نیما طاهری

1- Data cleaning

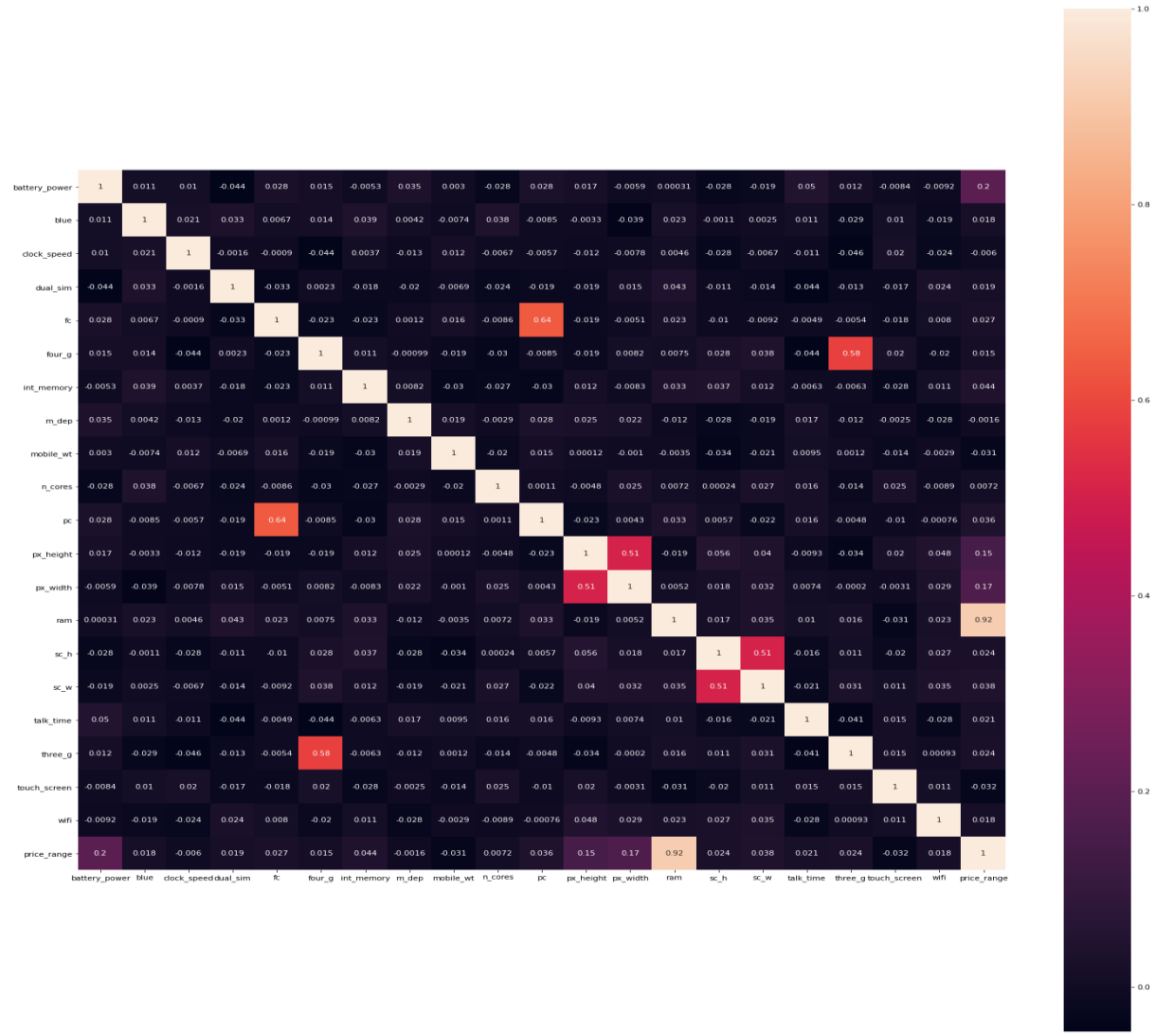
انجام پاکسازی داده ها: داده غیر موجود و موارد مشابهت نداشتیم و فقط 12 داده پرت را حذف کردیم.

