EVALUATING PERFORMANCE OF A MODEL

Supervised learning – classification

In supervised learning, one major task is classification. The responsibility of the classification model is to assign class label to the target feature based on the value of the predictor features.

For example, in the problem of predicting the win/loss in a cricket match, the classifier will assign a class value win/loss to target feature based on the values of other features like whether the team won the toss, number of spinners in the team, number of wins the team had in the tournament, etc. To evaluate the performance of the model, the number of correct classifications or predictions made by the model has to be recorded. A classification is said to be correct if, say for example in the given problem, it has been predicted by the model that the team will win and it has actually won.

Based on the number of correct and incorrect classifications or predictions made by a model, the accuracy of the model is calculated. If 99 out of 100 times the model has classified correctly, e.g. if in 99 out of 100 games what the model has predicted is same as what the outcome has been, then the model accuracy is said to be 99%. However, it is quite relative to say whether a model has performed well just by looking at the accuracy value.

For example, 99% accuracy in case of a sports win predictor model may be reasonably good but the same number may not be acceptable as a good threshold when the learning problem deals with predicting a critical illness. In this case, even the 1% incorrect prediction may lead to loss of many lives. So the model performance needs to be evaluated in light of the learning problem in question.

Also, in certain cases, erring on the side of caution may be preferred at the cost of overall accuracy. For that reason, we need to look more closely at the model accuracy and also at the same time look at other measures of performance of a model like sensitivity, specificity, precision, etc. So, let's start with looking at model accuracy more closely. And let's try to understand it with an example.

There are four possibilities with regards to the cricket match win/loss prediction:

1. the model predicted win and the team won
2. the model predicted win and the team lost
3. the model predicted loss and the team won
4. the model predicted loss and the team lost

In this problem, the obvious class of interest is 'win'

- The first case, i.e. the model predicted win and the team won is a case where the model has correctly classified data instances as the class of interest. These cases are referred as True Positive (TP) cases.
- The second case, i.e. the model predicted win and the team lost is a case where the model incorrectly classified data instances as the class of interest. These cases are referred as False Positive (FP) cases.
- The third case, i.e. the model predicted loss and the team won is a case where the model has incorrectly classified as not the class of interest. These cases are referred as False Negative (FN) cases.

