

IT326: Data Mining

Mobile Price

1-

Student Name	ID
Areen Aljarbou	444200820
Njoud Alduraib	444200849
Alhanouf Alkhudhayri	444200815
Maha alswed	443200734

Problem:

The problem at hand involves predicting or analyzing data pricing. In today's increasingly digital world, pricing data is crucial for businesses to make informed decisions related to pricing strategies, customer segmentation, and market trends. Accurate pricing data helps businesses optimize their product pricing, maximize

revenue, and stay competitive. However, determining the optimal price for data or services can be complex, as it depends on various factors such as market demand, competition, and product attributes.

This problem is important because, without reliable pricing insights, businesses risk underpricing (losing potential revenue) or overpricing (losing customers). By understanding the factors that influence data pricing, businesses can more effectively set prices that reflect market conditions and maximize profitability.

2-Data Mining Task:

The data mining task can be formulated as a regression task if we are predicting the exact price of data or services based on various input features (such as demand, product characteristics, and market factors). Alternatively, it could also be approached as a classification task if we categorize data prices into discrete ranges (e.g., low, medium, high pricing tiers).

- If it's a regression task, the goal would be to predict the exact price (continuous variable) based on features from the dataset.
- If it's a classification task, the goal would be to classify the price into predefined categories or classes, such as "cheap," "moderate," and "expensive."

In either case, the class attribute (for classification) or the target variable (for regression) would be the price of the data. The main goal is to build a predictive model that can accurately forecast the price of data based on relevant factors in the dataset.

3-Data:

the source of dataset is kaggle :

<https://www.kaggle.com/datasets/iabhishekoofficial/mobile-price-classification/data?select=train.csv>

Information about the dataset:

Number of attributes:21

Attribute types: Numeric, Binary

Number of objects:2000

Class label: price_range

Attributes' description

Attribute	Description	Data Type	Possible Values
battery_power	The capacity of the phone's battery, typically measured in milliampere-hours (mAh).	Numeric	Positive integer values (mAh), e.g., 1000, 3000, 4000.
blue	Indicates whether the phone has Bluetooth connectivity (1 for yes, 0 for no).	Binary	0 (No), 1 (Yes).
clock_speed	The speed of the phone's processor, measured in GHz.	Numeric	Floating-point numbers (GHz), e.g., 1.2, 2.5, 3.0.
dual_sim	Indicates if the phone supports two SIM cards (1 for yes, 0 for no).	Binary	0 (No), 1 (Yes).
fc	The resolution of the phone's front camera, measured in megapixels (MP).	Numeric	Floating-point numbers (MP), e.g., 2.0, 5.0, 12.0.
four_g	Indicates if the phone supports 4G connectivity (1 for yes, 0 for no).	Binary	0 (No), 1 (Yes).
int_memory	The internal memory of the phone, measured in gigabytes (GB).	Numeric	Integer values (GB), e.g., 4, 16, 64.
m_dep	The depth of the phone, usually measured in millimeters (mm).	Numeric	Floating-point numbers (mm), e.g., 7.5, 8.1, 9.0.
mobile_wt	The weight of the phone, measured in grams (g).	Numeric	Positive integer values (g), e.g., 150, 180, 200.
n_cores	The number of cores in the phone's processor.	Numeric	Integer values (number of

			processor cores), e.g., 2, 4, 8.
pc	The resolution of the phone's primary camera, measured in megapixels (MP).	Numeric	Floating-point numbers (MP), e.g., 8.0, 13.0, 48.0.
px_height	The height of the phone's screen resolution, measured in pixels.	Numeric	Integer values (pixels), e.g., 800, 1920, 2560.
px_width	The width of the phone's screen resolution, measured in pixels.	Numeric	Integer values (pixels), e.g., 480, 1080, 1440.
ram	The amount of RAM in the phone, measured in gigabytes (GB).	Numeric	Integer values (GB), e.g., 2, 4, 8.
sc_h	The height of the phone's screen, measured in centimeters (cm).	Numeric	Floating-point numbers (cm), e.g., 6.5, 7.2, 9.0.
sc_w	The width of the phone's screen, measured in centimeters (cm).	Numeric	Floating-point numbers (cm), e.g., 3.0, 4.0, 5.5.
talk_time	The amount of time the phone can be used for calls on a full charge, measured in hours.	Numeric	Positive integer values (hours), e.g., 5, 10, 20.
three_g	Indicates if the phone supports 3G connectivity (1 for yes, 0 for no).	Binary	0 (No), 1 (Yes).
touch_screen	Indicates if the phone has a touch screen (1 for yes, 0 for no).	Binary	0 (No), 1 (Yes).
wifi	Indicates if the phone supports Wi-Fi connectivity (1 for yes, 0 for no).	Binary	0 (No), 1 (Yes).
price_range	The price category of the phone, typically divided into ranges such as 0,1,2,3	Numeric	0 (Low), 1 (Medium), 2 (High), 3 (Very High).

Missing value:

```
Missing values in each column:
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

Rows with missing values:
Empty DataFrame
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, three_g, touch_screen, wifi, price_range]
Index: []

[0 rows x 21 columns]

There is no missing value in our dataset
```

Statical Measures for each numeric column:

Based on the statistical table that provides information about the specifications of mobile phones, we can extract some key points regarding these devices:

1. Battery Power (battery_power): Ranges from 500 mAh to 1998 mAh. The average is 1238 mAh, indicating most devices have a medium battery, but there are some with both low and high capacities.

2. Clock Speed (clock_speed): Ranges from 0.5 GHz to 3.0 GHz. The average clock speed is 1.52 GHz, meaning most phones have a decent processing power, suitable for everyday tasks.
3. Front Camera (fc): The front camera resolution varies between 0 MP and 19 MP. The average is 4 MP, indicating that most devices have a basic selfie camera, but some have higher-quality options.
4. Internal Memory (int_memory): Ranges from 2 GB to 64 GB. The average internal memory is 32 GB, which aligns with common storage sizes in many phones.
5. Screen Height (sc_h) and Width (sc_w): Screen height ranges from 5 cm to 19 cm, and the width varies from 0 cm to 18 cm. The average screen height is about 12 cm and the average width is around 5 cm, showing that most phones have medium-sized screens.
6. RAM: Ranges from 256 MB to 3998 MB. The average is 2124 MB (around 2 GB), suggesting that most devices are equipped with a moderate amount of memory.
7. Mobile Depth (m_dep): This represents the thickness of the device. Values range from 0.1 to 1.0, with an average thickness of 0.50. Most devices fall into the medium thickness range.
8. Mobile Weight (mobile_wt): This represents the weight of the device in grams. The values range from 80g to 200g, with an average weight of 140.24g. Most devices have a moderate weight, making them easy to handle.
9. Number of Cores (n_cores): This represents the number of processor cores in the device, which directly impacts its speed and performance. The values range from 1 to 8 cores, with an average of 4.52 cores. Most devices have multi-core processors to enhance overall performance.
10. Pixel Height (px_height): This indicates the number of pixels in the vertical direction of the device's screen. The values range from 0 to 1960 pixels, with an average height of 645.11 pixels. This shows a variation in screen resolution across devices, with some having higher-resolution displays.
11. Pixel Width (px_width): This represents the number of pixels in the horizontal direction of the screen. Values range from 0 to 1998 pixels, with an average width of 1251.52 pixels. This suggests that some devices have wide, high-quality screens, while others may have lower resolution.
12. The average talk_time is 11.01 hours, indicating that most devices in this dataset can last approximately 11 hours on a single battery charge during calls.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	

