

Bigdata Analysis for Management

ch11. Machine Learning – Supervised Learning

맹 윤 호

Yunho Maeng

Latest update : 2018-11-06

yunho0130@gmail.com

Intro

프로젝트팀의 리더로서 팀원을 이끄는 법

주제를 정할 때, Why?에 대해 이야기를 나눠보세요.



<https://www.youtube.com/watch?v=qp0HfF3SfI4>

삼성전자, 대졸 비 개발 직군에도 파이썬 질문

프로그래밍적 사고를 더 이상 특정 직군의 소양으로만 생각하지 않겠다는 움직임.

- ✓ 코딩을 절대 놓지마라. 그 자체로 진입장벽이다. 놓지만 않으면 언젠가 쉬워진다.

☰ 30th 아시아경제 ▾ 뉴스 ▾

검색

삼성전자, 대졸공채에 '파이썬' 도입... "오픈소스로 생태계 선점"

이번 하반기 공채부터 적용...비 개발 직군에도 파이썬 가능여부 질문

최종수정 2018.11.19 12:08 기사입력 2018.11.19 12:08

◀▶ 가+ 가-



<http://cm.asiae.co.kr/view.htm?no=2018111911582199902>

Python Experts Salaries



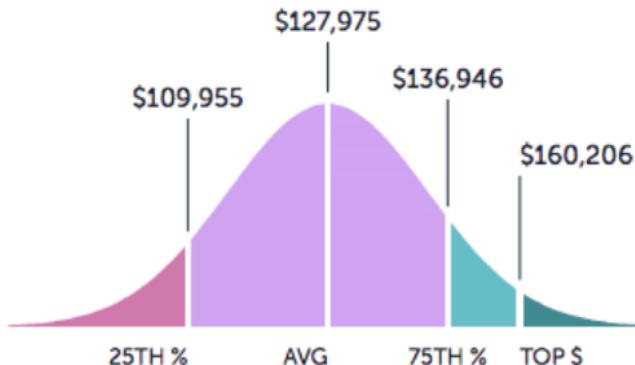
Python Experts Salaries

Data based on salaries for the most recent job postings requiring this expertise



The average Salary for Salaried Employees who know Python is \$127,975.

People that know python earn an average of \$127,975, ranging from \$109,955 at the 25th percentile to \$136,946 at the 75th percentile, with top earners (the top 10%) earning more than \$160,206. Compensation is derived from salaries for the most recent job postings requiring this expertise, including base salary, equity and bonus.



Salaries for Salaried Employees who know
Python

TOTAL COMPENSATION		BREAK DOWN: AVERAGE
25TH PERCENTILE	\$109,955	BASE
AVERAGE	\$127,975	EQUITY
75TH PERCENTILE	\$136,946	BONUS
TOP EARNERS	\$160,206	SIGNING BONUS

http://tinytechreview.com/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4-%EC%97%94%EC%A7%80%EB%8B%88%EC%96%B4-%EC%8A%A4%ED%82%AC%EB%B3%84-%EC%97%B0%EB%B4%89-%EB%B9%84%EA%B5%90/?fbclid=IwAR3SwW3bWT-ZHHG-YJuvojEG4f_KHneXJTlsJTK0TaNoWmheqKH3ywUUgKs

프로젝트 가이드

프로젝트 평가 기준

상대평가이고, 기여도 차등이 있기 때문에 같은 팀이라고 하더라도 성적의 차이가 많이 날 수 있음

- ✓ 데이터의 가치 (데이터를 얼마나 수집하기 어려웠나? 혹은 수집된 데이터의 전처리나 병합에 얼마나 많은 노력을 들였는가?)
- ✓ 데이터 분석 절차 준수 (밸런싱, F1 Score 등)
- ✓ 분석 결과물의 수준 - 분석 난이도 및 보고서의 완성도
- ✓ 분석 결과의 전달력 - 발표 및 시간준수 (5분 발표준비 + 25분 발표 + 10분 질의응답)
- ✓ 결과물의 정확성 및 해석의 적합성
- ✓ 기타고려사항 - 팀원 기여도, 질의응답, 기타 가산점 등



프로젝트 평가 기준

내용 구성

- ✓ 수업시간에 자세히 배우지 않은 분석 방법이 있거나 새로운 분석 방법을 사용했을 경우, 해당 내용을 함께 설명하여 공유.
- ✓ 가장 좋은 성능을 낸 분석 방법과 해당 분석 방법을 도출하기 까지 시도했던 분석 기법들 공유.
- ✓ 각자 어떤 파트를 담당했는지 언급

주의 사항

- ✓ Github에 회의록을 비롯한 프로젝트의 진행 사항을 추적할 수 있도록 반드시 남길 것. 그렇지 않은 경우 상당한 불이익이 갈 것임.
- ✓ 프로젝트 다 진행하고 마지막에만 Github에 소드코드 올려서 제출하지 말라는 뜻.
- ✓ 그 때 그 때 Commit하고 해당 내역을 통해 누가 얼마나 기여했는지 추적이 가능하도록 할 것.
- ✓ 출처를 밝히지 않은 외부코드를 본인들의 분석에 사용할 경우 0점 처리 (반드시, 출처를 명시할 것)

Cloud Tool: Google Colaboratory

For Window Users

Google Colaboratory

Python Jupyter Notebook을 구글 드라이브에서 실행한다고 생각하면 편함.

- ✓ GPU 사용까지 무료로 지원 (단, 무료 사용자가 많을 수록 대기시간이 길어짐)
- ✓ 맥이 없는 윈도우 사용자를 위한 실행환경

The screenshot shows the Google Colaboratory interface. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by the URL 'colab.research.google.com'. On the right of the bar are 'SHARE', 'Sign in', and other account-related buttons. Below the bar, the main content area has a title 'Hello, Colaboratory' with a 'CO' logo icon. It features a 'COPY TO DRIVE' button and 'CONNECT' and 'EDITING' dropdowns. A sidebar on the left contains a 'Table of contents' section with links to 'Getting Started', 'Highlighted Features', 'TensorFlow execution', 'GitHub', 'Visualization', 'Forms', 'Examples', and 'Local runtime support'. There's also a '+ SECTION' button. The main content area displays a 'Welcome to Colaboratory!' message with the 'CO' logo, stating that Colaboratory is a free Jupyter notebook environment running entirely in the cloud. It includes a link to the 'FAQ'. Below this, the 'Getting Started' section lists several topics: Overview of Colaboratory, Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage, Importing libraries and installing dependencies, Using Google Cloud BigQuery, Forms, Charts, Markdown, & Widgets, TensorFlow with GPU, TensorFlow with TPU, Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow, and Using Colab with GitHub. At the bottom of the content area, there's a 'Highlighted Features' section with a 'Seedbank' link, which is described as a place to discover interactive Colab notebooks.

<https://colab.research.google.com/>

Google Colaboratory

Remind GPGPU

- ✓ Nvidia 그래픽 카드를 일반 연산 목적용으로 사용할 수 있도록 SW레벨에서 최적화

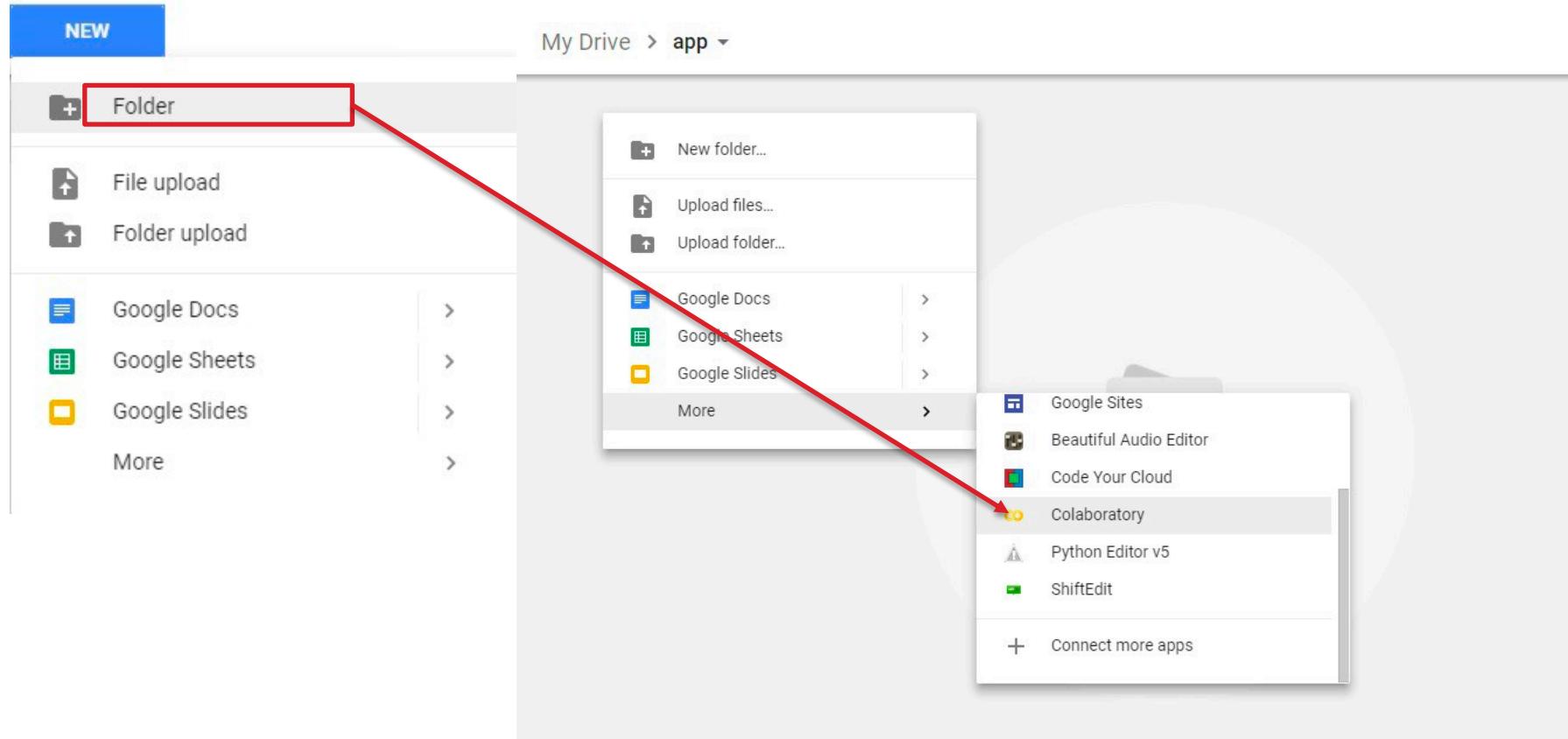


<https://www.youtube.com/watch?v=-P28LKWTzrl>

Google Colaboratory

Google Drive에서 Colaboratory 생성 및 실행

- ✓ 구글 드라이브에 빈 폴더를 생성하고 Colaboratory 파일 생성
- ✓ 기존의 Jupyter Notebook과 유사하게 사용가능함

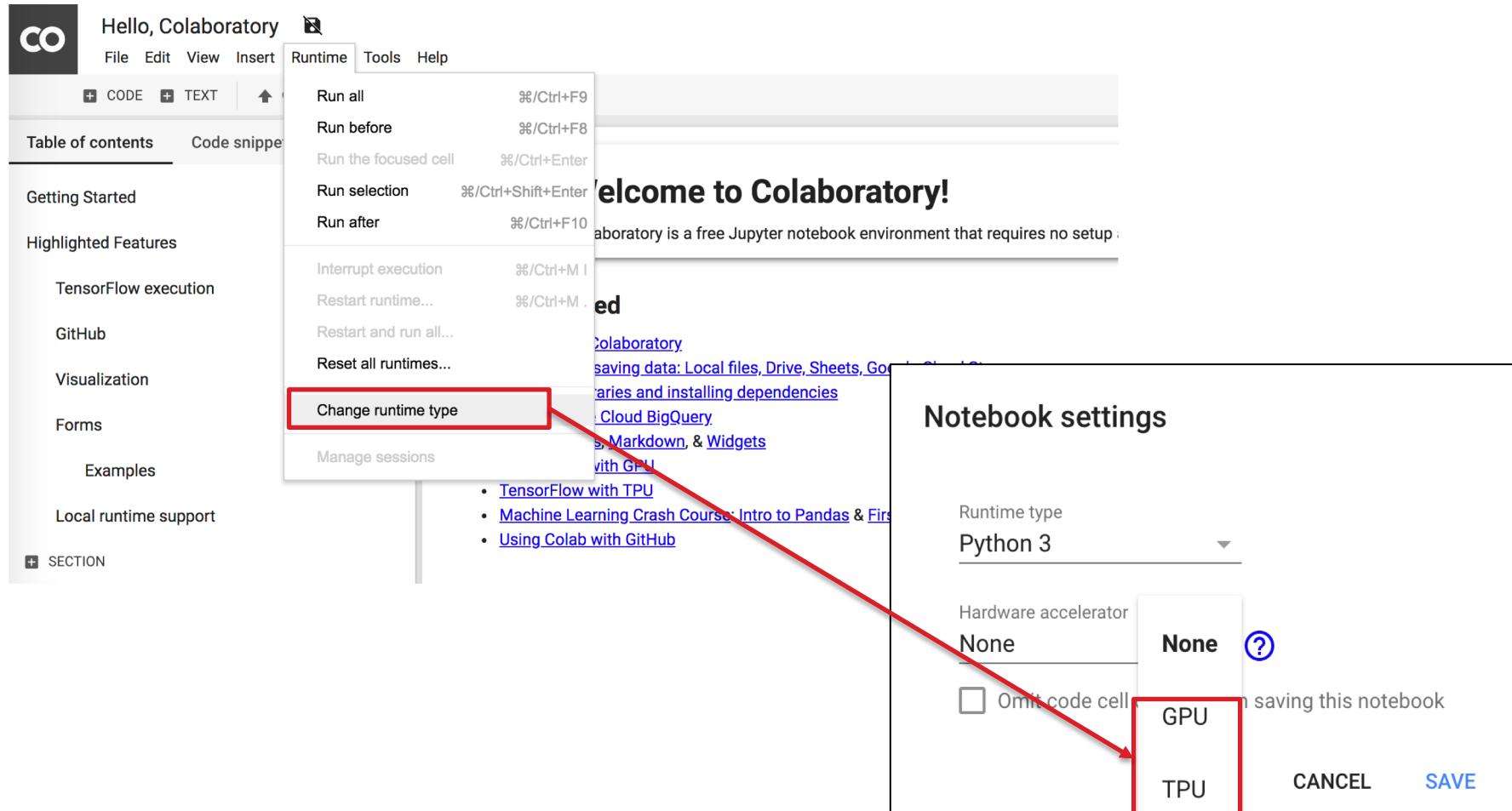


<https://colab.research.google.com/>

Google Colaboratory

하드웨어 가속인 GPU 및 TPU 설정방법

- ✓ Runtime > Change Runtime Type > Notebook settings > Hardware Accelerator > GPU 혹은 TPU



Google Colaboratory

TPU는 구글이 자체적으로 개발한 딥러닝을 위한 하드웨어 가속 연산 유닛.

