DEPARTMENT OF ELECTRONIC AND TELECOMMUNICATION ENGINEERING

UNIVERSITY OF MORATUWA

EN3150 Assignment 01

Learning from data and related

challenges and linear models for regression

PUSHPAKUMARA H.M.R.M.

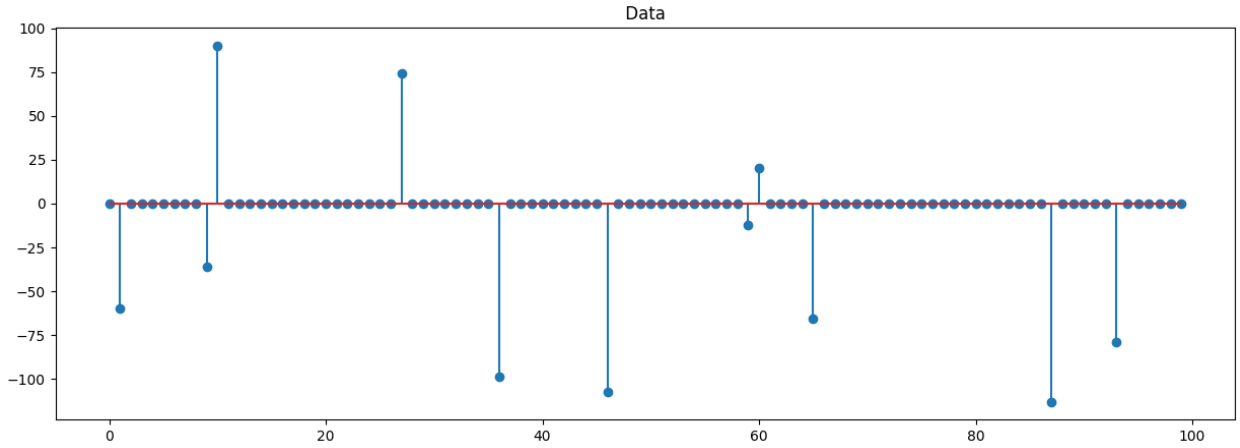
200488E

September 11, 2023

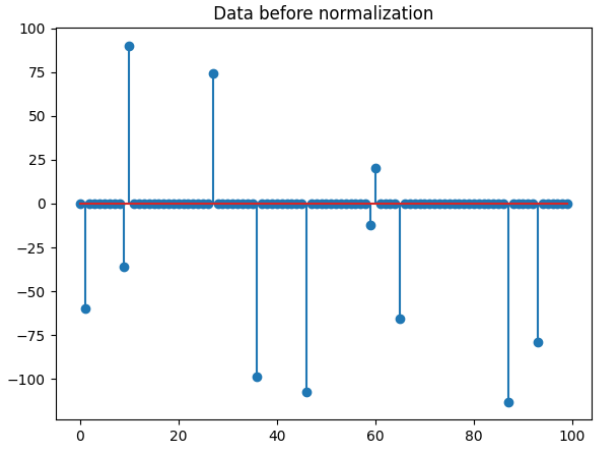
Data preprocessing

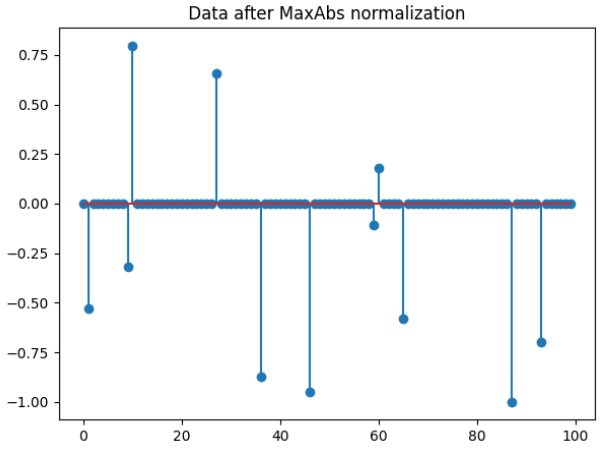
1. Index Number: 200488E. I used 200488 to generate customized data.

