Project Documentation: Global Mobile Market Analysis (2025)

NAME: PRANOY K H
COURSE: DATASCIENCE
EMAIL: khpranoy17@gmail.com

1. Introduction

This project analyzes the "Global Mobile Prices 2025 Extended" datasets, which contains detailed specifications and pricing information for approximately 1,000 mobile devices available in the global market. The datasets spans various brands, operating systems, and hardware configurations, offering a snapshot of the competitive landscape in 2025.

2. Aim

The primary aim of this analysis is to:

- Uncover pricing trends and market segmentation strategies across major mobile brands.
- Identify the correlation between specific hardware components (e.g., RAM, Storage) and device pricing.
- Determine which brands and models offer the best "Value for Money" based on technical specifications.

3. Business Problem

In a highly saturated mobile market, both consumers and manufacturers face specific challenges:

- **Consumers** struggle to identify which devices offer the best performance relative to their cost.
- **Manufacturers** need to understand how their pricing strategy compares to competitors within specific market segments (e.g., Budget vs. Flagship). This analysis seeks to address these problems by quantifying "value" and visualizing market positioning.

4. Project Workflow

The analysis follows a structured pipeline:

1. **Import Libraries**: Importing libraries which are required for analyzing,cleaning and plotting

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import date
```

2. **Data Ingestion:** Loading raw data from CSV format.

```
data = pd.read_csv(r'C:\Users\khpra\Downloads\archive
(2)\Global_Mobile_Prices_2025_Extended.csv')
```

3. **Preprocessing:** Cleaning strings, handling structural inconsistencies, and checking for duplicates.

```
data.head()
data.tail()
data.info()
data.columns
data.isnull().sum()
data.describe()
```

4. **Feature Engineering:** Creating new metrics to quantify market segments and value.
5. **Statistical Analysis:** Detecting outliers and calculating descriptive statistics.
6. **Exploratory Data Analysis (EDA):** Visualizing data distributions and relationships.
7. **Reporting:** Synthesizing findings into actionable insights.

5. Data Understanding

- **Dataset Size:** 1,000 Rows, 15 Columns (Initial).

	brand	model	price_usd	ram_gb	storage_gb	camera_mp	battery_mah	display_size_inch	charging_watt	5g_support	os	processor	rating	release_month	year
0	Oppo	A98 111	855	16	128	108	6000	6.6	33	Yes	Android	Helio G99	3.8	February	2025
1	Realme	11 Pro+ 843	618	6	128	64	4500	6.9	100	Yes	Android	Tensor G4	4.4	August	2025
2	Xiaomi	Redmi Note 14 Pro 461	258	16	64	64	4000	6.8	44	Yes	Android	A18 Pro	4.1	March	2025
3	Vivo	V29e 744	837	6	512	48	4500	6.0	65	Yes	Android	Exynos 2400	4.1	August	2025
4	Apple	iPhone 16 Pro Max 927	335	12	128	200	5000	6.9	100	Yes	iOS	Dimensity 9300	3.5	February	2025

- **Data Types:** Mix of Numerical (Float/Int) and Categorical (Object) data.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   brand              1000 non-null   object
 1   model              1000 non-null   object
 2   price_usd          1000 non-null   int64
 3   ram_gb             1000 non-null   int64
 4   storage_gb         1000 non-null   int64
 5   camera_mp          1000 non-null   int64
 6   battery_mah        1000 non-null   int64
 7   display_size_inch  1000 non-null   float64
 8   charging_watt      1000 non-null   int64
 9   5g_support         1000 non-null   object
 10  os                 1000 non-null   object
 11  processor          1000 non-null   object
 12  rating             1000 non-null   float64
 13  release_month      1000 non-null   object
 14  year               1000 non-null   int64
dtypes: float64(2), int64(7), object(6)
memory usage: 117.3+ KB
```

- **Key Variables:**
  - *Categorical:* Brand, Model, OS, Processor, 5G Support.
  - *Numerical:* Price (USD), RAM, Storage, Battery (mAh), Camera (MP), Screen Size.

6. Data Cleaning

Data cleaning ensures the reliability of the analysis. The following steps were executed:

- **Model Name Separation:** The raw model column contained combined names and numbers (e.g., "iPhone 16 999"). This was split into distinct Model name and model_no columns.

```
data[['Model','model_no']] = data['model'].str.rsplit(' ', n=1, expand=True)

data.head()
```

OUTPUT

	brand	model	price_usd	ram_gb	storage_gb	camera_mp	battery_mah	display_size_inch	charging_watt	5g_support	os	processor	rating	release_month	year	Model	model_no
0	Oppo	A98 111	855	16	128	108	6000	6.6	33	Yes	Android	Helio G99	3.8	February	2025	A98	111
1	Realme	11 Pro+ 843	618	6	128	64	4500	6.9	100	Yes	Android	Tensor G4	4.4	August	2025	11 Pro+	843
2	Xiaomi	Redmi Note 14 Pro 461	258	16	64	64	4000	6.8	44	Yes	Android	A18 Pro	4.1	March	2025	Redmi Note 14 Pro	461
3	Vivo	V29e 744	837	6	512	48	4500	6.0	65	Yes	Android	Exynos 2400	4.1	August	2025	V29e	744
4	Apple	iPhone 16 Pro Max 927	335	12	128	200	5000	6.9	100	Yes	iOS	Dimensity 9300	3.5	February	2025	iPhone 16 Pro Max	927

