



دانشکده مهندسی برق

دوره کارشناسی ارشد مهندسی برق-کنترل

مینیپروژه سوم درس یادگیری ماشین

توسط:

محمد رضا امانی - احمد رضا طاهری

استاد راهنما:

دکتر مهدی علیاری شوره‌دلی

۱۴۰۴ بهار

Code:

[Question 1](#)

[Question 2](#)

الله اکبر

فهرست مطالب

صفحه	عنوان
	فصل ۱ SVM
۱	- ریاضیات
۲	- دیتاست
۲	- دانلود و بررسی ویژگی دیتاست
۳	- خواندن دیتاست و تحلیل اولیه
۶	- بررسی و مدیریت مقادیر گمشده
۹	- داده‌های categorical
۱۰	- داده‌های پرت
۱۲	- دسته‌بندی استاندارد غلظت PM2.5
۱۳	- ایجاد ویژگی‌های جدید
۲۰	- مصورسازی داده‌ها
۲۱	- بررسی توزیع داده‌ها
۲۲	- نرمالسازی داده‌ها
۲۲	- تقسیم داده‌ها به مجموعه‌های آموزش و آزمون و ارزیابی
۲۳	- مدلسازی با استفاده از SVM برای دسته‌بندی
۳۲	- مدلسازی با استفاده از SVM برای پیش‌بینی
۳۳	- معرفی الگوریتم PSO
۳۴	- پیاده‌سازی SVM با PSO
۳۶	- بررسی و پیاده‌سازی مقاله
۳۶	- عنوان مقاله
۳۶	- دیتاست استفاده شده
۳۶	- مدل استفاده شده
۳۷	- نتایج
۳۷	- پیاده‌سازی
۳۸	- محصول نهایی
۳۹	- بخش امتیازی (تحقیقاتی)
۴۱	- کاهش ابعاد
۴۱	- سوالات

٤١	(ف)
٤٢	(ب)
٤٣	(ج)
٤٤	(د)
٤٤	(ه)
٤٥	(و)
٤٦	(ز)
٤٨	- ٢-٢ امتيازى

SVM - فصل ١

١-١ رياضيات

```
X = np.array([
    [1, -1, 1],
    [-3, 1, 1],
    [-3, 1, -1],
    [1, 2, 1],
    [-1, -1, 2]
], dtype=float)

y = np.array([-1, 1, -1, -1, 1], dtype=float)

n_samples = X.shape[0]

K = np.dot(X, X.T)
Q = np.outer(y, y) * K

P = matrix(Q)
q = matrix(-np.ones(n_samples))
G = matrix(-np.eye(n_samples))
h = matrix(np.zeros(n_samples))
A = matrix(y.reshape(1, -1))
b = matrix(np.zeros(1))

sol = solvers.qp(P, q, G, h, A, b)
alphas = np.ravel(sol['x'])

support_vectors_idx = np.where(alphas > 1e-5)[0]
alphas_sv = alphas[support_vectors_idx]
X_sv = X[support_vectors_idx]
y_sv = y[support_vectors_idx]

w = np.sum(alphas_sv[:, None] * y_sv[:, None] * X_sv, axis=0)
b = np.mean([y_sv[i] - np.dot(w, X_sv[i]) for i in range(len(X_sv))])

print("w:", w)
print("b:", b)
```

```
w: [-5.00000183e-01 -2.77304760e-07  1.00000005e+00]
b: -1.500001123456355
```

۱-۲- دیتاست

۱-۲-۱- دانلود و بررسی ویژگی دیتاست

نام ویژگی	نوع داده	توضیح	واحد
No	عدد صحيح (int)	شماره‌ی ردیف داده	ندارد
year	عدد صحيح	سال ثبت داده	ندارد
month	عدد صحيح	ماه ثبت داده (۱ تا ۱۲)	ندارد
day	عدد صحيح	روز ماه	ندارد
hour	عدد صحيح	ساعت روز (۰ تا ۲۴)	ساعت
pm2.5	عدد اعشاری (float)	غلظت ذرات معلق، ریز (کمتر از ۰.۰۳ میکرومتر/m³)	میکروگرم بر مترمکعب ($\mu\text{g}/\text{m}^3$)
DEWP	عدد صحيح	نقطه شبنم	درجة سلسیوس (C°)
TEMP	عدد اعشاری	دمای هوا	درجة سلسیوس (C°)
PRES	عدد اعشاری	فشار هوا	هکتوپاسکال (hPa)
cwd	متّی (object)	جهات جغرافیایی (شمال، شرق، غرب، جنوب)	NW, NE, SE, SW (cv)
Iws	عدد اعشاری	سرعت تجمعی باد	متر بر ثانیه (m/s) یا مشابه
Is	عدد صحيح	cumulated hours of snow	ساعت
Ir	عدد صحيح	cumulated hours of rain	ساعت

ویژگی‌های مرتبط با آلودگی هوا (PM2.5) و (TEMP)

دما می‌تواند بر پایداری هوا تأثیر بگذارد.

هوای گرم‌تر ممکن است باعث تشدید واکنش‌های شیمیایی شود.

DEWP (نقطه شبنم):

ارتباط با رطوبت دارد.

رطوبت زیاد می‌تواند سبب ماندگاری بیشتر آلاینده‌ها شود.

PRES (فشار هوا):

فشار بالا معمولاً باعث سکون هوا می‌شود → افزایش آلودگی.

cbwd (جهت باد) و **lws** (سرعت باد):

باد قوی باعث پراکندگی آلاینده‌ها می‌شود.

جهت باد تعیین می‌کند آیا آلودگی از مناطق صنعتی وارد می‌شود یا خیر.

ls, Ir (بارش برف/باران):

بارش می‌تواند ذرات را از هوا بشوید → کاهش PM2.5

۱-۲-۲ - خواندن دیتاست و تحلیل اولیه

فایل CSV را بارگذاری کرده و در قالب DataFrame نمایش می‌دهیم:

data = pd.read_csv('PRSA_data_2010.1.1-2014.12.31.csv')														
data														
✓	0.0s	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	lws	ls	Ir
0	1	2010	1	1	0	Nan	-21	-11.0	1021.0	NW	1.79	0	0	
1	2	2010	1	1	1	Nan	-21	-12.0	1020.0	NW	4.92	0	0	
2	3	2010	1	1	2	Nan	-21	-11.0	1019.0	NW	6.71	0	0	
3	4	2010	1	1	3	Nan	-21	-14.0	1019.0	NW	9.84	0	0	
4	5	2010	1	1	4	Nan	-20	-12.0	1018.0	NW	12.97	0	0	
...	
43819	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	231.97	0	0	
43820	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	237.78	0	0	
43821	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	242.70	0	0	
43822	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	246.72	0	0	
43823	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	249.85	0	0	
43824 rows × 13 columns														

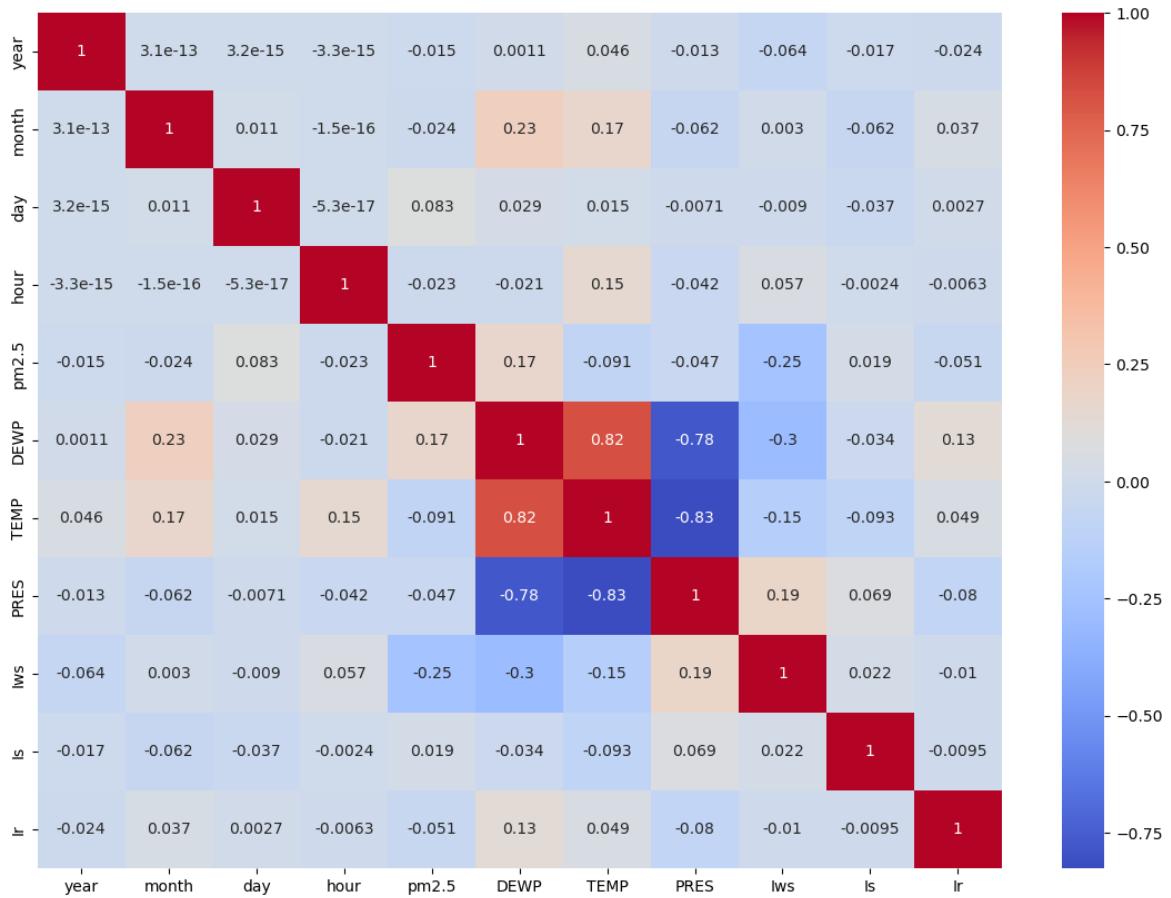
مشاهده می‌شود که ویژگی pm2.5 دارای مقادیر NaN است. نوع دادگان موجود در دیتاست به شرح زیر است:

Variable Name	Role	Type	Description	Units	Missing Values
No	ID	Integer			no
year	Feature	Integer			no
month	Feature	Integer			no
day	Feature	Integer			no
hour	Feature	Integer			no
pm2.5	Target	Integer			yes
DEWP	Feature	Integer			no
TEMP	Feature	Integer			no
PRES	Feature	Integer			no
cbwd	Feature	Categorical			no

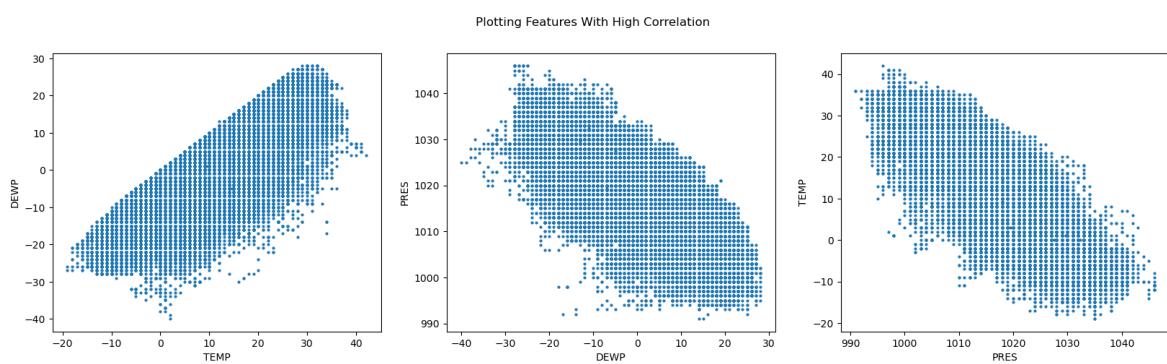
هر چند که به نظر می‌رسد ویژگی‌های PRES و TEMP همگی دارای مقادیر گستره هستند.
با اعمال متدهای describe بر روی دادگان، دید اولیه‌ای به آن بدست می‌آوریم:

	data.describe()												
	0.0s												
	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	Iws	Is	Ir	
count	43824.000000	43824.000000	43824.000000	43824.000000	43824.000000	41757.000000	43824.000000	43824.000000	43824.000000	43824.000000	43824.000000	43824.000000	
mean	21912.500000	2012.000000	6.523549	15.727820	11.500000	98.613215	1.817246	12.448521	1016.447654	23.889140	0.052734	0.194916	
std	12651.043435	1.413842	3.448572	8.799425	6.922266	92.050387	14.433440	12.198613	10.268698	50.010635	0.760375	1.415867	
min	1.000000	2010.000000	1.000000	1.000000	0.000000	0.000000	-40.000000	-19.000000	991.000000	0.450000	0.000000	0.000000	
25%	10956.750000	2011.000000	4.000000	8.000000	5.750000	29.000000	-10.000000	2.000000	1008.000000	1.790000	0.000000	0.000000	
50%	21912.500000	2012.000000	7.000000	16.000000	11.500000	72.000000	2.000000	14.000000	1016.000000	5.370000	0.000000	0.000000	
75%	32868.250000	2013.000000	10.000000	23.000000	17.250000	137.000000	15.000000	23.000000	1025.000000	21.910000	0.000000	0.000000	
max	43824.000000	2014.000000	12.000000	31.000000	23.000000	994.000000	28.000000	42.000000	1046.000000	585.600000	27.000000	36.000000	

ماتریس correlation ویژگی‌های عددی دیتابست به شرح زیر است:

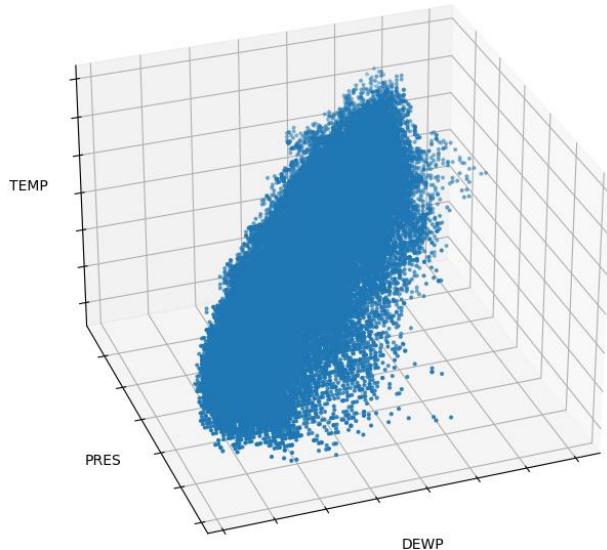


مشاهده می‌شود ویژگی‌های PRES، TEMP و DEWP با یکدیگر ارتباط بالایی دارند. حال این سه ویژگی را دو به دو ترسیم می‌کنیم:



ترسیم هر سه ویژگی با هم به صورت سه بعدی

3D Plot Of Features With High Correlation



به نظر می‌رسد که این ویژگی‌ها با یکدیگر ارتباط خطی معناداری دارند.

۱-۲-۳ - پررسی و مدیریت مقادیر گمشده

با استفاده از دستور `data.isna().sum()` تعداد مقادیر NaN در دیتاست را بدست می‌آوریم:

	data.isna().sum()
	✓ 0.0s
No	0
year	0
month	0
day	0
hour	0
pm2.5	2067
DEWP	0
TEMP	0
PRES	0
cbwd	0
Iws	0
Is	0
Ir	0
dtype:	int64

مشاهده می‌شود که تنها ویژگی pm2.5 دارای مقادیر NaN است. این مقادیر به دو صورت در دیتاست پخش شده‌اند. بخش اول شامل ۲۴ ردیف اول دیتا است که تمامی این ۲۴ نمونه به صورت پشت هم مقدار NaN دارند. بخش دوم شامل مقادیری است که در میان دادگان با مقدار قرار دارند.

ابتدا ۲۴ نمونه اول را حذف می‌کنیم:

data.iloc[0:25]													
✓ 0.0s													
No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	lws	ls	lr	
0	1	2010	1	1	0	NaN	-21	-11.0	1021.0	NW	1.79	0	0
1	2	2010	1	1	1	NaN	-21	-12.0	1020.0	NW	4.92	0	0
2	3	2010	1	1	2	NaN	-21	-11.0	1019.0	NW	6.71	0	0
3	4	2010	1	1	3	NaN	-21	-14.0	1019.0	NW	9.84	0	0
4	5	2010	1	1	4	NaN	-20	-12.0	1018.0	NW	12.97	0	0
5	6	2010	1	1	5	NaN	-19	-10.0	1017.0	NW	16.10	0	0
6	7	2010	1	1	6	NaN	-19	-9.0	1017.0	NW	19.23	0	0
7	8	2010	1	1	7	NaN	-19	-9.0	1017.0	NW	21.02	0	0
8	9	2010	1	1	8	NaN	-19	-9.0	1017.0	NW	24.15	0	0
9	10	2010	1	1	9	NaN	-20	-8.0	1017.0	NW	27.28	0	0
10	11	2010	1	1	10	NaN	-19	-7.0	1017.0	NW	31.30	0	0
11	12	2010	1	1	11	NaN	-18	-5.0	1017.0	NW	34.43	0	0
12	13	2010	1	1	12	NaN	-19	-5.0	1015.0	NW	37.56	0	0
13	14	2010	1	1	13	NaN	-18	-3.0	1015.0	NW	40.69	0	0
14	15	2010	1	1	14	NaN	-18	-2.0	1014.0	NW	43.82	0	0
15	16	2010	1	1	15	NaN	-18	-1.0	1014.0	cv	0.89	0	0
16	17	2010	1	1	16	NaN	-19	-2.0	1015.0	NW	1.79	0	0
17	18	2010	1	1	17	NaN	-18	-3.0	1015.0	NW	2.68	0	0
18	19	2010	1	1	18	NaN	-18	-5.0	1016.0	NE	1.79	0	0
19	20	2010	1	1	19	NaN	-17	-4.0	1017.0	NW	1.79	0	0
20	21	2010	1	1	20	NaN	-17	-5.0	1017.0	cv	0.89	0	0
21	22	2010	1	1	21	NaN	-17	-5.0	1018.0	NW	1.79	0	0
22	23	2010	1	1	22	NaN	-17	-5.0	1018.0	NW	2.68	0	0
23	24	2010	1	1	23	NaN	-17	-5.0	1020.0	cv	0.89	0	0
24	25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0

data=data.iloc[24:].reset_index(drop=True)													
data													
✓ 0.0s													
No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	lws	ls	lr	
0	25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0
1	26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	2.68	0	0
2	27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	3.57	0	0
3	28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	5.36	1	0
4	29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	6.25	2	0
...
43795	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	231.97	0	0
43796	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	237.78	0	0
43797	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	242.70	0	0
43798	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	246.72	0	0
43799	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	249.85	0	0

43800 rows × 13 columns

روش‌های مختلفی برای مدیریت داده‌های NaN وجود دارد:

- حذف سطرها: اگر تعداد سطرهای گمشده کم باشد.

- میانگین‌گیری (**Mean Imputation**): جایگزینی با میانگین — ساده ولی ممکن است واریانس را کاهش دهد.

- میانگین متحرک (**Rolling Mean**): مناسب برای داده‌های زمانی.

- درون‌یابی (**Interpolation**): روش توصیه شده برای داده‌های زمانی، مخصوصاً اگر داده‌ها پشت سرهم و مرتب باشند.

از روش حذف سطراها برای ۲۴ داده اول استفاده کردیم. سه روش میانگین‌گیری، میانگین متحرک و درون‌یابی به دلیل اینکه مقادیر ویژگی pm2.5، گستته هستند، برای این ویژگی مناسب نیستند. بنابراین برای پر کردن باقی مقادیر از روش backward fill یا forward fill استفاده می‌کنیم. در این روش‌ها دادگان بدون مقدار به ترتیب با مقادیر نمونه قبلی یا بعدی پر می‌شوند.

```
data['pm2.5'] = data['pm2.5'].fillna(method='ffill')
data
```

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
0	25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0
1	26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	2.68	0	0
2	27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	3.57	0	0
3	28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	5.36	1	0
4	29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	6.25	2	0
..
43795	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	231.97	0	0
43796	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	237.78	0	0
43797	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	242.70	0	0
43798	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	246.72	0	0
43799	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	249.85	0	0

43800 rows × 13 columns

در نهایت تمامی نمونه‌های با مقادیر NaN مدیریت می‌شوند:

```
data.isna().sum()
```

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0
..
43795	0	0	0	0	0	0	0	0	0	0	0	0	0
43796	0	0	0	0	0	0	0	0	0	0	0	0	0
43797	0	0	0	0	0	0	0	0	0	0	0	0	0
43798	0	0	0	0	0	0	0	0	0	0	0	0	0
43799	0	0	0	0	0	0	0	0	0	0	0	0	0

dtype: int64

ویژگی cbwd در این دیتاست دارای مقادیر categorical است.

```
data['cbwd'].value_counts()
0.0s
cbwd
SE    15290
NW    14150
cv     9387
NE     4997
Name: count, dtype: int64
```

دو روش رایج تبدیل ویژگی‌های دسته‌ای به مقادیر عددی:

:Label Encoding

هر مقدار متنی را به یک عدد صحیح منحصر به‌فرد تبدیل می‌کند.

مناسب زمانی است که داده‌ها ترتیبی دارند.

:One-Hot Encoding

برای هر دسته یک ستون جدید ایجاد می‌شود (با مقدار ۰ یا ۱).

مناسب زمانی که داده‌ها فقط دسته‌ای هستند و ترتیب ندارند.

با استفاده از روش label encoding ستون جدید اضافه می‌کنیم. به دلیل سادگی و تنها وجود یک ویژگی از این روش استفاده می‌کنیم. لیبل‌های در نظر گرفته شده:

```
{0: 'NE', 1: 'NW', 2: 'SE', 3: 'cv'}
```

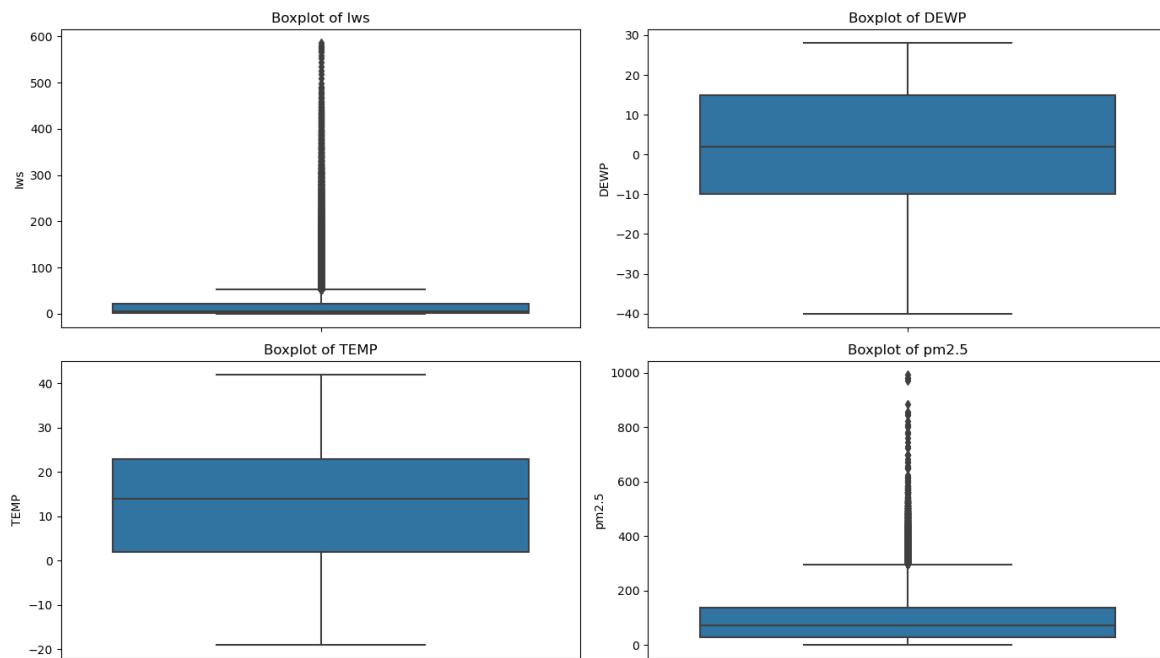
در نهایت مجموعه داده به شکل زیر خواهد بود:

No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir	cbwd_code	
0	25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0	2
1	26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	2.68	0	0	2
2	27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	3.57	0	0	2
3	28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	5.36	1	0	2
4	29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	6.25	2	0	2
...	
43795	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	231.97	0	0	1
43796	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	237.78	0	0	1
43797	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	242.70	0	0	1
43798	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	246.72	0	0	1
43799	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	249.85	0	0	1

43800 rows × 14 columns

- ۱-۲-۵ - داده‌های پرت

با استفاده از boxplot، توزیع دادگان را بررسی می‌کنیم. در این روش چارک‌های اول، دوم و سوم محاسبه شده و پراکندگی دادگان با توجه به این چارک‌ها ترسیم می‌شود.



در نمودارهای boxplot بالا، توزیع داده‌ها و نقاط پرت (outliers) برای ویژگی‌های کلیدی بررسی شده است:

۱. PM2.5

مقدار میانه (Median) در حدود ۹۰-۷۰ قرار دارد.

داده‌های بسیار زیادی در ناحیه بالایی قرار دارند. (right-skewed)

نقاط پرت بسیار زیاد دیده می‌شود، مخصوصاً در بالای نمودار.

• علت:

- آلودگی بالا در برخی روزهای خاص (وارونگی دما، آلودگی صنعتی، شب‌های بی‌باد

زمستانی).

- پدیده‌های غیرعادی جوی یا افزایش ترافیک در تعطیلات.

۲. TEMP

توزیع نسبتاً متقارن است.

فقط تعداد بسیار کمی نقطه پرت وجود دارد.

• علت:

دما داده‌ای فصلی است و در بازه‌ای محدود (مثالاً ۲۰ تا ۴۰ درجه سانتی‌گراد) تغییر می‌کند.

اندازه‌گیری دما نسبتاً دقیق و پایدار است.

DEWP . ۳

دارای داده‌های پرت در هر دو انتهای نمودار.

مقادیر بسیار پایین ممکن است ناشی از رطوبت بسیار پایین (هوای خشک) در زمستان باشد.

مقادیر بالا نیز در تابستان و هوای مرطوب دیده می‌شود.

Iws . ۴

دارای داده‌های پرت به‌ویژه در بالا.

در بیشتر زمان‌ها سرعت باد کم بوده، اما چند نمونه با سرعت بسیار زیاد (باد شدید) دیده می‌شود.

• علت:

◦ طوفان‌های مقطعي یا بادهای فصلی شدید.

◦ دستگاه‌های اندازه‌گیری ممکن است گاهی مقادیر بالا ثبت کنند.

• نتیجه‌گیری:

ویژگی‌هایی مثل TEMP و PRES تقریباً فاقد outlier شدید هستند.

اما Iws و pm2.5 داده‌های پرت مشخص دارند که:

◦ می‌توان آن‌ها را نگه داشت اگر پدیده‌ای واقعی باشند.

◦ یا با تکنیک‌هایی مثل IQR filtering یا winsorization کنترل با حذف کرد، اگر مدل تحت تأثیر قرار گیرد.

اگر بخواهیم دادگان پرت هر یک از ویژگی‌ها را حذف کنیم، می‌توانیم از روش interquantile filtering استفاده کنیم. در این روش دادگانی که بزرگتر از مجموع چارک سوم و $1.5 \times \text{IQR}$ (اختلاف چارک سوم) یا کوچکتر از تفاضل $1.5 \times \text{IQR}$ از چارک اول باشند، حذف می‌شوند.

```

def remove_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df.reset_index(drop=True)

columns_to_clean = ['pm2.5', 'Iws', 'DEWP']
data_cleaned = remove_outliers_iqr(data, columns_to_clean)
#data = remove_outliers_iqr(data, columns_to_clean)

```

- ۱-۲-۶ دسته‌بندی استاندارد غلظت PM2.5

جدول دسته‌بندی به شرح زیر است:

AQI Category	Index Values	Revised Breakpoints ($\mu\text{g}/\text{m}^3$, 24-hour average)
Good	0 - 50	0.0 – 12.0
Moderate	51 - 100	12.1 – 35.4
Unhealthy for Sensitive Groups	101 – 150	35.5 – 55.4
Unhealthy	151 – 200	55.5 – 150.4
Very Unhealthy	201 – 300	150.5 – 250.4
	301 – 400	250.5 – 350.4
Hazardous	401 – 500	350.5 – 500

بنابراین به صورت زیر دادگان را دسته‌بندی می‌کنیم:

```

def categorize_pm25(value):
    if value <= 12.0:
        return 'Good'
    elif value <= 35.4:
        return 'Moderate'
    elif value <= 55.4:
        return 'Unhealthy for Sensitive Groups'
    elif value <= 150.4:
        return 'Unhealthy'
    elif value <= 250.4:
        return 'Very Unhealthy'
    else:
        return 'Hazardous'

```

در نهایت تعداد هر یک از دسته‌ها به شرح زیر خواهد بود:

data['pm25_category'].value_counts()
✓ 0.0s
pm25_category
Unhealthy 16405
Moderate 9151
Very Unhealthy 6160
Unhealthy for Sensitive Groups 5267
Good 3707
Hazardous 3110
Name: count, dtype: int64

ایجاد ویژگی‌های جدید - ۱-۲-۷

ویژگی lag

ویژگی lag به ما این امکان را می‌دهد که مقدار گذشته یک ویژگی را به عنوان ورودی برای مدل در نظر بگیریم. در سری‌های زمانی، این کار برای تشخیص وابستگی زمانی بین مشاهدات بسیار مفید است.

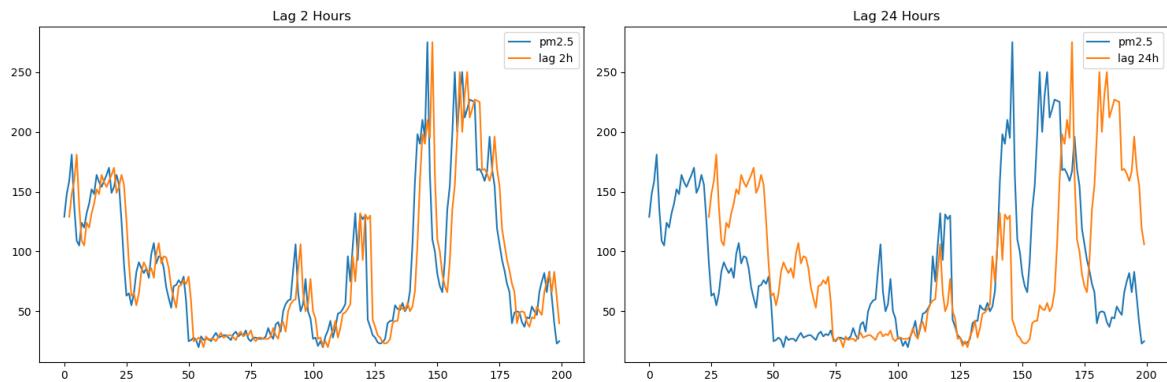
کاربردهای انواع lag:

- ۱ تا ۳ ساعت: بررسی تأثیرات سریع تغییرات شرایط جوی یا ترافیک
- ۲۴ ساعت: شناسایی الگوهای روزانه (مثلاً افزایش آلودگی در ساعات خاصی از روز)
- ۱۶۸ ساعت: شناسایی الگوهای هفتگی (تأثیر فعالیت‌های آخر هفته، الگوهای کاری)

با شیفت دادن دیتا، تاخیر مورد نظر در دیتا را می‌سازیم:

```
data['pm2.5_lag2'] = data['pm2.5'].shift(2)
data['pm2.5_lag24'] = data['pm2.5'].shift(24)
```

سپس نمودار تاخیرهای ایجاد شده را در کنار سیگنال اصلی ترسیم می‌کنیم:



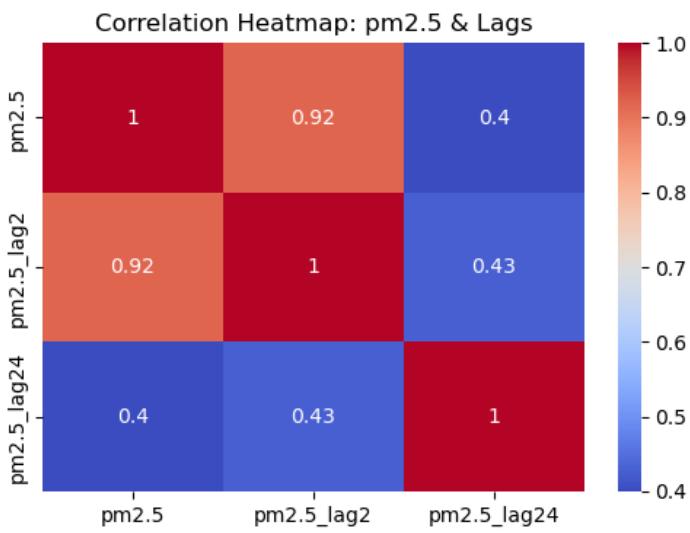
مشاهده می‌شود که نمودارها به صورت تاخیر دلخواه درآمده‌اند. دلیل به وجود آمدن مقادیر NaN نیز همین پدیده است. به طور مثال برای تاخیر دو ساعته، باید دادگان را از ردیف دوم در ستون ساعت انتخاب کنیم. بنابراین ردیف‌های ۰ و ۱ در ستون جدید pm2.5_lag2، مقدار NaN دارند و اولین مقدار در ردیف سوم قرار می‌گیرد. مقدار این عدد، برابر مقدار ردیف صفر در ستون pm2.5 است. تاخیر ۲۴ ساعته نیز به همین صورت است. یعنی ۲۴ ردیف اول در pm2.5_lag24 برابر NaN خواهند شد.

No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir	cbwd_code	pm25_category	pm2.5_lag2	pm2.5_lag24	
0	25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	1.79	0	0	2	Unhealthy	NaN	NaN
1	26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	2.68	0	0	2	Unhealthy	NaN	NaN
2	27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	3.57	0	0	2	Very Unhealthy	129.0	NaN
3	28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	5.36	1	0	2	Very Unhealthy	148.0	NaN
4	29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	6.25	2	0	2	Unhealthy	159.0	NaN
...	
43795	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	231.97	0	0	1	Good	9.0	35.0
43796	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	237.78	0	0	1	Good	10.0	26.0
43797	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	242.70	0	0	1	Good	8.0	20.0
43798	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	246.72	0	0	1	Good	10.0	8.0
43799	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	249.85	0	0	1	Good	10.0	16.0

43800 rows × 17 columns

برای مدیریت مقادیر NaN، می‌توان آن‌ها را حذف کرد. اما فعلاً ردیف‌های مورد نظر را حذف نمی‌کنیم تا ویژگی‌های مورد نظر دیگر را از ویژگی pm2.5 استخراج کنیم. در انتهای پس از استخراج تمامی ویژگی‌ها، ردیف‌های دارای مقدار NaN را حذف می‌کنیم.

ماتریس correlation ویژگی‌های تاخیر به صورت زیر است:



مشاهده می‌شود که همبستگی مقادیر اصلی و تاخیر دو ساعته بسیار بالا و همبستگی مقادیر اصلی و تاخیر ۲۴ ساعته متوسط است. بنابراین می‌توان از این ویژگی‌های برای مدلسازی آلودگی هوای استفاده کرد.

ویژگی Rolling statistics

Rolling statistics یا «آماره‌های متحرک» ابزاری برای بررسی روند متغیرها در طول زمان است. این روش با حرکت یک پنجره ثابت (مثلاً ۲۴ ساعته) روی داده‌ها، آماره‌هایی مانند میانگین، واریانس، انحراف معیار را محاسبه می‌کند.

- این ویژگی‌ها خصوصیات مختلفی از رفتار PM2.5 در زمان را نمایش می‌دهند.
- با ترکیب این آماره‌ها با ویژگی‌های دیگر می‌توان دقیق‌تر مدل‌های یادگیری ماشین را افزایش داد.
- برای مثال skewness و crest factor برای تشخیص رخدادهای ناگهانی و غیرعادی مفید هستند.

از ویژگی‌های زمانی زیر استفاده می‌کنیم تا تفسیرپذیری و دقیقیت مدل خود را بالا ببریم.

ریشه میانگین مربعات (RMS) :

یک معیار آماری مهم که بزرگی کلی یا محتوای انرژی یک سیگنال ارتعاشی را محاسبه می‌کند و نشان دهنده قدرت سیگنال است. و از رابطه زیر محاسبه می‌شود :

$$RMS = \sqrt{\frac{1}{N} \sum_{i=1}^n x_i^2}$$

اوج :

مقدار اوج به حداقل مقدار مطلق دامنه سیگنال در یک بازه زمانی مشخص اشاره دارد و برای شناسایی ناهنجاری های ناگهانی در سیستم استفاده می شود.

$$peak = max(|x_i|)$$

ضریب تاج :

نسبت مقدار اوج سیگنال به RMS می باشد که به عنوان معیاری برای اندازه گیری شدت و تیزی اوج های سیگنال مورد استفاده قرار می گیرد.

$$crest factor = \frac{peak}{RMS}$$

انحراف معیار :

یک معیار آماری است که میزان پراکندگی یا تغییرات دامنه سیگنال از مقدار میانگین آن را نشان می دهد. اگر میانگین را با μ نمایش دهیم انحراف معیار از رابطه زیر قابل محاسبه است.

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^n (x_i - \mu)^2}$$

چولگی :

این معیار آماری عدم تقارن توزیع دامنه حول میانگین آن را نشان میدهد و می تواند به وقایع ضربه ای و شوک ها در سیستم اشاره کند.

$$skewness = \frac{N \sum_{i=1}^n (x_i - \mu)^3}{(N-1)(N-2)\sigma^3}$$

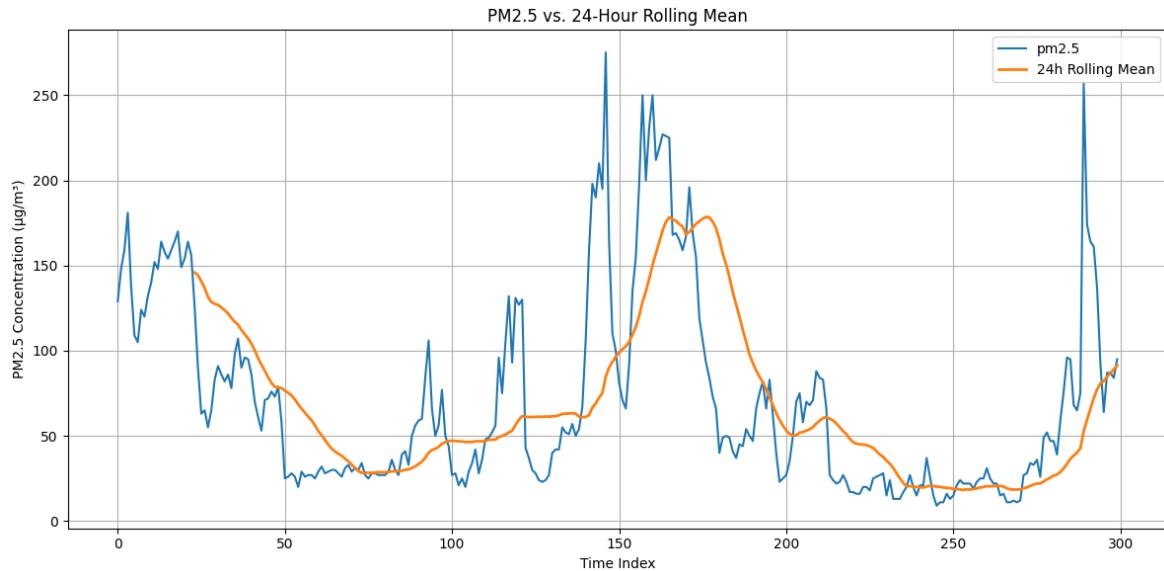
هم چنین سایر ویژگی های استخراج شده از حوزه زمان عبارتند از :

میانگین : اطلاعات مرکزی سیگنال

کشیدگی (Kurtosis) : حساس به پیک های بالا در سیگنال

نرخ عبور از صفر (Zero Crossing Rate) : نشان دهنده فعالیت سیگنال در بازه زمانی

نمودار زیر نشان دهنده اعمال ویژگی میانگین در پنجره ۲۴ ساعته است.



- خط آبی (pm2.5) دارای نوسانات شدید است.
- خط نارنجی (rolling mean) روان‌تر و نمایانگر میانگین تغییرات در ۲۴ ساعت اخیر است.
- این نمودار نشان می‌دهد که استفاده از rolling mean باعث صاف‌شدن روندها و حذف نویز می‌شود.
- چرا برای ۲۴ ردیف اول مقدار NaN است؟ زیرا تا قبل از ردیف ۲۴، پنجره کامل ۲۴ ساعته وجود ندارد. از ردیف ۲۳ به بعد، این مقدار قابل محاسبه است.

در نهایت این ویژگی‌ها را علاوه بر سیگنال pm2.5 بر روی سیگنال‌های TEMP، PRES، DEWP، Iws، Is، Ir نیز اعمال می‌کنیم تا هم در classification و هم در regression از آن‌ها استفاده کنیم. دقت داشته باشید که نباید از ویژگی‌های سیگنال pm2.5 در classification استفاده کرد.

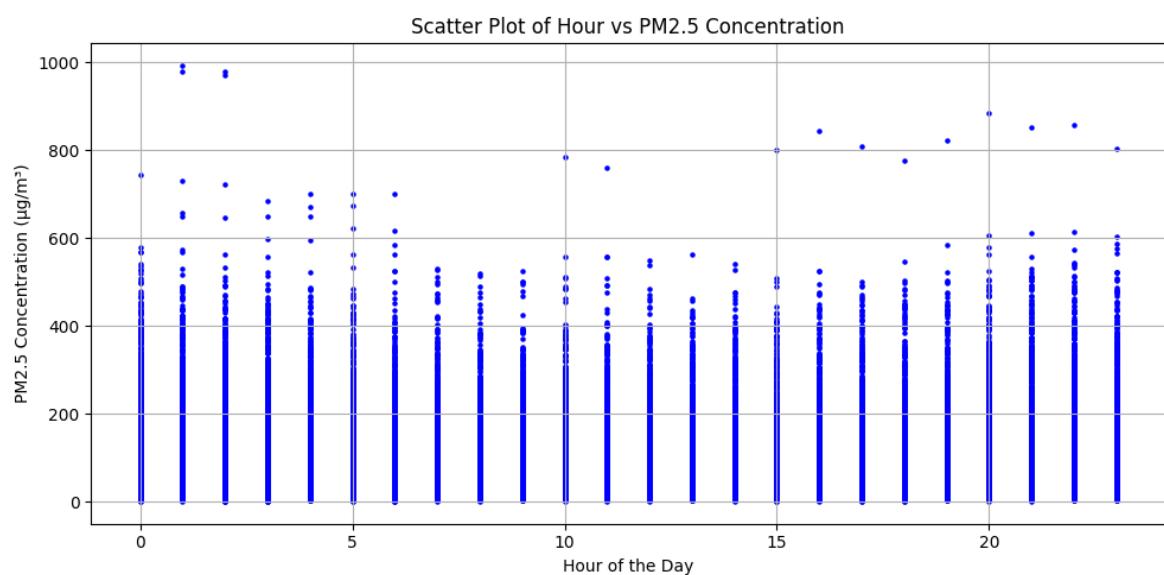
No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	...	ls_roll_rms	ls_roll_peak	ls_roll_crest	ls_roll_entropy	lr_roll_mean_24h	lr_roll_std	lr_roll_rms	lr_roll_peak	lr_roll_crest	lr_roll_entropy
0	25	2010	1	2	0	129.0	-16	-4.0	1020.0	SE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	26	2010	1	2	1	148.0	-15	-4.0	1020.0	SE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	27	2010	1	2	2	159.0	-11	-5.0	1021.0	SE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	28	2010	1	2	3	181.0	-7	-5.0	1022.0	SE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	29	2010	1	2	4	138.0	-7	-5.0	1022.0	SE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
43795	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	...	0.0	0.0	0.0	2.162327e-08	0.0	0.0	0.0	0.0	2.162327e-08
43796	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	...	0.0	0.0	0.0	2.162327e-08	0.0	0.0	0.0	0.0	2.162327e-08
43797	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	...	0.0	0.0	0.0	2.162327e-08	0.0	0.0	0.0	0.0	2.162327e-08
43798	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	...	0.0	0.0	0.0	2.162327e-08	0.0	0.0	0.0	0.0	2.162327e-08
43799	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	...	0.0	0.0	0.0	2.162327e-08	0.0	0.0	0.0	0.0	2.162327e-08

انکود کردن ویژگی‌های تناوبی

ویژگی‌های زیر تناوبی (دوره‌ای) هستند:

- ساعت روز، چرخه ۲۴ ساعته Hour
- روز ماه، چرخه ۳۱ روزه day
- چرخه ۱۲ ماهه month
- دو روزه - هنوز در دیتاست نیست، باید بسازیم weekday

روش‌هایی مانند One-Hot-Encoding برای ویژگی‌های تناوبی مناسب نیستند، چون مثلاً ساعت ۲۳ و ۰ باید به هم نزدیک باشند، اما در OHE این طور نیست. بنابراین از **Sin/Cos Encoding** استفاده می‌کنیم. ابتدا ویژگی weekday را با تبدیل تاریخ کامل به زمان ایجاد می‌کنیم، سپس انکودینگ سینوسی/کسینوسی را اعمال می‌کنیم و نمودار را رسم می‌کنیم.



نمودار بالا رابطه‌ی بین ساعت شبانه‌روز و مقدار آلودگی PM2.5 را نشان می‌دهد.

تحلیل نمودار: Hour vs PM2.5

- افزایش آلودگی در ساعات اولیه روز قابل مشاهده است.
- کاهش نسبی در ساعات عصر، و سپس افزایش مجدد شبانه (مثلاً ۲۱ تا ۲۳) نیز دیده می‌شود.
- این روند نشان‌دهنده الگوی تناوبی آلودگی مرتبط با فعالیت انسانی (ترافیک صبح و شب) است.

ویژگی زمانی پیشرفته
آلودگی بر حسب فصل:

(season	
Winter	109.872500
Autumn	101.582509
Summer	91.739764
Spring	88.245380

بیشترین آلودگی در فصل زمستان رخ می‌دهد زیرا در زمستان وارونگی دما (Inversion) رخ می‌دهد که مانع از پخش آلاینده‌ها می‌شود.

آلودگی بر حسب روز هفته:

weekday_name	
Saturday	103.153736
Friday	99.845353
Sunday	98.595945
Wednesday	97.284483
Thursday	96.920353
Tuesday	96.249840
Monday	92.443008

بالاترین مربوط به روز شنبه است زیرا در تعطیلات آخر هفته و پایان هفته (جمعه و شنبه)، افزایش ترافیک و فعالیت صنعتی ممکن است رخ دهد.

تأثیر فشار هوا، باد و دما (همبستگی با آلودگی)

pm2.5	1.000000
PRES	-0.057439
TEMP	-0.077869
Iws	-0.243067

ضریب همبستگی (با PM2.5):

Iws (سرعت باد): → باد بیشتر → پراکندگی آلاینده‌ها → آلودگی کمتر
 TEMP (دما): → دمای بالاتر معمولاً آلودگی را کم می‌کند
 PRES (فارش هوای): → تأثیر نسبتاً کم ولی منفی

الگوی ساعتی:

۲- صبح: بیشترین (تا ۱۱۲)

۸-۵ صبح: کاهش تدریجی

۱۶-۱۴: کمترین مقدار ۸۵

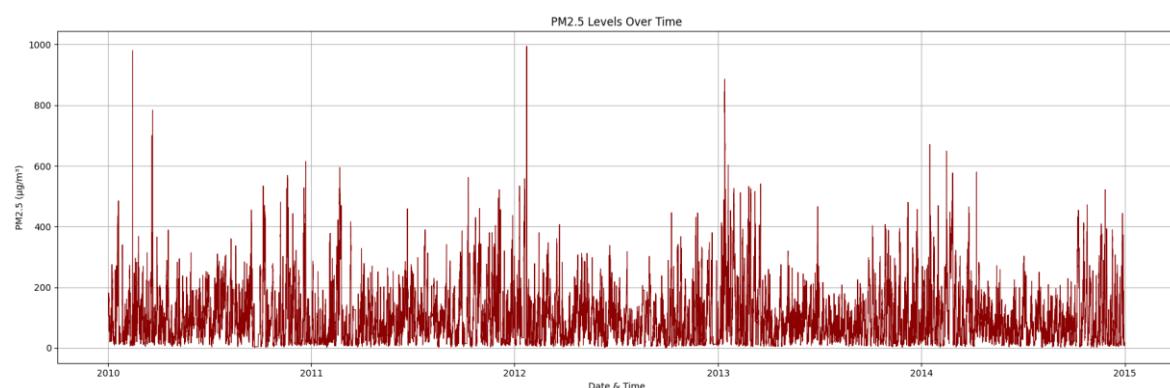
۲۳-۲۰: دوباره افزایش شدید

- ۱-۲-۸ مصورسازی داده‌ها

با استفاده از ستون‌های year, month, day, hour، یک ستون جدید به نام datetime ساختیم که امکان رسم نمودار سری زمانی (Time Series) برای ویژگی pm2.5 را فراهم می‌کند.

```
# Create datetime column  
data['datetime'] = pd.to_datetime(data[['year', 'month', 'day', 'hour']])
```

مقدار Pm2.5 بر حسب تاریخ و زمان:



بیشینه آلودگی:

تاریخ بیشترین مقدار PM2.5:

۰ ساعت ۲۰ ۱۲ ژانویه ۲۰۱۴

مقدار PM2.5 در آن لحظه: $994 \mu\text{g}/\text{m}^3$

میانگین PM2.5 ماه

ژانویه (January) 109.97

دسامبر (December) 95.02

چرا ژانویه بیشتر از دسامبر است؟

- وارونگی دما در ژانویه شدیدتر است (بیشترین سرمای هوا).
- سیستم‌های گرمایشی خانگی و صنعتی بهشدت فعال هستند.
- در برخی کشورها تعطیلات سال نو تا اواسط ژانویه است و فعالیت‌ها بیشتر است.
- هوای سردتر باعث ماندگاری بیشتر ذرات معلق در هوا می‌شود.

در نهایت پس از استخراج تمامی ویژگی‌ها، دادگان ۲۴ سطر اول که NaN هستند را حذف می‌کنیم و دیتا مورد نظر خود را ذخیره می‌کنیم:

No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	...	hour_sin	hour_cos	month_sin	month_cos	day_sin	day_cos	weekday_sin	weekday_cos	season	weekday_name	
0	49	2010	1	3	0	90.0	-7	-6.0	1027.0	SE	...	0.000000	1.000000	5.00000e-01	0.866025	5.712682e-01	0.820763	-0.781831	0.623490	Winter	Sunday
1	50	2010	1	3	1	63.0	-8	-6.0	1026.0	SE	...	0.258819	0.965926	5.00000e-01	0.866025	5.712682e-01	0.820763	-0.781831	0.623490	Winter	Sunday
2	51	2010	1	3	2	65.0	-8	-7.0	1026.0	SE	...	0.500000	0.866025	5.00000e-01	0.866025	5.712682e-01	0.820763	-0.781831	0.623490	Winter	Sunday
3	52	2010	1	3	3	55.0	-8	-7.0	1025.0	SE	...	0.707107	0.707107	5.00000e-01	0.866025	5.712682e-01	0.820763	-0.781831	0.623490	Winter	Sunday
4	53	2010	1	3	4	65.0	-8	-7.0	1024.0	SE	...	0.866025	0.500000	5.00000e-01	0.866025	5.712682e-01	0.820763	-0.781831	0.623490	Winter	Sunday
...		
43771	43820	2014	12	31	19	8.0	-23	-2.0	1034.0	NW	...	-0.965926	0.258819	-2.449294e-16	1.000000	-2.449294e-16	1.000000	0.974928	-0.222521	Winter	Wednesday
43772	43821	2014	12	31	20	10.0	-22	-3.0	1034.0	NW	...	-0.866025	0.500000	-2.449294e-16	1.000000	-2.449294e-16	1.000000	0.974928	-0.222521	Winter	Wednesday
43773	43822	2014	12	31	21	10.0	-22	-3.0	1034.0	NW	...	-0.707107	0.707107	-2.449294e-16	1.000000	-2.449294e-16	1.000000	0.974928	-0.222521	Winter	Wednesday
43774	43823	2014	12	31	22	8.0	-22	-4.0	1034.0	NW	...	-0.500000	0.866025	-2.449294e-16	1.000000	-2.449294e-16	1.000000	0.974928	-0.222521	Winter	Wednesday
43775	43824	2014	12	31	23	12.0	-21	-3.0	1034.0	NW	...	-0.258819	0.965926	-2.449294e-16	1.000000	-2.449294e-16	1.000000	0.974928	-0.222521	Winter	Wednesday

43776 rows × 77 columns

بررسی توزیع داده‌ها - ۱-۲-۹

توزیع دادگان در کلاس‌های مورد نظر طبق AQI به شرح زیر است:

pm25_category	count
Unhealthy	16393
Moderate	9151
Very Unhealthy	6148
Unhealthy for Sensitive Groups	5267
Good	3707
Hazardous	3110
Name: count, dtype: int64	

بنابراین، برای متوازن کردن دادگان، با استفاده از روش downsampling، تعداد تمامی کلاس‌ها را همانند از تعداد کمترین کلاس یعنی hazardous با ۳۱۱۰ داده می‌کنیم:

pm25_category	
Unhealthy	3110
Moderate	3110
Very Unhealthy	3110
Unhealthy for Sensitive Groups	3110
Good	3110
Hazardous	3110
Name: count, dtype: int64	

۱-۲-۱- نرمالسازی داده‌ها

توجه داشته باشید که نرمالسازی دادگان باید بعد از تقسیم دادگان به آموزش و آزمایش انجام شود، چرا که می‌بایست نرمالسازی تنها روی دادگان آموزش fit شده و روی دادگان آزمایش تنها transform شود. دخیل کردن دادگان آزمایش در فرایند نرمالسازی اشتباه است. بنابراین در فایل شبیه‌سازی، ابتدا دادگان تقسیم شده و سپس نرمالسازی انجام می‌شود.

نرمالسازی دادگان تقسیم شده به صورت زیر انجام می‌شود. به دلیل اینکه با دادگان سری زمانی سر و کار داریم از روش standard scaler استفاده می‌کنیم.

```
# Normalize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

X_train_scaled.shape, X_val_scaled.shape, X_test_scaled.shape
```

۱-۲-۱-۱- تقسیم داده‌ها به مجموعه‌های آموزش و آزمون و ارزیابی

دادگان را به نسبت ۷۰ - ۱۵ - ۱۵ تقسیم می‌کنیم (قبل از نرمالسازی)

```

#df=df.dropna()
X = df.drop(['pm25_category_encoded'],axis=1)
y = df['pm25_category_encoded']

# Split into train (70%), temp (30%)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, train_size=0.7, random_state=rs, stratify=y)

# Split temp into validation (15%) and test (15%)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, train_size=0.5, random_state=rs, stratify=y_temp)

```

۱-۲-۱۲ - مدلسازی با استفاده از SVM برای دسته‌بندی

در الگوریتم SVM با کرنل‌های مختلف، پارامترهایی وجود دارد که هر کدام معنای خاصی دارند و نقش کلیدی در عملکرد مدل دارند. در ادامه، مفهوم هر پارامتر در کرنل‌های مختلف را توضیح داده می‌شود:

پارامتر مشترک بین همه کرنل‌ها

C (Cost / Regularization Parameter) •

مفهوم: تعادل بین درست طبقه‌بندی کردن نمونه‌های آموزش و حداکثر کردن margin فاصله بین مرز تصمیم و نقاط داده

مقدار کم (مثل ۰.۱): مدل سعی می‌کند margin بزرگ‌تری داشته باشد، حتی اگر بعضی نمونه‌ها را اشتباه طبقه‌بندی کند → ممکن است underfitting ایجاد شود.

مقدار زیاد (مثل ۱۰۰): مدل سعی می‌کند تمام نمونه‌ها را درست طبقه‌بندی کند حتی اگر کوچک شود → ممکن است overfitting شود.

RBF

gamma •

مفهوم: شاع تأثیر هر نقطه داده. کنترل می‌کند که چقدر یک نمونه آموزش روی تصمیم‌گیری تأثیر بگذارد.

مقدار بالا (مثل ۱): هر نقطه فقط در ناحیه بسیار نزدیک خودش تأثیر دارد → مرز تصمیم پیچیده و ممکن است overfit شود.

مقدار پایین (مثل ۰۰۰۱): تأثیر هر نقطه وسیع‌تر است → مرز تصمیم نرم‌تر و احتمال بیشتر می‌شود.

Polynomial

degree •

مفهوم: درجه چندجمله‌ای که برای ساختن کرنل استفاده می‌شود.

مقدار بالا (مثالاً ۴ یا ۵) باعث می‌شود مرز تصمیم پیچیده‌تر شود.

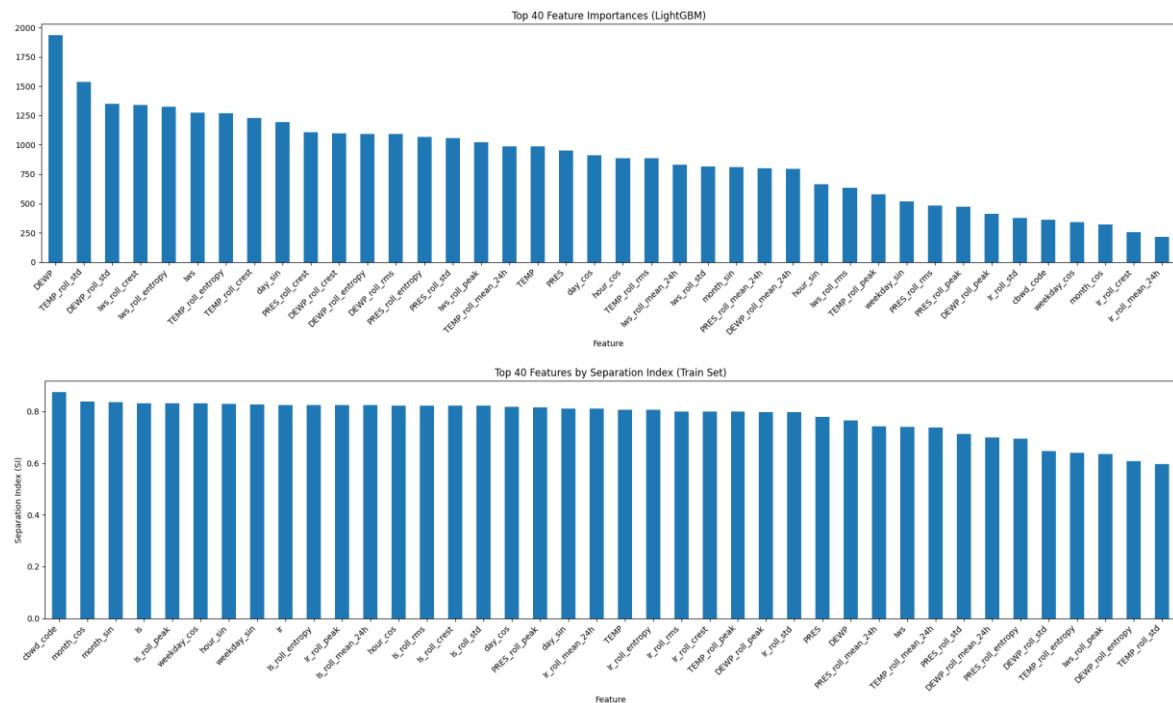
RBF Gamma •

کنترل کننده تأثیر ویژگی‌ها در کرنل چندجمله‌ای.

Linear

فقط پارامتر C دارد. چون کرنل خطی به صورت مستقیم از حاصل ضرب داخلی داده‌ها استفاده می‌کند و پارامترهای دیگری مثل gamma در آن بی‌معنی هستند.

برای آموزش مدل طبقه‌بندی SVM می‌توان از ویژگی‌هایی که در مراحل قبل استخراج کردیم استفاده کنیم. همچنین برای کاهش ابعاد و آموزش مدل بر حسب مهم‌ترین ویژگی‌ها در طبقه‌بندی، می‌توانیم ویژگی‌های مهم را با استفاده از دو روش lightgbm و AI بدست آوریم و اشتراک ویژگی‌های بدست آمده از این دو روش را به عنوان ویژگی‌های نهایی در طبقه‌بندی در نظر بگیریم:



هر چند به دلیل زیاد بدون حجم محاسبات و زیاد بودن نمونه‌ها برای آموزش مدل، از لحاظ زمانی، آموزش با استفاده از دو روش ذکر شده، یعنی استفاده از کل ویژگی‌ها یا تنها استفاده از ویژگی‌های مهم انتخاب شده، در این شبیه‌سازی چندان تفاوتی وجود نداشت. ویژگی‌های استفاده شده در این شبیه‌سازی عبارتند از:

```

selected_columns=[

'DEWP', 'TEMP', 'PRES',
'Iws', 'Is', 'Ir', 'cbwd_code', 'hour_sin', 'hour_cos',
'month_sin', 'month_cos', 'day_sin', 'day_cos', 'weekday_sin',
'weekday_cos', 'pm25_category_encoded',
'PRES_roll_mean_24h', 'PRES_roll_std', 'PRES_roll_rms',
'PRES_roll_peak', 'PRES_roll_crest',
'PRES_roll_entropy', 'DEWP_roll_mean_24h', 'DEWP_roll_std',
'DEWP_roll_rms', 'DEWP_roll_peak', 'DEWP_roll_crest',
'DEWP_roll_entropy', 'TEMP_roll_mean_24h',
'TEMP_roll_std', 'TEMP_roll_rms', 'TEMP_roll_peak', 'TEMP_roll_crest',
'TEMP_roll_entropy',
'Iws_roll_mean_24h', 'Iws_roll_std', 'Iws_roll_rms', 'Iws_roll_peak',
'Iws_roll_crest', 'Iws_roll_entropy',
'Is_roll_mean_24h', 'Is_roll_std', 'Is_roll_rms', 'Is_roll_peak',
'Is_roll_crest', 'Is_roll_entropy',
' Ir_roll_mean_24h', 'Ir_roll_std', 'Ir_roll_rms', 'Ir_roll_peak',
' Ir_roll_crest', 'Ir_roll_entropy',
]

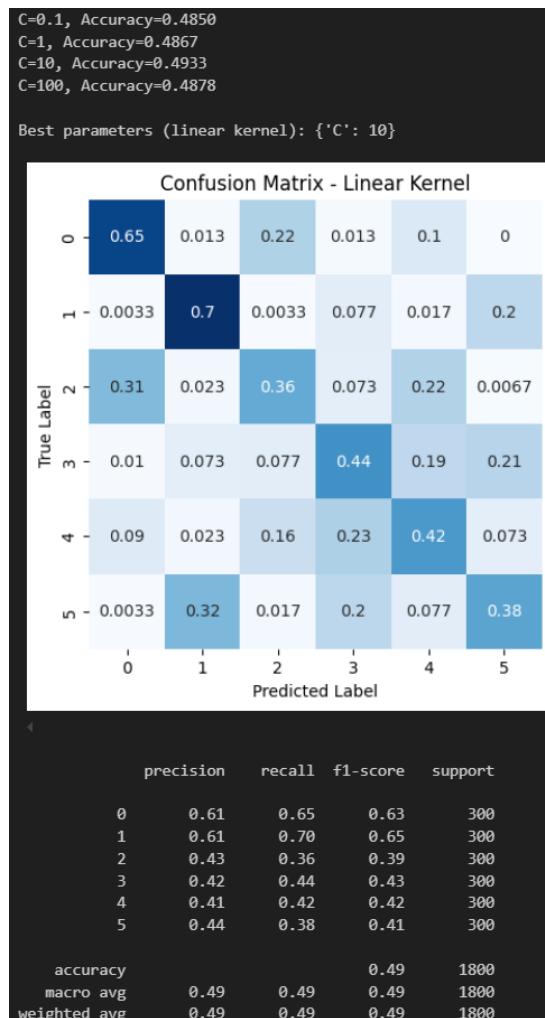
```

همچنین به دلیل حجم محاسبات بالا، تعداد نمونه استفاده شده هر کلاس از ۳۱۱۰ نمونه به ۲۰۰۰ نمونه کاهش یافته است، تا برنامه در زمان سربعینتری انجام شود. طبیعتاً استفاده از نمونه‌های بیشتر دقت مدل را افزایش می‌دهد.

pm25_category	
Unhealthy	2000
Moderate	2000
Very Unhealthy	2000
Unhealthy for Sensitive Groups	2000
Good	2000
Hazardous	2000
Name: count, dtype: int64	

در ادامه فرایند grid search برای توابع svm با کرنل‌های مختلف انجام شده و ماتریس درهم‌ریختگی و معیارهای سنجش دقت و صحت برای بهترین مدل در هر کرنل گزارش می‌شود:

نتایج شبیه‌سازی با استفاده از linear grid search برای کرنل به شرح زیر است:

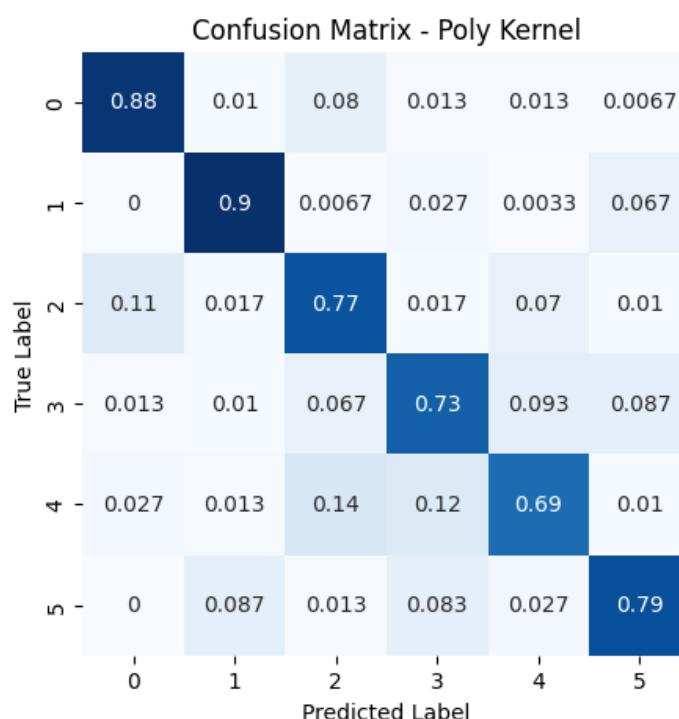


نتایج شبیه‌سازی با استفاده از polynomial grid search برای کرنل به شرح زیر است:

C=0.1, degree=2, gamma=1, Accuracy=0.6617
 C=0.1, degree=2, gamma=0.1, Accuracy=0.5622
 C=0.1, degree=2, gamma=0.01, Accuracy=0.3294
 C=0.1, degree=3, gamma=1, Accuracy=0.7750
 C=0.1, degree=3, gamma=0.1, Accuracy=0.7139
 C=0.1, degree=3, gamma=0.01, Accuracy=0.3067
 C=0.1, degree=4, gamma=1, Accuracy=0.7922
 C=0.1, degree=4, gamma=0.1, Accuracy=0.7861
 C=0.1, degree=4, gamma=0.01, Accuracy=0.2267
 C=1, degree=2, gamma=1, Accuracy=0.7094
 C=1, degree=2, gamma=0.1, Accuracy=0.6194
 C=1, degree=2, gamma=0.01, Accuracy=0.4656
 C=1, degree=3, gamma=1, Accuracy=0.7750
 C=1, degree=3, gamma=0.1, Accuracy=0.7694
 C=1, degree=3, gamma=0.01, Accuracy=0.4506

C=1, degree=4, gamma=1, Accuracy=0.7922
 C=1, degree=4, gamma=0.1, Accuracy=0.7900
 C=1, degree=4, gamma=0.01, Accuracy=0.3306
 C=10, degree=2, gamma=1, Accuracy=0.7400
 C=10, degree=2, gamma=0.1, Accuracy=0.6611
 C=10, degree=2, gamma=0.01, Accuracy=0.5622
 C=10, degree=3, gamma=1, Accuracy=0.7750
 C=10, degree=3, gamma=0.1, Accuracy=0.7683
 C=10, degree=3, gamma=0.01, Accuracy=0.5939
 C=10, degree=4, gamma=1, Accuracy=0.7922
 C=10, degree=4, gamma=0.1, Accuracy=0.7922
 C=10, degree=4, gamma=0.01, Accuracy=0.5006
 C=100, degree=2, gamma=1, Accuracy=0.7483
 C=100, degree=2, gamma=0.1, Accuracy=0.7094
 C=100, degree=2, gamma=0.01, Accuracy=0.6194
 C=100, degree=3, gamma=1, Accuracy=0.7750
 C=100, degree=3, gamma=0.1, Accuracy=0.7750
 C=100, degree=3, gamma=0.01, Accuracy=0.7139
 C=100, degree=4, gamma=1, Accuracy=0.7922
 C=100, degree=4, gamma=0.1, Accuracy=0.7922
 C=100, degree=4, gamma=0.01, Accuracy=0.7022

Best parameters (poly kernel): {'C': 0.1, 'degree': 4, 'gamma': 1}

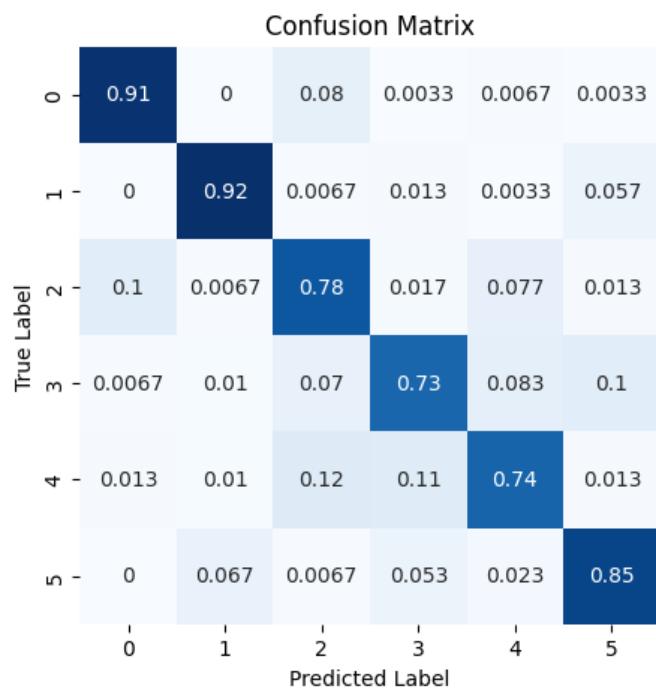


	precision	recall	f1-score	support
0	0.85	0.88	0.86	300
1	0.87	0.90	0.88	300
2	0.71	0.77	0.74	300
3	0.74	0.73	0.73	300
4	0.77	0.69	0.73	300
5	0.81	0.79	0.80	300
accuracy			0.79	1800
macro avg	0.79	0.79	0.79	1800
weighted avg	0.79	0.79	0.79	1800

نتایج شبیه‌سازی با استفاده از grid search برای کرنل RBF به شرح زیر است:

```
C=0.1, gamma=1, Accuracy=0.4606
C=0.1, gamma=0.1, Accuracy=0.4928
C=0.1, gamma=0.01, Accuracy=0.4356
C=0.1, gamma=0.001, Accuracy=0.3528
C=1, gamma=1, Accuracy=0.7144
C=1, gamma=0.1, Accuracy=0.7539
C=1, gamma=0.01, Accuracy=0.5428
C=1, gamma=0.001, Accuracy=0.4244
C=10, gamma=1, Accuracy=0.7211
C=10, gamma=0.1, Accuracy=0.8178
C=10, gamma=0.01, Accuracy=0.6761
C=10, gamma=0.001, Accuracy=0.4900
C=100, gamma=1, Accuracy=0.7211
C=100, gamma=0.1, Accuracy=0.8206
C=100, gamma=0.01, Accuracy=0.7522
C=100, gamma=0.001, Accuracy=0.5733
```

```
Best parameters found: {'C': 100, 'gamma': 0.1}
```

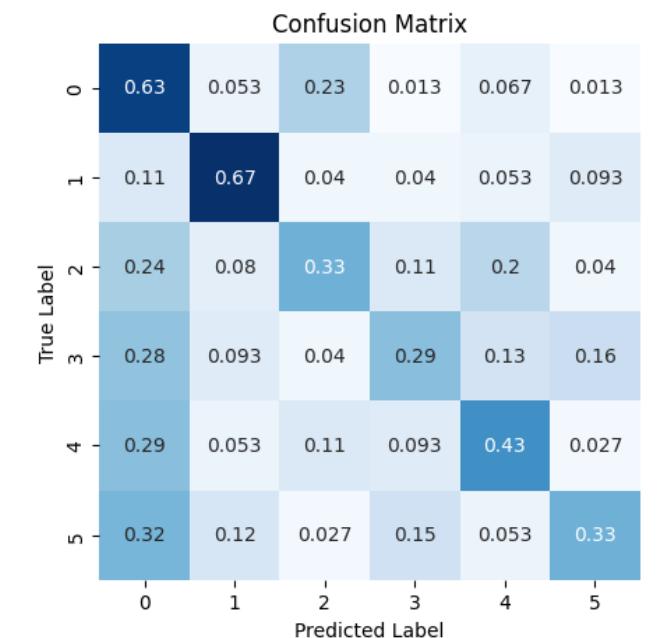


	precision	recall	f1-score	support
0	0.88	0.91	0.89	300
1	0.91	0.92	0.91	300
2	0.74	0.78	0.76	300
3	0.79	0.73	0.76	300
4	0.79	0.74	0.76	300
5	0.82	0.85	0.83	300
accuracy			0.82	1800
macro avg	0.82	0.82	0.82	1800
weighted avg	0.82	0.82	0.82	1800

بهترین مدل به دست آمده، مدل با کرنل RBF زمان کمتری برای محاسبات نیاز دارد، بنابراین grid search روی آن راحت‌تر اجرا می‌شود.

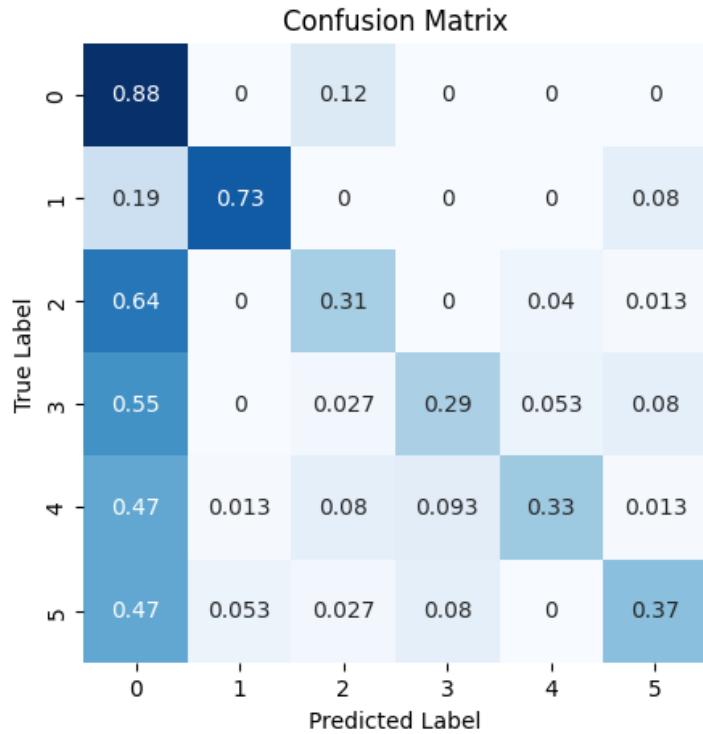
الگوریتم SVM را بدون استفاده از توابع آماده به صورت scratch توسعه می‌دهیم. در این روش از کتابخانه cvxopt برای انجام الگوریتم بهینه‌سازی در SVM استفاده می‌کنیم. می‌توان از روش‌های گرادیان محور نیز استفاده کرد، اما به دلیل تصادفی بودن این الگوریتم‌ها، ممکن است تایع هزینه در نقاط کمینه محلی توقف کند. به دلیل کند بودن توابع scratch نوشته شده و همچین محاسبات زیر، تعداد ۵۰۰ داده از هر کلاس را آموزش داده و آزمایش می‌کنیم:

نتایج شبیه‌سازی برای کرنل polynomial با درجه ۳ و ۱ scratch svm



	precision	recall	f1-score	support
0	0.34	0.63	0.44	75
1	0.62	0.67	0.65	75
2	0.43	0.33	0.38	75
3	0.42	0.29	0.35	75
4	0.46	0.43	0.44	75
5	0.50	0.33	0.40	75
accuracy			0.45	450
macro avg	0.46	0.45	0.44	450
weighted avg	0.46	0.45	0.44	450

نتایج شبیه‌سازی برای کرنل RBF با $\gamma = 0.1$ برای scratch svm



	precision	recall	f1-score	support
0	0.28	0.88	0.42	75
1	0.92	0.73	0.81	75
2	0.55	0.31	0.39	75
3	0.63	0.29	0.40	75
4	0.78	0.33	0.47	75
5	0.67	0.37	0.48	75
accuracy			0.49	450
macro avg	0.64	0.49	0.50	450
weighted avg	0.64	0.49	0.50	450

به دلیل کند بودن این الگوریتم، استفاده از grid search برای توابع scratch محدود نیست. همچنین با افزایش تعداد دادگان آموزش، قطعاً دقت مدل افزایش خواهد یافت. اما به دلیل بهینه نبودن این الگوریتم‌ها، بهترین مدل در این بخش sklearn rbf است که دقت ۸۲ درصدی دارد.

برای پیش‌بینی نیز همانند قسمت طبقه‌بندی عمل می‌کنیم. با این تفاوت که می‌توانیم از ویژگی‌های استخراج شده از سیگنال pm2.5 را نیز در مدل دخیل کنیم.

تفاوت مدل پیش‌بینی کننده و طبقه‌بندی کننده در عملکرد اینگونه است که، تابع طبقه‌بند، برای دسته‌بندی خروجی، از توابعی مانند آستانه استفاده می‌کند تا مقدار تابع هدف در هر نمونه را با توجه به آستانه دسته بندی کند. اما در پیش‌بینی سعی می‌شود تا کمترین میزان خطأ نسبت به سیگنال خروجی اخذ شود.

نتایج grid search برای کرنل polynomial برای پیش‌بینی:

```
reg = SVR(kernel='poly',degree=4,gamma=1,C=0.1)
reg.fit(X_train_scaled,y_train)
y_pred=reg.predict(X_test_scaled)

acc = reg.score(X_test_scaled,y_test)
acc
✓ 50.5s
0.7206601997575086
```

نتایج grid search برای کرنل rbf برای پیش‌بینی:

```
C=0.1, gamma=1, Accuracy=-0.2644
C=0.1, gamma=0.1, Accuracy=-0.2365
C=0.1, gamma=0.01, Accuracy=0.0377
C=0.1, gamma=0.001, Accuracy=-0.1831
C=1, gamma=1, Accuracy=-0.2473
C=1, gamma=0.1, Accuracy=-0.0177
C=1, gamma=0.01, Accuracy=0.6557
C=1, gamma=0.001, Accuracy=0.2969
C=10, gamma=1, Accuracy=-0.0989
C=10, gamma=0.1, Accuracy=0.5749
C=10, gamma=0.01, Accuracy=0.8045
C=10, gamma=0.001, Accuracy=0.7142
C=100, gamma=1, Accuracy=0.5028
C=100, gamma=0.1, Accuracy=0.8890
C=100, gamma=0.01, Accuracy=0.8837
C=100, gamma=0.001, Accuracy=0.8031

Best parameters found: {'C': 100, 'gamma': 0.1}
```

صحت برای بهترین مدل:

```
acc = reg.score(X_test_scaled,y_test)
acc
✓ 4.7s
0.8889853124497524
```

طبعاً به دلیل اینکه طبقه‌بندی انجام نمی‌شود، نمی‌توانن ماتریس درهم‌ریختگی را برای مدل پیش‌بینی کننده نمایش داد.

۱-۲-۱۴ - معرفی الگوریتم PSO

الگوریتم بهینه‌سازی ازدحام ذرات (Particle Swarm Optimization - PSO) یکی از روش‌های بهینه‌سازی مبتنی بر جمعیت است که از رفتار اجتماعی پرندگان و ماهی‌ها الهام گرفته شده است. در PSO، هر ذره نمایانگر یک راه حل ممکن برای مسئله است و در یک فضای جستجو حرکت می‌کند تا بهترین نقطه را پیدا کند. هر ذره دارای موقعیت و سرعت است که به مرور زمان بهروزرسانی می‌شود. متغیرهای این الگوریتم عبارتند از:

- تعداد کل ذرات: N
- تعداد کل ابعاد: d
- موقعیت ذره i در لحظه t : $X_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{id}^t)$
- سرعت ذره i در لحظه t : $V_i^t = (v_{i1}^t, v_{i2}^t, \dots, v_{id}^t)$
- بهترین موقعیت شخصی ذره i در لحظه t : $P_i = (P_{i1}, P_{i2}, \dots, P_{id})$
- بهترین موقعیت جهانی ذره i در لحظه t : $G = (g_1, g_2, \dots, g_d)$

حرکت هر ذره تحت تأثیر دو عامل است:

۱. تجربه شخصی (pBest): ذره به موقعیت قبلی خود که بهترین مقدار تابع هدف را داشته است جذب می‌شود.
۲. تجربه جمعی (gBest): ذره به بهترین موقعیت کل جمعیت تمایل دارد.
فرمول‌های اصلی برای بهروزرسانی سرعت و موقعیت هر ذره به شکل زیر هستند:

$$V_i^{t+1} = wV_i^t + c_1r_1(P_i - X_i^t) + c_2r_2(G - X_i^t)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

که در آن w ضریب اینرسی، c_1 و c_2 ، ضرایب یادگیری که شدت pBest و gBest را مشخص می‌کنند و r_1 و r_2 اعداد تصادفی بین $[0,1]$ هستند.

در مدل SVM با کرنل RBF، پارامترهای کلیدی که نیاز به بهینه‌سازی دارند:

- C : پارامتر پنالتی برای کنترل خطای مدل.
- γ : پارامتر کرنل RBF که میزان تأثیر نمونه‌ها را تعیین می‌کند.

PSO می‌تواند با جستجو در فضای این دو پارامتر بهترین ترکیب را بیابد که منجر به بیشترین دقیقیت مدل شود.

۱-۲-۱۵ پیاده‌سازی SVM با PSO

این الگوریتم با سرعت خوبی به جواب بهینه همگرا می‌شود و مزیتی که نسبت به grid search دارد این است که، ممکن است در grid search، مقدایر انتخابی برای جست و جو مناسب نباشند و جواب بهینه حاصل نشود. دقیقیت نهایی ۸۳ درصد است که تنها ۱ درصد از روش rbf با grid search بهتر است. در این شبیه‌سازی، مقادیر C و Gammap را بهینه کرده‌ایم. برای مقادیر C بین ۰.۱ تا ۱۰۰ و گاما بین ۰.۱ تا ۱، این بهینه‌سازی انجام شده است.

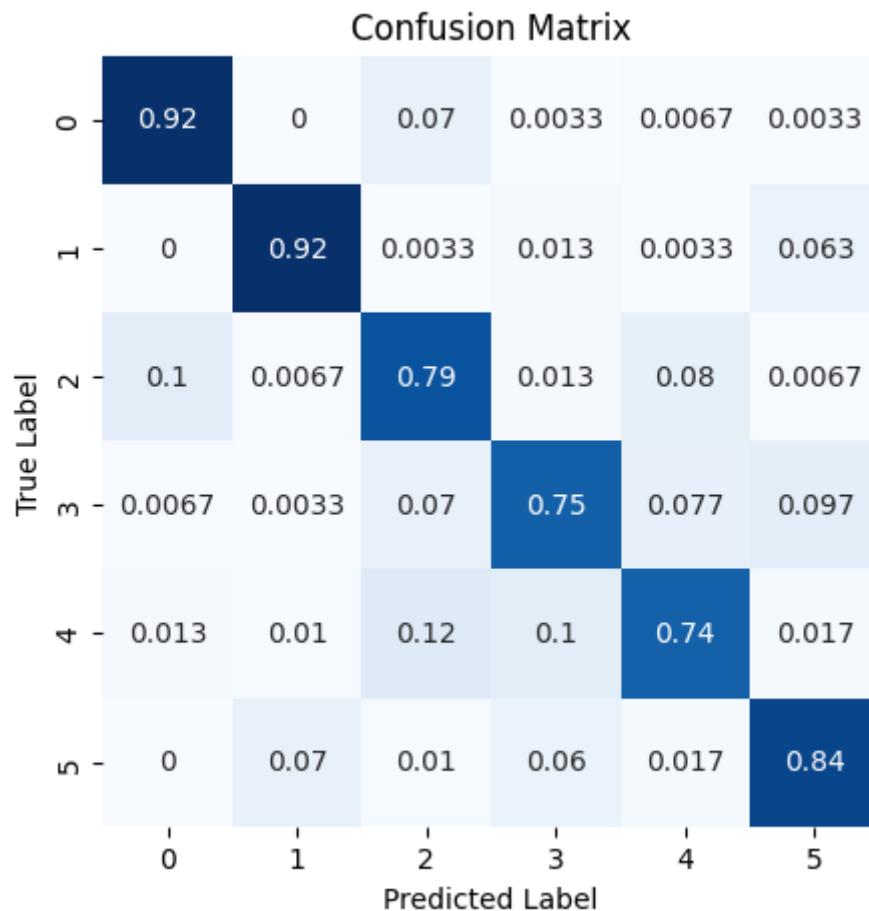
```
lb = [0.1, 0.01]
ub = [100, 1]

best_params, _ = pso(fitness_function, lb, ub, swarmsize=20, maxiter=30)
C_opt, gamma_opt = best_params
```

```

C_opt,gamma_opt
✓ 0.0s
(71.52809506216477, 0.07263745094688884)

```



	precision	recall	f1-score	support
0	0.88	0.92	0.90	300
1	0.91	0.92	0.91	300
2	0.75	0.79	0.77	300
3	0.80	0.75	0.77	300
4	0.80	0.74	0.77	300
5	0.82	0.84	0.83	300
accuracy			0.83	1800
macro avg	0.83	0.83	0.83	1800
weighted avg	0.83	0.83	0.83	1800

۱-۳-۱- بررسی و پیاده‌سازی مقاله

۱-۳-۱- عنوان مقاله

۱-۳-۲- دیتاست استفاده شده

در مقاله از دیتاست کیفیت هوای پکن (Beijing PM2.5) استفاده شده است. این دیتاست شامل اندازه‌گیری‌های کیفیت هوا در طول زمان بوده و معمولاً شامل ویژگی‌هایی مانند:

PM2.5 (ذرات معلق با قطر کمتر از ۲.۵ میکرون)

PM10

دما (Temperature)

فشار هوا (Pressure)

رطوبت نسبی (Relative Humidity)

سرعت و جهت باد

تاریخ و زمان

نوع داده‌ها: زمان-س्रی و عددی پیوسته

تعداد نمونه‌ها: حدوداً بین ۴۰,۰۰۰ تا ۴۵,۰۰۰ رکورد (بسته به نسخه دیتاست)

تعداد ویژگی‌ها: حدوداً ۱۲ ویژگی ورودی + برچسب طبقه‌بندی شده کیفیت هوا

۱-۳-۳- مدل استفاده شده

مدل اصلی مقاله، یک ماشین بردار پشتیبان (SVM) است که با استفاده از یک الگوریتم بهینه‌سازی ترکیبی به نام Differential Gravitational Fireworks Algorithm (DGFA) بهینه شده است.

ویژگی‌های نوآورانه:

DGFA ترکیبی از سه الگوریتم است:

Differential Evolution (DE): برای جستجوی بهینه جهانی در فضای هایپرپارامترها.

Gravitational Search Algorithm (GSA): الگوریتم مبتنی بر جاذبه فیزیکی برای همگرایی دقیق‌تر.

الگوریتم انفجاری برای اکتشاف محلی دقیق :Fireworks Algorithm (FWA)

هایپرپارامترهای SVM مانند C ، γ (گاما) در کرنل RBF، و نوع کرنل توسط این الگوریتم ترکیبی بهینه شده‌اند.

- ۱-۳-۴ نتایج

مدل بهینه‌شده با DGFA در مقایسه با:
استاندارد SVM
DE با SVM
PSO با SVM
GA با SVM

نتایج نشان داده‌اند که مدل DGFA-SVM دقت بالاتری در دسته‌بندی کیفیت هوا دارد (تا ~٪۹۳). سرعت همگرایی مناسبی دارد، به خصوص در مقایسه با GA پیچیدگی محاسباتی متوسط است، ولی در برابر دقت، قابل قبول است.

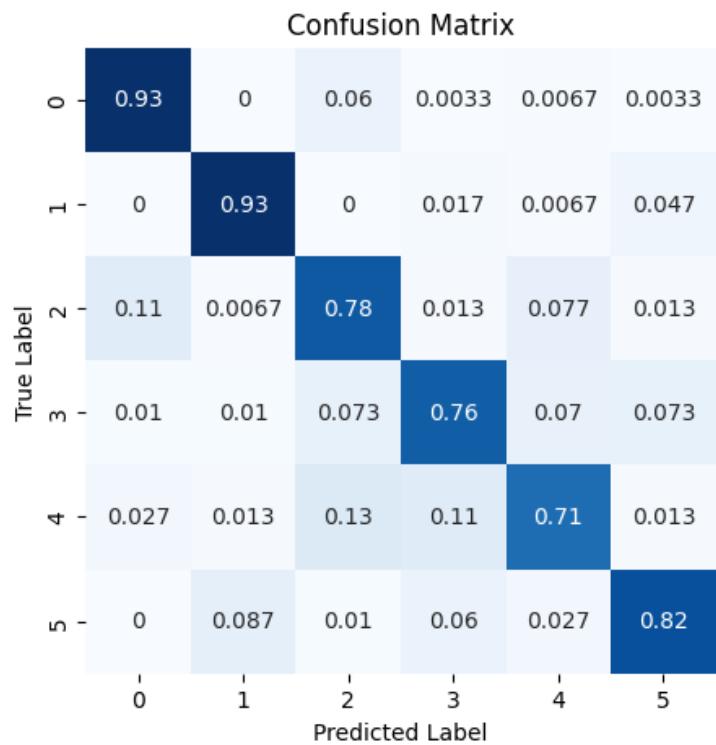
- ۱-۳-۵ پیاده‌سازی

```
bounds = [(0.1, 100),      # C
           (1e-2, 1)]       # gamma

result = differential_evolution(objective, bounds, strategy='best1bin',
                                  maxiter=20, popsize=15, tol=1e-4, seed=42)
```

```
Best parameters found by DE:
C = 34.0088, gamma = 0.051721
```

قابل ذکر است که مقدار C بهینه بدست آمده از این روش کمتر از روش pso است. این بدان معناست که مدل generalization بهتری نسبت به مدل pso دارد. اما باید در نظر داشت که هزینه محاسبات در روش DE بسیار بالا بود و باعث می‌شود تا در مقایسه با الگوریتم pso، این روش بسیار کند همگرا شود.



	precision	recall	f1-score	support
0	0.87	0.93	0.90	300
1	0.89	0.93	0.91	300
2	0.74	0.78	0.76	300
3	0.79	0.76	0.78	300
4	0.79	0.71	0.75	300
5	0.84	0.82	0.83	300
accuracy			0.82	1800
macro avg	0.82	0.82	0.82	1800
weighted avg	0.82	0.82	0.82	1800

محصول نهایی - ۱-۳-۶

• چه معیارهایی برای ارزیابی عملکرد این روش‌ها استفاده می‌شود؟

از جمله رایج‌ترین معیارهای ارزیابی عملکرد مدل‌های یادگیری ماشین به‌ویژه در دسته‌بندی می‌توان به موارد زیر اشاره کرد:

Accuracy (دقت): نسبت نمونه‌هایی که به درستی طبقه‌بندی شده‌اند به کل نمونه‌ها.

Recall (بازخوانی): توانایی مدل در شناسایی نمونه‌های مثبت واقعی.

Precision (دقت مثبت): نسبت نمونه‌های مثبت پیش‌بینی شده که واقعاً مثبت هستند.

F1-Score: میانگین هارمونیک precision و recall، مناسب برای داده‌های نامتوازن.

AUC-ROC: مساحت زیر منحنی ROC، برای بررسی توانایی مدل در تمایز بین کلاس‌ها.

• در چه شرایطی هر معیار مناسب‌تر است؟

Accuracy زمانی که داده‌ها متوازن هستند و خطاهای مثبت و منفی اهمیت یکسانی دارند.

Recall زمانی که نادیده گرفتن نمونه‌های مثبت خطرناک است (مثل تشخیص بیماری).

Precision زمانی که هزینه اشتباه مثبت بالا است (مثل فیلتر اسپم).

F1-Score زمانی که داده نامتوازن است و نیاز به تعادل بین precision و recall وجود دارد.

AUC-ROC زمانی که عملکرد مدل در تمایز بین کلاس‌ها مهم است، به‌ویژه در مقایسه چند مدل.

• آیا ترکیب مدل‌های SVM با روش‌های مختلف بهینه‌سازی می‌تواند باعث بهبود عملکرد شود؟

بله. ترکیب SVM با الگوریتم‌های بهینه‌سازی مانند:

Genetic Algorithm (GA)

Gravitational Search Algorithm (GSA)

Firefly Algorithm (FA)

می‌تواند در تنظیم بهینه‌ی پارامترهای SVM (C و γ) نقش مؤثری داشته و منجر به بهبود عملکرد مدل در شرایط مختلف شود.

• آیا این روش‌ها در شرایط مختلف داده مانند داده‌های نامتوازن یا نویزی کاربرد پذیر هستند؟

بله. الگوریتم‌های ترکیبی و بهینه‌سازی می‌توانند باعث شوند مدل‌های SVM با دقت بیشتری در شرایط زیر عمل کنند:

داده‌های نامتوازن: با استفاده از معیارهایی مثل F1 و تکنیک‌هایی مثل oversampling (SMOTE) یا weighting.

داده‌های نویزی: استفاده از soft margin SVM و فیلترهای پیش‌پردازش برای کاهش اثر نویز.

فصل ۲ - کاهش ابعاد

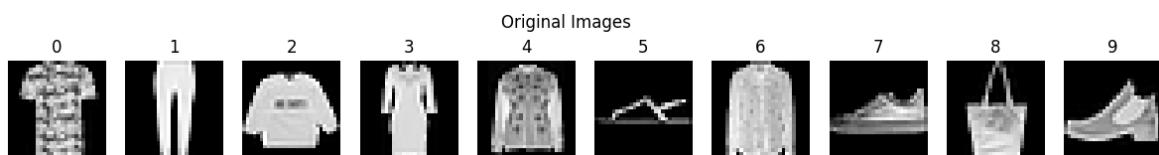
۲-۱ سوالات

دادگان fashion-mnist را بارگزاری می‌کنیم. در این سوال از فایل train این مجموعه داده که دارای ۶۰۰۰۰ تصویر ۲۸ در ۲۸ است استفاده می‌کنیم.

label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	2	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
1	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
2	6	0	0	0	0	0	0	0	5	0	0	0	0	30	43	0	0	0	0	
3	0	0	0	0	1	2	0	0	0	...	3	0	0	0	0	1	0	0	0	
4	3	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
...	
59995	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
59996	1	0	0	0	0	0	0	0	0	...	73	0	0	0	0	0	0	0	0	
59997	8	0	0	0	0	0	0	0	0	...	160	162	163	135	94	0	0	0	0	
59998	8	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
59999	7	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	

(الف)

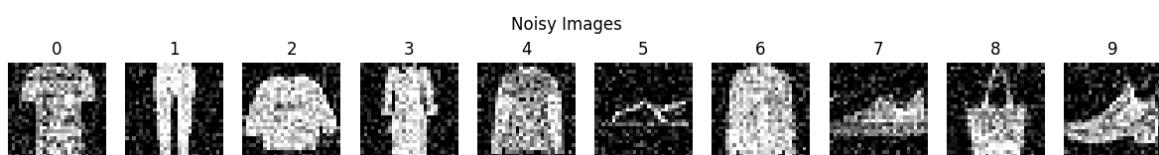
تصاویر اولیه از هر کلاس را به صورت رديفی نشان می‌دهیم:



سپس با اضافه کردن نویز، تصاویر نویزی را نمایش می‌دهیم:

```
noise = np.random.normal(loc=0.0, scale=0.2, size=selected_images.shape)
noisy_images = selected_images + noise
noisy_images = np.clip(noisy_images, 0.0, 1.0)
```

تصاویر نویزی:



(ب)

الگوریتم PCA:

```
def pca(X, n_components=2):
    X_meaned = X - np.mean(X, axis=0)

    cov_matrix = np.cov(X_meaned, rowvar=False)

    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

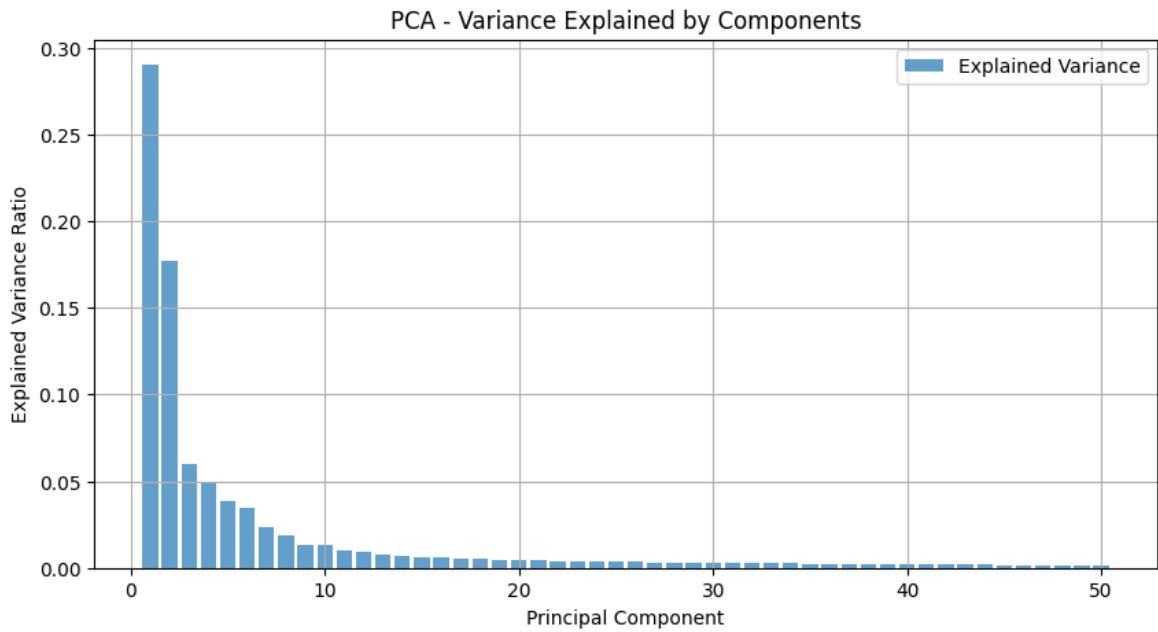
    sorted_idx = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[sorted_idx]
    eigenvectors = eigenvectors[:, sorted_idx]

    eigenvectors_subset = eigenvectors[:, :n_components]

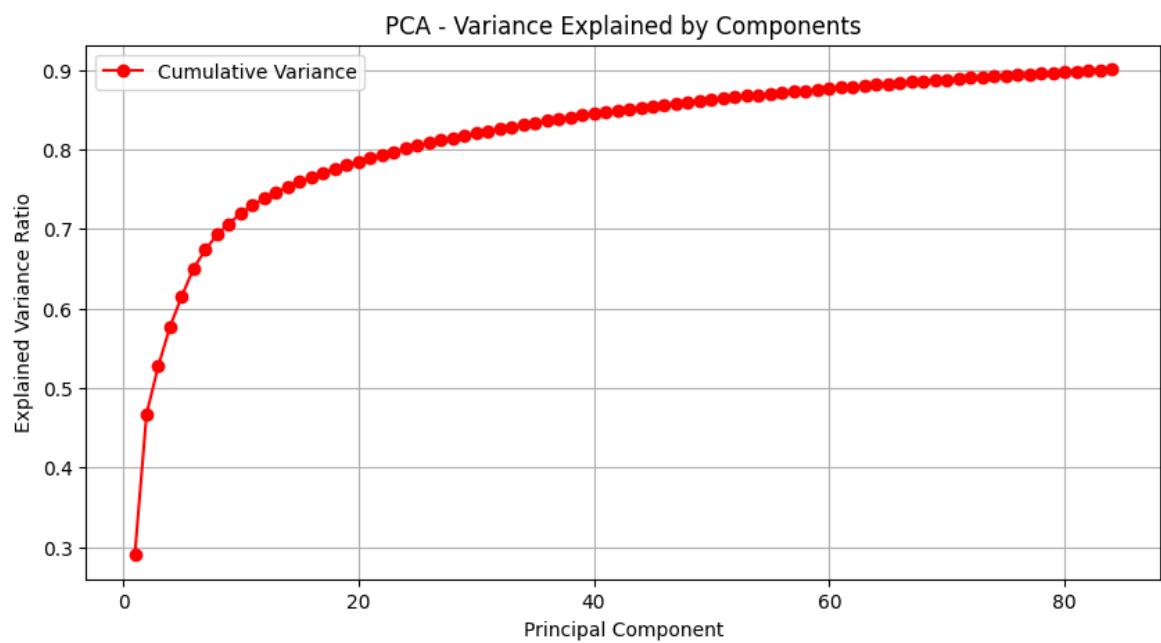
    X_reduced = np.dot(X_meaned, eigenvectors_subset)

    return X_reduced, eigenvalues[:n_components], eigenvectors_subset
```

واریانس مولفه‌های اصلی به صورت زیر خواهد بود:

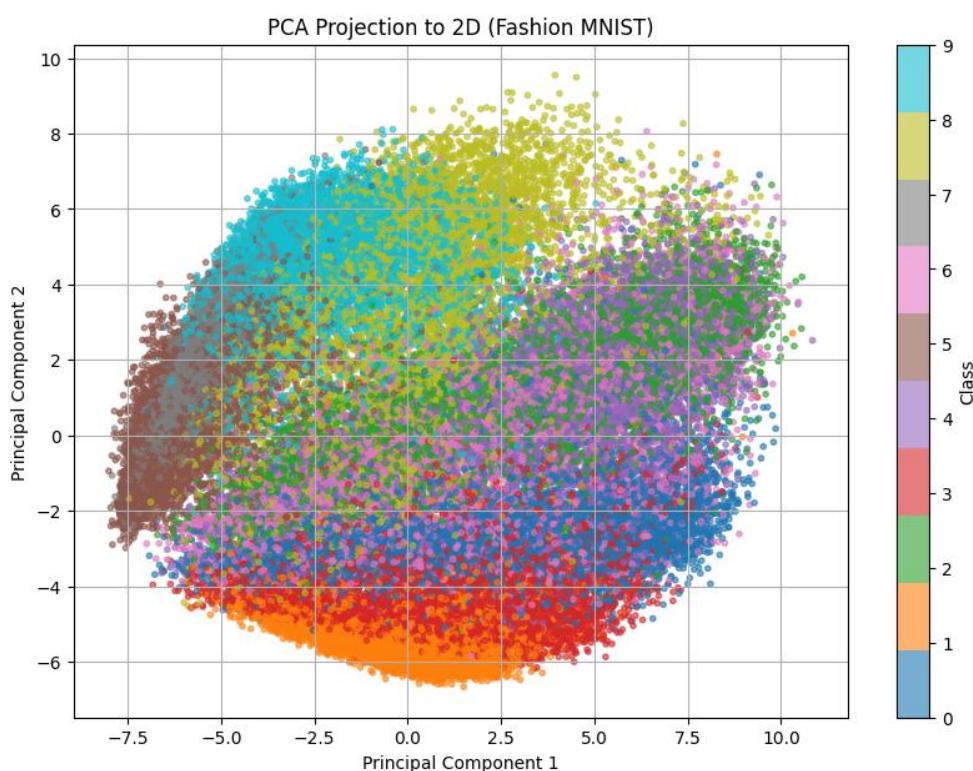


واریانس تجمعی مولفه‌های اصلی نیز به صورت زیر خواهد بود. تعداد مولفه مورد نیاز برای حفظ حداقل ۹۰ درصد واریانس برای این دادگان برابر ۸۴ است. در نمودار نیز قابل مشاهده است که میزان واریانس تجمعی مولفه‌های اصلی در تعداد ۸۴، برابر ۹۰ درصد خواهد شد:



(ج)

با استفاده از کتابخانه `sklearn`، الگوریتم `pca` را اجرا می‌کنیم. خروجی دادگان در ۲ بعد به صورت زیر است:

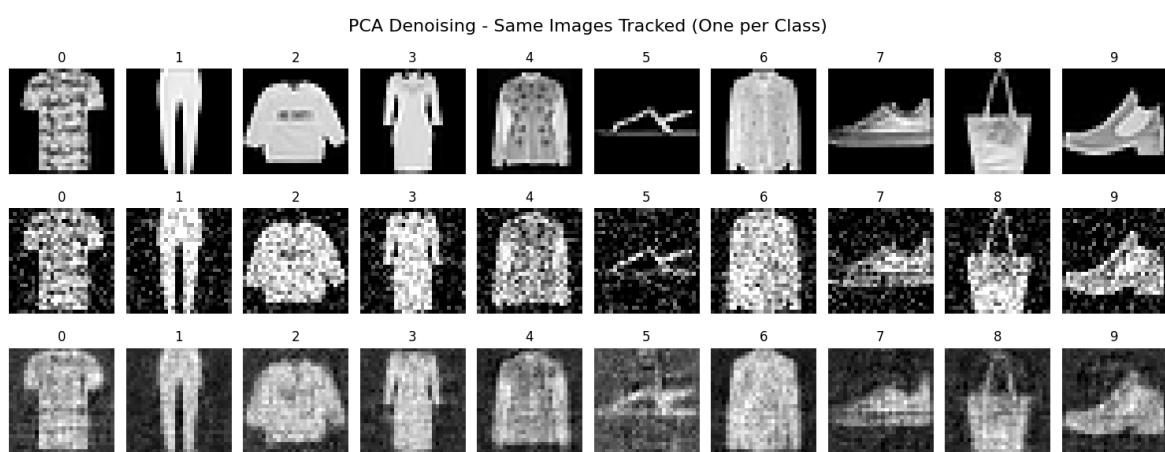


(۴)

به صورت زیر نویز را به تصاویر اصلی اضافه می‌کنیم و سپس تلاش می‌کنیم تا تصاویر را از روی تصاویر نویزی بازسازی کنیم:

```
noise = np.random.normal(0.0, 0.2, selected_images.shape)
noisy_images = selected_images + noise
noisy_images = np.clip(noisy_images, 0.0, 1.0)

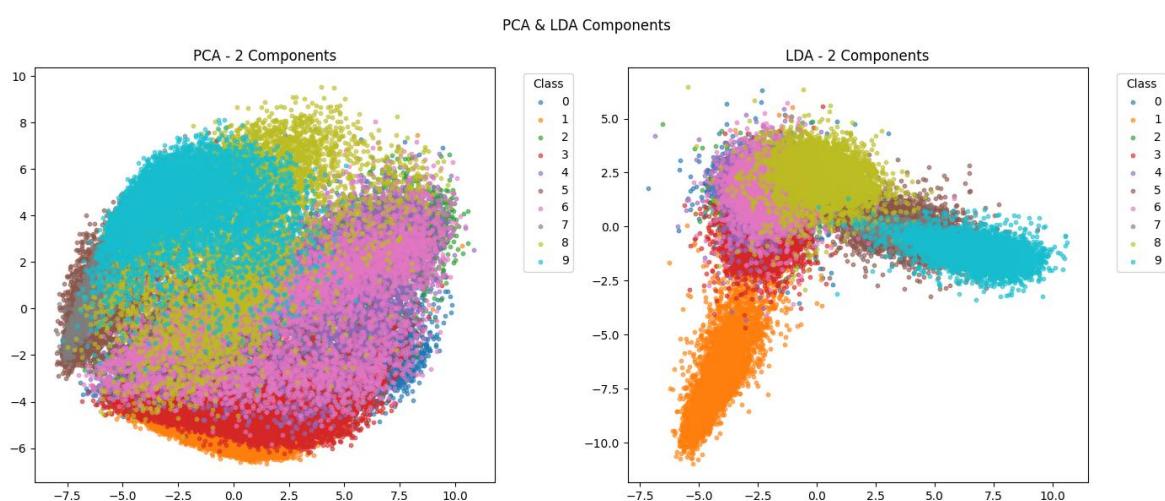
pca = PCA(n_components=100)
compressed = pca.fit_transform(noisy_images)
reconstructed = pca.inverse_transform(compressed)
```



تصاویر رديف اول اصلی، رديف دوم نویزی و رديف سوم بازسازی شده با ۱۰۰ مولفه هستند.

