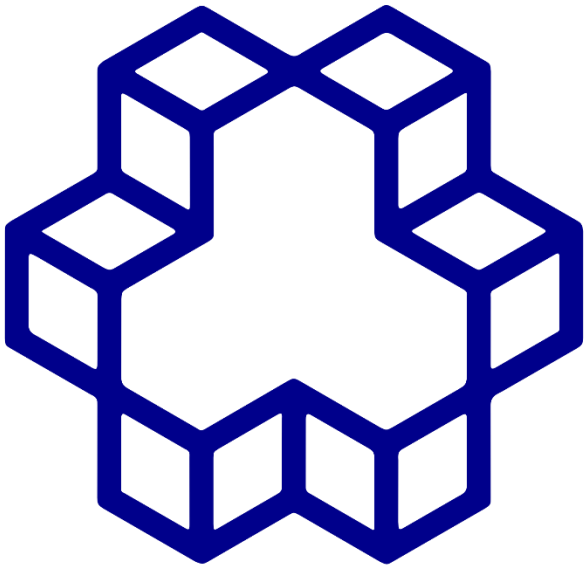


دانشگاه صنعتی خواجه نصیرالدین طوسی

پروژه درس مبانی سیستم های هوشمند

استاد درس: دکتر علیاری

دانشجو: علی خدارحمی ۹۸۲۱۵۲۳

آدرس کدها در گیت هاب:

https://github.com/AliKhodarahmy/machinelearning2023/tree/main/Finalproject

خواندن دیتاست و اطلاعات آن

داده از گوگل درایو در محیط کولب اجرا میکنیم:

```
!pip install --upgrade --no-cache-dir gdown
!gdown 1peAb1oNexs50eXbeWJ0_AVk25VskKw2L
```

کتابخانه های مورد نیاز (و احتمالا مورد نیاز) را فراخوانی میکنیم:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from imblearn.under_sampling import RandomUnderSampler
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import random
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.metrics import r2_score
```

مشاهده دیتاست:

	Hydrogen	Oxigen	Nitrogen	Methane	CO	CO2	Ethylene	Ethane	Acethylene	DBDS	Power factor	Interfacial V	Dielectric rigidity	Water content	Health index
0	2845	5860	27842	7406	32	1344	16684	5467	7	19.0	1.00	45	55	0	95.2
1	12886	61	25041	877	83	864	4	305	0	45.0	1.00	45	55	0	85.5
2	2820	16400	56300	144	257	1080	206	11	2190	1.0	1.00	39	52	11	85.3
3	1099	70	37520	545	184	1402	6	230	0	87.0	4.58	33	49	5	85.3
4	3210	3570	47900	160	360	2130	4	43	4	1.0	0.77	44	55	3	85.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
465	15	227	52900	3	60	853	3	84	0	0.0	1.00	32	56	28	13.4
466	15	334	47100	3	64	622	3	108	0	0.0	1.00	32	55	12	13.4
467	15	1280	35000	2	675	2530	0	0	0	5.0	0.30	45	58	8	13.4
468	15	169	50600	5	77	532	0	72	0	0.0	1.21	33	54	11	13.4
469	15	308	39700	3	64	581	5	27	0	0.0	1.00	32	60	18	13.4

470 rows × 15 columns

ابتدا اطلاعات و توضیحات آن را مشاهده و سپس به توضیح میپردازیم:

```
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Hydrogen              470 non-null    int64
 1   Oxigen                470 non-null    int64
 2   Nitrogen              470 non-null    int64
 3   Methane               470 non-null    int64
 4   CO                    470 non-null    int64
 5   CO2                   470 non-null    int64
 6   Ethylene              470 non-null    int64
 7   Ethane                470 non-null    int64
 8   Acethylene            470 non-null    int64
 9   DBDS                  470 non-null    float64
 10  Power factor          470 non-null    float64
 11  Interfacial V         470 non-null    int64
 12  Dielectric rigidity   470 non-null    int64
 13  Water content         470 non-null    int64
 14  Health index          470 non-null    float64
dtypes: float64(3), int64(12)
```

	Hydrogen	Oxigen	Nitrogen	Methane	CO	CO2	Ethylene	Ethane
count	470.000000	470.000000	470.000000	470.000000	470.000000	470.000000	470.000000	470.000000
mean	404.261702	8357.372340	47759.561702	79.695745	244.000000	1816.414894	162.923404	81.940426
std	2002.142678	14164.233283	13760.451816	489.320336	237.267485	2256.790519	1323.811504	342.573636
min	0.000000	57.000000	3600.000000	0.000000	10.000000	48.000000	0.000000	0.000000
25%	4.000000	496.000000	41700.000000	2.000000	66.000000	641.750000	0.000000	0.000000
50%	9.000000	3810.000000	49100.000000	3.000000	150.500000	1125.000000	3.000000	4.000000
75%	34.000000	14875.000000	55875.000000	7.000000	361.750000	2257.500000	6.000000	69.750000
max	23349.000000	249900.000000	85300.000000	7406.000000	1730.000000	24900.000000	16684.000000	5467.000000

Acethylene	DBDS	Power factor	Interfacial V	Dielectric rigidity	Water content	Health index
470.000000	470.000000	470.000000	470.000000	470.000000	470.000000	470.000000
91.491489	17.036596	1.849043	38.434043	53.495745	16.282979	27.504043
644.365828	46.735057	6.144009	6.178830	6.458906	17.115646	17.741458
0.000000	0.000000	0.050000	21.000000	27.000000	0.000000	13.400000
0.000000	0.000000	0.570000	32.000000	51.000000	5.000000	13.400000
0.000000	0.000000	1.000000	39.000000	54.000000	12.000000	13.400000
0.000000	2.000000	1.000000	44.000000	56.000000	21.000000	38.550000
9740.000000	227.000000	73.200000	57.000000	75.000000	183.000000	95.200000

توضیح هر بخش:

۱- این مجموعه داده دارای ۴۷۰ سمپل است که دارای ۱۵ ویژگی است.

۲- تمام ویژگی ها کامل اند و مقدار دارند.

۳- تمامی ویژگی ها دارای مقادیر عددی هستند و نیازی به تبدیل به مقادیر عددی در این دیتاست نداریم.

۴- توضیح ویژگی ها:

a. در روغن یکسری گازها در اثر عوامل مختلف به وجود می آیند و میتوانند و مواردی مانند ولتاژ شکست در روغن را تحت تاثیر قرار دهند. واحد این گاز ها ppm در واحد روغن است.

b. دی بنزیل دی سولفید (DBDS) رایج ترین گوگرد خورنده در روغن ترانسفورماتور است. با سیم‌پیچ‌های ترانسفورماتور واکنش نشان می‌دهد و سولفید مس (Cu2S) تولید می‌کند و بر روی سطح کاغذ عایق رسوب می‌کند که منجر به خطاهای وقفه‌ای در سیم‌پیچ‌های ترانسفورماتور می‌شود.

c. ضریب قدرت: ضریب توان پایین می تواند مشکلات سیستم عایق را نشان دهد.

d. ولتاژ رابط: نشان دهنده ولتاژ سطحی در ترانسفورماتور است .نظارت بر ولتاژ سطحی برای ارزیابی وضعیت روغن عایق مهم است.

e. سختی دی الکتریک: سفتی دی الکتریک روغن ترانسفورماتور را نشان می دهد .سفتی دی الکتریک معیاری برای سنجش توانایی روغن برای مقاومت در برابر استرس الکتریکی بدون شکستگی است.

f. محتوای آب :نشان دهنده غلظت آب در روغن ترانسفورماتور است .محتوای آب یک پارامتر حیاتی است، زیرا رطوبت بیش از حد می تواند خواص عایق روغن را کاهش دهد و منجر به خطا شود.

g. describe(). : به شرح زیر است:

i. شمردن :تعداد ورودی های غیر پوچ برای هر ویژگی را نشان می دهد. در این مورد، ۴۷۰ نمونه در مجموعه داده وجود دارد .

ii. متوسط: نشان دهنده مقدار متوسط برای هر ویژگی است. مثلا میانگین غلظت هیدروژن تقریباً ۴۰۴.۲۶ است.

iii. انحراف معیار: مثلا برای هیدروژن، انحراف معیار نسبتاً زیاد است (۲۰۰۲.۱۴)، که نشان دهنده تنوع گسترده در غلظت هیدروژن است .

iv. حداقل :حداقل مقدار را برای هر ویژگی نشان می دهد .مثال: حداقل غلظت هیدروژن ۰ است.

v. صدک: مقادیری را که درصد معینی از مشاهدات زیر آنها قرار می گیرد را مشخص کنید .مثال: صدک ۵۰ درصد (میانگین) هیدروژن ۹ است، یعنی ۵۰ درصد نمونه ها غلظت هیدروژن کمتر یا مساوی ۹ دارند.

vi. حداکثر: حداکثر مقدار را برای هر ویژگی نشان می دهد .مثال: حداکثر غلظت هیدروژن ۲۳۳۴۹ است.

همچنین:

۱. غلظت هیدروژن: میانگین غلظت هیدروژن در حدود ۴۰۴.۲۶ است، با یک انحراف معیار قابل توجه (۲۰۰۲.۱۴)، که نشان دهنده تنوع در داده ها است . حداقل غلظت هیدروژن ۰ و حداکثر ۲۳۳۴۹ است.

۲. اکسیژن، نیتروژن، متان، CO، CO2، اتیلن، اتان، استیلن: مشاهدات مشابهی را می توان برای این گازها انجام داد، با مقادیر میانگین، انحرافات معیار و محدوده بینش هایی را در مورد تغییرپذیری و توزیع هر غلظت گاز ارائه می دهد

۳. DBDS ضریب توان، Vرابط، سفتی دی الکتریک، محتوای آب: به نظر می رسد این ویژگی ها دارای تنوع کمتری هستند، با انحرافات معیار کوچکتر در مقایسه با غلظت گاز.

۴. شاخص سلامت: میانگین شاخص سلامت ۲۷.۵ با دامنه ۰ تا ۹۵.۲ است. **همین مورد توجیهی برای کلاس بندی نهایی است که چرا بیشتر داده ها در کلاس پنجم قرار خواهند گرفت.**

خروجی شاخص سلامت را مشاهده میکنیم:

```
print (y)
[13.4 13.4 60.5 13.4 13.4 13.4 13.4 13.4 13.4 38.3 13.4 48.2 13.4 26.7
 48.2 13.4 38.3 21.9 26.6 13.4 13.4 13.4 38.3 13.4 13.4 48.2 13.4 13.4
 49.9 13.4 48.2 13.4 13.4 13.4 13.5 13.4 38.3 13.4 13.4 13.4 13.4 13.4
 13.4 13.4 13.4 13.4 38.3 48.2 13.4 13.4 46.6 51.6 13.4 13.4 13.6 45.3
 13.4 51.5 48.2 13.4 13.4 13.4 13.4 49.2 13.4 13.4 48.9 38.3 85.3 13.4
 13.4 13.4 13.4 13.4 13.4 13.4 13.5 38.3 13.4 56.  13.4 13.4 13.8 48.2
 38.3 38.3 60.5 38.3 13.4 13.4 85.3 38.  48.3 13.4 38.3 48.2 38.3 50.7
 13.4 13.4 38.3 13.4 48.2 13.4 55.1 50.7 13.4 59.3 38.4 48.2 51.5 48.2
 13.4 48.5 13.4 48.2 13.4 48.2 13.4 48.2 50.  13.4 13.4 13.4 13.4 13.4
 13.4 13.4 13.4 38.3 13.4 48.2 38.3 22.5 13.4 26.7 26.7 13.4 13.4 13.4
 13.4 13.4 48.2 38.3 13.4 38.3 13.4 13.4 13.4 49.2 13.4 38.3 13.4 13.4
 48.2 13.4 13.4 13.4 26.7 48.2 13.4 38.3 60.5 13.4 50.6 48.2 50.7 13.4
 13.4 48.2 13.4 48.2 13.4 13.9 60.5 13.5 48.2 13.4 38.3 48.2 13.4 38.3
 38.3 13.7 38.3 48.2 38.3 13.4 13.4 13.4 21.6 13.4 13.4 13.4 13.4 60.5
 38.1 13.4 60.5 49.  13.4 13.4 13.4 50.7 36.6 31.  38.3 19.8 13.9 13.5
 13.4 13.4 48.2 13.4 38.3 38.3 47.9 13.4 13.4 38.3 48.2 13.4 48.2 13.4
 38.3 13.4 13.4 48.2 13.4 13.4 13.4 13.4 13.4 13.4 58.  13.4 13.4 13.4
 21.7 13.4 38.3 13.4 48.2 13.4 38.  48.5 13.4 13.4 13.4 13.5 13.4 13.4
 13.4 13.4 48.2 13.4 48.2 48.2 50.7 38.3 63.4 38.3 13.4 38.3 13.4 68.
 13.4 38.3 13.4 16.6 38.3 38.3 13.4 73.2 13.4 13.4 13.4 72.8 38.3 55.8
 13.4 48.2 13.4 13.4 38.3 50.6 19.5 85.5 13.4 13.4 13.4 13.4 48.2 13.4
 13.4 13.4 40.  50.6 38.3 13.4 38.6 13.4 48.2 13.4 48.2 13.4 13.4 13.4
 26.7 13.4 26.6 13.4 38.3 13.4 13.4 13.4 13.4 38.3 48.2 13.5 13.4 13.4
 48.2 13.4 13.4 48.2 13.4 36.4 48.2 50.3 75.6 60.5 13.4 13.4 48.2 38.3
 13.4 13.5 13.8 26.7 13.4 13.4 13.4 51.6 38.3 38.  48.2 26.6 17.5 38.3
 13.4 13.4 26.6 38.3 13.4 38.3 13.4 13.4 38.3 50.7 13.8 50.7 49.1 13.4
 13.4 13.4 13.4 50.7 13.4 13.4 13.4 38.3 38.3 48.2 13.4 13.4 13.4 48.2
 13.4 61.3 13.4 50.6 21.9 13.4 38.3 60.5 48.2 38.3 13.4 13.4 13.4 13.4
 13.4 38.3 51.5 13.4 85.2 60.5 13.6 13.4 13.4 13.4 38.3 13.5 13.4 13.4
 38.3 13.4 13.4 13.4 13.4 63.4 13.4 48.2 13.4 48.2 13.4 13.4 13.4 13.4
 26.7 13.4 13.4 13.4 26.7 13.4 38.3 13.4 50.7 23.  13.4 13.4 26.7 13.4
 48.1 33.6 38.3 13.4 13.4 26.7 50.7 13.4 75.6 13.4 13.4 48.2 95.2 13.4
 13.4 13.4 13.4 48.2 38.3 13.4 13.7 50.7 13.4 13.4 48.2 58.3 13.4 28.1
 13.4 48.2 13.4 16.2 48.2 57.4 26.7 38.3]
```

نیاز است تا این داده های پیوسته در چند دسته قرار بگیرند.

برای اینکار از جدولی که در سایت Kaggle در کنار همین دیتاست معرفی شده است استفاده میکنیم:

HEALTH INDEX AND TRANSFORMER EXPECTED LIFETIME

HI%	Condition	Expected Lifetime	Requirements
85 – 100	Very Good	More than 15 years	Normal maintenance
70 - 85	Good	More than 10 years	Normal maintenance
50 - 70	Fair	From 3 – 10 years	Increase diagnostic testing, possible remedial work or replacement needed depending on criticality
30 - 50	Poor	Less than 3 years	Start planning process to replace or rebuild considering risk and consequences of failure
0 - 30	Very Poor	Near to the end of life	Immediately assess risk; replace or rebuild based on assessment

در نهایت با توجه به این جدول اعداد را در ۵ کلاس قرار میدهیم:

```
def categorize_health_index(y):
  categories = []
  for value in y:
    if value >= 85:
      category = 1
    elif value >= 70:
      category = 2
    elif value >= 50:
      category = 3
    elif value >= 30:
      category = 4
    else:
      category = 5
    categories.append(category)
  return categories

y_categorized = categorize_health_index(y)
y = y_categorized
print(y)
```

خروجی به شکل زیر خواهد بود:

[5, 5, 3, 5, 5, 5, 5, 5, 5, 4, 5, 4, 5, 5, 4, 5, 4, 5, 5, 5, 5, 5, 4, 5, 5, 4, 5, 5, 4, 5, 4, 5, 5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4,
4, 5, 5, 4, 3, 5, 5, 5, 4, 5, 3, 4, 5, 5, 5, 5, 4, 5, 5, 4, 4, 1, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 3, 5, 5, 5, 4, 4, 4, 3, 4, 5, 5, 1, 4, 4, 5, 4,
4, 4, 3, 5, 5, 4, 5, 4, 5, 3, 3, 5, 3, 4, 4, 3, 4, 5, 4, 5, 4, 5, 4, 5, 4, 3, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4,
4, 5, 4, 5, 5, 5, 4, 5, 4, 5, 5, 4, 5, 5, 5, 5, 4, 5, 4, 3, 5, 3, 4, 3, 5, 5, 4, 5, 4, 5, 5, 3, 5, 4, 5, 4, 4, 5, 4, 4, 5, 4, 4, 4, 5, 5, 5, 5,
5, 5, 5, 5, 3, 4, 5, 3, 4, 5, 5, 5, 3, 4, 4, 4, 5, 5, 5, 5, 5, 4, 5, 4, 4, 4, 5, 5, 4, 4, 5, 4, 5, 4, 5, 5, 4, 5, 5, 5, 5, 5, 5, 3, 5, 5, 5, 5,
5, 4, 5, 4, 5, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 4, 4, 3, 4, 3, 4, 5, 4, 5, 3, 5, 4, 5, 5, 4, 4, 5, 2, 5, 5, 5, 2, 4, 3, 5, 4, 5, 5, 4, 3, 5,
1, 5, 5, 5, 5, 4, 5, 5, 5, 4, 3, 4, 5, 4, 5, 4, 5, 4, 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 5, 5, 4, 4, 5, 5, 5, 4, 5, 5, 4, 5, 4, 5, 4, 4, 3, 2, 3, 5, 5, 4,
4, 5, 5, 5, 5, 5, 5, 5, 3, 4, 4, 4, 5, 5, 4, 5, 5, 5, 4, 5, 4, 5, 5, 4, 3, 5, 3, 4, 5, 5, 5, 5, 3, 5, 5, 5, 4, 4, 4, 5, 5, 5, 4, 5, 3, 5, 3, 5,
5, 4, 3, 4, 4, 5, 5, 5, 5, 5, 4, 3, 5, 1, 3, 5, 5, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5, 5, 5, 3, 5, 4, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 4, 5, 3, 5, 5,
5, 5, 5, 4, 4, 4, 5, 5, 5, 3, 5, 2, 5, 5, 4, 1, 5, 5, 5, 5, 5, 4, 4, 5, 5, 3, 5, 5, 4, 3, 5, 5, 5, 4, 5, 5, 4, 3, 5, 4]

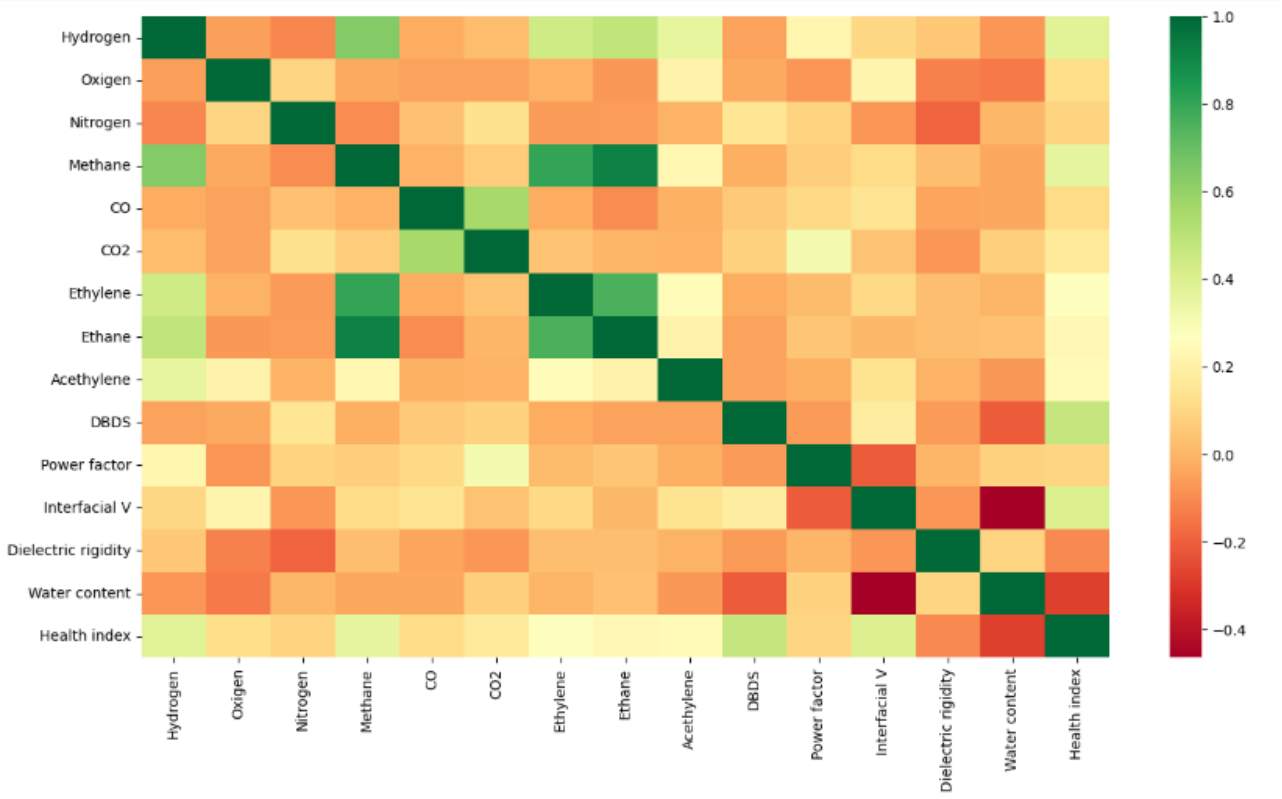
حال وابستگی خروجی به هر یک از ۱۴ ویژگی دیگر را بررسی میکنیم:

```
correlation_matrix = data.corr()['Health index'].sort_values(ascending
=False)
correlation_matrix
```

Health index	1.000000
DBDS	0.468809
Interfacial V	0.400216
Hydrogen	0.377388
Methane	0.361770
Ethylene	0.271504
Acethylene	0.240143
Ethane	0.236507
CO2	0.168777
Oxigen	0.121009
CO	0.112751
Power factor	0.092729
Nitrogen	0.089455
Dielectric rigidity	-0.104426
Water content	-0.281165

به صورت گرافیکی نیز نمایش میدهیم:

```
# Set up the matplotlib figure
plt.figure(figsize=(20, 20))

# Create a heatmap using seaborn
sns.heatmap(data.corr(), cmap="RdYlGn")

# Show the plot
plt.show()
```



در ادامه مسئله را به سه بخش تقسیم میکنیم:

۱- آموزش با استفاده از تمام ویژگی ها

۲- آموزش با استفاده از ویژگی هایی که همبستگی بیشتری با خروجی دارند.

۳- افزایش داده و تکرار بخش ۱

بخش اول: آموزش با تمام ویژگی ها

MLPClassifier()

ابتدا ویژگی ها را بین صفر و یک نگاشت میدهیم تا شبکه عصبی و تاثیر تغییر وزن ها در دقت شبکه افزایش یابد:

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
import warnings

scaler = MinMaxScaler()
X1 = scaler.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X1, y,
test_size=0.2,random_state=7)
print(X)
print(X1)
```

قبل از نگاشت:

```
[[3.6000e+01 2.4700e+02 5.4000e+04 ... 3.2000e+01 5.7000e+01 1.6000e+01]
 [4.0000e+01 2.6600e+02 5.6300e+04 ... 3.2000e+01 5.7000e+01 2.1000e+01]
 [2.3349e+04 2.4750e+03 2.8011e+04 ... 5.2000e+01 7.0000e+01 2.0000e+00]
 ...
 [8.7000e+01 3.6200e+02 7.6100e+04 ... 4.3000e+01 5.2000e+01 4.0000e+00]
 [1.4000e+01 2.6900e+03 1.0300e+04 ... 4.7000e+01 5.4000e+01 2.0000e+00]
  [1.2000e+01 1.5200e+04 6.2800e+04 ... 4.4000e+01 5.6000e+01 1.4000e+01]]
```

بعد از نگاشت:

```
[[1.54182192e-03 7.60477580e-04 6.16891065e-01 ... 3.05555556e-01
 6.25000000e-01 8.74316940e-02]
[1.71313547e-03 8.36525338e-04 6.45042840e-01 ... 3.05555556e-01
 6.25000000e-01 1.14754098e-01]
[1.00000000e+00 9.67807783e-03 2.98788250e-01 ... 8.61111111e-01
 8.95833333e-01 1.09289617e-02]
...
[3.72606964e-03 1.22076664e-03 8.87392901e-01 ... 6.11111111e-01
 5.20833333e-01 2.18579235e-02]
[5.99597413e-04 1.05386183e-02 8.20073439e-02 ... 7.22222222e-01
 5.62500000e-01 1.09289617e-02]
[5.13940640e-04 6.06100631e-02 7.24602203e-01 ... 6.38888889e-01
 6.04166667e-01 7.65027322e-02]]
```

با اینکار و در این روش، دقت از ۶۱ درصد به حدود ۷۵ درصد تغییر میکند. (نتایج ۶۱ درصد طی آموزش شبکه قبل از نگاشت به دست آمده بود)

پیدا کردن بهترین پارامترها برای کلاسیفیکیشن mlp از sklearn:

برای اینکار از یک حلقه های تو در تو استفاده میکنیم. از دستور آمده GridSearchCV برای اینکار کمک میگیریم. GridSearchCV تابعی است که توسط scikit-learn ارائه می‌شود که جستجوی جامع را روی یک شبکه پارامتر مشخص انجام می‌دهد و یک مدل را برای هر ترکیبی از پارامترها اعتبارسنجی و ارزیابی می‌کند. معمولاً برای تنظیم فراپارامترها استفاده میشود و به یافتن بهترین مجموعه فراپارامترها برای یک مدل یادگیری ماشینی معین کمک می کند.

```
warnings.filterwarnings("ignore")

param_grid = {
    'hidden_layer_sizes': [(10,10), (10,20), (10,30),(20,10), (20,20),
(20,20),(30,10), (30,10), (30,30)],  # Add more values if needed
    #'activation': ['logistic', 'relu','tanh'],  # Add more activation
functions if needed
    'max_iter': [200, 300, 400, 500],  # Add more values if needed
    #'solver': ['adam', 'sgd', 'lbfgs'],  # Add more solvers if needed
    #'batch_size': [5, 10, 20],  # Add more batch sizes if needed
}

mlp = MLPClassifier(solver='adam', alpha=0.0001, batch_size='auto',
learning_rate='constant', learning_rate_init=0.001,
                    power_t=0.5, shuffle=True, random_state=7, tol=0.0001,
verbose=False, warm_start=True,
                    momentum=0.9, nesterovs_momentum=True,
early_stopping=False, validation_fraction=0.1, beta_1=0.9,
                    beta_2=0.999, epsilon=1e-08, n_iter_no_change=10,
max_fun=15000)

grid_search = GridSearchCV(mlp, param_grid, cv=3, scoring='accuracy')
grid_search.fit(x_train, y_train)
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Evaluate the model on the test set
best_model = grid_search.best_estimator_
test_score = best_model.score(x_test, y_test)
print("Test Accuracy:", test_score)
```

یک دیکشنری تعریف میکنیم و فراپارامترهایی که میخواهیم پیدا کنیم به کمک خود سایت توضیحات آن، تشکیل میدهیم و اجرا میکنیم.

توضیح کامل نتایج:

۱- این بخش از دستور، بارها اجرا شده و هر بار زمان بالای ۲۰ دقیقه صرف میشود و نتیجه ای یافت نمیشد. برای همین یکسری از فراپارامترها همان طور که در کد مشخص است، به حالت کامنت در آمده اند و تنها برخی از آنها جست و جو شده اند. بقیه آن ها به صورت تجربی (تمارین قبل) و سعی و خطا به دست آمده اند.

۲- همین طور یک سری از فراپارامتر ها که در این کلاسیفیکیشن آمده وجود دارند اما در ادامه در آموزش استفاده نشده اند، به این دلیل است که با تغییر آنها تغییر در دقت به دست نیامد (مانند آلفا) و نهایتا به همین فراپارامترها بسنده شد.

۳- نتیجه کد بالا به صورت زیر است:

```
Best Parameters: {'hidden_layer_sizes': (20, 20), 'max_iter': 300}
Test Accuracy: 0.7127659574468085
```

اما به این نتایج بسنده نشد و به دلیل اینکه انتظار میرفت که همچنان دقت میتواند با تغییر همین پارامترها بیشتر شود مجدد به تغییر تعداد لایه ها و تعداد نورون ها (یک لایه پنهان، دو لایه پنهان و سه لایه پنهان) پرداخته شد و در نهایت شبکه به صورت زیر آموزش داده شد:

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(40,5), activation='tanh',
solver='adam',batch_size=20, learning_rate='constant',
learning_rate_init=0.001,max_iter=400,random_state=7)

model.fit(x_train, y_train)
model.score(x_test, y_test)
```

۴- batch_size به صورت سعی و خطا اعداد بین ۱۰ تا ۴۰ تست شده است و در نهایت بهترین عدد ۲۰ به دست آمده است. این عدد یعنی

۵- همین طور نرخ یادگیری اولیه از ۰.۰۱ تا ۰.۰۰۰۱ با گام های بزرگ در این بازه به صورت دستی تغییر یافتند و همان عدد پیش فرضی که از ابتدا بود یعنی ۰.۰۰۱ نهایتا انتخاب شد.

۶- activation='tanh': تابع فعال سازی خروجی هر نورون را در یک شبکه عصبی تعیین می کند. این به مدل کمک می کند تا الگوها و روابط پیچیده را در داده ها ثبت کند. **یک نتیجه به این صورت است**

که احتمالا با توجه به دیتاست که بازه هر ویژگی متفاوت و همین طور همسبتگی ویژگی ها همگی کمتر از ۰.۵ هستند، رابطه بین ورودی ها و خروجی یک رابطه پیچیده باشد و به همین دلیل انتظار میرفت که تابع فعال ساز tanh انتخاب شود.

۷- Adam حل کننده الگوریتم بهینه سازی است که برای تنظیم وزن شبکه عصبی در طول تمرین استفاده می شود. بهینه‌ساز Adam به دلیل نرخ یادگیری تطبیقی و به‌روزرسانی‌های مبتنی بر تکانه، انتخاب محبوبی است. در طیف وسیعی از سناریوها موثر است و اغلب به تنظیم کمتری در مقایسه با سایر بهینه‌سازها مانند نزول گرادیان تصادفی (SGD) نیاز دارد و **انتظار میرفت که به خودی خود چون پیشرفته تر محسوب میشود کارایی بهتری نیز روی این دیتاست داشته باشد.**

۸- نرخ یادگیری اولیه: این پارامتر نرخ یادگیری اولیه را برای الگوریتم بهینه سازی (برای Adam) تنظیم می کند. نرخ یادگیری اندازه مرحله به روز رسانی وزن را در طول تمرین کنترل می کند.

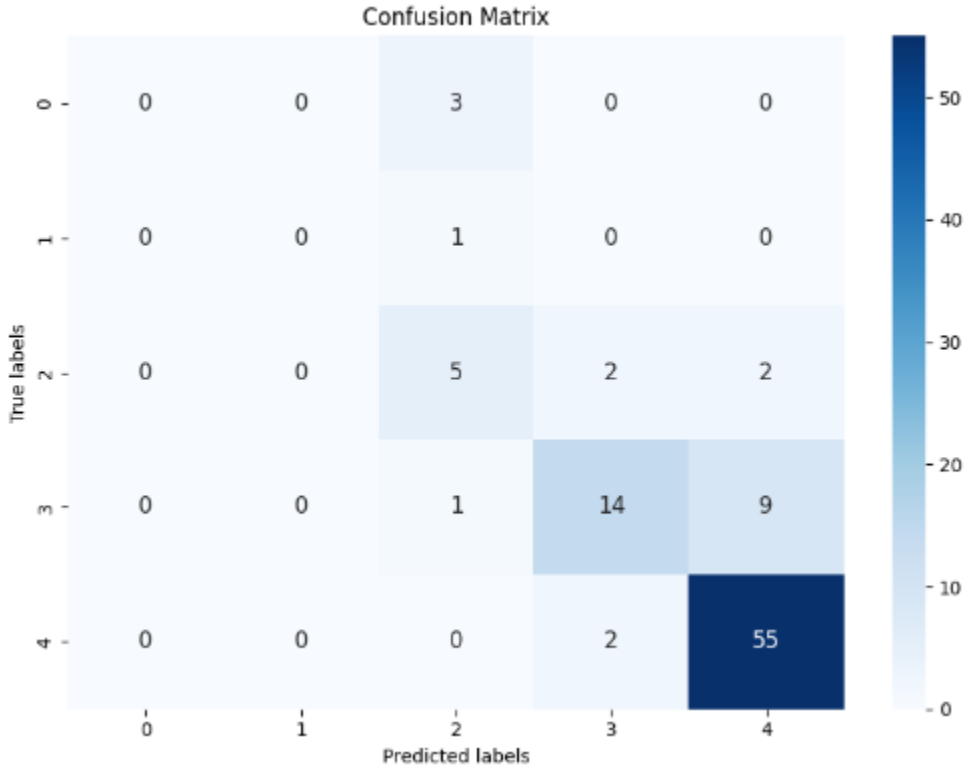
در نهایت نتیجه به صورت زیر حاصل میشود:

0.7872340425531915

ماتریس درهمریختگی را رسم میکنیم:

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Making predictions on the test set
y_pred = model.predict(x_test)

# Calculating confusion matrix
cf_matrix = confusion_matrix(y_test, y_pred)

# Plotting confusion matrix as a heatmap with fitted text
plt.figure(figsize=(8, 6))
sns.heatmap(cf_matrix, annot=True, fmt='d', cmap='Blues',
annot_kws={"size": 12})

# Get the axis to modify layout
```

```
plt.gca().set_ylim(len(np.unique(y_test)), 0)  # Fix for matplotlib 3.1.1
and 3.1.2
plt.title('Confusion Matrix')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')

# Save the plot as PNG
plt.tight_layout()
plt.savefig('confusion_matrix.png', dpi=300)
plt.show()

# Printing classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
1	0.00	0.00	0.00	3
2	0.00	0.00	0.00	1
3	0.50	0.56	0.53	9
4	0.78	0.58	0.67	24
5	0.83	0.96	0.89	57
accuracy			0.79	94
macro avg	0.42	0.42	0.42	94

```
weighted avg       0.75      0.79      0.76        94
```

همانطور که انتظار میرفت داده به دلیل اینکه در کلاس های انتهایی داده بیشتری وجود دارد برای همین شبکه بهتر برای آن کلاس ها آموزش دیده است و دقت بیشتر است.

LogisticRegression()

مجدد به پیدا کردن بهترین فراپارامترها میپردازیم:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
param_grid = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    #'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'max_iter': [50, 100, 200, 300],
}

# Create the Logistic Regression model
logistic_model = LogisticRegression()

# Create GridSearchCV
grid_search = GridSearchCV(logistic_model, param_grid, cv=3,
scoring='accuracy')

# Fit the model
grid_search.fit(x_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Evaluate the model on the test set
best_model = grid_search.best_estimator_
test_score = best_model.score(x_test, y_test)
print("Test Accuracy:", test_score)
```

```
Best Parameters: {'max_iter': 50, 'penalty': 'none'}
Test Accuracy: 0.7021276595744681
```

اینبار نیز به این دقت بسنده نمیکنیم و دیگر فراپارمترها را دوباره به صورت دستی آنقدر تغییر میدهیم تا به نتیجه بالاتری برسیم. در نهایت داریم:

```
model = LogisticRegression(solver='newton-cg',max_iter=۳00,
random_state=7,penalty='none')
model.fit(x_train, y_train)
model.predict(x_test)
```

```
model.score(x_test, y_test)
```

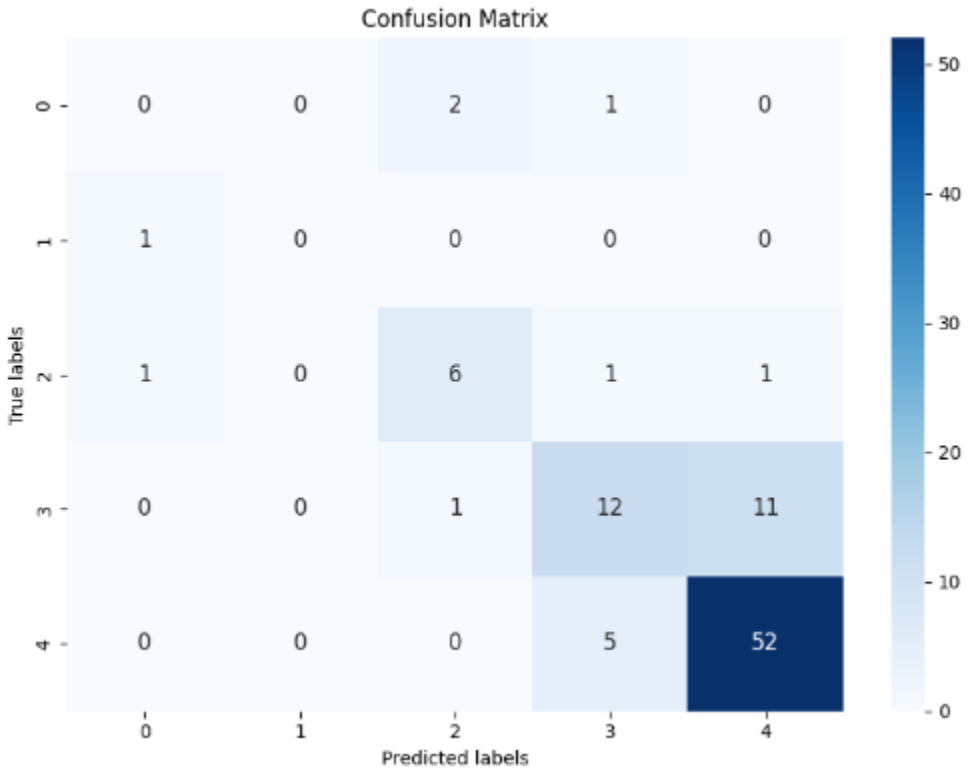
دقت : 0.7446808510638298

توضیح: در اینجا جز تعداد تکرار، تنها دو فراپارامتر تنظیم شده است:

۱- حل کننده newton-cg: حل کننده الگوریتم بهینه سازی است که برای یافتن وزن های بهینه در طول آموزش رگرسیون لجستیک استفاده می شود 'newton-cg' مخفف Newton Conjugate Gradient است. حل کننده Newton-CG از روش نیوتن با گرادیان مزدوج برای بهینه سازی وزن ها استفاده می کند. برای مشکلات با تعداد نسبتاً کم ویژگی مناسب است. روش نیوتن می‌تواند سریع‌تر از سایر حل‌کننده‌ها همگرا شود، اما ممکن است برای مجموعه داده‌های بزرگ از نظر محاسباتی گران باشد. **در نهایت میتوانستیم این طور پیش بینی کنیم که این روش احتمالا برای همین انتخاب میشود که مجموعه داده کوچکی داریم.**

۲- پنالتی = هیچی: عبارت پنالتی یک عبارت منظم سازی است که به تابع هدف رگرسیون لجستیک اضافه می شود تا از برازش بیش از حد جلوگیری شود. "هیچ" نشان می دهد که هیچ تنظیمی اعمال نشده است. در رگرسیون لجستیک، منظم سازی معمولاً برای جلوگیری از پیچیده شدن مدل و بیش از حد برازش داده های آموزشی معرفی می شود. اصطلاحات منظم سازی مانند L1 (کند) و L2 (ریج) وزنه های بزرگ را جریمه می کنند. با تنظیم penalty='none' ، ما به صراحت مشخص می‌کنیم که هیچ عبارت جریمه‌ای وجود ندارد و به مدل اجازه می‌دهد تا داده‌ها را بدون هیچ گونه منظم‌سازی برازش دهد. **علت اینکه این پنالتی برای شبکه به دست آمده است این است که به دلیل اینکه تعداد داده کم است و میخواهیم کمی شبکه تلاش بیشتری برای افزایش دقت و دسته بندی بهتری داشته باشد محدودیت کمتری به آن اعمال میکنیم.**

ماتریس در همریختگی را رسم میکنیم:




```
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         1
           3       0.67      0.67      0.67         9
           4       0.63      0.50      0.56        24
           5       0.81      0.91      0.86        57

    accuracy                           0.74        94
   macro avg       0.42      0.42      0.42        94
weighted avg       0.72      0.74      0.73        94
```

در مجموع دقت نسبت به حالت قبل کمتر شد ولی شبکه :

۱- هم در پیدا کردن فراپارامترها سریع تر عمل کرد و اصلا فراپارامتر کمتری داشتیم.

۲- شبکه سریع تر آموزش پیدا کرد.

۳- اما دقت نسبت به MLP کمتر بود.

DecisionTree

در این روش نیز ابتدا دوفراپارامترها تنظیم میکنیم:

```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

best_score = -np.inf
# Define the parameter grids
ccp_alpha_values = np.linspace(0, 0.005, num=100)
max_depth_values = range(1, 30)

for ccp_alpha in ccp_alpha_values:
    for max_depth in max_depth_values:
        reg = DecisionTreeClassifier(ccp_alpha=ccp_alpha,
max_depth=max_depth, random_state=7)
        reg.fit(x_train, y_train)
        score = reg.score(x_test, y_test)

        if score > best_score:
            best_score = score
            best_ccp_alpha = ccp_alpha
            best_max_depth = max_depth
```

بیشترین عمق و یک فراپارامتر هرس را انتخاب میکنیم و نتایج زیر حاصل میشود:

```
print(best_ccp_alpha)
print(best_max_depth)
```

```
0.004595959595959596
4
```

آموزش:

```
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier(ccp_alpha=best_ccp_alpha, criterion='gini',
max_depth=best_max_depth, random_state=7)
clf.fit(x_train, y_train)

y_hat = clf.predict(x_test)
accuracy = accuracy_score(y_test, y_hat)
```

```
print(f"Accuracy: {accuracy * 100:.2f}%")
```

دقت:

Accuracy: 84.04%

توضیح:

۱- ccp_alpha مخفف Cost-Complexity Pruning alpha است. این پارامتری است که بر روند هرس درخت تصمیم تاثیر می گذارد. یک ccp_alpha غیر صفر معیار پیچیدگی هزینه را برای هرس معرفی می کند. هدف الگوریتم درخت تصمیم، به حداقل رساندن ناخالصی کل (ناخالصی جینی، در این مورد) و هزینه درخت است، که در آن هزینه متناسب با تعداد گره ها است. پارامتر ccp_alpha تعادل بین به حداقل رساندن ناخالصی و به حداقل رساندن تعداد گره ها را کنترل می کند. ccp_alpha=0.00459595959595596 به این معنی است که فرآیند هرس شاخه‌ها را با معیار پیچیدگی هزینه بالاتر از این آستانه هرس می‌کند و منجر به درخت منظم‌تر می‌شود.

۲- max_depth حداکثر عمق درخت تصمیم است. حداکثر تعداد سطوح درخت را از گره ریشه تا گره های برگ کنترل می کند. تنظیم max_depth=4 به این معنی است که درخت تصمیم بیشتر از چهار سطح رشد نخواهد کرد. محدودیت در عمق با محدود کردن پیچیدگی درخت به جلوگیری از برازش بیش از حد کمک می کند.

۳- معیار تابعی است که برای اندازه گیری کیفیت تقسیم در درخت تصمیم استفاده می شود. 'gini' یکی از معیارهایی است که معمولاً استفاده می شود و به ناخالصی جینی اشاره دارد. ناخالصی جینی ناخالصی یا بی نظمی مجموعه ای از نقاط داده را اندازه گیری می کند. معیار آنتروپی نیز مورد بررسی قرار گرفت و نتیجه زیر حاصل شد:

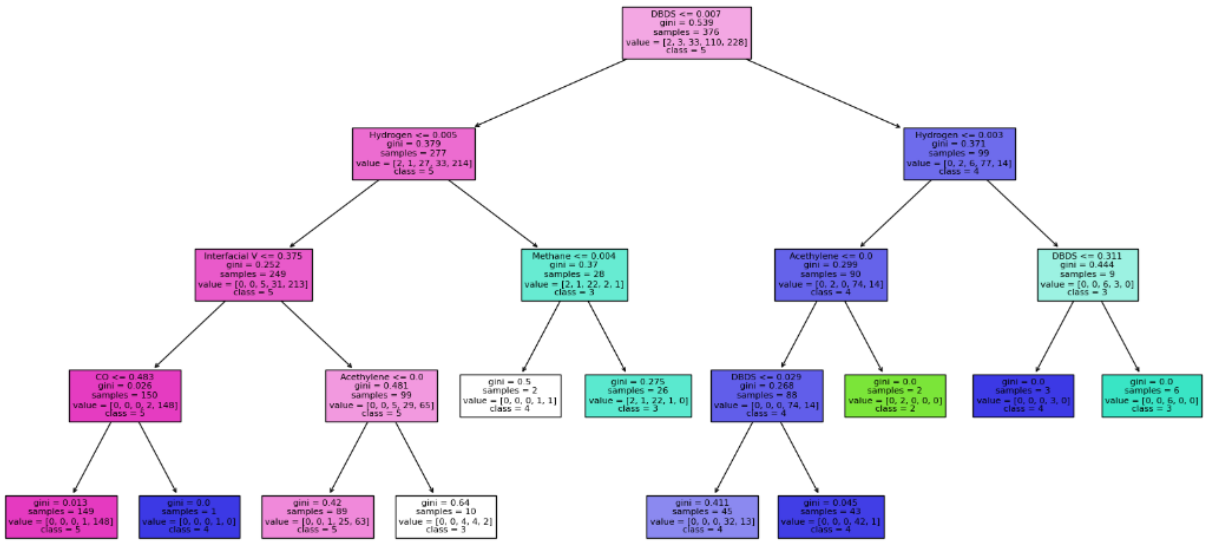
```
Accuracy: 79.79%
```

نهایتا همان gini انتخاب میشود.

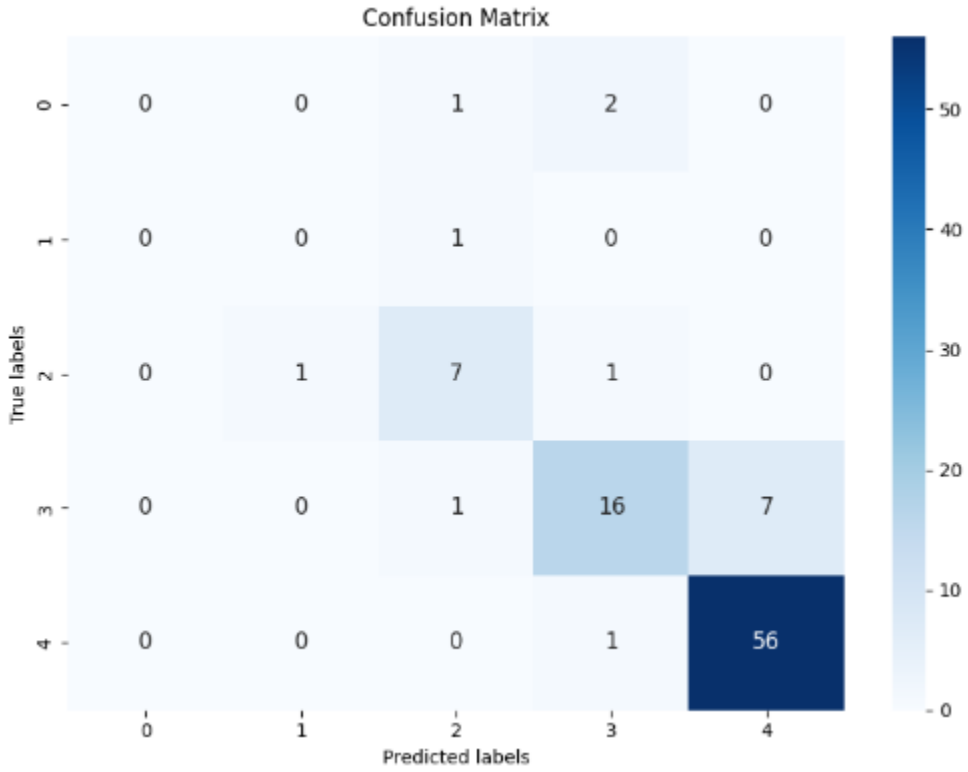
رسم درخت:

```
# Convert class names to strings
class_names = list(map(str, clf.classes_))
plt.figure(figsize=(20, 10))
```

```
tree.plot_tree(clf, filled=True, feature_names=X.columns,
class_names=class_names)
plt.show()
```



رسم نمودار درهمریختگی:



Classification Report:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	3
2	0.00	0.00	0.00	1
3	0.70	0.78	0.74	9
4	0.80	0.67	0.73	24
5	0.89	0.98	0.93	57
accuracy			0.84	94
macro avg	0.48	0.49	0.48	94
weighted avg	0.81	0.84	0.82	94

بخش دوم: آموزش با حذف برخی از ویژگی ها

با توجه به ماتریس همبستگی:

```
Health index        1.000000
DBDS                0.468809
Interfacial V       0.400216
Hydrogen            0.377388
Methane             0.361770
Ethylene            0.271504
Acethylene          0.240143
Ethane              0.236507
CO2                 0.168777
Oxigen              0.121009
CO                  0.112751
Power factor        0.092729
Nitrogen            0.089455
Dielectric rigidity -0.104426
Water content       -0.281165
Name: Health index, dtype: float64
```

به طور رندوم ویژگی های بین یک دهم تا منفی یک دهم را حذف میکنیم. به کمک هوش مصنوعی این گزارش داده شد که از صفر تا سه دهم همسبتگی خیلی ضعیف حساب میشود ولی به دلیل اینکه دیتاهای ما دارای همبستگی خیلی کم هستند و همین طور دیتاست کوچکی داریم نمیتوانیم ویژگی های زیادی را حذف کنیم **در نتیجه این بازه را انتخاب میکنیم چرا که بسیار به صفر نزدیک است:**

```
remove_features = correlation_matrix[(correlation_matrix > -0.1) &
(correlation_matrix < 0.1)].index

data_filtered = data.drop(columns=remove_features)

print(data_filtered)
correlation_matrix = data_filtered.corr()['Health
index'].sort_values(ascending=False)
data = data_filtered
```

```
     Hydrogen  Oxigen  Methane    CO    CO2  Ethylene  Ethane  Acethylene  \
253        36     247        4    58    776         4     157           0
243        40     266        4    86   1110         3     106           0
13      23349    2475     5045   156     48      5588    3532        2951
287        23   14800        1   449   2720         3       0           0
370         4    4990        7   760   2600         0       0           0
..        ...     ...      ...   ...    ...       ...     ...         ...
211        70     371        1    12   1750         7     117           0
67         25     508       12   317   2750         4      14           0
25         87     362       47   560   3920         5      31           0
196        14    2690        3   155    735         0       0           0
175        12   15200        4   413   4720         5       0           0

DBDS  Interfacial V  Dielectric rigidity  Water content  Health index
253    0.0             32                   57             16          13.4
```

```
243     0.0            32                57              21           13.4
13      0.0            52                70               2           60.5
287     0.0            32                57              89           13.4
370     0.0            33                56              11           13.4
..      ...           ...               ...             ...           ...
211     0.0            36                53              62           16.2
67      0.0            45                56               4           48.2
25    184.0            43                52               4           57.4
196     0.0            47                54               2           26.7
175     5.0            44                56              14           38.3
```

در نتیجه دیتاست بالا را مجدد با سه روش و اینبار با فراپارامترهای آماده آموزش میدهیم:

MLPClassifier()

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(40,5), activation='tanh',
solver='adam',
                      batch_size=20, learning_rate='constant',
learning_rate_init=0.001,
                      max_iter=400,random_state=7)

model.fit(x_train, y_train)
model.score(x_test, y_test)
```

دقت : 0.7978723404255319

```
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         1
           3       0.50      0.44      0.47         9
           4       0.76      0.67      0.71        24
           5       0.86      0.96      0.91        57

    accuracy                           0.80        94
   macro avg       0.42      0.42      0.42        94
weighted avg       0.76      0.80      0.78        94
```

نسبت به حالت قبل و همین روش، یک درصد دقت افزایش یافت. دقت کلاس ۳ کاهش یافت ولی دقت کلاس های ۴ و۵ افزایش یافت که دلیلش همان فراوانی داده در این کلاس هاست.

LogisticRegression()

```
model = LogisticRegression(solver='newton-cg',max_iter=300,
random_state=7,penalty='none')
model.fit(x_train, y_train)
model.predict(x_test)
model.score(x_test, y_test)
```

0.7446808510638298 : دقت

```
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         1
           3       0.67      0.67      0.67         9
           4       0.65      0.46      0.54        24
           5       0.80      0.93      0.86        57

    accuracy                           0.74        94
   macro avg       0.42      0.41      0.41        94
weighted avg       0.72      0.74      0.72        94
```

نسبت به حالت قبل خود هیچ تغییری نکرد.

DecisionTree

```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
clf = DecisionTreeClassifier(ccp_alpha=0.004595959595959596,
criterion='gini', max_depth=4, random_state=7)
clf.fit(x_train, y_train)

y_hat = clf.predict(x_test)
accuracy = accuracy_score(y_test, y_hat)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

Accuracy: 85.11%

```
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         1
           3       0.70      0.78      0.74         9
           4       0.81      0.71      0.76        24
           5       0.90      0.98      0.94        57

    accuracy                           0.85        94
   macro avg       0.48      0.49      0.49        94
weighted avg       0.82      0.85      0.83        94
```

دقت کل نسبت به حالت قبل همین روش ۱.۰۷ درصد افزایش یافت. نتایج برای کلاس های ۴ و۵ نتابج بالاتری به دست آمد.

بخش سوم: افزایش داده و تکرار بخش ۱

داده ها را به روش RandomOverSampler از imbalanced-learn افزایش میدهیم:

```
import pandas as pd
import numpy as np
from imblearn.over_sampling import RandomOverSampler

# Assuming 'data' is your DataFrame
# X contains the features, and y contains the target variable
X = data.drop('Health index', axis=1)
y = data['Health index']

def categorize_health_index(y):
    categories = []
    for value in y:
        if value >= 85:
            category = 1
        elif value >= 70:
            category = 2
        elif value >= 50:
            category = 3
        elif value >= 30:
            category = 4
        else:
            category = 5
        categories.append(category)
    return categories

y_categorized = categorize_health_index(y)
y = y_categorized

# Initialize RandomOverSampler
ros = RandomOverSampler(random_state=42)

# Fit and apply the resampling
X_resampled, y_resampled = ros.fit_resample(X, y)

# Convert y_resampled to a NumPy array or Pandas Series
y_resampled = np.array(y_resampled)
print("Shape of upsampled target variable:", y_resampled.shape)
df_resampled = pd.concat([pd.DataFrame(X_resampled, columns=X.columns),
pd.Series(y_resampled, name='Health index')], axis=1)

print("Shape of original dataset:", data.shape)
print("Shape of upsampled dataset:", df_resampled.shape)
```

output
Shape of upsampled target variable: (1425,)
Shape of original dataset: (470, 15)
Shape of upsampled dataset: (1425, 15)

تابعی که در کد بالا قرار دارد همان تبدیل پیوسته به گسسته خروجی است.

```
correlation_matrix = df_resampled.corr()['Health
index'].sort_values(ascending=False)
correlation_matrix
```

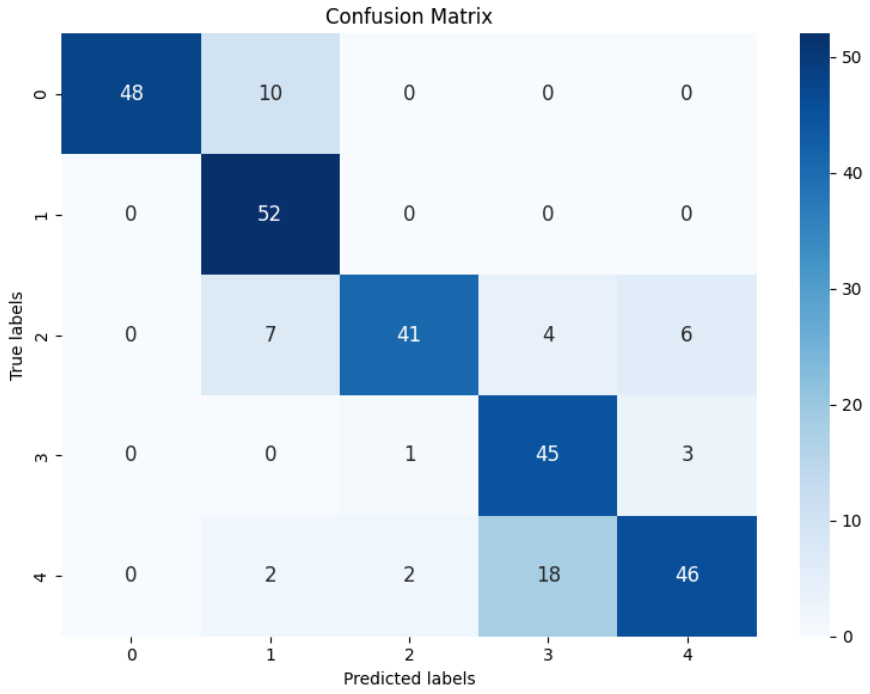
Health index 1.000000
Water content 0.374049
Nitrogen 0.205471
CO 0.091871
CO2 0.071363
Oxigen 0.036068
Power factor -0.034338
Dielectric rigidity -0.047644
Acethylene -0.125744
DBDS -0.151012
Interfacial V -0.224637
Ethylene -0.290199
Ethane -0.352239
Methane -0.418598
Hydrogen -0.433307
Name: Health index, dtype: float64

- ✓ **ابتدای شکل ها و نتایج هر سه روش قرار داده میشود و در انتهای گزارش، توضیح داده میشود.**
- ✓ **برای ارزیابی دیتاست جدید باید ابتدا شبکه را آموزش دهیم و چهارشاخص ارزیابی را بررسی کنیم.**
- ✓ **برای هر سه روش جز درخت تصمیم، از همان پارامترهایی که در بخش ۱ تنظیم کرده ایم استفاده میکنیم.**

MLPClassifier

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(40,5), activation='tanh',
solver='adam',
                      batch_size=20, learning_rate='constant',
learning_rate_init=0.001,
                      max_iter=400,random_state=7)

model.fit(x_train, y_train)
model.score(x_test, y_test)
```

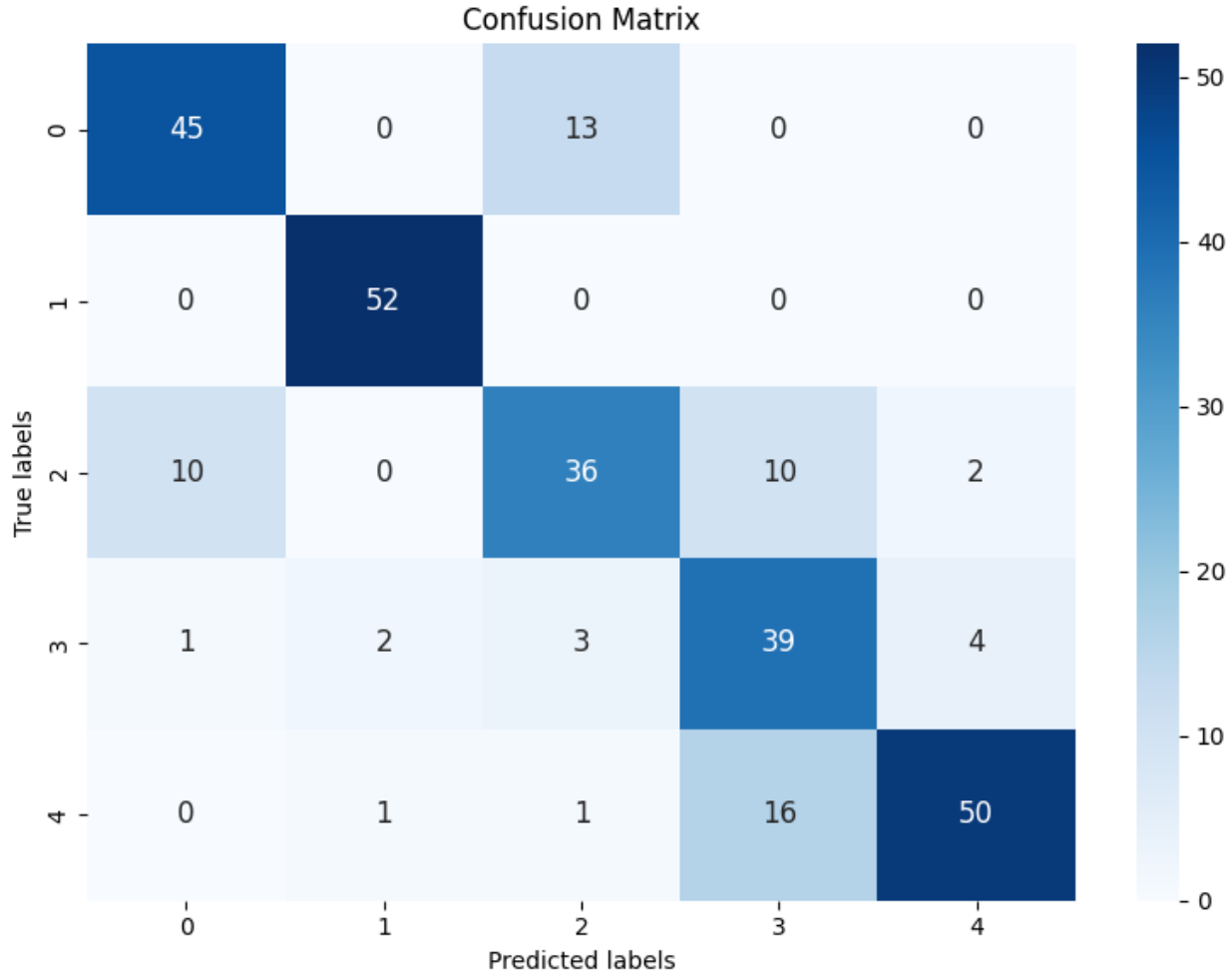
	precision	recall	f1-score	support
1	1.00	0.83	0.91	58
2	0.73	1.00	0.85	52
3	0.93	0.71	0.80	58
4	0.67	0.92	0.78	49
5	0.84	0.68	0.75	68
accuracy			0.81	285
macro avg	0.83	0.83	0.82	285
weighted avg	0.84	0.81	0.81	285



LogisticRegression

