1장. 데이터 마이닝 소개

예제 1. 핵심 인물 찾아라

사용자 목록 (그래프 노드)

In [78]:
```
users = [
    { "id": 0, "name": "Hero" },
    { "id": 1, "name": "Dunn" },
    { "id": 2, "name": "Sue" },
    { "id": 3, "name": "Chi" },
    { "id": 4, "name": "Thor" },
    { "id": 5, "name": "Clive" },
    { "id": 6, "name": "Hicks" },
    { "id": 7, "name": "Devin" },
    { "id": 8, "name": "Kate" },
    { "id": 9, "name": "Klein" }
]
#delete cells => D,D
```

친구 관계 목록 (그래프 에지)

In [79]:
```
friendship_pairs = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),
                    (4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]
```

Q. 친구 관계 목록 (인접 리스트)

In [80]:
```
# Initialize the dict with an empty list for each user id:
friendships = {user["id"]: [] for user in users}  #loop로 list형태
#friendships에 각각 유저 id대입

# And loop over the friendship pairs to populate it:
for i, j in friendship_pairs:
    friendships[i].append(j) #j를 i의 친구로 추가
    friendships[j].append(i) #i를 j의 친구로 추가
```

Q. 친구 수 세기

In [81]:
```
def number_of_friends(user):
    """How many friends does _user_ have?"""
    user_id = user["id"]
    friend_ids = friendships[user_id]
    return len(friend_ids)
```

Q. 전체 친구 수 세기

In [82]:
```
total_connections = sum(number_of_friends(user)
                        for user in users)


assert total_connections == 24
```

평균 친구 수 세기

In [83]:
```
num_users = len(users)                              # length of the users list
avg_connections = total_connections / num_users     # 24 / 10 == 2.4


assert num_users == 10
assert avg_connections == 2.4
```

Q. 사용자 별 친구 수 목록 생성

[(1, 3), (2, 3), (3, 3), (5, 3), (8, 3), (0, 2), (4, 2), (6, 2), (7, 2), (9, 1)]

In [84]:
```
# Create a list (user_id, number_of_friends).
num_friends_by_id = [(user["id"], number_of_friends(user)) for user in users]
```

In [85]:
```
num_friends_by_id.sort(                                 # Sort the list
       key=lambda id_and_friends: id_and_friends[1],    # by num_friends
       reverse=True)                                    # largest to smallest

print(num_friends_by_id)
# Each pair is (user_id, num_friends):
# [(1, 3), (2, 3), (3, 3), (5, 3), (8, 3),
#  (0, 2), (4, 2), (6, 2), (7, 2), (9, 1)]

assert num_friends_by_id[0][1] == 3     # several people have 3 friends
assert num_friends_by_id[-1] == (9, 1)  # user 9 has only 1 friend
```

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

예제2. 친구 추천하기

친구의 친구 목록 만들기 (않좋은 버전)

In [86]:
```
def foaf_ids_bad(user):
    """foaf is short for "friend of a friend" """
    return [foaf_id
            for friend_id in friendships[user["id"]]
            for foaf_id in friendships[friend_id]]

[0, 2, 3, 0, 1, 3]

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

In [87]:
```
print(friendships[0])  # [1, 2]
print(friendships[1])  # [0, 2, 3]
print(friendships[2])  # [0, 1, 3]

assert friendships[0] == [1, 2]
assert friendships[1] == [0, 2, 3]
assert friendships[2] == [0, 1, 3]
```

```
[1, 2]
[0, 2, 3]
[0, 1, 3]
```

Q. 친구의 친구 목록 만들기

In [88]:
```
from collections import Counter                    # not loaded by default

# 나와 친구를 제외한 친구의 친구 목록 만들기
def friends_of_friends(user):
    user_id = user["id"]
    return Counter(
        foaf_id
        for friend_id in friendships[user_id]   #각각의 친구들을 대입
        for foaf_id in friendships[friend_id]    #그들의 친구 찾아라. 대신
        if foaf_id != user_id                  #친구목록 만드는데 자기자신 빼고
        and foaf_id not in friendships[user_id] #그리고 내 친구들도 빼고
    )
print(friends_of_friends(users[3]))               # Counter({0: 2, 5: 1})
assert friends_of_friends(users[3]) == Counter({0: 2, 5: 1})
```

```
Counter({0: 2, 5: 1})
```

같은 관심을 갖는 친구 추천하기

In [89]:
```
interests = [
    (0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),
    (0, "Spark"), (0, "Storm"), (0, "Cassandra"),
    (1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),
    (1, "Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"),
    (2, "numpy"), (2, "statsmodels"), (2, "pandas"), (3, "R"), (3, "Python"),
    (3, "statistics"), (3, "regression"), (3, "probability"),
    (4, "machine learning"), (4, "regression"), (4, "decision trees"),
    (4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),
    (5, "Haskell"), (5, "programming languages"), (6, "statistics"),
    (6, "probability"), (6, "mathematics"), (6, "theory"),
    (7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),
    (7, "neural networks"), (8, "neural networks"), (8, "deep learning"),
    (8, "Big Data"), (8, "artificial intelligence"), (9, "Hadoop"),
    (9, "Java"), (9, "MapReduce"), (9, "Big Data")
]
```

Q. 관심 별 사용자 목록 구성

In [90]:
```
def data_scientists_who_like(target_interest):
    """Find the ids of all users who like the target interest."""
    return [user_id
            for user_id, user_interest in interests
            if user_interest == target_interest]
```

Q. 사용자 별 관심 목록 구성

In [91]:
```
from collections import defaultdict
# Keys are user_ids, values are lists of interests for that user_id.
user_ids_by_interest = defaultdict(list)

for user_id, interest in interests:
    interests_by_user_id[user_id].append(interest)
        #list형태에 user_id원소 자체를 넣음
```

Q. 같은 관심을 갖는 사람들 목록 (Counter 형태로 반환)

In [92]:
```
def most_common_interests_with(user):
    return Counter(
            interested_user_id
            for interest in interests_by_user_id[user["id"]]
            for interested_user_id in user_ids_by_interest[interest]
            if interested_user_id != user["id"]
    )
```

In [93]:
```
print(most_common_interests_with(users[0]).most_common())
```

```
[]
```

예제3. 연봉과 근속연수의 관계를 찾아라

직원들의 연봉 및 근속연수 테이블

In [94]:
```
salaries_and_tenures = [(83000, 8.7), (88000, 8.1),
                        (48000, 0.7), (76000, 6),
                        (69000, 6.5), (76000, 7.5),
                        (60000, 2.5), (83000, 10),
                        (48000, 1.9), (63000, 4.2)]
```

근속연수 별 평균 연봉 계산

In [95]:
```
# Keys are years, values are lists of the salaries for each tenure.
salary_by_tenure = defaultdict(list)

for salary, tenure in salaries_and_tenures:
    salary_by_tenure[tenure].append(salary)

# Keys are years, each value is average salary for that tenure.
average_salary_by_tenure = {
    tenure: sum(salaries) / len(salaries)
    for tenure, salaries in salary_by_tenure.items()
}
```

In [96]:
```
assert average_salary_by_tenure == {
    0.7: 48000.0,
    1.9: 48000.0,
    2.5: 60000.0,
    4.2: 63000.0,
    6: 76000.0,
    6.5: 69000.0,
    7.5: 76000.0,
    8.1: 88000.0,
    8.7: 83000.0,
    10: 83000.0
}
```

근속연수 버킷 구성

In [97]:
```
def tenure_bucket(tenure):
    if tenure < 2:
        return "less than two"
    elif tenure < 5:
        return "between two and five"
    else:
        return "more than five"
```

근속연수 버킷 별 연봉 리스트 구성

In [98]:
```
# Keys are tenure buckets, values are lists of salaries for that bucket.
salary_by_tenure_bucket = defaultdict(list)

for salary, tenure in salaries_and_tenures:
    bucket = tenure_bucket(tenure)
    salary_by_tenure_bucket[bucket].append(salary)
```

근속연수 버킷 별 평균 연봉 계산

In [99]:
```
# Keys are tenure buckets, values are average salary for that bucket
average_salary_by_bucket = {
  tenure_bucket: sum(salaries) / len(salaries)
  for tenure_bucket, salaries in salary_by_tenure_bucket.items()
}
```

In [100]:
```
assert average_salary_by_bucket == {
    'between two and five': 61500.0,
    'less than two': 48000.0,
    'more than five': 79166.66666666667
}
```

예제4. 유료 계정 전환 대상자를 찾아라

In [101]:
```
def predict_paid_or_unpaid(years_experience):
  if years_experience < 3.0:
    return "paid"
  elif years_experience < 8.5:
    return "unpaid"
  else:
    return "paid"
```

시각화 (Visualization) 과제

In [1]:

