UNIT II

Introduction, selecting a Model, training a Model (for Supervised Learning), Model Representation and Interpretability, Evaluating Performance of a Model, Improving Performance of a Model Basics of Feature Engineering: Introduction, Feature Transformation, Feature Subset Selection.

INTRODUCTION

The learning process of machines may seem quite magical to somebody who is new to machine learning. The thought that a machine is able to think and take intelligent action , human learning by applying mathematical and statistical formulations. In that sense, both human and machine learning strives to build formulations or mapping based on a limited number of observations.

The basic learning process, irrespective of the fact that the learner is a human or a machine, can be divided into three parts:

1. Data Input

2. Abstraction

3. Generalization

A criminal is going to launch an attack on the main candidate. However, it is not known who the person is and quite obviously the person might use some disguise. The only thing that is for sure is the person is a history sheeter or a criminal having a long record of serious crime. From the criminal database, a list of such criminals along with their photographs has been collected. Also, the photos taken by security cameras positioned at different places near the gathering are available with the detective department. They have to match the photos from the criminal database with the faces in the gathering to spot the potential attacker. So the main problem here is to spot the face of the criminal based on the match with the photos in the criminal database. This can be done using human learning where a person from the detective department can scan through each shortlisted photo and try to match that photo with the faces in the gathering. A person having a strong memory can take a glance at the photos of all criminals in one shot and then try to find a face in the gathering which closely resembles one of the criminal photos that she has viewed. Easy, isn't it? But that is not possible in reality. The number of criminals in the database and hence the count of photos runs in hundreds, if not thousands. the learning process, abstraction is a significant step as it represents raw input data in a summarized and structured format, such that a meaningful insight is obtained from the data. This structured representation of raw input data to the meaningful pattern is called a model. The model might have different forms. It might be a mathematical equation, it might be a graph or tree structure, it might be a computational block, etc. The decision regarding which model is to be selected for a specific data set is taken by the learning task, based on the problem to be solved and the type of data.

The process of assigning a model, and fitting a specific model to a data set is called model training. Once the model is trained, the raw input data is summarized into an abstracted form. However, with abstraction, the learner is able to only summarize the knowledge. consisting of a huge number of feature-based data and inter-relations. To generate actionable insight from such broad-based knowledge is very difficult. This is where generalization comes into play. Generalization searches through the huge set of abstracted knowledge to come up with a small and manageable set of key findings .

SELECTING A MODEL

The basic learning process and have understood model abstraction and generalization in that context, let's try to formalize it in context of a motivating example. Continuing the thread of the potential attack during the election campaign, New City Police department has succeeded in foiling the bid to attack the electoral candidate. Input variables can be denoted by X, while individual input variables are represented as X , X , X , …, X and output variable by symbol Y. The relationship between X and Y is represented in the general form: Y = f (X) + e, where 'f ' is the target function and 'e' is a random error term. that there are three broad categories of machine learning approaches used for resolving different types of problems. Quickly recapitulating, they are ,

1. Supervised
    1. Classification
    2. Regression
2. Unsupervised
    1. Clustering
    2. Association analysis
3. Reinforcement

Multiple factors play a role when we try to select the model for solving a machine learning problem. The most important factors are (i) the kind of problem we want to solve using machine learning and (ii) the nature of the underlying data. The problem may be related to the prediction of a class value like whether a tumour is malignant or benign, whether the next day will be snowy or rainy, etc. It may be related to prediction – but of some numerical value like what the price of a house should be in the next quarter, what is the expected growth of a certain IT stock in the next 7 days, etc. Certain problems are related to grouping of data like finding customer segments that are using a certain product, movie genres which have got more box office success in the last one year, etc. Machine learning algorithms are broadly of two types: models for supervised learning, which primarily focus on solving predictive problems and models for unsupervised learning, which solve descriptive problems.

PREDICTIVE MODELS

Models for supervised learning or predictive models, as is understandable from the name itself, try to predict certain value using the values in an input data set. The learning model attempts to establish a relation between the target feature, i.e. the feature being predicted, and the predictor features. The predictive models have a clear focus on what they want to learn and how they want to learn.

Predictive models, in turn, may need to predict the value of a category or class to which a data instance belongs to. Below are some examples:

1. Predicting win/loss in a cricket match

2. Predicting whether a transaction is fraud

3. Predicting whether a customer may move to another product

The target feature is known as a class and the categories to which classes are divided into are called levels. Some of the popular classification models include k-Nearest Neighbor (kNN), Naïve Bayes, and Decision Tree.

Predictive models may also be used to predict numerical values of the target feature based on the predictor features. Below are some examples:

1. Prediction of revenue growth in the succeeding year

2. Prediction of rainfall amount in the coming monsoon

3. Prediction of potential flu patients and demand for flu shots next winter

The models which are used for prediction of the numerical value of the target feature of a data instance are known as regression models. Linear Regression and Logistic Regression models are popular regression models.

DESCRIPTIVE MODELS

Models for unsupervised learning or descriptive models are used to describe a data set or gain insight from a data set. There is no target feature or single feature of interest in case of unsupervised learning. Based on the value of all features, interesting patterns or insights are derived about the data set.

Descriptive models which group together similar data instances, i.e. data instances having a similar value of the different features are called clustering models. Examples of clustering include

1. Customer grouping or segmentation based on social, demographic, ethnic, etc. factors

2. Grouping of music based on different aspects like genre, language, time-period, etc.

3. Grouping of commodities in an inventory

The most popular model for clustering is k-Means.

