Data Science – Semester 5 – Fall 2020/2021

# INTRODUCTION TO DATA MINING & WAREHOUSING

## Lecture 6: Classification Performance Evaluation

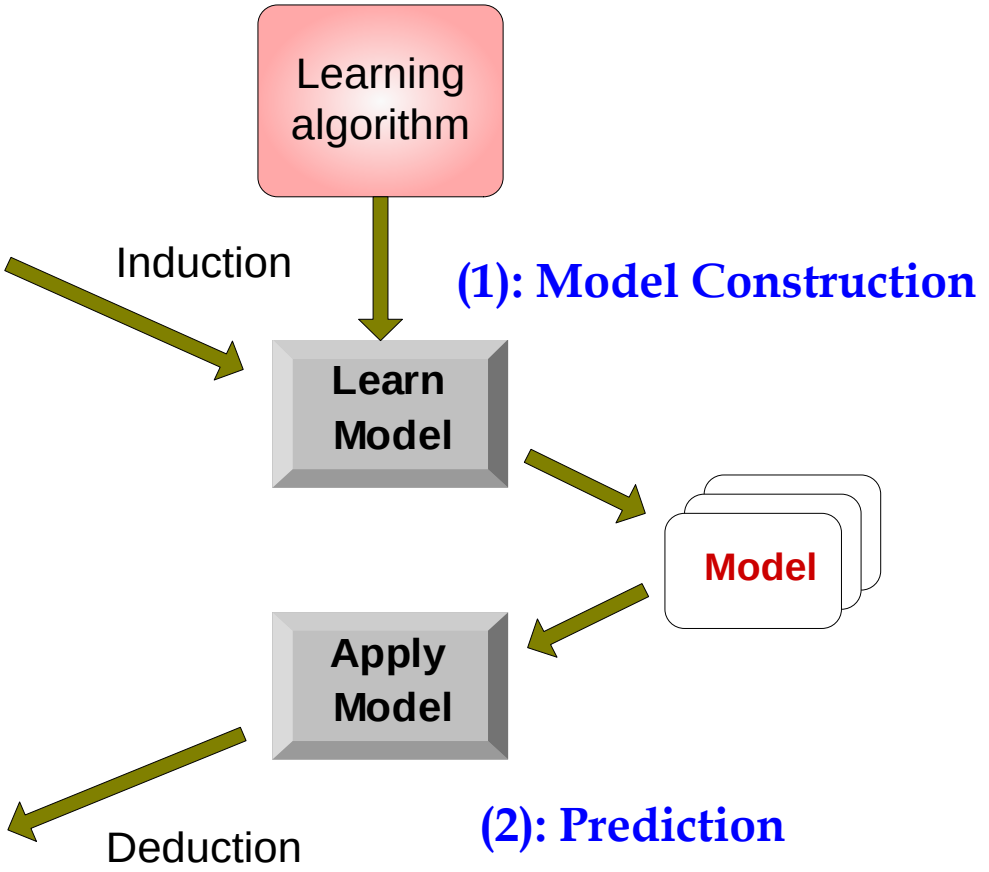
# General Approach for Building Classification Model

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



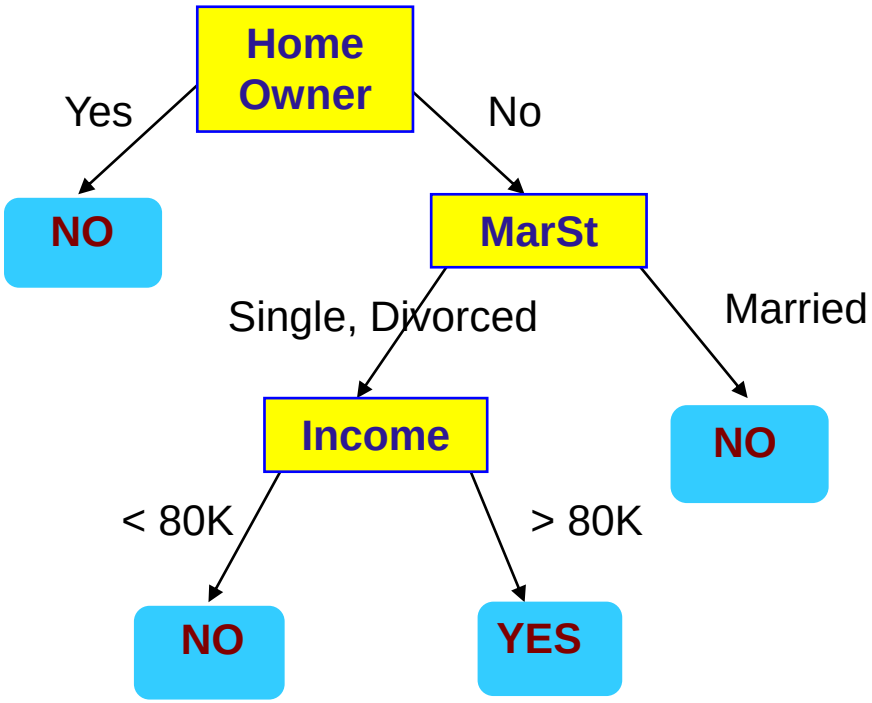
# What is the testing phase?

- Measures the performance of the resulting classifier to estimate how accurately it will perform in practice.

- Why evaluate:
  - Multiple methods are available to classify or predict
  - For each method, multiple choices are available for settings
  - To choose best model, need to assess each model's performance

# Underfitting/Overfitting

Decision tree learned from the training data



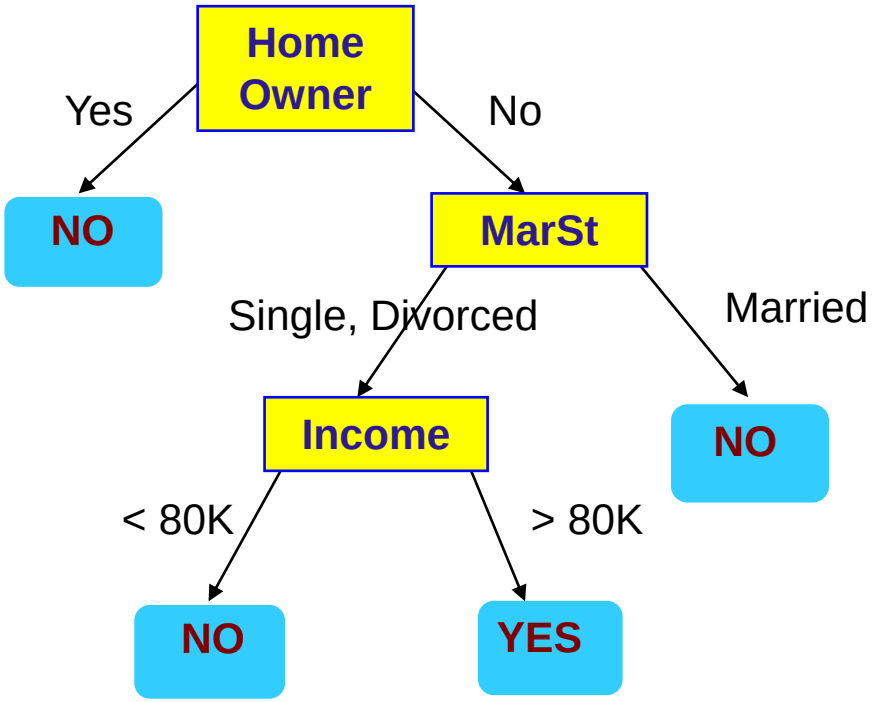

Training Data

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

**Poor performance on the training data results in Underfitting: the input features are not expressive enough to describe the target well.**

