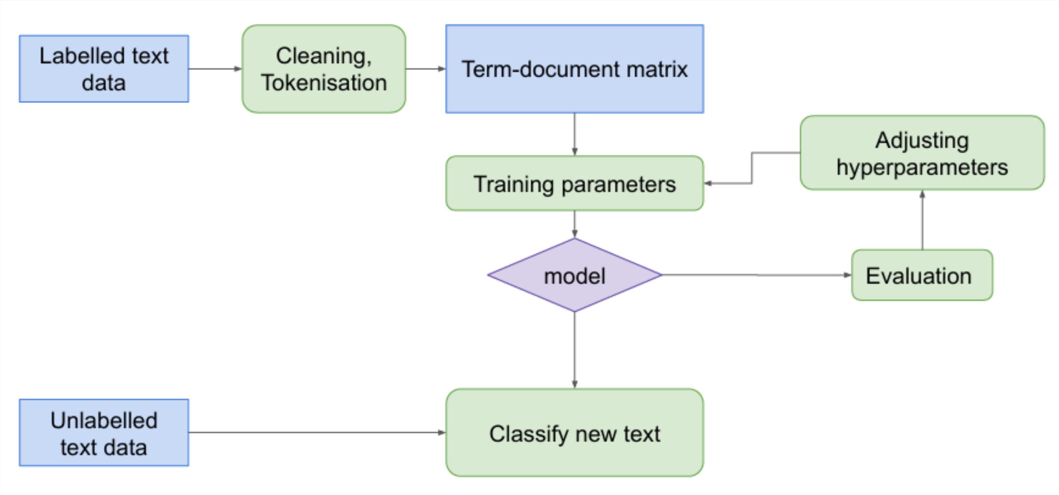
Multiclass Classification

Involves more than two classes.

Examples:

- News: Sports, Politics, Tech
- Product types: Phone, Laptop, Tablet

Classification Workflow



Algorithms for Classification

Classification Methods

Popular Classification Algorithms:

Logistic Regression

Support Vector Machine (SVM)

K-Nearest Neighbors (KNN)

Naive Bayes

Decision Tree

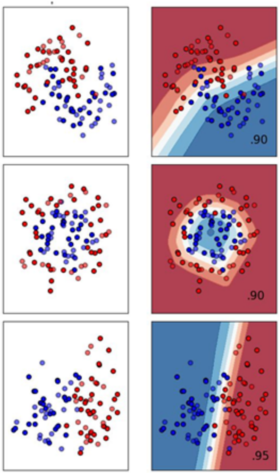
Ensemble Classifiers

We will dive deeper into each method in the upcoming slides.

Logistic Regression

Logistic Regression is used for binary classification tasks, predicting the probability of a given input belonging to one of two classes.

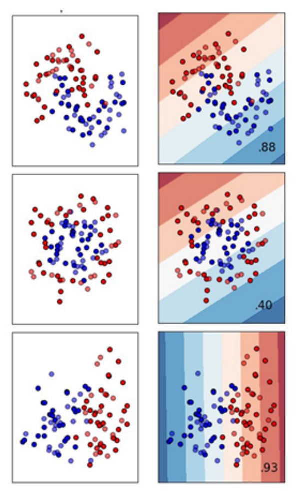
- **Assigns weights to input features** to calculate probabilities for classification.
- **Output is normalized** to a probability distribution (values between 0 and 1).
- Pros: Simple and fast to train and provides probabilistic outputs
- Cons: Assumes linear decision boundary and not suitable for non-linear data



Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm that is used for classification tasks. It works by finding a hyperplane that best separates data points of different classes.

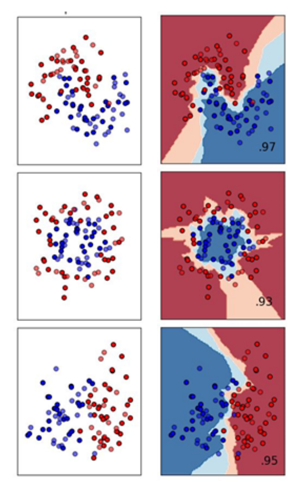
- **Finds a hyperplane** that separates the data into classes with maximum margin.
- Pros: Less sensitive to noisy data, making it effective for high-dimensional spaces.
- Cons: Requires linear separation, and performance may degrade if data is not linearly separable.



