16장. 로지스틱 회귀 분석

1. 선형 회귀와 분류 문제

1.1 데이터셋 정의

In [2]:
```
tuples = [(0.7,48000,1),(1.9,48000,0),(2.5,60000,1),(4.2,63000,0),(6,76000,0),(6.5,69000,0),(7.5,76000,0),(8.1,88000,0),(8.7,83
```

1.2 입력과 레이블로 분리

In [3]:
```
data = [list(row) for row in tuples]

xs = [[1.0] + row[:2] for row in data]  # [1, experience, salary]
ys = [row[2] for row in data]           # paid_account
```

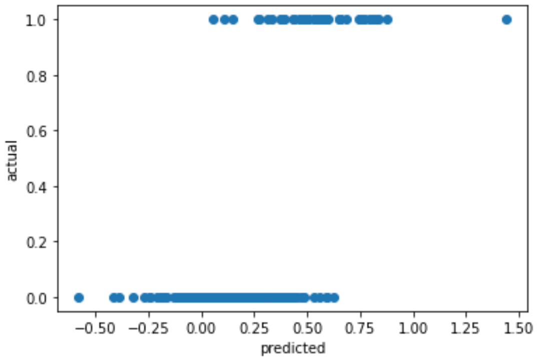
1.3 선형 회귀 분석

In [4]:
```
from matplotlib import pyplot as plt
from scratch.working_with_data import rescale
from scratch.multiple_regression import least_squares_fit, predict
from scratch.gradient_descent import gradient_step

learning_rate = 0.001
rescaled_xs = rescale(xs)
beta = least_squares_fit(rescaled_xs, ys, learning_rate, 1000, 1)
# [0.26, 0.43, -0.43]
predictions = [predict(x_i, beta) for x_i in rescaled_xs]
```

```
least squares fit: 100%|██████████████████████████████████████████████████| 1000/1000 [00:01<00:00, 544.00it/s]
```

1.4 실제값과 예측 결과의 산포도

In [5]:
```
plt.scatter(predictions, ys)
plt.xlabel("predicted")
plt.ylabel("actual")
plt.show()
```



2. 로지스틱 회귀

2.1 로지스틱 함수 (Logistic Function)

로지스틱 함수

In [6]:
```
from matplotlib import pyplot as plt

def logistic(x: float) -> float:
    return 1.0 / (1 + math.exp(-x))
```

로지스틱 함수의 미분

In [7]:

```
def logistic_prime(x: float) -> float:
    y = logistic(x)
    return y * (1 - y)
```

2.2 손실 함수

음의 로그 우도 (NLL : Negative Log Likelihood)

In [8]:

```
import math
from scratch.linear_algebra import Vector, dot

def _negative_log_likelihood(x: Vector, y: float, beta: Vector) -> float:
    """The negative log likelihood for one data point"""
    if y == 1:
        return -math.log(logistic(dot(x, beta)))
    else:
        return -math.log(1 - logistic(dot(x, beta)))
```

전체 데이터셋에 대해 NLL 합산

In [9]:

```
from typing import List

def negative_log_likelihood(xs: List[Vector],
                            ys: List[float],
                            beta: Vector) -> float:
    return sum(_negative_log_likelihood(x, y, beta)
               for x, y in zip(xs, ys))
```

2.3 그래디언트 (Gradient)

β_j에 대한 NLL 편미분

In [10]:

```
from scratch.linear_algebra import vector_sum

def _negative_log_partial_j(x: Vector, y: float, beta: Vector, j: int) -> float:
    """
    The j-th partial derivative for one data pont
    here i is the index of the data point
    """
    return -(y - logistic(dot(x, beta))) * x[j]
```

β에 대한 그래디언트

In [11]:

```
def _negative_log_gradient(x: Vector, y: float, beta: Vector) -> Vector:
    """
    The gradient for one data point
    """
    return [_negative_log_partial_j(x, y, beta, j)
            for j in range(len(beta))]
```

전체 데이터 셋에 대해 그래디언트 합산

In [12]:

```
def negative_log_gradient(xs: List[Vector],
                          ys: List[float],
                          beta: Vector) -> Vector:
    return vector_sum([_negative_log_gradient(x, y, beta)
                       for x, y in zip(xs, ys)])
```

2.4 데이터 분리

In [13]:

```
from scratch.machine_learning import train_test_split
import random
import tqdm

random.seed(0)
x_train, x_test, y_train, y_test = train_test_split(rescaled_xs, ys, 0.33)
```

2.5 경사하강법 적용

In [14]:

```
# pick a random starting point
beta = [random.random() for _ in range(3)]

with tqdm.trange(5000) as t:
    for epoch in t:
        gradient = negative_log_gradient(x_train, y_train, beta)
        beta = gradient_step(beta, gradient, -learning_rate)
        loss = negative_log_likelihood(x_train, y_train, beta)
        t.set_description(f"loss: {loss:.3f} beta: {beta}")
```

```
loss: 39.967 beta: [-2.000793672929925, 4.621634674821908, -4.400062646173013]: 100%|█| 5000/5000 [00:13<00:00, 363.88i
```

2.6 입력 데이터를 원래 스케일로 복구

In [15]:

```
from scratch.working_with_data import scale

means, stdevs = scale(xs)
beta_unscaled = [(beta[0]
                  - beta[1] * means[1] / stdevs[1]
                  - beta[2] * means[2] / stdevs[2]),
                 beta[1] / stdevs[1],
                 beta[2] / stdevs[2]]
# [8.9, 1.6, -0.000288]
assert (negative_log_likelihood(xs, ys, beta_unscaled) ==
        negative_log_likelihood(rescaled_xs, ys, beta))
print(beta)
print(beta_unscaled)
```

```
[-2.000793672929925, 4.621634674821908, -4.400062646173013]
[8.776267817669087, 1.6231222550232345, -0.00028319979791175]
```

2.7 정밀도와 재현율 계산

In [16]:

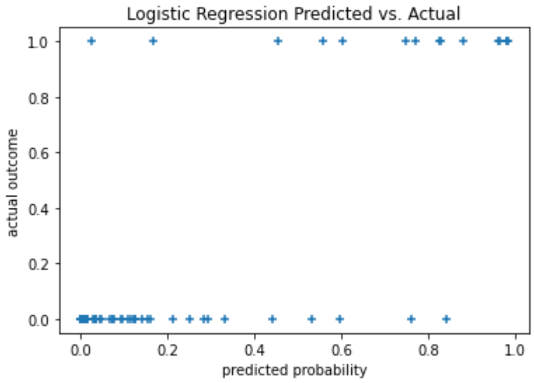
```
true_positives = false_positives = true_negatives = false_negatives = 0

for x_i, y_i in zip(x_test, y_test):
    prediction = logistic(dot(beta, x_i))

    if y_i == 1 and prediction >= 0.5:  # TP: paid and we predict paid
        true_positives += 1
    elif y_i == 1:                      # FN: paid and we predict unpaid
        false_negatives += 1
    elif prediction >= 0.5:             # FP: unpaid and we predict paid
        false_positives += 1
    else:                               # TN: unpaid and we predict unpaid
        true_negatives += 1
```

In [17]:

```
precision = true_positives / (true_positives + false_positives)
recall = true_positives / (true_positives + false_negatives)

print(precision, recall)

assert precision == 0.75
assert recall == 0.8
```

```
0.75 0.8
```

In [18]:
```
predictions = [logistic(dot(beta, x_i)) for x_i in x_test]
plt.scatter(predictions, y_test, marker='+')
plt.xlabel("predicted probability")
plt.ylabel("actual outcome")
plt.title("Logistic Regression Predicted vs. Actual")
plt.show()
```



20장. 군집화 (Clustering)

1. K-평균 군집화 (K-means clustering)

In [1]:

```
from scratch.linear_algebra import Vector
```

1.1 해밍 거리 (hamming distance)

두 벡터의 다른 값을 갖는 요소 개수

In [2]:

```
def num_differences(v1: Vector, v2: Vector) -> int:
    assert len(v1) == len(v2)
    return len([x1 for x1, x2 in zip(v1, v2) if x1 != x2])

assert num_differences([1, 2, 3], [2, 1, 3]) == 2
assert num_differences([1, 2], [1, 2]) == 0
```

1.2 클러스터 평균

In [3]:

```
from typing import List
from scratch.linear_algebra import vector_mean

def cluster_means(k: int,
                  inputs: List[Vector],
                  assignments: List[int]) -> List[Vector]:
    # clusters[i] contains the inputs whose assignment is i
    clusters = [[] for i in range(k)]
    for input, assignment in zip(inputs, assignments):
        clusters[assignment].append(input)

    # if a cluster is empty, just use a random point
    return [vector_mean(cluster) if cluster else random.choice(inputs)
            for cluster in clusters]
```

1.3 K-Means 알고리즘

In [4]:

```
import itertools
import random
import tqdm
from scratch.linear_algebra import squared_distance

class KMeans:
    def __init__(self, k: int) -> None:
        self.k = k                      # number of clusters
        self.means = None

    def classify(self, input: Vector) -> int:
        """return the index of the cluster closest to the input"""
        return min(range(self.k),
                   key=lambda i: squared_distance(input, self.means[i]))

    def train(self, inputs: List[Vector]) -> None:
        # Start with random assignments
        assignments = [random.randrange(self.k) for _ in inputs]

        with tqdm.tqdm(itertools.count()) as t:
            for _ in t:
                # Compute means and find new assignments
                self.means = cluster_means(self.k, inputs, assignments)
                new_assignments = [self.classify(input) for input in inputs]

                # Check how many assignments changed and if we're done
                num_changed = num_differences(assignments, new_assignments)
                if num_changed == 0:
                    return

                # Otherwise keep the new assignments, and compute new means
                assignments = new_assignments
                t.set_description(f"changed: {num_changed} / {len(inputs)}")
```

1.4 예시 : 회원 모임 장소 정하기

In [5]:

```
inputs: List[List[float]] = [[-14,-5],[13,13],[20,23],[-19,-11],[-9,-16],[21,27],[-49,15],[26,13],[-46,5],[-34,-1],[11,15],[-
```

1.4.1 세번 모임 개최를 위한 3-클러스터

In [6]:

```
random.seed(12)                   # so you get the same results as me
clusterer = KMeans(k=3)
clusterer.train(inputs)
means = sorted(clusterer.means)   # sort for the unit test

assert len(means) == 3

# Check that the means are close to what we expect.
assert squared_distance(means[0], [-44, 5]) < 1
assert squared_distance(means[1], [-16, -10]) < 1
assert squared_distance(means[2], [18, 20]) < 1
```

```
changed: 5 / 20: : 1it [00:00, 1001.98it/s]
```

1.4.2 두번 모임 개최를 위한 2-클러스터

In [7]:

```
random.seed(0)
clusterer = KMeans(k=2)
clusterer.train(inputs)
means = sorted(clusterer.means)

assert len(means) == 2
assert squared_distance(means[0], [-26, -5]) < 1
assert squared_distance(means[1], [18, 20]) < 1
```

```
changed: 4 / 20: : 2it [00:00, 669.05it/s]
```

1.5 손실 곡선을 보고 K 선택하기

1.5.1 손실 계산

In [8]:

```
from matplotlib import pyplot as plt

def squared_clustering_errors(inputs: List[Vector], k: int) -> float:
    """finds the total squared error from k-means clustering the inputs"""
    clusterer = KMeans(k)
    clusterer.train(inputs)
    means = clusterer.means
    assignments = [clusterer.classify(input) for input in inputs]

    return sum(squared_distance(input, means[cluster])
               for input, cluster in zip(inputs, assignments))
```

1.5.2 손실 곡선 그려보기

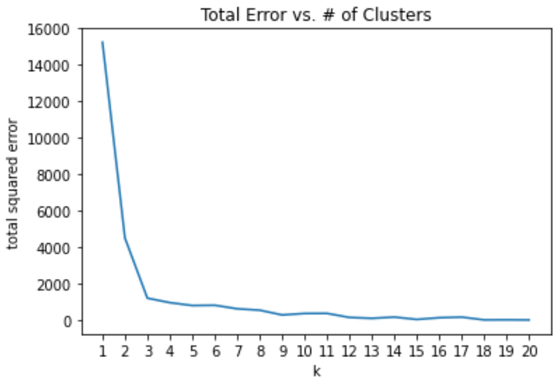
In [9]:

```
# now plot from 1 up to len(inputs) clusters
ks = range(1, len(inputs) + 1)
errors = [squared_clustering_errors(inputs, k) for k in ks]

plt.plot(ks, errors)
plt.xticks(ks)
plt.xlabel("k")
plt.ylabel("total squared error")
plt.title("Total Error vs. # of Clusters")
# plt.show()
```

```
0it [00:00, ?it/s]
changed: 8 / 20: : 1it [00:00, 496.66it/s]
changed: 2 / 20: : 2it [00:00, 71.62it/s]
changed: 1 / 20: : 3it [00:00, 745.39it/s]
changed: 5 / 20: : 2it [00:00, 334.25it/s]
changed: 4 / 20: : 2it [00:00, 502.22it/s]
changed: 2 / 20: : 2it [00:00, 334.31it/s]
changed: 1 / 20: : 3it [00:00, 429.89it/s]
changed: 1 / 20: : 3it [00:00, 752.21it/s]
changed: 2 / 20: : 2it [00:00, 660.73it/s]
changed: 7 / 20: : 2it [00:00, 250.66it/s]
changed: 1 / 20: : 4it [00:00, 445.30it/s]
changed: 3 / 20: : 3it [00:00, 501.17it/s]
changed: 2 / 20: : 6it [00:00, 334.28it/s]
changed: 1 / 20: : 3it [00:00, 431.66it/s]
changed: 2 / 20: : 4it [00:00, 442.64it/s]
changed: 3 / 20: : 2it [00:00, 222.79it/s]
changed: 2 / 20: : 4it [00:00, 286.47it/s]
changed: 5 / 20: : 2it [00:00, 334.01it/s]
changed: 1 / 20: : 3it [00:00, 498.51it/s]
```

Out[9]:

```
Text(0.5, 1.0, 'Total Error vs. # of Clusters')
```



1.6 예시 : 색 군집화하기

1.6.1 이미지 읽기

In [10]:

```
image_path = r"test_image.jpg"     # wherever your image is
import matplotlib.image as mpimg
img = mpimg.imread(image_path) / 256  # rescale to between 0 and 1
```

1.6.2 색 군집화

In [11]:

```
# .tolist() converts a numpy array to a Python list
pixels = [pixel.tolist() for row in img for pixel in row]

clusterer = KMeans(5)
clusterer.train(pixels)   # this might take a while
```

```
changed: 2 / 50246: : 29it [00:16,  1.72it/s]
```

1.6.3. 이미지 압축

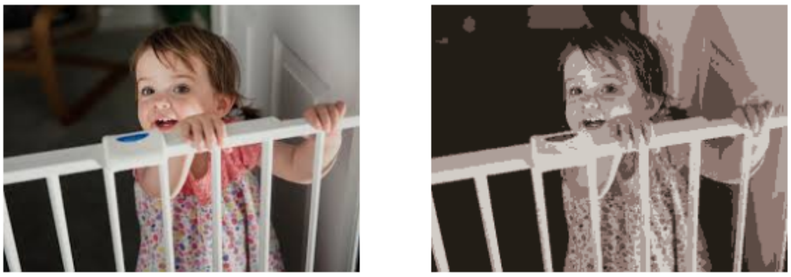
In [12]:

```
def recolor(pixel: Vector) -> Vector:
    cluster = clusterer.classify(pixel)        # index of the closest cluster
    return clusterer.means[cluster]            # mean of the closest cluster

new_img = [[recolor(pixel) for pixel in row]   # recolor this row of pixels
            for row in img]                    # for each row in the image
```

1.6.4. 결과 확인

In [13]:

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.imshow(img)
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(new_img)
plt.axis('off')
plt.show()
```



2. 계층 군집화 (Hierarchical Clustering)

2.1 자료 구조 정의

2.1.1 리프 노드, 병합 노드, 클러스터 정의

In [14]:

```
from typing import NamedTuple, Union

class Leaf(NamedTuple):
    value: Vector

class Merged(NamedTuple):
    children: tuple
    order: int

Cluster = Union[Leaf, Merged]
```

2.1.2 병합 노드 생성 예시

In [15]:

```
leaf1 = Leaf([10,  20])
leaf2 = Leaf([30, -15])

merged = Merged((leaf1, leaf2), order=1)
```

2.2 상향식 군집화 관련 함수

2.2.1 군집 데이터 목록

In [16]:

```
def get_values(cluster: Cluster) -> List[Vector]:
    if isinstance(cluster, Leaf):
        return [cluster.value]
    else:
        return [value
                for child in cluster.children
                for value in get_values(child)]

assert get_values(merged) == [[10, 20], [30, -15]]
```

2.2.2 군집 간 거리

In [17]:

```
from typing import Callable
from scratch.linear_algebra import distance

def cluster_distance(cluster1: Cluster,
                     cluster2: Cluster,
                     distance_agg: Callable = min) -> float:
    """
    compute all the pairwise distances between cluster1 and cluster2
    and apply the aggregation function _distance_agg_ to the resulting list
    """
    return distance_agg([distance(v1, v2)
                         for v1 in get_values(cluster1)
                         for v2 in get_values(cluster2)])
```

2.2.3 병합 순서

In [18]:

```
def get_merge_order(cluster: Cluster) -> float:
    if isinstance(cluster, Leaf):
        return float('inf')  # was never merged
    else:
        return cluster.order
```

2.2.4 자식 노드

In [19]:

```
from typing import Tuple

def get_children(cluster: Cluster):
    if isinstance(cluster, Leaf):
        raise TypeError("Leaf has no children")
    else:
        return cluster.children
```

2.3 상향식 계층 군집화

In [20]:

```
def bottom_up_cluster(inputs: List[Vector],
                      distance_agg: Callable = min) -> Cluster:
    # Start with all leaves
    clusters: List[Cluster] = [Leaf(input) for input in inputs]

    def pair_distance(pair: Tuple[Cluster, Cluster]) -> float:
        return cluster_distance(pair[0], pair[1], distance_agg)

    # as long as we have more than one cluster left...
    while len(clusters) > 1:
        # find the two closest clusters
        c1, c2 = min(((cluster1, cluster2)
                      for i, cluster1 in enumerate(clusters)
                      for cluster2 in clusters[:i]),
                      key=pair_distance)

        # remove them from the list of clusters
        clusters = [c for c in clusters if c != c1 and c != c2]

        # merge them, using merge_order = # of clusters left
        merged_cluster = Merged((c1, c2), order=len(clusters))

        # and add their merge
        clusters.append(merged_cluster)

    # when there's only one cluster left, return it
    return clusters[0]
```

2.4 군집 생성

In [21]:

```
def generate_clusters(base_cluster: Cluster,
                      num_clusters: int) -> List[Cluster]:
    # start with a list with just the base cluster
    clusters = [base_cluster]

    # as long as we don't have enough clusters yet...
    while len(clusters) < num_clusters:
        # choose the last-merged of our clusters
        next_cluster = min(clusters, key=get_merge_order)
        # remove it from the list
        clusters = [c for c in clusters if c != next_cluster]

        # and add its children to the list (i.e., unmerge it)
        clusters.extend(get_children(next_cluster))

    # once we have enough clusters...
    return clusters
```

2.5 예시 : 회원 모임 장소 정하기

In [22]:

```
from typing import Tuple
inputs: List[List[float]] = [[-14,-5],[13,13],[20,23],[-19,-11],[-9,-16],[21,27],[-49,15],[26,13],[-46,5],[-34,-1],[11,15],[-
```

2.5.1 세번 모임 개최를 위한 3-클러스터 (최소 거리 기준)

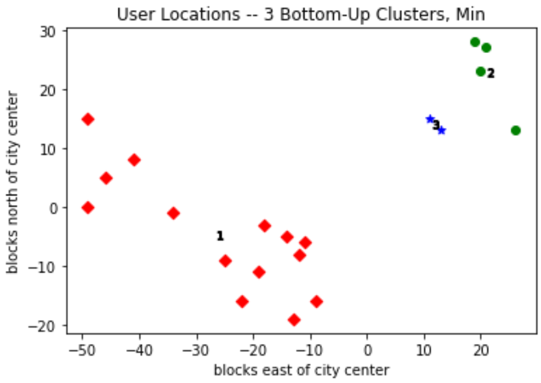
In [23]:

```
base_cluster = bottom_up_cluster(inputs)

three_clusters = [get_values(cluster)
                  for cluster in generate_clusters(base_cluster, 3)]


# sort smallest to largest
tc = sorted(three_clusters, key=len)
assert len(tc) == 3
assert [len(c) for c in tc] == [2, 4, 14]
assert sorted(tc[0]) == [[11, 15], [13, 13]]
```

군집화 결과 확인

In [24]:

```
for i, cluster, marker, color in zip([1, 2, 3],
                                      three_clusters,
                                      ['D','o','*'],
                                      ['r','g','b']):
    xs, ys = zip(*cluster)  # magic unzipping trick
    plt.scatter(xs, ys, color=color, marker=marker)

    # put a number at the mean of the cluster
    x, y = vector_mean(cluster)
    plt.plot(x, y, marker='$' + str(i) + '$', color='black')

plt.title("User Locations -- 3 Bottom-Up Clusters, Min")
plt.xlabel("blocks east of city center")
plt.ylabel("blocks north of city center")
plt.show()
```



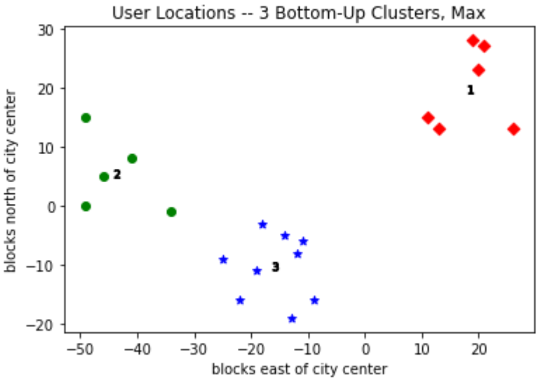
2.5.2 세번 모임 개최를 위한 3-클러스터 (최대 거리 기준)

In [25]:
```
base_cluster_max = bottom_up_cluster(inputs, max)
three_clusters_max = [get_values(cluster)
                      for cluster in generate_clusters(base_cluster_max, 3)]
```

군집화 결과 확인

In [26]:
```
for i, cluster, marker, color in zip([1, 2, 3],
                                     three_clusters_max,
                                     ['D','o','*'],
                                     ['r','g','b']):
    xs, ys = zip(*cluster)  # magic unzipping trick
    plt.scatter(xs, ys, color=color, marker=marker)

