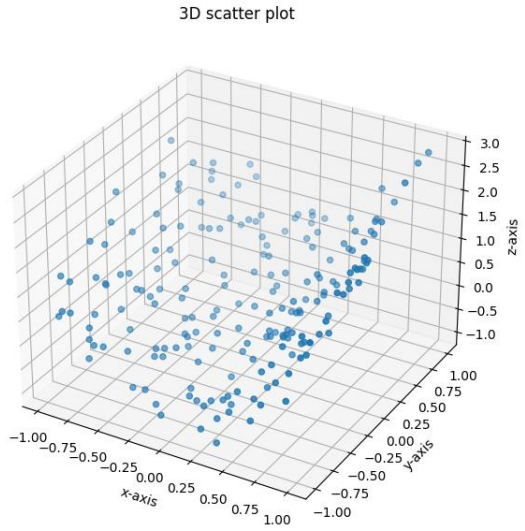
Xiangyu Zheng 21331025 CS7CS4 Machine Learning week3 assignment

(i) (a)

The following is the picture drawn:



It looks like on a curved surface.

(b)

1.First we call:

```
polyFeatures = PolynomialFeatures(degree=5)
print(polyFeatures.get_feature_names)
```

2.We can get 21 features:

['1', 'x0', 'x1', 'x0^2', 'x0 x1', 'x1^2', 'x0^3', 'x0^2 x1', 'x0 x1^2', 'x1^3', 'x0^4', 'x0^3 x1', 'x0^2 x1^2', 'x0 x1^3', 'x1^4', 'x0^5', 'x0^4 x1', 'x0^3 x1^2', 'x0^2 x1^3', 'x0 x1^4', 'x1^5']

3. We use polynomial features to get new training set data:

```
X_Poly = polyFeatures.fit_transform(X_train)
```

According to the cost function of lasso regression in the sklearn document 1.0:

4. I use alpha=0.001 alpha=0.1 alpha=1 and alpha=10 to get the trained parameters of the lasso regression model:

alpha=0.001:

coefficient(Corresponding to 21 features in step2):

[0.00000,-0.00201,0.92001,1.93782,0.06344,0.00000,-0.02853,0.08028,0.00000,0.00000,0.07988,-0.00000,0.00000,0.00028,-0.00000,-0.00000,0.00000,0.03607,0.00000,-0.00000,0.07060]

alpha=0.1:

coefficient(Corresponding to 21 features in step2):

[0.00000,0.00000,0.57765,0.79780,0.00000,-
0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,-
0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000]

alpha=1.0:

coefficient(Corresponding to 21 features in step2):

[0.00000,0.00000,0.00000,0.00000,0.00000,-
0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,-
0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000]

alpha=10:

coefficient(Corresponding to 21 features in step2):

[0.00000,0.00000,0.00000,0.00000,0.00000,-
0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,-
0.00000,0.00000,0.00000,0.00000,0.00000,0.00000,0.00000]

We can see that as the alpha value increases, the coefficients of many features become 0,

When alpha=1 and 10, all coefficients are so small that they can only be displayed with 0 or are already 0.

According to the cost function of lasso regression in the sklearn 1.0 document:

$$\min_{\omega} \frac{1}{2n_{samples}} \|X\omega - y\|_2^2 + \alpha\|\omega\|_1$$

Alpha is a constant. The larger the number, the greater the penalty will be, so it may happen that the coefficient of some features becomes 0 quickly.

I used DummyRegressor as the baseline predictor to calculate Mean square error to compare with the model. Due to the relatively small amount of data, I ran 100 times to calculate the average. The results are as follows:

Alpha=0.001:

Mean square error: Lasso regression with alpha=0.001:0.035567000244699526
dummy:0.6239296864126294

Alpha=0.1:

square error: Lasso regression with alpha=0.1:0.21138985011189354
dummy:0.6357500355013349

Alpha=1:

square error: Lasso regression with alpha=1:0.6224942150003617
dummy:0.6224942150003617

Alpha=10:

square error: Lasso regression with alpha=10:0.6385585924111834
dummy:0.6385585924111834

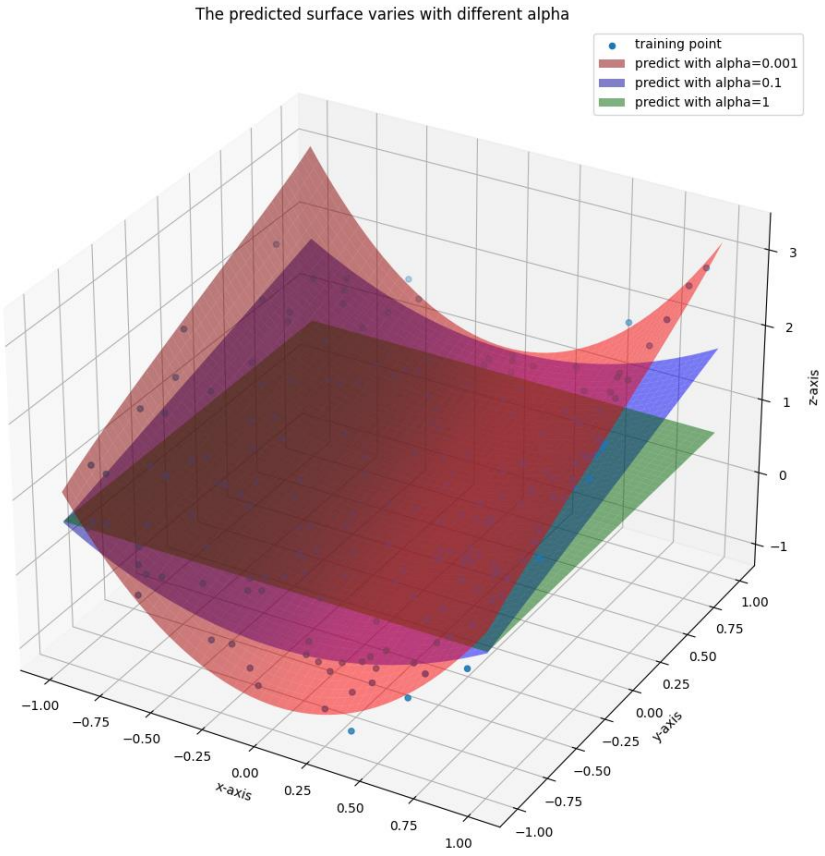
We can see that the error is the smallest when alpha=0.001, which may be due to the data set being too small.

When alpha>=1, lasso regression has the same performance as dummyRegressor

(c)

I use the following command to get the 3D picture below(Since all coefficients are displayed as 0 starting from alpha=1, no graph with alpha=10 is drawn):

```
ax.scatter(dataset[:, 0], dataset[:, 1], dataset[:, 2],
label='training point')
Curve1 = ax.plot_surface(Xitest, Yitest, Z_test, rstride=1,
cstride=1, color='r', label='predict with alpha=0.001', alpha=0.5)
Curve2 = ax.plot_surface(Xitest, Yitest, Z_test2, rstride=1,
cstride=1, color='b', label='predict with alpha=0.1', alpha=0.5)
Curve3 = ax.plot_surface(Xitest, Yitest, Z_test3, rstride=1,
cstride=1, color='g', label='predict with alpha=1', alpha=0.5)
```



We can see from the figure that when alpha=1, since all coefficients are almost 0, the predicted value (Z) is equal to the intercept, so it looks like a plane. When alpha=0.1, only the two features coefficients x1 and x0^2 are not 0, so the predicted value looks like a curved surface. As the alpha becomes smaller and smaller, we see more and more coefficients acting on the curve. It gets steeper and steeper until the highest point in the upper right corner of the picture

(d)

Through the data of b and the picture of c, we can know that in Lasso regression, according to the formula:

Too large alpha value will cause all coefficient values to approach 0 or become 0, and all predictions will become a plane. This is under-fitting, while over-fitting is the opposite. Smaller alpha values will make a lot of The features are all involved in the calculation, which will make the prediction fit the training data very well, which will lead to overfitting. Both overfitting and underfitting can lead to inaccurate predictions.

At the moment, we can set a large alpha value and a small alpha value, and then adjust between the two to achieve a balance between over-fitting and under-fitting.

(e)

Ridge regression:

First, use the same alpha as the Lasso regression, and you can get the parameters and graphs as follows:

Alpha=0.001

[0.00000,0.10403,0.95744,1.85515,0.11818,0.19266,-0.66558,-0.06472,0.56106,

-0.11089,0.21874,-0.11400,-0.01949,0.02830,-0.20109,0.46131,0.11628,0.22131,0.22534,

-0.78410,0.13251]

Alpha=0.1

[0.00000,0.05874,0.90504,1.73928,0.11819,0.15725,

-0.35788,0.01209,0.33811,0.03052,0.33383,-0.13157,0.02844,0.03206,

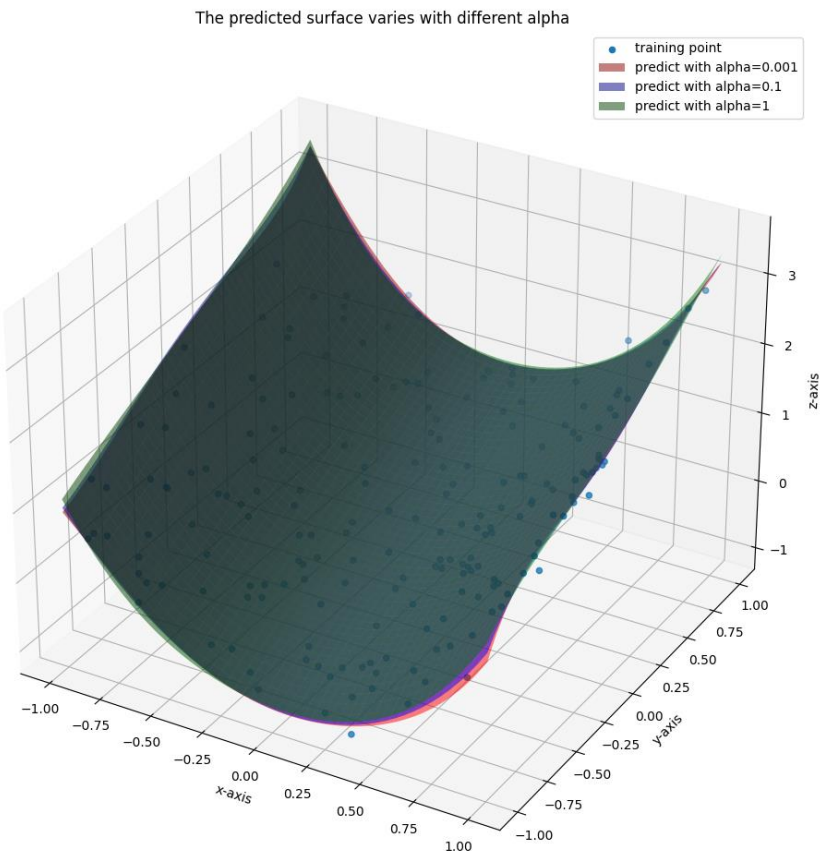
-0.17726,0.18456,0.12399,0.21060,0.10150,-0.51752,0.04219]

Alpha=1

0.00000,0.01628,0.82104,1.34114,0.08438,0.04200,

-0.08823,0.11966,0.07914,0.14773,0.64801,-0.08052,0.20422,0.01449,-0.11256,

-0.00504,0.08470,0.07817,0.01128,-0.15145,0.01271



We can see that the coefficient of each feature has a value, and the predicted surface is similar.

So we based on the cost function of ridge regression in the sklearn document:

$$\min_{w} \|Xw - y\|_2^2 + \alpha\|w\|_2^2$$

We need a larger alpha and a larger interval to change the coefficients of the model, so we reset the parameters to get:

Alpha=1

[0.00000,0.01628,0.82104,1.34114,0.08438,0.04200,-0.08823,0.11966,0.07914,0.14773,0.64801,-0.08052,0.20422,0.01449,-0.11256,-0.00504,0.08470,0.07817,0.01128,-0.15145,0.01271]

Alpha=50

[0.00000,0.02062,0.37686,0.40346,0.05398,-0.03087,0.06037,0.07134,-0.01392,0.19637,0.34309,0.04032,0.10295,0.04486,-0.03318,0.07309,0.01914,0.00598,0.03749,-0.01228,0.12984]

Alpha=1000

[0.00000,0.01008,0.04740,0.03093,0.00518,-0.00489,0.01169,0.01196,0.00131,0.02832,0.02728,0.00373,0.00654,0.00415,-0.00438,0.01139,0.00545,0.00273,0.00749,0.00082,0.02038]

I still use dummyRegressor and model to find mean square error:

Alpha=1

square error: Ridge regression with alpha=1:0.03881332342730842
dummy:0.6073408329876667

