

classification

m o B i l E

price

laew tae ja kidz



About Dataset

Bob ได้ก่อตั้งบริษัทโทรศัพท์มือถือของตัวเอง และต้องการแบ่งข้อมูลกับบริษัทใหญ่ๆ เช่น Apple, Samsung ซึ่งเขาไม่รู้ว่าจะประเมินราคากลางโทรศัพท์มือถือที่บริษัทของเขายังไง ในตลาดโทรศัพท์มือถือที่มีการแบ่งข้อมูลสูงนี้ เขายังสามารถที่จะสรุปสิ่งต่างๆ ได้เพื่อแก้ปัญหา เขายังรวมข้อมูลการขายโทรศัพท์มือถือของบริษัทต่างๆ ไว้ ซึ่ง Bob ต้องการค้นหาความสัมพันธ์บางอย่างระหว่างคุณสมบัติของโทรศัพท์มือถือ (เช่น RAM, Internal Memory ฯลฯ) กับช่วงราคาที่จะขาย



Data Explorer

train.csv

2000 rows
21 columns

test.csv

1000 rows
21 columns

column metadata

X

L O V E
Y

# battery_power	# blue	# clock_speed	# dual_sim	# fc
Total energy a battery can store in one time measured in mAh	Has bluetooth or not	speed at which microprocessor executes instructions	Has dual sim support or not	Front Camera mega pixels
# four_g	# int_memory	# m_dep	# mobile_wt	# n_cores
Has 4G or not	Internal Memory in Gigabytes	Mobile Depth in cm	Weight of mobile phone	Number of cores of processor
# pc	# px_height	# px_width	# ram	# sc_h
Primary Camera mega pixels	Pixel Resolution Height	Pixel Resolution Width	Random Access Memory in Mega Bytes	Screen Height of mobile in cm
# sc_w	# talk_time	# three_g	# touch_screen	# wifi
Screen Width of mobile in cm	longest time that a single battery charge will last when you are	Has 3G or not	Has touch screen or not	Has wifi or not

price_range

This is the target variable with value of 0(low cost), 1(medium cost), 2(high cost) and 3(very high cost).

for test set

id
ID



Data preparation

```
1 train[['m_dep','px_height','sc_w']].describe()
```

	m_dep	px_height	sc_w
count	2000.000000	2000.000000	2000.000000
mean	0.501750	645.108000	5.767000
std	0.288416	443.780811	4.356398
min	0.100000	0.000000	0.000000
25%	0.200000	282.750000	2.000000
50%	0.500000	564.000000	5.000000
75%	0.800000	947.250000	9.000000
max	1.000000	1960.000000	18.000000

```
1 train[['m_dep','px_height','sc_w']].describe()
```

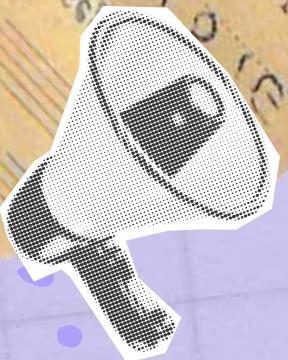
	m_dep	px_height	sc_w
count	2000.000000	2000.000000	2000.000000
mean	0.626000	646.529000	6.199420
std	0.165159	441.845917	3.891793
min	0.500000	65.000000	2.540000
25%	0.500000	282.750000	2.540000
50%	0.500000	564.000000	5.000000
75%	0.800000	947.250000	9.000000
max	1.000000	1960.000000	18.000000

ปรับค่า min ใหม่

ไฟล์ train มี features m_dep, px_height และ sc_w
ที่มีค่า min ไม่สมเหตุสมผล

กำหนดขั้นต่ำ
ความลึก (m_dep) คือ 0.5 เซนติเมตร
ความสูงของความลักษณ์พิกเซล (px_height) คือ 65 พิกเซล
ความกว้างหน้าจอ (sc_w) คือ 2.54 เซนติเมตร

check missing



1 train.isnull().sum()

```
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  
dtype: int64
```

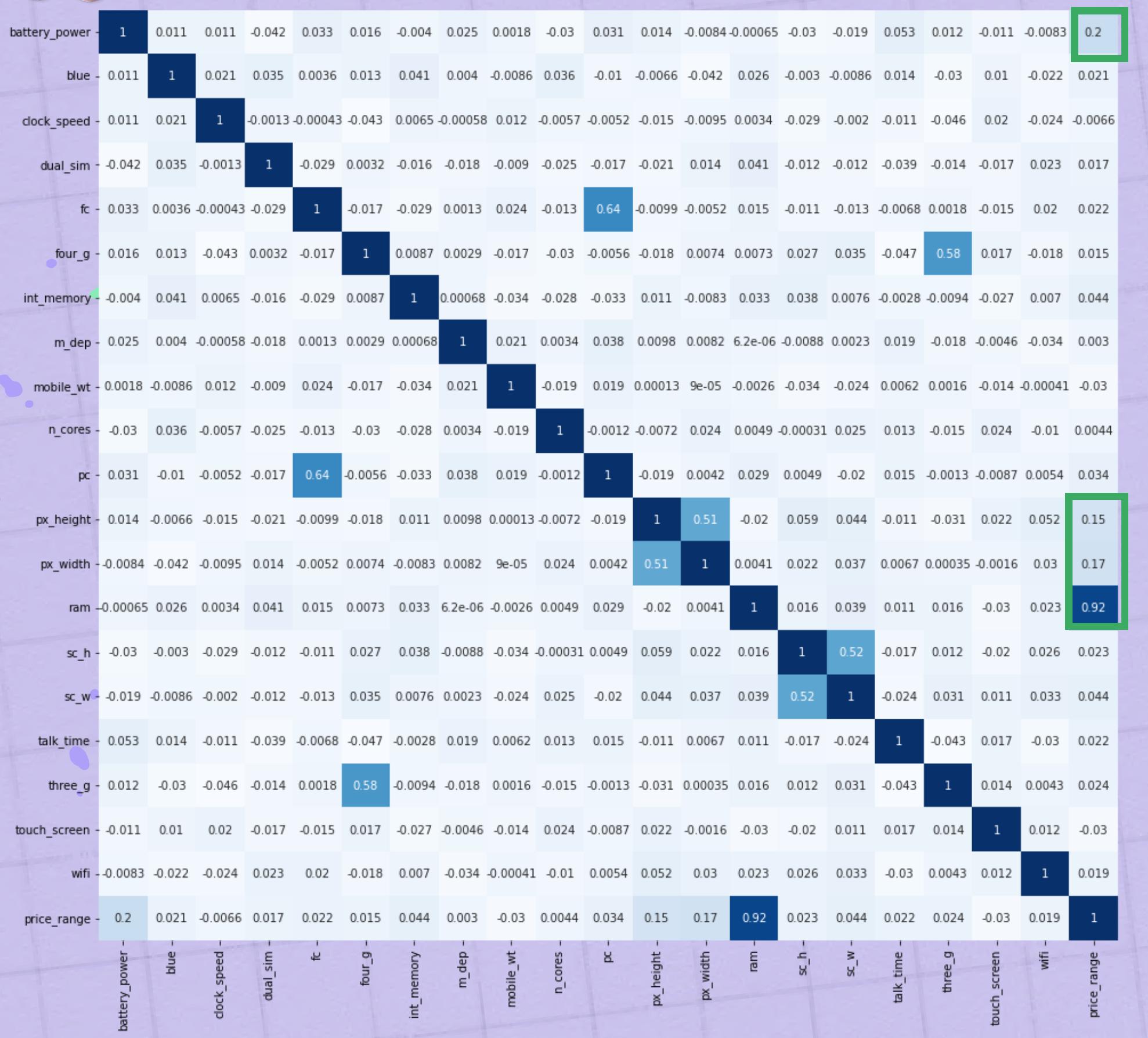
1 test.isnull().sum()

```
id                0  
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  
dtype: int64
```