- **String Sanitization:** All object columns were stripped of whitespace and standardized to Title Case to ensure consistency (e.g., unifying "apple" and "Apple").

```
# List of object columns to clean
object_cols = ['brand', 'model', '5g_support', 'os', 'processor', 'release_month']

# Clean up string columns: strip whitespace and convert to title case (except for 5g_support and os for
simplicity, which looked clean)
for col in object_cols:
    if data[col].dtype == 'object':
        data[col] = data[col].str.strip()
        if col not in ['5g_support', 'os']:
            data[col] = data[col].str.title()

# Check unique values after cleaning for key categorical columns to verify consistency
print("--- Unique Values after String Cleaning ---")
print("Brand:", data['brand'].unique())
print("OS:", data['os'].unique())
print("Processor:", data['processor'].unique())
print("5G Support:", data['5g_support'].unique())
```

OUTPUT

```
# Check unique values after cleaning for key categorical columns to verify consistency
print("--- Unique Values after String Cleaning ---")
print("Brand:", data['brand'].unique())
print("OS:", data['os'].unique())
print("Processor:", data['processor'].unique())
print("5G Support:", data['5g_support'].unique())
```

- **Duplicate Handling:** A check for duplicate records was performed (0 duplicates found).

```
# Handle duplicates
initial_rows = data.shape[0]
data.drop_duplicates(inplace=True)
rows_after_dropping = data.shape[0]
duplicates_removed = initial_rows - rows_after_dropping

print(f"\n--- Duplicates Handling ---")
print(f"Initial rows: {initial_rows}")
print(f"Rows after removing duplicates: {rows_after_dropping}")
print(f"Number of duplicates removed: {duplicates_removed}")
print(f"\nUnique models after cleaning: {data['model'].nunique()}")
```

OUTPUT

```
--- Duplicates Handling ---
Initial rows: 1000
Rows after removing duplicates: 1000
Number of duplicates removed: 0
Unique models after cleaning: 992
```

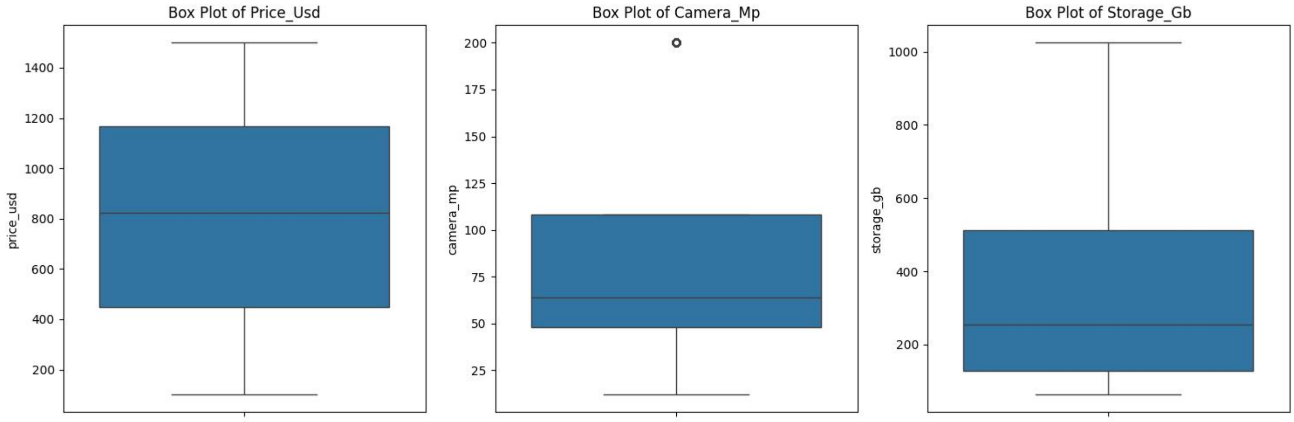
- **Outlier Detection:** The Interquartile Range (IQR) method was used to identify outliers. Specifically, the camera_mp column showed 18.8% of data points as outliers, indicating a niche segment of high-megapixel camera phones.

```
numerical_cols = ['price_usd', 'camera_mp', 'storage_gb', 'ram_gb', 'battery_mah', 'display_size_inch',
'charging_watt', 'rating']

# Function to detect and report outliers using IQR
def detect_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]
    print(f"--- Outliers in {column} (using IQR) ---")
    print(f"Count: {len(outliers)}, Percentage: {len(outliers)/len(data) * 100:.2f}%")
    return outliers, lower_bound, upper_bound

# Report outliers without capping yet, as a high-spec phone isn't an error
for col in numerical_cols:
    detect_outliers_iqr(data, col)
```

