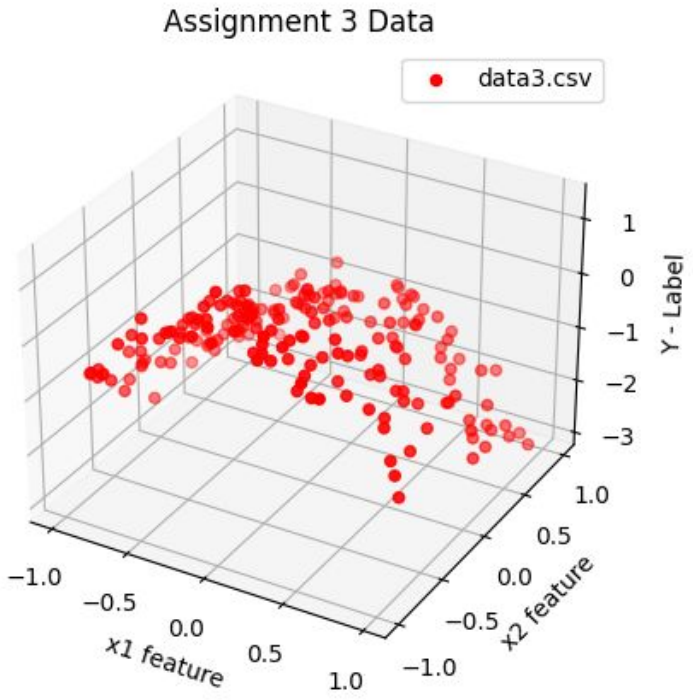
Machine Learning Assignment 3

Saul O'Driscoll (17333932)

DATA USED = # id:24--48--24

1 A)



Above I have plotted the data (in data3.csv) for this assignment as a 3d scatter plot. Everything is labeled. When rotating the image it looks like the training data lies on a curve.

DATA :

id:24--48--24
id:24--48--24
id:24--48--24
id:24--48--24
id:24--48--24
id:24--48--24
id:24--48--24

1 B) *After training lasso regression using multiple C values I got the following solutions. Listed are the C and Alpha values (Alpha = 1 / (2 * C)), as well as the coefficients IF THEY ARE NON-Zero and the intercept.*

C: 0.1 Alpha: 5.0 -> Accuracy: -0.01 (+/- 0.02)

Intercept = -0.6443946939418528

C: 1 Alpha: 0.5 -> Accuracy: -0.01 (+/- 0.02)

Intercept = -0.6443946939418528

C: 10 Alpha: 0.05 -> Accuracy: 0.89 (+/- 0.04)

(0, 2)	-0.8544316629410227
(0, 3)	-1.461151517993141

Intercept = -0.180587430535607

C: 50 Alpha: 0.01 -> Accuracy: 0.93 (+/- 0.04)

(0, 1)	-0.014492851381584768
(0, 2)	-0.9640243522839852
(0, 3)	-1.8923968835550347

Intercept = -0.04204465027655102

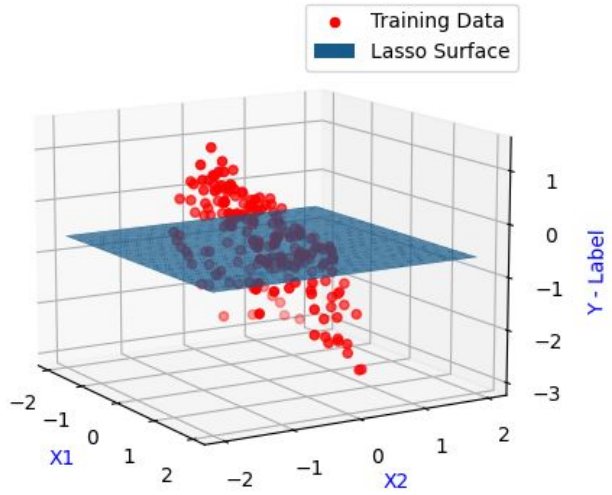
C: 1000 Alpha: 0.0005 -> Accuracy: 0.93 (+/- 0.04)

(0, 1)	-0.010055668034510596
(0, 2)	-1.0385778713957956
(0, 3)	-1.952952224454759
(0, 4)	0.008232974304626037
(0, 5)	-0.02125820285510815
(0, 6)	0.05055543332320097
(0, 8)	-0.21735791214343805
(0, 9)	0.07733618417265328
(0, 11)	-0.07931140834459838
(0, 12)	-0.11639781748561143
(0, 13)	-0.035574338207832415
(0, 16)	0.029334172420879816

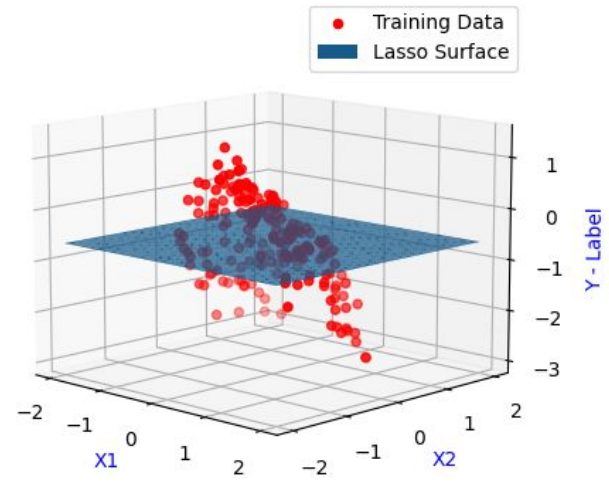
Intercept = -0.001904758435416598

As C increases above 1, alpha shrinks and we begin to see the hyperplane fit itself to the data. We can see this because coefficients are changing and being updated with each meaningful iteration. By meaningful we mean that C is large enough to influence the fitting process. We see no large accuracy increase from C=50 to C=1000. Coefficients are being changed but this new hyperplane does not lead to greater accuracy of the model. We see change in the plane but no better accuracy and this may be a sign of overfitting. **FOLLOWING PAGE IS START OF 1 C)**

