From:
DS100 UC Berkeley

Classification:
Logistic Regression II

Introduction to Data Science
Spring 1403

Yadollah Yaghoobzadeh

Goals for this Lecture

Introducing Maximum likelihood estimation for logistic regression

Performance metrics for classification

Agenda

- Maximum likelihood estimation
- Performance metrics

Maximum Likelihood Estimation

$$R(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right)$$

It may have seemed like we just pulled cross-entropy loss out of thin air.

CE loss is justified by a probability analysis technique called maximum likelihood estimation. Read on if you would like to learn more.

Recorded walkthrough: link.

The No-Input Binary Classifier

Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

{0, 0, 1, 1, 1, 1, 0, 0, 0, 0}

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

The No-Input Binary Classifier

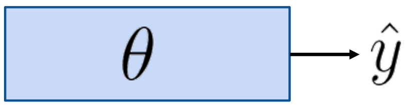
Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

{0, 0, 1, 1, 1, 1, 0, 0, 0, 0}

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).



Parameter θ: Probability that flip == 1 (Heads)

Prediction: 1 or 0

1. Of the below, which is the best theta θ? Why?
A. 0.8 B. 0.5 C. 0.4
D. 0.2 E. Something else

2. For the next flip, would you predict 1 or 0?

The No-Input Binary Classifier

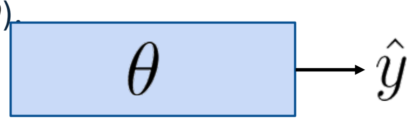
Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

{0, 0, 1, 1, 1, 1, 0, 0, 0, 0}

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).



1. Of the below, which is the best theta θ? Why?

A. 0.8 **B.** 0.5 **C.** 0.4
D. 0.2 **E.** Something else

The No-Input Binary Classifier

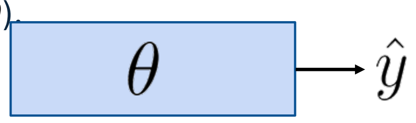
Suppose you observed some outcomes of a coin (1 = Heads, 0 = Tails):

{0, 0, 1, 1, 1, 1, 0, 0, 0, 0}

Training data has only responses $\mathbb{Y}$ (no features $\mathbb{X}$)

For the next flip, do you predict heads or tails?

A reasonable model is to **assume all flips are IID** (i.e., same coin; same prob. of heads θ).



1. Of the below, which is the best theta θ? Why?

A. 0.8 **B.** 0.5 **C.** 0.4
D. 0.2 **E.** Something else

0.4 is the most "intuitive" for two reasons:

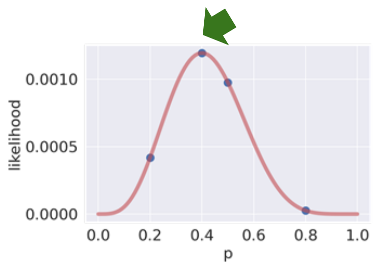
1. Frequency of heads in our data.
2. Maximizes the **likelihood** of our data.

Likelihood of Data; Definition of Probability

A Bernoulli random variable Y with parameter p has distribution:

$$P(Y=y)=\begin{cases} p & \text{if } y=1 \\ 1-p & \text{if } y=0 \end{cases}$$

Given that all flips are IID from the same coin (probability of heads = p), the **likelihood** of our data is **proportional to** the probability of observing the datapoints.



Training data: [0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

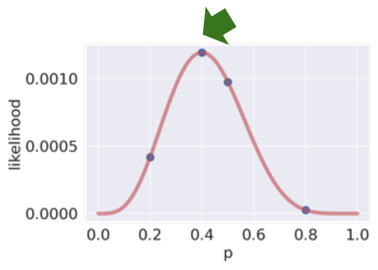
Data likelihood: $p^4(1-p)^6$.

