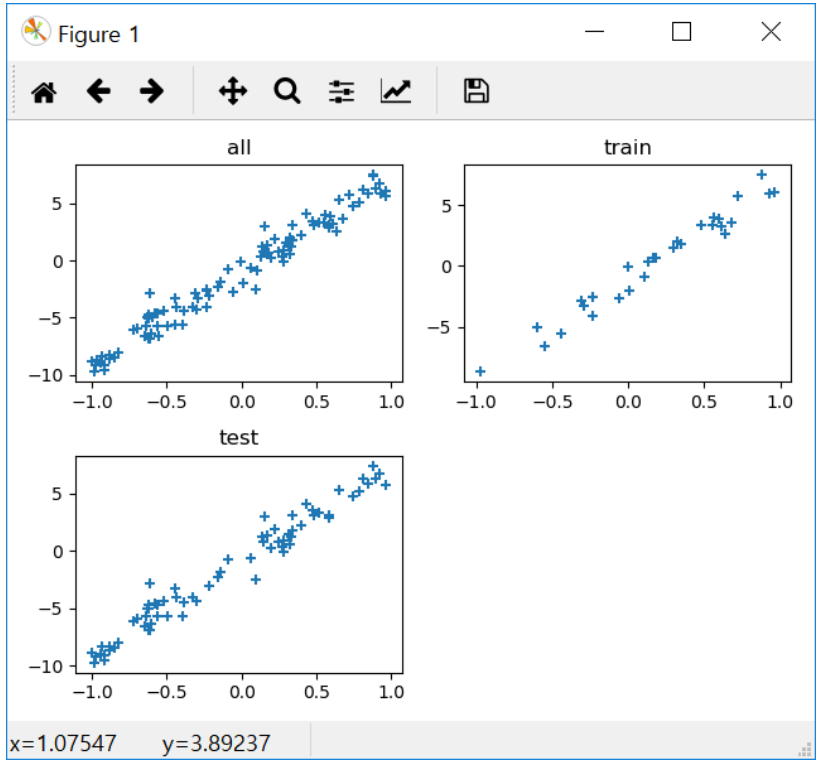
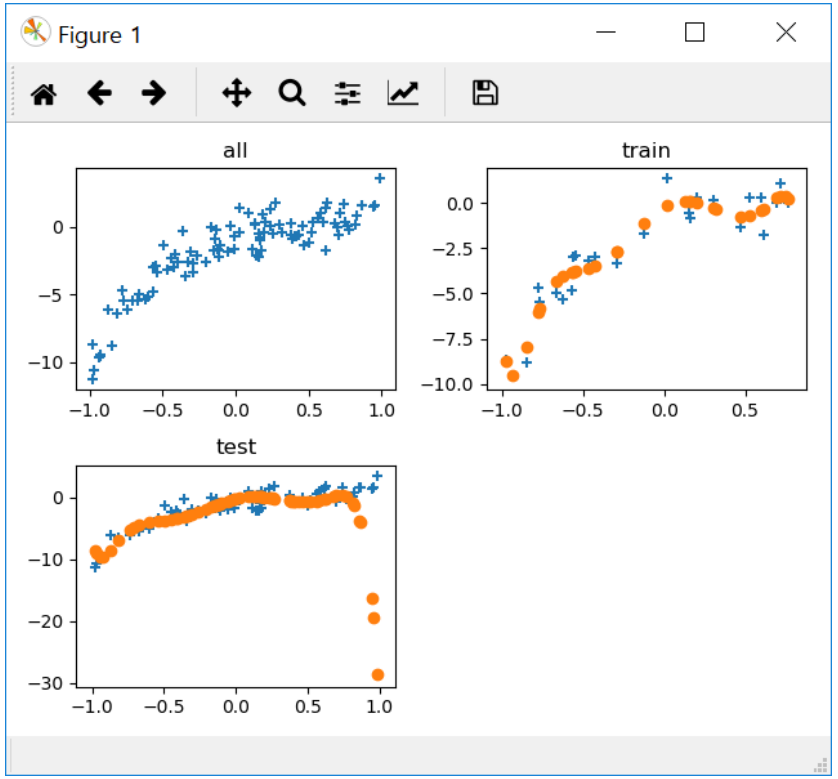
과적합

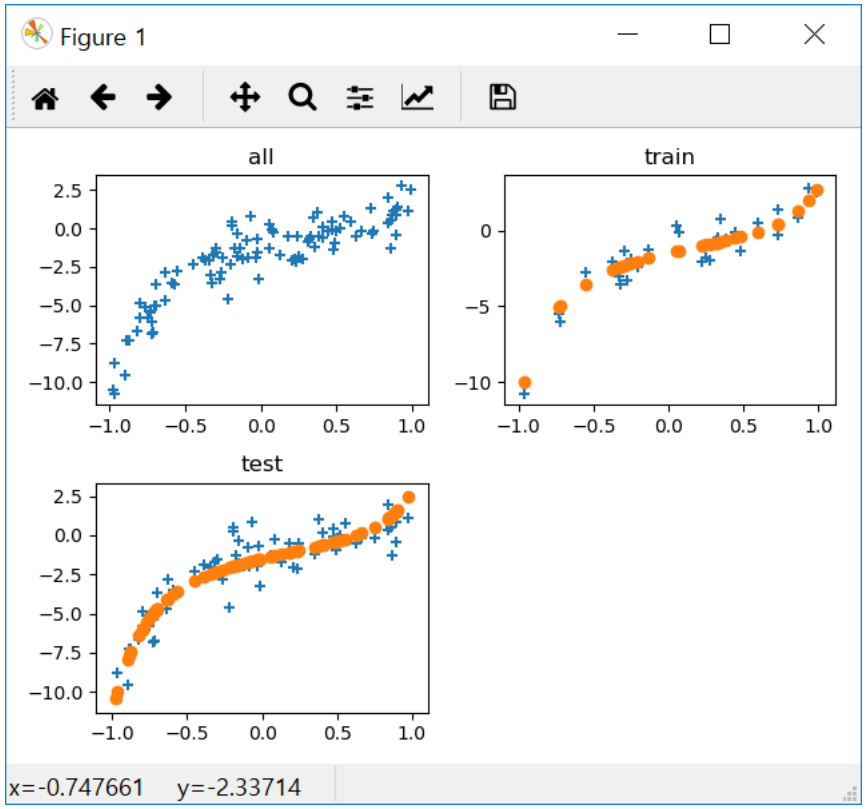
- 주어진 학습 데이터에 너무 적응해서 미지의 데이터에 적합하지 않은 상태
