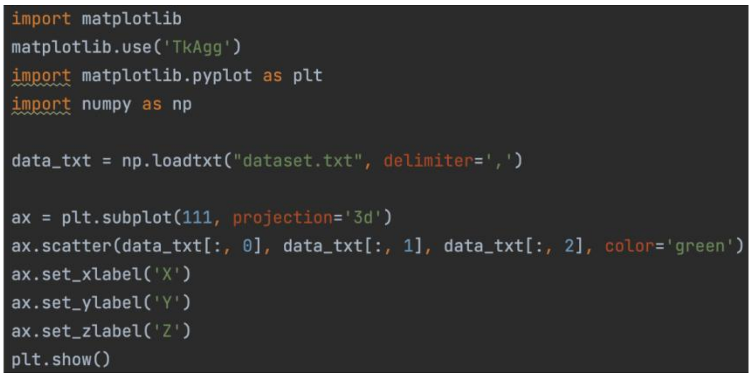
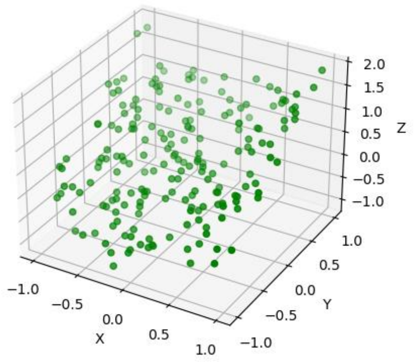
Machine Learning Week 3 Assignment

Jiaming Deng 22302794 id: 4-4-4

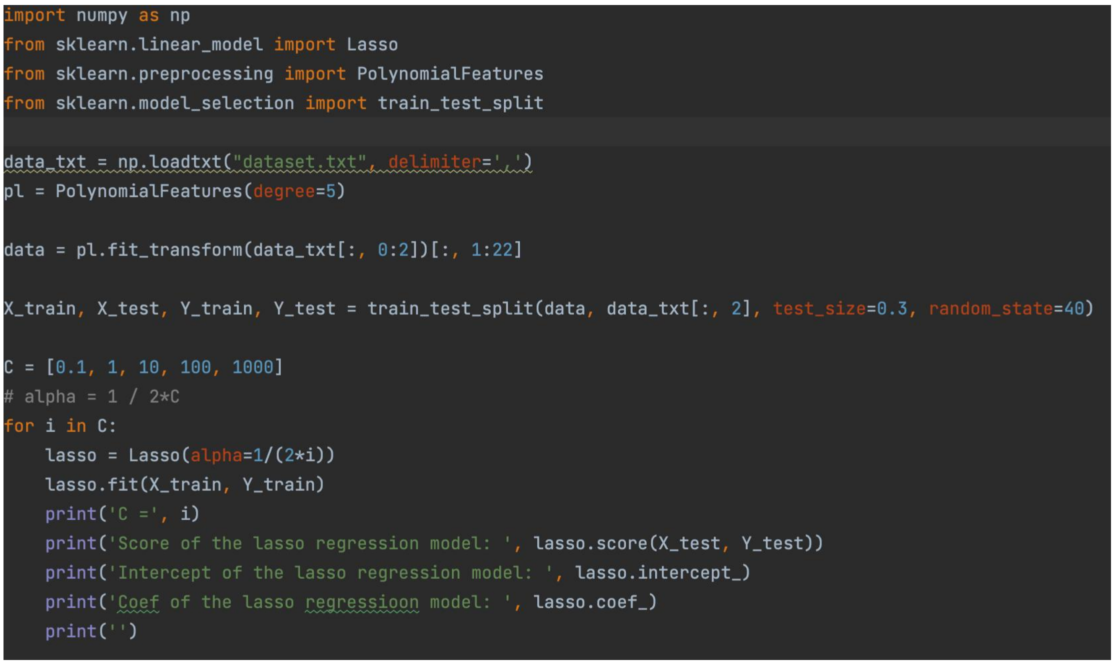
i **(a)** Firstly, I read in the data using numpy's load_txt function, next I use the subplot function in matplotlib and set up the plotted image as 3D. Finally the names of the axes are added. It looks like a curve.



```
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import numpy as np

data_txt = np.loadtxt("dataset.txt", delimiter=',')

ax = plt.subplot(111, projection='3d')
ax.scatter(data_txt[:, 0], data_txt[:, 1], data_txt[:, 2], color='green')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```

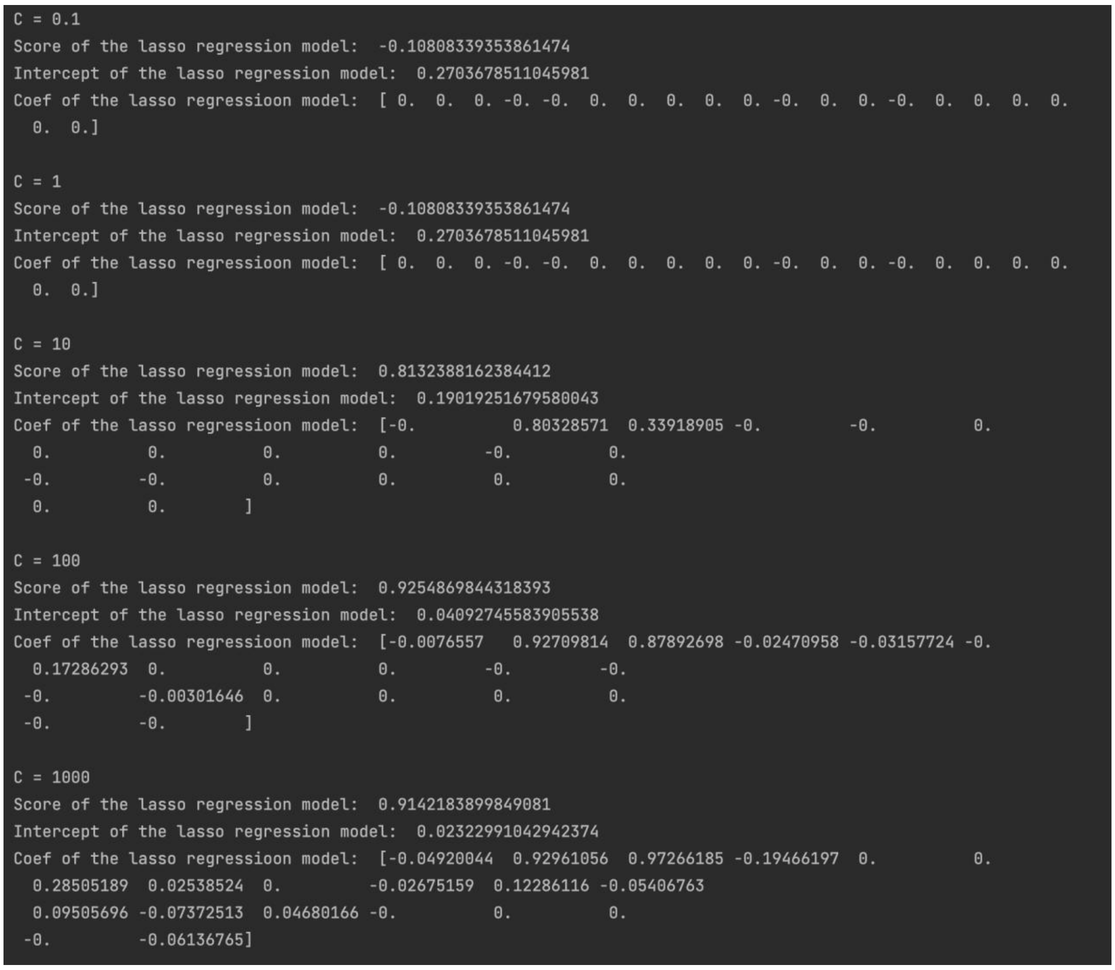


i **(b)** Firstly, using the PolynomialFeatures function and specifying degree=5 to expand X1 and X2 in the original dataset to all combinations of powers of this two features up to power 5. Secondly, I divide the data set into a 70% training set and a 30% test set. Lastly, Training Lasso regression models with these polynomial features for a large range of values of C which are [0.1, 1, 10, 100, 1000.



