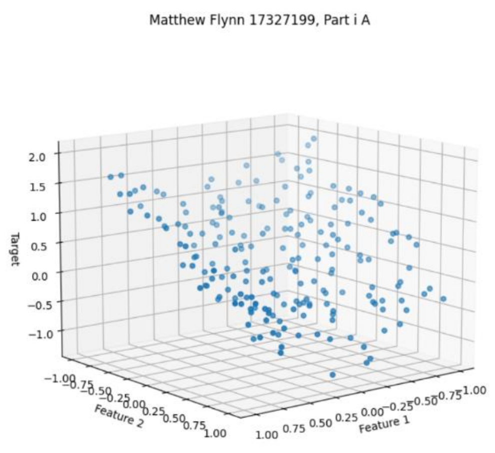
Machine Learning Week 3 assignment

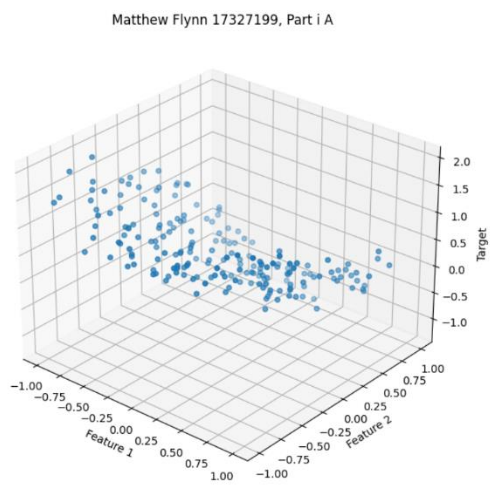
Matthew Flynn 17327199

Dataset Id : # id:7-7--7

Part I (a)

Test data scatter plot.





I took screenshots from 2 different angles to give a better idea of the shape of the data. The data looks like it is on a curve.

(b)

The values I set for C are the following, [1, 10, 100, 1000]. The values produced when using the Lasso model we get the following intercepts and coefficients:

C = 1
coef = [0, -0, -0, 0, 0, -0, -0, -0, -0, -0, 0, 0, 0, 0, -0, -0, -0, -0, -0, -0, -0]
intercept = 0.3324799191782795

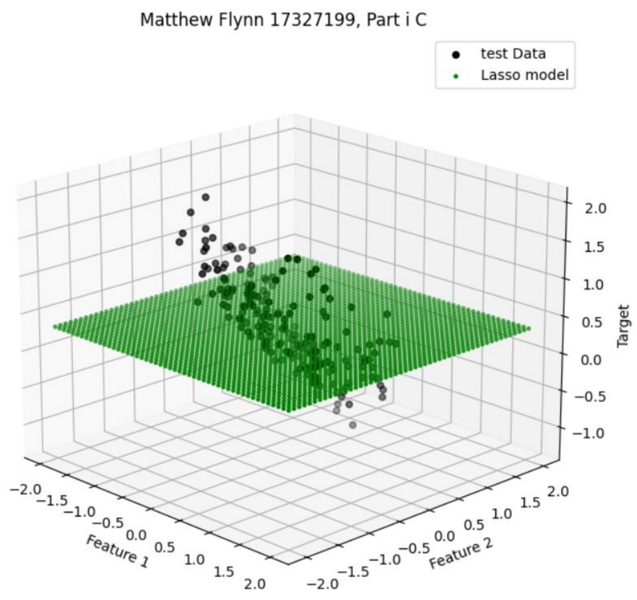
C = 10
coef = [0, -0, -0.79173723, 0.50283409, 0, -0, -0, -0, 0, -0, 0, 0, 0, 0, -0, -0, -0, -0, -0, 0, -0,]
intercept = 0.18161644115255401

C = 100
coef = [0, -0.00902392, -0.93099048, 0.86073334, 0, -0, -0.02576815, -0. 0.07342229, -
0.03072034, 0.17259307, -0, -0,-0, -0, -0, -0, 0, -0, 0,-0,]
intercept = 0.029072844294896094

C = 1000
coef = [0, -0.05175506 -0.92062334, 0.84874786, 0.01056326, 0.11373495, -0, -0.01145208,
0.20235292, -0, 0.22636628, 0, -0, -0.06343616, -0.09942005, -0.12249479, -0, 0.17080365,
0.03141969, 0, -0.12310162]
intercept = 0.0022298048700070128