    # put a number at the mean of the cluster
    x, y = vector_mean(cluster)
    plt.plot(x, y, marker='$' + str(i) + '$', color='black')

plt.title("User Locations -- 3 Bottom-Up Clusters, Max")
plt.xlabel("blocks east of city center")
plt.ylabel("blocks north of city center")
```

Out[26]:
```
Text(0, 0.5, 'blocks north of city center')
```



10장. 차원 축소 (Dimension Reduction)

1. 차원 축소 (Dimension Reduction)

1.1 평균 빼기

In [1]:

```
from typing import List
from scratch.linear_algebra import Vector, subtract

def de_mean(data: List[Vector]) -> List[Vector]:
    """Recenters the data to have mean 0 in every dimension"""
    mean = vector_mean(data)
    return [subtract(vector, mean) for vector in data]
```

1.2 단위 벡터 만들기

In [2]:

```
from scratch.linear_algebra import magnitude

def direction(w: Vector) -> Vector:
    mag = magnitude(w)
    return [w_i / mag for w_i in w]
```

1.3 단위 벡터 방향으로 분산 구하기 (목적 함수)

In [3]:

```
from scratch.linear_algebra import dot

def directional_variance(data: List[Vector], w: Vector) -> float:
    """
    Returns the variance of x in the direction of w
    """
    w_dir = direction(w)
    return sum(dot(v, w_dir) ** 2 for v in data)
```

1.4 단위 벡터 방향의 분산에 대한 그래디언트

In [4]:

```
def directional_variance_gradient(data: List[Vector], w: Vector) -> Vector:
    """
    The gradient of directional variance with respect to w
    """
    w_dir = direction(w)
    return [sum(2 * dot(v, w_dir) * v[i] for v in data)
            for i in range(len(w))]
```

1.5 경사 상승법 (Gradient Ascent)

In [5]:

```
from scratch.gradient_descent import gradient_step
import tqdm

def first_principal_component(data: List[Vector],
                              n: int = 100,
                              step_size: float = 0.1) -> Vector:
    # Start with a random guess
    guess = [1.0 for _ in data[0]]

    with tqdm.trange(n) as t:
        for _ in t:
            dv = directional_variance(data, guess)
            gradient = directional_variance_gradient(data, guess)
            guess = gradient_step(guess, gradient, step_size)
            t.set_description(f"dv: {dv:.3f}")

    return direction(guess)
```

1.6 투영

In [6]:

```
from scratch.linear_algebra import scalar_multiply

def project(v: Vector, w: Vector) -> Vector:
    """return the projection of v onto the direction w"""
    projection_length = dot(v, w)
    return scalar_multiply(projection_length, w)
```

1.7 주성분 투영 제거

In [7]:

```
from scratch.linear_algebra import subtract

def remove_projection_from_vector(v: Vector, w: Vector) -> Vector:
    """projects v onto w and subtracts the result from v"""
    return subtract(v, project(v, w))

def remove_projection(data: List[Vector], w: Vector) -> List[Vector]:
    return [remove_projection_from_vector(v, w) for v in data]
```

1.8 PCA 알고리즘

In [8]:

```
def pca(data: List[Vector], num_components: int) -> List[Vector]:
    components: List[Vector] = []
    for _ in range(num_components):
        component = first_principal_component(data)
        components.append(component)
        data = remove_projection(data, component)

    return components
```

1.9 차원 축소

In [9]:

```
def transform_vector(v: Vector, components: List[Vector]) -> Vector:
    return [dot(v, w) for w in components]

def transform(data: List[Vector], components: List[Vector]) -> List[Vector]:
    return [transform_vector(v, components) for v in data]
```

Numpy Tutorial

In [1]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:₩Users₩이준용₩.conda₩envs₩data_mining₩lib₩site-packages (1.19.2)
```

1. 넘파이 소개

In [2]:
```
import numpy as np
import time
```

Numpy 속도 테스트

In [3]:
```
#test speed
def sum_trad():
    start = time.time()
    X = range(10000000)
    Y = range(10000000)
    Z = []
    for i in range(len(X)):
        Z.append(X[i] + Y[i])
    return time.time() - start

def sum_numpy():
    start = time.time()
    X = np.arange(10000000)
    Y = np.arange(10000000)
    Z=X+Y
    return time.time() - start

print ('time sum:',sum_trad(),'  time sum numpy:',sum_numpy())
```

```
time sum: 8.752399921417236   time sum numpy: 0.12499189376831055
```

2. 배열 생성

2.1 단일 값으로 초기화

In [4]:
```
# 단일 값으로 초기화된 배열 생성
print(np.full((3, 3), np.inf))
print(np.full((3, 3), 10.1))
```

```
[[inf inf inf]
 [inf inf inf]
 [inf inf inf]]
[[10.1 10.1 10.1]
 [10.1 10.1 10.1]
 [10.1 10.1 10.1]]
```

In [5]:
```
# 배열을 단일 값으로 리셋
arr = np.array([10, 20, 33], float)
print(arr)
```

```
[10. 20. 33.]
```

In [6]:
```
arr.fill(1)
print(arr)
```

```
[1. 1. 1.]
```

2.2 랜덤 초기화

In [7]:
```
# 정수 순열로 초기화된 배열 생성
np.random.permutation(3)
```

Out[7]:
```
array([1, 0, 2])
```

In [8]:

```
# 균등분포로 초기화된 배열 생성
np.random.random(5)
```

Out[8]:

```
array([0.49268642, 0.73910484, 0.29046511, 0.03140259, 0.13389864])
```

In [9]:

```
# 균등분포로 초기화된 2차원 배열 생성
np.random.rand(2,3)
```

Out[9]:

```
array([[0.05621246, 0.26423437, 0.52018664],
       [0.96686911, 0.84310316, 0.0886853 ]])
```

In [10]:

```
# 정규분포로 초기화된 배열 생성
np.random.normal(0,1,5) # 평균, 표준 편차
```

Out[10]:

```
array([ 0.46859293,  2.81061461,  0.78674686,  0.62688269, -1.82745497])
```

In [11]:

```
# 다변량 정규분포로 초기화된 2차원 배열 생성
np.random.multivariate_normal([10, 0], [[3, 1], [1, 4]], size=[5,]) # 평균, 공분산, 배열 형태
```

Out[11]:

```
array([[ 7.48487466, -0.57755571],
       [11.33813017, -0.23236418],
       [10.6453999 , -0.65072269],
       [ 6.16884455, -1.61783138],
       [12.02430554,  0.26126015]])
```

2.3 리스트에서 생성

In [12]:

```
#리스트에서 배열 생성
arr = np.array([2, 6, 5, 9], float)
print (arr)
print (type(arr))
```

```
[2. 6. 5. 9.]
<class 'numpy.ndarray'>
```

In [13]:

```
# 배열에서 리스트로 변환
arr = np.array([1, 2, 3], float)
print (arr.tolist())
print (list(arr))
```

```
[1.0, 2.0, 3.0]
[1.0, 2.0, 3.0]
```

2.4 배열 복사

In [14]:

```
arr = np.array([1, 2, 3], float)
arr1 = arr
arr2 = arr.copy()
arr[0] = 0
print (arr)
print (arr1)
print (arr2)
```

```
[0. 2. 3.]
[0. 2. 3.]
[1. 2. 3.]
```

2.5 단위 행렬

In [15]:

```
np.identity(3, dtype=int)
```

Out[15]:

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])
```

In [16]:

```
np.identity(3)
```

Out[16]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

2.6 영행렬, 1행렬

```
In [17]: np.zeros(6, dtype=int)
```

```
Out[17]: array([0, 0, 0, 0, 0, 0])
```

```
In [18]: np.ones((2,3), dtype=float)
```

```
Out[18]: array([[1., 1., 1.],
                [1., 1., 1.]])
```

```
In [19]: # 주어진 배열과 동일한 차원의 영행렬, 1행렬 생성
         arr = np.array([[13, 32, 31], [64, 25, 76]], float)
         np.zeros_like(arr)
```

```
Out[19]: array([[0., 0., 0.],
                [0., 0., 0.]])
```

```
In [20]: np.ones_like(arr)
```

```
Out[20]: array([[1., 1., 1.],
                [1., 1., 1.]])
```

3. 배열 관리

3.1 배열의 크기와 데이터 타입

```
In [21]: # 배열의 모양 확인
         arr.shape
```

```
Out[21]: (2, 3)
```

```
In [22]: # 1차원의 길이
         arr = np.array([[ 4., 5., 6.], [ 2., 3., 6.]], float)
         len(arr)
```

```
Out[22]: 2
```

```
In [23]: # 데이터 타입 확인
         arr.dtype
```

```
Out[23]: dtype('float64')
```

```
In [25]: # 데이터 타입 변환
         int_arr = matrix.astype(np.int32)
         int_arr
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-25-417850dc9a7f> in <module>
      1 # 데이터 타입 변환
----> 2 int_arr = matrix.astype(np.int32)
      3 int_arr

NameError: name 'matrix' is not defined
```

3.2 배열 읽기/쓰기

```
In [26]: arr = np.array([2., 6., 5., 5.])
         print(arr[:3])
         print(arr[3])
         arr[0] = 5.
         print(arr)
```

```
[2. 6. 5.]
5.0
[5. 6. 5. 5.]
```

3.3 배열 슬라이싱

In [27]:
```
# 슬라이싱
matrix = np.array([[ 4., 5., 6.], [2, 3, 6]], float)
print(matrix)
```

```
[[4. 5. 6.]
 [2. 3. 6.]]
```

In [28]:
```
arr = np.array([[ 4., 5., 6.], [ 2., 3., 6.]], float)
print(arr[1:2,2:3])
print(arr[1,:])
print(arr[:,2])
print(arr[-1:,-2:])
```

```
[[6.]]
[2. 3. 6.]
[6. 6.]
[[3. 6.]]
```

3.4 배열 쿼리

In [29]:
```
# 인덱스 배열로 쿼리
arr1 = np.array([1, 4, 5, 9], float)
arr2 = np.array([0, 1, 1, 3, 1, 1, 1], int)
print(arr1[arr2])
```

```
[1. 4. 4. 9. 4. 4. 4.]
```

In [30]:
```
# 다차원 배열 인덱스 배열로 쿼리
arr1 = np.array([[1, 2], [5, 13]], float)
arr2 = np.array([1, 0, 0, 1], int)
arr3 = np.array([1, 1, 0, 1], int)
print(arr1[arr2,arr3])
```

```
[13.  2.  1. 13.]
```

In [31]:
```
# 논리 행렬로 쿼리
arr = np.arange(25).reshape(5,5)
print(arr % 2 == 0)
arr[arr % 2 == 0] = 0
print(arr)
```

```
[[ True False  True False  True]
 [False  True False  True False]
 [ True False  True False  True]
 [False  True False  True False]
 [ True False  True False  True]]
[[ 0  1  0  3  0]
 [ 5  0  7  0  9]
 [ 0 11  0 13  0]
 [15  0 17  0 19]
 [ 0 21  0 23  0]]
```

In [32]:
```
# where 조건으로 쿼리
a = np.arange(10)
print(a)
np.where(a < 5, a, 10*a)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

Out[32]:
```
array([ 0,  1,  2,  3,  4, 50, 60, 70, 80, 90])
```

3.5 중복 제거

In [33]:
```
arr = np.array([2., 6., 5., 5.])
print(np.unique(arr))
```

```
[2. 5. 6.]
```

3.6 정렬/섞기

In [34]:
```
# 정렬
np.sort(arr)
```

Out[34]:
```
array([2., 5., 5., 6.])
```

In [35]:

```
# 정렬해서 인덱스 배열 생성
np.argsort(arr)
```

Out[35]:

```
array([0, 2, 3, 1], dtype=int64)
```

In [36]:

```
# 랜덤하게 섞기
np.random.shuffle(arr)
arr
```

Out[36]:

```
array([5., 5., 2., 6.])
```

In [37]:

```
# 배열 비교
np.array_equal(arr,np.array([1,3,2]))
```

Out[37]:

```
False
```

3.7 1차원 배열로 펴기

In [38]:

```
# Flattening
arr = np.array([[10, 29, 23], [24, 25, 46]], float)
print(arr)
print(arr.flatten())
```

```
[[10. 29. 23.]
 [24. 25. 46.]]
[10. 29. 23. 24. 25. 46.]
```

3.8 재배열

In [39]:

```
# 배열의 재배열
arr = np.array(range(8), float)
print(arr)
arr = arr.reshape((4,2))
print(arr)
print(arr.shape)
```

```
[0. 1. 2. 3. 4. 5. 6. 7.]
[[0. 1.]
 [2. 3.]
 [4. 5.]
 [6. 7.]]
(4, 2)
```

3.9 전치 행렬

In [40]:

```
arr = np.array(range(6), float).reshape((2, 3))
print(arr)
print(arr.transpose())
```

```
[[0. 1. 2.]
 [3. 4. 5.]]
[[0. 3.]
 [1. 4.]
 [2. 5.]]
```

In [41]:

```
# 행렬의 T 속성으로 전치하기
matrix = np.arange(15).reshape((3, 5))
print(matrix)
print(matrix .T)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
[[ 0  5 10]
 [ 1  6 11]
 [ 2  7 12]
 [ 3  8 13]
 [ 4  9 14]]
```

3.10 차원 늘리기

In [42]:

```
# newaxis로 차원 늘리기
arr = np.array([14, 32, 13], float)
print(arr)
print(arr[:,np.newaxis])
print(arr[:,np.newaxis].shape)
print(arr[np.newaxis,:])
print(arr[np.newaxis,:].shape)
```

```
[14. 32. 13.]
[[14.]
 [32.]
 [13.]]
(3, 1)
[[14. 32. 13.]]
(1, 3)
```

3.11 배열 결합

In [43]:

```
arr1 = np.array([[11, 12], [32, 42]], float)
arr2 = np.array([[54, 26], [27, 28]], float)
print(np.concatenate((arr1,arr2)))

# 1차원 방향으로 결합
print(np.concatenate((arr1,arr2), axis=0))

# 2차원 방향으로 결합
print(np.concatenate((arr1,arr2), axis=1))
```

```
[[11. 12.]
 [32. 42.]
 [54. 26.]
 [27. 28.]]
[[11. 12.]
 [32. 42.]
 [54. 26.]
 [27. 28.]]
[[11. 12. 54. 26.]
 [32. 42. 27. 28.]]
```

3.12 배열 쌓기

In [44]:

```
x = np.array([2, 3, 4])
y = np.array([3, 4, 5])
```

In [45]:

```
# 1차원으로 쌓기 (행으로 쌓기)
np.stack((x, y))
```

Out[45]:

```
array([[2, 3, 4],
       [3, 4, 5]])
```

In [46]:

```
# 2차원으로 쌓기 (열로 쌓기)
np.stack((x, y), axis=-1)
```

Out[46]:

```
array([[2, 3],
       [3, 4],
       [4, 5]])
```

4. 배열 연산

4.1 산술연산

In [47]:

```
#array operations
arr1 = np.array([1,2,3], float)
arr2 = np.array([1,2,3], float)
print(arr1+arr2)
print(arr1-arr2)
print(arr1 * arr2)
print(arr2 / arr1)
print(arr1 % arr2)
print(arr2**arr1)
```

```
[2. 4. 6.]
[0. 0. 0.]
[1. 4. 9.]
[1. 1. 1.]
[0. 0. 0.]
[ 1.  4. 27.]
```

4.2 브로드캐스팅

In [48]:

```
arr1 = np.zeros((2,2), float)
arr2 = np.array([1., 2.], float)

print(arr1)
print(arr2)
print(arr1 + arr2)
```

```
[[0. 0.]
 [0. 0.]]
[1. 2.]
[[1. 2.]
 [1. 2.]]
```

In [49]:

```
# 배열의 브로드캐스팅 방식을 명시하고 싶다면 newaxis 상수를 이용해서 확장해야 할 축을 지정
arr1 = np.zeros((2,2), float)
arr2 = np.array([1., 2.], float)
print( arr1 + arr2[np.newaxis,:])
print(arr1 + arr2[:,np.newaxis])
```

```
[[1. 2.]
 [1. 2.]]
[[1. 1.]
 [2. 2.]]
```

4.3 논리 연산

In [50]:

```
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
X1 = arr%2 == 0
print(X1)
```

```
[[False  True False]
 [ True False  True]]
```

In [51]:

```
X2 = arr >= 4
print(X2)
```

```
[[False False False]
 [ True  True  True]]
```

In [52]:

```
X3 = X1*1
print(X3)
```

```
[[0 1 0]
 [1 0 1]]
```

In [53]:

```
np.logical_and(X1, X2)
```

Out[53]:

```
array([[False, False, False],
       [ True, False,  True]])
```

5. 선형대수 연산

5.1 벡터의 곱

In [54]:
```
arr1 = np.array([12, 43, 10], float)
arr2 = np.array([21, 42, 14], float)
```

In [55]:
```
# Inner Product
np.inner(arr1, arr2)
```

Out[55]:
```
2198.0
```

In [56]:
```
# outer Product
np.outer(arr1, arr2)
```

Out[56]:
```
array([[ 252.,  504.,  168.],
       [ 903., 1806.,  602.],
       [ 210.,  420.,  140.]])
```

In [57]:
```
# Cross Product
np.cross(arr1, arr2)
```

Out[57]:
```
array([ 182.,   42., -399.])
```

5.2 행렬곱

In [58]:
```
#linear algebra operations
X = np.arange(15).reshape((3, 5))
print(X)
print(X.T)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
[[ 0  5 10]
 [ 1  6 11]
 [ 2  7 12]
 [ 3  8 13]
 [ 4  9 14]]
```

In [59]:
```
np.dot(X.T, X)
```

Out[59]:
```
array([[125, 140, 155, 170, 185],
       [140, 158, 176, 194, 212],
       [155, 176, 197, 218, 239],
       [170, 194, 218, 242, 266],
       [185, 212, 239, 266, 293]])
```

5.3 행렬식, 역행렬

In [60]:
```
# 행렬식
matrix = np.array([[74, 22, 10], [92, 31, 17], [21, 22, 12]], float)
print(matrix)
print(np.linalg.det(matrix))
```

```
[[74. 22. 10.]
 [92. 31. 17.]
 [21. 22. 12.]]
-2852.000000000003
```

In [61]:
```
# 역행렬
inv_matrix = np.linalg.inv(matrix)
print(inv_matrix)
print(np.dot(inv_matrix,matrix))
```

```
[[ 0.00070126  0.01542777 -0.02244039]
 [ 0.26192146 -0.23772791  0.11851332]
 [-0.48141655  0.4088359  -0.09467041]]
[[ 1.00000000e+00 -1.66533454e-16 -5.55111512e-17]
 [ 3.99680289e-15  1.00000000e+00  8.88178420e-16]
 [-3.02535774e-15 -1.05471187e-15  1.00000000e+00]]
```

5.4 고윳값, 고유벡터

In [62]:

```
matrix = np.array([[74, 22, 10], [92, 31, 17], [21, 22, 12]], float)
# 고윳값, 고유벡터
vals, vecs = np.linalg.eig(matrix)
print(vals)
```

```
[107.99587441  11.33411853  -2.32999294]
```

In [63]:

```
print(vecs)
```

```
[[-0.57891525 -0.21517959  0.06319955]
 [-0.75804695  0.17632618 -0.58635713]
 [-0.30036971  0.96052424  0.80758352]]
```

5.5 통계 함수

In [64]:

```
arr = np.random.rand(8, 4)
arr.mean() # 평균
```

Out[64]:

```
0.49576006558389374
```

In [65]:

```
np.mean(arr) # 평균
```

Out[65]:

```
0.49576006558389374
```

In [66]:

```
arr.std() # 표준 편차
```

Out[66]:

```
0.2914325371496337
```

In [67]:

```
arr.var() # 분산
```

Out[67]:

```
0.08493292370947261
```

In [68]:

```
arr.sum() # 합산
```

Out[68]:

```
15.8643220986846
```

In [69]:

```
arr.min() # 최소
```

Out[69]:

```
0.047952919314352 97
```

In [70]:

```
arr.max() # 최대
```

Out[70]:

```
0.9939627530921408
```

In [71]:

```
arr.argmin() # 최소 값의 인덱스
```

Out[71]:

```
11
```

In [72]:

```
arr.argmax() # 최대 값의 인덱스
```

Out[72]:

```
14
```

10장. 차원 축소 (Dimension Reduction)

1. 차원 축소 (Dimension Reduction)

1.1 평균 빼기

In [1]:

```
from typing import List
from scratch.linear_algebra import Vector, subtract

def de_mean(data: List[Vector]) -> List[Vector]:
    """Recenters the data to have mean 0 in every dimension"""
    mean = vector_mean(data)
    return [subtract(vector, mean) for vector in data]
```

1.2 단위 벡터 만들기

In [2]:

```
from scratch.linear_algebra import magnitude

def direction(w: Vector) -> Vector:
    mag = magnitude(w)
    return [w_i / mag for w_i in w]
```

1.3 단위 벡터 방향으로 분산 구하기 (목적 함수)

In [3]:

```
from scratch.linear_algebra import dot

def directional_variance(data: List[Vector], w: Vector) -> float:
    """
    Returns the variance of x in the direction of w
    """
    w_dir = direction(w)
    return sum(dot(v, w_dir) ** 2 for v in data)
```

1.4 단위 벡터 방향의 분산에 대한 그래디언트

In [4]:

```
def directional_variance_gradient(data: List[Vector], w: Vector) -> Vector:
    """
    The gradient of directional variance with respect to w
    """
    w_dir = direction(w)
    return [sum(2 * dot(v, w_dir) * v[i] for v in data)
            for i in range(len(w))]
```

1.5 경사 상승법 (Gradient Ascent)

In [5]:

```
from scratch.gradient_descent import gradient_step
import tqdm

def first_principal_component(data: List[Vector],
                              n: int = 100,
                              step_size: float = 0.1) -> Vector:
    # Start with a random guess
    guess = [1.0 for _ in data[0]]

    with tqdm.trange(n) as t:
        for _ in t:
            dv = directional_variance(data, guess)
            gradient = directional_variance_gradient(data, guess)
            guess = gradient_step(guess, gradient, step_size)
            t.set_description(f"dv: {dv:.3f}")

    return direction(guess)
```

1.6 투영

In [6]:

```
from scratch.linear_algebra import scalar_multiply

def project(v: Vector, w: Vector) -> Vector:
    """return the projection of v onto the direction w"""
    projection_length = dot(v, w)
    return scalar_multiply(projection_length, w)
```

1.7 주성분 투영 제거

In [7]:

```
from scratch.linear_algebra import subtract

def remove_projection_from_vector(v: Vector, w: Vector) -> Vector:
    """projects v onto w and subtracts the result from v"""
    return subtract(v, project(v, w))

def remove_projection(data: List[Vector], w: Vector) -> List[Vector]:
    return [remove_projection_from_vector(v, w) for v in data]
```

1.8 PCA 알고리즘

In [8]:

```
def pca(data: List[Vector], num_components: int) -> List[Vector]:
    components: List[Vector] = []
    for _ in range(num_components):
        component = first_principal_component(data)
        components.append(component)
        data = remove_projection(data, component)

    return components
```

