Week12

Lecture

Default of Credit Card Clients Dataset

- This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005
  - ID: ID of each client
  - LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
  - SEX: Gender (1=male, 2=female)
  - EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
  - MARRIAGE: Marital status (1=married, 2=single, 3=others)
  - AGE: Age in years

Default of Credit Card Clients Dataset

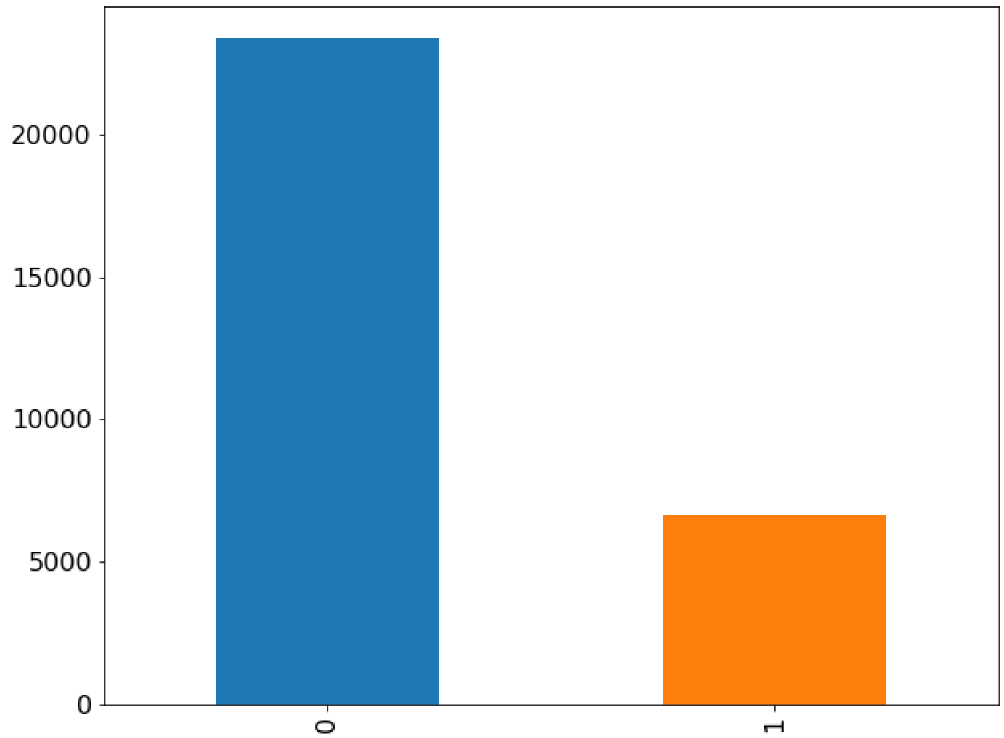
- This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005
  - PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
  - PAY_2: Repayment status in August, 2005 (scale same as above)
  - PAY_3: Repayment status in July, 2005 (scale same as above)
  - PAY_4: Repayment status in June, 2005 (scale same as above)
  - PAY_5: Repayment status in May, 2005 (scale same as above)
  - PAY_6: Repayment status in April, 2005 (scale same as above)
  - BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
  - BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
  - BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
  - BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
  - BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
  - BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)

Default of Credit Card Clients Dataset

- This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005
  - PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
  - PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
  - PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
  - PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
  - PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
  - PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)
  - **default.payment.next.month: Default payment (1=yes, 0=no)**

Default of Credit Card Clients Dataset

- The distribution of target classes
  - # of default(1) = 6636 (22.12%)



Data Preparation

- Data Load

```
import pandas as pd

data=pd.read_csv('https://drive.google.com/uc?export=download&id=1gd2jStJinE_egX7LCKnh1-_lxafRI-zF')
```

- Convert categorical variables to dummy variables
  - Which variables are categorical?