<u>OUTPUT</u>

```
--- Outliers in price_usd (using IQR) ---
Count: 0, Percentage: 0.00%
--- Outliers in camera_mp (using IQR) ---
Count: 188, Percentage: 18.80%
--- Outliers in storage_gb (using IQR) ---
Count: 0, Percentage: 0.00%
--- Outliers in ram_gb (using IQR) ---
Count: 0, Percentage: 0.00%
--- Outliers in battery_mah (using IQR) ---
Count: 0, Percentage: 0.00%
--- Outliers in display_size_inch (using IQR) ---
Count: 0, Percentage: 0.00%
--- Outliers in charging_watt (using IQR) ---
Count: 0, Percentage: 0.00%
--- Outliers in rating (using IQR) ---
Count: 0, Percentage: 0.00%
```

```
# Plot boxplots for a visual check (Price, Camera MP, Storage)
plt.figure(figsize=(15, 5))
for i, col in enumerate(['price_usd', 'camera_mp', 'storage_gb'], 1):
    plt.subplot(1, 3, i)
    sns.boxplot(y=data[col])
    plt.title(f'Box Plot of {col.title()}')

plt.tight_layout()
plt.show()
```



7. Derived Metrics

To deepen the analysis, two key features were engineered:

- **Price Segment (**price_segment**):** Phones were binned into market categories:
  - Budget (<$400)
  - Mid-Range ($400-$800)
  - Upper Mid-Range ($800-$1200)
  - Flagship (>$1200)

```
# derived metrics
# 1. Price_Segment
price_bins = [0, 400, 800, 1200, np.inf]
price_labels = ['Budget', 'Mid-Range', 'Upper Mid-Range', 'Flagship']
data['price_segment'] = pd.cut(data['price_usd'], bins=price_bins, labels=price_labels, right=False)
```

	brand	Model	model_no	processor	ram_gb	storage_gb	camera_mp	battery_mah	price_usd	year	display_size_inch	charging_watt	5g_support	os	rating	price_segment
0	Oppo	A98	111	Helio G99	16	128	108	6000	855	2025	6.6	33	Yes	Android	3.8	Upper Mid-Range
1	Realme	11 Pro+	843	Tensor G4	6	128	64	4500	618	2025	6.9	100	Yes	Android	4.4	Mid-Range
2	Xiaomi	Redmi Note 14 Pro	461	A18 Pro	16	64	64	4000	258	2025	6.8	44	Yes	Android	4.1	Budget
3	Vivo	V29e	744	Exynos 2400	6	512	48	4500	837	2025	6.0	65	Yes	Android	4.1	Upper Mid-Range
4	Apple	iPhone 16 Pro Max	927	Dimensity 9300	12	128	200	5000	335	2025	6.9	100	Yes	iOS	3.5	Budget

- **Spec Value (**spec_value**):** A calculated metric representing "Value for Money."
  - *Formula:* (Rating * (RAM + Storage/100 + Battery/1000 + Camera/100)) / Price
  - This normalizes specs to a similar scale and divides by price to find efficiency.

```
# 2. Spec_Value (Value for money)
# A composite score: (Rating * (RAM + Storage_GB/100 + Battery_mAh/1000 + Camera_MP/100)) / Price_USD
# The denominators (100, 1000, 100) are for rough normalization to bring these features into a similar range
as RAM (4-16)
data['spec_value'] = (
    data['rating'] * (
        data['ram_gb'] +
        (data['storage_gb'] / 100) +
        (data['battery_mah'] / 1000) +
        (data['camera_mp'] / 100)))/ data['price_usd']
```

	brand	model	price_usd	ram_gb	storage_gb	camera_mp	battery_mah	display_size_inch	charging_watt	5g_support	os	processor	rating	release_month	year	Model	model_no	price_segment	spec_value
0	Oppo	A98 111	855	16	128	108	6000	6.6	33	Yes	Android	Helio G99	3.8	February	2025	A98	111	Upper Mid-Range	0.108267
1	Realme	11 Pro+ 843	618	6	128	64	4500	6.9	100	Yes	Android	Tensor G4	4.4	August	2025	11 Pro+	843	Mid-Range	0.088427
2	Xiaomi	Redmi Note 14 Pro 461	258	16	64	64	4000	6.8	44	Yes	Android	A18 Pro	4.1	March	2025	Redmi Note 14 Pro	461	Budget	0.338171
3	Vivo	V29E 744	837	6	512	48	4500	6.0	65	Yes	Android	Exynos 2400	4.1	August	2025	V29e	744	Upper Mid-Range	0.078865
4	Apple	Iphone 16 Pro Max 927	335	12	128	200	5000	6.9	100	Yes	iOS	Dimensity 9300	3.5	February	2025	iPhone 16 Pro Max	927	Budget	0.211881

8. Filtering Data

Data subsets were managed as follows:

- **Column Filtering:** The redundant original `model` column was dropped after splitting.

```
data.drop(columns=['model'], inplace=True)

data.head()
```

	brand	price_usd	ram_gb	storage_gb	camera_mp	battery_mah	display_size_inch	charging_watt	5g_support	os	processor	rating	release_month	year	Model	model_no
0	Oppo	855	16	128	108	6000	6.6	33	Yes	Android	Helio G99	3.8	February	2025	A98	111
1	Realme	618	6	128	64	4500	6.9	100	Yes	Android	Tensor G4	4.4	August	2025	11 Pro+	843
2	Xiaomi	258	16	64	64	4000	6.8	44	Yes	Android	A18 Pro	4.1	March	2025	Redmi Note 14 Pro	461
3	Vivo	837	6	512	48	4500	6.0	65	Yes	Android	Exynos 2400	4.1	August	2025	V29e	744
4	Apple	335	12	128	200	5000	6.9	100	Yes	iOS	Dimensity 9300	3.5	February	2025	iPhone 16 Pro Max	927

