AWP Project

Created analysis of a dataset containing information about various smartphone models and their features using Python. The dataset includes attributes such as brand name, model, price, rating, 5G capability, NFC support, processor brand, RAM capacity, battery capacity, screen size, camera specifications, operating system, and more.

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

numpy (imported as np): A powerful library for numerical computing. It provides support for arrays and matrices, along with a collection of mathematical functions to operate on these data structures.

pandas (imported as pd): A library for data manipulation and analysis. It provides data structures like DataFrame for handling tabular data, allowing for easy data cleaning, filtering, and analysis.

matplotlib.pyplot (imported as plt): A plotting library used for creating static, interactive, and animated visualizations in Python. It provides a wide range of plotting functions to create graphs and charts.

seaborn (imported as sns): A data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Preprocessing of Data:-

```
df1 = pd.read_csv('C:/Users/Windows/OneDrive/Documents/data_scienc/PYTHON/AWP/AWP PROJECT/3_smartphones_data.csv')
```

```
df1.head()
```

	brand_name	model	price	rating	has_5G	has_nfc	has_IR_blaster	processor_brand	num_cores	processor_speed	...	screen_size	resolution	refresh_rate	num_re
0	oneplus	OnePlus 11 5G	54999	89.0	True	True	False	snapdragon	8	3.2	...	6.70	1440 x 3216	120.0	
1	oneplus	OnePlus Nord CE 2 Lite 5G	19989	81.0	True	False	False	snapdragon	8	2.2	...	6.59	1080 x 2412	120.0	
2	samsung	Samsung Galaxy A14 5G	16499	75.0	True	False	False	exynos	8	2.4	...	6.60	1080 x 2408	90.0	
3	motorola	Motorola Moto G62 5G	14999	81.0	True	False	False	snapdragon	8	2.2	...	6.55	1080 x 2400	120.0	
4	realme	Realme 10 Pro Plus	24999	82.0	True	False	False	dimensity	8	2.6	...	6.70	1080 x 2412	120.0	

Activate Win
Go to Settings to

Above code loads the smartphone data from the specified CSV file into a pandas DataFrame named df1.

And df1.head() prints first 5 observations

```
df1.shape
```

```
(980, 25)
```

Above code shows the number of rows and columns in the DataFrame, ie. 980 rows and 25 columns.

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 980 entries, 0 to 979
Data columns (total 25 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   brand_name                 980 non-null    object
 1   model                      980 non-null    object
 2   price                      980 non-null    int64
 3   rating                     879 non-null    float64
 4   has_5G                     980 non-null    bool
 5   has_nfc                    980 non-null    bool
 6   has_IR_blaster             980 non-null    bool
 7   processor_brand            980 non-null    object
 8   num_cores                  980 non-null    int64
 9   processor_speed            938 non-null    float64
 10  ram_capacity               980 non-null    float64
 11  internal_memory            980 non-null    float64
 12  battery_capacity           969 non-null    float64
 13  fast_charging_available    980 non-null    int64
 14  fast_charging              769 non-null    float64
 15  screen_size                980 non-null    float64
 16  resolution                 980 non-null    object
 17  refresh_rate               980 non-null    float64
 18  num_rear_cameras           980 non-null    int64
 19  num_front_cameras          976 non-null    float64
 20  primary_rear_camera        980 non-null    float64
 21  primary_front_camera       975 non-null    float64
 22  extended_memory_available  980 non-null    int64
 23  os                         980 non-null    object
 24  extended_upto              507 non-null    float64
dtypes: bool(3), float64(12), int64(5), object(5)
memory usage: 171.4+ KB
```

Above code provides summary of the DataFrame df1.

The number of rows

The number of columns

Column names

Data types of each column

The number of non-null (non-missing) values in each column

Memory usage of the DataFrame

```
df1.describe()
```

	price	rating	num_cores	processor_speed	ram_capacity	internal_memory	battery_capacity	fast_charging_available	fast_charging	screen_size	refr
count	980.000000	879.000000	980.000000	938.000000	980.000000	980.000000	969.000000	980.000000	769.000000	980.000000	98(
mean	32520.504082	78.258248	7.761224	2.427217	6.560204	141.036735	4817.748194	0.854082	46.129649	6.536765	92
std	39531.812669	7.402854	0.845662	0.464090	2.744378	107.134516	1009.540054	0.353205	34.275769	0.349162	28
min	3499.000000	60.000000	4.000000	1.200000	1.000000	8.000000	1821.000000	0.000000	10.000000	3.540000	6(
25%	12999.000000	74.000000	8.000000	2.050000	4.000000	64.000000	4500.000000	1.000000	18.000000	6.500000	6(
50%	19994.500000	80.000000	8.000000	2.300000	6.000000	128.000000	5000.000000	1.000000	33.000000	6.580000	9(
75%	35491.500000	84.000000	8.000000	2.840000	8.000000	128.000000	5000.000000	1.000000	66.000000	6.670000	12(
max	650000.000000	89.000000	8.000000	3.220000	18.000000	1024.000000	22000.000000	1.000000	240.000000	8.030000	24(

Activate Wi

Above code generates descriptive statistics of the DataFrame df1.

Count

Mean

Standard Deviation

Min and Max

25th, 50th (median), and 75th percentiles

```
]: df1.isnull().sum()
```

```
]: brand_name                0
   model                     0
   price                     0
   rating                  101
   has_5G                    0
   has_nfc                   0
   has_IR_blaster            0
   processor_brand           0
   num_cores                 0
   processor_speed          42
   ram_capacity              0
   internal_memory           0
   battery_capacity         11
   fast_charging_available   0
   fast_charging           211
   screen_size               0
   resolution                0
   refresh_rate              0
   num_rear_cameras          0
   num_front_cameras         4
   primary_rear_camera       0
   primary_front_camera      5
   extended_memory_available 0
   os                        0
   extended_upto           473
   dtype: int64
```

Above code checks for missing (null) values in the DataFrame df1 and returns the total count of missing values for each column.

To handel missing values:-

```
df1['rating'] = df1['rating'].fillna(0)
df1['processor_speed'] = df1['processor_speed'].fillna(df1['processor_speed'].median())
df1['battery_capacity'] = df1['battery_capacity'].fillna(df1['battery_capacity'].mean())
df1['fast_charging'] = df1['fast_charging'].fillna(df1['fast_charging'].mean())
df1['extended_upto'] = df1['extended_upto'].fillna(0)
df1['primary_front_camera'] = df1['primary_front_camera'].fillna(0)
df1['num_front_cameras'] = df1['num_front_cameras'].fillna(0)
```

df1['rating'] = df1['rating'].fillna(0): Replaces null values in the rating column with 0.

df1['processor_speed'] = df1['processor_speed'].fillna(df1['processor_speed'].median()): Replaces null values in the processor_speed column with the median value of that column.

df1['battery_capacity'] = df1['battery_capacity'].fillna(df1['battery_capacity'].mean()): Replaces null values in the battery_capacity column with the mean value of that column.

df1['fast_charging'] = df1['fast_charging'].fillna(df1['fast_charging'].mean()): Replaces null values in the fast_charging column with the mean value of that column.

df1['extended_upto'] = df1['extended_upto'].fillna(df1['extended_upto'].mean()): Replaces null values in the extended_upto column with 0.

df1['primary_front_camera'] = df1['primary_front_camera'].fillna(0): Replaces null values in the primary_front_camera column with 0.

df1['num_front_cameras'] = df1['num_front_cameras'].fillna(0): Replaces null values in the num_front_cameras column with 0.

Now it all null values are removed.