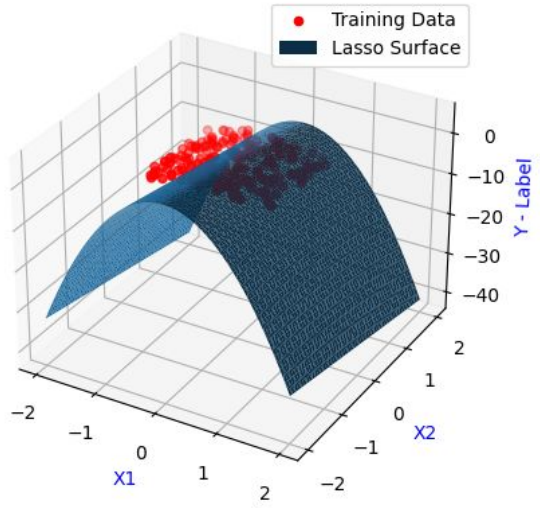
Lasso Prediction Plane with c=0.1 / alpha=5.0



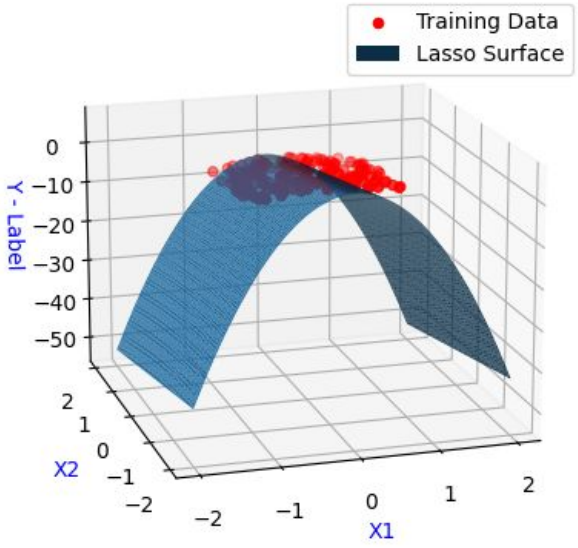
Lasso Prediction Plane with c=1 / alpha=0.5



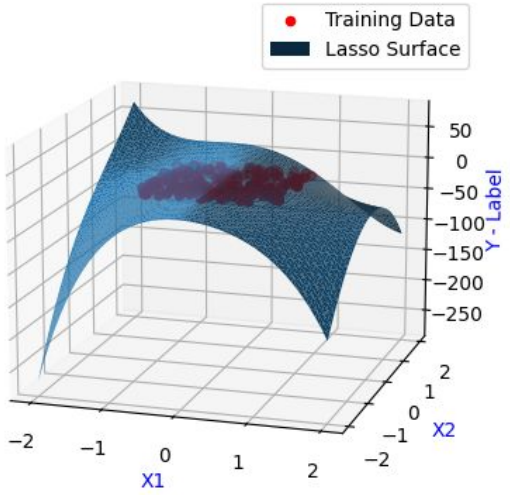
Lasso Prediction Plane with c=10 / alpha=0.05



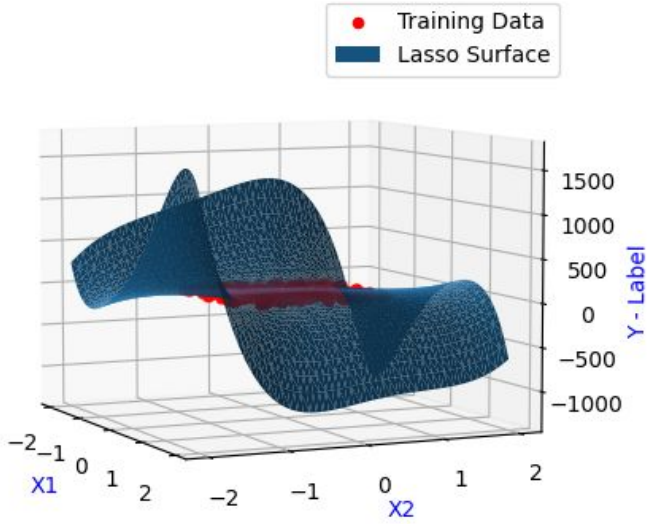
Lasso Prediction Plane with c=50 / alpha=0.01



Lasso Prediction Plane with c=1000 / alpha=0.0005



Lasso Prediction Plane with c=10000 / alpha=5e-05



1 C) Above I have displayed 6 plots. Each plot corresponds to a fit of the lasso regression model to our data. The only thing varied each time is the Alpha parameter which we derive from a C we decide beforehand. In 1 B) we looked at the accuracy of various models and how they change with regard to parameter C/Alpha. Here we can see the different influences of C/Alpha on the surface that represents how the model is fit to the training data. For low C values like 0.1 and 1 we don't see a change in the plane as no coefficients are changed. As we increase C we can see the surface morph to fit the data. At very high values we start seeing the surface crimp or change more dramatically.

1D) The surface in 1C) for high C values like 1000 and 10000 is changing its shape drastically to adjust for the outliers in the data, we call this overfitting. We see the model tolerate less and less training error in an effort to become perfectly fit to the training data. This has the downside that the model becomes a poor classifier of predicting labels for future test data it is shown. As we know from our numbers in 1B) our model is already quite accurate at around **C = 50** with an accuracy of ~93%. In an attempt to fit itself even better it becomes overfit.

I find the bias-variance dilemma or trade off explains the under- and overfitting phenomenon very well. The trade off is between high accuracy on training data meaning the model is sensitive to small data fluctuations in the training set (overfitting = high variance) and on the other hand high bias which can cause the model to miss the relevant relations between features and target outputs (underfitting)

We use C to manage this trade off by implicitly giving the model a margin of error it should or shouldn’t tolerate during training. High C = low tolerance for errors during training. I think.

