Post Session Document - Introduction to Supervised Learning Classification

In the real world, there are several classes of tasks. One of the most important among them is the **classification** type of task. It is **parallelly** important to regression tasks and perhaps more frequent than regression. In a classification task, the input features might be **continuous** or **categorical** or a **mix** of them but the output feature is a **discrete** one. It has a **finite** number of values that it can possess. In supervised learning, there are many algorithms that are used to solve such problems.

Let us take some examples of classification problems -

1. Does a patient have cancer or not? -
   Here the outcome has two possibilities. One the patient has cancer and the other he does not. Such problems are called Binary classification problems.

2. Is email spam or not? -
   An email might be spam or non-spam. So this is again a two-class or binary classification problem.

3. Digit recognition: which digit does the image contain?
   Given an image of a digit, the target is to identify which digit it is among the listed 10. Here the number of possible outcomes is 10. So this is a multiclass classification problem.

4. Which image is a cat and which is a dog?
   Given an image of a cat or dog, the target is to identify whether it is a cat or a dog. This is a binary classification problem.

Based on the number of values the output feature classification problems can be classified as follows -

1. **Binary** classification
   In a binary classification problem, the number of outcomes is **two** only. For example (yes, no), (True, False), (Pass, Fail), etc.

2. **Non-binary** classification
   It is also called a multiclass classification problem. Here the number of outcomes is more than two. For example - Identifying the digit in the image, which is a 10 class problem.

To understand the binary classification problem let us take an example of a loan default case in a bank. In any bank that provides loans to the customers, in general, the customer pays back the loan but in very few cases it also happens that the customer is unable to repay the loan. There might be a genuine personal reason or some **illegal** reason associated with that also. To

deal with this the bank does a certain type of prediction on a new customer whether he will default the loan or not. The outcome “default” means the customer will not return back the loan amount and the outcome “no default” means the customer will repay the loan amount. Hence this is a binary classification problem. Doing so will save banks from either opportunity loss or resource loss. Let us go into some more details -

Example: Prediction of loan default

Let us consider the below inputs for the sample binary classification problem. The inputs are as follows -

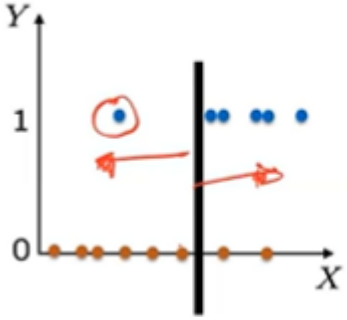
1. **Student** - Whether the customer is a student or not.
2. **Default** - This is the target feature that indicates whether the borrower will default or not.
3. **Balance** - What is the bank balance of the customer at the time of borrowing the loan.
4. **Income** - The income of the borrower.

Based on these inputs the target features “Default” has to be predicted by training a classification model. It is not advisable to consider the economic breakdown conditions because that will unnecessarily add more complexity to the model and hence it will become much less interpretable.

So this is how a classification problem is formulated. Now it is important to see, how does the classification work? How do the data points separate? How to determine the line or curve that will decide the best segregation of data points? What criterion is taken under consideration while creating such lines or curves?

Predictors and classifiers

In the case of linear regression, a predictor is a line. For new data points, the linear equation took the inputs and gave the output. The predictor is determined by using the mean squared error. However in the case of a classification problem the predictor is a **model** that will predict the class of the record. The metric used here is the probability of a mistake committed by the model. The model should be prepared in such a way that the number of **misclassifications** are kept to a minimum.



Here in the above figure, any point lying to the right of the black bar is classified as blue and to the left is classified as red. The misclassification can be seen here easily. On the right of the black bar, the red dot is visible where only blue dots should be. While on the left of the black bar, blue dots are visible when only red dots should be there. The black bar is the predictor curve that will decide the output label of the new record. It should be selected in such a way that minimal misclassification error occurs while training the model.

