NOVA SCHOOL OF BUSINESS & ECONOMICS

3a *Video Lecture*

LOGISTIC REGRESSION

Log odds

- **Odds of an event** : $Odds = \frac{\text{Probability of the event happening}}{\text{Probability of the event not happening}} = \frac{p}{1-p}$

- Logarithm of the odds is called **log-odds** and takes values in the range from $-\infty$ to $+\infty$:

$$\text{log-odds} = \log\left(\frac{p}{1-p}\right)$$

Probability	Odds	Log-odds
0.5	50:50 or 1	0
0.9	90:10 or 9	2.19
0.999	999:1 or 999	6.9
0.01	1:99 or 0.0101	–4.6
0.001	1:999 or 0.001001	–6.9

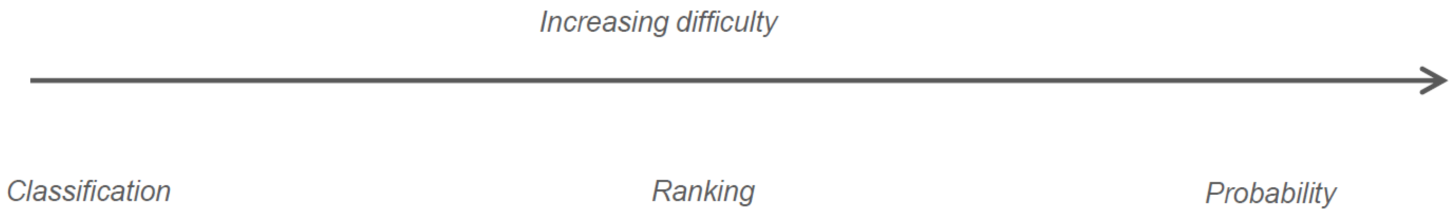
- We can **model log-odds with the same linear function** that we have seen before:

$$\text{log-odds} = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n$$

This model is called **logistic regression**.

Ranking Instances and Probability Class Estimation

- **Class membership probability**
  - *Which clients are most likely to respond to this offer?*
  - *Which clients are most likely to leave when their contracts expire?*
- In other cases, we do not need probabilities, only a **score that will rank** cases by the likelihood of belonging to one class or the other
  - *For targeted marketing we may have a limited budget for targeting prospective customers. We would like to have a list of consumers ranked by their predicted likelihood of responding positively to our offer.*



Logistic Regression is not a regression algorithm

- For logistic regression, **the model produces a numeric estimate**.
- However, the **values of the target variable in the data are categorical**.
- Linear model is capturing the log-odds of class membership.
- It is a **class probability estimation model** and not a regression model.

Calculating probabilities

- p is the **probability that the instance with feature vector x belongs to class 1**
  (1-p: probability that the instance with feature vector x belongs to class 0)

- log odds: $\log\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1\, x_1 + \cdots + \theta_n\, x_n = \boldsymbol{\theta}^T x$

  product is a scalar (number)

  scalars (numbers)

  vectors

Solving for probability:

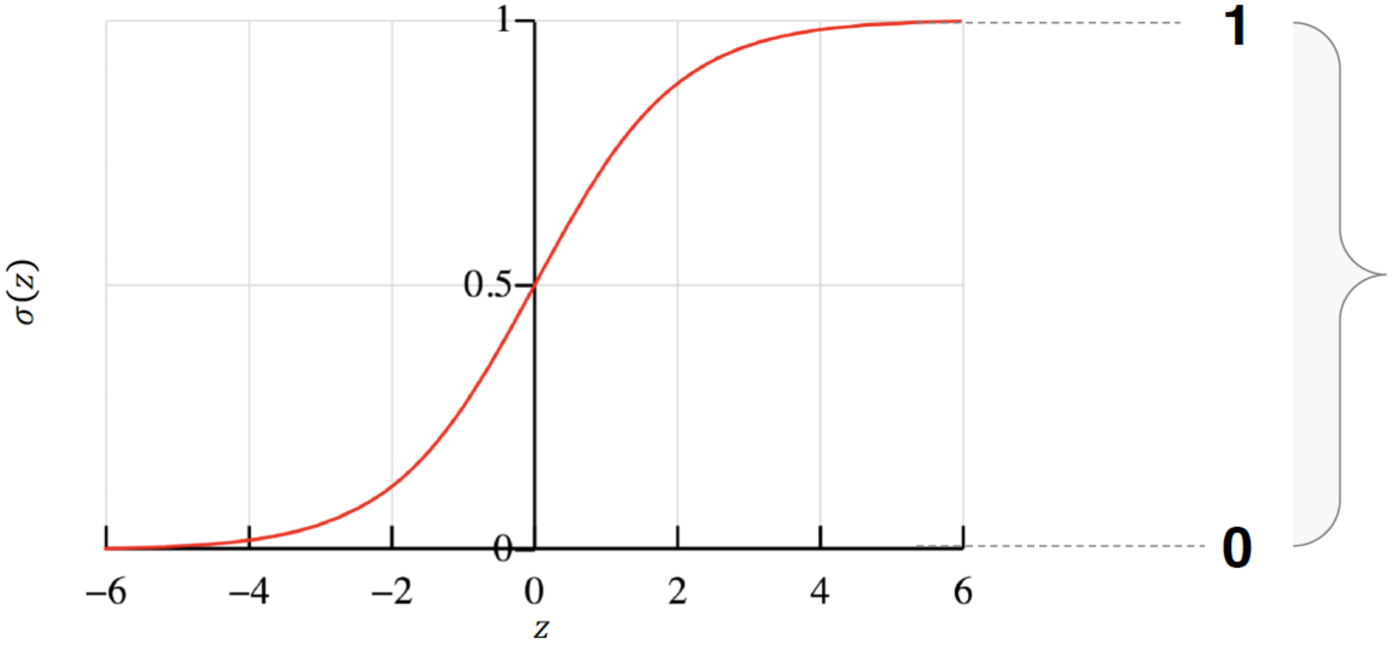
- $p = \frac{1}{1+e^{-(\theta_0+\theta_1\, x_1+\cdots+\theta_n\, x_n)}} = \frac{1}{1+e^{-\boldsymbol{\theta}^T x}}$

Sigmoid function

"Sigmoid" or "logistic" function – it is an S-shaped function that "squashes" the value of $\theta^T x$ into the range [0,1].

$$p = \frac{1}{1+e^{-\theta^T x}}$$

$$S(z) = \frac{1}{1+e^{-z}}$$



$$h_\theta(x) = \sigma(z) = \frac{1}{1+e^{(-z)}}$$

Squash the value z into [0, 1], using sigmoid function

Sigmoid function squashes the value (any value) and gives the value between 0 and 1

- $\sigma(z) \geq 0.5$ *when* $z \geq 0$
- $\sigma(z)$ tends towards 1 as $z \to \infty$

- $\sigma(z) \leq 0.5$ *when* $z \leq 0$
- $\sigma(z)$ tends towards 0 as $z \to -\infty$.

$\sigma(z)$, and hence also $h_\theta(x)$, is always bounded between 0 and 1.

Sigmoid function is also used in neural networks.

How to find the value of θ for fitting the model?

