PROJECT 3

ReCell

Supervised Learning - Foundations

Date: 02 – Dec - 2022

Name: Ann Mariya Jomon

CONTENTS / AGENDA

1. 

Executive Summary

2. 

Business problem overview & Solution approach

3. 

EDA Results

4. 

Data Preprocessing

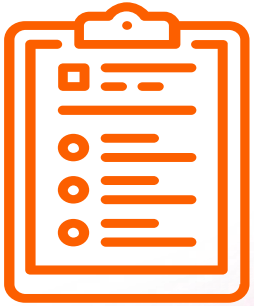
5. 

Model performance summary

6. 

Appendix

1. Executive Summary



Insights

EDA

- The average normalized used price is between 4 to 5.
- The normalized used price ranges from 1.54 to 6.62.
- The most common brand is Samsung followed by Huawei.
- More than 90% of the devices operate on Android OS.
- Only 8% of the devices were released in 2020.
- Normalized new price and normalized used price have high positive correlation.
- Normalized used price and days used have a negative correlation.
- The normalized used price has had a steady increase over the years 2013 – 2020.
- 67.6% of the devices have 4G.
- 95.6% of the devices have 5G.
- The normalized used price for devices with 4G or with 5G have higher normalized used price than the devices without them.

Insights

Model

- The model is able to explain 84% of the variation in the data.
- The key factors that influence the normalized used price are:
  - Main camera mp
  - Selfie camera mp
  - RAM
  - Weight
  - Years since release
  - With / without 4G
  - With / without 5G
- The train and test RMSE and MAE are low and comparable. So, the model is not overfitting.
- The MAPE on the test data suggests that we can predict within 4.56% of the normalized used price.
- Hence, we can conclude that the model is good for prediction as well as inferences.

Recommendations

- In order to increase demand, ReCell can focus on selling devices that have good quality rear and selfie cameras since people care a lot about the camera quality for taking selfies an photos. According to the results, Huawei, Vivo, Oppo, Xiaomi, Samsung and Sony are the top brands with good quality cameras.
- Storage is another factor which people look into. OnePlus is the top brand in this area. ReCell can consider including more of OnePlus devices. Another factor that ReCell must consider is whether there are customers who use the devices for gaming or video editing purposes. They will need RAM up to 8GB.
- ReCell can classify its customers based on their personal preferences in a phone or tablet. This will make it easier for ReCell to focus on different needs and cater to it accordingly.
- ReCell can use the equation of linear regression to predict the prices. They can adjust the price according to the predictors given. They can also see what factors influence the price the most and what factors do not influence it at all.
- ReCell can also compare the prices with the price of the actual device and see if its reasonable enough since customers look for lowest prices.

2. Business problem overview & Solution approach

PROBLEM OVERVIEW & SOLUTION APPROACH

Problem overview

The demand for an ML-based solution to create a dynamic pricing strategy for used and refurbished devices is fueled by the growing potential of the used devices industry.

ReCell, a startup looking to capitalize on this market's potential, plans to examine the data and create a linear regression model to forecast the price of a used phone or tablet and find variables that have a substantial impact on it.

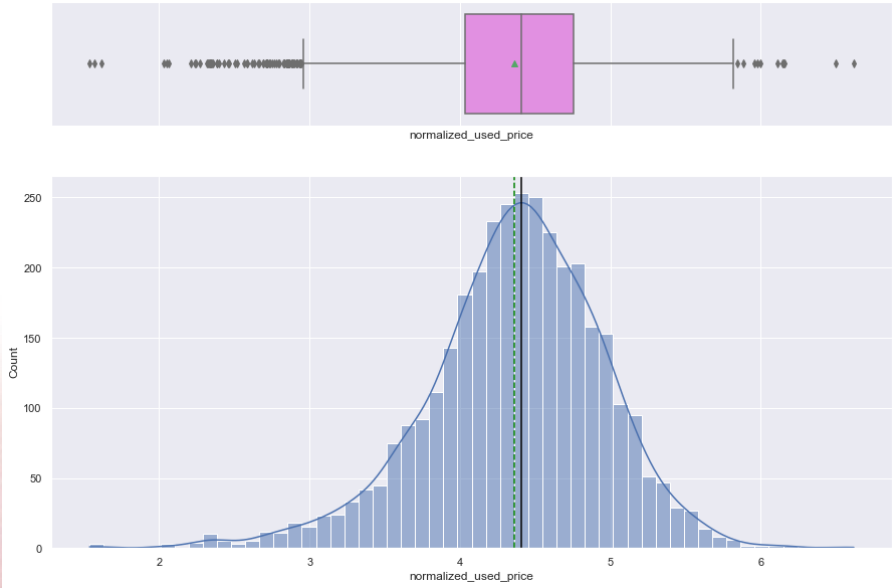
Solution approach

1. **EDA**:
   univariate and bivariate analysis
2. **Data preprocessing:**
   duplicate value check, missing value check, outlier check, feature engineering, and data preparation for modeling.
3. **Testing model assumptions:**
   multicollinearity, linearity & independence, normality, homoscedasticity
4. **Model performance summary:**
   overview of the model, parameters of the model, key performance indicators.
5. **Insights & Recommendations**

3. EDA Results

UNIVARIATE ANALYSIS

Normalized used price

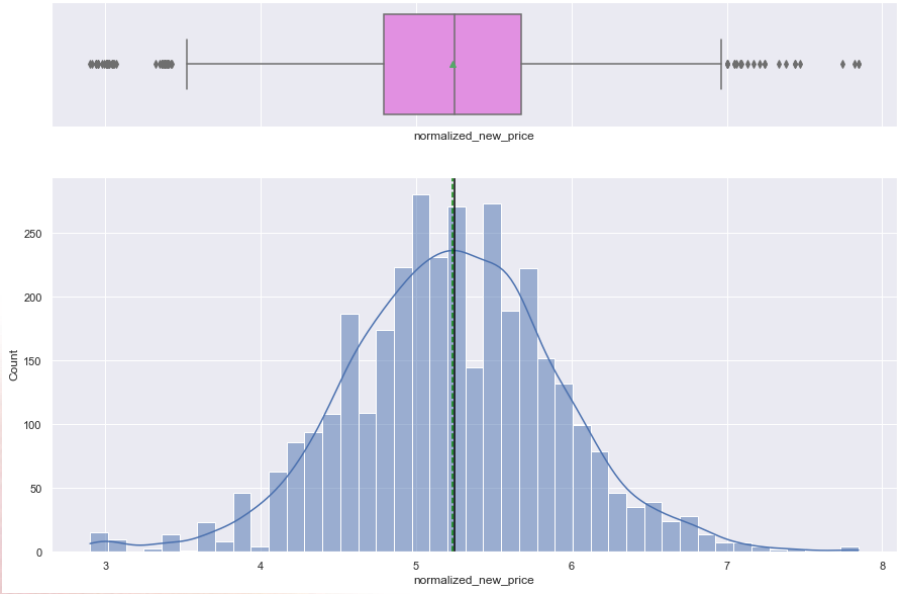


Observations

- The normalized used price displays a normal distribution.
- **Majority of the devices range between 4 to 5.**
- The average price is between 4 to 5.

UNIVARIATE ANALYSIS

Normalized new price

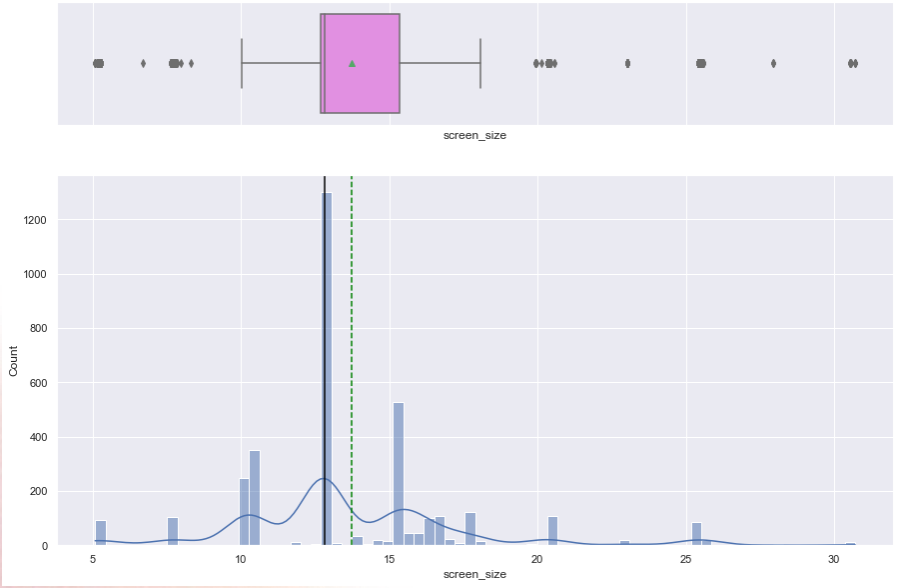


Observations

- The normalized new price kind of displays a normal distribution with the bell-shaped curve but its not exact.
- **The average rate is between 5 to 6.**

