Department of Electronic & Telecommunication Engineering

University of Moratuwa

EN3150 – Pattern Recognition

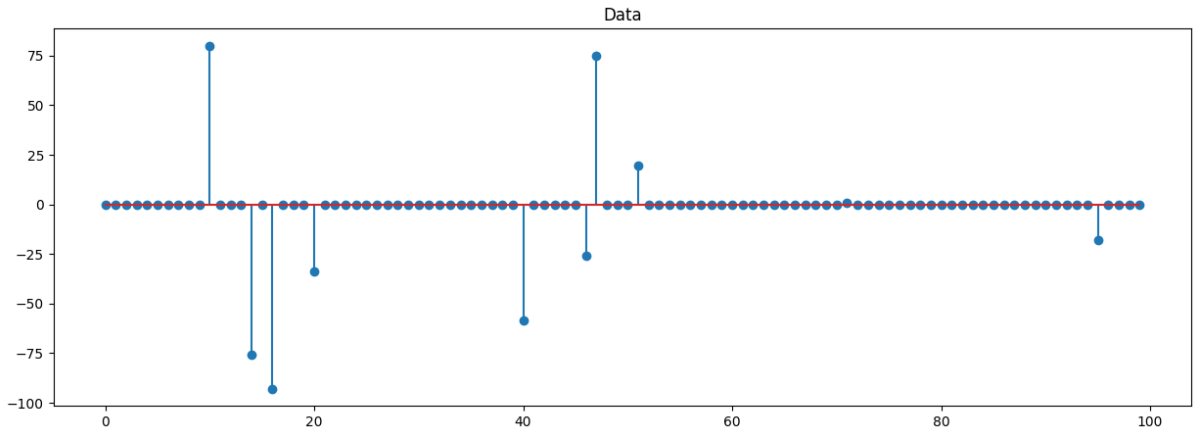
Learning from data and related challenges and linear models for regression

(Assignment 01 - Report)

200087A Chandrasiri Y.U.K.K.

1. Data Processing

Index Number for Data Generation: 200087 (From 200087A)



Data Normalization

1. MaxAbsScaler

$$MaxAbsScaler(x) = \frac{x}{\max(|x|)}$$

```python
from sklearn import preprocessing

MaxAbsScaler = preprocessing.MaxAbsScaler().fit(signal)
scaled_data_max_abs = MaxAbsScaler.transform(signal)
```

Python

2. Min-max Normalization

$$x_{scaled} = \frac{(x - min(x))}{(max(x) - min(x))}$$

```python
def min_max_scale(data):
    min_val = np.min(data)
    max_val = np.max(data)
    print("min of data", min_val, "max of data", max_val)
    scaled_data = (data - min_val) / (max_val - min_val)
    return scaled_data

scaled_data_min_max = min_max_scale(signal)
```

Python

3. Standard Normalization

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

Where:

