دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی صنایع و سیستم‌های مدیریت

تمرین اول درس داده کاوی

نگارش

نیما محمودیان ۴۰۲۱۲۵۰۰۵

محسن فراغه ۴۰۱۲۲۵۰۰۲

معصومه بهبهانی زاده ۴۰۲۱۹۱۰۰۳

استاد

دکتر عباس احمدی

اسفند ۱۴۰۲

فهرست

فهرست اشکال

فهرست جداول

چکیده

در این مطالعه تحلیل بر روی مجموعه داده[1] سرطان سینه انجام شده است. داده‌های مورد بررسی از UCI[2] گرفته شده است. پس از معرفی مجموعه داده، فرآیند آماده سازی داده‌ها و بررسی مقادیر گمشده و پرت را در دستور کار داریم، سپس به استاندارد سازی داده‌ها با استفاده از روشهای شاخص z و مین مکس می‌پردازیم. در ادامه کار با استفاده از پنج روش فیلتر، جست و جوی رو به جلو، جست و جوی رو به عقب، جست و جوی رو به جلو-رو به عقب و الگوریتم ژنتیک، ویژگی‌های مورد نیاز را از بین ویژگی‌های موجود انتخاب می‌کنیم. در ادامه کار روش تحلیل مولفه اصلی نیز مورد بررسی قرار گرفت.

۱-معرفی مجموعه داده

مجموعه داده در دسترس برای این گزارش، بیماری سرطان سینه را مورد بررسی قرار می‌دهد. هدف از این دیتاست، بررسی و پیش‌بینی پیشرفت این بیماری می‌باشد. که به دو دسته خوش‌خیم (B[3]) و بدخیم (M[4]) تقسیم بندی می‌شوند.

ویژگی‌های مورد بررسی از یک تصویر دیجیتالی از توده پستان ایجاد شده است، که ویژگی‌های هسته موجود در سلول سرطان را توصیف می‌کنند. این دیتاست شامل ۳۰ ستون و ۵۶۹ سطر می‌باشد، یک ستون هدف و سایر ستون‌ها، ویژگی‌ها را نمایش می‌دهند. سطرها، هرکدام اطلاعات یک بیمار را نشان می‌دهند.

۱-۲ ویژگی‌ها

تمامی ویژگی‌های مورد بررسی و نوع هر یک را در جدول زیر مشاهده می‌کنید:

[1] Data Set

[2] مخزن داده‌های استاندارد دانشگاه کالیفرنیا

[3] Benign

[4] Malignant

جدول ۱-۱ معرفی ویژگی‌ها

Variable Name	Role	Type
D	ID	Categorical
Diagnosis	Target	Categorical
radius1	Feature	Continuous
texture1	Feature	Continuous
perimeter1	Feature	Continuous
area1	Feature	Continuous
smoothness1	Feature	Continuous
compactness1	Feature	Continuous
concavity1	Feature	Continuous
concave_points1	Feature	Continuous
symmetry1	Feature	Continuous
fractal_dimension1	Feature	Continuous
radius2	Feature	Continuous
texture2	Feature	Continuous
perimeter2	Feature	Continuous
area2	Feature	Continuous
smoothness2	Feature	Continuous
compactness2	Feature	Continuous
concavity2	Feature	Continuous
concave_points2	Feature	Continuous
symmetry2	Feature	Continuous
fractal_dimension2	Feature	Continuous
radius3	Feature	Continuous
texture3	Feature	Continuous
perimeter3	Feature	Continuous
area3	Feature	Continuous
smoothness3	Feature	Continuous
compactness3	Feature	Continuous
concavity3	Feature	Continuous
concave_points3	Feature	Continuous
symmetry3	Feature	Continuous
fractal_dimension3	Feature	Continuous

۲- فراخوانی داده‌ها

۲-۱ سوال

مجموعه داده breast cancer Wisconsin را از مخزن داده‌های اســـتلندارد دانشـــگاه کالیفرنیا UCI (Machine Learning Repository) انتخاب کنید. آنرا به دو قسمت مجموعه داده آموزشی (۶۰) train %و مجموعه داده آزمایشی (۴۰) test %تقسیم کنید.

الف ابتدا داده‌های گم شده و نویز را کنار بگذارید)با روش شش سیگما(و سپس استانداردسازی داده-ها)با روشهای شاخص Z و مین مکس(را انجام دهید. میانگین، انحراف معیار هر ویژگی را بدست آورید. همبستگی هر کدام از ویژگیها با یکدیگر و مشخصه هدف را گزارش کنید.

۲-۲ کدهای اولیه و فراخوانی داده‌ها

برای فراخوانی داده‌ها روش‌های متعددی از جمله استفاده از فایل CSV ، فراخوانی از گوگل کولب و ... وجود دارد، ولی در این جا ما به صورت مستقیم داده‌ها را از سایت UCI و فراخوانی کرده و از آن‌ها استفاده کرده‌ایم.

```python
from ucimlrepo import fetch_ucirepo
import pandas as pd
# fetch dataset
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# data (as pandas dataframes)
X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# metadata
print(breast_cancer_wisconsin_diagnostic.metadata)

# variable information
print(breast_cancer_wisconsin_diagnostic.variables)
```

شکل ۲-۱ کد فراخوانی داده‌ها

در ادامه یک نمای کلی از مجموعه داده را نمایش می‌دهیم:

جدول ۲-۲ نمای کلی مجموعه داده (الف)

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...	texture3	perimeter3
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20
...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16

569 rows × 31 columns

جدول ۲-۲ نمای کلی مجموعه داده (ب)

area3	smoothness3	compactness3	concavity3	concave_points3	symmetry3	fractal_dimension3	Diagnosis
2019.0	0.16220	0.66560	0.7119	0.2654	0.4601	0.11890	M
1956.0	0.12380	0.18660	0.2416	0.1860	0.2750	0.08902	M
1709.0	0.14440	0.42450	0.4504	0.2430	0.3613	0.08758	M
567.7	0.20980	0.86630	0.6869	0.2575	0.6638	0.17300	M
1575.0	0.13740	0.20500	0.4000	0.1625	0.2364	0.07678	M
...	...	...	...	...	...	...	...
2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115	M
1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637	M
1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820	M
1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400	M
268.6	0.08996	0.06444	0.0000	0.0000	0.2871	0.07039	B

برای ادامه کار، لازم است که نوع هر یک از داده‌ها را مشخص می‌کنیم، این مهم، یک دید کلی نسبت به ماهیت هر یک از ویژگی‌ها را در اختیار ما قرار می‌دهد. در این قسمت نوع داده‌های Diagnoisis را Object تشخیص می‌دهد، از این رو آن را به String تبدیل می‌کنیم. در این مجموعه داده سایر ویژگی‌ها نیازی به اصلاح ندارند. کد استفاده شده را در زیر قابل مشاهده است:

```
In [10]: df["Diagnosis"] = df["Diagnosis"].astype("string")

In [11]: df["Diagnosis"].dtypes
Out[11]: string[python]
```

شکل ۲-۲ کد اصلاح نوع داده

۲-۳ تحلیل توصیفی داده‌ها

در گذشته همانگونه که در شکل ۲-۳ مشاهده کردید، یک قالب داده تحت عنوان "df" از کل مجموعه داده‌ای که در اختیار داشتیم، ایجاد کردیم. برای تسلط بیشتر بر روی مجموعه داده اقدامات زیر را انجام می‌دهیم:

df.head : برای مشاهده پنج سطر اول از مجموعه داده استفاده می‌شود.

df.tail: برای مشاهده پنج سطر آخر از مجموعه داده استفاده می‌شود.

df.nunique: تعداد مقادیر منحصر به فرد را در هر ستون از چارچوب داده برمیگرداند. این به ما نشان می‌دهد که در هر ستون چند مقدار مختلف وجود دارد و داده‌ها چقدر متنوع هستند.

df.describe: تعداد داده‌های هر ستون، مقادیر میانگین، انحراف معیار، کمترین و بیشترین مقدار هر ستون و چارک‌ها را نمایش می‌دهد.

df.info : خلاصه‌ای از داده‌ها شامل فهرست، نام ستون‌ها، انواع داده‌ها، مقادیر غیرتهی، میزان استفاده از حافظه و سایر اطلاعات را چاپ می‌کند. این یک نمای کلی از چارچوب داده و ویژگی‌های آن به ما می‌دهد.

df.shape: تعداد ردیف‌ها و ستون‌ها را به ما برمی‌گرداند.

برخی از خروجی‌های کدهای بالا را مشاهده می‌کنید:

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...

8 rows × 30 columns

شکل ۲-۳ خروجی df.describe

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   radius1             569 non-null    float64
 1   texture1            569 non-null    float64
 2   perimeter1          569 non-null    float64
 3   area1               569 non-null    float64
 4   smoothness1         569 non-null    float64
 5   compactness1        569 non-null    float64
 6   concavity1          569 non-null    float64
 7   concave_points1     569 non-null    float64
 8   symmetry1           569 non-null    float64
 9   fractal_dimension1  569 non-null    float64
 10  radius2             569 non-null    float64
 11  texture2            569 non-null    float64
 12  perimeter2          569 non-null    float64
 13  area2               569 non-null    float64
 14  smoothness2         569 non-null    float64
 15  compactness2        569 non-null    float64
 16  concavity2          569 non-null    float64
 17  concave_points2     569 non-null    float64
```

شکل ۲-۴ خروجیdf.info (الف)

```
 18  symmetry2           569 non-null    float64
 19  fractal_dimension2  569 non-null    float64
 20  radius3             569 non-null    float64
 21  texture3            569 non-null    float64
 22  perimeter3          569 non-null    float64
 23  area3               569 non-null    float64
 24  smoothness3         569 non-null    float64
 25  compactness3        569 non-null    float64
 26  concavity3          569 non-null    float64
 27  concave_points3     569 non-null    float64
 28  symmetry3           569 non-null    float64
 29  fractal_dimension3  569 non-null    float64
 30  Diagnosis           569 non-null    object
dtypes: float64(30), object(1)
memory usage: 137.9+ KB
```

شکل ۲-۴ خروجیdf.info (ب)

۲-۴ پیش پردازش داده‌ها

۲-۴-۱مشخص کردن داده‌های تکراری

در کد شکل ۳-۱، باید اطمینان حاصل کنیم که هیچ داده تکراری در مجموعه داده ما وجود ندارد. اگر ستون‌های کاملا مشابهی در مجموعه داده ما وجود دارد، باید حذف شوند. که در مجموعه داده مورد بررسی ما طبق کد دستوری زیر، هیچ دو ستون مشابهی موجود نمی‌باشد.

```
In [9]: df.duplicated().sum()
Out[9]: 0
```

شکل ۲-۵ مشخص کردن داده‌های تکراری

۲-۴-۲ مشخص کردن مقادیر گمشده

یکی از مهم‌ترین چالش‌هایی که در آماده سازی داده‌ها با آن سر و کار داریم، داشتن مقادیر گمشده می‌باشد. مقادیر گمشده ممکن است در پیش‌بینی‌های آینده یا در روند کار برای ما مشکلاتی را ایجاد کنند، از این رو شناسایی آن‌ها حائز اهمیت است. طبق بررسی صورت گرفته در این دیتاست، همانگونه که مشاهده می‌کنید، مقادیر گمشده‌ای موجود نمی‌باشد، از این رو به ادامه آماده سازی داده‌ها می‌پردازیم.

