딥러닝/클라우드

Chapter 5

Decision Tree
Support Vector Machine

Sejong Oh
Bio Information Technology Lab.

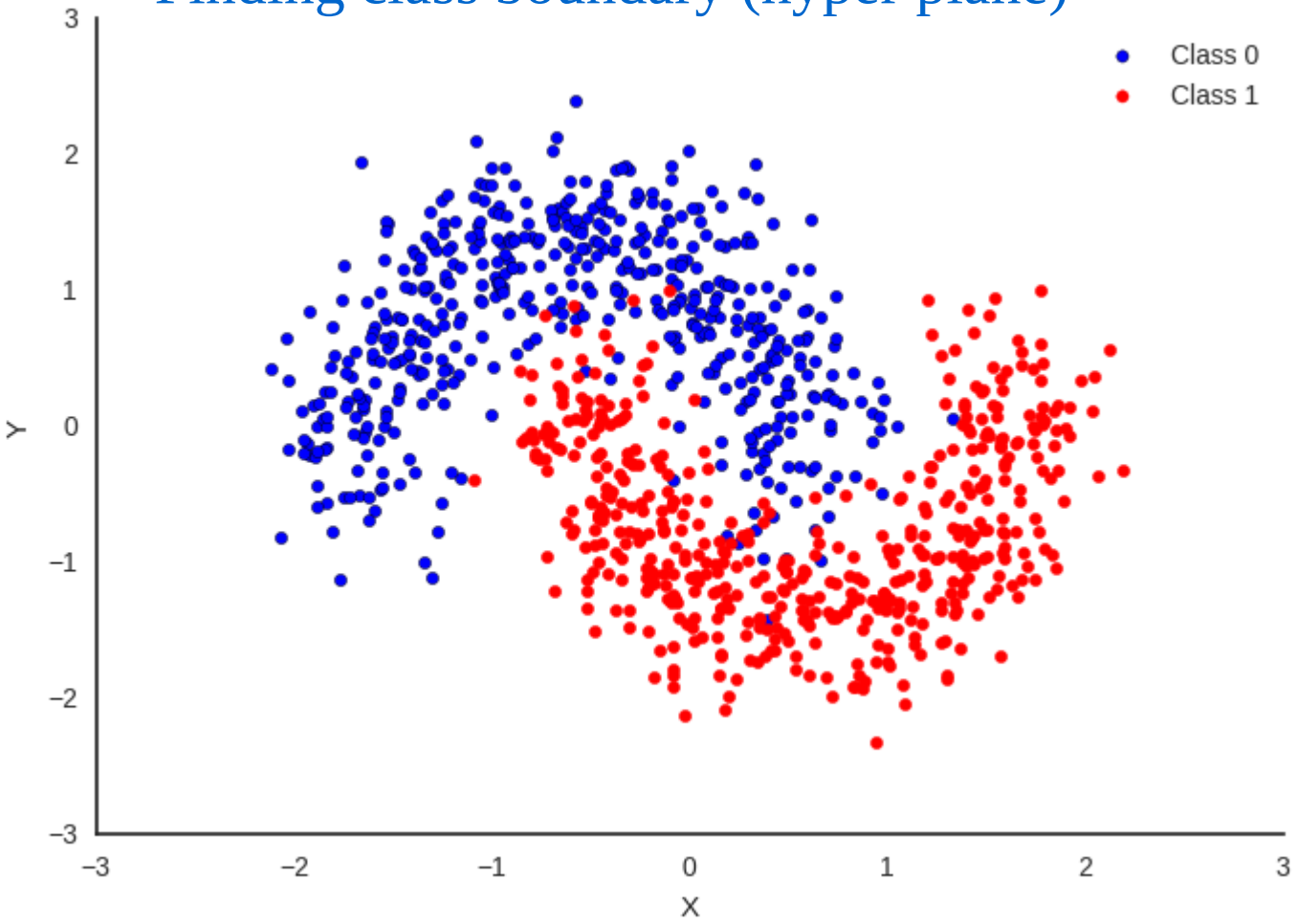
Contents

- Summary
- Concept of decision tree
- DecisionTreeClassifier (Scikit-learn)
- Support Vector Machine (SVM)
- Random forest
- xgboost

Summary

- Learning for classification is …

Finding class boundary (hyper plane)



https://www.kdnuggets.com/2017/04/must-know-evaluate-binary-classifier.html

Summary

- Type of classification algorithms

- Tree based
  - Decision Tree
  - Random forest
  - xgboost
- Neural net based
  - ANN
- etc
  - Logistic regression
  - Naïve Bayes
  - KNN
  - SVM

Summary

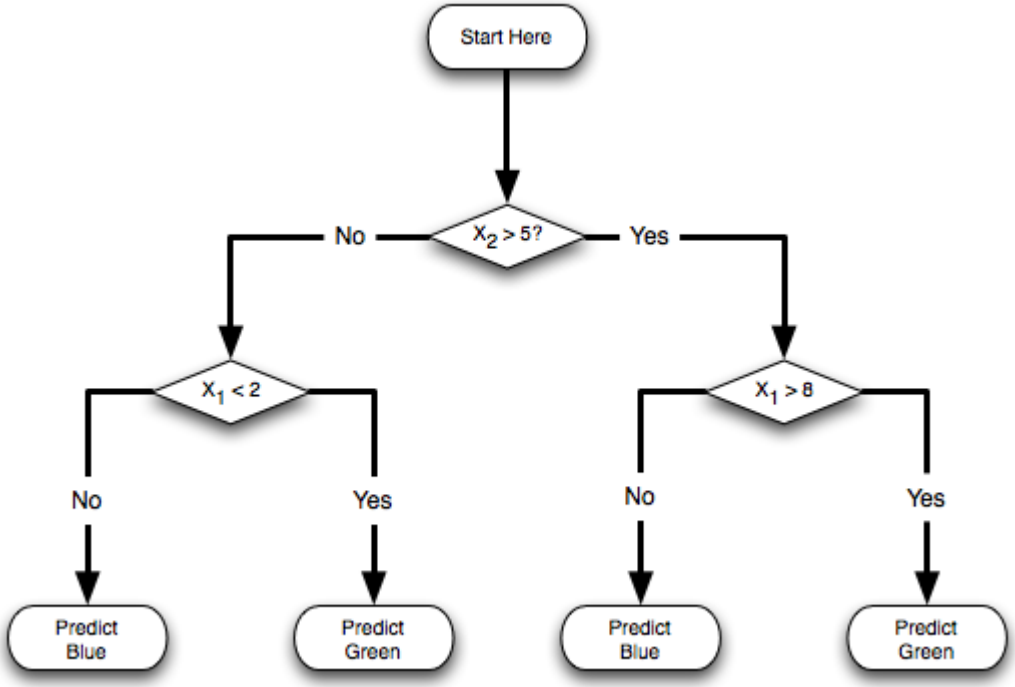
- Scikit-learn classifiers
  - Logistic regression
  - KNN
  - Support Vector Machine (SVM)
  - Naïve Bayes
  - Decision Tree
  - Random Forest
  - AdaBoost
- xgboost (needs install. Not in scikit-learn)

모든 데이터에 대해 최고의 성능을 보이는 classifier 는 없다
가급적 많은 classifier 를 테스트해 보고 그중에 좋은 것을 선택해야 함

주요 알고리즘들은 regression, classification 모두에 적용될 수 있다.

1. Concept of decision tree

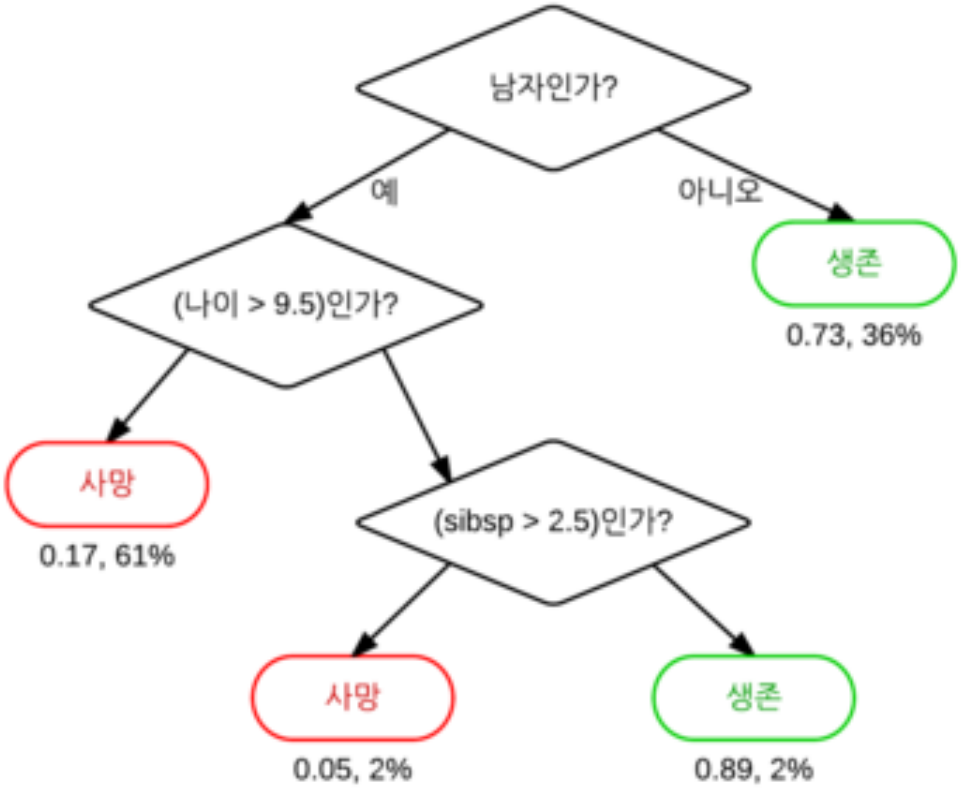
- Decision tree
  - 큰 문제를 작은 문제들의 조각으로 나누어 해결하는 기법
  - 인간의 의사결정 과정과 유사
  - 의사결정 또는 예측 결과의 의 근거가 명확히 제시되어야 하는 경우 많이 이용되는 학습 모델



x_1 과 x_2 의 값을 보고 색깔을 예측하는 결정 트리

1. Concept of decision tree

- 타이타닉호 탑승객의 생존 여부를 나타내는 결정 트리



("sibsp"는 탑승한 배우자와 자녀의 수를 의미한다.) 잎 아래의 숫자는 각각 생존 확률과 탑승객이 그 잎에 해당될 확률을 의미한다.

