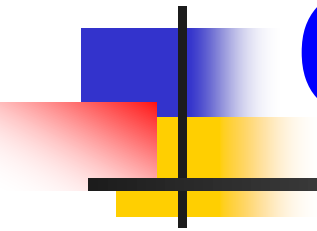


Chapter 12-2

기계학습(Machine Learning) : 활용(scikit-learn)

강의 내용

- 기계학습을 위한 파이썬 라이브러리
- scikit-learn 소개
- 대표적인 scikit-learn의 모델 클래스
- 기계학습 구현을 위한 간단한 실습 예제
- 실습

1. 기계학습을 위한 파이썬 라이브러리

Libraries

numpy	: 다차원 배열
pandas	: 데이터 분석 (행과 열이 존재하는 데이터 프레임)
matplotlib	: 그래프 그리기
scikit-learn	: 머신러닝의 모든 일반적인 알고리즘을 제공 : 기계학습 분야에 가장 인기있는 라이브러리

2. scikit-learn 소개

- scikit-learn 이란?
  - 다양한 기계학습 알고리즘을 구현한 python 라이브러리
  - python에서 기계학습용으로 가장 인기있는 라이브러리
  - 일관되고 간결한 API
  - 유용하고 완전한 온라인 문서 제공
    - https://scikit-learn.org/

- scikit-learn의 설치
  - Anaconda Prompt에서 다음 명령어 실행
    - conda install scikit-learn
  - seaborn 라이브러리도 같이 설치 (matplotlib 라이브러리를 기반으로 만든 라이브러리)
    - conda install seaborn

  - seaborn: Matplotlib를 개선하여 Pandas의 DataFrame 기능과 통합 용이

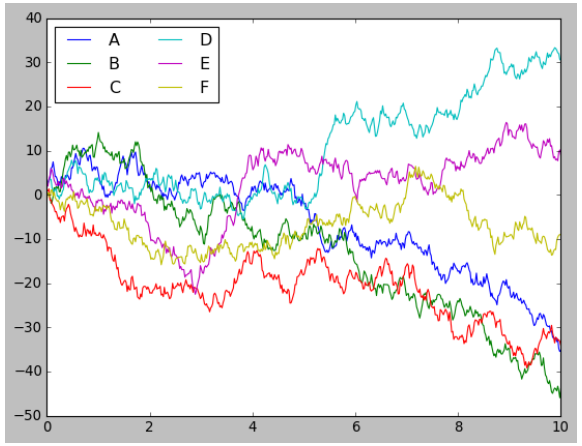
matplotlib와 seaborn의 차이

- matplotlib 기본값으로 plot 그리기

In [1]:
```
import matplotlib.pyplot as plt
plt.style.use('classic')
import numpy as np
import pandas as pd
```

In [2]:
```
rng = np.random.RandomState(0)
x = np.linspace(0, 10, 500)
y = np.cumsum(rng.randn(500, 6), 0)
```

In [3]:
```
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left')
```



- seaborn으로 스타일 설정 후 plot

In [4]:
```
import seaborn as sns
sns.set()
```

In [5]:
```
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left')
```

3. 대표적인 scikit-learn의 모델 클래스

알고리즘	Estimator 클래스와 인스턴스화
Linear regression	from sklearn.linear_model import LinearRegression model = LinearRegression()
k-Nearest Neighbor	from sklearn.neighbors import KNeighborsClassifier model = KNeighborsClassifier(n_neighbors=3)
Support Vector Machine	from sklearn import svm model = svm.SVC()
Decision Tree	from sklearn.tree import DecisionTreeClassifier model = DecisionTreeClassifier()
K-means Clustering	from sklearn.cluster import KMeans model = KMeans(n_clusters=2)

