**MCA: 304A Data Warehousing and Data**

**Mining**

Unit 4

Classification

Methods &

improving

accuracy of classification

Prepared by

S NOORTAJ

Asst. professor

KMMIPS

TIRUPATHI

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

**Classifiers Of Machine Learning:**

1. Decision Trees
2. Rule Based Classifiers
3. Bayesian Classifiers 4. K-Nearest Neighbour **Advantages:**
* Mining Based Methods are cost-effective and efficient
* Helps in identifying criminal suspects
* Helps in predicting the risk of diseases
* Helps Banks and Financial Institutions to identify defaulters so that they may approve Cards, Loan, etc.

**Disadvantages:**

Privacy: When the data is either are chances that a company may give some information about their customers to other vendors or use this information for their profit.

Accuracy Problem: Selection of Accurate model must be there in order to get the best accuracy and result.

**Applications:**

* Marketing and Retailing
* Manufacturing
* Telecommunication Industry
* Intrusion Detection
* Education System
* Fraud Detection

# General approach to solve classification problem

**Data classification** is a two-step process, consisting of a *learning step* (where a classification model is constructed) and a *classification step* (where the model is used to predict class labels for given data). The process is shown for the loan application data of below figure (The data are simplified for illustrative purposes. In reality, we may expect many more attributes to be considered.



The data classification process: (a) *Learning*: Training data are analyzed by a classification algorithm. Here, the class label attribute is *loan decision*, and the learned model or classifier is represented in the form of classification rules.



(b) *Classification*: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

In the first step (a), a classifier is built describing a predetermined set of data classes or concepts.

* This is the **learning step** (or training phase), where a classification algorithm builds the classifier by analyzing or “learning from” a **training set** made up of database tuples and their associated class labels.
* A tuple, ***X***, is represented by an *n*-dimensional **attribute vector**, ***X*** D .*x*1, *x*2, : : : , *xn*/, depicting *n* measurements made on the tuple from *n* database attributes, respectively, *A*1, *A*2, : : : , *An*.
* Each tuple, ***X***, is assumed to belong to a predefined class as determined by another database attribute called the **class** **label attribute**.
* The class label attribute is discrete-valued and unordered.
* It is *categorical* (or nominal) in that each value serves as a category or class.
* The individual tuples making up the training set are referred to as **training tuples** and are randomly sampled from the database under analysis.
* In the context of classification, data tuples can be referred to as *samples, examples, instances, data points*, or *objects*.
* Because the class label of each training tuple *is provided*, this step is also known as **supervised learning** (i.e., the learning of the classifier is “supervised” in that it is told to which class each training tuple belongs).
* It contrasts with **unsupervised learning** (or **clustering**), in which the class label of each training tuple is not known, and the number or set of classes to be learned may not be known in advance. For example, if we did not have the *loan decision* data available for the training set, we could use clustering to try to determine “groups of like tuples,” which may correspond to risk groups within the loan application data.

 the second step (b), the model is used for classification.

* First, the predictive accuracy of the classifier is estimated. If we were to use the training set to measure the classifier’s accuracy, this estimate would likely be optimistic, because the classifier tends to **overfit** the data (i.e., during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall).
* Therefore, a **test set** is used, made up of **test tuples** and their associated class labels. They are independent of the training tuples, meaning that they were not used to construct the classifier.

Decision Trees:

**Decision tree algorithm** falls under the category of [supervised learning.](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) They can be used to solve both **regression** and **classification problems**. Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.



 Example for a decision tree

**Tree Pruning**

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

There are two approaches to prune a tree −

* **Pre-pruning** − The tree is pruned by halting its construction early.
* **Post-pruning** - This approach removes a sub-tree from a fully grown tree.

*Decision Tree Induction Algorithm*

A machine researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach. In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner.

**ID3 in brief**

ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step.

Invented by [Ross Quinlan,](https://en.wikipedia.org/wiki/Ross_Quinlan) ID3 uses a **top-down greedy** approach to build a decision tree. In simple words, the **top-down** approach means that we start building the tree from the top and the **greedy** approach means that at each iteration we select the best feature at the present moment to create a node. Most generally ID3 is only used for classification problems with [nominal](https://corporatefinanceinstitute.com/resources/knowledge/other/nominal-data/) features only.

**Metrics in ID3:**

**Entropy** is the measure of disorderand the Entropy of a dataset is the measure of disorderin the target feature of the dataset. In the case of binary classification (where the target column has only two types of classes) entropy is **0** if all valuesin the target column are homogenous(similar) and will be **1** if the target column has equal number values for both the classes.

