SLF-Project-Presentation

Recell Project and PGP DSBA

08/02/2023

Contents / Agenda

Executive Summary

- **Actionable Insights:-**

- Newly released phones have high used price which is relatively correct as new the phone the higher new price thus used phones would be affected by this. As the older the phone the lower used price as most customers wants phones in demand.

- Phones with 4g and Gionee brands phones have lower the used price. They are not in demand so may be it can discontinue.

- Operating systems of devices other than Android and IOS and windows have negative coefficients. As they increase price of used device decreases.

Executive Summary

- **Recommendations:-**

- Future data collection needs to be done on the age of customer purchasing products since age can be a major drive.

- Future data collection on income can also be an important factor as we can even go through the thing as in more income group people what features and everything they prefer to buy.

- 5g network phones have high resale price and should be in demand and focused more rather than 4g phones.

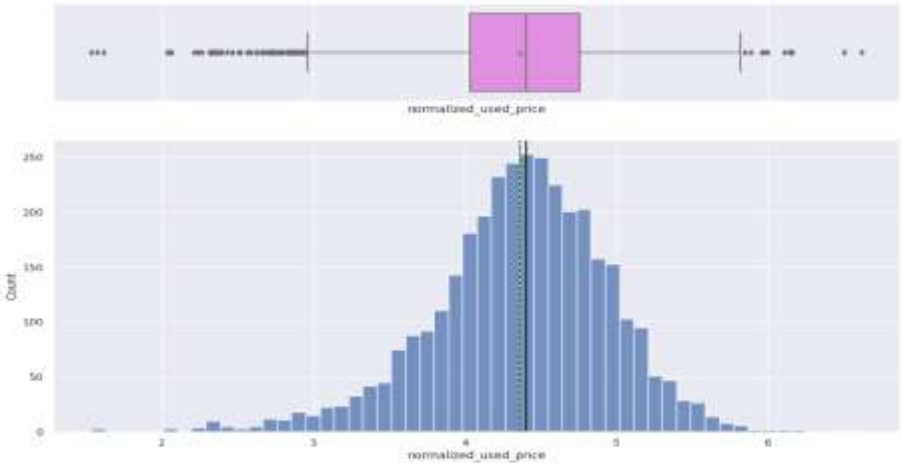
Business Problem Overview and Solution Approach

- **Define the problem:-**
  - Recell is a startup company which is engaged in buying and selling of used phones and tablets. Over the past decade the market of used and refurbished device has grown considerably.
  - The company is aiming to tap in the rising up the potential market so they need an ML based solution to develop a dynamic strategy for used and refurbished devices.
- **Solution approach:-**
  - The company has hired a data scientist to analyze the data , build the linear regression model and to predict price of used phone and tablet and make them aware about the factors that influence it.
  - The company has even collected the data of used phone and tablets which includes brand name, OS, screen size, 4g, 5g ram, battery, weight and many other factors that influence the market.

EDA Results

- **Univariate Analysis**
  - Function for histogram and boxplot with labeled of every data point.

1. **Normalized-used price:-**

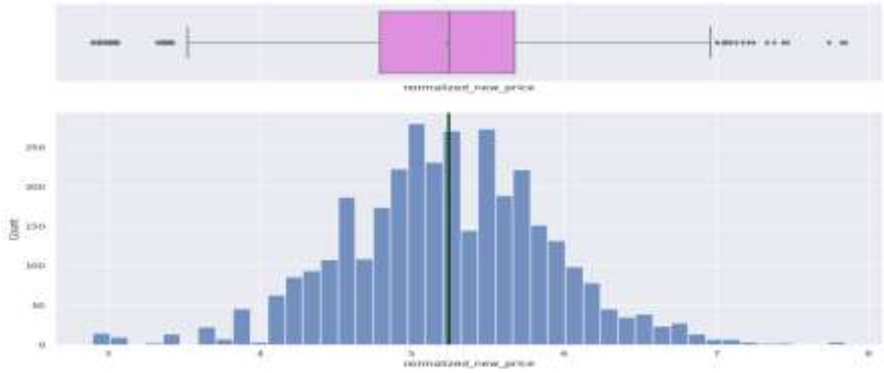


Observations:-

- The distribution of used phone price is normally distributed. Used phones are comparatively expensive to others.

EDA Results

2. Normalized new price:-

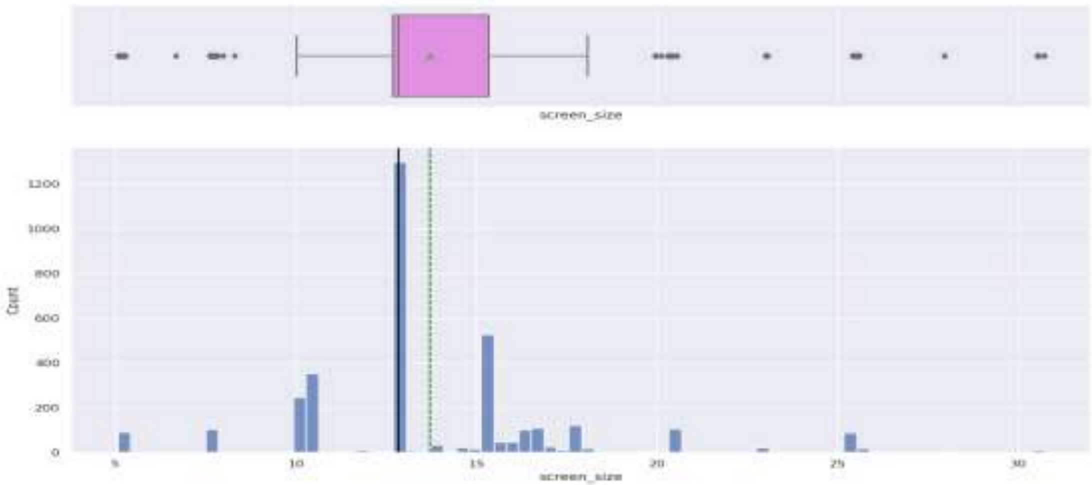


Observations:-

- The distribution of new price is almost at a normal distribution.

EDA Results

3. Screen Size:-

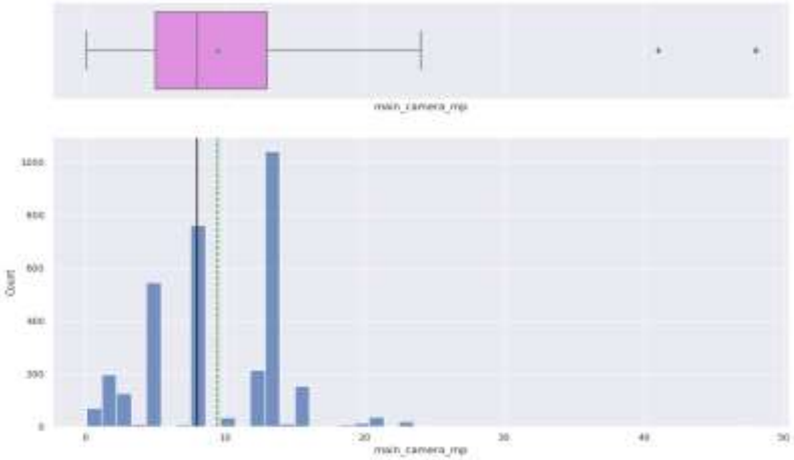


Observations:-

- The distribution is seen to be rightly skewed with outliners on both sides . It has median about 15cm.

EDA Results

4. Main Camera mp:-

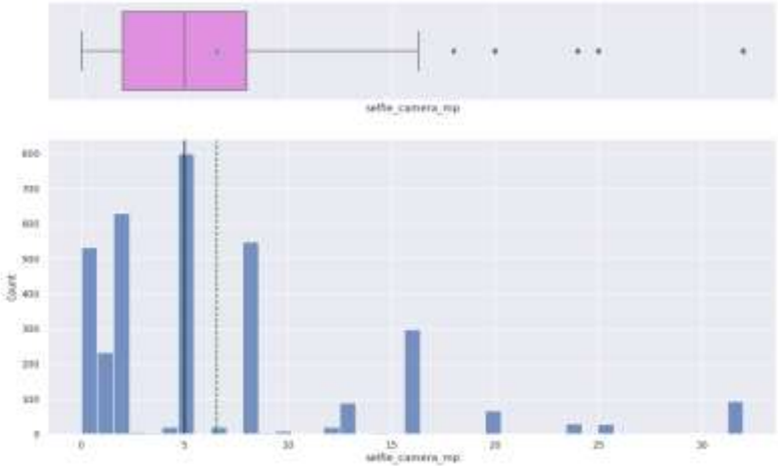


Observation:-

- The main camera pixels are mostly normally distributed.

EDA Results

5. Selfie Camera mp:-

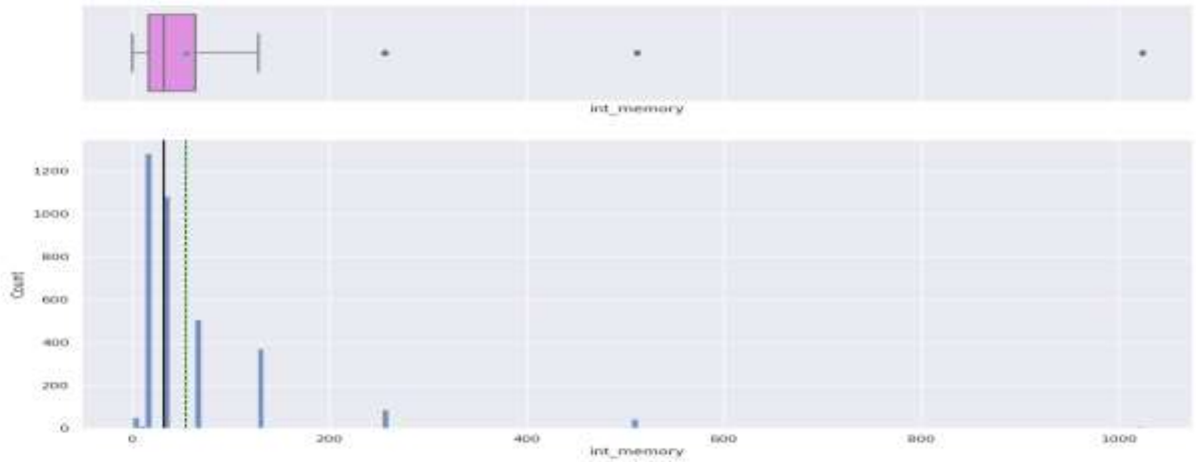


Observation:-

- The distribution of selfie camera pixels is rightly skewed.

EDA Results

6. Int memory:-

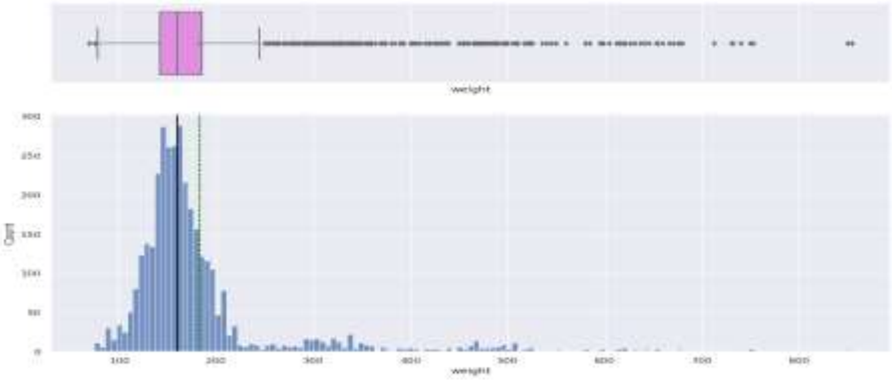


Observation:-

- The distribution of internal memory is rightly skewed.

EDA Results

7. Weight:-

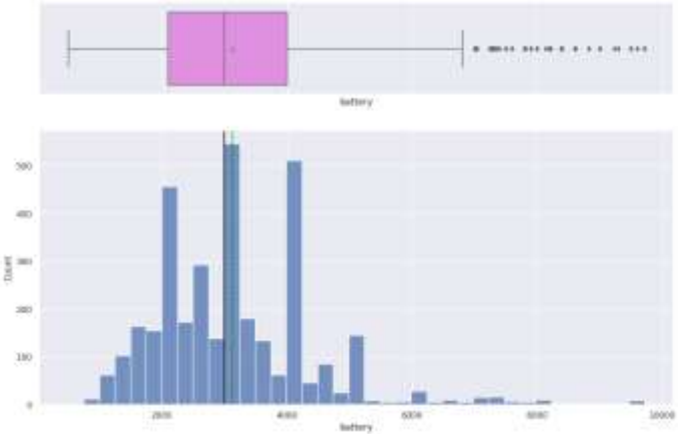


Observation:-

- The weight of column is right skewed. Further during model building this will help to reduce the skewness.

