Decision Making Technologies for Business

2018/2019 Semester II

Associate Professor
HUANG, Ke-Wei

Today's Class

1. "Accuracy" Metrics for selecting models.
   - Predicting Categories (classification)
   - Predicting Probabilities (classification)
   - Predicting Continuous Variables
2. Procedure to evaluate and select models.
   - Training Set, Validation Set, and Test Set
3. Basics of Cost Sensitive Classification
   - Learn ROC and AUC.
   - How to train your model?

Issues Affecting Algorithms Selection

选择 algorithms 的标准

- Accuracy (Most important)
- **Speed (probably 2nd most important)** speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
  - Matters more for large but not small dataset
  - Matters more for time-critical task: intra-day trading or other real-time decision ⇔ not all tasks are time-critical.
  - Big data and cloud computing alleviates this problem.
- **Scalability in terms of computer memory usage**
- **Interpretability**
- **Robustness**: handling noise and missing values

1-1: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	1	0
1	**True Positives (TP)**	**False Negatives (FN)**
0	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*

Performance Metrics

1. (Important) Accuracy
2. (Important) Error Rate
3. (Important) Precision
4. (Important) Recall
5. (Important) F-Measure
6. (Rarely used) True Positive Rate = **Sensitivity**
7. (Rarely used) True Negative Rate = **Specificity**

这两个都是被正确分的

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	1	0	
1	**TP**	**FN**	**P**
0	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Less Common terms**
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

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

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

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

- In general, there is a tradeoff between precision and recall. More on this soon when we talk about ROC curve. In other words, inverse relationship between precision & recall

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- Because there is a tradeoff between two metrics, it is helpful to have one metric that consider both.
- ***F* measure (*F_1* or *F*-score)**: harmonic mean of precision and recall,

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

- ***F_β*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

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

Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity)*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

1. Depending on your data mining problem, you need to judge and choose one metric for your problem.
2. Later this lecture we will discuss ROC curve, which considers both TP and FP at the same time.
3. Cost sensitive classification later this lecture also provides one solution related to the choice of metric.

Kappa Statistics

- Kappa is another performance metrics that you may see in classification output in R or Python.
- It is especially useful when we try to predict unbalanced label.
- The formula is

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}$$

- Pr(a) is the probability that your classifier is "accurately" predicting the label is 0 or 1.
- Pr(e) is the "expected accuracy" by random guessing.
- Kappa is 0 to 1, and intuitively, it is like among the unpredictable cases, how many percentage you can predict it accurately.

Kappa Statistics

- For example,

	Predicted Yes	No	Total
Actual Yes	20	5	25
No	10	15	25
Total	30	20	50

- Pr(a) = (20+15)/50 = 0.7
- Pr(e) = [(20+10)/50]*[(20+5)/50] + [(5+15)/50]*[(10+15)/50]
  - The first term is the proportion my classifier classified as 1, among it, by prior probability (20+5)/50 proportion is 1 (accurate).
  - The second term is the proportion my classifier classified as 0, among it, by prior probability (10+15)/50 proportion is 0 (accurate).
  - So the expected accuracy is the sum of these two terms = 0.5
- Kappa = (0.7-0.5)/(1-0.5) = 0.4

1-2 Predicting Probabilities

- In some cases, confusion matrix may not be good enough to help us selecting the best model.
- For example, assume the true answer is 1. Also, if
  classifier 1 predicts 95% => 1 and 5% => 0;
  classifier 2 predicts 60% => 1 and 40% => 0.
  then performance may be considered as the same if we only consider 0-1 loss function, as that in the confusion matrix.
- Most classification algorithms also output label's class predicted probabilities
- Depending on the application, we might want to check the "accuracy" of the probability estimates.
- 0-1 loss is not good enough if small differences leads huge differences in dollar amount or human lives.
- **AUC** (later) is the best solution. The other 2 are provided next.

Solution: Quadratic Loss Function

- $p_1 \ldots p_k$ are probability estimates for an example
- $a_1 \ldots a_k$ are actual probabilities
- *Similar to sum-squared-error, Quadratic Loss* is:

$$\sum_j (p_j - a_j)^2$$

- We can show that the expected value of this is minimized when $p_j = a_j$, where the latter are the true probabilities.

- The other possible metric is called **Information Loss Function** shown below (p* is the actual probability)

$$-p_1^* \log_2 p_1 - p_2^* \log_2 p_2 - \ldots - p_k^* \log_2 p_k$$

1-3 Evaluating numeric prediction

- Actual target values: $a_1\ a_2 \ldots a_n$
- Predicted target values: $p_1\ p_2 \ldots\ p_n$
- Most intuitive measure: *mean-squared error*

$$\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}$$

- Most commonly used: The *root mean-squared error (RMSE)* $\sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}}$

- The *mean absolute error (MAE)* is less sensitive to outliers than the mean-squared error:

$$\frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{n}$$

Improvement on the mean

- The *relative squared error* is:

$$\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \cdots + (a_n - \bar{a})^2}$$

(in this formula and the following two, $\bar{a}$ is the mean value over the training data)

- The *root relative squared error* and the *relative absolute error* are:

$$\sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \cdots + (a_n - \bar{a})^2}} \qquad \frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{|a_1 - \bar{a}| + \cdots + |a_n - \bar{a}|}$$

- The metrics on the previous slides, especially RMSE, could be the most frequently used one.

