کتابخانه scikit learn در پایتون

نوشته شده توسط معین حاج ملک

کتابخانه scikit learnدر پایتون یکی از محبوب‌ترین کتابخانه‌های ماشین لرنینگ در پایتون می‌باشد که برای آموزش و پیاده‌سازی الگوریتم‌های یادگیری ماشین استفاده می‌شود. در ادامه به طور خلاصه آموزش کار با این کتابخانه را شرح می‌دهیم:

1. نصب کتابخانه: در ابتدا باید کتابخانه scikit-learn را نصب کنید. برای نصب این کتابخانه می‌توانید از دستور زیر استفاده کنید:

```
pip install scikit-learn
```

2. بارگذاری داده‌ها: بعد از نصب کتابخانه، باید دیتاست خود را به پروژه اضافه کنید. برای بارگذاری داده‌ها می‌توانید از کتابخانه Pandas استفاده کنید:

```
import pandas as pd
data = pd.read_csv('path/to/dataset.csv')
```

3. آماده‌سازی داده‌ها: پس از بارگذاری داده‌ها، باید آن‌ها را به شکل مناسب برای الگوریتم‌های یادگیری ماشین آماده کنید. برای مثال، ممکن است بخواهید داده‌ها را به شکل بردارهای عددی (numeric vectors) تبدیل کنید. برای این کار می‌توانید از کلاسهای مختلفی مانند LabelEncoder و OneHotEncoder استفاده کنید.
4. انتخاب الگوریتم: بعد از آماده‌سازی داده‌ها، باید یک الگوریتم یادگیری ماشین را انتخاب کنید. برای مثال، اگر بخواهید یک مدل پیش‌بینی کننده را ایجاد کنید، می‌توانید از الگوریتم‌های مختلفی مانند رگرسیون خطی (Linear Regression)، رگرسیون لجستیک(Logistic Regression) ، شبکه‌های عصبی (Neural Networks) و یا درخت تصمیم (Decision Trees) استفاده کنید.

5. آموزش مدل: پس از انتخاب الگوریتم، باید مدل را با استفاده از داده‌های آموزشی آموزش دهید. برای این کار می‌توانید از تابع fit که در کلاس الگوریتم موجود است استفاده کنید. برای مثال آموزش کتابخانه scikit-learn در پایتون، اگر بخواهید از رگرسیون لجستیک استفاده کنید، کد زیر را برای آموزش مدل استفاده کنید:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

6. ارزیابی مدل: پس از آموزش مدل، باید آن را برای داده‌های تست ارزیابی کنید. برای این کار می‌توانید از تابع predict که در کلاس الگوریتم موجود است استفاده کنید. برای مثال، اگر بخواهید مدل رگرسیون لجستیک را برای داده‌های تست ارزیابی کنید، کد زیر را برای این کار استفاده کنید:

```
predictions = model.predict(X_test)
```

7. بهبود عملکرد مدل: در مراحل قبل، ممکن است متوجه شوید که مدل شما به درستی عمل نمی‌کند. برای بهبود عملکرد مدل، می‌توانید از تکنیک‌های مختلفی مانند تغییر پارامترها(Hyperparameter Tuning) ، کاهش ابعاد داده‌ها (Dimensionality Reduction)و یا افزایش تعداد داده‌های آموزشی (Increasing Training Data)استفاده کنید.

این چند مرحله اساسی برای کار با آموزش کتابخانه scikit-learn در پایتون بود.

آموزش کتابخانه Scikit-learn در پایتون

آموزش کتابخانه scikit-learn در پایتون به عنوان یکی از محبوب‌ترین کتابخانه‌های ماشین لرنینگ در پایتون، قابلیت‌های بسیاری برای کار با داده‌های مختلف دارد. در اینجا به برخی از این قابلیت‌ها اشاره می‌کنیم:

1. بارگذاری داده‌ها: برای بارگذاری داده‌ها درscikit-learn ، می‌توانید از توابع موجود در ماژول datasets استفاده کنید. این ماژول شامل دیتاست‌های متداول در ماشین لرنینگ مانند دیتاست‌هایiris ، digitsو wine می‌باشد. برای بارگذاری دیتاست iris می‌توانید کد زیر را استفاده کنید:

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
```

2. تقسیم داده‌ها به دو دسته آموزشی و تست: برای تقسیم داده‌ها به دو دسته آموزشی و تست، می‌توانید از تابع train_test_split در ماژول model_selection استفاده کنید. این تابع، داده‌ها را به دو دسته آموزشی و تست تقسیم می‌کند. برای تقسیم دیتاست iris به دو دسته 80 درصد آموزشی و 20 درصد تست می‌توانید کد زیر را استفاده کنید:

```
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)
```

3. پیش‌پردازش داده: برای پیش‌پردازش داده‌ها، می‌توانید از توابع موجود در ماژول preprocessingاستفاده کنید. این ماژول شامل توابعی مانند StandardScaler برای مقیاس‌بندی داده‌ها، LabelEncoderبرای تبدیل داده‌های رشته‌ای به عددی و OneHotEncoderبرای تبدیل داده‌های دسته‌ای به بردارهای باینری می‌باشد. برای مقیاس‌بندی داده‌های iris می‌توانید کد زیر را استفاده کنید:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

4. تحلیل ترکیبی داده: برای تحلیل ترکیبی داده‌ها، می‌توانید از توابع موجود در ماژول decomposition استفاده کنید. این ماژول شامل توابعی مانند PCA برای تحلیل مؤلفه‌های اصلی و NMF برای تحلیل عاملی می‌باشد. برای تحلیل مؤلفه‌های اصلی داده‌های iris می‌توانید کد زیر را استفاده کنید:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

طبقه بندی با روش SVM در کتابخانه Scikit-learn

SVMیکی از الگوریتم‌های پرکاربرد در مسائل طبقه‌بندی ماشین لرنینگ است که در کتابخانه scikit-learn نیز پیاده‌سازی شده‌است. در ادامه، به طور خلاصه نحوه استفاده از الگوریتم SVM در scikit-learn را شرح می‌دهیم:

1. بارگذاری داده‌ها: در ابتدا باید داده‌های خود را به پروژه اضافه کنید. برای مثال، فرض کنید که می‌خواهید یک مدل SVM برای دسته‌بندی داده‌های ابرصفحه‌ای (linearly separable) ایجاد کنید. به این منظور، می‌توانید از دیتاست iris که در scikit-learn موجود است استفاده کنید. برای بارگذاری این دیتاست می‌توانید از کد زیر استفاده کنید:

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:100, :] # برای نمایش داده های خطی، فقط دو ویژگی اول انتخاب شده اند.
y = iris.target[:100]
```

2. تقسیم داده‌ها به دو دسته آموزشی و تست: برای تقسیم داده‌ها به دو دسته آموزشی و تست، می‌توانید از تابع train_test_split در ماژول model_selectionاستفاده کنید. برای تقسیم داده‌های iris به دو دسته 80 درصد آموزشی و 20 درصد تست، می‌توانید کد زیر را استفاده کنید:

```
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)
```

3. استفاده از SVM: در مرحله بعد، باید یک شی از کلاس SVM در scikit-learnتعریف کنید و آن را با داده‌های خود آموزش دهید. برای این کار، می‌توانید از کد زیر استفاده کنید:

```
from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
```

در این کد، یک شی از کلاس SVC تعریف شده است و با انتخاب kernel=’linear’، SVMخطی به‌کار رفته است. سپس، با استفاده از تابعfit ، مدل SVM با داده‌های آموزشی آموزش داده شده‌است.

4. پیش‌بینی: بعد از آموزش مدل، می‌توانید با استفاده از تابعpredict ، برای داده‌های تست پیش‌بینی‌های الگوریتم را انجام دهید. برای پیش‌بینی برچسب داده‌های تست، می‌توانید از کد زیر استفاده کنید:

```
y_pred = clf.predict(X_test)
```

5. ارزیابی مدل: در مرحله نهایی، برای ارزیابی عملکرد مدل، می‌توانید از توابع موجود در ماژول metrics استفاده کنید. به عنوان مثال، برای محاسبه دقت مدل SVM می‌توانید از تابع accuracy_score استفاده کنید. برای محاسبه دقت مدل SVM برای داده‌های تست، می‌توانید از کد زیر استفاده کنید:

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

باینری کردن

این تکنیک پیش‌پردازش زمانی به کار می‌رود که می‌خواهیم مقادیر عددی را به مقادیر بولین (منطقی) تبدیل کنیم.

```
import numpy as np
from sklearn import preprocessing
Input_data = np.array(
   [2.1, -1.9, 5.5],
   [-1.5, 2.4, 3.5],
   [0.5, -7.9, 5.6],
   [5.9, 2.3, -5.8]]
)
```

```
data_binarized =
preprocessing.Binarizer(threshold=0.5).transform(input_
data)
print("\nBinarized data:\n", data_binarized)
```

پارامتر threshhold_value=0.5 برای این است که داده‌هایی با مقدار بزرگتر از 0.5 به 1 و مقدار کوچکتر یا مساوی 0.5 به صفر تبدیل شوند.

```
Binarized data:

[

   [ 1. 0. 1.]

   [ 0. 1. 1.]

   [ 0. 0. 1.]

   [ 1. 1. 0.]

]
```

2حذف میانگین یاMean Removal

این روش برای حذف میانگین داده‌ها از بردار ویژگی است که منجر به نرمال‌سازی ویژگی‌ها با محدودیت صفر می‌شود.

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

```
Input_data = np.array(
    [2.1, -1.9, 5.5],
    [-1.5, 2.4, 3.5],
    [0.5, -7.9, 5.6],
    [5.9, 2.3, -5.8]]
)
#displaying the mean and the standard deviation of the
input data
print("Mean =", input_data.mean(axis=0))
print("Stddeviation = ", input_data.std(axis=0))
#Removing the mean and the standard deviation of the input
data
data_scaled = preprocessing.scale(input_data)
print("Mean_removed =", data_scaled.mean(axis=0))
print("Stddeviation_removed =", data_scaled.std(axis=0))
```

خروجی کد بالا به شکل زیر است:

```
Mean = [ 1.75 -1.275 2.2 ]


Stddeviation = [ 2.71431391 4.20022321 4.69414529]


Mean_removed = [ 1.11022302e-16 0.00000000e+00 0.00000000e+00]


Stddeviation_removed = [ 1. 1. 1.]
```

3.مقیاس‌دهی یا scaling

تابع scale روشی سریع و راحت برای انجام عمل مقیاس‌دهی به کار می‌برد. به کد زیر توجه کنید:

```
from sklearn import preprocessing
import numpy as np
X_train = np.array([[ 1., -1.,  2.],
                    [ 2.,  0.,  0.],
                    [ 0.,  1., -1.]])
X_scaled = preprocessing.scale(X_train)
X_scaled
```

خروجی کد به شکل زیر است:

```
array([[ 0.  ..., -1.22...,  1.33...],
       [ 1.22...,  0.  ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

داده‌ی مقیاس‌بندی شده‌ی حاصل، میانگین صفر و واریانس واحد دارد:

```
>>> X_scaled.mean(axis=0)
array([0., 0., 0.])
>>> X_scaled.std(axis=0)
array([1., 1., 1.])
```

ماژول preprocessing کلاسی تحت عنوان stanardscaler دارد که حذف میانگین و تقسیم واریانس را بر روی داده‌های آموزشی اجرا می‌کند و توسط transformer API خود امکان اعمال حذف همین مقدار میانگین و تقسیم همان واریانس محاسبه شده‌ی حاصل از داده‌های آموزشی را بر روی داده‌های تست فراهم می‌کند.

```
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> scaler
StandardScaler()
>>> scaler.mean_
array([1. ..., 0. ..., 0.33...])
```

```
>>> scaler.scale_
array([0.81..., 0.81..., 1.24...])
>>> scaler.transform(X_train)
array([[ 0.   ..., -1.22...,  1.33...],
       [ 1.22...,  0.   ..., -0.26...],
       [-1.22...,  1.22..., -1.06...]])
```

در زیر همین تبدیل را روی داده‌های تست اعمال می‌کنیم:

```
>>> X_test = [[-1., 1., 0.]]
>>> scaler.transform(X_test)
array([[-2.44...,  1.22..., -0.26...]])
```

اگر به هر دلیلی بخواهیم در کلاس standardscaler امکان حذف میانگین و یا تقسیم شدن داده به انحراف معیار را حذف کنیم، به ترتیب می‌توانیم تنظیمات را به صورت with_mean=False و with_std=False اعمال کنیم.

یک روش استاندارد جایگزین مقیاس‌دهی به ویژگی‌ها، مقداردهی به آن‌ها در بازه‌ی حداقلی و حداکثری (معمولاً بین 0 و 1) است. روش دیگر، تعیین حداکثر مقدار مطلق یک ویژگی به اندازه‌ی واحد است.

4.نرمال‌سازی

نرمال‌سازی بردار ویژگی برای این که ویژگی‌ها در مقیاس یکسانی قرار گیرند به کار می‌آید. دو روش نرمال‌سازی داریم:

- **نرمال‌سازی L1**

این روش که به آن حداقل انحراف مطلق نیز می‌گویند، مقادیر ویژگی‌ها را به گونه‌ای تغییر می‌دهد که مجموع مقادیر مطلق در هر سطر حداکثر 1 باقی بماند.

در مثال زیر اجرای نرمال‌سازی L1 را روی داده‌های ورودی می‌بینیم:

```
import numpy as np
from sklearn import preprocessing
Input_data = np.array(
    [
        [2.1, -1.9, 5.5],
        [-1.5, 2.4, 3.5],
        [0.5, -7.9, 5.6],
        [5.9, 2.3, -5.8]
    ]
)
data_normalized_l1 = preprocessing.normalize(input_data,
norm='l1')
print("\nL1 normalized data:\n", data_normalized_l1)
```

خروجی حاصل به شکل زیر است:

```
L1 normalized data:

[

   [ 0.22105263 -0.2 0.57894737]

   [-0.2027027 0.32432432 0.47297297]

   [ 0.03571429 -0.56428571 0.4 ]

   [ 0.42142857 0.16428571 -0.41428571]
```

```
]
```

. نرمال‌سازی L2

این روش که حداقل مربعات نیز نامیده می‌شود، مقادیر را به گونه‌ای تغییر می‌دهد که مجموع مربعات در هر سطر حداکثر یک باقی بماند. به مثال زیر توجه کنید:

```
import numpy as np
from sklearn import preprocessing
Input_data = np.array(
    [
        [2.1, -1.9, 5.5],
        [-1.5, 2.4, 3.5],
        [0.5, -7.9, 5.6],
        [5.9, 2.3, -5.8]
    ]
)
data_normalized_l2 = preprocessing.normalize(input_data,
norm='l2')
print("\nL1 normalized data:\n", data_normalized_l2)
```

خروجی حاصل به شکل زیر است:

```
L2 normalized data:


[


   [ 0.33946114 -0.30713151 0.88906489]


   [-0.33325106 0.53320169 0.7775858 ]
```

```
   [ 0.05156558 -0.81473612 0.57753446]


   [ 0.68706914 0.26784051 -0.6754239 ]


]
```

این چند مرحله‌ای که برای طبقه‌بندی با روش SVM در آموزش کتابخانه scikit-learn در پایتون باید انجام داد، نشان می‌دهد که استفاده از این الگوریتم در این کتابخانه بسیار ساده و قابل استفاده می‌باشد.

منابع

Mihanpy.com

7learn.com