EDA Results

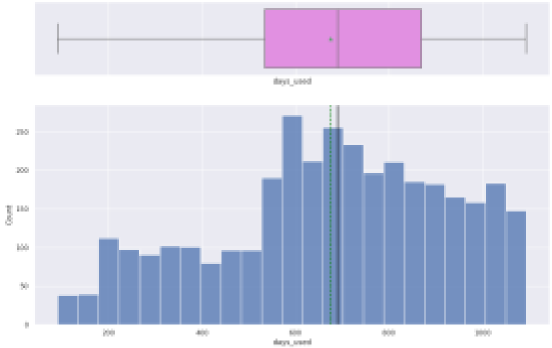
8. Battery:-



Observation:-

There is a right skewness in battery column.

9. Days Used:-

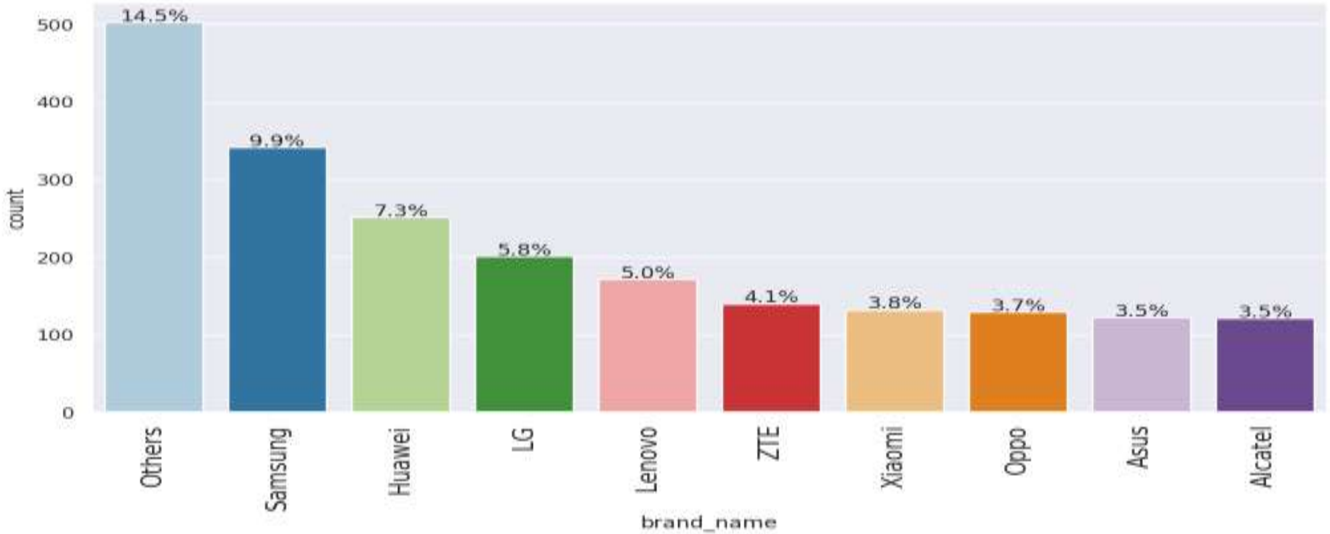


Observation:-

The days are normally distributed.

EDA Results

Labelled Brand name:-

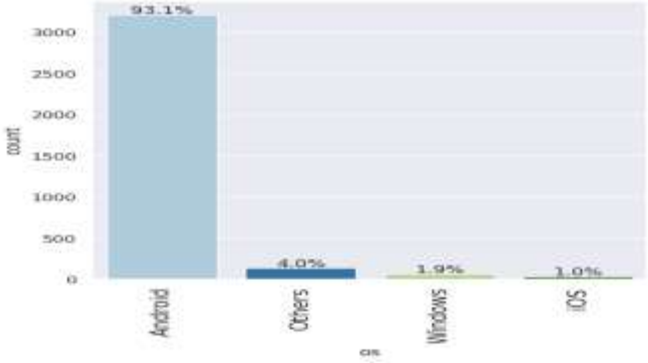


Observation:-

- Most of the brand names were not given and they fall under category of others.
- Samsung have a higher percentage compared to others. This means customers buy Samsung refurbished phones more.

EDA Results

- Operating System:-

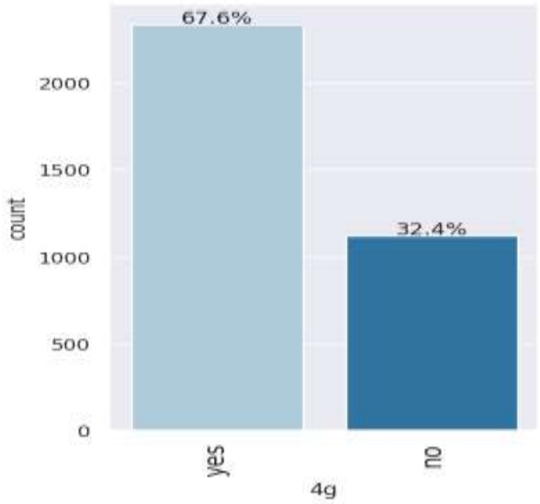


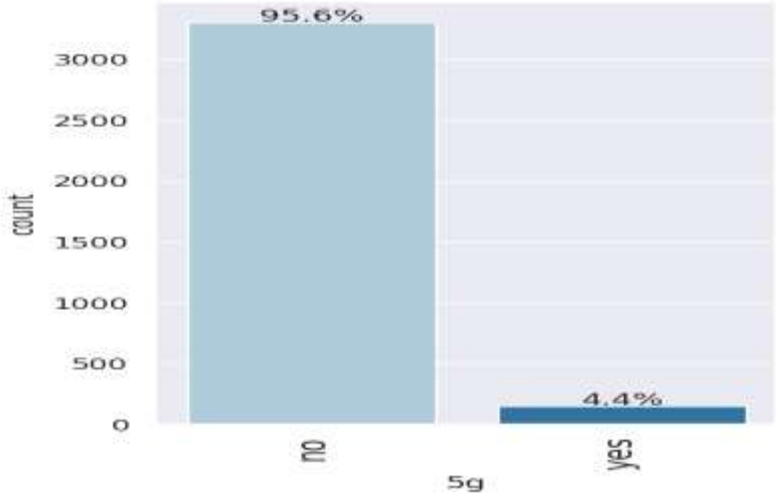
Observation:-

- In refurbished market android device are most refurbished one with a 93.1%
- IOS devices are the least refurbished ones with 1.0%.

