In [4]:

অধ্যায় ৫ঃ ডেটা প্রিপ্রসেসিং

Author: MD. Emdadul Hoque Tareque (Lead, Phitron)

Provide by: **Phitron AI/ML**

Community Link: **Phitron AI/ML**

মেশিন লার্নিং অ্যালগরিদম গুলো ডাটা থেকে শিখে। তাই ডাটাগুলোকে এমনভাবে প্রিপেসেস করতে হয় যেন মেশিন লার্নিং এলগরিদম ডেটাগুলো থেকে প্যাটার্ন বের করতে পারে। এই অধ্যায়ে আমরা জানব কিভাবে মেশিন লার্নিং অ্যালগরিদম জন্য ডাটা প্রস্তুত করতে হয়।

এই অধ্যায়ে আমরা যা যা শিখব

1. Rescale data/Normalize data
2. Standardize data
3. Binarize data

আমি প্রতিটি ডেটা প্রিপ্রসেসিং টেকনিক (Rescale/Normalize, Standardize, Binarize) সম্পর্কে বিস্তারিত ব্যাখ্যা দেব এবং Python-এর scikit-learn লাইব্রেরি ব্যবহার করে কোডের উদাহরণ সহ বোঝাব। প্রতিটি টেকনিক কেন গুরুত্বপূর্ণ এবং কীভাবে মেশিন লার্নিং মডেলের জন্য ডেটা প্রস্তুত করতে সাহায্য করে তাও ব্যাখ্যা করব।

৫.১ Rescale Data / Normalize Data

ব্যাখ্যা:

Rescale বা Normalize করার উদ্দেশ্য হল ডেটাকে একটি নির্দিষ্ট রেঞ্জে (সাধারণত [0, 1] বা [-1, 1]) নিয়ে আসা। এটি বিশেষভাবে গুরুত্বপূর্ণ যখন ডেটার ফিচারগুলোর স্কেল ভিন্ন ভিন্ন হয়। উদাহরণস্বরূপ, যদি একটি ফিচারে মান 0-1000 এবং অন্যটিতে 0-10 হয়, তাহলে মেশিন লার্নিং অ্যালগরিদম (যেমন Gradient Descent ভিত্তিক মডেল) বড় স্কেলের ফিচার দ্বারা বেশি প্রভাবিত হতে পারে। Normalization এই সমস্যা সমাধান করে।

পদ্ধতি:

- **Min-Max Scaling**: ডেটাকে একটি নির্দিষ্ট রেঞ্জে রূপান্তর করে। সূত্র:

  $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$ এটি ডেটাকে [0, 1] রেঞ্জে নিয়ে আসে।

- Normalization সাধারণত Neural Networks, SVM, বা KNN-এর মতো অ্যালগরিদমে কার্যকর।

কোড উদাহরণ:

In [ ]:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Sample Data
data = np.array([[100], [50], [30], [20], [10]])

# MinMaxScaler object
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
# ডData Rescale
rescaled_data = scaler.fit_transform(data)

print("Main Data:\n", data)
print("Rescale Data (0-1 Range):\n", rescaled_data)
```

```
Main Data:
 [[100]
 [ 50]
 [ 30]
 [ 20]
 [ 10]]
Rescale Data (0-1 Range):
 [[1.        ]
 [0.44444444]
 [0.22222222]
 [0.11111111]
 [0.        ]]
```

ব্যাখ্যা:

- মূল ডেটার মানগুলো 10 থেকে 100 পর্যন্ত ছিল। MinMaxScaler ব্যবহার করে সেগুলো [0, 1] রেঞ্জে রূপান্তরিত হয়েছে।
- এটি মেশিন লার্নিং মডেলের জন্য ফিচারগুলোর স্কেলকে সমান করে।

৫.২ Standardize Data

ব্যাখ্যা:

Standardization হল ডেটাকে এমনভাবে রূপান্তর করা যেন এর মিন (mean) 0 এবং স্ট্যান্ডার্ড ডেভিয়েশন (standard deviation) 1 হয়। এটি ডেটাকে একটি স্ট্যান্ডার্ড নরমাল ডিস্ট্রিবিউশন (Gaussian distribution) অনুসারে রূপান্তর করে। এটি সাধারণত তখন ব্যবহৃত হয় যখন ডেটা গাউসিয়ান বা প্রায় গাউসিয়ান ডিস্ট্রিবিউশন অনুসরণ করে।

সূত্র:

$X_{std} = \frac{X-\mu}{\sigma}$ যেখানে (\mu) হল মিন এবং (\sigma) হল স্ট্যান্ডার্ড ডেভিয়েশন।

কখন ব্যবহার করবেন:

- Linear Regression, Logistic Regression, PCA, বা SVM-এর মতো অ্যালগরিদমে Standardization কার্যকর।
- যদি ডেটার স্কেল ভিন্ন হয় এবং গাউসিয়ান ডিস্ট্রিবিউশন অনুসরণ করে, তবে এটি উপযোগী।

In [ ]:
```
import numpy as np
from sklearn.preprocessing import StandardScaler

# Sample data
data = np.array([[100], [50], [30], [20], [10]])

# Create StandardScaler object
scaler = StandardScaler()

# Standardize the data
standardized_data = scaler.fit_transform(data)

print("Original Data:\n", data)
print("Standardized Data (mean=0, standard deviation=1):\n", standardized_data)
print("Mean of Standardized Data:", np.mean(standardized_data))
print("Standard Deviation of Standardized Data:", np.std(standardized_data))
```

```
Original Data:
 [[100]
 [ 50]
 [ 30]
 [ 20]
 [ 10]]
Standardized Data (mean=0, standard deviation=1):
 [[ 1.81962183]
 [ 0.25098232]
 [-0.37647348]
 [-0.69020139]
 [-1.00392929]]
Mean of Standardized Data: 0.0
Standard Deviation of Standardized Data: 1.0
```

ব্যাখ্যা:

- ডেটার মিন 0 এবং স্ট্যান্ডার্ড ডেভিয়েশন 1 এ রূপান্তরিত হয়েছে।
- এটি মেশিন লার্নিং অ্যালগরিদমের জন্য ডেটাকে আরও সামঞ্জস্যপূর্ণ করে তোলে।

৫.৩ Binarize Data

ব্যাখ্যা:

Binarization হল ডেটাকে বাইনারি মানে (0 বা 1) রূপান্তর করা। এটি তখন ব্যবহৃত হয় যখন আমরা ডেটাকে একটি নির্দিষ্ট থ্রেশহোল্ডের উপর ভিত্তি করে দুটি শ্রেণীতে ভাগ করতে চাই। উদাহরণস্বরূপ, যদি থ্রেশহোল্ড 50 হয়, তবে 50-এর বেশি মান 1 এবং 50-এর কম মান 0 হবে।

কখন ব্যবহার করবেন:

- যখন ফিচারগুলোকে বাইনারি ফর্মে প্রকাশ করতে হয়, যেমন Decision Trees বা Naive Bayes-এর ক্ষেত্রে।
- ইমেজ প্রসেসিং বা টেক্সট প্রসেসিংয়ে প্রায়ই ব্যবহৃত হয়।

সূত্র:

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

In [ ]:
```python
import numpy as np
from sklearn.preprocessing import Binarizer

# Sample data
data = np.array([[100], [50], [30], [20], [10]])

# Create Binarizer object (threshold = 50)
binarizer = Binarizer(threshold=50)

# Binarize the data
binarized_data = binarizer.fit_transform(data)

print("Original Data:\n", data)
print("Binarized Data (threshold=50):\n", binarized_data)
```

```
Original Data:
 [[100]
 [ 50]
 [ 30]
 [ 20]
 [ 10]]
Binarized Data (threshold=50):
 [[1]
 [0]
 [0]
 [0]
 [0]]
```

ব্যাখ্যা:

- থ্রেশহোল্ড 50-এর উপরে শুধুমাত্র 100 আছে, তাই এটি 1 হয়েছে। বাকি সব 0।
- এটি ডেটাকে সিম্পলিফাই করে এবং বাইনারি ক্লাসিফিকেশনের জন্য উপযোগী করে।

কখন কোনটি ব্যবহার করবেন?

- **Rescale/Normalize**: যখন ফিচারগুলোর স্কেল ভিন্ন এবং মডেল স্কেল-সংবেদনশীল (যেমন Neural Networks, KNN)।
- **Standardize**: যখন ডেটা গাউসিয়ান ডিস্ট্রিবিউশন অনুসরণ করে এবং মডেল মিন ও স্ট্যান্ডার্ড ডেভিয়েশনের উপর নির্ভর করে (যেমন Linear Regression, PCA)।
- **Binarize**: যখন ডেটাকে বাইনারি ফর্মে রূপান্তর করতে হয়, বিশেষ করে ক্লাসিফিকেশন বা ইমেজ/টেক্সট প্রসেসিংয়ে।

অতিরিক্ত টিপস:

- **scikit-learn** এর প্রিপ্রসেসিং মডিউল ব্যবহার করা সহজ এবং দ্রুত। তবে ট্রেনিং এবং টেস্ট ডেটার জন্য একই স্কেলার ব্যবহার করতে হবে। উদাহরণস্বরূপ, `scaler.fit()` শুধু ট্রেনিং ডেটার উপর করবেন এবং `scaler.transform()` টেস্ট ডেটার উপর।
- ডেটা প্রিপ্রসেসিংয়ের আগে ডেটার ডিস্ট্রিবিউশন (histogram, boxplot) দেখে নেওয়া ভালো।

৫.৪ এই অধ্যায়ে যা যা শিখলাম

এই অধ্যায়ে আমরা মেশিন লার্নিং এলগরিদমের জন্য ডেটাকে কয়েকটা টেকনিকে তৈরি করা দেখলাম। তবে আরো কিছু টেকনিক ব্যবহার করে ডেটাকে প্রিপ্রসেস করা যায় তা আমরা সামনে শিখব । এই অধ্যায় যে যে টেকনিকগুলো শিখেছি --

1. Rescale data/Normalize data
2. Standardize data
3. Binarize data

এছাড়াও আরো কিছু বেশ কিছু ডেটা প্রিপ্রসেসিং টেকনিক রয়েছে। যা আমরা পরের অধ্যায় শিখব।

In [3]: