Prediction of Used Phone and Tablet Price

Introduction

The used and refurbished device market has grown considerably over the past decade as it provide cost-effective alternatives to both consumers and businesses that are looking to save money when purchasing one. Maximizing the longevity of devices through second-hand trade also reduces their environmental impact and helps in recycling and reducing waste. In this project we will be predicting the used device price by considering various features in the dataset. We will also try to find out which features are responsible in deciding the price of the device.

Data Description

Features of the dataset:

1)device_brand: Name of manufacturing brand
2)os: OS on which the device runs
3)screen_size: Size of the screen in cm
4)4g: Whether 4G is available or not
5)5g: Whether 5G is available or not
6)front_camera_mp: Resolution of the rear camera in megapixels
7)back_camera_mp: Resolution of the front camera in megapixels
8)internal_memory: Amount of internal memory (ROM) in GB
9)ram: Amount of RAM in GB
10)battery: Energy capacity of the device battery in mAh
11)weight: Weight of the device in grams
12)release_year: Year when the device model was released
13)days_used: Number of days the used/refurbished device has been used
14)new_price: Price of a new device of the same model

Target Variable:
used_price (TARGET): Price of the used/refurbished device

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import LabelEncoder
from scipy.stats import shapiro
import pingouin as pg
from scipy.stats import kruskal
from statsmodels.stats.weightstats import ztest
from scipy.stats import mannwhitneyu
```

In [2]:
```
df= pd.read_csv('used_device_data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight	release_year	da
0	Honor	Android	14.50	yes	no	13.0	5.0	64.0	3.0	3020.0	146.0	2020	
1	Honor	Android	17.30	yes	yes	13.0	16.0	128.0	8.0	4300.0	213.0	2020	
2	Honor	Android	16.69	yes	yes	13.0	8.0	128.0	8.0	4200.0	213.0	2020	
3	Honor	Android	25.50	yes	yes	13.0	8.0	64.0	6.0	7250.0	480.0	2020	
4	Honor	Android	15.32	yes	no	13.0	8.0	64.0	3.0	5000.0	185.0	2020	

In [4]:
```
df.describe()
```

Out[4]:

	screen_size	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight	release_year	days_used
count	3454.000000	3275.000000	3452.000000	3450.000000	3450.000000	3448.000000	3447.000000	3454.000000	3454.000000
mean	13.713115	9.460208	6.554229	54.573099	4.036122	3133.402697	182.751871	2015.965258	674.869716
std	3.805280	4.815461	6.970372	84.972371	1.365105	1299.682844	88.413228	2.298455	248.580166
min	5.080000	0.080000	0.000000	0.010000	0.020000	500.000000	69.000000	2013.000000	91.000000
25%	12.700000	5.000000	2.000000	16.000000	4.000000	2100.000000	142.000000	2014.000000	533.500000
50%	12.830000	8.000000	5.000000	32.000000	4.000000	3000.000000	160.000000	2015.500000	690.500000
75%	15.340000	13.000000	8.000000	64.000000	4.000000	4000.000000	185.000000	2018.000000	868.750000
max	30.710000	48.000000	32.000000	1024.000000	12.000000	9720.000000	855.000000	2020.000000	1094.000000

In [5]:

```
df.shape
```

Out[5]:

```
(3454, 15)
```

In [6]:

```
df.rename(columns={'4g':'Includes_4g'},inplace=True)
df.rename(columns={'5g':'Includes_5g'},inplace=True)
```

In [7]:

```
df.isnull().sum()
```

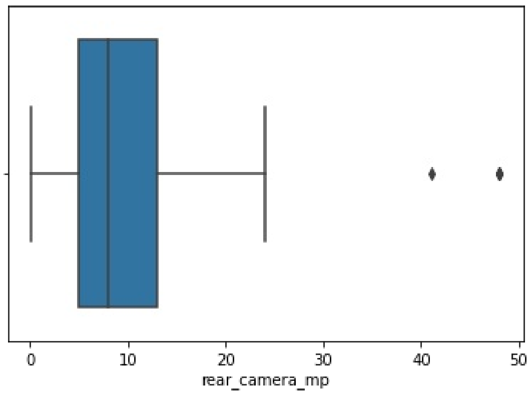
Out[7]:

