Assignment 01

Learning from Data, Related Challenges, Linear Models for Regression

submitted for

EN3150 - Pattern Recognition

Department of Electronic and Telecommunication Engineering
University of Moratuwa

Udugamasooriya P. H. J.
220658U
Progress on GitHub

20 August 2025

1 Impact of Outliers on Linear Regression

Question 02 We represent the independent variables in a matrix

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix},$$

the dependent variables in a vector

$$\mathbf{y} = \begin{pmatrix} y_1 & \cdots & y_n \end{pmatrix}^\top,$$

and directly use the result that

$$\mathbf{w}_{\text{OLS}} = \begin{pmatrix} w_0 & w_1 \end{pmatrix}^\top = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^2 = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}.$$

This is exactly what we implement in the code in Listing 1, and the results obtained from it are as follows:

```
Ordinary Least Squares Weights (w): [ 3.91672727 -3.55727273]
```

Hence,

$$\mathbf{w}_{\text{OLS}} = \begin{pmatrix} 3.91672727 \\ -3.55727273 \end{pmatrix},$$

and the predicted linear model is

$$y = 3.91672727 - 3.55727273x.$$

A plot of the given data points against the predicted values is shown in Figure 1.

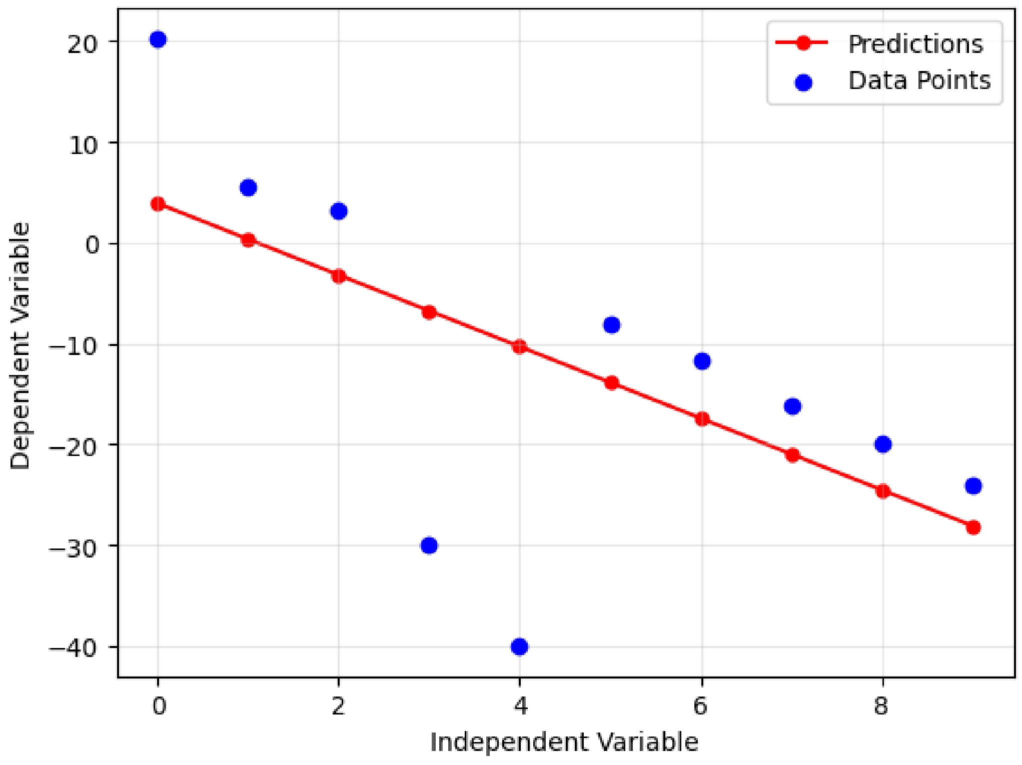

Figure 1: Given data points against predicted linear model

Question 04 The code in Listing 2 was used to calculate the loss for each model for each given value of β. The output of the code was the following:

```
Model 1 : [12 -4]
  Loss for beta = 1      : 0.435416262490386
  Loss for beta = 1e-06  : 0.9999999998258207
  Loss for beta = 1000.0 : 0.0002268287498440988
Model 2 : [ 3.91 -3.55]
  Loss for beta = 1      : 0.9728470518681676
  Loss for beta = 1e-06  : 0.9999999999999718
  Loss for beta = 1000.0 : 0.00018824684654645654
```