2. Generated data

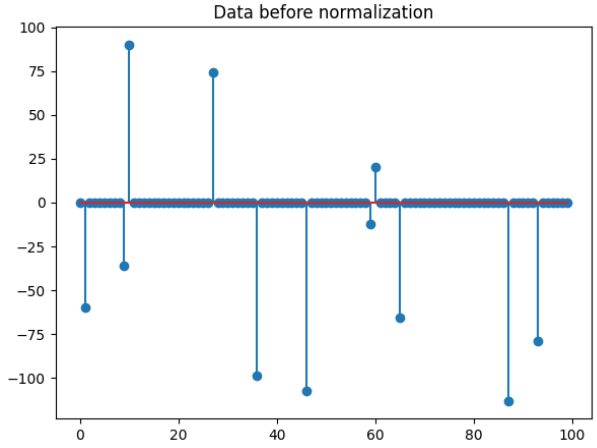


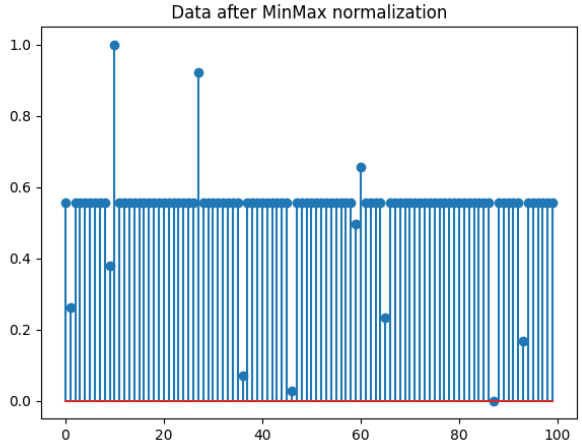
3,4. **MaxAbsScaler**



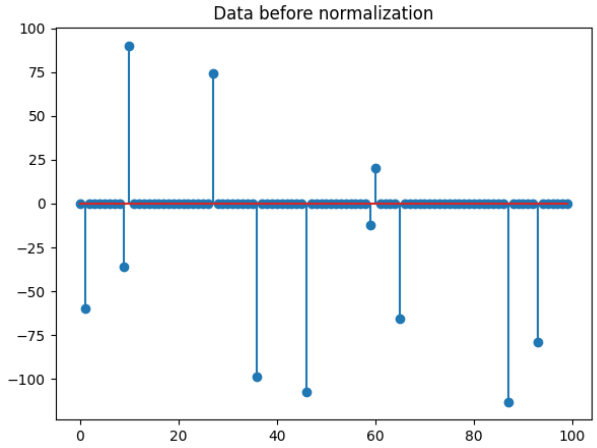


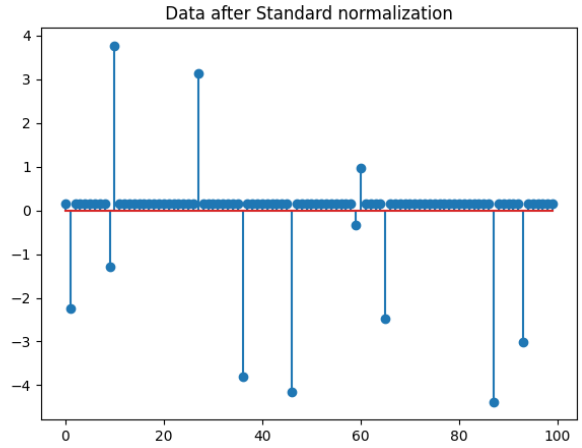
Min-Max Scaler





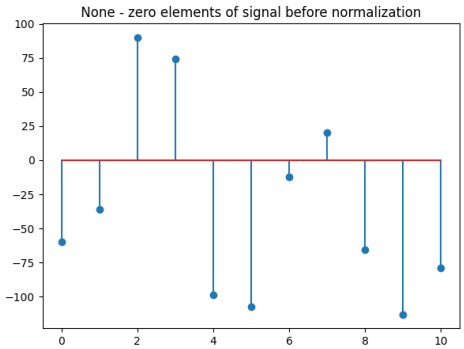
Standard Scaler

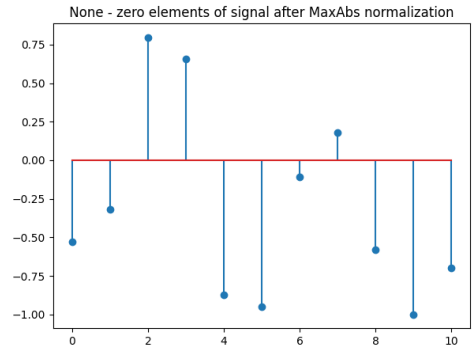




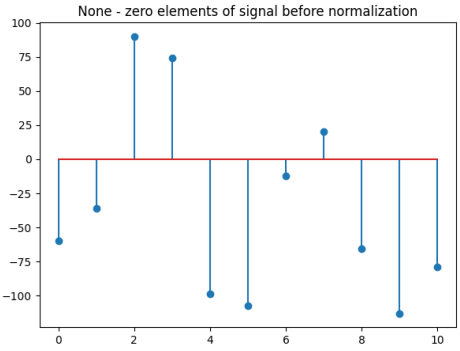
5. Number of non-zero elements in the signal **before normalization**: 11