**Entropy(S) = - ∑ pᵢ \* log**₂**(pᵢ) ; i = 1 to n**

Information Gain calculates the reduction in the entropy and measures how well a given feature separates or classifies the target classes. The feature with the **highest Information Gain** is selected as the **best** one.

**IG(S, A) = Entropy(S) - ∑((|Sᵥ| / |S|) \* Entropy(Sᵥ))**

**ID3 Steps**

1. Calculate the Information Gain of each feature.
2. Considering that all rows don’t belong to the same class, split the dataset **S** into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

using a sample dataset of COVID-19 infection. A preview of the entire dataset is shown below.

**| ID | Fever | Cough | Breathing issues | Infected |**

| 1 | NO | NO | NO | NO |

| 2 | YES | YES | YES | YES |

| 3 | YES | YES | NO | NO |

| 4 | YES | NO | YES | YES |

| 5 | YES | YES | YES | YES |

| 6 | NO | YES | NO | NO |

| 7 | YES | NO | YES | YES |

| 8 | YES | NO | YES | YES | | 9 | NO | YES | YES | YES | | 10 | YES | YES | NO | YES |

| 11 | NO | YES | NO | NO |

| 12 | NO | YES | YES | YES |

| 14 | YES | YES | NO | NO |

**Implementation on our Dataset**

From the total of 14 rows in our dataset **S**, there are **8** rows with the target value **YES** and **6** rows with the target value **NO**. The entropy of **S** is calculated as:

**Entropy(S) = — (8/14) \* log₂(8/14) — (6/14) \* log₂(6/14) = 0.99** We now calculate the Information Gain for each feature:

**IG calculation for Fever:**

In this(Fever) feature there are **8** rows having value **YES** and **6** rows having value **NO.**

As shown below, in the **8** rows with **YES for** Fever, there are **6** rows having target

value **YES** and **2** rows having target value **NO.**

|  |
| --- |
| **| Fever | Cough | Breathing issues | Infected |** |
| | YES | YES | YES | YES  |

|

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | YES  | |  | YES  | |  | NO  | |  | NO  | | |
| |  | YES  | |  | NO  | |  | YES  | |  | YES  | | |
| |  | YES  | |  | YES  | |  | YES  | |  | YES  | | |
| |  | YES  | |  | NO  | |  | YES  | |  | YES  | | |
| |  | YES  | |  | NO  | |  | YES  | |  | YES  | | |
| |  | YES  | |  | YES  | |  | NO  | |  | YES  | | |
| |  | YES  | |  | YES  | |  | NO  | |  | NO  | | |

As shown below, in the **6** rows with **NO**, there are **2** rows having target

|  |
| --- |
| **| Fever | Cough | Breathing issues | Infected |** | NO | NO | NO | NO || NO | YES | NO | NO || NO | YES | YES | YES || NO | YES | NO | NO || NO | YES | YES | YES || NO | YES | YES | NO | |
| The block, below, demonstrates the calculation of Information Gain for **Fever.**  |
| # total rows|S| = 14For v = YES, |Sᵥ| = 8Entropy(Sᵥ) = - (6/8) \* log₂(6/8) - (2/8) \* log₂(2/8) = 0.81For v = NO, |Sᵥ| = 6Entropy(Sᵥ) = - (2/6) \* log₂(2/6) - (4/6) \* log₂(4/6) = 0.91 # Expanding the summation in the IG formula:IG(S, Fever) = Entropy(S) - (|Sʏᴇꜱ| / |S|) \* Entropy(Sʏᴇꜱ) - (|Sɴᴏ| / |S|) \* Entropy(Sɴᴏ)∴ IG(S, Fever) = 0.99 - (8/14) \* 0.81 - (6/14) \* 0.91 = 0.13  |
| calculate the IG for the features “Cough” and “Breathing issues”. |   |

**IG(S, Cough) = 0.04 IG(S, BreathingIssues) = 0.40**

the feature **Breathing issues** have the highest Information Gain it is used to create the root node. Here , after this initial step our tree look like this:

from the remaining two unused features, namely, **Fever** and **Cough**, we decide which one is the best for the left branch of **Breathing Issues**.

