Practical Aspects of Data Science

Data Science Retreat - 2021/B27
Patrick Baier

About me

Patrick Baier

Short Bio:

- Since 2020:: Professor for Machine Learning, Hochschule Karlsruhe
- 2015-2020: Lead Data Scientist at Zalando
- Before: PhD in Computer Science, University Stuttgart

Interests:

- Applied Machine Learning
- Reinforcement Learning

https://www.linkedin.com/in/patrickbaier/

About you

1. What is your name?
2. What did you do in your pre-DSR life?
3. What is your experience in building software?

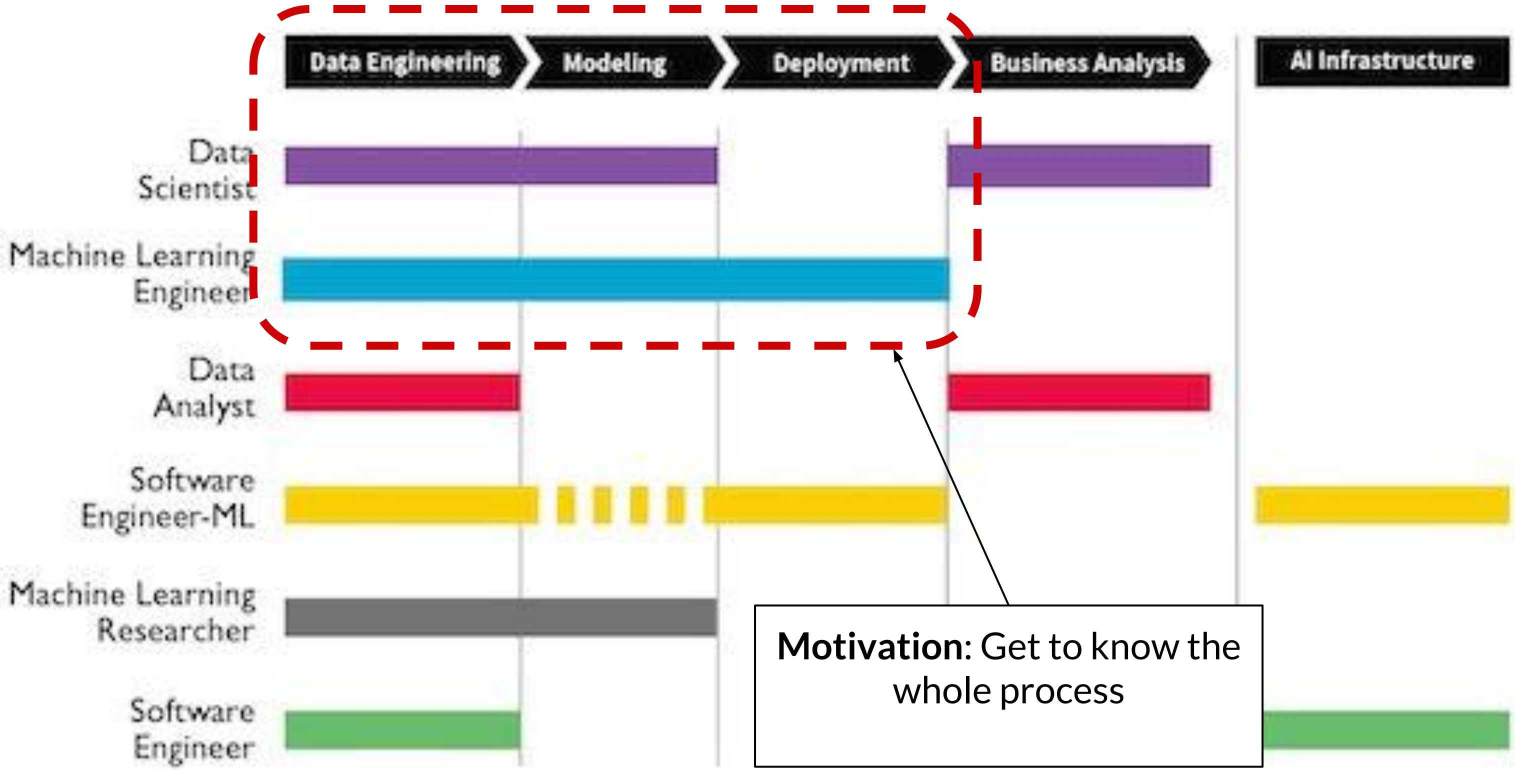
Introduction

Course Goal

The goal of this course is to:

1. Prepare you for data science challenges which are beyond model training.
2. Give you insights into daily work life of a data scientist.
3. Run you through a ML project from model training to production.
4. Prepare you for your job interview.

Motivation



Motivation: Get to know the whole process

Course Format

The course will be a mixture of:

1. Slides.
2. Exercises (coding).
3. Presentations about DS in practice.

Course Overview

- ➔ **Model Learning (Day 1)**
  Model training, classifier evaluation, imbalanced data
- ➔ **Model Operation (Day 2)**
  Probability calibration, Model deployment, missing features, monitoring, DS organization

Time Schedule - Day 1

09:30 - 10:00	Introduction
10:00 - 13:00	Model Training
13:00 - 14:00	Lunch break
14:00 - 16:00	Model Evaluation
16:00 - 18:00	Imbalanced Datasets

Running Example

Running Example

Given:

- Data about customers buying goods at an online book shop
- Label:
    - Class zero: people have not sent back their books
    - Class one: people have sent back their books

Task: Built a binary classification model that predicts in real-time the probability if a customers is going to sent back the ordered items :

- Model must be good at any possible classification threshold/cutoff.
- Model probability should be well calibrated.

Toy data set

Given:

- Data about customers buying goods at an online book shop
- Label:
    - Class zero: people have not sent back their books
    - Class one: people have sent back their books

Format:

- We have log files per day (produced by a web server)
- Every line is one order, represented by a json string

Data Set

Data is given as log files. One file per day.

```
➜  data ls -l return-data
total 8184
-rw-r--r--@ 1 pbaier  113584762  110876 Mar  2 09:59 2017-01-01.txt
-rw-r--r--@ 1 pbaier  113584762  110726 Mar  2 09:59 2017-01-02.txt
-rw-r--r--@ 1 pbaier  113584762  110275 Mar  2 09:59 2017-01-03.txt
-rw-r--r--@ 1 pbaier  113584762  110374 Mar  2 09:59 2017-01-04.txt
-rw-r--r--@ 1 pbaier  113584762  110850 Mar  2 09:59 2017-01-05.txt
```

Data Set

Data is given as log files. One file per day.

```
➜  data ls -l return-data
total 8184
-rw-r--r--@ 1 pbaier  113584762  110876 Mar  2 09:59 2017-01-01.txt
-rw-r--r--@ 1 pbaier  113584762  110726 Mar  2 09:59 2017-01-02.txt
-rw-r--r--@ 1 pbaier  113584762  110275 Mar  2 09:59 2017-01-03.txt
-rw-r--r--@ 1 pbaier  113584762  110374 Mar  2 09:59 2017-01-04.txt
-rw-r--r--@ 1 pbaier  113584762  110850 Mar  2 09:59 2017-01-05.txt
```

filenames

Data Set

Every line of such a file represents one order (in json format):

```
➜  fraud-data head 2017-01-01.txt
{"transactionId": 6707871407, "basket": [1], "zipCode": 2196, "
{"transactionId": 3459351507, "basket": [2, 1, 5, 4, 2], "zipCo
{"transactionId": 7881605492, "basket": [0, 4, 5, 1, 4], "zipCo
{"transactionId": 8168380925, "basket": [3, 4, 2, 2, 0, 4, 3],
{"transactionId": 4691340970, "basket": [2, 4, 5], "zipCode": 3
{"transactionId": 8555449630, "basket": [2, 4, 0], "zipCode": 4
{"transactionId": 5083761599, "basket": [1, 1, 1, 1, 1, 3, 3, 0
{"transactionId": 6396332618, "basket": [3, 3, 5], "zipCode": 3
{"transactionId": 2771228668, "basket": [5], "zipCode": 8607, "
{"transactionId": 3339586925, "basket": [2], "zipCode": 7840, "
```

Data Set

One of these jsons:

```
➜  return-data cat 2017-01-01.txt | head -n 1 | jq .
{
  "transactionId": 6630251676,
  "basket": [
    4,
    1,
    5,
    4
  ],
  "zipCode": 3798,
  "totalAmount": 484,
  "returnLabel": 0
}
```

Data Description

transactionId = running number for orders in the system

basket: Array of item categories that were purchased in this order
→ Example: [4, 1, 5, 4]
= customer bought 2 items of cat. 4 and 1 item of cat. 1 and 1 item of cat. 5

totalAmount = sum of all items prices in the basket in Euro

zipCode = zip code of the customer's address

→ the time dimension does not matter (ignore the date from the file)

Task 1

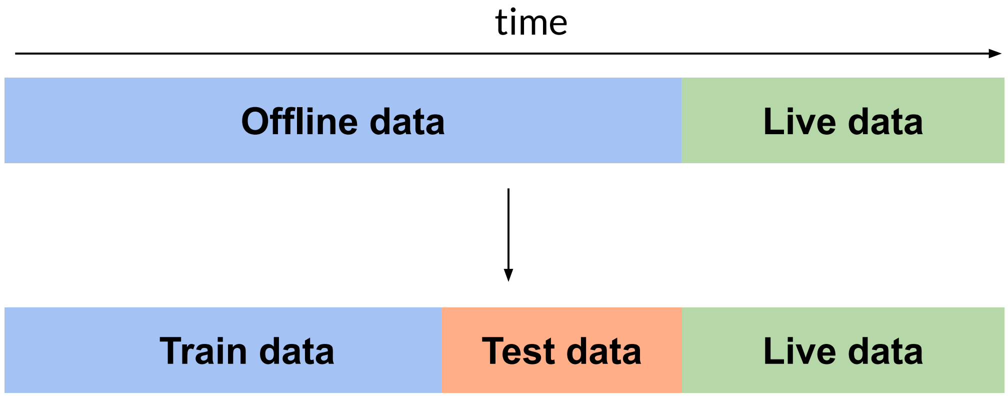
- Download the file `data.zip` from the chat and extract it locally.
- Start a jupyter notebook
- Read in the data as one dataframe (containing all data from all files)
- Train a vanilla* logistic regression:
    - Craft some features (but let's discuss this first once you are ready)
    - Use the `returnLabel` as label
    - Split data into training (70%) and test (30%), see next slides.
    - Learn the classification model
- Do the same for Gradient boosted tree (gbt)
- Compare the two models on the test data and decide for one