- **Reordering:** Columns were reordered to place identifiers (Brand, Model) first, followed by specs and price, facilitating easier manual inspection.

```
column_orders = ['brand', 'Model', 'model_no', 'processor', 'ram_gb', 'storage_gb', 'camera_mp', 'battery_mah',
'price_usd','year','display_size_inch','charging_watt','5g_support','os','rating']
data = data[column_orders]
data.head()
```

	brand	Model	model_no	processor	ram_gb	storage_gb	camera_mp	battery_mah	price_usd	year	display_size_inch	charging_watt	5g_support	os	rating
0	Oppo	A98	111	Helio G99	16	128	108	6000	855	2025	6.6	33	Yes	Android	3.8
1	Realme	11 Pro+	843	Tensor G4	6	128	64	4500	618	2025	6.9	100	Yes	Android	4.4
2	Xiaomi	Redmi Note 14 Pro	461	A18 Pro	16	64	64	4000	258	2025	6.8	44	Yes	Android	4.1
3	Vivo	V29e	744	Exynos 2400	6	512	48	4500	837	2025	6.0	65	Yes	Android	4.1
4	Apple	iPhone 16 Pro Max	927	Dimensity 9300	12	128	200	5000	335	2025	6.9	100	Yes	iOS	3.5

9. Statistical Analysis

- **Descriptive Statistics:** Summary statistics (Mean, Median, Std Dev) were generated for all numerical columns.
  - *Key Finding:* The average phone price in the dataset is approximately $813.

```
data.describe()
```

	ram_gb	storage_gb	camera_mp	battery_mah	price_usd	year	display_size_inch	charging_watt	rating
count	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	1000.0	1000.000000	1000.000000	1000.000000
mean	9.17200	402.880000	83.534000	5012.000000	813.478000	2025.0	6.380600	63.791000	4.229900
std	4.32633	349.405893	62.504958	711.591429	411.708367	0.0	0.496841	36.333751	0.439965
min	4.00000	64.000000	12.000000	4000.000000	101.000000	2025.0	5.500000	18.000000	3.500000
25%	6.00000	128.000000	48.000000	4500.000000	449.250000	2025.0	6.000000	33.000000	3.800000
50%	8.00000	256.000000	64.000000	5000.000000	822.000000	2025.0	6.400000	65.000000	4.200000
75%	12.00000	512.000000	108.000000	5500.000000	1166.250000	2025.0	6.800000	100.000000	4.600000
max	16.00000	1024.000000	200.000000	6000.000000	1499.000000	2025.0	7.200000	120.000000	5.000000

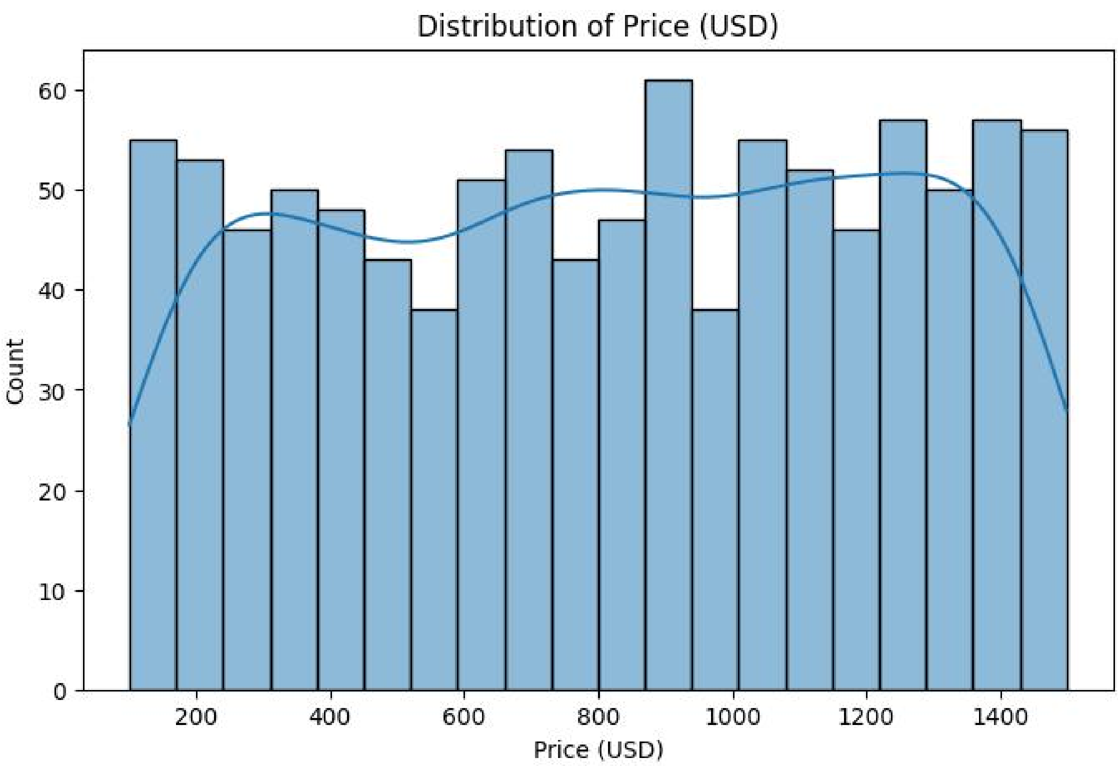
- **Correlation Analysis:** A correlation matrix was calculated to quantify the linear relationships between hardware specs (RAM, Storage, Battery) and Price.

