Lecture 4: Data Processing for ML

기계학습개론

박상효

학습목표

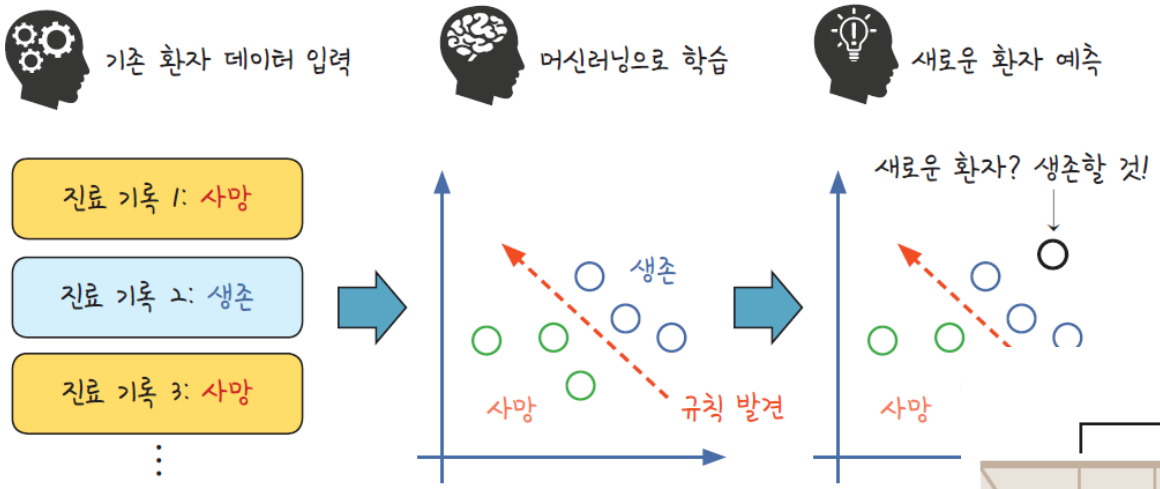
- 데이터의 중요성 이해하고 설명할 수 있음
- 파이썬 데이터 처리방식 이해하고 코드를 이해할 수 있음
- 과제1을 통한 데이터 처리 방식 및 성능 변화를 체험하고 이해할 수 있음

핵심용어

- Dataset

데이터의 중요성

(강의 2강)
리뷰: 폐암 수술 환자의 생존율 예측하기



Class 0: 사망
Class 1: 생존

속성 | 클래스

항목 \ 줄	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	293	1	3.8	2.8	0	0	0	0	0	0	12	0	0	0	1	0	62	0
2	1	2	2.88	2.16	1	0	0	0	1	1	14	0	0	0	1	0	60	0
3	8	2	3.19	2.5	1	0	0	0	1	0	11	0	0	1	1	0	66	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
470	447	8	5.2	4.1	0	0	0	0	0	0	12	0	0	0	0	0	49	0

(강의 2강)
리뷰: 폐암 수술 환자의 생존율 예측하기

Class 0: 사망
Class 1: 생존

Cropped dataset:
Totally **409** rows
(61 rows are removed)

속성 | 클래스

항목 \ 줄	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	293	1	3.8	2.8	0	0	0	0	0	0	12	0	0	0	1	0	62	0
2	1	2	2.88	2.16	1	0	0	0	1	1	14	0	0	0	1	0	60	0
3	8	2	3.19	2.5	1	0	0	0	1	0	11	0	0	1	1	0	66	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
470	447	8	5.2	4.1	0	0	0	0	0	0	12	0	0	0	0	0	49	0

In [1]:

```python
import pandas as pd
import numpy as np

# 실행할 때마다 같은 결과를 출력하기 위해 설정하는 부분입니다.
np.random.seed(3)

# 준비된 수술 환자 데이터를 불러들입니다.
Data_set = np.loadtxt("../dataset/ThoracicSurgery_evildata.csv", delimiter=",")

# 환자의 기록과 수술 결과를 X와 Y로 구분하여 저장합니다.
x = Data_set[:,0:17]
y = Data_set[:,17]

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver='lbfgs')

model.fit(x, y)

```

← Very simple ML algorithm (Detail: **Lecture 6**)

```
C:₩Users₩seasik_corner₩Anaconda3₩envs₩iml₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[1]:

```
LogisticRegression()
```

(강의 2강)
리뷰: 폐암 수술 환자의 생존율 예측하기

- 결과 :

In [2]:

```
1  y_pred = model.predict(x)
2
3  from sklearn.metrics import accuracy_score
4  y_pred = y_pred // 1000
5  print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 97.80%
```

(강의 2강)
리뷰: 폐암 수술 환자의 생존율 예측하기

- 결과 :

In [2]:

```
1  y_pred = model.predict(x)
2
3  from sklearn.metrics import accuracy_score
4  y_pred = y_pred // 1000
5  print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 97.80%
```

In [12]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import tensorflow as tf

tf.random.set_seed(3)

model = Sequential()
model.add(Dense(60, input_dim=17, activation='relu'))
model.add(Dense(60, activation='relu'))
model.add(Dense(60, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 딥러닝을 실행합니다.
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x, y, epochs=30, batch_size=10)
```

Using deep learning?

```
Epoch 2/30
41/41 [==============================] - 0s 688us/step - loss: 0.1553 - accuracy: 0.9780
Epoch 3/30
41/41 [==============================] - 0s 740us/step - loss: 0.1294 - accuracy: 0.9780
Epoch 4/30
41/41 [==============================] - 0s 730us/step - loss: 0.1883 - accuracy: 0.9780
Epoch 5/30
41/41 [==============================] - 0s 594us/step - loss: 0.1691 - accuracy: 0.9780
Epoch 6/30
41/41 [==============================] - 0s 608us/step - loss: 0.1689 - accuracy: 0.9756
Epoch 7/30
41/41 [==============================] - 0s 584us/step - loss: 0.1564 - accuracy: 0.9756
Epoch 8/30
41/41 [==============================] - 0s 632us/step - loss: 0.1411 - accuracy: 0.9756
Epoch 9/30
41/41 [==============================] - 0s 632us/step - loss: 0.1541 - accuracy: 0.9756
```

(강의 2강)
리뷰: 폐암 수술 환자의 생존율 예측하기

- 결과 :

In [13]:
```
y_pred = model.predict(x)

y_pred = y_pred // 1000
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 97.80%
```

- LogisticRegression과 소숫점 둘째자리까지 동일함!

