Week 3 Assignment

CS7CS4/CSU44061 Machine Learning

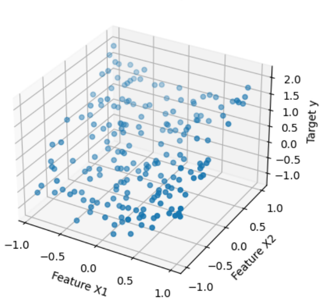
Cormac Madden – 18319983

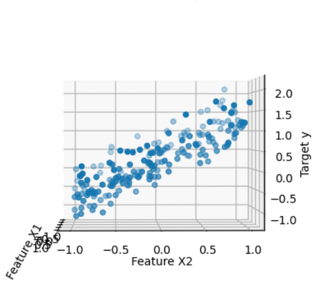
Dataset ID: # id:11-11-11

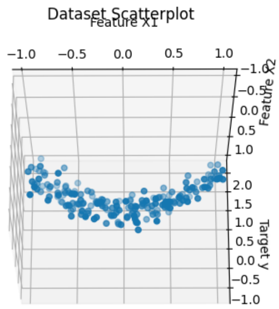
1)

a) I plotted the data using the code provided in the assignment.

It appears the training data lies on a curved plane. From the X2,Y perspective the data appears linear (Fig 2), however from another perspective the data appears to follow a quadratic curve (Fig 3).







b) Below are the features of the model for the value of C:1,10,100,1000,10000

00: 0.0	00: 0.0	00: 0.0	00: 0.0	00: 0.0
01: -0.0	01: -0.0	01: -0.0006400523883016784	01: -0.001973981623250821	01: 0.03432405104559226
02: 0.0	02: 0.8616309233753672	02: 0.9766899233130936	02: 1.003330346104757	02: 1.1123546020698984
03: 0.0	03: 0.40742645751449386	03: 0.9242314532863658	03: 0.9957504969950444	03: 1.0326335840043146
04: -0.0	04: -0.0	04: -0.0	04: -0.0	04: 0.0
05: 0.0	05: 0.0	05: -0.0	05: -0.0	05: -0.07857668674130136
06: -0.0	06: -0.0	06: -0.0	06: -0.0	06: -0.0247431333566697
07: 0.0	07: 0.0	07: 0.0	07: -0.0	07: -0.3542807447781366
08: -0.0	08: -0.0	08: -0.0	08: -0.018799465359951584	08: -0.16986275794985073
09: 0.0	09: 0.0	09: 0.0	09: -0.043847964853931345	09: -0.2966456527904106
10: 0.0	10: 0.0	10: 0.0	10: 0.0	10: -0.0
11: -0.0	11: -0.0	11: -0.0	11: 0.0	11: 0.012258172583318924
12: 0.0	12: 0.0	12: 0.0	12: -0.0451804756843597	12: -0.12279392940046434
13: -0.0	13: -0.0	13: -0.0	13: -0.038513587183262636	13: -0.07143498092555088
14: 0.0	14: 0.0	14: -0.0	14: 0.0	14: 0.10976189083633862
15: -0.0	15: 0.0	15: -0.0	15: -0.006796150007602049	15: -0.06116497721561562
16: 0.0	16: 0.0	16: 0.0	16: 0.0535067861013716	16: 0.4622029692061395
17: -0.0	17: -0.0	17: -0.0	17: 0.0	17: 0.24269383848967147
18: 0.0	18: 0.0	18: 0.0	18: -0.0	18: -0.008301016178062573
19: -0.0	19: -0.0	19: -0.0	19: -0.0	19: 0.0
20: 0.0	20: 0.0	20: 0.0	20: -0.0	20: 0.1903635380145325

J(θ) = mean squared error

C	1	10	100	1000	10000
J(θ)	0.488932	0.071785	0.037024	0.036140	0.035303

The first output has 0 features because the penalty of C=1 is too strong. It gives the same mean squared error as the baseline (0.488932) which is just the mean. When C=1 there are two features, for b and a^2. This reduces the error significantly to 0.071785, and when drawn on a graph would show the curve we predicted in a).