Descriptive models related to pattern discovery is used for market basket analysis of transactional data. In market basket analysis, based on the purchase pattern available in the transactional data, the possibility of purchasing one product based on the purchase of another product is determined.

TRAINING A MODEL (FOR SUPERVISED LEARNING):-

HOLDOUT METHOD

In case of supervised learning, a model is trained using the labelled input data. However, how can we understand the performance of the model? The test data may not be available immediately. Also, the label value of the test data is not known. That is the reason why a part of the input data is held back (that is how the name holdout originates) for evaluation of the model. This subset of the input data is used as the test data for evaluating the performance of a trained model. In general 70%–80% of the input data (which is obviously labelled) is used for model training. The remaining 20%–30% is used as test data for validation of the performance of the model. However, a different proportion of dividing the input data into training and test data is also acceptable. To make sure that the data in both the buckets are similar in nature, the division is done randomly. Random numbers are used to assign data items to the partitions. This method of partitioning the input data into two parts – training and test data (depicted in Figure 3.1), which is by holding back a part of the input data for validating the trained model is known as holdout method.

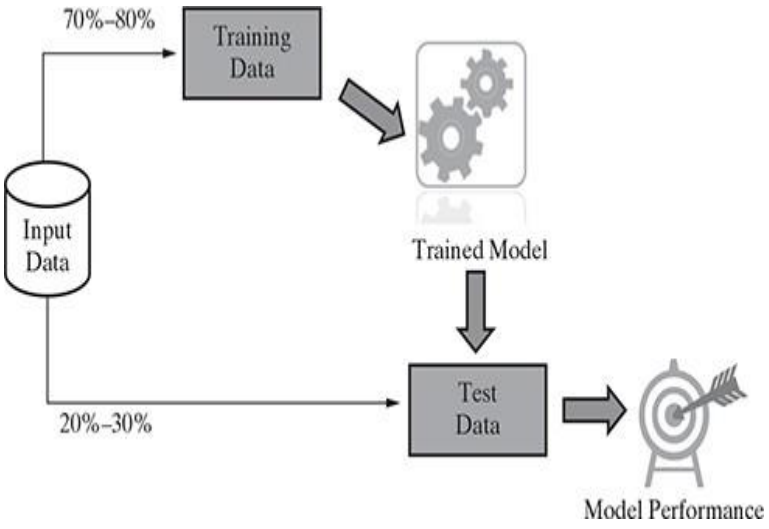


FIG. 3.1 Holdout method

The division of data of different classes into the training and test data may not be proportionate. This situation is worse if the overall percentage of data related to certain classes is much less compared to other classes. This may happen despite the fact that random sampling is employed for test data selection. This problem can be addressed to some extent by applying stratified random sampling in place of sampling. In case of stratified random sampling, the whole data is broken into several homogenous groups or strata and a random sample is selected from each such stratum. This ensures that the generated random partitions have equal proportions of each class.

K-FOLD CROSS-VALIDATION METHOD

Holdout method employing stratified random sampling approach still heads into issues in certain specific situations. Especially, the smaller data sets may have the challenge to divide the data of some of the classes proportionally amongst training and test data sets. A special variant of holdout method, called repeated holdout, is sometimes employed to ensure the randomness of the composed data sets. This process of repeated holdout is the basis of k-fold cross-validation technique. In k-fold cross-validation, the data set is divided into kcompletely distinct or non-overlapping random partitions called folds. Figure 3.2 depicts an overall approach for k-fold cross-validation.

The value of 'k' in k-fold cross-validation can be set to any number. However, there are two approaches which are extremely popular:

1. 10-fold cross-validation (10-fold CV)

2. Leave-one-out cross-validation (LOOCV)

10-fold cross-validation is by far the most popular approach. In this approach, for each of the 10-folds, each comprising of approximately 10% of the data, one of the folds is used as the test data for validating model performance trained based on the remaining 9 folds (or 90% of the data). This is repeated 10 times, once for each of the 10 folds being used as the test data and the remaining folds as the training data.The average performance across all folds is being reported. The entire data set is broken into 'k' folds – out of which one fold is selected in each iteration as the test data set. The fold selected as test data set in each of the 'k' iterations is different. the circles resemble the records in the input data set, the contiguous circles represented as folds do not mean that they are subsequent records in the data set. This is more a virtual representation and not a physical representation. As already mentioned, the records in a fold are drawn by using random sampling technique.

