Project 1: Network Traffic Type Classification

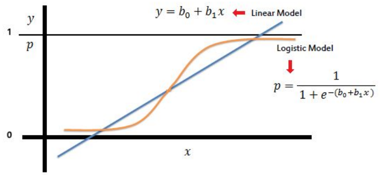
Chaeeun Ryu

23.05.08

Table of Contents

01
Classifier selection

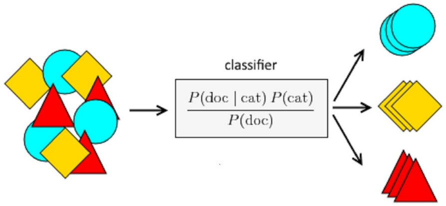
Training Accuracy with Naive Models



Logistic Regression

Train acc: **0.7902**

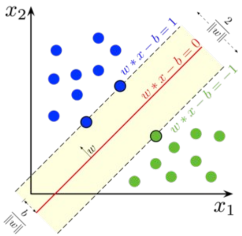
...



Naive Bayes

Train acc: **0.7996**

...

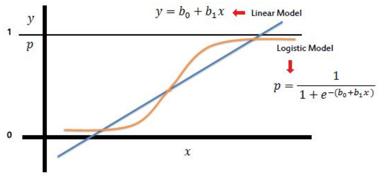


Support Vector Machine

Train acc: **0.8431**

...

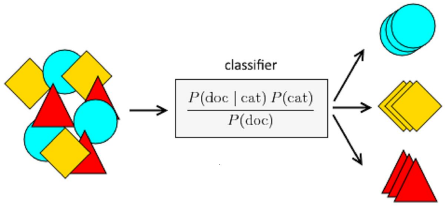
Training Accuracy with Naive Models



Logistic Regression

Train acc: **0.7902**

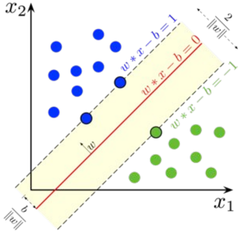
...



Naive Bayes

Train acc: **0.7996**

...



Support Vector Machine

Train acc: **0.8431**

...

+ Kernel => Better Performance!

Kernelizing SVM with RBF kernel

Ref:

- *CS229 Lecture- SVM with Kernels*
- *CS229 Problem Set*
- *github.com/superbunny38/MachineLearning*

RBF Kernel Equation

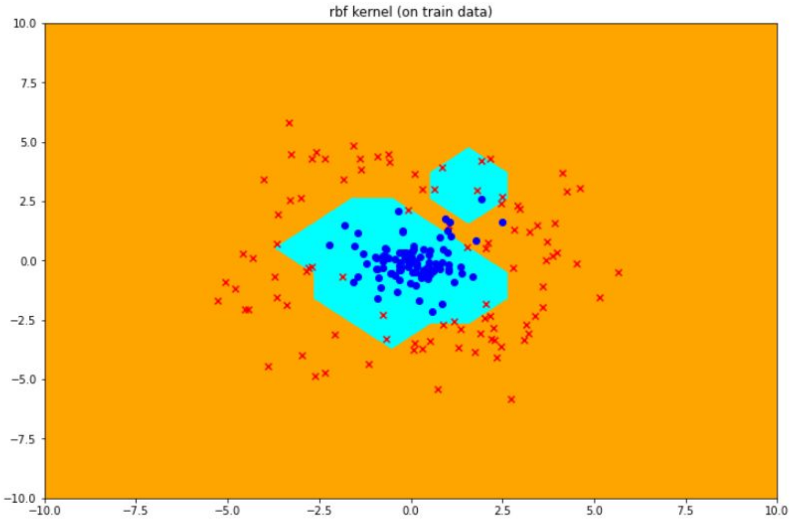
$$K(X_1, X_2) = \exp(-\frac{\|X_1 - X_2\|^2}{2\sigma^2})$$

Weight Update Rule

$$\beta_{i+1} = \alpha(y^{(i+1)} - sign(\sum_{j=1}^{i} \beta_j \times K(x^{(j)}, x^{(i+1)})))$$

Inference Rule

$$pred(x^{(i)}) = sign\sum_{j}^{n_{train}} \beta_j K(x^{(i)}, x_{train}^{(j)})$$



Note in naive SVM:

$$\overrightarrow{X}.\overrightarrow{w} - c \geq 0$$

putting $-c$ as b, we get

$$\overrightarrow{X}.\overrightarrow{w} + b \geq 0$$

hence

$$y = \begin{cases} +1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + b \geq 0 \\ -1 & \text{if } \overrightarrow{X}.\overrightarrow{w} + b < 0 \end{cases}$$

Kernelizing SVM with RBF kernel

RBF Kernel Equation

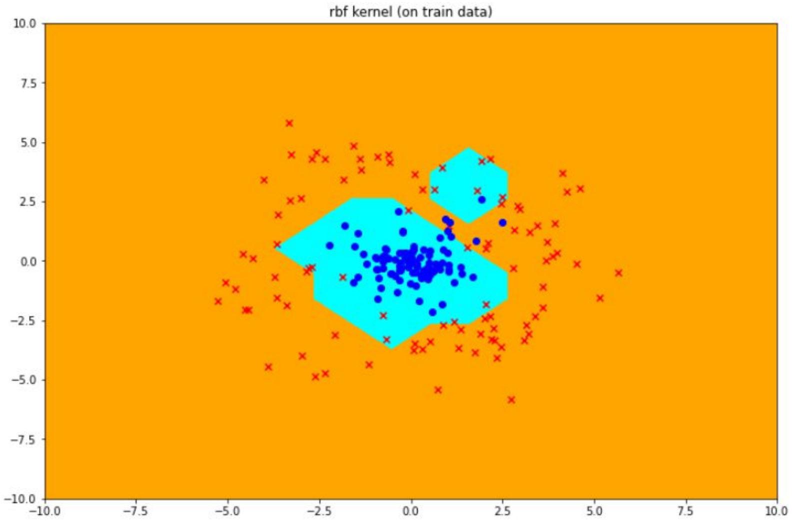
$$K(X_1, X_2) = \exp(-\frac{\|X_1 - X_2\|^2}{2\sigma^2})$$

