



DS

*boosting your data
exploration*

MOBILE PRICE CLASSIFICATION


THỰC HIỆN: Nguyễn Duy Đạt

GIẢNG VIÊN: Phạm Đình Khánh

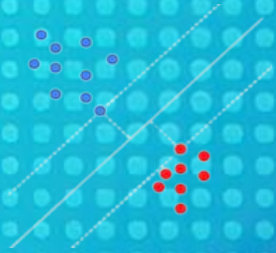
KHÓA HỌC: Machine Learning-Hands on



Chủ nhật, 02/01/2022

- 
1. Import package
 2. Đọc dữ liệu và thống kê mô tả
 3. Visualize distribution
 4. Tiền xử lý dữ liệu
 5. Đánh giá chéo (Cross-validation)
 6. Đánh giá trên nhiều mô hình
 7. GridSearch

1. Import package



SVM



MLP



NumPy



Pandas



Matplotlib

seaborn



Seaborn



RandomForest



GradientBoosting



2. Đọc dữ liệu và thống kê mô tả

- + Đây là bộ dữ liệu về thông tin các tính năng của điện thoại di động khác nhau. (kích thước 2000x21)
- + Bộ dữ liệu gồm 20 biến đầu vào và 1 biến mục tiêu, trong biến đầu vào có 13 biến numeric và 7 biến category

- 1.battery_power: Tổng năng lượng mà pin có thể lưu trữ trong một giờ được tính bằng mAh (numeric)
- 2.blue: Có bluetooth hay không (có = 1, không = 0) (category)
- 3.clock_speed: tốc độ mà bộ vi xử lý thực hiện các lệnh (numeric)
- 4.dual_sim: Có hỗ trợ sim kép hay không (có = 1, không = 0) (category)
- 5.fc: số mega pixels camera trước (numeric)
- 6.four_g: Có mạng 4G (có = 1, không = 0) (category)
- 7.int_memory: dung lượng bộ nhớ trong Gigabytes (numeric)
- 8.m_dep: chiều sâu của điện thoại tính theo cm (numeric)
- 9.mobile_wt: trọng lượng (numeric)
- 10.n_cores: Số lõi của bộ xử lý (category)
- 11.pc: số mega pixels camera chính (numeric)
- 12.px_height: Chiều cao độ phân giải pixel (numeric)
- 13.px_width: Chiều rộng độ phân giải pixel (numeric)
- 14.ram: Bộ nhớ truy cập ngẫu nhiên tính bằng Mega Byte (numeric)
- 15.sc_h: Chiều cao màn hình của điện thoại di động tính bằng cm (numeric)
- 16.sc_w: Chiều rộng màn hình của điện thoại di động tính bằng cm (numeric)
- 17.talk_time: thời gian dài nhất mà một lần sạc pin sẽ kéo dài (numeric)
- 18.three_g: Có mạng 3G (có = 1, không = 0) (category)
- 19.touch_screen: Có màn hình cảm ứng hay không (có = 1, không = 0) (category)
- 20.wifi: Có bắt wifi được không (có = 1, không = 0) (category)
- 21.price_range: Đây là biến mục tiêu có giá trị 0 (chi phí thấp), 1 (chi phí trung bình), 2 (chi phí cao) và 3 (chi phí rất cao).

2. Đọc dữ liệu và thống kê mô tả

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000
xx_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
00.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
45.108000	1251.515500	2124.213000	12.306500	5.767000	11.011000	0.761500	0.503000	0.507000	1.500000
43.780811	432.199447	1084.732044	4.213245	4.356398	5.463955	0.426273	0.500116	0.500076	1.118314
0.000000	500.000000	256.000000	5.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
82.750000	874.750000	1207.500000	9.000000	2.000000	6.000000	1.000000	0.000000	0.000000	0.750000
64.000000	1247.000000	2146.500000	12.000000	5.000000	11.000000	1.000000	1.000000	1.000000	1.500000
47.250000	1633.000000	3064.500000	16.000000	9.000000	16.000000	1.000000	1.000000	1.000000	2.250000
60.000000	1998.000000	3998.000000	19.000000	18.000000	20.000000	1.000000	1.000000	1.000000	3.000000