```
df1.isnull().sum()
```

```
brand_name                  0
model                       0
price                       0
rating                      0
has_5G                      0
has_nfc                     0
has_IR_blaster              0
processor_brand             0
num_cores                   0
processor_speed             0
ram_capacity                0
internal_memory             0
battery_capacity            0
fast_charging_available     0
fast_charging               0
screen_size                 0
resolution                  0
refresh_rate                0
num_rear_cameras            0
num_front_cameras           0
primary_rear_camera         0
primary_front_camera        0
extended_memory_available   0
os                          0
extended_upto               0
dtype: int64
```

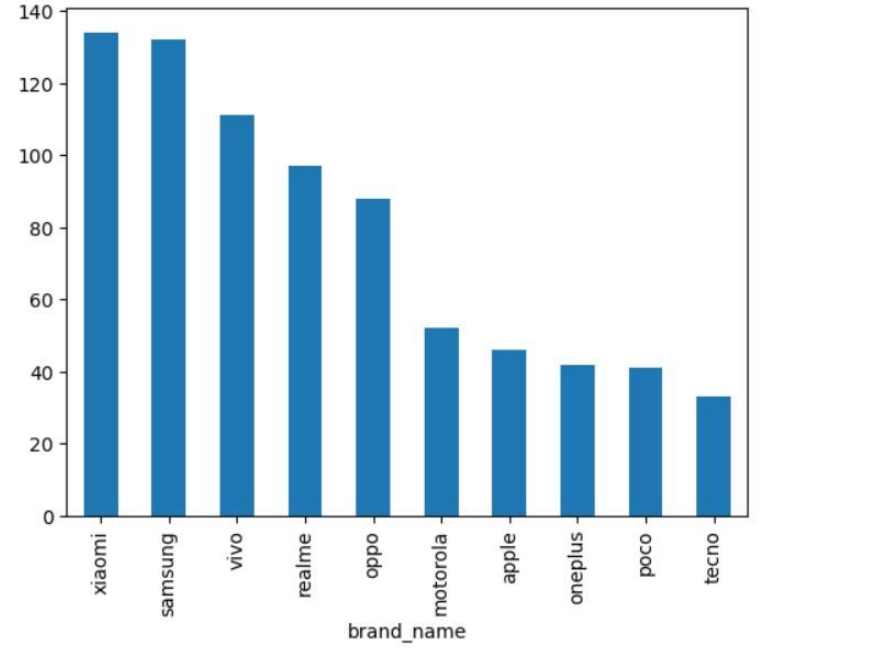
For all columns null value count is zero

All preprocessing of data is done.

Count Analysis:-

```
df1['brand_name'].value_counts().head(10).plot(kind='bar')
```

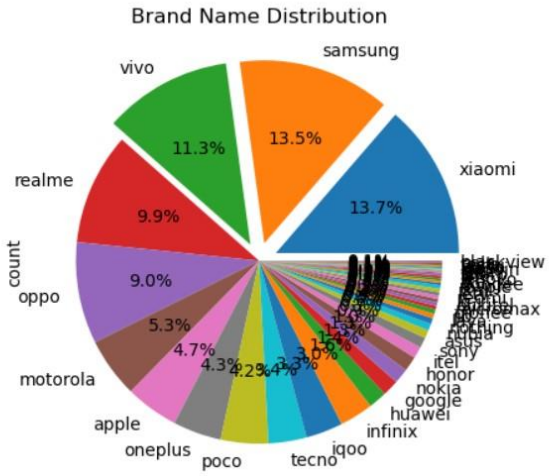
```
<Axes: xlabel='brand_name'>
```



df1['brand_name'].value_counts(): Counts the unique value in the brand_name column and returns counts. head(10): Selects the top 10 brand names.

plot(kind='bar'): Creates a bar plot. It is method of a pandas DataFrame we can specify the type of plot you want to create. We can use barh, pie, histetc.

```
[201]:
brand_counts = df1['brand_name'].value_counts()

explode = [0.1 if val in brand_counts.nlargest(3).values else 0 for val in brand_counts]

brand_counts.plot(kind='pie', autopct='%0.1f%%', explode=explode)
plt.title('Brand Name Distribution')
plt.show()
```



Above code generates a pie chart showing the proportion of each smartphone brand in the brand_name column. The autopct='%0.1f%%' displays the percentage on each slice. Maximum 3 Brand percentage slices are exploded by 0.1.

```
df1['price'].skew()
```

```
6.591790999665567
```

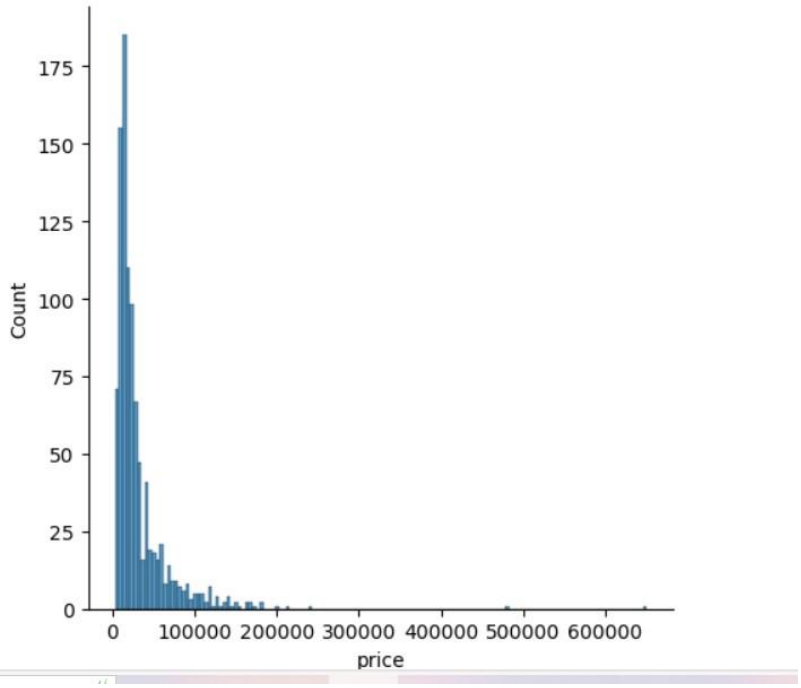
Above code calculates the skewness of the price column in the DataFrame df1.

Skewness measures the asymmetry of the distribution of values.

A positive value indicates a right-skewed distribution, while a negative value indicates a left-skewed distribution

```
sns.displot(kind='hist',data=df1,x='price')
```

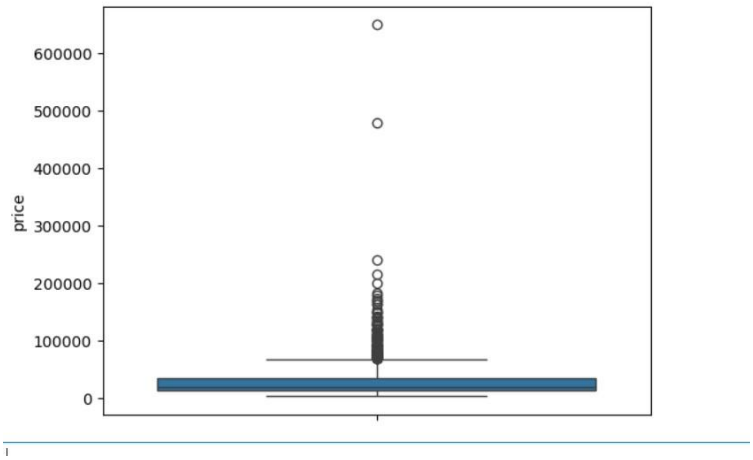
```
<seaborn.axisgrid.FacetGrid at 0x2125b3b20f0>
```



Above code generates a histogram of smartphone prices using Seaborn

```
sns.boxplot(df1['price'])
```

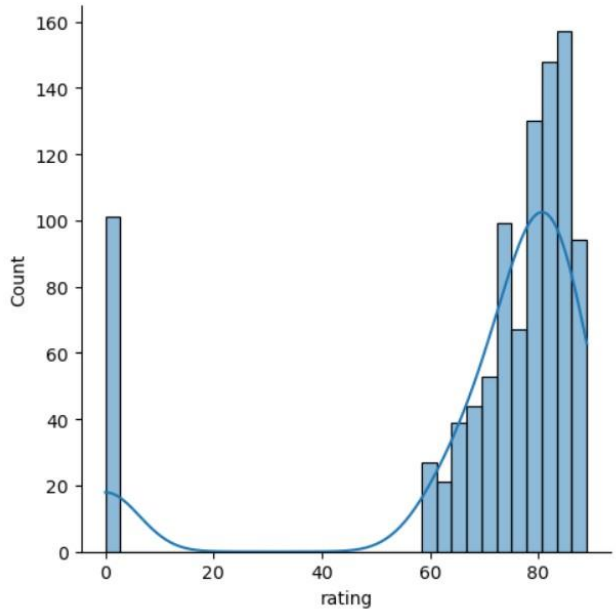
```
<Axes: ylabel='price'>
```



Above code creates a box plot using Seaborn for the price column in the DataFrame df1

```
sns.displot(kind='hist',data=df1,x='rating',kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x2125dd5d6a0>
```

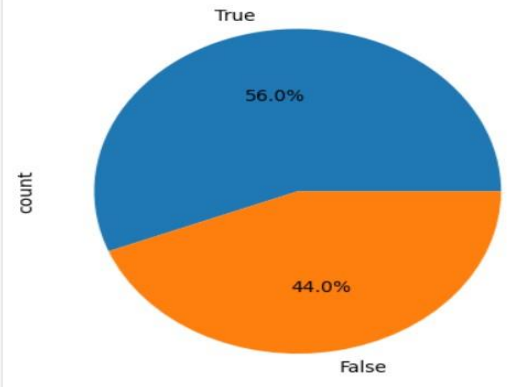


creates a histogram of smartphone ratings with KDE plot using Seaborn.

It shows the distribution of ratings and provides a smoothed curve to represent the probability density function.

