KNN

Interview Questions -1

(Practice Project)

PW SKILLS

Easy

1. How does the choice of k in k-NN affect bias and variance in the model?

Answer: As k increases, bias tends to increase (more generalization) while variance decreases (less sensitive to individual data points). A smaller k leads to a more flexible model but risks overfitting, while a larger k produces a smoother decision boundary but may underfit.

2. Describe an efficient way to implement k-NN for large-scale datasets that don't fit in memory.

Answer: Use techniques like data chunking and out-of-core learning. Implement a disk-based k-D tree or Ball tree, or use approximate nearest neighbor methods like Locality Sensitive Hashing (LSH) combined with a distributed computing framework like Apache Spark.

3. How would you handle class imbalance in a k-NN classifier?

Answer: Approaches include: 1) Adjusting class weights inversely proportional to class frequencies, 2) Oversampling minority classes or undersampling majority classes, 3) Using distance-weighted voting, giving more weight to closer neighbors from minority classes.

4. Explain how you would adapt k-NN for streaming data where new points arrive continuously.

Answer: Implement an online k-NN algorithm using techniques like: 1) Maintaining a fixed-size sample of recent points, 2) Using a sliding window approach, 3) Implementing an incremental k-D tree that can efficiently insert and delete points, or 4) Using probabilistic data structures like Count-Min Sketch for approximate nearest neighbor search.

5. Compare and contrast k-NN with Support Vector Machines (SVM) for classification tasks.

Answer: k-NN is non-parametric and instance-based, making local decisions, while SVM is parametric and constructs a global optimal hyperplane. k-NN is simpler but can be computationally expensive for large datasets, whereas SVM is more complex but often performs better in high-dimensional spaces.

6. How would you implement a distributed k-NN algorithm using MapReduce?

Answer: In the Map phase, partition the data and compute partial k-NN lists for each partition. In the Reduce phase, merge the partial lists to find the global k nearest neighbors. Use techniques like parameter servers for sharing large datasets across nodes and implement efficient merging algorithms to minimize communication overhead.

Medium:

7. What are the main differences between k-D trees and Ball trees for nearest neighbor search?

Answer: k-D trees partition space using axis-aligned hyperplanes, while Ball trees use hyperspheres. Ball trees often perform better in high dimensions and with non-uniform data distributions, but k-D trees can be more efficient for low-dimensional data.

8. Describe how you would implement a k-NN regressor that provides uncertainty estimates for its predictions.

Answer: Implement a probabilistic k-NN regressor by: 1) Using the variance of the k nearest neighbors' target values as an uncertainty measure, 2) Applying bootstrap aggregating (bagging) to create multiple k-NN models and use their prediction variance, or 3) Implementing a Gaussian Process regression with a kernel based on k-NN distances.

9. How does the curse of dimensionality affect k-NN, and what techniques can mitigate its impact?

Answer: The curse of dimensionality makes distances less meaningful in high dimensions, reducing k-NN's effectiveness. Mitigation techniques include: dimensionality reduction (e.g., PCA, t-SNE), feature selection, using adaptive distance metrics, or switching to approximate nearest neighbor methods.

10. Explain how you would implement a multi-label k-NN classifier that can handle correlations between labels.

Answer: Implement a multi-label k-NN by: 1) Using a binary relevance approach with label-specific distance weighting, 2) Implementing a label powerset method to capture label correlations, 3) Using a label-dependent feature space where each feature vector is augmented with label information from its neighbors.

11. How would you handle missing data in a k-NN model without resorting to imputation?

Answer: Implement a custom distance metric that ignores missing values when computing distances between points. For categorical variables, treat missing values as a separate category. Alternatively, use a partial distance strategy where only available features contribute to the distance calculation.

12. Describe an approach to automatically adapt the value of k in k-NN based on local data density.

Answer: Implement an adaptive k-NN by: 1) Using a density estimator (e.g., kernel density estimation) to determine local data density, 2) Adjusting k inversely proportional to the local density, 3) Implementing a model that learns to predict the optimal k for each query point based on its local neighborhood characteristics.

13. How does k-NN perform in terms of runtime complexity for training and prediction? How does this compare to other common algorithms?

Answer: k-NN has O(1) training time (just storing the data) and O(nd) prediction time for brute force (n = dataset size, d = dimensions). This differs from algorithms like linear regression (O(nd^2) training, O(d) prediction) or decision trees (O(nd log n) training, O(log n) prediction).

Hard:

14. Explain how you would implement a k-NN classifier that can handle both numerical and categorical features, including high-cardinality categorical variables.

Answer: Implement a hybrid distance metric that combines numerical and categorical distances. For high-cardinality categorical variables, use techniques like feature hashing or learned embeddings to create a dense representation. Normalize distances across different feature types to ensure balanced contribution.

15. How would you apply k-NN to time series data for classification or forecasting tasks?

Answer: For time series, use distance measures like Dynamic Time Warping (DTW) or edit distance. Implement sliding window techniques to capture local patterns. For forecasting, use k-NN regression on lagged variables or implement a k-NN based local linear regression model.

16. Describe how you would implement a privacy-preserving k-NN algorithm that protects individual data points while still providing accurate predictions.

Answer: Implement differential privacy by adding controlled noise to distance calculations or k-NN outputs. Use techniques like secure multi-party computation or homomorphic encryption to perform k-NN computations on encrypted data. Alternatively, implement a model that learns a private representation of the data for k-NN computations.

17. How would you handle outliers in a k-NN model, both during training and prediction?

Answer: During training, use outlier detection methods (e.g., LOF, Isolation Forest) to identify and remove or downweight outliers. During prediction, implement a distance threshold to ignore neighbors beyond a certain distance, or use robust averaging techniques that are less sensitive to outliers.

18. Explain how you would implement a k-NN based anomaly detection system for high-dimensional, streaming data.

Answer: Implement an online k-NN anomaly detector by: 1) Maintaining a representative sample of normal points using reservoir sampling, 2) Using approximate nearest neighbor search (e.g., LSH) for efficiency, 3) Computing local outlier factors (LOF) in sliding windows, 4) Adapting distance thresholds based on data distribution changes over time.

19. How does the performance of k-NN compare to neural networks for image classification tasks? In what scenarios might k-NN be preferred?

Answer: Neural networks generally outperform k-NN for large-scale image classification due to their ability to learn hierarchical features. However, k-NN might be preferred for small datasets, interpretability needs, or when fast adaptation to new classes is required (e.g., few-shot learning scenarios).

20. Describe how you would implement a k-NN based recommender system that can handle both user-item interactions and content-based features, while scaling to large datasets.

Answer: Implement a hybrid k-NN recommender by: 1) Using collaborative filtering k-NN in the user-item interaction space, 2) Implementing content-based k-NN using item features, 3) Combining both approaches using a weighted scheme or learning to rank model. Scale to large datasets using techniques like LSH for approximate nearest neighbor search, model-based dimensionality reduction, or distributed computing frameworks.