For C=100-10000 more features are added, which adds more curves to the prediction model. This reduces the error but may also be unnecessary and be over-fitting the data. A simpler model such as for C=100 is probably adequate, as the decrease in error is marginal after that.

Below is the code used to create the data above:

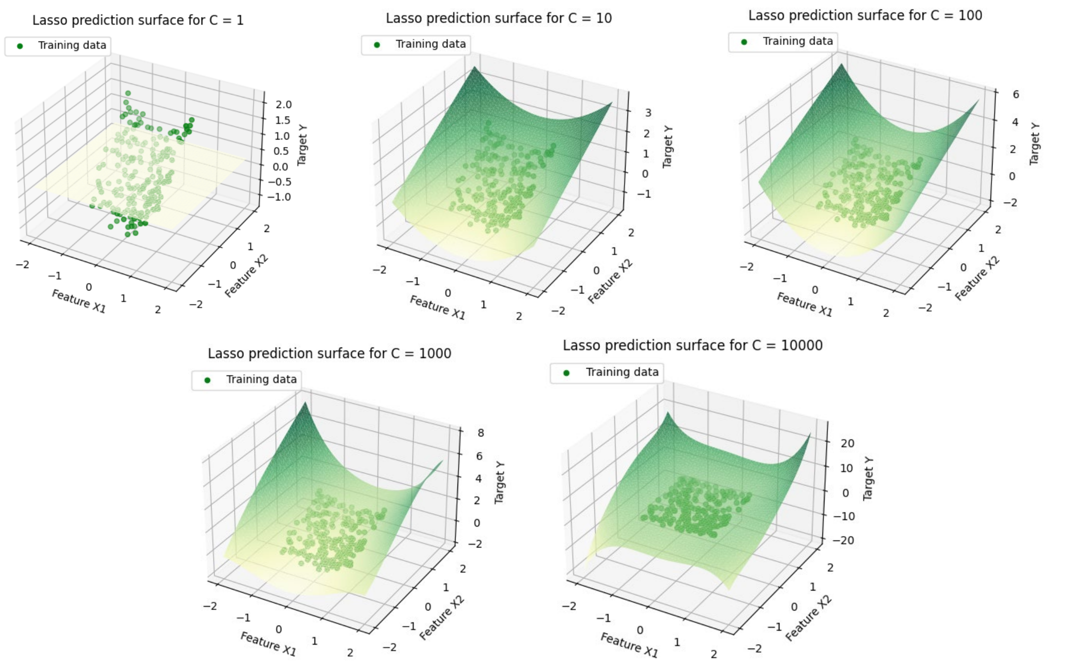
```
#Xpoly
Xpoly = PolynomialFeatures(degree = 5).fit_transform(X)

baseline = DummyRegressor(strategy="mean").fit(Xpoly, y)
print("J(θ_baseline) = %f\n"%mean_squared_error(y, baseline.predict(Xpoly)))

C = [1,10,100,1000,10000]
lassos = []
for Ci in C:
    model = Lasso(alpha=1/(2*Ci)).fit(Xpoly,y)
    lassos.append(model)
    print("\nC value: "+str(Ci))
    print("Lasso coef: "+str(model.coef_))
    print("Lasso intercept: "+str(model.intercept_))
    print("J(θ) = %f\n"%mean_squared_error(y, model.predict(Xpoly)))

```

c)



I decided to show the predictions on several different plots, in order to make it easier to read.

As C is increased the curve fits more tightly to the training data, but as the plane extends from the training data is bends sharply in unexpected ways. Note the difference in scale on the Target Y axis on the five plots. It is possible but highly unlikely that this is representative of the data. Where C=10 or 100 it looks as we had anticipated in question a) and would extend out in that logical direction.