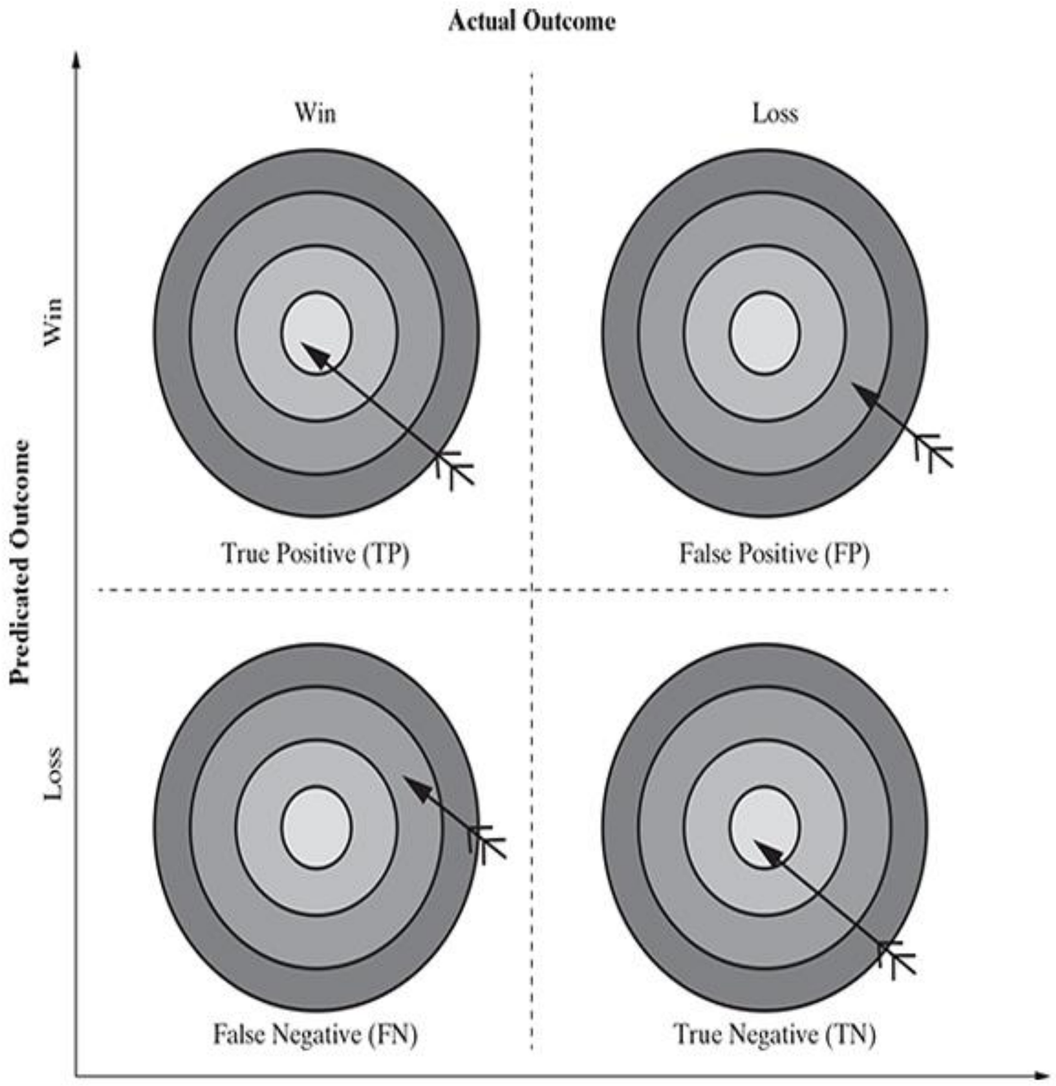

FIG. 3.7 Details of model classification

- The fourth case, i.e. the model predicted loss and the team lost is a case where the model has correctly classified as not the class of interest. These cases are referred as True Negative (TN) cases. All these four cases are depicted in Figure 3.7 .

For any classification model, model accuracy is given by total number of correct classifications (either as the class of interest, i.e. True Positive or as not the class of interest, i.e. True Negative) divided by total number of classifications done.

$$\text{Model accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

A matrix containing correct and incorrect predictions in the form of TPs, FPs, FNs and TNs is known as confusion matrix. The win/loss prediction of cricket match has two classes of interest – win and loss. For that reason it will generate a 2 × 2 confusion matrix. For a classification problem involving three classes, the confusion matrix would be 3 × 3, etc.

Let’s assume the confusion matrix of the win/loss prediction of cricket match problem to be as below.

	ACTUAL WIN	ACTUAL LOSS
Predicted Win	85	4
Predicted Loss	2	9

In context of the above confusion matrix, total count of TPs = 85, count of FPs = 4, count of FNs = 2 and count of TNs = 9.

$$\therefore \text{Model accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} = \frac{85 + 9}{85 + 4 + 2 + 9} = \frac{94}{100} = 94\%$$

The percentage of misclassifications is indicated using error rate which is measured as

$$\text{Error rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

In context of the above confusion matrix,

$$\text{Error rate} = \frac{FP + FN}{TP + FP + FN + TN} = \frac{4 + 2}{85 + 4 + 2 + 9} = \frac{6}{100} = 6\%$$
$$= 1\text{–Model accuracy}$$

Sometimes, correct prediction, both TPs as well as TNs, may happen by mere coincidence. Since these occurrences boost model accuracy, ideally it should not happen. Kappa value of a model indicates the adjusted the model accuracy. It is calculated using the formula below

$$\text{Kappa value (k)} = \frac{P(a) - P(p_r)}{1 - P(p_r)}$$

P(a) = Proportion of observed agreement between actual and predicted in overall data set

$$= \frac{TP + TN}{TP + FP + FN + TN}$$

$P(p_r)$ = Proportion of expected agreement between actual and predicted data both in case of class of interest as well as the other classes

$$= \frac{TP + FP}{TP + FP + FN + TN} \times \frac{TP + FN}{TP + FP + FN + TN} + \frac{FN + TN}{TP + FP + FN + TN} \times \frac{FP + TN}{TP + FP + FN + TN}$$

In context of the above confusion matrix, total count of TPs = 85, count of FPs = 4, count of FNs = 2 and count of TNs = 9.