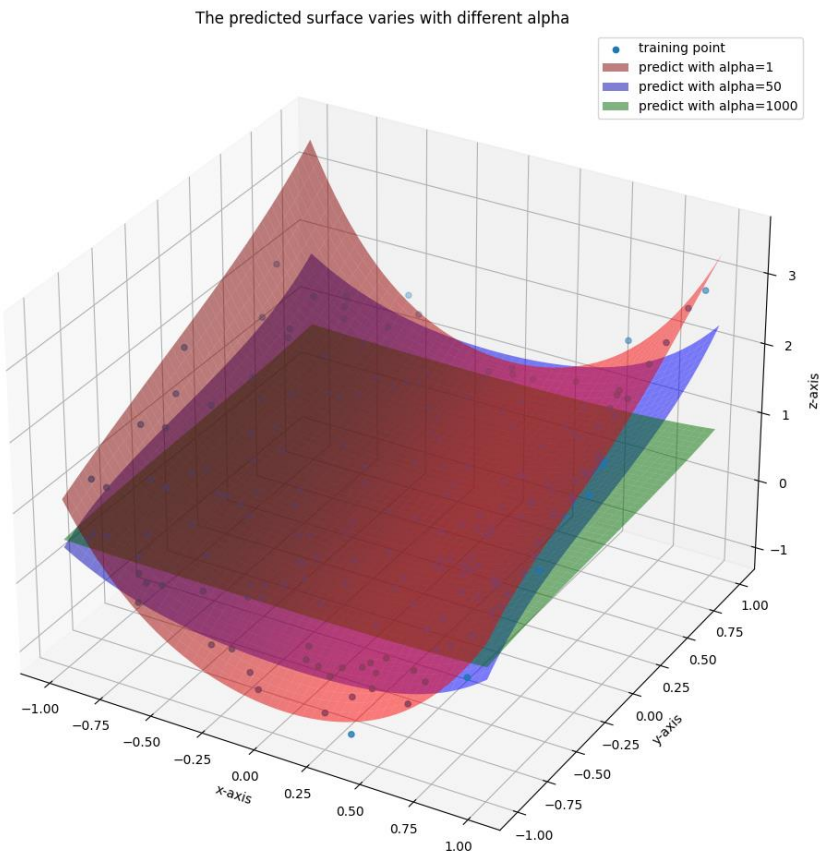
Alpha=50

square error: Ridge regression with alpha=1:0.23864386320786657
dummy:0.6208646774111177

Alpha=1000

square error: Ridge regression with alpha=1:0.598390552836333
dummy:0.6498425837571997

Finally drew a 3D picture



We can see that the predicted surface has changed, resulting in over-fitting and under-fitting, but all the other 20 features except the first feature (1) have values. With the change of C, the proportion of each feature has produced Variety. This shows that Lasso regression focuses on removing useless features, while ridge regression focuses on extracting every feature.

(ii) Using the Lasso model with polynomial features from (i) you'll now look at using cross-validation to select C.

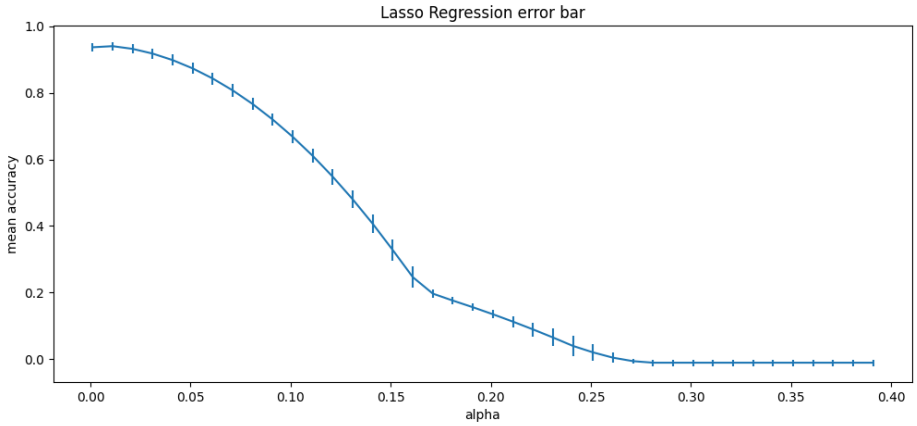
(a)

According to the further experiments of the 3D graphs in (i)(c), when alpha<0.001, the curve cannot become more tortuous, and when alpha>0.4, the prediction has become a plane. So I use 0.001 as the start of the error bar and 0.4 as the end.