- Mean thấp nhất là 0.495, cao nhất là 2124.2
- Cần chuẩn hóa biến trước khi xây dựng mô hình

2. Đọc dữ liệu và thống kê mô tả

```
<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
```

Không có missing value

3. Visualize distribution

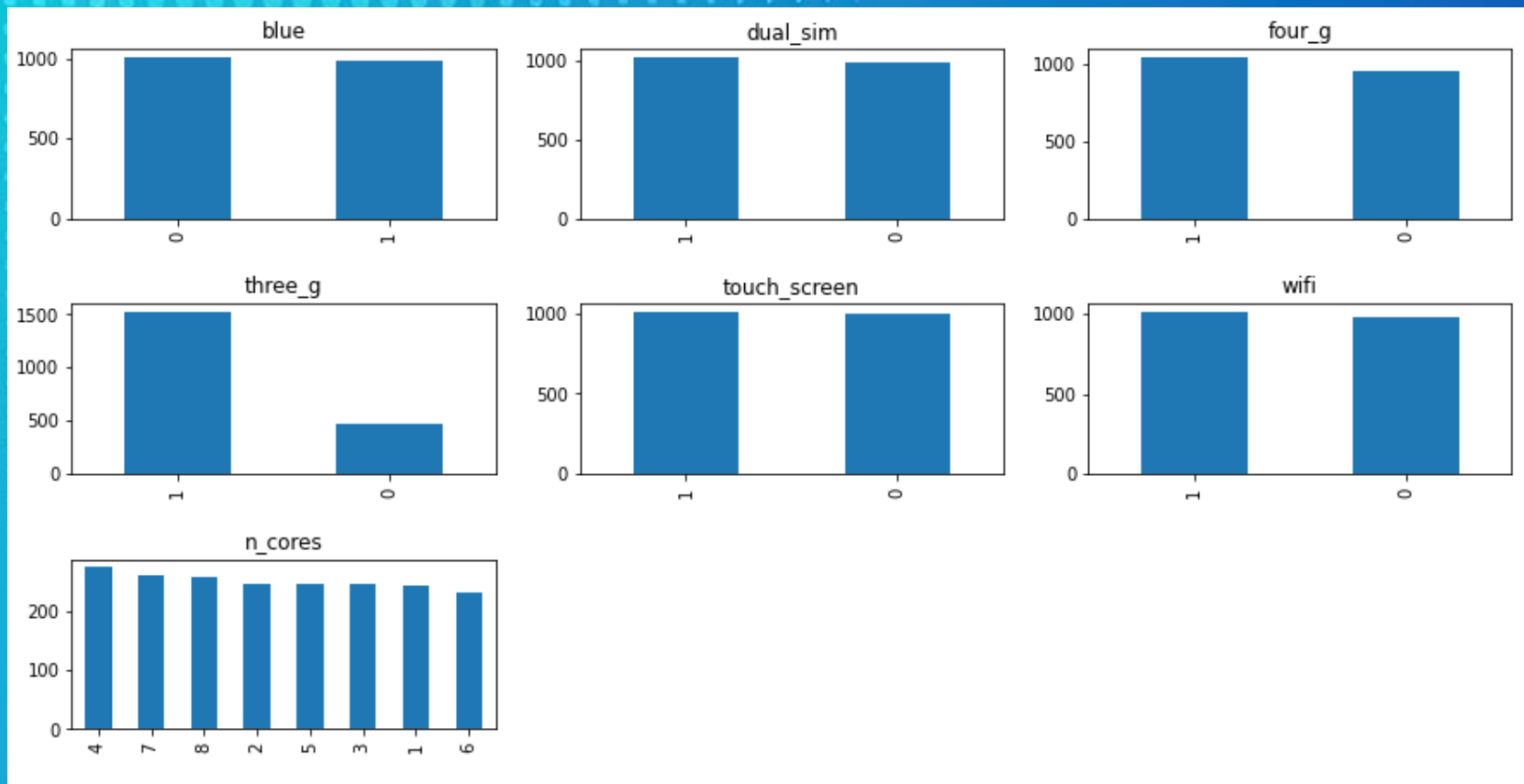
**Biến liên tục
(numeric)**



Biến fc, px_height, sc_w
đều giảm dần !