데이터의 중요성

In [11]:

```
print(type(x), x.shape)

j = 0
for i in range(len(y)):
    if y[i] == 0:
        j += 1
print("Total: {}, Died: {}".format(len(y), j))
print("Percentage of died patients: {:.0%}".format(j / len(y)))
```

```
<class 'numpy.ndarray'> (409, 17)
Total: 409, Died: 400
Percentage of died patients: 98%
```

데이터 이해하기

피마 인디언 데이터 분석(모11)

피마 인디언 데이터 분석(모11)

- 파일 위치 : dataset/pima-indians-diabetes.csv

속성 / 클래스

	정보 1	정보 2	정보 3	…	정보 8	당뇨병 여부
1번째 인디언	6	148	72	…	50	1
2번째 인디언	1	85	66	…	31	0
3번째 인디언	8	183	64	…	32	1
…	…	…	…	…	…	…
768번째 인디언	1	93	70	…	23	0

샘플

- 샘플 수: 768
- 속성: 8
  - 정보 1 (pregnant): 과거 임신 횟수
  - 정보 2 (plasma): 포도당 부하 검사 2시간 후 공복 혈당 농도(mm Hg)
  - 정보 3 (pressure): 확장기 혈압(mm Hg)
  - 정보 4 (thickness): 삼두근 피부 주름 두께(mm)
  - 정보 5 (insulin): 혈청 인슐린(2-hour, mu U/ml)
  - 정보 6 (BMI): 체질량 지수(BMI, weight in kg/(height in m^2))
  - 정보 7 (pedigree): 당뇨병 가족력
  - 정보 8 (age): 나이
- 클래스: 당뇨(1), 당뇨 아님(0)

피마 인디언 데이터 분석(모11)

In [2]:
```
# -*- coding: utf-8 -*-
# 코드 내부에 한글을 사용가능 하게 해주는 부분입니다.

# pandas 라이브러리를 불러옵니다.
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 피마 인디언 당뇨병 데이터셋을 불러옵니다. 불러올 때 각 컬럼에 해당하는 이름을 지정합니다.
df = pd.read_csv('../dataset/pima-indians-diabetes.csv',
                 names = ["pregnant", "plasma", "pressure", "thickness", "insulin", "BMI", "pedigree", "ag
```

In [3]:
```
# 처음 5줄을 봅니다.
print(df.head(5))
```

```
   pregnant  plasma  pressure  thickness  insulin   BMI  pedigree  age  class
0         6     148        72         35        0  33.6     0.627   50      1
1         1      85        66         29        0  26.6     0.351   31      0
2         8     183        64          0        0  23.3     0.672   32      1
3         1      89        66         23       94  28.1     0.167   21      0
4         0     137        40         35      168  43.1     2.288   33      1
```

피마 인디언 데이터 분석(모11)

In [5]:
```
# 각 정보별 특징을 좀더 자세히 출력합니다.
print(df.describe())
```

```
          pregnant      plasma    pressure   thickness     insulin         BMI  ₩
count   768.000000  768.000000  768.000000  768.000000  768.000000  768.000000
mean      3.845052  120.894531   69.105469   20.536458   79.799479   31.992578
std       3.369578   31.972618   19.355807   15.952218  115.244002    7.884160
min       0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%       1.000000   99.000000   62.000000    0.000000    0.000000   27.300000
50%       3.000000  117.000000   72.000000   23.000000   30.500000   32.000000
75%       6.000000  140.250000   80.000000   32.000000  127.250000   36.600000
max      17.000000  199.000000  122.000000   99.000000  846.000000   67.100000

          pedigree         age       class
count   768.000000  768.000000  768.000000
mean      0.471876   33.240885    0.348958
std       0.331329   11.760232    0.476951
min       0.078000   21.000000    0.000000
25%       0.243750   24.000000    0.000000
50%       0.372500   29.000000    0.000000
75%       0.626250   41.000000    1.000000
max       2.420000   81.000000    1.000000
```

상관관계(correlation)

- Pearson correlation coefficient:
  - is a statistic that measures **linear correlation** between two variables X and Y.