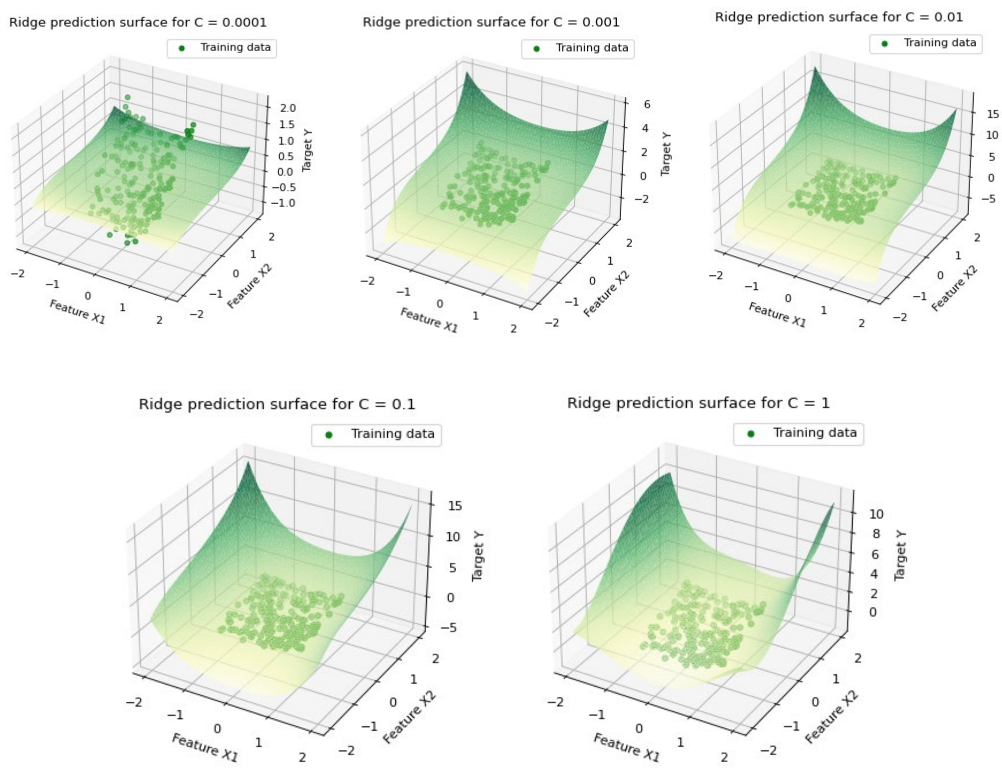
For the higher C values, smaller clusters of data at the edge of the dataset are having a significant impact on the direction of the extremities of the prediction plane.

d) Under-fitting is when the model is too simple to represent the data. It will struggle to predict the data and example of this can be seen in the first graph from question c).

Over-fitting is where the model tries to adapt to too much of the training data and as a result performs poorly on new data. An example of this is in the last graph from question c)

The penalty weight parameter C can be used to manage how closely the model fits the training data. As can be seen in the figures above, when a small value of C is used, the model uses less features and can tend towards underfitting the data. Whereas when large values of C are used the model uses more features and tends towards overfitting the data.

e) The following figures are the graphs produced from creating a Ridge Regression model and using various values for C. C in this case affects the cost function and is referred to as a hyperparameter.



For this model all the features are used but the penalty affects the impact the training data will have on the prediction curve. As you can see from the data all the features in the first model are close to zero, whereas in the later models some of the feaures increase quite significantly.