1.9 차원 축소

In [9]:

```
def transform_vector(v: Vector, components: List[Vector]) -> Vector:
    return [dot(v, w) for w in components]

def transform(data: List[Vector], components: List[Vector]) -> List[Vector]:
    return [transform_vector(v, components) for v in data]
```

17장. 결정 트리 (Decision Tree)

1. 데이터셋 (인터뷰 후보자 합격 예측)

1.1 데이터 NamedTuple

In [1]:

```
from typing import NamedTuple, Optional

class Candidate(NamedTuple):
    level: str
    lang: str
    tweets: bool
    phd: bool
    did_well: Optional[bool] = None  # allow unlabeled data
```

1.2 데이터셋

In [2]:

```
                  #  level     lang     tweets  phd  did_well
inputs = [Candidate('Senior', 'Java',   False, False, False),
          Candidate('Senior', 'Java',   False, True,  False),
          Candidate('Mid',    'Python', False, False, True),
          Candidate('Junior', 'Python', False, False, True),
          Candidate('Junior', 'R',      True,  False, True),
          Candidate('Junior', 'R',      True,  True,  False),
          Candidate('Mid',    'R',      True,  True,  True),
          Candidate('Senior', 'Python', False, False, False),
          Candidate('Senior', 'R',      True,  False, True),
          Candidate('Junior', 'Python', True,  False, True),
          Candidate('Senior', 'Python', True,  True,  True),
          Candidate('Mid',    'Python', False, True,  True),
          Candidate('Mid',    'Java',   True,  False, True),
          Candidate('Junior', 'Python', False, True,  False)
         ]
```

2. 엔트로피 (Entropy)

2.1 엔트로피

In [3]:

```
from typing import List
import math

def entropy(class_probabilities: List[float]) -> float:
    """Given a list of class probabilities, compute the entropy"""
    return sum(-p * math.log(p, 2)
               for p in class_probabilities
               if p > 0)                     # ignore zero probabilities
```

In [4]:

```
assert entropy([1.0]) == 0
assert entropy([0.5, 0.5]) == 1
assert 0.81 < entropy([0.25, 0.75]) < 0.82
```

2.2 클래스 별 확률

In [5]:

```
from typing import Any
from collections import Counter

def class_probabilities(labels: List[Any]) -> List[float]:
    total_count = len(labels)
    return [count / total_count
            for count in Counter(labels).values()]
```

2.3 데이터 엔트로피

In [6]:

```
def data_entropy(labels: List[Any]) -> float:
    return entropy(class_probabilities(labels))
```

In [7]:

```
assert data_entropy(['a']) == 0
assert data_entropy([True, False]) == 1
assert data_entropy([3, 4, 4, 4]) == entropy([0.25, 0.75])
```

3. 노드 분할 방식

3.1 속성 값에 따라 데이터 분할

In [8]:

```
from typing import Dict, TypeVar
from collections import defaultdict

T = TypeVar('T')  # generic type for inputs

def partition_by(inputs: List[T], attribute: str) -> Dict[Any, List[T]]:
    """Partition the inputs into lists based on the specified attribute."""
    partitions: Dict[Any, List[T]] = defaultdict(list)
    for input in inputs:
        key = getattr(input, attribute)  # value of the specified attribute
        partitions[key].append(input)    # add input to the correct partition
    return partitions
```

3.2 분할 엔트로피

In [9]:

```
def partition_entropy(subsets: List[List[Any]]) -> float:
    """Returns the entropy from this partition of data into subsets"""
    total_count = sum(len(subset) for subset in subsets)

    return sum(data_entropy(subset) * len(subset) / total_count
               for subset in subsets)
```

3.3 데이터 분할 및 분할 엔트로피 계산

In [10]:

```
def partition_entropy_by(inputs: List[Any],
                         attribute: str,
                         label_attribute: str) -> float:
    """Compute the entropy corresponding to the given partition"""
    # partitions consist of our inputs
    partitions = partition_by(inputs, attribute)

    # but partition_entropy needs just the class labels
    labels = [[getattr(input, label_attribute) for input in partition]
              for partition in partitions.values()]

    return partition_entropy(labels)
```

3.4 분할 속성 탐색

3.4.1 1차 분할 속성 탐색

In [11]:

```
for key in ['level','lang','tweets','phd']:
    print(key, partition_entropy_by(inputs, key, 'did_well'))

assert 0.69 < partition_entropy_by(inputs, 'level', 'did_well')  < 0.70
assert 0.86 < partition_entropy_by(inputs, 'lang', 'did_well')   < 0.87
assert 0.78 < partition_entropy_by(inputs, 'tweets', 'did_well') < 0.79
assert 0.89 < partition_entropy_by(inputs, 'phd', 'did_well')    < 0.90
```

```
level 0.6935361388961919
lang 0.8601317128547441
tweets 0.7884504573082896
phd 0.8921589282623617
```

3.4.2 2차 분할 속성 탐색 (직급 별로 파티션 엔트로피 비교)

In [12]:

```
senior_inputs = [input for input in inputs if input.level == 'Senior']

assert 0.4 == partition_entropy_by(senior_inputs, 'lang', 'did_well')
assert 0.0 == partition_entropy_by(senior_inputs, 'tweets', 'did_well')
assert 0.95 < partition_entropy_by(senior_inputs, 'phd', 'did_well') < 0.96
```

In [13]:

```
mid_inputs = [input for input in inputs if input.level == 'Mid']

print(partition_entropy_by(mid_inputs, 'lang', 'did_well'))
print( partition_entropy_by(mid_inputs, 'tweets', 'did_well'))
print(partition_entropy_by(mid_inputs, 'phd', 'did_well'))
```

```
0.0
0.0
0.0
```

In [14]:

```
junior_inputs = [input for input in inputs if input.level == 'Junior']

print(partition_entropy_by(junior_inputs, 'lang', 'did_well'))
print( partition_entropy_by(junior_inputs, 'tweets', 'did_well'))
print(partition_entropy_by(junior_inputs, 'phd', 'did_well'))
```

```
0.9509775004326938
0.9509775004326938
0.0
```

4. 의사 결정 트리 만들기

4.1 리프/결정 노드 정의

In [15]:

```
from typing import NamedTuple, Union, Any

class Leaf(NamedTuple):
    value: Any

class Split(NamedTuple):
    attribute: str
    subtrees: dict
    default_value: Any = None

DecisionTree = Union[Leaf, Split]
```

4.2 트리 구성 예시

In [16]:

```
hiring_tree = Split('level', {   # First, consider "level".
    'Junior': Split('phd', {     # if level is "Junior", next look at "phd"
        False: Leaf(True),       #   if "phd" is False, predict True
        True: Leaf(False)        #   if "phd" is True, predict False
    }),
    'Mid': Leaf(True),           # if level is "Mid", just predict True
    'Senior': Split('tweets', {  # if level is "Senior", look at "tweets"
        False: Leaf(False),      #   if "tweets" is False, predict False
        True: Leaf(True)         #   if "tweets" is True, predict True
    })
})
```

4.3 클래스 분류

In [17]:

```
def classify(tree: DecisionTree, input: Any) -> Any:
    """classify the input using the given decision tree"""

    # If this is a leaf node, return its value
    if isinstance(tree, Leaf):
        return tree.value

    # Otherwise this tree consists of an attribute to split on
    # and a dictionary whose keys are values of that attribute
    # and whose values of are subtrees to consider next
    subtree_key = getattr(input, tree.attribute)

    if subtree_key not in tree.subtrees:   # If no subtree for key,
        return tree.default_value          # return the default value.

    subtree = tree.subtrees[subtree_key]   # Choose the appropriate subtree
    return classify(subtree, input)        # and use it to classify the input.
```

4.4 의사 결정 트리 생성 (build_tree_id3)

In [18]:

```
def build_tree_id3(inputs: List[Any],
                   split_attributes: List[str],
                   target_attribute: str) -> DecisionTree:
    # Count target labels
    label_counts = Counter(getattr(input, target_attribute)
                           for input in inputs)
    most_common_label = label_counts.most_common(1)[0][0]

    # If there's a unique label, predict it
    if len(label_counts) == 1:
        return Leaf(most_common_label)

    # If no split attributes left, return the majority label
    if not split_attributes:
        return Leaf(most_common_label)

    # Otherwise split by the best attribute

    def split_entropy(attribute: str) -> float:
        """Helper function for finding the best attribute"""
        return partition_entropy_by(inputs, attribute, target_attribute)

    best_attribute = min(split_attributes, key=split_entropy)

    partitions = partition_by(inputs, best_attribute)
    new_attributes = [a for a in split_attributes if a != best_attribute]

    # recursively build the subtrees
    subtrees = {attribute_value : build_tree_id3(subset,
                                                 new_attributes,
                                                 target_attribute)
                for attribute_value, subset in partitions.items()}

    return Split(best_attribute, subtrees, default_value=most_common_label)
```

4.5 테스트

In [19]:

```
tree = build_tree_id3(inputs,
                      ['level', 'lang', 'tweets', 'phd'],
                      'did_well')

# Should predict True
assert classify(tree, Candidate("Junior", "Java", True, False))

# Should predict False
assert not classify(tree, Candidate("Junior", "Java", True, True))

# Should predict True
assert classify(tree, Candidate("Intern", "Java", True, True))
```

17장. 결정 트리 (Decision Tree)

1. 패키지 설치

1.1 머신 러닝 패키지 scikit-learn

In [1]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (0.0)
Requirement already satisfied: scikit-learn in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from sklearn) (0.2
4.2)
Requirement already satisfied: numpy>=1.13.3 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from scikit-learn
->sklearn) (1.20.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from sciki
t-learn->sklearn) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from scikit-learn-
>sklearn) (1.0.1)
Requirement already satisfied: scipy>=0.19.1 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from scikit-learn
->sklearn) (1.6.2)
```

1.2 시각화 패키지 graphviz

- 바이너리 설치 : https://graphviz.org/download/ (https://graphviz.org/download/)
- 패키지 설치

In [2]:
```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (0.16)
```

2. 데이터셋

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import learning_curve, train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import graphviz
%matplotlib inline
```

2.2 데이터셋 로딩

In [4]:
```
import requests
import os

dataset_path = os.path.join('data', 'iris.data')
if os.path.exists(dataset_path) is False:
    data = requests.get("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")
    with open(dataset_path, "w") as f:
      f.write(data.text)
```

In [5]:
```
import pandas as pd

column_names = ['sepal length', 'sepal width', 'petal length', 'petal width', 'species']
class_names = ['Iris-setosa', 'Iris-virginica', 'Iris-versicolor']
dataset = pd.read_csv(dataset_path, names=column_names)
dataset.sample(5)
```

Out[5]:

	sepal length	sepal width	petal length	petal width	species
109	7.2	3.6	6.1	2.5	Iris-virginica
5	5.4	3.9	1.7	0.4	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
112	6.8	3.0	5.5	2.1	Iris-virginica
49	5.0	3.3	1.4	0.2	Iris-setosa

In [6]:
```
X = dataset[dataset.columns[:-1]]
dataset['target'] = (dataset['species']=='Iris-setosa')*0 + ₩
                    (dataset['species']=='Iris-virginica')*1 + ₩
                    (dataset['species']=='Iris-versicolor')*2
y = dataset['target']
```

2.3 데이터 분할

In [7]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2)
```

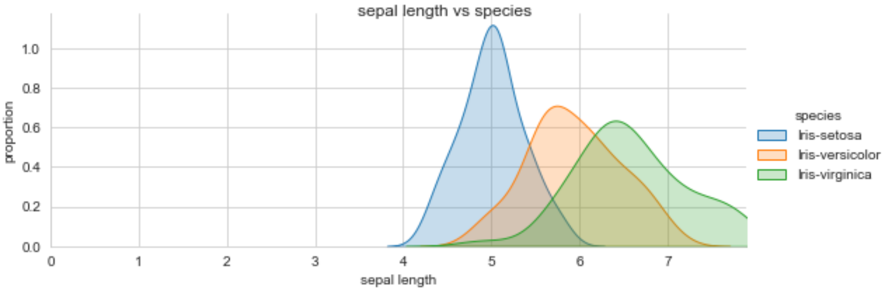
3. 데이터 탐색

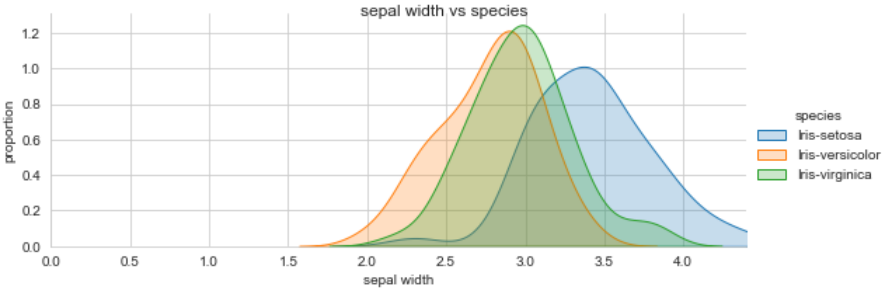
3.1 레이블 별 특징 분포 그래프

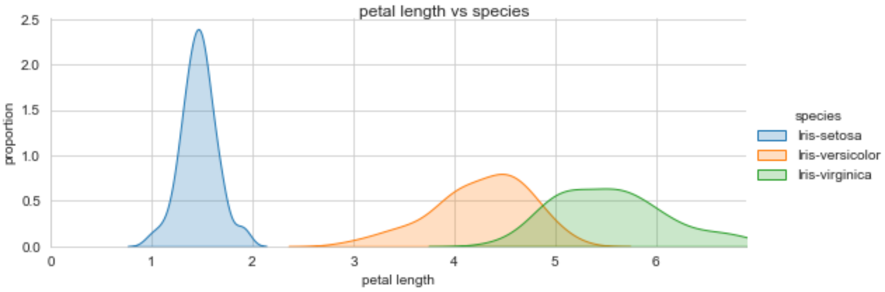
In [8]:
```
def plot_feature_by_label(dataframe, feature_name, label_name, title):
    sns.set_style("whitegrid")
    ax = sns.FacetGrid(dataframe, hue=label_name,aspect=2.5)
    ax.map(sns.kdeplot,feature_name,shade=True)
    ax.set(xlim=(0, dataframe[feature_name].max()))
    ax.add_legend()
    ax.set_axis_labels(feature_name, 'proportion')
    ax.fig.suptitle(title)
    plt.show()
```

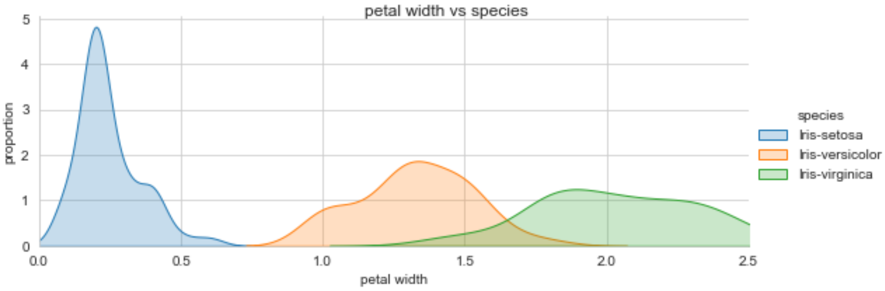
3.2 특징 별 분포

In [9]:
```
for feature_name in X.keys():
    plot_feature_by_label(dataset, feature_name, 'species', feature_name + ' vs species')
```









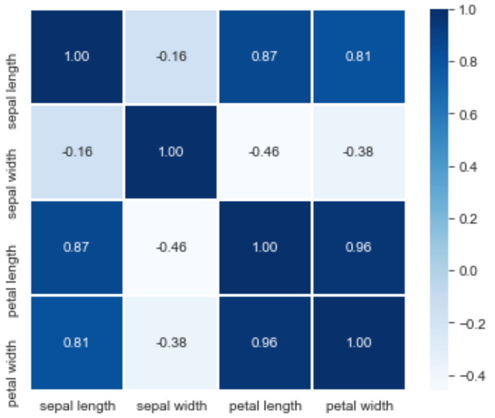
4. 특징 선택 (Feature Selection)

4.1 원래의 특징

4.1.1 히트맵으로 상관관계 확인

In [10]:
```
fig, ax = plt.subplots(figsize=(6, 5))
sns.heatmap(X_train.corr(), linewidths=.5, annot=True, fmt=".2f", cmap='Blues')
```

Out[10]:
```
<AxesSubplot:>
```



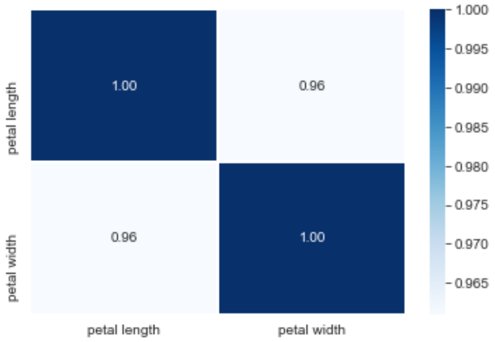
4.2 특징 선택 (Feature Selection)

In [11]:
```
dataset2 = dataset.drop(['sepal length', 'sepal width'], axis=1)
```

4.2.2 히트맵으로 상관성 재확인

In [12]:
```
X2 = dataset2[dataset2.columns[:-2]]
y2 = dataset2.target
X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X2, y2, test_size=0.2)
sns.heatmap(X_train2.corr(), linewidths=.5, annot=True, fmt=".2f", cmap='Blues')
```

Out[12]:
```
<AxesSubplot:>
```



4.3 특징 추출 (Feature Extraction)

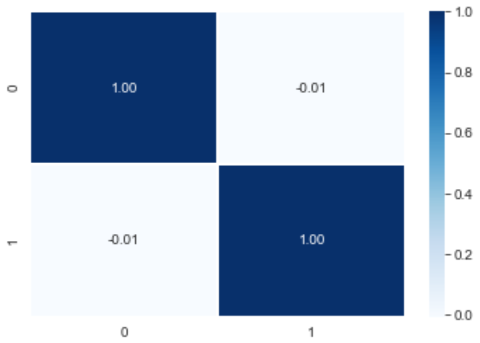
4.3.1 PCA 차원 축소

In [13]:
```
from sklearn.decomposition import PCA
X3,y3 = X,y
variance_pct = 2
pca = PCA(n_components=variance_pct) # Create PCA object
X_transformed = pca.fit_transform(X3,y3) # Transform the initial features
```

4.3.2 히트맵으로 상관성 재확인

In [14]:
```
X3pca = pd.DataFrame(X_transformed) # Create a data frame from the PCA'd data
X_train3, X_test3, Y_train3, Y_test3 = train_test_split(X3pca, y3, test_size=0.2)
sns.heatmap(X_train3.corr(), linewidths=.5, annot=True, fmt=".2f", cmap='Blues')
```

Out[14]:
```
<AxesSubplot:>
```



5. 모델 훈련 및 성능 비교

5.1 세 모델 훈련

In [15]:
```
clf1 = tree.DecisionTreeClassifier(max_depth=2,min_samples_leaf=12)
clf1.fit(X_train, Y_train)
clf2 = tree.DecisionTreeClassifier(max_depth=2,min_samples_leaf=12)
clf2.fit(X_train2, Y_train2)
clf3 = tree.DecisionTreeClassifier(max_depth=2,min_samples_leaf=12)
clf3.fit(X_train3, Y_train3)
```

Out[15]:
```
DecisionTreeClassifier(max_depth=2, min_samples_leaf=12)
```

5.2 세 모델의 성능

In [16]:
```
print('Accuracy of Decision Tree classifier on original training set: {:.2f}'.format(clf1.score(X_train, Y_train)))
print('Accuracy of Decision Tree classifier on original test set: {:.2f}'.format(clf1.score(X_test, Y_test)))
print('Accuracy of Decision Tree classifier on reduced training set: {:.2f}'.format(clf2.score(X_train2, Y_train2)))
print('Accuracy of Decision Tree classifier on reduced test set: {:.2f}'.format(clf2.score(X_test2, Y_test2)))
print('Accuracy of Decision Tree classifier on PCA-transformed training set: {:.2f}'.format(clf3.score(X_train3, Y_train3)))
print('Accuracy of Decision Tree classifier on PCA-transformed test set: {:.2f}'.format(clf3.score(X_test3, Y_test3)))
```

```
Accuracy of Decision Tree classifier on original training set: 0.97
Accuracy of Decision Tree classifier on original test set: 0.93
Accuracy of Decision Tree classifier on reduced training set: 0.96
Accuracy of Decision Tree classifier on reduced test set: 0.93
Accuracy of Decision Tree classifier on PCA-transformed training set: 0.94
Accuracy of Decision Tree classifier on PCA-transformed test set: 0.97
```

6. 의사 결정 트리 및 주요 특징 시각화

6.1 세 모델의 특징 이름

In [17]:
```
feature_names1 = X.columns.values
feature_names2 = X2.columns.values
feature_names3 = X3pca.columns.values # [0 1 2 3 4]
```

6.2 원래의 특징

6.2.1 의사 결정 트리

In [18]:
```
def plot_decision_tree(tree_clf, feature_names, target_names):
    dot_data = tree.export_graphviz(
                        tree_clf,
                        out_file=None,
                        feature_names=feature_names,
                        class_names=target_names,
                        filled=False,
                        rounded=True,
                        special_characters=False)
    graph = graphviz.Source(dot_data)
    return graph
```

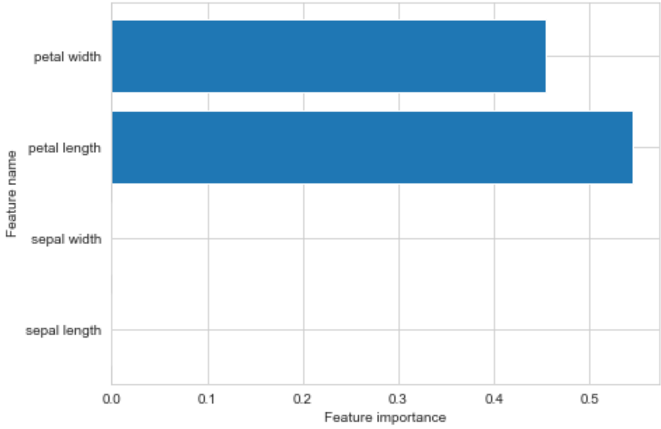
In [19]:
```
plot_decision_tree(clf1,feature_names1, class_names)
```

Out[19]:
```
<graphviz.files.Source at 0x1fb65f52460>
```

5.1.2 주요 특징 확인

In [20]:
```
def plot_feature_importances(clf, feature_names):
    c_features = len(feature_names)
    plt.barh(range(c_features), clf.feature_importances_)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature name")
    plt.yticks(np.arange(c_features), feature_names)
    plt.show()
```

In [21]:
```
fig, ax = plt.subplots(figsize=(7, 5))
plot_feature_importances(clf1, feature_names1)
```



6.3 특징 선택 (Feature Selection)

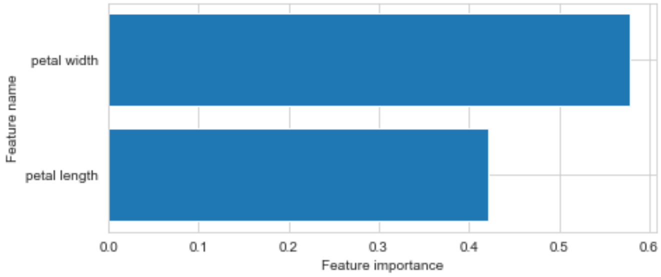
6.3.1 의사 결정 트리

In [22]:
```
plot_decision_tree(clf2,feature_names2, class_names)
```

Out[22]:
```
<graphviz.files.Source at 0x1fb63ed2b80>
```

6.3.2 주요 특징

In [23]:
```
fig, ax = plt.subplots(figsize=(7, 3))
plot_feature_importances(clf2, feature_names2)
```



6.4 특징 추출 (Feature Extraction)

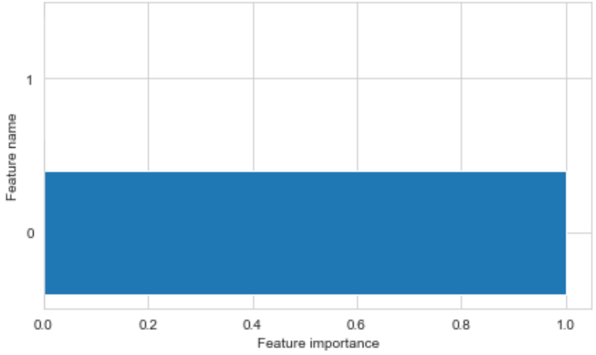
6.3.1 의사 결정 트리

In [24]:
```
plot_decision_tree(clf3,feature_names3, class_names)
```

Out[24]:
```
<graphviz.files.Source at 0x1fb6811f160>
```

6.4.2 주요 특징

In [25]:
```
fig, ax = plt.subplots(figsize=(7, 4))
plot_feature_importances(clf3, feature_names3)
```



7. 랜덤 포레스트

7.1 학습 곡선 그래프

In [26]:
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    """
    Plots a learning curve. http://scikit-learn.org/stable/modules/learning_curve.html
    """
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
    plt.legend(loc="best")
    return plt
```

