Machine Learning

| Precision/Recall Trade-off

Prof. Dr. Mostafa Elhosseini

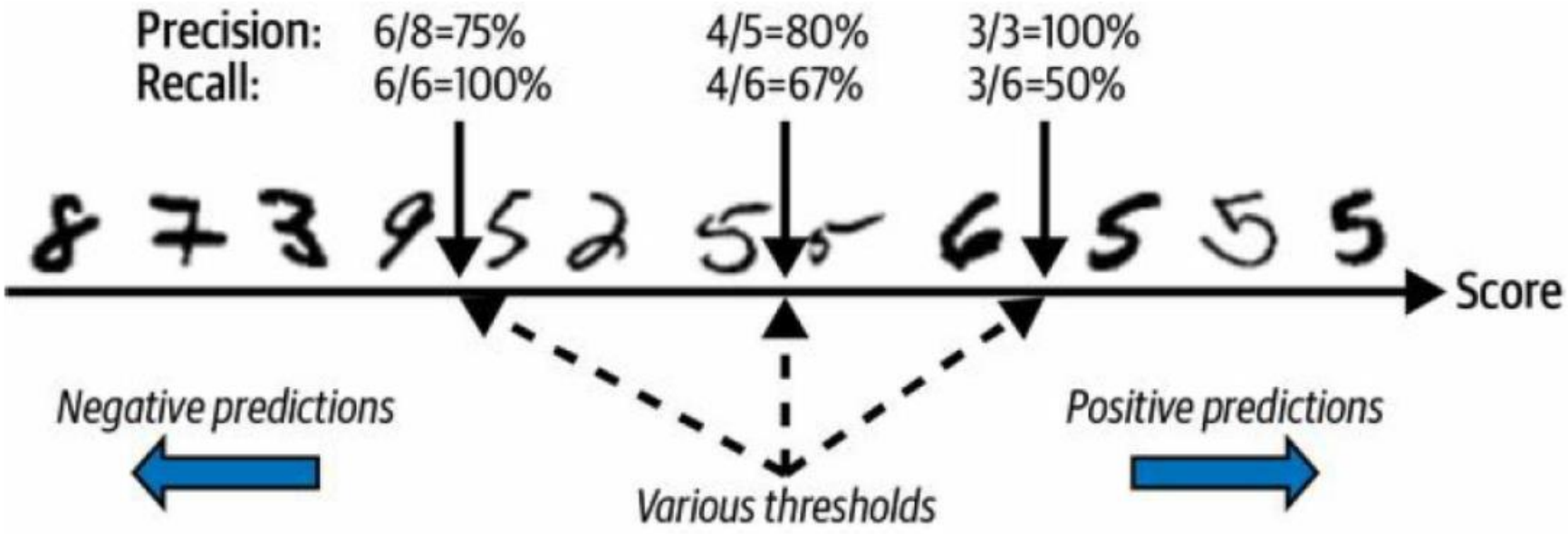
https://youtube.com/@ProfElhosseiniTechVerse

Agenda

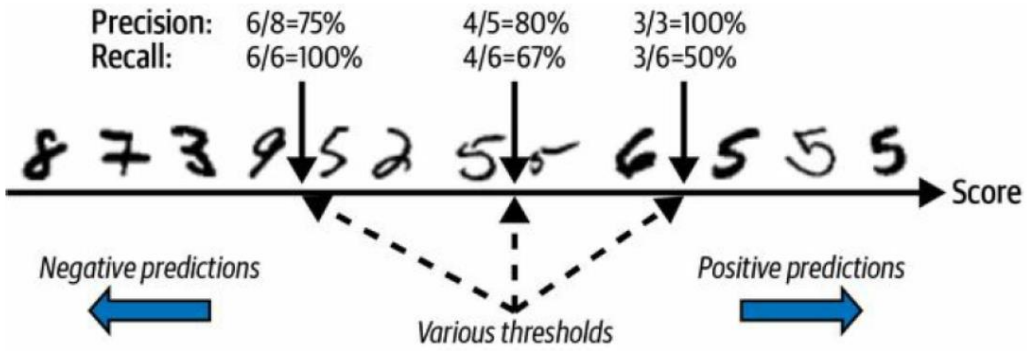
- Precision/Recall Trade-off

Precision/Recall Trade-off

- For each instance, the classifier **computes a score** based on a decision function, and if that score is greater than a **threshold**, it assigns the instance to the **positive class**, or **else** it assigns it to the negative class



Precision/Recall Trade-off



- Now if you raise the threshold (move it to the arrow on the right), the false positive (the 6) becomes a true negative, thereby increasing precision (up to 100% in this case), but one true positive becomes a false negative, decreasing recall down to 50%.
- Conversely, lowering the threshold increases recall and reduces precision - So **how can you decide** which threshold to use?