* no regularization, no feature scaling

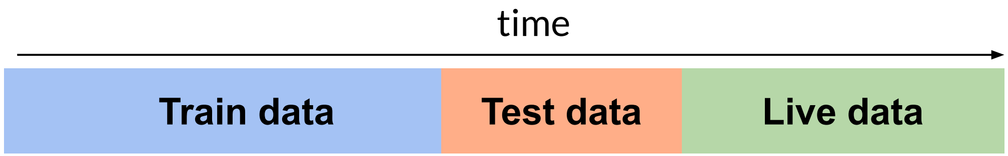
Train-Test-Split

There is one important thing to consider if we are dealing with data that can have some time dependency (e.g. seasonality, shift of distributions):

The test data should be as close as possible to the live data (in terms of time)!

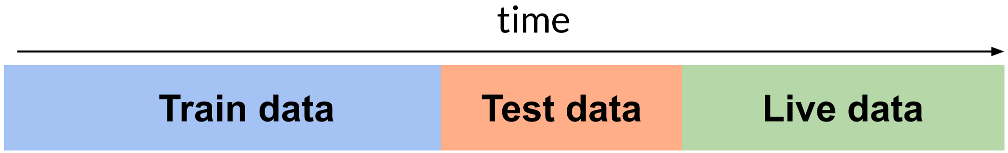


Train-Test-Split



The test date should be as close as possible to the live data. Why?

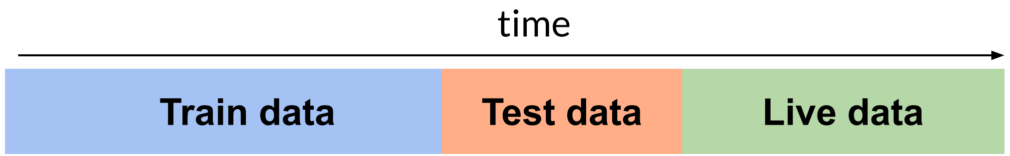
Train-Test-Split



The test date should be as close as possible to the live data. Why?

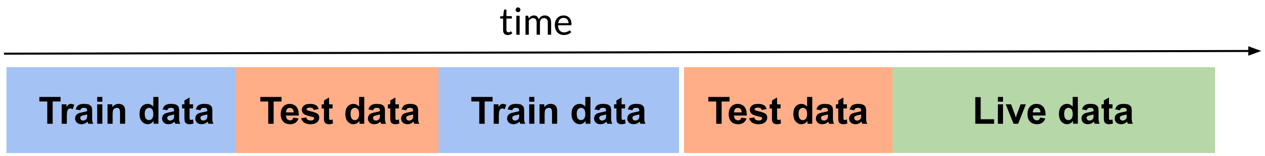
1. Similarity of test data and live data.
2. Avoid data leakage.

Similarity of test and live



- Machine learning models work best if the data on which they predict is similar to the data on which they were trained on.
- In many use cases, data distributions do change over time due to seasonality and trends.
- We evaluate our classifier on the test data but what we really want is the classifier to perform good on the live data.
- Choose test data as similar as possible to the live data.
  Assumption: If model performs good on test, it is likely that it performs also good on live.

Avoid data leakage

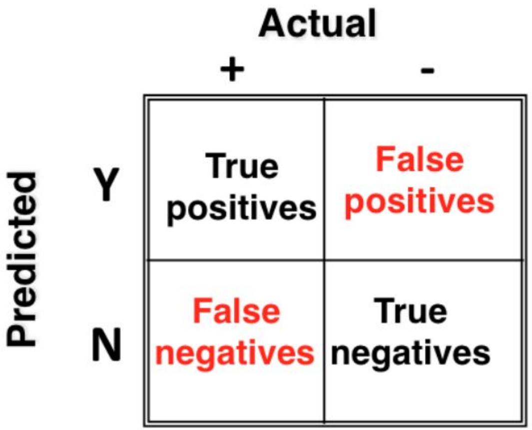


- If the test data is interleaved with the training data (as in the example above), then the classifier might already carry information from the future when predicting on the test data.
- Example:
    - The classifier remembers the book-return-ratio for every customer.
    - If the model predicts on the first test data chunk it may leverage information that is only learned in the second train data junk.
    - This would not be possible in the live data! As a result, we might be overconfident regarding the classifier performs on test data..

Classifier evaluation

Confusion matrix

- In binary classification, we predict a datapoint to be class *zero* or *one*.
- By comparing our prediction against the actual (= ground truth) label we get the confusion matrix:



Accuracy

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

“The fraction of examples classified correctly”

Accuracy

→ Very intuitive and very often used.

<u>But</u>: Very misleading on imbalanced datasets!

<u>Example</u>:
- Given is a dataset in which 1% of data points are positive (class=”1”).
- Also given: a stupid classifiers which always predicts “0”.
- The accuracy of this stupid classifier is 99%!

Precision

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

"Out of those which I classified as positives, how many are correct?"

