Practical Aspects of Data Science

Data Science Retreat - 2018/01

Patrick Baier

About me

Patrick Baier

Short Bio:

- Machine Learning Engineer at Zalando (since ~ 3 years)
- PhD in Computer Science 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 (that we discuss).
2. Exercises (coding and research).
3. Presentations about DS in practice.

Course Overview

- **Model Learning (Day 1)**
  Classifier evaluation, imbalanced data problems, probability calibration
- **Model Operation (Day 2)**
  Model deployment, missing features, monitoring, model traceability

Running Example

Running Example

Given:

- Data about customers buying goods at an online shop
- Fraud label:
    - Class zero: people have paid their goods after receiving them
    - Class one: people have NOT paid their goods after receiving them

Task: Built a binary classification model that predicts in real-time the probability if a customers is good or a fraudsters :

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

Data Set

Data is given as day wise logs

```
➜  data ls -l fraud-data/
total 6696
-rw-r--r--  1 pbaier  domänen-benutzer  107832 Feb 13 18:46 2017-01-01.txt
-rw-r--r--  1 pbaier  domänen-benutzer  107855 Feb 13 18:46 2017-01-02.txt
-rw-r--r--  1 pbaier  domänen-benutzer  107265 Feb 13 18:46 2017-01-03.txt
-rw-r--r--  1 pbaier  domänen-benutzer  107483 Feb 13 18:46 2017-01-04.txt
-rw-r--r--  1 pbaier  domänen-benutzer  107979 Feb 13 18:46 2017-01-05.txt
```

Data Set

Every line of a daily file is one order in json format

```
➜  data head fraud-data/2017-01-01.txt
{"customerID": 6304965406, "basket": [4, 3, 0, 3, 1, 1, 2, 0, 2], "zipCode"
{"customerID": 1798132339, "basket": [3, 2, 1, 1], "zipCode": 29419, "total
{"customerID": 6840682240, "basket": [0, 1, 0], "zipCode": 49538, "totalAmo
{"customerID": 5544259179, "basket": [2, 3, 2, 3, 0, 3], "zipCode": 78753,
{"customerID": 5806156120, "basket": [3, 4, 1, 0], "zipCode": 58780, "total
{"customerID": 2505682934, "basket": [3], "zipCode": 24259, "totalAmount":
```

Data Set

One of these jsons:

```
➜ data head -n 1 fraud-data/2017-01-01.txt | jq .
{
  "customerID": 6304965406,
  "basket": [
    4,
    3,
    0,
    3,
    1,
    1,
    2,
    0,
    2
  ],
  "zipCode": 27291,
  "totalAmount": 810,
  "fraudLabel": 0
}
```

Task 1

- Start a jupyter notebook
- Read in the data as one dataframe
- Learn a vanilla* logistic regression:
    - Craft some features (but let's discuss this first once you are ready)
    - Use the `fraudLabel` 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

Accuracy

The number of examples the classifier classifies correctly:

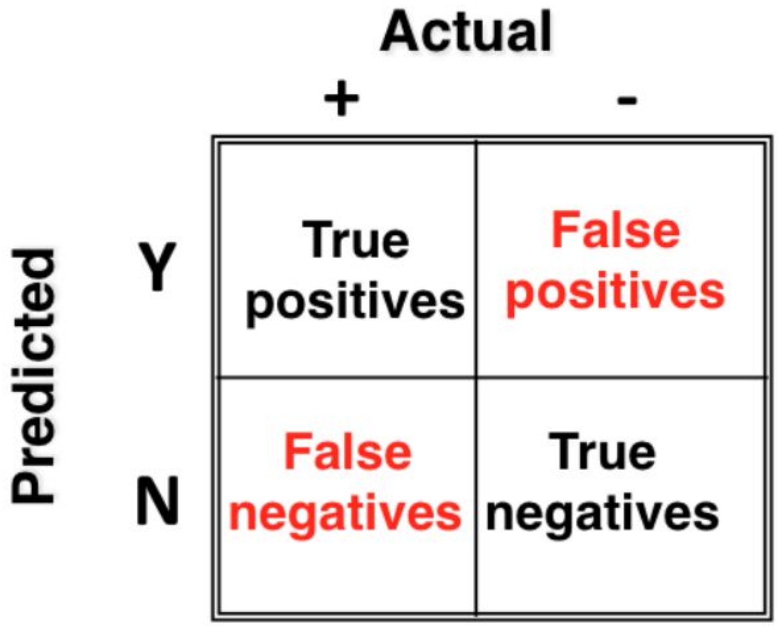
correct predictions / # all predictions

→Very intuitive and used very often

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

We need something else!

Confusion matrix



Quiz:

What is:
- true positive rate
- false positive rate
- sensitivity
- specificity
- recall
- precision
- F-score

They all rely on a certain cutoff c!

Classifier probability

In binary classification (class zero and class one), a model gives us the probability that data point belongs to class one, i.e. p = 0.7.

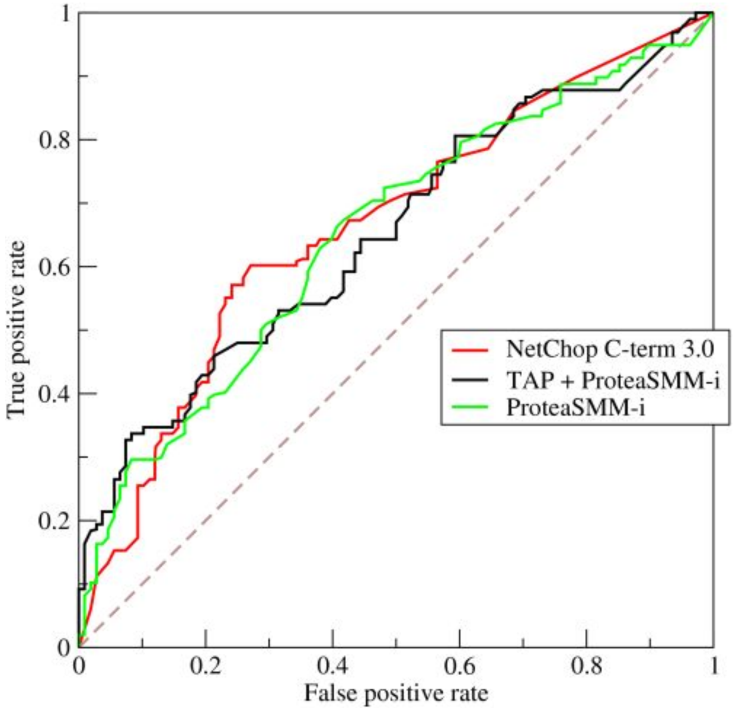
To decide which class we assign the data point to, we need a threshold:

$p => c \quad \rightarrow$ data point is in class one
$p < c \quad \rightarrow$ data point is in class zero

As default c is often set to 0.5. But often *c* is changed according to some business demands. Hence c is not know at evaluation time!

Receiver Operating Characteristic (roc curve)

- Shows for every threshold:
  - True positive rate (tpr):
    True positives / all positives
  - False positive rate (fpr):
    False positives / all positives
- 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

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.

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

Class Imbalance

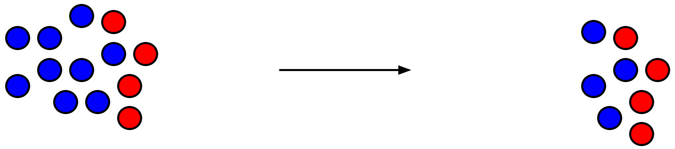
- Fraud detection is a classical example of an imbalanced dataset.
  → Positives data points (fraud) are only a small fraction of all data.

- Imbalanced datasets can be 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, synthetic data, adjust algorithm

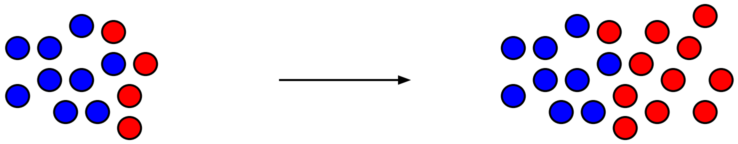
[1] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. 2015. When is Undersampling E ective in Unbalanced Classi cation Tasks?. In Machine Learn- ing and Knowledge Discovery in Databases. Springer International Publishing, 200–215.
[2] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi.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

Undersampling



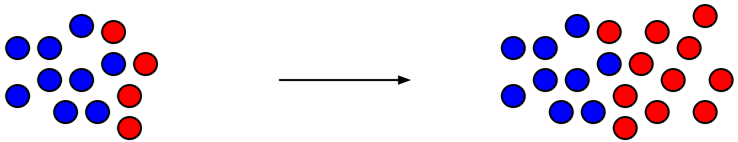
Randomly undersample the majority class
(= randomly remove non-fraud data points)

Oversampling



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

Synthetic Data



Create synthetic data points for the minority class, which are in some sense (e.g. distribution) similar to the minority class

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 model of task 1.
2. Use oversampling for the logistic regression model of task 1.
3. Compare the results to the previous performance in terms of auc.

Task 3

4. What methods are out there for generating synthetic data?
5. What does the “class weights” parameter do for Logistic regression in sklearn?

Probability Calibration

Problem description

In binary classification (class zero and class one), a model gives us the probability that a data point belongs to class one (i.e. $p = 0.7$).

How accurate/well-calibrated is this probability?

Example:
Assume an ML predictor outputs $p_{fraud} = 0.7$ for an order. If the predictor is well-calibrated, we can assume that 70% of the orders that have the same features as this orders are fraud.

Calibration

Why is this important?

Many real-world problems require a realistic probability rather than a (binary) decision.
→ Estimation of money at risk ($= \texttt{p}_{\texttt{fraud}}$ `* basketValue`).
→ Estimate click probability of a banner.

Different machine learning models are better or less suited:
→ Logistic regression: Naturally gives well-calibrated predictions.
→ Tree ensembles: Not so good calibrated predictions.
→ Oversampling leads to unrealistic probabilities.

Calibration Error (Brier Score)

$$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$

f_t predicted probability

o_t observed probability

N number of observations

Calibration Error (Brier Score)

How do we measure the calibration of our predictions?

1. Predict on test data.
2. Sort by raw probabilities in ascending order.
3. Create buckets for every x data points in ascending order.
4. For every bucket calculate:
   a. p_{real} : real probability (#ones / #all)
   b. p_{pred} : average predicted probability

5. RMSE(p_{real} , p_{pred}) over all buckets

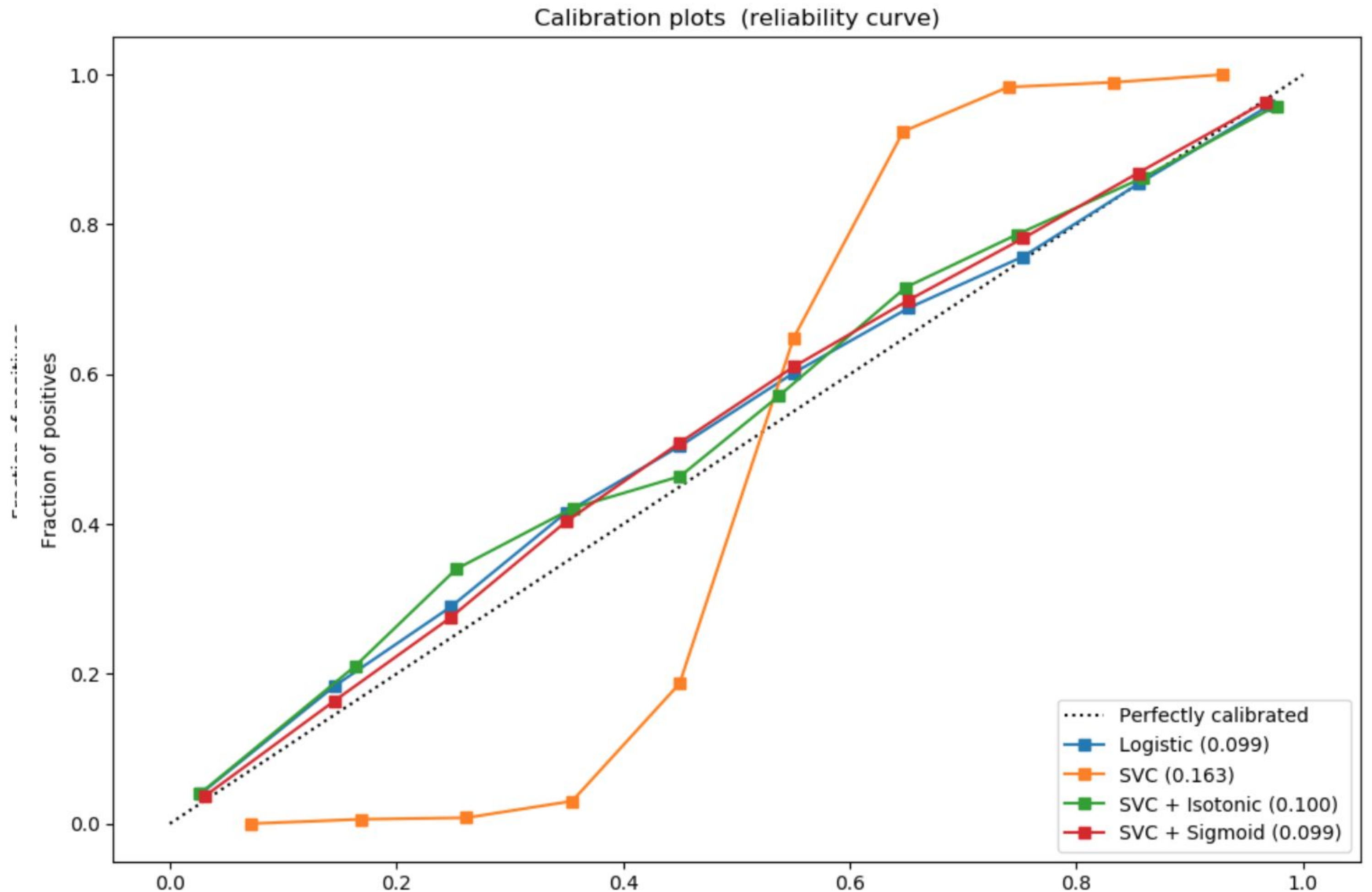
prediction	real label
...	...
0.51	0
0.53	0
0.54	1
0.56	1
0.58	1
...	...

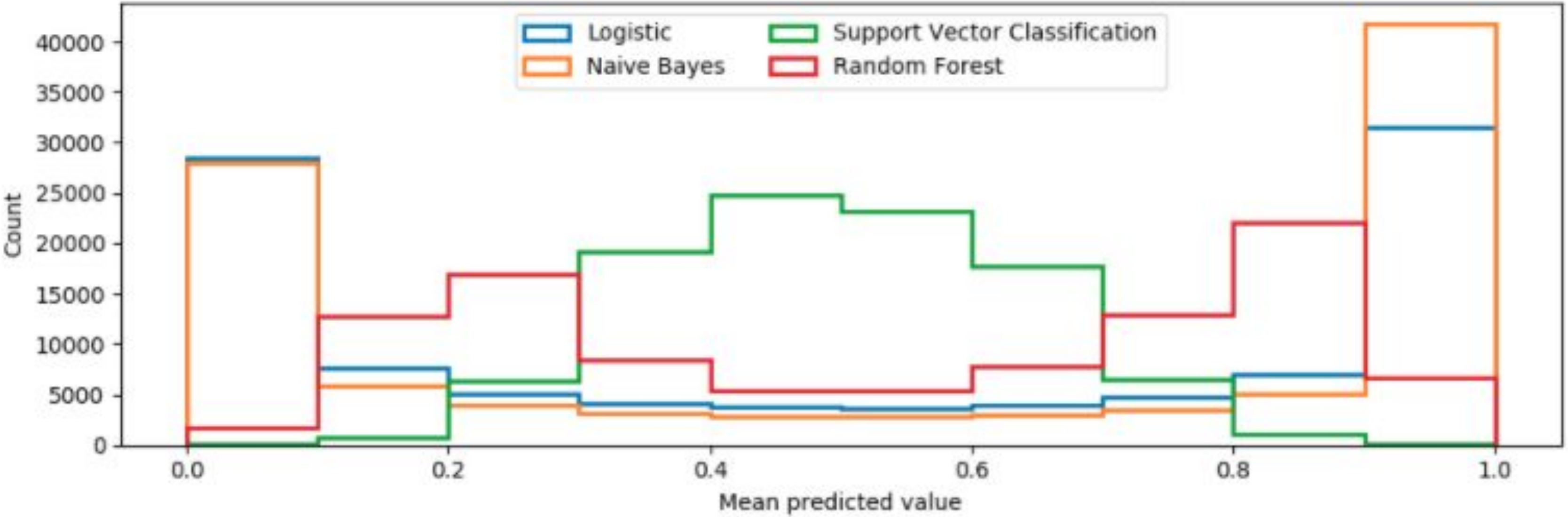
bucketSize = 5

$$p_{real} = 3 / 5 = 0.6$$

$$p_{pred} = (0.51+0.53+0.54+0.56+0.58)/5 = 0.544$$

$$err = 0.6-0.544$$


Calibration plots (reliability curve)
Fraction of positives
Perfectly calibrated
Logistic (0.099)
SVC (0.163)
SVC + Isotonic (0.100)
SVC + Sigmoid (0.099)
0.0
0.2
0.4
0.6
0.8
1.0



Very distribution depending on algorithm:

- Logistic regression has predictions all over the range
- Random Forest has only few predictions close to zero and one

(Simple) Calibration Algorithm

Learn calibration function on top of existing ML model on training data. This function maps raw prediction to calibrated prediction.

Learning:

1. Create buckets (as on previous slide) based on training predictions.
2. For every bucket: Calculate real fraud-probability (using true labels) and remember mapping: bucket → calibrated fraud-probability

Prediction:

1. Predict on data point with ML model.
2. Lookup bucket for prediction and output calibrated fraud-probability.

Task 4

- Check the calibration error of the models from task 1:
    - Form buckets of 100 consecutive predictions (starting from lowest probability).
    - Plot the error (x-axis: real probability, y-axis: predicted prob.)
    - Calculate the mean error over all buckets.
- Learn a calibration algorithm on the training data of task 1 and use it on the prediction of the test data.
- Compare the outcome to the uncalibrated case.

More Advanced Algorithms

There are more advanced algorithms for probability calibration (which in most cases do not turn out to be much better).

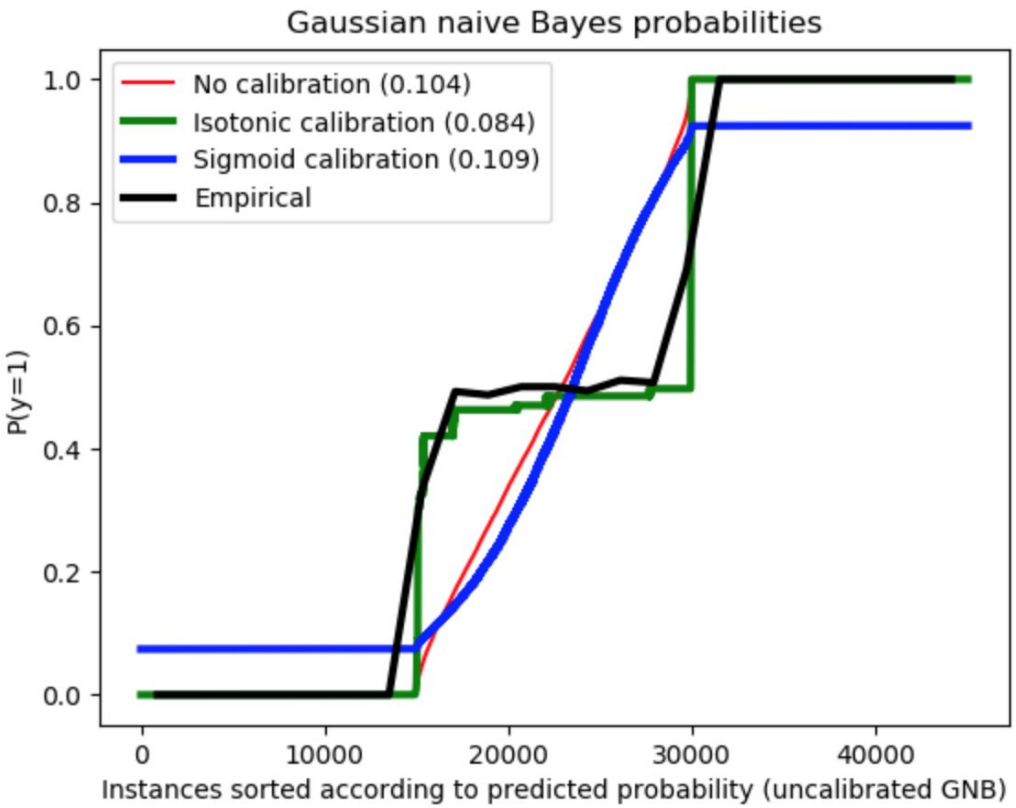
- Sigmoid calibration: Learn a logistic regression on top of the prediction

  $$p_{real} = 1 / (1 + e^{(a*p_{pred} + b)})$$

- Isotonic calibration:
  - Finds a non-decreasing approximation of a function
  - Results in a function that is piecewise linear (see next slide)
  - http://fa.bianp.net/blog/2013/isotonic-regression/

http://scikit-learn.org/stable/modules/calibration.html

PROBABILITY CALIBRATION



MIMIC Calibration Algorithm

One calibrated that works good in practice proposed by Magnetic.

Self study:
Read the following blog post and note down the characteristics of the calibration algorithm (one person needs to present the details).
How is it different from the one we implemented before?

http://tech.magnetic.com/2015/06/click-prediction-with-vowpal-wabbit.html