Electronic and Telecommunication engineering

University of Moratuwa

EN3150 - Pattern Recognition

A01 Learning from data and related challenges and linear models for regression

RANAVIRAJA R.W.H.M.T.A. - Index No. 200520X

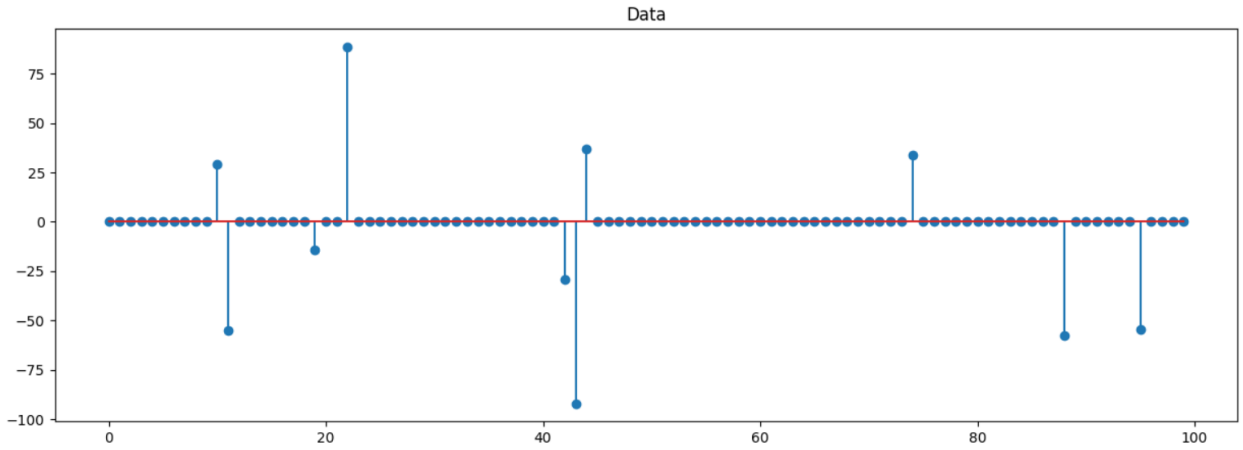
GitHub: GitHub

10th September 2023

1 Data preprocessing

1). Index number = 200520

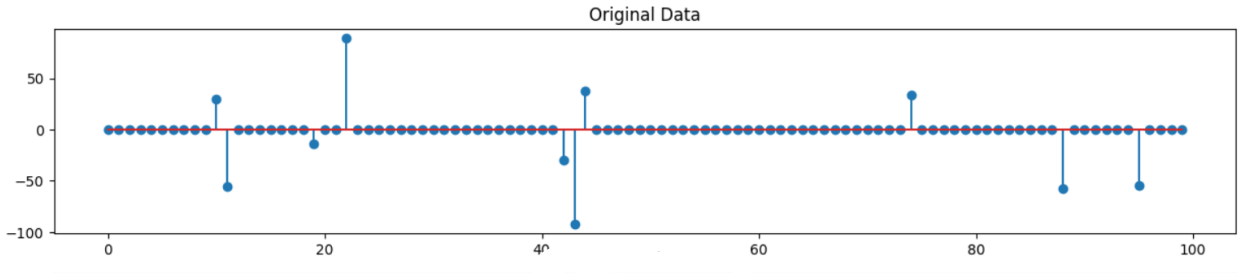
2).



