PROJECT 02

TEAM MEMBERS

ROHIT PARIDA

VYSYARAJU VENUGOPALA RAJU

RIYA CHAUHAN

TABLE OF CONTENTS

ABSTRACT

Mobile phones are available in a wide range of costs, functionality, and other criteria. A key component of consumer strategy is the estimate and forecast of prices. For a product to be successful on the market, choosing the right pricing is crucial. In order for people to feel comfortable purchasing a new product, it must be priced appropriately.

We used Linear regression, Ridge regression, Automatic Relevance Determination(ARD), Stochastic gradient descent, *Bayesian Ridge Regression,* LassoLars , Least-angle regression (LARS), Elastic Net, Lasso, XGB regressor in this report. XGB regression model outperformed all the other models with a low RMSE error of 0.08 percent.

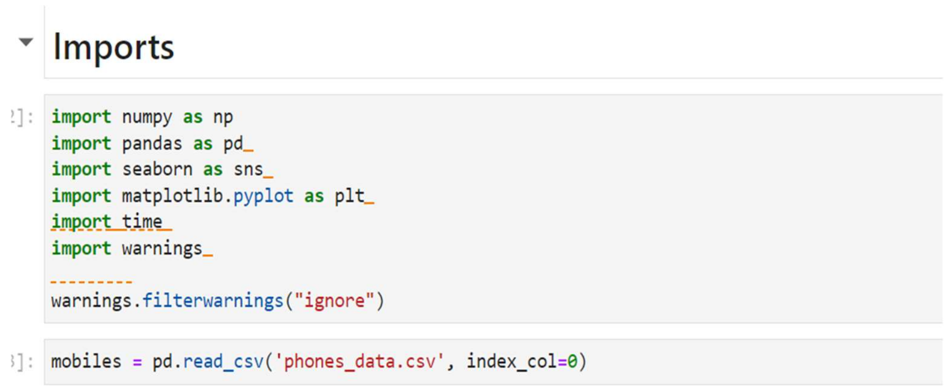
DATA

The data includes details on the specifications, attributes, and price range of mobile phones. The numerous features and data may be utilised to estimate a mobile phone's pricing range.

The data features are as follows:

1. Battery Power in mAh
2. Screen size
3. Release data
4. Brand
5. Operating System
6. Internal Memory in Gigabytes
7. Model name
8. Popularity
9. Lowest price
10. Highest price
11. Seller amount

Import Libraries

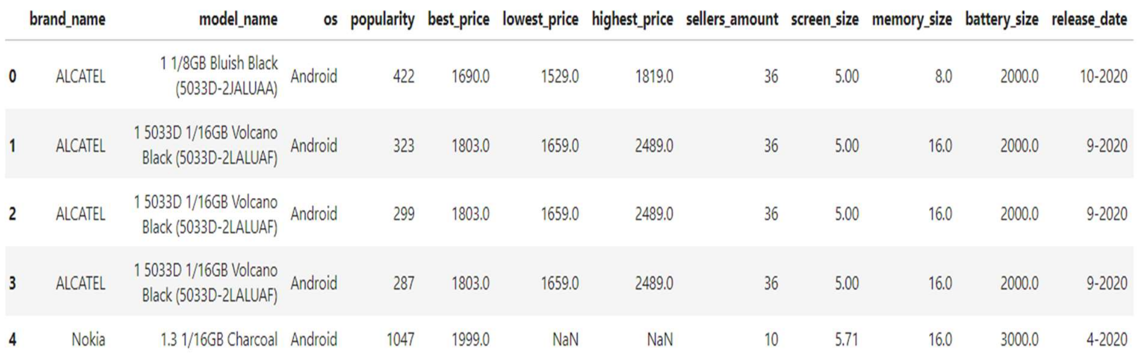


Imports

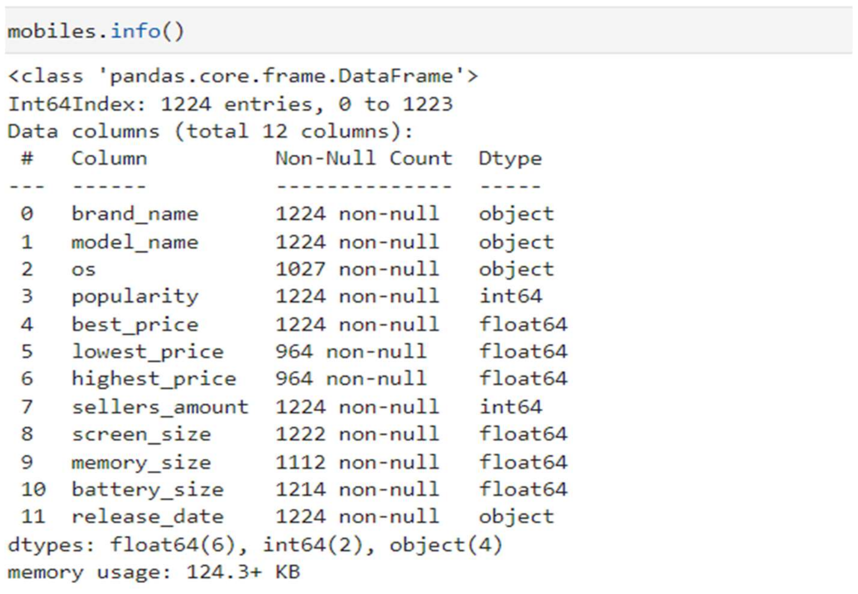
```
]: import numpy as np
   import pandas as pd
   import seaborn as sns
   import matplotlib.pyplot as plt
   import time
   import warnings

   warnings.filterwarnings("ignore")
```

```
]: mobiles = pd.read_csv('phones_data.csv', index_col=0)
```

OVERVIEW OF DATA



	brand_name	model_name	os	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size	battery_size	release_date
0	ALCATEL	1 1/8GB Bluish Black (5033D-2JALUAA)	Android	422	1690.0	1529.0	1819.0	36	5.00	8.0	2000.0	10-2020
1	ALCATEL	1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	Android	323	1803.0	1659.0	2489.0	36	5.00	16.0	2000.0	9-2020
2	ALCATEL	1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	Android	299	1803.0	1659.0	2489.0	36	5.00	16.0	2000.0	9-2020
3	ALCATEL	1 5033D 1/16GB Volcano Black (5033D-2LALUAF)	Android	287	1803.0	1659.0	2489.0	36	5.00	16.0	2000.0	9-2020
4	Nokia	1.3 1/16GB Charcoal	Android	1047	1999.0	NaN	NaN	10	5.71	16.0	3000.0	4-2020



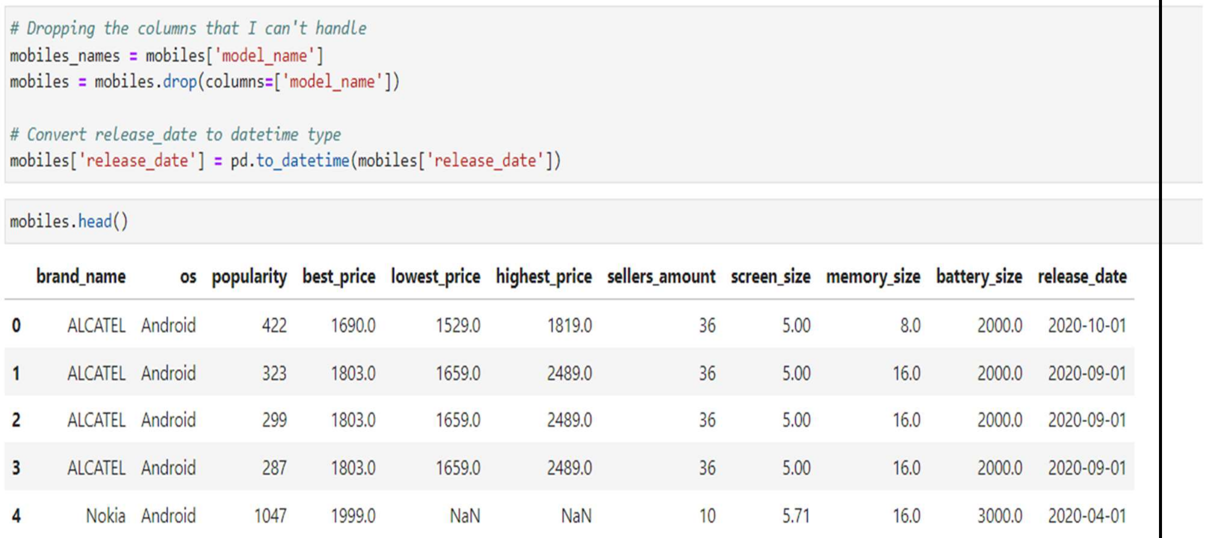
```
mobiles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1224 entries, 0 to 1223
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   brand_name      1224 non-null   object 
 1   model_name      1224 non-null   object 
 2   os              1027 non-null   object 
 3   popularity      1224 non-null   int64  
 4   best_price      1224 non-null   float64
 5   lowest_price    964 non-null    float64
 6   highest_price   964 non-null    float64
 7   sellers_amount  1224 non-null   int64  
 8   screen_size     1222 non-null   float64
 9   memory_size     1112 non-null   float64
 10  battery_size    1214 non-null   float64
 11  release_date    1224 non-null   object 
dtypes: float64(6), int64(2), object(4)
memory usage: 124.3+ KB
```

DATA CLEANING

The text data must be pre-processed before it can be mined, thus this is an important step. Projects involving data mining and machine learning are particularly susceptible to the adage "garbage in, garbage out." The techniques used to collect data are frequently not tightly regulated, which leads to out-of-range numbers, impossible data combinations, missing information, etc.