- $y=4x^3-3x^2+2x-1$



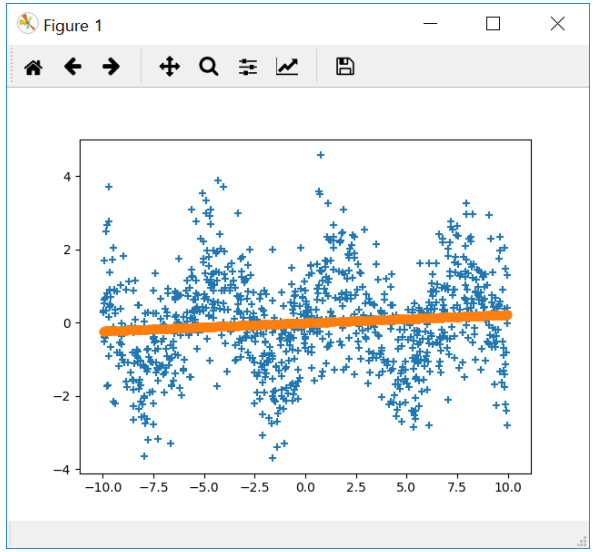


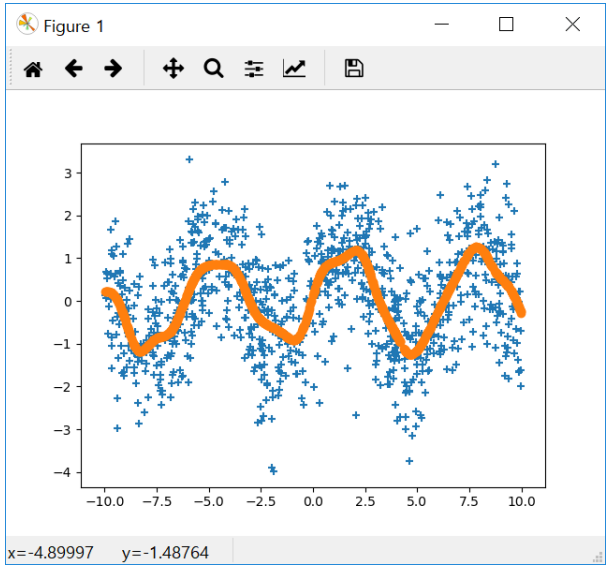
과적합 대응

- 리지 모델
  - sklearn.linear_model.Ridge에 구현되어 있다.

