Assignment Week 3

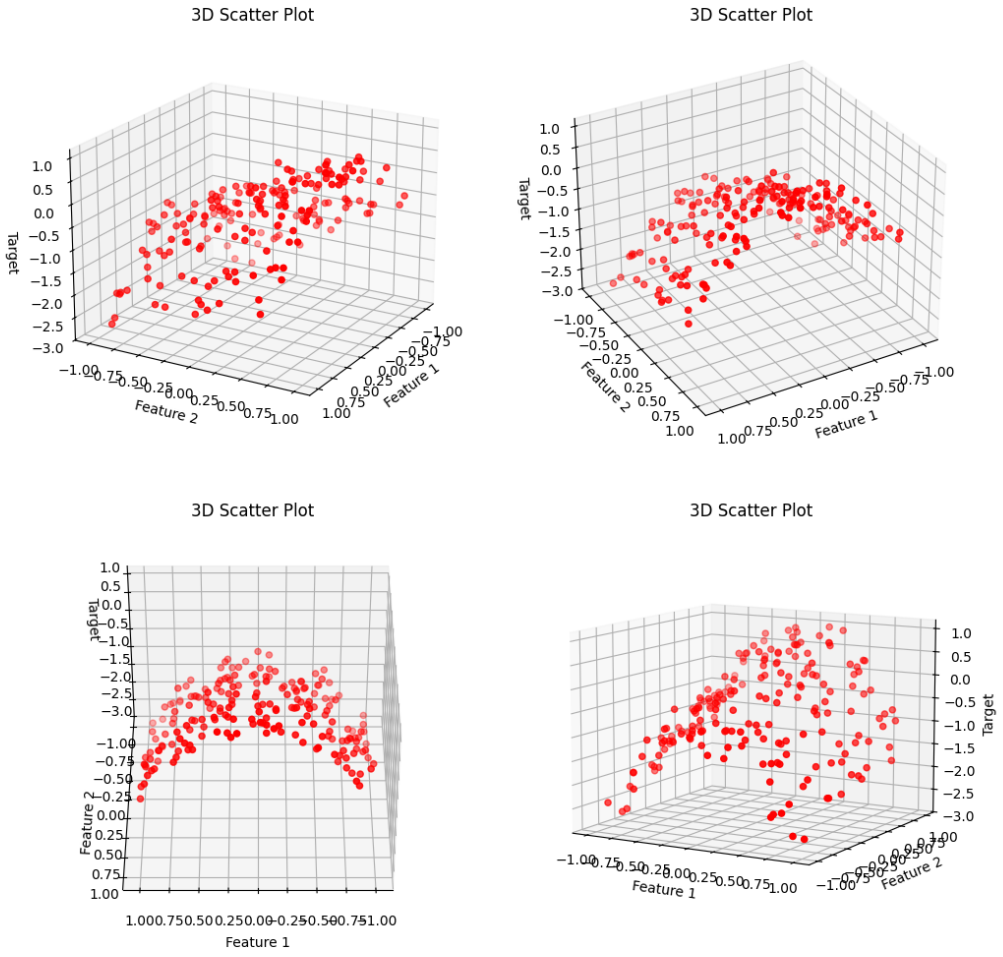
Student Name: Patrick Farmer Student Number: 20331828

Date: 2024-07-10 Dataset: 20–40-20

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

I(A)

The data was read in from the csv into X which held feature 1 and 2 and to Y which held the target. The data was then plotted onto a 3d scatter plot. We can see from the four different angles of the plot below that the data lies on a curve.



I(B)

Features up to the power of 5 were added to the dataset by using the PolynomialFeatures function in sklearn. Using this new data, a lasso regression model was trained with a number of different c values. The C Values chosen were (1, 10, 1000 and 10000). The value 1 was started with because it had all coefficients of 0. The value 10 was then chosen second as it demonstrated a significant improvement in the accuracy of the model with a mean square error of 0.7, there was only 2 coefficients that were non 0 in this case, both of which coefficients from feature 1. The value of C then chosen next was 1000, this is because this is when some more coefficients started to become non 0, all of the coefficients are still

