Anomaly Detection and AutoML

Simon Kluettermann

ls9 tu Dortmund

24. Oktober 2022

Simon Kluettermann

Anomaly Detection

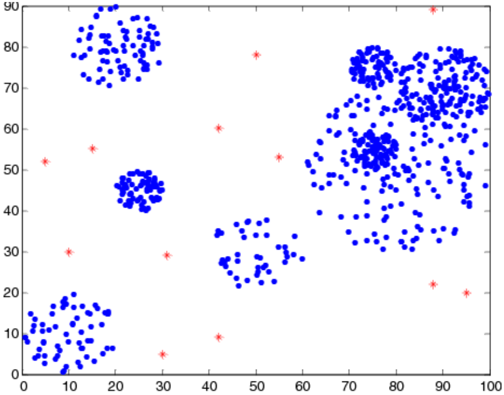
- Find strange (unexpected) samples.
- $\Rightarrow$If a traffic light is constantly yellow, probably something broke
- But this could happen in a lot of different ways
- $\Rightarrow$Most likely the traffic light is just off. But it could also fluctuate quickly or start smoking
- How to cover all possible anomalies?
- $\Rightarrow$Unsupervised Machine Learning

Unsupervised Machine Learning

- Normal machine learning: Input - Label
- Here: Only Input.
- $\Rightarrow$Instead of classifying different types, try to understand your given dataset
- Deviations from this understanding are anomalies
  - x: training samples
  - tx: test samples
  - ty: test labels (is a certain sample an anomaly or not)
- Useful: *peak /global/cardio.npz*

kNN

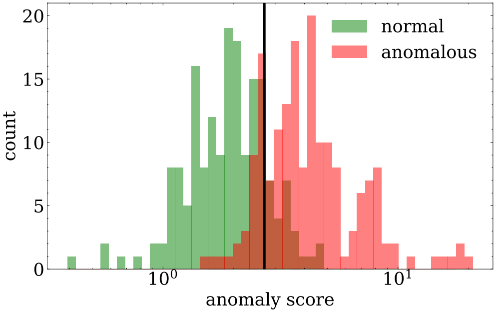
- How to do this? Here one algorithm: kNN
- Goal: Generate an anomaly score (high value⇒highly anomalous)
- Here: The anomaly score is the distance to the kth closest samples

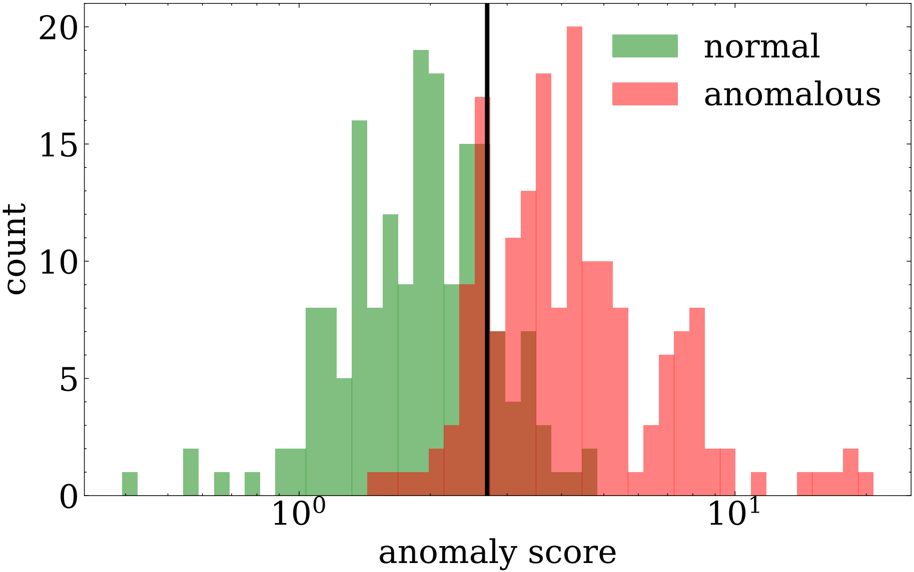


: [Yang, Huang 08]

kNN

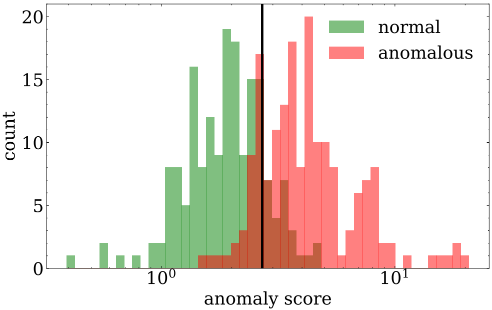
- How to do this? Here one algorithm: kNN
- Goal: Generate an anomaly score (high value$\Rightarrow$highly anomalous)
- Here: The anomaly score is the distance to the kth closest samples




20
15
10
5
0
count
10^0
10^1
anomaly score
normal
anomalous

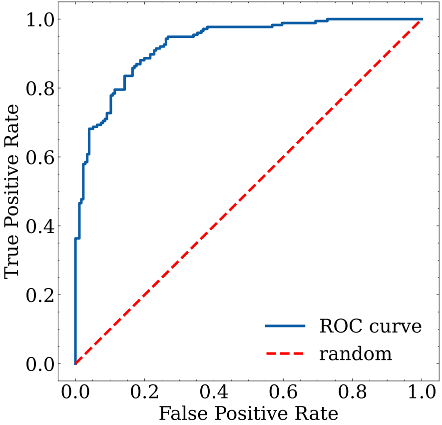
AUC Score

	True Positive	True Negative
Predicted Positive	147	27
Predicted Negative	29	149



AUC Score

- Iterate every threshold
- Plot fpr vs tpr
- False Positive Rate
  - $\frac{FP}{FP+TN}$
- True Positive Rate
  - $\frac{TP}{TP+FN}$
- ROC-AUC: Integral of this curve!