Recall

Other names: true positive rate, sensitivity

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

"Out of all positives, how many did I found?"

True/False positive rate

$$\text{TPR} = \frac{\text{true positives}}{\text{all positives}}$$

$$\text{FPR} = \frac{\text{false positives}}{\text{all negatives}}$$

Target for Error Types

Sometimes the business is more sensitive towards certain types of errors.

Example: Predictive maintenance (= predict if a component breaks)

"We can tolerate false positives, but we cannot tolerate false negatives".

false positive = "We unnecessarily replace the component."
false negative = "We miss to repair sth and the train crashes."

Good news: We can target for certain error types! (see next slides)

Classifier probability

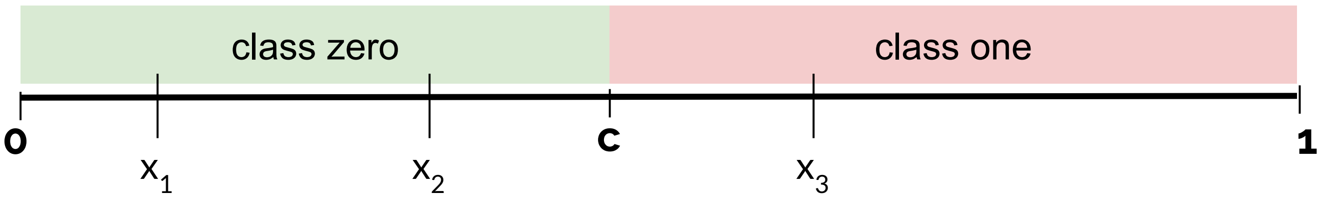
In binary classification a model not only predicts a class but also gives the probability that a data point belongs to class one, i.e. p = 0.7.

`predict` (X)	Predict class labels for samples in X.
`predict_log_proba` (X)	Log of probability estimates.
`predict_proba` (X)	Probability estimates.

Classifier probability

To decide which class we assign the data point to, we need a cutoff threshold c in [0, 1] (as default c is often set to 0.5).

p => c → data point is in class one
$p < c$ → data point is in class zero

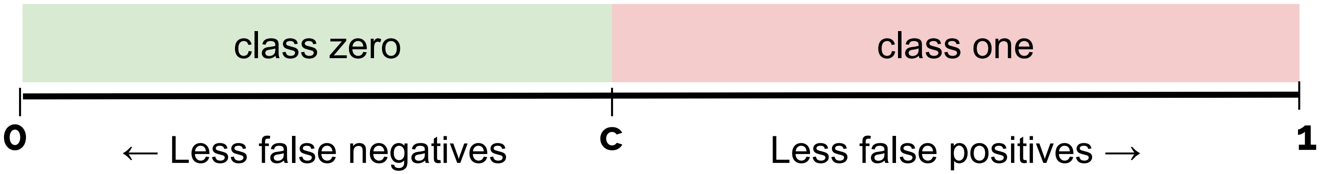


$p(x_1 = 1) = 0.2$ —> class zero
$p(x_2 = 1) = 0.4$ —> class zero

$p(x_3 = 1) = 0.6$ —> class one

Classifier probability

Depending on where the threshold is set, the ratio between false negatives and false positives can vary:



Choose cutoff

Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0

class one

class zero

Cutoff c = 0.5

False positives = 0
False negatives = 1

Choose cutoff

	Prediction	True label
class one	0.9	1
	0.8	1
	0.4	0
	0.2	1
class zero	0.1	0

Cutoff c = 0.2

False positives = 1
False negatives = 0

But: How to we evaluate the performance of a model if cutoff is not know a-priori?

Cut-off selection

- The concrete value of the cut-off is adjusted to business demands.
- Consider using the return prediction model from the bookstore to decide if a customer should get free shipping for their returns or not.
    - Higher threshold: Fewer customers will be “punished”, but probably we will also see many unwanted returns.
    - Lower threshold: More customers get “punished”, but returns will go down.
- The business can adapt the threshold to steer between these extremes.

→ The cut-off can change during the lifetime of a model.
But: How to we evaluate the performance of a model if cutoff is not fixed?

Cut-off selection

The best cut-off in a concrete application scenarios depends on the value of the different outcomes.

For example:

- false-positive: -100€
- true-positive: 100€
- false-negative: -1000€
- True-negative: 1000€

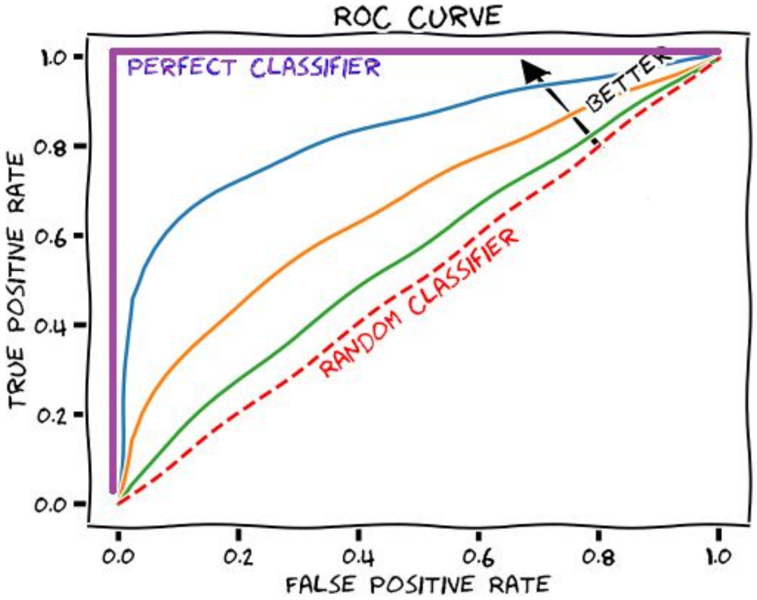
profit = #tp*100 - #fp*100 + tn*1000 - #fn*1000