# Underfitting/Overfitting

Decision tree learned from the training data




Test Data

| ID | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | Yes |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | Yes |
| 5 | No | Divorced | 95K | Yes |

**Poor performance on the test data results in Overfitting: the model performs well on the training data but does not perform well on the test data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples.**

# Model Evaluation

- How reliable are the predicted results?
- How much should we believe in what was learned?
- Error on the training data is not a good indicator of performance on future data
  - The classifier was computed from the very same training data, any estimate based on that data will be optimistic
  - In addition, new data will probably not be exactly the same as the training data!

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

# Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- **Confusion Matrix (Binary Classification):**

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | **TP** | **FN** |
| | Class=No | **FP** | **TN** |

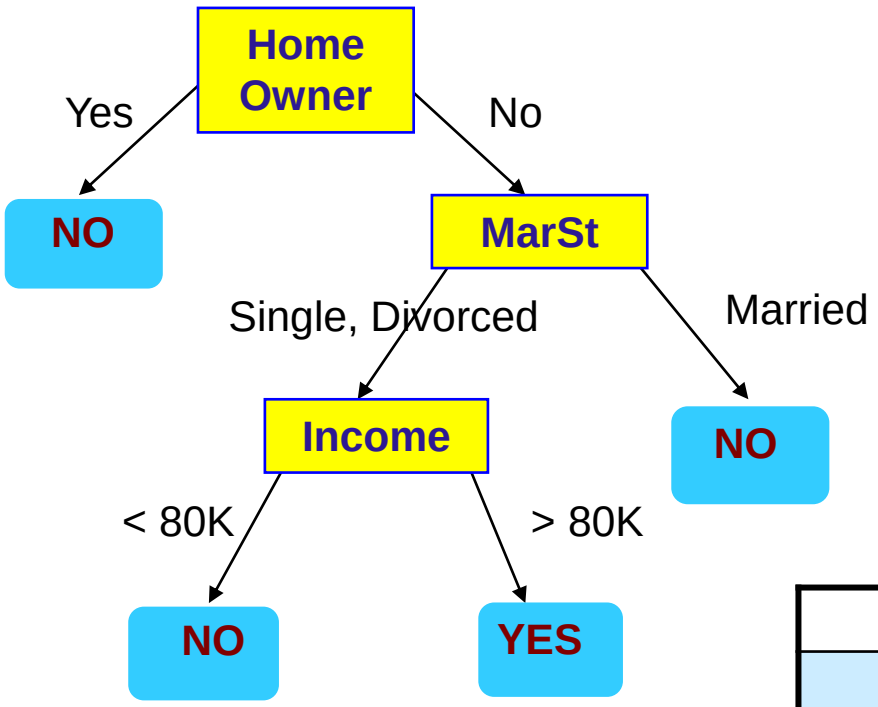
TP (true positive)

FN (false negative)

FP (false positive)

TN (true negative)

# Example




## Test Data

| Home Owner | Marital Status | Annual Income | Defaulted Borrower |
|---|---|---|---|
| No | Married | 80K | **Yes** |
| No | Single | 75k | **No** |
| Yes | Single | 85k | **Yes** |
| Yes | Married | 90k | **No** |
| Yes | Divorced | 79k | **No** |

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | 2 | 2 |
| | Class=No | 0 | 1 |

# Metrics for Performance Evaluation

- Most widely-used metric:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | Class=Yes | a (TP) | b (FN) |
| | Class=No | c (FP) | d (TN) |

# Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

# Cost Matrix

| | | PREDICTED CLASS | |
|---|---|---|---|
| | | **Class=Yes** | **Class=No** |
| ACTUAL CLASS | **Class=Yes** | C(TP) = w1 | C(FN) = w2 |
| | **Class=No** | C(FP) = w3 | C(TN) = w4 |

**C(x):** Cost of misclassifying examples of type x

$$\text{Cost} = w_1 a + w_2 b + w_3 c + w_4 d$$

# Computing Cost of Classification

| Cost Matrix | PREDICTED CLASS | | |
|---|---|---|---|
| | C(i\|j) | + | - |
| ACTUAL CLASS | + | -1 | 100 |
| | - | 1 | 0 |

| Model $M_1$ | PREDICTED CLASS | | |
|---|---|---|---|
| | | + | - |
| ACTUAL CLASS | + | 150 | 40 |
| | - | 60 | 250 |

Accuracy = 80%

Cost = 3910

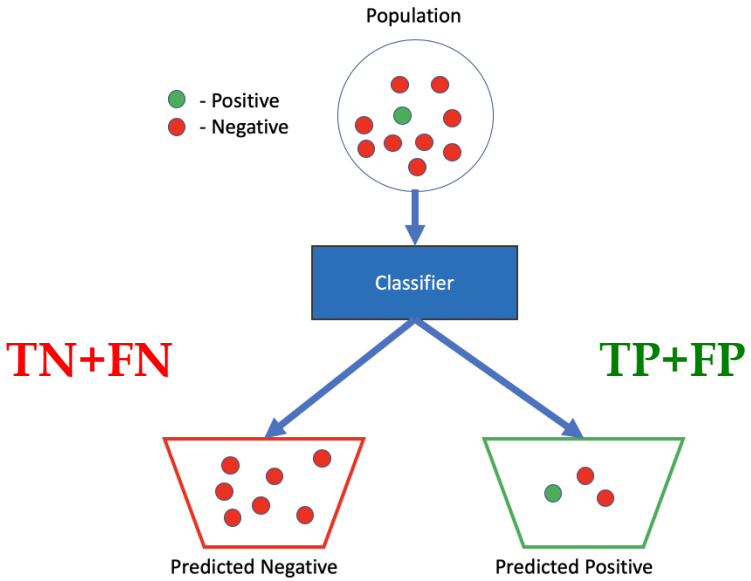
| Model $M_2$ | PREDICTED CLASS | | |
|---|---|---|---|
| | | + | - |
| ACTUAL CLASS | + | 250 | 45 |
| | - | 5 | 200 |

