НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
з дисципліни «Математичне моделювання»
на тему

Моделювання системи для прогнозування популярності мобільних пристроїв

Виконала:
студентка групи КМ-83
Макаренко А.С.

Керівник:
Норкін Б.В.

Київ — 2021

Вступ

Постановка задачі

Предметною областю даного дослідження буде сфера продажу мобільних пристроїв. Вона є актуальною, адже майже в кожної людини є мобільний телефон. Основна задача - це **прогнозування популярності мобільного пристрою за його параметрами** (об'єм пам'яті, розмір екрану, операційною системою тощо). Результати дослідження будуть корисними для корпорацій-виробників мобільних телефонів, які бажають дізнатися чи буде користуватися попитом та чи інша модель телефону. На даному етапі основна ціль - підготувати датасет до можливості побудови математичної моделі на його основі. Це передбачає завантаження даних, їхню очистку, редагування.

В даній роботі було обрано задачу лассо-регресії. Було взято датасет "Mobile Phones Data". Задачею моделювання в рамках конкретного контексту є моделювання системи для передбачення популярності мобільного пристрою.

Огляд літератури

1. https://pythobyte.com/lasso-regression-73016c58/
2. The Elements of Statistical Learning Data Mining/ Inference/and Prediction/Trevor Hastie Robert Tibshirani Jerome Friedman
3. /python-data-science-machine-learning-tutorial/
4. https://uaspectr.com/2021/01/05/top-brendiv-smartfoniv-v-ukrayini-ta-sviti-2020/
5. http://www.machinelearning.ru/wiki/

Проектування математичного забезпечення

Перелік методів розв'язання задачі

З точки зору статистики та машинного навчання, задача прогнозування є задачею регресії. Для розв'язання задачі регресії необхідно змоделювати взаємоз'язок між змінними (залежними та незалежними). Система повинна прогнозувати популярність мобільного телефону, а отже, необхідно знайти зв'язок між його параметрами.

Для розв'язання задачі регресії розроблені алгоритми спеціальні алгоритми:

-Лінійна Регресія

-Лассо-регресія

-Логістична Регресія

-Поліноміальна Регресія

-Дерева Прийняття Рішень

Лінійна регресія будує логістичну криву, яка вираховує ймовірність виникнення певної події, саме тому вона більше підходить для задач класифікації. З плюсів можна виділити швидке моделювання, а серед мінусів - складність будування поліноміальних моделей.

Лассо-регресія - ще аналог лінійної регресії. Дозволяє позбутися від зайвих незалежних змінних, які не впливають на залежну змінну.

Логістична регресія використовується, коли залежна змінна є двійковою, тому цей метод більше підходить до задач класифікації, ніж до задачі регресії.

Поліноміальна регресія будує криву довільного порядку (квадратичну, кубічну і тд).

Дерево рішень - простий для розуміння алгоритм. Щоб змоделювати значення залежної змінної, перевіряються різні умови, які за своєю структурою

нагадують дерево. Кінцевим результатом є дерево із вузлами прийняття рішень (decision nodes) та листя (leafs). На ребрах («гілках») дерева рішення записані ознаки, від яких залежить цільова функція, а в «листі» записані значення цільової функції, а в інших вузлах - ознаки, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення.

Для вирішення даної задачі, було обрано метод лассо-регресії. Цей алгоритм легко інтерпретувати (тобто зрозуміти, що саме намагається зробити цей алгоритм). Було виявлено змінні, які майже не впливають на рейтинг телефонів, наприклад найнижча ціна.

Контрольний приклад

Для виконання цього завдання буде використано датасет, який був завантажений на веб-сайті Kaggle. Він містить дані про мобільні телефони, випущені за останні 4 роки та які можна купити в Україні.

Для подальшої роботи з датасетом, скористаємося бібліотекою pandas. Зчитати csv-файл можна за допомогою pandas.read_csv.

```
import pandas
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
import numpy as np
plt.style.use('ggplot')
from matplotlib.pyplot import figure
phones = 'drive/MyDrive/phones_data.csv'

phones = pandas.read_csv(phones)

phones
```

	Unnamed: 0	brand_name	model_name	os	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size	battery_size	release_date
0	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	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	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	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	4	Nokia	1.3 1/16GB Charcoal	Android	1047	1999.0	NaN	NaN	10	5.71	16.0	3000.0	4-2020
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1219	1219	Apple	iPhone XS Max 64GB Gold (MT522)	iOS	1101	22685.0	16018.0	27900.0	61	6.50	64.0	3174.0	9-2018
1220	1220	Apple	iPhone XS Max Dual Sim 64GB Gold (MT732)	iOS	530	24600.0	21939.0	33720.0	28	6.50	64.0	3174.0	9-2018
1221	1221	HUAWEI	nova 5T 6/128GB Black (51094MEU)	Android	1174	8804.0	7999.0	9999.0	18	6.26	128.0	3750.0	11-2019
1222	1222	ZTE	nubia Red Magic 5G 8/128GB Black	Android	752	18755.0	18500.0	19010.0	2	6.65	128.0	4500.0	10-2020
1223	1223	Sigma mobile	x-style 35 Screen	NaN	952	907.0	785.0	944.0	75	3.50	NaN	1750.0	1-2020

1224 rows × 13 columns

Візуалізація (елементарний опис зібраного набору даних)

Розглянемо детальніше вміст датасету, а саме його розмірність (кількість записів та кількість колонок) та характеристики записів.

```
phones.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1224 entries, 0 to 1223
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      1224 non-null   int64
 1   brand_name      1224 non-null   object
 2   model_name      1224 non-null   object
 3   os              1027 non-null   object
 4   popularity      1224 non-null   int64
 5   best_price      1224 non-null   float64
 6   lowest_price    964 non-null    float64
 7   highest_price   964 non-null    float64
 8   sellers_amount  1224 non-null   int64
 9   screen_size     1222 non-null   float64
 10  memory_size     1112 non-null   float64
 11  battery_size    1214 non-null   float64
 12  release_date    1224 non-null   object
dtypes: float64(6), int64(3), object(4)
memory usage: 124.4+ KB
```

З опису даного датасету на сторінці Kaggle можна дізнатись вміст усіх колонок датасету:

brand_name- назва бренду телефону;

model_name - назва моделі мобільного пристрою;

os - операційна система;

popularity - популярність;

best_price - найкраща ціна;

lowest_price - найнижча ціна;

highest_price - найвища ціна;

sellers_amount - кількість продавців;

screen_size - розмір екрану;

memory_size - розмір пам'яті пристрою;

battery_size - розмір батареї;

release_date - дата випуску.

Первинна очистка даних

Оскільки для нашої таблиці вже існує стовпчик з порядковим номером, можна видалити стовпець unnamed:

```
phones_new = phones.drop(columns="Unnamed: 0")
phones_new
```

	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
5	Honor	10 6/64GB Black	Android	71	10865.0	10631.0	11099.0	2	5.80	64.0	3400.0	6-2018
...	...	...	...	...	...	...	...	...	...	...	...	...
1219	Apple	iPhone XS Max 64GB Gold (MT522)	iOS	1101	22685.0	16018.0	27900.0	61	6.50	64.0	3174.0	9-2018
1220	Apple	iPhone XS Max Dual Sim 64GB Gold (MT732)	iOS	530	24600.0	21939.0	33720.0	28	6.50	64.0	3174.0	9-2018
1221	HUAWEI	nova 5T 6/128GB Black (51094MEU)	Android	1174	8804.0	7999.0	9999.0	18	6.26	128.0	3750.0	11-2019
1222	ZTE	nubia Red Magic 5G 8/128GB Black	Android	752	18755.0	18500.0	19010.0	2	6.65	128.0	4500.0	10-2020
1223	Sigma mobile	x-style 35 Screen	NaN	952	907.0	785.0	944.0	75	3.50	NaN	1750.0	1-2020

1224 rows × 12 columns

Наступним кроком буде усунення NaN, Null значень:

phones_new.isnull().sum()

```
brand_name          0
model_name          0
os                197
popularity          0
best_price          0
lowest_price      260
highest_price     260
sellers_amount      0
screen_size         2
memory_size       112
battery_size       10
release_date        0
dtype: int64
```

```
phones_new.dropna(subset=['os','lowest_price', 'highest_price',
'screen_size', 'memory_size', 'battery_size'], inplace=True)
phones_new
```

```
phones_new.isnull().sum()
```

```
brand_name        0
model_name        0
os                0
popularity        0
best_price        0
lowest_price      0
highest_price     0
sellers_amount    0
screen_size       0
memory_size       0
battery_size      0
release_date      0
dtype: int64
```

Як бачимо, NaN, Null значення в датасеті тепер відсутні. Переходимо до наступного кроку - видалення записів-дублікатів, якщо такі наявні. Перевіримо:

```
phones_new.duplicated().sum()
```

```
0
```

Отже, записи-дублікати в датасеті відсутні.

Розвідувальний аналіз даних (EDA)

Застосуємо описову статистику: знайдемо кількість non-NA/null значень, середні арифметичні, стандартні відхилення, мінімальне та максимальне значення кожної характеристики, скориставшись функцією describe():

```
phones_new.describe()
```

	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size	battery_size
count	780.000000	780.000000	780.000000	780.000000	780.000000	780.000000	780.000000	780.000000
mean	729.238462	10203.064103	9214.326923	11751.229487	20.226923	6.060462	112.287179	4121.117949
std	338.209052	9612.168455	8652.375633	11693.361223	22.330371	0.626583	117.754426	1406.849509
min	2.000000	1036.000000	899.000000	1059.000000	2.000000	2.400000	4.000000	1500.000000
25%	475.500000	3759.000000	3499.000000	3999.000000	4.000000	5.700000	32.000000	3174.000000
50%	783.500000	5931.000000	5547.000000	6512.500000	11.000000	6.215000	64.000000	4000.000000
75%	1017.250000	13904.250000	12425.500000	16057.750000	29.000000	6.500000	128.000000	5000.000000
max	1224.000000	55338.000000	45799.000000	64999.000000	125.000000	8.100000	1000.000000	13000.000000

Додатково знайдемо дисперсію кожної характеристики. Для цього скористаємося вбудованою в бібілотеку pandas функцією var().

phones_new.var()

```
popularity        1.143854e+05
best_price        9.239378e+07
lowest_price      7.486360e+07
highest_price     1.367347e+08
sellers_amount    4.986455e+02
screen_size       3.926059e-01
memory_size       1.386610e+04
battery_size      1.979226e+06
dtype: float64
```

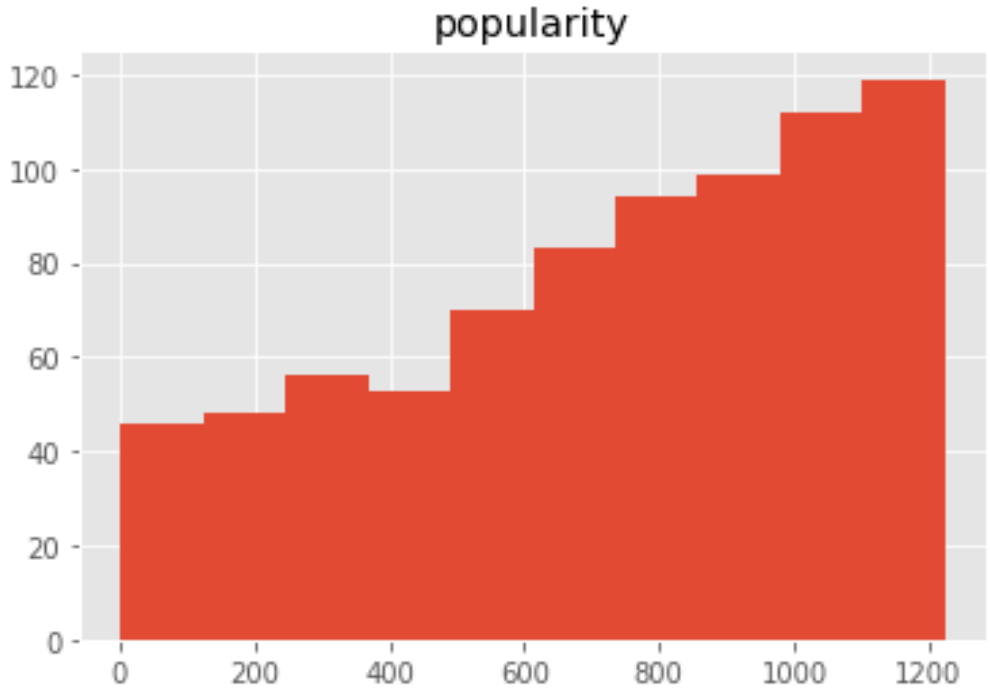
Знайдемо медіану, скориставшись вбудованою в бібілотеку pandas функцією median().

phones_new.median()