$$\therefore P(a) = \frac{TP + TN}{TP + FP + FN + TN} = \frac{85 + 9}{85 + 4 + 2 + 9} = \frac{94}{100} = 0.94$$

$$P(p_r) = \frac{85 + 4}{85 + 4 + 2 + 9} \times \frac{85 + 2}{85 + 4 + 2 + 9} + \frac{2 + 9}{85 + 4 + 2 + 9} \times \frac{4 + 9}{85 + 4 + 2 + 9}$$

$$= \frac{89}{100} \times \frac{87}{100} + \frac{11}{100} \times \frac{13}{100} = 0.89 \times 0.87 + 0.11 \times 0.13 = 0.7886$$

$$\therefore k = \frac{0.94 - 0.7886}{1 - 0.7886} = 0.7162$$

As discussed earlier, in certain learning problems it is critical to have extremely low number of FN cases, if needed, at the cost of a conservative classification model. Though it is a clear case of misclassification and will impact model accuracy adversely, it is still required as missing each class of interest may have serious consequence. This happens more in problems from medical domains like disease prediction problem. For example, if a tumor is malignant but wrongly classified as benign by the classifier, then the repercussion of such misclassification is fatal. It does not matter if higher number of tumours which are benign are wrongly classified as malignant. In these problems there are some measures of model performance which are more important than accuracy. Two such critical measurements are sensitivity and specificity of the model.

The sensitivity of a model measures the proportion of TP examples or positive cases which were correctly classified. It is measured as

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

In the context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{85}{85 + 2} = \frac{85}{87} = 97.7\%$$

So, again taking the example of the malignancy prediction of tumours, class of interest is ‘malignant’. Sensitivity measure gives the proportion of tumours which are actually malignant and have been predicted as malignant. It is quite obvious that for such problems the most critical measure of the performance of a good model is sensitivity. A high value of sensitivity is more desirable than a high value of accuracy.

Specificity is also another good measure to indicate a good balance of a model being excessively conservative or excessively aggressive. Specificity of a model measures the proportion of negative examples which have been correctly classified. In the context, of malignancy prediction of tumours, specificity gives the proportion of benign tumours which have been correctly classified. In the context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{9}{9 + 4} = \frac{9}{13} = 69.2\%$$

A higher value of specificity will indicate a better model performance. However, it is quite understandable that a conservative approach to reduce False Negatives might actually push up the number of FPs. Reason for this is that the model, in order to reduce FNs, is going to classify more tumours as malignant. So the chance that benign tumours will be classified as malignant or FPs will increase.

There are two other performance measures of a supervised learning model which are similar to sensitivity and specificity. These are precision and recall. While precision gives the proportion of positive predictions which are truly positive, recall gives the proportion of TP cases over all actually positive cases.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision indicates the reliability of a model in predicting a class of interest. When the model is related to win / loss prediction of cricket, precision indicates how often it predicts the win correctly. In context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{85}{85 + 4} = \frac{85}{89} = 95.5\%$$

Recall indicates the proportion of correct prediction of positives to the total number of positives. In case of win/loss prediction of cricket, recall resembles what proportion of the total wins were predicted correctly

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

In the context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{85}{85 + 2} = \frac{85}{87} = 97.7\%$$

F-measure

F-measure is another measure of model performance which combines the precision and recall. It takes the harmonic mean of precision and recall as calculated as

$$F\text{-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

In context of the above confusion matrix for the cricket match win prediction problem,

$$F\text{-measure} = \frac{2 \times 0.955 \times 0.977}{0.955 + 0.977} = \frac{1.866}{1.932} = 96.6\%$$

As a combination of multiple measures into one, F-score gives the right measure using which performance of different models can be compared.