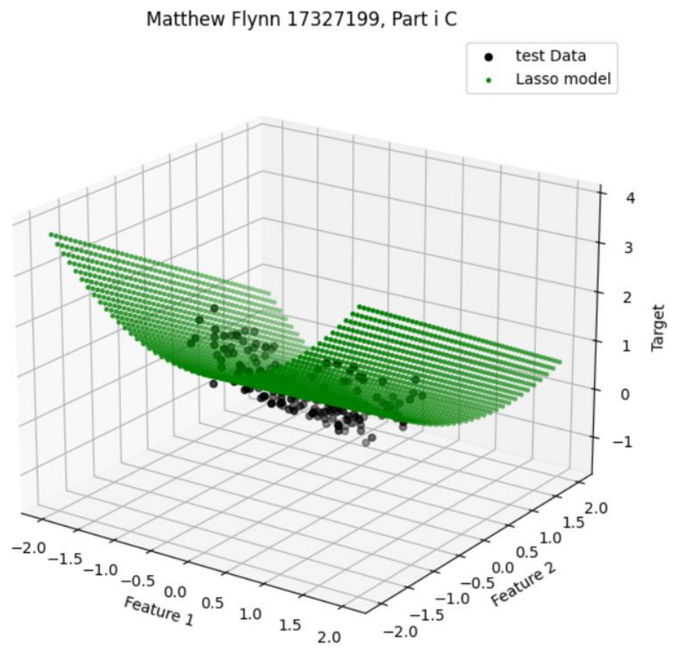
As C changes we get different values for the intercept and coef. Lasso uses the L1 penalty which tries to make as many Coef=0 as possible. So as C increases the penalty decreases. When C = 1 the penalty is the greatest and thus all the coefs are equal to 0. As the value of C increases the penalty decreases, therefore less and less of the coefficients are equal to 0.

(c)
Using the same values for c the following graphs are produced when the grid is extended beyond the dataset values.
C = 1



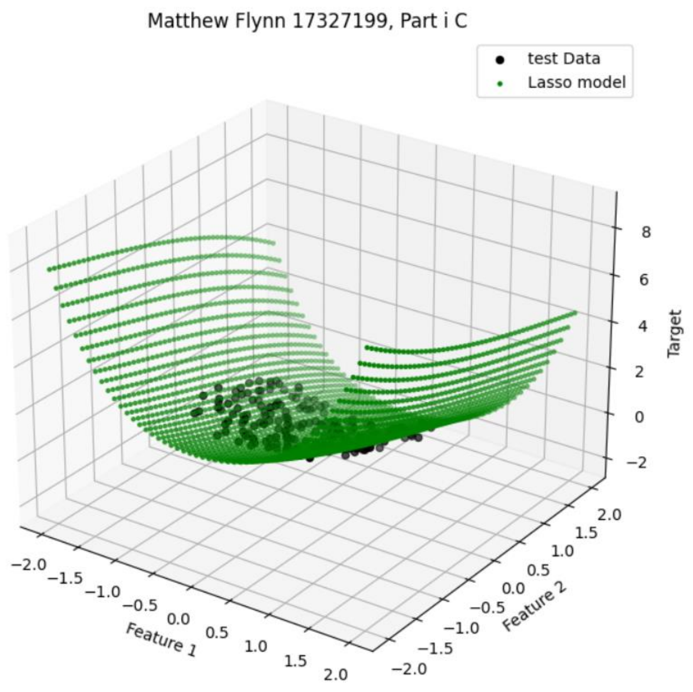
In this situation the penalty is too much and as a result the lasso model doesn't match the test data at all.

C = 10



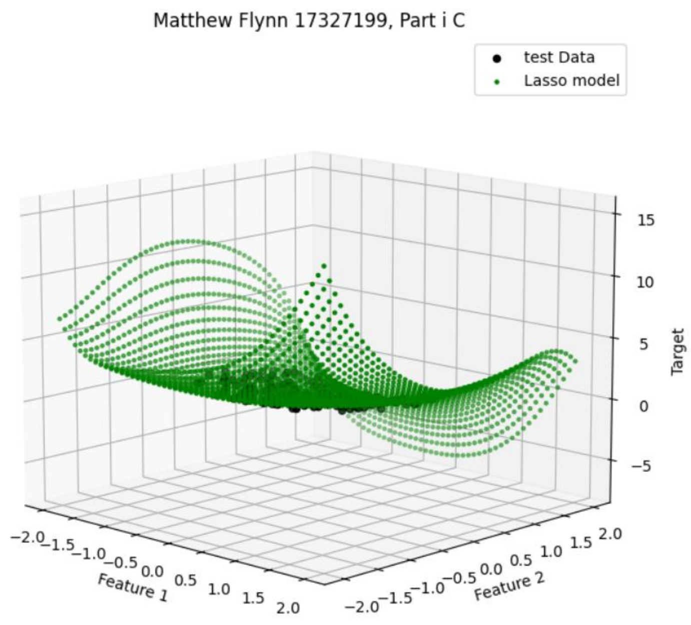
When the penalty is decreased the model begins to match the data better. Here it has a similar shape but still isn't matching the test data. There are still a large difference between the data and the model.