→ Calculate profit for all possible thresholds

→ Take threshold that results in highest profit

Receiver Operating Characteristic (roc curve)

- Shows for every threshold:
  - True positive rate (tpr):
    True positives / all positives
  - False positive rate (fpr):
    False positives / all negatives
- Worst case: diagonal (= random)
- Best case: upper left corner
- Performance metric: AUC
  (= area under the curve)



https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/

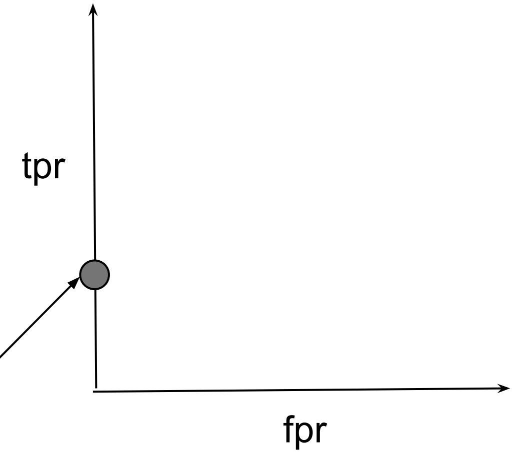
ROC construction

class one

class zero

Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0

c = 0.9



false positives = 0
all positives = 3
tpr = 0.33

true positives = 1
all negatives = 2
fpr = 0

ROC construction

class one

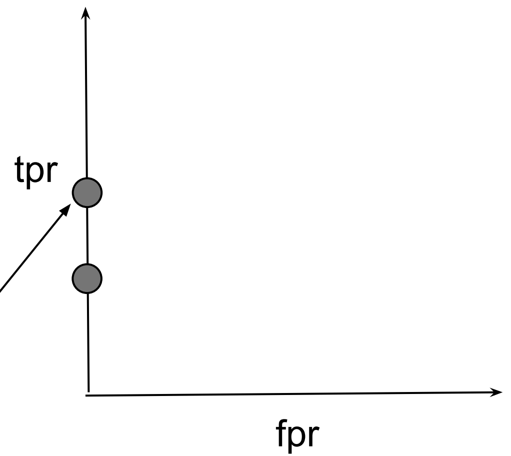
class zero

Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0

c = 0.8

false positives = 0
all positives = 3
tpr = 0.5

true positives = 2
all negatives = 2
fpr = 0



ROC construction

class one