Since the left branch of **Breathing Issues** denotes **YES,** we will work with the subset of the original data i.e the set of rows having **YES** as the value in the Breathing Issues column.These **8 rows** are shown below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **|**  | **Fever**  |  | **|**  |  | **Cough**  | **|Breathing**  | **issues|**  | **Infected**  | **|** |
|   |  |  |  |  |  |  |  |  |  |
| |  | YES  |  | |  |  | YES  | | YES  | |  | YES  | | |
| |  | YES  |  | |  |  | NO  | | YES  | |  | YES  | | |
| |  | YES  |  | |  |  | YES  | | YES  | |  | YES  | | |
| |  | YES  |  | |  |  | NO  | | YES  | |  | YES  | | |
| |  | YES  |  | |  |  | NO  | | YES  | |  | YES  | | |
| |  | NO  |  | |  |  | YES  | | YES  | |  | YES  | | |
| |  | NO  |  | |  |  | YES  | | YES  | |  | YES  | | |
| |  | NO  |  | |  |  | YES  | | YES  | |  | NO  | | |

Next, we calculate the IG for the features Fever and Cough using the subset **S**ʙʏ (**S**et **Breathing** Issues **Y**es) which is shown above :

*Note: For* ***IG*** *calculation the Entropy will be calculated from the subset* ***S****ʙʏ and not the original dataset* ***S.***

**IG(Sʙʏ, Fever) = 0.20**

**IG(Sʙʏ, Cough) = 0.09**

IG of Fever is greater than that of Cough, so we select **Fever** as the left branch of

Breathing Issues:

Our tree now looks like this:



Next find the feature with the maximum IG for the right branch of **Breathing Issues**. But, since there is only one unused feature left we have no other choice but to make it the

right branch of the root node. So our tree now looks like this:

There are no more unused features, so we stop here and jump to the final step of creating the leaf nodes. For the left leaf node of Fever, we see the subset of rows from the original data set that has **Breathing Issues** and **Fever** both values as **YES**.

**| Fever | Cough | Breathing issues | Infected |**

| YES | YES | YES | YES |

| YES | NO | YES | YES |

| YES | YES | YES | YES |

| YES | NO | YES | YES | | YES | NO | YES | YES |

Since all the values in the target column are **YES,** we label the left leaf node as **YES**, but to make it more logical we label it **Infected.** Similarly, for the right node of Fever we see the subset of rows from the original data set that have **Breathing Issues** value as **YES** and **Fever** as **NO**.

**| Fever | Cough | Breathing issues | Infected |**

| NO | YES | YES | YES |

| NO | YES | YES | NO |

| NO | YES | YES | NO |

Here not all but **most** of the **values** are **NO,** hence **NO** or **Not Infected** becomes our **right leaf node.**

Our tree, now, looks like this:



We repeat the same process for the node **Cough**, however here both left and right leaves turn out to be the same i.e. **NO** or **Not Infected** as shown below:



**C4.5:**

The decision tree algorithm C4.5 improves ID3 in the following ways:

* **Missing data:** When the decision tree is built, missing data are simply ignored. That is, the gain ratio is calculated by looking only at the other records that have a value for that attribute. To classify a record with a missing attribute value, the value for that item can be predicted based on what is known about the attribute values for the other records.
* **Continuous data:** The basic idea is to divide the data into ranges based on the attribute values for that item that are found the training sample.
* **Pruning:** There are two primary pruning strategies proposed in C4.5:
	+ With sub tree replacement, a sub tree is replaced by a leaf node if this replacement results in an error rate close to that of the original tree. Sub tree replacement works from the bottom of the tree up to the root.
	+ Another pruning strategy, called sub tree raising, replaces a sub tree by its most used sub tree. Here a sub tree is raised from its current location to a node higher up in the tree. Again, we must determine the increase in error rate for this replacement.

* **Rules:** C4.5 allows classification via either decision trees or rules generated from them. In addition, some techniques to simplify complex rules are proposed. One approach is to replace the left-hand side of a rule by a simpler version if all records the training set are treated identically. An "otherwise" type of rule can be used to indicate what should be done if no other rules apply.
* **Splitting:** The ID3 approach favors attributes with many divisions and thus may lead to overfitting. In the extreme, an attribute that has a unique value for each tuple in the training set would be the best because there would be only one tuple (and thus one class) for each division.