Likelihood of Data; Definition of Probability

A Bernoulli random variable Y with parameter p has distribution:

$$P(Y=y)=\begin{cases} p & \text{if } y=1 \\ 1-p & \text{if } y=0 \end{cases}$$

Given that all flips are IID from the same coin (probability of heads = p), the **likelihood** of our data is **proportional to** the probability of observing the datapoints.



Training data: [0, 0, 1, 1, 1, 1, 0, 0, 0, 0]

Data likelihood: $p^4(1-p)^6$.

An example of a bad estimate is parameter since the likelihood of observing the training data is going to be :

$$(0.1)^4(0.9)^6 = 0.000053$$

An example of a bad estimate is parameter since the likelihood of observing the training data is going to be :

$$(0.4)^4(0.6)^6 = 0.001194$$

Generalization of the Coin Demo

$$\begin{tikzcd} \boxed{\theta} \arrow[r] & \hat{y} \end{tikzcd}$$

Parameter θ: Probability that IID flip == 1 (Heads)

Prediction: 1 or 0

For training data: {0, 0, 1, 1, 1, 1, 0, 0, 0, 0}

0.4 is the most "intuitive" θ for two reasons:

1. Frequency of heads in our data
2. Maximizes the **likelihood** of our data:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \left(\theta^4(1-\theta)^6\right)$$

How can we generalize this notion of likelihood to **any** random binary sample?

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

A Compact Representation of the Bernoulli Probability Distribution

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

How can we generalize this notion of likelihood to **any** random binary sample?

Let Y be Bernoulli(p). The probability distribution can be written compactly:

$$P(Y=y) = \underbrace{p^y}_{\text{For P(Y = 1), only this term stays}}\underbrace{(1-p)^{1-y}}_{\text{For P(Y = 0), only this term stays}}$$

(long, non-compact form):

$$P(Y=y) = \begin{cases} p & \text{if } y=1 \\ 1-p & \text{if } y=0 \end{cases}$$

A Compact Representation of the Bernoulli Probability Distribution

How can we generalize this notion of
likelihood to **any** random binary sample?

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

Let Y be Bernoulli(p). The probability distribution can be written compactly:

$$P(Y = y) = \underbrace{p^y}_{\substack{\text{For P(Y = 1), only} \\ \text{this term stays}}} \underbrace{(1-p)^{1-y}}_{\substack{\text{For P(Y = 0), only} \\ \text{this term stays}}}$$

If binary data are **IID with same** probability p, then the likelihood of the data is:

$$\prod_{i=1}^{n} p^{y_i}(1-p)^{(1-y_i)}$$

Ex: $\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\} \rightarrow p^4(1-p)^6$

A Compact Representation of the Bernoulli Probability Distribution

How can we generalize this notion of
likelihood to **any** random binary sample?

$$\underbrace{\{y_1, y_2, \ldots, y_n\}}_{\text{data (1's and 0's)}} \Rightarrow \hat{\theta} = \underset{\theta}{\operatorname{argmax}} \underbrace{(???)}_{\text{likelihood}}$$

If binary data are **IID with same** probability p, then the likelihood of the data is:

$$\prod_{i=1}^{n} p^{y_i}(1-p)^{(1-y_i)}$$

Ex: $\{0, 0, 1, 1, 1, 1, 0, 0, 0, 0\} \rightarrow p^4(1-p)^6$

If data are independent with **different** probability p_i, then the likelihood of the data is:

$$\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$$

(spoiler: for logistic regression, $p_i = \sigma(X_i^T\theta)$)

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

- For i = 1, 2, ..., n, let Y_i be independent Bernoulli(p_i) and Observed data $\{y_1, y_2, \ldots, y_n\}$
- We'd like to estimate $p_1, p_2, \ldots, p_n$

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize** $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

Maximum Likelihood Estimation (MLE)

Our **maximum likelihood estimation** problem:

- For i = 1, 2, ..., n, let Y_i be independent Bernoulli(p_i). Observed data $\{y_1, y_2, \ldots, y_n\}$
- We'd like to estimate $p_1, p_2, \ldots, p_n$