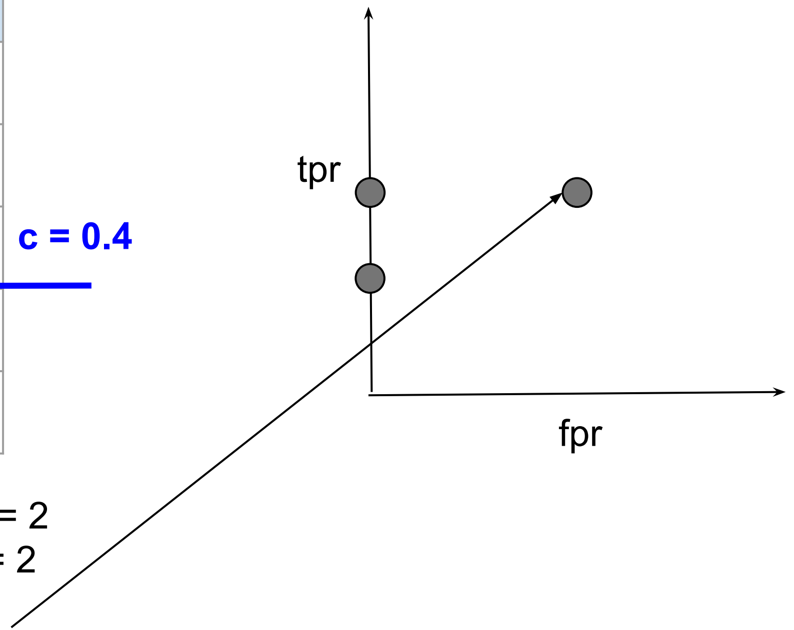
class zero

Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0

c = 0.4

false positives = 1
all positives = 3
tpr = 0.5

true positives = 2
all negatives = 2
fpr = 0.5



ROC construction

class one

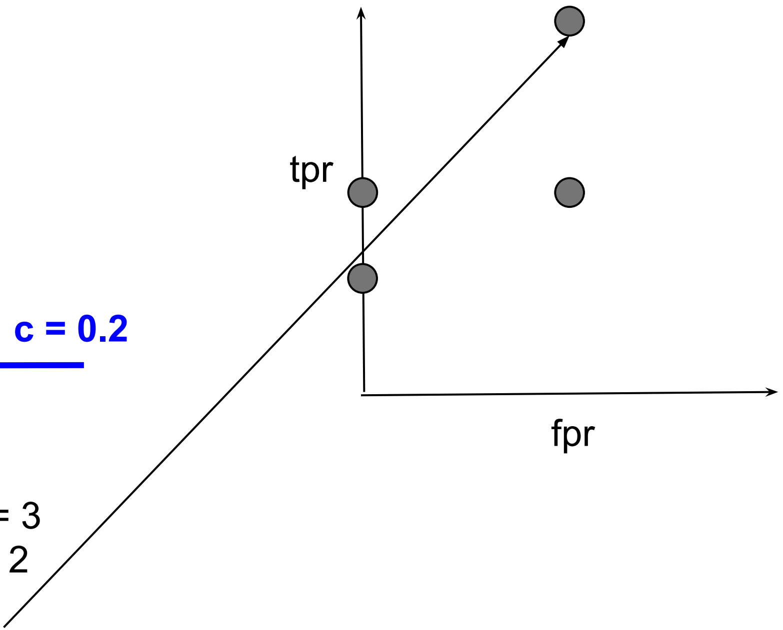
Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0

c = 0.2

class zero

false positives = 1
all positives = 3
tpr = 1

true positives = 3
all negatives = 2
fpr = 0.5



ROC construction

class one

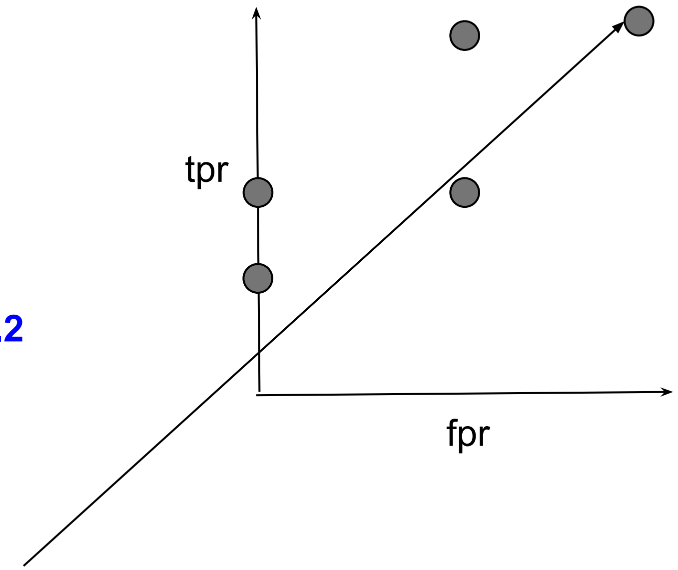
class zero

Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0

c = 0.2

false positives = 2
all positives = 3
tpr = 1

true positives = 3
all negatives = 2
fpr = 1

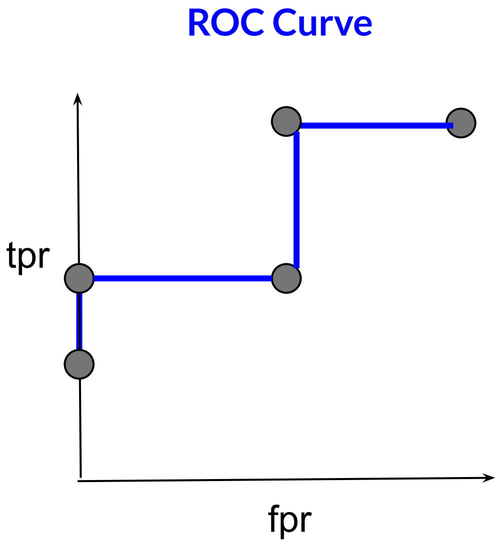


ROC construction

class one

class zero

Prediction	True label
0.9	1
0.8	1
0.4	0
0.2	1
0.1	0



Constructing a roc curve

Given columns:

- prediction (of ML model)
- (true) label

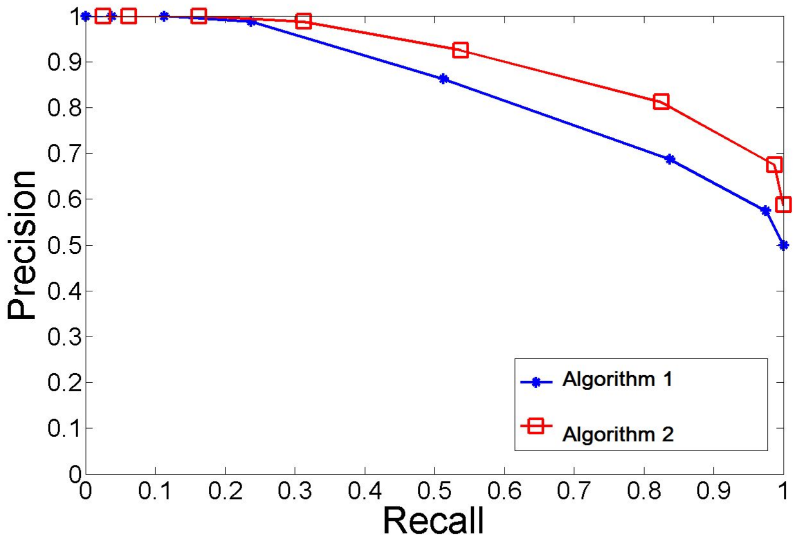
tpr: *True positives / all positives*
fpr: *False positives / all negatives*

Construct roc:

1. Sort prediction column in descending order
2. Start with largest prediction and calculate fpr and tpr if threshold was at this point
3. Plot point in roc plot
4. Do this with every prediction value (going in desc order)

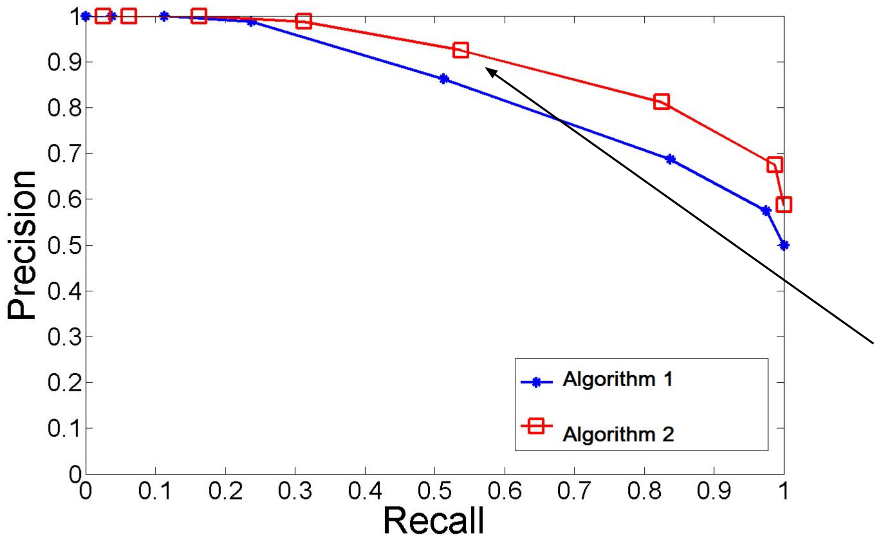
Precision recall curve

PR curves are another often used performance measure for classification.



Precision recall curve

PR curves are another often used performance measure for classification.



At this cut-off, we have a precision of 90% and a recall of 55%.

PRC vs ROC curve

Equivalence Theorem [1]: “A curve dominates in ROC space if and only if it dominates in PR space”.

→ If we compare two algorithms, it is usually sufficient to look at roc curve.

→ “the precision-recall plot changes depending on the ratio of positives and negatives, and it is also more informative than the ROC plot when applied to imbalanced datasets” [2]

[1] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves.
[2] https://classeval.wordpress.com/simulation-analysis/roc-and-precision-recall-with-imbalanced-datasets/

Task 2

- Implement the generation of a roc curve.
- Implement the calculation of auc.
- Use this function to generate the roc curves for the predictions on test data from Task 1.
- Compare them to the roc curves produced by the sklearn library.
- Bonus: Do the same for PR curves.

Data Imbalance

Data Imbalance

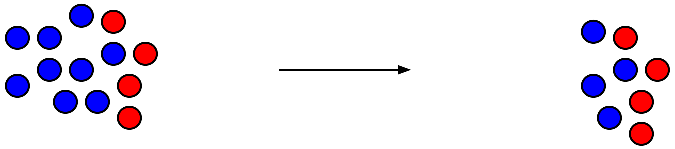
- Positive (or negative) data points are only a small fraction of all data.
  → Return prediction is typically an example of an imbalanced dataset.
  → Others: Credit card fraud, spam classification, machine malfunction, ...

- Imbalanced datasets are a problem for machine learning models [1, 2].
  → Many algorithms are designed for equally balanced classes
  → Default values are often assuming equally balanced classes

- Counter measures: Data sampling, data augmentation, adjust algorithm

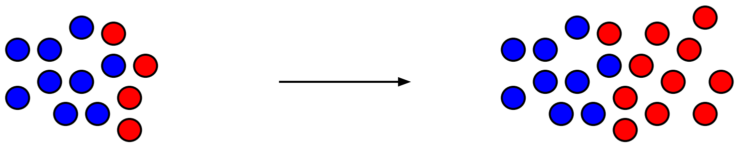
[1] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. 2015. When is Undersampling E ective in Unbalanced Classi cation Tasks?. In Machine Learn- ing and Knowledge Discovery in Databases. Springer International Publishing, 200–215.
[2] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. 2015. Calibrating Probability with Undersampling for Unbalanced Classi cation. In 2015 IEEE Symposium Series on Computational Intelligence. 159–166.