EDA Results

- 4G and 5G:-



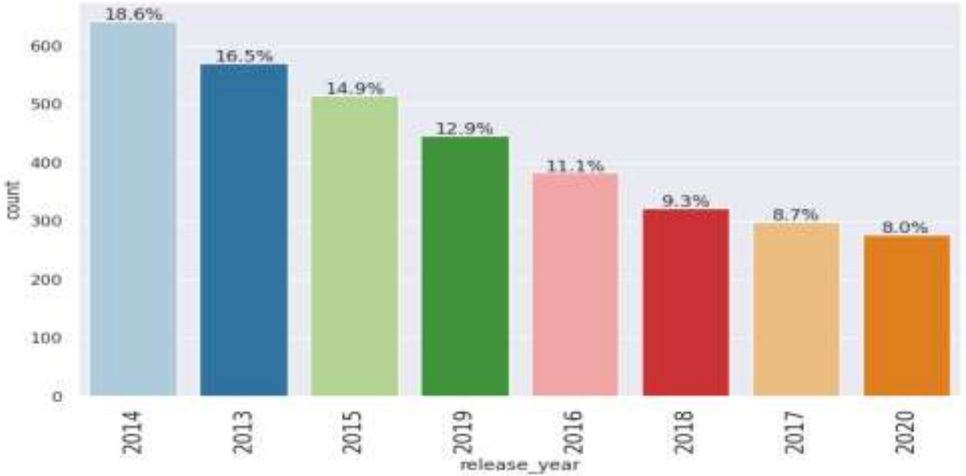


Observations:-

- If we compare the devices with the 5G and 4G a lot of 4G devices were refurbished as compared to 5G.

EDA Results

- Release Year:-



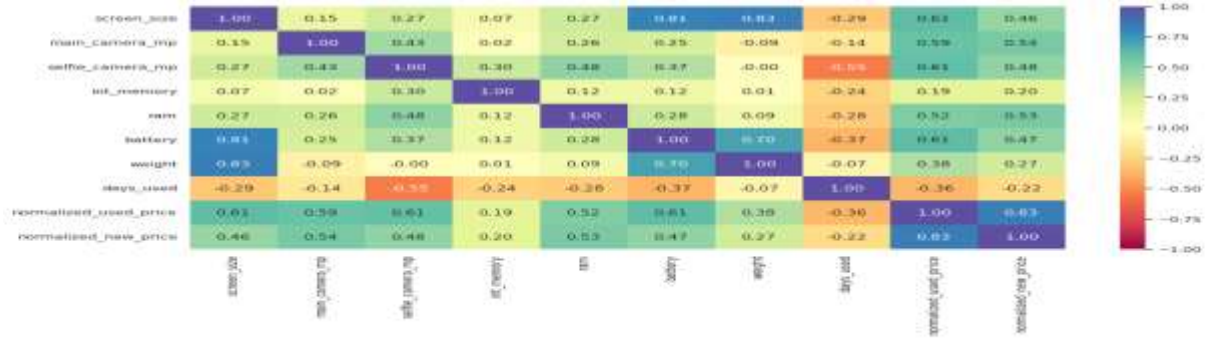
Observation:-

- Devices released in 2014 were the most refurbished ones.

EDA Results

- **Bivariate Analysis**

❖ Correlation Check:

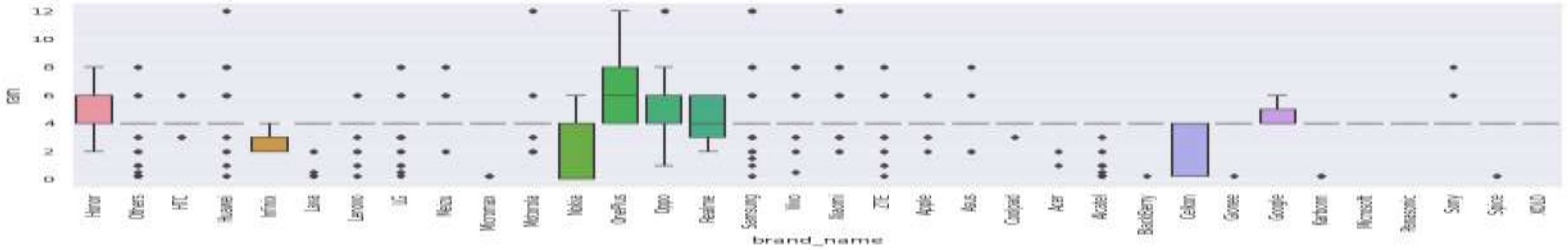


Observation:-

- Weight and screen size is highly correlated
- Battery and screen size is highly correlated
- Negative Correlation between days used and Selfie camera.
- Negative Correlation between days used and normalized used price.

EDA Results

- **The amount of RAM is important for the smooth functioning of a device. Let's see how the amount of RAM varies across brands.**

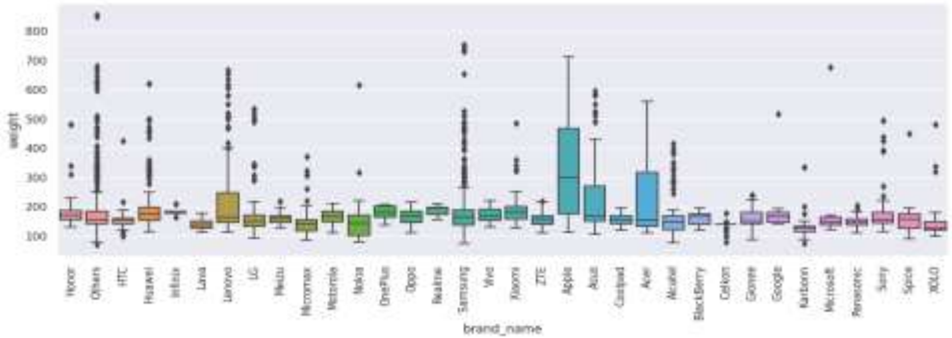


Observation:-

- One plus device gives more RAM across different brands.