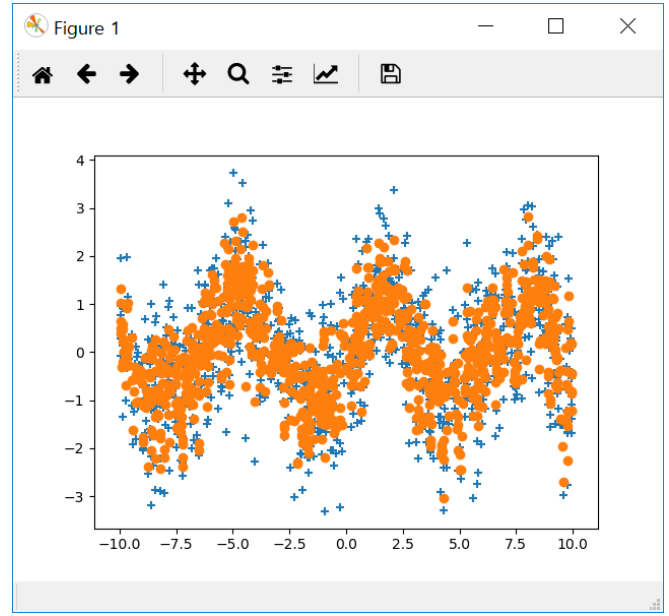


다양한 회귀모델





- 서포트 벡터 머신(SVM)
  - 회귀용 클래스
    - sklearn.svm.SVR
  - 결정 계수 수치는 좋지 않지만 조금 찌그러진 형태이면서 사인파의 분포를 따라간 예측 값이다.
- 랜덤 포레스트
  - 회귀용 클래스
    - sklearn.ensemble.RandomForestRegressor



- scikit-learn
  - iris 데이터를 읽을 수 있도록 API
  - sklearn.datasets.load_iris()
- from sklearn import datasets
- iris = datasets.load_iris()

  - print(iris['DESCR'])
    - 붓꽃데이터를 설명

```
Iris Plants Database
====================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:
```

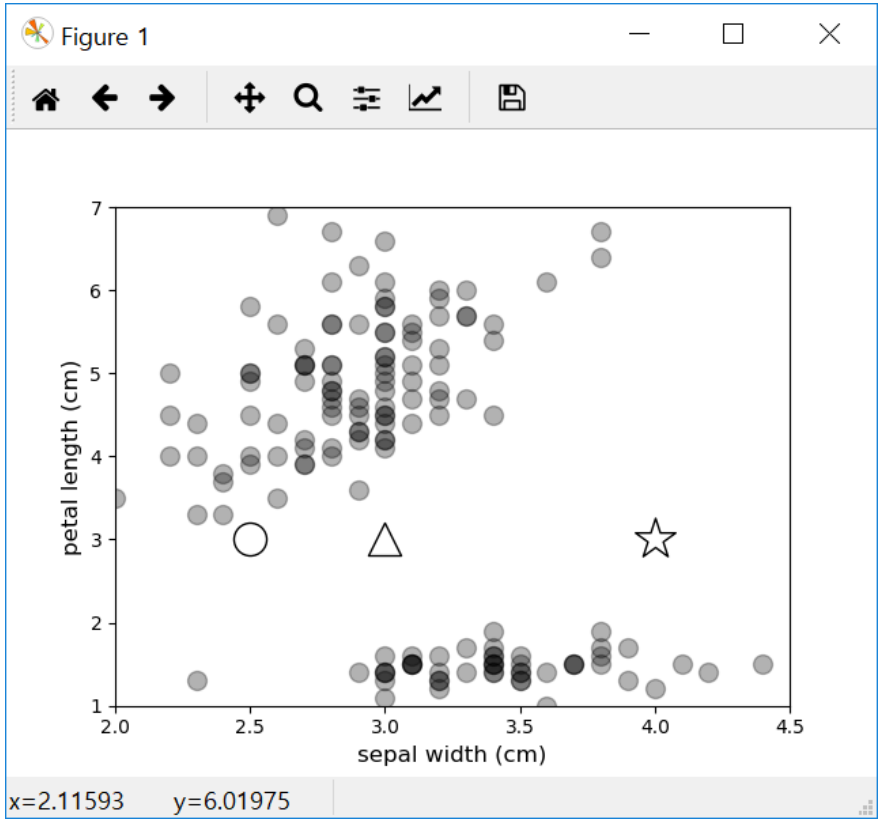
- print(iris['data']) : 붓꽃의 측정값
- print(iris['target']) : 품종이 ID번호로 등록
- print(iris['target_names']) : 품종 등록