```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib
₩site-packages (1.1.3)
Requirement already satisfied: pytz>=2017.2 in c:₩users₩이준용₩.conda₩envs₩data_mini
ng₩lib₩site-packages (from pandas) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:₩users₩이준용₩.conda₩envs
₩data_mining₩lib₩site-packages (from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.15.4 in c:₩users₩이준용₩.conda₩envs₩data_min
ing₩lib₩site-packages (from pandas) (1.19.2)
Requirement already satisfied: six>=1.5 in c:₩users₩이준용₩.conda₩envs₩data_mining₩l
ib₩site-packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

In [2]:

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:₩users₩이준용₩.conda₩envs₩data_mining
₩lib₩site-packages (3.3.2)
Requirement already satisfied: pillow>=6.2.0 in c:₩users₩이준용₩.conda₩envs₩data_min
ing₩lib₩site-packages (from matplotlib) (8.0.1)
Requirement already satisfied: cycler>=0.10 in c:₩users₩이준용₩.conda₩envs₩data_mini
ng₩lib₩site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:₩users
₩이준용₩.conda₩envs₩data_mining₩lib₩site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:₩users₩이준용₩.conda₩envs₩d
ata_mining₩lib₩site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: certifi>=2020.06.20 in c:₩users₩이준용₩.conda₩envs₩da
ta_mining₩lib₩site-packages (from matplotlib) (2020.6.20)
Requirement already satisfied: numpy>=1.15 in c:₩users₩이준용₩.conda₩envs₩data_minin
g₩lib₩site-packages (from matplotlib) (1.19.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:₩users₩이준용₩.conda₩envs₩data
_mining₩lib₩site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: six in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib₩si
te-packages (from cycler>=0.10->matplotlib) (1.15.0)
```

다음 블로그에 있는 시각화 예시와 데이터셋을 이용해서 시각화를 구현해본다.

https://towardsdatascience.com/10-viz-every-ds-should-know-4e4118f26fc3
(https://towardsdatascience.com/10-viz-every-ds-should-know-4e4118f26fc3)

In [3]:

```
import matplotlib.pyplot as plt
import pandas as pd
```

1. 히스토그램 (Histograms)

1) 데이터 읽기

In [4]:

```
dataset_path = "data/thermostat_rebates_by_zip_1000.csv"
dataset = pd.read_csv(dataset_path)

dataset.tail()
```

Out[4]:

	zip-code	rebate-usd	lat	lng	median-household-income	mean-household-income	population
995	40385	100	37.758499	-84.132959	43280	51428	3131
996	72433	100	36.030397	-91.049037	31934	36651	3067
997	90014	67	34.043478	-118.251931	13832	30121	7005
998	8021	90	39.807377	-75.002697	55858	63779	45515
999	68067	100	42.152506	-96.471658	39062	51461	1397

In [5]:

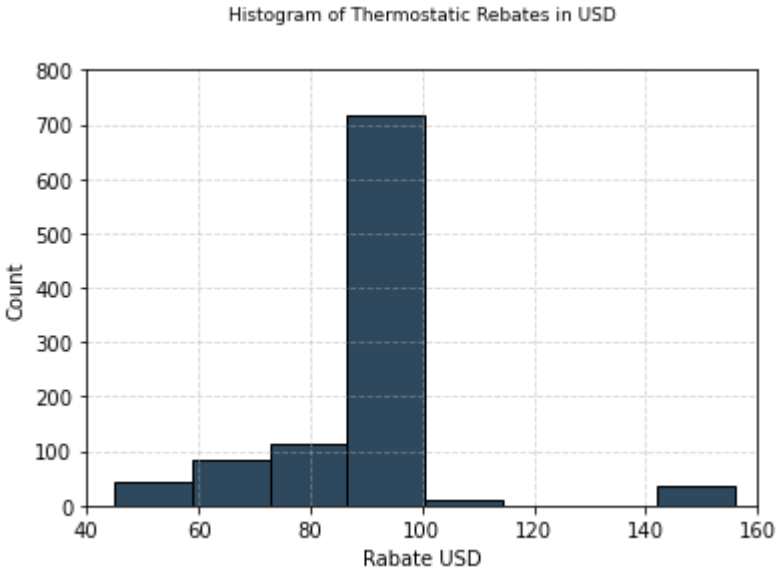
```
rebate = dataset["rebate-usd"]
print(rebate)
```

```
0      88
1      88
2     100
3     100
4     100
     ...
995   100
996   100
997    67
998    90
999   100
Name: rebate-usd, Length: 1000, dtype: int64
```

2) 시각화

In [6]:

```
n, bins, patches = plt.hist(rebate,
8,
facecolor="#2E495E",
edgecolor=(0, 0, 0),
align='mid',
)
# align은 tick과 막대의 위치를 조절합니다.
#디폴트는 'center' 인데 'edge'로 설정하면
#막대의 아래쪽 끝에 y_tick이 표시됩니다.

plt.axis([40,160,0,800])    # 축 범위 지정 [x min,x max , y min,y max]
plt.grid(True, alpha=0.5, linestyle='--')  # 격자 grid 생성
title_font = {
    'fontsize' : 9
}
plt.title("Histogram of Thermostatic Rebates in USD",title_font,pad=25)
plt.xlabel("Rabate USD")
plt.ylabel("Count")
plt.show()
```



2. 막대/파이 차트 (Bar/Pie charts)

1) 데이터 읽기

In [7]:

```
dataset_path = "data/drugs_data.csv"
dataset = pd.read_csv(dataset_path)

dataset.tail()
```

Out[7]:

	Age	Sex	BP	Cholesterol	NA_to_K	Drug
195	56	F	LOW	HIGH	11.566830	drugC
196	16	M	LOW	HIGH	12.006286	drugC
197	52	M	NORMAL	HIGH	9.894478	drugX
198	23	M	NORMAL	NORMAL	14.019550	drugX
199	40	F	LOW	NORMAL	11.348969	drugX

In [8]:

```
BP = dataset["BP"].value_counts()
BP
```

Out[8]:

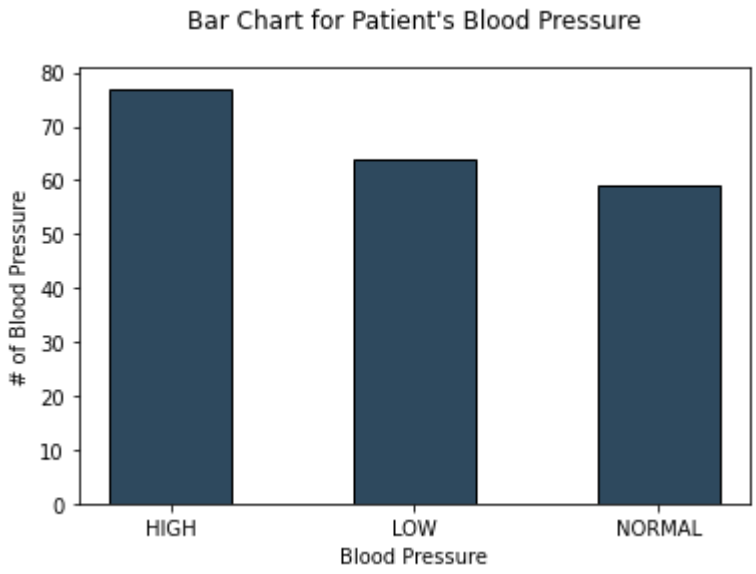
```
HIGH      77
LOW       64
NORMAL    59
Name: BP, dtype: int64
```

2) 시각화

막대 그래프

In [9]:

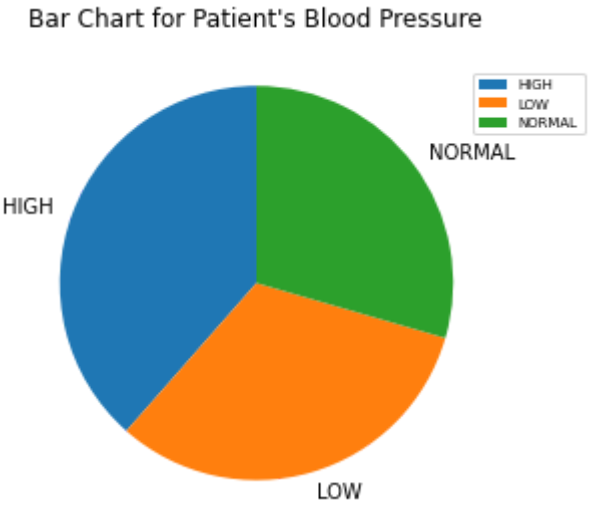
```
plt.bar(BP.keys(),
        BP,                                 # Give each bar its correct height
        0.5,                                # Give each bar a width of 0.5
        facecolor="#2E495E",
        edgecolor=(0, 0, 0)                 # Black edges for each bar
       )
# align은 tick과 막대의 위치를 조절합니다.
#디폴트는 'center' 인데 'edge'로 설정하면
#막대의 아래쪽 끝에 y_tick이 표시됩니다.


#plt.grid(True, alpha=0.5, linestyle='--')  # 격자 grid 생성

plt.title("Bar Chart for Patient's Blood Pressure", loc='center', pad=20)
#for loc; supported values are 'left', 'center', 'right'


plt.xlabel("Blood Pressure")
plt.ylabel("# of Blood Pressure")

plt.show()
```



파이 그래프

In [10]:

```
plt.pie(BP,labels=BP.keys(),startangle=90)
plt.title("Bar Chart for Patient's Blood Pressure",pad=20)
                                        # loc='center',pad =20
plt.axis('equal')
plt.legend(loc=1,fontsize=7)     #Location String 'upper right' =1
                                # legend의 폰트 사이즈 조절 가능
plt.show()
```



3. 산점도/직선 그래프 (Scatter/Line plots)

1) 데이터 읽기

In [11]:

```
dataset_path = "data/square-feet_and_house-price.csv"
dataset = pd.read_csv(dataset_path)

dataset.tail()
```

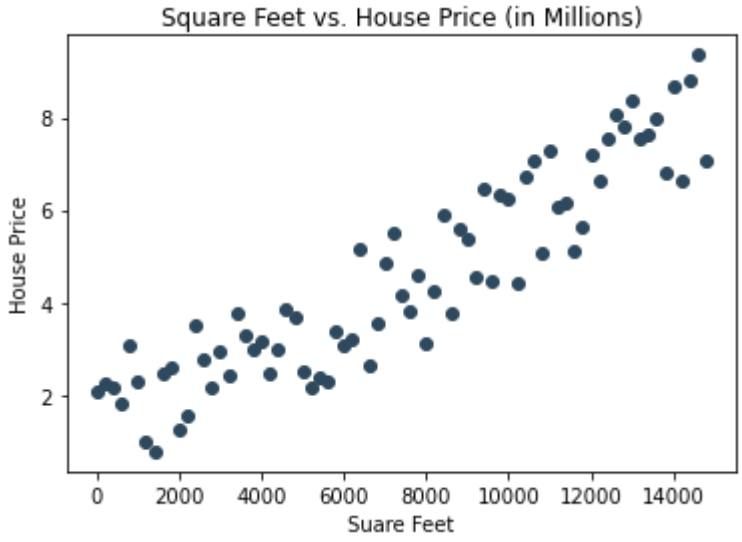
Out[11]:

	square-feet	house-price
70	14000.0	8.678987
71	14200.0	6.636067
72	14400.0	8.787156
73	14600.0	9.358178
74	14800.0	7.071544

2) 시각화

In [12]:

```
sq = dataset["square-feet"]
sq
hp = dataset["house-price"]
hp
plt.scatter(sq,hp,facecolor="#2E495E")

plt.title("Square Feet vs. House Price (in Millions)",pad=5)
                                # loc='center',pad =20
plt.xlabel("Suare Feet")
plt.ylabel("House Price")
plt.show()
```



4. 시계열 그래프 (Time series plot)

1) 데이터 읽기

In [13]:

```
dataset_path = "data/tesla_stock.csv"
dataset = pd.read_csv(dataset_path)

dataset.tail()
```

Out[13]:

	Date	Open	High	Low	Close	Volume
749	2015-01-08	212.81	213.7999	210.0100	210.615	3442509.0
750	2015-01-07	213.35	214.7800	209.7800	210.950	2968390.0
751	2015-01-06	210.06	214.2000	204.2100	211.280	6261936.0
752	2015-01-05	214.55	216.5000	207.1626	210.090	5368477.0
753	2015-01-02	222.87	223.2500	213.2600	219.310	4764443.0

In [14]:

```
sorted_dataset = dataset.sort_values(by='Date')
sorted_dataset.tail()
```

Out[14]:

	Date	Open	High	Low	Close	Volume
4	2017-12-22	329.51	330.9214	324.82	325.20	4186131.0
3	2017-12-26	323.83	323.9400	316.58	317.29	4321909.0
2	2017-12-27	316.00	317.6800	310.75	311.64	4645441.0
1	2017-12-28	311.75	315.8200	309.54	315.36	4294689.0
0	2017-12-29	316.18	316.4100	310.00	311.35	3727621.0

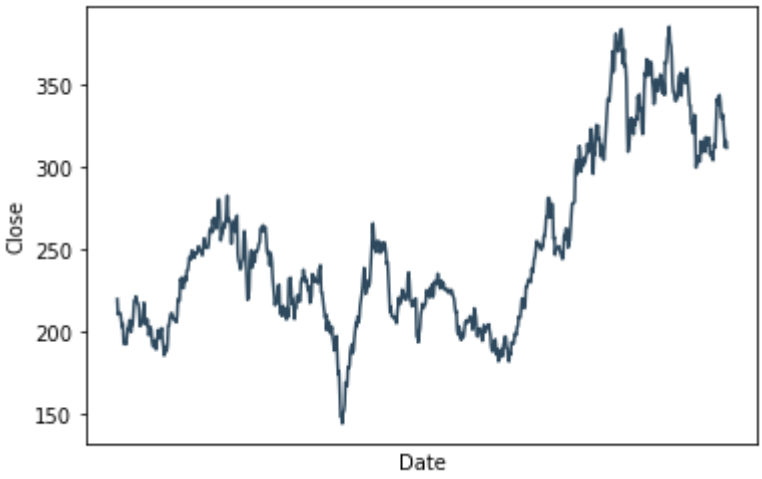
2) 시각화

In [15]:

```
Dt = sorted_dataset["Date"]
Cl = sorted_dataset["Close"]
plt.xticks([])     # x축 표시 안함
plt.xlabel("Date")
plt.ylabel("Close")
plt.plot(Dt,Cl,color="#2E495E")
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x2804dd032b0>]
```



k 최근접 이웃 (kNN) 과제

1. k-NN 분류기

In [1]:

```
from typing import List
from collections import Counter
```

1.1 투표 함수

In [2]:

```
def majority_vote(labels: List[str]) -> str:
    """Assumes that labels are ordered from nearest to farthest."""
    vote_counts = Counter(labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    num_winners = len([count
                       for count in vote_counts.values()
                       if count == winner_count])

    if num_winners == 1:
        return winner                     # unique winner, so return it
    else:
        return majority_vote(labels[:-1]) # try again without the farthest
```

1.2 데이터 포인트

In [3]:

```
from typing import NamedTuple
from scratch.linear_algebra import Vector, distance
```

In [4]:

```
class LabeledPoint(NamedTuple):
    point: Vector
    label: str
```

1.3 k-NN 분류기

In [5]:

```
def knn_classify(k: int,
                 labeled_points: List[LabeledPoint],
                 new_point: Vector) -> str:

    # Order the labeled points from nearest to farthest.
    by_distance = sorted(labeled_points,
                         key=lambda lp: distance(lp.point, new_point))

    # Find the labels for the k closest
    k_nearest_labels = [lp.label for lp in by_distance[:k]]

    # and let them vote.
    return majority_vote(k_nearest_labels)
```

2. 유방 암 데이터 분류

In [6]:

```
import matplotlib.pyplot as plt
import os
from typing import Dict
import csv
from collections import defaultdict
```

2.1 데이터셋 다운로드

위스콘신 유방암 진단 데이터셋 (Wisconsin Breast Cancer Diagnostic dataset)
https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)

In [7]:

```
import requests

data = requests.get("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wiscons
dataset_path = os.path.join('data', 'wdbc.data')

with open(dataset_path, "w") as f:
  f.write(data.text)
```

2.2 데이터 파싱 (Q)

In [8]:

```
def parse_cancer_row(row: List[str]) -> LabeledPoint:
    measurements = [float(value) for value in row[2:]]
    label = row[1]
    return LabeledPoint(measurements, label)
# assertion error로 인하여 Restart & Run All 기능은 사용하지 않고 일일히 한개씩 run했습니다.
# e-class에 올라온 다른 학우들의 QnA에 저와 비슷한 error가 있기에 해결 될 경우 보고 고칠 예정 (20210
```

2.3 데이터 읽기

In [9]:

```
with open(dataset_path) as f:
    reader = csv.reader(f)
    cancer_data = [parse_cancer_row(row) for row in reader if row]
```

데이터 탐색 단계의 시각화를 위해 데이터 행렬 생성

In [10]:

```
columns = [
    "radius_mean", "texture_mean", "perimeter_mean", "area_mean", "smoothness_mean",
    "compactness_mean", "concavity_mean", "points_mean", "symmetry_mean", "dimension_mean",
    "radius_se", "texture_se", "perimeter_se", "area_se", "smoothness_se",
    "compactness_se", "concavity_se", "points_se", "symmetry_se", "dimension_se",
    "radius_worst", "texture_worst", "perimeter_worst", "area_worst", "smoothness_worst",
    "compactness_worst", "concavity_worst", "points_worst", "symmetry_worst", "dimension_worst",
    ]
```

In [11]:

```
from scratch.linear_algebra import get_column, shape
def make_matrix(dataset):
    matrix = []
    for datapoint in dataset:
        matrix.append(datapoint.point)
    return matrix
```

In [12]:

```
cancer_matrix = make_matrix(cancer_data)
# print(cancer_matrix[1])
print(shape(cancer_matrix))
```

```
(569, 30)
```

2.4 데이터 탐색

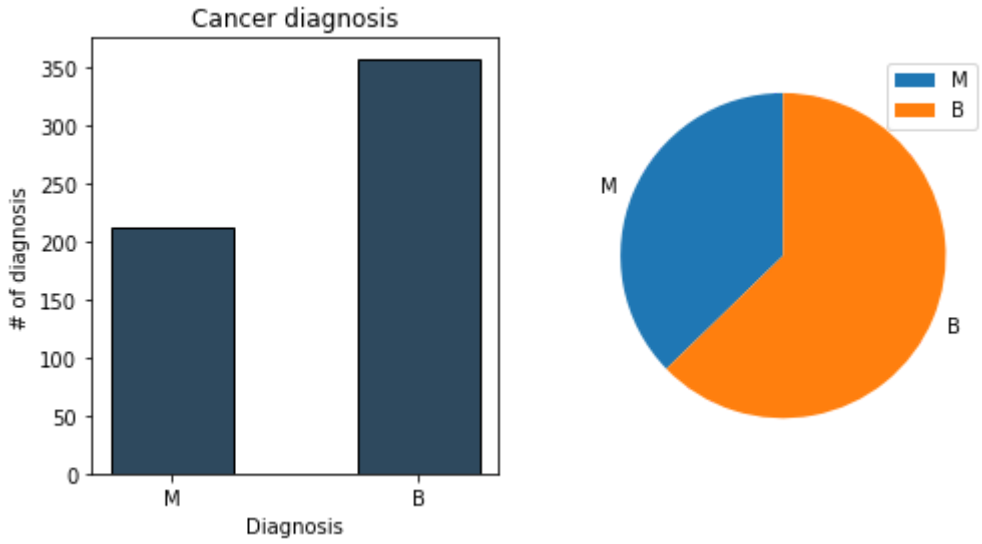
2.3.1 클래스 비율 확인

In [13]:

```
label_type = defaultdict(int)
for cancer in cancer_data:
    label_type[cancer.label] += 1
```

In [14]:

```
plt.figure(figsize=(8,4))
plt.subplot(1, 2, 1)
plt.bar(label_type.keys(),
        label_type.values(),
        0.5,
        facecolor="#2E495E",
        edgecolor=(0, 0, 0))                # Black edges for each bar

plt.xlabel("Diagnosis")
plt.ylabel("# of diagnosis")
plt.title("Cancer diagnosis")