```
#5 Has_5g
df1['has_5G'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

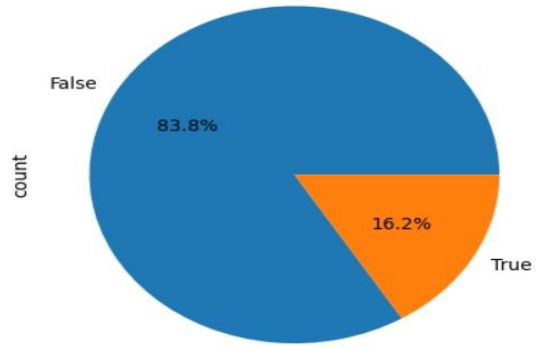
```
<Axes: ylabel='count'>
```



Pie chart showing 5g feaure available or not percentage

```
#5 Has_IR_blaster
df1['has_IR_blaster'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

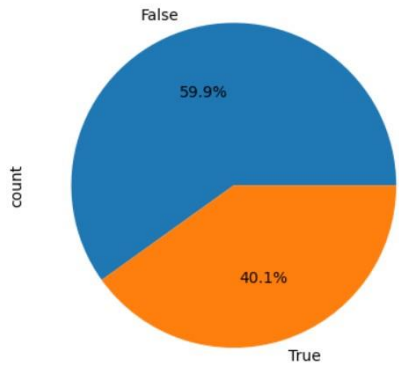
```
<Axes: ylabel='count'>
```



Pie chart showing has IR blaster feature available or not percentage .

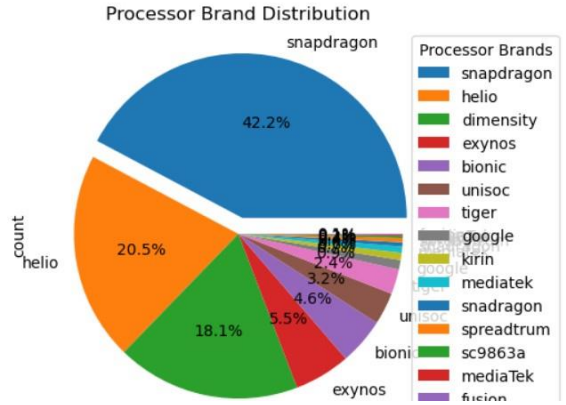
```
df1['has_nfc'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

```
<Axes: ylabel='count'>
```



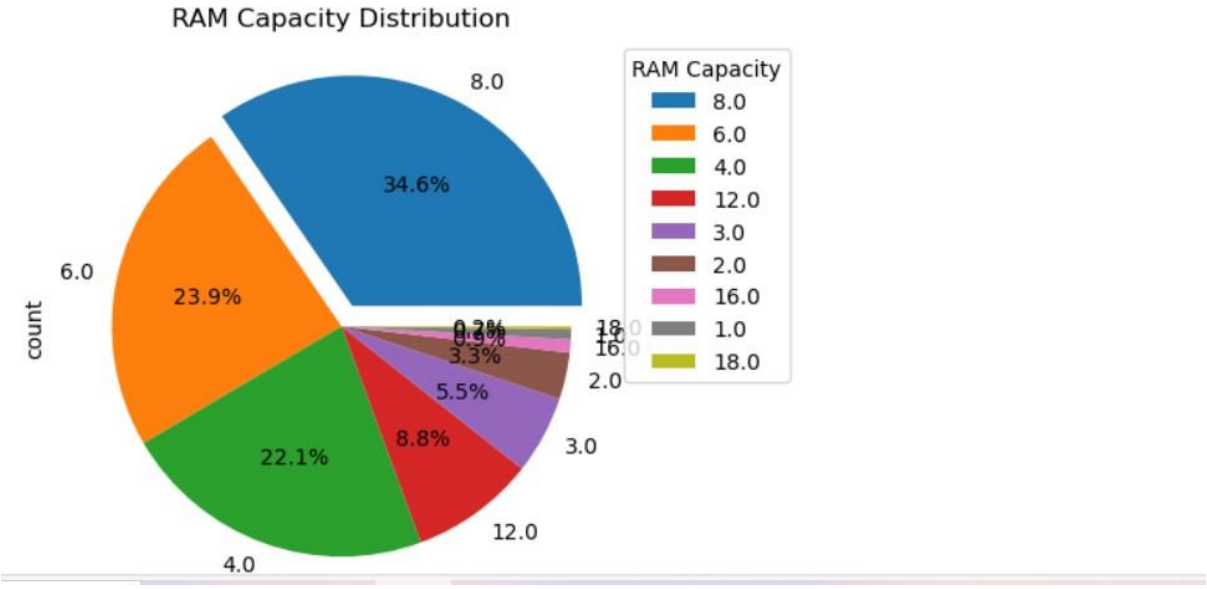
Pie chart showing has NFC feature available or not percentage .

```
processor_counts = df1['processor_brand'].value_counts()


explode = [0.1 if val == processor_counts.max() else 0 for val in processor_counts]

processor_counts.plot(kind='pie', autopct='%0.1f%%', explode=explode)
plt.title('Processor Brand Distribution')

plt.legend(title='Processor Brands', labels=processor_counts.index, loc='upper right', bbox_to_anchor=(1.3, 1))

plt.show()
```



pie chart showing the smartphones with different processor brands.Highest count processor brand is highlighted by exploding

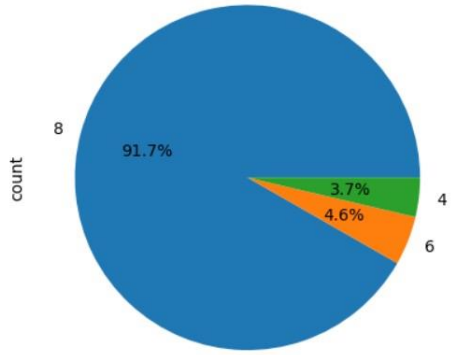
```
ram_counts = df1['ram_capacity'].value_counts()

explode = [0.1 if val == ram_counts.max() else 0 for val in ram_counts]

ram_counts.plot(kind='pie', autopct='%0.1f%%', explode=explode)
plt.title('RAM Capacity Distribution')


plt.legend(title='RAM Capacity', labels=ram_counts.index, loc='upper right', bbox_to_anchor=(1.3, 1))

plt.show()
```



pie chart showing the smartphones with different RAM capacities. Highest count Ram Capacity is highlighted by exploding

```
df1['num_cores'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

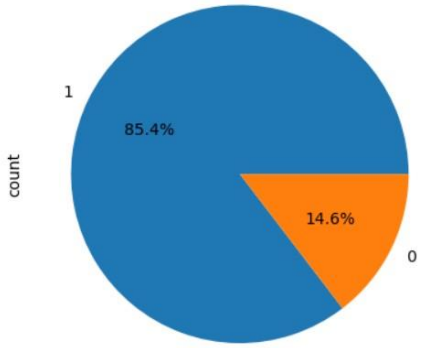
```
<Axes: ylabel='count'>
```



pie chart showing the smartphones with different numbers of processor cores with percentage

```
df1['fast_charging_available'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

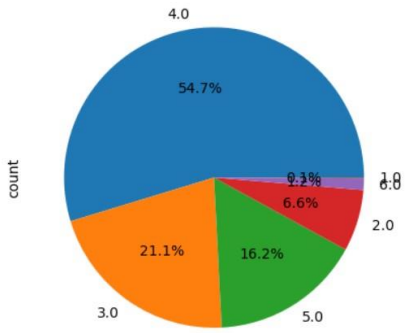
```
<Axes: ylabel='count'>
```



pie chart showing the smartphones with and without fast charging availability with percentage

```
#Cameras
# total cameras
(df1['num_rear_cameras'] + df1['num_front_cameras']).value_counts().plot(kind='pie',autopct='%0.1f%%')
```