00: 0.0	00: 0.0	00: 0.0	00: 0.0	00: 0.0
01: -0.0018513651828837591	01: -0.013330541577483172	01: -0.034032317531469625	01: -0.029105927295751302	01: 0.006093259509003689
02: 0.014194453771848754	02: 0.11513199493041919	02: 0.4260062350189855	02: 0.7561249152094907	02: 0.965960954977086
03: 0.0035243390931487457	03: 0.03315752063960001	03: 0.21129711408975393	03: 0.5041525111820386	03: 0.8071335008486811
04: -0.0007481686884566327	04: -0.006379643472456882	04: -0.024287495958057466	04: -0.034084035896193125	04: -0.037143117308321115
05: 0.00025606870782667203	05: 0.002045458423798274	05: 0.0008697514802404693	05: -0.030892860287433918	05: -0.060137417057416265
06: -0.0007372033614864935	06: -0.004501466288812932	06: -0.0006119147168303501	06: -0.0010475454119090455	06: -0.005051751541955235
07: 0.004638968776310768	07: 0.03649474457481089	07: 0.1081360793099748	07: 0.09247861996938787	07: -0.029819335484788146
08: -0.0008679906646579862	08: -0.00634923179635491	08: -0.01710003145912746	08: -0.020276374497566852	08: -0.06269900731619074
09: 0.008950464037193	09: 0.07122831788494287	09: 0.22361959208649293	09: 0.19137605018888834	09: 0.026371912597227088
10: 0.0028927044648048962	10: 0.027360990601534428	10: 0.1737728369433076	10: 0.34671276841008775	10: 0.20084825397215625
11: -0.0004921312937470811	11: -0.004345920624407818	11: -0.018857817569212836	11: -0.009862441657054199	11: 0.03683160072611849
12: 0.001424434536115778	12: 0.013246774186073165	12: 0.07731571408693153	12: 0.10947378851167515	12: -0.040103495499691104
13: -0.0004583930655945829	13: -0.003713453923592708	13: -0.009065301552519977	13: 0.003570749948975174	13: -0.03320793560182124
14: 0.00034591643216745404	14: 0.002801097026853019	14: 0.005418803216653	14: -0.0016444618291025555	14: 0.06520852569949732
15: -0.0002869785932901978	15: -0.000980489980499502	15: 0.012528857715671762	15: 0.0129259663644238	15: -0.04107792585690653
16: 0.0027863644785770187	16: 0.021673248069543787	16: 0.059389176033251986	16: 0.05492949092297768	16: 0.16416912466851666
17: -0.00035422246270718367	17: -0.0022010202553973597	17: -7.645763864313644e-05	17: 0.011957718728948238	17: 0.09998717276097367
18: 0.0030150128928475906	18: 0.023313344704950858	18: 0.056786099582813124	18: -0.017934439381892774	18: -0.0862213682738656
19: -0.0004757552507678632	19: -0.003396384050024869	19: -0.00849191527059129	19: -0.011958204256844008	19: -0.01915017463942704
20: 0.006594857282701048	20: 0.051893505608734264	20: 0.14595728598831176	20: 0.041131057875342616	20: -0.012976547887069164

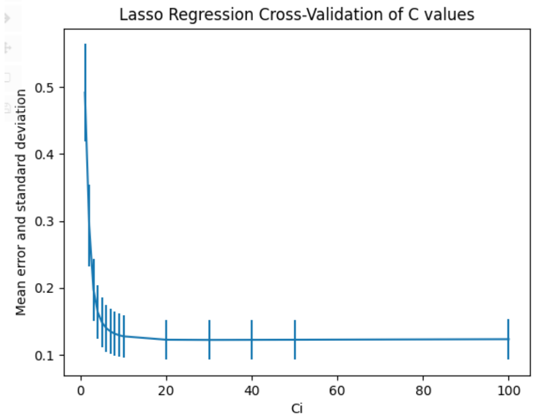
The mean square error J(θ) for each Ridge model was also found using the same baseline as in question (i)(b) and (c), and is shown in the table below.

C	0.0001	0.001	0.01	0.1	1
J(θ)	0.468832	0.341606	0.106170	0.042716	0.036295

There is a notable difference between the Ridge and Lasso model, in that the Lasso model will cancel small parameters which makes it easy to classify the data as linear, quadratic etc. The Ridge model on the other hand only adjusts the significance of each parameter without cancelling any. This means the model will always be more complex and difficult to interpret and manually manipulate.

