Machine Learning

LVC 3: 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** task. It is parallelly important to regression tasks and perhaps more frequent. In a classification task, the input features might be **continuous** or **categorical** or a **mix** of them, but the output feature is categorical. In supervised learning, there are many algorithms used to solve classification problems.

Let's see some examples of classification problems:

1. **Does a patient have cancer or not?**
   Here, the output feature has two classes. First, the patient has cancer, and second, the patient does not have cancer.

2. **Is email spam or not?**
   Again, we have two classes as an email would either be spam or non-spam.

3. **Digit recognition: which digit does the image display?**
   Given an image of a digit, the target is to identify which digit it is among the 10 digits (0 to 9). Here, the number of classes is 10.

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

Based on the number of distinct classes in the output feature, classification problems can be categorized 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:
   If the number of classes in the output features is more than two, it is called a non-binary or **multi-class** classification problem. For example, identifying the digit in an image is a 10-class problem.

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

Example: Prediction of loan default

To understand the binary classification problem, let us take an example of bank loan default. 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 unwarranted reason associated with that. To deal with this, the bank does a certain type of prediction on a new customer to know whether the person will default or not. The outcome “default” means the customer will not repay the loan and the outcome “no default” means the customer will repay the loan. Hence, this is a binary classification problem. Doing so will save the bank from monetary and resource losses. Let us go into some more details.

There are 10,000 observations in our example and the features are as follows:

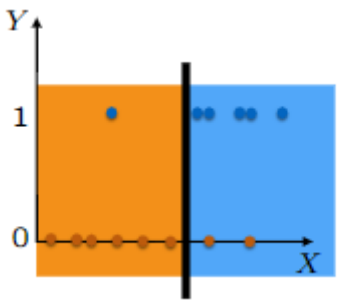
1. **Student**: Categorical feature indicating whether the customer is a student or not.
2. **Default**: This is the target feature that indicates whether the borrower will default or not.
3. **Balance**: The bank balance of the customer at the time of borrowing the loan.
4. **Income**: The income of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

Based on three inputs features, the target feature “default” has to be predicted by training a classification model. It is not advisable to consider the economic breakdown conditions because that will unnecessarily make the model more complex and hence less interpretable.

Now, it is important to see, how does the classification work? How are the data points separated into different classes? 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 case of linear regression, the 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 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 minimum**.



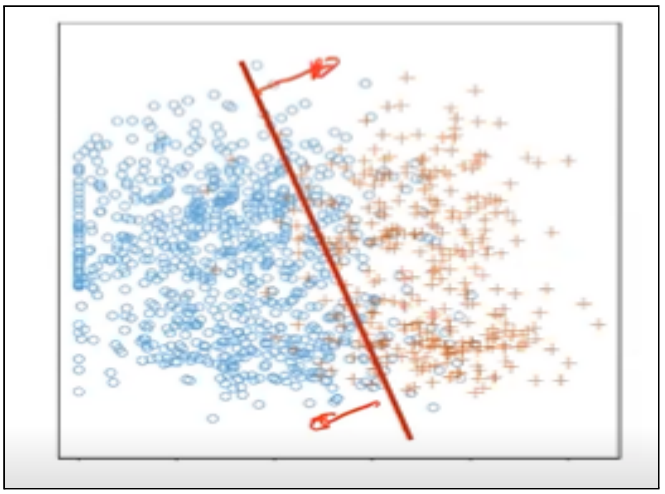
This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

Here in the above figure, any point lying to the right of the black vertical bar is classified as blue and any point to the left is classified as orange. The misclassifications can be observed here easily. On the right of the black bar, two orange points are visible while only blue dots should be there. Similarly, on the left of the black bar, one blue dot is visible while only orange dots should be there. The black bar is the predictor curve that will decide the output label of the new record. It should be estimated in such a way that minimal misclassifications occur while training the model.

A predictor can be of many types depending on which shape of the curve gives the minimum misclassification error by allowing the model to not overfit or underfit. Let's discuss some of the possible shapes of the curve and the corresponding classification models.

Type of classifiers

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.