It includes: Removal of unwanted observations, Fixing Structural errors, managing outliers, Handling missing data.



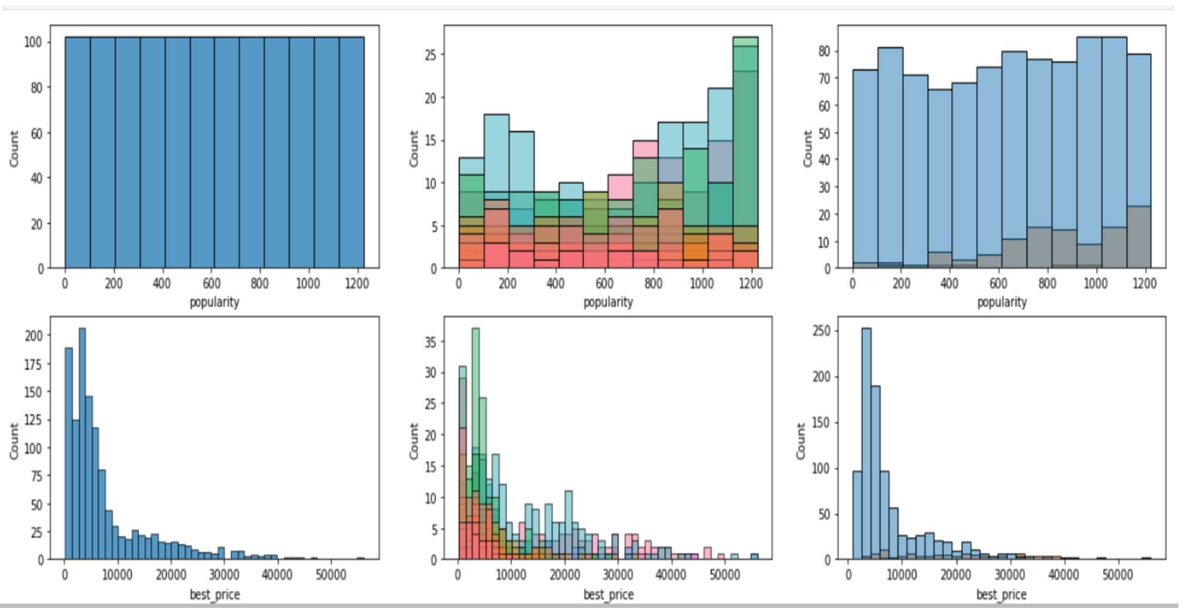
```
# Dropping the columns that I can't handle
mobiles_names = mobiles['model_name']
mobiles = mobiles.drop(columns=['model_name'])

# Convert release_date to datetime type
mobiles['release_date'] = pd.to_datetime(mobiles['release_date'])
```

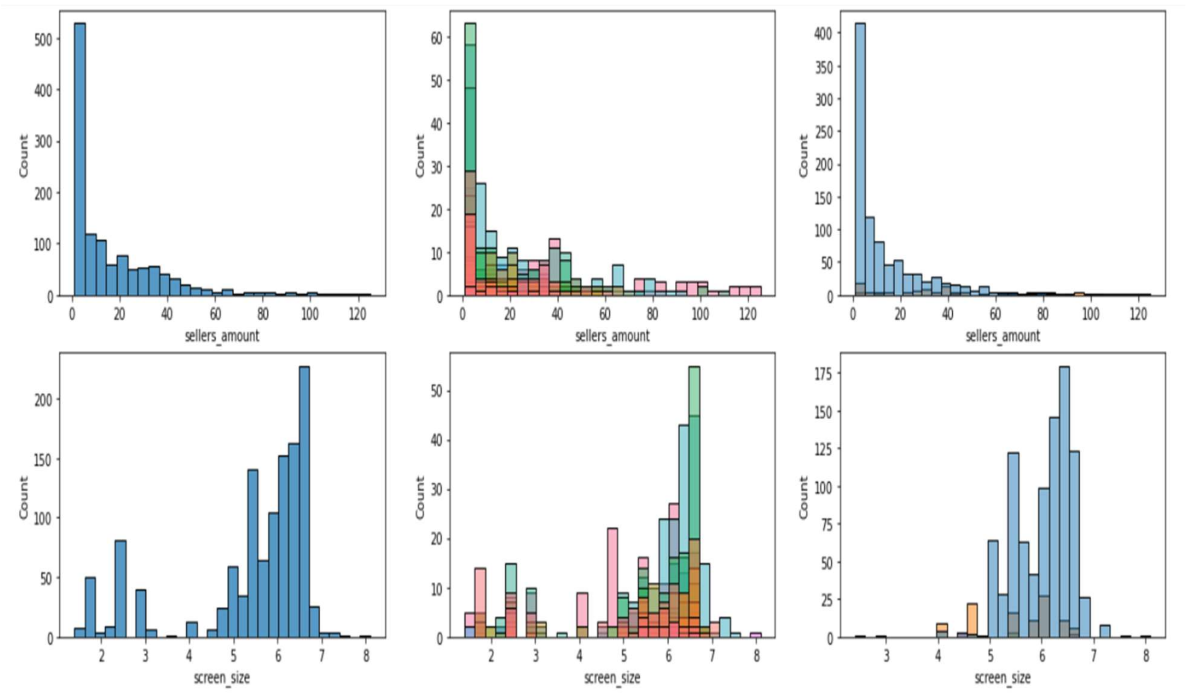
```
mobiles.head()
```

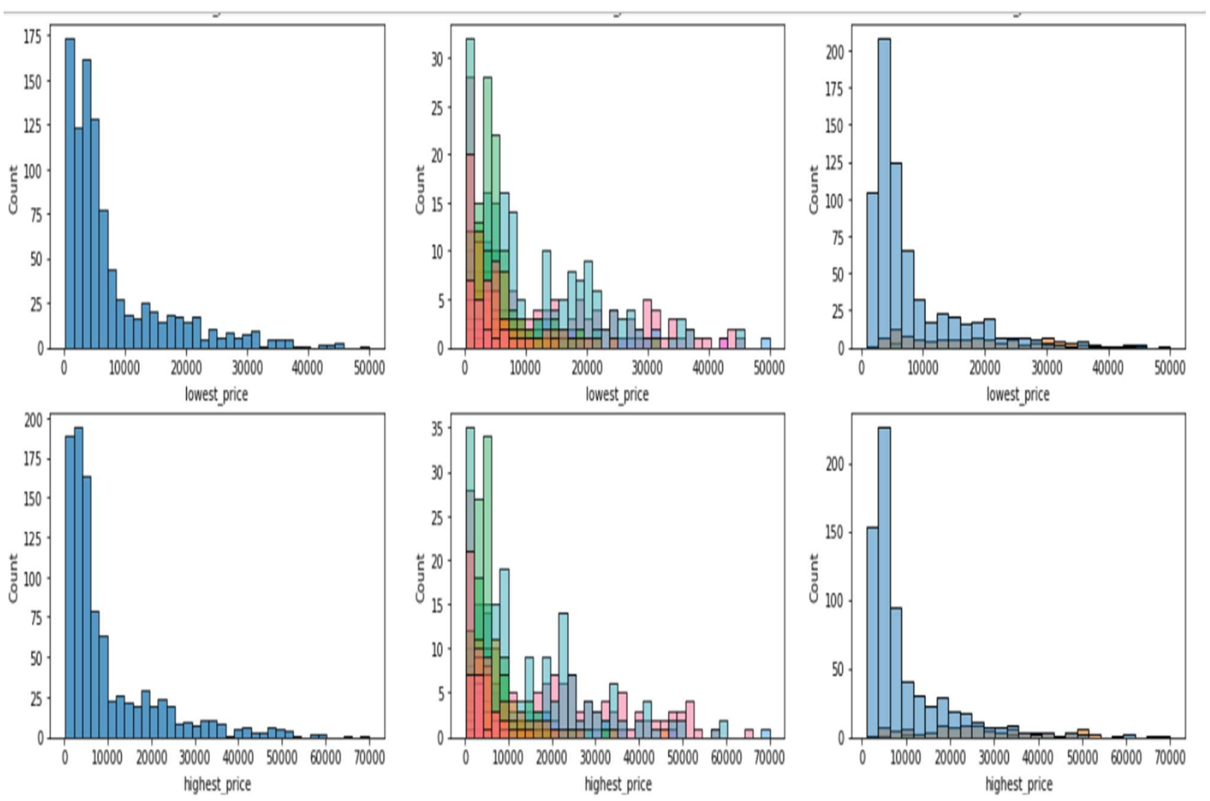
```
  brand_name       os  popularity  best_price  lowest_price  highest_price  sellers_amount  screen_size  memory_size  battery_size release_date
0    ALCATEL  Android         422      1690.0        1529.0         1819.0              36         5.00          8.0        2000.0   2020-10-01
1    ALCATEL  Android         323      1803.0        1659.0         2489.0              36         5.00         16.0        2000.0   2020-09-01
2    ALCATEL  Android         299      1803.0        1659.0         2489.0              36         5.00         16.0        2000.0   2020-09-01
3    ALCATEL  Android         287      1803.0        1659.0         2489.0              36         5.00         16.0        2000.0   2020-09-01
4      Nokia  Android        1047      1999.0           NaN            NaN              10         5.71         16.0        3000.0   2020-04-01
```