2)

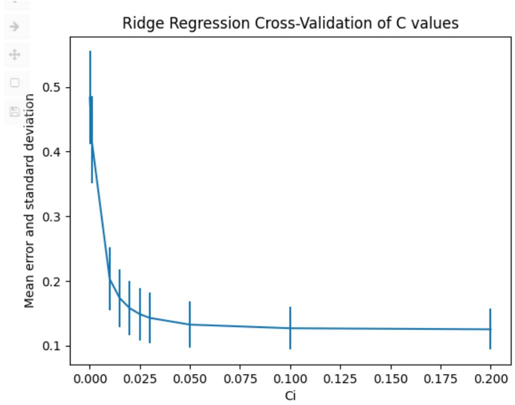
a) Below is a plot of the mean and the standard deviation of the prediction error, on the Lasso Regression model, against various values for C.
The range of values for C were chosen by repeatedly creating and plotting the model to find a reasonable range of values that would illustrate a clear trend.



b) Based off the graph and the data presented earlier I would recommend using a C value of 100. We know from earlier that a C of 100 only uses three features so it is still a simple model, and the mean error and standard deviation is adequately low.

c) Below is a plot of the mean and the standard deviation of the prediction error, on the Ridge Regression model, against various values for C.

The range of values for C were chosen by repeatedly creating and plotting the model to find a reasonable range of values that would illustrate a clear trend.



Based off the graph and the data presented earlier I would recommend using a C value of 0.1. It has a small standard deviation and appears to be a reasonable balance between representing the data and overfitting.

