UNIT-II:

Nearest Neighbor-Based Models:Introduction to Proximity Measures, Distance Measures, Non-Metric Similarity Functions, Proximity Between Binary Patterns, Different Classification Algorithms Based on the Distance Measures ,K-Nearest Neighbor Classifier, Radius Distance Nearest Neighbor Algorithm, KNN Regression, Performance of Classifiers, Performance of Regression Algorithms.

Nearest Neighbor-Based Models: Introduction to Proximity Measures:

Introduction to Proximity Measures

Proximity measures quantify how similar or dissimilar two data points are. The choice of proximity measure directly impacts the performance of nearest neighbor algorithms, such as k-Nearest Neighbors (k-NN). These measures can be broadly classified into **distance-based** and **similarity-based** metrics.

1. Distance-Based Proximity Measures

Distance metrics evaluate how far apart two points are in a feature space. Common distance measures include:

Euclidean Distance

- Formula: $d(x,y)=SQRT\sum_{i=1}^{n}(xi-yi)^2$

Suitable for continuous variables and when all features are on the same scale.

Manhattan Distance (L1 Norm)

- Formula: $d(x,y)=\sum_{i=1}^{n}|xi-yi|$

Measures the distance between points along axes at right angles (useful for grid-like path problems).

Minkowski Distance

Generalization of Euclidean and Manhattan distances.

Formula: $d(x,y)=(\sum_{i=1}^{n}|xi-yi|^p)^{1/p}$

Parameter ppp determines the type of distance (e.g., p=1 for Manhattan, p=2 for Euclidean).

Chebyshev Distance

Formula: $d(x,y)=\max(|xi-yi|)$

Suitable when the maximum difference in any dimension is crucial.

Hamming Distance

Measures the number of positions at which the corresponding elements differ.

Used for categorical variables and binary strings.

$$d(x,y)= \sum_{i=1}^{n} II(xi!=yi)$$

- where II is the indicator function.

Use Cases:

- For categorical variables or binary strings.
- Common in text mining and bioinformatics (e.g., DNA sequence analysis).

2. Similarity-Based Proximity Measures

These measures determine how closely related two data points are, rather than how far apart they are:

Cosine Similarity

Formula: $sim(x,y)=x \cdot y/\|x\|\|y\|$

Measures the cosine of the angle between two vectors. Useful in text analysis and high-dimensional spaces.

Jaccard Similarity

Formula: $sim(x,y)=|x \cap y|/|x \cup y|$

Ideal for binary and set-based data, especially in recommendation systems.

Pearson Correlation Coefficient

Measures linear correlation between two variables.

Ranges from -1 to 1, where 1 indicates perfect positive correlation.

3. Applications of Nearest Neighbor-Based Models

- **Classification**: k-NN is a simple yet effective classifier where the class of a data point is determined by the majority class among its nearest neighbors.
- **Regression**: In k-NN regression, the value of a point is predicted based on the average of its neighbors' values.
- **Recommendation Systems**: Collaborative filtering uses proximity measures to recommend items similar to what a user has liked in the past.
- **Anomaly Detection**: Points that have few or distant neighbors can be flagged as anomalies.

4. Considerations in Choosing Proximity Measures

- **Feature scaling**: Distance measures like Euclidean distance are sensitive to the scale of data. Normalization or standardization may be required.
- **Data type**: The choice of measure depends on whether the data is continuous, categorical, or binary.
- **Dimensionality**: In high-dimensional data, measures like cosine similarity are preferred due to the "curse of dimensionality" affecting distance measures.

Distance Measures:

Nearest neighbor-based models, such as **k-Nearest Neighbors (k-NN)** for classification and regression, rely heavily on distance measures to identify the "closest" data points. The choice of a distance measure can significantly influence the performance and accuracy of these models.

Euclidean Distance

Formula: $d(x,y)=SQRT\sum_{i=1}^{n}(xi-yi)$

Suitable for continuous variables and when all features are on the same scale.

- **Use Cases**:
    - Suitable for continuous variables.
    - Works best when data features are on the same scale.
- **Key Points**:
    - Sensitive to the scale of data.
    - Requires normalization when features have different units.

Manhattan Distance (L1 Norm)

- Formula: $d(x,y)=\sum_{i=1}^{n}|xi-yi|$
- Measures the distance between points along axes at right angles (useful for grid-like path problems).
- **Use Cases**:
    - Suitable for high-dimensional data.
    - Works well when the path along axes matters (e.g., grid-based maps).
- **Key Points**:
    - Less sensitive to outliers than Euclidean distance.