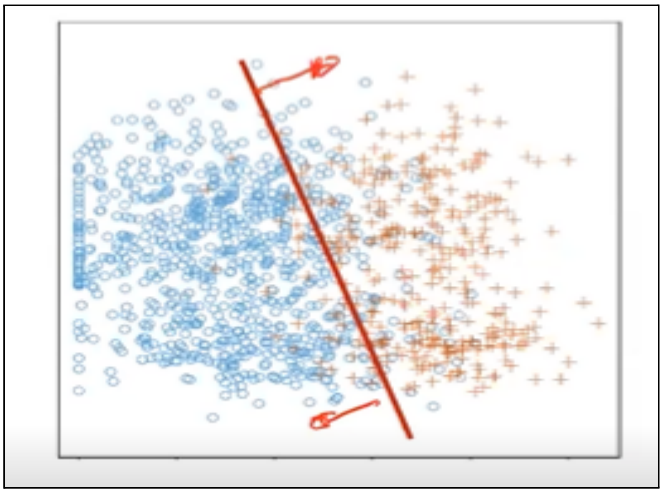
A predictor can be visually of many types depending on which shape gives the minimum misclassification error by allowing the model to not overfit or underfit. It might be a **linear** one, a **curve** shape one. In curve shape also it might be of many types. Let us take below some of the possible shapes of the predictor curve and the corresponding classification models.

Type of classifiers

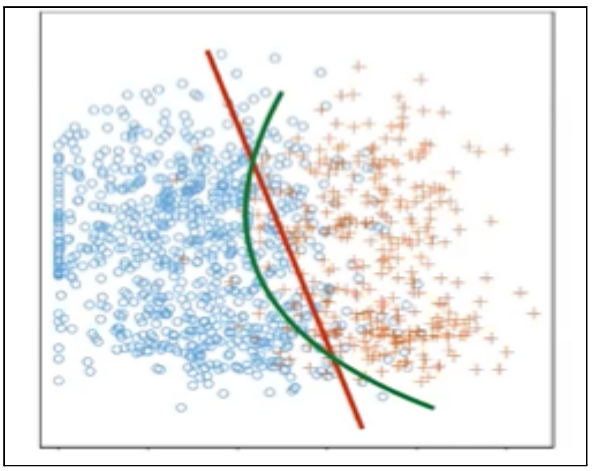
Below are some types of classifiers given in detail.

1. **Linear classifier** - In two-dimensional feature space, this is a linear equation or a straight line that will create the distinction between the available classes. In the below figure it can be seen that there are red dots and blue dots present. The line is situated at a place in such a way that most of the red dots are on the right of the line and most of the blue dots are on the left of the line. Still, on the right, there are blue dots and on the left, there are red dots. These dots are called **misclassifications** or the error made by the model. While setting the line of separation this misclassification error has to be minimum so that the model can make trustworthy predictions on the unseen data.

   In higher dimensions, this won't be a line but a plane that will segregate the data points in multiple classes.



2. **Non-linear classifiers** - It is a slightly different type of classifier where the separator is a non-linear curve. It is not a straight line. It might be a polynomial. In the below figure it can be seen that the green curve represents a **nonlinear** classifier. It is clear from the figure that the non-linear separator performs better than the linear one. The misclassification rate is less in the case of the nonlinear curve while in the case of the linear one it is a bit high.



Corresponding to any real-life case, which classifier is used depends on the problem. The one that is performing better in a certain case should be used. In some cases, the linear classifier is enough while in some other cases polynomials of different degrees can also fit best.

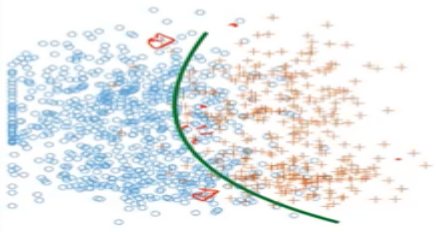
For any machine learning model, it is customary to make errors while making any sort of prediction. No model is ideal and 100% accurate over unseen data in general. In the case of classification, the error made by the classifier can be majorly understood as follows -

Error types and confusion matrix

Let us understand this with the below figure. Here it is a binary classifier with two colors of dots namely red and blue. The green curve is a classifier that is creating the separation between the two types of points. The errors made in this can be listed below -

- If the truth is blue and the model predicts it to be red.
- If the truth is red and the model predicts it to be blue.

These two errors are clearly visible in the below figure. On the right of the curve instead of only red points, a few blue points are there, and on the left of the curve instead of only blue points, a few red points are also there.



The ultimate objective of the classification is to commit as few errors as possible on the new examples. While making the prediction if the classifier is nudged, the classifier line or curve will find that the count of **misclassifications** or **errors** is changing. So there used to be a tradeoff between the two types of errors on nudging the classifier line. In such a case empirical risk minimization is done.

In classification problems to represent the frequency of errors of both types and also the correct predictions, a matrix is used that is named as a confusion matrix. It is a tabular representation of the frequency of different types of combinations of outputs of predicted and actual outcomes.

To solve the classification problem and inherently to get the classifier below are the few approaches used.

(Gaussian) Model-based approach.

In this approach, the data is passed to a **probabilistic model**. For every possible outcome class, the model predicts the probability of occurrence. Then on the test set of data the model is deployed to make the prediction. On the predicted probability a certain **threshold** is implied to create the outcome label. For example, in the given set of predicted probability if the threshold probability is 0.6 then if the predicted value is greater than 0.6 the outcome will be 1 (say) and if it is less than 0.6 the outcome will be 0 (say).

At a deeper level let us understand this with an example. Suppose there is a binary classification of people being good and bad at a certain skill. Using a good amount of data the classifier is prepared. For a good person, there will be a certain set of records supporting that and correspondingly a distribution associated with that. So is the case with the bad person. When a new data point comes, it is found to which distribution it belongs to and accordingly the final output is given to that new record. ie. if a certain new record belongs to the distribution of good people then the corresponding outcome for that person will be that he is a good person in that skill.

In the Gaussian model, there is a probability associated with every class. With this probability, a unique input vector is also associated with a normal distribution. The mathematical expression of probability can be given as follows -

$$P(Y = k) = \pi_k (k = 1, 2, 3, ... m)$$

It is the probability of the outcome being k.

$$P(X|Y = k) = \gamma_k \, exp\{- \frac{(X-\mu_k)^2}{2\sigma_k^2} \}$$

For multivariate problems, it is given as follows -

$$P(X|Y = k) = \gamma_k exp\{\frac{-1}{2}(X - \mu_k)^T C_k^{-1}(X - \mu_k)\}$$

Where μ_k: $mean\ vector\ E[X/Y = k]$

And C_k: $covariance\ matrix$: $E[(X - \mu_k)(X - \mu_k)^T|Y = k]$

Where μ_k is the mean vector and C_k is the covariance matrix. For multivariate problems, the shape of the normal distribution is a bell-shaped curve. The curve is multidimensional and at constant heights contours are available. **Contours** are curves at a constant height from the bottom while the mean vector represents the center of the curve. The shape of the curve is defined by the **covariance** of the variable along the mean vector. The contours are **ellipses** and their orientation is defined by the covariance matrix. For different data points, different contours can be found.

Bayes rule

Bayes rule is the central rule in the theory of probability. A huge number of applications are associated with the Bayes theorem. It gives the conditional probability that goes the other way. It converts the prior probability to posterior probability. The prior probability is defined as: "given a certain outcome y, what is the probability that it is derived by the certain input x". It can be given as follows.

$$Prior\ probability = P(Y|X)$$

While the posterior probability is: "given a certain x what is the probability that it belongs to class label y". It can be given as follows -

$$Posterior\ probability = P(X|Y)$$

The prior probability is a constant number. For example: when a person is born how likely is he to be a good or a bad person. This is a prior probability.

These two are associated with the following expression.

$$P(Y = k|X) = \frac{\pi_k . P(X|Y=k)}{P(X)}$$

Given the input vector x, find k for the maximum posterior probability. The above model is supposed to minimize the error. For each one of the classes k, the error term is given as follows -

$$P(error) = \pi_k \gamma_k \, exp\{\frac{-1}{2} (X - \mu_k)^T C_k^{-1} (X - \mu_k)\}$$

As the above expression seems to be an exponential one it is preferable to compare the logarithmic term of the above equation. It is found that the logarithmic equation is a quadratic equation in x, the input features. The boundary between choosing one class and the other is a convex shape or a quadratic equation. This is called **quadratic discriminant analysis or QDA**.