```
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split

data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)

data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

X_train, X_test, Y_train, Y_test = train_test_split(data, data_txt[:, 2], test_size=0.3, random_state=40)

C = [0.1, 1, 10, 100, 1000]
# alpha = 1 / 2*C
for i in C:
    lasso = Lasso(alpha=1/(2*i))
    lasso.fit(X_train, Y_train)
    print('C =', i)
    print('Score of the lasso regression model: ', lasso.score(X_test, Y_test))
    print('Intercept of the lasso regression model: ', lasso.intercept_)
    print('Coef of the lasso regressioon model: ', lasso.coef_)
    print('')
```

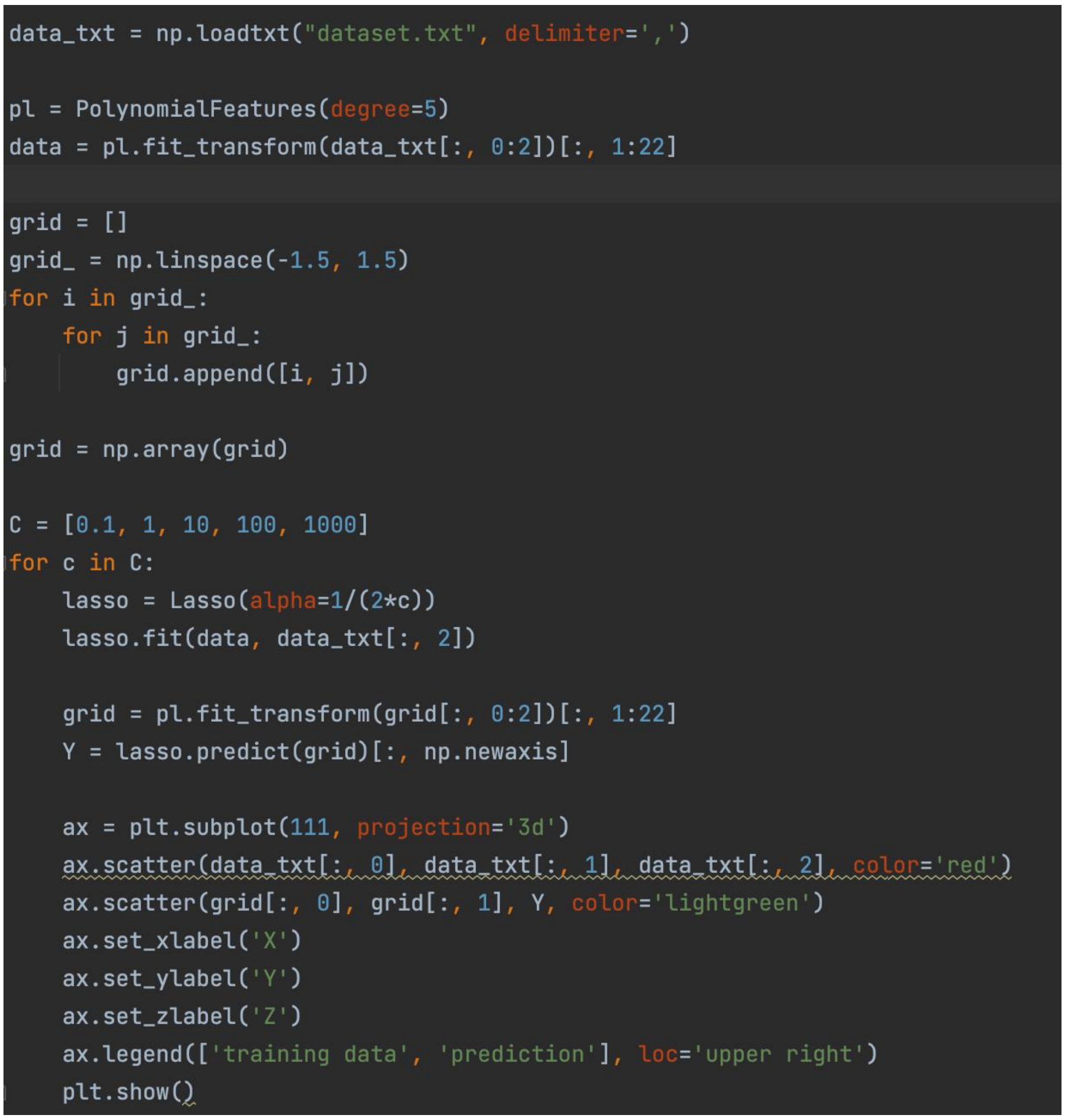
When C equals 0.1 and 1, all parameters have a value of 0. When C equals 10, w2 equals 0.80328571, w3 equals 0.33918905 and all other parameters are 0. As C increases, the number of parameters that are not 0 is increasing.



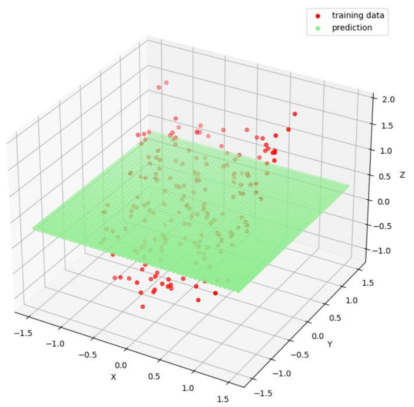
```
C = 0.1
Score of the lasso regression model:  -0.10808339353861474
Intercept of the lasso regression model:  0.2703678511045981
Coef of the lasso regressioon model:  [ 0.  0.  0. -0. -0.  0.  0.  0.  0.  0. -0.  0.  0. -0.  0.  0.  0.  0.
  0.  0.]

C = 1
Score of the lasso regression model:  -0.10808339353861474
Intercept of the lasso regression model:  0.2703678511045981
Coef of the lasso regressioon model:  [ 0.  0.  0. -0. -0.  0.  0.  0.  0.  0. -0.  0.  0. -0.  0.  0.  0.  0.
  0.  0.]

C = 10
Score of the lasso regression model:  0.8132388162384412
Intercept of the lasso regression model:  0.19019251679580043
Coef of the lasso regressioon model:  [-0.          0.80328571  0.33918905 -0.         -0.          0.
  0.          0.          0.          0.         -0.          0.
 -0.         -0.          0.          0.          0.          0.
  0.          0.        ]

C = 100
Score of the lasso regression model:  0.9254869844318393
Intercept of the lasso regression model:  0.04092745583905538
Coef of the lasso regressioon model:  [-0.0076557   0.92709814  0.87892698 -0.02470958 -0.03157724 -0.
  0.17286293  0.          0.          0.         -0.         -0.
 -0.         -0.00301646  0.          0.          0.          0.
 -0.         -0.        ]

C = 1000
Score of the lasso regression model:  0.9142183899849081
Intercept of the lasso regression model:  0.02322991042942374
Coef of the lasso regressioon model:  [-0.04920044  0.92961056  0.97266185 -0.19466197  0.          0.
  0.28505189  0.02538524  0.         -0.02675159  0.12286116 -0.05406763
  0.09505696 -0.07372513  0.04680166 -0.          0.          0.
 -0.         -0.06136765]
```

i (c) Firstly, using python's min and max functions to find the minimum and maximum values of the first two variables in the dataset. Second, using numpy's linspace function to create a test set that completely covers the uniform distribution of the first two variables in the dataset. Finally, the lasso model is trained at values of C of [0.1, 1, 10, 100, 1000]. Using matplotlib's 3d plotting function, the points in the training set are drawn as red dots and the points in the test set are drawn as bright green dots.

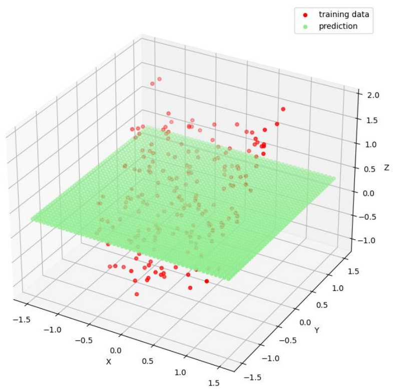
When C=0.1 and 1, all parameters of the Lasso regression model are equal to 0, so the prediction points are plotted and the predicted score is negative. When C=10, 100 and 1000, some of the parameters of the Lasso regression model are not 0, so the prediction points are curves and the predicted scores are integers, but the highest score of 0.9254 occurs at C=100.



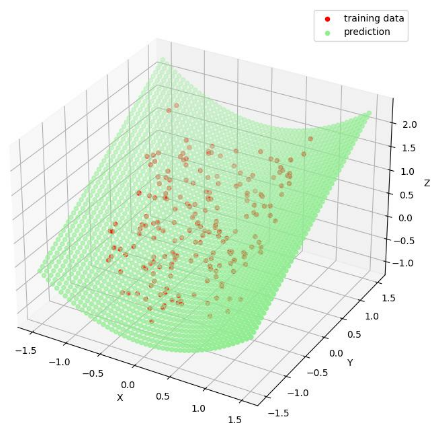
```python
data_txt = np.loadtxt("dataset.txt", delimiter=',')

pl = PolynomialFeatures(degree=5)
data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

grid = []
grid_ = np.linspace(-1.5, 1.5)
for i in grid_:
    for j in grid_:
        grid.append([i, j])

grid = np.array(grid)

C = [0.1, 1, 10, 100, 1000]
for c in C:
    lasso = Lasso(alpha=1/(2*c))
    lasso.fit(data, data_txt[:, 2])

    grid = pl.fit_transform(grid[:, 0:2])[:, 1:22]
    Y = lasso.predict(grid)[:, np.newaxis]

