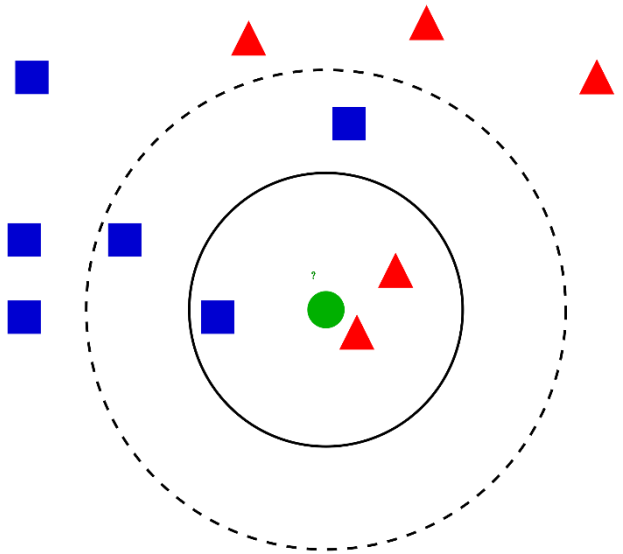
Model Evaluation (P1)

Nguyen Ngoc Thao
nnthao@fit.hcmus.edu.vn

Content outline

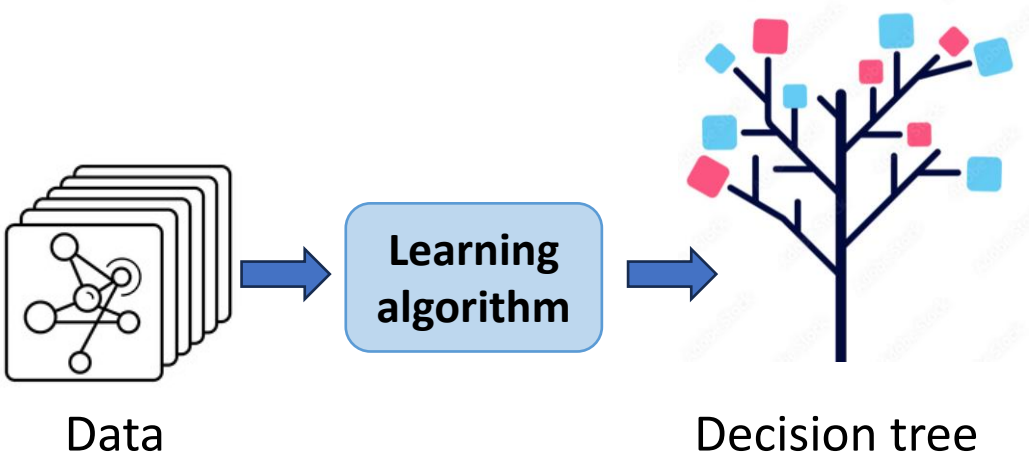
- k-nearest neighbors
- Evaluation metrics for classification
- Regression analysis
- k-means clustering
- Evaluation metrics for clustering

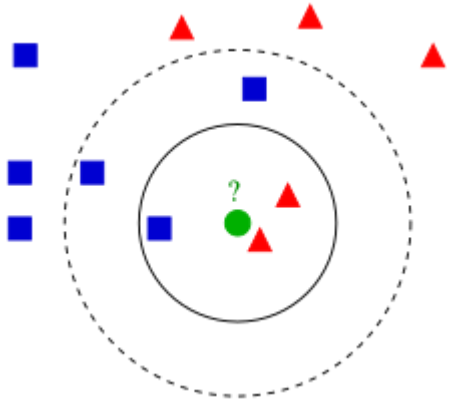
k-nearest neighbors



Lazy learning vs. Eager learning

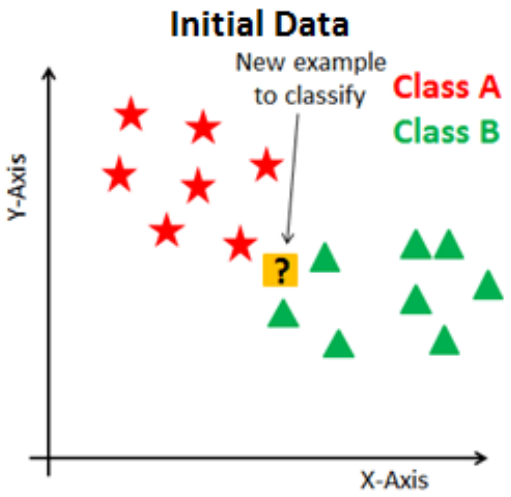
- Eager learning: Build a classification model from a given set of training examples before classifying new data.
- Lazy learning Simply store the training data (or only minor processing) and delay until a test example comes.
  - Less time in training, yet more time in predicting.

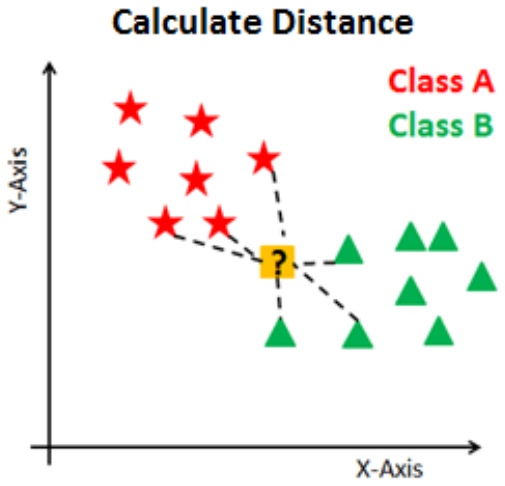




k-nearest neighbors (k-NN)

- Majority voting: Classify an object to the class most common among its k nearest neighbors.
- k is usually a small positive integer, e.g., 1, 3, 5, etc.





