

# Machine Learning

## Evaluation Metrics 7

**Mostafa S. Ibrahim**

*Teaching, Training and Coaching for more than a decade!*

*Artificial Intelligence & Computer Vision Researcher*

*PhD from Simon Fraser University - Canada*

*Bachelor / MSc from Cairo University - Egypt*

*Ex-(Software Engineer / ICPC World Finalist)*

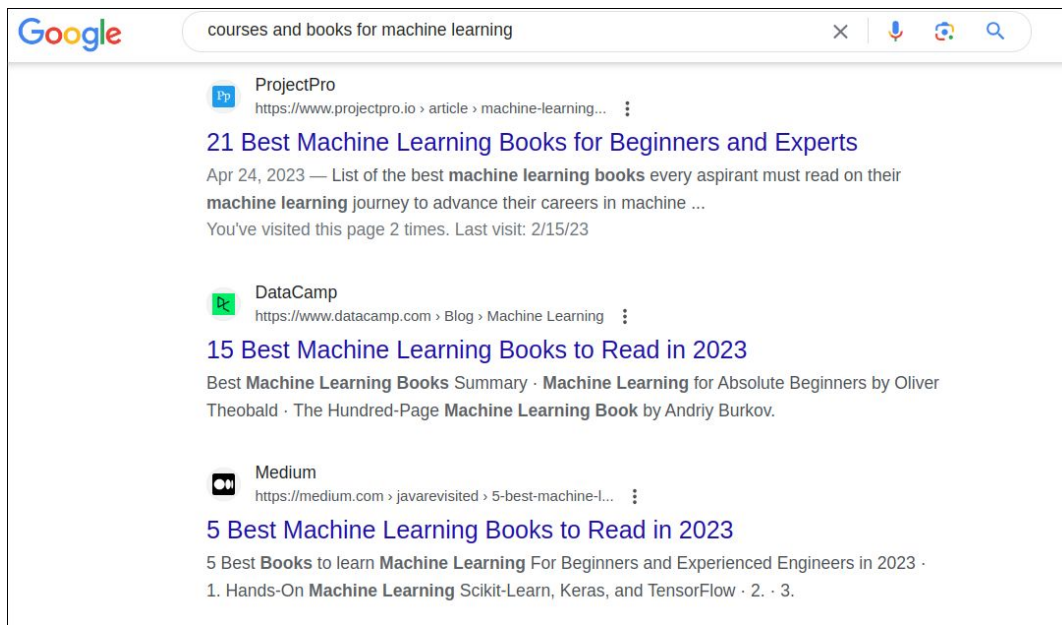


© 2023 All rights reserved.

Please do not reproduce or redistribute this work without permission from the author

# Search Engines

- If you Googled something like: *courses and books for machine learning*:
  - What will make you happy about the results?
  - How many pages will you explore?



What about the Twitter feed?



**عبدالعزيز الشثري** @A\_alsh3 · 24m

معنى أن تتجاوز مخاوف الحياة بأمان ؟  
هو أن تقترب من القرآن أكثر .

6 55 121

University of Pennsylvania · [See more](#)



**Lingjie Liu** @LingjieLiu1 · 12h

Thrilled to announce that I will be joining the University of Pennsylvania as an Assistant Professor in January 2023! I'm extremely grateful to all those who have supported me all the way and I look forward to working with students and colleagues at Penn! [@PennEngineers](#) [@CIS\\_Penn](#)

43 19 369

 [Show this thread](#)

You might like · [See more](#)



**Zhiqin Chen** @ZhiqinChen3 · 7h

(1/4) Announcing Neural Dual Contouring (NDC), a new data-driven approach to reconstructing meshes from all kinds of inputs: grids of signed or unsigned distances, binary voxels, or point clouds (without normals).