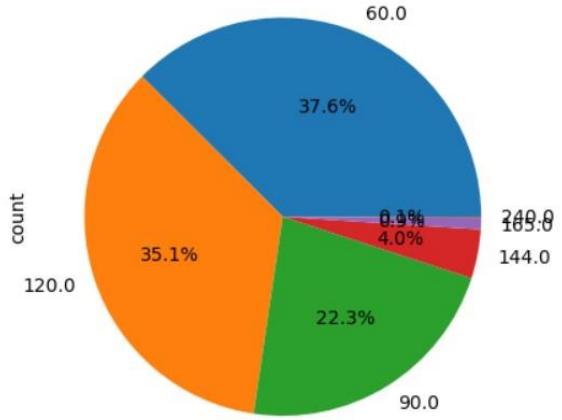
```
<Axes: ylabel='count'>
```



pie chart showing the smartphones with different total numbers of cameras (rear and front combined) with percentage

```
df1['refresh_rate'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

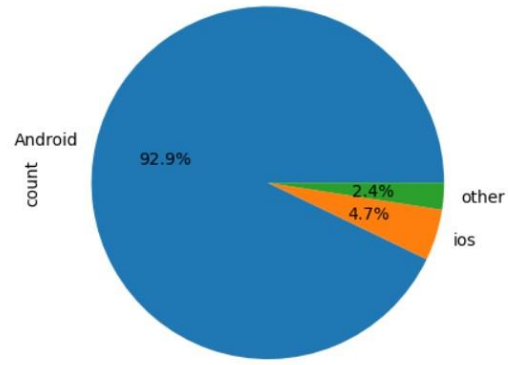
```
<Axes: ylabel='count'>
```



pie chart showing the smartphones with different refresh rates with percentage

```
df1['os'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

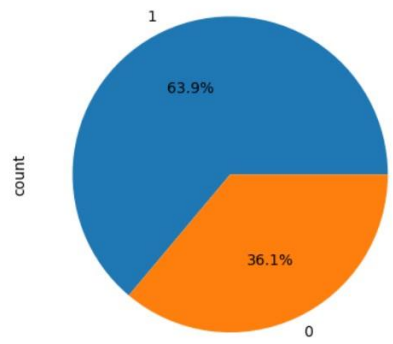
```
<Axes: ylabel='count'>
```



pie chart showing the smartphones with different operating systems with percentage

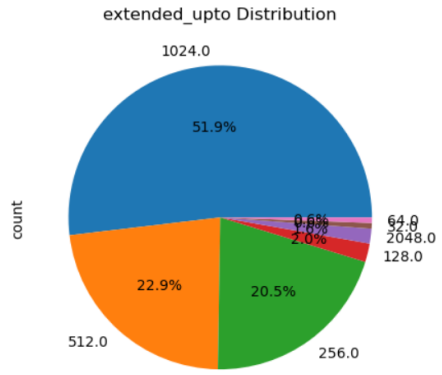
```
]: df1['extended_memory_available'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

```
]: <Axes: ylabel='count'>
```



This code generates a pie chart showing the smartphones with and without extended memory availability with percentage

```
filtered_df = df1[df1['extended_upto'] != 0]
filtered_df['extended_upto'].value_counts().plot(kind='pie',autopct='%0.1f%%')
plt.title('extended_upto Distribution')
plt.show()
#out of the phones which provide the extended memory option near about 50% of them comes up with the op
#23% comes up with option of extension upto 512Gb with 21% provides the option of extension upto 256 Gb
```



pie chart showing the smartphones with different extended memory capacities with percentage

