Department of Electronic and Telecommunication Engineering
University of Moratuwa

EN 3150: Pattern Recognition

Assignment 01: Learning from data and related challenges and linear models for regression.

Clarance L.G.S. - 200094R

September 11, 2023

1. Data Preprocessing

```
import numpy as np
import matplotlib.pyplot as plt
# Data Generation
def generate_signal(signal_length, num_nonzero):

    signal = np.zeros(signal_length)
    nonzero_indices = np.random.choice(signal_length, num_nonzero, replace=False)
    nonzero_values = 50*np.random.randn(num_nonzero)
    signal[nonzero_indices] = nonzero_values
    return signal

signal_length = 100  # Total length of the signal
num_nonzero = 10    # Number of non-zero elements in the signal
your_index_no= 200094 # Enter without english letter and without leading zeros
signal = generate_signal(signal_length, num_nonzero)
signal[10] = (your_index_no % 10)*10 + 10
if  your_index_no % 10 == 0:
  signal[10] = np.random.randn(1) + 30

# Plotting the Signal
signal=signal.reshape(signal_length,1)
plt.figure(figsize=(15,5))
plt.subplot(1, 1, 1)
plt.title("Data")
plt.stem(signal)
```

- This code generates a synthetic signal with non-zero values (11) at random positions, modifies a specific element in the signal based on **your_index_no**, and then plots the signal for visualization. The modifications to **signal[10]** depend on the last digit of **your_index_no**.

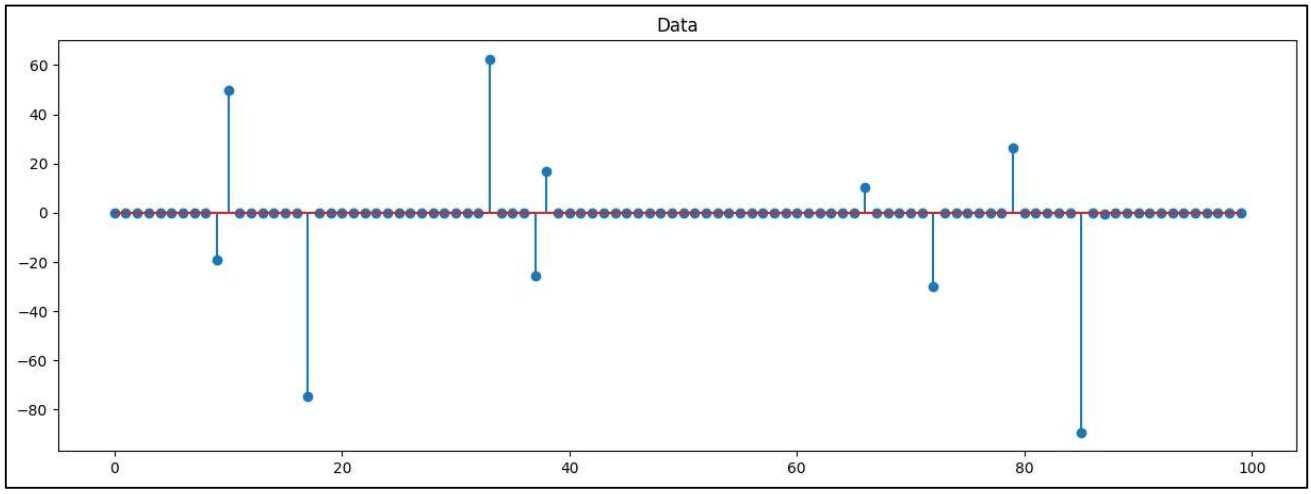

Figure 1: Generated Signal

```
# MaxAbsScaler
from sklearn.preprocessing import MaxAbsScaler
scaler = MaxAbsScaler()
scaled_data_max_abs = scaler.fit_transform(signal)
```

- Here Max-Abs normalization is applied using **preprocessing.MaxAbsScaler()** from **sklearn.preprocessing.**

```
# MinMaxScaler
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)
```

- Here Min-Max normalization is implemented by using user defined function. As the Min-Max scaling equation is