UNIVARIATE ANALYSIS

Screen Size

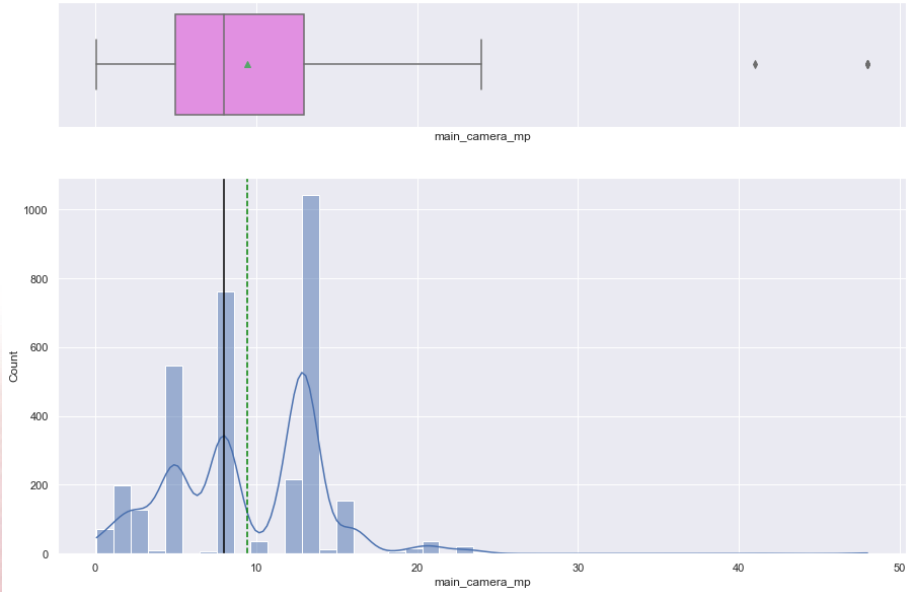


Observations

- **The distribution is a bit right-skewed with the most common screen size around 13 cm.**
- The mean and median screen size are different.

UNIVARIATE ANALYSIS

Main camera mp

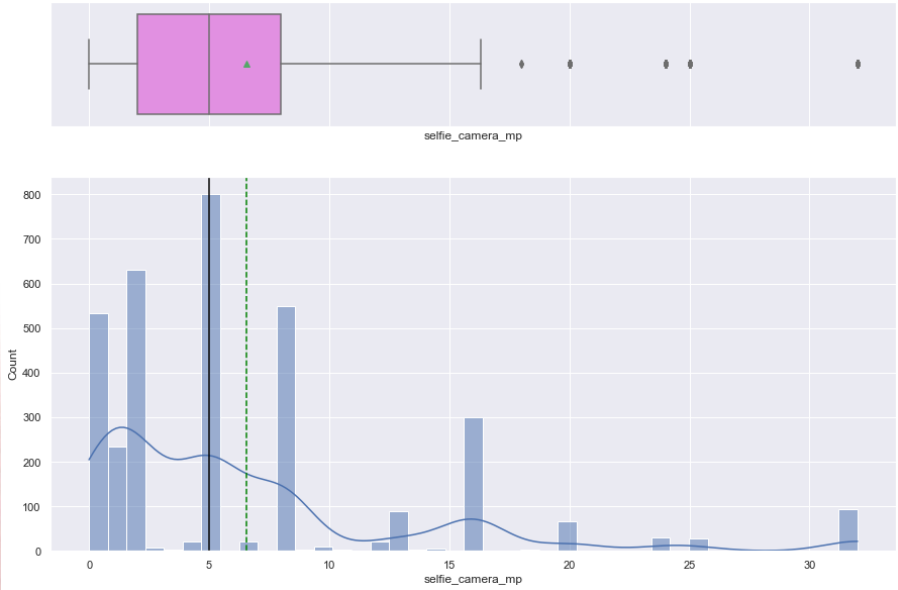


Observations

- The main camera mp distribution is **right skewed**.
- **The main camera mp for 75% of the devices is below 13.**
- There is a spike at around 13 mp.
- The mean and median main camera mp are different.
- The mean is greater than the median.

UNIVARIATE ANALYSIS

Selfie camera mp

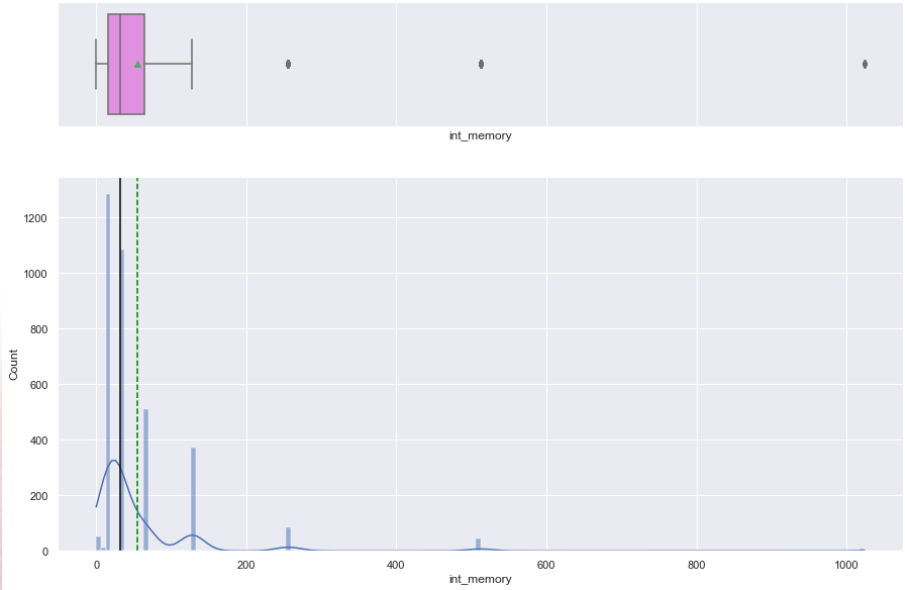


Observations

- The selfie camera mp distribution is **right-skewed.**
- The median is at 5 mp and is lesser than the mean.
- **Majority of the devices have 5 mp for their selfie camera.**

UNIVARIATE ANALYSIS

Internal memory

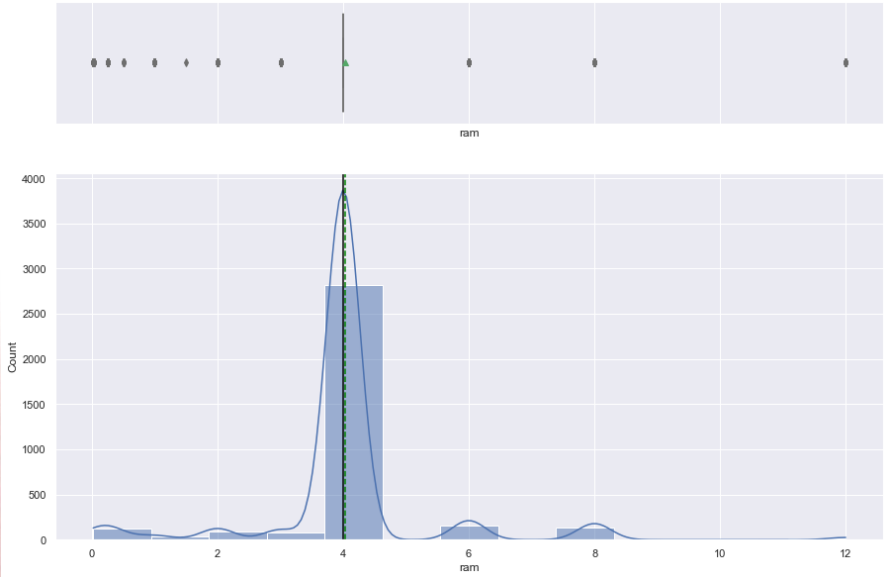


Observations

- The internal memory distribution is **heavily right-skewed**.
- Majority of the devices have internal memory of 16 GB & 32 GB.
- **More than 1200 devices have an internal memory of 16 GB.**

