# EVALUATION OF MACHINE LEARNING CLASSIFIERS

## Machine Learning


**Dr. Dinesh K Vishwakarma**

Professor, Department of Information Technology

**Delhi Technological University, Delhi-110042, India**

**Email: dinesh@dtu.ac.in**

**Mobile:9971339840**

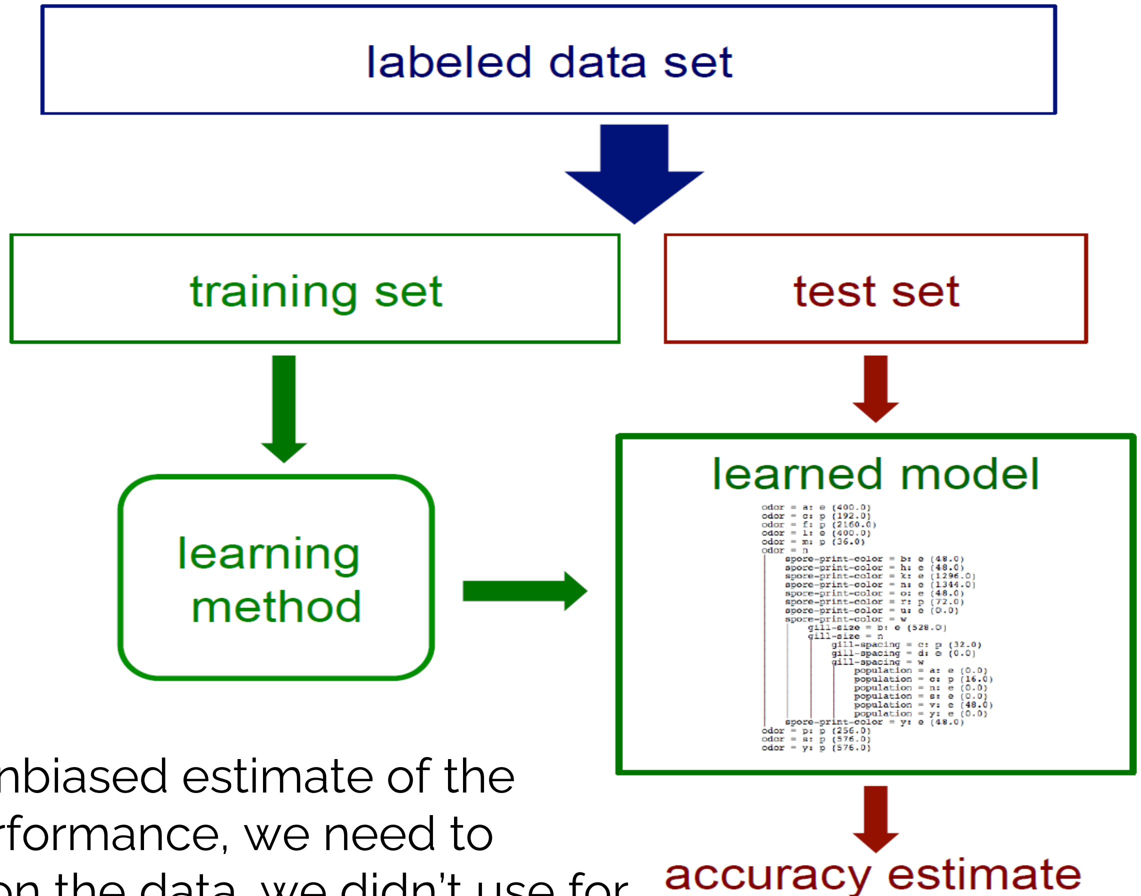
Webpage: http://www.dtu.ac.in/web/departments/informationtechnology/faculty/dkvishwakarma.php

# Outline: Evaluation Parameters

- Precision
- Recall
- Accuracy
- F-Measure
- True Positive Rate
- False Positive Rate
- Sensitivity
- ROC

# Experiment: Training and Testing

- Objective: Unbiased estimate of accuracy




Simplest way to split the data is to use the train-test split method

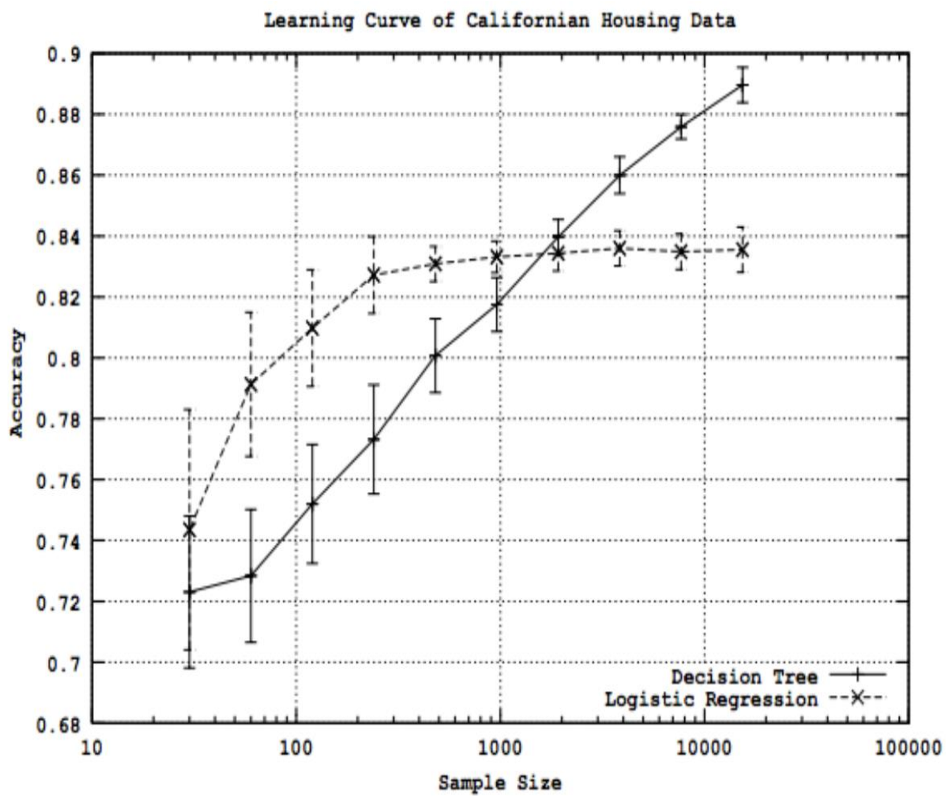
To get an unbiased estimate of the model's performance, we need to evaluate it on the data, we didn't use for training

# Experiment: Training and Testing…

- How can we get an unbiased estimate of the accuracy of a learned model?
  - ✓ when learning a model, you should pretend that you don’t have the test data yet (it is “in the mail”)*
  - ✓ if the test-set labels influence the learned model in any way, accuracy estimates will be biased
- * In some applications it is reasonable to assume that you have access to the feature vector (i.e. ***x***) but not the ***y*** part of each test instance

# Learning Curve

- How does the accuracy of a learning method change as a function of the training-set size?
  - ✓ This can be assessed by plotting *learning curves*



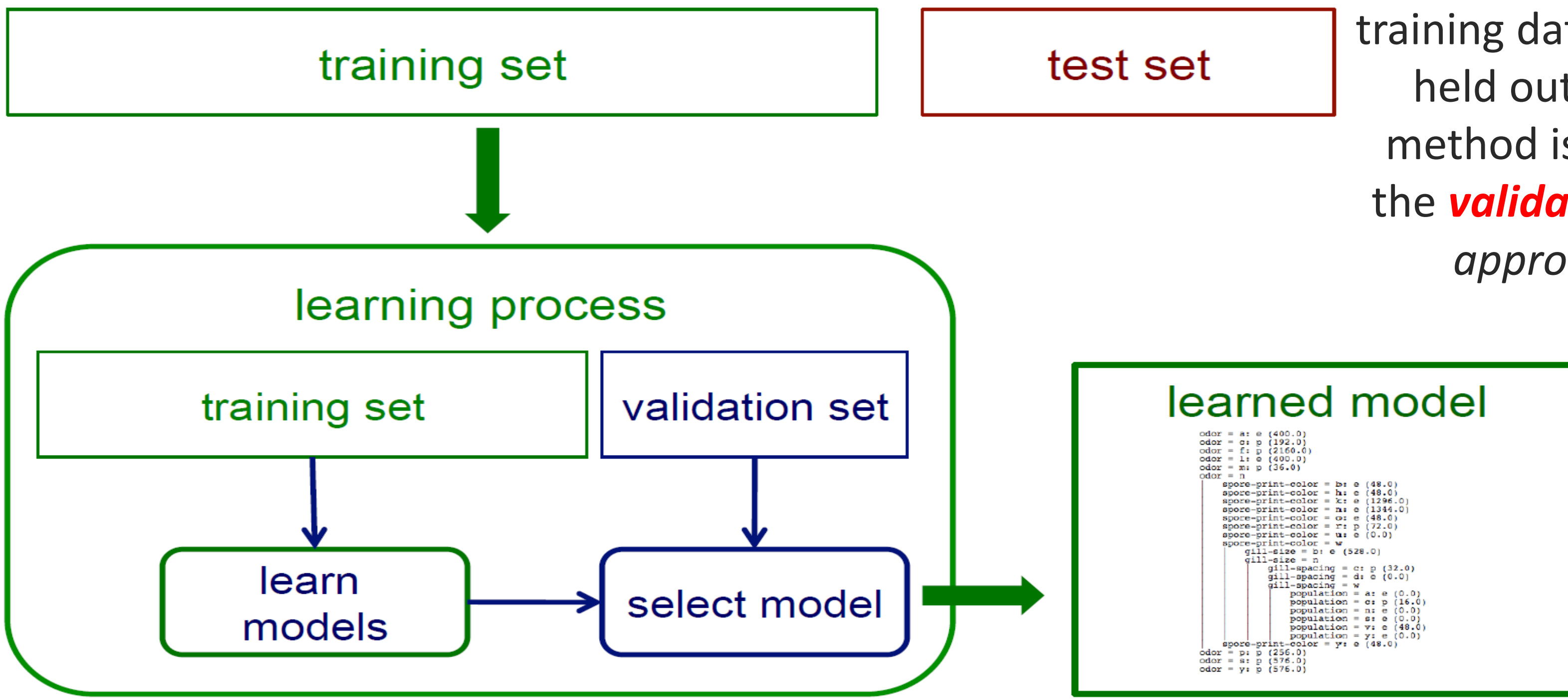

*#Given training/test set partition*

- *for each sample size s on learning curve*
- *(optionally) repeat n times*
- *randomly select s instances from training set*
  - *learn model*
- *evaluate model on test set to determine accuracy a*
- *plot (s, a) or (s, avg. accuracy and error bars)*

# Validation (Tuning) Set

- Consider we want unbiased estimates of accuracy during the learning process (e.g. to choose the best level of decision-tree pruning)?

***holding*** *out* a portion or subset o training data that i held out. This method is called the ***validation set*** *approach*



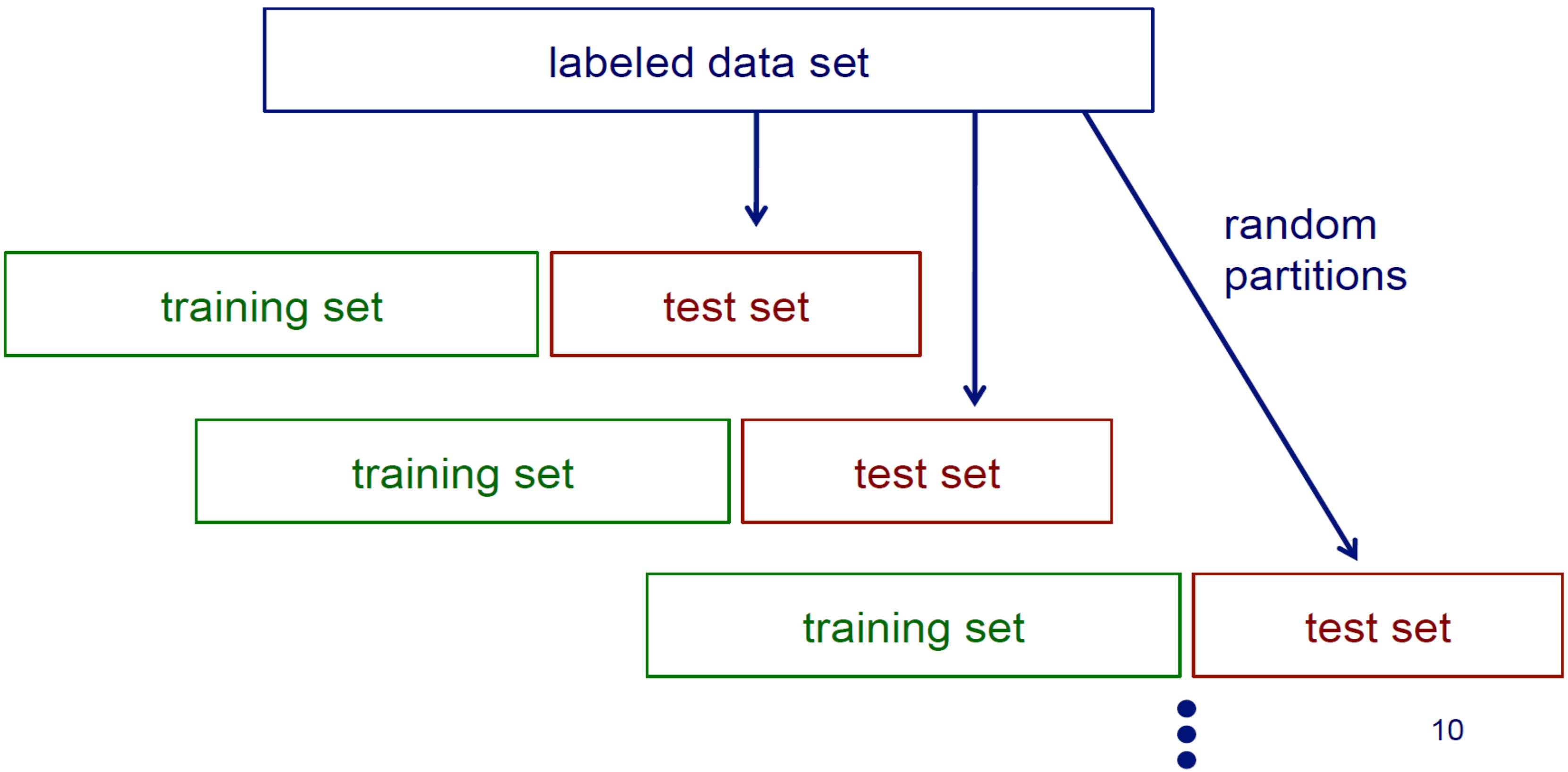

*Partition training data into separate training/validation sets*

# Limitation of Single Training/Test Partition

- We may not have enough data to make sufficiently large
  - ✓ training and test sets a larger test set gives us more reliable estimate of accuracy (i.e. a lower variance estimate)
  - ✓ but… a larger training set will be more representative of how much data we actually have for learning process
- A single training set doesn't tell us how sensitive accuracy is to a particular training sample

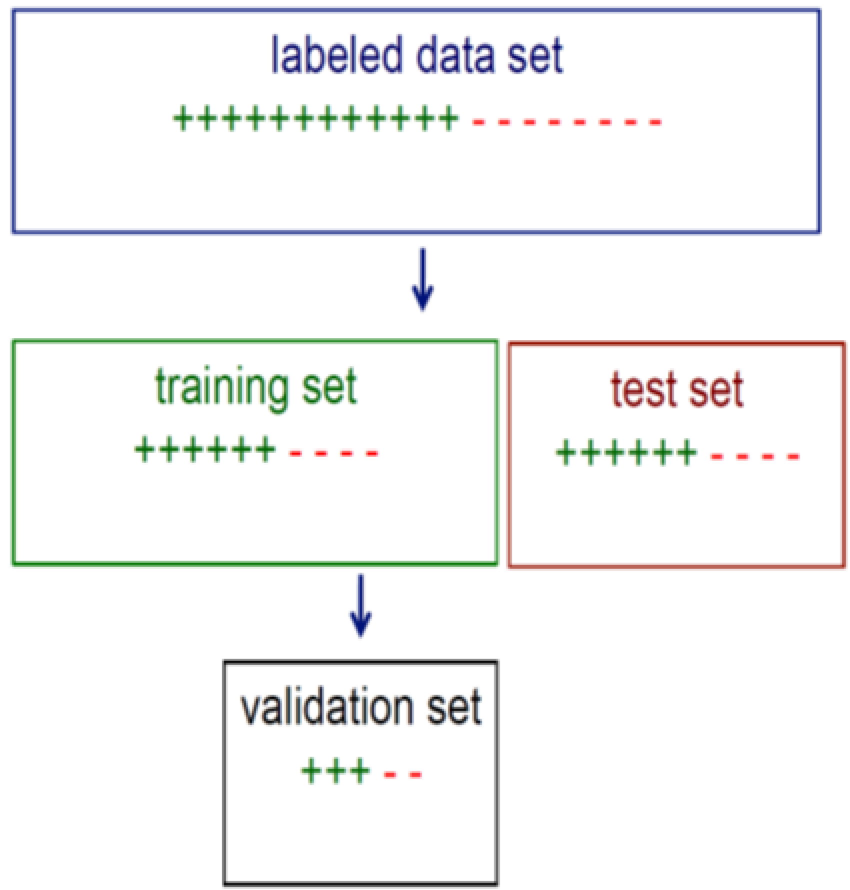
# Random Sampling

- It can be addressed the second issue by repeatedly randomly partitioning the available data into training and set sets.



# Random Sampling…

- When randomly selecting training or validation sets, we may want to ensure that class proportions are maintained in each selected set.
- This can be done via stratified sampling: first stratify (divider) instances by class, then randomly select instances from each class proportionally.



# Cross Validation

- The train and test split has limitations such as the **dataset is small, the method is prone to high variance.**
- Due to the random partition, the results can be entirely different for different test sets because in some partitions, samples that are easy to classify get into the test set, while in others, the test set receives the ‘difficult’ ones.
- **To deal with this issue, we use cross-validation to evaluate the performance of a machine learning model.**

# Cross Validation…

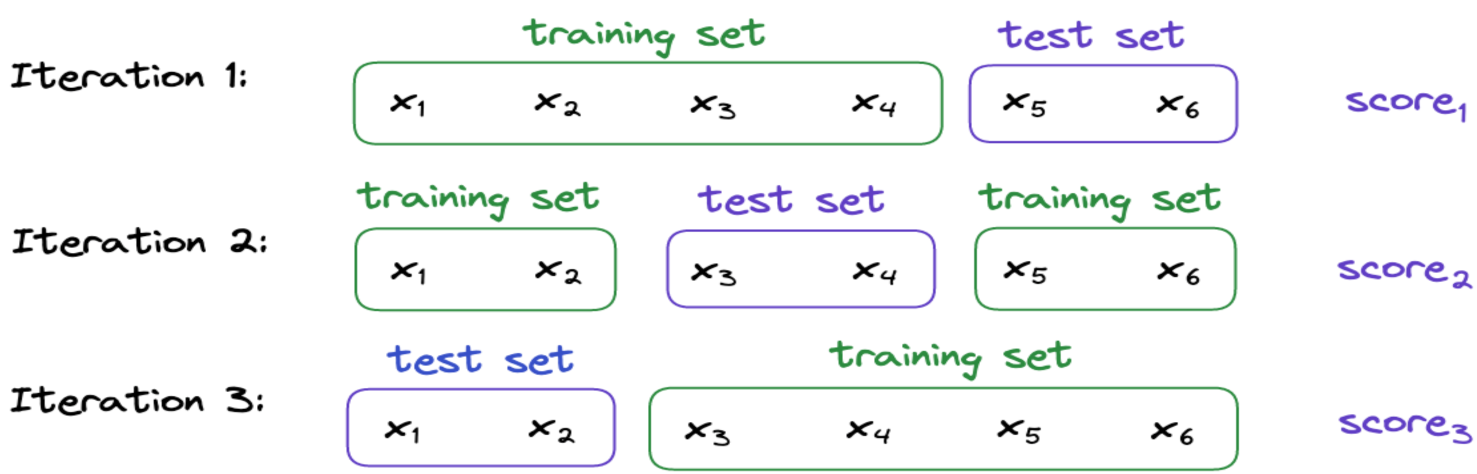
- **K-Fold Cross Validation**
  - In k-fold CV, we first divide our dataset into k equally sized subsets. Then, we repeat the train-test method **k times** such that each time one of the **k subsets** is used as a **test set** and the rest **k-1** subsets are used together as a training set.
  - Finally, we compute the estimate of the model's performance estimate by averaging the scores over the k trials.

# K-Fold Cross Validation

- **Example of 3-fold**

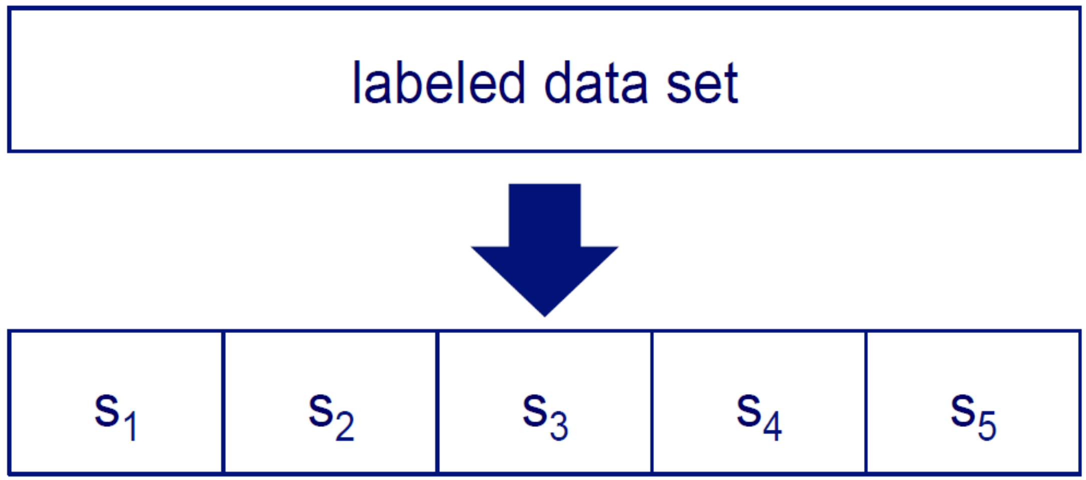
$$Overall\ Score = \frac{Score_1 + Score_2 + Score_3}{3}$$

  - For example, let's suppose that we have a dataset $S = \{x_1, x_2, x_3, x_4, x_5, x_6\}$,
  - First we divide the samples in to 3-fold as $S_1 = \{x_1, x_2), S_2 = \{x_3, x_4\}, S_3 = \{x_5, x_6\}$. Then we evaluate the model as



# 5-fold Cross Validation

*Partition data into n subsamples (S)*




*Iteratively leave one subsample out for the test set, train on the rest*

| iteration | train on | test on |
|---|---|---|
| 1 | $s_2$ $s_3$ $s_4$ $s_5$ | $s_1$ |
| 2 | $s_1$ $s_3$ $s_4$ $s_5$ | $s_2$ |
| 3 | $s_1$ $s_2$ $s_4$ $s_5$ | $s_3$ |
| 4 | $s_1$ $s_2$ $s_3$ $s_5$ | $s_4$ |
| 5 | $s_1$ $s_2$ $s_3$ $s_4$ | $s_5$ |

# 5-fold Cross Validation…

- Suppose we have 100 instances, and we want to estimate accuracy with cross validation.

| iteration | train on | test on | correct |
|---|---|---|---|
| 1 | $s_2$ $s_3$ $s_4$ $s_5$ | $s_1$ | 11 / 20 |
| 2 | $s_1$ $s_3$ $s_4$ $s_5$ | $s_2$ | 17 / 20 |
| 3 | $s_1$ $s_2$ $s_4$ $s_5$ | $s_3$ | 16 / 20 |
| 4 | $s_1$ $s_2$ $s_3$ $s_5$ | $s_4$ | 13 / 20 |
| 5 | $s_1$ $s_2$ $s_3$ $s_4$ | $s_5$ | 16 / 20 |

*10-fold cross validation is common, but* **smaller values** *of n are often used when learning takes a* **lot of time.**

accuracy = 73/100 = 73%

# Leave-One-Out Cross-Validation

- LOOCV
  - we train our machine-learning model $n$ times where $n$ is dataset size.
  - Each time, only one sample is used as a test set while the rest are used to train our model.
  - **LOO on previous example** $S = \{x_1, x_2, x_3, x_4, x_5, x_6\}$**,**
  - $S_1 = \{x_1)$
  - $S_2 = \{x_2\}$
  - $S_3 = \{x_3\}$
  - $S_4 = \{x_4\}$
  - $S_5 = \{x_5\}$
  - $S_6 = \{x_6\}$

# LOOCV…

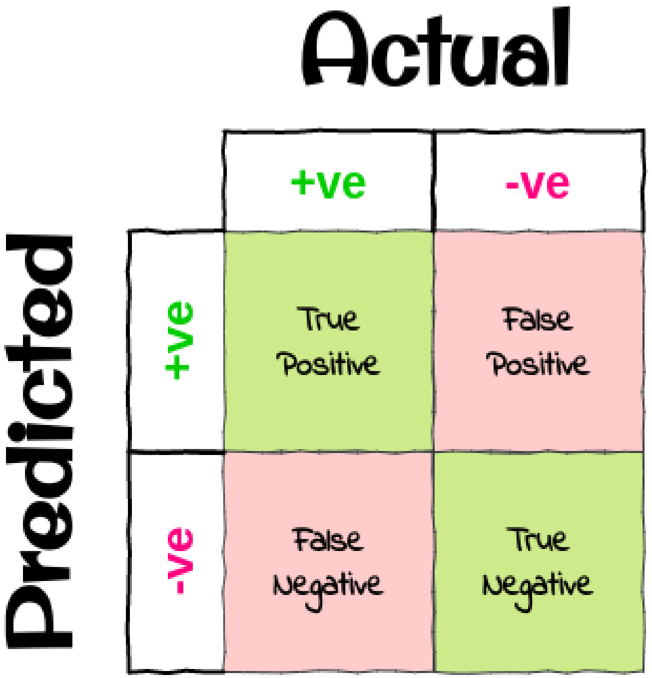

$$Overall\ Score = \frac{Score_1 + Score_2 + Score_3 + Score_4 + Score_5 + Score_6}{6}$$

# Cross Validation… Summary

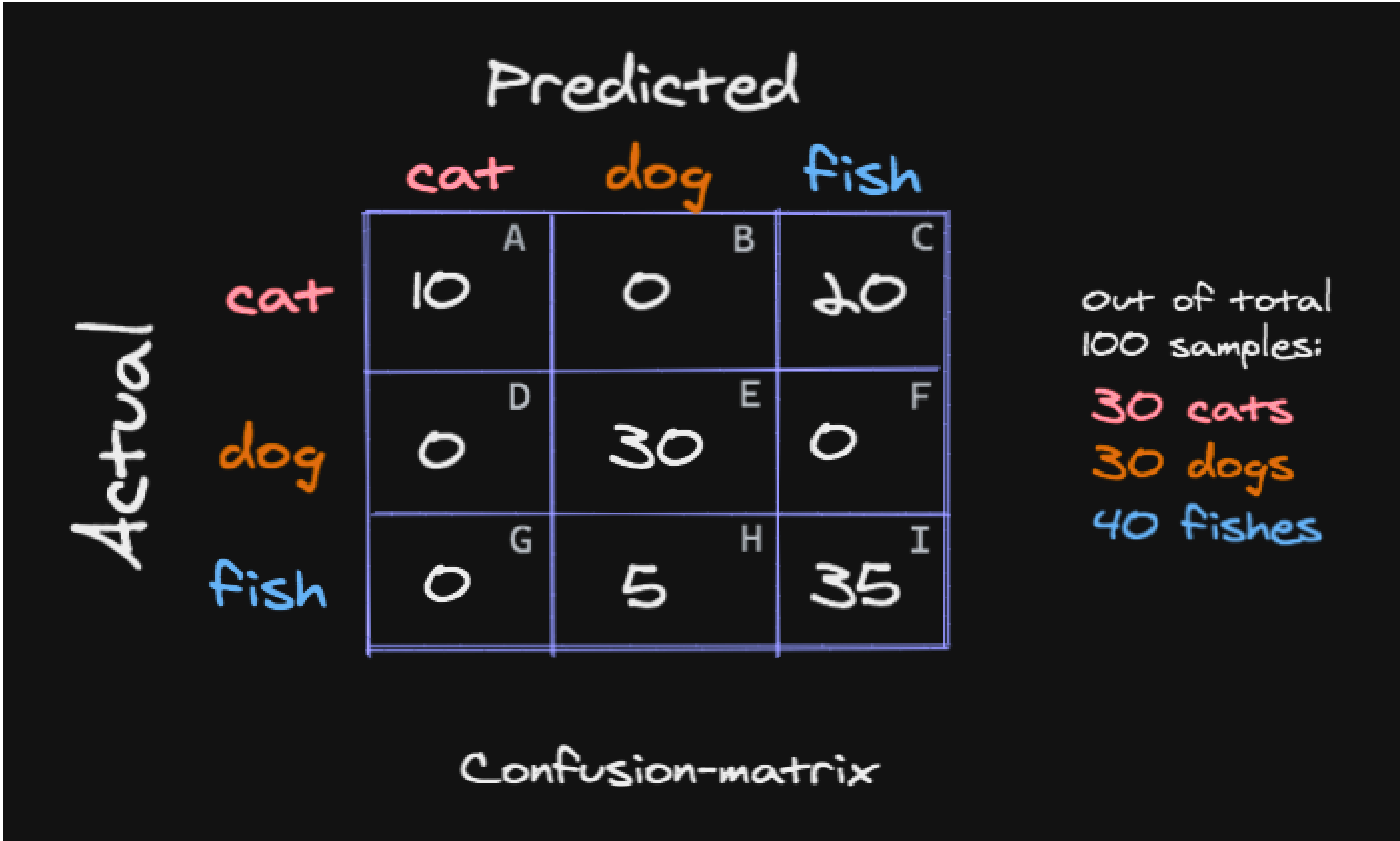
- **When the size is small, LOOCV is more appropriate** since it will use more training samples in each iteration.
- Conversely, **we use k-fold cross-validation to train a model on a large dataset, which reduces the training time.**
- Using k-Fold Cross-Validation over LOOCV is one of the examples of Bias-Variance Trade-off. It reduces the variance shown by LOOCV and introduces some bias by holding out a substantially large validation set

# Confusion Matrix

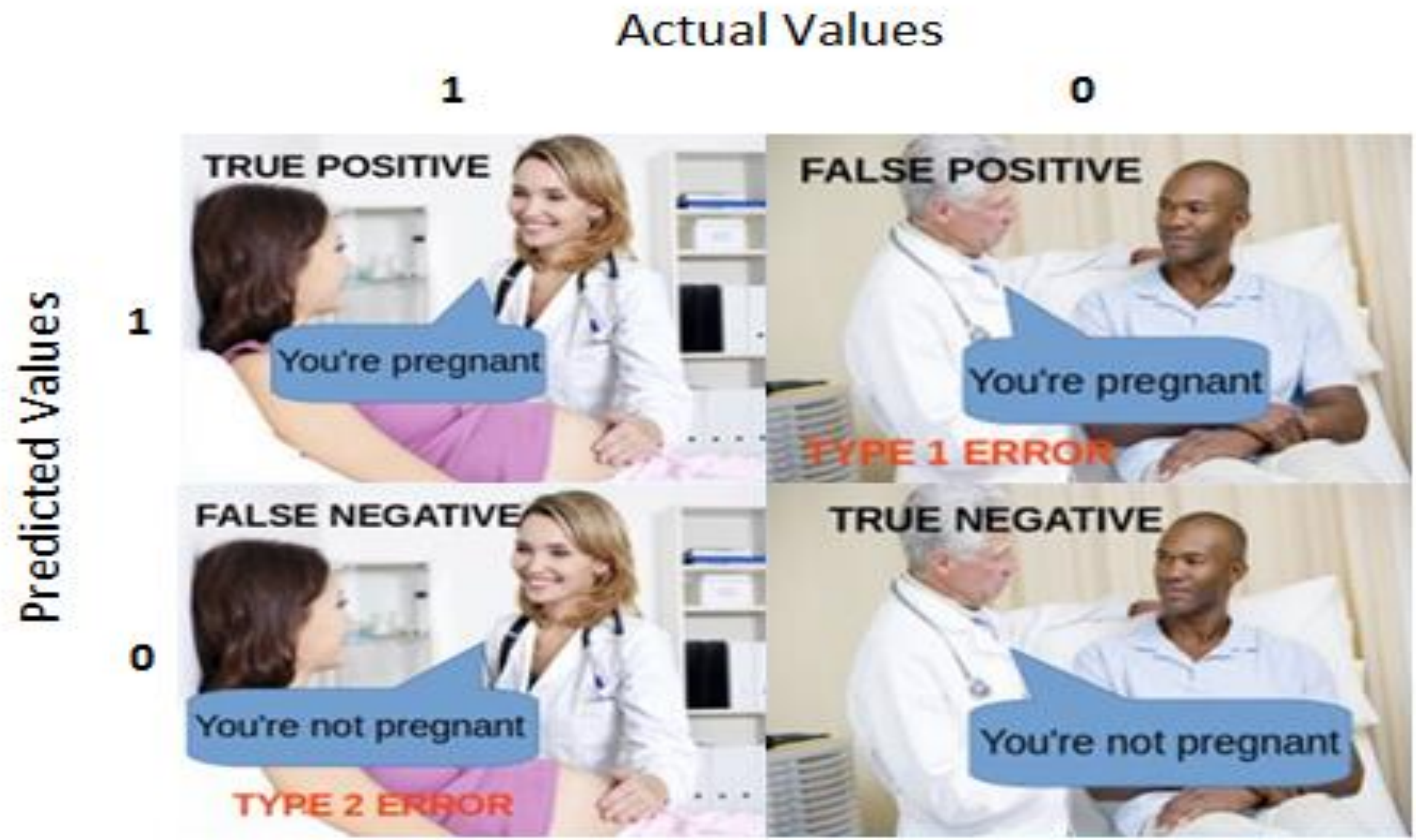
- It is also called as **prediction table**.
- It is an $N \times N$ **matrix** used for evaluating the performance of a classification model, where $N$ is the number of **target classes**
- It **compares** the actual target values with those predicted.
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable.



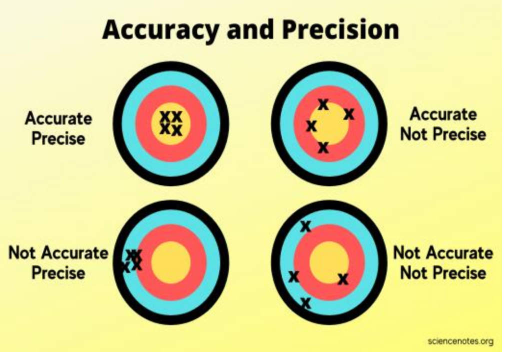
# Confusion Matrix…



# Type-I and Type-II Error



# Precision




- **Precision:** measures the correctness achieved in true prediction. Also, tells us how many predictions are actually positive out of all the total positive predicted. **Precision should be high(ideally 1)**
- “Precision is a useful metric in cases where False Positive is a higher concern than False Negatives”
- Precision/ Positive Prediction Value $P = \frac{t_p}{t_p+f_p}$
- Recall $\mathrm{R} = \frac{t_p}{t_p+f_n}$

# Issues with "Precision & Recall"

| True class → | Pos | Neg |
|---|---|---|
| Yes | 200 *TP* | 100 *FP* |
| No | 300 *FN* | 400 *TN* |
| | P=500 | N=500 |

| True class → | Pos | Neg |
|---|---|---|
| Yes | 200 | 100 |
| No | 300 | 0 |
| | P=500 | N=100 |

- Both classifiers gives the ***same*** *precision* and *recall* values of 66.7% and 40% (Note: the data sets are different)
- They exhibit very different behaviours:
  - ✓ Same **positive** recognition rate
  - ✓ Extremely different ***negative*** recognition rate: strong on the left / nil on the right
- Note: Accuracy has no problem catching this!

# A combined measure: *F*

- Combined measure that assesses **precision/recall** tradeoff is **F measure** (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- **high F1 score** if both Precision and Recall are high
- **low F1 score** if both Precision and Recall are low
- **medium F1 score** if one of Precision and Recall is low and the other is high

- People usually use balanced $F_1$ measure
  - i.e., with $\beta = 1$ or $\alpha = ½$
- Harmonic mean is a conservative average.

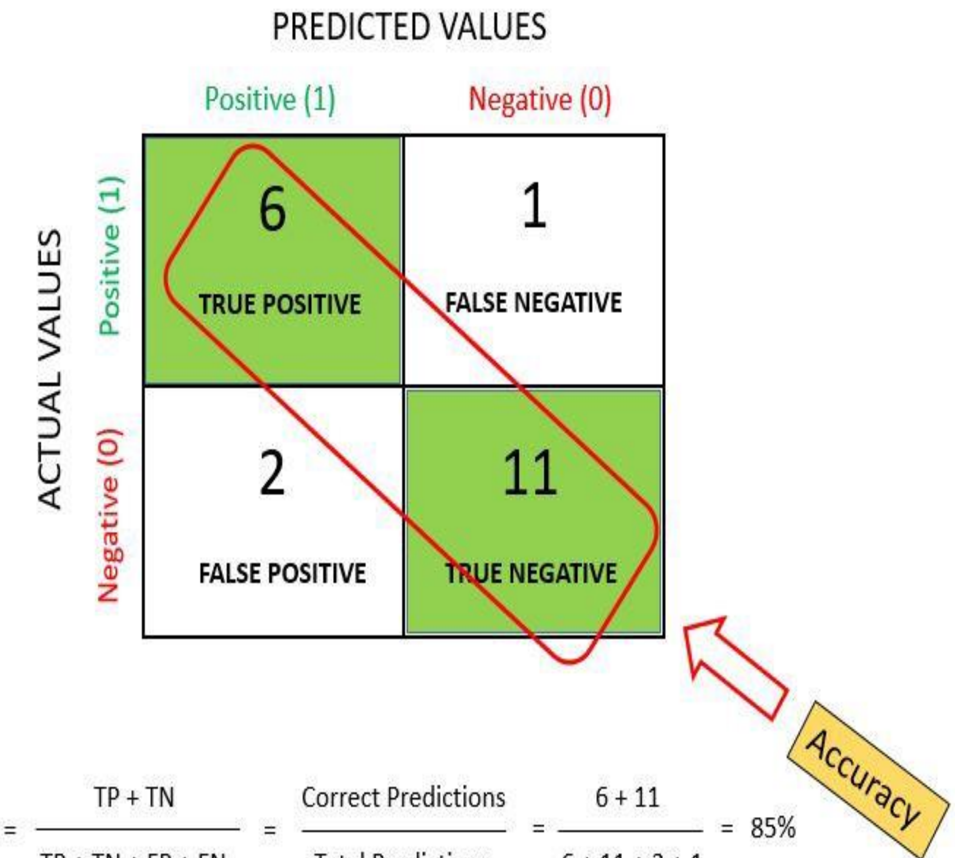
# Accuracy

- **Measures the correct predictions.**
- The **accuracy metric** is ***not suited*** for **imbalanced classes**.
- **Accuracy** has its own *disadvantages*, for **imbalanced data**, when the model predicts that each point belongs to the majority class label, the accuracy will be high. But, the model is not accurate.
- **Accuracy** is a valid choice of ***evaluation for classification problems*** which are ***well balanced*** and ***not skewed or there is no class imbalance.***

# Accuracy Measure

- The **accuracy** of an engine: the fraction of these classifications that are correct.

- $\boldsymbol{Accuracy(\%) = \frac{(t_p + t_n)}{(t_p + t_n + f_n + f_p)} \times 100}$




$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{6 + 11}{6 + 11 + 2 + 1} = 85\%$$

# Accuracy Measure

| $y$ labelled Value (0-Negative, 1-Positive) | $\hat{y}$ predicted value | Output at threshold (0.5) |
|---|---|---|
| 0 | 0.3 | 0 |
| 1 | 0.4 | 0 |
| 0 | 0.7 | 1 |
| 1 | 0.8 | 1 |
| 0 | 0.4 | 0 |
| 1 | 0.7 | 1 |

*Confusion Matrix*

| | |
|---|---|
| **TP=2** | **FP=1** |
| FN=1 | TN=2 |

$$Accuracy = \frac{4}{6} = .666$$

$$Recall = \frac{TP}{TP + FN} = \frac{2}{3} = .666$$

$$Precision = \frac{2}{3} = .666$$

# Issues with 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*

# Issues with Accuracy…

| True class → | Pos | Neg |
|---|---|---|
| Yes | 200 | 100 |
| No | 300 | 400 |
| | P=500 | N=500 |

| True class → | Pos | Neg |
|---|---|---|
| Yes | 400 | 300 |
| No | 100 | 200 |
| | P=500 | N=500 |

- Both classifiers gives 60% accuracy.
- They exhibit very different behaviors:
  - ✓ **On the left:** weak positive recognition rate/strong negative recognition rate
  - ✓ **On the right:** strong positive recognition rate/weak negative recognition rate

# Is accuracy adequate measure?

- **Accuracy may not be useful measure in cases where**
  - there is a large class skew
    - ✓ Is 98% accuracy good if 97% of the instances are negative?
  - there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong.
    - ✓ Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease
  - we are most interested in a subset of high-confidence predictions

# Miss Classification Error

- Recognition rate=accuracy=success rate
- Miss classification rate= failure rate

| n=165 | Predicted: NO | Predicted: YES |
|---|---|---|
| Actual: NO | 50 | 10 |
| Actual: YES | 5 | 100 |

- $Miss\ Classification\ Error = \frac{5+10}{50+10+5+100} = 0.09$
- Error in percentage=$\frac{FN+FP}{TP+FP+TN+FN} * 100$

# Sensitivity & Specificity

- **Sensitivity** is the metric that evaluates a model's ability to predict true positives of each available category.
- **Specificity** is the metric that evaluates a model's ability to predict true negatives of each available category.

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

# Find Sensitivity and Specificity

**Table 2.2** A confusion matrix of a model

| | **Predicted +1** | **Predicted –1** |
|---|---|---|
| Actual +1 | 95 | 7 |
| Actual –1 | 4 | 94 |

# Other form of Accuracy Metrics

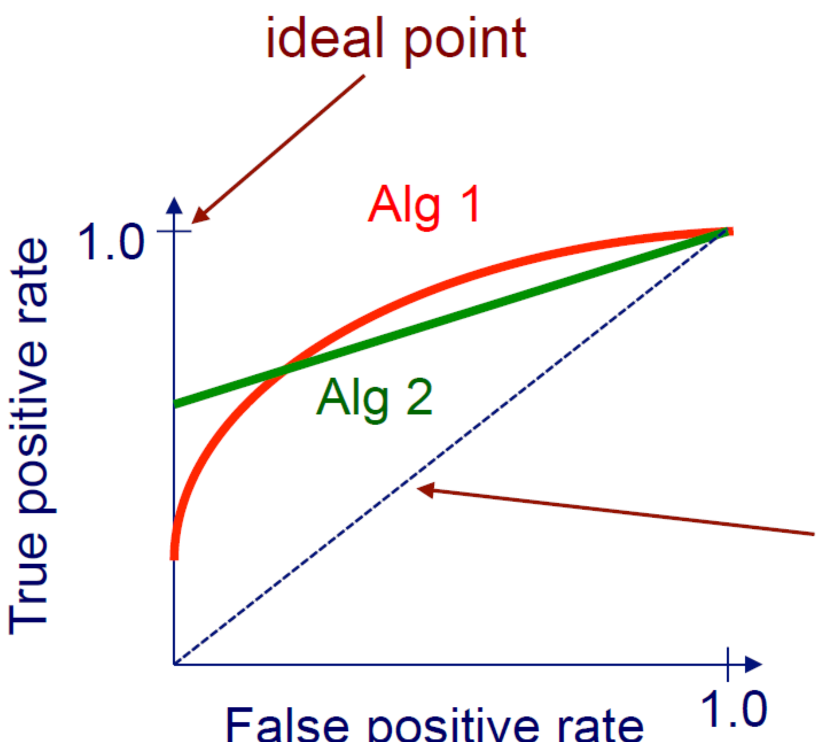
| | | actual class | |
|---|---|---|---|
| | | positive | negative |
| predicted class | positive | true positives (TP) | false positives (FP) |
| | negative | false negatives (FN) | true negatives (TN) |

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

# ROC/AUC

- A Receiver Operating Characteristic (ROC)/Area Under Curve plots the TP-rate vs. the FP-rate as a threshold on the confidence of an instance being positive is varied.

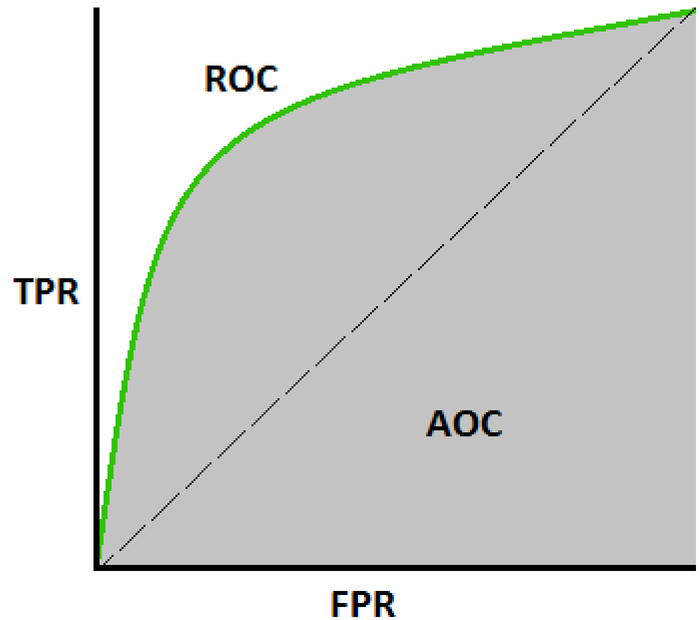



Different methods can work better in different parts of ROC space. This depends on cost of false + vs. false -

*expected curve for random guessing*

# Area Under the Receiver Operating Characteristics

- AUC-ROC curve measure the performance at various **threshold settings**.
- ROC is a probability curve and AUC represents the degree or **measure of separability**.
- AUC tells the model **capability of distinguishing between classes**.
- **Higher AUC**, the better the model is at predicting 0 classes as 0 and 1 classes as 1.
- The ROC curve is plotted between TPR & FPR, where TPR is on the y-axis and FPR is on the x-axis.

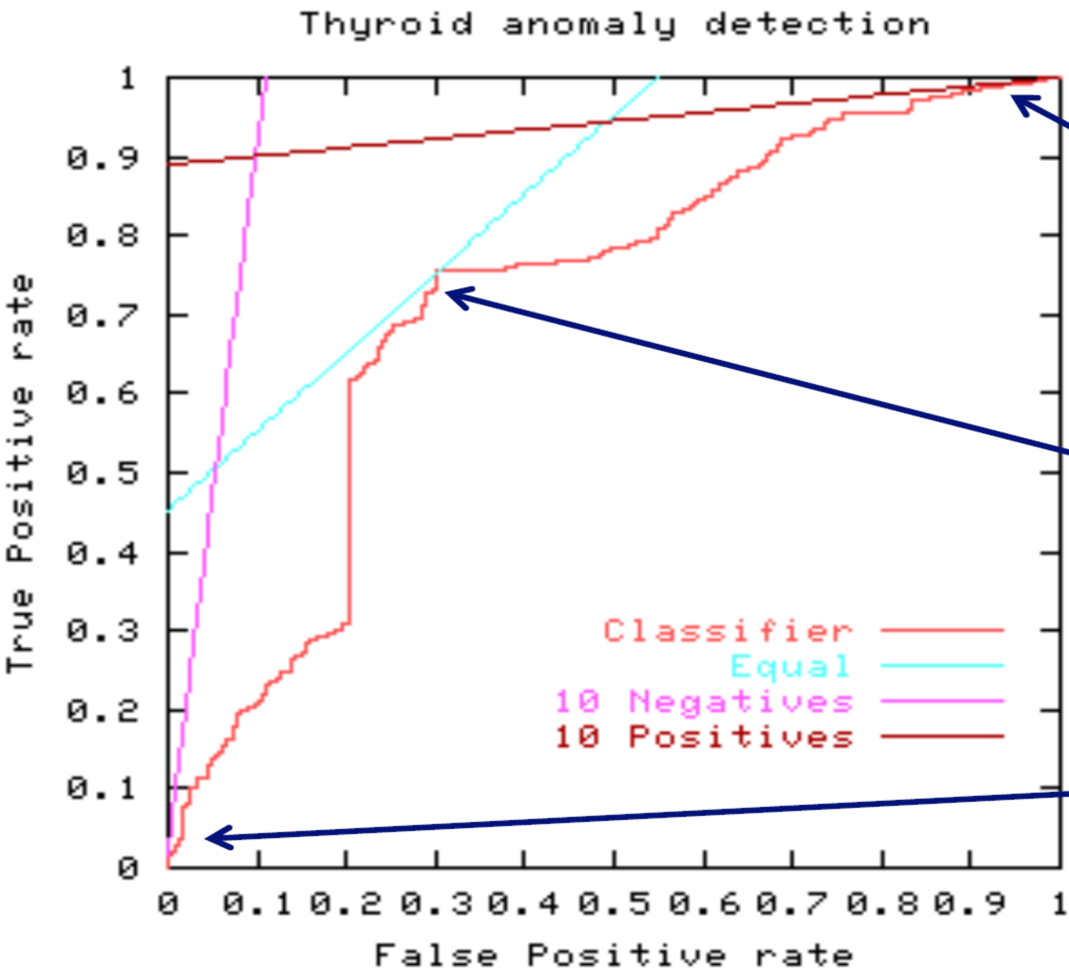



$$\text{TPR / Recall / Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{FPR} = 1 - \text{Specificity}$$

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

# ROC curves & Misclassification costs




*Best operating point when **FN** costs 10× **FP***

*Best operating point when cost of misclassifying positives and negatives is equal*

*Best operating point when FP costs 10× FN*

# Create ROC of a model

- Consider a prediction table at different threshold setting

| $y$ labelled Value (0-Negative, 1-Positive) | $\hat{y}$ predicted value | Output at threshold (0.5) | Output at threshold (0.6) | Output at threshold (0.72) | Output at threshold (0.8) |
|---|---|---|---|---|---|
| 0 | 0.3 | 0 | 0 | 0 | 0 |
| 1 | 0.55 | 1 | 0 | 0 | 0 |
| 0 | 0.75 | 1 | 1 | 1 | 0 |
| 1 | 0.8 | 1 | 1 | 1 | 1 |
| 0 | 0.4 | 0 | 0 | 0 | 0 |
| 1 | 0.7 | 1 | 1 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Threshold Setting (0.5)** | **TP=3** | **FP=1** | **TN=2** | **FN=0** | **TPR=3/(3+0)=1** | **FPR=1/(2+1)=.33** |

# Create ROC of a model…

- Threshold setting (0.6)

| $\boldsymbol{y}$ labelled Value (0- Negative, 1- Positive) | $\hat{y}$ predicted value | Output at threshold (0.5) | Output at threshold (0.6) | Output at threshold (0.72) | Output at threshold (0.8) |
|---|---|---|---|---|---|
| 0 | 0.3 | 0 | 0 | 0 | 0 |
| 1 | 0.55 | 1 | 0 | 0 | 0 |
| 0 | 0.75 | 1 | 1 | 1 | 0 |
| 1 | 0.8 | 1 | 1 | 1 | 1 |
| 0 | 0.4 | 0 | 0 | 0 | 0 |
| 1 | 0.7 | 1 | 1 | 0 | 0 |

| **Threshold Setting (0.6)** | **TP=2** | **FP=1** | **TN=2** | **FN=1** | **TPR=2/(2+1)=.66** | **FPR=1/(1+2)=.33** |
|---|---|---|---|---|---|---|

# Create ROC of a model…

- Threshold setting (0.72)

| $\boldsymbol{y}$ labelled Value (0- Negative, 1- Positive) | $\hat{y}$ predicted value | Output at threshold (0.5) | Output at threshold (0.6) | Output at threshold (0.72) | Output at threshold (0.8) |
|---|---|---|---|---|---|
| 0 | 0.3 | 0 | 0 | 0 | 0 |
| 1 | 0.55 | 1 | 0 | 0 | 0 |
| 0 | 0.75 | 1 | 1 | 1 | 0 |
| 1 | 0.8 | 1 | 1 | 1 | 1 |
| 0 | 0.4 | 0 | 0 | 0 | 0 |
| 1 | 0.7 | 1 | 1 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Threshold Setting (0.72)** | **TP=1** | **FP=1** | **TN=2** | **FN=2** | **TPR=1/(1+2)=.33** | **FPR=1/(1+2)=.33** |

# Create ROC of a model…

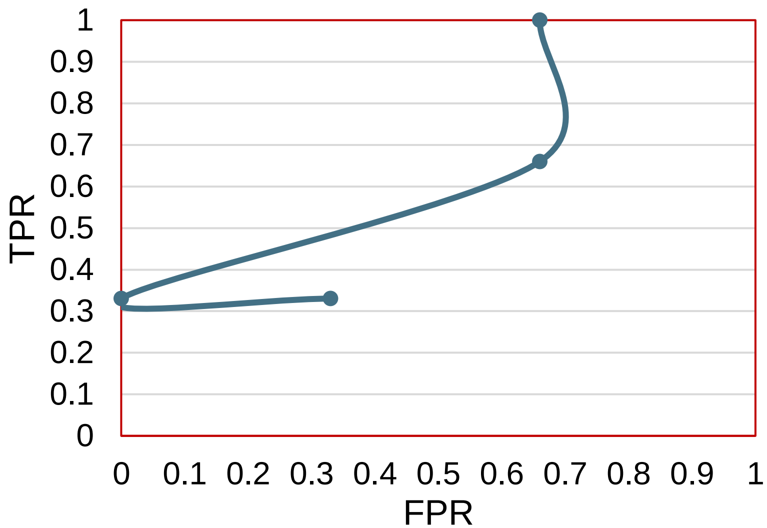
- Threshold setting (0.80)

| $y$ labelled Value (0-Negative, 1-Positive) | $\hat{y}$ predicted value | Output at threshold (0.5) | Output at threshold (0.6) | Output at threshold (0.72) | Output at threshold (0.8) |
|---|---|---|---|---|---|
| 0 | 0.3 | 0 | 0 | 0 | 0 |
| 1 | 0.55 | 1 | 0 | 0 | 0 |
| 0 | 0.75 | 1 | 1 | 1 | 0 |
| 1 | 0.8 | 1 | 1 | 1 | 1 |
| 0 | 0.4 | 0 | 0 | 0 | 0 |
| 1 | 0.7 | 1 | 1 | 0 | 0 |

| Threshold Setting (0.80 | TP=1 | FP=0 | TN=3 | FN=2 | TPR=1/(1+2)=.33 | FPR=0 |
|---|---|---|---|---|---|---|

# Plot of ROC

| | | | | | | |
|---|---|---|---|---|---|---|
| **Threshold Setting (0.5)** | **TP=3** | **FP=1** | **TN=2** | **FN=0** | **TPR=3/(3+0)=1** | **FPR=1/(2+1)=.33** |
| **Threshold Setting (0.6)** | **TP=2** | **FP=1** | **TN=2** | **FN=1** | **TPR=2/(2+1)=.66** | **FPR=1/(1+2)=.33** |
| **Threshold Setting (0.72)** | **TP=1** | **FP=1** | **TN=2** | **FN=2** | **TPR=1/(1+2)=.33** | **FPR=1/(1+2)=.33** |
| **Threshold Setting (0.80** | **TP=1** | **FP=0** | **TN=3** | **FN=2** | **TPR=1/(1+2)=.33** | **FPR=0** |

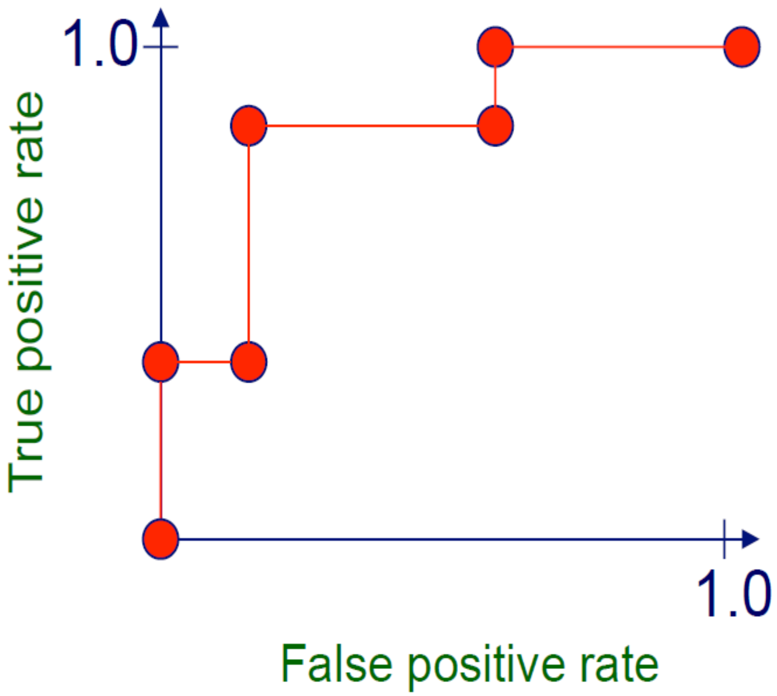


# Step to create ROC

- Sort test-set predictions according to confidence that each instance is positive.
- Step through sorted list from high to low confidence
  - ✓ locate a *threshold* between instances with opposite classes (keeping instances with the same confidence value on the same side of threshold)
  - ✓ compute TPR, FPR for instances above threshold
  - ✓ output (FPR, TPR) coordinate

# Example of ROC Plot

| instance | confidence positive | | correct class |
|---|---|---|---|
| Ex 9 | .99 | | + |
| Ex 7 | .98 | TPR= 2/5, FPR= 0/5 | + |
| Ex 1 | .72 | TPR= 2/5, FPR= 1/5 | - |
| Ex 2 | .70 | | + |
| Ex 6 | .65 | TPR= 4/5, FPR= 1/5 | + |
| Ex 10 | .51 | | - |
| Ex 3 | .39 | TPR= 4/5, FPR= 3/5 | - |
| Ex 5 | .24 | TPR= 5/5, FPR= 3/5 | + |
| Ex 4 | .11 | | - |
| Ex 8 | .01 | TPR= 5/5, FPR= 5/5 | - |



# Example of ROC Plot …

- **Rearrange the samples according to class**

| Correct class | Instance | Confidence Positive |
|---|---|---|
| + | Ex 9 | 0.99 |
| + | Ex 7 | 0.98 |
| + | Ex 2 | 0.70 |
| + | Ex 6 | 0.65 |
| + | Ex 5 | 0.24 |
| - | Ex 1 | 0.72 |
| - | Ex10 | 0.51 |
| - | Ex 3 | 0.39 |
| - | Ex 4 | 0.11 |
| - | Ex 8 | 0.01 |

Positive Class

Negative Class

# Example of ROC Plot …

- **For Threshold 0.72**

| Correct class | Instance | confidence positive | predicted class |
|---|---|---|---|
| + | Ex 9 | 0.99 | + |
| + | Ex 7 | 0.98 | + |
| + | Ex 2 | 0.70 | - |
| + | Ex 6 | 0.65 | - |
| + | Ex 5 | 0.24 | - |
| - | Ex 1 | 0.72 | + |
| - | Ex10 | 0.51 | - |
| - | Ex 3 | 0.39 | - |
| - | Ex 4 | 0.11 | - |
| - | Ex 8 | 0.01 | - |