C = 100



When the penalty is this low the model matches the data much better. There is far less difference between the model and the test data

C = 1000



Here there is now warping of the model outside the test data. This could be due to overfitting as the data matches the test data with great accuracy however outside the test data causes the curve to have strange predictions.

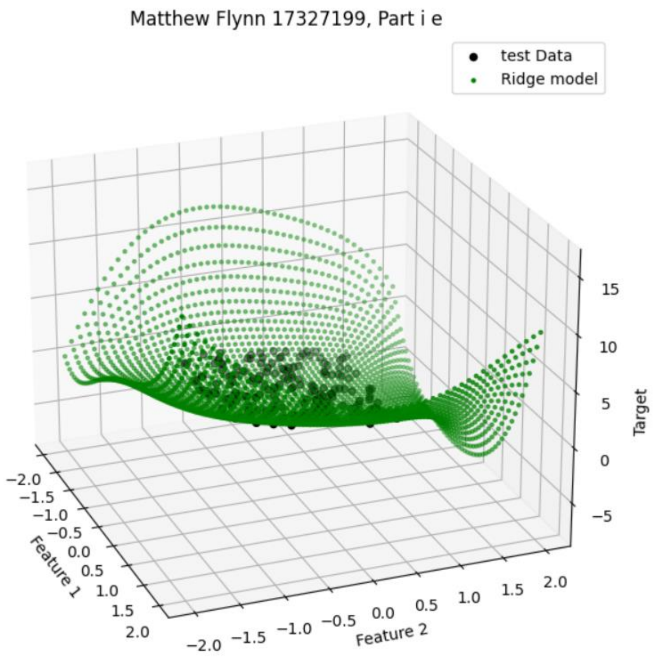
(d)

When the penalty is too great there is very little fitting occurring. Therefore, the model doesn't really match the test data and as such is not a good model. This is clear in the above example when C = 1. This means the penalty is too great and thus there is almost no fitting occurring. The inaccurate model produced is the result. This is underfitting

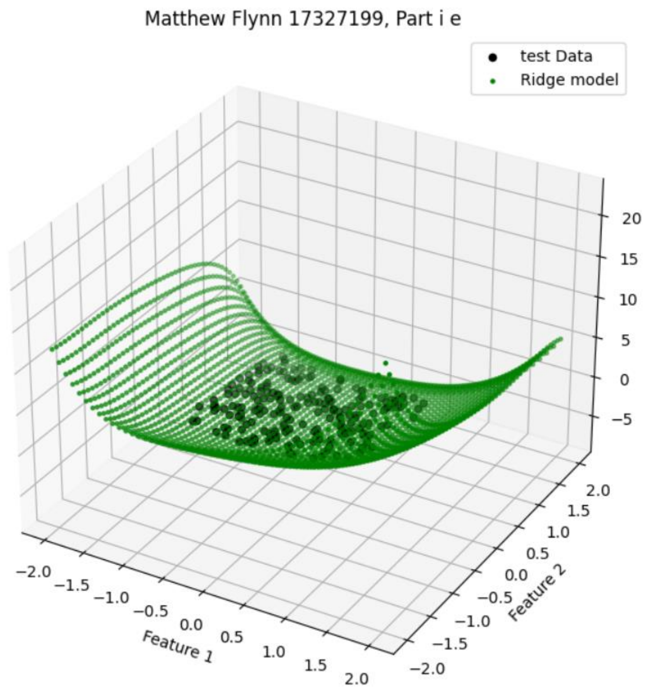
Overfitting is the opposite. This is when the model is so accurate to the test data that any results outside the test data become strange and warped. We can see this when C = 1000. This means the penalty is essentially too low and as a result the model matches the test data extremely closely but

once you go outside the test data the model warps and becomes very extreme and odd. This is overfitting.
Both overfitting and underfitting are not good for our model so we need to select a C so the penalty is reasonable. This is how we control the effect of over and underfitting using C.