Minkowski Distance

Generalization of Euclidean and Manhattan distances.

Formula: $d(x,y)=(\sum_{i=1}^{n}|xi-yi|^{p})^{1/p}$

Parameter ppp determines the type of distance (e.g., p=1 for Manhattan, p=2 for Euclidean).

Parameters:

- p=1p = 1p=1: Manhattan Distance
- p=2p = 2p=2: Euclidean Distance

Key Points:

- Flexible depending on the value of ppp.
- Higher values of ppp emphasize larger differences.

Chebyshev Distance

Formula: $d(x,y)=\max(|xi-yi|)$

Suitable when the maximum difference in any dimension is crucial.

Use Cases:

- Suitable when the maximum deviation in any feature dimension is crucial.

Key Points:

- Often used in scenarios like chess (e.g., king's movement on a chessboard).

Hamming Distance

Measures the number of positions at which the corresponding elements differ.

Used for categorical variables and binary strings.

$d(x,y)= \sum_{i=1}^{n} II(xi!=yi)$

- where II is the indicator function.
-

Use Cases:

For categorical variables or binary strings.

Common in text mining and bioinformatics (e.g., DNA sequence analysis).

<u>Nearest Neighbor-Based Models: Non-Metric Similarity Functions:</u>

some scenarios, **non-metric similarity functions** are more appropriate, especially when:

The data does not naturally fit into a metric space.

The similarity measure is derived from domain-specific knowledge.

Relationships in the data violate the triangle inequality or symmetry.

Key Characteristics of Non-Metric Similarity Functions:

Non-metric functions may not satisfy one or more metric properties. For example:

Non-Symmetry:

$S(x,y) \neq S(y,x)$

Example: In recommendation systems, the influence of user A on user B's preferences may differ from the influence of user B on user A.

Triangle Inequality Violation:

$S(x,z)$ may not be less than or equal to $S(x,y)+S(y,z)$

Example: In semantic similarity tasks, concept relationships may skip intermediate nodes that would otherwise enforce the triangle inequality.

Domain-Specific Similarity:

Graph-based similarity measures (e.g., SimRank) or kernel functions that don't adhere to metric space rules.

Examples of Non-Metric Similarity Functions:

Jaccard Similarity for Sets (non-metric in certain weighted forms):

$S_{Jaccard}(A,B)=|A \cap B|/|A \cup B|$

-

SimRank (Graph Similarity):

Assumes two objects are similar if they are referenced by similar objects. It's recursive and often non-metric.

Kullback-Leibler (KL) Divergence (Information Theory):

$D_{KL}(P \| Q)=\sum iP(i)\log P(i)/Q(i)$

KL divergence is non-symmetric and fails to meet metric properties.

Rank-Based Measures:

For example, Kendall's tau may not satisfy all metric conditions in some ranking applications.

Applications of Non-Metric Similarity in Nearest Neighbor Models:

Recommender Systems: User-item relationships might be asymmetric, e.g., one user follows another, but not vice versa.

Natural Language Processing (NLP): Semantic similarity between words or phrases where context changes relationships dynamically.

Bioinformatics: Protein structure comparison where biological relevance matters more than geometric distances.

Social Network Analysis: Influence and connectivity patterns that are inherently directional.

Challenges and Solutions:

Computational Complexity:

Without metric properties, standard optimizations like tree-based indexing (e.g., KD-trees) may not work efficiently.

Solution: Approximate nearest neighbor (ANN) methods like *Locality-Sensitive Hashing (LSH)* or *graph-based approaches*.

Scalability:

Large datasets require efficient methods for similarity search.

Solution: Leverage deep learning embeddings combined with non-metric similarity tailored to the application.

Interpretability:

Non-metric similarities may lack intuitive geometric interpretation.

Solution: Visualization techniques and careful feature engineering.

<u>Proximity Between Binary Patterns:</u>

Nearest neighbor-based models determine the "closeness" of data points using similarity or distance measures. When dealing with **binary patterns** (vectors with values of 0 or 1), traditional distance metrics must adapt to the discrete nature of the data. Such patterns arise in various fields, including text mining (e.g., presence or absence of words), bioinformatics (e.g., gene activation), and pattern recognition.