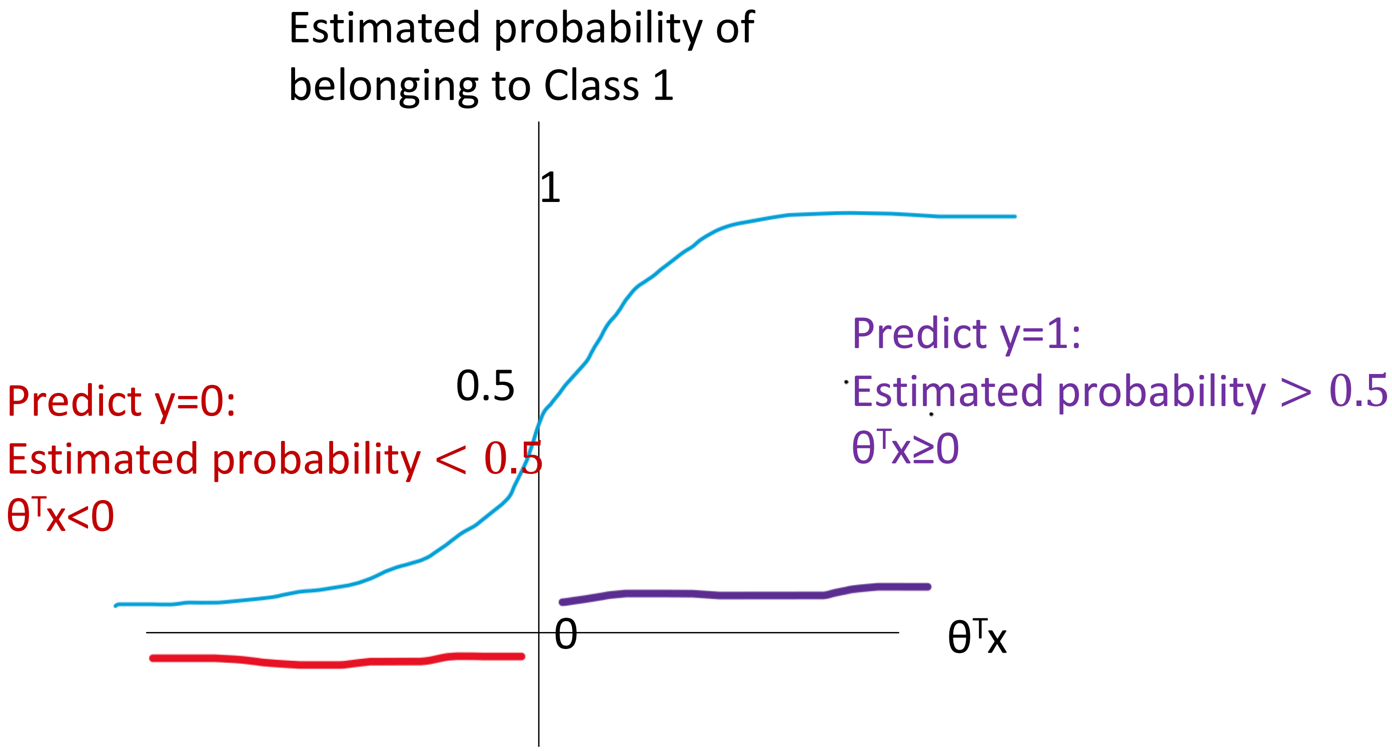
- Logistic regression does not use MSE loss function.
- Our goal is to search for a value of θ so that the model estimates high probability of belonging to class 1 for instances of Class 1 (y = 1) and low for instances of Class 0 (y = 0).
- Log-loss Cost function (cross-entropy):

$$-\frac{1}{m}\sum_{i}^{m}\left[y_i \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right) + (1-y_i)\log\left(1-\frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right)\right]$$

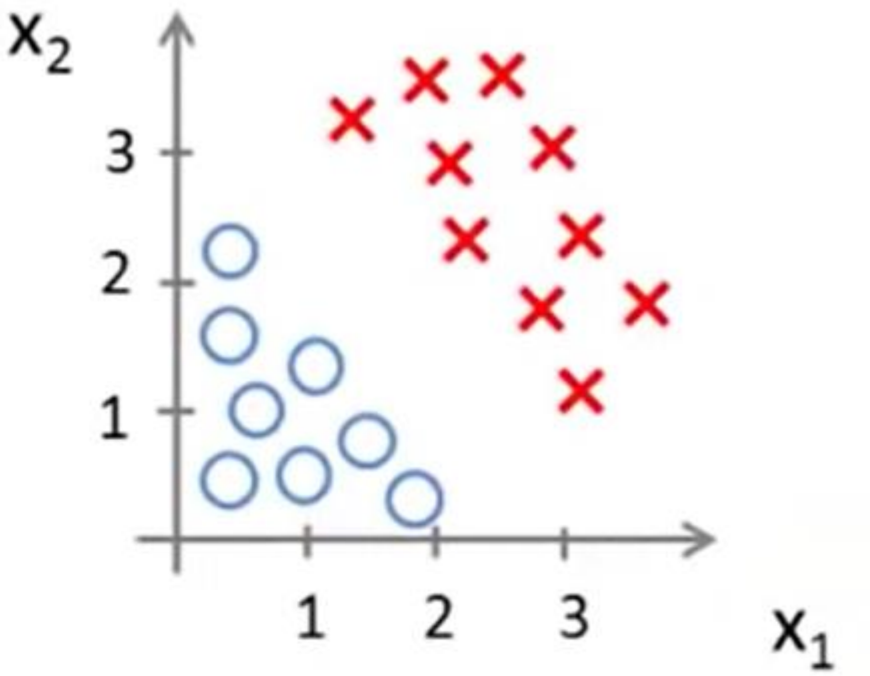
m: number of training data points

Using the logistic regression model for class prediction

Once the Logistic Regression model has estimated the probability $\hat{p}$ that an instance x belongs to the positive class, it can make its prediction ŷ easily. The simplest (but not always the best) approach:



Logistic regression example



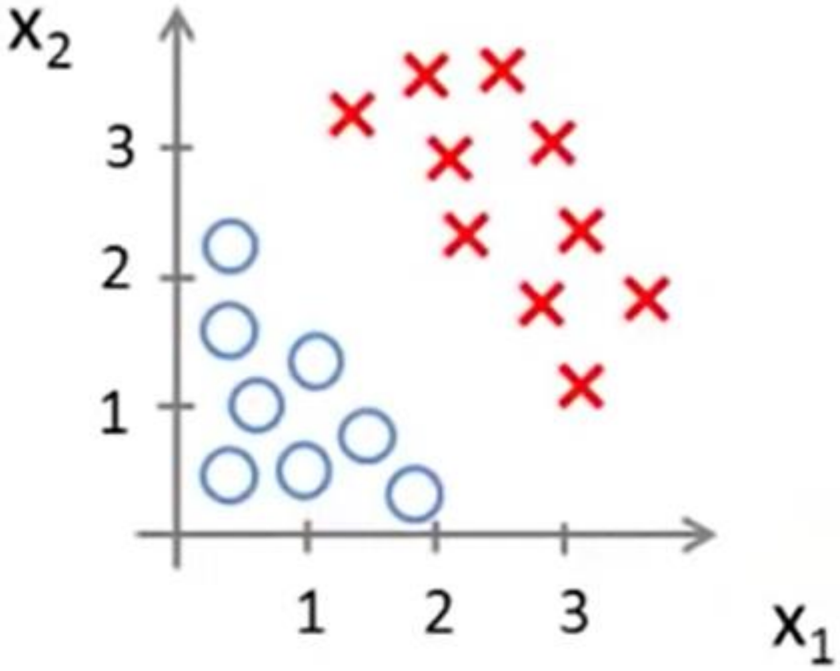
Let's say during training we found that the values of our parameter vector is θ=[-3,1,1].