AUC Score

- calculcate with *sklearn.metrics.roc_auc_score*
- Higher AUC score$\Rightarrow$better
- $AUC = 1.0$$\Rightarrow$Perfect seperation
- $AUC = 0.5$$\Rightarrow$Random model
- $AUC = 0.0$$\Rightarrow$Inverse seperation (every anomaly is normal, and every normal sample is anomalous)

AUC Scores

Student	Algorithm	AUC
Priyanka	ocsvm	0,942
Shubham	lof	0,938
Rama	iforest	0,939
Upanishadh	knn	0,927
Kartik	gmm	0,929
Abir	pca	**0,947**

AutoML

- But: We can beat this!
- How? Hyperparameter
  - Every algorithm has hyperparameter that control how it works
  - For example: k in kNN (number of close points considered)
- Lets take the worst algorithm (kNN: 0.927) and try to improve it

Optimize

```python
import numpy as np
from pyod.models.knn import KNN

from sklearn.metrics import roc_auc_score

f=np.load("/global/cardio.npz")

x,tx,ty=f["x"],f["tx"],f["ty"]

def train_one(**hyper):
    model=KNN(**hyper)
    model.fit(x)
    return roc_auc_score(ty,model.decision_function(tx))
```

Optimize

```
n_neighbors_options=[1,2,3,4,5,6,7,8,9,10]
method_options=["median","mean","largest"]


best_result=0.0
best_hyper={}
for n_neighbors in n_neighbors_options:
    for method in method_options:
        hyper={"n_neighbors":n_neighbors,"method":method}
        result=train_one(**hyper)
        if result>best_result:
            best_result=result
            best_hyper=hyper

print("Best result: ",best_result)
print("Best hyper: ",best_hyper)

#Best result:  0.9585808367768595
#Best hyper:  {'n_neighbors': 2, 'method': 'median'}
```

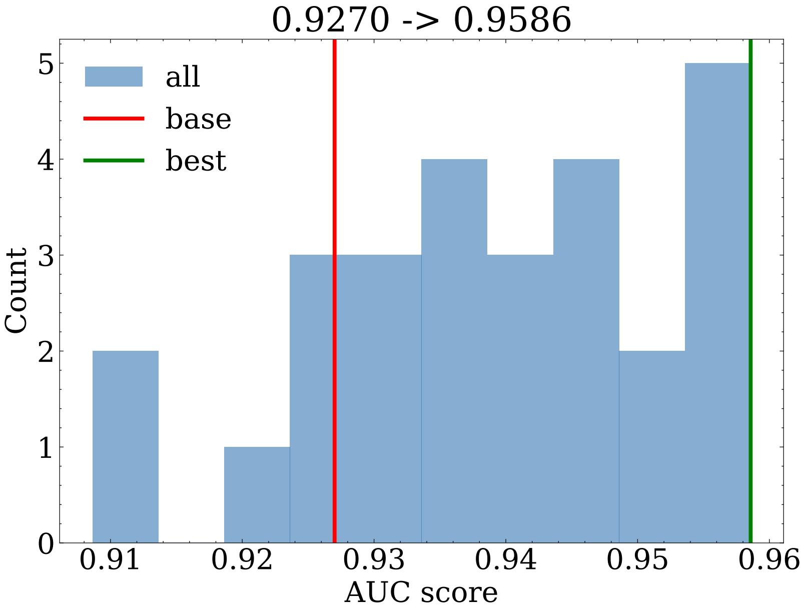
flaml

- *source folder/bin/activate*
- *pip install flaml*

```
import time
def optimization(config: dict):
    t0=time.time()
    auc=train_one(**config)
    t1=time.time()

    return {"score":auc,"evaluation_cost":t1-t0}
```

flaml

```
from flaml import tune

hyperparameters={"n_neighbors":tune.randint(lower=1,upper=10),"method":tune.choice(["median","mean","largest"])}


sol=tune.run(optimization,metric="score",mode="max",config=hyperparameters,resources_per_trial={"cpu":4,"gpu":0},num_samples=1000,time_budget_s
=60)
```


0.9270 -> 0.9586
all
base
best
Count
0
1
2
3
4
5
0.91
0.92
0.93
0.94
0.95
0.96
AUC score

Your Turn

- Remember your last algorithm
- Find its hyperparameters (Tip: pyod website)
- Optimize your algorithm and give me a new AUC!
- Bonus Question: Is there a problem with what we are doing?