Find $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ that **maximize** $\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{maximize}} \quad \log\left(\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$$

(log is an increasing function. If a > b, then log(a) > log(b).)

$$= \sum_{i=1}^{n} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right) = \sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

Maximum Likelihood Estimation (MLE)

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{maximize}} \quad \log\left(\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}\right)$$

(log is an increasing function. If a > b, then log(a) > log(b).)

$$= \sum_{i=1}^{n} \log\left(p_i^{y_i}(1-p_i)^{(1-y_i)}\right) = \sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{minimize}} \quad \underbrace{-\frac{1}{n}}\sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

Maximizing Likelihood == Minimizing Average Cross-Entropy

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{maximize}} \quad \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$$

Log is increasing; max/min properties

$$\underset{p_1, p_2, \ldots, p_n}{\textbf{minimize}} \quad -\frac{1}{n}\sum_{i=1}^{n} \left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right)$$

For logistic regression, let $p_i = \sigma(X_i^T\theta)$

$$\underset{\theta}{\textbf{minimize}} \quad \underbrace{-\frac{1}{n}\sum_{i=1}^{n} \left(y_i \log(\sigma(X_i^T\theta) + (1-y_i)\log(1-\sigma(X_i^T\theta))\right)}_{\textbf{Cross-Entropy Loss!!}}$$

Minimizing cross-entropy loss is equivalent to **maximizing the likelihood of the training data**.

- We are choosing the model parameters that are "most likely", given this data.

Assumption: all data drawn **independently** from the same logistic regression model with parameter θ

We Did it!

	Regression ($y \in \mathbb{R}$)	Classification ($y \in \{0, 1\}$)
1. Choose a model ✓	Linear Regression $\hat{y} = f_\theta(x) = x^T\theta$	Logistic Regression $\hat{P}_\theta(Y = 1 \vert x) = \sigma(x^T\theta)$
2. Choose a loss function ✓	Squared Loss or Absolute Loss	Average Cross-Entropy Loss $-\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\sigma(X_i^T\theta)) + (1 - y_i)\log(1 - \sigma(X_i^T\theta))\right)$
3. Fit the model	Regularization Sklearn/Gradient descent	Regularization Sklearn/Gradient descent
4. Evaluate model performance	R^2, Residuals, etc.	?? (next time)

Why More Performance Metrics?

We've already introduced cross-entropy loss – why do we need additional ways of assessing how well our models perform?

- In linear regression, we made numerical predictions and used a loss function to determine how "good" these predictions were.
- In logistic regression, our ultimate goal is to *classify* data – we are more concerned with whether or not each datapoint was assigned the correct class using the decision rule.

The performance metrics we are about to introduce will assess the quality of classifications, not the predicted probabilities.

Classifier Accuracy

Now that we actually have our classifier, let's try and quantify how well it performs.

$$\text{accuracy} = \frac{\#\text{ of points classified correctly}}{\#\text{ points total}}$$

The most basic evaluation metric for a classifier is **accuracy**.

```
def accuracy(X, Y):
    return np.mean(model.predict(X) == Y)

accuracy(X, Y) # 0.794
```

```
model.score(X, Y) # 0.794
```

(sklearn documentation)

While widely used, the accuracy metric is **not so meaningful** when dealing with **class imbalance**.

Pitfalls of Accuracy: A Case Study

Suppose we're trying to build a classifier to filter spam emails (Project B!).

- Each email is **spam** (1) or **ham** (0).

Let's say we have 100 emails, of which only **5** are truly **spam**, and the remaining **95** are truly **ham**.

Your friend ("Friend 1"):

Classify *every* email as **ham** (0).

$$\hat{y} = \text{classify}_{\text{friend}}(x) = 0$$

1. What is the accuracy of your friend's classifier?
2. Is accuracy a good metric of this classifier's performance?

Pitfalls of Accuracy: A Case Study

Suppose we're trying to build a classifier to filter spam emails.