https://ko.wikipedia.org/wiki/%EA%B2%B0%EC%A0%95_%ED%8A%B8%EB%A6%AC_%ED%95%99%EC%8A%B5%EB%B2%95

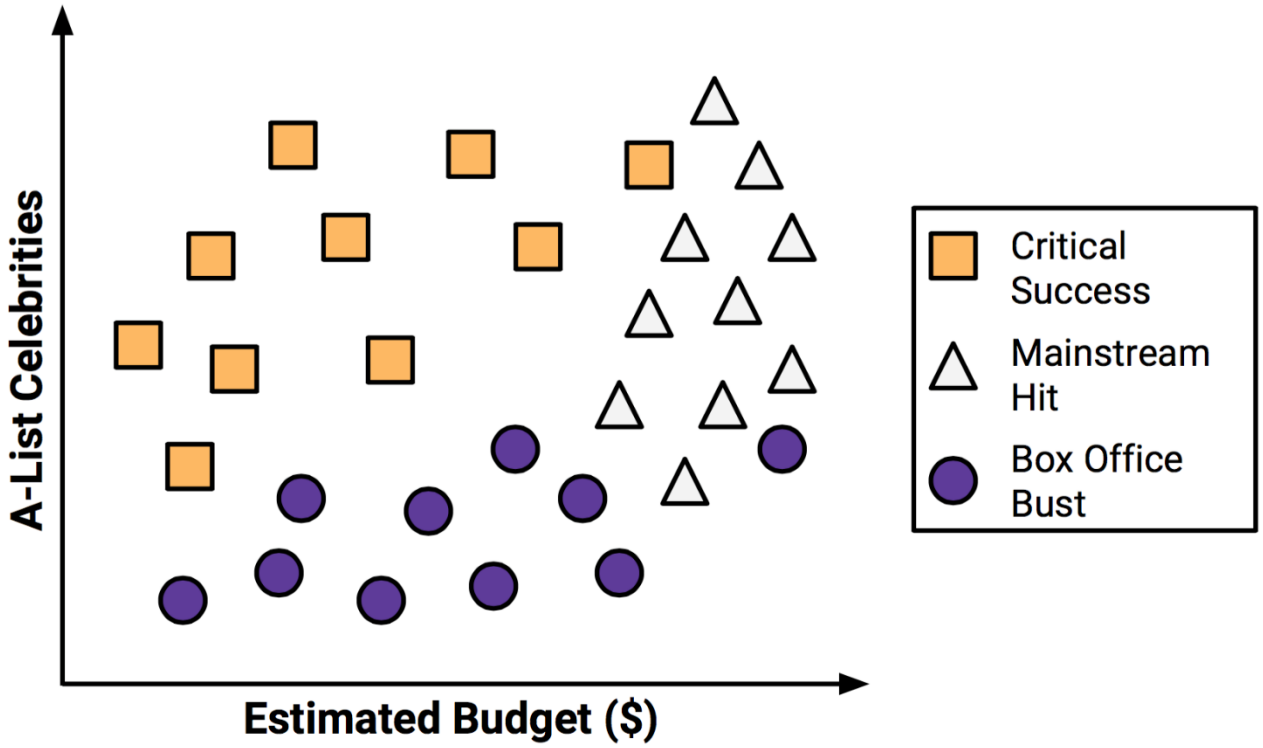
1. Concept of decision tree

- 알고리즘이 생성한 결정 트리 예제

```
Decision tree:

total_day_minutes > 264.4:
:...voice_mail_plan = yes:
:   :...international_plan = no: no (45/1)
:   :   international_plan = yes: yes (8/3)
:   voice_mail_plan = no:
:   :...total_eve_minutes > 187.7:
:       :...total_night_minutes > 126.9: yes (94/1)
:       :   total_night_minutes <= 126.9:
:       :   :...total_day_minutes <= 277: no (4)
:       :       total_day_minutes > 277: yes (3)
:       total_eve_minutes <= 187.7:
:       :...total_eve_charge <= 12.26: no (15/1)
:           total_eve_charge > 12.26:
:           :...total_day_minutes <= 277:
:               :...total_night_minutes <= 224.8: no (13)
:               :   total_night_minutes > 224.8: yes (5/1)
:               total_day_minutes > 277:
:               :...total_night_minutes > 151.9: yes (18)
:                   total_night_minutes <= 151.9:
:                   :...account_length <= 123: no (4)
:                       account_length > 123: yes (2)
total_day_minutes <= 264.4:
:...number_customer_service_calls > 3:
    :...total_day_minutes <= 160.2:
```

1. Concept of decision tree

- Decision Tree example
  - 영화 대본 및 기본 정보를 이용하여 영화가 흥행할지를 예측하는 모델을 만들고자 한다.
  - 사용하는 정보는 유명 배우의 수와 추정 제작비
  - 예측 결과는 "매우 흥행", "어느정도 흥행", "폭망"

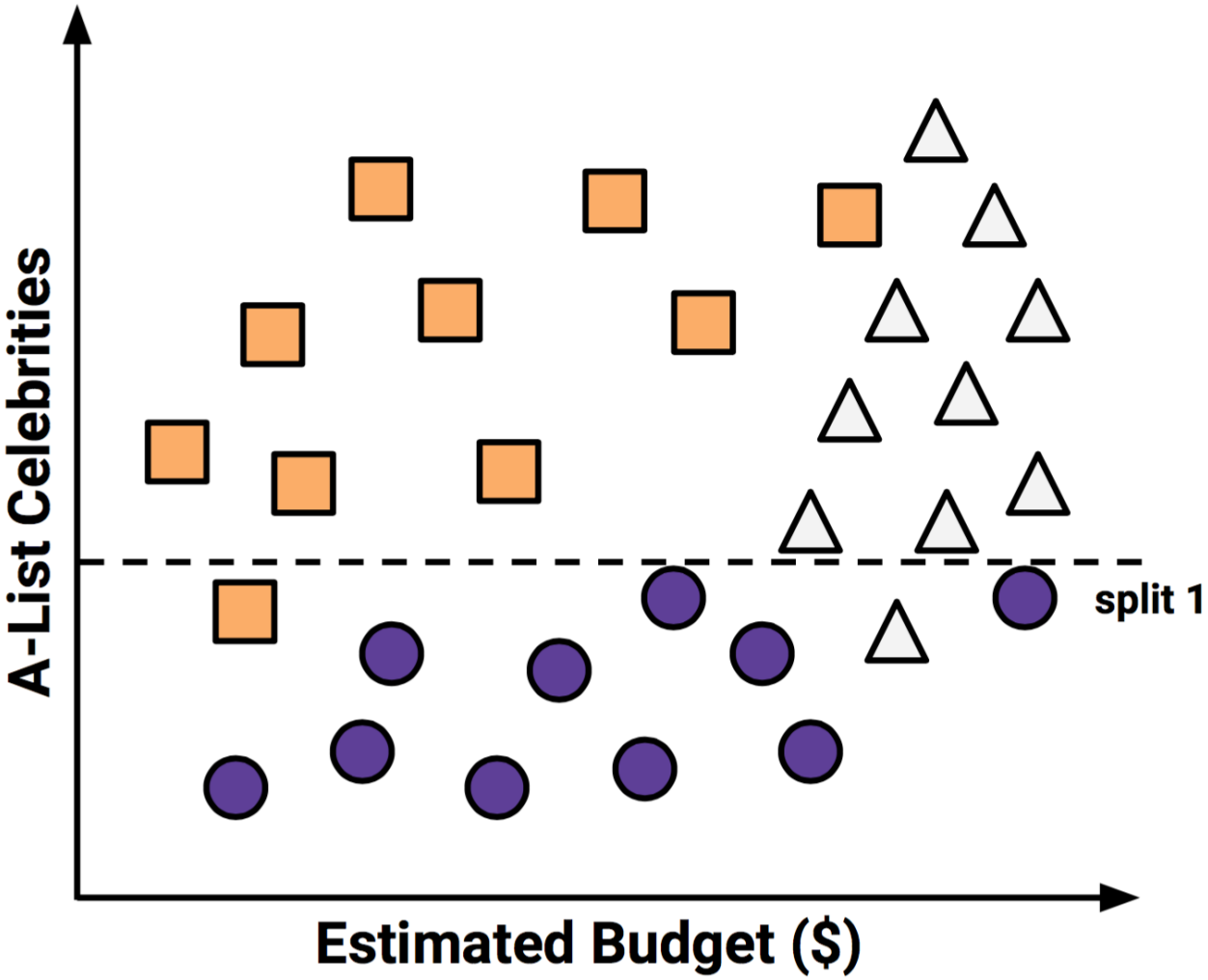


https://www.packtpub.com/books/content/divide-and-conquer-%E2%80%93-classification-using-decision-trees-and-rules