- print(iris['feature_names']) : 데이터 속성의 이름

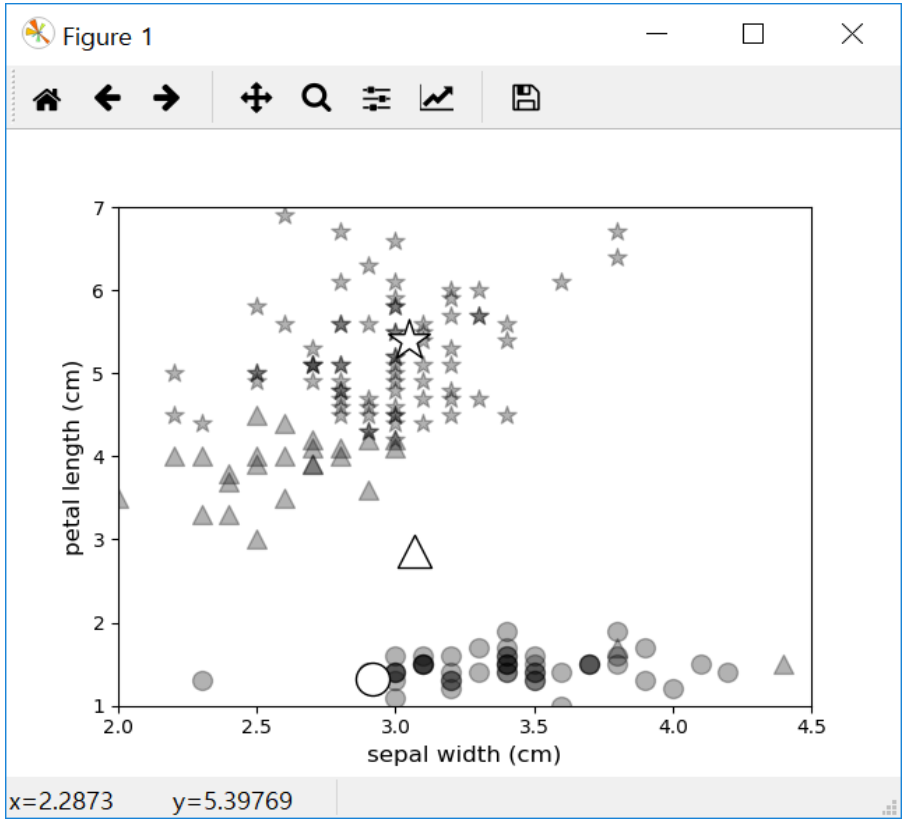
```
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

대표적인 클러스터링 : k-means

- 데이터의 성질에서 데이터 덩어리(클러스터)를 만드는 방법이다.
- k-means를 사용하는 클러스터링 절차
  - 1. 각 데이터를 적절한 방법으로 클러스터에 할당한다. 클러스터 중심을 처음에 정해 초기 클러스터를 형성할 때도 있다.
    - 초기화 방법은 랜덤이어도 상관없지만, 나중에 계산을 효율적으로 할 수 있는 k-means방법을 자주 이용한다.
  - 2. 클러스터마다 중심을 계산한다. 보통은 클러스터에 속한 데이터 점의 산술 평균을 많이 이용한다.
  - 3. 각 데이터에서 클러스터 중심으로 거리를 구한다. 데이터가 가까운 클러스터가 아닌 다른 클러스터에 속한 것 같으면 데이터를 가장 가까운 클러스터 소유로 변경한다.
  - 4. 3에서 클러스터를 변경하지 않거나 미리 정한 문턱 값보다 변화량이 작으면 처리를 종료한다.
  - 5. 새로운 클러스터 할당을 사용해 2부터 다시 처리한다.

- K-means에서 클래스를 형성해 가는 모습
  - 데이터를 3개로 나누는 실행 예
    - 1에서는 클러스터화하지 않기 때문에 모두 ●이다.