Which measure?

- Best to look at all of them
- Often it is consistent across metrics and it doesn't matter
- Example:

	A	B	C	D
Root mean-squared error	67.8	91.7	63.3	57.4
Mean absolute error	41.3	38.5	33.4	29.2
Root rel squared error	42.2%	57.2%	39.4%	35.8%
Relative absolute error	43.1%	40.1%	34.8%	30.4%
Correlation coefficient	0.88	0.88	0.89	0.91

- D best
- C second-best
- A, B have inconsistent ordering but A seems better.

Today's Class

1. Criteria for Selecting Models.
   - Predicting Categories = Classification
   - Predicting Probabilities
   - Predicting Continuous Variables
2. Procedure to evaluate and select models.
   - Training Set, Validation Set, and Test Set
3. Cost Sensitive Classification
   - ROC and AUC?
   - How to train your model?

Evaluating Classifier Accuracy: Summary

- In our previous examples, we split the training dataset into two sets: one for training and one for testing.
- This may not be a good enough method to evaluate the performance of different algorithms.
- The reasons is that one algorithm may work well only on this particular split of training and testing dataset. Some algorithms (decision tree) are quite unstable across training sets.
- We need a better procedure to evaluate the classifier's performance => 10-fold cross validation method.

Evaluating Classifier Accuracy

K-fold Cross-validation is the solution

- Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
- *k*-fold: k = 10 is most popular
- At *i*-th iteration, use D_i as test set and others as training set

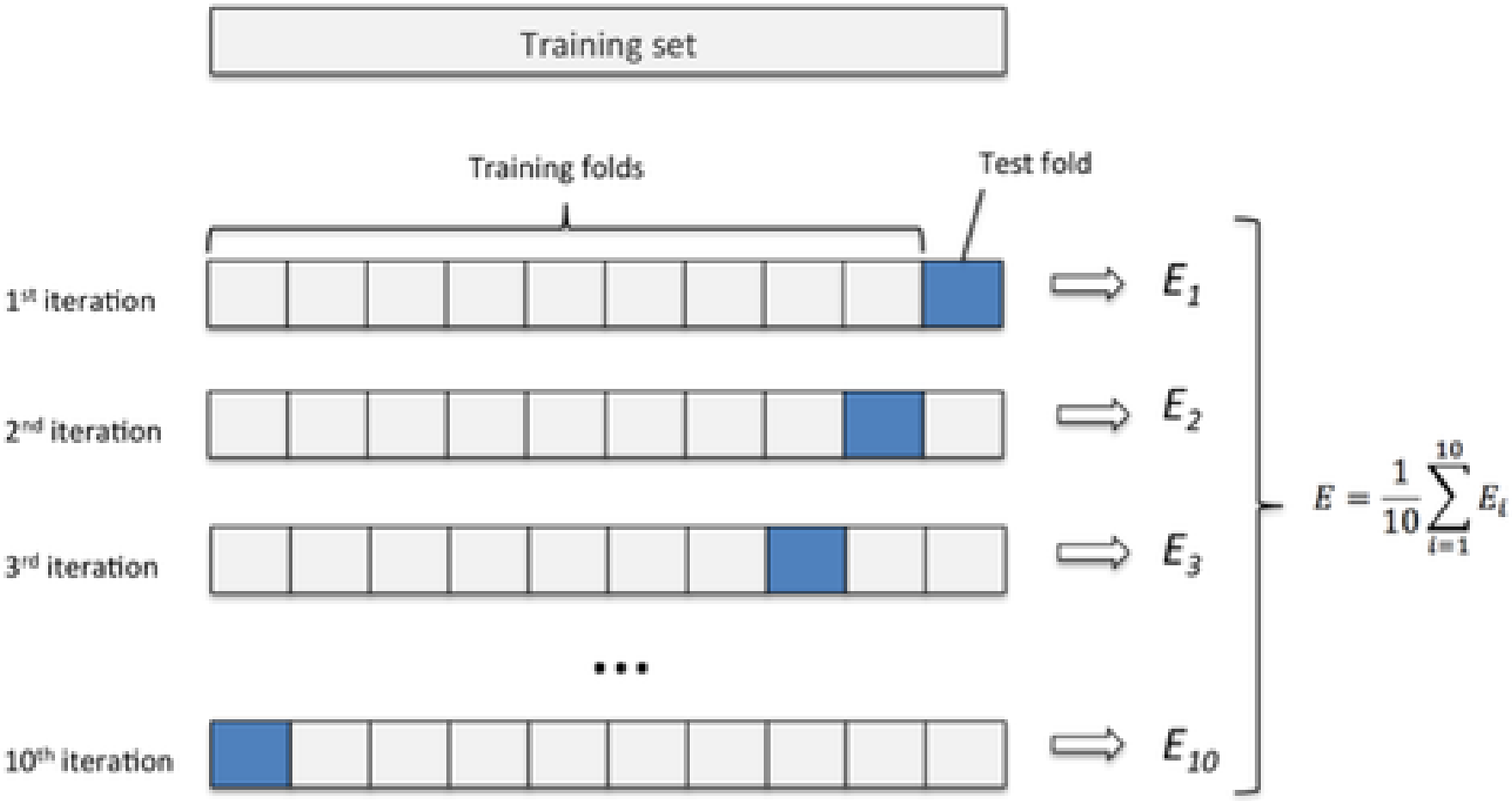
Two other famous methods:

- Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
- ***Stratified cross-validation***: folds are stratified so that label dist. in each fold is approx. the same as that in the initial data. For example, if your dataset's DV has 70% of 1, 30% of 0. In each fold, the DV should still has 70% of 1.

Cross-validation: Simplified approach

1. First step: split data into *k* subsets of equal size
2. Second step: **use each subset in turn for testing, the remainder for training**. For each fold, we try the same **one algorithm with one set of parameters**.
   a) Repeat this step *k* times for using each subset as the test set. The same procedure is applied to *k* different training sets with the same algorithm+ fixed parameters.
   b) After *k* iterations, you can calculate the accuracy or other performance metrics by all examples because each example is used in test set once.
   c) This step is to help you evaluate the "general performance" of an algorithm with a specific set of parameter values! You do not have a specific predictive model yet.

10 Fold Cross-Validation



Cross-validation: Simplified approach

3. Third Step: We repeat k-fold cross-validations (Step 2) many times by changing algorithms and changing parameters.

a) By the same algorithm, we try all possible combinations of parameter values. This is like we are tuning the parameters to find the best set of parameter values.

b) We also need to rotate to next algorithm and try all combinations of parameter values of the next algorithm.

4. After Step 3, we have the performance metrics of all algorithms with all reasonable combinations of parameters. Once evaluation is done and we decide the best algorithm and best parameters, ***all data*** **can be used to build the final classifier.**

More on cross-validation

- Why 10-fold? Extensive experiments have shown that this is the best choice to get an accurate estimate
- Stratified 10-fold cross-validation is theoretically the best but most people just use non-stratified approach.
- Stratification can reduce the estimate's variance
- Even better but uncommon: repeated stratified cross-validation
  - E.g., stratified ten-fold cross-validation is repeated M times and results are averaged (reduces the variance because of your splits of dataset)
  - But it takes too much training time because even 10-fold alone may take quite long for NN and SVM families of algorithms.

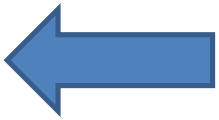
Leave-one-out cross-validation

- Leave-one-out:
  a particular form of *k*-fold cross-validation:
  - Set number of folds to number of training instances
  - I.e., for *n* training instances, build classifier *n* times
- Makes best use of the data
- **Involves no random subsampling**
- Very computationally expensive and thus this can be implemented only with very fast algorithms, not too large training dataset, or you have unlimited computing power.
- So in practice, we use 10-fold or even 5-fold cross-validation.

Today's Class

1. Criteria for Selecting Models.
   - Predicting Categories = Classification
   - Predicting Probabilities
   - Predicting Continuous Variables

2. Procedure to evaluate and select models.
   - Training Set, Validation Set, and Test Set

3. ROC and AUC 
   - Cost sensitive classification
   - Maximizing your objective function

Cost sensitive classification

- In practice, different types of classification errors often incur different costs.
- Examples:
  - Terrorist profiling: “Not a terrorist” correct 99.99…% of the time;
    - FP cost => depending on what is the next step; some innocent citizen will be harassed or not…. It could be costs of more detailed security check
    - FN cost => could be huge because of terrorists attack
  - Loan decisions: (default is 1) most won’t default;
    - FP => bank declines a profitable customer (only opportunity cost);
    - FN => bank lends to someone won’t repay (huge cost!)
  - Promotional mailing: response rate is low; (response is 1)
    - FP => waste of (e)mailing cost, which is small for email, not small for real mail with large quantities
    - FN => lost a potential customer.

