# Mobile Price Classification

در این تمرین دیتاستی مربوط به قیمت موبایل شامل 2000 ریکورد داریم. با بررسی این داده ها متوجه می شویم که داده ها به 4 کلاس قیمتی تقسیم می شوند که تعداد داده های در هر کلاس برابر 500 می باشد و یعنی این دیتاست متوازن است.

| | |
|---|---|
| 1 | 500 |
| 2 | 500 |
| 3 | 500 |
| 0 | 500 |

همچنین مشاهده می شود که در این دیتاست ریکوردی وجود ندارد که دارای مقدار null در هرکدام از فیچر ها باشد.

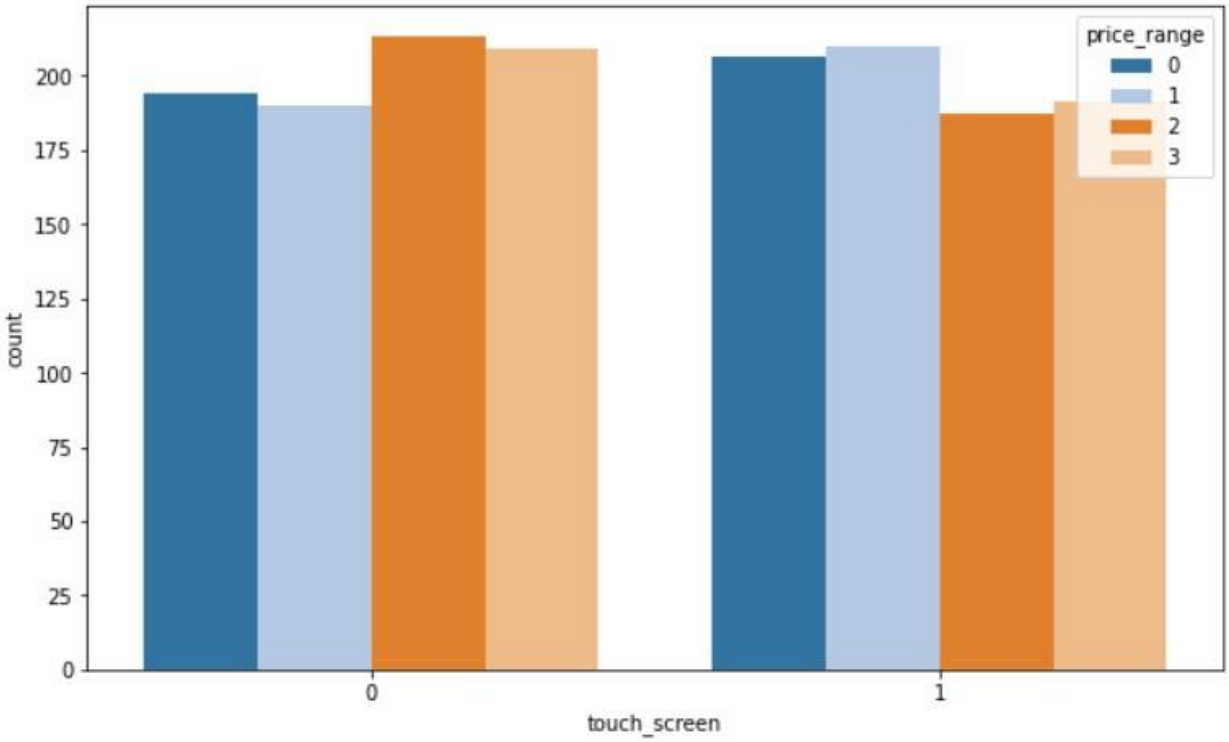
| | |
|---|---|
| battery_power | 0 |
| blue | 0 |
| clock_speed | 0 |
| dual_sim | 0 |
| fc | 0 |
| four_g | 0 |
| int_memory | 0 |
| m_dep | 0 |
| mobile_wt | 0 |
| n_cores | 0 |
| pc | 0 |
| px_height | 0 |
| px_width | 0 |
| ram | 0 |
| sc_h | 0 |
| sc_w | 0 |
| talk_time | 0 |
| three_g | 0 |
| touch_screen | 0 |
| wifi | 0 |
| price_range | 0 |

حال این دیتاست را به دو بخش تست و ترین با نسبت 80 به 20 تقسیم می کنیم به این صورت که در داده های ترین و تست نیز توازن برقرار باشد یعنی از هر کلاس در داده های ترین 400 ریکورد و در داده های تست 100 ریکورد قرار می دهیم.

از تست chi_2 که برای بررسی وجود ارتباط بین دو فیچر کتگوریکال است، برای بررسی وجود ارتباط price_range و touch_screen استفاده می کنیم.
عدم وجود ارتباط بین این دو فیچر را به عنوان فرض صفر در نظر می گیریم. با اجرای این تست، مقدار P_value را در زیر می بینیم:

```
p value is 0.2873700303633279
Probably independent(H0 accepted)
```

از آنجایی که مقدار P_value بیشتر از 0.05 است، فرض صفر ما تایید می شود و این یعنی بین این دو فیچر ارتباطی وجود ندارد.
شکل زیر نیز نتیجه همین تست را نشان می دهد. این شکل نشان می دهد که تعداد ریکورد های هر کلاس در دو حالت لمسی بودن و یا نبودن، تقریبا برابر است.




حال این تست را بر روی فیچرهای price_range و three_g اجرا می کنیم که نتیجه را در زیر می بینیم:

```
p value is 0.7736527543297278
Probably independent(H0 accepted)
```

این تست نشان می دهد که این دو فیچر نیز ارتباطی با یکدیگر ندارند.

از one sample Ttest بر روی فیچر battery_power استفاده می کنیم تا میانگین این فیچر را بررسی کنیم. با توجه به مقادیر max و min که در شکل زیر نشان داده شده است، حدس می زنیم که میانگین این فیچر برابر 1400 باشد و این را به عنوان فرض صفر در نظر می گیریم.

```
mean     1239.182500
std       439.309407
min       501.000000
25%       856.000000
50%      1225.000000
75%      1617.000000
max      1998.000000
```

نتیجه این تست نشان می دهد که فرض صفر رد می شود و میانگین این فیچر(که در واقع برابر با 1239 می باشد)، فاصله معناداری با 1400 دارد.

```
P value =1.1770923789759746e-45
H0 rejected, mean of this sample is not equal to 1400.
```

این تست را بر روی فیچر ram با این فرض صفر که مقدار میانگین این فیچر برابر با 2100 باشد، اجرا می کنیم.

```
mean     2127.402500
std      1084.028794
min       256.000000
25%      1219.500000
50%      2130.500000
75%      3079.250000
max      3998.000000
```

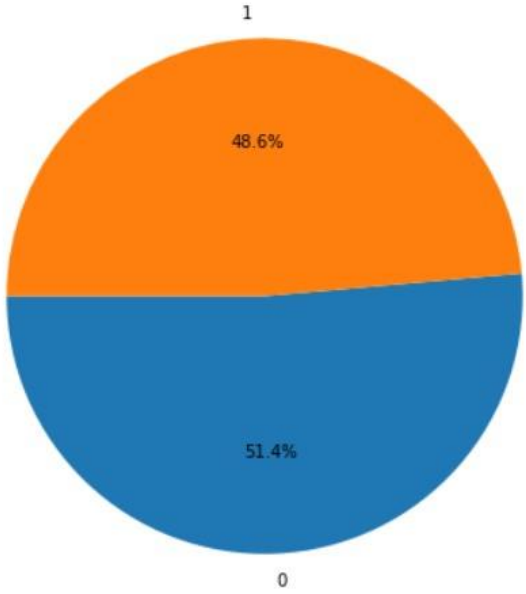
نتیجه این تست در زیر آمده است که به ما نشان می دهد که فاصله میانگین واقعی این فیچر(که 2127 است) فاصله معناداری با 2100 ندارد:

```
P value =0.312104557629628
H0 accepted, mean of this sample is equal to 2100.
```

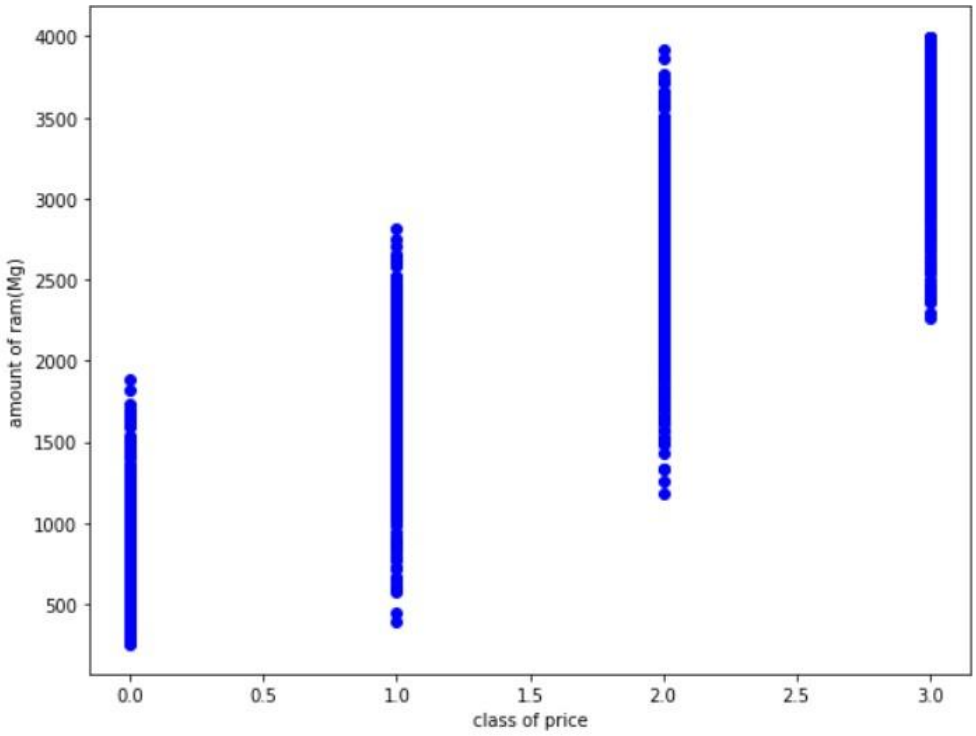
از تست spearman برای بررسی وجود رابطه بین دو فیچر عددی استفاده می شود و این تست را بر روی دو فیچر mobile_wt و m_dep اجرا می کنیم. فرض صفر را عدم وجود ارتباط بین دو فیچر در نظر می گیریم و با توجه به نتیجه این تست که در زیر نمایش داده شده، فرض صفر تایید می شود پس ارتباطی بین این دو فیچر وجود ندارد.

```
P value =0.28183540492732034
H0 accepted, There is not a significant linear relationship.
```

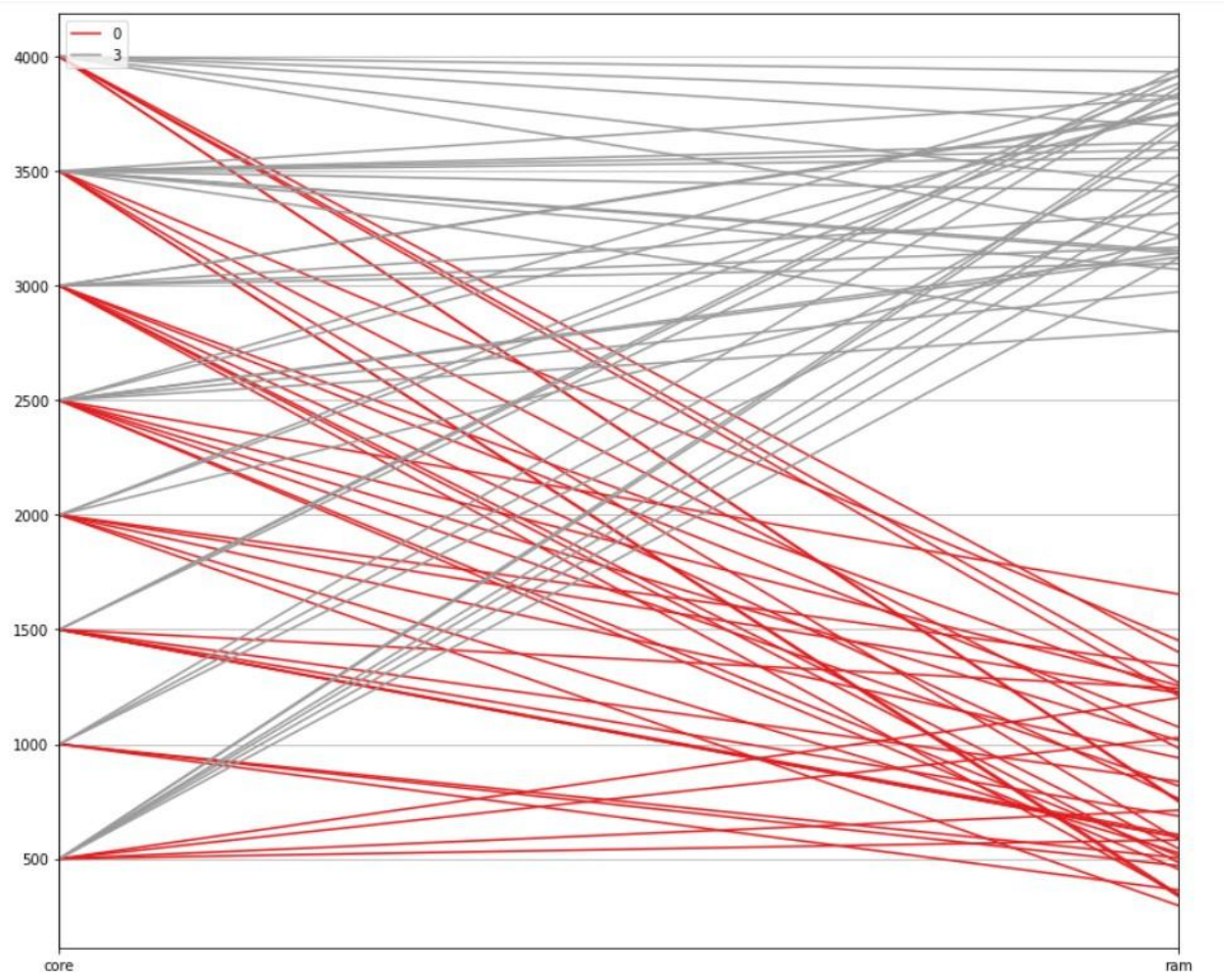
نمودار زیر نحوه توزیع داده ها در کلاس اول قیمتی از نظر بلوتوث را نشان می دهد که بخش آبی رنگ آن که درصد بیشتری از داده ها را پوشش می دهد، مربوط به گوشی های کلاس 0 قیمتی می باشد که بلوتوث ندارند.



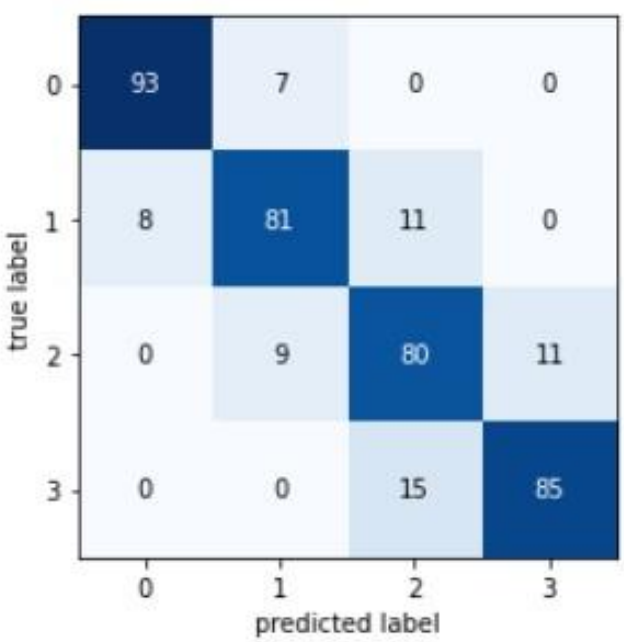

شکل زیربیانگر این است که در هر کلاس قیمتی، گوشی ها از چه مقدار ram برخوردار هستند. همان طور که حدس زده می شود و از شکل نیز پیداست، با افزایش مقدار ram قیمت نیز افزایش می یابد. این حقیقت به وضوح با مقایسه کلاس های 0 و شماره 3 قابل فهم است.



با کمک نمودار زیر می توانیم رابطه بین مقدار ram و تعداد core ها را در کلاس های 0 و 3 قیمتی بررسی کنیم(خطوط قرمز مربوط به کلاس اول قیمتی و خطوط خاکستری نیز مربوط به کلاس آخر قیمتی می باشند).
همان طور که از شکل قبل فهمیدیم، با افزایش مقدار ram مقدار قیمت نیز افزایش می یابد ولی به نظر رابطه خاصی بین مقدار ram و تعداد هسته های آن وجود ندارد. (این نمودار به علت شلوغ شدن شکل و سخت شدن فهم نمودار، روی همه داده ها رسم نشده بلکه 40 داده از کلاس 0 و 40 داده از کلاس شماره 3 به عنوان نمونه انتخاب شده اند. همچنین برای راحت تر شدن فهم نمودار و یکسان شدن scale دو فیچر، مقادیر فیچر core (تعداد هسته) در عدد ثابت 500 ضرب شده است.)

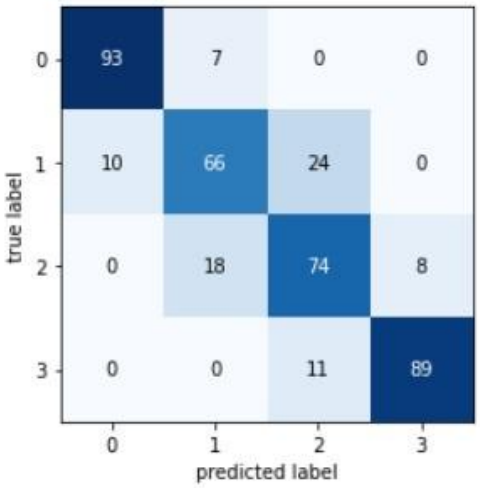


در این تمرین از 3 مدل کلاسیفایر برای پیش بینی کلاس قیمتی داده ها استفاده شده است که در ادامه به صورت مفصل آن ها را بررسی می کنیم.
اولین مدل استفاده شده مدل Decision Tree می باشد. OVR و OVO برای مدل هایی که کاربردشان در مسائل binary است می باشد و از آنجایی که این مدل ذاتا برای مسائل multuclass است بنابراین OVR و OVO برای این مدل معنایی ندارد.
مقدار score مربوط به اجرای این مدل بر روی داده ها برابر 84.75% می باشد و همچنین confusion matrix را در زیر می بینیم:



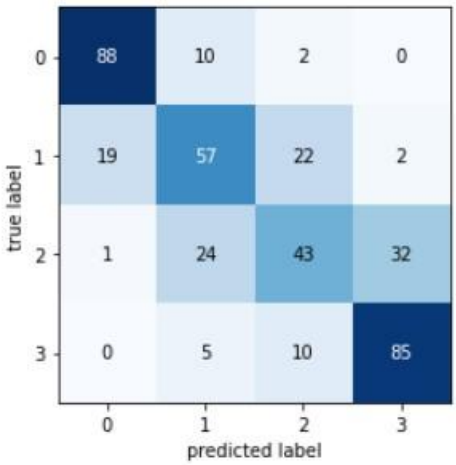

از آنجایی که در داده های تست از هر کلاس قیمتی 100 ریکورد وجود دارد، مشاهده می شود که مدل بر روی داده های کلاس 0 عملکرد خوبی داشته و آن ریکورد هایی که نتوانسته به عنوان این کلاس تشخیص دهد را به عنوان کلاس شماره یک شناسایی کرده که تفاوت زیادی با کلاس 0 ندارند و بسیار نزدیک بوده. بدترین عملکر نیز مربوط به کلاس شماره 2 می باشد.

مدل بعدی که استفاده کردیم Naive Bayes می باشد که این مدل نیز مثل مدل قبلی ذاتا برای مسائل multiclass می باشد. مقدار score برای این مدل برابر 80.5 می باشد که نسبت به درخت تصمیم عملکرد ضعیف تری داشته.



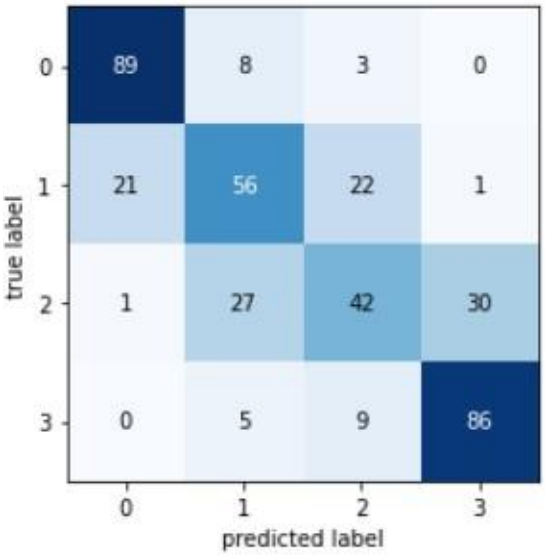

در این مدل هم داده های کلاس 0 به خوبی تشخیص داده شده اند ولی ضعیف ترین عملکرد بر روی داده های مربوط به کلاس شماره یک بوده زیرا تعداد زیادی از ریکورد های مربوط به این کلاس را به طور اشتباه به عنوان ریکورد های کلاس شماره 2 شناخته.

مدل بعدی که استفاده شده، Logistic Regression است. با کمک پارامتر multi_class که در هنگام تعریف مردن این مدل می توان آن را ست کرد، این مدل را به صورت OVR تعریف می کنیم. مقدار score اجرای این مدل بر روی داده های ترین برابر 68.25 است که نسبت به دو مدل قبلی بسیار ضعیف تر عمل کرده.




این مدل نیز مانند مدل های قبلی بر روی داده های کلاس 0 بهترین عملکرد را داشته ولی بر روی داده های کلاس شماره 2 عملکرد بسیار بدی داشته.

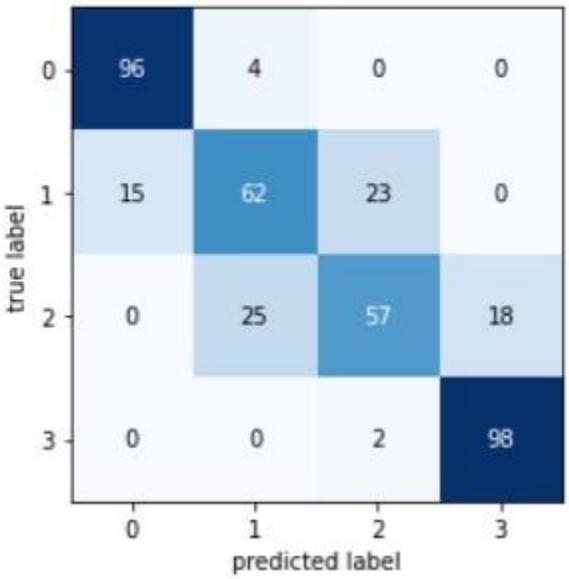
یک بار دیگر مدل را با ایجاد تغییر در پارامتر های آن تعریف می کنیم به این صورت که مقدار max_iter را برابر 200 قرار می دهیم که این مقدار به صورت پیش فرض برابر 100 می باشد. مقدار score نسبت به حالت قبلی تغییری نمی کند و همان 68.25 به دست می آید. البته با توجه در confusion matrix که در زیر آمده است، متوجه مقداری تغییرات جزئی می شویم.




از آنجایی که دو برابر کردن تعداد iteration تاثیر خاصی نداشت، یک بار دیگر در پارامتر های ورودی مدل تغییر ایجاد می کنیم و مدل را به شکل زیر تعریف می کنیم:

```
LogisticRegression(multi_class='ovr', solver='liblinear')
```

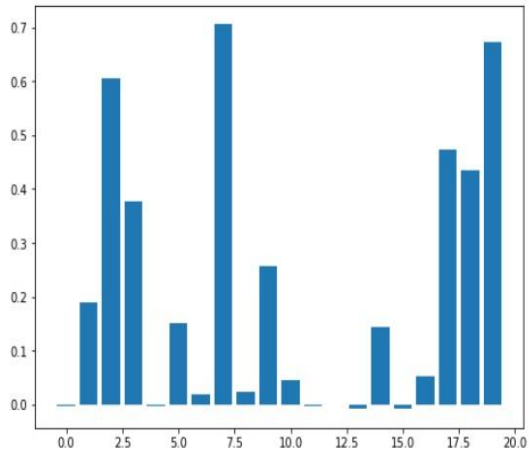
اجرای این مدل مقدار score را به مقدار قابل توجه بهبود می دهد و به مقدار آن به 78.25 می رسد.



عملکرد این کدل بر روی داده های کلاس 0 از مدل های قبلی بهتر بوده همچنین بر روی داده های کلاس 3 نیز به خوبی عمل کرده و به دقت 98% بر روی این کلاس رسید که نسبت به مدل های قبلی پیشرفت چشمگیری بوده.

با کمک coef که همگرا بودن فیچر ها در تابع تصمیم گیری را نشان می دهد، مقدار تاثیر گذاری هر فیچر بر روی این مدل را در زیر مشاهده می کنیم:

```
Feature: battery_power,   Score: -0.0032190544425082825
Feature: blue,   Score: 0.18993736311794207
Feature: clock_speed,   Score: 0.6053635801079029
Feature: dual_sim,   Score: 0.37709421790352365
Feature: fc,   Score: -0.0026652975433267344
Feature: four_g,   Score: 0.1501863819862536
Feature: int_memory,   Score: 0.019287494731801597
Feature: m_dep,   Score: 0.7051967916184363
Feature: mobile_wt,   Score: 0.022613907615278116
Feature: n_cores,   Score: 0.25773435197556166
Feature: pc,   Score: 0.044136813441349426
Feature: px_height,   Score: -0.002163392513048264
Feature: px_width,   Score: -0.0014376738317832735
Feature: ram,   Score: -0.006640928060468122
Feature: sc_h,   Score: 0.1432903033579298
Feature: sc_w,   Score: -0.007490001567736572
Feature: talk_time,   Score: 0.05173485650721136
Feature: three_g,   Score: 0.4738842866020957
Feature: touch_screen,   Score: 0.4350590211686442
Feature: wifi,   Score: 0.6734053644662799
```




بهترین مدل Logistic Regression را یک بار با فیچر هایی که تاثیر مثبتی بر روی مدل دارند و یک بار نیز با فیچر هایی که مقدار score بدست آمده آن ها به کمک coef بیشتر از 0.09 باشد اجرا می کنیم.

```
over_9_features
```

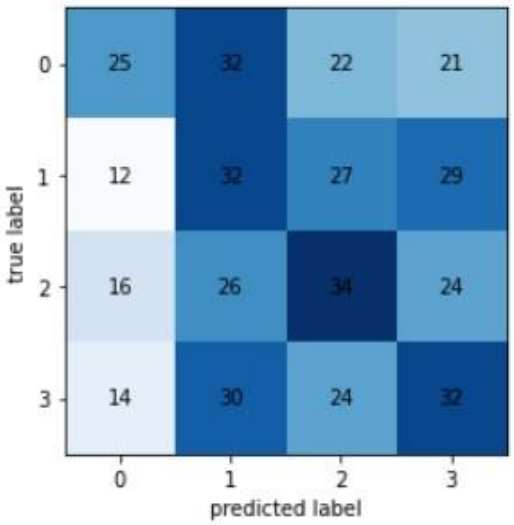
```
['blue',
 'clock_speed',
 'dual_sim',
 'four_g',
 'm_dep',
 'n_cores',
 'sc_h',
 'three_g',
 'touch_screen',
 'wifi']
```

```
positive_features
```

```
['blue',
 'clock_speed',
 'dual_sim',
 'four_g',
 'int_memory',
 'm_dep',
 'mobile_wt',
 'n_cores',
 'pc',
 'sc_h',
 'talk_time',
 'three_g',
 'touch_screen',
 'wifi']
```

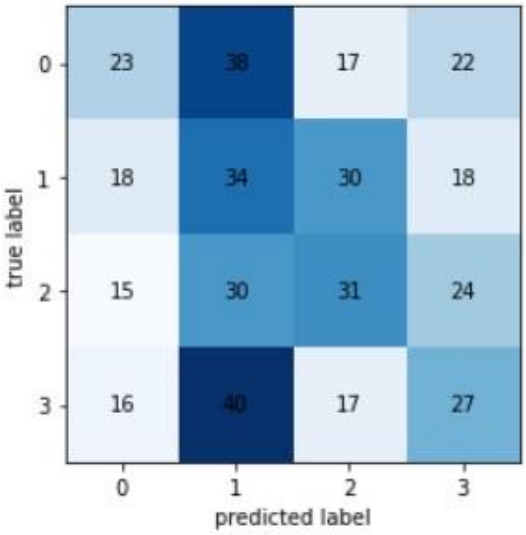
لیست فیچر هایی که با در دو حالت کفته شده، انتخاب شدند

مقدار score مدلی که با فیچر های با تاثیر مثبت اجرا شد، به عدد 30.75 رسید.

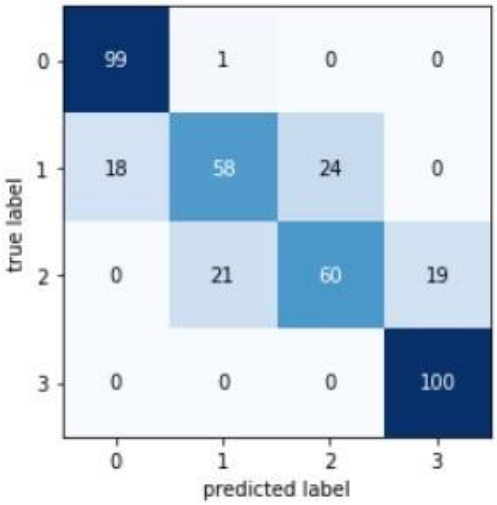



با توجه به confusion matrix متوجه می شویم که مدل در هر کلاس تقریبا 30% از ریکورد ها را به درستی تشخیص داده بنابراین احتمالا فیچر هایی که حذف کردیم به مدل کمک می کردند تا بتواند تفاوت های بین کلاس های مختلف را تشخیص دهد.

در حالتی که مدل را با فیچر هایی که score بیشتر از 0.09 داشتند اجرا کردیم، مقدار score به 28.75 کاهش یافت.

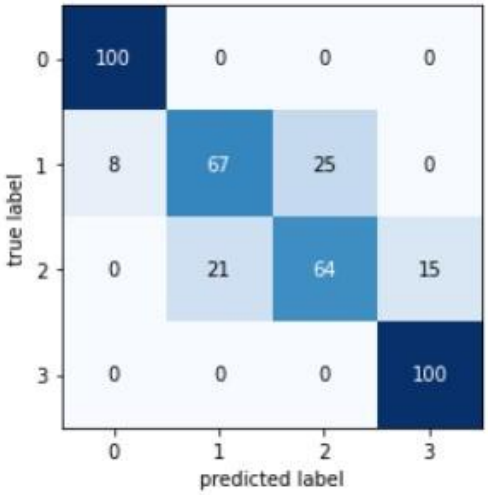


از MinMax Scaler و Standard Scaler بر روی داده ها با همه فیچر ها استفاده می کنیم تا ببینیم آیا scale کردن داده ها تاثیری در عملکرد مدل دارد یا خیر.
score حاصل از اجرای مدل Logistic Regression بر روی داده هایی که تحت MinMax Scale قرار گرفتند برابر 79.25 می باشد.




مشاهده می شود که مدل بر روی داده های کلاس 0 و کلاس شماره 3 همه ریکورد ها را درست پیش بینی کرده و در کلاس های شماره یک و 2 نیز تغییر زیادی اعمال نشد.

score حاصل از اجرای مدل Logistic Regression بر روی داده هایی که تحت Standard Scale قرار گرفتند بهبود یافته و برابر 82.25 می باشد.



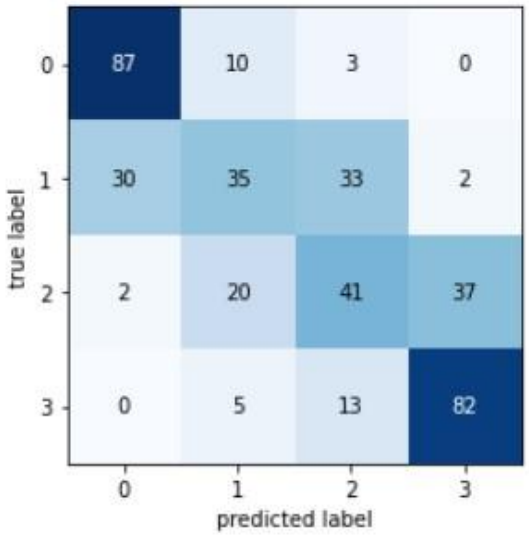

این بهبود عملکرد واضحا به دلیل پیش بینی های بهتر کلاس های شماره یک و 2 می باشد.

در زیر نتیجه اعمال PCA بر روی داده ها با POV های مختلف نشان داده شده:

## POV = 0.75

```
number of features with pov 0.75  -> 13
```

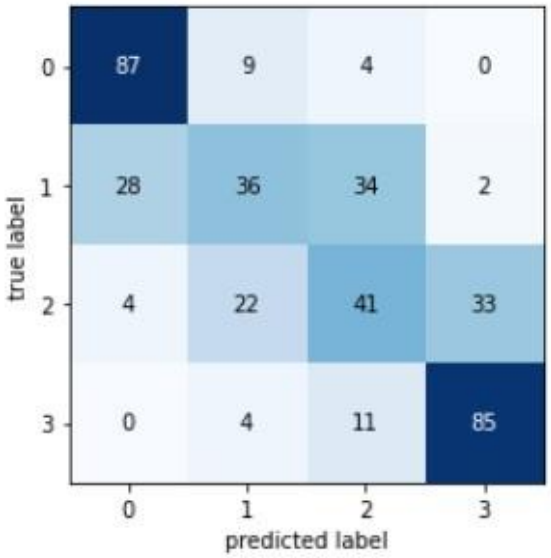
مقدار score در این حالت برابر 61.25 می باشد.




## POV = 0.80

```
number of features with pov 0.80  -> 14
```

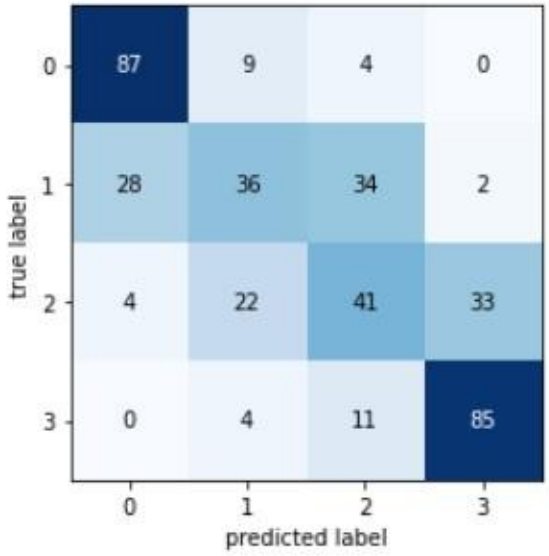
مقدار score در این حالت برابر 62.25 می باشد.



## POV = 090

```
number of features with pov 0.90  -> 16
```

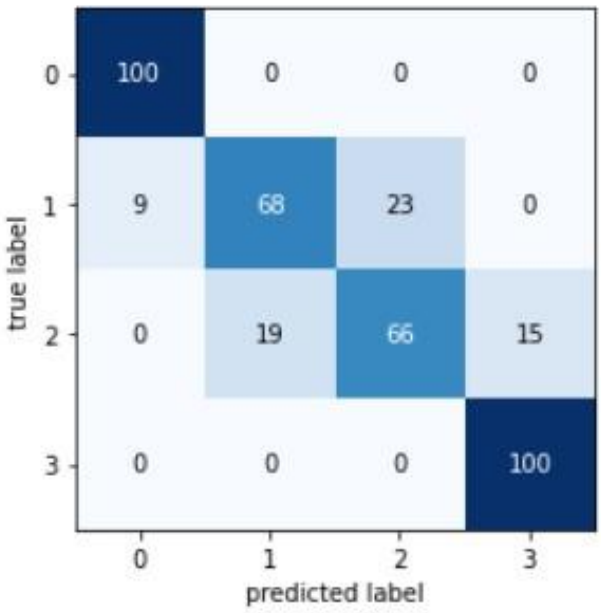
مقدار score در این حالت برابر 83.25 می باشد.




## POV = 0.95

```
number of features with pov 0.95  -> 18
```

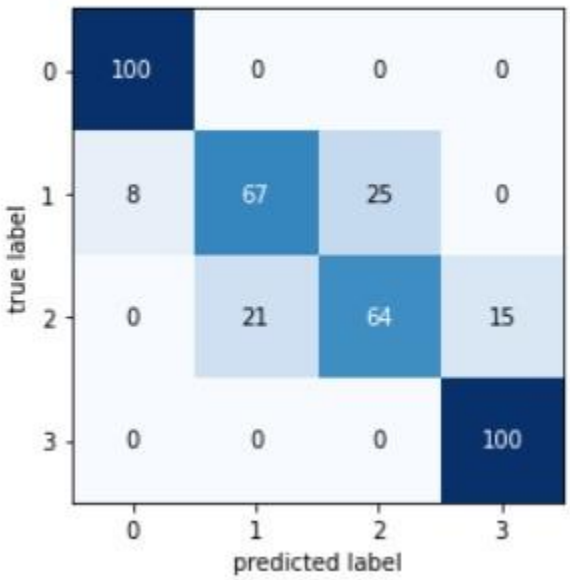
مقدار score در این حالت برابر 83.5 می باشد.



## POV = 0.99

```
number of features with pov 0.99  -> 20
```

مقدار score در این حالت برابر 82.75 می باشد.

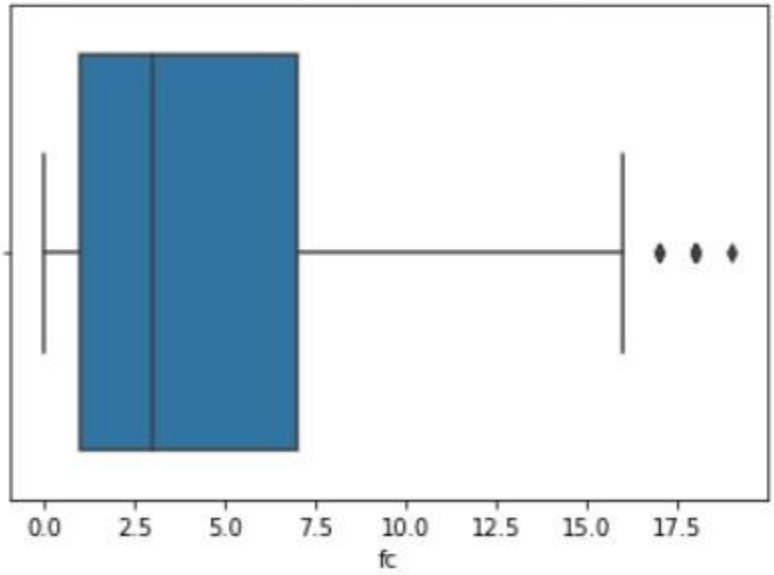



از الگوریتم PCA استفاده می کنیم تا هزینه محاسباتی را کاهش دهیم. این الگوریتم به ما کمک می کند تا فیچر هایی که با یکدیگر همبستگی دارند را ترکیب کنیم و noise را کاهش دهیم. به وضوح این امر سبب می شود که مدل یادگیری بهتری داشته باشد چون اطلاعات مهم تر را می بیند و کمتر گیج می شود ولی اگر مقدار تعداد فیچر ها را خیلی کاهش دهیم منجر به این می شود که برخی اطلاعات مهم نیز از دست می روند و مدل ورودی لازم برای یادگیری خوب را نخواهد داشت و عملکرد مدل کاهش می یابد. بر همین اساس در POV=0.99 که تعداد فیچر ها را برابر 20 می کند، دیتاست هنوز شامل noise می باشد ولی وقتی مقدار POV را به 0.95 کاهش می دهیم، تعداد فیچر ها برابر 18 می شود و مدل بهترین عملکرد را دارد و حتی این عملکرد از نتیجه اجرای مدل بر روی داده هایی که تحت Standard Scale قرار داشتند هم بهتر است. حال هرچه مقدار POV را کاهش دهیم، عملکرد مدل ضعیف تر می شود زیرا اطلاعات مهمی از دست می روند به طوری که وقتی تحت POV=0.8 تعداد فیچر ها به 14 می رسد، دقت مدل به شدت کاهش می یابد.

راه های مختلفی برای کار کردن با دیتاست نامتوازن وجود دارند. یکی از این راه ها این است که اگر امکان جمع آوری داده های بیشتری وجود دارد، به ریکورد های دیتاست بیافزاییم. یکی دیگر از راه ها resample کردن است، یعنی تعدادی از ریکورد های کلاسی که تعداد ریکورد های زیادی دارد را حذف کنیم تا توازن برقرار شود (down sampling) یا اینکه به ریکورد های کلاسی که تعداد ریکورد های آن کمتر است، بیافزاییم(up sampling). یکی دیگر از راه های کار کردن با دیتاست نامتوازن این است که الگوریتم مورد استفاده خود را تغییر دهیم، مثلا الگوریتم Decision Tree معمولا روی دیتاست های نامتوازن عملکرد خوبی دارد. یک راه دیگر این است که از الگوریتم های penalized استفاده کنیم تا مقدار تابع هزینه برای ریکورد

های کلاسی که اندازه کوچک تری دارد، افزایش یابد تا از طریق مقدار تابع هزینه بتوانیم کلاس را تشخیص دهیم.
با نامتوازن کردن دیتاست و اجرای مدل، دقت مدل به 98.75 می رسد ولی می دانیم نتیجه روی دیتاست نامتوازن معمولا قابل استناد نمی باشد.
با کمک resample از پکیج sklearn ،تعداد داده های کلاسی که تعداد کمتری ریکورد دارند را به تعداد ریکورد های کلاسی که تعداد بیشتری دارد، می رسانیم. یعنی دو کلاس داریم که هر کدام شامل 1200 ریکورد می باشند. نتیجه اجرای مدل در این حالت برابر 98.25 می باشد.

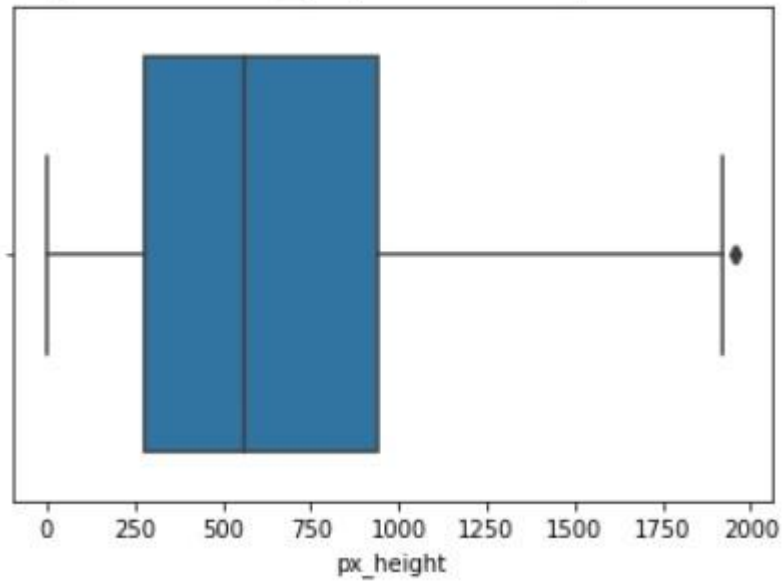
نمودار جعبه ای زیر مربوط به مقادیر داده ها در فیچر fc می باشد که به وضوح دارای مقادیر پرت می باشد.




با کمک روش IQR این داده ها را شناسایی می کنیم و از دیتا ست حذف می کنیم. در زیر تعداد داده ها قبل و بعد از حذف داده های پرت مربوط به فیچر fc را مشاهده می کنیم:

```
Old Shape:  (1600, 21)
New Shape:  (1572, 21)
```

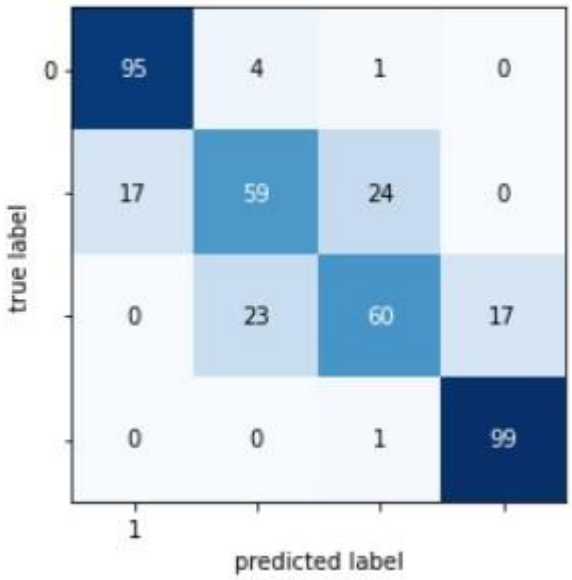
نمودار جعبه ای زیر نیز مربوط به مقادیر داده ها در فیچر px_height می باشد که مشاهده می شود این فیچر هم دارای مقادیر پرت می باشد.




این فیچر فقط دو داده پرت دارد.

```
Old Shape:  (1572, 21)
New Shape:  (1570, 21)
```

حال این دیتاست که داده های پرت از آن حذف شده اند را به مدل می دهیم و نتیجه آن 78.25 می باشد یعنی تغییری در نتیجه مدل وقتی داده های پرت آن حذف نشده بودند، ایجاد نشد.



امیرحسین باباجانی
97222009

https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/