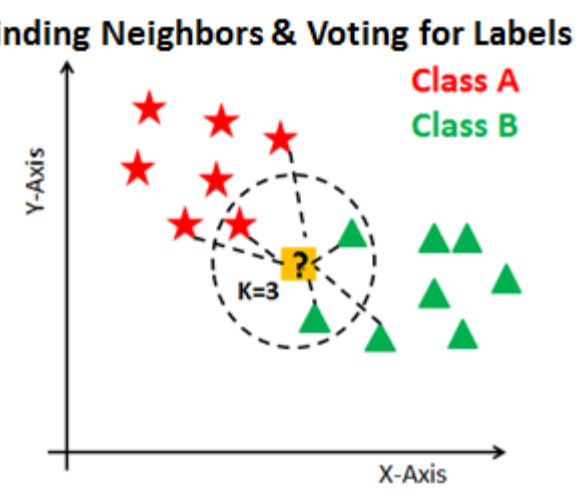
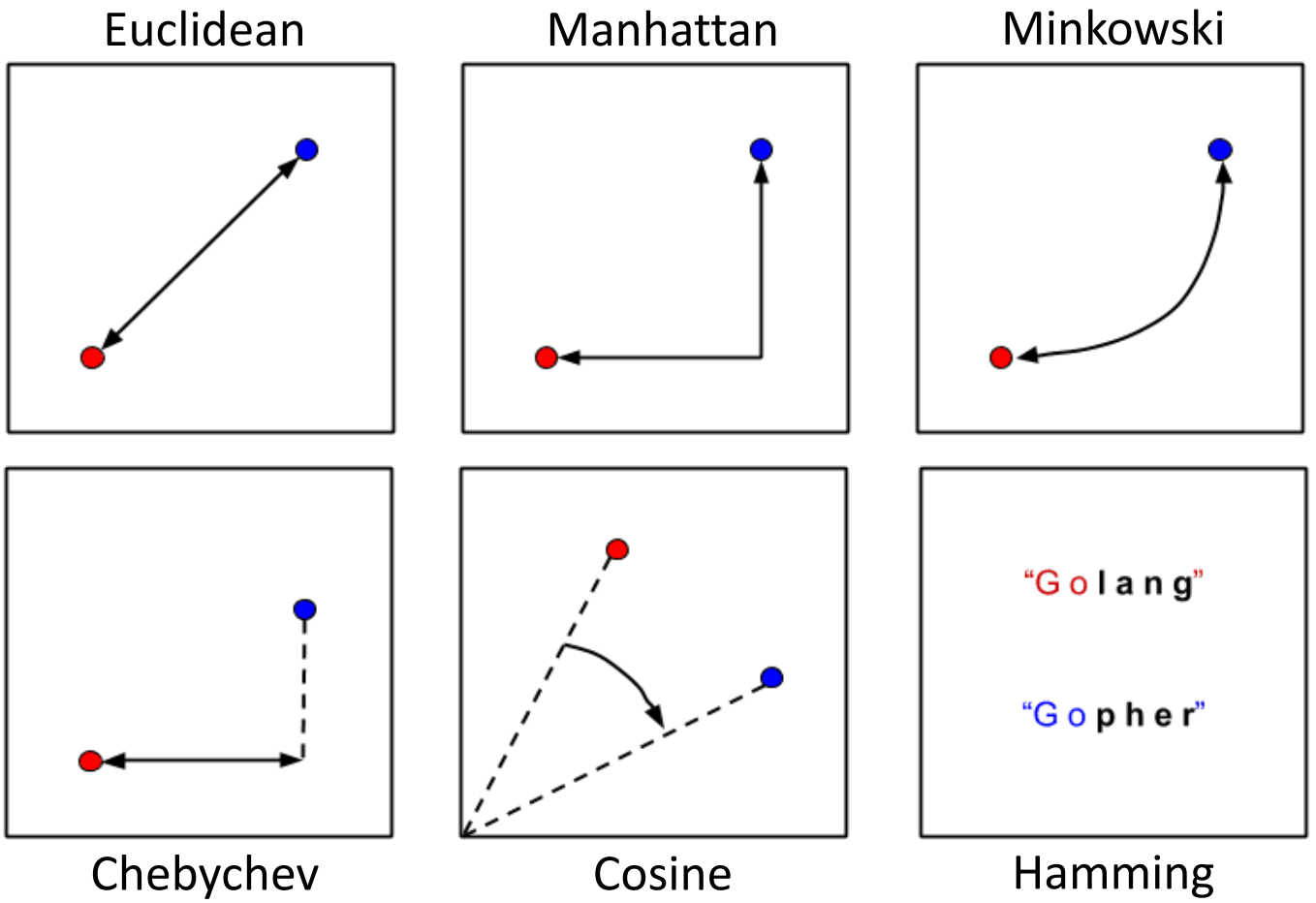


Image credit: kdnuggets

k-nearest neighbors (k-NN)

- The nearest neighbors are defined using a distance metrics.



k-nearest neighbors: An example

ID	Age	Income (K)	No. Cards	Response	L2-dist to unseen record
1	35	35	3	Yes	22.14
2	22	50	2	No	20.9
3	28	40	1	Yes	21.35
4	45	100	2	No	44.11
5	20	30	3	Yes	34.06
6	34	55	2	No	8.12
7	63	200	1	No	145.54
8	55	140	2	Yes	85.01
9	59	170	1	No	115.28
10	25	40	4	Yes	23.37
Unseen	**42**	**56**	**3**	**?**	

- **k = 5**: 3 “Yes” samples and 2 “No” samples → the class assigned is **Yes**

k-nearest neighbors: Normalization

- The attributes in the given data may have different scales, causing direct distance calculations to be inaccurate.
- For example, annual income is in dollars, and age is in years
  $\rightarrow$ income has a higher influence on the distance calculated

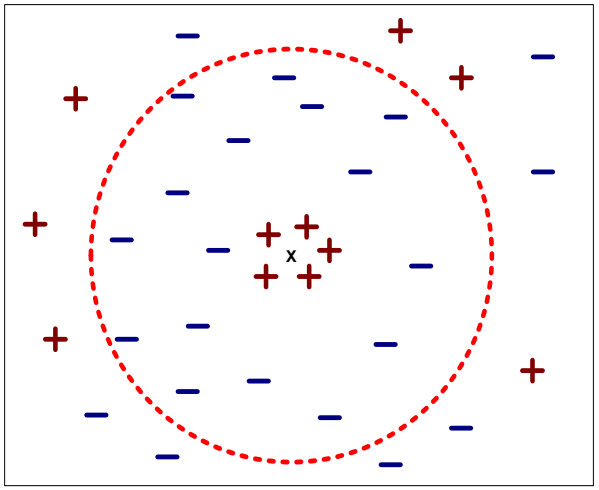
k-nearest neighbors: Normalization

ID	Age	Income (K)	No. Cards	Response	L2-dist to unseen record
1	0.35	0.03	0.67	Yes	0.21
2	0.05	0.12	0.33	No	0.57
3	0.19	0.06	0	Yes	0.75
4	0.58	0.41	0.33	No	0.43
5	0	0	0.67	No	0.53
6	0.33	0.15	0.33	No	0.38
7	1	1	0	No	1.19
8	0.81	0.65	0.33	Yes	0.67
9	0.91	0.82	0	No	1.03
10	0.12	0.06	1	Yes	0.52
Unseen	0.51	0.15	0.67	?	

- **k = 5**: 2 “Yes” samples and 3 “No” samples → the class assigned is **No**

k-nearest neighbors: Efficiency

- Easy to implement.
- Robust to noisy data by averaging k nearest neighbors



- All samples are stored $\rightarrow$ the test phase is time-consuming
- The value of k heavily affects the algorithm effectiveness.
  - Too small k: insufficient information for making decision
  - Too large k: noisy values may be included, or the neighborhood violate the areas of other classes

Quiz 01: k-nearest neighbors

1. Given the data set of vegetables and fruits below. We use two features, Sweet and Crunch, to classify food (Food Type).

No.	Object	Sweet	Crunch	Food Type
1	Grape	8	5	Fruit
2	Green bean	3	7	Vegetable
3	Nuts	3	6	Protein
4	Orange	7	3	Fruit

   Apply k-nn to predict the food type for Tomato (Sweet 6, Crunch = 4).

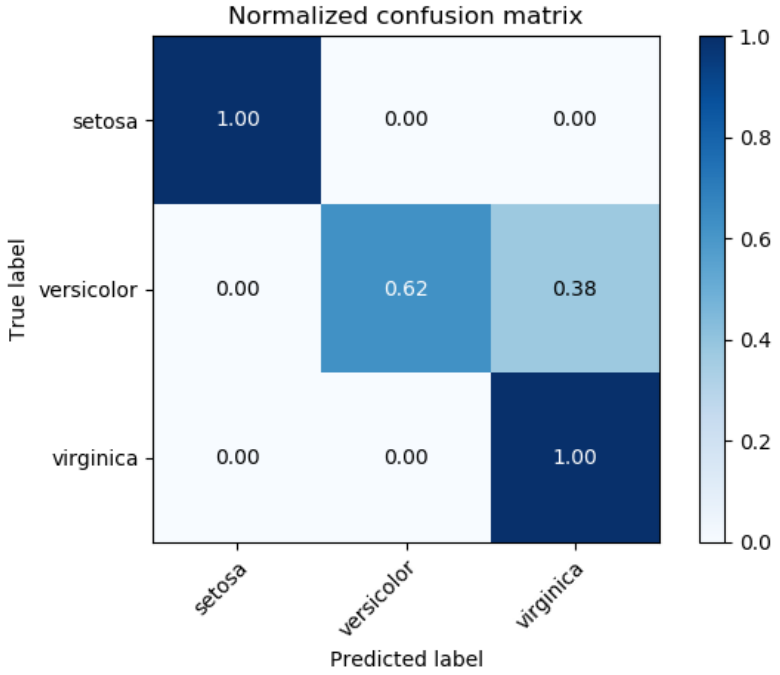
   - What kind of food does Tomato belong to if k = 1? Explain.
   - What kind of food does Tomato belong to if k = 3? Explain.

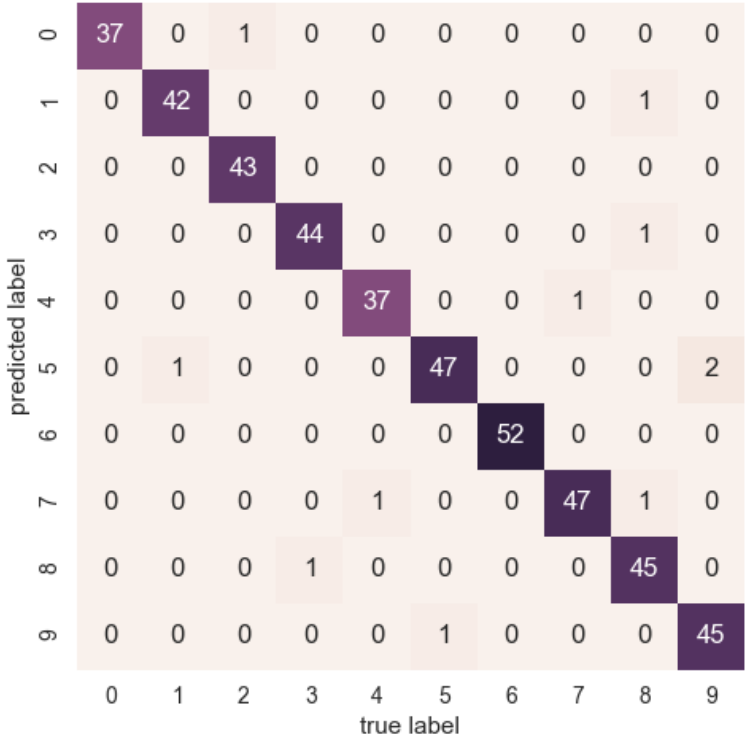
2. How to perform k-nearest neighbors in **scikit-learn**?

