Metrics for Evaluating Classifier Performance

Sohrab Hossain

CSE, EDU

Performance

- Measures for assessing how good or how "accurate" the classifier is at predicting the class label of tuples

- **Performance Measure**: **accuracy** (also known as recognition rate), **sensitivity** (or recall), **specificity**, **precision**, F_1, *and* F_β.

Evaluation measures

Measure	*Formula*
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

***TP,TN,FP,P, N refer to the number of true positive, true negative, false positive, positive,* & negative samples**

Some Terminology

- **positive tuples (tuples of the** main class of interest)→ positive samples
- **negative tuples** (all other tuples) → negative samples
- **Example:**
- Given two classes, positive tuples: *buys_computer = yes*
- *negative_tuples: buys computer = no*

- Suppose we use our classifier on a test set of labeled tuples.
- *P is the* number of positive tuples
- *N is the number of negative tuples.*
- *For each tuple, we* compare the classifier's class label prediction with the tuple's known class label.

- **True positives (*TP*):** 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 = no* for which the classifier predicted *buys_computer = yes).* Let FP be the number of false positives.
- **False negatives (*FN):*** These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys_computer = yes for which the classifier predicted *buys_computer = no).* Let FN be the number of false negatives.

Confusion matrix

- The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes.
- TP & TN tell us when the classifier is getting things right, while FP & FN tell us when the classifier is getting things wrong (i.e., mislabeling)

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

Structure

- Given m classes (where m ≥ 2), a **confusion matrix is a table of at least** size **m by m**.
- An entry, $CM_{i,j}$ in the first m rows & m columns indicates the number of tuples of class i that were labeled by the classifier as class j.
- For a classifier to have good accuracy, ideally most of the tuples would be represented along the diagonal of the confusion matrix, from entry $CM_{1,1}$ to entry $CM_{m,m}$, with the rest of the entries being zero or close to zero.
- That is, ideally, FP and FN are around zero.

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

- The table may have additional rows or columns to provide totals
- In addition, P` is the number of tuples that were labeled as positive (TP +FP)
- N` is the number of tuples that were labeled as negative (TN + FN).
- The total number of tuples is **TP +TN +FP +TN**, or **P +N**, or **P` +N`**.

Evaluation measures

- **Accuracy:** The **accuracy of a** classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier

$$accuracy = \frac{TP + TN}{P + N}$$

Also referred to as the **overall recognition rate** of the classifier, that is, it reflects how well the classifier recognizes tuples of the various classes

- Confusion matrix for the classes buys_computer = yes & buys_computer = no,
  -where an entry in row i & column j shows the number of tuples of class i that were labeled by the classifier as class j.
  -Ideally, the nondiagonal entries should be zero or close to zero.

Classes	*buys_computer = yes*	*buys_computer = no*	*Total*	*Recognition (%)*
buys_computer = yes	**6954**	**46**	7000	99.34
buys_computer = no	**412**	**2588**	3000	86.27
Total	7366	2634	10,000	95.42

Error rate or misclassification rate

- **error rate or misclassification rate of a classifier (*M*): 1 – accuracy(M),** *where* ***accuracy(M)*** *is the accuracy of M.*

$$error\ rate = \frac{FP + FN}{P + N}$$

-If we were to use the training set (instead of a test set) to estimate the error rate of a model, this quantity is known as the **resubstitution error.**

-This error estimate is optimistic of the true error rate (& similarly, the corresponding accuracy estimate is optimistic) because the model is not tested on any samples that it has not already seen.

Class imbalance problem

- The **class imbalance problem,** where the main class of interest is rare.
- That is, the data set distribution reflects a significant majority of the negative class & a minority positive class.

❑ For example, in fraud detection applications, the class of interest (or positive class) is “fraud,” which occurs much less frequently than the negative “nonfraudulant” class.

- In medical data, there may be a rare class, such as “cancer.”
- Suppose that you have trained a classifier to classify medical data tuples, where the class label attribute is “cancer” & the possible class values are “yes” and “no.”

- An accuracy rate of, say, 97% may make the classifier seem quite accurate, but what if only, say, 3% of the training tuples are actually cancer?
- Clearly, an accuracy rate of 97% may not be acceptable—the classifier could be correctly labeling only the noncancer tuples, for instance, & misclassifying all the cancer tuples.
- Instead, we need other measures, which access how well the classifier can recognize the positive tuples (cancer = yes) & how well it can recognize the negative tuples (cancer = no).

