گزارش دیتاست اول تمرین سری دوم یادگیری ماشین

عرفان کرمی – ۹۸۲۲۲۰۷۹

در این بخش از تمارین، تمرکز اصلی بر روی استخراج ویژگی، انتخاب ویژگی و اعمال تبدیل های مختلف بر روی داده ها است.

*** برای سوالات ۱ تا ۴ که به هم مربوط اند، فرایند مدیریت داده های گمشده، حذف داده های پرت، تغییر مقیاس فیچر ها و ... به منظور مراحل پیش پردازش استفاده شده اند.

اطلاعات مربوط به دیتاست در بخش زیر آورده شده است:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

۱. در سوال اول صرفا پیاده سازی $forward\ selection$ خواسته شده است.

۲. الگوریتم پیاده سازی شده را بر روی داده ها اعمال میکنیم و در نهایت فیچر های خروجی توسط $forward\ selectin\ algorithm$ عبارتند از:

```
['ram',
 'mobile_wt',
 'int_memory',
 'blue',
 'clock_speed',
 'dual_sim',
 'fc',
 'four_g',
 'm_dep',
 'n_cores']
```

حال اگر الگوریتم رگرسیون لجیستیک با $random\_state = 42, \max\_iter = 3000$ را بر روی فیچر های انتخاب شده اجرا میکنیم و نتایج زیر به دست آمده اند:

Performance on train data:

```
precision for train data:  0.780952380952381

recall for train data:  0.8793565683646113

f1-score for train data:  0.8272383354350568
```

Performance on test data:

```
precision for test data:  0.7663230240549829

recall for test data:  0.8955823293172691

f1-score for test data:  0.8259259259259261
```

trained coefficients:

```
array([[ 8.00611646, -4.24560378, -1.01709698, -0.0151324 , -0.04777429
,
        -0.01560425, -0.11398382, -0.01536768, -0.01684771, -0.12891172
]])
```

مقادیر به دست آمده نشان میدهد که کم برازش مدل بر روی داده ها اتفاق افتاده است.که این به این معناست که مدل های پیچیده تر یا استفاده از فیچر های مرتبه بالاتر مورد نیاز است.

۳. در این بخش خواسته شده است که با استفاده از الگوریتم PCA همان تعداد فیچر های مرحله قبل را استخراج کنیم که این عمل در نوتبوک انجام شده است.

۴. در این سوال خواسته شده است که الگوریتم رگرسیون لجیستیک را بر روی داده های به دست آمده ازسوال قبل را اجرا و خروجی را گزارش کنیم.

نتیجه خروجی الگوریتم رگرسیون لجیستیک بر روی داده های به دست آمده از الگوریتم PCA به صورت زیر است:

Performance on train data:

```
precision for train data:  0.7801932367149759
recall for train data:  0.8659517426273459
f1-score for train data:  0.8208386277001272
```

Performance on test data:

```
precision for test data:  0.7829181494661922
recall for test data:  0.8835341365461847
f1-score for test data:  0.830188679245283
```

مقادیر به دست آمده مجددا نشان میدهد که کم برازش مدل بر روی داده ها اتفاق افتاده است.که این به این معناست که مدل های پیچیده تر یا استفاده از فیچر های مرتبه بالاتر مورد نیاز است.

۵. در این بخش سوالی پرسیده نشده است ☺

۶. در این بخش خواسته شده است که چند متد مختلف که در مهندسی ویژگی تعریف شده است بر روی داده ها اعمال شود. قبل از اعمال این روش ها یکبار الگوریتم رگرسیون لجیستیک بر روی داده های اصلی اعمال شده است تا در صورت لزوم مقایسه صورت بگیرد.

نتایج این مدل بر روی داده های ترین و تست به شرح زیر است:

نتیجه بر روی داده ترین:

	precision	recall	f1-score	support
0	0.88	0.85	0.86	375
1	0.68	0.68	0.68	375
2	0.60	0.56	0.58	375
3	0.75	0.81	0.78	375
accuracy			0.73	1500
macro avg	0.73	0.73	0.73	1500
weighted avg	0.73	0.73	0.73	1500

نتیجه بر روی داده تست:

	precision	recall	f1-score	support
0	0.94	0.84	0.89	129
1	0.61	0.69	0.65	121
2	0.53	0.48	0.50	133
3	0.69	0.75	0.71	114
accuracy			0.69	497
macro avg	0.69	0.69	0.69	497
weighted avg	0.69	0.69	0.69	497

که مجددا نشان دهنده کم برازش مدل بر روی داده هاست.حدس میزنیم دلایلی که باعث شده است که دقت مدل بر روی همه فیچر ها از حالت انتخاب زیر مجموعه ای از فیچر ها در مراحل قبل کم تر باشد،تعداد کمتر تکرار الگوریتم بهینه ساز و نیز عدم اعمال تغییر مقیاس بر روی داده هاست ه در مراحل قبل هر دو عملیات انجام گرفت.

a) – آ: در این مرحله سه نوع $binning$ با استفاده از کلاس $KBinsDiscritizer$ بر روی فیچر $battery\_power$ انجام گرفته است. $binning$ عبارتست از عملیاتی که در آن مقادیر عددی پیوسته به مقدیری گسسته تبدیل میشوند.نتایج به شرح زیر میباشند:

- استفاده از استراتژی $quantile$ و تعداد بین برابر ۴: در این روش مرز بین ها به گونه ای انتخاب میشوند که بین ها تعداد یکسانی از داده ها را در بر داشته باشند. مرز های حاصل برای بین ها به صورت زیر هستند:

```
[ 501.  ,  851.75, 1226.  , 1615.25, 1998.  ]
```