K-Nearest Neighbour (KNN)

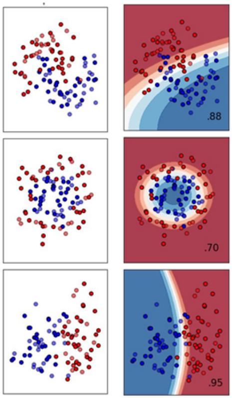
K-Nearest Neighbour (KNN) is a simple, powerful classification algorithm that predicts the label of a text based on its "nearest" neighbors in the training data.

- **Classifies based on proximity** to the nearest neighbors in the training dataset.
- **Proximity measured** using the term-document matrix.
- **Takes the average** of the k nearest neighbors for classification.
- Pros: No assumptions about linearity or independence of features.
- Cons: For large datasets, quick to train but slow to classify.



Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' Theorem that assumes independence between features (words). It is simple and efficient for text classification.



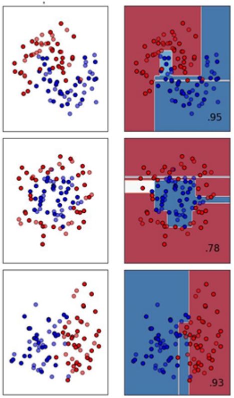
- **Assumes independence:** Features (words) are assumed to be independent.
- **Estimates probabilities:** Calculates probability distributions for each word given the label.
- **Classify based on likelihood:** Uses probabilities to predict the most likely label for text.

- Pros: Efficient for large datasets and works well with imperfect assumptions.
- Cons: Assumes feature independence and may struggle with correlated features.

Decision Tree

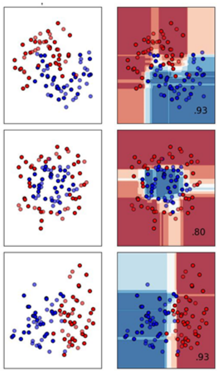
Decision trees are a type of model that splits data based on features, making decisions by choosing the most informative feature at each branch to classify data.

- **Generate a decision tree:** Choose the most informative feature at each branch to separate data.
- **Hyperparameter:** Maximum depth of the tree; deeper trees capture more detail.



- Pros: Can capture complex relationships and is interpretable.
- Cons: Prone to overfitting, especially with deep trees, and sensitive to noisy data.

Ensemble Classifiers



Ensemble classifiers combine the predictions of multiple models to improve overall performance and reduce the volatility of single classifiers.

- **Random Forest Classifier:** Combines multiple decision trees and averages predictions to improve stability.
- **Voting Classifier:** Uses multiple classifiers to "vote" on the result, with potential for classifiers of different types.

- Pros: Increased accuracy, reduces overfitting compared to a single model.
- Cons: Computationally expensive, requires more resources and time.

Evaluation Metrics

Accuracy, Precision, and Recall

Key Metrics for Evaluating Classifiers:

- **Accuracy** — Measures overall correctness: *How many predictions were right?*
- **Precision** — Focuses on predicted positives: *Of what we predicted as positive, how many were truly positive?*
- **Recall** — Focuses on actual positives: *Of all actual positive cases, how many did we correctly identify?*

F1-score: A balance between precision and recall:

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Multiclass Metrics

In multiclass tasks, we need overall scores to evaluate performance across all classes.

Macro F1

- Compute F1 for each class separately
- Take the average of all F1-scores
- All classes are treated equally
- Good for balanced datasets

Micro F1

- Count all true/false positives and negatives globally
- Then compute F1 from total counts
- Weighs larger classes more
- Better for imbalanced datasets

Tip: Use *Macro F1* when all classes are important. Use *Micro F1* when performance on frequent classes matters.

What Does Low Performance Mean?