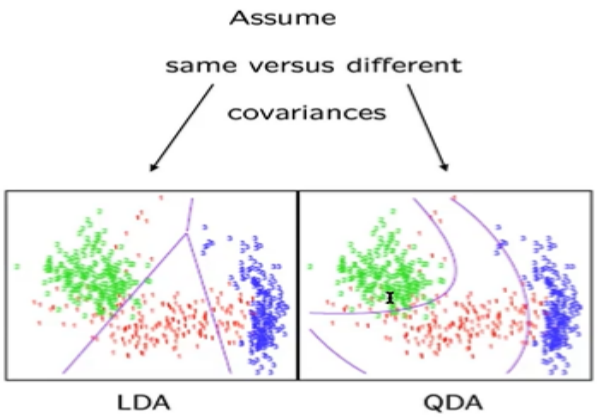
When all the covariance matrices are not the same it is a quadratic function but when they are the same it becomes a linear discriminant analysis.

LDA versus QDA

In the case where classes are **linearly separable,** then LDA is used while when the QDA is more useful it is used. It is quite important to understand when to use LDA and when to use QDA.

For two populations corresponding to different classes, if means are different and covariances are the same, LDA is applied and it gives us a linear classifier.

For two populations corresponding to two different classes, if means and covariances are different then the classifier is an **ellipse** or a quadratic one. Hence QDA is applied in such a case. In the below figure it is clear that LDA creates a linear separation curve while QDA creates a quadratic applied in their corresponding circumstances.



Here it is well demonstrated that if means are the same and covariances are also the same then LDA is applied while if they are different then QDA is applied here.

As in the actual data of the loan default example, only 3.33% are misclassified. So if all the data points are labeled as **"no default"** then the misclassification rate will be automatically low. The accuracy will be very high and it will be around 97%. This seems a pretty high accuracy, but it is not useful to get such accuracy. The misclassification rates of different methods are as follows -

Making predictions using a single feature of input data.

LDA : 2.81 % (Misclassification rate)
QDA : 2.73 %.

Make prediction using all of x :

LDA : 2.75 %
QDA : 2.70%

When machine learning models commit errors, they do not make any distinction between different kinds of errors. But the business and economic aspects of the problem make the difference between the errors. Committing one error might cost more to a firm than the other.

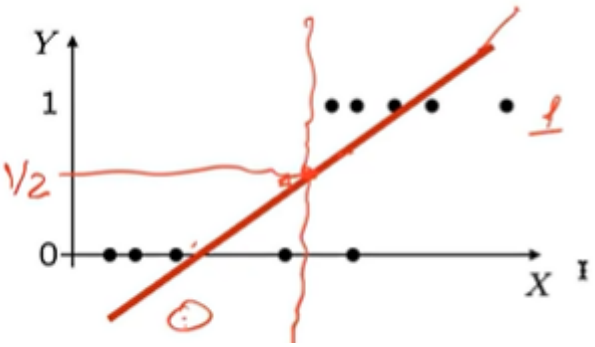
Unbalanced cost

An imbalanced dataset is one where the proportion of each output class in the dataset is not the same. For a binary classification problem if each class output is not equal to 50% or the difference in the proportion is high then it is an imbalanced data set. While doing classification on such cases the committed errors cost differently. The cost of committing one kind of error is different from the cost of committing other errors. In the case of the loan default example, if we are predicting a real defaulter as a "non-defaulter" then it is a **loss of resource**. This is because when we predict him to be a non-defaulter then he is **eligible** for the **loan** and the bank will pay him the loan. But that customer is not going to return that money. Hence it is a monetary loss for the bank and so it is a loss of resources. On the other hand, if the model is predicting a real "non-defaulter" as a "defaulter", then that is an **opportunity loss**. This is because in predicting "defaulter" the bank is going to decide that no loan amount will be given to that customer. But in reality, he was going to repay the loan. Due to this, the customer will not take the loan and the bank will lose a loyal customer who would have given some financial benefit in terms of interest on the loan. Hence it is an opportunity loss. The bank has lost the opportunity to provide a loan to that customer.

Now let us get into the models deployed to solve the classification problems in machine learning. Let us begin by simply applying the linear regression model to the classification task. As

we know that this model is a straight-line model. Due to the straight-line shape, the model is not going to capture the existing patterns in the data. It is more prone to underfitting the data. Hence a straight linear regression model is not preferred for any sort of classification task. So this is an approach that does not work in the case of classification problems.

From the below diagram it can be seen that the linear regression model is unable to capture the outcome classes of the data.