    - 2이후로는 클러스터를 형성해 나간다.(작은 ●,▲,★이 데이터 점), 큰 ○,△,☆은 해당 클러스터의 중심을 나타낸다.
    - 2~5를 반복하며 클러스터를 형성했음을 알 수 있다.





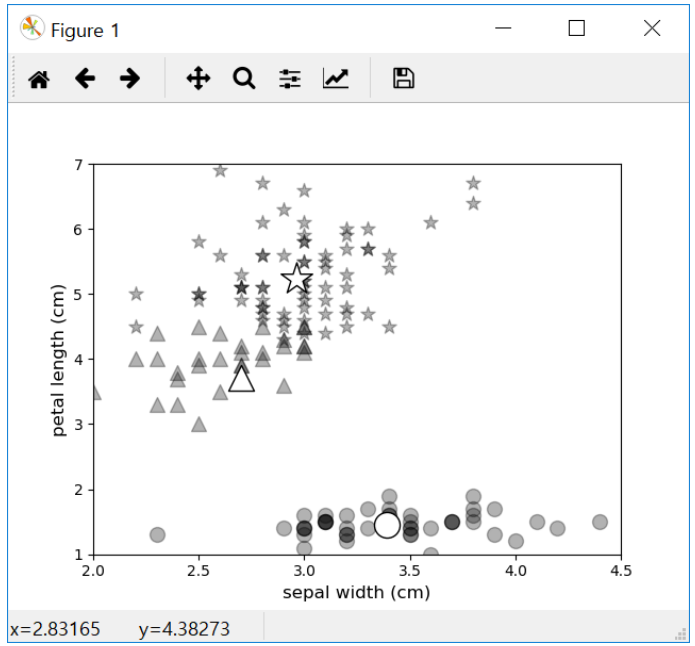


Figure 1
petal length (cm)
sepal width (cm)
x=2.83165 y=4.38273

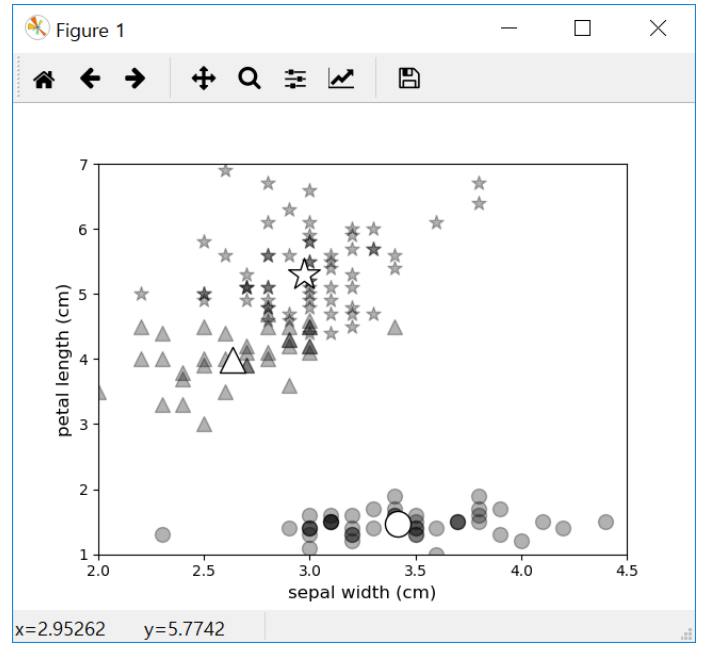