```
df.isnull().sum()
```

شکل ۲-۶ کد تشخیص داده‌های گم شده

```
radius1              0
texture1             0
perimeter1           0
area1                0
smoothness1          0
compactness1         0
concavity1           0
concave_points1      0
symmetry1            0
fractal_dimension1   0
radius2              0
texture2             0
perimeter2           0
area2                0
smoothness2          0
compactness2         0
concavity2           0
concave_points2      0
symmetry2            0
fractal_dimension2   0
radius3              0
texture3             0
perimeter3           0
area3                0
smoothness3          0
compactness3         0
concavity3           0
concave_points3      0
symmetry3            0
fractal_dimension3   0
Diagnosis            0
```

شکل ۲-۷ مقادیر گم شده در هر ستون

۲-۴-۳ شناسایی نویزها[5] و داده‌های پرت[6]

نویز، داده‌هایی را شامل می‌شود که به هر دلیلی دارای مقادیر نادرست یا غلط باشند. مانند عکس‌های تار در دوربین ثبت تخلف، تفاوت در واحد پول چند ویژگی مختلف

داده‌های پرت نیز به داده‌هایی گفته می‌شود که در فاصله‌ی غیرعادی از سایر مقادیر داده در یک نمونه‌ی تصادفی از یک جمعیت مشاهده می‌شود.

[5] Noise
[6] Outlier

شناسایی نویزها و داده‌های پرت امری حیاتی و تاثیرگذار در مبحث آماده سازی داده‌ها می‌باشد. از این رو باید در صدد رفع و شناسایی علت آن‌ها باشیم. روش‌های شناسایی این مشکلات عبارتند از:

- شناسایی نقطه پرت بر اساس مفهوم آماری پراکندگی
- استفاده از روش‌های خوشه بندی
- گسسته کردن داده‌ها

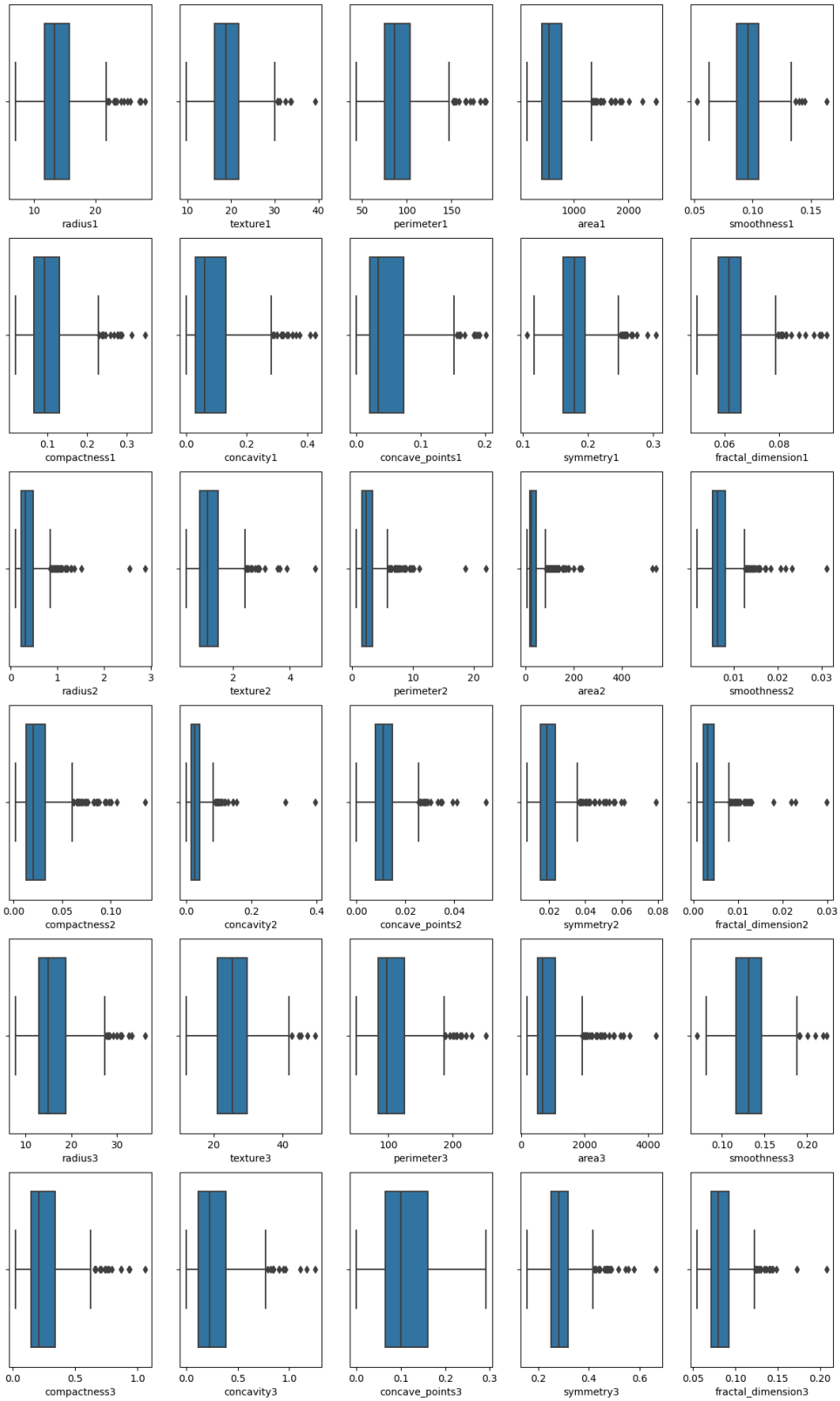
در این پژوهش، برای رفع نویزها از روش شش سیگما[7] استفاده کرده‌ایم. این روش داده‌هایی را که در محدوده شش سیگما نیستند را حذف میکند.

در ابتدا نمودار جعبه‌ای[8] مربوط به مجموعه داده سرطان سینه را مشاهده می‌کنید(شکل ۲-۹). این نمودار دید تجربی بسیار خوبی برای مشاهده میزان پرت بودن داده‌های هر ستون در اختیار ما قرار می‌دهد. با استفاده از کد زیر این نمودار را رسم کرده‌ایم:

```
import seaborn as sns
# Calculate the number of rows and columns needed for subplots
num_rows = 6
num_cols = 5

# Create subplots
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 25))

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Plot boxplots for each column
for i, col in enumerate(df.columns):
    sns.boxplot(x=df[col], ax=axes[i])

# Hide any remaining empty subplots
for i in range(len(df.columns), len(axes)):
    axes[i].axis('off')


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

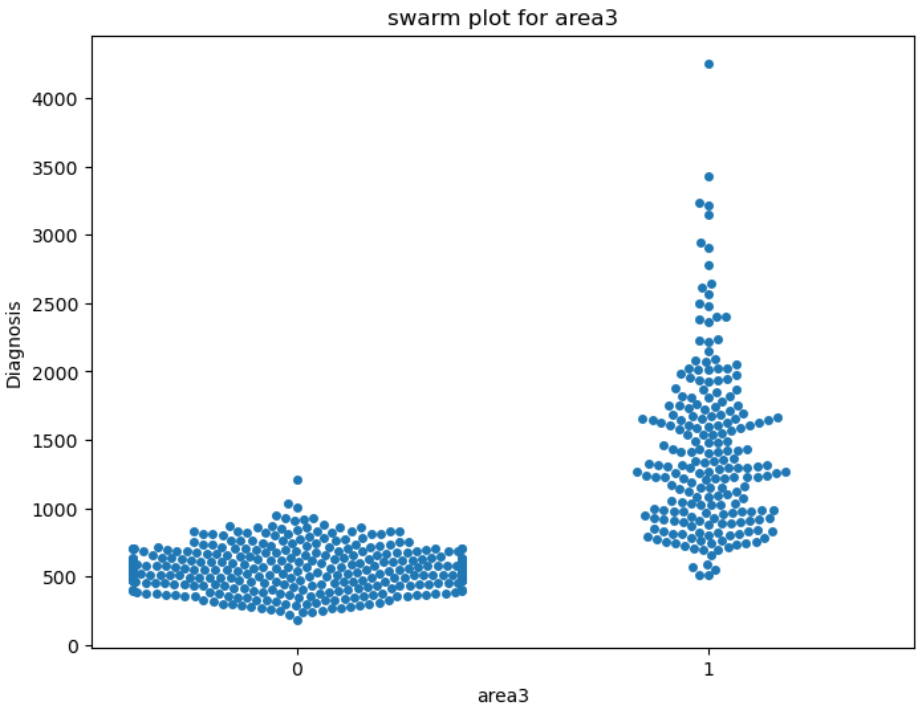
شکل ۲-۸ کد برای نمودار جعبه‌ای

[7] Six Sigma
[8] Box Plot



شکل۲-۹ نمودار جعبه‌ای

نمودار دیگری که در تشـــخیص بهتر داده‌های پرت و نهویزها به ما کمک میکند، نمودار پراکندگی[9] می‌باشـــد. برای نمونه و درک بهتر یکی از این نمودارها را در زیر آورده‌ایم:



شکل ۲-۱۰ نمودار پراکندگی

کد مربوط به رسم این نمودار نیز در زیر قابل مشاهده است.

```
for col in df.columns[:-1]:  # Exclude the label column
    plt.figure(figsize=(8, 6))
    sns.swarmplot(x=df["Diagnosis"], y=df[col])
    plt.title(f"swarm plot for {col}")
    plt.xlabel(col)
    plt.ylabel("Diagnosis")
    plt.show()
```

شکل ۲-۱۱ کد برای رسم نمودار پراکندگی

[9] Swarm Plot

همانگونه که در اشکال ۲-۹ و ۲-۱۰ قابل مشاهده است، ما با تعدادی داده نویز و پرت سر و کار داریم. با استفاده از کدهای زیر و روش شش سیگما، این داده‌ها را شناسایی می‌کنیم.

```
for col in df.columns:
    if col != "Diagnosis" :
      lower_limit = df[col].mean() -3*df[col].std()
      upper_limit = df[col].mean() +3*df[col].std()
      df = df[~((df[col] > upper_limit) | (df[col]< lower_limit))]
df
```

شکل ۲-۱۲ شناسایی داده نویز

در اینجا، تعداد ردیف‌های ما از ۵۶۹ به ۴۲۷ رسید (شکل ۲-۱۳) جدول داده‌ها پس از حذف نویزها را مشاهده می‌کنید:

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...	texture3	perimeter3
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.07017	0.1812	0.05667	...	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.12790	0.2069	0.05999	...	25.53	152.50
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.10430	0.1809	0.05883	...	16.67	152.20
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.07400	0.1794	0.05742	...	27.66	153.20
7	13.71	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.05985	0.2196	0.07451	...	28.14	110.60
...	...	...	...	...	...	...	...	...	...	...	...	...	...
556	10.16	19.59	64.73	311.7	0.10030	0.07504	0.005025	0.01116	0.1791	0.06331	...	22.88	67.88
558	14.59	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.03736	0.1454	0.06147	...	27.27	105.90
560	14.05	27.15	91.38	600.4	0.09929	0.11260	0.044620	0.04304	0.1537	0.06171	...	33.17	100.20
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.144000	0.09791	0.1752	0.05533	...	38.25	155.00
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.092510	0.05302	0.1590	0.05648	...	34.12	126.70

427 rows × 31 columns

شکل۲-۱۳ مجموعه داده بدون نویزها (الف)

area3	smoothness3	compactness3	concavity3	concave_points3	symmetry3	fractal_dimension3	Diagnosis
1956.0	0.1238	0.1866	0.24160	0.18600	0.2750	0.08902	1
1709.0	0.1444	0.4245	0.45040	0.24300	0.3613	0.08758	1
1575.0	0.1374	0.2050	0.40000	0.16250	0.2364	0.07678	1
1606.0	0.1442	0.2576	0.37840	0.19320	0.3063	0.08368	1
897.0	0.1654	0.3682	0.26780	0.15560	0.3196	0.11510	1
...	...	...	...	...	...	...	...
347.3	0.1265	0.1200	0.01005	0.02232	0.2262	0.06742	0
733.5	0.1026	0.3171	0.36620	0.11050	0.2258	0.08004	0
706.7	0.1241	0.2264	0.13260	0.10480	0.2250	0.08321	0
1731.0	0.1166	0.1922	0.32150	0.16280	0.2572	0.06637	1
1124.0	0.1139	0.3094	0.34030	0.14180	0.2218	0.07820	1