Undersampling



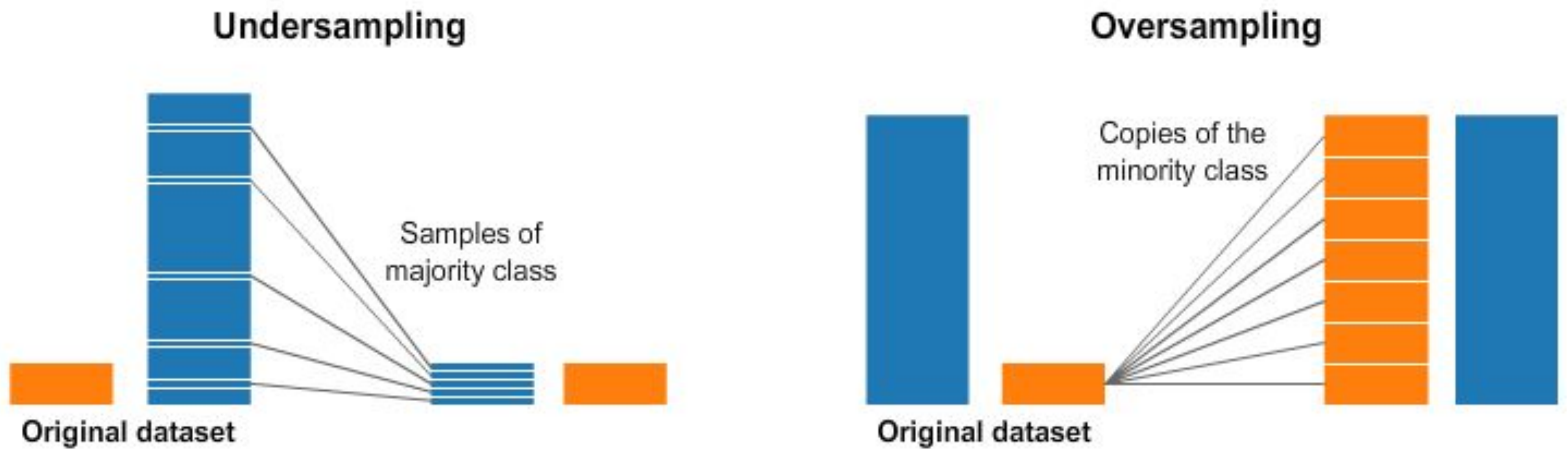
Randomly undersample the majority class
(= randomly remove negative data points)

Oversampling



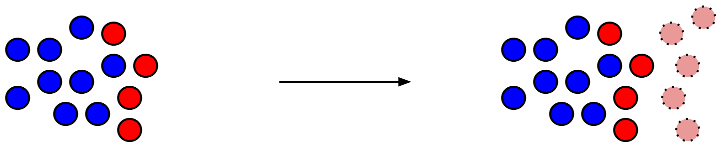
Randomly oversample the minority class with replacement
(= randomly duplicate positive data points)

Under/Over-sampling



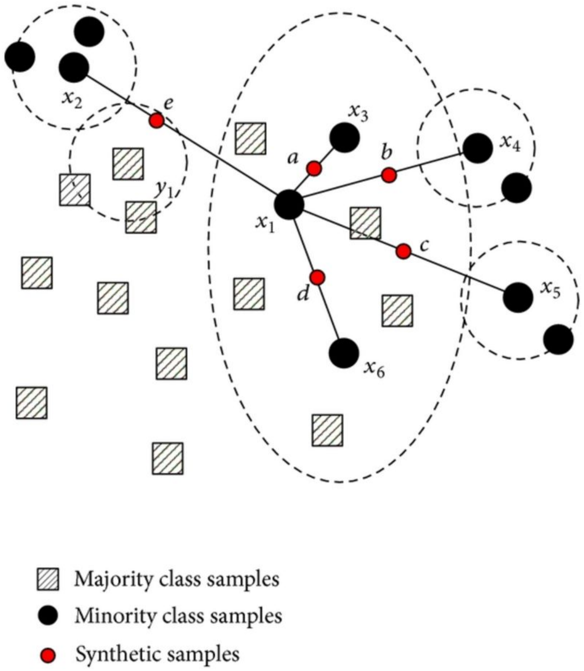
Important: Only do sampling on the train data (and not on the test data).
Remember: Test data has to be representative for live data.

Data Augmentation



Create synthetic data points for the minority class, which are in some sense (e.g. distribution) similar to the minority class
→ SMOTE algorithm (next slide)
→ E.g. image processing: flip, scale and rotate input image

SMOTE



For every point in minority class:

- Find n-nearest neighbors in the minority class
- Draws line between the neighbors an generates random point on the lines.

Source: https://medium.com/coinmonks/smote-and-adasyn-handling-imbalanced-data-set-34f5223e167

Adjust algorithm

Some machine learning algorithm have ways to adjust the “importance” of classes. For instance with Logistic Regression:

```
# Create decision tree classifer object
clf = LogisticRegression(random_state=0, class_weight='balanced')

# Train model
model = clf.fit(X_std, y)
```

This will punish errors more that are made on the minority class.
For details check [this](#) good blog post.

General note

- In general, these techniques <u>can</u> improve your prediction performance (they not necessarily will do that).

- Typically the workflow is to:
  1. First build a simple model without these techniques
  2. Introduce data imbalance techniques when you are in the phase in which you gradually improve your model (and want to squeeze out the last percentage points).

- <u>Important</u>: Apply sampling only on train data (not on test data!)

Task 3

1. Use undersampling for the logistic regression/gbt model of task 1.
2. Use oversampling for the logistic regression/gbt model of task 1.
3. Compare the results to the previous performance in terms of auc.
4. Bonus: Try to implement SMOTE algorithm