Evaluation metrics for classification

Confusion matrix

- An entry $[i, j]$ indicates the number of tuples in class i that were labeled by the classifier as class j.





Accuracy, Error rate, Sensitivity

		Predicted class		
		C_1	$\neg C_1$	
Actual Class	C_1	**True Positives (TP)**	**False Negatives (FN)**	**P**
	$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**	**N**
		P'	**N'**	**All**

- $Accuracy = (TP + TN)/All$
- $Error\ rate = 1 - Accuracy = \frac{FP+FN}{All}$
- $Sensitivity = TP/P$ (TP recognition rate)
- $Specificity = TN/N$ (TN recognition rate)

Precision, Recall, and F-measure

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

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

- **Recall:** completeness – % of positive tuples that the classifier labeled as positive.

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

- **F1-score**: harmonic mean of precision and recall

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

Evaluation metrics: An example

- Accuracy = (90+9560)/10000 = 0.964
- Error rate = 1 – 0.964 = 0.036
- Sensitivity = 90 / 300 = 0.3
- Specificity = 9560 / 9700 = 0.986
- Precision = 90/230 = 0.391
- Recall = 90/300 = 0.3

		Predicted class		**Total**
		cancer = yes	cancer = no	
Actual Class	cancer = yes	**90**	**210**	300
	cancer = no	**140**	**9560**	9700
Total		230	9770	10000

Quiz 02: Evaluation metrics

1. Given a confusion matrix that shows the classification results on three classes. Compute the evaluation metrics for each class.

		Actual Class			Total
		Cat	Dog	Monkey	
Predicted Class	Cat	15	10	5	**30**
	Dog	10	20	20	**50**
	Monkey	20	10	10	**40**
Total		**45**	**40**	**35**	**120**

2. How to compute these metrics in **scikit-learn**: accuracy, precision, recall, and F1?

Statistical tests of significance

- Given the two classifiers, M_1 and M_2, whose mean error rates (10-fold cross-validation) are $\overline{err}(M_1)$ and $\overline{err}(M_2)$.
- Which model is better?
- These mean error rates are just estimates of error on the true population of future data cases.
- What if the difference between the two error rates is just attributed to chance?
- Use a **test of statistical significance** to obtain **confidence limits** for error estimates.

Student's t-test

- Assume that the samples follow a t-distribution with $k-1$ degrees of freedom (in the case, $k=10$).
- Student's t-test null hypothesis $\boldsymbol{H_0}$: The two models, M_1 and M_2, have a zero difference in mean error rate.
- If $\boldsymbol{H_0}$ is rejected, the difference between M_1 and M_2 is statistically significant $\rightarrow$ choose model with lower error rate.

t-test with a single test set

- The same test set can be used for both M_1 and M_2.
- Pairwise comparison
  - For i^{th} round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
  - Average over 10 rounds to get $\overline{err}(M_1)$ and $\overline{err}(M_2)$.
- ***t*-test** computes t-statistic with $k - 1$ degrees of freedom.

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

where

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

t-test with two different test sets

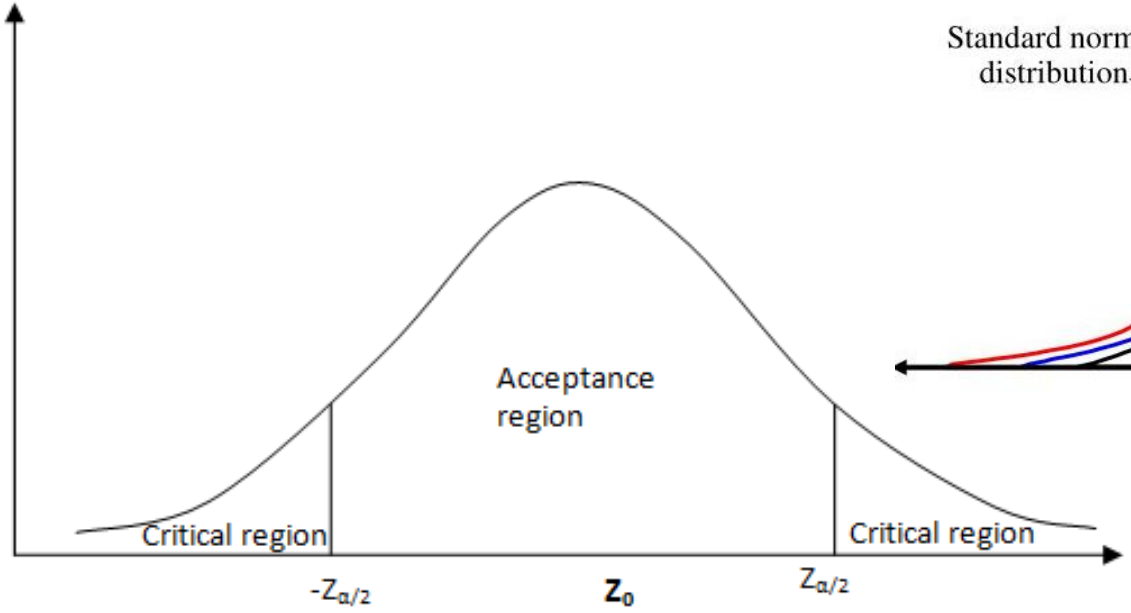
- Non-paired t-test: The variance between the means of the two models is estimated as