EDA Results

- **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. Let's create a new dataframe of only those devices which offer a large battery and analyze.**



Observation:-

- Apple devices has larger battery with a large energy capacity.

EDA Results

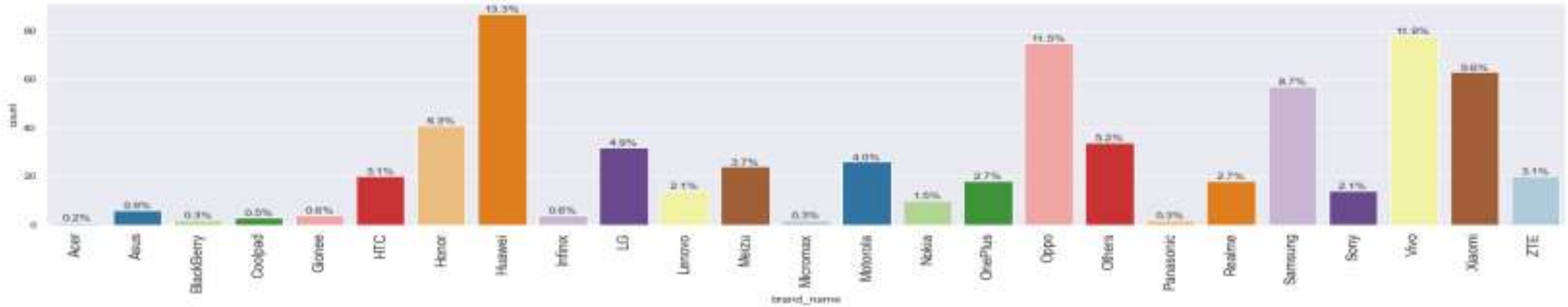
- **People who buy phones and tablets primarily for entertainment purposes prefer a large screen as they offer a better viewing experience. Let's create a new dataframe of only those devices which are suitable for such people and analyze.**

Observation:-

- Huawei brand name has highest percentage of devices with a screen size larger than 6 inches.

EDA Results

- **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. Let's create a new dataframe of only those devices which are suitable for this customer segment and analyze.**

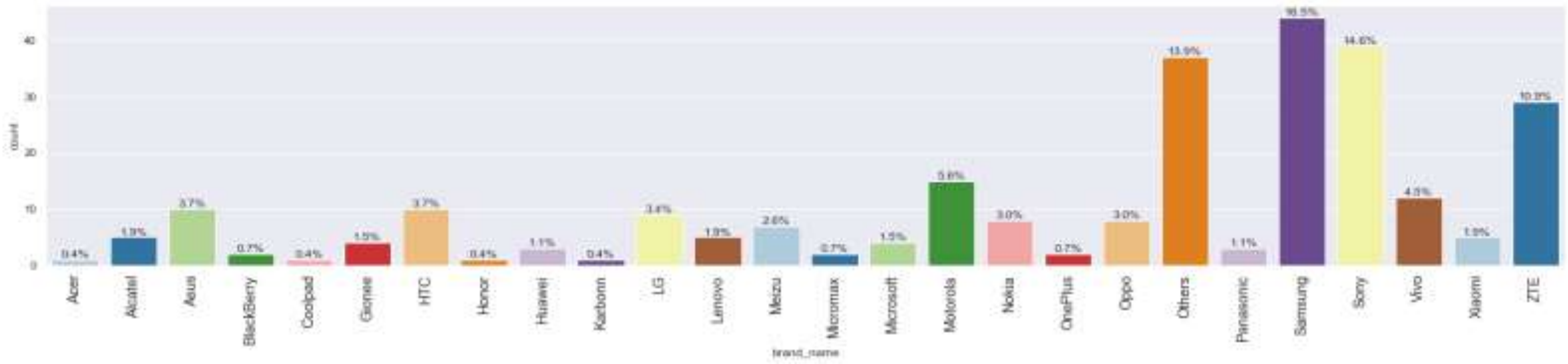


Observation:-

- Android offers the greatest number of devices with selfie camera mega pixels greater than 8. This could be the reason of biggest number in market.
- IOS device does not offer selfie camera mega pixels greater than 8.

EDA Results

Let's do a similar analysis for rear cameras.

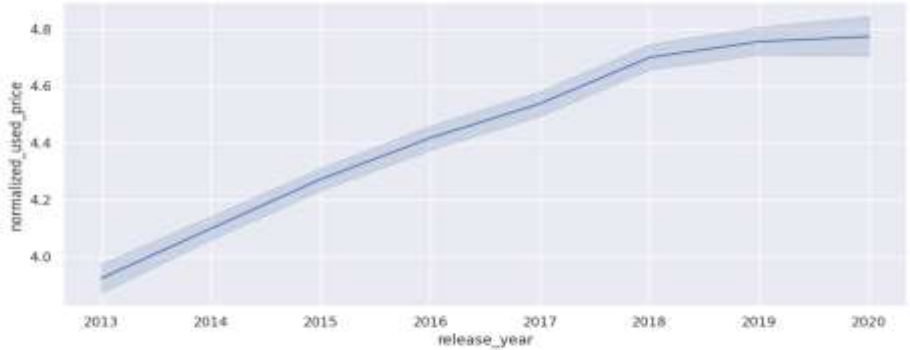


Observation:-

- Samsung has the biggest count for rear camera with a rate of pixels greater than 13.
- Android has the highest percentage with the rear camera with a rate of pixels greater than 13.

