- 구매자로부터 가장 사고싶은 과일 사진을 보내면
- 구매자가 가장 많이 요청하는 과일을 판매 목록 선정
- 사람들이 과일 사진을 너무 많이 보내줬는데, 이걸 하나하나 무슨 과일인지 체크할 사람이 없음

> 데이터: 사람들이 보낸 과일 사진, 정답은 따로 X

> 정답 X => 분류 모델 사용할 수 없음

> 클러스터링(군집) => 비지도학습(정답 X) -> 보험사에서 클러스터링 모델 많이 사용

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
```

1. 데이터 로드

In [2]:

```
fruits = np.load('fruits_300.npy')
```

In [3]:

```
# 총 300장 사진, 사진 1장 당 100*100 픽셀->행과 열이 100개로 쪼개져있음
print(fruits.shape)
```

```
(300, 100, 100)
```

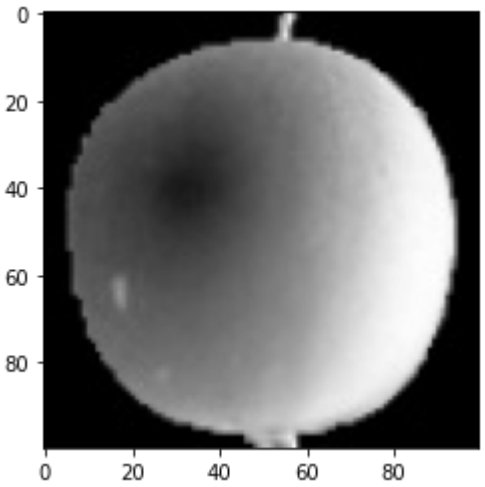
In [4]:

```
# 흑백사진 0~255의 정수값
print(fruits[0,0,:])
```

```
[  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2
1
   2   2   2   2   2   2   1   1   1   1   1   1   1   1   2   3   2
1
   2   1   1   1   1   2   1   3   2   1   3   1   4   1   2   5   5
5
  19 148 192 117  28   1   1   2   1   4   1   1   3   1   1   1   1
1
   2   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
1
   1   1   1   1   1   1   1   1   1   1]
```

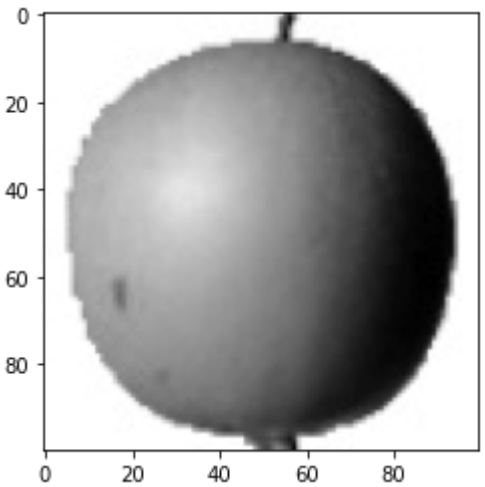
In [5]:

```
# 사과사진
# 숫자가 0에 가까울수록 검게 나타남
plt.imshow(fruits[0], cmap='gray')
plt.show()
```



In [6]:

```
plt.imshow(fruits[0], cmap='gray_r')  #r=reverse, 반전시킴
plt.show()  #배경이 흰색이면 배경에 집중이 됨 때문에 배경이 검정으로되게 해야함 -맞나??
```

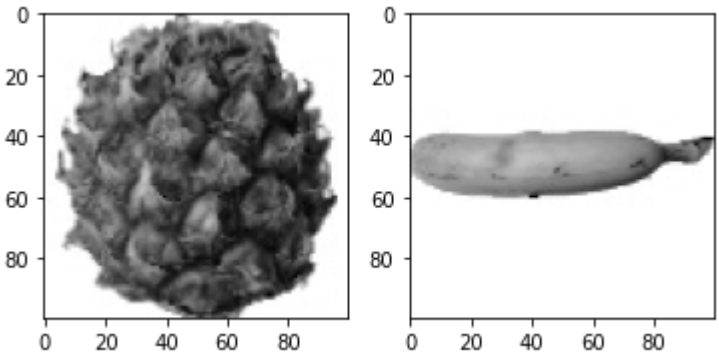


In [7]:

```
#파인애플, 바나나
fig, axs = plt.subplots(1,2)
axs[0].imshow(fruits[100], cmap='gray_r')
axs[1].imshow(fruits[200], cmap='gray_r')
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x7fad92bab3d0>
```



분류(Classification) 모델

- 지도학습(정답 필요) => 카테고리 명확!!
- 결정트리 -> 랜덤 포레스트

군집(Clustering) 모델

- 비지도학습(정답 불필요) => 무엇을 분류해야할지 모름!
- k-평균 클러스터링

2. 픽셀값 분석하기

In [8]:

```
# 픽셀 평균값 계산해보기! - 사과사진 100장을 겹쳤을 때의 z축의 평균값
# 픽셀 평균값 계산위해 100*100 => 10000으로 변경

apple = fruits[0:100].reshape(-1, 100*100)
pineapple = fruits[100:200].reshape(-1, 100*100)
banana = fruits[200:300].reshape(-1, 100*100)
```

In [9]:

```
#(100,100,100) => (100,10000)
print(apple.shape)
```

```
(100, 10000)
```

In [10]:

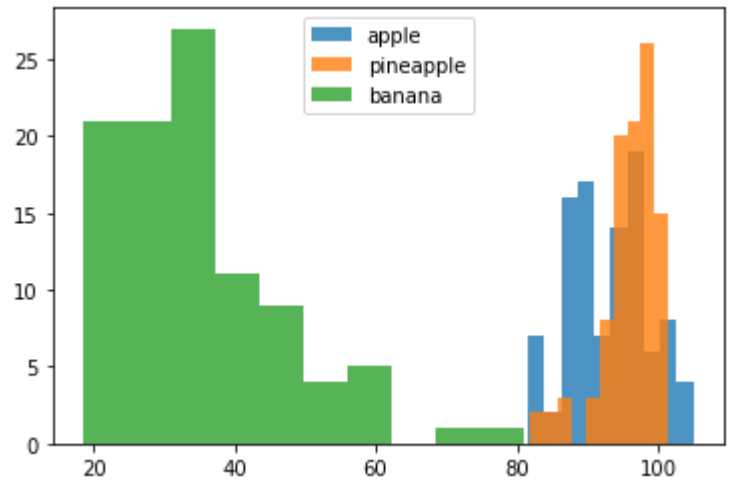
```
# 사과 샘플 100개의 평균값 계산
print(apple.mean(axis=1))
```

```
[ 88.3346  97.9249  87.3709  98.3703  92.8705  82.6439  94.4244  95.59
99
  90.681   81.6226  87.0578  95.0745  93.8416  87.017   97.5078  87.20
19
  88.9827 100.9158  92.7823 100.9184 104.9854  88.674   99.5643  97.24
95
  94.1179  92.1935  95.1671  93.3322 102.8967  94.6695  90.5285  89.07
44
  97.7641  97.2938 100.7564  90.5236 100.2542  85.8452  96.4615  97.14
92
  90.711  102.3193  87.1629  89.8751  86.7327  86.3991  95.2865  89.17
09
  96.8163  91.6604  96.1065  99.6829  94.9718  87.4812  89.2596  89.52
68
  93.799   97.3983  87.151   97.825  103.22    94.4239  83.6657  83.51
59
 102.8453  87.0379  91.2742 100.4848  93.8388  90.8568  97.4616  97.50
22
  82.446   87.1789  96.9206  90.3135  90.565   97.6538  98.0919  93.62
52
  87.3867  84.7073  89.1135  86.7646  88.7301  86.643   96.7323  97.26
04
  81.9424  87.1687  97.2066  83.4712  95.9781  91.8096  98.4086 100.78
23
 101.556  100.7027  91.6098  88.8976]
```

In [11]:

```
# 바나나는 평균값 40 아래 집중
# 사과와 파인애플은 평균값 90~100 사이에 집중되어 있다.
# 바나나는 평균값만 보고 구별 가능(왜? - 바나나 사진이 차지하는 비율이 작음(흰 공간이 많음))
# 사과와 파인애플은 동그랗고 사진에서 차지하는 영역이 비슷함(구별이 잘 안됨)

plt.hist(np.mean(apple, axis=1), alpha=0.8)  #hist = 히스토그램을 그리는 명령어, alpha 값은
plt.hist(np.mean(pineapple, axis=1), alpha=0.8)
plt.hist(np.mean(banana, axis=1), alpha=0.8)
plt.legend(['apple', 'pineapple', 'banana'])  # 그래프에 범례 추가하기
plt.show()
```

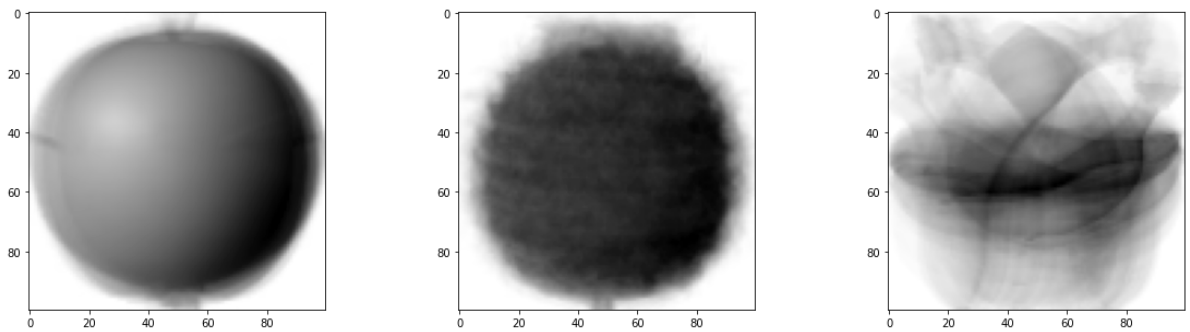


In [15]:

```
### 100*100 으로 shape을 변경하고 평균값을 이미지로 출력

apple_mean = np.mean(apple, axis=0).reshape(100,100)
pineapple_mean = np.mean(pineapple, axis=0).reshape(100,100)
banana_mean = np.mean(banana, axis=0).reshape(100,100)

fig, axs = plt.subplots(1,3,figsize=(20,5))
axs[0].imshow(apple_mean, cmap='gray_r')
axs[1].imshow(pineapple_mean, cmap='gray_r')
axs[2].imshow(banana_mean, cmap='gray_r')
plt.show()
```



3. K-Means 클러스터링

- 비지도학습(정답 X)
- 패턴을 분석하여 비슷한 것들끼리 군집을 묶음
- 동작

1. 무작위로 K개의 클러스터 중심점(센트로이드) 설정
2. 각 샘플에서 가장 가까운 클러스터 중심을 찾아 해당 클러스터 샘플로 지정
3. 클러스터에 속한 샘플의 평균값으로 클러스터 중심 변경
4. 클러스터 중심에 변화가 없을 때까지 [2,3]번 반복

4주차 1 자료에 관련 내용 있음

값의 평균 값으로 중심점을 이동하는 것

In [17]:

```
fruits = np.load('fruits_300.npy')   #(300,100,100)
fruits_2d = fruits.reshape(-1, 100*100)  #(300,10000)
```

In [22]:

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=3, random_state=42)
km.fit(fruits_2d)
```

Out[22]:

```
KMeans(n_clusters=3, random_state=42)
```

In [24]:

```
print(km.labels_)
```

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

In [25]:

```
print(np.unique(km.labels_, return_counts=True))  #n개가 잘 묶였는지,
```

```
(array([0, 1, 2], dtype=int32), array([111,  98,  91]))
```