Accuracy = 90%

Cost = 4255

# Precision and Recall

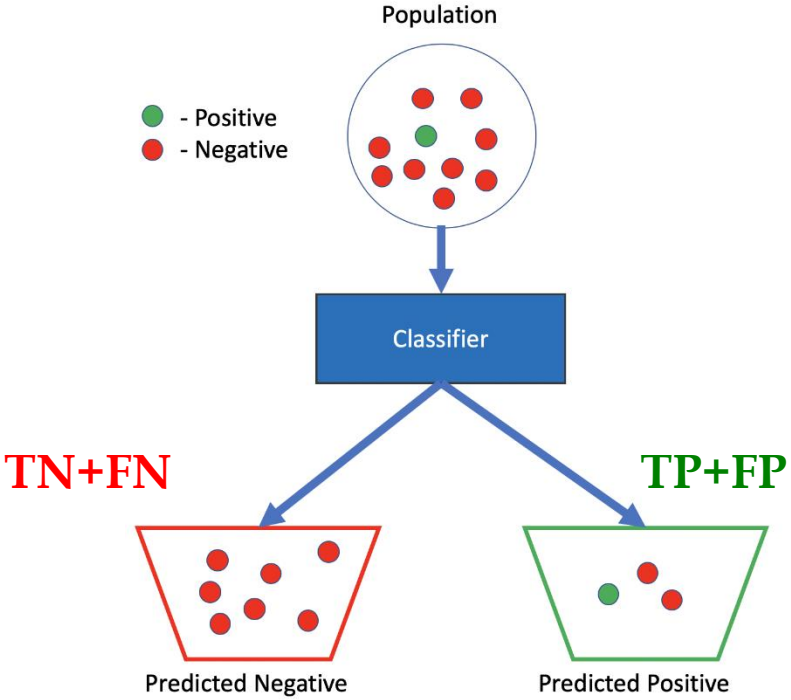
- Alternatives to accuracy, introduced in the area of information retrieval and search engine
- Precision refers to the percentage of predictions that are correct
- Recall refers to the percentage of total relevant results correctly classified by your algorithm.




| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=+ | Class=- |
| ACTUAL CLASS | Class=+ | 1 (TP) | 0 (FN) |
| | Class=- | 2 (FP) | 7 (TN) |

# Precision and Recall

- Precision refers to the percentage of predictions that are correct
- Recall refers to the percentage of total relevant results correctly classified by your algorithm.

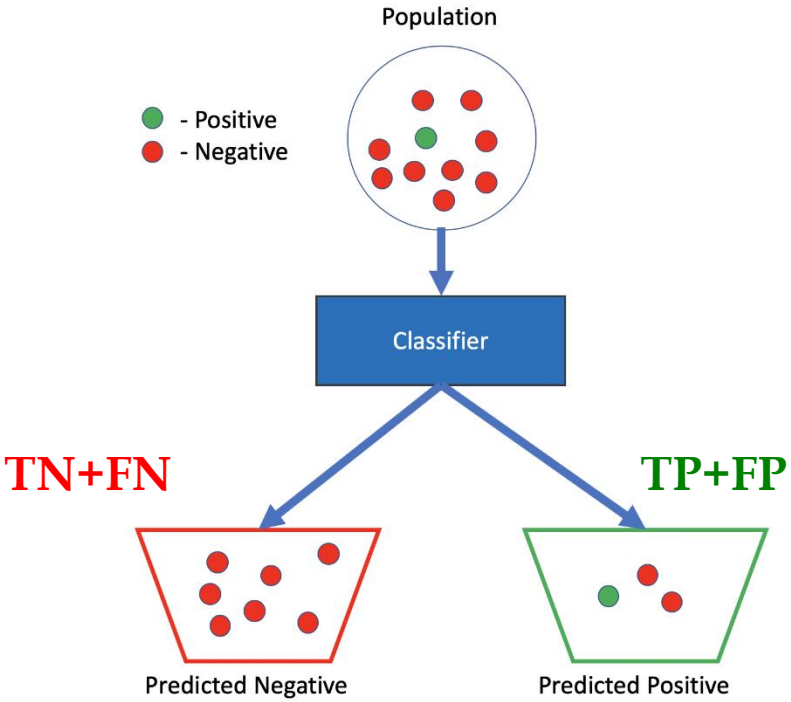



| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=+ | Class=- |
| ACTUAL CLASS | Class=+ | 1 (TP) | 0 (FN) |
| | Class=- | 2 (FP) | 7 (TN) |

**Low precision (1/3): if the classifier predicts true, we can't be sure, because 2/3 could be FP (so negative in reality)**

# Precision and Recall

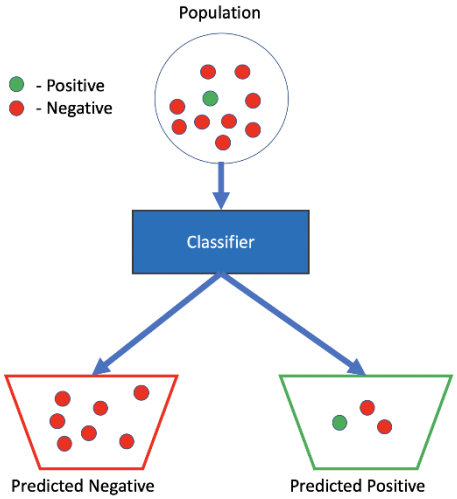
- Precision refers to the percentage of predictions that are correct
- Recall refers to the percentage of total relevant results correctly classified by your algorithm.




| | | PREDICTED CLASS | |
|---|---|---|---|
| | | Class=+ | Class=- |
| ACTUAL CLASS | Class=+ | 1 (TP) | 0 (FN) |
| | Class=- | 2 (FP) | 7 (TN) |

**High Recall (100%): All positive examples are correctly classified. If the example is positive, you can trust your classifier.**

