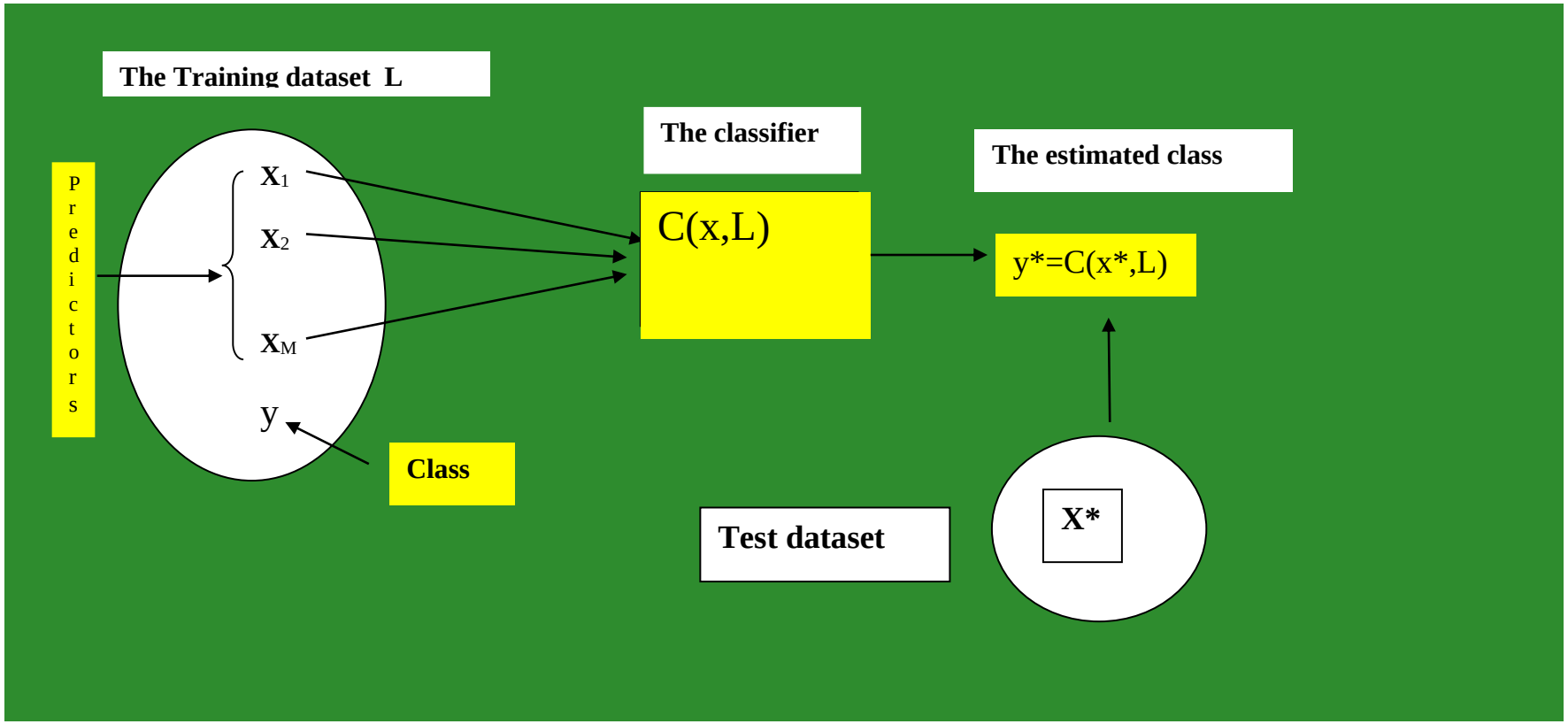
Data Mining and Machine Learning

LECTURE 8
Supervised classification

Dr. Edgar Acuna
Department of Mathematics

Universidad de Puerto Rico- Mayaguez
academic.uprm.edu/eacuna

The Supervised classification problem



Supervised Classification vs. Regression

- Supervised Classification:
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and uses it in classifying new data
- Regression:
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

Supervised Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting may occur

Supervised classification methods

1. Bayes Classifiers
   a. Linear Discriminant Analysis.
   b. Naive Bayes.
   c. k-nearest neighbors
   d. Logistic Regression
2. Decision Trees.
3. Ensembles (XGboosting).
4. Random Forest
5. Neural Networks and Deep Learning
6. Support vector machines.