plt.subplot(1, 2, 2)
pies = plt.pie(label_type.values(),
               labels=label_type.keys(),
               startangle=90)
plt.legend()
plt.show()
# 링크를 타고 들어가서 데이터 확인했습니다. 교수님께서 주신 pdf에는
# 막대그래프와 파이차트에 파라미터가 'B'가 먼저오고 'M'이 두번째로
# 그려지는데 이점은 assertion error로 인하여 계속 코드를 변경했지만
# 다른곳에서 오류가 데이터가 바뀌지지 않았습니다. (202103123)
# 데이터를 다시 확인후 처음 row[1]의 처음 데이터는 'M'으로 시작하고
# 마지막 행의 데이터는 'B' 끝나는것을 확인할 수 있었습니다.
# 데이터를 맨끝부터 읽으면 barplot에서 'B'부터 그려지지 않을까?
# 하는 생각에 구글링 해보았지만 찾을 수 없습니다.(~20210330)
```



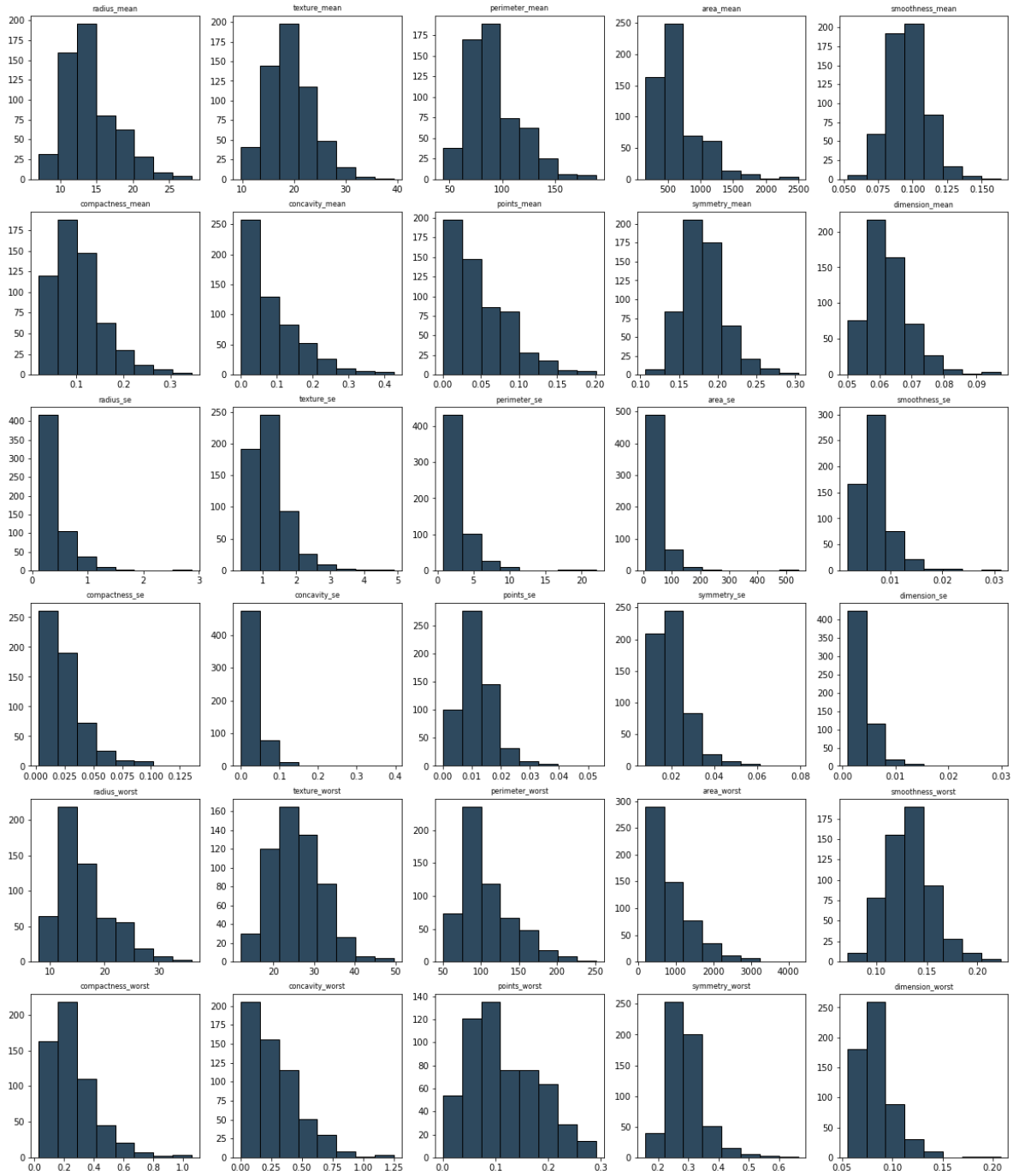
2.3.2 특징 별 히스토그램

In [15]:

```
def histogram(ax, col : int):

    n, bins, patches = ax.hist(get_column(cancer_matrix, col),
                               8,
                               facecolor="#2E495E",
                               edgecolor=(0, 0, 0))

    ax.set_title(columns[col], fontsize=8)
```

In [16]:

```
from matplotlib import pyplot as plt
num_rows = 6
num_cols = 5

fig, ax = plt.subplots(num_rows, num_cols, figsize=(num_cols*4, num_rows*4))
for row in range(num_rows):
    for col in range(num_cols):
        histogram(ax[row][col], num_cols * row + col)
plt.show()
```



2.3.3 특징 쌍 별 산포도

In [17]:

```
points_by_diagnosis: Dict[str, List[Vector]] = defaultdict(list)
for cancer in cancer_data:
    points_by_diagnosis[cancer.label].append(cancer.point)
```

In [18]:

```
start = 0
end = start + 10
pairs = [(i, j) for i in range(start, end) for j in range(i+1, end) if i < j]
print(pairs)
marks = ['+', '.']
```

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

In [19]:

```
from matplotlib import pyplot as plt
num_rows = 9
num_cols = 5

fig, ax = plt.subplots(num_rows, num_cols, figsize=(num_cols*3, num_rows*3))

for row in range(num_rows):
    for col in range(num_cols):
        i, j = pairs[num_cols * row + col]
        ax[row][col].set_title(f"{columns[i]} vs {columns[j]}", fontsize=8)
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])

        for mark, (diagnosis, points) in zip(marks, points_by_diagnosis.items()):
            xs = [point[i] for point in points]
            ys = [point[j] for point in points]
            ax[row][col].scatter(xs, ys, marker=mark, label=diagnosis)

ax[-1][-1].legend(loc='lower right', prop={'size': 6})
plt.show()
```


radius_mean vs texture_mean
radius_mean vs perimeter_mean
radius_mean vs area_mean
radius_mean vs smoothness_mean
radius_mean vs compactness_mean
radius_mean vs concavity_mean
radius_mean vs points_mean
radius_mean vs symmetry_mean
radius_mean vs dimension_mean
texture_mean vs perimeter_mean
texture_mean vs area_mean
texture_mean vs smoothness_mean
texture_mean vs compactness_mean
texture_mean vs concavity_mean
texture_mean vs points_mean
texture_mean vs symmetry_mean
texture_mean vs dimension_mean
perimeter_mean vs area_mean
perimeter_mean vs smoothness_mean
perimeter_mean vs compactness_mean
perimeter_mean vs concavity_mean
perimeter_mean vs points_mean
perimeter_mean vs symmetry_mean
perimeter_mean vs dimension_mean
area_mean vs smoothness_mean
area_mean vs compactness_mean
area_mean vs concavity_mean
area_mean vs points_mean
area_mean vs symmetry_mean
area_mean vs dimension_mean
smoothness_mean vs compactness_mean
smoothness_mean vs concavity_mean
smoothness_mean vs points_mean
smoothness_mean vs symmetry_mean
smoothness_mean vs dimension_mean
compactness_mean vs concavity_mean
compactness_mean vs points_mean
compactness_mean vs symmetry_mean
compactness_mean vs dimension_mean
concavity_mean vs points_mean
concavity_mean vs symmetry_mean
concavity_mean vs dimension_mean
points_mean vs symmetry_mean
points_mean vs dimension_mean
symmetry_mean vs dimension_mean
M
B

2.5 데이터셋 분리 (Q)

In [22]:

```
import random
from scratch.machine_learning import split_data

random.seed(12)
cancer_train, cancer_test = split_data(cancer_data, 0.70)
# assert len(cancer_train) == 0.7 * 569
# assert len(cancer_test) == 0.3 * 569
# print(len(cancer_train),len(cancer_test))
# print(cancer_test[32])
```

2.6 데이터 표준화 (Standardization)

In [26]:

```
from scratch.working_with_data import scale, rescale

def normalization(dataset):
    return rescale(make_matrix(dataset))
```

In [27]:

```
cancer_train_matrix = normalization(cancer_train)
cancer_test_matrix = normalization(cancer_test)
```

2.7 예측 (Q)

In [28]:

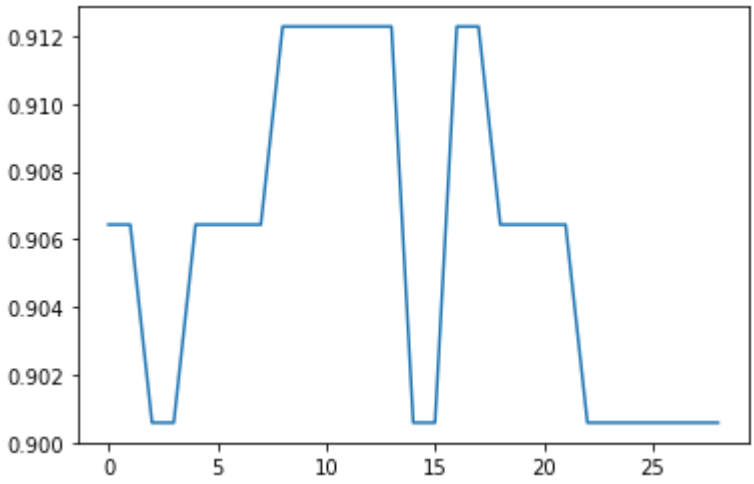
```
from typing import Tuple
#
confusion_matrix: Dict[Tuple[str, str], int] = defaultdict(int)
def prediction(k : int) -> Tuple[float, Dict[Tuple[str, str], int]]:
    num_correct = 0
    for cancer in cancer_test:
        predicted = knn_classify(k, cancer_train, cancer.point)
        actual = cancer.label

        if predicted == actual:
            num_correct +=1

        confusion_matrix[(predicted, actual)] += 1

    pct_correct = num_correct / len(cancer_test)

    return pct_correct, confusion_matrix
```

2.8 엘보 방법 (Elbow method)으로 k 선정 (Q)

In [29]:

```
k_candidate = (k for k in range(1, 30))
optimal_k = 0

acc_list : List[float] = []
for k in k_candidate:
    accuracy, confusion_matrix = prediction(k)
    acc_list.append(accuracy)

optimal_k = acc_list.index(max(acc_list)) + 1  # k는 1부터 시작하므로 최대값의
                                               # 인덱스에 더하기 1함

# print(acc_list)
print("")
print("Optimal k = ", optimal_k)
plt.plot(acc_list)
plt.show()
```

```
Optimal k =  9
```



In [ ]:

```
```

13장. 나이브 베이즈 (NaiveBayes) 과제

1. 데이터 읽기

1.1 SpamAssassin 데이터셋 다운 로드

In [1]:

```
from io import BytesIO
import requests
import tarfile
import os
from scratch.machine_learning import split_data

BASE_URL = "https://spamassassin.apache.org/old/publiccorpus/"
FILES = ["20021010_easy_ham.tar.bz2",
         "20021010_hard_ham.tar.bz2",
         "20021010_spam.tar.bz2"]
OUTPUT_DIR = 'spam_data'

if os.path.exists(OUTPUT_DIR) is False:
    for filename in FILES:
        content = requests.get(f"{BASE_URL}/{filename}").content

        fin = BytesIO(content)

        with tarfile.open(fileobj=fin, mode='r:bz2') as tf:
            tf.extractall(OUTPUT_DIR)
```

1.2 메세지 클래스 정의

In [2]:

```
from typing import NamedTuple

class Message(NamedTuple):
    text: str
    is_spam: bool
```

1.3 이메일 본문 디코딩

In [3]:

```
from email.parser import Parser

def decode_email(msg_str):
    p = Parser()
    message = p.parsestr(msg_str)
    decoded_message = ''
    for part in message.walk():
        if part.get_content_type() not in ('text/plain', 'text/html'): continue

        charset = part.get_content_charset()
        part_str = part.get_payload(decode=1)
        try:
            decoded_message += part_str.decode(charset)
        except:
            decoded_message += str(part_str)

    return decoded_message
```

1.3 이메일 텍스트 읽기 (Q)

이메일의 제목과 본문 텍스트를 읽어서 Message 타입을 만들고 데이터 리스트를 생성하시오.

In [4]:

```
import glob
import email
from typing import List
import re

def read_emails(include_body : bool = False) -> List[Message]:
    # modify the path to wherever you've put the files
    path = 'spam_data/*/*'

    data: List[Message] = []
    # glob.glob returns every filename that matches the wildcarded path
    for filename in glob.glob(path):
        is_spam = "ham" not in filename
        # is_spam = "spam" in filename
        # There are some garbage characters in the emails, the errors='ignore'
        # skips them instead of raising an exception.
        with open(filename, errors='ignore') as email_file:
            raw_email = email_file.read()
            msgobj = email.message_from_string(raw_email)
            message = msgobj['Subject'] or ""
            message = message.lower()
            message = message.replace("₩n","")

            body = decode_email(raw_email)
            body1 = body.replace("₩n","")
            bdoy2 = body1.strip('₩n')
            # body = body.lower()
            result = message + body
            # data.append(Message(message, is_spam))

            # data.append(Message(body, is_spam))
            data.append(Message(result, is_spam))


    return data
```

In [5]:

```
include_body = True
data = read_emails(include_body= True)
print("읽은 email 개수 :", len(data))

# print(type(data))
# print(data[30:35])
```

읽은 email 개수 : 3302

2. NLTK 설치 및 테스트

2.1 NLTK설치 및 리소스 다운로드

In [6]:

```
! pip install nltk
```

```
Requirement already satisfied: nltk in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib₩s
ite-packages (3.5)
Requirement already satisfied: click in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib
₩site-packages (from nltk) (7.1.2)
Requirement already satisfied: regex in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib
₩site-packages (from nltk) (2020.10.15)
Requirement already satisfied: tqdm in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib₩s
ite-packages (from nltk) (4.50.2)
Requirement already satisfied: joblib in c:₩users₩이준용₩.conda₩envs₩data_mining₩lib
₩site-packages (from nltk) (0.17.0)
```

In [7]:

```
import nltk
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml (ht
tps://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[7]:

```
True
```

2.1 단어 토큰화 및 어간 추출 테스트

단어 토큰화

In [8]:

```
from nltk.tokenize import TreebankWordTokenizer
text="This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all th
tokenizer=TreebankWordTokenizer()
words = tokenizer.tokenize(text)
print("tokens : ", words)
```

```
tokens :  ['This', 'was', 'not', 'the', 'map', 'we', 'found', 'in', 'Billy', "Bone
s's", 'chest', ',', 'but', 'an', 'accurate', 'copy', ',', 'complete', 'in', 'all',
'things', '--', 'names', 'and', 'heights', 'and', 'soundings', '--', 'with', 'the',
'single', 'exception', 'of', 'the', 'red', 'crosses', 'and', 'the', 'written', 'note
s', '.']
```

불용어

In [9]:

```
from nltk.corpus import stopwords
stopwords_set = set(stopwords.words('english'))
print("stopwords : ", stopwords_set)
```

```
stopwords :  {'doing', 'after', 'are', "doesn't", 'was', 'most', 'being', 'i', 'by',
'out', 't', 'hasn', 'his', 'once', "you'd", 'between', 'such', 'which', 'm', "were
n't", 'in', 'for', "don't", 'ours', 'those', 'y', "she's", 'll', 'isn', 'only', 'mig
htn', 'my', 'if', 'needn', 'through', 'shan', 'you', 'whom', "aren't", 'do', "have
n't", 'each', 'too', 'nor', 'myself', 'from', 'then', 'him', 'what', 've', 'more',
"mightn't", 'haven', 'ourselves', 'that', 'these', 'up', 'me', 'as', 'further', 'no
w', "that'll", 'didn', "you're", 'does', 'couldn', 'under', 'and', 'this', 'not', 'i
ts', 'because', 'herself', 'same', 'than', 'have', 'an', 'ma', 'she', 'some', 'unti
l', 'has', 'having', 'own', "needn't", 'be', 'before', 'am', 'he', 'other', "shoul
d've", 'above', 'the', "you'll", 'again', 'both', 'a', 'there', "won't", 'o', "sha
n't", 'yours', 'our', "hadn't", 'to', 'is', 's', 'about', 'so', "wasn't", 'their',
"you've", 'but', 'wouldn', 'did', 'off', 'should', 'will', 'mustn', "wouldn't", 'whe
n', 'won', 'where', 'can', 'shouldn', 'how', 'theirs', 'it', 'over', 'we', 'itself',
'at', 'few', 'any', "didn't", 'who', 'very', 'of', 'down', "it's", 'd', 'they', 'you
rself', 'during', "hasn't", 'below', 'doesn', "mustn't", 'hers', 'with', 'just', 'ai
n', 'here', 'into', 'aren', 'her', "isn't", "shouldn't", 'yourselves', 'why', 'on',
'weren', 'against', 'don', 'were', 'themselves', 're', 'all', "couldn't", 'been', 'h
imself', 'them', 'your', 'had', 'while', 'wasn', 'no', 'hadn', 'or'}
```