Summarizing these results in a table, we have

β	Model 1 Loss	Model 2 Loss
1	0.4354	0.9728
10^{-6}	0.9999	1.0000
10^{3}	0.0002	0.0002

Table 1: Losses for the two models each value of β

Question 05 We propose setting $\beta = 1$ to mitigate the impact of outliers. To see why, consider the following.

With very small values of β, the squared error term starts to dominate, and the loss becomes approximately equal to 1, i.e.,

$$\frac{(y_i - \hat{y}_i)^2}{(y_i - \hat{y}_i)^2 + \beta^2} \approx \frac{(y_i - \hat{y}_i)^2}{(y_i - \hat{y}_i)^2} = 1,$$

making the result almost independent of the model used, and making it difficult to distinguish between several models.

Very large values of β would cause the loss to be approximately proportional to the squared error and very close to 0, i.e.,

$$\frac{(y_i - \hat{y}_i)^2}{(y_i - \hat{y}_i)^2 + \beta^2} \approx \frac{(y_i - \hat{y}_i)^2}{\beta^2} \approx 0,$$

again making it difficult to distinguish between models. Further, inimizing the loss in this case would yield approximately the same result as that of minimizing the mean squared error.

It can be seen from the results above that $\beta = 10^3$ is too large, as the resulting losses from both models are both approximately equal, and very small and close to zero, making it difficult to distinguish between the two models.

It can also be seen that $\beta = 10^{-6}$ is too small, as the resulting losses from the models are again both approximately equal, but this time close to 1, leading to the same problem as above.

Hence, $\beta = 1$ is the best choice of the given options, as it has yielded comparable losses for both models.

Question 06 We will fix $\beta = 1$. The loss for Model 1 then, is 0.4354, whereas the loss for Model 2 is 0.9728. Model 1 has a lower loss and is therefore the better/more suitable model.

Furthermore, the plots in Figure 2 show that Model 1 passes through the inliers more closely whereas Model 2 is farther away from the inliers (due to the influence of the outliers). This verifies that Model 1 is indeed the better model.