UNIVARIATE ANALYSIS

RAM

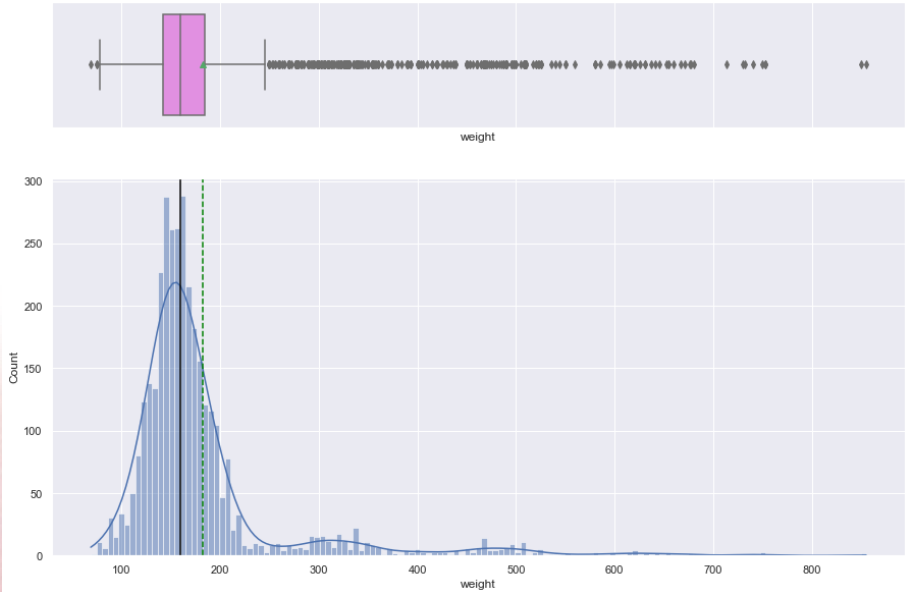


Observations

- The distribution of RAM is slightly right-skewed.
- The boxplot shows no box. This could be because, either there is not big variance in the data and hence there are no whiskers meaning there are no outliers, or the values are too close that the interquatile range is negligible.
- The latter seems more suitable in this situation.
- **More than 2500 devices have ram of 4GB. The remaining are outliers ranging till 12 GB.**

UNIVARIATE ANALYSIS

Weight

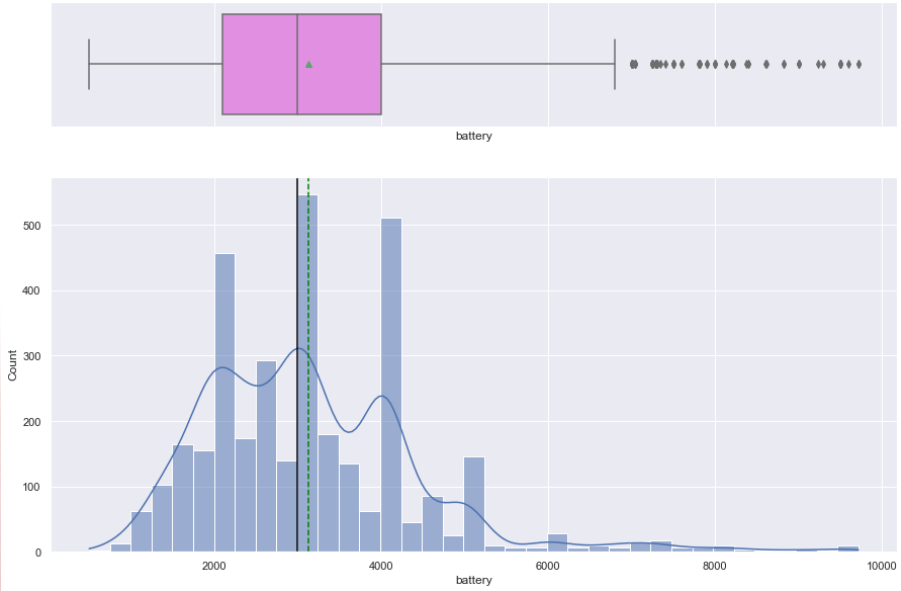


Observations

- The distribution of weight is **heavily right-skewed**.
- **Majority of the devices have weights ranging between 100 to 200 grams.**
- The mean weight is greater than the median weight.

UNIVARIATE ANALYSIS

Battery

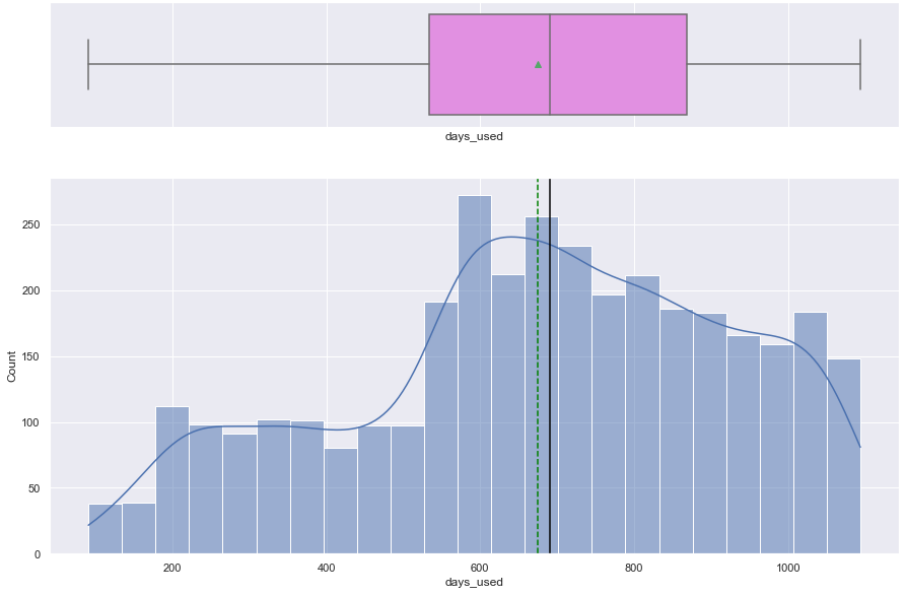


Observations

- Battery displays a multi-modal distribution.
- **More than 500 devices have battery of 3000 & 4000 mAh.**
- More than 400 devices have batter of 2000 mAh.
- The remaining devices are distributed between battery range of more than 4000 mAh but less than 10000 mAh.
- **50% of the devices have battery of less than 3000 mAh.**

UNIVARIATE ANALYSIS

Days used

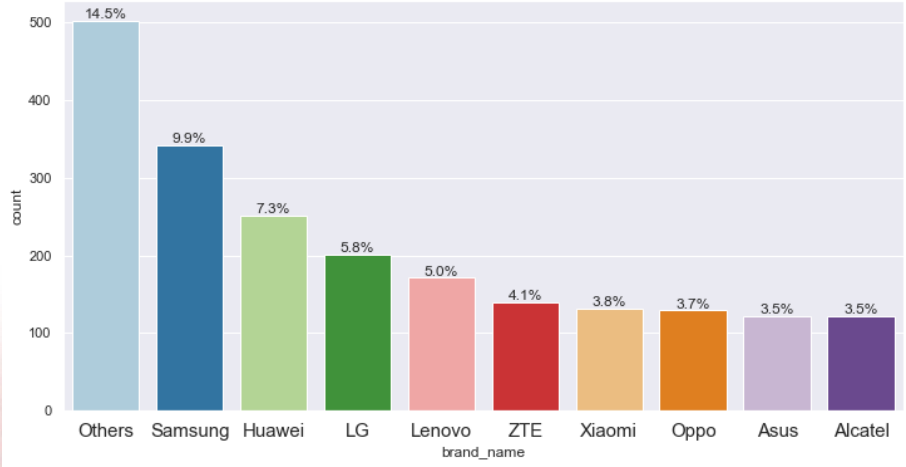


Observations

- The distribution of days used is slightly left-skewed.
- **75% of the devices have been used for more than 500 days.**
- 25% of the devices have been used for more than 900 days.
- The mean number of days used is less than the median.

UNIVARIATE ANALYSIS

Brand name

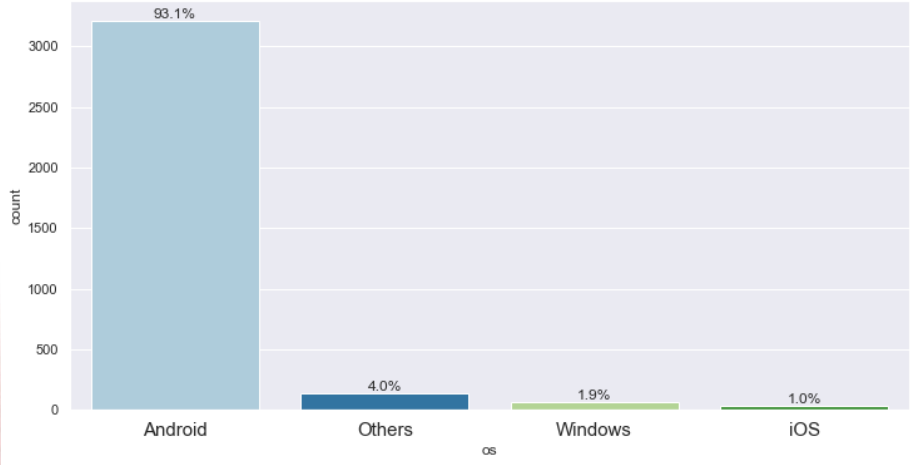


Observations

- **The most common brand is Samsung representing 9.9%, followed by Huawei at 7.3%.**
- Xiaomi, Oppo, Asus, and Alcatel have an almost equal distribution.
- Other brands represent 14.5%.

UNIVARIATE ANALYSIS

OS

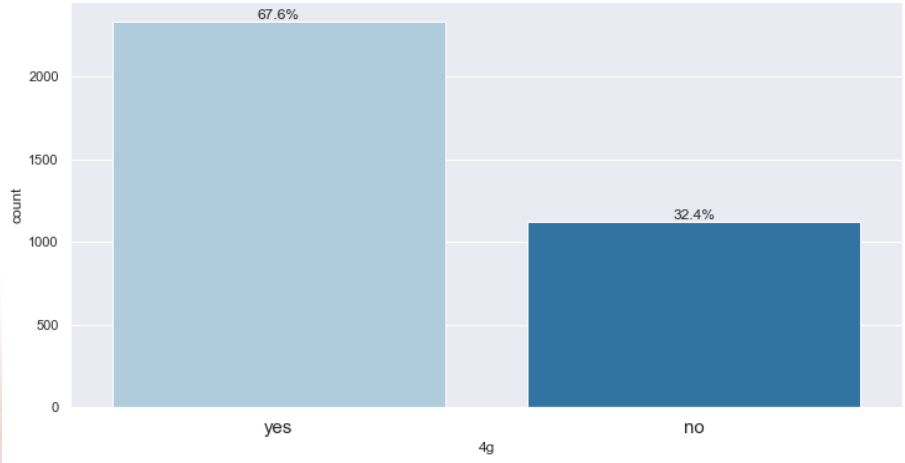


Observations

- **Majority of the devices operate on Android OS.**
- iOS is the least common one.

UNIVARIATE ANALYSIS

4g

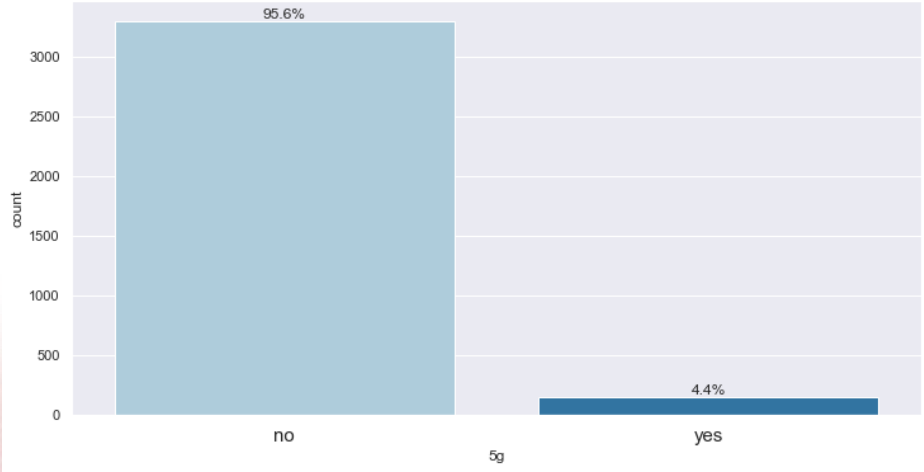


Observations

- **67.6% of the devices have 4G.**
- 32.4% of the devices do not have 4G.

UNIVARIATE ANALYSIS

5g

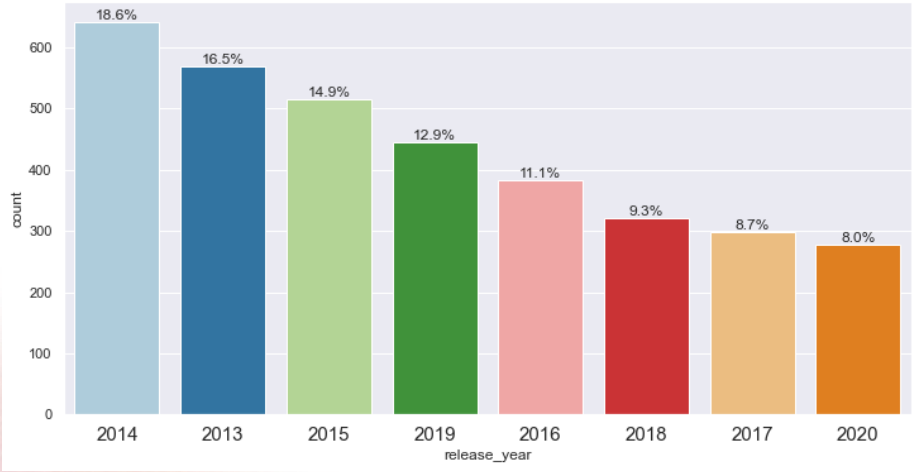


Observations

- **95.6% of the devices have 5G.**
- 4.4% of the devices do not have 5G.

UNIVARIATE ANALYSIS

Release Year

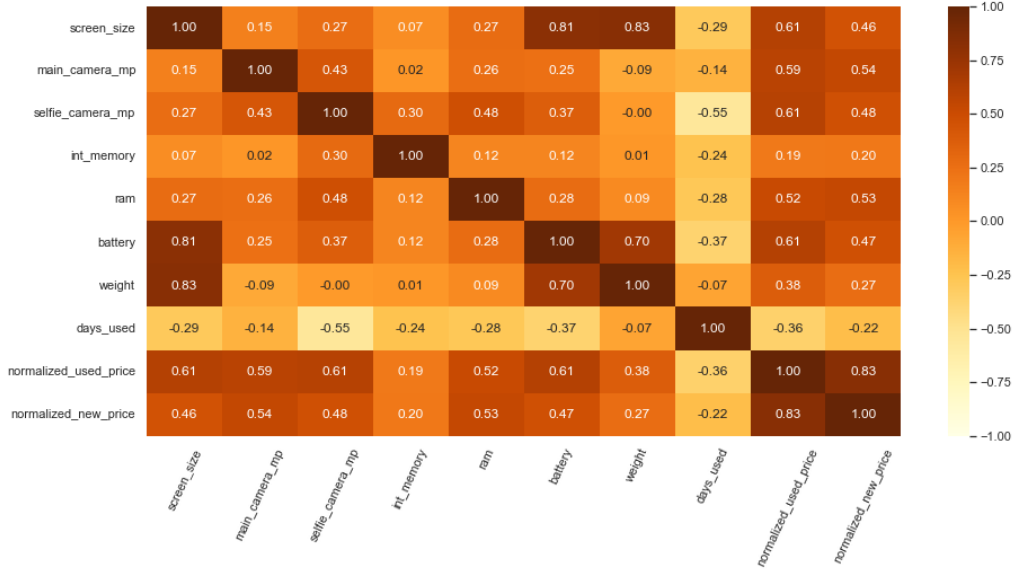


Observations

- **18.6% of the devices were released in 2014.**
- Only 8% of the devices were released in 2020.