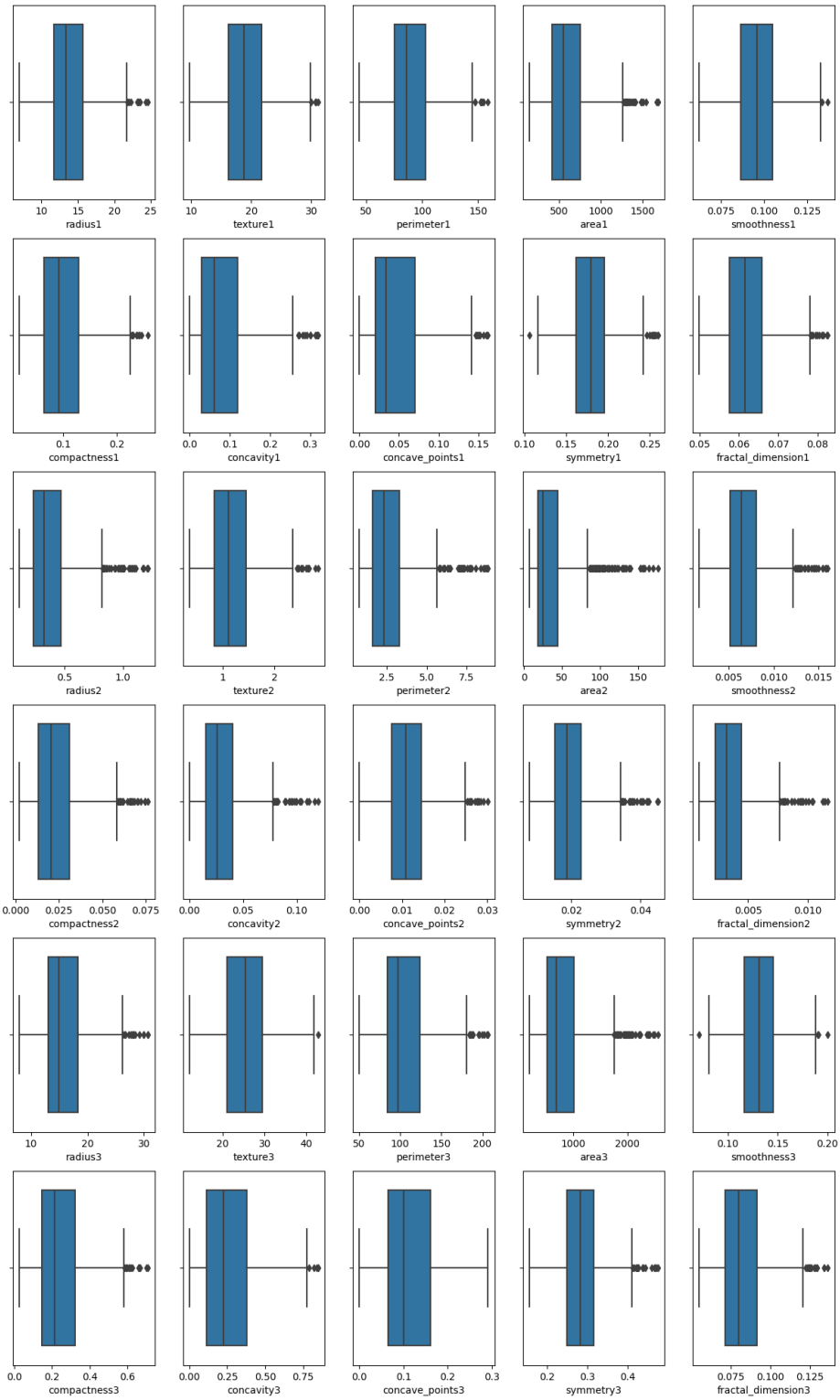
شکل۲-۱۳ مجموعه داده بدون نویزها (ب)

همانطور که میدانیم، پس از شناسایی نویزها، چندین رویکرد را در رابطه با رفتار با آن‌ها میتوانیم پیش بگیریم. ساده ترین آن‌ها حذف تمامی داده‌های نویز می‌باشد. رویکرد بعدی جایگزینی اعداد با استفاده از داده‌های بالا یا پایین داده‌ی نویز و رویکرد سوم، جایگذاری داده نویز با مقدار میانگین ستون مربوطه می‌باشد. که در این بخش به دلیل کم بودن تعداد داده‌ها، تصمیم گرفتیم که داده‌های نویز را با روش سوم جایگذاری کنیم.

```
        # Replace values outside the threshold with the mean of the column
        cleaned_df[col] = df[col].apply(lambda x: df[col].mean() if (x < lower_limit or x > upper_limit) else x)
cleaned_df
```

شکل ۲-۱۴ جایگزینی داده‌های نویز

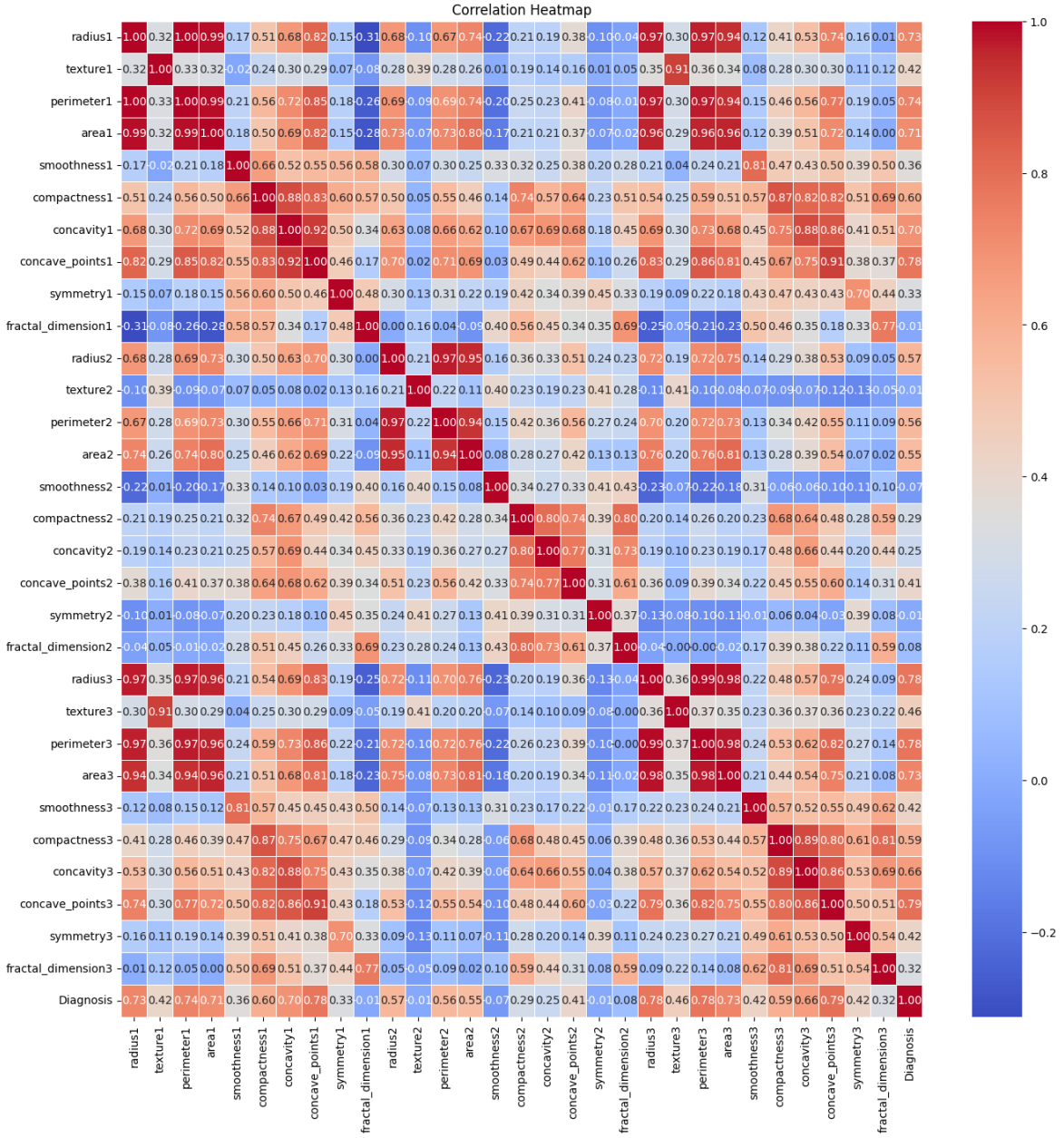
پس از جایگذاری داده‌ها با مقدار میانگین هر ستون، دوباره نمودار جعبه‌ای دیگری را رسم می‌کنیم. تفاوت در داده‌های پرت و عدم وجود آن‌ها را در مقایسه دو شکل ۲-۹ و ۲-۱۵ میتوانید احساس کنید.



شکل ۲-۱۵ نمودار جعبه‌ای

۲-۴-۴ همبستگی[10] بین داده‌ها

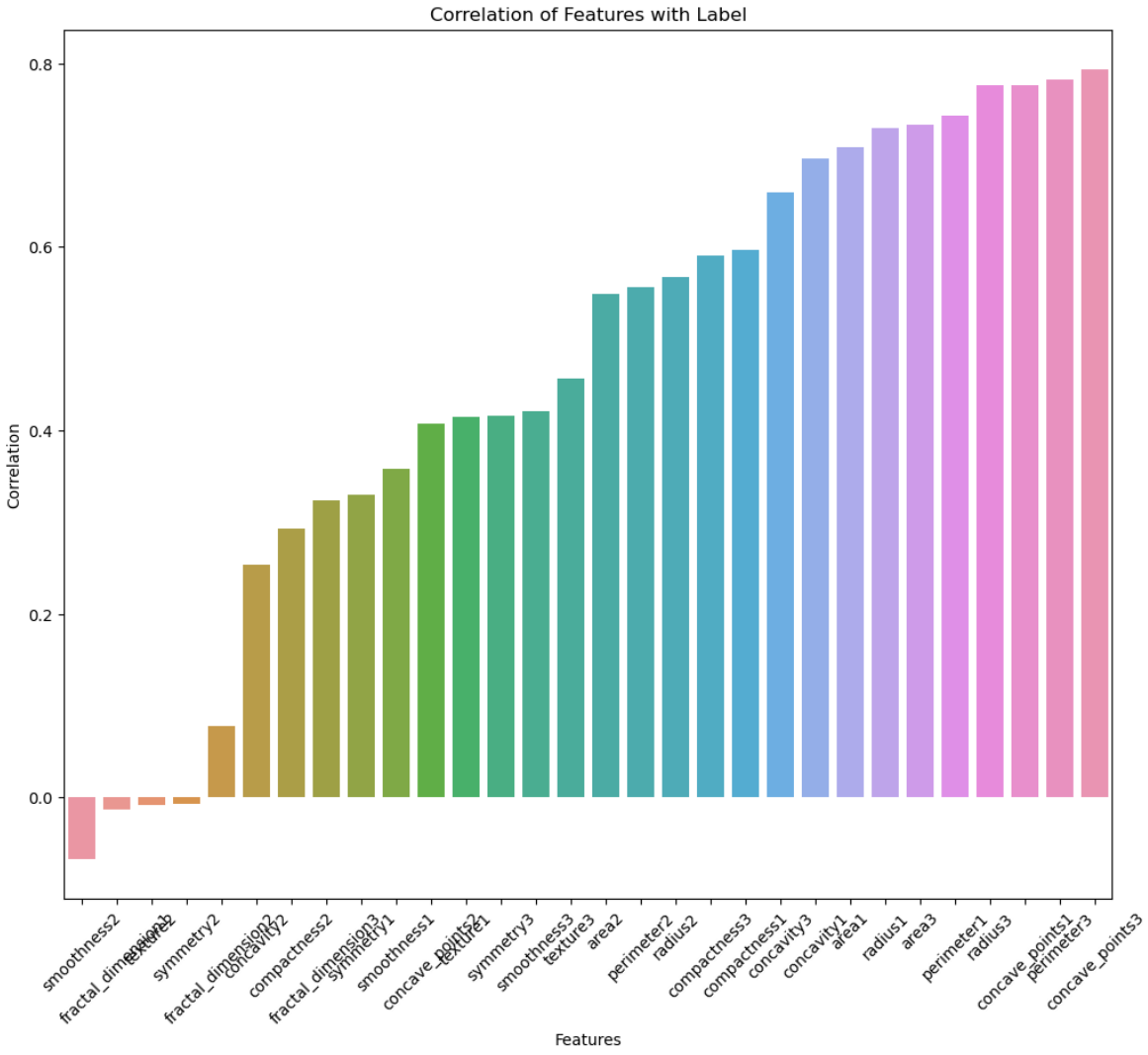
همبستگی، یک رابطه آماری بین دو متغیر تصادفی یا دو دسته داده است که لزوما به معنی ارتباط علی و معلولی آن‌ها نیست. همبستگی روابط بین متغیرها را به صورت دو به دو و جدا از تاثیر همزمان سایر متغیرها بررسی می‌کند. دو نمودار ۲-۱۶ و ۲-۱۷ برای مشاهده همبستگی ویژگی‌های مجموعه داده، رسم شده اند.