Key Concepts of Proximity for Binary Patterns

1. Binary Pattern Representation

A binary pattern can be represented as:

$X=(x1,x2,\ldots,xn), xi \in \{0,1\}$

where each dimension indicates the presence (1) or absence (0) of a feature.

2. Basic Counts in Binary Comparisons

To compute proximity between two binary patterns X and Y, we define:

- a: Number of positions where X=1 and Y=1 (both have the feature).
- b: Number of positions where X=1 and Y=0.
- ccc: Number of positions where X=0 and Y=1.
- ddd: Number of positions where X=0 and Y=0.

Proximity Measures for Binary Patterns

A. Similarity Measures

- **Jaccard Similarity**
  - Focuses only on the presence of features (ignores d).

  $S_{Jaccard}(X,Y)=a/a+b+c$

  Simple Matching Coefficient (SMC)
  - Considers both matches of 1s and 0s.

  $S_{SMC}(X,Y)=a+d/a+b+c+d$

  Dice Similarity Coefficient
  - Gives more weight to matches of 1s.

  $S_{Dice}(X,Y)=2a/2a+b+c$

  Hamming Similarity
  - Based on the number of identical bits.

  $S_{Hamming}(X,Y)=1-\text{Hamming distance}/n$

B. Distance Measures

- **Hamming Distance**
  - Counts the number of positions where the binary patterns differ.

  $D_{Hamming}(X,Y)=b+c$

  Jaccard Distance

- Complement of Jaccard similarity.

$D_{Jaccard}(X,Y)=1-S_{Jaccard}(X,Y)$

Russell-Rao Distance

- Proportion of mismatches relative to total features.

$D_{Russell-Rao}(X,Y)=1-a/n$

Yule's Q-Statistic

- Measures association based on concordance and discordance.

$Q=ad-bc/ad+bc$

Choosing the Right Measure

Measure	Best For	Key Insight
Jaccard	Sparse data with rare features	Ignores co-absence of features (zeros).
SMC	Balanced feature presence and absence	Considers both matches of 1s and 0s.
Dice	When positive matches are more significant	Prioritizes co-occurrences of 1s.
Hamming Distance	Error correction, bit-flip scenarios	Simple and intuitive for binary codes.
Yule's Q-Statistic	Association in binary classifications	Sensitive to pattern correlation.

Applications of Binary Pattern Proximity

Text Mining:

Document similarity based on word presence/absence (e.g., bag-of-words models).

Genomics:

Comparing genetic sequences where genes are either expressed (1) or not (0).

Image Recognition:

Binary image patterns (e.g., black-and-white pixel comparisons).

Recommender Systems:

User preference patterns (e.g., watched/not watched, liked/not liked).

Fault Detection:

Identifying error patterns in digital communication and storage.

Challenges and Considerations

- **Dimensionality:** High-dimensional binary data can be sparse. Jaccard similarity and Hamming distance are often preferred for sparse data.
- **Feature Importance:** All features are treated equally in basic measures. In practice, some features might need weighting.
- **Scalability:** For large datasets, approximate nearest neighbor search (like *LSH*) can improve performance.

Different Classification Algorithms Based on the Distance Measures:

Distance-based classification algorithms determine the class of a given data point by evaluating how "close" it is to other points in the feature space. The choice of **distance measure** is crucial, as it directly affects model performance, especially in high-dimensional or domain-specific datasets.

1. k-Nearest Neighbors (k-NN)

Overview:

- **k-NN** classifies a data point based on the majority label among its **k** nearest neighbors.
- It's a **non-parametric** and **lazy learning** algorithm (no training phase).

Common Distance Measures in k-NN:

- **Euclidean Distance:** $d(x,y)= d(x,y)=SQRT\sum_{i=1}^{n}(xi-yi)^2$
- **Manhattan Distance:** $d(x,y)=\sum_{i}|xi-yi|$
- **Minkowski Distance:** Generalization of Euclidean and Manhattan.
- **Hamming Distance:** For categorical or binary data.
- **Cosine Distance:** For text data and high-dimensional sparse data.
- **Jaccard Distance:** For binary data (e.g., document similarity).

Use Cases:

- Handwritten digit recognition (e.g., MNIST dataset).
- Recommender systems.

- Text classification.

2. Radius Neighbors Classifier

Overview:

- Similar to k-NN but uses **all neighbors within a fixed radius** instead of a fixed number of neighbors.
- Sensitive to data density and the choice of radius.

Distance Measures:

- Same as k-NN (Euclidean, Manhattan, Minkowski, etc.).
- **Mahalanobis Distance** can be useful for handling correlated features.

Use Cases:

- Geographical classification (e.g., predicting climate zones).
- Outlier detection (when points have no neighbors within a radius).

3. Support Vector Machines (SVM) with Distance-Based Kernels

Overview:

- SVM finds a hyperplane that maximizes the margin between classes.
- While not inherently distance-based, **kernel functions** can be interpreted as similarity measures.

Kernel (Distance) Functions:

- **Radial Basis Function (RBF) Kernel:** $K(x,y)=\exp(-\gamma\|x-y\|^2$
- **Polynomial Kernel:** Measures similarity via polynomial combinations.
- **Cosine Similarity Kernel:** For text and NLP tasks.

Use Cases:

- Image classification.
- Text and sentiment analysis.
- Bioinformatics (e.g., protein classification).

4. Distance-Weighted k-NN

Overview:

- A variant of k-NN where **closer neighbors have more influence** than distant ones.
- Weights are typically inversely proportional to distance: $w_i=1/d(x,xi)^2$

- Distance Measures:
- Same as k-NN, but weighting amplifies the impact of closer points.

Use Cases:

- Real-time recommendation systems.
- Medical diagnosis with imbalanced datasets.

5. Learning Vector Quantization (LVQ)

Overview:

- A **prototype-based supervised classification** algorithm.
- LVQ adjusts prototype vectors during training to represent classes better.

Distance Measures:

- **Euclidean Distance:** Most commonly used.
- **Mahalanobis Distance:** For correlated features.
- Custom distances for domain-specific tasks.

Use Cases:

- Speech recognition.
- Fault diagnosis in industrial applications.

6. Nearest Centroid Classifier

Overview:

- Classifies points based on the **nearest centroid** of each class.
- Each class's centroid is computed as the mean of its points.

Distance Measures:

- **Euclidean Distance** (default).
- **Cosine Distance** for text classification.
- **Manhattan Distance** when dealing with absolute deviations.

Use Cases:

- Face recognition.
- Text document classification.

7. Prototype-Based Classifiers

These classifiers use representative points (prototypes) for each class.

Notable Algorithms:

- **Rocchio Classifier:** Used in information retrieval; relies on centroids.
- **Self-Organizing Maps (SOM):** Unsupervised, but with supervised variants for classification.

Distance Measures:

- Typically use Euclidean distance but can incorporate others depending on the data type.

Use Cases:

- Text categorization.
- Customer segmentation.
-

K-Nearest Neighbor Classifier:

The **k-Nearest Neighbor (k-NN)** classifier is one of the simplest and most intuitive machine learning algorithms. It classifies data points based on the labels of the **k** closest points in the feature space. It's a **non-parametric** and **lazy learning** algorithm, meaning it makes no assumptions about the underlying data distribution and has no explicit training phase

Key Concepts

- **k:** The number of nearest neighbors considered for classification.
- **Distance Metric:** Determines how the "nearest" neighbors are chosen (e.g., Euclidean, Manhattan, Minkowski).
- **Majority Voting:** The class with the most representatives among the nearest neighbors is assigned to the data point.
- **Lazy Learning:** The algorithm delays processing until classification time (no explicit training).

How k-NN Works

- **Choose k:** Select the number of neighbors.
- **Calculate Distance:** Compute the distance between the new point and existing points using a chosen metric.
- **Find Nearest Neighbors:** Identify the k data points with the smallest distances.

- **Vote for Class:** Assign the class that is most frequent among the k neighbors.
- **Euclidean Distance**
    - Formula: $d(x,y)=SQRT\sum_{i=1}^{n}(xi-yi)^2$
    - Suitable for continuous variables and when all features are on the same scale.
- **Manhattan Distance (L1 Norm)**
    - Formula: $d(x,y)=\sum_{i=1}^{n}|xi-yi|$
    - Measures the distance between points along axes at right angles (useful for grid-like path problems).
- **Minkowski Distance**
    - Generalization of Euclidean and Manhattan distances.
    - Formula: $d(x,y)=(\sum_{i=1}^{n}|xi-yi|^p)^{1/p}$
    - Parameter ppp determines the type of distance (e.g., p=1 for Manhattan, p=2 for Euclidean).
- **Chebyshev Distance**
    - Formula: $d(x,y)=max\ (|xi-yi|)$
    - Suitable when the maximum difference in any dimension is crucial.
- **Hamming Distance**
    - Measures the number of positions at which the corresponding elements differ.
    - Used for categorical variables and binary strings.