BIVARIATE ANALYSIS

Correlation check

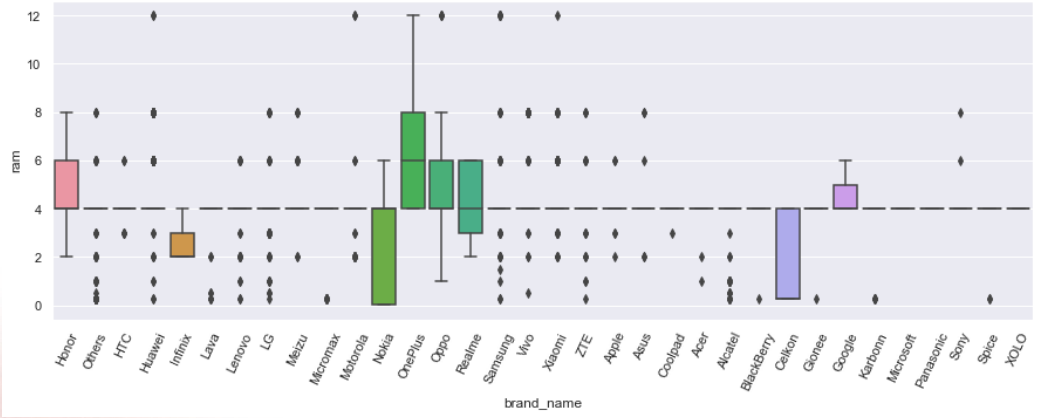


Observations

- **Normalized new price and normalized used price have high positive correlation.**
- Weight and screen size have high positive correlation.
- Battery and screen size have high positive correlation.
- Battery and weight also has a positive correlation.
- **Normalized used price and days used have a negative correlation. Same with normalized new price.**
- Days used also have a negative correlation with selfie_camera_mp, int_memory, ram, battery, and weight.

BIVARIATE ANALYSIS

RAM vs Brand name

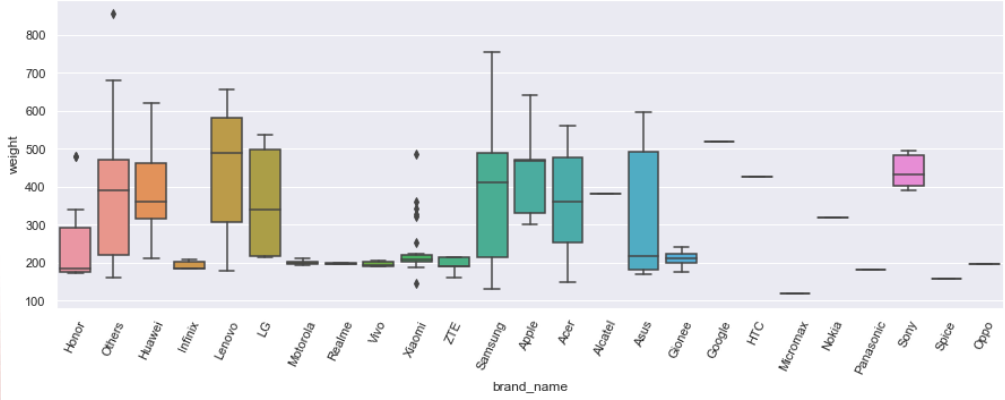


Observations

- The amount of RAM is important for the smooth functioning of a device.
- **The highest range of RAM is for OnePlus devices ranging between 4 – 12 GB.**
- **The lowest range of RAM belongs to Nokia ranging between 0 - 6 GB.**
- 75% of Honor and Oppo devices ranges between 4 – 8 GB.
- There are no whiskers for Celkon devices meaning that the lower quartile is equal to the minimum and upper quartile is equal to the maximum.
- **Samsung and Huawei only have a flat line indicating that most of their devices have a RAM of 4 GB.**

BIVARIATE ANALYSIS

*Weight vs Brand name



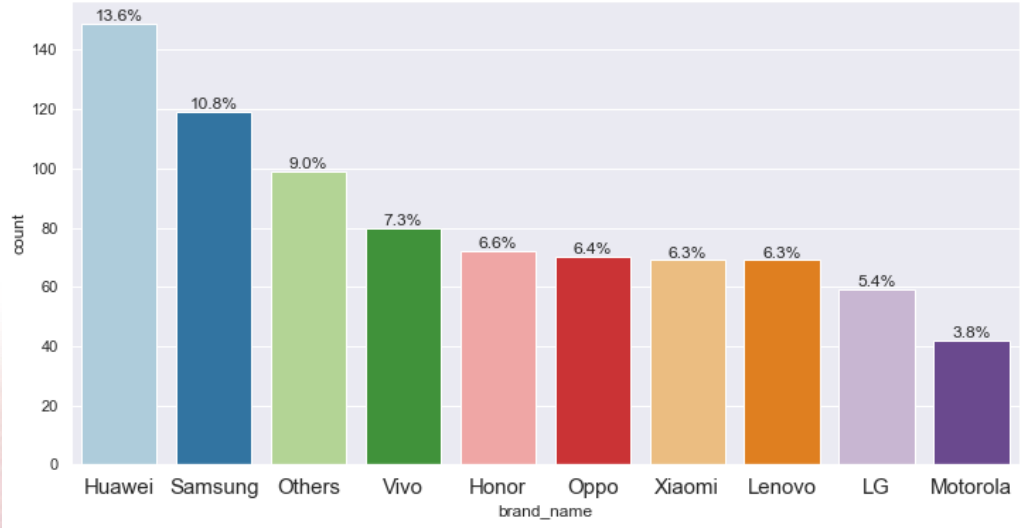
*Weight: weights of devices with large battery sizes (>4500 mAh)

Observations

- People who travel frequently require devices with large batteries to run through the day. But large battery often increases weight, making it feel uncomfortable in the hands.
- **Samsung devices have weights going up to 700 – 800 grams which is the highest weight.**
- The top 5 devices with highest weights as per their battery sizes are, Samsung, Lenovo, Apple, Huawei, Asus.

BIVARIATE ANALYSIS

Brands with large screen*



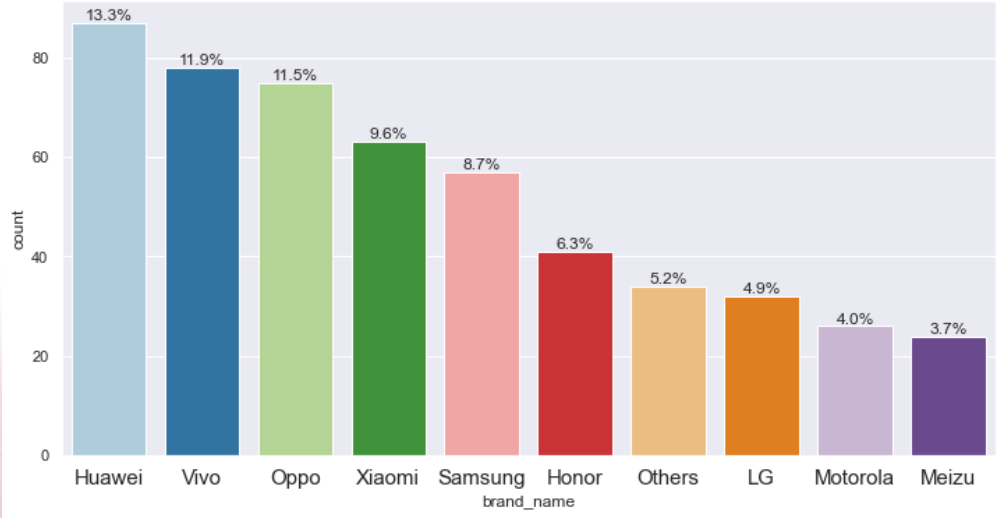
*Large screen: Screen size greater than 6cm

Observations

- People who buy phones and tablets primarily for entertainment purposes prefer a large screen as they offer a better viewing experience.
- **Huawei devices have the largest screens followed by Samsung.**
- The top 5 devices with large screens are, Huawei, Samsung, Vivo, Honor, and Oppo.

BIVARIATE ANALYSIS

Brands with better quality selfie camera*



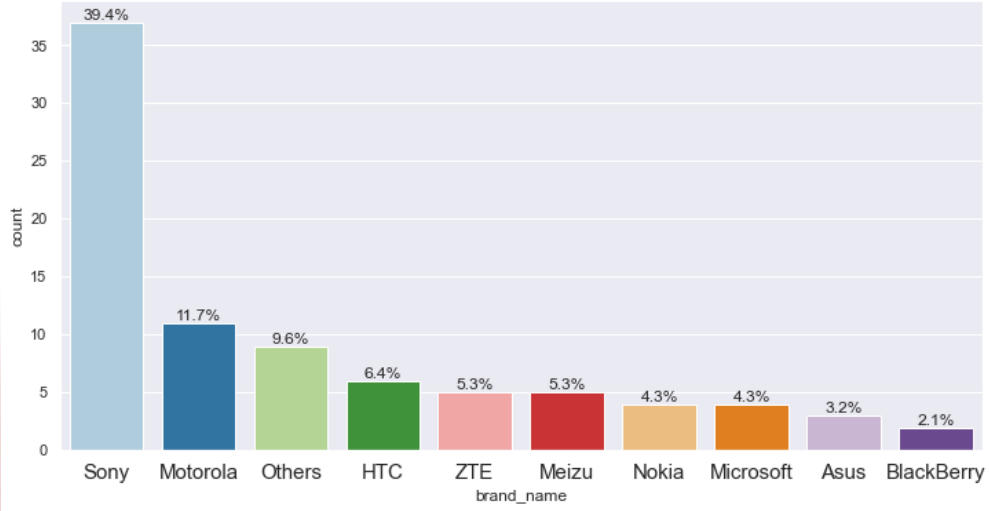
*Selfie camera: selfie camera mp > 8

Observations

- Everyone likes a good camera to capture their favorite moments with loved ones. Some customers specifically look for good front cameras to click cool selfies.
- **Huawei devices have the best selfie cameras followed by Vivo.**
- The top 5 devices with best selfie cameras are Huawei, Vivo, Oppo, Xiaomi, and Samsung.