# Precision, Recall, F1-score



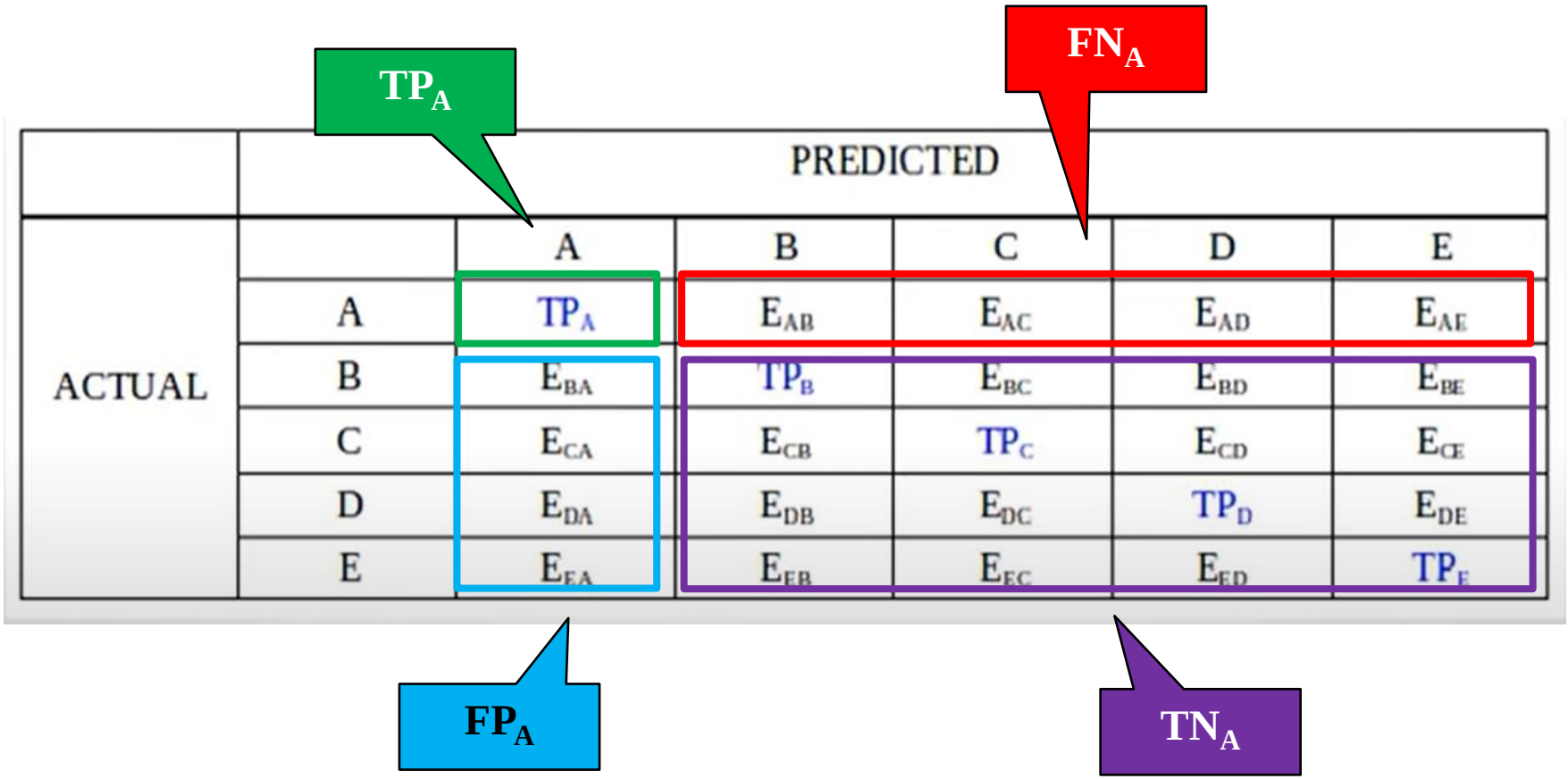

- precision = TP/(TP+FP)
- recall = TP/(TP+FN)
- F1-score = (2 x precision recall)/(precision + recall)
  (harmonic mean of precision and recall)

- Precision is biased towards TP & FN
- Recall is biased towards TP & FP
- F-measure is biased towards all except TN

| | | predicted class + | predicted class - | |
|---|---|---|---|---|
| actual class | + | TP | FN | P = TP+FN |
| actual class | - | FP | TN | N = FP+TN |
| | | P' = TP+FP | N' = FN+TN | K = TP+FN+FP+TN |

# Confusion Matrix for Multi-Classification

- Let's assume a 5-class problem with the classes A,B,C,D and E and following confusion matrix



# Confusion Matrix for Multi-Classification

- $TP_B$, $FN_B$, $FP_B$, $TN_B$

| | | PREDICTED | | | | |
|---|---|---|---|---|---|---|
| | | A | B | C | D | E |
| ACTUAL | A | $TP_A$ | $E_{AB}$ | $E_{AC}$ | $E_{AD}$ | $E_{AE}$ |
| | B | $E_{BA}$ | $TP_B$ | $E_{BC}$ | $E_{BD}$ | $E_{BE}$ |
| | C | $E_{CA}$ | $E_{CB}$ | $TP_C$ | $E_{CD}$ | $E_{CE}$ |
| | D | $E_{DA}$ | $E_{DB}$ | $E_{DC}$ | $TP_D$ | $E_{DE}$ |
| | E | $E_{EA}$ | $E_{EB}$ | $E_{EC}$ | $E_{ED}$ | $TP_E$ |

# Confusion Matrix for Multi-Classification

- The total number of test examples of any class would? (TP+FN)
- Sum of FN of a class? (sum of the row excluding TP)
- Sum of FP of a class? (sum of column excusing TP)
- Sum of TN of a class? (sum of everything excluding row and column of that class)

| | | PREDICTED | | | | |
|---|---|---|---|---|---|---|
| | | A | B | C | D | E |
| ACTUAL | A | $TP_A$ | $E_{AB}$ | $E_{AC}$ | $E_{AD}$ | $E_{AE}$ |
| | B | $E_{BA}$ | $TP_B$ | $E_{BC}$ | $E_{BD}$ | $E_{BE}$ |
| | C | $E_{CA}$ | $E_{CB}$ | $TP_C$ | $E_{CD}$ | $E_{CE}$ |
| | D | $E_{DA}$ | $E_{DB}$ | $E_{DC}$ | $TP_D$ | $E_{DE}$ |
| | E | $E_{EA}$ | $E_{EB}$ | $E_{EC}$ | $E_{ED}$ | $TP_E$ |

$TP_A+FN_A$

Sum $FP_A$

# Performance metrics for multi-classification

- Accuracy: sum of correct classifications divided by the total number of classifications

- Accuracy of a specific class (e.g. class A)

$$\text{Accuracy}_A = \frac{TP_A + TN_A}{TP_A + TN_A + FP_A + FN_A}$$

- Overall Accuracy:

$$\text{Accuracy} = \frac{TP_A + TP_B + TP_C + TP_D + TP_E}{\text{total number of classifications}}$$

# Performance metrics for multi-classification

- Precision, recall, F1-score: computed for a specific class
- $\text{precision}_A = TP_A/(TP_A+FP_A)$
- $\text{recall}_A = TP_A/(TP_A+FN_A)$
- $\text{F1-score}_A = (2 \times \text{precision}_A \times \text{recall}_A)/(\text{precision}_A + \text{recall}_A)$

# Example

- Overall accuracy: (25+32+15)/(25+5+2+3+32+4+1+15)

- $\text{precision}_B$: (TPB)/(TPB+FPB)
- 32/(32+5+0)

Predicted

| Actual | | A | B | C |
|---|---|---|---|---|
| | A | 25 | 5 | 2 |
| | B | 3 | 32 | 4 |
| | C | 1 | 0 | 15 |

- $\text{recall}_B$: TPB/TPB+FNB

32/(32+3+4)