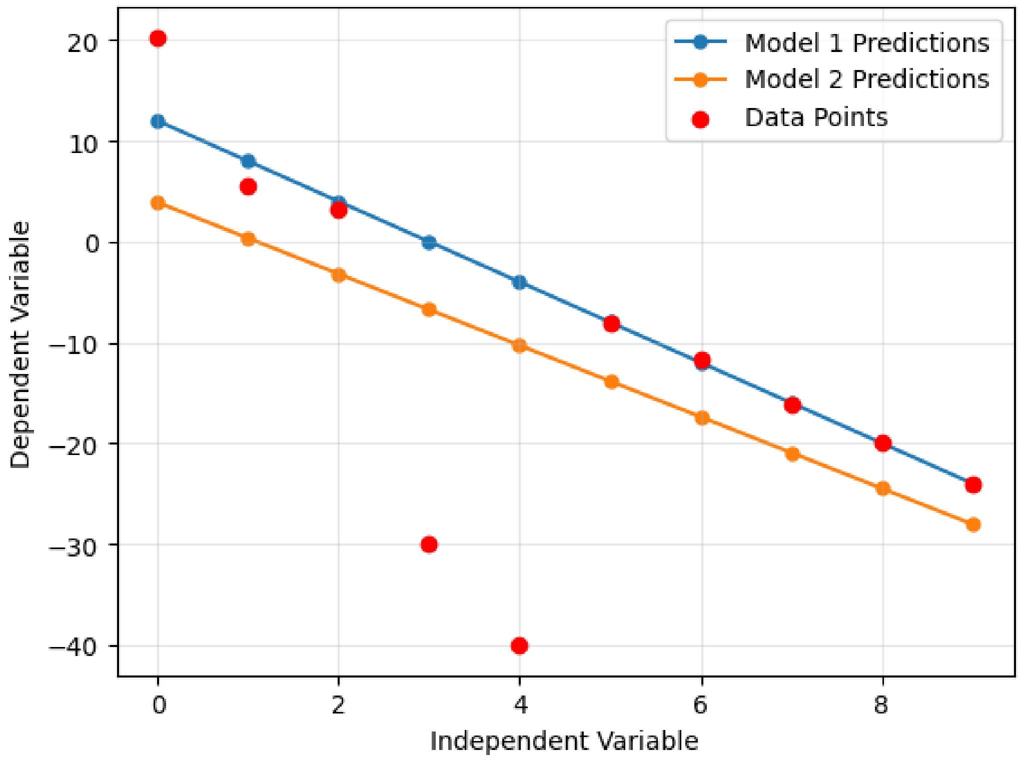

Figure 2: Comparison of Models 1 and 2

Question 07 Let us start by rewriting the loss for a single data point as follows:

$$L\left(y_i, x_i, \theta, \beta\right) = \begin{cases} \dfrac{1}{1 + \frac{\beta^2}{\left(y_i - \hat{y_i}\right)^2}}, & y_i \neq \hat{y_i}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\hat{y_i}$ is the prediction for the data point x_i, obtained by applying the model described by the parameters θ.

Now because $\left(y_i - \hat{y_i}\right)^2 \geq 0$ for all $\hat{y_i}$, it is clear that $0 \leq L\left(y_i, x_i, \theta, \beta\right) < 1$ also, with small values of $\left(y_i - \hat{y_i}\right)^2$ resulting in losses close to 0 and large values resulting in losses close to 1. Importantly, we observe that this property remains true even for outliers.

This has the effect of keeping the loss always restricted to a finite range, even in the presence of outliers, unlike with unbounded losses such as the squared error, which can grow to very large values.

Note also that because the loss due to each data point is always non-negative, the mean loss over all the data points is minimized when the loss from each individual data point is as small as possible.

Suppose we choose some threshold $E \in (0, 1)$ such that we would like to have

$$\frac{1}{1 + \frac{\beta^2}{\left(y_i - \hat{y_i}\right)^2}} \leq E$$

for each data point. But this is equivalent to requiring

$$\left(y_i - \hat{y_i}\right)^2 \leq \frac{E}{1 - E} \cdot \beta^2, \quad \text{or} \quad \left|y_i - \hat{y_i}\right| \leq \sqrt{\frac{E}{1 - E}} \cdot \beta,$$

i.e., it defines an interval of values centered around the actual value y_i that the predicted $\hat{y_i}$ might lie within, for which the loss can still be considered "small enough".

In this context, we can see β acts as a hyperparameter that allows us to control how big or small the above interval is. It can be chosen big enough to include the distance that the outliers are away from what one might expect their true values to be.

Hence, with this modification in place, we prevent the outliers from introducing very large losses, and encourage the model to focus more on minimzing the loss due to the inliers, which significantly outnumber the outliers and thereby contribute more significantly to the mean loss.

Question 08 Based on the nature of the outliers in the given data, we need a loss that does not completely discard them, as the outliers in this case still seem to follow the same general trend as the inliers and are not extremely far away from them.

Hence, a loss function that produces a significant loss for inliers, but a more slowly increasing loss for outliers is appropriate.

We therefore propose the Huber Loss function, defined as follows ([1]):

$$H\left(y_i, x_i, \theta, \delta\right) = \begin{cases} \frac{1}{2}\left(y_i - \hat{y_i}\right)^2, & \left|y_i - \hat{y_i}\right| \leq \delta \\ \delta \cdot \left(\left|y_i - \hat{y_i}\right| - \frac{1}{2}\delta\right), & \text{otherwise,} \end{cases}$$

where y_i, x_i, θ and $\hat{y_i}$ stand for the same quantities as introduced above, and δ is a hyperparameter that determines an interval of variation for points that can be considered inliers; the Huber Loss is exactly the squared error for such points, but just an absolute difference for other points, i.e., outliers.