BIVARIATE ANALYSIS

Brands with better quality rear camera*



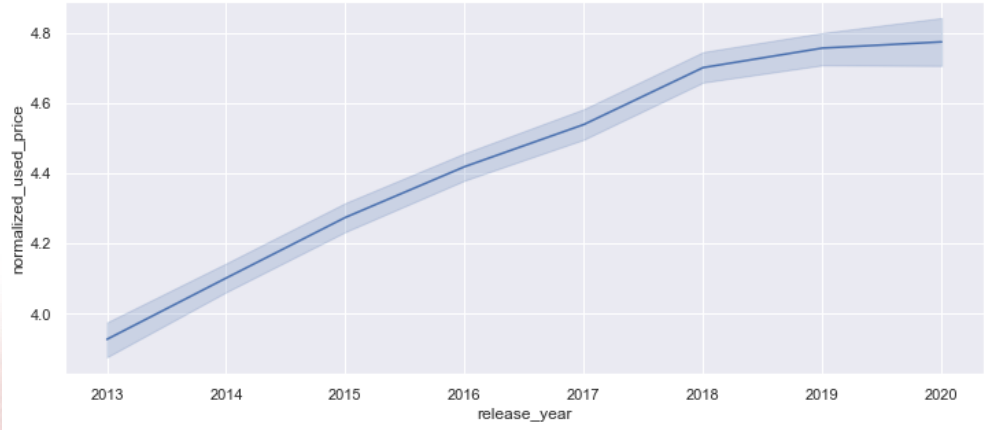
*Rear camera: main camera mp > 16

Observations

- Rear cameras generally have a better resolution than front cameras.
- **Sony devices have the best rear cameras representing 39.4% of total devices.**
- The other devices are significantly behind Sony.

BIVARIATE ANALYSIS

Normalized used price vs Release year

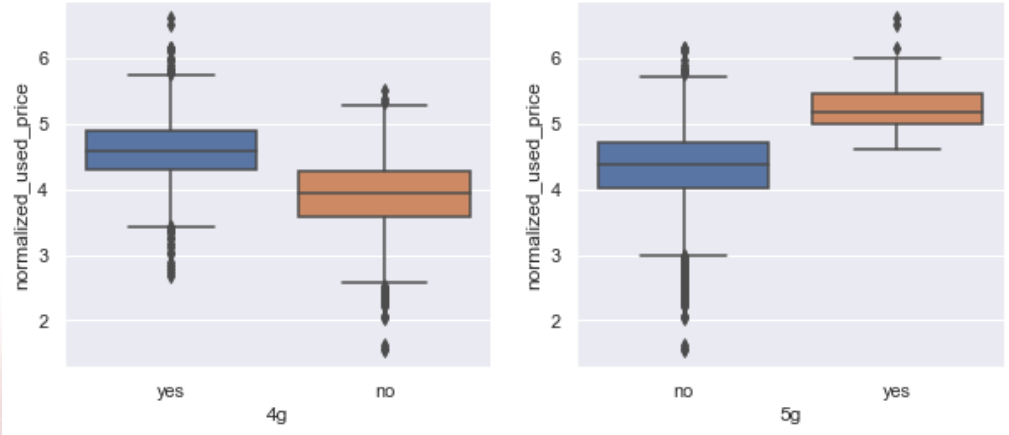


Observations

- **The normalized used price has had a steady increase over the years 2013 – 2020.**
- The normalized used price during the years 2018 – 2020 have not had significant variance. It ranges between 4.6 – 4.8.

BIVARIATE ANALYSIS

Normalized used price vs 4g / 5g



Observations

- The normalized used price for devices with 4G is higher than devices without it.
- Similarly, devices with 5G have higher normalized used price than the devices without 5G.
- **The prices are also higher for devices with 5G than devices with 4G.**

4. Data Preprocessing

DUPLICATE VALUE CHECK

Checking for duplicate values

In [7]:
```
data[data.duplicated()] ## Complete the code to check duplicate entries in the data
```

Out[7]:

brand_name	os	screen_size	4g	5g	main_camera_mp	selfie_camera_mp	int_memory	ram	battery	w

No duplicate values

MISSING VALUE TREATMENT

```
In [40]: # checking for missing values
         df1.isnull().sum() ## Complete the

Out[40]: brand_name                0
         os                        0
         screen_size               0
         4g                        0
         5g                        0
         main_camera_mp          179
         selfie_camera_mp          2
         int_memory                4
         ram                       4
         battery                   6
         weight                    7
         release_year              0
         days_used                 0
         normalized_used_price     0
         normalized_new_price      0
         dtype: int64
```

- There are **6 missing values**
- Imputing the missing values in the data by the **column medians** grouped by release_year and brand_name.

```
Out[41]: brand_name                0
         os                        0
         screen_size               0
         4g                        0
         5g                        0
         main_camera_mp          179
         selfie_camera_mp          2
         int_memory                0
         ram                       0
         battery                   6
         weight                    7
         release_year              0
         days_used                 0
         normalized_used_price     0
         normalized_new_price      0
         dtype: int64
```

- int_memory and ram has no more missing values.
- There are still **4 missing values**.
- Imputing the remaining missing values in the data by the **column medians** grouped by brand_name

```
Out[42]: brand_name                0
         os                        0
         screen_size               0
         4g                        0
         5g                        0
         main_camera_mp           10
         selfie_camera_mp          0
         int_memory                0
         ram                       0
         battery                   0
         weight                    0
         release_year              0
         days_used                 0
         normalized_used_price     0
         normalized_new_price      0
         dtype: int64
```

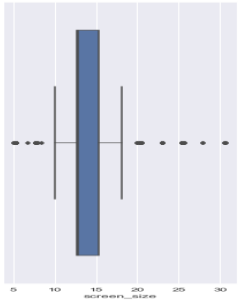
- **main_camera_mp** still has 10 missing values.
- Fill the remaining missing values in the main_camera_mp column by the **column median**.

```
Out[43]: brand_name                0
         os                        0
         screen_size               0
         4g                        0
         5g                        0
         main_camera_mp            0
         selfie_camera_mp          0
         int_memory                0
         ram                       0
         battery                   0
         weight                    0
         release_year              0
         days_used                 0
         normalized_used_price     0
         normalized_new_price      0
         dtype: int64
```

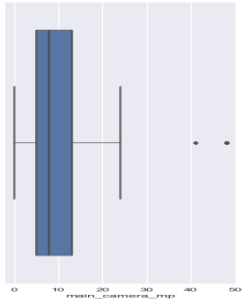
- No more missing values

OUTLIER CHECK

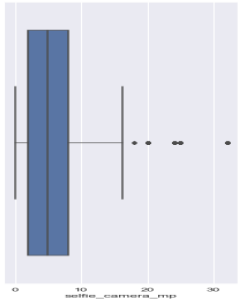
screen_size



main_camera_mp

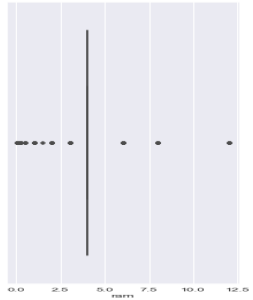


selfie_camera_mp

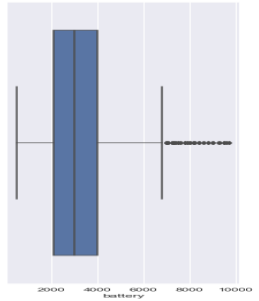


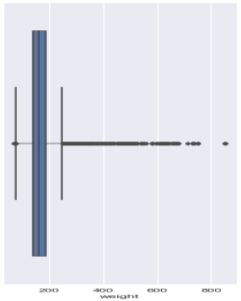
int_memory

ram

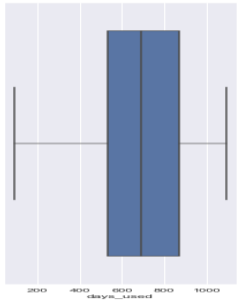


battery

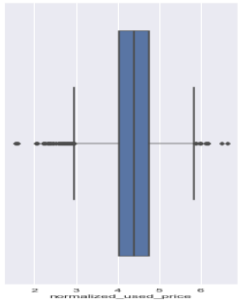




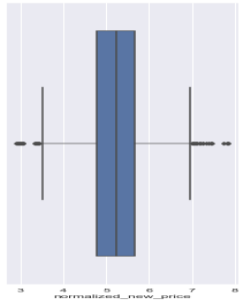
weight



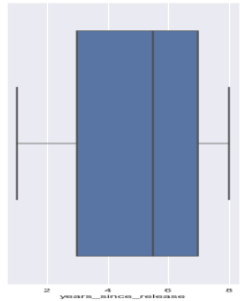
days_used



normalized_used_price



normalized_new_price



years_since_release

All numerical variables except for years_since_release has outliers.