An improvement can be made by taking into account the cardinality of each division. This approach uses the GainRatio as opposed to Gain. The GainRatio is defined as:



**CART( Classification And Regression Tree)** is a variation of the decision tree algorithm. It can handle both [classification and regression](https://www.geeksforgeeks.org/ml-classification-vs-regression/) tasks. [Scikit-Learn](https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/) uses the Classification And Regression Tree (CART) algorithm to train  [Decision Trees](https://www.geeksforgeeks.org/decision-tree/) (also called “growing” trees). CART was first produced by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984.

**CART Algorithm**

CART is a predictive algorithm used in [Machine learning](https://www.geeksforgeeks.org/machine-learning/) and it explains how the target variable’s values can be predicted based on other matters. It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.

In the decision tree, nodes are split into sub-nodes on the basis of a threshold value of an attribute. The root node is taken as the training set and is split into two by considering the best attribute and threshold value. Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree.

The CART algorithm works via the following process:

* The best split point of each input is obtained.
* Based on the best split points of each input in Step 1, the new “best” split point is identified.
* Split the chosen input according to the “best” split point.
* Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



* + CART algorithm uses Gini Impurity to split the dataset into a decision tree .
	+ It does that by searching for the best homogeneity for the sub nodes, with the help of the Gini index criterion.

**Gini index/Gini impurity**

The Gini index is a metric for the classification tasks in CART. It stores the sum of squared probabilities of each class.

* + It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient.
	+ It works on categorical variables, provides outcomes either “successful” or “failure” and hence conducts binary splitting only.

The degree of the Gini index varies from 0 to 1,

* Where 0 depicts that all the elements are allied to a certain class, or only one class exists there.
* The Gini index of value 1 signifies that all the elements are randomly distributed across various classes, and
* A value of 0.5 denotes the elements are uniformly distributed into some classes. Mathematically, we can write Gini Impurity as follows:



where pi is the probability of an object being classified to a particular class.

**Classification tree**

A classification tree is an algorithm where the target variable is categorical.

* The algorithm is then used to identify the “Class” within which the target variable is most likely to fall.
* Classification trees are used when the dataset needs to be split into classes that belong to the response variable(like yes or no)

**Regression tree**

* A Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value.
* Regression trees are used when the response variable is continuous. For example, if the response variable is the temperature of the day.

# Rule based classifiers

A rule-based classifier consists of a set of “IF-THEN” rules obtained by statistically apprehending the training data.

* Each rule of the classifier consists of an antecedent and a consequent. The antecedent part contains one or more terms, where each term is comprised of a variable name, an operator, and a value.
* In the cases where an antecedent contains more than one term, the terms are joined by the “AND” conjunction.
* On the other hand, the consequent part of the rule represents the class label associated with the rule.

IF-THEN Rules

Rule-based classifier makes use of a set of IF-THEN rules for classification. We can express a rule in the following from −

IF condition THEN conclusion

IF age = youth AND student = yes THEN buy\_computer = yes

**Points to remember −**

* The IF part of the rule is called **rule antecedent** or **precondition**.
* The THEN part of the rule is called **rule consequent**.
* The antecedent part the condition consist of one or more attribute tests and these tests are logically ANDed.
* The consequent part consists of class prediction.

**Note** − We can also write rule R1 as follows −

R1: (age = youth) ^ (student = yes))(buys computer = yes) Rule Extraction how to build a rule-based classifier by extracting IF-THEN rules from a decision tree.

**Points to remember −**

To extract a rule from a decision tree −

* One rule is created for each path from the root to the leaf node.
* To form a rule antecedent, each splitting criterion is logically ANDed.
* The leaf node holds the class prediction, forming the rule consequent.

Nearest Neighbour Classifiers o K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors** o **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum. o **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:
* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.

Consider the below image:



* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:
* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.



* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.
* There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
* A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
* Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

* It is simple to implement.
* It is robust to the noisy training data o It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

* Always needs to determine the value of K which may be complex some time.
* The computation cost is high because of calculating the distance between the data points for all the training samples.

# Bayesian classifiers

Bayesian classification is based on Bayes' Theorem. Bayesian classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Bayesian classification uses Bayes theorem to predict the occurrence of any event. Bayesian classifiers are the statistical classifiers with the Bayesian probability understandings. The theory expresses how a level of belief, expressed as a probability.

Bayes Theorem Formula

If A and B are two events, then the **formula for the Bayes theorem** is given by:

Where P(A|B) is the probability of condition when event A is occurring while event B has already occurred.

Bayes Theorem Derivation

Bayes Theorem can be derived for events and [random variables](https://byjus.com/maths/random-variable/) separately using the definition of conditional probability and density.