First set the alpha value to 0.001, judge that it is less than 0.4 and stop the while loop. Each time the loop alpha value is increased by 0.01, a new model is generated according to the alpha value in each loop, and use 5-fold cross-validation to obtain mean accuracy and standard deviation of the prediction error.

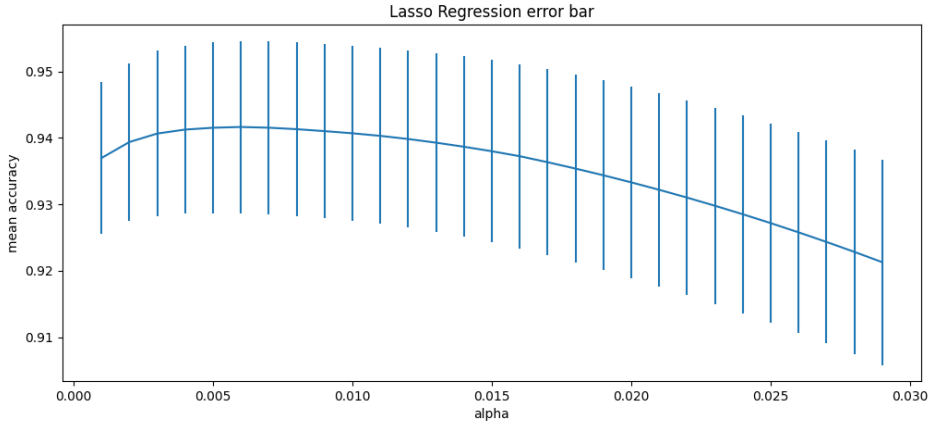
```
alpha = 0.001
alphas,means,stds = [],[],[]
while alpha<0.4:
    LassoR = Lasso(alpha=alpha)
    scores = cross_val_score(LassoR, Xtrain_poly, y_train, cv=5)
    alphas.append(alpha)
    means.append(scores.mean())
    stds.append(scores.std())
    alpha = alpha + 0.01
```

Draw the error bar chart according to the above code, the x-axis is the value of alpha, the y-axis is the mean accuracy, and the vertical line on each point is the standard deviation of the prediction error:



(b)

According to the figure in (ii)(a), we can see that the best value may be between 0.001 and 0.03, so we further draw the errorbar, starting from 0.001 to ending at 0.03, and increasing by 0.001 at each step, we get the following figure:



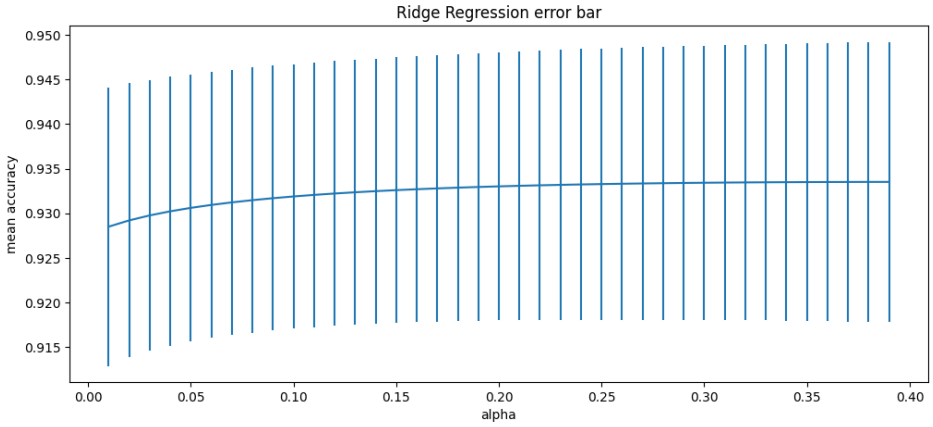
According to the figure, and the formula:

$$\min_{\omega} \frac{1}{2n_{samples}} \|X\omega - y\|_2^2 + \alpha\|\omega\|_1$$

I might choose alpha=0.005 as the best C, Because when alpha=0.005, the accuracy of the model is the highest and the standard deviation of the prediction error is the smallest

(c)

According to the same steps of (ii)(a)-(ii)(b) (just replacing the model), we can get the errorbar of the ridge regression:



I might choose alpha=0.3 as the best C because the accuracy is high and the standard deviation of the prediction error is the smallest

Appendix

code

i-a

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

#Get data from file
data = str(open("data").read()).split('\n')
dataset = []
for d in data[1:]:
    nums = d.split(',')
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)
x = dataset[:, 0]
```

```
y = dataset[:, 1]
z = dataset[:, 2]

#Draw a 3D scatter plot
fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(projection='3d')
ax.scatter(dataset[:,0],dataset[:,1],dataset[:,2])
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D scatter plot' )
plt.show()
```

ib-ic

```
import numpy as np
from scipy.interpolate import griddata
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso

#Get data from file
data = str(open("data").read()).split('\n')
dataset = []
for d in data[1:]:
    nums = d.split(',')
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)

#Get an array of x, y, z
x = dataset[:, 0]
y = dataset[:, 1]
z = dataset[:, 2]

#Generate training data and training target array
X_train = dataset.take([0, 1], axis=1)
y_train = dataset[:, 2]

#Add polynomial
polyFeatures = PolynomialFeatures(degree=5)
print(polyFeatures.get_feature_names)
X_Poly = polyFeatures.fit_transform(X_train)
print(polyFeatures.get_feature_names())

#Generate Lasso regression model and train the model
LassoR = Lasso(alpha=0.001)
LassoR.fit(X_Poly, y_train)

#Generate test set and add polynomial
Xtest = []
grid = np.linspace(-1, 1)
for i in grid:
    for j in grid:
        Xtest.append([i, j])
Xtest = np.array(Xtest)
Xtest_poly = polyFeatures.fit_transform(Xtest)

#Use the test set to predict and get the target
X_pre = LassoR.predict(Xtest_poly)

Xtest_x = Xtest[:, 0]
```

```
Xtest_y = Xtest[:, 1]

#Generate plot_surface data
xi = np.linspace(min(x), max(x))
yi = np.linspace(min(y), max(y))
X, Y = np.meshgrid(xi, yi)
Z = griddata((x, y), z, (X, Y), 'nearest')
xitest = np.linspace(min(Xtest_x), max(Xtest_x))
yitest = np.linspace(min(Xtest_y), max(Xtest_y))
Xitest, Yitest = np.meshgrid(xitest, yitest)
Z_test = griddata((Xtest_x, Xtest_y), X_pre, (Xitest, yitest), 'nearest')

#Create 3D canvas
fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(projection='3d')

#Draw the first 3D surface
ax.scatter(dataset[:, 0], dataset[:, 1], dataset[:, 2], label='training poin
t')
Curve1 = ax.plot_surface(Xitest, Yitest, Z_test, rstride=1, cstride=1, color
='r', label='predict with alpha=0.001', alpha=0.5)
Curve1._facecolors2d = Curve1._facecolor3d
Curve1._edgecolors2d = Curve1._edgecolor3d

#Generate the second model and draw the surface
LassoR2 = Lasso(alpha=0.1)
LassoR2.fit(X_Poly, y_train)
X_pre2 = LassoR2.predict(Xtest_poly)
Z_test2 = griddata((Xtest_x, Xtest_y), X_pre2, (Xitest, yitest), 'nearest')

Curve2 = ax.plot_surface(Xitest, Yitest, Z_test2, rstride=1, cstride=1, colo
r='b', label='predict with alpha=0.1', alpha=0.5)
Curve2._facecolors2d = Curve2._facecolor3d
Curve2._edgecolors2d = Curve2._edgecolor3d

#Generate the third model and draw the surface
LassoR3 = Lasso(alpha=1)
LassoR3.fit(X_Poly, y_train)
X_pre3 = LassoR3.predict(Xtest_poly)
Z_test3 = griddata((Xtest_x, Xtest_y), X_pre3, (Xitest, yitest), 'nearest')

Curve3 = ax.plot_surface(Xitest, Yitest, Z_test3, rstride=1, cstride=1, colo
r='g', label='predict with alpha=1', alpha=0.5)
Curve3._facecolors2d = Curve3._facecolor3d
Curve3._edgecolors2d = Curve3._edgecolor3d