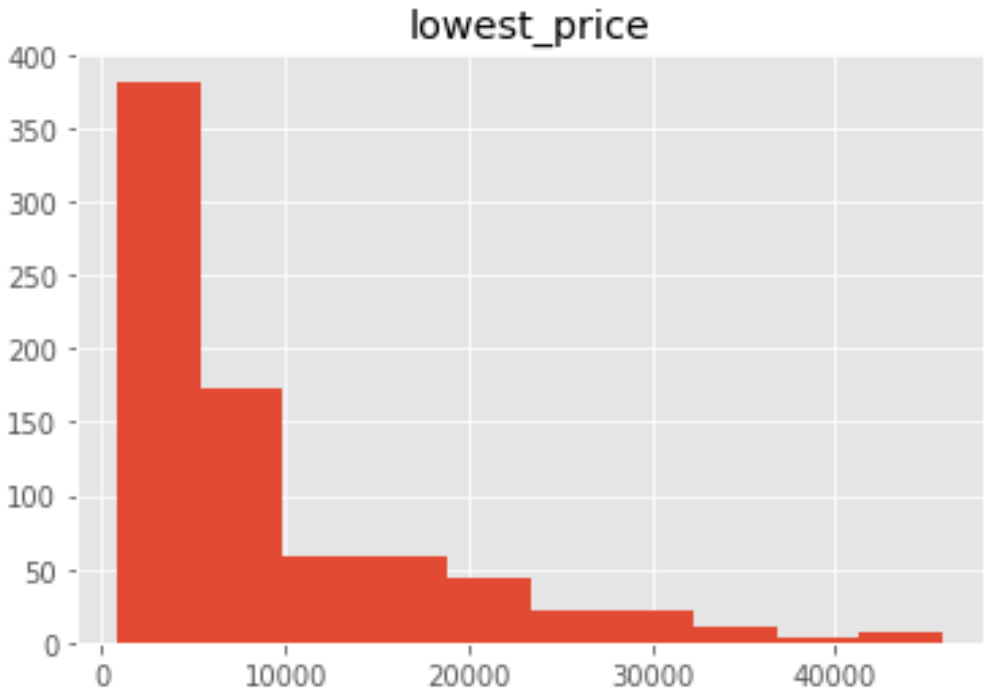
```
popularity          783.500
best_price         5931.000
lowest_price       5547.000
highest_price      6512.500
sellers_amount       11.000
screen_size           6.215
memory_size          64.000
battery_size       4000.000
dtype: float64
```

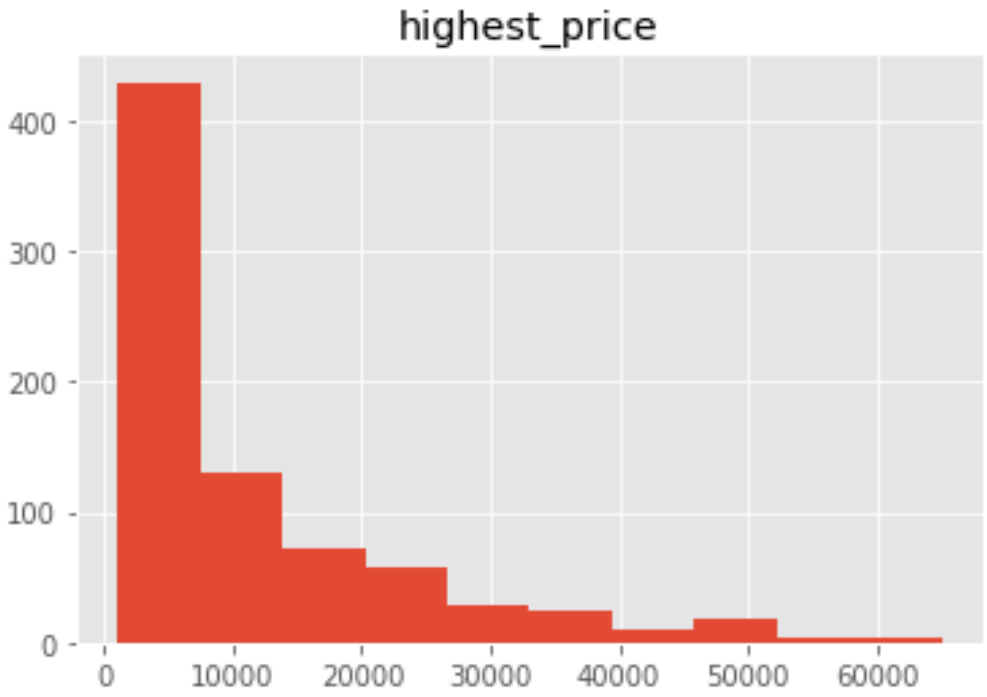
Побудуємо гістограми розподілів числових ознак датасету. Для побудови графіків скористаємося біліотекою matplotlib:

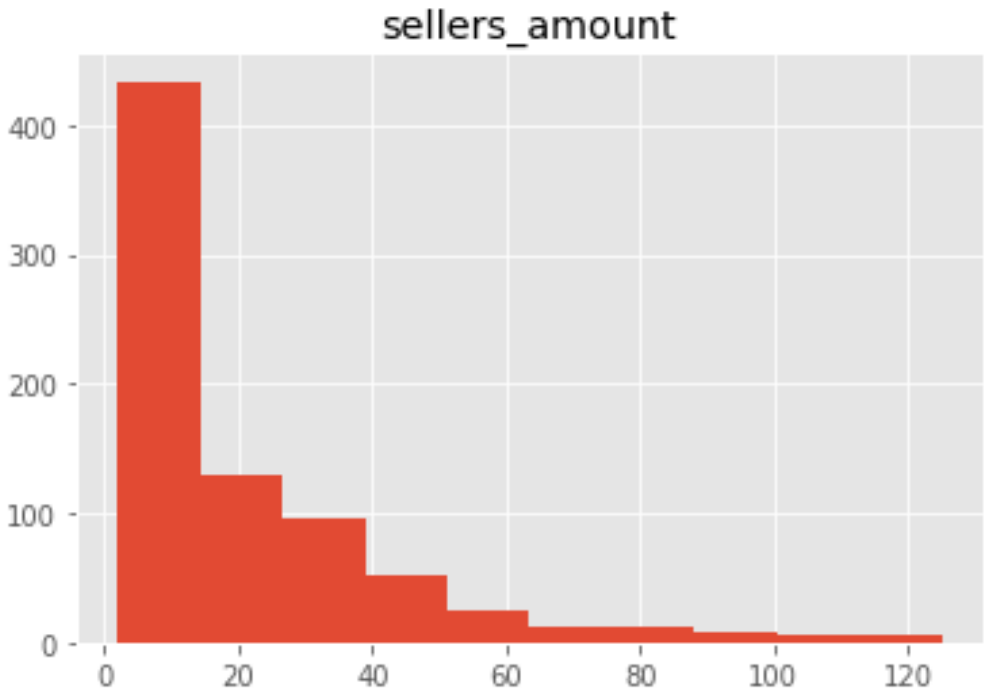
```
phones_new_numeric = phones_new.select_dtypes(include=[np.number])
numeric = phones_new_numeric.columns.values
for i in numeric:
    plt.hist(phones_new[i])
    plt.title(i)
    plt.show()
```