10. EDA (Univariate, Bivariate, Multivariate)

Visualizations were used to explore the data:

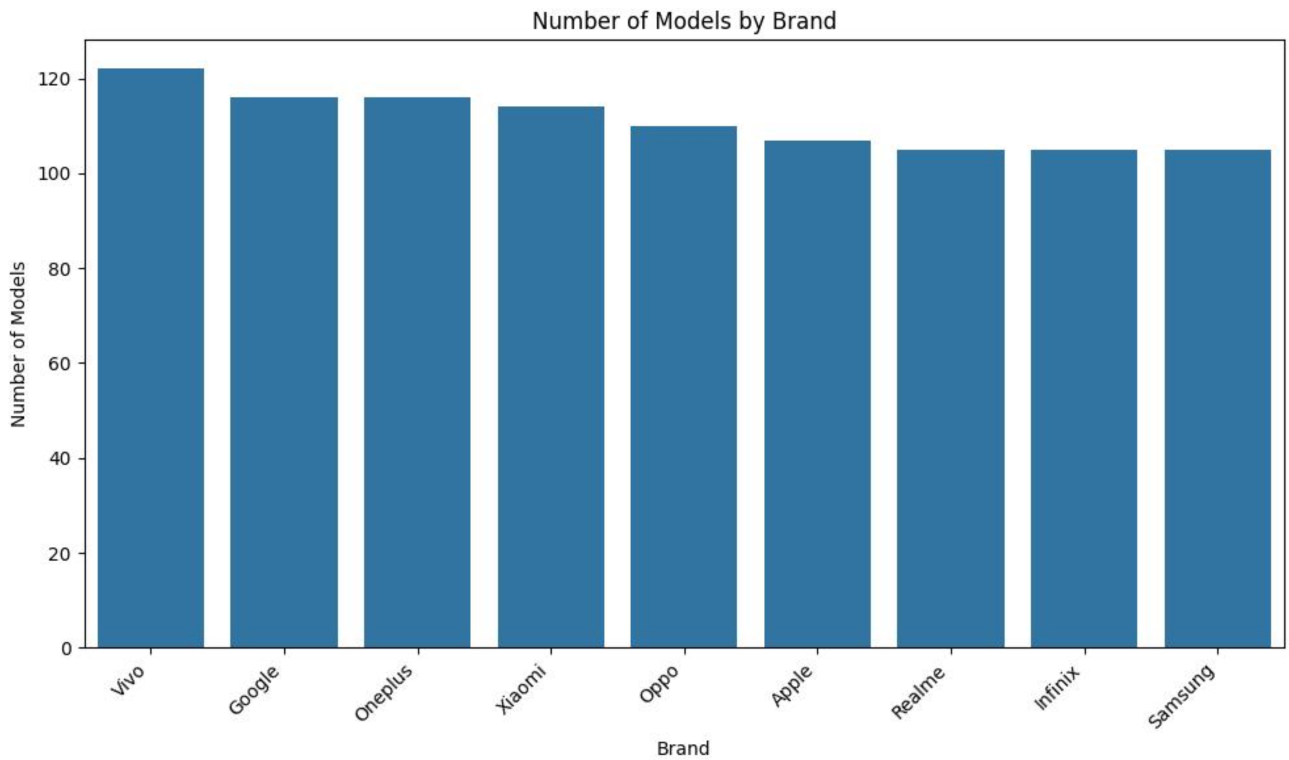
- **Univariate Analysis:**
  - *Price Distribution:* Histogram showing the frequency of different price points.

```
# Univariate Analysis and Visualize
# 1. Distribution of price_usd (Histogram)
plt.figure(figsize=(8, 5))
sns.histplot(data['price_usd'], kde=True, bins=20)
plt.title('Distribution of Price (USD)')
plt.xlabel('Price (USD)')
plt.ylabel('Count')
plt.show()
```