Figure 1
petal length (cm)
sepal width (cm)
x=2.95262 y=5.7742

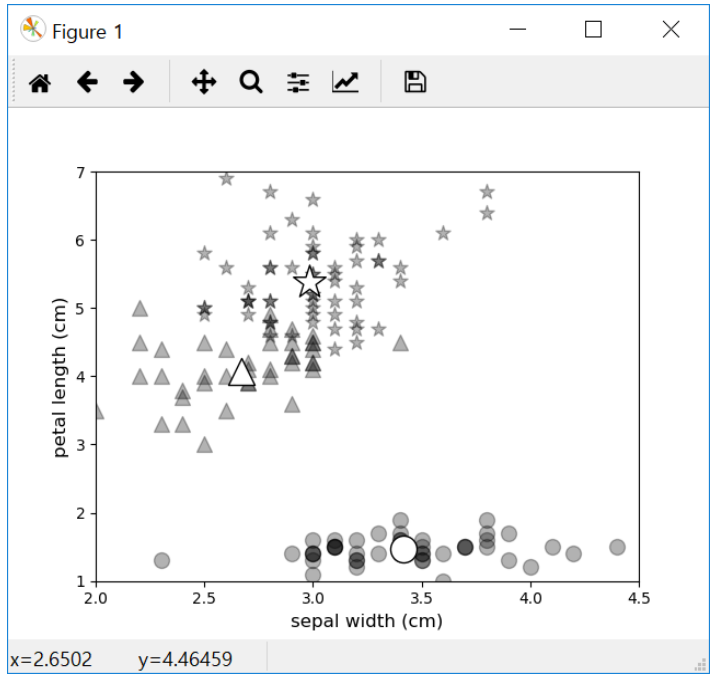


Figure 1
petal length (cm)
sepal width (cm)
x=2.6502 y=4.46459

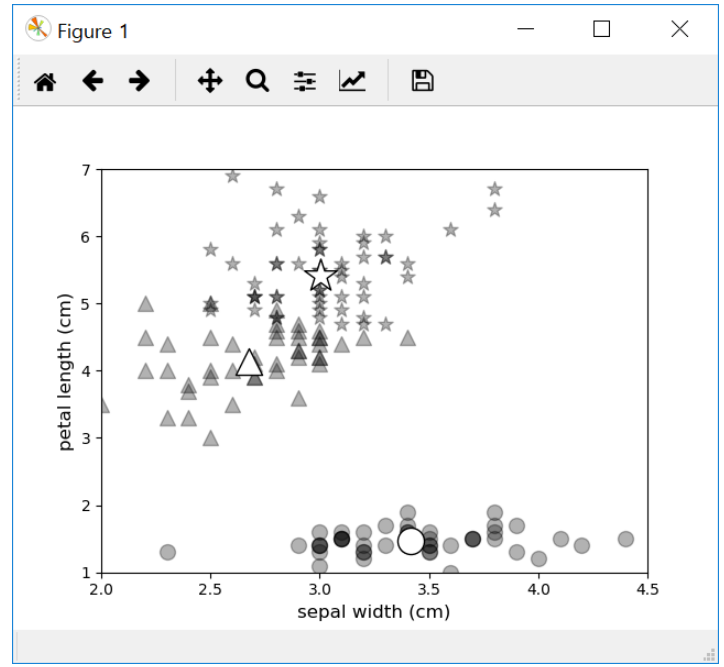
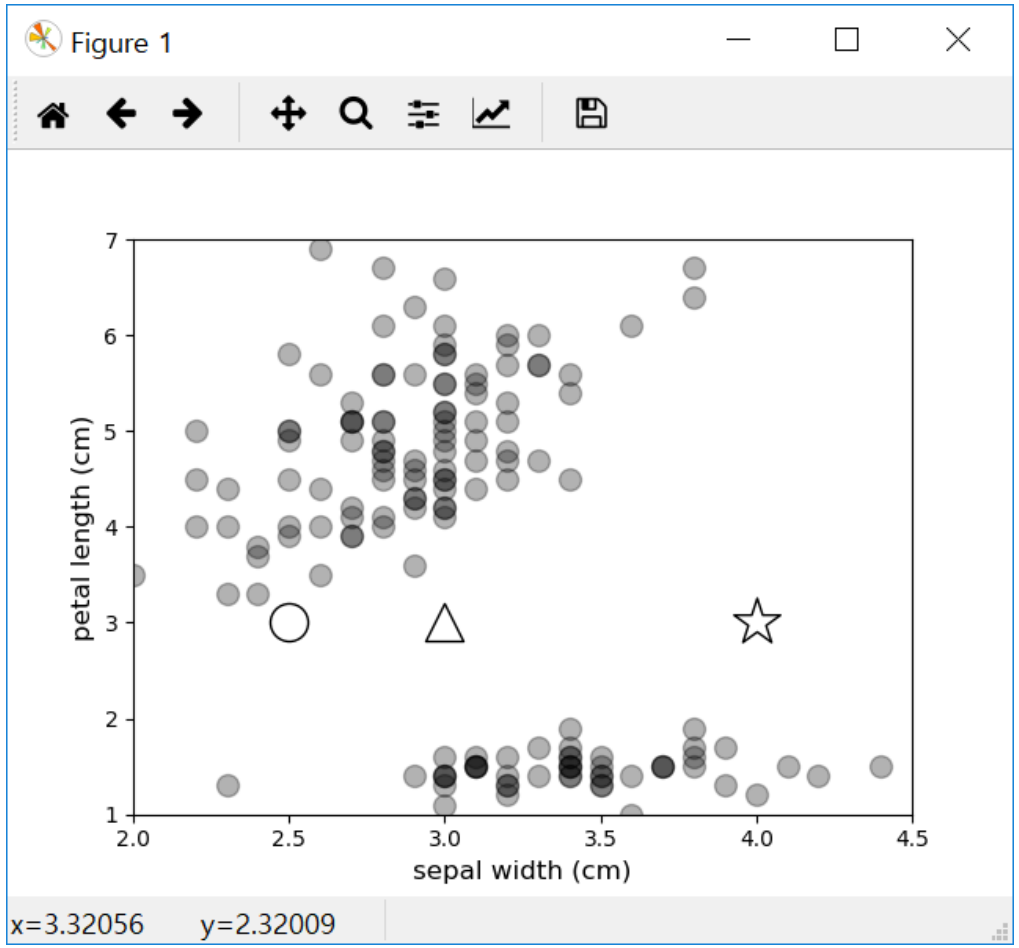