    ax = plt.subplot(111, projection='3d')
    ax.scatter(data_txt[:, 0], data_txt[:, 1], data_txt[:, 2], color='red')
    ax.scatter(grid[:, 0], grid[:, 1], Y, color='lightgreen')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.legend(['training data', 'prediction'], loc='upper right')
    plt.show()
```



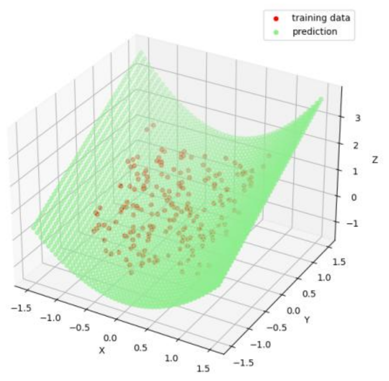
C = 0.1



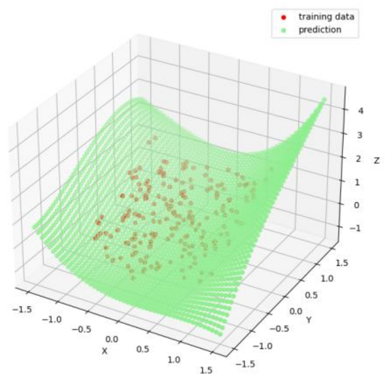
C = 1



C = 10



C = 100



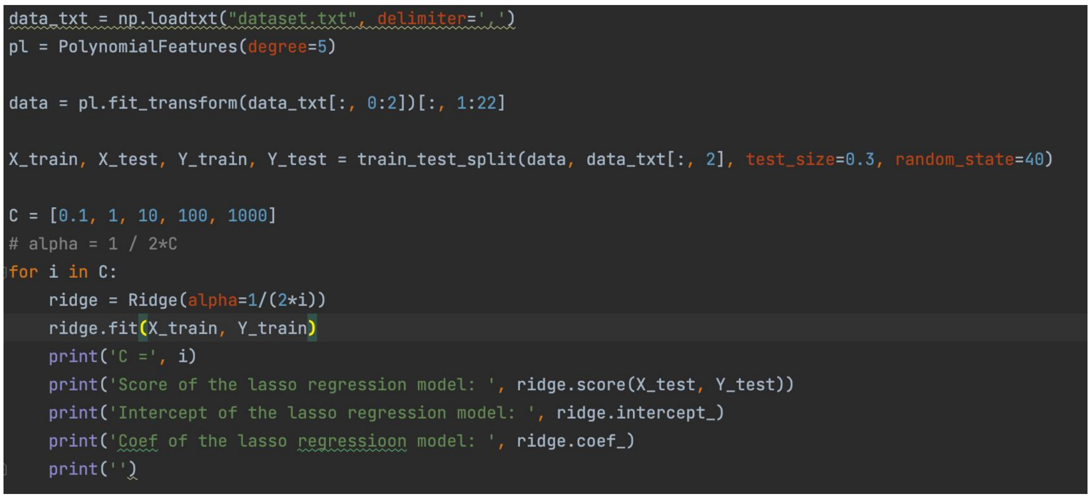
C = 1000

i (d) Over-fitting is the difference between the training error of the training set and the training error of the test set is large. Specifically, it means that the model performs well on the training set and poorly on the test set as the model fit the "noise" in the training data. Under-fitting is when the model performs poorly on both the training and test sets because the model is too simple to be able to capture the behavior of the data.

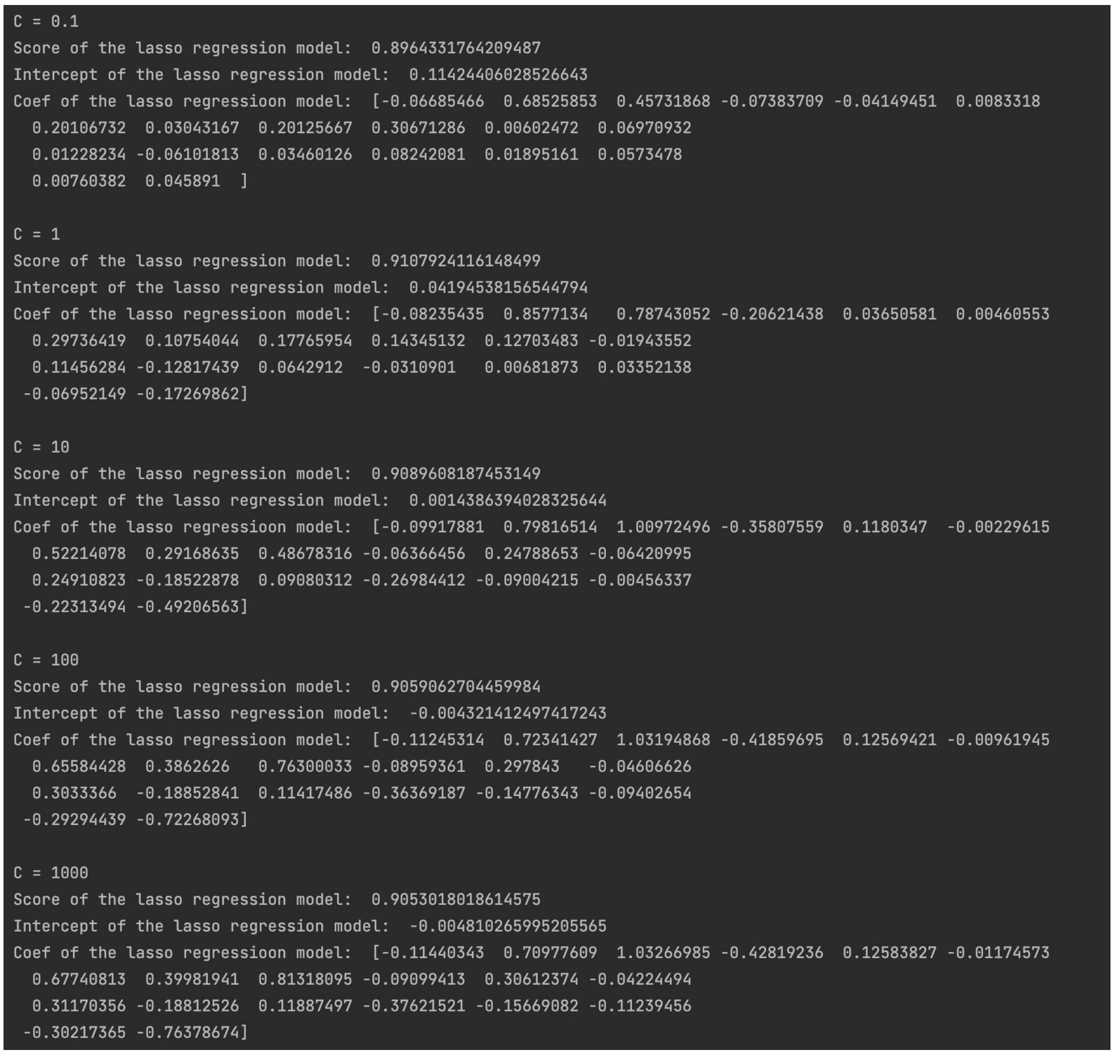
When C=0.1 or C=1, the parameters of the model are both 0, which corresponds to the model having no ability to predict the data at all, and is therefore an under-fitting. when C=100 the model predicts a higher score than when C=10, so the model is still an under-fitting when C=10. However, when C=1000, the model predicts a lower score than C=100, so this is an over-fitting.

i (e) The code idea is exactly the same as i(b), except that the machine learning model is changed to a Ridge Regression model.

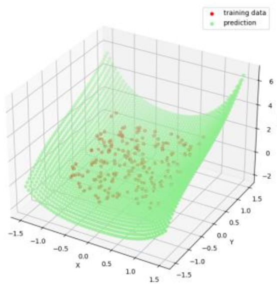
Lasso regression and ridge regression are both modifications of the cost function on top of standard linear regression and can be used to solve the overfitting problem of standard linear regression. The difference between them is that Lasso regression adds L1 regularisation to the cost function, while ridge regression adds L2 regularisation to the cost function, so that lasso regression is more likely to make the weights zero, while ridge regression is more likely to make the weights close to zero, as shown by comparing the parameters of the two models.



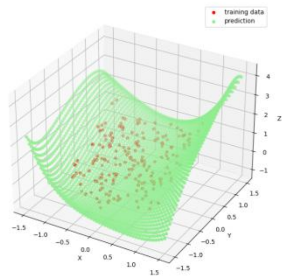
```python
data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)

data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

X_train, X_test, Y_train, Y_test = train_test_split(data, data_txt[:, 2], test_size=0.3, random_state=40)

C = [0.1, 1, 10, 100, 1000]
# alpha = 1 / 2*C
for i in C:
    ridge = Ridge(alpha=1/(2*i))
    ridge.fit(X_train, Y_train)
    print('C =', i)
    print('Score of the lasso regression model: ', ridge.score(X_test, Y_test))
    print('Intercept of the lasso regression model: ', ridge.intercept_)
    print('Coef of the lasso regressioon model: ', ridge.coef_)
    print('')
```



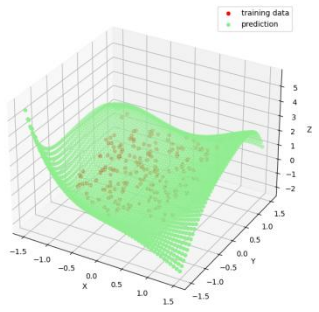
```
C = 0.1
Score of the lasso regression model:  0.8964331764209487
Intercept of the lasso regression model:  0.11424406028526643
Coef of the lasso regressioon model:  [-0.06685466  0.68525853  0.45731868 -0.07383709 -0.04149451  0.0083318
  0.20106732  0.03043167  0.20125667  0.30671286  0.00602472  0.06970932
  0.01228234 -0.06101813  0.03460126  0.08242081  0.01895161  0.0573478
  0.00760382  0.045891  ]

C = 1
Score of the lasso regression model:  0.9107924116148499
Intercept of the lasso regression model:  0.04194538156544794
Coef of the lasso regressioon model:  [-0.08235435  0.8577134   0.78743052 -0.20621438  0.03650581  0.00460553
  0.29736419  0.10754044  0.17765954  0.14345132  0.12703483 -0.01943552
  0.11456284 -0.12817439  0.0642912  -0.0310901   0.00681873  0.03352138
 -0.06952149 -0.17269862]

C = 10
Score of the lasso regression model:  0.9089608187453149
Intercept of the lasso regression model:  0.0014386394028325644
Coef of the lasso regressioon model:  [-0.09917881  0.79816514  1.00972496 -0.35807559  0.1180347  -0.00229615
  0.52214078  0.29168635  0.48678316 -0.06366456  0.24788653 -0.06420995
  0.24910823 -0.18522878  0.09080312 -0.26984412 -0.09004215 -0.00456337
 -0.22313494 -0.49206563]