Build a Classifier

- Build a classifier for prediction default
  - Split data into two subsets: training and validation sets

```
from sklearn.model_selection import train_test_split
trnX,valX,trnY,valY=train_test_split(X,y, test_size=0.2, random_state=10)
```

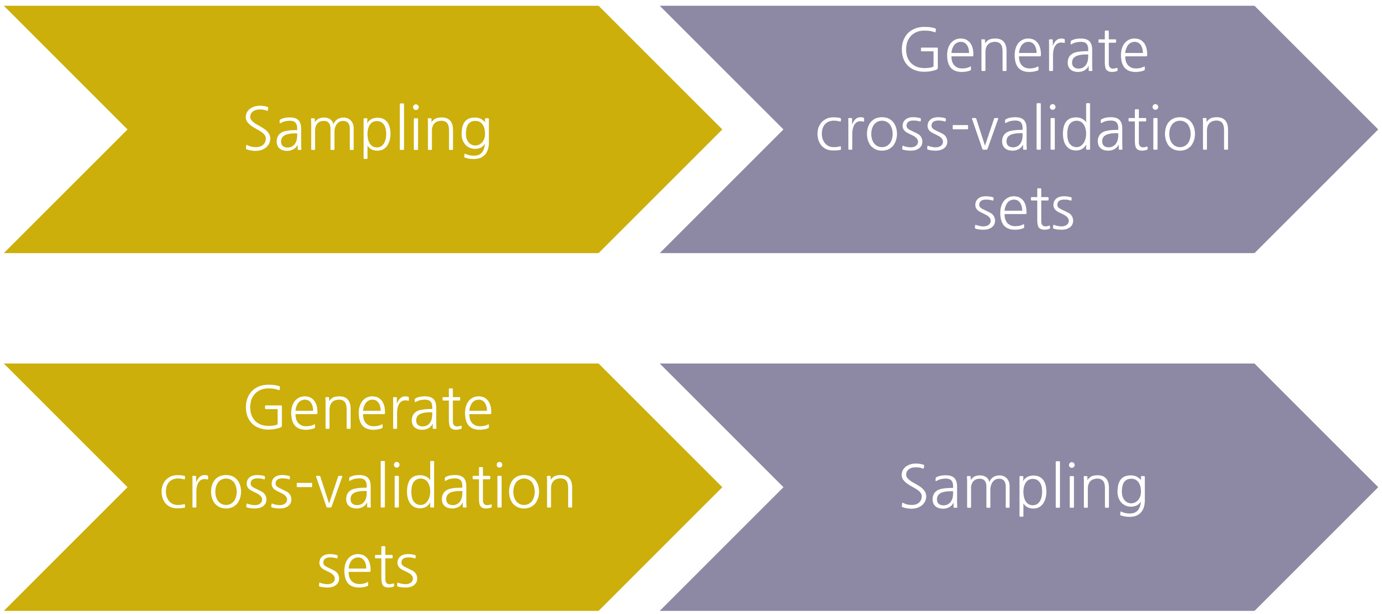
  - Apply logistic regression with $C = 1$ on the training set