# Similarities with other Systems

- There are MANY newsfeed systems out there
  - Facebook timeline
  - Instagram feed
  - TikTok feed
  - LinkedIn feed
  - X (Twitter previously)
- In all these systems we must **personalize** our feed so that the user gets **engaged** with the content
- Search engines, recommender systems, ads prediction also share some similarity here: the **ranking**
  - We need to find items and rank (sort) them in order of **relevance** for the user

# Solving a ranking problem

- We learned classification and prediction. **Ranking** is another big area
- Our models assign a ranking score or **probability** to each item and then **sorting** the items based on these scores. +
  - In ranking, we care about the **order** of our predictions
- One way to compute such probability is **binary classifiers**
  - The probability expresses the **relevance**
- Classifier (document1, query)  $\Rightarrow$  0.7

Query: courses and books for machine learning



Relevance: 0.7



DataCamp

<https://www.datacamp.com> > Blog > Machine Learning

15 Best Machine Learning Books to Read in 2023

Best Machine Learning Books Summary · Machine Learning for Absolute Beginners by Oliver Theobald · The Hundred-Page Machine Learning Book by Andriy Burkov.

# Precision and Recall in **information retrieval**

- **Precision** is the fraction of **relevant** instances **among** the **retrieved** instances

$$\begin{aligned}\text{precision} &= \frac{\text{tp}}{\text{tp} + \text{fp}} \\ &= \frac{\text{retrieved and relevant documents}}{\text{all retrieved documents}}\end{aligned}$$

- **Recall** is the fraction of **relevant** instances that **were** retrieved

$$\begin{aligned}\text{recall} &= \frac{\text{tp}}{\text{tp} + \text{fn}} \\ &= \frac{\text{retrieved and relevant documents}}{\text{all relevant documents}}\end{aligned}$$

# Ranking Metrics

- Ranking metrics are evaluation measures for ranking models
  - Quantify how well the **ranked** list of items aligns with the **ground truth ordering**
- There are many metrics, but I would like to focus on **average precision**
  - Mainly, as it has many details that can give you hard time to understand later
  - Other metrics:
    - Precision at K
    - Recall at K
    - Normalized Discounted Cumulative Gain (**NDCG**)
    - Expected Reciprocal Rank

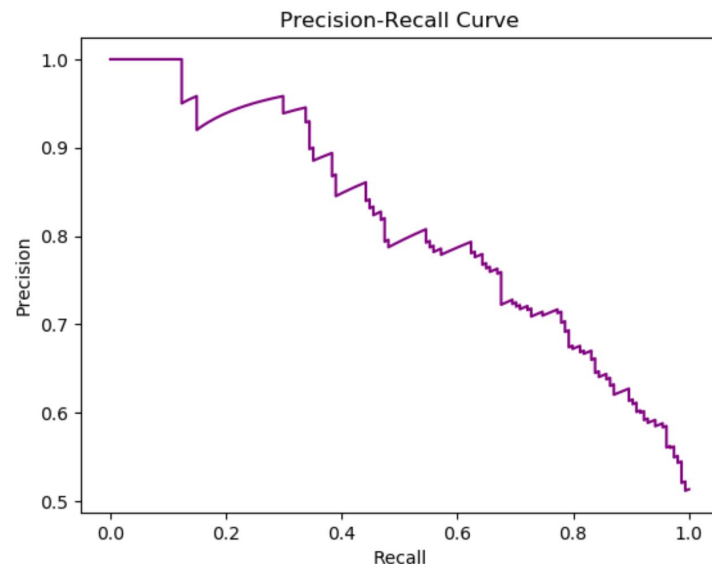
# Average Precision (AP)

- A) Imagine that we have a query for search engine. The customer wants the most relevant documents to appear first (ranking by definition)
- B) Imagine a binary classifier is evaluated on a dataset. We get images of items and their probability. What if we want to understand how correct the classifier about its **top predictions** not just its accuracy?
  - With a threshold of 0.6, correctly classified cats with probabilities 0.7 and 0.95 are both deemed correct
  - We want to reward more for being more confident and correct
- In such cases, these problems require **ranking evaluation**
- Start with ordering your models output in **decreasing** order of probability/score