EDA Results

- **Let's see how the price of used devices varies across the years.**

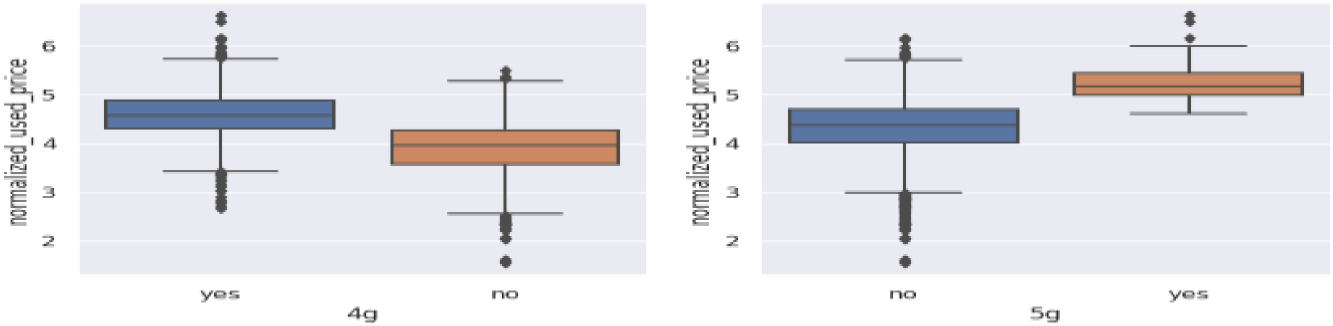


Observation:-

- The price of refurbished phones keeps increasing over years. As the year increases the price also increases.

EDA Results

- **Let's check how the prices vary for used phones and tablets offering 4G and 5G networks.**



Observation:-

- The prices for 4G for used price of phones and tablets has higher network people want to consider rather than 5G network.

Data Preprocessing

- **Missing value Imputation:-**

```
# checking for missing values
df1.isnull().sum() ## Complete the code to check missing values in all the columns
```

```
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
```

Observation:-

- There are certain missing values which will be imputed by different brand name and try to make it 0.

Data Preprocessing

- Let us impute missing values in the columns with median of the columns grouped by release year and brand name.

```
for col in cols_impute:
    df1[col] = df1[col].fillna(
        value=df1.groupby(['release_year'])[col].transform("median")
    )  ## Complete the code to impute missing values in cols_impute with median by grouping the data on release year and brand name

# checking for missing values
df1.isnull().sum() ## Complete the code to check missing values after imputing the above columns
```

```
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
```

Observation:-

- ○ We tried using the code and tried to make 0 missing values in release year and brand name.

Data Preprocessing

- We will fill the remaining missing values in the main_camera_mp column by the column median.

```
[10] df1["main_camera_mp"] = df1["main_camera_mp"].fillna(df1["main_camera_mp"].median()) ## Complete the code to impute the data with median

     # checking for missing values
     df1.isnull().sum() ## Complete the code to check missing values after imputing the above columns
```

```
brand_name              0
os                      0
screen_size             0
4g                      0
5g                      0
main_camera_mp          0
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
```

Observation:-

- We can observe here that main camera has no missing values.

Data Preprocessing

- **Feature Engineering:-**

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

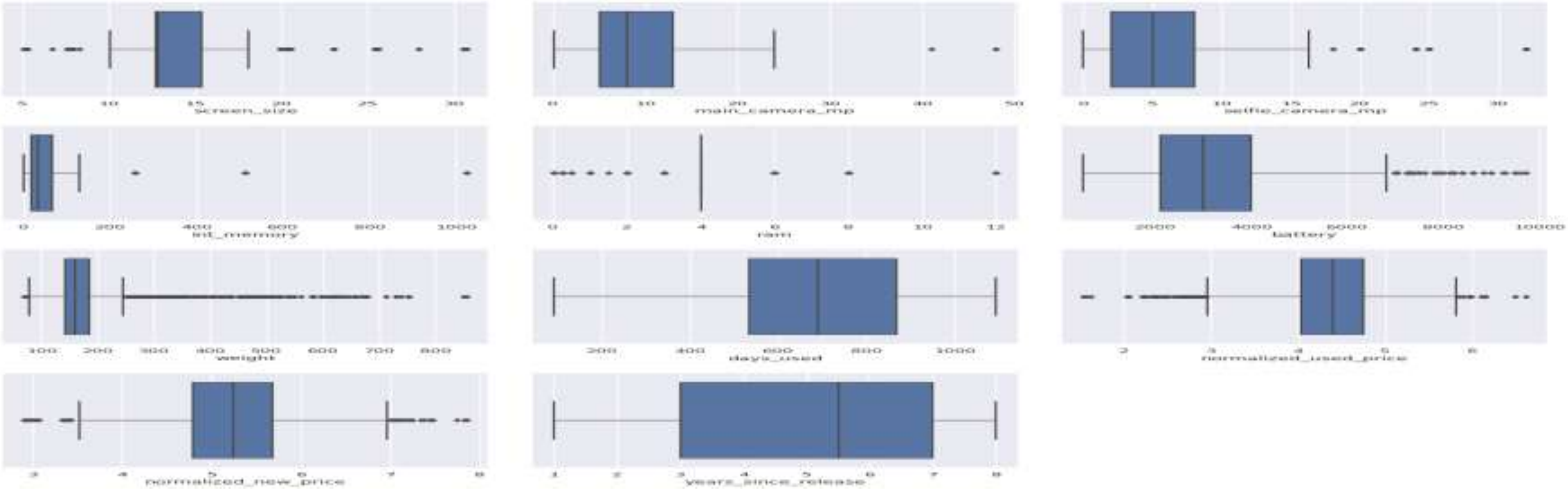
```
count    3454.000000
mean        5.034742
std         2.298455
min         1.000000
25%         3.000000
50%         5.500000
75%         7.000000
max         8.000000
Name: years_since_release, dtype: float64
```

Observation:-

- We observe here that release year is drop and after that we get to know the mean . Std with a max of 8 and min of 1.

Data Preprocessing

- **Outline check:-**



Observation:-

- All the outliners in the independent columns are treated rather than RAM column .We will omit RAM column as doing so it will remove the variation in column and most likely make it a constant which is not desirable so omitted.

Data Preprocessing

- Data Preparation for modeling:-

```
  brand_name       os  screen_size   4g   5g  main_camera_mp  \
0      Honor  Android        14.50  yes   no            13.0
1      Honor  Android        17.30  yes  yes            13.0
2      Honor  Android        16.69  yes  yes            13.0
3      Honor  Android        25.50  yes  yes            13.0
4      Honor  Android        15.32  yes   no            13.0

   selfie_camera_mp  int_memory  ram  battery  weight  release_year  \
0               5.0        64.0  3.0   3020.0   146.0          2020
1              16.0       128.0  8.0   4300.0   213.0          2020
2               8.0       128.0  8.0   4200.0   213.0          2020
3               8.0        64.0  6.0   7250.0   480.0          2020
4               8.0        64.0  3.0   5000.0   185.0          2020

   days_used  normalized_new_price
0        127              4.715100
1        325              5.519018
2        162              5.884631
3        345              5.630961
4        293              4.947837

0     4.307572
1     5.162097
2     5.111084
3     5.135387
4     4.389995
```

Observations:-

- Here are some snap shots of the code and the output which I got while preparation of the model which helped me to evaluate for same.