어간 추출

In [10]:

```
from nltk.stem import PorterStemmer
s = PorterStemmer()
stem_words = [s.stem(w) for w in words]
print("stem tokens : ", len(stem_words), stem_words)
print("")
stem_words_wo_stopwords = [s.stem(w) for w in words if w not in stopwords_set]
print("stem tokens without stopwords: ",
      len(stem_words_wo_stopwords),
      stem_words_wo_stopwords)
```

```
stem tokens :  41 ['thi', 'wa', 'not', 'the', 'map', 'we', 'found', 'in', 'billi',
"bones'", 'chest', ',', 'but', 'an', 'accur', 'copi', ',', 'complet', 'in', 'all',
'thing', '--', 'name', 'and', 'height', 'and', 'sound', '--', 'with', 'the', 'sing
l', 'except', 'of', 'the', 'red', 'cross', 'and', 'the', 'written', 'note', '.']

stem tokens without stopwords:  24 ['thi', 'map', 'found', 'billi', "bones'", 'ches
t', ',', 'accur', 'copi', ',', 'complet', 'thing', '--', 'name', 'height', 'sound',
'--', 'singl', 'except', 'red', 'cross', 'written', 'note', '.']
```

2.3 토큰화 함수 (Q)

Treebank Tokenizer를 사용해서 어간 추출 및 불용어 제거를 해보자.

In [11]:

```
from nltk.tokenize import TreebankWordTokenizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from typing import Set

def tokenize(text: str) -> Set[str]:
    tokenizer=TreebankWordTokenizer()
    text = tokenizer.tokenize(text)
    stop_words = set(stopwords.words('english'))
    result = []
    for w in text:
        w = w.lower()
        if w not in stop_words:
            if len(w) > 2:
                result.append(w)
    s = PorterStemmer()
    resultword = [s.stem(w) for w in result if w not in stopwords_set]
    words = resultword

    return set(words)                       # remove duplicates.
print(len(tokenize(str(data))))
# print(tokenize(str(data[0:5])))
```

107608

3. 나이브 베이즈 분류기

3.1 NaiveBayesClassifier (Q)

단어의 최소 빈도수를 설정해서 그 이하로 나오는 단어는 무시하도록 _thresholding_tokens 함수 작성하시오

In [12]:

```
from typing import List, Tuple, Dict, Iterable
import math
from collections import defaultdict
import matplotlib.pyplot as plt

class NaiveBayesClassifier:
    def __init__(self, k: float = 0.5) -> None:
        self.k = k  # smoothing factor

        self.tokens: Set[str] = set()
        self.token_spam_counts: Dict[str, int] = defaultdict(int)
        self.token_ham_counts: Dict[str, int] = defaultdict(int)
        self.spam_messages = self.ham_messages = 0

    def train(self,
              messages: Iterable[Message],
              threshold: int = 0,
              verbos : bool = True) -> None:
        self._count_tokens(messages)
        del_spam_count, del_ham_count = self._thresholding_tokens(messages, threshold)

        if verbos :
            print(del_spam_count, "tokens are deleted in spams ")
            print(del_ham_count, "tokens are deleted in hams ")
            print("spam ", self.spam_messages)
            print("ham ", self.ham_messages)
            print("token", len(self.tokens))
            print("spam token", len(self.token_spam_counts))
            print("ham token", len(self.token_ham_counts))

            print("======= token probabilities ======= ")
            self.print_token_probilities()

    def _count_tokens(self, messages: Iterable[Message]) -> None:
        for message in messages:
            # print(message)
            # Increment message counts
            if message.is_spam:
                self.spam_messages += 1
            else:
                self.ham_messages += 1

            # Increment word counts
            for token in tokenize(message.text):
                if message.is_spam:
                    self.token_spam_counts[token] += 1
                else:
                    self.token_ham_counts[token] += 1
    #    print(len(self.token_spam_counts.items()),len(self.token_ham_counts.items()))
    #    print(self.spam_messages)
      # **********************최소 빈도수 이하 토큰 삭제*****************

    def _thresholding_tokens(self,
                             messages: Iterable[Message],
                             threshold: int = 0) -> Tuple[int, int]:

        self.tokens = [w for w, c in self.token_spam_counts.items()]
        self.tokens = [w for w, c in self.token_ham_counts.items()]
        # print(self.token_spam_counts.items())
```

```
        del_spam_count = len([w for w,c in self.token_spam_counts.items() if c <= 5])
        del_ham_count = len([w for w,c in self.token_ham_counts.items() if c <= 5])


        return del_spam_count, del_ham_count

    def print_token_probilities(self, count=10):
        for token in self.tokens:
            p_token_spam, p_token_ham = self._probabilities(token)
            print(token, "(spam:", p_token_spam, "ham:", p_token_ham, ")")
            count -= 1
            if count == 0 : return

    def token_histogram(self):
        plt.figure(figsize=(15,8))
        plt.subplot(2, 1, 1)
        n, bins, patches = plt.hist(self.token_spam_counts.values(),
                                    200,
                                    facecolor="#2E495E",
                                    edgecolor=(0, 0, 0))
        plt.title("Spam words")
        plt.xlabel("")
        plt.ylabel("Word Count")

        plt.subplot(2, 1, 2)
        n, bins, patches = plt.hist(self.token_ham_counts.values(),
                                    200,
                                    facecolor="#2E495E",
                                    edgecolor=(0, 0, 0))
        plt.title("Ham words")
        plt.xlabel("")
        plt.ylabel("Word Count")

        plt.show()

    def _probabilities(self, token: str) -> Tuple[float, float]:
        """returns P(token | spam) and P(token | not spam)"""
        spam = self.token_spam_counts[token]
        ham = self.token_ham_counts[token]

        p_token_spam = (spam + self.k) / (self.spam_messages + 2 * self.k)
        p_token_ham = (ham + self.k) / (self.ham_messages + 2 * self.k)

        return p_token_spam, p_token_ham

    def token_histogram(self):
        plt.figure(figsize=(15,8))
        plt.subplot(2, 1, 1)
        n, bins, patches = plt.hist(self.token_spam_counts.values(),
                                    200,
                                    facecolor="#2E495E",
                                    edgecolor=(0, 0, 0))
        plt.title("Spam words")
        plt.xlabel("")
        plt.ylabel("Word Count")

        plt.subplot(2, 1, 2)
        n, bins, patches = plt.hist(self.token_ham_counts.values(),
                                    200,
```

```
                                    facecolor="#2E495E",
                                    edgecolor=(0, 0, 0))
        plt.title("Ham words")
        plt.xlabel("")
        plt.ylabel("Word Count")

        plt.show()

    def predict(self, text: str) -> float:
        text_tokens = tokenize(text)
        log_prob_if_spam = log_prob_if_ham = 0.0

        # Iterate through each word in our vocabulary.
        for token in self.tokens:
            prob_if_spam, prob_if_ham = self._probabilities(token)

            # If *token* appears in the message,
            # add the log probability of seeing it;
            if token in text_tokens:
                log_prob_if_spam += math.log(prob_if_spam)
                log_prob_if_ham += math.log(prob_if_ham)

            # otherwise add the log probability of _not_ seeing it
            # which is log(1 - probability of seeing it)
            else:
                log_prob_if_spam += math.log(1.0 - prob_if_spam)
                log_prob_if_ham += math.log(1.0 - prob_if_ham)

        prob_if_spam = math.exp(log_prob_if_spam)
        prob_if_ham = math.exp(log_prob_if_ham)
        try :
            posterior = prob_if_spam / (prob_if_spam + prob_if_ham)
        except ZeroDivisionError:
            posterior = 0

        return posterior
```