شکل ۲-۱۶ نمودار حرارتی همبستگی بین ویژگی‌ها

[10] Correlation

در نمودار حرارتی[11] رسم شده هر چه به سمت رنگ قرمز تیره پیش‌روی کنیم، مقدار همبستگی دو به دوی ویژگی‌ها بیشتر خواهد شد و هر چه به سمت آبی تیره پیش‌روی کنیم از این مقدار کاسته می‌شود تا به صفر برسد.



شکل ۲-۱۷ همبستگی بین ویژگی‌ها

[11] Heat map

همانطور که پیشتر مطرح گردید، ستون هدف ما شامل دو کلاس خوش‌خیم و بدخیم می‌باشد، برای به دست آوردن یک دید کلی نسبت به ستون هدف و همچنین چک کردن متوازن بودن یا نبودن دیتاست، بر اساس تعداد هر یک از کلاس‌ها، هیستوگرام رسم می‌کنیم.

۳ استاندارد سازی داده‌ها

۳-۱ استاندارد سازی به روش مین مکس[۱۲]

این روش، نوعی الگوریتم بهینه سازی است که برای یافتن بهترین راه حل ممکن برای یک مسئله معین با جست و جو در تمام راه حل‌های ممکن و انتخاب راه حل با بالترین ارزش یا کمترین هزینه استفاده می‌شود. با به حداقل رساندن حداکثر هزینه یا به حداکثر رساندن حداقل مقدار یک مجموعه معین از پارامترها کار می‌کند.

این روش استاندارد سازی با استفاده از تبدیل زیر به دست می‌آید:

$$x'_{ij} = \frac{x_{ij} - x_{\min,j}}{x_{\max,j} - x_{\min,j}} (x'_{\max,j} - x'_{\min,j}) + x'_{\min,j}$$

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the data
scaled_data = scaler.fit_transform(cleaned_df)

# Convert the scaled array back to a DataFrame
scaled_df = pd.DataFrame(scaled_data, columns=cleaned_df.columns)

scaled_df
```

شکل ۳-۱ روش مین مکس

با استفاده از کد شکل ۳-۱ ، داده‌های ما به روش مین مکس استاندارد می‌شوند. همانگونه که در جدول زیر مشاهده می‌کنید مقادیر هر یک از ستون‌ها در بازه صفر و یک قرار دارند.

[12] Min Max Method

جدول ۳-۱ داده‌های استاندارد شده به روش مین مکس

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...
0	0.623775	0.031294	0.686387	0.555916	0.749296	0.356649	0.937520	0.908025	0.884190	0.880551	...
1	0.769959	0.376460	0.774129	0.766613	0.298029	0.248762	0.271478	0.433148	0.489265	0.205513	...
2	0.720097	0.539000	0.748936	0.686872	0.631318	0.589875	0.616682	0.789506	0.656474	0.307198	...
3	0.251516	0.498365	0.293545	0.157277	0.453818	0.356649	0.754139	0.649383	1.000000	0.393189	...
4	0.754094	0.216254	0.793241	0.747812	0.506636	0.476115	0.618557	0.643827	0.487313	0.271669	...
...	...	...	...	...	...	...	...	...	...	...	...
564	0.826052	0.592247	0.853184	0.865802	0.650087	0.405172	0.761949	0.857407	0.433312	0.192037	...
565	0.745028	0.865950	0.759361	0.724473	0.473120	0.352699	0.449859	0.604383	0.450228	0.164472	...
566	0.545017	0.858010	0.560420	0.463274	0.295482	0.348082	0.289003	0.327284	0.344828	0.199694	...
567	0.771658	0.916394	0.836678	0.727066	0.741252	0.356649	0.277411	0.938272	0.869876	0.618683	...
568	0.044138	0.692667	0.035879	0.024311	0.453818	0.101755	0.000000	0.000000	0.342876	0.271975	...

۳-۲ استاندارد سازی به روش Z

روش کلی محاسبه، تعیین میانگین توزیع و انحراف استاندارد برای هر ویژگی است. سپس میانگین را از هر ویژگی کم می کنیم و مقادیر (میانگین قبلاً کم شده) هر ویژگی را بر انحراف معیار آن تقسیم می‌کنیم.

```
from sklearn.preprocessing import StandardScaler
# Initialize StandardScaler
scaler = StandardScaler()

# Fit and transform the data
standardized_data = scaler.fit_transform(cleaned_df)

# Convert the standardized array back to a DataFrame
standardized_df = pd.DataFrame(standardized_data, columns=cleaned_df.columns)

standardized_df
```

شکل ۳-۲ استاندارد سازی به روش z

در این بخش، پس از استاندارد سازی، داده‌ها مقادیر بین ۳+ تا ۳- سیگما را شامل می‌شوند.

جدول ۳-۲ داده‌های استاندارد شده به روش z

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...
0	1.199347	-2.152393	1.425768	1.191960	1.687718	0.062225	3.050377	2.761256	2.396453	2.604679	...
1	1.978077	-0.345027	1.877614	2.250745	-0.857119	-0.482851	0.037193	0.629055	0.035417	-0.923787	...
2	1.712463	0.506074	1.747876	1.850035	1.022401	1.240540	1.598904	2.229106	1.035065	-0.392276	...
3	-0.783698	0.293299	-0.597249	-0.811262	0.021423	0.062225	2.220762	1.599950	3.088816	0.057205	...
4	1.893563	-1.183899	1.976036	2.156269	0.319282	0.665794	1.607384	1.575006	0.023748	-0.577984	...
...	...	...	...	...	...	...	...	...	...	...	...
564	2.276892	0.784883	2.284723	2.749188	1.128247	0.307372	2.256095	2.533983	-0.299096	-0.994228	...
565	1.845270	2.218058	1.801561	2.038988	0.130272	0.042268	0.844195	1.397900	-0.197964	-1.138312	...
566	0.779799	2.176481	0.777078	0.726420	-0.871484	0.018939	0.116480	0.153723	-0.828092	-0.954205	...
567	1.987132	2.482192	2.199722	2.052019	1.642355	0.062225	0.064036	2.897065	2.310880	1.235878	...
568	-1.888408	1.310706	-1.924155	-1.479436	0.021423	-1.225567	-1.190976	-1.315785	-0.839762	-0.576383	...

569 rows × 31 columns

۴ -تقسیم بندی داده‌ها به دو دسته آموزشی و آزمایشی

یکی از کارهای پایه در یادگیری ماشین تقسیم داده به دو قسمت آموزش و تست می‌باشد. با داده‌های آموزش، مدل را آموزش می‌دهیم و با داده‌های تست، مدل آموزش یافته را تست می‌کنیم. در اینجا ۶۰ درصد از داده‌ها را به داده آموزشی و ۴۰ درصد از آن‌ها را به داده آزمایشی تقسیم بندی می‌کنیم.

```
from sklearn.model_selection import train_test_split
# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(scaled_df.drop("Diagnosis" , axis=1),
                                                  scaled_df["Diagnosis"], test_size=0.4, random_state=42)
```

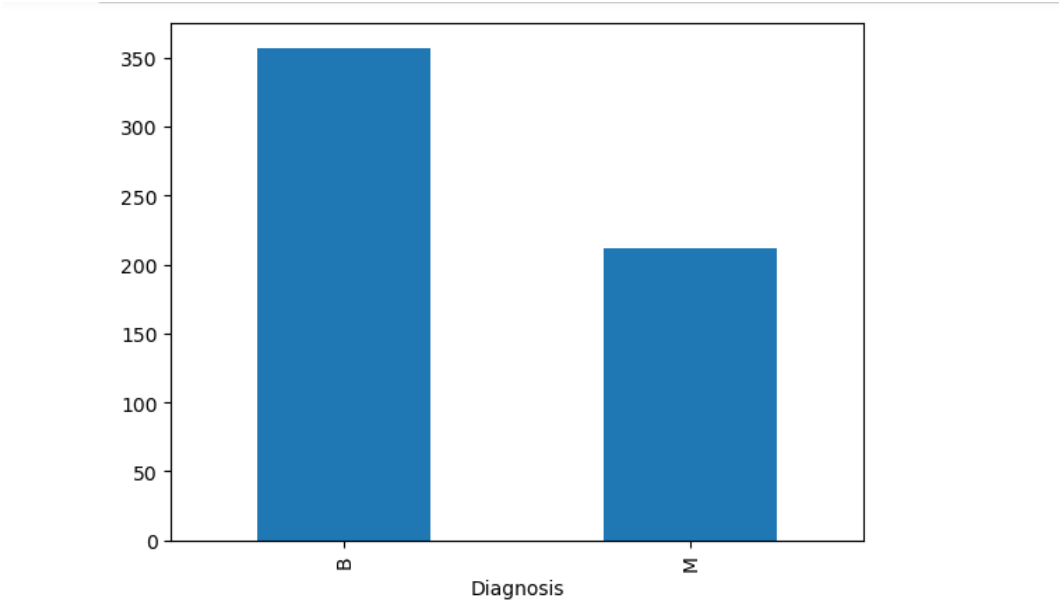
شکل ۴-۱ تقسیم بندی داده‌ها به داده‌های آموزشی و آزمایشی

۴-۱ متعادل کردن داده‌های آموزشی

کد درج شده در شکل ۴-۲ ، تعداد داده‌های هر کلاس را نمایش می‌دهد. همانگونه که مشاهده می‌کنید باتوجه به اختلاف موجود در بین تعداد داده‌های هر کلاس، داده‌ها در این بخش متعادل نیستند. برای ادامه کار با این مجموعه داده، نیازمند متعادل کردن داده‌ها هستیم.

```
In [13]: import matplotlib.pyplot as plt
         counts = df["Diagnosis"].value_counts()
         counts

Out[13]: Diagnosis
         B    357
         M    212
         Name: count, dtype: Int64
```

شکل ۴-۲ فراوانی هر کلاس



شکل ۴-۳ نمودار هیستوگرام

برای ایجاد تعادل در بین داده‌ها، دو رویکرد کلی وجود دارد:

۱.روش Over Sampling

برای کلاس اقلیت، به اندازه اختلاف بین دو کلاس، داده‌های مصنوعی ایجاد می‌کند. بیشتر برای زمانی کاربرد دارد که تعداد داده‌های در دسترس ما اندک است و یا برای انجام پیش‌بینی دقیق نیازمند تمامی داده‌های موجود باشیم.

۲. روش Under Sampling

در کلاس اکثریت، داده‌هایی که از میانگین دورتر هستند را حذف می‌کند تا به تعداد کلاس اقلیت برسد. بیشتر برای مواقعی کاربرد دارد که تعداد داده‌ها زیاد هستند و کاهش کلاس اقلیت، در مجموع دقت الگوریتم‌ها را کاهش نمی‌دهد.

در این پژوهش، به علت تعداد بالای داده‌ها، ما از روش under sampling برای متعادل سازی داده‌ها استفاده کرده‌ایم.

ابتدا کتابخانه زیر را فراخوانی کرده و در ادامه خواهیم داشت:

%pip install imbalanced-learn

```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(X_train, y_train)
X_resampled.shape
```

```
(264, 30)
```

شکل ۴-۴ متعادل سازی مجموعه داده

پس از متعادل کردن مجموعه داده، هر دو کلاس ما شامل ۱۳۲ سطر خواهند بود.

۵- روش‌های انتخاب ویژگی

۵-۱ سوال

ب) روش‌های کاهش ویژگی زیر را بر روی داده‌ها اعمال کنید و ویژگی‌های بدست آمده را ارائه، تحلیل و مقایسه (از نظر تعداد و کیفیت) کنید. هر جا لازم بود از روش دسته‌بندی نزدیکترین همسایگی (NN) استفاده کنید.

-روش فیلتر

-روش جستجوی رو به جلو

-روش جستجوی رو به عقب

-روش جستجوی ترکیبی روبه جلو و رو به عقب

-روش الگوریتم ژنتیک

۵-۲ انتخاب ویژگی

انتخاب ویژگی، به معنی انتخاب زیرمجموعه‌ای از ویژگی‌ها یا فاکتورهای مرتبط با مجموعه داده‌ها می‌باشد که بیشترین اطلاعات مفید را برای مدلسازی فراهم می‌کنند. زمانی که از روش انتخاب ویژگی استفاده می‌کنیم، تعداد ویژگی‌های ورودی به مدل کاهش پیدا می‌کند. دلیل استفاده از این روش این است که معمولاً مدل‌های یادگیری ماشین برای مجموعه داده‌ها با تعداد ویژگی‌های بالا ، به دلیل ابعاد بالا و مواجه شدن با عملیات محاسباتی زیاد و در نهایت نتایج ضعیف، می‌توانند با مشکل مواجه شوند. بنابراین، با کاهش تعداد ویژگی‌ها، می‌توانیم هر دو مشکل را حل کنیم. با این کار، میتوانیم از بار محاسباتی کمتری در مدلسازی استفاده کرده و در مقایسه با مجموعه داده‌های اولیه، مدل ایجاد شده با دقت بیشتری پیش‌بینی و پایداری داشته باشد.

در ابتدا برای مقایسه هر یک از روش‌های انتخاب ویژگی، یک تابع تعریف می‌کنیم که میزان دقت هر یک از روش‌ها را به آسانی محاسبه کند.

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
def accuracy_evaluation(xTrain, yTrain, xTest, yTest):
  model = KNeighborsClassifier(n_neighbors=1)
  model.fit(xTrain , yTrain)
  y_pred = model.predict(xTest)
  score = accuracy_score(yTest, y_pred)
  print(f'Accuracy is:{score}')
```

شکل ۵-۱ تابع دقت

۵-۳ روش فیلتر

روش‌های انتخاب ویژگی مبتنی بر فیلتر، از معیارهای آماری برای امتیاز دادن به همبستگی یا وابستگی بین متغیرهای ورودی و تعیین رابطه بین آن‌ها، استفاده می‌کنند تا آن‌ها را برای انتخاب مرتبط‌ترین ویژگی‌ها فیلتر نمایند. در این روش، برای هر ویژگی داده‌ای که مورد بررسی قرار می‌گیرد، یک معیار اندازه گیری تعریف می‌شود. سپس با استفاده از این معیارها، ویژگی‌هایی که ارتباط ضعیف‌تری با داده‌ها دارند حذف می‌شوند . واریانس اندازه‌گیری میزان تفاوت یک ویژگی در مقادیر آن است. ویژگی‌های با واریانس کم اطلاعات کمی دارند و برای مدل‌های یادگیری ماشین خیلی مفید نیستند.

```python
from sklearn.feature_selection import VarianceThreshold
import numpy as np

# Initialize VarianceThreshold with a threshold (e.g., 0.5)
threshold = 0.01
selector = VarianceThreshold(threshold)

# Fit the selector to your data
selector.fit(X_resampled)

# Get the indices of the features that are selected
selected_features_idx = selector.get_support(indices=True)

# Get the selected feature names
selected_features = X_resampled.columns[selected_features_idx]

# Filter the original DataFrame to keep only selected features
selected_data = X_resampled[selected_features]
selected_data
```

شکل ۵-۲ روش فیلتر

مشکل اصلی روش‌های نظارتی مبتنی بر فیلتر این است که گاهی دارای تعداد زیادی متغیر هستند. این متغیرها توسعه و آموزش مدل تصمیم‌گیری را کند کرده؛ به مقدار زیادی حافظه نیاز دارند و کیفیت عملکرد سیستم را پایین می‌آورند. در ادامه خواهیم دید که پس از اجرای روش فیلتر، تعداد ویژگی‌های انتخابی تنها یک عدد کاهش یافته است. درواقع فیلتر تمامی ویژگی‌ها، به جز یکی را انتخاب کرده است.

جدول ۵-۱ ویژگی انتخابی روش فیلتر

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...
220	0.377868	0.161140	0.383025	0.275786	0.455155	0.284317	0.121462	0.158210	0.195185	0.412864	...
361	0.358037	0.553947	0.360090	0.261005	0.312508	0.186172	0.104467	0.149630	0.491217	0.214395	...
388	0.243017	0.270434	0.257058	0.161102	0.283416	0.386282	0.314589	0.170185	0.487964	0.690965	...
480	0.293444	0.388603	0.299713	0.202139	0.380212	0.247670	0.091097	0.094259	0.262850	0.394487	...
189	0.301377	0.289117	0.304404	0.207585	0.245207	0.223113	0.120087	0.102099	0.394925	0.146401	...
...	...	...	...	...	...	...	...	...	...	...	...
87	0.682135	0.695002	0.679437	0.604538	0.372436	0.424901	0.458607	0.510556	0.581002	0.193874	...
330	0.512720	0.270901	0.538702	0.421199	0.434375	0.494165	0.376132	0.434630	0.469746	0.300153	...
214	0.408465	0.658571	0.426375	0.302885	0.430621	0.466879	0.348329	0.398889	0.764476	0.440123	...
121	0.661737	0.346100	0.674225	0.605186	0.575010	0.380405	0.455170	0.534877	0.589460	0.372741	...
435	0.396566	0.462868	0.411172	0.295624	0.583054	0.394257	0.351765	0.398951	0.396226	0.474119	...

264 rows × 30 columns

با این حال این روش از دقت خوبی برخوردار می‌باشد.

```
accuracy_evaluation(X_resampled[selected_features],y_resampled , X_val[selected_features], y_val)
```

```
Accuracy is:0.9210526315789473
```

شکل ۵-۳ دقت روش فیلتر

۵-۴ روش جست و جو رو به جلو

روش رو به جلو در انتخاب ویژگی تکنیکی است برای انتخاب زیرمجموع‌های از ویژگی‌ها از داده‌های اصلی که بیشترین ارتباط را برای کار پیش‌بینی دارند. ایده این است که با مجموعه‌ای خالی از ویژگی‌ها شروع کنید و هر بار یک ویژگی را بر اساس میزان بهبود عملکرد مدل اضافه کنید. این فرآیند تا زمانی تکرار میشود که با افزودن ویژگی‌های بیشتر، بهبود بیشتری حاصل نشود.

```python
from sklearn.neighbors import KNeighborsClassifier


# Initialize an empty set of selected features
selected_features = []
best_features = []
best_score = -float("inf")

# Define a stopping criterion (e.g., maximum number of features to select)

for feature in X_resampled.columns:
    # Initialize a model (e.g., K Nearest Neighbors)
    model = KNeighborsClassifier(n_neighbors=1)
    selected_features.append(feature)
    model.fit(X_resampled[selected_features] , y_resampled)
    # Evaluate model performance
    y_pred = model.predict(X_val[selected_features])
    score = accuracy_score(y_val, y_pred)
    # Check if this feature improves the model
    if score > best_score:
        best_score = score
        best_features.append(feature)
    else:
        break
```

شکل ۵-۴ روش جست و جو رو به جلو

ویژگی‌های انتخابی برای این روش عبارتند از:

```
['radius1', 'texture1', 'perimeter1']
```

```
accuracy_evaluation(X_resampled[best_features],y_resampled , X_val[best_features], y_val)
```

```
Accuracy is:0.8596491228070176
```

شکل ۵-۵ دقت روش جست و جو رو به جلو

۵-۵ روش جست و جو رو به عقب

روش رو به عقب در انتخاب ویژگی تکنیکی است برای انتخاب زیرمجموعه‌ای از ویژگی‌ها از داده‌های اصلی که بیشترین ارتباط را برای کار پیش بینی دارند. ایده این است که با مجموع‌های کامل از ویژگی‌ها شروع کنید و هر بار یک ویژگی را بر اساس میزان تأثیر آن بر عملکرد مدل حذف کنید. این روند تا زمانی تکرار می‌شود که با حذف ویژگی‌های بیشتر، بهبود بیشتری حاصل نشود.

```python
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Initialize an empty set of selected features
selected_features = []
best_features = []
best_score = -float("inf")

# Define a stopping criterion (e.g., maximum number of features to select)

for feature in X_resampled.columns[::-1]:
    # Initialize a model (e.g., K Nearest Neighbors)
    model = KNeighborsClassifier(n_neighbors=1)
    selected_features.append(feature)
    model.fit(X_resampled[selected_features] , y_resampled)
    # Evaluate model performance
    y_pred = model.predict(X_val[selected_features])
    score = accuracy_score(y_val, y_pred)
    # Check if this feature improves the model
    if score > best_score:
        best_score = score
        best_features.append(feature)
    else:
        break
```

شکل ۵-۶ روش جست و جو رو به عقب

ویژگی‌های انتخابی برای این روش عبارتند از:

```
['fractal_dimension3', 'symmetry3', 'concave_points3', 'concavity3']
```

```
accuracy_evaluation(X_resampled[best_features],y_resampled , X_val[best_features], y_val)
```

```
Accuracy is:0.8903508771929824
```

شکل ۵-۷ دقت روش جست و رو به عقب

۵-۶ روش جست و جو رو به جلو رو به عقب

در این روش، ابتدا با مجوعه خالی از مجموع ویژگی‌ها شـــروع می‌کنیم و زوش جســـت و جو رو به جلو را انجام می‌دهیم، سپس با مجموعه انتخابی از روش رو به جلو، روش رو به عقب را پیاده سازی می‌کنیم.

```
def forwardSelection(X_train, X_val, y_train, y_val):
    # Initialize an empty set of selected features
    selected_features = []
    best_features = []
    best_score = -float("inf")
    for feature in X_train.columns:
        # Initialize a model (e.g., K Nearest Neighbors)
        model = KNeighborsClassifier(n_neighbors=1)
        selected_features.append(feature)
        model.fit(X_train[selected_features] , y_train)
        # Evaluate model performance
        y_pred = model.predict(X_val[selected_features])
        score = accuracy_score(y_val, y_pred)
        # Check if this feature improves the model
        if score > best_score:
            best_score = score
            best_features.append(feature)
        else:
            break
    return best_features
```

شکل ۵-۸ روش جست و جو رو به جلو رو به عقب (الف)

```
forwardFeatures = forwardSelection(X_resampled, X_val, y_resampled, y_val)
reversedFeatures = list(reversed(forwardFeatures))
selected_features=[]
best_features=[]
best_score = -float("inf")
for feature in reversedFeatures:
    # Initialize a model (e.g., K Nearest Neighbors)
    model = KNeighborsClassifier(n_neighbors=1)
    selected_features.append(feature)
    model.fit(X_resampled[selected_features] , y_resampled)
    # Evaluate model performance
    y_pred = model.predict(X_val[selected_features])
    score = accuracy_score(y_val, y_pred)
    # Check if this feature improves the model
    if score > best_score:
        best_score = score
        best_features.append(feature)
    else:
        break
```

شکل ۵-۸ روش جست و جو رو به جلو رو به عقب (ب)

ویژگی‌های انتخابی برای این روش عبارتند از:

```
['perimeter1', 'texture1']
```

```
accuracy_evaluation(X_resampled[best_features],y_resampled , X_val[best_features], y_val)
```

```
Accuracy is:0.8596491228070176
```

شکل ۵-۹ دقت روش جست و جو رو به جلو روبه عقب

۵-۷ الگوریتم ژنتیک

الگوریتم ژنتیک (GA | Genetic Algorithms)، خانواده‌ای از «مدل‌های محاسباتی» (Computational Models) است که از مفهوم «تکامل» (Evolution) الهام گرفته شده‌اند. این دسته از الگوریتم‌ها، «جواب‌های محتمل» (Potential Solutions) یا «جواب‌های کاندید» (Candidate Solutions) و یا «فرضیه‌های محتمل» (Possible Hypothesis)

برای یک مسـأله خاص را در یک سـاختار داده‌ای «کروموزوم مانند» (Chromosome-like) کدبندی می‌کنند. الگوریتم ژنتیک از طریق اعمال «عملگرهای بازترکیب» (Recombination Operators) روی سـاختارهای داده‌ای کروموزوم مانند، اطلاعات حیاتی ذخیره شده در این ساختارهای داده‌ای را حفظ می‌کند.

پیاده‌سـازی الگوریتم ژنتیک، معمولا با تولید جمعیتی از کروموزوم‌ها (جمعیت اولیه از کروموزوم‌ها در الگوریتم‌های ژنتیک، معمولا تصادفی تولید می‌شود و مقید به حد بالا و پایین متغیرهای مسأله هستند) آغاز می‌شود. در مرحله بعد، ساختارهای داده‌ای تولید شـده (کروموزوم‌ها) ارزیابی می‌شـوند و کروموزوم‌هایی که به شـکل بهتری می‌توانند جواب بهینه (Optimal Solution) مسـأله مورد نظر (هدف) را نمایش دهند، شـانس بیشـتری برای تولید مثل نسـبت به جواب‌های ضـعیف‌تر پیدا می‌کنند. به عبارت دیگر، فرصـت‌های تولید مثل بیشـتری به این دسـته از کروموزوم‌ها اختصـاص داده می‌شـود. میزان خوب بودن یک جواب، معمولا نسبت به جمعیت جواب‌های کاندید فعلی سنجیده می‌شود.