Cost sensitive classification

- More Examples:
  - Healthcare: cancer rate is low; (cancer = 1)
    - FP => cost of further checking and mental cost of patient;
    - FN => someone will die
  - Faulty product diagnosis: defect/fault rate is low; (defect = 1)
    - FP=> further checking cost or throw away a good product; (it depends on the next step and the product's raw material cost)
    - FN=> allow a defect product to pass. (high for Apple/Toyota but low for white-brand manufacturers)

Counting the cost

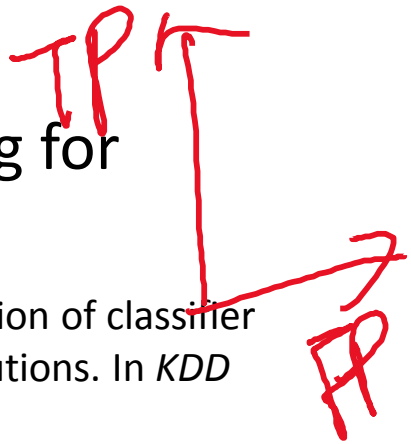
- The *confusion matrix again*:

		Predicted class	
		Yes	No
Actual class	Yes	True positive	False negative
	No	False positive	True negative

- Different misclassification costs can be assigned to false positives and false negatives
- Same idea can be generalized to 3 class types or more types.
- For binary DV like this, ROC curve is a very powerful and general method to consider FP and FN at the same time.

ROC curves

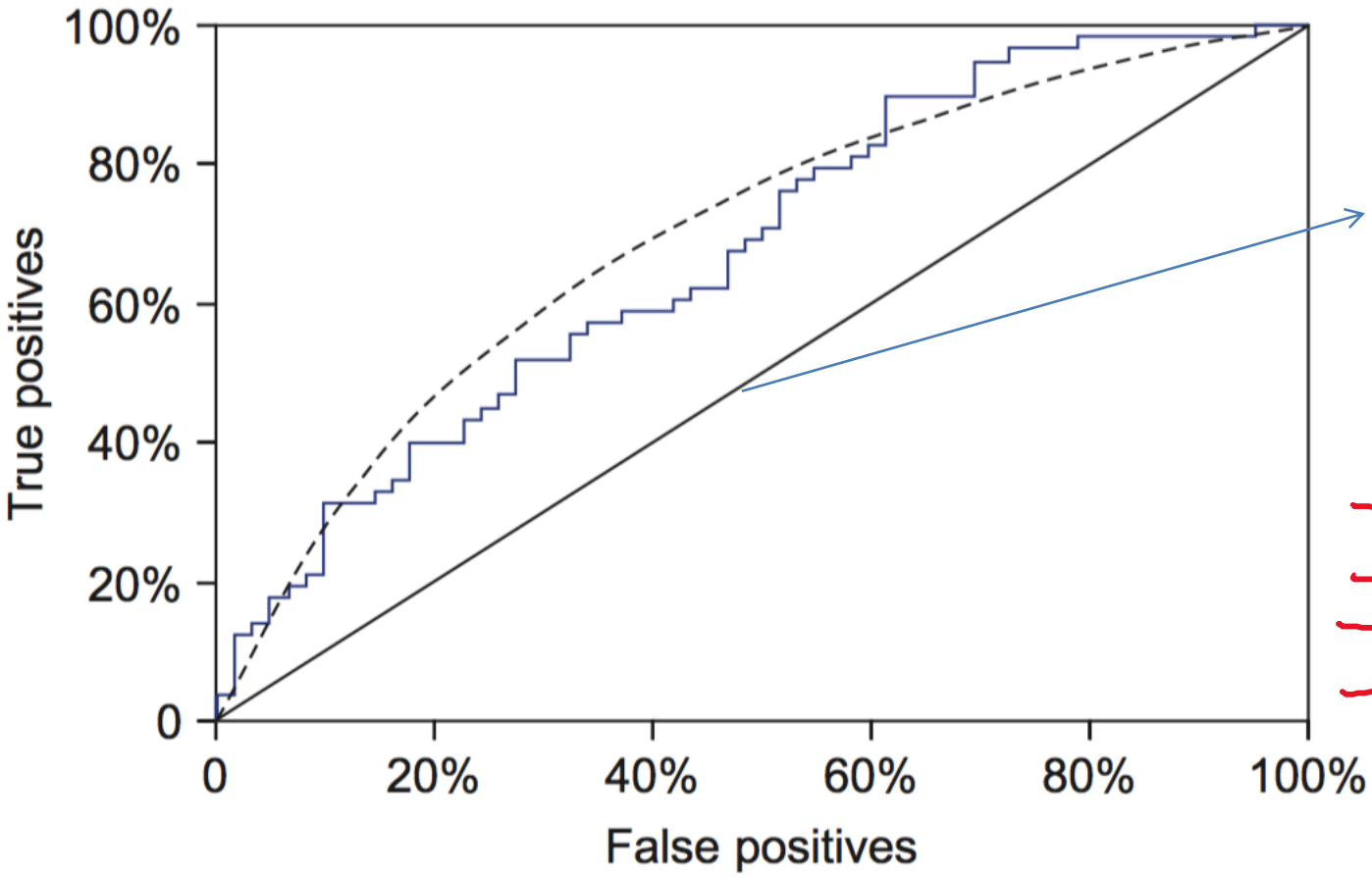
- *ROC* stands for "receiver operating characteristic".
  - It was originally used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel.
  - Later, it was introduced into the data mining for classification performance evaluation.
    - Provost, F. J., & Fawcett, T. (1997, August). Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In *KDD* (Vol. 97, pp. 43-48).
    - They are the authors of the optional textbook. Foster Provost was my professor at NYU who taught me PhD data mining.
- *y* axis shows percentage of **true positives** in test sample
- *x* axis shows percentage of **false positives** in test sample.
- Both are in %.