Sensitivity & Specificity

- The **sensitivity & specificity** measures can be used, respectively, for this purpose.
- **Sensitivity** is also referred to as the true positive (recognition) rate (i.e., the proportion of positive tuples that are correctly identified),
- while **specificity** is the true negative rate (i.e., the proportion of negative tuples that are correctly identified).

$$sensitivity = \frac{TP}{P}$$

$$specificity = \frac{TN}{N}$$

- It can be shown that accuracy is a function of sensitivity & specificity

$$accuracy = \frac{TP + TN}{P + N}$$

$$accuracy = sensitivity \frac{P}{(P+N)} + specificity \frac{N}{(P+N)}.$$

Example: Sensitivity & specificity

- a confusion matrix for medical data where the class values are *yes* & *no* for a class label attribute, **cancer**.

Classes	*yes*	*no*	*Total*	*Recognition (%)*
yes	**90**	**210**	300	30.00
no	**140**	**9560**	9700	98.56
Total	230	9770	10,000	96.40

sensitivity of the classifier is 90/300 = 30.00%

specificity is 9560/9700 = 98.56%

classifier's **overall accuracy** is

9650 [9560+90]/10,000 = 96.50%.

- although the classifier has a high accuracy, it's ability to correctly label the positive (rare) class is poor given its low sensitivity.
- It has high specificity, meaning that it can accurately recognize negative tuples.

- The precision & recall measures are also widely used in classification.
- ***Precision*** can be thought of as a measure of *exactness (i.e., what percentage of tuples labeled as* positive are actually such)
- **Recall** is a measure of completeness (what percentage of positive tuples are labeled as such).

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

Example: Precision and recall

Confusion matrix for the classes *cancer = yes* & *cancer = no.*

Classes	*yes*	*no*	*Total*	*Recognition (%)*
yes	**90**	**210**	300	30.00
no	**140**	**9560**	9700	98.56
Total	230	9770	10,000	96.40

- **The precision of the classifier for the *yes* *class is***
- 90/230 = 39.13%.
- The recall is
- 90/300 = 30.00%,

Perfect Precision & recall

- A perfect precision score of 1.0 for a class C means that every tuple that the classifier labeled as belonging to class C does indeed belong to class C.
- However, it does not tell us anything about the number of class C tuples that the classifier mislabeled.
- A perfect recall score of 1.0 for C means that every item from class C was labeled as such,
- but it does not tell us how many other tuples were incorrectly labeled as belonging to class C.

- There tends to be an inverse relationship between precision & recall, where it is possible to increase one at the cost of reducing the other.
- For example, medical classifier may achieve high precision by labeling all cancer tuples that present a certain way as cancer, but may have low recall if it mislabels many other instances of cancer tuples.
- Precision & recall scores are typically used together, where precision values are compared for a fixed value of recall, or vice versa.
- For example, we may compare precision values at a recall value of, say, 0.75.

Alternative way to use precision & recall

- To combine them into a single measure.
- This is the approach of the F measure

- F_1 score or F-score
- *F_β measure*

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

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall},$$

where β is a non-negative real number

- The F measure is the harmonic mean of precision & recall.
- It gives equal weight to precision & recall.
- The F_β measure is a weighted measure of precision & recall.
- It assigns β times as much weight to recall as to precision.
- Commonly used F_β measures:
  - $\mathbf{F_2}$ (which weights recall twice as much as precision)
  - $\mathbf{F_{0.5}}$ (which weights precision twice as much as recall)

Are there other cases where accuracy may not be appropriate?"

- In classification problems, it is commonly assumed that all tuples are uniquely classifiable, that is, that each training tuple can belong to only one class.
- Yet, owing to the wide diversity of data in large databases, it is not always reasonable to assume that all tuples are uniquely classifiable.
- Rather, it is more probable to assume that each tuple may belong to more than one class.
- **How then can the accuracy of classifiers on large databases be measured?**
- The accuracy measure is not appropriate, because it does not take into account the possibility of tuples belonging to more than one class.