Supervised classification from a Bayesian point of view

Suposse that we known beforehand the prior probabilities π_i (i=1, 2,…G) that an object belongs to the class C_i (most of the times these probabilities are taken as the class frequency relative). If no additional information is available then the best decision rule will classify a new object as belonging to the class C_i if

$$\pi_i > \pi_j \text{ for } i=1,2,.....G,\ i \neq j \qquad (3.1)$$

However, usually some addtional information is known, such as a vector of measurements x made on the object to be classified. In this case, we compare the probability of belonging to each class for an object with vector of measurements x and the object is classified as of class Ci if

$$P(Ci/\mathbf{x}) > P(Cj/\mathbf{x}) \text{ para todo } i \neq j \qquad (3.2)$$

This decision rule is called the **Bayes rule of minimum error**.

Notice that $i=\text{argmax}_k\ P(C_k/\mathbf{x})$ for all k in 1,2….G.

Supervised classification from a Bayesian point of view

Conditional Probability: Given A and B, two events of a same sample space. The Probability of A given that B occurs is given by:

$$P(A/B)=P(AyB)/P(B) \text{ assuming } P(B)>0$$

Product Rule for probabillties:

$$P(A \text{ and } B)=P(A)P(B/A) \text{ o } P(B)P(A/B)$$

Chain Rule for probabilities:

$$P(A \text{ and } B \text{ and} C)=P(A)P(B/A)P(C/A \text{ and } B)$$

Bayes Rule:

$$P(A/B)=[P(A)P(B/A)]/P(B)$$

P(A) is called the prior probability
P(A/B) is called the posterior probability

Supervised classification from a Bayesian point of view

In a supervised classification context:

A: event that any object is assigned to the class A

B: event that a object has the measurements given by a random vector x.

P(A) is estimated by the class' relative frequency.

P(B/A) represents the probability distribution of the random vector x in the class A.

P(B) represents the probability distribution of the random vector x regardless the class. It does not matter to know it, because it cancels out in the computations.

P(A/B) is the probability that a objecto is assigned to class A given that the object has the measurements x.

Bayesian Classification

The probabillities P(Ci/x) are called posterior probabilities. Unfortunately rarely the posterior probabiities are known and they must be estimated. This happens in knn classification and logistic regression.

A more convenient formulation of the rule (eq. 3.2) can be obtained by applying Bayes Theorem, which states that

$$P(Ci/\mathbf{x}) = \frac{f(\mathbf{x}/C_i)\pi_i}{f(\mathbf{x})} \qquad (3.3)$$

Therefore an object will be classified as of class C_i if

$$f(\mathbf{x}/C_i)\pi_i > f(\mathbf{x}/C_j)\pi_j \qquad (3.4)$$

para todo i $\neq$j. That is, x is cllasified as of class C_i such that i=argmax$_k$f(x/C_k)π_k

If the class conditional densities f(x/ Ci) are known then the classification problem is solved explicitly, like it occurs in both linear and quadratic discriminant.

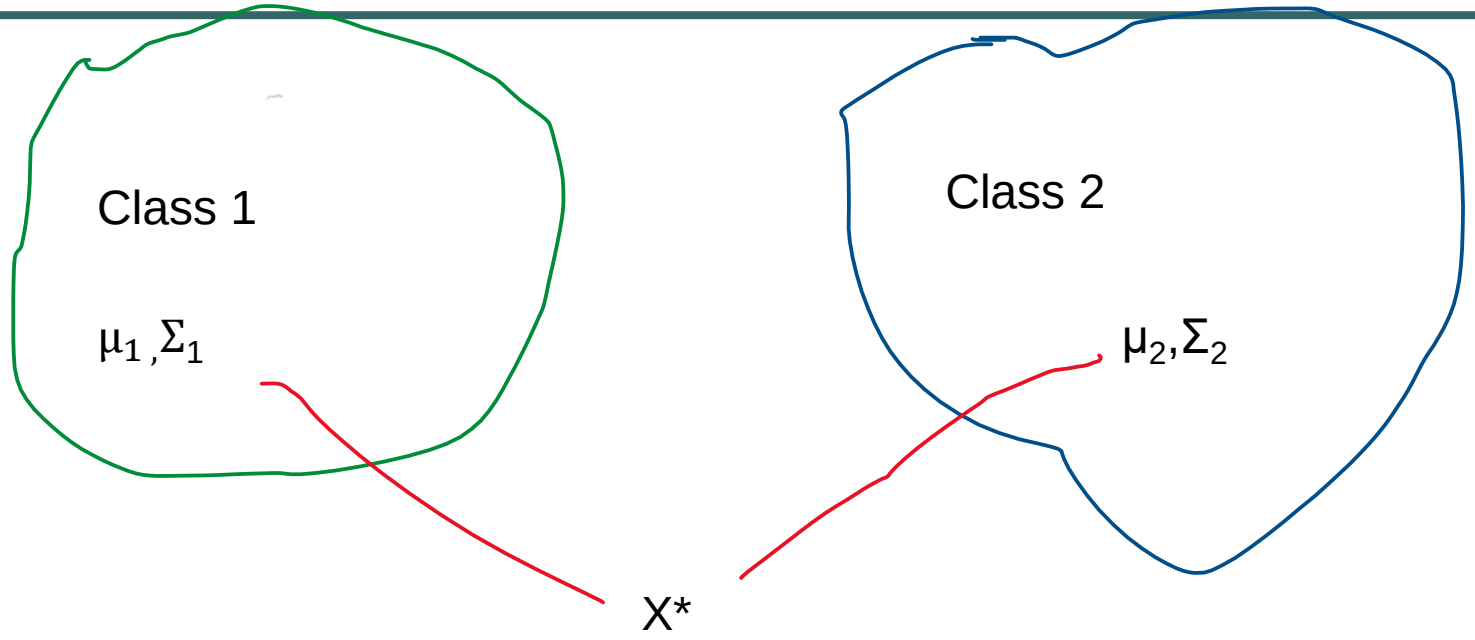
But, most of the time, the f(x/ Ci) are unknown and they must be estimated using the training dataset. This is the case of k-nn classifiers, kernel density classifiers and gaussian mixtures classifiers.

Linear Discriminant Analysis

Consider the following training sample with p features and two classes

Y	X_1	X_2	…	X_p
1	X_{11}	X_{21}	….	X_{p1}
1	X_{12}	X_{22}	…	X_{p2}
..	..	..	..	..
1	X_{1n1}	X_{2n1}	…	X_{pn1}
2	$X_{1,n1+1}$	$X_{2,n1+1}$	…	$X_{p,n1+1}$
2	$X_{1,n1+2}$	$X_{2,n1+2}$	…	$X_{p,n1+2}$
..	..	..	…	..
2	$X_{1,n1+n2}$	$X_{2,n1+n2}$	…	$X_{p,n1+n2}$

LDA from a geometric perspective



If Dist(x*,μ_1)<Dist(x*,μ_2) then x* is assigned to class 1.

Linear discriminant Analysis

Let us consider μ_1 and μ_2 as the mean vector of the respective class populations

Let us assume that both populations have the same covariance matrix, ie $\Sigma_1=\Sigma_2=\Sigma$. This is known as the homocedasticity property.

For now, we do not need to assume that the random vector of predictor **x**=(x1,…..xp) is normally distributed.

Linear discrimination is based on the following fact: An instance (object) x is assigned to the class C1 if

$$D(\mathbf{x}, C1)<D(\mathbf{x},C2) \quad (3.5)$$

where $D(\mathbf{x},Ci)=)=(\mathbf{x}-\boldsymbol{\mu}_i)'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$, for i=1,2, represents the ***squared Mahalanobis distance*** *of* ***x*** *to the center of the* C_i ***class.***

Linear Discriminant Analysis (cont)

The expression (3.5) can be written as

$$(\mu_1 - \mu_2)'\Sigma^{-1}[x - (1/2)(\mu_1 + \mu_2)] > 0 \quad (3.6)$$

Using the training sample, μ_i can be estimated by $\bar{X}_i$ (the mean vector of the p featuresij the o-th class), and Σ is estimated by S, the pooled covariance matrix, which is calculated by

$$S = \frac{(n_1 - 1)S_1 + (n_2 - 1)S_2}{n_1 + n_2 - 2}$$

where, S1 and S2 represent the sample covariance matrices of the random vector of predictors in each class. Therefore the sample version of (2.2) is given by

$$(\bar{X}_1 - \bar{X}_2)'S^{-1}[x - (1/2)(\bar{X}_1 + \bar{X}_2)] > 0 \quad (3.7)$$

The left hand side of expression (3.7) is called **the linear discriminant function.**

Example: Predicting the grade in a class

In a math class consisting of 32 students the following attributes are measured.

E1: Score in the first examen (0-100)
E2: Score in the second examen 2 (0-100)

Class: Asume the value P if the student pass the class and F if the student fail the class.
The first five instances are shown below:

	E1	E2	Class
1	96	100	p
2	96	94	p
3	100	91	p
4	93	96	p
5	90	94	p

The goal is to predict the final grade of a student that will take the class the next semester under the same conditions(same professor, same textbook, students of similar level , etc).

Example(cont)

```
#Finding the centroids(vector if means)  of each class
pasan=df[df['Nota']=="p"][["E1",'E2']]
pasan.mean()
E1 75.541667
E2 73.750000
fail=df[df['Nota']=="f"][["E1",'E2']]
fail.mean()
E1 59.5
E2 34.0
#Finding the pooled covariance
pcov=pasan.cov()
fcov=fail.cov()
npa=len(pasan)
nf=len(fail)
pool_cov=((npa-1)*pcov+(nf-1)*fcov)/(npa+nf-2.0)
pool_cov
```

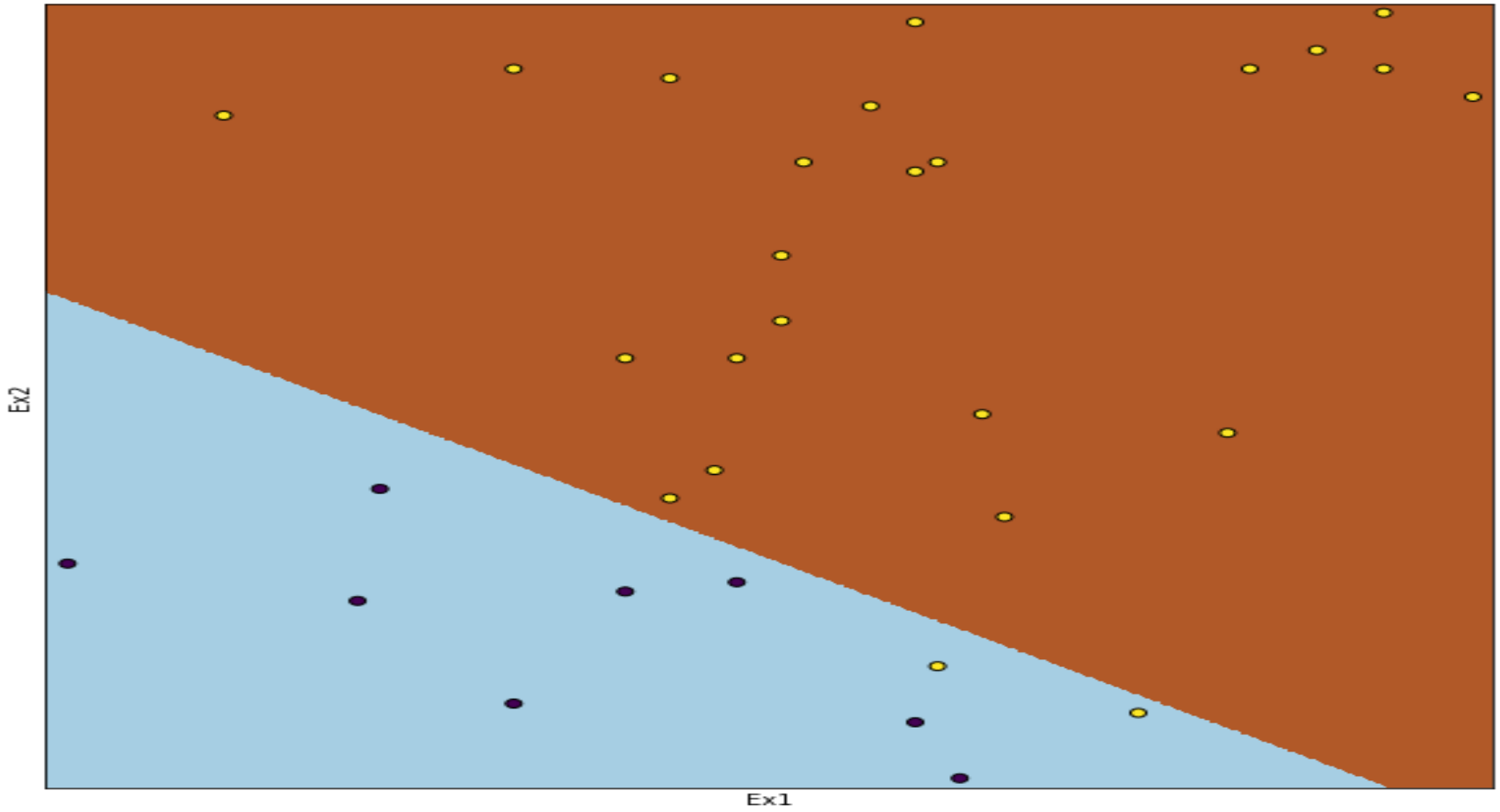
Example (cont)

#Finding the coeficients of the decision line
mean_vec=pasan.mean()-fail.mean()
np.array(mean_vec).dot(np.linalg.inv(np.array(pool_cov)))
array([0.08415817, 0.09539823])

#Finding the independent term
mean_vec1=pasan.mean()+fail.mean()
0.5*np.array(mean_vec).dot(np.linalg.inv(np.array(pool_cov))).dot(np.array(mean_vec1))
10.8220097479138 1

This the decision line is:
.084E1+.095E2=10.822

The Linear Discriminant Function



Tests for homogeneity of covariance matrices(homocedasticity)

- The most well known test for cheking homocedasticity (homogeneity of covariance matrices) is the Bartlett test. This test is a modification of the likelihood ratio test, however it is subject to the assumption of multivariate normal distribution. It makes use of a Chi-Square distribution. Bartlett test is available in scipy.stats. The Mardia test is one of several test to check multivariate normality.
- Other alternative is to extent the Levene's test for comparing the variance of several univariate populations.
- Other statistical tests for homocedasticity are the Box's and the Van Valen test.

Linear discriminant analysis as a Bayesian classifier

Let us consider that we have two classes C1 y C2 that follow a multivariate normal distribution, Np($\mathbf{u}_1$,Σ_1) and Np($\mathbf{u}_2$,Σ_2) respectively y que además tienen igual matriz de covarianza Σ_1=Σ_2 =Σ. Then the equation 3.4 can be written as

$$\frac{\exp[-1/2(\mathbf{x}-\mathbf{u}_1)'\Sigma^{-1}(\mathbf{x}-\mathbf{u}_1)}{\exp[-1/2(\mathbf{x}-\mathbf{u}_2)'\Sigma^{-1}(\mathbf{x}-\mathbf{u}_2)} > \frac{\pi_2}{\pi_1} \quad (3.8)$$

Taking logarithms in both sides, one obtains

$$-1/2[(\mathbf{x}-\mathbf{u}_1)'\Sigma^{-1}(\mathbf{x}-\mathbf{u}_1)-(\mathbf{x}-\mathbf{u}_2)'\Sigma^{-1}(x-\mathbf{u}_2)] > Ln(\frac{\pi_2}{\pi_1}) \quad (3.9)$$

After some simplifications one gets

$$(u_1-u_2)'\Sigma^{-1}(x-1/2(u_1+u_2))> \quad Ln(\frac{\pi_2}{\pi_1}) \quad (3.10)$$

Estimating the population parameters we obtain:

$$(\bar{X}_1 - \bar{X}_2)'S^{-1}[x - (1/2)(\bar{X}_1 + \bar{X}_2)] > Ln(\pi_2/\pi_1)$$

This inequality is similar to the one given in (3.5), except by the term in the right hand side, but considering equal prior probabilities l ($\pi_1=\pi_2$) then both equations are the same.