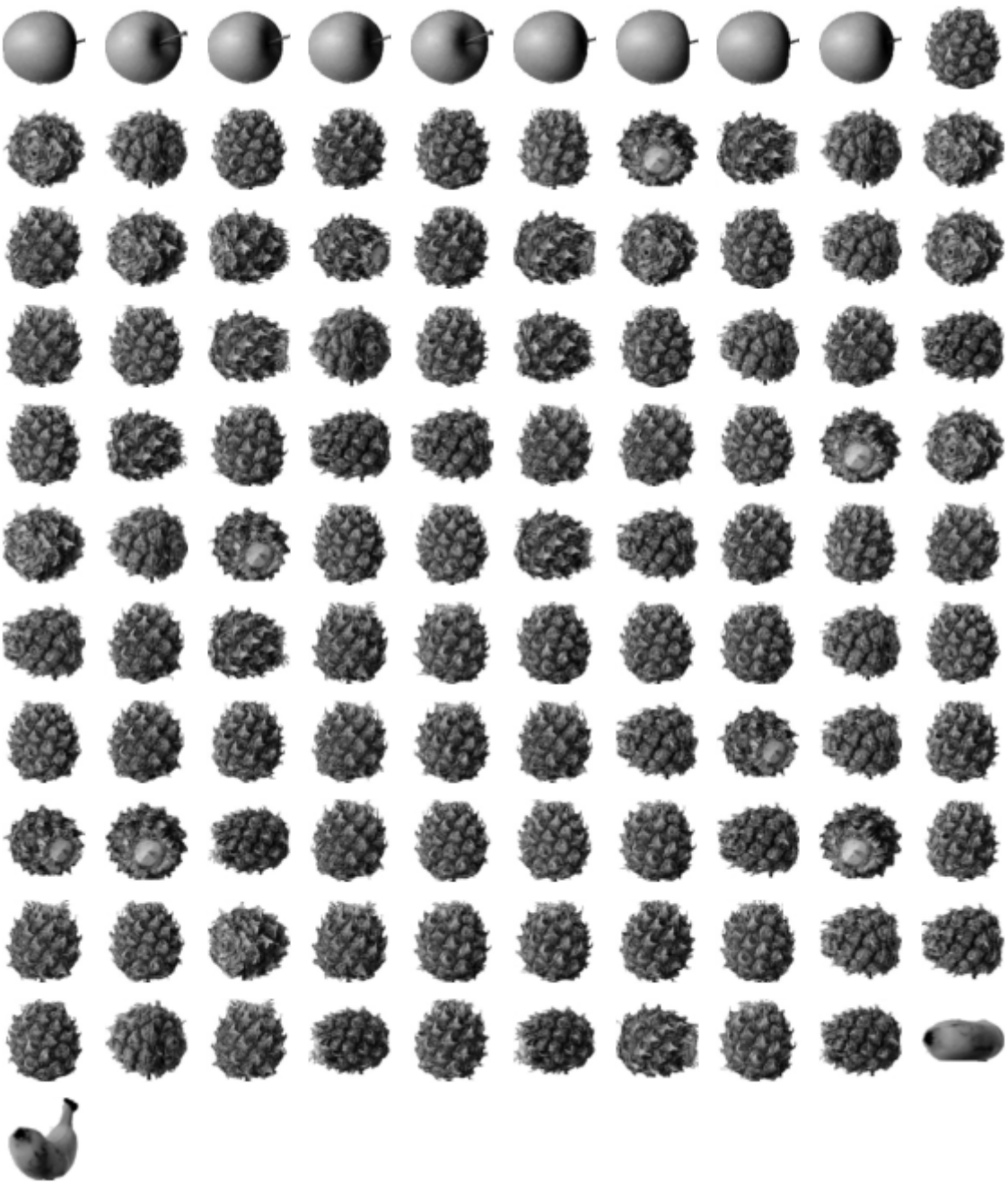
In [31]:

```
import matplotlib.pyplot as plt

def draw_fruits(arr, ratio=1):
    n = len(arr)
    rows = int(np.ceil(n/10))
    cols = n if rows < 2 else 10
    fig, axs = plt.subplots(rows, cols, figsize=(cols*ratio, rows*ratio), squeeze=Fa
    for i in range(rows):
        for j in range(cols):
            if i*10+j < n:
                axs[i,j].imshow(arr[i*10+j], cmap='gray_r')
            axs[i,j].axis('off')
    plt.show()
```