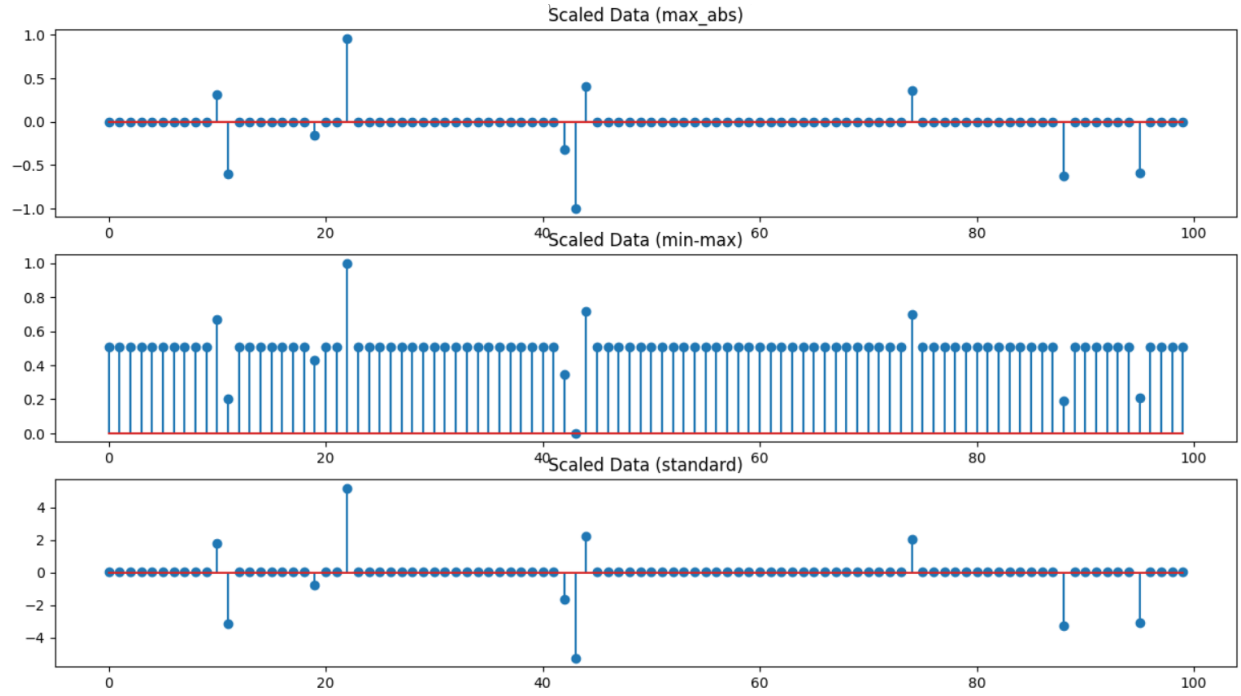
3).

```python
from sklearn.preprocessing import MaxAbsScaler

# Create a MaxAbsScaler object
max_abs_scaler = MaxAbsScaler()

# implement max-abs scaling
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

# Implement standard normalization (z-score)
def standard_scale(data):
    mean = np.mean(data)
    std_dev = np.std(data)
    scaled_data = (data - mean) / std_dev
    return scaled_data

# using  MaxAbsScaler
scaled_data_max_abs = max_abs_scaler.fit_transform(signal)
# using min_max_scale
scaled_data_min_max = min_max_scale(signal)
# using standard_scale
scaled_data_standard = standard_scale(signal)
```

4). Original data(data before normalization)



Normalized Data



5).

```
# print length of signal
print("Length of signal: ", len(signal))

# print number of non-zero elements in signal
print("Number of non-zero elements in signal: ", np.count_nonzero(signal))

# print number of non-zero elements in scaled_data_min_max
print("Number of non-zero elements in scaled_data_max_abs: ", np.count_nonzero(scaled_data_max_abs))

# print number of non-zero elements in scaled_data_min_max
print("Number of non-zero elements in scaled_data_min_max: ", np.count_nonzero(scaled_data_min_max))

# print number of non-zero elements in scaled_data_standard
print("Number of non-zero elements in scaled_data_standard: ", np.count_nonzero(scaled_data_standard))
```

```
Length of signal:  100
Number of non-zero elements in signal:  11
Number of non-zero elements in scaled_data_max_abs:  11
Number of non-zero elements in scaled_data_min_max:  99
Number of non-zero elements in scaled_data_standard:  100
```

6).