1. Concept of decision tree

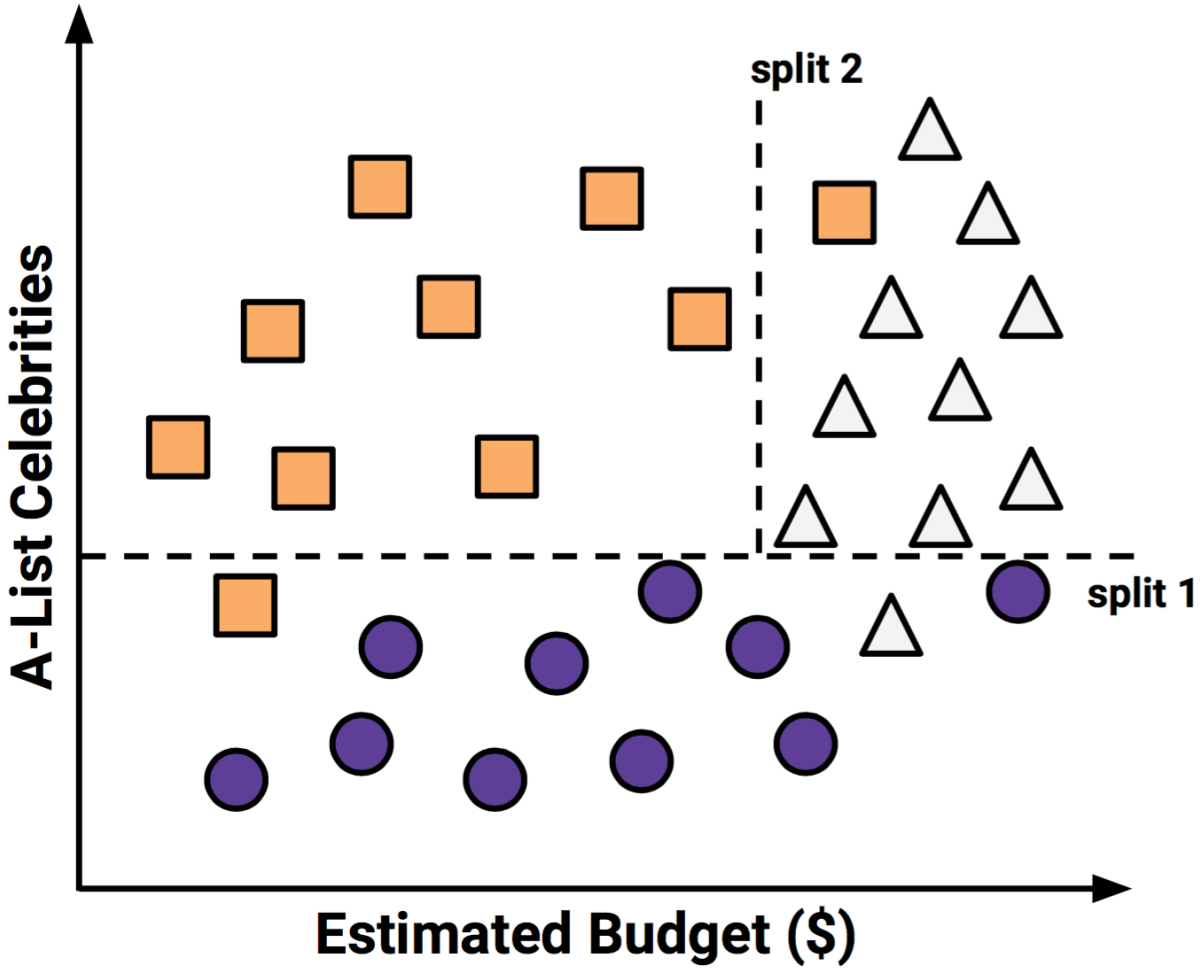
- Step 1

Impurity를 최소화 하는
Boundary를 찾는다

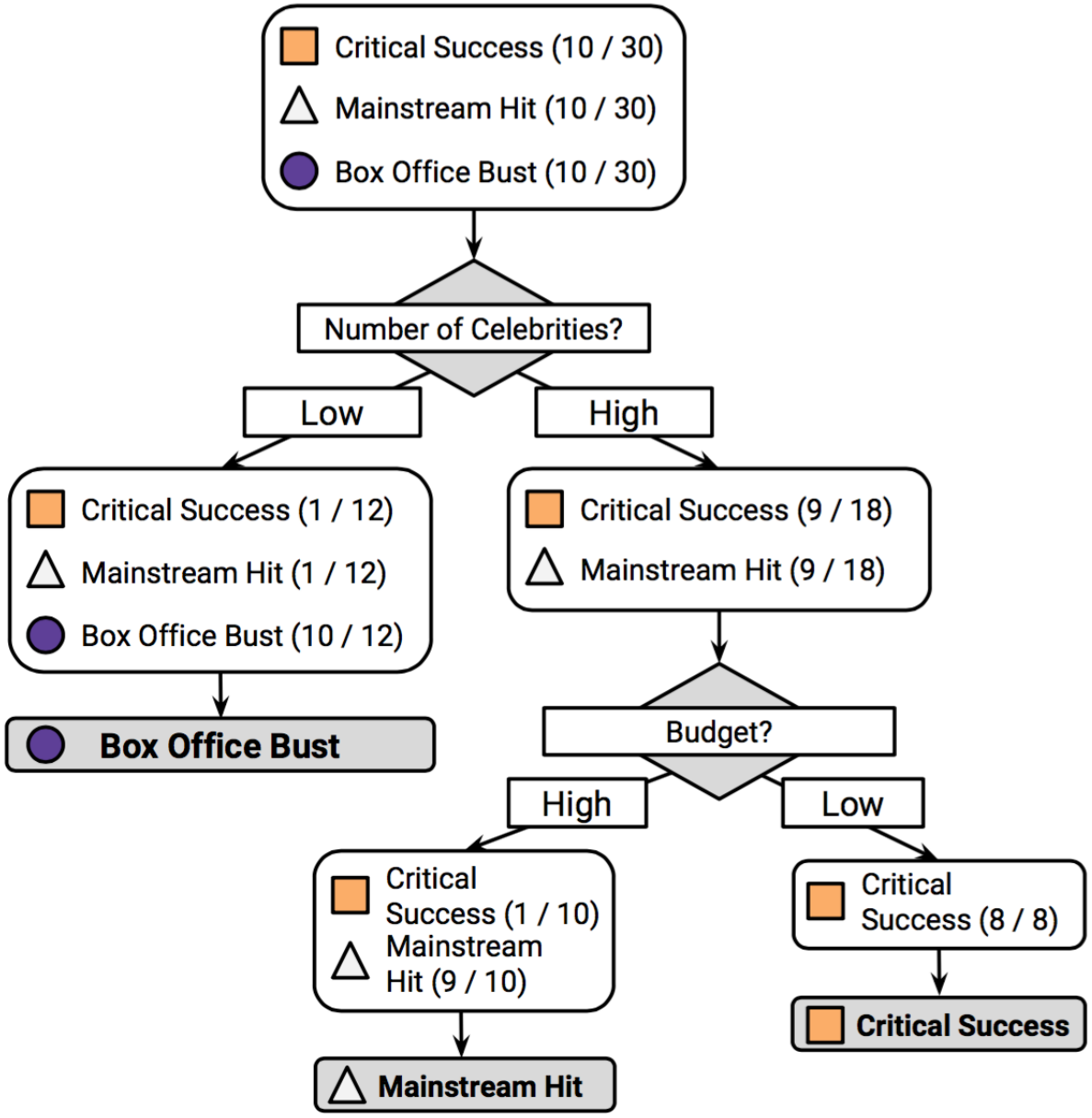


1. Concept of decision tree

- Step 2



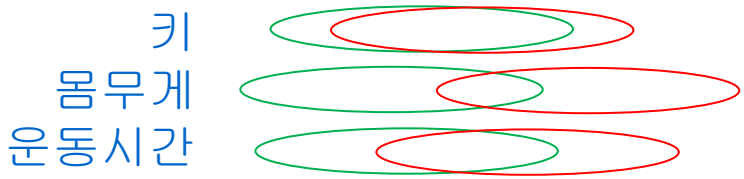
1. Concept of decision tree



1. Concept of decision tree

- Issues in decision tree

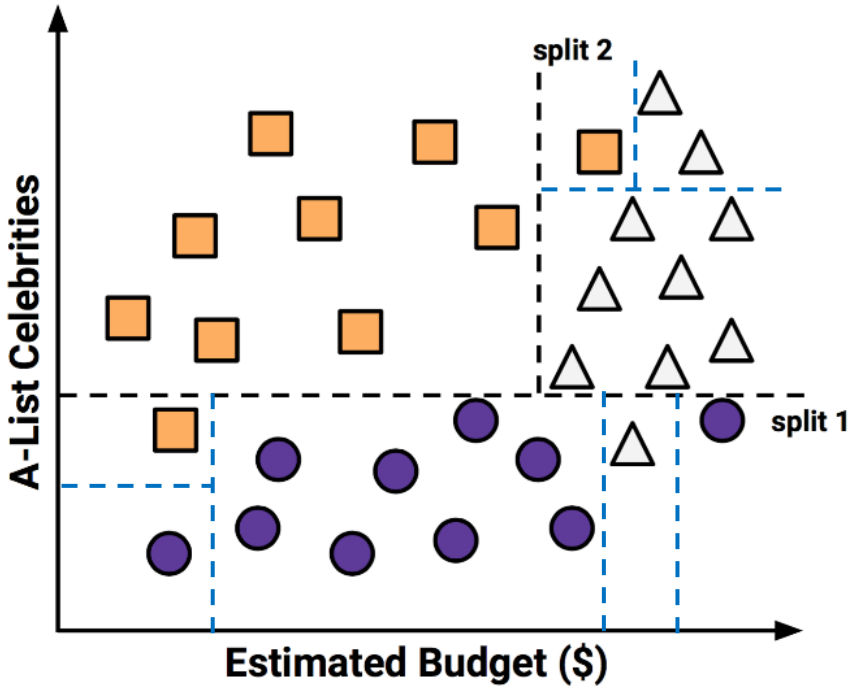
- (1) 트리의 node 를 선택 할 때 데이터셋에서 어떤 속성부터 선택할 것인가
  - 키, 몸무게, 운동시간 데이터를 가지고 고혈압 여부를 예측하는 모델을 만들고자 했을 때 root node 에는 어떤 속성이 있어야 하는가?
  - Feature evaluation 이 필요 (불순도(impurity) 계산)



불순도 계산 방법에는 여러가지가 있음

1. Concept of decision tree

- (2) 트리를 split 할 때 언제 중단할 것인가
  - 트리의 가지를 계속 뻗어 나가면 모든 instance를 100% 식별할 수 있는 tree를 만들 수 있다. => overfitting 발생
  - 따라서 적당할 때 트리 생성을 중단해야 한다 => 가지치기(pruning)