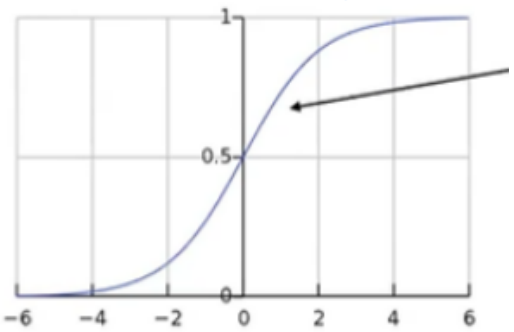
Due to this a new model is deployed in such binary classification problems. It is named the Logistic regression model. This model is stated below -

The logistic model

The logistic regression model is associated with the same linear input as it is with the linear regression model. The only difference is that input is passed to a transformation function. Namely the **sigmoid** function. This function is mathematically given as follows -
For a single input feature x, it is given as follows -

$$y = \frac{e^x}{e^x + 1}$$

The corresponding plot of the function is given as follows -



If the predicted probability y is greater than 0.5 then the output is 1 else it is 0. In the case of multiple features, the equation of the sigmoid function will be a bit modified. It can be given as follows -

$$y = \frac{e^{\theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n}}{e^{(\theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n)} + 1}$$

Here all the θ are the associated weights to the corresponding features. Based on different values of θ the shape of the sigmoid function also varies in terms of the flexibility of the curve. The shape of the curve matches with the English letter **S** (Stands for summation) hence the name sigmoid.

For higher dimensions, it considers multiple variables with multiple parameters. It is an exponential of the linear function of the input features x. Here the parameters are chosen by using some cost function.

Training the logistic model: Maximum likelihood

Now we have understood the concept of the hypothesis function that is the sigmoid function in the case of logistic regression. To establish the relation between the output and the input features it is crucial to find the weight parameters of the model. To find the weights the likelihood function is something that comes into the picture. For the entire training example, the likelihood function can be defined as the product of the probability of occurrence of the given outcome provided the input feature data. In a more general term if the records of the training example are presented as (x1,1), (x2,0), ……. (Xn,0). Then the expression of likelihood can be given as follows -

$$Likelihood = \prod_{i=1}^{n} P(Y_i \mid X_i)$$

The likelihood function is a function of parameters and it is the product of the probability of all the training examples.

Taking the logarithm will make the equation simple and linear -

$$log(Likelihood) = \sum_{i=1}^{n} P(Y_i \mid X_i)$$

This might look like a messy equation but it is easy to compute gradients on the logarithmic equation. Also, it becomes simple to do any sort of computations.

Let us gather the results found till now by applying different models.

Results

If “no default” is always predicted, then the misclassification rate is only 3.33%.
If the prediction is made on only one component of X.

LDA: 2.81%
QDA: 2.73%
Logistic regression: 2.73%

When the prediction is made using all of X.

LDA: 2.75%
QDA: 2.70%
Logistic regression : 3.33%.
Logistic regression with absolute value regularization: 2.66%.

Unbalanced data sets and costs

We are now aware of the fact that in the case of unbalanced data, directly computing the accuracy does not work. The cost of making different errors is different from the business perspective. In such cases, precision and recall will work better. To handle such a situation the cost function can be added with some extra weights. If the error term is more then add more penalty or weight term to that error and if it is less then penalize it less. Mathematically this can be given as follows :

$$log\ L(data; \theta) = \sum_{i=1}^{n} P(Y = 0 \mid X) + w \sum_{i=1}^{n} P(Y = 1 \mid X)$$

The additional weight w can be updated and it should be greater than 1. It is updated until we get a satisfying accuracy. w is a hyperparameter.

Till now a few of the algorithms used to solve classification problems are described here. Another necessary kind of algorithm that works nicely in case of solving such problems is the **Nearest neighbors algorithm**. In nearest neighbors algorithms, there are no weights or parameters associated with the model. Instead, it seeks the closest neighbors for any target point in the given dataset. The output class that is in majority in points in the vicinity will become the output of the target point.

One very common question that arises while deploying the nearest neighbors algorithm is how many nearest points to consider to find the output class of the test data point. To understand this let us get into the algorithm named the **k-NN classifier**.

The k-NN classifier

One such model is **k nearest neighbors** (k-NN algorithm), where k is the count of neighbors to consider. This model is not associated with any type of **weight** and parameters. So given a new record x, find the k closest points in the dataset. This is calculated by measuring the distance between points. To find the output label, the majority of the output of the k selected points will be taken.