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
# StandardScaler
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)
```

- Here Standard-Scalar normalization is implemented by using user defined function. As the Standardization scaling equation is

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

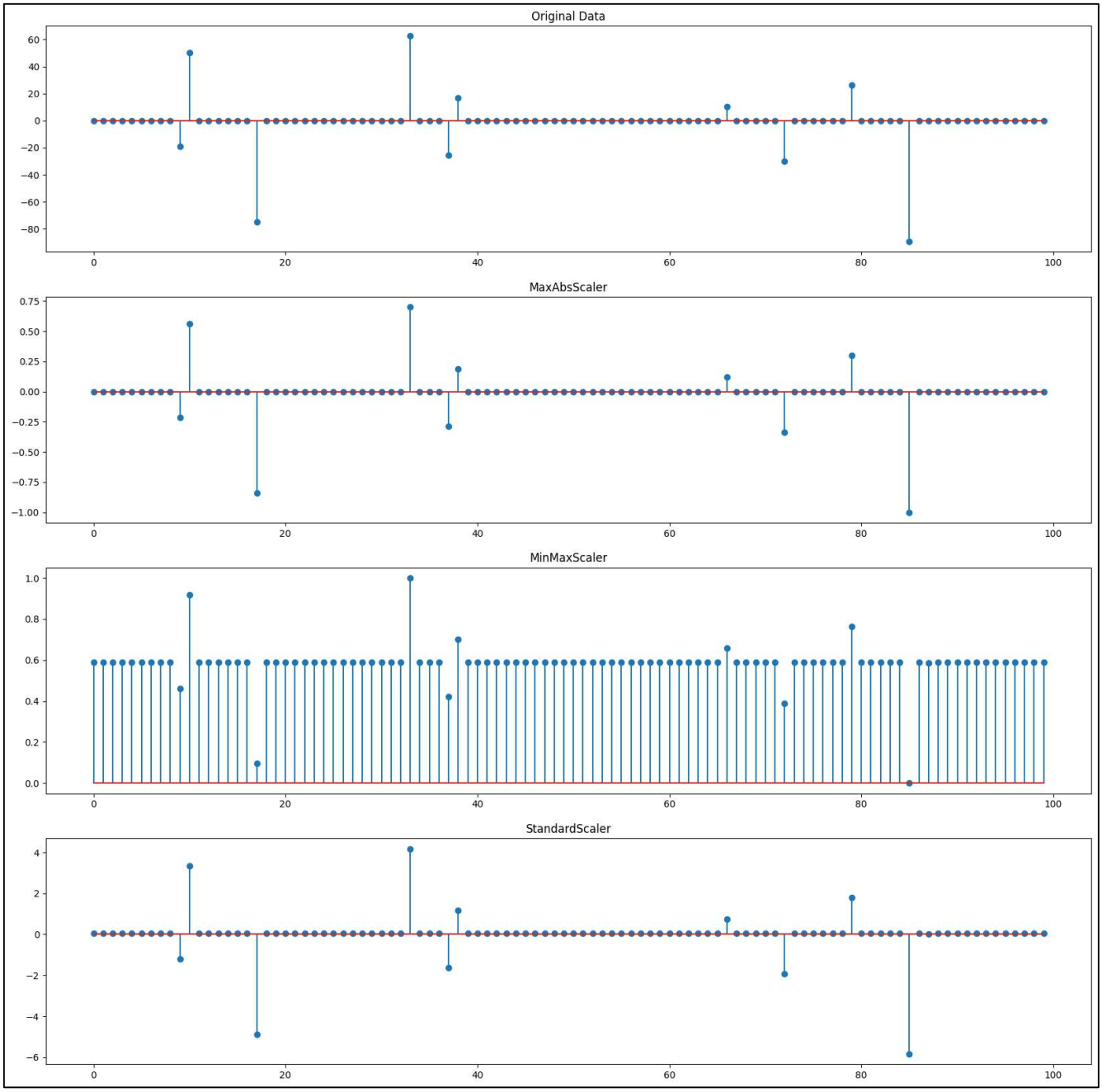

Figure 2: Stem plots of original and normalized data

- Max-Abs Scaler
  - It scales features by dividing each value by the maximum absolute value within that feature.
  - It ensures that all feature values lie within the range of -1 to 1.
  - It will not affect the number of non-zero terms (11) in the signal.
  - The structure/shape of the signal is not affected.
- Min-Max Scalar
  - It transforms features to a specified range, usually between 0 and 1, by subtracting the minimum value and dividing it by the range (max - min).
  - It affected the number of non-zero terms (99) in signal.
  - The structure of the signal is also affected.

- Standard Scalar
  - It standardizes features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing it by the standard deviation.
  - It will not affect the number of non-zero terms (11) in the signal.
  - The structure/shape of the signal is not affected.

- The generated signal is a sparse signal. So, it would be appropriate if we used Max-Abs scalar for normalization.
- We can retain the original scale of the data and the relationships between data points so Max-Abs scaling could be a good choice.

Linear Regression on real world data

```
import numpy as np
import pandas as pd
# Loading data from CSV
file_path = r'E:\Pattern Recognition\EN3150-Pattern-Recognition\Assignment 1\Supporting
Materials\Advertising.csv '
df = pd. read_csv ( file_path )
print (df. head ())
```

- The code is loading the data form CSV.
- This data set has 200 samples and 4 features. The features are 'TV', 'Radio', 'Newspaper' and 'Sales'. The 'Sales' is the target variable.