3. Visualize distribution

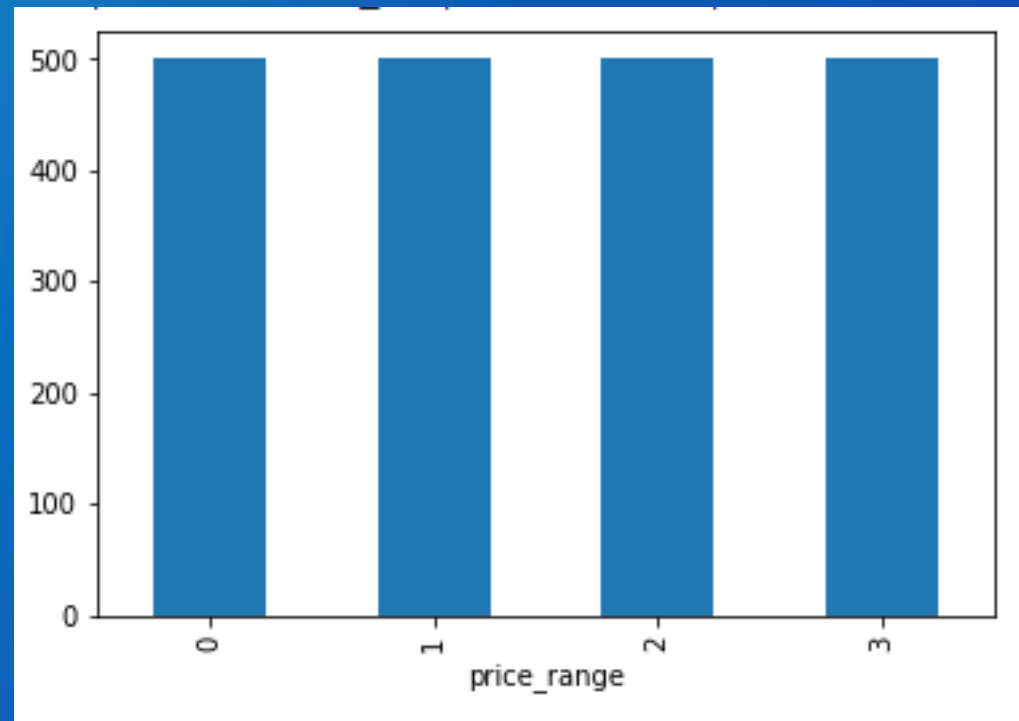
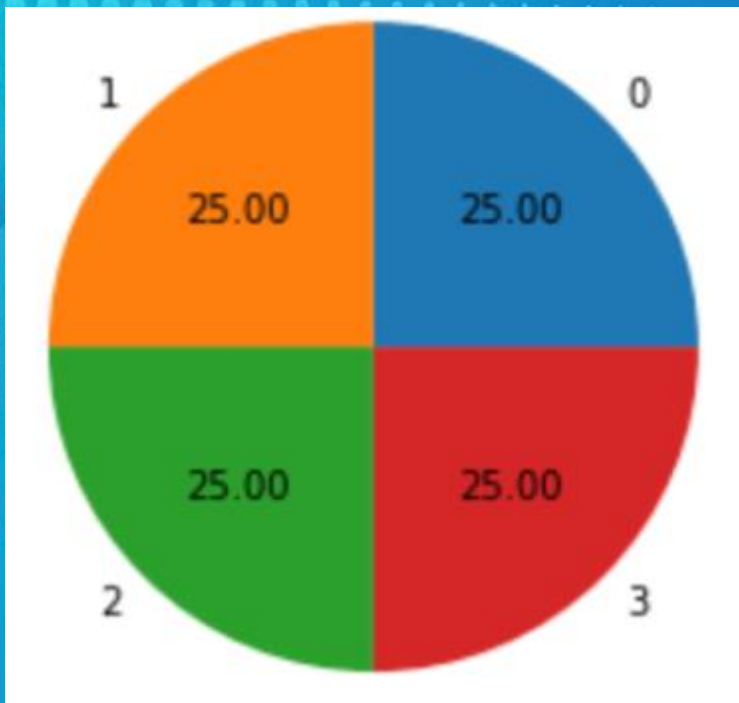
Biến phân loại (category)