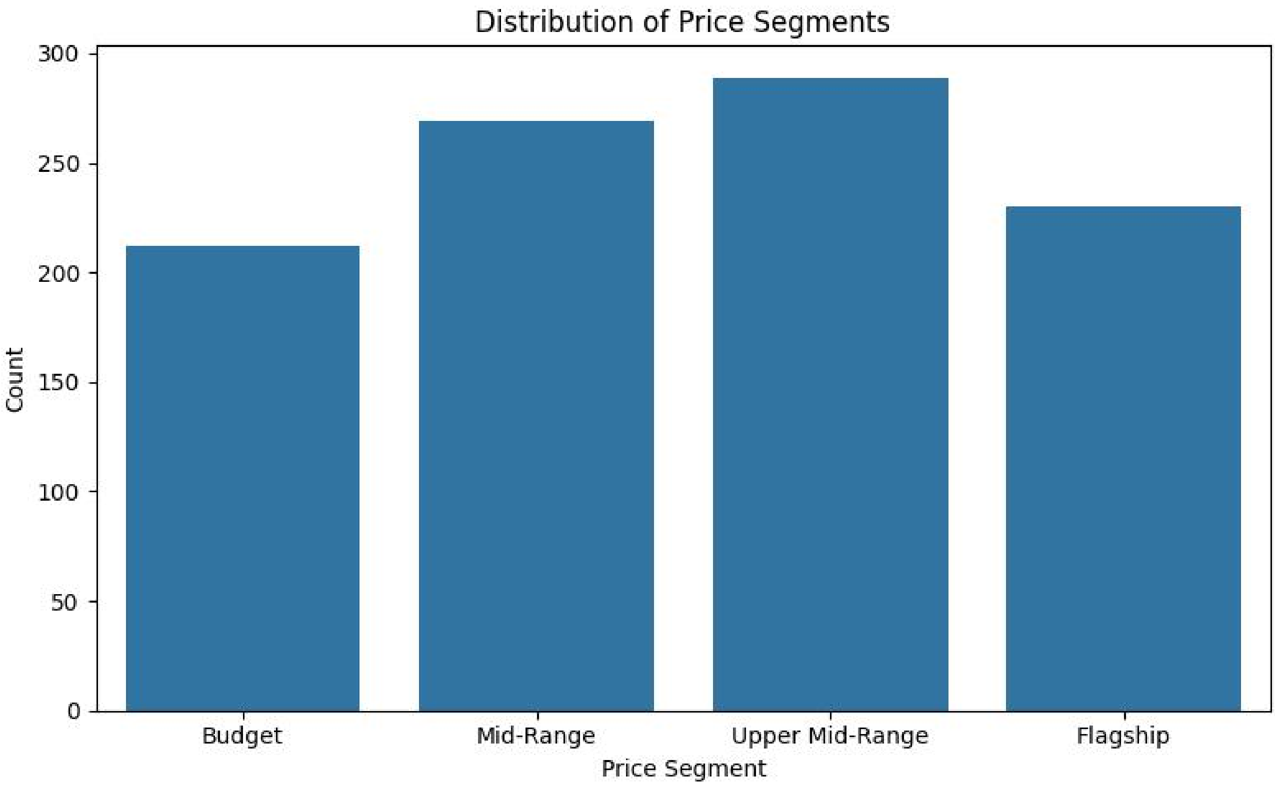
- *Brand Counts:* Bar chart showing the volume of models per brand.

```
# 2. Frequency of brand (Bar plot)
brand_counts = data['brand'].value_counts().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=brand_counts.index, y=brand_counts.values, )
plt.title('Number of Models by Brand')
plt.xlabel('Brand')
plt.ylabel('Number of Models')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



- *Segment Counts:* Bar chart showing the distribution of phones across Price Segments.

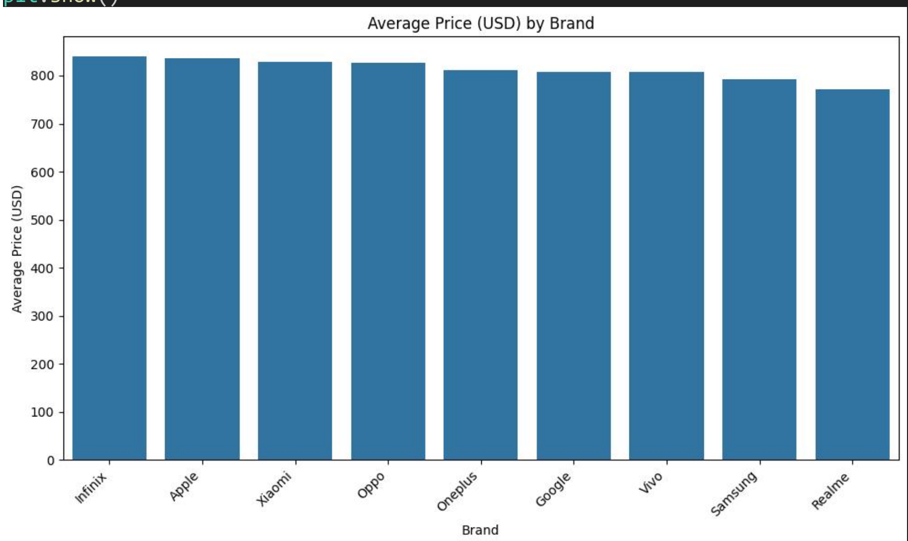
```
# 3. Frequency of price_segment (Bar plot)
segment_counts = data['price_segment'].value_counts()
# Order the segments correctly
ordered_segments = ['Budget', 'Mid-Range', 'Upper Mid-Range', 'Flagship']
segment_counts = segment_counts.reindex(ordered_segments)

plt.figure(figsize=(8, 5))
sns.barplot(x=segment_counts.index, y=segment_counts.values,)
plt.title('Distribution of Price Segments')
plt.xlabel('Price Segment')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



