K-Nearest Neighbors (KNN):

1. Model Overview

K-Nearest Neighbors (KNN) is a **non-parametric, instance-based (lazy learning)** algorithm used for both **classification** and **regression** tasks. It predicts outcomes based on the majority class (for classification) or average of nearest neighbors (for regression).

---

2. Key Aspects to Analyze for KNN

A. Assumptions

1. Similar data points exist close to each other in the feature space (locality assumption).
2. The dataset is **representative and balanced**, with all relevant features captured.
3. Distance metrics used (e.g., Euclidean, Manhattan) meaningfully represent similarity.
4. Features are **scaled properly** (normalization or standardization is required).
5. The data is **low-dimensional** — KNN suffers from the "curse of dimensionality."

B. Limitations

1. **Computationally expensive** for large datasets — requires storing all data points.
2. **Slow prediction time** — since distances are calculated for each query instance.
3. **Sensitive to noisy or irrelevant features**.
4. **Performance degrades** with high-dimensional data (many features).
5. **Choice of 'K' and distance metric** significantly impacts accuracy.
6. **Imbalanced classes** can bias the prediction toward the majority class.

C. Attributes / Input Features

1. Works best with **continuous and numerical** features.
2. Categorical features can be handled using **appropriate encoding** and **distance functions** (e.g., Hamming distance).
3. **Feature scaling** (e.g., Min-Max or Standard Scaler) is **mandatory**.
4. Outliers should be minimized since KNN uses distance-based measures.

D. Internal Model Variations / Subtypes

1. **Weighted KNN** — assigns higher weights to closer neighbors.
2. **Distance Metrics Variants**:
   - Euclidean Distance
   - Manhattan Distance
   - Minkowski Distance
   - Cosine Similarity
3. **Approximate Nearest Neighbor (ANN)** — uses indexing structures for faster searches.
4. **KNN Regression** — predicts continuous values using the mean (or weighted mean) of neighbors.

E. Hyperparameters

1. **K (Number of Neighbors):**
   - Low K → High variance (overfitting)
   - High K → High bias (underfitting)
   - Optimal K often found using **cross-validation** or **Elbow method**

2. **Distance Metric:** Defines how closeness is measured.
3. **Weight Function:** Uniform vs distance-based weighting.
4. **Algorithm:** 'auto', 'ball_tree', 'kd_tree', or 'brute' (in scikit-learn).

F. Performance Evaluation Metrics

For **Classification**:

- Accuracy, Precision, Recall, F1-score, ROC-AUC

For **Regression**:

- Mean Squared Error (MSE), Mean Absolute Error (MAE), R^2 Score

---

G. Use Cases

- Recommendation systems
- Pattern recognition (e.g., handwriting, facial recognition)
- Medical diagnosis (disease classification)
- Credit risk and fraud detection

H. Optimization Tips

1. **Feature scaling** before model training.
2. **Dimensionality reduction (PCA)** for high-dimensional data.
3. **Remove outliers** and irrelevant features.
4. **Tune K** using validation curves or grid search.
5. Use **KDTree or BallTree** for large datasets to improve performance.

I. Model Interpretation

- **Transparent and explainable** — predictions are based on visible neighbors.
- No internal coefficients or weights like linear models — relies purely on distance logic.

J. Summary Table

Aspect	Description
Type	Non-parametric, instance-based
Learning Type	Lazy (no training phase)
Key Parameter	K (number of neighbors)
Requires Scaling	Yes
Handles Outliers Well	No
Sensitive to Dimensionality	Yes
Ideal Dataset Size	Small to medium
Output Type	Classification or Regression