- Each email is **spam** (1) or **ham** (0).

Let's say we have 100 emails, of which only **5** are truly **spam**, and the remaining **95** are **ham**.

Your friend ("Friend 1"):

Classify every email as **ham** (0).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

High accuracy…
…but we detected **none** ⚠️ of the spam!!!

Pitfalls of Accuracy: A Case Study

Suppose we're trying to build a classifier to filter spam emails.

- Each email is **spam** (1) or **ham** (0).

Let's say we have 100 emails, of which only **5** are truly **spam**, and the remaining **95** are **ham**.

Accuracy is not always a good metric for classification, particularly when your data have **class imbalance** (e.g., very few 1's compared to 0's).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

High accuracy…
…but we detected **none** ⚠️ of the spam!!!

Types of Classifications

- There are four different classifications that our model might make:
  - True positive: correctly classify a positive point as being positive ($y = 1,\ \hat{y} = 1$)
  - False positive: incorrectly classify a negative point as being positive ($y = 0,\ \hat{y} = 1$)
  - False negative: incorrectly classify a positive point as being negative ($y = 1,\ \hat{y} = 0$)
  - True negative: correctly classify a negative point as being negative ($y = 0,\ \hat{y} = 0$)

"**positive**" means a prediction of **1**.
"**negative**" means a prediction of **0**.

A confusion matrix plots these quantities for a particular classifier and dataset.

Prediction $\hat{y}$

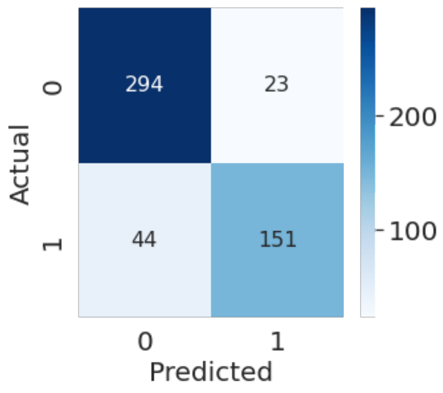
Actual y	**0**	**1**
0	True **negative** (TN)	False **positive** (FP)
1	False **negative** (FN)	True **positive** (TP)

Types of Classifications

A confusion matrix plots these quantities for a particular classifier and dataset.