(۵)

تصاویر مقایسه دو الگوریتم PCA و LDA با یکدیگر



تحلیل و مقایسه:

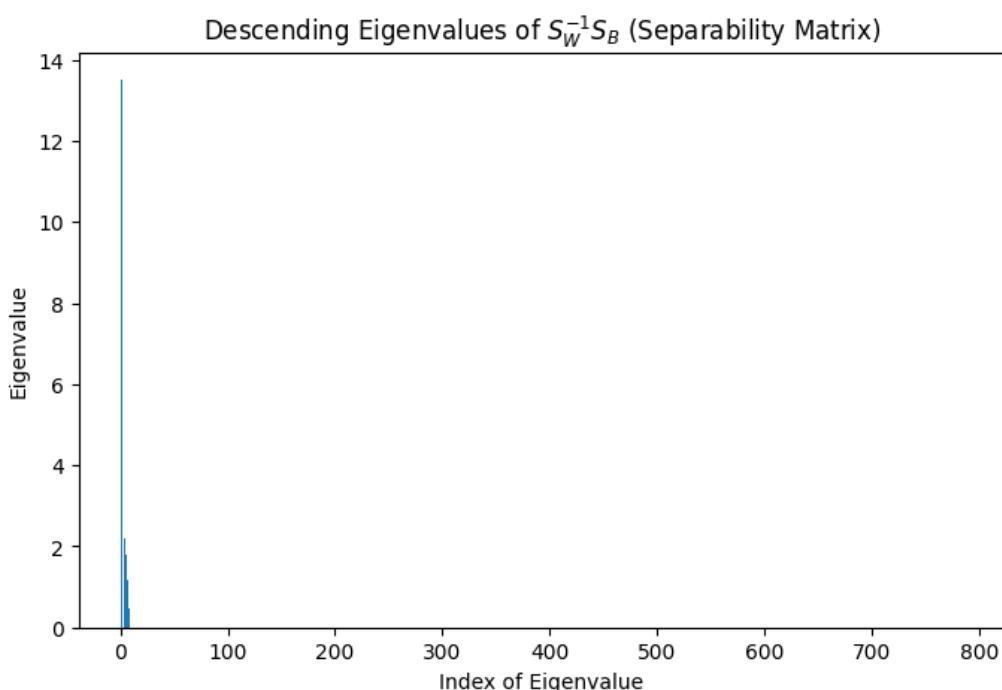
PCA	LDA
حفظ بیشترین واریانس کل	بیشترین جداسازی بین کلاس‌ها
بدون ناظارت (unsupervised)	با ناظارت (supervised)
نمایش ساختار کلی داده‌ها	تفکیک بهتر کلاس‌ها در فضای جدید
کلاس‌ها ممکن است همپوشانی داشته باشند	کلاس‌ها بهتر تفکیک شده‌اند

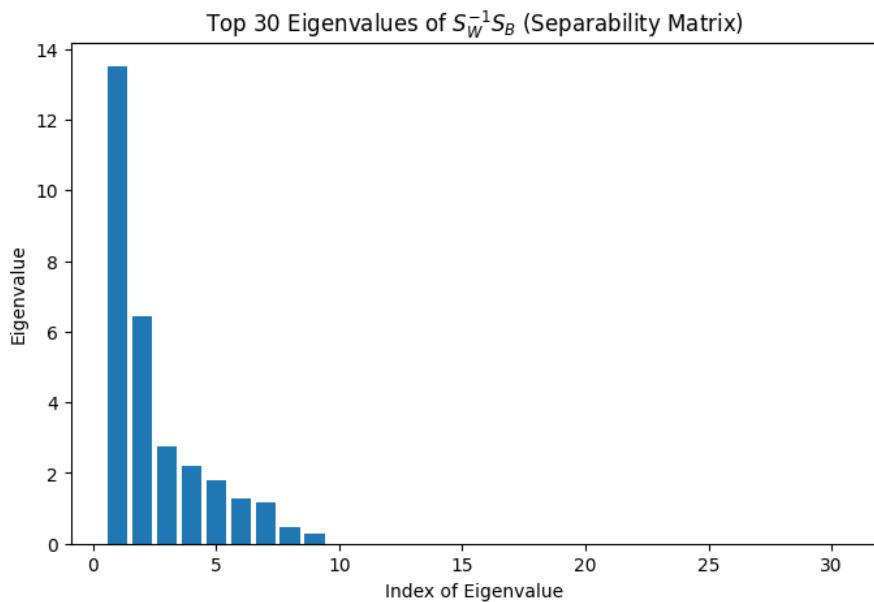
نتیجه:

- اگر هدف فشرده‌سازی داده‌ها بدون توجه به برچسب باشد، PCA گزینه خوبی است.
- اما اگر هدف تفکیک کلاس‌ها و طبقه‌بندی بهتر باشد، LDA عملکرد بهتری خواهد داشت.

(و)

همان‌طور که مشخص است، فقط تعداد بسیار کمی از مقادیر ویژه مقدار قابل توجهی دارند، و بقیه تقریباً صفر هستند. این ویژگی معمول در تحلیل تشخیص خطی (LDA) است.



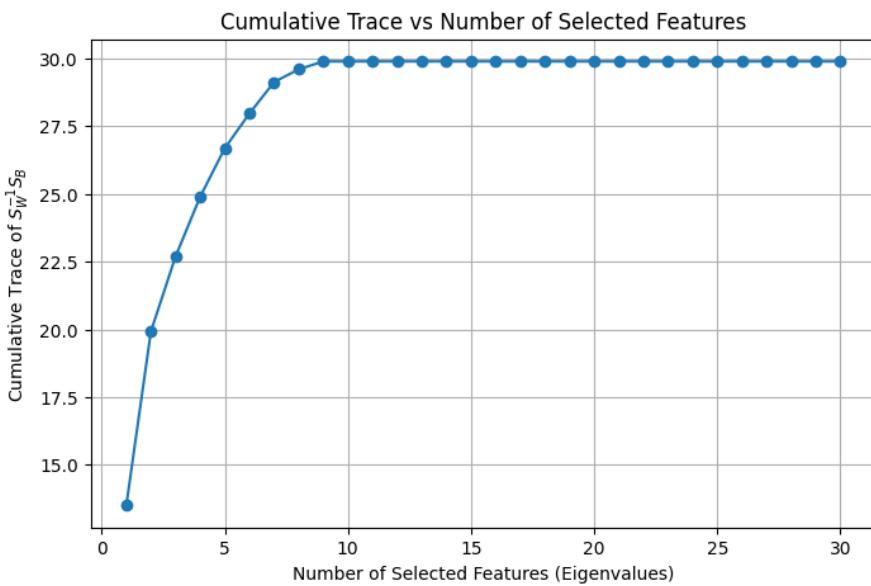
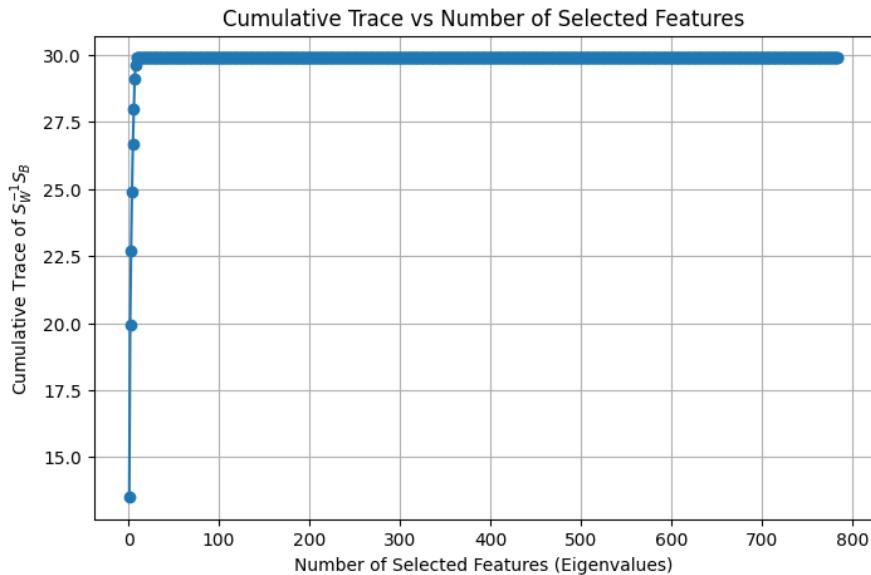


(ز)

برای انجام این تحلیل، ابتدا باید مقدار ردیابی (trace) ماتریس جدایی‌بزیری را نسبت به تعداد ویژگی‌های انتخاب شده محاسبه کنیم. در تحلیل مؤلفه‌های خطی تفکیکی (LDA)، این به این معناست که مجموع مقادیر ویژه انتخاب شده (از بزرگترین به کوچک‌ترین) را بررسی کنیم.

گام‌ها:

- محاسبه مقادیر ویژه مرتب شده نزولی (انجام شده).
- محاسبهٔ مقدار تجمعی Trace برای k مؤلفه‌ی اول
- رسم نمودار Trace بر حسب تعداد ویژگی‌های انتخاب شده
- تحلیل نقطه زانو (Elbow Point) برای یافتن تعداد بهینه ویژگی‌ها.



در ابتدای نمودار، با افزایش تعداد ویژگی‌ها (مؤلفه‌ها)، مقدار Trace به سرعت افزایش می‌یابد. دلیل آن این است که مقادیر ویژه ابتدایی بزرگ‌تر و اطلاعات تفکیکی بیشتری دارند.

در ادامه، نرخ افزایش کاهش می‌یابد و نمودار به سمت اشباع می‌رود، چون مقادیر ویژه بعدی تقریباً صفرند یا اطلاعات جدیدی نمی‌افزایند.

نقطه‌ی زانو (Elbow Point) جایی است که افزودن مؤلفه‌های بیشتر باعث افزایش قابل توجهی در Trace نمی‌شود. آنجا تعداد بهینه ویژگی‌هاست. در اینجا تعداً ۸ یا ۹ مناسب است.

۲-۲ امتیازی

t-SNE توضیح مفهومی

هدف:

تبديل داده‌های با ابعاد بالا (مثلاً ۷۸۴×۷۸۴ ویژگی در تصاویر MNIST) به ابعاد پایین‌تر (معمولًاً ۲ یا ۳) به‌گونه‌ای که ساختار محلی داده‌ها (همسایگی‌ها) حفظ شود.

مراحل کلی t-SNE:

محاسبه شباهت‌ها در فضای اصلی (high-dimensional)

- شباهت بین دو نمونه با یک احتمال تعریف می‌شود:

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

تعریف شباهت‌ها در فضای بعد پایین (low-dimensional)

- در فضای ۲بعدی یا ۳بعدی، از توزیع تی دانش‌آموز (Student-t distribution) استفاده می‌شود:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

کمینه‌سازی اختلاف بین p_{ij} و q_{ij} با تابع:

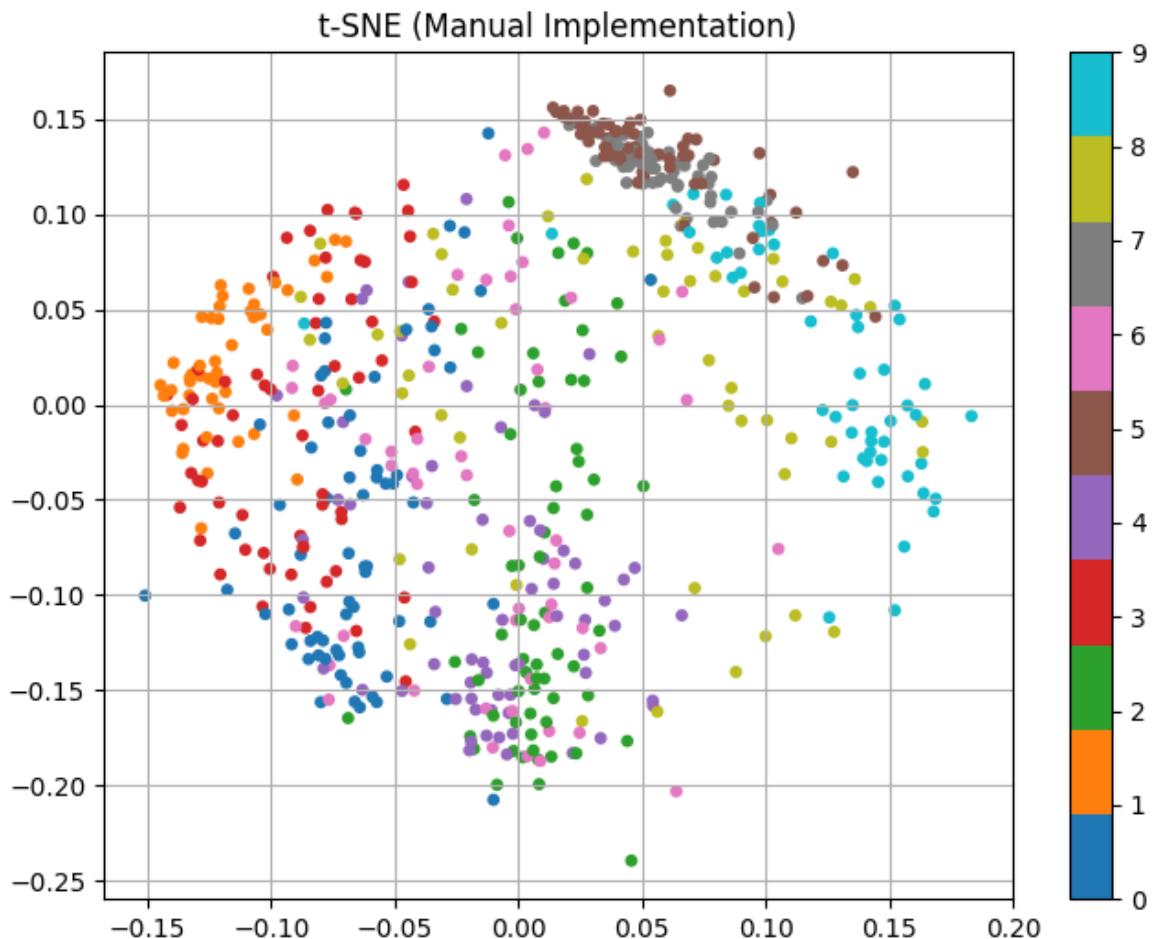
$$\text{KL}(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

به دلیل کندی الگوریتم scratch، تنها از ۶۰۰ نمونه استفاده می‌کنیم:

```
df_sample = df.sample(frac=0.01,random_state=24)
df_sample
✓ 0.0s
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
28815	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5819	0	0	0	0	0	0	0	0	0	37	...	28	103	1	0	0	0	0	0	0	0
44218	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8746	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
12352	2	0	0	0	0	0	0	0	0	0	...	1	1	0	0	136	180	120	0	0	0
...
39100	5	0	0	0	0	0	0	0	0	0	...	105	118	103	215	254	254	248	123	0	0
3391	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
41252	8	0	0	0	0	0	0	0	0	0	...	164	156	181	106	0	0	0	0	0	0
32312	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
32470	1	0	0	0	0	0	0	0	0	0	...	46	0	0	0	0	0	0	0	0	0

600 rows × 785 columns



ویژگی	PCA	LDA	t-SNE
نوع کاهش بعد	خطی	خطی	غیرخطی (حفظ همسایگی‌ها)
در نظر گرفتن برچسب کلاس	✗	✓	✗
بیشینه‌سازی جدایی کلاس‌ها بیشینه‌سازی واریانس کلی هدف			حفظ ساختار محلی
سرعت	سریع	نسبتاً سریع	کند (به خصوص برای داده‌های بزرگ)
پیش‌پردازش، فشرده‌سازی کاربرد		طبقه‌بندی	تجسم و تحلیل
تفسیرپذیری	بالا		پایین (ابعاد یادگرفته شده معنای مستقیم ندارند) بالا