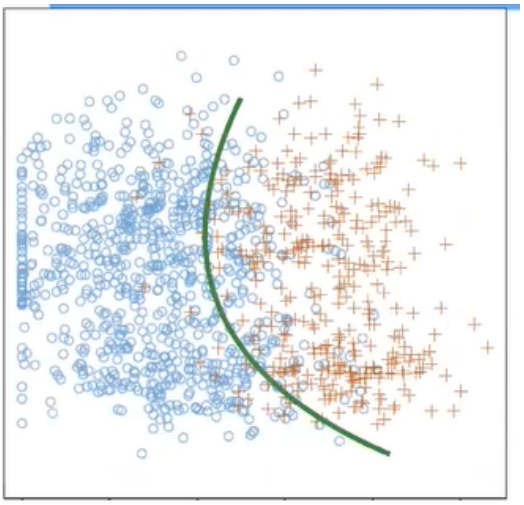
In the above figure, it can be seen that there are blue dots and red dots present. The line is situated at a place in such a way that most of the red dots are on the right side of the line and most of the blue dots are on the left side of the line. Still, on the right, there are some blue dots and on the left, there are some 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, the line would generalize to a plane that will segregate the classes.

2. **Non-linear classifier**: 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 observed that the green curve represents a **non-linear** classifier. It can also be observed from the figure that the non-linear classifier performs slightly better than the

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

linear classifier. The misclassifications are less for the non-linear classifier in comparison to the linear classifier.



In practical application, the choice of classifier depends on the problem at hand. The one that is performing better 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. In general, it is unlikely that a model is ideal and 100% accurate over unseen data. In case of classification problems, there are different metrics can be majorly understood as follows:

Error types and confusion matrix

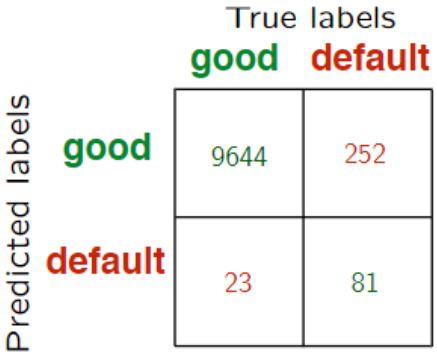
In above two figures, we observed that classifiers may have misclassifications. The misclassfiication or the error made by the classifier can be due to one of the following two cases:

- The data point is blue but the model predicts it to be red.
- The data point is red but the model predicts it to be blue.

The ultimate objective of the classification is to commit as few errors as possible on the new examples. In classification problems, we can represent the frequency of errors and the correct predictions as a matrix known as a confusion matrix. It is a tabular representation of the frequency of different combinations of predicted and actual outcomes. Using this, the performance of the classification model can be determined in terms of the fraction of misclassifications or accuracy gained by the model and some other metrics can also be calculated.

As an example, the below figure shows the confusion matrix for our loan default example. The green labels show the number of correct predictions (9644 and 81) and red represents the number of misclassifications (252 and 23).

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.



There are many classification algorithms to solve a classification problem. Let's discuss some of the most commonly used algorithms.

Model-based approach (Gaussian)

In this approach, the data is passed to a **probabilistic model**. For every possible outcome class, the model predicts the probability of each output label. Then the model is applied on the test data. On the predicted probability, a certain **threshold** is set to create the outcome labels. For example, if the threshold probability is 0.6 and the predicted probability is greater than 0.6, then the output label will be 1 (say) and if it is less than 0.6, then the output label 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, i.e., if a certain new record belongs to the distribution of good people, then the corresponding output label for that person will be a good person.

In the Gaussian model, there is a probability associated with every class k. This probability is known as the **prior probability** of the class. The mathematical expression of probability can be given as follows:

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

Where, π_k is the prior probability of class k and m is the number of classes.

For example, if the count of good people is 60 out of a total of 100 people, then the prior probability of good people is 60/100 = 0.6 and that for the bad people is 40/100 = 0.4.

Now, every person has some distinct features that help to classify them into classes of good and bad. If there is only one independent variable, the probability of a person to have some feature X given that the person is belonging to class k is given in the form of the following normal distribution:

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

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

Where, $P(X|Y = k)$ is the probability of X given the class of the person is k, γ_k is the normalizing factor, μ_k and σ_k are the mean and the standard deviation of the normal distribution.