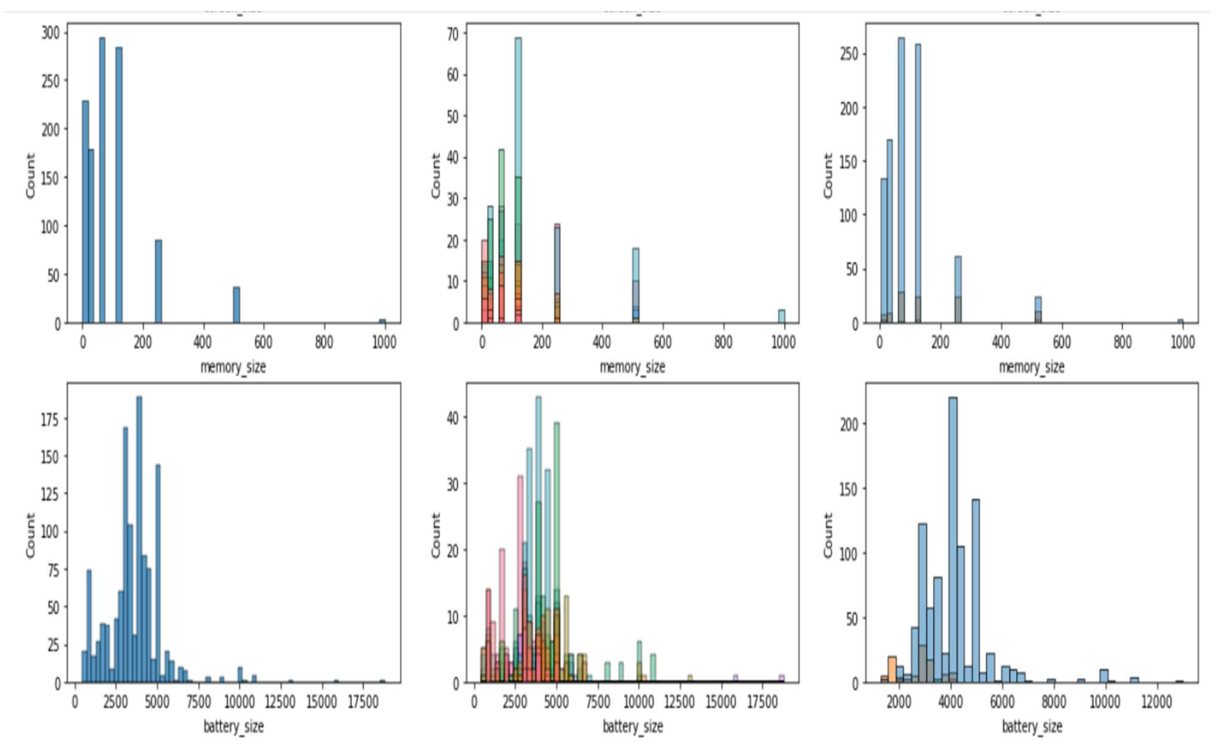
Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

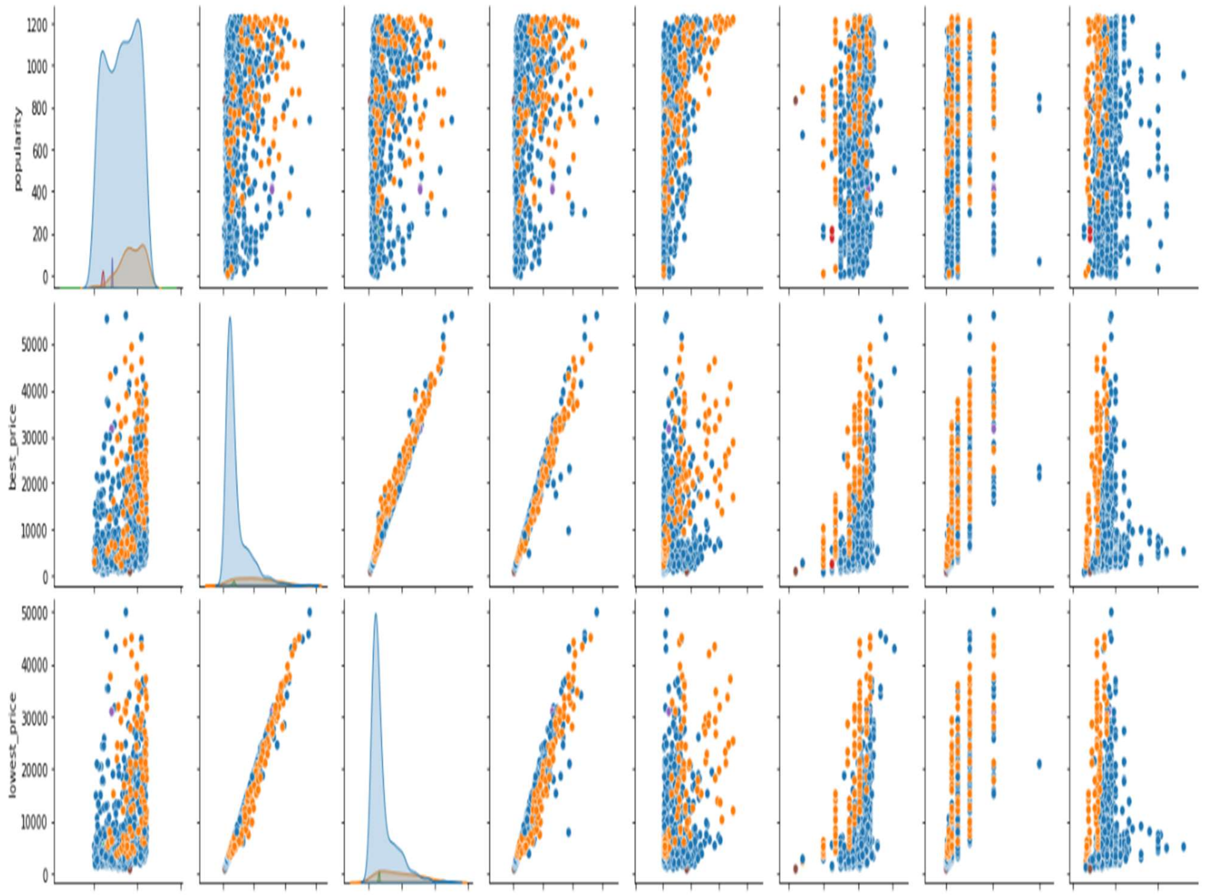

Count
popularity
Count
popularity
Count
popularity
Count
best_price
Count
best_price
Count
best_price

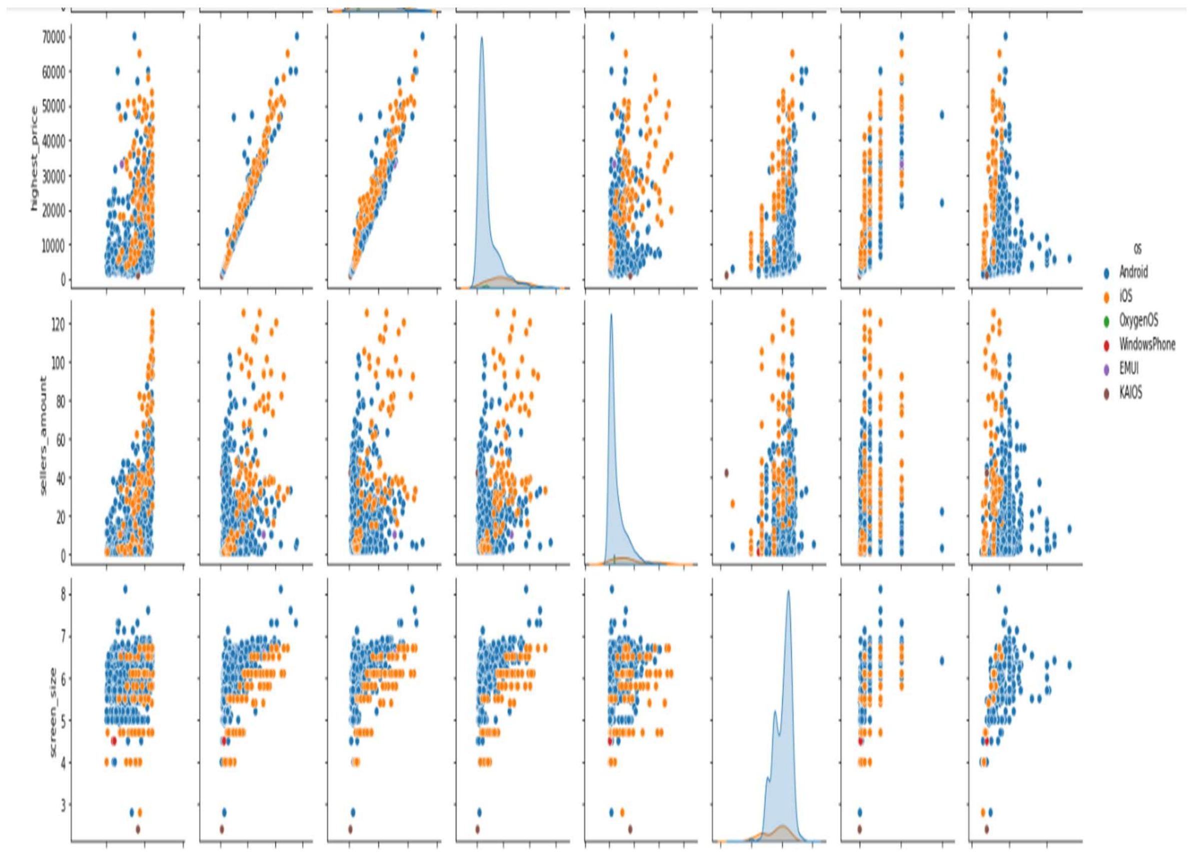

Count
sellers_amount
Count
sellers_amount
Count
sellers_amount
Count
screen_size
Count
screen_size
Count
screen_size