- Rather than returning a class label, it is useful to return a probability class distribution.
- Accuracy measures may then use a **second guess** heuristic, whereby a class prediction is judged as correct if it agrees with the first or second most probable class.
- Although this does take into consideration, to some degree, the non-unique classification of tuples, it is not a complete solution.

In addition to accuracy-based measures, classifiers can also be compared with respect to the additional aspects

- **Speed:** This refers to the computational costs involved in generating and using the given classifier.
- **Robustness:** This is the ability of the classifier to make correct predictions given noisy data or data with missing values.
  - Robustness is typically assessed with a series of synthetic data sets representing increasing degrees of noise & missing values.
- **Scalability:** This refers to the ability to construct the classifier efficiently given large amounts of data.
  - Scalability is typically assessed with a series of data sets of increasing size.

- **Interpretability:** This refers to the level of understanding & insight that is provided by the classifier or predictor.
  - Interpretability is subjective & therefore more difficult to assess.
  - Decision trees & classification rules can be easy to interpret, yet their interpretability may diminish the more they become complex

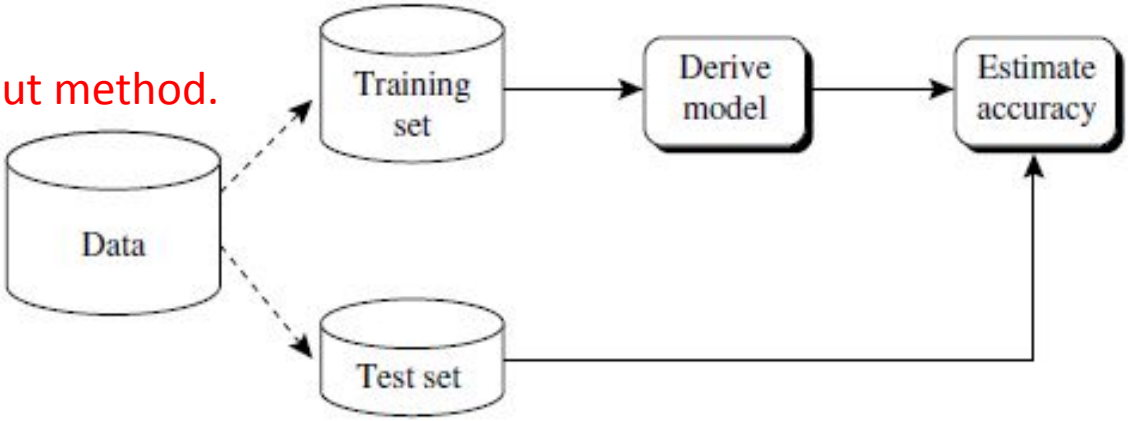
At a glance…

- The accuracy measure works best when the data classes are fairly evenly distributed.
- Other measures, such as sensitivity (or recall), specificity, precision, *F, & F_β, are better suited to the class imbalance* problem, where the main class of interest is rare.

Holdout Method and Random Subsampling

- **holdout method**
  - 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, & the remaining one-third is allocated to the test set.
  - The training set is used to derive the model.
  - The model's accuracy is then estimated with the test set

Estimating accuracy with the holdout method.



- 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

- ***k-fold cross-validation***,

✔ the initial data are randomly partitioned into k mutually exclusive subsets or "folds," D_1, D_2, ……., D_k, each of approximately equal size.

✔ Training & testing is performed k times.

✔ In iteration i, partition D_i is reserved as the test set, & the remaining partitions are collectively used to train the model.

- That is, in the first iteration, subsets D_2, ……, D_k collectively serve as the training set to obtain a first model, which is tested on D_1;
- the second iteration is trained on subsets D_1, D_3, …. , D_k & tested on D_2; and so on.
- Unlike the holdout & random subsampling methods, here each sample is used the same number of times for training & 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.

- **Leave-one-out:** is a special case of k-fold cross-validation where k is set to the number of initial tuples. That is, only one sample is “left out” at a time for the test set.
- **Stratified cross-validation,** the folds are stratified so that the class distribution of the tuples in each fold is approximately the same as that in the initial data.
  - In general, stratified 10-fold cross-validation is recommended for estimating accuracy (even if computation power allows using more folds) due to its relatively low bias & variance.

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.

Bootstrap methods

- A commonly used one is the **.632 bootstrap:**
- 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,
- the remaining 36.8% will form the test set (hence, the name, .632 bootstrap).

Where does the figure, 63.2%, come from?

- Each tuple has a probability of 1/d of being selected, so the probability of not being chosen is (1-1/d).
- We have to select d times, so the probability that a tuple will not be chosen during this whole time is $(1-1/d)^d$.
- If d is large, the probability approaches
- $e^{-1} = 0.368$.
- Thus, 36.8% of tuples will not be selected for training and thereby end up in the test set,
- the remaining 63.2% will form the training set.

- We can repeat the sampling procedure k times, where in each iteration, we use the current test set to obtain an accuracy estimate of the model obtained from the current bootstrap sample.
- The overall accuracy of the model, *M, is then estimated as*

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set}),$$

- where $Acc(M_i)_{test\_set}$ is the accuracy of the model obtained with bootstrap sample i when it is applied to test set i.
- $Acc(M_i)_{train\_set}$ is the accuracy of the model obtained with bootstrap sample i when it is applied to the original set of data tuples.
- Bootstrapping tends to be overly optimistic. It works best with small data sets.

Model Selection Using Statistical Tests of Significance

- Suppose that we have generated two classification models, ***M1 & M2***, from our data.
- We have performed 10-fold cross-validation to obtain a mean error rate for each.
- **How can we determine which model is best?**

- It may seem intuitive to select the model with the lowest error rate;
- however, the mean error rates are just estimates of error on the true population of future data cases.
- There can be considerable variance between error rates within any given 10-fold cross-validation experiment.
- Although the mean error rates obtained for M1 & M2 may appear different, that difference may not be statistically significant.
- What if any difference between the two may just be attributed to chance?

- To determine if there is any "**real**" difference in the mean error rates of two models,
- Need to employ a test of **statistical significance**.
- In addition, need to obtain some confidence limits for our mean error rates so that we can make statements like, "Any observed mean will not vary by ± two standard errors 95% of the time for future samples"
- or "One model is better than the other by a margin of error of 4% ±"

What do we need to perform the statistical test?

- Suppose that for each model, we did 10-fold cross-validation, say, 10 times, each time using a different 10-fold data partitioning.
- Each partitioning is independently drawn.
- We can average the 10 error rates obtained each for M_1 & M_2, respectively, to obtain the mean error rate for each model.
- For a given model, the individual error rates calculated in the cross-validations may be considered as different, independent samples from a probability distribution.

- In general, they follow a t-distribution with k-1 degrees of freedom where, here, k = 10.
- This allows us to do hypothesis testing where the significance test used is the **t-test,** or **Student's t-test.**
- Our hypothesis is that the two models are the same, or in other words, that the difference in mean error rate between the two is zero.
- If we can reject this hypothesis (referred to as the null hypothesis),

-then we can conclude that the difference between the two models is statistically significant,

-in which case we can select the model with the lower error rate.

- We may often employ a single test set, that is, the same test set can be used for both M1 & M2.
- In such cases, we do a pair-wise comparison of the two models for each 10-fold cross-validation round.
- That is, for the i^{th} round of 10-fold cross-validation, the same cross-validation partitioning is used to obtain an error rate for M1 & for M2.

- Let $err(M_1)_i$ (or $err(M_2)_i$) be the error rate of model M_1 (or M_2) on round i.
- The error rates for M_1 are averaged to obtain a mean error rate for M_1, denoted $\overline{err}(M_1)$.
- For M_2 : $\overline{err}(M_2)$
- The variance of the difference between the two models is denoted $var(M_1 - M_2)$.
- The t-test computes the t-statistic with k -1 degrees of freedom for k samples.
- In our example we have *k = 10 since, here,* the k samples are our error rates obtained from ten 10-fold cross-validations for each model.

- The t-statistic for pair-wise comparison is computed as follows:

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}},$$

$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))]^2.$$

- To determine whether M_1 & M_2 are significantly different, we compute t & select a **significance level, sig.**
- In practice, a significance level of 5% or 1% is typically used.
- We then consult a table for the t-distribution, available in standard textbooks on statistics.
- This table is usually shown arranged by degrees of freedom as rows & significance levels as columns.