```
model = LogisticRegression(solver='newton-cg',max_iter=300,
random_state=7,penalty='none')
model.fit(x_train, y_train)
model.predict(x_test)
model.score(x_test, y_test)
```

	precision	recall	f1-score	support
1	0.80	0.78	0.79	58
2	0.95	1.00	0.97	52
3	0.68	0.62	0.65	58
4	0.60	0.80	0.68	49
5	0.89	0.74	0.81	68
accuracy			0.78	285
macro avg	0.78	0.79	0.78	285
weighted avg	0.79	0.78	0.78	285



DecisionTree

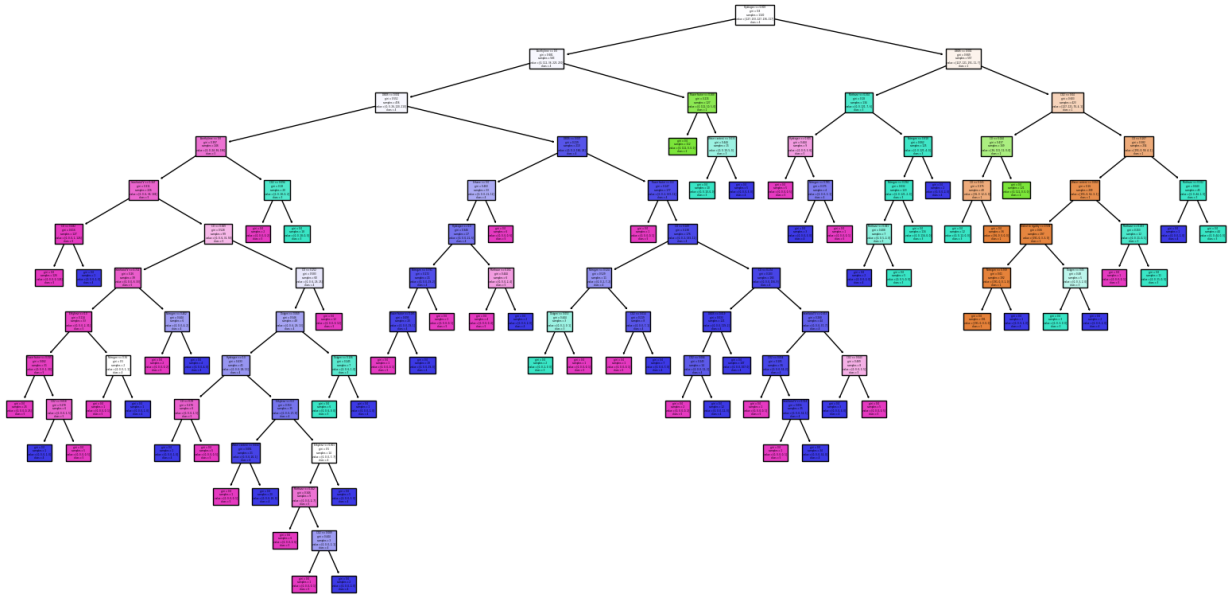
```
ccp_alpha_values = np.linspace(0, 0.2, num=100)
max_depth_values = range(1, 30)

for ccp_alpha in ccp_alpha_values:
    for max_depth in max_depth_values:
        reg = DecisionTreeClassifier(ccp_alpha=ccp_alpha,
max_depth=max_depth, random_state=7)
        reg.fit(x_train, y_train)
        score = reg.score(x_test, y_test)

        if score > best_score:
            best_score = score
            best_ccp_alpha = ccp_alpha
            best_max_depth = max_depth
```

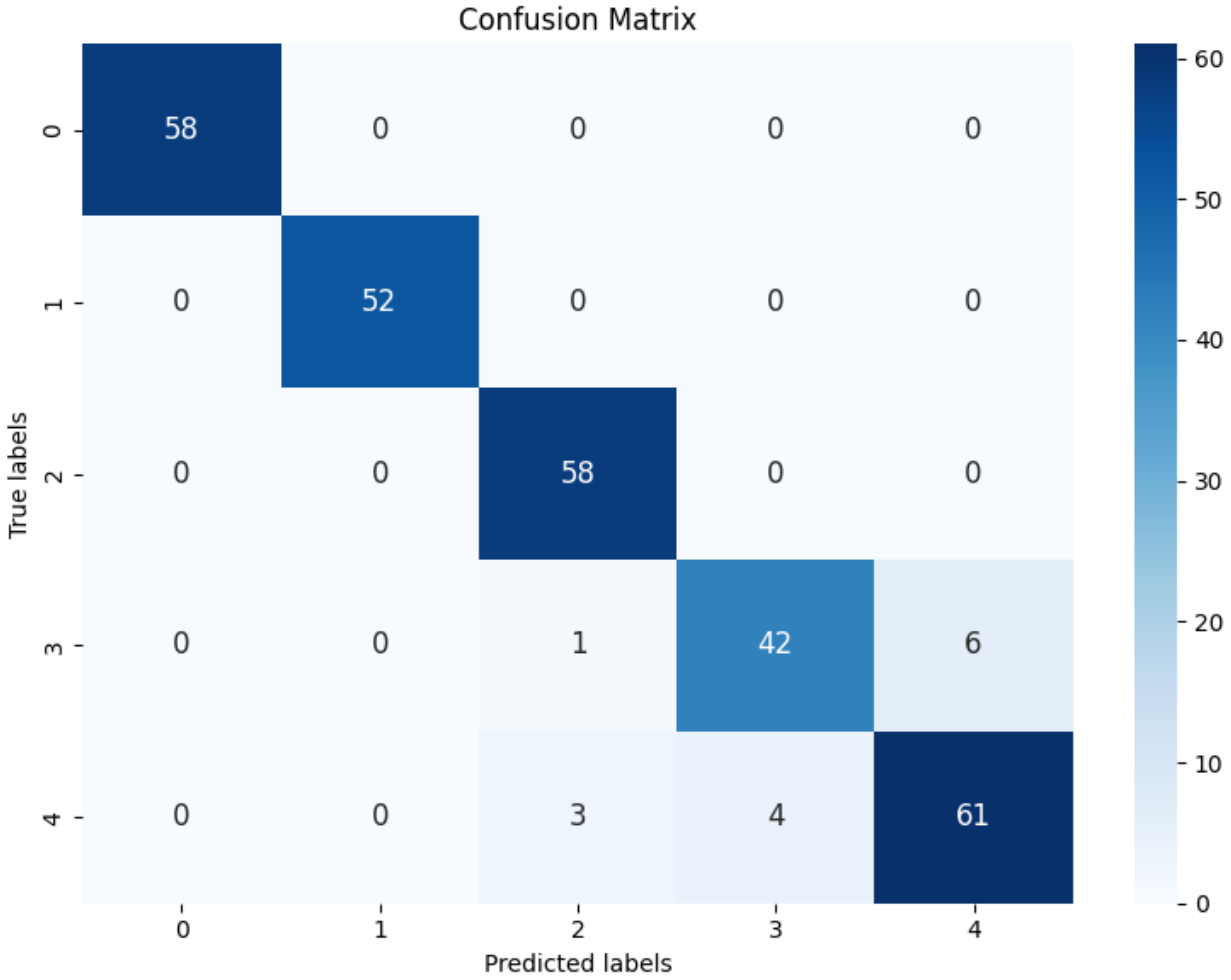
```
print(best_ccp_alpha)
print(best_max_depth)
```

```
0.0
13
```

```
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier(ccp_alpha=best_ccp_alpha, criterion='gini',
max_depth=best_max_depth, random_state=7)
clf.fit(x_train, y_train)

y_hat = clf.predict(x_test)
accuracy = accuracy_score(y_test, y_hat)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        58
           2       1.00      1.00      1.00        52
           3       0.94      1.00      0.97        58
           4       0.91      0.86      0.88        49
           5       0.91      0.90      0.90        68

    accuracy                           0.95       285
   macro avg       0.95      0.95      0.95       285
weighted avg       0.95      0.95      0.95       285
```




Confusion Matrix
True labels
Predicted labels
0
1
2
3
4
58 0 0 0 0
0 52 0 0 0
0 0 58 0 0
0 0 1 42 6
0 0 3 4 61
60
50
40
30
20
10
0

توضیح نتایج:

روشی که برای افزایش داده ها استفاده شد، «نمونه برداری بیش از حد» نامیده می شود. نمونه برداری بیش از حد شامل ایجاد نمونه های اضافی از کلاس اقلیت برای متعادل کردن توزیع کلاس در مجموعه داده شما است. روش‌های مختلفی برای نمونه‌گیری بیش‌ازحد وجود دارد، و روشی که به‌طور خاص استفاده کردیم، به‌عنوان «نمونه‌برداری تصادفی» شناخته می‌شود .در این مورد ، از RandomOverSampler ازکتابخانه imbalanced-learn استفاده کرده ایم، که به طور تصادفی نمونه هایی از کلاس اقلیت را برای متعادل کردن توزیع کلاس کپی می کند. این روش معمولاً برای رسیدگی به مسائل عدم تعادل کلاس در مسائل طبقه بندی استفاده می شود.

برای بررسی نتایج با توجه به مرور مفهوم شاخص های زیر توضیح میدهیم:

دقت (Accuracy): نسبت داده‌های درست دسته‌بندی‌شده به کل داده‌ها.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$$

حساسیت (Sensitivity یا Recall): نسبت داده‌های مثبتی که به درستی تشخیص داده‌شده‌اند به تعداد کل داده‌های مثبت واقعی.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

دقت مثبت (Precision): نسبت داده‌های مثبتی که به درستی تشخیص داده‌شده‌اند به تعداد کل داده‌های مثبت تشخیص داده‌شده.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$$

ارزیابی F1: یک معیار ترکیبی از دقت و حساسیت است.

$$F1 = \frac{2 \times \text{Precision} \times \text{Sensitivity}}{\text{Precision}+\text{Sensitivity}}$$

دقت در هر سه روش به ترتیب درخت تصمیم و از آن کمتر `mlp` و از آن کمتر رگرسیون لجستیک به میزان قابل قبولی افزایش یافت

در مورد سه شاخص دیگر نیز ، تا قبل از آن کلاس های ۱ و ۲و ۳ دارای داده بسیار کمی بودند و دقت مناسبی به دست نمیامد ولی با افزایش داده های این کلاس ها هر سه شاخص دارای عددی قابل قبول و معقول تر از قبل شدند و یک یادگیری ماشین بهتری اتفاق افتاد.

آدرس کدها در گیت هاب:

https://github.com/AliKhodarahmy/machinelearning2023/tree/main/Finalproject