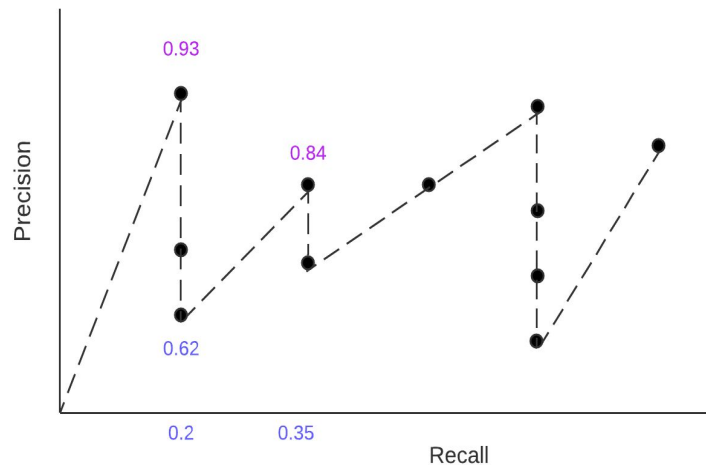
# Precision-Recall Curve and Ranking

- Looking into the y-axis
  - It represents increasing in recall (due to decreasing in threshold)
- How is that related to ranking?
- At each recall/threshold, we have evaluation for the **top predictions** so far!
- For example:
  - Threshold 0.90  $\Rightarrow$  TP = 15 and FP = 2
  - Threshold 0.85  $\Rightarrow$  TP = 25 and FP = 7
  - Threshold 0.80  $\Rightarrow$  TP = 35 and FP = 21
    - Decreasing precisions with less threshold



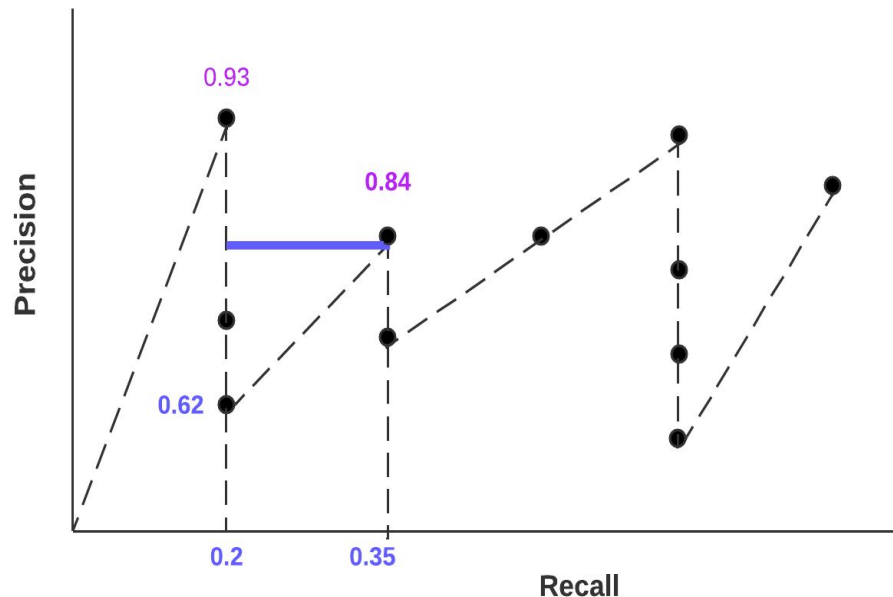
# Precision-Recall Curve and the area

- To compute the curve area, we can divide the space into many little **trapezoids**, compute their area and sum them
- Here is another way based on **rectangles**
- Let's approximate the numbered area
  - Recall from 0.2 to 0.35
    - This is a rectangle base
  - What about the rectangle height?
    - Note, 0.84 point has 0.62 before it
      - Multiple points of same recall