7.2 혼동 행렬 그래프

In [27]:

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [28]:

```
dict_characters = {0: 'Iris-setosa', 1: 'Iris-virginica', 2: 'Iris-versicolor'}
```

7.3. 모델 훈련

In [29]:

```
(X1, y1) = (X, y)
X_train1,X_test1,Y_train1,Y_test1=train_test_split(X1,y1,random_state=0)
clf = RandomForestClassifier(max_features=4,random_state=0)
clf.fit(X_train1,Y_train1)
print('Accuracy of Random Forest Classifier on training data: {:.2f}'.format(clf.score(X_train1,Y_train1)))
print('Accuracy of Random Forest Classifier on testing data: {:.2f}'.format(clf.score(X_test1,Y_test1)))
```

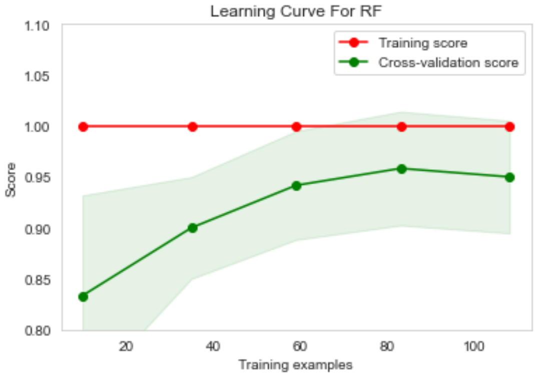
```
Accuracy of Random Forest Classifier on training data: 1.00
Accuracy of Random Forest Classifier on testing data: 0.97
```

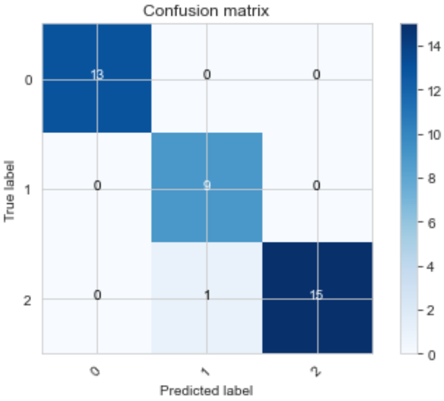
7.4. 모델 평가 (정확도 97%)

In [30]:

```
model = clf
prediction = model.predict(X_test1)
cnf_matrix = confusion_matrix(Y_test1, prediction)
```

```
In [31]: plot_learning_curve(model, 'Learning Curve For RF', X_train, Y_train, (0.80,1.1), 10)
         plt.show()
         plot_confusion_matrix(cnf_matrix, classes=dict_characters, title='Confusion matrix')
         plt.show()
```





23장. 추천 시스템 (Recommender Systems)

1. 인기 순위로 추천

1.1 데이터셋 정의

In [1]:

```
users_interests = [
    ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],
    ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],
    ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],
    ["R", "Python", "statistics", "regression", "probability"],
    ["machine learning", "regression", "decision trees", "libsvm"],
    ["Python", "R", "Java", "C++", "Haskell", "programming languages"],
    ["statistics", "probability", "mathematics", "theory"],
    ["machine learning", "scikit-learn", "Mahout", "neural networks"],
    ["neural networks", "deep learning", "Big Data", "artificial intelligence"],
    ["Hadoop", "Java", "MapReduce", "Big Data"],
    ["statistics", "R", "statsmodels"],
    ["C++", "deep learning", "artificial intelligence", "probability"],
    ["pandas", "R", "Python"],
    ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],
    ["libsvm", "regression", "support vector machines"]
]
```

1.2 관심 종목 인기 순위

In [2]:

```
from collections import Counter

popular_interests = Counter(interest
                            for user_interests in users_interests
                            for interest in user_interests)
print(popular_interests)
```

```
Counter({'Python': 4, 'R': 4, 'Big Data': 3, 'HBase': 3, 'Java': 3, 'statistics': 3, 'regression': 3, 'probability': 3, 'Hado
op': 2, 'Cassandra': 2, 'MongoDB': 2, 'Postgres': 2, 'scikit-learn': 2, 'statsmodels': 2, 'pandas': 2, 'machine learning': 2,
'libsvm': 2, 'C++': 2, 'neural networks': 2, 'deep learning': 2, 'artificial intelligence': 2, 'Spark': 1, 'Storm': 1, 'NoSQ
L': 1, 'scipy': 1, 'numpy': 1, 'decision trees': 1, 'Haskell': 1, 'programming languages': 1, 'mathematics': 1, 'theory': 1,
'Mahout': 1, 'MapReduce': 1, 'databases': 1, 'MySQL': 1, 'support vector machines': 1})
```

1.3 인기 순위로 추천

In [3]:

```
from typing import Dict, List, Tuple

def most_popular_new_interests(
        user_interests: List[str],
        max_results: int = 5) -> List[Tuple[str, int]]:
    suggestions = [(interest, frequency)
                   for interest, frequency in popular_interests.most_common()
                   if interest not in user_interests]
    return suggestions[:max_results]
```

In [4]:

```
print(most_popular_new_interests(users_interests[1]))
```

```
[('Python', 4), ('R', 4), ('Big Data', 3), ('Java', 3), ('statistics', 3)]
```

2. 사용자 기반 협업 필터링 (User-Based Collaborative Filtering)

2.1 관심사 목록

In [5]:
```
unique_interests = sorted({interest
                           for user_interests in users_interests
                           for interest in user_interests})

print(unique_interests)
assert unique_interests[:6] == [
    'Big Data',
    'C++',
    'Cassandra',
    'HBase',
    'Hadoop',
    'Haskell',
    # ...
]
```

```
['Big Data', 'C++', 'Cassandra', 'HBase', 'Hadoop', 'Haskell', 'Java', 'Mahout', 'MapReduce', 'MongoDB', 'MySQL', 'NoSQL', 'P
ostgres', 'Python', 'R', 'Spark', 'Storm', 'artificial intelligence', 'databases', 'decision trees', 'deep learning', 'libsv
m', 'machine learning', 'mathematics', 'neural networks', 'numpy', 'pandas', 'probability', 'programming languages', 'regress
ion', 'scikit-learn', 'scipy', 'statistics', 'statsmodels', 'support vector machines', 'theory']
```

2.2 사용자 별 관심사 벡터

In [6]:
```
def make_user_interest_vector(user_interests: List[str]) -> List[int]:
    """
    Given a list ofinterests, produce a vector whose ith element is 1
    if unique_interests[i] is in the list, 0 otherwise
    """
    return [1 if interest in user_interests else 0
            for interest in unique_interests]
```

In [7]:
```
user_interest_vectors = [make_user_interest_vector(user_interests)
                         for user_interests in users_interests]
```

2.3 사용자 유사도 행렬

2.3.1 코사인 유사도 (cosine similarity)

In [8]:
```
from scratch.linear_algebra import dot, Vector
import math

def cosine_similarity(v1: Vector, v2: Vector) -> float:
    return dot(v1, v2) / math.sqrt(dot(v1, v1) * dot(v2, v2))
```

2.3.2 사용자 유사도 행렬

In [9]:
```
user_similarities = [[cosine_similarity(interest_vector_i, interest_vector_j)
                      for interest_vector_j in user_interest_vectors]
                     for interest_vector_i in user_interest_vectors]
```

In [10]:
```
# Users 0 and 9 share interests in Hadoop, Java, and Big Data
assert 0.56 < user_similarities[0][9] < 0.58, "several shared interests"

# Users 0 and 8 share only one interest: Big Data
assert 0.18 < user_similarities[0][8] < 0.20, "only one shared interest"
```

2.4 유사한 사용자 목록

In [11]:
```
def most_similar_users_to(user_id: int) -> List[Tuple[int, float]]:
    pairs = [(other_user_id, similarity)                      # Find other
             for other_user_id, similarity in                 # users with
                enumerate(user_similarities[user_id])         # nonzero
             if user_id != other_user_id and similarity > 0]  # similarity.

    return sorted(pairs,                                      # Sort them
                  key=lambda pair: pair[-1],                  # most similar
                  reverse=True)                               # first.
```

In [12]:

```
most_similar_to_zero = most_similar_users_to(0)
print(most_similar_to_zero)
user, score = most_similar_to_zero[0]
assert user == 9
assert 0.56 < score < 0.57
user, score = most_similar_to_zero[1]
assert user == 1
assert 0.33 < score < 0.34
```

```
[(9, 0.5669467095138409), (1, 0.3380617018914066), (8, 0.1889822365046136), (13, 0.1690308509457033), (5, 0.154303349962091
9)]
```

2.5 사용자 기반 추천

In [13]:

```
from collections import defaultdict

def user_based_suggestions(user_id: int,
                           include_current_interests: bool = False):
    # Sum up the similarities.
    suggestions: Dict[str, float] = defaultdict(float)
    for other_user_id, similarity in most_similar_users_to(user_id):
        for interest in users_interests[other_user_id]:
            suggestions[interest] += similarity

    # Convert them to a sorted list.
    suggestions = sorted(suggestions.items(),
                         key=lambda pair: pair[-1],  # weight
                         reverse=True)

    # And (maybe) exclude already-interests
    if include_current_interests:
        return suggestions
    else:
        return [(suggestion, weight)
                for suggestion, weight in suggestions
                if suggestion not in users_interests[user_id]]
```

In [14]:

```
ubs0 = user_based_suggestions(0)
print(ubs0)
interest, score = ubs0[0]
assert interest == 'MapReduce'
assert 0.56 < score < 0.57
interest, score = ubs0[1]
assert interest == 'MongoDB'
assert 0.50 < score < 0.51
```

```
[('MapReduce', 0.5669467095138409), ('MongoDB', 0.50709255283711), ('Postgres', 0.50709255283711), ('NoSQL', 0.33806170189140
66), ('neural networks', 0.1889822365046136), ('deep learning', 0.1889822365046136), ('artificial intelligence', 0.1889822365
046136), ('databases', 0.1690308509457033), ('MySQL', 0.1690308509457033), ('Python', 0.1543033499620919), ('R', 0.1543033499
620919), ('C++', 0.1543033499620919), ('Haskell', 0.1543033499620919), ('programming languages', 0.1543033499620919)]
```

3. 아이템 기반 협업 필터링 (Item-Based Collaborative Filtering)

3.1 아이템-사용자 행렬

In [15]:

```
interest_user_matrix = [[user_interest_vector[j]
                         for user_interest_vector in user_interest_vectors]
                        for j, _ in enumerate(unique_interests)]
```

In [16]:

```
interest_user_matrix[0]
```

Out[16]:

```
[1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
```

3.2 아이템 유사도 행렬

In [17]:

```
interest_similarities = [[cosine_similarity(user_vector_i, user_vector_j)
                          for user_vector_j in interest_user_matrix]
                         for user_vector_i in interest_user_matrix]
```

3.3 유사한 아이템 목록

In [18]:
```
def most_similar_interests_to(interest_id: int):
    similarities = interest_similarities[interest_id]
    pairs = [(unique_interests[other_interest_id], similarity)
             for other_interest_id, similarity in enumerate(similarities)
             if interest_id != other_interest_id and similarity > 0]
    return sorted(pairs,
                  key=lambda pair: pair[-1],
                  reverse=True)
```

In [19]:
```
msit0 = most_similar_interests_to(0)
print(msit0)
assert msit0[0][0] == 'Hadoop'
assert 0.815 < msit0[0][1] < 0.817
assert msit0[1][0] == 'Java'
assert 0.666 < msit0[1][1] < 0.667
```

```
[('Hadoop', 0.8164965809277261), ('Java', 0.6666666666666666), ('MapReduce', 0.5773502691896258), ('Spark', 0.577350269189625
8), ('Storm', 0.5773502691896258), ('Cassandra', 0.4082482904638631), ('artificial intelligence', 0.4082482904638631), ('deep
learning', 0.4082482904638631), ('neural networks', 0.4082482904638631), ('HBase', 0.3333333333333333)]
```

3.4 아이템 기반 추천

In [20]:
```
def item_based_suggestions(user_id: int,
                           include_current_interests: bool = False):
    # Add up the similar interests
    suggestions = defaultdict(float)
    user_interest_vector = user_interest_vectors[user_id]
    for interest_id, is_interested in enumerate(user_interest_vector):
        if is_interested == 1:
            similar_interests = most_similar_interests_to(interest_id)
            for interest, similarity in similar_interests:
                suggestions[interest] += similarity

    # Sort them by weight
    suggestions = sorted(suggestions.items(),
                         key=lambda pair: pair[-1],
                         reverse=True)

    if include_current_interests:
        return suggestions
    else:
        return [(suggestion, weight)
                for suggestion, weight in suggestions
                if suggestion not in users_interests[user_id]]
```

In [21]:
```
ibs0 = item_based_suggestions(0)
print(ibs0)
assert ibs0[0][0] == 'MapReduce'
assert 1.86 < ibs0[0][1] < 1.87
assert ibs0[1][0] in ('Postgres', 'MongoDB')  # A tie
assert 1.31 < ibs0[1][1] < 1.32
```

```
[('MapReduce', 1.861807319565799), ('MongoDB', 1.3164965809277263), ('Postgres', 1.3164965809277263), ('NoSQL', 1.28445705037
61732), ('MySQL', 0.5773502691896258), ('databases', 0.5773502691896258), ('Haskell', 0.5773502691896258), ('programming lang
uages', 0.5773502691896258), ('artificial intelligence', 0.4082482904638631), ('deep learning', 0.4082482904638631), ('neural
networks', 0.4082482904638631), ('C++', 0.4082482904638631), ('Python', 0.28867513459481229), ('R', 0.28867513459481229)]
```

4. 잠재 요인 기반 협업 필터링

4.1 데이터 타입 정의

4.1.1 평점 파일 경로

In [22]:
```
# This points to the current directory, modify if your files are elsewhere.
MOVIES = "ml-100k₩₩u.item"   # pipe-delimited: movie_id|title|...
RATINGS = "ml-100k₩₩u.data"  # tab-delimited: user_id, movie_id, rating, timestamp
```

4.1.2 평점 NamedTuple

In [23]:
```
from typing import NamedTuple

class Rating(NamedTuple):
    user_id: str
    movie_id: str
    rating: float
```

4.2 데이터 읽기

In [24]:
```
import csv
# We specify this encoding to avoid a UnicodeDecodeError.
# see: https://stackoverflow.com/a/53136168/1076346
with open(MOVIES, encoding="iso-8859-1") as f:
    reader = csv.reader(f, delimiter="|")
    movies = {movie_id: title for movie_id, title, *_ in reader}

# Create a list of [Rating]
with open(RATINGS, encoding="iso-8859-1") as f:
    reader = csv.reader(f, delimiter="₩t")
    ratings = [Rating(user_id, movie_id, float(rating))
               for user_id, movie_id, rating, _ in reader]

# 1682 movies rated by 943 users
assert len(movies) == 1682
assert len(list({rating.user_id for rating in ratings})) == 943
```

4.3 데이터 탐색 (스타워즈 평점 확인)

In [25]:
```
import re

# Data structure for accumulating ratings by movie_id
star_wars_ratings = {movie_id: []
                     for movie_id, title in movies.items()
                     if re.search("Star Wars|Empire Strikes|Jedi", title)}

# Iterate over ratings, accumulating the Star Wars ones
for rating in ratings:
    if rating.movie_id in star_wars_ratings:
        star_wars_ratings[rating.movie_id].append(rating.rating)

# Compute the average rating for each movie
avg_ratings = [(sum(title_ratings) / len(title_ratings), movie_id)
               for movie_id, title_ratings in star_wars_ratings.items()]

# And then print them in order
for avg_rating, movie_id in sorted(avg_ratings, reverse=True):
    print(f"{avg_rating:.2f} {movies[movie_id]}")
```

```
4.36 Star Wars (1977)
4.20 Empire Strikes Back, The (1980)
4.01 Return of the Jedi (1983)
```

4.4 데이터 분리

In [26]:
```
import random
random.seed(0)
random.shuffle(ratings)

split1 = int(len(ratings) * 0.7)
split2 = int(len(ratings) * 0.85)

train = ratings[:split1]              # 70% of the data
validation = ratings[split1:split2]   # 15% of the data
test = ratings[split2:]               # 15% of the data
```

4.5 베이스라인 생성

In [27]:

```
avg_rating = sum(rating.rating for rating in train) / len(train)
baseline_error = sum((rating.rating - avg_rating) ** 2
                     for rating in test) / len(test)

print(avg_rating)
print(baseline_error)

# This is what we hope to do better than
assert 1.26 < baseline_error < 1.27
```

```
3.530842857142857
1.2609526646939684
```

4.6 임베딩 생성

In [28]:

```
# Embedding vectors for matrix factorization model

from scratch.deep_learning import random_tensor

EMBEDDING_DIM = 2

# Find unique ids
user_ids = {rating.user_id for rating in ratings}
movie_ids = {rating.movie_id for rating in ratings}

# Then create a random vector per id
user_vectors = {user_id: random_tensor(EMBEDDING_DIM)
                for user_id in user_ids}
movie_vectors = {movie_id: random_tensor(EMBEDDING_DIM)
                    for movie_id in movie_ids}
```

```
<Figure size 432x288 with 0 Axes>
```

4.7 모델 학습

In [29]:

```
# Training loop for matrix factorization model

from typing import List
from tqdm.notebook import tqdm
from scratch.linear_algebra import dot

def fit(dataset: List[Rating],
            learning_rate: float = None) -> float:
    loss = 0.0
    for i, rating in enumerate(dataset):
        movie_vector = movie_vectors[rating.movie_id]
        user_vector = user_vectors[rating.user_id]
        predicted = dot(user_vector, movie_vector)
        error = predicted - rating.rating
        loss += error ** 2

        if learning_rate is not None:
            #     predicted = m_0 * u_0 + ... + m_k * u_k
            # So each u_j enters output with coefficent m_j
            # and each m_j enters output with coefficient u_j
            user_gradient = [error * m_j for m_j in movie_vector]
            movie_gradient = [error * u_j for u_j in user_vector]

            # Take gradient steps
            for j in range(EMBEDDING_DIM):
                user_vector[j] -= learning_rate * user_gradient[j]
                movie_vector[j] -= learning_rate * movie_gradient[j]
    return loss/len(dataset)
```

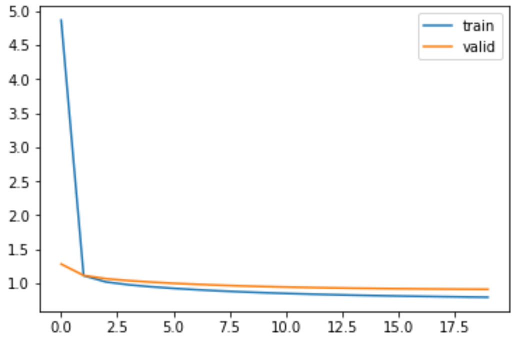
In [30]:

```
def evaluate(dataset: List[Rating]) -> float:
    loss = 0.0
    for i, rating in enumerate(dataset):
        movie_vector = movie_vectors[rating.movie_id]
        user_vector = user_vectors[rating.user_id]
        predicted = dot(user_vector, movie_vector)
        error = predicted - rating.rating
        loss += error ** 2
    return loss/len(dataset)
```

In [31]:
```
from tqdm import trange, tqdm
learning_rate = 0.05

train_history = []
validation_history = []
with trange(20) as t:
    for epoch in t:
        learning_rate *= 0.9
        train_loss = fit(train, learning_rate=learning_rate)
        valid_loss = evaluate(validation)
        t.set_description(f"train loss: {train_loss :.2f}, valid loss: {valid_loss :.2f}")
        train_history.append(train_loss)
        validation_history.append(valid_loss)

test_loss = evaluate(test)
print(f"test loss: {test_loss:.2f}")
```

```
train loss: 0.80, valid loss: 0.91: 100%|██████████████████████████████████████████| 20/20 [00:05<00:00,  3.61it/s]

test loss: 0.9067054891525651
```

In [32]:
```
import matplotlib.pyplot as plt

plt.plot(range(20), train_history, label="train")
plt.plot(range(20), validation_history, label='valid')
plt.legend()
plt.show()
```



4.8 모델 성능 비교

4.8.1 전체 평점 예측

In [33]:
```
def predict(user_id: int, movie_id) -> float:
    movie_vector = movie_vectors[movie_id]
    user_vector = user_vectors[user_id]
    return dot(user_vector, movie_vector)
```