Weight Update Rule

$$\beta_{i+1} = \alpha(y^{(i+1)} - sign(\sum_{j=1}^{i} \beta_j \times K(x^{(j)}, x^{(i+1)})))$$

Inference Rule

$$pred(x^{(i)}) = sign \sum_{j}^{n_{train}} \beta_j K(x^{(i)}, x^{(j)}_{train})$$



Kernelizing SVM with RBF kernel

RBF Kernel Equation

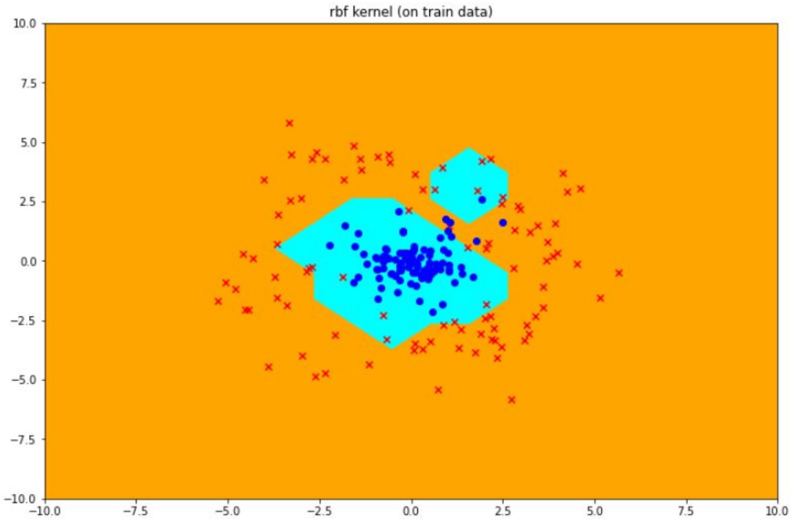
$$K(X_1, X_2) = \exp(-\frac{\|X_1 - X_2\|^2}{2\sigma^2})$$

Weight Update Rule

$$\boxed{\beta_{i+1} = \alpha(y^{(i+1)} - sign(\sum_{j=1}^{i} \beta_j \times K(x^{(j)}, x^{(i+1)})))}$$

Inference Rule

$$\boxed{pred(x^{(i)}) = sign\sum_{j}^{n_{train}} \beta_j K(x^{(i)}, x^{(j)}_{train})}$$



Very costly!

Kerneliz

RBF Kernel Equation

$$K(X_1, X_2) = \exp(-\frac{\|X_1 - X}{2\sigma^2}$$

Weight Update Rule

$$\boxed{\beta_{i+1} = \alpha(y^{(i+1)} - sign(\sum_{j=1}^{i} \beta_j \times K(x^{(j)}, x^{(i+1)})))}$$

Very costly!

[Complexity for Training one Multi-class Classifier]

(Note. Data was enlarged due to oversampling N > original N)
For a single datapoint: O(N)
For all training data: O(N^2)
For n iterations: n*O(N^2)
For 4 classes: 4*n*O(N^2)
For 5-fold stratified partitioned cross validation: 5*4*n*O(N^2)
For Hyper-parameter search over sigma and learning rate with K combinations: **K*5*4*n*O(N^2)**

```
lrs = [0.009,0.01,0.02,0.03,0.05]
sigmas = [0.09,0.1,0.2,0.3,0.5]
```

-> RAM ran out in my desktop :(

[slide]

Kerneliz

[Complexity for Training one Multi-class Classifier]

(Note. Data was enlarged due to oversampling N > original N)
For a single datapoint: O(N)
For all training data: O(N^2)
For n iterations: n*O(N^2)
For 4 classes: 4*n*O(N^2)
For 5-fold stratified partitioned cross validation: 5*4*n*O(N^2)
For Hyper-parameter search over sigma and learning rate with K combinations: **K*5*4*n*O(N^2)**

-> RAM ran out in my desktop :(

RBF Kernel Equation

$$K(X_1, X_2) = \exp(-\frac{\|X_1 - X}{2\sigma^2}$$

Weight Update Rule

$$\boxed{\beta_{i+1} = \alpha(y^{(i+1)} - sign(\sum_{j=1}^{i} \beta_j \times K(x^{(j)}, x^{(i+1)})))}$$

Very costly!

Inference Rule

$$pred(x^{(i)}) = sign \sum_j^{\boxed{n_{train}}} \beta_j K(x^{(i)}, x_{train}^{(j)})$$

02

Feature Engineering

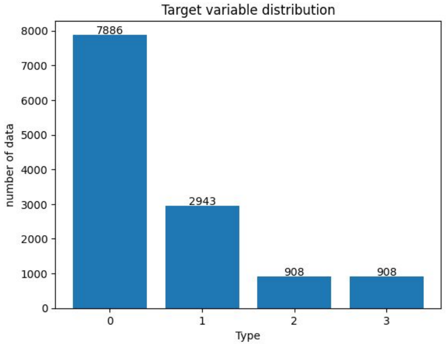
Adding more non-linearity in data

Target variable을 제외한 모든 서로 다른 두 쌍의 input variables로 나눠서 새로운 feature를 더함

```
from itertools import combinations
combi_list = sorted(combinations(np.arange(11),2))
for combin in list(combi_list):
    i,j = list(combin)[0],list(combin)[1]
    train_df[f'new_feature{i}_{j}'] = train_df.iloc[:,i]/train_df.iloc[:,j]
```

	traffic(t-10)	traffic(t-9)	traffic(t-8)	traffic(t-7)	traffic(t-6)	traffic(t-5)	traffic(t-4)	traffic(t-3)	traffic(t-2)	traffic(t-1)	...	new_feature6_7	new_feature6_8	new_feature6_9	new_feature6_10	new_feature7_8	new_feature7_9
0	3.0	2.0	5.0	6.0	6.0	4.0	6.0	6.0	3.0	12.0	...	1.00	2.0	0.5	1.000000	2.00	0.5
1	2.0	5.0	6.0	6.0	4.0	6.0	6.0	3.0	12.0	6.0	...	2.00	0.5	1.0	1.000000	0.25	0.5
2	5.0	6.0	6.0	4.0	6.0	6.0	3.0	12.0	6.0	6.0	...	0.25	0.5	0.5	0.750000	2.00	2.0
3	6.0	6.0	4.0	6.0	6.0	3.0	12.0	6.0	6.0	4.0	...	2.00	2.0	3.0	2.000000	1.00	1.5
4	6.0	4.0	6.0	6.0	3.0	12.0	6.0	6.0	4.0	6.0	...	1.00	1.5	1.0	0.352941	1.50	1.0

5 rows × 67 columns



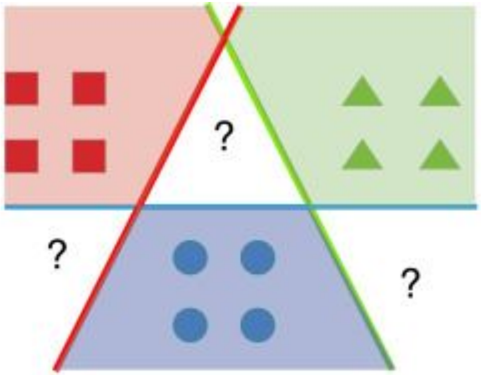
Label 0에 대한 accuracy 증가, label 1에 대한 accuracy 감소

-> trade off exists, but better! (& needs guidance for label 1)

03

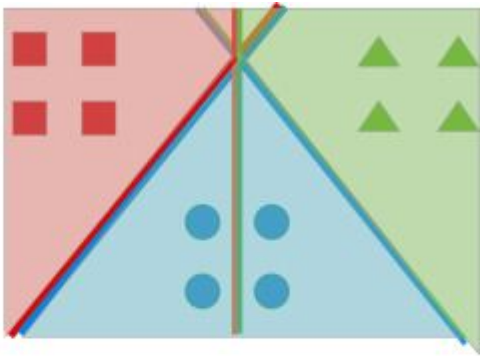
Dealing with Multiclass Classification

Method Selected



(a) Separation with OvA.

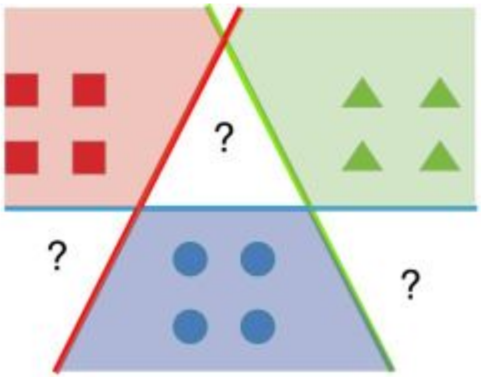
One vs All
= 4 Classes
= 4 Classifiers



(b) Separation with OvO.

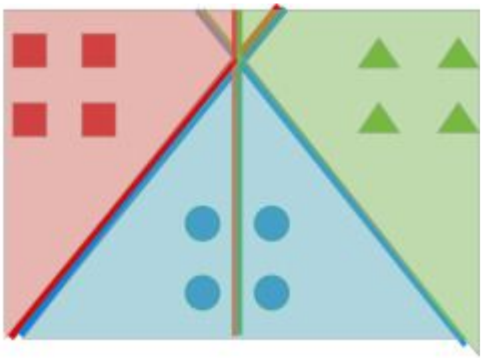
One vs One
= 4 Classes
= (4)(4-1)/2 Classifiers
= 6 Classifiers

Method Selected



(a) Separation with OvA.

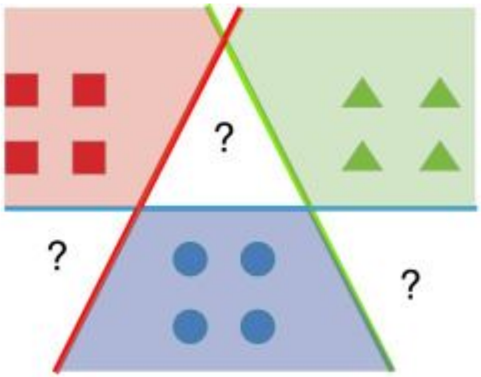
One vs All
= 4 Classes
= 4 Classifiers



(b) Separation with OvO.