From the definition of conditional probability, Bayes theorem can be derived for events as given below:

P(A|B) = P(A  B)/ P(B), where P(B) ≠ 0

P(B|A) = P(B  A)/ P(A), where P(A) ≠ 0

Here, the joint probability P(A  B) of both events A and B being true such that,

P(B  A) = P(A  B)

P(A  B) = P(A | B) P(B) = P(B | A) P(A)

P(A|B) = [P(B|A) P(A)]/ P(B), where P(B) ≠ 0

Bayes theorem can be derived for n events



**Example 1:**

 Bag1 contains 4 white and 8 black balls and Bag2 contains 5 white and 3 black balls. From one of the bag one ball is drawn at random and the ball which is drawn comes out as black. Find the probability that the ball is drawn from Bag1.

**Solution:**

Given,

Let E1, E2 and A be the three events where,

E1 = Event of selecting Bag1

E2 = Event of selecting Bag2

A = Event of drawing black ball

Now,

P(E1) = P(E2) = 1/2

P(drawing a black ball from Bag1) = P(A|E1) = 8/12 = 2/3

P(drawing a black ball from Bag2) = P(A|E2) = 3/8

By using Bayes' Theorem, the probability of drawing a black ball from Bag1,

P(E1|A) = P(A|E1) \* P(E1) / P(A|E1) \* P(E1) + P(A|E2) \* P(E2)

 **[**P(A|E1) \* P(E1) + P(A|E2) \* P(E2) = Total Probability**]**

= (2/3 \* 1/2) / (2/3 \* 1/2 + 3/8 \* 1/2)

 = 16/25

Hence, the probability that the ball is drawn from Bag1 is 16/25

 **Example 2:**

A bag I contains 4 white and 6 black balls while another Bag II contains 4 white and 3 black balls. One ball is drawn at random from one of the bags, and it is found to be black. Find the probability that it was drawn from Bag I.

**Solution:**

Let E1 = event of choosing the bag 1, E2 = event of choosing the bag 2.

Let A be event of drawing a black ball.

P(E1) = P(E2) = 1/2.

 Also, P(A|E1) = P(drawing a black ball from Bag 1) = 6/10 = 3/5.

P(A|E2) = P(drawing a black ball from Bag 2) = 3/7.

By using Bayes’ theorem, the probability of drawing a black ball from bag 1 out of two bags is-:

P(E1 | A) = P(E1)P(A | E1)/( P(E1)P(A│E1)+P(E2)P(A | E2)) = (1/2 × 3/5) / ((1/2 × 3/7)) + (1/2 × 3/5)) = 7/12.

**Hypothesis Testing:**

Hypothesis testing attempts to find a model that explains the observed data by first creating a hypothesis and then testing the hypothesis against the data. the hypothesis holds for the sample, it is assumed to hold for the population in general. Given a population, the initial (assumed) hypothesis to be tested, Ho, is called the null hypothesis. Rejection of the null hypothesis causes another hypothesis, Ht , called the alternative hypothesis, to be made.

One technique to perform hypothesis testing is based on the use of the chi-squared statistic. Actually, there is a set of procedures referred to as chi squared.

* These procedures can be used to test the association between two observed variable values and to determine if a set of observed variable values is statistically significant (i.e., if it differs from the expected case).
* A hypothesis is first made, and then the observed values are compared based on this hypothesis .
* Assuming that 0 represents the observed data and E is the expected values based on the hypothesis, the chi-squared statistic, x 2, is defined as:



**Example :**

Suppose that there are five schools being compared based on students' results on a set of standardized achievement tests. The school district expects that the results will be the same for each school. They know that the total score for the schools is 375, so the expected result would be that each school has an average score of 75. The actual average scores from the schools are: 50, 93, 67, 78, and 87. The district administrators want to determine if this is statistically significant. Or in simpler te1ms, should they be worried about the distribution of scores. The chi-squared measure here is

**Sol:**  Given O=50, 93, 67, 78, and 87

 Average score (E)= 75

X2=(50 - 75)2/75 + (93 - 75)2 /75+ (67 - 75)2/75 + (78 - 75)2/75+ (87 - 75)2/75

X2=15.55

Estimating Predictive accuracy of classification methods:

Holdout Method and Random Subsampling: The holdout method is what we have alluded to so far in our discussions about accuracy. In this method, the given data are randomly partitioned into two independent sets, a training set and a test set.