Then our $\theta^T x$ is as follows: $\theta^T x = -3 + x_1 + x_2$

Then our hypothesis function is as follows:

$$h_\theta(x) = \frac{1}{1 + e^{3 - x_1 - x_2}}$$

Decision boundary



- Let's say during training we found that the values of our parameter vector is θ=[-3,1,1].
- Then our $\theta^T x$ is as follows: $\theta^T x = -3 + x_1 + x_2$
- Then our hypothesis function is as follows:

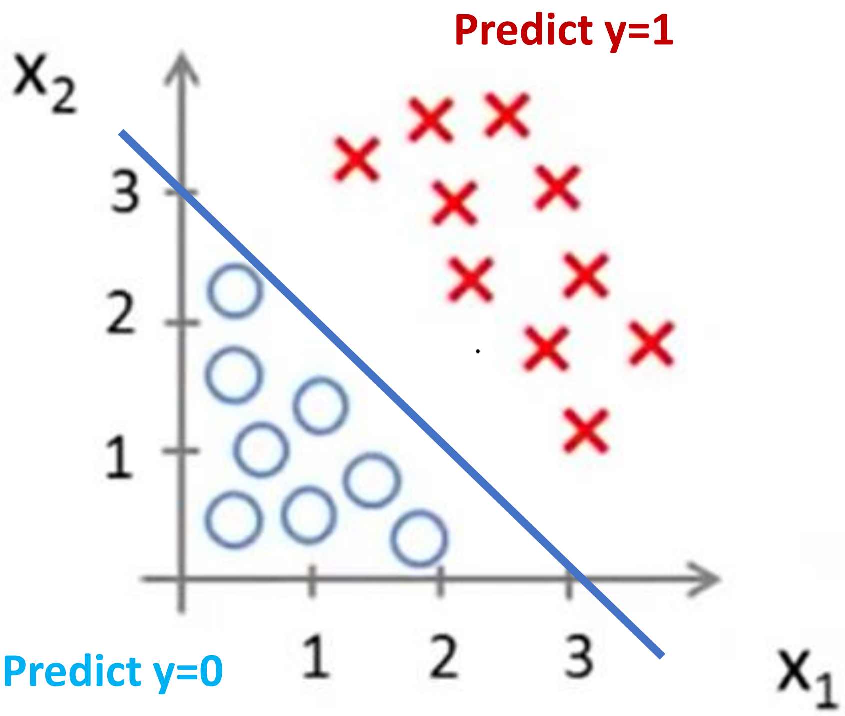
$$h_\theta(x) = \frac{1}{1 + e^{3 - x_1 - x_2}}$$

- We know that: Predict y=1:
  $\theta^T x \geq 0$
- In our case we get that we should predict 1 when:

$$x_1 + x_2 \geq 3$$

What is a decision boundary?

Decision boundary



- Let's say during training we found that the values of our parameter vector is θ=[-3,1,1].
- Then our $\theta^T x$ is as follows: $\theta^T x = -3 + x_1 + x_2$
- Then our hypothesis function is as follows:

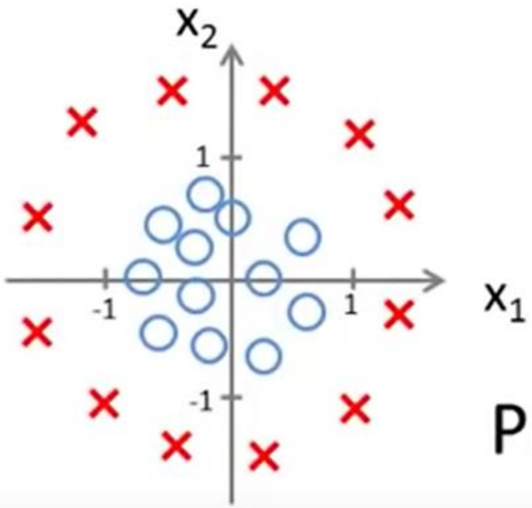
$$h_\theta(x) = \frac{1}{1 + e^{3 - x_1 - x_2}}$$

- We know that: Predict y=1: $\theta^T x \geq 0$
- In our case we get that we should predict 1 when:

$$x_1 + x_2 \geq 3$$

Non-linear decision boundaries

- Just like in the case of linear regression, instad of using only the observed values x, we can use functions of x.



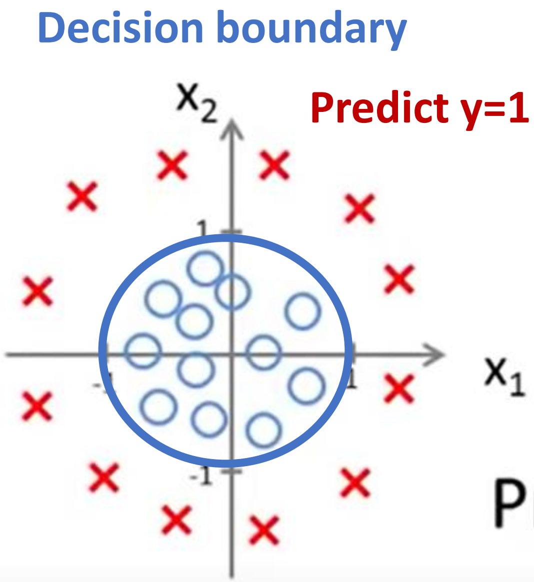
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

Non-linear decision boundaries

- Just like in the case of linear regression, instad of using only the observed values x, we can use functions of x.

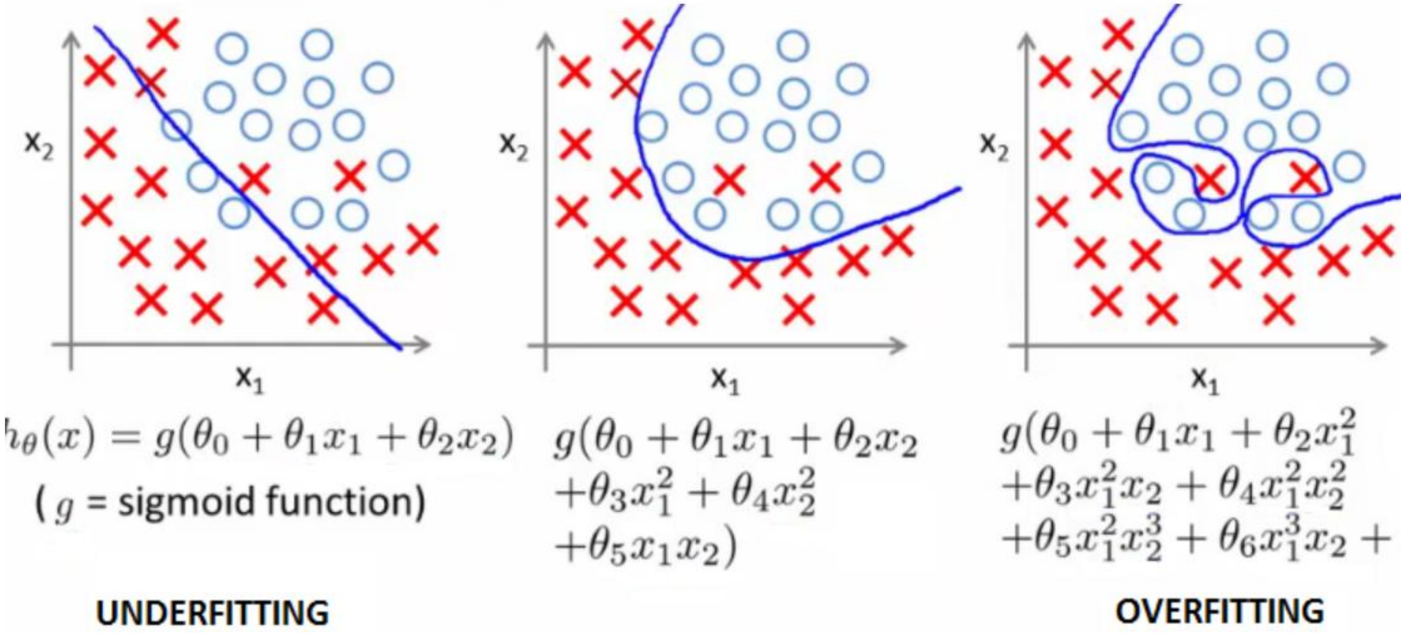


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

Regularization in Logistic Regression



- Again, L1 or L2 regularization
- Penalty term added to the cost function
- Importance of penalty: hyperparameter
- Sklearn by default uses L2 regularization

3b Video Lecture

EVALUATION METRICS
BINARY CLASSIFICATION

Binary classification basics

- Two classes
    - **positive**
    - **negative**
- Percentage of two classes typically uneven: **imbalanced classification**

- Model evaluation
    - measures how will our model generalize to predict the target on new and future data
    - performed on test data (holdout) data by comparing the predicted values with hidden true values

    How do we quantify the performance?

