

ReCell Project

Post-Grad Program in Data Science and Business Analytics

Date: 9/25/2022

By: Manik Chhabra

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- EDA Results
- Data Preprocessing
- Model Performance Summary
- Appendix

Executive Summary

- The model can explain 84% of the variation in data and within 4.5% of the used devices price on the test data, which is exceptionally good
 - Hence, we can conclude that the model is good for prediction and interpretation
- If the **main_camera_mp** raises by one unit, then the normalized used device price increases by 0.0210, all other variables held constant
- If the **selfie_camera_mp** raises by one unit, then the normalized used device price increases by 0.0138, all other variables held constant
 - The company ReCell can buy higher megapixel cameras (selfie and main) to resell the used product at a higher price
- If the **ram** raises by one unit, then the normalized used device price increases by 0.0207, all other variables held constant
 - The company ReCell can buy more used products with higher ram to resell it at a higher price
- If the **weight** raises by one unit, then the normalized used device price increases by 0.0017, all other variables held constant
 - Buying phones with higher weight can increase the resell value of the phone
- If the **normalized_new_price** raises by one unit, then the normalized used device price increases by 0.4415, all other variables held constant
- According to the results from bivariate analysis, devices that use **4G** or **5G** network have higher **normalized used prices**; therefore the company ReCell can buy devices that are only **4G** or **5G** to earn higher profits

Business Problem Overview and Solution Approach

- Business Problem
 - ReCell wants to hire a data scientist that can build a linear regression model to predict the price of a used phone/tablet and identify factors that significantly influence it
- The solution approach / methodology
 - The solution to the business problem is to first discover a relationship between the normalized used device price and the other variables in the dataset through exploratory data analysis
 - Second, preprocess the data before building the linear regression model
 - Test all 5 assumptions for linear regression modelling
 - Lastly build a model that has an exceptionally good fit and is not overfitting nor underfitting based on the comparison between the training and testing data

EDA Results

- Univariate Analysis
 - Variable - **Normalized Used Price**
 - According to **Figure 1**, the distribution for **normalized used price** is skewed to the left
 - The mean and median for the **normalized used price** is respectively 4.36 Euros and 4.41 Euros
 - Outliers exist on both sides of the Boxplot
 - Variable – **Normalized New Price**
 - According to **Figure 2**, the distribution for **normalized new price** appears evenly distributed
 - The mean and median for the **normalized new price** is respectively 5.23 Euros and 5.25 Euros
 - Outliers exist on both sides of the Boxplot

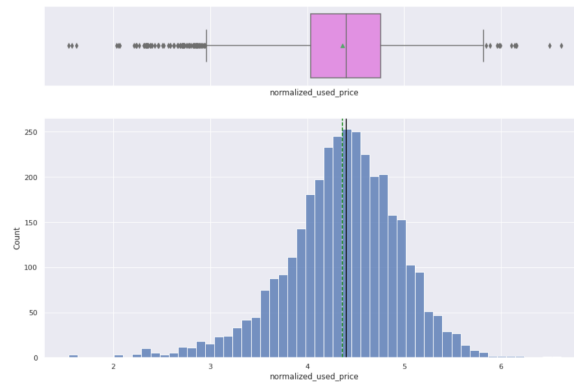


Figure 1

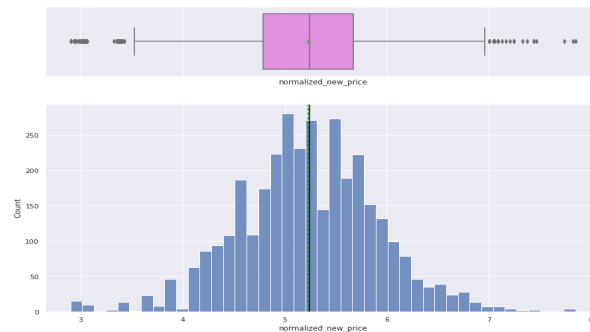


Figure 2

[Link to Appendix slide on data background check](#)

EDA Results

- Univariate Analysis
 - Variable – **Screen Size**
 - According to **Figure 3**, the distribution for **screen size** is skewed to the right
 - The mean and median for the **screen size** is respectively 13.71cm and 12.83 cm
 - Outliers exist on both sides of the Boxplot
 - Variable – **Main Camera MP**
 - According to **Figure 4**, the distribution for **main camera mp** is heavily skewed to the right
 - The mean and median for **main camera mp** is respectively 9.46 mp and 8.00 mp
 - Outliers exist beyond the max value of the boxplot

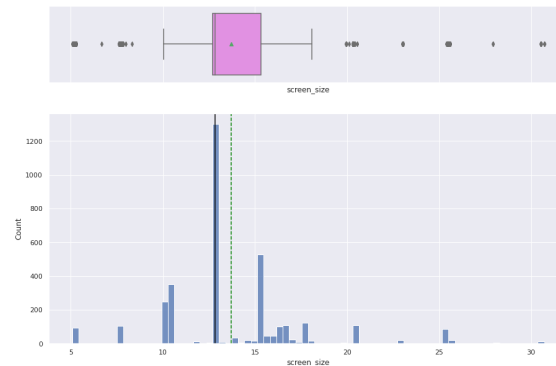


Figure 3

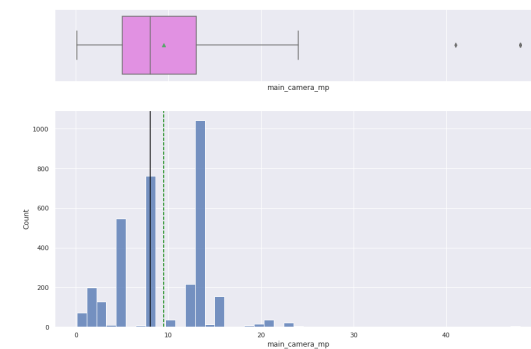


Figure 4

[Link to Appendix slide on data background check](#)