- Suppose we want to ascertain whether the difference between M_1 & M_2 is significantly different for 95% of the population, that is, sig = 5% or 0.05.
- We need to find the t-distribution value corresponding to k -1 degrees of freedom (or 9 degrees of freedom for our example) from the table.
- However, because the t-distribution is symmetric, typically only the upper percentage points of the distribution are shown.
- Therefore, we look up the table value for **z= sig/2**, which in this case is 0.025, where z is also referred to as a **confidence limit**

- If $t > z$ or $t < -z$, then our value of t lies in the rejection region, within the distribution's tails.
- This means that we can reject the null hypothesis that the means of M_1 & M_2 are the same & conclude that there is a statistically significant difference between the two models.
- Otherwise, if we cannot reject the null hypothesis, we conclude that any difference between M_1 & M_2 can be attributed to chance.

- If two test sets are available instead of a single test set, then a non-paired version of the t -test is used, where the variance between the means of the two models is estimated as

$$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

- k_1 & k_2 are the number of cross-validation samples used for M_1 & M_2, respectively.
- This is also known as the **two sample t-test.**
- When consulting the table of t-distribution, the number of degrees of freedom used is taken as the minimum number of degrees of the two models.

Comparing Classifiers Based on Cost–Benefit and ROC Curves

- The true positives, true negatives, false positives, & false negatives are also useful in assessing the **costs & benefits (or risks & gains)** associated with a classification model.
- The cost associated with a false negative (such as incorrectly predicting that a cancerous patient is not cancerous) is far greater than those of a false positive (incorrectly yet conservatively labeling a noncancerous patient as cancerous).

- In such cases, we can outweigh one type of error over another by assigning a different cost to each.
- These costs may consider the danger to the patient, financial costs of resulting therapies, and other hospital costs.
- Similarly, the benefits associated with a true positive decision may be different than those of a true negative.

- we can incorporate costs & benefits by instead computing the average cost (or benefit) per decision.
- Other applications involving cost–benefit analysis include loan application decisions & target marketing mailouts.
- For example, the cost of loaning to a defaulter greatly exceeds that of the lost business incurred by denying a loan to a non-defaulter.
- Similarly, in an application that tries to identify households that are likely to respond to mailouts of certain promotional material, the cost of mailouts to numerous households that do not respond may outweigh the cost of lost business from not mailing to households that would have responded.
- Other costs to consider in the overall analysis include the costs to collect the data and to develop the classification tool.

Receiver operating characteristic curves

- ROC curves are a useful visual tool for comparing two classification models.
- ROC curves come from signal detection theory that was developed during World War II for the analysis of radar images.
- An ROC curve for a given model shows the trade-off between the true positive rate (TPR) & the false positive rate (FPR).

- Given a test set & a model,

- TPR is the proportion of positive (or “yes”) tuples that are correctly labeled by the model;
- FPR is the proportion of negative (or “no”) tuples that are mislabeled as positive.
- Given that TP, FP, P, & N are the number of true positive, false positive, positive, and negative tuples

- **TPR = TP/P** , which is *sensitivity*.
- **FPR = FP/N** , which is *1-specificity*

- For a two-class problem, an ROC curve allows us to visualize the trade-off between the rate at which the model can accurately recognize positive cases versus the rate at which it mistakenly identifies negative cases as positive for different portions of the test set.
- Any increase in TPR occurs at the cost of an increase in FPR.
- The area under the ROC curve is a measure of the accuracy of the model.

- To plot an ROC curve for a given classification model, M, the model must be able to return a probability of the predicted class for each test tuple.
- With this information, we rank & sort the tuples so that the tuple that is most likely to belong to the positive or "yes" class appears at the **top** of the list, & the tuple that is least likely to belong to the positive class lands at the **bottom** of the list.

Plotting an ROC curve

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

Tuples sorted by decreasing score, where the score is the value returned by a probabilistic classifier

- Column 1 is merely a tuple identification number, which aids in our explanation.
- Column 2 is the actual class label of the tuple: ***P = 5*** *and* ***N = 5****.*
- we can determine the values of the remaining columns, *TP, FP, TN, FN, TPR, and FPR.*

We start with tuple 1, which has the highest probability score, and take that score as our threshold, that is, *t = 0.9.*

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

- Thus, the classifier considers tuple 1 to be positive, & all the other tuples are considered negative
- Since the actual class label of tuple 1 is positive, we have a true positive, hence *TP = 1 and FP = 0.*

- Among the remaining nine tuples, which are all classified as negative, five actually are negative (thus, TN = 5)

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

- The remaining four are all actually positive, thus, ***FN = 4***
  compute $TPR = TP/P = 1/5 = 0.2$
- **FPR = FP/P = 0**

Thus, we have the point **(0.2,0)** for the ROC curve.

