Practical Aspects of Data Science

Data Science Retreat - 2019/B20

Patrick Baier

About me

Patrick Baier

Short Bio:
- Data Science Lead at Zalando
- Freelance DS-Trainer/Consultant
- PhD in CS from University Stuttgart

Interests:
- Big Data Processing
- Data Science

Contact:
- https://www.linkedin.com/in/patrickbaier/
- petz2000@gmail.com

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.

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 day wise logs

```
➜  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

Every line of a daily file is 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

basket: Array of item categorizes that were bought in this order
→ [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 items in the basket in euro

transactionId = running number for orders in the system

zipCode = zip code of the customer address

—> customers are unique
—> the time dimension does not matter

Task 1

- Extract the training data, run: `python genData.py`
- Start a jupyter notebook
- Read in the data as one dataframe (containing all data from all files)
- Learn a vanilla* logistic regression:
    - Craft some features (but let's discuss this first once you are ready)
    - Use the `returnLabel` as label
    - Split data randomly (seed = 0) into training (70%) and test (30%)
    - 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

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:

		Actual +	Actual -
Predicted	Y	True positives	False positives
Predicted	N	False negatives	True negatives

Accuracy

The number of examples the classifier classifies correctly:

correct predictions / # all predictions

→Very intuitive and used very often

But: Works very bad on imbalanced datasets!
I.e. if you only have a few return cases, you already have a good accuracy if you always predict not-return.

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

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

Out of all positives, how many did I found?

Other names: true positive rate, sensitivity

True/False positive rate

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

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

Target for Error Types

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

“We can tolerate false positives, but we cannot tolerate false negatives”.

Example: Predictive maintenance (= predict if a component breaks)
false positive = “We unnecessarily replace the component.”
false negative = “The train crashes.”

In classification we can target for certain error types by adjusting the cutoff.

Classifier probability

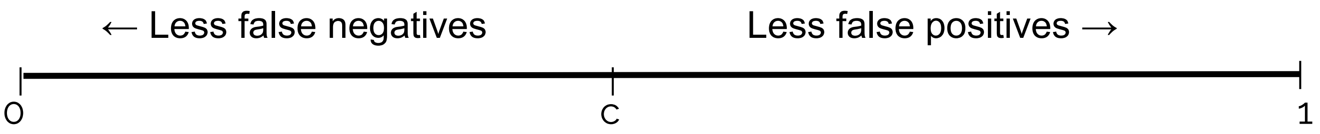
In binary classification a model not only predicts a class but also gives the probability that data point belongs to class one, i.e. p = 0.7.

`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



Choose cutoff

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

Cutoff c = 0.5

False positives = 0
False negatives = 1

Choose cutoff

class one

class zero

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

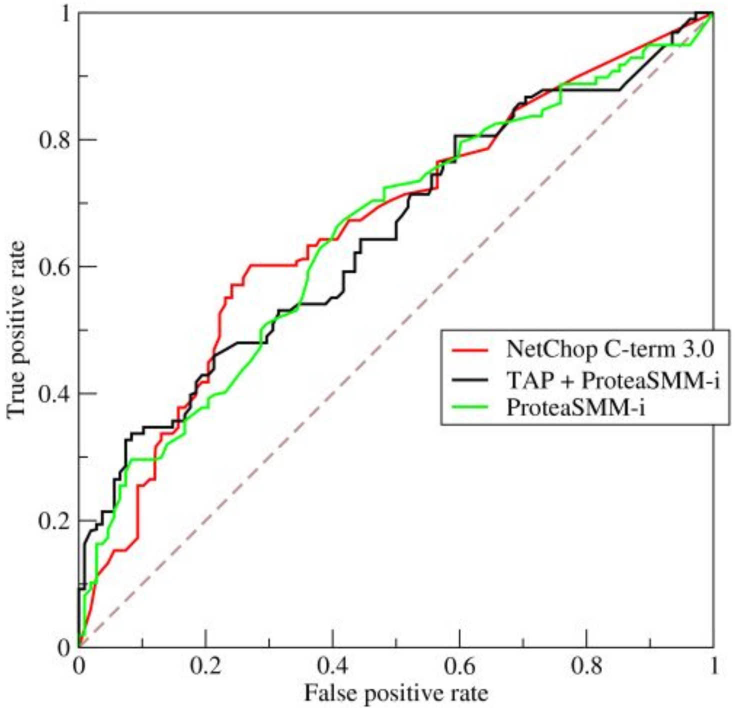
Cutoff c = 0.2

False positives = 1
False negatives = 0

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

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)



Constructing a roc curve

Given columns:
- prediction (of ML model)
- (true) label

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

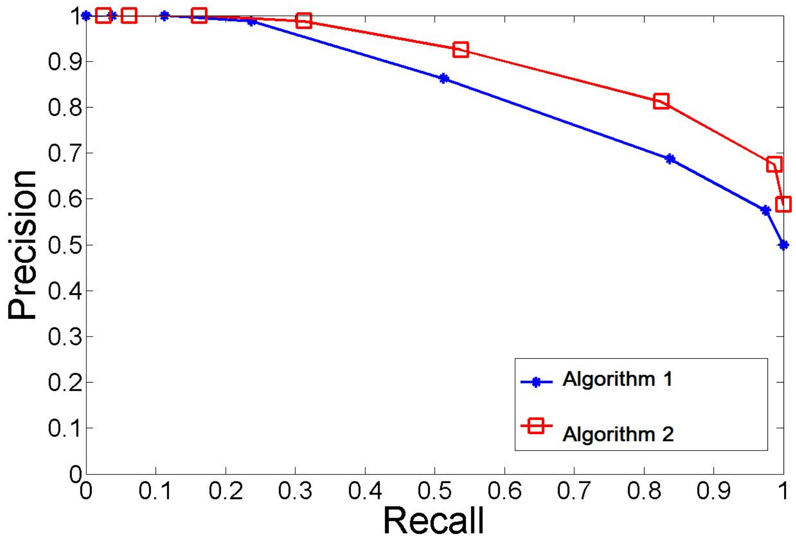
Construct roc:
1. Sort prediction column in descending order
2. Start with largest prediction and calculate fpr and tpr if threshold was at this point
3. Plot point in roc plot
4. Do this with every prediction value (going in desc order)

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: Look into PR-curves

Precision recall curve

The other often used performance measure for classification.



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/

Data Imbalance

Data Imbalance

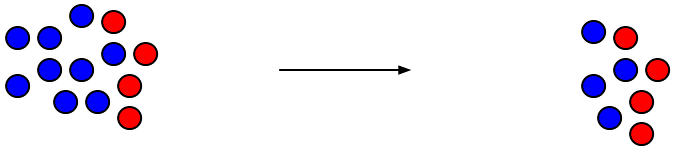
- Positive (or negative) data points are only a small fraction of all data.
  → Return prediction is typically an example of an imbalanced dataset.

- Imbalanced datasets are a problem for machine learning models [1, 2].
  → conventional algorithms are often biased towards the majority class
  → their loss functions attempt to optimize quantities such as error rate

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

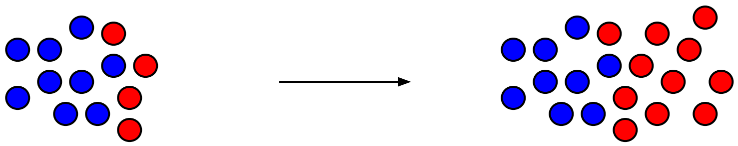
[1] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. 2015. When is Undersampling E ective in Unbalanced Classi cation Tasks?. In Machine Learn- ing and Knowledge Discovery in Databases. Springer International Publishing, 200–215.
[2] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi.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

Undersampling



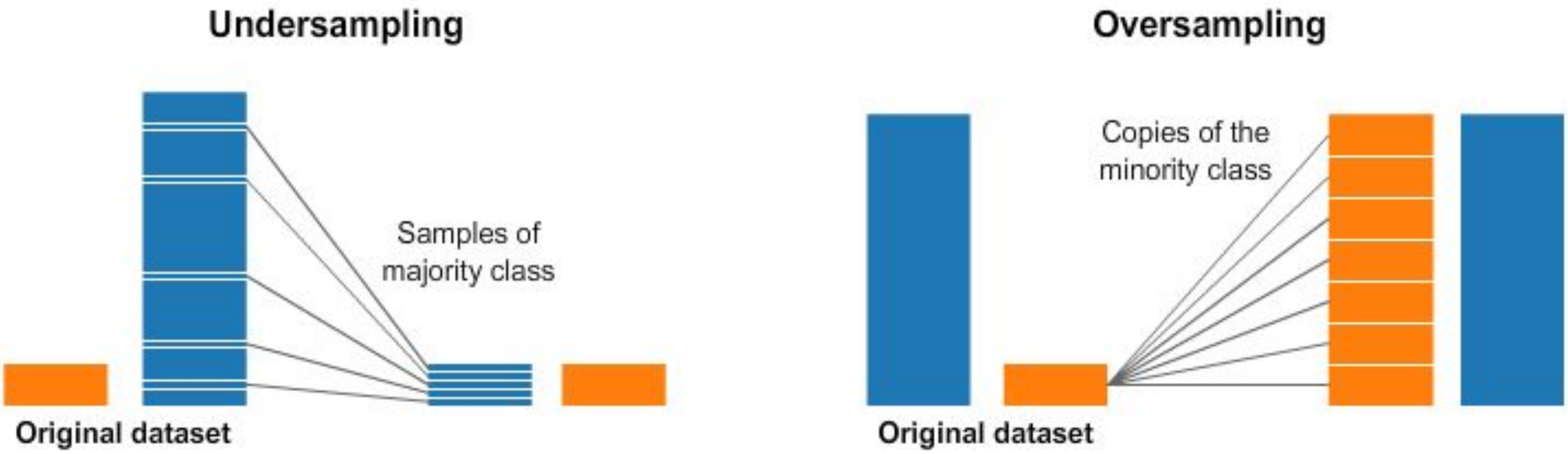
Randomly undersample the majority class
(= randomly remove negative data points)

Oversampling

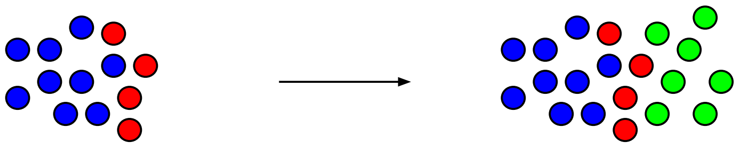


Randomly oversample the minority class with replacement
(= randomly duplicate positive data points)

Under/Over-sampling

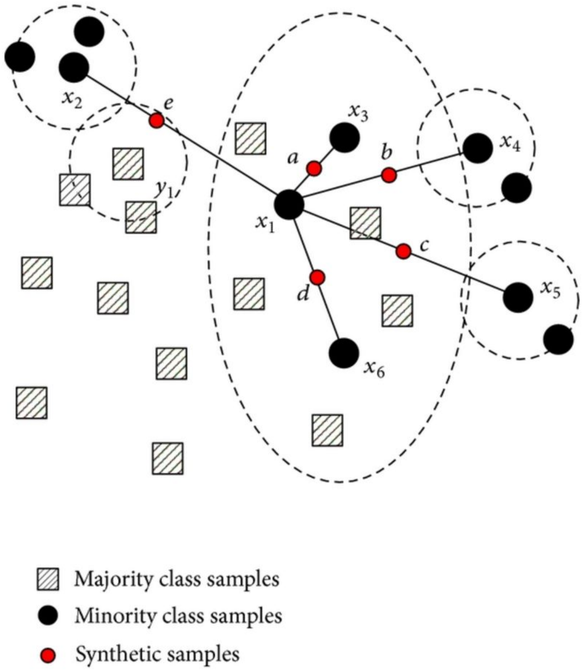


Data Augmentation



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

Many machine learning toolkits have ways to adjust the “importance” of classes.

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

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

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