4. 기계학습 구현을 위한 간단한 실습 예제

- "온라인 뮤직 스토어"가 있다고 가정
  - 사용자가 온라인 뮤직 스토어에 가입할 때 연령과 성별을 입력
- 연령과 성별을 기초로 한 프로필을 기반으로, 사용자가 플레이할 것 같은 다양한 음악 앨범을 추천해주고 음악 판매량을 지속적으로 늘리고자 함

- 본 프로젝트에서 우리는 기존 사용자를 기반으로 한 샘플 데이터와 머신러닝을 활용하여 모델을 생성
- 새로운 사용자가 온라인 뮤직 스토어에 가입했을 때, 기존 비슷한 프로필을 가진 사용자가 어떤 종류의 음악에 흥미가 있는지 사용자에게 제안하고자 함

MyMusic
STORE

실습 예제를 위한 데이터 생성

- 예제 데이터
  - age 컬럼: 나이
  - gender 컬럼: 성별
    - 0: 여성
    - 1: 남성
  - genre 컬럼: 음악 장르

- 예제 데이터를 csv 파일로 만들어 활용

age	gender	genre
20	1	HipHop
23	1	HipHop
25	1	HipHop
26	1	Jazz
29	1	Jazz
30	1	Jazz
31	1	Classical
33	1	Classical
37	1	Classical
20	0	Dance
21	0	Dance
25	0	Dance
26	0	Acoustic
27	0	Acoustic
30	0	Acoustic
31	0	Classical
34	0	Classical
35	0	Classical

Importing a Data set

- Jupyter에서 data set 읽어오기
  - 데이터 시각화를 위해 pandas 이용

In [1]:
```
import pandas as pd
music_data = pd.read_csv('music.csv')
music_data
```

- 파일명 입력 (파일이 위치한 경로 체크)

Out [7] :

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

Preparing the data (1/3)

- 데이터 정리
  - 중복된 데이터 제거, null 값 제거 등

- 데이터 구분
  - **Input set:** 모델에 입력할 데이터 컬럼
  - **Output set:** 모델을 통해 예측되는 컬럼

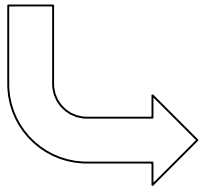
Input set Output set

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

Preparing the data (2/3)

- Input set 추출하기

```
import pandas as pd
music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
X
```

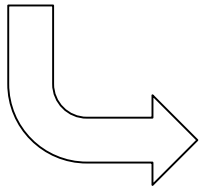


	age	gender
0	20	1
1	23	1
2	25	1
3	26	1
4	29	1
5	30	1
6	31	1
7	33	1
8	37	1
9	20	0
10	21	0
11	25	0
12	26	0
13	27	0
14	30	0
15	31	0
16	34	0
17	35	0

Preparing the data (3/3)

- Output set 추출하기

```
import pandas as pd
music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
Y
```



```
0         HipHop
1         HipHop
2         HipHop
3           Jazz
4           Jazz
5           Jazz
6      Classical
7      Classical
8      Classical
9          Dance
10         Dance
11         Dance
12      Acoustic
13      Acoustic
14      Acoustic
15     Classical
16     Classical
17     Classical
Name: genre, dtype: object
```

Learning (1/3)

- 기계학습 알고리즘을 사용하여 모델을 구축
  - 많은 알고리즘이 있고 각 알고리즘에는 장단점이 존재

- 수월한 모델 구축을 위해 scikit-learn 활용
  - 다양한 기계학습 알고리즘들이 scikit-learn 라이브러리에 이미 구현되어 있음
  - 명시적으로 프로그램할 필요 없이 바로 활용 가능

- 본 예제에서는 의사결정트리 (Decision tree) 알고리즘을 활용하여 모델 구축하고자 함

Learning (2/3)

- scikit-learn에서 import 하기

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
```

Decision tree 알고리즘이 구현되어 있음

- Decision tree 모델 생성하기

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']

model = DecisionTreeClassifier()
```

Decision tree 모델 생성

Learning (3/3)