In `sklearn`:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_true, Y_pred)
```



Accuracy, Precision, and Recall

$$\text{accuracy} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?

Prediction

Actual	0	1
0	TN	FP
1	FN	TP

Accuracy, Precision, and Recall

$$\text{accuracy} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?

Precision and recall are two commonly used metrics that measure performance even in the presence of class imbalance.

Prediction

Actual	0	1
0	TN	FP
1	FN	TP

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

Of all observations that were predicted to be 1, what proportion were actually 1?

- How **precise** is our classifier **when it is positive**?
- Penalizes false positives.

Accuracy, Precision, and Recall

$$\text{accuracy} = \frac{TP + TN}{n}$$

What proportion of points did our classifier classify correctly?

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

Precision and **recall** are two commonly used metrics that measure performance even in the presence of class imbalance.

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

Of all observations that were predicted to be 1, what proportion were actually 1?

- How accurate is our classifier when it is positive?
- Penalizes false positives.

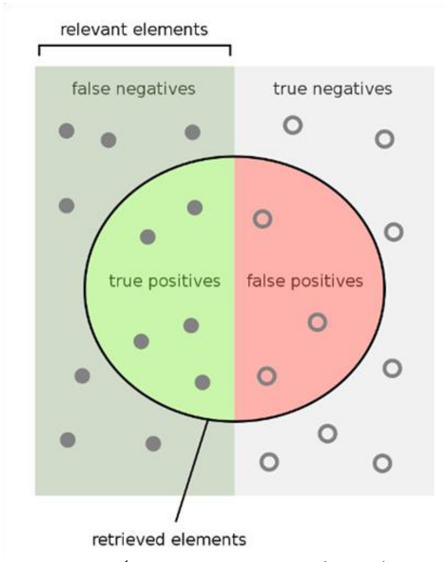
$$\text{recall} = \frac{TP}{TP + FN}$$

Of all observations that were actually 1, what proportion did we predict to be 1? (Also known as sensitivity.)

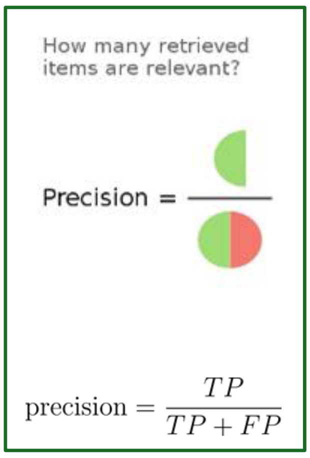
- How **sensitive** is our classifier to **positives**?
- Penalizes false negatives.

One of the Most Valuable Graphics on Wikipedia

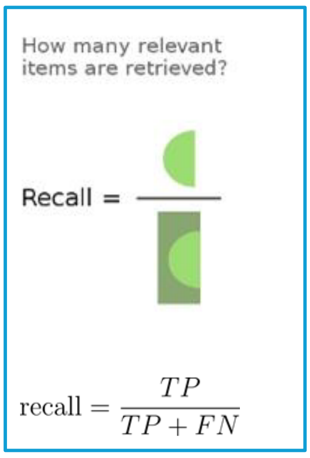
(*i.e., true class is 1)



(i.e., positive; predicted class is 1)



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



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

[adapted from Wikipedia]

Back to the Spam

$$\text{accuracy} = \frac{TP + TN}{n}$$
$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$