```
# Splitting the data into train and test
from sklearn.model_selection import train_test_split
# Independent variables
X = df[['TV', 'radio', 'newspaper']]
# Dependent variable
y = df['sales']
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
```

- The data set is split into 80% training set and 20% testing set.
- **train_test_split** function in the **sklearn.model_selection** library is used to split the data.

```
# Training a Linear Regression Model
from sklearn.linear_model import LinearRegression
# Creating a Linear Regression object
model = LinearRegression()
# Training the model
model.fit(X_train, y_train)
```

- **LinearRegression** function in the **sklearn.linear_model** library is used to train a linear regression model.

- Residual Sum of Squares (RSS): **88.12**
  - It is the sum of the squared differences between the observed (actual) values and the predicted values in a regression model.

- Residual Standard Error (RSE): **1.54**
  - An estimate of the standard deviation of the residuals in a regression model. It provides a measure of how much the actual responses deviate from the predicted responses.

- Mean Squared Error (MSE): **2.20**
  - A common metric used to measure the average squared difference between observed and predicted values. It's calculated by taking the mean of the squared residuals. A lower MSE indicates a better-fitting model.

- R2 statistic: **0.93**
  - A measure of the proportion of the variance in the dependent variable that is explained by the independent variables in a regression model. It ranges from 0 to 1, with higher values indicating a better fit. It is often interpreted as the goodness-of-fit of the model.

- Standard Errors for each feature: **TV 0.001445 Radio 0.010720 Newspaper 0.007592**
  - The standard error for each feature in a regression model quantifies the precision of the estimated coefficients for that feature. It indicates how much the coefficient estimate is expected to vary from the true population value.

- t-statistic for each feature: **TV 37.319943 Radio 20.537269 Newspaper 1.990823**
  - The t-statistic measures the significance of each feature's coefficient in a regression model. It is calculated by dividing the estimated coefficient by its standard error. Higher absolute t-values indicate greater significance.

- p-value for each feature: **TV 5.863407e-80 Radio 2.465181e-46 Newspaper 4.823571e-02**
  - The p-value associated with each feature's coefficient in a regression model assesses the null hypothesis that the coefficient is equal to zero (i.e., the feature has no effect). A low p-value indicates that the feature is statistically significant in predicting the outcome.

- Code snippet for calculating the above statistics.

```
# Making predictions on the test set
y_pred = model.predict(X_test)
# Calculating the residuals
residuals = y_test - y_pred
## Calculating sum of squares (RSS) ##
rss = np.sum(np.square(residuals))
# Total number of samples in the test set
N = len(y_test)
# Number of features
d = X_test.shape[1]
## Residual Standard Error (RSE) ##
rse = np.sqrt(rss / (N - d))
## Mean Squared Error (MSE) ##
```

```
mse = mean_squared_error(y_test, y_pred)
## R2 statistic ##
r2 = r2_score(y_test, y_pred)
# OLS model
ols_model = sm.OLS(y_train, X_train).fit()
## Standard Errors for each feature ##
se_b = ols_model.bse
## t-statistic for each feature ##
t_stat = ols_model.tvalues
## p-value for each feature ##
p_values = ols_model.pvalues
```

- Yes, there is a relationship between advertising budgets and sales, as indicated by the linear regression model's performance metrics:
  - R2 statistic is 0.93, which means that approximately 93% of the variance in sales can be explained by the advertising budgets. This indicates a strong relationship.

- To determine which independent variable contributes most to sales, need to look at the t-statistic and p-value for each feature:
  - TV: The t-statistic is 37.32, and the p-value is very close to zero (5.86e-80). This indicates that the budget allocated to TV advertising has a highly significant and positive contribution to sales.
  - Radio: The t-statistic is 20.54, and the p-value is also very close to zero (2.47e-46). This indicates that the budget allocated to radio advertising also has a highly significant and positive contribution to sales.
  - Newspaper: The t-statistic is 1.99, and the p-value is 0.0482. While the newspaper budget still has a positive t-statistic, it has a higher p-value, suggesting that its contribution to sales is less significant compared to TV and radio.

  - **Therefore, both TV and radio advertising budgets contribute significantly to sales, with TV having a stronger impact.**

- For the Strategy of Allocating $25,000 to Both TV and Radio Advertising:
  - This strategy leverages the strengths of both TV and radio advertising, which have been shown to have significant and positive contributions to sales according to the model.
- For the Strategy of Allocating $50,000 to Either TV or Radio Advertising Alone:
  - Based on the model, both TV and radio advertising individually contribute significantly to sales, so allocating the entire budget to either channel should yield positive results.

  - **But the combined budget of $50,000 may allow for more comprehensive advertising campaigns on both channels, potentially reaching a broader audience.**
  - **This strategy might lead to a synergistic effect, where the combined impact of TV and radio advertising is greater than the sum of their individual impacts.**

Linear Regression impact on outliers