2 Loss Functions

Question 01 We calculate the squared error

$$\text{SE}(\hat{y_i}, y_i) = (\hat{y_i} - y_i)^2$$

and binary cross entropy

$$\text{BCE}(\hat{y_i}, y_i) = -y_i \log(\hat{y_i}) - (1 - y_i) \log(1 - \hat{y_i})$$

of each predicted value $\hat{y_i}$ against the given corresponding target value y_i.

True Value (y_i)	Predicted Value ($\hat{y_i}$)	SE($\hat{y_i}, y_i$)	BCE($\hat{y_i}, y_i$)
1	0.005	0.9900	5.2983
1	0.01	0.9801	4.6052
1	0.05	0.9025	2.9957
1	0.1	0.8100	2.3026
1	0.2	0.6400	1.6094
1	0.3	0.4900	1.2040
1	0.4	0.3600	0.9163
1	0.5	0.2500	0.6931
1	0.6	0.1600	0.5108
1	0.7	0.0900	0.3567
1	0.8	0.0400	0.2231
1	0.9	0.0100	0.1054
1	1.0	0.0000	0.0000
	Mean	0.4402	1.4407

Table 2: Loss for each predicted value from both choices for the loss function

A plot of the different losses against the predicted values is shown in Figure 3.

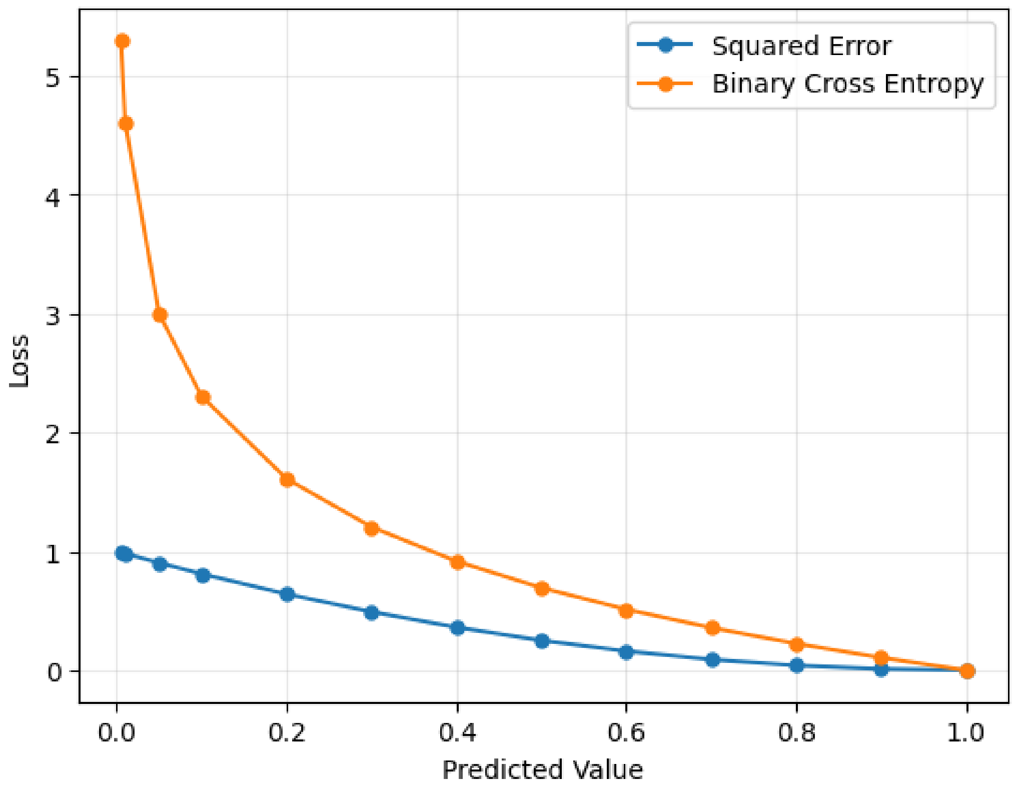

Figure 3: MSE and BCE loss against predicted value

Question 02 The MSE loss is more appropriate for Application 1 and the BCE loss is more appropriate for Application 2.

