Model evaluation
DS351

Imbalanced data

Example:

model 1:

y_i	0	0	0	0	0	0	0	0	0	1
$\hat{y}_i$	0	0	0	0	0	0	0	0	0	0

vs

model 2:

y_i	0	0	0	0	0	0	0	0	0	1
$\hat{y}_i$	0	0	0	0	0	0	0	0	1	1

both have 90% accuracy, but which model would you prefer?

Prediction errors

A model can make two types of error:

		Label	
		1	**0**
Prediction	**1**	correct	False positive
	0	False negative	correct

- Type 1: **False Positive** (0 classified as 1)
  Ex: False alarm
- Type 2: **False Negative** (1 classified as 0)
  Ex: Dangerous items passing a security check

Confusion matrix

y_i	0	0	0	0	0	1	1	1	1	1
$\hat{y}_i$	0	0	0	1	0	1	1	0	1	1

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

- **True Positive**: an instance correctly classified as 1
- **True Negative**: an instance correctly classified as 0

True Positive Rate

y_i	0	0	0	0	0	1	1	1	1	1
$\hat{y}_i$	0	0	0	1	0	1	1	0	1	1

Label

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

True Positive Rate (Recall or Sensitivity):

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

i.e. proportion of positives that are correctly classified as positives

True Negative Rate

y_i	0	0	0	0	0	1	1	1	1	1
$\hat{y}_i$	0	0	0	1	0	1	1	0	1	1

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

True Negative Rate (Specificity):

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

i.e. proportion of negatives that are correctly classified as negatives

Precision

y_i	0	0	0	0	0	1	1	1	1	1
$\hat{y}_i$	0	0	0	1	0	1	1	0	1	1

Label

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

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

i.e. proportion of positive predictions that are actually positive

Accuracy

y_i	0	0	0	0	0	1	1	1	1	1
$\hat{y}_i$	0	0	0	1	0	1	1	0	1	1

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

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

i.e. proportion of all instances that are predicted correctly

Example 1

y_i	0	0	0	0	0	0	0	0	0	1
$\hat{y}_i$	0	0	0	0	0	0	0	0	0	0

$$\begin{aligned}
\text{Recall (TPR)} &= \frac{\text{TP}}{\text{TP} + \text{FN}} = \\
\text{TNR} &= \frac{\text{TN}}{\text{TN} + \text{FP}} = \\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} =
\end{aligned}$$

Example 2

y_i	0	0	0	0	0	0	0	0	0	1
$\hat{y}_i$	0	0	0	0	0	0	0	0	1	1

$$
\begin{aligned}
\text{Recall (TPR)} &= \frac{\text{TP}}{\text{TP} + \text{FN}} = \\
\text{TNR} &= \frac{\text{TN}}{\text{TN} + \text{FP}} = \\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} =
\end{aligned}
$$

What to use?

- Use **Recall** if we want the model to "see" all the positive instances.
  Examples: security check, tests for deadly diseases

What to use?

- Use **Recall** if we want the model to “see” all the positive instances.
  Examples: security check, tests for deadly diseases
- Use **Precision** if we only care about correct positive predictions.
  Examples: Youtube video recommendation, hiring workers

But in some situation, we might want to find a balance between these two scores.

- want a way to combine both Precision and Recall

Precision & Recall

How about average of the two?

y_i	0	0	0	0	0	1	1	1	1	1
$\hat{y}_i$	1	1	1	1	1	1	1	1	1	1

$$\frac{\text{Recall} + \text{Precision}}{2} = \frac{1 + 0.5}{2} = 0.75$$

...probably too high for such a simple model.

F-score

F-score or **F1-score** is used to find a model that has a nice balance between Precision and Recall

$$F_1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Fact: The value F_1 is always between Precision and Recall

Expected value

However, these scores are hard to interpret in practice...what is the meaning of F-score, really?

In business, it might make more sense to evaluate the model in terms of **expected value**

Expected value

Suppose your company want to implement a simple model that presents advertisement to the users:

$$\textbf{Feature} = \textbf{Ads banner}, \quad \textbf{Label} = \begin{cases} 0 \text{ if user does not click the ads} \\ 1 \text{ if user does click the ads} \end{cases}$$

Expected value

Suppose your company want to implement a simple model that presents advertisement to the users:

$$\textbf{Feature} = \textbf{Ads banner}, \quad \textbf{Label} = \begin{cases} 0 \text{ if user does not click the ads} \\ 1 \text{ if user does click the ads} \end{cases}$$

Suppose that the the cost of a single ads is 200, and if a user click the ads, you will gain a profit of 100

Expected value

Left: Model's performance

		Label	
		1	**0**
Prediction	**1**	80	20
	0	40	60

Right: Cost matrix

		Label	
		1	**0**
Prediction	**1**		
	0		

Expected value =

Good evaluation practice

Don't evaluate on training data

- **Training data** is the data that you use to fit the model/learn the parameters

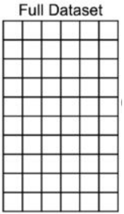
Don't evaluate on training data

- **Training data** is the data that you use to fit the model/learn the parameters
- Bad idea to evaluate the model on the training data, as the model has already "seen" this data

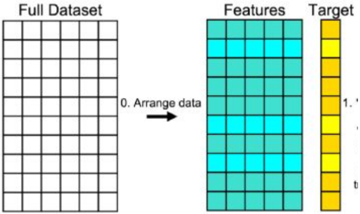
Don't evaluate on training data

- **Training data** is the data that you use to fit the model/learn the parameters
- Bad idea to evaluate the model on the training data, as the model has already "seen" this data
- Instead, the model must be evaluated on unseen/future data
- ...but how can we obtain future data?

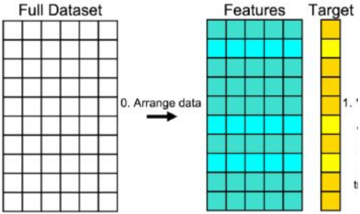
Train-test split



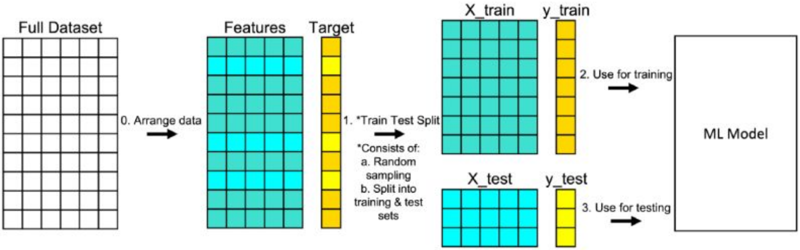
Train-test split



Train-test split



Train-test split



k-fold cross-validation

allows for more variation in the test data

k-fold cross-validation

allows for more variation in the test data

Splitting data in `sklearn`

Train-test split:
`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

k-fold split: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html`