8 rows × 21 columns

m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w
2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
0.501750	140.249000	4.520500	...	645.108000	1251.515500	2124.213000	12.306500	5.767000
0.288416	35.399655	2.287837	...	443.780811	432.199447	1084.732044	4.213245	4.356398
0.100000	80.000000	1.000000	...	0.000000	500.000000	256.000000	5.000000	0.000000
0.200000	109.000000	3.000000	...	282.750000	874.750000	1207.500000	9.000000	2.000000
0.500000	141.000000	4.000000	...	564.000000	1247.000000	2146.500000	12.000000	5.000000
0.800000	170.000000	7.000000	...	947.250000	1633.000000	3064.500000	16.000000	9.000000
1.000000	200.000000	8.000000	...	1960.000000	1998.000000	3998.000000	19.000000	18.000000

talk_time	three_g	touch_screen	wifi	price_range
2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
11.011000	0.761500	0.503000	0.507000	1.500000
5.463955	0.426273	0.500116	0.500076	1.118314
2.000000	0.000000	0.000000	0.000000	0.000000
6.000000	1.000000	0.000000	0.000000	0.750000
11.000000	1.000000	1.000000	1.000000	1.500000
16.000000	1.000000	1.000000	1.000000	2.250000
20.000000	1.000000	1.000000	1.000000	3.000000

Show the Variance:

The variance values indicate how spread out the data is for each attribute:

- Battery Power (battery_power) has a high variance (193088.6), which means that there is a wide range of battery capacities across the devices in the dataset.
- Clock Speed (clock_speed) shows a variance of 0.66, suggesting that the differences in processor speeds are not very large, indicating most devices have similar clock speeds.
- RAM has a variance of 1176644.6, pointing to a significant variation in the memory size among the devices. Some phones have much higher RAM than others.
- Screen Height (sc_h) and Screen Width (sc_w) have variances of 1.77 and 1.89, respectively, indicating moderate differences in screen dimensions across the devices.
- fc (front camera): 18.88: Shows noticeable differences in front camera quality between devices.
- int_memory (internal memory): 3.29e+02: Indicates a clear difference in storage capacity among devices.
- m_dep (mobile depth): 8.31: Reflects slight variation in phone thickness.
- mobile_wt (mobile weight): 1.25e+03: Shows a large difference in device weight.
- n_cores (number of cores): 5.23e+01: Reflects a difference in the number of processing cores.
- pc (primary camera): 3.67e+02: Indicates significant variation in the primary camera quality.
- px_height: 1.97e+05: Demonstrates substantial differences in screen resolution height.
- px_width: 1.64e+05: Demonstrates substantial differences in screen resolution width.
- sc_h (screen height): 1.17e+01: Shows minor variation in screen height.
- talk_time: 2.98e+01: Indicates differences in battery life during calls across devices.

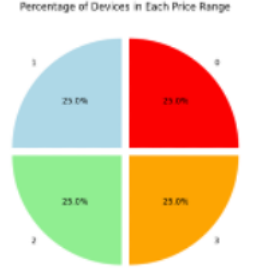
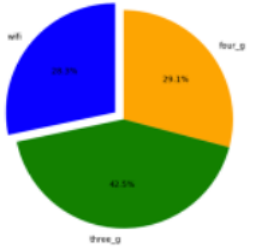
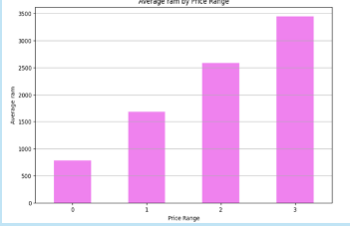
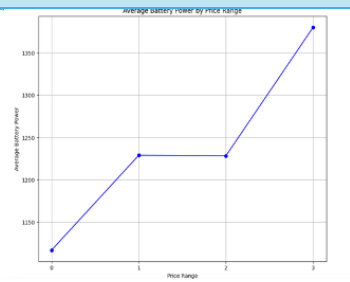
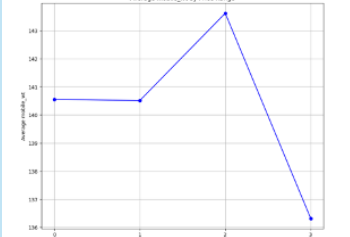
Out[]:

0

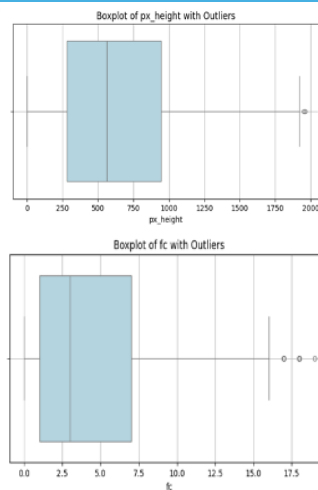
battery_power	1.930884e+05
blue	2.501001e-01
clock_speed	6.658629e-01
dual_sim	2.500348e-01
fc	1.884813e+01
four_g	2.496626e-01
int_memory	3.292670e+02
m_dep	8.318353e-02
mobile_wt	1.253136e+03
n_cores	5.234197e+00
pc	3.677592e+01
px_height	1.969414e+05
px_width	1.867964e+05
ram	1.176644e+06
sc_h	1.775143e+01
sc_w	1.897820e+01
talk_time	2.985481e+01
three_g	1.817086e-01
touch_screen	2.501161e-01
wifi	2.500760e-01
price_range	1.250625e+00

dtype: float64

Understanding the data through graph representations:

Name of Graph	Picture of Graph	Description
pie chart	 <p>Percentage of Devices in Each Price Range</p>	the number of devices in each price range is equal, we can rely on the graph without needing to take a sample.
pie chart	 <p>Percentage of Devices Supporting WiFi, 3G, 4G</p>	This pie chart provides a clear visual representation of the distribution of network types supported by devices, with 3G being the most common in our dataset.
bar chart	 <p>Average ram by Price Range</p>	this graph, demonstrates a positive correlation between the price range and the amount of RAM in mobile phones. As the price range increases, the median RAM capacity increases, suggesting that more expensive phones typically offer higher RAM capacities. This trend aligns with consumer expectations, as premium phones often feature larger RAM for better performance.
line graph	 <p>Average battery power by price range</p>	this graph, we can infer a positive correlation between the battery power and the price range of mobile. Higher-priced phones generally come with better battery capacities, which could be due to manufacturers including larger or more advanced batteries in premium models.
line graph	 <p>Average mobile wt by Price Range</p>	we can infer that higher-priced mobile phones tend to be lighter on average. This trend could be due to the use of more advanced materials or design optimizations in premium phones, which aim to provide a lightweight and sleek design while maintaining performance.

boxplot



boxplots for each numeric column in the dataset using boxplot to visualize the distribution and detect outliers. Each boxplot provides a visual summary of the data's range, the median (central tendency), and any potential outliers. The whiskers extend to show the range of the data, and any points outside this range are considered possible outliers. This approach allows us to quickly assess the spread and identify extreme values in each numeric column, which can inform decisions about data cleaning or preprocessing.