For Application 1, where the quantity being predicted is a continuous value, we would like the loss to vary gradually with difference between the predicted value and true value.

For Application 2, the goal is to map a continuous value to a binary value based on a certain threshold. In this case, we would like to heavily penalize predictions the farther they away below (or above) the threshold, but give little to no penalty for predictions on the other side of the threshold, with the hope that it will quickly be forced to the correct side if it happens to land on the wrong side.

3 Data Pre-Processing

Question 01 We start by visualizing the distribution of each feature value to decide on a suitable method of scaling. Figure 4 contains plots obtained of the distribution of each feature in the dataset.

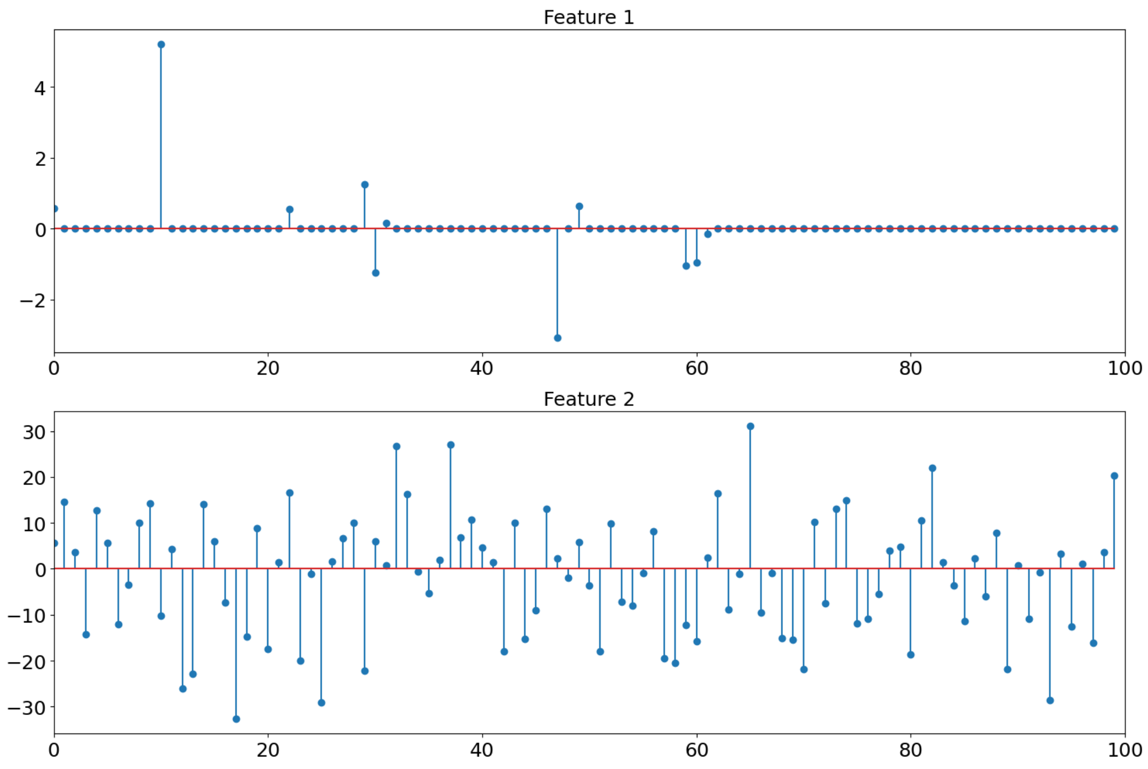

Figure 4: Raw values of Feature 1 and Feature 2 for each data point

We then run the code in Listing 3 to obtain the following summary statistics of each feature:

```
Feature 1
  Mean               : 0.019184855901577556
  Standard Deviation : 0.6522329762544182
  Maximum            : 5.2
  Minimum            : -3.068923231010246
  Range              : 8.268923231010247
Feature 2
  Mean               : -1.7085235048185898
  Standard Deviation : 13.302554852300073
  Maximum            : 31.13858494901702
  Minimum            : -32.69009199576038
  Range              : 63.8286769447774
```

Based on the plots and the summary statistics above, we conclude the following;

- both features have means close to zero
- the features vary on different scales, as they have significantly different standard deviations
- both features take on both positive and negative values
- Feature 1 is sparsely distributed, with most values being equal to 0

To bring the values of both features to a similar scale while still preserving the structure and properties of each feature, we consider the three following scaling methods;

1. standard scaling,
2. min-max scaling, and
3. max-abs scaling.

We rule out min-max scaling as it would destroy the sparsity of Feature 1 by mapping all zeros to non-zero values, and limit both feature values to a range between 0 and 1, affecting their “symmetric” variation among both negative and positive values.

To choose between standard and max-abs scaling, we consider the sparsity of Feature 1.

We observe that standard scaling would map zeros of Feature 1 to non-zero values, resulting in a loss of sparsity—a propertsy one would likely want to preserve. On the other hand, max-abs scaling would not affect sparsity, as it would map zeros to zeros. It also maps to a range between -1 and 1, so negative values map to negative values, and positives to positives.

Hence, we choose max-abs scaling as the method of scaling for both feature values. The code in Listing 4 implements max-abs scaling. Plots of both feature values after such scaling are shown in Figure 5.

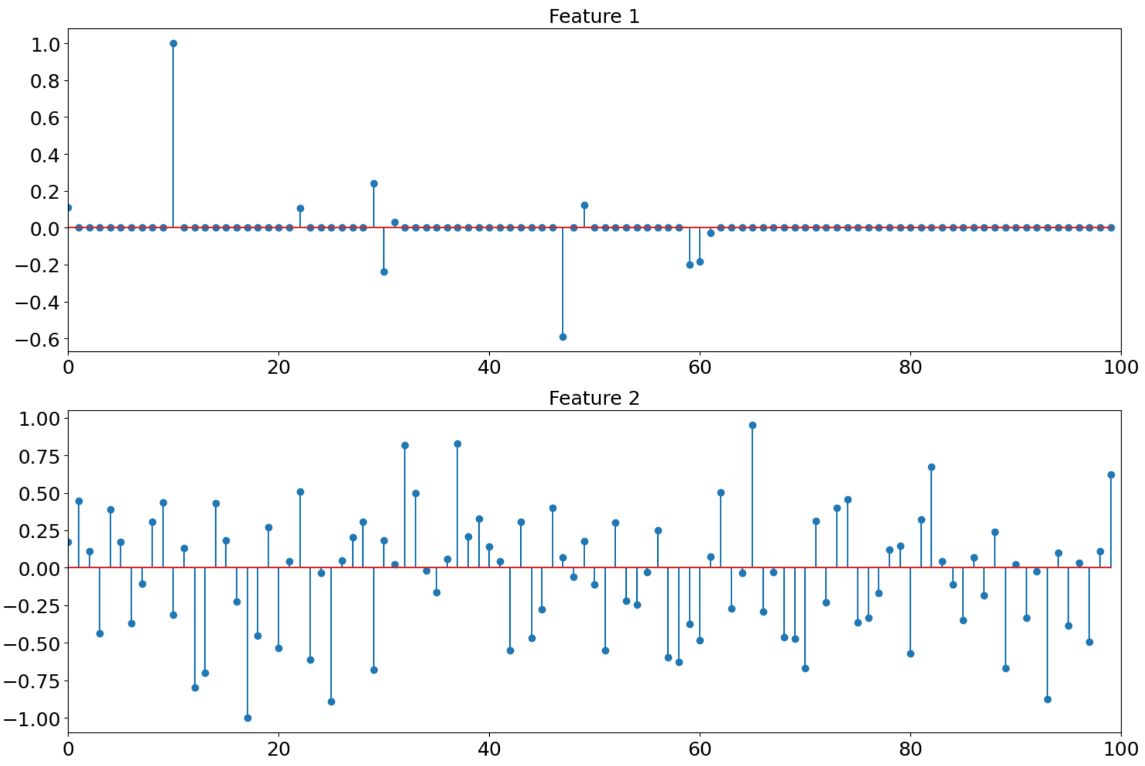

Figure 5: Max-abs scaled values of Feature 1 and Feature 2 for each data point

The recalculated summary statistics for the scaled features are as follows:

```
Feature 1 - Max-Abs Scaled
  Mean                  : 0.0036893953656879897
  Standard Deviation    : 0.12542941851046505
  Maximum               : 1.0
  Minimum               : -0.5901775444250472
  Range                 : 1.5901775444250472
Feature 2 - Max-Abs Scaled
  Mean                  : -0.052264261141882686
  Standard Deviation    : 0.40692925715918127
  Maximum               : 0.9525389207548091
  Minimum               : -1.0
  Range                 : 1.952538920754809
```

For the sake of comparison, we also include plots of min-max scaling and standard scaling applied to both features in Figure 6. The code implementing min-max and standard scaling is included in Listing 5.

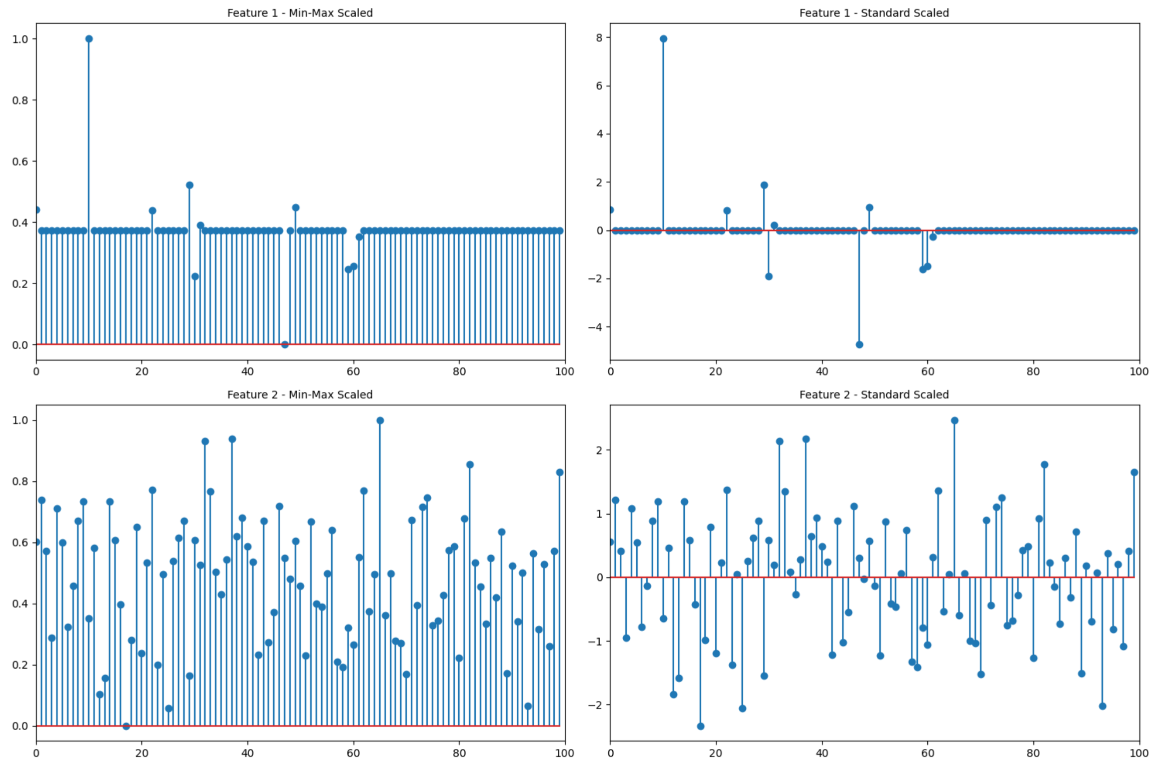

Figure 6: Min-Max and Standard Scaled values of Feature 1 and Feature 2 for each data point

References

[1] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, Mar. 1964.