In [34]:
```
predicted_list = []
predicted_sum = 0
total_count = len(user_ids)*len(movie_ids)
for user_id in user_ids:
    for movie_id in movie_ids:
        predicted = predict(user_id, movie_id)
        predicted_list.append(predicted)
        predicted_sum += predicted
```

4.8.2 베이스라인과 비교

In [35]:
```
avg_predicted = predicted_sum/total_count
print(f"avg rating : {avg_rating}, avg predicted : {avg_predicted}")

predicted_error = sum((predicted - avg_predicted) ** 2
                      for predicted in predicted_list) /total_count
print(f"baseline error : {baseline_error}, avg error : {predicted_error}")
```

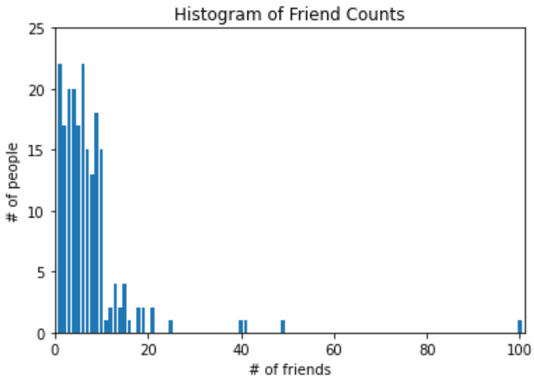
```
avg rating : 3.530842857142857, avg predicted : 2.9643352760961514
baseline error : 1.2609526646939684, avg error : 1.5160841782959116
```

4.9 영화 임베딩 주성분 분석

In [36]:

```
from scratch.working_with_data import pca, transform
original_vectors = [vector for vector in movie_vectors.values()]
components = pca(original_vectors, 2)
```

```
dv: 4478.365: 100%|████████████████████████████████████████████████████████████| 100/100 [00:00<00:00, 105.11it/s]
dv: 946.596: 100%|█████████████████████████████████████████████████████████████| 100/100 [00:00<00:00, 100.07it/s]
```



4.9.1 주성분으로 사영

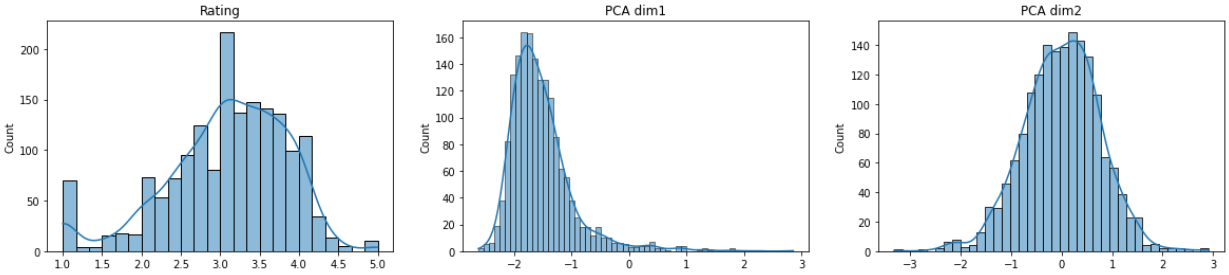
In [37]:

```
ratings_by_movie = defaultdict(list)
for rating in ratings:
    ratings_by_movie[rating.movie_id].append(rating.rating)

vectors = [
    (movie_id,
        sum(ratings_by_movie[movie_id]) / len(ratings_by_movie[movie_id]),
        movies[movie_id],
        vector)
    for movie_id, vector in zip(movie_vectors.keys(),
                                transform(original_vectors, components))
]
```

4.9.2 평점과 임베딩 분포

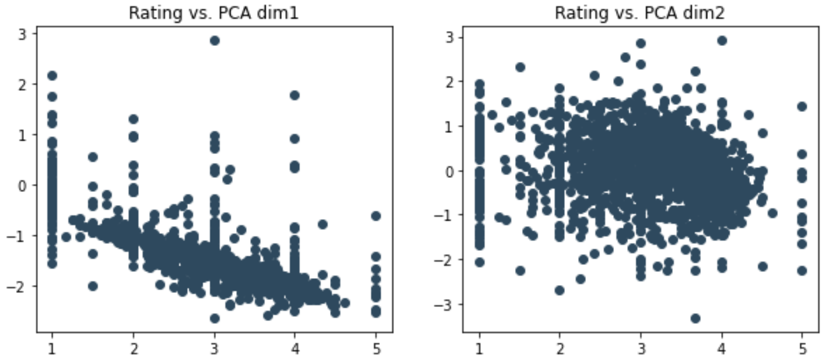
In [38]:

```
import matplotlib.pyplot as plt
import seaborn as sns
ratings = [vector[1] for vector in vectors]
pca_dim1 = [vector[-1][0] for vector in vectors]
pca_dim2 = [vector[-1][1] for vector in vectors]

plt.figure(figsize=[20,4])
plt.subplot(1,3,1)
sns.histplot(data=ratings, kde=True)
plt.title('Rating')
plt.subplot(1,3,2)
sns.histplot(data=pca_dim1, kde=True)
plt.title('PCA dim1')
plt.subplot(1,3,3)
sns.histplot(data=pca_dim2, kde=True)
plt.title('PCA dim2')
plt.show()
```



4.9.3 평점과 임베딩의 상관성

In [39]:
```
plt.figure(figsize=[10,4])
plt.subplot(1,2,1)
plt.scatter(ratings, pca_dim1, facecolor="#2E495E")
plt.title('Rating vs. PCA dim1')
plt.subplot(1,2,2)
plt.scatter(ratings, pca_dim2, facecolor="#2E495E")
plt.title('Rating vs. PCA dim2')
plt.show()
```



9장. 데이터 수집하기 (Getting Data) - 웹 스크랩핑

In [1]:

```
!pip install beautifulsoup4 requests html5lib
```

```
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.9.3-py3-none-any.whl (115 kB)
Requirement already satisfied: requests in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (2.25.1)
Collecting html5lib
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
Collecting soupsieve>1.2
  Downloading soupsieve-2.2.1-py3-none-any.whl (33 kB)
Collecting webencodings
  Using cached webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: six>=1.9 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from html5lib) (1.15.
0)
Requirement already satisfied: chardet<5,>=3.0.2 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from request
s) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from request
s) (2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from requests) (2.
10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from requ
ests) (1.26.3)
Installing collected packages: webencodings, soupsieve, html5lib, beautifulsoup4
Successfully installed beautifulsoup4-4.9.3 html5lib-1.1 soupsieve-2.2.1 webencodings-0.5.1
```

1. HTML 문서 가져오기

In [2]:

```
from bs4 import BeautifulSoup
import requests

# I put the relevant HTML file on GitHub. In order to fit
# the URL in the book I had to split it across two lines.
# Recall that whitespace-separated strings get concatenated.
url = ("https://raw.githubusercontent.com/joelgrus/data/master/getting-data.html")
html = requests.get(url).text
soup = BeautifulSoup(html, 'html5lib')
print(soup)
```

```
<!DOCTYPE html>
<html lang="en-US"><head>
    <title>Getting Data</title>
    <meta charset="utf-8"/>
</head>
<body>
    <h1>Getting Data</h1>
    <div class="explanation">
        This is an explanation.
    </div>
    <div class="comment">
        This is a comment.
    </div>
    <div class="content">
        <p id="p1">This is the first paragraph.</p>
        <p class="important">This is the second paragraph.</p>
    </div>
    <div class="signature">
        <span id="name">Joel</span>
        <span id="twitter">@joelgrus</span>
        <span id="email">joelgrus-at-gmail</span>
    </div>


</body></html>
```

1.1 첫번째 Paragraph

In [3]:

```
first_paragraph = soup.find('p')        # or just soup.p
print(first_paragraph)
assert str(soup.find('p')) == '<p id="p1">This is the first paragraph.</p>'
```

```
<p id="p1">This is the first paragraph.</p>
```

1.2 단어로 쪼개기

In [4]:

```
first_paragraph_text = soup.p.text
first_paragraph_words = soup.p.text.split()
print(first_paragraph_words)
assert first_paragraph_words == ['This', 'is', 'the', 'first', 'paragraph.']
```

```
['This', 'is', 'the', 'first', 'paragraph.']
```

1.3 딕셔너리처럼 사용하기

In [5]:

```
first_paragraph_id = soup.p['id']       # raises KeyError if no 'id'
first_paragraph_id2 = soup.p.get('id')  # returns None if no 'id'


assert first_paragraph_id == first_paragraph_id2 == 'p1'
```

1.4 여러 태그 불러오기

In [6]:

```
all_paragraphs = soup.find_all('p')  # or just soup('p')
paragraphs_with_ids = [p for p in soup('p') if p.get('id')]

assert len(all_paragraphs) == 2
assert len(paragraphs_with_ids) == 1
```

1.5 특정 클래스 태그 불러오기

In [7]:

```
important_paragraphs = soup('p', {'class' : 'important'})
important_paragraphs2 = soup('p', 'important')
important_paragraphs3 = [p for p in soup('p')
                         if 'important' in p.get('class', [])]

assert important_paragraphs == important_paragraphs2 == important_paragraphs3
assert len(important_paragraphs) == 1
```

1.6 div에 포함된 span가져오기

In [8]:

```
spans_inside_divs = [span
                     for div in soup('div')     # for each <div> on the page
                     for span in div('span')]   # find each <span> inside it


assert len(spans_inside_divs) == 3
```

2. 예시 : 의회 감시하기

In [9]:

```
from bs4 import BeautifulSoup
import requests

url = "https://www.house.gov/representatives"
text = requests.get(url).text
soup = BeautifulSoup(text, "html5lib")

all_urls = [a['href']
            for a in soup('a')
            if a.has_attr('href')]

print(len(all_urls))  # 965 for me, way too many
```

```
967
```

2.1 정규식을 이용해서 의원의 URL만 필터링

In [10]:

```
import re

# Must start with http:// or https://
# Must end with .house.gov or .house.gov/
regex = r"^https?://.*₩.house₩.gov/?$"

# Let's write some tests!
assert re.match(regex, "http://joel.house.gov")
assert re.match(regex, "https://joel.house.gov")
assert re.match(regex, "http://joel.house.gov/")
assert re.match(regex, "https://joel.house.gov/")
assert not re.match(regex, "joel.house.gov")
assert not re.match(regex, "http://joel.house.com")
assert not re.match(regex, "https://joel.house.gov/biography")
```

In [11]:

```
good_urls = [url for url in all_urls if re.match(regex, url)]

print(len(good_urls))  # still 862 for me
```

```
870
```

2.2 중복 제거

In [12]:

```
num_original_good_urls = len(good_urls)
good_urls = list(set(good_urls))

print(len(good_urls))  # only 431 for me
assert len(good_urls) < num_original_good_urls
```

```
435
```

2.3 jayapal 홈페이지에서 PR 링크 찾기

In [13]:

```
html = requests.get('https://jayapal.house.gov').text
soup = BeautifulSoup(html, 'html5lib')

# Use a set because the links might appear multiple times.
links = {a['href'] for a in soup('a') if 'press releases' in a.text.lower()}

print(links) # {'/media/press-releases'}
```

```
{'https://jayapal.house.gov/category/news/', 'https://jayapal.house.gov/category/press-releases/'}
```

