多項式回帰

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

np.random.seed(0)

data\_size = 200

X = np.random.uniform(-3, 3.5, data\_size)

y = 50\*X\*\*5 - 90\*X\*\*4 - 550\*X\*\*3 + 900\*X\*\*2 + 500\*X + 500 + 500\*np.random.randn(data\_size)

fig = plt.figure()

ax = fig.add\_subplot(1,1,1)

ax.scatter(X, y)

ax.set\_title(u"x and y scatter plot")

ax.set\_xlabel("X")

ax.set\_ylabel("y")

plt.show()

X\_train, X\_test, y\_train, y\_test = \

 train\_test\_split(X, y, test\_size=0.3, random\_state=123)

print("X\_trian :", X\_train.shape)

print("X\_test :", X\_test.shape)

print("y\_trian :", y\_train.shape)

print("y\_test :", y\_test.shape)

fig = plt.figure(figsize=(12,15))

for i in range(1,7):

 ax = fig.add\_subplot(3, 2, i)

 polynomial\_features = PolynomialFeatures(degree=i,

 include\_bias=False)

 linear\_regression = LinearRegression()

 pipeline = Pipeline([("polynomial\_features", polynomial\_features),

 ("linear\_regression", linear\_regression)])

 pipeline.fit(X\_train.reshape(-1, 1), y\_train)

 y\_pred = pipeline.predict(X\_test[np.argsort(X\_test)].reshape(-1, 1))

 ax.scatter(X\_test, y\_test, edgecolor='b', s=10, label="test\_data")

 ax.plot(X\_test[np.argsort(X\_test)],

 y\_pred,

 label="Model", color = 'red')

 ax.set\_xlabel("x")

 ax.set\_ylabel("y")

 ax.legend(loc="best")

 ax.set\_title("Degree:{} , MSE:{:.4f}"\

 .format(i, mean\_squared\_error(y\_test[np.argsort(X\_test)], y\_pred)))

plt.show()