آماده سازی داده (preparing data):

مقدمه:

الگوریتم های یادگیری ماشین کاملا وابسته به داده هستند، زیرا داده، حیاتی ترین جنبه ای است که آموزش مدل را ممکن می سازد. از طرف دیگر، اگر پیش از ارسال داده به الگوریتم های یادگیری ماشین، قادر به درک مفهوم آن نباشیم، ماشین بی مصرف خواهد بود. به بیان ساده تر، ما باید همیشه داده درست را برای مساله ای که می خواهیم ماشین آن را حل کند، به ماشین وارد کنیم، مثلا، داده در مقیاس و قالب صحیح که شامل ویژگی های معنا دار باشد.

این مورد، آماده سازی داده را به مهمترین مرحله در پردازش یادگیری ماشین تبدیل کرده است. ممکن است آماده سازی داده به عنوان روندی که مجموعه داده ما را برای پردازش یادگیری ماشین مناسب تر میکند، تعریف شود.

چرا پیش پردازش(pre-processing) داده؟

پس از انتخاب داده خام برای آموزش یادگیری ماشین، مهمترین کار پیش پردازش داده است. به معنای گسترده، پیش پردازش داده، داده انتخاب شده را به قالبی که بتوانیم با آن کار کنیم یا بتوانیم به الگوریتم های ML بدهیم، تبدیل خواهد کرد. همیشه نیاز داریم تا عمل پیش پردازش را روی داده خود انجام دهیم، تا بتواند مطابق انتظار الگوریتم های یادگیری ماشین باشد.

روش های پیش پردازش داده:

روش های پیش پردازش داده که در ادامه معرفی می شوند، جهت تولید داده برای الگوریتم های ML ، می توانند روی مجموعه داده اعمال شوند.

مقیاس بندی(scaling) :

احتمالا مجموعه داده ما از ویژگی هایی با مقیاس متغیر تشکیل شده است، اما ما نمی توانیم چنین داده ای را به الگوریتم ML دهیم، زیرا به مقیاس بندی مجدد نیاز خواهد داشت. مقیاس بندی مجدد داده، این اطمینان را می دهد که ویژگی ها در مقیاس مشابه قرار دارند. به طور کلی، ویژگی ها در بازه بین صفر و یک مجددا مقیاس بندی می شوند. الگوریتم های یادگیری ماشین مانند gradient descent و k-Nearest Neighbors ، به داده مقیاس بندی شده نیاز دارند. با کمک کلاس *MinMaxScaler* از کتابخانه پایتون *scikit-learn* ، می توانیم داده ها را مقیاس بندی مجدد کنیم.

مثال:

در این مثال، داده‌ی مجموعه داده دیابتی های Pima Indians که پیشتر استفاده کردیم را مجددا مقیاس بندی می کنیم. ابتدا، همانند بخش های قبلی، داده CSV بارگیری می شود، سپس، با کمک کلاس *MinMaxScaler*، در بازه بین صفر و یک مقیاس بندی می شود.

چند خط اول اسکریپت زیر مشابه آن چیزی است که در بخش های قبلی حین بارگیری داده CSV نوشته بودیم.

```
from pandas import read_csv
from numpy import set_printoptions
from sklearn import preprocessing
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test',
'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

حال می توانیم برای مقیاس بندی داده در بازه بین صفر و یک از کلاس *MinMaxScaler* استفاده کنیم.

```
data_scaler =
preprocessing.MinMaxScaler(feature_range=(0,1))
data_rescaled = data_scaler.fit_transform(array)
```

همچنین می توانیم مطابق خواسته خود، داده را برای خروجی خلاصه کنیم. در اینجا، دقت را روی ۱ قرار می دهیم و ۱۰ سطر اول را در خروجی نشان می دهیم.

```
set_printoptions(precision=1)
print ("\nScaled data:\n", data_rescaled[0:10])
```

خروجی:

```
Scaled data:
[[0.4 0.7 0.6 0.4 0.  0.5 0.2 0.5 1. ]
[0.1  0.4 0.5 0.3 0.  0.4 0.1 0.2 0. ]
[0.5  0.9 0.5 0.  0.  0.3 0.3 0.2 1. ]
[0.1  0.4 0.5 0.2 0.1 0.4 0.  0.  0. ]
[0.   0.7 0.3 0.4 0.2 0.6 0.9 0.2 1. ]
[0.3  0.6 0.6 0.  0.  0.4 0.1 0.2 0. ]
[0.2  0.4 0.4 0.3 0.1 0.5 0.1 0.1 1. ]
[0.6  0.6 0.  0.  0.  0.5 0.  0.1 0. ]
[0.1  1.  0.6 0.5 0.6 0.5 0.  0.5 1. ]
[0.5  0.6 0.8 0.  0.  0.  0.1 0.6 1. ]]
```

بر اساس خروجی بالا، همه داده در بازه بین صفر و یک مقیاس بندی مجدد شده اند.

عادی سازی(Normalization) :

عادی سازی یکی دیگر از روش های مفید پیش پردازش داده است. از آن برای مقیاس بندی هر سطر از داده، برای اینکه دارای طول ۱ باشد، استفاده می شود. این روش اساسا برای مجموعه داده های پراکنده(Sparse) که در آن صفر های بسیاری داریم مفید است. می توان با کمک کلاس *Normalizer* از کتابخانه پایتون *scikit-learn* داده را مقیاس بندی کرد.

انواع عادی سازی:

دو نوع روش برای عادی سازی پیش پردازش در یادگیری ماشین وجود دارد به نام های L1 Normalization و L2 Normalization.

Binarization: همانطور که از نام آن بر می آید، این روشی است که به کمک آن می توانیم داده خود را دودویی(binary) کنیم. می توانیم از یک آستانه دودویی(binary threshold) برای دودویی کردن داده خود استفاده کنیم. مقادیر بالای آن آستانه به ۱ و زیر آستانه به صفر تبدیل می شوند.

برای مثال، اگر مقدار آستانه را ۰/۵ تعیین کنیم، در نتیجه مقادیر مجموعه داده که بالاتر از آن هستند تبدیل به ۱ و کمتر از آن تبدیل به صفر می شوند. به همین دلیل است که می توانیم آن را دودویی کردن(**binarizing**) یا آستانه بندی(**thresholding**) داده بنامیم. این روش زمانی مفید خواهد بود که در مجموعه داده خود احتمالات داشته باشیم و بخواهیم آن ها را به مقادیر قاطع تبدیل کنیم.

می توانیم با کمک کلاس ***Binarizer*** از کتابخانه پایتون ***scikit-learn*** داده را دودویی کنیم.

مثال:

در این مثال، داده‌ی مجموعه داده دیابتی های Pima Indians که پیشتر استفاده کردیم را مجددا مقیاس بندی می کنیم. ابتدا، داده CSV بارگیری می شود، سپس، با کمک کلاس ***Binarizer***، به مقادیر دودویی تبدیل می شود، مثلا صفر و یک، که به مقدار آستانه بستگی دارد. ما ۰/۵ را به عنوان مقدار آستانه در نظر می گیریم.

چند خط اول اسکریپت زیر مشابه آن چیزی است که در بخش های قبلی حین بارگیری داده CSV نوشته بودیم.

```
from pandas import read_csv
from sklearn.preprocessing import Binarizer
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test',
'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

حال، برای تبدیل داده به مقادیر دودویی می توانیم از کلاس ***Binarize*** استفاده کنیم.

```
binarizer = Binarizer(threshold=0.5).fit(array)
Data_binarized = binarizer.transform(array)
```

در اینجا، ۵ سطر اول را در خروجی نشان می دهیم.

```
print ("\nBinary data:\n", Data_binarized [0:5])
```

خروجی:

```
Binary data:
[[1. 1. 1. 1. 0. 1. 1. 1. 1.]
[1.  1. 1. 1. 0. 1. 0. 1. 0.]
[1.  1. 1. 0. 0. 1. 1. 1. 1.]
[1.  1. 1. 1. 1. 1. 0. 1. 0.]
[0.  1. 1. 1. 1. 1. 1. 1. 1.]]
```

استاندارد سازی(standardization): روش مفید دیگری برای پیش پردازش داده، استاندارد سازی است، که اساسا برای تبدیل ویژگی های داده توسط یک توزیع گوسی(Gaussian distribution) استفاده می شود. این روش میانگین(mean) و SD (انحراف از معیار - Standard Deviation) را به یک توزیع گوسی استاندارد با میانگین صفر و SD ۱ تبدیل می کند. این روش نیز مانند رگرسیون خطی(linear regression) و رگرسیون لجستیک(logistic regression) که یک توزیع گوسی را در مجموعه داده ورودی در نظر می گیرند، و نتایج بهتری با مقیاس بندی مجدد داده به دست می آورند، برای الگوریتم های ML مفید است. می توانیم با کمک کلاس *StandardScaler* از کتابخانه پایتون *scikit-learn*، داده را استاندارد سازی(میانگین =۰ و SD = ۱) کنیم.

مثال:

در این مثال، داده مجموعه داده دیابتی های Pima Indians که پیشتر استفاده کردیم را مجددا مقیاس بندی می کنیم. ابتدا، داده CSV بارگیری می شود، سپس، با کمک کلاس *StandardScaler* ، به توزیع گوسی با میانگین صفر و SD ۱ تبدیل می شود.

چند خط اول اسکریپت زیر مشابه آن چیزی است که در بخش های قبلی حین بارگیری داده CSV نوشته بودیم.

```
from sklearn.preprocessing import StandardScaler
from pandas import read_csv
from numpy import set_printoptions
path = r'C:\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test',
'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
```

حال، می توانیم از کلاس *StandardScaler* برای مقیاس بندی داده استفاده کنیم.

```
data_scaler = StandardScaler().fit(array)
data_rescaled = data_scaler.transform(array)
```

همچنین می توانیم داده خروجی را منطبق با خواسته خود خلاصه کنیم. در اینجا، دقت را روی ۲ قرار می دهیم و ۵ سطر اول را در خروجی نشان می دهیم.

```
set_printoptions(precision=2)
print ("\nRescaled data:\n", data_rescaled [0:5])
```

خروجی:

```
Rescaled data:
[[ 0.64 0.85  0.15  0.91 -0.69  0.2   0.47  1.43  1.37]
[-0.84 -1.12 -0.16  0.53 -0.69 -0.68 -0.37 -0.19 -0.73]
[ 1.23  1.94 -0.26 -1.29 -0.69 -1.1   0.6  -0.11  1.37]
[-0.84 -1.   -0.16  0.15  0.12 -0.49 -0.92 -1.04 -0.73]
[-1.14  0.5  -1.5   0.91  0.77  1.41  5.48 -0.02  1.37]]
```

برچسب داده(data labeling) :

درباره اهمیت داده خوب برای الگوریتم ها ML و همچنین برخی از روش های پیش پردازش داده، قبل از ارسال آنها به الگوریتم های ML صحبت کردیم. یکی دیگر از جنبه های مربوطه، برچسب داده است. همچنین بسیار مهم است که داده را به همراه برچسب مناسب برای الگوریتم های ML ارسال کنیم. برای مثال، در زمینه مسائل طبقه بندی(classification)، برچسب های بسیاری در قالب کلمات، اعداد و غیره در داده وجود خواهند داشت.

رمز گذاری برچسب چیست؟

اکثر توابع sklearn انتظار دارند برچسب های عددی و نه برچسب های کلمه ای به همراه داده باشد. اگر چه، ما باید چنین برچسب هایی را به برچسب های عددی تبدیل کنیم. این روند رمزگذاری برچسب نام دارد.

با کمک تابع *LabelEncoder()* از کتابخانه پایتون *scikit-learn* میتوانیم رمزگذاری برچسب را روی داده ها اعمال کنیم.

مثال:

در مثال زیر، اسکریپت پایتون رمزگذاری برچسب را انجام می دهد.

ابتدا، کتابخانه های پایتون مورد نیاز را به صورت زیر وارد(import) کنید.

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

حال، باید برچسب های ورودی را به صورت زیر ارائه کنیم.

```
input_labels =
['red','black','red','green','black','yellow','white'
]
```

خط بعدی کد، رمز گذار برچسب را تولید می کند و آن را آموزش می دهد.

```
encoder = preprocessing.LabelEncoder()
encoder.fit(input_labels)
```

خط بعدی اسکریپت با رمزگذاری یک لیست مرتب شده تصادفی، کارایی را بررسی می کند.

```
test_labels = ['green','red','black']
encoded_values = encoder.transform(test_labels)
print("\nLabels =", test_labels)
print("Encoded values =", list(encoded_values))
encoded_values = [3,0,4,1]
decoded_list =
encoder.inverse_transform(encoded_values)
```

با کمک اسکریپت پایتون زیر می توانیم لیست مقادیر رمزگذاری شده را به دست آوریم.

```
print("\nEncoded values =", encoded_values)
print("\nDecoded labels =", list(decoded_list))
```

خروجی:

```
 Labels = ['green', 'red', 'black']
Encoded values = [1, 2, 0]
Encoded values = [3, 0, 4, 1]
```

```
Decoded labels = ['white', 'black', 'yellow', 'green']
```