LDA for more than two classes

For G classes, the LDA assigns an object with attributes vector x to the class i such that

$$i=\text{argmax}_k\ \mu'_k\Sigma^{-1}x-(1/2)\ \mu'_k\Sigma^{-1}\ \mu'_k+\text{Ln}(\pi_k)$$

For all k in 1,2,…G. As before the right hand-side is estimated using the training sample.

Example: Vehicle dataset

```
#Carrying out LDA and calculating the accuracy
y=df1['Class']
X=df1.iloc[:,0:18]
y1=y.as_matrix()
X1=X.as_matrix()
ldadis = LinearDiscriminantAnalysis().fit(X1,y1)
#Accuracy Rate
ldadis.score(X1, y1)
0.797872340425531 9
```

LDA's Disadvantages

It works only for datasets with continuous features
It requires equal variability of the feature among the classes.
Its computatcion is very expensive. Specifically, **LDA** has $O(mnt + t^3)$ **time complexity** and requires $O(mn + mt + nt)$ memory, where m is the number of samples, n is the number of features and $t = \min(m,n)$.

The misclassification error rate

The misclassification error rate R(d) is the probability that the classifier d classifies incorrectly an instance coming from a sample (test sample) obtained in a later stage than the training sample. Also is called the True error or the actual error.
It is an unknown value that needs to be estimated.

Methods for estimation of the misclassification error rate

i) **Resubstitution or Aparent Aparente** (Smith, 1947). This is merely the proportion of instances in the training sample that are incorrectly classified by the classification rule. In general is an estimator too optimistic and it can lead to wrong conclusions if the number of instances is not large compared with the number of features. This estimator has a large bias.

ii) **"Leave one out" estimation. (**Lachenbruch, 1965). In this case an instance is omitted from the training sample. Then the classifier is built and the prediction for the omitted instances is obtained. One must register if the instance was correctly or incorrectly classfied. The process is repeated for all the instances in the training sample and the estimation of the ME will be given by the proportion of instances incorrectly classified. This estimator has low bias but its variance tends to be large.

Methods for estimation of the misclassification error rate

iii) Cross validation. (Stone, 1974) In this case the training sample is randomly divided in v parts (v=10 is the most used). Then the classifier is built using all the parts but one. The omitted part is considered as the test sample and the predictions for each instance on it are found. The CV misclassification error rate is found by adding the misclassification on each part and dividing them by the total number of instances. The CV estimated has low bias but high variance. In order to reduce the variability we usually repeat the estimation several times.

iv) The holdout method. A percentage (70%) of the dataset is considered as the training sample and the remaining as the test sample. The classifier is evaluated in the test sample. The experiment is repeated several times and then the average is taken.

The Confusion Matrix and metrics of a classifier performance

	Actual class=Yes	Actual class=No
Predicted class =Yes	True Positives=TP	False Positives=FP
Predicted class=No	False Negative=FN	True Negatives=TN

Accuracy=(TP+TN)/Total. This is the most used metric.
Missclassification rate=(FP+FN)/Total =1- Accuracy.
True Positives Rate=Sensitivity=Recall=TP/(TP+FN). For inbalanced classes.
False Positives Rate =FP(/FP+TN). For inbalanced classes.
Specifity=TN/(FP+TN)=1-False Positive Rate. For inbalanced classes.
Precision:TP/(TP+FP). True Positives among the positives detected by the classifier. For inbalanced classes.
Prevalence=(TP+FN)/Total.
F1-Score:2*(precision-recall)/(precision+recall)