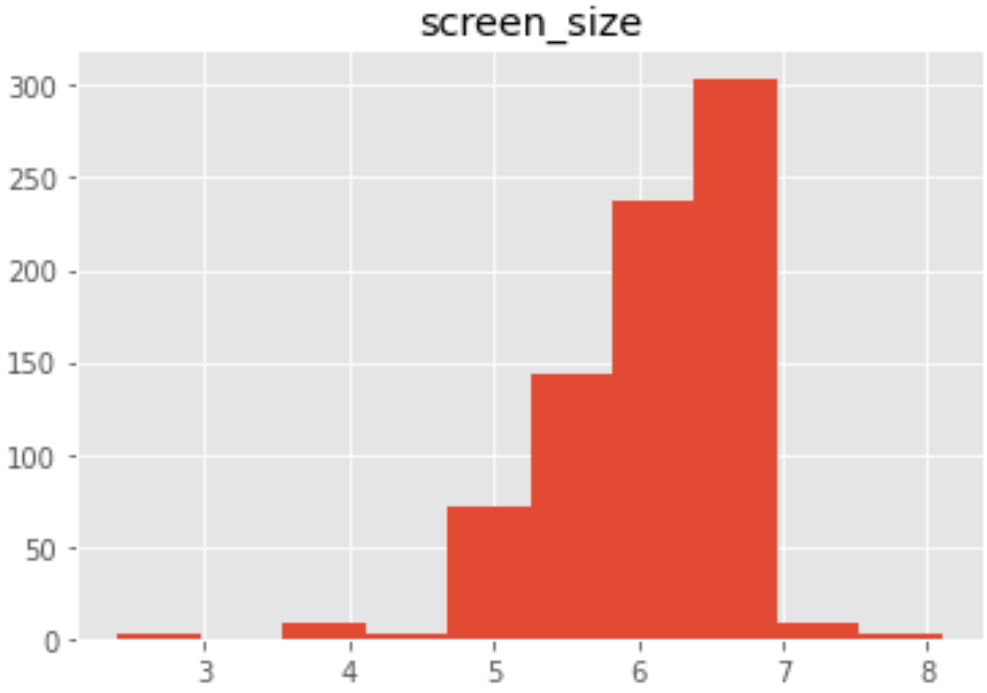


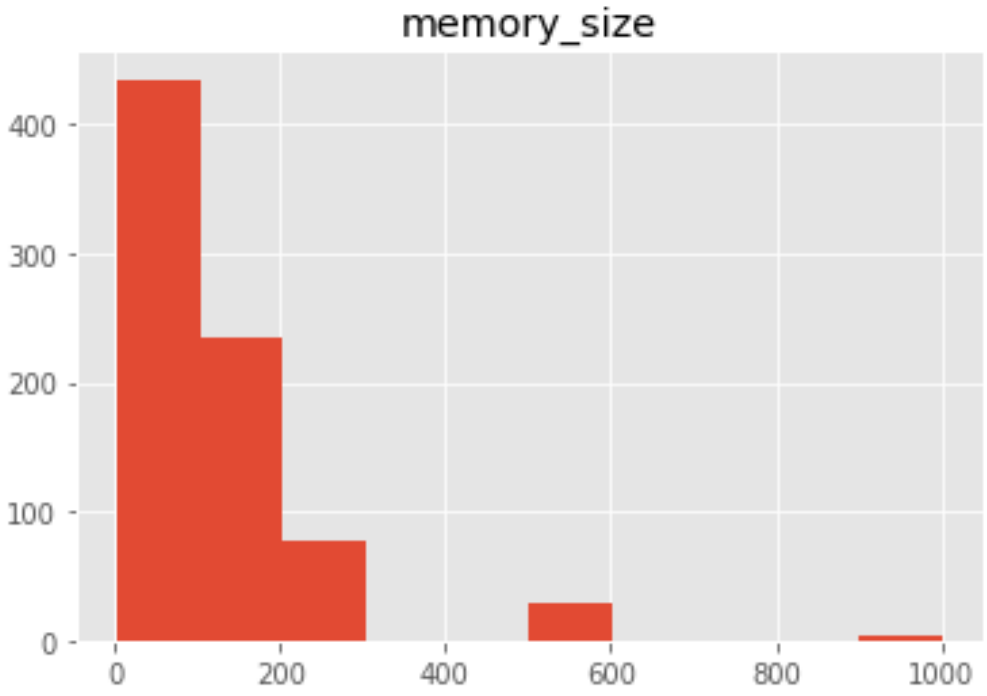

best_price
400
350
300
250
200
150
100
50
0
0
10000
20000
30000
40000
50000

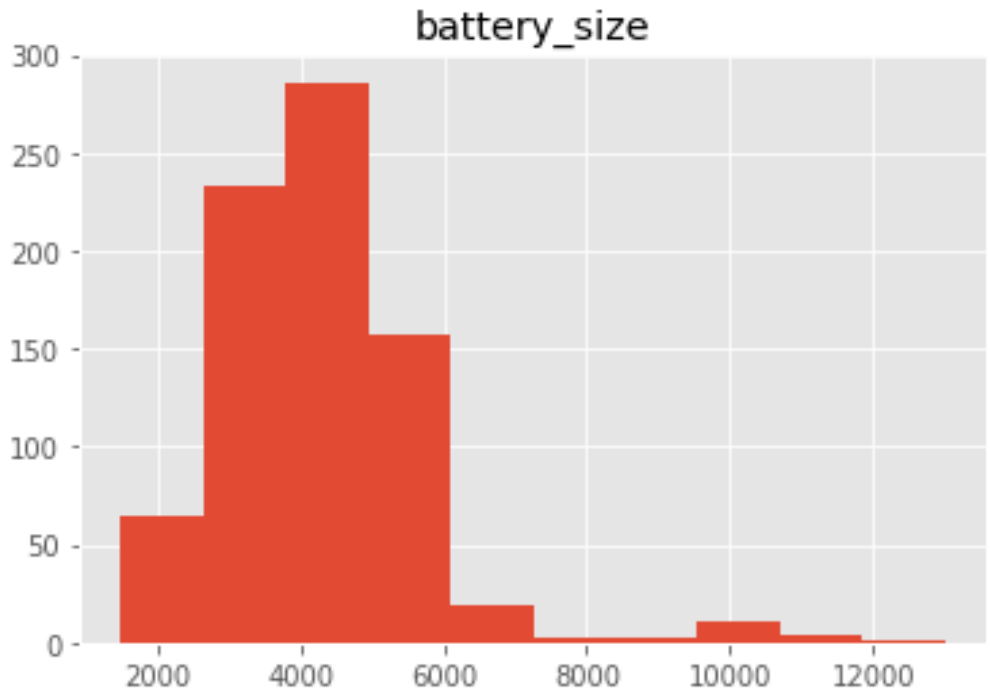

lowest_price
400
350
300
250
200
150
100
50
0
0
10000
20000
30000
40000


highest_price
400
300
200
100
0
0
10000
20000
30000
40000
50000
60000


sellers_amount
400
300
200
100
0
0
20
40
60
80
100
120

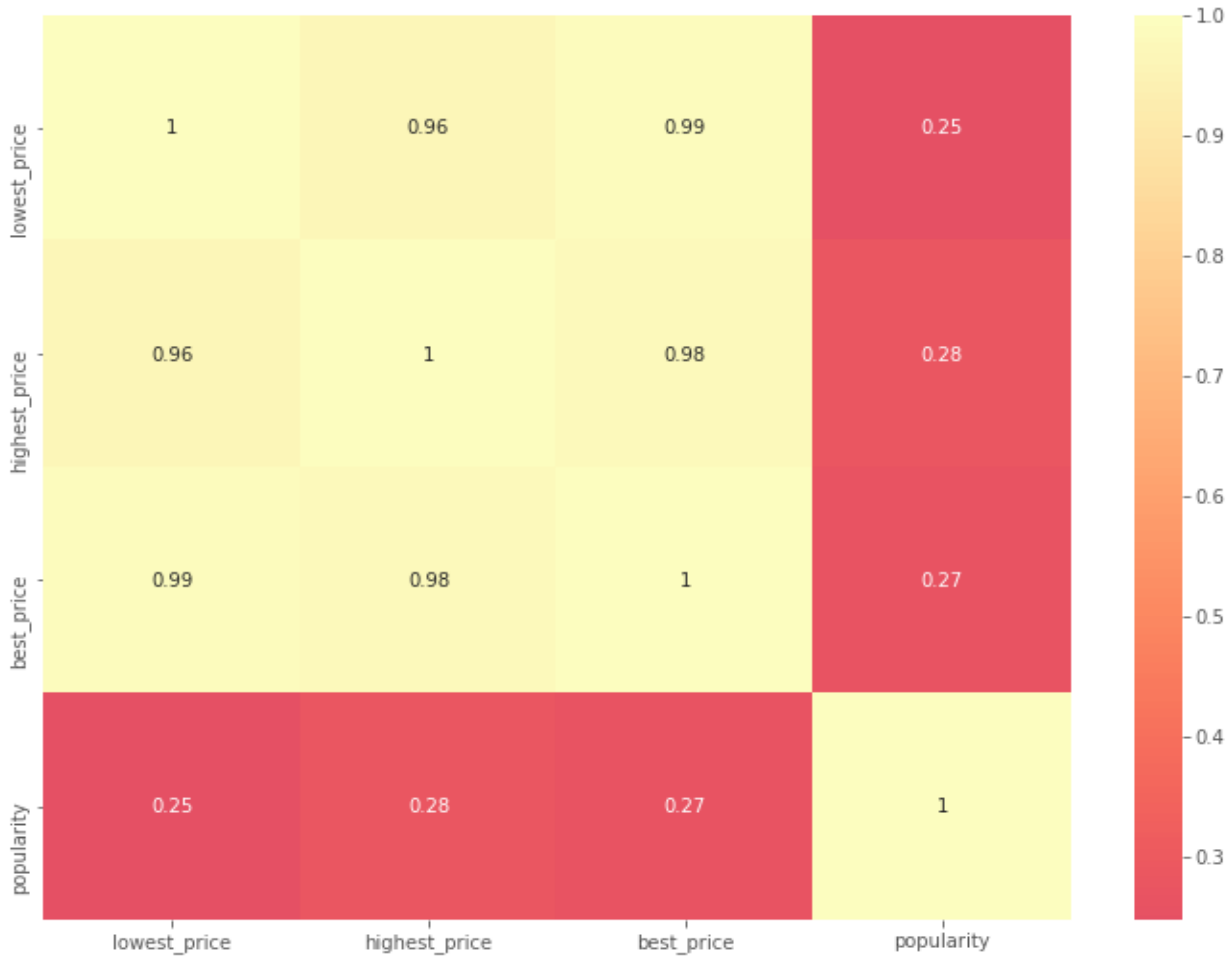

screen_size
300
250
200
150
100
50
0
3
4
5
6
7
8


memory_size
400
300
200
100
0
0
200
400
600
800
1000



Проведемо кореляційний аналіз. Скориставшись вбудованою в pandas функцією corr(), побудуємо кореляційну матрицю числових ознак датасету:

```
dumm =
pandas.get_dummies(phones_new[['lowest_price','highest_price','best
_price','popularity']])
corr = pandas.DataFrame(dumm).corr()
fig, ax = plt.subplots(1, 1, figsize=(12, 9))
sns.heatmap(corr, cmap='magma', center=0, annot=True, ax=ax)
plt.show()
```



З кореляційної матриці бачимо, що взаємозв'язок між багатьма ознаками є дуже сильний. Найнижчу кореляцію мають popularity і lowest_price(популярність та найнижча ціна). Найкраща ціна та популярність сильно корельовані між собою.