```
def plott(column):
  sns.displot(kind='hist',kde=True, data=df1,x=column, label= column)
  sns.catplot(kind='box',data=df1,x=column)
```

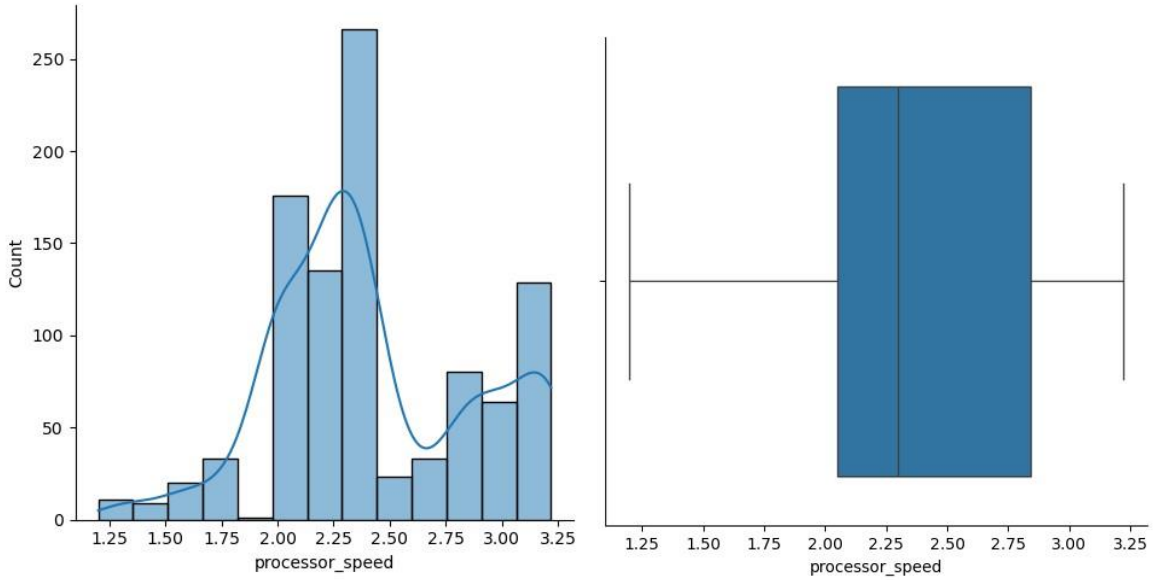
```
columns=df1.select_dtypes(include=['float64','int64']).iloc[:,[3,6,8,9,13,14,16]].columns
columns
```

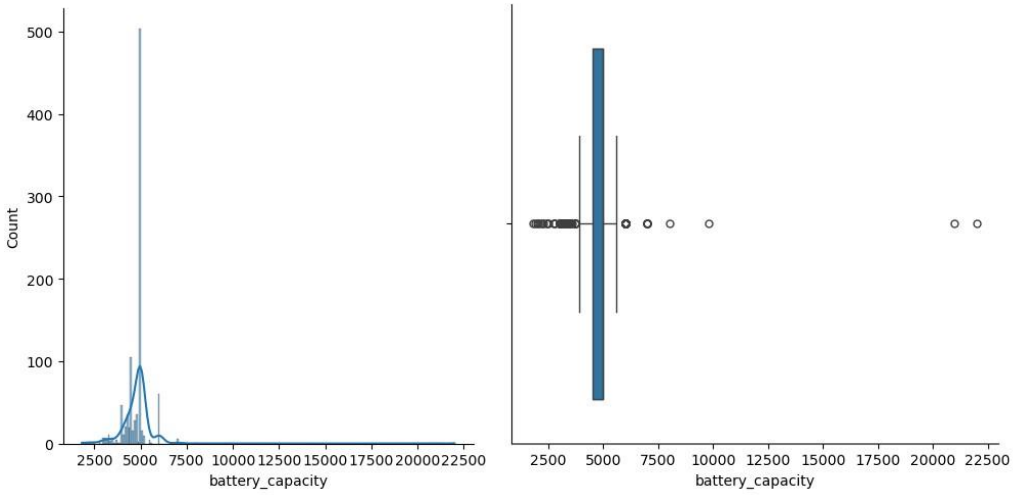
```
Index(['processor_speed', 'battery_capacity', 'fast_charging', 'screen_size',
       'primary_rear_camera', 'primary_front_camera', 'extended_upto'],
      dtype='object')
```

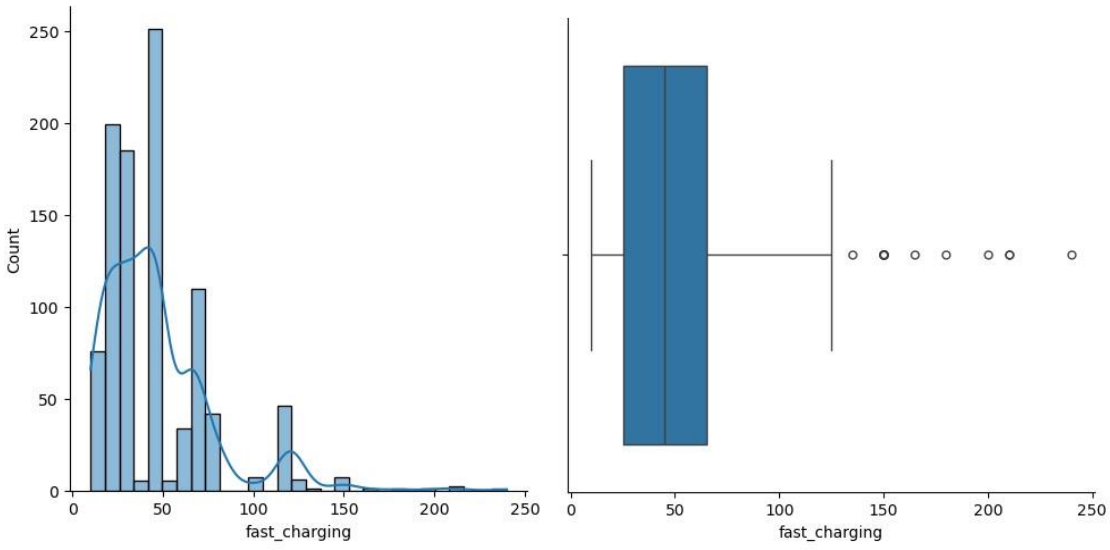
```
for col in columns:
  plott(col)
```

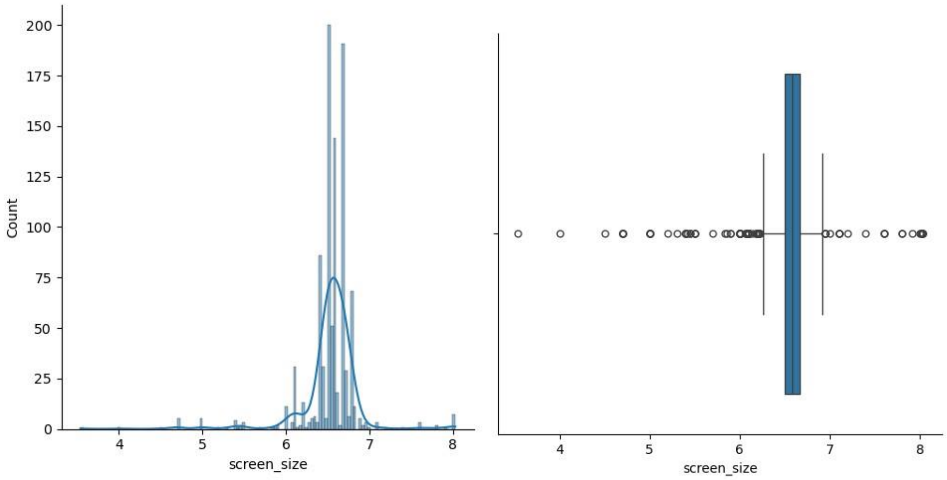
This code create histogram and box plot for selected numeric column in iteration.

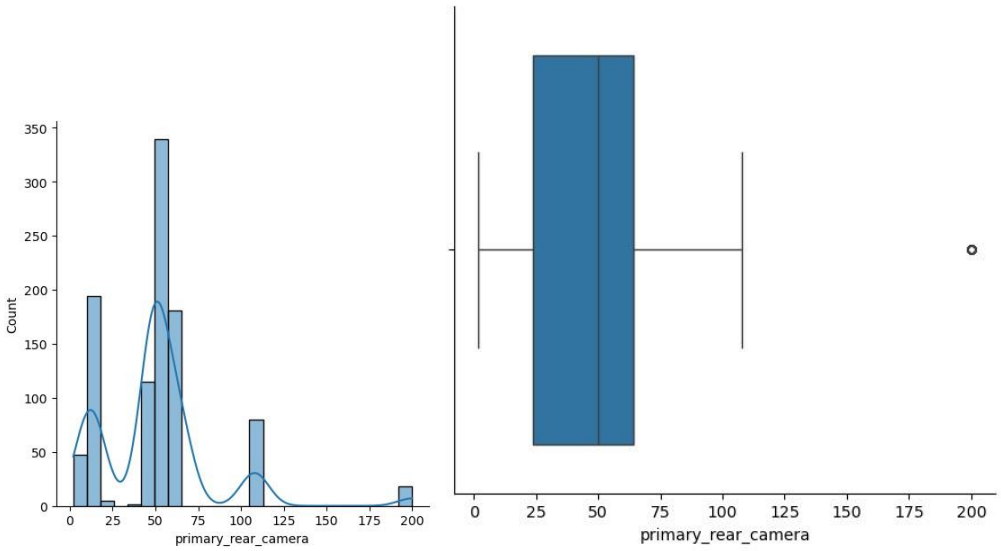
List of numeric column is passed as index of column. And its Histogram and box plot is created.

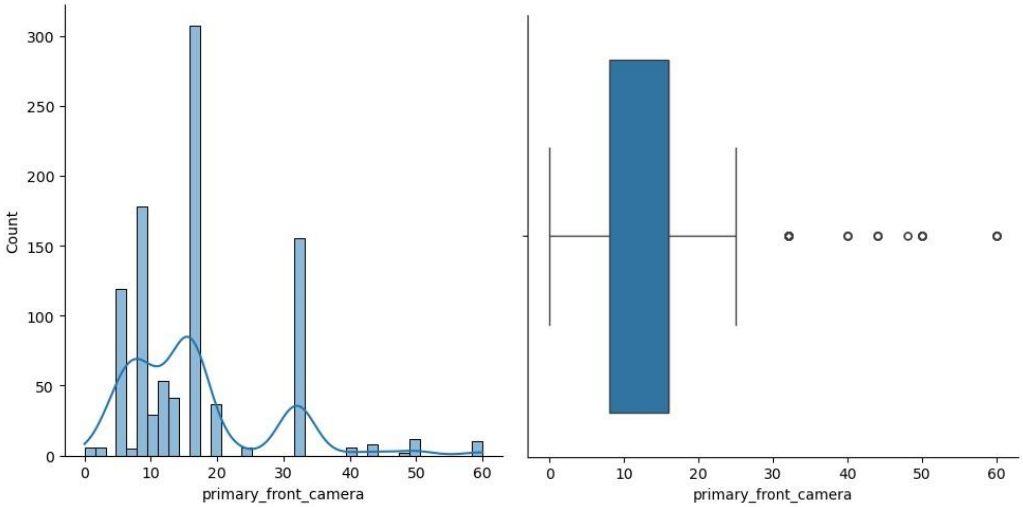

Count
processor_speed
processor_speed

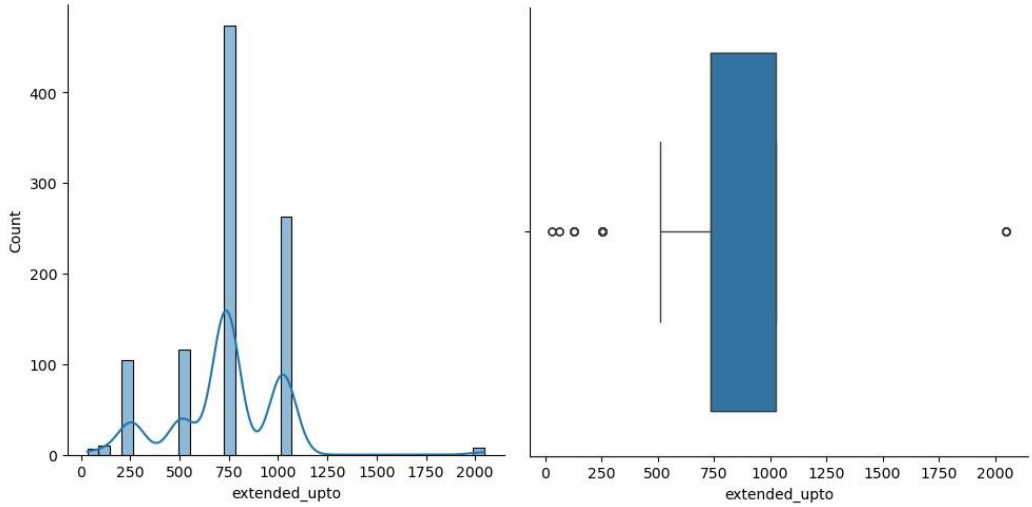

Count
battery_capacity
battery_capacity


Count
fast_charging
fast_charging


Count
screen_size
screen_size

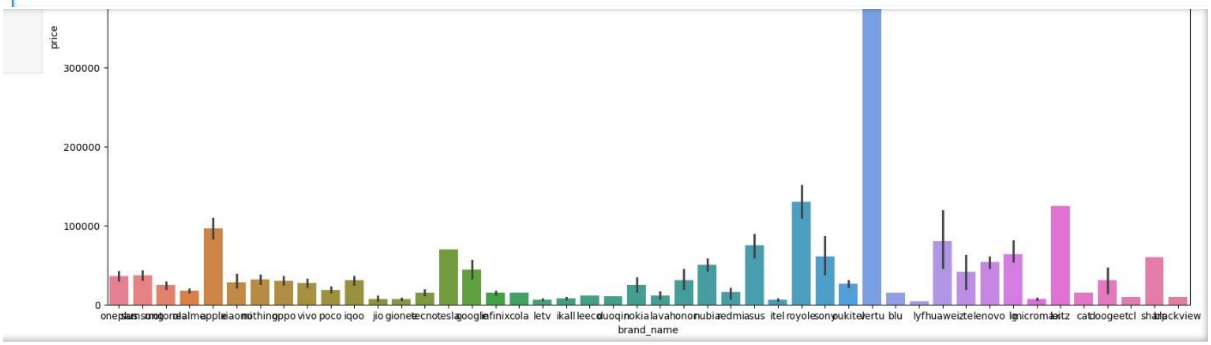

Count
primary_rear_camera
primary_rear_camera


Count
primary_front_camera
primary_front_camera



Price Analysis:-