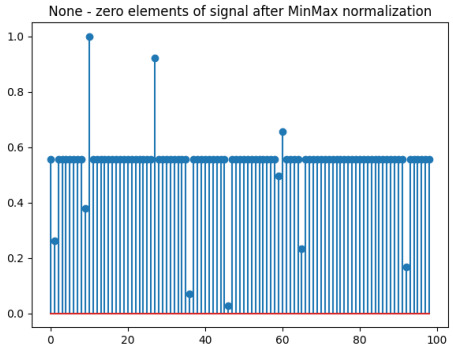
Number of non-zero elements in signal after **MaxAbs normalization**: 11



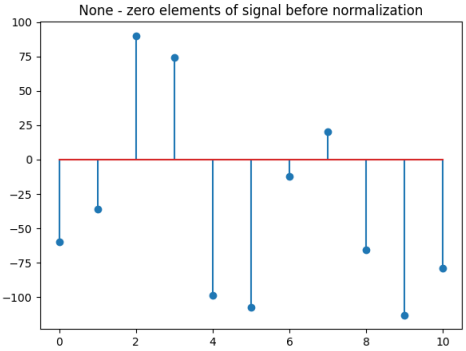


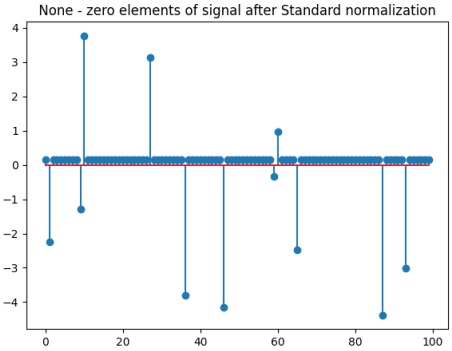
Number of non-zero elements in signal after **MinMax normalization**: 99





Number of non-zero elements in signal after **Standard normalization**: 100





6. **MaxAbsScaler**

MaxAbsScaler scales each feature by its maximum absolute value. It essentially divides each feature by the maximum absolute value present in that feature. This scaling technique is useful when our data contains both positive and negative values and we want to maintain the original sign of the data.

MaxAbsScaler is often used when the distribution of the data is not necessarily Gaussian and when the data has outliers.

Min-Max Scaler

Min-Max Scaler, also known as Min-Max normalization, scales features to a specified range, typically between 0 and 1. It linearly transforms each feature so that the minimum value becomes 0, and the maximum value becomes 1, with values in between scaled proportionally.

Min-Max Scaler is widely used when you want to scale features to a common range and maintain the relationships between data points. It is especially useful when you have features with different units or scales.

Standard Scaler

Standard Scaler standardizes features by subtracting the mean and dividing by the standard deviation of each feature. This transformation results in each feature having a mean of 0 and a standard deviation of 1. It assumes that the data follows a Gaussian (normal) distribution.

Standard Scaler is commonly used when your data has a Gaussian distribution, and you want to center the data around zero with unit variance. It's useful for many machine learning algorithms that assume standardized data.

7. **MaxAbsScaler**

Scaling Range: The data is scaled within the range [-1, 1] because it divides each feature by its maximum absolute value.

```
count  100.000000
mean    -0.034306
std      0.221582
min     -1.000000
25%      0.000000
50%      0.000000
75%      0.000000
max      0.796255
```

Impact on Structure:

Mean: The mean is shifted closer to 0 but still retains the same sign as the original data.

Standard Deviation: Reduced to a fraction of the original, making the data less spread out.

Min and Max: Scaled within the [-1, 1] range.

Percentiles: The percentiles retain their relative positions but are scaled to the [-1, 1] range.

Overall Impact: MaxAbsScaler maintains the data's relative relationships and signs while significantly reducing the magnitude. It's suitable when preserving the sign and relative magnitude is important, but it doesn't preserve the original scale.

Min-Max Scaler

Scaling Range: The data is scaled within the range [0, 1] because it linearly transforms each feature to this range.

count	100.000000
mean	0.537615
std	0.123358
min	0.000000
25%	0.556714
50%	0.556714
75%	0.556714
max	1.000000

Impact on Structure:

Mean: The mean is shifted to a value between 0 and 1.

Standard Deviation: Reduced to a fraction of the original, making the data less spread out within [0, 1].

Min and Max: Transformed to 0 and 1, respectively.

Percentiles: The percentiles retain their relative positions but are scaled to the [0, 1] range.

Overall Impact: MinMaxScaler scales the data proportionally within a bounded range [0, 1], preserving relative relationships while enforcing the new scale.

Standard Scaler

Scaling Range: The data is scaled to have a mean of 0 and a standard deviation of 1.

count	1.000000e+02
mean	2.886580e-17
std	1.005038e+00
min	-4.380143e+00
25%	1.556026e-01
50%	1.556026e-01
75%	1.556026e-01
max	3.767214e+00

Impact on Structure:

Mean: Centered around 0.