2- Feature analyzing and data visualization

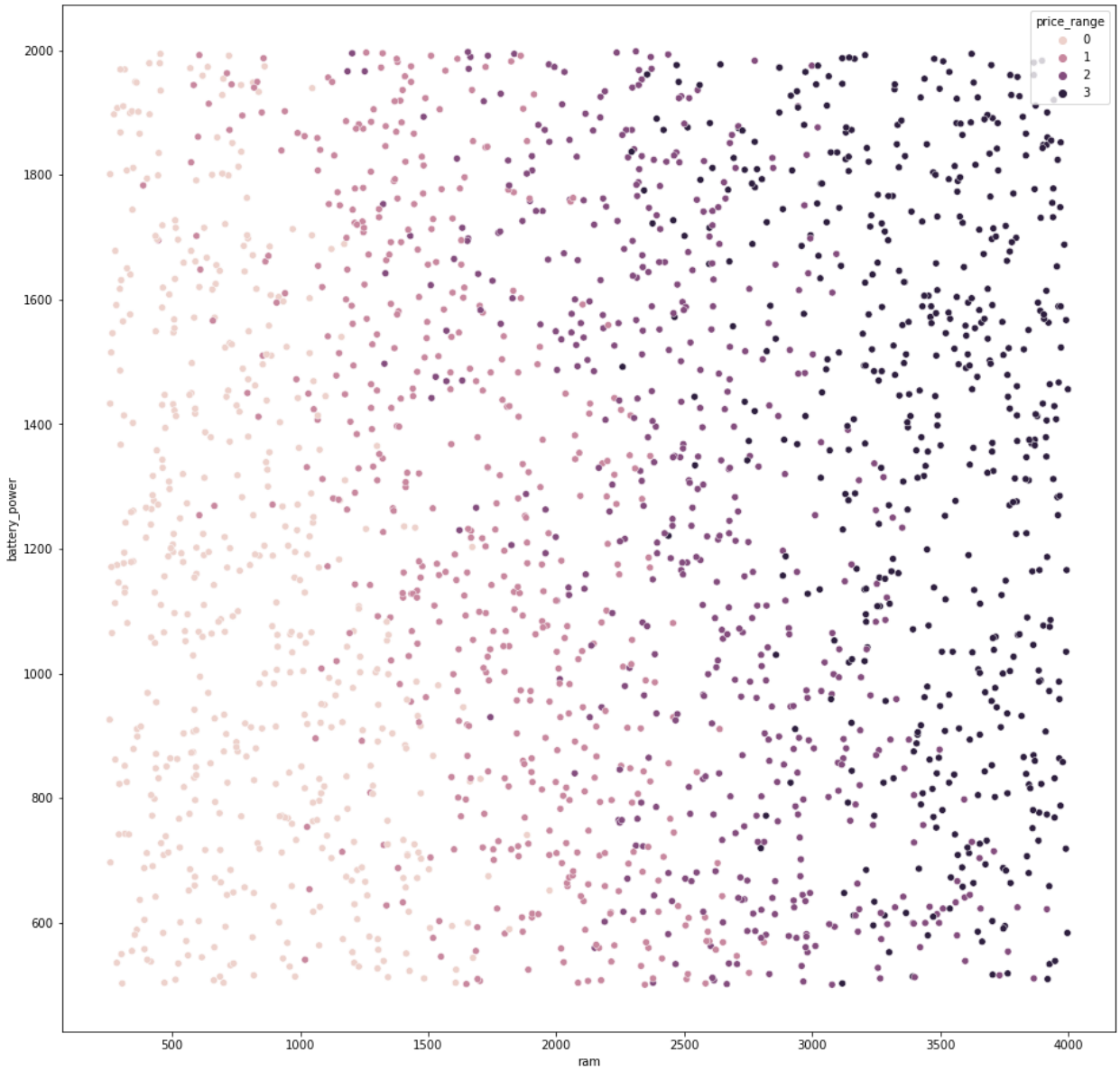
از تابع describe() مختص دیتافریم که در پکیج pandas قرار دارد برای برسی فیچر ها و اطلاعات آماری مربوط به آنها استفاده کردیم.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
count	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988	1988
mean	1237.593058	0.495473	1.522133	0.509054	4.226358	0.520624	32.089034	0.502012	140.159457	4.524145	9.858149	643.926559	1251.53571	2126.54427	12.30835	5.77163	11.014588	0.760563	0.502515	0.50503	1.501509
std	439.498835	0.500105	0.816811	0.500044	4.220051	0.4997	18.128175	0.288438	35.361078	2.28998	6.035626	442.961039	432.086772	1084.18632	4.215626	4.361399	5.459398	0.426847	0.500119	0.5001	1.118314
min	501	0	0.5	0	0	0	2	0.1	80	1	0	0	500	256	5	0	2	0	0	0	0
25%	851	0	0.7	0	1	0	16	0.2	109	3	5	282	874.75	1210.75	9	2	6	1	0	0	1
50%	1225	0	1.5	1	3	1	32	0.5	141	4	10	564	1247	2147.5	12	5	11	1	1	1	2
75%	1615	1	2.2	1	7	1	48	0.8	170	7	15	945.25	1632.25	3066.5	16	9	16	1	1	1	3
max	1998	1	3	1	17	1	64	1	200	8	20	1960	1998	3998	19	18	20	1	1	1	3

از heatmap برای نشان دادن ارتباط بین فیچر ها با هم و با محدوده قیمت استفاده کردیم و همانطور که مشخص است رم به طور قابل توجهی با قیمت در ارتباط مستقیم است.



با استفاده از scatterplot ارتباط دو فیچر با قیمت را نشان میدهیم و مشاهده میکنیم که در حالت اول کلاس ها تا حد خوبی ار یکدیگر قابل تشخیص اند.

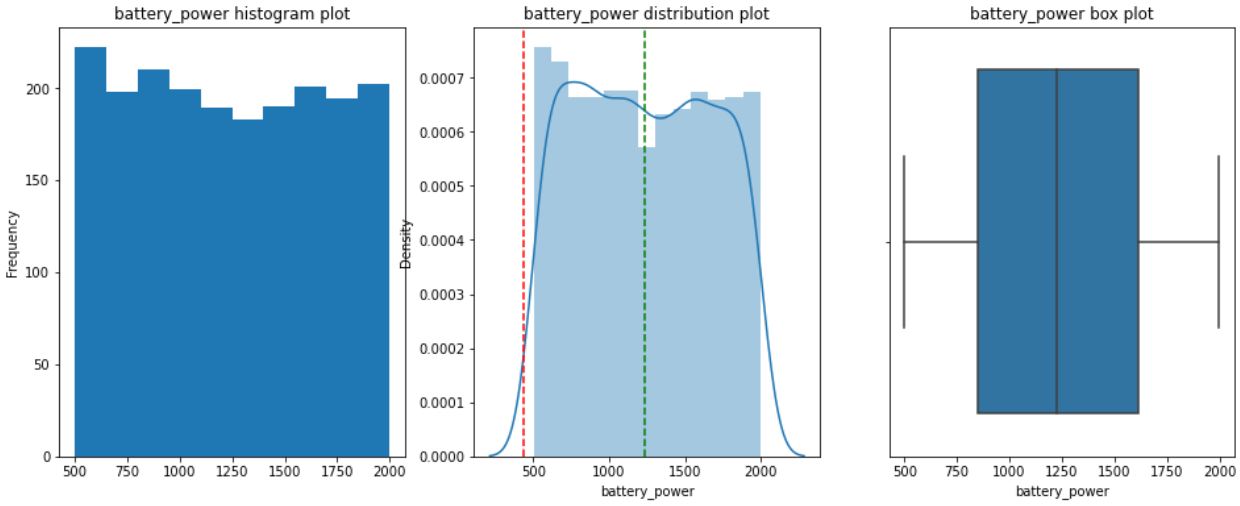


برای هر فیچر سه نمودار (histogram plot, distribution plot, box plot) را رسم میکنیم که با استفاده از این مصور سازی بتوانیم به طور بهتری داده ها را آنالیز کنیم.

```
def feat_plot(feature):
    plt.figure(figsize=(16, 6))
    plt.subplot(1, 3, 1)
    feature.plot(kind = 'hist')
    plt.title(f'{feature.name} histogram plot')

    plt.subplot(1, 3, 2)
    mu, sigma = stats.norm.fit(feature)
    sns.distplot(feature)
```

```
    plt.axvline(mu, linestyle = '--', color = 'green')
    plt.axvline(sigma, linestyle = '--', color = 'red')
    plt.title(f'{feature.name} distribution plot')

    plt.subplot(1, 3, 3)
    sns.boxplot(feature)
    plt.title(f'{feature.name} box plot')
    plt.show()
```

برای نمونه برای battery_power داریم:



Hypothesis test -3

3-1: فرض کردیم که میانگین مقادیر Bluetooth برابر 1 است. از t-test یک نمونه ای استفاده کردیم و چون p value بدست آمده از 0.05 کمتر بود نتیجه گرفتیم که فرض اولیه غلط است.

3-2: فرض کردیم که 3G, 4G به هم وابسته اند. از spearmanr استفاده کردیم و چون p value بدست آمده از 0.05 کمتر بود نتیجه گرفتیم که فرض اولیه درست است.

3-3: فرض کردیم که battery power روی قیمت تاثیر دارد. از pearsonr استفاده کردیم و نتیجه شد که تاثیر دارد.

3-4: فرض کردیم که رم و n_cores وابسته اند. از kendalltau استفاده کردیم و با توجه به p value نتیجه میگیریم که مستقل اند.

3-5: فرض کردیم که wifi, Bluetooth از توضیع یکسانی برخوردارند. با استفاده از ttest_ind و p value بزرگتر از 0.05 نتیجه شد که همینطور است.

Classification -4

از Logistic regression که در scikit learn هست استفاده میکنیم.

داده ها را به نسبت 80 به 20 برای train و test اختصاص میدهیم.

از سه نوع solver(lbfgs, liblinear, newton-cg) استفاده میکنیم و مدل را بر اساس همانها آموزش میدهیم.

Classification report using confusion matrix -5

متوجه میشویم که مدل کلاسیفایر به میزان یکسانی در همه کلاس ها عمل نکرده چون به طروق مختلفی عمل آموزش انجان شده و بسته به روش انجام کار عملکرد ها متفاوت میشود.

```
              precision    recall  f1-score   support

           0       0.88      0.81      0.85       113
           1       0.54      0.55      0.55        94
           2       0.41      0.47      0.44        88
           3       0.71      0.68      0.69       103

    accuracy                           0.64       398
   macro avg       0.64      0.63      0.63       398
weighted avg       0.65      0.64      0.65       398

------------------------------------------------------------
[[92 20  1  0]
 [12 52 25  5]
 [ 0 23 41 24]
 [ 0  1 32 70]]

 Classification report for model 1
```

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       113
           1       0.82      0.71      0.76        94
           2       0.76      0.75      0.75        88
           3       0.93      0.99      0.96       103

    accuracy                           0.87       398
   macro avg       0.86      0.86      0.86       398
weighted avg       0.87      0.87      0.87       398

------------------------------------------------------------
[[112   1   0   0]
 [  7  67  20   0]
 [  0  14  66   8]
 [  0   0   1 102]]

