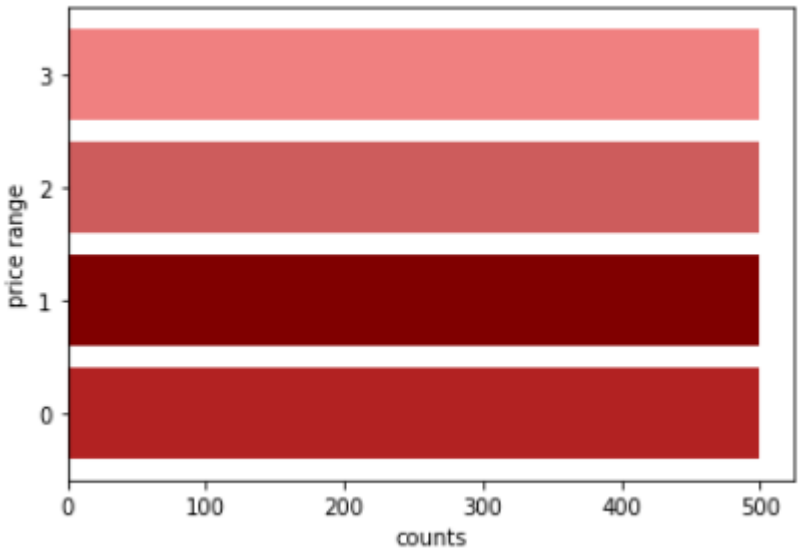
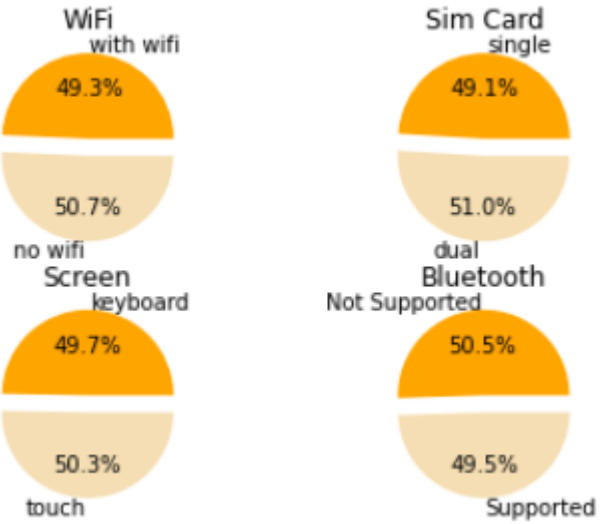
Visualization.1: در اولین مرحله کار مصور سازی را انجام می‌دهیم. در این مرحله می‌توان اطلاعات زیادی راجع به داده ها و وضعیت آن ها به دست آورد و با استفاده از آنها راجع به کیفیت داده بحث کرد .

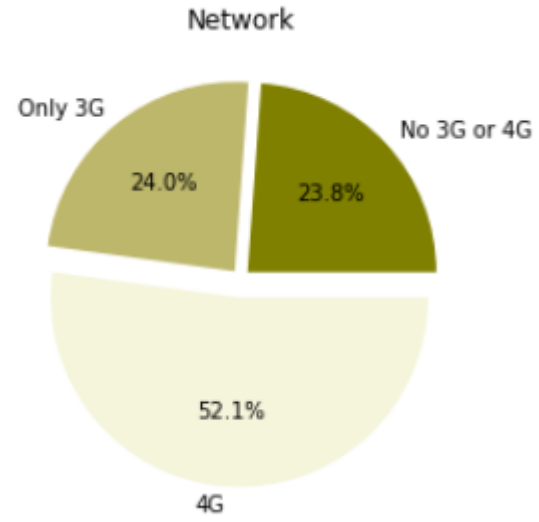
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 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
```

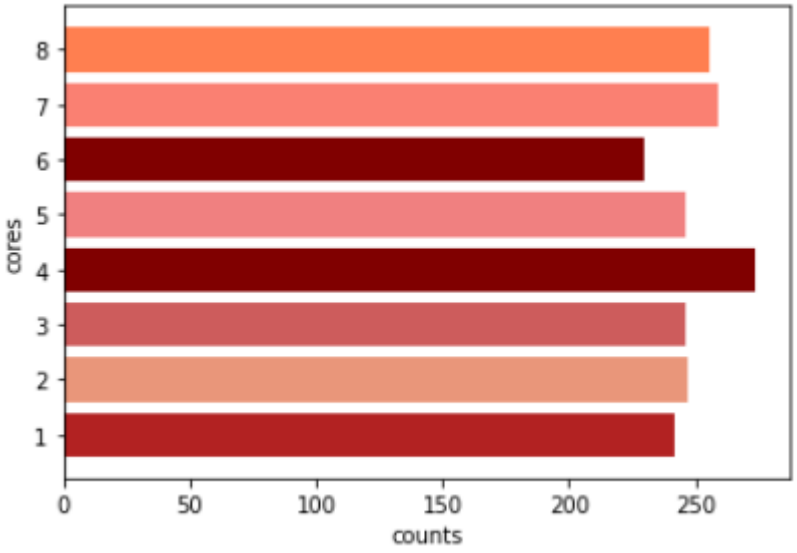
دیتاست این تمرین شامل ۲ هزار رکورد با ۲۱ فیچر است که دیتا تایپ آنها در اینجا نمایش داده شده است.



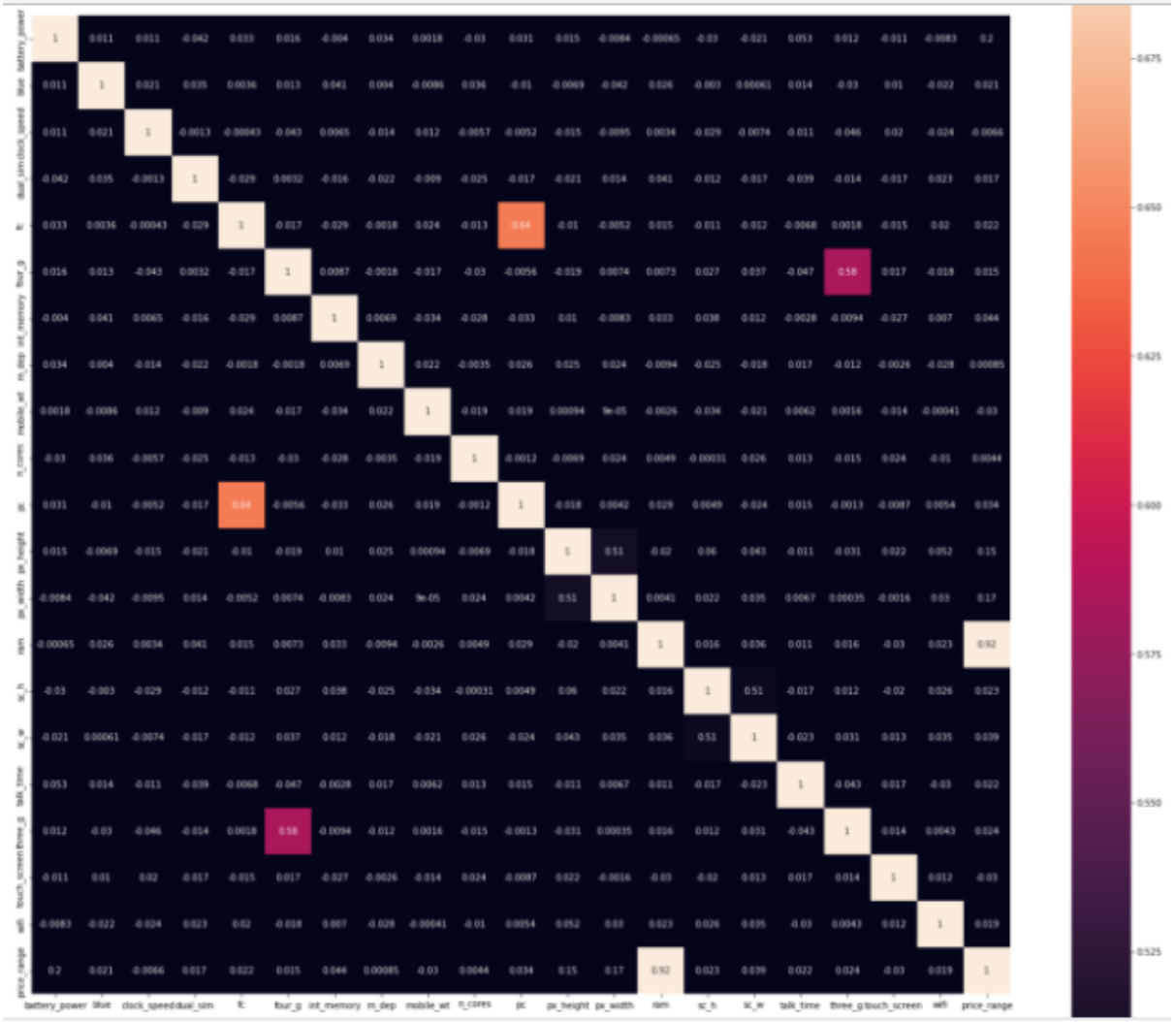
این نمودار نشان می دهد که برای هر رنج به تعداد مساوی 500 رکورد ثبت شده است است در نتیجه دیتا بالانس است.



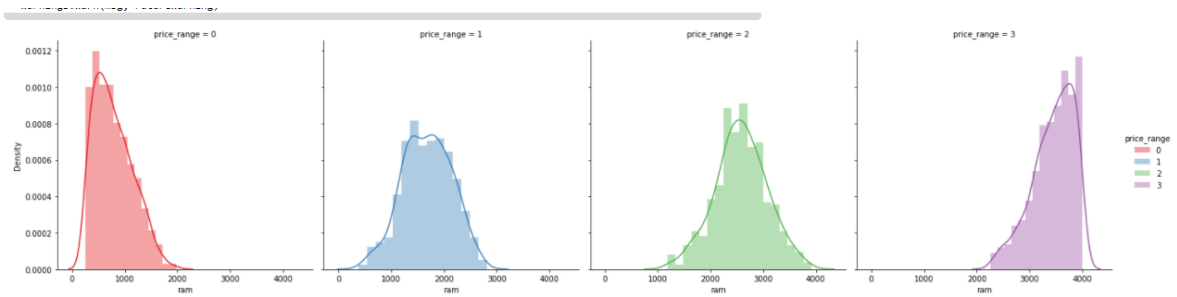




این نمودارها وضعیت دستگاه ها را در مورد فیچرهای مختلف نشان میدهدد.



با استفاده از این نمودار میشود فیچرهایی(به جز تارگت) که correlation بالایی دارند را تشخیص داد و در صورت لزوم یکی از انهارا دراپ کرد.(چنین موردی وجود ندارد)
همچنین میشود فیچرهایی که با تارگت correlation بالایی دارند شناسایی کرد.(مشاهده میشود که Ram بیشترین correlation با price_range را دارد.)



نشان دادن correlation بین Ram و price_range به شکلی دیگر

همچنین این نمودار نشان دهنده این است که price_range دارای correlation بالاتری با ram است نسبت به battery_power .

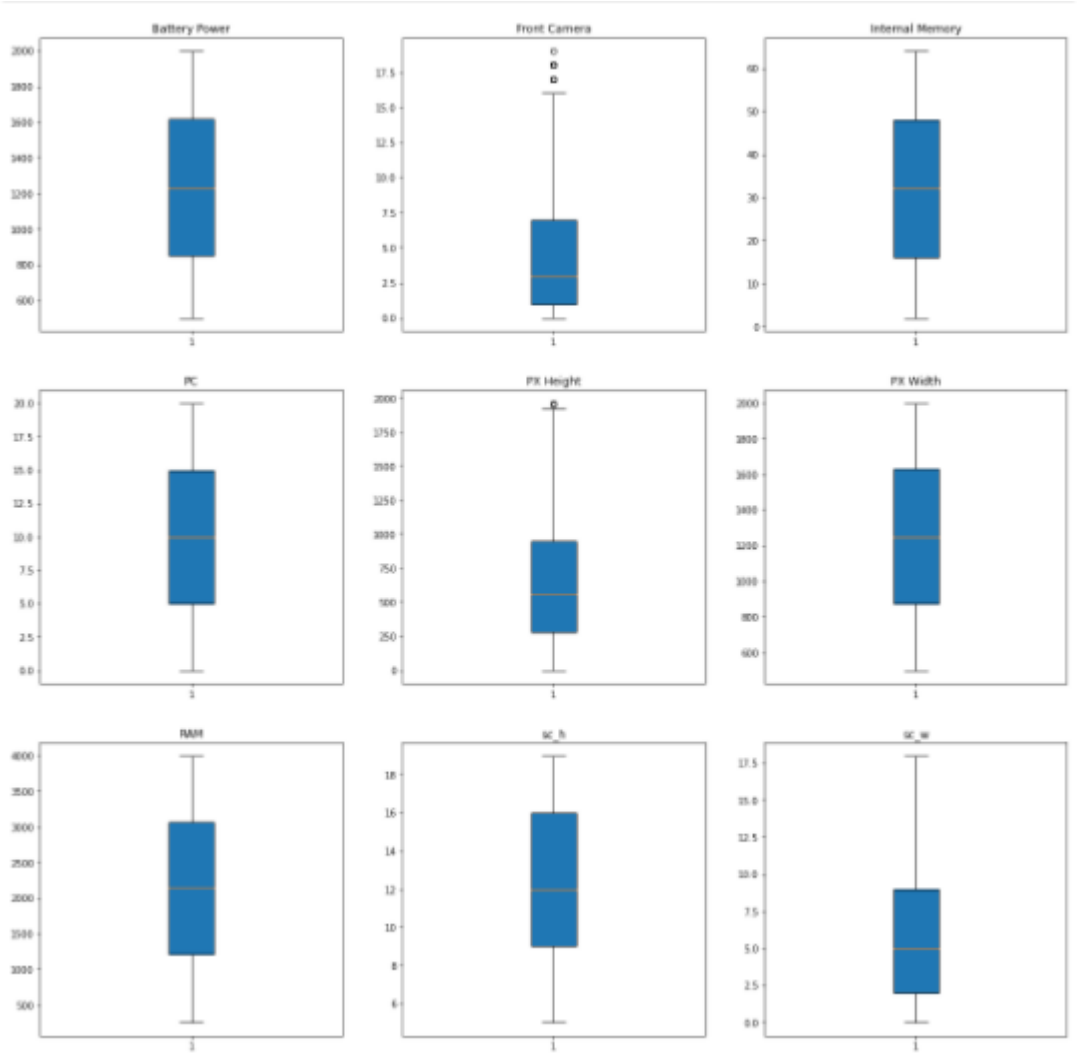
Preprocessing.2: در مرحله دوم با توجه به نتتایج به دست آمده از مرحله قبل و با استفاده از تحقیقات بیشتر دیتا را پاکسازی می‌کنیم.

```
price_range      1.000000
ram              0.917046
battery_power    0.200723
px_width         0.165818
px_height        0.148858
int_memory       0.044435
sc_w             0.038711
pc               0.033599
three_g          0.023611
sc_h             0.022986
fc               0.021998
talk_time        0.021859
blue             0.020573
wifi             0.018785
dual_sim         0.017444
four_g           0.014772
n_cores          0.004399
m_dep            0.000853
clock_speed     -0.006606
mobile_wt       -0.030302
touch_screen    -0.030411
Name: price_range, dtype: float64
```

بار دیگر correlation بین price_range و دیگر فیچرهارا به دست می‌آوریم و به غیر از 10 فیچر اول، بقیه را دراپ می‌کنیم.

```
ram              0
battery_power    0
px_width         0
px_height        0
int_memory       0
sc_w             0
pc               0
three_g          0
sc_h             0
fc               0
dtype: int64
```

این تصویر نشان می‌دهد که در هیچ کدام از 10 ستون باقی مانده داده نال نداریم.



از بین فیچر های باقی مانده، three_g دارای مقدار باینری است. و با توجه به نمودارهای بالا از 0 فیچر غیر باینری، 2 تای آنها دارای داده های پرت هستند که نیاز است پاک شوند.(با استفاده از کد پایین)

```
for column in new_data.columns:
    if not new_data[column].isin([0, 1]).all():
        upper = new_data[column].mean() + 3 * new_data[column].std()
        lower = new_data[column].mean() - 3 * new_data[column].std()
        rows = new_data[(new_data[column] > upper) | (new_data[column] < lower)].index

        new_data.drop(rows, inplace=True)
        target.drop(rows, inplace=True)
new_data
```

:Statistical hypothesis tests.3

از آزمون های فرض برای بررسی ادعاها یا فرض‌ها دربارهٔ پارامترهای توزیع در جوامع آماری استفاده می‌شود.

آزمون فرض اول:
میانگین قدرت باتری 1500 است.

```
72 from scipy import stats
   alpha = 0.05
   tstat, p_value = stats.ttest_1samp(new_data['battery_power'],popmean = 1500)
   print('t stat : {} , p_value : {}'.format(tstat,p_value))
   if p_value < alpha:
       print("null hypothesis rejected")
   else:
       print("null hypothesis accepted")
```

```
t stat : -26.621086802647067 , p_value : 8.560743994781834e-134
null hypothesis rejected
```

از آزمون ttest_1samp استفاده میکنیم، ttest برای بررسی میزان شباهت میانگین گروه است و با استفاده از 1samp میشود میانگین یک گروه را با یک مقدار عددی مقایسه کرد.
مقدار p value خیلی کمتر از 0.05 است در فرض رد می‌شود.

آزمون فرض دوم:
میانگین حافظه ی داخلی 32 است.

```
73 from scipy import stats
   alpha = 0.05
   tstat, p_value = stats.ttest_1samp(new_data['int_memory'],popmean = 32)
   print('t stat : {} , p_value : {}'.format(tstat,p_value))
   if p_value < alpha:
       print("null hypothesis rejected")
   else:
       print("null hypothesis accepted")
```

```
t stat : 0.21898330046738493 , p_value : 0.8266855466007182
null hypothesis accepted
```

مانند آزمون قبل، از ttest_samp1 استفاده میکنیم. در اینجا p_value بیشتر از 0.05 است و آزمون اکسپت می‌شود.

آزمون فرض سوم:
فیچرهای battery_power و ram مستقل اند.

```
   sstat, p_value = stats.spearmanr(new_data["ram"], new_data["battery_power"])
   print('t stat : {} , p_value : {}'.format(tstat,p_value))
   if p_value < alpha:
       print("null hypothesis rejected")
   else:
       print("null hypothesis accepted")
```

```
t stat : 0.21898330046738493 , p_value : 0.9837786460094284
null hypothesis accepted
```

با استفاده از متود spearmanr استقلال این دو فیچر را بررسی می‌کنیم و نشان داده می‌شود که مستقل اند.

آزمون فرض چهارم:
Px_height و px_width وابسته اند.

```
sstat, p_value = stats.spearmanr(new_data["px_height"], new_data["px_width"])
print('t stat : {} , p_value : {}'.format(tstat,p_value))
if p_value >= alpha:
    print("null hypothesis rejected")
else:
    print("null hypothesis accepted")
```

```
t stat : 0.21898330046738493 , p_value : 3.0744719356102394e-107
null hypothesis accepted
```

با استفاده از متود spearmanr استقلال این دو فیچر را بررسی می‌کنیم، p_value مقدار بسیار کمی دارد و از این نتیجه می‌شود که این دو فیچر وابسته اند.

آزمون فرض پنجم:
M_depبر price_range تاثیری ندارد.

```
fstat,p_value = stats.pearsonr(data["m_dep"],data["price_range"])

print('f stat : {} , p_value : {}'.format(fstat,p_value))
if p_value <= alpha:
    print("null hypothesis rejected")
else:
    print("null hypothesis accepted")
```

```
f stat : 0.0008530365050864429 , p_value : 0.9695879315808612
accept null hypothesis
```

Pearsonr متود دیگری برای برررسی استقلال استو نشان میدهد که فرض پذیرفته شده است.

classification.4: در این مرحله مدل برای کلاس بندی داده ها پیشنهاد می‌شود.

ابتدا با استفاده از train_test_split داده هارا به نسبت 8 به 2 تقسیم می‌کنیم و با solver=newton-cg و penalty=l2 ، Logistic regression را اجرا میکنیم.

```
Accuracy of Logistic Regression: 95.22613065326632

--------------------------------------------
              precision    recall  f1-score   support

           0       0.99      0.96      0.97        98
           1       0.92      0.94      0.93        86
           2       0.94      0.94      0.94       108
           3       0.96      0.97      0.97       106

    accuracy                           0.95       398
   macro avg       0.95      0.95      0.95       398
weighted avg       0.95      0.95      0.95       398
```

مدل train شده مشخصات بالا را دارد.

- 'solver='lbfgs', penalty='l2

```
Accuracy of Logistic Regression: 53.266331658291456

-------------------------------------------
              precision    recall  f1-score   support

           0       0.81      0.67      0.74        98
           1       0.44      0.47      0.45        86
           2       0.35      0.23      0.28       108
           3       0.52      0.76      0.62       106

    accuracy                           0.53       398
   macro avg       0.53      0.53      0.52       398
weighted avg       0.53      0.53      0.52       398
```

- 'solver='lbfgs', penalty='none

```
Accuracy of Logistic Regression: 53.015075376884425

-------------------------------------------
              precision    recall  f1-score   support

           0       0.81      0.67      0.74        98
           1       0.43      0.44      0.44        86
           2       0.35      0.24      0.29       108
           3       0.52      0.76      0.62       106

    accuracy                           0.53       398
   macro avg       0.53      0.53      0.52       398
weighted avg       0.53      0.53      0.52       398
```

با توجه به نتایج مدل اولین بهترین مدل می‌باشد.

5. Confusion matrix:

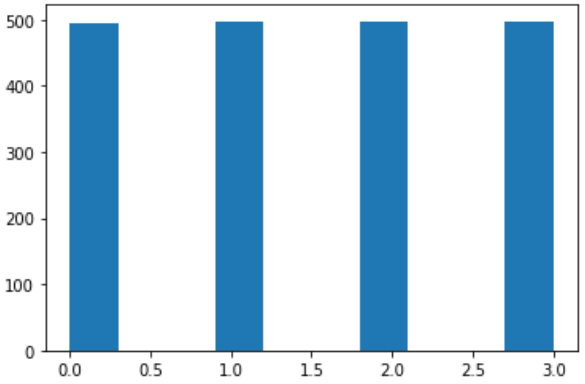
بررسی confusion matrix این سه مدل به ترتیب تایید کننده ی این است که اولین مدل بهتر از 3 مدل دیگر است زیرا تراکم داده ها روی قطر اصلی ماتریس است. دلیل این تفاوت این است که مدل دوم از حل کننده متفاوتی استفاده می کند و مدل سوم از پنالتی استفاده نمی کند.