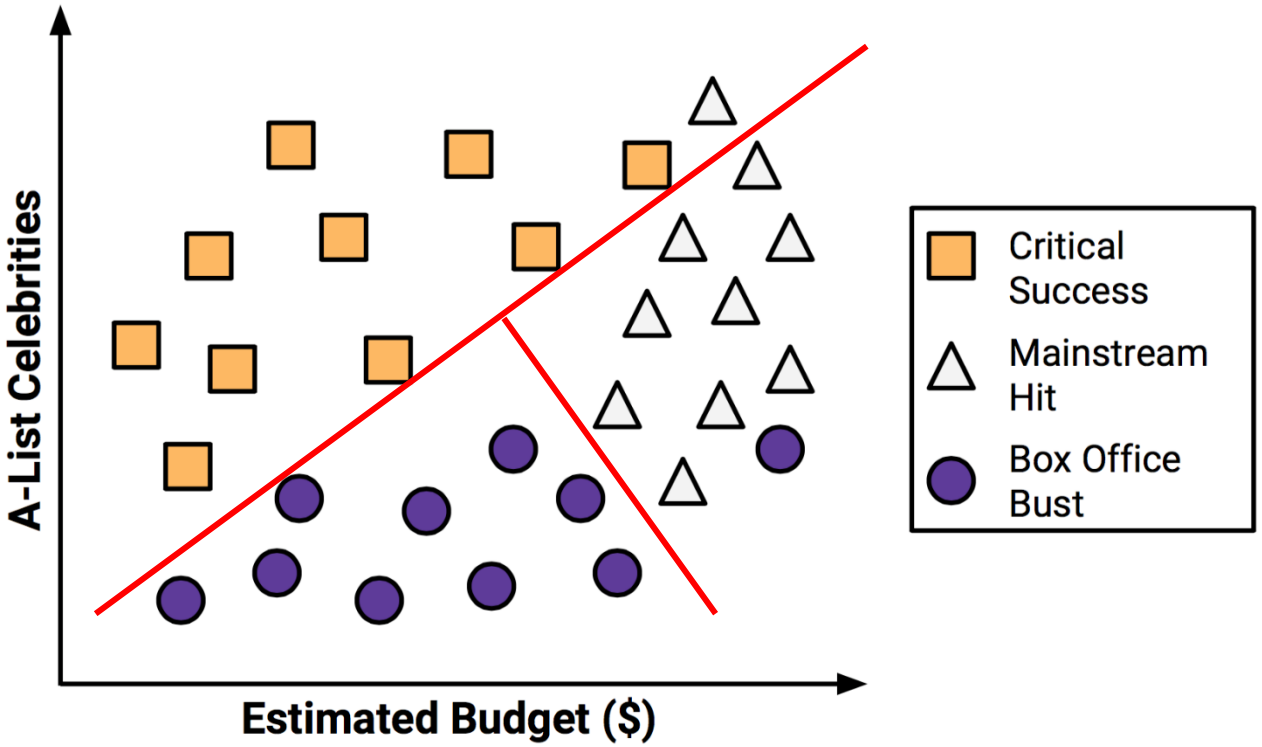


1. Concept of decision tree

- 장점
  - 모든 문제에 적합하다
  - 결측치, 범주형 속성, 수치 속성을 처리하기에 용이
  - 여러 속성중 중요한 속성들만을 사용하여 예측
  - 매우 많은 수 또는 상대적을 적은 훈련 데이터로도 모델 구축 가능
  - 수학적 배경이 없이도 해석 가능한 모델
  - 단순한 이론적 근거에 비해 높은 효율성
- 단점
  - 모델이 쉽게 과대적합 하거나 과소적합화 됨
  - 축에 평행한 구분선을 사용하기 때문에 일부 관계를 모델화 하는데 문제가 있음
  - 훈련 데이터에 대한 약간의 변경이 결정 논리에 큰 변화를 줌
  - 큰 트리는 이해하기 어렵고 직관적이지 않음

1. Concept of decision tree

- Decision tree 는 아래와 같은 구분선은 만들 수 없다 (아래 구분선은 의사결정 규칙으로 표현이 안된다)



1. Concept of decision tree

- Advanced decision tree algorithms
  - ID3 (Iterative Dichotomiser 3)
  - C4.5 (successor of ID3)
  - C5.0 (successor of C4.5)
  - CART (Classification And Regression Tree)
  - CHAID (CHi-squared Automatic Interaction Detector)
    - 이 알고리즘은 분류 트리를 계산할 때 다단계 분할을 수행
  - MARS (Multivariate adaptive regression splines)
    - 더 많은 수치 데이터를 처리하기 위해 결정 트리를 사용
  - 조건부 추론 트리 (Conditional Inference Trees)
    - 과적합을 피하기 위해 여러 테스트에 대해 보정 분할 기준으로 비 - 파라미터 테스트를 사용하는 통계 기반의 방법.
    - 이 방법은 편견 예측 선택 결과와 가지 치기가 필요하지 않다.

2. DecisionTreeClassifier

- sklearn.tree.DecisionTreeClassifier
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.**DecisionTreeClassifier**(*, *criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) [source]

- Dataset : liver.csv

2. DecisionTreeClassifier

- liver.csv
  - 간 장애(Liver Disorders) 자료
  - 레이블(category) + 혈액검사 결과(6개 변수)

```
category
mcv
alkphos
sgpt
sgot
gammagt
drinks
```

`category` : 클래스 정보. 0-정상, 1-간 장애 있음

2. DecisionTreeClassifier

05_decision_tree.py

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd

# prepare the iris dataset
df = pd.read_csv('D:/data/liver.csv')
print(df.head())
print(df.columns)   # column names

df_X = df.loc[:, df.columns != 'category']
df_y = df['category']
```

관행적으로 독립변수의 집합은 대문자 X로, 종속변수는 소문자 y 로 표기한다.

2. DecisionTreeClassifier

```
In [428]: print(df.head())
   category  mcv  alkphos  sgpt  sgot  gammagt  drinks
0         0   85       64    59    32       23     0.0
1         0   86       54    33    16       54     0.0
2         0   91       78    34    24       36     0.0
3         0   87       70    12    28       10     0.0
4         0   98       55    13    17       17     0.0

In [429]: print(df.columns)   # column names
Index(['category', 'mcv', 'alkphos', 'sgpt', 'sgot', 'gammagt', 'drinks'], dtype='object')
```

```
# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train_test_split(df_X, df_y, test_size=0.3,\
                     random_state=1234)
```

2. DecisionTreeClassifier

```
# Define learning model (basic)
model =  DecisionTreeClassifier(random_state=1234)

# Train the model using the training sets
model.fit(train_X, train_y)

# performance evaluation
print('Train accuracy :', model.score(train_X, train_y))
print('Test accuracy :', model.score(test_X, test_y))
```

```
In [431]: print('Train accuracy :',model.score(train_X, train_y))
Train accuracy : 1.0

In [432]: print('Test accuracy :',model.score(test_X, test_y))
Test accuracy : 0.6442307692307693
```

과적합(overfitting)

2. DecisionTreeClassifier

```
# Define learning model (tuning)
pred_y = model.predict(test_X)
confusion_matrix(test_y, pred_y)
```

```
In [439]: ped_y = model.predict(test_X)
     ...: confusion_matrix(test_y, pred_y)
Out[439]:
array([[41, 23],
       [14, 26]], dtype=int64)
```

Confusion matrix

test_y (actual) \ pred_y	0	1
0	41	23
1	14	26

2. DecisionTreeClassifier

```
# Define learning model (tuning)
model =  DecisionTreeClassifier(max_depth=4, random_state=1234)

# Train the model using the training sets
model.fit(train_X, train_y)

# performance evaluation
print('Train accuracy :', model.score(train_X, train_y))
print('Test accuracy :', model.score(test_X, test_y))
```

조율모수 (tuning/hyper parameter)

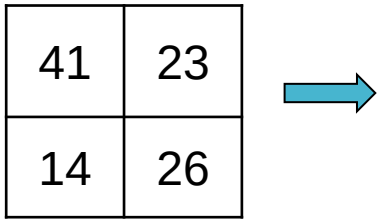
```
In [434]: print('Train accuracy :',model.score(train_X, train_y))
Train accuracy : 0.8008298755186722

In [435]: print('Test accuracy :',model.score(test_X, test_y))
Test accuracy : 0.6923076923076923
```

과적합 가능성

2. DecisionTreeClassifier

```
# Define learning model (tuning)
pred_y = model.predict(test_X)
confusion_matrix(test_y, pred_y)
```