```
palette = sns.color_palette("husl", len(df1['brand_name'].unique()))

plt.figure(figsize=(20,10))
sns.barplot(data=df1, x='brand_name', y='price', hue='brand_name', palette=palette)
plt.title('Brandname Vs Price')
plt.show()
```



Above code creates a bar plot showing the smartphone brand names and their prices, with each bar colored differently based on the brand. The palette ensures unique colors for each brand.

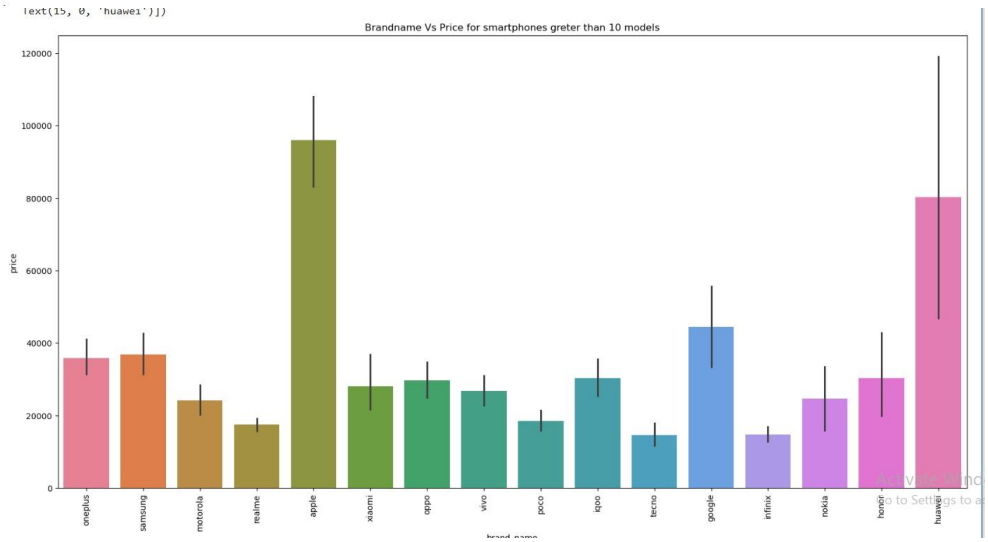
```
x=df1.groupby('brand_name').count()['model']
x
```

```
brand_name
apple         46
asus           7
blackview      1
blu            1
cat            1
cola           1
doogee         2
duoqin         1
gionee         3
google        14
honor         13
huawei        16
ikall          3
infinix       29
iqoo          32
itel          10
jio            4
lava           4
leeco          1
leitz          1
lenovo         2
letv           3
lg             3
lyf            2
micromax       3
motorola      52
nokia         13
```

Groups the DataFrame df1 by the brand_name and counts the number of the model for each brand.

```
temp_df = df1[df1['brand_name'].isin(x[x > 10].index)]
```

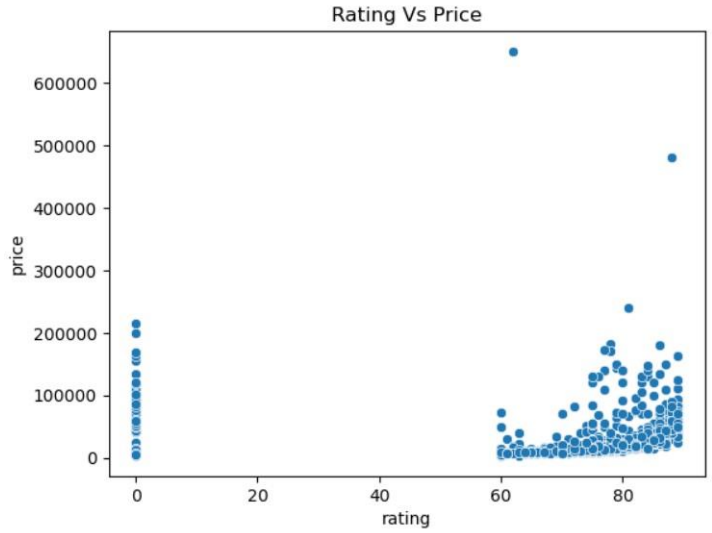
```
palette = sns.color_palette("husl", len(temp_df['brand_name'].unique()))
plt.figure(figsize=(20,10))
sns.barplot(data=temp_df,x='brand_name',y='price', hue='brand_name', palette=palette)
plt.title('Brandname Vs Price for smartphones greter than 10 models')
plt.xticks(rotation='vertical')
```



Above code filters the DataFrame df1 to include only brands with more than 10 models and creates a bar plot of brand names versus prices.

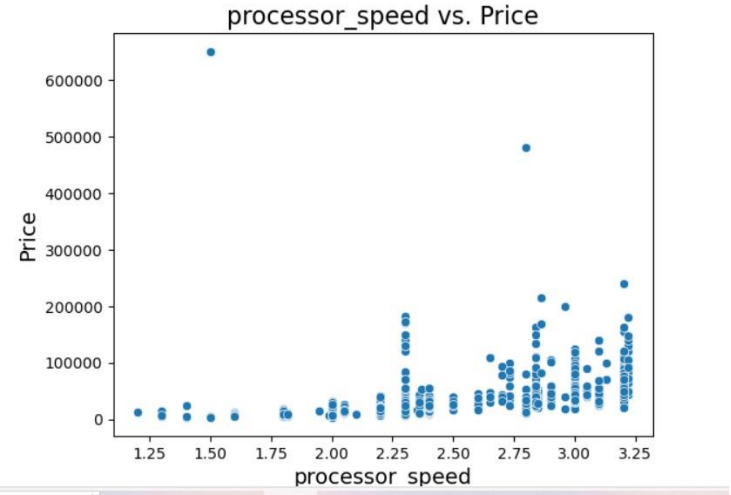
```
sns.scatterplot(data=df1,x='rating',y='price', marker='o')
plt.title('Rating Vs Price')
```

```
Text(0.5, 1.0, 'Rating Vs Price')
```



Above code creates a scatter plot of smartphone ratings versus prices.

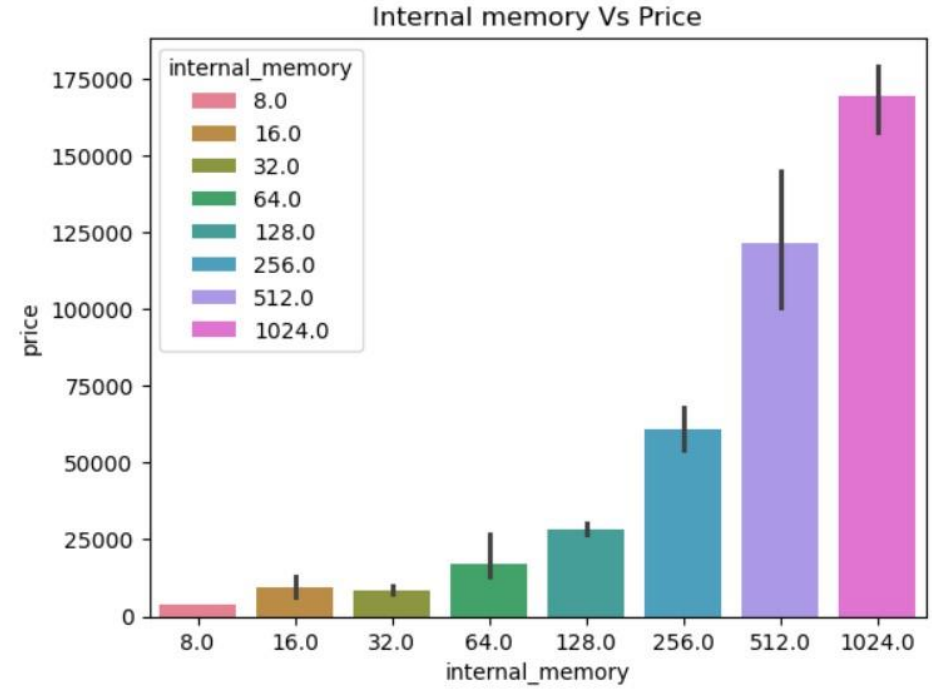
```
sns.scatterplot(data=df1,x='processor_speed',y='price')
plt.title('processor_speed vs. Price', fontsize=16)
plt.xlabel('processor_speed', fontsize=14)
plt.ylabel('Price', fontsize=14)
plt.show()
```



Above code shows relation between processor speed vs price