ROC curves

- One algorithm with fixed parameters gives you one point on this chart. If you plot one algorithm with one parameter changing, it is quite often you can observe a "ROC" curve.
- The best classifier has 100% TP and 0% FP (upper left).
- The worst classifier has 0% TP and 100% FP **(before flipping results)** (lower right).
- So the worst case actually cannot be worse than the 45 degree line; otherwise, your prediction results are so bad that you should simply flip your predicted values from 0 to 1 and 1 to 0, and the performance is better.

A sample ROC curve



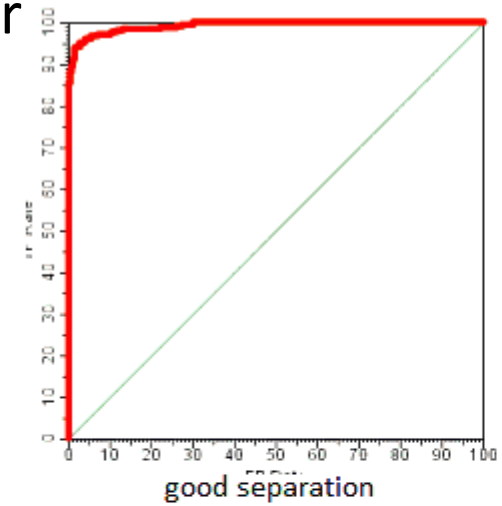
Random guess can produces a classifier at the 45 degree line.

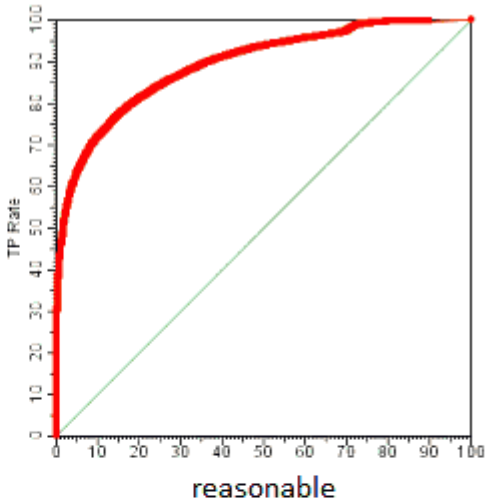
No model should be worse than this line, why?

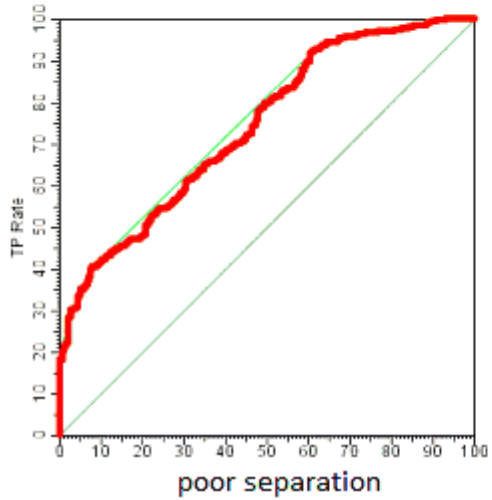
- Jagged curve—one set of test data
- Smoother curve—use cross-validation

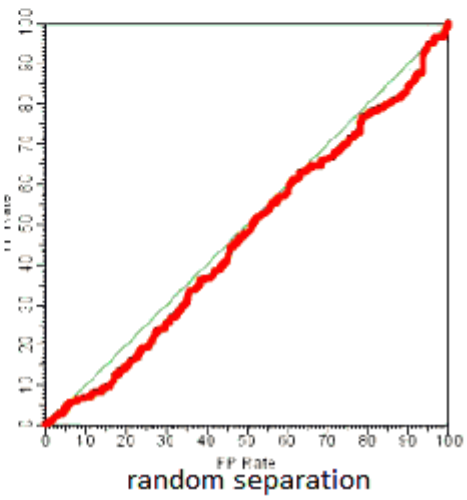
Area-Under-Curve (AUC)

- (Important) AUC is another well-known performance metric highly dependent on ROC.
- It is the area under the ROC curve.
- AUC under the 45% degree line is 0.5=50%.
- AUC under the perfect classifier (0 FP and 100% TP) is 1=100%.
- **What if your AUC is below 50%?**