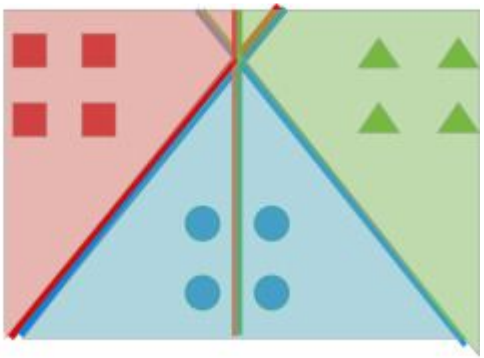
One vs One
= 4 Classes
= (4)(4-1)/2 Classifiers
= 6 Classifiers

Method Selected



(a) Separation with OvA.

One vs All
= 4 Classes
= 4 Classifiers



(b) Separation with OvO.

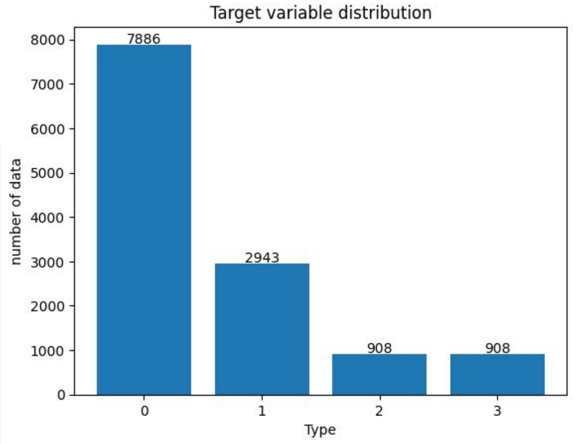
One vs One
= 4 Classes
= (4)(4-1)/2 Classifiers
= 6 Classifiers

Lr = 0.001, sigma = 0.1, n_iters = 20
Accuracy on unseen validation set: ≈0.91

```
predicted train dist: Counter({0: 7130, 1: 2181, 2: 560, 3: 245})
predicted test dist: Counter({0: 1791, 1: 556, 2: 118, 3: 64})
(C:0.01 gamma:0.1) acc: 0.9187 val acc:0.901
```

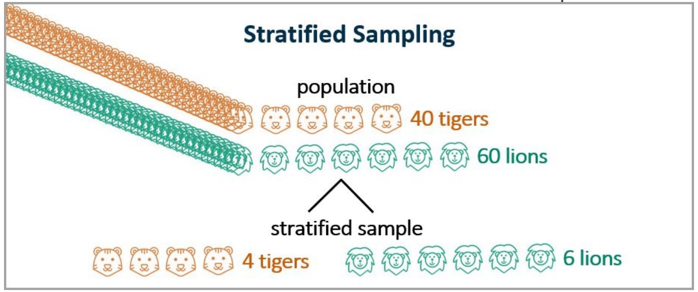
Result for 1 iteration with lr = 0.01, sigma = 0.1

04
Dealing with Data Imbalance



Strategies for Dealing with Data Imbalance

Stratified Partitioned 5-fold Cross Validation
: <u>All folds have very similar distribution</u>

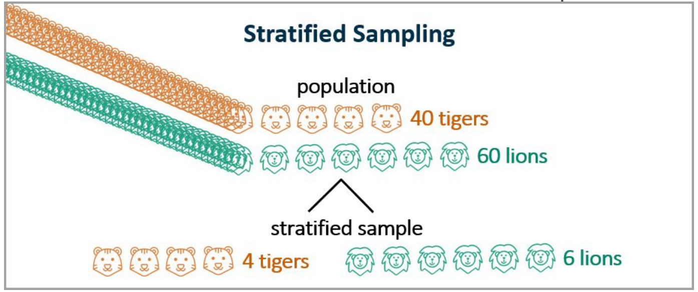


Target distribution for every fold

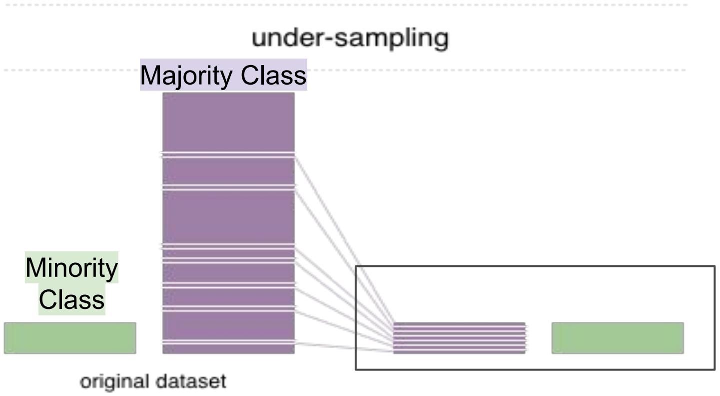
Fold1 - Counter({0: 1578, 1: 588, 2: 182, 3: 181})
Fold2 - Counter({0: 1577, 1: 589, 2: 182, 3: 181})
Fold3 - Counter({0: 1577, 1: 589, 3: 182, 2: 181})
Fold4 - Counter({0: 1577, 1: 589, 3: 182, 2: 181})
Fold5 - Counter({0: 1577, 1: 588, 2: 182, 3: 182})

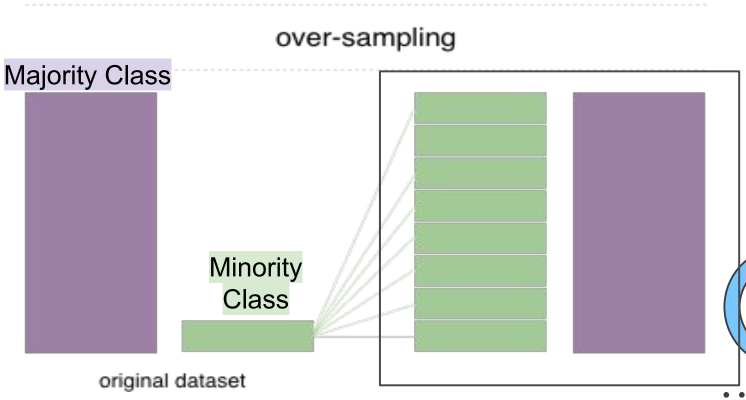
Strategies for Dealing with Data Imbalance

Stratified Partitioned 5-fold Cross Validation
: <u>All folds have very similar distribution</u>



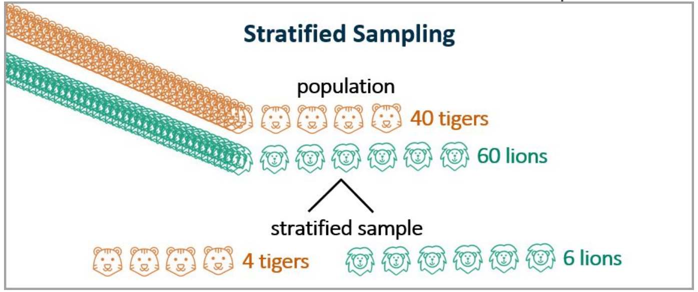
Sampling during Training



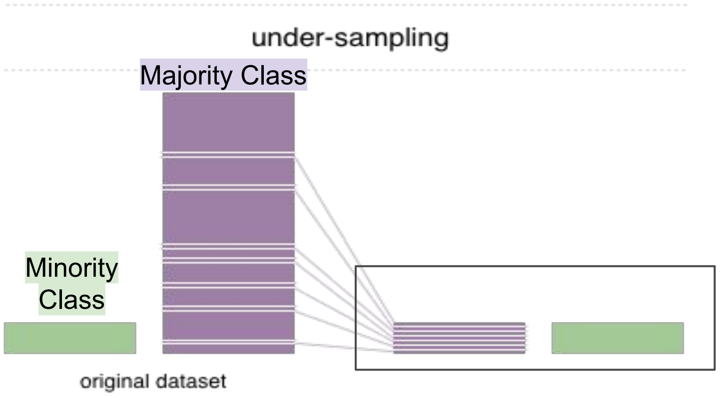


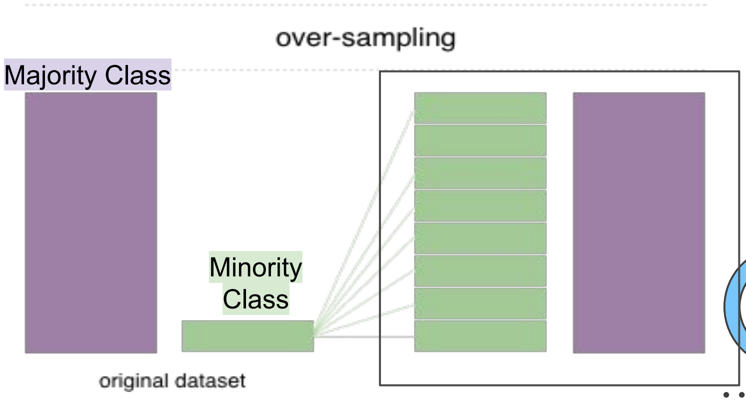
Strategies for Dealing with Data Imbalance

Stratified Partitioned 5-fold Cross Validation
: <u>All folds have very similar distribution</u>



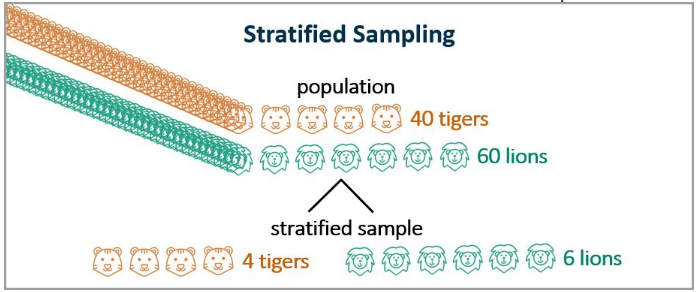
Sampling during Training





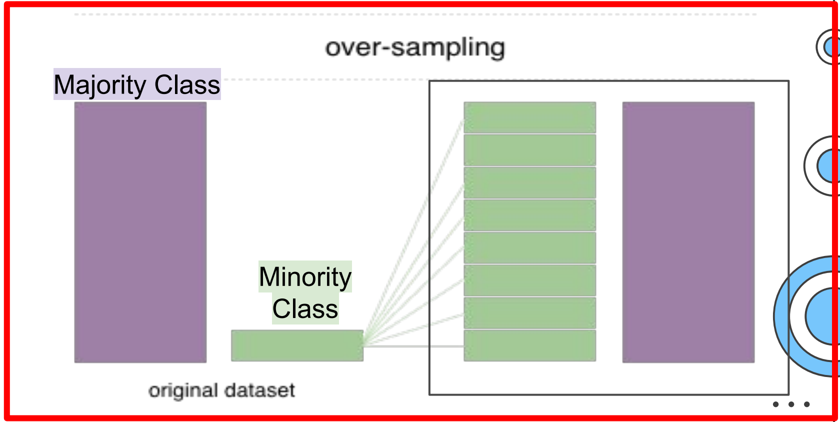
Strategies for Dealing with Data Imbalance

Stratified Partitioned 5-fold Cross Validation
: <u>All folds have very similar distribution</u>