Accuracy

- Simple metric, easy to compute

$$\text{accuracy} = \frac{\text{Number of correct decisions made}}{\text{Total number of decisions made}}$$

- Accuracy=1 – error rate

Accuracy in imbalanced classification

- Dataset: positive class only a tiny portion of the observed data.
  - Only 1% of bank transactions are fraudulent.

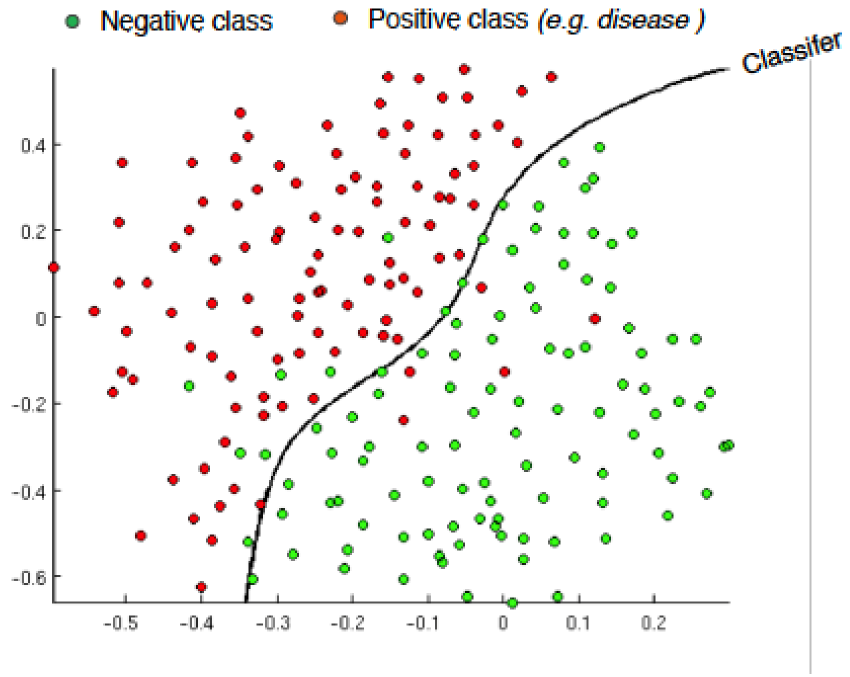
- Model: for every instance predicts negative class: no fraud

- Accuracy

$$\text{accuracy} = \frac{\text{Number of correct decisions made}}{\text{Total number of decisions made}}$$

- 99% of data is negative, accuracy is 0.99!

Confusion matrix



Display the right and wrong predictions in the table

Confusion matrix		Predicted Class	
		Predicted Value : **Positive (+)**	Predicted Value **Negative (-)**
Actual Class	Actual Value : **Positive (+)**	**TP** True Positive	**FN** False Negative
	Actual Value : **Negative (-)**	**FP** False Positive	**TN** True Negative

Binary classification example

In this example, the model correctly classifies many classes but also makes some mistakes:

- some positive classes are wrongly classified as negative values.
- some negative classes are wrongly classified as positive classes

- **True Positive** - we predicted "+" and the true class is "+"
- **True Negative** - we predicted "-" and the true class is "-"
- **False Positive** - we predicted "+" and the true class is "-" (Type I error)
- **False Negative** - we predicted "-" and the true class is "+" (Type II error)

The cost of prediction error

The model can make mistakes

Confusion Matrix		Predicted Class	
		Predicted Value : Positive (+)	Predicted Value Negative (-)
Actual Class	Actual Value : Positive (+)	**TP** True Positive	**FN** False Negative
	Actual Value : Negative (-)	**FP** False Positive	**TN** True Negative

Type II error (FN)

Type I error (FP)

The cost of wrong predictions

It is important to realize there are two types of errors – false positives and false negatives - which often have a different associated cost[1].

It will be great if both False Positive and False Negative are low at the same time. However, it's not ideal in real situation.

Which Is Worse, False Positive or False Negative?
It depends on the business cases.

Medical Diagnosis

Spam Detection

Disease 1, Not disease 0
Type II error more costly

Spam 1, Not spam 0
Type I error more costly

Accuracy, Precision, Recall

Accuracy

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} = \frac{\text{Correct predictions}}{\text{Total data points}}$$

Precision

$$Precision = \frac{TP}{TP+FP} = \frac{\text{Correctly Predicted Positive}}{\text{All Predicted Positive}}$$

Precision : the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) is divided by the total number of elements predicted as belonging to the positive class

Recall

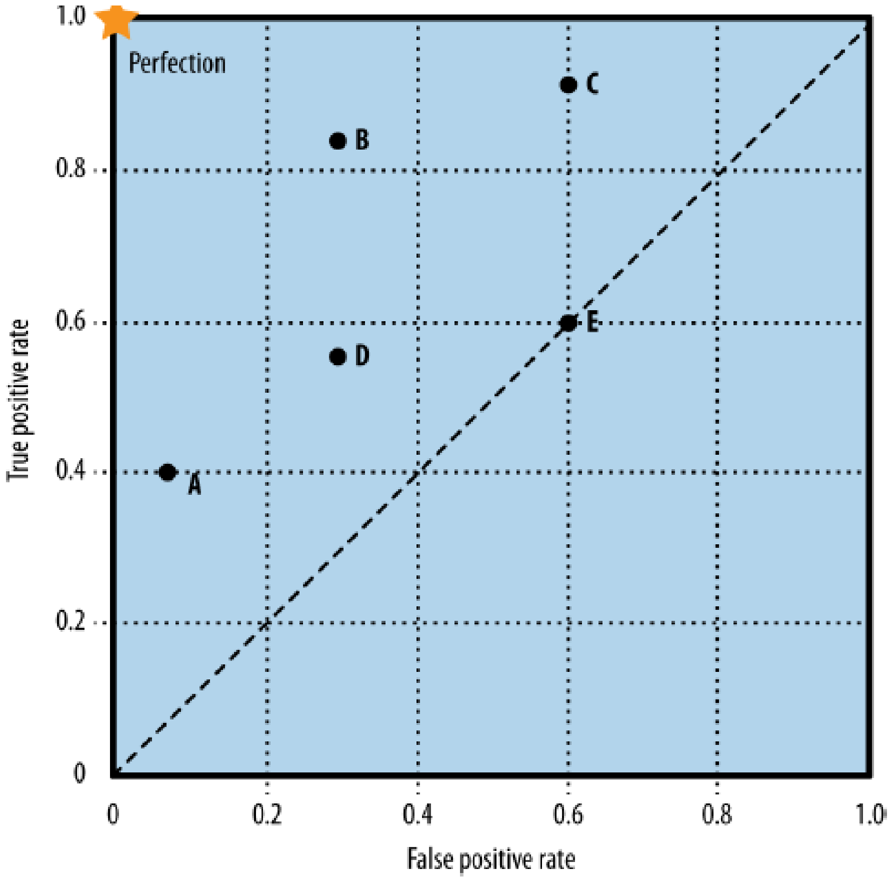
$$Recall = \frac{TP}{TP+FN} = \frac{\text{Correctly Predicted Positive}}{\text{All Real Positive}}$$

Recall : number of true positives is divided by the total number of elements that actually belong to the positive class