MaxAbsScaler changes the scale of the data while preserving its structure and distribution. It maps the data points to value between -1 to 1. This method is suitable for spares data sets, but this is not suitable for outliers.

Min-Max Scaling changes the structure and distribution. Because it scales the data set within the range [0, 1]. This destroys the sparsity of the data and also not suitable for outliers.

Standardization is centering the data around 0 and adjusting the spread (standard deviation) without changing the shape of the distribution. But this reduces the sparsity of the data set. This is more robust to outliers.

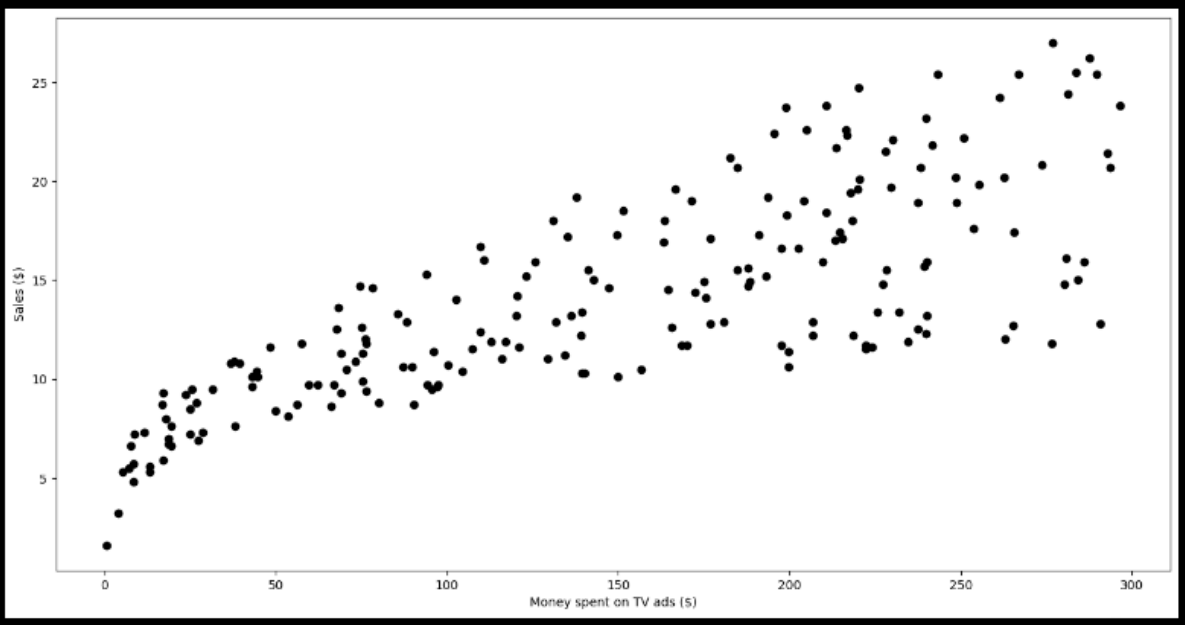
7).

According to the given data set, most suitable normalization technique is MaxAbsScaler method. Although it is not robust to outliers, it preserves its structure and distribution while scale down to value between -1 to 1.

2 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



2).

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

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

3).

```
model = LinearRegression()
model.fit(X_train, y_train)

# Coefficients
coefficients = model.coef_
intercept = model.intercept_
print("Coefficients (w1, w2, w3):", coefficients)
print("Intercept (w0):", intercept)
```

```
Coefficients (w1, w2, w3): [0.04472952 0.18919505 0.00276111]
Intercept (w0): 2.9790673381226256
```

4).