FEATURE ENGINEERING

```
In [44]: df1["years_since_release"] = 2021 - df1["release_year"]
         df1.drop("release_year", axis=1, inplace=True)
         df1["years_since_release"].describe()

Out[44]: count    3454.000000
         mean        5.034742
         std         2.298455
         min         1.000000
         25%         3.000000
         50%         5.500000
         75%         7.000000
         max         8.000000
         Name: years_since_release, dtype: float64
```

- Creating a new column "**years_since_release**".
- Base year = 2021
- years_since_release = 2021 - release_year column
- The oldest devices were released 8 years back.
- The newest devices were released only a year back.
- 50% of the devices were released less than 5.5 years back

```
In [45]: df1.head()
```

Out[45]:

5g	main_camera_mp	selfie_camera_mp	int_memory	ram	battery	weight	days_used	normalized_used_price	normalized_new_price	years_since_release
no	13.0	5.0	64.0	3.0	3020.0	146.0	127	4.307572	4.715100	1
yes	13.0	16.0	128.0	8.0	4300.0	213.0	325	5.162097	5.519018	1
yes	13.0	8.0	128.0	8.0	4200.0	213.0	162	5.111084	5.884631	1
yes	13.0	8.0	64.0	6.0	7250.0	480.0	345	5.135387	5.630961	1
no	13.0	8.0	64.0	3.0	5000.0	185.0	293	4.389995	4.947837	1

DATA PREPARATION FOR MODELING

1

Define dependent & independent variables

```
In [47]:
## Complete the code to define the dependent an
X = df1.drop('normalized_used_price',axis=1)
y = df1['normalized_used_price']

print(X.head())
print()
print(y.head())
```

Independent variable: **X**

Dependent variable: **y** (Normalized used price)

2

Add intercept / constant

```
In [48]:
# let's add the intercept to data
X = sm.add_constant(X)
```

3

Create dummy variables

```
In [49]:
# creating dummy variables
X = pd.get_dummies(
    X,
    columns=X.select_dtypes(include=["object", "category"]).columns.tolist(),
    drop_first=True,
)  ## Complete the code to create dummies for independent features

X.head()
```

Dummy variables created using "**one hot encoding**"

Categorical variables: **brand_name, os, 4g, 5g**

DATA PREPARATION FOR MODELING

4

Splitting data (70:30 ratio)

```
In [50]: # splitting the data in 70:30 ratio for train to test data

x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=1
) ## Complete the code to split the data into train and test
```

5

Checking number of rows in train and test data

```
In [51]: print("Number of rows in train data =", x_train.shape[0])
         print("Number of rows in test data =", x_test.shape[0])

         Number of rows in train data = 2417
         Number of rows in test data = 1037
```

5. Model Performance Summary

OVERVIEW OF ML MODEL

- The model uses **95% confidence interval.**
- The model is able to explain **84%** of the variation in the data.
- If the **main camera mp** increases by 1 unit, the used price increases by 0.0210 units, all other variables held constant.
- If the **selfie camera mp** increases by 1 unit, the used price increases by 0.0138 units, all other variables held constant.
- If the **ram** increases by 1 unit, the used price increases by 0.0207 units, all other variables held constant.
- If the **weight** increases by 1 unit, the used price increases by 0.0017 units, all other variables held constant.
- If the **years since release** increases by 1 unit, the used price decreases by 0.0292 units, all other variables held constant.
- The normalized used price of devices with **4G** will be 0.0502 units more than those without 4G.
- The normalized used price of devices with **5G** will be 0.0673 units less than those without 5G.

OLS Regression Results

Dep. Variable:	normalized_used_price	R-squared:	0.839
Model:	OLS	Adj. R-squared:	0.838
Method:	Least Squares	F-statistic:	895.7
Date:	Tue, 22 Nov 2022	Prob (F-statistic):	0.00
Time:	13:31:18	Log-Likelihood:	80.645
No. Observations:	2417	AIC:	-131.3
Df Residuals:	2402	BIC:	-44.44
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.5000	0.048	30.955	0.000	1.405	1.595
main_camera_mp	0.0210	0.001	14.714	0.000	0.018	0.024
selfie_camera_mp	0.0138	0.001	12.858	0.000	0.012	0.016
ram	0.0207	0.005	4.151	0.000	0.011	0.030
weight	0.0017	6e-05	27.672	0.000	0.002	0.002
normalized_new_price	0.4415	0.011	39.337	0.000	0.419	0.463
years_since_release	-0.0292	0.003	-8.589	0.000	-0.036	-0.023
brand_name_Karbonn	0.1156	0.055	2.111	0.035	0.008	0.223
brand_name_Samsung	-0.0374	0.016	-2.270	0.023	-0.070	-0.005
brand_name_Sony	-0.0670	0.030	-2.197	0.028	-0.127	-0.007
brand_name_Xiaomi	0.0801	0.026	3.114	0.002	0.030	0.130
os_Others	-0.1276	0.027	-4.667	0.000	-0.181	-0.074
os_iOS	-0.0900	0.045	-1.994	0.046	-0.179	-0.002
4g_yes	0.0502	0.015	3.326	0.001	0.021	0.080
5g_yes	-0.0673	0.031	-2.194	0.028	-0.127	-0.007

Omnibus:	246.183	Durbin-Watson:	1.902
Prob(Omnibus):	0.000	Jarque-Bera (JB):	483.879
Skew:	-0.658	Prob(JB):	8.45e-106
Kurtosis:	4.753	Cond. No.	2.39e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.39e+03. This might indicate that there are strong multicollinearity or other numerical problems.

PARAMETERS OF ML MODEL

Equation of linear regression

normalized_used_price = 1.5000 + 0.0210 * (main_camera_mp) + 0.0138 * (selfie_camera_mp) + 0.0207 * (ram) + 0.0017 * (weight) + 0.4415 * (normalized_new_price) + -0.0292 * (years_since_release) + 0.1156 * (brand_name_Karbonn) + -0.0374 * (brand_name_Samsung) + -0.06670 * (brand_name_Sony) + 0.0801 * (brand_name_Xiaomi) + -0.1276 * (os_Others) + -0.0900 * (os_iOS) + 0.0502 * (4g_yes) + -0.0673 * (5g_yes)

ML Model Parameters

```
In [74]: olsmodel_final.params

Out[74]: const                  1.499981
         main_camera_mp         0.020967
         selfie_camera_mp       0.013827
         ram                    0.020713
         weight                 0.001662
         normalized_new_price   0.441472
         years_since_release   -0.029170
         brand_name_Karbonn     0.115598
         brand_name_Samsung    -0.037423
         brand_name_Sony       -0.066993
         brand_name_Xiaomi      0.080068
         os_Others             -0.127575
         os_iOS                -0.090006
         4g_yes                 0.050213
         5g_yes                -0.067259
         dtype: float64
```

KEY PERFORMANCE METRICS

Performance metrics	Training data	Testing data
RMSE	0.23403	0.241434
MAE	0.182751	0.186649
R-squared	0.83924	0.838387
Adj. R-squared	0.838235	0.836013
MAPE	4.395407	4.556349

- The model is able to explain **84%** of the variation in the data.
- The train and test **RMSE and MAE are low** and comparable. So, the model is not overfitting.
- The **MAPE** on the test data suggests that we can predict within **4.56%** of the normalized used price.
- Hence, we can conclude that the model olsmodel_final is good for prediction as well as inferences.