Receiver operating characteristic (ROC) curves

Receiver Operating Characteristic (ROC) curve helps in visualizing the performance of a classification model. It shows the efficiency of a model in the detection of true positives while avoiding the occurrence of false positives. To refresh our memory, true positives are those cases where the model has correctly classified data instances as the class of interest. For example, the model has correctly classified the tumours as malignant, in case of a tumour malignancy prediction problem. On the other hand, FPs are those cases where the model incorrectly classified data instances as the class of interest. Using the same example, in this case, the model has incorrectly classified the tumours as malignant, i.e. tumours which are actually benign have been classified as malignant

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

In the ROC curve, the FP rate is plotted (in the horizontal axis) against true positive rate (in the vertical axis) at different classification thresholds. If we assume a lower value of classification threshold, the model classifies more items as positive. Hence, the values of both False Positives and True Positives increase. The area under curve (AUC) value, as shown in figure 3.8a , is the area of the two-dimensional space under the curve extending from (0, 0) to (1, 1), where each point on the curve gives a set of true and false positive values at a specific classification threshold. This curve gives an indication of the predictive quality of a model. AUC value ranges from 0 to 1, with an AUC of less than 0.5 indicating that the classifier has no predictive ability. Figure 3.8b shows the curves of two classifiers – classifier 1 and classifier 2. Quite obviously, the AUC of classifier 1 is more than the AUC of classifier 2. So, we can draw the inference that classifier 1 is better than classifier 2

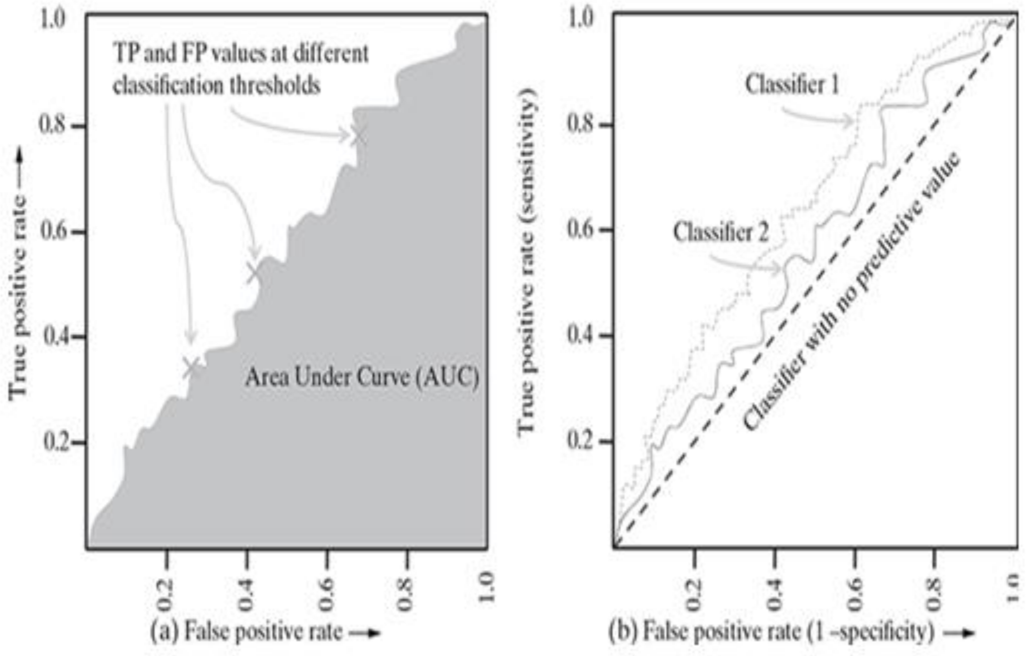

FIG. 3.8 ROC curve

A quick indicative interpretation of the predictive values from 0.5 to 1.0 is given below:

0.5 – 0.6 ➔ Almost no predictive ability

0.6 – 0.7 ➔ Weak predictive ability

0.7 – 0.8 ➔ Fair predictive ability

0.8 – 0.9 ➔ Good predictive ability

0.9 – 1.0 ➔ Excellent predictive ability.

Supervised learning – regression

A well-fitted regression model churns out predicted values close to actual values. Hence, a regression model which ensures that the difference between predicted and actual values is low can be considered as a good model. Figure 3.9 represents a very simple problem of real estate value prediction solved using linear regression model. If 'area' is the predictor variable (say x) and 'value' is the target variable (say y), the linear regression model can be represented in the form:

$$y = \alpha + \beta x$$

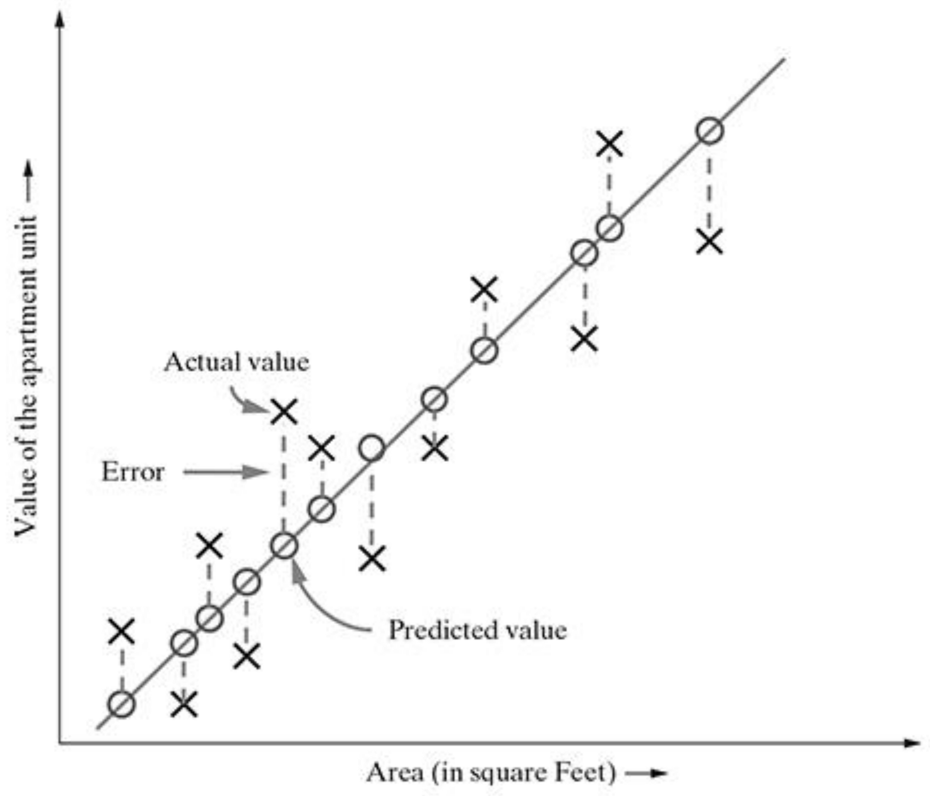

FIG. 3.9 Error – Predicted vs. actual value

For a certain value of x, say x̂, the value of y is predicted as ŷ whereas the actual value of y is Y (say). The distance between the actual value and the fitted or predicted value, i.e. ŷ is known as residual. The regression model can be considered to be fitted well if the difference between actual and predicted value, i.e. the residual value is less.

R-squared is a good measure to evaluate the model fitness. It is also known as the coefficient of determination, or for multiple regression, the coefficient of multiple determination. The R-squared value lies between 0 to 1 (0%–100%) with a larger value representing a better fit. It is calculated as:

$$R^2 = \frac{SST - SSE}{SST}$$

Sum of Squares Total (SST) = squared differences of each observation from the overall mean $= \sum_{i=1}^{n} (y_i - \bar{y})^2$ where $\bar{y}$ is the mean.

Sum of Squared Errors (SSE) (of prediction) = sum of the squared residuals $= \sum_{i=1}^{n} (Y_i - \hat{y})^2$ where $\hat{y}_i$ is the predicted value of y_i and Y_i is the actual value of y_i.

IMPROVING PERFORMANCE OF A MODEL

One effective way to improve model performance is by tuning model parameter. Model parameter tuning is the process of adjusting the model fitting options. For example, in the popular classification model k-Nearest Neighbour (kNN), using different values of ‘k’ or the number of nearest neighbours to be considered, the model can be tuned. In the same way, a number of hidden layers can be adjusted to tune the performance in neural networks model. Most machine learning models have at least one parameter which can be tuned.

As an alternate approach of increasing the performance of one model, several models may be combined together. The models in such combination are complimentary to each other, i.e. one model may learn one type data sets well while struggle with another type of data set. Another model may perform well with the data set which the first one struggled with. This approach of combining different models with diverse strengths is known as ensemble (depicted in Figure 3.11). Ensemble helps in averaging out biases of the different underlying models and also reducing the variance. Ensemble methods combine weaker learners to create stronger ones. A performance boost can be expected even if models are built as usual and then ensembled. Following are the typical steps in ensemble process:

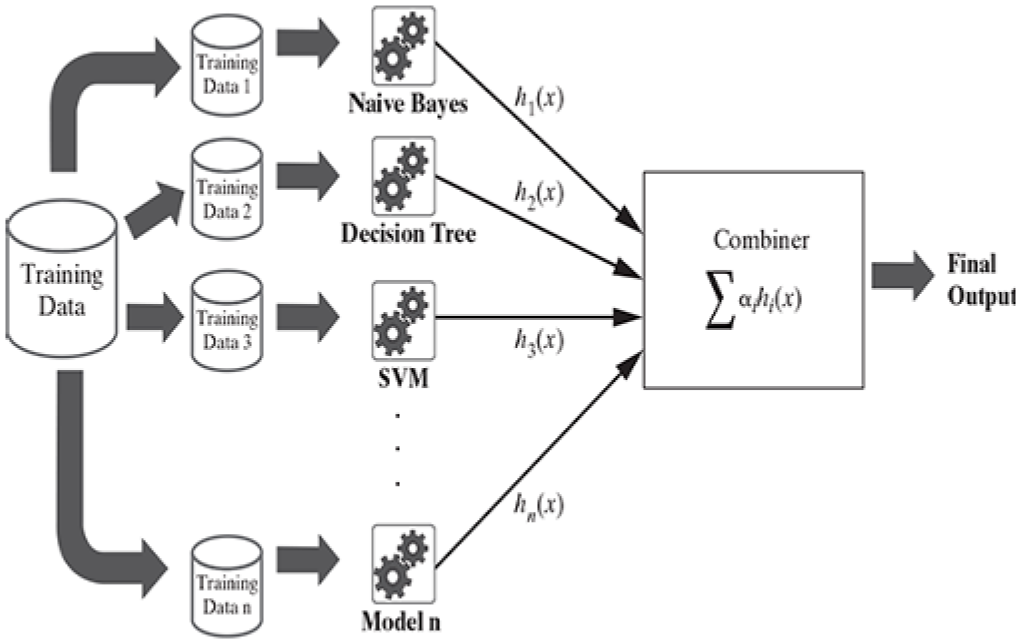

FIG. 3.11 Ensemble

One of the earliest and most popular ensemble models is bootstrap aggregating or bagging. Bagging uses bootstrap sampling method (refer section 3.3.3) to generate multiple training data sets. These training data sets are used to generate (or train) a set of models using the same learning algorithm. Then the outcomes of the models are combined by majority voting (classification) or by average (regression). Bagging is a very simple ensemble technique which can perform really well for unstable learners like a decision tree, in which a slight change in data can impact the outcome of a model significantly.

Just like bagging, boosting is another key ensemble-based technique. In this type of ensemble, weaker learning models are trained on resampled data and the outcomes are combined using a weighted voting approach based on the performance of different models. Adaptive boosting or AdaBoost is a special variant of boosting algorithm. It is based on the idea of generating weak learners and slowly learning.

What is a feature?

A feature is an attribute of a data set that is used in a machine learning process. There is a view amongst certain machine learning practitioners that only those attributes which are meaningful to a machine learning problem are to be called as features, but this view has to be taken with a pinch of salt. In fact, selection of the subset of features which are meaningful for machine learning is a sub-area of feature engineering which draws a lot of research interest. The features in a data set are also called its dimensions. So a data set having 'n' features is called an n-dimensional data set.

What is feature engineering?

Feature engineering refers to the process of translating a data set into features such that these features are able to represent the data set more effectively and result in a better learning performance.

As we know already, feature engineering is an important pre-processing step for machine learning. It has two major elements:

1. feature transformation
2. feature subset selection

Feature transformation transforms the data – structured or unstructured, into a new set of features which can represent the underlying problem which machine learning is trying to solve. There are two variants of feature transformation:

1. feature construction
2. feature extraction

Both are sometimes known as feature discovery

Feature construction process discovers missing information about the relationships between features and augments the feature space by creating additional features. Hence, if there are 'n' features or dimensions in a data set, after feature construction 'm' more features or dimensions may get added. So at the end, the data set will become 'n + m' dimensional.

Feature extraction is the process of extracting or creating a new set of features from the original set of features using some functional mapping.