Evaluation of Classification Algorithms
Cost Sensitive Learning

João Gama
jgama@fep.up.pt

LIAAD-INESC TEC, University of Porto, Portugal

October 2019

Outline

Overview

- Metrics for Two Classes Problems:
  Evaluation Measures
  True positive rate (recall), false positive rate, precision, etc.
  Calculation of metrics for 2 classes problems
- Cost-sensitive classification
  Different scenarios where cost is taken into account; Costs matrix,
  - Considering costs with trained models
  - Exploiting probabilistic classification and costs
  - Cost-sensitive ensemble methods

Motivation

- Consider a credit card classification problem. For each transaction your classifier will predict if the transaction is legal or fraudulent.
  - There are much more legal transactions than fraudulent transactions. For example, in 1000 transactions only one is fraudulent.
  - If your classifier always predict "Legal" the error rate will be 1/1000
  - It looks a great classifier, but fails in characterizing the target: "Fraud"
- Applications involving fraud detection, anomaly detection, intrusion detection, failure detection, etc exhibit similar behaviors!
- In 2-class imbalance problems the positive class is the minority and most relevant class.

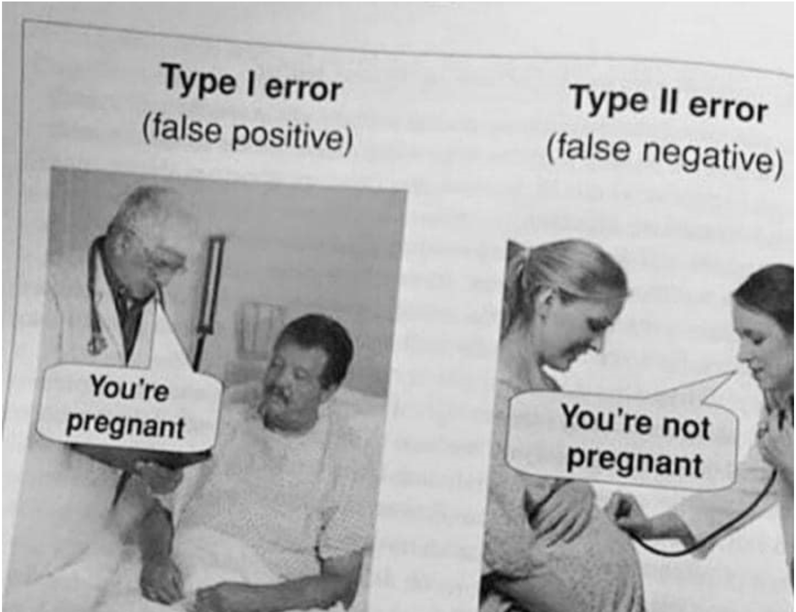
Metrics for Two Classes Problems

	Predict Classe	
True Classe	**Positive**	**Negative**
Positive	True Positives	False negatives
Negative	False Positives	True Negatives

Confusion matrix: shows successes and errors per each class

- **True positives (TP)**: nº of examples correctly classified with respect to prediction `Positive`
- **False positives (FP)**: nº of examples incorrectly classified with respect to prediction `Positive`
- **True negatives (TN)**: nº of examples correctly classified with respect to prediction `Negative`
- **False negatives (FN)**: nº of examples correctly classified with respect to prediction `Negative`

Type I and Type II Errors



Evaluation Measures

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive,** Power	**False positive,** Type I error
	Predicted condition negative	**False negative,** Type II error	**True negative**
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$

Relevant Metrics

- **Precision**: TP/(TP+FP)

	+^	-^
+	TP	FN
-	FP	TN

How many of those who we labeled as *Pos* are actually *Pos*?

- **Recall**: TP/(TP+FN)

	+^	-^
+	TP	FN
-	FP	TN

Of all the *Pos*, how many of those we correctly predict *Pos* ?

- **Specificity**: TN/(FP+TN)

	+^	-^
+	TP	FN
-	FP	TN

Of all the *Neg*, how many of those did we correctly predict?