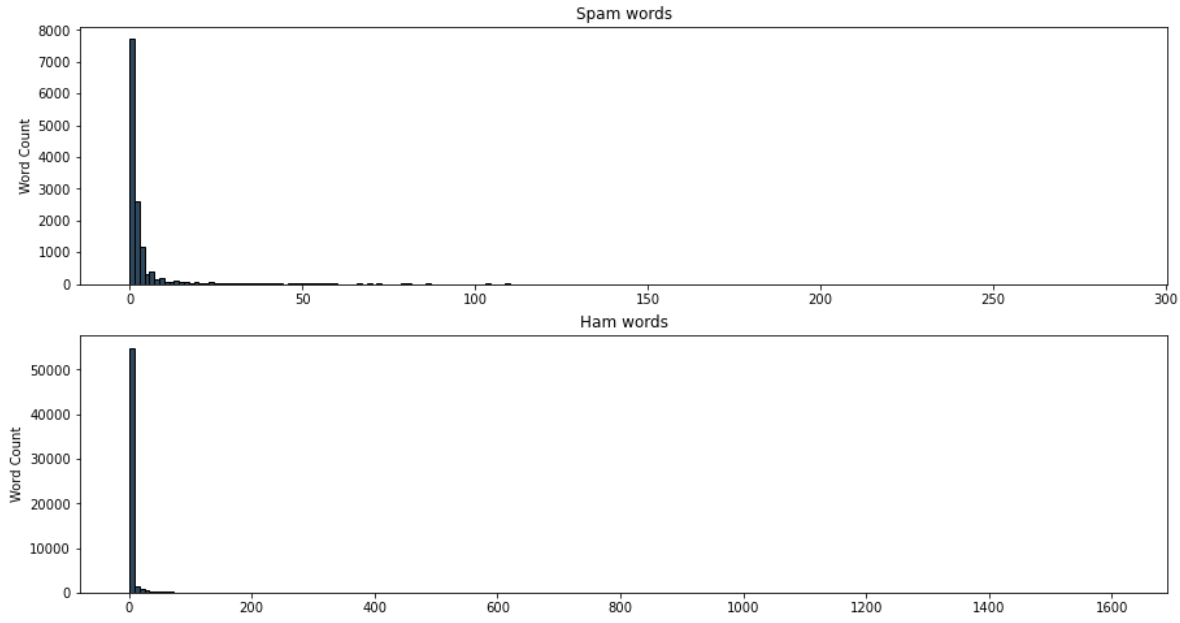
3.2 모델 훈련

In [13]:

```
import random
from scratch.machine_learning import split_data
                                    split data into fractions [prob, 1-prob]
random.seed(0)      # just so you get the same answers as me
train_messages, test_messages = split_data(data, 0.75)
model = NaiveBayesClassifier()

model.train(train_messages,15)
model.token_histogram()
```

def split_data(data: List[x], prob: float)
→ Tuple[List[x], List[x]]:
data = data[:] make a shallow copy
random.shuffle(data) because shuffle modifies the list
cut = int(len(data) * prob) use prob to find a cutoff
return data[:cut], data[cut:] and split the shuffled list there

```
11800 tokens are deleted in spams
52719 tokens are deleted in hams
spam  371
ham  2105
token 58928
spam token 13513
ham token 58928
======= token probabilities =======
seem (spam: 0.028225806451612902 ham: 0.14506172839506173 )
sell (spam: 0.10080645161290322 ham: 0.07003798670465337 )
follow (spam: 0.18413978494623656 ham: 0.08000949667616335 )
juli (spam: 0.006720430107526882 ham: 0.04249762583095917 )
noth (spam: 0.05779569892473118 ham: 0.057692307692307696 )
said (spam: 0.04973118279569892 ham: 0.13509021842355176 )
evolv (spam: 0.0013440860215053765 ham: 0.011158594491927826 )
sinc (spam: 0.06854838709677419 ham: 0.11419753086419752 )
'thou (spam: 0.0013440860215053765 ham: 0.0011870845204178537 )
speaker (spam: 0.004032258064516129 ham: 0.0097340930674264 )
```



3.3 예측 및 성능 평가

예측

In [14]:

```
from collections import Counter

predictions = [(message, model.predict(message.text))
               for message in test_messages]
```

```
def countLetters(word):
    counter = {}
    for letter in word:
        if letter not in counter:
            counter[letter] = 0
        counter[letter] += 1
    return counter
countLetters('hello world')
{'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}
```

혼동 행렬

In [15]:

```
# Assume that spam_probability > 0.5 corresponds to spam prediction
# and count the combinations of (actual is_spam, predicted is_spam)
confusion_matrix = Counter((message.is_spam, spam_probability > 0.5)
                           for message, spam_probability in predictions)

print(confusion_matrix)
```

```
Counter({(False, False): 694, (True, True): 70, (True, False): 60, (False, True):
2})
```

정확도, 정밀도, 재현율 F1점수 (Q)

혼동 행렬 결과를 이용해서 정확도, 정밀도, 재현율 F1점수를 계산해 보시오.

In [16]:

```
from scratch.machine_learning import accuracy, precision, recall, f1_score
tp = confusion_matrix.get((True, True))
tn = confusion_matrix.get((False,False))
fp = confusion_matrix.get((True, False))
fn = confusion_matrix.get((False, True))

print("accuracy :",accuracy(tp,fp,fn,tn))
print("precision :",precision(tp,fp,fn,tn))
print("recall :", recall(tp,fp,fn,tn))
print("f1_score :", f1_score(tp,fp,fn,tn))
```

```
accuracy : 0.9249394673123487
precision : 0.5384615384615384
recall : 0.9722222222222222
f1_score : 0.693069306930693
```

3.4 스팸과 햄을 대표하는 단어 확인

In [17]:

```
def p_spam_given_token(token: str, model: NaiveBayesClassifier) -> float:
    # We probably shouldn't call private methods, but it's for a good cause.
    prob_if_spam, prob_if_ham = model._probabilities(token)

    return prob_if_spam / (prob_if_spam + prob_if_ham)

words = sorted(model.tokens, key=lambda t: p_spam_given_token(t, model))

print("spammiest_words", words[-10:])
print("hammiest_words", words[:10])
```

```
spammiest_words ['bordercolordark=', 'transaction.', '111111', 'mailings.', '00006
6', 'opt-out', '000080', "type='hidden", 'content-languag', 'border-collaps']
hammiest_words ['wrote', 'freshrpms.net', 'rpm-list', '//lists.freshrpms.net/mailma
n/listinfo/rpm-list', 'aug', 'rpm', 'satalk', 'matthia', 'spamassassin', 'spambay']
```

In [ ]:

```
혼동행렬.

                실제
                positive        Negative
예측  Positive    True positive   False Negative
      Negative    False positive  True Negative

정확도 ACC = (TP + TN) / (TP + TN + FP + FN)   데이터가 불균형할때 잘못된 지표로 사용될 가능성 있음.

정밀도 precision : 모델이 True라고 예측한 것 중 실제로 True인 비율 → 혼동행렬 지표.   tp / (tp + fn)

재현율 recall : 실제 데이터가 True 라고 예측한 것 중에서 모델이 True 라고 예측한 비율. → 민감도 sensitivity 라고도함 혼동행렬의 지표.   tp / (tp + fn)

F1-Score : 재현율과 정밀도의 중요성이 같다고 가정하고, 두 지표의 조화평균으로 만들어진 지표
F1 = 2 * (precision * recall) / (precision + recall)

특이도 : 모델이 false 라고 예측한 것들 중에 실제로 false인 비율.
specifity = TN / (FP + TN)
```