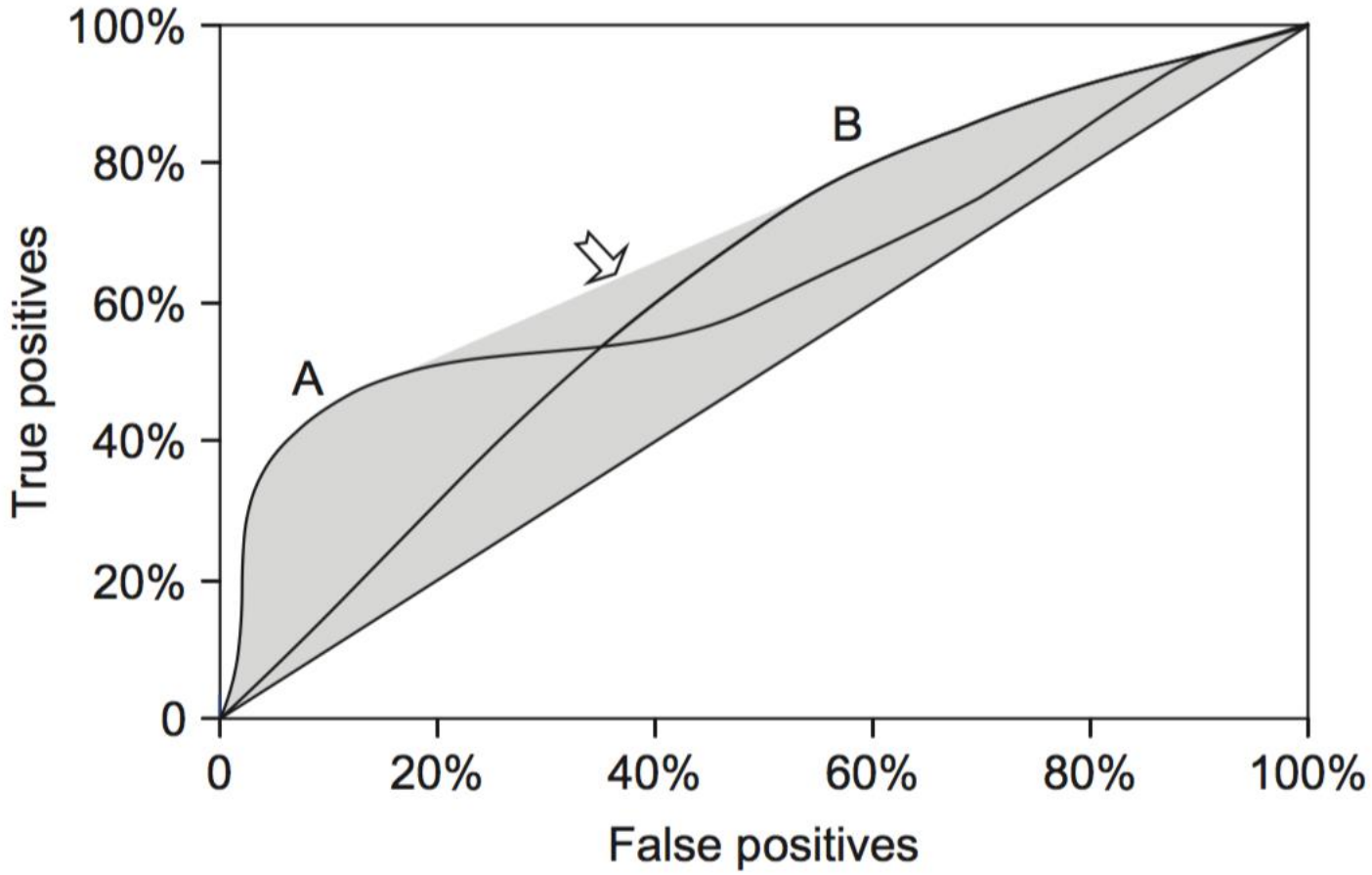




Predicted Probability and AUC

- AUC is one of the most common metric for assessing the prediction performance when your output is a predicted probability. Why?
- This is because we can use a threshold rule to convert predicted probability into binary dependent variable. For example, the default threshold is 50%. If the predicted probability p is > 50% than Y is predicted to be 1 and p<=50% is 0. Given the predicted Y as a binary variable, you can create a confusion matrix and get the values of FP, FN ⇔ one point on ROC.
- You can change "50%" to any other threshold between 0% and 100%. Each threshold value maps to one point on ROC. Changing from 0 to 100% gives you the ROC and AUC.
- AUC Is from 50% to 100%, can be interpreted easily as an percentage, similar to simple accuracy.

ROC curves from two classifiers



You may have multiple learning schemes and one dominates the others only in some intervals. The figure illustrates this kind of case of two learning schemes.