```
In [455]: pred_y = model.predict(test_X)
     ...: confusion_matrix(test_y, pred_y)
Out[455]:
array([[49, 15],
       [17, 23]], dtype=int64)
```

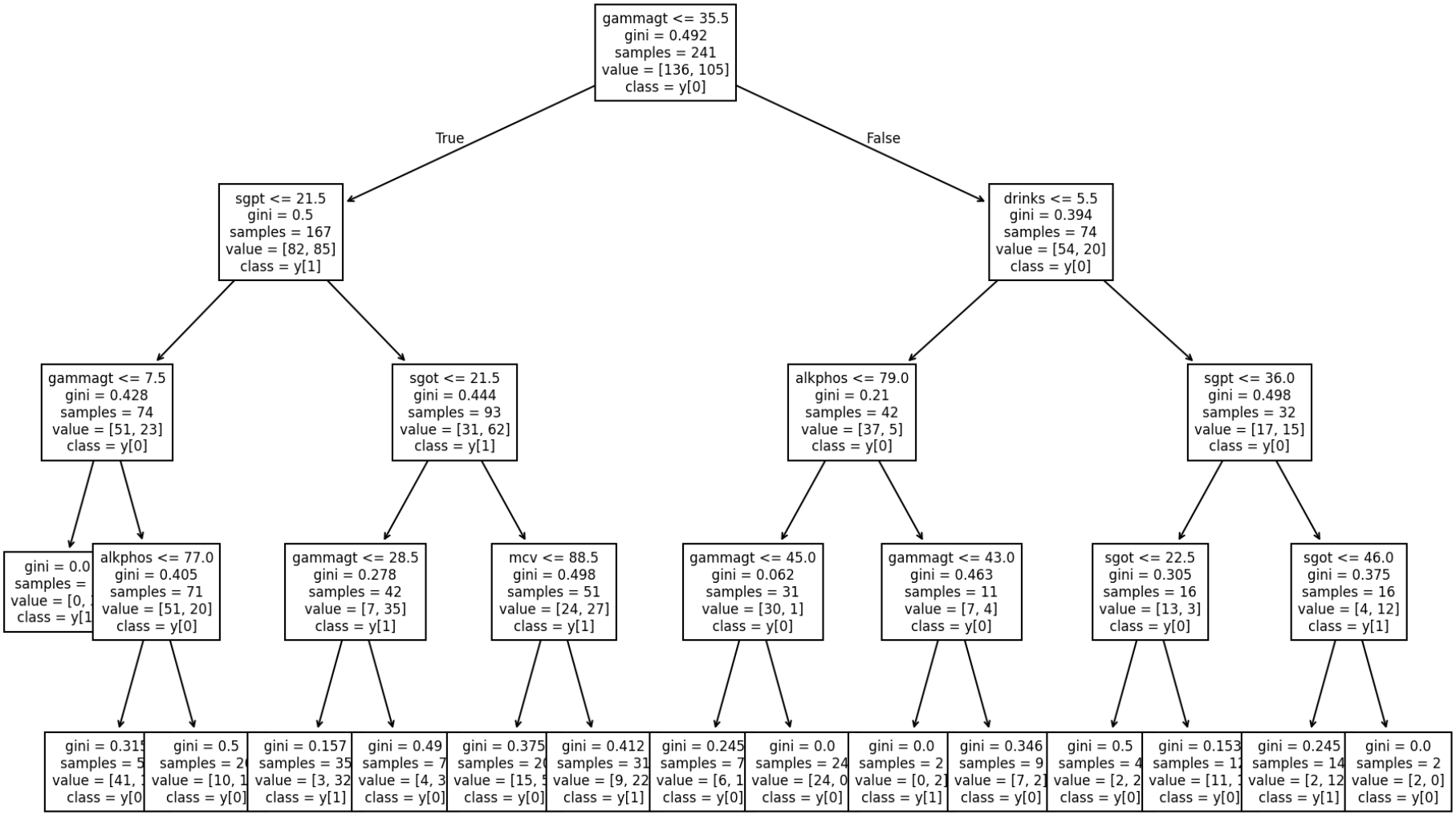


Confusion matrix

test_y (actual) \ pred_y	0	1
0	49	15
1	17	23

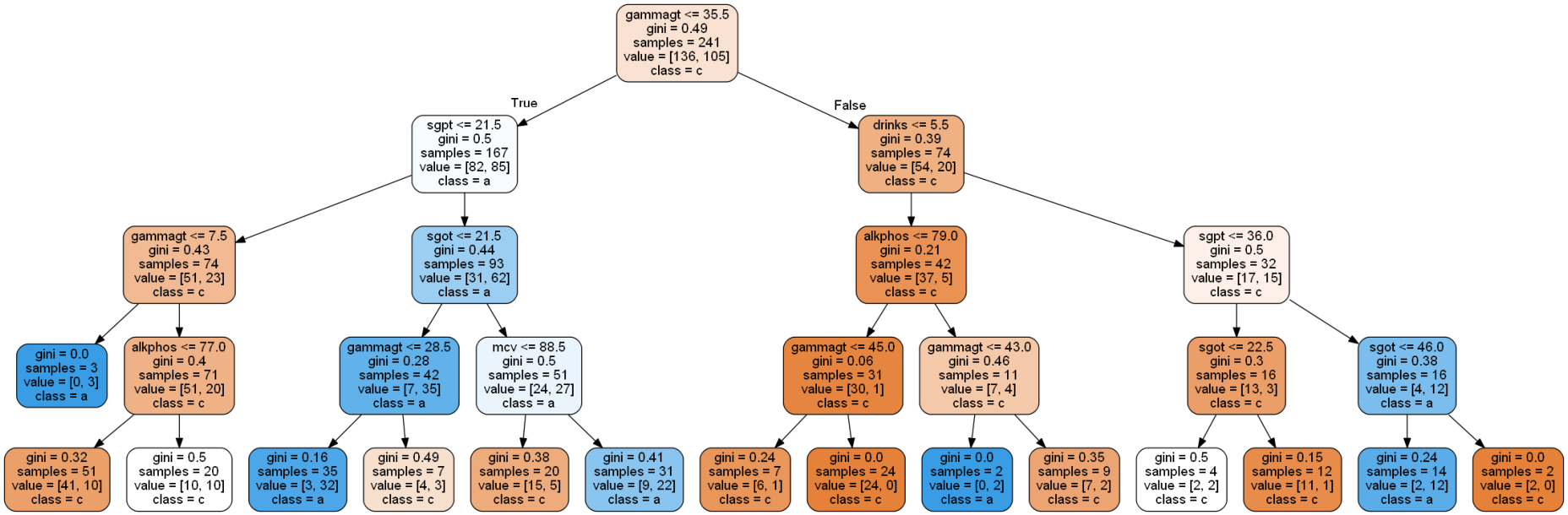
2. DecisionTreeClassifier

```
# visualize decision tree

from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

plot_tree(model,
          fontsize=8,
          feature_names=df_X.columns.to_list(),
          class_names=True)
plt.show()
```


gammagt <= 35.5
gini = 0.492
samples = 241
value = [136, 105]
class = y[0]
True
False
sgpt <= 21.5
gini = 0.5
samples = 167
value = [82, 85]
class = y[1]
drinks <= 5.5
gini = 0.394
samples = 74
value = [54, 20]
class = y[0]
gammagt <= 7.5
gini = 0.428
samples = 74
value = [51, 23]
class = y[0]
sgot <= 21.5
gini = 0.444
samples = 93
value = [31, 62]
class = y[1]
alkphos <= 79.0
gini = 0.21
samples = 42
value = [37, 5]
class = y[0]
sgpt <= 36.0
gini = 0.498
samples = 32
value = [17, 15]
class = y[0]
alkphos <= 77.0
gini = 0.405
samples = 71
value = [51, 20]
class = y[0]
gammagt <= 28.5
gini = 0.278
samples = 42
value = [7, 35]
class = y[1]
mcv <= 88.5
gini = 0.498
samples = 51
value = [24, 27]
class = y[1]
gammagt <= 45.0
gini = 0.062
samples = 31
value = [30, 1]
class = y[0]
gammagt <= 43.0
gini = 0.463
samples = 11
value = [7, 4]
class = y[0]
sgot <= 22.5
gini = 0.305
samples = 16
value = [13, 3]
class = y[0]
sgot <= 46.0
gini = 0.375
samples = 16
value = [4, 12]
class = y[1]

2. DecisionTreeClassifier

- Note.
  - GraphViz 를 이용하여 시각화 할 수도 있다



2. DecisionTreeClassifier

Hyper parameters

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- http://june3471.pythonanywhere.com/pages/deeplearning/26/

criterion : String, optional (default = “gini”)

Decision Tree의 가지를 분리 할 때 어떤 기준으로 정보 획득량을 계산하고 가지를 분리를 할 것인지를 정하는 Parameter이다. “gini”와 “entropy” 총 두 가지가 있으며, 기본 값은 gini이다.

- “gini”는 “entropy”보다 연산속도는 빠르지만 한쪽으로 편향된 결과를 낼 수 있다.
- “Entropy”는 “gini”에 비해 조금 더 균형 잡힌 model을 만들 수 있다고 한다.

max_depth : int or None, optional (default = None)

Decision Tree의 최대 깊이 제한을 줄 수 있는 Parameter이다. 기본 값은 None이며 None일 때는 모든 잎이 min_sample_split보다 작거나 불순도가 0일 때까지 노드가 확장된다. max_depth를 활용해 사전가지치기를 하고 overfitting을 방지할 수 있다.

min_samples_split : int, float optional (default = 2)

노드에서 가지를 분리할 때 필요한 최소 sample 개수에 대한 제한을 줄 수 있는 Parameter이다. Parameter의 주어진 값에 type에 따라 다음과 같이 기능한다.

- int일 경우, 주어진 값을 그대로 사용한다.
- float일 경우, 0에서 1사이의 값을 줄 수 있으며 ceil(전체 데이터 수*min_sample_split)의 값을 사용한다.

2. DecisionTreeClassifier

min_samples_leaf : int, float optional (default = 2)

한 노드에서 가지고 있어야 할 최소 sample 개수에 대한 제한을 줄 수 있는 Parameter이다. Parameter의 주어진 값에 type에 따라 다음과 같이 기능한다.

- int일 경우, 주어진 값을 그대로 사용한다.
- float일 경우, 0에서 1사이의 값을 줄 수 있으며 ceil(전체 데이터 수*min_samples_leaf)의 값을 사용한다.

max_features : int, float, string or None, optional (default=None)

Decision Tree model을 만들 때 사용할 수 있는 변수의 개수를 제한을 줄 수 있는 Parameter이다.

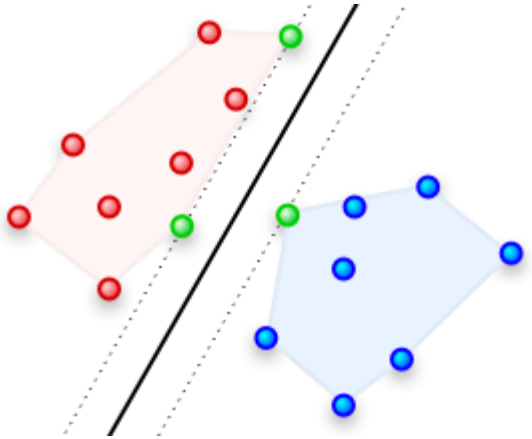
- int일 경우, 주어진 값을 그대로 사용한다.
- float일 경우, int(max_features * 총 변수 개수)를 사용한다.
- ~~"auto"~~,"sqrt" 일 경우, 를 사용한다. sqrt(변수의 개수)
- "log2"일 경우, 를 사용한다. $\log_2$변수의 개수
- None일 경우, 총 변수 개수를 사용한다.

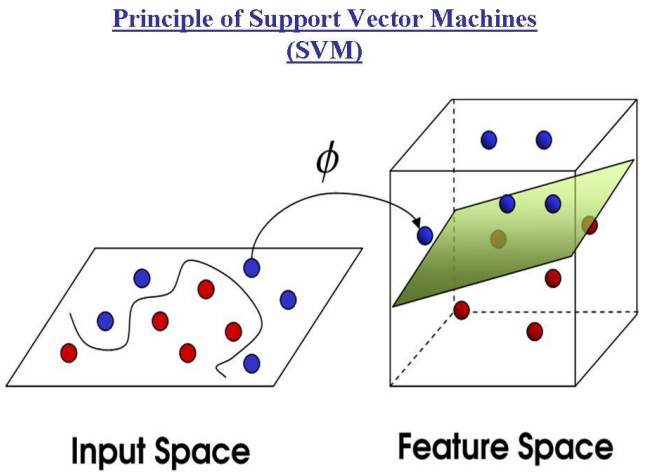
class_weight : *dict, list of dict or "balanced", default=None*

Weights associated with classes in the form `{class_label: weight}`. If None, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

3. Support Vector Machine

- Introduction
  - One of classification algorithm
  - Original SVM can be applied on two class problem
    - Extend to multi-class problem
  - SVM is basically linear discrimination method
  - It has two ideas
    - Finding maximum-margin hyperplane
    - Move input data onto higher dimension space (kernel method)

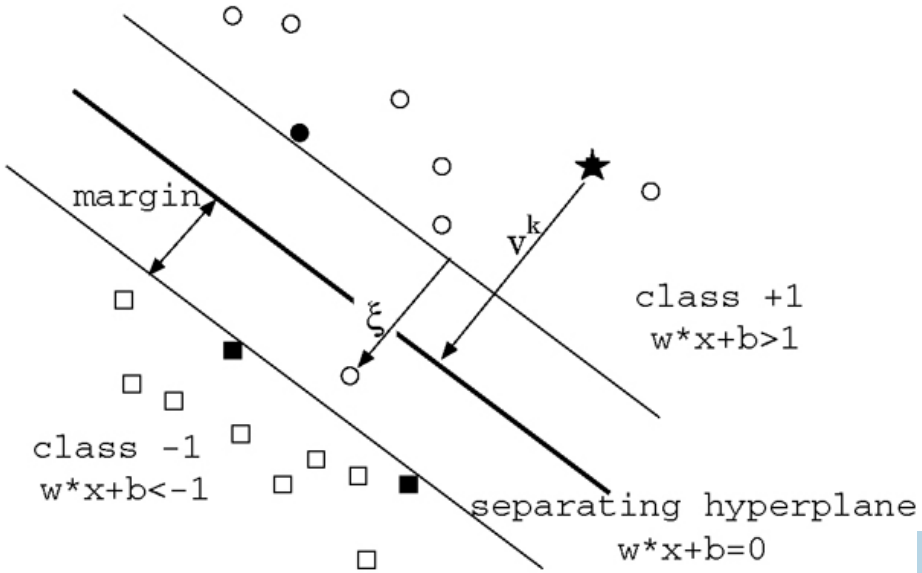




연두색 점 : support vector

3. Support Vector Machine

- Support vectors
  - The hypothesis space is given by the functions f(x) = sng(*w*x + *b*), where *w* and *b* are parameters that are learned
  - Samples on the margin are the *support vectors*
  - Optimization problem: the norm of w and loss are minimized subject to right classification within the allowed error(s), ξ (soft margin)
  - Writing the classification rule in its unconstrained dual form reveals that classification is only a function of the support vectors



3. Support Vector Machine

sklearn.svm: Support Vector Machines

The sklearn.svm module includes Support Vector Machine algorithms.

User guide: See the Support Vector Machines section for further details.

Estimators

svm.LinearSVC([penalty, loss, dual, tol, C, ...])	Linear Support Vector Classification.
svm.LinearSVR(*[, epsilon, tol, C, loss, ...])	Linear Support Vector Regression.
svm.NuSVC(*[, nu, kernel, degree, gamma, ...])	Nu-Support Vector Classification.
svm.NuSVR(*[, nu, C, kernel, degree, gamma, ...])	Nu Support Vector Regression.
svm.OneClassSVM(*[, kernel, degree, gamma, ...])	Unsupervised Outlier Detection.
svm.SVC(*[, C, kernel, degree, gamma, ...])	C-Support Vector Classification.
svm.SVR(*[, kernel, degree, gamma, coef0, ...])	Epsilon-Support Vector Regression.

svm.l1_min_c(X, y, *[, loss, fit_intercept, ...])	Return the lowest bound for C such that for C in (l1_min_C, infinity) the model is guaranteed not to be empty.

- https://scikit-learn.org/stable/modules/classes.html#module-sklearn.svm

3. Support Vector Machine

- C-Support Vector Classification. (sklearn.svm.SVC)

  - The implementation is based on libsvm.
  - The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples.
  - For large datasets consider using sklearn.svm.LinearSVC or sklearn.linear_model.SGDClassifier instead, possibly after a sklearn.kernel_approximation. Nystroem transformer.

3. Support Vector Machine

05_svm.py