Suppose we're trying to build a classifier to filter spam emails.

- Each email is **spam** (1) or **ham** (0).

Let's say we have 100 emails, of which only **5** are truly **spam**, and the remaining **95** are **ham**.

Your friend:

Classify every email as **ham** (0).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

$$\text{precision}_1 = \frac{0}{0+0} = \text{undefined}$$

$$\text{recall}_1 = \frac{0}{0+5} = 0$$

	0	**1**
0	TN: 95	FP: 0
1	FN: 5	TP: 0

Back to the Spam

$$\text{accuracy} = \frac{TP + TN}{n}$$
$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$

Suppose we're trying to build a classifier to filter spam emails.

- Each email is **spam** (1) or **ham** (0).

Let's say we have 100 emails, of which only **5** are truly **spam**, and the remaining **95** are **ham**.

Your friend:

Classify every email as **ham** (0).

$$\text{accuracy}_1 = \frac{95}{100} = 0.95$$

$$\text{precision}_1 = \frac{0}{0+0} = \text{undefined}$$

$$\text{recall}_1 = \frac{0}{0+5} = 0$$

Your other friend ("Friend 2"):

Classify every email as **spam** (1).

$$\text{accuracy}_2 = \frac{5}{100} = 0.05$$

$$\text{precision}_2 = \frac{5}{5+95} = 0.05$$

Many false positives!

$$\text{recall}_2 = \frac{5}{5+0} = 1.0$$

No false negatives!

	0	**1**
0	TN: 0	FP: 95
1	FN: 0	TP: 5

Precision vs. Recall

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Precision penalizes false positives, and Recall penalizes false negatives.

This suggests that there is a **tradeoff** between precision and recall; they are often inversely related.

- Ideally, both would be near 100%, but that's unlikely to happen.

Which Performance Metric?

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

Precision penalizes false positives, and Recall penalizes false negatives.

In many settings, there might be a higher "cost" to missing positive or negative cases.

Some examples:
Detecting if someone tests positive (1) or negative (0) for a disease.
Determining if someone should be sentenced to prison (1) or not (0).