1 E)

C: 0.001 Alpha: 500.0 -> Accuracy: 0.18 (+/- 0.03)

Coefficients = [0. -0.00140272 -0.11371078 -0.06999827 -0.00982546 0.00689955
-0.0018227 -0.03099416 -0.00522501 -0.06475686 -0.05966411 -0.00822672
-0.02042607 -0.0047734 0.00613468 -0.002128 -0.01762769 -0.00485426
-0.01724185 -0.00301586 -0.0451385]

Intercept = -0.6136336552870518

C: 0.01 Alpha: 50.0 -> Accuracy: 0.67 (+/- 0.07)

Coefficients = [0. 0.00336492 -0.44921601 -0.44115101 -0.04805578 0.03126777
0.00467821 -0.10384477 -0.02312044 -0.21702328 -0.36877167 -0.04559017
-0.12992453 -0.02878088 0.02810136 -0.00110714 -0.05300889 -0.02018816
-0.0477533 -0.00999459 -0.13543256]

Intercept = -0.4377132865000458

C: 1 Alpha: 0.5 -> Accuracy: 0.92 (+/- 0.04)

Coefficients = [0. -0.04570195 -0.99689589 -1.55363842 0.0964183 -0.02925952
0.16641843 -0.07516123 -0.20969411 -0.03793368 -0.37061981 -0.13786599
-0.23866819 -0.121913 0.05916902 -0.07607989 0.10843396 -0.05043343
-0.00263041 0.05136541 0.08720317]

Intercept = -0.05455783349942067

C: 10 Alpha: 0.05 -> Accuracy: 0.92 (+/- 0.04)

Coefficients = [0. -0.07197795 -1.09424699 -1.95707518 0.14939773 -0.12263685
0.41258522 0.04371533 -0.50773129 0.21929272 0.01278763 -0.20182303
-0.10498678 -0.16766041 0.12391716 -0.30671203 0.1578981 0.02204192
-0.26732328 0.35012747 -0.05374599]

Intercept = 0.008180373672225794

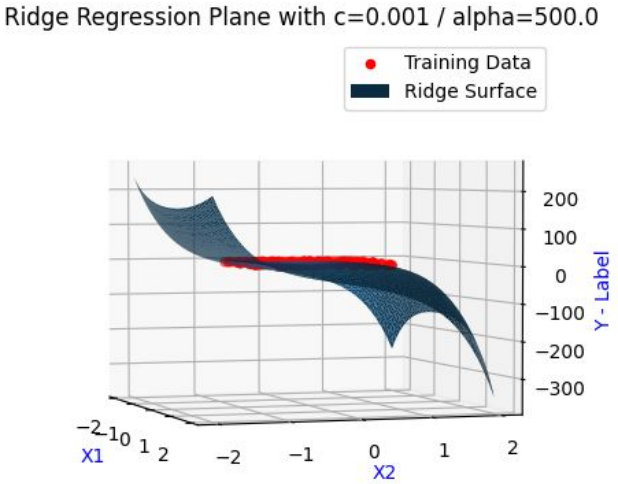
C: 100 Alpha: 0.005 -> Accuracy: 0.92 (+/- 0.04)

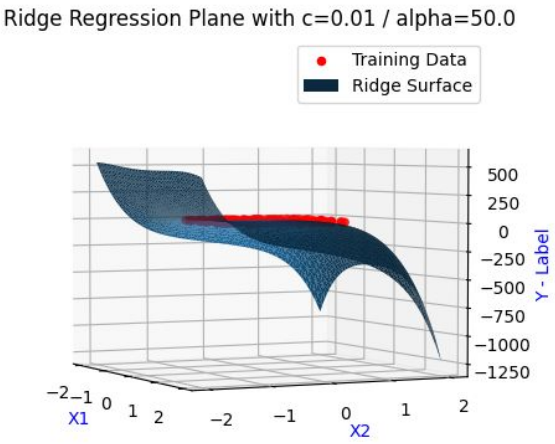
Coefficients = [0. -0.07288214 -1.14834152 -2.04061386 0.16453313 -0.15132353
0.50076142 0.15501198 -0.66426859 0.3817517 0.09192005 -0.21996154
-0.06098174 -0.18481925 0.14636338 -0.40259276 0.12644877 0.08466922
-0.41655574 0.49831267 -0.16207958]

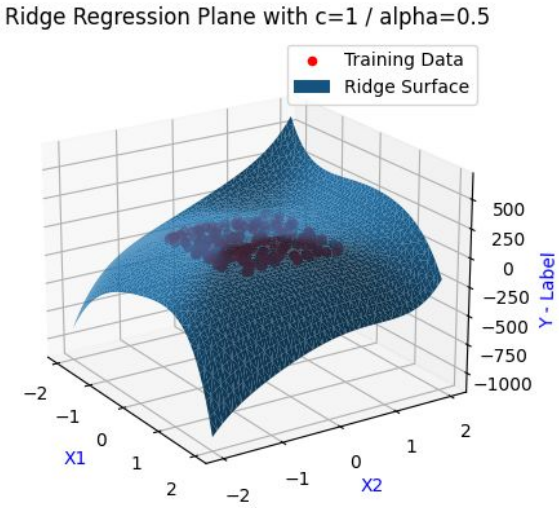
Intercept = 0.021255791290906934

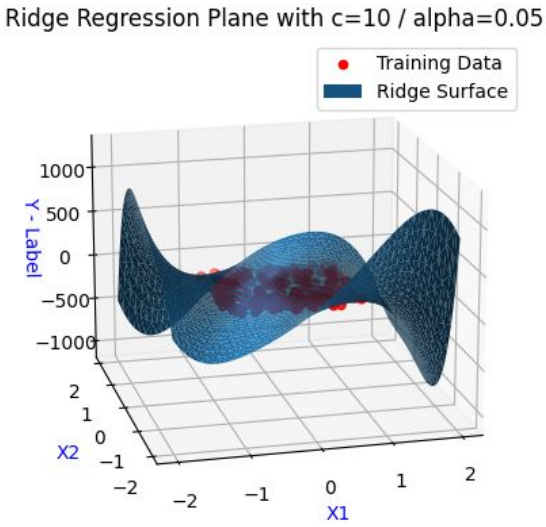
As C increases, alpha shrinks and we begin to see the hyperplane fit itself to the data. We can see this because coefficients are changing and being updated with each meaningful iteration. We can see that between C = 1 and C = 100 we don't see any increase in accuracy. We have likely reached the limits of our model with tweaking just the C parameter. We see change in the surface but no improvement and this may be a sign of overfitting.

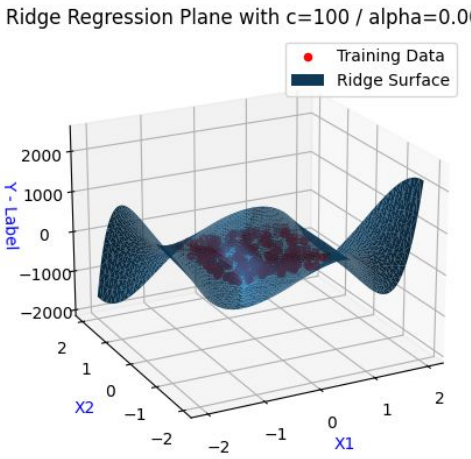
Comparing this to Lasso regression we see that ridge regression does not seem to benefit very much from larger Cs above 10 or so while lasso regression does. Also noting that ridge regression seems to always have coefficients set while lasso does not. This surely has to do with the way the model evolves and updates itself. Lasso starts from a flat plane and “lassoing” in other values into the plane. Ridge starting with an outgoing surface and morphing as needed.











2 A) As per the Assignment I have used C=1 for this particular exercise. We learned that C = 1 does not fit to the data well in part 1 B) and will leave me with bogus accuracy. I would rather use a value like 5 or 10. Anyway.

Folds:2 - Accuracy: -0.01 (+/- 0.00)
Folds:5 - Accuracy: -0.01 (+/- 0.03)
Folds:10 - Accuracy: -0.04 (+/- 0.10)
Folds:25 - Accuracy: -0.09 (+/- 0.32)
Folds:50 - Accuracy: -1.44 (+/- 14.34)
Folds:60 - Accuracy: -4.66 (+/- 48.24)

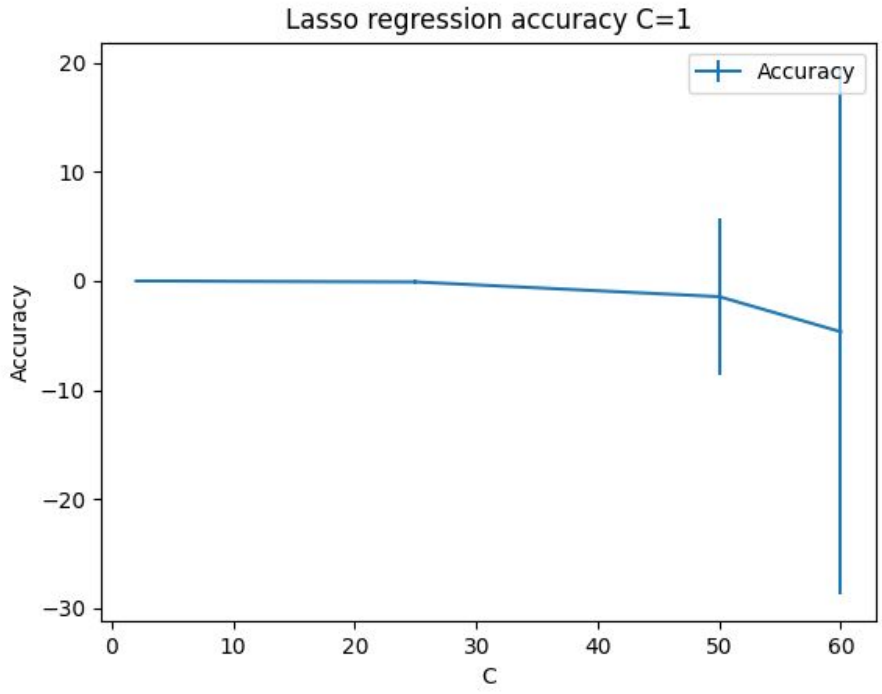
Here is what the code outputs. We can see the amount of folds and the estimates of accuracy along with the standard deviation. Because C = 1 does not work well we get bogus accuracy. However we see that the standard deviation fans out with greater values so we can assume that lower values will lead to a lower spread of values which is good when predicting values. At large fold values the numbers stop making sense because the R^2 score is not well-defined with less than two samples so I did not go up to 100

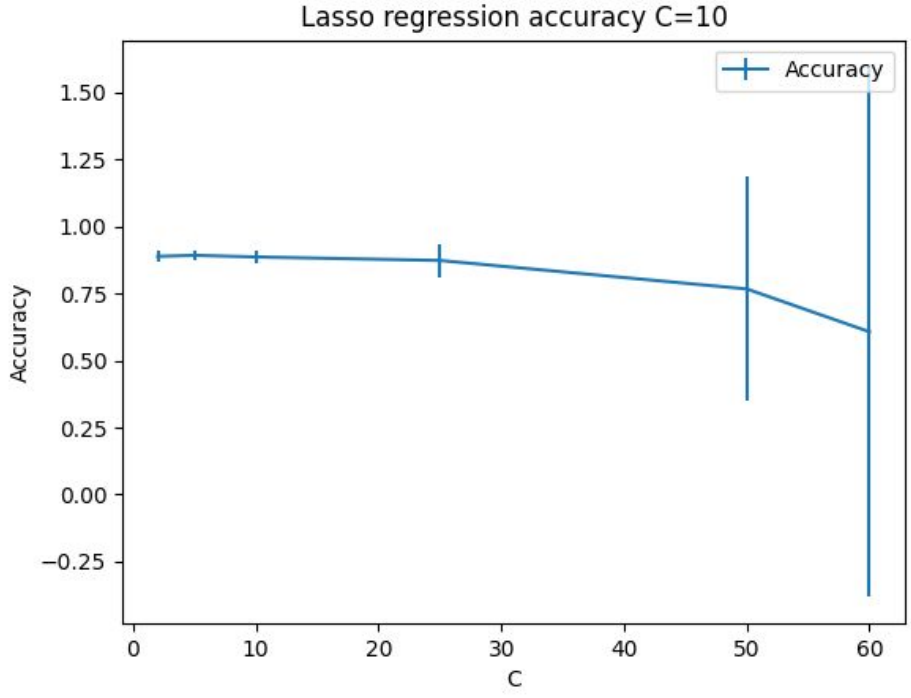
Here are better values with a C=10:

Folds:2 - Accuracy: 0.89 (+/- 0.05)
Folds:5 - Accuracy: 0.89 (+/- 0.03)
Folds:10 - Accuracy: 0.89 (+/- 0.05)
Folds:25 - Accuracy: 0.87 (+/- 0.13)
Folds:50 - Accuracy: 0.77 (+/- 0.84)
Folds:60 - Accuracy: 0.61 (+/- 1.98)

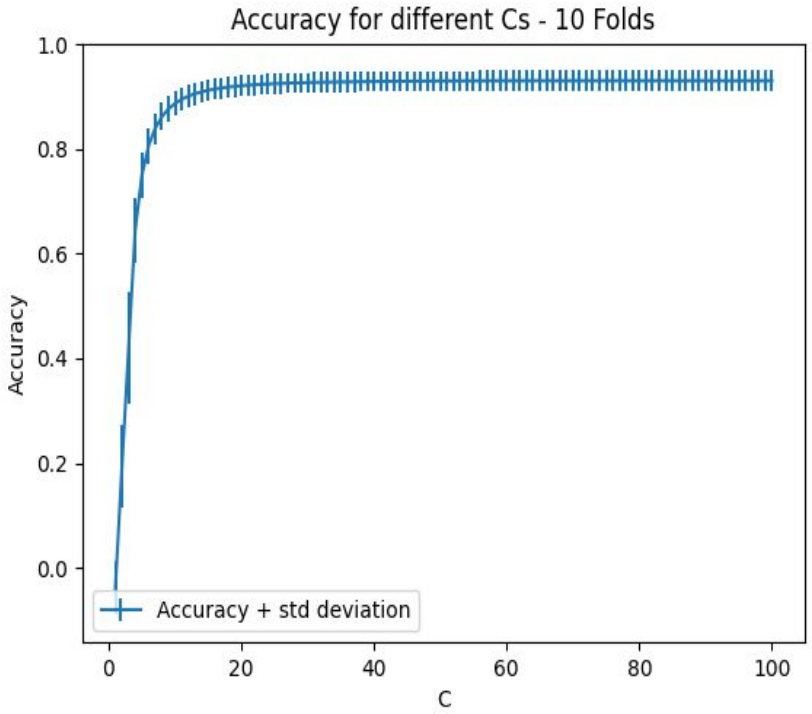
Accuracy decreases between 10-60 folds and standard deviation increases.
Too many folds leads to less accuracy because usually large means less (pessimistic) bias. Too large also means that only a low number of sample combinations is possible, limiting the number of iterations that are different.

Below are the graphs for both fold tests with both Cs.


Lasso regression accuracy C=1
Accuracy
20
10
0
−10
−20
−30
Accuracy
0
10
20
30
40
50
60
C

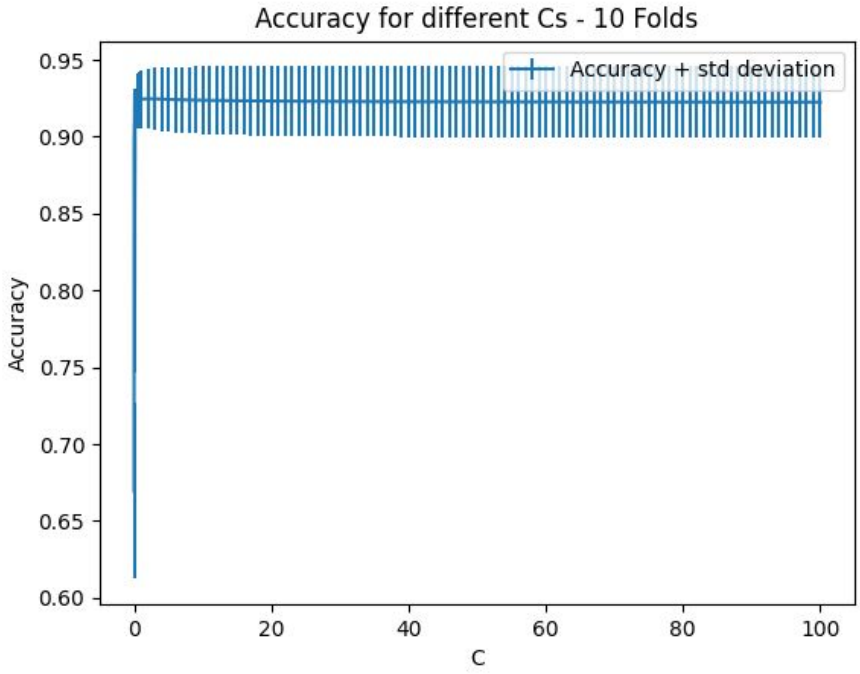

Lasso regression accuracy C=10
Accuracy
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
Accuracy
0
10
20
30
40
50
60
C

2 B + C)
C:1 - Accuracy: -0.04 (+/- 0.10)
C:2 - Accuracy: 0.19 (+/- 0.16)
C:3 - Accuracy: 0.42 (+/- 0.21)
C:4 - Accuracy: 0.65 (+/- 0.12)
C:5 - Accuracy: 0.75 (+/- 0.09)
C:6 - Accuracy: 0.81 (+/- 0.07)
C:7 - Accuracy: 0.84 (+/- 0.06)
C:8 - Accuracy: 0.86 (+/- 0.05)
C:9 - Accuracy: 0.88 (+/- 0.05)
C:10 - Accuracy: 0.89 (+/- 0.05)
C:11 - Accuracy: 0.89 (+/- 0.04)
C:12 - Accuracy: 0.90 (+/- 0.04)
C:13 - Accuracy: 0.91 (+/- 0.04)
C:14 - Accuracy: 0.91 (+/- 0.04)
C:15 - Accuracy: 0.91 (+/- 0.04)
C:16 - Accuracy: 0.91 (+/- 0.04)
C:17 - Accuracy: 0.92 (+/- 0.04)
C:18 - Accuracy: 0.92 (+/- 0.04)
C:19 - Accuracy: 0.92 (+/- 0.04)
C:20 - Accuracy: 0.92 (+/- 0.04)
C:21 - Accuracy: 0.92 (+/- 0.04)
C:22 - Accuracy: 0.92 (+/- 0.04)
C:23 - Accuracy: 0.92 (+/- 0.04)
C:24 - Accuracy: 0.92 (+/- 0.04)
C:25 - Accuracy: 0.92 (+/- 0.04)
C:26 - Accuracy: 0.93 (+/- 0.04)
* * * * * * * * * * * *
C:98 - Accuracy: 0.93 (+/- 0.04)
C:99 - Accuracy: 0.93 (+/- 0.04)
C:100 - Accuracy: 0.93 (+/- 0.04)



I chose this range of Cs because I also tested greater Cs than 100 with the same results up to 1000. Cs between 0-1 have abysmal accuracy so I did not test them either. 2 C) Above we have the iteration through Cs with a cross fold validation fold amount of 10. Between 26 and 100 Cs we don't see a change in accuracy or standard deviation. Based on this I would probably choose a C of 10. I will take the 2 point hit on accuracy for the peace of mind that the model is not possibly overfit. Good models are 95%+ anyway so I don't mind unless that's the threshold I hit. The fact that a model with C = 26 and C = 100 yields the same accuracy leads me to believe that C = 26 is too high.

2 D) C:0.01 - Accuracy: 0.67 (+/- 0.11)
C:0.02 - Accuracy: 0.79 (+/- 0.09)
C:0.03 - Accuracy: 0.84 (+/- 0.08)
C:0.04 - Accuracy: 0.87 (+/- 0.07)
C:0.05 - Accuracy: 0.88 (+/- 0.06)
C:0.06 - Accuracy: 0.89 (+/- 0.06)
C:0.07 - Accuracy: 0.90 (+/- 0.05)
C:0.08 - Accuracy: 0.90 (+/- 0.05)
C:0.09 - Accuracy: 0.91 (+/- 0.05)
C:0.1 - Accuracy: 0.91 (+/- 0.05)
* * * * * * * * * * * * *
C:0.5 - Accuracy: 0.92 (+/- 0.04)
* * * * * * * * * * * * *
C:0.7 - Accuracy: 0.92 (+/- 0.04)
* * * * * * * * * * * * *
C:1 - Accuracy: 0.92 (+/- 0.04)
C:2 - Accuracy: 0.92 (+/- 0.04)
C:3 - Accuracy: 0.92 (+/- 0.04)
C:4 - Accuracy: 0.92 (+/- 0.04)
C:5 - Accuracy: 0.92 (+/- 0.04)
C:6 - Accuracy: 0.92 (+/- 0.04)
C:7 - Accuracy: 0.92 (+/- 0.04)
C:8 - Accuracy: 0.92 (+/- 0.04)
C:9 - Accuracy: 0.92 (+/- 0.04)
C:10 - Accuracy: 0.92 (+/- 0.04)
C:11 - Accuracy: 0.92 (+/- 0.04)
C:12 - Accuracy: 0.92 (+/- 0.04)
C:13 - Accuracy: 0.92 (+/- 0.04)
C:14 - Accuracy: 0.92 (+/- 0.04)
C:15 - Accuracy: 0.92 (+/- 0.04)
* * * * * * * * * * * * *
C:98 - Accuracy: 0.92 (+/- 0.05)
C:99 - Accuracy: 0.92 (+/- 0.05)
C:100 - Accuracy: 0.92 (+/- 0.05)



I chose this range of Cs because I also tested greater and lower Cs. Very low Cs have bad accuracy so I did not test them. Above we have the iteration through Cs with a cross fold validation fold amount of 10. Between 0.1 and 100 Cs we don't see a change in accuracy or standard deviation. Based on this I would probably choose a C of 1 because the simplest solution is often the correct one. Occam's razor :). Thanks.