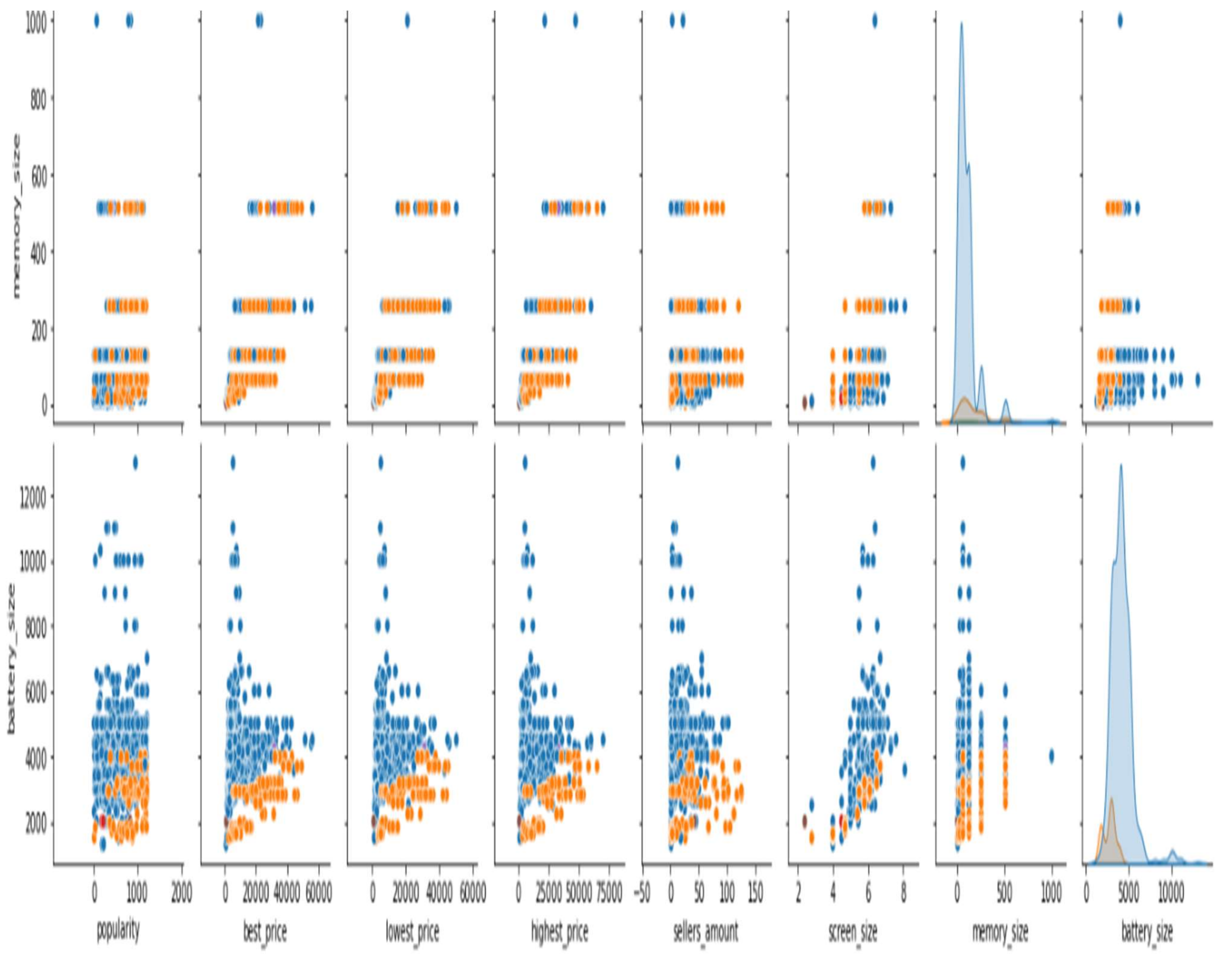

Count
lowest_price
Count
lowest_price
Count
lowest_price
Count
highest_price
Count
highest_price
Count
highest_price


Count
memory_size
Count
memory_size
Count
memory_size
Count
battery_size
Count
battery_size
Count
battery_size