      $d(x,y)= \sum_{i=1}^{n} II(xi!=yi)$

  where II is the indicator function.

Use Cases:

- For categorical variables or binary strings.
- Common in text mining and bioinformatics (e.g., DNA sequence analysis).

<u>Advantages of k-NN</u>

✓ Simple and intuitive.
✓ Works well with multi-class classification.
✓ No training phase (good for dynamic datasets).
✓ Effective for low-dimensional data.

Disadvantages of k-NN

⚠ Computationally expensive at prediction time (especially with large datasets).
⚠ Sensitive to irrelevant or redundant features.
⚠ Performance degrades with high-dimensional data (**curse of dimensionality**).
⚠ Requires careful distance metric and k-value selection.

KNN Regression:

The **k-Nearest Neighbors Regression (k-NN Regression)** algorithm predicts the output for a data point by averaging the target values of its **k nearest neighbors**. It's a **non-parametric**, **instance-based** learning method suitable for both simple and complex regression tasks.

Key Concepts of k-NN Regression

- **k (Number of Neighbors):**
  - Determines how many neighbors contribute to the prediction.
  - Small k → captures local patterns (more variance).
  - Large k → smoother predictions (more bias).
- **Distance Metric:**
  - Determines the "closeness" between data points. Common metrics include:
    - **Euclidean Distance:** For continuous data.
    - **Manhattan Distance:** For grid-like data.
    - **Minkowski Distance:** General form of Euclidean and Manhattan.
    - **Hamming Distance:** For categorical/binary data.
- **Prediction Methods:**
  - **Uniform Weighting:** All neighbors contribute equally.
  - **Distance Weighting:** Closer neighbors have a stronger influence. $y^{\wedge}=\sum_{i=1}^{k} y_i/d(x,x_i)/\sum_{i=1}^{k} 1/d(x,x_i)$

How k-NN Regression Works

- **Choose k:** Decide the number of neighbors.

- **Compute Distances:** Calculate distances between the target point and all others.
- **Select Neighbors:** Identify the **k** closest data points.
- **Predict Output:**
    - **Uniform:** Mean of neighbors' target values.
    - **Weighted:** Weighted mean based on distance.

Advantages of k-NN Regression

✓ Simple and easy to implement.
✓ Non-linear—can capture complex relationships.
✓ No explicit training phase (good for dynamic datasets).
✓ Naturally handles multi-output regression.

⚡ Disadvantages of k-NN Regression

✓ Computationally expensive at prediction time.
✓ Sensitive to noisy data and outliers.
✓ Requires proper scaling of features.
✓ Performance decreases with high-dimensional data (**curse of dimensionality**).

Applications of k-NN Regression

- **Stock price prediction:** Based on historical trends.
- **House price estimation:** Considering features like location, size, and amenities.
- **Demand forecasting:** For retail and supply chain management.
- **Medical risk prediction:** Estimating patient risk scores.

<u>Performance of Classifiers:</u>

The **k-Nearest Neighbors (k-NN)** classifier is a simple yet powerful machine learning algorithm that classifies a data point based on how its **k nearest neighbors** are classified. The performance of the k-NN classifier is influenced by factors such as the choice of **k**, **distance metric**, **feature scaling**, and **data characteristics**.

Key Factors Affecting k-NN Classifier Performance

Choice of k (Number of Neighbors)

- **Small k (e.g., k=1):**
  - Low bias, high variance.
  - Sensitive to noise and outliers.
- **Large k:**
  - High bias, low variance.
  - Smoother decision boundary but may underfit.
- **Tip:** Use cross-validation to find the optimal k.

Distance Metrics

- Determines how the "closeness" of neighbors is measured.
- Common metrics:

Euclidean Distance (default): For continuous features.

Manhattan Distance: For high-dimensional data.

Minkowski Distance: Generalization of Euclidean and Manhattan.

Hamming Distance: For categorical data.

Tip: Select based on data type and feature distribution.

Feature Scaling

k-NN is sensitive to the scale of features because it relies on distance calculations.

