1.
KNN (K-Nearest Neighbors) is a lazy learning algorithm used for classification and regression. It finds the K closest data points using a distance metric and assigns the majority class
(classification) or averages values (regression). It is simple but computationally expensive for
large datasets.

2.
KNN Classification assigns the majority class among the K nearest neighbors, making it suitable for categorical labels. KNN Regression, on the other hand, predicts continuous values by averaging the K nearest numerical values. Classification works with discrete outputs, whereas regression deals with continuous outputs.

3.
The distance metric in KNN determines how similarity between points is measured. Common metrics include Euclidean (straight-line distance), Manhattan (sum of absolute differences), and Minkowski (generalized form). Choosing an appropriate distance metric significantly impacts the accuracy and efficiency of KNN models.

4.
The Curse of Dimensionality in KNN occurs when higher dimensions make all points appear equidistant, reducing model effectiveness. This increases computational cost, affects distance calculations, and leads to poor classification performance. PCA or feature selection can help mitigate this issue by reducing dimensionality.

5.
The best value of K in KNN is chosen using cross-validation. A small K may cause overfitting, while a large K can lead to underfitting. The **elbow method** helps determine the optimal K by plotting accuracy against different K values and selecting where performance stabilizes.

6.
KD Tree and Ball Tree are optimized data structures for faster nearest-neighbor searches. KD Tree is useful for low-dimensional spaces and partitions data using hyperplanes, while Ball Tree is more effective for high-dimensional datasets, grouping points into hierarchical spherical clusters for efficient searches.

7.
Use **KD Tree** when the dataset has low dimensions (≤ 20) since it efficiently partitions the space. **Ball Tree** is preferred for higher dimensions (> 20) as it groups data into hierarchical structures, handling complex distance metrics better than KD Tree.

8.
Disadvantages of KNN include high computation time for large datasets, sensitivity to irrelevant features, poor performance in high dimensions, and difficulty handling imbalanced datasets. It requires distance calculations for every prediction, making it inefficient without optimizations like KD Tree or Ball Tree.

9.
Feature scaling (standardization or normalization) ensures all features contribute equally to distance calculations. Without scaling, features with larger ranges dominate the distance metric, leading to biased results. Common techniques include **Min-Max Scaling** (scaling between 0 and 1) and **Z-score Standardization** (zero mean, unit variance).

10.
Principal Component Analysis (PCA) is a dimensionality reduction technique that converts correlated features into a smaller set of uncorrelated principal components while retaining maximum variance. It helps improve efficiency, visualization, and performance in machine learning models by reducing redundant data.

11.
PCA works by standardizing data, computing the covariance matrix, finding eigenvalues and eigenvectors, selecting principal components, and transforming data into a new subspace. It helps reduce dimensions while preserving the most significant variance in the dataset.

12.
The geometric intuition behind PCA is that it finds new orthogonal axes that maximize variance. The first principal component captures the highest variance, the second captures the next highest variance, and so on. This transformation projects data onto a lower-dimensional hyperplane while retaining essential information.

13.
Feature Selection selects the most relevant features from the dataset, whereas Feature Extraction (PCA) creates new transformed features by combining existing ones. Feature

Selection keeps original features, while Feature Extraction derives new ones to improve performance by reducing dimensionality.

14.
Eigenvalues and eigenvectors in PCA determine the principal components. **Eigenvalues** measure variance explained by each component, while **eigenvectors** define the direction of these components in feature space. Higher eigenvalues correspond to components that retain more data variance.

15.
The number of components to keep in PCA is chosen based on the **explained variance ratio**. Typically, enough components are selected to retain **95% or more variance**. The **scree plot** (elbow method) helps visualize how many components contribute significantly.

16.
PCA can be used for classification as a **preprocessing step** to reduce dimensionality, improve computational efficiency, and remove redundant features. However, PCA itself is not a classification algorithm—it only transforms features before applying a classifier like KNN, SVM, or Logistic Regression.

17.
PCA has limitations such as loss of interpretability, sensitivity to feature scaling, assumption of linearity, and inability to handle non-linear relationships. It works best with normally distributed data and may not always improve model accuracy in classification tasks.

18.
KNN and PCA complement each other because PCA reduces dimensionality, making KNN computationally efficient. PCA removes correlated features, improving distance-based classification in KNN. This combination works well when dealing with high-dimensional datasets where KNN alone suffers from slow performance.

19.
KNN handles missing values using **KNN Imputation**, where missing values are replaced with the mean, median, or mode of their nearest neighbors. This method preserves relationships in data but requires careful selection of K to avoid incorrect imputation.

20.

PCA focuses on **maximizing variance** in an unsupervised way, while Linear Discriminant Analysis (LDA) is a **supervised technique** that maximizes class separation. PCA works for both regression and classification, whereas LDA is specifically designed for classification tasks.