A Code Snippets

A.1 Impact of Outliers on Linear Regression

A.1.1 Question 02

```
# Load the data set
x = np.arange(10).T
y = np.array([20.26, 5.61, 3.14, -30, -40, -8.13, -11.73, -16.08, -19.95,
    -24.03]).T

# Length of the data set
N = len(x)

# Construct a matrix X with each data point in one row, and each column
    representing a feature
# The first entry of each row will be 1
X = np.ones((N, 2))
X[:, 1] = x
```

```python
# We want to choose a matrix of weights w such that (y - xW)^T * (y - xW) is
    minimized
# The w that minimizes the above--the ordinary least squares weights--is given by
w_OLS = np.linalg.inv(X.T @ X) @ X.T @ y

# Print the learned ordinary least squares weights
print ("Ordinary Least Squares Weights (w):", w_OLS)

# Calculate the prediction for each data point based on the learned w
y_hat = X @ w_OLS

# Plot the original data points and the predicted values
plt.plot(y_hat, label='Predictions', color='red', markevery=x, marker='o',
    markersize=5)
plt.scatter(x, y, label='Data Points', color='blue')

plt.legend()
plt.xlabel('Independent Variable')
plt.ylabel('Dependent Variable')
plt.grid(True, alpha=0.3)

plt.show()
```

Listing 1: Ordinary Least Squares Regression and Plots

A.1.2 Question 04

```python
# Set up the weight vectors representing the two given modelss
w_1 = np.array([12, -4])
w_2 = np.array([3.91, -3.55])

# Iterate through each model
for i in range(2):
    w = [w_1, w_2][i]
    print ("Model", i + 1, ":", w)

    # Run the model on the data and find the residuals
    y_hat = X @ w
    residuals = y - y_hat

    # Plot the values predicted by the model
    plt.plot(y_hat, label='Model ' + str(i + 1) + ' Predictions', markevery=x,
        marker='o', markersize=5)

    # Vary beta through each of the given values and find the loss
    for beta in [1, 1E-6, 1E3]:
        loss = (1 / N) * np.sum((residuals ** 2) / (residuals ** 2 + beta ** 2))
        print ("\t Loss for beta =", "{:<6}".format(beta), ":", loss)
```

Listing 2: Loss Calculation for Different Models and Values of β

A.2 Data Pre-Processing

A.2.1 Question 01

```python
def print_summary_statistics(feature):
    mean = np.mean(feature)
    std_dev = np.std(feature)
    maximum = np.max(feature)
    minimum = np.min(feature)
    range_ = maximum - minimum

    print ("\t Mean \t\t\t:", mean)
    print ("\t Standard Deviation \t:", std_dev)
    print ("\t Maximum \t\t:", maximum)
    print ("\t Minimum \t\t:", minimum)
    print ("\t Range \t\t\t:", range_)

for i in range(2):
    feature = [sparse_signal, epsilon][i]

    print ("Feature", i + 1)
    print_summary_statistics(feature)
```

Listing 3: Calculation of Summary Statistics

```python
def max_abs_scale(feature):
    max_abs = np.max(np.abs(feature))
    if max_abs == 0: # Avoid division by zero
        return feature
    return feature / max_abs
```

Listing 4: Max-Abs Scaling

```python
def min_max_scale(feature):
    minimum = np.min(feature)
    maximum = np.max(feature)
    if maximum == minimum: # Avoid division by zero
        return feature
    return (feature - minimum) / (maximum - minimum)

def standard_scale(feature):
    mean = np.mean(feature)
    std_dev = np.std(feature)
    if std_dev == 0: # Avoid division by zero
        return feature
    return (feature - mean) / std_dev
```

Listing 5: Min-Max and Standard Scaling