Розіб'ємо dataset на набори даних: тренувальний, валідаційний та тестовий набори. Скористаємося для цього функцією train_test_split().

```
del phones_new['release_date']
X = phones_new[phones_new.columns[:-1]]
y = phones_new[phones_new.columns[-1]]
X_train, X_test, y_train, y_test = train_test_split(X, y)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train)

print(f'Train: X = {X_train.head()}, y = {y_train.head()}')
```

```
print(f'Test: X = {X_test.head()}, y = {y_test.head()}')
print(f'Validation: X = {X_val.head()}, y = {y_val.head()}')
```

```
Train: X =     brand_name  ... memory_size
816     Xiaomi  ...        32.0
681   Motorola  ...       128.0
197    Ulefone  ...       128.0
162       OPPO  ...        64.0
849       OPPO  ...       256.0

[5 rows x 10 columns], y = 816    4100.0
681    5000.0
197    6600.0
162    5000.0
849    4000.0
Name: battery_size, dtype: float64
Test: X =         brand_name  ... memory_size
63           Nokia  ...         4.0
1172         Apple  ...       128.0
1064          vivo  ...        64.0
792         Xiaomi  ...        32.0
997   Sigma mobile  ...        32.0

[5 rows x 10 columns], y = 63      1500.0
1172    1715.0
1064    5000.0
792     5020.0
997     6500.0
Name: battery_size, dtype: float64
Validation: X =      brand_name  ... memory_size
875      Lenovo  ...        64.0
410     Samsung  ...       128.0
516     Samsung  ...       256.0
1067     HUAWEI  ...        16.0
326          2E  ...         8.0

[5 rows x 10 columns], y = 875     3500.0
410     4000.0
516     5000.0
1067    3020.0
326     2500.0
Name: battery_size, dtype: float64
```

Застосуємо описову статистику: середні арифметичні, стандартні відхилення, мінімальне та максимальне значення кожної характеристики, скориставшись функцією describe():

phones_new.describe()

	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size	battery_size
count	780.000000	780.000000	780.000000	780.000000	780.000000	780.000000	780.000000	780.000000
mean	729.238462	10203.064103	9214.326923	11751.229487	20.226923	6.060462	112.287179	4121.117949
std	338.209052	9612.168455	8652.375633	11693.361223	22.330371	0.626583	117.754426	1406.849509
min	2.000000	1036.000000	899.000000	1059.000000	2.000000	2.400000	4.000000	1500.000000
25%	475.500000	3759.000000	3499.000000	3999.000000	4.000000	5.700000	32.000000	3174.000000
50%	783.500000	5931.000000	5547.000000	6512.500000	11.000000	6.215000	64.000000	4000.000000
75%	1017.250000	13904.250000	12425.500000	16057.750000	29.000000	6.500000	128.000000	5000.000000
max	1224.000000	55338.000000	45799.000000	64999.000000	125.000000	8.100000	1000.000000	13000.000000

Додатково знайдемо дисперсію кожної характеристики. Для цього скористаємося вбудованою в бібілотеку pandas функцією var().

phones_new.var()

```
popularity         1.143854e+05
best_price         9.239378e+07
lowest_price       7.486360e+07
highest_price      1.367347e+08
sellers_amount     4.986455e+02
screen_size        3.926059e-01
memory_size        1.386610e+04
battery_size       1.979226e+06
dtype: float64
```

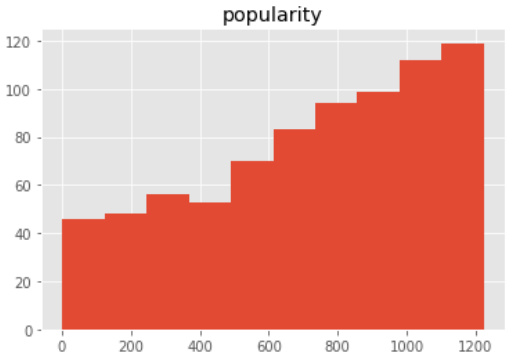
Знайдемо медіану, скориставшись вбудованою в бібілотеку pandas функцією median().

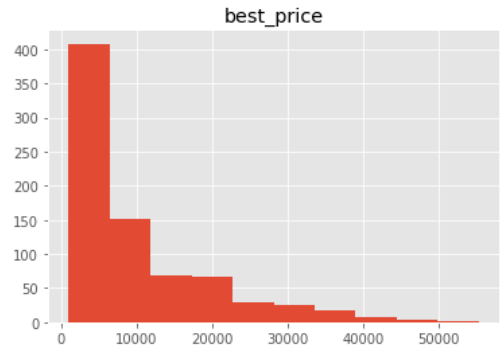
phones_new.median()

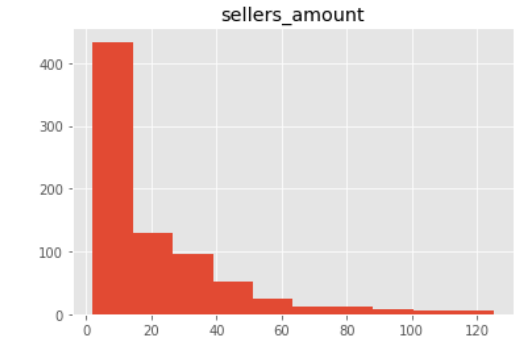
```
popularity         783.500
best_price        5931.000
lowest_price      5547.000
highest_price     6512.500
sellers_amount      11.000
screen_size          6.215
memory_size         64.000
battery_size      4000.000
dtype: float64
```

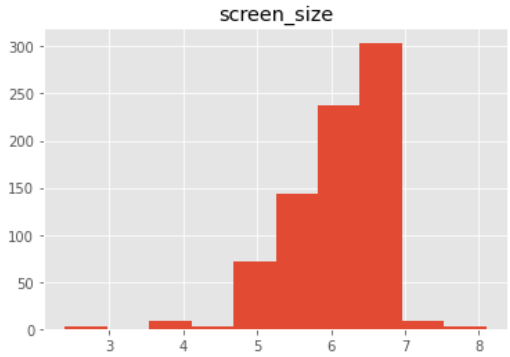
Побудуємо гістограми розподілів числових ознак датасету. Для побудови графіків скористаємося біліотекою matplotlib:

```
phones_new_numeric = phones_new.select_dtypes(include=[np.number])
numeric = phones_new_numeric.columns.values
for i in numeric:
    plt.hist(phones_new[i])
    plt.title(i)
    plt.show()
```









Проведемо кореляційний аналіз. Скориставшись вбудованою в pandas функцією corr(), побудуємо кореляційну матрицю числових ознак датасету:

З кореляційної матриці бачимо, що взаємозв'язок між багатьма ознаками є дуже сильний. Найнижчу кореляцію мають popularity і lowest_price(популярність та найнижча ціна). Найкраща ціна та популярність сильно корельовані між собою.

Формалізація цільової функції оптимізації. Визначення метрик оцінки ефективності моделі

Нехай X - матриця, стовпчики якої - це значення незалежних змінних, а рядки - це окремі мобільні пристрої.

Цільова функція оптимізації : $loss(B) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - X_i B)^2 + \alpha \sum_{k=1}^{K} |B_k|$.

Метрика оцінки ефективності моделі -- коефіцієнт детермінації R^2. Він показує, яка частка загальної дисперсії даних пояснюється регресією.