- 학습하기
  - Decision tree 모델에 data set를 입력하여 패턴을 학습

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']

model = DecisionTreeClassifier()
model.fit(X, Y)
```

X: Input set
Y: Output set

Predicting (1/3)

- 예측하기: 생성된 모델을 통해 예측하기
  - 새로운 회원가입자들이 온라인 뮤직 스토어에 등록했을 경우 각자 좋아하는 장르의 음악이 무엇인지 예측
  - (예) 21세 남성과 22세 여성이 온라인 뮤직 스토어에 가입했을 때 좋아하는 장르의 음악이 무엇인지 예측해서 보여주세요.

age	gender	genre
20	1	HipHop
23	1	HipHop
25	1	HipHop
26	1	Jazz
29	1	Jazz
30	1	Jazz
31	1	Classical
33	1	Classical
37	1	Classical
20	0	Dance
21	0	Dance
25	0	Dance
26	0	Acoustic
27	0	Acoustic
30	0	Acoustic
31	0	Classical
34	0	Classical
35	0	Classical

20세 남성: HipHop
23세 남성: HipHop
25세 남성: HipHop
- 21세 남성의 경우, 좋아하는 장르의 정보(샘플)가 없음
- 21세의 남성이 좋아하는 장르의 음악을 요청하면 HipHop이라고 예측할 것이라 예상됨

20세 여성: Dance
23세 여성: Dance
25세 여성: Dance
- 22세 여성의 경우, 좋아하는 장르의 정보(샘플)가 없음
- 22세의 여성이 좋아하는 장르의 음악을 요청하면 Dance라고 예측할 것이라 예상됨

Predicting (2/3)

- 예측하기
  - (예) 21세 남성과 22세 여성이 온라인 뮤직 스토어에 가입했을 때 좋아하는 장르의 음악이 무엇인지 예측해서 보여주세요.

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']

model = DecisionTreeClassifier()
model.fit(X, Y)
predictions = model.predict([ [21, 1], [22, 0] ])
predictions
```

21세 남성: [21, 1]
22세 여성: [22, 0]

```
array(['HipHop', 'Dance'], dtype=object)
```

Predicting (3/3)

- 예측의 어려움
  - 모델을 만들고 정확하게 예측하는 것은 항상 쉬운 것이 아님
  - 모델을 생성하고 난 이후, 모델의 정확성(accuracy)를 측정
  - 모델이 정확하지 않은 경우?
    - 파라미터를 미세 조정
    - 다른 기계학습 알고리즘을 사용하여 모델 구축

Calculating the accuracy (1/7)

- 모델의 정확성 정도를 측정해보기
  - Data set을 분할하기
    - Training set: 모델을 학습시키기 위한 데이터 셋
    - Testing set: 생성된 모델의 정확성을 시험해보기 위한 데이터 셋
    - 일반적인 분할 비율은 Training set은 70~80%, Testing set은 20~30%
  - Training set을 이용해서 모델을 구축하고,
  - Testing set의 값을 모델에 입력하여 도출되는 예측값을 testing set에서 제공되고 있는 정답값과 비교!
  - => 생성된 모델이 얼마나 정확한지 측정 가능

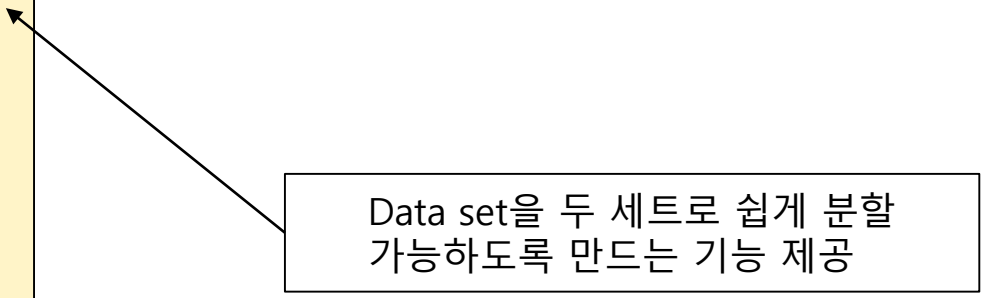
Calculating the accuracy (2/7)