For Testing Data

```
from scipy import stats

# Step 4: Evaluate the model on testing data
y_pred_test = model.predict(X_test)

# Calculate RSS (Residual Sum of Squares)
RSS_test = np.sum((y_test - y_pred_test) ** 2)

# Calculate RSE (Residual Standard Error)
N = len(y_test)
d = X_test.shape[1]
RSE_test = np.sqrt(RSS_test / (N - d))

# Calculate MSE (Mean Squared Error)
MSE_test = mean_squared_error(y_test, y_pred_test)

# Calculate R2 statistic
R2_test = 1-RSS_test/np.sum((y_test - np.mean(y_test)) ** 2)

# Standard error of test coefficients
SE_TV_test = RSE_test/np.sqrt(np.sum((X_test['TV'] - np.mean(X_test['TV'])) ** 2))
SE_radio_test = RSE_test/np.sqrt(np.sum((X_test['radio'] - np.mean(X_test['radio'])) ** 2))
SE_newspaper_test = RSE_test/np.sqrt(np.sum((X_test['newspaper'] - np.mean(X_test['newspaper'])) ** 2))

# t-statistic and p-value for each feature
t_TV_test = (model.coef_[0])/SE_TV_test
t_radio_test = (model.coef_[1])/SE_radio_test
t_newspaper_test = (model.coef_[2])/SE_newspaper_test


p_TV_test = stats.t.sf(np.abs(t_TV_test), N-2)*2
p_radio_test = stats.t.sf(np.abs(t_radio_test), N-2)*2
p_newspaper_test = stats.t.sf(np.abs(t_newspaper_test), N-2)*2
```

```
Stastics for testing data:
RSS: 126.96389415904417
RSE: 1.8524191207426806
MSE: 3.174097353976104
R^2: 0.899438024100912

Std. Error for each feature:
TV        :  0.0032413433750177988
radio     :  0.0196438403742906
newspaper:  0.011004988002276353
t-statistic for each feature:
TV        :  13.799684974280366
radio     :  9.631266118512684
newspaper:  0.2508966244030504
p-value for each feature:
TV        :  2.2186557990034996e-16
radio     :  9.598125207694734e-12
newspaper:  0.8032457656245205
```

For Training Data

```python
# Evaluate the model on training data
y_pred_train = model.predict(X_train)

# Calculate RSS (Residual Sum of Squares)
RSS_train = np.sum((y_train - y_pred_train) ** 2)

# Calculate RSE (Residual Standard Error)
N = len(y_train)
d = X_train.shape[1]
RSE_train = np.sqrt(RSS_train / (N - d))

# Calculate MSE (Mean Squared Error)
MSE_train = mean_squared_error(y_train, y_pred_train)

# Calculate R2 statistic
R2_train = r2_score(y_train, y_pred_train)

# Standard error of train coefficients
SE_TV_train = RSE_train/np.sqrt(np.sum((X_train['TV'] - np.mean(X_train['TV'])) ** 2))
SE_radio_train = RSE_train/np.sqrt(np.sum((X_train['radio'] - np.mean(X_train['radio'])) ** 2))
SE_newspaper_train = RSE_train/np.sqrt(np.sum((X_train['newspaper'] - np.mean(X_train['newspaper'])) ** 2))

# t-statistic and p-value for each feature
t_TV_train = (model.coef_[0])/SE_TV_train
t_radio_train = (model.coef_[1])/SE_radio_train
t_newspaper_train = (model.coef_[2])/SE_newspaper_train

p_TV_train = stats.t.sf(np.abs(t_TV_train), N-2)*2
p_radio_train = stats.t.sf(np.abs(t_radio_train), N-2)*2
p_newspaper_train = stats.t.sf(np.abs(t_newspaper_train), N-2)*2
```

```
Stastics for training data:
RSS: 432.82070769302624
RSE: 1.660367367248314
MSE: 2.705129423081414
R^2: 0.8957008271017817

Std. Error for each feature:
TV        :  0.0015597901085700973
radio     :  0.008893871829610971
newspaper:  0.00647486168883072
t-statistic for each feature:
TV        :  28.67662592739552
radio     :  21.27251863518841
newspaper:  0.4264360343218125
p-value for each feature:
TV        :  1.6383428843357603e-64
radio     :  3.087650079938613e-48
newspaper:  0.6703705771810318
```

5).

Yes. There is a relationship between advertising budgets and sales. Based on the p-values for each feature, it appears that there is a significant relationship between advertising budgets for TV and radio and sales, but there is no significant relationship between the advertising budget for newspapers and sales.

6).

Radio

Based on the results of coefficients of linear regression model “Radio” has the highest value. Therefore, the independent variable, “Radio” contributes highly on sales. After that “TV” contributes on sales. But contribution of “Newspaper” is negligible.

7).

```
Predicted Sales for allocating 25 000 for both TV & Radio : 5851.093359915446
Predicted Sales for allocating 50 000 for TV              : 2239.4549407739387
Predicted Sales for allocating 50 000 for Radio           : 9462.731779056954
```

```
# Scenario 1
scenario_1 = np.array([[25000, 25000, 0]])  # TV budget, Radio budget, Newspaper budget
sales_scenario_1 = model.predict(scenario_1)

# Scenario 2
scenario_2 = np.array([[50000,0, 0]])  # TV budget, Radio budget, Newspaper budget
sales_scenario_2 = model.predict(scenario_2)

# Scenario 2
scenario_3 = np.array([[0,50000, 0]])  # TV budget, Radio budget, Newspaper budget
sales_scenario_3 = model.predict(scenario_3)

print("Predicted Sales for allocating 25 000 for both TV & Radio :", sales_scenario_1[0])
print("Predicted Sales for allocating 50 000 for TV              :", sales_scenario_2[0])
print("Predicted Sales for allocating 50 000 for Radio           :", sales_scenario_3[0])
```

Numerical figures confirm that allocating 50,000 dollars in radio can lead to increased sales. Because predicted sales for allocating 50,000 dollars for radio is greater than both allocating only for TV or allocating 25,000 dollars both television advertising and radio advertising individually.

Not only from that but also, we can say that allocating 50,000 for radio advertising is better than other methods because it has the highest coefficient for linear regression model.

3 Linear regression impact on outliers

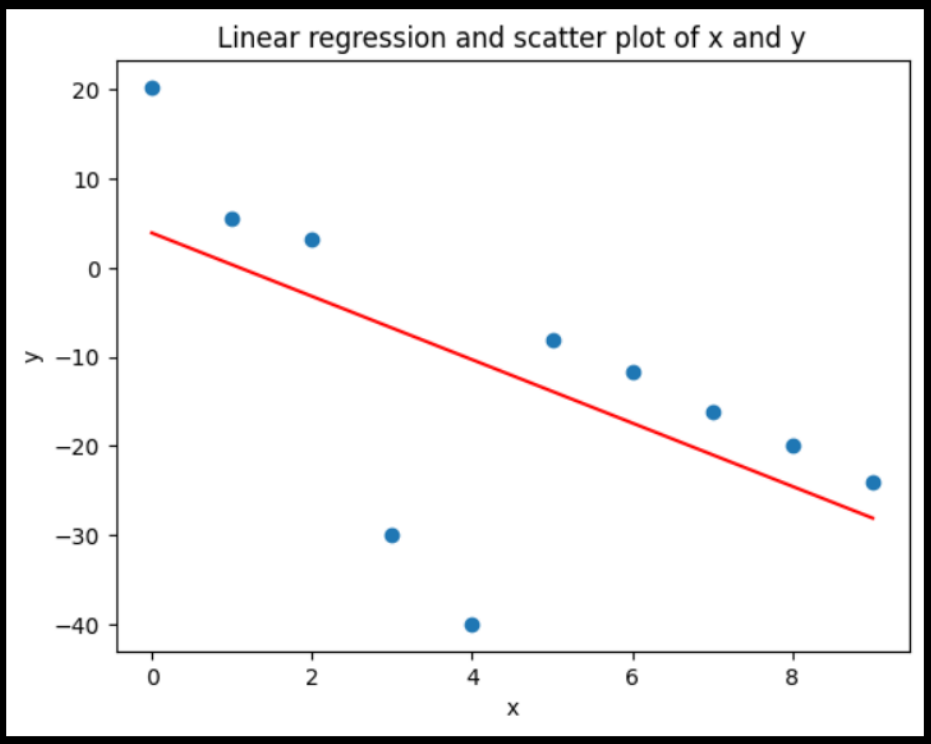
1).

```
xi=[0,1,2,3,4,5,6,7,8,9]
yi=[20.26,5.61,3.14,-30.00,-40.00,-8.13,-11.73,-16.08,-19.95,-24.03]
```