- Next, threshold t is set to 0.8, the probability value for tuple 2, so this tuple is now also considered positive, while tuples 3 through 10 are considered negative.

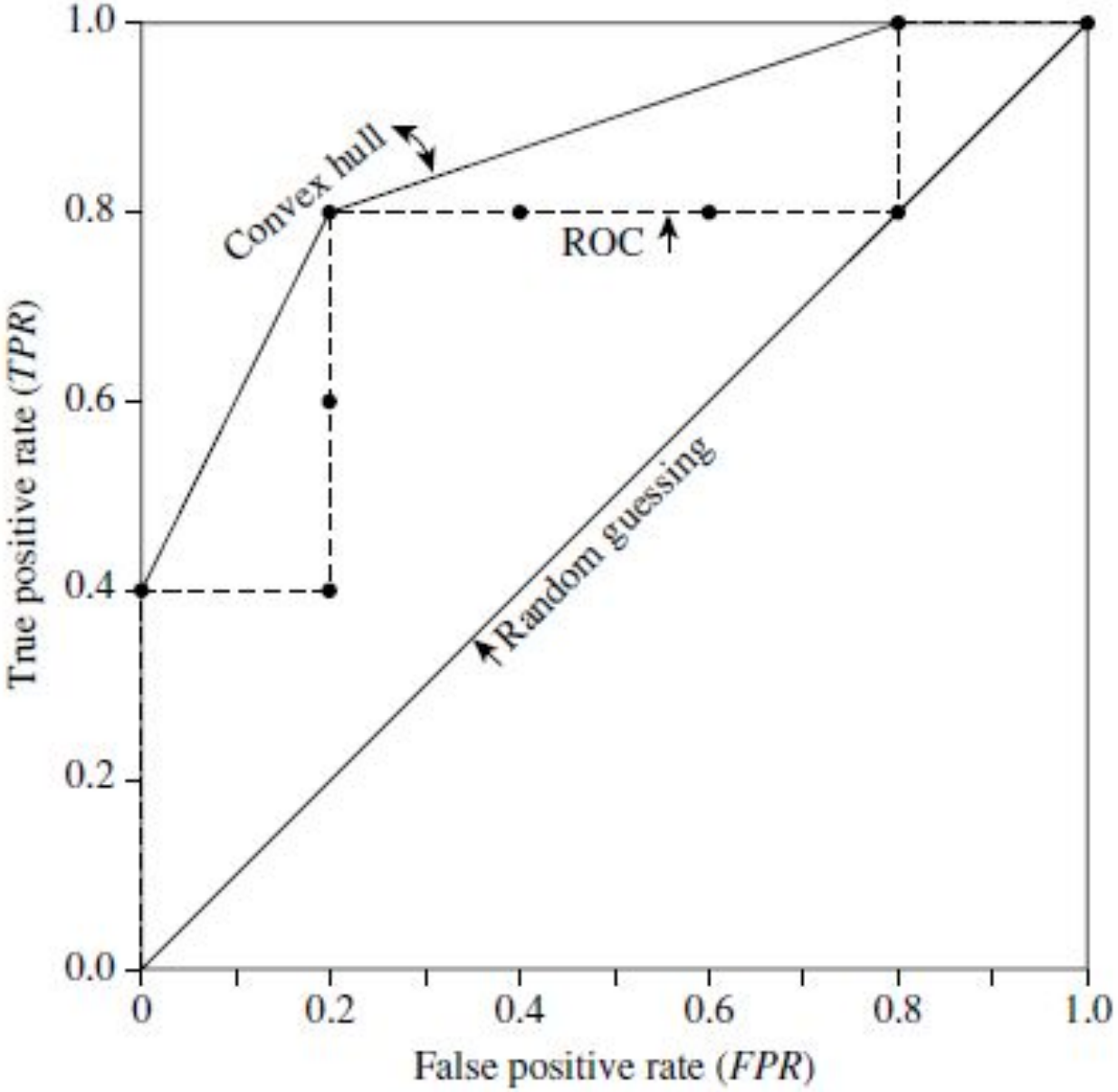
Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	0	1	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

- The actual class label of tuple 2 is positive, thus now TP = 2.
- The rest of the row can easily be computed, resulting in the **point (0.4,0)**.