- ✓ 그래픽 카드를 일반 연산 목적용으로 SW레벨에서 최적화 하는 접근과 달리, 하드웨어 설계부터 딥러닝 연산에 최적화 되었음. <https://cloud.google.com/tpu/>

The screenshot shows a web browser window for cloud.google.com. The navigation bar includes links for Google Cloud, Why Google, Products (which is underlined), Solutions, Pricing, Security, Documentation, Customers, Partners, Support, a search icon, Console, and Sign in. Below the navigation bar, there's a header for 'AI & Machine Learning Products' with 'Contact sales' and 'Try free' buttons. The main content area features a section titled 'CLOUD TPU' with the subtext 'Train and run machine learning models faster than ever before.' It includes 'VIEW DOCUMENTATION' and 'GET STARTED' buttons. A large image of a TPU Accelerator board is shown with several orange heat sinks and grey cables. Below this image, the text 'Accelerated machine learning' is followed by a detailed paragraph about how machine learning has enabled breakthroughs across business and research problems, and how the TPU was built from the ground up for machine learning.

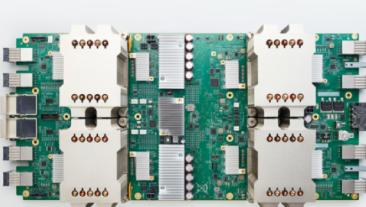
Machine learning (ML) has enabled breakthroughs across a variety of business and research problems, from strengthening network security to improving the accuracy of medical diagnoses. Because training and running deep learning models can be computationally demanding, we built the Tensor Processing Unit (TPU), an ASIC designed from the ground up for machine learning that powers several of our major products, including [Translate](#), [Photos](#), [Search](#), [Assistant](#), and [Gmail](#). Cloud TPU empowers businesses everywhere to access this accelerator technology to speed up their machine learning workloads on Google Cloud.

Google Colaboratory

TPU는 구글이 자체적으로 개발한 딥러닝을 위한 하드웨어 가속 연산 유닛.

- ✓ TPU는 그 자체로 슈퍼컴퓨터급 연산 속도를 갖추었음.
- ✓ 우리나라의 자체기술로 서울대에서 개발한 슈퍼컴퓨터인 천동(2012)은 106.8 TFlops의 성능을 지녔음

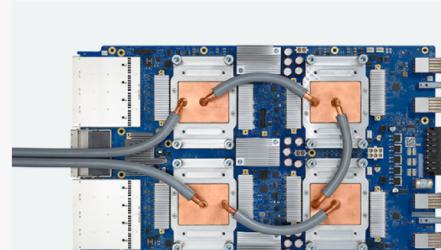
Cloud TPU offering



Cloud TPU v2

180 teraflops

64 GB High Bandwidth Memory (HBM)



Cloud TPU v3 Beta

420 teraflops

128 GB HBM



Cloud TPU v2 Pod Alpha

11.5 petaflops

4 TB HBM

2-D toroidal mesh network

Why do I need a local machine setup?

여러분들이 앞으로 기업이나 기관에서 분석하게 될 매우 민감한 데이터가 될 가능성이 높습니다.

- ✓ 금융권의 경우, 법률적으로도 2019년에서야 클라우드 이용이 허락되었으나, 개인정보 역외이전에 관한 법률을 추가로 검토해야할 뿐만 아니라, 기업 사내 보안과도 연계되는 중요한 이슈.
- ✓ 클라우드에 데이터를 올린다고 할 때, 데이터의 격리가 항상 보장되는 것이 아니며, 법률적으로 사용이 가능해진 것과는 다른 차원으로 기업 외부로 유출이 민감한 데이터의 경우 로컬에서의 분석이 강제될 수 밖에 없는 상황.
- ✓ 로컬 머신은 이미 구매한 자원이고, 클라우드 사용료는 추가적으로 지출되어야 할 부분

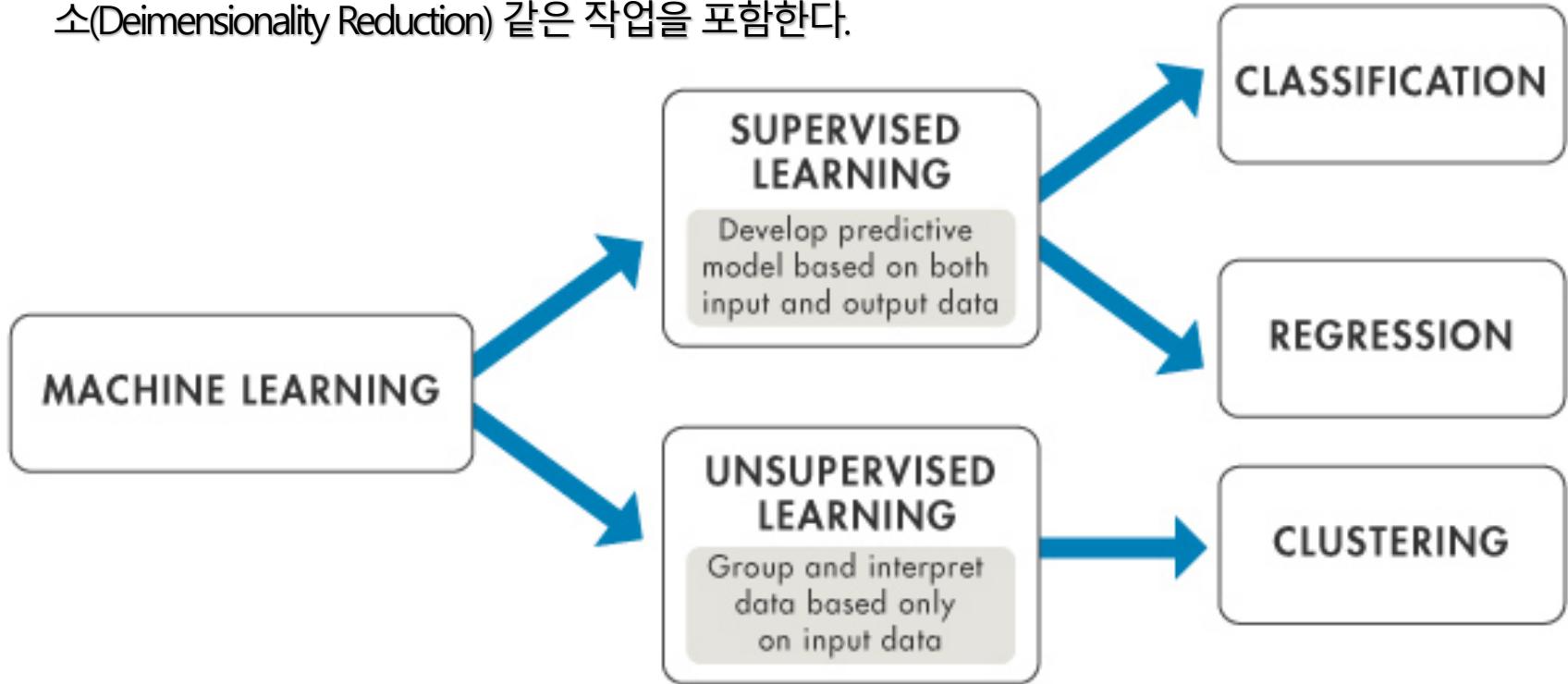


Machine Learning

Machine Learning의 구분

머신러닝이란, 데이터 모델을 구축하는 하나의 수단으로서 조정 가능한 모수(Parameter)를 통해 데이터로부터 모델을 학습하는 방식을 말한다. [1]

- ✓ 지도학습Supervised Learning: 데이터의 측정된 특징(Feature)과 데이터와 관련된 레이블(Label) 사이의 관계를 모델링 하는 것이다. 대표적으로 분류(Classification)와 회귀(Regression)로 나뉜다.
- ✓ 비지도 학습Unsupervised Learning: 레이블을 참조하지 않고 데이터 세트의 특징을 모델링 하는 것으로, 종종 데이터 세트가 스스로 말하게 하는 것이다. 대표적으로 군집화(Clustering)와 차원축소(Deimensionality Reduction) 같은 작업을 포함한다.



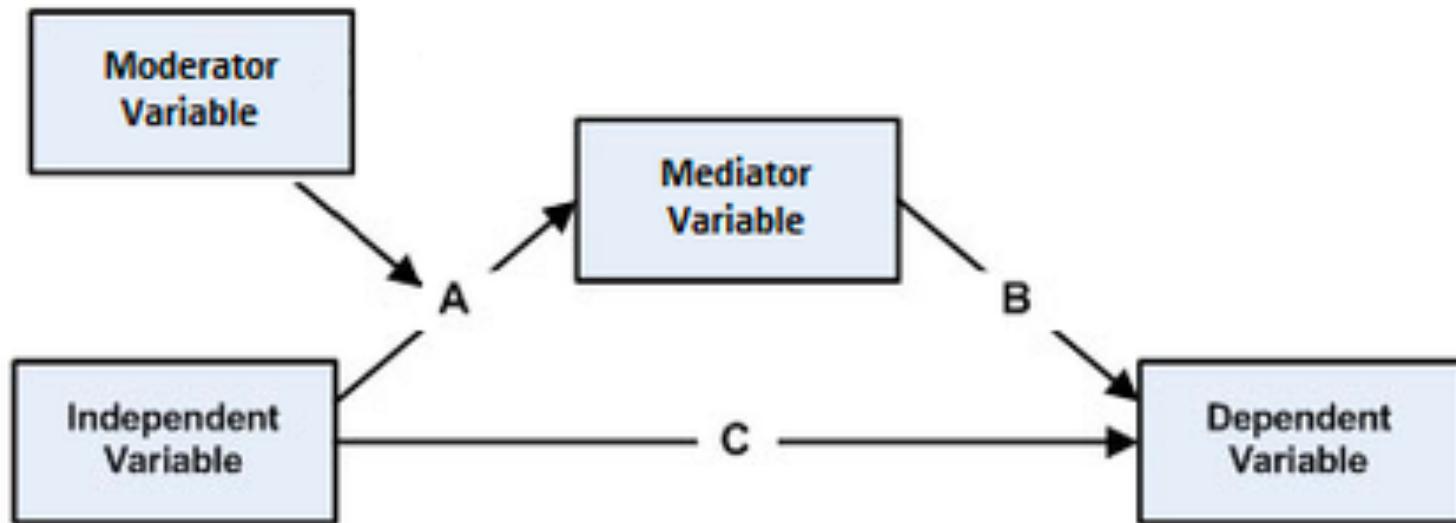
[1] VanderPlas, J. (2016). Python data science handbook: essential tools for working with data.
" O'Reilly Media, Inc.".

[2] <https://kr.mathworks.com/help/stats/machine-learning-in-matlab.html>

Machine Learning의 구분

지도학습(Supervised Learning)과 비지도학습(Unsupervised Learning)은 목표변수(Target Variable)의 존재 유무로 나뉜다.

- ✓ 목표변수(Target Variable)는 레이블(Label)혹은 종속변수(Dependent Variable)라고도 한다.

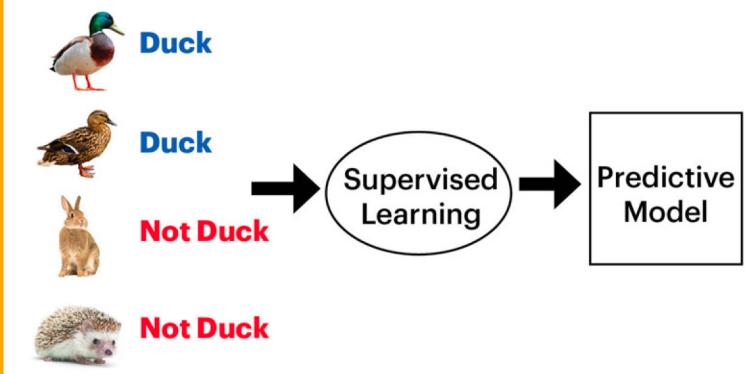


[http://www.wikiwand.com/en/Mediation_\(statistics\)](http://www.wikiwand.com/en/Mediation_(statistics))

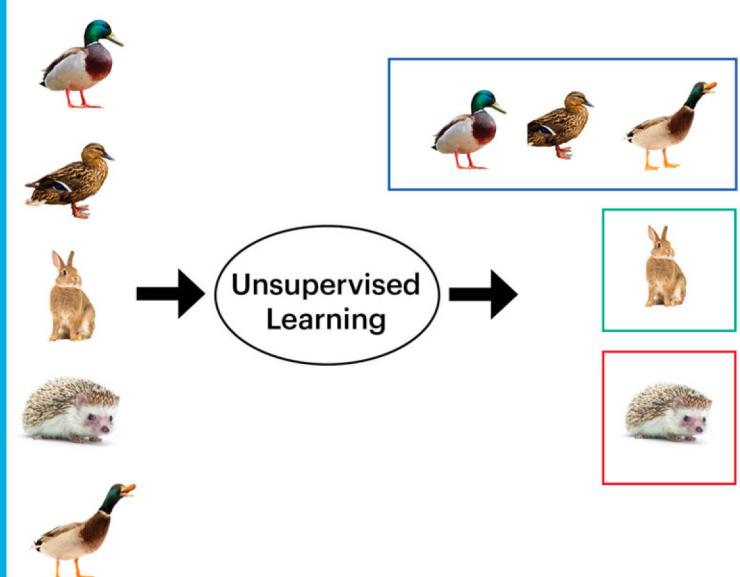
Machine Learning의 구분

지도학습(Supervised Learning)과 비지도학습(Unsupervised Learning)으로 나뉘는 머신러닝(Machine Learning)의 갈래

Supervised Learning (Classification Algorithm)