2.4 모든 의원의 홈페이지에서 PR 링크 찾기

In [14]:
```
from typing import Dict, Set

press_releases: Dict[str, Set[str]] = {}

for house_url in good_urls:
    html = requests.get(house_url).text
    soup = BeautifulSoup(html, 'html5lib')
    pr_links = {a['href'] for a in soup('a') if 'press releases' in a.text.lower()}
    print(f"{house_url}: {pr_links}")
    press_releases[house_url] = pr_links
```

```
https://kilmer.house.gov: (https://kilmer.house.gov:) {'https://kilmer.house.gov/news/press-releases'}
https://crenshaw.house.gov/: (https://crenshaw.house.gov/:) {'/press-releases'}
https://arrington.house.gov: (https://arrington.house.gov:) set()
https://eshoo.house.gov/: (https://eshoo.house.gov/:) {'/media/press-releases'}
https://biggs.house.gov: (https://biggs.house.gov:) {'/media/press-releases'}
https://loudermilk.house.gov: (https://loudermilk.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://morelle.house.gov: (https://morelle.house.gov:) {'/media/press-releases'}
https://sarajacobs.house.gov: (https://sarajacobs.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://cartwright.house.gov: (https://cartwright.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=2442'}
https://comer.house.gov/: (https://comer.house.gov/:) {'/press-release'}
https://panetta.house.gov: (https://panetta.house.gov:) {'/media/press-releases'}
https://trahan.house.gov: (https://trahan.house.gov:) set()
https://budd.house.gov: (https://budd.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mccollum.house.gov: (https://mccollum.house.gov:) {'/media/press-releases'}
https://napolitano.house.gov/: (https://napolitano.house.gov/:) {'/media/press-releases'}
https://franklin.house.gov: (https://franklin.house.gov:) {'/media/press-releases'}
https://stevens.house.gov/: (https://stevens.house.gov/:) {'/media/press-releases'}
https://cline.house.gov: (https://cline.house.gov:) {'/media/press-releases'}
https://posey.house.gov/: (https://posey.house.gov/:) {'#tab-2', '/News/DocumentQuery.aspx?DocumentTypeID=1487'}
https://bacon.house.gov: (https://bacon.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://cardenas.house.gov: (https://cardenas.house.gov:) {'https://cardenas.house.gov/media-center/press-releases'}
https://banks.house.gov: (https://banks.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://chu.house.gov/: (https://chu.house.gov/:) {'/media-center/press-releases'}
https://auchincloss.house.gov: (https://auchincloss.house.gov:) {'/media/press-releases'}
https://mchenry.house.gov: (https://mchenry.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=418'}
https://barr.house.gov/: (https://barr.house.gov/:) {'/press-releases'}
https://omar.house.gov/: (https://omar.house.gov/:) {'/media/press-releases'}
https://emmer.house.gov/: (https://emmer.house.gov/:) {'/press-releases'}
https://calvert.house.gov/: (https://calvert.house.gov/:) {'/media/press-releases'}
https://bourdeaux.house.gov: (https://bourdeaux.house.gov:) {'/media/press-releases'}
https://kinzinger.house.gov/: (https://kinzinger.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2665'}
https://watsoncoleman.house.gov/: (https://watsoncoleman.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://defazio.house.gov/: (https://defazio.house.gov/:) {'/media-center/press-releases'}
https://jackson.house.gov: (https://jackson.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mikerogers.house.gov/: (https://mikerogers.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mikegarcia.house.gov/: (https://mikegarcia.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://grijalva.house.gov/: (https://grijalva.house.gov/:) {'/media/press-releases'}
https://case.house.gov/: (https://case.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27', '/news/documentsingle.asp
x?DocumentID=574'}
https://kaygranger.house.gov: (https://kaygranger.house.gov:) {'/press-releases'}
https://debbiedingell.house.gov/: (https://debbiedingell.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27', '/new
s/'}
https://hern.house.gov: (https://hern.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://adamsmith.house.gov/: (https://adamsmith.house.gov/:) {'/press-releases', '#recentposts-pressreleases'}
https://waltz.house.gov: (https://waltz.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://vantaylor.house.gov/: (https://vantaylor.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mikelevin.house.gov: (https://mikelevin.house.gov:) {'https://mikelevin.house.gov/media/press-releases'}
https://newhouse.house.gov: (https://newhouse.house.gov:) {'/media-center/press-releases'}
https://katko.house.gov/: (https://katko.house.gov/:) {'/media-center/press-releases'}
https://schiff.house.gov/: (https://schiff.house.gov/:) {'https://schiff.house.gov/news/press-releases'}
https://adriansmith.house.gov/: (https://adriansmith.house.gov/:) {'/newsroom/press-releases'}
https://tlaib.house.gov/: (https://tlaib.house.gov/:) {'/media/press-releases'}
https://mace.house.gov: (https://mace.house.gov:) {'/media/press-releases'}
https://mfume.house.gov/: (https://mfume.house.gov/:) {'/media/press-releases'}
https://lawson.house.gov: (https://lawson.house.gov:) {'https://lawson.house.gov/media/press-releases'}
https://reed.house.gov/: (https://reed.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://horsford.house.gov: (https://horsford.house.gov:) {'/media/press-releases'}
https://lesko.house.gov: (https://lesko.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://buck.house.gov/: (https://buck.house.gov/:) set()
https://fleischmann.house.gov/: (https://fleischmann.house.gov/:) {'/media/press-releases'}
https://mccaul.house.gov: (https://mccaul.house.gov:) {'/media-center/press-releases'}
https://clyburn.house.gov/: (https://clyburn.house.gov/:) {'/press-releases'}
https://cleaver.house.gov: (https://cleaver.house.gov:) {'/media-center/press-releases'}
https://norton.house.gov/: (https://norton.house.gov/:) {'/media-center/press-releases'}
https://nunes.house.gov/: (https://nunes.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2133'}
https://marymiller.house.gov: (https://marymiller.house.gov:) {'/media/subscribe-press-releases', '/media/press-releases'}
https://quigley.house.gov/: (https://quigley.house.gov/:) {'/media-center/press-releases'}
https://trentkelly.house.gov/: (https://trentkelly.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://long.house.gov/: (https://long.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://guthrie.house.gov/: (https://guthrie.house.gov/:) set()
https://pallone.house.gov: (https://pallone.house.gov:) {'/media/press-releases'}
https://kathleenrice.house.gov: (https://kathleenrice.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://fulcher.house.gov/: (https://fulcher.house.gov/:) {'/press-releases'}
https://chabot.house.gov/: (https://chabot.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2508'}
https://courtney.house.gov/: (https://courtney.house.gov/:) {'/media-center/press-releases'}
```

```
https://himes.house.gov/: (https://himes.house.gov/:) set()
https://barragan.house.gov: (https://barragan.house.gov:) {'https://barragan.house.gov/category/press-releases/'}
https://austinscott.house.gov/: (https://austinscott.house.gov/:) {'/press-releases'}
https://jasonsmith.house.gov: (https://jasonsmith.house.gov:) set()
https://buchanan.house.gov/: (https://buchanan.house.gov/:) set()
https://lamb.house.gov: (https://lamb.house.gov:) {'/media/press-releases'}
https://allen.house.gov: (https://allen.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://castor.house.gov/: (https://castor.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=821'}
https://youngkim.house.gov: (https://youngkim.house.gov:) {'/media/press-releases'}
https://halrogers.house.gov/: (https://halrogers.house.gov/:) {'/press-releases'}
https://steube.house.gov/: (https://steube.house.gov/:) {'/media/press-releases'}
https://crawford.house.gov/: (https://crawford.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2080'}
https://doyle.house.gov: (https://doyle.house.gov:) {'/media/press-releases'}
https://mcmorris.house.gov/: (https://mcmorris.house.gov/:) set()
https://pence.house.gov/: (https://pence.house.gov/:) {'/media/press-releases'}
https://bergman.house.gov: (https://bergman.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://upton.house.gov: (https://upton.house.gov:) {'/News/DocumentQuery.aspx?DocumentTypeID=1828'}
https://jayapal.house.gov: (https://jayapal.house.gov:) {'https://jayapal.house.gov/category/news/', 'https://jayapal.house.g
ov/category/press-releases/'}
https://webster.house.gov/: (https://webster.house.gov/:) {'/press-releases'}
https://troycarter.house.gov: (https://troycarter.house.gov:) {'/media/press-releases'}
https://johnrose.house.gov/: (https://johnrose.house.gov/:) {'/media/press-releases'}
https://morgangriffith.house.gov/: (https://morgangriffith.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2235'}
https://bass.house.gov/: (https://bass.house.gov/:) {'/media-center/press-releases'}
https://slotkin.house.gov/: (https://slotkin.house.gov/:) {'/media/press-releases'}
https://roy.house.gov: (https://roy.house.gov:) {'/media/press-releases'}
https://mceachin.house.gov: (https://mceachin.house.gov:) {'/media/press-releases'}
https://bonamici.house.gov: (https://bonamici.house.gov:) {'/media/press-releases'}
https://herrell.house.gov: (https://herrell.house.gov:) {'/media/press-releases-c'}
https://seanmaloney.house.gov: (https://seanmaloney.house.gov:) set()
https://rodneydavis.house.gov: (https://rodneydavis.house.gov:) {'#tab1', '/news/documentquery.aspx?DocumentTypeID=2427'}
https://kuster.house.gov: (https://kuster.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://pelosi.house.gov/: (https://pelosi.house.gov/:) {'/news/press-releases'}
https://fernandez.house.gov: (https://fernandez.house.gov:) {'/media/press-releases'}
https://hice.house.gov/: (https://hice.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://nikemawilliams.house.gov: (https://nikemawilliams.house.gov:) {'https://nikemawilliams.house.gov/media/press-release
s', '/media/press-releases'}
https://manning.house.gov: (https://manning.house.gov:) {'/media/press-releases'}
https://jordan.house.gov/: (https://jordan.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=1611'}
https://price.house.gov/: (https://price.house.gov/:) {'/newsroom/press-releases'}
https://bustos.house.gov: (https://bustos.house.gov:) {'https://bustos.house.gov/category/press-release/'}
https://newman.house.gov: (https://newman.house.gov:) {'/media/press-releases'}
https://bice.house.gov: (https://bice.house.gov:) {'/media/press-releases'}
https://vandrew.house.gov: (https://vandrew.house.gov:) {'/media/press-releases'}
https://williams.house.gov: (https://williams.house.gov:) {'/media-center/press-releases'}
https://millermeeks.house.gov/: (https://millermeeks.house.gov/:) {'/media/press-releases'}
https://miller.house.gov/: (https://miller.house.gov/:) {'/media/press-releases'}
https://reschenthaler.house.gov/: (https://reschenthaler.house.gov/:) {'/media/press-releases'}
https://garretgraves.house.gov/: (https://garretgraves.house.gov/:) {'/media-center/press-releases'}
https://dankildee.house.gov: (https://dankildee.house.gov:) {'/media/press-releases'}
https://salazar.house.gov: (https://salazar.house.gov:) {'/media/press-releases'}
https://palmer.house.gov/: (https://palmer.house.gov/:) {'/media-center/press-releases', 'https://palmer.house.gov/media-cent
er/press-releases'}
https://fletcher.house.gov: (https://fletcher.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://desaulnier.house.gov/: (https://desaulnier.house.gov/:) {'/media-center/press-releases'}
https://swalwell.house.gov: (https://swalwell.house.gov:) {'/media-center/press-releases'}
https://golden.house.gov: (https://golden.house.gov:) {'/media/press-releases'}
https://kim.house.gov/: (https://kim.house.gov/:) {'/media/press-releases'}
https://issa.house.gov: (https://issa.house.gov:) {'/media/press-releases'}
https://meng.house.gov: (https://meng.house.gov:) {'/media-center/press-releases'}
https://luria.house.gov: (https://luria.house.gov:) {'/media/press-releases'}

https://sablan.house.gov/: (https://sablan.house.gov/:) set()
https://gibbs.house.gov/: (https://gibbs.house.gov/:) {'/media-center/press-releases'}
https://mast.house.gov: (https://mast.house.gov:) {'/press-releases'}
https://moulton.house.gov/: (https://moulton.house.gov/:) set()
https://castro.house.gov: (https://castro.house.gov:) {'https://castro.house.gov/media-center/press-releases'}
https://schakowsky.house.gov: (https://schakowsky.house.gov:) {'/media/press-releases'}
https://delgado.house.gov: (https://delgado.house.gov:) {'/media/press-releases'}
https://bluntrochester.house.gov: (https://bluntrochester.house.gov:) set()
https://kustoff.house.gov: (https://kustoff.house.gov:) {'/media/press-releases'}
https://johnjoyce.house.gov/: (https://johnjoyce.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://pascrell.house.gov/: (https://pascrell.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://stanton.house.gov/: (https://stanton.house.gov/:) {'/media/press-releases'}
https://harshbarger.house.gov: (https://harshbarger.house.gov:) {'/media/press-releases'}
https://bush.house.gov: (https://bush.house.gov:) {'/media/press-releases'}
https://scalise.house.gov/: (https://scalise.house.gov/:) {'/media/press-releases'}
https://larsen.house.gov: (https://larsen.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://hinson.house.gov: (https://hinson.house.gov:) {'/media/press-releases'}
https://dean.house.gov: (https://dean.house.gov:) {'/press-releases'}
https://timryan.house.gov/: (https://timryan.house.gov/:) {'/media/press-releases'}
https://mikethompson.house.gov/: (https://mikethompson.house.gov/:) {'/newsroom/press-releases'}
https://babin.house.gov: (https://babin.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://underwood.house.gov/: (https://underwood.house.gov/:) {'/media/press-releases'}
https://desjarlais.house.gov/: (https://desjarlais.house.gov/:) set()
https://casten.house.gov: (https://casten.house.gov:) {'/media/press-releases'}
https://ferguson.house.gov: (https://ferguson.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
```

```
https://huffman.house.gov: (https://huffman.house.gov:) set()
https://matsui.house.gov: (https://matsui.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://nadler.house.gov: (https://nadler.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=1753'}
https://algreen.house.gov: (https://algreen.house.gov:) {'/press-releases'}
https://fitzpatrick.house.gov/: (https://fitzpatrick.house.gov/:) {'/press-releases'}
https://wilson.house.gov/: (https://wilson.house.gov/:) set()
https://feenstra.house.gov: (https://feenstra.house.gov:) {'/media/press-releases'}
https://sarbanes.house.gov/: (https://sarbanes.house.gov/:) {'/media-center/press-releases'}
https://clayhiggins.house.gov: (https://clayhiggins.house.gov:) {'/media/press-releases'}
https://vela.house.gov: (https://vela.house.gov:) {'/press-release'}
https://wittman.house.gov/: (https://wittman.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2670'}
https://greene.house.gov: (https://greene.house.gov:) {'/media/press-releases'}
https://delbene.house.gov: (https://delbene.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://ocasio-cortez.house.gov/: (https://ocasio-cortez.house.gov/:) {'/media/press-releases'}
https://luetkemeyer.house.gov/: (https://luetkemeyer.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2270'}
https://curtis.house.gov/: (https://curtis.house.gov/:) {'https://curtis.house.gov/category/press-releases/'}
https://lucas.house.gov: (https://lucas.house.gov:) {'/news/press-releases'}
https://evans.house.gov/: (https://evans.house.gov/:) {'/media-center/press-releases'}
https://garamendi.house.gov/: (https://garamendi.house.gov/:) {'/media/press-releases'}
https://lynch.house.gov/: (https://lynch.house.gov/:) {'/press-releases'}
https://fischbach.house.gov: (https://fischbach.house.gov:) {'/media/press-releases'}
https://frankel.house.gov: (https://frankel.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://perlmutter.house.gov/: (https://perlmutter.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://lofgren.house.gov/: (https://lofgren.house.gov/:) {'/media/press-releases'}
https://murphy.house.gov: (https://murphy.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://westerman.house.gov/: (https://westerman.house.gov/:) set()
https://keating.house.gov/: (https://keating.house.gov/:) set()
https://sherrill.house.gov/: (https://sherrill.house.gov/:) {'/media/press-releases'}
https://gimenez.house.gov: (https://gimenez.house.gov:) {'/media/press-releases'}
https://huizenga.house.gov/: (https://huizenga.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2041'}
https://bera.house.gov: (https://bera.house.gov:) {'/media-center/press-releases'}
https://doggett.house.gov: (https://doggett.house.gov:) set()
https://sires.house.gov: (https://sires.house.gov:) {'/media-center/press-releases'}
https://womack.house.gov/: (https://womack.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2067'}
https://armstrong.house.gov: (https://armstrong.house.gov:) {'/media/press-releases'}
https://buddycarter.house.gov/: (https://buddycarter.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://cole.house.gov: (https://cole.house.gov:) {'/media-center/press-releases'}
https://clyde.house.gov: (https://clyde.house.gov:) {'/media/press-releases'}
https://adams.house.gov: (https://adams.house.gov:) {'/media-center/press-releases'}
https://jeffduncan.house.gov/: (https://jeffduncan.house.gov/:) {'/media/press-releases'}
https://norman.house.gov: (https://norman.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://norcross.house.gov: (https://norcross.house.gov:) set()
https://latta.house.gov/: (https://latta.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=1456'}
https://weber.house.gov: (https://weber.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://allred.house.gov/: (https://allred.house.gov/:) {'/media/press-releases'}
https://chuygarcia.house.gov/: (https://chuygarcia.house.gov/:) {'/media/press-releases'}
https://vanduyne.house.gov: (https://vanduyne.house.gov:) {'/media/press-releases'}
https://balderson.house.gov: (https://balderson.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://jhb.house.gov/: (https://jhb.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2113'}
https://lieu.house.gov/: (https://lieu.house.gov/:) {'/media-center/press-releases'}
https://wagner.house.gov: (https://wagner.house.gov:) {'/media-center/press-releases'}
https://joewilson.house.gov/: (https://joewilson.house.gov/:) {'/media-center/press-releases'}
https://brooks.house.gov/: (https://brooks.house.gov/:) set()
https://gregmurphy.house.gov: (https://gregmurphy.house.gov:) {'/media/press-releases'}
https://dustyjohnson.house.gov/: (https://dustyjohnson.house.gov/:) {'/media/press-releases'}
https://tonko.house.gov/: (https://tonko.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mcclintock.house.gov/: (https://mcclintock.house.gov/:) {'/newsroom/press-releases'}
https://sherman.house.gov: (https://sherman.house.gov:) {'/media-center', '/media-center/press-releases'}
https://ritchietorres.house.gov: (https://ritchietorres.house.gov:) {'/media/press-releases'}
https://grothman.house.gov: (https://grothman.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://cohen.house.gov/: (https://cohen.house.gov/:) {'/media-center/press-releases'}
https://ruiz.house.gov: (https://ruiz.house.gov:) {'/media-center/press-releases'}
https://sessions.house.gov: (https://sessions.house.gov:) {'/media/press-releases'}
https://davidscott.house.gov/: (https://davidscott.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=377'}
https://porter.house.gov/: (https://porter.house.gov/:) set()
https://simpson.house.gov: (https://simpson.house.gov:) {'/News/DocumentQuery.aspx?DocumentTypeID=1515'}
https://obernolte.house.gov: (https://obernolte.house.gov:) {'/media/press-releases'}
https://estes.house.gov/: (https://estes.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://bobbyscott.house.gov: (https://bobbyscott.house.gov:) {'/media-center/press-releases'}
https://juliabrownley.house.gov: (https://juliabrownley.house.gov:) {'https://juliabrownley.house.gov/category/press-relea
ses/'}
https://gonzalez.house.gov: (https://gonzalez.house.gov:) {'/media/press-releases'}
https://danbishop.house.gov: (https://danbishop.house.gov:) {'/media/press-releases'}
https://bowman.house.gov: (https://bowman.house.gov:) {'/press-releases'}
https://scottpeters.house.gov: (https://scottpeters.house.gov:) {'/media-center/press-releases'}
https://mckinley.house.gov/: (https://mckinley.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mooney.house.gov/: (https://mooney.house.gov/:) {'/media-center/press-releases'}
https://maloney.house.gov/: (https://maloney.house.gov/:) {'/news/press-releases'}
https://bennietthompson.house.gov/: (https://bennietthompson.house.gov/:) {'/media/press-releases'}
https://teddeutch.house.gov/: (https://teddeutch.house.gov/:) set()
https://lindasanchez.house.gov/: (https://lindasanchez.house.gov/:) {'/media-center/press-releases'}
https://cooper.house.gov/: (https://cooper.house.gov/:) {'/media-center/press-releases'}
https://sylviagarcia.house.gov/: (https://sylviagarcia.house.gov/:) {'/media/press-releases'}
https://tiffany.house.gov/: (https://tiffany.house.gov/:) {'/media/press-releases'}
https://bost.house.gov/: (https://bost.house.gov/:) {'/media-center/press-releases'}
https://lamalfa.house.gov: (https://lamalfa.house.gov:) {'/media-center/press-releases'}
https://larson.house.gov/: (https://larson.house.gov/:) {'/media-center/press-releases'}
```

```
https://rosendale.house.gov: (https://rosendale.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://pingree.house.gov/: (https://pingree.house.gov/:) set()
https://beyer.house.gov: (https://beyer.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://steel.house.gov: (https://steel.house.gov:) {'/media/press-releases'}
https://schneider.house.gov: (https://schneider.house.gov:) {'/media/press-releases'}
https://schrader.house.gov/: (https://schrader.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2382'}
https://butterfield.house.gov: (https://butterfield.house.gov:) {'/media-center/press-releases'}
https://stewart.house.gov: (https://stewart.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://mcclain.house.gov: (https://mcclain.house.gov:) {'/media/press-releases'}
https://davids.house.gov/: (https://davids.house.gov/:) {'/media/press-releases'}
https://costa.house.gov/: (https://costa.house.gov/:) {'/media-center/press-releases'}
https://suozzi.house.gov: (https://suozzi.house.gov:) {'/media/press-releases'}
https://crow.house.gov/: (https://crow.house.gov/:) {'/media/press-releases'}
https://moolenaar.house.gov/: (https://moolenaar.house.gov/:) {'/media-center/press-releases'}
https://clarke.house.gov/: (https://clarke.house.gov/:) {'https://clarke.house.gov/category/press-releases/'}
https://houlahan.house.gov/: (https://houlahan.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://bucshon.house.gov/: (https://bucshon.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://fallon.house.gov: (https://fallon.house.gov:) {'/media/press-releases'}
https://neal.house.gov/: (https://neal.house.gov/:) {'/press-releases'}
https://ross.house.gov: (https://ross.house.gov:) {'/media/press-releases'}
https://chrissmith.house.gov/: (https://chrissmith.house.gov/:) set()
https://lamborn.house.gov/: (https://lamborn.house.gov/:) {'/media/press-releases'}
https://kevinmccarthy.house.gov/: (https://kevinmccarthy.house.gov/:) {'/media-center/press-releases'}
https://davis.house.gov: (https://davis.house.gov:) set()
https://axne.house.gov/: (https://axne.house.gov/:) {'/media/press-releases'}
https://pfluger.house.gov: (https://pfluger.house.gov:) {'/media/press-releases'}
https://lowenthal.house.gov: (https://lowenthal.house.gov:) {'/media/press-releases'}
https://higgins.house.gov: (https://higgins.house.gov:) {'/press-releases/archived-press-releases-2008', '/media-center/pr
ess-releases'}

https://keller.house.gov: (https://keller.house.gov:) {'/media/press-releases'}
https://carter.house.gov/: (https://carter.house.gov/:) {'/news/press-releases'}
https://langevin.house.gov: (https://langevin.house.gov:) {'/press-releases'}
https://scanlon.house.gov/: (https://scanlon.house.gov/:) set()
https://escobar.house.gov: (https://escobar.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://schweikert.house.gov/: (https://schweikert.house.gov/:) {'/media-center/press-releases'}
https://demings.house.gov: (https://demings.house.gov:) {'/media/press-releases'}
https://takano.house.gov: (https://takano.house.gov:) {'https://takano.house.gov/newsroom/press-releases'}
https://zeldin.house.gov/: (https://zeldin.house.gov/:) {'/media-center/press-releases'}
https://bilirakis.house.gov/: (https://bilirakis.house.gov/:) {'/media/press-releases'}
https://rutherford.house.gov: (https://rutherford.house.gov:) {'/media/press-releases'}
https://meijer.house.gov: (https://meijer.house.gov:) {'/media/press-releases'}
https://donyoung.house.gov/: (https://donyoung.house.gov/:) {'/News/'}
https://gonzales.house.gov: (https://gonzales.house.gov:) {'/media/press-releases'}
https://gooden.house.gov: (https://gooden.house.gov:) {'/media/press-releases'}
https://thompson.house.gov: (https://thompson.house.gov:) {'/media-center/press-releases'}
https://smucker.house.gov/: (https://smucker.house.gov/:) {'/media/press-releases'}
https://billjohnson.house.gov/: (https://billjohnson.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2085', '/news/doc
umentquery.aspx?DocumentTypeID=2085'}
https://strickland.house.gov: (https://strickland.house.gov:) {'/media/press-releases'}
https://harder.house.gov/: (https://harder.house.gov/:) {'/media/press-releases'}
https://anthonygonzalez.house.gov: (https://anthonygonzalez.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://cloud.house.gov: (https://cloud.house.gov:) set()
https://boyle.house.gov/: (https://boyle.house.gov/:) {'/media-center/press-releases'}
https://espaillat.house.gov: (https://espaillat.house.gov:) {'/media/press-releases'}
https://gwenmoore.house.gov: (https://gwenmoore.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://susielee.house.gov: (https://susielee.house.gov:) {'/media/press-releases'}
https://fitzgerald.house.gov: (https://fitzgerald.house.gov:) {'/media/press-releases'}
https://fortenberry.house.gov/: (https://fortenberry.house.gov/:) {'/news/press-releases'}
https://mikejohnson.house.gov: (https://mikejohnson.house.gov:) {'/news/documentsingle.aspx?DocumentID=888', '/news/documentq
uery.aspx?DocumentTypeID=1951', '/news/documentsingle.aspx?DocumentID=875', '/news/documentquery.aspx?DocumentTypeID=27', '/n
ews/documentsingle.aspx?DocumentID=884', '/news/documentsingle.aspx?DocumentID=889', '/news/documentsingle.aspx?DocumentID=87
4'}
https://gaetz.house.gov: (https://gaetz.house.gov:) {'/media/press-releases'}
https://owens.house.gov: (https://owens.house.gov:) {'/media/press-releases'}
https://mcnerney.house.gov/: (https://mcnerney.house.gov/:) {'/media-center/press-releases'}
https://steil.house.gov: (https://steil.house.gov:) {'/media/press-releases'}
https://hartzler.house.gov/: (https://hartzler.house.gov/:) {'/media-center/press-releases'}
https://perry.house.gov/: (https://perry.house.gov/:) set()
https://rice.house.gov: (https://rice.house.gov:) {'/press-releases'}
https://andylevin.house.gov/: (https://andylevin.house.gov/:) {'/media/press-releases'}
https://cicilline.house.gov/: (https://cicilline.house.gov/:) {'/press-releases'}
https://gonzalez-colon.house.gov: (https://gonzalez-colon.house.gov:) {'/media/press-releases'}
https://anthonybrown.house.gov: (https://anthonybrown.house.gov:) {'/news/documentsingle.aspx?DocumentID=1377', '/news/docume
ntquery.aspx?DocumentTypeID=27', '/news/documentsingle.aspx?DocumentID=1376'}
https://rubengallego.house.gov/: (https://rubengallego.house.gov/:) {'/media-center/press-releases'}
https://malinowski.house.gov/: (https://malinowski.house.gov/:) {'/media/press-releases'}
https://wexton.house.gov/: (https://wexton.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://amodei.house.gov: (https://amodei.house.gov:) set()
https://malliotakis.house.gov: (https://malliotakis.house.gov:) {'/media/press-releases'}
https://kelly.house.gov: (https://kelly.house.gov:) {'/press-releases'}
https://walorski.house.gov: (https://walorski.house.gov:) set()
https://gomez.house.gov/: (https://gomez.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://blakemoore.house.gov: (https://blakemoore.house.gov:) {'/media/press-releases'}
https://carbajal.house.gov: (https://carbajal.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://spanberger.house.gov: (https://spanberger.house.gov:) {'/news/documentsingle.aspx?DocumentID=3852', '/news/documentqu
ery.aspx?DocumentTypeID=27', '/news/documentsingle.aspx?DocumentID=3851', '/news/documentsingle.aspx?DocumentID=3849', '/new
s/documentsingle.aspx?DocumentID=3850', '/news/documentsingle.aspx?DocumentID=3839', '/news/documentsingle.aspx?DocumentID=38
48'}
https://timmons.house.gov/: (https://timmons.house.gov/:) {'/media/press-releases'}
https://walberg.house.gov/: (https://walberg.house.gov/:) {'/media/press-releases'}
https://carson.house.gov/: (https://carson.house.gov/:) {'/newsroom/press-releases'}
https://beatty.house.gov: (https://beatty.house.gov:) {'/media-center/press-releases'}
https://burgess.house.gov/: (https://burgess.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=75'}
https://baird.house.gov/: (https://baird.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://waters.house.gov: (https://waters.house.gov:) {'/media-center/press-releases'}
https://mcgovern.house.gov/: (https://mcgovern.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2472'}
https://lahood.house.gov/: (https://lahood.house.gov/:) set()
https://khanna.house.gov: (https://khanna.house.gov:) {'/media/press-releases'}
https://letlow.house.gov: (https://letlow.house.gov:) {'/media/press-releases'}
https://stauber.house.gov: (https://stauber.house.gov:) {'/media/press-releases'}
https://palazzo.house.gov/: (https://palazzo.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=2519'}
https://robinkelly.house.gov/: (https://robinkelly.house.gov/:) {'/media-center/press-releases'}
https://kevinbrady.house.gov/: (https://kevinbrady.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2657'}
https://ohalleran.house.gov: (https://ohalleran.house.gov:) {'/media/press-releases'}
https://foxx.house.gov/: (https://foxx.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=2367'}
https://nehls.house.gov: (https://nehls.house.gov:) {'/media/press-releases'}
https://bishop.house.gov/: (https://bishop.house.gov/:) {'/media-center/press-releases'}
https://krishnamoorthi.house.gov: (https://krishnamoorthi.house.gov:) {'/media/press-releases'}
https://turner.house.gov/: (https://turner.house.gov/:) {'/frontpage?qt-home_page_tabs=0#qt-home_page_tabs', '/media-center/p
ress-releases'}
https://cuellar.house.gov/: (https://cuellar.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=1232'}
https://plaskett.house.gov/: (https://plaskett.house.gov/:) {'/news/documentquery.aspx?documenttypeid=27'}
https://pocan.house.gov: (https://pocan.house.gov:) {'/media-center/press-releases'}
https://gosar.house.gov/: (https://gosar.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://ruppersberger.house.gov: (https://ruppersberger.house.gov:) {'/news-room/press-releases'}
https://carl.house.gov: (https://carl.house.gov:) {'/media/press-releases'}
https://pressley.house.gov: (https://pressley.house.gov:) {'/media/press-releases'}
https://degette.house.gov: (https://degette.house.gov:) {'/newsroom/press-releases'}
https://valadao.house.gov: (https://valadao.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://rush.house.gov: (https://rush.house.gov:) {'/media-center/press-releases'}
https://aguilar.house.gov/: (https://aguilar.house.gov/:) {'/media-center/press-releases'}
https://yarmuth.house.gov/: (https://yarmuth.house.gov/:) set()
https://dunn.house.gov: (https://dunn.house.gov:) {'/press-releases'}
https://wassermanschultz.house.gov/: (https://wassermanschultz.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://tenney.house.gov/: (https://tenney.house.gov/:) {'/media/press-releases'}
https://graves.house.gov/: (https://graves.house.gov/:) {'/media-center?media_type=congress_press_release'}
https://kind.house.gov/: (https://kind.house.gov/:) {'/media-center/press-releases'}
https://delauro.house.gov/: (https://delauro.house.gov/:) {'/media-center/press-releases'}
https://cawthorn.house.gov: (https://cawthorn.house.gov:) {'/media/press-releases'}
https://harris.house.gov/: (https://harris.house.gov/:) {'/media/press-releases'}
https://davidson.house.gov/: (https://davidson.house.gov/:) {'/press-releases'}
https://cheney.house.gov: (https://cheney.house.gov:) {'https://cheney.house.gov/category/press_release/'}
https://radewagen.house.gov/: (https://radewagen.house.gov/:) {'/media-center/press-releases'}
https://payne.house.gov: (https://payne.house.gov:) {'/media/press-releases'}
https://blumenauer.house.gov/: (https://blumenauer.house.gov/:) {'/media-center/press-releases'}
https://raskin.house.gov: (https://raskin.house.gov:) {'/press-releases'}
https://laturner.house.gov: (https://laturner.house.gov:) {'/media/press-releases'}
https://wild.house.gov: (https://wild.house.gov:) {'/media/press-releases'}
https://hudson.house.gov: (https://hudson.house.gov:) {'/frontpage?qt-keep_up_to_date_with_me=0#qt-keep_up_to_date_with_me',
 '/media/press-releases'}
https://mann.house.gov: (https://mann.house.gov:) {'/media/press-releases'}
https://meeks.house.gov: (https://meeks.house.gov:) {'/media/press-releases'}
https://hollingsworth.house.gov: (https://hollingsworth.house.gov:) set()
https://mariodiazbalart.house.gov/: (https://mariodiazbalart.house.gov/:) {'/media-center/press-releases'}
https://sewell.house.gov/: (https://sewell.house.gov/:) {'/frontpage?qt-home_page_tabs=0#qt-home_page_tabs', '/media-center/p
ress-releases'}
https://stefanik.house.gov/: (https://stefanik.house.gov/:) {'/media-center/press-releases'}
https://kahele.house.gov: (https://kahele.house.gov:) {'/media/press-releases'}
https://good.house.gov: (https://good.house.gov:) {'/media/press-releases'}
https://boebert.house.gov: (https://boebert.house.gov:) {'/media/press-releases'}
https://crist.house.gov: (https://crist.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://connolly.house.gov/: (https://connolly.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=1952'}
https://donalds.house.gov: (https://donalds.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://hankjohnson.house.gov/: (https://hankjohnson.house.gov/:) {'/media-center/press-releases'}
https://aderholt.house.gov/: (https://aderholt.house.gov/:) {'/media-center/press-releases'}
https://mrvan.house.gov: (https://mrvan.house.gov:) {'/media/press-releases'}
https://schrier.house.gov: (https://schrier.house.gov:) {'/media/press-releases'}
https://roybal-allard.house.gov: (https://roybal-allard.house.gov:) {'#tab2', '/news/documentquery.aspx?DocumentTypeID=1465'}
https://jacksonlee.house.gov/: (https://jacksonlee.house.gov/:) {'/media-center/press-releases'}
https://trone.house.gov: (https://trone.house.gov:) {'https://trone.house.gov/category/congress_press_release/'}
https://jeffries.house.gov: (https://jeffries.house.gov:) {'https://jeffries.house.gov/category/press-release/'}
https://foster.house.gov: (https://foster.house.gov:) {'/media/press-releases'}
https://hagedorn.house.gov/: (https://hagedorn.house.gov/:) {'/media/press-releases'}
https://ebjohnson.house.gov: (https://ebjohnson.house.gov:) {'/media-center/press-releases'}

https://massie.house.gov: (https://massie.house.gov:) set()
https://soto.house.gov: (https://soto.house.gov:) {'/media/press-releases'}
https://mullin.house.gov: (https://mullin.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://spartz.house.gov: (https://spartz.house.gov:) {'/media/press-releases'}
https://kaptur.house.gov/: (https://kaptur.house.gov/:) {'/media-center/press-releases'}
https://kirkpatrick.house.gov/: (https://kirkpatrick.house.gov/:) set()
https://mcbath.house.gov: (https://mcbath.house.gov:) {'/press-releases'}
https://cammack.house.gov: (https://cammack.house.gov:) {'/media/press-releases'}
https://gallagher.house.gov: (https://gallagher.house.gov:) {'/media/press-releases'}
```

```
https://hill.house.gov/: (https://hill.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://jacobs.house.gov: (https://jacobs.house.gov:) {'/media/press-releases'}
https://hayes.house.gov: (https://hayes.house.gov:) {'/media/press-releases'}
https://guest.house.gov: (https://guest.house.gov:) {'/media/press-releases'}
https://joyce.house.gov: (https://joyce.house.gov:) {'/press-releases'}
https://bentz.house.gov: (https://bentz.house.gov:) {'/media/press-releases'}
https://lawrence.house.gov/: (https://lawrence.house.gov/:) {'/media-center/press-releases'}
https://welch.house.gov/: (https://welch.house.gov/:) {'/media-center/press-releases'}
https://pappas.house.gov: (https://pappas.house.gov:) {'/media/press-releases'}
https://vargas.house.gov: (https://vargas.house.gov:) {'/media-center/press-releases'}
https://gohmert.house.gov/: (https://gohmert.house.gov/:) {'/News/DocumentQuery.aspx?DocumentTypeID=1954'}
https://burchett.house.gov: (https://burchett.house.gov:) {'/media/press-releases'}
https://titus.house.gov/: (https://titus.house.gov/:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://phillips.house.gov/: (https://phillips.house.gov/:) {'/media/press-releases'}
https://velazquez.house.gov: (https://velazquez.house.gov:) {'/media-center/press-releases'}
https://torres.house.gov/: (https://torres.house.gov/:) {'/media-center/press-releases'}
https://wenstrup.house.gov: (https://wenstrup.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=2491'}
https://neguse.house.gov/: (https://neguse.house.gov/:) {'https://neguse.house.gov/media/press-releases'}
https://speier.house.gov/: (https://speier.house.gov/:) {'/press-releases'}
https://barrymoore.house.gov: (https://barrymoore.house.gov:) {'/media/press-releases'}
https://gottheimer.house.gov: (https://gottheimer.house.gov:) {'/news/documentquery.aspx?DocumentTypeID=27'}
https://meuser.house.gov: (https://meuser.house.gov:) {'/media/press-releases'}
https://markgreen.house.gov/: (https://markgreen.house.gov/:) {'/press-releases'}
https://sannicolas.house.gov: (https://sannicolas.house.gov:) set()
https://garbarino.house.gov: (https://garbarino.house.gov:) {'/media/press-releases'}
https://correa.house.gov: (https://correa.house.gov:) {'/press/table'}
https://rouzer.house.gov/: (https://rouzer.house.gov/:) {'/press-releases'}
https://craig.house.gov: (https://craig.house.gov:) {'/media/press-releases'}
https://jones.house.gov: (https://jones.house.gov:) {'/media/press-releases'}
https://veasey.house.gov: (https://veasey.house.gov:) {'/media-center/press-releases'}
https://hoyer.house.gov/: (https://hoyer.house.gov/:) set()
https://lee.house.gov/: (https://lee.house.gov/:) set()
```