Standard Deviation: Scaled to 1.

Min and Max: Outliers may be significantly below -3 or above 3, suggesting a departure from the normal distribution.

Percentiles: Shifted and scaled to be centered around 0 with a standard deviation of 1.

Overall Impact: Standard Scaler significantly changes the scale and distribution of the data, centering it around zero with unit variance. It may not preserve the original data's distribution or relative relationships.

Linear regression on real-world data

1.

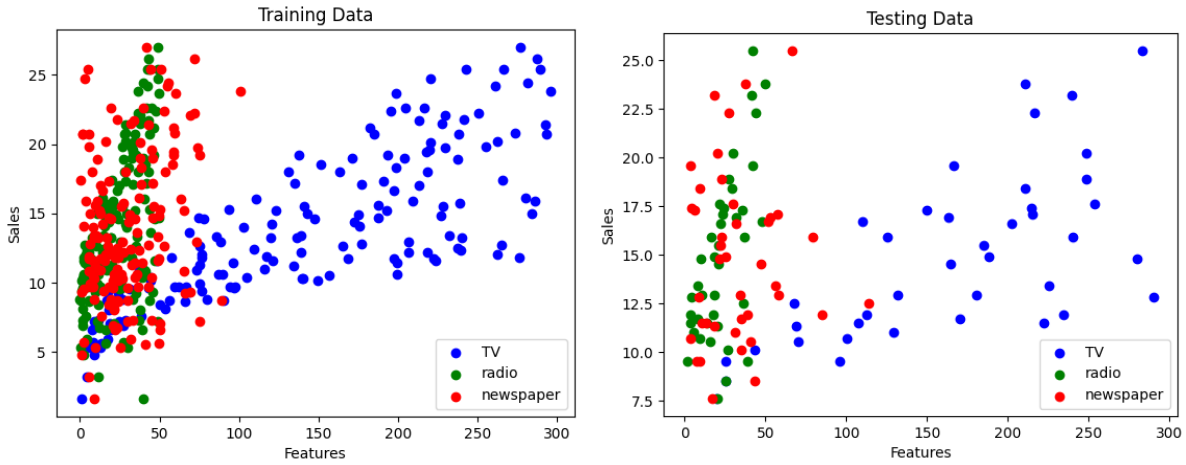
Sample	Index	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

```
   sample index     TV  radio  newspaper  sales
0             1  230.1   37.8       69.2   22.1
1             2   44.5   39.3       45.1   10.4
2             3   17.2   45.9       69.3    9.3
3             4  151.5   41.3       58.5   18.5
4             5  180.8   10.8       58.4   12.9
```

2.

```
# Split data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
X_train shape:  (160, 3)
X_test shape:  (40, 3)
y_train shape:  (160,)
y_test shape:  (40,)
```



3.

```
# Train the model using Linear Regression
# Model parameters: x1=TV, x2=radio, x3=newspaper, y=sales
#y = w0 + w1*x1 + w2*x2 + w3*x3

# Import LinearRegression from sklearn
from sklearn.linear_model import LinearRegression

# Create a LinearRegression object
model = LinearRegression()

# Train the model using training data
model.fit(X_train, y_train)
```

$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$

Model Parameters;

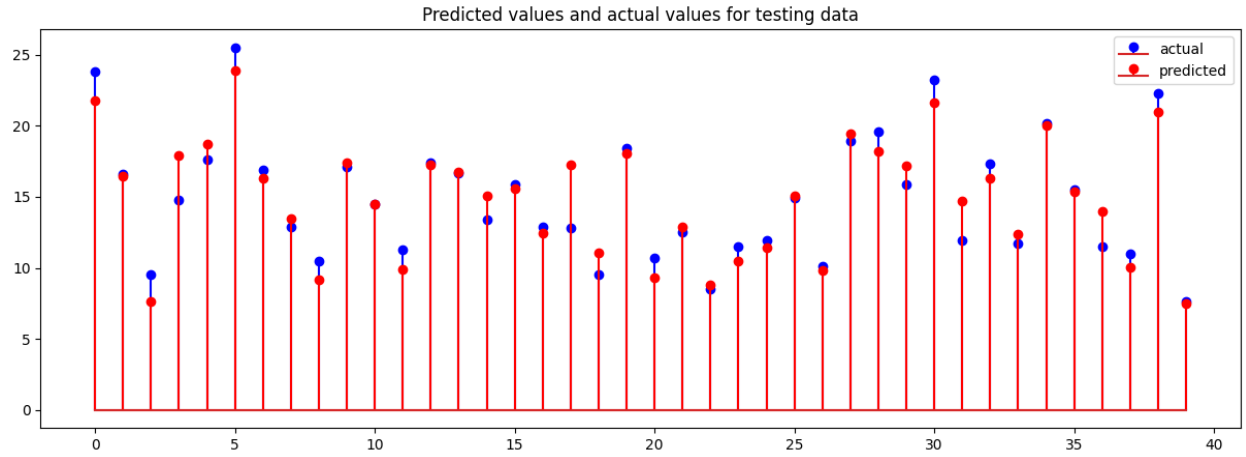
w_0 = 2.907947020816433 (Intercept)

w_1= 0.0468431

w_2 = 0.17854434

w_3 = 0.00258619

4.