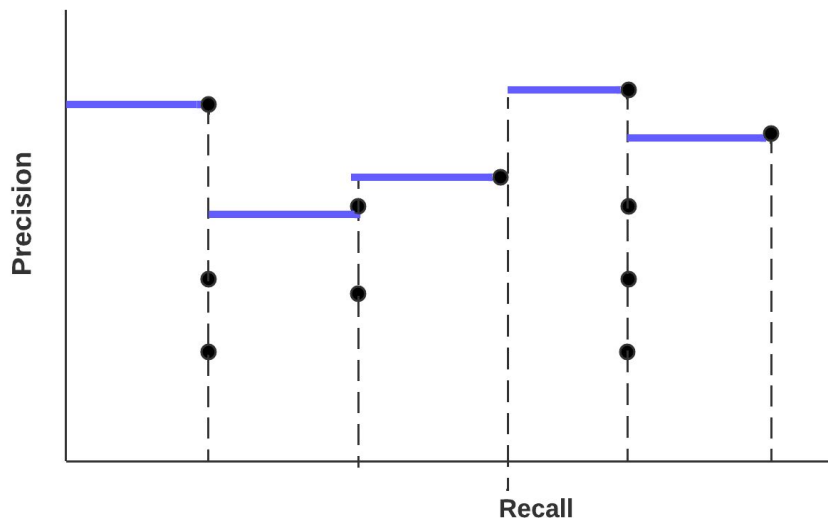
# Precision-Recall Curve and the area

- One way is to take the **2nd** precision of this region as the height of the rectangle
  - Now height: 0.84 precision
  - Base is:  $0.35 - 0.2 = 0.15$
  - Area of this region:  
 $0.15 * 0.84 = \mathbf{0.126}$



# Precision-Recall Curve and the area

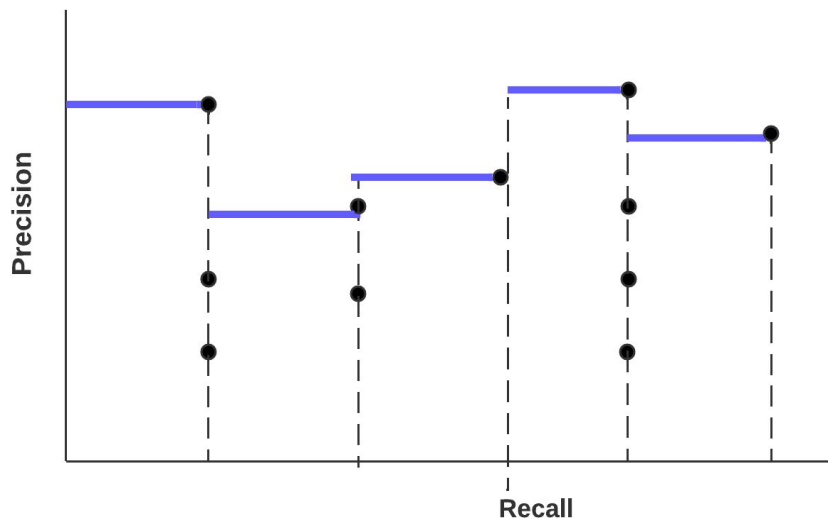
- Then, for each 2 consecutive different recalls, just add these areas
  - Same recalls: width = 0
- This is what we approximate
- Let's make a formula



# Precision-Recall Curve and the area

- **For every example**, we compute its added rectangle area as:
  - The increase in the recall from the last example
  - The current precision
- Example by example
  - $\text{recall} = \text{TP} / P$
  - If no new tp, then the same recall
    - Hence zero difference
  - If a new tp example
    - Then the difference is  $1 / P$ 
      - $(\text{tp}+1 - \text{tp}) / P$