Common Causes of Low Model Performance:

- △ **Noisy Data or Missing Features**
  Irrelevant, inconsistent, or incomplete data disrupts learning.

- △ **Insufficient Training Data**
  Too little data limits the model's ability to capture patterns.

- △ **Underfitting or Overfitting**
  Model complexity issues cause poor generalization on new data.

Responsible Classification

Responsible Classification

Ensuring machine learning models are fair, transparent, and ethical is crucial for trustworthy AI.

- Fairness: Prevent biased decisions and promote equality.
- Transparency: Make model decisions explainable and understandable.
- Ethical Use: Avoid harm and ensure responsible application.

Building responsible classifiers helps create AI systems people can trust and rely on.

Impact of Noisy Data and Missing Features

Garbage In, Garbage Out

A model is only as good as the data it's trained on.

- ✗ **Mislabeled Training Data:** Incorrect labels mislead the model and hurt performance.
- ✗ **Corrupted or Irrelevant Features:** Features with errors or inconsistencies reduce learning quality.
- ✗ **Missing Key Information:** Important features might not exist in the dataset at all.

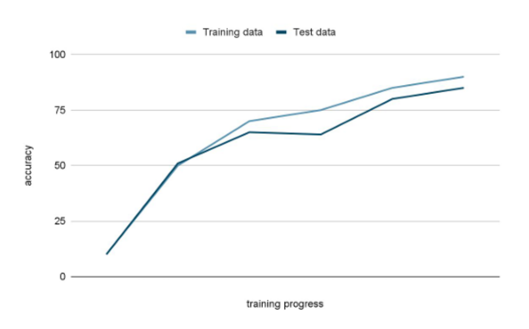
→ **Solution:** Use data cleaning, handle missing values, and validate labels to improve model reliability.

When There's Not Enough Data

"More data = Better learning."
Without enough examples, even good models can't learn effectively.

- ✗ Model has limited exposure to patterns.
- ✗ High chance of overfitting or unstable predictions.
- ✔ Adding data improves generalization and reliability.

Tip: Try data augmentation or transfer learning when data is scarce.



Is Your Model Too Simple? (Underfitting)

Underfitting occurs when the model is too simple to capture patterns in the data.

- ✗ Fails to detect important trends.
- ✗ Poor training and testing performance.
- ✗ Adding data doesn't help.
- ✔ Use a more complex model or richer features.

Think of fitting a straight line to a curve! it misses the shape completely.

Overfitting: When Models Memorize Too Much

Overfitting happens when a model learns the training data *too well*, including noise and irrelevant details.

Symptoms:

- ✔ Very low training error
- ✗ High test error

Why it happens:

- Model too complex
- Too little training data

How to fix it:

- Simplify the model
- Add regularization
- Use more data

Train–Validation–Test: Why the Split Matters

Building reliable models requires keeping training honest. Splitting the dataset prevents overfitting and gives a realistic performance check.

Train Set — *"Learn"*

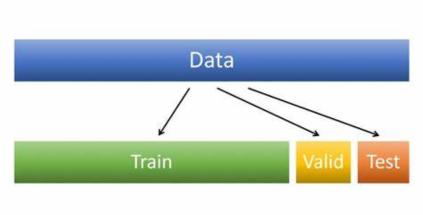
- Used to fit model parameters.
- The model learns patterns here.

Validation Set — *"Tune"*

- Used to adjust hyperparameters.
- Helps detect overfitting.

Test Set — *"Judge"*

- Evaluates final model performance.
- Never seen by the model during training.



Rule of Thumb: Train (60–80%), Validation (10–20%), Test (10–20%)

Conclusion

Summary: Key Takeaways

- **Text classification** uses supervised learning to automatically assign meaningful labels.
- **Model evaluation** goes beyond accuracy — precision, recall, and F1 reveal deeper insights into performance.
- **Balance matters!** Avoid overfitting (memorizing noise) and underfitting (missing patterns) for reliable results.
- **Data and tuning** are your best friends — good splits and hyperparameter choices make all the difference.

Practical 2