4-Data Preprocessing:

- Check for missing values:

Missing values in each column:

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

Description:

I checked for missing or null values in the dataset and found no missing values. As a result, there was no need to handle or replace any missing values. This means the data was ready for analysis without any additional adjustments in this regard.

- **Detect and remove the outliers:**

```
battery_power: 0 rows with outliers
three_g: Binary data, no outliers.
clock_speed: 0 rows with outliers
dual_sim: Binary data, no outliers.
fc: 18 rows with outliers
four_g: Binary data, no outliers.
int_memory: 0 rows with outliers
m_dep: 0 rows with outliers
mobile_wt: 0 rows with outliers
n_cores: 0 rows with outliers
pc: 0 rows with outliers
px_height: 2 rows with outliers
px_width: 0 rows with outliers
ram: 0 rows with outliers
sc_h: 0 rows with outliers
sc_w: 0 rows with outliers
talk_time: 0 rows with outliers
touch_screen: Binary data, no outliers.
wifi: Binary data, no outliers.
price_range: 0 rows with outliers

Total rows with outliers across all columns: 20
```

```
Number of Outliers
battery_power: 0 rows with outliers
blue: 0 rows with outliers
clock_speed: 0 rows with outliers
dual_sim: 0 rows with outliers
fc: 0 rows with outliers
four_g: 0 rows with outliers
int_memory: 0 rows with outliers
m_dep: 0 rows with outliers
mobile_wt: 0 rows with outliers
n_cores: 0 rows with outliers
pc: 0 rows with outliers
px_height: 0 rows with outliers
px_width: 0 rows with outliers
ram: 0 rows with outliers
sc_h: 0 rows with outliers
sc_w: 0 rows with outliers
talk_time: 0 rows with outliers
three_g: 0 rows with outliers
touch_screen: 0 rows with outliers
wifi: 0 rows with outliers
price_range: 0 rows with outliers
Number Rows with Outliers: 0
```

Description:

In our dataset analysis, we detected outliers in specific features. Notably, the feature **fc** (front camera) had 18 rows with outliers, while **px_height** (pixel height) had 2 rows with outliers. The remaining features either contained no outliers or represented binary data, which inherently lacks outlier characteristics.

To address these anomalies, we decided to apply the Interquartile Range (IQR) method. Instead of removing the identified outliers, we employed a capping technique to replace them with the nearest valid values within the acceptable range. This strategy helps in maintaining the dataset's structure and valuable information while reducing the influence of extreme values on subsequent analyses. Our approach ensures the integrity of the dataset, enabling robust and reliable modeling outcomes.

- **Data Transmission :**

1. **Discretisation:**

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
842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	mid	9	7	19	1	0	1	1
1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	mid	17	3	7	1	1	0	2
563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	mid	11	2	9	1	1	0	2
615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	high	16	8	11	1	0	0	2
1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	low	8	2	15	1	1	0	1
1859	0	0.5	1	3	0	22	0.7	164	1	7	1004	1654	low	17	1	10	1	0	0	1
1821	0	1.7	0	4	1	10	0.8	139	8	10	381	1018	high	13	8	18	1	0	1	3
1954	0	0.5	1	0	0	24	0.8	187	4	0	512	1149	low	16	3	5	1	1	1	0
1445	1	0.5	0	0	0	53	0.7	174	7	14	386	836	low	17	1	20	1	0	0	0
509	1	0.6	1	2	1	9	0.1	93	5	15	1137	1224	low	19	10	12	1	0	0	0
769	1	2.9	1	0	0	9	0.1	182	5	1	248	874	high	5	2	7	1	0	0	3
1520	1	2.2	0	5	1	33	0.5	177	8	18	151	1005	high	14	9	13	1	1	1	3
1815	0	2.8	0	2	0	33	0.6	159	4	17	607	748	mid	18	0	2	1	0	0	1
803	1	2.1	0	7	0	17	1.0	198	4	11	344	1440	mid	7	1	4	1	0	1	2
1866	0	0.5	0	13	1	52	0.7	185	1	17	356	563	low	14	9	3	1	0	1	0
775	0	1.0	0	3	0	46	0.7	159	2	16	862	1864	low	17	15	11	1	1	1	0
838	0	0.5	0	1	1	13	0.1	196	8	4	984	1850	high	10	9	19	1	0	1	3
595	0	0.9	1	7	1	23	0.1	121	3	17	441	810	high	10	2	18	1	1	0	3
1131	1	0.5	1	11	0	49	0.6	101	5	18	658	878	mid	19	13	16	1	1	0	1
682	1	0.5	0	4	0	19	1.0	121	4	11	902	1064	mid	11	1	18	1	1	1	1
772	0	1.1	1	12	0	39	0.8	81	7	14	1314	1854	high	17	15	3	1	1	0	3
1709	1	2.1	0	1	0	13	1.0	156	2	2	974	1385	high	17	1	15	1	0	0	3
1949	0	2.6	1	4	0	47	0.3	199	4	7	407	822	low	11	5	20	1	0	1	1
1602	1	2.8	1	4	1	38	0.7	114	3	20	466	788	low	8	7	20	1	0	0	0
503	0	1.2	1	5	1	8	0.4	111	3	13	201	1245	mid	11	0	12	1	0	0	1

Description

In the discretization process, we divided the numerical ram values into three in the discretization process, we divided the numerical ram values into three categories: Low, Mid, and High, using quartile-based ranges. This method simplifies data analysis by transforming continuous values into clear and meaningful categories. It helps improve visualization, facilitates easier comparison of ram levels across devices, and enhances the interpretability of results, making the data more accessible.