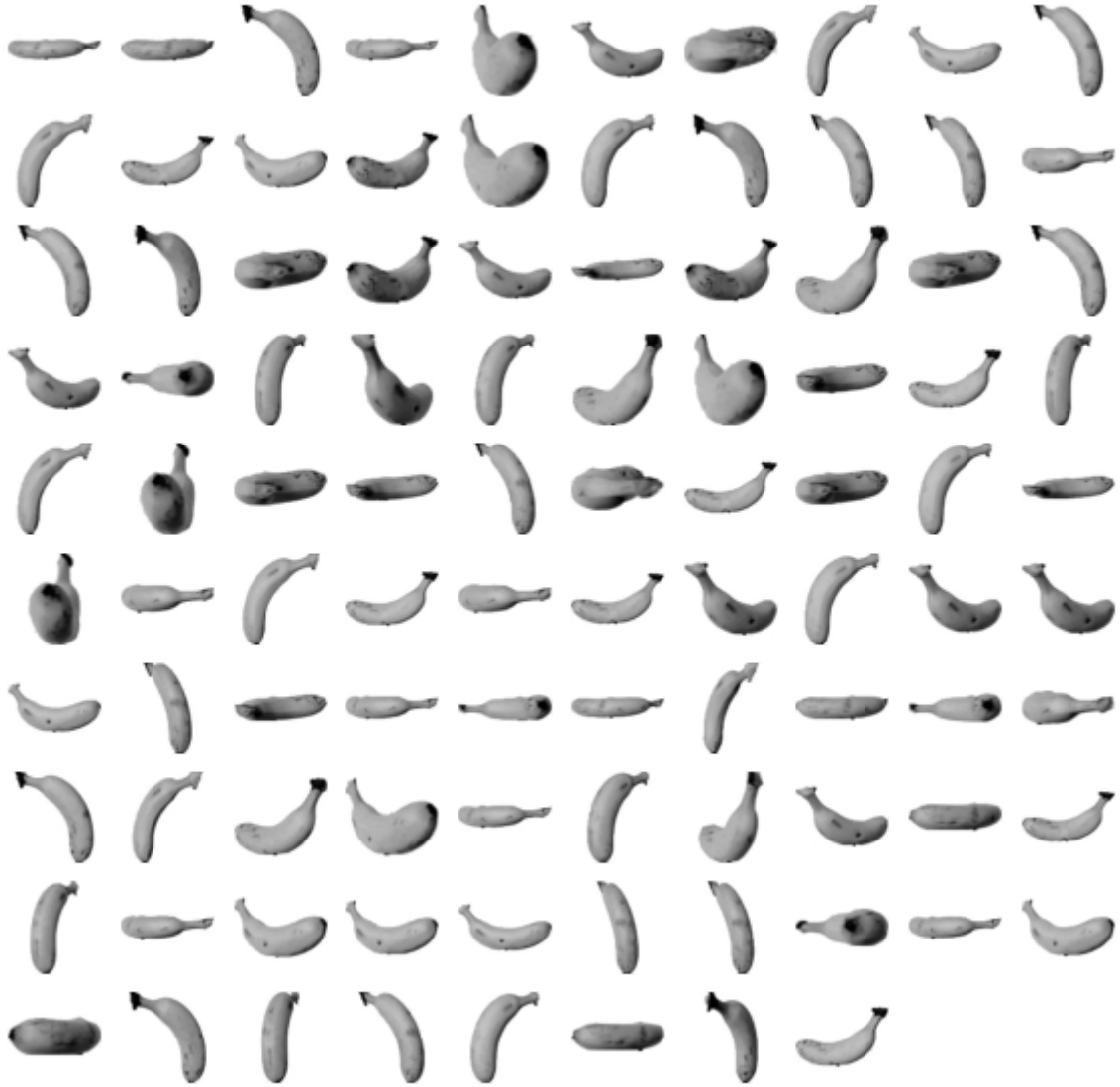
In [36]:

```
draw_fruits(fruits[km.labels_==0])  #1번 그룹, 컴퓨터와 코드에 따라 다 다르게 나옴 현재 파인애플+
```



In [38]:

```
draw_fruits(fruits[km.labels_==1])  #2번 그룹, 바나나 분류
```



In [35]:

```
draw_fruits(fruits[km.labels_==2])  #3번 그룹, 사과분류
```