- F-measure =

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

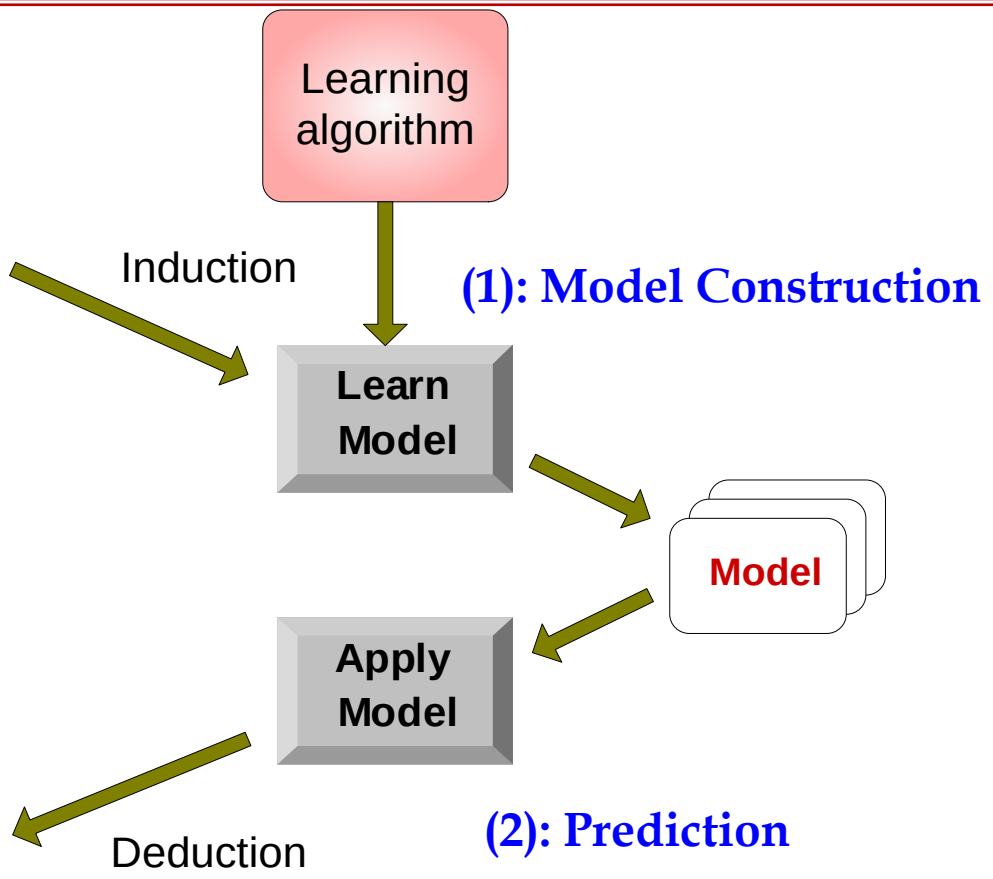
# How to get training/test sets?

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 1 | Yes | Large | 125K | No |
| 2 | No | Medium | 100K | No |
| 3 | No | Small | 70K | No |
| 4 | Yes | Medium | 120K | No |
| 5 | No | Large | 95K | Yes |
| 6 | No | Medium | 60K | No |
| 7 | Yes | Large | 220K | No |
| 8 | No | Small | 85K | Yes |
| 9 | No | Medium | 75K | No |
| 10 | No | Small | 90K | Yes |

Training Set

| *Tid* | Attrib1 | Attrib2 | Attrib3 | Class |
|---|---|---|---|---|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set



# Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?
  - The purpose is model checking, not model building
- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

# Methods of Estimation

- Holdout
  - Reserve **2/3** for training and **1/3** for testing
- Random subsampling
  - One sample may be biased -- Repeated holdout
- Cross validation
  - Partition data into **k** disjoint subsets
  - **k**-fold: train on **k-1** partitions, test on the remaining one
  - Leave-one-out: **k=n**
    - Guarantees that each record is used the same number of times for training and testing
- Bootstrap (read section 8.5.4 in the textbook)
  - Sampling with replacement
  - ~63% of records used for training, ~27% for testing

# Holdout evaluation and "small datasets"

- Reserves a certain amount for testing and uses the remainder for training, typically
  - Reserve ½ for training and ½ for testing
  - Reserve 2/3 for training and 1/3 for testing
- For small or "unbalanced" datasets, samples might not be representative
  - For instance, it might generate training or testing datasets with few or none instances of some classes
- Stratified sampling
  - Makes sure that each class is represented with approximately equal proportions in both subsets

# Repeated Holdout

- Holdout estimate can be made more reliable by repeating the process with different subsamples
- In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
- The error rates on the different iterations are averaged to yield an overall error rate
- Still not optimum since the different test sets overlap

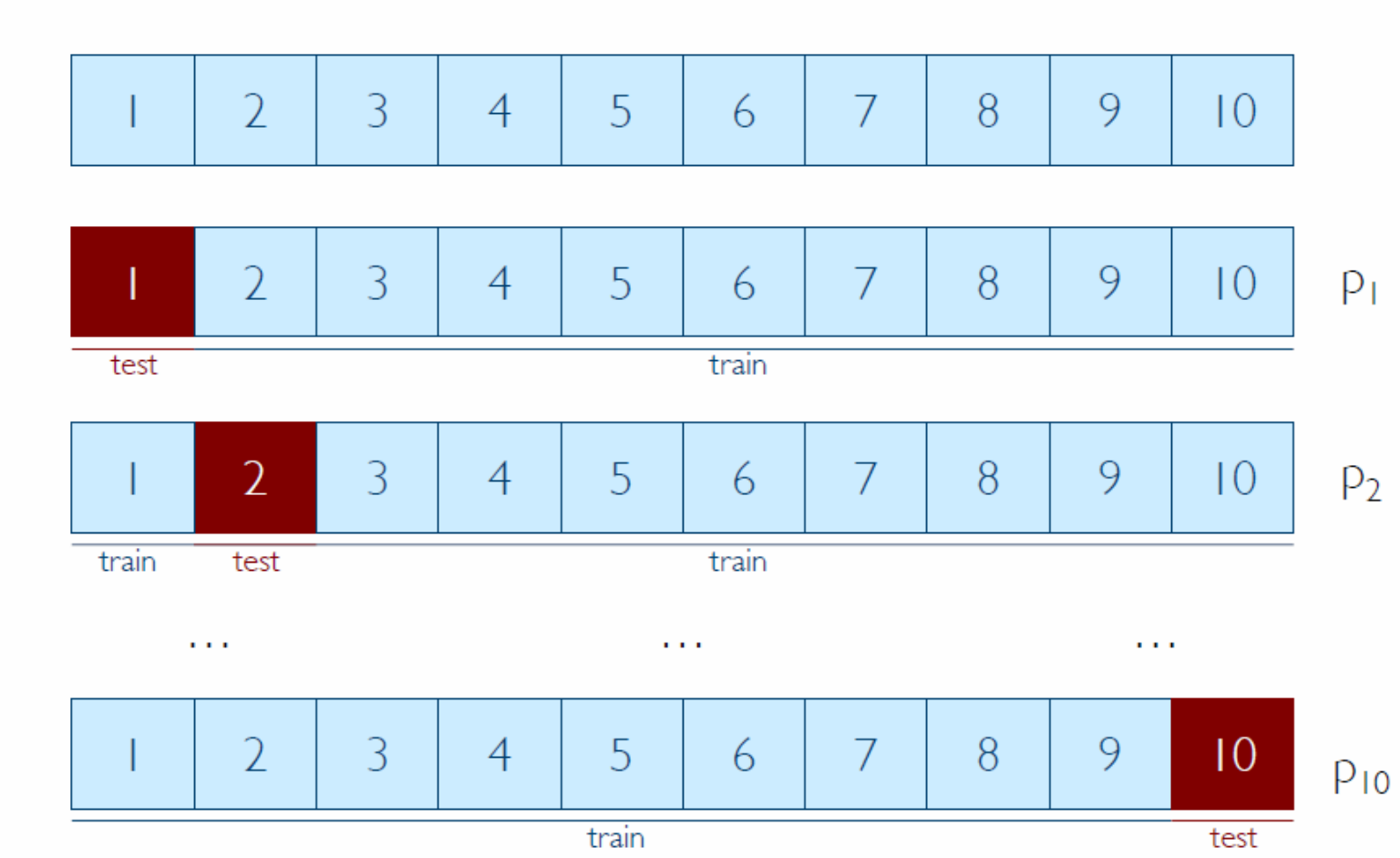
# Cross Validation

- First step
  - Data is split into k subsets of equal size
- Second step
  - Each subset in turn is used for testing and the remainder for training

- This is called k-fold cross-validation and avoids overlapping test sets
  - Often the subsets are stratified before cross-validation is performed

- The error estimates are averaged to yield an overall error estimate

# 10-fold Cross Validation




**The final performance is computed as the average pi**

# Cross Validation

- Standard method for evaluation is stratified ten-fold cross-validation
- Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate

- Even better: repeated stratified cross-validation
- E.g. 10-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

- Other approaches appear to be robust, e.g., 5x2 cross validation

# Leave-One-Out Cross-Validation

- A particular form of 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
- Computationally expensive

**Once evaluation is complete, all the data can be used to build the final classifier**

# Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

# ROC and AUC

- Assume we have a probabilistic prediction
  - How to compare the performance with different thresholds?

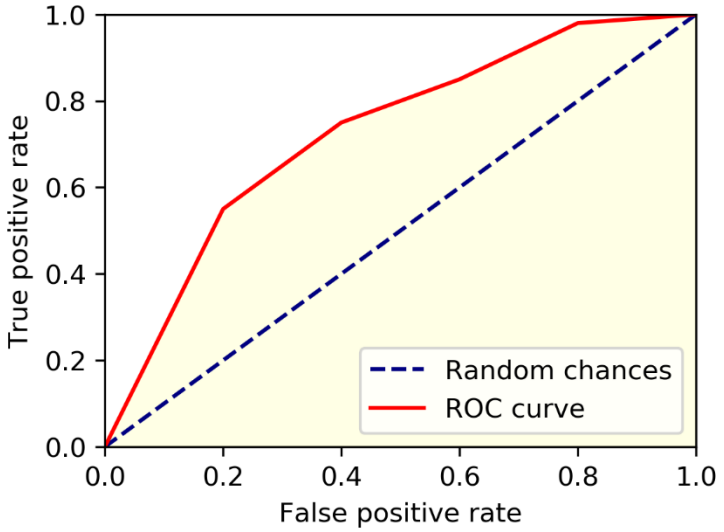
| ID | Actual | Prediction Probability | >0.6 | >0.7 | > 0.8 |
|---|---|---|---|---|---|
| 1 | 0 | 0.98 | 1 | 1 | 1 |
| 2 | 1 | 0.67 | 1 | 0 | 0 |
| 3 | 1 | 0.58 | 0 | 0 | 0 |
| 4 | 0 | 0.78 | 1 | 1 | 0 |
| 5 | 1 | 0.85 | 1 | 1 | 1 |
| 6 | 0 | 0.86 | 1 | 1 | 1 |
| 7 | 0 | 0.79 | 1 | 1 | 0 |
| 8 | 0 | 0.89 | 1 | 1 | 1 |
| 9 | 1 | 0.82 | 1 | 1 | 1 |
| 10 | 0 | 0.86 | 1 | 1 | 1 |

- We can generate different confusion matrices for different threshold cutoffs and compare the various metrics
- Instead, we can generate a plot between some of these metrics so that we can easily visualize which threshold is giving us a better result.

# ROC and AUC

- ROC: Receiver Operating Characteristics
- AUC: Area Under Curve

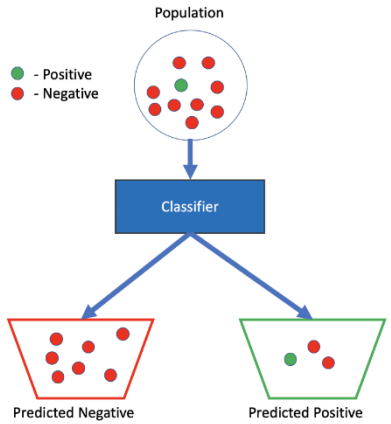
- **ROC** curve plots **True Positive Rate TPR** (on the **y**-axis) against **False Positive Rate FPR** (on the **x**-axis) at **different classification thresholds**

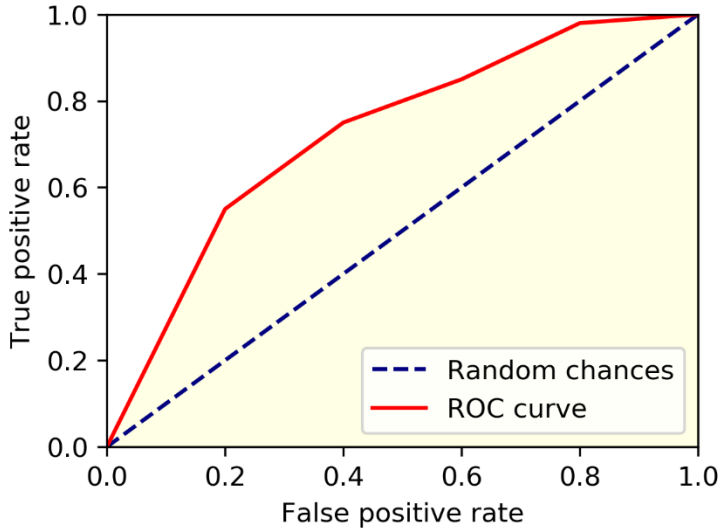


# ROC and AUC

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

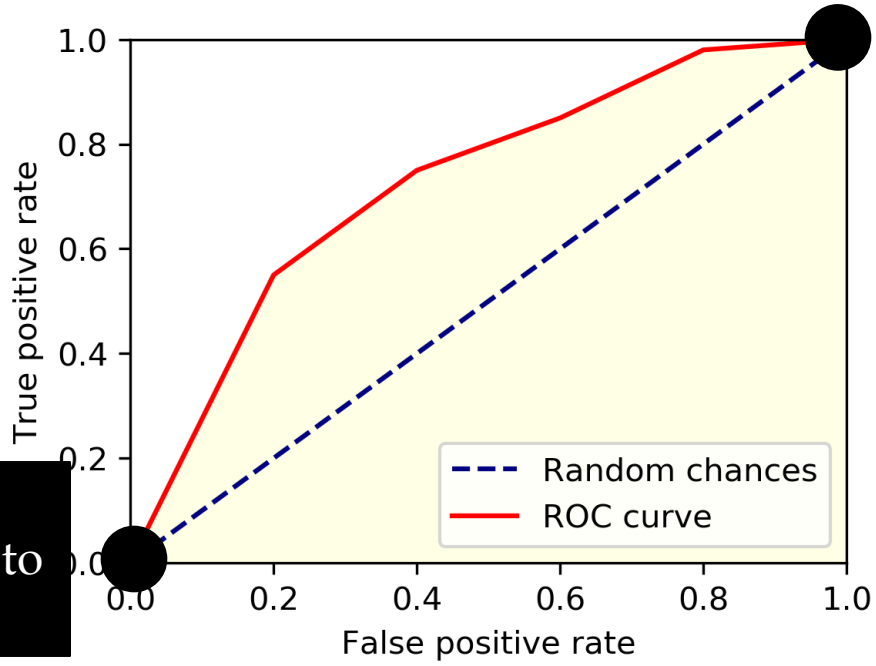
$$FPR = \frac{FP}{FP + TN}$$



# ROC and AUC

- Ideally the curve will climb quickly toward the top-left meaning the model correctly predicted the cases
- We want the ROC curve to be as far as possible from the blue line (random classifier)



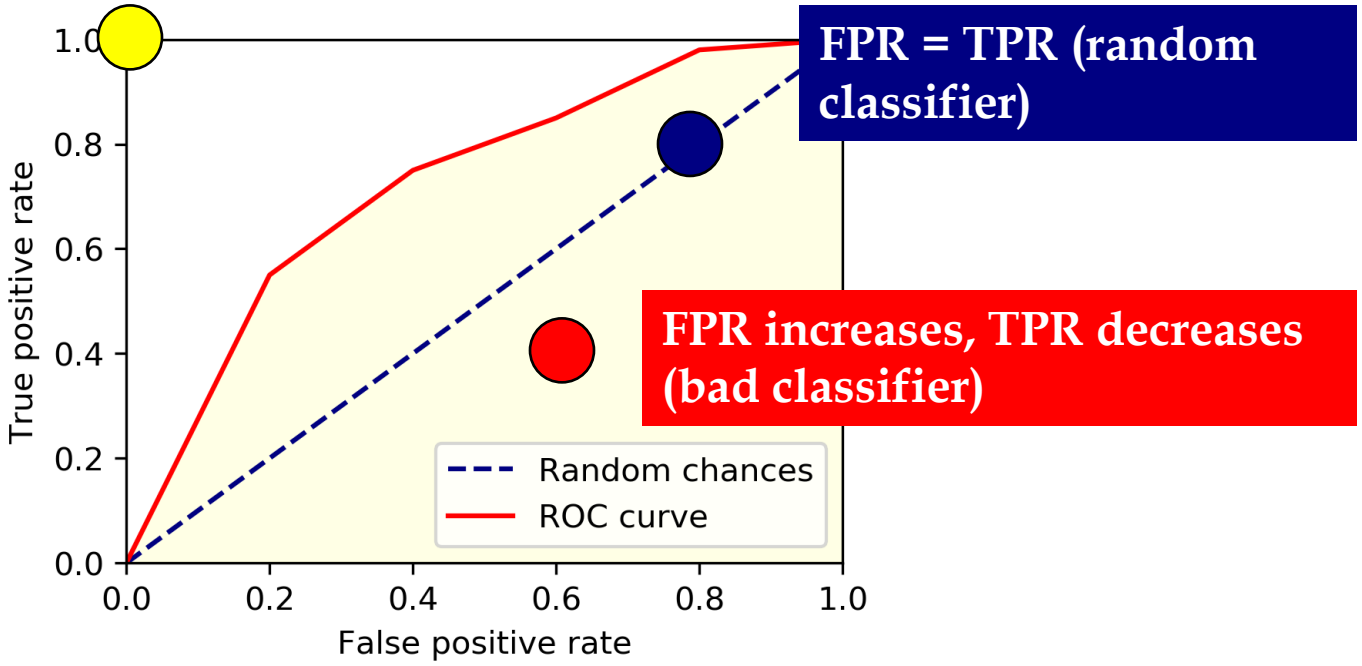

TPR = 1, FPR = 1
declare everything to be positive

TPR = 0, FPR = 0
declare everything to be negative

# ROC and AUC

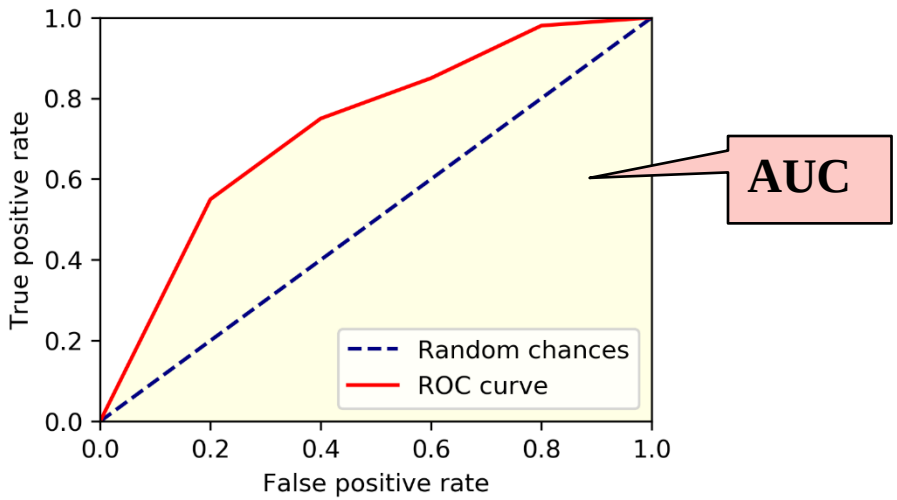
- Ideally the curve will climb quickly toward the top-left meaning the model correctly predicted the cases
- We want the ROC curve to be as far as possible from the blue line (random classifier)

TPR = 1, FPR = 0 (perfect classifier)



# ROC and AUC

- The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve
  - provides an aggregate measure of performance across all possible classification thresholds.
  - The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.
  - AUC ranges in value from 0 to 1. A model whose predictions are **100% wrong** has an **AUC of 0**; one whose predictions are **100% correct** has an **AUC of 1**.

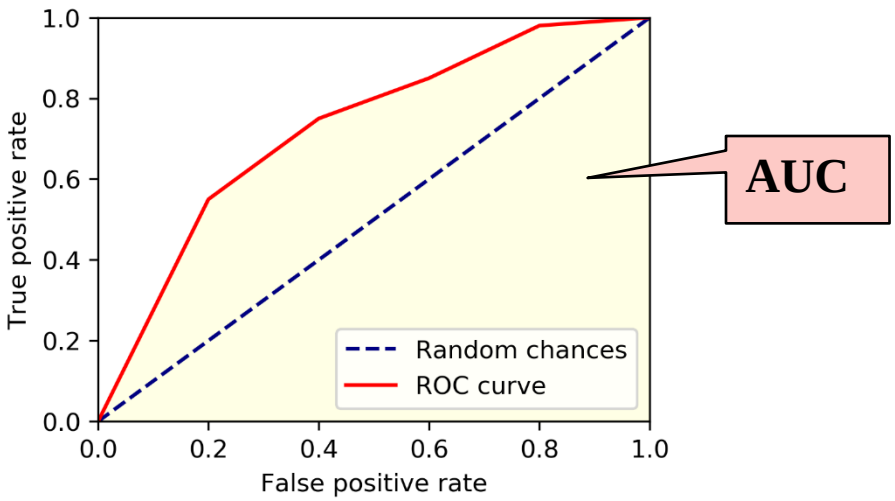


# ROC and AUC

- An **AUC of 0.8** means that a randomly selected case from the group with predicted class=Y (positive) has a score larger than that for a randomly chosen case from the group with predicted class= N (negative) in 80% of the time

- When a classifier **cannot distinguish** between the two groups, **AUC = 0.5** (ROC curve coincides with the diagonal)

- When there is a **perfect separation** between the two groups (no overlapping of the distributions), **AUC reaches 1** (ROC curve reaches the upper left corner)



# How to Construct a ROC curve

- You have your actual and predicted
- Internally, the model generate prediction probability according to a cut-off threshold

| inst | actual | predicted | | Prob. Y | Prob. N |
|---|---|---|---|---|---|
| 1 | Y | N | | 0.35 | 0.65 |
| 2 | N | N | | 0.23 | 0.77 |
| 3 | N | Y | | 0.55 | 0.45 |
| 4 | Y | N | | 0.32 | 0.68 |
| 5 | Y | Y | | 0.54 | 0.46 |
| 6 | N | N | | 0.47 | 0.53 |

# How to Construct a ROC curve

1. Sort data according to the probability of Positive class

| inst | actual | predicted | | Prob. Y | Prob. N |
|---|---|---|---|---|---|
| 1 | Y | N | | 0.35 | 0.65 |
| 2 | N | N | | 0.23 | 0.77 |
| 3 | N | Y | | 0.55 | 0.45 |
| 4 | Y | N | | 0.32 | 0.68 |
| 5 | Y | Y | | 0.54 | 0.46 |
| 6 | N | N | | 0.47 | 0.53 |

| actual | Prob Y |
|---|---|
| N | 0.55 |
| Y | 0.54 |
| N | 0.47 |
| Y | 0.35 |
| Y | 0.32 |
| N | 0.23 |

# How to Construct a ROC curve

2. Compute TPR (x-axis) and FPR y-axis) for different cutoff-threshold

| inst | actual | predicted | | Prob. Y | Prob. N |
|---|---|---|---|---|---|
| 1 | Y | N | | 0.35 | 0.65 |
| 2 | N | N | | 0.23 | 0.77 |
| 3 | N | Y | | 0.55 | 0.45 |
| 4 | Y | N | | 0.32 | 0.68 |
| 5 | Y | Y | | 0.54 | 0.46 |
| 6 | N | N | | 0.47 | 0.53 |

| actual | Prob Y |
|---|---|
| N | 0.55 |
| Y | 0.54 |
| N | 0.47 |
| Y | 0.35 |
| Y | 0.32 |
| N | 0.23 |

**Classified as Y**

**Classified as N**

- **For a cutoff 0.5:**
  - 1 Y classified as Y
  - 1 N classified as Y
  - 2 N's classified as N
  - 2 Y's classified as N

| | | Actual Y | N |
|---|---|---|---|
| Predicted | Y | 1 | 1 |
| | N | 2 | 2 |

TPR = TP/TP+FN = 1/1+2 = 0.33

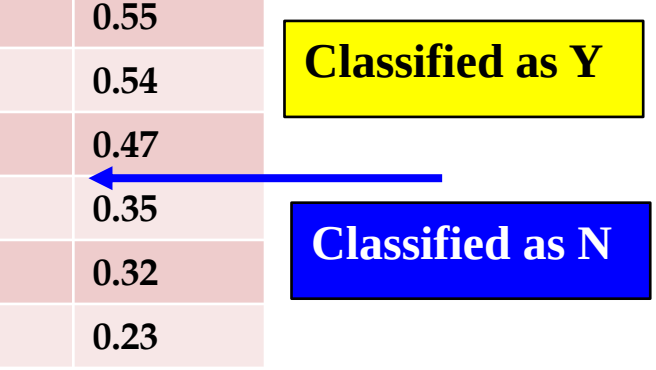
FPR = FP/FP + TN = 1/1+2 = 0.33

# How to Construct a ROC curve

3. Plot TPR and FPR for different thresholds and draw the curve

| inst | actual | predicted | | Prob. Y | Prob. N |
|---|---|---|---|---|---|
| 1 | Y | N | | 0.35 | 0.65 |
| 2 | N | N | | 0.23 | 0.77 |
| 3 | N | Y | | 0.55 | 0.45 |
| 4 | Y | N | | 0.32 | 0.68 |
| 5 | Y | Y | | 0.54 | 0.46 |
| 6 | N | N | | 0.47 | 0.53 |

| actual | Prob Y |
|---|---|
| N | 0.55 |
| Y | 0.54 |
| N | 0.47 |
| Y | 0.35 |
| Y | 0.32 |
| N | 0.23 |




- **For a cutoff 0.4:**
  - 2 N's classified as Y
  - 1 N classified as Y
  - 2 Y's classified as N
  - 1 N classified as N

# Final Notes

- Poor performance on the training data (Underfitting) could be because the model is too simple (the input features are not expressive enough) to describe the target well. To increase model flexibility, try to add new domain-specific features

- If your model is overfitting the training data, it makes sense to take actions that reduce model flexibility. To reduce model flexibility, try the following:
  - Feature selection: consider using fewer feature combinations, decrease the number of numeric attribute bins.

- Accuracy on training and test data could be poor because the learning algorithm did not have enough data to learn from. You could improve performance by doing the following:
  - Increase the amount of training data examples.
  - Increase the number of passes on the existing training data.