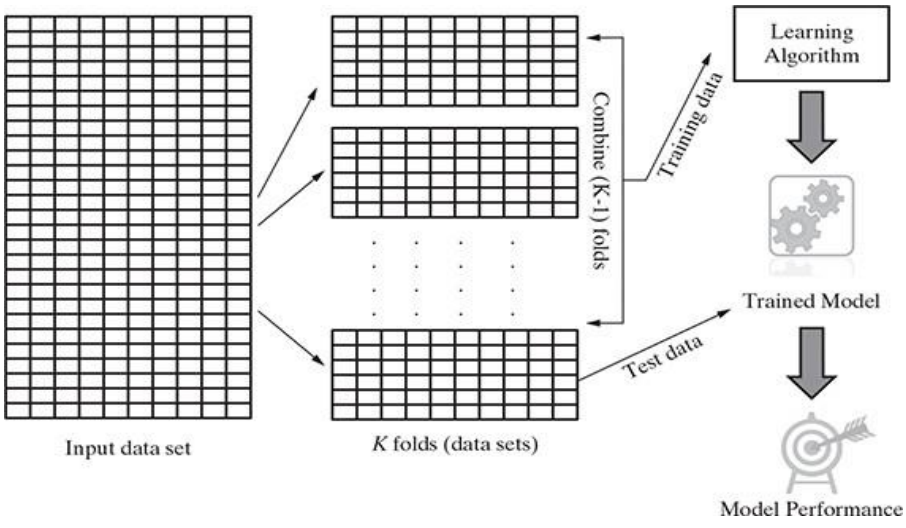


FIG. 3.2 Overall approach for K-fold cross-validation

BOOTSTRAP SAMPLING

Bootstrap sampling or simply bootstrapping is a popular way to identify training and test data sets from the input data set. It uses the technique of Simple Random Sampling with Replacement (SRSWR), which is a wellknown technique in sampling theory for drawing random samples. We have seen earlier that k-fold crossvalidation divides the data into separate partitions – say 10 partitions in case of 10-fold cross-validation. Then it uses data instances from partition as test data and the remaining partitions as training data. Unlike this approach adopted in case of k-fold cross- validation, bootstrapping randomly picks data instances from the input data set, with the possibility of the same data instance to be picked multiple times. This essentially means that from the input data set having 'n' data instances, bootstrapping can create one or more training data sets having 'n' data instances, some of the data instances being repeated multiple times. Figure 3.4 briefly presents the approach followed in bootstrap sampling.

This technique is particularly useful in case of input data sets of small size, i.e. having very less number of data instances.

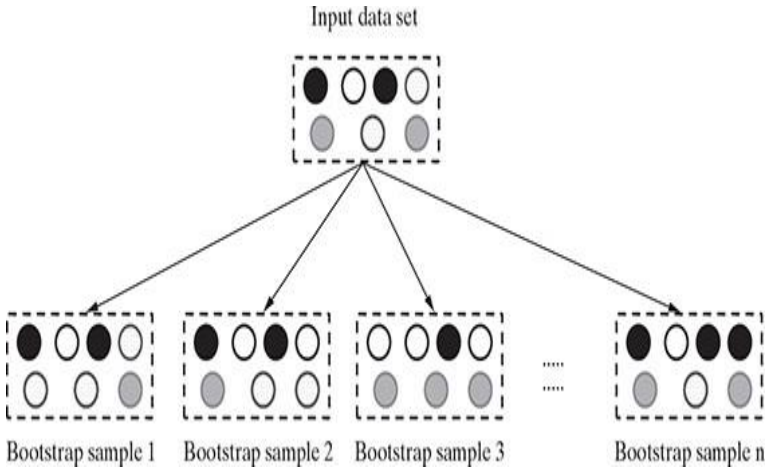


FIG. 3.4 Bootstrap sampling

CROSS-VALIDATION	BOOTSTRAPPING
It is a special variant of holdout method, called repeated holdout. Hence uses stratified random sampling approach (without replacement). Data set is divided into 'k' random partitions, with each partition containing approximately $\frac{n}{k}$ number of unique data elements, where 'n' is the total number of data elements and 'k' is the total number of folds.	It uses the technique of Simple Random Sampling with Replacement (SRSWR). So the same data instance may be picked up multiple times in a sample.
The number of possible training/test data samples that can be drawn using this technique is finite.	In this technique, since elements can be repeated in the sample, possible number of training/test data samples is unlimited.

LAZY VS. EAGER LEARNER

It tries to construct a generalized, inputindependent target function during the model training phase. It follows the typical steps of machine learning, i.e. abstraction and generalization and comes up with a trained model at the end of the learning phase. Hence, when the test data comes in for classification, the eager learner is ready with the model and doesn't need to refer back to the training data. Eager learners take more time in the learning phase than the lazy learners. Some of the algorithms which adopt eager learning approach include Decision Tree, Support Vector Machine, Neural Network, etc.

Lazy learning, on the other hand, completely skips the abstraction and generalization processes, as explained in context of a typical machine learning process. In that respect, strictly speaking, lazy learner doesn't 'learn' anything. It uses the training data in exact, and uses the knowledge to classify the unlabelled test data. They are also called non-parametric learning. Lazy learners take very little time in training because not much of training actually happens. However, it takes quite some time in classification as for each tuple of test data, a comparison-based assignment of label happens. One of the most popular algorithm for lazy learning is k-nearest neighbor.

MODEL REPRESENTATION AND INTERPRETABILITY

We have already seen that the goal of supervised machine learning is to learn or derive a target function which can best determine the target variable from the set of input variables. A key consideration in learning the target function from the training data is the extent of generalization. This is because the input data is just a limited, specific view and the new, unknown data in the test data set may be differing quite a bit from the training data.

Fitness of a target function approximated by a learning algorithm determines how correctly it is able to classify a set of data it has never seen.