2.5 Paragraph에 keyword가 존재하는지 확인

In [15]:
```
def paragraph_mentions(text: str, keyword: str) -> bool:
    """
    Returns True if a <p> inside the text mentions {keyword}
    """
    soup = BeautifulSoup(text, 'html5lib')
    paragraphs = [p.get_text() for p in soup('p')]

    return any(keyword.lower() in paragraph.lower()
               for paragraph in paragraphs)
```

In [16]:
```
text = """<body><h1>Facebook</h1><p>Twitter</p>"""
assert paragraph_mentions(text, "twitter")       # is inside a <p>
assert not paragraph_mentions(text, "facebook")  # not inside a <p>
```

2.6 어떤 의원의 보도자료에 data를 언급했는지 찾기

In [17]:
```
for house_url, pr_links in press_releases.items():
    for pr_link in pr_links:
        url = f"{house_url}/{pr_link}"
        text = requests.get(url).text

        if paragraph_mentions(text, 'data'):
            print(f"{house_url}")
            break  # done with this house_url
```

```
https://panetta.house.gov (https://panetta.house.gov)
https://banks.house.gov (https://banks.house.gov)
https://reed.house.gov/ (https://reed.house.gov/)
https://gibbs.house.gov/ (https://gibbs.house.gov/)
https://johnjoyce.house.gov/ (https://johnjoyce.house.gov/)
https://bush.house.gov (https://bush.house.gov)
https://allred.house.gov/ (https://allred.house.gov/)
https://lieu.house.gov/ (https://lieu.house.gov/)
https://davidscott.house.gov/ (https://davidscott.house.gov/)
https://mcclain.house.gov (https://mcclain.house.gov)
https://higgins.house.gov (https://higgins.house.gov)
https://donyoung.house.gov/ (https://donyoung.house.gov/)
https://anthonygonzalez.house.gov (https://anthonygonzalez.house.gov)
https://gwenmoore.house.gov (https://gwenmoore.house.gov)
https://kelly.house.gov (https://kelly.house.gov)
https://gomez.house.gov/ (https://gomez.house.gov/)
https://ohalleran.house.gov (https://ohalleran.house.gov)
https://boebert.house.gov (https://boebert.house.gov)
```

3. GitHub API 사용하기

In [18]:
```
!pip install python-dateutil
```

```
Requirement already satisfied: python-dateutil in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (2.8.1)
Requirement already satisfied: six>=1.5 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from python-dateutil)
(1.15.0)
```

3.1 JSON 객체 파싱

In [19]:
```
import requests, json

github_user = "joelgrus"
endpoint = f"https://api.github.com/users/{github_user}/repos"

repos = json.loads(requests.get(endpoint).text)
```

3.2 월별/요일별 저장소 생성 통계

In [20]:
```
from collections import Counter
from dateutil.parser import parse

dates = [parse(repo["created_at"]) for repo in repos]
month_counts = Counter(date.month for date in dates)
weekday_counts = Counter(date.weekday() for date in dates)
```

3.3 가장 최근에 만들어진 저장소 5개에 사용된 언어

In [21]:
```
last_5_repositories = sorted(repos,
                             key=lambda r: r["pushed_at"],
                             reverse=True)[:5]

last_5_languages = [repo["language"]
                    for repo in last_5_repositories]
```

In [22]:
```
print(last_5_repositories)
```

```
[{'id': 26382146, 'node_id': 'MDEwOlJlcG9zaXRvcnkyNjM4MjE0Ng==', 'name': 'data-science-from-scratch', 'full_name': 'joelgr
us/data-science-from-scratch', 'private': False, 'owner': {'login': 'joelgrus', 'id': 1308313, 'node_id': 'MDQ6VXNlcjEzMDg
zMTM=', 'avatar_url': 'https://avatars.githubusercontent.com/u/1308313?v=4', 'gravatar_id': '', 'url': 'https://api.githu
b.com/users/joelgrus', 'html_url': 'https://github.com/joelgrus', 'followers_url': 'https://api.github.com/users/joelgrus/
followers', 'following_url': 'https://api.github.com/users/joelgrus/following{/other_user}', 'gists_url': 'https://api.git
hub.com/users/joelgrus/gists{/gist_id}', 'starred_url': 'https://api.github.com/users/joelgrus/starred{/owner}{/repo}', 's
ubscriptions_url': 'https://api.github.com/users/joelgrus/subscriptions', 'organizations_url': 'https://api.github.com/use
rs/joelgrus/orgs', 'repos_url': 'https://api.github.com/users/joelgrus/repos', 'events_url': 'https://api.github.com/user
s/joelgrus/events{/privacy}', 'received_events_url': 'https://api.github.com/users/joelgrus/received_events', 'type': 'Use
r', 'site_admin': False}, 'html_url': 'https://github.com/joelgrus/data-science-from-scratch', 'description': 'code for Da
ta Science From Scratch book', 'fork': False, 'url': 'https://api.github.com/repos/joelgrus/data-science-from-scratch', 'f
orks_url': 'https://api.github.com/repos/joelgrus/data-science-from-scratch/forks', 'keys_url': 'https://api.github.com/re
pos/joelgrus/data-science-from-scratch/keys{/key_id}', 'collaborators_url': 'https://api.github.com/repos/joelgrus/data-sc
ience-from-scratch/collaborators{/collaborator}', 'teams_url': 'https://api.github.com/repos/joelgrus/data-science-from-sc
ratch/teams', 'hooks_url': 'https://api.github.com/repos/joelgrus/data-science-from-scratch/hooks', 'issue_events_url': 'h
ttps://api.github.com/repos/joelgrus/data-science-from-scratch/issues/events{/number}', 'events_url': 'https://api.github.
com/repos/joelgrus/data-science-from-scratch/events', 'assignees_url': 'https://api.github.com/repos/joelgrus/data-science
-from-scratch/assignees{/user}', 'branches_url': 'https://api.github.com/repos/joelgrus/data-science-from-scratch/branches
{/branch}', 'tags_url': 'https://api.github.com/repos/joelgrus/data-science-from-scratch/tags', 'blobs_url': 'https://api.
```

In [23]:
```
print(last_5_languages)
```

```
['Python', 'JavaScript', 'Python', 'Python', 'Python']
```

4. 트위터 API 사용하기

In [24]:
```
!pip install twython
```

```
Collecting twython
  Using cached twython-3.8.2-py3-none-any.whl (33 kB)
Collecting requests-oauthlib>=0.4.0
  Using cached requests_oauthlib-1.3.0-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: requests>=2.1.0 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from twython)
(2.25.1)
Requirement already satisfied: chardet<5,>=3.0.2 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from requests
>=2.1.0->twython) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from request
s>=2.1.0->twython) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from requ
ests>=2.1.0->twython) (1.26.3)
Requirement already satisfied: idna<3,>=2.5 in c:₩users₩fermi_2₩.conda₩envs₩data_mining₩lib₩site-packages (from requests>=2.
1.0->twython) (2.10)
Collecting oauthlib>=3.0.0
  Using cached oauthlib-3.1.0-py2.py3-none-any.whl (147 kB)
Installing collected packages: oauthlib, requests-oauthlib, twython
Successfully installed oauthlib-3.1.0 requests-oauthlib-1.3.0 twython-3.8.2
```

4.1 API Key와 Secret Key

In [25]:
```
import os

# Feel free to plug your key and secret in directly
CONSUMER_KEY = os.environ.get("TWITTER_CONSUMER_KEY")
CONSUMER_SECRET = os.environ.get("TWITTER_CONSUMER_SECRET")
```

4.2 클라이언트 인스턴스 만들기

In [26]:
```
import webbrowser
from twython import Twython

# Get a temporary client to retrieve an authentication url
temp_client = Twython(CONSUMER_KEY, CONSUMER_SECRET)
temp_creds = temp_client.get_authentication_tokens()
url = temp_creds['auth_url']

# Now visit that URL to authorize the application and get a PIN
print(f"go visit {url} and get the PIN code and paste it below")
webbrowser.open(url)
PIN_CODE = input("please enter the PIN code: ")

# Now we use that PIN_CODE to get the actual tokens
auth_client = Twython(CONSUMER_KEY,
                      CONSUMER_SECRET,
                      temp_creds['oauth_token'],
                      temp_creds['oauth_token_secret'])
final_step = auth_client.get_authorized_tokens(PIN_CODE)
ACCESS_TOKEN = final_step['oauth_token']
ACCESS_TOKEN_SECRET = final_step['oauth_token_secret']

# And get a new Twython instance using them.
twitter = Twython(CONSUMER_KEY, CONSUMER_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
```

```
go visit https://api.twitter.com/oauth/authenticate?oauth_token=387PgAAAAAABNC5QAAABecK_9Do (https://api.twitter.com/oauth/au
thenticate?oauth_token=387PgAAAAAABNC5QAAABecK_9Do) and get the PIN code and paste it below
please enter the PIN code: 6600070
```

4.3 몇몇 트윗 받기

In [27]:

```
for status in twitter.search(q='"data science"')["statuses"]:
    user = status["user"]["screen_name"]
    text = status["text"]
    print(f"{user}: {text}₩n")
```

```
Topschoolworkh1: Term papers and classes help

Math
Accounting
Microeconomics
Business
CALC,,
Political science
Health
Thesis,,
Law… https://t.co/1dkXfHZjuv (https://t.co/1dkXfHZjuv)

Topschoolworkh1: RT @Topschoolworkh1: Term papers and classes help

Math
Accounting
Microeconomics
Business
CALC,,
Political science
Health
Thesis
Sociology…

Christo48746457: RT @freeCodeCamp: If you want to practice your machine learning skills, try building an end-to-end ML projec
t.

It's especially fun with a…

BotForEquality: RT @OsoroSatComs: Kindly dm if you’re/know any #lady who can tutor data science/machine learning in Nairobi
 sometimes in late June/ Early J…

OsoroSatComs: Kindly dm if you’re/know any #lady who can tutor data science/machine learning in Nairobi sometimes in late Ju
ne/ E… https://t.co/3dHC0QEtnj (https://t.co/3dHC0QEtnj)

Deep__AI: Level up your data science vocabulary: Beta Distribution https://t.co/LGNDdWdZRv (https://t.co/LGNDdWdZRv) #Probabi
lity #BetaDistribution

LatinoLdnOnt: Data Science Is a High-Paying, Fast-Growing Field—Here’s What You Need to Know About Entering It @TheMuse http
s://t.co/qkDdESXsno (https://t.co/qkDdESXsno)

JStingone: RT @askdrstats: Data Collection &amp; Integration to Enhance Public Health: Making Sense of a Patchwork of Data @n
asem Signup https://t.co/jXsQ… (https://t.co/jXsQ…)

miinaraut: RT @tenkahen: Interested in doing a PhD in spatial data science? The position for doing your doctoral studies at @
AaltoUniversity  is still…

AICareerLab: Accelerate your career at the Virtual Ai+ Professionals Expo with talks designed to help you navigate the compet
iti… https://t.co/wuqe50YlyC (https://t.co/wuqe50YlyC)

bungobot1: RT @DD_NaNa_: To better align data teams with business operations, a new organizational structure is needed http
s://t.co/L1jx8p3dFZ (https://t.co/L1jx8p3dFZ)

SiDa_Industrie: RT @DataSharingEU: We welcome our new participant @UM_IDS that specialises in FAIR data, formalised knowledge
representation, social networ…

GuidanceHT: RT @UoL_UGrad_LSE: Earn a BSc from the University of London online for less than $30,000. Designed by the London
 School of Economics and Po…

etsii_urjc: · Máster en ciberseguridad y privacidad (11 Junio 17:00)
· Máster en cloud apps: desarrollo y despliegue de aplicac… https://t.co/bJb9sveyEK (https://t.co/bJb9sveyEK)

tlearningagency: 👀  We saw +250 organizations make a commitment to the Data Science for Everyone campaign. Will you too?… h
ttps://t.co/9iyCZQMAHW (https://t.co/9iyCZQMAHW)
```

4.4 스트림으로 대량의 트윗 받기

In [28]:

```
from twython import TwythonStreamer

# Appending data to a global variable is pretty poor form
# but it makes the example much simpler
tweets = []

class MyStreamer(TwythonStreamer):
    def on_success(self, data):
        """
        What do we do when twitter sends us data?
        Here data will be a Python dict representing a tweet
        """
        # We only want to collect English-language tweets
        if data.get('lang') == 'en':
            tweets.append(data)
            print(f"received tweet #{len(tweets)}")

        # Stop when we've collected enough
        if len(tweets) >= 100:
            self.disconnect()

    def on_error(self, status_code, data):
        print(status_code, data)
        self.disconnect()
```

4.5 Data가 포함된 트윗 다운로드

In [29]:

```
stream = MyStreamer(CONSUMER_KEY, CONSUMER_SECRET,
                    ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

# starts consuming public statuses that contain the keyword 'data'
stream.statuses.filter(track='data')

# if instead we wanted to start consuming a sample of *all* public statuses
# stream.statuses.sample()
```

```
received tweet #1
received tweet #2
received tweet #3
received tweet #4
received tweet #5
received tweet #6
received tweet #7
received tweet #8
received tweet #9
received tweet #10
received tweet #11
received tweet #12
received tweet #13
received tweet #14
received tweet #15
received tweet #16
received tweet #17
received tweet #18
received tweet #19
received tweet #20
received tweet #21
received tweet #22
received tweet #23
received tweet #24
received tweet #25
received tweet #26
received tweet #27
received tweet #28
received tweet #29
received tweet #30
received tweet #31
received tweet #32
received tweet #33
received tweet #34
received tweet #35
received tweet #36
received tweet #37
received tweet #38
received tweet #39
received tweet #40
received tweet #41
received tweet #42
received tweet #43
received tweet #44
received tweet #45
received tweet #46
received tweet #47
received tweet #48
received tweet #49
received tweet #50
received tweet #51
received tweet #52
received tweet #53
received tweet #54
received tweet #55
received tweet #56
received tweet #57
received tweet #58
received tweet #59
received tweet #60
received tweet #61
received tweet #62
received tweet #63
received tweet #64
received tweet #65
received tweet #66
received tweet #67
received tweet #68
received tweet #69
received tweet #70
received tweet #71
received tweet #72
received tweet #73
received tweet #74
received tweet #75
received tweet #76
received tweet #77
received tweet #78
received tweet #79
received tweet #80
received tweet #81
received tweet #82
received tweet #83
received tweet #84
received tweet #85
received tweet #86
received tweet #87
received tweet #88
received tweet #89
received tweet #90
received tweet #91
received tweet #92
received tweet #93
received tweet #94
received tweet #95
received tweet #96
received tweet #97
received tweet #98
received tweet #99
received tweet #100
```

In [30]:

```
print(tweets[0])
```

```
{'created_at': 'Mon May 31 14:07:19 +0000 2021', 'id': 1399366886102536193, 'id_str': '1399366886102536193', 'text': '@Empero
rBTC 7 years working on projects related to startups, specializing in competitor analysis, data management,… https://t.co/4w
jmTJKhg9', (https://t.co/4wjmTJKhg9',) 'display_text_range': [12, 140], 'source': '<a href="https://mobile.twitter.com" rel
="nofollow">Twitter Web App</a>', 'truncated': True, 'in_reply_to_status_id': 1399365824641568768, 'in_reply_to_status_id_st
r': '1399365824641568768', 'in_reply_to_user_id': 183951857, 'in_reply_to_user_id_str': '183951857', 'in_reply_to_screen_nam
e': 'EmperorBTC', 'user': {'id': 1325623166, 'id_str': '1325623166', 'name': 'Fox', 'screen_name': '9Jzs', 'location': 'Barce
lona', 'url': None, 'description': None, 'translator_type': 'none', 'protected': False, 'verified': False, 'followers_count':
125, 'friends_count': 344, 'listed_count': 1, 'favourites_count': 6871, 'statuses_count': 8379, 'created_at': 'Wed Apr 03 22:
59:04 +0000 2013', 'utc_offset': None, 'time_zone': None, 'geo_enabled': False, 'lang': None, 'contributors_enabled': False,
 'is_translator': False, 'profile_background_color': '000000', 'profile_background_image_url': 'http://abs.twimg.com/images/t
hemes/theme1/bg.png', 'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme1/bg.png', 'profile_bac
kground_tile': False, 'profile_link_color': '1B95E0', 'profile_sidebar_border_color': '000000', 'profile_sidebar_fill_color':
'000000', 'profile_text_color': '000000', 'profile_use_background_image': False, 'profile_image_url': 'http://pbs.twimg.com/p
rofile_images/1291664142164725760/WKeP-M2-_normal.jpg', 'profile_image_url_https': 'https://pbs.twimg.com/profile_images/1291
664142164725760/WKeP-M2-_normal.jpg', 'profile_banner_url': 'https://pbs.twimg.com/profile_banners/1325623166/1615897027', 'd
efault_profile': False, 'default_profile_image': False, 'following': None, 'follow_request_sent': None, 'notifications': Non
e, 'withheld_in_countries': []}, 'geo': None, 'coordinates': None, 'place': None, 'contributors': None, 'is_quote_status': Fa
lse, 'extended_tweet': {'full_text': '@EmperorBTC 7 years working on projects related to startups, specializing in competitor
analysis, data management, etc., as well as developing strategic plans. You have helped me in trading so it would be an honor
to help you in whatever you need.', 'display_text_range': [12, 247], 'entities': {'hashtags': [], 'urls': [], 'user_mention
s': [{'screen_name': 'EmperorBTC', 'name': 'Emperor👑', 'id': 183951857, 'id_str': '183951857', 'indices': [0, 11]}], 'symbol
s': []}}, 'quote_count': 0, 'reply_count': 0, 'retweet_count': 0, 'favorite_count': 0, 'entities': {'hashtags': [], 'urls':
 [{'url': 'https://t.co/4wjmTJKhg9', 'expanded_url': 'https://twitter.com/i/web/status/1399366886102536193', 'display_url':
 'twitter.com/i/web/status/1…', 'indices': [116, 139]}], 'user_mentions': [{'screen_name': 'EmperorBTC', 'name': 'Emperor
👑', 'id': 183951857, 'id_str': '183951857', 'indices': [0, 11]}], 'symbols': []}, 'favorited': False, 'retweeted': False, 'f
ilter_level': 'low', 'lang': 'en', 'timestamp_ms': '1622470039283'}
```