برای اجرای این الگوریتم، ابتدا تابع‌های زیر را فراخوانی می‌کنیم:

```
%pip install sklearn-genetic
```

```
%pip install sklearn-genetic-opt
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn_genetic import GAFeatureSelectionCV
from sklearn_genetic.space import Categorical, Integer, Continuous



# Initialize a classifier (e.g., K Nearest Neighbors)
classifier = KNeighborsClassifier(n_neighbors=1)
# Initialize GeneticSelectionCV for feature selection
selector = GAFeatureSelectionCV(estimator=classifier,
                                scoring='accuracy',
                                population_size=50,
                                generations=20,
                                n_jobs=-1)
# Perform feature selection
selector.fit(X_resampled, y_resampled)
```

```
selected_df = scaled_df.loc[:,np.append(selector.best_features_ , True)]
selected_df
```

شکل ۵-۱۰ الگوریتم ژنتیک

الگوریتم ژنتیک، از بین تمامی ویژگی‌های موجود ۱۱ تا از آن‌ها را انتخاب می‌کند. دقت این الگوریتم را می‌توانید در شـکل ۵-۱۲ مشاهده کنید.

	texture1	concave_points1	texture2	smoothness2	radius3	texture3	smoothness3	compactness3	concavity3	concave_points3	Diagnosis
0	0.031294	0.908025	0.220171	0.327075	0.764680	0.172571	0.703315	0.936337	0.838615	0.912027	1.0
1	0.376460	0.433148	0.150941	0.245132	0.747590	0.370166	0.406629	0.233692	0.284604	0.639175	1.0
2	0.539000	0.789506	0.172348	0.309695	0.685364	0.439064	0.565788	0.582667	0.530569	0.835052	1.0
3	0.498365	0.649383	0.321431	0.516298	0.305872	0.470588	0.472832	0.332950	0.809165	0.884880	1.0
4	0.216254	0.643827	0.170086	0.682418	0.640228	0.151121	0.511705	0.260683	0.471198	0.558419	1.0
...	...	...	...	...	...	...	...	...	...	...	...
564	0.592247	0.857407	0.361822	0.599358	0.767748	0.467338	0.539519	0.269924	0.483803	0.761512	1.0
565	0.865950	0.604383	0.849342	0.283102	0.690622	0.852454	0.351001	0.241906	0.378725	0.559450	1.0
566	0.858010	0.327284	0.288715	0.292455	0.484224	0.718232	0.330140	0.413827	0.400872	0.487285	1.0
567	0.916394	0.938272	0.498748	0.335660	0.780456	0.890478	0.724948	0.332950	0.320637	0.910653	1.0
568	0.692667	0.000000	0.431295	0.382215	0.066871	0.596360	0.145175	0.054495	0.000000	0.000000	0.0

569 rows × 11 columns

شکل ۵-۱۱ ویژگی‌های انتخابی توسط الگوریتم ژنتیک

```
accuracy_evaluation(X_resampled.loc[:,selector.best_features_],y_resampled , X_val.loc[:,selector.best_features_], y_val)
```

```
Accuracy is:0.9342105263157895
```

شکل ۵-۱۲ دقت الگوریتم ژنتیک

۶- روش تحلیل مولفه اصلی(PCA)

۶-۱ مبنای ریاضی و توضیح روش PCA

PCA یک روش آماری است که برای کاهش ابعاد داده‌ها و استخراج اطلاعات مهم و معنادار از داده‌های چندمتغیره استفاده می‌شود.

مبنای ریاضی PCA به صورت زیر است:

محاسبه ماتریس کوواریانس:
ابتدا برای داده‌های ورودی، ماتریس کوواریانس را محاسبه می‌کنیم. این ماتریس نشان می‌دهد که چه میزان و چگونه متغیرها با یکدیگر همبستگی دارند.

محاسبه بردارهای ویژه و مقادیر ویژه:
با محاسبه بردارهای ویژه و مقادیر ویژه ماتریس کوواریانس، می‌توانیم مولفه‌های اصلی را استخراج کنیم. بردارهای ویژه نشان دهنده جهت‌هایی هستند که داده‌ها در آن‌ها بیشترین واریانس را دارند و مقادیر ویژه نشان دهنده میزان واریانس مربوط به هر مولفه است.

انتخاب تعداد مولفه‌ها:
با توجه به مقادیر ویژه، می‌توانیم تعداد مولفه‌هایی را که می‌خواهیم استفاده کنیم را تعیین کنیم. این تعداد معمولاً بر اساس مقادیر ویژه واریانس توضیح داده شده توسط مولفه‌ها انتخاب می‌شود. معمولاً مولفه‌هایی که مقادیر ویژه بزرگتری دارند، بیشترین اطلاعات را در خود جای داده‌اند.

تبدیل داده‌ها:
سپس داده‌ها را با استفاده از مولفه‌های اصلی تبدیل می‌کنیم. این تبدیل مجموعه‌ای از عملیات خطی است که داده‌ها را از فضای اولیه چندمتغیره به فضای جدید چندمتغیره منتقل می‌کند. در این فضا، متغیرهای جدید یعنی مولفه‌های اصلی، مستقل از یکدیگر هستند.

در حالت کلی پیش از به‌کارگیری روش تحلیل مولفه اصلی بهتر است داده‌ها را استاندارد کرده، به نحوی که مقادیر همه مشخصه‌ها در یک بازه مشخص قرار گیرد.

میانگین مشخصه‌های a_j با استفاده از تبدیل زیر مساوی صفر می‌شود:

$$\tilde{x}_{ij} = x_{ij} - \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$

(X: ماتریس به دست آمده از تبدیل بالا)

$\mathbf{V} = \mathbf{X}'\mathbf{X}$ ماتریس کواریانس مشخصه‌ها

۶-۲ کاهش ویژگی با روش مولفه اصلی (PCA):

در ادامه با توجه به توضیحات داده شده می‌خواهیم در داده‌هایی که داریم کاهش ویژگی را با روش مولفه اصلی اجرا کنیم. کدهای نوشته شده برای این بخش به شرح شکل ۶-۱ و شکل ۶-۲ می‌باشد که توضیحات آن به شرح زیر است:

```python
#standardize Data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaler = scaler.fit_transform(X_resampled)
X_test_scaler = scaler.transform(X_val)
pca = PCA(n_components = 30)
X_train_PCA = pca.fit_transform(X_train_scaler)
X_test_PCA = pca.transform(X_test_scaler)
print(X_train_PCA)


prop_var = pca.explained_variance_ratio_
print(prop_var)

PC_number = np.arange(pca.n_components_) + 1
print(PC_number)

# Get the feature names as an Index object
feature_names = list(X_resampled.columns)
print(feature_names)

#variance Cumulative
prop_var_ = np.array(prop_var)
Cumulative = prop_var_.cumsum()
z = pd.DataFrame(np.column_stack((feature_names,prop_var_, Cumulative)))
z.columns = ['feature_names', 'variance ratio', 'Cumulative']
print(z)
```

شکل ۶-۱ کدهای روش PCA قسمت ۱

scree plot

```python
plt.figure(figsize=(15,9))
plt.plot(PC_number,
         prop_var,
         'go-')
plt.title('Scree Plot',
          fontsize = 15)
plt.xlabel('component number',
           fontsize = 15)
plt.ylabel('proporation of variance',
           fontsize = 15)
plt.grid()
plt.show()

plt.figure(figsize=(15,9))
plt.plot(PC_number,
         prop_var,
         'go-')
plt.title('Scree Plot (n_component =9 & Cumulative = 0.90% )',
          fontsize = 15)
plt.xlabel('component number',
           fontsize = 15)
plt.ylabel('proporation of variance',
                fontsize = 15)
plt.axhline(y= 0.0172,
            color = 'blue',
            linestyle = '--')
plt.axvline(x=9,
            color='red',
            linestyle = '--')
plt.grid()
plt.show()

# Get the feature names as an Index object
feature_names = list(X_train.columns)
print(feature_names)

# Print the feature names
print(feature_names)
```

شکل ۶-۲ کدهای روش PCA قسمت ۲

۱. اولین قدم، استاندارد کردن داده‌ها است، یعنی مقیاس‌بندی ویژگی‌ها به گونه‌ای که میانگین و واریانس واحد صفر داشته باشند. این کار با استفاده از کلاس «StandardScaler» از ماژول «sklearn.preprocessing» انجام می‌شود. ما در کد نمونه‌ای از این کلاس را ایجاد می‌کنیم و روش "fit_transform" را روی داده‌های آموزشی ("X_resampled") و روش "transform" را روی داده‌های آزمایشی ("X_val") فراخوانی می‌کنیم. این کد تضمین می‌کند که مقیاس‌بندی در هر دو مجموعه داده سازگار است.

۲. مرحله بعدی اعمال PCA روی داده‌های استلندارد شده است که با استفاده از کلاس `PCA` از ماژول `sklearn.decomposition` انجام می‌شود. کد نمونه‌ای از این کلاس را ایجاد می‌کند و روی کل مؤلفه‌ها که ۳۰ تا بود اعمال می کند .

۳. سپس، روش «fit_transform» را روی داده‌های آموزشی («X_train_scaler») و روش «transform» را در داده‌های آزمون («X_test_scaler») فراخوانی می‌کند. این ویژگی‌های جدید ('X_train_PCA' و "X_test_PCA") را که ترکیبی خطی از ویژگی‌های اصلی هستند برمی‌گرداند.

۴. کد مقادیر "X_train_PCA" را چاپ می‌کند که داده‌های آموزشی تبدیل شده هستند. هر ردیف مربوط به یک مشاهده و هر ستون مربوط به یک جزء اصلی است.

۵. کد همچنین مقدار «prop_var» را چاپ می‌کند، که نسبت واریانس توضیح داده شده توسط هر جزء است. این یک ویژگی از «PCA» است که حاوی آرایه‌ای از نسبت‌های واریانس هر جزء به کل واریانس داده‌ها است. هر چه این نسبت بیشتر باشد، مولفه اطلاعات بیشتری می‌گیرد. همچنین کد «PC_number» تعداد مولفه ها را چاپ می‌کند، که آرایه‌ای از اعداد ۱ تا ۳۰ است.

۶. کد همچنین مقدار 'feature_names' را چاپ می‌کند، که لیستی از نام ویژگی‌های اصلی از داده‌های آموزشی ('X_resampled') است. این‌ها نام ستون‌ها در مجموعه داده اصلی، قبل از اعمال PCA است.

۷. سپس کد نسبت تجمعی واریانس توضیح داده شده توسط هر مؤلفه را محاسبه می‌کند که مجموع نسبت‌های همه مؤلفه‌ها تا آن نقطه است. این کار با استفاده از روش «cumsum» در آرایه «prop_var» انجام می‌شود که همراه با نام مؤلفه ها و نسبت‌های واریانس تک به تک مولفه ها در یک ماتریس 'z' چاپ می‌کند.

۸. در این مرحله از کد scree plot را ترسیم می‌کند، که نموداری از نسبت واریانس توضیح داده شده توسط هر مولفه در برابر شماره مؤلفه است. این کار با استفاده از تابع 'plt.plot' از ماژول 'matplotlib.pyplot' انجام می‌شود. کد همچنین یک عنوان، برچسب ها و یک شبکه به طرح اضافه می‌کند. طرح Scree به تجسم اینکه هر جزء چقدر به واریانس کمک می‌کند و تعداد بهینه مولفه‌ها را شناسایی می‌کند. یک معیار رایج این است که به دنبال نقطه بگردید، که منحنی شروع به صاف شدن می‌کند، که نشان می‌دهد افزودن مولفه های بیشتر چندان تاثیری روی دقت حاصل شده ندارد .