Bivariate Analysis:

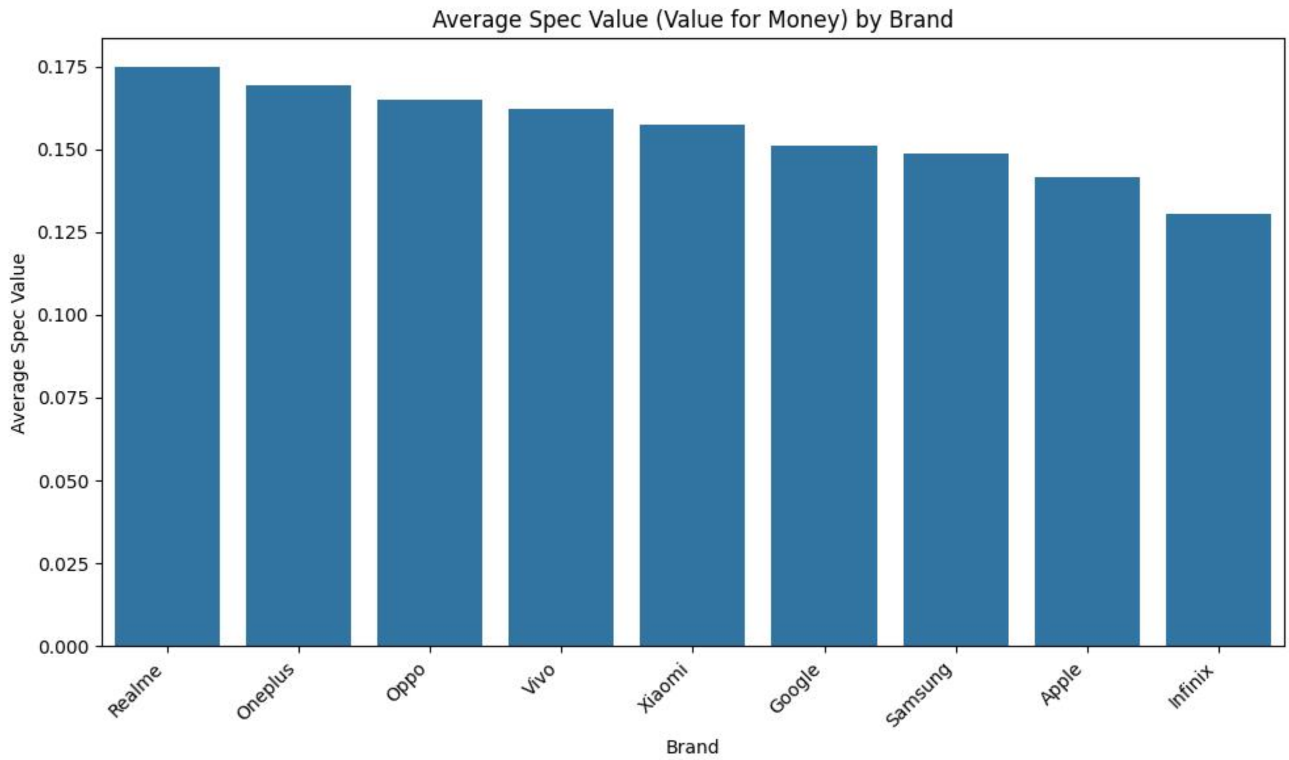
- *Price by Brand:* Bar chart comparing average pricing strategies.

```
# Average price_usd by brand (Bar plot)
avg_price_brand = data.groupby('brand')['price_usd'].mean().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_price_brand.index, y=avg_price_brand.values,)
plt.title('Average Price (USD) by Brand')
plt.xlabel('Brand')
plt.ylabel('Average Price (USD)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



- *Value by Brand:* Bar chart identifying brands with the highest average spec_value.

```
avg_spec_value_brand = data.groupby('brand')['spec_value'].mean().sort_values(ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x=avg_spec_value_brand.index, y=avg_spec_value_brand.values)
plt.title('Average Spec Value (Value for Money) by Brand')
plt.xlabel('Brand')
plt.ylabel('Average Spec Value')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