4.6 가장 많이 나오는 해시테그 찾기

In [31]:

```
from collections import Counter

top_hashtags = Counter(hashtag['text'].lower()
                       for tweet in tweets
                       for hashtag in tweet['entities']['hashtags'])
print(top_hashtags.most_common(5))
```

```
[('ai', 8), ('besmartlikebts', 4), ('judicialdeeppockets', 1), ('pot', 1), ('fbo', 1)]
```

23장. 추천 시스템 (Recommender Systems)

1. 인기 순위로 추천

1.1 데이터셋 정의

In [1]:

```
users_interests = [
    ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],
    ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],
    ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],
    ["R", "Python", "statistics", "regression", "probability"],
    ["machine learning", "regression", "decision trees", "libsvm"],
    ["Python", "R", "Java", "C++", "Haskell", "programming languages"],
    ["statistics", "probability", "mathematics", "theory"],
    ["machine learning", "scikit-learn", "Mahout", "neural networks"],
    ["neural networks", "deep learning", "Big Data", "artificial intelligence"],
    ["Hadoop", "Java", "MapReduce", "Big Data"],
    ["statistics", "R", "statsmodels"],
    ["C++", "deep learning", "artificial intelligence", "probability"],
    ["pandas", "R", "Python"],
    ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],
    ["libsvm", "regression", "support vector machines"]
]
```

1.2 관심 종목 인기 순위

In [2]:

```
from collections import Counter

popular_interests = Counter(interest
                            for user_interests in users_interests
                            for interest in user_interests)
print(popular_interests)
```

```
Counter({'Python': 4, 'R': 4, 'Big Data': 3, 'HBase': 3, 'Java': 3, 'statistics': 3, 'regression': 3, 'probability': 3, 'Hado
op': 2, 'Cassandra': 2, 'MongoDB': 2, 'Postgres': 2, 'scikit-learn': 2, 'statsmodels': 2, 'pandas': 2, 'machine learning': 2,
'libsvm': 2, 'C++': 2, 'neural networks': 2, 'deep learning': 2, 'artificial intelligence': 2, 'Spark': 1, 'Storm': 1, 'NoSQ
L': 1, 'scipy': 1, 'numpy': 1, 'decision trees': 1, 'Haskell': 1, 'programming languages': 1, 'mathematics': 1, 'theory': 1,
'Mahout': 1, 'MapReduce': 1, 'databases': 1, 'MySQL': 1, 'support vector machines': 1})
```

1.3 인기 순위로 추천

In [3]:

```
from typing import Dict, List, Tuple

def most_popular_new_interests(
        user_interests: List[str],
        max_results: int = 5) -> List[Tuple[str, int]]:
    suggestions = [(interest, frequency)
                   for interest, frequency in popular_interests.most_common()
                   if interest not in user_interests]
    return suggestions[:max_results]
```

In [4]:

```
print(most_popular_new_interests(users_interests[1]))
```

```
[('Python', 4), ('R', 4), ('Big Data', 3), ('Java', 3), ('statistics', 3)]
```

2. 사용자 기반 협업 필터링 (User-Based Collaborative Filtering)

2.1 관심사 목록

In [5]:
```
unique_interests = sorted({interest
                           for user_interests in users_interests
                           for interest in user_interests})

print(unique_interests)
assert unique_interests[:6] == [
    'Big Data',
    'C++',
    'Cassandra',
    'HBase',
    'Hadoop',
    'Haskell',
    # ...
]
```

```
['Big Data', 'C++', 'Cassandra', 'HBase', 'Hadoop', 'Haskell', 'Java', 'Mahout', 'MapReduce', 'MongoDB', 'MySQL', 'NoSQL', 'P
ostgres', 'Python', 'R', 'Spark', 'Storm', 'artificial intelligence', 'databases', 'decision trees', 'deep learning', 'libsv
m', 'machine learning', 'mathematics', 'neural networks', 'numpy', 'pandas', 'probability', 'programming languages', 'regress
ion', 'scikit-learn', 'scipy', 'statistics', 'statsmodels', 'support vector machines', 'theory']
```

2.2 사용자 별 관심사 벡터

In [6]:
```
def make_user_interest_vector(user_interests: List[str]) -> List[int]:
    """
    Given a list ofinterests, produce a vector whose ith element is 1
    if unique_interests[i] is in the list, 0 otherwise
    """
    return [1 if interest in user_interests else 0
            for interest in unique_interests]
```

In [7]:
```
user_interest_vectors = [make_user_interest_vector(user_interests)
                         for user_interests in users_interests]
```

2.3 사용자 유사도 행렬

2.3.1 코사인 유사도 (cosine similarity)

In [8]:
```
from scratch.linear_algebra import dot, Vector
import math

def cosine_similarity(v1: Vector, v2: Vector) -> float:
    return dot(v1, v2) / math.sqrt(dot(v1, v1) * dot(v2, v2))
```

2.3.2 사용자 유사도 행렬

In [9]:
```
user_similarities = [[cosine_similarity(interest_vector_i, interest_vector_j)
                      for interest_vector_j in user_interest_vectors]
                     for interest_vector_i in user_interest_vectors]
```

In [10]:
```
# Users 0 and 9 share interests in Hadoop, Java, and Big Data
assert 0.56 < user_similarities[0][9] < 0.58, "several shared interests"

# Users 0 and 8 share only one interest: Big Data
assert 0.18 < user_similarities[0][8] < 0.20, "only one shared interest"
```

2.4 유사한 사용자 목록

In [11]:
```
def most_similar_users_to(user_id: int) -> List[Tuple[int, float]]:
    pairs = [(other_user_id, similarity)                      # Find other
             for other_user_id, similarity in                 # users with
                enumerate(user_similarities[user_id])         # nonzero
             if user_id != other_user_id and similarity > 0]  # similarity.

    return sorted(pairs,                                      # Sort them
                  key=lambda pair: pair[-1],                  # most similar
                  reverse=True)                               # first.
```

```
In [12]: most_similar_to_zero = most_similar_users_to(0)
         print(most_similar_to_zero)
         user, score = most_similar_to_zero[0]
         assert user == 9
         assert 0.56 < score < 0.57
         user, score = most_similar_to_zero[1]
         assert user == 1
         assert 0.33 < score < 0.34
```

```
[(9, 0.5669467095138409), (1, 0.3380617018914066), (8, 0.1889822365046136), (13, 0.1690308509457033), (5, 0.154303349962091
9)]
```

2.5 사용자 기반 추천

```
In [13]: from collections import defaultdict

         def user_based_suggestions(user_id: int,
                                    include_current_interests: bool = False):
             # Sum up the similarities.
             suggestions: Dict[str, float] = defaultdict(float)
             for other_user_id, similarity in most_similar_users_to(user_id):
                 for interest in users_interests[other_user_id]:
                     suggestions[interest] += similarity

             # Convert them to a sorted list.
             suggestions = sorted(suggestions.items(),
                                  key=lambda pair: pair[-1],  # weight
                                  reverse=True)

             # And (maybe) exclude already-interests
             if include_current_interests:
                 return suggestions
             else:
                 return [(suggestion, weight)
                         for suggestion, weight in suggestions
                         if suggestion not in users_interests[user_id]]
```

```
In [14]: ubs0 = user_based_suggestions(0)
         print(ubs0)
         interest, score = ubs0[0]
         assert interest == 'MapReduce'
         assert 0.56 < score < 0.57
         interest, score = ubs0[1]
         assert interest == 'MongoDB'
         assert 0.50 < score < 0.51
```

```
[('MapReduce', 0.5669467095138409), ('MongoDB', 0.50709255283711), ('Postgres', 0.50709255283711), ('NoSQL', 0.33806170189140
66), ('neural networks', 0.1889822365046136), ('deep learning', 0.1889822365046136), ('artificial intelligence', 0.1889822365
046136), ('databases', 0.1690308509457033), ('MySQL', 0.1690308509457033), ('Python', 0.1543033499620919), ('R', 0.1543033499
620919), ('C++', 0.1543033499620919), ('Haskell', 0.1543033499620919), ('programming languages', 0.1543033499620919)]
```

3. 아이템 기반 협업 필터링 (Item-Based Collaborative Filtering)

3.1 아이템-사용자 행렬

```
In [15]: interest_user_matrix = [[user_interest_vector[j]
                                   for user_interest_vector in user_interest_vectors]
                                  for j, _ in enumerate(unique_interests)]
```

```
In [16]: interest_user_matrix[0]
```

```
Out[16]: [1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
```

3.2 아이템 유사도 행렬

```
In [17]: interest_similarities = [[cosine_similarity(user_vector_i, user_vector_j)
                                    for user_vector_j in interest_user_matrix]
                                   for user_vector_i in interest_user_matrix]
```

3.3 유사한 아이템 목록

In [18]:
```
def most_similar_interests_to(interest_id: int):
    similarities = interest_similarities[interest_id]
    pairs = [(unique_interests[other_interest_id], similarity)
             for other_interest_id, similarity in enumerate(similarities)
             if interest_id != other_interest_id and similarity > 0]
    return sorted(pairs,
                  key=lambda pair: pair[-1],
                  reverse=True)
```

In [19]:
```
msit0 = most_similar_interests_to(0)
print(msit0)
assert msit0[0][0] == 'Hadoop'
assert 0.815 < msit0[0][1] < 0.817
assert msit0[1][0] == 'Java'
assert 0.666 < msit0[1][1] < 0.667
```

```
[('Hadoop', 0.8164965809277261), ('Java', 0.6666666666666666), ('MapReduce', 0.5773502691896258), ('Spark', 0.577350269189625
8), ('Storm', 0.5773502691896258), ('Cassandra', 0.4082482904638631), ('artificial intelligence', 0.4082482904638631), ('deep
learning', 0.4082482904638631), ('neural networks', 0.4082482904638631), ('HBase', 0.3333333333333333)]
```

3.4 아이템 기반 추천

In [20]:
```
def item_based_suggestions(user_id: int,
                           include_current_interests: bool = False):
    # Add up the similar interests
    suggestions = defaultdict(float)
    user_interest_vector = user_interest_vectors[user_id]
    for interest_id, is_interested in enumerate(user_interest_vector):
        if is_interested == 1:
            similar_interests = most_similar_interests_to(interest_id)
            for interest, similarity in similar_interests:
                suggestions[interest] += similarity

    # Sort them by weight
    suggestions = sorted(suggestions.items(),
                         key=lambda pair: pair[-1],
                         reverse=True)

    if include_current_interests:
        return suggestions
    else:
        return [(suggestion, weight)
                for suggestion, weight in suggestions
                if suggestion not in users_interests[user_id]]
```

In [21]:
```
ibs0 = item_based_suggestions(0)
print(ibs0)
assert ibs0[0][0] == 'MapReduce'
assert 1.86 < ibs0[0][1] < 1.87
assert ibs0[1][0] in ('Postgres', 'MongoDB')  # A tie
assert 1.31 < ibs0[1][1] < 1.32
```

```
[('MapReduce', 1.861807319565799), ('MongoDB', 1.3164965809277263), ('Postgres', 1.3164965809277263), ('NoSQL', 1.28445705037
61732), ('MySQL', 0.5773502691896258), ('databases', 0.5773502691896258), ('Haskell', 0.5773502691896258), ('programming lang
uages', 0.5773502691896258), ('artificial intelligence', 0.4082482904638631), ('deep learning', 0.4082482904638631), ('neural
networks', 0.4082482904638631), ('C++', 0.4082482904638631), ('Python', 0.2886751345948129), ('R', 0.2886751345948129)]
```

4. 잠재 요인 기반 협업 필터링

4.1 데이터 타입 정의

4.1.1 평점 파일 경로

In [22]:
```
# This points to the current directory, modify if your files are elsewhere.
MOVIES = "ml-100k₩u.item"   # pipe-delimited: movie_id|title|...
RATINGS = "ml-100k₩u.data"  # tab-delimited: user_id, movie_id, rating, timestamp
```

4.1.2 평점 NamedTuple

In [23]:

```
from typing import NamedTuple

class Rating(NamedTuple):
    user_id: str
    movie_id: str
    rating: float
```

4.2 데이터 읽기

In [24]:

```
import csv
# We specify this encoding to avoid a UnicodeDecodeError.
# see: https://stackoverflow.com/a/53136168/1076346
with open(MOVIES, encoding="iso-8859-1") as f:
    reader = csv.reader(f, delimiter="|")
    movies = {movie_id: title for movie_id, title, *_ in reader}

# Create a list of [Rating]
with open(RATINGS, encoding="iso-8859-1") as f:
    reader = csv.reader(f, delimiter="₩t")
    ratings = [Rating(user_id, movie_id, float(rating))
               for user_id, movie_id, rating, _ in reader]

# 1682 movies rated by 943 users
assert len(movies) == 1682
assert len(list({rating.user_id for rating in ratings})) == 943
```

4.3 데이터 탐색 (스타워즈 평점 확인)

In [25]:

```
import re

# Data structure for accumulating ratings by movie_id
star_wars_ratings = {movie_id: []
                     for movie_id, title in movies.items()
                     if re.search("Star Wars|Empire Strikes|Jedi", title)}

# Iterate over ratings, accumulating the Star Wars ones
for rating in ratings:
    if rating.movie_id in star_wars_ratings:
        star_wars_ratings[rating.movie_id].append(rating.rating)

# Compute the average rating for each movie
avg_ratings = [(sum(title_ratings) / len(title_ratings), movie_id)
               for movie_id, title_ratings in star_wars_ratings.items()]

# And then print them in order
for avg_rating, movie_id in sorted(avg_ratings, reverse=True):
    print(f"{avg_rating:.2f} {movies[movie_id]}")
```

```
4.36 Star Wars (1977)
4.20 Empire Strikes Back, The (1980)
4.01 Return of the Jedi (1983)
```

4.4 데이터 분리

In [26]:

```
import random
random.seed(0)
random.shuffle(ratings)

split1 = int(len(ratings) * 0.7)
split2 = int(len(ratings) * 0.85)

train = ratings[:split1]               # 70% of the data
validation = ratings[split1:split2]    # 15% of the data
test = ratings[split2:]                # 15% of the data
```

4.5 베이스라인 생성

In [27]:

```
avg_rating = sum(rating.rating for rating in train) / len(train)
baseline_error = sum((rating.rating - avg_rating) ** 2
                     for rating in test) / len(test)

print(avg_rating)
print(baseline_error)

# This is what we hope to do better than
assert 1.26 < baseline_error < 1.27
```

```
3.530842857142857
1.2609526646939684
```

4.6 임베딩 생성

In [28]:

```
# Embedding vectors for matrix factorization model

from scratch.deep_learning import random_tensor

EMBEDDING_DIM = 2

# Find unique ids
user_ids = {rating.user_id for rating in ratings}
movie_ids = {rating.movie_id for rating in ratings}

# Then create a random vector per id
user_vectors = {user_id: random_tensor(EMBEDDING_DIM)
                for user_id in user_ids}
movie_vectors = {movie_id: random_tensor(EMBEDDING_DIM)
                 for movie_id in movie_ids}
```

```
<Figure size 432x288 with 0 Axes>
```

4.7 모델 학습

In [29]:

```
# Training loop for matrix factorization model

from typing import List
from tqdm.notebook import tqdm
from scratch.linear_algebra import dot

def fit(dataset: List[Rating],
            learning_rate: float = None) -> float:
    loss = 0.0
    for i, rating in enumerate(dataset):
        movie_vector = movie_vectors[rating.movie_id]
        user_vector = user_vectors[rating.user_id]
        predicted = dot(user_vector, movie_vector)
        error = predicted - rating.rating
        loss += error ** 2

        if learning_rate is not None:
            #     predicted = m_0 * u_0 + ... + m_k * u_k
            # So each u_j enters output with coefficent m_j
            # and each m_j enters output with coefficient u_j
            user_gradient = [2*error*m_j for m_j in movie_vector]
            movie_gradient = [2*error*u_j for u_j in user_vector]

            # Take gradient steps
            for j in range(EMBEDDING_DIM):
                user_vector[j] -= learning_rate * user_gradient[j]
                movie_vector[j] -= learning_rate * movie_gradient[j]
    return loss/len(dataset)
```

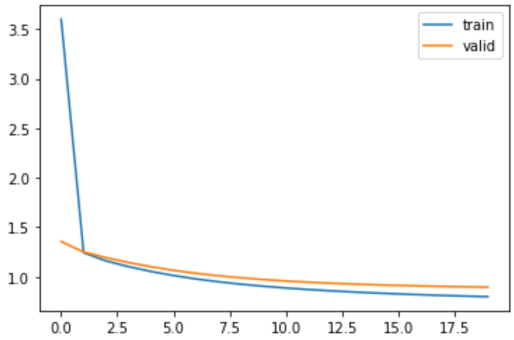
In [30]:

```
def evaluate(dataset: List[Rating]) -> float:
    loss = 0.0
    for i, rating in enumerate(dataset):
        movie_vector = movie_vectors[rating.movie_id]
        user_vector = user_vectors[rating.user_id]
        predicted = dot(user_vector, movie_vector)
        error = predicted - rating.rating
        loss += error ** 2
    return loss/len(dataset)
```

```
In [31]: from tqdm import trange, tqdm
         epoch = 20
         learning_rate = 0.05

         train_history = []
         validation_history = []
         with trange(epoch) as t:
             for epoch in t:
                 learning_rate *= 0.9
                 train_loss = fit(train, learning_rate=learning_rate)
                 valid_loss = evaluate(validation)
                 t.set_description(f"epoch : {epoch}, train loss: {train_loss :.2f}, valid loss: {valid_loss :.2f}")
                 train_history.append(train_loss)
                 validation_history.append(valid_loss)
```

```
epoch : 19, train loss: 0.80, valid loss: 0.90: 100%|██████████████████████████████| 20/20 [00:05<00:00,  3.70it/s]
```

```
In [32]: import matplotlib.pyplot as plt
         plt.plot(train_history, label="train")
         plt.plot(validation_history, label='valid')
         plt.legend()
         plt.show()
```



4.8 모델 테스트

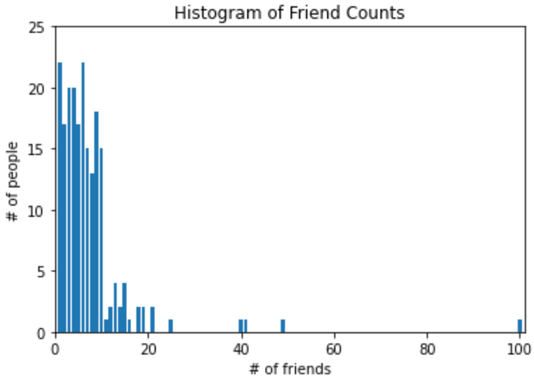
```
In [33]: test_loss = evaluate(test)
         print(f"baseline error : {baseline_error}, predicted error : {test_loss}")
```

```
baseline error : 1.2609526646939684, predicted error : 0.9055570858779906
```

4.9 영화 임베딩 주성분 분석

```
In [34]: from scratch.working_with_data import pca, transform
         original_vectors = [vector for vector in movie_vectors.values()]
         components = pca(original_vectors, 2)
```

```
dv: 4715.223: 100%|████████████████████████████████████████████████████████| 100/100 [00:00<00:00, 148.49it/s]
dv: 956.953: 100%|█████████████████████████████████████████████████████████| 100/100 [00:00<00:00, 130.96it/s]
```

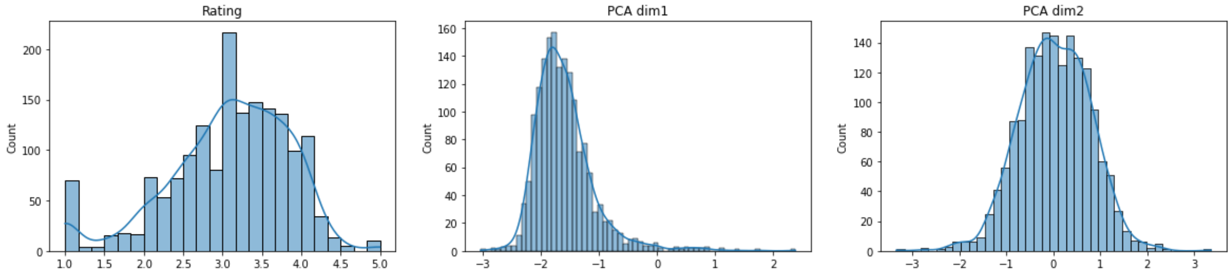


4.9.1 주성분으로 사영

In [35]:
```
ratings_by_movie = defaultdict(list)
for rating in ratings:
    ratings_by_movie[rating.movie_id].append(rating.rating)

vectors = [
    (movie_id,
        sum(ratings_by_movie[movie_id]) / len(ratings_by_movie[movie_id]),
        movies[movie_id],
        vector)
    for movie_id, vector in zip(movie_vectors.keys(),
                                transform(original_vectors, components))
]
```

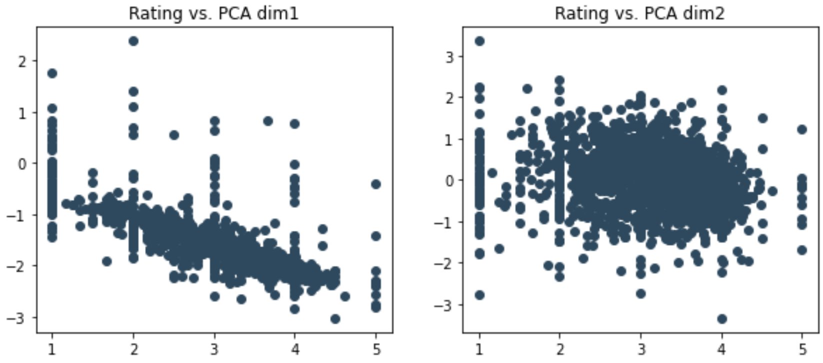
4.9.2 평점과 임베딩 분포

In [36]:
```
import matplotlib.pyplot as plt
import seaborn as sns
ratings = [vector[1] for vector in vectors]
pca_dim1 = [vector[-1][0] for vector in vectors]
pca_dim2 = [vector[-1][1] for vector in vectors]

plt.figure(figsize=[20,4])
plt.subplot(1,3,1)
sns.histplot(data=ratings, kde=True)
plt.title('Rating')
plt.subplot(1,3,2)
sns.histplot(data=pca_dim1, kde=True)
plt.title('PCA dim1')
plt.subplot(1,3,3)
sns.histplot(data=pca_dim2, kde=True)
plt.title('PCA dim2')
plt.show()
```



4.9.3 평점과 임베딩의 상관성

In [37]:
```
plt.figure(figsize=[10,4])
plt.subplot(1,2,1)
plt.scatter(ratings, pca_dim1, facecolor="#2E495E")
plt.title('Rating vs. PCA dim1')
plt.subplot(1,2,2)
plt.scatter(ratings, pca_dim2, facecolor="#2E495E")
plt.title('Rating vs. PCA dim2')
plt.show()
```


Rating vs. PCA dim1
Rating vs. PCA dim2