1.

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

x = np.random.rand(100, 1)        # 0~1까지 난수를 100개 만든다
x = x * 2 - 1

y = 4 * x * 3 - 3 * x * 2 + 2 * x - 1

y += np.random.randn( 100, 1 )  # 표준 정규 분포(평균0, 표준 편차1)
                                #난수를 더한다
# 학습 데이터 30개
x_train = x[:30]
y_train = y[:30]

# 테스트 데이터 70개
x_test = x[30:]
y_test = y[30:]

plt.subplot( 2, 2, 1 )
plt.scatter( x, y, marker='+' )
plt.title( 'all' )
plt.subplot( 2, 2, 2 )
plt.scatter( x_train, y_train, marker='+' )
plt.title( 'train' )

plt.subplot( 2, 2, 3 )
plt.scatter( x_test, y_test, marker='+' )
plt.title( 'test' )

plt.tight_layout()
plt.show()
```

2.

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


### 분산 y = 4x^3 - 3x^2 + 2x - 1 데이터 생성

x = np.random.rand(100, 1)  # 0~1까지 난수를 100개 만든다
x = x * 2 - 1               # 값의 범위를 -2~2로 변경

y = 4 * x**3 - 3 * x**2 + 2 * x - 1

y += np.random.randn(100, 1)  # 표준 정규 분포(평균 0, 표준 편차 1)의 난수를 추가함

# 학습 데이터 30개
x_train = x[:30]
y_train = y[:30]

# 테스트 데이터 30개
x_test = x[30:]
y_test = y[30:]


### 최소제곱법으로 9차식으로 회귀를 취해 본다

from sklearn import linear_model

# 학습용 입력 데이터
X_TRAIN = np.c_[x_train**9, x_train**8, x_train**7, x_train**6, x_train**5,
                x_train**4, x_train**3, x_train**2, x_train]

model = linear_model.LinearRegression()
model.fit(X_TRAIN, y_train)


### 계수, 절편, 학습 데이터에 의한 결정계수를 표시

print('계수 ( 학습 데이터 ) ', model.coef_)
print('절편 ( 학습 데이터 ) ', model.intercept_)
```

```
print('결정계수 ( 학습 데이터 ) ', model.score(X_TRAIN, y_train))


### 테스트 데이터에 의한 결정계수를 표시

X_TEST = np.c_[x_test**9, x_test**8, x_test**7, x_test**6, x_test**5,
               x_test**4, x_test**3, x_test**2, x_test]

print('결정계수 ( 테스트 데이터 ) ', model.score(X_TEST, y_test))


### 그래프 표시

plt.subplot(2, 2, 1)
plt.scatter(x, y, marker='+')
plt.title('all')

plt.subplot(2, 2, 2)
plt.scatter(x_train, y_train, marker='+')
plt.scatter(x_train, model.predict(X_TRAIN), marker='o')
plt.title('train')

plt.subplot(2, 2, 3)
plt.scatter(x_test, y_test, marker='+')
plt.scatter(x_test, model.predict(X_TEST), marker='o')
plt.title('test')

plt.tight_layout()
plt.show()
```

3.

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


### 분산 y = 4x^3 - 3x^2 + 2x - 1 데이터 생성

x = np.random.rand(100, 1)  # 0~1까지 난수를 100개 만든다
x = x * 2 - 1               # 값의 범위를 -2~2로 변경

y = 4 * x**3 - 3 * x**2 + 2 * x - 1

y += np.random.randn(100, 1)  # 표준정규분포 ( 평균 0,표준편차 1 ) 의 난수를 더한다

# 학습 데이터 30개
x_train = x[:30]
y_train = y[:30]

# 테스트 데이터 30개
x_test = x[30:]
y_test = y[30:]


### Ridge 로 9차식으로서 회귀를 취해 본다

from sklearn import linear_model

# 학습용의 입력 데이터
X_TRAIN = np.c_[x_train**9, x_train**8, x_train**7, x_train**6, x_train**5,
                x_train**4, x_train**3, x_train**2, x_train]

model = linear_model.Ridge()
model.fit(X_TRAIN, y_train)