- **Sensitivity**: TP/(TP+FN)

	+^	-^
+	TP	FN
-	FP	TN

= Recall

Sensitivity versus Specificity

Sensitivity and **specificity** are statistical measures of the performance of a binary classification test:

- **Sensitivity** (also called the true positive rate, the recall, or probability of detection) measures the proportion of actual positives that are correctly identified as such.
  (e.g., the percentage of sick people who are correctly identified as having the condition).
- **Specificity** (also called the true negative rate) measures the proportion of actual negatives that are correctly identified as such.
  (e.g., the percentage of healthy people who are correctly identified as not having the condition).

Sensitivity quantifies the avoiding of false negatives, and specificity does the same for false positives.

Precision versus Recall

- **Precision** (also called positive predictive value) is the fraction of relevant instances among the retrieved instances,
- **Recall** (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

Both precision and recall are therefore based on an understanding and measure of **relevance**.
It is possible to interpret precision and recall not as ratios but as probabilities:

- Precision is the probability that a (randomly selected) retrieved document is relevant.
- Recall is the probability that a (randomly selected) relevant document is retrieved in a search.

Computing Evaluation Measures for Class *bad*

Consider the following matrix of successes / errors for labor dataset:

	Predict Classe	
True Classe	**bad**	**good**
bad	19	1
good	6	31

TP=19, FN=1, FP = 6, TN=31
Assuming that class bad is the reference class, we get:

- TP Rate = TP / (TP + FN) = 19/(19+1) = 0.95
- FP Rate = FP / (FP + TN) = 6/(6+31) = 0.162
- Precision = TP /(TP + FP) = 19/(19 + 6) = 0.76
- Sensitivity = Recall = TP Rate = 0.95

Computing Evaluation Measures for Class *good*

TN=19, FP=1 FN=6, TP=31

For class good we get:

- TP Rate = TP/ (TP + FN) = 31 / (6+31) = 0.838
- FPRate =FP/(FP+TN) =1/ (19+1) =0.05
- Precision = TP/ (TP + FP) = 31 / (1+31) = 0.969
- Sensitivity= Recall = TP Rate = 0.838

F-score

- In information retrieval, the positive predictive value is called precision, and sensitivity is called recall.
- Unlike the Specificity vs Sensitivity tradeoff, these measures are both independent of the number of true negatives, which is generally unknown and much larger than the actual numbers of relevant and retrieved documents.
  This assumption of very large numbers of true negatives versus positives is rare in other applications.
- The `F-score` can be used as a single measure of performance of the test for the positive class. The `F-score` is the harmonic mean of precision and recall:

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

Exercise

- Diagnose of a rare disease

Model B Confusion Matrix			
		Disease	
		absent	present
Diagnose	negative	TN = 63	FN = 2
	positive	FP = 27	TP = 8

Model C Confusion Matrix			
		Disease	
		absent	present
Diagnose	negative	TN = 68	FN = 7
	positive	FP = 22	TP = 3

- The accuracy for both models is 71%.
- The goal is to achieve a good performance on the rare but most important cases.
- Which Model do you prefer?

Outline

Receiver Operating Characteristic

The term **ROC** (=Receiver Operating Characteristic), comes originally from the area of engineering

ROC curves typically plot:

- TP rate (vertical axis) = TP/(TP+FN) (=recall =sensitivity)
- FP rate (horizontal axis) = FP/(FP+TN) (= 1-specificity)

Different classifiers are represented by:

- different points (we have as many points as classifiers)
- different curves (one curve per classifier), if certain parameters are varied (see later)

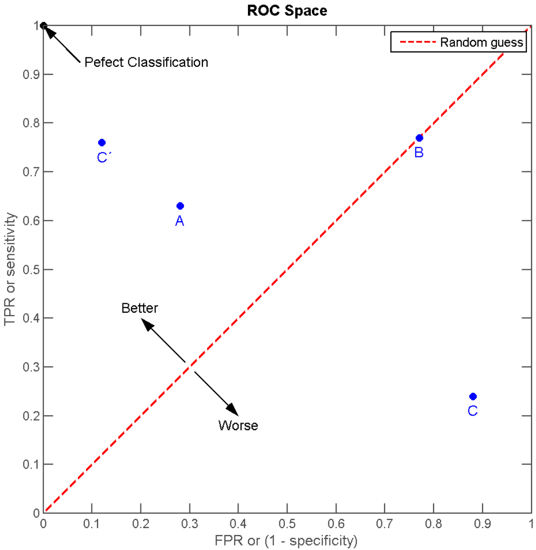
The objective is to identify the best classifier.