I used a good few different python files to separate my work so there is quite a bit of duplicate code.

```
# id:24--48--24
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

df = pd.read_csv("data.csv",header=None, comment="#", names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]

poly = PolynomialFeatures(5)
polydata = poly.fit_transform(X_all)
print(polydata)

fig = plt.figure()
ax = fig.gca(projection='3d')
scatter = ax.scatter(X_all["X1"], X_all["X2"], X_labels.to_numpy(), c='r', marker='o',
label="data3.csv")
ax.legend()
ax.set_title("Assignment 3 Data")
ax.set_xlabel('x1 feature', c="b")
ax.set_ylabel('x2 feature', c="b")
ax.set_zlabel('Y - Label', c="b")
plt.show()
```

```
# id:24--48--24
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix
from sklearn import linear_model
from matplotlib import cm
from sklearn.model_selection import cross_val_score, cross_val_predict
```

```
# np.set_printoptions(formatter={'float': lambda x: "{0:0.6f}".format(x)})
#Importing Xtrain + Labels
df = pd.read_csv("data.csv",header=None, comment="#", names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]

poly = PolynomialFeatures(5)
polydata = poly.fit_transform(X_all)

#Converting C to alphas for easier use
c_array = [0.001, 0.01, 1, 2, 5, 10, 100, 1000]
c_array = [0.1, 1, 10, 50, 1000]
def cToAlpha(c_array):
        alpha_array = []
        for c in c_array:
        alpha_array.append(1/(2*c))

        return alpha_array
alpha_array = cToAlpha(c_array)

def lassoing(a, c):
        clf1 = linear_model.Lasso(alpha=a)
        cross_val_score(clf1, polydata, X_labels, cv=8)
        clf1.fit(polydata, X_labels)
        print(clf1.sparse_coef_)
        coefs = clf1.coef_
        intercept = clf1.intercept_
        # print("Coefficients = ", coefs)
        print("Intercept = ", intercept)
        # Xtest = makeXTest(5)
        # poly_Xtest = poly.fit_transform(Xtest)
        # predictions = clf.predict(poly_Xtest)
        # fig = plt.figure()
        # Xtest = makeXTest(2)
        # ax = fig.gca(projection='3d')
        # ax.set_title("Lasso Prediction Plane with c={0} / alpha={1}".format(c, a))
        # ax.set_xlabel('X')
        # ax.set_ylabel('Y')
        # ax.set_zlabel('Z')
        # ax.scatter(X_all["X1"], X_all["X2"], X_labels, c='r')
```

```
        # surf = ax.plot_trisurf(Xtest[:,0],Xtest[:,1], predictions)
        # plt.show()


def crossValScorePrinting(a,c,cvs=8):
        clf = linear_model.Lasso(alpha=a)
        scores = cross_val_score(clf, polydata, X_labels, cv=cvs)
        # lasso_mean_array = []
        # lasso_std_array = []
        print("C: " + str(c) + " Alpha: " + str(a) +  " -> Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))
        # lasso_mean_array.append(scores.mean())
        # lasso_std_array.append(scores.std())
        # print(lasso_mean_array, lasso_std_array)
        # return scores

for a,c in zip(alpha_array,c_array):
        print("----------------------------------------")
        crossValScorePrinting(a, c)
        lassoing(a, c)
```

```
# id:24--48--24
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix
from sklearn import linear_model
from matplotlib import cm
from sklearn.model_selection import cross_val_score, cross_val_predict

# np.set_printoptions(formatter={'float': lambda x: "{0:0.6f}".format(x)})
#Importing Xtrain + Labels
df = pd.read_csv("data.csv",header=None, comment="#", names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]
```

```
poly = PolynomialFeatures(5)
polydata = poly.fit_transform(X_all)

#Converting C to alphas for easier use
c_array = [0.001, 0.01, 1, 10, 100]
c_array = [0.001, 0.01, 1, 10, 50, 100]
def cToAlpha(c_array):
        alpha_array = []
        for c in c_array:
        alpha_array.append(1/(2*c))

        return alpha_array
alpha_array = cToAlpha(c_array)

def ridging(a, c):
        clf1 = linear_model.Ridge(alpha=a)
        cross_val_score(clf1, polydata, X_labels, cv=8)
        clf1.fit(polydata, X_labels)
        coefs = clf1.coef_
        intercept = clf1.intercept_
        print("Coefficients = ", coefs)
        print("Intercept = ", intercept)
        # Xtest = makeXTest(5)
        # poly_Xtest = poly.fit_transform(Xtest)
        # predictions = clf.predict(poly_Xtest)
        # fig = plt.figure()
        # Xtest = makeXTest(2)
        # ax = fig.gca(projection='3d')
        # ax.set_title("Lasso Prediction Plane with c={0} / alpha={1}".format(c, a))
        # ax.set_xlabel('X')
        # ax.set_ylabel('Y')
        # ax.set_zlabel('Z')
        # ax.scatter(X_all["X1"], X_all["X2"], X_labels, c='r')
        # surf = ax.plot_trisurf(Xtest[:,0],Xtest[:,1], predictions)
        # plt.show()


def crossValScorePrinting(a,c,cvs=8):
        clf = linear_model.Ridge(alpha=a)
        scores = cross_val_score(clf, polydata, X_labels, cv=cvs)
```

```
        # lasso_mean_array = []
        # lasso_std_array = []
        print("C: " + str(c) + " Alpha: " + str(a) +  " -> Accuracy: %0.2f (+/- %0.2f)" %
(scores.mean(), scores.std() * 2))
        # lasso_mean_array.append(scores.mean())
        # lasso_std_array.append(scores.std())
        # print(lasso_mean_array, lasso_std_array)
        # return scores

for a,c in zip(alpha_array,c_array):
        print("----------------------------------------")
        crossValScorePrinting(a, c)
        ridging(a, c)
```

```
# id:24--48--24
import numpy as np
import pandas as pd
import matplotlib.patches as mpatches
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix
from sklearn import linear_model
from matplotlib import cm
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import KFold
```

```
#Importing Xtrain + Labels
df = pd.read_csv("data.csv",header=None, comment="#", names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]

poly = PolynomialFeatures(5)
polydata = poly.fit_transform(X_all)

def makeXTest(dimension=5):
        Xtest = []
        grid = np.linspace(-dimension,dimension)
        for i in grid:
        for j in grid:
        Xtest.append([i,j])
        Xtest = np.array(Xtest)
        return Xtest


#Converting C to alphas for easier use
c_array = [0.001, 0.1, 1, 10, 50, 100, 1000, 10000]
c_array = [0.001, 0.01, 1, 10, 100]
def cToAlpha(c_array):
        alpha_array = []