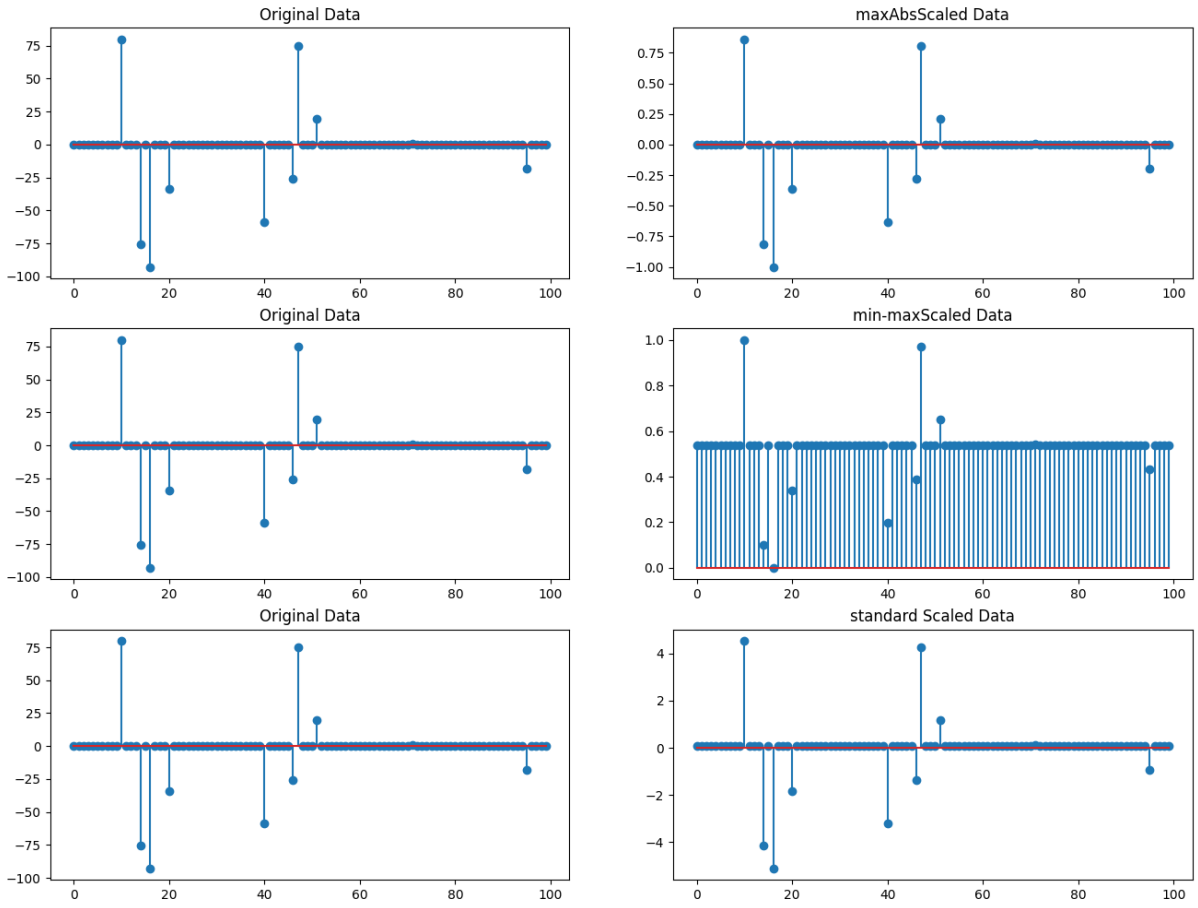
μ is the mean (average) of the data.

σ is the standard deviation of the data.

```python
def standard_scale(data):
    mean = np.mean(data)
    std = np.std(data)
    print("mean of data", mean, "std of data", std)
    scaled_data = (data - mean) / std
    return scaled_data

scaled_data_standard = standard_scale(signal)
```

Python

Data Before and After Normalization:



Non-zero elements in data before the normalization: 10

Non-zero elements in data after applying MaxAbs normalization: 10

Non-zero elements in data after applying Min-Max normalization: 99

Non-zero elements in data after applying Standard normalization: 100

How normalization method scales the data and its impact on structure

	MaxAbsScalar	min-max Normalization	Std. Normalization
count	100.00000	100.00000	100.00000
mean	-0.013979	0.529875	1.1685e-16
std	0.193931	0.104216	1.005038
min	-1.000000	0.000000	-5.110020
25%	0.000000	0.537387	0.072443
50%	0.000000	0.537387	0.072443
75%	0.000000	0.537387	0.072443
max	0.860855	1.000000	4.533794

The MaxAbsScaler rescales data by dividing each data point by the maximum absolute value found in the dataset.

The Min-MaxScaler rescales data by subtracting the minimum value observed in the dataset from each data point, and then dividing the resulting values by the range between the maximum and minimum values in the dataset.

The Standard Scaler standardizes data by subtracting the mean value of the dataset from each data point and then dividing the result by the dataset's standard deviation.

Impact on Data Structure:

When using MaxAbsScaler, the scaled dataset is constrained within the range of [-1, 1]. This scaling method maintains the linear relationship between data points, keeping their signs intact and preserving the original data distribution. Consequently, it doesn't mitigate the influence of outliers within the dataset.

In the case of MinMaxScaler, the scaled dataset is restricted to the range [0, 1]. This scaler simultaneously introduces an offset and compression to the data while retaining its inherent distribution. However, like MaxAbsScaler, MinMaxScaler is not ideal for handling outliers.

In standard scaling, the mean of the scaled dataset is adjusted to 0, and the standard deviation becomes 1. This results in the dataset being centered around zero. Standard Scaler is sensitive to outliers within the data.

Recommended Normalization Approach

In this scenario, no apparent outliers are present within the dataset. Consequently, our main goal is to rescale the data into a more manageable range to facilitate further processing. Taking these considerations into account, the MaxAbsScaler is opted to employ, which reconfigures the data within the interval of [-1, 1].

Moreover, this dataset exhibits a significant number of zero values, indicating sparsity. Preserving these zero values is beneficial for computations and helps reduce computational overhead at the hardware level. Notably, MaxAbsScaler is the sole scaler that conserves these zero values, rendering it the optimal choice in this particular context.

2. Linear Regression on Real World Data

Data from Advertising.csv

```python
import numpy as np
import pandas as pd

# Load data from CSV
file_path = r'./Advertising.csv'
df = pd . read_csv ( file_path )
print ( df . head () )
```

✓ 0.0s Python

```
   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
```

Split the data into training and testing sets with 80% of data points for training and 20% of data points for testing.

```python
X = df[['TV', 'radio', 'newspaper']]
y = df['sales']

from sklearn.model_selection import train_test_split

X_train , X_test , y_train , y_test = train_test_split ( X , y , test_size = 0.2 , random_state = 42 )
```

Test size 0.2 is corresponding to the 20% of data points are for the testing.

Training a linear regression model and estimating the coefficients

The linear regression model is in the form of $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$.

x_1, x_2 *and* x_3 are the features of advertising budgets of TV, radio, and newspaper in order. w_0 is the intercept and w_1, w_2, w_3 are coefficients of the model accordingly.

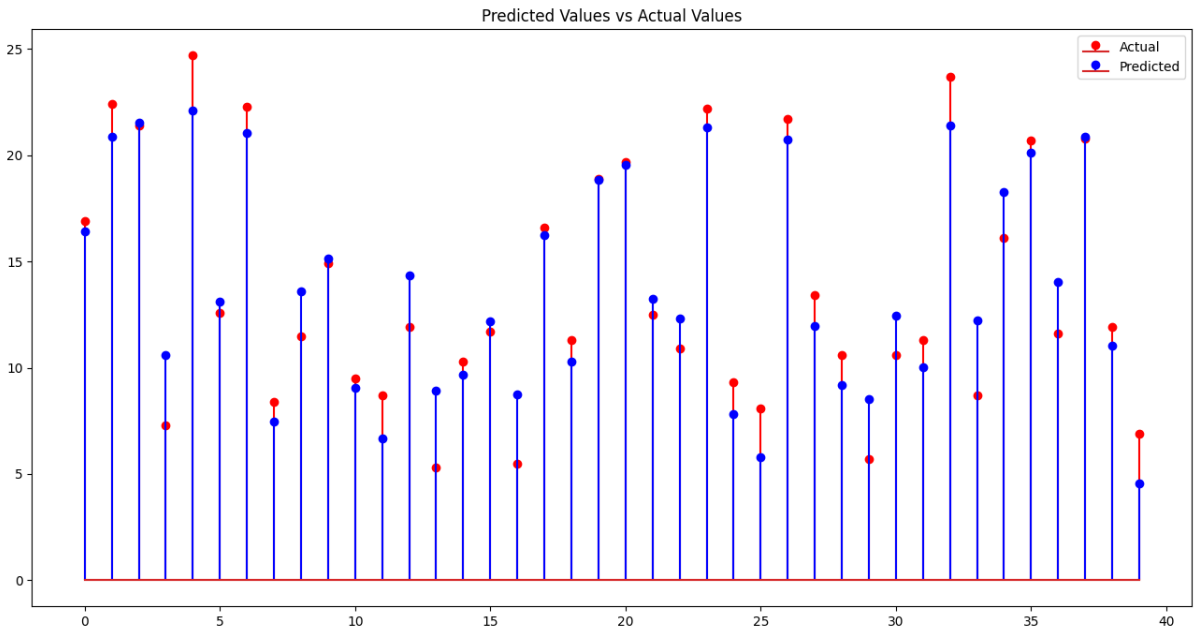
```
from sklearn.linear_model import LinearRegression

model = LinearRegression(fit_intercept=True).fit(X_train, y_train)
coefficients = model.coef_
intercept = model.intercept_
```