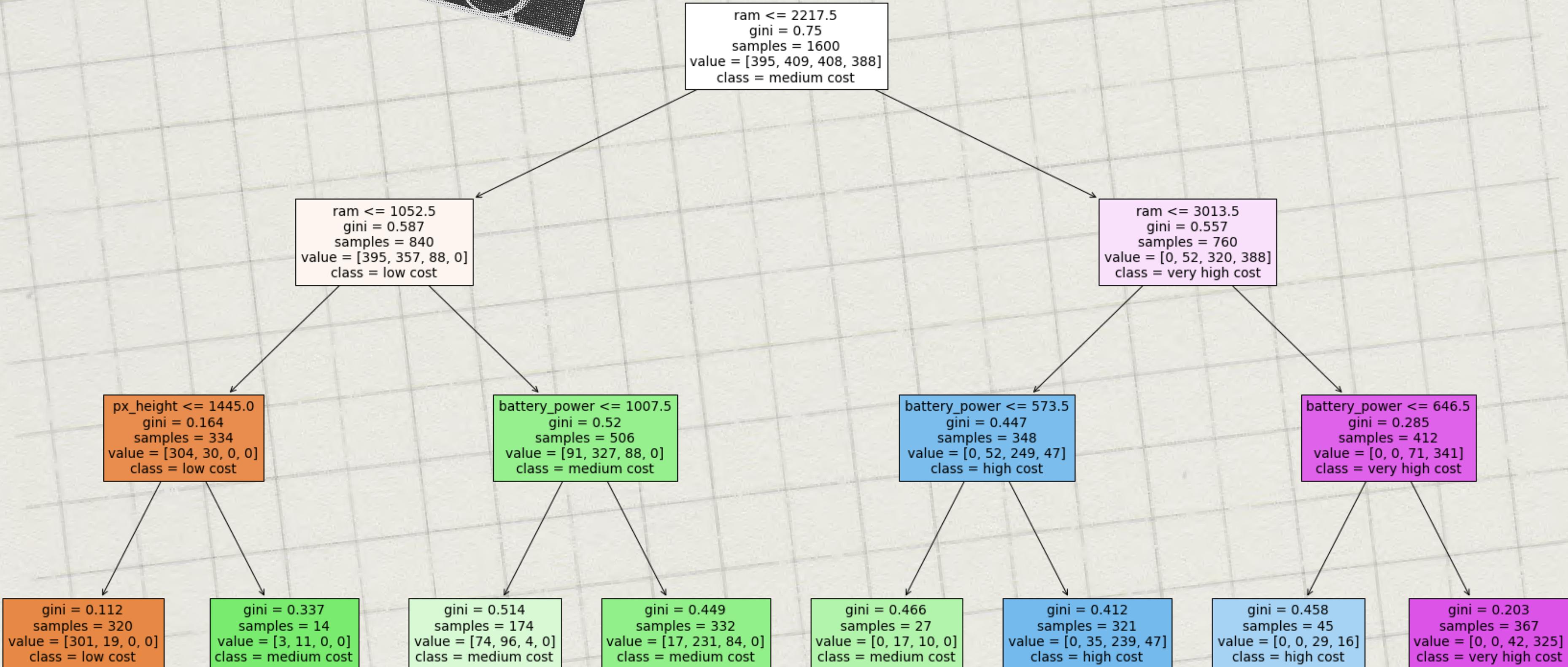
yleS

การวิเคราะห์ความสัมพันธ์

จาก heatmap จะพบว่า หน่วยความจำ, แบตเตอรี่, ความกว้างของความล廓เอียดของภาพ และ ความสูงของความล廓เอียดของภาพ มีความสัมพันธ์กับช่วงราคา



Decision Tree



Decision Tree

KNN

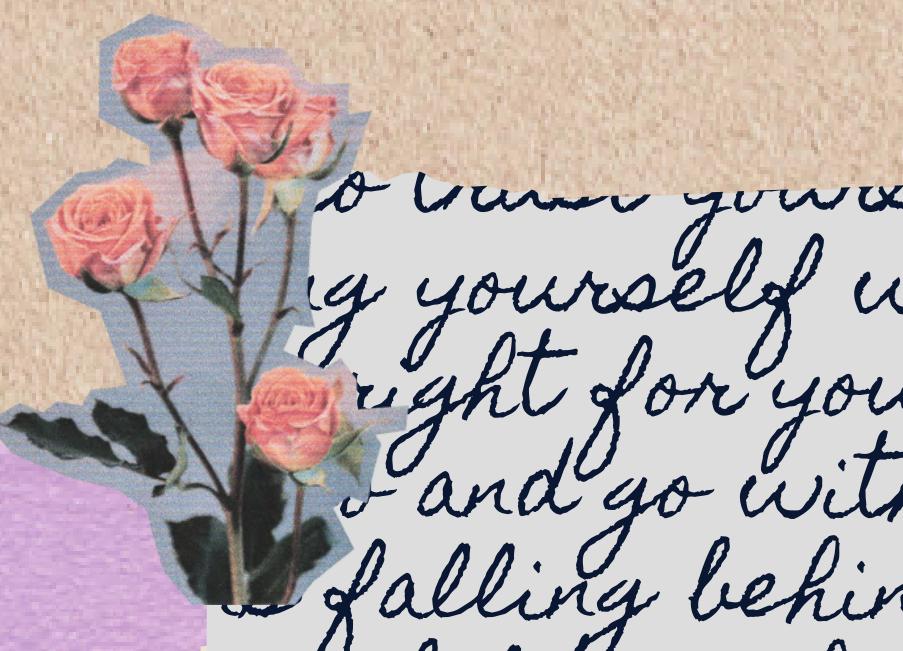
Naive Bayes

Decision Tree accuracy score: 0.7650

KNN accuracy score: 0.9275

Naive Bayes accuracy score: 0.8150

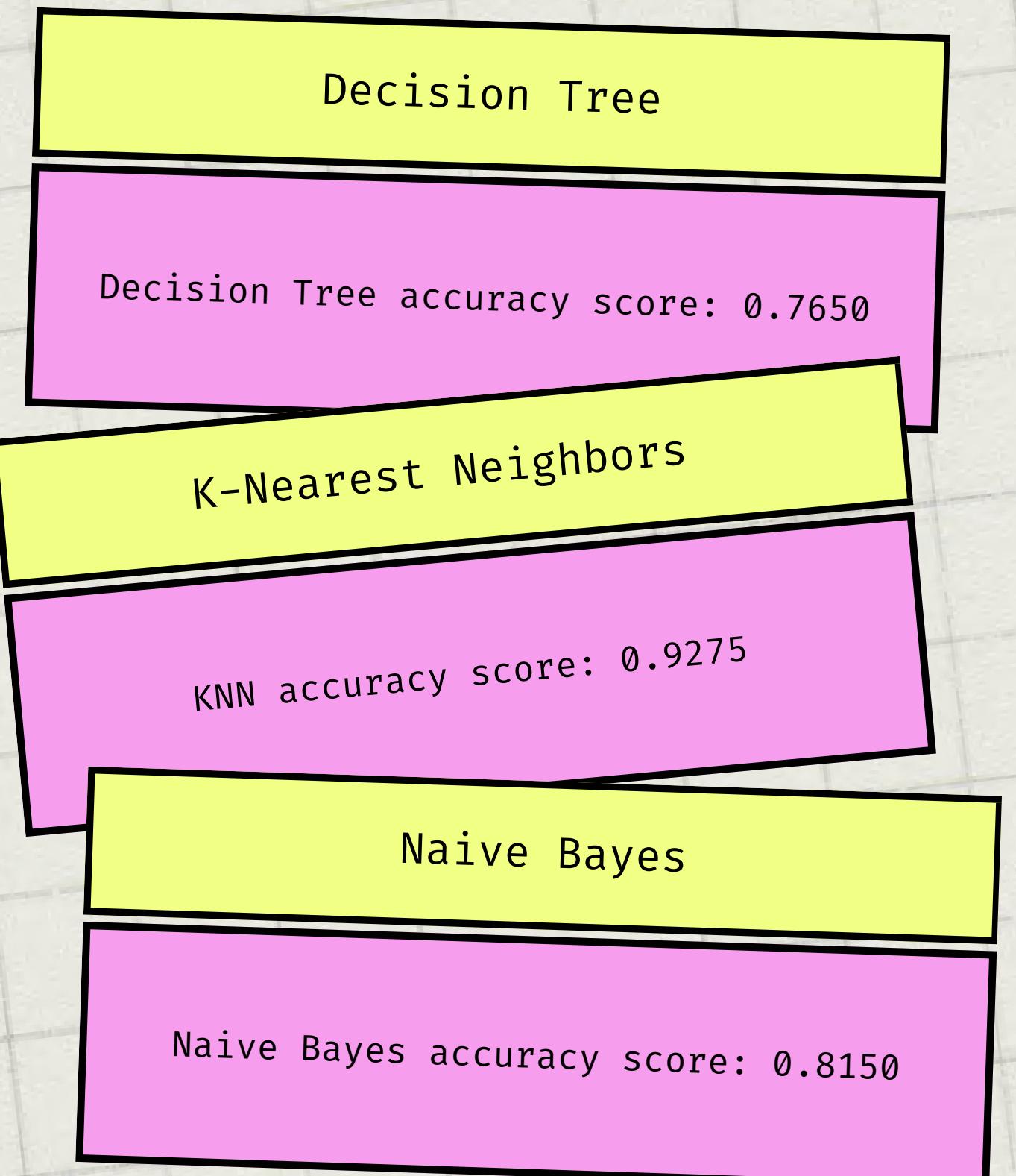
จากการเปรียบเทียบค่า accuracy score Model KNN
สามารถคาดการณ์ได้แม่นยำที่สุด เพราะมีค่า accuracy score สูงที่สุด



Classification

การคาดการณ์ช่วงราคาของโทรศัพท์มือถือโดยการสร้างแบบจำลองที่คำนึงถึงคุณลักษณะต่างๆ ที่มีอยู่ในชุดข้อมูล เราจะใช้วิธี Decision Tree, K-Nearest Neighbors และ Naive Bayes

จากการเปรียบเทียบค่า accuracy score จะพบว่าวิธี K-Nearest Neighbors แม่นยำที่สุด



Predicting Values for test.csv

จากที่ KNN ได้ accuracy score สูงที่สุด จึงนำ model KNN มาคาดการณ์ช่วงราคาสำหรับโทรศัพท์มือถือเครื่องใหม่ในไฟล์ test ได้ดังตารางนี้

test

	battery_power	px_height	px_width	ram	predicted_price
0	1043	226	1412	3476	very high cost
1	841	746	857	3895	very high cost
2	1807	1270	1366	2396	very high cost
3	1546	295	1752	3893	very high cost
4	1434	749	810	1773	medium cost
...
995	1700	644	913	2121	high cost
996	609	1152	1632	1933	medium cost
997	1185	477	825	1223	low cost
998	1533	65	832	2509	high cost
999	1270	457	608	2828	high cost

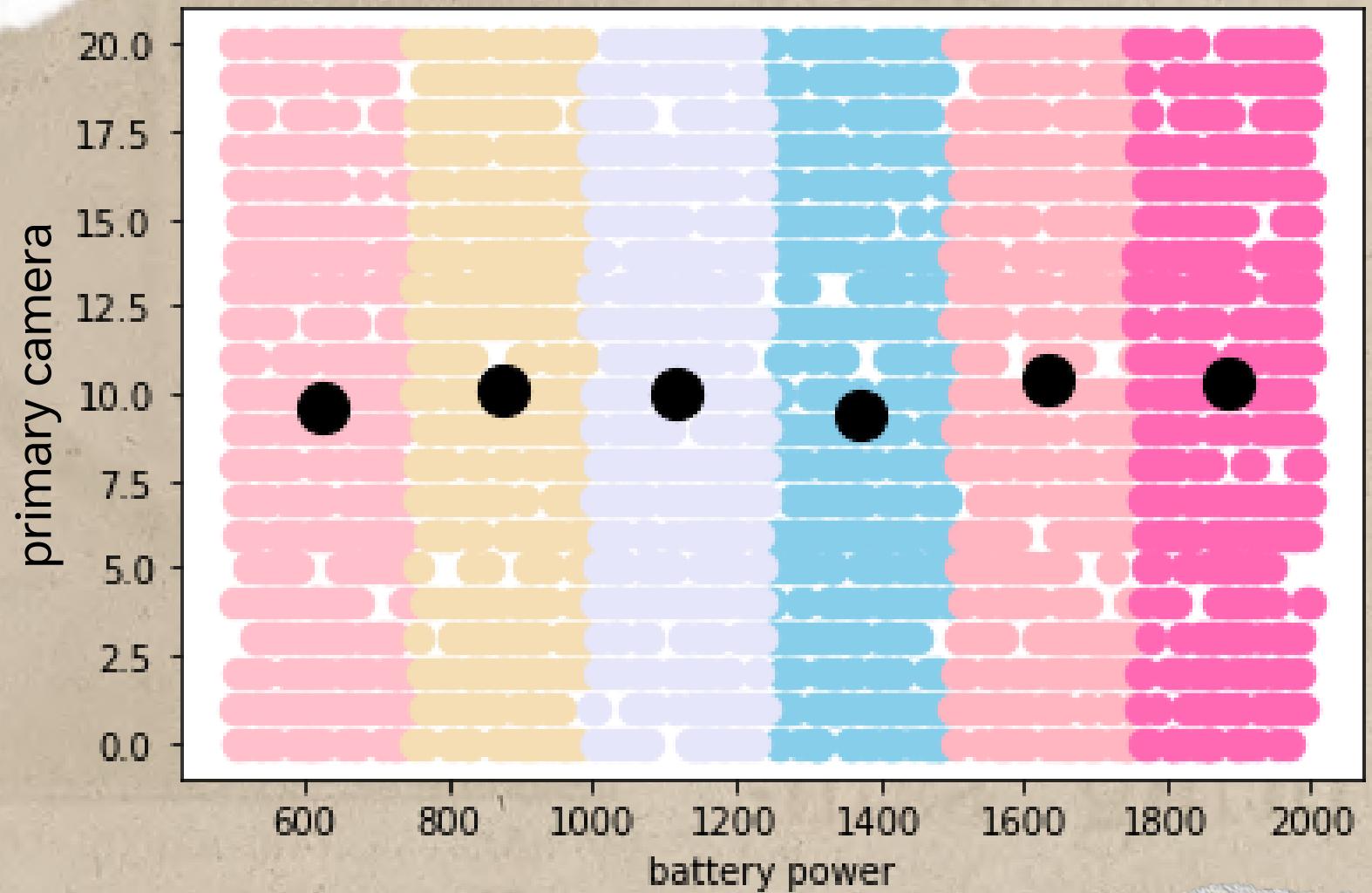
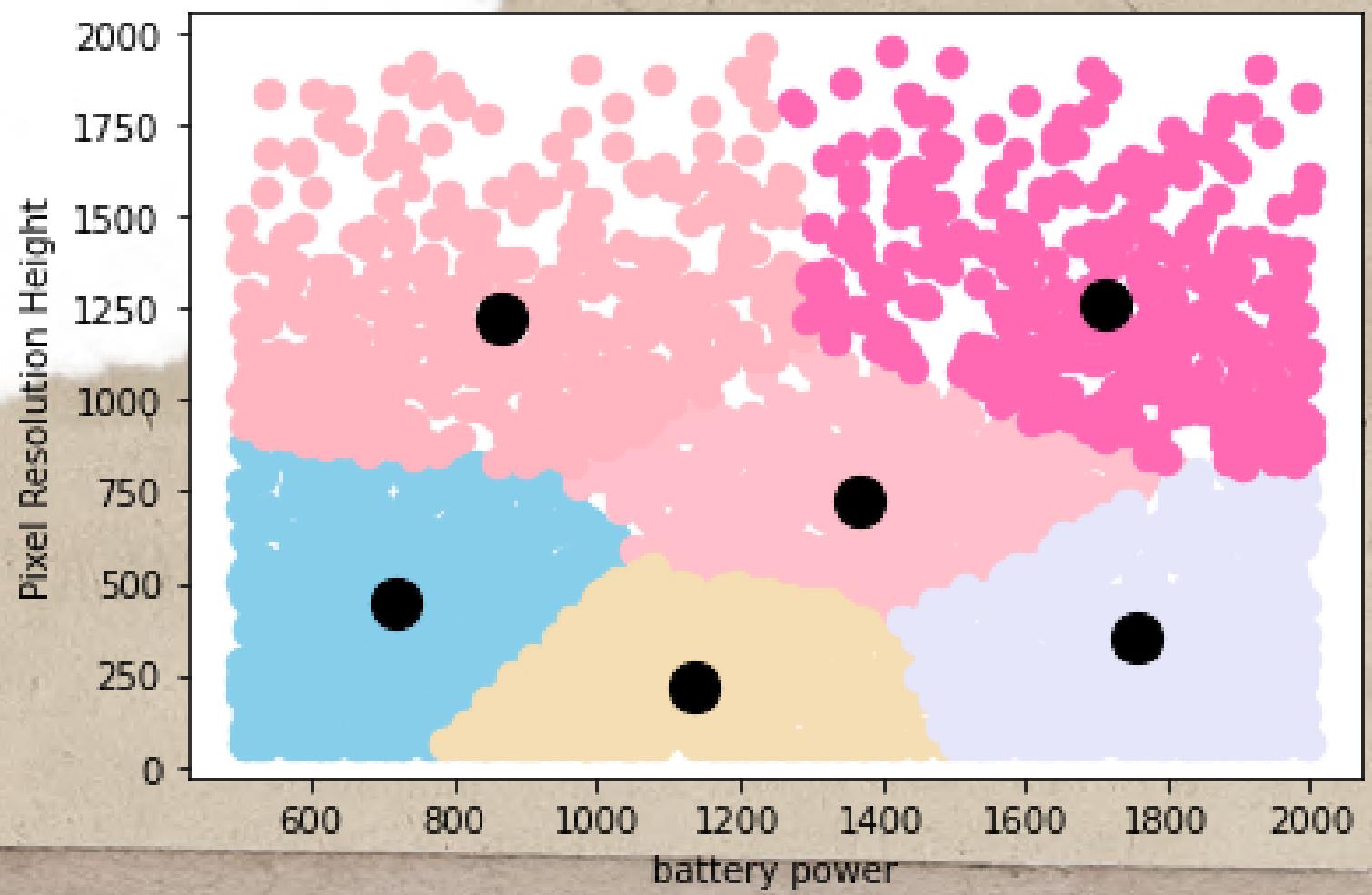
1000 rows × 5 columns

```
[564] test['predicted_price'].value_counts()  
very high cost    270  
low cost          262  
high cost         247  
medium cost       221  
Name: predicted_price, dtype: int64
```