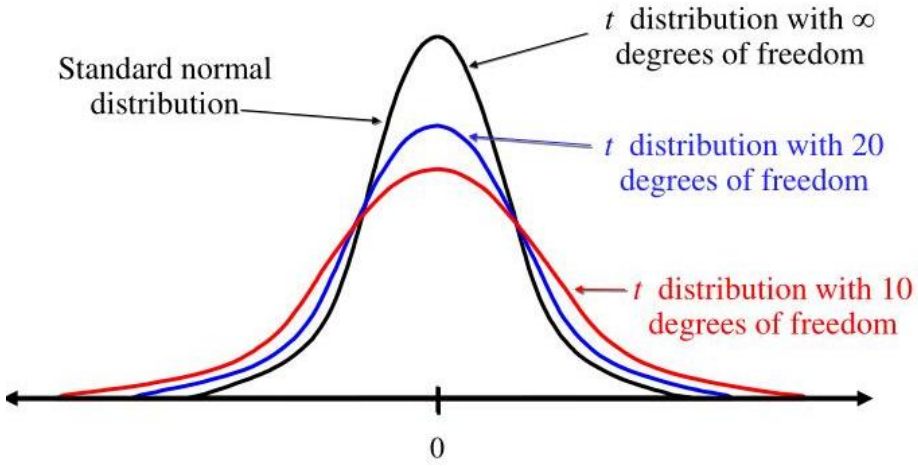
$$\boldsymbol{var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}}}$$

  - k_1 and k_2 are the number of cross-validation samples (in our case, 10-fold rounds) used for M_1 and M_2, respectively.

t-distribution

- A significance level of 5% indicates that there is a 95% confidence that your decision (reject H_0 or not) is correct, given all assumptions of the test are met.
- **Confidence limit**, $\boldsymbol{z = sig/2}$, e.g., 0.025 in this case.
- If $t > z$ or $t < -z$, the value of t lies in the rejection region, within the distribution's tails.



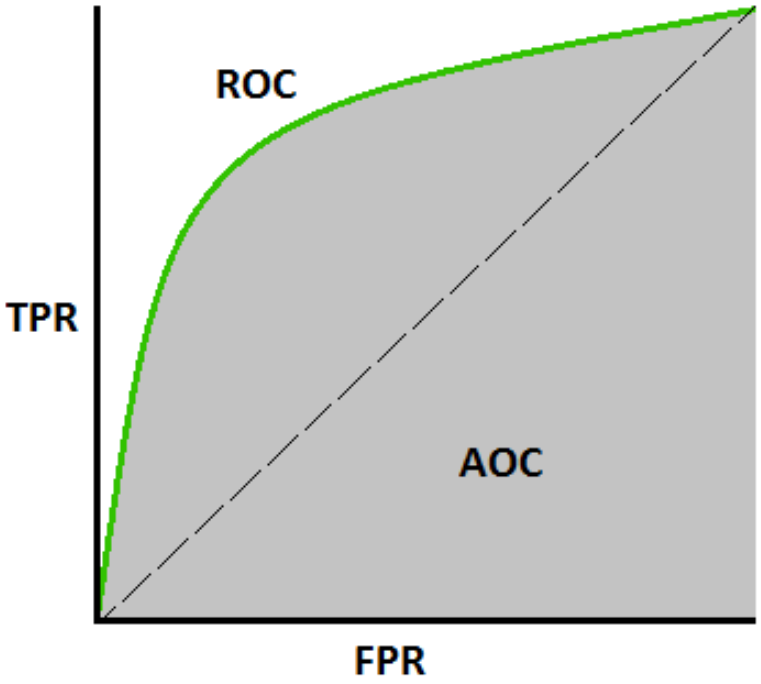


Costs and Benefits

- The cost associated with a false negative is sometimes far greater than those of a false positive.
  - E.g., incorrectly predicting a cancerous patient as not cancerous vs. incorrectly labeling a noncancerous patient as cancerous
- **Solution:** Assign a different cost to each type to outweigh one type of error over another.
  - E.g., the danger to the patient, financial costs of resulting therapies, and other hospital costs, etc.
- Similarly, the benefits associated with a true positive may be different than those of a true negative.

Receiver Operating Characteristics

- The ROC curve show the trade-off between the true positive rate (TPR) and the false positive rate (FPR).