```
# Data
xi = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
yi = np.array([20.26, 5.61, 3.14, -30.00, -40.00, -8.13, -11.73, -16.08, -19.95, -24.03])
# Calculating the Coefficients of the Linear Regression Model
slope, intercept = np.polyfit(xi, yi, 1)
print("Slope(m): ", slope)
print("Intercept(b): ", intercept)
# Creating the Linear Regression Model
regression_line = slope * xi + intercept
```

- Linear regression model is created for the given data.

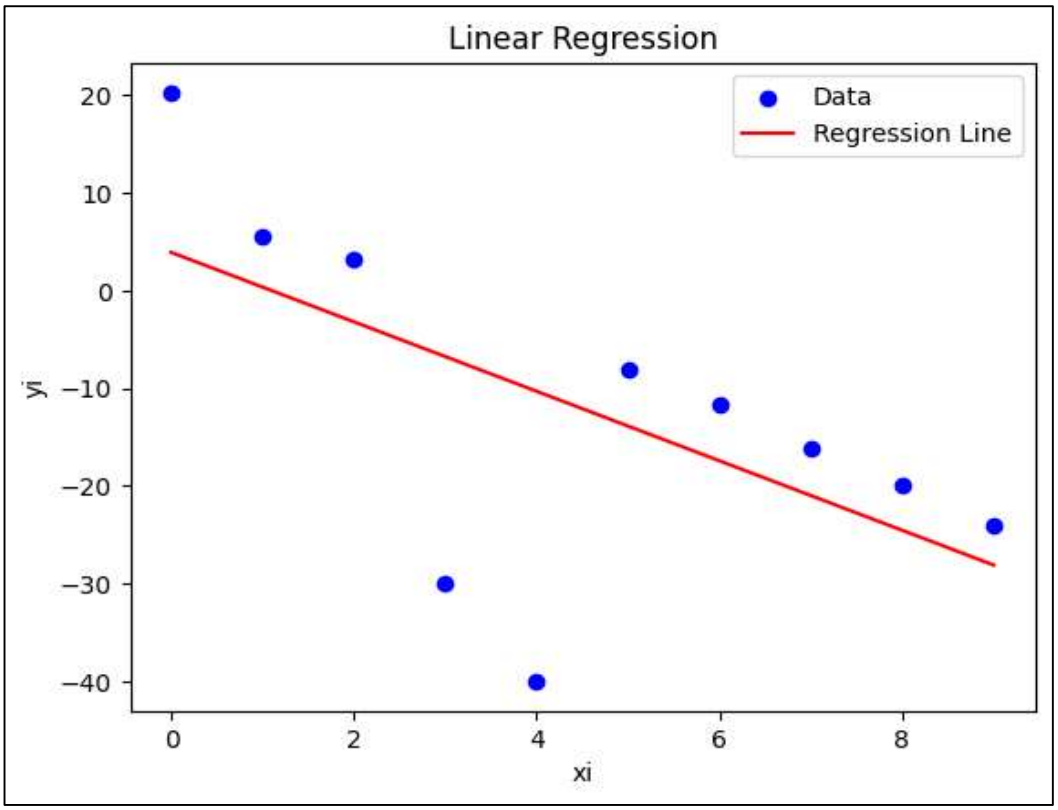

Figure 3: Scatter Plot of Data and Linear regression model

- Slope(m): -3.5572727272727285
- Intercept(b): 3.9167272727272775

- A robust estimator is introduced to reduce the impact of the outliers. The robust estimator finds model parameters which minimize the following loss function.

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