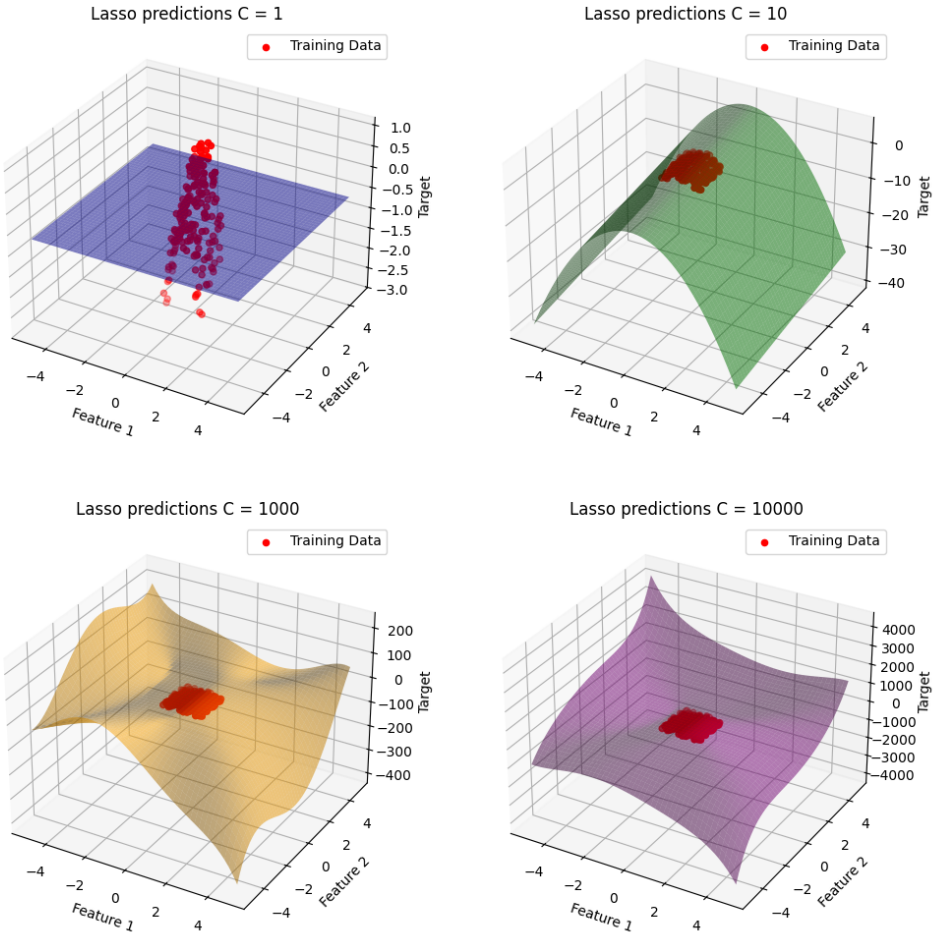
from feature 1 but it is every power of feature 1 (excluding power of 1) that has an impact on the target, there is a substantial difference again in the mean square error, it is half of the MSE for c=10. The final value of C chosen was 10000, this is to demonstrate the effect of overfitting. There is more of an emphasis on the higher power of feature 1 and also of feature 2 in this case which means that it is likely to not generalise as well to new data. The mean square error is less on the training data (even though marginally) but it is likely to be higher on new data.
The results printed to terminal is as follows:

```
Lasso regression:
C: 1
Coefficients: [ 0. -0.  0. -0.  0.  0. -0.  0. -0.  0. -0. -0. -0.  0. -0. -0.  0. -0.
  0. -0.  0.]
Intercept: -0.6620942711094104
Mean square error: 0.7052321665957307


C: 10
Coefficients: [ 0.         -0.          0.84066564 -1.38608729  0.          0.
 -0.          0.         -0.          0.         -0.          0.
 -0.          0.         -0.         -0.          0.         -0.
  0.         -0.          0.        ]
Intercept: -0.20024920720097728
Mean square error: 0.07923880695629874


C: 1000
Coefficients: [ 0.          0.0360211   0.98093878 -1.91335315  0.02995874 -0.00369941
  0.          0.          0.         -0.          0.         -0.0711885
  0.          0.16497518  0.         -0.08311     0.01187887  0.12169111
  0.03993382 -0.07826324  0.        ]
Intercept: -0.018001936262590612
Mean square error: 0.042884824527337134


C: 10000
Coefficients: [ 0.         -0.03856134  1.13393801 -2.04536396  0.06532606 -0.08089148
  0.31691552 -0.29712296  0.1638204  -0.48722808  0.1330193  -0.21899704
  0.07159164  0.2654704   0.0782562  -0.40746191  0.20309085  0.28945988
  0.32091182 -0.37310009  0.33203645]
Intercept: 0.005803973720044486
Mean square error: 0.041553454785013544
```

