**Interview Questions for Logistic Regression:**

**1. What are the key hyperparameters in KNN?  
Ans.**In the **K-Nearest Neighbors (KNN)** algorithm, several key hyperparameters can be tuned to optimize the model's performance. The most important ones are:

**1. Number of Neighbors (k)**

* **Description**: This is the number of nearest neighbors to consider when making a prediction. The algorithm looks at the kkk closest data points to the query point and makes a prediction based on the majority class (in classification) or the average value (in regression).
* **Impact**:
  + **Low k**: A small kkk (e.g., k=1k = 1k=1) makes the model more sensitive to noise and can lead to overfitting.
  + **High k**: A large kkk makes the model more robust to noise but might underfit the data, smoothing out nuances in the dataset.
* **Tuning**: kkk is usually tuned using cross-validation to find the optimal value that balances bias and variance.

**2. Distance Metric**

* **Description**: The distance metric determines how the distance between data points is calculated. The most common distance metrics are:
  + **Euclidean Distance**: ∑i=1n(xi−yi)2\sqrt{\sum\_{i=1}^n (x\_i - y\_i)^2}∑i=1n​(xi​−yi​)2​
  + **Manhattan Distance (L1 norm)**: ∑i=1n∣xi−yi∣\sum\_{i=1}^n |x\_i - y\_i|∑i=1n​∣xi​−yi​∣
  + **Minkowski Distance**: A generalization of Euclidean and Manhattan, controlled by a parameter ppp: (∑i=1n∣xi−yi∣p)1/p\left( \sum\_{i=1}^n |x\_i - y\_i|^p \right)^{1/p}(i=1∑n​∣xi​−yi​∣p)1/p
  + **Hamming Distance**: Used for categorical variables to calculate the number of positions at which the corresponding elements differ.
* **Impact**:
  + **Euclidean**: Sensitive to the scale of the data; often requires feature scaling.
  + **Manhattan**: More robust to outliers than Euclidean.
  + **Minkowski**: Allows customization with the parameter ppp to control the distance calculation.
* **Tuning**: The choice of distance metric can be tested using cross-validation to find the best fit for the specific data.

**3. Weighting of Neighbors**

* **Description**: In standard KNN, all neighbors contribute equally to the prediction. However, you can assign weights to the neighbors based on their distance to the query point:
  + **Uniform Weights**: All neighbors contribute equally.
  + **Distance Weights**: Closer neighbors have more influence on the prediction than farther ones.
* **Impact**:
  + **Uniform**: Simplifies the model but may not capture the nuances in data where closer points should have more influence.
  + **Distance-Weighted**: Can improve accuracy by giving more importance to nearer neighbors, especially in cases where closer neighbors are more likely to be similar to the query point.
* **Tuning**: Choosing between uniform and distance-weighted voting can be done using cross-validation.

**4. Algorithm for Nearest Neighbors Search**

* **Description**: Determines the algorithm used to compute the nearest neighbors. Common choices include:
  + **Brute Force**: Computes the distance between the query point and all points in the dataset. This is computationally expensive for large datasets.
  + **KD-Tree**: A space-partitioning data structure that can speed up the search process in low-dimensional data.
  + **Ball Tree**: Similar to KD-Tree but better suited for higher-dimensional data.
* **Impact**:
  + **Brute Force**: Simple but slow for large datasets.
  + **KD-Tree/Ball Tree**: Improves speed, especially for large datasets, but may not perform well in very high-dimensional spaces.
* **Tuning**: The choice of search algorithm is often based on the dataset size and dimensionality.

**5. Leaf Size (for KD-Tree or Ball Tree)**

* **Description**: In KD-Tree and Ball Tree algorithms, leaf size is a parameter that controls the number of points at which the algorithm switches from tree traversal to brute-force search within a leaf.
* **Impact**:
  + **Small Leaf Size**: Increases the depth of the tree, potentially improving precision but at the cost of higher computation time.
  + **Large Leaf Size**: Reduces tree depth, speeding up the search at the expense of precision.
* **Tuning**: Leaf size can be adjusted to find a balance between speed and accuracy.

**6. p Parameter (for Minkowski Distance)**

* **Description**: The ppp parameter in the Minkowski distance metric determines the type of distance:
  + p=1p = 1p=1: Equivalent to Manhattan distance.
  + p=2p = 2p=2: Equivalent to Euclidean distance.
  + Higher ppp values give more weight to larger differences in the feature values.
* **Impact**: Adjusting ppp allows customization of the distance metric to better match the characteristics of the data.
* **Tuning**: The value of ppp is typically selected based on the dataset characteristics and cross-validation performance.

### 2. What distance metrics can be used in KNN? Ans. 1. Euclidean Distance

* **Formula**: d(x,y)=∑i=1n(xi−yi)2d(x, y) = \sqrt{\sum\_{i=1}^n (x\_i - y\_i)^2}d(x,y)=i=1∑n​(xi​−yi​)2​
* **Description**:
  + Euclidean distance is the most commonly used distance metric. It calculates the straight-line (or "as-the-crow-flies") distance between two points in a multidimensional space.
* **Use Case**:
  + Best suited for continuous variables and situations where the data is in a similar scale across all dimensions.

**2. Manhattan Distance (L1 Distance or Taxicab Distance)**

* **Formula**: d(x,y)=∑i=1n∣xi−yi∣d(x, y) = \sum\_{i=1}^n |x\_i - y\_i|d(x,y)=i=1∑n​∣xi​−yi​∣
* **Description**:
  + Manhattan distance measures the distance between two points by summing the absolute differences of their coordinates. It is akin to navigating a grid-like path, such as streets in a city.
* **Use Case**:
  + Useful when dealing with data in which different dimensions might have different scales or when you want to measure the total variation across dimensions.

**3. Minkowski Distance**

* **Formula**: d(x,y)=(∑i=1n∣xi−yi∣p)1/pd(x, y) = \left( \sum\_{i=1}^n |x\_i - y\_i|^p \right)^{1/p}d(x,y)=(i=1∑n​∣xi​−yi​∣p)1/p
* **Description**:
  + Minkowski distance is a generalization of both Euclidean and Manhattan distances. The parameter ppp controls the type of distance:
    - p=1p = 1p=1: Manhattan distance.
    - p=2p = 2p=2: Euclidean distance.
    - p>2p > 2p>2: Gives more weight to dimensions with larger differences.
* **Use Case**:
  + Provides flexibility to adjust the distance metric based on the specific characteristics of the data.

**4. Chebyshev Distance (L∞ Distance)**

* **Formula**: d(x,y)=max⁡i=1n∣xi−yi∣d(x, y) = \max\_{i=1}^n |x\_i - y\_i|d(x,y)=i=1maxn​∣xi​−yi​∣
* **Description**:
  + Chebyshev distance measures the greatest absolute difference between any pair of dimensions. It represents the maximum movement needed to go from one point to another in any dimension.
* **Use Case**:
  + Useful in situations where the maximum difference in any dimension is of primary concern (e.g., in certain chessboard moves).

**5. Hamming Distance**

* **Formula**: d(x,y)=∑i=1n1(xi≠yi)d(x, y) = \sum\_{i=1}^n \mathbf{1}(x\_i \neq y\_i)d(x,y)=i=1∑n​1(xi​=yi​) where 1(xi≠yi)\mathbf{1}(x\_i \neq y\_i)1(xi​=yi​) is an indicator function that is 1 if xix\_ixi​ is not equal to yiy\_iyi​, and 0 otherwise.
* **Description**:
  + Hamming distance is used to measure the difference between two strings of equal length by counting the number of positions at which the corresponding elements differ.
* **Use Case**:
  + Particularly useful for categorical or binary data where the attributes are represented as strings or bit vectors.

**6. Mahalanobis Distance**

* **Formula**: d(x,y)=(x−y)TS−1(x−y)d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}d(x,y)=(x−y)TS−1(x−y)​ where S−1S^{-1}S−1 is the inverse of the covariance matrix of the data.
* **Description**:
  + Mahalanobis distance takes into account the correlations between variables and is scale-invariant, meaning it considers the distribution of the data. It is effective in identifying outliers and measuring the distance between points in a correlated dataset.
* **Use Case**:
  + Suitable for data with correlated features or when you want to account for different variances along different dimensions.

**7. Cosine Similarity (Cosine Distance)**

* **Formula**: Cosine Similarity(x,y)=x⋅y∥x∥∥y∥\text{Cosine Similarity}(x, y) = \frac{x \cdot y}{\|x\| \|y\|}Cosine Similarity(x,y)=∥x∥∥y∥x⋅y​ Cosine Distance=1−Cosine Similarity\text{Cosine Distance} = 1 - \text{Cosine Similarity}Cosine Distance=1−Cosine Similarity
* **Description**:
  + Cosine similarity measures the cosine of the angle between two vectors in a multidimensional space, capturing their orientation rather than magnitude.
* **Use Case**:
  + Often used in text mining and natural language processing (NLP) to compare document vectors or in cases where the magnitude of the data points is not as important as their direction.

**8. Jaccard Distance**

* **Formula**: Jaccard Distance=1−∣X∩Y∣∣X∪Y∣\text{Jaccard Distance} = 1 - \frac{|X \cap Y|}{|X \cup Y|}Jaccard Distance=1−∣X∪Y∣∣X∩Y∣​
* **Description**:
  + Jaccard distance measures the dissimilarity between two sets by comparing their shared elements relative to their combined total. It is particularly used for binary or categorical data.
* **Use Case**:
  + Common in cases where the data is represented as sets, such as in document comparison or clustering.