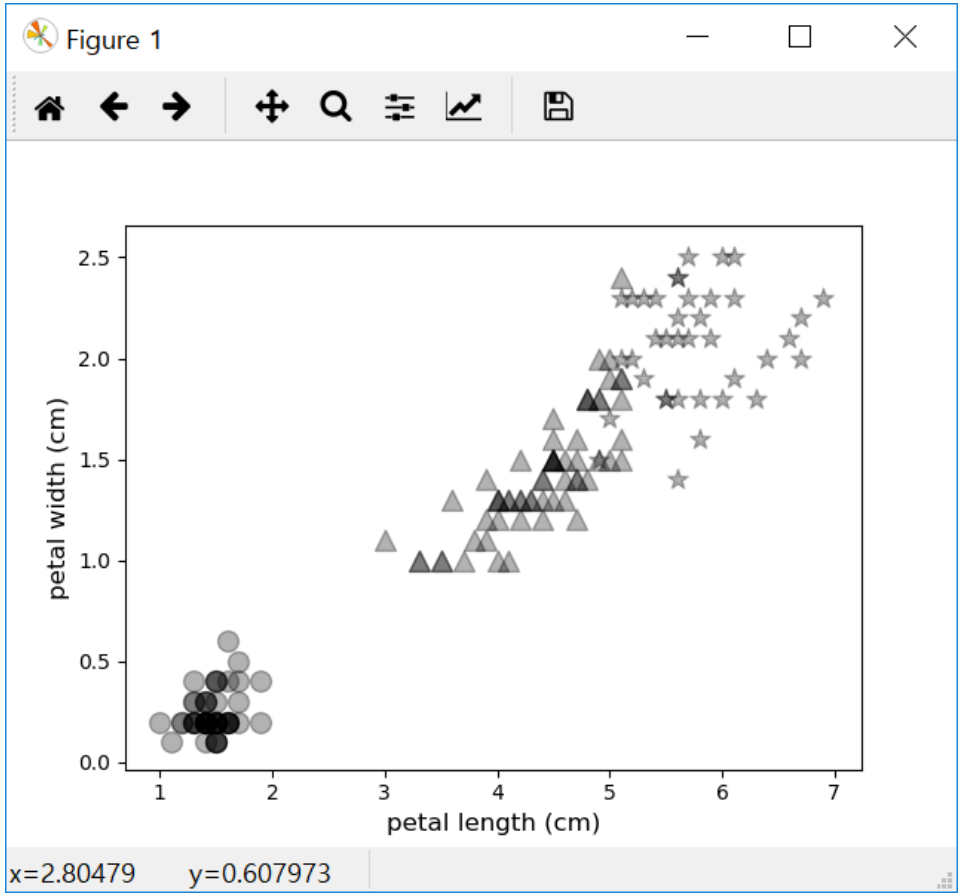
Figure 1
petal length (cm)
sepal width (cm)

- K-means 실행
  - Iris 데이터를 로드해 k-means에서 클러스터를 3개 만드는 코드

  - labels_
    - 각 점에 대한 레이블



- 잎의 길이와 폭으로 클러스터링한 결과의 산포도
  - 클러스터 개수 3으로 하고 있는데, 그중 1개는 잘 분리함, 나머지 2개는 일부 섞여 있음.