Filtering an email as spam (1) or ham (0).

True and False Positive Rates

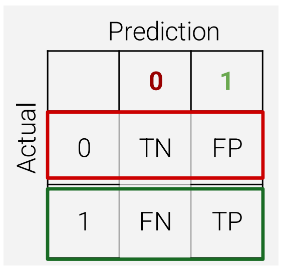
Keeping things interesting – two more performance metrics.

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

False Positive Rate (FPR): out of all datapoints that had Y=0, how many did we classify **incorrectly**?

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

True Positive Rate (TPR): out of all datapoints that had Y=1, how many did we classify correctly? Same as recall.



		Prediction	
		0	**1**
Actual	0	TN	FP
	1	FN	TP

Adjusting the Classification Threshold

Thresholds

We commonly make decision rules by specifying a **threshold**, T. If the predicted probability is greater than T, predict Class 1. Otherwise, predict Class 0.

$$p = P(Y = 1 \,|\, x) = \frac{1}{1 + e^{-x^T\theta}}$$

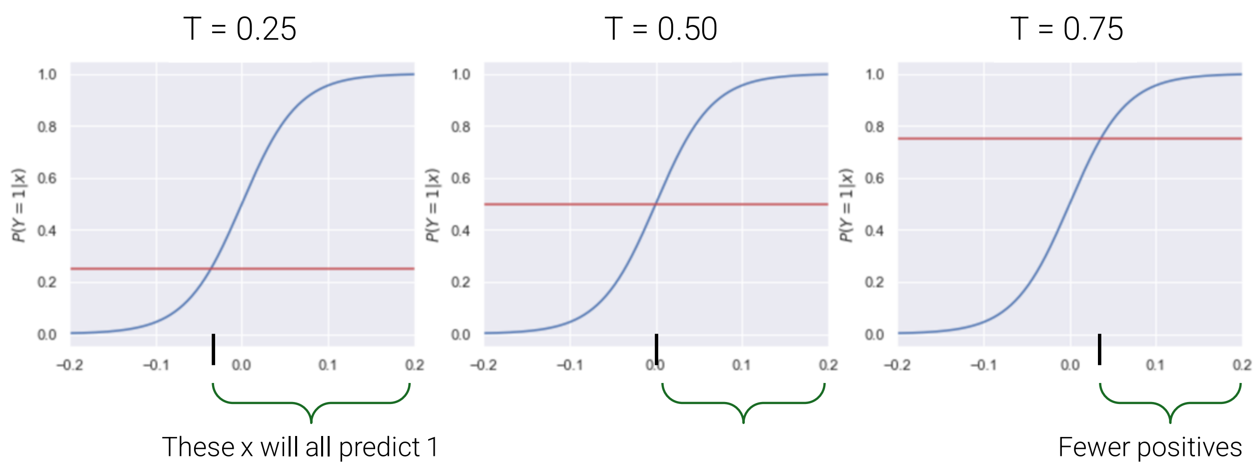
$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class } 1 & p \geq T \\ \text{Class } 0 & p < T \end{cases}$$

What happens if we set T to something other than 0.5?

Changing the Threshold

$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class } 1 & p \geq T \\ \text{Class } 0 & p < T \end{cases}$$

As we increase the threshold T, we “raise the standard” of how confident our classifier needs to be to predict 1 (i.e., “positive”).

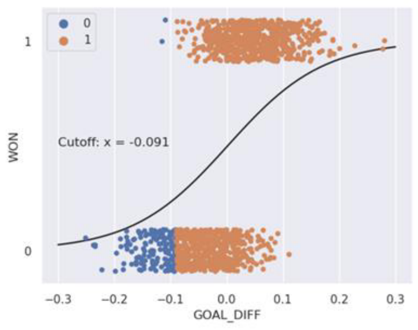


Changing the Threshold

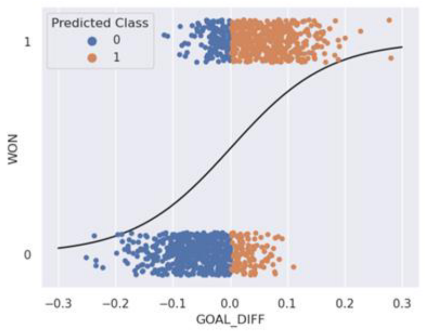
$$\hat{y} = \text{classify}(x) = \begin{cases} \text{Class } 1 & p \geq T \\ \text{Class } 0 & p < T \end{cases}$$

As we increase the threshold T, we “raise the standard” of how confident our classifier needs to be to predict 1 (i.e., “positive”).

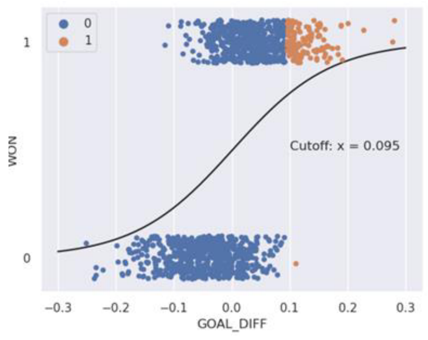
T = 0.25



T = 0.50

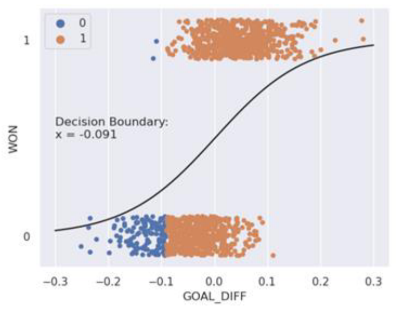


T = 0.75

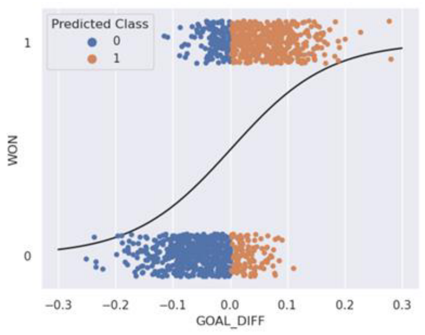


Changing the Threshold

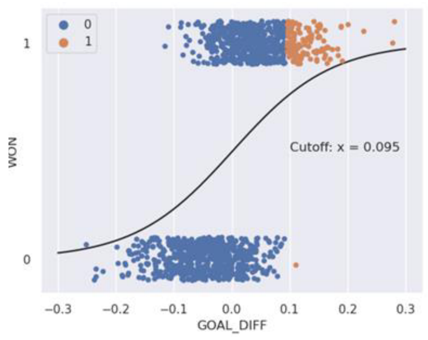
T = 0.25



T = 0.50



T = 0.75



How to interpret a higher classification threshold: the model needs to predict a higher probability of a point belonging to Class 1 before it can confidently classify it as Class 1

- As T increases, we predict fewer positives!

Changing the threshold allows us to finetune how “confident” we want our model to be before making a positive prediction

Precision-Recall Curves

Earlier, we noticed that there is a tradeoff between precision (penalizes false positives) and recall (penalizes false negatives).

We should choose a threshold that keeps both precision and recall high.

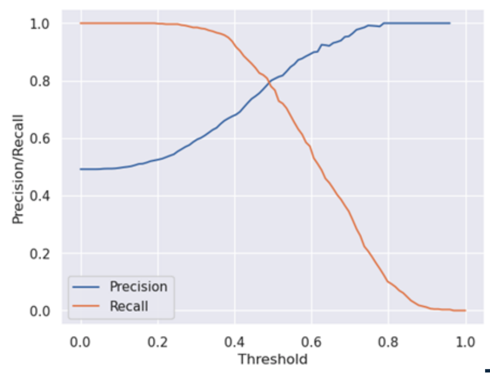
In a precision-recall curve, we:

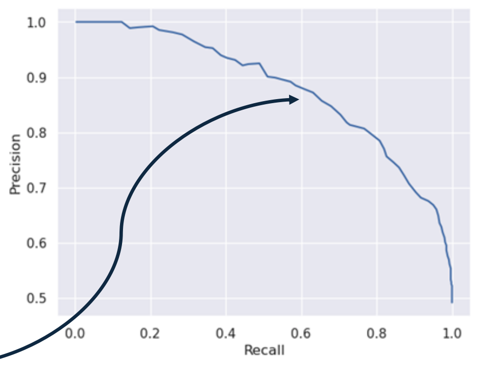
1) Test out many different possible thresholds
2) For each threshold, compute the precision and recall of the classifier
3) Choose the threshold at the "corner" of the curve

Precision-Recall Curves

In a precision-recall curve, we:

1) Test out many different possible thresholds
2) For each threshold, compute the precision and recall of the classifier





We should choose a threshold that keeps both precision and recall high.

F measure

- Combines precision and recall using harmonic mean, the traditional F-measure

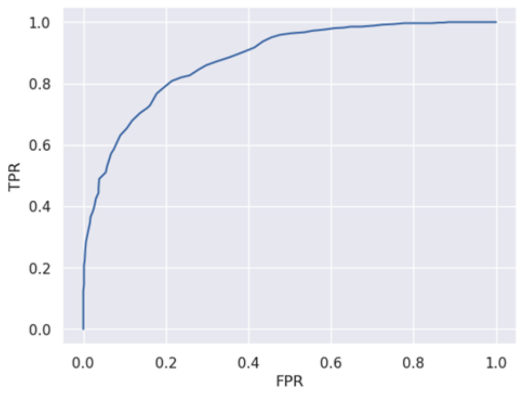
$$F = 2.\frac{precision.recall}{precision + recall}$$

ROC Curves

We can perform a similar process with FPR and TPR

1) Try many thresholds
2) Compute the FPR and TPR for each threshold
3) Choose a threshold that keeps FPR *low* and TPR *high*

ROC = "receiver operating characteristic", comes from radar in WWII

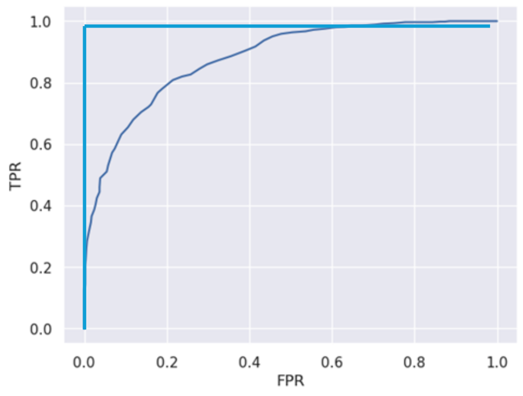


Threshold is low: FPR high, TPR high (many positive predictions)

Threshold is high: FPR low, TPR low (few positive predictions)

ROC Curves

A **perfect predictor** has TPR = 1 and FPR = 0

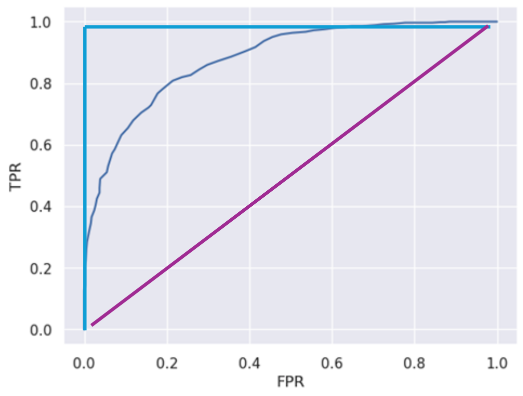


The **Area Under Curve (AUC)** of the perfect predictor is 1.

Because we want our classifier to be as close as possible to the perfect predictor, we aim to maximize the AUC.

ROC Curves

A predictor that **predicts randomly** has an AUC of 0.5


TPR
FPR