C = 100
Score of the lasso regression model:  0.9059062704459984
Intercept of the lasso regression model:  -0.004321412497417243
Coef of the lasso regressioon model:  [-0.11245314  0.72341427  1.03194868 -0.41859695  0.12569421 -0.00961945
  0.65584428  0.3862626   0.76300033 -0.08959361  0.297843   -0.04606626
  0.3033366  -0.18852841  0.11417486 -0.36369187 -0.14776343 -0.09402654
 -0.29294439 -0.72268093]

C = 1000
Score of the lasso regression model:  0.9053018018614575
Intercept of the lasso regression model:  -0.004810265995205565
Coef of the lasso regressioon model:  [-0.11440343  0.70977609  1.03266985 -0.42819236  0.12583827 -0.01174573
  0.67740813  0.39981941  0.81318095 -0.09099413  0.30612374 -0.04224494
  0.31170356 -0.18812526  0.11887497 -0.37621521 -0.15669082 -0.11239456
 -0.30217365 -0.76378674]
```



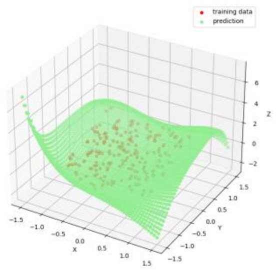
C = 0.1



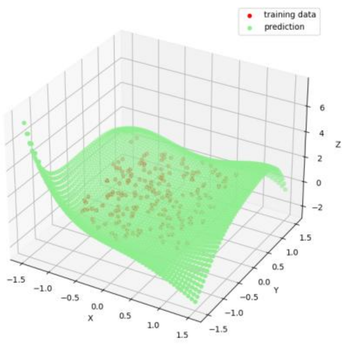
C = 1



C = 10

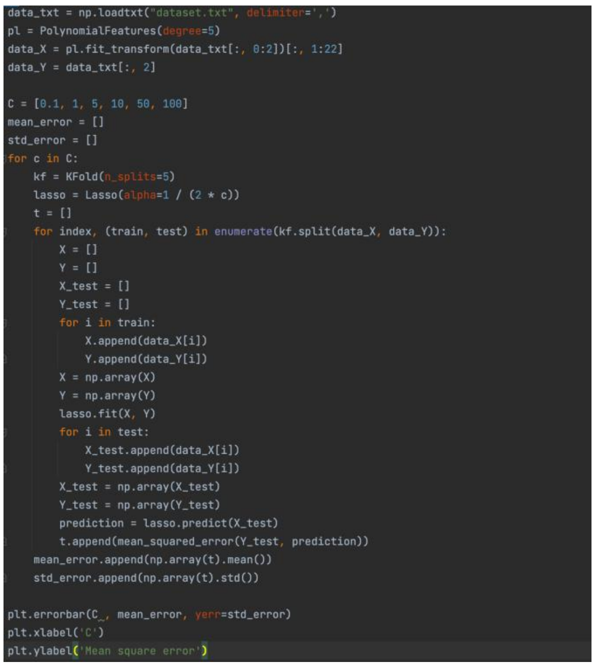


C = 100

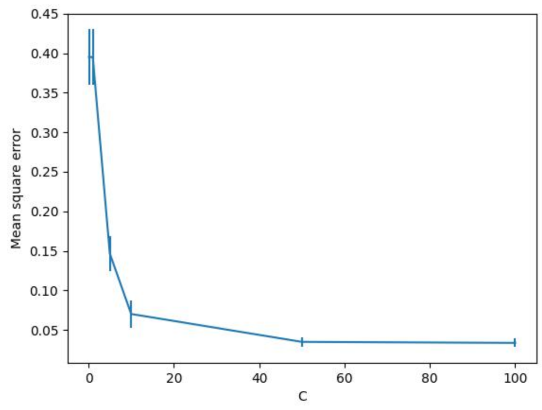


C = 1000

ii (a) I choose C of [0.1, 1, 5, 10, 50, 100], cycle the data set through the kFold function into 4 training sets and 1 test set, and calculate the square of the test error each time, find the mean and standard deviation of all errors, and finally use the errorbar function to plot the square of the mean error of the model for different C.

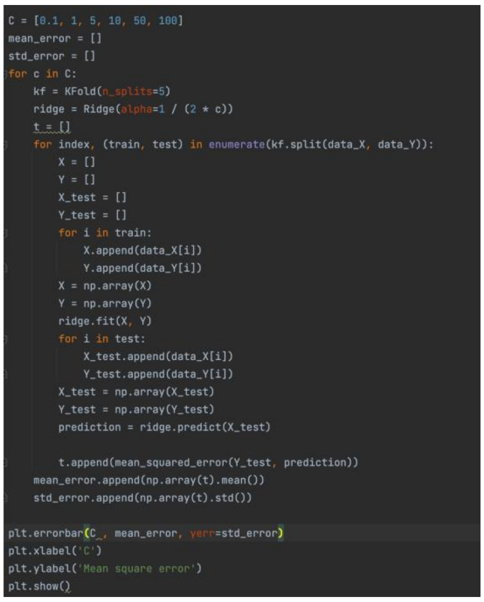


```
data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)
data_X = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]
data_Y = data_txt[:, 2]

C = [0.1, 1, 5, 10, 50, 100]
mean_error = []
std_error = []
for c in C:
    kf = KFold(n_splits=5)
    lasso = Lasso(alpha=1 / (2 * c))
    t = []
    for index, (train, test) in enumerate(kf.split(data_X, data_Y)):
        X = []
        Y = []
        X_test = []
        Y_test = []
        for i in train:
            X.append(data_X[i])
            Y.append(data_Y[i])
        X = np.array(X)
        Y = np.array(Y)
        lasso.fit(X, Y)
        for i in test:
            X_test.append(data_X[i])
            Y_test.append(data_Y[i])
        X_test = np.array(X_test)
        Y_test = np.array(Y_test)
        prediction = lasso.predict(X_test)
        t.append(mean_squared_error(Y_test, prediction))
    mean_error.append(np.array(t).mean())
    std_error.append(np.array(t).std())