- Scaling methods:

Standardization (Z-score normalization): Mean = 0, Std = 1.

Min-Max Scaling: Transforms features to a 0–1 range.

Tip: Always scale features before applying k-NN.

Performance Metrics for k-NN Classifier

Metric	Description	When to Use
Accuracy	Correct predictions / Total predictions	Balanced datasets.
Precision	TP / (TP + FP)	When false positives are costly.
Recall	TP / (TP + FN)	When false negatives are

		costly.
F1 Score	Harmonic mean of precision and recall	Imbalanced datasets.
ROC-AUC Score	Measure of model's ability to distinguish classes	Binary classification.
Confusion Matrix	Summarizes prediction results	Detailed error analysis.

Performance of Regression Algorithms

The **performance of regression algorithms** is evaluated based on how accurately they predict continuous outcomes. Understanding the performance metrics and factors influencing these models is essential for selecting and fine-tuning the best regression approach for a given task.

Key Metrics for Evaluating Regression Performance

Mean Absolute Error (MAE)

- **Definition:** Average of absolute differences between actual and predicted values.
- **Formula:** $MAE=1/n\sum_{i=1}^{n}|yi-y\hat{}i|$
- **Pros:** Interpretable and robust to outliers.
- **Cons:** Ignores direction of errors.

Mean Squared Error (MSE)

- **Definition:** Average of squared differences between actual and predicted values.
- **Formula:** $MSE=1/n\sum_{i=1}^{n}(yi-y\hat{}i)^2$
- **Pros:** Penalizes larger errors more than smaller ones.
- **Cons:** Sensitive to outliers.

Root Mean Squared Error (RMSE)

- **Definition:** Square root of MSE; interpretable in the same units as the target variable.
- **Formula:** RMSE=SQRT OFMSE
- **Pros:** Penalizes large errors; widely used.
- **Cons:** Still sensitive to outliers.

R^2 Score (Coefficient of Determination)

- **Definition:** Measures the proportion of variance in the dependent variable that is predictable from the independent variables.
- **Formula:** $R^2=1-SS_{res}SS_{tot}$
- **Pros:** Easy interpretation (ranges from 0 to 1).
- **Cons:** Can be misleading for non-linear relationships.

Adjusted R^2 Score

- **Definition:** Modified R^2 that adjusts for the number of predictors in the model.
- **Pros:** Better for comparing models with different numbers of predictors.

Mean Absolute Percentage Error (MAPE)

- **Definition:** Measures the average magnitude of the errors as a percentage.
- **Pros:** Easy interpretation in percentage terms.
- **Cons:** Cannot handle zero values in actual data.

Radius Distance Nearest Neighbor Algorithm

The **Radius Distance Nearest Neighbor (Radius Neighbors)** algorithm is a variant of the **k-Nearest Neighbors (k-NN)** algorithm. Instead of selecting a fixed number of neighbors (**k**), it selects **all points within a specified radius** (**r**) from a query point. This makes it particularly useful when data density varies, as it can adapt the number of neighbors considered based on local data distribution.

How Radius Neighbors Algorithm Works

Define a radius (r): A fixed radius distance around the query point.
Find neighbors: Identify all training points within that radius.
Classification or Regression:

Classification: The query point is assigned the most common class among neighbors.

Regression: The query point's value is the average (or weighted average) of the neighbors' values.
Edge case: If no neighbors are found within the radius, a fallback strategy is needed (e.g., return the mean target value).

Key Parameters

radius: The radius around each query point to search for neighbors.

weights: How to weight the contribution of neighbors. Options include:

'uniform': All neighbors have equal weight.

'distance': Closer neighbors have more influence.

algorithm: Algorithm used to compute nearest neighbors ('auto', 'ball_tree', 'kd_tree', 'brute').

metric: Distance metric (e.g., 'euclidean', 'manhattan', 'minkowski').

Advantages of Radius Neighbors Algorithm

✓ Adaptive: Automatically adjusts the number of neighbors based on local data density.
✓ Handles varying density better than k-NN.
✓ Flexible with different distance metrics.
✓ Can provide more stable predictions in dense regions.

Disadvantages of Radius Neighbors Algorithm

Sensitive to the choice of radius—too small may exclude neighbors; too large may include irrelevant points.
Computationally intensive for large datasets.
May require tuning of radius for optimal performance.