ROCSpace

Consider the classifiers:

A			B			C			C′		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=76	FP=12	88
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=24	TN=88	112
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
PPV = 0.69			PPV = 0.50			PPV = 0.21			PPV = 0.86		
F1 = 0.66			F1 = 0.61			F1 = 0.23			F1 = 0.81		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

ROCSpace


ROC Space
Random guess
Pefect Classification
C´
A
B
C
Better
Worse
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

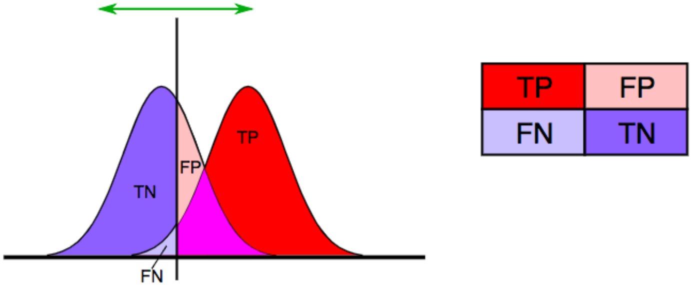
ROCSpace

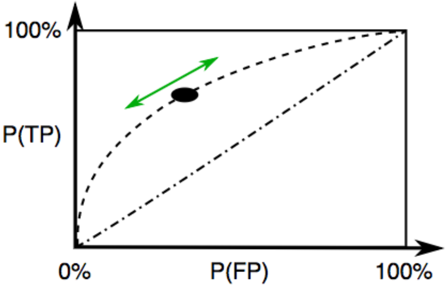
- The result of method A clearly shows the best predictive power among A, B, and C.
- The result of B lies on the random guess line (the diagonal line), and it can be seen in the table that the accuracy of B is 50%.
- However, when C is mirrored across the center point $(0.5, 0.5)$, the resulting method C' is even better than A.
  - When the C method predicts p or n, the C' method would predict n or p, respectively.

ROC Curves:
Varying threshold to obtain a ROC Curve

- Probabilistic classification outputs a probability p, indicating that a particular example e_i belongs to the given class C_j with and the associated probability is p.
- In a two classes problem, we say that e_i belongs to class C_j, if $p = P(C_j|e_i) > 0.5$.
- However, it is not necessary to assume that the threshold is 0.5.
- If the value of the threshold is increased (decreased), the nº of examples classified in C_j gets reduced (augmented).
- If we alter the threshold and apply this to all examples, we will obtain a different distribution of errors (hence FP rate, TP rate).
- If we repeat the process, we obtain a series of points for the ROC graph.

ROC Curves: Varying threshold to obtain a ROC Curve





ROC Curves

Rank	Score	Label	Predict
1	0.997	1	1
2	0.993	1	1
3	0.986	1	1
4	0.982	1	1
5	0.971	0	0
6	0.965	1	0
7	0.964	0	0
8	0.961	0	0
9	0.953	0	0
10	0.932	1	0
11	0.918	0	0
12	0.873	0	0
13	0.854	0	0
14	0.839	0	0
15	0.777	0	0
16	0.723	0	0
17	0.634	0	0
18	0.512	0	0
19	0.487	0	0
20	0.473	0	0

Predicted Positive: 4

Predicted Negative: 16

Population

- Total Label Positives: 6
- Total Label Negatives: 14
- Prevalence 6/20=0.3

For threshold > 0.980 or top k=4:

		True Condition	
	Total Population	Positive	Negative
Predicted Condition	Positive	TP 4	FP 0
	Negative	FN 2	TN 14