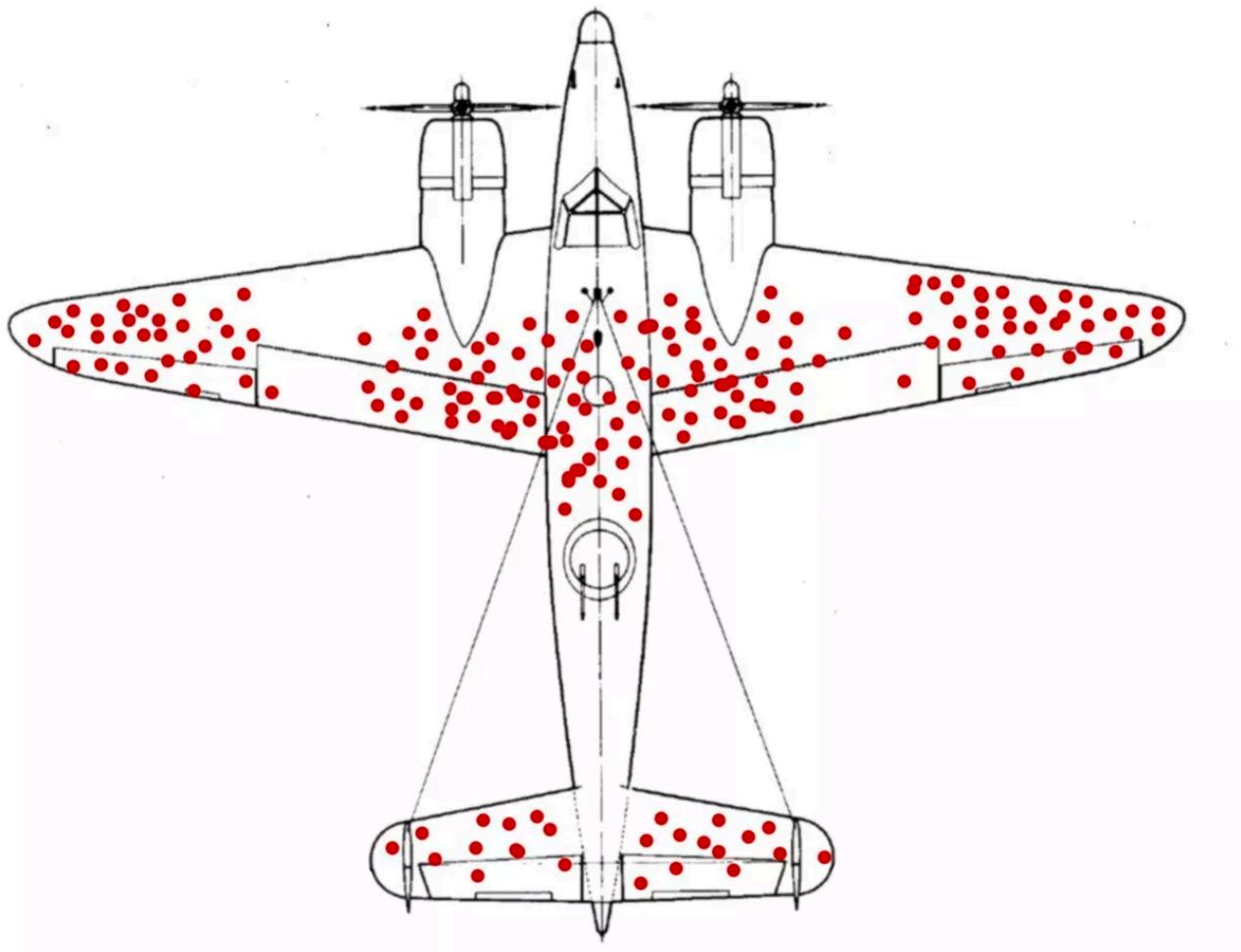


Unsupervised Learning (Clustering Algorithm)



Survivorship Bias (생존자 편향의 오류)

때는 2차 세계대전, 당신은 전장에서 돌아온 전투기들의 외상을 분석하여 취약 부분을 보강하는 계획을 담당했습니다. 어디에 장갑을 보강해야 할까요?

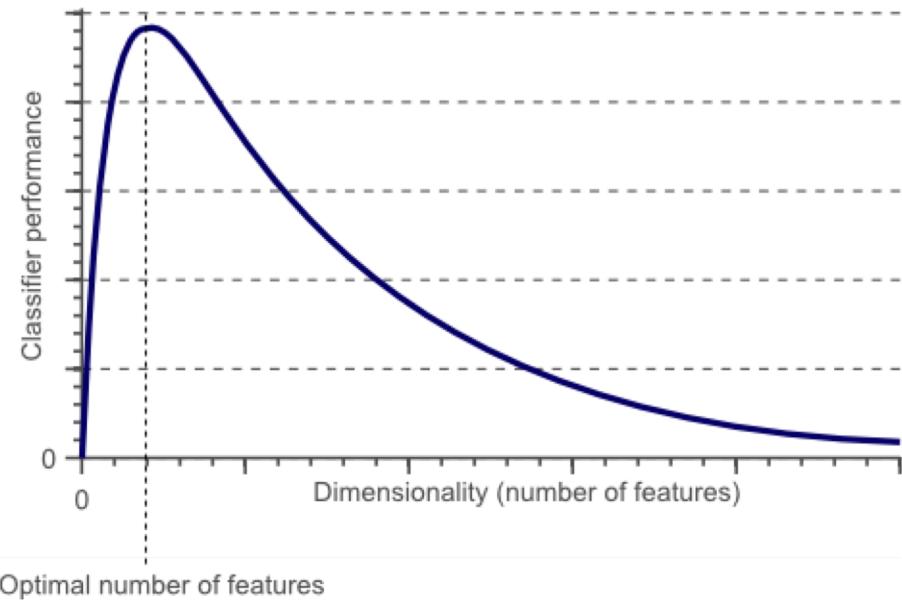
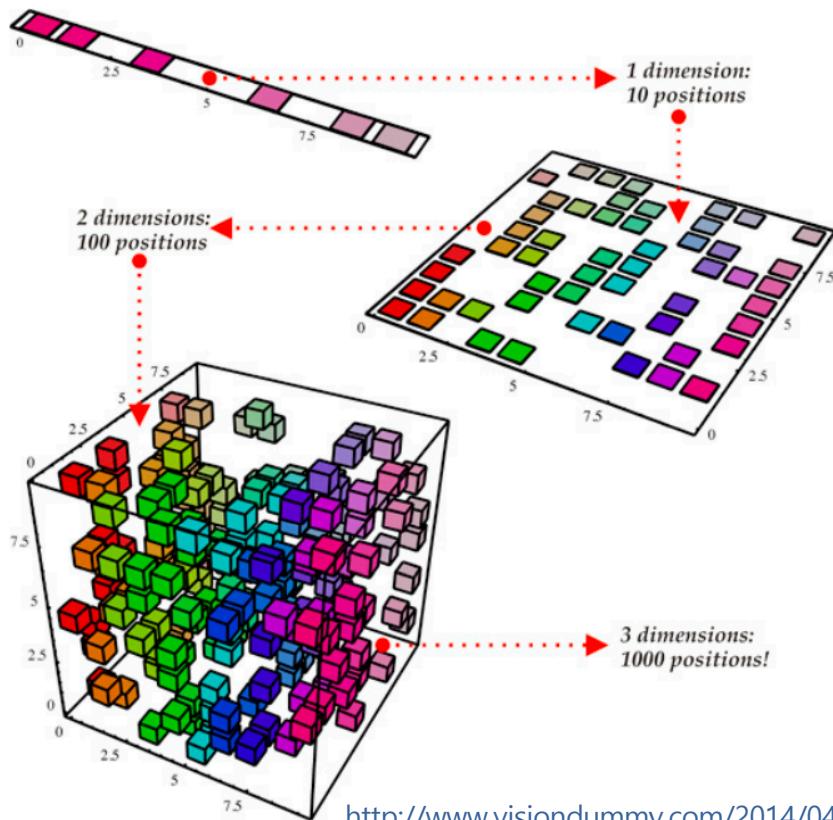


<http://www.andrewahn.co/silicon-valley/survivorship-bias/>

Curse of dimensionality (차원의 저주)

예측에 필요한 요소(Features)들이 많아질 수록, 예측의 성능(Performance)이 급격히 떨어지는 현상.

- ✓ 정확도가 떨어지는 것이 아니라, 성능(Performance)임에 주의
- ✓ 최적의 숫자와 최적의 Feature 값을 찾는 과정을 Feature Engineering이라고 한다.
- ✓ 서로 상관성이 있는 변수를 제거하거나, 후행변수와 선행변수, 파생변수를 구분하고 통제할 줄 알아야 함.

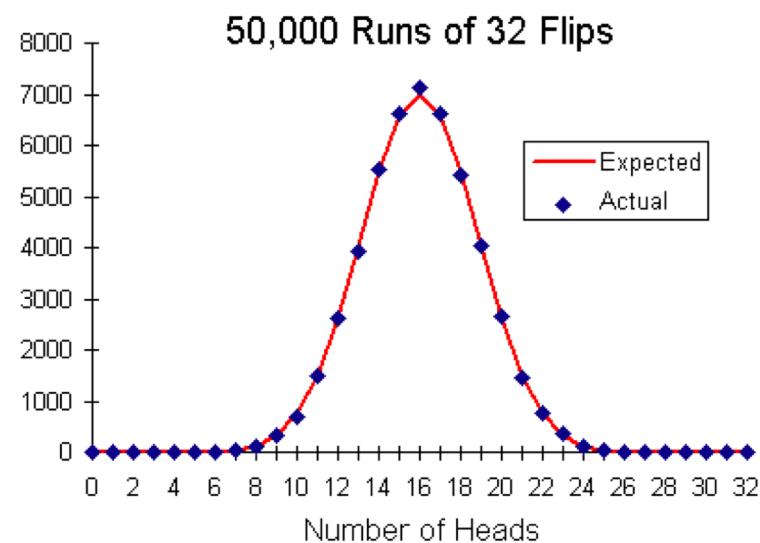


<http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/>
<https://towardsdatascience.com/that-cursing-dimensionality-ac317fb0fdcc>

그 외 데이터 분석시 주의할 점

Patterns May Not Represent Any Underlying Rule 패턴은 근본적인 법칙을 대표하지 않을 수 있다.

- ✓ Figures don't lie, but liars do figure. 통계는 거짓말을 하지 않지만, 거짓말쟁이들은 통계를 쓸 수 있다.
- ✓ So many ways to construct patterns exist that any random set of data points reveals one if examined long enough. 데이터가 많고 오래되면 수많은 패턴이 생김
- ✓ Learning things that aren't true 사실이 아닌 것 학습. 이게 제일 문제. 복잡한 방법일수록 주의해야 함 "Garbage in, Garbage out"
- ✓ Learning things that are true, but not useful. 사실이지만 유용하지 않은 것 학습
- ✓ The challenge for data miners is to figure out which patterns are useful and which are not 앞서 동전의 앞면이 5번 나와도 확률은 50:50이다. 데이터 분석가의 일은 예측력 있는 패턴과 없는 패턴을 구분할 수 있어야 함. e.g. 미국 리그가 월드시리즈에서 우승하면 공화당이 집권한다. (예측력 없는 패턴)



Supervised Learning

분류 Classification

~/code/06_00_Figure_Code.ipynb

분류Classification은 레이블이 이산형 범주일 때, 데이터를 구분하는 방식이다.

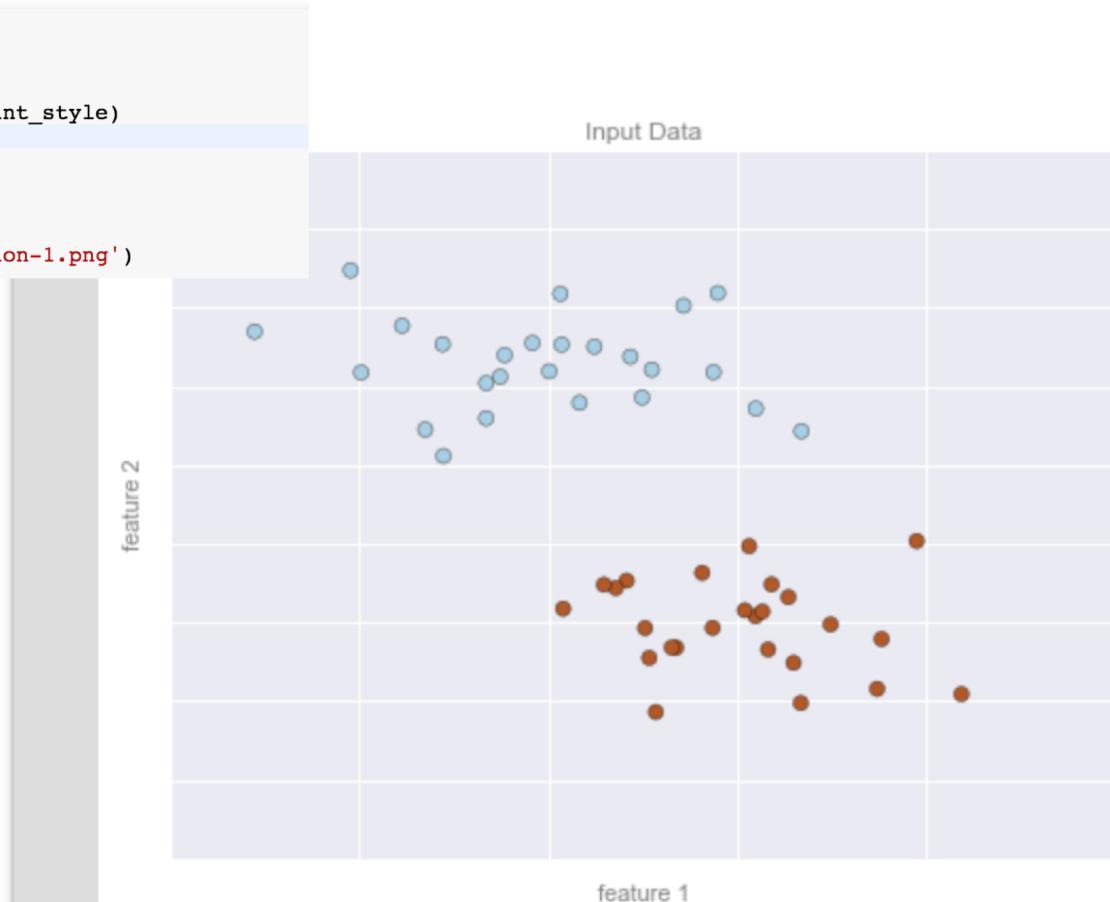
- ✓ 다음과 같은 데이터가 있다고 하자. x, y 를 Feature로 갖고, 파란색과 빨간색을 label로 같은 데이터 셋이다.

Classification Example Figure 1

```
# plot the data
fig, ax = plt.subplots(figsize=(8, 6))
point_style = dict(cmap='Paired', s=50)
ax.scatter(X[:, 0], X[:, 1], c=y, **point_style)

# format plot
format_plot(ax, 'Input Data')
ax.axis([-1, 4, -2, 7])

fig.savefig('figures/05.01-classification-1.png')
```



분류 Classification

~/code/06_00_Figure_Code.ipynb

분류Classification은 레이블이 이산형 범주일 때, 데이터를 구분하는 방식이다.

- ✓ 간단하게 하나의 직선을 그려서, 파란색과 빨간색을 분류할 수 있다. 이제 새로운 데이터 셋을 예측해보자.