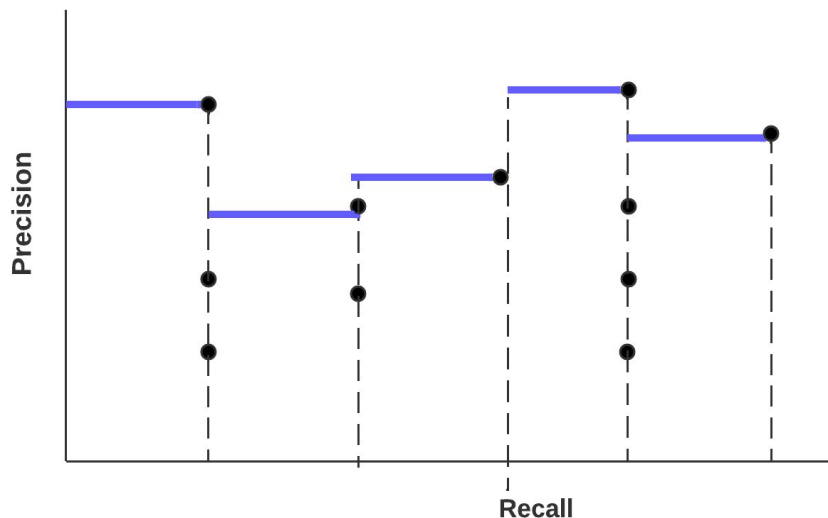
$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n$$



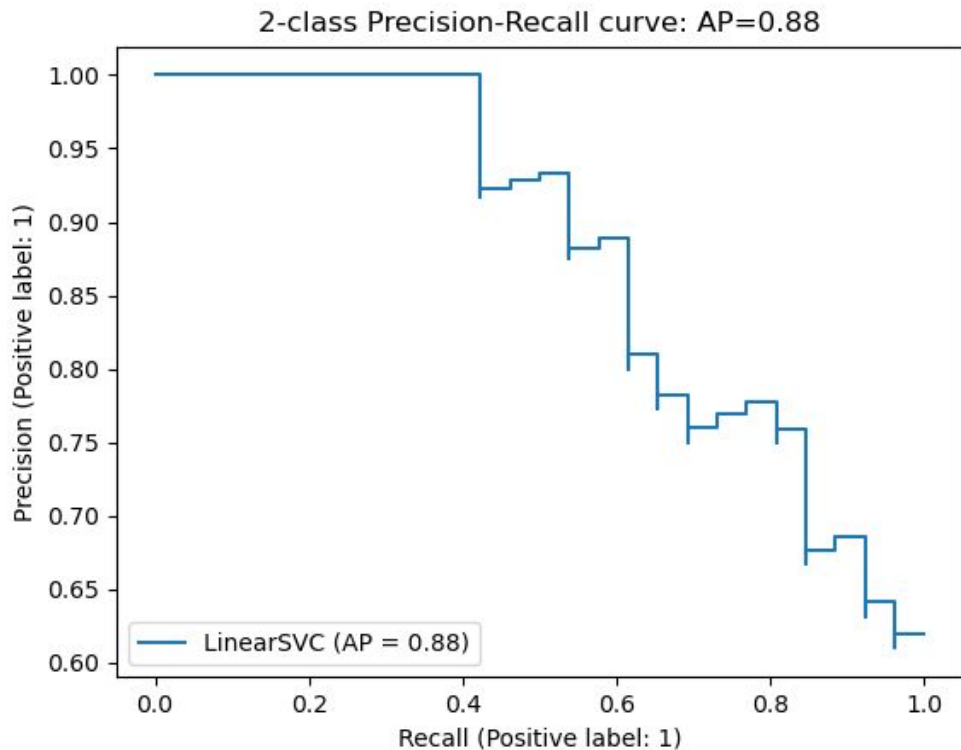
# Precision-Recall Curve and **AP**

- In other words, precision values for each relevant item is averaged
- This average is the **average precision**
  - $ap = \frac{1}{P} \sum \text{Precision}[n]$  if nth item is tp
  - P = total ground truth positive examples
- Below formula is generic
  - If you want to specific **specific** recalls instead of every example