- Question
  - Calculate evaluation metrics

The Issue on Cross-validation with Sampling

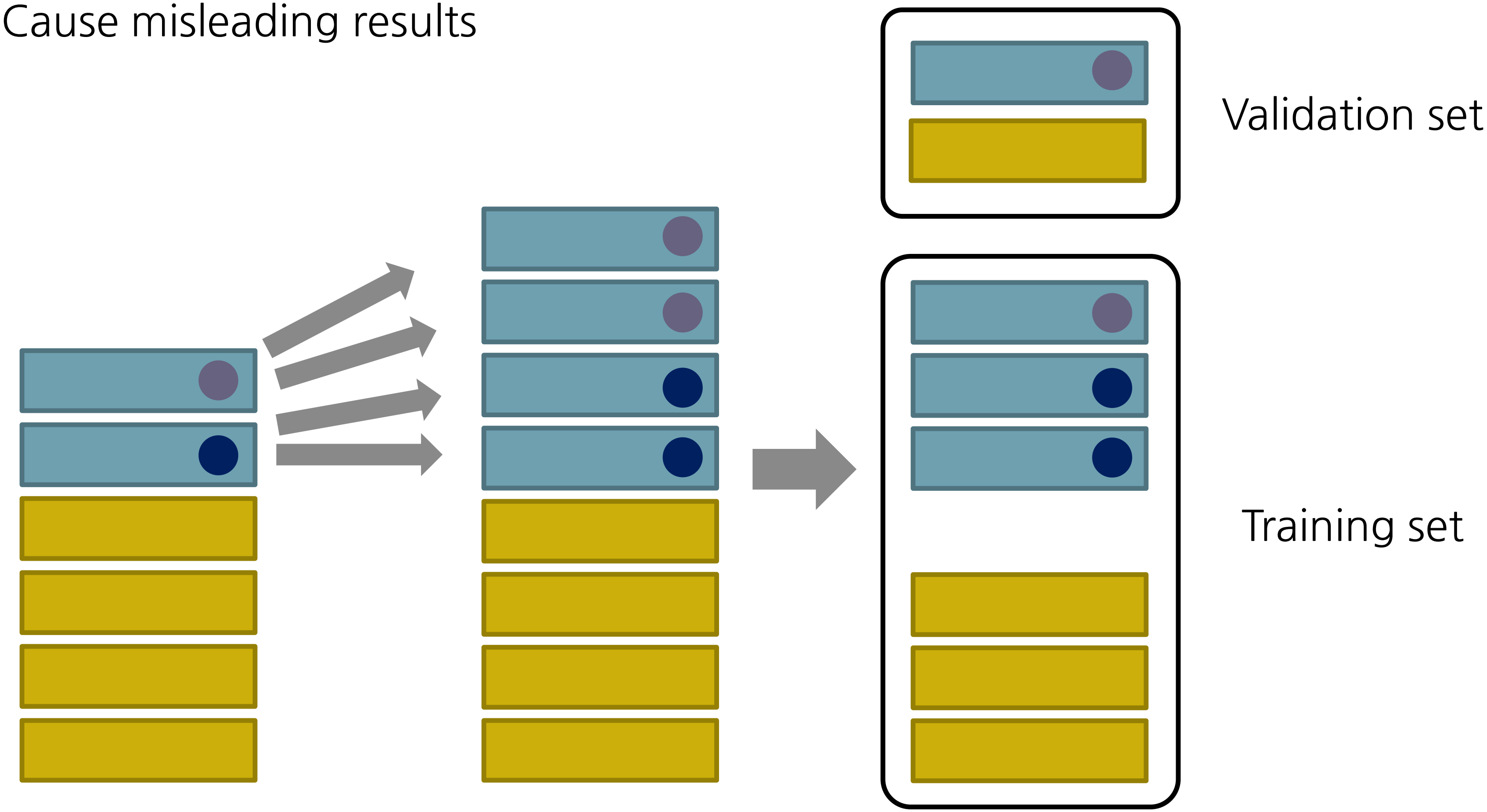
- For model selection, cross-validation might be applied

How about the order?



The Issue on Cross-validation

- In the case of over sampling, it might happen that validation set contains some of the same samples in training set
  - Cause misleading results