plt.errorbar(C, mean_error, yerr=std_error)
plt.xlabel('C')
plt.ylabel('Mean square error')
```

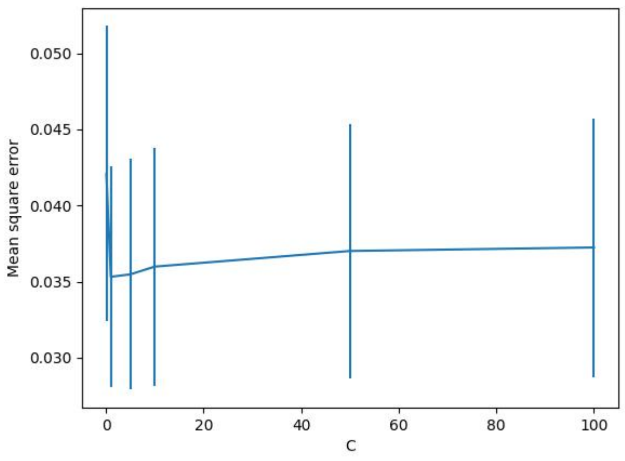


ii (b) I recommend using C=50. through the first question I found that as C gets smaller, the mean error gets larger, which means that the instability of the prediction is higher, so a C less than 50 is not recommended. When C is greater than 50, you can see from the graph that the average error is hardly reduced. From the previous question I know that the larger the C, the higher the probability of overfitting, so it is better to choose a smaller C when the prediction scores of C=50 and C=100 are close. However, this choice is very rough. If you want to choose a better C, you should set the interval of C values smaller and do experiments to get a C value with a smaller error and a higher prediction score.

ii (c) The code idea is exactly the same as ii(a), except that the machine learning model is changed to a Ridge Regression model. I recommend using C=1 because the error of C=1 in the Ridge model is relatively small and the prediction score is high.

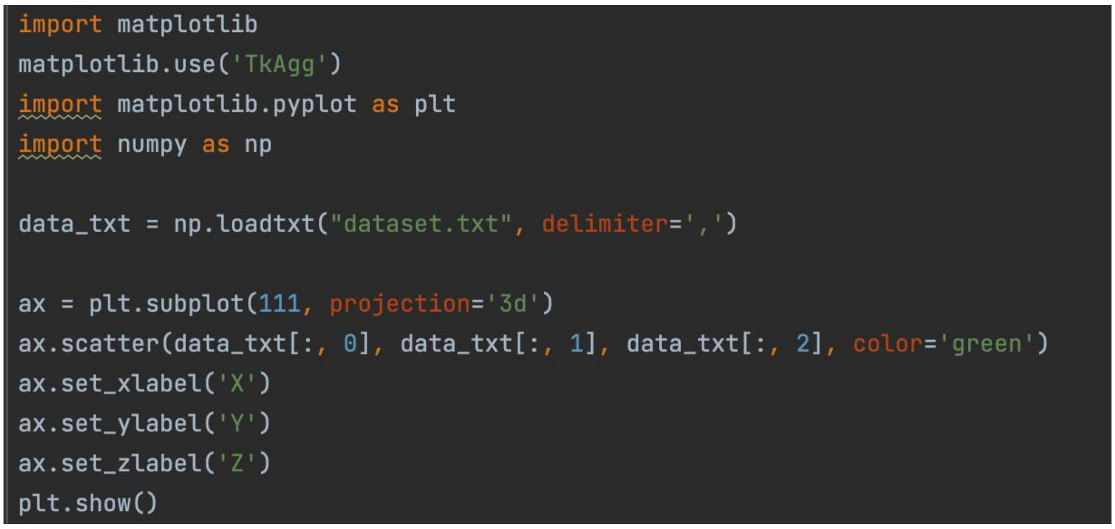


```
C = [0.1, 1, 5, 10, 50, 100]
mean_error = []
std_error = []
for c in C:
    kf = KFold(n_splits=5)
    ridge = Ridge(alpha=1 / (2 * c))
    t = []
    for index, (train, test) in enumerate(kf.split(data_X, data_Y)):
        X = []
        Y = []
        X_test = []
        Y_test = []
        for i in train:
            X.append(data_X[i])
            Y.append(data_Y[i])
        X = np.array(X)
        Y = np.array(Y)
        ridge.fit(X, Y)
        for i in test:
            X_test.append(data_X[i])
            Y_test.append(data_Y[i])
        X_test = np.array(X_test)
        Y_test = np.array(Y_test)
        prediction = ridge.predict(X_test)

        t.append(mean_squared_error(Y_test, prediction))
    mean_error.append(np.array(t).mean())
    std_error.append(np.array(t).std())

plt.errorbar(C, mean_error, yerr=std_error)
plt.xlabel('C')
plt.ylabel('Mean square error')
plt.show()
```



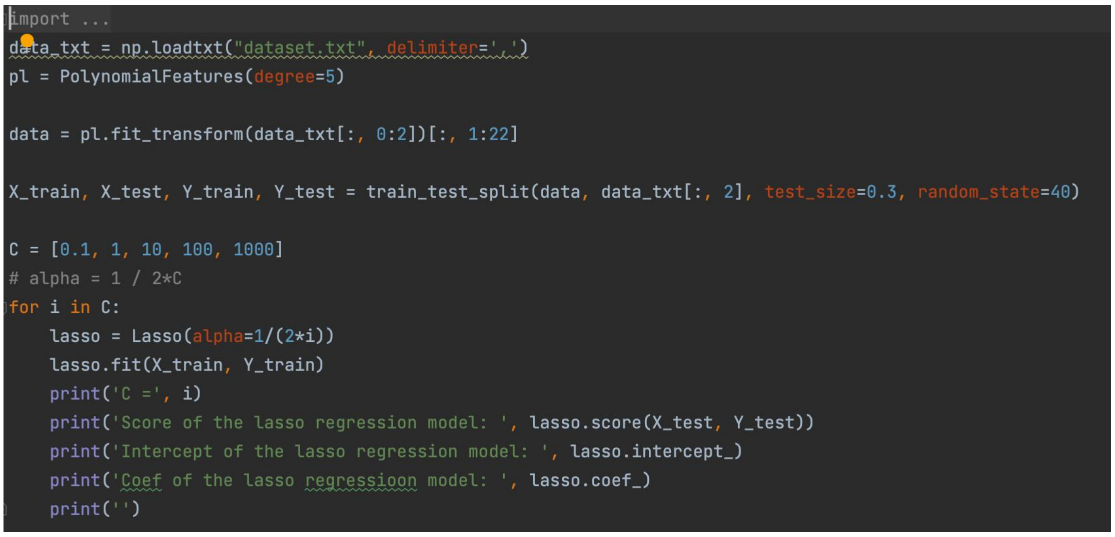
Appendix

i (a)



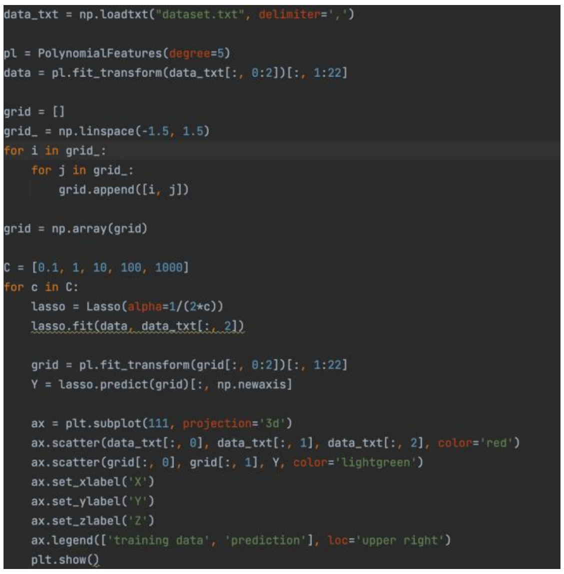
```python
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import numpy as np

data_txt = np.loadtxt("dataset.txt", delimiter=',')

ax = plt.subplot(111, projection='3d')
ax.scatter(data_txt[:, 0], data_txt[:, 1], data_txt[:, 2], color='green')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```

i (b)