2).

```
# linear regression
m,b = np.polyfit(xi,yi,1)
print('m = ',m)
print('b = ',b)
```

```
m =  -3.5572727272727285
b =  3.9167272727272775
```



4).

```
N=10
B=1


def model1(x):
    return -4*np.array(x)+12

def model2(x):
    return -3.55*np.array(x)+3.91

def loss(y,y_hat):
    L=0
    for i in range(N):
        L+=((y[i]-y_hat[i])**2)/(((y[i]-y_hat[i])**2)+B**2)/N
    return L

y_hat1=model1(xi)
y_hat2=model2(xi)

L1=loss(yi,y_hat1)
L2=loss(yi,y_hat2)


print('Loss for model 1 = ',L1)
print('Loss for model 2 = ',L2)
```

```
Loss for model 1 =  0.435416262490386
Loss for model 2 =  0.9728470518681676
```

5).

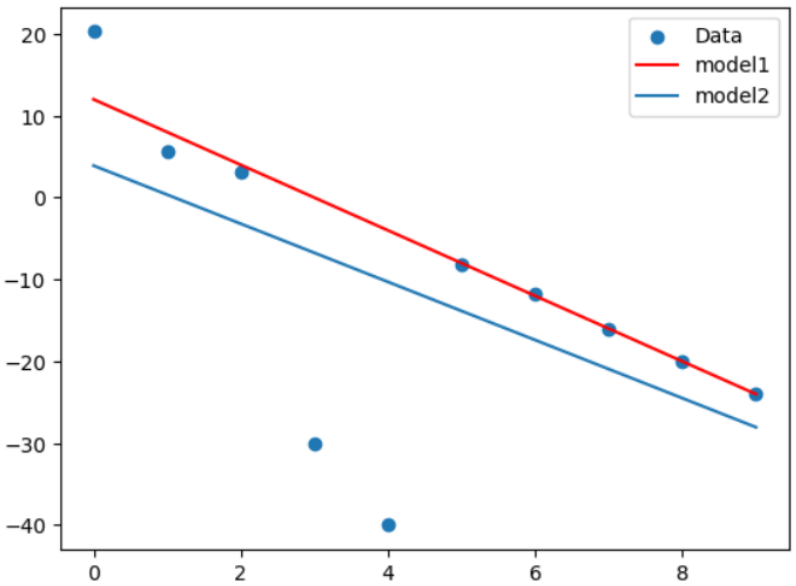
According to the values got for the loss function, the minimum value is for model 1. Therefore, model 1 is the most suitable model. (y=−4x+12)

6).

By decreasing the weight of the contribution of outliers to the loss function, the robust estimator reduces the effect of outliers. For that we have to select appropriate beta value for the loss function. For data points with large errors (outliers), the loss function will be larger.

7).

```
plt.scatter(xi,yi,label='Data')
plt.plot(xi,-4*np.array(xi)+12,c='r',label='model1')
plt.plot(xi,-3.55*np.array(xi)+3.91,label='model2')
plt.legend()
plt.show()
```



8). If β is decreased below the optimal range, the model becomes level to overtraining. In this scenario, it excessively adapts to expected values, leading to overfitting issues.

Conversely, if the value of β is increased beyond the optimal range, the optimization process tends to neglect extreme data points. Consequently, the model parameters converge to a specific value, but this value doesn't accurately predict the data points.

Therefore, according to the data points we have to select appropriate value for β.