        for c in c_array:
        alpha_array.append(1/(2*c))

        return alpha_array

alpha_array = cToAlpha(c_array)

def lassoing():
        for a,c in zip(alpha_array,c_array):
        clf = linear_model.Lasso(alpha=a)
        clf.fit(polydata, X_labels)
        Xtest = makeXTest(5)
        poly_Xtest = poly.fit_transform(Xtest)
        predictions = clf.predict(poly_Xtest)
        fig = plt.figure()
        Xtest = makeXTest(2)
        ax = fig.gca(projection='3d')
```

```
        ax.set_title("Lasso Prediction Plane with c={0} / alpha={1}".format(c, a))
        ax.set_xlabel('X1', c="b")
        ax.set_ylabel('X2', c="b")
        ax.set_zlabel('Y - Label', c="b")
        scatter = ax.scatter(X_all["X1"], X_all["X2"], X_labels, c='r', label="data3.csv")
        surf = ax.plot_trisurf(Xtest[:,0],Xtest[:,1], predictions)
        surf._facecolors2d=surf._facecolors3d
        surf._edgecolors2d=surf._edgecolors3d
        plt.legend([scatter, surf], ["Training Data", "Lasso Surface"])
        plt.show()

lassoing()

def rideRegression():
        for a,c in zip(alpha_array,c_array):
        clf = linear_model.Ridge(alpha=a)
        clf.fit(polydata, X_labels)
        Xtest = makeXTest(5)
        poly_Xtest = poly.fit_transform(Xtest)
        predictions = clf.predict(poly_Xtest)
        Xtest = makeXTest(2)
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        ax.set_title("Ridge Regression Plane with c={0} / alpha={1}".format(c, a))
        ax.set_xlabel('X1', c="b")
        ax.set_ylabel('X2', c="b")
        ax.set_zlabel('Y - Label', c="b")
        scatter = ax.scatter(X_all["X1"], X_all["X2"], X_labels, c='r', label="data3.csv")
        surf = ax.plot_trisurf(Xtest[:,0],Xtest[:,1], predictions)
        surf._facecolors2d=surf._facecolors3d
        surf._edgecolors2d=surf._edgecolors3d
        plt.legend([scatter, surf], ["Training Data", "Ridge Surface"])
        plt.show()

rideRegression()
```

```
# id:24--48--24
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.model_selection import cross_val_score, cross_val_predict

df = pd.read_csv("data.csv",header=None, comment="#", names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]

#Generating Xtest
def makeXTest(dimension=5):
    Xtest = []
    grid = np.linspace(-dimension,dimension)
    for i in grid:
        for j in grid:
            Xtest.append([i,j])
    Xtest = np.array(Xtest)
    return Xtest

poly = PolynomialFeatures(5)
polydata = poly.fit_transform(X_all)
print(polydata)

c_array = [0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.5,0.7]
for i in range(1,101):
    c_array.append(i)

def cToAlpha(c_array):
    alpha_array = []
    for c in c_array:
        alpha_array.append(1/(2*c))
    return alpha_array
alpha_array = cToAlpha(c_array)
```

```
def crossValScore(a,c,cvs=10):
    clf = linear_model.Lasso(alpha=a)
    scores = cross_val_score(clf, polydata, X_labels, cv=cvs)
    return scores

lasso_mean_array = []
lasso_std_array = []
for a,c in zip(alpha_array,c_array):
    scores = crossValScore(a,c)
    print("C:" + str(c) + " - Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() *
2))
    lasso_mean_array.append(scores.mean())
    lasso_std_array.append(scores.std())
    # print(lasso_mean_array, lasso_std_array)

#Assignment 3 part 2
# crossFoldArray = [2, 5, 10, 25, 50, 60]
# cross_mean_array = []
# cross_std_array = []
# for folds in crossFoldArray:
#     clf = linear_model.Lasso(alpha=1/(2*10))
#     scores = cross_val_score(clf, polydata, X_labels, cv=folds)
#     print("Folds:" + str(folds) + " - Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std() * 2))
#     cross_mean_array.append(scores.mean())
#     cross_std_array.append(scores.std())
#     # print(cross_mean_array, cross_std_array)

plt.errorbar(c_array, lasso_mean_array, yerr=lasso_std_array, label="Accuracy + std
deviation")
# plt.errorbar(crossFoldArray, cross_mean_array, yerr=cross_std_array,
label="Accuracy")
plt.xlabel("C")
plt.ylabel("Accuracy")
plt.title("Accuracy for different Cs - 10 Folds")
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------
# id:24--48--24
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.model_selection import cross_val_score, cross_val_predict

df = pd.read_csv("data.csv",header=None, comment="#", names=["X1","X2","X3"])
X_all = df.drop(columns=["X3"])
X_labels = df["X3"]

#Generating Xtest
def makeXTest(dimension=5):
        Xtest = []
        grid = np.linspace(-dimension,dimension)
        for i in grid:
        for j in grid:
        Xtest.append([i,j])
        Xtest = np.array(Xtest)
        return Xtest

poly = PolynomialFeatures(5)
polydata = poly.fit_transform(X_all)
print(polydata)

c_array = [0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.5,0.7]
for i in range(1,101):
        c_array.append(i)

print(c_array)
def cToAlpha(c_array):
        alpha_array = []
        for c in c_array:
        alpha_array.append(1/(2*c))
        return alpha_array
alpha_array = cToAlpha(c_array)
```

```
def crossValScore(a,c,cvs=10):
	clf = linear_model.Ridge(alpha=a)
	scores = cross_val_score(clf, polydata, X_labels, cv=cvs)
	return scores

ridge_mean_array = []
ridge_std_array = []
for a,c in zip(alpha_array,c_array):
	scores = crossValScore(a,c)
	print("C:" + str(c) + " - Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std() * 2))
	ridge_mean_array.append(scores.mean())
	ridge_std_array.append(scores.std())
	# print(lasso_mean_array, lasso_std_array)

#Assignment 3 part 2
# crossFoldArray = [2, 5, 10, 25, 50, 60]
# cross_mean_array = []
# cross_std_array = []
# for folds in crossFoldArray:
#	clf = linear_model.Lasso(alpha=1/(2*10))
#	scores = cross_val_score(clf, polydata, X_labels, cv=folds)
#	print("Folds:" + str(folds) + " - Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std() * 2))
#	cross_mean_array.append(scores.mean())
#	cross_std_array.append(scores.std())
#	# print(cross_mean_array, cross_std_array)

plt.errorbar(c_array, ridge_mean_array, yerr=ridge_std_array, label="Accuracy + std
deviation")
# plt.errorbar(crossFoldArray, cross_mean_array, yerr=cross_std_array,
label="Accuracy")
plt.xlabel("C")
plt.ylabel("Accuracy")
plt.title("Accuracy for different Cs - 10 Folds")
plt.legend()
plt.show()
```