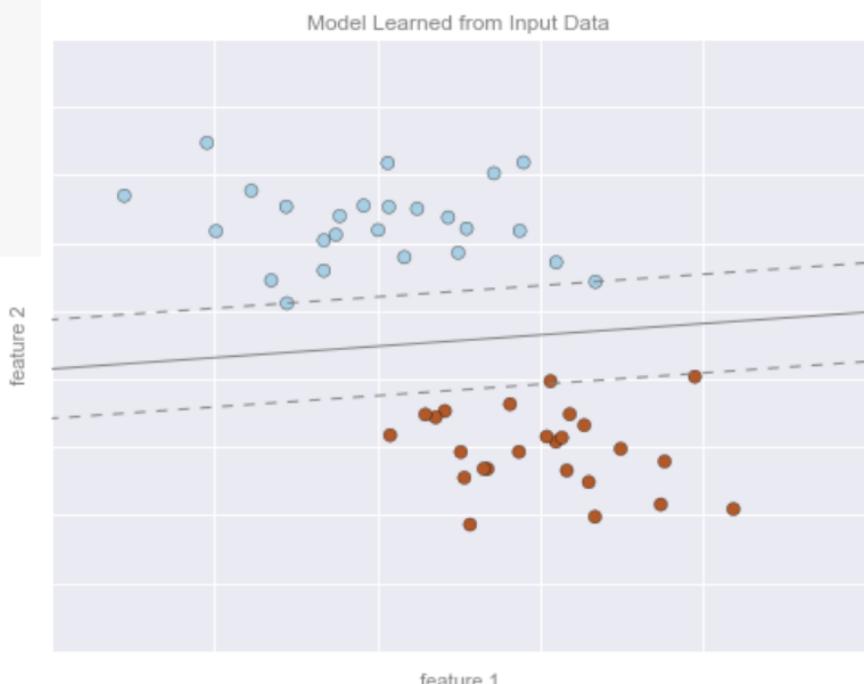
Classification Example Figure 2

```
[ ] # Get contours describing the model
xx = np.linspace(-1, 4, 10)
yy = np.linspace(-2, 7, 10)
xy1, xy2 = np.meshgrid(xx, yy)
Z = np.array([clf.decision_function([t])
             for t in zip(xy1.flat, xy2.flat)]).reshape(xy1.shape)

# plot points and model
fig, ax = plt.subplots(figsize=(8, 6))
line_style = dict(levels = [-1.0, 0.0, 1.0],
                  linestyles = ['dashed', 'solid', 'dashed'],
                  colors = 'gray', linewidths=1)
ax.scatter(X[:, 0], X[:, 1], c=y, **point_style)
ax.contour(xy1, xy2, Z, **line_style)

# format plot
format_plot(ax, 'Model Learned from Input Data')
ax.axis([-1, 4, -2, 7])

fig.savefig('figures/05.01-classification-2.png')
```



분류 Classification

~/code/06_00_Figure_Code.ipynb

레이블이 없는 데이터에 학습시킨 모델을 적용시켜 레이블을 생성할 수 있다. 이를 예측이라 한다.

Classification Example Figure 3

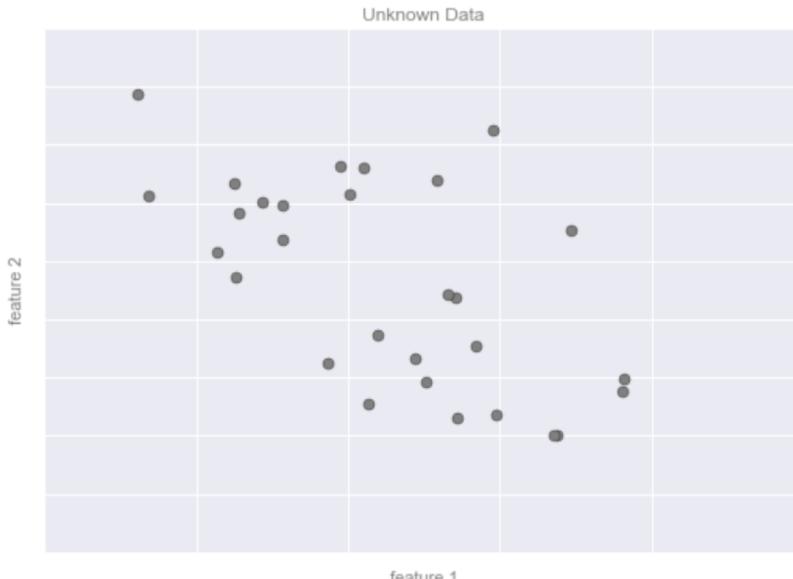
```
[ ] # plot the results
fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)

ax[0].scatter(X2[:, 0], X2[:, 1], c='gray', **point_style)
ax[0].axis([-1, 4, -2, 7])

ax[1].scatter(X2[:, 0], X2[:, 1], c=y2, **point_style)
ax[1].contour(xyl, xy2, z, **line_style)
ax[1].axis([-1, 4, -2, 7])

format_plot(ax[0], 'Unknown Data')
format_plot(ax[1], 'Predicted Labels')

fig.savefig('figures/05.01-classification-3.png')
```



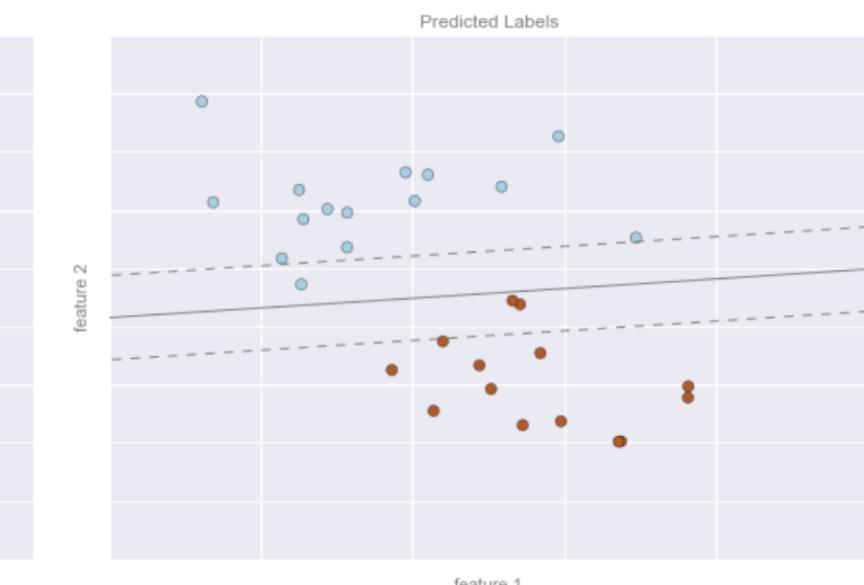
```
[ ] from sklearn.datasets.samples_generator import make_blobs
from sklearn.svm import SVC

# create 50 separable points
X, y = make_blobs(n_samples=50, centers=2,
                   random_state=0, cluster_std=0.60)

# fit the support vector classifier model
clf = SVC(kernel='linear')
clf.fit(X, y)

# create some new points to predict
X2, _ = make_blobs(n_samples=80, centers=2,
                    random_state=0, cluster_std=0.80)
X2 = X2[50:]

# predict the labels
y2 = clf.predict(X2)
```



선형회귀 Linear Regression

~/code/06_00_Figure_Code.ipynb

회귀Regression은 분류와는 달리, 레이블Label이 연속형인 숫자인 경우를 의미한다.

- ✓ 아래의 데이터를 살펴보면, 레이블Label의 색상이 이전과는 다르게 다채로운 색상으로 구성되어 있는 것을 확인할 수 있다.

▼ Regression Example Figures

Figure Context

The following code generates the figures from the regression section.

```
[13] from sklearn.linear_model import LinearRegression

# Create some data for the regression
rng = np.random.RandomState(1)

X = rng.randn(200, 2)
y = np.dot(X, [-2, 1]) + 0.1 * rng.randn(X.shape[0])

# fit the regression model
model = LinearRegression()
model.fit(X, y)

# create some new points to predict
X2 = rng.randn(100, 2)

# predict the labels
y2 = model.predict(X2)
```

▼ Regression Example Figure 1

```
[14] # plot data points
fig, ax = plt.subplots()
points = ax.scatter(X[:, 0], X[:, 1], c=y, s=50,
                     cmap='viridis')

# format plot
format_plot(ax, 'Input Data')
ax.axis([-4, 4, -3, 3])

fig.savefig('figures/05.01-regression-1.png')
```



선형회귀 Linear Regression

~/code/06_00_Figure_Code.ipynb

레이블이 여러개이므로, 선을 하나만 그을 수 없기 때문에, 선의 집합인 평면으로 나누어보자.

▼ Regression Example Figure 2

```
from mpl_toolkits.mplot3d.art3d import Line3DCollection

points = np.hstack([X, y[:, None]]).reshape(-1, 1, 3)
segments = np.hstack([points, points])
segments[:, 0, 2] = -8

# plot points in 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:, 0], X[:, 1], y, c=y, s=35,
           cmap='viridis')
ax.add_collection3d(Line3DCollection(segments, colors='gray', alpha=0.2))
ax.scatter(X[:, 0], X[:, 1], -8 + np.zeros(X.shape[0]), c=y, s=10,
           cmap='viridis')

# format plot
ax.patch.set_facecolor('white')
ax.view_init(elev=20, azim=-70)
ax.set_zlim3d(-8, 8)
ax.xaxis.set_major_formatter(plt.NullFormatter())
ax.yaxis.set_major_formatter(plt.NullFormatter())
ax.zaxis.set_major_formatter(plt.NullFormatter())
ax.set(xlabel='feature 1', ylabel='feature 2', zlabel='label')

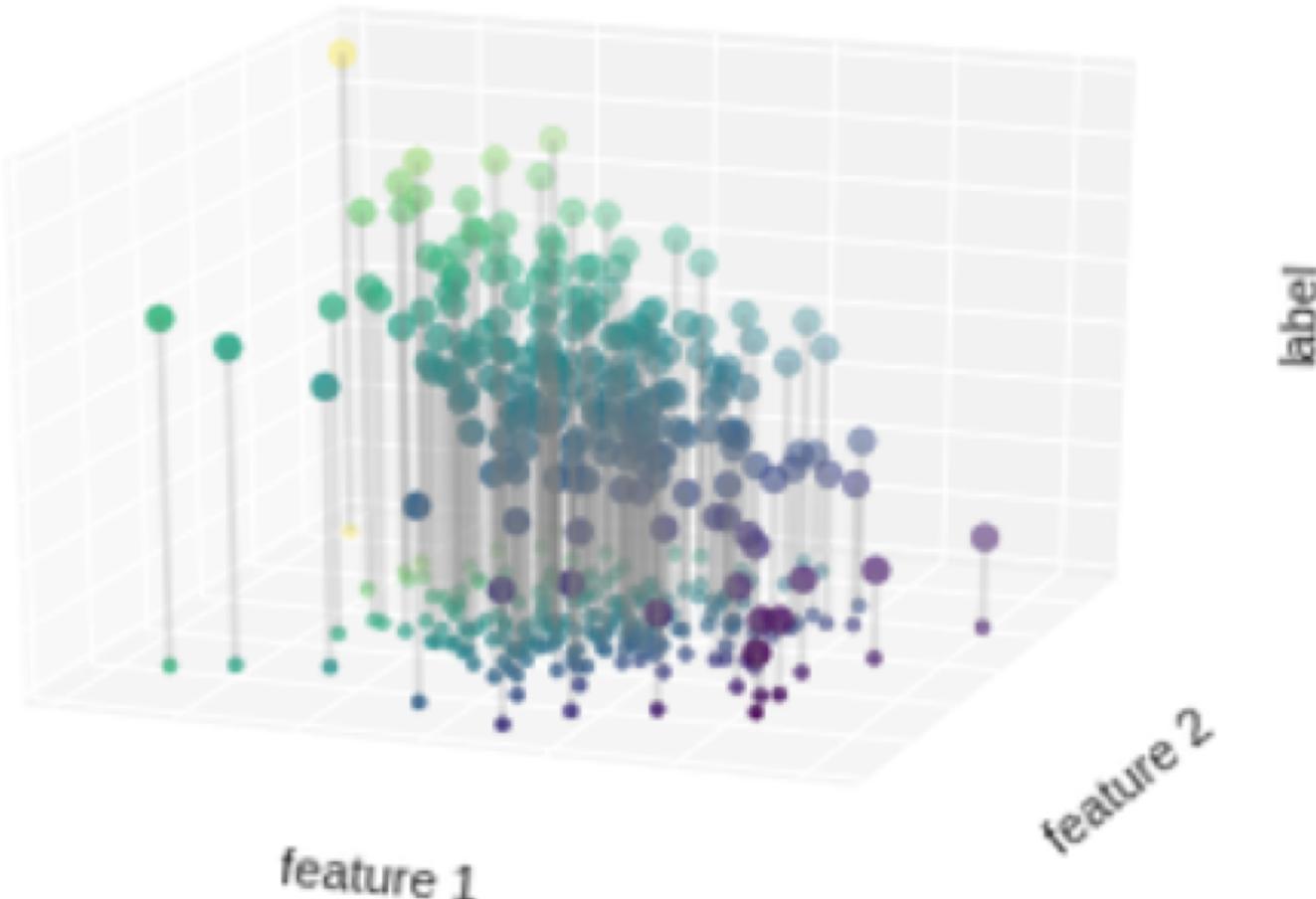
# Hide axes (is there a better way?)
ax.w_xaxis.line.set_visible(False)
ax.w_yaxis.line.set_visible(False)
ax.w_zaxis.line.set_visible(False)
for tick in ax.w_xaxis.get_ticklines():
    tick.set_visible(False)
for tick in ax.w_yaxis.get_ticklines():
    tick.set_visible(False)
for tick in ax.w_zaxis.get_ticklines():
    tick.set_visible(False)

fig.savefig('figures/05.01-regression-2.png')
```

선형회귀 Linear Regression

~/code/06_00_Figure_Code.ipynb

점들의 집합을 가로지르는 평면을 모델으로 데이터를 잘 구분할 수 있다고 할 수 있다.



선형회귀 Linear Regression

~/code/06_00_Figure_Code.ipynb

결과적으로, 2차원 평면으로 옮겨오면 마치 이미지 필터처럼 각 데이터들을 구분하게 된다.

▼ Regression Example Figure 3

```
▶ from matplotlib.collections import LineCollection

# plot data points
fig, ax = plt.subplots()
pts = ax.scatter(X[:, 0], X[:, 1], c=y, s=50,
                  cmap='viridis', zorder=2)

# compute and plot model color mesh
xx, yy = np.meshgrid(np.linspace(-4, 4),
                      np.linspace(-3, 3))
Xfit = np.vstack([xx.ravel(), yy.ravel()]).T
yfit = model.predict(Xfit)
zz = yfit.reshape(xx.shape)
ax.pcolorfast([-4, 4], [-3, 3], zz, alpha=0.5,
              cmap='viridis', norm=pts.norm, zorder=1)

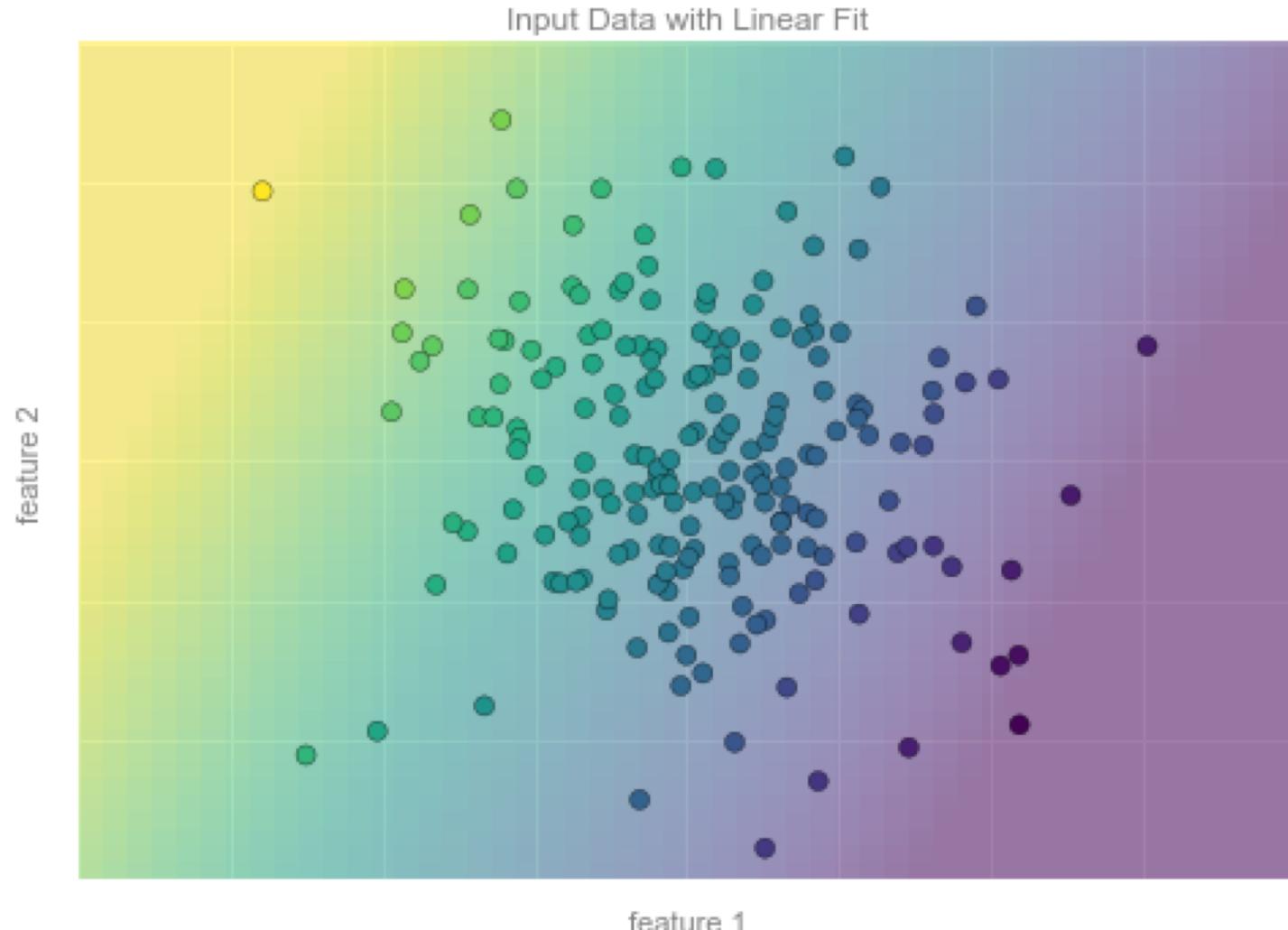
# format plot
format_plot(ax, 'Input Data with Linear Fit')
ax.axis([-4, 4, -3, 3])

fig.savefig('figures/05.01-regression-3.png')
```

선형회귀 Linear Regression

~/code/06_00_Figure_Code.ipynb

결과적으로, 2차원 평면으로 옮겨오면 마치 이미지 필터처럼 각 데이터들을 구분하게 된다.



선형회귀 Linear Regression

~/code/06_00_Figure_Code.ipynb

해당 필터를 다음의 새로운 데이터에 적용해 볼 수 있다.

Regression Example Figure 4

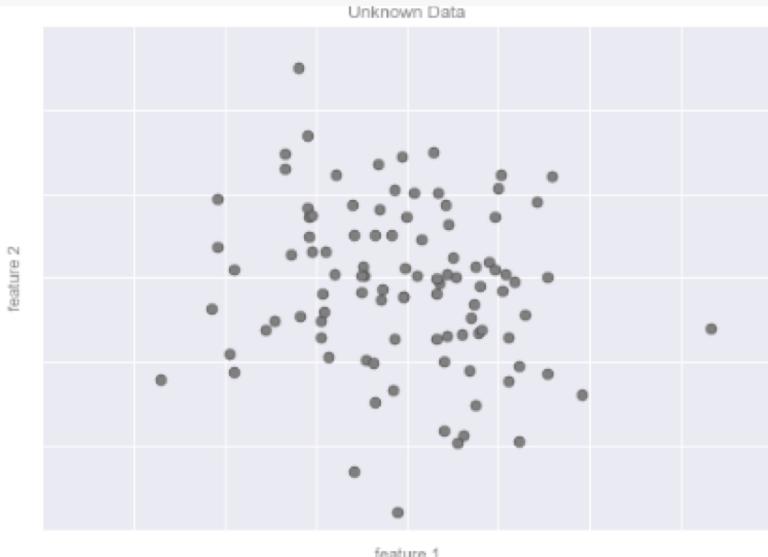
```
[ ] # plot the model fit
fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)

ax[0].scatter(X2[:, 0], X2[:, 1], c='gray', s=50)
ax[0].axis([-4, 4, -3, 3])

ax[1].scatter(X2[:, 0], X2[:, 1], c=y2, s=50,
               cmap='viridis', norm=pts.norm)
ax[1].axis([-4, 4, -3, 3])

# format plots
format_plot(ax[0], 'Unknown Data')
format_plot(ax[1], 'Predicted Labels')

fig.savefig('figures/05.01-regression-4.png')
```



```
[4] from sklearn.linear_model import LinearRegression

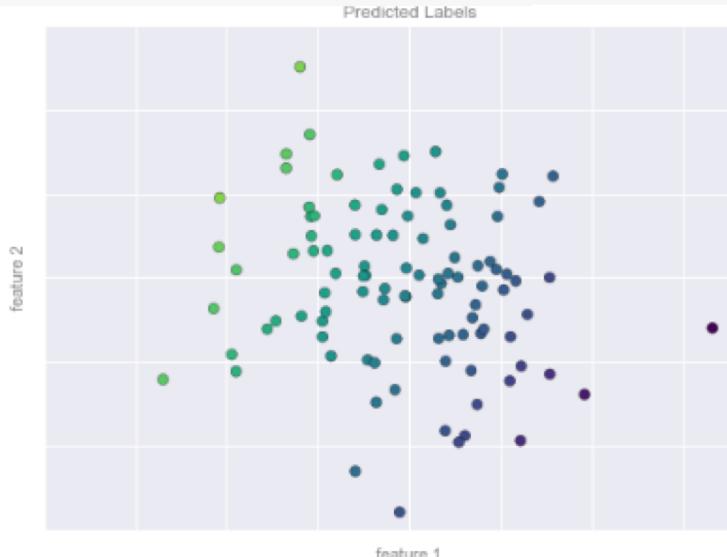
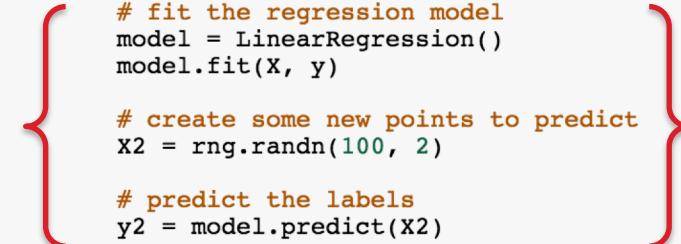
# Create some data for the regression
rng = np.random.RandomState(1)

X = rng.randn(200, 2)
y = np.dot(X, [-2, 1]) + 0.1 * rng.randn(X.shape[0])

# fit the regression model
model = LinearRegression()
model.fit(X, y)

# create some new points to predict
X2 = rng.randn(100, 2)

# predict the labels
y2 = model.predict(X2)
```



베이지안 정리

~/code/05_02_Introducing_Scikit_Learn.ipynb

각 속성(Features)들이 독립일 때, 기존 데이터에서의 조건부 확률을 이용한 새로운 데이터의 분류

- ✓ 한 학교의 학생이 여학생인 확률이 $P(A)$ 라고 하고, 학생이 키가 160이 넘는 확률을 $P(B)$ 라고 했을 때, 여학생 중에서, 키가 160이 넘는 확률은 B 의 조건부 확률이 되며 $P(B|A)$ 로 표현 한다.

$$P(A | B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(B | A)P(A)}{P(B)}$$

Bayesian Classification

Naive Bayes classifiers are built on Bayesian classification methods. These rely on Bayes's theorem, which is an equation describing the relationship of conditional probabilities of statistical quantities. In Bayesian classification, we're interested in finding the probability of a label given some observed features, which we can write as $P(L | \text{features})$. Bayes's theorem tells us how to express this in terms of quantities we can compute more directly:

$$P(L | \text{features}) = \frac{P(\text{features} | L)P(L)}{P(\text{features})}$$

If we are trying to decide between two labels—let's call them L_1 and L_2 —then one way to make this decision is to compute the ratio of the posterior probabilities for each label:

$$\frac{P(L_1 | \text{features})}{P(L_2 | \text{features})} = \frac{P(\text{features} | L_1)}{P(\text{features} | L_2)} \frac{P(L_1)}{P(L_2)}$$

All we need now is some model by which we can compute $P(\text{features} | L_i)$ for each label. Such a model is called a *generative model* because it specifies the hypothetical random process that generates the data. Specifying this generative model for each label is the main piece of the training of such a Bayesian classifier. The general version of such a training step is a very difficult task, but we can make it simpler through the use of some simplifying assumptions about the form of this model.

This is where the "naive" in "naive Bayes" comes in: if we make very naive assumptions about the generative model for each label, we can find a rough approximation of the generative model for each class, and then proceed with the Bayesian classification. Different types of naive Bayes classifiers rest on different naive assumptions about the data, and we will examine a few of these in the following sections.

We begin with the standard imports:

Gaussian Naive Bayes

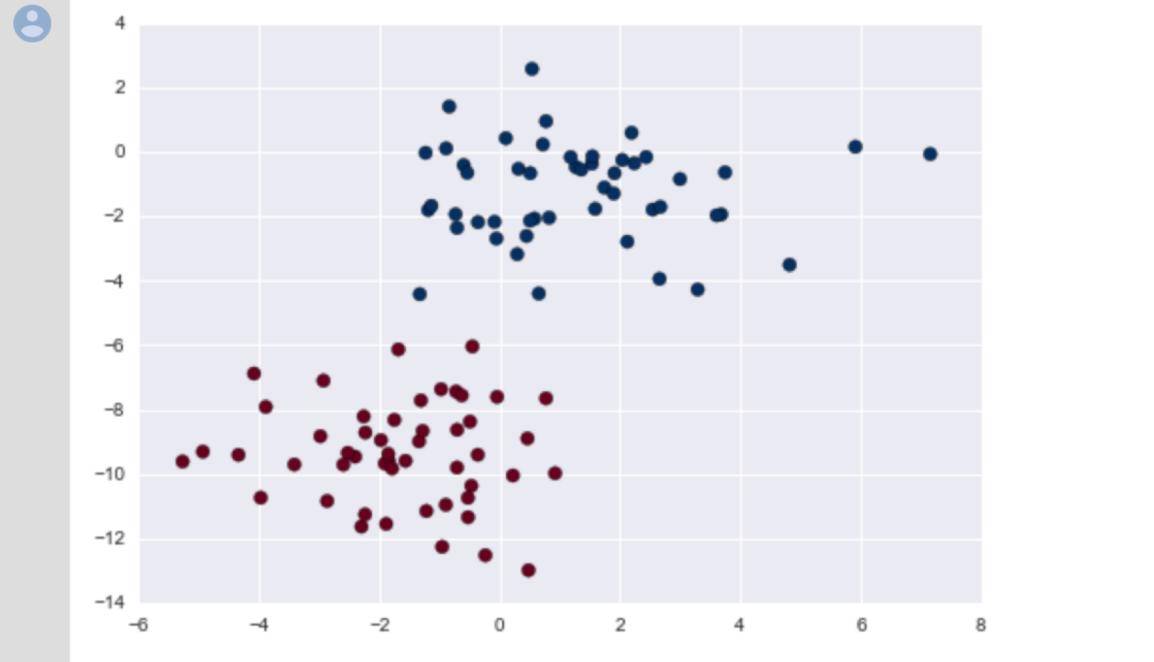
~/code/05.05-Naive-Bayes.ipynb

가우시안 나이브 베이지안 분석기법은, 레이블링 된 Feature들이 가우시안 분포를 따른다는 가정하에 새로운 데이터를 분류한다.