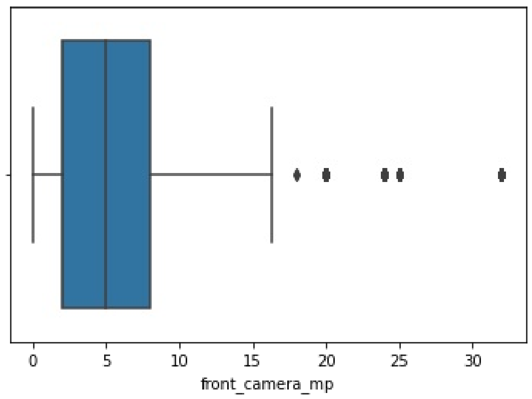
```
device_brand          0
os                    0
screen_size           0
Includes_4g           0
Includes_5g           0
rear_camera_mp      179
front_camera_mp       2
internal_memory       4
ram                   4
battery               6
weight                7
release_year          0
days_used             0
used_price            0
new_price             0
dtype: int64
```

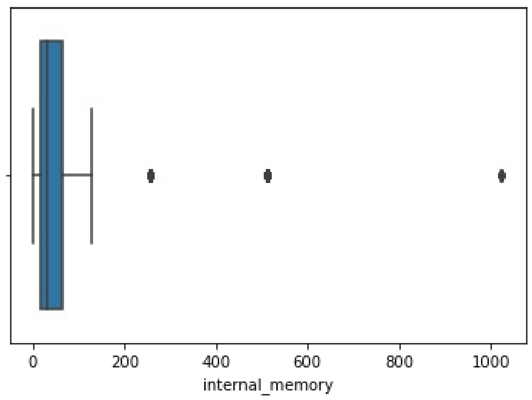
The dataset contains some null values, and removing them is not a viable option due to the small number of samples. Instead, we will handle them appropriately. Before doing so, we will investigate whether there are any outliers in the specified columns.

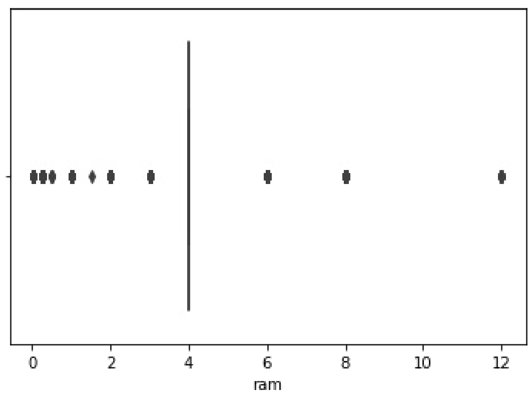
In [8]:

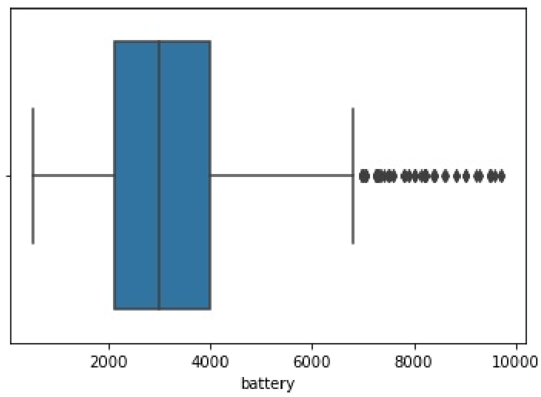
```
import seaborn as sns
import matplotlib.pyplot as plt

features=['rear_camera_mp','front_camera_mp','internal_memory','ram','battery','weight']

for i in features:
    sns.boxplot(df[i])
    plt.show()
```