EDA Results

- Univariate Analysis
 - Variable – **Selfie Camera MP**
 - According to **Figure 5**, the distribution for **Selfie Camera MP** is heavily skewed to the right
 - The mean and median for the **Selfie Camera MP** is respectively 6.55 mp and 5 mp
 - Outliers exist beyond the max value of the boxplot
 - Variable – **Internal Memory**
 - According to **Figure 6**, the distribution for **Internal Memory** is heavily skewed to the right
 - The mean and median for **Internal memory** is respectively 54.57 GB and 32 GB
 - Outliers exist beyond the max value of the boxplot

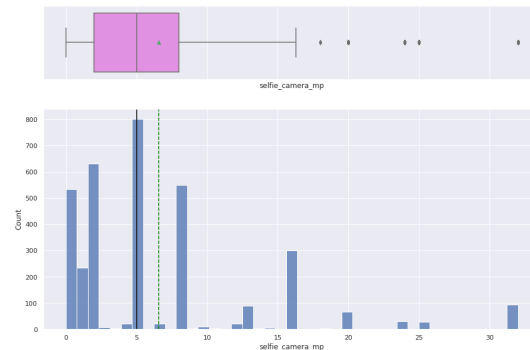


Figure 5

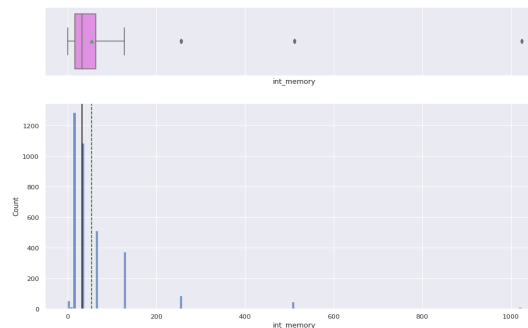


Figure 6

[Link to Appendix slide on data background check](#)

EDA Results

- Univariate Analysis
 - Variable – **Ram**
 - According to **Figure 7**, the distribution for **Ram** is unevenly distributed; A boxplot was not formed
 - The mean and median for the **Ram** is respectively 4.04 GB and 4 GB
 - Variable – **Weight**
 - According to **Figure 8**, the distribution for **Weight** is heavily skewed to the right
 - The mean and median for **Weight** is respectively 182.75 g and 160 g
 - Outliers exist beyond the max value of the boxplot

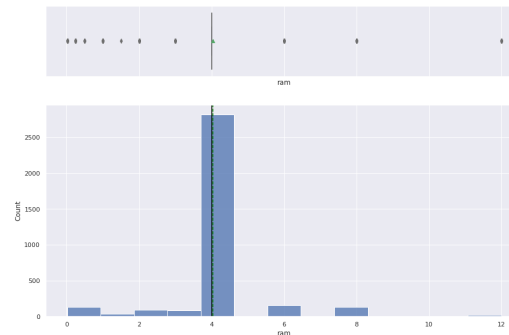


Figure 7

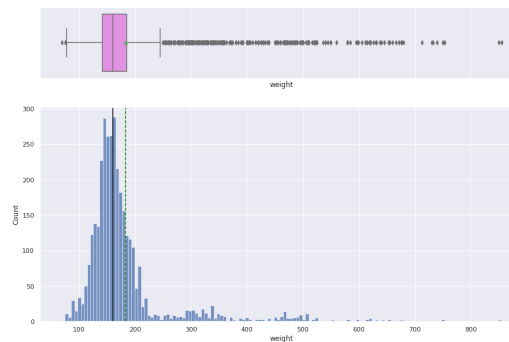


Figure 8

[Link to Appendix slide on data background check](#)

EDA Results

- Univariate Analysis
 - Variable – **Battery**
 - According to **Figure 9**, the distribution for **Battery** is heavily skewed to the right
 - The mean and median for the **Battery** is respectively 3133.40 mAh and 3000 mAh
 - Outliers exist beyond the max value of the boxplot
 - Variable – **Days Used**
 - According to **Figure 10**, the distribution for **Days Used** is skewed to the left
 - The mean and median for **Days Used** is respectively 674.87 and 690.50
 - Outliers do not exist for this variable

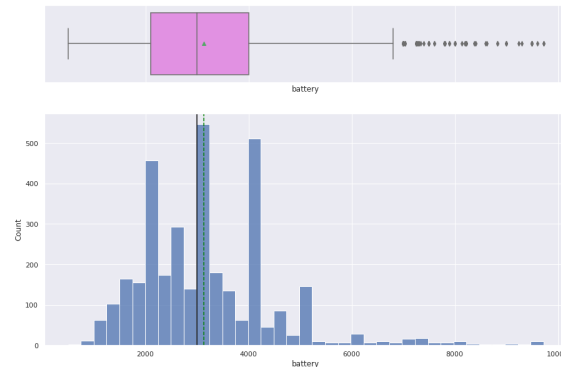


Figure 9

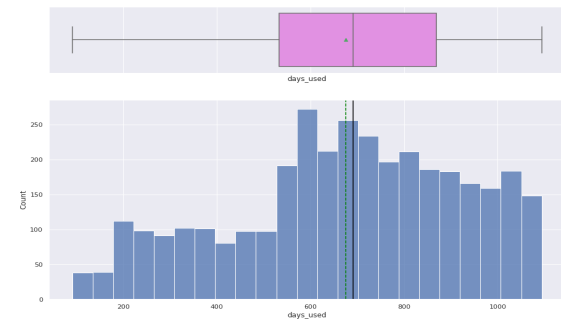


Figure 10

[Link to Appendix slide on data background check](#)

EDA Results

- Univariate Analysis

- Variable – **Brand Name**

- According to **Figure 11**, the top 3 brands in the data set (in terms of count) are **Others** (14.5%), **Samsung** (9.9%), and **Huawei** (7.3%).