Vertical axis represents the TPR while the horizontal axis for FPR

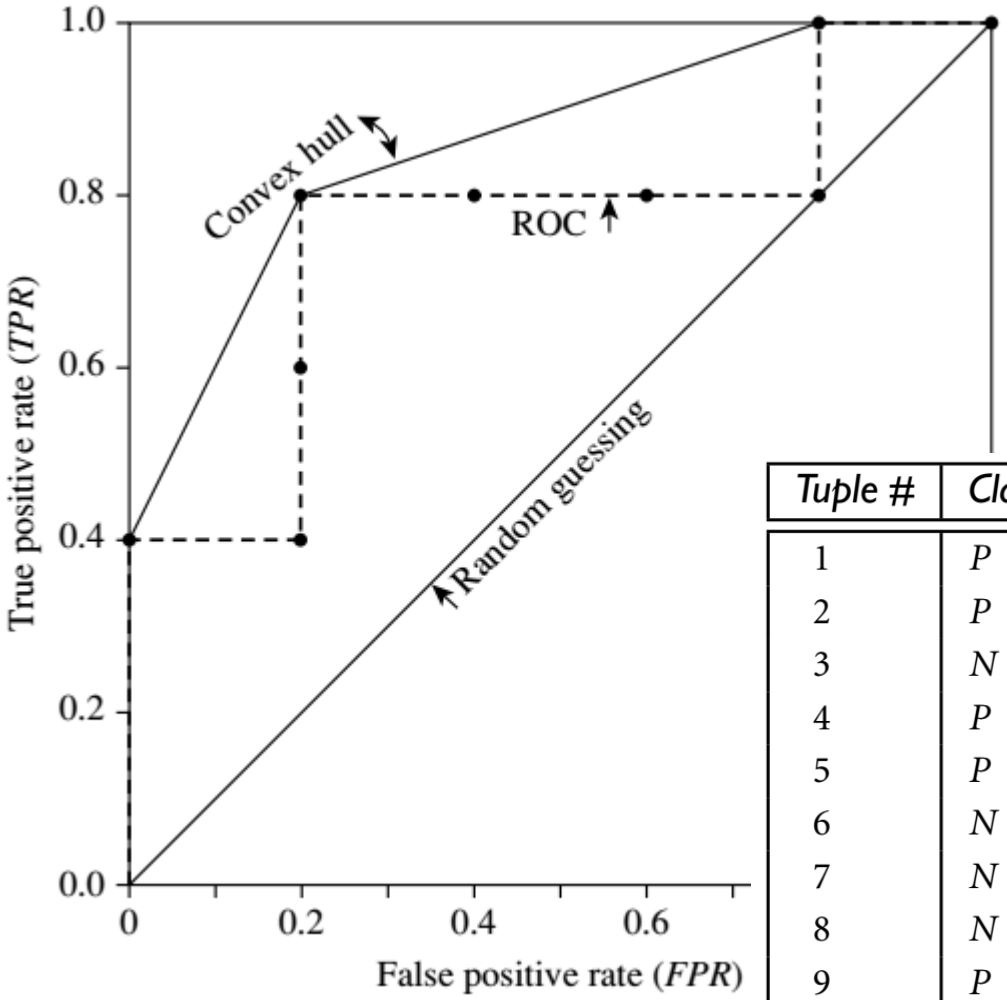
A diagonal line presents random guessing.

The area under the ROC curve is a measure of the accuracy of the model

ROC curves: Calculation

- The tuples are sorted in descending order of their likelihoods of belonging to the positive class.
  - The model M has to return a probability of the predicted class for a test tuple (e.g., naïve Bayesian, backpropagation classifiers, etc.).
- Let the value that a probabilistic classifier returns for a given tuple $\boldsymbol{X}$ be $f(\boldsymbol{X}) \rightarrow [0,1]$.
- For a binary problem, a threshold t is typically selected so that tuples where $f(\boldsymbol{X}) \geq t$ are considered positive

ROC curves: An example



There are five positive tuples and five negative tuples.

$P = 5$ and $N = 5$.

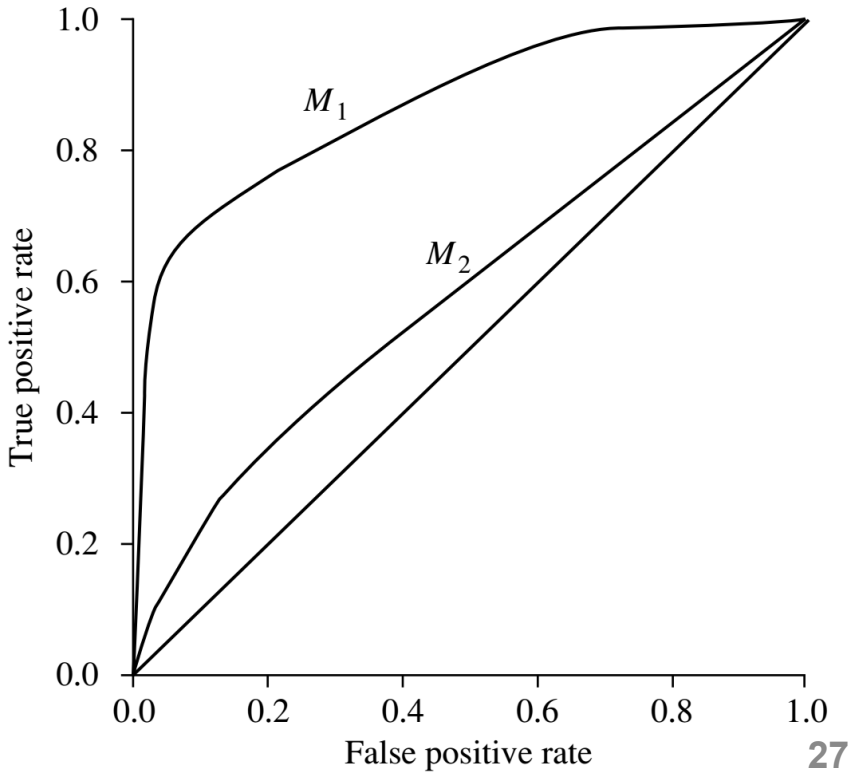
Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	0	1	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

ROC curves: Model selection

- We measure the area under the curve to assess the model.
  - A model with perfect accuracy has an area of 1.0.
- The closer the ROC curve of a model is to the diagonal line, the less accurate the model.

Which one, M_1 or M_2, is more accurate?

The closer the area is to 0.5, the less accurate the model is.



Quiz 03: ROC curves

The aside table shows the ten tuples in a test set, sorted by decreasing probability order. There are 4 positive tuples and 6 negative tuples in the test set. Each tuple is shown in a separate row.