UNDERFITTING

If the target function is kept too simple, it may not be able to capture the essential nuances and represent the underlying data well. A typical case of underfitting may occur when trying to represent a non-linear data with a linear model as demonstrated by both cases of underfitting shown in figure 3.5. Many times underfitting happens due to unavailability of sufficient training data. Underfitting results in both poor performance with training data as well as poor generalization to test data. Underfitting can be avoided by

1. using more training data

2. reducing features by effective feature selection

OVERFITTING

Overfitting refers to a situation where the model has been designed in such a way that it emulates the training data too closely. In such a case, any specific deviation in the training data, like noise or outliers, gets embedded in the model. It adversely impacts the performance of the model on the test data. Overfitting, in many cases, occur as a result of trying to fit an excessively complex model to closely match the training data. This is represented with a sample data set in figure 3.5 . The target function, in these cases, tries to make sure all training data points are correctly partitioned by the decision boundary. However, more often than not, this exact nature is not replicated in the unknown test data set. Hence, the target function results in wrong classification in the test data set. Overfitting results in good performance with training data set, but poor generalization and hence poor performance with test data set. Overfitting can be avoided by

1. using re-sampling techniques like k-fold cross validation

2. hold back of a validation data set

3. remove the nodes which have little or no predictive power for the given machine learning problem.

BIAS – VARIANCE TRADE OFF

In supervised learning, the class value assigned by the learning model built based on the training data may differ from the actual class value. This error in learning can be of two types – errors due to 'bias' and error due to 'variance'. Let's try to understand each of them in details.

Errors due to 'Bias'

Errors due to bias arise from simplifying assumptions made by the model to make the target function less complex or easier to learn. In short, it is due to underfitting of the model. Parametric models generally have high bias making them easier to understand/interpret and faster to learn. These algorithms have a poor performance on data sets, which are complex in nature and do not align with the simplifying assumptions made by the algorithm. Underfitting results in high bias.

Errors due to 'Variance'

Errors due to variance occur from difference in training data sets used to train the model. Different training data sets (randomly sampled from the input data set) are used to train the model. Ideally the difference in the data sets should not be significant and the model trained using different training data sets should not be too different. However, in case of overfitting, since the model closely matches the training data, even a small difference in training data gets magnified in the model.

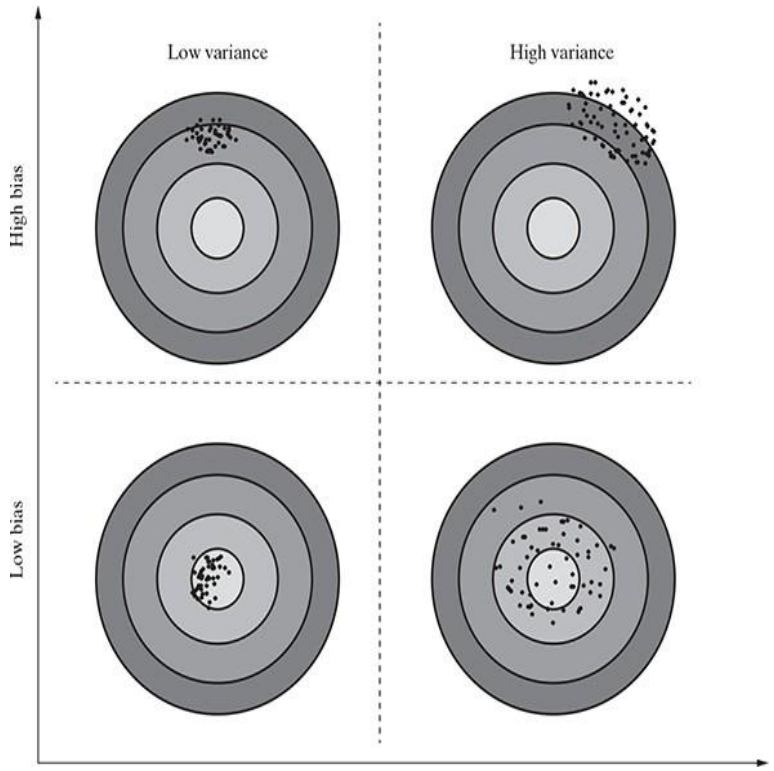


FIG. 3.6 Bias-variance trade-off

On one hand, parametric algorithms are generally seen to demonstrate high bias but low variance. On the other hand, non-parametric algorithms demonstrate low bias and high variance.

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:

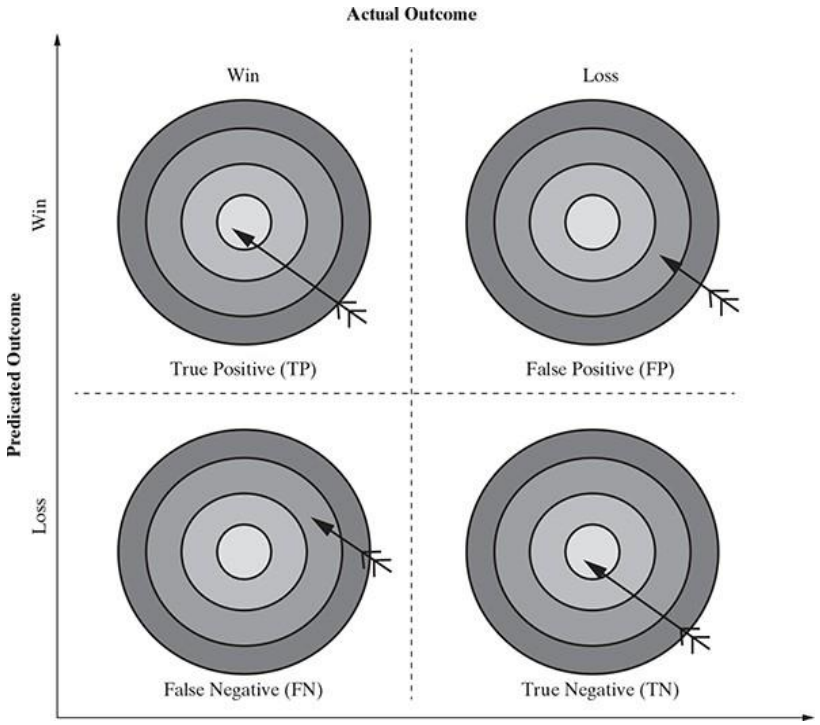
1. The model predicted win and the team won

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

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.