```
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
import pydot

# prepare the iris dataset
df = pd.read_csv('D:/data/liver.csv')

df_X = df.loc[:, df.columns != 'category']
df_y = df['category']

# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train_test_split(df_X, df_y, test_size=0.3,\
                     random_state=1234)
```

3. Support Vector Machine

```
# Define learning model (basic)
#########################################
model = svm.SVC()
# Train the model using the training sets
model.fit(train_X, train_y)

# performance evaluation
print('Train accuracy :', model.score(train_X, train_y))
print('Test accuracy :', model.score(test_X, test_y))

pred_y = model.predict(test_X)
confusion_matrix(test_y, pred_y)
```

```
Train accuracy : 0.7178423236514523
Test accuracy : 0.7788461538461539
Out[2]:
array([[57,  7],
       [16, 24]], dtype=int64)
```

3. Support Vector Machine

```
# Define learning model (poly kernel)  #############
model = svm.SVC(kernel='poly')
# Train the model using the training sets
model.fit(train_X, train_y)

# performance evaluation
print('Train accuracy :', model.score(train_X, train_y))
print('Test accuracy :', model.score(test_X, test_y))

pred_y = model.predict(test_X)
confusion_matrix(test_y, pred_y)
```

```
Train accuracy : 0.7385892116182573
Test accuracy : 0.7596153846153846
Out[3]:
array([[50, 14],
       [11, 29]], dtype=int64)
```

3. Support Vector Machine

Hyper parameters

C : *float, default=1.0*

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'*

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, default=3*

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

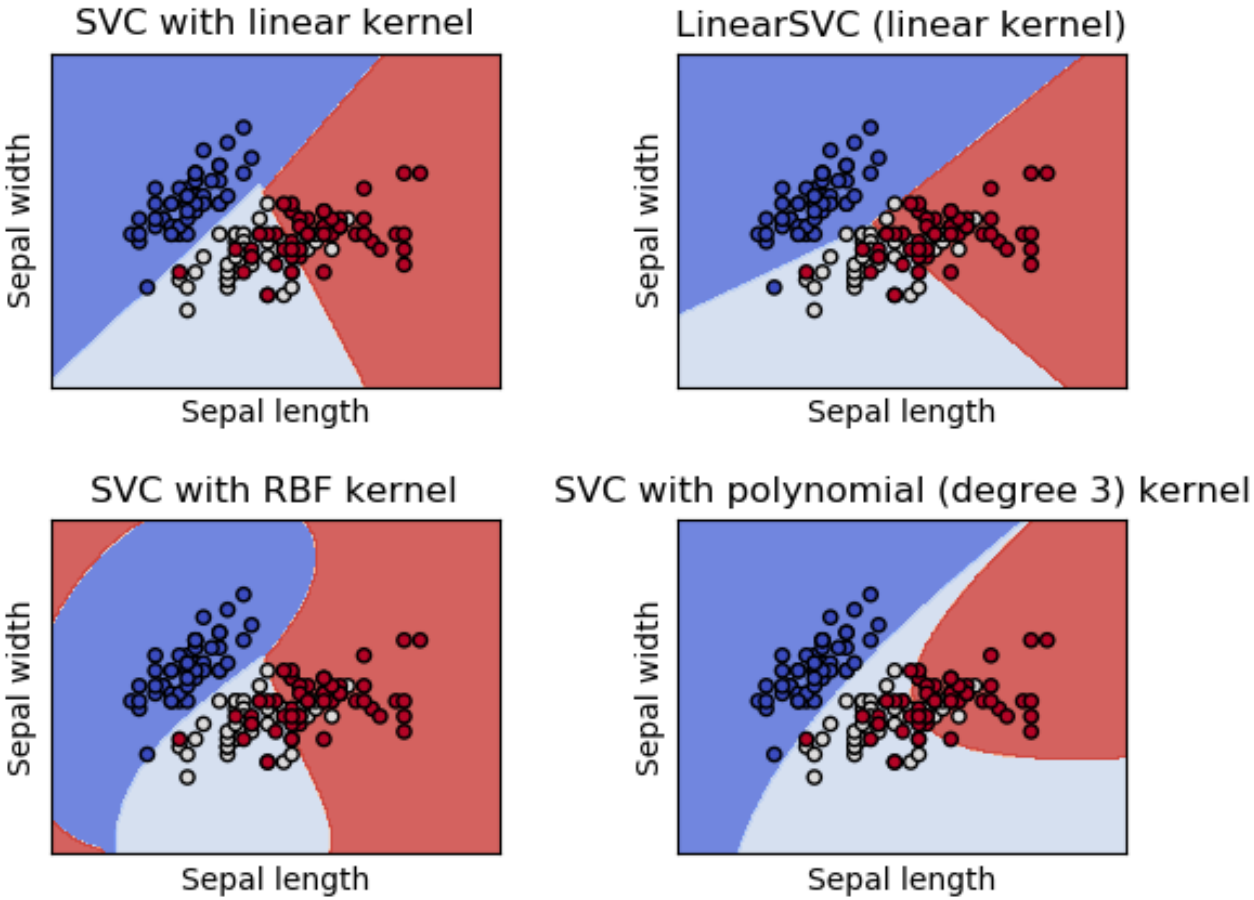
gamma : *{'scale', 'auto'} or float, default='scale'*