(e)
When using the Ridge regression and the same variables for C and same test data the following coefficients and intercepts were produced. NOTE: in the below examples
C = 1
coef = [0, -0.0478487, -0.83582969, 0.72716212, 0.05911705, 0.15054047, -0.03645372, -0.12115554, 0.15961452, -0.1406845, 0.35521828,-0.01871295
-0.01171328, -0.12150327, -0.13153717, -0.08687087, -0.02045762, 0.176453,
0.19607801, 0.06470858, -0.08953504]
intercept = 0.011575051302860462

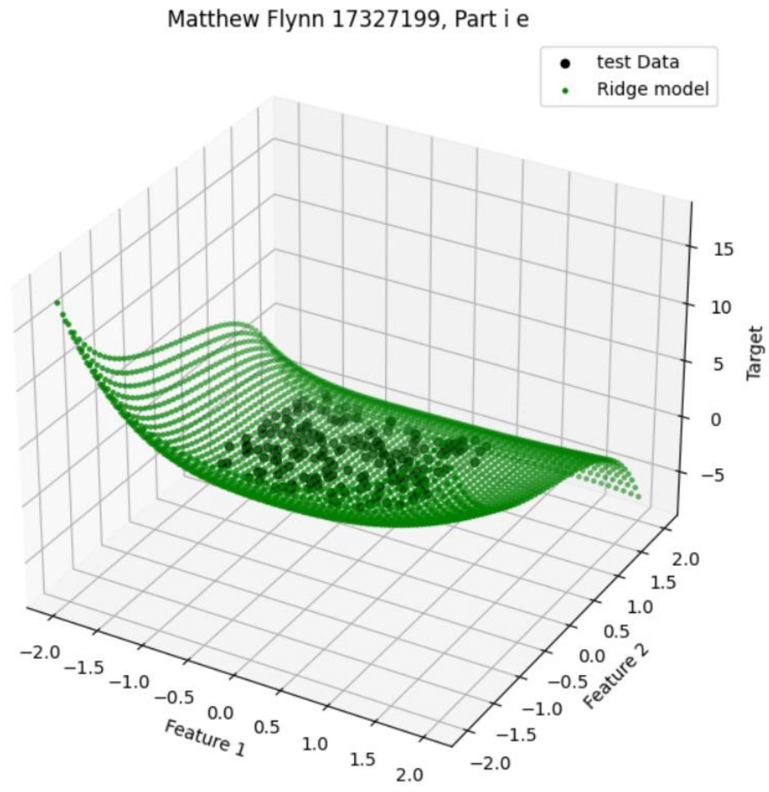


C = 10
```
coef =  [ 0.         -0.06444344 -0.85264001  0.87986479  0.03203218  0.3067771
  0.05957115 -0.28298197  0.2554263  -0.05319607  0.2280027   0.01921662
 -0.11662585 -0.1124265  -0.28031534 -0.20894961  0.06546439  0.24140296
  0.38543232 -0.11132865 -0.18534963]
```
intercept = -0.025551336358685395

