# Module 3 Comprehensive Guide

## Other Supervised Learning Models

## 📌 Introduction to Supervised Learning

Supervised learning involves training models on labeled data to predict outcomes for new, unseen data. The key objective is to minimize errors while ensuring generalization to new datasets, develop models that can generalize well to unseen data.

In Module 3, we focus on:

* **K-Nearest Neighbors (KNN)** – An instance-based algorithm that classifies data based on similarity.
* **Support Vector Machines (SVMs)** – A powerful classification algorithm that finds the optimal decision boundary.
* **Bias-Variance Tradeoff** – Understanding and balancing model complexity.
* **Ensemble Learning (Bagging & Boosting)** – Improving model performance by combining multiple classifiers.

## 📌 K-Nearest Neighbors (KNN)

### 🔹 What is K-Nearest Neighbors (KNN)?

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for both classification and regression. Unlike other machine learning algorithms that explicitly learn a model during training, KNN memorizes the dataset and makes predictions based on similarity.

KNN assumes that data points close to each other share similar properties and that classification can be determined based on proximity to labeled examples.

✔ It is a **lazy learning algorithm**, meaning no actual training phase occurs—instead, computations happen at the time of prediction.

✔ It is **simple and effective**, working well with small datasets and low-dimensional feature spaces.

✔ It **does not assume any distribution** about the data, making it useful for non-linear decision boundaries.

### 🔹 How KNN Works

The core principle of KNN is that it assigns a label (classification) or predicts a value (regression) for a new data point based on its K closest neighbors in the training set.

The process involves four key steps:

1. Choose a value for K (the number of nearest neighbors to consider).
2. Calculate the distance between the new data point and all training data points.
3. Find the K-nearest neighbors (smallest distances).
4. Make a prediction:
   * For classification → Assign the most frequent class among the K neighbors (majority vote).
   * For regression → Take the average (or weighted average) of the target values of the K neighbors.

✔ The performance of KNN depends on choosing an optimal K value and an appropriate distance metric.

### 🔹 Choosing the Right K Value

Selecting the optimal K value is a crucial step in achieving a well-performing KNN model. The choice of K impacts the bias-variance tradeoff:

* Small K (e.g., K=1, K=3) → Leads to Overfitting

✔The model becomes too sensitive to noise and classifies based on just one or two points.

✔High variance and low bias (too flexible).

* Large K (e.g., K=20, K=50) → Leads to Underfitting

✔The model becomes too general and fails to capture local structures in data.

✔Low variance and high bias (too rigid).

* Optimal K → Usually found via cross-validation.

### 🔹 Finding the Optimal K Using Accuracy Analysis

We determined the best value of K by training multiple KNN models with different values of K and evaluating their accuracy on a test set.

The process followed these steps:

1. **Define a range of K values to test**
   * We started with a range of small to large values of K (e.g., from 1 to 50).
   * The goal was to observe how accuracy changes with increasing K.
2. **Train multiple KNN models**
   * A separate KNN model was trained for each selected K value.
   * Each model made predictions on the test set and the accuracy was recorded.
3. **Plot accuracy vs. K values**
   * The accuracy scores were plotted against the corresponding K values.
   * This helped visualize how accuracy changes with different K values.
4. **Identify the best K**
   * The optimal K was chosen as the one that maximized accuracy while maintaining stability.
   * If accuracy fluctuated significantly, we selected a slightly larger K for smoother performance.

**Interpreting the Results:**

* If the accuracy is highest at very low K values (K=1, K=3), the model is likely overfitting to the training data.
* If the accuracy remains stable for mid-range K values (K=5 to K=15), these values are usually good choices.
* If the accuracy declines for large K values (K > 30), the model is underfitting, failing to capture important decision boundaries.

✔ The best practice is to select the K value that provides the highest accuracy while preventing excessive fluctuations due to noise.

### 🔹 Distance Metrics in KNN

Since KNN relies on measuring distances, selecting the right distance metric is critical for performance.

1. **Euclidean Distance (Most Common)**

The default metric, representing the shortest straight-line distance between two points:

✔ Works well with continuous numerical data.