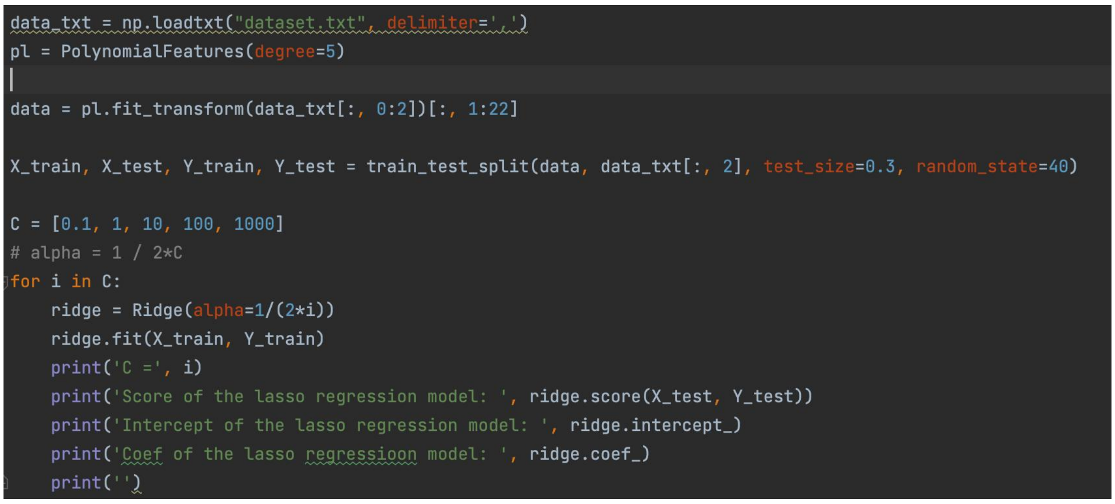
```python
import ...
data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)

data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

X_train, X_test, Y_train, Y_test = train_test_split(data, data_txt[:, 2], test_size=0.3, random_state=40)

C = [0.1, 1, 10, 100, 1000]
# alpha = 1 / 2*C
for i in C:
    lasso = Lasso(alpha=1/(2*i))
    lasso.fit(X_train, Y_train)
    print('C =', i)
    print('Score of the lasso regression model: ', lasso.score(X_test, Y_test))
    print('Intercept of the lasso regression model: ', lasso.intercept_)
    print('Coef of the lasso regressioon model: ', lasso.coef_)
    print('')
```

i (c)

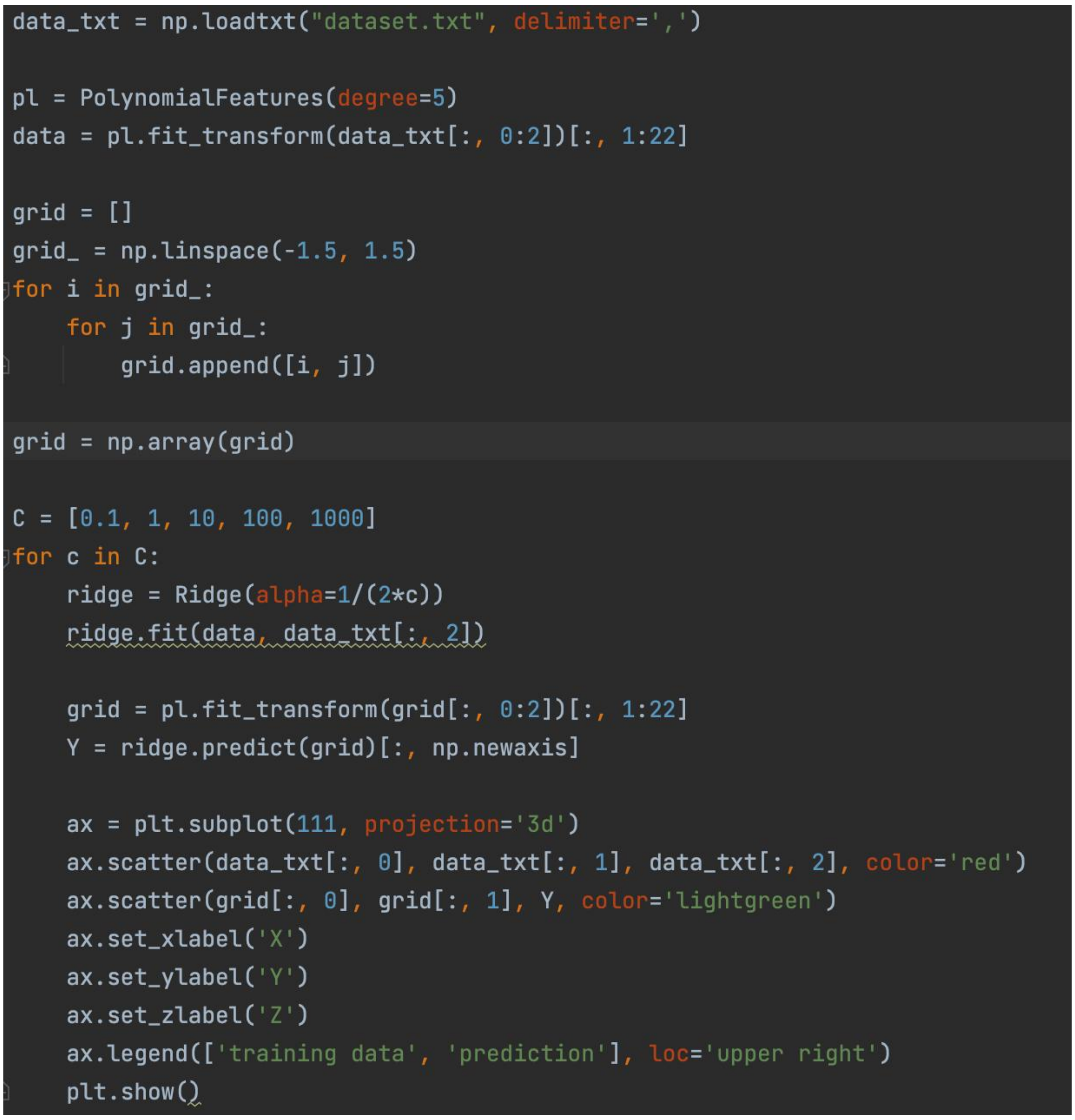


```python
data_txt = np.loadtxt("dataset.txt", delimiter=',')

pl = PolynomialFeatures(degree=5)
data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

grid = []
grid_ = np.linspace(-1.5, 1.5)
for i in grid_:
    for j in grid_:
        grid.append([i, j])

grid = np.array(grid)

C = [0.1, 1, 10, 100, 1000]
for c in C:
    lasso = Lasso(alpha=1/(2*c))
    lasso.fit(data, data_txt[:, 2])

    grid = pl.fit_transform(grid[:, 0:2])[:, 1:22]
    Y = lasso.predict(grid)[:, np.newaxis]

    ax = plt.subplot(111, projection='3d')
    ax.scatter(data_txt[:, 0], data_txt[:, 1], data_txt[:, 2], color='red')
    ax.scatter(grid[:, 0], grid[:, 1], Y, color='lightgreen')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.legend(['training data', 'prediction'], loc='upper right')
    plt.show()
```

i (e)