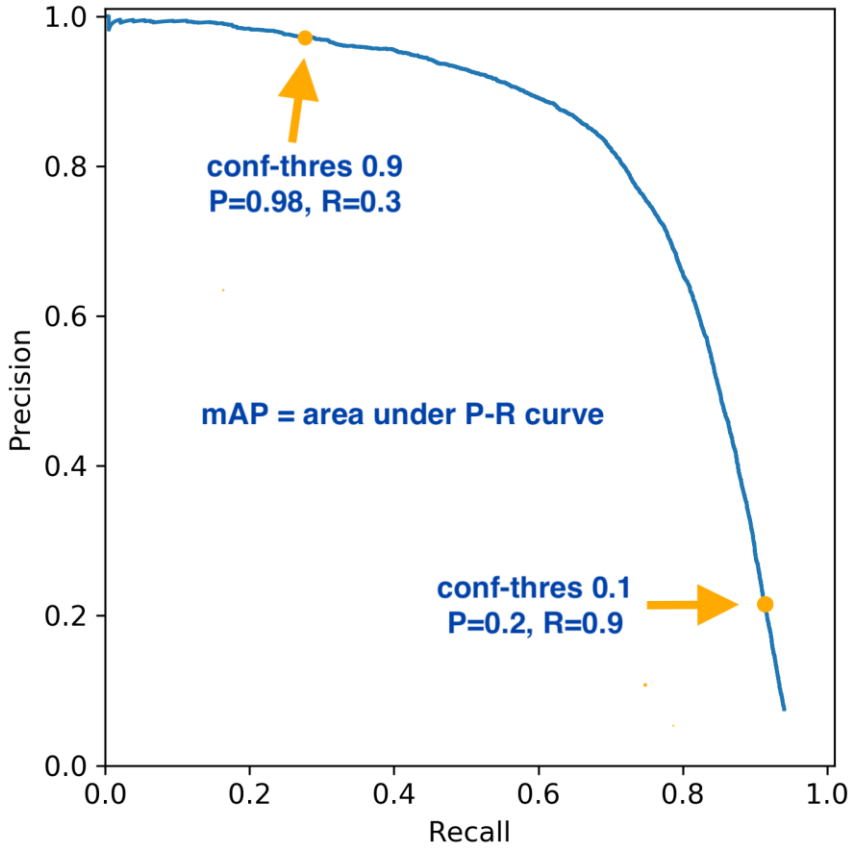
For each tuple, the first column denotes the ranking, the second column shows the actual class label of the tuples, and the third column is the probability returned by a probabilistic classifier.

1. Draw the ROC curve.
2. Can we use **scikit-learn** for ROC curves? If not, suggest another Python library.

Rank#	Class	Prob
1	**1**	0.91
2	**1**	0.86
3	0	0.85
4	**1**	0.57
5	0	0.54
6	0	0.26
7	**1**	0.18
8	0	0.16
9	0	0.14
10	0	0.13

Precision – Recall (PR) Curve

- The PR curve show the trade-off between the precision and the recall values.

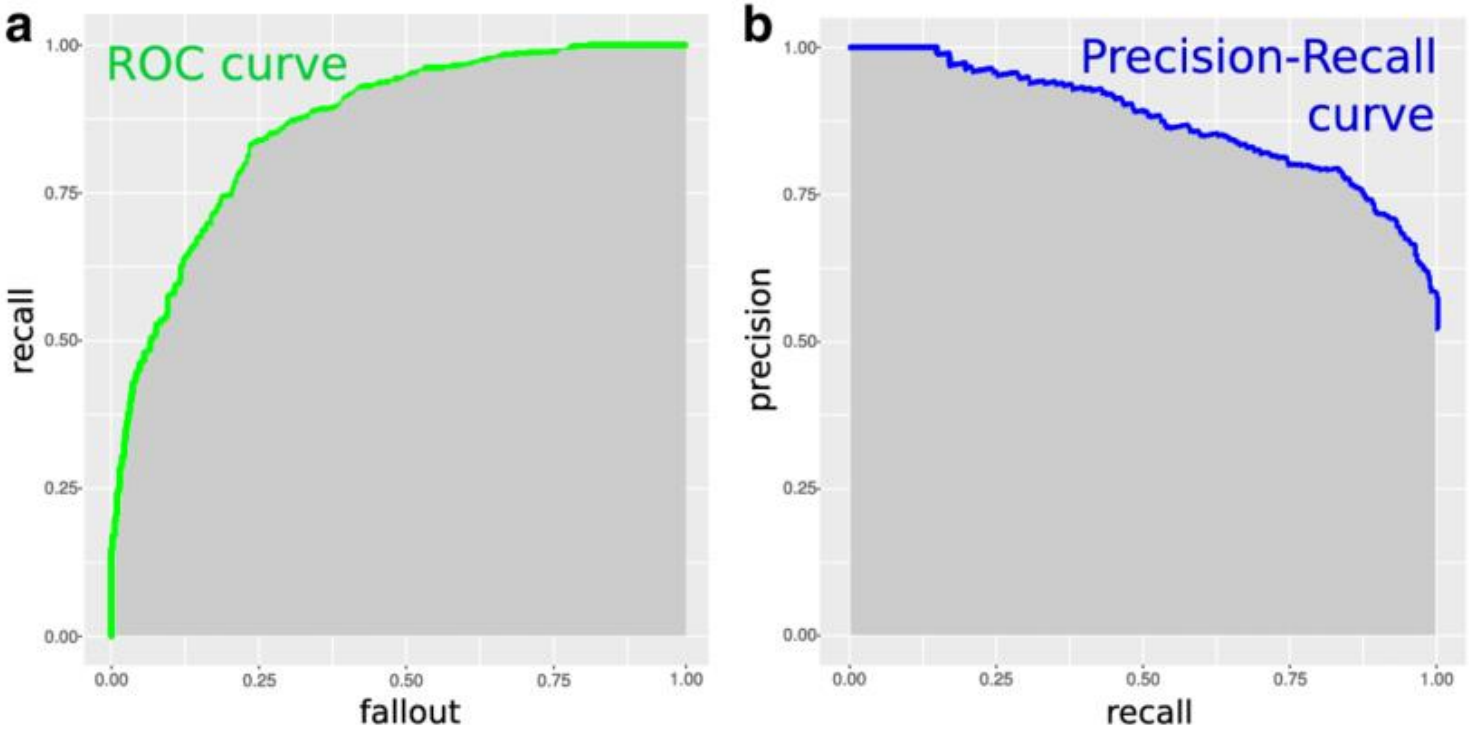


Vertical axis represents the precision while the horizontal axis for recall.