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Changed in version 0.22: The default value of `gamma` changed from 'auto' to 'scale'.

3. Support Vector Machine



https://scikit-learn.org/stable/modules/svm.html#classification

3. Support Vector Machine

- Advantage
  - 범주나 수치데이터 모두에 사용가능
  - 노이즈 데이터에 영향을 크게 받지 않고 overfitting 이 잘 일어나지 않음
  - 높은 정확도

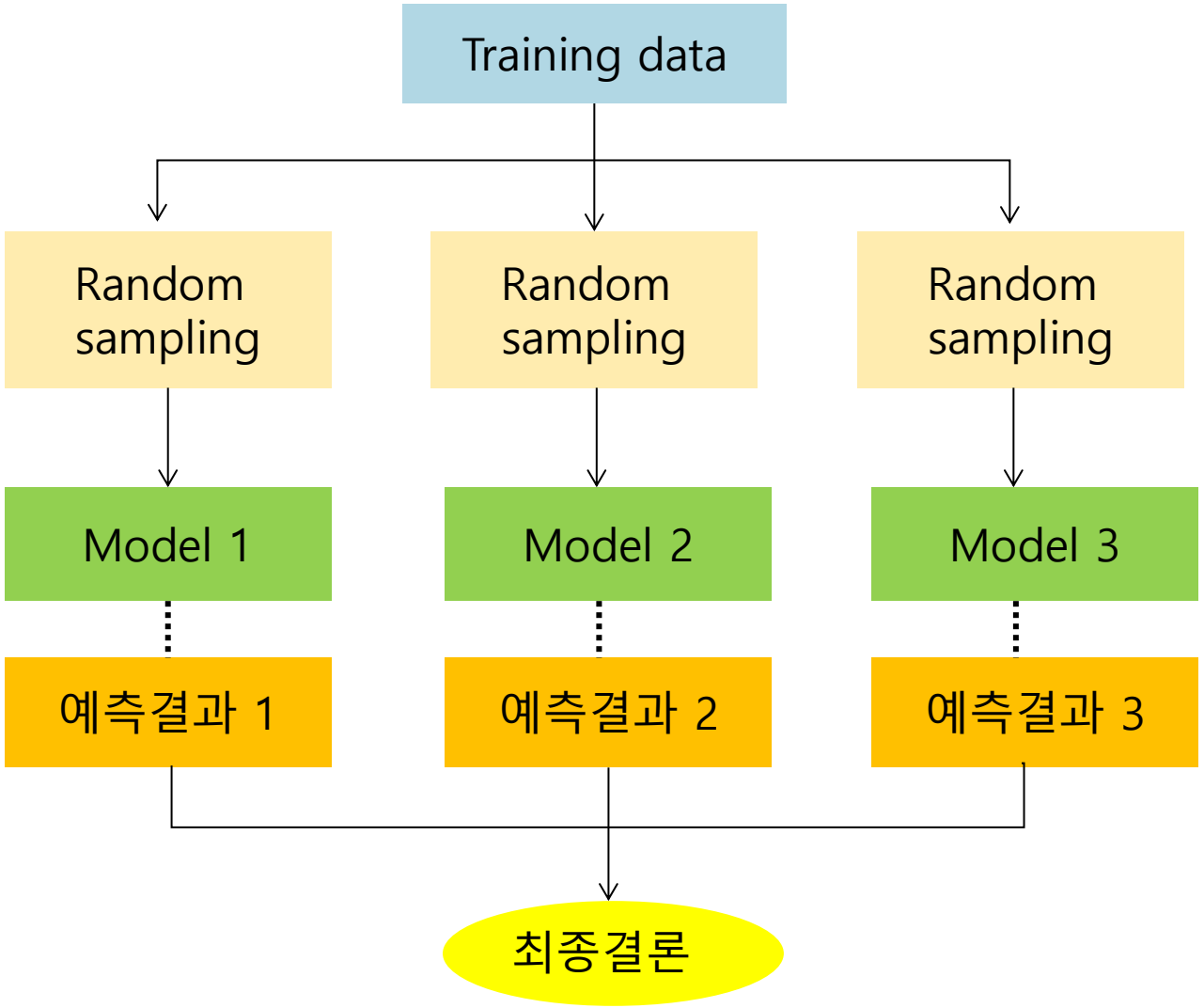
- Disadvantage
  - 최적의 모델을 찾기 위해 커널과 기타 조율모수(hyper parameter)의 여러 조합을 테스트해보아야 한다
  - 입력 데이터셋이 feature 수와 데이터 sample 수가 많으면 훈련시간이 많이 소요될 수 있다.
  - 모델의 해석이 불가능하지는 않지만 어렵다

4. Ensemble

- '다수 협의에 의한 결정' 이라는 원리를 예측 문제 해결에 적용한 것이 앙상블(Ensemble)
  - 일반적으로 예측문제에는 하나의 모델을 사용
  - 여러 개의 모델을 학습시킨 뒤 그 모델들의 예측 결과들을 취합하여 최종 결정을 내린다면 예측 정확도 향상에 도움이 됨
  - 결정 트리 기반 모델들에서 많이 사용이 됨
  - 앙상블에는 크게 두가지 종류가 있음
    - 배깅 (bagging) : Random Forest
    - 부스팅 (boosting) : AdaBoost, GBM, LightGBM, XGBoost

4. Ensemble

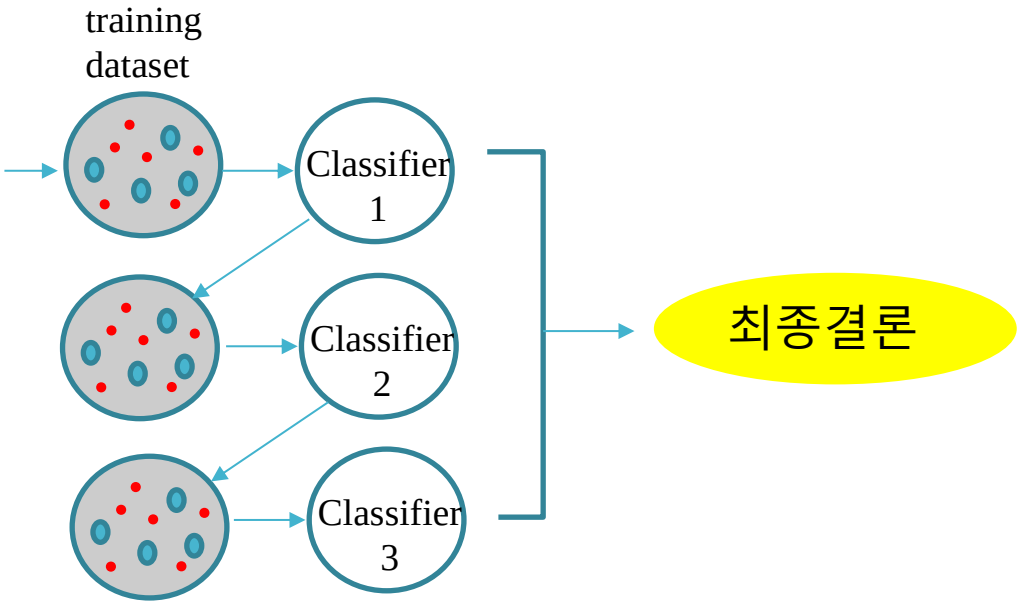
- 배깅(bagging)



4. Ensemble

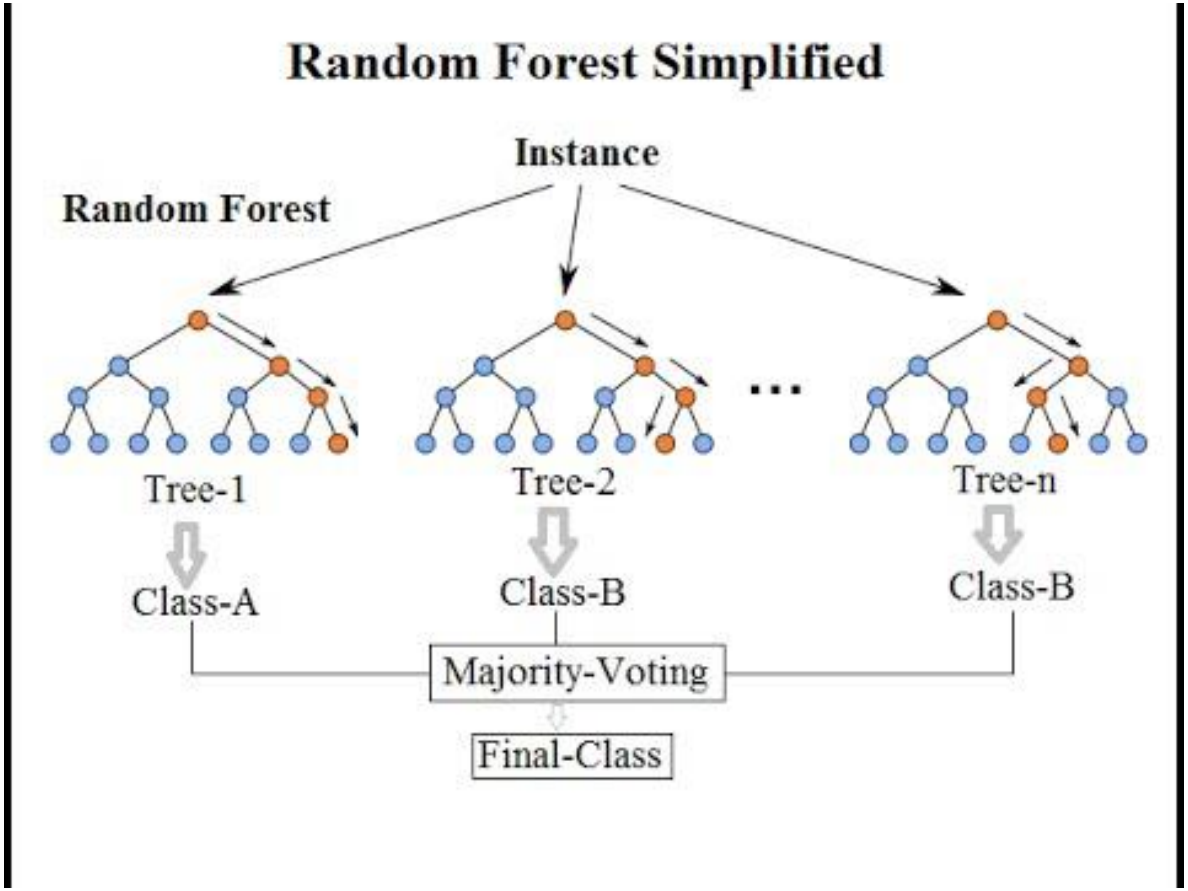
- 부스팅 (boosting)
  - 예측모델 1 의 결과를 보고 오답에는 가중치를 부여
  - 예측모델 2 는 예측 모델 1에의 오답 부분을 집중적으로 학습, 오답을 낮춤
  - 예측모델 3은 예측 모델 2의 오답 부분을 집중적으로 학습, 오답을 낮춤
  - 이 과정을 반복

  - 부스팅은 배깅에 비해 성능이 더 좋으나 과적합(overfitting)이 더 쉽게 발생하는 것으로 알려짐



5. Random Forest

- N개의 Decision Tree가 투표를 통해 결정하는 방식
- 앙상블의 Bagging approach



https://www.youtube.com/watch?v=ajTc5y3OqSQ

5. Random Forest

- "Bagging" 은 Bootstrap Aggregation의 약자
- 주어진 데이터(training set)에서 랜덤하게 subset을 N번 sampling해서 (좀 더 정확하게는 instances과 features들을 random하게 sampling) N개의 예측모형을 생성
- 개별 예측모형이 voting하는 방식으로 예측결과를 결정하여 Low Bias는 유지하고 High Variance는 줄이는 방법

- Random Forest는 이런 Bagging 계열의 가장 대표적이고 예측력 좋은 알고리즘
- 예측결과의 정확성(Low Bias)은 개별 예측모형에 쓰이는 알고리즘(decision tree)의 평균값으로 유지되는 반면 높은 분산(High Variance)은 Central Limit Theorem에 의해 낮아진다.

- Regression도 지원