```
palette = sns.color_palette("husl", len(df1['internal_memory'].unique()))
sns.barplot(data=df1,x='internal_memory',y='price',hue='internal_memory', palette=palette)
plt.title('Internal memory Vs Price')
```

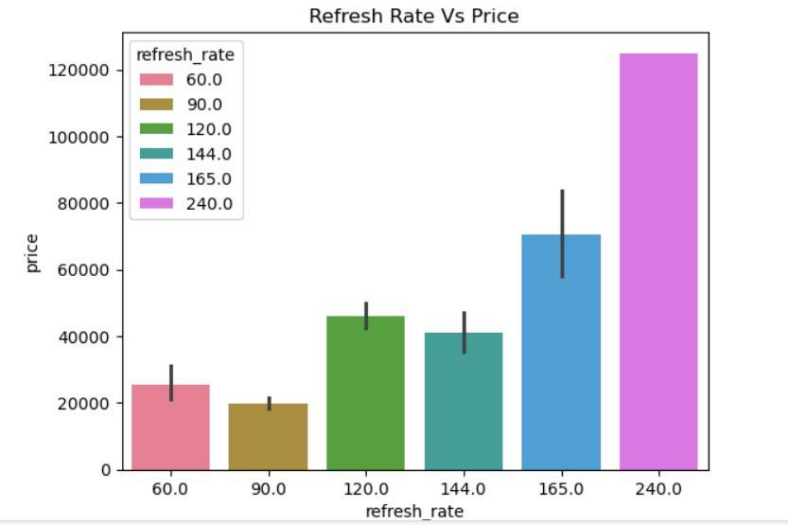
```
Text(0.5, 1.0, 'Internal memory Vs Price')
```



Above code creates a bar plot showing the relationship between internal memory sizes and prices of smartphones, using different colors for each memory size

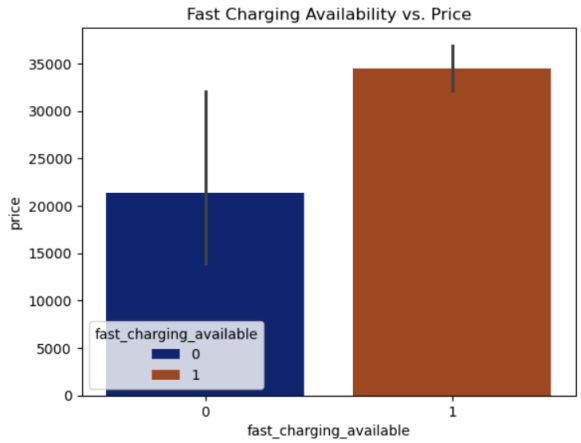
```
palette = sns.color_palette("husl", len(df1['refresh_rate'].unique()))

sns.barplot(data=df1,x='refresh_rate',y='price',hue='refresh_rate', palette=palette)
plt.title('Refresh Rate Vs Price')
```

```
Text(0.5, 1.0, 'Refresh Rate Vs Price')
```



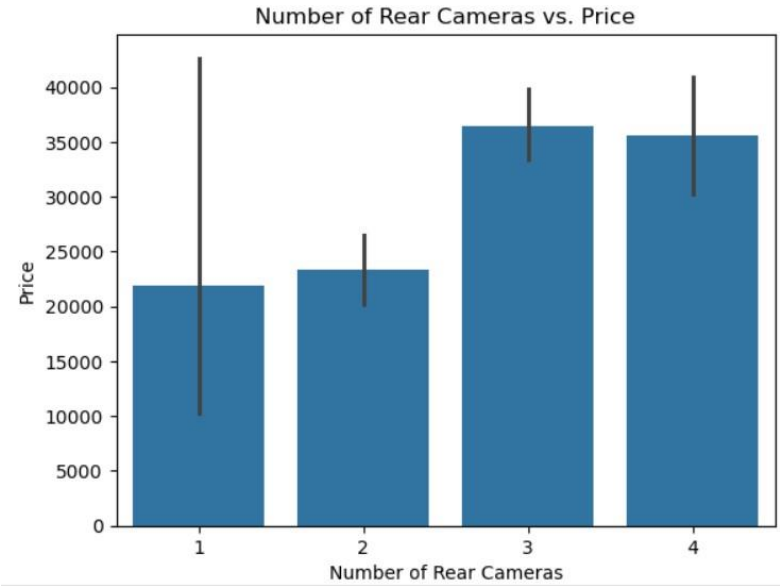
Above code creates a bar plot showing the relationship between Refresh rate and prices of smartphones, using different colors for each Refresh rate

```
sns.barplot(data=df1, x='fast_charging_available', y='price', hue='fast_charging_available', palette='dark')
plt.title('Fast Charging Availability vs. Price')
plt.show()
#higher price for fast charging
```



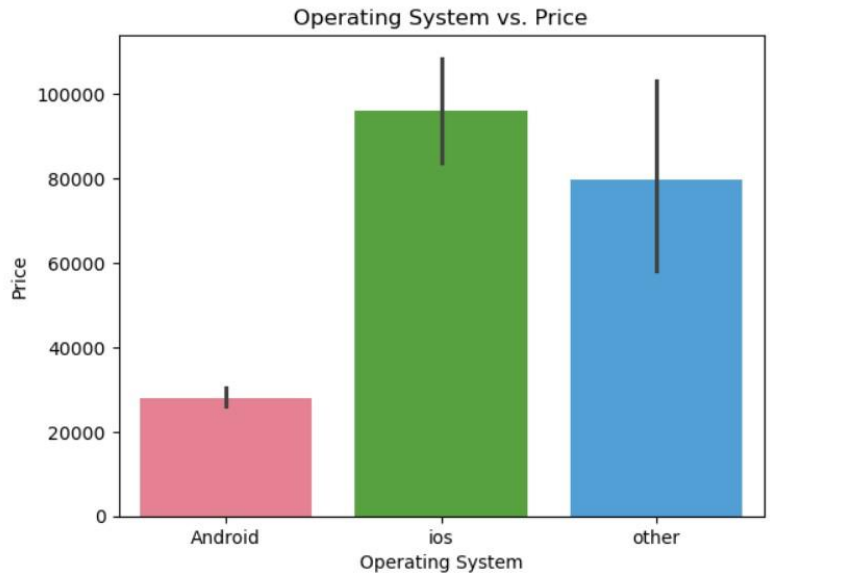
Above code creates a Bar plot showing the relationship between Fast charging availability and prices of smartphones,

```
sns.barplot(data=df1,x='num_rear_cameras',y='price')
plt.xlabel('Number of Rear Cameras')
plt.ylabel('Price')

plt.title('Number of Rear Cameras vs. Price')
plt.show()
```



Above code creates a bar plot showing the relationship between number of rear cameras and prices of smartphones.

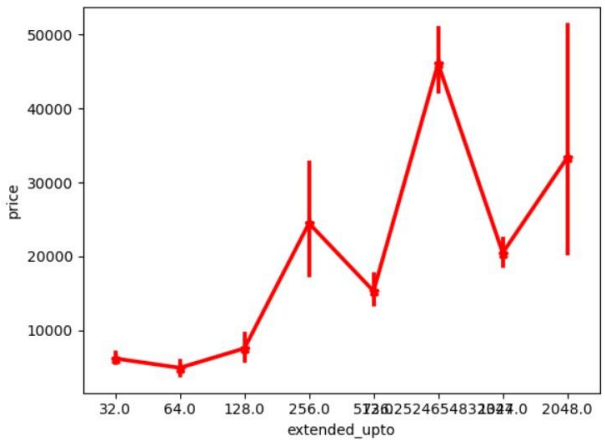
```
palette = sns.color_palette("husl", len(df1['os'].unique()))
sns.barplot(data=df1, x='os', y='price',hue='os', palette=palette)
plt.title('Operating System vs. Price')
plt.xlabel('Operating System')
plt.ylabel('Price')
plt.show()
```



Above code creates a bar plot showing the relationship between Operating System and prices of smartphones