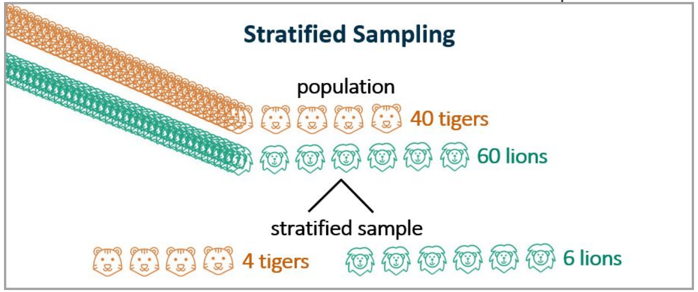
Sampling during Training

Fold1 - {0: 1578, 1: 588, 2: 182, 3: 181})
Fold2 - {0: 1577, 1: 589, 2: 182, 3: 181})
Fold3 - {0: 1577, 1: 589, 3: 182, 2: 181})
Fold4 - {0: 1577, 1: 589, 3: 182, 2: 181})
Fold5 - {0: 1577, 1: 588, 2: 182, 3: 182})



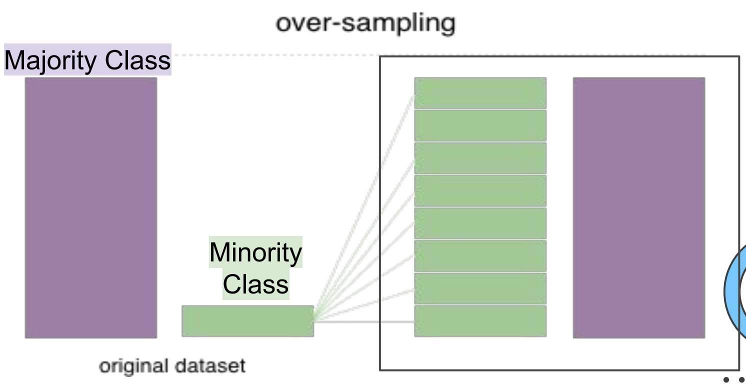
Strategies for Dealing with Data Imbalance

Stratified Partitioned 5-fold Cross Validation
: All folds have very similar distribution



Sampling during Training

Fold1 - {0: 1578, 1: **1578**, 2: **1578** 3: **1578**})
Fold2 - {0: 1577, 1: **1577**, 2: **1577** 3: **1577**})
Fold3 - {0: 1577, 1: **1577**, 3: **1577** 2: **1577**})
Fold4 - {0: 1577, 1: **1577**, 3: **1577** 2: **1577**})
Fold5 - {0: 1577, 1: **1577**, 2: **1577** 3: **1577**})



05

Classifier Guidance with Intersection of Predictions for Minority Classes

Base Classifier Guidance

[Base model]
Multi-class Classifier with unified hyper parameters for the binary classifier for each class

Classifier for Type 0
(lr: 1e-2, sigma: 0.1)

Classifier for Type 1
(lr: 1e-2, sigma: 0.1)

Classifier for Type 2
(lr: 1e-2, sigma: 0.1)

Classifier for Type 3
(lr: 1e-2, sigma: 0.1)

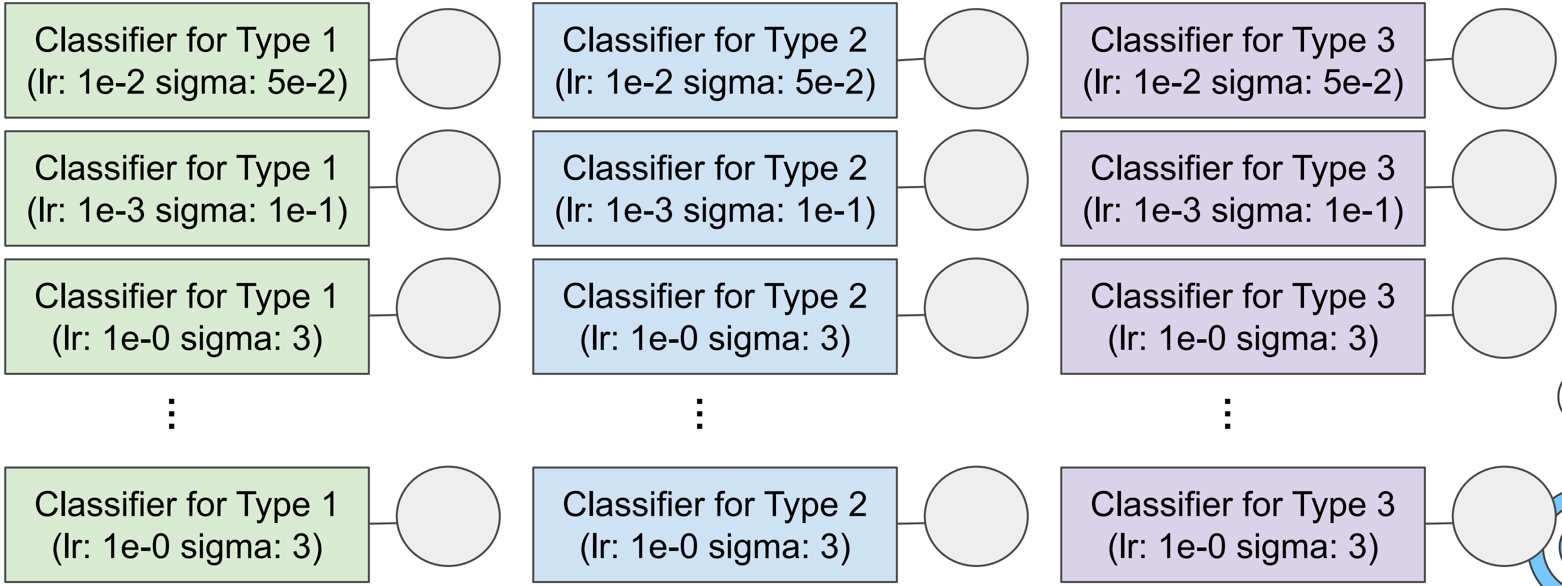
예측값

≈0.91 validation accuracy

Time Time..

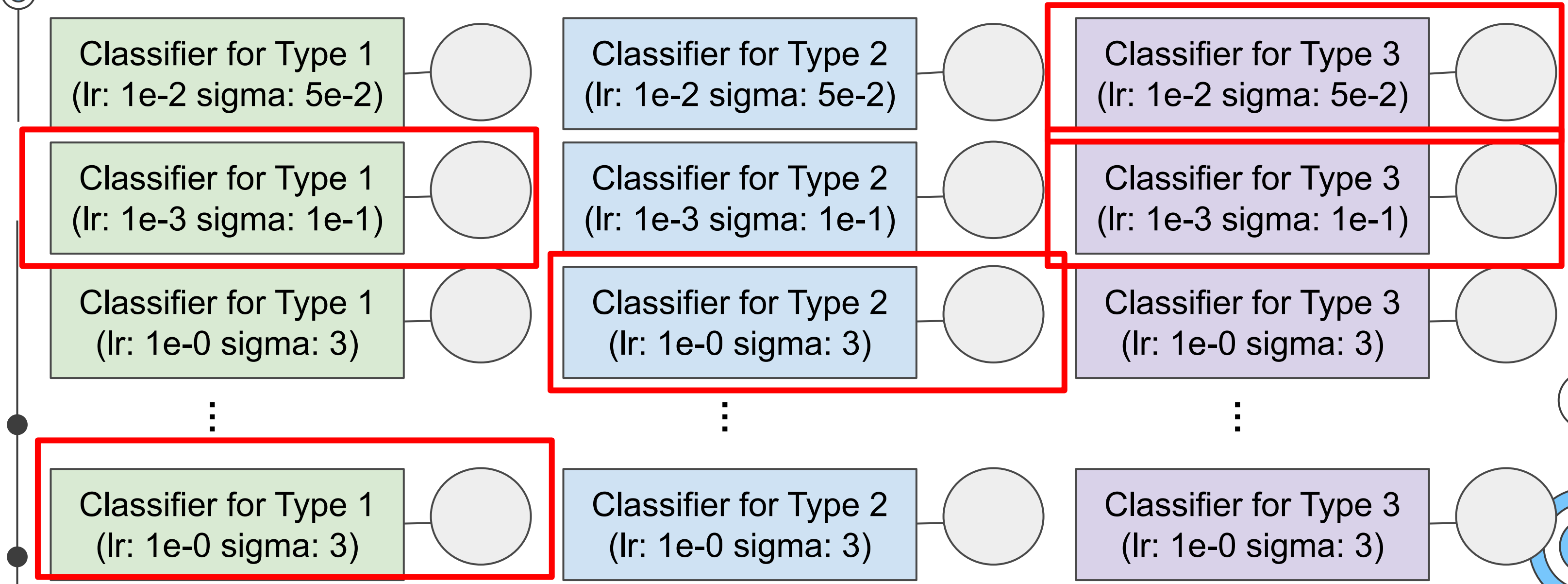
Base Classifier Guidance

[Binary Model]
Binary Classifier for One class

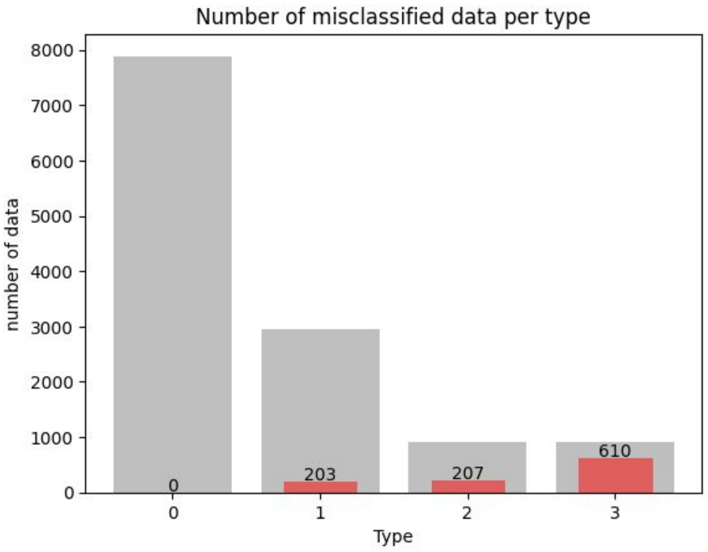


Base Classifier Guidance

[Binary Model]
Binary Classifier for One class



Base Classifier Guidance



Base model

Intersection

Prediction with model of 0.9512 accuracy for type 1
Prediction with model of 0.9619 accuracy for type 1
Prediction with model of 0.9744 accuracy for type 1
Prediction with model of 0.9862 accuracy for type 1