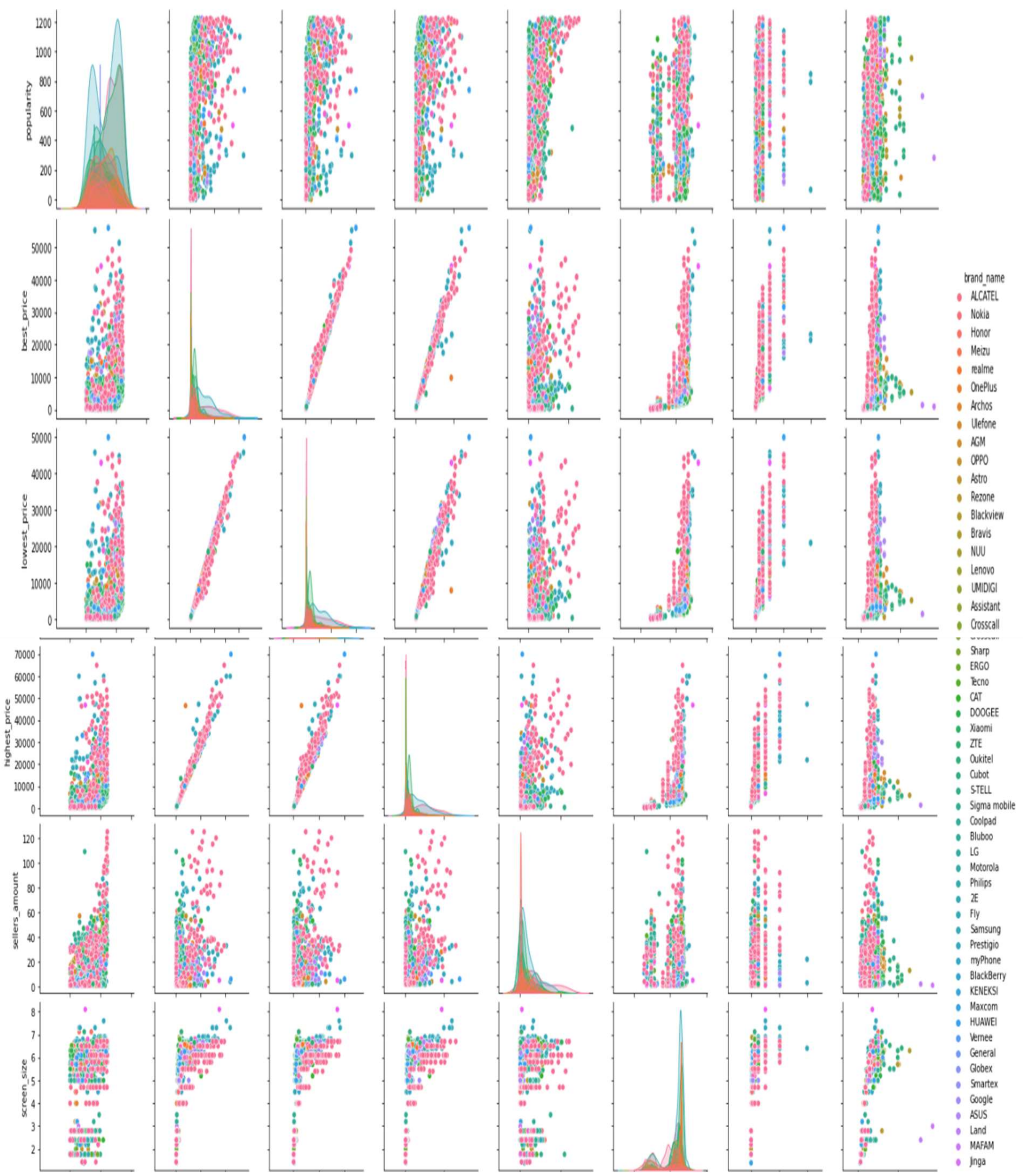
OPERATING SYSTEM

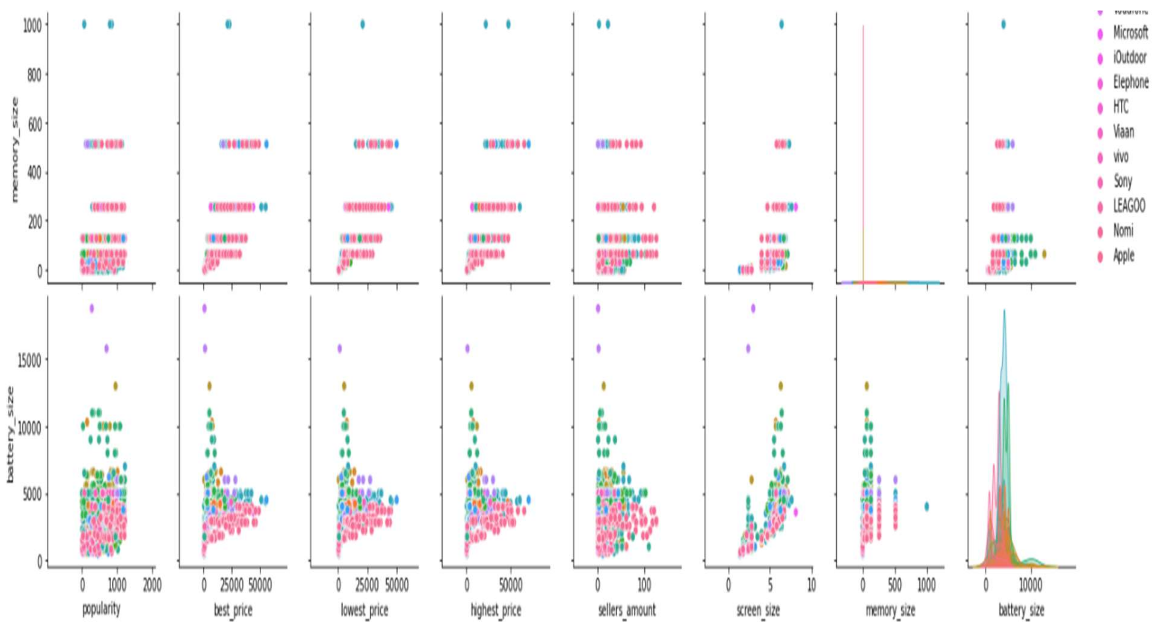


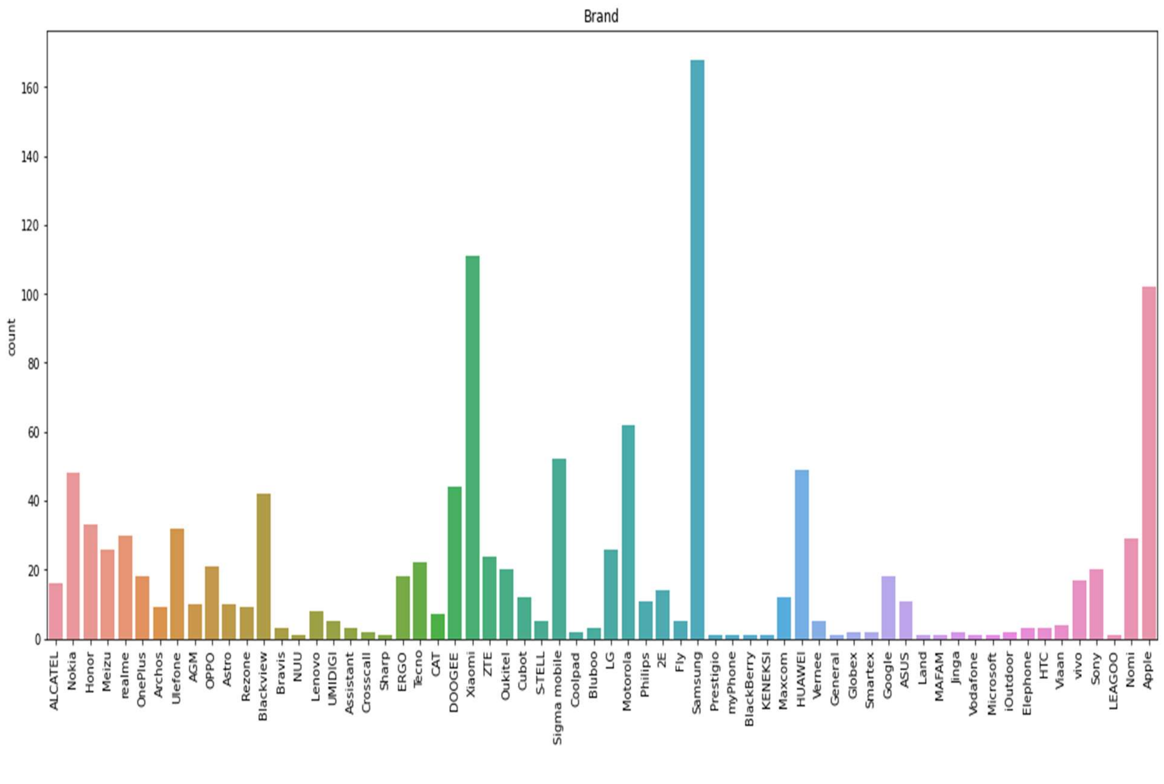

highest_price
sellers_amount
screen_size
os
Android
iOS
OxygenOS
WindowsPhone
EMUI
KAIOS

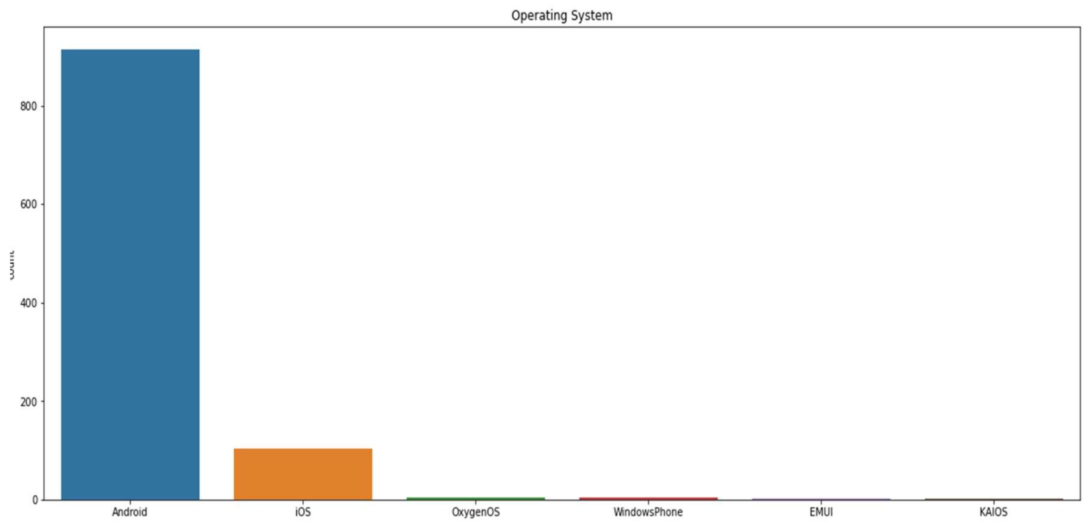

memory_size
battery_size
popularity
best_price
lowest_price
highest_price
sellers_amount
screen_size
memory_size
battery_size

BY BRAND

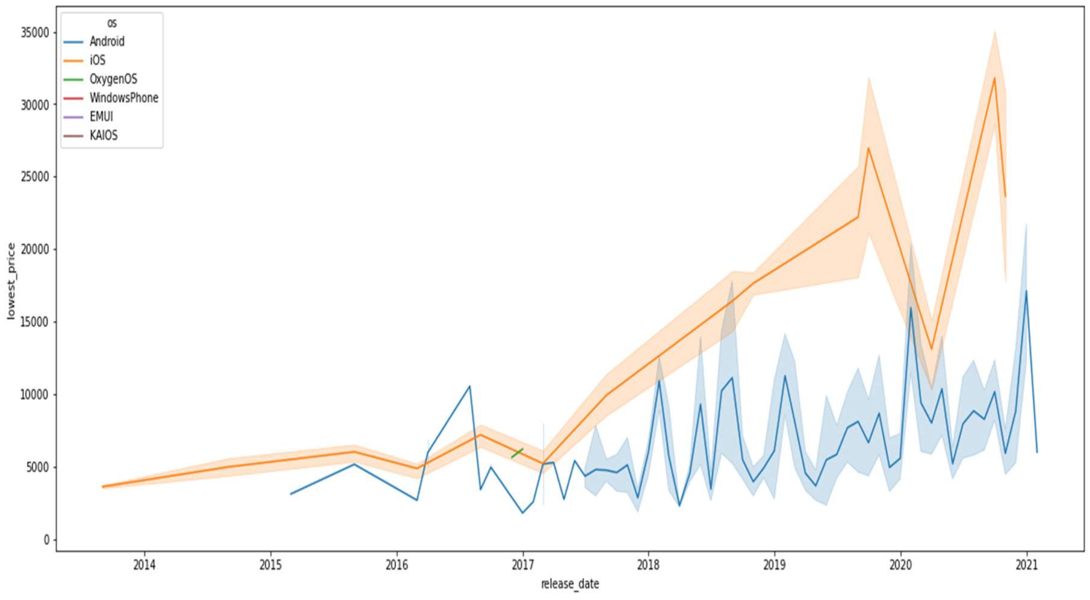

popularity
0
200
400
600
800
1000
1200
best_price
0
10000
20000
30000
40000
50000
lowest_price
0
10000
20000
30000
40000
50000
highest_price
0
10000
20000
30000
40000
50000
60000
70000
sellers_amount
0
20
40
60
80
100
120
screen_size
2
3
4
5
6
7
8
brand_name
ALCATEL
Nokia
Honor
Meizu
realme
OnePlus
Archos
Ulefone
AGM
OPPO
Astro
Rezone
Blackview
Bravis
NUU
Lenovo
UMIDIGI
Assistant
Crosscall
Sharp
ERGO
Tecno
CAT
DOOGEE
Xiaomi
ZTE
Oukitel
Cubot
S-TELL
Sigma mobile
Coolpad
Bluboo
LG
Motorola
Philips
2E
Fly
Samsung
Prestigio
myPhone
BlackBerry
KENEKSI
Maxcom
HUAWEI
Vernee
General
Globex
Smartex
Google
ASUS
Land
MAFAM
Jinga


memory_size
battery_size
popularity
best_price
lowest_price
highest_price
sellers_amount
screen_size
memory_size
battery_size
Microsoft
iOutdoor
Elephone
HTC
Viaan
vivo
Sony
LEAGOO
Nomi
Apple

