KNN & PCA Assignment

Q1. What is K-Nearest Neighbors (KNN) and how does it work?

Ans. K-Nearest Neighbors (KNN) is a **supervised learning algorithm** used for **classification and regression** tasks. It is a **non-parametric** and **lazy learning** algorithm, meaning it makes no assumptions about the data distribution and does not learn a model during training. Instead, it memorizes the training dataset and makes predictions based on similarity.

How KNN Works?

1. **Choose a value for K (number of neighbors).**
    - A small K makes the model sensitive to noise.
    - A large K smoothens the decision boundary but may overlook finer patterns.
2. **Calculate the distance between the test data and all training data points.**
    - Common distance metrics:
        - **Euclidean Distance** (most common)
        - Manhattan Distance
        - Minkowski Distance
3. **Find the K-nearest neighbors.**
    - Sort the training samples by their distance to the test sample.
    - Select the top K nearest points.
4. **Make a prediction.**
    - **For classification** → Assign the most common class (majority vote).
    - **For regression** → Take the average of the K nearest values.

Q2. What is the difference between KNN Classification and KNN Regression?

Ans. K-Nearest Neighbors (KNN) can be used for both **classification** and **regression**, but they differ in how predictions are made.

1. KNN Classification

- **Purpose**: Assigns a class label to the input based on majority voting.
- **Prediction Method**:
    - Finds the K nearest neighbors.
    - Determines the **most common** class among them.
    - Assigns this majority class to the new data point.
- **Output**: A **categorical label** (e.g., "Spam" or "Not Spam").
- **Example Use Case**: Email spam detection, medical diagnosis, sentiment analysis.

Example:
If K = 5 and among the 5 nearest neighbors:

- 3 belong to **Class A**

- 2 belong to **Class B**
  Then, the new data point is classified as **Class A**.

2. KNN Regression

- **Purpose**: Predicts a continuous numerical value.
- **Prediction Method**:
  - Finds the K nearest neighbors.
  - Computes the **average (or weighted average)** of their values.
  - Assigns this value to the new data point.
- **Output**: A **continuous value** (e.g., price, temperature, sales).
- **Example Use Case**: House price prediction, weather forecasting, stock price prediction.

Example:
If K = 3 and the nearest neighbors have values **50, 60, and 70**,

- The predicted value would be **(50 + 60 + 70) / 3 = 60**.

Key Differences

Feature	KNN Classification	KNN Regression
Output Type	Categorical (Labels)	Continuous (Numbers)
Decision Method	Majority Voting	Average or Weighted Average
Use Case	Spam detection, Disease diagnosis	House price prediction, Temperature forecasting

Q3. What is the role of the distance metric in KNN?

Ans.

Role of the Distance Metric in KNN

The **distance metric** in K-Nearest Neighbors (KNN) plays a crucial role in determining how "close" or "similar" data points are. It directly affects the **accuracy and performance** of the model.

Why is Distance Important?

1. **Neighbor Selection** → The K closest data points are chosen based on the distance metric.
2. **Prediction Accuracy** → A good distance metric ensures that similar data points influence predictions more effectively.
3. **Feature Scaling Sensitivity** → Some distance metrics are sensitive to the scale of data, requiring feature normalization.

Q4. What is the Curse of Dimensionality in KNN?

Ans. The **Curse of Dimensionality** refers to the problems that arise when working with high-dimensional data, where the number of features (**dimensions**) increases significantly. This phenomenon negatively impacts **K-Nearest Neighbors (KNN)** and other machine learning algorithms.

How Does It Affect KNN?

1. **Distance Becomes Less Meaningful**
   - In high dimensions, **all points tend to become equidistant** from each other.
   - KNN relies on finding the **nearest neighbors** based on distance metrics (e.g., Euclidean), but when distances become similar, the algorithm struggles to distinguish between relevant and irrelevant points.
2. **Increased Computational Cost**
   - KNN is a **lazy learner**, meaning it calculates distances at runtime.
   - As dimensions grow, computing distances becomes **exponentially more expensive**, making KNN slower.
3. **Sparsity of Data**
   - In high-dimensional space, data points spread out, and the dataset becomes sparse.
   - The nearest neighbors might not be **truly "near"**, leading to poor classification or regression performance.
4. **Overfitting Risk**
   - More dimensions mean more **noise**, which increases the likelihood of **overfitting**.
   - The model may learn **irrelevant patterns** instead of meaningful ones.