True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$

- True Positive Rate (Recall) = 4/6 = 0.66
- False Positive Rate: 0/14=0

- False Negative Rate = 2/6=0.33
- True Negative Rate = 14/14=1.0
- Precision = 4/4 =1.0

ROC Curves

Rank	Score	Label	Predict
1	0.997	1	1
2	0.993	1	1
3	0.986	1	1
4	0.982	1	1
5	0.971	0	1
6	0.965	1	1
7	0.964	0	1
8	0.961	0	1
9	0.953	0	1
10	0.932	1	1
11	0.918	0	0
12	0.873	0	0
13	0.854	0	0
14	0.839	0	0
15	0.777	0	0
16	0.723	0	0
17	0.634	0	0
18	0.512	0	0
19	0.487	0	0
20	0.473	0	0

Predicted Positive: 10

Predicted Negative: 10

Population

- Total Label Positives: 6
- Total Label Negatives: 14
- Prevalence 6/20=0.3

True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$

For threshold > 0.920 or top k=10:

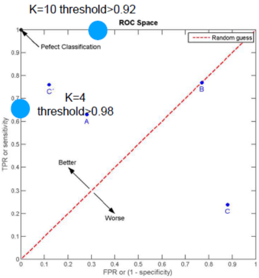
		True Condition	
	Total Population	Positive	Negative
Predicted Condition	Positive	TP 6	FP 4
	Negative	FN 0	TN 10

- True Positive Rate (Recall) = 6/6 = 1.0
- False Positive Rate: 4/14=0.29

- False Negative Rate = 0/6=0
- True Negative Rate = 10/14=0.71
- Precision = 6/10 =0.6

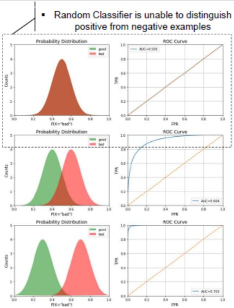
ROC Curves

A. ROC COORDINATES



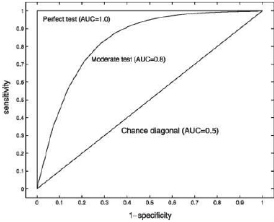
- To get ROC coordinates we define a threshold for 1/0 predictions and then we calculate the False Positive rate and Recall values for that threshold.

B. ROC PROFILES



- ROC curve is an analytical tool to assess model performance (if your dataset is not highly unbalanced)

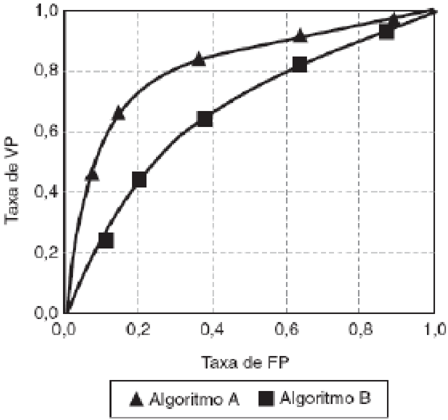
C. AUC



- AUC is the area under the ROC expresses how much a model can distinguish two classes along the score distribution in a given sample.

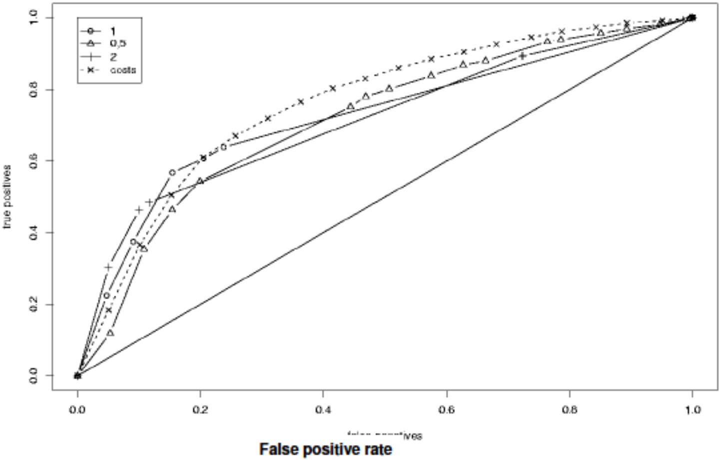
ROC Curves

The following graph shows two ROC curves.
The curve of classifier A dominates the curve of classifier B (it is always above / better).



