Makine Öğrenmesi

Scikit-Learn

İlhan AYDIN

Veri Yükleme-Pandas

- Python Sınıfları
- Bu bölüm, bu derste scikit-learn API'sinin temel düzeyde nasıl çalıştığını anlamak için uygun olan Python'daki "sınıflar" kavramını göstermektedir.
- Python'un nesne yönelimli bir dil olduğunu ve Python'daki her şeyin bir nesne olduğunu unutmamak gerekir.
- Sınıflar, nesneler oluşturmak için "şablonlardır" (buna nesneler "örnekleme" denir).
- Bir nesne, özel «fonksyionlar" (bir nesnenin veya sınıfın "işlevi"ne "yöntem" denir) ve nitelikler topluluğudur.
- self niteliğinin, bir sınıfa veya bir sınıfın somutlaştırılmış nesnesine, "kendisine" atıfta bulunmak için özel bir anahtar kelime olduğuna dikkat etmek gerekir.

```
import numpy as np
class VehicleClass():

    def __init__(self, horsepower):
        "This is the 'init' method"
        # this is a class attribute:
        self.horsepower = horsepower

    def horsepower_to_torque(self, rpm):
        "This is a regular method"
        numerator = self.horsepower * 33000
        denominator = 2* np.pi * 5000
        return numerator/denominator

    def tune_motor(self):
        self.horsepower *= 2

    def _private_method(self):
        print('this is private')

    def __very_private_method(self):
        print('this is very private')
```

Python Sınıfları

```
# instantiate an object:
car1 = VehicleClass(horsepower=123)
print(car1.horsepower)
```

```
123
```

```
car1.horsepower_to_torque(rpm=5000)
```

```
129.20198280200063
```

```
car1.tune_motor()
car1.horsepower_to_torque(rpm=5000)
```

```
258.40396560400126
```

Python Sınıfları

```
car1._private_method()
```

- Python'un "burada hepimiz yetişkiniz" sloganı vardır; bu, bir kullanıcının bir geliştiriciyle aynı şeyleri yapabileceği anlamına gelir (diğer programlama dillerinin aksine, örneğin Java).
- Bir önceki alt çizgi, bir yöntemin «private" olarak kabul edildiğinin bir göstergesidir - bu, bu yöntemin dahili olarak kullanılması gerektiği, ancak doğrudan kullanıcı tarafından kullanılmadığı anlamına gelir
- Çift alt çizgi, özel olması gereken yöntemler için "daha güçlü" bir göstergedir.

Python Sınıfları

Python Sınıfları-Miras

- Sınıfları kullanmanın bir başka yararlı yönü de "miras" kavramıdır.
- Kalıtımı kullanarak, yeniden kullanım için bir üst sınıftan yöntemleri ve nitelikleri "miras alabiliriz".
- Örneğin, VehicleClass'ı CarClass'tan daha genel bir sınıf olarak düşünün - yani bir araba, kamyon veya motosiklet, bir aracın özel durumlarıdır.
- Aşağıda, VehicleClass'tan yöntemleri miras alan ve belirli bir self.num_wheels=4 özniteliği ekleyen bir CarClass örneği verilmiştir -- örneğin, bir BikeClass oluşturacak olsaydık, bunu self.num_wheels=2 olarak ayarlayabilirdik.

```
class CarClass(VehicleClass):

    def __init__(self, horsepower):
        super(CarClass, self).__init__(horsepower)
        self.num_wheels = 4

new_car = CarClass(horsepower=123)
print('Number of wheels:', new_car.num_wheels)
print('Horsepower:', new_car.horsepower)
new_car.tune_motor()
print('Horsepower:', new_car.horsepower)
```

```
Number of wheels: 4
Horsepower: 123
Horsepower: 246
```

Python Sınıfları-Miras

Scikit-Learn API

- Aşağıda, sınıflandırma ve regresyon modellerini/algoritmalarını uygulamak için kullanılan scikit-learn tahmincisi API'sine genel bir bakış yer almaktadır.
- Skor yöntemi, dahili olarak özellikler (X) üzerinde tahmin yürütür ve ardından öngörülen hedefleri gerçek hedefler y ile karşılaştırarak performansı hesaplar.
- Sınıflandırma modelleri durumunda, puan yöntemi sınıflandırma doğruluğunu ([0, 1] aralığında) - yani doğru tahmin edilen etiketlerin oranını hesaplar. Regresyon modelleri durumunda, puan yöntemi belirleme katsayısını hesaplar.

```
class SupervisedEstimator(...):

    def __init__(self, hyperparam_1, ...):
        self.hyperparm_1
        ...

    def fit(self, X, y):
        ...
        self.fit_attribute_
        return self

    def predict(self, X):
        ...
        return y_pred

    def score(self, X, y):
        ...
        return score

    def _private_method(self):
        ...
    ...
```

```
import pandas as pd

df=pd.read_csv('/content/drive/MyDrive/Iris.csv')
df=pd.DataFrame(df)
d = {'Iris-setosa': 0,
     'Iris-versicolor': 1,
     'Iris-virginica': 2}
df['Species'] = df['Species'].map(d)
```

```
import numpy as np
indices = np.arange(df.shape[0])
rng = np.random.RandomState(123)
permuted_indices = rng.permutation(indices)
permuted_indices
```

Veri Yükleme

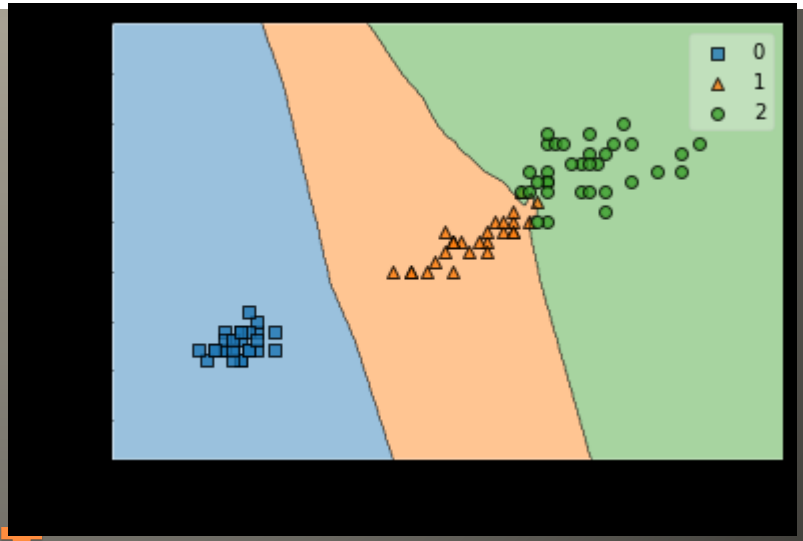
```
train_size, valid_size = int(0.65*df.shape[0]), int(0.15*d
f.shape[0])
test_size = df.shape[0] - (train_size + valid_size)
print(train_size, valid_size, test_size)
```

```
train_ind = permuted_indices[:train_size]
valid_ind = permuted_indices[train_size:(train_size + vali
d_size)]
test_ind = permuted_indices[(train_size + valid_size):]
```

```
X_train,y_train = df.values[train_ind,1:5],df.values[train_
ind,5]
X_valid, y_valid = df.values[valid_ind,1:5], df.values[vali
d_ind,5]
X_test, y_test = df.values[test_ind,1:5], df.values[test_in
d,5]
```

Veri Yükleme

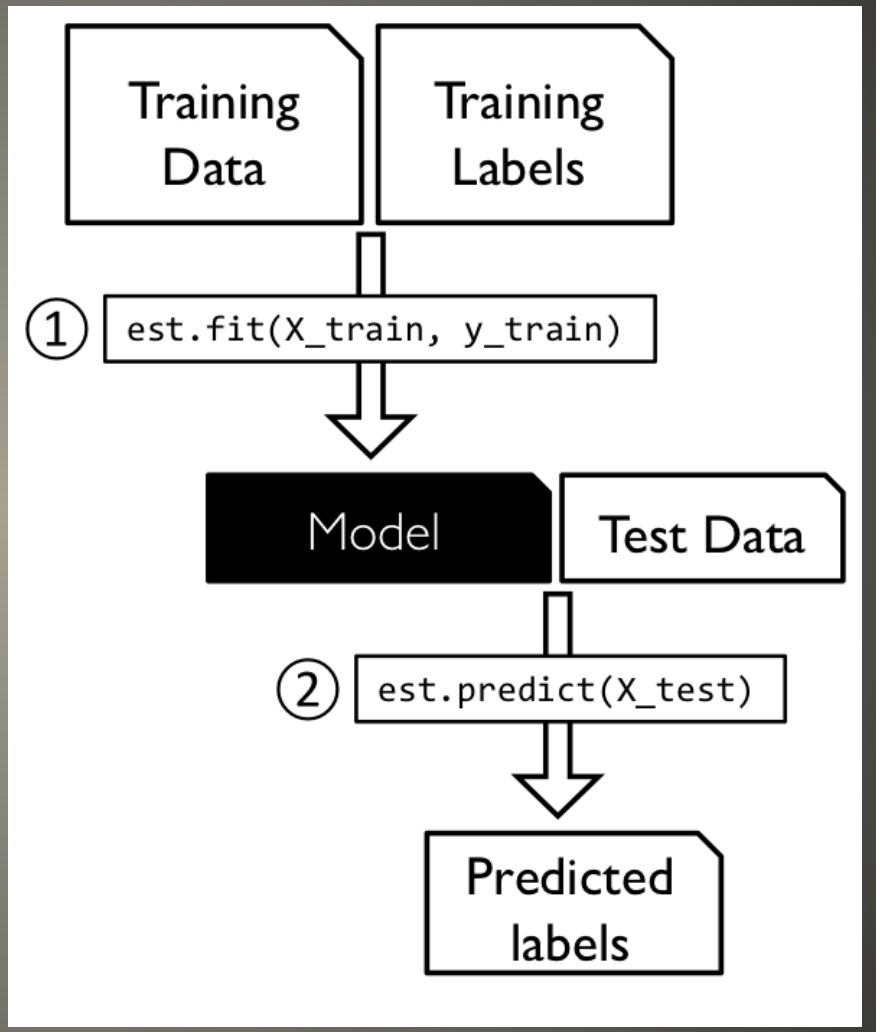
```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.plotting import plot_decision_regions
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train[:, 2:], y_train)
y=y_train.astype(int)
plot_decision_regions(X_train[:, 2:], y, knn_model)
plt.xlabel('petal length[cm]')
plt.ylabel('petal width[cm]')
plt.show()
```



Scikit-Learn API

Scikit-Learn API

- Yan taraftya verilen grafik SupervisedEstimator api'sinin çalışma şeklini açıklar.



Scikit-Learn Verileri Bölme

- Önceden, bir veri kümesini önemli bir dezavantajı olan eğitim, doğrulama ve test alt kümelerine karıştırmak ve bölmek için kendi kodumuzu yazdık.
- Küçük veri kümeleriyle çalışıyor ve onu rastgele alt kümelere bölüyorsak, örneklerdeki sınıf dağılımını etkileyecektir --
- Makine öğrenimi algoritmaları/modelleri, güvenilir modeller ve genelleme performansı tahminleri üretmek için eğitim, doğrulama ve test örneklerinin aynı dağılımlardan alındığını varsaydığından bu sorunludur.

```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/Iris.csv')
df=pd.DataFrame(df)
d = {'Iris-setosa': 0,
     'Iris-versicolor': 1,
     'Iris-virginica': 2}
df['Species'] = df['Species'].map(d)
df.head()
```

```
import numpy as np
from sklearn.model_selection import train_test_split
X=df.values[1:150,1:5]
y=df.values[1:150,5]
X_temp, X_test, y_temp, y_test = train_test_split(X, y, t
est_size=0.2, shuffle=True, random_state=123, stratify=y)
np.bincount(y_temp.astype(int))
```

Verileri Bölme

Scikit-Learn Veri Ölçeklendirme

- Iris veri kümesi durumunda, tüm boyutlar santimetre olarak ölçülmüştür, bu nedenle özellikleri farklı şekilde ağırlıklandırmak istemediğimiz sürece kNN bağlamında "ölçeklendirme" özellikleri gerekli olmayacaktır.
- Özelliklerin ölçeklenip ölçeklenmeyeceği, eldeki soruna bağlıdır ve sizin karar vermenizi gerektirir.
- Ancak, veriler ortalanmışsa ve daha küçük bir aralığa sahipse çok daha iyi çalışan (daha sağlam, sayısal olarak kararlı ve daha hızlı yakınsayan) birkaç algoritma vardır.
- Özellikleri ölçeklendirmenin birçok farklı yolu vardır; burada sadece en yaygın "normalleştirme" şemalarını ele alıyoruz: min-maks ölçekleme ve z-skor standardizasyonu.

Normalizasyon-Min-Max Scaling

- Min-maks ölçekleme, özellikleri [0, 1] aralığına sıkıştırır ve bu, tek bir giriş i için aşağıdaki denklem aracılığıyla elde edilebilir:
- $x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$
- Aşağıda, NumPy aracılığıyla 1B giriş vektörü (1 özellik) verilen 6 veri örneğinde min-maks ölçeklendirmeyi nasıl uygulayabileceğimize ve uygulayabileceğimize bir örnek verilmiştir.

```
x = np.arange(6).astype(float)
print(x)
x_norm = (x - x.min()) / (x.max() - x.min())
print(x_norm)
```

- Uygun bir şekilde, NumPy ve Pandaların ikisi de standart sapmayı hesaplayan bir std yöntemi uygular. Aşağıda gösterilen farklı uygulayabileceğimize bir örnek verilmiştir.

```
df = pd.DataFrame([1, 2, 1, 2, 3, 4])
print(df[0].std())
print(df[0].values.std())
1.1690451944500122
1.067187372905474 8
```

Pandas "örnek" standart sapmayı hesaplarken, NumPy "popülasyon" standart sapmasını hesapladığı için sonuçlar farklıdır.

Normalizasyon-Min-Max Scaling

```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/Iris.csv')
df=pd.DataFrame(df)
d = {'Iris-setosa': 0,
     'Iris-versicolor': 1,
     'Iris-virginica': 2}
df['Species'] = df['Species'].map(d)
```

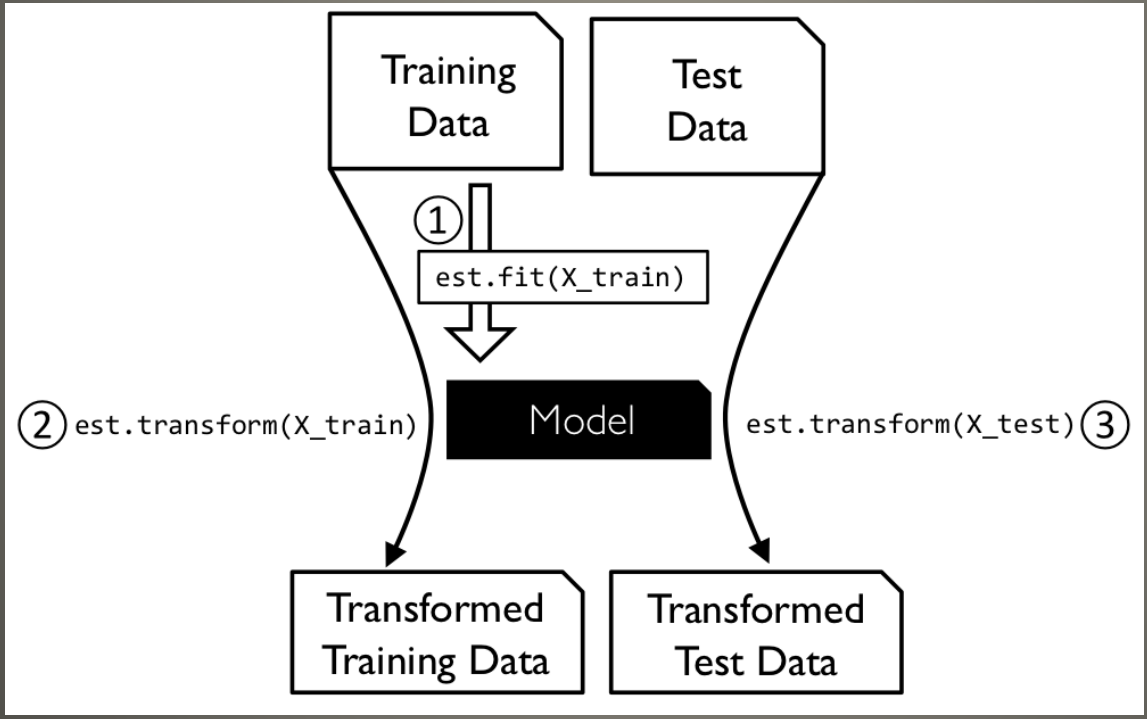
```
import numpy as np
from sklearn.model_selection import train_test_split
X=df.values[1:150,1:5]
y=df.values[1:150,5]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                            shuffle=True, random_state=123, stratify=y)
np.bincount(y_train.astype(int))
```

```
mu, sigma = X_train.mean(axis=0), X_train.std(axis=0)
X_train_std = (X_train - mu) / sigma
X_valid_std = (X_valid - mu) / sigma
X_test_std = (X_test - mu) / sigma
print(X_train_std)
```

Normalizasyon-Standardization

Scikit-Learn Transformer API

- scikit-learn'deki dönüştürücü API'si, tahmin edici API'ye çok benzer; temel fark, transformatörlerin tipik olarak "denetimsiz" olmalarıdır, yani sınıf etiketlerini veya hedef değerleri kullanmazlar.



```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
scaler.fit(X_train)
X_train_std = scaler.transform(X_train)
X_valid_std = scaler.transform(X_valid)
X_test_std = scaler.transform(X_test)
```

Scikit-Learn Transformer API

Kategorik Veri

- Bir veri kümesini bir makine öğrenimi algoritmasına girdi olarak önceden işlediğimizde, kategorik değişkenleri nasıl ele aldığımıza dikkat etmeliyiz.
- İki geniş kategorik değişken kategorisi vardır: nominal (sıra ima edilmez) ve sıralı (sıra ima edilir).

```
import pandas as pd
df=pd.read_csv("/content/drive/MyDrive/categoricaldata.csv")
df
```

```
mapping_dict = {'M': 2,
                'L': 3,
                'XXL': 5}


df['size'] = df['size'].map(mapp
ing_dict)
df
```

	color	size	price	classlabel
0	green	2	10.1	class1
1	red	3	13.5	class2
2	blue	5	15.3	class1

Kategorik Veri

- Bir veri kümesini bir makine öğrenimi algoritmasına girdi olarak önceden işlediğimizde, kategorik değişkenleri nasıl ele aldığımıza dikkat etmeliyiz.
- İki geniş kategorik değişken kategorisi vardır: nominal (sıra ima edilmez) ve sıralı (sıra ima edilir).

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['classlabel'] = le.fit_transform(df['classlabel'])
d
```

	color	size	price	classlabel
0	green	2	10.1	0
1	red	3	13.5	1
2	blue	5	15.3	0