Confusion matrix		Predicted Class	
		Predicted Value : **Positive (+)**	Predicted Value **Negative (-)**
Actual Class	Actual Value : **Positive (+)**	**TP** True Positive	**FN** False Negative
	Actual Value : **Negative (-)**	**FP** False Positive	**TN** True Negative

Recall: True positive rate (TPR), Sensitivity

Receiver Operating Characteristics (ROC) plot



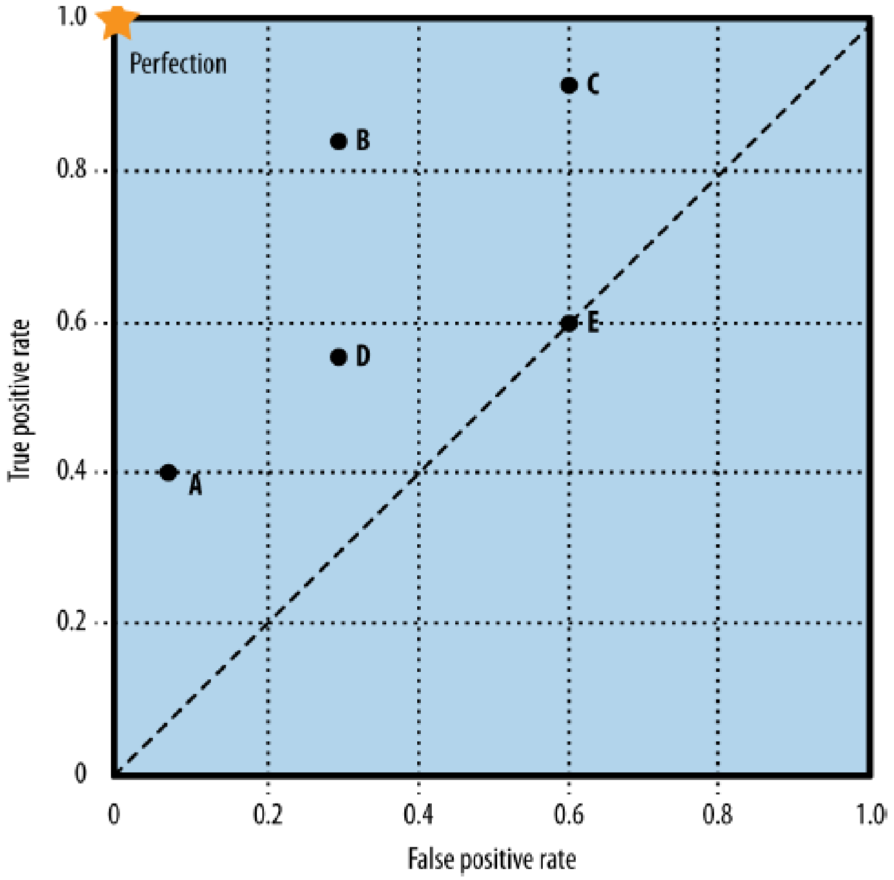
- X axis: **false positive rate**

$$FPR = \frac{\#False\ Positive}{\#False\ Positive + \#True\ Negative}$$

- Y axis: **true positive rate on the y axis**.

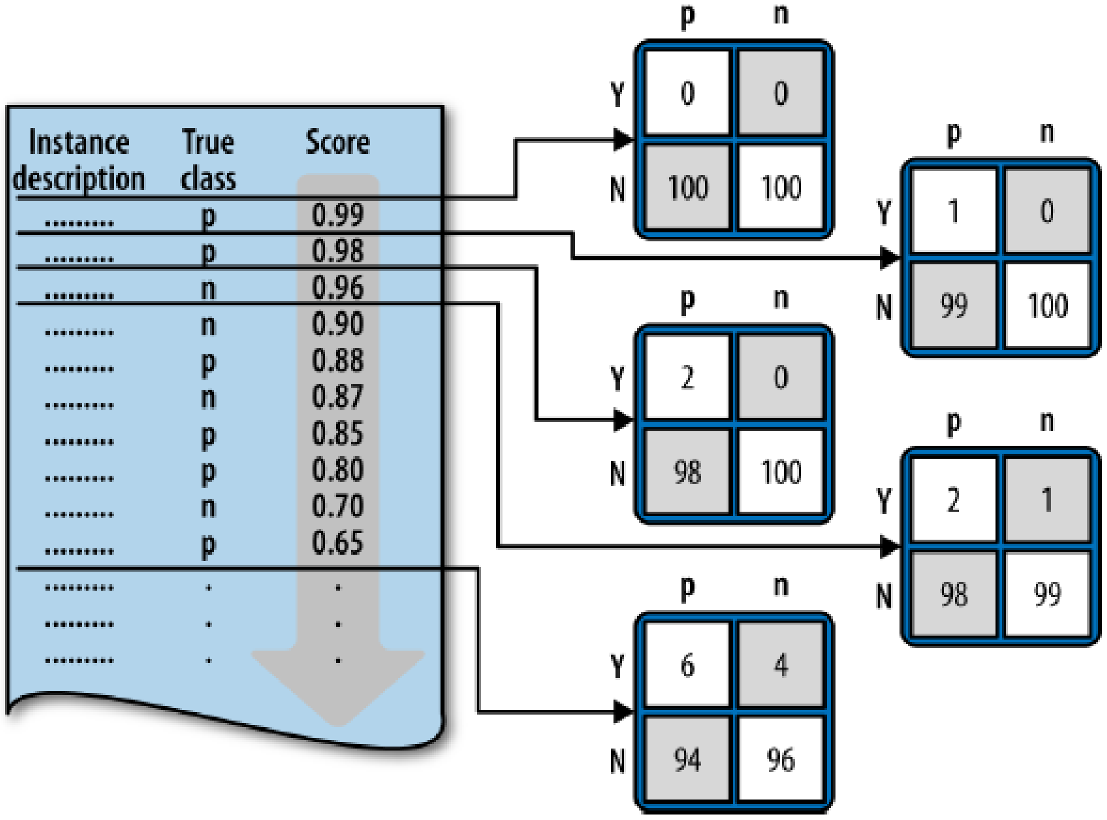
$$TPR = \frac{\#True\ Positive}{\#True\ Positive + \#False\ Negative}$$

ROC plot: Example



Plot shows the performance of 5 discrete classifiers that output only a class label (as opposed to a ranking).

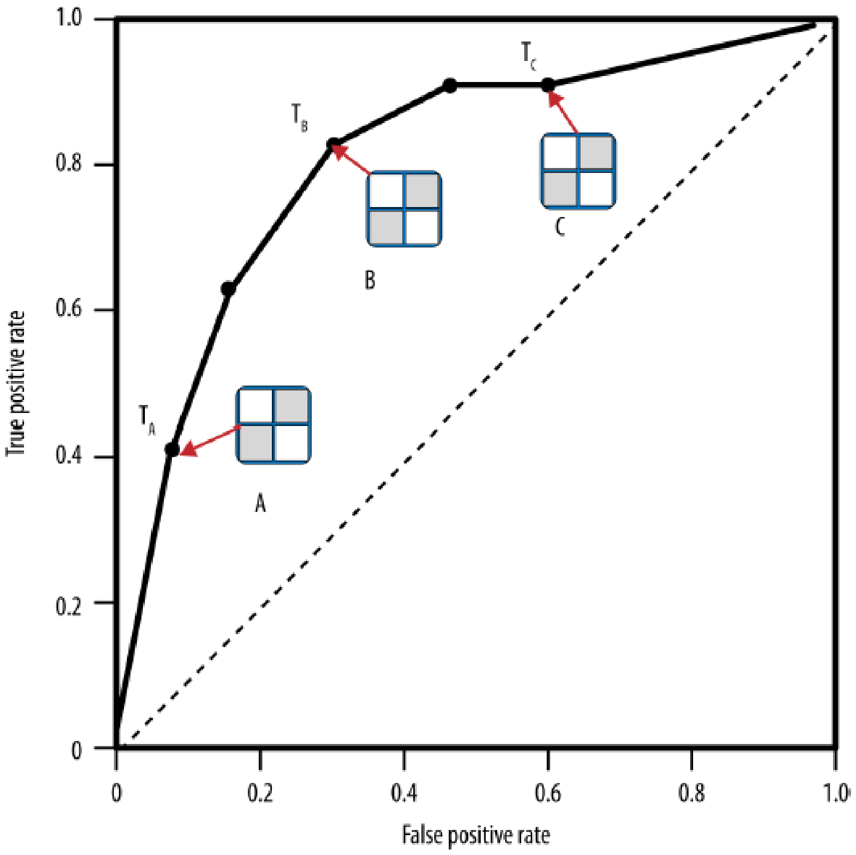
Ranking classifier



Sort instances by decreasing scores:

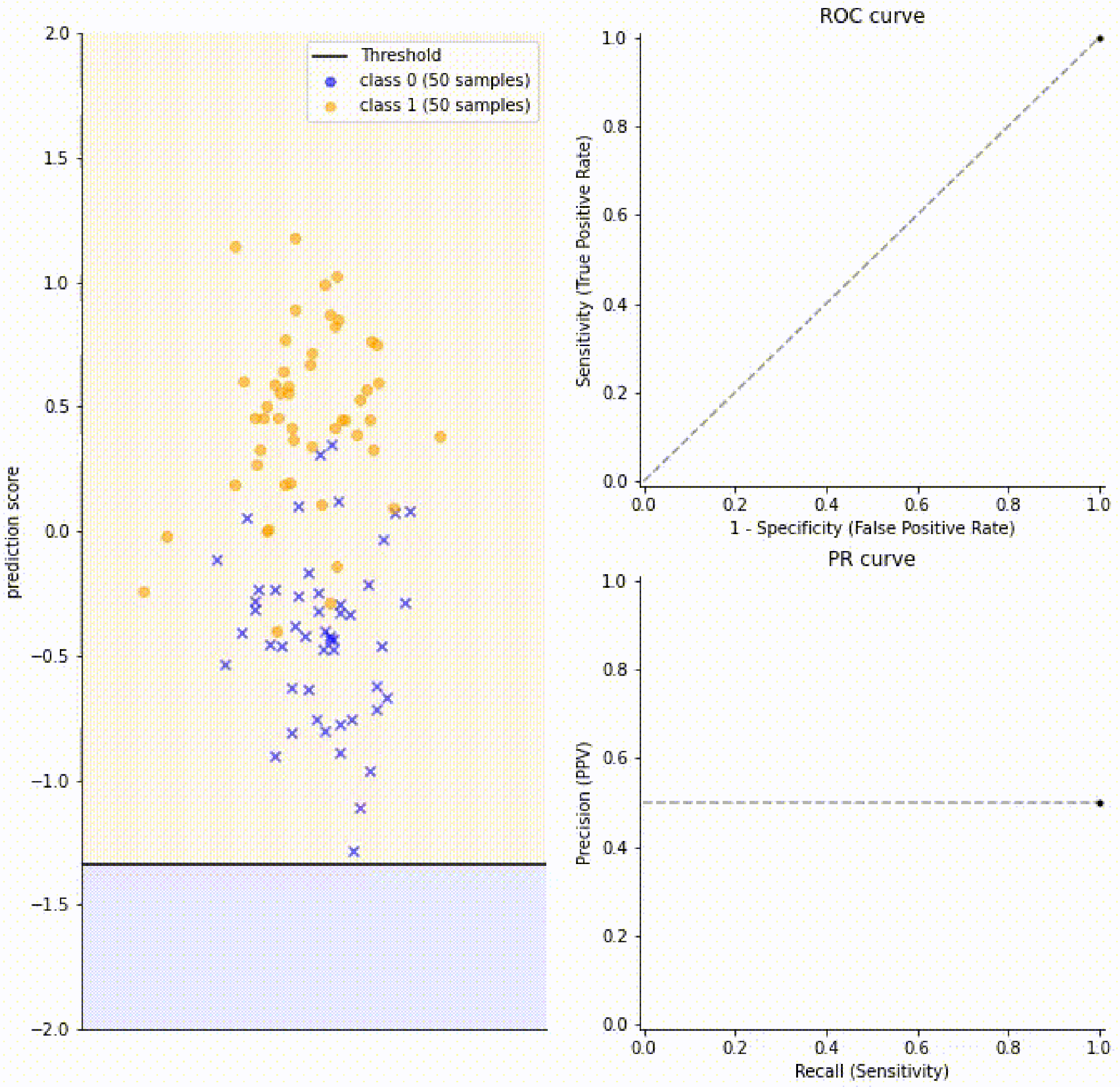
- choose a threshold (represented by a horizontal line)
- classify all instances above the threshold as positive and those below it as negative.
- calculate the confusion matrix
- repeat

A ranking classifier produces a set of points in (ROC) space



Each threshold value produces a different point in ROC space.

Constructing ROC curve



Comparing ROC curves of different classifiers

The closer this curve is to the upper left corner; the better the classifier's performance is.