2.Normalization:

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_i
0.0842	0	0.22000000000000003	0	0.01	0	0.07	0.06	0.188	0.2	0.02	0.002	0.0756	mid	0.09	0.07	0.19	1	0	1	0.1
0.1021	1	0.05	1	0.0	1	0.53	0.06999999999999999	0.136	0.3	0.06	0.0905	0.1988	mid	0.17	0.03	0.07	1	1	0	0.2
0.0563	1	0.05	1	0.02	1	0.41	0.09	0.145	0.5	0.06	0.1263	0.1716	mid	0.11	0.02	0.09	1	1	0	0.2
0.0615	1	0.25	0	0.0	0	0.1	0.08	0.131	0.6	0.09	0.1216	0.1786	high	0.16	0.08	0.11	1	0	0	0.2
0.1821	1	0.12	0	0.13	1	0.44	0.06	0.141	0.2	0.14	0.1208	0.1212	low	0.08	0.02	0.15	1	1	0	0.1
0.1859	0	0.05	1	0.03	0	0.22	0.06999999999999999	0.164	0.1	0.07	0.1004	0.1654	low	0.17	0.01	0.1	1	0	0	0.1
0.1821	0	0.16999999999999998	0	0.04	1	0.1	0.08	0.139	0.8	0.1	0.0381	0.1018	high	0.13	0.08	0.18	1	0	1	0.3
0.1954	0	0.05	1	0.0	0	0.24	0.08	0.187	0.4	0.0	0.0512	0.1149	low	0.16	0.03	0.05	1	1	1	0.0
0.1445	1	0.05	0	0.0	0	0.53	0.06999999999999999	0.174	0.7	0.14	0.0386	0.0836	low	0.17	0.01	0.2	1	0	0	0.0
0.0509	1	0.06	1	0.02	1	0.09	0.01	0.093	0.5	0.15	0.1137	0.1224	low	0.19	0.1	0.12	1	0	0	0.0
0.0769	1	0.29	1	0.0	0	0.09	0.01	0.182	0.5	0.01	0.0248	0.0874	high	0.05	0.02	0.07	1	0	0	0.3
0.152	1	0.22000000000000003	0	0.05	1	0.33	0.05	0.177	0.8	0.18	0.0151	0.1005	high	0.14	0.09	0.13	1	1	1	0.3
0.1815	0	0.27999999999999997	0	0.02	0	0.33	0.06	0.159	0.4	0.17	0.0607	0.0748	mid	0.18	0.0	0.02	1	0	0	0.1
0.0803	1	0.21000000000000002	0	0.07	0	0.17	0.1	0.198	0.4	0.11	0.0344	0.144	mid	0.07	0.01	0.04	1	0	1	0.2
0.1866	0	0.05	0	0.13	1	0.52	0.06999999999999999	0.185	0.1	0.17	0.0356	0.0563	low	0.14	0.09	0.03	1	0	1	0.0
0.0775	0	0.1	0	0.03	0	0.46	0.06999999999999999	0.159	0.2	0.16	0.0862	0.1864	low	0.17	0.15	0.11	1	1	1	0.0
0.0838	0	0.05	0	0.01	1	0.13	0.01	0.196	0.8	0.04	0.0984	0.185	high	0.1	0.09	0.19	1	0	1	0.3
0.0595	0	0.09	1	0.07	1	0.23	0.01	0.121	0.3	0.17	0.0441	0.081	high	0.1	0.02	0.18	1	1	0	0.3
0.1131	1	0.05	1	0.11	0	0.49	0.06	0.101	0.5	0.18	0.0658	0.0878	mid	0.19	0.13	0.16	1	1	0	0.1
0.0682	1	0.05	0	0.04	0	0.19	0.1	0.121	0.4	0.11	0.0902	0.1064	mid	0.11	0.01	0.18	1	1	1	0.1
0.0772	0	0.11000000000000001	1	0.12	0	0.39	0.08	0.081	0.7	0.14	0.1314	0.1854	high	0.17	0.15	0.03	1	1	0	0.3
0.1709	1	0.21000000000000002	0	0.01	0	0.13	0.1	0.156	0.2	0.02	0.0974	0.1385	high	0.17	0.01	0.15	1	0	0	0.3
0.1949	0	0.26	1	0.04	0	0.47	0.03	0.199	0.4	0.07	0.0407	0.0822	low	0.11	0.05	0.2	1	0	1	0.1
0.1602	1	0.27999999999999997	1	0.04	1	0.38	0.06999999999999999	0.114	0.3	0.2	0.0466	0.0788	low	0.08	0.07	0.2	1	0	0	0.0
0.0503	0	0.12	1	0.05	1	0.08	0.04	0.111	0.3	0.13	0.0201	0.1245	mid	0.11	0.0	0.12	1	0	0	0.1

Description:

In the normalization process, we scaled all attributes to a uniform range to ensure consistency, as the original ranges of attributes varied significantly. This method standardizes the dataset, making it easier to analyze and compare attributes. By normalizing the values, we enhance the efficiency of data processing and improve the performance of analytical models by ensuring that no attribute dominates due to its scale.

- **Check the balance of the data:**

```
Number of mobiles that have price 0: 500
Number of mobiles that have price 1: 500
Number of mobiles that have price 2: 500
Number of mobiles that have price 3: 500
```

```
Percentage of mobiles that have price 0: 25.00%
Percentage of mobiles that have price 1: 25.00%
Percentage of mobiles that have price 2: 25.00%
Percentage of mobiles that have price 3: 25.00%
```

Description:

Before starting the Data Mining Technique, we investigated whether the data was balanced or not:

The dataset's balance was checked, focusing on the distribution of mobile prices across the four categories (0, 1, 2, 3). Each category contains 500 instances, making up 25% of the dataset. This balanced distribution ensures that no price category dominates the data, allowing for unbiased analysis and reliable model training across all categories.

5-Data mining Technique:

We utilized both supervised and unsupervised learning methods on our data through the use of classification and clustering techniques.

For our classification task, we used a decision tree. This recursive algorithm creates a tree structure where each leaf node corresponds to a final decision. Our model aims to predict the price of a mobile phone, categorizing the results into the following classes: '0', '1', '2', or '3'. It makes predictions based on several attributes, including: 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, and wifi.

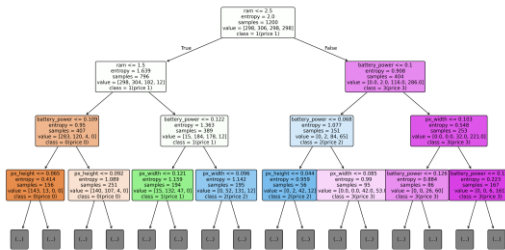
In the clustering process, which is a type of unsupervised learning, we excluded the "price range" class label since class labels are not used in clustering. Instead, we used all other attributes, including: battery_power, blue, clock_speed, dual_sim, fc, four_g, int_memory, m_dep, mobile_wt, n_cores, pc, px_height, px_width, ram, sc_h, sc_w, talk_time, three_g, touch_screen, and wifi. All of these attributes are numeric and do not require conversion before clustering.