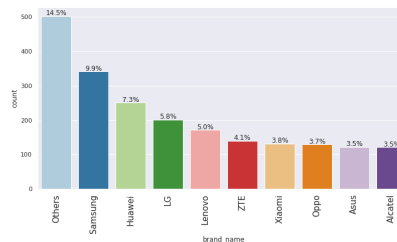


Figure 11

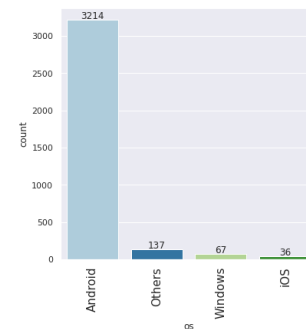


Figure 12

- Variable – **OS**

- According to **Figure 12**, most of the phones/tablets in the dataset run on the **Android OS** (3214)
 - **iOS** phones/tablets have the lowest count (36) in the data set

- Variable – **4G**

- According to **Figure 13**, there are more than twice as much **4G** phones/tablets (2335) than non-**4G** phones/tablets (1119)

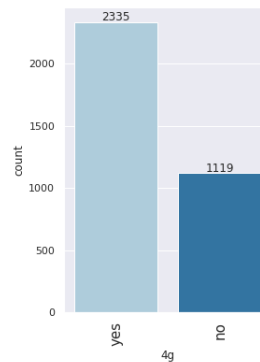


Figure 13

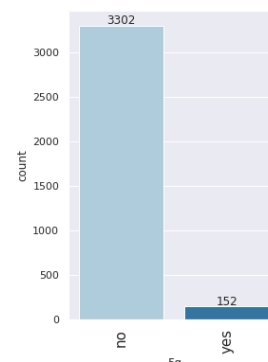


Figure 14

- Variable – **5G**

- According to **Figure 14**, 3302 phones/tablets are not run on **5G**
 - 152 phones/tablets are run on **5G**

[Link to Appendix slide on data background check](#)

EDA Results

- Univariate Analysis
 - Variable – **Release Year**
 - According to **Figure 15**, the top 3 amount of phones/tablets in the data set were released in 2014 (642), 2013 (570) , and 2015 (515)
 - The least amount of phones in the dataset were released in 2020 (277).

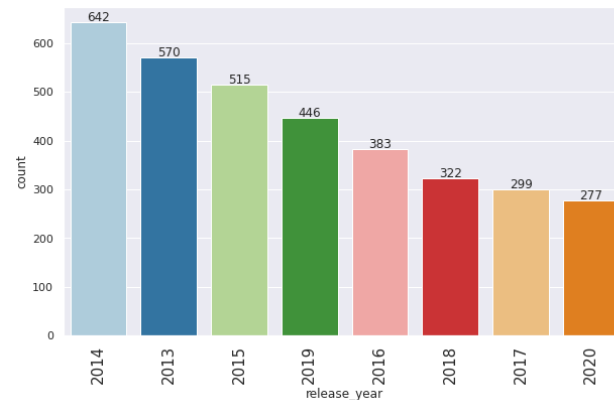


Figure 15

[Link to Appendix slide on data background check](#)

EDA Results

- Bivariate Analysis

- Heat Map

- According to **Figure 16**, 'Selfie camera mp', 'Screen_size', and 'Battery' are moderately positively correlated with 'normalized_used_price' – 0.61 correlation
- 'Main_camera_mp' is moderately positively correlated with 'normalized_used_price' – 0.59
- 'Normalized_used_price' and 'normalized_new_price' are highly positively correlated with each other – 0.83

- Release year vs Normalized Used Price – Figure 17

- A strong positive correlation exists between **release year** and **normalized used price** from 2013-2018
 - Used phone prices increase with the newer years
- After 2018, there was a weaker positive correlation(2019- 2020 – no correlation existed) between **release year** and **normalized used price**

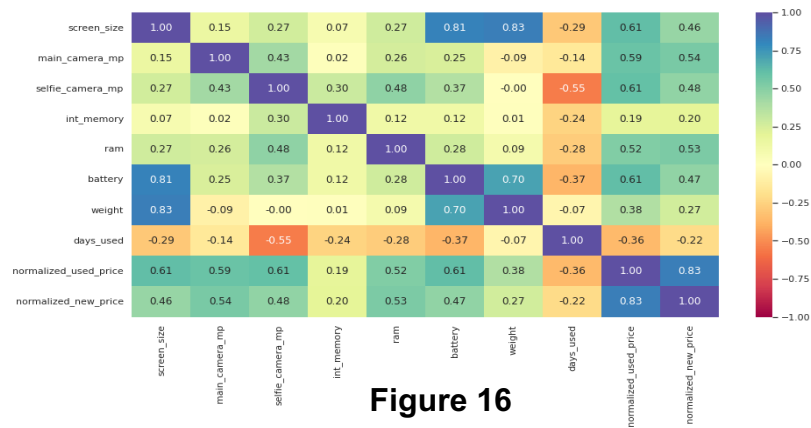


Figure 16

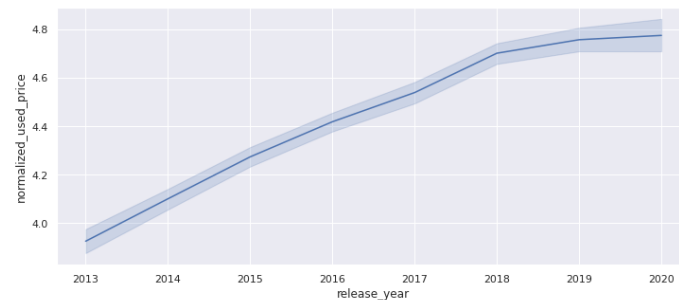


Figure 17

[Link to Appendix slide on data background check](#)

