دوره جامع پایتون:

بخش یادگیری ماشین

جلسه بیستم و سوم

دکتر ذبیح اله ذبیحی

مثال

- مجموعه داده ای مربوط به مشتریان یک توزیع کننده عمده فروشی در اختیار هست. این شامل اطلاعاتی مانند هزینه های سالانه مشتریان برای محصولات تازه ، محصولات شیر ، محصولات خواربار و غیره است.

- FRESH: annual spending (m.u.) on fresh products (Continuous)
- MILK: annual spending (m.u.) on milk products (Continuous)
- GROCERY: annual spending (m.u.) on grocery products (Continuous)
- FROZEN: annual spending (m.u.) on frozen products (Continuous)
- DETERGENTS_PAPER: annual spending (m.u.) on detergents and paper products (Continuous)
- DELICATESSEN: annual spending (m.u.) on delicatessen products (Continuous)
- CHANNEL: customer channels - Horeca (Hotel/Restaurant/Cafe) or Retail channel (Nominal)
- REGION: customer regions - Lisnon, Oporto or Other (Nominal)

```
# Import required packages
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
data = pd.read_csv('Wholesale customers data.csv')
#print(data)
categorical_features = ['Channel', 'Region']
continuous_features = ['Fresh', 'Milk',
    'Grocery', 'Frozen', 'Detergents_Paper', 'Delicassen']
#print(data[continuous_features].describe())

mms = MinMaxScaler()
mms.fit(data)
data_transformed = mms.transform(data)
```

```
Sum_of_squared_distances = []
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k)
    km = km.fit(data_transformed)
    Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

نکته: sklearn.preprocessing

StandardScaler:

Z=(x-u)/s

u is the mean of the training samples

---

min_max_scaler: [0,1]

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

X_scaled = X_std * (max - min) + min

MaxAbsScaler:[-1,1]

مثال: تشخیص زن یا مرد بود

- با توجه به جدول فردی با قد ۱۳۳ و اندازه مو ۳۷ مرد هست یا زن:
- یادگیری تحت نظارت : رگرسیون لجستیک
- یادگیری تحت نظارت : نزدیکترین همسایه
- یادگیری بدون نظارت: خوشه بندی kmean

زن/مرد	سایز مو	قد	ردیف		زن/مرد	سایز مو	قد	ردیف
m	۱۰	۱۷۹	۱۰		m	۱۹	۱۶۹	۱
w	۳۴	۱۳۶	۱۱		w	۳۲	۱۷۵	۲
m	۲	۱۸۶	۱۲		w	۳۵	۱۳۶	۳
w	۲۵	۱۲۶	۱۳		m	۶۵	۱۷۴	۴
w	۲۸	۱۷۶	۱۴		w	۲۸	۱۴۱	۵
w	۳۸	۱۱۲	۱۵		m	۱۵	۱۷۶	۶
m	۹	۱۶۹	۱۶		w	۳۲	۱۳۱	۷
w	۳۶	۱۷۱	۱۷		m	۶	۱۶۶	۸
w	۲۵	۱۱۶	۱۸		w	۳۲	۱۲۸	۹
m	۲۵	۱۹۶	۱۹					

```
import numpy as np
#w=0 , m=1
data=np.array([[169,19,1],[175,32,0],[136,35,0],[174,65,1],
    [141,28,0],[176,15,1],[131,32,0],[166,6,1],
    [128,32,0],[179,10,1],[136,34,0],[186,2,1],
    [126,25,0],[176,28,0],[112,38,0],[169,9,1],
    [171,36,0],[116,25,0],[196,25,0]])

x=data[:,:2]
y=data[:,2]

from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
```

```
from sklearn.linear_model import LogisticRegression

model1=LogisticRegression()
model1.fit(x,y)
print("logisticregression=",model1.predict([[133,37]]))
```

```
from sklearn.neighbors import KNeighborsClassifier

model2=KNeighborsClassifier()
model2.fit(x,y)
print("KNeighborsClassifier=",model2.predict([[133,37]]))
```

```
from sklearn.cluster import KMeans

model3=KMeans(n_clusters=2)
model3.fit(x)
print(model3.labels_)
print("kmeans=", model3.predict([[133,37]]))
```

مثال: تشخیص تومور خوش خیم و بد خیم

- با توجه به جدول فردی با سن ۲۸ و اندازه تومور ۵ دارای تومور خوش خیم است یا بد خیم:
- یادگیری تحت نظارت : رگرسیون لجستیک
- یادگیری تحت نظارت : نزدیکترین همسایه
- یادگیری بدون نظارت: خوشه بندی kmean

خوش خیم ۰ بدخیم ۱	سن	سایز تومور	ردیف		خوش خیم ۰ بدخیم ۱	سن	سایز تومور	ردیف
۰	۲۰	۵	۱۰		۰	۲۳	۵	۱
۰	۲۵	۵	۱۱		۰	۲۵	۱۰	۲
۱	۳۰	۵	۱۲		۱	۲۳	۱۵	۳
۰	۱۵	۱۰	۱۳		۰	۱۷	۵	۴
۱	۴۵	۱۵	۱۴		۱	۳۵	۵	۵
۱	۴۳	۱۵	۱۵		۰	۱۹	۱۰	۶
۱	۱۵	۱۰	۱۶		۱	۳۰	۱۵	۷
۰	۲۰	۱۰	۱۷		۱	۳۵	۱۵	۸
۱	۴۰	۱۰	۱۸		۱	۴۰	۱۰	۹

```
import numpy as np
#۱=بدخیم , ۰= خوش خیم
data=np.array([[5,23,0],[10,25,0],[15,23,1],[5,17,0],
    [5,35,1],[10,19,0],[15,30,1],[15,35,1],
    [10,40,1],[5,20,0],[5,25,0],[5,30,1],
    [10,15,0],[15,45,1],[15,43,1],[10,15,1],
    [10,20,0],[10,40,1]])

x=data[:,:2]
y=data[:,2]

from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
```

```
from sklearn.linear_model import LogisticRegression

model1=LogisticRegression()
model1.fit(x,y)
print("logisticregression=",model1.predict([[5,28]]))
```

```
from sklearn.neighbors import KNeighborsClassifier


model2=KNeighborsClassifier()
model2.fit(x,y)
print("KNeighborsClassifier=",model2.predict([[5,28]]))
```

```
from sklearn.cluster import KMeans


model3=KMeans(n_clusters=2)
model3.fit(x)
print(model3.labels_)
print("kmeans=",model3.predict([[5,28]]))
```

الگوریتم خوشه بندی DBSCANمبتنی بر غلظت

- Density Based Spatial Clustering of Applications with Noise
- خوشه‌بندی فضایی مبتنی بر چگالی در کاربردهای دارای نویز

- بَعد از الگوریتمِ خوشه‌بندی KMeans، الگوریتمِ DBSCANرا می توان معروف‌ترین الگوریتمْ در حوزه‌ی یادگیری ماشین بدون نظارت خوشه‌بندیِ داده‌ها دانست.
- تفاوت اصلی الگوریتم DBSCAN با KMeans این است که الگوریتم DBSCANنیاز به تعیین تعدادِ خوشه توسط کاربر ندارد و خودِ الگوریتم می‌تواند خوشه‌ها را مبتنی بر غلظتِ آن‌ها شناسایی کند.
- گروه بندیِ داده ها بر حسبِ تراکمی است که که در آن قرار دارند.

- دو پارامتر مهم الگوریتم DBSCAN f به ترتیب Epsilon (شعاع) وMinsamples (حداقل نقاط موجود در یک خوشه) می باشند.
- هر چه شعاع را کوچکتر در نظر بگیرید، خوشه‌های بیشتر و کوچکتری تشکیلی می‌شود. همچنین هر چه قدر Minsamplesرا بزرگ‌تر در نظر بگیرید، احتمال ایجادِ خوشه‌ها، کمتر می‌شود زیرا الگوریتم باید تعداد نمونه‌های بیشتری را در یک شعاع خاص ببیَنید تا خوشه را تشکیل دهد.
- به نقاطی که حداقل میزان Minsamplesرا در شعاع خود داشته باشند، **نقاط هسته یا Core**می گویند.

نحوه عملکرد الگوریتم

- ابتدا یک نمونه (که همان یک نقطه در فضای برداری می‌شود) را انتخاب می‌کند و با توجه به شعاعِ Epsilonبه دنبال همسابه برای این نقطه در فضا می‌گردد. اگر الگوریتم در آن شعاعِ مشخَصِ Epsilonحداقل توانست به تعدادِ Minsamplesنقطه پیدا کند، آن‌گاه همه‌ی آنَ نقطه‌هاَ با هم به یک خوشه تعلق می‌گیرند.َ
- الگوریتمْ سپس به دنبال یکی از نقطه‌های همجوار نقطه فعلی می‌رود تا دوباره با شعاعِ Epsilonدر آن نقطه به دنبالِ نقاط همسایه دیگر بگردد و اگر تعداد نقاط همسایه‌ی جدیدَ بازهم پیدا شوند، این الگوریتم دَوباره همه آن نقاط جدید را با نقاط قبلَی به َیک خوشه متعلق می‌کند و اگر نقطه‌ی جدیدی در همسایگی پیدا نکرد این خوشه تمام شده است و برای پیدا کردنِ خوشه‌های دیگر در نقاط دیگر، به صورت تصادفی یک نقطه دیگر را انتخاب کرده و شروعَ به یافتن همسایه و تشکیلِ خوشه‌ی جدید برای آن نقطه می‌کند. این کار آنقدر ادامه پیدا می کند تا تمامی نقاط بررسیَ شوند.

مثال

```
from sklearn.cluster import DBSCAN
import numpy as np
X = np.array([[1, 2], [2, 2], [2, 3],
              [8, 7], [8, 8], [25, 80]])
model = DBSCAN(eps=3, min_samples=2).fit(X)
print(model.labels_)
```

Noisy samples are given the label -1

مثال

- ما دو دسته ماشین (مثلا پراید و اتوبوس) داریم. ما فقط اطلاعات ماشین ها (طول و ارتفاع) را داریم اما نوع ماشین را نمی دانیم (برای اطلاعات برچسب نداریم).

ارتفاع ماشین	طول ماشین	
۳	۷	۱
۴	۷	۲
۳	۸	۳
۳	۹	۴
۲	۵	۵
۳	۴	۶
۴	۳	۷

```
import numpy as np
X = np.array([[7, 3], [7, 4], [8, 3],[9, 3], [5, 2], [4, 3],[3,3]])
from sklearn.cluster import DBSCAN
model = DBSCAN(eps=2, min_samples=2).fit(X)

print(model.labels_)
#print(model.fit_predict([[8,4]]))
```

مثال: اگر یک محصول با وزن ۱۲۵ ، اندازه ۴۵ و رنگ آبی دیده شود مربوط به کدام دسته A یا B خواهد بود.

برحسب	رنگ	اندازه (سانتی متر)	وزن (کیلوگرم)	نمونه
A	1	50	120	1
B	2	20	60	2
A	1	65	145	3
A	3	45	130	4
B	2	15	50	5

```
import numpy as np
X = np.array([[120, 50, 1],[60, 20, 2],
              [145, 65, 1],[130, 45, 3],
              [50, 15, 2]])

from sklearn.cluster import DBSCAN
model = DBSCAN(eps=60, min_samples=2).fit(X)

print(model.labels_)
#print(model.fit_predict([[125,45,1]]))
```

الگوریتم های کاهش ابعاد (Dimensional Reduction Algorithms) :

- بعضی از دیتاست ها ممکن است که متغیر های زیادی داشته باشند که این موضوع باعث سخت تر شدن کار با آنها می شود.به ویژه امروزه، اطلاعات جمع آوری شده در سیستم ها به دلیل وجود منابع بیش از اندازه ، در سطح بسیار دقیق رخ می دهد. در چنین مواردی، مجموعه داده ها ممکن است شامل هزاران متغیر باشد و اکثر آنها نیز غیر ضروری می باشند.

- در این موارد، تقریبا غیرممکن است که متغیرهایی را که بیشترین تاثیر را در پیش بینی ما دارند را شناسایی کنیم. الگوریتم های کاهش اندازه در این نوع شرایط استفاده می شود.با استفاده از الگوریتم های دیگر مانند جنگل تصادفی و درخت تصمیم برای شناسایی مهم ترین متغیرها اقدام میکنیم.

```
import numpy as np
from sklearn.decomposition import PCA
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
pca = PCA(n_components=2)
pca.fit(X)
PCA(n_components=2)
print(pca.explained_variance_ratio_)
print(pca.singular_values_)
```

تمرین

داده های جنایی جمع آوری شده در ایالت های مختلف ایالات متحده شامل جنایاتی است مانند: حمله ، قتل و تجاوز به قتل در هر ۱۰۰۰۰۰ ساکن در هر یک از ۱۵ ایالت آمریکا در سال ۱۹۷۳ در جدول زیر گرداوری شده است. تعداد خوشه های بهینه برای این مجموعه داده را استخراج کنید. (روش Kmean)

row.names	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7
Connecticut	3.3	110	77	11.1
Delaware	5.9	238	72	15.8
Florida	15.4	335	80	31.9
Georgia	17.4	211	60	25.8
Hawaii	5.3	46	83	20.2
Idaho	2.6	120	54	14.2
Illinois	10.4	249	83	24.0
Indiana	7.2	113	65	21.0
Iowa	2.2	56	57	11.3