Statistics	For Training	For Testing
Residual sum of squares (RSS)	482.6928972255484	79.67542207315162
Residual Standard Error (RSE)	1.7590296298315473	0.7146606782832712
Mean Squared Error (MSE)	3.0168306076596774	0.49797138795719764
R2 statistic	0.8959372632325174	0.8927605914615384

For training data

	Standard Error	t-statistic	p-value
w_0	0.2657589245679816	10.942048420551066	0.0
TV	0.0015915882844085901	29.431671265660643	0.0
radio	0.009153205539098987	19.506209387105205	0.0
newspaper	0.006616785718567478	0.3908523267924272	0.6964396551006735

For testing data

	Standard Error	t-statistic	p-value
w_0	0.25523244755384117	11.393328115944204	0.0
TV	0.001499017049088985	31.24921307963703	0.0
radio	0.008583483986745346	20.800917679183065	0.0
newspaper	0.004669375784666661	0.5538612039925214	0.5804668591108091

6.

Based on the statistics for training and testing data, it appears that there is a relationship between advertising budgets (TV, radio, and newspaper) and sales. Because,

1. **R2 statistic:** The R2 statistic (coefficient of determination) measures the proportion of the variance in the dependent variable (sales) that is explained by the independent variables (advertising budgets). In both the training and testing data sets, the R2 statistic is close to 1, which indicates a strong relationship between advertising budgets and sales. Specifically, the training data has an R2 of approximately 0.896, and the testing data has an R2 of approximately 0.893.
2. **p-values for coefficients:** In both the training and testing data, the p-values for the coefficients of TV and radio are very close to zero ($p \approx 0.0$), which suggests that these variables are statistically significant predictors of sales. The p-values for newspaper are higher but still relatively low ($p \approx 0.696$ for training and $p \approx 0.580$ for testing), indicating that it may also have some influence on sales.

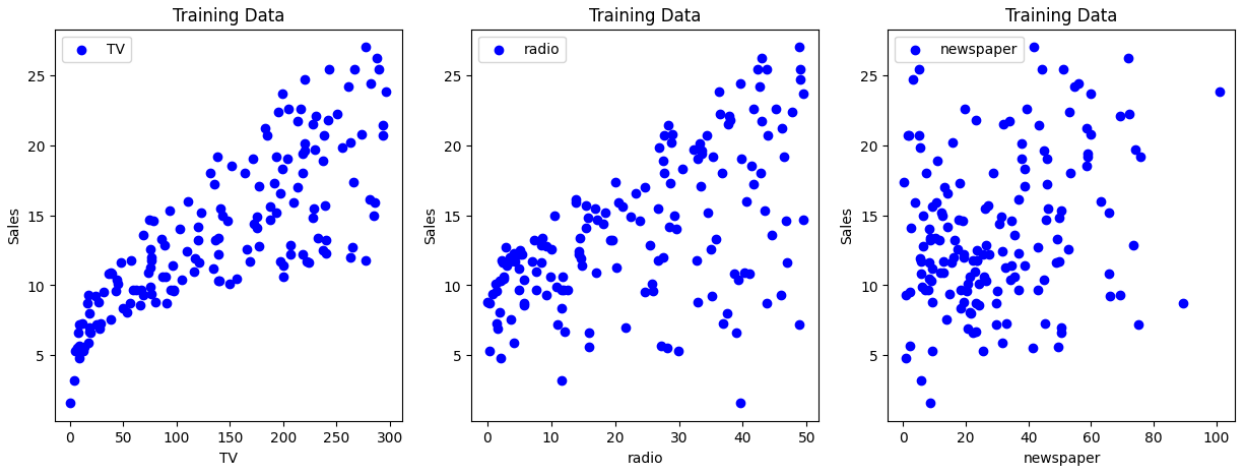
3. **T-statistics:** The t-statistics for TV and radio in both the training and testing data are significantly different from zero, with high absolute values. This indicates that the coefficients for these variables are highly statistically significant.
4. **Standard Errors:** The standard errors for TV and radio are relatively low, indicating that the estimated coefficients for these variables are precise and reliable.