Appendix

```
# %%
%matplotlib widget
import numpy as np
import math
from matplotlib import cm
import pandas as pd
import scipy as sp
import seaborn as sn
import matplotlib.pyplot as plt
import matplotlib as matlib
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Ridge

from sklearn.model_selection import KFold
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import train_test_split

from sklearn.svm import LinearSVC
from sklearn.dummy import DummyRegressor
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Lasso
from sklearn.preprocessing import PolynomialFeatures
from mpl_toolkits.mplot3d import Axes3D

df = pd.read_csv('week3.csv')
df.columns = ["X1","X2","y"]
X1=df.iloc[:,0]
X2=df.iloc[:,1]
y=df.iloc[:,2]
X=np.column_stack((X1,X2))

# %% [markdown]
# a i)

# %%
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X1,X2,y)
ax.set_xlabel('Feature X1')
ax.set_ylabel('Feature X2')
ax.set_zlabel('Target y')
plt.title('Dataset Scatterplot')
plt.show()

# %% [markdown]
# a) ii)

# %%
Xpoly = PolynomialFeatures(degree = 5).fit_transform(X)

baseline = DummyRegressor(strategy="mean").fit(Xpoly, y)
print("J(θ_baseline) = %f\n"%mean_squared_error(y, baseline.predict(Xpoly)))

C = [1,10,100,1000,10000]
lassos = []
for Ci in C:
    model = Lasso(alpha=1/(2*Ci)).fit(Xpoly,y)
    lassos.append(model)
    print("\nC value: "+str(Ci))
    print("Lasso coef: "+str(model.coef_))
```

```
    print("Lasso intercept: "+str(model.intercept_))
    print("J(θ) = %f\n"%mean_squared_error(y, model.predict(Xpoly)))


# %%
### i) c)
Xtest = []
grid = np.linspace(-2,2)
for i in grid:
    for j in grid:
        Xtest.append([i,j])

Xtest = np.array(Xtest)
Xtest = PolynomialFeatures(5).fit_transform(Xtest)

from matplotlib import cm
Xpoly = PolynomialFeatures(5).fit_transform(X)

C_range = [1, 10, 100, 1000, 10000]
for Ci in C_range:
    model = Lasso(alpha=1/(2*Ci))
    model.fit(Xpoly, y)
    y_pred=model.predict(Xtest)

    fig = plt.figure(dpi=100)
    ax = fig.add_subplot(111, projection='3d')

    ax.scatter(X1, X2, y, c='g', label="Training data")
    surf = ax.plot_trisurf(Xtest[:,1], Xtest[:,2], y_pred, cmap=cm.YlGn, alpha=0.8, linewidth=0,
antialiased=True)
    ax.set_title('Lasso prediction surface for C = %.0f'%Ci)
    ax.set(xlabel='X1', ylabel='X2', zlabel='Target Y')
    ax.legend(loc='upper left')

    plt.xlabel("Feature X1"); plt.ylabel("Feature X2")

    plt.show()


# %%
# (i)(e)

baseline = DummyRegressor(strategy="mean").fit(Xpoly, y)
print("J(θ_baseline) = %f\n"%mean_squared_error(y, baseline.predict(Xpoly)))

ridges = []
C_ridge = [0.0001,0.001,0.01,0.1,1,10]
for Ci in C_ridge:
    model = Ridge(alpha=1/(2*Ci)).fit(Xpoly, y)
    theta = np.insert(model.coef_, 0, model.intercept_)
    ridges.append(model)

    print("\nC value: "+str(Ci))
    print("θ =", theta)
    print("J(θ) = %f\n"%mean_squared_error(y, model.predict(Xpoly)))

    fig = plt.figure(dpi=80)
    ax = fig.add_subplot(111, projection='3d')

    ax.scatter(X[:,0], X[:,1], y, c='g', label="Training data")
    surf = ax.plot_trisurf(Xtest[:,1], Xtest[:,2], model.predict(Xtest), cmap=cm.YlGn, alpha=0.8,
linewidth=0, antialiased=True)

    ax.set_title('Ridge prediction surface for C = ' + str(Ci))
```

```
    ax.set(xlabel='Feature X1', ylabel='Feature X2', zlabel='Target Y')
    ax.legend()

    plt.show()

print("exit")


# %%
### ii a)

kf = KFold(n_splits=5)
mean_error=[]
std_error=[]
Cs2 = [1,2,3,4,5,6,7,8,9,10,20,30,40,50,100]
for Ci in Cs2:
    model = Lasso(alpha=1/(2*Ci))
    print("\nC: "+str(Ci))
    kf = KFold(n_splits=5)
    temp = []
    for train, test in kf.split(X):
        model.fit(X[train],y[train])
        ypred = model.predict(X[test])
        print("Lasso coeff: "+str(model.coef_))

        from sklearn.metrics import mean_squared_error
        temp.append(mean_squared_error(y[test],ypred))
    mean_error.append(np.array(temp).mean())
    std_error.append(np.array(temp).std())

plt.figure(dpi=100)
plt.errorbar(Cs2,mean_error,yerr=std_error)
plt.xlabel('Ci'); plt.ylabel('Mean error and standard deviation')
plt.title('Lasso Regression Cross-Validation of C values')
plt.show()

### ii c)
mean_error=[]
std_error=[]
C_ridge2 = [0.0001,0.001,0.01,0.015,0.02,0.025,0.03,0.05,0.1,0.2]
for Ci in C_ridge2:
    model = Ridge(alpha=1/(2*Ci))

    from sklearn.model_selection import KFold
    kf = KFold(n_splits=5)
    temp = []
    print("\nC: "+str(Ci))

    for train, test in kf.split(X):
        model.fit(X[train],y[train])
        ypred = model.predict(X[test])
        print("Ridge coeff: "+str(model.coef_))
        from sklearn.metrics import mean_squared_error
        temp.append(mean_squared_error(y[test],ypred))
    mean_error.append(np.array(temp).mean())
    std_error.append(np.array(temp).std())

plt.figure(dpi=100)
plt.errorbar(C_ridge2,mean_error,yerr=std_error)
plt.xlabel('Ci'); plt.ylabel('Mean error and standard deviation')
plt.title('Ridge Regression Cross-Validation of C values')
plt.show()
```