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. 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.

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

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

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

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.

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

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.

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.

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

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

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

As a combination of multiple measures into one, Fscore gives the right measure using which performance of different models can be compared. However, one assumption the calculation is based on is that precision and recall have equal weight, which may not always be true in reality.

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.

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

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

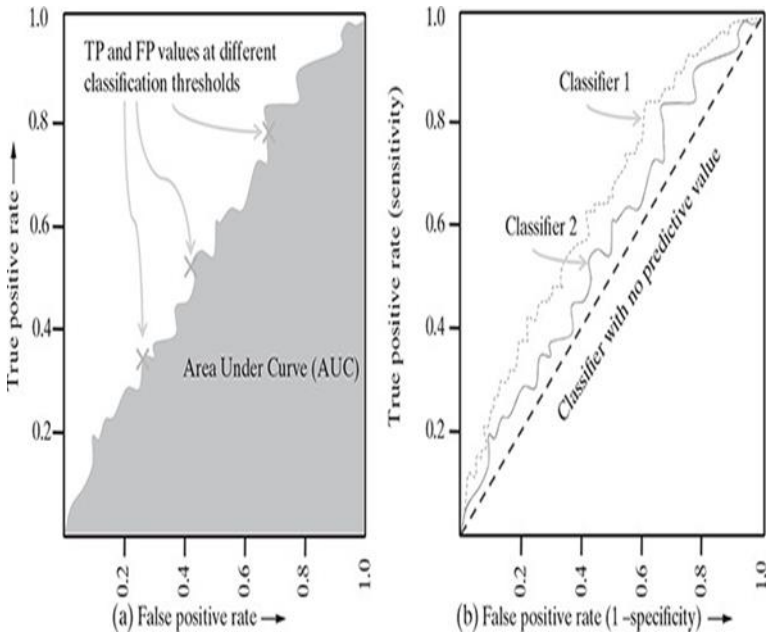


FIG. 3.8 ROC curve

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:

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

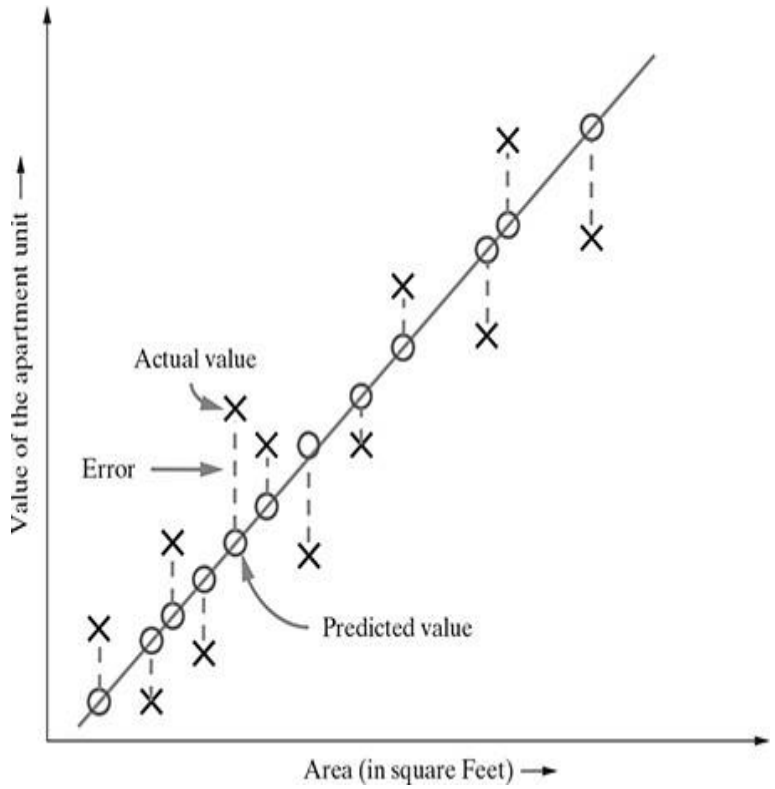


FIG. 3.9 Error – Predicted vs. actual value

For a certain value of x, say $\hat{x}$, the value of y is predicted as $\hat{y}$ whereas the actual value of y is Y (say). The distance between the actual value and the fitted or predicted value, i.e. $\hat{y}$ is known as residual.

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 is the predicted value of y and Y is the actual value of y_i

UNSUPERVISED LEARNING – CLUSTERING

Clustering algorithms try to reveal natural groupings amongst the data sets. However, it is quite tricky to evaluate the performance of a clustering algorithm. Clustering, by nature, is very subjective and whether the cluster is good or bad is open for interpretations. It was noted, 'clustering is in the eye of the beholder'. This stems from the two inherent challenges which lie in the process of clustering:

1. It is generally not known how many clusters can be formulated from a particular data set. It Is completely open-ended in most cases and provided as a user input to a clustering algorithm.