For multivariate problems, it is given as follows:

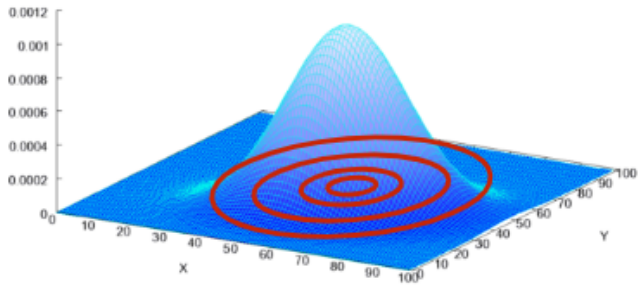
$$P(X|Y = k) = \gamma_k \, exp\{\frac{-1}{2} (X - \mu_k)^T C_k^{-1} (X - \mu_k)\}$$

Where, μ_k *is the mean vector*: $E[X|Y = k]$

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

For multivariate problems, the normal distribution is multidimensional and at constant heights contours are available. The mean vector represents the center of the distribution and the covariance matrix determines the shape of the distribution.

A **contour** is a locus of points on which the distribution takes a constant value. For example, in the below figure, each red curve is a contour at a specific height of the normal distribution.



For the multivariate normal distribution, the contours are **elliptical** and their orientation is defined by the covariance matrix.

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 posterior probability is defined as "given a certain X, what is the probability that it belongs to class label k". Mathematically, it can be given as follows:

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

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

The prior probability is a constant number, and it is associated with the posterior probability by the following expression, known as the **Bayes rule**.

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

Using the expression of $P(X|Y = k)$ and the Bayes rule, the numerator ntbhof the posterior probability can be given as follows:

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

Given the input vector X, choose the class with the maximum posterior probability.

If we simplify the above expression by taking the logarithmic, the expression can be written as:

$$log(\pi_k \gamma_k) - \frac{1}{2} (X - \mu_k)^T C_k^{-1} (X - \mu_k)$$

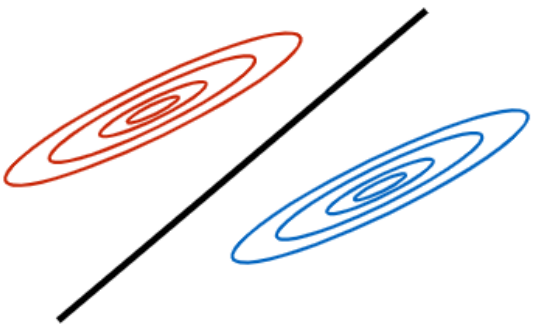
We compare the above expression for different classes k. When the above expression has the same value of some k and k', the boundary generated is quadratic in X. Hence, the algorithm is known as **Quadratic Discriminant Analysis (QDA)**.

When all the **covariance matrices are the same**, i.e., $C_k = C$ for all k, the boundary generated is linear in X. Hence, in that case, the algorithm is known as **Linear Discriminant Analysis (LDA).**

LDA versus QDA

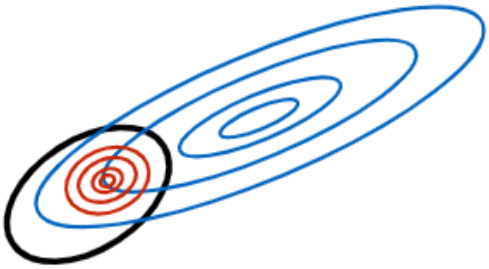
In the case where classes are **linearly separable,** LDA is used while in case of non-linearly separable classes QDA is more useful. It is quite important to understand when to use LDA and when to use QDA.

As in classification, it is important to keep data points of the same class together and data points with different classes apart from each other. In order to do so, LDA minimizes the distance within classes and maximizes the distance between the classes. For two populations corresponding to different classes, if means are different but covariances are the same, LDA is applied and it gives us a linear classifier (the black line in the below figure).