✔ Assumes all features are equally important.

1. **Manhattan Distance (City Block Distance)**

The sum of absolute differences between features values

✔ More robust for high-dimensional data.

✔ Works well when features have different units of measurement.

1. **Minkowski Distance (Generalized Distance Metric)**

A generalization of Euclidean and Manhattan distances:

* + When **p = 1**, Minkowski reduces to **Manhattan Distance**.
  + When **p = 2**, Minkowski reduces to **Euclidean Distance**.

✔ Allows flexibility in distance calculations.

### 🔹 Handling Imbalanced Classes in KNN

KNN can be biased towards the majority class if the dataset is imbalanced.

To address this, we can:

✔ **Use Weighted KNN** → Assign higher weights to closer neighbors, reducing the influence of distant points.

✔ **Use Data Resampling** → Oversample the minority class or undersample the majority class.

Weighted KNN formula:

where:

* + is the inverse distance weight .

✔ This approach ensures that closer neighbors contribute more to predictions.

### 🔹 Handling Feature Scaling in KNN

Since distance-based algorithms are sensitive to feature magnitudes, feature scaling is essential.

✔ Min-Max Scaling → Scales data to the [0,1] range.

✔ Standardization (Z-score Normalization) → Centers data with mean μ=0 and standard deviation 𝜎 = 1.

✔ Feature scaling ensures that all features contribute equally to distance calculations.

### 🔹 KNN vs. Other Classification Algorithms

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Strengths** | **Weaknesses** |
| **KNN** | ✔Simple, non-parametric, works well with small data. | ✖Computationally expensive for large datasets. |
| **Decision Tree** | ✔Fast inference, interpretable rules | ✖Prone to overfitting without pruning. |
| **SVM** | ✔Effective for high-dimensional spaces. | ✖Sensitive to noise, requires careful tuning. |
| **Logistic Reg.** | ✔Works well for linearly separable data. | ✖Struggles with complex decision boundaries. |

✔ Use KNN when the dataset is small and interpretability is needed.

✔ Avoid KNN for large datasets due to slow prediction speed.

## 📌 Support Vector Machines (SVM

### 🔹 What is Support Vector Machines (SVM)?

Support Vector Machines (SVMs) are supervised learning algorithms used for both classification and regression, with a primary focus on finding the optimal decision boundary that best separates different classes in a dataset.

✔ SVM is powerful for **high-dimensional** spaces where clear class separation is required.

✔ It **works well for both linearly and non-linearly separable data** by mapping data into higher dimensions using kernel functions.

✔ It aims to maximize the margin between data points belonging to different classes.

Unlike traditional classifiers like KNN or Logistic Regression, SVM does not rely on probability scores but instead finds the most optimal hyperplane that distinctly classifies data points.

### 🔹 How SVM works

SVM classifies data by finding the best possible hyperplane that separates different classes while maximizing the margin (distance between the closest data points of different classes, known as support vectors).

Key Concept: Support Vectors

✔ Support Vectors are the closest data points to the hyperplane.

✔ These points define the margin and influence the decision boundary.

✔ The model does not consider other data points when forming the boundary, making SVM highly robust to noise.

1. Identifying the Optimal Hyperplane

* In two-dimensional space, the hyperplane is simply a line that separates data points.
* In three-dimensional space, the hyperplane is a plane that divides the dataset.
* In higher-dimensional spaces, the hyperplane is a mathematical decision boundary.

1. Hard Margin vs Soft Margin

✔ Hard Margin SVM (Strict separation)

* + The model assumes that all data points can be perfectly separated by a linear boundary.
  + Works well for **clean**, noise-free datasets.
  + **Limitations** → If classes overlap even slightly, the model fails completely

✔ Soft Margin SVM (Tolerates misclassifications)

* + Introduces a flexibility parameter (C) that allows some misclassification to balance generalization.
  + The lower the C parameter, the more misclassification is allowed (higher bias, lower variance).
  + The higher the C, the stricter the boundary (lower bias, higher variance).

✔ In real-world applications, Soft Margin SVM is preferred as real-world data is rarely perfectly separable.