Q5. How can we choose the best value of K in KNN?

Ans. Choosing the optimal value of **K (number of neighbors)** is crucial for achieving the best performance in **K-Nearest Neighbors (KNN)**. The right K balances **bias vs. variance** and improves classification or regression accuracy.

Q6. What are KD Tree and Ball Tree in KNN?

Ans. When implementing **K-Nearest Neighbors (KNN)**, searching for the nearest neighbors can be computationally expensive, especially for **large datasets**. To speed up this search, we use **KD Tree** and **Ball Tree**, which are efficient **spatial data structures**.

1. KD Tree (K-Dimensional Tree)

What is it?

- A **binary tree** used for organizing points in a **K-dimensional space**.
- Efficient for low to **moderate-dimensional** datasets (e.g., up to ~20 dimensions).
- Reduces search time from **O(N)** (brute force) to **O(log N)** in the best case.

How it Works?

1. Select a **splitting dimension** (e.g., x, y, z, etc.).
2. Choose a **median value** along that dimension.

3. Recursively divide data into **left (≤ median) and right (> median) subtrees**.
4. At query time, use **tree traversal** to efficiently find the K nearest neighbors.

When to Use KD Tree?

Works well for **low-dimensional** data (e.g., ≤ 20 dimensions).
Performance degrades in **high dimensions** due to the **Curse of Dimensionality**.

2. Ball Tree

What is it?

- A **hierarchical tree structure** where data points are grouped into **nested hyperspheres (balls)**.
- Useful for **high-dimensional data** where KD Tree struggles.

How it Works?

1. Build a tree by recursively **dividing data into clusters (balls)**.
2. Each node has:
   - A center point.
   - A **radius** that encloses all points inside it.
3. At query time, prune entire **balls** that are too far from the query point, speeding up the search.

When to Use Ball Tree?

Works better than KD Tree for **high-dimensional** (e.g., >20 dimensions) or **non-Euclidean distance**.
Handles **sparse data** well (e.g., text embeddings, word vectors).

Q7. When should you use KD Tree vs. Ball Tree?

Ans. Both **KD Tree** and **Ball Tree** speed up **K-Nearest Neighbors (KNN) search**, but their efficiency depends on the **data's dimensionality and distribution**. Here's how to decide which one to use:

1. Use KD Tree When:

Data has low to moderate dimensions (≤ 20 features).
Distance metric is Euclidean (L2 norm).
Dataset is dense (not sparse).
You need **fast nearest neighbor search** in structured data like **image recognition, GPS navigation, and spatial data**.

Avoid KD Tree if the data has **too many dimensions** (e.g., >20D), as it suffers from the **Curse of Dimensionality**, making searches inefficient.

2. Use Ball Tree When:

Data has high dimensions (>20 features).
Supports any distance metric (e.g., Minkowski, Mahalanobis, Hamming).
Handles sparse data well (e.g., text embeddings, word vectors).
Works well for **non-Euclidean spaces** (e.g., cosine similarity in NLP).

Avoid Ball Tree if the data is **low-dimensional**, as KD Tree is usually faster in such cases.

Q8. What are the disadvantages of KNN?

Ans. K-Nearest Neighbors (KNN) is a simple yet powerful algorithm, but it has several drawbacks that can impact its performance.

1. High Computational Cost (Slow for Large Datasets)
2. Sensitive to High Dimensions (Curse of Dimensionality)
3. Poor Performance on Imbalanced Data
4. Requires Careful Selection of K
5. Does Not Work Well with Categorical Data
6. Memory-Intensive
7. Not Robust to Outliers

Q9. How does feature scaling affect KNN?

Ans. Feature scaling is **crucial** for K-Nearest Neighbors (KNN) because KNN **relies on distance metrics** to determine the nearest neighbors. If features are on different scales, KNN may produce biased or incorrect results.

Q10. What is PCA (Principal Component Analysis)?

Ans. Principal Component Analysis (PCA) is a **dimensionality reduction technique** used to transform high-dimensional data into a lower-dimensional space while preserving as much information (variance) as possible.

It helps:
Reduce computation time for ML models.
Avoid overfitting by eliminating redundant features.
Visualize high-dimensional data in 2D or 3D.

Q11. How does PCA work?

Ans.

Step 1: Standardize the Data

- PCA is sensitive to feature scaling, so we **normalize** or **standardize** the data to ensure all features contribute equally.

Step 2: Compute the Covariance Matrix

- Measures relationships between features to identify **correlated variables**.

Step 3: Compute Eigenvalues & Eigenvectors

- **Eigenvectors** represent the **principal components (PCs)** (new feature axes).
- **Eigenvalues** indicate the **importance (variance explained)** of each principal component.

Step 4: Select Top K Principal Components

- We pick the **K most significant principal components** based on their eigenvalues.

Step 5: Transform Data to New Space

- Original data is projected onto the new **reduced-dimension space**.

Q12. What is the geometric intuition behind PCA?

Ans. PCA can be understood **geometrically** as a way of finding a new coordinate system that best represents the data in a lower-dimensional space. It does this by identifying the directions (principal components) along which the data **varies the most**.

1. Visualizing PCA in 2D

Step 1: Original Data in High Dimensions

Imagine we have **2D data points** spread in an **elliptical shape** (not axis-aligned).

- The data has two original axes (**X and Y**).
- However, the data is **more spread out in one direction** than the other.

Step 2: Finding the Principal Components

- **PCA finds the directions of maximum variance** (spread) in the data.
- These directions are represented as **new axes**, called **Principal Components (PCs)**.
- The **first principal component (PC1)** is the direction with the **most variance** (most information).
- The **second principal component (PC2)** is **perpendicular to PC1** and captures the remaining variance.

Step 3: Rotating the Axes

- PCA effectively **rotates the data** to align with the new principal axes (PC1 and PC2).
- The new coordinate system **eliminates correlation** between the variables.

Step 4: Projecting onto Lower Dimensions

- If we want to reduce dimensions (e.g., from **2D → 1D**), we **drop the least important principal components** (e.g., PC2).
- The **data is projected onto PC1**, **preserving maximum variance while reducing dimensions**.

2. Extending to 3D and Higher Dimensions

- In **3D**, the process is the same:
  - **PC1:** Captures the most variance.
  - **PC2:** Perpendicular to PC1, captures the second most variance.
  - **PC3:** Perpendicular to both, captures the least variance.
- If reducing **3D → 2D**, we drop the component with the least variance (PC3) and keep PC1 & PC2.

In **higher dimensions (e.g., 100D → 10D)**, PCA generalizes this idea by **finding the top K directions** that retain most of the data's variability.

3. PCA as an Optimization Problem

- PCA finds a new set of **orthogonal** axes by solving an **eigenvalue problem** on the covariance matrix.
- **Mathematically**, PCA finds the eigenvectors (principal components) and eigenvalues (amount of variance explained).

Q13. What is the difference between Feature Selection and Feature Extraction?

Ans. Both **Feature Selection** and **Feature Extraction** are **dimensionality reduction** techniques, but they differ in **how they reduce the number of features**.

Aspect	**Feature Selection**	**Feature Extraction**
Approach	Keeps only important features	Creates new transformed features
Interpretability	Retains original feature meanings	New features may not be interpretable
Methods	Correlation, RFE, LASSO	PCA, LDA, t-SNE, Autoencoders
Use Case	Reduce noise & redundancy	Reduce dimensionality & improve performance
Example	Choosing 5 out of 10 features	Creating 2 new features from 10

Q14. What are Eigenvalues and Eigenvectors in PCA?

Ans. Eigenvalues and eigenvectors play a crucial role in **Principal Component Analysis (PCA)** by helping to find the principal components (new axes) that best represent the data.