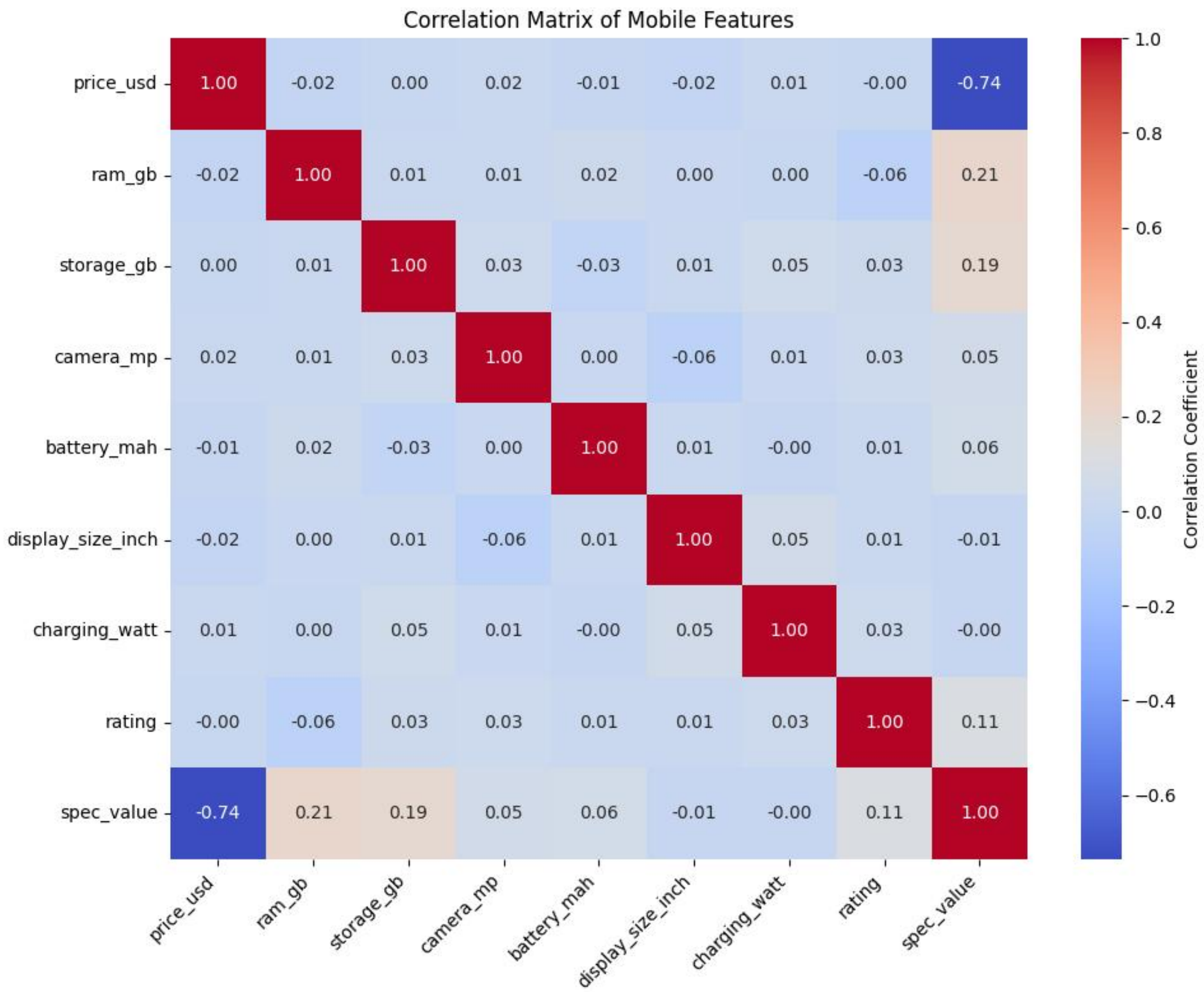
- **Multivariate Analysis:**

  - *Price vs. Rating by OS:* Scatter plot analyzing how Price and OS impact user ratings.

```
# 1. Relationship between price_usd and rating (Scatter plot)
plt.figure(figsize=(8, 6))
sns.scatterplot(x=data['price_usd'], y=data['rating'], alpha=0.6, hue=data['os'])
plt.title('Price vs. Rating by OS')
plt.xlabel('Price (USD)')
plt.ylabel('User Rating')
plt.legend(title='OS')
plt.grid(True, linestyle='--')
plt.show()
```

- *Feature Correlation:* Heatmap displaying relationships between all numerical variables.

```
corr_cols = ['price_usd', 'ram_gb', 'storage_gb', 'camera_mp', 'battery_mah', 'display_size_inch',
'charging_watt', 'rating', 'spec_value']
correlation_matrix = data[corr_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', cbar_kws={'label': 'Correlation
Coefficient'})
plt.title('Correlation Matrix of Mobile Features')
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



11. Insights

- **Market Positioning:** Certain brands dominate the "Flagship" high-price tier, while others focus heavily on volume in the "Budget" and "Mid-Range" sectors.
- **Spec-Price Relationship:** There is a positive correlation between Price and features like RAM and Storage, but the correlation is not perfectly linear, indicating brand premium plays a role.
- **Value Leaders:** The spec_value analysis reveals that specific brands (often not the most expensive ones) offer significantly higher raw specifications per dollar spent.
- **Camera Trends:** The camera megapixel distribution is right-skewed, with a distinct group of "camera-centric" phones pushing the upper limits of the market.

12. Conclusion

The analysis of the 2025 Global Mobile Market reveals a diverse landscape where price does not always equate to raw specification value. While premium brands command higher prices, mid-range competitors often provide superior "spec-per-dollar" value.

- **Recommendation for Consumers:** Buyers prioritizing pure hardware performance should look towards high `spec_value` brands identified in the mid-range segment.
- **Next Steps:** Future analysis could benefit from incorporating regional sales data to weight popularity against technical specifications.