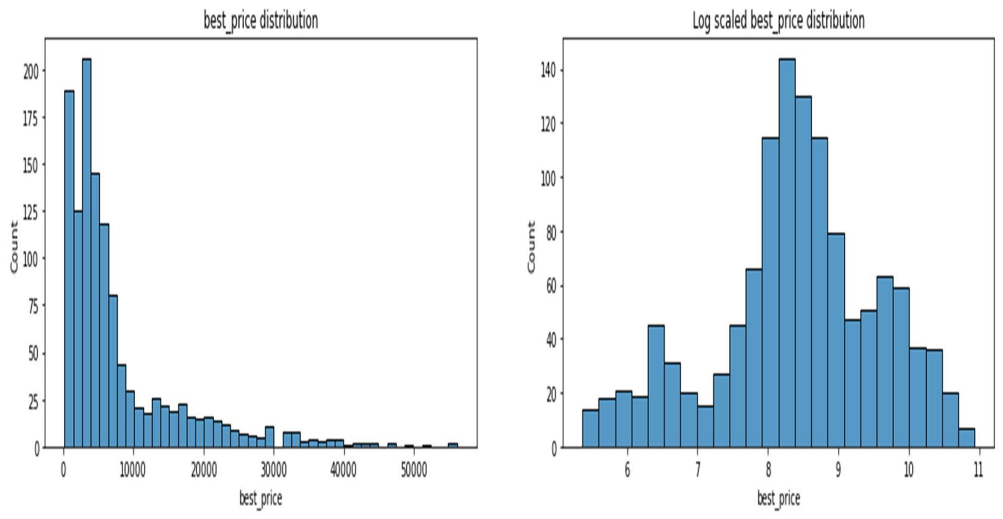

Brand
count
ALCATEL
Nokia
Honor
Meizu
realme
OnePlus
Archos
Ulefone
AGM
OPPO
Astro
Rezone
Blackview
Bravis
NUU
Lenovo
UMIDIGI
Assistant
Crosscall
Sharp
ERGO
Tecno
CAT
DOOGEE
Xiaomi
ZTE
Oukitel
Cubot
S-TELL
Sigma mobile
Coolpad
Bluboo
LG
Motorola
Philips
2E
Fly
Samsung
Prestigio
myPhone
BlackBerry
KENEKSI
Maxcom
HUAWEI
Vernee
General
Globex
Smartex
Google
ASUS
Land
MAFAM
Jinga
Vodafone
Microsoft
iOutdoor
Elephone
HTC
Viaan
vivo
Sony
LEAGOO
Nomi
Apple



PRICES EVOLUTION

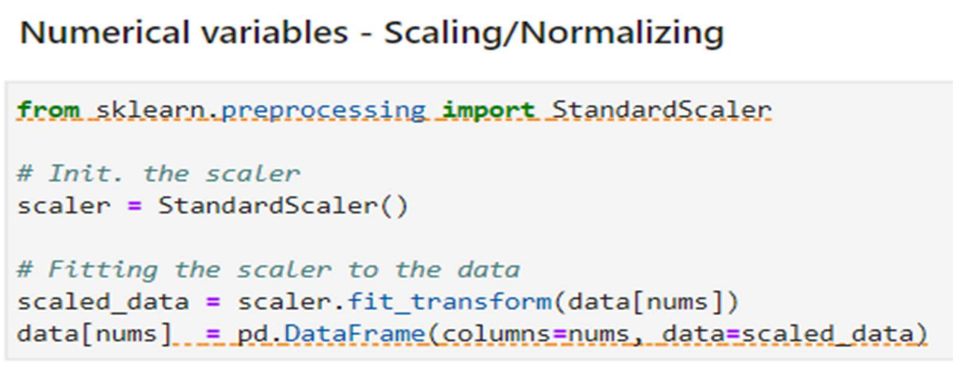


MODELLING



PRE-PROCESSING

NORMALIZING: Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

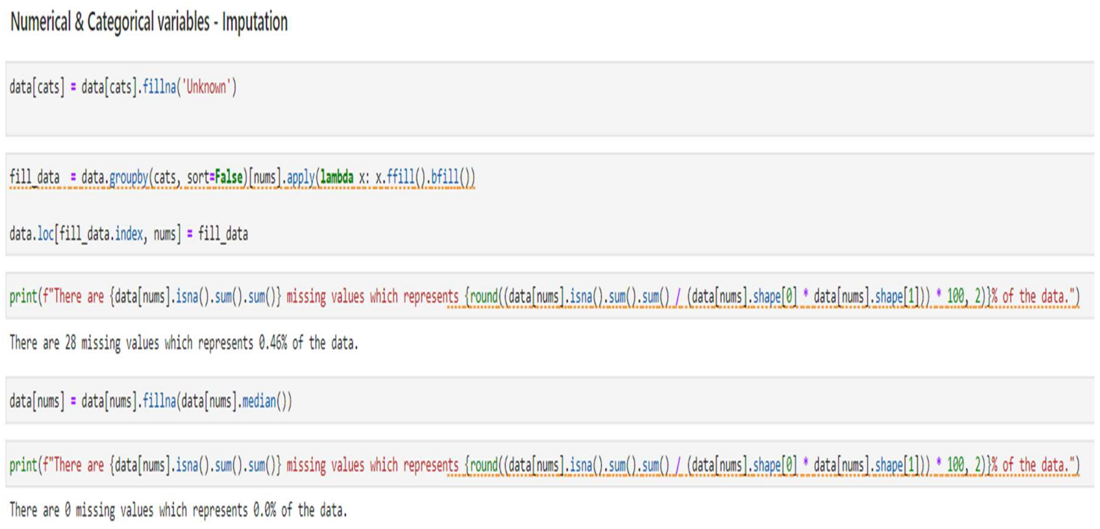


Numerical variables - Scaling/Normalizing

```python
from sklearn.preprocessing import StandardScaler

# Init. the scaler
scaler = StandardScaler()

# Fitting the scaler to the data
scaled_data = scaler.fit_transform(data[nums])
data[nums]  = pd.DataFrame(columns=nums, data=scaled_data)
```

IMPUTATION

Missing values may occur in datasets, which may be problematic for many machine learning techniques. Therefore, before modelling your prediction job, it is a good idea to find and replace any missing values for each column in your input data. Imputation of missing data, or imputing, is the term used to describe this. The process of calculating a statistical value for each column (such as a mean) and substituting that statistic for all missing values for that column is known as data imputation. It is a well-liked method as the statistic is simple to compute using the training dataset and it frequently yields high performance.



Numerical & Categorical variables - Imputation

```python
data[cats] = data[cats].fillna('Unknown')
```

```python
fill_data  = data.groupby(cats, sort=False)[nums].apply(lambda x: x.ffill().bfill())

data.loc[fill_data.index, nums] = fill_data
```

```python
print(f"There are {data[nums].isna().sum().sum()} missing values which represents {round((data[nums].isna().sum().sum() / (data[nums].shape[0] * data[nums].shape[1])) * 100, 2)}% of the data.")
```

```
There are 28 missing values which represents 0.46% of the data.
```

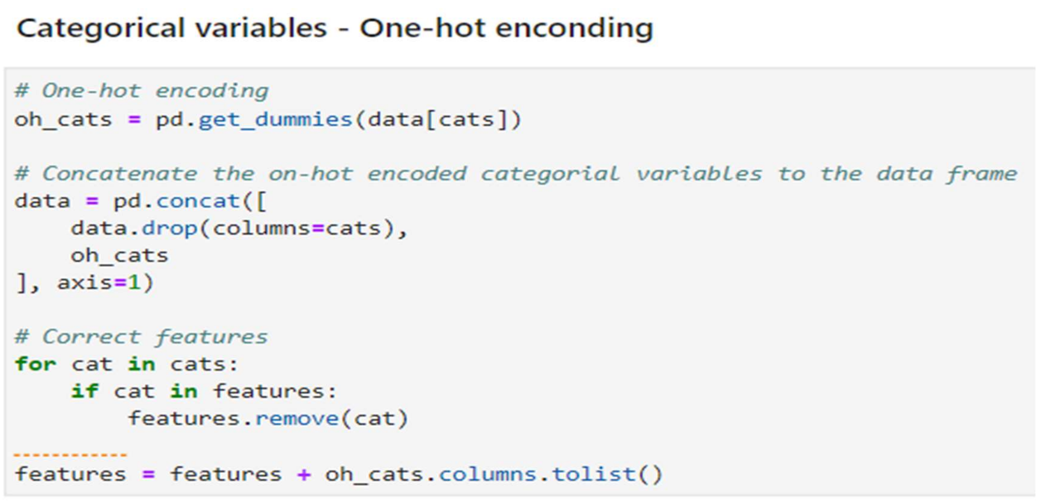
```python
data[nums] = data[nums].fillna(data[nums].median())
```

```python
print(f"There are {data[nums].isna().sum().sum()} missing values which represents {round((data[nums].isna().sum().sum() / (data[nums].shape[0] * data[nums].shape[1])) * 100, 2)}% of the data.")
```

```
There are 0 missing values which represents 0.0% of the data.
```

CATEGORICAL VARIABLES ONE HOT ENCODING

The performance of a machine learning model not only depends on the model and the hyperparameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numerical variables, pre-processing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model is able to understand and extract valuable information.