2. Even if the number of clusters is given, the same number of clusters can be formed with different groups of data instances.

(a) Internal evaluation

In this approach, the cluster is assessed based on the underlying data that was clustered. The internal evaluation methods generally measure cluster quality based on homogeneity of data belonging to i i i. the same cluster and heterogeneity of data belonging to different clusters. The homogeneity/heterogeneity is decided by some similarity measure. For example, silhouette coefficient, which is one of the most popular internal evaluation methods, uses distance (Euclidean or Manhattan distances most commonly used) between data elements as a similarity measure. The value of silhouette width ranges between –1 and +1, with a high value indicating high intracluster homogeneity and inter-cluster heterogeneity

For a data set clustered into 'k' clusters, silhouette width is calculated as:

$$\text{Silhouette width} = \frac{b(i) - a(i)}{\max\{a(i),\ b(i)\}}$$

In the same way, let's calculate the distance of an arbitrary data element 'i' in cluster 1 with the different data elements from another cluster, say cluster 4 and take an average of all those distances. Hence,

$$b_{14}(\text{average}) = \frac{b_{14}(1) + b_{14}(2) + \ \ldots\ \ + b_{14}(n_4)}{(n_4)}$$

where n is the total number of elements in cluster 4. In the same way, we can calculate the values of b (average) and b (average). b (i) is the minimum of all these values. Hence, we can say that,

b(i) = minimum [b (average), b (average), b (average)]

(b) External evaluation

In this approach, class label is known for the data set subjected to clustering. However, quite obviously, the known class labels are not a part of the data used in clustering. The cluster algorithm is assessed based on how close the results are compared to those known class labels. For example, purity is one of the most popular measures of cluster algorithms – evaluates the extent to which clusters contain a single class.

For a data set having 'n' data instances and 'c' known class labels which generates 'k' clusters, purity is measured as:

$$\text{Purity} = \frac{1}{n}\sum_{k}\max(k \cap c)$$

IMPROVING PERFORMANCE OF A MODEL

Now we have almost reached the end of the journey of building learning models. We have got some idea about what modelling is, how to approach about it to solve a learning problem and how to measure the success of our model. Now comes a million dollar question. Can we improve the performance of our model?

If so, then what are the levers for improving the performance? In fact, even before that comes the question of model selection – which model should be selected for which machine learning task? We have already discussed earlier that the model selection is done one several aspects:

1. Type of learning the task in hand, i.e. supervised or unsupervised

2. Type of the data, i.e. categorical or numeric

3. Sometimes on the problem domain

4. Above all, experience in working with different models to solve problems of diverse domains

So, assuming that the model selection is done, what are the different avenues to improve the performance of models?

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:

- Build a number of models based on the training data
- For diversifying the models generated, the training data subset can be varied using the allocation function. Sampling techniques like bootstrapping may be used to generate unique training data sets.
- Alternatively, the same training data may be used but the models combined are quite varying, e.g, SVM, neural network, kNN, etc.
- The outputs from the different models are combined using a combination function. A very simple strategy of combining, say in case of a prediction task using ensemble, can be majority voting of the different models combined. For example, 3 out of 5 classes predict 'win' and 2 predict 'loss' – then the final outcome of the ensemble using majority vote would be a 'win'.

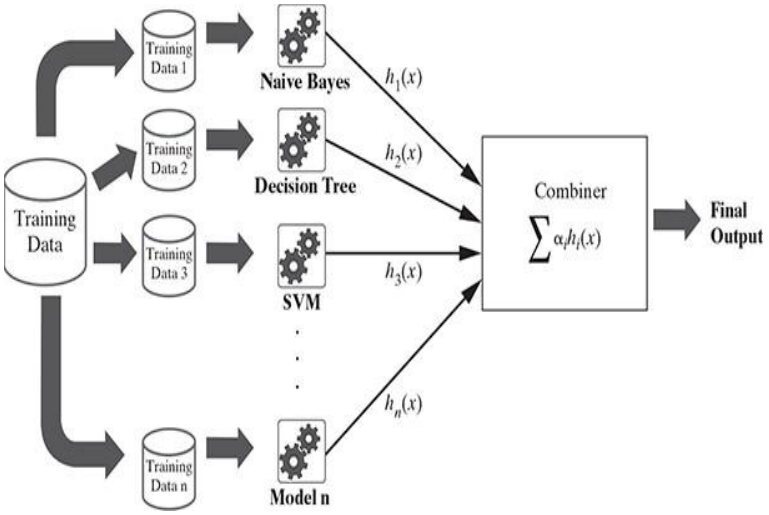


FIG. 3.11 Ensemble

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 ensemblebased 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

Random forest is another ensemble-based technique. It is an ensemble of decision trees – hence the name random forest to indicate a forest of decision trees.

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.

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:

- feature construction
- 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.

  Unlike feature transformation, in case of feature subset selection (or simply feature selection) no new feature is generated. The objective of feature selection is to derive a subset of features from the full feature set which is most meaningful in the context of a specific machine learning problem. So, essentially the job of feature selection is to derive a subset F (F , F , …, F) of F (F , F , …, F), where m < n, such that F is most meaningful and gets the best result for a machine learning problem. We will discuss these concepts in detail in the next section.

FEATURE TRANSFORMATION

Engineering a good feature space is a crucial prerequisite for the success of any machine learning model. However, often it is not clear which feature is more important. For that reason, all available attributes of the data set are used as features and the problem of identifying the important features is left to the learning model.