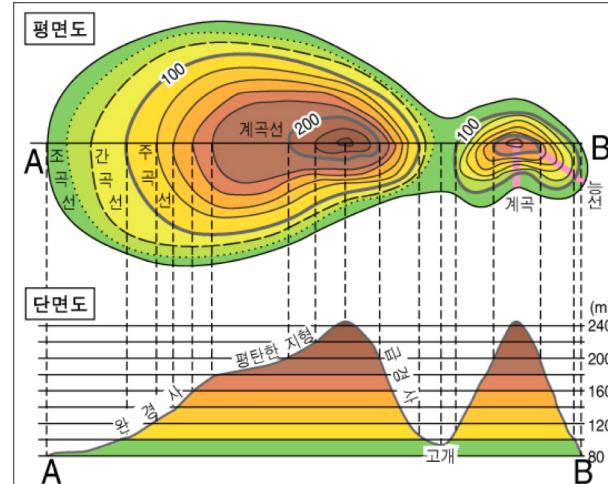
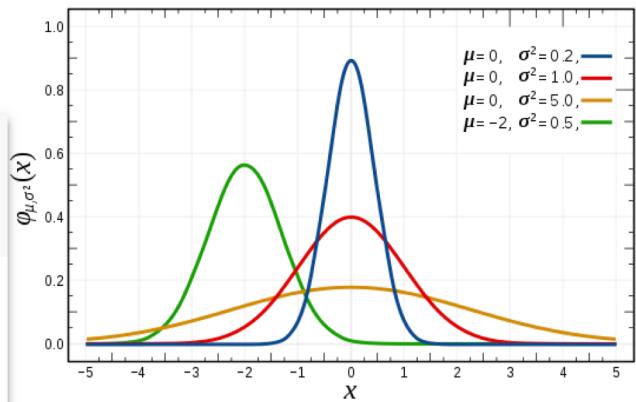
▼ Gaussian Naive Bayes

Perhaps the easiest naive Bayes classifier to understand is Gaussian naive Bayes. In this classifier, the assumption is that *data from each label is drawn from a simple Gaussian distribution*. Imagine that you have the following data:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(100, 2, centers=2, random_state=2, cluster_std=1.5).
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu');
```



✓ 참고로 가우시안 분포는 정규분포라고도 한다.



Gaussian Naive Bayes

~/code/05.05-Naive-Bayes.ipynb

가우시안 나이브 베이지안 분석기법은, 레이블링 된 Feature들이 가우시안 분포를 따른다는 가정하에 새로운 데이터를 분류한다.

- ✓ 하지만 모든 Feature가 정규분포를 따를 것이라는 가정은 너무 Naive 하다. 하지만, 많은 분야에서 Feature의 숫자가 충분히 크면, 설명력이 높은 경우가 나타난다.

Gaussian Naive Bayes Example

Figure Context

```
[ ] from sklearn.datasets import make_blobs
X, y = make_blobs(100, 2, centers=2, random_state=2, cluster_std=1.5)

fig, ax = plt.subplots()

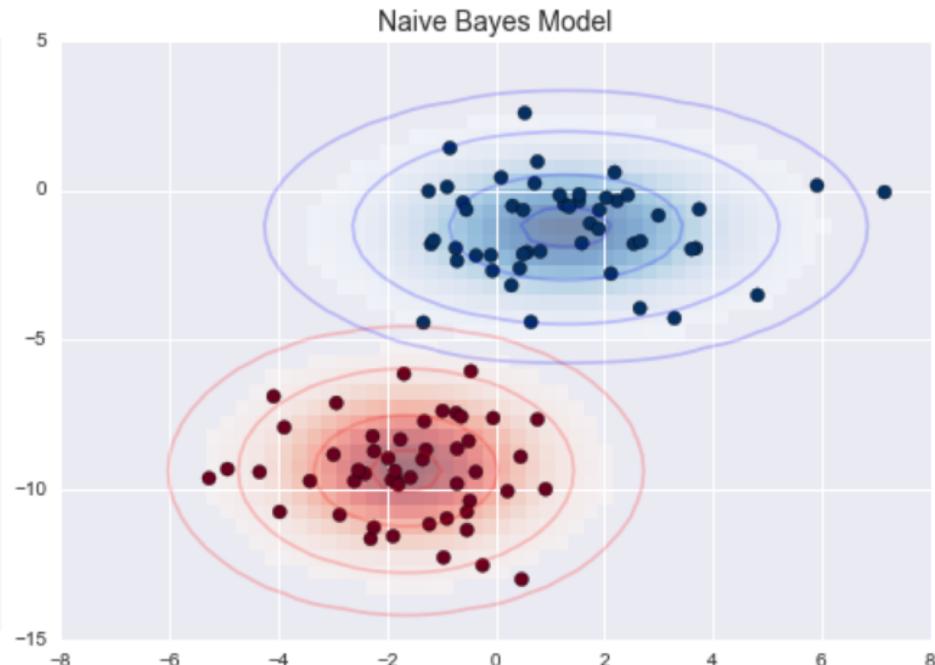
ax.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu')
ax.set_title('Naive Bayes Model', size=14)

xlim = (-8, 8)
ylim = (-15, 5)

xg = np.linspace(xlim[0], xlim[1], 60)
yg = np.linspace(ylim[0], ylim[1], 40)
xx, yy = np.meshgrid(xg, yg)
Xgrid = np.vstack([xx.ravel(), yy.ravel()]).T

for label, color in enumerate(['red', 'blue']):
    mask = (y == label)
    mu, std = X[mask].mean(0), X[mask].std(0)
    P = np.exp(-0.5 * (Xgrid - mu) ** 2 / std ** 2).prod(1)
    Pm = np.ma.masked_array(P, P < 0.03)
    ax.pcolorfast(xx, yg, Pm.reshape(xx.shape), alpha=0.5,
                  cmap=color.title() + 's')
    ax.contour(xx, yy, P.reshape(xx.shape),
               levels=[0.01, 0.1, 0.5, 0.9],
               colors=color, alpha=0.2)

ax.set(xlim=xlim, ylim=ylim)
fig.savefig('figures/05.05-gaussian-NB.png')
```

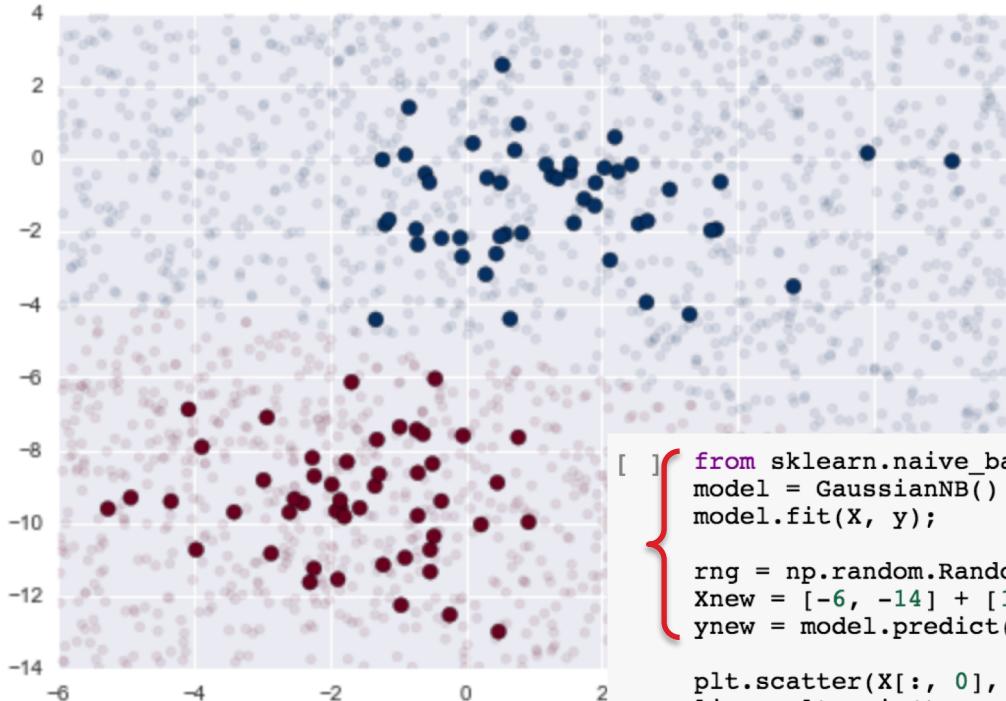


Gaussian Naive Bayes

~/code/05.05-Naive-Bayes.ipynb

가우시안 나이브 베이지안 분석기법은, 레이블링 된 Feature들이 가우시안 분포를 따른다는 가정하에 새로운 데이터를 분류한다.

- ✓ 각 Feature의 정규분포도에 따라 모르는 데이터에 대해 레이블링을 실시할 수 있음.



```
[ ] { from sklearn.naive_bayes import GaussianNB
    model = GaussianNB()
    model.fit(X, y);

    rng = np.random.RandomState(0)
    Xnew = [-6, -14] + [14, 18] * rng.rand(2000, 2)
    ynew = model.predict(Xnew)

    plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu')
    lim = plt.axis()
    plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, cmap='RdBu', alpha=0.1)
    plt.axis(lim); }
```

Multinomial Naive Bayes

~/code/05.05-Naive-Bayes.ipynb

여러개의 변수가 있을 때의 나이브 베이지안 분석

- ✓ 텍스트 분석에 많이 사용된다.
- ✓ 정말 각 분야에서 희소하게 등장하는 단어를 Count해서 해당 글을 분류하는 모델

```
from sklearn.datasets import fetch_20newsgroups

data = fetch_20newsgroups()
data.target_names

categories = ['talk.religion.misc', 'soc.religion.christian',
              'sci.space', 'comp.graphics']
train = fetch_20newsgroups(subset='train', categories=categories)
test = fetch_20newsgroups(subset='test', categories=categories)

print(train.data[5])

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(train.data, train.target)
labels = model.predict(test.data)

from sklearn.metrics import confusion_matrix
mat = confusion_matrix(test.target, labels)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=train.target_names, yticklabels=train.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label');
```

```
[ ] print(train.data[5])
```

From: dmcgee@uluhe.soest.hawaii.edu (Don McGee)
 Subject: Federal Hearing
 Originator: dmcgee@uluhe
 Organization: School of Ocean and Earth Science and Technology
 Distribution: usa
 Lines: 10

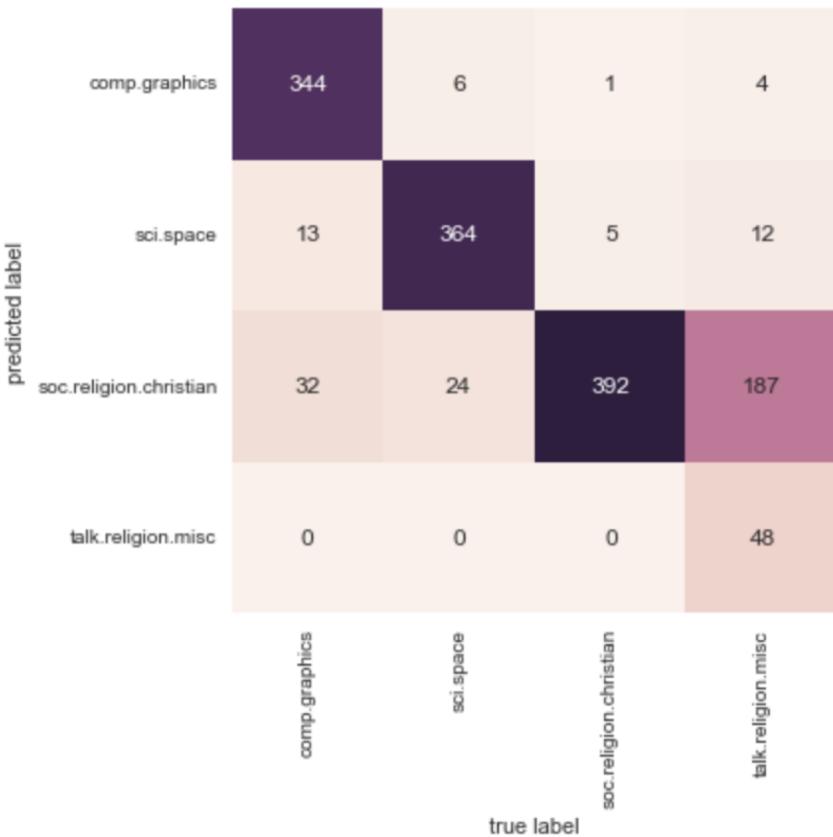
Fact or rumor....? Madalyn Murray O'Hare an atheist who eliminated the use of the bible reading and prayer in public schools 15 years ago is now going to appear before the FCC with a petition to stop the reading of the Gospel on the airways of America. And she is also campaigning to remove Christmas programs, songs, etc from the public schools. If it is true then mail to Federal Communications Commission 1919 H Street Washington DC 20054 expressing your opposition to her request. Reference Petition number 2493.

Multinomial Naive Bayes

~/code/05.05-Naive-Bayes.ipynb

여러개의 변수가 있을 때의 나이브 베이지안 분석

- ✓ 텍스트 분석에 많이 사용된다.
- ✓ 정말 각 분야에서 희소하게 등장하는 단어를 Count해서 해당 글을 분류하는 모델



```
[ ] def predict_category(s, train=train, model=model):
    pred = model.predict([s])
    return train.target_names[pred[0]]
```

Let's try it out:

```
[ ] predict_category('sending a payload to the ISS')
```

'sci.space'

```
[ ] predict_category('discussing islam vs atheism')
```

'soc.religion.christian'

```
[ ] predict_category('determining the screen resolution')
```

'comp.graphics'

SVM(Support Vector Machine)

~/code/05.07-Support-Vector-Machines.ipynb

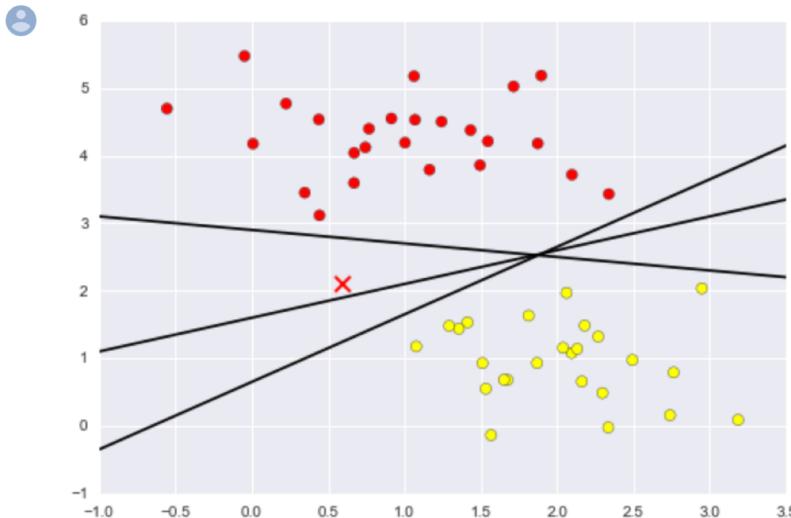
서포트 벡터 머신은 분류와 회귀 모형 모두에 사용할 수 있는 유연한 지도학습 알고리즘

- ✓ 어떠한 선을 그었을 때, 같은 거리로 양쪽에 선을 그었을 때 넓이가 최대로 되게끔 선을 긋는 것.
- ✓ 아래의 왼쪽 그래프에서 x 지점에 있는 데이터를 어떻게 구분할 것인가가 문제가 된다. 이럴 때, 마진을 최대로 하는 선을 분류 모델로 선택하는 것이 SVM이다.
- ✓ 머신러닝 분야에서 성능이 좋은편으로 알려져 있다.

```
[ ] xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.plot([0.6], [2.1], 'x', color='red', markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m * xfit + b, '-k')

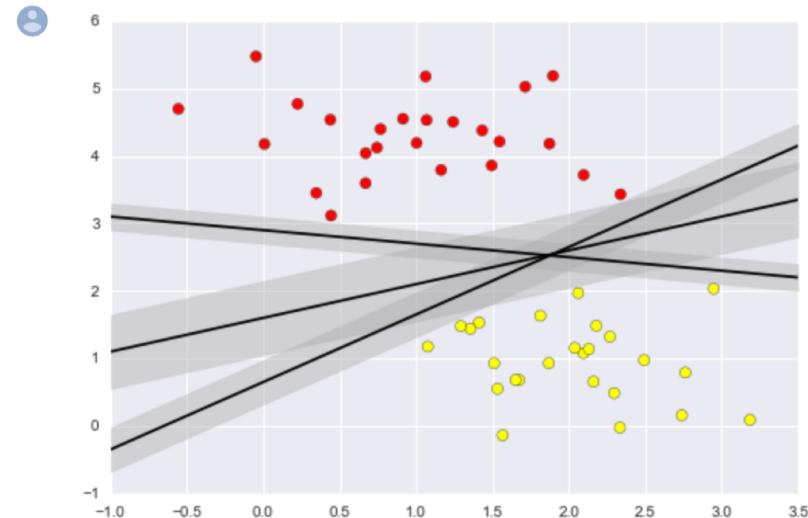
plt.xlim(-1, 3.5);
```



```
[ ] xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
                     color='AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);
```



SVM(Support Vector Machine)

~/code/05.07-Support-Vector-Machines.ipynb

서포트 벡터 머신은 분류와 회귀 모형 모두에 사용할 수 있는 유연한 지도학습 알고리즘

- ✓ 어떠한 선을 그었을 때, 같은 거리로 양쪽에 선을 그었을 때 넓이가 최대로 되게끔 선을 긋는 것.