### 계수、절편、학습 데이터에 의한 결정계수를 표시
```

```
print('계수 ( 학습 데이터 ) ', model.coef_)
print('절편 ( 학습 데이터 ) ', model.intercept_)

print('결정계수 ( 학습 데이터 ) ', model.score(X_TRAIN, y_train))


### 테스트 데이터에 의한 결정계수를 표시

X_TEST = np.c_[x_test**9, x_test**8, x_test**7, x_test**6, x_test**5,
               x_test**4, x_test**3, x_test**2, x_test]

print('결정계수 ( 테스트 데이터 ) ', model.score(X_TEST, y_test))


### 그래프 표시

plt.subplot(2, 2, 1)
plt.scatter(x, y, marker='+')
plt.title('all')

plt.subplot(2, 2, 2)
plt.scatter(x_train, y_train, marker='+')
plt.scatter(x_train, model.predict(X_TRAIN), marker='o')
plt.title('train')

plt.subplot(2, 2, 3)
plt.scatter(x_test, y_test, marker='+')
plt.scatter(x_test, model.predict(X_TEST), marker='o')
plt.title('test')

plt.tight_layout()
plt.show()
```

4.

```
import math

import numpy as np
import matplotlib.pyplot as plt


###  사인파 데이터를 작성

x = np.random.rand(1000, 1)  # 0~1까지 난수를 1000개 만든다
x = x * 20 - 10              # 값의 범위를 -10~10으로 변경

y = np.array([math.sin(v) for v in x])  # 사인파 커브
y += np.random.randn(1000)  # 표준 정규 분포(평균0, 표준 편차1) 난수를 더한다


### 학습: 최소제곱법

from sklearn import linear_model


model = linear_model.LinearRegression()
model.fit(x, y)


### 계수, 절편, 결정 계수를 표시

print('계수', model.coef_)
print('절편', model.intercept_)

r2 = model.score(x, y)
print('결정계수', r2)

### 그래프 표시

plt.scatter(x, y, marker='+')
plt.scatter(x, model.predict(x), marker='o')
plt.show()
```

5.

```
import math

import numpy as np
import matplotlib.pyplot as plt


### 분산이 있는 사인파 데이터를 작성

x = np.random.rand(1000, 1)  # 0~1까지 난수를 1000개 만든다
x = x * 20 - 10                #값의 범위를 -10~10으로 변경

y = np.array([math.sin(v) for v in x])  # 사인파 커브
y += np.random.randn(1000)  # 표준 정규 분포(평균0, 표준 편차1) 난수를 더한다


### 학습: Support Vector Machine

from sklearn import svm


model = svm.SVR()
model.fit(x, y)


### 결정계수 표시

r2 = model.score(x, y)
print('결정계수', r2)


### 그래프 표시

plt.scatter(x, y, marker='+')
plt.scatter(x, model.predict(x), marker='o')
plt.show()
```

6.

```
import math

import numpy as np
import matplotlib.pyplot as plt


### 분산이 있는 사인파 데이터를 작성

x = np.random.rand(1000, 1)  # 0~1까지 난수를 1000개 만든다
x = x * 20 - 10              # 값의 범위를 -10~10으로 변경

y = np.array([math.sin(v) for v in x])  # 사인파 커브
y += np.random.randn(1000)  # 표준 정규 분포(평균0, 표준 편차1) 난수를 더한다


### 학습: Random Forest

from sklearn import ensemble


model = ensemble.RandomForestRegressor()
model.fit(x, y)


### 결정계수 표시

r2 = model.score(x, y)
print('결정계수', r2)


### 그래프 표시

plt.scatter(x, y, marker='+')
plt.scatter(x, model.predict(x), marker='o')
plt.show()
```