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \qquad \text{(Eq.1)}$$

cov is the covariance

σ_X is the standard deviation of X

σ_Y is the standard deviation of Y

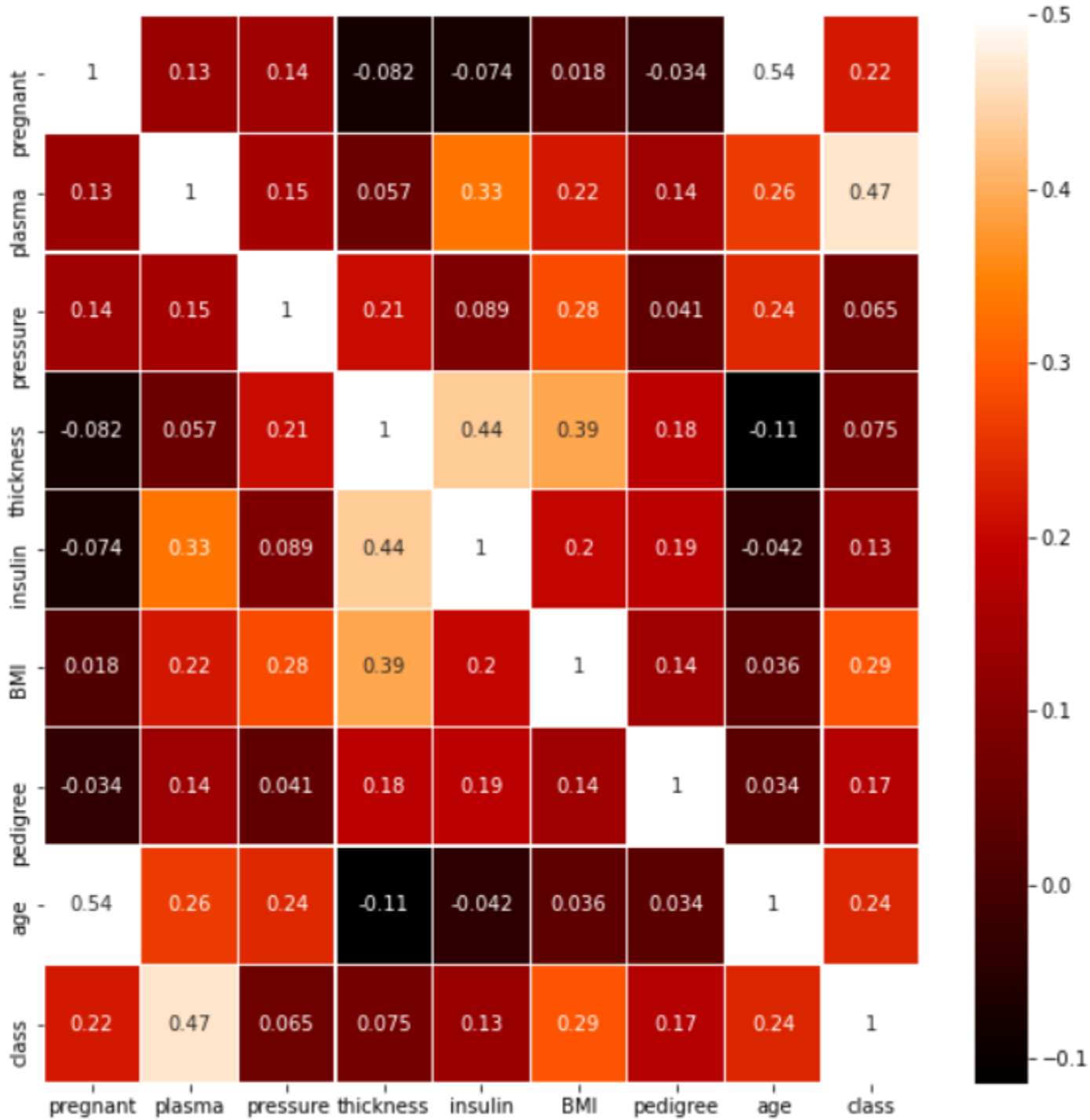
*출처 : https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

상관관계(correlation)

- Pearson correlation coefficient:
  - is a statistic that measures **linear correlation** between two variables X and Y.

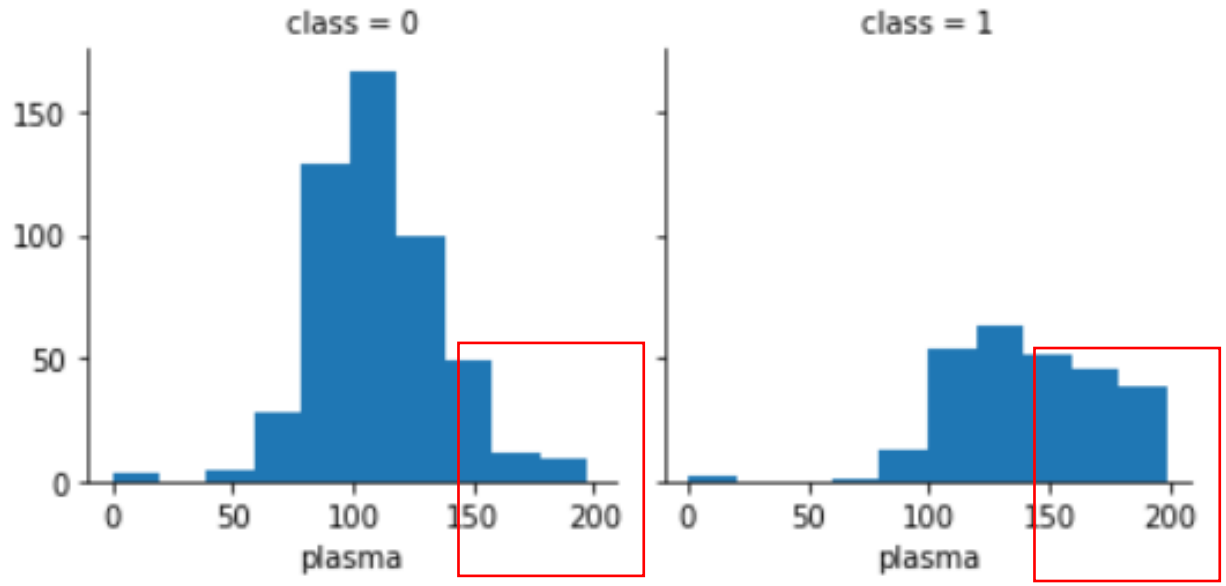
In [6]:

```
# 데이터 간의 상관관계를 그래프로 표현해 봅니다.

colormap = plt.cm.gist_heat   #그래프의 색상 구성을 정합니다.
plt.figure(figsize=(10,10))   #그래프의 크기를 정합니다.

# 그래프의 속성을 결정합니다. vmax의 값을 0.5로 지정해 0.5에 가까울 수록 밝은 색으로 표시되게 합니다.
sns.heatmap(df.corr(),linewidths=0.1,vmax=0.5, cmap=colormap, linecolor='white', annot=True)
plt.show()
```