Формула: $R^2 = 1 - \frac{\text{сума квадратів нев'язок}}{\text{загальна сума квадратів}}$

Створення моделі

Для задачі лассо-регресії зручно використати вбудований модуль sklearn (scikit-learn). Це бібліотека Python, яка містить засоби для різноманітних задач машинного навчання: класифікації, регресії, кластеризації, зменшення розмірності простору (dimensionality reduction), вибору моделей тощо. За проведення лассо-регресії відповідає функція sklearn.linear_model.Lasso.

```
lasso = linear_model.Lasso(alpha=1, normalize=True)
```

За навчання моделі в бібліотеці sklearn відповідає функція fit. sklearn самостійно проведе регресію.

```
lasso.fit(X_train, y_train)
coefs = lasso.coef_
intercept = lasso.intercept_
print("b_0 =", intercept)
pd.DataFrame(data=np.expand_dims(coefs, 0),
columns=X_train.columns)
```

```
b_0 = -1805.684390313682
```

	popularity	best_price	lowest_price	highest_price	sellers_amount	screen_size	memory_size
0	0.0	-0.0	-0.0	-0.031688	0.433565	1020.020834	0.644057

Розглянемо вільний член b_0 . Якщо популярність мобільного пристрою, ціна, розмір екрану та інші характеристики дорівнюють нулю, то його рейтин буде становити -1805.684390313682. Щодо від'ємного рейтингу - то це одна з особливостей лінійної регресії. Лінійна функція (якщо це не константа) в певній точці досягає нуля і набуває від'ємних значень.

Підрахуємо коефіцієнт детермінації R, який чисельно показує, яка частина варіації залежної змінної пояснена моделлю:

```
r2_train = lasso.score(X_train, y_train)
r2_valid = lasso.score(X_val, y_val)
print(r2_train, r2_valid)
```

```
0.6318844061494144 0.5922100004904121
```

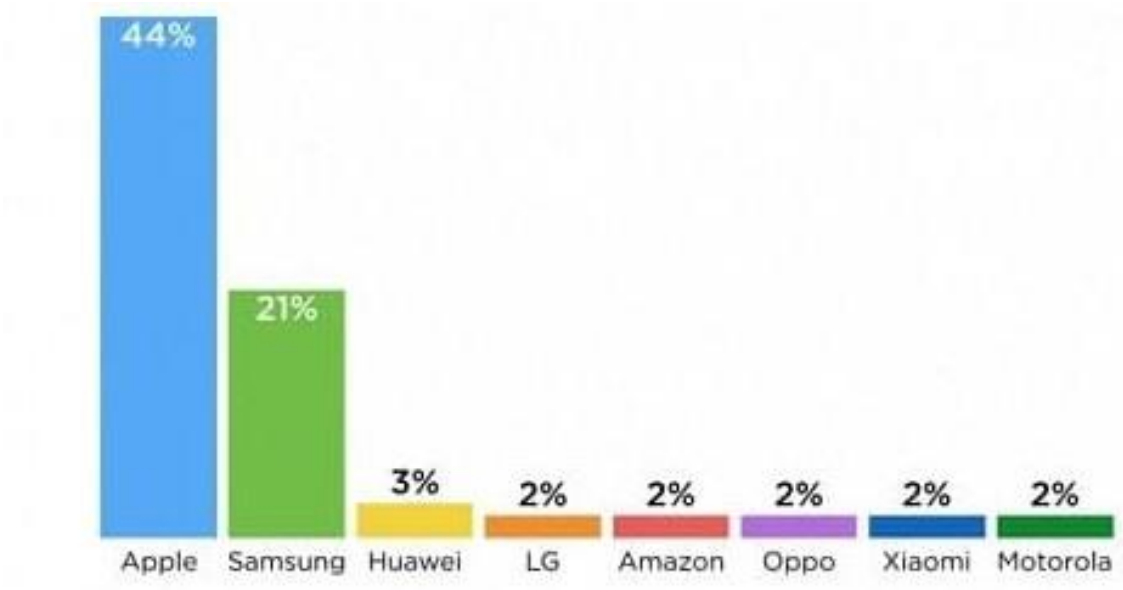
Для тренувального сету він дорівнює 0.632, а для валідаційного - 0.592. Тобто наша модель пояснює мінливість даних у датасеті в середньому на 62%.

Знайдемо для різних α модель та знайдемо коефіцієнт R^2. Оберемо ту модель, в якій цей коефіцієнт найбільший.

```
alpha_arr = [10000, 1000, 300, 200, 100, 30, 10, 3, 1, 0.3, 0.1,
0.03, 0.01]
models = []
for alpha in alpha_arr:
  lasso = linear_model.Lasso(alpha=alpha, normalize=True)
  lasso.fit(X_train, y_train)
  models.append(lasso)
  r2 = lasso.score(X_val, y_val)
  print(f"alpha = {alpha}, R^2 = {r2}")
```

```
alpha = 10000, R^2 = 0.498364120058316
alpha = 1000, R^2 = 0.498364120058316
alpha = 300, R^2 = 0.498364120058316
alpha = 200, R^2 = 0.498364120058316
alpha = 100, R^2 = 0.498364120058316
alpha = 30, R^2 = 0.498364120058316
alpha = 10, R^2 = 0.649596663479415
alpha = 3, R^2 = 0.6943734254890065
alpha = 1, R^2 = 0.692210000490412
alpha = 0.3, R^2 = 0.6870610538963667
alpha = 0.1, R^2 = 0.6860871338727959
alpha = 0.03, R^2 = 0.6850111041705766
alpha = 0.01, R^2 = 0.6845582951247622
```

Найвищу точність отримали при alpha=3.




```
results = pd.DataFrame({'y_true': np.squeeze(Y_test),
'y_pred':Y_pred})
results.head(10)
```

y_true	y_pred
287	295.699476
71	65.487055
421	400.560198
134	128.699476
477	499.614457
215	240.752694
179	176.946943
10	9.049274
262	255.877957

Бачимо, що рейтинг прогнозований нашою моделлю (друга колонка) трохи відрізняється від реального значення популярності мобільного телефону. Якщо популярність телефону складає 287, то модель передбачила 296. Можна

помітити закономірність, що чим менша популярність телефону, тим точніше наша модель змогла її спрогнозувати.

Інтерпретація отриманих результатів

В контексті задачі моделювання популярності мобільних пристроїв був обраний метод лассо-регресія. Цей алгоритм легко інтерпретувати (тобто зрозуміти, що саме намагається зробити цей алгоритм). Було виявлено які змінні майже не впливають на рейтинг телефонів, наприклад найнижча ціна.