**Confidence > threshold**
**Positive class**
**Else**
**Negative class**

**TP=2**
**FP=1**
**TN=4**
**FN=3**
**TPR=TP/TP+FN=2/5**
**FPR=FP/FP+TN=1/5**

# Example of ROC Plot …

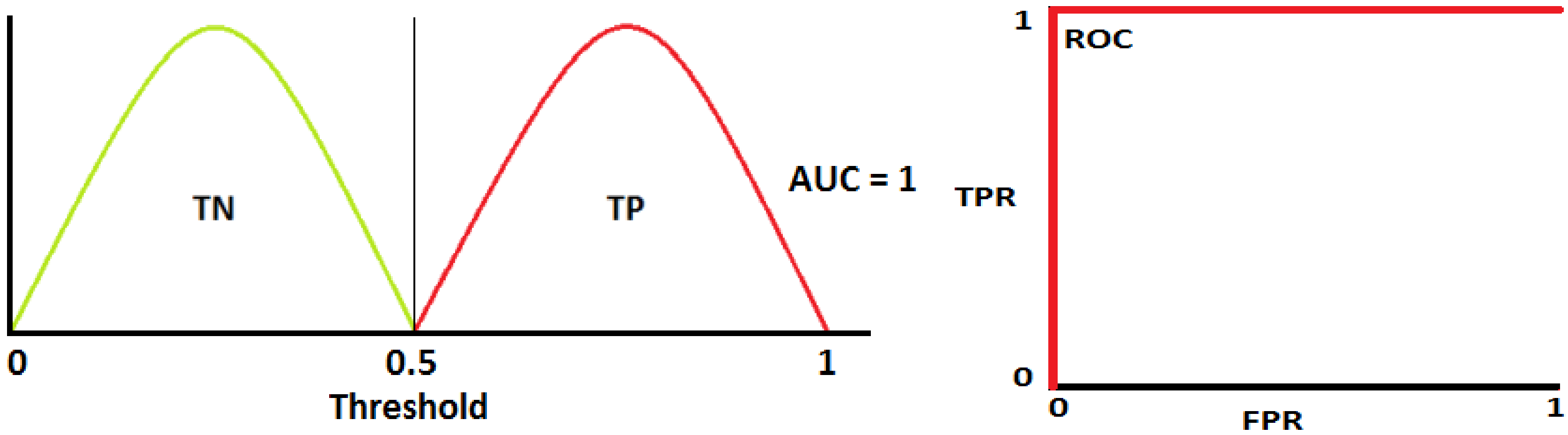
- **For Threshold 0.65**

| Correct class | Instance | confidence positive | predicted class |
|---|---|---|---|
| + | Ex 9 | 0.99 | + |
| + | Ex 7 | 0.98 | + |
| + | Ex 2 | 0.70 | + |
| + | Ex 6 | 0.65 | + |
| + | Ex 5 | 0.24 | - |
| - | Ex 1 | 0.72 | + |
| - | Ex10 | 0.51 | - |
| - | Ex 3 | 0.39 | - |
| - | Ex 4 | 0.11 | - |
| - | Ex 8 | 0.01 | - |

**Confidence > threshold**
**Positive class**
**Else**
**Negative class**

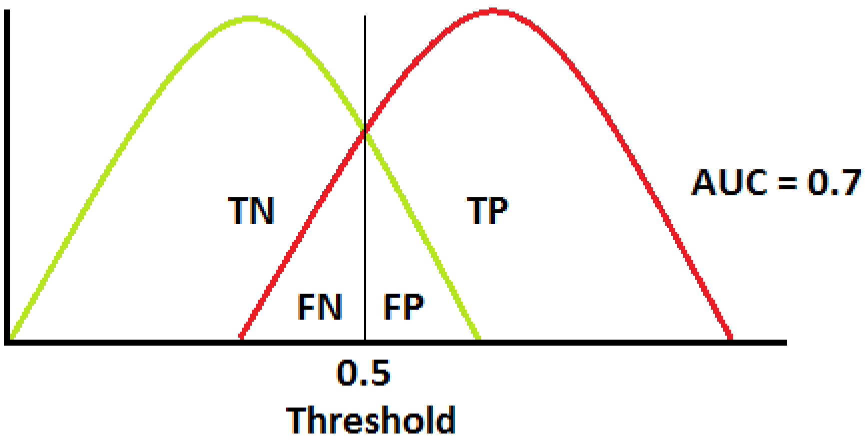
**TP=4**
**FP=1**
**TN=4**
**FN=1**
**TPR=TP/TP+FN=4/5**
**FPR=FP/FP+TN=1/5**

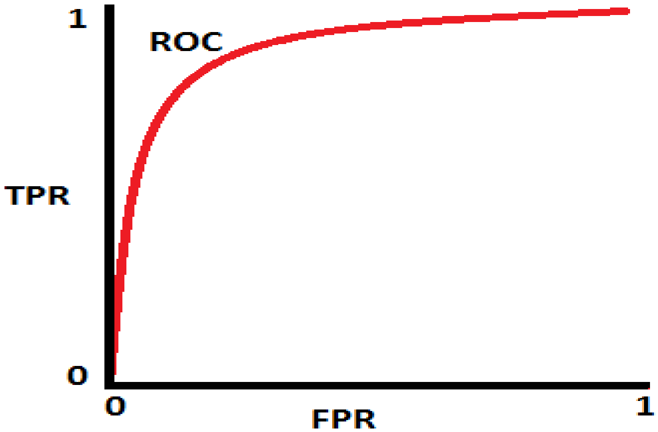
# Significance of ROC




- This is an ideal situation, when two curves don't overlap at all, means model has an ideal measure of separability.
- It is perfectly able to distinguish between positive class and negative class.

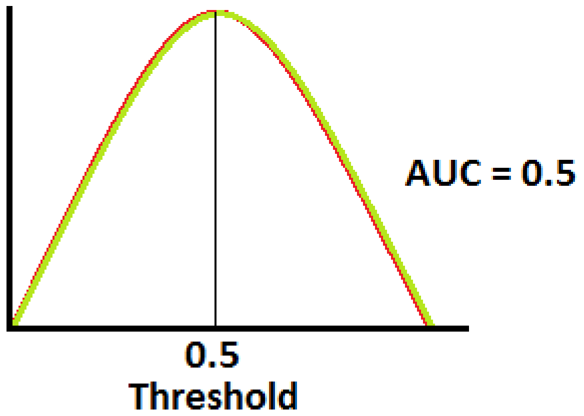
# Significance of ROC…

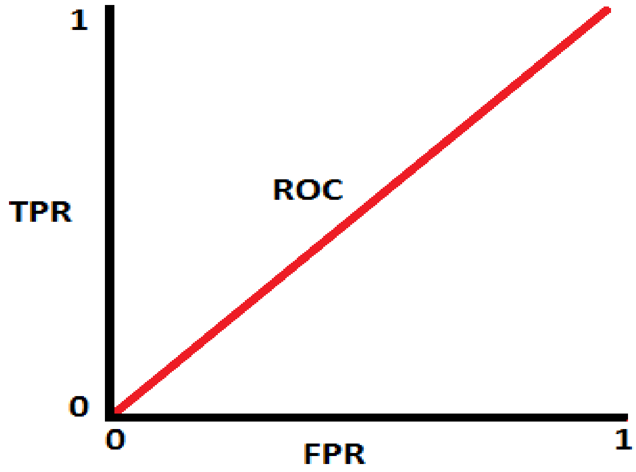



- When two distributions overlap, then type 1 and type 2 errors are introduced.
- Depending upon the threshold, it can be minimized or maximized. When AUC is 0.7, it means there is a 70% chance that the model will be able to distinguish between positive class and negative class.

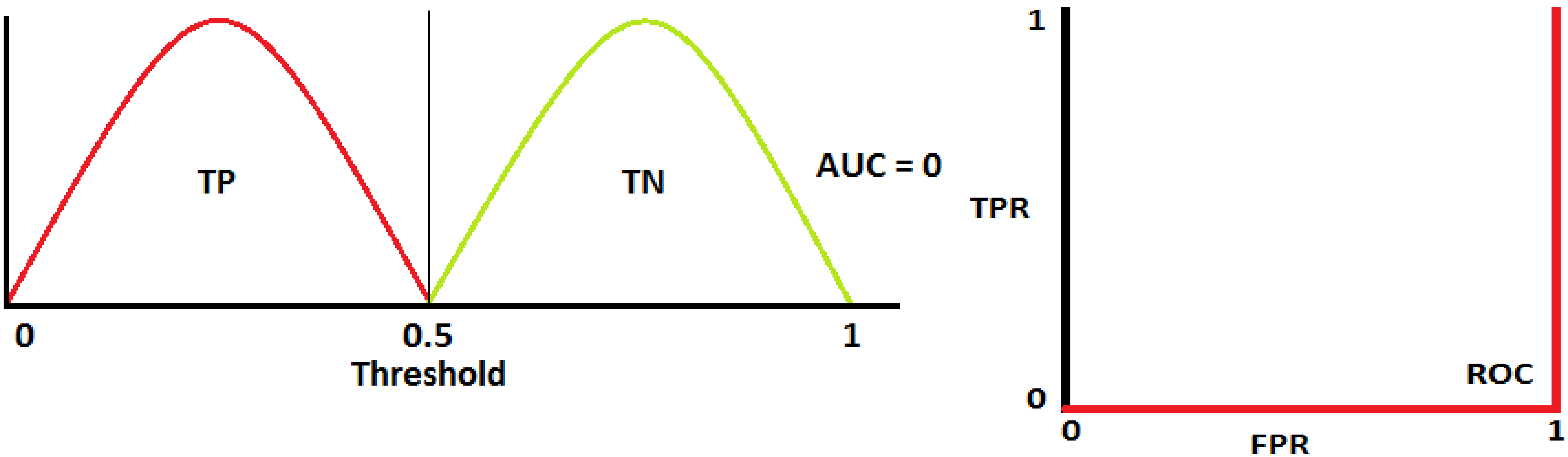
# Significance of ROC…




- This is the worst situation.
- When AUC is approximately 0.5, the model has no discrimination capacity to distinguish between positive class and negative class.

# Significance of ROC…




- When AUC is approximately 0, the model is actually reciprocating the classes. It means the model is predicting a negative class as a positive class and vice versa.