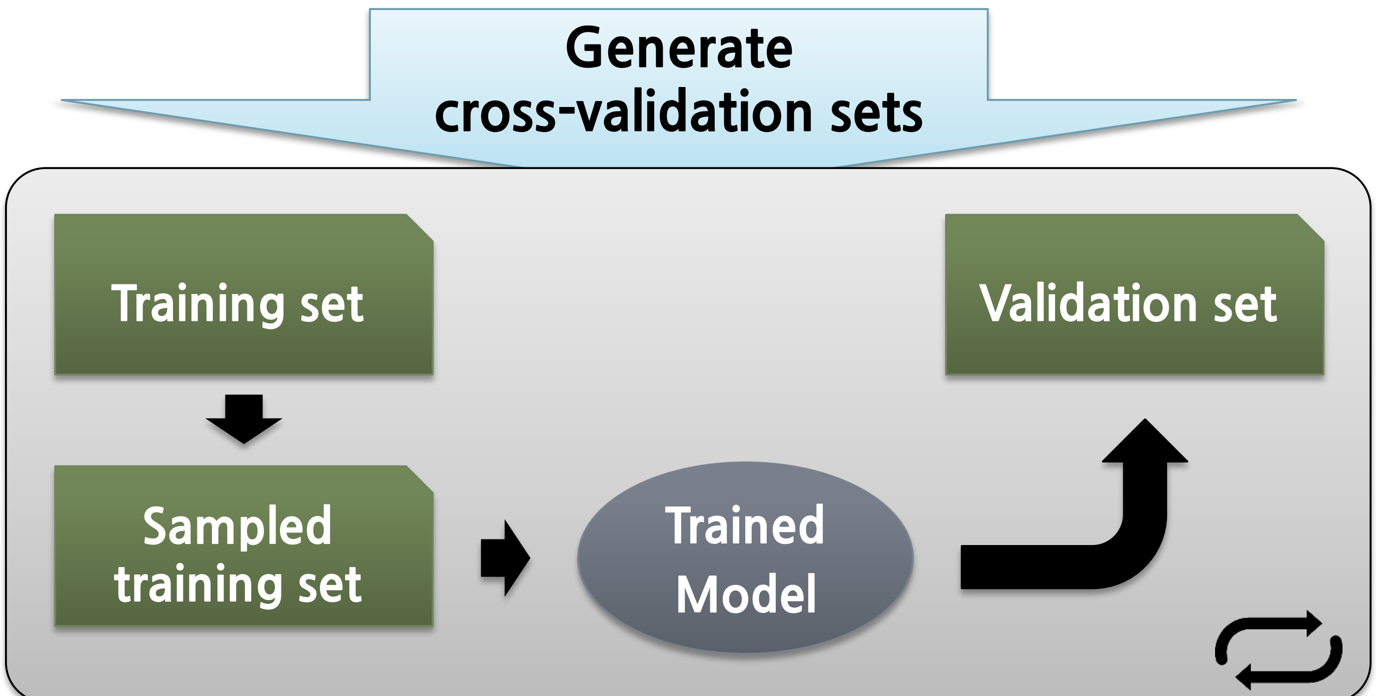


- The test set is biased from the original distribution

The Issue on Cross-validation

- The goal of cross validation is to define a dataset to "test" the model in the training phase
  - Generate validation set to estimate the general performance of models

- Sampling method aims to generate better training set to learn a better model for imbalance data



Apply Sampling Methods

- Test different sampling methods
  - Over sampling
  - Under sampling
  - SMOTE
  - ADASYN
  - NearMiss
  - Tomek link
  - One-sided selection

Apply Sampling Methods

- Test different sampling methods

```
from imblearn.over_sampling import RandomOverSampler, SMOTE,
ADASYN
from imblearn.under_sampling import RandomUnderSampler, NearMiss,
TomekLinks, OneSidedSelection

ros = RandomOverSampler(random_state=0, sampling_strategy='auto')
rus = RandomUnderSampler(random_state=0, sampling_strategy='auto')
sm = SMOTE(random_state=0,k_neighbors=5)
ada = ADASYN(random_state=0, n_neighbors=5)
nm1 = NearMiss(version=1)
nm2 = NearMiss(version=2)
nm3 = NearMiss(version=3)
tl = TomekLinks(sampling_strategy='auto')
oss = OneSidedSelection(random_state=0, n_neighbors=1, n_seeds_S=1)
```

Build Classifiers on Sampled Data

- Apply logistic regression on the sampled data

```
from sklearn.linear_model import LogisticRegression
clf=LogisticRegression(C=1)
clf.fit(trnX,trnY)
```

- Calculate evaluation metrics
  - Accuracy
  - Recall
  - Precision
  - F1

- Draw roc curves

Compare Sampling Methods

- Results

	Accuracy	Recall	Precision	F1
Original				
Over sampling				
Under sampling				
SMOTE				
ADASYN				
NearMiss-1				
NearMiss-2				
NearMiss-3				
Tomek links				
One-sided selection				

Penalized Models: Cost-sensitive Learning

- Cost-sensitive learning methods try to minimize the cost of misclassification
  - For imbalanced data, misclassification cost of the minor class is larger than that of the major class

		Real	
		Minor	Major
Model	Minor	0	C_1
	Major	C_2	0

    - $C_1 < C_2$

Cost-sensitive Learning

- Give more importance to certain classes (e.g. the minor class)
  - This function is usually implemented by changing weights of classes or weights of samples
    - Set large weights on the minor class

- Example) compare two classifiers with and without setting weights on classes
  - First, generate samples from two different distributions

```
import numpy as np

np.random.seed(0)
n_samples_1 = 1000
n_samples_2 = 100
X = np.r_[1.5*np.random.randn(n_samples_1, 2),
          0.5*np.random.randn(n_samples_2, 2) + [2, 2]]
y = [0]*(n_samples_1) + [1]*(n_samples_2)
```