```
sklearn.ensemble.RandomForestRegressor
```


http://blog.heartcount.io/random-forest-ver-10

5. Random Forest

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier¶

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

5. Random Forest

05_random_forest.py

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd

# prepare the iris dataset
df = pd.read_csv('D:/data/liver.csv')
print(df.head())
print(df.columns)   # column names

df_X = df.loc[:, df.columns != 'category']
df_y = df['category']

# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train_test_split(df_X, df_y, test_size=0.3,\
                     random_state=1234)
```

5. Random Forest

05_random_forest.py

```
# Define learning model (# of tree: 10) ##################
model = RandomForestClassifier(n_estimators=10, random_state=1234)
# Train the model using the training sets
model.fit(train_X, train_y)

# performance evaluation
print('Train accuracy :', model.score(train_X, train_y))
print('Test accuracy :', model.score(test_X, test_y))

pred_y = model.predict(test_X)
confusion_matrix(test_y, pred_y)
```

```
Train accuracy : 0.9875518672199171
Test accuracy : 0.7307692307692307
Out[465]:
array([[52, 12],
       [16, 24]], dtype=int64)
```

5. Random Forest

```
# Define learning model (# of tree: 50) ##################
model = RandomForestClassifier(n_estimators=50, random_state=1234)
# Train the model using the training sets
model.fit(train_X, train_y)

# performance evaluation
print('Train accuracy :', model.score(train_X, train_y))
print('Test accuracy :', model.score(test_X, test_y))

pred_y = model.predict(test_X)
confusion_matrix(test_y, pred_y)
```

```
Train accuracy : 1.0
Test accuracy : 0.7692307692307693
Out[466]:
array([[53, 11],
       [13, 27]], dtype=int64)
```

5. Random Forest

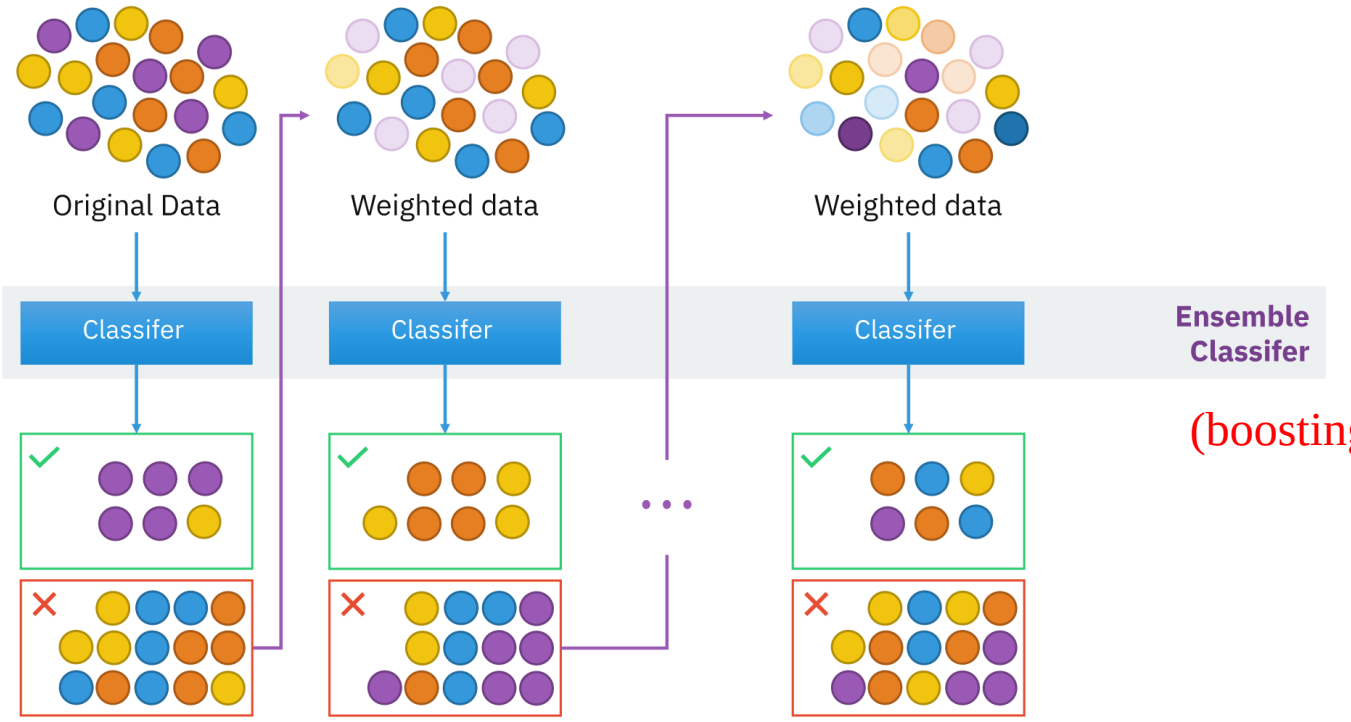
Hyper parameters

- n_estimators : 생성하는 트리의 개수. 많을수록 성능이 좋아짐. 일반적으로는 500, 1000 이면 충분
- max_features : The number of features to consider when looking for the best split. 기본값은 'auto'
- criterion: measure the quality of a split. 기본값('gini') 사용 추천

- 기타 parameters
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

6. xgboost

- 앙상블의 Boosting 방법



(boosting)

https://en.wikipedia.org/wiki/Boosting_(machine_learning)#/media/File:Ensemble_Boosting.svg

6. xgboost

- XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.
- It implements machine learning algorithms under the Gradient Boosting framework.
- XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

- Not supported by scikit-learn

6. xgboost

- Install xgboost module
  - https://xgboost.readthedocs.io/en/stable/install.html#python

```
pip install xgboost
```

- Xgboost example
- https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7

6. xgboost

05_xgboost.py

```
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

# prepare the credit dataset
df = pd.read_csv('D:/data/liver.csv')

df_X = df.loc[:, df.columns != 'category']
df_y = df['category']

# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train_test_split(df_X, df_y, test_size=0.3,\
                     random_state=1234)

# Convert data into xgboost fromat
D_train = xgb.DMatrix(train_X, label=train_y)
D_test = xgb.DMatrix(test_X, label=test_y)
```

6. xgboost

```
# Define & train learning model  ###################################
param = {
    'eta': 0.2,
    'max_depth': 3,
    'objective': 'binary:logistic',
    'eval_metric': 'error'}

steps = 20  # The number of training iterations

model = xgb.train(param, D_train, steps)

pred = model.predict(D_test)
round_preds = np.round(pred) # real -> [0,1]

accuracy_score(test_y, round_preds)
```

```
In [55]: pred
Out[55]:
array([0.6890685 , 0.83535254, 0.75385916, 0.18295208,
0.2898296 ,
       0.34704235, 0.3224759 , 0.22696435, 0.55630124,
0.2496487 ,
       0.2785892 , 0.20648101, 0.35500503, 0.75385916,
0.34956467
```

```
In [54]: accuracy_score(test_y, round_preds)
Out[54]: 0.7692307692307693
```

6. xgboost

- Hyper parameters
  - https://xgboost.readthedocs.io/en/stable/parameter.html
  - https://www.kaggle.com/code/lifesailor/xgboost

  - eta: Learning rate (일반적으로 0.01 - 0.2)
  - max_depth: Tree 깊이 수
  - objective: 목적함수 종류
    - reg:squarederror : regression with squared loss.
    - reg:squaredlogerror

    - binary:logistic (이진 분류, 결과가 확률로 나옴)
    - multi:softmax (다중 분류)
    - multi-softprob (다중 확률)

6. xgboost

- Hyper parameters
  - eval_metric: 평가 지표
    - rmse – root mean square error
    - mae – mean absolute error
    - logloss – negative log-likelihood
    - error – Binary classification error rate (0.5 threshold)
    - merror – Multiclass classification error rate
    - mlogloss – Multiclass logloss
    - auc: Area under the curve
  - seed

[실습 과제]

- PimaIndiansDiabetes.csv 에 대해 다음의 과제를 수행하시오

1	pregnant	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
2	6	148	72	35	0	33.6	0.627	50	pos
3	1	85	66	29	0	26.6	0.351	31	neg
4	8	183	64	0	0	23.3	0.672	32	pos
5	1	89	66	23	94	28.1	0.167	21	neg
6	0	137	40	35	168	43.1	2.288	33	pos
7	5	116	74	0	0	25.6	0.201	30	neg
8	3	78	50	32	88	31	0.248	26	pos

- Decision tree, SVM, randomforest, xgboost 알고리즘으로 예측 모델을 개발하고, 그 결과를 비교하시오
  - ppt의 코드 활용
  - accuracy 가 최대한 높게 나오도록 각 알고리즘의 파라메터를 조정해보시오
  - 결과는 다음과 같은 형식이 되도록 한다.

```
[Algorithm: xxxx]
Train accuracy : oooo
Test accuracy : oooo
[Algorithm: yyyy]
Train accuracy : oooo
Test accuracy : oooo
…
```

과제제출시
실행 화면 캡쳐는
이 부분만 하면 됨