7.

Based on the statistics, it appears that the independent variable "TV" contributes highly to sales. Because,

1. **Coefficient Significance:** The p-value for the coefficient of "TV" is very close to zero ($p \approx 0.0$) for both the training and testing data. A low p-value indicates that the coefficient is statistically significant.
2. **T-statistic:** The t-statistic for "TV" is significantly different from zero, with a high absolute value. A high t-statistic suggests that the coefficient for "TV" is highly statistically significant.
3. **R2 Statistic:** The R2 statistic, which measures the proportion of variance in sales explained by the independent variables, is relatively high for "TV" in both the training and testing data sets (close to 0.896 and 0.893, respectively). This indicates a strong relationship between "TV" and sales.
4. **Standard Error:** The standard error for "TV" is relatively low, indicating that the estimated coefficient for "TV" is precise and reliable.

We can see this when we plot features vs. sales.



7.

```
# Investing 25k dollers for TV and radio advertising
model_sm.predict([1, 25, 25, 0])
```

```
array([8.5426332])
```

```
# Investing 50k dollers only for TV advertising
model_sm.predict([1, 50, 0, 0])
```

```
array([5.25010218])
```

```
# Investing 10k dollers only for radio advertising
model_sm.predict([1, 0, 50, 0])
```

✓ 0.0s

```
array([11.83516421])
```

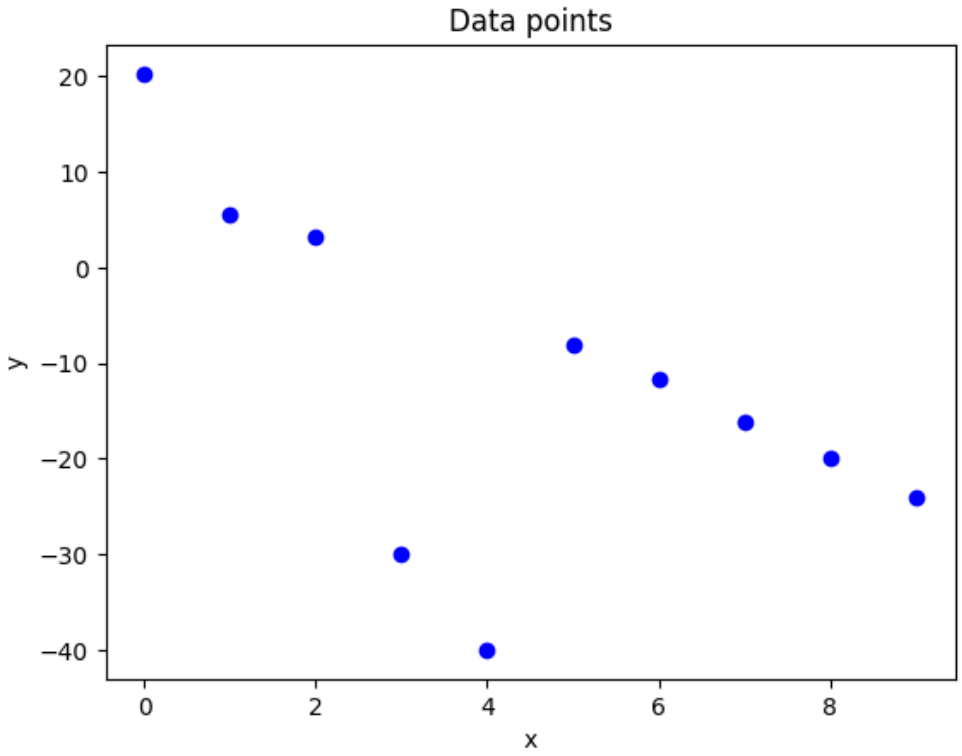
According to model prediction,

1) If we allocate 25,000 dollars for TV and radio it shows 8542 sales.
2) If we allocate 50,000 dollars only for TV, it shows 5250 sales.
3) If we allocate 50,000 dollars only for radio, it shows 11835 sales.

Therefore, according to our model, allocating 25,000 dollars for TV and radio equally doesn't yield higher sales compared with allocating 50,000 dollars for radio only.

Linear regression impact on outliers

1.



2.