```
{
from sklearn.svm import SVC # "Support vector classifier"
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)

def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

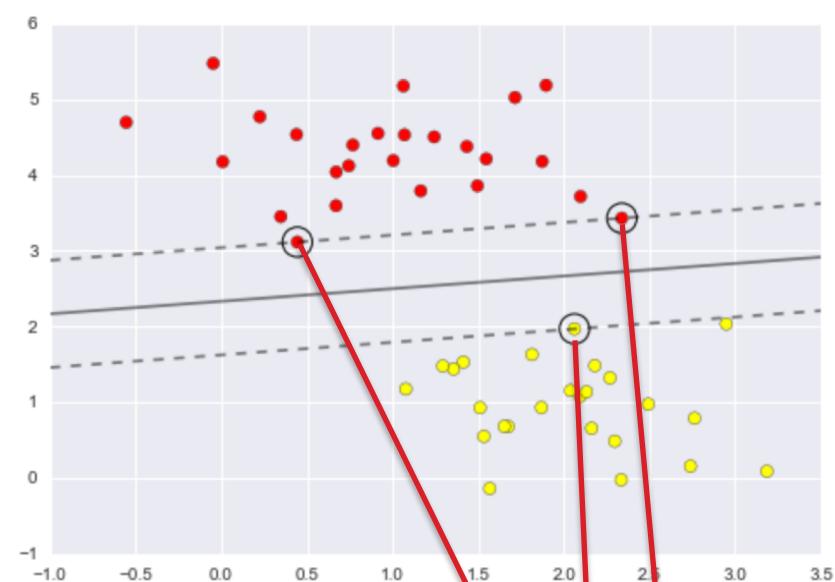
    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    X, Y = np.meshgrid(x, y)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, facecolors='none');

ax.set_xlim(xlim)
ax.set_ylim(ylim)}
```

```
[ ] plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
```

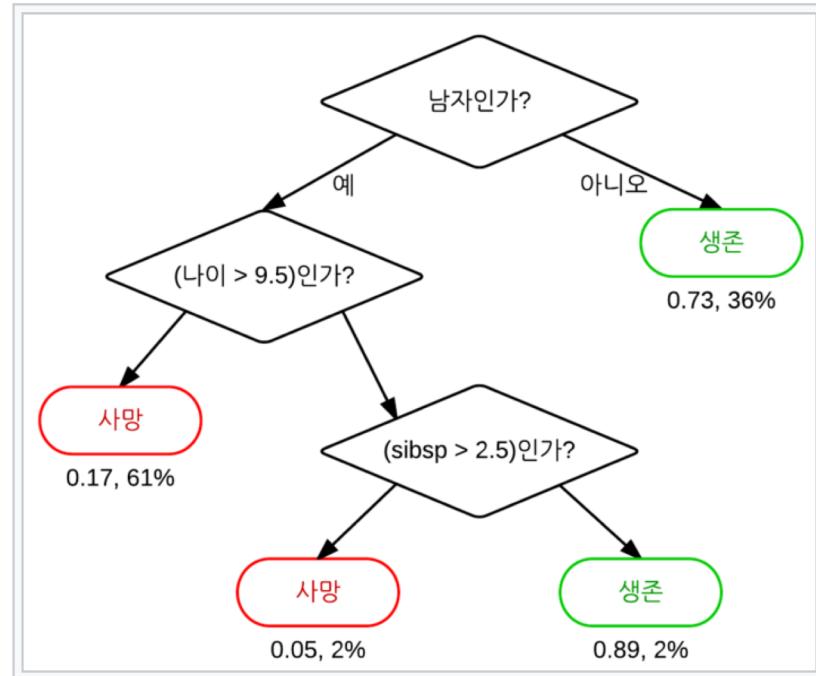
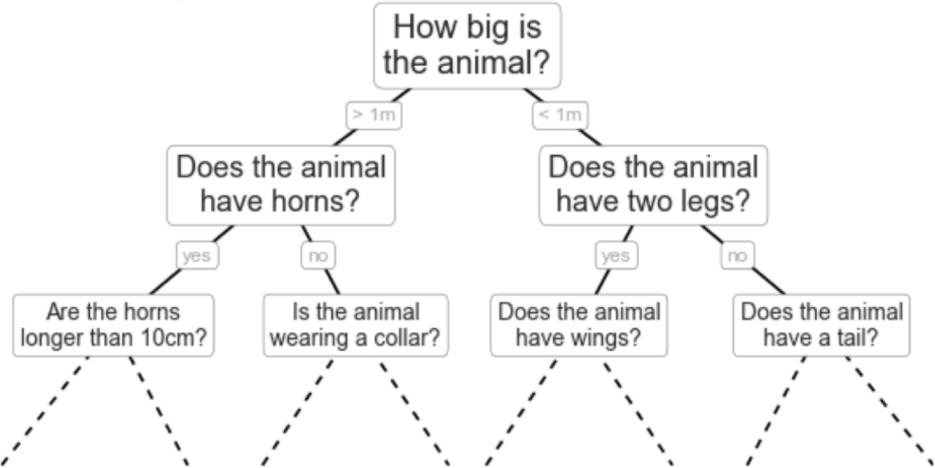


```
[ ] model.support_vectors_
array([[ 0.44359863,  3.11530945],
       [ 2.33812285,  3.43116792],
       [ 2.06156753,  1.96918596]])
```

Decision Tree

의사결정나무는 학습에 사용되는 자료 집합을 적절한 분할 기준 또는 분할 테스트에 따라 부분 집합들로 나누는 과정이다.

Example Decision Tree: Animal Classification



타이타닉호 탑승객의 생존 여부를 나타내는 결정 트리. ("sibsp"는 탑승한 배우자와 자녀의 수를 의미한다.) 앞 아래의 숫자는 각각 생존 확률과 탑승객이 그 앞에 해당될 확률을 의미한다.

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

Decision Tree

~/code/06_00_Figure_Code.ipynb

시각화 코드

▼ Decision Tree Example

```
[ ]  fig = plt.figure(figsize=(10, 4))
ax = fig.add_axes([0, 0, 0.8, 1], frameon=False, xticks=[], yticks[])
ax.set_title('Example Decision Tree: Animal Classification', size=24)

def text(ax, x, y, t, size=20, **kwargs):
    ax.text(x, y, t,
            ha='center', va='center', size=size,
            bbox=dict(boxstyle='round', ec='k', fc='w'), **kwargs)

text(ax, 0.5, 0.9, "How big is\nthe animal?", 20)
text(ax, 0.3, 0.6, "Does the animal\nhave horns?", 18)
text(ax, 0.7, 0.6, "Does the animal\nhave two legs?", 18)
text(ax, 0.12, 0.3, "Are the horns\nlonger than 10cm?", 14)
text(ax, 0.38, 0.3, "Is the animal\nwearing a collar?", 14)
text(ax, 0.62, 0.3, "Does the animal\nhave wings?", 14)
text(ax, 0.88, 0.3, "Does the animal\nhave a tail?", 14)

text(ax, 0.4, 0.75, "> 1m", 12, alpha=0.4)
text(ax, 0.6, 0.75, "< 1m", 12, alpha=0.4)

text(ax, 0.21, 0.45, "yes", 12, alpha=0.4)
text(ax, 0.34, 0.45, "no", 12, alpha=0.4)

text(ax, 0.66, 0.45, "yes", 12, alpha=0.4)
text(ax, 0.79, 0.45, "no", 12, alpha=0.4)

ax.plot([0.3, 0.5, 0.7], [0.6, 0.9, 0.6], '-k')
ax.plot([0.12, 0.3, 0.38], [0.3, 0.6, 0.3], '-k')
ax.plot([0.62, 0.7, 0.88], [0.3, 0.6, 0.3], '-k')
ax.plot([0.0, 0.12, 0.20], [0.0, 0.3, 0.0], '--k')
ax.plot([0.28, 0.38, 0.48], [0.0, 0.3, 0.0], '--k')
ax.plot([0.52, 0.62, 0.72], [0.0, 0.3, 0.0], '--k')
ax.plot([0.8, 0.88, 1.0], [0.0, 0.3, 0.0], '--k')
ax.axis([0, 1, 0, 1])

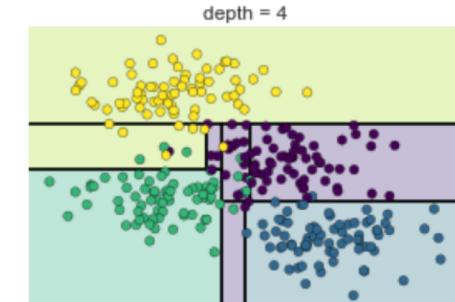
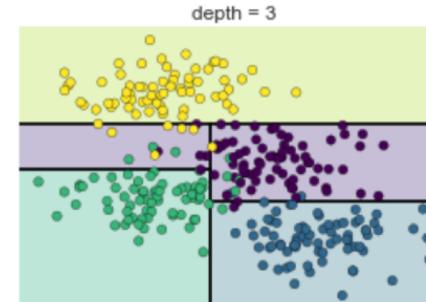
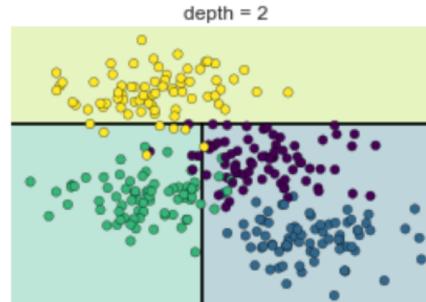
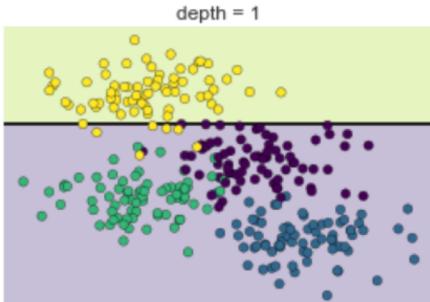
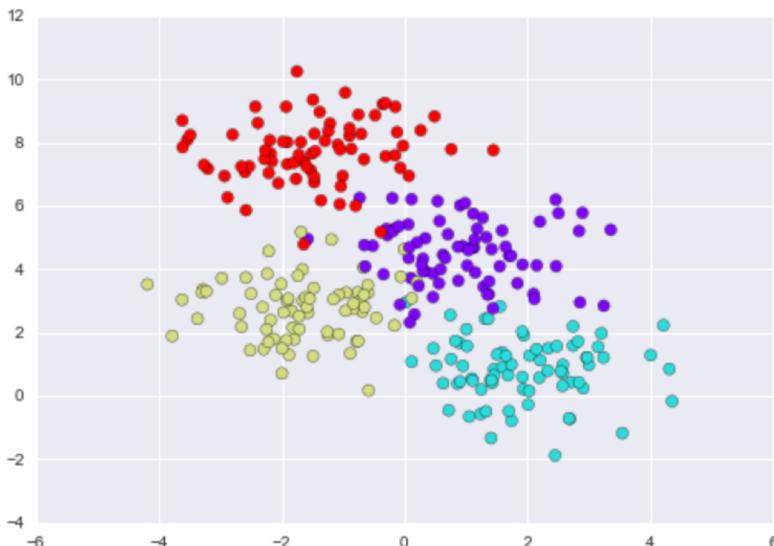
fig.savefig('figures/05.08-decision-tree.png')
```

Decision Tree Levels

~/code/05.08-Random-Forests.ipynb

각 트리를 분기할 때, 깊이 depth에 따라 분류 기준을 정하여 주어진 데이터의 수준Level을 구분한다.

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=300, centers=4,
                   random_state=0, cluster_std=1.0)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='rainbow');
```



Decision Tree Levels

```
[ ]  from helpers_05_08 import visualize_tree
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.datasets import make_blobs

      fig, ax = plt.subplots(1, 4, figsize=(16, 3))
      fig.subplots_adjust(left=0.02, right=0.98, wspace=0.1)

      X, y = make_blobs(n_samples=300, centers=4,
                         random_state=0, cluster_std=1.0)

      {
          for axi, depth in zip(ax, range(1, 5)):
              model = DecisionTreeClassifier(max_depth=depth)
              visualize_tree(model, X, y, ax=axi)
              axi.set_title('depth = {}'.format(depth))

      fig.savefig('figures/05.08-decision-tree-levels.png')
```

Decision Tree Levels

~/code/05.08-Random-Forests.ipynb

의사결정나무 분류기 시각화 Decision Tree Classifier Visualization

```
def visualize_classifier(model, X, y, ax=None, cmap='rainbow'):
    ax = ax or plt.gca()

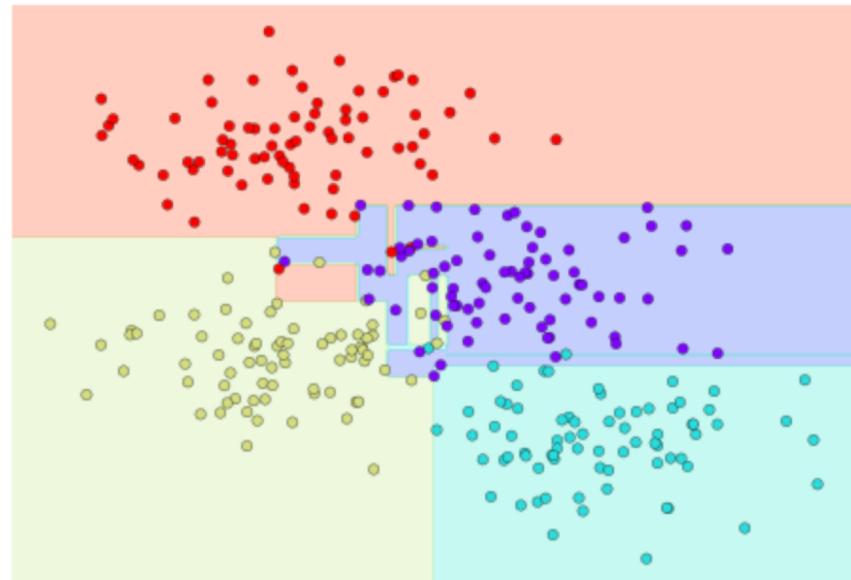
    # Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=cmap,
               clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # fit the estimator
    model.fit(X, y)
    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
                         np.linspace(*ylim, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    # Create a color plot with the results
    n_classes = len(np.unique(y))
    contours = ax.contourf(xx, yy, Z, alpha=0.3,
                           levels=np.arange(n_classes + 1) - 0.5,
                           cmap=cmap, clim=(y.min(), y.max()),
                           zorder=1)

    ax.set(xlim=xlim, ylim=ylim)
```