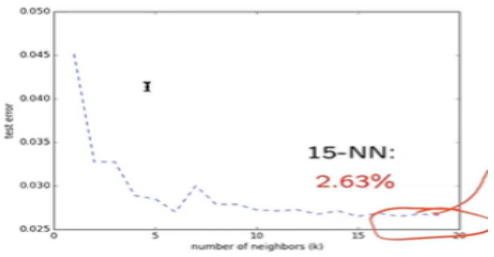
There is no hard and fast rule to understand a good value of k. It is more of an experimental aspect. One needs to try out different values of k.

As k increases the accuracy increases because the model captures more of the existing patterns of the data. Considering the loan default case in the case of very high k the majority is **"no default"**. So hundred percent accuracy will be there on the training set. In conclusion, very low and very high values of k, are both not good.

Selecting k

One of the generic processes of selecting k is to use the validation set approach. Set aside the 20% (for example) of data in the validation set and experiment with different values of k. For every model just check the performance on the validation set. The model with a certain value of k that is performing best should be the ideal value.

The k-NN method is not suitable for **image** tasks because the distances between two images will be very high and it is a computationally challenging task to accomplish. Also finding meaningful features in the case of images is not a simple task. k-NN is useful when the input features are a bit meaningful.



The above figure shows a plot of test error with different values of k nearest neighbors. As it can be seen, for very small k the error is very high because it will be an underfit model which will rely on the values of a very small number of data points. When the value of k is high then the test error is very low. This is also not a suitable choice because of overfitting chances. So it is advisable to get a value of k somewhere in between these two extremes. Such a value of k will serve better in terms of accuracy and generalization on unseen data.

Effect of Scale

As is a bit clear, the k-NN algorithm is dependent on distance computations to find the nearest points. Distance between any two points is totally dependent on the coordinate values of the points associated. Changing the **unit and scale** of one point will make a significant difference in the calculated distance. This scale has a significant effect on the working and performance of the k-NN algorithm.

In the below figure it can be seen that by changing the scale of the input data the set of nearest points changes significantly. The target is to find the nearest neighbors for the red point in the middle. In the first part that is without any change in scale, the closest points are the two blue dots. And hence the outcome will be that associated with these blue points.

When a change in the scale is made now the closest points are the brown ones. Hence the output will be “brown” for the target point.

So scale plays an important role in identifying the outcome of a data point while using k-NN algorithms.

There are some other possibilities of algorithms that work in the case of classification -

1. Weighted NN
2. Kernel regression or local regression

Let us get into them one by one -

1. **Weighted nearest neighbor (Weighted NN)**

   Similar to the nearest neighbors algorithms, one of the possibilities is weighted nearest neighbors. In the k-NN algorithm majority vote between k neighbors is considered.

The points can be associated with weights depending on the corresponding distance with the target point. It is obvious that less weight will be given to the points that are far away and more weight will be given to the points that are close to each other. In such an approach every point can vote in determining the output label.

2. **Kernel regression**

   A kernel is a fancy name for picking weights. Here we create a regression line with the points in the vicinity of the target point. Using that only we make predictions for the corresponding point. This is called either local regression or kernel regression.

   If the entire population for regression is used, there is a chance that the variance will be less and this method is suitable for some cases. But if it is needed to use similar data points to make better predictions then it is better to go with the kernel regression.

   Let us take an example of three villages where covid patients are there. Now for a certain patient belonging to the first village to predict whether he is suffering from covid or not it is advisable to use patient data of that corresponding village. In such a case it is not useful to use the patient data of all three villages. This is the use of kernel regression.

Along with these algorithms, there are a certain set of algorithms that are useful in classification. They are listed below -

1. **Support vector machine** - This is an algorithm that works on the basis of the margin of a point from the specified hyperplane. The more the margin the better the quality of the classifier.
2. **Decision trees** - It is a rule-based split of data to get the most homogeneous split of data.
3. **Neural networks**

Interpretability of results

Commonly for regression or classification algorithms, the associated hyperparameters give some interpretation about the model and the prediction criterion.

In nearest neighbors, there are no coefficients associated. So they are less interpretable. In general, more complex machine learning algorithms are less interpretable. For example, deep learning models are very less interpretable or can be said to be non-interpretable.