**Machine Learning Case Study**

M. Ruthvik Reddy

CB.SC.P2AIE25013

**K-Nearest Neighbors**

K-Nearest Neighbors (KNN) is a non-parametric, supervised learning algorithm that predicts a label or value for a new point by looking at the labels/values of its k closest training examples and aggregating them (majority vote for classification, mean for regression)

**How it works:**

* Store the labeled training data; there is no explicit model fitted, which is why KNN is often called a lazy, instance-based learner.
* Define a distance metric (Euclidean Distance, Manhattan Distance, or Minkowski distance) and, for a query point, find its k nearest neighbors under that metric.
* For classification, assign the class by plurality/majority vote of those neighbors; for regression, predict the average of their target values.

**Case Study**

**Dataset and Processing:** In this case study for KNN Cancer Dataset was used. And it contains 32 different features, namely:

| ['id', 'diagnosis', 'radius\_mean', 'texture\_mean', 'perimeter\_mean', 'area\_mean', 'smoothness\_mean', 'compactness\_mean', 'concavity\_mean', 'concave points\_mean', 'symmetry\_mean','fractal\_dimension\_mean', 'radius\_se', 'texture\_se', 'perimeter\_se', 'area\_se', 'smoothness\_se', 'compactness\_se', 'concavity\_se', 'concave points\_se', 'symmetry\_se','fractal\_dimension\_se', 'radius\_worst', 'texture\_worst','perimeter\_worst', 'area\_worst', 'smoothness\_worst', 'compactness\_worst', 'concavity\_worst', 'concave points\_worst','symmetry\_worst', 'fractal\_dimension\_worst', 'Unnamed: 32'] |
| --- |

* From these features 'diagnosis' is considered to be dependent feature, where it is to be found.
* And there are two features 'id' and 'Unnamed: 32' which are unnecessary, as these features do not provide any value to the predictions.
* After removing unnecessary features, below are the columns present:

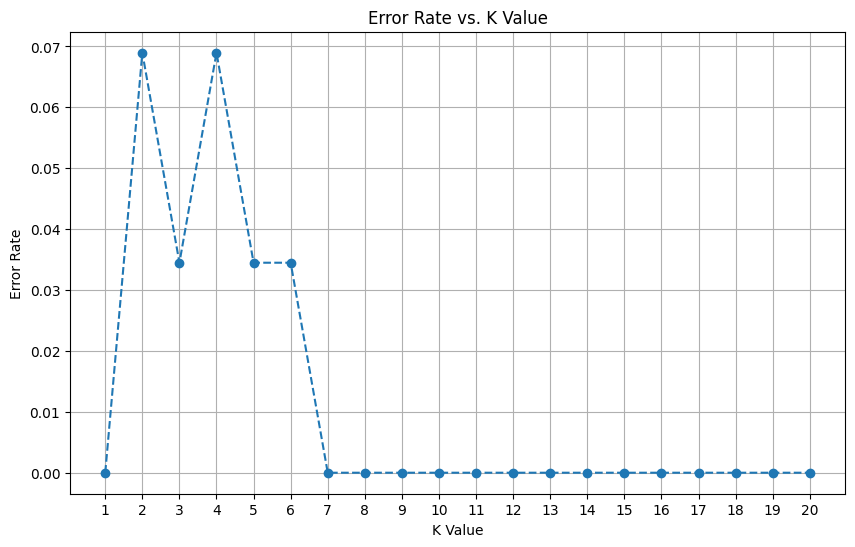
| ['diagnosis', 'radius\_mean', 'texture\_mean', 'perimeter\_mean', 'area\_mean', 'smoothness\_mean', 'compactness\_mean', 'concavity\_mean', 'concave points\_mean', 'symmetry\_mean','fractal\_dimension\_mean', 'radius\_se', 'texture\_se', 'perimeter\_se', 'area\_se', 'smoothness\_se', 'compactness\_se', 'concavity\_se', 'concave points\_se', 'symmetry\_se','fractal\_dimension\_se', 'radius\_worst', 'texture\_worst','perimeter\_worst', 'area\_worst', 'smoothness\_worst', 'compactness\_worst', 'concavity\_worst', 'concave points\_worst','symmetry\_worst', 'fractal\_dimension\_worst'] |
| --- |

* Now Considering y as 'diagnosis' and other features as X, then split them into X\_train, X\_val, y\_train, y\_val, in this 95% of data was considered for training and other 5% was considered for validation (or testing).

**KNN**

To find the best possible K value

| **error\_rates = [] k\_values = range(1, 21) # Test k from 1 to 20  for k in k\_values:  knn = KNeighborsClassifier(n\_neighbors=k, metric="euclidean")  knn.fit(X\_train, y\_train)  y\_pred = knn.predict(X\_val)  error\_rates.append(1 - accuracy\_score(y\_val, y\_pred))  min\_error = min(error\_rates) optimal\_k = k\_values[error\_rates.index(min\_error)] print(f"Optimal K: {optimal\_k} with Minimum Error Rate: {min\_error:.4f}")** |
| --- |



* And k=1 was giving minimum error rate, but k=7 is considered as it also has minimum error rate and to consider multiple neighbors.

For the case study, the data was tested with KNN with three different distance metrics: Euclidean Distance, Manhattan Distance, Cosine Distance. And here are the classification reports for these different distances:

| Classification Report (knn\_euclidean):   precision recall f1-score support   0 1.00 1.00 1.00 6  1 1.00 1.00 1.00 23   accuracy 1.00 29  macro avg 1.00 1.00 1.00 29 weighted avg 1.00 1.00 1.00 29 |
| --- |

| Classification Report (knn\_manhattan):   precision recall f1-score support   0 1.00 1.00 1.00 6  1 1.00 1.00 1.00 23   accuracy 1.00 29  macro avg 1.00 1.00 1.00 29 weighted avg 1.00 1.00 1.00 29 |
| --- |

| Classification Report (knn\_cosine):   precision recall f1-score support   0 1.00 1.00 1.00 6  1 1.00 1.00 1.00 23   accuracy 1.00 29  macro avg 1.00 1.00 1.00 29 weighted avg 1.00 1.00 1.00 29 |
| --- |

**Conclusion:** So, for every distance metric used inferred the same outcome, and every KNN model predicted correct outcomes with 100% accuracy (based on the classification reports). And the best K value was found with iterating over with all possible K values, which provided the best K value with the least error rate.