#Set picture parameters and draw 3D pictures flatly
ax.legend()
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('The predicted surface varies with different alpha')
plt.show()
```

ib for compare

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso
from sklearn.dummy import DummyRegressor
from sklearn.metrics import mean_squared_error

#Get data from file
data = str(open("data").read()).split('\n')
```

```
dataset = []
for d in data[1:]:
    nums = d.split(',')
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)

x = dataset[:, 0]
y = dataset[:, 1]
z = dataset[:, 2]
X_train = dataset.take([0, 1], axis=1)
y_train = dataset[:, 2]

times = 100
modelSq, dummySq = 0, 0
for i in range(0,times):
    Xtrain, Xtest, ytrain, ytest = train_test_split(X_train, y_train, test_size=0.2)
    Xtrain_poly = PolynomialFeatures(degree=5).fit_transform(Xtrain)
    Xtest_poly = PolynomialFeatures(degree=5).fit_transform(Xtest)
    lassoModel = Lasso(alpha=0.01).fit(Xtrain_poly, ytrain)
    dummy = DummyRegressor(strategy='mean').fit(Xtrain_poly, ytrain)
    ypred = lassoModel.predict(Xtest_poly)
    ydummy = dummy.predict(Xtest_poly)
    modelSq1 = mean_squared_error(ytest, ypred)
    dummySq1 = mean_squared_error(ytest, ydummy)
    print(modelSq1)
    print(dummySq1)
    modelSq = modelSq+modelSq1
    dummySq = dummySq+dummySq1

print('square error: Lasso regression with alpha=10:'+str(modelSq/times)+' dummy:'+str(dummySq/times))
```

ie

```
import numpy as np
from scipy.interpolate import griddata
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge

#Code function is the same as (i)(b-c)
data = str(open("data").read()).split('\n')
dataset = []
for d in data[1:]:
    nums = d.split(',')
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)

x = dataset[:, 0]
y = dataset[:, 1]
z = dataset[:, 2]

X_train = dataset.take([0, 1], axis=1)
y_train = dataset[:, 2]

polyFeatures = PolynomialFeatures(degree=5)
```

```
print(polyFeatures.get_feature_names)
X_Poly = polyFeatures.fit_transform(X_train)
print(polyFeatures.get_feature_names())
ridgeR = Ridge(alpha=1)
ridgeR.fit(X_Poly, y_train)

Xtest = []
grid = np.linspace(-1, 1)
for i in grid:
    for j in grid:
        Xtest.append([i, j])
Xtest = np.array(Xtest)
Xtest_poly = polyFeatures.fit_transform(Xtest)

X_pre = ridgeR.predict(Xtest_poly)

Xtest_x = Xtest[:, 0]
Xtest_y = Xtest[:, 1]

xi = np.linspace(min(x), max(x))
yi = np.linspace(min(y), max(y))
X, Y = np.meshgrid(xi, yi)
Z = griddata((x, y), z, (X, Y), 'nearest')

xitest = np.linspace(min(Xtest_x), max(Xtest_x))
yitest = np.linspace(min(Xtest_y), max(Xtest_y))
Xitest, Yitest = np.meshgrid(xitest, yitest)
Z_test = griddata((Xtest_x, Xtest_y), X_pre, (Xitest, yitest), 'nearest')

fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(projection='3d')

ax.scatter(dataset[:, 0], dataset[:, 1], dataset[:, 2], label='training poin
t')
Curve1 = ax.plot_surface(Xitest, Yitest, Z_test, rstride=1, cstride=1, color
='r', label='predict with alpha=0.001', alpha=0.5)
Curve1._facecolors2d = Curve1._facecolor3d
Curve1._edgecolors2d = Curve1._edgecolor3d

ridgeR2 = Ridge(alpha=50)
ridgeR2.fit(X_Poly, y_train)
X_pre2 = ridgeR2.predict(Xtest_poly)
Z_test2 = griddata((Xtest_x, Xtest_y), X_pre2, (Xitest, yitest), 'nearest')

Curve2 = ax.plot_surface(Xitest, Yitest, Z_test2, rstride=1, cstride=1, colo
r='b', label='predict with alpha=0.1', alpha=0.5)
Curve2._facecolors2d = Curve2._facecolor3d
Curve2._edgecolors2d = Curve2._edgecolor3d

ridgeR3 = Ridge(alpha=1000)
ridgeR3.fit(X_Poly, y_train)
X_pre3 = ridgeR3.predict(Xtest_poly)
Z_test3 = griddata((Xtest_x, Xtest_y), X_pre3, (Xitest, yitest), 'nearest')

Curve3 = ax.plot_surface(Xitest, Yitest, Z_test3, rstride=1, cstride=1, colo
r='g', label='predict with alpha=1', alpha=0.5)
Curve3._facecolors2d = Curve3._facecolor3d
Curve3._edgecolors2d = Curve3._edgecolor3d

ax.legend()
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('The predicted surface varies with different alpha')
plt.show()
```