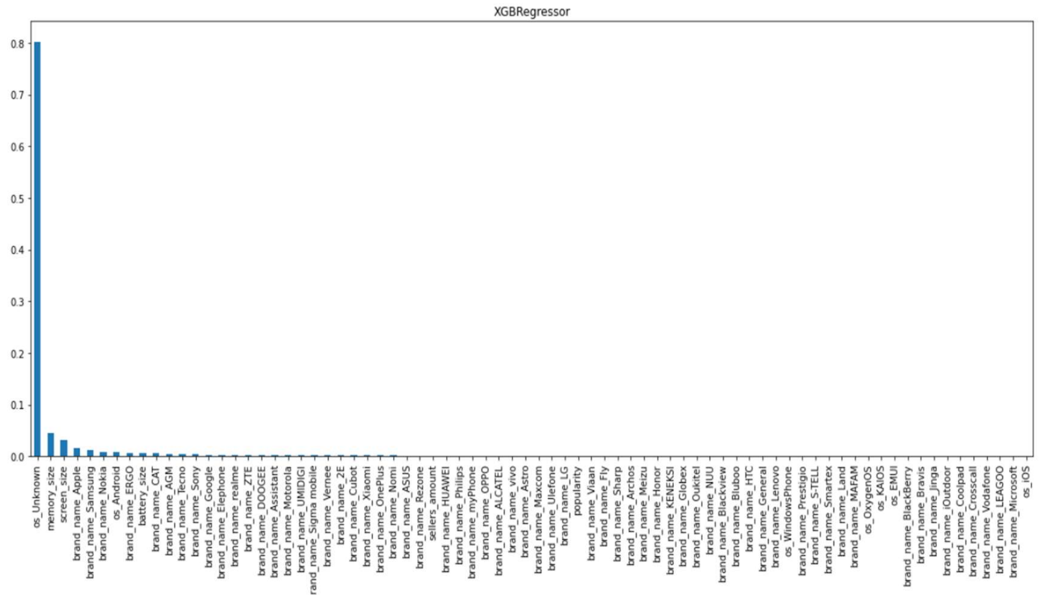


Categorical variables - One-hot enconding

```
# One-hot encoding
oh_cats = pd.get_dummies(data[cats])

# Concatenate the on-hot encoded categorial variables to the data frame
data = pd.concat([
    data.drop(columns=cats),
    oh_cats
], axis=1)

# Correct features
for cat in cats:
    if cat in features:
        features.remove(cat)

features = features + oh_cats.columns.tolist()
```

XGBOOST

Also known as extreme gradient boosting. The XGBoost classifier is a boosting classifier that combines hundreds of tree models with lower classification accuracy to produce a high accurate and low false positive model through iteration. XGBoost uses parallelized implementation to tackle the process of sequential tree construction. As a result, the loop order is switched to boost run time by using initialization through a global scan of all instances and sorting using parallel threads. It develops the tree to its maximum depth, then prunes it backwards until the loss function improvement falls below a certain threshold. The algorithm was created to make the most of available hardware resources. This is accomplished by assigning internal buffers in each thread to keep gradient statistics and thereby caching awareness. Additional improvements, such as "out-of-core" computation, allow for better utilisation of available disc space while processing large data frames that don't fit into memory. To minimise over-fitting, it penalises more complicated models using both LASSO (L1) and Ridge (L2) regularisation. By automatically "learning" the optimum missing value based on training loss, XGBoost naturally allows sparse features for input, and it handles different sorts of sparsity patterns in the data more effectively. At each iteration, the algorithm includes a built-in cross-validation procedure, eliminating the requirement to implement this search directly and specify the precise number of boosting rounds necessary in a single run. The heart of XGBoost is the gradient boosting (GBM) framework. Nonetheless, it outperforms the GBM framework on its own. It is used to solve supervised machine learning challenges.



Linear Regression

Linear regression is a quiet and simple statistical regression method used for predictive analysis and shows the relationship between the continuous variables. Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), consequently called linear regression. If there is a single input variable (x), such linear regression is called **simple linear regression**. And if there is more than one input variable, such linear regression is called **multiple linear regression.** The linear regression model gives a sloped straight line describing the relationship within the variables.

RIDGE REGRESSION

Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. Also known as **Tikhonov regularization**, named for Andrey Tikhonov, it is a method of regularization of ill-posed problems.[a] it is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters.[3] In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias.

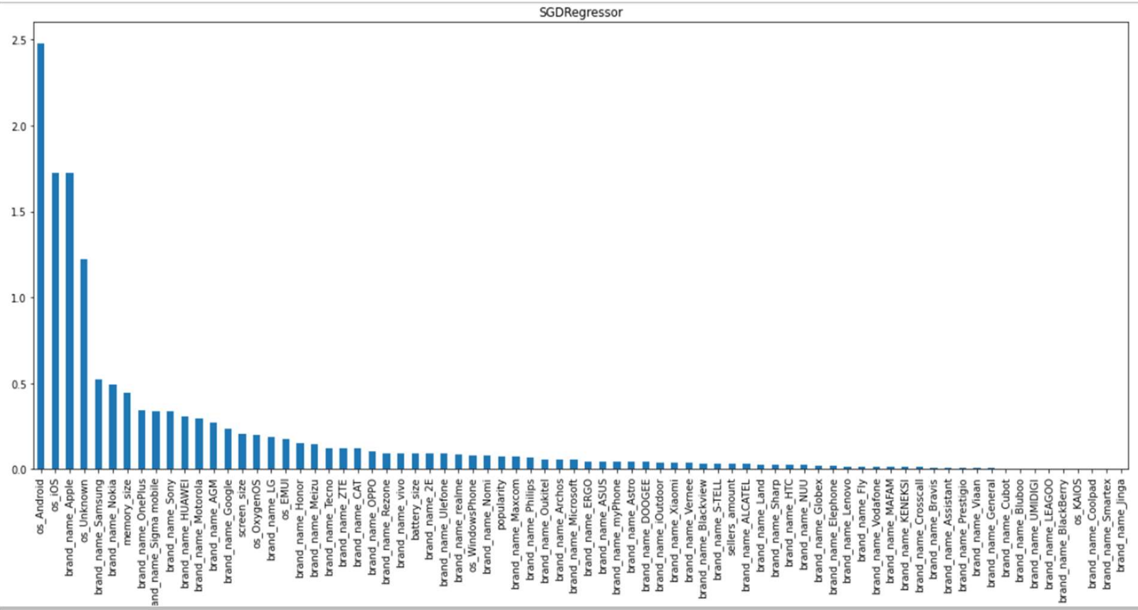
Ridge regression was developed as a possible solution to the imprecision of least square estimators when linear regression models have some multicollinear (highly correlated) independent variables—by creating a ridge regression estimator (RR). This provides a more precise ridge parameters estimate, as its variance and mean square estimator are often smaller than the least square estimators previously derived. Ridge Regressor:

- penalizes the size (square of the magnitude) of the regression coefficients
- enforces the *B* (slope/partial slope) coefficients to be lower, but not 0
- does not remove irrelevant features, but minimizes their impact

STOCHASTIC GRADIENT DESCENT

Applying the **Stochastic Gradient Descent (SGD)** method to the linear classifier or regressor provides the efficient estimator for classification and regression problems.

Scikit-learn API provides the SGD Regressor class to implement SGD method for regression problems. The SGD regressor applies regularized linear model with SGD learning to build an estimator. A regularizer is a penalty (L1, L2, or Elastic Net) added to the loss function to shrink the model parameters. The SGD regressor works well with large-scale datasets.



LASSO

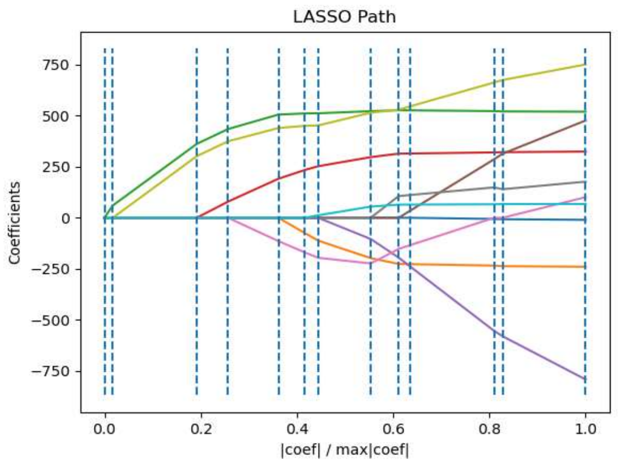
Lasso is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be (in general) reduced, and many will tend to be zeros. During training, the objective function become:

$$\frac{1}{2m}\sum_{i=1}^{m}(y-Xw)^2+alpha\sum_{j=1}^{p}\left|w_j\right|$$

Lasso introduces a new hyperparameter, *alpha*, the coefficient to penalize weights. Lasso tend to give sparse weights (most zeros), because the l1 regularization cares equally about driving down big weights to small weights, or driving small weights to zeros.

LARS LASSO

Lasso Lars is a lasso model implemented using the LARS algorithm, and unlike the implementation based on coordinate descent, this yields the exact solution, which is piecewise linear as a function of the norm of its coefficients.



ELASTIC NET

ElasticNet is a linear regression model trained with both ℓ1 and ℓ2-norm regularization of the coefficients. This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge. We control the convex combination of ℓ1 and ℓ2 using the l1_ratio parameter.

Elastic-net is useful when there are multiple features that are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.

A practical advantage of trading-off between Lasso and Ridge is that it allows Elastic-Net to inherit some of Ridge’s stability under rotation.

The objective function to minimize is in this case

Least Angle Regression

Least-angle regression (LARS) is a regression algorithm for high-dimensional data, developed by Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani. LARS is similar to forward stepwise regression. At each step, it finds the feature most correlated with the target. When there are multiple features having equal correlation, instead of continuing along the same feature, it proceeds in a direction equiangular between the features.

The advantages of LARS are:

- It is numerically efficient in contexts where the number of features is significantly greater than the number of samples.
- It is computationally just as fast as forward selection and has the same order of complexity as ordinary least squares.
- It produces a full piecewise linear solution path, which is useful in cross-validation or similar attempts to tune the model.
- If two features are almost equally correlated with the target, then their coefficients should increase at approximately the same rate. The algorithm thus behaves as intuition would expect, and also is more stable.
- It is easily modified to produce solutions for other estimators, like the Lasso.

AUTOMATIC RELEVANCE DETERMINATION REGRESSION

The Automatic Relevance Determination (as being implemented in ARDRegression) is a kind of linear model which is very similar to the Bayesian Ridge Regression, but that leads to sparser coefficients w.

