Q1. What is the KNN algorithm?

KNN (K-Nearest Neighbors) is a **supervised machine learning algorithm** used for **classification** and **regression**.

- It works on the principle of **similarity**: given a new data point, it looks at the K closest points (neighbors) in the training set and predicts based on them.
- For **classification**, it predicts the most frequent class among neighbors.
- For **regression**, it predicts the average (or weighted average) of the neighbors' values.
- **Key idea:** "Birds of a feather flock together" — similar data points are near each other in the feature space.

Q2. How do you choose the value of K in KNN?

- **Small K (e.g., 1 or 3):**
    - Captures fine patterns but is sensitive to noise → **high variance**, risk of overfitting.
- **Large K (e.g., 20 or more):**
    - Smooths predictions, more robust → **high bias**, risk of underfitting.
- **Practical approach:**
    - Use **cross-validation** to test multiple K values and pick the one with the best performance.
    - Often, **odd values** are chosen in classification to avoid ties.

Q3. What is the difference between KNN classifier and KNN regressor?

Aspect	KNN Classifier	KNN Regressor
Task	Classification	Regression

Prediction	Most frequent class among K neighbors	Average (or weighted) value of K neighbors
Output	Discrete label	Continuous value
Example	Predict if an email is spam or not	Predict house price based on nearby houses

---

Q4. How do you measure the performance of KNN?

- **For classification:**
    - Accuracy, Precision, Recall, F1-score, Confusion Matrix, ROC-AUC.
- **For regression:**
    - Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R^2 score.

Q5. What is the curse of dimensionality in KNN?

- As the **number of features (dimensions)** increases, the distance between points becomes less meaningful.
- In high dimensions:
    - All points start to appear **equidistant**.
    - KNN struggles to find truly “nearest” neighbors → performance drops.
- **Solution:** Dimensionality reduction (PCA, feature selection) before applying KNN.

Q6. How do you handle missing values in KNN?

- **Option 1:** Impute missing values using KNN itself:

    - Find neighbors based on non-missing features and fill missing values using neighbors' mean (regression) or mode (classification).

- **Option 2:** Remove data points or features with too many missing values.
- **Option 3:** Use other imputation techniques (mean, median, or model-based).

Q7. Compare and contrast the performance of KNN classifier and regressor. Which one is better for which type of problem?

- **KNN Classifier:**
    - Works well when classes are well-separated and boundaries are simple.
    - Sensitive to **noisy labels**.
- **KNN Regressor:**
    - Works best when target varies smoothly with input features.
    - Sensitive to **outliers**, since they affect the mean.
- **Rule of thumb:**
    - Classification → KNN classifier.
    - Continuous target → KNN regressor.
- Neither is inherently "better"; performance depends on **problem type, dimensionality, and noise**.

Q8. Strengths and weaknesses of KNN and how to address them

Strengths:

- Simple, intuitive, non-parametric (no assumption about data distribution).
- Naturally handles multi-class problems.

- Can adapt to both classification and regression.

Weaknesses:

1. **Computationally expensive** at prediction (needs distance calculation to all points).
    - *Solution:* Use KD-Trees, Ball-Trees, approximate nearest neighbors.
2. **Sensitive to irrelevant features and scaling.**
    - *Solution:* Feature scaling (MinMax, Standardization), feature selection.
3. **Affected by noise and outliers.**
    - *Solution:* Use larger K, distance weighting.
4. **Curse of dimensionality.**
    - *Solution:* Dimensionality reduction (PCA, t-SNE) or feature selection.

Q9. Difference between Euclidean distance and Manhattan distance in KNN

Distance	Formula (2D)	Characteristics
Euclidean	$\sqrt{((x1 - x2)^2 + (y1 - y2)^2)}$	“Straight line” distance, sensitive to large differences
Manhattan	x1 - x2	

- Choice depends on **data nature**: Euclidean for continuous, dense features; Manhattan for high-dimensional or sparse features.

Q10. Role of feature scaling in KNN

- KNN relies on **distance metrics**, so features with larger scales dominate the distance calculation.

- **Example:** Height (cm) vs. weight (kg). Height might overpower weight if unscaled.
- **Solution:** Normalize or standardize features:
    - **Min-Max Scaling:** scales to [0,1]
    - **Standardization:** mean = 0, std = 1

Without scaling, KNN predictions can be highly biased toward features with larger ranges.