What Are Eigenvalues and Eigenvectors?

Eigenvectors

- **Eigenvectors define directions** in which data spreads the most.
- They represent the **principal components (PCs)** in PCA.
- Each eigenvector is a **unit vector (length = 1)** that points in a specific direction in the data space.

Eigenvalues

- **Eigenvalues represent the magnitude of variance** in the direction of its corresponding eigenvector.
- A **higher eigenvalue** means **more variance** along that eigenvector.
- The sum of all eigenvalues represents **total variance in the dataset**.

Q15. How do you decide the number of components to keep in PCA?

Ans. Choosing the right number of principal components (**K**) is crucial in PCA to balance between **dimensionality reduction** and **information retention**. Here are some common methods to decide **how many principal components to keep**:

1. Explained Variance (Variance Retention Method)

The most common approach is to **retain enough components** to explain a high percentage (e.g., 95%) of the total variance.

Steps:

1. Compute **eigenvalues** (which represent variance captured by each principal component).
2. Compute the **cumulative explained variance**.
3. Choose the smallest number of components that explain at least **90-95%** of the variance.

2. Scree Plot (Elbow Method)

A scree plot helps visualize the eigenvalues (variance explained by each component).
The **elbow point** is where adding more components **gives diminishing returns**.

How It Works?

- Plot eigenvalues (variance explained) against component numbers.
- Find the **elbow point** where variance stops decreasing significantly.

3. Kaiser's Rule (Eigenvalue > 1 Rule)

Keep components with **eigenvalues > 1** (significant variance contribution).
Works well when dealing with correlation matrices in PCA.

- Compute eigenvalues.
- Retain only components with eigenvalues greater than **1**.

4. Cross-Validation (Model-Based Approach)

If PCA is used for **machine learning**, we can evaluate different K values using **cross-validation**.

Steps:

1. Try different values of K.
2. Train an ML model (e.g., logistic regression) on transformed data.
3. Choose the K that gives the best performance.

5. Domain Knowledge

If you have domain expertise, you can **manually** decide based on what makes sense.
Example:

- In **image processing**, reducing 1000D to 50D might be fine.
- In **finance**, 5 key factors (like market trends) may be sufficient.

Q16. Can PCA be used for classification?

Ans. PCA is NOT a classification algorithm, but it can be used as a **preprocessing step** to improve classification performance.

How PCA Helps in Classification

Dimensionality Reduction

- High-dimensional datasets can cause **overfitting** in classification models.
- PCA **reduces the number of features** while retaining important information.

Noise Reduction

- PCA **removes redundant and less informative features**, improving model accuracy.

Better Visualization

- PCA projects data onto **2D or 3D** space, making it easier to visualize class separation.

Q17. What are the limitations of PCA?

Ans. While **PCA** is a powerful dimensionality reduction technique, it has several limitations that you should be aware of.

1. Loss of Interpretability

PCA **transforms original features** into new principal components.
These new features are **linear combinations** of original ones, making them **hard to interpret**.
Example: If you analyze customer behavior, PCA won't tell you **which original features** (e.g., age, income) are most important.

2. Assumes Linearity

PCA finds **linear** relationships between variables.
If your data has **non-linear patterns**, PCA **may not be effective**.
Alternative: **t-SNE or UMAP** for non-linear dimensionality reduction.

3. Sensitive to Scaling

PCA is affected by **different feature scales** (e.g., height in cm vs. weight in kg).
If features have different ranges, **PCA may give more importance to higher-magnitude features**.
Solution: Always **standardize (z-score) or normalize** data before applying PCA.

4. Sensitive to Outliers

PCA uses **variance** to determine principal components.
Outliers can distort variance, leading to misleading principal components.
Solution: Remove outliers before applying PCA.

5. Does Not Work Well with Sparse Data

PCA works well with **dense datasets**.
If data is **sparse** (e.g., text data with many zeros), PCA may **not perform well**.
Solution: Use **Truncated SVD** (a variant of PCA) for sparse data.

6. Can Remove Important Features

PCA keeps components with the **highest variance**.
If a feature has **low variance** but is **important for classification**, PCA **might discard it**.
Example: In fraud detection, rare fraud transactions have low variance but are crucial.