ROC Curves

Curves may cross (no one dominates the others)

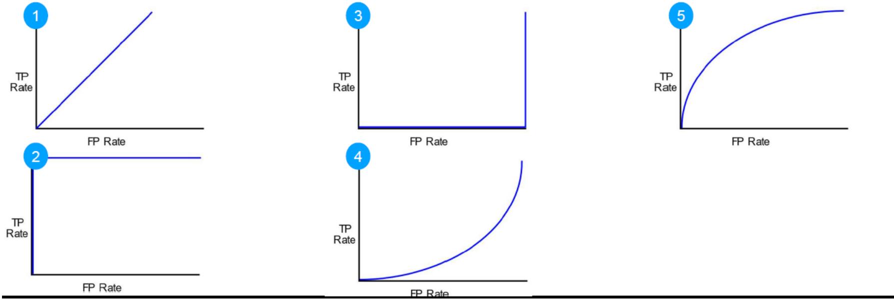


ROC Curves: Analysis

- The objective is to identify a convex hull (a convex curve that encloses all curves)
- Then, for a certain range of FP values (given by the user) the aim is to identify the classifiers that are closest to this hull.
- Area Under Curve (AUC)
  - Good measure of overall performance, if a simple metric is needed
  - Gives probability that the model will rank a positive case higher than a negative case.
- AUC is equivalent to Wilcoxon-Man-Whitney (WMW) statistic which has become popular as a quality measure in ranking problems.

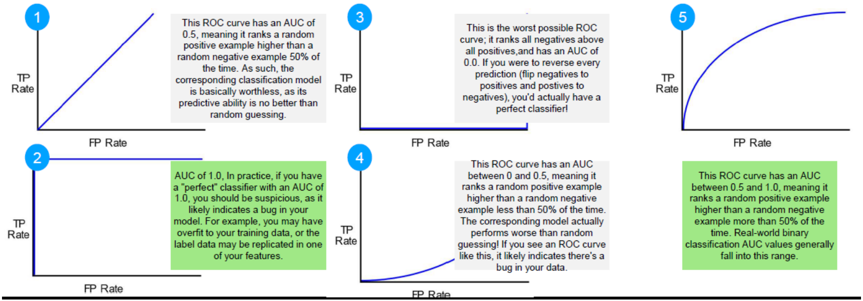
ROC Curves

Which of the following ROC curves produce AUC values greater than 0.5?



ROC Curves

Which of the following ROC curves produce AUC values greater than 0.5?

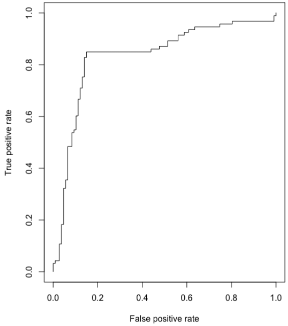


Constructing a ROC curve using ROCR

```
#Install package ROCR and execute:
> library(ROCR)
#Illustrative example using the data available in ROCR:
> data(ROCR.simple)
> ROCR.simple
$predictions
[1] 0.612547843 0.364270971 0.432136142 0.140291078
0.384895941 0.244415489 ..
$labels
[1] 1 1 0 0 0 1 ..
#The data 'ROCR.simple' includes both predictions and
true values (labels) that we need for the next step.
```

Constructing a ROC curve using ROCR

```
> pred.objects <-
+  prediction(ROCR.simple$predictions,ROCR.simple$labels)
> perf.ROC <- performance(pred.objects, 'tpr', 'fpr' )
> plot( perf.ROC)
```