I(C)

The previously trained models were then used to predict target values and plotted onto a 3d scatter plot with bounds exceeding the range of the training data. This was done using the plot_surface function in the matplotlib library. As mentioned with the coefficients we can see that the lower values of C have a more generalised model and the higher values of C have a more overfitted model. We can also see in the graph how with the final 2 models the model predictions get very erratic towards the bounds of the graph due to the higher emphasis on the higher powers of the features. We can also see that the graphs themselves change shape significantly as the value of C changes. The shape of C=1000 has much more peaks and troughs than c=10.

The graphs produced are as follows: **If hard to read, zoom in, quality should scale up**



Appendix

I(A)

```
def load_data(file_name):
    file_path = os.path.join(os.path.dirname(__file__), file_name)
    column_names = ['Feature1', 'Feature2', 'Target']
    return pd.read_csv(file_path, names=column_names, skiprows=1)

def plot_3d_scatter(data, angles, output_file):
    fig, axes = plt.subplots(2, 2, subplot_kw={'projection': '3d'}, figsize=(12, 12))
    for ax, (elev, azim) in zip(axes.flatten(), angles):
        ax.scatter(data['Feature1'], data['Feature2'], data['Target'], color='red')
```

```
        ax.set_xlabel('Feature 1')
        ax.set_ylabel('Feature 2')
        ax.set_zlabel('Target')
        ax.set_title(f'3D Scatter Plot')
        ax.view_init(elev=elev, azim=azim)
    plt.savefig(output_file)
    plt.close()

data = load_data('week3.csv')
angles = [(20, 30), (30, 60), (40, 90), (10, -60)]
plot_3d_scatter(data, angles, 'i(a).png')
```

I(B)

```
def regression_analysis(model_class, C_values, png_name):
    data = load_data('week3.csv')
    X = data[['Feature1', 'Feature2']].values
    y = data['Target'].values

    poly = PolynomialFeatures(degree=5)
    X_poly = poly.fit_transform(X)

    for C in C_values:
        model = model_class(alpha=1/(2*C), max_iter=10000)
        model.fit(X_poly, y)

        print(f'C: {C}')
        print(f'Coefficients: {model.coef_}')
        print(f'Intercept: {model.intercept_}')
        error = mean_squared_error(y, model.predict(X_poly))
        print(f'Mean square error: {error}')
        print("\n")

    plot_regression_results(model_class, C_values, X, y, X_poly, poly, png_name)

def plot_regression_results(model_class, C_values, X, y, X_poly, poly, png_name):
    X_test = []
    for i in np.linspace(-5, 5):
        for j in np.linspace(-5, 5):
            X_test.append([i, j])
    X_test = np.array(X_test)
    X_test_poly = poly.transform(X_test)

    fig, axes = plt.subplots(1, 4, subplot_kw={'projection': '3d'}, figsize=(24, 6))
    colors = ['blue', 'green', 'orange', 'purple']

    for ax, C, color in zip(axes, C_values, colors):
        model = model_class(alpha=1/(2*C), max_iter=10000)
        model.fit(X_poly, y)
        predictions = model.predict(X_test_poly).reshape((50, 50))

        ax.plot_surface(X_test[:, 0].reshape((50, 50)), X_test[:, 1].reshape((50, 50)), predictions,
```

```
        ax.scatter(X[:, 0], X[:, 1], y, color='red', label='Training Data')

        ax.set_xlabel('Feature 1')
        ax.set_ylabel('Feature 2')
        ax.set_zlabel('Target')
        ax.set_title(f'{model_class.__name__} predictions C = {C}')
        ax.legend()

print("\033[91m" + "Lasso regression:" + "\033[0m")
C_values = [1, 10, 1000, 10000]
regression_analysis(Lasso, C_values, 'i(c).png')

print("\033[91m" + "Ridge regression:" + "\033[0m")
C_values = [0.000001, 0.001, 1, 10]
regression_analysis(Ridge, C_values, 'i(e).png')
```