-Hầu như các biến đều cân bằng ngoại lệ biến `three_g` có sự chênh lệch lớn !

3. Visualize distribution

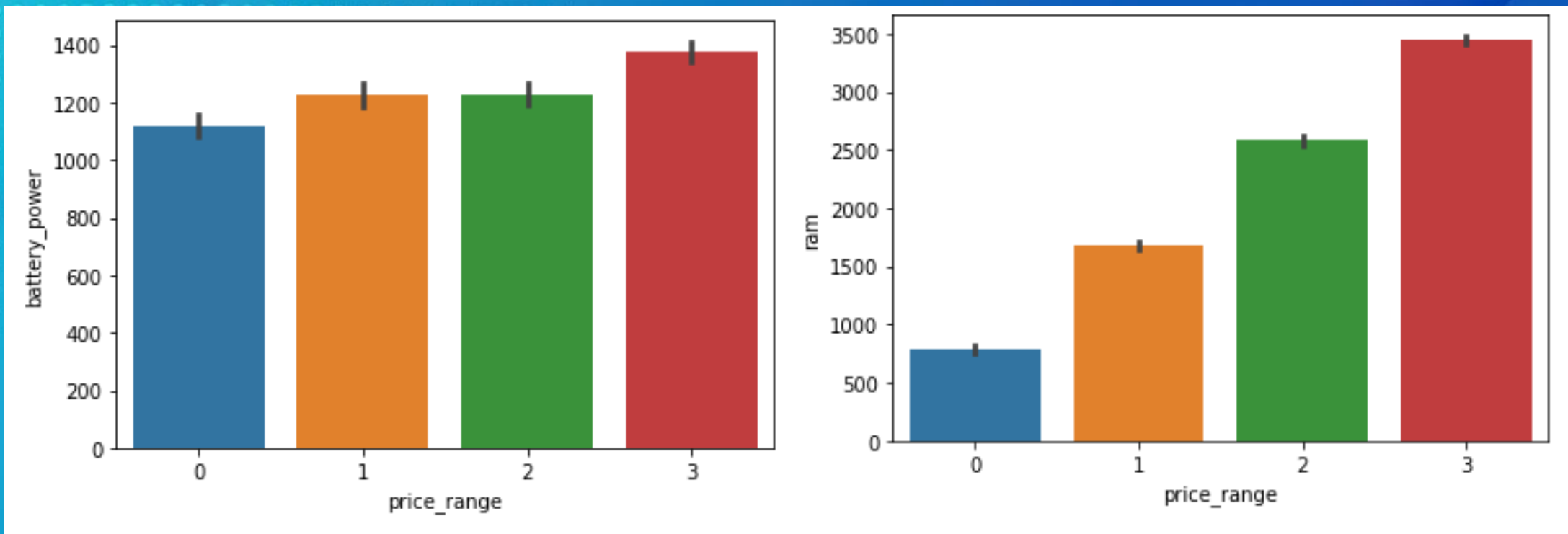
Biến mục tiêu (price_range)