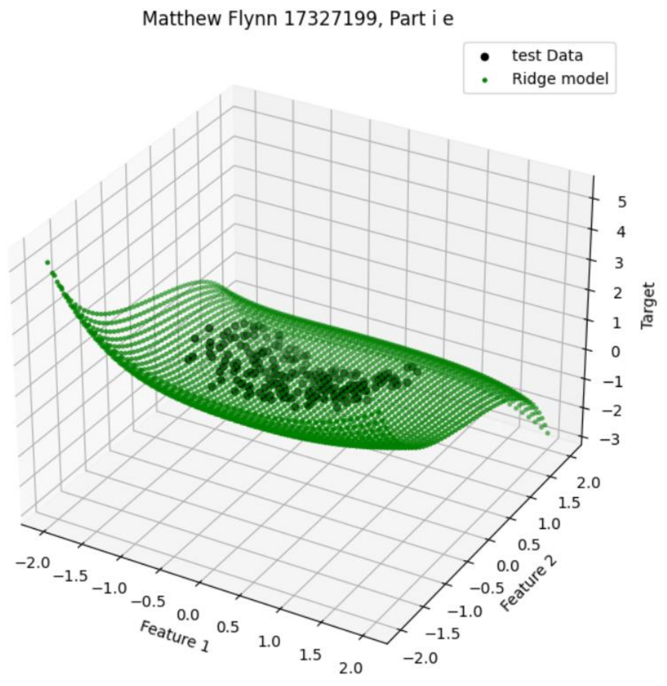


C = 100

```
coef =  [ 0.         -0.08321242 -0.84908402  0.92057565  0.00465893  0.35720069
  0.13236809 -0.37989351  0.32108402 -0.02732242  0.19006318  0.04487302
 -0.14779802 -0.09004729 -0.33086189 -0.27923662  0.13277938  0.23459284
  0.46080741 -0.19329722 -0.22035491]
intercept =  -0.03517266834278626
```



```
C = 1000
coef =  [ 0.00000000e+00 -8.63237280e-02 -8.48254187e-01  9.25861288e-01
  4.53897628e-04  3.63923041e-01  1.44013713e-01 -3.94313402e-01
  3.31076161e-01 -2.44926682e-02  1.85086695e-01  4.87303498e-02
 -1.52024593e-01 -8.64855227e-02 -3.37628382e-01 -2.89962523e-01
  1.43116593e-01  2.32331886e-01  4.71416613e-01 -2.04828533e-01
 -2.24640043e-01]
intercept =  -0.03641092343954505
```



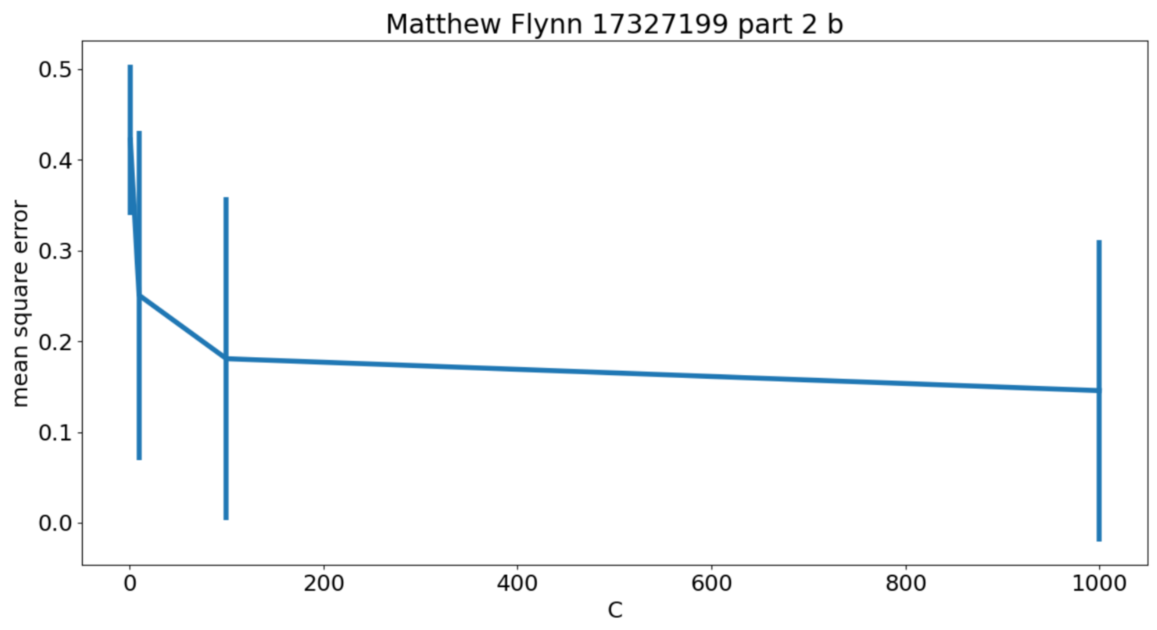
As can be observed the model is far more warped when the penalty is high then when the penalty is low in ridge regression. This is the reverse of the lasso model.