The area under the PR curve is a measure of the accuracy of the model

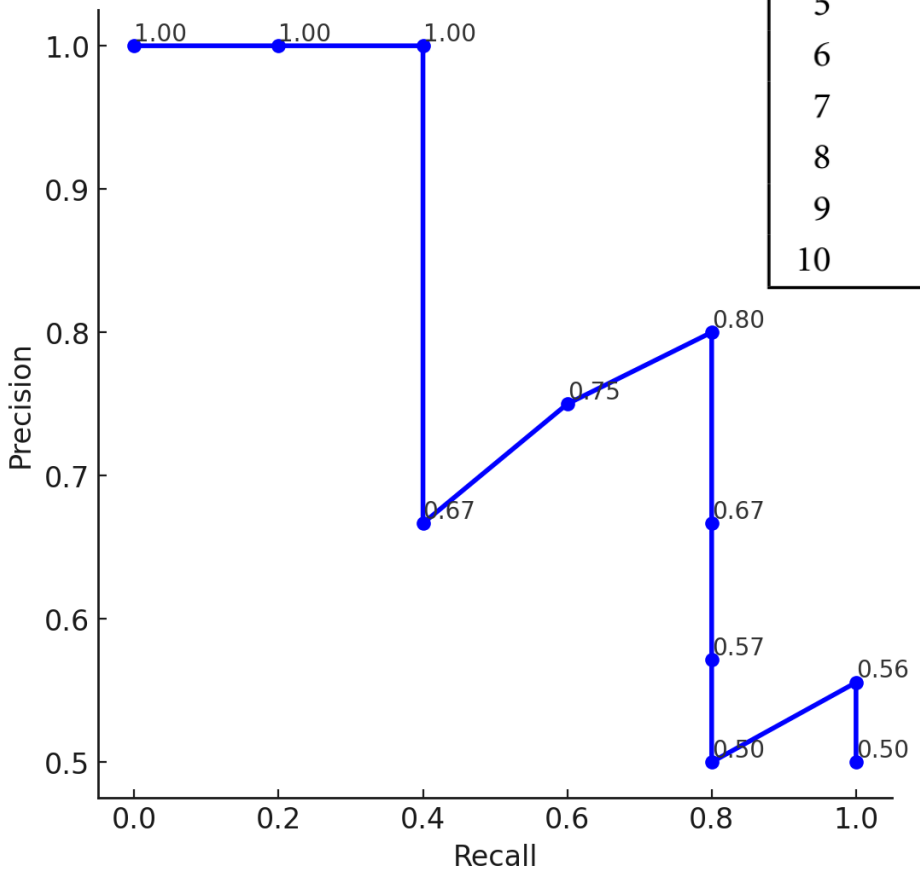
PR curve: Model selection

- Similar to ROC curves, we measure the area under the curve to assess the model.
- A model with perfect accuracy has an area of 1.0.



PR curves: An example

Tuple #	Class	Prob.	TP	FP	TN	FN	Rec	Prec
1	P	0.90	1	0	5	4	0.2	1
2	P	0.80	2	0	5	3	0.4	1
3	N	0.70	2	1	4	3	0.4	0.67
4	P	0.60	3	1	4	2	0.6	0.75
5	P	0.55	4	1	4	1	0.8	0.8
6	N	0.54	4	2	3	1	0.8	0.67
7	N	0.53	4	3	2	1	0.8	0.57
8	N	0.51	4	4	1	1	0.8	0.5
9	P	0.50	5	4	0	1	1.0	0.56
10	N	0.40	5	5	0	0	1.0	0.5



There are five positive tuples and five negative tuples.

$P = 5$ and $N = 5$.

ROC curves vs. PR curves

- ROC curve is best for balanced datasets.
  - The proportion of positive and negative classes is roughly equal.
- It evaluates performance across all decision thresholds and considers both classes equally.

- PR curve is ideal for imbalanced datasets where the positive class is rare.
- It focuses on the performance for the positive class and is more sensitive to changes in predicting positives.

Issues affecting model selection

- Classifier accuracy when predicting the class label
- Time to construct the model (training time) and time to use the model (classification / prediction time).
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability: the insights provided by the model
- Other measures: rules' goodness and compactness, and decision tree's size, etc.

Quiz 04: PR curves

The aside table shows the ten tuples in a test set, sorted by decreasing probability order. There are 4 positive tuples and 6 negative tuples in the test set. Each tuple is shown in a separate row.

For each tuple, the first column denotes the ranking, the second column shows the actual class label of the tuples, and the third column is the probability returned by a probabilistic classifier.

1. Draw the PR curve.
2. Can we use **scikit-learn** for PR curves? If not, suggest another Python library.

Rank#	Class	Prob
1	**1**	0.91
2	**1**	0.86
3	0	0.85
4	**1**	0.57
5	0	0.54
6	0	0.26
7	**1**	0.18
8	0	0.16
9	0	0.14
10	0	0.13