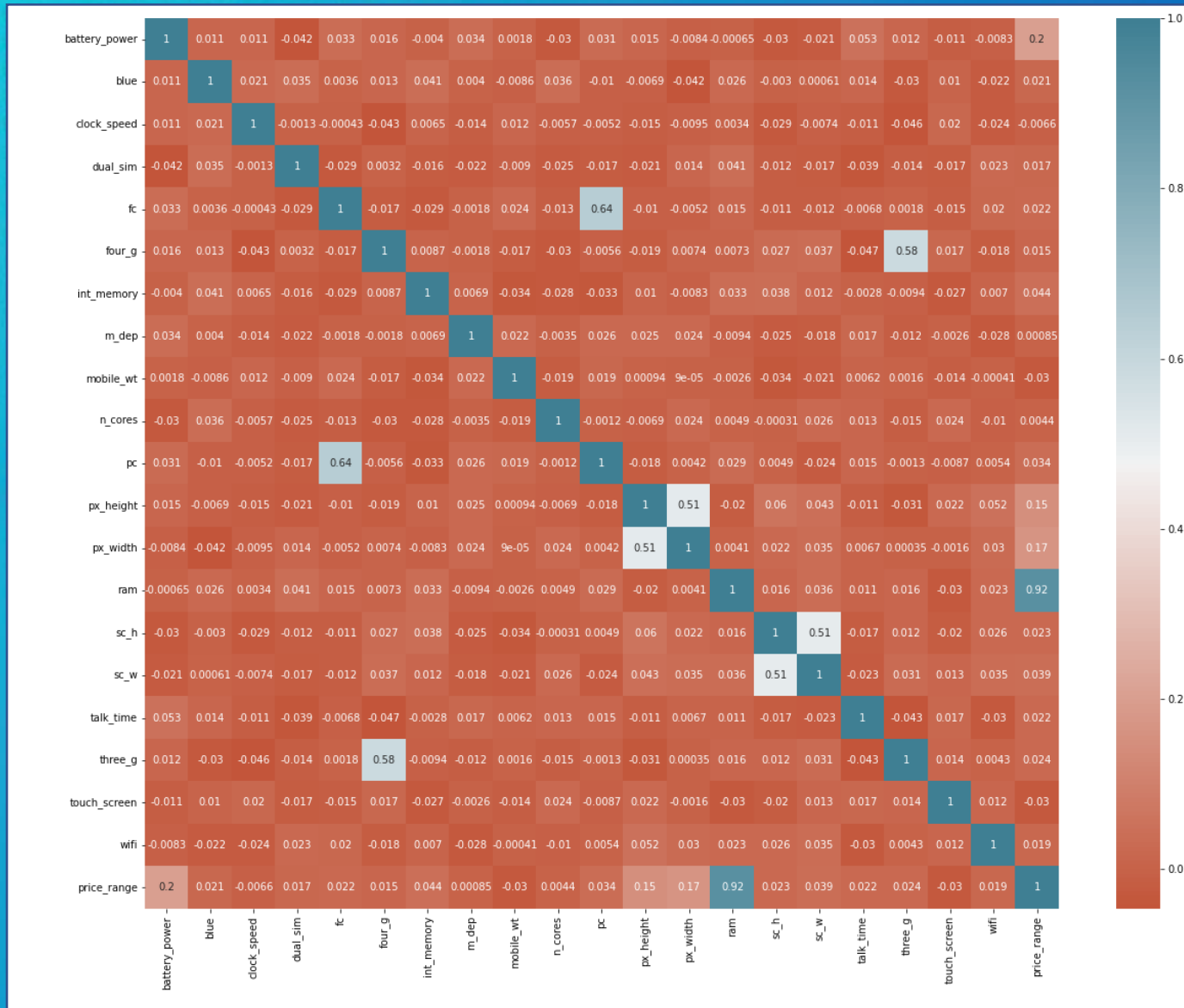
-Bộ dữ liệu cân bằng giữa 4 nhãn 0,1,2,3 của biến mục tiêu nên ta có thể dùng thang đo Accuracy để đánh giá.

3. Visualize distribution

Visualize 1 vài phân phối khác:



3. Visualize distribution



-Biến ram tương quan rất cao với biến mục tiêu price_range (0.92)

-fc tương quan với pc : 0.64

-three_g và four_g, px_height và px_width lần lượt tương quan là 0.58, 0.51

4. Tiền xử lí dữ liệu

```
df_train, df_val = train_test_split(train, test_size=0.2, stratify = train['price_range'])
X_train = df_train.copy()
y_train = X_train.pop("price_range")

X_val = df_val.copy()
y_val = X_val.pop("price_range")
print(X_train.shape, y_train.shape)
print(X_val.shape, y_val.shape)

(1600, 20) (1600,)
(400, 20) (400,)
```

- Tạo pipeline với mô hình RandomForestClassifier và chuẩn hóa MinMaxScaler
- >Accuracy trên tập train: 1.00
- >Accuracy trên tập test: 0.88

=> Có hiện tượng Overfitting

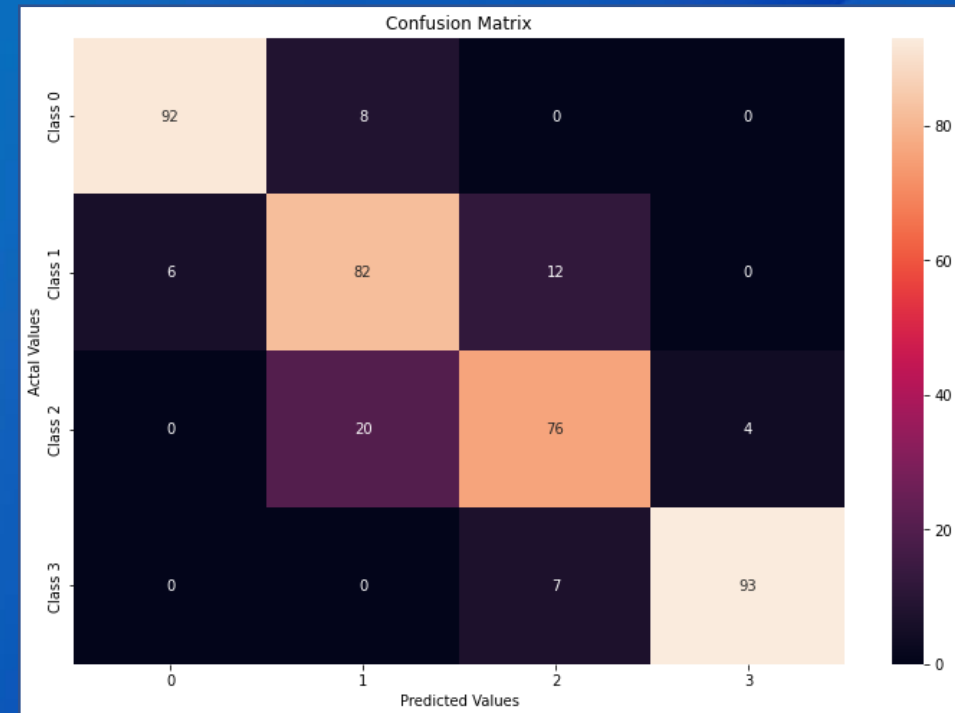
5. Đánh giá chéo (Cross-validation)

-> Mean Accuracy: 0.872

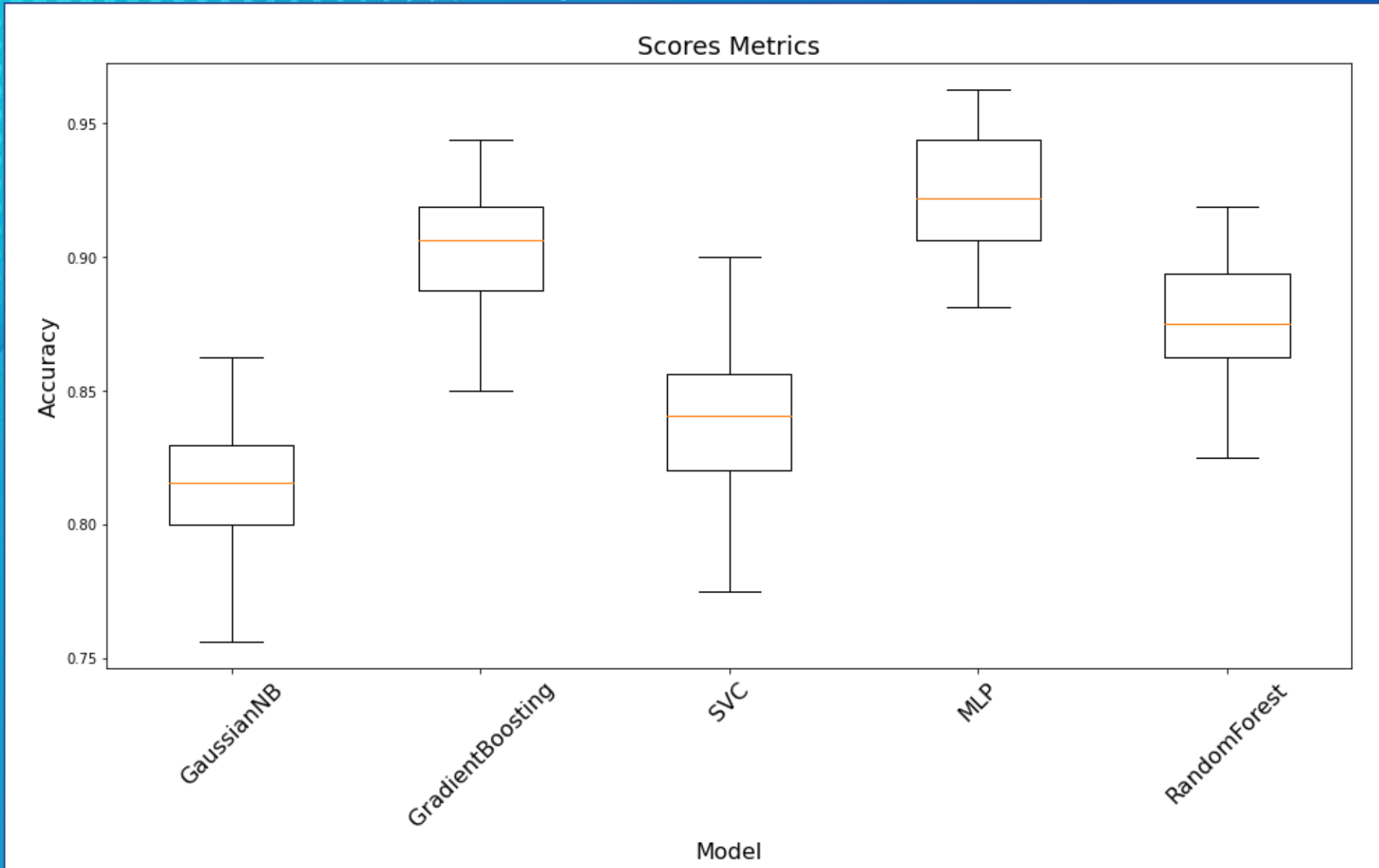
-> Std: 0.026

=> Có thể chấp nhận được độ lệch này

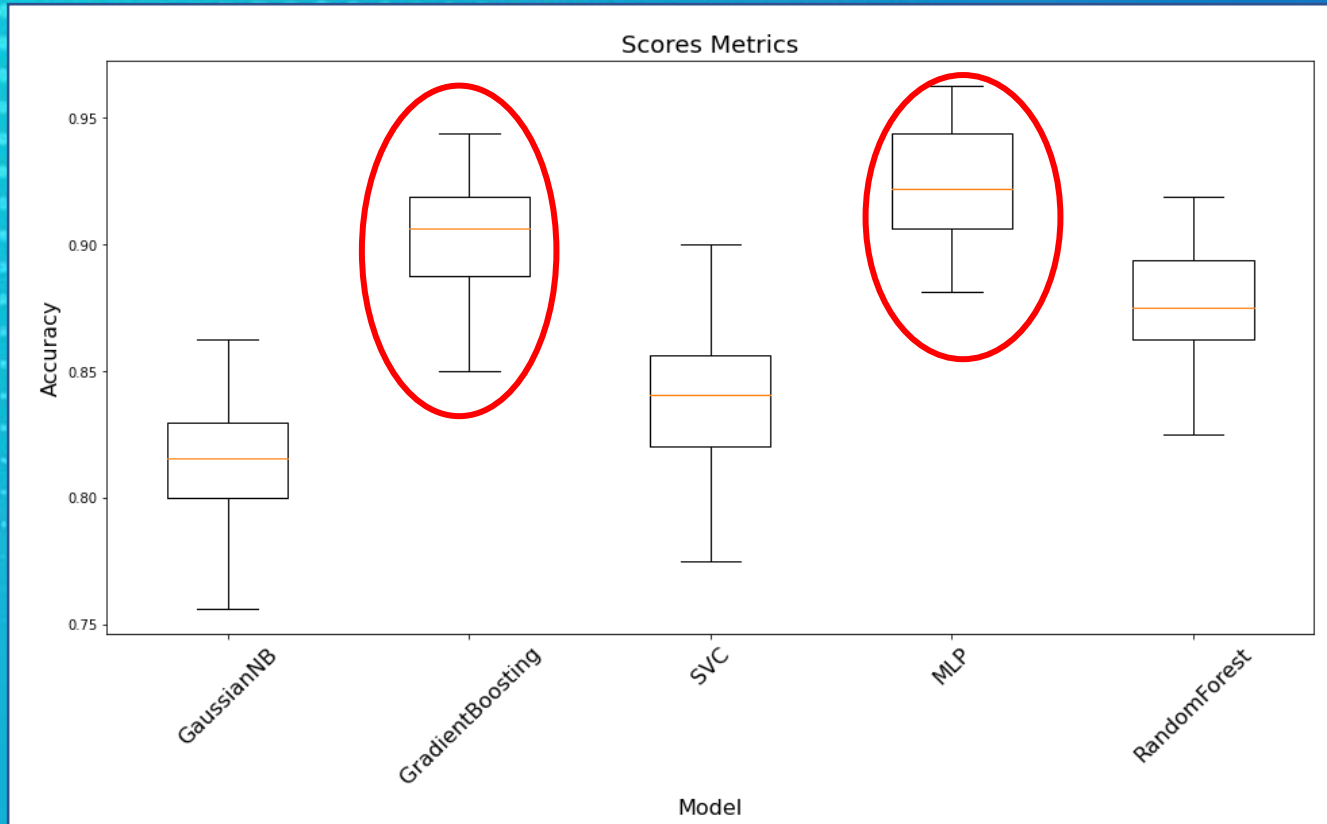
	precision	recall	f1-score	support
class 0	0.92	0.95	0.94	100
class 1	0.82	0.82	0.82	100
class 2	0.83	0.83	0.83	100
class 3	0.96	0.93	0.94	100
accuracy			0.88	400
macro avg	0.88	0.88	0.88	400
weighted avg	0.88	0.88	0.88	400



6. Đánh giá trên nhiều mô hình



7.GridSearch



- Thực hiện GridSearch:
+ MLP: 0.65
+ GradientBoosting: 0.9043

- Thực hiện GridSearch kết hợp
RandomForest:

+ MLP: 0.91875

+ SVC: 0.93499