Data Preprocessing

```
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()
```

	const	screen_size	main_camera_mp	selfie_camera_mp	int_memory	ram	battery	weight	release_year	days_used	...	brand_name_Spice	brand_name_Vivo	brand_n
0	1.0	14.50	13.0	5.0	64.0	3.0	3020.0	146.0	2020	127	...	0	0	
1	1.0	17.30	13.0	16.0	128.0	8.0	4300.0	213.0	2020	325	...	0	0	
2	1.0	16.69	13.0	8.0	128.0	8.0	4200.0	213.0	2020	162	...	0	0	
3	1.0	25.50	13.0	8.0	64.0	6.0	7250.0	480.0	2020	345	...	0	0	
4	1.0	15.32	13.0	8.0	64.0	3.0	5000.0	185.0	2020	293	...	0	0	

5 rows × 49 columns

```
[15] # 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=42) ## Complete the code to split the data into train and test in specified
```

```
[16] 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
```

Model Performance Summary

- Model Building-Linear Regression:-

```
                              OLS Regression Results
==============================================================================
Dep. Variable:     normalized_used_price   R-squared:                       0.835
Model:                               OLS   Adj. R-squared:                  0.834
Method:                    Least Squares   F-statistic:                     812.6
Date:                   Fri, 01 Apr 2022   Prob (F-statistic):               0.00
Time:                           01:53:37   Log-Likelihood:                 52.372
No. Observations:                   2417   AIC:                            -72.74
Df Residuals:                       2401   BIC:                             19.90
Df Model:                             15
Covariance Type:               nonrobust
==========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    -0.0729      0.169     -0.430      0.667      -0.405       0.259
screen_size               0.0408      0.004     10.693      0.000       0.033       0.048
main_camera_mp            0.0208      0.002     13.350      0.000       0.018       0.024
selfie_camera_mp          0.0163      0.002      9.478      0.000       0.013       0.020
int_memory                0.0004      0.000      2.279      0.023    5.78e-05       0.001
ram                       0.0231      0.005      4.616      0.000       0.013       0.033
battery                1.133e-06   7.38e-06      0.153      0.878   -1.33e-05    1.56e-05
days_used              6.431e-05   3.11e-05      2.068      0.039    3.34e-06       0.000
normalized_new_price      0.4116      0.012     33.725      0.000       0.388       0.436
weight_log                0.2584      0.040      6.418      0.000       0.179       0.337
years_since_release      -0.0115      0.005     -2.420      0.016      -0.021      -0.002
os_Others                -0.0582      0.028     -2.050      0.040      -0.114      -0.003
os_Windows                0.0565      0.037      1.520      0.127      -0.016       0.129
os_iOS                    0.0147      0.046      0.322      0.747      -0.075       0.104
```

Observation:-

- Adjusted R-squared is equal to 0.835 which is a good value.
- The y-intercept is equal to the value of const coefficient which is -0.0729.
- The co-efficient of normalized new price is 0.4116.

Model Performance Summary

- **Model Performance Check:-**

Checking model performance on train set:-

Training Performance

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	0.236784	0.183228	0.835434	0.834337	4.414471

Checking model performance on test set:-

Test Performance

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	0.243144	0.18733	0.83609	0.833518	4.577868

Observation:-

- MAE suggests that model can predict the price of used device with a mean error of 0.187 in test set.
- MAPE of 4.577on test data means we are able to predict within 4.6% of used device prices.
- The training R-square is 0.835 so model is not under fitting.

Model Performance Summary

- **Checking Linear Regression Assumptions:-**

Test for multicollinearity:-

0	const	1227.232818
1	screen_size	5.020059
2	main_camera_mp	2.130616
3	selfie_camera_mp	3.613245
4	int_memory	2.149691
5	ram	2.061785
6	battery	3.511445
7	days_used	2.579919
8	normalized_new_price	2.795831
9	weight_log	4.297022
10	years_since_release	5.073806
11	os_Others	1.328570
12	os_Windows	1.023320
13	os_iOS	1.094783
14	4g_yes	2.294751
15	5g_yes	1.709624

Observation:-

If VIF is between 1 to 5 than there is low multicollinearity.

If VIF is between 5 to 10 than there is moderate multicollinearity.

If VIF is above 10 than there is high multicollinearity.

Thus screen size and years since release shows moderate collinearity.

Model Performance Summary

- **Dropping high p-value variables**

Battery, const, days used, OS Windows, OS IOS, and 5g have p-value>0.05 so they are not up to mark so we will drop them.

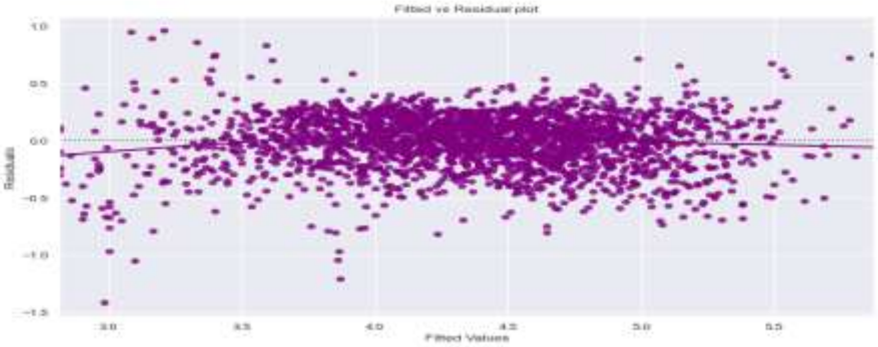
```
                              OLS Regression Results
==============================================================================
Dep. Variable:     normalized_used_price   R-squared:                       0.835
Model:                               OLS   Adj. R-squared:                  0.834
Method:                    Least Squares   F-statistic:                     1350.
Date:                   Fri, 01 Apr 2022   Prob (F-statistic):               0.00
Time:                           01:53:37   Log-Likelihood:                 46.606
No. Observations:                   2417   AIC:                            -73.21
Df Residuals:                       2407   BIC:                            -15.31
Df Model:                              9
Covariance Type:               nonrobust
==========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    -0.0826      0.157     -0.528      0.598      -0.390       0.224
screen_size               0.0417      0.003     11.912      0.000       0.035       0.049
main_camera_mp            0.0213      0.002     13.825      0.000       0.018       0.024
selfie_camera_mp          0.0173      0.002     11.335      0.000       0.014       0.020
int_memory                0.0004      0.000      2.320      0.020    6.24e-05       0.001
ram                       0.0194      0.004      4.372      0.000       0.011       0.028
normalized_new_price      0.4056      0.011     36.826      0.000       0.384       0.427
weight_log                0.2612      0.038      6.814      0.000       0.186       0.336
os_Others                -0.0672      0.028     -2.399      0.017      -0.122      -0.012
4g_yes                    0.0432      0.014      3.101      0.002       0.016       0.070
==============================================================================
```