EDA Results

- Bivariate Analysis
 - **4G vs Normalized Used Price – Figure 18**
 - According to the Boxplot, normalized used tablets/phones prices are higher for tablets/phones that use 4G network (~4.5 euros) vs non-4G network (~4 euros)
 - **5G vs Normalized Used Price – Figure 19**
 - According to the Boxplot, normalized used tablets/phones prices are higher for tablets/phones that use 5G network (~5.2 euros) vs non-5G network (~4.5 euros)

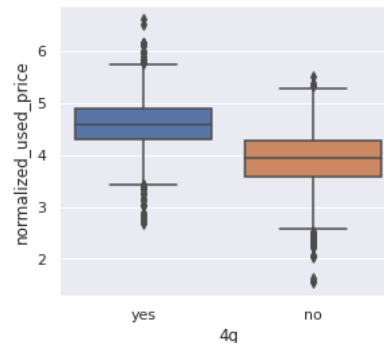


Figure 18

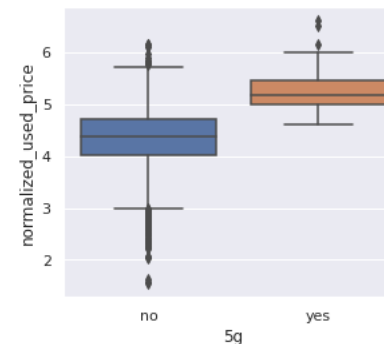


Figure 19

[Link to Appendix slide on data background check](#)

Data Preprocessing

- Duplicate value check
- Missing value treatment
- Outlier check (treatment if needed)
- Feature engineering
- Data preparation for modeling

Data Preprocessing

- Duplicate value check
 - No duplicate values exist in the dataset
- Missing Value Treatment
 - Used the code below to check for missing values in the dataset across all columns; Missing value is showing below per column:

```
# 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	

Data Preprocessing

● Missing Value Treatment

- After 3 rounds of imputing the missing values with the column medians grouped by 'brand_name' and 'release_year', by 'brand_name', and no grouping, no missing values appeared for each column after round 3

Round 1

```
cols_impute = ["main_camera_mp",
               "selfie_camera_mp",
               "int_memory",
               "ram",
               "battery",
               "weight"]

for col in cols_impute:
    df[col] = df[col].fillna(value=df1.groupby(['release_year', "brand_name"])[col].transform("median"))

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

Round 3

```
df1["main_camera_mp"] = df1["main_camera_mp"].fillna(df1["main_camera_mp"].median())

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

Round 2

```
cols_impute = [
    "main_camera_mp",
    "selfie_camera_mp",
    "battery",
    "weight",
]

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

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


Data Preprocessing

- Outlier Check
 - Outliers exist for most of the variables except 'release_year' and 'days_used'
 - However, they will not be treated as they are proper values

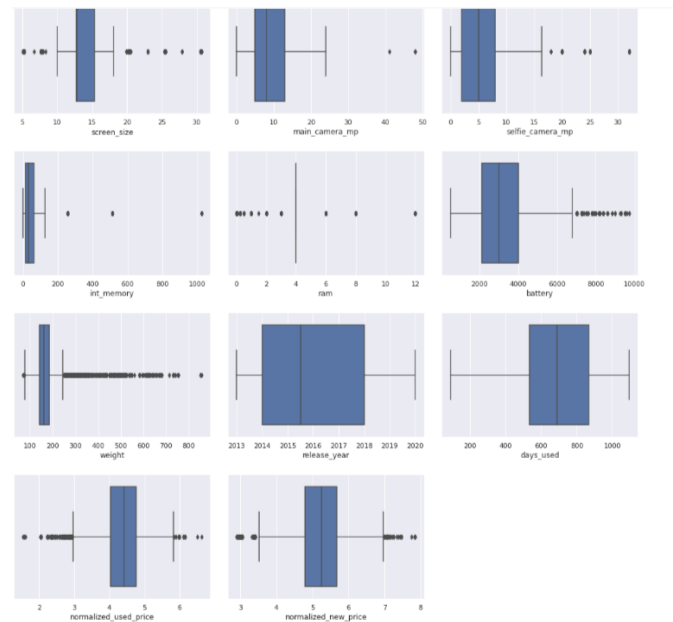


Figure 20

Data Preprocessing

- Feature Engineering
 - 'years_since_release' column was created from the 'release_year' column
 - 'Release_year' was then subsequently removed
 - In **Figure 21**, is a statistical summary of the new column, 'years_since_release':
 - Number of years since release varies from 1-8.
 - Mean is 5 years

Figure 21

```
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
```

Data Preprocessing

- Data Preparation for Modeling

- Purpose is to predict the normalized price of used devices
- Defined the dependent and independent variables in **Figure 22**

- y (dependent variable) = 'normalized used price' values
- X (independent variable) = all the predictor variables that are not 'normalized used price'

- Added the intercept to the data in **Figure 23**
- Encoded categorical features of the dataset in **Figure 24**
 - Dummy values were created for the categorical variables, 'brand_name', 'os,' '4g', and '5g' before building the model