pregnant 1 0.13 0.14 -0.082 -0.074 0.018 -0.034 0.54 0.22
plasma 0.13 1 0.15 0.057 0.33 0.22 0.14 0.26 0.47
pressure 0.14 0.15 1 0.21 0.089 0.28 0.041 0.24 0.065
thickness -0.082 0.057 0.21 1 0.44 0.39 0.18 -0.11 0.075
insulin -0.074 0.33 0.089 0.44 1 0.2 0.19 -0.042 0.13
BMI 0.018 0.22 0.28 0.39 0.2 1 0.14 0.036 0.29
pedigree -0.034 0.14 0.041 0.18 0.19 0.14 1 0.034 0.17
age 0.54 0.26 0.24 -0.11 -0.042 0.036 0.034 1 0.24
class 0.22 0.47 0.065 0.075 0.13 0.29 0.17 0.24 1
pregnant plasma pressure thickness insulin BMI pedigree age class
0.5 0.4 0.3 0.2 0.1 0.0 −0.1

Plasma에 따른 class 비교

In [7]:
```
1 grid = sns.FacetGrid(df, col='class')
2 grid.map(plt.hist, 'plasma',  bins=10)
3 plt.show()
```



피마 인디언의 당뇨병 예측 실행 (기존데이터 – logistic regression)

In [3]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt("../dataset/pima-indians-diabetes.csv", delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8]

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(solver='lbfgs')

model.fit(x, y)

y_pred = model.predict(x)

y_pred = y_pred // 1000
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 65.10%
```

피마 인디언의 당뇨병 예측 실행 (기존데이터 – random forest)

In [3]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt("../dataset/pima-indians-diabetes.csv", delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8]

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(solver='lbfgs')

model.fit(x, y)

y_pred = model.predict(x)

y_pred = y_pred // 1000
print("Accuracy: {:.2%}".format(ac
```

```
Accuracy: 65.10%
```

In [4]:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=10)

model.fit(x, y)    # 학습
y_pred = model.predict(x) # 예측
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 97.40%
```

피마 인디언의 당뇨병 예측 실행 (데이터수정 – logistic regression)

In [3]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt(
X = dataset[:,0:8]
Y = dataset[:,8]

from sklearn.linear_mode
from sklearn.metrics im

model = LogisticRegressi

model.fit(x, y)

y_pred = model.predict(x

y_pred = y_pred // 1000
print("Accuracy: {:.2%}"
```

```
Accuracy: 65.10%
```

In [1]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt("../dataset/pima-indians-diabetes_processed.csv", delimiter=",")
x = dataset[:,0:3]
y = dataset[:,3]

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(solver='lbfgs')

model.fit(x, y)

y_pred = model.predict(x)

y_pred = y_pred // 1000
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 65.10%
```

피마 인디언의 당뇨병 예측 실행 (데이터수정 – logistic regression)

In [3]:

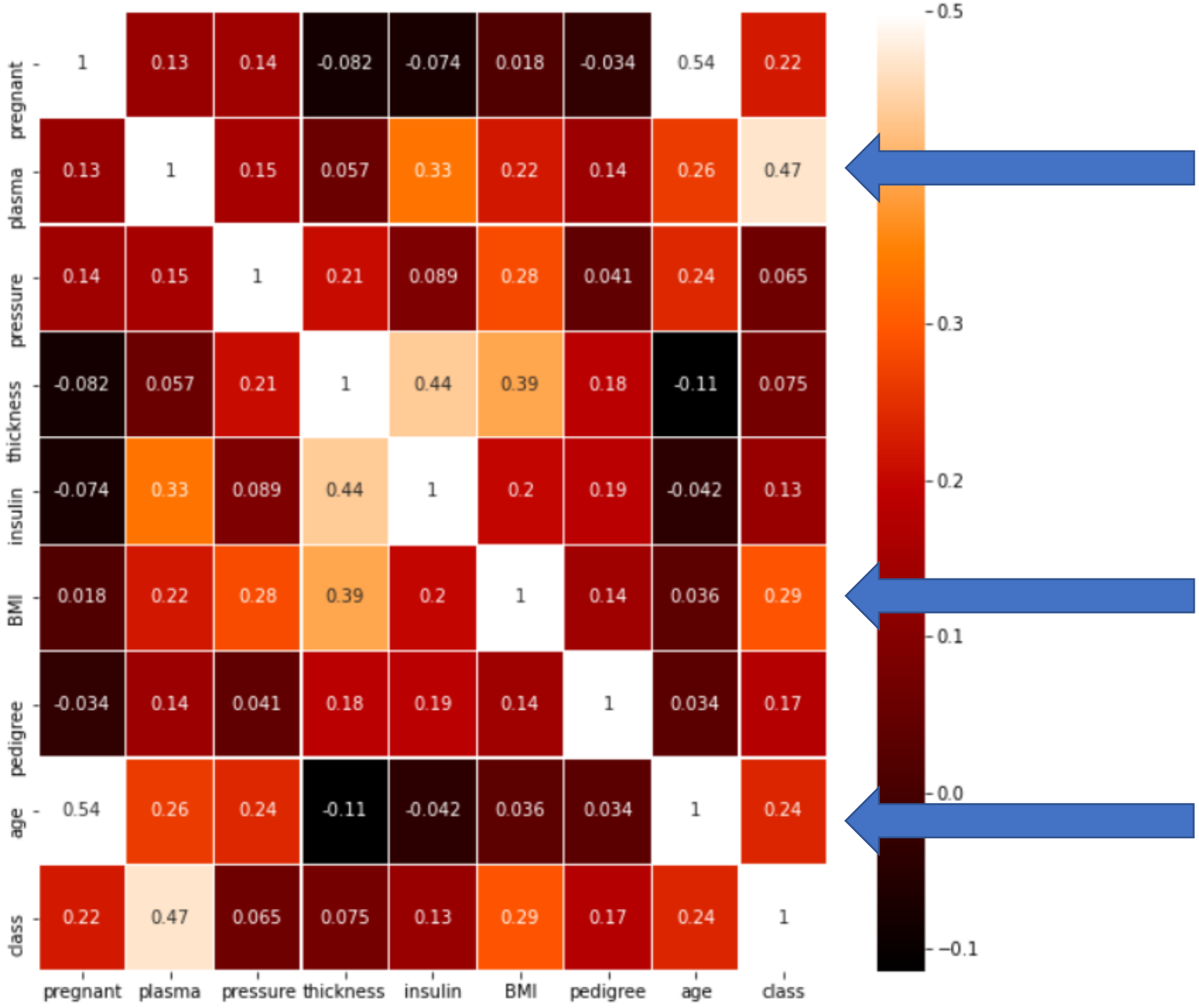
```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt(
X = dataset[:,0:8]
Y = dataset[:,8]