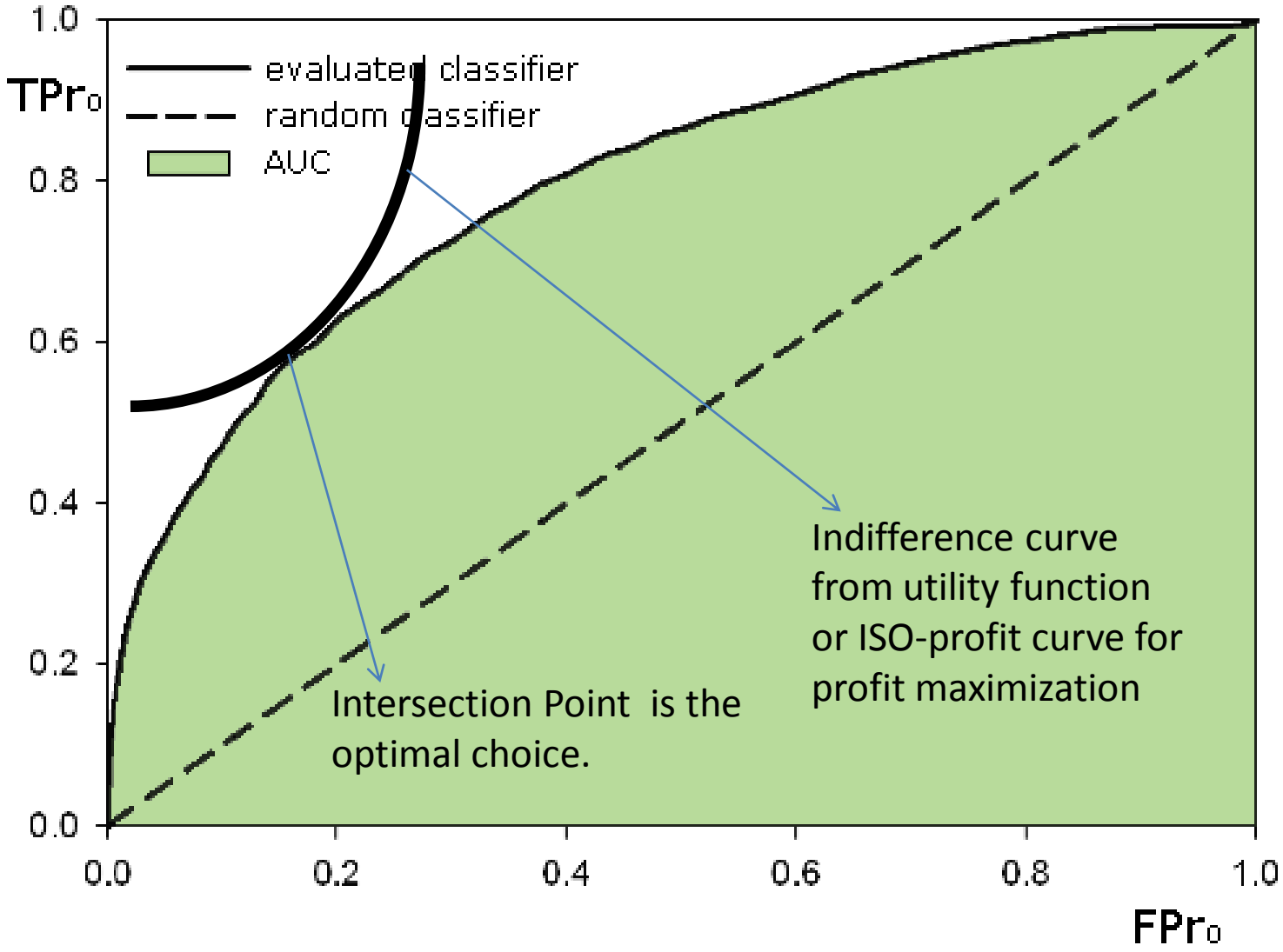
The Convex Hull

- Given two classifiers we can achieve any point on the convex hull (visualization of convex hull in the footnote)!
- So we should use the convex hull of multiple learning schemes for classification.
- TP and FP rates for scheme 1: t_1 and f_1
- TP and FP rates for scheme 2: t_2 and f_2
- If scheme 1 is used to predict q % of the cases and scheme 2 for the rest, then
  - TP rate for combined scheme:
    $q \times t_1 + (1\text{-}q) \times t_2$
  - FP rate for combined scheme:
    $q \times f_1 + (1\text{-}q) \times f_2$
- This is different from boosting or bagging or random forests that I will cover later this semester.

ROC Curve and Performance Optimization

- In general, if you conduct business analytics, you will have an "objective function" f(), which is a function of the number of all 4 cases TP, FP, TN, and FN.
- TP TN may bring you some values/profits.
- FP FN may incur tangible or intangible costs.
- You insert values of the benefits or costs of TP/FP/TN/FN into your objective function and then you can numerically decide which classifier is the best choice for your business analytics problem.
- If you have background in economics or Operations Research, then you can conceptually refer to the figure on the next page.