Figure 22

```
## Complete the code to define the dependent and independent variables
X = df1.drop('normalized_used_price',axis=1)
y = df1['normalized_used_price']

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

Figure 23

```
# let's add the intercept to data
X = sm.add_constant(X)
```

Figure 24

```
# 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()
```

	screen_size	screen_size_cat	os	os_cat	ram	ram_cat	weight	weight_cat	days_used	normalized_used_price	brand_name	brand_name_cat	brand_name_2500	brand_name_3160	brand_name_3770	os_0others	os_1android	os_101	os_102	os_103	os_104	os_105	os_106	os_107	os_108	os_109	os_110	os_111	os_112	os_113	os_114	os_115	os_116	os_117	os_118	os_119	os_120	os_121	os_122	os_123	os_124	os_125	os_126	os_127	os_128	os_129	os_130	os_131	os_132	os_133	os_134	os_135	os_136	os_137	os_138	os_139	os_140	os_141	os_142	os_143	os_144	os_145	os_146	os_147	os_148	os_149	os_150	os_151	os_152	os_153	os_154	os_155	os_156	os_157	os_158	os_159	os_160	os_161	os_162	os_163	os_164	os_165	os_166	os_167	os_168	os_169	os_170	os_171	os_172	os_173	os_174	os_175	os_176	os_177	os_178	os_179	os_180	os_181	os_182	os_183	os_184	os_185	os_186	os_187	os_188	os_189	os_190	os_191	os_192	os_193	os_194	os_195	os_196	os_197	os_198	os_199	os_200	os_201	os_202	os_203	os_204	os_205	os_206	os_207	os_208	os_209	os_210	os_211	os_212	os_213	os_214	os_215	os_216	os_217	os_218	os_219	os_220	os_221	os_222	os_223	os_224	os_225	os_226	os_227	os_228	os_229	os_230	os_231	os_232	os_233	os_234	os_235	os_236	os_237	os_238	os_239	os_240	os_241	os_242	os_243	os_244	os_245	os_246	os_247	os_248	os_249	os_250	os_251	os_252	os_253	os_254	os_255	os_256	os_257	os_258	os_259	os_260	os_261	os_262	os_263	os_264	os_265	os_266	os_267	os_268	os_269	os_270	os_271	os_272	os_273	os_274	os_275	os_276	os_277	os_278	os_279	os_280	os_281	os_282	os_283	os_284	os_285	os_286	os_287	os_288	os_289	os_290	os_291	os_292	os_293	os_294	os_295	os_296	os_297	os_298	os_299	os_300	os_301	os_302	os_303	os_304	os_305	os_306	os_307	os_308	os_309	os_310	os_311	os_312	os_313	os_314	os_315	os_316	os_317	os_318	os_319	os_320	os_321	os_322	os_323	os_324	os_325	os_326	os_327	os_328	os_329	os_330	os_331	os_332	os_333	os_334	os_335	os_336	os_337	os_338	os_339	os_340	os_341	os_342	os_343	os_344	os_345	os_346	os_347	os_348	os_349	os_350	os_351	os_352	os_353	os_354	os_355	os_356	os_357	os_358	os_359	os_360	os_361	os_362	os_363	os_364	os_365	os_366	os_367	os_368	os_369	os_370	os_371	os_372	os_373	os_374	os_375	os_376	os_377	os_378	os_379	os_380	os_381	os_382	os_383	os_384	os_385	os_386	os_387	os_388	os_389	os_390	os_391	os_392	os_393	os_394	os_395	os_396	os_397	os_398	os_399	os_400	os_401	os_402	os_403	os_404	os_405	os_406	os_407	os_408	os_409	os_410	os_411	os_412	os_413	os_414	os_415	os_416	os_417	os_418	os_419	os_420	os_421	os_422	os_423	os_424	os_425	os_426	os_427	os_428	os_429	os_430	os_431	os_432	os_433	os_434	os_435	os_436	os_437	os_438	os_439	os_440	os_441	os_442	os_443	os_444	os_445	os_446	os_447	os_448	os_449	os_450	os_451	os_452	os_453	os_454	os_455	os_456	os_457	os_458	os_459	os_460	os_461	os_462	os_463	os_464	os_465	os_466	os_467	os_468	os_469	os_470	os_471	os_472	os_473	os_474	os_475	os_476	os_477	os_478	os_479	os_480	os_481	os_482	os_483	os_484	os_485	os_486	os_487	os_488	os_489	os_490	os_491	os_492	os_493	os_494	os_495	os_496	os_497	os_498	os_499	os_500	os_501	os_502	os_503	os_504	os_505	os_506	os_507	os_508	os_509	os_510	os_511	os_512	os_513	os_514	os_515	os_516	os_517	os_518	os_519	os_520	os_521	os_522	os_523	os_524	os_525	os_526	os_527	os_528	os_529	os_530	os_531	os_532	os_533	os_534	os_535	os_536	os_537	os_538	os_539	os_540	os_541	os_542	os_543	os_544	os_545	os_546	os_547	os_548	os_549	os_550	os_551	os_552	os_553	os_554	os_555	os_556	os_557	os_558	os_559	os_560	os_561	os_562	os_563	os_564	os_565	os_566	os_567	os_568	os_569	os_570	os_571	os_572	os_573	os_574	os_575	os_576	os_577	os_578	os_579	os_580	os_581	os_582	os_583	os_584	os_585	os_586	os_587	os_588	os_589	os_590	os_591	os_592	os_593	os_594	os_595	os_596	os_597	os_598	os_599	os_600	os_601	os_602	os_603	os_604	os_605	os_606	os_607	os_608	os_609	os_610	os_611	os_612	os_613	os_614	os_615	os_616	os_617	os_618	os_619	os_620	os_621	os_622	os_623	os_624	os_625	os_626	os_627	os_628	os_629	os_630	os_631	os_632	os_633	os_634	os_635	os_636	os_637	os_638	os_639	os_640	os_641	os_642	os_643	os_644	os_645	os_646	os_647	os_648	os_649	os_650	os_651	os_652	os_653	os_654	os_655	os_656	os_657	os_658	os_659	os_660	os_661	os_662	os_663	os_664	os_665	os_666	os_667	os_668	os_669	os_670	os_671	os_672	os_673	os_674	os_675	os_676	os_677	os_678	os_679	os_680	os_681	os_682	os_683	os_684	os_685	os_686	os_687	os_688	os_689	os_690	os_691	os_692	os_693	os_694	os_695	os_696	os_697	os_698	os_699	os_700	os_701	os_702	os_703	os_704	os_705	os_706	os_707	os_708	os_709	os_710	os_711	os_712	os_713	os_714	os_715	os_716	os_717	os_718	os_719	os_720	os_721	os_722	os_723	os_724	os_725	os_726	os_727	os_728	os_729	os_730	os_731	os_732	os_733	os_734	os_735	os_736	os_737	os_738	os_739	os_740	os_741	os_742	os_743	os_744	os_745	os_746	os_747	os_748	os_749	os_750	os_751	os_752	os_753	os_754	os_755	os_756	os_757	os_758	os_759	os_760	os_761	os_762	os_763	os_764	os_765	os_766	os_767	os_768	os_769	os_770	os_771	os_772	os_773	os_774	os_775	os_776	os_777	os_778	os_779	os_780	os_781	os_782	os_783	os_784	os_785	os_786	os_787	os_788	os_789	os_790	os_791	os_792	os_793	os_794	os_795	os_796	os_797	os_798	os_799	os_800	os_801	os_802	os_803	os_804	os_805	os_806	os_807	os_808	os_809	os_810	os_811	os_812	os_813	os_814	os_815	os_816	os_817	os_818	os_819	os_820	os_821	os_822	os_823	os_824	os_825	os_826	os_827	os_828	os_829	os_830	os_831	os_832	os_833	os_834	os_835	os_836	os_837	os_838	os_839	os_840	os_841	os_842	os_843	os_844	os_845	os_846	os_847	os_848	os_849	os_850	os_851	os_852	os_853	os_854	os_855	os_856	os_857	os_858	os_859	os_860	os_861	os_862	os_863	os_864	os_865	os_866	os_867	os_868	os_869	os_870	os_871	os_872	os_873	os_874	os_875	os_876	os_877	os_878	os_879	os_880	os_881	os_882	os_883	os_884	os_885	os_886	os_887	os_888	os_889	os_890	os_891	os_892	os_893	os_894	os_895	os_896	os_897	os_898	os_899	os_900	os_901	os_902	os_903	os_904	os_905	os_906	os_907	os_908	os_909	os_910	os_911	os_912	os_913	os_914	os_915	os_916	os_917	os_918	os_919	os_920	os_921	os_922	os_923	os_924	os_925	os_926	os_927	os_928	os_929	os_930	os_931	os_932	os_933	os_934	os_935	os_936	os_937	os_938	os_939	os_940	os_941	os_942	os_943	os_944	os_945	os_946	os_947	os_948	os_949	os_950	os_951	os_952	os_953	os_954	os_955	os_956	os_957	os_958	os_959	os_960	os_961	os_962	os_963	os_964	os_965	os_966	os_967	os_968	os_969	os_970	os_971	os_972	os_973	os_974	os_975	os_976	os_977	os_978	os_979	os_980	os_981	os_982	os_983	os_984	os_985	os_986	os_987	os_988	os_989	os_990	os_991	os_992	os_993	os_994	os_995	os_996	os_997	os_998	os_999	os_1000
--	-------------	-----------------	----	--------	-----	---------	--------	------------	-----------	-----------------------	------------	----------------	-----------------	-----------------	-----------------	------------	-------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------

Data Preprocessing

- Data Preparation for Modeling
 - Splitting the data between test and train data to evaluate the model built on the train data in **Figure 25**
 - Data was split in 70:30 ratio for train to test data

Figure 25