{ visualize_classifier(DecisionTreeClassifier(), X, y) }



Decision Tree & Overfitting

~/code/05.08-Random-Forests.ipynb

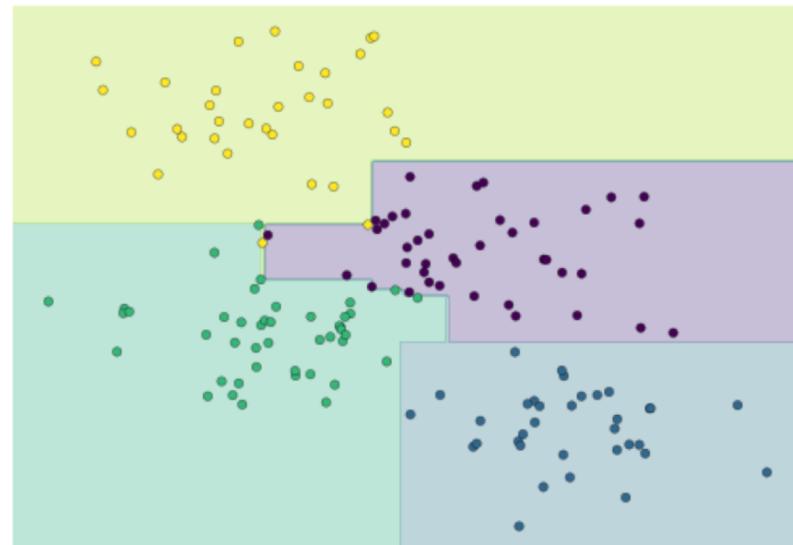
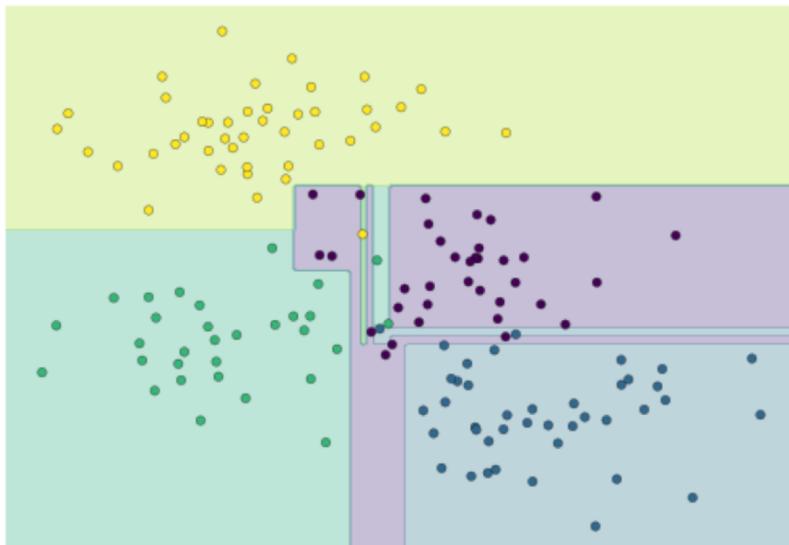
의사결정나무의 깊이가 너무 깊으면, 아래의 그림과 같이 과적합이 나타날 수 있다.

- ✓ 적절한 깊이를 지정한 뒤, 추가 분석작업을 진행할 수 있다.
- ✓ 내가 만든 의사결정나무의 깊이가 너무 깊은지는, 데이터의 부분집합을 통해 분류를 진행해보면 알 수 있다.

```
model = DecisionTreeClassifier()
```

```
fig, ax = plt.subplots(1, 2, figsize=(16, 6))
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)
visualize_tree(model, X[::2], y[::2], boundaries=False, ax=ax[0])
visualize_tree(model, X[1::2], y[1::2], boundaries=False, ax=ax[1])

fig.savefig('figures/05.08-decision-tree-overfitting.png')
```



Ensembles of Estimators: Random Forests

~/code/05.08-Random-Forests.ipynb

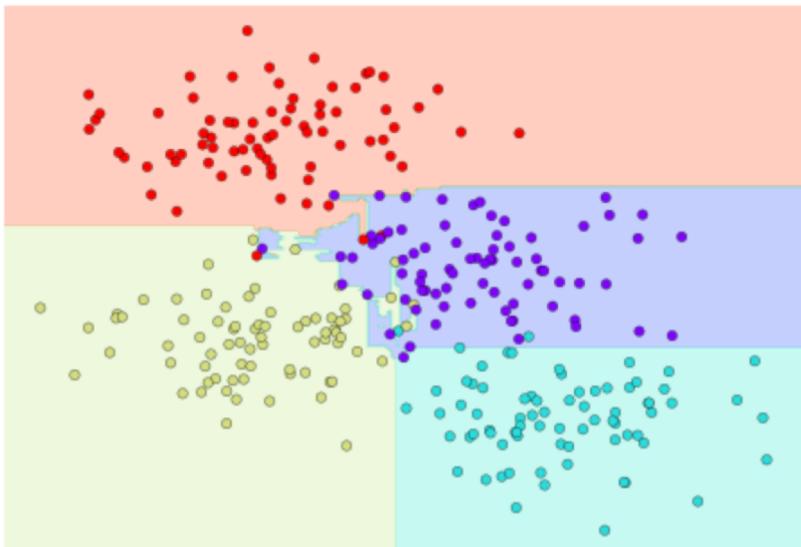
랜덤포레스트는 의사결정 나무를 기반으로 한 양상을 학습 방법의 한 예다.

- ✓ 의사결정나무의 깊이를 부분집합들을 통해 과적합 여부를 관찰할 수 있다면, 각 부분집합들의 평균을 내면 과적합이 최소화 되지 않을까? 이러한 개념을 배깅(Bagging)이라고 한다.
- ✓ 이러한 배깅을 활용한 기법이 랜덤포레스트이다.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

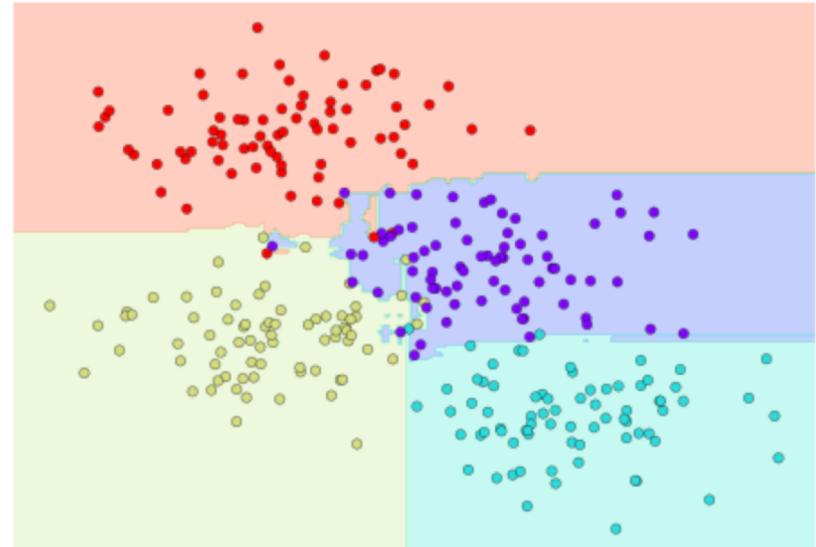
tree = DecisionTreeClassifier()
bag = BaggingClassifier(tree, n_estimators=100, max_samples=0.8,
                       random_state=1)

bag.fit(X, y)
visualize_classifier(bag, X, y)
```



```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_state=0)
visualize_classifier(model, X, y);
```



심화 예제: Face Recognition

SVM(Support Vector Machine)

~/code/05.07-Support-Vector-Machines.ipynb

얼굴 인식Face Recognition 예제



Colin Powell



George W Bush



George W Bush



George W Bush



Hugo Chavez



George W Bush



Junichiro Koizumi



George W Bush



Tony Blair



Ariel Sharon



George W Bush



Donald Rumsfeld



George W Bush



George W Bush



George W Bush

```
from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=60)
print(faces.target_names)
print(faces.images.shape)

fig, ax = plt.subplots(3, 5)
for i, ax_i in enumerate(ax.flat):
    ax_i.imshow(faces.images[i], cmap='bone')
    ax_i.set(xticks=[], yticks=[],
             xlabel=faces.target_names[faces.target[i]])
```

SVM(Support Vector Machine)

~/code/05.07-Support-Vector-Machines.ipynb

얼굴 인식Face Recognition 예제

- ✓ 선을 직선으로 그리지 않고, 여러 형태의 데이터 분포에 그릴 수 있는데, 이를 커널Kernel 이라 한다.
- ✓ 커널Kernel에 rbf를 적용해서 꼭 선형으로 구분하는 것이 아닌, 다른 데이터 분포에도 적용이 되도록 설정한 것을 확인할 수 있다.

```

from sklearn.svm import SVC
from sklearn.decomposition import RandomizedPCA
from sklearn.pipeline import make_pipeline
{
# Modeling
pca = RandomizedPCA(n_components=150, whiten=True, random_state=42)
svc = SVC(kernel='rbf', class_weight='balanced')
model = make_pipeline(pca, svc)

# Data sampling for cross validation
from sklearn.cross_validation import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(faces.data, faces.target,
                                                random_state=42)

# Searching optiomal values
from sklearn.grid_search import GridSearchCV
param_grid = {'svc__C': [1, 5, 10, 50],
              'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(model, param_grid)

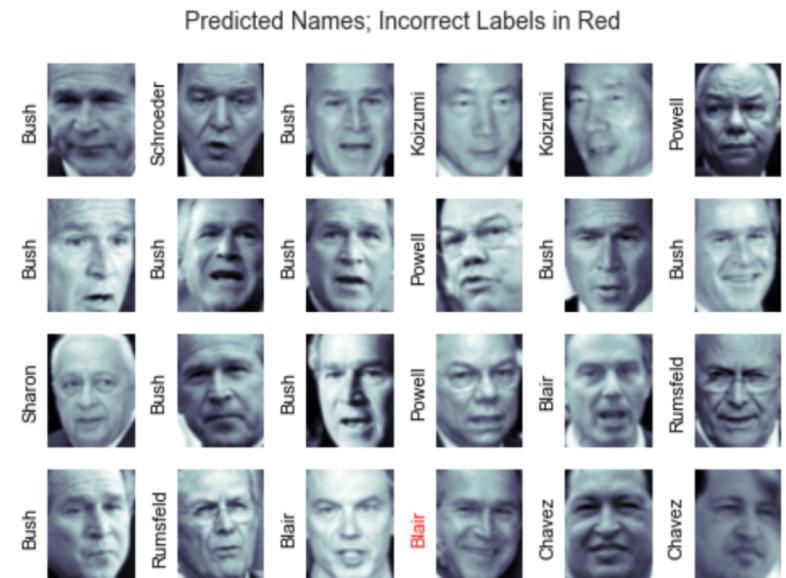
#time grid.fit(Xtrain, ytrain)
#print(grid.best_params_)

model = grid.best_estimator_
yfit = model.predict(Xtest)

CPU times: user 47.8 s, sys: 4.08 s, total: 51.8 s
Wall time: 26 s
{'svc__gamma': 0.001, 'svc__C': 10}
  
```

```

fig, ax = plt.subplots(4, 6)
for i, axi in enumerate(ax.flat):
    axi.imshow(Xtest[i].reshape(62, 47), cmap='bone')
    axi.set_xticks([]), axi.set_yticks([])
    axi.set_ylabel(faces.target_names[yfit[i]].split()[-1],
                  color='black' if yfit[i] == ytest[i] else 'red')
fig.suptitle('Predicted Names; Incorrect Labels in Red', size=14);
  
```



SVM(Support Vector Machine)

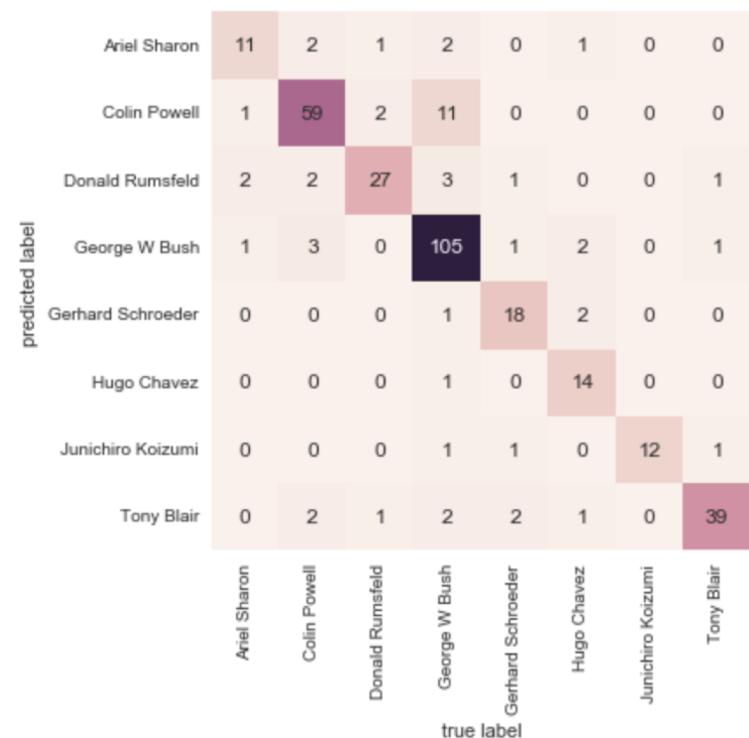
~/code/05.07-Support-Vector-Machines.ipynb

얼굴 인식Face Recognition 예제, 성능 평가

- ✓ 각 예측 대상별 f1-score를 확인할 수 있고, 해당 예측한 상황을 시각화 하여 살펴볼 수 있다.

	precision	recall	f1-score	support
Ariel Sharon	0.65	0.73	0.69	15
Colin Powell	0.81	0.87	0.84	68
Donald Rumsfeld	0.75	0.87	0.81	31
George W Bush	0.93	0.83	0.88	126
Gerhard Schroeder	0.86	0.78	0.82	23
Hugo Chavez	0.93	0.70	0.80	20
Junichiro Koizumi	0.80	1.00	0.89	12
Tony Blair	0.83	0.93	0.88	42
avg / total	0.85	0.85	0.85	337

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(ytest, yfit)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=faces.target_names,
            yticklabels=faces.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label');
```



감사합니다