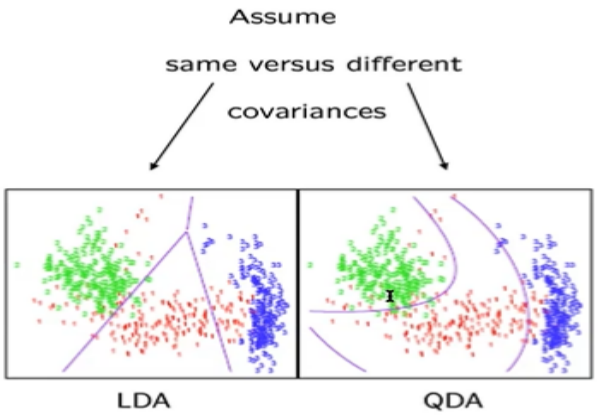


This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

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 (the black curve in the below figure).



In the below figure, it is clear that LDA creates a linear separation curve while QDA creates a quadratic curve applied in their corresponding circumstances.



Here, it is well demonstrated that if covariances are the same for each class, then LDA is applied while if they are different, then QDA is applied.

Prediction on Loan Deafult example

In our example of loan default, only **3.33% of the records corresponds to the "default" class**. So, even the model is dumb and all the data points are labeled as "no default"**,** the misclassification rate will only be 3.33% and the accuracy will be very high (~ 96.67%). This seems pretty high accuracy, but it is not useful.

The **misclassification rates** using different algorithms are as follows:

- Making predictions using a single feature ("balance") from the independent features:
  - LDA: 2.81%
  - QDA: 2.73%
- Make prediction using all the features:
  - LDA: 2.75%
  - QDA: 2.70%

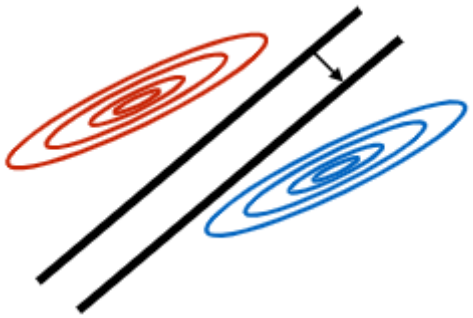
This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

When machine learning models make 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 **unbalanced dataset** is one where the proportion of output classes in very different. While doing classification on such problems, the committed errors might cost differently. The cost of committing one kind of error might be more costly than 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 the bank would lose money by giving loan to a defaulter. On the other hand, if the model is predicting a real “non-defaulter” as a “defaulter”, then the bank is losing a customer. The bank has lost the opportunity to provide a loan to that customer.

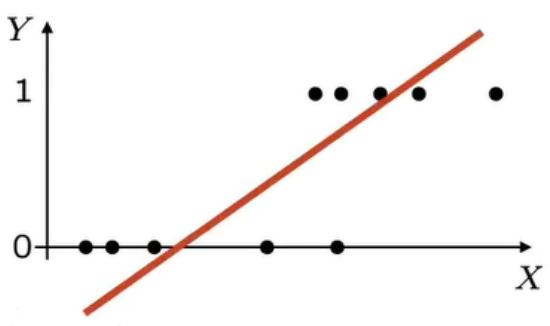
We can compare which error is more costly depending on the business context and shift the thresholds to reduce the number of costly errors.



Applying Regression for Classification

Now, let us answer a simple question: **why can’t we apply the linear regression algorithm to solve classification problems?**

Let us begin by simply applying the linear regression model to a classification task where the classes are 0 and 1.



This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

As we can observe in the above figure, the model is not capturing the pattern in the data. Also, in classification tasks, we want to know the probability of a record belonging to a class but since the linear regression line is a straight line that does not have a fixed range, it cannot give us probabilites. Furthermore, the error criterion used in linear regression (mean squared error) is not meaningful for classification tasks. Hence, a straight linear regression model is not preferred for any sort of classification task. Due to this, a new model is used to solve classification problems.