```
# 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 in specified ratio

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

- Overview of ML model and its parameters
- Summary of most important factors used by the ML model for prediction
- Summary of key performance metrics for training and test data in tabular format for comparison

[Link to Appendix slide on model assumptions](#)

Model Performance Summary

- Overview of ML model (final) and its parameters, **Figure 26**
 - The Adjusted R^2 is 0.838, which is an exceptionally good model
 - The Adjusted R^2 values generally range from 0 to 1, where a higher value suggests a better fit, considering certain conditions are met
 - The constant coefficient, also known as the Y-intercept, is 1.50
 - To further explain, if the coefficients for the predictor variables are 0, then the output would be the constant coefficient
 - The coefficients of the predictor variables (14 total) is highlighted in **Figure 26**
 - For example, if the coefficient for 'main_camera_mp' increases by 1 unit and all other coefficients are constant, then the y (output) would change by 0.02
 - All features have a p-value of less than 0.5, which is why they were kept in the model

Figure 26

D. 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:	Sun, 25 Sep 2022	Prob (F-statistic):	0.00
Time:	06:33:50	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.

[Link to Appendix slide on model assumptions](#)

Model Performance Summary

- Summary of most important factors used by the ML model for prediction, **Figure 26**
 - The equation for the ML model would be:
 - 'normalized_used_price' = 1.5 + 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_Karbons)-0.0374('brand_name_Samsung')-0.0670('brand_name_Sony')+0.0801('brand_name_Xiaomi')-0.1276('os_Others')-0.0900('os_iOS')+0.0502('4g_yes')-0.0673('5g_yes')
 - The above features are the most important factors used by the ML model for prediction

Figure 26

D. 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:	Sun, 25 Sep 2022	Prob (F-statistic):	0.00
Time:	06:33:50	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_Karbons	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.