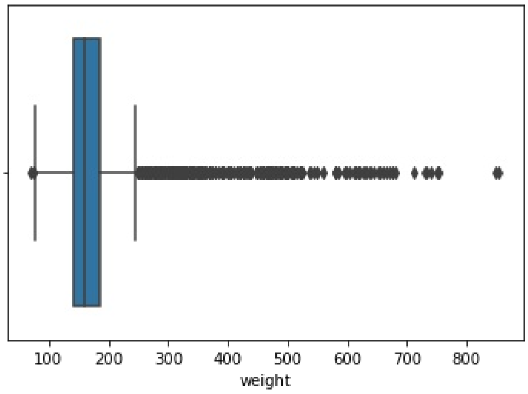



0
5
10
15
20
25
30
front_camera_mp


0
200
400
600
800
1000
internal_memory


0
2
4
6
8
10
12
ram


2000
4000
6000
8000
10000
battery



It has been observed that the data contains outliers.
It is not advisable to replace the null values with mean values in the presence of outliers, as the mean is highly sensitive to outliers and may significantly alter the central tendency of the data.
Hence, the median is preferred over the mean when the data contains outliers.

In [9]:
```
for i in features:
    median = df[i].median()
    df[i]=df[i].fillna(median)
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
device_brand        0
os                  0
screen_size         0
Includes_4g         0
Includes_5g         0
rear_camera_mp      0
front_camera_mp     0
internal_memory     0
ram                 0
battery             0
weight              0
release_year        0
days_used           0
used_price          0
new_price           0
dtype: int64
```

In [11]:
```
df['os'].unique()
```

Out[11]:
```
array(['Android', 'Others', 'iOS', 'Windows'], dtype=object)
```

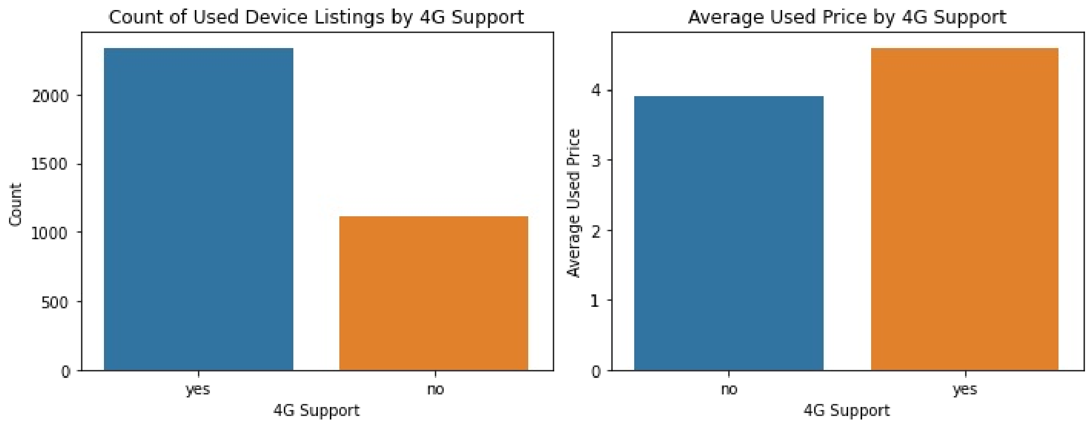
In [12]:
```
df.head()
```

Out[12]:

	device_brand	os	screen_size	Includes_4g	Includes_5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
0	Honor	Android	14.50	yes	no	13.0	5.0	64.0	3.0	3020.0	146.0
1	Honor	Android	17.30	yes	yes	13.0	16.0	128.0	8.0	4300.0	213.0
2	Honor	Android	16.69	yes	yes	13.0	8.0	128.0	8.0	4200.0	213.0
3	Honor	Android	25.50	yes	yes	13.0	8.0	64.0	6.0	7250.0	480.0
4	Honor	Android	15.32	yes	no	13.0	8.0	64.0	3.0	5000.0	185.0

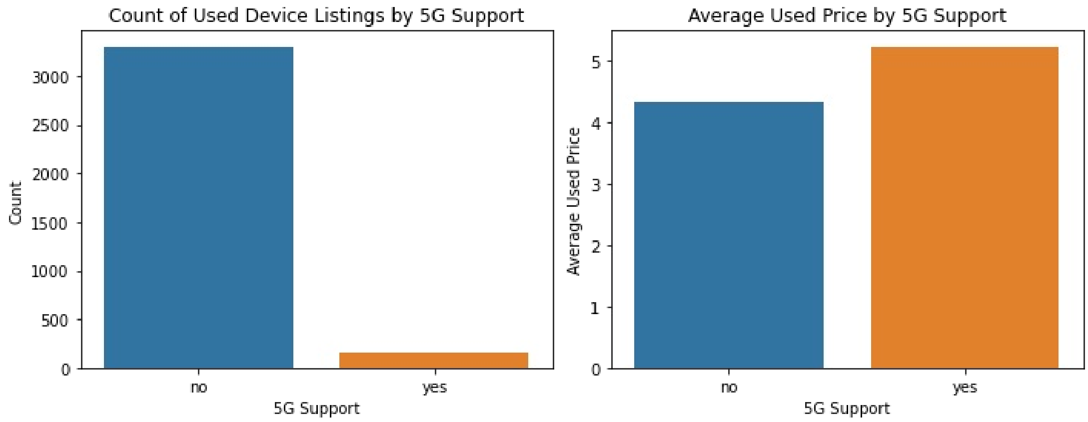
In [13]:
```
# Set the figure size
plt.figure(figsize=(10, 4))

# Create subplot 1 - Count plot of 4G support
plt.subplot(1, 2, 1)
sns.countplot(x='Includes_4g', data=df)
plt.title('Count of Used Device Listings by 4G Support')
plt.xlabel('4G Support')
plt.ylabel('Count')

# Create subplot 2 - Bar plot of average used price by 4G support
plt.subplot(1, 2, 2)
avg_price_by_4g = df.groupby('Includes_4g')['used_price'].mean().reset_index()
sns.barplot(x='Includes_4g', y='used_price', data=avg_price_by_4g)
plt.title('Average Used Price by 4G Support')
```

```python
plt.xlabel('4G Support')
plt.ylabel('Average Used Price')

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plots
plt.show()
```



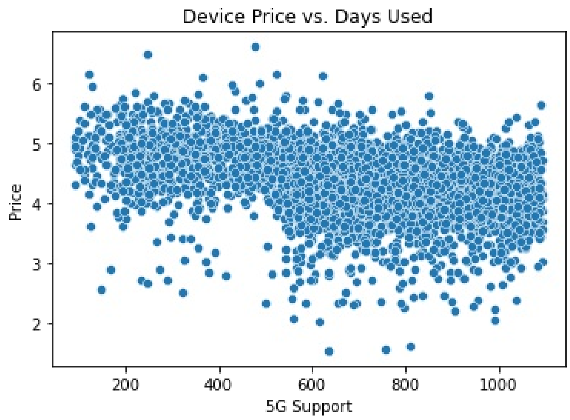
The visualization helps to understand the relationship between the 4G support and the used price of the devices.

In [14]:
```python
# Set the figure size
plt.figure(figsize=(10, 4))

# Create subplot 1 - Count plot of 5G support
plt.subplot(1, 2, 1)
sns.countplot(x='Includes_5g', data=df)
plt.title('Count of Used Device Listings by 5G Support')
plt.xlabel('5G Support')
plt.ylabel('Count')

# Create subplot 2 - Bar plot of average used price by 5G support
plt.subplot(1, 2, 2)
avg_price_by_5g = df.groupby('Includes_5g')['used_price'].mean().reset_index()
sns.barplot(x='Includes_5g', y='used_price', data=avg_price_by_5g)
plt.title('Average Used Price by 5G Support')
plt.xlabel('5G Support')
plt.ylabel('Average Used Price')

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plots
plt.show()
```



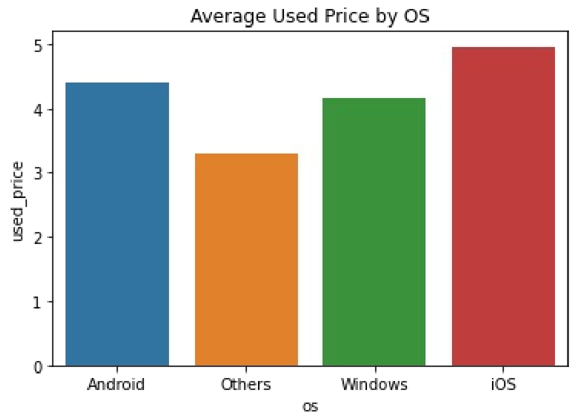
The visualization helps to understand the relationship between the 5G support and the used price of the devices.