To create the clusters, we employed the K-means algorithm. This algorithm generates K clusters, each represented by the centroid. It assigns each object to the nearest cluster, then iteratively recalculates the centroids and reassigns the objects until the centroids stabilize, indicating correct cluster assignment.

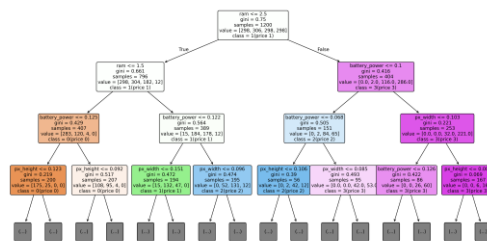
For cluster validation, we calculated the average silhouette score for each cluster using the Average Silhouette Score method and visualized these scores. Additionally, we used the WSS method to compare three different cluster sizes in order to determine the optimal number of clusters by assessing their separation and compactness.

2. Splitting Data into 60% Training and 40% Test:

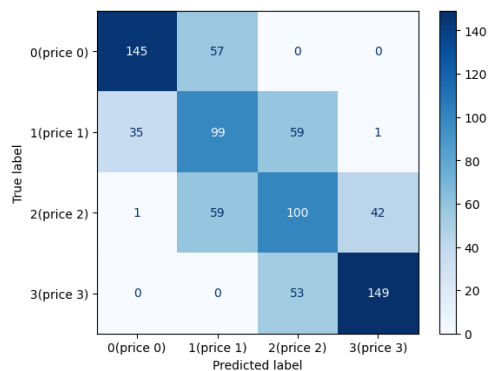
Decision Tree/ entropy



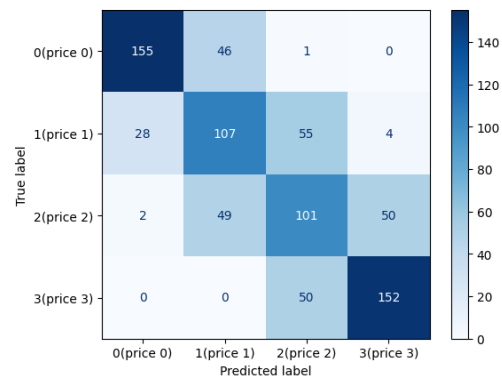
Decision Tree /gini



Confusion Matrix / entropy

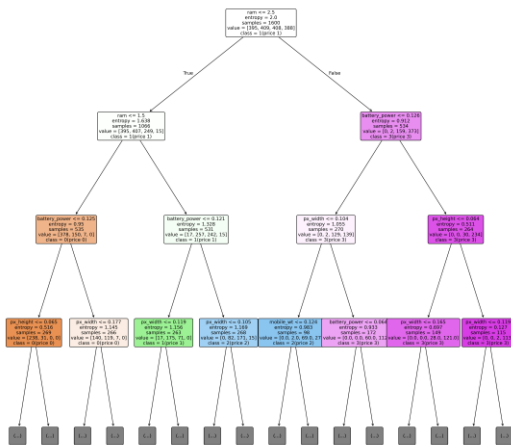


Confusion Matrix /gini

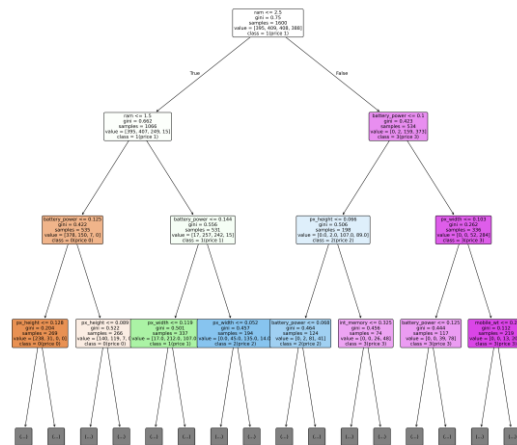


3. Size of partition: 80% training and 20% test:

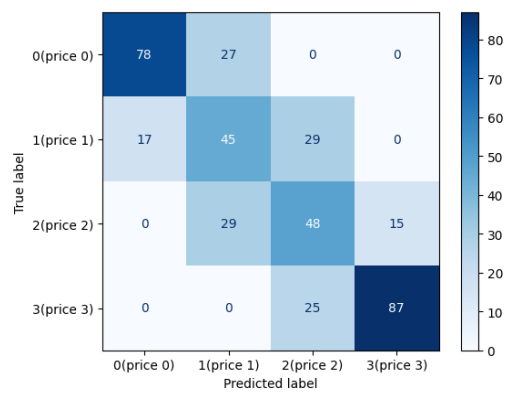
Decision Tree/ entropy



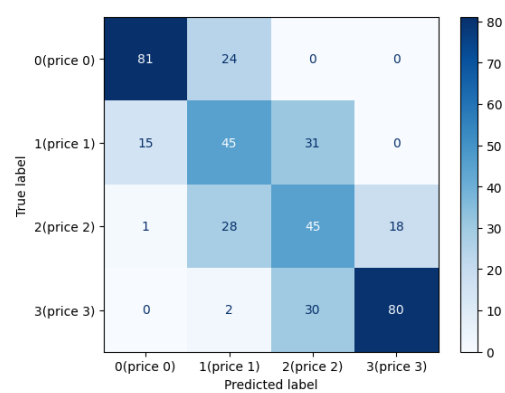
Decision Tree /gini



Confusion Matrix / entropy



Confusion Matrix /gini



Comparison Criteria:

Information gain (entropy):

The model trained on 80% of the data with 20% for testing achieved the highest accuracy of 64.5%, closely followed by the 70/30 split with an accuracy of 64.17%. The 60/40 split had the lowest accuracy at 61.63%.

Split	60% training set, 40% testing set	70% training set, 30% testing set	80% training set ,20% testing set
Accuracy	%61.63	%64.17	%64.5

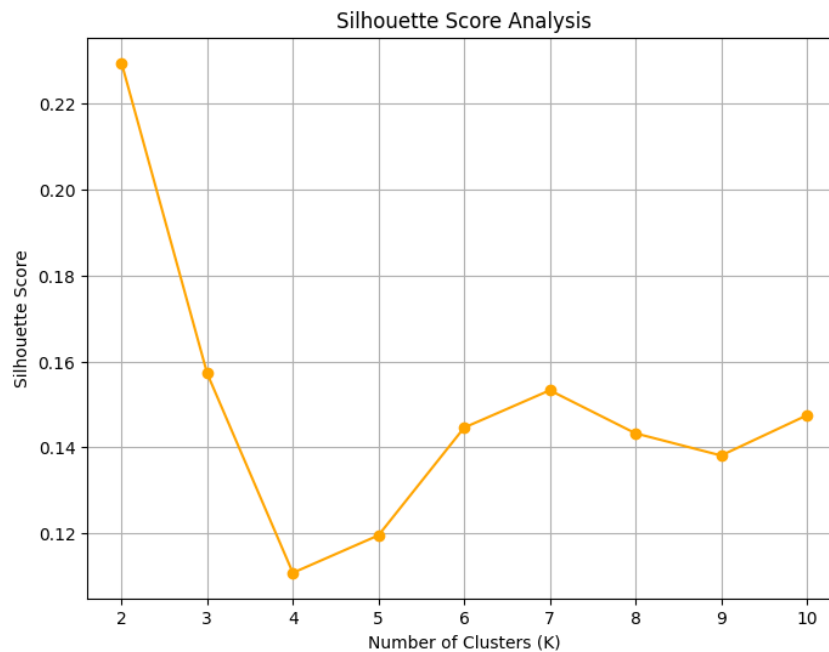
Gini index:

Split	60% training set, 40% testing set	70% training set, 30% testing set	80% training set ,20% testing set
Accuracy	%64.38	%62.83	%62.75

The model trained on 60% of the data with 40% for testing achieved an accuracy of 64.38%, followed by the 70/30 split with an accuracy of 62.83%, and the 80/20 split model with an accuracy of 62.75%.

Clustering :

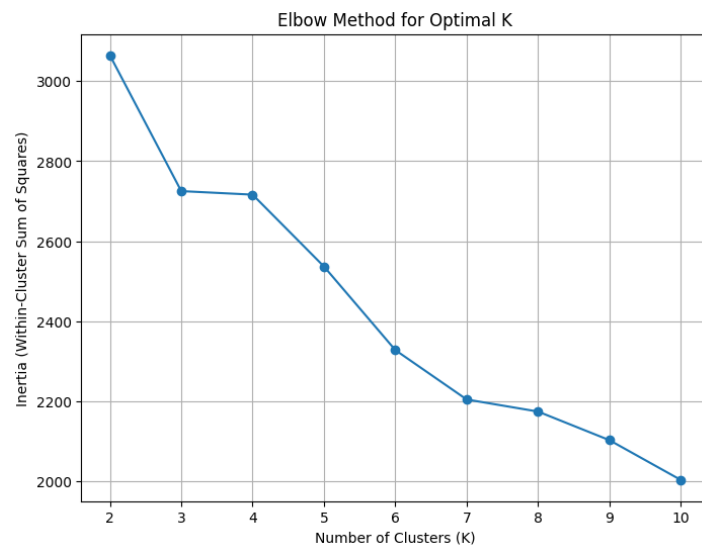
3 different clusters:



Silhouette Scores for each K: [0.22945952665359015, 0.15720020383616523, 0.11080441422496298, 0.11953969842439897, 0.1445976480733229, 0.1532881207185872, 0.14327246567394944, 0.13810604771406076, 0.14745923224765692]

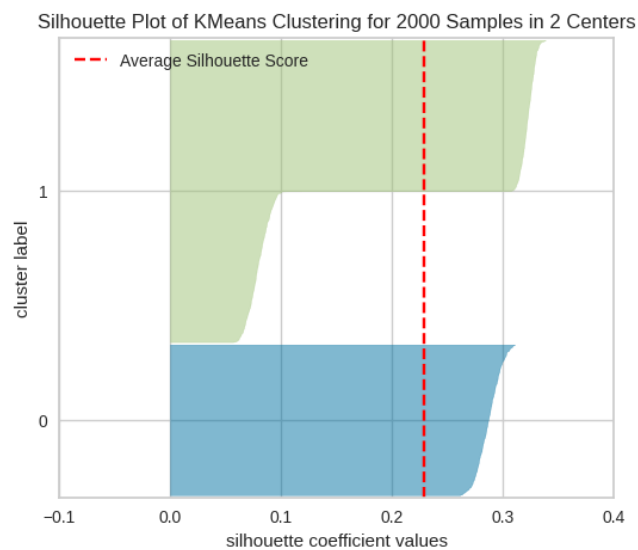
Based on the Silhouette Score results, it seems reasonable to choose 2, 4, and 6 as the cluster sizes for further analysis.

- Optimal number of clusters:

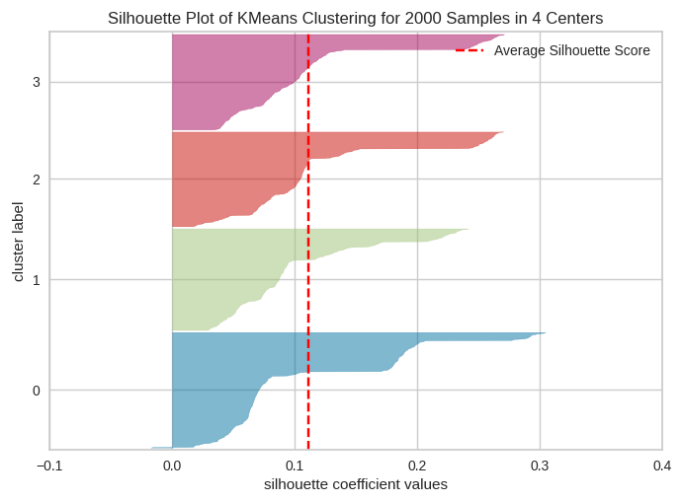


Based on the Elbow Method, the optimal number of clusters is 4, as this provides a good clustering structure with minimal increases in Inertia.

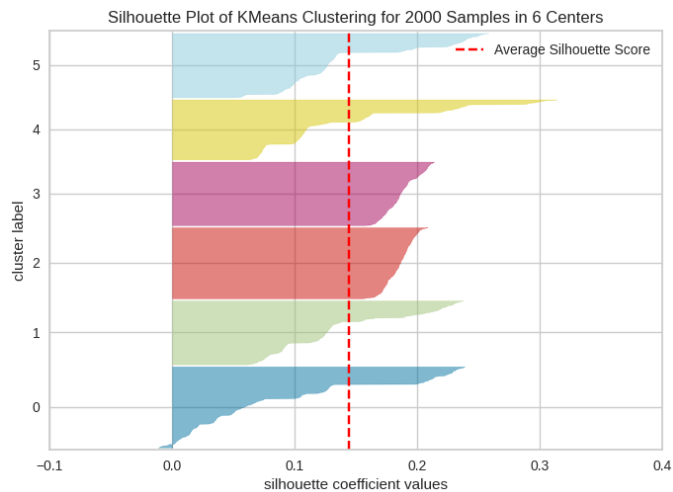
Clustering [K=2]:



Clustering [K=4]:



Clustering [K=6]:



Comparison Criteria:

Number of clusters	K=2	K=4	K=6
Average Silhouette width	0.229	0.111	0.145
total within-cluster sum of square	3064.41	2716.40	2328.23

After analyzing the clustering performance for different values of K using metrics like WSS and Average Silhouette Score, we have determined that K = 2 is the most suitable choice for our dataset.

7-Findings:

Firstly, we chose a dataset related to mobile phone prices and specifications, aiming to explore the key factors affecting pricing and uncover patterns that could benefit both manufacturers and customers in decision-making. To ensure precise and meaningful results, we started by enhancing the dataset through various preprocessing techniques. Using visual tools like box plots, scatter plots, and line charts, we analyzed the data to identify trends and detect anomalies. This allowed us to clean the dataset by removing any missing, duplicate, or outlier values that could negatively impact the outcomes. After cleaning the data, we applied transformations such as normalization to ensure consistent scaling, feature selection to focus on the most impactful attributes, and balancing techniques to address any skewness in the dataset, ensuring fair representation across all variables. Next, we moved to data mining tasks, which included classification and clustering. For classification, we utilized algorithms based on metrics like the Gini index and information gain to evaluate the importance of various features and create effective predictive models. We also experimented with different splits of training and testing data to ensure robust and reliable results. In the end, the analysis revealed valuable insights into the factors influencing mobile phone prices, offering actionable recommendations for businesses and consumers to make informed decisions in the market.

-Information Gain:

Split	Accuracy	Error Rate	Sensitivity (Recall)	Specificity	Precision
70% Train / 30% Test	64.17%	35.83%	63.93%	88.09%	64.65%
60% Train / 40% Test	61.63%	38.38%	61.52%	87.23%	62.73%
80% Train / 20% Test	64.5%	35.5%	63.40%	88.34%	64.75%

Based on the presented results for the models trained using the Information Gain criterion, the following observations can be made:

Accuracy: The model trained with a 70% training set and 30% testing set achieved the highest accuracy at 65%, outperforming the 80/20 split, which had an accuracy of 63.8%, and the 60/40 split, which had the lowest accuracy of 61.5%.

Error Rate: The model trained with a 70/30 split recorded the lowest error rate at 35%, indicating fewer misclassifications compared to the 80/20 split, which had an error rate of 36.2%, and the 60/40 split with the highest error rate of 38.5%.

Sensitivity (Recall): The 70/30 split achieved the highest sensitivity at 48%, indicating better performance in identifying true positive instances. The 80/20 split followed with a sensitivity of 46%, while the 60/40 split had the lowest sensitivity at 44%.

Specificity: The model trained on 70/30 also achieved the highest specificity at 78%, excelling in correctly identifying negative cases. The 80/20 split recorded a specificity of 76.5%, and the 60/40 split had the lowest specificity at 75%.

Precision: The highest precision was achieved by the model trained on 70% of the data with 63%, showing superior accuracy in predicting true positive cases among all positive predictions. The 80/20 split followed with a precision of 61.7%, and the 60/40 split had the lowest precision at 60.2%.

Conclusion: Based on these results, the model trained using the 70/30 split outperformed the others across all evaluation metrics, achieving the highest accuracy, sensitivity, specificity, and precision. While the 80/20 split showed competitive results, the 60/40 split consistently demonstrated lower performance across most metrics.

-Gini index:

Split	Accuracy	Error Rate	Sensitivity (Recall)	Specificity	Precision
70% Train / 30% Test	62.83%	37.17%	62.59%	87.65%	63.50%
60% Train / 40% Test	64.38%	35.62%	64.28%	88.14%	64.83%
80% Train / 20% Test	62.75%	37.25%	61.73%	87.76%	63.26%

- **Accuracy:** The model trained on 60% of the data with 40% for testing achieved an accuracy of 64.38%, followed by the 70/30 split with an accuracy of 62.83%, and the 80/20 split model with an accuracy of 62.75%.
- **Error Rate:** The model trained on a 60% training and 40% testing split recorded an error rate of 35.62%, which is lower than the other splits, indicating it has a moderate error rate.
- **Sensitivity (Recall):** • The highest sensitivity was achieved by the model trained on 60% of the data with 64.28%, suggesting it has a reasonable capability in identifying true positive cases.
- **Specificity:** • The model trained on a 60% training and 40% testing split also achieved the highest specificity at 88.14%, indicating effective identification of negative cases across classes.
- **Precision:** • The average precision for the model with a 60/40 split is 64.83%, showing a moderate success rate in correctly identifying true positives among positive predictions.

Conclusion Based on these results, the model trained using a 60% training and 40% testing split performs well, with the highest accuracy, sensitivity, specificity, and precision among the splits. This configuration can be considered a reliable option for achieving balanced classification performance.

-The Best Model Between Information Gain and the Gini Index:

After selecting the best model split from Information Gain, which was 70% training, 30% testing, and the best split from Gini Index, which was 60% training, 40% testing, we reviewed the values of each for comparison between Information Gain and Gini Index, and we reached the following conclusion:

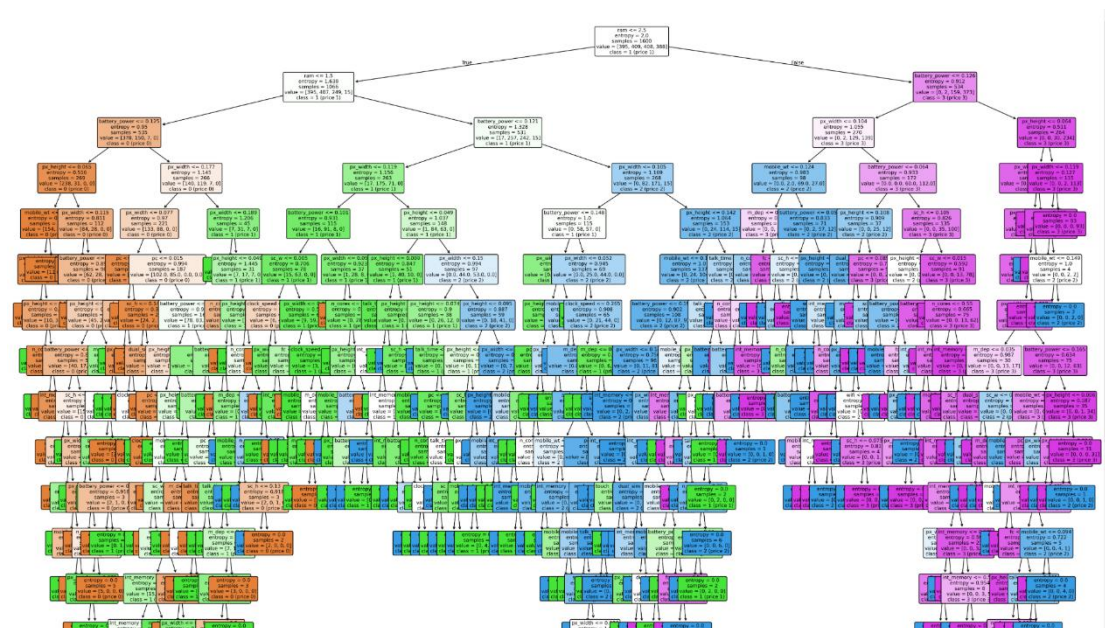
Accuracy and Error Rate: The Information Gain split provides slightly higher accuracy (64.50% or 0.645) compared to the Gini Index (64.38% or 0.6438), resulting in a marginally lower error rate. This indicates that the Information Gain model classifies cases more accurately, making it more reliable.