[Link to Appendix slide on model assumptions](#)

Model Performance Summary

- Summary of key performance metrics for training and test data in tabular format in **Figure 27**
 - As a result, the adjusted R^2 (for the training data) is 0.84 (able to explain 84% of the variation in data), therefore the model is not underfitting
 - The train and test RMSE and MAE are low and comparable, so the model is not overfitting either
 - MAE suggests that the model can predict used phone prices within a mean error of 0.18 on the test data
 - MAPE (Mean Absolute Percentage Error) of 4.5 on the test data means that we can predict within 4.5% of the used phone prices
 - As mentioned previously, the model is neither underfitting nor overfitting since the adjusted R-squared values for Training and Test performance are within 5% of each other

Figure 27

Training Performance					
	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	0.229884	0.180326	0.844886	0.841675	4.326841

Test Performance					
	RMSE	MAE	R-squared	Adj. R-squared	MAPE
0	0.238358	0.184749	0.842479	0.834659	4.501651

[Link to Appendix slide on model assumptions](#)

APPENDIX

Data Background and Contents

- The data set consists of information about the attributes of used/refurbished phones and tablets
- There are 3454 rows of cell phones/tablets and 15 columns
- The columns consist of :

- 'brand_name'
- 'os'
- 'screen_size'
- '4G'
- '5G'
- 'main_camera_mp'
- 'selfie_camera_mp'
- 'int_memory'
- 'ram'
- 'battery'
- 'weight'
- 'release_year'
- 'days_used'
- 'normalized_used_price'
- 'normalized_new_price'

First 5 rows of the dataset

	brand_name	os	screen_size	4g	5g	main_camera_mp	selfie_camera_mp	int_memory	ram	battery	weight	release_year	days_used	normalized_used_price	normalized_new_price
0	Honor	Android	14.50	yes	no	13.0	5.0	64.0	3.0	3020.0	146.0	2020	127	4.307572	4.715100
1	Honor	Android	17.30	yes	yes	13.0	16.0	128.0	8.0	4300.0	213.0	2020	325	5.162097	5.519018
2	Honor	Android	16.69	yes	yes	13.0	8.0	128.0	8.0	4200.0	213.0	2020	162	5.111084	5.884631
3	Honor	Android	25.50	yes	yes	13.0	8.0	64.0	6.0	7250.0	480.0	2020	345	5.135387	5.630961
4	Honor	Android	15.32	yes	no	13.0	8.0	64.0	3.0	5000.0	185.0	2020	293	4.389995	4.947837

Data Background and Contents

- No duplicate values exist in the data
- Missing values do exist.
 - Under the **5g** column, there are 179 missing values
 - Under the **main_camera_mp**, there are 2 missing values
 - Under the **selfie_camera_mp**, there are 4 missing values
 - Under the **int_memory** column, there are 4 missing values
 - Under the **ram** column, there are 6 missing values
 - Under the **battery** column, there are 7 missing values
- In regards to the types of variables in the data set, there are 9 float types, 2 integer types, and 4 object types
- Statistical summary of the numerical values:
 - The range for the normalized used price (the target variable) is from 1.54 – 6.62 Euros
 - The mean and median for the normalized used price is respectively 4.36 and 4.41 Euros
 - The range for the normalized new price is 2.90 – 7.85 Euros
 - The mean and median for the normalized new price is respectively 5.23 and 5.25 Euros
 - The release year for the phones/tablets varies from 2013-2020
 - The phones/tablets weight varies from 69-855 grams
 - The battery varies from 900 – 9720 mAh

Data Background and Contents

- Statistical summary of the numerical values
 - The ram of the phones/tablets varies from 0.20 – 12 GB
 - The internal memory of the phones/tablets varies from 0.01 – 1024 GB
 - Resolution of the front camera of the phones/tablets varies from 0 – 32 MP
 - Resolution of the rear camera of the phones/tablets varies from 0.08 – 48 MP
 - The screen size of the phones/tablets varies from 5.08 – 30.71 CM

Model Assumptions

- Tests conducted for checking model assumptions and the results obtained

- Test for Multicollinearity**

- 48 features were utilized to test for multicollinearity
- The dummy variables were excluded
- 'Screen_size' and 'weight' were the two columns that had VIFs >5 according to **Figure 28**
 - If VIF is greater than 5 for a predictor variable, then that variable should be dropped
- To remove multicollinearity
 - each column that has VIF>5 is dropped one by one,
 - look at the adjusted R^2 and RMSE (in **Figure 29**) to discover the variable making the least change in adjusted R^2 after being dropped

Figure 28

● `checking_vif(x_train)` *## Complete the code to check VIF on train data*

	feature	vif
0	cpuval	227.74581
1	screen_size	7477390
2	main_camera_mp	0.289551
3	selfie_camera_mp	0.215473
4	int_memory	1.364152
5	ram	0.282352
6	battery	4.081780
7	weight	0.298719
8	days_used	0.860588
9	normalized_new_price	3.119430
10	years_since_release	4.895007
11	brand_name_Accu	0.455593
12	brand_name_Apple	10.557668
13	brand_name_Asu	0.330338
14	brand_name_BlackBerry	1.632378
15	brand_name_Canon	1.774721
16	brand_name_Corpus	1.488056
17	brand_name_Gionee	1.951272
18	brand_name_Google	1.521778
19	brand_name_HTC	0.410361
20	brand_name_Honor	0.368687
21	brand_name_Huawei	6.983852
22	brand_name_Infinix	1.283855
23	brand_name_Karbon	1.973702
24	brand_name_LG	0.668832
25	brand_name_Lava	1.711360
26	brand_name_Lenovo	4.558441
27	brand_name_Maui	0.179807
28	brand_name_Micromax	0.360351
29	brand_name_Microsoft	1.880751
30	brand_name_Motorola	0.274558

Figure 29

● `col_list = ['screen_size', 'weight']` *## Complete the code to specify the columns with high VIF*

`res = treating_multicollinearity(x_train, y_train, col_list)` *## Complete the code to check the effect*

`res`

	col	Adj. R-squared after dropping col	RMSE after dropping col
0	screen_size	0.680881	0.238703
1	weight	0.680071	0.234928

Model Assumptions

- Please mention the tests conducted for checking model assumptions and the results obtained
 - **Test for Multicollinearity**
 - To remove multicollinearity
 - 'Screen size' was dropped since its adjusted R^2 changed the least
 - After removing 'Screen size' and checking the VIF for the new training set, all predictor variables had $VIF < 5$:

, VIF after dropping screen_size

	feature	VIF
0	const	202.673906
1	main_camera_mp	2.281835
2	selfie_camera_mp	2.809009
3	int_memory	1.362043
4	ram	2.282350
5	battery	3.842989
6	weight	2.993855
7	days_used	2.648929
8	normalized_new_price	3.077650
9	years_since_release	4.730315

Model Assumptions

- Please mention the tests conducted for checking model assumptions and the results obtained
- Dropping p-values (not a part of the 5 assumptions for linear regression)**
 - Predictor variables that had a p-value > 0.05 were dropped as they did not significantly have an impact on the target variable
 - After dropping the p-values, the OLS model was run again and the results are to the right
 - As you can see, the p-values were all below 0.05 in the Regression results

Figure 30

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:	Sun, 25 Sep 2022	Prob (F-statistic):	0.00
Time:	06:30:04	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

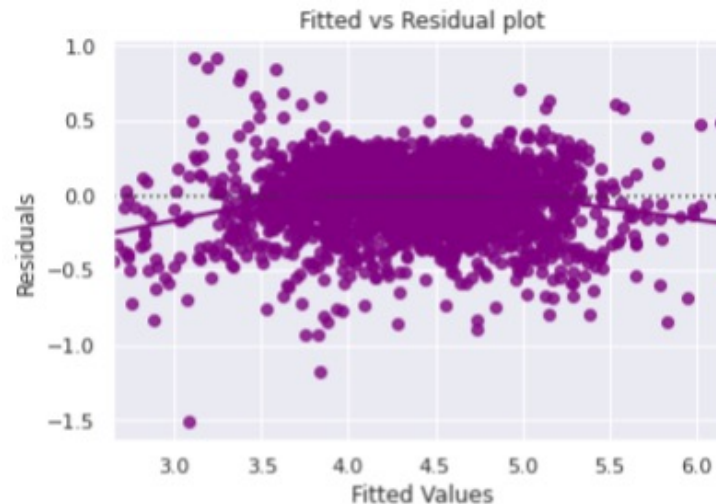
Notes:

- [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.

Model Assumptions

- Please mention the tests conducted for checking model assumptions and the results obtained
 - **Test for Linearity and Independence**
 - Since no pattern exists in **Figure 31** (Fitted Values vs Residuals graph), the model is linear and the residuals are independent
 - Otherwise, if a pattern was discovered, the model would be non-linear and residuals dependent

Figure 31



Model Assumptions

- Please mention the tests conducted for checking model assumptions and the results obtained

- Test for Normality**

- Based on the results from the distribution of residuals (**Figure 32**), the data resembles a normally distributed curve
- The Q-Q plot of residuals in **Figure 33** suggests that the residuals mostly follow a straight line except for the tails
- Based on the results from the Shapiro Wilk's test (**Figure 34**), the residuals are not normal since the p value is less than 0.05. In strict terms, the residual values are not normal but they are close to a normal distribution. Therefore, the assumption is satisfied.

Figure 32

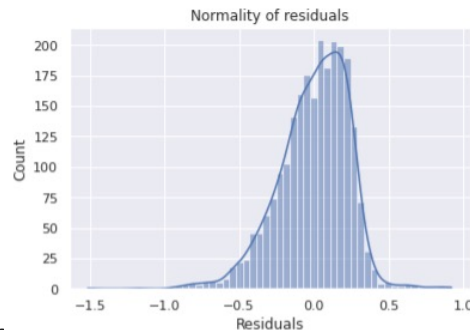


Figure 33

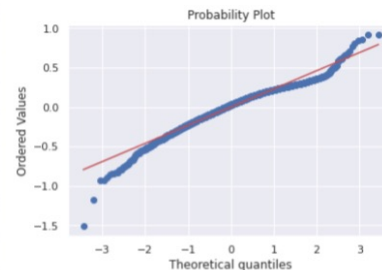


Figure 34

```
[ ] stats.shapiro(df_pred['Residuals']) #
ShapiroResult(statistic=0.9676972031593323, pvalue=6.995328206686811e-23)
```

Model Assumptions

- Please mention the tests conducted for checking model assumptions and the results obtained
 - **Test for homoscedasticity**
 - Goldfeldquandt test was used to test for homoscedasticity in **Figure 35**
 - Since the p-value is greater than 0.05, the residuals are homoscedastic

Figure 35

```
[ ] 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)
    lzip(name, test)

[('F statistic', 1.008750419910676), ('p-value', 0.4401970650667301)]
```



Happy Learning !