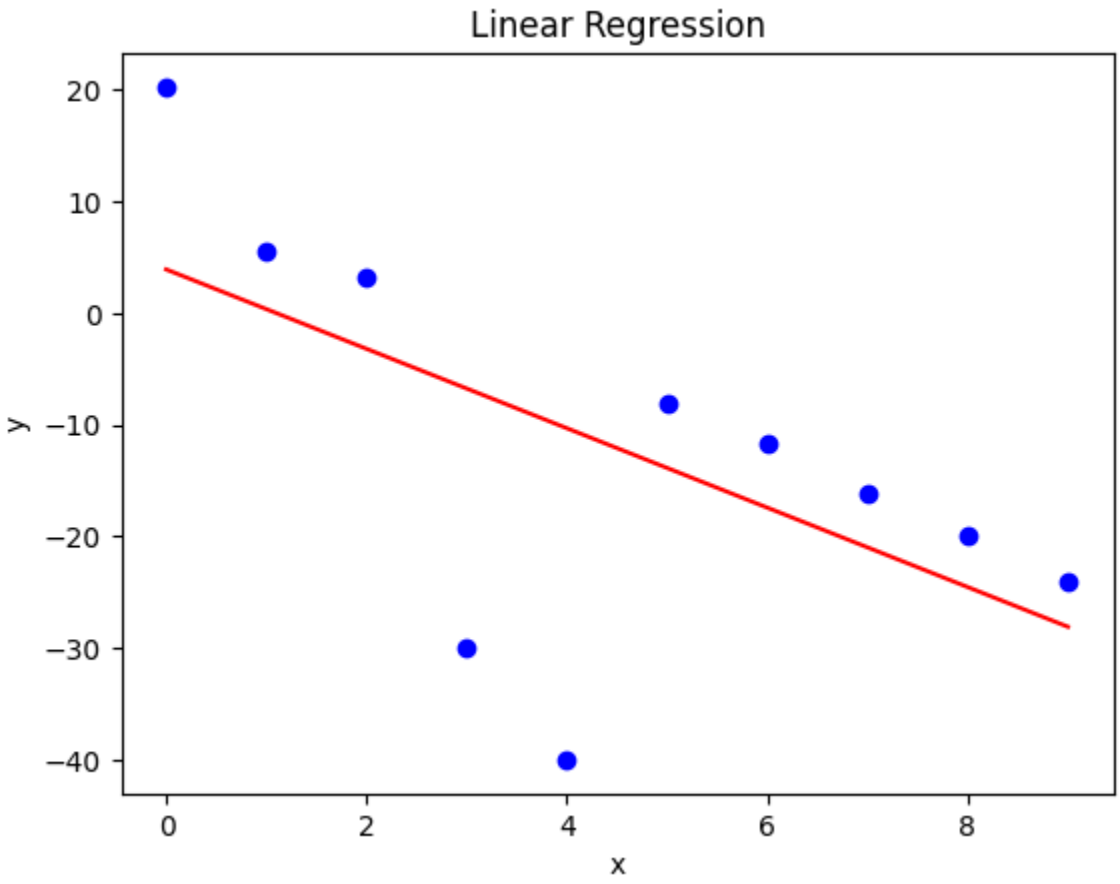
```python
# import LinearRegression class from sklearn.linear_model
from sklearn.linear_model import LinearRegression
# Create linear regression object
model = LinearRegression()

# Train the model using the training sets
model.fit(x, y)

print('Intercept:', model.intercept_)
print('Slope:', model.coef_)

# Plot the regression line and scatter points
y_pred = model.intercept_ + model.coef_ * x
plt.plot(x, y_pred, color = 'red')
plt.scatter(x, y, color = 'blue')
plt.title('Linear Regression')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

$\mathbf{y = w_0 + w_1 x}$

where;

$w_0 = 3.916727272727277$

$w_1 = -3.55727273$



3.

```python
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]).reshape((-1, 1))
y = np.array([20.26, 5.61, 3.14, -30.00, -40.00, -8.13, -11.73, -16.08, -19.95, -24.03])

# Combine the new data samples into a list of (x, y) pairs
data_samples = list(zip(x.ravel(), y))

def calculate_loss(model, data_samples, beta=1):

    # Extract model parameters (slope and intercept)
    slope, intercept = model

    # Calculate L(θ, β) for each data point and sum them up
    total_loss = 0
    for xi, yi in data_samples:
        y_hat_i = slope * xi + intercept
        loss_i = ((yi - y_hat_i)**2) / (((yi - y_hat_i)**2) + beta**2)
        total_loss += loss_i