6. Appendix

DATA BACKGROUND & CONTENTS

Data Dictionary

SL. No.	Variable	Description
1	brand_name	Name of manufacturing brand
2	os	OS on which the device runs
3	screen_size	Size of screen (cm)
4	4g	4G is available or not
5	5g	5G is available or not
6	main_camera_mp	Resolution of rear camera (megapixels)
7	selfie_camera_mp	Resolution of front camera (megapixels)
8	int_memory	Amount of internal memory ROM (GB)
9	ram	Amount of RAM (GB)
10	battery	Energy capacity of device battery (mAh)
11	weight	Weight of device (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	normalized_new_price	Normalized price of a new device of the same model (euros)
15	normalized_used_price	Normalized price of the used/refurbished device (euros)

The data contains the different attributes of used/refurbished phones and tablets. The data was collected in the year **2021**.

DATA BACKGROUND & CONTENTS

Data Overview

Shape of the dataset

Rows	3454
Columns	15

Datatypes

integer	2
float	9
Object	4

Statistical summary

	count	mean	std	min	25%	50%	75%	max
screen_size	3454	13.71	3.81	5.08	12.7	12.83	15.34	30.71
main_camera_mp	3275	9.46	4.82	0.08	5	8	13	48
selfie_camera_mp	3452	6.55	6.97	0	2	5	8	32
int_memory	3450	54.57	84.97	0.01	16	32	64	1024
ram	3450	4.04	1.37	0.02	4	4	4	12
battery	3448	3133.4	1299.68	500	2100	3000	4000	9720
weight	3447	182.75	88.41	69	142	160	185	855
release_year	3454	2015.97	2.30	2013	2014	2015.5	2018	2020
days_used	3454	674.87	248.58	91	533.5	690.5	868.75	1094
normalized_used_price	3454	4.36	0.59	1.54	4.03	4.41	4.76	6.62
normalized_new_price	3454	5.23	0.68	2.90	4.79	5.25	5.67	7.85

MODEL ASSUMPTIONS

TEST FOR MULTICOLLINEARITY

1

Check VIFs > 5

- screen_size & weight have higher VIFs.

	feature	VIF
0	const	227.744081
1	screen_size	7.677290
2	main_camera_mp	2.285051
3	selfie_camera_mp	2.812473
4	int_memory	1.364152
5	ram	2.282352
6	battery	4.081780
7	weight	6.396749
8	days_used	2.660269

2

Remove multicollinearity

- Compare adj. R-squared for screen_size & weight.
- Drop the column which has the least difference from the actual adj. R-squared.

	col	Adj. R-squared after_dropping col	RMSE after dropping col
0	screen_size	0.838381	0.234703
1	weight	0.838071	0.234928

- Therefore, we should drop screen_size.

3

Drop variables with p-values > 0.05

- Variables after dropping:

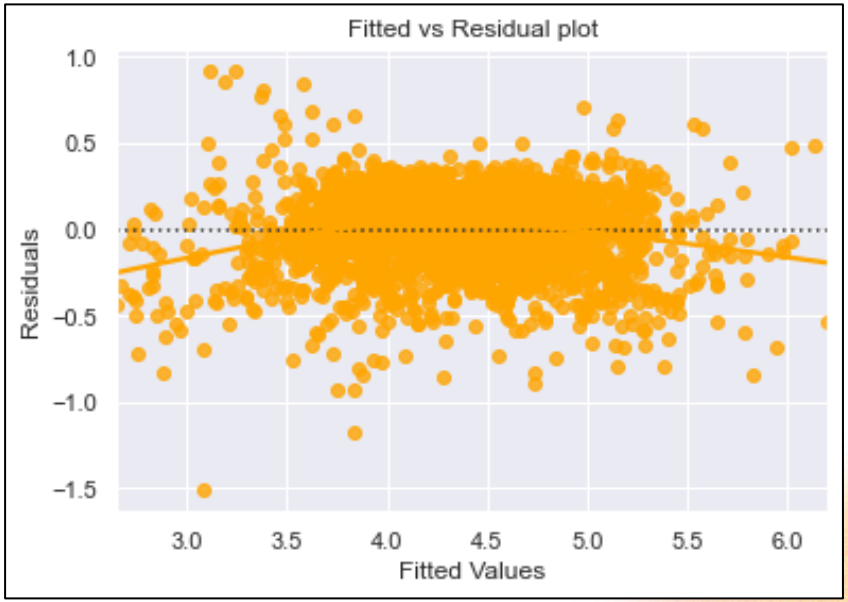
	coef	std err	t	P>|t|
const	1.5000	0.048	30.955	0.000
main_camera_mp	0.0210	0.001	14.714	0.000
selfie_camera_mp	0.0138	0.001	12.858	0.000
ram	0.0207	0.005	4.151	0.000
weight	0.0017	6e-05	27.672	0.000
normalized_new_price	0.4415	0.011	39.337	0.000
years_since_release	-0.0292	0.003	-8.589	0.000
brand_name_Karbonn	0.1156	0.055	2.111	0.035
brand_name_Samsung	-0.0374	0.016	-2.270	0.023
brand_name_Sony	-0.0670	0.030	-2.197	0.028
brand_name_Xiaomi	0.0801	0.026	3.114	0.002
os_Others	-0.1276	0.027	-4.667	0.000
os_iOS	-0.0900	0.045	-1.994	0.046
4g_yes	0.0502	0.015	3.326	0.001
5g_yes	-0.0673	0.031	-2.194	0.028

MODEL ASSUMPTIONS

TEST FOR LINEARITY & INDEPENDENCE

- Test for linearity and independence is performed by making a plot of fitted values vs residuals (on the right) and checking for patterns.
- Since the plot does not show any specific patterns, it can be concluded that the model is linear and the residuals are independent.

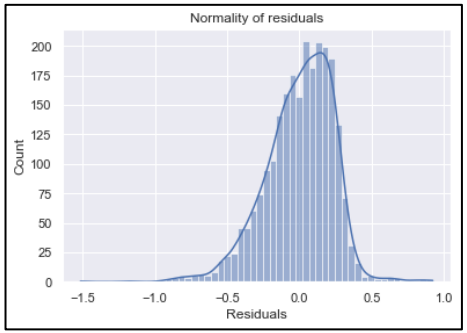
	Actual Values	Fitted Values	Residuals
3026	4.087488	3.867319	0.220169
1525	4.448399	4.602001	-0.153602
1128	4.315353	4.286957	0.028395
3003	4.282068	4.195169	0.086899
2907	4.456438	4.490563	-0.034125



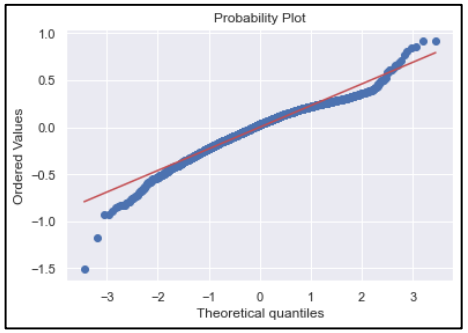
MODEL ASSUMPTIONS

TEST FOR NORMALITY

Distribution of residuals



Q-Q plot of residuals



Shapiro-Wilk test

```
In [71]: stats.shapiro(df_pred['Residuals']) ## Complete the code to apply the Shap
Out[71]: ShapiroResult(statistic=0.9676972031593323, pvalue=6.995328206686811e-23)
```

- The residual terms are normally distributed.
- Most of the plots are lying on a straight line in QQ plot.
- Since p-value < 0.05, the residuals are not normal as per shapiro test.
- Strictly speaking - the residuals are not normal. However, as an approximation, we might be willing to accept this distribution as close to being normal

MODEL ASSUMPTIONS

TEST FOR HOMOSCEDASTICITY

- **Homoscedacity** - Variance of the residuals are symmetrically distributed across the regression line
- **Heteroscedacity** - variance is unequal for the residuals across the regression line

Goldfeldquandt test (p > 0.05)

Null hypothesis: Residuals are homoscedastic

Alternate hypothesis: Residuals have hetroscedasticity

```
In [73]: import statsmodels.stats.api as sms
         from statsmodels.compat import lzip

         name = ["F statistic", "p-value"]
         test = sms.het_goldfeldquandt(df_pred["Residuals"], x_train3) ## Comple
         lzip(name, test)

Out[73]: [('F statistic', 1.0087504199106763), ('p-value', 0.4401970650667071)]
```

- p-value = 0.44
- p-value > 0.05
- Fail to reject null hypothesis
- Residuals are homoscedastic

THANK YOU!