Main advantage of ROC

- **ROC curves are insensitive to class distribution/ unbalanced datasets**

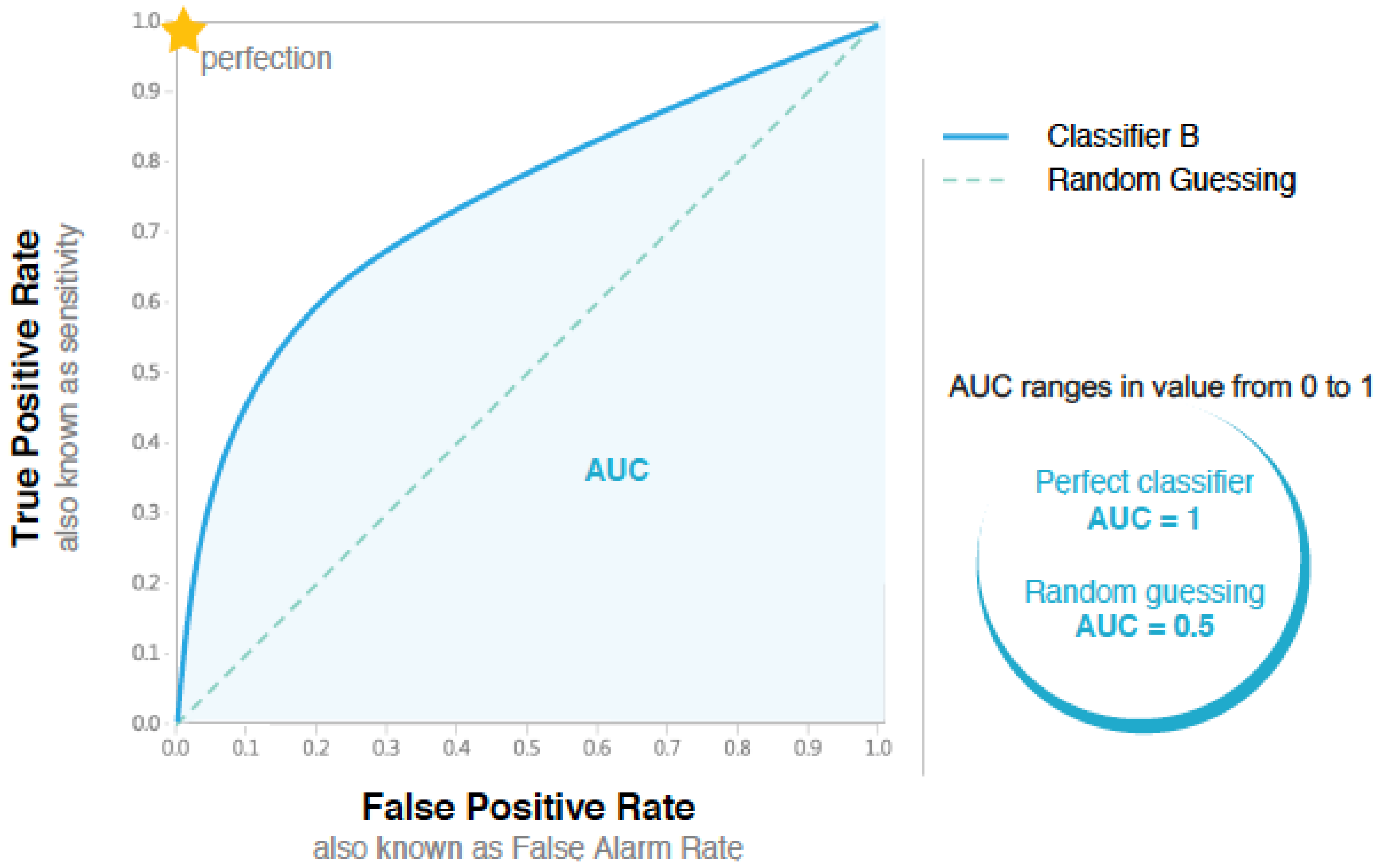
In this example

- Classifier A is better than B
- Classifier B is better than random guessing

AUC ROC (area under the ROC curve)

AUC is (arguably) the best way to summarize model's performance in a single number

- Compare multiple models by a single number
- Useful for model selection
- Higher AUC will be better



Choosing a threshold

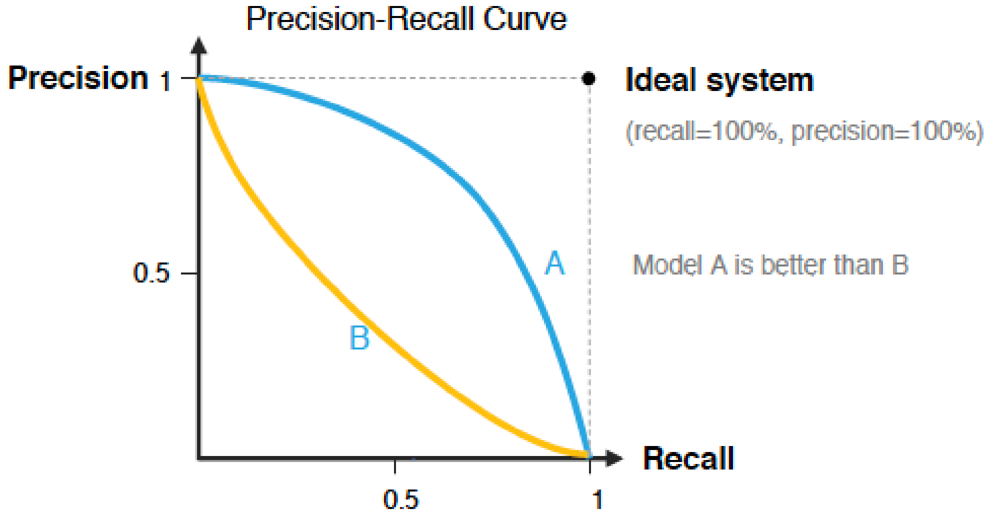
- Depending on the cost of different errors
- Point on the curve that has high true positive rate for low false positive rate

Combining Precision and Recall

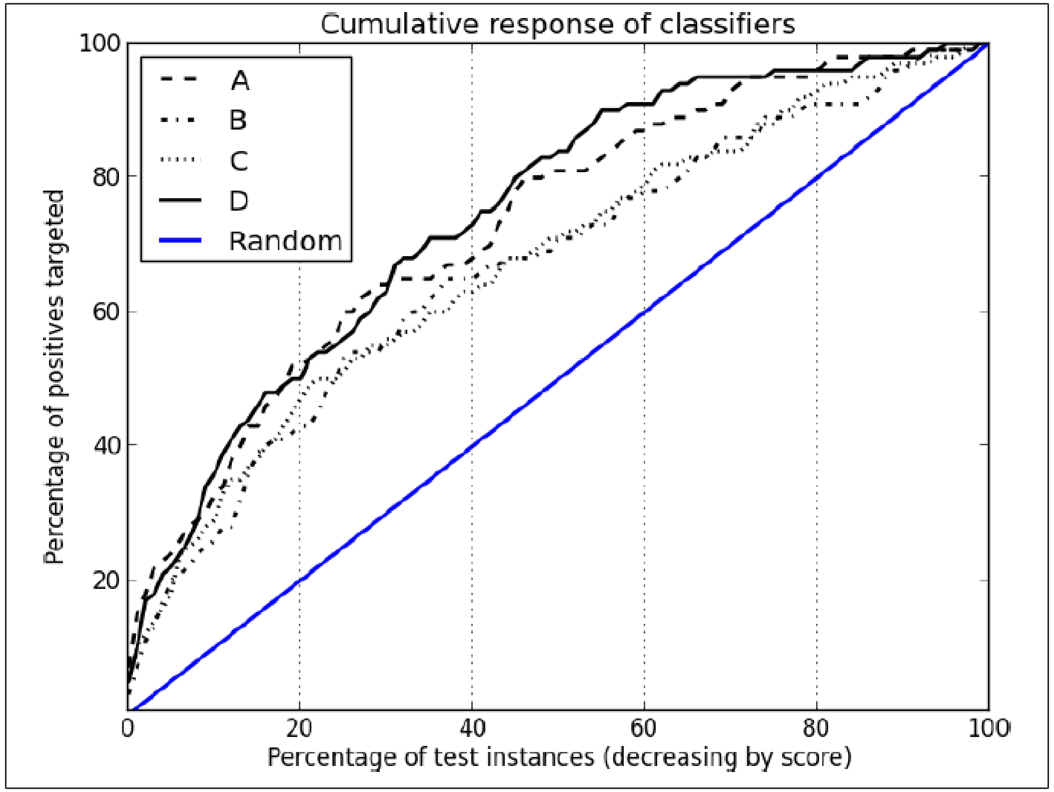
	Precision(P)	Recall(R)	F_1 Score
Algorithm 1	0.5	0.4	0.444 ✓
Algorithm 2	0.7	0.1	0.175
Algorithm 3	0.02	1	0.0392