    # Calculate the average loss over all data points
    avg_loss = total_loss / len(data_samples)

    return avg_loss

# Define the linear models as tuples (slope, intercept)
model1 = (-4, 12)
model2 = (-3.55, 3.91)

# Calculate L(θ, β) for Model 1 and Model 2
beta = 1
loss_model1 = calculate_loss(model1, data_samples, beta)
loss_model2 = calculate_loss(model2, data_samples, beta)
```

4. **For model 1**

L (θ, β) = 0.435416262490386

For model 2

L (θ, β) = 0.9728470518681676

5.

To determine the most suitable model for the provided dataset using the robust estimator, we need to select the model with the lowest loss function value ($L(\theta, \beta)$). In this case, we have two models with their respective loss function values:

Model 1: $L(\theta, \beta) = 0.435416262490386$, Model 2: $L(\theta, \beta) = 0.9728470518681676$

The lower the loss function value, the better the model performs according to the robust estimator. Therefore, in this context, Model 1 is the most suitable model because it has the lowest loss function value (0.435416262490386) compared to Model 2 (0.9728470518681676).

Justification:

- The robust estimator is designed to minimize the impact of outliers and provide more reliable model parameter estimates. In this case, Model 1 has a significantly lower loss function value, indicating that it provides a better fit to the data and minimizes the impact of outliers better than Model 2.

Therefore, based on the loss function values and the goal of reducing the impact of outliers, Model 1 is the more suitable choice for the dataset.

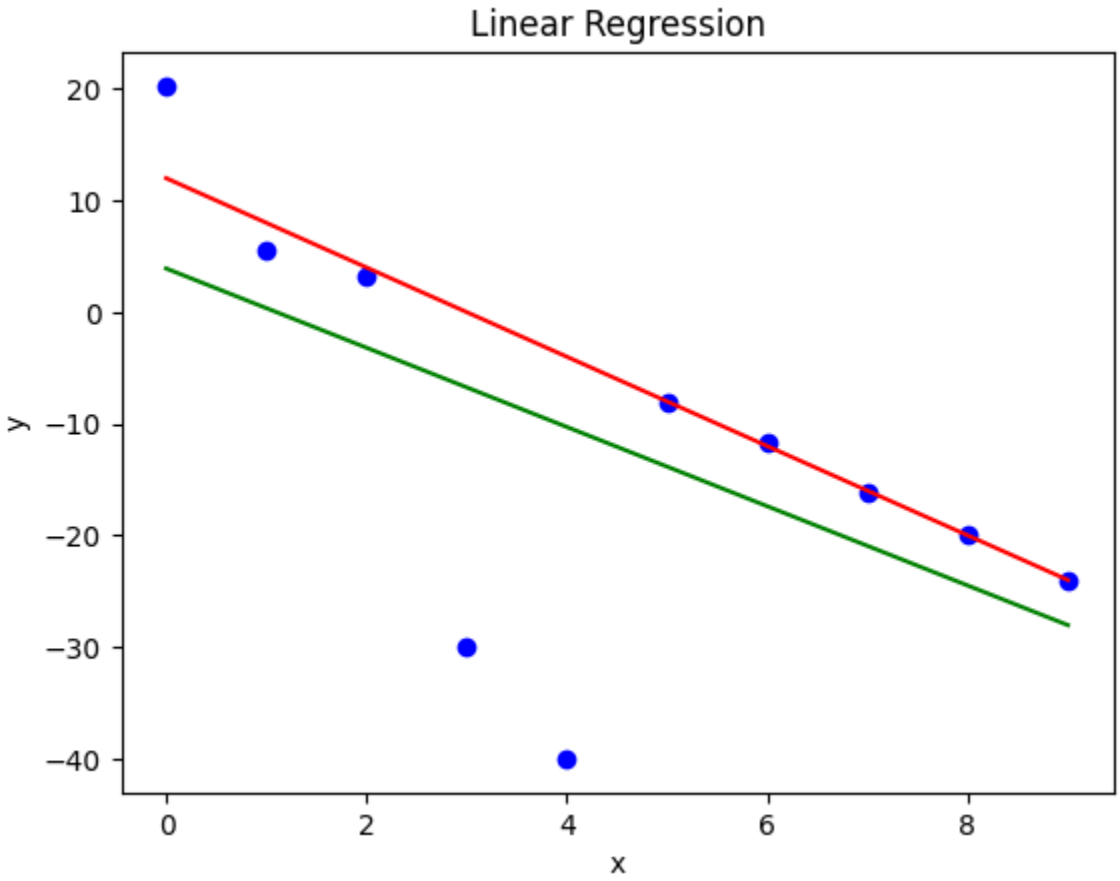
6.

For data points with small residuals (i.e., data points that are close to the model's predictions), the denominator in the loss function is dominated by the squared residual term, and the weight assigned to these data points is relatively high.

For data points with large residuals (i.e., outliers), the denominator in the loss function is influenced by both the squared residual and β^2. As β increases, the impact of the squared residual on the weight decreases, leading to a smaller weight for outliers.

By adjusting the value of β, you can control how much emphasis the loss function places on reducing the impact of outliers. Smaller values of β give more emphasis to robustness against outliers, while larger values of β make the loss function less robust.

7.



8.

The parameter β controls the impact of the denominator on the weighting. When β is small, the denominator is close to the squared residual, and the weight assigned to the data point is relatively high. When β is large, the denominator becomes closer to β^2, reducing the weight assigned to the data point.