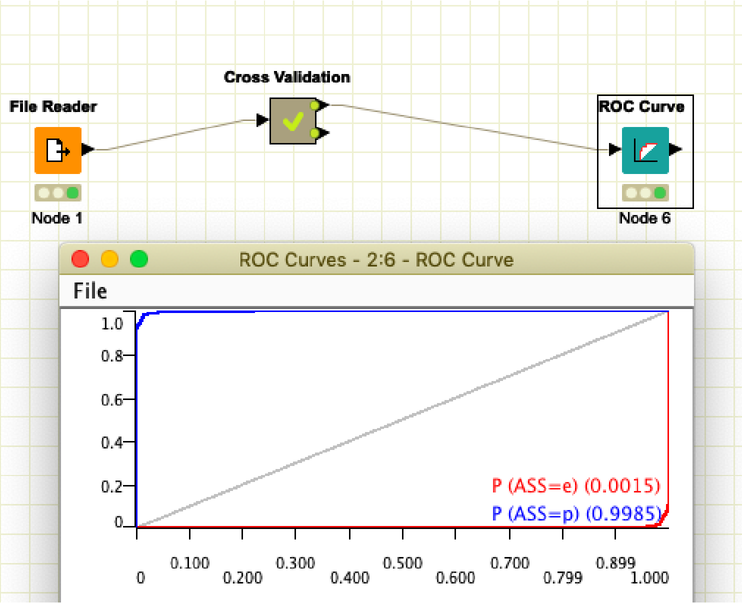


AUC - The Area Under the ROC Curve

```
AUC can be obtained using:
> perf.AUC <- performance(pred.objects, ' auc ')
> perf.AUC@y.values
[[1]]
[1] 0.8341875
```

The instruction performance(..) can take different arguments permitting to obtain many other performance values (see help(..)).

ROC in KNIME


Cross Validation
File Reader
ROC Curve
Node 1
Node 6
ROC Curves - 2:6 - ROC Curve
File
1.0
0.8
0.6
0.4
0.2
0
P (ASS=e) (0.0015)
P (ASS=p) (0.9985)
0
0.100
0.200
0.300
0.400
0.500
0.600
0.700
0.799
0.899
1.000

ROC in Python

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score

X, y = load_breast_cancer(return_X_y=True)
clf = GaussianNB()
clf.fit(X, y)
y_pred = clf.predict_proba(X)
preds = roc_auc_score(y, clf.predict_proba(X)[:, 1])
print("AUC:", preds)
fpr, tpr, _ = roc_curve(y, y_pred[:, 1])
```

ROC in Python

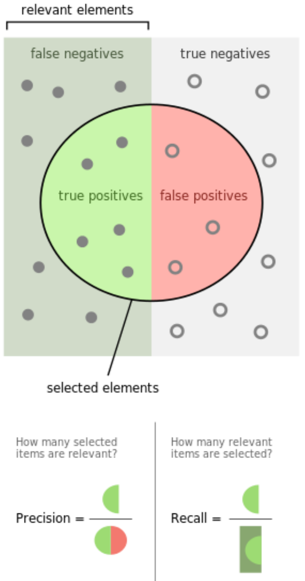
```
plt.figure()
plt.plot(
    fpr, tpr,
    color="darkred",
    lw=2,
    label="ROC curve (area = %0.3f)" % preds,
)
plt.plot([0, 1],[0, 1],color="navy",lw=2,linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```

Interpretation of the ROC curve

- the intercept of the ROC curve with the line at 45 degrees orthogonal to the no-discrimination line - the balance point where **Sensitivity = Specificity**
- the intercept of the ROC curve with the tangent at 45 degrees parallel to the no-discrimination line that is closest to the error-free point (0,1) - also called Youden's J statistic and generalized as Informedness
- the area between the ROC curve and the no-discrimination line multiplied by two - **Gini Coefficient**
- the area between the full ROC curve and the triangular ROC curve including only (0,0), (1,1) and one selected operating point (tpr,fpr) - **Consistency**

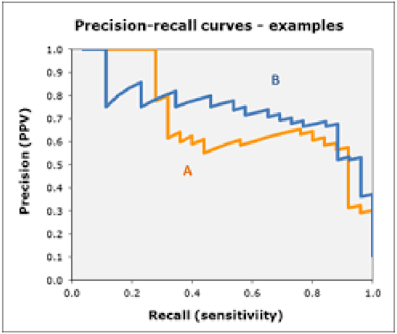
Outline

Precision-Recall Curves



- Precision=TP/(TP+FP) high precision means that the algorithm returned substantially more relevant results than irrelevant ones,
- Recall = TP/(TP+FN) high recall means that the algorithm returned most of the relevant results.

Precision-Recall Curves



- Precision=TP/(TP+FP)
  Recall = TP/(TP+FN)
- PR Curve: trade-off between recall and precision as the discrimination threshold for the two classes varies.
- As it does not account for TN, it is more suited for problems with class imbalance.

Precision-Recall Curve

```
library(ROCR)
data(ROCR.simple)
pred < − prediction( ROCR.simple$predictions, ROCR.simple$labels)
perf < − performance(pred,"tpr","fpr")
plot(perf)
# precision/recall curve (x-axis: recall, y-axis: precision)
perf1 < − performance(pred, "prec", "rec")
plot(perf1)
```

Precision-Recall Curve

You can first get the precision and recall values

```
x <- perf1@x.values[[1]] # Recall values
y <- perf1@y.values[[1]] # Precision values
```

Precision-Recall Curve

ROCR can calculate AUC directly:

```
perf <- performance(pred, "auc")
perf@y.values[[1]]
```

Exercise

Construct a ROC curve for dataset *mushrooms* (or any other). The solution involves the following steps:

- Copy data into a text file 'mushrooms.txt' on Desktop.
- Open R; Change directory to where the text file is; Reading in the data using read.scv('mushrooms.txt').
- Create data frames with train and test data
- Create a decision tree using rpart on the train data
- Obtain the predictions of the decision tree on the test data using:

```
> preds <- predict(..)
```

- If the model is called 'arvore' and the test data 'dados.teste' then:

```
> preds <- predict(arvore,dados.teste)
```

AUC - The Area Under the ROC Curve

Then we invoke prediction(..) which requires two inputs:

- predictions
  Predictions were stored previously in variable/object 'preds'.
  As the predictions include two probabilities for a problem with two classes we need to select one (by preds[,2])
- true class values (correct labels)
  The true class values for mushroom dataset are identified by name 'ASS'. Hence the true class values can be accessed by 'dados.teste$ASS'. The instruction is thus:
  ```
  > pred.objects<-prediction(preds[,2],dados.teste$ASS)
  ```
  Generete a ROC curve and plot it:
  ```
  > perf.ROC <- performance (pred.objects, 'tpr', 'fpr')
  > plot( perf.ROC)
  ```

Outline

Cost Sensitive Classification

- The classifications generated by the system
  lead to actions
  that lead to certain costs and benefits (utility, payoff).
- In many domains the errors have unequal costs:
  - Credit domain:
    cost of incorrectly giving credit is not the same as loss of not giving credit to a good customer;
  - Marketing:
    cost of useless mailing is not the same as loss of not mailing to a potential customer;
  - Fraud detection:
    cost of useless investigation is not the same as loss of not investigating a real fraud.

Cost Benefit Matrix

	Predict Classe	
True Classe	**P**	**N**
P	$C(P,\hat{P})$	$C(P,\hat{N})$
N	$C(N,\hat{P})$	$C(N,\hat{N})$

- $C(P,\hat{P})$ = benefit of TP (True Positives) (benefit of correctly classifying class P)
- $C(P,\hat{N}) = C_{FN}$ = cost of FN (False Negatives) (cost of incorrectly classifying class P)
- $C(N,\hat{P}) = C_{FP}$ = cost of FP (False Positives) (cost of incorrectly classifying class N)
- $C(N,\hat{N})$= benefit of TN (True Negatives) (benefit of correctly classifying class N)

Total Cost and Mean Cost

- Total cost (all cases):

$$C_{Tot} = FN * C(P, \hat{N}) + FP * C(N, \hat{P}) = FN * C_{FN} + FP * C_{FP}$$

- Average cost per case:

$$C_{Mean} = C_{Tot}/n = (FN * C(P, \hat{N}) + FP * C(N, \hat{P}))/n$$

- In practice, it is difficult to estimate the exact value of the costs. This problem is mitigated by:
  - setting one of the costs to 1,
  - determining the value of the other cost (e.g. 5, 10 etc.).

Different Scenarios when dealing with Costs

- Usually, ML algorithms do not consider costs, but costs are considered when evaluating a trained model to determine which model should be chosen.
- ML algorithms considers costs when classifying a case (not in training); Methods that deal with attribute costs:
- ML algorithm is reprogrammed to consider attribute costs in the training phase We obtain e.g. a cost sensitive decision tree.
- Costs are considered when constructing an ensemble involving multiple ML algorithms

The Mailing Example

Cost of mailing:

	Predict Classe	
True Classe	**P**	**N**
P	0	1000
N	1	0

Errors Classifier 1:

	Predict	
True	**P**	**N**
P	100	200
N	0	3700

Error rate = 200 / 4000 = 5%

Total cost = 200*1000= 200.000

Errors Classifier 2:

	Predict	
True	**P**	**N**
P	100	50
N	2000	1700

Error rate = 2050 / 4000 = 51.25%

Total cost = 50*1000+2000*1= 52.000

Exploiting probabilistic classification & costs

- We need a classifier that outputs a probability distribution of classes for each case.
- Costs are used to determine the classification for a particular case.

Consider a two classes problem: $y \in \{yes, no\}$. For a given example x, the classifier outputs $P(yes|x) = 0.1$ and $P(no|x) = 0.9$.

- Consider the class *yes*:
  The chance of the error is 0.1. Suppose the associated cost is 500. The probable cost is $0.1 \times 500 = 50$.
- For the class *no*:
  The chance of error is 0.9. Suppose the associated cost is 1. The probable cost is $0.9 \times 1 = 0.9$.

The class *no* minimizes the cost.

Adjusting Classification Using Costs

- Representing the outcome of probabilistic classification: $P(C_i|Ex)$
- Given a cost matrix, where $C_{i,j}$ is the cost of misclassifying an example that belongs to class i in class j.
- Calculate the cost estimate, CostE, considering all possibilities (as in decision theory):
  $CostE(Class_k|Ex) = \sum(P(Classj|ex) * Cost(Class_j, Class_k))$

Select the class that minimizes the cost estimate

Adjusting Classification Using Costs

Example:

- $P(yes|ex) = 0.9$ and $P(no|ex) = 0.1$
- $Cost(no, \hat{yes}) = 500$ and $Cost(yes, \hat{no}) = 1$
- Calculate the cost estimate considering all possibilities :
  - $CostE(\hat{yes}|ex) =$
    $P(yes|ex) \times Cost(yes, \hat{yes})) + P(no|ex) \times Cost(no, \hat{yes}) =$
    $0.9 \times 0 + 0.1 \times 500 = 50$
  - $CostE(\hat{no}|ex) =$
    $P(yes|ex) * Cost(yes, \hat{no})) + P(no|ex) * Cost(no, \hat{no}) =$
    $0.9 \times 1 + 0.1 \times 0 = 0.9$
- Select class *no*, as the cost estimate is smaller.

Outline

Bibliography

- J.Gama, A.de Carvalho, K.Facelli, A. C. Lorena, M.Oliveira: Extração de Conhecimento de Dados, Cap. 9, Ed. Sílabo, 2017.
- Wikipedia: https://en.wikipedia.org/wiki/Receiver_operating_characteristic
- Wikipedia: https://en.wikipedia.org/wiki/Sensitivity_and_specificity
- Fawcett, Tom (2006). "An Introduction to ROC Analysis", Pattern Recognition Letters. 27 (8): 861?874
- S. Lomax, S. Vadera: A Survey of Cost-sensitive Decision Tree Induction Algorithms, Computing Surveys, 2011.