7. Not Suitable for Categorical Data

PCA is designed for **continuous numerical data**.
It **doesn't work well** with **categorical variables** (e.g., gender, country).
Solution: Use **one-hot encoding** before PCA or try **Factor Analysis for categorical data**.

8. Computationally Expensive for Large Datasets

PCA requires **eigenvalue decomposition or Singular Value Decomposition (SVD)**.
For **very large datasets**, these operations become **computationally expensive**.
Solution: Use **Randomized PCA or Incremental PCA** for large datasets.

Q18. How do KNN and PCA complement each other?

Ans. K-Nearest Neighbors (**KNN**) and Principal Component Analysis (**PCA**) can work together effectively to improve classification and regression tasks. PCA helps **prepare the data**, and KNN makes **predictions based on neighbors**.

Why Use PCA Before KNN?

Dimensionality Reduction

KNN is sensitive to **high-dimensional data** due to the **Curse of Dimensionality** (distance calculations become less meaningful in high dimensions).
PCA reduces the number of features while **preserving most of the variance**, making KNN more effective.

Noise Reduction

PCA removes **irrelevant and redundant features**, which helps KNN focus on more informative data.

Faster Computation

KNN is **computationally expensive** because it calculates distances for every query point.
By using PCA, we **reduce the number of features**, making distance calculations **faster**.

When Should You Use PCA Before KNN?

Use PCA before KNN when:

- The dataset has **many features (high-dimensional)**.
- There is **multicollinearity** (high correlation between features).
- You want to **speed up KNN** computations.
- Your dataset contains **redundant or noisy features**.

Don’t use PCA before KNN if:

- The dataset is **already low-dimensional** (e.g., less than 5 features).
- Feature relationships are important (PCA removes original feature meaning).
- The dataset is **non-linear** (PCA finds only linear patterns).

Q19. How does KNN handle missing values in a dataset?

Ans. K-Nearest Neighbors (**KNN**) does not handle missing values directly, but you can use **imputation techniques** before applying KNN. Since KNN relies on **distance calculations**, missing values can cause issues in finding the nearest neighbors.

1. Common Strategies for Handling Missing Values in KNN

A. Remove Rows with Missing Values (Not Recommended)

Simple and quick.
Risk: If too many rows are removed, it can lead to **loss of important data**.

B. Mean/Median/Mode Imputation

Replace missing values with the **mean (for numerical), median, or mode (for categorical)**.
Risk: This method **does not consider the relationships between features**.

C. KNN Imputation (Best for KNN Classifier/Regressor)

Uses **K-Nearest Neighbors** to predict missing values based on the most similar points.
More accurate because it considers **feature relationships**.
Computationally expensive for large datasets.

2. When to Use KNN Imputation?

Use KNN Imputation if:

- You have **many missing values** but don’t want to drop rows.
- You believe missing values **depend on other features**.
- Your dataset is not **too large** (KNN is computationally expensive).

Don’t use KNN Imputation if:

- Your dataset is **too large**, as KNN is slow.
- The missing values are **completely random** (mean/median may be enough).

Q20. What are the key differences between PCA and Linear Discriminant Analysis (LDA)?

Ans. Both **Principal Component Analysis (PCA)** and **Linear Discriminant Analysis (LDA)** are dimensionality reduction techniques, but they serve **different purposes**.

Feature	**PCA (Principal Component Analysis)**	**LDA (Linear Discriminant Analysis)**
Goal	Reduce dimensions by maximizing variance	Reduce dimensions while maximizing class separation
Type of Learning	Unsupervised	Supervised (requires class labels)
Works on	Features (ignores class labels)	Features & class labels
Mathematical Basis	Eigenvalues & Eigenvectors of covariance matrix	Fisher's Discriminant (maximize between-class variance)
Max Components	At most **number of features**	At most **number of classes - 1**
Interpretability	Principal Components may not always be meaningful	Features are projected to separate classes better
When to Use?	High-dimensional data, removing redundancy	Classification tasks where class separation is key