- Training set 및 Testing set으로 데이터 분할하기

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

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']

model = DecisionTreeClassifier()
model.fit(X, Y)
predictions = model.predict([ [21, 1], [22, 0] ])
predictions
```



Calculating the accuracy (3/7)

- Training set 및 Testing set으로 데이터 분할하기
  - Training set의 경우 X_train 및 Y_train에 할당
  - Testing set의 경우 X_test 및 Y_test에 할당

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

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X, Y)
predictions = model.predict([ [21, 1], [22, 0] ])
predictions
```

Testing set 크기를 20%로 설정

train_test_split()은 tuple을 리턴

Calculating the accuracy (4/7)

- Training set을 이용하여 모델 생성하기
  - 앞의 예제에서는 전체 data set을 모두 활용하여 모델을 생성하였으나,
    - 이번에는 Training set을 이용하여 모델을 생성하고
    - 모델의 정확성 측정을 위해 Testing set을 이용하여 모델을 평가하고자 함

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

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict([ [21, 1], [22, 0] ])
predictions
```

Training set을 이용하여 모델 생성
기존 X대신 X_train
기존 Y대신 Y_train

Calculating the accuracy (5/7)

- Testing set을 이용하여 모델 평가하기 (정확도 측정하기)
  - Training set을 기반으로 생성된 모델에 Testing set을 입력해주기
  - 예제에서 Testing set의 입력부분은 X_test에 할당되어 있음

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

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_test)
predictions
```

X_test에는 Training set을 통해 생성된 모델을 테스트하기위한 입력값이 포함되어 있음

Calculating the accuracy (6/7)

- Testing set을 이용하여 모델 평가하기 (정확도 측정하기)
  - X_test를 모델에 입력하여 예측된 결과값을 실제값인 Y_test과 비교

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

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_test)

score = accuracy_score(Y_test, predictions)
score
```

정확도를 쉽게 계산해내기 위한 기능 제공

결과는 0에서 1사이의 값
(% 변환 가능)

Y_test에는 실제값
predictions에는 모델을 통해 얻은 예측값
accuracy_score는 0에서 1사이의 값 도출

Calculating the accuracy (7/7)

- Testing set을 이용하여 모델 평가하기 (정확도 측정하기)
  - 프로그램을 실행할때마다 다른 결과를 도출
    - ∵ train_test_split를 통해 training set와 testing set으로 분할하면 데이터 분할을 수행할때마다 random으로 데이터를 선택하기 때문
    - 즉, 프로그램 실행할때마다 training set과 testing set이 변경됨
  - 결과는 0.5, 0.75, 1.0 등의 값이 도출

Testing set 사이즈 변경에 따른 정확도 변화

- Testing set의 size 변경에 따른 결과값 변화
  - Testing set의 크기를 0.2에서 0.8로 수정해보면?
  - 결과는 0.2, 0.266, 0.333…, 0.4, 0.4666… 등으로 모델의 정확성 하락
  - Training set이 줄어 모델이 학습하는데 아주 적은 데이터를 활용하기 때문

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

music_data = pd.read_csv('music.csv')
X = music_data.drop(columns=['genre'])
Y = music_data['genre']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.8)

model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_test)

score = accuracy_score(Y_test, predictions)
score
```