* Typically, two-thirds of the data are allocated to the training set,  one-third is allocated to the test set.
* The training set is used to derive the model. The estimate is pessimistic because only a portion of the initial data is used to derive the model.



* Random subsampling is a variation of the holdout method in which the holdout method is repeated k times. The overall accuracy estimate is taken as the average of the accuracies obtained from each iteration.

Cross-Validation: In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or “folds,” D1, D2,..., Dk , each of approximately equal size. Training and testing is performed k times. In iteration i, partition Di is reserved as the test set, and the remaining partitions are collectively used to train the model.

* That is, in the first iteration, subsets D2,..., Dk collectively serve as the training set to obtain a first model, which is tested on D1;
* the second iteration is trained on subsets D1, D3,..., Dk and tested on D2; and so on. Unlike the holdout and random subsampling methods, here each sample is used the same number of times for training and once for testing.
* For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.

Bootstrap: Unlike the accuracy estimation methods just mentioned, the bootstrap method samples the given training tuples uniformly with replacement.

* That is, each time a tuple is selected, it is equally likely to be selected again and re-added to the training set. For instance, imagine a machine that randomly selects tuples for our training set. In sampling with replacement, the machine is allowed to select the same tuple more than once.
* There are several bootstrap methods. A commonly used one is the .632 bootstrap, which works as follows. Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a bootstrap sample or training set of d samples. It is very likely that some of the original data tuples will occur more than once in this sample. The data tuples that did not make it into the training set end up forming the test set. Suppose we were to try this out several times. As it turns out, on average, 63.2% of the original data tuples will end up in the bootstrap sample, and the remaining 36.8% will form the test set (hence, the name, .632 bootstrap).

# Improving accuracy of classification methods

Bagging: Given a set, D, of d tuples, bagging works as follows. For iteration i(i = 1, 2,..., k), a training set, Di , of d tuples is sampled with replacement from the original set of tuples, D.

* Note that the term bagging stands for bootstrap aggregation. Each training set is a bootstrap sample. Because sampling with replacement is used, some of the original tuples of D may not be included in Di , whereas others may occur more than once.

Boosting : In boosting, weights are also assigned to each training tuple.

* A series of k classifiers is iteratively learned. After a classifier, Mi , is learned, the weights are updated to allow the subsequent classifier, Mi+1, to “pay more attention” to the training tuples that were misclassified by Mi .
* The final boosted classifier, M, combines the votes of each individual classifier, where the weight of each classifier’s vote is a function of its accuracy.

suppose that as a patient, you have certain symptoms. Instead of consulting one doctor, you choose to consult several. Suppose you assign weights to the value or worth of each doctor’s diagnosis, based on the accuracies of previous diagnoses they have made. The final diagnosis is then a combination of the weighted diagnoses. This is the essence behind boosting.

Random Forests: Imagine that each of the classifiers in the ensemble is a decision tree classifier so that the collection of classifiers is a “forest.” The individual decision trees are generated using a random selection of attributes at each node to determine the split.

# Evaluation criteria for classification methods

* The classifier evaluation measures presented in this section are summarized in below Figure. They include accuracy (also known as recognition rate), sensitivity (or recall), specificity, precision, *F*1, and *F.*
* Using training data to derive a classifier and then estimate the accuracy of the resulting learned model can result in misleading overoptimistic estimates due to overspecialization of the learning algorithm to the data
* Instead, it is better to measure the classifier’s accuracy on a *test set* consisting of class-labeled tuples that were not used to train the model.
* Before we discuss the various measures, we need to become comfortable with some terminology. Recall that we can talk in terms of **positive tuples** (tuples of the main class of interest) and **negative tuples** (all other tuples).



Evaluation measures

**True positives** .*TP*/: These refer to the positive tuples that were correctly labeled by the classifier. Let *TP* be the number of true positives.

**True negatives** .*TN*/: These are the negative tuples that were correctly labeled by the classifier. Let *TN* be the number of true negatives.

**False positives** .*FP*/: These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class *buys computer* D *no* for which the classifier predicted *buys computer* D *yes*). Let *FP* be the number of false positives.

**False negatives** .*FN*/: These are the positive tuples that were mislabeled as negative

These terms are summarized in the **confusion matrix** of below figure.

The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes. *TP* and *TN* tell us when the classifier is getting things right, while *FP* and *FN* tell us when the classifier is getting things wrong.



Confusion matrix, shown with totals for positive and negative tuples