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

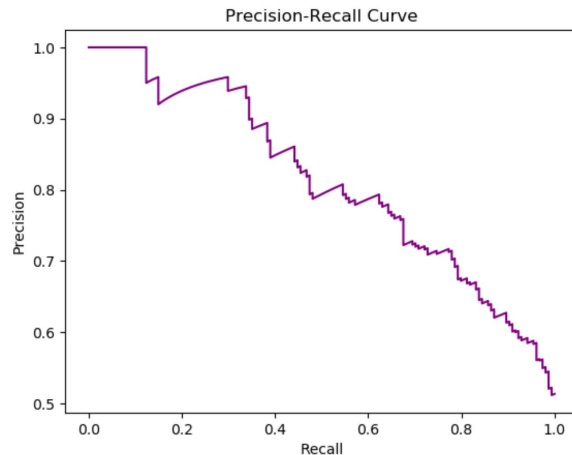


# Precision-Recall Curve and **AP**



# Precision-Recall Curve and AP

- Average Precision is a **single** scalar value that **summarizes** the **precision-recall curve** into a single measure
- It quantifies the **overall ranking** performance of the model by **averaging** the **precision** values at **different recall** levels
- What does it mean that the binary classifier has average precision of 0.333?
- on average, the classifier achieves a precision of 33.3% when making predictions.






# Implementing AP

- First of all, be careful there are more than a formulas
  - E.g. 11 Point Interpolation algorithm only evaluates at recalls:  $[0, 0.1, 0.2, 0.3, \dots, 1]$
  - Don't do paper comparisons unless the SAME formula
- Implementing AP is tricky!
  - What if a single example?
  - What if the first element (highest score) is TP?
    - We need to add this element.
    - Formula gives sense we start from index = 1 and compare with index 0
  - What if we have 2 equal scores: one with  $gt = 1$  and one with  $gt = 0$ 
    - In practice, rare to happen
  - Sorting then can lead to
    - $[1 \ 0] \Rightarrow \text{Precision} = 1 \qquad 1/1$
    - $[0 \ 1] \Rightarrow \text{Precision} = 0.5 \qquad 1/2$

```

4 def average_precision_score_our(ground_truth, scores):
5     # Sort the ground truth in descending order of scores
6     sorted_indices = np.argsort(scores)[::-1]
7     sorted_ground_truth = [ground_truth[i] for i in sorted_indices]
8     precision_sum, true_positives = 0, 0
9
10    for k in range(len(ground_truth)): # tip: bug to miss index = 0
11        if sorted_ground_truth[k]: # tp
12            true_positives += 1
13            pr = true_positives / (k + 1)
14            precision_sum += pr
15
16    return precision_sum / sum(sorted_ground_truth)
17
18
19 if __name__ == '__main__':
20     scores = [0.8, 0.6, 0.3, 0.2, 0.9, 0.75, 0.81, 0.92] # avoid duplicates
21     ground_truth = [1, 0, 0, 1, 0, 0, 1, 1]
22
23

```




Index	Score	GT	tps	Precision: tps/(idx+1)
0	0.92	1	1	1/1
1	0.9	0	1	False Positive
2	0.81	1	2	2/3
3	0.8	1	3	3/4
4	0.75	0	3	False Positive
5	0.6	0	3	False Positive
6	0.3	0	3	False Positive
7	0.2	1	4	4/8
				$AP = (1 + \frac{2}{3} + \frac{3}{4} + 4/8) / 4 = 0.729$

Notes:

- tps = accumulation gt
- idx +1 = TP + FP
- 4 = P = total GT +positives

```
y_prop = [0.99, 0.98, 0.97, 0.96, 0.95, 0.94, 0.93, 0.92, 0.91] # sorted
print(average_precision_score([1, 0, 0, 0, 0, 0, 0, 0, 0], y_prop)) # 1
print(average_precision_score([1, 1, 1, 1, 0, 0, 0, 0, 0], y_prop)) # 1
print(average_precision_score([1, 0, 0, 0, 0, 0, 0, 0, 1], y_prop)) # 0.611
print(average_precision_score([0, 1, 0, 0, 0, 0, 0, 0, 0], y_prop)) # 1/2
print(average_precision_score([0, 1, 0, 1, 0, 0, 0, 0, 0], y_prop)) # 1/2
print(average_precision_score([0, 1, 0, 1, 0, 1, 0, 1, 0], y_prop)) # 1/2 each 2nd
print(average_precision_score([0, 0, 1, 0, 0, 0, 0, 0, 0], y_prop)) # 1/3
print(average_precision_score([0, 0, 1, 0, 0, 1, 0, 0, 0], y_prop)) # 1/3
print(average_precision_score([0, 0, 1, 0, 0, 1, 0, 0, 1], y_prop)) # 1/3 each 3rd elem
print(average_precision_score([0, 0, 0, 1, 0, 0, 0, 1, 0], y_prop)) # 1/4 each 4th elem
print(average_precision_score([0, 0, 0, 0, 0, 0, 0, 0, 1], y_prop)) # 0.11
print(average_precision_score([0, 0, 0, 0, 0, 0, 0, 1, 1], y_prop)) # 0.173
print(average_precision_score([1, 0, 0, 0, 0, 0, 0, 1, 1], y_prop)) # 0.527
print(average_precision_score([1, 1, 0, 0, 0, 1, 0, 1, 1], y_prop)) # 0.71
print(average_precision_score([1, 1, 0, 0, 0, 0, 1, 1, 1], y_prop)) # 0.696
print(average_precision_score([1, 1, 1, 1, 1, 1, 1, 1, 0], y_prop)) # 1.00
print(average_precision_score([1, 1, 1, 1, 1, 1, 1, 0, 1], y_prop)) # 0.98
print(average_precision_score([1, 1, 1, 1, 1, 1, 0, 1, 1], y_prop)) # 0.97
print(average_precision_score([1, 1, 1, 1, 1, 0, 1, 1, 1], y_prop)) # 0.95
print(average_precision_score([1, 1, 1, 1, 0, 1, 1, 1, 1], y_prop)) # 0.93
print(average_precision_score([1, 1, 1, 0, 1, 1, 1, 1, 1], y_prop)) # 0.90
print(average_precision_score([1, 1, 0, 1, 1, 1, 1, 1, 1], y_prop)) # 0.87
print(average_precision_score([1, 0, 1, 1, 1, 1, 1, 1, 1], y_prop)) # 0.83
print(average_precision_score([0, 1, 1, 1, 1, 1, 1, 1, 1], y_prop)) # 0.77
```



# Understanding AP

- AP considers the ordering of **both** relevant and non-relevant items and averages the **precision** values based on the position of **relevant** items in the list.
  - Hence, once we reach the last 1 in gt, all zeros (non-relevant) items after that doesn't matter
    - [1, 1, 1, 0, 0, 0, 0] for a sorted list has  $AP = 1$
- AP **rewards** models that place **relevant** items higher in the ranking order and **penalizes** those that rank **relevant** items lower
- AP takes into account the precision at different recall levels

# Mean Average Precision (mAP)

- Assume our dataset has 4 classes: cat, dog, cow, rate
- How do we apply AP?
- Assume we have an evaluation dataset of 1000 examples
- Imagine we query your classifier with cat
  - Prepare ground truth with 1 is cat and 0 is non-cat
  - Prepare your probabilities to express that
  - Compute AP-cat for this result
- Repeat the process for each category
- Now we have 4 APs, one per class
- Average them. We call this mAP. We use it with multiclass datasets