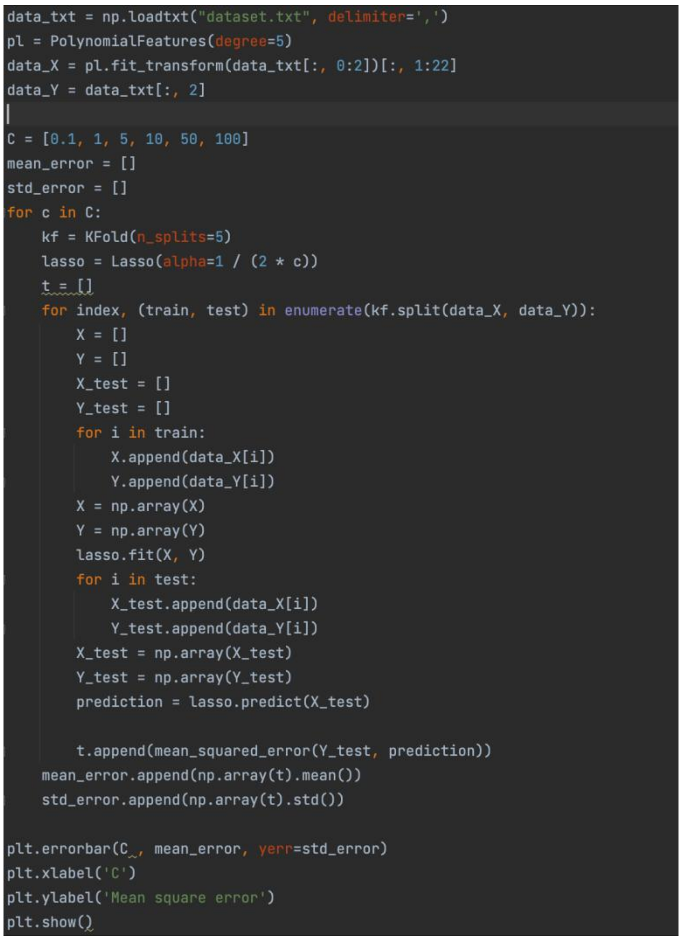


```python
data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)

data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

X_train, X_test, Y_train, Y_test = train_test_split(data, data_txt[:, 2], test_size=0.3, random_state=40)

C = [0.1, 1, 10, 100, 1000]
# alpha = 1 / 2*C
for i in C:
    ridge = Ridge(alpha=1/(2*i))
    ridge.fit(X_train, Y_train)
    print('C =', i)
    print('Score of the lasso regression model: ', ridge.score(X_test, Y_test))
    print('Intercept of the lasso regression model: ', ridge.intercept_)
    print('Coef of the lasso regressioon model: ', ridge.coef_)
    print('')
```

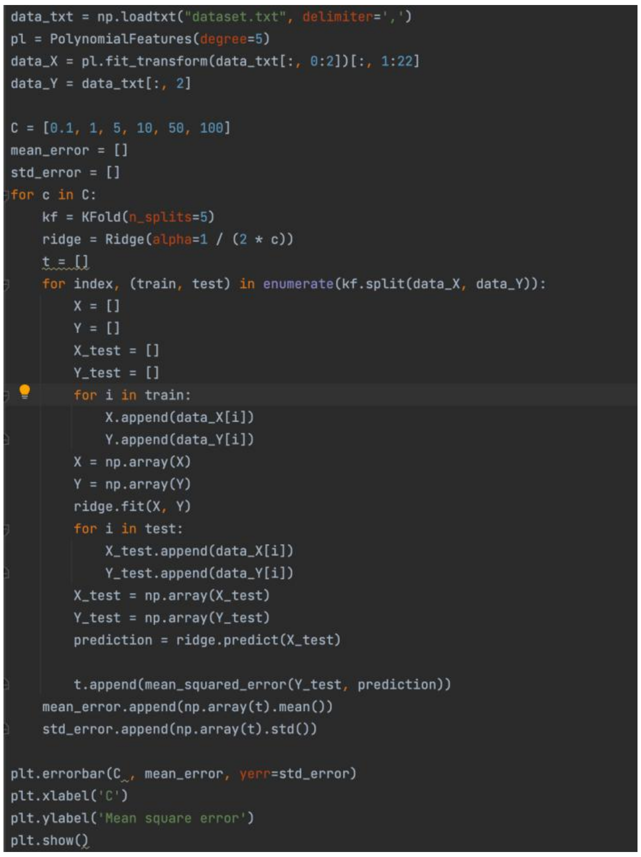


```python
data_txt = np.loadtxt("dataset.txt", delimiter=',')

pl = PolynomialFeatures(degree=5)
data = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]

grid = []
grid_ = np.linspace(-1.5, 1.5)
for i in grid_:
    for j in grid_:
        grid.append([i, j])

grid = np.array(grid)

C = [0.1, 1, 10, 100, 1000]
for c in C:
    ridge = Ridge(alpha=1/(2*c))
    ridge.fit(data, data_txt[:, 2])

    grid = pl.fit_transform(grid[:, 0:2])[:, 1:22]
    Y = ridge.predict(grid)[:, np.newaxis]

    ax = plt.subplot(111, projection='3d')
    ax.scatter(data_txt[:, 0], data_txt[:, 1], data_txt[:, 2], color='red')
    ax.scatter(grid[:, 0], grid[:, 1], Y, color='lightgreen')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.legend(['training data', 'prediction'], loc='upper right')
    plt.show()
```

ii



```
data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)
data_X = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]
data_Y = data_txt[:, 2]

C = [0.1, 1, 5, 10, 50, 100]
mean_error = []
std_error = []
for c in C:
    kf = KFold(n_splits=5)
    lasso = Lasso(alpha=1 / (2 * c))
    t = []
    for index, (train, test) in enumerate(kf.split(data_X, data_Y)):
        X = []
        Y = []
        X_test = []
        Y_test = []
        for i in train:
            X.append(data_X[i])
            Y.append(data_Y[i])
        X = np.array(X)
        Y = np.array(Y)
        lasso.fit(X, Y)
        for i in test:
            X_test.append(data_X[i])
            Y_test.append(data_Y[i])
        X_test = np.array(X_test)
        Y_test = np.array(Y_test)
        prediction = lasso.predict(X_test)

        t.append(mean_squared_error(Y_test, prediction))
    mean_error.append(np.array(t).mean())
    std_error.append(np.array(t).std())

plt.errorbar(C, mean_error, yerr=std_error)
plt.xlabel('C')
plt.ylabel('Mean square error')
plt.show()
```



```
data_txt = np.loadtxt("dataset.txt", delimiter=',')
pl = PolynomialFeatures(degree=5)
data_X = pl.fit_transform(data_txt[:, 0:2])[:, 1:22]
data_Y = data_txt[:, 2]

C = [0.1, 1, 5, 10, 50, 100]
mean_error = []
std_error = []
for c in C:
    kf = KFold(n_splits=5)
    ridge = Ridge(alpha=1 / (2 * c))
    t = []
    for index, (train, test) in enumerate(kf.split(data_X, data_Y)):
        X = []
        Y = []
        X_test = []
        Y_test = []
        for i in train:
            X.append(data_X[i])
            Y.append(data_Y[i])
        X = np.array(X)
        Y = np.array(Y)
        ridge.fit(X, Y)
        for i in test:
            X_test.append(data_X[i])
            Y_test.append(data_Y[i])
        X_test = np.array(X_test)
        Y_test = np.array(Y_test)
        prediction = ridge.predict(X_test)

        t.append(mean_squared_error(Y_test, prediction))
    mean_error.append(np.array(t).mean())
    std_error.append(np.array(t).std())

plt.errorbar(C, mean_error, yerr=std_error)
plt.xlabel('C')
plt.ylabel('Mean square error')
plt.show()
```