- استفاده از استراتژی $uniform$ و تعداد بین برابر با ۱۰: در این روش مرز بین بین ها به گونه ای انتخاب میشود که بازه بین ها با هم یکسان باشد. بدیهی است که در این روش تعداد داده های موجود در بین ها با هم برابر نیست.مرز های حاصل برای بین ها به صورت زیر هستند:

```
[ 501. ,  650.7,  800.4,  950.1, 1099.8, 1249.5, 1399.2, 1548.9,
       1698.6, 1848.3, 1998. ]
```

- استفاده از استراتژی $kmeans$ و تعداد بین برابر ۶: در این استراتژی بین ها به گونه ای انتخاب میشوند که فاصله نقاط در هر بین تا مرکز ثقل داده های آن بین کمترین مقدار ممکن باشد. مرز های حاصل از این روش عبارتند از:

```
[ 501.        ,  746.17836176,  995.49664652, 1247.23007589,
      1499.04597994, 1751.12441469, 1998.        ]
```

b) در این بخش خواسته شده است که تبدیل $one\ hot\ encoding$ بر روی فیچر های کتگوریکال دیتاست اصلی اعمال شود. فیچر های کتگوریکال دیتاست عبارتند از:

['blue', 'dual_sim', 'four_g','three_g','touch_screen', 'wifi']

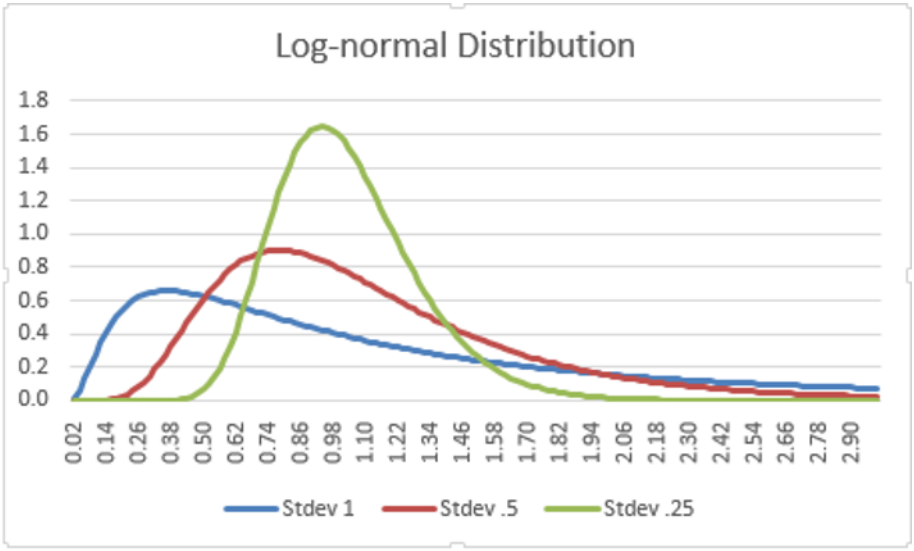
با توجه به اینکه تمامی مقادیر موجود در این فیچر ها تنها دو مقدار دارند پس عملا اتفاقی که خواهد افتاد $binary\ one\ hot\ encoding$ خواهد بود.

ما به طور کلی برای تبدیل فیچر های کتگوریکال به فیچر های عددی دو رویکرد اصلی را پیش روی خود خواهیم داشت:

- $ordinal\ encoding$: در این حالت به هر کدام از کتگوری ها یک عدد نسبت داده میشود و این عدد با این کتگوری در ستون جدید جایگذاری میشود. در واقع در این روش خروجی برای هر فیچر کتگوریکال یک ستون خواهد بود. اشکال اصلی این روش ان است که اگر تعداد کتگوری های یک فیچر n باشد در نهایت اعداد $\{0, \dots, n-1\}$ به کتگوری ها نسبت داده میشود. اما ممکن است دو عدد که به هم نزدیک باشند به کتگوری هایی نسبت داده شوند که اصلا به هم شبیه نیستند یا دو عدد که از هم دور باشند به کتگوری هایی نسبت داده شوند که کاملا به هم شبیه باشند، در نتیجه عملا ممکن است کشف روند ها و الگو ها برای الگوریتم یادگیری سخت و حتی ناممکن شود.
- $one\ hot\ encoding$: در این روش برای هر کتگوری هر فیچر یک ستون در نظر گرفته میشود که اگر داده به آن کتگوری تعلق داشته باشد در آن ستون دارای مقدار ۱ و در ستون بقیه کتگوری ها مقدار ۰ خواهد داشت. در این روش مشکلی که در روش قبل به وجود داشت پیش نمی آید بنابراین معمولا تمایل داریم که از این روش استفاده کنیم. البته دقت داریم که اگر تعداد فیچر های کتگوریکال و همچنین مقادیر آنها افزایش یابد تعداد ستون های حاصل افزایش می یابد که ممکن است آموزش مدل را سخت تر کند.

در حالت هایی که عملا استفاده از $one\ hot\ encoding$ ناممکن میشود یک راه حل میتواند استفاده از $ordinal\ encoding$ باشد با این رویکرد که ابتدا همبستگی داده های هر کتگوری با تارگت محاسبه شود و سپس کتگوری ها بر حسب این مقادیر به صورت صعودی مرتب شوند و اعداد از کم به زیاد به آنها نسبت داده شود. اینگونه میتوان به مشکلی که ذکر شد غلبه کرد.

C) به صورت کلی بهره گیری از تبدیلاتی مانند تبدیلات لگاریتمی یا نمایی بر روی داده ها برای تغییر دادن توزیع داده ها خواهد بود . مثلا اگر داده های ما از یک توزیع لاگ-نرمال باشند و تبدیل لگاریتمی را بر روی آنها اعمال کنیم توزیع داده ها تقریبا به نرمال تغییر خواهد کرد. این باعث خواهد شد چولگی چپ از بین برود و این نکته مهمی است چرا که ما مطمئن میشویم مشکل دم بلند یا سوگیری مدل به بخشی از داده ها کاهش خواهد یافت. از طرفی میدانیم استفاده از توزیع نرمال از این جهت اهمیت دارد که بسیاری از آزمون های آماری برای این توزیع تعریف شده اند و از طرفی هم بعضی مدل های یادگیری ماشین پیش فرضشان این است که داده ها از توزیع نرمال پیروی میکنند یا با استفاده از این فرض حجم محاسبات آنها کاهش می یابد.

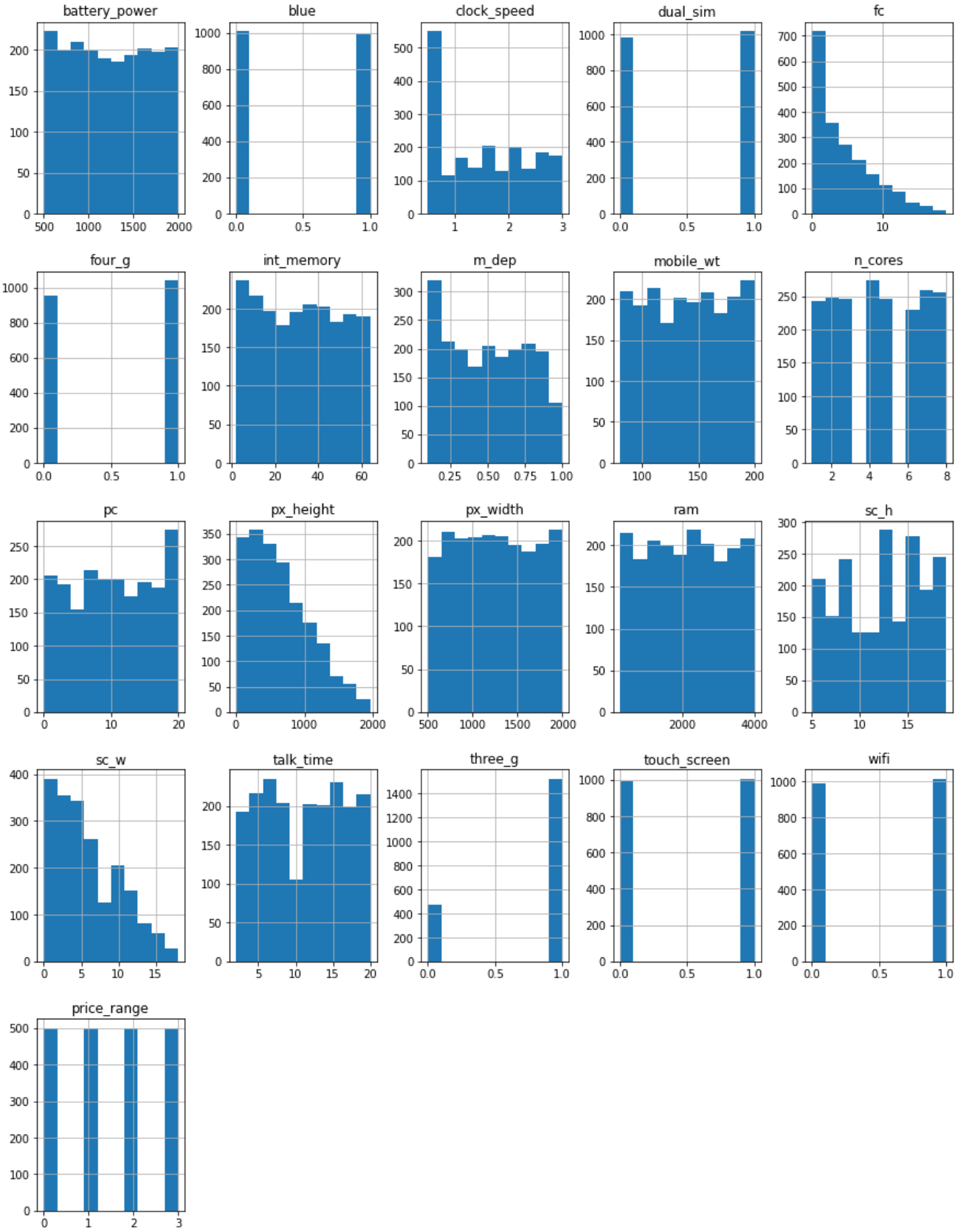


ما دو تبدیل لگاریتمی و نمایی را بر روی داده های غیر کتگوریکال اعمال کرده ایم.

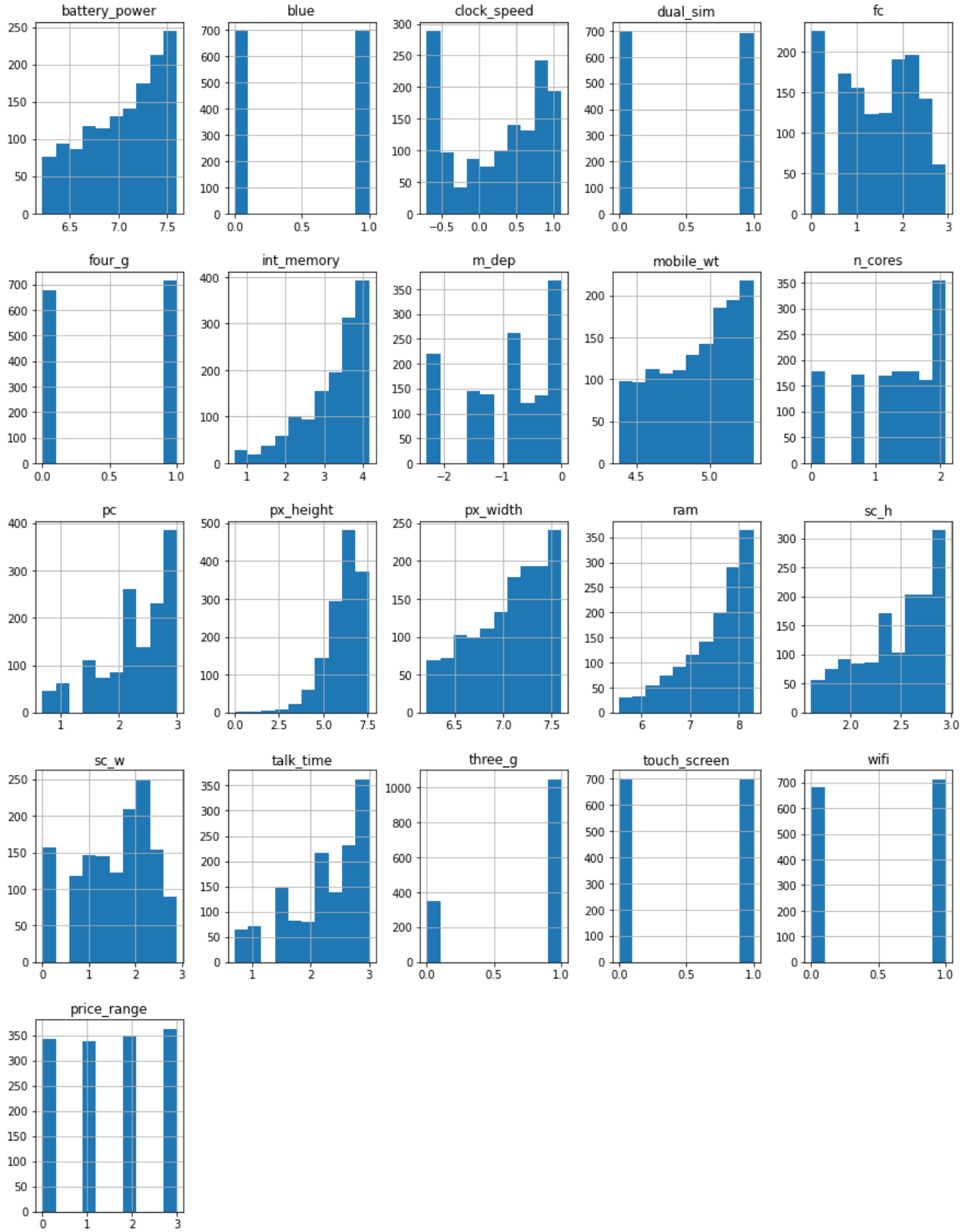
- تبدیل لگاریتمی:

اگر ابتدا به نمودار هیستوگرام داده ها نگاه کنیم متوجه میشویم که توزیع داده اکثرا یکنواخت است در نتیجه پیشبینی میکنیم که تبدیل لگاریتمی در این مسئله مفید نباشد. البت نکته ای که باید به آن دقت کنیم این است که پس از اعمال این تبدیل تعداد زیادی از داده ها عملا برابر با بینهایت شدند در نتیجه حجم عظیمی از داده ها در اثر این تبدیل از بین رفت که این مشکلی بزرگ در اعمال این تبدیل است.

هیستوگرام داده ها قبل از اعمال تبدیل:

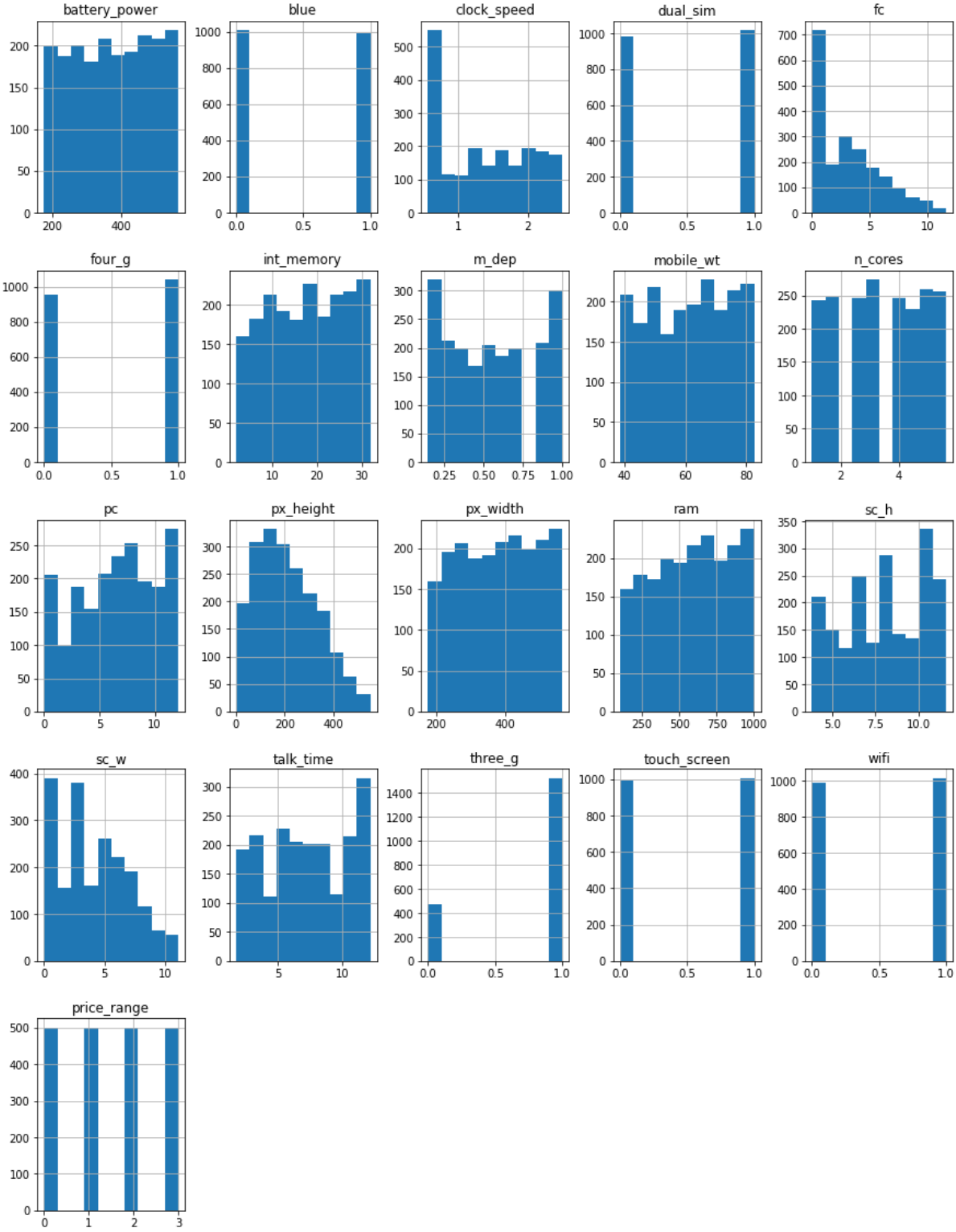


نمودار هیستوگرام داده ها پس از اعمال تبدیل:



همانطور که حدس میزدیم توزیع داده با به توزیع نرمال تبدیل نشد.

- تبدیل نمایی: در این تبدیل فیچر ها به یک توان دلخواه مثبت میرسند. این تبدیل را بر روی داده ها انجام دادیم و هیستگرام داده های حاصل به صورت زیر است:



شکل نشان میدهد که چولگی و انحراف به یک سمت در این تبدیل کمتر از تبدیل قبلی بوده است.

d) در این بخش فیچر جدیدی که همان مساحت است با نام $mobile\_area$ به داده ها اضافه شده است که میتوان رابطه تولید آنرا به صورت زیر نوشت:

$$mobile\_area = sc\_h * sc\_w$$

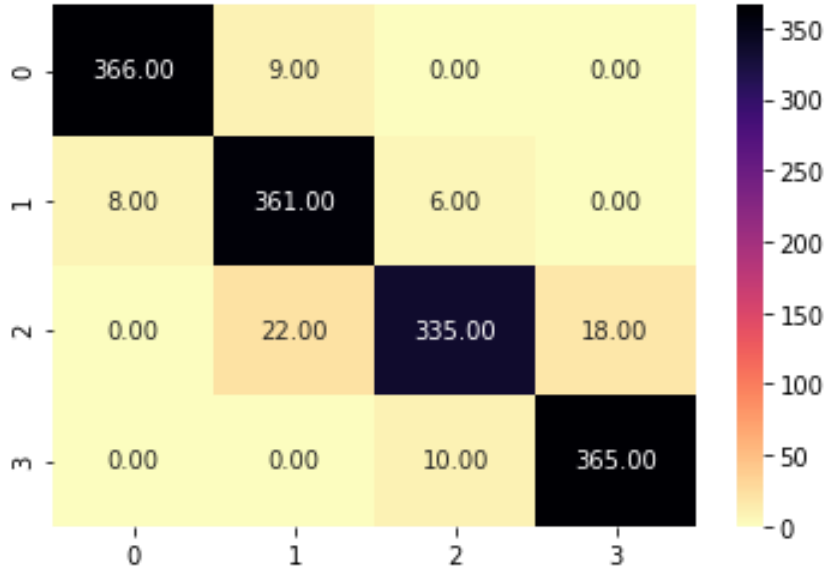
۷. در این بخش خواسته شده است که الگوریتم ماشین بردار پشتیبان بر روی داده های حاصل از تک تک مراحل قبل اعمال و نتیجه گزارش شود:

- نتیجه بر روی دیتاست اصلی:

نتیجه بر روی داده ترین

:Result on train data

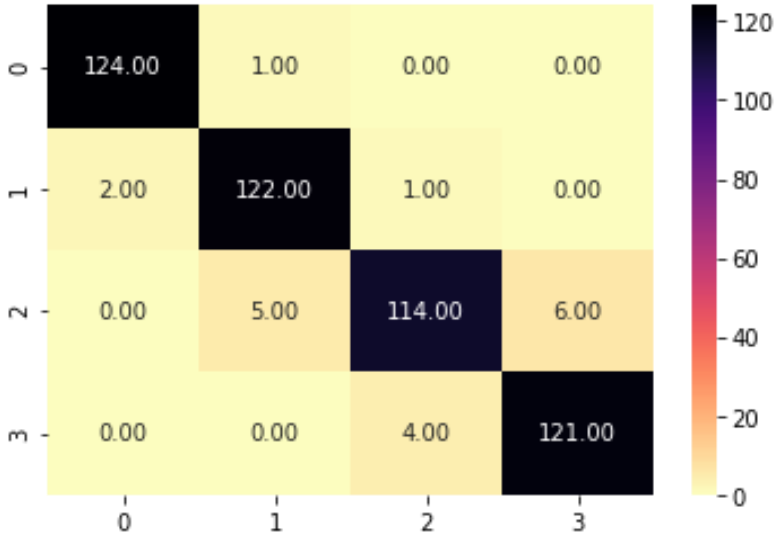
	precision	recall	f1-score	support
0	0.98	0.98	0.98	375
1	0.92	0.96	0.94	375
2	0.95	0.89	0.92	375
3	0.95	0.97	0.96	375
accuracy			0.95	1500
macro avg	0.95	0.95	0.95	1500
weighted avg	0.95	0.95	0.95	1500



نتیجه مدل بر روی داده تست:

:Result on test dataset

	precision	recall	f1-score	support
0	0.98	0.99	0.99	125
1	0.95	0.98	0.96	125
2	0.96	0.91	0.93	125
3	0.95	0.97	0.96	125
accuracy			0.96	500
macro avg	0.96	0.96	0.96	500
weighted avg	0.96	0.96	0.96	500

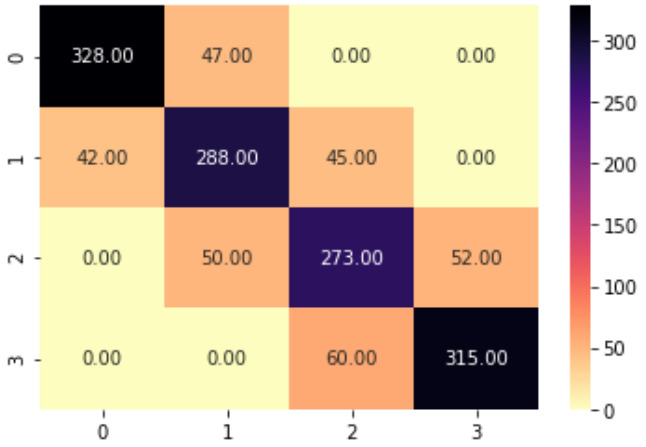


نتایج نشان میدهند که مدل بر روی دیتاست اولیه به خوبی فیت شده است.

- نتایج بر روی داده های حاصل از $binning$:

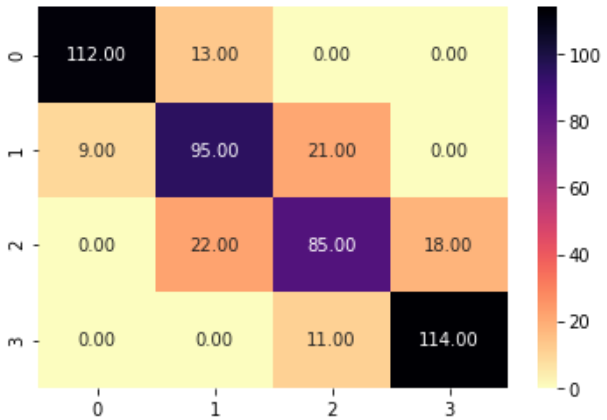
Result on train dataset:

	precision	recall	f1-score	support
0	0.89	0.87	0.88	375
1	0.75	0.77	0.76	375
2	0.72	0.73	0.73	375
3	0.86	0.84	0.85	375
accuracy			0.80	1500
macro avg	0.80	0.80	0.80	1500
weighted avg	0.80	0.80	0.80	1500



Result on test dataset:

	precision	recall	f1-score	support
0	0.93	0.90	0.91	125
1	0.73	0.76	0.75	125
2	0.73	0.68	0.70	125
3	0.86	0.91	0.89	125
accuracy			0.81	500
macro avg	0.81	0.81	0.81	500
weighted avg	0.81	0.81	0.81	500

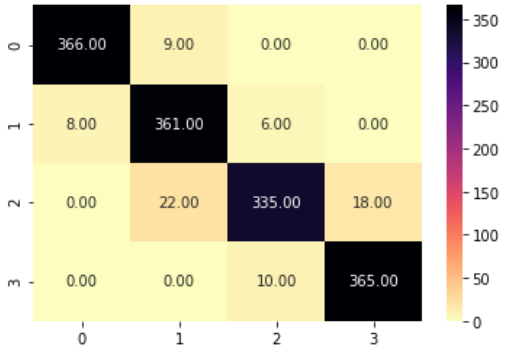


که نتایج به وضوح از افت عملکرد مدل خبر میدهند.

- نتایج بر روی داده های حاصل از $one\ hot\ encoding$:

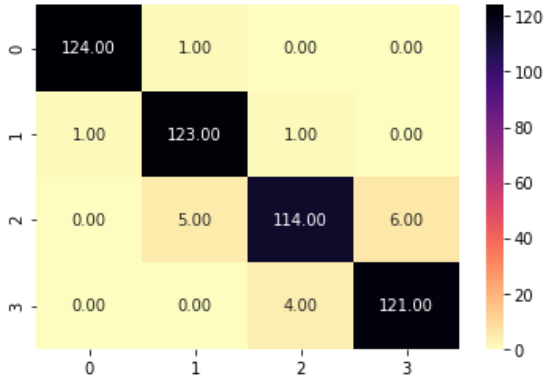
Result on train dataset:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	375
1	0.92	0.96	0.94	375
2	0.95	0.89	0.92	375
3	0.95	0.97	0.96	375
accuracy			0.95	1500
macro avg	0.95	0.95	0.95	1500
weighted avg	0.95	0.95	0.95	1500



Result on test dataset:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	125
1	0.95	0.98	0.97	125
2	0.96	0.91	0.93	125
3	0.95	0.97	0.96	125
accuracy			0.96	500
macro avg	0.96	0.96	0.96	500
weighted avg	0.96	0.96	0.96	500

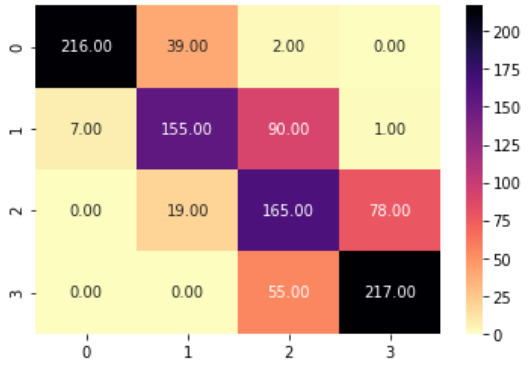


که نشان میدهد که عملکرد مدل بر روی داده های حاصل از این تبدیل قابل قبول بوده است هرچند پیشرفتی نسبت به داده های اصلی حاصل نشده است.

- نتایج بر روی داده های حاصل ازاعمال تبدیل لگاریتمی:

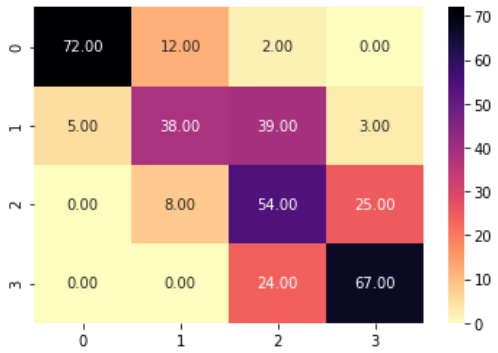
:Result on train dataset

	precision	recall	f1-score	support
0	0.97	0.84	0.90	257
1	0.73	0.61	0.67	253
2	0.53	0.63	0.57	262
3	0.73	0.80	0.76	272
accuracy			0.72	1044
macro avg	0.74	0.72	0.73	1044
weighted avg	0.74	0.72	0.73	1044



:Result on test dataset

	precision	recall	f1-score	support
0	0.94	0.84	0.88	86
1	0.66	0.45	0.53	85
2	0.45	0.62	0.52	87
3	0.71	0.74	0.72	91
accuracy			0.66	349
macro avg	0.69	0.66	0.66	349
weighted avg	0.69	0.66	0.67	349

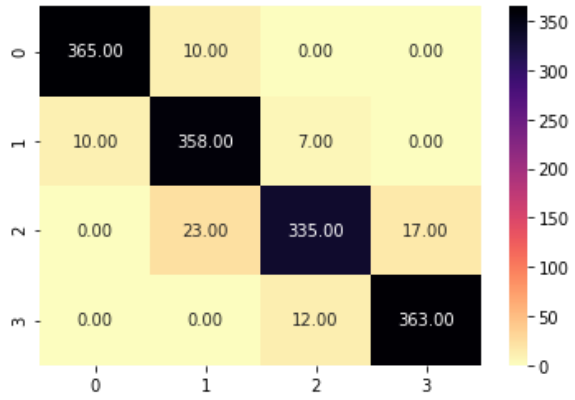


همانطور که حدس می زدیم تبدیل لگاریتمی بر روی داده هایی که دارای توزیع تقریبا یکنواخت بوده اند نه تنها موجب بهبود عملکرد نشده است بلکه موجب افت عملکرد مدل شده است.

- نتایج بر روی داده های حاصل ازاعمال تبدیل لگاریتمی:

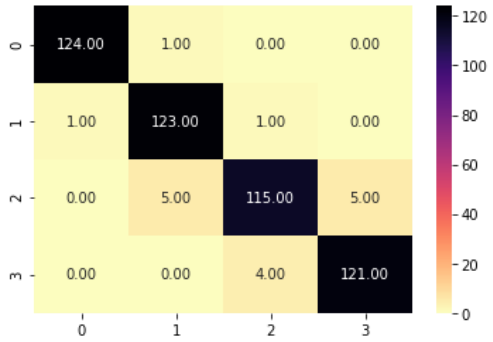
:Result on train dataset

	precision	recall	f1-score	support
0	0.97	0.97	0.97	375
1	0.92	0.95	0.93	375
2	0.95	0.89	0.92	375
3	0.96	0.97	0.96	375
accuracy			0.95	1500
macro avg	0.95	0.95	0.95	1500
weighted avg	0.95	0.95	0.95	1500



:Result on test dataset

	precision	recall	f1-score	support
0	0.99	0.99	0.99	125
1	0.95	0.98	0.97	125
2	0.96	0.92	0.94	125
3	0.96	0.97	0.96	125
accuracy			0.97	500
macro avg	0.97	0.97	0.97	500
weighted avg	0.97	0.97	0.97	500

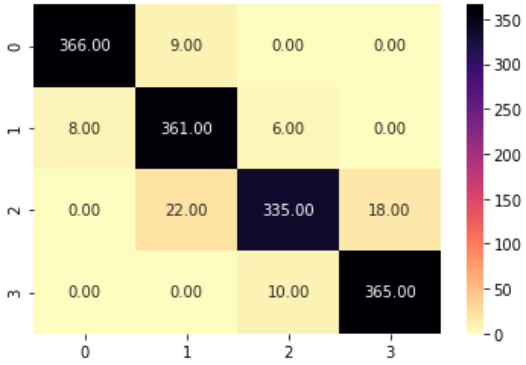


مجددا همانطور که حدس میزدیم این تبدیل نسبت به تبدیل قبل بهتر بوده و توانسته کمی عملکرد با بهبود ببخشد

- نتایج بر روی داده های حاصل ازایجاد فیچر جدید مساحت:

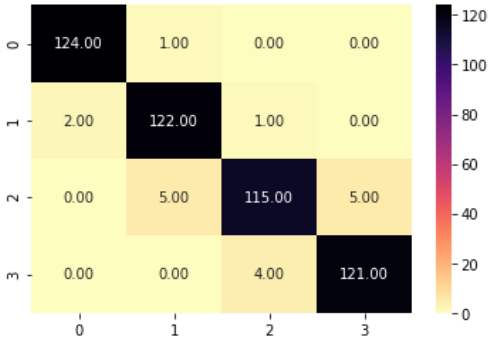
:Result on train dataset

	precision	recall	f1-score	support
0	0.98	0.98	0.98	375
1	0.92	0.96	0.94	375
2	0.95	0.89	0.92	375
3	0.95	0.97	0.96	375
accuracy			0.95	1500
macro avg	0.95	0.95	0.95	1500
weighted avg	0.95	0.95	0.95	1500



:Result on test dataset

	precision	recall	f1-score	support
0	0.98	0.98	0.98	375
1	0.92	0.96	0.94	375
2	0.95	0.89	0.92	375
3	0.95	0.97	0.96	375
accuracy			0.95	1500
macro avg	0.95	0.95	0.95	1500
weighted avg	0.95	0.95	0.95	1500

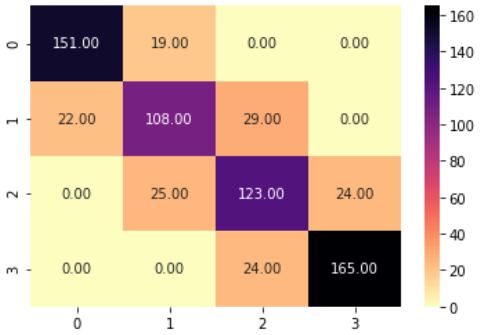


که نتایج نشان میدهند که این تبدیل تاثیر خاصی بر روی نتیجه و عملکرد مدل نداشته است.

- نتایج بر روی داده های حاصل از تبدیلات همزمان قبل:

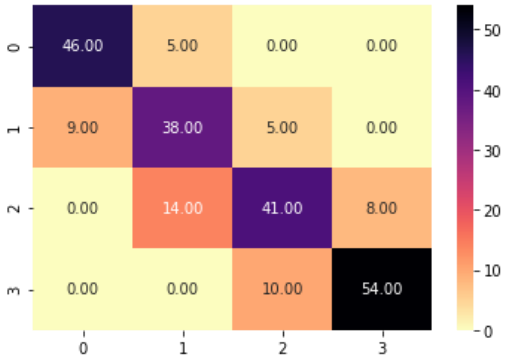
:Result on train dataset

	precision	recall	f1-score	support
0	0.87	0.89	0.88	170
1	0.71	0.68	0.69	159
2	0.70	0.72	0.71	172
3	0.87	0.87	0.87	189
accuracy			0.79	690
macro avg	0.79	0.79	0.79	690
weighted avg	0.79	0.79	0.79	690



:Result on test dataset

	precision	recall	f1-score	support
0	0.84	0.90	0.87	51
1	0.67	0.73	0.70	52
2	0.73	0.65	0.69	63
3	0.87	0.84	0.86	64
accuracy			0.78	230
macro avg	0.78	0.78	0.78	230
weighted avg	0.78	0.78	0.78	230



که افت عملکرد مدل به خوبی مشهود است.

۸. $bootstrapping$ یک روش نمونه گیری از مجموعه ای داده است به منظور تخمین پارامتر هایی از قبیل MSE تابع تخمین داده شده و تابع اصلی. در این روش برخلاف $cross\ validation$ که داده ها به K بخش که اشتراکی با هم ندارند تقسیم میشوند و سپس در K مرحله هر بار یک بخش به عنوان داده تست و بقیه به عنوان داده ترین استفاده میشوند در

این روش K دیتاست ساخته میشود که هر دیتاست حاصل نمونه گیری تصادفی با جایگذاری از داده های اصلی است. نشان داده اند که در این روش هر دیتاست حاوی حدود ۶۶ درصد از داده های دیتاست اصلی خواهد بود . حال میتوان مدل را بر روی این دیتاست ها ترین کرد و در نهایت پارامتر های مورد نظر را تخمین زد.

اصلی ترین کاربرد این متد در جایی است که دیتاست اولیه ما تعداد داده کمی داشته باشد و یا استفاده از آزمون های آماری دیگر برای تخمین امکان پذیر نباشد.

۹. خالی

۱۰. بله در شکل مورد نظر میتوان با استفاده از روش با استفاده از روش $elbow$ بهترین مرتبه مدل را یافت اما باید دقت داشته باشیم که این متد همواره کاربردی نیست چون همواره در این متد اولین جایی که خطا کاهش محسوس داشته باشد و سپس نرخ کاهش خطا کم باشد آن نقطه را به عنوان بهترین نقطه خروجی میدهد اما فرض کنید که نمودار به گونه ای باشد که چندین آرنج داشته باشد که اتفاقا آخرین آرنج بیشترین کاهش در میزان خطا را داشته باشد. با توجه با اینکه این روش اولین آرنج را خروجی میدهد و در ابتدای نمودار مقدار بایاس زیاد است و خطا هم با مربع بایاس متناسب است پس در واقع مدل خروجی در این حالت اصلا مرتبه مناسبی نخواهد داشت.

بخش امتیازی:

در بخش امتیازی ما الگوریتم $backward\ selection$ را پیاده سازی کردیم که ۱۰ فیچر خروجی توسط این الگوریتم در زیر آوده شده اند:

```
['int_memory',
 'mobile_wt',
 'pc',
 'ram',
 'sc_h',
 'sc_w',
 'talk_time',
 'three_g',
 'touch_screen',
 'wifi']
```

و عملکرد مدل رگرسیون لجیستیک بر روی این زیر مجموعه از فیچر ها نشان داده شده است:

```
precision for train data:  0.780952380952381
recall for train data:  0.8793565683646113
f1-score for train data:  0.8272383354350568


precision for test data:  0.7663230240549829
recall for test data:  0.8955823293172691
f1-score for test data:  0.8259259259259261
```

trained coefficients:

```
array([[-1.01651268, -4.2430088 , -0.26512859,  8.00314177, -0.36365244
,
        -0.15604157, -0.32506609, -0.02215078, -0.01758225, -0.01434175
]])
```