```
confusion matrix for second model
[[66 25  5  2]
 [15 40 20 11]
 [ 0 21 25 62]
 [ 0  4 21 81]]
```

```
confusion matrix for third model
[[66 25  5  2]
 [15 38 22 11]
 [ 0 21 26 61]
 [ 0  4 21 81]]
```

```
confusion matrix for first model
[[ 94   4   0   0]
 [  1  81   4   0]
 [  0   3 101   4]
 [  0   0   3 103]]
```

:data balancing.6



همانطور که در بخش visualization هم اشاره شد داده ها بالانس هستند.
اما 3 راه برای بالانس کردن به صورت زیر است:

- Random under-sampling
- Random over-sampling
- NearMiss

:scaling.7

برای اسکیل کردن داده ها از دو روش mianmax و standard استفاده می‌کنیم.انجام تست و نتایج نشان می‌دهد که standard scaler روش بهتری برای اسکیل کردن این داده ها است.

:train test split.8

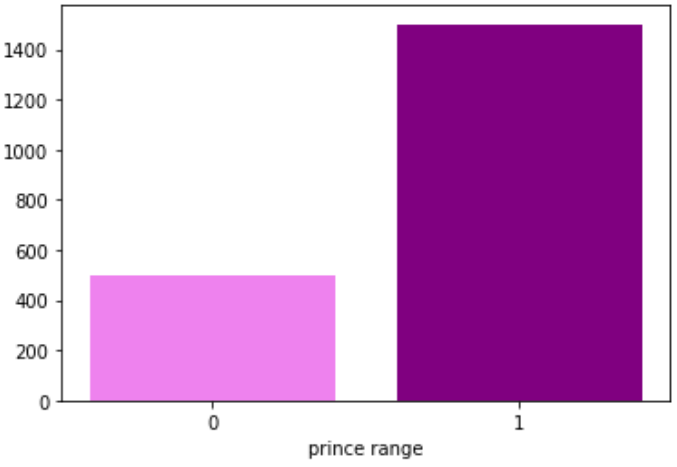
در بخش 4 با نسبت خواسته شده انجام شد.

:PCA.9

روشی است برای کاهش فیچرها. در این بخش با کاهش pov تعداد پارامتر ها هم کاهش میابد. به همین دلیل دقت مدل نیز کمتر میشود. مزیت کاهش پارامتر ها شامل کاهش هزینه محاسبه و همچنین راحتی visualization است.

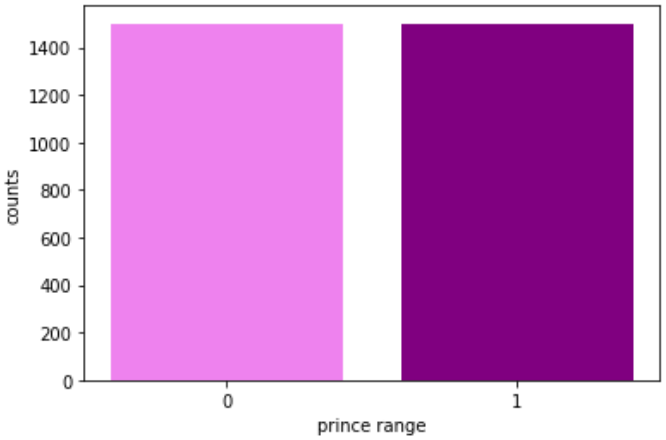
:imbalance data.10

در ابتدا داده های مربوط به کلاس 1و2و3 را تبدیل به کلاس 1 میکنیم.



مدل را روی داده های جدید فیت می‌کنیم و اسکور آن را ثبت میکنیم.

سپس داده ها را بالانس میکنیم.



و دوباره مدل را روی داده های بالانس شده فیت میکنیم.
دقت مدل را بررسی میکنیم و مشاهده میشود که بعد از بالانس کردن 0.002 افزایش یافته است.