❑Next, we examine the class label of tuple 3 and let t be 0.7, the probability value returned by the classifier for that tuple.

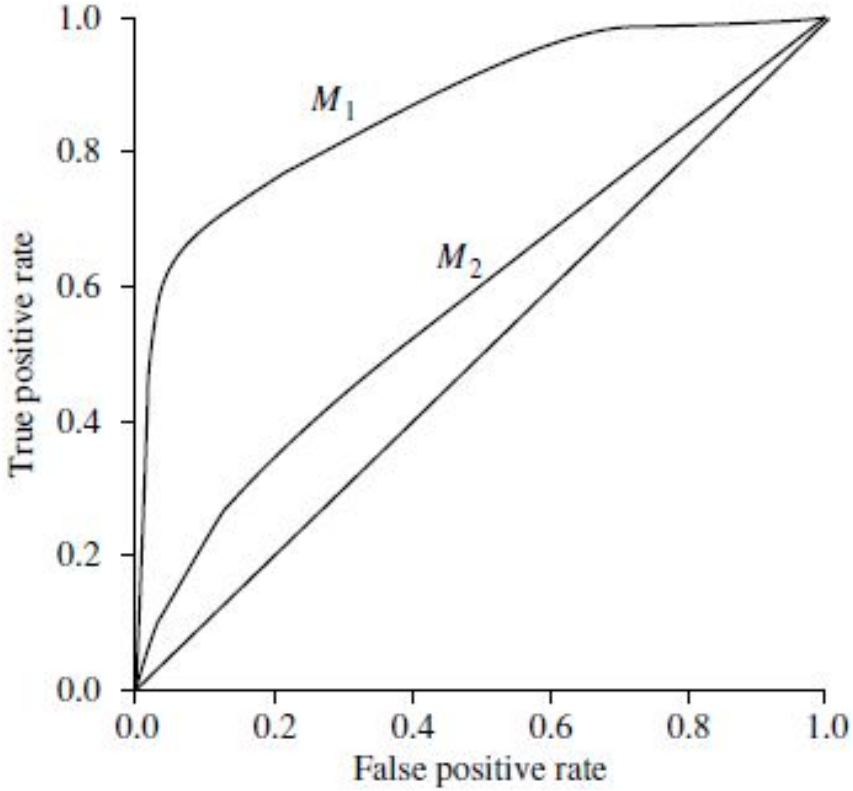
Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0

- Thus, tuple 3 is considered positive, yet its actual label is negative, and so it is a false positive.
- Thus, TP stays the same and FP increments so that FP = 1.
- The rest of the values in the row can also be easily computed, yielding the **point (0.4, 0.2).**


1.0
0.8
0.6
0.4
0.2
0.0
True positive rate (TPR)
Convex hull
ROC
Random guessing
0
0.2
0.4
0.6
0.8
1.0
False positive rate (FPR)

- There are many methods to obtain a curve out of these points, the most common of which is to use a convex hull.
- The plot also shows a diagonal line where for every true positive of such a model, we are just as likely to encounter a false positive.
- For comparison, this line represents random guessing.

- Fig. shows the ROC curves of two classification models.
- The diagonal line representing random guessing



-ROC curves of two classification models, M_1 & M_2.
-The diagonal shows where, for every true positive, we are equally likely to encounter a false positive.
-The closer an ROC curve is to the diagonal line, the less accurate the model is.
Thus, M1 is more accurate here.

- Thus, the closer the ROC curve of a model is to the diagonal line, the less accurate the model.
- If the model is really good, initially we are more likely to encounter true positives as we move down the ranked list.

- Thus, the curve moves steeply up from zero.
- Later, as we start to encounter fewer & fewer true positives,
- more & more false positives, the curve eases off
- becomes more horizontal.

- To assess the accuracy of a model, we can measure the area under the curve.
- Several software packages are able to perform such calculation.
- **The closer the area is to 0.5, the less accurate the corresponding model is.**
- **A model with perfect accuracy will have an area of 1.0.**