۹. سپس کد یک طرح خطی جایگزین را ترسیم می‌کند، که نمودار مشابه را نشان می‌دهد اما با خطوط افقی و عمودی که نقطه‌ای را نشان می‌دهد که نسبت تجمعی واریانس توضیح داده شده به ۰.۹۰ (یا ۹۰٪) با ۹ جزء

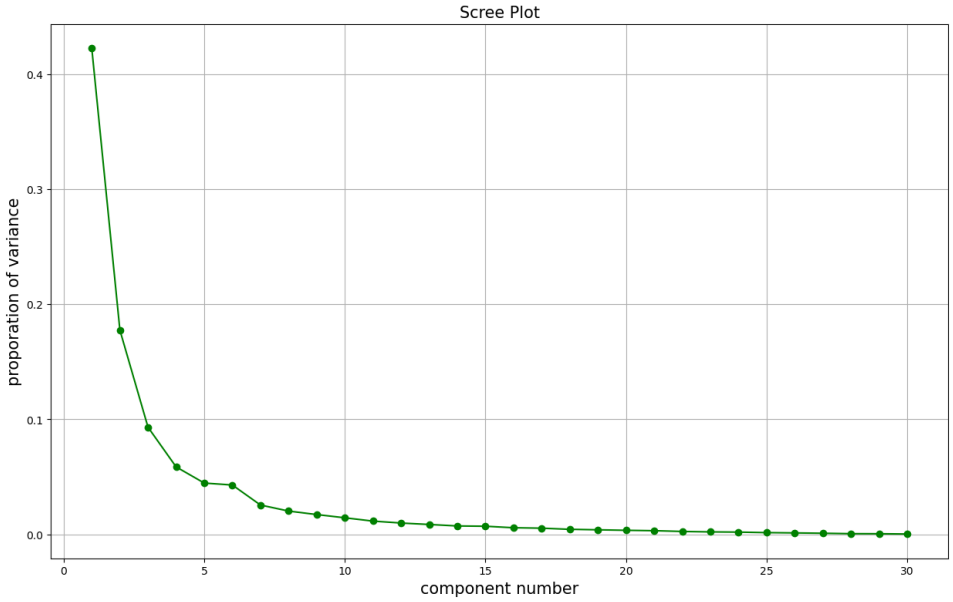
می‌رسد. کد با استفاده از توابع 'plt.axhline' و 'plt.axvline' خطوط را اضافه می‌کند و بر این اساس عنوان نمودار را تغییر می‌دهد.

۱۰. در نهایت، خروجی‌های کد اجرا شده به شرح «شکل ۶-۳ » می‌باشد.

	feature_names	variance ratio	Cumulative
0	radius1	0.42240346564734305	0.42240346564734305
1	texture1	0.17749014831118856	0.5998936139585316
2	perimeter1	0.0931270294327396	0.6930206433912711
3	area1	0.05866036074105881	0.75168100413233
4	smoothness1	0.04463332673220741	0.7963143308645374
5	compactness1	0.04291698560756818	0.8392313164721056
6	concavity1	0.02553611976236258	0.8647674362344682
7	concave_points1	0.02029674648847445	0.8850641827229426
8	symmetry1	0.01725840111253516	0.9023225838341962
9	fractal_dimension1	0.014440539707793314	0.9167631235419895
10	radius2	0.011575026973597352	0.9283381505155869
11	texture2	0.009889540367786514	0.9382276908833734
12	perimeter2	0.008637363009952064	0.9468650538933255
13	area2	0.007351587780089667	0.9542166416734152
14	smoothness2	0.007101840666636849	0.9613184823400521
15	compactness2	0.005804529060132049	0.9671230114001842
16	concavity2	0.005415374204559846	0.972538385604744
17	concave_points2	0.004429587266299663	0.9769679728710438
18	symmetry2	0.004001683931996742	0.9809696568030405
19	fractal_dimension2	0.003578366739871753	0.9845480235429123
20	radius3	0.0032416958627111835	0.9877897194056234
21	texture3	0.002561024624278845	0.9903507440299023
22	perimeter3	0.002193441956187545	0.9925441859860898
23	area3	0.0020118236778211727	0.994556009663911
24	smoothness3	0.0015459115700776273	0.9961019212339887
25	compactness3	0.001303716903390156	0.9974056381373788
26	concavity3	0.0010141318122045722	0.9984197699495834
27	concave_points3	0.0006420738599167645	0.9990618438095001
28	symmetry3	0.0005888886435777823	0.9996507324530779
29	fractal_dimension3	0.0003492675469224199 5	1.0000000000000002

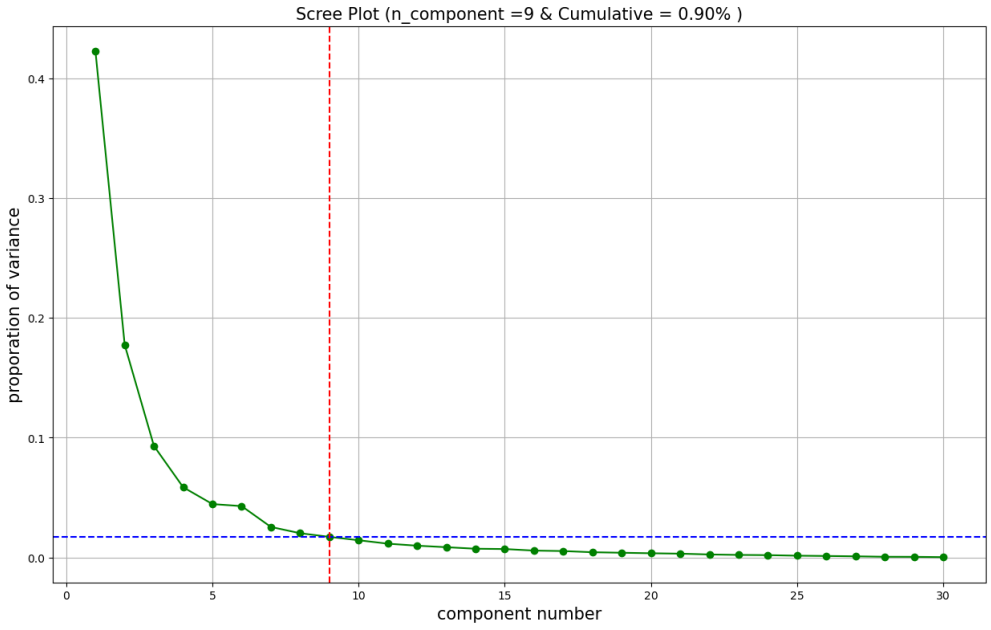
شکل ۶-۳ خروجی جدولی PCA بخش اول کد

نمودار Scree برای کد اجرا شده در مرحله قبل به شرح شکل ۶-۴ می‌باشد.



شکل ۶-۴ نمودار Scree نسبت واریانس‌ها به Component

همچنین در نمودار زیر که به شرح شکل۶-۵ می‌باشد،



شکل ۶-۵ تعداد ۹ component و Cumulative ٪۹۰

۶-۳ نتایج بدست آمده

با توجه به نتایج بدست آمده در شکل ۶-۳ و همچنین اعمال نسبت‌های محاسبه شده هر مولفه و ترسیم نتایج حاصل در نمودار scree plot که در شکل ۶-۴ آورده شده است می‌توان مشاهده نمود که از مولفه ی ۹ به بعد فاصله‌ی مقادیرنسبت‌های واریانس مولفه‌ها قابل چشم‌پوشی بوده (قابل ذکر است که انتخاب تعداد مولفه ها نسبت به خبره و نوع داده‌ها می‌توان متفاوت در نظر گرفت، چه بسا می‌توان تعداد مولفه بسنده کرد و یا اینکه نسبت به مقدار تجمعه نسبت واریانس‌ها تصمیم‌گیری کرد {در روش NN که در ادامه به ان خواهیم پرداخت با تغییر مقدار مولفه‌ها می توان دقت بدست آمده از آن تعداد مولفه را مشاهده نمود و تصمیم‌گیری کرد.})

۶-۴نمایش در فضای ۱-بعدی، ۲- بعدی، ۳- بعدی

ابتدا با توجه به کد نوشته شده در شکل ۶-۶ ، نتایج مرتبط با روش تحلیل مولفه اصلی در شکل ۶-۷ بدین صورت خواهد بود که تعداد اجزای اصلی (PC) را که می خواهید حفظ کنیم را مشخص می کنیم که در اینجا ۳ است:

```python
scaler = StandardScaler()
X_train_scaler = scaler.fit_transform(X_resampled)

n_components = 3

pca = PCA(n_components=n_components)
X_pca = pca.fit_transform(X_train_scaler)

selected_features = pd.DataFrame(data=X_pca, columns=[f'PC{i}' for i in range(1, n_components + 1)])

selected_data = pd.concat([selected_features, y_resampled], axis=1)

print("Explained Variance Ratios for Principal Components:")
explained_variances = pca.explained_variance_ratio_
for i, explained_variance in enumerate(explained_variances, 1):
    print(f"PC{i}: {explained_variance:.2%}")

cumulative_variance = pca.explained_variance_ratio_.cumsum()
print("Cumulative Explained Variance Ratio:")
for i, cumulative_explained_variance in enumerate(cumulative_variance, 1):
    print(f"PC{i}: {cumulative_explained_variance:.2%}")
```

شکل ۶-۶ کد روش PCA برای ۳ مولفه اصلی

```
Explained Variance Ratios for Principal Components:
PC1: 42.24%
PC2: 17.75%
PC3: 9.31%
Cumulative Explained Variance Ratio:
PC1: 42.24%
PC2: 59.99%
PC3: 69.30%
```

شکل ۶-۷ نتایج Cumulative و Variance Ratios برای ۳ مولفه اصلی

با در نظر گرفتن مؤلفه اصلی وتحلیل آن‌ها در فضای۱_بعدی، ۲_بعدی، ۳_بعدی. کد ۶-۹، ۶-۹ و ۶-۱۰ ؛ نتایج اشکال ۶-۱۱ ، ۶-۱۲ و ۶-۱۳ را می‌توان به صورت زیر ارائه داد:

PCA 1D

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))
plt.scatter(selected_data['PC1'], selected_data["Diagnosis"], marker='o')
plt.xlabel('Principal Component 1')
plt.ylabel('Target Variable')
plt.title('Data in 1D Space')
plt.grid(True)
plt.show()
```

شکل ۶-۸ کد PCA-PC1

PCA 2D

```python
from mpl_toolkits.mplot3d import Axes3D

plt.figure(figsize=(8, 6))
plt.scatter(selected_data['PC1'], selected_data['PC2'],
            c=selected_data["Diagnosis"], cmap='cividis', marker='o')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Data in 2D Space')
plt.grid(True)
plt.show()
```

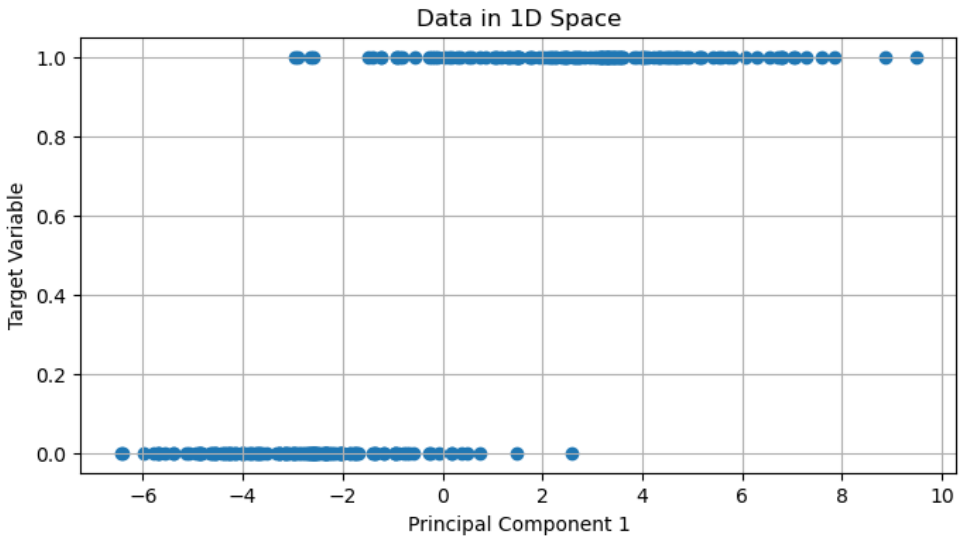
شکل ۶-۹ کد PCA-PC2

PCA 3D

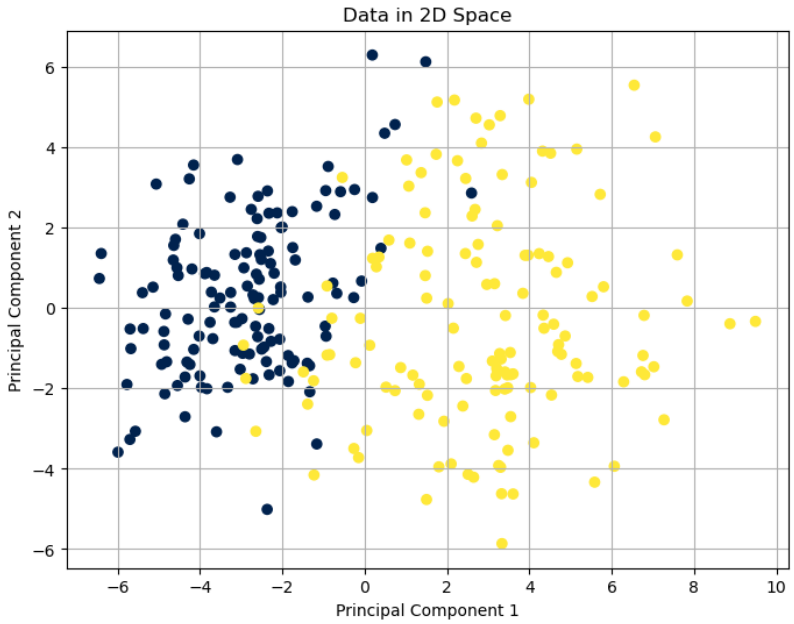
```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(selected_data['PC1'], selected_data['PC2'], selected_data['PC3'],
                c=selected_data["Diagnosis"], cmap='cividis', marker='o')
ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
plt.title('Data in 3D Space')
fig.colorbar(sc)
plt.show()
```

شکل ۶-۱۰ کد PCA-PC3

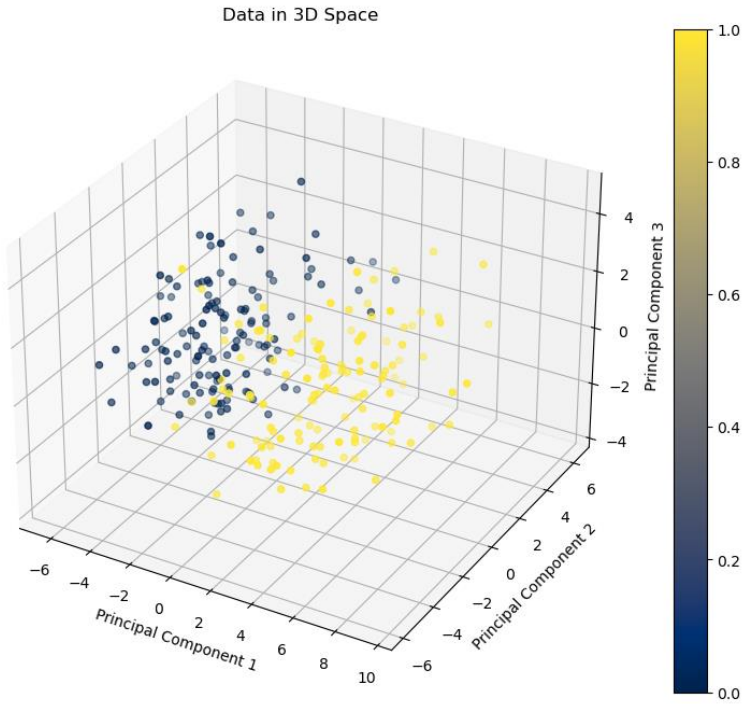
۶-۵ نتایج به دست آمده



شکل۶-۱۱ نمودار داده ها در فضای یک بعدی



شکل ۶-۱۲ نمودار داده ها در فضای دو بعدی



شکل ۶-۱۳ نمودار داده ها در فضای سه بعدی

در واقع در این سه حالت هر بار یک مولفه را به عنوان PC و یا جز منتخب برگزیدیم و هر بار در ترسیم محورهای مختصات طوری رسم کردیم که عمود بر محور انتخاب شده قبلی باشد در فضای ۱ بعدی که تنها یک مولفه در نظر گرفتیم و تمامی مولفه های دیگر را نسبت به مولفه ی انتخاب شده در شکل ۶-۱۱ نمایش دادیم در فضای ۲ بعدی نیز بعد از انتخاب PC1 برای انتخاب PC2 حالت عمود بودن بر محور مولفه منتخب را با توجه به کدها در نظر گرفتیم ودر شـــکل ۶-۱۲ نمایش دادیم در آخر در فضـــای ۳ بعدی نیز بعد از انتخاب PC1 برای انتخاب PC2 بی نهایت حللت برای عمود بودن بر محور مولفه منتخب داشـــتیم که بعد از انتخاب شـــدن PC2 مجدد برای انتخاب PC3 تنها یک حالت و عمود بر محور های دو مولفه منتخب قبلی عمل کردیم که در شکل ۶-۱۳ نمایش داده شده است .

۶-۶ تاثیر مولفه های اصلی در عملکرد روش دسته بندی (NN) و مقایسه با روش های دیگر:

کدهای بعدی در ادامه آورده شده است، نتایج هر یک را نیز مشاهده می‌کنید. در اینجا به توضیح آن‌ها می‌پردازیم:

NN Classifier For comparing PCA

```
pcaNN = PCA(n_components = 11)
X_train_PCA = pcaNN.fit_transform(X_train_scaler)
X_test_PCA = pcaNN.transform(X_test_scaler)
accuracy_evaluation(X_train_PCA,y_resampled , X_test_PCA, y_val)
```

شکل ۶-۱۴ کد تاثیر روی عملکرد روش دسته بندی NN

```
prop_varNN = pcaNN.explained_variance_ratio_
PC_numberNN = np.arange(pcaNN.n_components_) + 1

plt.figure(figsize=(15,9))
plt.plot(PC_number,
         prop_var,
         'go-')
plt.title('Scree Plot (n_component =11 & Cumulative = 93.85% )',
           fontsize = 15)
plt.xlabel('component number',
            fontsize = 15)
plt.ylabel('proporation of variance',
                  fontsize = 15)
plt.axhline(y= 0.0115,
            color = 'blue',
            linestyle = '--')
plt.axvline(x=11,
            color='red',
            linestyle = '--')
plt.grid()

plt.figure(figsize=(15,9))
plt.plot(PC_numberNN,
         prop_varNN,
         'go-')
plt.title('Scree Plot (n_component =11)',
           fontsize = 15)
plt.xlabel('component number',
            fontsize = 15)
plt.ylabel('proporation of variance',
                  fontsize = 15)
plt.grid()
plt.show()

```

شکل ۶-۱۵ کد ترسیم Scree برای تاثیر روی عملکرد روش دسته بندی NN

ابتدا یک شـــی PCA با ۱۱ جزء ایجاد می‌کند، به این معنی که ابعاد داده‌ها را به ۱۱ ویژگی کاهش می‌دهد. تبدیل PCA را به داده‌های آموزشی اعمال می‌کند که توسط برخی از شی مقیاس کننده مقیاس شده است. این ماتریس شکل جدیدی را برمی‌گرداند (تعداد مولفه، ۱۱) که شـــامل اجزای اصـــلی داده‌های آموزشـــی اســـت و همچنین همان تبدیل PCA را به داده‌های آزمایشی اعمال می‌کند که توسط همان شی مقیاس‌کننده نیز مقیاس‌بندی می‌شود. این یک ماتریس جدید از شکل (تعداد مولفه، ۱۱) را برمی‌گرداند که شامل اجزای اصلی داده‌های آزمایش است.

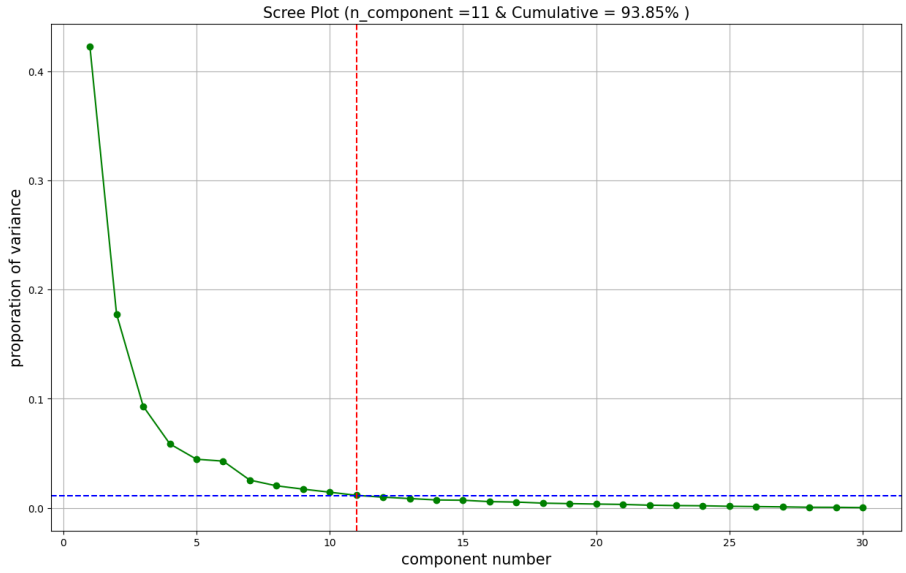
در نهایت با استفاده از تابعی که در قسمت های قبل برای روش دسته بندی و بدست آوردن نتایج دقت در این روش است را با داده های ورودی مقیاس شده در PCA، خروجی میگیریم.

```
Accuracy is:0.9385964912280702
```

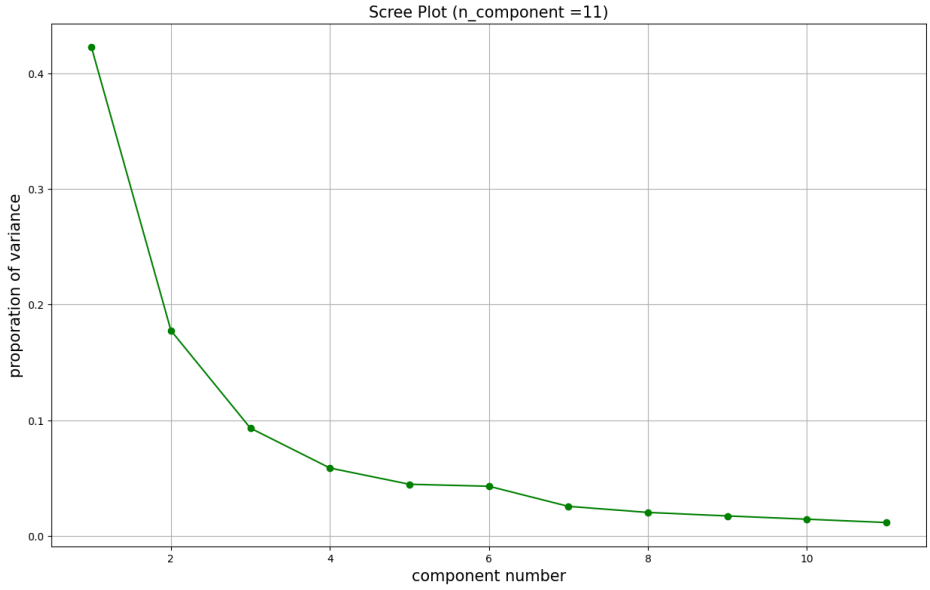
شکل ۶-۱۶ میزان دقت روش دسته بندی NN در تعداد ۱۱ مولفه اصلی

همانطور که قبلا در مورد نحوه در نظر گرفتن تعداد مولفه‌ها توضیح داده شده بود در اینجا نیز با در نظر گرفتن ۱۱ مولفه به عنوان مولفه‌های منتخب و محاسبه دقت حاصل از این تعداد مولفه‌ها با روش NN مشاهده کردیم که دقت بدست آمده

برابر با ۹۳.۸۵٪ حاصل شد که در شکل ۶-۱۷ و ۶-۱۸ مجدد scree plot حاصل از ۱۱ مولفه منتخب نمایش داده شده است.



شکل ۶-۱۷ تعداد ۱۱ component و Cumulative ٪۹۳.۸۵



شکل ۶-۱۸ Scree برای ۱۱ Component

حال با مقایســـه دقت به دســـت آمده بر مبنای معیار ارزیابی NN در همه روش های مربوط به کاهش ویژگی متوجه می شویم که روش تحلیل مولفه های اصلی بهتر است.(نتایج در جدول ۶-۱)

جدول۶-۱ مقایسه دقت بدست آمده در روش های مختلف

Method	Accuracy
Variance	0.921
Forward	0.85
Backward	0.89
forward backward	0.85
Genetic Algorithm	0.934
PCA	0.938