- 0.2에서 0.8로 수정
- Testing set 사이즈를 전체 데이터셋의 80%로 설정

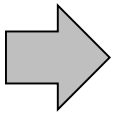
모델의 정확성 향상을 위한 고려사항

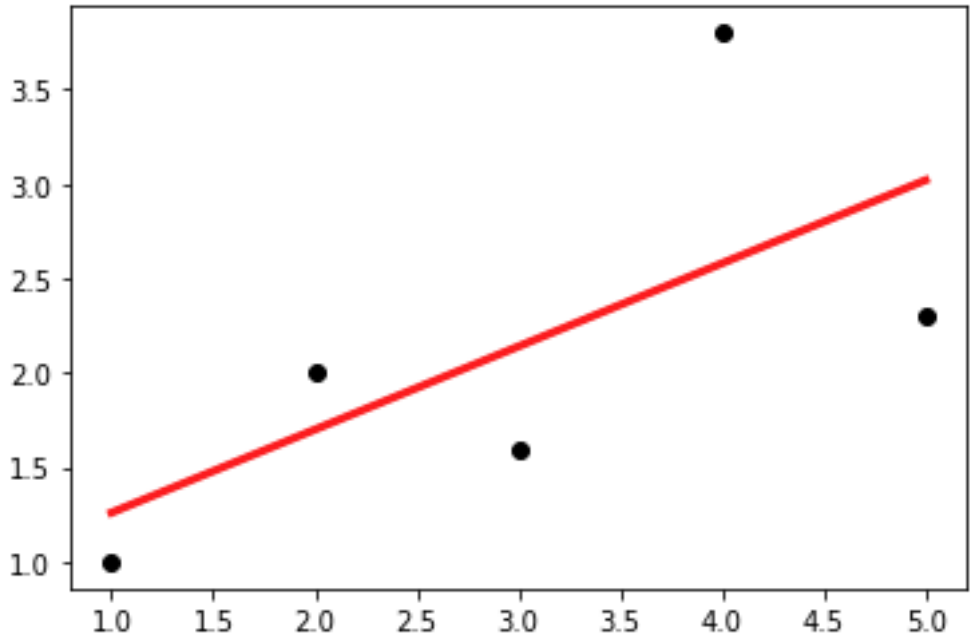
- 기계학습의 핵심 개념
  - 기계학습을 통해 모델의 정확성를 높이기 위해서는,
    - (1) 가능한 많은 수의 데이터를 제공해야 함 (수천, 수백만 개 또는 그 이상)
    - (2) 데이터가 잘 정제되어 있어야 함 (중복된 데이터, 관련성없는 데이터, 불완전한 데이터들은 모델이 잘못된 패턴을 학습하도록 만들기 때문)
  - 복잡한 문제일수록 모델의 정확성을 높이기 위해서는 더 많은 수의 데이터가 필요함
    - 본 예제에서는 3개의 컬럼으로만 구성된 데이터를 이용
    - 하지만, 어떠한 사진이 고양이인지 개인지 사자인지 등등을 판별해낼 수 모델을 만들기 위해서는 수백만장 이상의 사진 데이터가 필요

실습

- 선형 회귀를 이용하여 다음과 같은 예제 데이터를 가장 잘 설명하는 직선을 찾아보자.

x	y
1.0	1.0
2.0	2.0
3.0	1.6
4.0	3.8
5.0	2.3





실습

In [1]:

```
import matplotlib.pylab as plt
from sklearn import linear_model

reg = linear_model.LinearRegression()

x = [[1.0], [2.0], [3.0], [4.0], [5.0]] # LinearRegression()은 2차원 배열 형
태의 훈련 데이터셋만 사용하기 때문에 2차원 배열 형태로 표시
y = [1.0, 2.0, 1.6, 3.8, 2.3]
reg.fit(x, y) # 학습

# 학습 데이터와 y값을 산포도로 그리기
plt.scatter(x, y, color='black')

# 학습 데이터를 입력으로 하여 예측값을 계산
y_prediction = reg.predict(x)

# 학습 데이터와 예측값으로 선그래프 그리기
# 계산된 기울기와 y 절편을 가지는 직선이 그려짐
plt.plot(x, y_prediction, color='red', linewidth=3)
plt.show()
```