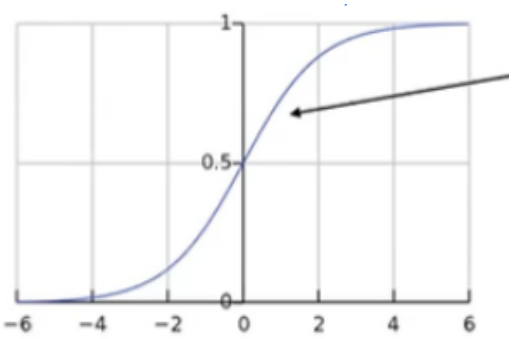
The Logistic model

The logistic regression model is associated with the same linear input as it is with the linear regression model. The only difference is that the 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:



The above figure shows that the value of the Sigmoid function lies between 0 and 1. Hence, the output can be considered as a probability.

If the predicted probability y is greater than the threshold, usually taken as 0.5, then the output is 1 else it is 0. In the case of multiple features, the equation of the sigmoid function can be given as follows:

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

Here, θ_i is the associated weight to the corresponding feature x_i. Based on different values of θ, the shape of the sigmoid function also varies in terms of the flexibility of the curve.

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

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 feature and the input features, it is crucial to find the weights 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, for each record. 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 weights** and it is the product of the probabilities 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 this equation has some mathematical advantages, for example, it is convex and gradients are easy to compute. Also, it is an efficient algorithm in practice.

Remark: Many techniques and concepts covered in linear regression apply to logistic regression as well. Some of those concepts are listed below:

1. Good predictions do not imply causality.
2. We can add a combination of features and/or transformed features to the data.
3. We can add regularization terms to the log-likelihood function to regularize the model.
4. We can use a separate validation set or K-Fold cross-validation to evaluate the performance of the model.

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

Results on the Loan Default example

As mentioned earlier, even if “no default” is always predicted, then the misclassification rate is only 3.33%.

- Making predictions using a single feature (“balance”) from the independent features:
    - LDA: 2.81%

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

- QDA: 2.73%
- Logistic regression: 2.73%

- Making predictions using all the features:
  - 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, other metrics like precision and recall might work better.

To handle such a situation, the cost function can be added with some extra weights. If one type of error costs more, then add more penalty or give more weightage to that error. Mathematically, this can be given as follows:

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

The additional weight w can be updated and it should be greater than 1. w is a hyperparameter and it is updated until we get a desired model performance.

We have covered a few algorithms used to solve classification problems. Another necessary kind of algorithm that works nicely in case of solving such problems is the **Nearest Neighbors** algorithm. In the nearest neighbors algorithm, 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 is the majority class in the vicinity of the target point.

One question that arises while applying 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 the K nearest neighbors (K-NN algorithm), where K is the number 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 class from the K selected points will be taken.

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

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.

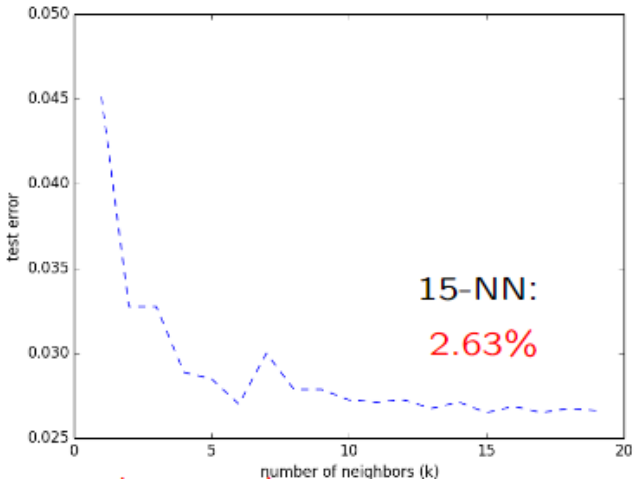
Let's consider extreme cases for values of K. If the value of K is equal to 1, the error on the training data will be 0 because each prediction would be the class of the data point itself. But the error on the test data will be high because the model is not general enough. So, we can say that the model will overfit the training data. On the other hand, if K is equal to the number of data points, it will predict the majority class in the full data for all the observations. Hence, the error will be high on the training data as well as the test data. So, we can say that the model will underfit the training data. In conclusion, very low and very high values of K, are not good.

Remark: K-NN is an algorithm that is suitable for both types of problems, namely regression, and classification in machine learning. In regression, we take the average of K nearest neighbors instead of the majority vote to make a prediction.

Selecting the value of K

One of the generic processes of selecting the value of K is to use the validation set approach. Set aside the 20% (for example) of data as the validation set and experiment with different values of K. For every value of K, build the model 100 times and check the average performance on the validation set. The model with a certain value of K that is performing the 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. As it can be seen, for very small value of K, the test error is very high because the prediction will rely on the values of a very small number of nearest neighbors. The test error is almost the same from from K=15 to K=20.

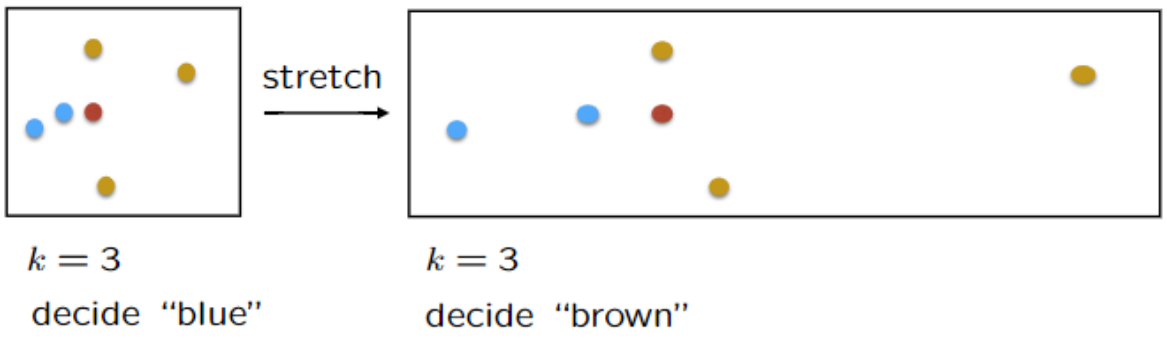
This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

So, we can chosen K=15 to keep the model simple. This is an example of **hyperparameter search** to choose between models/algorithms.

Effect of Scale

As mentioned earlier, the K-NN algorithm is dependent on distance computations to find the nearest neighbors. 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 neighbors changes significantly. The target is to find the nearest neighbors for the red point in the middle. In the first part which is without any change in scale, the closest points are the two blue dots, hence the predicted class will be “blue”. When a change in the scale is made, the closest points are the brown ones. Hence, the predicted class will be “brown”.



So, scale plays an important role in classifying a data point while using the K-NN algorithm.

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 discuss them briefly:

1. **Weighted Nearest Neighbors (Weighted NN)**

   Similar to the nearest neighbors algorithm, one of the methods is weighted nearest neighbors. In the K-NN algorithm, the majority vote between K neighbors is considered. The points can also be associated with weights depending on the corresponding distance with the target point. It is intuitive that less weight will be given to the points that are far

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

away and more weight will be given to the points that are close to each other.

2. **Kernel regression**

   A kernel is a fancy name for choosing 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 local regression or kernel regression.

   In some cases, if it is needed to use similar data points to make better predictions, then it is better to go with the kernel regression. The predictions in the local regression might be noisy because the data points are less in comparison to the full data. However, if the entire population is used, there is a chance that more irrelevant data points will be considered. So, local regression is more appropriate given that we have enough relevant or similar data points in the vicinity of the target point, otherwise, we cannot apply this method.

   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 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. We will study this algorithm in depth in the next course week.

3. **Neural Networks:** A neural network is a series of functions that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. We will study this algorithm in depth in the “Deep Learning” course week.

Interpretability of results

Interpreting the results from the model is a general issue in machine learning. Commonly for regression and classification algorithms, the associated hyperparameters give some interpretation of the model and the prediction criterion. For example, in linear regression, coefficients θ allow us to interpret the logic behind particular predictions.

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.

However, some algorithms are not interpretable. For example, in nearest neighbors, there are no coefficients associated. So, it is less interpretable. In general, more complex machine learning algorithms are less interpretable. For example, deep learning models are very less interpretable or even non-interpretable in most cases.

This file is meant for personal use by nehakinjal@gmail.com only.
Proprietary content. © Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited.
Sharing or publishing the contents in part or full is liable for legal action.