FEATURE CONSTRUCTION

Feature construction involves transforming a given set of input features to generate a new set of more powerful features. To understand more clearly, let's take the example of a real estate data set having details of all apartments sold in a specific region.

The data set has three features – apartment length, apartment breadth, and price of the apartment. If it is used as an input to a regression problem, such data can be training data for the regression model.

apartment_ length	apartment_ breadth	apartment_ price
80	59	23,60,000
54	45	12,15,000
78	56	21,84,000
63	63	19,84,000
83	74	30,71,000
92	86	39,56,000

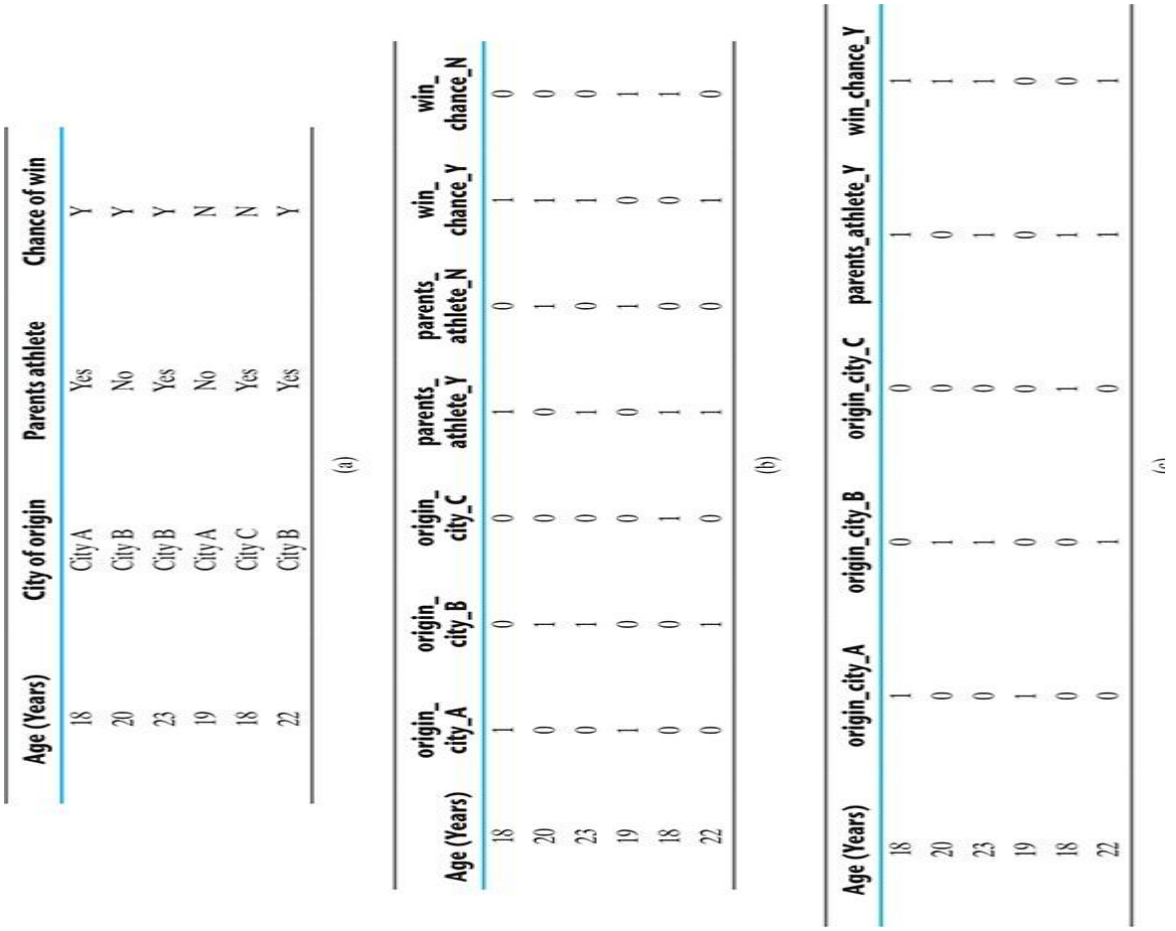
→

apartment_ length	apartment_ breadth	apartment_ area	apartment_ price
80	59	4,720	23,60,000
54	45	2,430	12,15,000
78	56	4,368	21,84,000
63	63	3,969	19,84,500
83	74	6,142	30,71,000
92	86	7912	39,56,000

FIG. 4.2 Feature construction (example 1)

- **Encoding Categorical (nominal) variables**

Let's take the example of another data set on athletes, as presented in Figure 4.3a. Say the data set has features age, city of origin, parents athlete (i.e. indicate whether any one of the parents was an athlete) and Chance of Win. The feature chance of a win is a class variable while the others are predictor variables. We know that any machine learning algorithm, whether it's a classification algorithm (like kNN) or a regression algorithm, requires numerical figures to learn from. So there are three features – City of origin, Parents athlete, and Chance of win, which are categorical in nature and cannot be used by any machine learning task.



Age (Years)	City of origin	Parents athlete	Chance of win
18	City A	Yes	Y
20	City B	No	Y
23	City B	Yes	Y
19	City A	No	N
18	City C	Yes	N
22	City B	Yes	Y

(a)

Age (Years)	origin_ city_A	origin_ city_B	origin_ city_C	parents_ athlete_Y	parents_ athlete_N	win_ chance_Y	win_ chance_N
18	1	0	0	1	0	1	0
20	0	1	0	0	1	1	0
23	0	1	0	1	0	1	0
19	1	0	0	0	1	0	1
18	0	0	1	1	0	0	1
22	0	1	0	1	0	1	0

(b)

Age (Years)	origin_city_A	origin_city_B	origin_city_C	parents_athlete_Y	win_chance_Y
18	1	0	0	1	1
20	0	1	0	0	1
23	0	1	0	1	1
19	1	0	0	0	0
18	0	0	1	1	0
22	0	1	0	1	1

FIG. 4.3 Feature construction (encoding nominal variables)

- **Encoding Categorical (ordinal) Variables**

Let's take an example of a student data set. Let's assume that there are three variable – science marks, maths marks and grade as shown in Figure 4.4a. As we can see, the grade is an ordinal variable with values A, B, C, and D. To transform this variable to a numeric variable, we can create a feature num_grade mapping a numeric value against each ordinal value. In the context of the current example, grades A, B, C, and D in Figure 4.4a is mapped to values 1, 2, 3, and 4 in the transformed variable shown in Figure 4.4b.

marks_science	marks_maths	Grade
78	75	B
56	62	C
87	90	A
91	95	A
45	42	D
62	57	B

(a)

marks_science	marks_maths	num_grade
78	75	2
56	62	3
87	90	1
91	95	1
45	42	4
62	57	2

(b)

FIG. 4.4 Feature construction (encoding ordinal variables)

- **Transforming Numeric (continuous) Features To Categorical Features**

Sometimes there is a need of transforming a continuous numerical variable into a categorical variable. For example, we may want to treat the real estate price prediction problem, which is a regression problem, as a real estate price category prediction, which is a classification problem. In that case, we can 'bin' the numerical data into multiple categories based on the data range. In the context of the real estate price prediction example, the original data set has a numerical feature apartment_price as shown in Figure 4.5a. It can be transformed to a categorical variable price-grade either as shown in Figure 4.5b or as shown in Figure 4.5c.

apartment_area	apartment_price
4,720	23,60,000
2,430	12,15,000
4,368	21,84,000
3,969	19,84,500
6,142	30,71,000
7,912	39,56,000

(a)

apartment_area	apartment_grade
4,720	Medium
2,430	Low
4,368	Medium
3,969	Low
6,142	High
7,912	High

(b)

apartment_area	apartment_grade
4,720	2
2,430	1
4,368	2
3,969	1
6,142	3
7,912	3

(c)

FIG. 4.5 Feature construction (numeric to categorical)

- **Text-Specific Feature Construction**

In the first place, the text data chunks that we can think about do not have readily available features, like structured data sets, on which machine learning tasks can be executed. All machine learning models need numerical data as input. So the text data in the data sets need to be transformed into numerical features. Text data, or corpus which is the more popular keyword, is converted to a numerical representation following a process is known as vectorization. In this process, word

occurrences in all documents belonging to the corpus are consolidated in the form of bag-of-words. There are three major steps that are followed:

1. tokenize
2. count
3. normalize

In order to tokenize a corpus, the blank spaces and punctuations are used as delimiters to separate out the words, or tokens. Then the number of occurrences of each token is counted, for each document. Lastly, tokens are weighted with reducing importance when they occur in the majority of the documents. A matrix is then formed with each token representing a column and a specific document of the corpus representing each row. Each cell contains the count of occurrence of the token in a specific document. This matrix is known as a document-term matrix (also known as a term-document matrix). Figure 4.6 represents a typical document-term matrix which forms an input to a machine learning model.

This	House	Build	Feeling	Well	Theatre	Movie	Good	Lonely	...
2	1	1	0	0	1	1	1	0	
0	0	0	1	1	0	0	0	0	
1	0	0	2	1	1	0	0	1	
0	0	0	0	1	0	1	1	0	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	

FIG. 4.6 Feature construction (text-specific)

FEATURE EXTRACTION

In feature extraction, new features are created from a combination of original features. Some of the commonly used operators for combining the original features include

1. For Boolean features: Conjunctions, Disjunctions, Negation, etc.

2. For nominal features: Cartesian product, M of N, etc.

3. For numerical features: Min, Max, Addition, Subtraction, Multiplication, Division, Average Equivalence, Inequality, etc.

Let's take an example and try to understand. Say, we have a data set with a feature set F (F , F , …, F). After feature extraction using a mapping function f

(F , F , …, F) say, we will have a set of features $\acute{F}_i(\acute{F}_1, \acute{F}_2, \ldots, \acute{F}_m)$

such that $\acute{F}_i = f(F_i)$ and m < n.

For example ,. This is depicted in Figure4.7

$Feat_A$	$Feat_B$	$Feat_C$	$Feat_D$
34	34.5	23	233
44	45.56	11	3.44
78	22.59	21	4.5
22	65.22	11	322.3
22	33.8	355	45.2
11	122.32	63	23.2

→

$Feat_1$	$Feat_2$
41.25	185.80
54.20	53.12
43.73	35.79
65.30	264.10
37.02	238.42
113.39	167.74

$$Feat_1 = 0.3 \times Feat_A + 0.9 \times Feat_A$$
$$Feat_2 = Feat_A + 0.5\, Feat_B + 0.6 \times Feat_C$$

FIG. 4.7 Feature extraction