from sklearn.linear_mode
from sklearn.metrics im

model = LogisticRegressi

model.fit(x, y)

y_pred = model.predict(x

y_pred = y_pred // 1000
print("Accuracy: {:.2%}"
```

```
Accuracy: 65.10%
```

In [1]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt("../dataset/pima-indians-diabetes_processed.csv", delimiter=",")
x = dataset[:,0:3]
y = dataset[:,3]

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(solver='lbfgs')

model.fit(x, y)

y_pred = model.predict(x)

y_pred = y_pred // 1000
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 65.10%
```


pregnant 1 0.13 0.14 -0.082 -0.074 0.018 -0.034 0.54 0.22
plasma 0.13 1 0.15 0.057 0.33 0.22 0.14 0.26 0.47
pressure 0.14 0.15 1 0.21 0.089 0.28 0.041 0.24 0.065
thickness -0.082 0.057 0.21 1 0.44 0.39 0.18 -0.11 0.075
insulin -0.074 0.33 0.089 0.44 1 0.2 0.19 -0.042 0.13
BMI 0.018 0.22 0.28 0.39 0.2 1 0.14 0.036 0.29
pedigree -0.034 0.14 0.041 0.18 0.19 0.14 1 0.034 0.17
age 0.54 0.26 0.24 -0.11 -0.042 0.036 0.034 1 0.24
class 0.22 0.47 0.065 0.075 0.13 0.29 0.17 0.24 1
pregnant plasma pressure thickness insulin BMI pedigree age class
0.5
0.3
0.2
0.1
0.0
−0.1

In [3]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtx
X = dataset[:,0:8]
Y = dataset[:,8]

from sklearn.linear_mo
from sklearn.metrics i

model = LogisticRegres

model.fit(x, y)

y_pred = model.predict

y_pred = y_pred // 100
print("Accuracy: {:.2%}
```

```
Accuracy: 65.10%
```

In [1]:

```
import numpy

numpy.random.seed(3)

dataset = numpy.loadtxt("../dataset/pima-indians-diabetes_processed.csv", delimiter=",")
x = dataset[:,0:3]
y = dataset[:,3]

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression(solver='lbfgs')

model.fit(x, y)

y_pred = model.predict(x)

y_pred = y_pred // 1000
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 65.10%
```

In [4]:

```
from sklearn.ensemble

model = RandomForestCl

model.fit(x, y)    # 
y_pred = model.predict
print("Accuracy: {:.2%}
```

```
Accuracy: 97.40%
```

In [2]:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=10)

model.fit(x, y)    # 학습
y_pred = model.predict(x) # 예측
print("Accuracy: {:.2%}".format(accuracy_score(y, y_pred)))
```

```
Accuracy: 97.40%
```

Clustering(군집화)

Unsupervised learning 예시

와인 데이터

- multi-class classification dataset

Data Set Characteristics:

Number of Instances:	178 (50 in each of three classes)
Number of Attributes:	13 numeric, predictive attributes and the class
Attribute Information:	• Alcohol • Malic acid • Ash • Alcalinity of ash • Magnesium • Total phenols • Flavanoids • Nonflavanoid phenols • Proanthocyanins • Color intensity • Hue • OD280/OD315 of diluted wines • Proline

- **class:**
  - class_0
  - class_1
  - class_2

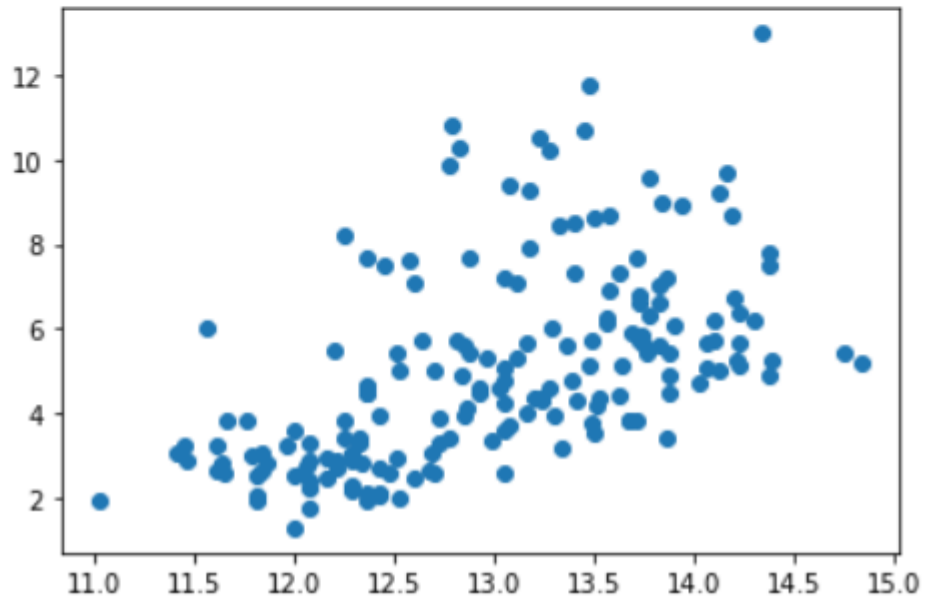
*출처 : https://scikit-learn.org/stable/datasets/index.html#wine-dataset