K-nearest neighbor

K - means clustering

$k=6$



Association rule

Association rule มักจะใช้หารูปแบบในผลิตภัณฑ์ที่ลูกค้า
มักจะซื้อร่วมกัน อาจใช้ไม่ได้โดยตรงกับปัญหาการจัดประเภทราคา
มือถือ เราเลยนำ Association rule มาสำรวจความ
สัมพันธ์ระหว่างคุณลักษณะต่างๆ แทน

LOVe

Association rule

```
1 rules.sort_values('lift', ascending=False).head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
107	(dual_sim_1, three_g_1, blue_1)	(four_g_1)	0.1890	0.5215	0.1365	0.722222	1.384894	0.037937	1.722600
114	(four_g_1)	(dual_sim_1, three_g_1, blue_1)	0.5215	0.1890	0.1365	0.261745	1.384894	0.037937	1.098536
108	(four_g_1, dual_sim_1)	(three_g_1, blue_1)	0.2665	0.3705	0.1365	0.512195	1.382443	0.037762	1.290475
113	(three_g_1, blue_1)	(four_g_1, dual_sim_1)	0.3705	0.2665	0.1365	0.368421	1.382443	0.037762	1.161375
171	(four_g_1, wifi_1)	(dual_sim_1, three_g_1)	0.2600	0.3850	0.1370	0.526923	1.368631	0.036900	1.300000
172	(dual_sim_1, three_g_1)	(four_g_1, wifi_1)	0.3850	0.2600	0.1370	0.355844	1.368631	0.036900	1.148790
130	(touch_screen_1, three_g_1, blue_1)	(four_g_1)	0.1885	0.5215	0.1345	0.713528	1.368222	0.036197	1.670319
139	(four_g_1)	(touch_screen_1, three_g_1, blue_1)	0.5215	0.1885	0.1345	0.257910	1.368222	0.036197	1.093533
150	(three_g_1, blue_1)	(four_g_1, wifi_1)	0.3705	0.2600	0.1315	0.354926	1.365099	0.035170	1.147155
145	(four_g_1, wifi_1)	(three_g_1, blue_1)	0.2600	0.3705	0.1315	0.505769	1.365099	0.035170	1.273696

ตัวอย่าง ถ้าในโทรศัพท์มือมีฟีเจอร์ รองรับ dual_sim, มี 3G และ มีบลูทูธ
จะมีฟีเจอร์ 4G ด้วย

TREAT
Yourself

Association rule

```
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

# Load the data
train_data = pd.read_csv('/content/drive/MyDrive/bsc_DPDM_data/Project/train.csv')

# Preprocess the data
X = train_data.drop('price_range', axis=1)
X = pd.get_dummies(X, columns=['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', 't

# Convert the data to a binary matrix
X = (X > 0).astype(int)

# Use Apriori to find frequent itemsets
frequent_itemsets = apriori(X, min_support=0.1, use_colnames=True)

# Use Association Rules to find interesting associations
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Print the top 10 rules sorted by lift
print(rules.sort_values('lift', ascending=False).head(10))
```

ตัวอย่าง ถ้าในโทรศัพท์มือมีฟีเจอร์ รองรับ dual_sim,
มี 3G และ มีบลูทูธ จะมีฟีเจอร์ 4G ด้วย



Reference

<https://www.kaggle.com/datasets/iabhishekofficial/mobile-price>

mEMBERS



Sucheara Nakkhum

633020449-1



Jirapat Thongprasert

633020547-1



Kanyarat Boonkong

633020548-9



Chutima Khamkhubon

633021014-1



THANK YOU