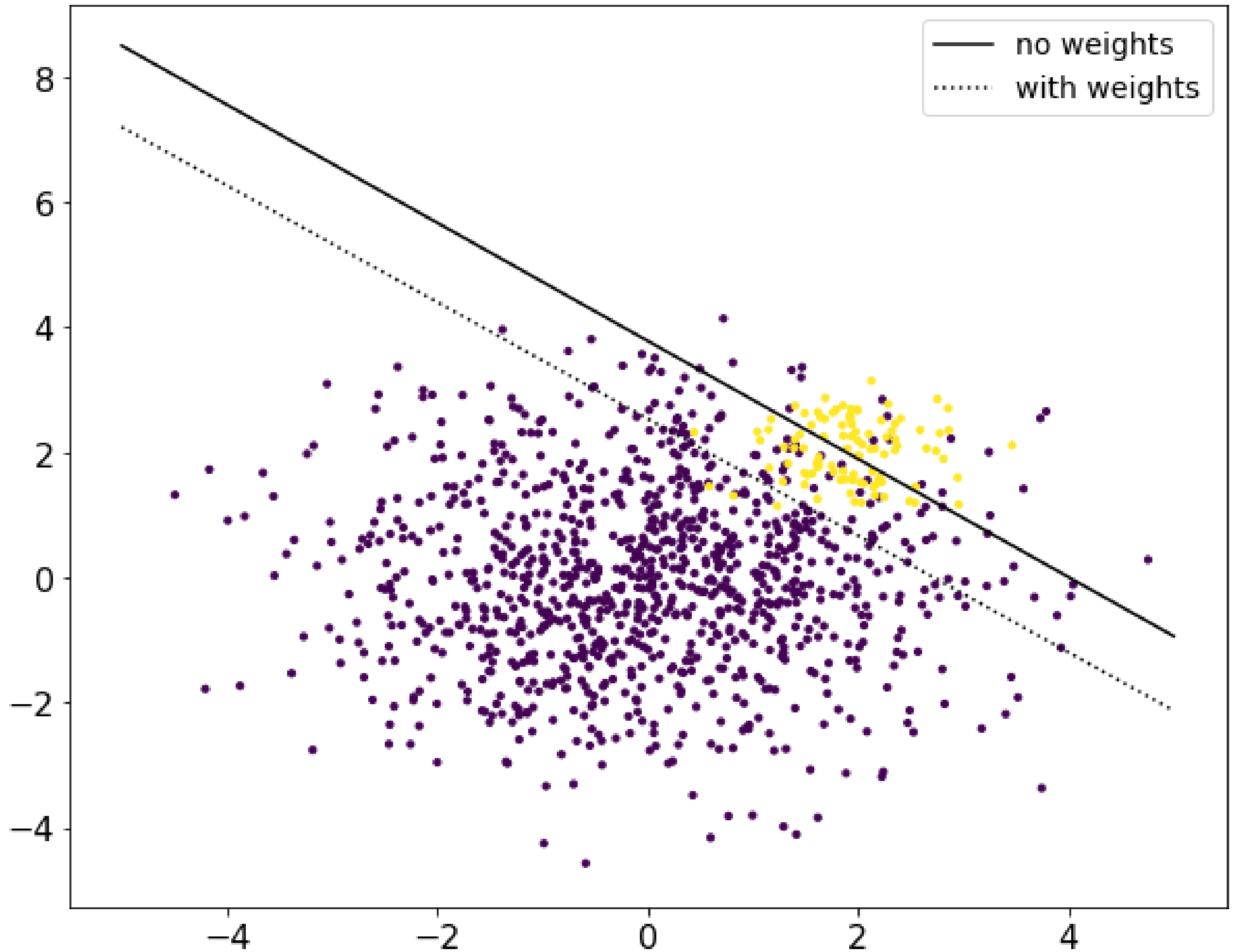
Cost-sensitive Learning

- Train support vector classifiers

```
clf=LogisticRegression(C=1)
clf.fit(X, y)

w = clf.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(-5, 5)
yy = a * xx - clf.intercept_[0] / w[1]

# get the separating hyperplane using weighted classes
wclf = svm.SVC(kernel='linear‘, class_weight={1: 10})
wclf.fit(X, y)

ww = wclf.coef_[0]
wa = -ww[0] / ww[1]
wyy = wa * xx - wclf.intercept_[0] / ww[1]
```