와인 데이터

- multi-class classification dataset

In [8]:
```
from sklearn.datasets import load_wine

data = load_wine()


x3 = data.data[:, [0]]
y3 = data.data[:, [9]]
plt.scatter(x3, y3)
```

Out[8]: <matplotlib.collections.PathCollection at 0x28f077653a0>



*출처 : https://scikit-learn.org/stable/datasets/index.html#wine-dataset

코드 : K-means & 시각화

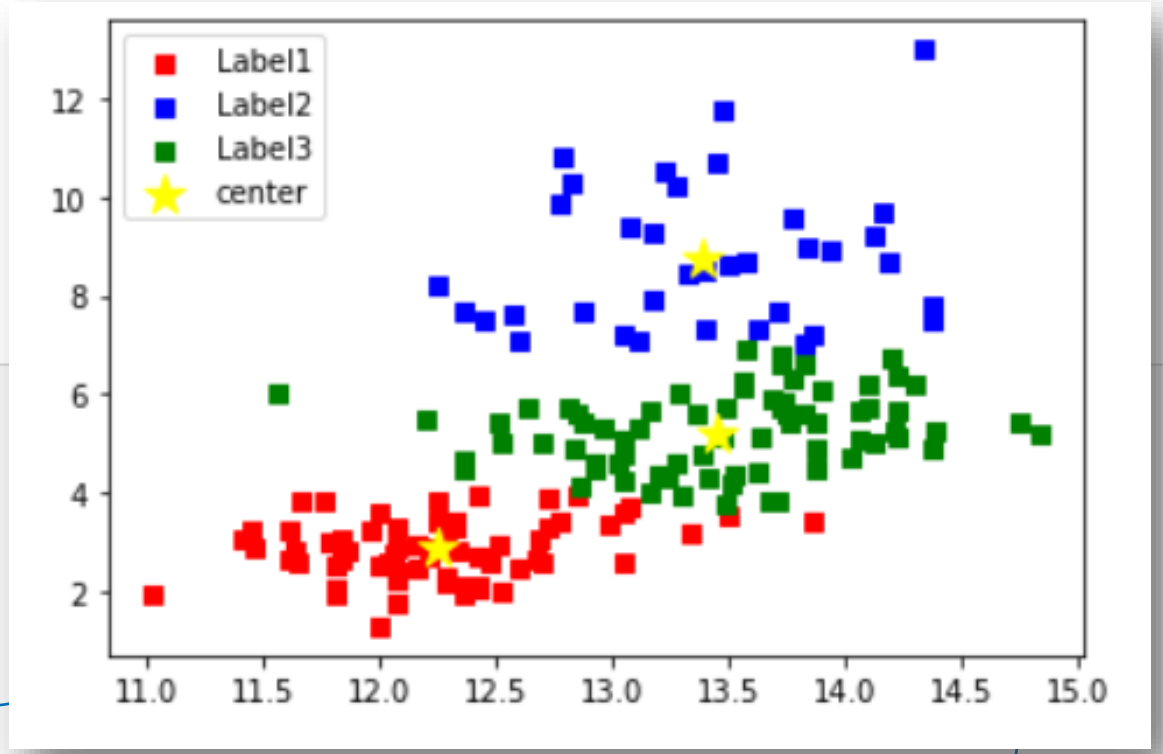
In [1]:

```
X = data.data[:, [0, 9]]

from sklearn.cluster import KMeans

n_cluster = 3
model = KMeans(n_clusters=n_cluster)
pred = model.fit_predict(X)

import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.scatter(X[pred==0, 0], X[pred==0, 1], color='red', marker='s', label='Label1')
ax.scatter(X[pred==1, 0], X[pred==1, 1], color='blue', marker='s', label='Label2')
ax.scatter(X[pred==2, 0], X[pred==2, 1], color='green', marker='s', label='Label3')
ax.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1], s=200, color='yellow', marker="*", label="center")

ax.legend()
plt.show()
```

결과

In [1]:

```
X = data.data[:, [0, 9]]

from sklearn.cluster import KMeans

n_cluster = 3
model = KMeans(n_clusters=n_cluster)
pred = model.fit_predict(X)

import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.scatter(X[pred==0, 0], X[pred==0, 1], color='red', marker='s', label='Label1')
ax.scatter(X[pred==1, 0], X[pred==1, 1], color='blue', marker='s', label='Label2')
ax.scatter(X[pred==2, 0], X[pred==2, 1], color='green', marker='s', label='Label3')
ax.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1], s=200, color='yellow', marker="*", label="center")

ax.legend()
plt.show()
```



Summary

- 데이터의 중요성
  - 편향데이터
  - 데이터 이해를 위한 분석 방법
    - (특성, 상관관계, 시각화)
  - 데이터에 따른 ML 성능 비교
- 데이터처리용 모듈/알고리즘
  - Numpy, pandas, matplotlib, seaborn
  - ML 알고리즘(logistic regression, random forest, K-means) 사례 비교

In the next lecture…

- 선형회귀
  - 개념, 코드
- 오차함수(목적함수)
  - 오차 정의
  - 다양한 gradient descent 알고리즘 및 주요 용어 이해
  - 개념, 코드

참고자료

- 머1.2
- 모11
- 오픈소스코드

기 : 기계학습, 오일석, 2017
핸 : 핸즈온머신러닝, 2/E, 2020 (번역)
모 : 모두의 딥러닝, 2/E, 2020
케 : 케라스 창시자에게 배우는…, 2018 (번역)
머 : 머신러닝 도감 그림으로…, 2019 (번역)
파 : Python machine learning, 2/E, 2019 (번역) → "머신러닝 교과서 with 파이썬, …" 2019

QnA

- 강의 업로드 시간 변경 요청?
  - 본래 수업시간에 업로드 → 수업요일 0:00에 업로드하여 퀴즈/과제 전에 영상 시청 가능하도록 변경
- 퀴즈(시험) 접속횟수 2회 이상?
  - 재시험대상자로 편입됨

QnA

- LMS질의응답 참고
  - Q1. y = wx+b 라는 식이 있을경우 w와 b의 값을 랜덤으로 바꾸어가면서 알맞은 방정식을 찾는다고 하셨는데 코드에서 위 부분에 해당하는 부분이 어느부분인지 알 수 있겠습니까??