 Classification report for model 2
```

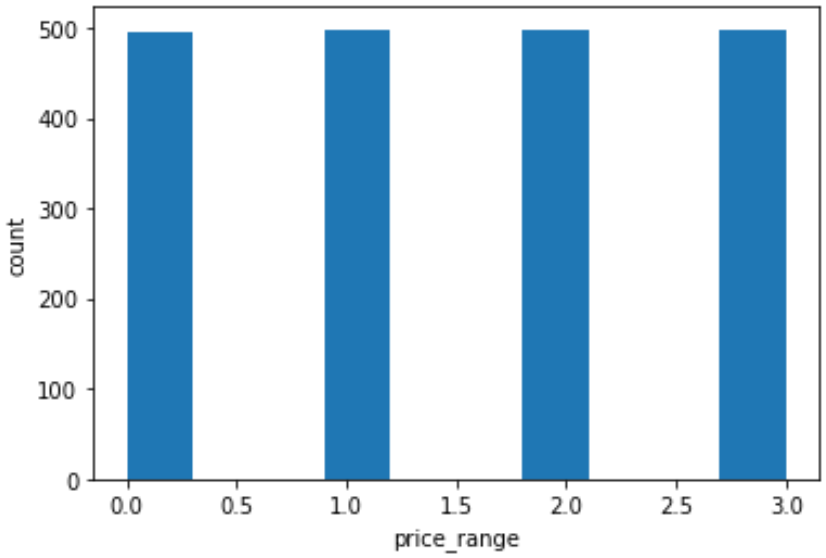
```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       113
           1       0.99      0.94      0.96        94
           2       0.95      0.95      0.95        88
           3       0.96      0.98      0.97       103

    accuracy                           0.97       398
   macro avg       0.97      0.97      0.97       398
weighted avg       0.97      0.97      0.97       398

------------------------------------------------------------
[[112   1   0   0]
 [  4  88   2   0]
 [  0   0  84   4]
 [  0   0   2 101]]

 Classification report for model 3
```

We conclude that the third model is the best

```
model3 = LogisticRegression(solver='newton-cg', penalty='l2',
multi_class='multinomial')
```

Data balancing -6

با توجه به نمودار هیستوگرام قیمت ها داده ها متوازن هستند.



راه حل برای متوازن نبودن داده ها:

a) استفاده ازf1score وrecall به جای accuracy. اینها تا حدی با متوازن نبودن داده ها مقابله میکنند و آن مشکلی که ممکن بود متوازن نبودن داده برای دقت خروجی به وجود بیاورد را ندارند.
b) کپی قسمتی از کلاس های با جمعیت کمتر را به آن کلاس اضافه کنیم و به این شکل جمعیت این کلاس ها را بیشتر کنیم یا قسمتی از دیتا های کلاس با جمعیت بیشتر را کم کنیم و کلاس هایمان را متوازن کنیم.
c) استفاده از الگوریتم Decision tree زیرا به متوازن نبودن داده ها حساس نیست.

Scaling -7

از سه scaling مختلف(min max scaler, standard scaler, quantile transformer) استفاده کردیم و آموزش دادیم در دو حالت دقت کم تر شد و در حالت standard دقت همان اندازه ای شد که قبل از scaling بود (0.97).

Train test split -8

در قسمت 4 این کار به طور کامل انجام شد.

PCA -9

با کاهش فیچر ها نتایج فرق چندانی نمیکند. اما نتیجه خوبی که دارد این است که سرعت محاسبات بالا میرود و اینکه هرچه pov بیشتر باشد دقت بالا میرود اما چون تعداد فیچر ها بیشتر میشود سرعت محاسبه کاهش میابد.

Data imbalance -10

عدم توازن باعث شد که مدل ها بهتر عمل کنند اما واقعا این طور نیست زیرا چون تعداد اعضای یک کلاس به طور قابل توجهی بیشتر از دیگری بود مدل در این کلاس به خوبی عمل میکند و دقت بالایی دارد اما در کلاس دیگر چون هم اندازه قبلی داده آموزشی نداشته خوب عمل نخواهد کرد.

برای متوازن کردن از روش upsampling استفاده شد که کپی قسمتی از کلاس با جمعیت کمتر را به آن کلاس اضافه کنیم و به این شکل جمعیت این کلاس را بیشتر شد.