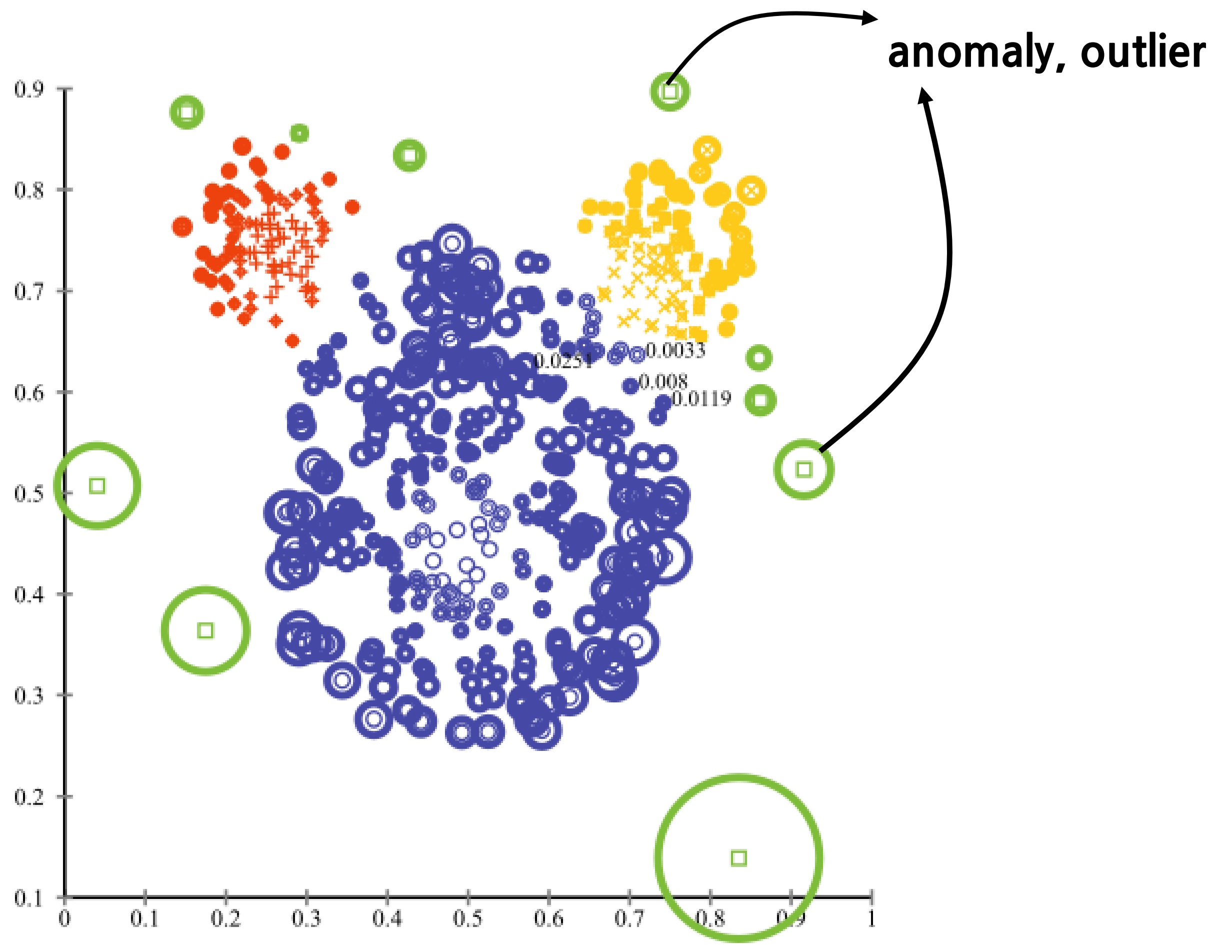
Cost-sensitive Learning

- Compare decision boundaries



Anomaly Detection

- Anomaly detection is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset



Presentation

3. Model Learning

- Create training dataset
  - Select explanatory variables

- Select the appropriate algorithms to learning models
  - Parameter selection

- Evaluate models
  - Model selection
  - Find the better way to improve the performance of the model

Next Week

4. Model Validation

- Try to make the better model
  - New explanatory variables?
  - Different preprocessing steps?
  - Different algorithms?