Coefficients:

$w_0 = 2.9791$ $(intercept)$

$w_1 = 0.0447,$ $w_2 = 0.1892,$ $w_3 = 0.0028$

Evaluation of train model on testing data



Statistics for testing and training data

Statistics	For Training Values		For Testing Values	
Residual sum of squares (RSS)	432.8207		126.9639	
Residual Standard Error (RSE)	1.6657		1.8780	
Mean Squared Error (MSE)	2.7051		3.1741	
R2 statistic	0.8957		0.8994	
Std. Error for each feature	w_0 w_1 w_2 w_3	0.3535 0.0016 0.0100 0.0070	w_0 w_1 w_2 w_3	0.7301 0.0033 0.0206 0.0117
t-statistic for each feature	w_o w_1 w_2 w_3	8.4270 28.5436 19.5180 0.3918	w_o w_1 w_2 w_3	4.0805 13.3825 9.1860 0.2361
p-value for each feature	w_0 w_1 w_2 w_3	0.0000 0.0000 0.0000 0.6958	w_0 w_1 w_2 w_3	0.0002 0.0000 0.0000 0.8147

Relationship between advertising budgets and sales

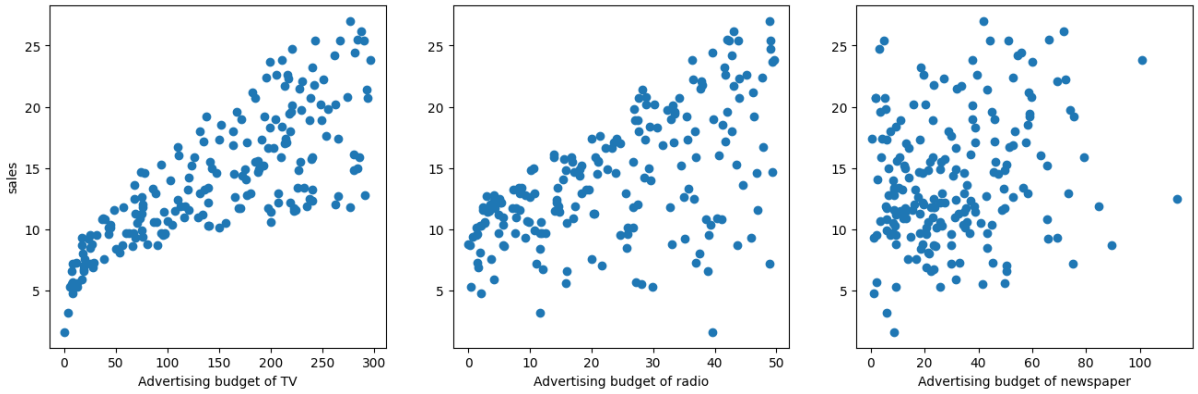


Figure: Advertising Budgets of TV, radio, and newspaper vs Sales

And when examining the p-values associated with the model parameters $w_1, w_2, and\ w_3$, which correspond to the advertising budgets of TV, radio, and newspaper, it becomes evident that $w_1\ and\ w_2$ exhibit remarkably low p-values. According to that we can signify statistical significance, both in the training and testing data. Consequently, we can deduce that a relationship exists between the advertising budget for TV and radio and the number of sales.

However, in the case of w_3, which represents the advertising budget for newspapers, the p-value is significantly high. That indicates that the relationship between sales and the newspaper advertising budget is very weak rather negligible.

The graphs presented above visually depict the mentioned relationships.

The independent variable that contributes highly on sales

Our linear model is $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3$.

Coefficients:

$w_0 = 2.9791$ $(intercept)$

$w_1 = 0.0447,$ $w_2 = 0.1892,$ $w_3 = 0.0028$

Based on the values of these coefficients in the linear model, the greatest influence on sales stems from the coefficient with the highest magnitude. In this instance, w_2 corresponds to the advertising budget for radio, indicating that it exerts the most significant impact on sales.

To illustrate, if we individually increase the budget by $1000 for each advertising channel, the expected sales increment would be as follows: for TV, an increase of 45 units in sales; for radio, a substantial increase of 189 units in sales; and for newspaper, a modest increase of only 3 units in sales.

Hence, it is evident that the independent variable that most significantly contributes to sales is the advertising budget allocated to radio.

Case Study

Our linear model is $y = 2.9791 + 0.0447x_1 + 0.1892x_2 + 0.0028x_3$.

Scenario 01:

Allocating $25,000 both television advertising and radio advertising individually

Predicting Sales = 5850 units

Scenario 02:

Investing $50,000 in television advertising solely

Predicting Sales = 2238 units

Scenario 03:

Investing $50,000 in radio advertising solely

Predicting Sales = 9463 units

Therefore Investing $50,000 in radio advertising solely yields higher sales compared to other cases.

3. Linear Regression impact on Outliers

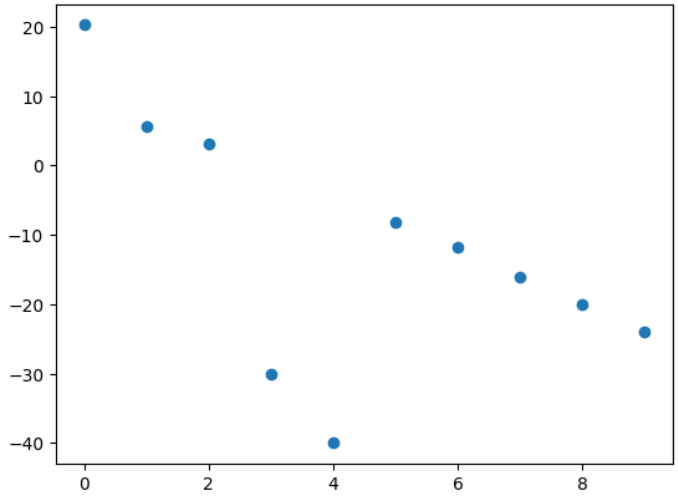


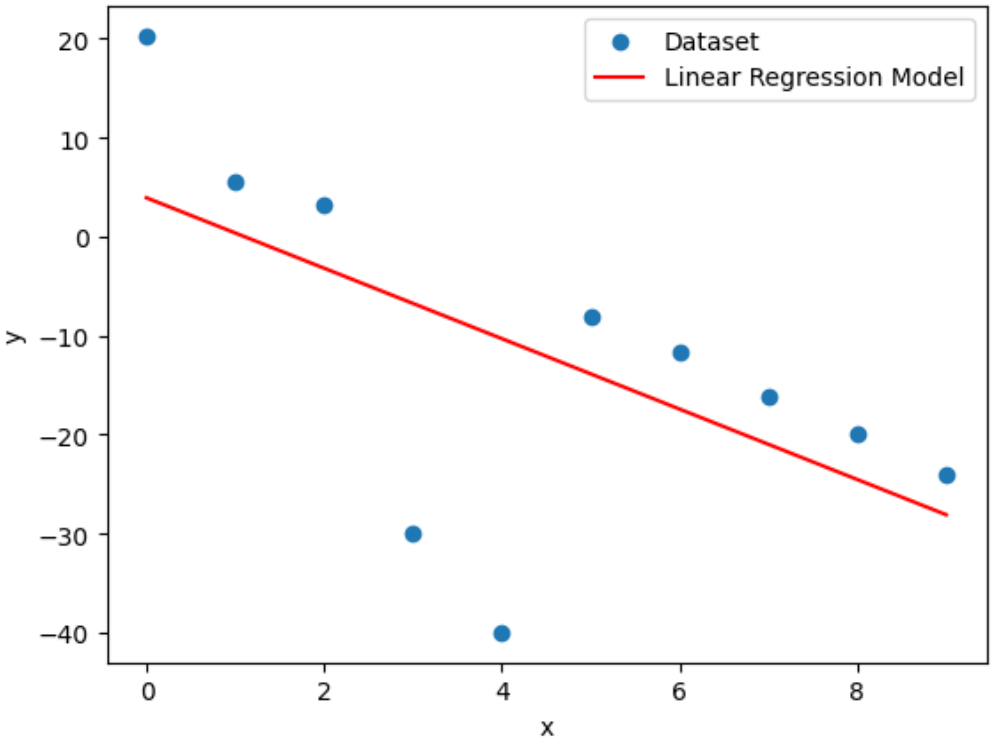
Figure: Graph plotted according to the given dataset

Linear Regression Model

We can suggest a linear regression model in the form of $y = w_0 + w_1 x$ for the given dataset after observing its strong linear correlation between the independent and dependent variables.

Here $w_0 = 3.9167 \; and \; w_1 = -3.5572$.

Therefore, the linear regression model is $y = 3.9167 - 3.5572\, x$



$Model\ 1:\ \ y\ =\ -4x + 12$

$Model\ 2:\ \ y\ =\ -3.55x + 3.91$

Loss Function

Loss Function $L(\theta, \beta) = \frac{1}{N} \sum_{i=1}^{N} \frac{(y_i - y_i')^2}{(y_i - y_i')^2 + \beta^2}$

Here, θ represents model parameters, β = 1 and number of data samples N = 10, respectively. Note the $y_i\ and\ y_i'$ are true and predicted i^{th} data sample, respectively.

```python
def LossFunction(w, Beta, x , y, N):
    L = 0
    for i in range(N):
        yi = y[i]
        y_i = w[0] + w[1]*x[i]
        RSS = (yi - y_i)**2
        L = L + (RSS/(RSS + Beta))
        L = L/N
    return L
```

0.0s Python

```python
w1 = np.array([12, -4])
w2 = np.array([3.91 , -3.55])

Loss1 = LossFunction(w1, 1, x, y, 10)
Loss2 = LossFunction(w2, 1, x, y, 10)
```

Loss Function value for model 1 = 0.00012928

Loss Function value for model 2 = 0.10474969

Most suitable model

When considering the above two models and the loss function introduced by the robust estimator, loss function value of model 1 is significantly lower than the loss function value of model 2. Therefore, the model 1 is the most suitable model for this data set.

How the robust estimator reduces the impact of the outliers

Typically, when we compute the loss value, we employ the Mean Square Error (MSE) metric to determine the model parameters that minimize this error. MSE involves squaring the difference between the actual and predicted values. However, in scenarios involving outliers, this difference can be substantially large, significantly inflating the MSE.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2$$

In contrast, the loss function introduced by a robust estimator incorporates a divisor that is also influenced by outliers. Consequently, this approach effectively diminishes the influence of outliers to a significant degree. Thus, the robust estimator markedly mitigates the impact of outliers on the overall context.

$$L(\theta, \beta) = \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i - y_i')^2}{(y_i - y_i')^2 + \beta^2}$$