A combined measure: F score

$$F_1 = 2\frac{PR}{P+R} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

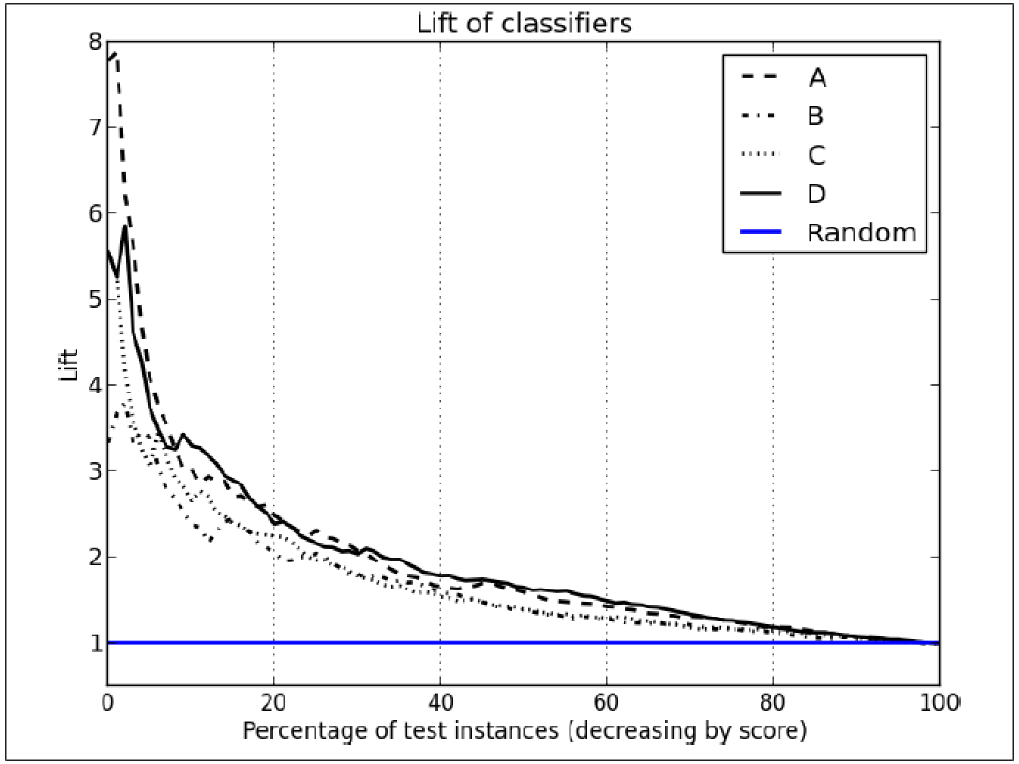


Cumulative response curves



- Y axis: true positive rate
- X axis: percentage of the population that is targeted
- Diagonal line x=y : random performance
- Test data should have the same class distribution as the true population (unlike ROC) .

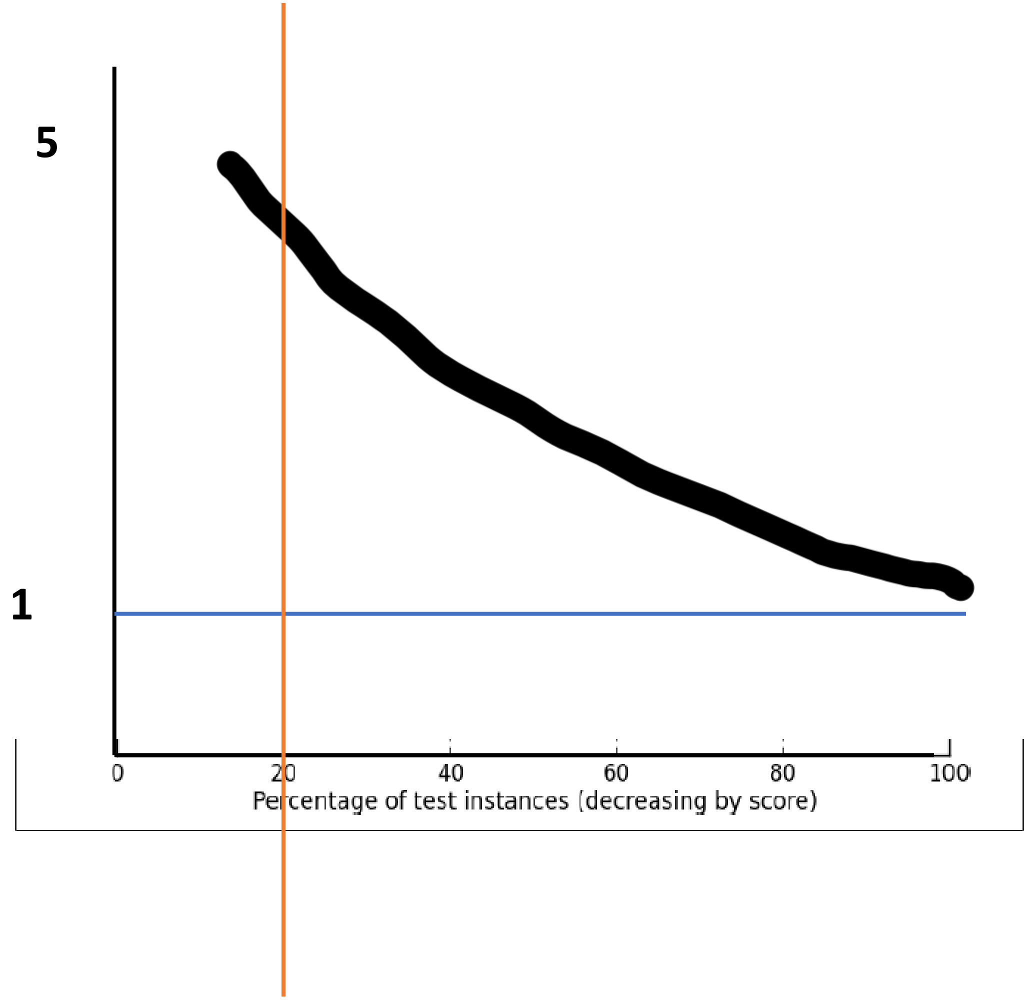
Lift curves



Again, test data should have class distribution representative of the true population (unlike ROC) .

- Value of the cumulative response curve at a given x point divided by the diagonal line (y=x) value at that point.
- y=1: random guessing

Lift curve: example



Churn detection

- churn: positive class
- no churn: negative class

- 100 customers, 20% churners
- If we scan down the list and stop at 20% population:
  - Random guessing: lift =0.2/0.2 = 1
  - Perfect classifier: lift = 1/0.2 = 5

3c Video Lecture

EVALUATION METRICS MULTICLASS CLASSIFICATION

Confusion matrix for Multiclass Classification

	Predicted class1	Predicted class2	Predicted class3	Total
Actual class1	103	7	110	220
Actual class2	3	59	62	124
Actual class3	9	12	5	26
Total	115	78	177	370

Combine with:

- Micro average
- Macro average
- Weighted average

Micro Average

	Predicted class1	Predicted class2	Predicted class3	Total
Actual class1	103	7	110	220
Actual class2	3	59	62	124
Actual class3	9	12	5	26
Total	115	78	177	370

1. Aggregate outcomes across all classes
2. Compute metric with aggregated outcomes

Micro Precision

precision=TP/(TP+FP)

precision=(103 + 59 + 5) / 370= 167 / 370=0.451

Class	Precision
1, 2, 3	(103 + 59 + 5) / 370 = 0.451

For multiclass classification, micro average precision equals micro average recall and equals accuracy.

$$PrecisionMicroAvg = \frac{(TP_1+TP_2+...+TP_n)}{(TP_1+TP_2+...+TP_n+FP_1+FP_2+...+FP_n)}$$

Macro Average

	Predicted class1	Predicted class2	Predicted class3	Total
Actual class1	103	7	110	220
Actual class2	3	59	62	124
Actual class3	9	12	5	26
Total	115	78	177	370

1. Compute metric for each class
2. Average resulting metrics across classes

Precision of class 1

precision=TP / (TP+FP)
precision=103 / (103+3+9) = 0.896

Class	Precision
1	103/115=0.896
2	59/78=0.756
3	5/177=0.028

$$PrecisionMacroAvg = \frac{(Prec_1 + Prec_2 + \ldots + Prec_n)}{n}$$

Macro-average precision:
(0.896 + 0.756 + 0.028) / 3 = 0.56

Weighted Average

	Predicted class1	Predicted class2	Predicted class3	Total
Actual class1	103	7	110	220
Actual class2	3	59	62	124
Actual class3	9	12	5	26
Total	115	78	177	370

1. Compute metric for each class
2. Average resulting metrics across classes weighted by the presence of instances

Precision of class 1

precision=TP / (TP+FP)
precision=103 / (103+3+9) = 0.896

Class	Precision
1	103 / 115 = 0.896
2	59 / 78 = 0.756
3	5 / 177 = 0.028

Weighted-average precision:
(0.896 * 220 + 0.756 * 124 + 0.028 * 26) / 370 = 0.788