#散布図

import numpy as np

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

%matplotlib inline

np.random.seed(100)

data\_size = 100

X = np.random.randn(data\_size)

y = 5 \* X + 10 + 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=0)

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

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

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

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

lr = LinearRegression()

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

print("傾き:",lr.coef\_)

print("切片:",lr.intercept\_)

y\_pred = lr.predict(X\_test.reshape(-1, 1))

print('予測結果\n', y\_test[:5])

print('答え\n', y\_pred[:5])

print('トレーニングデータの平均乗誤差：',\

 mean\_squared\_error(y\_train, lr.predict(X\_train.reshape(-1, 1))))

print('テストデータの平均2乗誤差：', mean\_squared\_error(y\_test, y\_pred))

#リッジ回帰

from sklearn.linear\_model import Ridge

rg = Ridge(alpha=0.3, random\_state=1234)

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

print("傾き:",rg.coef\_)

print("切片:",rg.intercept\_)

y\_pred = rg.predict(X\_test.reshape(-1, 1))

print('予測結果\n', y\_test[:5])

print('答え\n', y\_pred[:5])

print('トレーニングデータの平均乗誤差：',\

 mean\_squared\_error(y\_train, rg.predict(X\_train.reshape(-1, 1))))

print('テストデータの平均2乗誤差：', mean\_squared\_error(y\_test, y\_pred)

#LASSO回帰

from sklearn.linear\_model import Lasso

la = Lasso(alpha=0.3, random\_state=1234)

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

print("傾き:",la.coef\_)

print("切片:",la.intercept\_)

y\_pred = la.predict(X\_test.reshape(-1, 1))

print('予測結果\n', y\_test[:5])

print('答え\n', y\_pred[:5])

print('トレーニングデータの平均2乗誤差：', \

 mean\_squared\_error(y\_train, la.predict(X\_train.reshape(-1, 1))))

print('テストデータの平均2乗誤差：', mean\_squared\_error(y\_test, y\_pred))

fig = plt.figure()

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

ax.scatter(X\_test, y\_test)

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

ax.plot(X\_test, lr.predict(X\_test.reshape(-1, 1)), color = 'r',

 linestyle="-", label="LinearRegression")

ax.plot(X\_test, rg.predict(X\_test.reshape(-1, 1)), color = 'b',

 linestyle**="--", label="Ridge")**

**ax.plot(X\_test,** la.predict(X\_test.reshape(-1, 1)), color = 'g',

 linestyle=":", label="Lasso")

ax.set\_xlabel("x")

ax.set\_ylabel("y")

ax.legend(loc="best")

plt.show()

＃多項式フィッティング　testdata1.txtはｘ値、testdata2.txtはｙ値を別々に読み込む

import numpy as np

import matplotlib.pyplot as plt

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Ridge

from sklearn import linear\_model

from sklearn.metrics import mean\_squared\_error, r2\_score

np.random.seed(0)

X=np.loadtxt('testdata1.txt',delimiter=',')

y=np.loadtxt('testdata2.txt',delimiter=',')

X = X[:, np.newaxis]

y = y[:, np.newaxis]

print(X)

print(y)

# モデル：y = b0 + b1x の b0 と b1 を算出

model = LinearRegression()

model.fit(X, y)

# 推論

y\_pred = model.predict(X)

# 評価

rmse = np.sqrt(mean\_squared\_error(y, y\_pred))

r2 = r2\_score(y, y\_pred)

print(f'rmse : {rmse}')

print(f'R2 : {r2}')

# 可視化

plt.scatter(X, y)

plt.plot(X, y\_pred, color='r')

plt.show()

# 2次元の特徴量に変換

polynomial\_features= PolynomialFeatures(degree=2)

X\_poly = polynomial\_features.fit\_transform(X)

# y = b0 + b1x + b2x^2 の b0～b2 を算出

model = LinearRegression()

model.fit(X\_poly, y)

y\_pred = model.predict(X\_poly)

# 評価

rmse = np.sqrt(mean\_squared\_error(y, y\_pred))

r2 = r2\_score(y, y\_pred)

print(f'rmse : {rmse}')

print(f'R2 : {r2}')

# 可視化

plt.scatter(X, y)

plt.plot(X, y\_pred, color='r')

plt.show()

# 3次元の特徴量に変換

polynomial\_features= PolynomialFeatures(degree=3)

X\_poly = polynomial\_features.fit\_transform(X)

# y = b0 + b1x + b2x^2 + b3x^3 の b0～b3 を算出

model = LinearRegression()

model.fit(X\_poly, y)

y\_pred = model.predict(X\_poly)

# 評価

rmse = np.sqrt(mean\_squared\_error(y, y\_pred))

r2 = r2\_score(y, y\_pred)

print(f'rmse : {rmse}')

print(f'R2 : {r2}')

# 可視化

plt.scatter(X, y)

plt.plot(X, y\_pred, color='r')

plt.show()

# y = b0 + b1 \* x + b2 \* x \*\* 2 + b3 \* x \*\* 3

b0 = model.coef\_[0, 0]

b1 = model.coef\_[0, 1]

b2 = model.coef\_[0, 2]

b3 = model.coef\_[0, 3]

intercept = model.intercept\_[0]

print("data: y = 0.0 + 2x - 2x^2 + 0.5x^3 + random.normal")

print(f'model: y = {b0} + {b1}x + {b2}x^2 + {b3}x^3 + {intercept}')