```
sns.pointplot(data=df1,x='extended_upto', y='price',marker='*', color='red')
```

```
<Axes: xlabel='extended_upto', ylabel='price'>
```

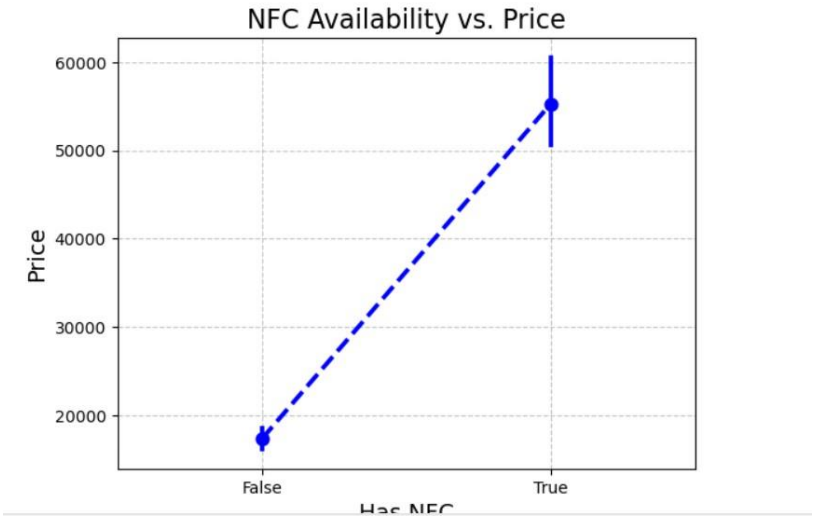


Above code creates a line plot showing the relationship between extended upto and prices of smartphones

```
sns.barplot(data=df1, x='has_5G', y='price', hue='has_5G',estimator=np.median, palette="viridis")
plt.title('5G Availability vs. Median Price', fontsize=16)
plt.xlabel('Has 5G', fontsize=14)
plt.ylabel('Median Price', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
medians = df1.groupby('has_5G')['price'].median()
for i, median in enumerate(medians):
    plt.text(i, median + 10, f'₹{int(median)}', ha='center', fontsize=12, color='black')
plt.show()
```

Above code bar plot of 5G availability vs. median smartphone price.

```
sns.pointplot(data=df1, x='has_nfc', y='price', color='blue', markers='o', linestyles='--')
plt.title('NFC Availability vs. Price', fontsize=16)
plt.xlabel('Has NFC', fontsize=14)
plt.ylabel('Price', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

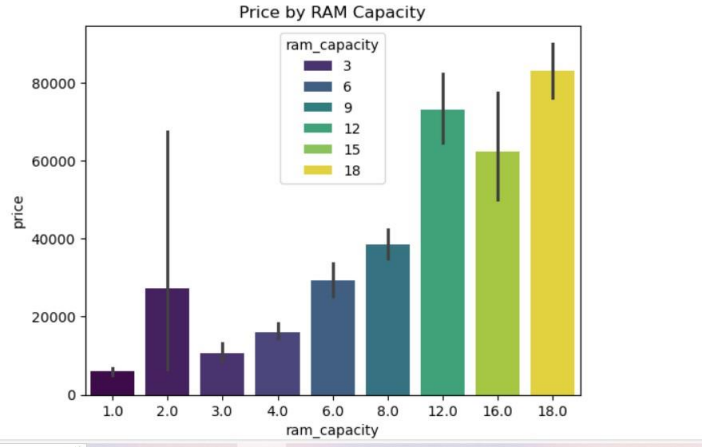


NFC feature Availibilty vs price line chart

```
sns.barplot(data=df1, x='has_IR_blaster', y='price',hue='has_IR_blaster', estimator=np.median, palette="coolwarm")
plt.title('IR Blaster Availability vs. Median Price', fontsize=16)
plt.xlabel('Has IR Blaster', fontsize=14)
plt.ylabel('Median Price', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
medians = df1.groupby('has_IR_blaster')['price'].median()
for i, median in enumerate(medians):
    plt.text(i, median + 10, f'₹{int(median)}', ha='center', fontsize=12, color='black')
plt.show()
```

Above code bar plot of IR Blaster availability vs. median smartphone price

```
sns.barplot(data=df1, x='ram_capacity', y='price',hue='ram_capacity', palette='viridis')
plt.title('Price by RAM Capacity')
plt.show()
```



Bove code showing ram capacity vs price bar graph

```
numeric_df1 = df1.select_dtypes(include=['float64', 'int64'])

correlation_with_price = numeric_df1.corr()['price']
correlation_with_price
```

```
price                       1.000000
rating                     -0.129441
num_cores                  -0.065820
processor_speed             0.437674
ram_capacity                0.386002
internal_memory             0.557168
battery_capacity           -0.157532
fast_charging_available     0.116739
fast_charging               0.194538
screen_size                 0.113253
refresh_rate                0.244115
num_rear_cameras            0.125330
num_front_cameras           0.055305
primary_rear_camera         0.092095
primary_front_camera        0.146122
extended_memory_available  -0.423621
extended_upto               0.020011
Name: price, dtype: float64
```

Above code shows the correlation coefficients between the price column and other columns in the DataFrame df1. Above output shows how each column is linearly related to the price column. A higher positive value indicates a positive correlation, while a lower negative value indicates a negative correlation.

Conclusion:

Conclusion Based on Data percentge

1. Brand- near about 50% of the market's phones are of only fives companies namely Samsung,Xiaomi,Vivo,realme and oppo
2. price- price data is higly skwed (skewed towards lower range) with mean around 32.5k with min and max of 3499 to 650000
3. Ratings: Right sided skew data with 10% of missing values
4. has_5g: just over half the phones are 5G phones
5. has_IR_blaster: near about 15% of the phones have IR_blaster in them
6. has_nfc: near about 60% of the phones have IR_blaster in them
7. processor_brand: About 80% of the phones in the market has either snapdragon,helio or dimensity as a processor
8. num_cores: over 90% of the processors are octacores only
9. ram_capacity : 1/3rd of the phones in the market has 8Gb of ram where as 6GB and 4GB comes next with about 24 and 22 percentage.

10. Fast_charging_available: More than 85% of the phones do have fast charging's feature available
11. Refersh_rates: The percentage of the phones with 60Hz and 120Hz are almost similar with 37% and 35% and 22% for the 90Hz making up for the 95%.
12. Total_no_Of_cameras=Half the phones have 4 cameras in total and about 20% have 3 total_cameras where as phones with 5 total_cameras are 16%
13. 93% phones in the market are andriod phones with 4.7 being ios and rest are others
14. extended_memory_available: 2/3rd of the phones do come up with extended memory option
15. extended_upto: out of the phones which provide the extended memory option near about 50% of them comes up with the option of extension upto 1024 GB and 23% comes up with option of extension upto 512Gb with 21% provides the option of extension upto 256 Gb

Conclusion Based on Price:-

1. brand-price:
    1. Apple phones are heavily priced
2. price-ratings
    1. Some outliers with Not a significant pattern and relationship in them

3.price-has_5G

1. significant impact of 5G phone on the price (price is higher for 5G phones)

4.price-has_Nfc

1. Significant relation between price and nfc

5.price-has_ir_blaster

1. No colcusive ration between the two of them

7. price vs processor_speed
    1. Price do increase with increase in processor speed but not very exclusive correlation
8. price vs ram_capacity
    1. Holds good linear relationship of price upto 12 GB of ram onwards it drops
9. price vs internal_memory
    1. strong linear relationship

10. price vs fast charging availabe
    1. higher price for fast charging

11. Price vs refresh_rate

    1. pretty significant positive linear realtion

12. price vs num_rear_cameras

    1. no_of_rear_cameras 1,2 doesnt'really differ in price but when we move to 3 then there is significant jump in the price but again there is no significant change from 3 to 4

13. price vs extended upto

    1. extended upto has positive relation with price with one sligh surge of slope in case of 256 GB

14. price vs os

    1. apple phones and other(exclusive phones) has higher price compared to andriod phones

price is coniderably dependent on following columns:
[internal_memory,ram_capacity,has_nfc,processor_speed]