In [7]:

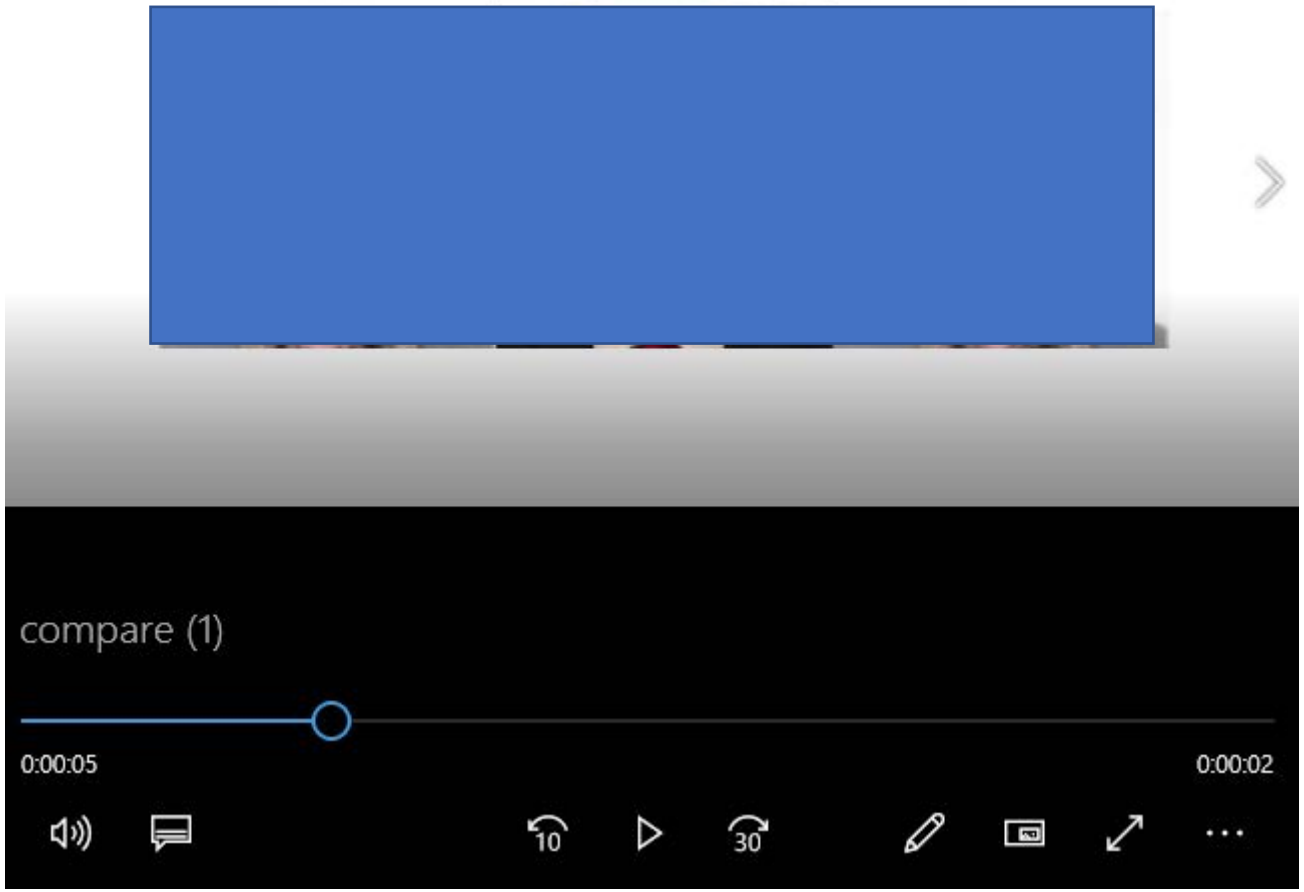
```
from __future__ import print_function

import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop

batch_size = 128
num_classes = 10
epochs = 20

# the data, split between train and test
(x_train, y_train), (x_test, y_test) = m

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(256, activation='relu', input_shape=(784,)))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])
model.summary()

history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
                    verbose=1, validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
60000/60000 [==============================] - 2s 33us/step - loss: 0.0118 - acc: 0.9968 - val_loss: 0.0727 - val_acc: 0.9813
Epoch 13/20
60000/60000 [==============================] - 2s 29us/step - loss: 0.0094 - acc: 0.9975 - val_loss: 0.0756 - val_acc: 0.9813
Epoch 14/20
60000/60000 [==============================] - 2s 30us/step - loss: 0.0078 - acc: 0.9980 - val_loss: 0.0788 - val_acc: 0.9804
Epoch 15/20
60000/60000 [==============================] - 2s 29us/step - loss: 0.0064 - acc: 0.9984 - val_loss: 0.0743 - val_acc: 0.9819
Epoch 16/20
60000/60000 [==============================] - 2s 28us/step - loss: 0.0051 - acc: 0.9989 - val_loss: 0.0792 - val_acc: 0.9820
Epoch 17/20
60000/60000 [==============================] - 2s 27us/step - loss: 0.0044 - acc: 0.9990 - val_loss: 0.0772 - val_acc: 0.9815
Epoch 18/20
60000/60000 [==============================] - 2s 30us/step - loss: 0.0038 - acc: 0.9992 - val_loss: 0.0833 - val_acc: 0.9809
Epoch 19/20
60000/60000 [==============================] - 2s 27us/step - loss: 0.0029 - acc: 0.9993 - val_loss: 0.0838 - val_acc: 0.9818
Epoch 20/20
60000/60000 [==============================] - 2s 28us/step - loss: 0.0025 - acc: 0.9995 - val_loss: 0.0867 - val_acc: 0.9818
Test loss: 0.08674974110474136
Test accuracy: 0.9818
```