```
# Loss Function Values
# Predicted data points for Model 1
yi_hat_model1 = -4 * xi + 12
# Predicted data points for Model 2
yi_hat_model2 = -3.55 * xi + 3.91
beta = 1.0
N = 10
L_model1 = (1 / N) * np.sum(np.power(yi - yi_hat_model1, 2) / (np.power(yi - yi_hat_model1, 2) +
beta**2))
L_model2 = (1 / N) * np.sum(np.power(yi - yi_hat_model2, 2) / (np.power(yi - yi_hat_model2, 2) +
beta**2))
```

- Loss function L(θ,β) value for Model 1(y = −4x+12): **0.435416262490386**
- Loss function L(θ,β) value for Model 2(y = -3.55x + 3.91): **0.9728470518681676**

- To determine the most suitable model for the provided dataset using the robust estimator, need to select the model with the lower loss function (L) value. Lower L values indicate a better fit to the data. In this case,
  - **Model 1 has a lower loss function value (0.4354) compared to Model 2 (0.9728). Therefore, Model 1 (y = −4x+12) is the most suitable model for the provided dataset according to the robust estimator.**

- Robust estimator uses robust loss functions that penalize outliers less severely than traditional loss functions. With a given loss function robust estimator aims to reduce the impact of outliers by assigning different weights to the data points based on the residuals and a tuning parameter β.

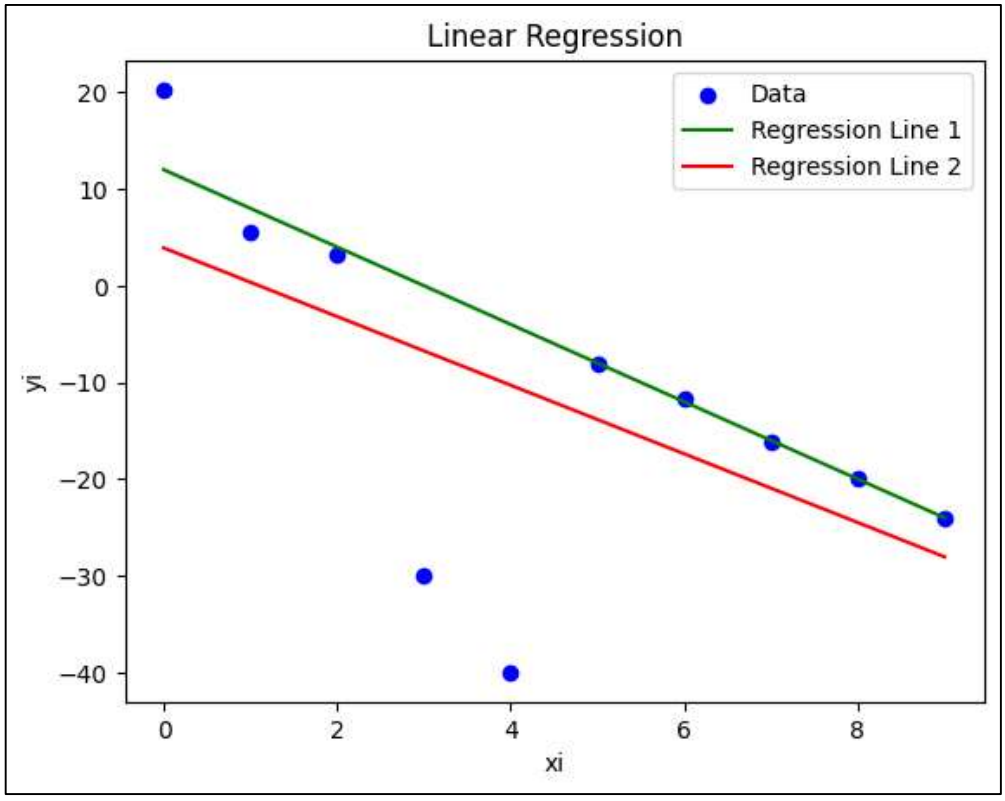

Figure 4: Scatter Plot of Data and Linear regression models

- In the context of reducing the impact of outliers, the parameter β plays a critical role. As β increases, the robust estimator becomes less sensitive to outliers.
    - Larger β values effectively down weight outliers in the loss function, making the model more robust to extreme data points.
    - Small β (close to 0) emphasizes the squared error term, and outliers have a significant impact on the loss.

    - Large β reduces the impact of the squared error term, making the model less sensitive to outliers.
    - The choice of the optimal β value depends on the specific characteristics of the data and the desired level of robustness. A larger β provides more robustness to outliers but may slightly reduce the model's fit to the majority of data points.

References

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- https://scikit-learn.org/stable/modules/preprocessing.html

GitHub Repository

- https://github.com/ClaranceLGS/EN3150-Pattern-Recognition/tree/main/Assignment%201