ie for compare

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.dummy import DummyRegressor
from sklearn.metrics import mean_squared_error

data = str(open("data").read()).split('\n')

dataset = []
for d in data[1:]:
    nums = d.split(',')
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)

x = dataset[:, 0]
y = dataset[:, 1]
z = dataset[:, 2]
X_train = dataset.take([0, 1], axis=1)
y_train = dataset[:, 2]




times = 100
modelSq, dummySq = 0, 0
for i in range(0,times):
    Xtrain, Xtest, ytrain, ytest = train_test_split(X_train, y_train, test_size=0.2)
    Xtrain_poly = PolynomialFeatures(degree=5).fit_transform(Xtrain)
    Xtest_poly = PolynomialFeatures(degree=5).fit_transform(Xtest)
    ridgeR = Ridge(alpha=1000).fit(Xtrain_poly, ytrain)
    dummy = DummyRegressor(strategy='mean').fit(Xtrain_poly, ytrain)
    ypred = ridgeR.predict(Xtest_poly)
    ydummy = dummy.predict(Xtest_poly)
    modelSq1 = mean_squared_error(ytest, ypred)
    dummySq1 = mean_squared_error(ytest, ydummy)
    print(modelSq1)
    print(dummySq1)
    modelSq = modelSq+modelSq1
    dummySq = dummySq+dummySq1

print('square error: Ridge regression with alpha=1:'+str(modelSq/times)+' dummy:'+str(dummySq/times))
```

iia-iib

```
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

#Get data from file
data = str(open("data").read()).split('\n')
dataset = []
for d in data[1:]:
    nums = d.split(',')
```

```
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)

#Get an array of x, y, z
x = dataset[:, 0]
y = dataset[:, 1]
z = dataset[:, 2]

#Generate training data and training target array
X_train = dataset.take([0, 1], axis=1)
y_train = dataset[:, 2]

#Add polynomial
Xtrain_poly = PolynomialFeatures(degree=5).fit_transform(X_train)

#Draw out the error bar
alpha = 0.001
alphas,means,stds = [],[],[]
while alpha<0.03:
    LassoR = Lasso(alpha=alpha)
    scores = cross_val_score(LassoR, Xtrain_poly, y_train, cv=5)
    alphas.append(alpha)
    means.append(scores.mean())
    stds.append(scores.std())
    alpha = alpha + 0.001

plt.figure(figsize=(12,5))
plt.title('Lasso Regression error bar')
plt.xlabel('alpha')
plt.ylabel('mean accuracy')
plt.errorbar(alphas,means,stds,mfc='red')
plt.show()
```

ii-c

```
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

#Code function is the same as (ii)(a)
data = str(open("data").read()).split('\n')
dataset = []
for d in data[1:]:
    nums = d.split(',')
    dxy = []
    dxy.append(float(nums[0]))
    dxy.append(float(nums[1]))
    dxy.append(float(nums[2]))
    dataset.append(dxy)
dataset = np.array(dataset)

x = dataset[:, 0]
y = dataset[:, 1]
z = dataset[:, 2]

X_train = dataset.take([0, 1], axis=1)
y_train = dataset[:, 2]
Xtrain_poly = PolynomialFeatures(degree=5).fit_transform(X_train)

```

```
27.
28. alpha = 0.01
29. alphas,means,stds = [],[],[]
30. while alpha<0.4:
31.     RidgeR = Ridge(alpha=alpha)
32.     scores = cross_val_score(RidgeR, Xtrain_poly, y_train, cv=5)
33.     alphas.append(alpha)
34.     means.append(scores.mean())
35.     stds.append(scores.std())
36.     alpha = alpha + 0.01
37.
38. plt.figure(figsize=(12,5))
39. plt.title('Ridge Regression error bar')
40. plt.xlabel('alpha')
41. plt.ylabel('mean accuracy')
42. plt.errorbar(alphas,means,stds,mfc='red')
43. plt.show()
```