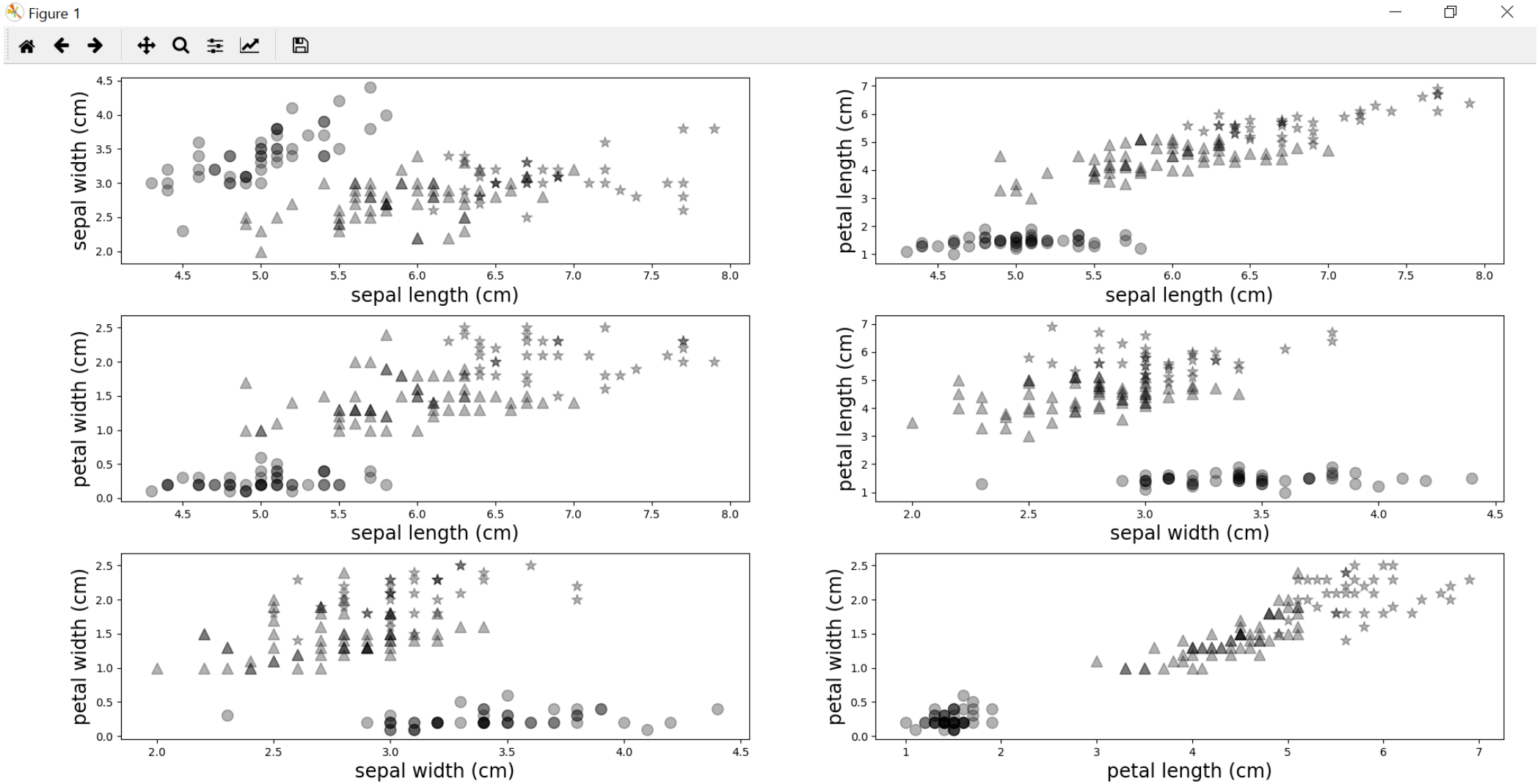


Figure 1
sepal width (cm)
sepal length (cm)
petal length (cm)
sepal length (cm)
petal width (cm)
sepal length (cm)
petal length (cm)
sepal width (cm)
petal width (cm)
sepal width (cm)
petal width (cm)
petal length (cm)

- 꽃의 종류와 클러스터 관계

- from sklearn import metrics
- print(metrics.confusion_matrix(iris['target'],model.labels_))

```
C:\JIN\Anaconda3\envs
[[50  0  0]
 [ 0  2 48]
 [ 0 36 14]]
```