Observation:-

- Now R-square is 0.834 so it shows that model is good.
- The adjusted R-square in olsmodel is 0.834 which shows that values we dropped are not affected to this variables.

Model Performance Summary

- **TEST FOR LINEARITY AND INDEPENDENCE:-**



	Actual Values	Fitted Values	Residuals
3026	4.087488	3.800785	0.286703
1525	4.448399	4.671627	-0.223227
1128	4.315353	4.312365	0.002987
3003	4.282068	4.203344	0.078724
2907	4.456438	4.494569	-0.038130

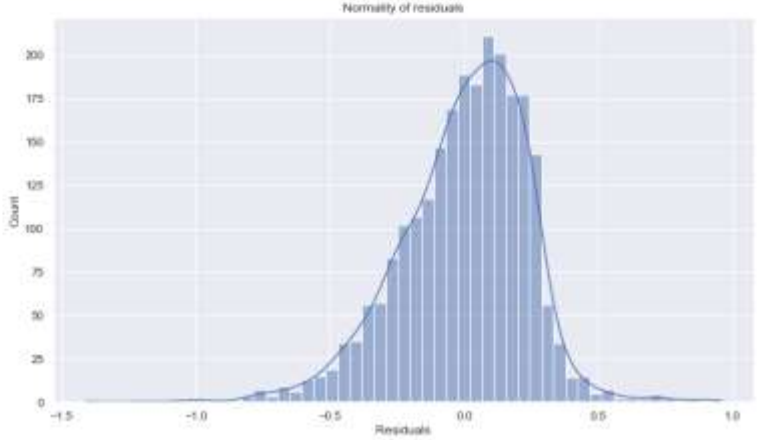
Observation:-

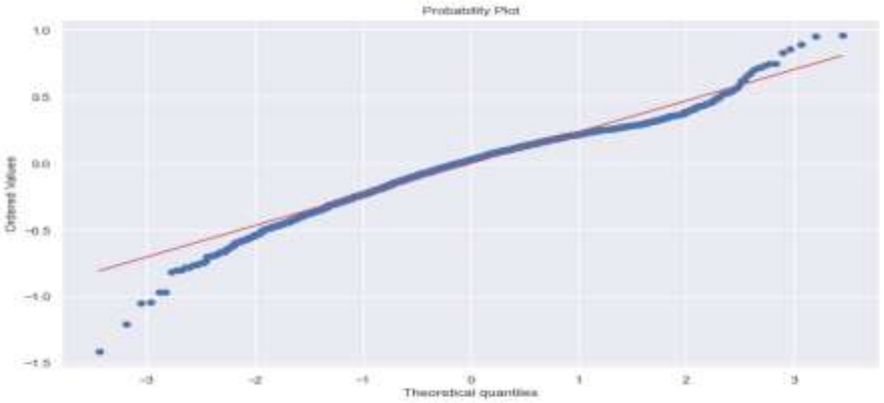
The scatter plot shows the distributions of residuals v/s fitted values.

There is no pattern so the test for linearity and independence assumption is satisfied.

Model Performance Summary

- **TEST FOR NORMALITY:-**





ShapiroResult(statistic=0.9711182117462158, pvalue=1.1405862401987462e-21)

Observation:-

- The histogram residuals have bell shaped curve.
- The probplot residuals have a straight line except for tails.
- P-value <0.05 so the residuals are not normal.
- From the above all instances the assumption is more or less satisfied.

Final Model Summary

- **Checking model performance for train and test data.**

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     normalized_used_price   R-squared:                       0.835
Model:                               OLS   Adj. R-squared:                  0.834
Method:                    Least Squares   F-statistic:                     1350.
Date:                   Fri, 01 Apr 2022   Prob (F-statistic):               0.00
Time:                           01:53:39   Log-Likelihood:                 46.606
No. Observations:                   2417   AIC:                            -73.21
Df Residuals:                       2407   BIC:                            -15.31
Df Model:                              9
Covariance Type:               nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                    -0.0826      0.157     -0.528      0.598      -0.390       0.224
screen_size               0.0417      0.003     11.912      0.000       0.035       0.049
main_camera_mp            0.0213      0.002     13.825      0.000       0.018       0.024
selfie_camera_mp          0.0173      0.002     11.335      0.000       0.014       0.020
int_memory                0.0004      0.000      2.320      0.020    6.24e-05       0.001
ram                       0.0194      0.004      4.372      0.000       0.011       0.028
normalized_new_price      0.4056      0.011     36.826      0.000       0.384       0.427
weight_log                0.2612      0.038      6.814      0.000       0.186       0.336
os_Others                -0.0672      0.028     -2.399      0.017      -0.122      -0.012
4g_yes                    0.0432      0.014      3.101      0.002       0.016       0.070
==============================================================================
Omnibus:                      217.210   Durbin-Watson:                   1.936
```

Training Performance

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	0.23735	0.183576	0.834647	0.83396	4.424958

Test Performance

	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	0.244273	0.188165	0.834565	0.832952	4.607551

Observation:-

- The model explains 82% of variation in data that is good.
- The train and test RSME and MAE is low that means it is not over fitting.
- MAE value is also low so that means we can predict the used device prices.

Happy Learning !