Union

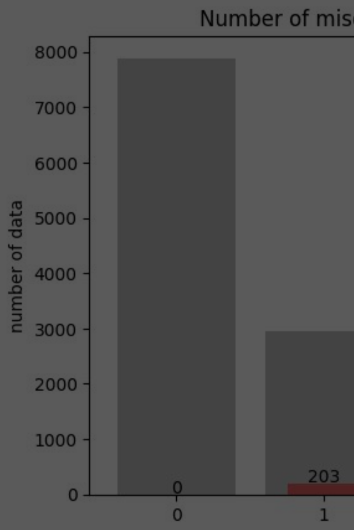
Intersection

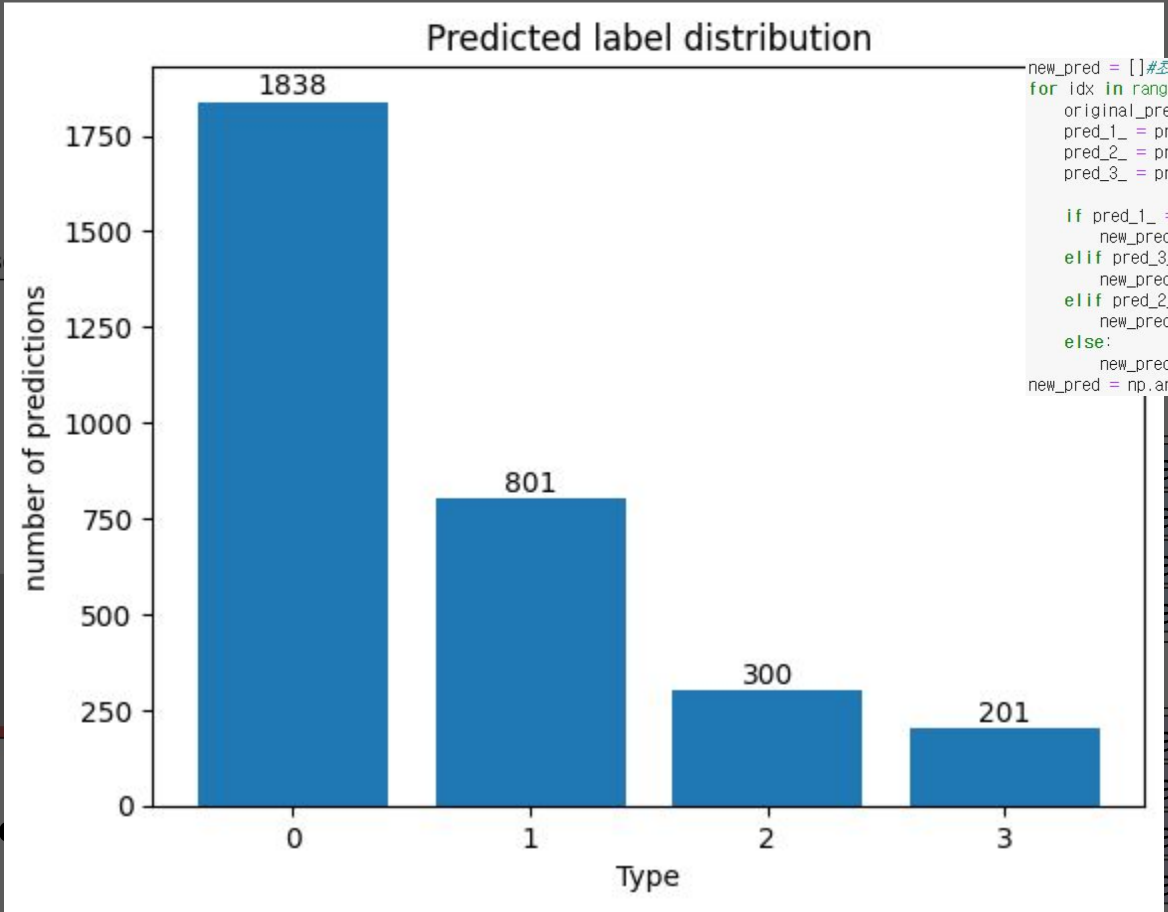
Prediction with model of 0.9512 accuracy for type 2
Prediction with model of 0.9619 accuracy for type 2
Prediction with model of 0.9744 accuracy for type 2
Prediction with model of 0.9862 accuracy for type 2

Intersection

Prediction with model of 0.9512 accuracy for type 3
Prediction with model of 0.9619 accuracy for type 3
Prediction with model of 0.9744 accuracy for type 3
Prediction with model of 0.9862 accuracy for type 3

≈ 0.909 Test Accuracy






```python
new_pred = []#최종 prediction
for idx in range(len(rbf_pred)):
    original_pred = rbf_pred[idx]#base model's prediction
    pred_1_ = pred_1[idx]#교집합 for label 1
    pred_2_ = pred_2[idx]#교집합 for label 2
    pred_3_ = pred_3[idx]#교집합 for label 3

    if pred_1_ == 1:
        new_pred.append(1)
    elif pred_3_ == 1:
        new_pred.append(3)
    elif pred_2_ == 1:
        new_pred.append(2)
    else:
        new_pred.append(original_pred)
new_pred = np.array(new_pred)
```

- label1_0.92.pt
- label1_0.94.pt
- label1_0.94_2.pt
- label1_0.95.pt
- label1_0.97.pt
- label1_0.97_2.pt
- label2_0.94.pt
- label3_0.93.pt
- label3_0.95.pt
- label3_0.96.pt
- not_div_label2_0.95.pt
- not_div_label2_0.96.pt
- os_5iters_new_multiclass_0.90.pt
- std_w_traindf (1).csv
- wo_rbf_0.92.pt

Bas

acy for t

acy for t

acy for t

acy for t

acy for t

acy for t

acy for t

acy for t

≈ 0.909 <u>Test</u> Accuracy

Thank you!