ARD Regression poses a different prior over w: it drops the spherical Gaussian distribution for a centred elliptic Gaussian distribution. This means each coefficient w_i can itself be drawn from a Gaussian distribution, cantered on zero and with a precision λ_i:

$$p(w|\lambda) = N(w|0, A^{-1})$$

with A being a positive definite diagonal matrix and $\text{diag}(A)=\lambda = \{\lambda_1,\ldots\ldots,\lambda_p\}$.

In contrast to the Bayesian Ridge Regression, each coordinate of w_i has its own standard deviation $1/\lambda_i$. The prior over all λ_i is chosen to be the same gamma distribution given by the hyperparameters λ_1 and λ_2.

ARD is also known in the literature as Sparse Bayesian Learning and Relevance Vector Machine. For a worked-out comparison between ARD and Bayesian Ridge Regression, see the example below.

BAYESIAN RIDGE REGRESSION

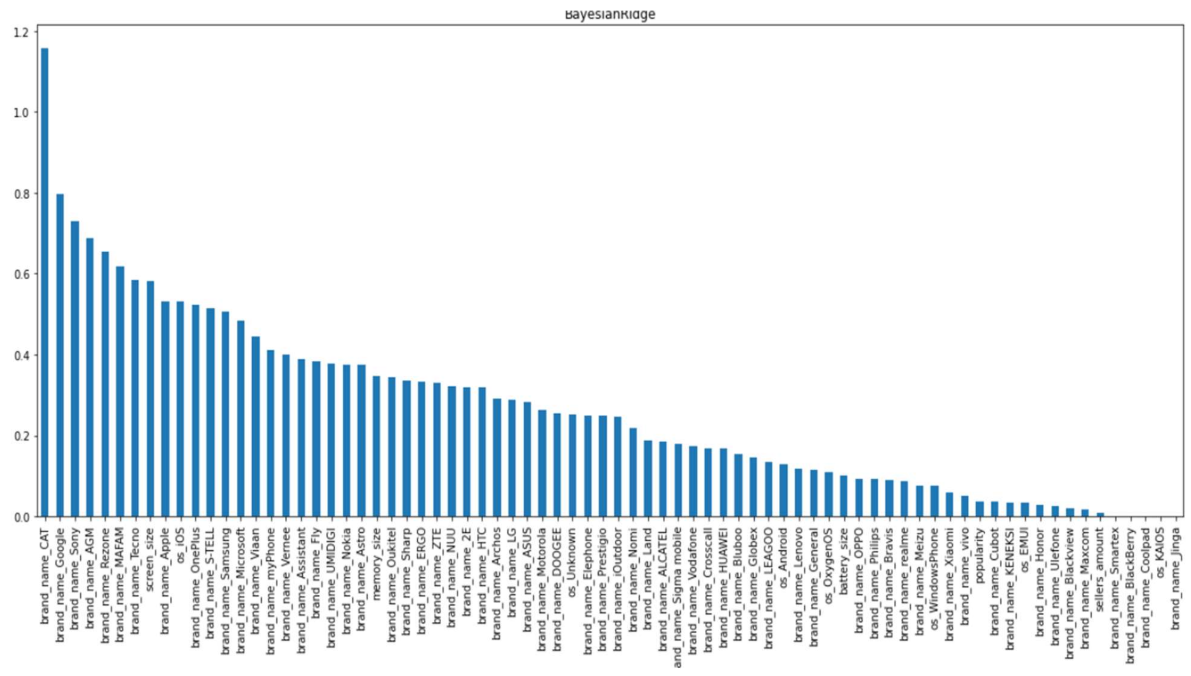
BayesianRidge estimates a probabilistic model of the regression problem as described above. The prior for the coefficient w is given by a spherical Gaussian:

$$p(w|\lambda)=N(w|0, \lambda^{-1} I_p)$$

The priors over α and λ are chosen to be gamma distributions, the conjugate prior for the precision of the Gaussian. The resulting model is called *Bayesian Ridge Regression*, and is similar to the classical Ridge.

The parameters w, α and λ are estimated jointly during the fit of the model, the regularization parameters α and λ being estimated by maximizing the *log marginal likelihood*. The scikit-learn implementation is based on the algorithm described in Appendix A of (Tipping, 2001) where the update of the parameters α and λ is done as suggested in (MacKay, 1992). The initial value of the maximization procedure can be set with the hyperparameters alpha_init and lambda_init.

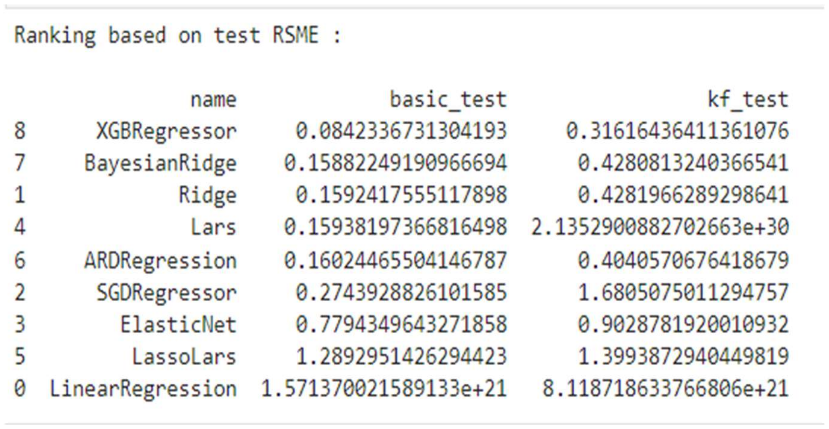
There are four more hyperparameters, α_1, α_2, λ_1 and λ_2 of the gamma prior distributions over α and λ. These are usually chosen to be *non-informative*. By default, $\alpha_1=\alpha_2=\lambda_1=\lambda_2=10^{-6}$.



Root Mean Square Error (RMSE)

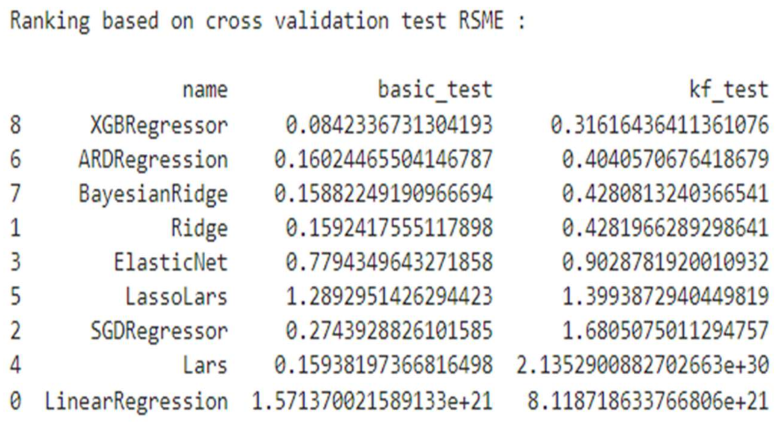
Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

RANKING BASED ON RMSE:



```
Ranking based on test RSME :

                name            basic_test               kf_test
8       XGBRegressor    0.0842336731304193    0.31616436411361076
7      BayesianRidge   0.15882249190966694     0.4280813240366541
1              Ridge    0.1592417555117898     0.4281966289298641
4               Lars   0.15938197366816498  2.1352900882702663e+30
6      ARDRegression   0.16024465504146787     0.4040570676418679
2       SGDRegressor    0.2743928826101585     1.6805075011294757
3         ElasticNet    0.7794349643271858     0.9028781920010932
5          LassoLars    1.2892951426294423     1.3993872940449819
0   LinearRegression  1.571370021589133e+21   8.118718633766806e+21
```

RANKING BASED ON CROSS VALIDATION TEST RSME:



```
Ranking based on cross validation test RSME :

               name           basic_test                kf_test
8      XGBRegressor   0.0842336731304193     0.31616436411361076
6     ARDRegression   0.16024465504146787     0.4040570676418679
7     BayesianRidge   0.15882249190966694     0.4280813240366541
1             Ridge   0.1592417555117898      0.4281966289298641
3        ElasticNet   0.7794349643271858      0.9028781920010932
5         LassoLars   1.2892951426294423      1.3993872940449819
2      SGDRegressor   0.2743928826101585      1.6805075011294757
4              Lars   0.15938197366816498  2.1352900882702663e+30
0  LinearRegression  1.571370021589133e+21   8.118718633766806e+21
```

FINANCIAL EQUATION

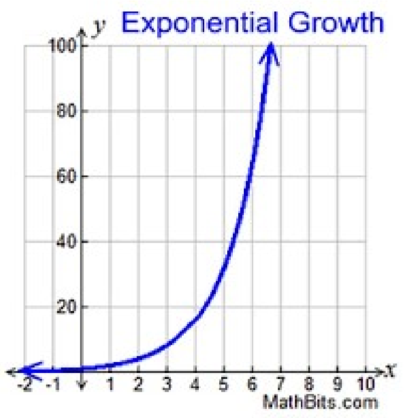
Variables as

Y= profit

X(t) = sales as function of time

M= price of products

C= maintenance cost.



Profit always be as revenue - expenses

Revenue is sales multiply by price of product

Hence y=m*x(t) -c

We reached to this equation as profits are always function of sales which most probably grows or steeps non linear hence we considered this change as function of time(t). This is included in sales. Hence we taken linear equation as stated above and calculated profit using OLS method.

When compared with time sales of mobile phones rose exponentially. Hence x(t) graph will be exponential. Maintenance cost like production cost, salary of employees, transportation, marketing are included which is not linear completely and the graph is random type.

…………………………………………ENDING…………………………………………