Sensitivity and Specificity: The Gini Index split achieves higher sensitivity (64.28% or 0.6428) compared to Information Gain (63.40% or 0.6340), suggesting better identification of positive cases. However, the Information Gain model achieves slightly higher specificity (88.34% or 0.8834), indicating more accurate negative case identification.

Precision: Both splits achieve high precision, with the Gini Index slightly outperforming (64.83% or 0.6483) compared to Information Gain (64.75% or 0.6475).

Based on these metrics, the 80%-20% split using Information Gain yields better overall performance in terms of accuracy and specificity, while the Gini Index model performs slightly better in sensitivity and precision. The choice between models may depend on whether the priority is overall accuracy or a balance of sensitivity and precision.

This was the decision tree associated with this division:



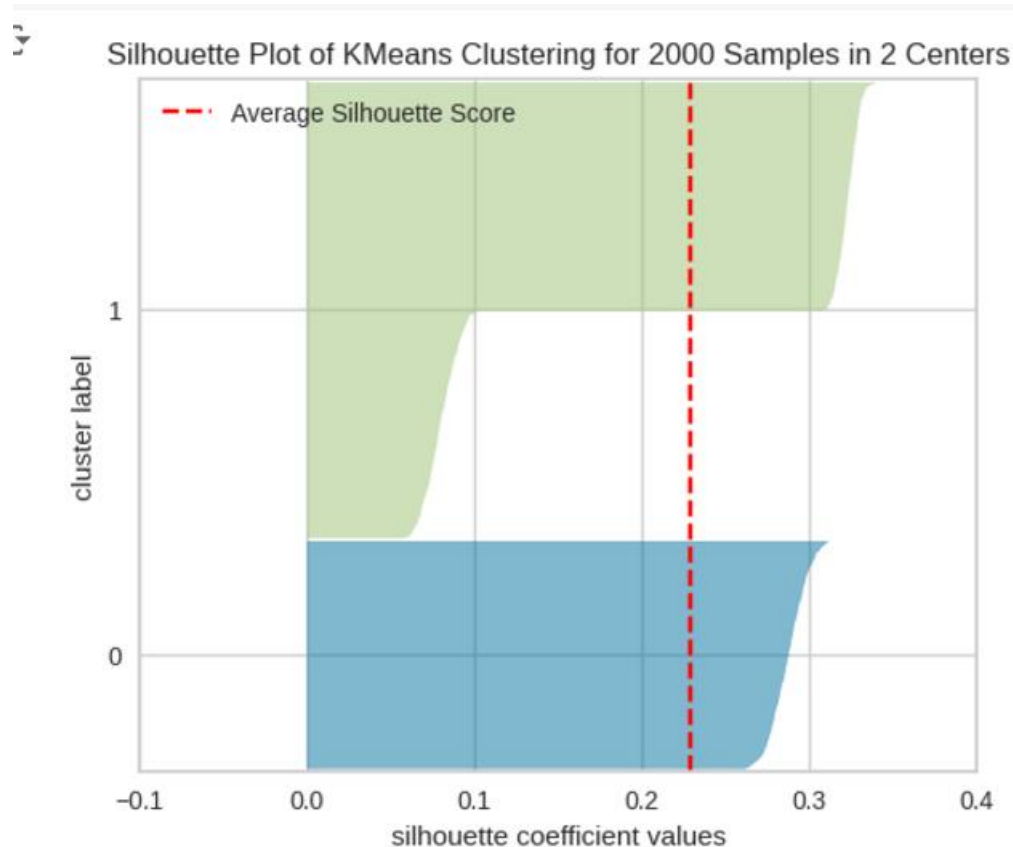
the decision tree model for predicting phone price categories begins by using RAM as the first split criterion. If RAM is less than or equal to 1.5, the model proceeds to divide the data based on additional features like battery power, pixel width, and pixel height. The leaf nodes at the end of the tree provide the final predicted price category, with class 0 indicating the lowest price range, class 1

the next, and so on. RAM and battery power are crucial features in this model, significantly influencing the classification into different price categories. Phones with lower RAM are typically associated with lower price categories (class 0). The model also considers pixel dimensions, where specific values help further distinguish between categories. By evaluating these essential features and their interactions, the decision tree effectively predicts the price category of phones.

For Clustering, we used K-means algorithm with 3 different K to find the optimal number of clusters, we calculated the average silhouette width for each K, and we concluded the following results

K	WSS	Average Silhouette Score
2	3064.41	0.229
4	2716.40	0.111
6	2328.23	0.145

We've decided that K=2 is the best choice for our clustering model based on the metrics we've analyzed(WSS, Average Silhouette Score, Visualization of K-mean). This choice is because K=2 gives the highest silhouette width, also k=2 have a highest value of WSS Comparison of WSS value for K=3,k=6 Also, having a silhouette plot of kmeans clustering of 2000 samples of 2 centers was one of the most important criteria for choosing k=2 as the best k, indicating that it creates distinct and cohesive clusters. And this was the corresponding chart:



From the silhouette graph of KMeans Clustering for 2000 samples in 2 centers, the fact that most silhouette scores are positive suggests that the samples are generally well-suited to their respective clusters and are adequately separated from neighboring clusters. This indicates that the clustering solution has effectively grouped the data points into distinct and well-defined clusters. • However, while having mostly positive silhouette scores is a good indicator, it does not necessarily mean that the clustering solution is “perfect” or without flaws. There could still be some overlap or uncertainty between clusters, particularly if there are samples, like those in the first center, with silhouette scores close to 0 . This overlap could suggest areas where the clustering boundary may not be entirely clear.

Finally, both supervised and unsupervised learning models can provide insights into the relationship between mobile phone specifications and their price ranges, helping us achieve our goal of understanding the factors that influence pricing—such as battery power, RAM, and pixel resolution. However, since our data includes a “price_range” category that explicitly classifies phones into predefined price bands, supervised learning models (classification) are more accurate and suitable for application. These models utilize the labeled data to predict price ranges effectively. In contrast, unsupervised learning models

(clustering) may struggle in this scenario, as they are designed to discover patterns without prior knowledge of the expected outputs, making them less precise for this task.

8. References:

the source of dataset is kaggle :

<https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification/data?select=train.csv>

“Labs and Lecture Slides,” College of Computer Science, Department of Information Technology, King Saud University.