ROC Curve and Profit Maximization

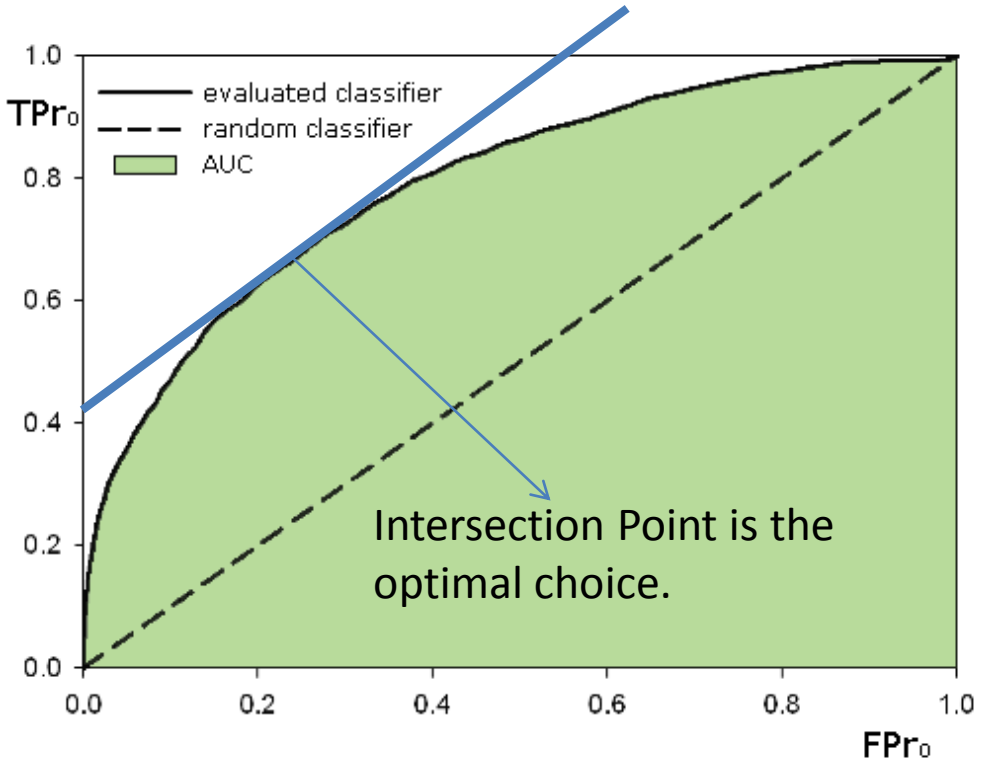


Cost-sensitive learning: Example

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- For example, for personal credit loan approval for banks and we label YES as the customer will pay back the loan. If the predicted value is NO, then decline the loan. If the predicted value is yes, we offer the loan.
- The profit of true positives is $15K SGD per one loan at $100K (implying 15% interest rate) for 1 year. If the customer defaults (FP), then the loss is $100K. Therefore, the objective function is 100*15K-10*100K=500K.
- This is because this equation is equal to
  (100/105)*(105*15K)-(10/60)*(60*100K)=500K
  $\Leftrightarrow$ TP * 105*15K – FP * 60 * 100K = 500K

ROC Curve and Profit Maximization



This is a good example of ROC in that if the bank is stricter in approving loan we will have smaller TP and FP. If the bank is less stricter in approving loans (by adjusting the threshold), we have larger TP and FP. The solution is the intercept of your ROC curve and straight line decided by the objective function

Appendix: Example of Caret Output

```
Confusion Matrix and Statistics

          Reference
Prediction  ham spam
      ham  1203   31
      spam    4  152

               Accuracy : 0.9748
                 95% CI : (0.9652, 0.9824)
    No Information Rate : 0.8683
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8825
 Mcnemar's Test P-Value : 1.109e-05

            Sensitivity : 0.8306
            Specificity : 0.9967
         Pos Pred Value : 0.9744
         Neg Pred Value : 0.9749
             Prevalence : 0.1317
         Detection Rate : 0.1094
   Detection Prevalence : 0.1122
      Balanced Accuracy : 0.9136
```

Textbook page #322

Appendix: Example of Caret Output

Predicted	Event	No Event
Event	A	B
No Event	C	D

The formulas used here are:

$$Sensitivity = A/(A+C)$$

$$Specificity = D/(B+D)$$

$$Prevalence = (A+C)/(A+B+C+D)$$

$$PPV = (sensitivity * prevalence)/((sensitivity * prevalence) + ((1-specificity) * (1-prevalence)))$$

$$NPV = (specificity * (1-prevalence))/(((1-sensitivity) * prevalence) + ((specificity) * (1-prevalence)))$$

$$DetectionRate = A/(A+B+C+D)$$

$$DetectionPrevalence = (A+B)/(A+B+C+D)$$

$$BalancedAccuracy = (sensitivity + specificity)/2$$

$$Precision = A/(A+B)$$

$$Recall = A/(A+C)$$

$$F1 = (1+beta^2) * precision * recall/((beta^2 * precision) + recall)$$

From https://www.rdocumentation.org/packages/caret/versions/6.0-78/topics/confusionMatrix