**TPR ⇧, FPR ⇧ and TPR ⇩, FPR ⇩**

# Issues with ROC/AUC

- AUC/ROC has adopted as replacement of accuracy but it has also some criticism such as:
  - The ROC curves on which the AUCs of different classifiers are based may cross, thus not giving an accurate picture of what is really happening.
  - The misclassification cost distributions used by the AUC are different for different classifiers.
  - Therefore, we may be comparing “apples and oranges” as the AUC may give more weight to misclassifying a point by classifier A than it does by classifier B. Ans: H-Measure

# Other Accuracy Metrics

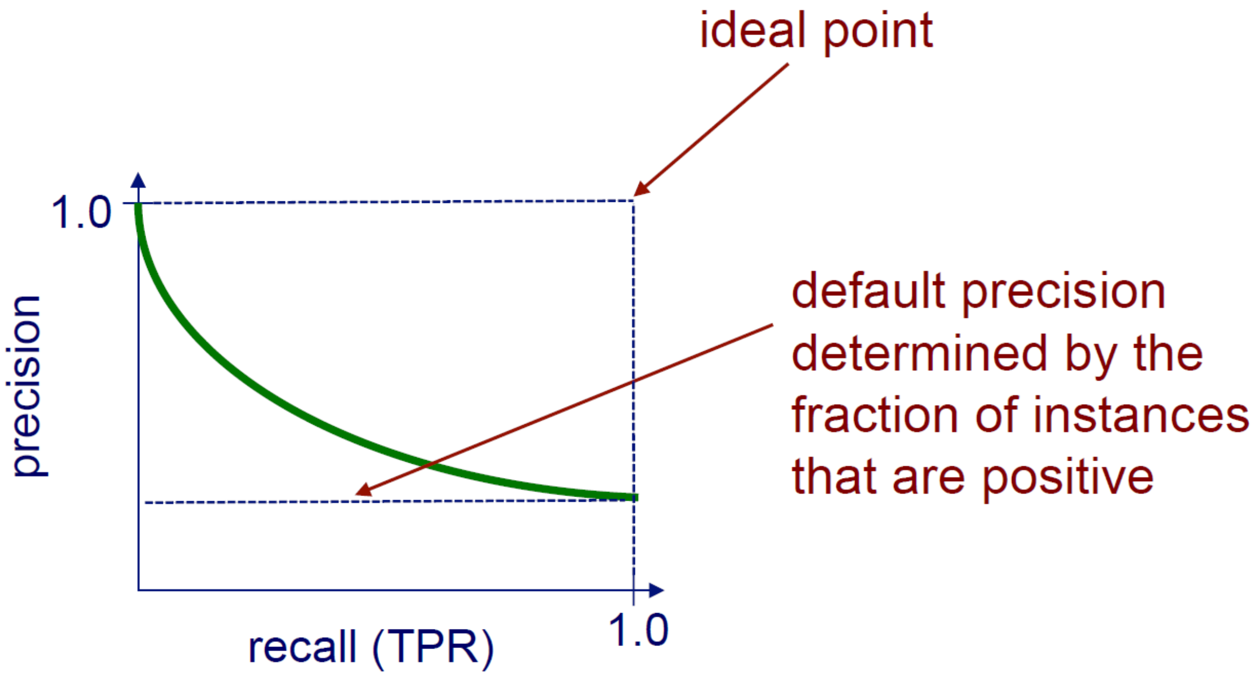
| | | actual class: positive | actual class: negative |
|---|---|---|---|
| predicted class | positive | true positives (TP) | false positives (FP) |
| | negative | false negatives (FN) | true negatives (TN) |

$$\text{recall (TP rate)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{precision} = \frac{\text{TP}}{\text{predicted pos}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

# Precision/recall curves

- A *precision/recall curve* plots the precision vs. recall (TP-rate) as a threshold on the confidence of an instance being positive is varied.



# Comment on ROC/PR Curve

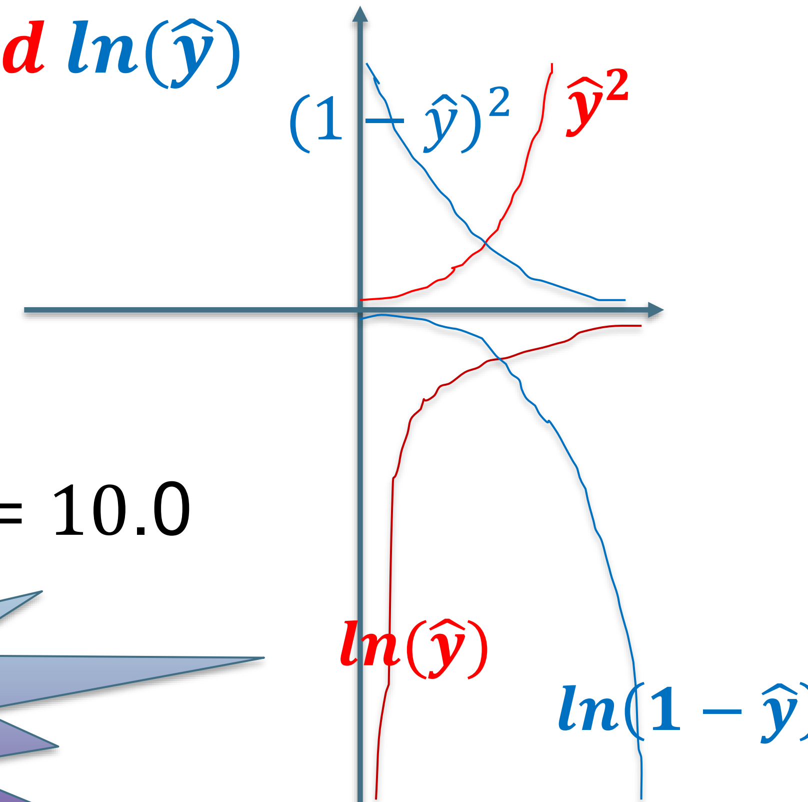
- Both
  - ✓ allow predictive performance to be assessed at various levels of confidence
  - ✓ assume binary classification tasks
  - ✓ sometimes summarized by calculating *area under the curve*
- ROC curves
  - ✓ insensitive to changes in class distribution (ROC curve does not change if the proportion of positive and negative instances in the test set are varied)
  - ✓ can identify optimal classification thresholds for tasks with differential misclassification costs
- Precision/Recall curves
  - ✓ show the fraction of predictions that are false positives
  - ✓ well suited for tasks with lots of negative instances

# Loss Function

- Mean Square Error Loss Function
  - ✓ It is used for regression problem
  - ✓ Mean square error loss for m-data point is defined as $L_{SE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \widehat{y_i})^2$
  - ✓ For single point $L_{SE}1 = (y - \hat{y})^2$.
- Binary Cross Entropy Loss Function
  - ✓ It is used for classification problem
  - ✓ BCELF is defined $L_{CE} = -\frac{1}{m}\sum_{i=1}^{m}[y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i)]$
  - ✓ For single point $L_{CE}1 = y\ln(\hat{y}) + (1 - y)\ln(1 - \hat{y})$ , negative sign ignored, because we interested more in shape.

# Example

- Consider a 2-class problem, where ground truth is $y = 0. then\ \boldsymbol{L_{SE}1 = \hat{y}^2}$ and, $y = 1\ L_{SE}1 = (1 - \hat{y})^2$.
- Similarly $\boldsymbol{L_{CE}1 = ln(1 - \hat{y}) and\ ln(\hat{y})}$
- Consider example,
- $y = 0, \&\ \hat{y} = 0.9, L_{SE} = 0.81$
- Similarly $L_{CE} = 2.3$
- Gradient $\frac{\partial L_{SE}}{\partial \hat{y}} = 1.8$ and $\frac{\partial L_{CE}}{\partial \hat{y}} = 10.0$




Cross entropy loss penalizes model more

# Practice question

**Q1. Suppose a computer program for recognizing dogs in photographs identifies eight dogs in a picture containing 12 dogs and some cats. Of the eight dogs identified, five actually are dogs while the rest are cats. Compute the Precision and recall of the computer program.**

**Solution:**

| | **Actual dogs** | **Actual cats** |
|---|---|---|
| **Predicted dogs** | TP 5 | FP 3 |
| **Predicted cats** | FN 7 | TN |

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+FP}$$

$$P = \frac{5}{5+3}$$

$$P = \frac{5}{8}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP}+FN}$$

$$R = \frac{5}{5+7}$$

$$R = \frac{5}{12}$$

# Practice question

**Q2. A database contains 80 records on a particular topic of which 55 are relevant to a certain investigation. A search was conducted on that topic and 50 records were retrieved. Of the 50 records retrieved, 40 were relevant. Construct the confusion matrix for the search and calculate the precision and recall scores for the search. Each record may be assigned a class label "relevant" or "not relevant".**

**Solution:** All the 80 records were tested for relevance. The test classified 50 records as "relevant". But only 40 of them were actually relevant.

| | Actual relevant | Actual not relevant |
|---|---|---|
| **Predicted relevant** | **40** | **10** |
| **Predicted not relevant** | **15** | **25** |

# Practice question

- TP = 40
- FP = 10
- FN = 15

The precision P is P = TP/( TP + FP)

= 40/( 40 + 10) = 4/ 5

The recall R is R = TP/( TP + FN)

= 40/( 40 + 15) = 40/ 55

# Practice question

- Using the data in the confusion matrix of a classifier of two-class dataset, several measures of performance can be calculated as well.
- Accuracy = (TP + TN)/( TP + TN + FP + FN ) = $\frac{65}{90}$
- Error rate = 1− Accuracy = 1- $\frac{65}{90} = \frac{25}{90}$
- Sensitivity = TP/( TP + FN) = $\frac{40}{55}$
- Specificity = TN /(TN + FP) = $\frac{25}{35}$
- F-measure = (2 × TP)/( 2 × TP + FP + FN) = $\frac{80}{105}$

# Practice question

**Q3. Let there be 10 balls (6 white and 4 red balls) in a box and let it be required to pick up the red balls from them. Suppose we pick up 7 balls as the red balls of which only 2 are actually red balls. What are the values of precision and recall in picking red ball?**

**Solution:**

- TP = 2
- FP = 7 − 2 = 5
- FN = 4 − 2 = 2

The precision P is P = TP/( TP + FP)

= 2/( 2 + 5) = 2/ 7

The recall R is R = TP/( TP + FN )

= 2/(2 + 2) = 1/2