Серед плюсів цього методу можна відмітити вирішення проблем корельованих вхідних даних.

Для підвищення точності моделі можна зібрати дані про більшу кількість мобільних пристроїв і додати нові характеристики про телефон, такі як інформація про процесор та камеру. Ці характеристики можуть позитивно вплинути на рейтинг мобільного пристрою.

Мета вирішення задачі моделювання: в даній роботі було обрано задачу регресії. Було взято датасет "Mobile Phones Data". Задачею моделювання в рамках конкретного контексту є моделювання системи для передбачення популярності мобільного пристрою.

Процеси/функції вирішення проблеми: процеси і алгоритми статистичного аналізу, які було застосовано: описова статистика (знаходження середніх арифметичних, стандартних відхилень, мінімального та максимального значень, дисперсії, моди та медіани кожної ознаки, а також побудова гістограм розподілу кожної ознаки), кореляційний аналіз (побудовано кореляційну матрицю та проаналізовано степінь взаємозв'язку).

Ресурси для реалізації і вирішення задачі:

інструменти та мета їх застосування: https://www.kaggle.com/ (джерело даних; збережено dataset), https://pandas.pydata.org/docs/ (бібліотека, за допомогою якої було подано дані у вигляді dataframe для зручності в роботі; було використано її

вбудовані функції для реалізації описової статистики та кореляційного аналізу), https://matplotlib.org/ (бібліотека, за допомогою якої виконувався побудова графіків, а саме гістограм розподілу при реалізації описової статистики).

ВИСНОВКИ

Перед початком реалізації моделювання системи для прогнозування рейтингу мобільних пристроїв було виконано:

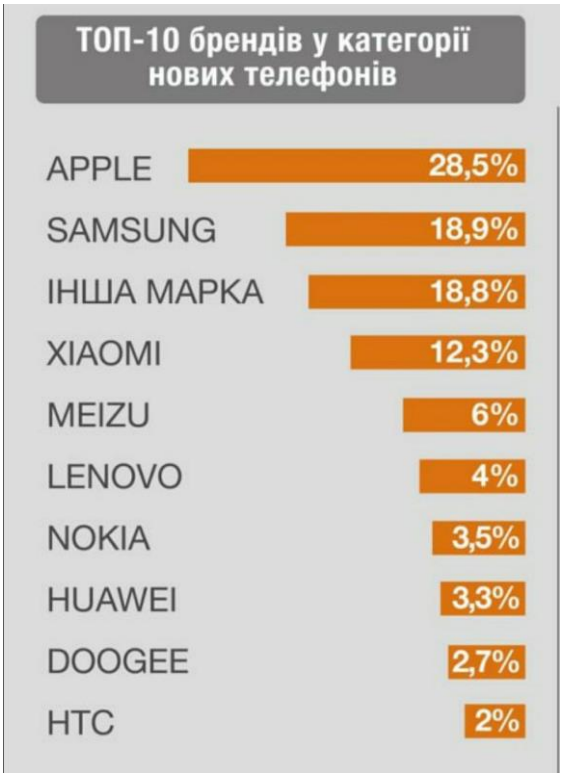
- первинну очистку сирих даних (видалено стовпець unnamed, усунення NaN/Null значень, перевірено, що записи-дублікати в датасеті відсутні).
- проведено розвідувальний аналіз (побудована кореляційна матриця, з якої було з'ясовано, що найнижчу кореляцію мають popularity і lowest_price)
- дослідження датасету із застосуванням описової статистики
- розбиття очищених та підготовлених даних на два табори (тренувальний та тестовий).

Це нам дало базу для виконання подальшого моделювання задачі, що була запланована на початку даної роботи.

Основна ціль моделі - прогнозування рейтингу мобільних телефонів, які можна купити в Україні, тобто, є задачею регресії. Було вибрано метод лассо-регресія та на його основі побудовано модель.

Оцінено точність моделі: на тестовому датасеті дана модель пояснює 62% усієї мінливості даних. Для побудови моделі ми використали бібліотеку для мови Python - sklearn.

Система прогнозування показала, що популярними телефонами, які продаються в Україні, є Apple та Samsung. Ця інформація відповідає дійсності.



Дана система може бути корисною для корпорацій-виробників мобільних телефонів, які бажають дізнатися чи буде користуватися попитом та чи інша модель телефону, а також для покупців, які зможуть орієнтуватися в рейтингу телефонів, що є одним з головних показників при покупці гаджетів.

Отже, поставлена задача виконана, а результат, отриманий по завершенню виконання задачі цілком влаштовує цілі, поставлені перед початком моделювання.

```
import pandas
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
import numpy as np
plt.style.use('ggplot')
from matplotlib.pyplot import figure
phones = 'drive/MyDrive/phones_data.csv'



phones_new = phones.drop(columns="Unnamed: 0")
phones_new


phones_new.isnull().sum()

phones_new.dropna(subset=['os','lowest_price', 'highest_price', 'screen_size',
'memory_size', 'battery_size'], inplace=True)
phones_new

phones_new.isnull().sum()

phones_new.describe()
```

```
phones_new.var()


phones_new.median()


phones_new_numeric = phones_new.select_dtypes(include=[np.number])

numeric = phones_new_numeric.columns.values

for i in numeric:

    plt.hist(phones_new[i])

    plt.title(i)

    plt.show()


dumm =
pandas.get_dummies(phones_new[['lowest_price','highest_price','best_price','
popularity']])
                                                                    In [22]:
corr = pandas.DataFrame(dumm).corr()
fig, ax = plt.subplots(1, 1, figsize=(12, 9))
sns.heatmap(corr, cmap='magma', center=0, annot=True, ax=ax)
plt.show()



lasso = linear_model.Lasso(alpha=1, normalize=True)


lasso.fit(X_train, y_train)

coefs = lasso.coef_

intercept = lasso.intercept_

print("b_0 =", intercept)

pd.DataFrame(data=np.expand_dims(coefs, 0), columns=X_train.columns)
```

```
alpha_arr = [10000, 1000, 300, 200, 100, 30, 10, 3, 1, 0.3, 0.1, 0.03, 0.01]

models = []

for alpha in alpha_arr:

  lasso = linear_model.Lasso(alpha=alpha, normalize=True)

  lasso.fit(X_train, y_train)

  models.append(lasso)

  r2 = lasso.score(X_val, y_val)

  print(f"alpha = {alpha}, R^2 = {r2}")

results = pd.DataFrame({'y_true': np.squeeze(Y_test), 'y_pred':Y_pred})

results.head(10)
```