**Q1. What is the KNN algorithm?**

The k-Nearest Neighbors (KNN) algorithm is a non-parametric machine learning algorithm that can be used for classification and regression tasks. In KNN, the input consists of the k closest training examples in the feature space. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k=1, then the object is simply assigned to the class of its nearest neighbor. KNN is a simple and effective algorithm that can be used for both binary and multiclass classification problems, as well as for regression tasks.

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

Cross-validation: The dataset is split into training and validation sets. The model is trained on the training set using different values of K and the performance is evaluated on the validation set. The value of K that results in the best performance is chosen.

Rule of thumb: A commonly used rule of thumb is to set K to the square root of the number of samples in the dataset. However, this may not always be optimal.

Grid search: The algorithm is run with different values of K and the performance is evaluated on a separate test set. The value of K that results in the best performance is chosen.

Domain knowledge: The value of K can also be chosen based on the knowledge of the problem at hand. For example, in a medical diagnosis problem, a value of K that is known to provide good accuracy in similar studies may be used.

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

KNN classifier is a supervised learning algorithm that is used for classification problems where the output variable is categorical. It predicts the class of a new observation based on the majority class of its k nearest neighbors in the feature space. In other words, it assigns the most common class among the k nearest neighbors to the new observation.

On the other hand, KNN regressor is also a supervised learning algorithm that is used for regression problems where the output variable is continuous. It predicts the value of a new observation by taking the average of the values of its k nearest neighbors in the feature space. In other words, it assigns the average value of the k nearest neighbors to the new observation.

Therefore, KNN classifier and KNN regressor differ in the type of output they produce, with KNN classifier predicting the class of a new observation and KNN regressor predicting the value of a new observation.

**Q4. How do you measure the performance of KNN?**

Accuracy: It measures the proportion of correct predictions made by the model out of the total predictions made. It is calculated as the number of correct predictions divided by the total number of predictions.

Precision: It measures the proportion of true positives among the predicted positives. It is calculated as the number of true positives divided by the sum of true positives and false positives.

Recall: It measures the proportion of true positives among the actual positives. It is calculated as the number of true positives divided by the sum of true positives and false negatives.

F1-score: It is the harmonic mean of precision and recall, and provides a combined measure of both metrics. It is calculated as 2 times the product of precision and recall divided by the sum of precision and recall.

Confusion matrix: It is a table that summarizes the actual and predicted class labels for a set of instances, and provides information on the number of true positives, true negatives, false positives, and false negatives.

**Q5. What is the curse of dimensionality in KNN?**

The curse of dimensionality in KNN refers to the problem of the increasing computational cost and decreasing predictive power of the KNN algorithm as the number of dimensions (features) in the dataset increases.

In high-dimensional spaces, the volume of the space grows exponentially with the number of dimensions, making it difficult to identify meaningful patterns or distances between data points. This can lead to overfitting, where the KNN model becomes too complex and memorizes the training data instead of learning general patterns, resulting in poor performance on new, unseen data.

To overcome the curse of dimensionality in KNN, various dimensionality reduction techniques can be used, such as principal component analysis (PCA), linear discriminant analysis (LDA), or t-distributed stochastic neighbor embedding (t-SNE). These techniques can help to reduce the number of dimensions and extract the most relevant features, improving the performance of the KNN algorithm.

**Q6. How do you handle missing values in KNN?**

Removal of instances with missing values: One way is to simply remove instances with missing values from the dataset. However, this method can lead to loss of important information and reduced accuracy of the model.

Imputation: Another way is to impute the missing values with a substitute value. The substitute value can be the mean, median or mode of the other values in the same feature or the predicted value using a regression model.

KNN Imputation: KNN can be used to impute the missing values. In this method, the KNN algorithm is applied to identify the K-nearest neighbors to the instance with missing values. Then, the missing values are replaced with the average or weighted average of the values of the corresponding features of the K-nearest neighbors.

Model-based imputation: Model-based imputation involves using a regression model to predict the missing values. This method is useful when there is a strong correlation between the features and the target variable.

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

The KNN classifier is suitable for classification problems, where the target variable is categorical. It works well when the classes are well-separated and the decision boundary is non-linear. However, it may not perform well when the classes are overlapping or when the data is imbalanced. In addition, it can be computationally expensive when dealing with large datasets.

On the other hand, the KNN regressor is suitable for regression problems, where the target variable is continuous. It works well when the relationship between the features and the target variable is non-linear. However, it may not perform well when the data is noisy or when there are outliers. It is also computationally expensive when dealing with large datasets.

**Q8. What are the strengths and weaknesses of the KNN algorithm for classification and regression tasks, and how can these be addressed?**

Strengths of KNN:

Simplicity: KNN is a simple and easy-to-implement algorithm that requires no assumptions about the underlying data distribution.

Non-parametric: KNN is a non-parametric algorithm, meaning it does not make any assumptions about the shape or form of the underlying data distribution.

Versatility: KNN can be used for both classification and regression tasks.

Robustness to noisy data: KNN is relatively robust to noisy data, as outliers tend to have less influence on the final prediction due to the averaging effect of the algorithm.

Weaknesses of KNN:

Computational complexity: KNN can be computationally expensive, especially for large datasets, as it requires calculating the distance between each pair of data points.

Sensitivity to feature scaling: KNN is sensitive to the scaling of features, as features with larger ranges can dominate the distance metric.

The curse of dimensionality: KNN is prone to the curse of dimensionality, meaning its performance can degrade as the number of features increases, as it becomes harder to find meaningful distances in high-dimensional spaces.

Bias-variance trade-off: KNN has a high variance, meaning its performance can be sensitive to the choice of K, and it can suffer from overfitting or underfitting.

These strengths and weaknesses can be addressed by various techniques:

To address computational complexity, techniques such as KD-trees or Ball-trees can be used to speed up the search for nearest neighbors.

To address sensitivity to feature scaling, it is often a good idea to normalize or standardize the features before applying KNN.

To address the curse of dimensionality, techniques such as Principal Component Analysis (PCA) or feature selection can be used to reduce the number of features.

To address the bias-variance trade-off, techniques such as cross-validation can be used to tune the value of K and prevent overfitting or underfitting.

**Q9. What is the difference between Euclidean distance and Manhattan distance in KNN?**

Euclidean distance is the straight-line distance between two points in a Euclidean space. It is calculated as the square root of the sum of the squared differences between the corresponding elements of the two points. In other words, it measures the geometric distance between two points, which is also known as the L2 norm.

On the other hand, Manhattan distance is the distance between two points measured along the axes at right angles. It is calculated as the sum of the absolute differences between the corresponding elements of the two points. In other words, it measures the distance between two points by moving along the grid-like structure formed by the features, which is also known as the L1 norm.

**Q10. What is the role of feature scaling in KNN?**

Feature scaling is an important step in KNN because it can affect the distance metric used to calculate the distance between two data points. In KNN, the distance between two data points is used to determine the similarity between them. If the features have different scales, the

```
In [ ]:
```