In [15]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Create a scatter plot of device price vs. 5G support
sns.scatterplot(x='days_used', y='used_price', data=df)
plt.title('Device Price vs. Days Used')
plt.xlabel('5G Support')
plt.ylabel('Price')
plt.show()
```



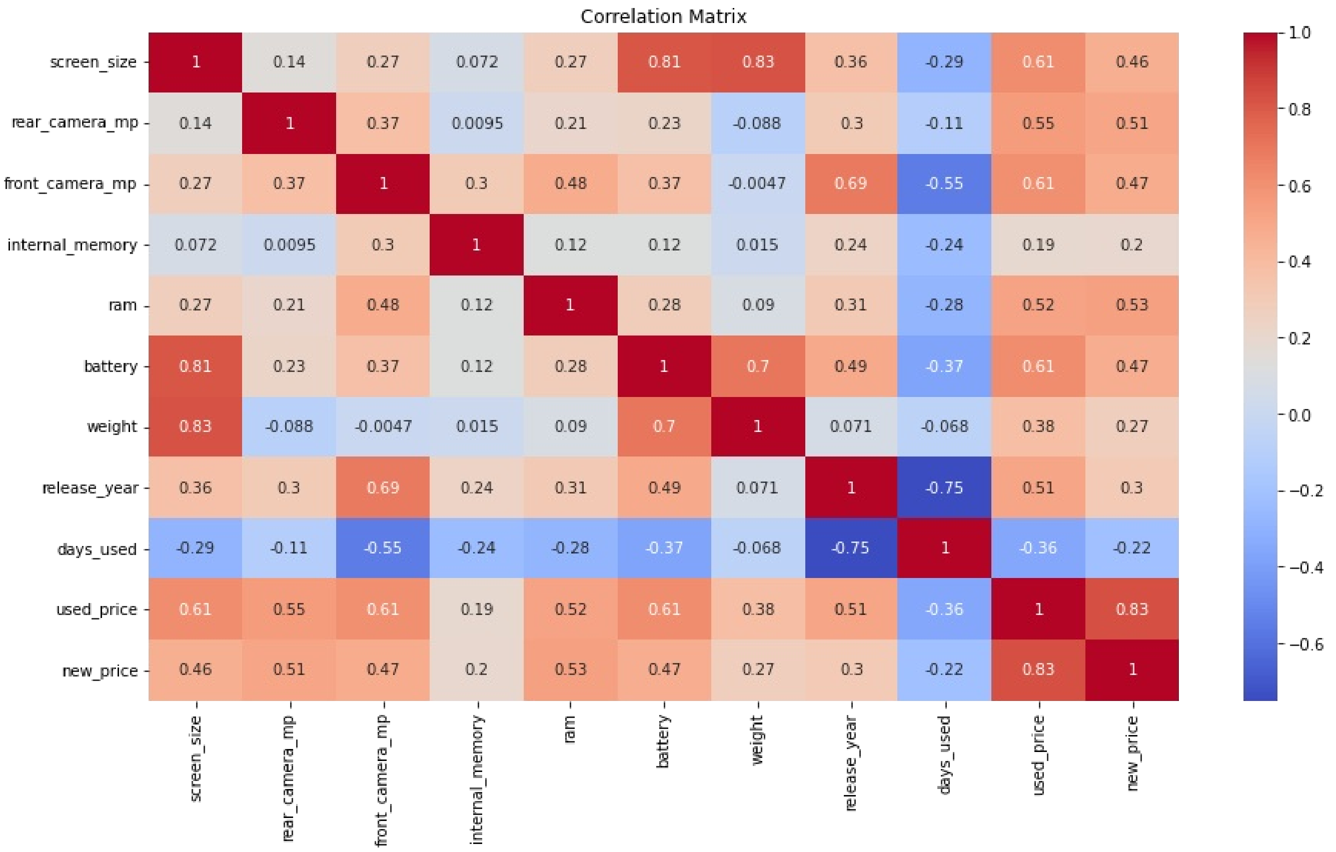
The above visualization is a scatter plot showing the relationship between the number of days a device has been used and its used price.From the plot, we can observe whether there is any correlation between the number of days a device has been used and its price.

In [16]:
```
# Calculate the average used price for each OS
avg_price_by_os = df.groupby('os')['used_price'].mean().reset_index()

# Create a barplot of the average used price by OS
sns.barplot(data=avg_price_by_os, x='os', y='used_price')
plt.title('Average Used Price by OS')
plt.show()
```



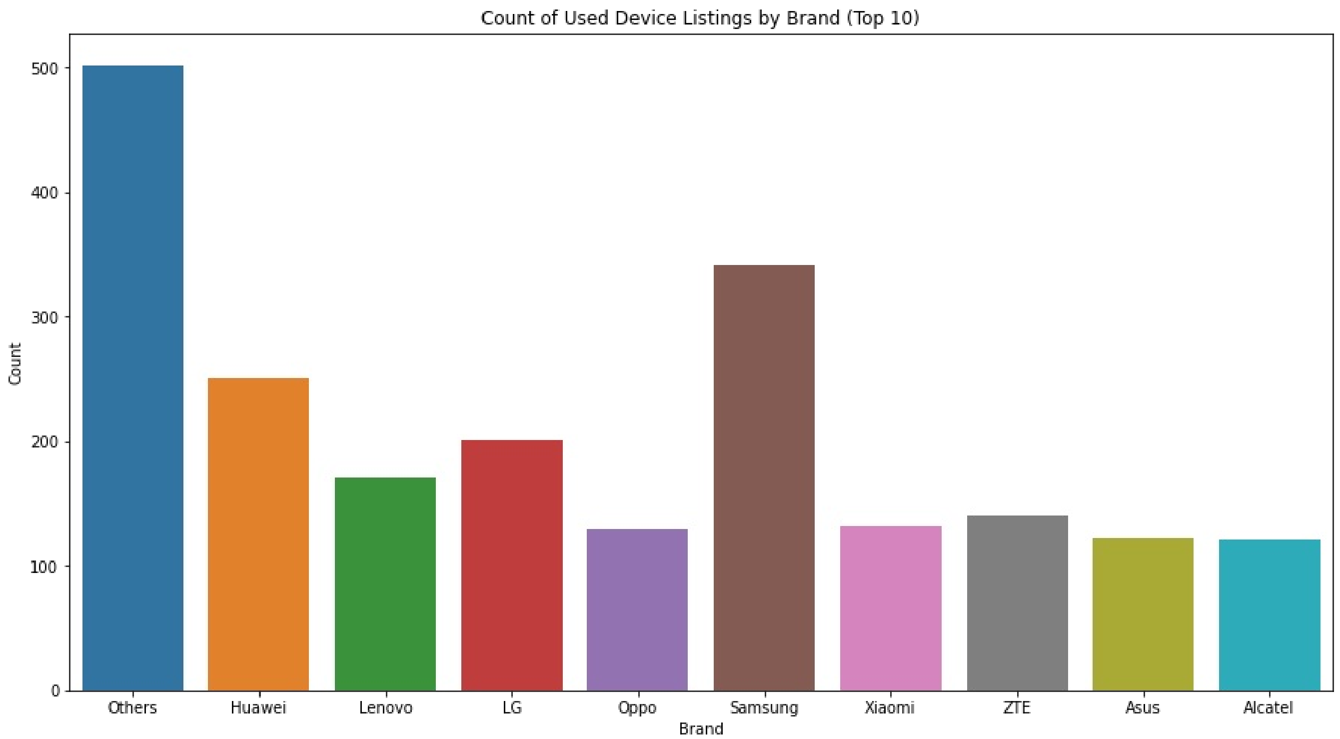
The plot provides an easy way to compare the average prices across different OS and can help in identifying which OS has higher or lower average prices.

In [17]:
```
# Compute the correlation matrix
corr = df.corr()

# Create a heatmap of the correlation matrix
plt.figure(figsize=(15, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



The factors of rear camera, front camera, battery, RAM, screen size, and new price have a strong correlation with the used device price, making them important in determining the final used device price.

In [18]:
```
# Get the top 10 brands by count
top_10_brands = df['device_brand'].value_counts().nlargest(10).index.tolist()

# Filter the dataframe to only include the top 10 brands
df_top_10_brands = df[df['device_brand'].isin(top_10_brands)]

# Create the countplot
plt.figure(figsize=(15, 8))
sns.countplot(x='device_brand', data=df_top_10_brands)
plt.title('Count of Used Device Listings by Brand (Top 10)')
plt.xlabel('Brand')
plt.ylabel('Count')
plt.show()
```



The plot helps in identifying the most popular brands among the used devices listed in the dataset.

In [19]:
```
encoder = LabelEncoder()
```

```
categorical_features = ['os','Includes_4g','Includes_5g','device_brand']
for i in categorical_features:
    df[i] = encoder.fit_transform(df[i])
```

4.1 - Comparing Two Sample

Shapiro Wilk Test

H0 - Data is normally distributed

H1 - Data is not normally distributed

In [20]:
```
features = df.columns.tolist()
data = df.copy()

for i in features:
    sample = data[i]
    stat, p = shapiro(sample)
    alpha = 0.05
    print('Feature - ',i)
    if p > alpha:
        print("The sample is likely to have been drawn from a normal distribution. (Fail to Reject H0)")
    else:
        print("The sample is not likely to have been drawn from a normal distribution.(reject H0)")
    print('--------------\n')
```

```
Feature -  device_brand
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  os
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  screen_size
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  Includes_4g
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  Includes_5g
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  rear_camera_mp
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  front_camera_mp
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  internal_memory
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  ram
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  battery
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  weight
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  release_year
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  days_used
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  used_price
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------

Feature -  new_price
The sample is not likely to have been drawn from a normal distribution.(reject H0)
--------------
```

Since, p - value < 0.05 we reject the Null Hypothesis in the Shapiro Wilk Test

Multivariate Normality Test

H0 - Data is normally distributed

H1 - Data is not normally distributed

In [21]:
```
#Multivariate Normality Test
test_stat, p_value, normal = pg.multivariate_normality(df)

alpha = 0.05
if p_value < alpha:
    print("The dataset is not multivariate normal.")
else:
    print("The dataset is multivariate normal.")
```

```
The dataset is not multivariate normal.
```

As the p - value < 0.05 we conclude that the data is not normally distributed and reject H0

In [22]:
```
data = df.copy()
```

In [23]:
```
includes_4g_data = data[data["Includes_4g"] == 1]
excludes_4g_data = data[data["Includes_4g"] == 0]
```

Z - test

Since the data doesnot follow a normal distribution, but the sample size is greater than 30 we use Z test.

H0: The population mean of the two samples are equal.

H1: The population mean of the two samples are not equal.

In [24]:
```
_, p_value = ztest(includes_4g_data['used_price'], excludes_4g_data['used_price'])
print('Includes 4g vs Excludes 4g')
alpha = 0.05
if p_value < alpha:
    print('Reject null hypothesis (H0: The population mean of the two samples are equal.): There is significant
else:
    print('Fail to reject null hypothesis (H0: The population mean of the two samples are equal.): There is not
```

```
Includes 4g vs Excludes 4g
Reject null hypothesis (H0: The population mean of the two samples are equal.): There is significant evidence t
o suggest that the population means of the two samples are different.
```

In [25]:
```
includes_5g_data = data[data["Includes_5g"] == 1]
excludes_5g_data = data[data["Includes_5g"] == 0]
```

In [26]:
```
_, p_value = ztest(includes_5g_data['used_price'], excludes_5g_data['used_price'])
print('Includes 5g vs Excludes 5g')
alpha = 0.05
if p_value < alpha:
    print('Reject null hypothesis (H0: The population mean of the two samples are equal.): There is significant
else:
    print('Fail to reject null hypothesis (H0: The population mean of the two samples are equal.): There is not
```

```
Includes 5g vs Excludes 5g
Reject null hypothesis (H0: The population mean of the two samples are equal.): There is significant evidence t
o suggest that the population means of the two samples are different.
```

4.2 - Analysis of Variance

One-way ANOVA (F-test)

As we assume the data to be normal since sample size is greater than 30. We use the F - test for analysis of variance.