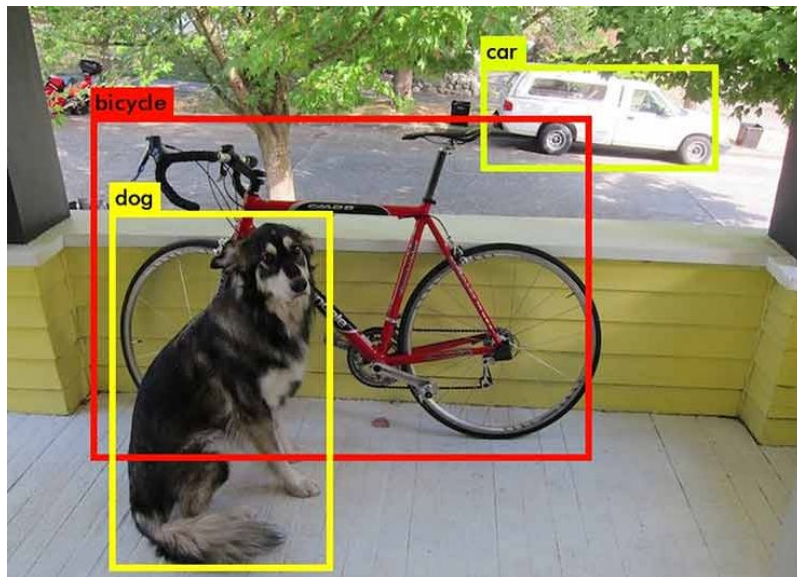
$$mAP = \frac{1}{k} \sum_i^k AP_i$$

# Drawbacks of AP

- We must know the relevant items count, which is not always possible
  - NDCG (normalized discounted cumulative gain) metric doesn't
- Sensitivity to the ranking order
  - As we saw, even a small change in the order can significantly affect the AP value
- The model penalizes severely for discovered TP so late rank
  - But in practice, sometimes it doesn't matter (e.g. user browses first 3 pages of google)
  - One solution: compute **AP@K** (only for first K elements)
- Drawback of mAP: it treats all classes equally. A model can work great for some classes and fail for some of them (e.g. as dataset is imbalance or the data of this class is noisy). Check class by class first

# Question!

- Apply the AP concepts for the object detection task!
- How can we compute AP for e.g. the dog category?



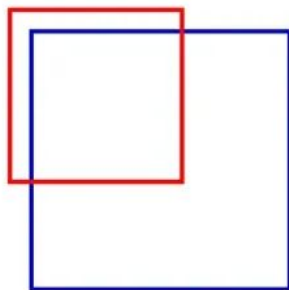
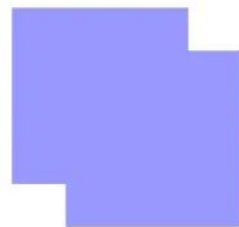
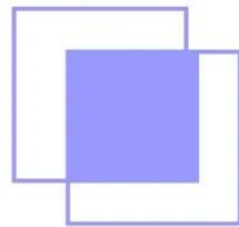


# AP for object detection

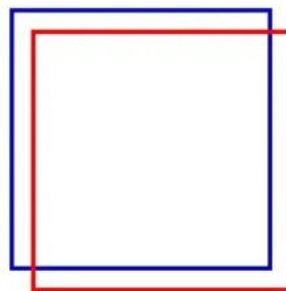
- Nothing especially
- Ground Truth: All the gt bounding boxes for the dog
- Your model output bounding boxes for all images and their scores
- Sort all your boxes by score
- If the model proposes a box
  - If this box matches a box in ground truth  $\Rightarrow$  TP
  - If this box doesn't match any box in ground truth  $\Rightarrow$  FP
- If there is unmatched box in gt  $\Rightarrow$  FN
- How can we decide 2 boxes match?

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

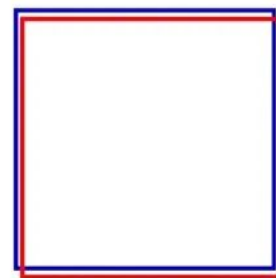
- Compute their intersection
- Divide by the union area
- Use high threshold, like 0.75



Poor



Good



Excellent

# Common interview question in CV

- Write python code to compute intersection over union (iou)
- Your job :)
  - Think how to intersect 2 rectangles (find a simple way, e.g. 1 line of code given coordinates)
- `def calculate_iou(box1, box2):`
  - Box is a list or numpy of 4 values: x1, y1, x2, y2
- Make sure your code is robust for corner cases

# Materials

- [Video](#) (Solid)
- [Article](#)
- AUC/AP: [Article](#), [Article](#), [Article](#), [Article](#), [Article](#)

*“Acquire knowledge and impart it to the people.”*

*“Seek knowledge from the Cradle to the Grave.”*