과제1 : 영상데이터 결과 분석

Feat. **colab**


compare (1)
0:00:05
0:00:02
10
30

과제1

- 논문
  - Title: **First Order Motion Model for Image Animation**
  - Presented ***NeurIPS***, 2019
  - [arXiv ver.] https://arxiv.org/pdf/2003.00196.pdf

- Colab ver. 데모코드
  - https://colab.research.google.com/github/AliaksandrSiarohin/first-order-model/blob/master/demo.ipynb

과제1

- Colab ver. 데모코드
  - https://colab.research.google.com/github/AliaksandrSiarohin/first-order-model/blob/master/demo.ipynb

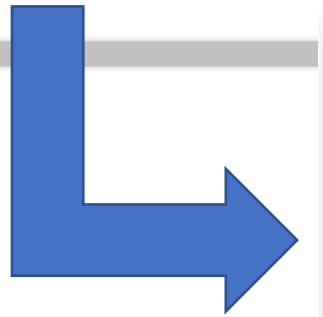
- Colab 기초설명
  - https://colab.research.google.com/notebooks/intro.ipynb
  - Colaboratory(또는 줄여서 'Colab')를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있습니다.
    - 구성 필요 없음
    - GPU 무료 액세스
    - 간편한 공유

과제1 설명

Mount your Google drive folder on Colab

```
[ ]  from google.colab import drive
     drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947

Enter your authorization code:
· · · · · · · · · · · ·
Mounted at /content/gdrive
```



Mount your Google drive folder on Colab

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4

Enter your authorization code:
```

```
from demo import make_animation
from skimage import img_as_ubyte

predictions = make_animation(source_image, driving_video, generator, kp_detector, relative=True)

#save resulting video
imageio.mimsave('../generated.mp4', [img_as_ubyte(frame) for frame in predictions])
#video can be downloaded from /content folder

HTML(display(source_image, driving_video, predictions).to_html5_video())
```

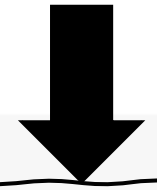
```
100%|██████████| 211/211 [00:07<00:00, 29.03it/s]
```

+ 코드 + 텍스트

In the cell above we use relative keypoint displacement to animate the objects. We can use absolute coordinates instead, but in this way all the object proporions will be inherited from the driving video. For example Putin haircut will be extended to match Trump haircut.

```
[ ] predictions = make_animation(source_image, driving_video, generator, kp_detector, relative=False, adapt_movement_scale=True)
```

과제1 : 아래 **코드 추가**



```
source_image = imageio.imread('/content/gdrive/My Drive/first-order-motion-model/your_img.png')
driving_video = imageio.mimread('/content/gdrive/My Drive/first-order-motion-model/04.mp4', memtest=False)


#Resize image and video to 256x256

source_image = resize(source_image, (256, 256))[..., :3]
driving_video = [resize(frame, (256, 256))[..., :3] for frame in driving_video]

predictions = make_animation(source_image, driving_video, generator, kp_detector, relative=True,
                             adapt_movement_scale=True)

HTML(display(source_image, driving_video, predictions).to_html5_video())
```

```
100%|██████████| 211/211 [00:25<00:00,  8.17it/s]
```

과제1 : 동영상 **화면캡쳐**

```
source_image = imageio.imread('/content/gdrive/My Drive/first-order-motion-model/your_img.png')
driving_video = imageio.mimread('/content/gdrive/My Drive/first-order-motion-model/04.mp4', memtest=False)


#Resize image and video to 256x256

source_image = resize(source_image, (256, 256))[..., :3]
driving_video = [resize(frame, (256, 256))[..., :3] for frame in driving_video]

predictions = make_animation(source_image, driving_video, generator, kp_detector, relative=True,
                             adapt_movement_scale=True)

HTML(display(source_image, driving_video, predictions).to_html5_video())
```

```
100%|██████████| 211/211 [00:25<00:00,  8.17it/s]
```

```
source_image = imageio.imread('/content/gdrive/My Drive/first-order-motion-model/spark.png')
driving_video = imageio.mimread('hinton.mp4', memtest=False)


#Resize image and video to 256x256

source_image = resize(source_image, [illegible], 256))[..., :3]
driving_video = [resize(frame, (256, 256))[..., :3] for frame in driving_video]

predictions = make_animation(source_image, driving_video, generator, kp_detector, relative=True,
                             adapt_movement_scale=True)

HTML(display(source_image, driving_video, predictions).to_html5_video())
```

```
100%|[illegible]| 240/240 [00:28<00:00,  8.30it/s]
```

과제1 요약

- Colab에서 다양한 영상데이터를 테스트해보기
- Requirement:
  - 자연스러운 화면 캡쳐 1
  - 부자연스러운 화면 캡쳐 1
  - 두 화면의 차이를 추론하여 보고

- 주의할 점
  - 두 캡쳐는 1) 동일한 input data로 생성하거나 2) 동일한 driving video로 생성해야 함.
    - Input or driving (target) video 둘 중 하나는 동일한 조건으로 맞출 것
  - 본인의 google drive 계정을 생성하되, 보안 등 만일을 대비하기 위하여 잘 안쓰는 구글계정 사용할 것을 권고함.
  - Input data는 새로운 데이터이어야 함
    - 해당 colab에서 제공하지 않는 영상데이터이며, **사람**임을 식별할 수 있어야 함.