Precision/Recall Trade-off

- As you can see, it is fairly easy to create a classifier with virtually any precision you want: just set a high enough threshold, and you're done.
- Hmm, not so fast. A high-precision classifier is not very useful if its recall is too low!
- If someone says "let's reach 99% precision," you should ask, "at what recall?"

Precision/Recall Trade-off

- In some situations, we might know that we want to maximize either recall or precision at the expense of the other metric
- We can’t have too good recall either because it is always a tradeoff. Having the perfect recall would most likely require allowing many false positives (terrible precision)

Precision/Recall Trade-off

- Scikit-Learn does not let you set the threshold directly, but it does **give you access** to the **decision scores** that it uses to make predictions. Instead of calling the classifier's predict() method, you can call its **decision_function()** method, which returns a score for each instance, and then **use any threshold** you want to make predictions based on those scores

Precision/Recall Trade-off

```
y_scores = sgd_clf.decision_function([some_digit])
y_scores
```

```
array([2164.22030239])
```

```
threshold = 0
y_some_digit_pred = (y_scores > threshold)
```

```
y_some_digit_pred
```

```
array([ True])
```

```
>>> threshold = 3000
>>> y_some_digit_pred = (y_scores > threshold)
>>> y_some_digit_pred
array([False])
```

Precision/Recall Trade-off

How do you decide which threshold to use?

- use the **cross_val_predict()** function - specify that you want to return **decision scores** instead of predictions:

```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                             method="decision_function")
```

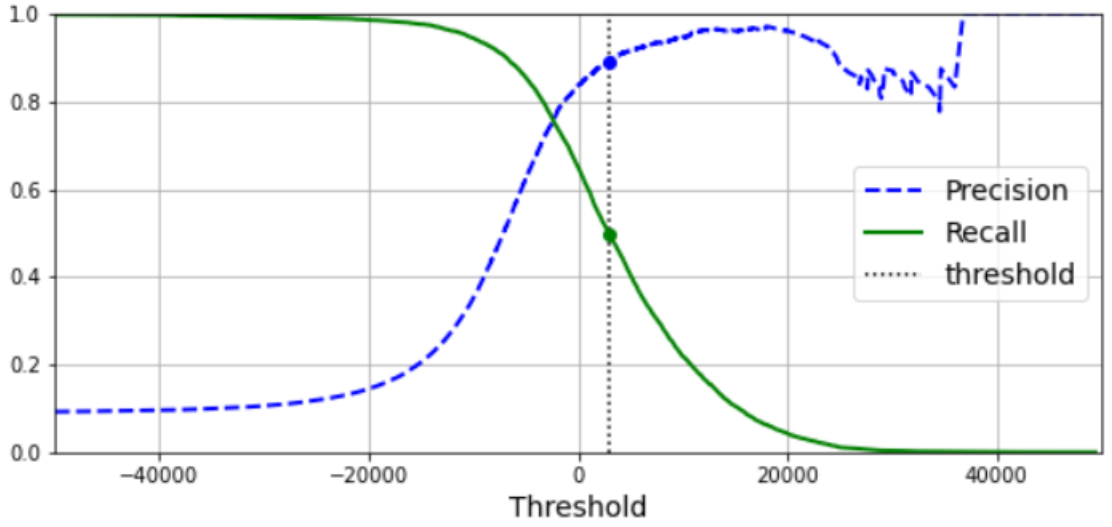
- With these scores, use the precision_recall_curve() function to compute precision and recall for all possible thresholds

```
from sklearn.metrics import precision_recall_curve

precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

Precision/Recall Trade-off

- Let's show the threshold of **3,000** we selected



- precision is near 90% and recall is
- You may wonder why the precision curve is **bumpier** than the recall curve

Precision/Recall Trade-off

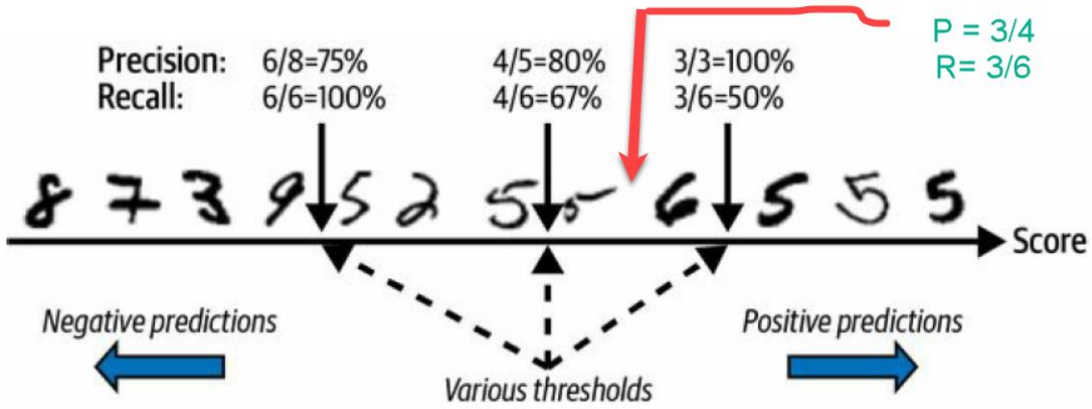
- You may wonder why the precision curve is **bumpier** than the recall curve
- The precision curve often appears bumpier than the recall curve due to the nature of how precision and recall are calculated and how they respond to changes in the classification threshold.
- As the **classification threshold changes**, the number of positive predictions (true positives + false positives) can vary more dramatically than the number of actual positive instances (true positives + false negatives).
- This is because the number of actual positive instances remains constant, while the number of predicted positive instances can change significantly with the threshold.

Precision/Recall Trade-off

- At high thresholds, the classifier makes fewer positive predictions, resulting in higher precision as the predictions are more conservative. However, small changes in the number of false positives can lead to significant changes in precision, causing bumps in the curve.
- As the threshold decreases, the classifier starts making more positive predictions, increasing the likelihood of false positives. The increase in false positives can cause sudden drops in precision, resulting in more bumps in the curve.

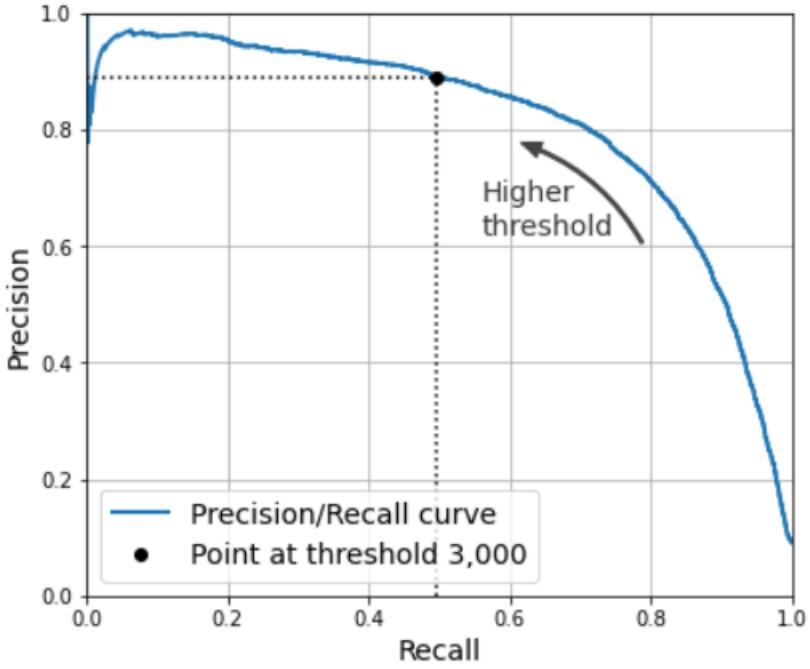
Precision/Recall Trade-off

- The bumpiness of the precision curve is due to its sensitivity to false positives, which can change more abruptly with the classification threshold compared to the change in false negatives that affects recall. The recall curve tends to be smoother because the number of actual positive instances remains constant, and the increase in true positives is typically more gradual as the threshold decreases.



Precision/Recall Trade-off

- Another way to select a good precision/recall tradeoff is to plot precision directly against recall



Precision/Recall Trade-off

- Suppose you decide to aim for 90% precision
- Search for the lowest threshold that gives you at least 90% precision

```
idx_for_90_precision = (precisions >= 0.90).argmax()
threshold_for_90_precision = thresholds[idx_for_90_precision]
threshold_for_90_precision
```

```
3370.0194991439557
```

Precision/Recall Trade-off

- To make predictions (on the training set for now), instead of calling the classifier's predict() method, you can run this code:

```
y_train_pred_90 = (y_scores >= threshold_for_90_precision)
```

```
precision_score(y_train_5, y_train_pred_90)
```

```
0.9000345901072293
```

```
recall_at_90_precision = recall_score(y_train_5, y_train_pred_90)
recall_at_90_precision
```

```
0.4799852425751706
```

Next Lecture

- The receiver operating characteristic (ROC) curve is another common tool used with binary classifiers