H0: The population means of all groups are equal.

H1: At least one population mean is different from the others.

In [27]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model = ols("used_price ~ os", data=data).fit()
anova_result = sm.stats.anova_lm(model, typ=1)
print(anova_result)
```

```
              df       sum_sq    mean_sq          F        PR(>F)
os           1.0    17.353165  17.353165  50.75617  1.267204e-12
Residual  3452.0  1180.213702   0.341893       NaN           NaN
```

From the above we can say that p value is less than 0.05 and hence we reject the null hypothesis.

4.3 - The Analysis of Categorical Data

The chi-square test of independence is a statistical method used to determine if there is a relationship between two categorical

or nominal variables.

H0 - There is no association between the two features

H1 - There is an association between the two features

In [28]:
```
import pandas as pd
from scipy.stats import chi2_contingency

def chi_square_test(df, column1, column2, alpha=0.05):
    contingency_table = pd.crosstab(df[column1], df[column2])
    chi2, p_value, dof, expected = chi2_contingency(contingency_table)
    print(f"Hypothesis statements:")
    print(f"H0: There is no association between {column1} and {column2}.")
    print(f"H1: There is an association between {column1} and {column2}.")
    print(f"Chi-square statistic: {chi2:.2f}")
    print(f"Degrees of freedom: {dof}")
    print(f"P-value: {p_value:.4f}")
    if p_value < alpha:
        print(f"Since p-value ({p_value:.4f}) < alpha ({alpha}), we reject the null hypothesis.")
    else:
        print(f"Since p-value ({p_value:.4f}) >= alpha ({alpha}), we fail to reject the null hypothesis.")
    return chi2, p_value
```

In [31]:
```
chi_square_test(data, 'Includes_4g', 'Includes_5g')
```

```
Hypothesis statements:
H0: There is no association between Includes_4g and Includes_5g.
H1: There is an association between Includes_4g and Includes_5g.
Chi-square statistic: 74.66
Degrees of freedom: 1
P-value: 0.0000
Since p-value (0.0000) < alpha (0.05), we reject the null hypothesis.
```

Out[31]:
```
(74.65651245295918, 5.601654350200409e-18)
```

In [32]:
```
chi_square_test(df, 'Includes_4g', 'os')
```

```
Hypothesis statements:
H0: There is no association between Includes_4g and os.
H1: There is an association between Includes_4g and os.
Chi-square statistic: 170.10
Degrees of freedom: 3
P-value: 0.0000
Since p-value (0.0000) < alpha (0.05), we reject the null hypothesis.
```

Out[32]:
```
(170.0965908090805, 1.212868051325614e-36)
```

In [33]:
```
chi_square_test(df, 'os', 'Includes_5g')
```

```
Hypothesis statements:
H0: There is no association between os and Includes_5g.
H1: There is an association between os and Includes_5g.
Chi-square statistic: 11.87
Degrees of freedom: 3
P-value: 0.0078
Since p-value (0.0078) < alpha (0.05), we reject the null hypothesis.
```

Out[33]:
```
(11.87282923701839, 0.00783173562088424)
```

4.4 Linear Regression

In [37]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd

X = df.drop(['used_price'],axis=1)
y = df['used_price']

x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=100)

model = LinearRegression().fit(x_train, y_train)
y_pred = model.predict(x_test)
```

In [38]:
```
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae

print("MSE =", mse(y_pred, y_test))
print("MAE =", mae(y_pred, y_test))
```

```
MSE = 0.06094559039832358
MAE = 0.19123345868446304
```

In [ ]: