Points covered for K-Nearest Neighbor.

1. Definition of algorithm
2. Aim/Goal
3. Objective
4. Advantages
5. Disadvantages
6. Performance Matrix
7. Regularization/Optimization

1. Definition:

- It is a supervised learning classification technique by which we are able to classify a new observation into a specific category.
- It uses the distance or similarity measure in order to classify data. It uses the Euclidean distance for classifying a particular category.
- It is also called a lazy algorithm, since it cannot learn from itself.
- If we use it for **Regression** it computes the **mean value**. (Avg. is Criterion)
- If we use it for **Classification** it computes the **mode value**. (Frequency is Criterion)
- If we have Even number of categories then the value of K should be Odd & Vice Versa.
- It is a **linear Model**.

2. Aim/Goal:

Its aim is to locate **all of the closest neighbors around a new unknown data point** in order **to figure out what class it belongs to. It's a distance-based approach**

For new observation, it will calculate the distance of new observation from each and every observation present in data. Then depending upon the value of K it will classify the new observation.

For Ex: If the value of K is given as 3 so it will select the nearest 3 data points, now if these 3 nearest data points belong to “X” category then the new observation will belong to category “X”.

If the value of K is 5 so it will select 5 nearest data points, now if these nearest data points have 2 data points from category “O” &

3 data points from category “X”

So by majority, the new observation will belong to category “X” again

If the value of K is 7 so it will select 7 nearest data points, now if these nearest data points have
4 data points from category “O” &
3 data points from category “X”
So by majority, the new observation will belong to category “O”.

3. Objective:

- How likely a data point is to be a member of one group or another depending on what group data points are nearest to it.

4. Advantage:

- Easy Implementation
- Gives High Accuracy

5. Disadvantage:

- It is Computationally Expensive.
- Does not work well with large datasets as calculating distances between each data point would be very costly.
- Sensitive to Missing Data.
- Does not work well with high **dimensionality** as this will complicate the distance calculation process to calculate distance for each dimension.
- Gives Over fitted results.

6. Performance Matrix:

- **Confusion Matrix**

  It is the squared matrix or table that is used to define the performance of a classification algorithm. It is also called as **mother of the accuracy & classification report**. By using the confusion matrix you can find both Error and Accuracy.

TN	**FN** Type-II Error More Dangerous
FP Type-I Error Less Dangerous	TP

TN⇒ Talks about Accuracy

FP⇒ Talks about Error Type-I Error Less Dangerous

FN⇒ Talks about Error Type-II Error More Dangerous

TP⇒ Talks about Accuracy

All **True values defines correctness**

All Talks about Accuracy

We should try to **reduce Type-II error (FN)**

- **Classification Report**

Accuracy Score ➔ $(TP + TN) \div (TP + TN + FP + FN)$

It tells how often your model was correct.

Precision ➔ $TP \div (TP + FP)$

It is known as a **Positive Predictive Value.** It is a measure of the amount of accurate positives your model claims compared to the number of positives actually.

Recall/Sensitivity⇒ $TP \div (TP + FN)$

It is known as a **True Positive Rate.** It is the measure of the amount of positives your model claims compared to the actual number of positives present throughout data. It gives **Actual Positives.**

Specificity⇒ $TN \div (TN + FP)$

It tells the proportion of true negatives that are correctly predicted by the model.

Error⇒ 1-Accuracy

F1 Score⇒ $2 \times (Precision \times Recall) \div (Precision + Recall)$

It talks about harmonic values of Precision & Recall.F1 Score will be good only when your Precision & Recall are good.

In **Regression** we focus on **error** but in **Classification** we focus on **Recall & F1 Score**

F1 Score is more important than Accuracy in case of classification.

7. Regularization/Optimization

It is the technique which is used for optimizing the model performance by adding some penalty terms in your error function. This is also called as penalizing.

sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5*, *, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *metric_params=None*, *n_jobs=None*)

We can do regularization by changing the value of **n_neighbors (int, default=5)**

- **Number of neighbors to use by default for kneighbors queries.**

- **weights{‘uniform’, ‘distance’} or callable, default=’uniform’**

  1. **‘uniform’** : uniform weights. All points in each neighborhood are weighted equally.
  2. **‘distance’** : weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

  3. **[callable]** : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

- You can also do Hyper parameter Tuning by using for loop/Grid Search CV for finding optimized parameters.