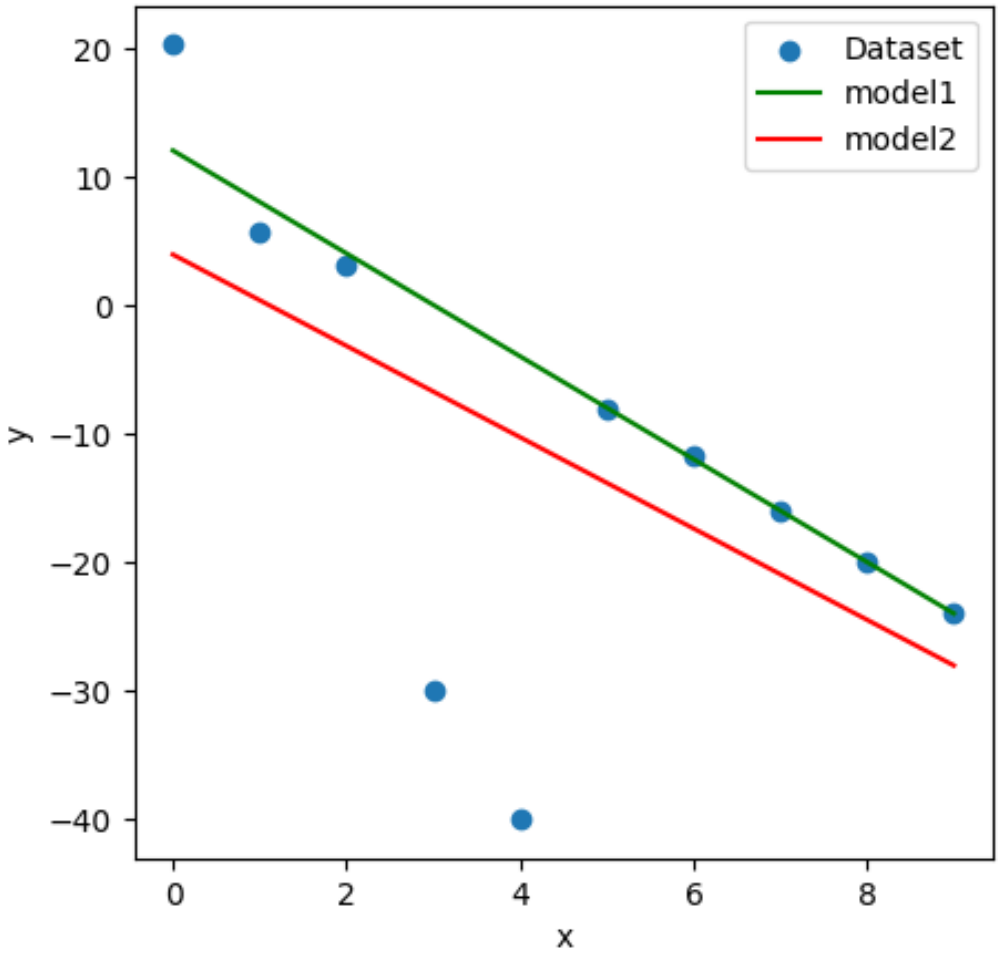


Figure: Visualization of the impact of the outliers

Impact of β

In the loss function introduced by the robust estimator, β is the factor that can be adjusted to lower the effects of outliers. The value of β is selected by observing the dataset and the situation. If the dataset has outliers or extreme values, a higher β value should be selected to increase the robustness of the loss function. If the dataset is free from outliers, a lower β value might be more appropriate.

When β is too high, the loss function approaches to the Mean Squared Error as $(y_i - y_i')^2 + \beta^2 \approx \beta^2$. Then the model fails to mitigate the impact of outliers.

When β is too small, the loss function's sensitivity towards the dataset decreases as $(y_i - y_i')^2 + \beta^2 \approx (y_i - y_i')^2$ and loss function value converges to 1.

Therefore, the β value should be appropriately selected based on the dataset.

4. GitHub Link for codes:

Pattern-Recognition/Linear Models for Regression at main · kavindukalinga/Pattern-Recognition (github.com)