Part ii a

```
5 fold ->       mean =  -0.42166755646783455 +/-  0.08294250227675941
10 fold ->      mean =  -0.4223772135788445 +/-  0.1281452022237
25 fold ->      mean =  -0.42305625077204945 +/-  0.1517446198208338
50 fold ->      mean =  -0.4232919078004458 +/-  0.2433178250540198
100 fold ->     mean =  -0.4209724272013984 +/-  0.36971260213465107
```

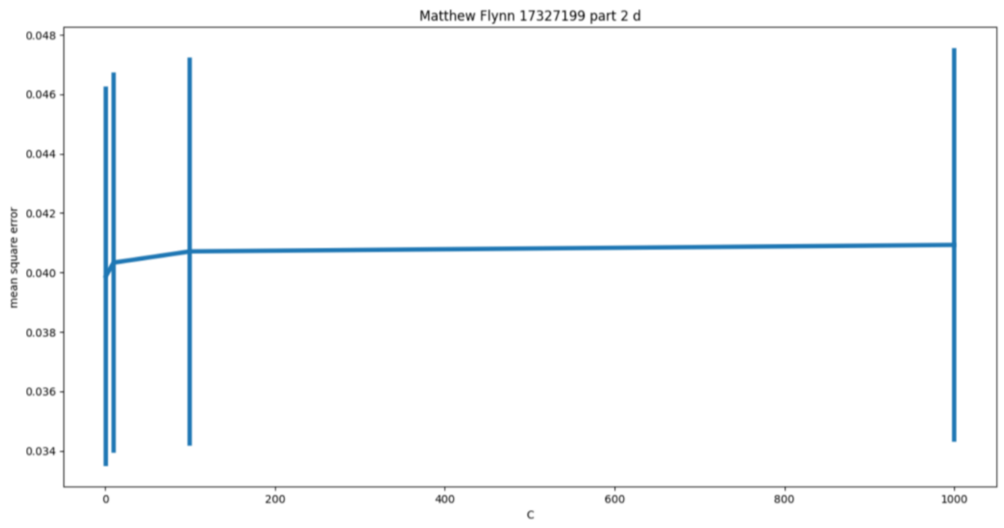
Our goal in this is to find the lowest mean with the lowest variance. All above means are similar however the lowest mean with the lowest variance is when we have 5 folds.

b



I used K = 5 because of the reasons given in part a. it has the least variation and the smallest mean. I choose C = [1,10,100,1000] to demonstrate the levelling off effect after c reaches a certain value. There is a large change in the mean square error when the value goes from 1 to 10 to 100 but once it gets above 100 the values decreases at a very slow rate.

c. Based on this cross validation data I could choose to use C = 100 as it has reached a levelling off point and the variance is about the same for all except when C=1. Therefore when C = 100 the mean square error as the lowest and the variance is the lowest and there is not a bunch of unnecessary calculations which need to happen for the C = 1000



d

Code Appendix:

```python
import numpy as np
import pandas as pd
from array import array
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error


def plot3d(Xtest,Ytest,x,y,legend):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection ='3d')
    ax.scatter(x[:,0],x[:,1],y, color='black')
    ax.scatter(Xtest[:,0],Xtest[:,1],Ytest, color='green', marker='.')
    ax.legend(legend)
    ax.set_title("Matthew Flynn 17327199, Part i C")
    ax.set_xlabel("Feature 1")
    ax.set_ylabel("Feature 2")
    ax.set_zlabel("Target")


# read in the data taken from week 2
# (i) a, Read in data and display it in a 3d diagram
df = pd.read_csv("week3.csv", comment ="#")
x1 = df.iloc[:, 0]
x2 = df.iloc[:, 1]
x = np.column_stack((x1, x2))
y = df.iloc[:, 2]

plot 3d figure. taken from his code
fig = plt.figure()
ax = fig.add_subplot(111, projection ='3d')
ax.scatter(x[:,0],x[:,1],y)
ax.set_title("Matthew Flynn 17327199, Part i A")
ax.set_xlabel("Feature 1")
ax.set_ylabel("Feature 2")
ax.set_zlabel("Target")
plt.show()

# (i) b add polynomial features up to power of 5
# then train a lasso regression model for C = 1, 10 ,1000
Xpoly = PolynomialFeatures(5).fit_transform(x)
C = [1,10, 100,1000]
```

```
for i in C:
    #print()
    clf = linear_model.Lasso(alpha = (1/(2*i)))
    lasso_model = clf.fit(Xpoly, y)
    print("coef = ", lasso_model.coef_)
    print("intercept = ", lasso_model.intercept_)

# (i) c predictions vs test data.
Xtest = []
Xpoly = PolynomialFeatures(5).fit_transform(x)
fig = plt.figure()
grid = np.linspace(-2 ,2)
for i in grid:
    for j in grid:
        Xtest.append([i,j])
Xtest = np.array(Xtest)
XtestPoly = PolynomialFeatures(5).fit_transform(Xtest)
C = [1,10, 100, 1000]
legend = ['test Data', 'Lasso model']
for i in C:
    clf = linear_model.Lasso(alpha = (1/(2*i)))
    lasso_model = clf.fit(Xpoly, y)
    Ytest = clf.predict(XtestPoly)
    plot3d(Xtest,Ytest,x,y,legend)
    plt.show()

# (e)
then train a Ridge regression model for C = 1, 10 ,1000
Xpoly = PolynomialFeatures(5).fit_transform(x)
C = [1,10, 100,1000]
for i in C:
    clf = linear_model.Ridge(alpha = (1/(2*i)))
    ridge_model = clf.fit(Xpoly, y)
    print("coef = ", ridge_model.coef_)
    print("intercept = ", ridge_model.intercept_)

Xtest = []
Xpoly = PolynomialFeatures(5).fit_transform(x)
grid = np.linspace(-2 ,2)
for i in grid:
    for j in grid:
        Xtest.append([i,j])
Xtest = np.array(Xtest)
XtestPoly = PolynomialFeatures(5).fit_transform(Xtest)
C = [1,10, 100, 1000]
legend = ['test Data', 'Ridge model']
for i in C:
    fig = plt.figure()
    clf = linear_model.Ridge(alpha = (1/2*i))
    ridge_model = clf.fit(Xpoly, y)
    Ytest = clf.predict(XtestPoly)
    ax = fig.add_subplot(111, projection ='3d')
    ax.scatter(x[:,0],x[:,1],y, color='black')
    ax.scatter(Xtest[:,0],Xtest[:,1],Ytest, color='green', marker='.')
    ax.legend(legend)
    ax.set_title("Matthew Flynn 17327199, Part i e")
    ax.set_xlabel("Feature 1")
    ax.set_ylabel("Feature 2")
    ax.set_zlabel("Target")
    plt.show()



# ii (a)
c_start = 1
x_Poly = PolynomialFeatures(5).fit_transform(x)
fold = [5,10,25,50,100]
for i in fold:
    k_f = KFold(n_splits=i)
    model = linear_model.Lasso(alpha=(1/(2*c_start)))
    for trainer, test in k_f.split(x_Poly):
        model.fit(x_Poly[trainer], y[trainer])
    scores = cross_val_score(model, x_Poly, y, cv=i, scoring="neg_mean_squared
_error")
    print("mean = ", scores.mean(), "+/- ", scores.std())

# ii graphing b
x_Poly = PolynomialFeatures(5).fit_transform(x)
mean_error = []
std_error =[]
tmp =[]
C = [1,10, 100,1000]
k_f = KFold(n_splits=5)
for i in C:
    model = linear_model.Lasso(alpha=1/(2*i))
    for trainer , tester in k_f.split(x_Poly):
        model.fit(x_Poly[trainer], y[trainer])
        Ytest = model.predict(x_Poly[tester])
        tmp.append(mean_squared_error(y[tester],Ytest))
    mean_error.append(np.array(tmp).mean())
    std_error.append(np.array(tmp).std())
plt.rc('font',size = 18)
plt.title("Matthew Flynn 17327199 part 2 b")
plt.rcParams['figure.constrained_layout.use'] = True
plt.errorbar(C , mean_error ,yerr=std_error, linewidth=4)
plt.xlabel("C")
plt.ylabel("mean square error")
```

```python
plt.show()


# ii (d)
mean_error = []
std_error =[]
tmp =[]
C = [1,10, 100,1000]
x_Poly = PolynomialFeatures(5).fit_transform(x)
for i in C:
    k_f = KFold(n_splits=10)
    model = Ridge(alpha=1/(2*i))
    for trainer , tester in k_f.split(x_Poly):
        model.fit(x_Poly[trainer], y[trainer])
        Ytest = model.predict(x_Poly[tester])
        tmp.append(mean_squared_error(y[tester],Ytest))
    mean_error.append(np.array(tmp).mean())
    std_error.append(np.array(tmp).std())
plt.errorbar(C , mean_error ,yerr=std_error, linewidth=4)
plt.title("Matthew Flynn 17327199 part 2 d")
plt.xlabel("C")
plt.ylabel("mean square error")
plt.show()
```